УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У ЧАЧКУ

Драгана М. Кнежевић

РАЗВОЈ МОДЕЛА ЗА ПРОЦЕНУ МЕСЕЧНЕ ПОТРОШЊЕ ЕЛЕКТРИЧНЕ ЕНЕРГИЈЕ ЗАСНОВАНОГ НА ТЕХНИКАМА МАШИНСКОГ УЧЕЊА

докторска дисертација

Чачак, 2024. године

UNIVERSITY OF KRAGUJEVAC
FACULTY OF TECHNICAL SCIENCES ČAČAK

Dragana M. Knežević

DESIGNING MODEL FOR ESTIMATING MONTHLY ELECTRICITY CONSUMPTION USING MACHINE LEARNING TECHNIQUES

Doctoral Dissertation

Čačak, 2024

Аутор
Име и презиме: Драгана Кнежевић
Датум и место рођења: 28.03.1987. године, Бајина Башта
Садашње запослење: Асистент, Академија струковних студија Западна Србија, Одсек Ужице
Докторска дисертација
Наслов: РАЗВОЈ МОДЕЛА ЗА ПРОЦЕНУ МЕСЕЧНЕ ПОТРОШЊЕ ЕЛЕКТРИЧНЕ ЕНЕРГИЈЕ ЗАСНОВАНОГ НА ТЕХНИКАМА МАШИНСКОГ УЧЕЊА
Број страница: 137
Број слика: 103
Број библиографских података: 150
Установа и место где је рад израђен: Универзитет у Крагујевцу, Факултет техничких наука у Чачку
Научна област (УДК): Електротехничко и рачунарско инжењерство (редни број 2.5 у поменутом Правилнику), односно њене следеће уже научне области: - Информационе технологије и системи (редни број 2.5.4 у поменутом Правилнику) и - Примењено рачунарство (редни број 2.5.11 у поменутом Правилнику).
Ментор: др Марија Благојевић, редовни професор, Универзитет у Крагујевцу, Факултет техничких наука у Чачку
Број и датум одлуке Већа универзитета о прихватању теме докторске дисертације: Број: IV-04-683/11, датум: 19.09.2023.

ЗАХВАЛНИЦА

„Путовање од хиљаду миља почиње једним кораком.“

Кинеска пословица

Ова дисертација јесте моје велико дело али је настала захваљујући низу ситуација и људи који су били уз мене током рада на њој.

Посвећујем је својој ћерки Софији, чије одрастање је највише трпело током година мог усавршавања, на путу до циља.

А на овим последњим корацима пре тог циља користим прилику да изразим захвалност најпре Богу за сваки дан здравља и што сам у свим моментима успона и падова имајући праве људе око себе, успела да истрајем и постигнем жељено.

У тој мојој великој жељи неизмерно ми је помогла менторка, проф. др Благојевић Марија, редовни професор Факултета техничких наука у Чачку, Универзитета у Крагујевцу од које сам још од основних студија добијала ветар у леђа и охрабрење за будуће кораке. Огроман допринос мом раду дао је и проф. др Ранковић Александар, редовни професор Факултета техничких наука у Чачку, Универзитета у Крагујевцу. Својим саветима, стрпљењем и посвећеношћу, проф. Благојевић и проф. Ранковић, учинили су да ово путовање буде, у најмању руку, лепо искуство и да сваком новом изазову приступим са усхићењем. Кроз неколико година сарадње несебично су на мене преносили своје искуство и знање и учинили да се целокупног процеса сећам са радошћу и поносом, због тога велико ХВАЛА.

Захвалила бих и мојим колегама са Академије струковних студија Западна Србија, који су имали разумевања током мог читавог усавршавања и на различитије начине допринели мом успеху. Највише др Диковић Љубици, др Миливојевић Миловану и др Станишевић Иљи који су ми и стручним и педагошким саветима помагали све ово време. Посебно бих истакла моју драгу колегиницу Ивану Маринковић, која ми је увек давала подстрек и била спремна за разговор у свако доба. Велико хвала и Срђану Обрадовићу, колеги, другу, мојој подршци и помоћи у критичним тренутцима.

Неизмерну захвалност желим да изразим према мојој породици без чије подршке успех не би био могућ. Посебно желим да захвалим мојој мајци, која ни у једном тренутку није посумњала у моју жељу и моје способности. Заједено са мојом сестром, пуне разумевања и љубави пратиле су ме годинама охрабрујући када год је то било потребно и ћутећи када је то било још потребније.

Иако га помињем на крају, мој највећи ослонац и подршка у сваком погледу је мој супруг. Од првог корака па до самог циља он је успевао да ми помогне и разговором и тишином, питањима и одговорима. И зато највеће ХВАЛА.

Хвала и свим другима које нисам истакла а који су својим разумевањем и постојањем допринели да странице које следе буду исписане.

Захвална,

Драгана Кнежевић

Резиме

Имајући у виду да растућа светска популација и технолошки развој доводе до повећане потражње за електричном енергијом, развој ефикасног и поузданог система за планирање и процену потрошње електричне енергије постаје императив и једна од кључних тема.

Основни циљ и мотивација ове дисертације су унапређење тачности и поузданости процене потрошње електричне енергије, што може допринети ефикаснијем планирању и управљању електро-енергетским системом, а тиме и смањењу губитака енергије.

У складу са тим, представљен је програмски оквир за подршку предвиђању и планирању потрошње електричне енергије, темељен на техникама машинског учења.

Кроз неколико поглавља, представљено је више различитих приступа базираних на машинском учењу, од поступка адаптације иницијалног скупа података до развоја модела праћеног евалуацијом резултата. Предложени модел настаје као продукт синергије вишегодишњег истраживања, логичких правила и начела, теоријско-епистемиолошких сазнања, дуготрајног практичног рада на развоју програмског оквира и на крају тестирања на стварном проблему.

Имајући у виду разноврсност потрошача обухваћеним приликом тестирања модела, јасно је да се стечена сазнања и предложена методологија могу користити у домаћинствима, паметним зградама, па чак и читавим квартовима, градовима али и за подршку планирању потрошње у индустрији. Резултати постигнути на овај начин могу наћи примену у широком спектру апликација из домена дистрибуције и потрошње електричне енергије.

На тај начин, потврђена је претпоставка да се технике машинског учења ефикасно могу користити и у овом домену чиме је дат допринос развоју знања о примени техника машинског учења на пољу планирања производње и потрошње електричне енергије.

Кључне речи: Вештачке неуронске мреже, Градијентни буст, Линеарна регресија, Машинско учење, Предикција потрошње електричне енергије, Регресија са потпорним векторима, Случајне шуме, Стабла одлуке, Хубер

Abstract

The growing global population and rapid technological development have caused the increasing electricity demand and therefore made the development of an efficient and reliable system for electricity consumption planning and estimating a must, and one of the burning issues.

This doctoral dissertation aims to improve the accuracy and reliability of electricity consumption estimation, which may increase the consumption planning efficiency, and improve the energy systems management, thus reducing energy losses as well. In line with that, a programme framework which supports the electrical energy consumption estimation and planning, and which is based on machine learning techniques is presented in the dissertation.

Several different approaches based on machine learning, ranging from the adaptation of the initial dataset to the development of the model and its evaluation, are described in several chapters. The proposed model emerged as the product of the synergy between the research which took years to conduct, logical norms and principles, theory and epistemology, long-lasting practical work, and finally the model's performance testing in the real world.

Taking into consideration the diversity of the consumers included in the model testing, it can be concluded that the acquired knowledge and proposed methodology can be used in households, smart buildings, entire neighbourhoods and cities, as well as to plan the industrial electricity consumption. The results obtained in this way can be used in a wide range of applications in the field of electrical energy distribution and consumption.

Therefore, the hypothesis on the efficiency of machine learning techniques in this domain has been confirmed, thus making a contribution to the development of knowledge on the use of machine learning techniques for the electricity distribution and consumption planning.

Key words: Artificial Neural Networks, Decision tree, Electrical Energy Consumption Prediction, Gradient Boosting, Huber, Linear Regression, Linear SVR, Machine learning, Random Forest

Ознаке и скраћенице:

Acc ..... Тачност, прецизност (енгл. Accuracy)

Adadelta ..... Алгоритам оптимизације (енгл. Adaptive Delta)

Adagard ..... Алгоритам оптимизације (енгл. Adaptive Gradient)

Adam ..... Алгоритам оптимизације (енгл. Adaptive Moment Estimation)

AI..... Вештачка интелигенција (енгл. Artificial intelligence)

AR^2 ..... Прилагођени коефицијент детерминације (енгл. Adjusted R Square)

ANN ..... Вештачка неуронска мрежа (енгл. Artificial Neural Network)

AUC ..... Површина испод ROC криве (енгл. Area Under the Curve)

BN ..... Нормализација на нивоу серије (енгл. Batch Normalization)

CNN ..... Конволутивне вештачке неуронске мреже (енгл. Convolutional Neural Networks)

FN..... Лажно негативно (енгл. False Negative)

FP ..... Лажно позитивно (енгл. False Positive)

FPR..... Лажно позитивна стопа (енгл. False Positive Rate)

GNN ..... Графовске вештачке неуронске мреже (енгл. Graph Neural Networks)

IQR ..... Интерквартилни опсег (енгл. Interquartile Range)

L1 ..... Тип регуларизације (енгл. Lasso)

L12 ..... Тип регуларизације (енгл. Elastic Net)

L2 ..... Тип регуларизације (енгл. Ridge)

Linear SVR.... Линеарна регресија са потпорним векторима (енгл. Linear Support Vector Regression)

LNL ..... Нормализација на нивоу слоја (енгл. Layer Normalization Layer)

LSTM ..... Вештачке неуронске мреже са дуготрајном меморијом (енгл. Long Short-Term Memory Networks)

MAE ..... Средња апсолутна грешка (енгл. Mean Absolute Error)

MAPE..... Средња апсолутна процентуална грешка (енгл. Mean Absolute Percentage Error)

ML ..... Машинско учење (енгл. Machine Learning)

MLP..... Вишеслојни перцептрон (енгл. Multilayer Perceptron)

MSE..... Средња квадратна грешка (енгл. Mean Square Error)

NN ..... Неуронска мрежа (енгл. Neural Network)

R^2 ..... Коефицијент детерминације (енгл. Coefficient of Determination)

ReLU ..... Ректификована, исправљена функција оптимизације (енгл. Rectifed Linear Units)

RMSE ..... Корен средње квадратне грешке (енгл. Root Mean Square Error)

RMSprop ..... Алгоритам оптимизације (енгл. Root Mean Square Propagation)

RNN ..... Рекурентне вештачке неуронске мреже (енгл. Recurrent Neural Networks)

ROC..... ROC крива (енгл. Receiver Operating Characteristic Curve).

RvNN ..... Рекурзивне вештачке неуронске мреже (енгл. Recursive Neural Networks)

TN..... Стварно негативно (енгл. True Negative)

TNR ..... Опозив негативне класе (енгл. True Negative Rate)

TP ..... Стварно позитивно(енгл. True Positive)

TPR..... Стварна, права позитивна стопа, Опозив позитивне класе (енгл. True Positive Rate)

WNL..... Слој са нормализацијом тежина (енгл. Weight Normalization Layers)

XGBoost..... Екстремни градијентни буст (енгл. Extreme Gradient Boosting)

САДРЖАЈ:

Попис слика:

Попис табела:

1. УВОДНА РАЗМАТРАЊА

Све је евидентнији вртоглавог развој како технике и науке тако и друштва у глобалном смислу. Обзиром да растућа светска популација уз пратећи технолошки развој доприноси повећаној потражњи за електричном енергијом, развој ефикасног и поузданог система за подршку планирању потрошње електричне енергије постаје једна од кључних тема данашњице. Равнотежу између увећане потрошње и потребе за уштедом електричне енергије могуће је успоставити правилном проценом потрошње у реалним системима.

Може се рећи да прогноза потрошње електричне енергије непосредно утиче на поузданост у снабдевању целокупног система што посредно доприноси стабилности једне земље у сваком а посебно у економском смислу.

У овој дисертацији биће представљен програмски оквир за подршку предвиђању потрошње али и планирању производње електричне енергије, темељен на техникама машинског учења.

Вештачке неуронске мреже, као један од најчешће коришћених алгоритама машинског учења, са великим успехом врше предвиђање и анализу података у различитим областима. Развијени програмски оквир за предвиђање и планирање потрошње електричне енергије заснован је управо на неуронским мрежама. У овом контексту модел неуронских мрежа треба да буде у стању да узме у обзир најразличитије факторе који утичу на потрошњу електричне енергије, претвори их у корисно знање и као резултат тога пружи висок степен прецизности и поузданости у предвиђању потрошњи за сваког потрошача појединачно.

Претходна истраживања су се углавном бавила предвиђањем потрошње електричне енергије на краткорочном нивоу, процењујући потрошње на неком ограниченом, мањем простору и то за наредни сат или дан.

Предвиђање потрошње и планирање производње електричне енергије су најчешће симултани процеси и одвијају се у оквиру једне организационе јединице предузећа које се бави производњом и дистрибуцијом исте. Међутим то су ипак два одвојена процеса и међу њима постоји значајна разлика. Планирање подразумева низ оперативних стратегија које зависе од предвиђања, док је предвиђање процена резултата до којих се долази реализацијом неког плана [1].

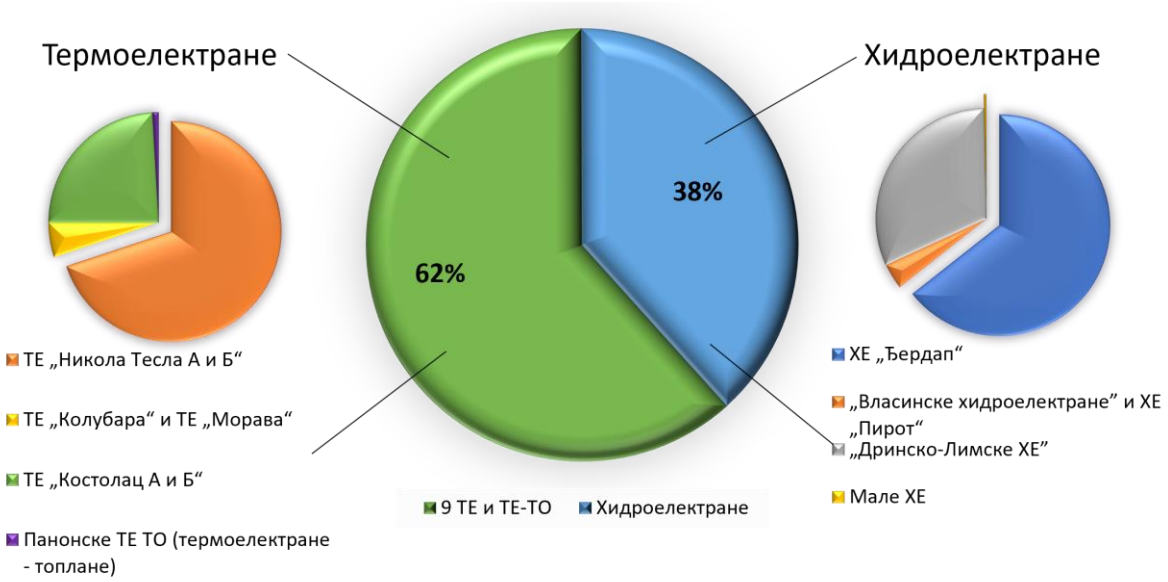
Тема ове дисертације јесте процена месечне потрошње електричне енергије и то на нивоу једног већег подручја, тачније, на подручју једног града. Како се потрошње обрачунавају и наплаћују на месечном нивоу, постоји и оправданост у процени потрошње на истом том нивоу.

1.1. Мотивација и предмет дисертације

Са убрзаним развојем друштва и појавом иновативних, паметних уређаја у последњих неколико деценија, драматично расте и потрошња електричне енергије. Упоредо са растом потрошње расту и макроекономски трошкови чинећи додатни притисак како на производне, тако и на дистрибутивне компаније. Решење ових проблема би се могло постићи развојем нових метода за процену потрошње електричне енергије а тиме и процене њеног оптималног нивоа што чини темеље мотивације ове дисертације.

Као једно од могућих решења, истражује се примена техника машинског учења у циљу процене месечних потрошњи електричне енергије, што би даље омогућило њено праћење и предвиђање. Циљ је успоставити такав систем који ће омогућити предвиђање за одређен наредни период не само за постојеће већ и за сваког новог потрошача.

Производња електричне енергије је сталан, континуалан процес који ангажује разноврсне ресурсе. У Србији постоје два основна начина производње електричне енергије: у хидроцентралама и термоелектранама. Иако хидроцентрале користе обновљиве извори енергије, још увек велики удео у производњи електричне енергије имају необновљиви извори. У протеклих десет година, просечна годишња производња електричне енергије у Србији износила је готово 35.000 GWh електричне енергије. Производни капацитети којима управља предузеће „Електропривреда Србије“ (ЕПС) су укупне снаге 7.855 MW, од тога, у термоелектранама произведе се око 62% електричне енергије, док се око 38% добија из 16 хидроелектрана (Слика 1-1), [2]. Све то појачава потребу за решавањем проблема планирања производње и потрошње електричне енергије.



Слика 1-1: Удео хидро и термоелектрана у производњи електричне енергије

Како наше поднебље има јасно изражена годишња доба, очекивано је да потрошња електричне енергије прати те сезонске промене и да се у складу са тим може и планирати производња. Производња константно великих количина електричне енергије и производња количина које су у дисбалансу са потребама, доводи до њеног “расипања” услед немогућности да се иста складишти или извезе. Са друге стране, производња премалих количина би резултирала потребом за прекомерним увозом електричне енергије и у крајњем случају условила рестрикције у снабдевању крајњих потрошача.

Успостављање равнотеже између производње и потрошње представља управо један од најтежих задатака и подстиче потребу за планском производњом. Како би планирање производње електричне енергије било изводљиво, неопходно је успоставити адекватан систем процене њене потрошње.

Поред проблема проистеклих из доступних извора енергије, један од стално присутних проблема јесте и појава губитака електричне енергије на шта утиче више различитих фактора и неефикасности у целокупном ланцу производње, дистрибуције али и потрошње електричне енергије. У том смислу се препознају две основне врсте губитака: технички и нетехнички губици.

Технички губици настају у свим елементима дистрибутивне мреже док се нетехнички губици јављају услед несавесне потрошње електричне енергије од стране потрошача (купаца електричне енергије). Како на ову врсту губитака утиче дугачак низ фактора из целокупне дистрибутивне мреже, даље се може разликовати вршити подела по различитим критеријумима. Основна подела техничких губитака би била на сталне и оне који се јављају повремено. Стални губици постоје независно од количине потрошње електричне енергије, односно оптерећења дистрибутивне мреже док се повремени јављају само ако (када) постоји потрошња електричне енергије, односно оптерећење дистрибутивне мреже [3].

На страни потрошача долази до губитака најчешће услед нерационалне потрошње, тј. непотребног укључивања уређаја или расвете. Ово се односи и на уређаје док су у стању мировања када и даље троше малу али не и занемарљиву количину енергије. Још један од узрочника непланиране и непотребне потрошње на страни потрошача јесте и лоша термална изолација у домаћинствима, посебно оним која електричну енергију користе за грејање у зимским и хлађење у летњим месецима. Поред домаћинстава, значајне потрошаче представљају и потрошачи из категорије комерцијалних потрошача који поседују електричне уређаје чији мотори могу генерисати реактивну енергију која не доприноси корисном раду уређаја већ оптерећује електричну мрежу. Таква потрошена енергије додатно отежава одржавање стабилности у редовном снабдевању.

Стога је за правилно и несметано снабдевање електричном енергијом од кључног значаја развој поузданог система за предикцију њене потрошње [4].

Употребом техника машинског учења у процесу процене потрошње електричне енергије на територији једног града има важну улогу и бенефите који се могу повољно одразити на енергетски сектор читаве земље. Развој прецизних и поузданих модела у циљу процене месечне потрошње електричне енергије је од изузетне важности за планирање и управљање енергетским ресурсима како на локалном тако и на глобалном нивоу. Оправданост процене потрошњи електричне енергије на месечном нивоу проистиче из чињенице да се и обрачунавање и наплаћивање врши на истом нивоу.

Са развојем друштва расте и потрошња електричне енергије па је изузетно тешко предвидети будуће потрошње. Један од највећих изазова и проблема у том процесу је проналажење најутицајнијих карактеристика потрошача али и окружења у коме се потрошач налази. Други проблем који се јавља већ у фази прикупљања података, јесте сама структура података. Подаци који чине базу података дистрибутивних компанија често као такви нису довољни за развој поузданог предиктивног модела. У оквиру истраживања чији су резултати приказани у дисертацији, дат је предлог адаптације постојећег скупа обележја, још у фази припреме података. Поступак адаптације подразумева да се постојећим карактеристикама потрошача придруже обележја прикупљена из других, екстерних извора.

У складу с тим, дисертација се бави развојем модела заснованог на неуронским мрежама са циљем предвиђања месечне потрошње електричне енергије потрошача на територији града, темељеног на скупу обележја добијених интеграцијом постојећих и додатно прикупљених обележја потрошача и околине. Предложени модел може бити користан

алат за директну процену потрошње електричне енергије у градској средини, што би индиректно омогућило боље управљање потрошњом и бољу искоришћеност ресурса на посматраном подручју.

Интеграција обележја потрошача и околине би требало да омогући тачније предвиђање потрошње, с обзиром на очекивања да многи фактори, као што су временски услови, доба године, тип потрошача, имају велики утицај на потрошњу електричне енергије.

1.2. Преглед сродних истраживања

Поред тога што убрзани техничко-технолошки развој на глобалном нивоу последњих деценија условљава све веће потрошње електричне енергије истовремено мотивише све већи број аутора да се баве њеном проценом како на макро тако и на микро нивоу.

Постојећа истраживања приступају проблему предвиђања потрошње електричне енергије са неколико становишта фокусирајући се на два основна елемента: простор на коме се врши предвиђање и временски период на који се предвиђање односи. Највећи број њих се односи на предвиђање у краћем временском периоду: за наредни дан, сат или чак минут. Резултати постигнути на тај начин, засновани су на посматрању једног сегмента потрошача, најчешће на нивоу једне зграде (пословног центра, јавне установе или стамбене зграде) не морају бити одраз стварних потрошњи у дужем периоду као ни карактеристика разноврсних потрошача на ширем простору.

Импозантан број истраживања који се бави овом темом датира из периода између 2006. и 2018. године када се драматично повећао број истраживања базираних на употреби различитих техника машинског учења [5], [6]. То потврђују и аутори у [7] истичући значај предвиђања потрошње електричне енергије уз употребу нових обновљивих извора енергије. Аутори у [8], са освртом на исту деценију, истичу све већу појаву нових техника машинског учења у циљу решавања широког спектра проблема у области енергетских система наводећи као посебно значајну истовремену употребу више различитих техника машинског учења и (тада) нових хибридних модела.

Било да се проблем потрошње електричне енергије посматра са економске стране као у [9], [10] и [11] или се техникама вештачке интелигенције покушава доћи до решења проблема у области енергетске ефикасности и обновљивих извора енергије [12] или унапређења самог процеса потрошње [13] ова тема све више привлачи пажњу истраживача.

Генерално, узимајући у обзир доступну литературу, издваја се неколико основних фактора који утичу на начин предвиђања: просторни (или географски) оквири на којима се врши предвиђање и временски интервал на који се предвиђање односи. Такође, неретко су аутори фокусирани на предвиђање потрошњи једне одређене категорије потрошача, на пример: домаћинстава, као у [13] и [14], индустријских потрошача у [15] али и само једне врсте електричних уређаја у улози потрошача, као у [16].

Зависно од простора и обима узорка за који се врши предвиђање, издваја се мноштво аутора који користе технике машинског учења за предикцију потрошњи на простору читавог региона [17], једне државе [18], [19], [20], града [21], [22], [23], [24], [25], за подручје једне трафо-станице [26], [27], тржног центра [28], јавне установе [29], пословне [30] или стамбене зграде [31] или универзитетског насеља [32].

Зависно од временског основа, аутори се баве краткорочним предвиђањем потрошњи, од неколико наредних минута до неколико сати или дана [14], [24], [33], [34], [35], [36], [37],

[38], [39], [40], [41], [42], [43], [44], [45], [46], за дужи временски период (квартал, годину или низ година) као у [18], [19], [20], [28], [47], [48], [49], [50] или пак на месечном нивоу [22], [31], [51], [52], [53], [54], [55].

Да технике машинског учења, посебно вештачке неуронске мреже, имају успеха и при решавању проблема из домена класификације и из домена предвиђања, показала су истраживања [21], [56] и [22]. У [21] аутори врше класификацију потрошача електричне енергије, док је у [22] циљ предвиђање износа потрошње електричне енергије (у kWh), за подручје града пратећи њихове потрошње током неколико година. Ова истраживања потврђују да се употребом вештачких неуронских мрежа, потрошачи могу веома успешно класификовати у одговарајуће потрошачке групе али и предвидети њихове потрошње.

Касније, у [54], исти аутори долазе до значајних закључака и истичу да је могуће постићи још боље резултате предвиђања потрошњи претходним кластеровањем потрошача по неком од четири предложена критеријума. Аутори предлажу да се у скупу података издвоје хомогеније групе над којим ће се вршити предвиђање месечних потрошњи помоћу вештачких неуронских мрежа заснованих на слојевима са нормализацијама тежина и слојевима са нормализацијом активације претходног слоја (објашњеним и у [57]). Аутори су показали да ови слојеви, имплементирани у модел за предвиђање потрошњи електричне енергије, дају значајан допринос његовој успешности иако је њихова употреба раније везана за другачије врсте проблема (на пример, процесирање слика као у [58], [59]).

Као и у [17], [21], [22], [53], [54], [60], [61], [62] у овој дисертацији потрошње електричне енергије се бележе на основу очитавања бројила изражене у kWh.

Узевши у обзир преглед доступне литературе и доступних објављених истраживања, јасно је да решавање проблема предвиђања месечних потрошњи на једном ширем подручју представља посебан изазов. Решавање овог проблема захтева сагледавање великог броја чинилаца истовремено. Управо због таквих карактеристика и оригиналности проблема, предмет дисертације јесте проналажење решења заснованог на техникама машинског учења са посебним освртом на неуронске мреже.

1.3. Циљеви истраживања

Из свега претходно наведеног увиђа се потреба за решавањем два проблема: утврђивање карактеристика које утичу на потрошњу електричне енергије и успостављање система за њено предвиђање. Стога, сврха истраживања се може посматрати кроз неколико појединачних циљева проистеклих из једног превасходног циља: унапредити процес планирања производње и потрошње електричне енергије разноврсних потрошача на једном ширем подручју, развојем прецизног и поузданог модела.

Други циљеви дисертације надовезују се на успешност остварења основног циља:

- развој алгоритма за подршку предвиђању месечних потрошњи електричне енергије, погодног за примену при планирању и управљању енергетским ресурсима како на једном ужем тако и на ширем простору;
- одабир скупа најутицајнијих обележја потрошача и средине у којој живи, издвајајући оне који доприносе најпрецизнијим предвиђањима месечне потрошње електричне енергије;

- развој архитектуре и параметрирање модела заснованог на машинском учењу;
- утврђивање успешности и поузданости развијеног модела кроз одговарајуће мере прецизности.

У складу са циљевима, биће истражени потенцијали техника машинског учења за предвиђање потрошњи електричне енергије. У наредном поглављу биће представљена методологија а у поглављу 4 резултати истраживања и одабир одговарајућег модела за процену потрошње електричне енергије заснован на техникама машинског учења.

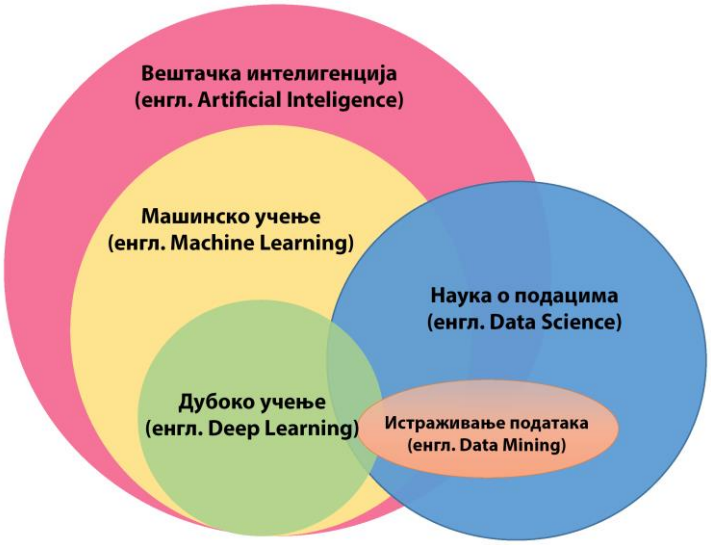
2. ТЕОРИЈСКИ ОКВИР

Када се говори о предвиђању било које појаве најчешће се као решење јавља нека од техника машинског учења. Појам машинског учења (енгл. *Machine learning,* скраћено *ML*) све више је присутан последњих деценија, а последњих година његове могућности, бар теоријски, су ненадмашне. Посебна примена ML-а се среће при решавању специфичних задатака који превазилазе људске могућности. С тим у вези, највећи значај ML је у томе што се његовом применом могу решити проблеми које човек чак и не мора знати да дефинише и у којима је прихватљива повремена грешка. Тако ML проналази своју примену у на пример: препознавању лица и различитих објеката на фотографијама и видео снимцима, препознавање тумора на медицинским снимцима, у аутономној вожњи и летењу, игрању логичких игара на табли попут шаха, рачунарских игара али и за класификацију текста, анализа осећања изражених у тексту, предвиђање тока болести код пацијената и препорука терапије, анализа друштвених мрежа, препознавање и синтеза говора, аутоматско описивање садржаја фотографија и тако даље. У многим од ових примена, ефикасност ML је увелико превазишла ниво ефикасности људских експерата [63].

Вртоглави развој ове области је последица све веће доступности података и све већег обима података који се чува, са једне стране, и све веће снаге рачунара без којих све то не би ни било могуће, са друге стране. Узимајући у обзир широк дијапазон примене и брзину развоја рачунарских компоненти пратећи захтеве експерата, јасно је да је ML жариште подједнако академске и индустријске јавности, па су реална очекивања да ће се потенцијали ML и даље ширити [64], [65], [66].

2.1. Основни појмови, објашњење

У литератури се појмови *машинско учење, вештачка интелигенција, дубоко учење* и *наука о подацима* често користе као синоними иако то није потпуно исправно. Наиме, појам вештачка интелигенција (енгл. *Artificial intelligence,* скраћено *AI*), представља најшири појам који обухвата машинско учење у оквиру кога се јавља дубоко учење (енгл. *Deep learning*) и рад са обимним подацима (енгл. *Big data*), а сваку од ових области прожима наука о подацима (енгл. *Data science*) (Слика 2-1). Инхерентна веза између ових појмова често доводи до њихове погрешне употребе.



Слика 2-1: Однос основних појмова у области AI [66]

Прецизније, ML представља подскуп AI и има за задатак да развије алгоритме способне да уче из историјских података о појавама и да научено употребе приликом доношења одлука. Учећи, алгоритми су у стању да мењају сопствени кôд како би били боље припремљени за будуће задатке [66]. Алгоритми машинског учења нису тачна математичка решења проблема, већ њихова апроксимација [67].

Пре самог приступања процесу решавања проблема помоћу ML-а мора бити познато неколико аспеката:

- Алгоритам за учење који ће бити примењен и како он учи.
  - То превасходно зависи од проблема који треба решити.
- Подаци којима се располаже и над којима се врши обука.
  - Подаци могу бити „обележени“ (енгл. *Labeled*) или „необележени“ (енгл. *Unlabeled*), без ознаке класе којој припадају али свакако мора постојати одговарајући број узорака над којима ће тренинг бити извршен.
- Репрезентација података.
  - Различити алгоритми за учење подржавају различите типове података из којих могу да уче.
- Крајњи циљ који је и покретач за учење.
  - Жељени исход ће одредити како и шта алгоритам за учење треба да користи приликом обуке.
- Шта је (која је) циљна варијабла.
  - Да ли је циљ класификација неозначених података, предикција вредности варијабле, итд. [67].

Када се говори о поделама техника ML издваја се неколико група а према основној подели три су главне групе: ненадгледано (енгл. *unsupervised*), надгледано (енгл. *supervised)* и подржано (енгл. *reinforcement)* учење [64], [68]:

1. ненадгледано учење подразумева да су познате улазне вредности док се алгоритму оставља да предвиди излаз, не дајући му пожељне излазе док надгледано учење карактерише управо супротно, уз вредности улаза познате су и

вредности које им одговарају на излазу, те је потребно установити однос који важи између њих;
2. надгледано учење се најчешће бави предвиђањем излаза (означених са *y*) на основу улаза представљених у виду вектора означених са *x*. Ознакама $x_1, x_2,...x_n$ се обележавају улази, обележја, атрибути (енгл. *features*) при чему *n* представља број улазних атрибута. Са друге стране, обележена са *y* је типично једна, циљна, таргет променљива *m* (енгл. *target variable*). Постоје и сценарији по којима су и излази (*y*) вишедимензионални, или и улази и излази представљени ненумеричким вредностима, већ у виду секвенци, графова и слично;
3. алгоритам за подржано учење је способан да на основу агента који интерагује са окружењем, учи из својих грешака, поступака и покушаја. Тако се будући поступци бирају на основу стечених искустава у садејству са новим изборима.

Проблему откривања веза између променљивих (обележја) на основу запажања може се приступити на различите начине. Када су предмет обраде велики подаци потребно је успоставити методе које су у стању да уочавају везе аутоматски. Такве методе обично налазе функције које описују везу између атрибута и вредности циљне променљиве, имају централни значај у машинском учењу и познатије су као *модели*. Модела може бити бесконачно много али не могу сви (па чак ни један) описивати поменуте законитости савршено добро. Од добро постављеног модела се очекује да добро генерализује, односно да на основу доступних атрибута, приликом предвиђања вредности циљне променљиве ретко прави грешке. Случајеви где модел не прави никакву грешку у пракси не постоје па је појам генерализације у машинском учењу један од најважнијих и заправо представља циљ коме се тежи.

2.2. Надгледано учење

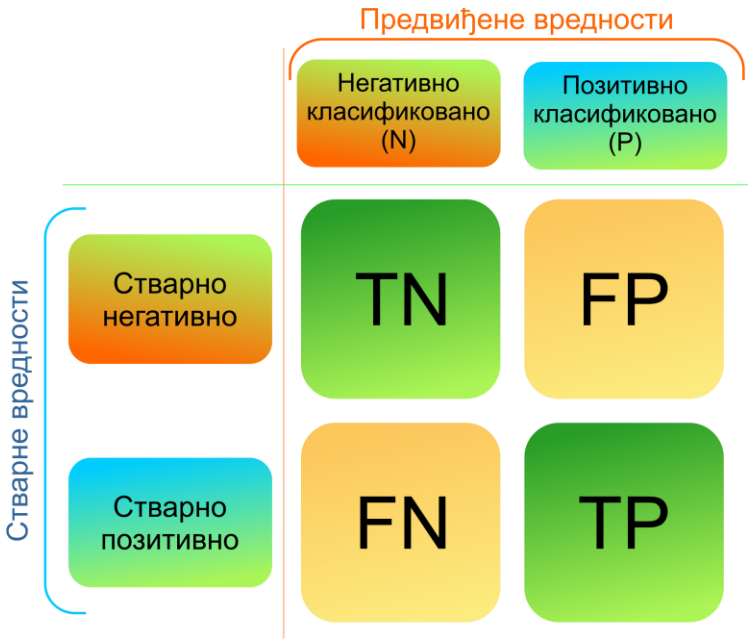
Основна одлика алгоритма надгледаног учења је да учи да предвиђа излаз за одређени улаз на основу парова улазних и излазних података који су му пружени у тренингу.

Две основне врсте проблема које се решавају помоћу надгледаног учења су *класификација* и *регресија* [68].

2.2.1. Класификација

Класификација је основни проблем у ML и има за циљ да се улазни подаци придруже некој од постојећих класа. Обично су класе дате у виду категоријских обележја која могу бити кодирана и тако представљена и у нумеричком облику.

Резултати класификације се најчешће приказују помоћу матрице тачности или конфузије (енгл. *Confusion matrix*). Матрица конфузије приказује стварне и предвиђене класе за различите комбинације, укључујући тачне позитивне, тачне негативне, лажно позитивне и лажно негативне (Слика 2-2).



Слика 2-2: Матрица конфузије

Где је:

- TP - проценат исправно класификованих примерака као позитивних (енгл. *True Positive*);
- FP - проценат неисправно класификованих примерака као позитивних (енгл. *False Positive*);
- FN - проценат неправилно класификованих примерака као негативних (енгл. *False Negative*);
- TN - проценат исправно класификованих примерака као негативних (енгл. *True Negative*) [69].

Из матрице конфузије могу се даље израчунати вредности као мере успешности класификатора, као што су тачност *(*енгл. *Accuracy,* скраћено *Acc),* прецизност *(*енгл. *Precision)* и опозив (енгл. *Recall*), према формулама (1), (2) и (3) респективно [69].

$$Accuracy = \frac{TP + TN}{P + N}, \tag{1}$$

$$Precision = \frac{TP}{TP + FP}, \tag{2}$$

$$Recall = \frac{TP}{TP + FN}. \tag{3}$$

Иако у теорији прецизност и опозив нису повезани, у пракси се висока прецизност постиже скоро увек на рачун опозива и висок опозив се постиже на рачун прецизности. Како од природе апликације зависи која је мера важнија, често се, ако је искључиво потребно дефинисати једну меру за упоређивање различитих класификатора, користи балансирана F_1 мера *(*енгл. *F_1-score* или *F-score*) која даје подједнак значај и прецизности и опозиву (формула (4)):

$$F_1 - score = \frac{2 * Precision}{Precision + Recall}. \quad (4)$$

F_1 мера представља хармонијску средину између прецизности и опозива која као таква има тенденцију да буде ближа мањој од њих, тако, да би F_1 мера била висока, обе вредности (и прецизност и опозив) морају бити високе.

Следећа битна мера је права позитивна стопа (енгл. *True Positive Rate*, скраћено *TPR*) дефинише се као део фракције стварно позитивних случајева који су тачно класификовани (формула (5)) [69].

$$TPR = \frac{TP}{TP + FN}. \quad (5)$$

Лажно позитивна стопа (енгл. *False Positive Rate*, скраћено *FPR*) се дефинише као фракција стварно негативних случајева класификованих као позитивних (формула (6)).

$$FPR = \frac{FP}{TN + FP}. \quad (6)$$

TPR је у ствари опозив позитивне класе и такође се назива и осећајност (енгл. *Sensitivity*). Са друге стране је специфичност (енгл. *Specificity*) која представља TNR или опозив негативне класе, дефинише се као (формула (7)):

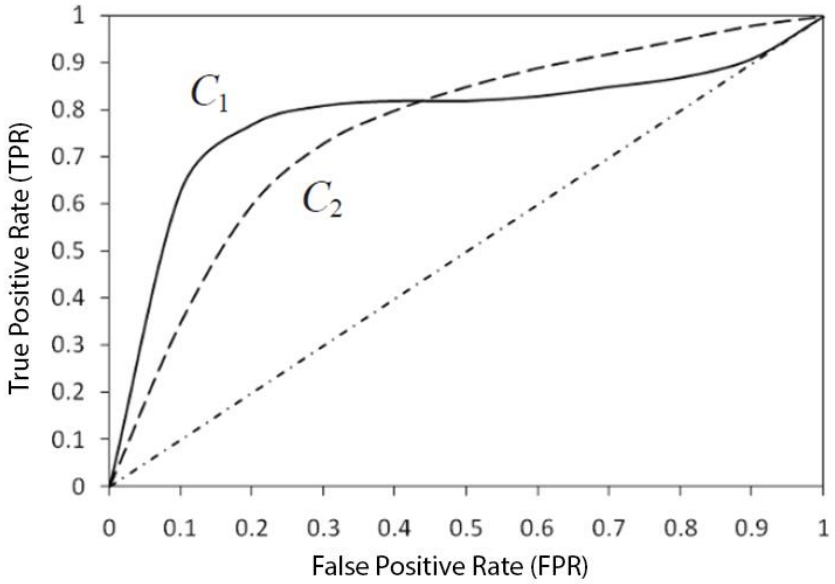
$$TNR = \frac{TN}{TN + FP}. \quad (7)$$

Из претходне две формуле следи (формула (8)):

$$FPR = 1 - TNR. \quad (8)$$

Још једна битна мера за евалуацију класификационог модела јесте и ROC крива (енгл. *Receiver Operating Characteristic Curve*). Вредност обележена као ROC крива је заправо површина која представља однос стварно позитивног стања насупрот лажно позитивног стања. Користи се често за процену резултата класификације са две задате класе. Поред ове треба истаћи и меру која се добија као површина испод ROC криве (енгл. *Area Under the Curve - AUC).* AUC је површина која показује тачно предиковане вредности као позитивне (TP) у односу на нетачно предиковане вредности као позитивне (FP). AUC се примењује за мерење перформанси бинарних класификатора, мери дискриминациону моћ класификатора, односно његову способност да разликује инстанце које припадају различитим класама. Вредности којима се исказује AUC се креће у интервалу од 0 до 1. Класификатор је бољи што је AUC вредност класификатора ближе 1. Вредности од 0,7 до 0,8 се сматрају прихватљивим, док су вредности од 0,8 до 0,9 јако добре, односно вредности више од 0,9 се сматрају одличним резултатом [69].

Слика 2-3 даје приказ ROC криве за пример два класификатора C_1 и C_2 над истим тест подацима. Криве полазе из координатног почетка (0,0) и завршавају се у (1,1). У тачки (0,0) сви тест примерци су класификовани као негативни док су у (1,1) су сви класификовани као позитивни. Дијагонална линија представља случајно погађање где је *FPR = TPR.*



Слика 2-3: ROC крива за два класификатора [69]

2.2.2. Регресија

Регресија је проблем предвиђања непрекидне циљне променљиве (и њене зависности од једне или више других променљивих) док се класификација односи на проблем предвиђања категоричке циљне променљиве (предвиђање којој од постојећих класа инстанца припада).

У овој дисертацији ће акценат бити на проблему вишеструке регресије. Регресија је статистички метод који се користи за анализу односа између улазних (независних) променљивих и излазне (зависне) променљиве. Основни циљ регресије јесте предвиђање вредности излазне променљиве и утврђивање зависности у односу на вредности улазних променљивих. Сврха регресије јесте да се утврди облик везе између посматраних појава што се постиже помоћу одговарајућег модела.

Један од најједноставнијих проблема представља линеарна регресија где се однос између улазних и излазних променљивих описује линеарном функцијом. То значи да се излазна променљива предвиђа на основу линеарне комбинације улазних променљивих, са одговарајућим коефицијентима.

У нелинеарној регресији, функција која описује однос између улазних и излазне променљиве је нелинеаран и комплекснији од једноставне линеарне зависности.

За случај регресије, није једноставно одредити која је то вредност циљне променљиве прихватљива као резултат већ се обично прати више параметара у циљу минимизације грешке модела.

Процес решавања проблема, односно минимизације просечне грешке модела назива се процесом обуке (обучавање модела, енгл. *Train*).

Обучавање модела је процес тренирања регресионог модела на тренинг скупу података. У овом кораку, регресиони модел ће научити како да предвиди излазну вредност на основу улазних података.

Када је модел обучен на тренинг скупу података, следи провера модела на тест скупу података. Тест скуп података се користи за проверу тачности предвиђања модела. Уколико је модел успешан у предвиђању излазних вредности на тест скупу података, може се сматрати успешним.

Генерално посматрано, за регресију важи да једној вредности атрибута не мора одговарати тачно једна вредност циљне променљиве већ се може говорити о расподели њених вредности при датим условима (вредностима атрибута). Жељена функција је математичка функција која описује однос између улазних и излазних података у регресионом моделу и назива се *регресиона функција*.

2.3. Линеарна регресија

Један од најједноставнијих модела у машинском учењу јесте модел линеарне регресије (енгл. *Linear Regression*). У контексту линеарне регресије, регресиона функција је линеарна комбинација улазних променљивих и дефинисана је изразом (9):

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n, \tag{9}$$

где је:

- y излазна променљива која се предвиђа,
- $x_1, x_2, \ldots, x_n$ су улазне променљиве,
- β_0 је интерсепт,
- $\beta_1, \beta_2, \ldots, \beta_n$ су коефицијенти који представљају утицај сваке улазне променљиве на излазну променљиву.

Очекивана вредност ($r(x)$) циљне променљиве, добијена помоћу регресионе функције може дефинисати изразом (10):

$$r(x) = E\left[y \middle| x \right] = \int y \, p(y|x) dy, \tag{10}$$

где је:

- $E\left[y \middle| x \right]$ математичко очекивање (очекивана вредност) циљне променљиве y када су познате вредности улазних података x;
- $p(y|x)$ условна вероватноћа да је циљна променљива y једнака некој вредности, ако су познате вредности података x и
- dy представља диференцијалну вредност циљне променљиве у контексту интеграла.

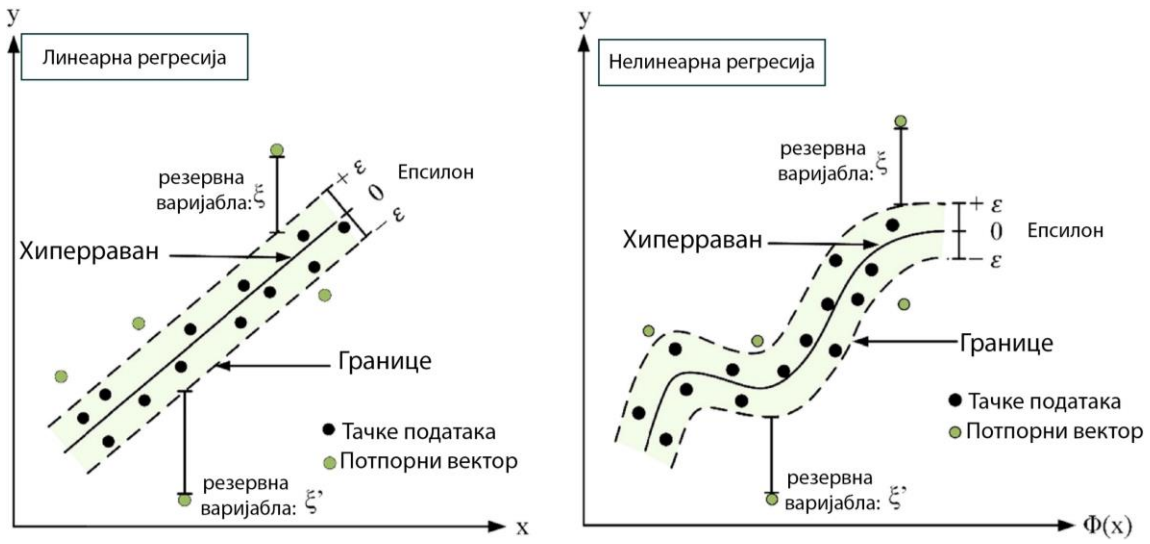
Како је ризик дат изразом $E\left[(y - f_w(x))^2 \right]$, свеприсутни приступ у решавању проблема регресије дат је следећом формулацијом (11):

$$\min_w \frac{1}{N} \sum_{i=1}^{N} (y_i - f_w(x_i))^2, \tag{11}$$

где је y_i стварна вредност циљне променљиве док је $f_w(x_i)$ вредност предвиђена помоћу регресионе функције.

2.4. Линеарна регресија са потпорним векторима

Линеарна регресија са потпорним векторима, (енгл. *Linear Support Vector Regression,* скраћено *Linear SVR*) је варијација *Support Vector Machine (*скраћено *SVM)* алгоритма који се користи за решавање проблема регресије. Линеарна варијанта овог алгоритма се фокусира на формирање линеарног модела регресије и посебно је погодан за обимније скупове података (Слика 2-4). За разлику од, на пример, просте линеарне регресије, овај алгоритам проналази хипер раван (енгл. *Hiperplane*) кроз коју пролази што више тачака (уместо једне линије). Овај принцип је посебно значајан за нелинеарне проблеме какви су и реални проблеми.



Слика 2-4: Линеарна и нелинеарна регресија са потпорним векторима [70]

Linear SVR користи линеарну функцију пресликавања како би моделирао однос између улазних атрибута (независних променљивих) и циљне променљиве (зависне променљиве). Ова линеарна функција пресликавања је облика $f(x) = wx + b$, где x представља улазне атрибуте, w су тежински фактори који моделирају утицај сваког атрибута, а b је пристрасност (биас) или слободни члан.

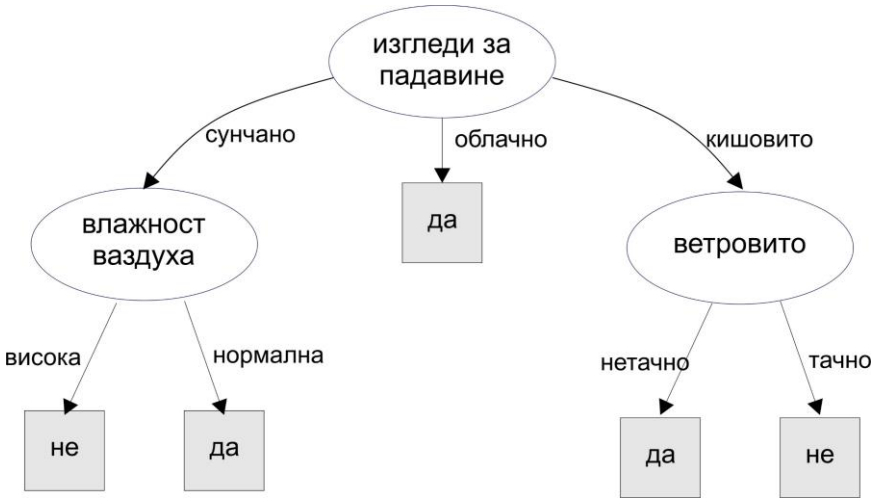
2.5. Хубер регресија

Хубер регресија (енгл. *Huber Regression*) је у основи унапређен модел линеарне регресије. Ова врста линеарне регресије користи Хубер функцију као функцију губитка уместо традиционалне методе најмањих квадрата [71]. Хубер функција губитка регресију чини робустном и отпорном на аутлајере (енгл. *Outliers*) што је посебно погодно за рад са скуповима података о стварним појавама.

Хубер функција губитка је комбинација квадратног губитка за тачке података чија су предвиђања блиска стварним и апсолутног губитка (енгл. *Absolute Loss*) за тачке података чија предвиђања нису блиска стварним [72], [73].

2.6. Стабла одлуке

Стабла одлуке (или одлучивања) (енгл. *Dessision Tree*) представљају једноставан и интерпретабилан модел машинског учења, који налази своју примену и у проблемима регресије и класификације [63], [74]. Структура овог модела подсећа на структуру стабла дрвета где се у сваком чвору стабла, осим листова, налази и по један тест – одлука (Слика 2-5). Даље гранање зависи од резултата теста који се спроводи у чвору – сваком исходу теста одговара једна грана стабла која води до следећег чвора. Листови стабла су означени вредностима које представљају предвиђања стабла. Тестови који се врше у сваком чвору представљају провере вредности појединачних атрибута. Тако, за категоријска обележја се у чворовима врши провера једнакости са неком конкретном вредношћу или провера припадности неком скупу вредности док у случају обележја са непрекидним вредностима тест представља проверу да ли је вредност атрибута већа или мања од неке друге вредности.



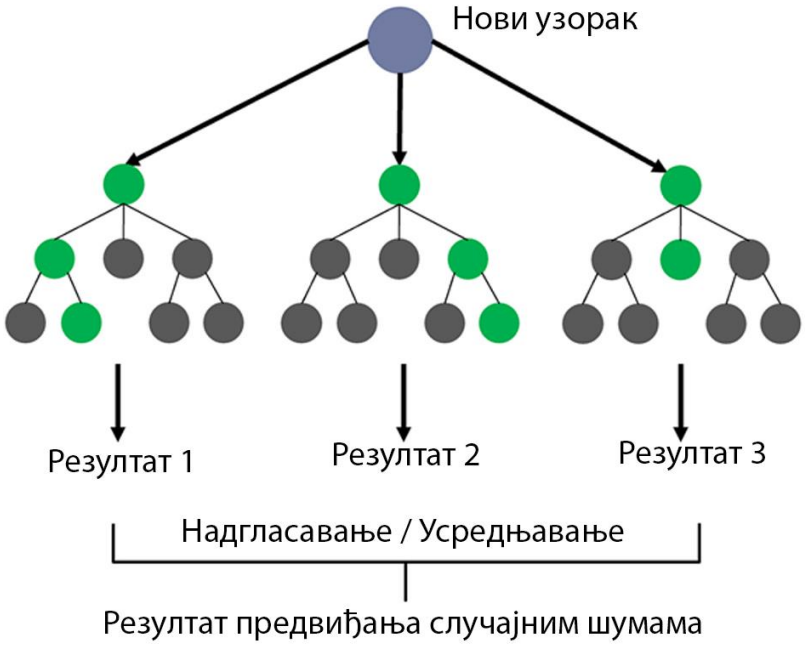
Слика 2-5: Приказ структуре стабла одлуке [63]

Стабла одлуке се не сматрају моделима високе прецизности [63]. Најчешћи проблем који се може јавити јесте проблем претприлагођавања што се најчешће дешава у случају дубоких стабала као последице великог броја тестова који се спроводе у чворовима. Са друге стране плитка стабла типично доводе до проблема потприлагођавања. Ипак, и поред ових мањкавости стабла одлуке налазе своју примену посебно због своје лаке интерпретације. Свака путања од корена до листа може приказати као *„if ... then“* правило у којем услов представља коњункцију свих исхода тестова дуж путање, док је одлука вредност у самом листу.

2.7. Случајне шуме

Случајне шуме (енгл. *Random Forest*) или колекција више стабала одлуке заснива се на простој агрегацији више стабала одлучивања [63], [75] (Слика 2-6). Успешност ове технике почива у самом начину изградње „шуме“. Током обуке, *m* стабала одлуке се тренира на различитим подскуповима података (формираним из скупа за обуку). Управо овај број појединачних стабала садржаних у моделу случајне шуме представља основни регуларизациони фактор чијим се повећавањем смањује опасност од претприлагођавања

[76]. Ипак, треба бити пажљив јер већи број стабала у колекцији повећава број рачунских операција а самим тим утиче на време извршавања. Иако их чини колекција више појединачних стабала одлуке, случајне шуме нису интерпретабилне као стабло одлуке.



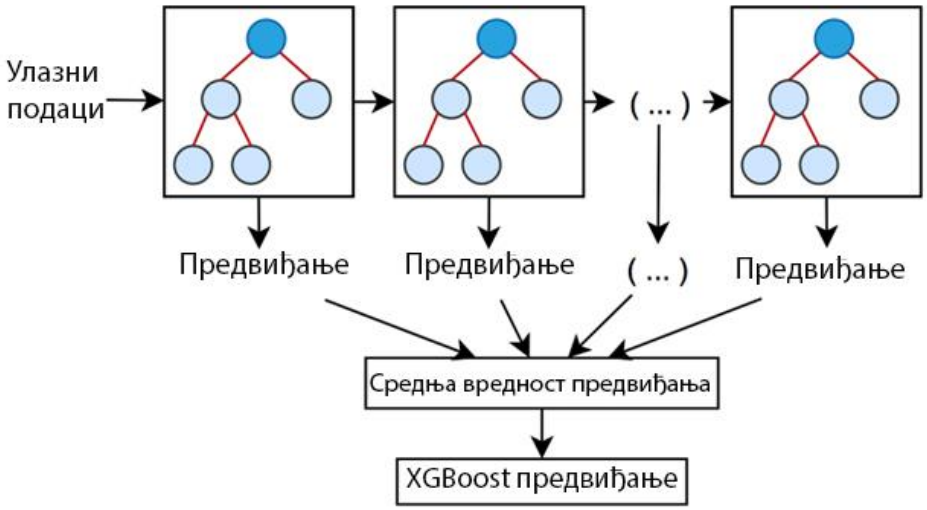
Слика 2-6: Приказ структуре случајне шуме [77]

2.8. Градијентно појачавање (буст) и екстремно градијентно појачавање

Приликом формирања ансамбла више модела (енгл. *Ensemble*) односно колекције, они се формирају потпуно независно а потом укључују у колекцију. За постизање што бољих резултата користи се принцип појачавања (енгл. *Boosting*) заснован на постепеном додавању једног по једног модела који чине колекцију [63]. При томе, сваки од појединачних модела се обучава тако да што боље надомести слабости до тада формиране колекције.

У том смислу методе засноване на градијентном појачавању, као на пример градијентни буст (енгл. *Gradient Boosting*) сматрају се најефикаснијим [63]. Основна идеја потиче од градијентних метода оптимизације, које се заснивају на поправљању текућег решења оптимизационог проблема додавањем вектора пропорционалног негативној вредности градијента функције која се минимизује.

Екстремни градијентни буст (енгл. *Extreme Gradient Boosting*, скраћено *XGBoost*) је још један од алгоритама погодан и за проблем класификације и за проблеме регресије. Попут случајних шума, представља колекцију више стабала одлуке уз додатак технике за појачавање ради постизања већих прецизности (Слика 2-7). Ова техника се заснива на једном једноставном, полазном стаблу одлуке. Израчунавањем грешке (разлика између предикованих и стварних вредности) додаје се следеће стабло које по правилу треба да умањи досадашњу грешку, понављајући процес све док се не постигне минимална грешка или достигну постављена ограничења. У односу на случајне шуме, ова техника се показала као доста бржа [78].



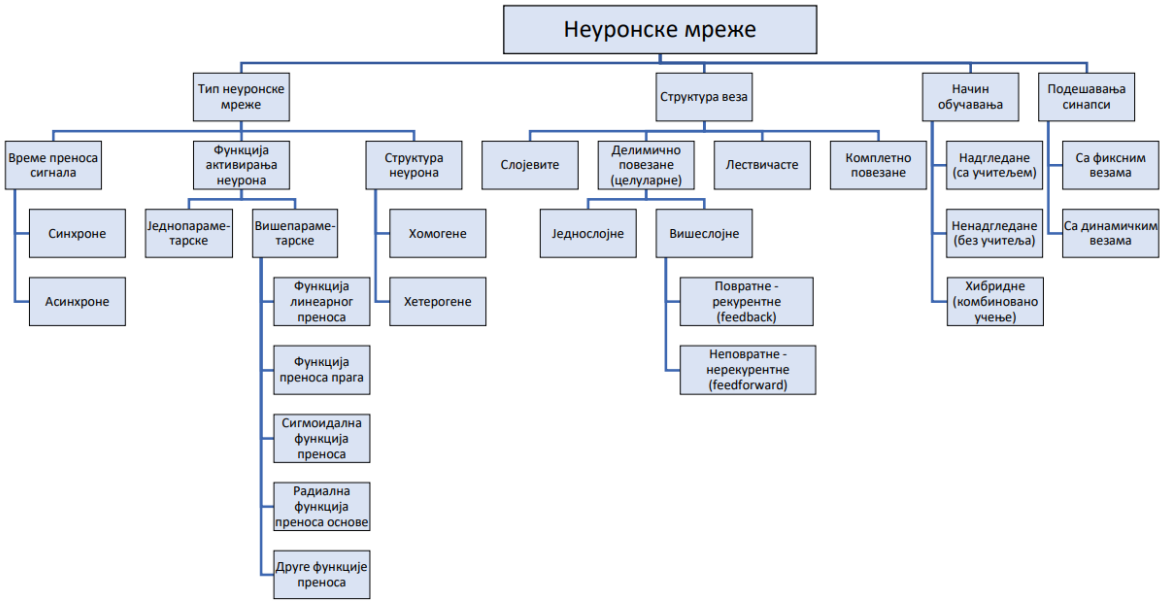
Слика 2-7: Приказ структуре екстремног градијентног буста [79]

2.9. Вештачке неуронске мреже

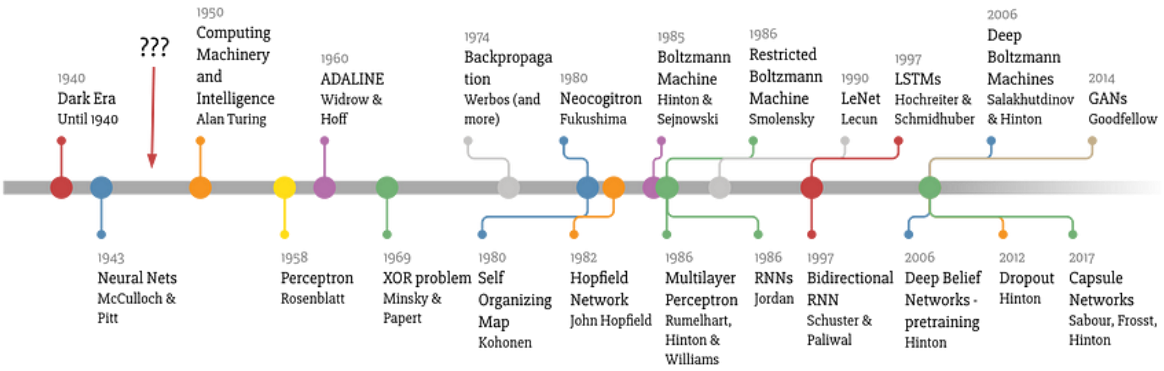
Како се однос између појава или обележја и циљне променљиве најчешће не може описати помоћу једноставнијих метода као што је већина претходно побројаних, већ то захтева сложенији нелинеарни приступ, употреба вештачких неуронских мрежа у том погледу пружа огромне бенефите. Вештачке неуронске мреже су познате као техника која са лакоћом врши обраду обимних података али и превазилази проблем недовољних података. Налазећи своје место примене у најразличитијим областима, вештачке неуронске мреже су у стању да препознају комплексне односе између зависног и независних обележја посебно захваљујући (активационим) функцијама које побуђују неуроне у сваком скривеном слоју. (О активационим функцијама ће више речи бити у поглављу 2.10.2.)

Како због своје сложености а тако и због распрострањености њихове примене поделу неуронских мрежа је могуће извести по основу више критеријума. Аутори у [80] и касније у [81] дају њихову свеобухватну поделу (Слика 2-8).

Иако се вештачке неуронске мреже сматрају граном која се још увек развија од њиховог помена прошло је 80 година. Аутор у [82] даје преглед значајних открића у овом домену од првог помена неуронских мрежа па до 2017. године али са нагласком на дубоко учење као шири појам од самог појма неуронских мрежа (Слика 2-9). Према приказу, први наговештај неуронских мрежа јавља се 40-их година двадесетог века иако 2000-их долази до најзначајнијих открића која су омогућила рад са мрежама какве су и данас актуелне. Ипак, још увек се трага за најбољим моделима, топологијом мреже и најбољим начинима за оптимизацију хиперпараметара модела. Паралелно се трага за најбољим начином *учења* (у смислу *машинског учења*), мислећи на процес аутоматског претраживања у циљу боље репрезентације података и знања која се црпе из њих. Период између 1940. и 1950. године према [82] нема важнијих достигнућа па је аутор обележава знаком питања („???“).



Слика 2-8: Класификација неуронских мрежа [80], [81]



Слика 2-9: Временска лента публикација из области неуронских мрежа према [82]

Вештачке неуронске мреже (енгл. *Artificial Neural Networks*, скраћено *ANN*) или скраћено „неуронске мреже“ (енгл. *Neural Network*, скраћено NN) је термин који је све присутнији и прожима готово све научне дисциплине представљајући најпопуларнију и најчешће коришћену методу машинског учења. Примена ANN је широка и помера домете вештачке интелигенције, рачунарства и примењене математике. ANN представљају параметризовану представу која може послужити за апроксимацију других функција [63]. За ANN се често каже да представљају једну сложену функцију насталу композицијом великог броја једноставних функција.

Основна идеја ANN јесте да се створи систем који ће препознати шаблоне у „понашању појава“ и деловати све више као људски мозак а мање као машина [83]. ANN су математички модели настали тако да симулирају живи нервни систем. Посебна моћ ANN јесте то што су у стању да генерализују научено и то примене на различите врсте проблема. ANN дају најбоље резултате у случају обимних података [84], [85].

Постоји више врста ANN:

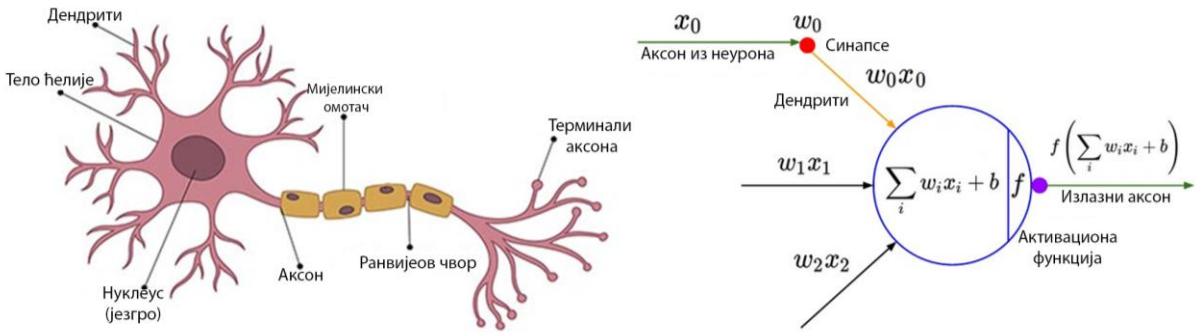
- потпуно повезане ANN - основна варијанта (енгл. *Fully Connected*);

- конволутивне ANN - за обраду слика (CNN, енгл. *Convolutional Neural Networks*);
- рекурентне ANN - за обраду података у виду низова варијабилне дужине (RNN, енгл. *Recurrent Neural Networks*);
- рекурзивне ANN - за податке који се могу представити стаблима одлучивања (RvNN, енгл. *Recursive Neural Networks*);
- графовске ANN - за обраду података који се могу представити графовима (GNN, енгл. *Graph Neural Networks*) [63].

Без обзира на тип, ANN се састоји од скупа јединица за обраду које се називају неурони. У овој дисертацији фокус ће бити на првој групи, на потпуно повезаним ANN .

Код потпуно повезаних ANN сваки неурон рачуна линеарну комбинацију својих аргумената и над њом рачуна неку нелинеарну трансформацију, познату као активациона функција. Како су неурони у мрежи груписани у слојеве, јединице једног слоја примају као своје аргументе вредности (тј. излазе) свих јединица из претходног слоја а затим их обрађују и прослеђују као своје излазе ка наредном слоју. Тако ANN формира нове атрибуте у сваком од скривених слојева надограђујући добијене атрибуте из претходног слоја дајући тако све сложеније и сложеније атрибуте. Ова способност ANN се посебно истиче код конволутивних мрежа а сматра се да успешност мрежа уопште проистиче из овог својства што посебно чини успешним дубоке ANN (енгл. *Deep Artificial Neural Network*) [63].

Основна градивна јединица ANN , вештачки неурон, представља грубу слику биолошког неурона и његове особине треба да опонашају особине биолошког неурона (Слика 2-10).



Слика 2-10: Поређење природног и вештачког неурона по њиховој структури [86], [87]

Сваки неурон у ANN се састоји од улазних веза (које симулирају дендрите у природним неуронима), активационе функције која трансформише пондерисану суму улаза у вредност на излазу и има само један излаз који може да се грана и може да буде улаз за више других неурона (слично као што се биолошки неурон везује за дендрите и ствара синапсе). ANN учи тако што подешава тежинске факторе аналогно јачинама синапси у биолошким.

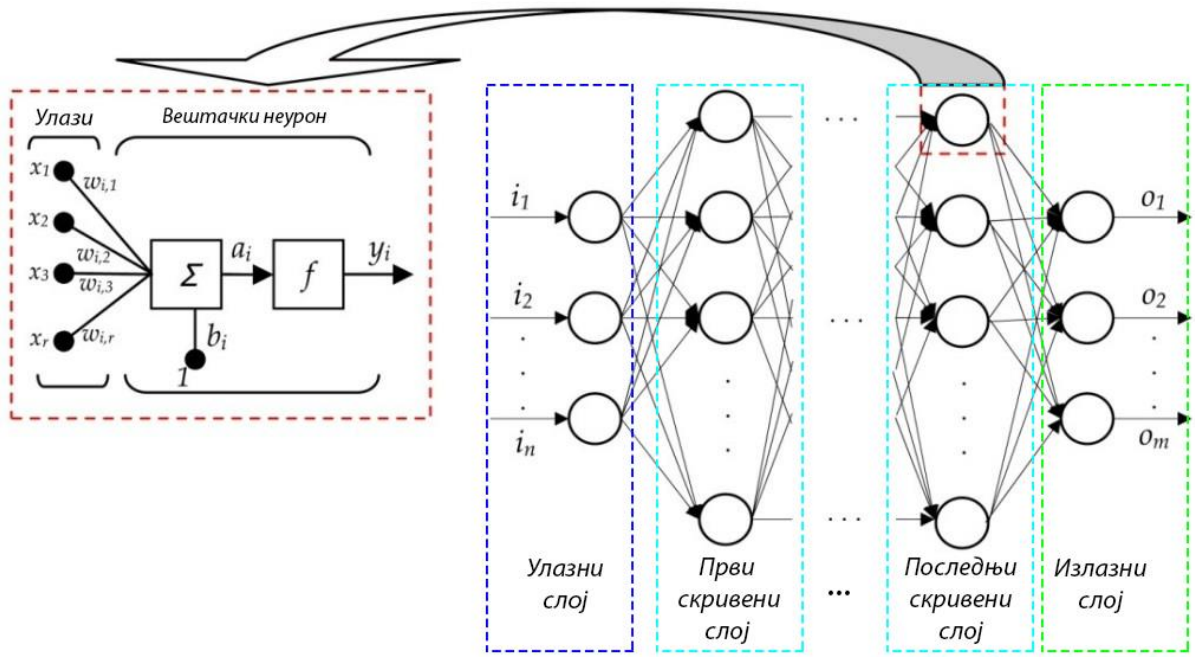
Како би ANN била у стању да препозна у будућности неки од шаблона, најпре мора проћи кроз фазу тренинга – обуке, алгоритам учења (енгл. *Learning algorithm*). Уз појам ANN и њихов тренинг, паралелно се срећу три важна појма: епоха (енгл. *Epoch*),

величина серије (енгл. *Batch size*)[1] и број итерација (енгл. *Iterations*). Једна епоха представља један пролаз напред и назад кроз све примерке тренинг скупа. Серија представља одређен број примерака тренинг скупа у једном проласку напред-назад. Величина серије се одређује као један од хиперпараметара (већа серија захтева већу меморију). Број итерација описује број пролазака напред и назад користећи величину серије.[2]

2.10. Архитектура ANN и хиперпараметри мреже

У најширем смислу речено, архитектура ANN одређује њен облик и структуру, односно број слојева, број неурона у њима и начин повезивања слојева и неурона у њима. Од саме структуре ANN зависи њена способност да извуче корисне закључке из података, претвори их у знање и генерализује стечено знање како би га применила на нове, непознате примерке података.

ANN се може описати као оријентисани, ациклични граф, а сваки неурон је функција преноса. Како су неурони организовани у слојеве онда се може рећи да базичну структуру чине три основна: улазни, излазни и скривени слој (или слојеви, опционо) [88], [89], [90], [91]. Број скривених слојева зависи од сложености проблема који се решава и чини својство дубине мреже. Слика 2-11 приказује структуру једне потпуно повезане ANN .



Слика 2-11: Општи приказ структуре потпуно повезане ANN [63]

[1] О серијама ће касније бити више речи.

[2] На пример: Ако у тренинг скупу има 1000 примерака а задата величина серије је 500, потребне су две итерације да би се извршила једна епоха.

У скривеним слојевима, за сваки неурон j сумирају се улазни сигнали x_i након што се помноже одговарајућим тежинама везе w_{ji}. Са становишта једног неурона, његов улаз чине промењљиве $x_1,...,x_r$ а промењљиве $w_{i,1},...w_{i,r}$ представљају тежине везе између неурона. Сваки неурон је параметризована функција која најпре врши линеарну комбинацију својих улаза (линеарни модел, као код линеарне или логистичке регресије) а онда вредност тог линеарног модела прослеђује у функцију f (активациону функцију). Ова функција треба да задовољи неке критеријуме: непрекидност, диференцијабилност и монотоност.

Узимајући за активациону функцију неку нелинеарну функцију, један неурон ће представљати један модел логистичке регресије. Понављајући тај принцип за сваки неурон ANN постаје композиција великог броја логистичких модела.

Излаз из сваког неурона је описан следећом формулацијом (12) [89]:

$$y_j = f(\sum w_{ji} x_i), \tag{12}$$

где је f представља активациону функцију, w_{ji} су тежински фактори а x_i улази у неурон.

Улазне јединице првог слоја се називају улазима у мрежу док је излазна јединица последњег слоја – излаз из мреже. Улазни слој прима податке и прослеђује их кроз ANN преко скривених слојева који повезују први и задњи слој. Уколико ANN садржи више од једног скривеног слоја она се назива дубоком ANN. Јединице из једног слоја примају све аргументе (као улазе) и прослеђују их јединицама наредног слоја. Најпростије речено, излази из једног слоја су улази у наредни слој, те се вредности неурона скривених слојева могу сматрати новим атрибутима објеката над којима остатак мреже учи апроксимацију циљне функције. У скривеним слојевима се конструишу нови атрибути, надграђујући постојеће чиме они постају све сложенији. Сваки неурон у слоју пролази кроз активациону функцију која се примењује на излазу неурона. Излазни слој се састоји од неурона који носе резултате и зависно од задатка може садржати један или више неурона.

Модел ANN би се могао дефинисати следећим изразом (13):

$$\begin{aligned} h_0 &= x \\ h_i &= g(W_i h_{i-1} + w_{i0}) \quad \text{где је } i = 1,2,...,L \end{aligned}, \tag{13}$$

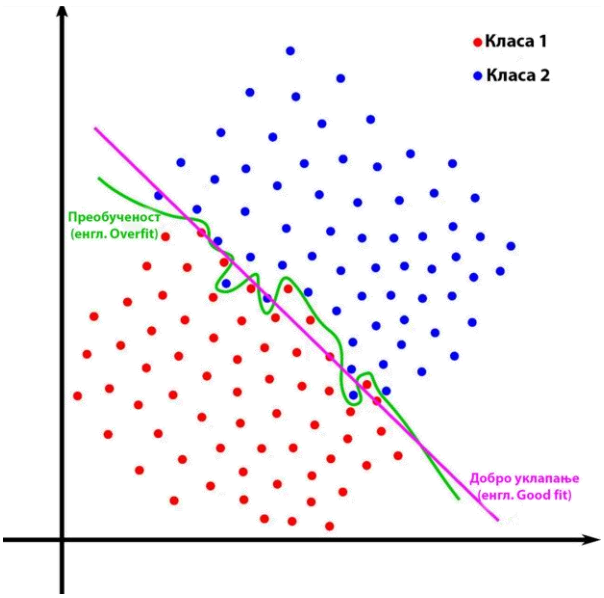
где је:

- x вектор улазних променљивих,
- L је број слојева,
- W_i је матрица чија j-та врста представља вектор вредности параметара јединице j у слоју i,
- w_{i0} представља вектор слободних чланова линеарних комбинација које јединице i-тог слоја израчунавају,
- g је нелинеарна активациона функција.

За вектор v, $g(v)$, представља вектор $(g(v_1), g(v_2),...,g(v_m))^T$, где је m димензионалност вектора.

Познато је да су перформансе ANN осетљиве како на број слојева тако и на број неурона у слојевима. Премало неурона може резултирати лошом апроксимацијом, док превише неурона може допринети проблемима преобучености (енгл. *Overfitting*, Слика 2-12) што јасно наглашава диспропорцију између постизања бољих перформанси мреже и

поједностављења топологије мреже. Преобучена мрежа има добро уклапање, фитовање (енгл. *Fitting*) само на тренинг скупу док за нове податке има јако слабу моћ препознавања шаблона.
Како би се превазишао проблем преобучавања прибегава се неком од типова регуларизације о којој ће касније бити више речи.



Слика 2-12: Илустрација преобучености (зелена линија) и доброг уклапања под утицајем регуларизације (љубичаста линија) [92]

Аутори у новијим радовима истичу важност ANN као и њихове дубине као најзначајнијег елемента, важнијег од броја неурона у слојевима [93], [94].

Архитектура модела ANN је одређена свим везама мреже и преносним функцијама неурона [95] кроз које се простиру информације. Према смеру у коме се врши простирање информација, вишеслојне ANN се могу поделити у две основне групе:

- мреже са простирањем унапред (енгл. *Feedforward Networks*), мреже код којих виши слојеви не враћају информацију у ниже слојеве и
- мреже са простирањем уназад (енгл. *Backward Networks*), мреже где виши слојеви враћају информације назад у ниже слојеве.

Говорећи о ANN, најчешће се мисли на мреже са пропагацијом унапред или вишеслојни перцептрони (енгл. *Multilayer Perceptron, MLP*). Основна структурална јединица MLP мреже јесте перцептрон, познат као Розенблатов перцептрон[3] (енгл. *Rosenblatt's Perceptron*) [96], [97]. MLP је модел са дубоким обучавањем [64], [98] и представља најчешће коришћен тип ANN са простирањем унапред, посебно због њиховог брзог рада и лакоће имплементације [89], [99], [100], [101]. Простирање унапред значи да ток

[3] Перцептрон је најранији модел ANN. Први радови на тему перцептрона датирају још из 1943. године и потичу од Warren McCulloch-a и Walter Pitts-a 1943. године а касније и од Rosenblatt-a (1958.), Minsky-ог и Papert-a (1969.).

сигнала кроз мрежу напредује у правцу напред, с лева на десно и на бази слој по слој. У овој мрежи су идентификоване две врсте сигнала [96]:

- функционални сигнали – улазни сигнал (стимулус) који долази на улазном крају мреже, шири се напред (неурон по неурон) кроз мрежу и појављује се на излазном крају мреже као излазни сигнал и
- сигнали грешке – сигнал грешке потиче од излазног неурона мреже и шири се уназад (слој по слој) кроз мрежу.

У процесу који се назива процес пропагације или простирања унапред (енгл. *Forward*) израчунава се излаз из ANN кроз три главна корака:

1. улаз мреже се копира у активацију улазних јединица;
2. скривени слојеви израчунавају своје активације тополошким редом;
3. излазни слој израчунава своју активацију и копира је у излаз мреже [102].

Мрежа са простирањем унапред дефинише мапирану функцију $y = f(x;w)$ и учи вредности параметара представљене у виду матрице w, што резултира најбољом функцијом апроксимације [64], [103], [104].

Промена тежина у сваком скривеном слоју пре излазног, израчунава се по следећем обрасцу:

$$w_{ji}(k+1) = w_{ji}(k) + \mu f'(net_j(k))(\sum_a \varepsilon_a(k) f'(net_a(k)) w_{aj}(k)) u_{ji}, \qquad (14)$$

где је $(\sum_a \varepsilon_a(k) f'(net_a(k)) w_{aj}(k))$ сума локалних грешака свих излазних неурона, помножених тежинским фактором.

Алгоритам MLP учи функцију $f(): R^n \rightarrow R^o$ обучавајући на означеном скупу података, где n представља димензију улаза, а o димензију излаза. Улазни параметри представљају се вектором $x = x_1, x_2, \ldots, x_n$, док се излази представљају вектором $y_{\text{pred}} = y_{1\text{pred}}, y_{2\text{pred}}, \ldots, y_{n\text{pred}}$.

Као и код других метода машинског учења, постављање оптималних параметара модела ANN се врши математичком оптимизацијом неког критеријума квалитета апроксимације и представља прави изазов [63].

Перформансе ANN зависе првенствено од њене архитектуре и параметара постављених при њеном моделовању. Дубља архитектура најчешће значи да је мрежа ефикаснија, а савремени тренд у теорији ANN истиче предности постојања што више скривених слојева, испред величине мреже [94]. Ипак, под архитектуром се не мисли само на број скривених слојева и неурона у њима, већ и на избор активационе функције, функције грешке коришћене за тренинг мреже, метрике којима ће бити измерена прецизност модела, итд.

Тешко је извести уопштени закључак када су подешавања хиперпараметара у питању. Аутори се одувек слажу да не постоји рецепт за успешност модела који би осигурао оптималне резултате већ да се до њих долази емпиријским путем на основу доброг познавања проблема који се решава. Ипак, аутори у [105] истичу неке од најзначајнијих закључака када је архитектура и поставка хиперпараметара у питању:

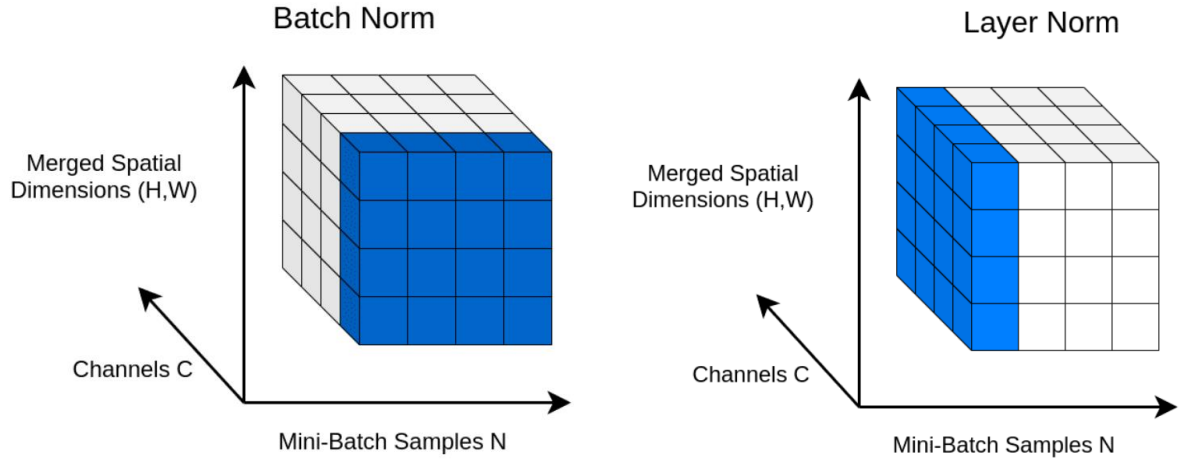
- за најбоље перформансе ANN треба да има 2 или 3 скривена слоја;
- број неурона у скривеним слојевима треба да буде прилагођен скупу података али најмање 100 неурона да би се постигле најбоље перформансе, па чак и до неколико стотина неурона;

- регуларизација у виду занемаривања дела неурона (енгл. *Dropout*) игра важну улогу, а најбоље резултате даје када се примени занемаривање од 50%;
- Гаусова дистрибуција (енгл. *Gaussian distribution*) за насумично изабране величине тежина и биаса (енгл. *Bias*) у сваком слоју дају најбоље резултате;
- број епоха не би требало ограничавати, треба да их буде најмање 300 у складу са могућностима. Иако велики број епоха често доводи до претренирања мреже, са употребом занемаривања дела неурона до тога не би требало да дође;
- за активациону функцију у општем случају препоручује се ReLU (енгл. *Rectifed linear units*) за скривене слојеве док се за излазни слој бира функција зависно од врсте проблема који се решава.

2.10.1. Скривени слојеви

За решавање једноставнијих проблема обично се користе густи (енгл. *Dense*) слојеви али за решавање већине реалних проблема то није довољно. Побољшању перформанси модела знатно доприноси нормализација улаза у слој ANN модела али често није довољна употреба нормализације само на улазу у први (енгл. *Input*) слој. Излазне јединице из сваког скривеног слоја (на пример, слоја *k-1*) представљају уједно и улазне јединице у наредни слој (слој *k*).

Хеуристички, откривено је да обични, густи слојеви неурона не дају довољно прецизне предикције. Стога су скривени слојеви у предложеном моделу сачињени од две врсте слојева. Прву врсту чине (густи) слојеви са нормализацијом тежина (енгл. *Weight Normalization Layers,* скраћено *WNL*) у сваком слоју, а другу врсту чине слојеви који врше нормализацију активације претходног слоја пре уласка у наредни слој (енгл. *Layer Normalization Layer,* скраћено *LNL*) [57]. Ова врста слоја врши нормализацију независно за сваки примерак у серији не обазирући се на величину серије за разлику од нормализације серије *(*енгл. *Batch Normalization,* скраћено *BN)* (Слика 2-13).



Слика 2-13: Поређење нормализације на нивоу серије и на нивоу слоја [57], [106], [107]

Слој нормализације (LNL) је први пут представљен 2016. године и убрзо је нашао примену у већем броју проблема у односу на друге типове нормализације [107]. Настаје из потребе да се превазиђу недостаци других видова нормализације тежина у слојевима што се постиже применом трансформације која одржава средњу активацију за сваки

примерак близу *0*, а сваку стандардну девијацију близу *1*. Број неурона на излазу овог слоја је исти као и на улазу, а нормализација сваког неурона у слоју се рачуна према формули (15):

$$\mu^1 = \frac{1}{H}\sum_{i=1}^{H} a_i^l$$
$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2}, \tag{15}$$

где важи:

- *H* означава број скривених јединица у слоју,
- под нормализацијом слоја, све скривене јединице у слоју деле исте услове нормализације за μ и σ , али различите инстанце имају међусобно различите услове нормализације.

LNL немају активациону функцију.

Слојеви који врше нормализацију тежина на нивоу слоја (WNL) раде то нормализујући тежине уместо активација. Нормализација тежине репараметризује тежине *w* (вектора) било ког слоја у ANN што је дато изразом (16):

$$w = \frac{g}{\|v\|} v\,. \tag{16}$$

Величина $\|w\| = g$ у претходној једначини је независна од параметра *v*. WNL одваја нормализацију вектора од његовог правца па је вектор који се обучава на даље заправо вектор *v*. Захваљујући својим одликама, WNL слојеви убрзавају конвергенцију не уводећи никакве зависности између примерака у мини-серији, а доприносећи бржем тренирању мреже.

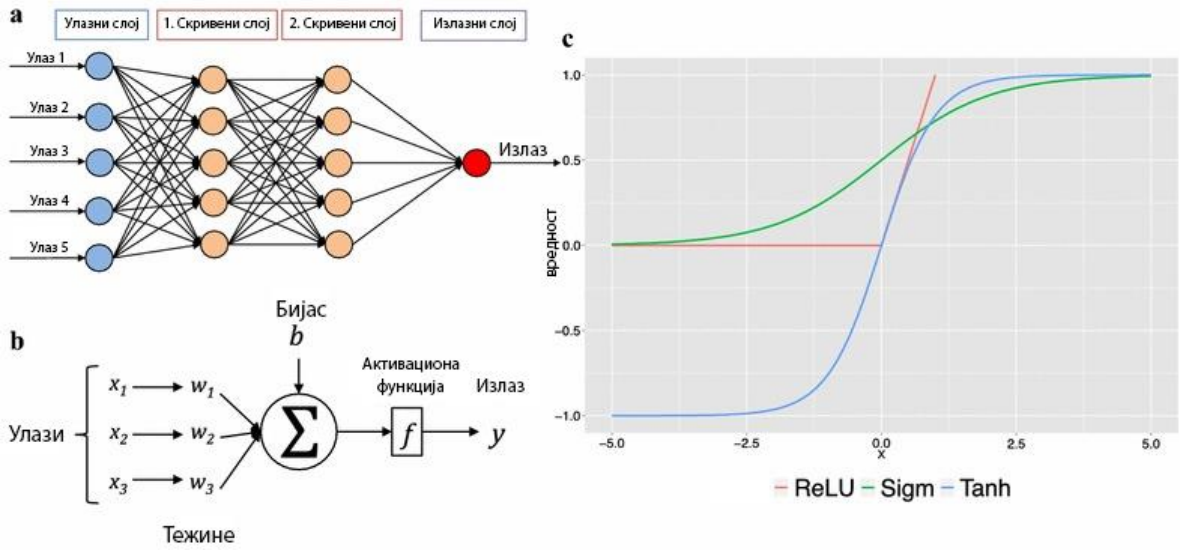
2.10.2. Активациона функција

Активациона функција (енгл. *Activation Function*) или како се може срести у литератури функција трансфера (енгл. *Transfer Function*) би се могла дефинисати као начин на који се тежински фактори обрачунати на улазу у слој трансформишу у излаз (од неурона до неурона у једном слоју). Активациона функција доприноси нелинеарној способности модела, односно, делинеаризује излаз из неурона како би се индивидуални неурони могли користити као улази у следећи слој. Постоји више активационих функција а у основи би се могле поделити у две велике групе:

1. линеарне активационе функције (енгл. *Linear Activation Function)* и
2. нелинеарне активационе функције (енгл. *Non-linear Activation Functions*).

Избор активационе функције има велики утицај на способност ANN да учи и користи научено а често одабир одговарајуће активационе функције представља један од највећих изазова. Неретко се дешава да за различите делове (слојеве) модела одговарају различите активационе функције па се у архитектури једног модела најчешће користи две или више врста активационих функција. Најчешће се за улазни слој користи једна активациона функција, за све скривене слојеве друга и опет за излазни слој, сходно проблему трећа активациона функција.

Иако, у основи било која функција може имати улогу активационе, обично се за активационе функције бирају оне функције чији облик делинеаризује улаз на тражени начин и чији извод је погодан за израчунавање у одређеном типу мреже. Према [105], за скривене слојеве најчешће се користи *ReLU* (ректификована, исправљена) али неретко и *Sigmoid* (логистичка), *Tanh* (хиперболична) или линеарна (најчешће за излазни слој) функција (Слика 2-14). Све три су по природи нелинеарне.



Слика 2-14: Активационе функције [105]

Сигмоидна (логистичка) функција оптимизације била је најчешће коришћена али она ипак није најпогоднија и не одговара сваком проблему. Сигмоидна функција (једначина (17)) је константна за све вредности осим за оне близу нули, па се њоме практично анулирају градијенти што отежава или готово потпуно онемогућава учење:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{17}$$

Тангенс хиперболична функција је дуго била у широкој употреби и то чак са већим успехом од сигмоидне јер је у близини нуле блиска идентитету што олакшава оптимизацију (једначина (18)):

$$\tanh(x) = \frac{\sin h(x)}{\cos h(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{18}$$

Трећа, ReLU (ректификована, исправљена линеарна јединица) представља најчешће коришћену активациону функцију посебно због своје једноставности али и својстава у оптимизацији. Иако ова функција има одређен степен недиференцијабилности, шансе да се дође до те тачке у процесу оптимизације нису велике, а ипак ако до тога и дође у каснијем току оптимизације та грешка ће бити надокнађена (једначина (19)):

$$\mathrm{Re}\,LU(x) = \max(0, x). \tag{19}$$

Извод у линеарном делу ове функције је константно једнак јединици што омогућава бржу конвергенцију. Раван део функције лево од нуле представља проблем за оптимизацију. Инстанце за које је активациона функција једнака нули не утичу на оптимизацију па се

као модификација ове функције добија *накошена* (енгл. *Leeky Rectified Linear Unit*), која за сваки параметар *x,* даје вредност максимума између *x* или *alpha*x* где је *alpha* константа са веома малом позитивном вредношћу (једначина (20)):

$$Leaky\,\mathrm{Re}\,LU(x) = \max(alpha * x, x)\,. \tag{20}$$

2.10.3. Функција оптимизације

У литератури се за функцију оптимизације појављује неколико синонима: функција губитка – цене (енгл. *Cost*), функција грешке (енгл. *Error*). Ипак, иако коришћени као синоними, у некој литератури термин *грешка* се односи на грешку измерену у једној тачки података, а *cost* је мера која израчунава грешку (просечно или збирно) у целом скупу података [83].

Функција оптимизације у ANN је заправо функција која се користи за оптимизацију тежина у мрежи како би се минимизирала грешка у предвиђању. Одабир функције оптимизације је један од кључних, јер различите функције могу утицати како на брзину конвергенције тако и на стабилност и сам квалитет решења.

Постоји више функција оптимизације које се могу користити у ANN а избор зависи од задатка тј. избора излаза [64], а неке од најчешће коришћених су:

- градијентни спуст (енгл. *Gradient Descent*) који представља и најчешће коришћену функцију оптимизације, која се заснива на рачунању градијента функције грешке по тежинама и коришћењу тог градијента за ажурирање тежина у мрежи. Као и већина оптимизационих функција, градијентни спуст се заснива на постепеном и итеративном приближавању решењу полазећи из насумично изабране тачке у правцу градијента. Ако се тачка полаза обележи са x_0 свака наредна тачка се добија применом правила из израза (21):

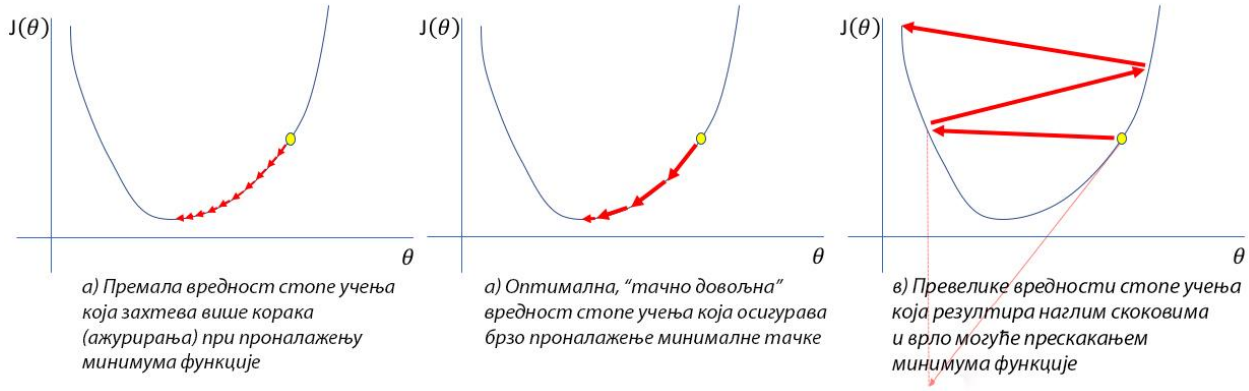
$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)\,, \tag{21}$$

  где је α_k дужина корака која се узима у супротном правцу од правца градијента;
- стохастички градијентни спуст (енгл. *Stochastic Gradient Descent*), сличан градијентном спусту, али се користи само подскуп узорака тренинг скупа за рачунање градијента, што чини алгоритам ефикаснијим и бржим за велике скупове података;
- *Adam* оптимизатор (енгл. *Adaptive Moment Estimation*) је адаптивни алгоритам оптимизације који користи процене првог и другог момента градијента функције грешке за ажурирање тежина;
- *RMSprop* оптимизатор (енгл. *Root Mean Square Propagation*) је алгоритам оптимизације који одржава покретне средње вредности квадрата градијента функције грешке како би се прилагодио различитим скалама градијента;
- *Adagard* оптимизатор (енгл. *Adaptive Gradient*) прилагођава стопу учења за сваки параметар на основу историје градијента;
- *Adadelta* оптимизатор (енгл. *Adaptive Delta*) је сличан *Adagard*-у, тј. представља његову унапређену верзију. Напредак овог оптимизатора у односу на *Adagard* огледа се у адаптивном скалирању степена учења [108].

2.10.4. Стопа учења

Стопа учења, (енгл. *Learning Rate*) је један од најважнијих (хипер)параметара који утиче на крајњи исход – брзину учења трениране ANN. Стопа учења би се најлакше могла објаснити као магнитуда промена, односно ажурирања тежина модела током процеса обуке са напретком уназад (енгл. *Backpropagation*). Вредности које експерт узима за стопу учења су најчешће врло мале (позитивне вредности између 0 и 1). У току обуке модела са пропагацијом уназад тежински фактори модела се ажурирају како би се смањиле грешке функције губитка (енгл. *Loss Function*). Уместо да се вредности тежина ажурирају користећи пуне износе градијента, оне се множе са вредношћу стопе учења. На пример, постављањем стопе учења на 0,2, значило би ажурирање пондера тако што се процењене тежине грешака (градијенти или промена укупне грешке) множе са 0,2.

Јасно је да стопа учења диктира колики ће бити степен (корак) оптимизације у потрази за минимумом функције губитка. Што је степен учења већи, већа су и ажурирања тежина, алгоритам ће учити брже али врло лако ово може условити прескакање минимума осциловањем око самог минимума (Слика 2-15, слика десно). Друга крајност, врло мале вредности за стопу учења, условиће мала ажурирања тежина што ће највероватније оптимизатор постепено водити до минимума али то може условити предуго конвергирање или чак довести до заглављивања у непожељном локалном минимуму (Слика 2-15, слика лево). Оно чему се тежи јесте баланс при коме је величина стопе учења „тачно довољна“ (енгл. *Just right*) али то се у пракси јако тешко постиже (Слика 2-15, слика у средини). У идеалном случају, стопа није премала да би алгоритам могао да конвергира али није ни превелика да би то увело алгоритам у стање прескакања оптимума [109].



а) Премала вредност стопе учења која захтева више корака (ажурирања) при проналажењу минимума функције

а) Оптимална, "тачно довољна" вредност стопе учења која осигурава брзо проналажење минималне тачке

в) Превелике вредности стопе учења која резултира наглим скоковима и врло могуће прескакањем минимума функције

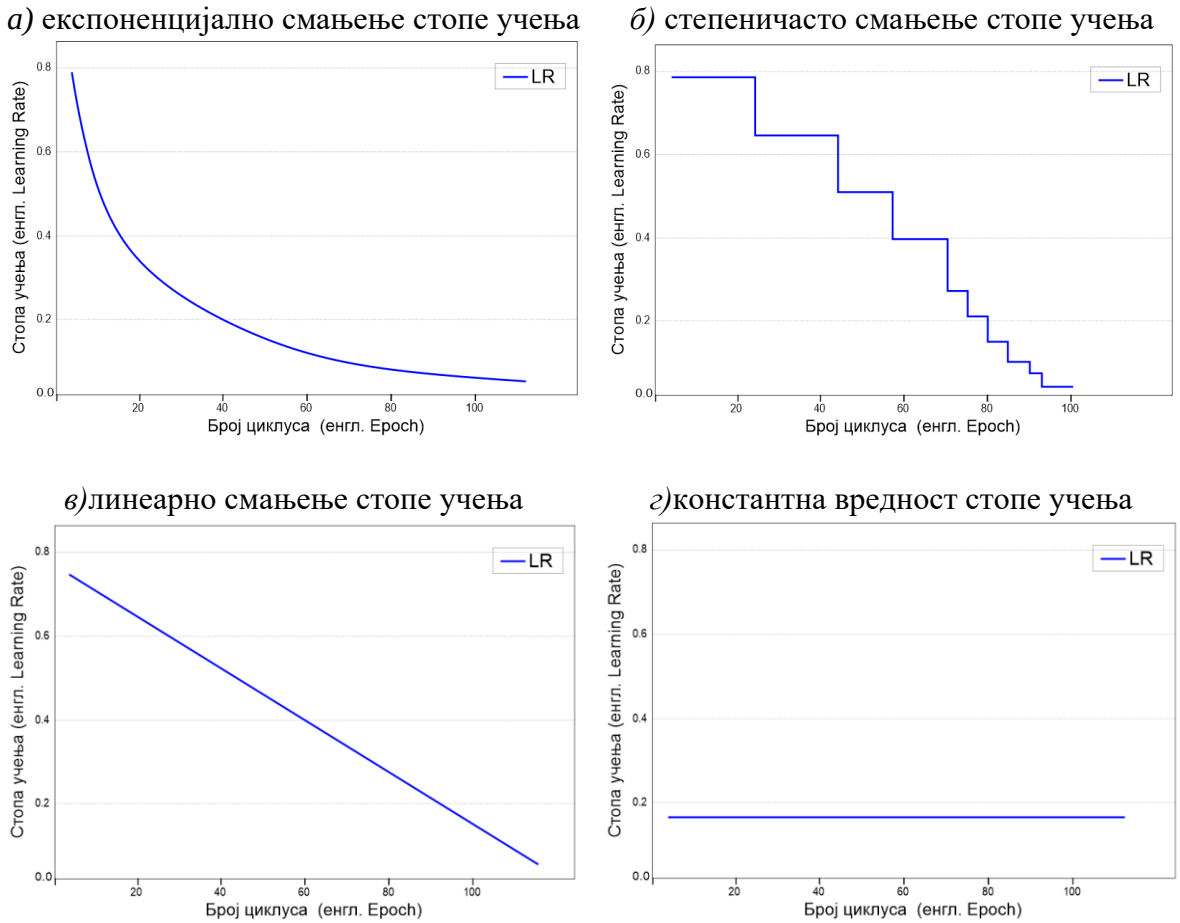
Слика 2-15: Стопа учења премала стопа учења *(а)*, оптимална стопа учења *(б)* и превелика стопа учења *(в)* [109]

Стопа учења не мора да има фиксну вредност па у складу са тим, уопштено посматрано, два најчешћа начина постављања њене вредности су:

- константна стопа учења – иста стопа учења током читавог процеса тренинга и
- променљива стопа учења – задаје се почетна стопа учења, а затим се у складу са постављеним условом (плански) смањује њена величина током процеса обуке.

Први начин се доста ређе користи па се проблеми са стопом учења решавају увођењем планске промене стопе учења (енгл. *Learning Rate Schedule*). Постоји много начина за

редукцију вредности стопе учења у току обучавања. Неки од приступа подразумевају да се стопа смањује након сваке (или одређеног броја) епоха или када метрике достигну задату вредност, итд. Генерално, планске промене стопе учења су унапред дефинисани оквири који прилагођавају брзину учења између итерација или епоха како тренинг одмиче. У том погледу треба имати на уму факторе који утичу на промену стопе учења. Ако се користи параметар пропадања (енгл. *Decay*) за смањење стопе учења, мора се експлицитно нагласити да ли се промене врше након сваке епохе или итерације, односно након сваког проласка кроз серију. Свакако, брзина и начин промене мора бити прилагођена проблему што се најчешће постиже хеуристички. На слици (Слика 2-16) приказани су неки од начина промене величине стопе учења. Слике од *а)* до *г)* респективно представљају експоненцијално, степеничасто и линеарно смањење и случај када степен учења има константно задату вредност.



Слика 2-16: Различити видови промене стопе учења током процеса тренинга

2.10.5. Величина серије за тренирање

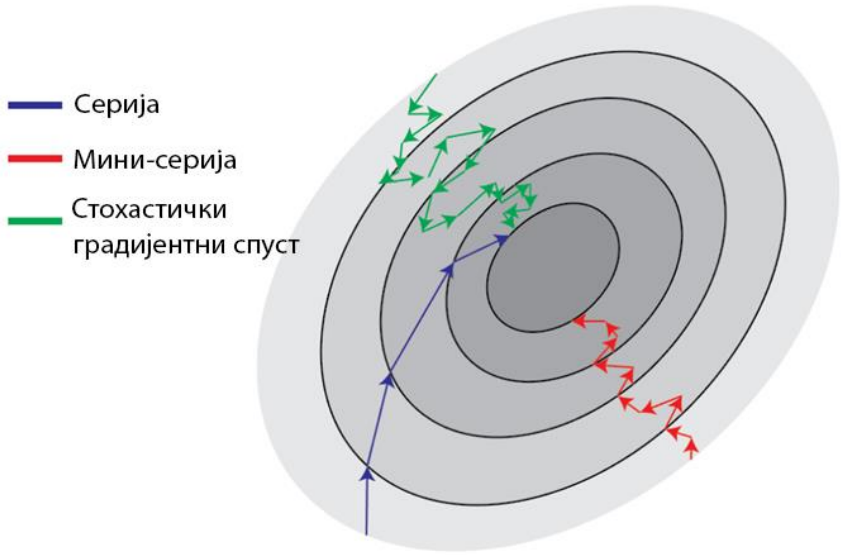
Серија је скуп улазних примерака (узорака) који се обрађују као једна целина у току тренирања модела. Коришћењем серија уместо обраде сваког улазног примера појединачно (енгл. *Online Learning*) знатно смањује време обраде и омогућава брже тренирање модела. Величина серије се најчешће одређује хеуристички и типично за

посматрани проблем и практично одређује колико улазних примерака ће бити процесирано као једна целина у току тренирања модела.

Постоје три типа градијентног спуста у односу на величину серије:

- серијски градијентни спуст (енгл. *Batch Gradient Descent*) – користи све узорке из скупа за обуку у једној епохи;
- мини-серијски градијентни спуст (енгл. *Mini-batch Gradient Descent*) – користи унапред дефинисани број узорака из скупа за обуку у једној епохи;
- стохастички градијентни спуст (енгл. *Stochastic Gradient Descent*) – користи само један случајни узорак из скупа за обуку у једној епохи.

На следећој слици (Слика 2-17) је приказан процес оптимизације функције губитка са све три врсте серије.



Слика 2-17: Поређење оптимизације функције губитка за читаву серију (плава путања), мини-серија (црвена путања) и стохастички градијентни спуст (зелена путања) [110]

Концентрични кругови на слици представљају контуре функције губитка са минимумом у центру. Сваки од поступака оптимизације тежи ка минимуму (најмањем кругу) али се зависно од величине скупа података, жељене брзине конвергенције функције али и ресурса рачунара врши одабир одговарајућег поступка.

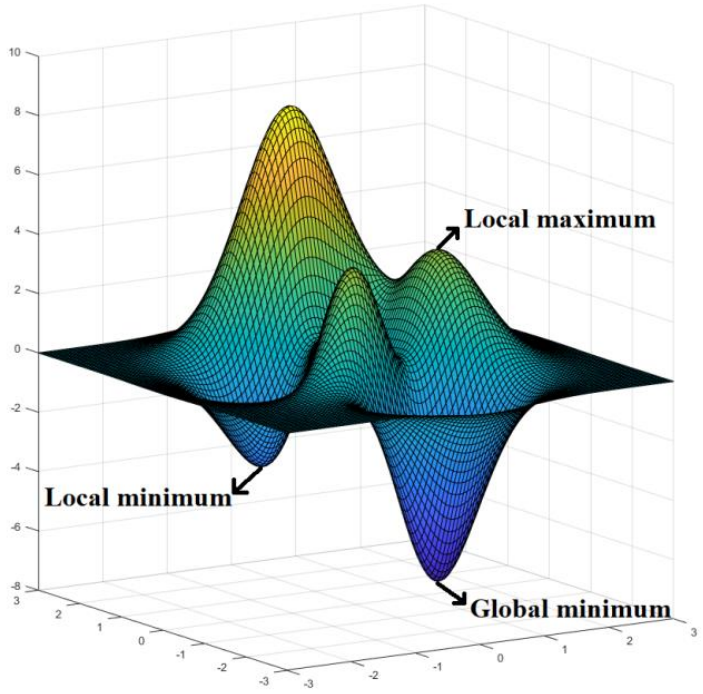
Серијски градијентни спуст (плава путања) може у идеалном случају брзо да доведе до минимума јер су кораци релативно велики али за разлику од стохастичког градијента сви тренинг подаци се разматрају у једном кораку због чега је трајање једног циклуса превелико.

Са друге стране, код стохастичког градијентног спуста може да се деси да се одабере погрешан правац у кретању ка минимуму па је самим тим путања са доста шума а често доста осцилација око минимума.

Мини-серија се у том погледу у пракси показала као најзахвалнија техника дајући генерално најбоље резултате вешто избегавајући локалне минимуме. Величина мини-серије утиче посебно на време обуке по свакој епохи, квалитет модела и слично. Обично

се одређује хеуристички, користећи као полаз број степенован на два, у распону између 16 и 512, водећи рачуна о величини серије у односу на величину тренинг скупа али и перформансе рачунара (односно величине CPU или GPU меморије рачунара).

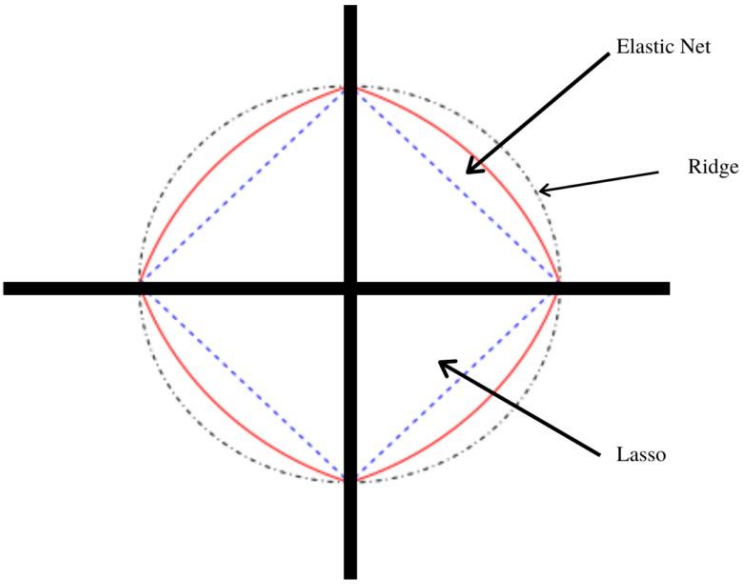
У општем случају најбоље се показала величина мини-серије од 32 (посебно за мање скупове), али то не мора бити препорука нити је рецепт за успех. Највећа предност употребе мини-серије у односу на бач јесте у избегавању локолних минимума [111]. Јасно је да функција може имати више локалних минимума и они врло лако могу заварати модел који доласком у било који од ових минимума ту и остаје (Слика 2-18).



Слика 2-18: Локални и глобални минимуми функције [112]

2.10.6. Регуларизација

Архитектура предложеног модела ANN је трослојна (улазни, излазни слој, а унутар њих скривени слојеви). Сваки скривени, густи слој пропушта сигнал у следећи слој. Као и у сваком другом проблему заснованом на ML, један од главних изазова јесте формирати такав модел који ће се одликовати добрим перформансама не само на скупу за обуку већ и за нове, непознате улазе (тест скуп). У том смислу се најпре мисли на мала одступања у предвиђањима, низак биас и мале варијансе што се у пракси решава најчешће на два начина: повећањем скупа за обуку или применом неког од типова регуларизације (*L1 - Lasso, L2 - Ridge* или комбинација оба типа *L12* tip, енгл. *Elastic Net)* [68], [94], [113] (Слика 2-19).



Слика 2-19: *Lasso (L1), Ridge (L2)* и *Elastic Net (L12)* регуларизација [114]

Проблем преучавања модела (енгл. *Overfitting*) често се јавља у случајевима комплексних модела са великим бројем параметара па се у функцију грешке уводи регуларизација као мера сложености модела.

Основна разлика између прва два приступа јесте у томе што *L1* доводи до стања да неки мање важни коефицијенти постану једнаки нули док се помоћу *L2* регуларизације смањују апсолутне вредности коефицијената, тако да велики број њих постаје мали али и даље различит од нуле.

Ради контролисања моћи учења представљеног модела и да би се избегла његова преученост, биће коришћен трећи тип регуларизације (*Elastic Net*). Губитак функције код *Elastic Net* типа регуларизације се дефинише као сума *обичног губитка* (квадратне разлике, на пример) и две компоненте регуларизације (*L1* и *L2*). Губитак се у том случају рачуна изразом (22):

$$обичан\,губитак + \alpha * (\lambda * L1 + 0.5 * (1 - \lambda) * L2), \tag{22}$$

где је:

- *обичан губитак* квадратна разлика,
- *L1* представља *L1* норму тежина која се рачуна као сума апсолутних вредности свих тежина,
- *L2* представља *L2* норму тежина израчунату као корен суме квадрата свих тежина,
- λ (*lambda*) је коефицијент који контролише равнотежу између претходне две регуларизације и обично се поставља између 0 и 1,
- α (алфа) је коефицијент који контролише укупну јачину регуларизације (већа вредност резултира јачом регуларизацијом).

Предност оваквог вида регуларизације проистиче из њене способности да занемари ситуације у којима постоји корелација између обележја, а истовремено је у стању да фаворизује неке ретке репрезентације.

2.10.7. Иницијализација параметара и биаса

Иницијализација биаса и параметара у сваком слоју се врши техником познатом као *He-normal* [115]. *He-normal* метода се појављује први пут 2015. године и по њој се почетни параметри биаса бирају из нормалне расподеле са средњом вредношћу од 0 и стандардном девијацијом скалираном према броју веза у одређеном неурону. Посебно повољне резултате овај тип иницијализације даје уз употребу *ReLU* активационе функције будући да она позитивне вредности оставља непромењене док негативне поставља на нулу, што је један од разлога зашто се у предложеном моделу користи управо овај вид иницијализације. Иницијализација тежина овом методом узима у обзир величину претходног слоја што помаже у постизању глобалног минимума функције трошкова брже и ефикасније. Тежине су и даље насумичне, али се разликују у опсегу у зависности од величине претходног слоја неурона. На овај начин се обезбеђује контролисана иницијализација са једне стране и брже и ефикасније спуштање градијента, са друге стране.

2.11. Процена успешности модела

Важно је имати на уму да ниједан модел машинског учења није савршен и да ће увек постојати могућности за побољшање. Редовно оцењивање и анализа квалитета у току обуке модела су веома корисни у идентификацији проблема и побољшању квалитета предвиђања.

Који ће метод за оцењивање успешности ANN модела бити коришћен зависи од конкретног задатка и података на којима је модел обучен. Постоје општи методи за оцењивање квалитета модела машинског учења који се могу применити на ANN као што су унакрсна валидација, праћење вредности задатих метрика, анализа грешке, праћење криве обучавања и валидације, упоређивање са другим моделима, итд [83]:

- унакрсна валидација (енгл. *Cross-validation*) подразумева поделу оригиналног скупа података на неколико подскупова (на пример, 5 или 10), након чега се обучавање и тестирање модела врши на сваком од подскупова, са каснијим упоређивањем резултата. Ово омогућава повећање поузданости процене квалитета модела и смањење вероватноће претераног прилагођавања;
- криве обучавања и валидације се користе за графичко приказивање зависности квалитета модела од броја примера у обучавајућем скупу података. Ово омогућава утврђивање како ће се квалитет модела променити са повећањем броја података, као и откривање могућих проблема са претприлагођавањем или потприлагођавањем;
- метод анализе грешака се користи за откривање типова грешака које модел прави и оцењивање њиховог утицаја на квалитет модела. На пример, ако модел прави велики број лажно-позитивних предвиђања, то може да укаже на потребу за променама параметара модела или увођењем додатних података;
- упоређивање са другим моделима који су обучени на истим или сличним подацима може да омогући боље разумевање степена успеха тренутно коришћеног модела и у том смислу прихватање или одбијање истог на најбржи начин;
- различите метрике квалитета могу се користити за процену ефикасности модела ANN у зависности од типа задатка. На пример, за задатке класификације могу се користити метрике тачности, опозива, прецизности и F_1 мера. За задатке регресије

најчешће се користе: коефицијент детерминације (енгл. *Coefficient of Determination,* скраћено R^2), средња апсолутна грешка (енгл. *Mean Absolute Error,* скраћено *MAE*), средња квадратна грешка (енгл. *Mean Square Error,* скраћено *MSE*), корен средње квадратне грешке (енгл. *Root Mean Square Error,* скраћено *RMSE*) и средња апсолутна процентуална грешка (енгл. *Mean Absolute Percentage Error,* скраћено *MAPE*).

Како је у овој дисертацији изучавани проблем регресионе природе, у наставку ће бити приказане метрике погоднија за процену регресионог модела.

2.11.1. Коефицијент детерминације (R^2) и прилагођени коефицијент детерминације (AR^2)

Коефицијент детерминације је пропорција варијације зависне варијабле која се може предвидети из независних варијабли (једначина (23)):

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{23}$$

где је y_i стварна вредност, y_i је предвиђена вредност, $\bar{y}$ је средња вредност стварних вредности и N укупан број примерака у скупу.

Недостатак ове мере јесте што не узима у обзир број независних променљивих у моделу. Из тог разлога развијена је варијација ове мере - прилагођени коефицијент детерминације (енгл. *Adjusted R Square,* скраћено AR^2), који при израчунавању узима у обзир и број независних варијабли и тако даје нешто прецизнију представу квалитета модела (једначина (24)).

$$AR^2 = 1 - \frac{(1-R^2)(N-1)}{N-p-1}, \tag{24}$$

где p број независних варијабли и N укупан број примерака у скупу. Вредности AR^2 су увек мање или једнаке R^2. Са порастом p вредност израза $(N-p-1)$ се смањује што утиче на смањење коначног резултата AR^2.

Вредност R^2 и AR^2 узимају вредности између 0 и 1, типично узимајући вредности ближе 1 за прецизније моделе, односно, што је вредност ових мера ближа 1 модел има већу моћ да се прилагођава подацима.

2.11.2. Средња апсолутна грешка

Средња апсолутна грешка (MAE) је метрика која се користи да измери колико се предвиђања модела разликују од стварних вредности. MAE се рачуна као просечна апсолутна вредност разлике између предвиђене вредности и стварне вредности (једначина (25)):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - y_i\right|. \tag{25}$$

Стварне вредности се обично представљају у облику низа или вектора а модел предвиђа вредности за сваку од њих. Вредности MAE су увек позитивне, и што је њихова вредност мања, то је модел бољи.

2.11.3. Средња апсолутна процентуална грешка

Средња апсолутна процентуална грешка (MAPE), представља просечну процентуалну грешку предвиђања модела и често се користи у прогнозирању вредности где је важно пратити процентуалну грешку. Представља просек апсолутних процената грешака сваког уноса у скупу података да би се израчунало колико су биле тачне предвиђене количине у поређењу са стварним количинама па је често ефикасан за анализу великих скупова података (једначина (26)):

$$MAPE(\%) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - y_i}{y_i}\right| * 100 . \tag{26}$$

2.11.4. Средња квадратна грешка

Средња квадратна грешка (MSE) је мера просечног квадрата грешке предвиђања модела.

(једначина (27)):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i)^2 . \tag{27}$$

2.11.5. Корен средње квадратне грешке

Корен средње квадратне грешке (RMSE), према [51]–[53] је једна од најчешће коришћених мера евалуације модела ANN . RMSE представља квадратни корен од просечног квадрата грешке (MSE) (једначина (28)):

$$RMSE = \sqrt{MSE}. \tag{28}$$

Неретко се јавља потреба да се ова мера изрази у процентима па формула за RMSE(%) гласи (једначина (29)):

$$RMSE(\%) = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i)^2}}{\bar{y}} * 100 . \tag{29}$$

2.12. Припрема улазних података

Како су подаци у свом изворном облику најчешће несређени и у неодговарајућем облику фаза предобраде, препроцесирање (енгл. *Preprocessing*) представља једну од најважнијих фаза од које умногоме зависи успешност развијеног модела [119], [120]. У оквиру ове фазе подразумева се рад са подацима у смислу њихове припреме за даљу обраду што обухвата више различитих техника (нормализацију, усредњавања, интеграцију података

из различитих извора, смањивање броја улазних обележја, чишћење података и допуну недостајућих, итд).

Како су ANN веома осетљиве на типове података и распоне вредности обележја, један од значајних поступака јесте трансформација категоријских обележја у облик погодан за обраду, у највећем броју случајева у нумеричке податке. Том приликом се може користити неколико различитих метода као што су: ординално кодирање (енгл. *Ordinal Encoding)*, етикетирање (енгл. *Label Encoding*), кодирање помоћу средњих вредности (енгл. *Target Encoding*, познато и као *Mean Encoding*) [121], [122], [123], неко од такозваних бинарних кодирања, заснованих на формирању нових варијабли на основу категорија у оквиру обележја (употребом нула и јединица). Такво кодирање је *One-Hot Encoding* (енгл. *One-Hot Encoding*) али и *Dummy Encoding. One-Hot Encoding* за сваку категорију обележја формира бинарни вектор, где ће присуство категорије бити обележено са 1, а одсуство са 0. Овакав вид бинарног кодирања је врло једноставан, али врло брзо долази до нагомилавања *нових* обележја из једног почетног, па се за категоријска обележја са великим бројем категорија препоручује употреба других поступака кодирања попут *Target Encoding* који категоријска обележја претвара у нумеричка, задржавајући се на оквирима једне колоне (не формирајући нова обележја) кодирањем помоћу средње вредности циљне (енгл. *Target*) варијабле. Оно што разликује *One-Hot* од *Dummy Encoding* поступка јесте што други формира једну колону мање у односу на број постојећих категорија што може само по себи бити разлог за баш његову употребу.

Поред ових постоји и право бинарно кодирање које је засновано на сличном принципу нула и јединица али се ова обележја формирају на нивоу једне колоне.

Поред поступка кодирања категоријских обележја, у препроцесинг фази подједнако важан јесте процес свођење нумеричких обележја у исти распон вредности, на исту скалу (скалирање података). Ово је посебно важно због интерпретабилности података али и због бољих рачунских својстава модела у који се укључују ови подаци, посебно због конвергенције модела. У том процесу може се користити неколико принципа и велики број њихових варијација [124], [125]. Неки од најчешће коришћених поступака у том смислу јесу нормализација, стандардизација, *L1* нормализација, *L2* нормализација итд. Иако се прва два термина могу срести као синоними, ради се о два потпуно различита принципа.

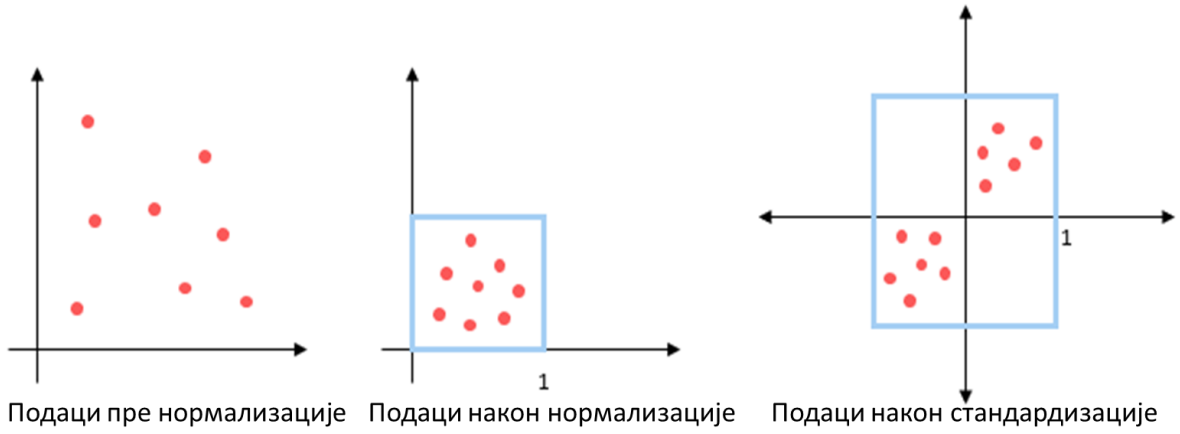
Стандардизација (позната и као *Z-score нормализација*) састоји се у томе да се од сваке вредности неког обележја одузме просек свих вредности тог обележја, па да се потом свака појединачна вредност подели стандардном девијацијом свих вредности тог обележја (формула (30)). Тиме се обезбеђује да сваки атрибут има просек 0 и стандардну девијацију 1 и посебно је корисна када подаци имају Гаусову расподелу вероватноће.

$$x_{std} = \frac{x - средња\ вредност(x)}{стандардна\ девијација(x)}. \tag{30}$$

Нормализација (позната и као *Min-Max нормализација*) је свођење вредности обележја на вредности између 0 и 1 али тако да за свако обележје минимална вредност добија вредност 0, а највећа 1, остале вредности су распоређене између према формули (31):

$$x_{norm} = \frac{x - \textit{најмања вредност}(x)}{\textit{највећа вредност}(x) - \textit{најмања вредност}(x)}. \quad (31)$$

Свака од два истакнута типа скалирања има своје предности зависно од карактеристика обележја. Један од можда најзначајнијих фактора које треба имати на уму приликом одабира типа скалирања података јесте његова осетљивост на аутлајере. У том погледу, нормализација је веома осетљива док је стандардизација доста робуснија (Слика 2-20).

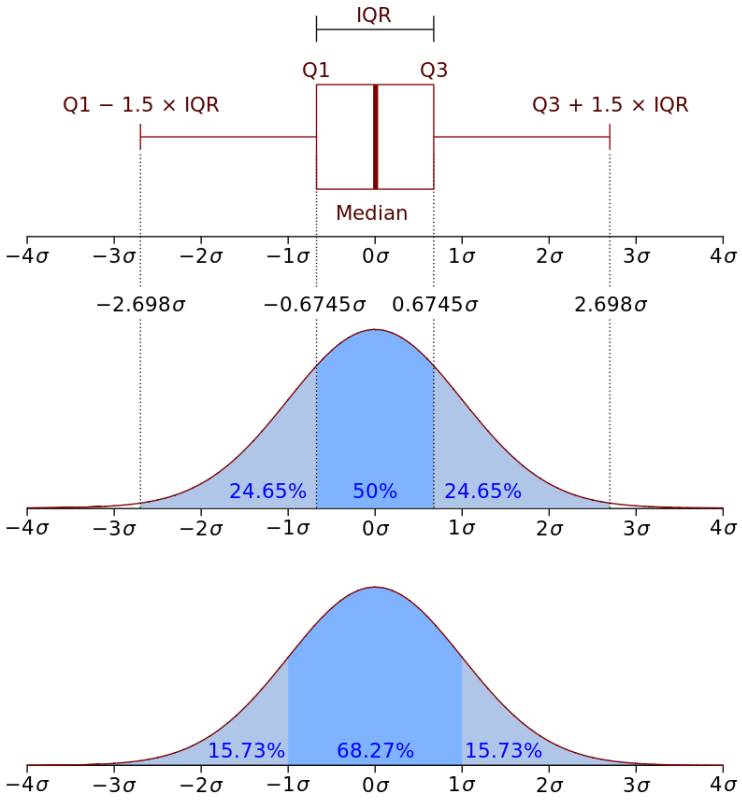


Слика 2-20: Утицај нормализације и стандардизације на податке [126]

Поред поменутих, као значајан механизам за скалирање података истиче се робустна верзија нормализације података (енгл. *Robust Scaler*). Математички гледано, ово је и потпуно природно узимајући у обзир да су таргет подаци резултат мерења стварне појаве и да се увек могу јавити подаци на ивици дистрибуције, са малом вероватноћом појављивања, али ипак постоје. Иако је оваквих података мало у односу на целокупан скуп, они имају утицај и могу искривити дистрибуцију вероватноће. Тиме ће и скалирање података употребом просте стандардизације бити отежано јер ће израчуната средња вредност и стандардна девијација бити искривљене под присуством одступајућих вредности [127]. Робустни скелер је принцип којим се приликом обрачуна средње вредности и стандардне девијације у обзир не узимају ови *одметнути* подаци (формула (32)).

$$x_{norm} = \frac{x - \textit{медијана}(x)}{Per75(x) - Per25(x)}. \quad (32)$$

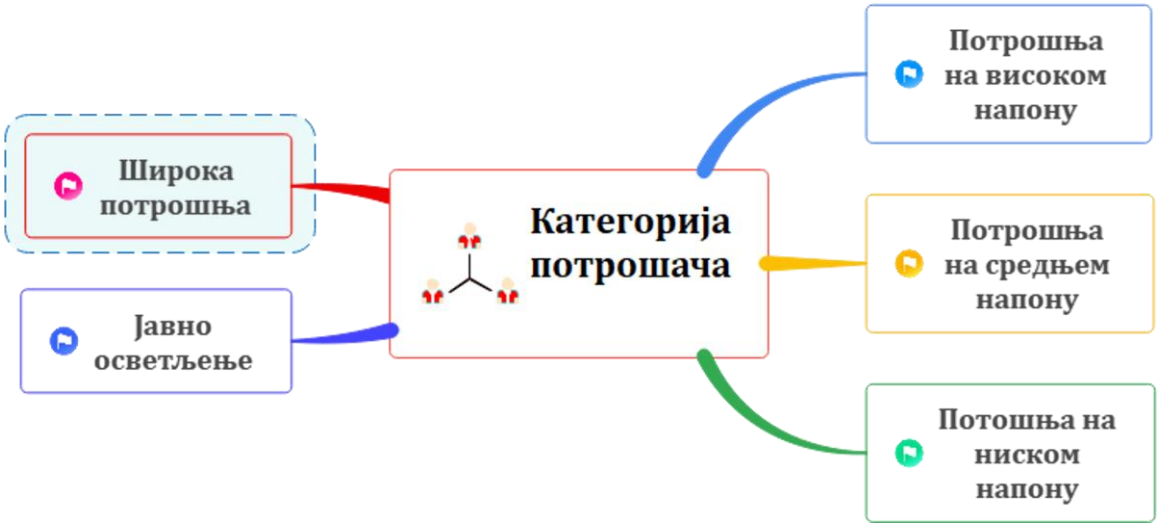
Per25 и *Per75* представљају 25-и *(Q1,* Слика 2-21*)* и 75-и *(Q3,* Слика 2-21*)* квартил, перцентил (енгл. *Percentile*). Па се према формули (32) вредности сваког обележја одузимају од медијане обележја и деле интерквартилним опсегом (енгл. *Interquartile Range, IQR).* Према слици (Слика 2-21) јасно је да је *IQR = Q3 - Q1*.



Слика 2-21: Подела дистрибуције обележја на квартиле [128]

2.13. Структура потрошача у систему за снабдевање електричном енергијом на посматраном (тестном) подручју

На посматраном подручју може се препознати неколико категорија и група потрошача. У зависности од номиналног напона мреже са које се предаје електрична енергија, начина мерења и других критеријума утврђених тарифним системом постоји пет категорија купаца (потрошача) електричне енергије [129] (Слика 2-22).



Слика 2-22: Основне категорије потрошача у електро мрежи Србије

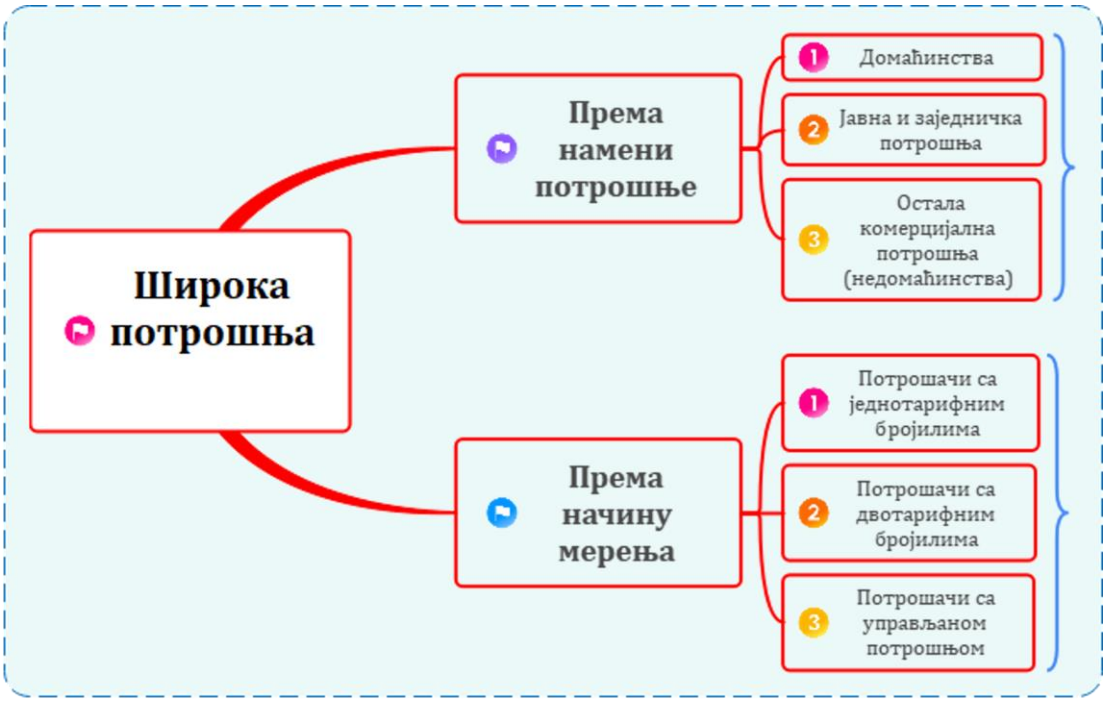
У складу са тим и у циљном скупу података се могу препознати исте категорије:

- "Потрошње на високом напону" обухвата потрошаче чији су објекти прикључени на електроенергетску мрежу напонског нивоа 110 kV или вишег напонског нивоа;
- "Потрошња на средњем напону" обухвата потрошаче чији су објекти прикључени на електроенергетску мрежу напонског нивоа изнад 1 kV, а нижег од 110 kV;
- "Потрошња на ниском напону" обухвата потрошаче чији су објекти прикључени на електроенергетску мрежу ниског напонског нивоа до 1 kV, односно имају електроенергетску сагласност на називну струју[4] прикључка већу од 63 A и којима се обрачунска снага, активна и реактивна енергија утврђују мерењем;
- "Широка потрошња" обухвата потрошаче чији су објекти прикључени на електроенергетску мрежу ниског напонског нивоа и који имају електроенергетску сагласност називне струје од највише 63 A, а користе електричну енергију за:
  - потребе домаћинства у становима, стамбеним зградама и објектима за одмор;
  - погон заједничких уређаја и инсталација у стамбеним зградама, заједничким и споредним просторијама (лифтови, осветљење степеништа и заједничких просторија и сл.);
  - погон уређаја и инсталација комуналне потрошње (котларнице за даљинско грејање, индивидуалне котларнице за грејање стамбених зграда и други уређаји и инсталације за снабдевање комуналним производима и услугама);
  - погон електромотора и апарата у пољопривредним домаћинствима, као и погон уређаја и инсталација кућних и заједничких сеоских водовода;
  - потребе осветљавања и загревања пословних објеката и пословних просторија и погон мотора и апарата у тим објектима и просторијама (објекти за комерцијални туризам, одмаралишта; објекти установа из области здравства, просвете, културе и дечје заштите, административни објекти и др.), као и погон заједничких уређаја и инсталација у тим објектима, заједничким и споредним просторијама (лифтови, осветљење степеништа и заједничких просторија, уређаји и инсталације комуналне потрошње и сл.);
  - потребе осветљавања споредних, економских објеката и гаража и прилаза тим објектима.
- "Јавно осветљење" обухвата потрошњу електричне енергије за осветљење улица, тргова, тунела, пешачких пролаза, паркова, путева, историјских и других обележја, уређаја за путну сигнализацију и друга потрошња за осветљење јавних површина и јавних објеката.

У овој дисертацији као циљна категорија узета је "Широка потрошња" која је обично најдоминантнија категорија потрошача на градској територији али уједно и најтежа за предикцију. Широка потрошња се даље може поделити по два критеријума (Слика 2-23):

- у зависности од начина мерења и
- у зависности од намене потрошње електричне енергије.

[4] Називна струја – максимална вредност струје која је по уговору одобрена потрошачу.



Слика 2-23: Групе потрошача широке потрошње

У зависности од начина мерења разликују се три основне групе потрошача:

- домаћинства, у којима се електрична енергија користи за потребе у становима (домаћинствима), стамбеним зградама и објектима за одмор, за потребе осветљавања помоћних објеката, економских објеката, гаража и прилаза као и за потребе пољопривредних домаћинстава;
- јавна и заједничка потрошња, где су купци установе које је основала држава или аутономна покрајина, локална самоуправа и
- остала комерцијална потрошња (у даљем тексту недомаћинства), где се електрична енергија користи за потребе обављања привредних и других делатности, осветљење и загревање пословних објеката и просторија чија намена није становање [129].

3. МЕТОДОЛОГИЈА ИСТРАЖИВАЊА

Ранијим приказаном прегледа доступне литературе указано је на оригиналност и актуелност природе предложеног истраживања. Увидом у референтна истраживања препознаје се да је већина радова базирана на краткорочном предвиђању потрошњи и то посматрајући одређене групе или категорије потрошача, на малом, ограниченом простору.
Основна идеја ове дисертације јесте решавање проблема предвиђања потрошњи електричне енергије потрошача на територији једног града и то на месечном нивоу.
На овај начин су обухваћени сви потрошачи, од малих потрошача какви су домаћинстава па све до оних који се баве неком производном делатношћу и самим тим имају неупоредиво веће потрошње. Поред тога, узима се у обзир зона у којој потрошачи живе, врста бројила на мерном месту, итд а све у циљу развоја модела који ће бити у стању да и у разноврсној структури потрошача препозна наизглед невидљиве шаблоне и обезбеди квалитетну процену потрошње за наредни месец.

3.1. Методе коришћене у истраживању

При изради дисертације а у оквиру целокупног спроведеног истраживања коришћено је више метода и приступа раду сходно проблему.

На самом почетку истраживања полази се од метода анализе и синтезе уз свеприсутни индуктивно-дедуктивни метод. За разумевање података и откривање међузависности обележја користи се више статистичких метода и приступа, као што су дескриптивна али и експликативна анализа, корелациона и регресиона анализа.

Претходно побројане методе засноване су на теоријско-епистемиолошким знањима стеченим кроз преглед доступне литературе. Теоријске парадигме о проблему истраживања и основна сазнајна начела о самој теми, послужила су као оријентир и главни путоказ кроз писање ове дисертације.

Целокупно истраживање заснива се на иницијалним, стварним подацима прикупљеним из званичних извора дистрибутивне компаније. Приказани случај требало би да постави добре темеље да се сличан поступак може спровести и на другим нивоима. Аналитички приступ у поступку интеграције обележја прикупљених из спољних извора са подацима садржаним у иницијалном скупу података, укључује и примену квантитативних, синтетичких и калкулативних метода.

Као инструмент у анализи података, израду модела и визуелну репрезентацију резултата користи се програмски језик Python уз подршку библиотека посебно дизајнираних за ML и анализу података.

Python је један од најпопуларнијих програмских језика за истраживање података [130], [131], [132]. Захваљујући веома активним програмерима и заједници отвореног кода, развијен је велики број библиотека намењених области машинског учења. Иако су перформансе интерпретираних језика, као што је Python, за рачунски интензивне задатке инфериорне у односу на програмске језике нижег нивоа, проширене библиотеке, као што су NumPy и SciPy, развијене су за надградњу Fortran и C имплементација нижег нивоа за брзе векторизоване операције у вишедимензионалним низовима [133].

3.2. Технике истраживања

У поступку прикупљања, обраде, анализе података и приказ добијених резултата користи се неколико техника:

- из званичне базе података дистрибутивне компаније прикупљени су подаци о потрошњама електричне енергије што чини иницијални скуп података;
- иницијални скуп података је адаптиран додатним обележјима прикупљеним из спољних извора, унетих хронолошким редом;
- поред обележја прикупљених из спољњих извора, иницијални сет је допуњен обележјима добијеним калкулативниим методама, на основу постојећих и накнадно прикупљених података;
- обрада и анализа података се врши помоћу програмских модула и статистичких процедура развијених у програмском језику Python (дескриптивна статистика, корелациона и регресиона анализа...);
- развој и евалуација развијених модела као и приказ резултата користе се библиотеке и пакети за подршку машинском учењу програмског језика Python.

3.3. Поступак истраживања

Целокупан поступак истраживања би се могао поделити у неколико фаза а у основи се разликују два приступа а самим тим и методологије:

I. предвиђање месечних потрошњи електричне енергије за све потрошаче на једном простору и
II. предвиђање месечних потрошњи електричне енергије у одређеном кластеру потрошача.

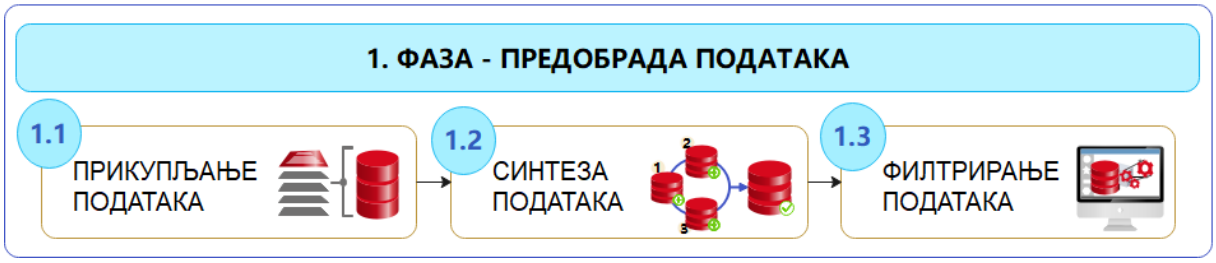
3.3.1. Фазе у истраживању

1. Прву фазу чине почетни методолошки поступци у процесу предобраде података и њихове припреме за наредне фазе (слично како је приказано и у [134]). Активности у првој фази су заједничке је за оба приступа (I и II).
   Познато је да на потрошњу електричне енергије утиче више фактора, почев од временских услова, доба године, врсте и категорије потрошача, зоне у којој се налази потрошач, итд. Било да делују истовремено или појединачно ови фактори имају последице и на њену процену (предвиђање). Управо из тих разлога је фаза предобраде података једна од најзначајнијих и има велики утицај на коначни исход – успешност модела. Због саме природе и сложености ова фаза се може разбити на неколико потфаза (1.1, 1.2, 1.3, Слика 3-1):
   1.1. *Прикупљање података*: Основна идеја је формирање универзалне методологије са акцентом на примени у реалним системима, за решавање стварних проблема. Да би се то постигло, ова потфаза обухвата активности на формирању сета података уз сагледавање свих потенцијалних извора. У складу са основном идејом, главни извор података у овом истраживању је званична евиденција потрошњи електричне енергије свих потрошача на подручју града (прва група података). Поред званичних података добијених од дистрибутивне компаније, а у циљу постизања што

прецизнијих резултата, потребно је податке проширити ослањајући се и на друге изворе. Ту се превасходно мисли на званичне базе података са записима о метеоролошким приликама и годишњим добима (друга група података). У завршници прве потфазе, на основу до сада прикупљених података, калкулативним методама долази се до треће групе података;

1.2 *Синтеза података из различитих извора у коначни сет*: Друга потфаза у оквиру препроцесирања обухвата синтезу података прикупљених у претходном кораку. Да би подаци прикупљени из различитих извора чинили целину са постојећим, раније прикупљеним подацима, посебна пажња се посвећује њиховој синтези. Раније је већ речено да основни сет (званичних) података садржи записе о потрошњама у току неколико година, па је јасно да прикључени подаци морају бити додати хронолошки, зависно од периода (месеца) на који се забележене потрошње односе. На тај начин се долази до сета података који улази у последњу фазу предобраде података;

1.3 *Филтрирање података:* Последња потфаза предобраде података обухвата даљи рад на скупу обележја произашлом из претходне потфазе. У овом кораку се врши селекција релевантних и одбацивање редудантних и ирелевантних обележја који ће бити улази у наредној фази;



Слика 3-1: Илустрација прве фазе у методологији

Резултати комплетне прве фазе (са потфазама) ће бити приказани у потпоглављу 4.1 а утицај адаптације сета испитан је у потпоглављу 4.3.2. Даље фазе у примењеној методологији се ослањају на правилно и успешно спроведену прву фазу чији ће детаљнији опис бити дат у наставку, засебно за оба раније поменута приступа (I и II).

I. Предвиђање месечних потрошњи електричне енергије за све потрошаче на једном простору (Приступ I)

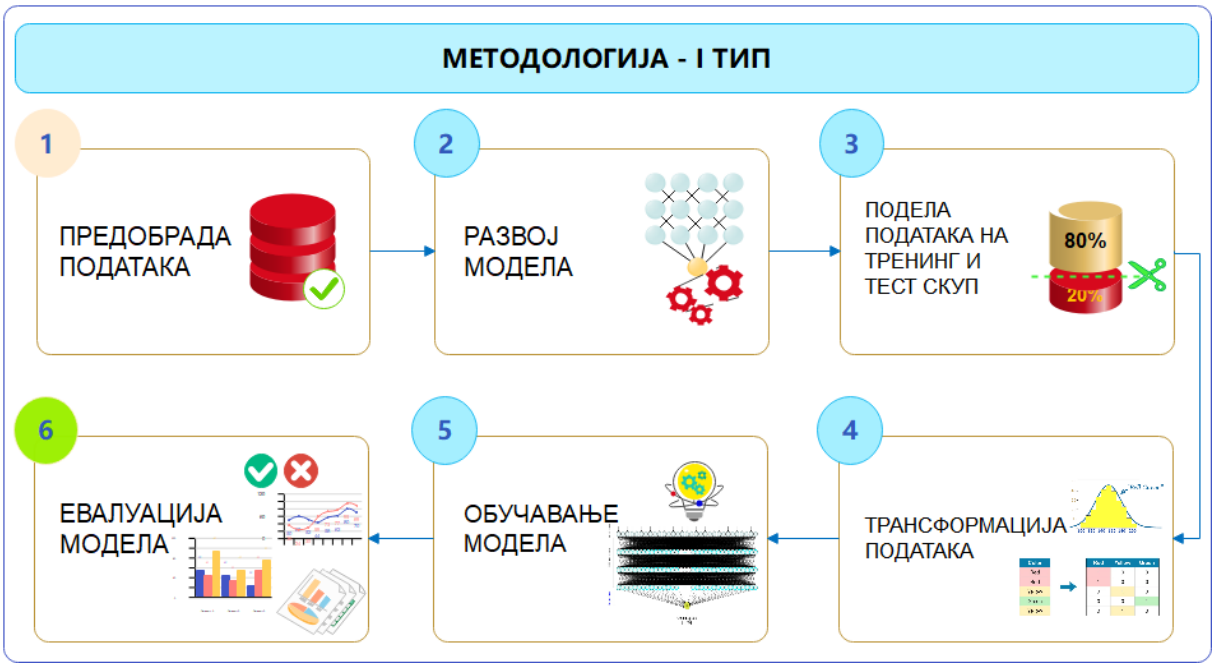
У општем случају предвиђање месечних потрошњи потрошача на територији једног града заснива се на посматрању свих потрошача на датом простору. У том случају након прве (1 – препроцесинг) фазе следи неколико нових фаза:

2. Друга фаза у овом случају обухвата активности на развоју модела машинског учења. Овде се превасходно мисли на одабир модела и подешавање (хипер)параметара. У конкретном случају, ова фаза подразумева развој и примену неколико модела заснованих на различитим техникама машинског учења (објашњеним у поглављу 2).
3. Трећа фаза наступа након одабира модела машинског учења, а пре његове примене. У овом кораку се врши подела података на два подсета: сет за тренирање

и сет за тестирање модела. У овом истраживању подела се врши у односу 80%:20%.

4. Како модели машинског учења могу да уче искључиво из нумеричких података, након формирања тренинг и тест скупа, у четвртој фази, приступа се процесу кодирања и скалирања података. Поступак кодирања категоријских обележја у нумеричка и поступак нормализације података спроведен је на начин који је описан у потпоглављу 2.12.
5. Пета фаза, која наступа одмах након поделе сета, подразумева процес обуке одабраног модела на скупу података одређеном за тренинг.
6. На крају у оквиру шесте фазе, помоћу метрика евалуације модела (побројаним у 2.11) врши се процена успешности модела и тестирање његових способности на скупу реалних, до тада моделу потпуно непознатих података (скуп за тестирање). У овој фази, врши се квантификација успешности формираног модела и на основу мера за оцену перформанси модела, врши одабир најповољнијег модела који ће у каснијим фазама бити усавршаван и тестиран у различитим условима.

Све побројане фазе (након друге) се понављају итеративно. Рад на изради архитектуре модела, посебно модела ANN се не завршава почетком фазе тренинга. Тестирањем модела утврђују се његове јаке и слабе тачке које се затим покушавају прилагодити поновним итерацијама, па се границе између ових фаза постављају више теоријски. Побројане фазе се могу представити графички као на следећој слици (Слика 3-2).



Слика 3-2: Илустрација фаза у оквиру методологије за предвиђање месечних потрошњи електричне енергије за све потрошаче

II. Предвиђање месечних потрошњи електричне енергије за одређену категорију потрошача (кластер) (Приступ II)

Још један од битних аспеката формирања модела односи се на његову способност да се прилагођава различитом обиму, тј. групама потрошача за које се врши предвиђање.

Посебно је погодно предложени модел тестирати не само над скупом података у целости већ и над најпре груписаним (кластерованим) подацима. Значај таквог приступа се истиче посматрањем реалног система и хипотетичким постављањем ситуације. У том погледу се јасно изводи закључак да је сам поступак формирања хомогених група потрошача доста једноставнији и у пракси доста јефтинији од поступка обраде целокупног скупа података без обзира о којој техници машинског учења је реч. Утицај кластеровања потрошача на предвиђање потрошњи електричне енергије биће испитан у потпоглављу 4.4.

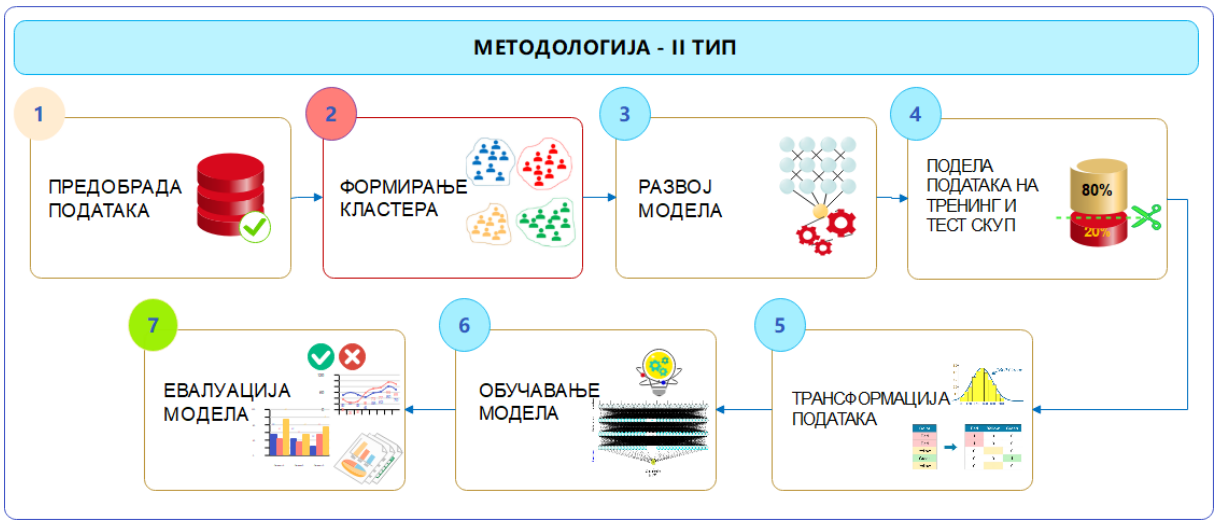
Увидом у структуру потрошача на посматраном подручју (представљеној у оквиру поглавља 2.13), уочава се неколико могућих сегмената који могу послужити као основ за кластеровање потрошача. Очекивано је да се на тај начин могу извршити још прецизнија предвиђања месечних потрошњи у односу на она добијена поступком из I.

Код оваквог приступа, фазе након прве се померају додајући другу фазу (2.) у којој се најпре врши кластеровање потрошача: према месецу у коме се потрошња остварује, годишњем добу, категорији потрошача или зони у којој потрошач живи.

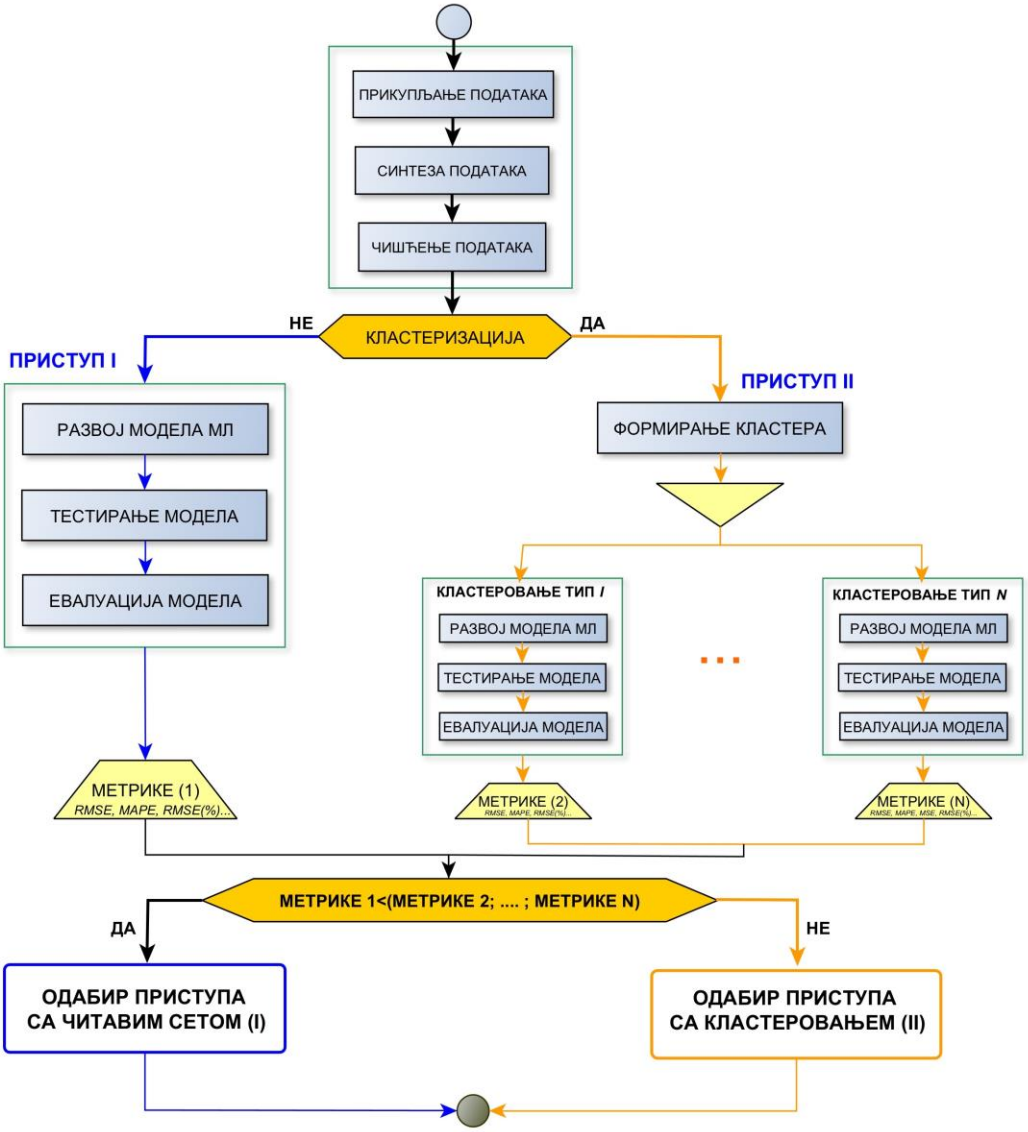
Овако добијени кластери се даље посматрају као потпуно независни сетови података. Као такви, подлежу свим фазама приказаним у оквиру првог приступа (I), као и сет који обухвата све потрошаче на посматраном подручју.

3. Након извршеног кластеровања, у наредној, сада трећој фази, врши се одабир модела машинског учења.
4. Четврта фаза обухвата активности на подели скупа података (кластера) на тренинг и тест скуп.
5. У петој фази се приступа процесу кодирања категоријских обележја а потом и скалирању свих обележја.
6. У шестој фази се врши тренинг одабраног модела.
7. У седмој фази се врши евалуација модела кроз претходно одабране метрике.

Методологија приказана у оквиру II би се могла графички приказати као на слици испод (Слика 3-3), док се комплетан поступак рада у истраживању може представити дијаграмом приказаном на слици иза (Слика 3-4).



Слика 3-3: Илустрација фаза у оквиру методологије за предвиђање месечних потрошњи електричне енергије за одређени кластер потрошача



Слика 3-4: Дијаграм тока истраживања

На основу предложене методологије и резултата који из ње проистичу биће испитани утицаји појединих аспеката а према постављеним хипотезама:

- Нерадни дани (празници) би теоријски могли знатно да утичу на потрошње електричне енергије. У потпоглављу 4.4.7 биће испитан њихов утицај.
- Резултати тестирања модела за подршку класификацији биће приказани кроз потпоглавље 4.5.
- У поглављу 4.6 биће представљен нацрт апликације за подршку предвиђању потрошње електричне енергије.

Резултати добијени на основу предложене методологије и приступа треба да омогуће најпре одабир најпоузданијег модела као основе система за предвиђање потрошње електричне енергије што ће даље осигурати и олакшање процеса управљања истом. Тако би резултати ове дисертације могли имати примену у широком спектру апликација, укључујући управљање потрошњом електричне енергије у домаћинствима и индустрији као и примене у паметним градовима и системима за управљање енергијом.

4. РЕЗУЛТАТИ ИСТРАЖИВАЊА

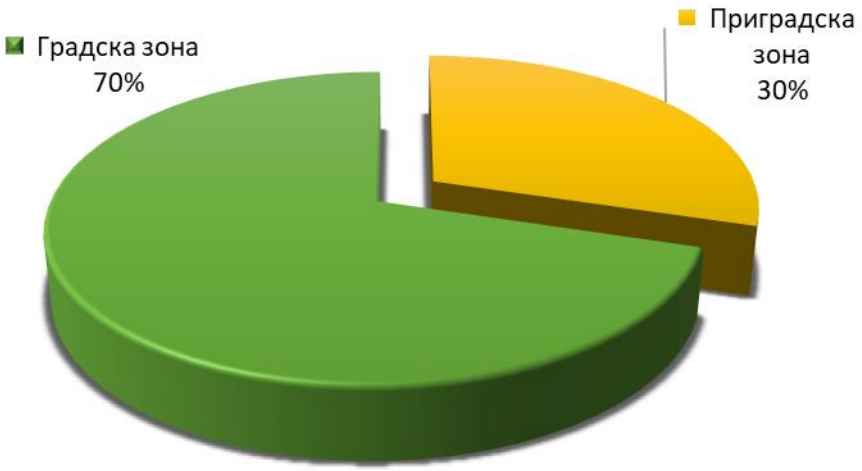
Сходно претходно описаном поступку (методологији) истраживања у наставку ће бити приказани резултати које ове две методологије дају.

Најпре ће кроз потпоглавље 4.1 бити испитан поступак препроцесирања података и утицај адаптације сета који се односи на прву фазу.

4.1. Препроцесирање улазних података

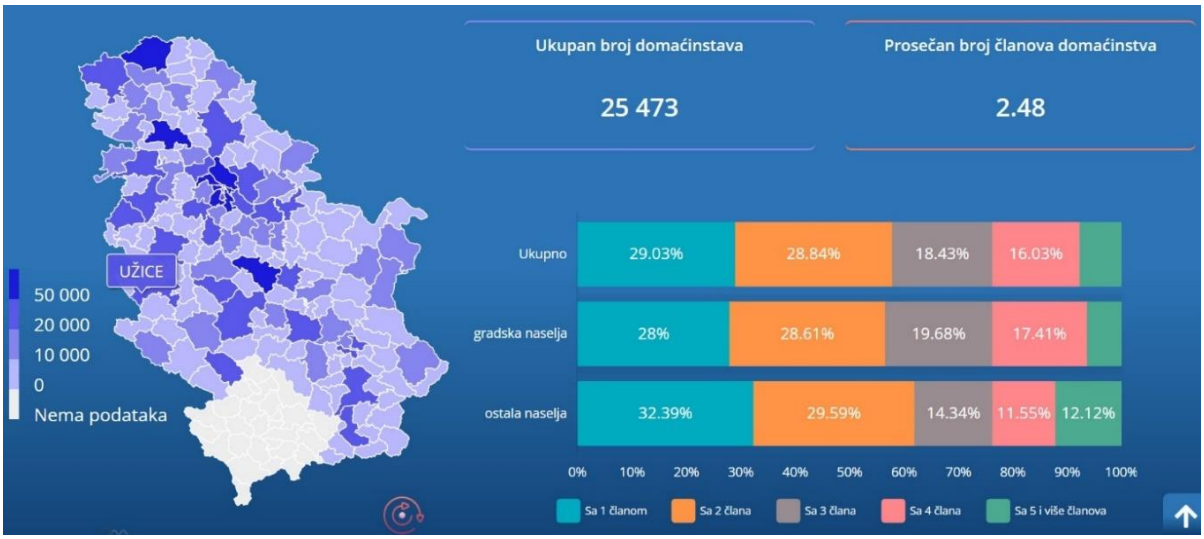
4.1.1. Прикупљање података и карактеристике скупа улазних података

У скупу података су садржани записи о потрошњама електричне енергије свих постојећих потрошача на територији града Ужица, који се чувају у бази података јавног предузећа ЕДС, Огранак Ужице. Подаци се односе на период од четири године и осам месеци и садржи укупно 1.845.537[5] записа за 40.548 мерних места (бројила). Према подацима из циљне базе података о потрошњама електричне енергије део потрошача се налази у приградској средини (означени као „приградска зона“ - 30% укупног броја мерних места), али је већина и даље настањена у урбаном делу града (означени као “градска зона” - 70% мерних места). Слика 4-1 даје графичку презентацију потрошача на овом простору, док су, према најновијим подацима Републичког завода за статистику [135], ове бројке нешто другачије. Према подацима Завода, на подручју града Ужица 24% становника живи у приградским насељима док је већина (76%) у градској средини (Слика 4-2).



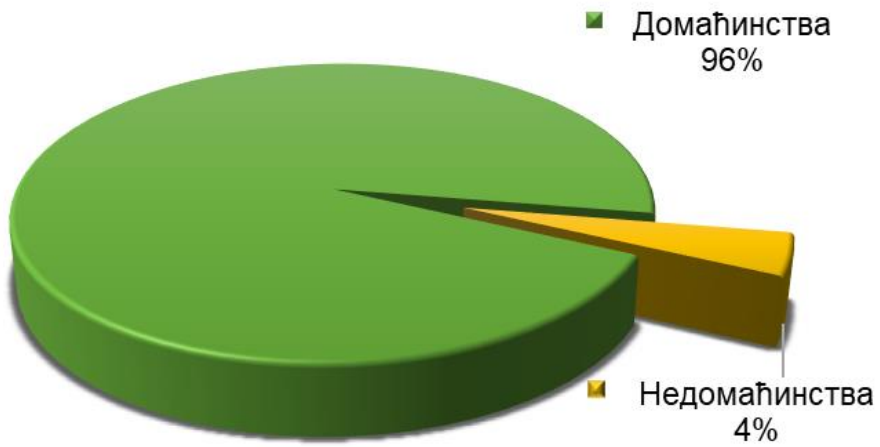
Слика 4-1: Однос броја потрошача према зони у којој живе

[5] Овај број не представља број мерних места већ сва појављивања различитих *потрошача* у сету података. Касније ће доћи до редукције броја мерења приликом чишћења података и уклањања редудантних записа.



Слика 4-2: Статистички подаци: однос броја домаћинстава у градској и приградској зони на територији града Ужица [135]

Даљим увидом у податке из тестног сета, на овом подручју чак 96% бројила се налази у домаћинствима[6] док свега 4% чине недомаћинства[7] (Слика 4-3). Потрошачи из домена јавне и заједничке потрошње (објашњени у поглављу 2.13) нису предмет истраживања.

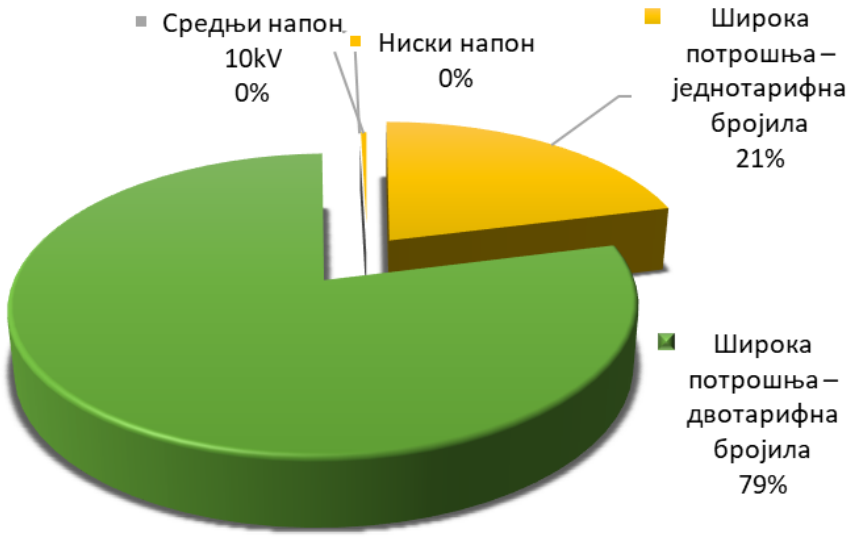


Слика 4-3: Однос броја потрошача према категорији потрошача

Посматрано према групи потрошача, сви потрошачи на посматраном подручју се могу разврстати у пет група: *Широка потрошња – једнотарифна бројила* – 21%, *Широка потрошња – двотарифна бројила* - 79% и са по мање од једног процента Средњи напон 10kV и *Ниски напон* (Слика 4-4).

[6] Под термином *домаћинства* у бази података подразумевају се потрошачи који живе и у кућама и становима.

[7] Под термином *недомаћинства* у бази података су обележени сви други потрошачи (сви они који електричну енергију не користе за потребе домаћинства).



Слика 4-4: Однос броја потрошача према групи

У циљном скупу података, различите групе и категорије потрошача узрокују и великим распоном потрошњи, израженим у kWh. Скуп података обухвата све потрошаче, од малих потрошача, који често могу представљати и куће за одмор које се користе само у појединим деловима године, па до великих потрошача који се баве различитим врстама производње и самим тим имају јако велике месечне потрошње.

4.1.2. Адаптација иницијалног скупа обележја

Скуп обележја које дистрибутивна компанија бележи у својој бази и који су доступни на почетку истраживања тичу се основних карактеристика мерног места сваког потрошача и обухватају следеће карактеристике:

- ознака потрошача,
- категорија потрошача,
- потрошачка група,
- зона у којој потрошач живи (зона мерног места),
- обрачунски период и
- остварена месечна потрошња.

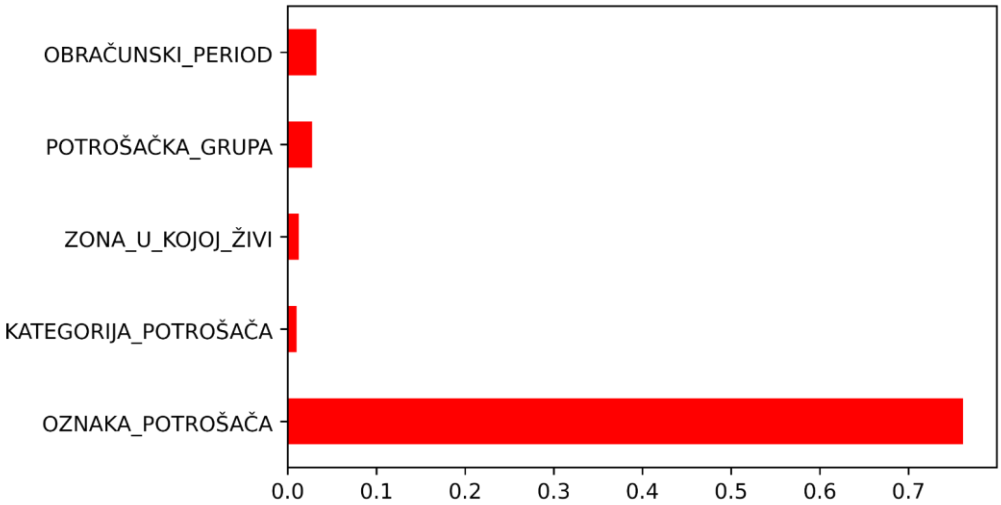
У почетној фази истраживања, постојали су покушаји да се проблем процене потрошње електричне енергије врши помоћу једноставних, лако применљивих техника машинског учења (попут линеарне регресије и стабала одлуке, који су описани у поглављу 2), али први резултати нису били охрабрујући. У Табели 1 су дати резултати ових покушаја кроз неколико мера евалуације модела који ће бити коришћени и касније. Иако први резултати не делују охрабрујуће, биће искоришћени као контролне вредности за испитивање утицаја како адаптације сета тако и за унапређење модела.

Табела 1: Резултати добијени употребом неколико техника машинског учења

Модел	**RMSE (%)**	**MAE (%)**	**MAPE (%)**	**AR^2 (%)**
Линеарна регресија	42,30	27,82	83,67	64,16
Градијент буст (*XGBoost)*	40,14	25,49	68,44	70,43
Линеарни SVR	43,09	26,99	73,14	65,92
Хубер регресија	42,80	27,03	75,26	70,43
Стабла одлуке	53,50	34,32	71,67	47,47
Тестни модел ANN	39,99	25,39	65,50	72,87

Већ у овој фази истраживања постаје јасно да обележја доступна у иницијалном скупу података нису довољна да би се један овакав проблем могао решавати техникама машинског учења.

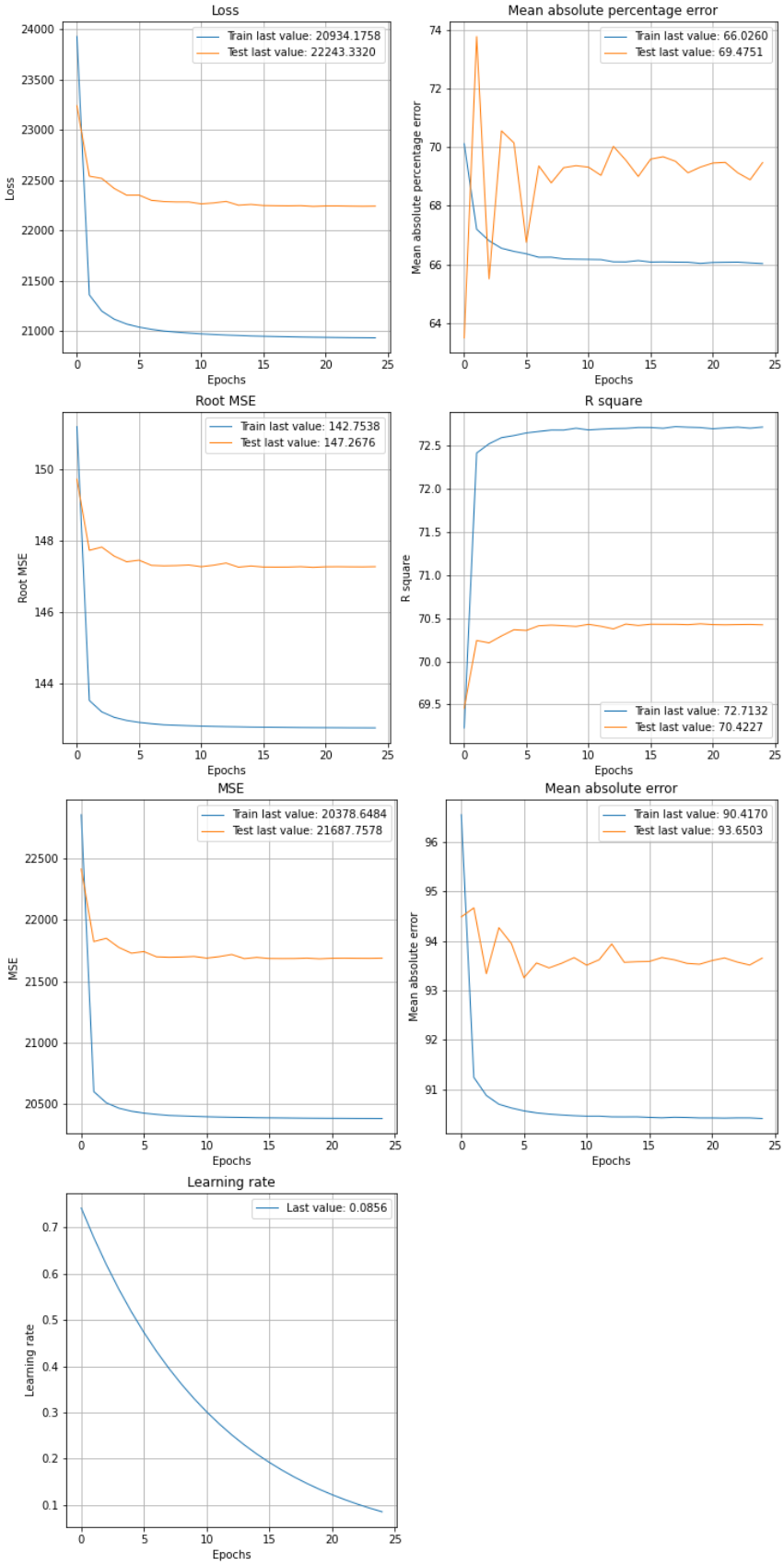
Провером утицајности (важности) обележја у скупу података (Слика 4-5) ово је и потврђено. Доступна обележја нису довољно садржајна да би се из њих исцрпело потребно знање и да би се резултати могли користити за решавање реалног проблема. Такође, резултати у Табели 1 показују да највећи потенцијал у решавању овог проблема има тестни ANN модел па је будући рад базиран управо на овом моделу уз поређење са горе наведеним моделима.



Слика 4-5: Важност обележја у иницијалном скупу података

Иако са највећим потенцијалом, чак ни резултати које даје модел ANN не могу се окарактерисати као довољно добри. Евалуацијом модела и праћењем метрика током процеса обуке и тестирања модела (Слика 4-6) добија се увид у неефикасност и недовршеност проузроковану мањком коректних обележја.

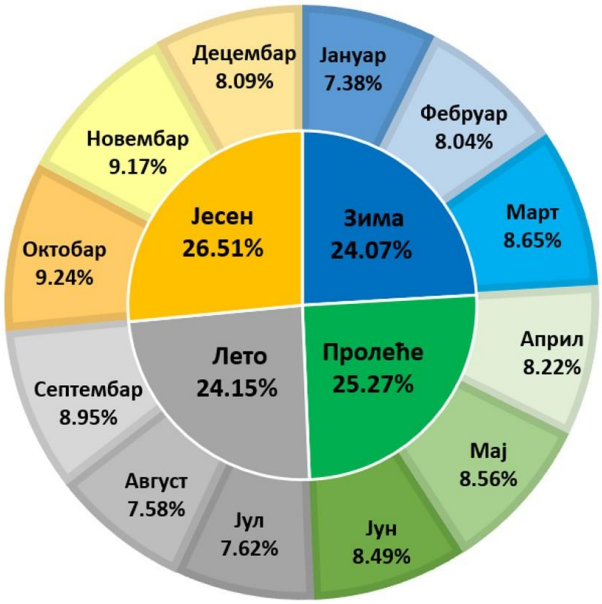
Овим се побија општа хипотеза постављена на почетку истраживања, јер на основу података о потрошњама електричне енергије потрошача које бележи дистрибутивна компанија, није могуће прецизно предвидети будуће потрошње постојећих или потрошње нових потрошача са сличним карактеристикама.



Слика 4-6: Дијаграми праћених метрика током процеса обучавања полазног модела ANN на иницијалном скупу података

У циљу добијања прецизнијих резултата неопходно је постојећи скуп обогатити додатним обележјима која се не тичу сваког појединачног потрошача већ амбијента и окружења у коме потрошачи живе, односно где се мерна места налазе. Обележја која се том приликом додатно укључују у иницијални скуп података, преузета су из поузданих извора, хронолошки за сваки месец у посматраном периоду:

- метеоролошки подаци (атрибути: *„Просечно трајање обданице у сатима“, „Број сунчаних сати у месецу“, „Просечно трајање (број) сунчаних сати“, „Број облачних дана“, „Број ведрих дана“, „Најнижа месечна температура“, „Највиша месечна температура“, „Број дана у месецу са падавинама“, „Падавине у mm“, „Влажност ваздуха у %“, „Просечна месечна температура“)* преузети су са званичне интернет презентације хидрометеоролошког завода Србије [136];
- подаци о броју радних и нерадних дана (атрибути: „*Број дана празника“, „Број дана викенда“* и *„Број радних дана“*) са странице [137];
- подела на годишња доба на основу климатолошке поделе према [138] (атрибут: *„Годишње доба“,* Слика 4-7);



Слика 4-7: Подела месеци на годишња доба са процентуалним уделом у потрошњи

- „*Зона потрошње“* се одређује на основу броја потрошених kWh [2], [139]. Такође, атрибут „*Зона потрошње у претходном месецу“* је добијен на исти начин само посматрајући потрошњу коју је потрошач имао у претходном месецу. Потрошње мање од 350 kWh се сврставају у зелену зону, од 350 до 1600 kWh у плаву зону и потрошње веће од 1600 kWh у црвену зону;
- атрибут „*Просечна потрошња“* је добијен израчунавањем просечне потрошње за сваког потрошача респективно;
- атрибут „*Тренд потрошње потрошача“* је добијен помоћу линеарне функције за сваког појединачног потрошача, методом најмањих квадрата (једначина (33)), а

на основу забележених потрошњи које конкретни потрошач има у свим посматраним месецима;

$$y = mx + b\,; \tag{33}$$

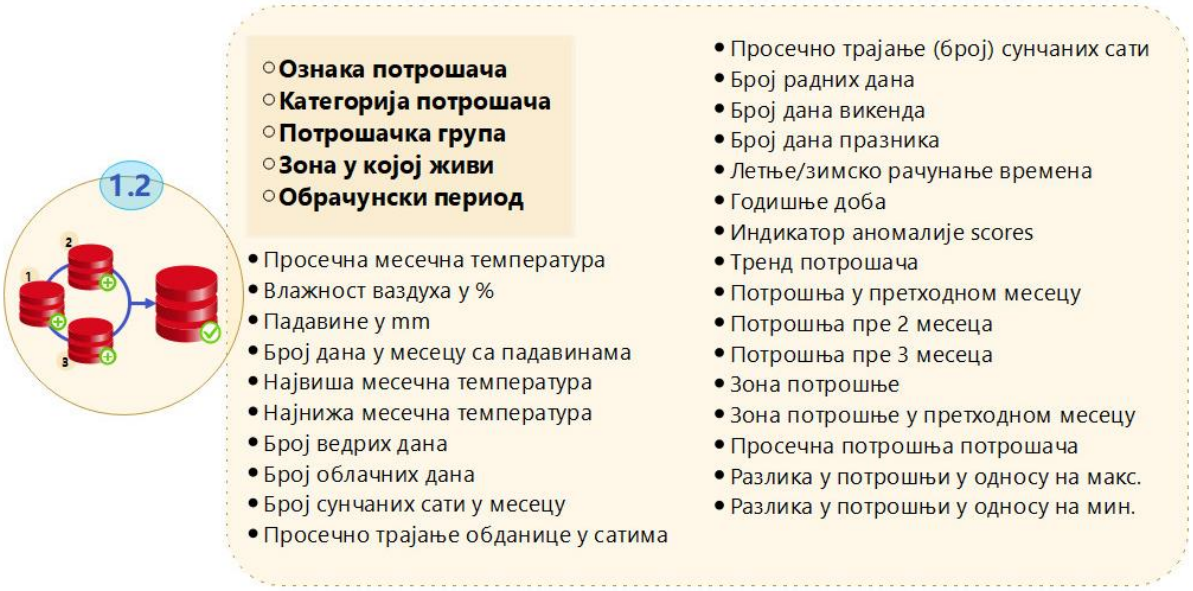
- атрибут „*Индикатор аномалије – scores“* је добијен методом заснованом на случајним шумама и колекцији бинарних стабала одлучивања, изолационим шумама (енгл. *Isolation Forest*), према [125]–[127]. То је касније послужило и за откривање аномалија у самом скупу, јер добијена вредност представља резултат (енгл. *Score*) колико је нека инстанца аномалија или не. Вредност овог обележја је дефинисана једначином (34) и има вредности од 0 до 1, при чему се вредности близу нуле сматрају аномалијом:

$$s\left(x,n\right) = 2^{-\frac{E\left(h\left(x\right)\right)}{c\left(n\right)}}\,, \tag{34}$$

где је:

  - *h(x)* дужина путање од чвора у корену до крајњег спољашњег чвора *x*;
  - *E(h(x))* је средња вредност од *h(x);*
  - *c(n)* је просек од *h(x)* за дато *n* (*c(n)* је коришћен за нормализацију *h(x)*);
- атрибути *„Разлика у потрошњи у односу на максималну“* и *„Разлика у потрошњи у односу на минималну“* су добијени као разлика од максималне и минималне потрошње коју су у посматраном периоду имали сваки од потрошача појединачно.

Овим се завршавају прве две фазе истраживања чији резултат је формиран скуп обележја (Слика 4-8). Потом следи фаза препроцесирања полазећи од селекције релевантних, ирелевантних и нерелевантних података.



Слика 4-8: Коначни скуп обележја формиран у првој фази истраживања

4.1.3. Кодирање и нормализација улазних података

Расподела нумеричких обележја циљног скупа података дата је на следећој слици (Слика 4-9) и показује да су подаци доста неуједначени. Ово наглашава потребу да у фази предобраде подаци буду скалирани.

Слика 4-9: Расподела вредности на нивоу сваког нумеричког обележја

Над циљним скупом података у овој дисертацији, ради одабира најповољнијег, тестирано је неколико типова скалирања података као и поступака кодирања категоричких података. За кодирање категоријских обележја користи се кодирање сваке категорије обележја помоћу 0 и 1 (*One-Hot Encoding*) и кодирање засновано на средњој вредности циљне варијабле како је објашњено у поглављу 2.12.

На следећим сликама дат је приказ података у изворном облику (Слика 4-10), након кодирања категоријских обележја (Слика 4-11) и финални облик података спремних за улаз у модел (након извршеног кодирања и скалирања) (Слика 4-12).

Кôд који прати поступак адаптације сета у фази препроцесирања, заједно са подацима коришћеним за адаптацију, дати су у виду прилога (Прилог 1 - 5).

1	2	3	4	5	6	7	8	9	10	11	12	13
CONSUMER	CONSUMER_CATEGORY	ZONE	CONSUMER_GROUP	CalculationPeriod	Seasons	AVG_TEMPERATURE	humidityIn%	precipitationIn_mm	NOfDwithPrecipitation	MAXtemp	MINtemp	NoOfClearDays
CONSUMER_1	Household	City	Single-rate-Large scale consumption	april 2014.	spring	10	79	156	18	21.3	-0.7	0
CONSUMER_2	Non-household	City	Single-rate-Large scale consumption	may 2014.	spring	13	76	218	19	25.4	0.5	2
CONSUMER_3	Household	City	Single-rate-Large scale consumption	june 2014.	spring	17	71	125	17	28	7	5
CONSUMER_4	Household	City	Single-rate-Large scale consumption	april 2015.	spring	9	61	53	7	23	-4.5	6
CONSUMER_5	Household	City	Single-rate-Large scale consumption	may 2015.	spring	16	65	52	11	30	4.9	4
CONSUMER_6	Household	City	Single-rate-Large scale consumption	june 2015.	spring	17	67	102	14	29	6.5	4
CONSUMER_7	Non-household	City	Low voltage	april 2016.	spring	13	60	46	11	26.2	-1.2	7
CONSUMER_8	Non-household	Countryside	Medium voltage 10KV	may 2016.	spring	13	70	151	19	27.3	1	4
CONSUMER_9	Non-household	City	Medium voltage 10KV	june 2016.	spring	19	71	175	18	29.2	9.6	2
CONSUMER_10	Non-household	Countryside	Low voltage	april 2017.	spring	9	68	98	11	23.4	-3.5	3
CONSUMER_12	Household	City	Single-rate-Large scale consumption	may 2017.	spring	14	73	100	16	25.4	2	5
CONSUMER_13	Household	City	Low voltage	june 2017.	spring	20	65	28	9	31	8.5	6
CONSUMER_14	Household	Countryside	Dual-rate-Large scale consumption	april 2018.	spring	15	61	24	10	26	-0.2	9
CONSUMER_17	Household	City	Single-rate-Large scale consumption	may 2018.	spring	17	69	76	13	26.4	6.2	5
CONSUMER_18	Household	City	Single-rate-Large scale consumption	june 2018.	spring	18	76	130	21	30.2	6.6	0
CONSUMER_19	Household	City	Dual-rate-Large scale consumption	april 2014.	spring	10	79	156	18	21.3	-0.7	0
CONSUMER_20	Household	Countryside	Dual-rate-Large scale consumption	may 2014.	spring	13	76	218	19	25.4	0.5	2
CONSUMER_21	Household	Countryside	Dual-rate-Large scale consumption	june 2014.	spring	17	71	125	17	28	7	5
CONSUMER_22	Non-household	Countryside	Low voltage	april 2015.	spring	9	61	53	7	23	-4.5	6
CONSUMER_23	Non-household	Countryside	Dual-rate-Large scale consumption	may 2015.	spring	16	65	52	11	30	4.9	4
CONSUMER_24	Non-household	Countryside	Medium voltage 10KV	june 2015.	spring	17	67	102	14	29	6.5	4
CONSUMER_25	Non-household	Countryside	Dual-rate-Large scale consumption	april 2016.	spring	13	60	46	11	26.2	-1.2	7
CONSUMER_26	Household	City	Dual-rate-Large scale consumption	may 2016.	spring	13	70	151	19	27.3	1	4
CONSUMER_28	Household	City	Dual-rate-Large scale consumption	june 2016.	spring	19	71	175	18	29.2	9.6	2
CONSUMER_29	Household	City	Dual-rate-Large scale consumption	april 2017.	spring	9	68	98	11	23.4	-3.5	3
CONSUMER_30	Household	City	Dual-rate-Large scale consumption	may 2017.	spring	14	73	100	16	25.4	2	5
CONSUMER_32	Household	City	Dual-rate-Large scale consumption	june 2017.	spring	20	65	28	9	31	8.5	6
CONSUMER_33	Household	City	Dual-rate-Large scale consumption	april 2018.	spring	15	61	24	10	26	-0.2	9
CONSUMER_34	Household	City	Dual-rate-Large scale consumption	may 2018.	spring	17	69	76	13	26.4	6.2	5
CONSUMER_35	Household	City	Dual-rate-Large scale consumption	june 2018.	spring	18	76	130	21	30.2	6.6	0
CONSUMER_36	Household	City	Dual-rate-Large scale consumption	april 2014.	spring	10	79	156	18	21.3	-0.7	0
CONSUMER_37	Household	City	Dual-rate-Large scale consumption	may 2014.	spring	13	76	218	19	25.4	0.5	2
CONSUMER_38	Household	City	Dual-rate-Large scale consumption	june 2014.	spring	17	71	125	17	28	7	5
CONSUMER_39	Household	City	Dual-rate-Large scale consumption	april 2015.	spring	9	61	53	7	23	-4.5	6
CONSUMER_41	Household	City	Dual-rate-Large scale consumption	may 2015.	spring	16	65	52	11	30	4.9	4

Слика 4-10: Подаци у изворном облику (насумични извод из скупа података)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
CONSUMER	CONSUMER	CONSUMER_	CONSUME	CONSUMER	Calculat	Calculati	Calculat	Calculati	Calculatio	Calculati	Calculati	Calculatio	CalculationP	Calc
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	1	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	1	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	1	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	1	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	0
360.6607142	0	0	0	1	0	0	0	0	0	0	0	0	0	1
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	1	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	1	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	1	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	1	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	1	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	1	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	1	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	1	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0
334.1090909	1	0	0	0	0	0	0	0	0	0	0	0	0	0

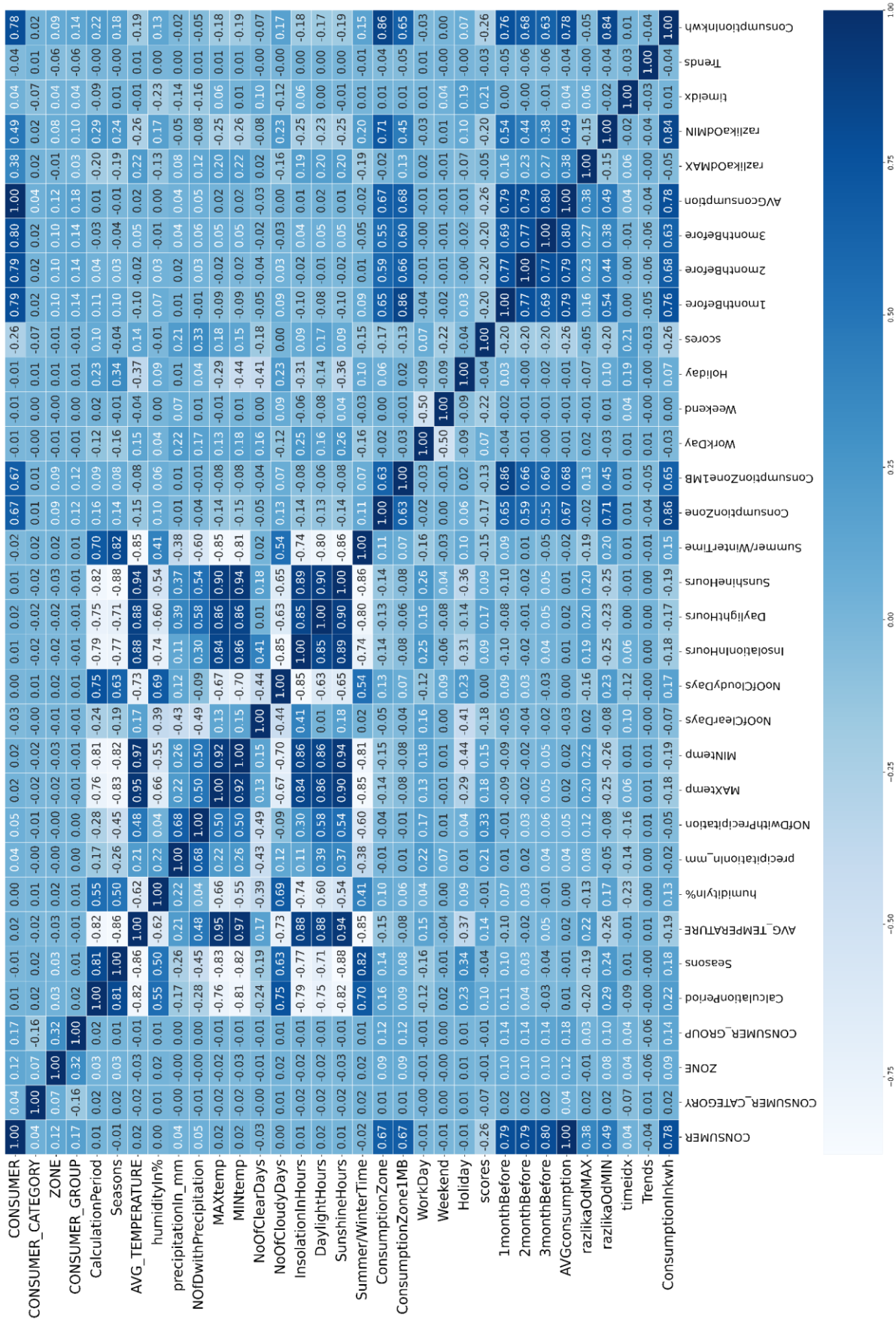
Слика 4-11: Подаци након кодирања (насумични извод из скупа података)

1	2	3	4	5	6	7	8	9	10	11	12	13
CONSUMER	CONSUMERZONE		CONSUMER_GROU	CalculatiSeasons		AVG_TEMPER	humidityIn%	precipitatic	NOfDwitMAXtemp		MINtemp	NoOfClearDays
0.143193160.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
-0.467958500.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.392144600.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.348518400.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.154281740.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.451561900.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.135022270.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.855599600.0		0.0	-1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.880254770.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.090360000.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.08829984-1.0		1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.161163860.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
-0.259565800.0		0.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.903374970.0		0.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.386893440.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
-0.726205300.0		0.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.675588410.0		0.0	-1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
-0.923327600.0		0.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.422279550.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
0.675804710.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.416537600.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2.059728770.0		0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
-0.182137700.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.346338960.0		0.0	-1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1.996976811-1.0		1.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.206471090.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.490993370.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.389406711.0		1.0	-1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.273673230.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Слика 4-12: Подаци након кодирања и скалирања (насумични извод из скупа података)

Још један значајан осврт на податке, са циљем идентификације образаца и веза између обележја, дат је визуелном корелационом матрицом (Слика 4-13). На сваком пољу укрштања два обележја колорном скалом (топлотном мапом) али и вредносно приказана је повезаност међу обележјима. Најсветлије нијансе боје на колорној мапи имају поља са најнижом – негативном корелацијом док најтамнија поља имају највишу позитивну корелацију. Иако је очекивано, и претпостављено првом посебном хипотезом да временске прилике (количина падавина, сунчани и облачни дани, просечна месечна температура итд) имају кључну улогу и велики утицај на потрошње електричне енергије, а самим тим и на процес њеног предвиђања, корелациона матрица показује другачију слику. Сваки од ових обележја има благу (занемарљиву) корелацију што доводи у питање њихово увођење у скуп обележја. Међутим, емпиријски се долази до закључка да и таква обележја имају утицаја на крајњи исход. Током тренинга ML модел проналази значајне везе између обележја које се голим оком и корелационом анализом не могу директно утврдити.

Фазе истраживања од 2-5 према постављеној методологији у поглављу 3.3.1 детаљније ће бити описани у наредним поглављима.



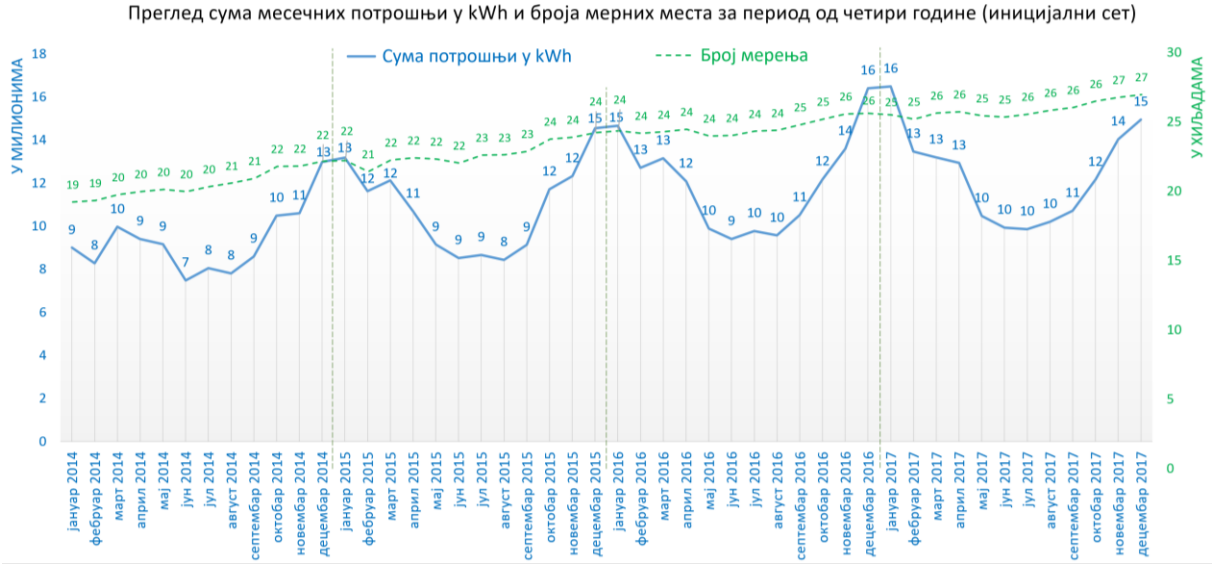
Слика 4-13: Корелација обележја у скупу података дата преко топлотне мапе

4.2. Истраживање података. Откривање законитости у циљном скупу података

Непосредно пре развоја модела за предвиђање потрошњи електричне енергије потребно је детаљно истражити податке у скупу података. Како су у скупу садржани подаци о потрошњама за четири године и осам месеци, ради лакше упоредивости података, записи о потрошњама за последњих осам месеци ће бити уклоњени и скуп ће бити посматран за оквир од 4 x 12 месеци.

Најбољи увид у структуру података се постиже њиховом графичком презентацијом. У наставку ће бити приказане расподеле потрошњи за све потрошаче на простору града Ужица.

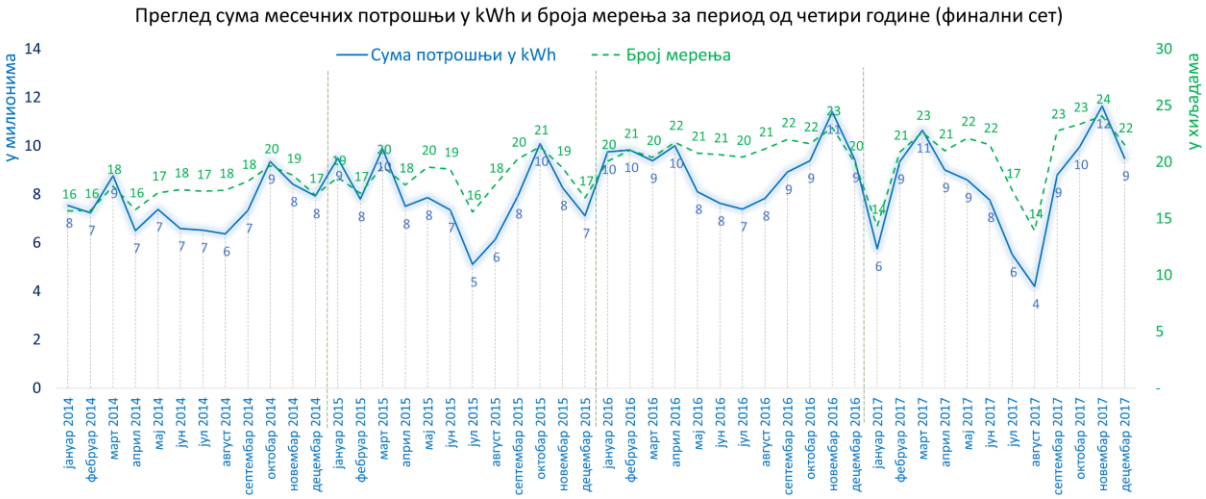
Слика 4-14 даје приказ укупне потрошње електричне енергије (плава линија, изражено у милионима kWh) и броја мерних места – мерења, очитавања бројила (зелена линија, изражено у хиљадама) на подручју града Ужица за период од четири године (посматране хронолошки). Дијаграм илуструје податке садржане у иницијалном скупу.



Слика 4-14: Укупна потрошња и број мерних места по месецима за период од четири године (иницијални скуп)

Посматрајући вредности слева на десно датог дијаграма (поредећи почетак и крај посматраног периода) може се приметити раст броја потрошача који није праћен истим трендом у погледу остварених потрошњи. Детаљним увидом у податке откривен је одређен број потрошача који у највећем броју месеци нема забележене никакве потрошње. Најчешће је реч о потрошачима који су одјавили бројило али дистрибутивна компанија таква мерна места не уклања из своје базе у одређеном периоду па су због тога и овде присутни. Поред тога, у иницијалном скупу су заступљени и потрошачи из неколико категорија које нису од значаја за предвиђање потрошњи и представљају најчешће велике потрошаче. Једна таква категорија је „Јавна расвета“.

Након чишћења података линије на дијаграму добијају другачији изглед (Слика 4-15) описујући финални скуп који ће бити коришћен за даље истраживање.

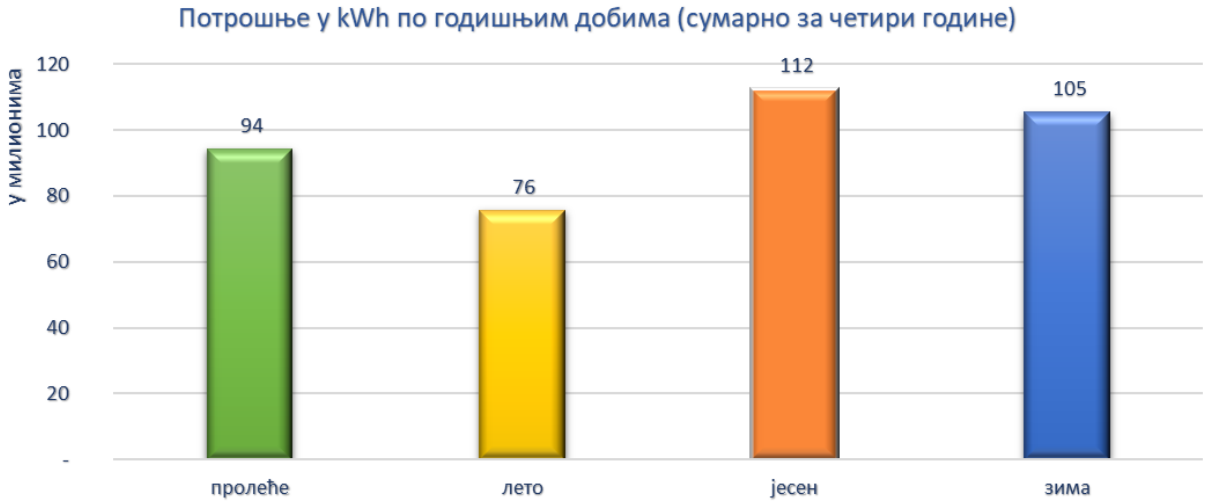


Слика 4-15: Укупна потрошња и број мерних места по месецима за период од четири године (финални скуп) [54]

Слика 4-15 даје, на први поглед, шаблонски распоред вредности и тиме наговештава да се потрошње могу пратити сезонски. Као што је показано у [143] потрошње су нешто веће у зимским него у летњим месецима. И даље је приметан благи пораст броја мерних места према крају обрачунског периода што је последица прикључивања нових потрошача на мрежу.

Месеци у којима се бележе оштри падови на обе линије представљају месеце када се до одређеног броја потрошача није могло доћи услед лоших временских прилика или одсуства власника што је условило неочитавање ових бројила. У таквим ситуацијама се потрошња обележава са нулом што ће се одразити на потрошњу у следећем месецу или месецима. Посматрано из угла појединца, то узрокује скокове на дијаграму потрошње и може створити привид повећане потрошње што додатно отежава поступак предвиђања.

Како би разлике (или сличности) између месеци и годишњих доба биле јасније, на следеће две слике дат је приказ сумарних потрошњи за све потрошаче по годишњим добима и месецима у години (Слика 4-16 и Слика 4-17).

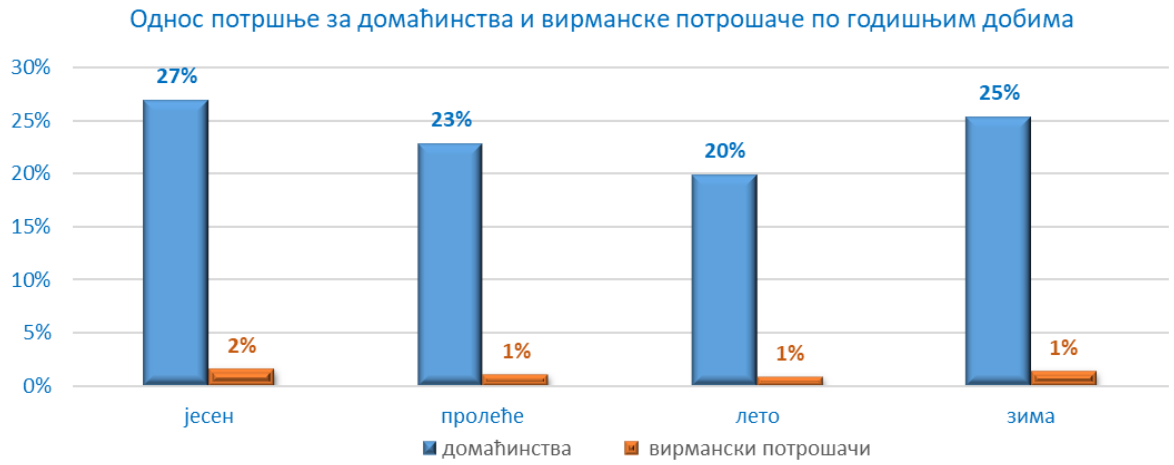


Слика 4-16: Укупна потрошња по годишњим добима (посматрано сумарно за четири године)

Слика 4-17: Укупна потрошња по месецима у години (посматрано сумарно за четири године)

Приметно је да се у току јесени и зиме (од октобра до марта) бележе нешто веће потрошње од оних у другом делу године. То се потврђује и месечним прегледом (Слика 4-17), одакле се види да се највеће потрошње остварују у новембру (односно марту и октобру) док најмање потрошње има у августу.

У погледу категорије и групе потрошача које доминирају (у складу са сликама: Слика 4-1 и Слика 4-4) занимљиво је показати и однос потрошњи ових категорија по годишњим добима и месецима (Слика 4-18 и Слика 4-19).



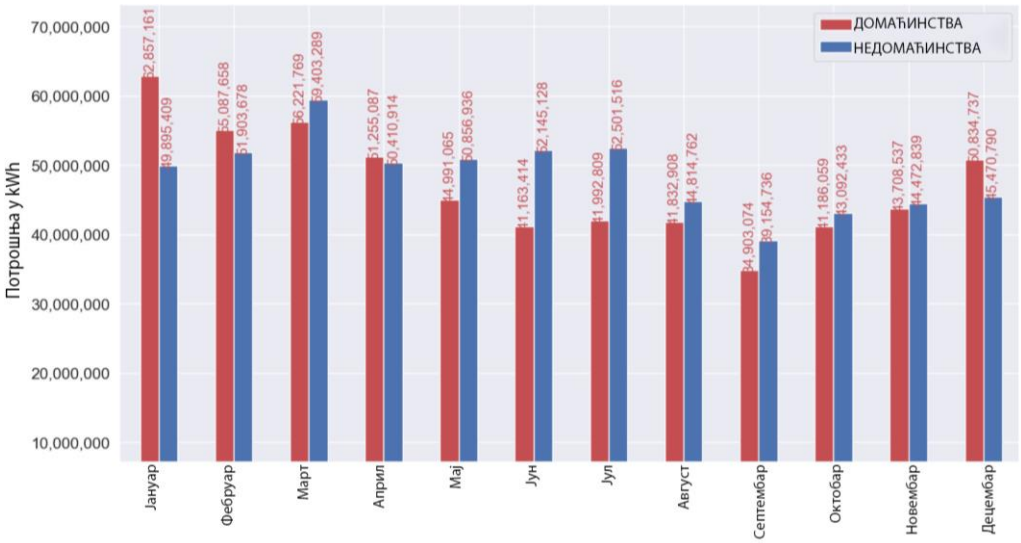
Слика 4-18: Потрошња према категорији потрошача (сумарно по годишњим добима)

Слика 4-19, показује јасну разлику у обиму потрошњи две раније поменуте категорије потрошача: домаћинства (плави стубићи) у односу на укупне потрошње потрошача ван домаћинстава (наранџасти стубићи). Тек свега 5% укупне остварене потрошње у финалном скупу припада потрошачима ван домаћинстава. Ово је потпуно природно обзиром на број активних бројила на нивоу сваког месеца (испрекидане линије и *y* оса са десне стране).

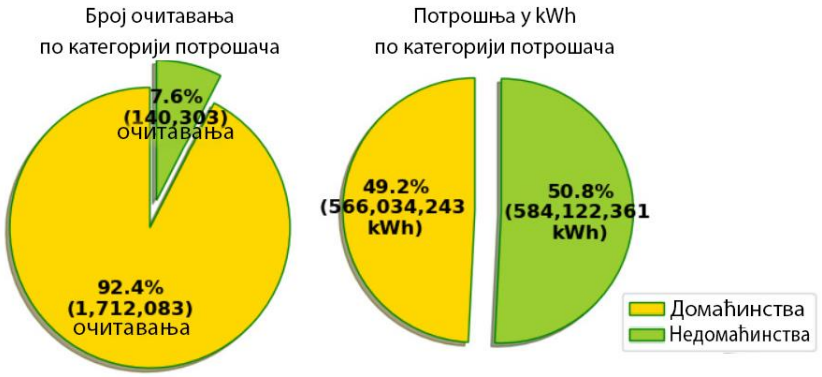
Слика 4-19: Потрошња према категорији потрошача (сумарно по месецима у години)

Насупрот томе, раније је у [143] приказана доста мања разлика између ове две категорије у погледу потрошње, чак и за поједине месеце приметан је већи износ потрошње за потрошаче ван домаћинстава у односу на оне у домаћинствима (Слика 4-20, слика (а)) док се у погледу броја очитавања (Слика 4-20, (b)) истовремено показује да нешто мање од 8% укупног броја очитаних бројила постиже више од 50% целокупног износа потрошњи.

а)



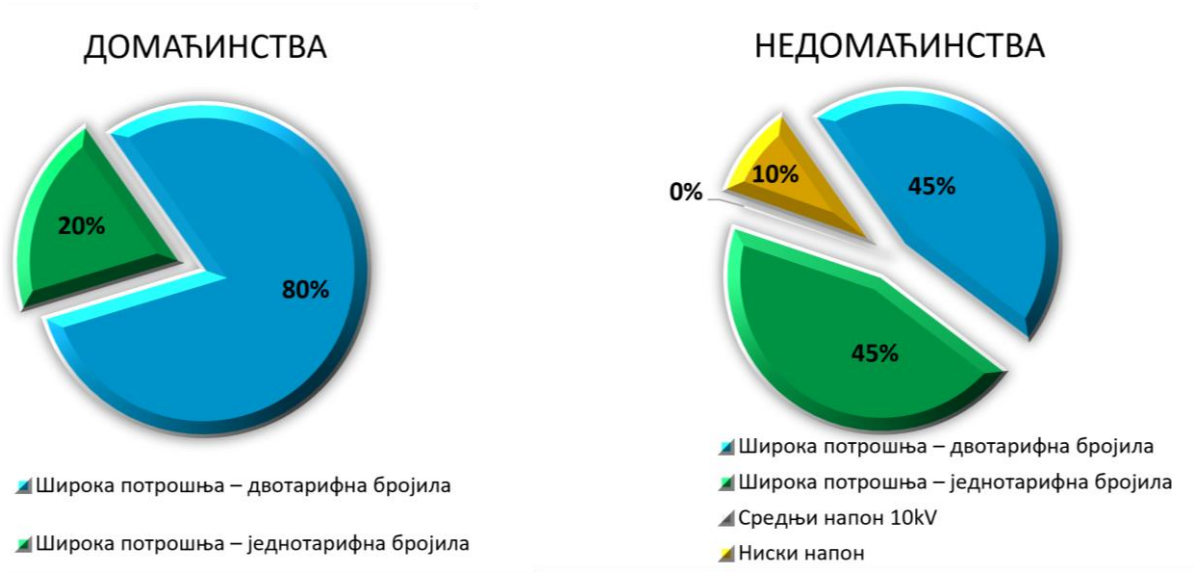
б)



Слика 4-20: a) Упоредни приказ остварене потрошње за сваки месец
b) Приказ броја очитавања бројила и остварене потрошње према категорији потрошача [143]

Разлог разлике приказане на сликама Слика 4-19 и Слика 4-20 долази из самог изворног скупа. У раду [143] су коришћени непречишћени подаци који обухватају све категорије и групе потрошача и мерних места укључујући и оне са екстремно великим потрошњама и мерна места јавне расвете која нису коришћена у даљем истраживању. Управо слике Слика 4-19 и Слика 4-20 оправдавају искључивање оваквих потрошача из даље обраде.

Из структуре потрошача у погледу категорија и група потрошача којој припадају може се закључити да су у категорији домаћинства присутне само две групе потрошача: *„Широка потрошња – једнотарифна бројила“* и *„Широка потрошња – двотарифна бројила“* док су у категорији недомаћинстава поред ове две, заступљене и све остале групе потрошача (Слика 4-21), што је и показано у [143].



Слика 4-21: Расподела група потрошача по категоријама

4.3. Предвиђање потрошње електричне енергије за све потрошаче: Приступ I

Након откривања основних структура у скупу података и упознавања карактеристика потрошача наступа фаза развоја модела машинског учења за предвиђање потрошњи електричне енергије.

На основу ранијег поређења резултата различитих техника машинског учења (Табела 1, потпоглавље 4.1.2) као модел са највише потенцијала истиче се вештачка неуронска мрежа. Из тог разлога ће најпре бити приказани резултати модела заснованог на неуронским мрежама. У поступку израде финалног ANN модела полази се од раније приказаног, тестног модела. Комплетан процес изводи се хеуристички, користећи сазнања из теоријских основа и знања стечених истраживањем података циљног скупа. Приликом формирања предложеног модела узете су у обзир разне технике, врсте слојева мреже, најразличитије комбинације хиперпараметара да би се дошло до финалног модела способног да предвиди месечне потрошње електричне енергије са минималном грешком. Рад са ANN након формирања модела, иако итеративан, може се посматрати кроз две основне фазе – фазу тренинга и фазу тестирања модела. Обе фазе су подједнако важне, модел треба да испољи добре особине и током тренинга модела али и да исте потврдити на тестном узорку. У том погледу су ANN посебно погодне, јер због своје адаптивности

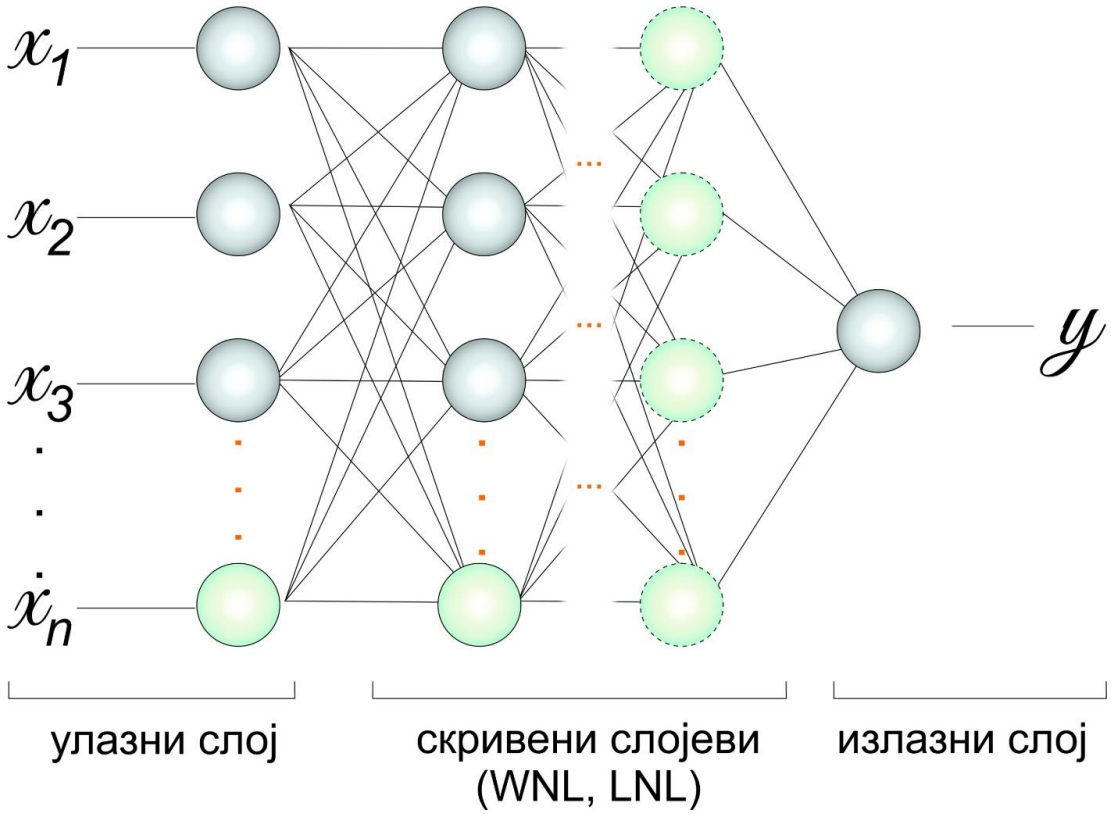
лако успевају да се прилагоде новим подацима. Током процеса обуке, ANN су у стању да истовремено узму у обзир велики број параметара који утичу на потрошњу електричне енергије и тиме препознају везе између наизглед неповезаних података. Са друге стране, истраживач може да прати вредности кључних параметара и врши евалуацију након сваког циклуса што му даје могућност да на исте утиче.

У наредним потпоглављима биће представљени појединачни хиперпараметри предложеног модела.

4.3.1. Архитектура предложеног модела ANN

Специфичност проблема, карактеристике потрошача и некохерентност података захтевају и специфично решење. Предвиђање потрошње електричне енергије није једноставно ни са теоријске нити практичне стране. Тако је и моделовање ANN модела прилично захтевно и подразумева доста експертског знања у области машинског учења, са једне стране, али и познавање социјалних аспеката потрошача и фактора који утичу на феномен потрошње електричне енергије.

За решавање проблема какав је предвиђање потрошње електричне енергије није довољна употреба потпуно повезаних густих слојева наслаганих између улазног и излазног слоја ANN модела. Раније, у поглављу 2.10.1 описана је природа WNL слоја који као такав не стоји сам већ се најчешће припаја уз конволуцијски слој, у предложеном моделу биће комбинован са густим слојем. Уз сваки WNL слој припојен је и LNL слој а груби приказ архитектуре предложеног модела дат је на следећој слици (Слика 4-22).



Слика 4-22: Груби приказ архитектуре предложеног модела

Активациона функција

Активациона функција коришћена за улазни и све скривене слојеве је *ReLU*, док се на излазном слоју примењује линеарна функција. Тестирањем је утврђено да се употребом *ReLU* активације неурона у слојевима постиже већа брзина у обрачуну него са, на пример, сигмоидном или тангенсном функцијом активације. Успешности *ReLU* функције доприноси њена једноставност што даље утиче на њену способност у избегавању проблема нестајућег градијента (ово је детаљније описано у поглављу 2.10.2).

Регуларизација

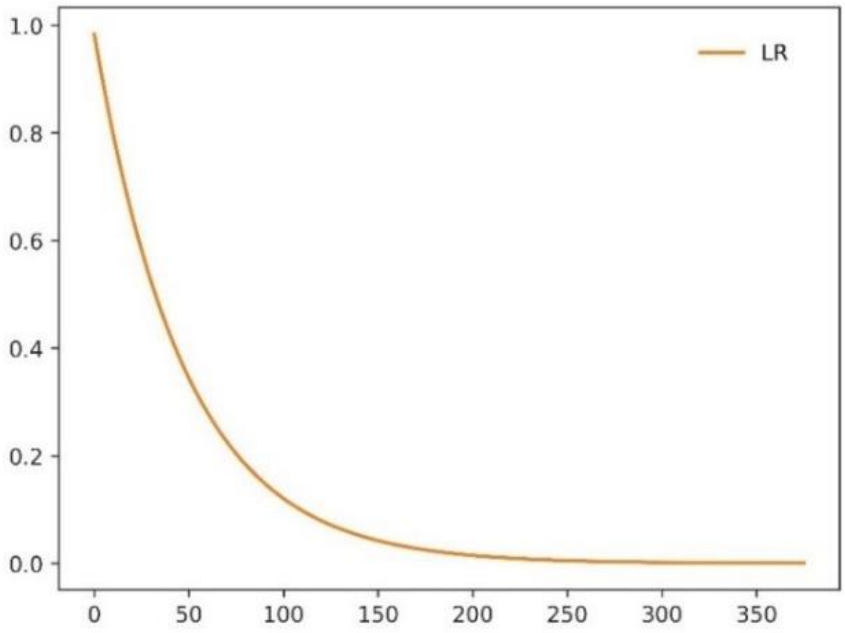
Архитектура предложеног модела ANN је трослојна (улазни, излазни слој, а унутар њих скривени слојеви). Како би предложени имао добре перформансе не само на тренинг скупу већ и за непознате, нове улазе коришћена је комбинација два типа регуларизације, познатија као *L12* регуларизација (описана раније у поглављу 2.10.6).

Оптимизациона функција и стопа учења

За оптимизациону функцију у предложеном моделу коришћен је оптимизатор *Adadelta* посебно због његове способности да ублажи агресивну, опадајућу стопу учења. Развијени модел представљен у овој дисертацији, полази од високих вредности стопе учења активационе функције, која је адаптивна током трајања обуке (Слика 4-23). Критеријум за адаптацију (умањење) стопе учења у предложеном моделу је постављен у односу на број епоха и може се представити изразом (35):

$$стопа\ учења = почетна\ стопа\ учења * e^{(-k*epoch)}, \tag{35}$$

где је: *k* задати коефицијент опадања стопе учења кроз епохе, са вредношћу од 0 до 1, а *epoch* број епохе, циклуса.



Слика 4-23: Промена стопе учења током епоха

Серија (мини-серија) као параметар модела

Величина мини-серије је један од важнијих хиперпараметара модела и његовим правилним постављањем процес обуке се може знатно убрзати али и нарушити уколико се задају вредности превише ниске или високе за дати обим података. За потребе ове дисертације имајући у виду велики тренинг узорак, коришћен је распон од 32 до 512 за величину серије. Експерименталном методом утврђено је да најбоље резултате (оптимално време уз минималну грешку) даје последња, највећа величина серије.

Како се у предложеном моделу стопа учења коригује током епоха раније поменутом методом, стога није потребно усклађивати величине стопе учења у односу на различите величине мини-серије, што важи и за остале хиперпараметре модела.

4.3.2. Утицај адаптације скупа и присуства слојева са нормализацијом тежина и слојева са нормализацијом активација слоја на прецизност предикције потрошње електричне енергије

Раније (у потпоглављу 4.1) је приказана структура обележја у иницијалном скупу података који, као такви, не обезбеђују довољно информација моделу за прецизно предвиђање потрошње електричне енергије. Адаптацијом скупа која је показана у истом поглављу добија се скуп обележја који садрже информације како о потрошачу тако и о амбијенту у коме се потрошач налази.

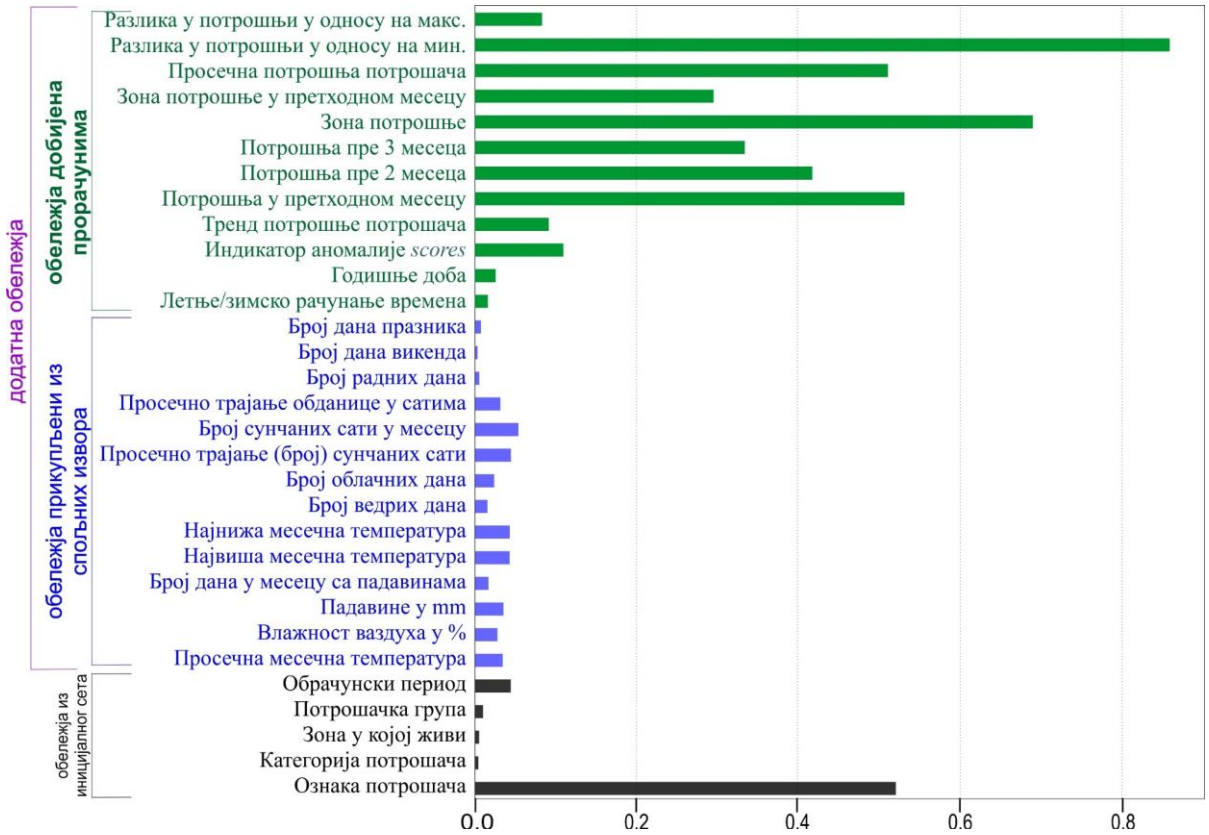
Приликом израде модела према архитектури предложеној у претходном потпоглављу (4.3.1) најпре је потребно испитати степен утицаја сваког појединачног обележја на потрошњу електричне енергије. Такође, треба испитати утицај параметара модела (попут броја неурона у скривеном слоју) на тачност предикције потрошње електричне енергије. У том смислу, значајно је поново одмерити важност сваког обележја у скупу података. Слика 4-24 показује скалу утицаја сваког појединачног обележја на таргет обележје. Према раније приказаном поступку адаптације скупа сва обележја се могу сврстати у две велике групе:

- обележја присутна у иницијалном скупу и
- додатна обележја.

Додатна обележја су она која су у препроцесинг фази укључена у скуп на један од два следећа начина:

- прикупљена из додатних, спољних извора и
- обележја добијена прорачунима на основу других обележја.

Јасно је да и у адаптираном скупу података, немају сва обележја једнак утицај на таргет обележје па то помаже у даљем току истраживања.

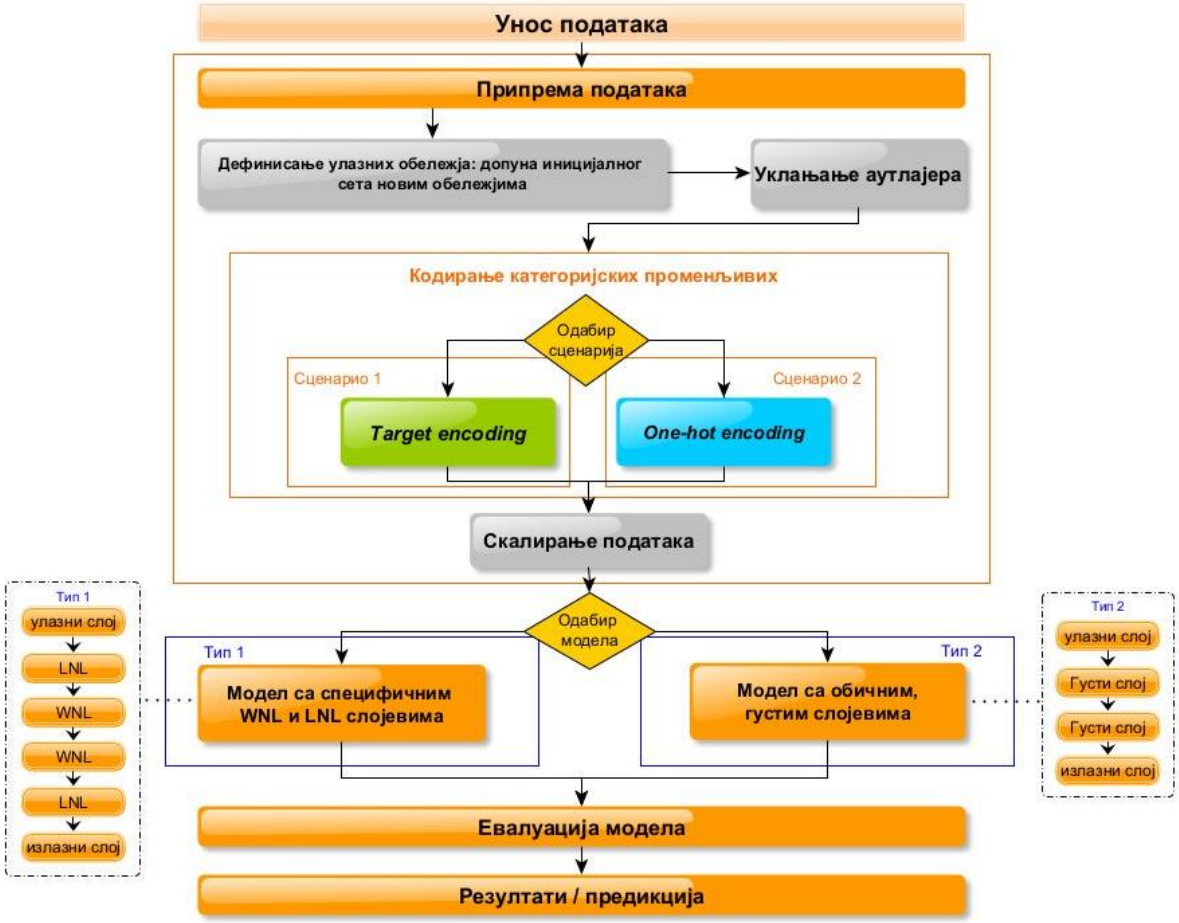


Слика 4-24: Важност обележја за предикцију потрошње електричне енергије

Слика 4-24 показује да обележја која потичу из иницијалног скупа (попут потрошачке зоне, групе и категорије потрошача) имају најмање утицаја на потрошњу. Значајан али не највећи утицај имају обележја прикупљена из спољних извора (углавном обележја која описују метеоролошке услове и временске прилике), док група обележја добијена прорачунима има највећи утицај на висину потрошње електричне енергије.

Како би се утврдио утицај скривених слојева са нормализацијом тежина у наставку ће бити поређени резултати добијени помоћу два модела: први је заснован на слојевима са нормализацијом тежина у густим слојевима и слојевима са нормализацијом активације (WNL и LNL слојеви описани у поглављу 2.10.1), а други садржи искључиво обичне густе слојеве. Такође, биће испитан утицај присуства ознаке потрошача као једног од обележја - улазних параметара, јер су већ први резултати испитивања важности обележја у иницијалном скупу (Слика 4-5), а потом и након адаптације (Слика 4-24) указали на њен велики значај.

Поступак испитивања утицаја слојева би се могао представити дијаграмом датим на следећој слици (Слика 4-25).



Слика 4-25: Дијаграм тока предложене методологије испитивања утицаја WNL скривених слојева [144]

Приметно је да се целокупан процес одвија фазно док се кроз неколико условних грана бирају одговарајући поступци [144].

1. Најпре се у препроцесинг фази дефинишу улазне варијабле и уклањају аутлајери, а затим се приступа кодирању података по једном од два предвиђена сценарија[8]:
   - сценарио 1: за кодирање свих категоријских обележја се користи кодирање по принципу средњих вредности таргет варијабле;
   - сценарио 2: за сва категоријска обележја се користи бинарно кодирање (*One-Hot Encoding)*, изузев за обележје које представља ознаку потрошача. Због превеликог броја различитих категорија за обележје ознака потрошача се користи кодирање из првог сценарија.
2. Након кодирања следи фаза скалирања података, а потом се подаци прослеђују у један од два (типа) модела:
   - тип 1 је сачињен из два, у основи густа скривена слоја са нормализацијом тежина (WNL слоја) којима следи и претходи по један слој који врши нормализацију активација претходног слоја (LNL слој);

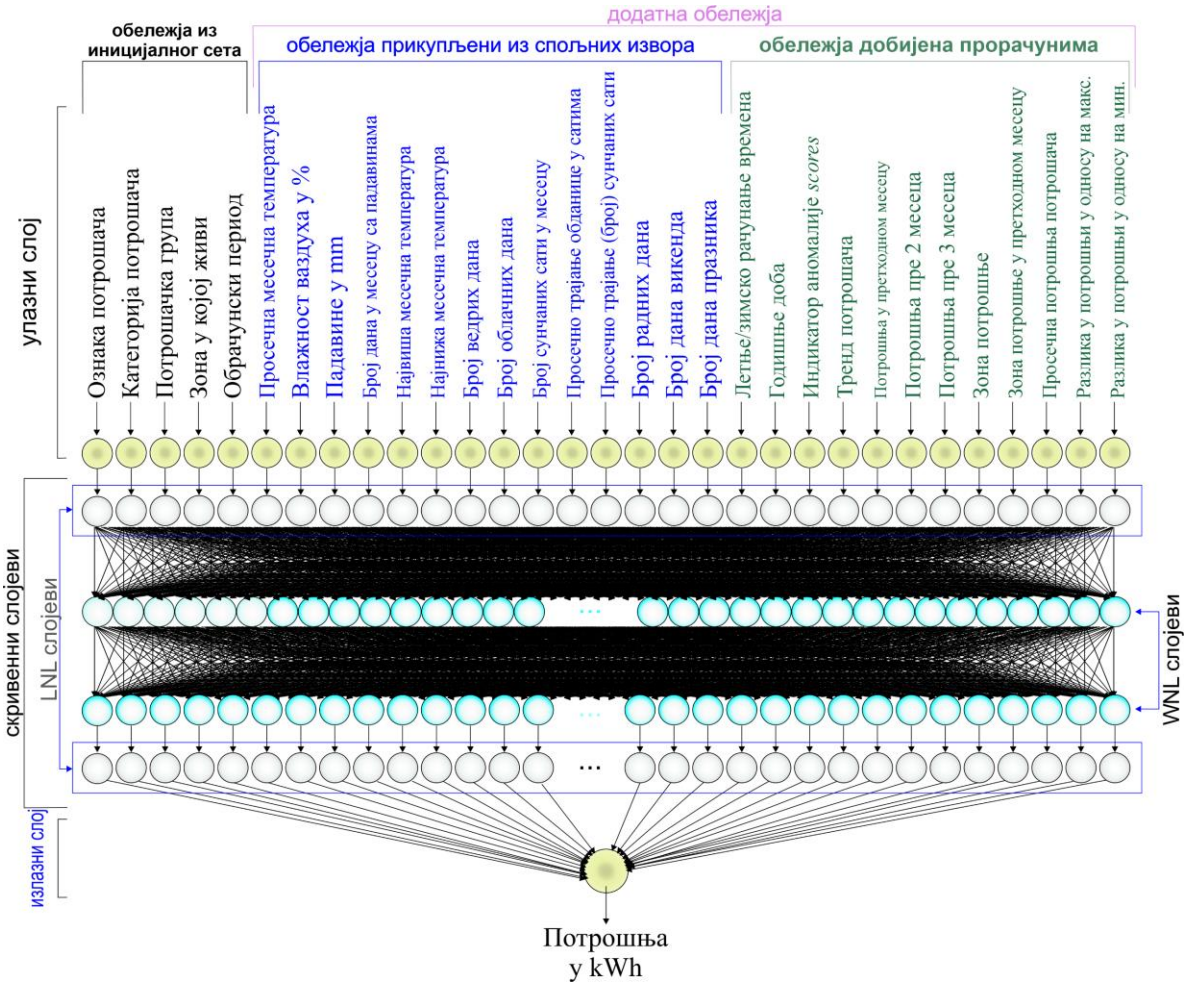
[8] Поступак је детаљније објашњен у поглављу 4.1

- тип 2 уместо два WNL садржи два обична густа слоја.

Сваки од типова модела је тестиран на мрежама са различитим бројем неурона у скривеним слојевима у распону од 15 до 85.

3. Евалуација модела, која следи након фазе тренинга је јединствена у оба случаја, и врши се кроз исте метрике: RMSE, RMSE (%), MAPE и R^2. У овој фази се испитује утицај присуства слојева са нормализацијом тежина и слојева са нормализацијом активација слоја на прецизност предикције потрошње електричне енергије.
4. Резултати евалуације су дати у Табели 2.

Архитектура модела типа 1 дата је на следећој слици (Слика 4-26).



Слика 4-26: Архитектура предложеног модела (тип 1) [144]

Табела 2: Резултати добијени различитим сценаријима и типовима ANN

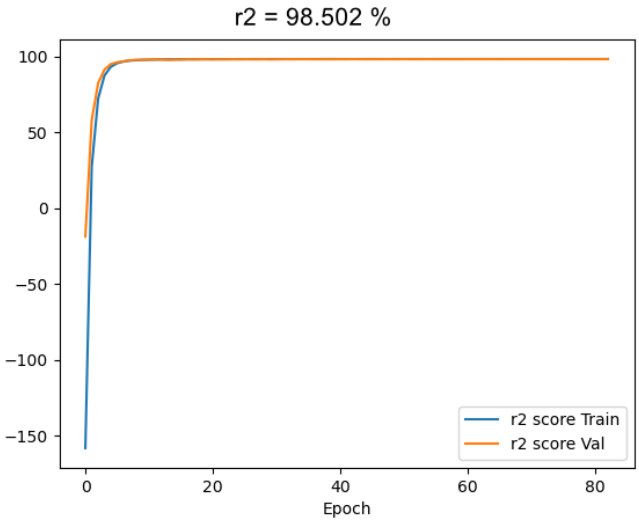
ANN модел тип:		сценарио 1 + тип 1		сценарио 1 + тип 2		**сценарио 2 + тип 1**		сценарио 2 + тип 2	
Ознака потрошача као улазни параметар: →		+	-	+	-	+	-	+	-
Модел са 85 неурона у сваком скривеном слоју	RMSE (kWh):	29,35	30,26	30,97	32,64	**25,1**	30,008	26,08	30,93
	RMSE (%):	7,02	7,24	7,4	7,81	**6,00**	7,18	6,19	7,40
	MAPE (%):	5,15	5,34	5,40	5,59	**4,11**	5,28	4,19	5,35
	R^2:	97,9%	97,8%	97,6%	97,4%	**98,5%**	97,9%	98,3%	97,7%
Модел са 65 неурона у сваком скривеном слоју	RMSE (kWh):	31,22	30,39	30,99	33,18	25,14	30,06	26,4	30,73
	RMSE (%):	7,47	7,27	7,41	7,94	6,01	7,19	6,31	7,35
	MAPE (%):	5,49	5,38	5,47	5,95	4,12	5,32	4,38	5,48
	R^2:	97,6%	97,7%	97,7%	97,3%	98,5%	97,8%	98,3%	97,7%
Модел са 35 неурона у сваком скривеном слоју	RMSE (kWh):	30,76	31,21	31,47	32,2	25,78	30,13	26,775	31,44
	RMSE (%):	7,36	7,46	7,53	7,70	6,17	7,21	6,40	7,52
	MAPE (%):	5,38	5,54	5,54	5,74	4,25	5,34	4,42	5,67
	R^2:	97,7%	97,6%	97,6%	97,5%	98,4%	97,8%	98,3%	97,6%
Модел са 15 неурона у сваком скривеном слоју	RMSE (kWh):	32,67	33,79	34,19	34,89	27,55	31,68	27,86	34,14
	RMSE (%):	7,81	8,08	8,18	8,34	6,66	7,58	6,66	8,26
	MAPE (%):	5,7	6,02	6,16	6,31	4,588	5,7	4,62	6,17
	R^2:	97,4%	98,2%	97,1%	97,0%	98,1%	97,5%	98,1%	97,2%

Из резултата приказаних у претходној табели јасно је да најбоље резултате даје предложени модел са WNL и LNL слојевима и то када се као један од улазних параметара користи ознака потрошача кодирана помоћу средњих вредности, док су остале категоријске променљиве кодиране помоћу *One-Hot Encoding* (сценарио 2 + тип 1, са ознаком потрошача, осенчена колона у Табели 2).

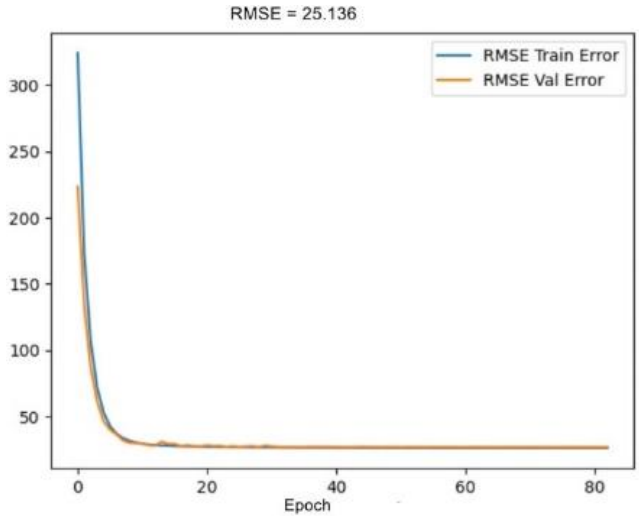
Поредећи резултате дате у претходној табели са почетним резултатима које дају други модели машинског учења (Табела 1), јасно је да адаптација скупа додатним обележјима пружа довољно информација ANN моделу који је у стању да предвиди месечне потрошње електричне енергије. Ово уједно потврђује трећу и четврту посебну хипотезу дисертације по којима се може формирати одговарајући скуп параметара ANN модела, са једне стране и одговарајући скуп обележја, са друге стране, који ће омогућити предвиђање месечних потрошњи електричне енергије на једном ширем подручју са потрошачима разуђених карактеристика.

У прилог томе, говоре и мере евалуације дате на следећој слици (Слика 4-27). Квалитет модела и његова успешност у току тренинг процеса дате су графицима корена средње квадратне грешке и средње просечне апсолутне грешке за најуспешнији случај (са 85 неурона). Као једна од најзначајнијих мера за утврђивање степена квалитета тренутног скупа обележја и њихове способности за правилно описивање зависне варијабле (успешност предвиђања потрошње) дат је дијаграм коефицијента детерминације чија вредност од 98,5% говори о великој прецизности.

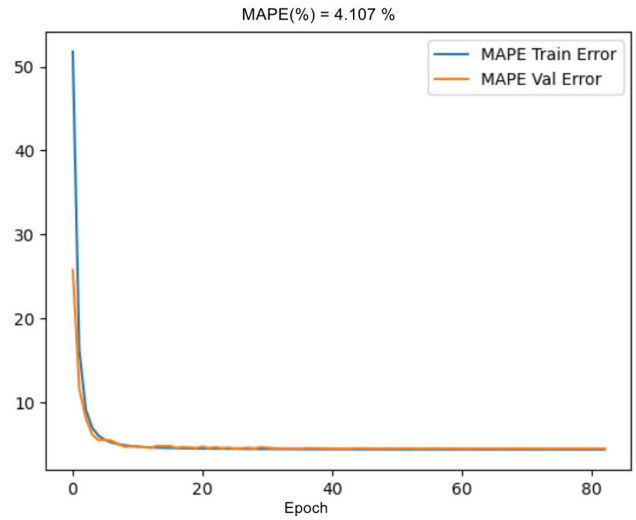
(а)



(б)



(в)



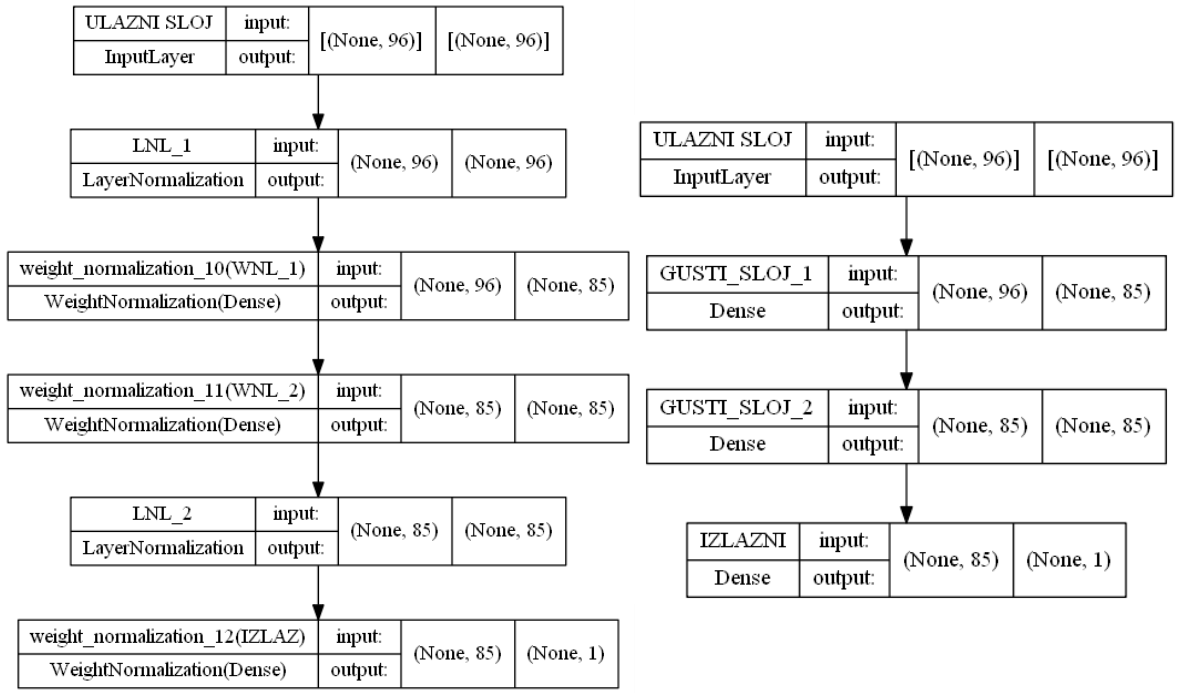
Слика 4-27: Коефицијент детерминације (а), корен средње квадратне грешке (б) и средња просечна апсолутна грешка (в) за сценарио 2 + модел типа 1 (модел са 85 неурона)

Претходна слика недвосмислено показује конзистентност модела како на тренинг (плава линија) тако и на тест скупу података (наранџаста линија). Приметна је јасна тежња

модела у приближавању *x* оси на сваком дијаграму показујући да се таргет варијабла може у готово 99% случајева описати помоћу датог скупа обележја, при чему је просечна апсолутна грешка мања од 5%, односно око 25 kWh, што у скупу са разноврсним потрошачима мора оценити као ниска грешка.

Како на први поглед (Слика 4-25) мрежа типа 1 (са WNL и LNL сојевима) има више слојева од мреже типа 2, то наводи на закључак да је и број неурона у мрежи неупоредив, а самим тим и резултати ове две мреже. Иако би се могло наслутити да дупло више скривених слојева у мрежи типа 1 значи и дупло више параметара који се тренирају али и доста боље шансе за повољан исход, због природе скривених слојева, то у овом случају није тако.

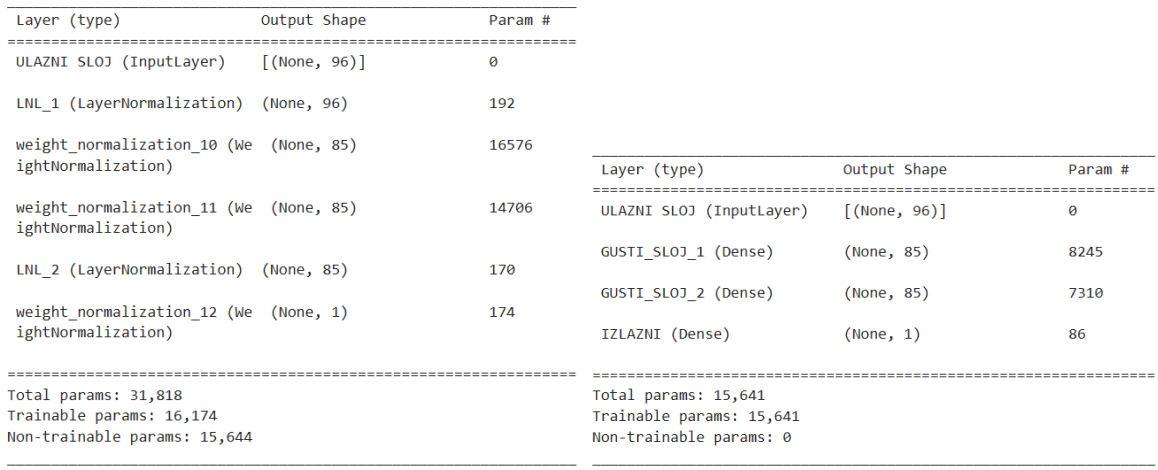
Слика 4-28 илуструјући архитектуру ANN оба типа показује и број неурона на улазу и излазу из сваког слоја. Приметно је да је код LNL скривених слојева број неурона на улазу и излазу исти. Разлог за то је што овај слој заправо не садржи неуроне који ће бити тренирани већ користи неуроне из претходног слоја како би их нормализовао и проследио наредном слоју.



а) б)

Слика 4-28: Програмска штампа (шема) архитектуре ANN типа 1 (а) и типа 2 (б) - (случај са 85 неурона у скривеним слојевима)

За увид у број параметара у случају обе мреже, дата је програмска штампа броја параметара за сваки од слојева у мрежама (Слика 4-29).



Layer (type)	Output Shape	Param #
ULAZNI SLOJ (InputLayer)	[(None, 96)]	0
LNL_1 (LayerNormalization)	(None, 96)	192
weight_normalization_10 (WeightNormalization)	(None, 85)	16576
weight_normalization_11 (WeightNormalization)	(None, 85)	14706
LNL_2 (LayerNormalization)	(None, 85)	170
weight_normalization_12 (WeightNormalization)	(None, 1)	174

Total params: 31,818
Trainable params: 16,174
Non-trainable params: 15,644

а)

Layer (type)	Output Shape	Param #
ULAZNI SLOJ (InputLayer)	[(None, 96)]	0
GUSTI_SLOJ_1 (Dense)	(None, 85)	8245
GUSTI_SLOJ_2 (Dense)	(None, 85)	7310
IZLAZNI (Dense)	(None, 1)	86

Total params: 15,641
Trainable params: 15,641
Non-trainable params: 0

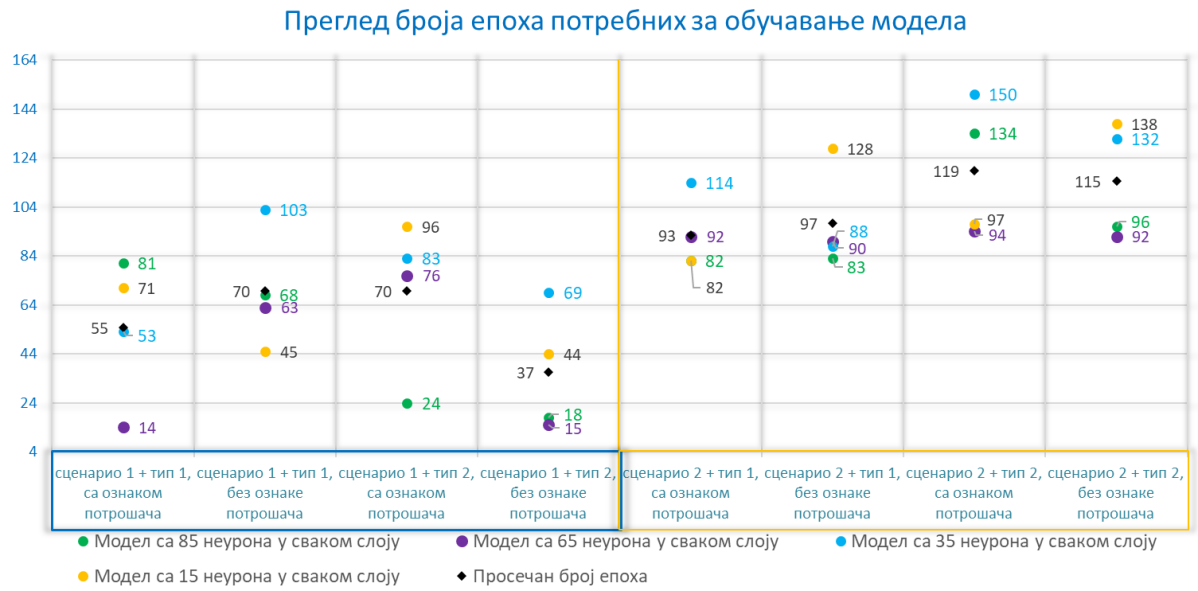
б)

Слика 4-29: Програмска штампа параметара ANN модела типа 1 (а) и модела типа 2 (б) - (случај са 85 неурона у скривеним слојевима)

Примећује се да број *"trainable"* параметара (параметара који се тренира) у мрежи са WNL и LNL слојевима (Слика 4-29 - а) није очекивано већи од броја параметара у оној са обичним густим слојевима (Слика 4-29 - б). Иако програмска штампа параметара модела показује и број параметара који се тренирају и број оних који не подлежу тренирању, сумарни приказ мреже са WNL и LNL слојевима не даје у потпуности тачан број неурона. Прозвана процедура за штампу параметара рачуна и параметре из LNL слоја као *"trainable"* иако теоријски овај слој нема параметре за обуку [57]. Како приказани број неурона у LNL слојевима представља неуроне пренесене из претходног слоја, исте не би требало рачунати у неуроне који се тренирају.

Поред броја неурона и начина кодирања, занимљиво је посматрати број циклуса (епоха) потребних за тренирање мрежа које су дале резултате приказане у Табели 2. Слика 4-30 показује да број епоха потребан за обучавање модела варира од 14 до 150. Занимљиво је да модел брже учи када је примењено кодирање категоријских обележја преко средњих вредности у односу на *One-Hot* кодирање (Слика 4-30, лева половина слике). Поредећи са резултатима метрика евалуације модела (из Табеле 2), сценарио 2 (Слика 4-30, десна половина слике), иако спорији, даје прецизнија предвиђања.

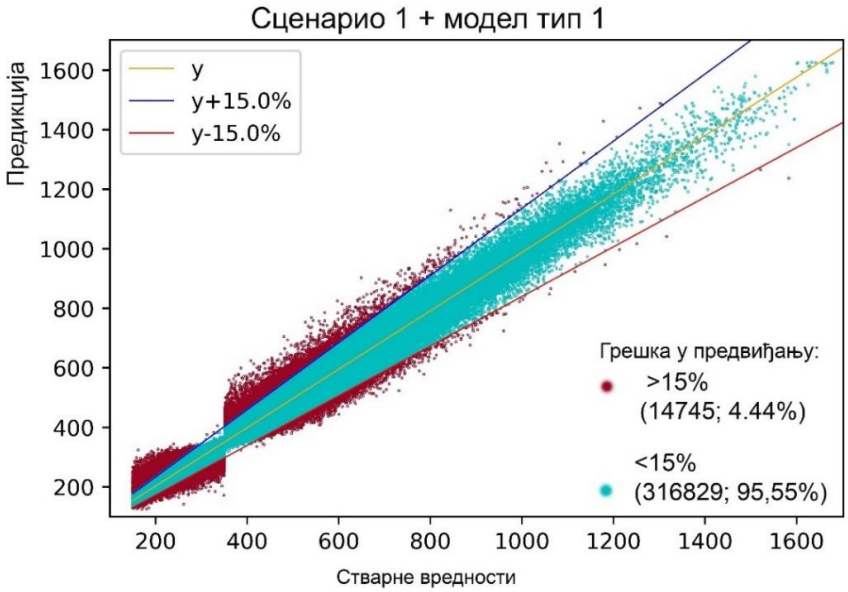
Посматрајући модел са највећим бројем неурона у сваком скривеном слоју (модел са 85 неурона, зелена ознака на претходној слици), приметно је да не постоји законитост која се може прозрети на први поглед. Оно што се може закључити јесте да је у случају сценарија 2, тј. у случају када се категоријске променљиве кодирају помоћу бинарних вредности, уз присутно обележје потрошача, моделу *типа 2* потребно знатно више циклуса (епоха) до постизања минимума грешке него моделу типа 1 уз исте услове (исти поступак кодирања података, исти број неурона у скривеном слоју и исте вредности хиперпараметара...).



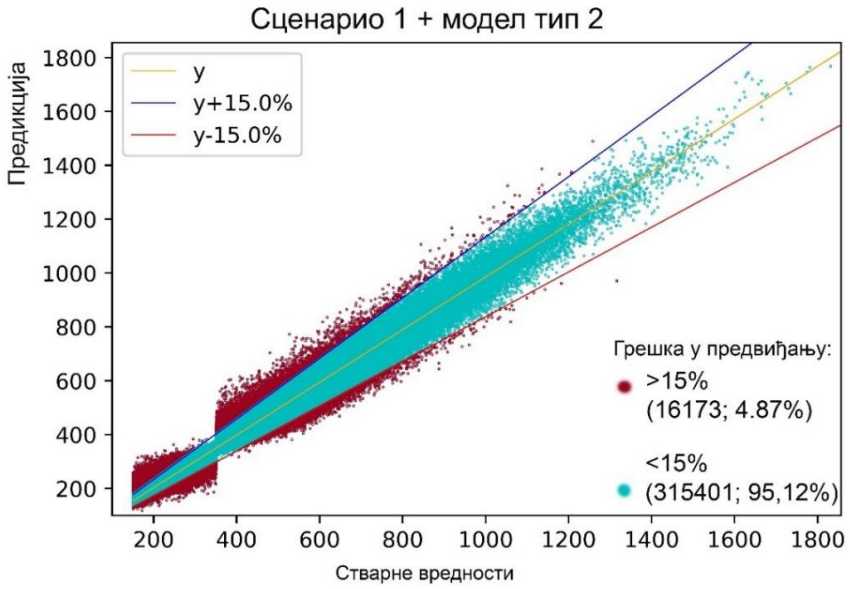
Слика 4-30: Број епоха за сваки од сценарија/модела/типова кодирања и броја неурона у мрежи [144]

Како би резултати били јаснији, на следећим сликама (Слика 4-31 до Слика 4-34, [144]) ће бити приказана дистрибуција вредности за средњу апсолутну грешку (MAPE(%)), за сваки од 4 раније поменута случаја. Како модел са 85 неурона у сваком скривеном слоју у комбинацији са ознаком потрошача као улазним параметром пружа највећу прецизност, приказане расподеле се односе управо на овај модел.

Следеће две слике (Слика 4-31 и Слика 4-32) се односе на случај када су категоријска обележја кодирана помоћу средње вредности таргет варијабле (сценарио 1).



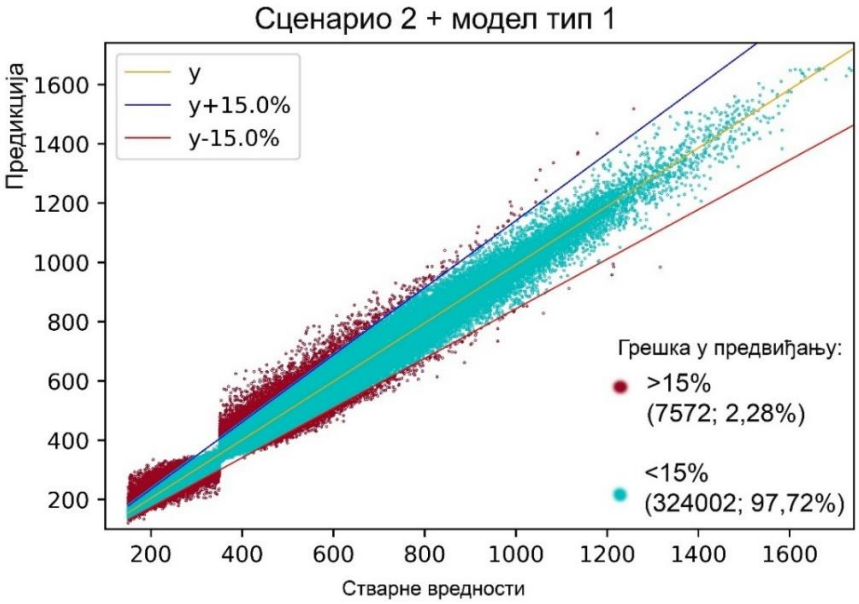
Слика 4-31: Расподела предикција добијена комбинацијом првог сценарија и модела типа 1



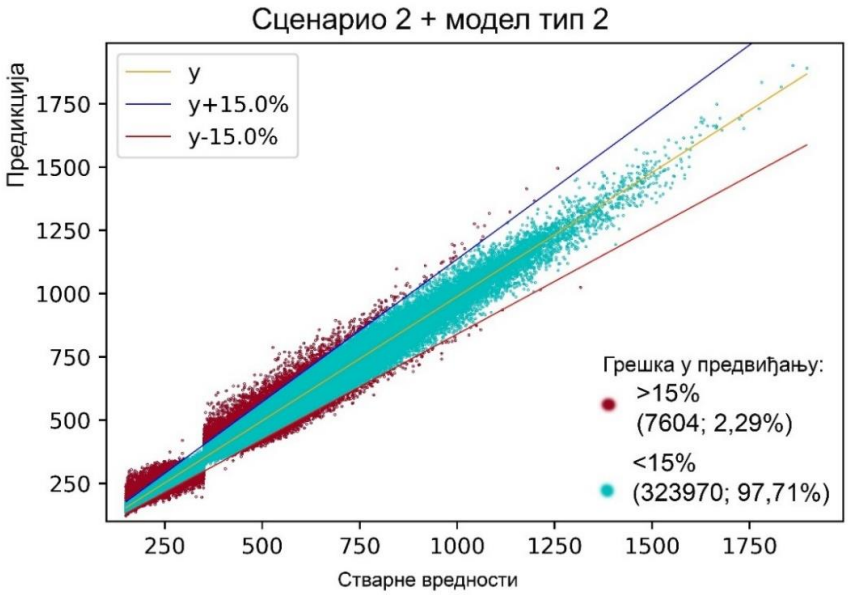
Слика 4-32: Расподела предикција добијена комбинацијом првог сценарија и модела типа 2

Слика 4-31 показује расподелу предикованих вредности добијену моделом са два обична, потпуно повезана, густа слоја док се Слика 4-32 односи на ANN модел са WNL и LN скривеним слојевима. Тачке (предикције) које се налазе између плаве и црвене линије представљају оне чија је грешка у распну од ±15% (MAPE). Тачке које се налазе ван ових линија (црвено обојене) представљају потрошаче за које је модел погрешио више од ±15% што за оба модела (модел тип 1 и модел тип 2) значи да преко 95% предикованих вредности има малу грешку, мању од 15%.

Наредне две слике (Слика 4-33 и Слика 4-34) се односе на сценарио 2, односно случајеве када су сва категоријска обележја кодирана помоћу *One-hot* енкодера (осим ознаке потрошача која је кодирана преко средњих вредности).



Слика 4-33: Расподела предикција добијена комбинацијом другог сценарија и модела типа 1



Слика 4-34: Расподела предикција добијена комбинацијом другог сценарија и модела типа 2

Слика 4-33 показује још већу прецизност модела 1 него уз примену сценарија 1 дајући поново благу предност моделу са слојевима са нормализацијом тежина (модел типа 1), који за свега 2,28% потрошача греши ван оквира од ±15%. Преведено у kWh, грешка коју прави модел 1 износи ≈25 kWh, а средња апсолутна процентуална грешка за све потрошаче износи ≈4,1%.

Обзиром да на посматраном подручју постоји изузетно хетерогена структура потрошача, а међу њима и оне са потрошњама од чак неколико хиљада kWh на месечном нивоу, грешка мања од 5% се може сматрати за јако малу грешку и потврђује успешност предложеног модела (модела типа 1).

Кôд развоја модела типа 1 и модела типа 2 дати су у виду прилога (Прилог 6 и Прилог 7).

4.3.3. Поређење резултата ANN модела са другим ML моделима

Достигнута прецизност предложеног ANN модела, приказана је у претходном поглављу. Ради указивања на тачност и поузданост предложеног модела, биће дата упоредна анализа вредности метрика евалуације које он омогућује наспрам вредности истих метрика добијених са неколико других алгоритама ML.

У том смислу, у наставку ће, узевши у обзир природу проблема и обим података, бити поређени резултати неколико раније описаних алгоритама:

1. линеарна (мултиваријантна) регресија,
2. градијент буст – *XGBoost*,
3. линеарни *SVR*,
4. Хубер регресиони модел и
5. стабла одлучивања.

При тестирању сваког од пет побројаних модела узета је иста подела скупа као код предложеног модела заснованог на ANN узимајући исти скуп обележја. У Табели 3,

приказани су резултати метрика евалуације добијени помоћу неколико модела машинског учења.

Табела 3: Поређење резултата предложеног модела са резултатима других техника ML

Модел	**RMSE (%)**	**MAE (%)**	**MAPE (%)**	**AR^2 (%)**
Линеарна регресија	10,82	7,92	8,61	95,26
Градијент буст (*XGBoost)*	7,93	5,64	5,94	97,45
Линеарни *SVR*	10,88	7,85	8,52	95,21
Хубер регресија	10,89	7,87	8,54	97,45
Стабла одлуке	10,61	6,69	6,94	95,44
Предложени модел ANN	7,9	5,68	5,46	96,25

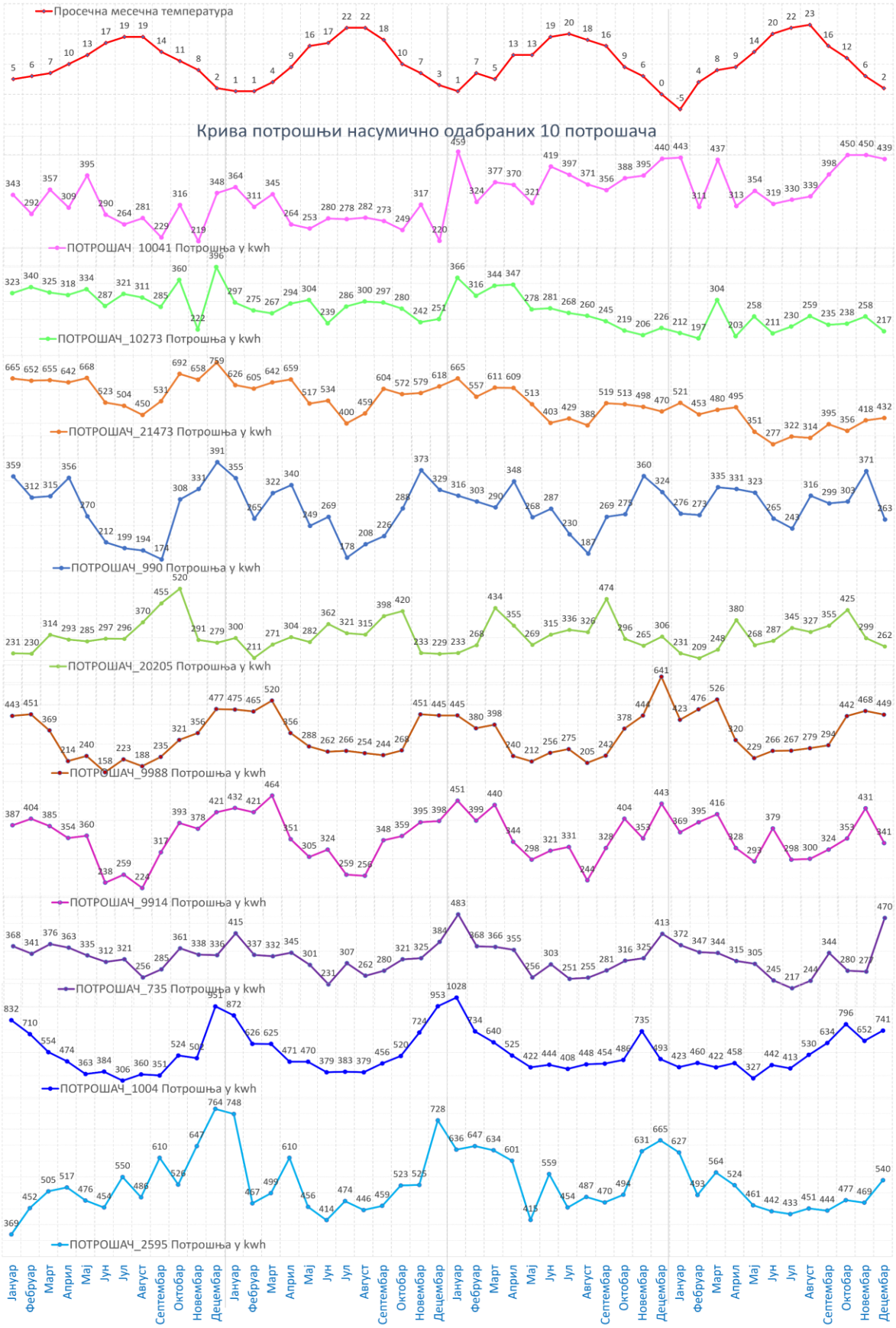
Разматрајући резултате евидентна је доминација предложеног модела у том контексту. Средња процентуална грешка (MAPE) са вредношћу 5,46% представља значајну предност у односу на друге тестиране моделе.

Најбоље се показао алгоритам XGBoost са вредношћу апсолутне просечне процентуалне грешке најближе резултату предложеног модела. Ипак овај модел захтева знатно више рачунарских ресурса, посебно у виду процесорског времена које може бити увећано и по неколико пута а највише зависи од обима података. Такође, не даје увид у поједине параметре током обрачунавања и припреме модела за тестирање па се самим тим не може утврдити његова директна успешност као што је то могуће за предложени модел.

Кôд развоја модела других техника ML чији су резултати приказани у Табели 3 дат је у виду прилога (Прилог 8).

4.4. Предвиђање потрошње електричне енергије за одређену категорију потрошача (кластер): Приступ II

Узимајући у обзир суме потрошњи електричне енергије за све потрошаче на циљном подручју у посматраном периоду (као што је приказано у поглављу 4.2 - Слика 4-17, Слика 4-18 и Слика 4-19) приметна су одређена осциловања на месечном нивоу. Иако на први поглед ове разлике не дају довољно материјала за било какав закључак, ANN је у стању да препозна и најситније законитости и искористи их за прецизније предвиђање. Томе у прилог говори и дијаграм (Слика 4-35) који илуструје потрошње 10 насумично одабраних потрошача током посматраног периода. Црвена линија на врху илуструје промене просечних температура за сваки посматрани месец, респективно.



Слика 4-35: Илустрација месечне потрошње током четири посматране године за 10 насумично одабраних потрошача у односу на просечне месечне температуре у истом периоду

Сви узорковани потрошачи су домаћинства и припадају широкој потрошњи. Половина домаћинства је смештена у граду а друга половина у приградској средини. Сви потрошачи осим потрошача: *ПОТРОШАЧ_2595* и *ПОТРОШАЧ_10273* поседују двотарифно бројило (Табела 4).

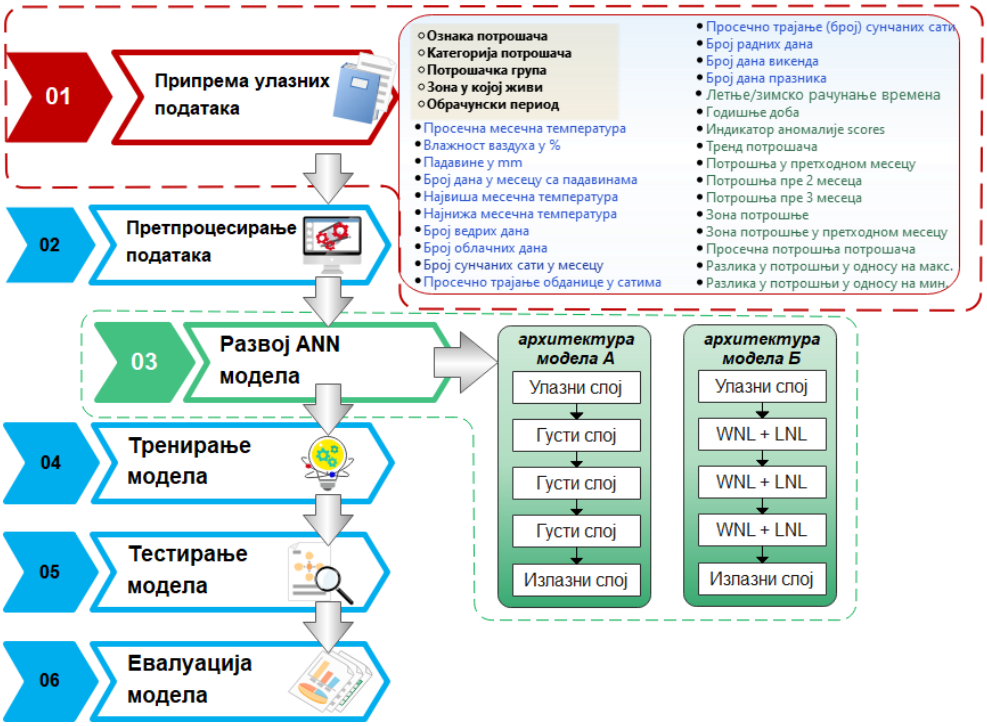
Табела 4: Основне карактеристике потрошача чије су потрошње илустроване на следећој слици (Слика 4-35)

ОЗНАКА ПОТРОШАЧА	КАТЕГОРИЈА ПОТРОШАЧА	ЗОНА (у којој се потрошач налази)	ВРСТА БРОЈИЛА
ПОТРОШАЧ_735	Домаћинство	Приградска зона	Двотарифно бројило
ПОТРОШАЧ_990	Домаћинство	Градска зона	Двотарифно бројило
ПОТРОШАЧ_2595	Домаћинство	Приградска зона	Једнотарифно бројило
ПОТРОШАЧ_9914	Домаћинство	Приградска зона	Двотарифно бројило
ПОТРОШАЧ_9988	Домаћинство	Градска зона	Двотарифно бројило
ПОТРОШАЧ_20205	Домаћинство	Градска зона	Двотарифно бројило
ПОТРОШАЧ_21473	Домаћинство	Градска зона	Двотарифно бројило
ПОТРОШАЧ_1004	Домаћинство	Градска зона	Двотарифно бројило
ПОТРОШАЧ_10041	Домаћинство	Приградска зона	Двотарифно бројило
ПОТРОШАЧ_10273	Домаћинство	Приградска зона	Једнотарифно бројило

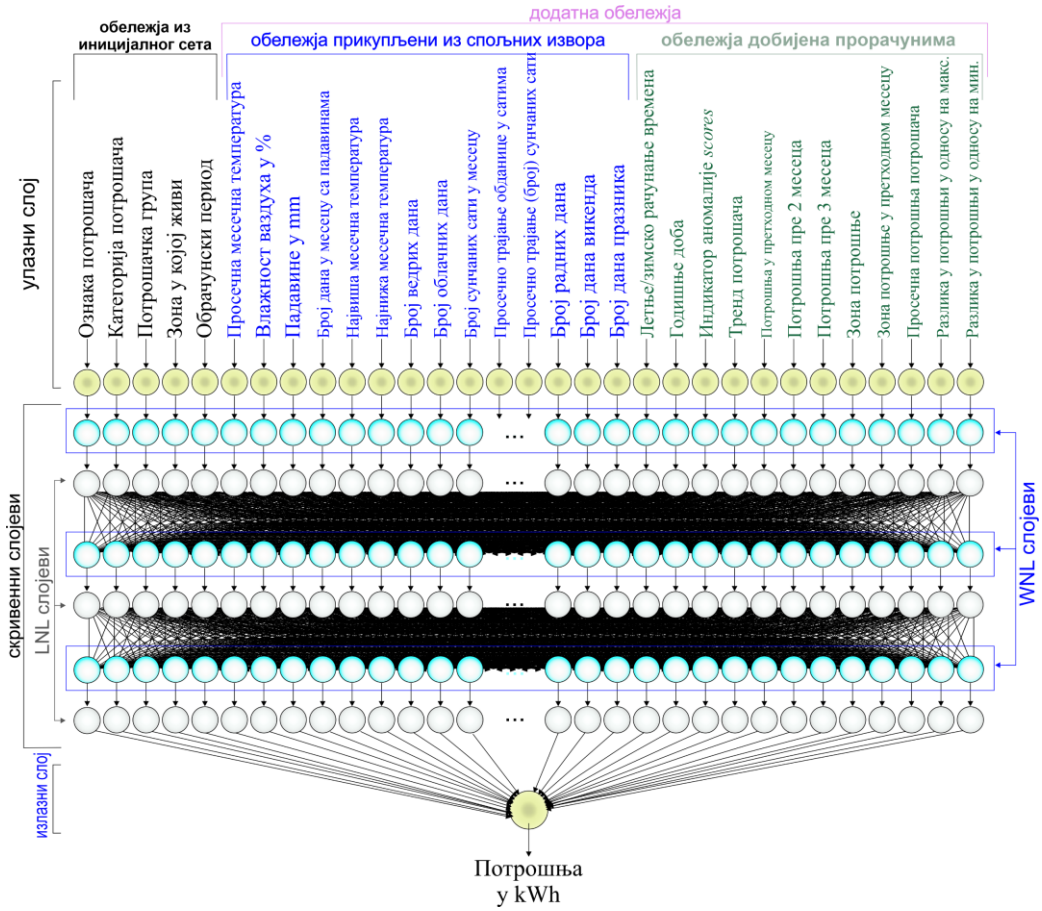
Појачане вертикалне линије на претходном дијаграму (Слика 4-35) представљају крај једне односно почетак наредне календарске године. Приметно је да у зимским месецима код свих потрошача долази до пораста потрошњи. Линија температуре (црвена линија) осетно прати линије потрошњи за већину посматраних потрошача, али обрнуто пропорционално: на местима где се потрошња повећава, температуре се смањују.

Из претходно наведеног би се могло закључити да временске прилике знатно утичу на месечне потрошње електричне енергије, посебно оних у домаћинствима. Такође, може се претпоставити да потрошачи на територији града Ужица у већој мери електричну енергију користе за грејање него за хлађење свог животног простора, обзиром да потрошње расту у зимским месецима. Стога, има смисла вршити предикцију потрошње у односу на различите временске периоде (период од месец дана, једног годишњег доба, календарске године, итд).

У наставку ће бити приказана архитектура формираног, универзалног модела, (представљеног у [54]). Модел је формиран са циљем предвиђања потрошњи електричне енергије за разноврсне потрошаче, за било који циљани месец у календарској години. И овај модел је заснован на нормализацији тежина улазних параметара у сваки густи слој. На следеће две слике дата је груба структура тока активности целокупног процеса представљеног у овом поглављу (Слика 4-36) и архитектура предложеног модела (Слика 4-37) а према методологији представљеној у 3.3.1, Слика 3-3.



Слика 4-36: Дијаграм тока активности у изради и оцени модела по фазама (према методологији представљеној у 3.3.1, Слика 3-3) [54]



Слика 4-37: Илустрација архитектуре предложеног модела *Б* (фаза 03, Слика 4-36)

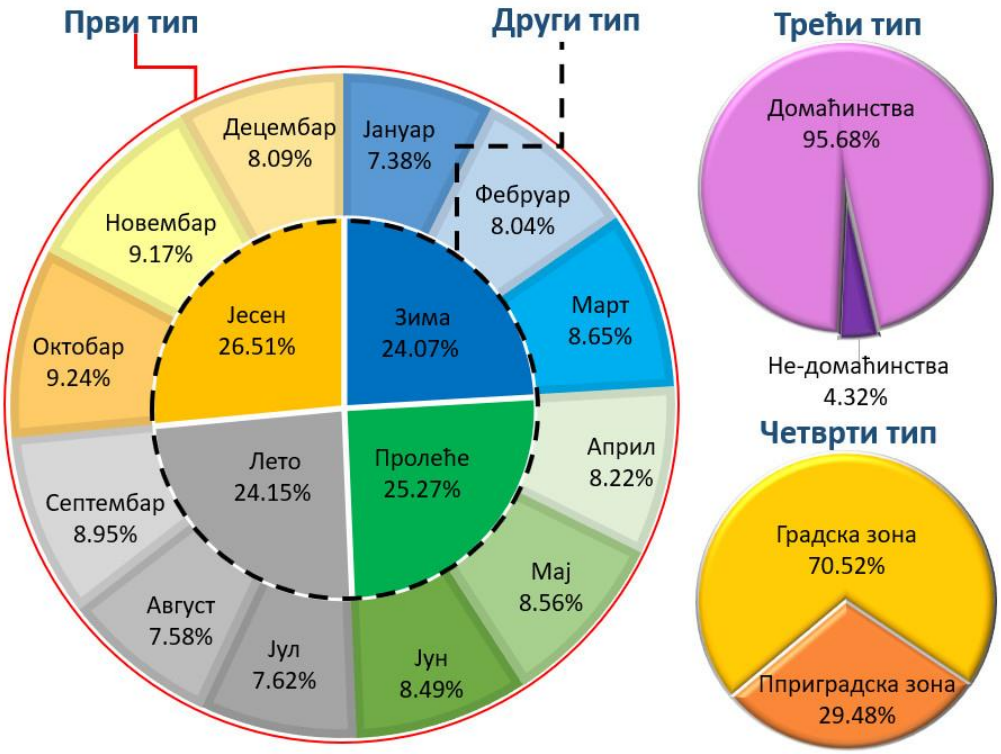
У првом кораку се врши селекција улазних података (фаза 01, Слика 4-36). Раније (кроз поглавље 4.2) су приказане особине и структура потрошача на простору града Ужица садржаних у сету података. По том основу, има смисла формирати више кластера групишући потрошаче по неколико критеријума. У том смислу издвајају се четири основа за кластеровање, илустрована на следећој слици (Слика 4-38).

Први тип (тип I) – кластеровање према месецу у години (дванаест кластера). Сваки кластер садржи податке о потрошњама из само једног месеца (за све постојеће потрошаче, за четири посматране године).

Други тип (тип II) – кластеровање према годишњем добу (четири кластера). Сваки од четири кластера садржи податке из три месеца која припадају одговарајућем годишњем добу (као што је описано раније у поглављу 4.1).

Трећи тип (тип III) – кластеровање према категорији потрошача. Према овом кластеровању постоје два кластера која праве разлику између потрошача у домаћинствима и оним ван домаћинстава.

Четврти тип (тип IV) – кластеровање на основу зоне у којој потрошач живи. И у овом типу кластеровања разликују се два кластера која деле потрошаче на оне који се налазе на територији града и оних у приградским насељима.



Слика 4-38: Расподела потрошача зависно од типа кластеровања [54]

Сходно приказаним типовима кластеровања, на сваки од кластера формирана је засебна мрежа, при чему се у сваком случају компаративно примењују два модела (модел *А* и модел *Б* - фаза 03, Слика 4-36). Ради утврђивања ефикасности модела у кластерима паралелно ће биће дати и резултати добијени над базним скупом.

За формирање модела искоришћена су сазнања стечена истраживањем приказаном у претходном поглављу (4.3.2). Позитиван утицај који су WNL и LNL скривени слојеви показали и раније, искоришћен је и за архитектуру овог модела уз додатак још једног скривеног слоја. Ранији налази су показали да је за решавање проблема предикције потрошње електричне енергије потребно користити слојеве са већим бројем неурона. Ипак, да би се формирао универзалан модел погодан за предикције у различитим скуповима података (различитих у смислу структуре обележја, а самим тим и броја неурона) треба узети у обзир више фактора за постизање оптималних услова.

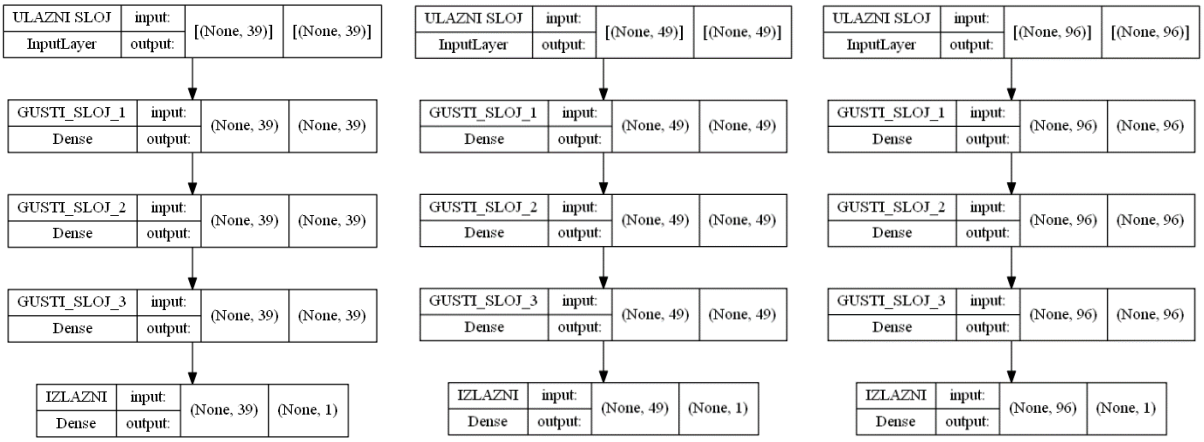
Један од најважнијих фактора је одређивање броја неурона у скривеним слојевима. За модел приказан у овом поглављу тај број (n_{hid}) одређен је у зависности од броја неурона у улазном слоју (n_{input}) (36):

$$n_{hid} = n_{input}. \tag{36}$$

На овај начин ће број неурона у скривеним слојевима бити одређен сразмерно броју неурона на улазу у мрежу. Разлог за то потиче из потребе да се за различите скупове података са различитим бројем неурона на улазу, одреди фер пропорција броја неурона у осталим слојевима (а у складу са налазима аутора у [145], [146]). Иако на први поглед не делује значајно, број неурона на улазу значајно варира од кластера до кластера. Ово се посебно истиче ако се посматра деловање модела над кластерима који садрже податке о само једном месецу у години у односу на кластер једног годишњег доба или у односу на базни скуп података. За постојање поменутих разлога главни кривац су категоријске променљиве које ће у зависности од типа кластеровања имати различите последице у виду броја могућих категорија. Једно такво обележје је ознака обрачунског периода чији је садржај веома важан за предикцију и може имати више различитих облика (тј. комбинација). У складу са тим, постоји неколико могућих случајева:

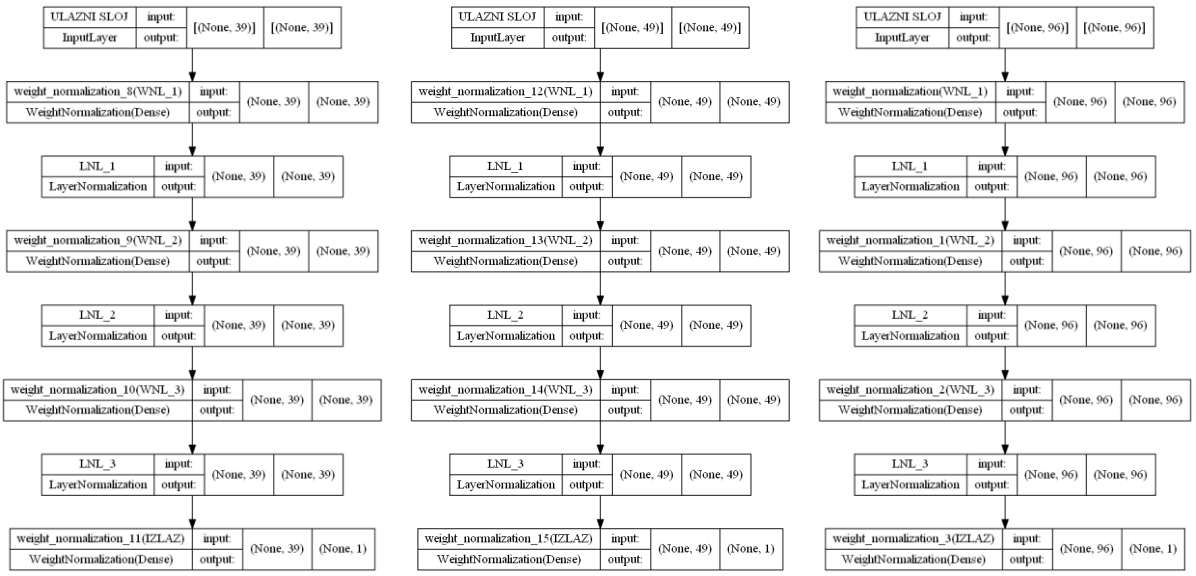
- 1. случај – ако се посматра кластер за један месец (у овом случају кластер *јануар*), конкретно обележје ће садржати 4 могуће категорије (по један јануар за сваку од четири посматране године), а самим тим након кодирања добијају се 4 обележја на месту једног;
- 2. случај - ако се посматра кластер за једно годишње доба, конкретно обележје ће садржати 12 различитих категорија (три месеца која обухвата једно годишње доба помножена са четири године), тачније нових 12 обележја након извршеног кодирања (односно 8 неурона више на улазу у односу на 1. случај);
- 3. случај - ако се посматра базни скуп података где постоји 48 различитих категорија (свака од посматране четири године помножена са 12 месеци), што доводи до 48 нових обележја након кодирања. То значи и 44 обележја више него у првом и 36 више него у другом случају и исто толико више неурона на улазу у мрежу.

На следеће две слике (Слика 4-39 и Слика 4-40) дат је пример модела *А* и модела *Б* (респективно) са бројем параметара за три претходно поменута случаја: за кластер једног месеца (а), кластер једног годишњег доба (б) и за базни скуп у целости (в).



(а) (б) (в)

Слика 4-39: Модел за кластер једног месеца (а), кластер једног годишњег доба (б) и за скуп у целости (в) – модел *А*



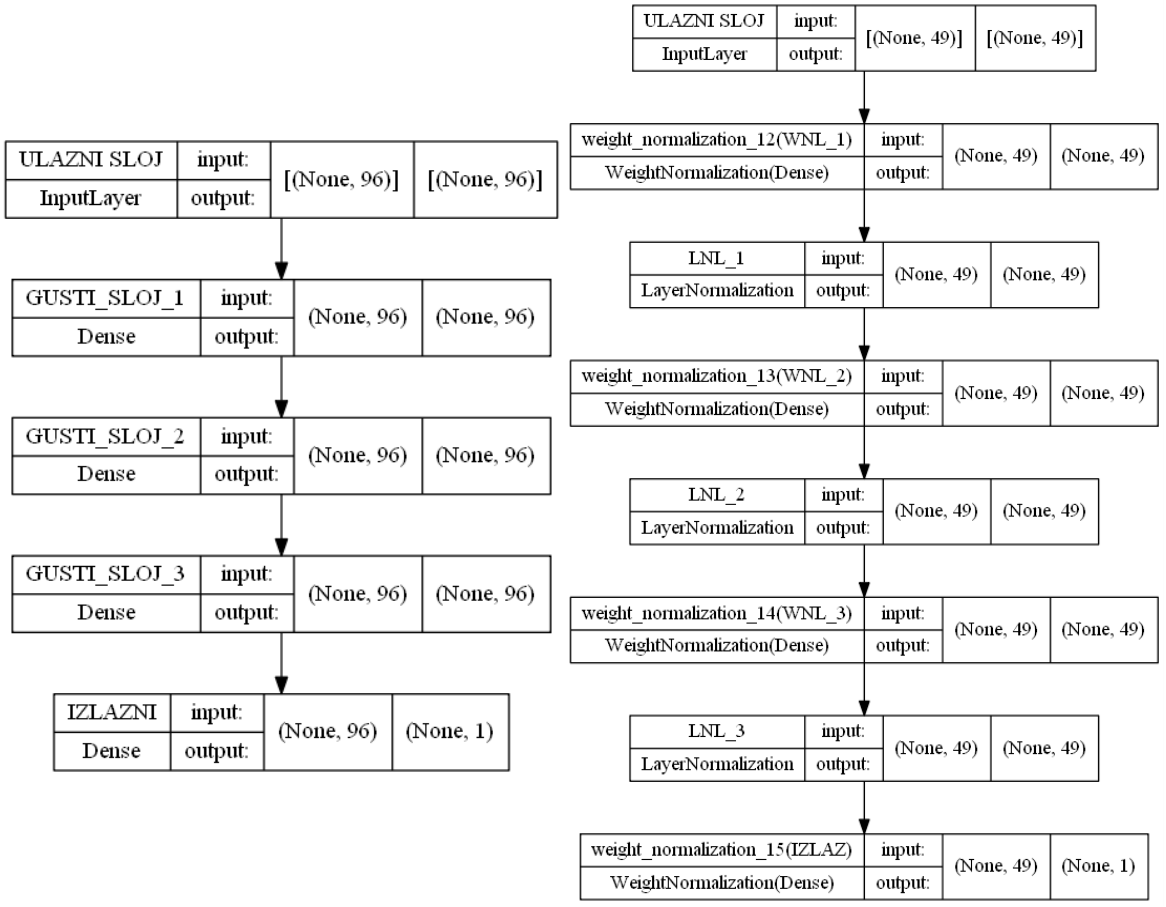
(а) (б) (в)

Слика 4-40: Модел за кластер једног месеца (а), кластер једног годишњег доба (б) и за скуп у целости (в) – модел *Б*

Јасно се види разлика у броју неурона у улазним али и скривеним слојевима између три приказана случаја па у том смислу резултати не би били упоредиви формирањем универзалног модела са унапред, фиксно одређеним бројем неурона за скривене слојеве. Постављањем броја неурона на 100 у сваком скривеном слоју, мрежа за 1. случај би имала три пута више неурона у скривеним слојевима него на самом улазу што би довело до структуре *обрнутог троугла* што је супротно препорукама аутора из [145], [146].

Иста законитост важи и за остале типове кластеровања (у односу на групу или категорију потрошача). Према томе, за сваку од мрежа ће број неурона у скривеним слојевима бити одређен зависно од структуре улазних неурона. Тако ће број јединица у скривеним слојевима бити различит за сваки тип кластеровања, а сам модел ће бити универзалан.

Архитектура мреже са бројем параметара, за случај са највећим бројем неурона на улазу, дата је на следеће две слике (Слика 4-41 и Слика 4-42). Поново је приметан нешто већи број тренираних параметара код модела са WNL и LNL слојевима (модел *Б*) који поново није у потпуности тачан обзиром да се неурони из LNL слојева штампају као параметри који се тренирају.[9]



(а) (б)

Слика 4-41: Модел *А* (а) и модел *Б* (б) - структура слојева

[9] Као и раније када је било више речи о броју параметара сваког од модела (Слика 4-29) и у овом случају важи исти принцип.

Layer (type)	Output Shape	Param #
ULAZNI SLOJ (InputLayer)	[(None, 96)]	0
GUSTI_SLOJ_1 (Dense)	(None, 96)	9312
GUSTI_SLOJ_2 (Dense)	(None, 96)	9312
GUSTI_SLOJ_3 (Dense)	(None, 96)	9312
IZLAZNI (Dense)	(None, 1)	97

Total params: 28,033
Trainable params: 28,033
Non-trainable params: 0

(а)

Layer (type)	Output Shape	Param #
ULAZNI SLOJ (InputLayer)	[(None, 96)]	0
weight_normalization (WeightNormalization)	(None, 96)	18721
LNL_1 (LayerNormalization)	(None, 96)	192
weight_normalization_1 (WeightNormalization)	(None, 96)	18721
LNL_2 (LayerNormalization)	(None, 96)	192
weight_normalization_2 (WeightNormalization)	(None, 96)	18721
LNL_3 (LayerNormalization)	(None, 96)	192
weight_normalization_3 (WeightNormalization)	(None, 1)	196

Total params: 56,935
Trainable params: 28,898
Non-trainable params: 28,037

(б)

Слика 4-42: Програмска штампа структуре слојева са бројем параметара: модел *А* (а) и модел *Б* (б)

У наставку ће табеларно бити приказани резултати предикције за оба модела наизменично и за сваки од поменутих типова кластеровања. У табелама су плавом бојом истакнуте најповољније просечне апсолутне грешке док су њима супротне означене црвеном бојом.

Важно је напоменути да ће модели (*А* и *Б*) бити тестирани увек под истим околностима (исти број скривених слојева, стопа учења, величина серије, функција оптимизације, итд.). Ово је посебно значајно јер се на тај начин тестира прилагодљивост предложеног модела како различитим скуповима података тако и њиховој величини која знатно варира (Слика 4-38).

4.4.1. Први тип кластеровања

У случају првог типа кластеровања сваки кластер садржи податке о потрошњама из једног месеца што ће обухватити потрошње за све постојеће потрошаче, за четири посматране године. Након чишћења података и одбацивања ирелевантних података (како је објашњено у потпоглављу 4.2) у првој фази предобраде података, сваки од кластера има приближно једнак број очитавања бројила, (између 8 и 9% укупног броја очитавања), Слика 4-38.

Табела 5 показује оцене предвиђања оба модела (*А* и *Б*) кроз вредности четири процентуално изражене метрике (*RMSE(%), MAPE, MAE* и AR^2) добијене тестирањем модела над тест скупом података (подаци из последње године посматраног периода).

Табела 5: Поређење метрика за први тип кластеровања [54]

МЕТРИКА	ANN ТИП	ЈАН	ФЕБ	МАР	АПР	МАЈ	ЈУН	ЈУЛ	АВГ	СЕП	ОКТ	НОВ	ДЕЦ	*БАЗНИ СКУП*
RMSE (%)	А	7,89	8,48	8,06	8,61	9,8	10,12	10,11	10,24	9,33	9,1	8,17	8,39	*8,34*
	Б	7,2	7,4	6,98	6,77	8,61	8,01	7,72	8,66	7,97	7,62	7,42	6,79	*7,9*
MAPE (%)	А	6,29	6,65	6,19	6,68	7,51	8,3	8,39	**8,63**	7,13	6,8	6,37	6,43	*6,2*
	Б	5,56	5,42	5,07	**4,79**	5,87	5,81	5,75	5,23	5,74	5,5	5,09	4,83	*5,68*
MAE (%)	А	5,92	6,18	5,81	6,27	7,09	7,57	7,71	7,96	6,7	6,47	5,83	6,18	*5,92*
	Б	5,3	5,22	4,84	4,74	5,77	5,65	5,57	6,63	5,58	5,33	4,86	4,72	*5,46*
AR^2 (%)	А	95,26	96,46	96,88	95,48	94,65	93,81	90,96	69,72	95,13	95,73	97,35	96,31	*95,67*
	Б	96,12	97,34	97,68	97,16	96,12	96,14	94,78	98,02	96,45	97,01	97,95	97,61	*96,25*

Претходна табела даје преглед метрика добијених тестирањем модела за сваки појединачни месец у години. Поредећи са резултатима из базног скупа за оба поређена модела (модел *А* и модел и *Б*) приметно је да знатно повољније вредности даје модел *Б*.

У том смислу, посебно треба истаћи вредности прилагођеног коефицијента детерминације који је приметно виши када је модел *Б* у питању. Вредности средње процентуалне грешке за све потрошаче и за сваки појединачни месец повољнија код модела *Б* него код модела *А*. Највећа просечна грешка се добија за месец август (8,63%) моделом типа *А*, док је најповољнија вредност постигнута за месец април у износу од 4,79%, моделом типа *Б*. Такве резултате прате и вредности прилагођеног коефицијента детерминације приказане у последња два реда претходне табеле.

4.4.2. Други тип кластеровања

Резултат примене другог типа кластеровања су четири кластера за четири годишња доба. За разлику од уобичајене поделе на годишња доба, ради усклађивања са месечним обрачунима потрошњи, овде је извршена расподела како је раније приказано у поглављу 3, Слика 4-7. Резултати добијени моделима *А* и *Б* за други тип кластеровања дати су у Табели 6.

Табела 6: Поређење метрика за други тип кластеровања [54]

МЕТРИКА	ANN ТИП	ЛЕТО	ЈЕСЕН	ЗИМА	ПРОЛЕЋЕ
RMSE (%)	А	9,83	8,20	7,87	9,03
	Б	7,42	7,75	6,92	6,80
MAPE (%)	А	**7,93**	6,20	5,79	6,97
	Б	5,47	4,89	**4,81**	5,28
MAE (%)	А	7,27	5,78	5,57	6,51
	Б	5,13	5,34	4,70	4,72
AR^2 (%)	А	93,42	96,67	96,60	95,12
	Б	96,75	95,97	97,66	97,45

Као и код претходног типа кластеровања, прецизнији је модел типа *Б*, дајући најмању просечну грешку од 4,81%, док су вредности коефицијента AR^2 нешто ниже у односу на први тип кластеровања.

4.4.3. Трећи тип кластеровања

Према трећем типу кластеровања, сви потрошачи се деле у два кластера разликујући потрошаче који електричну енергију користе искључиво за потребе у домаћинству и оне који исту користе за друге потребе (недомаћинства, правни субјекти, итд).

Према графику са слике Слика 4-38, више од 95% потрошача на подручју града Ужица чине домаћинства, а преостало су потрошачи из групе недомаћинства.

Табела 7 показује оцене предвиђања оба модела (*А* и *Б*) кроз четири метрике (*RMSE(%), MAPE, MAE* и AR^2) за кластере који разликују потрошаче у домаћинствима и оне ван њих.

Табела 7: Поређење метрика за трећи тип кластеровања [54]

МЕТРИКА	ANN ТИП	НЕ-ДОМАЋИНСТВА	ДОМАЋИНСТВА
RMSE (%)	А	16,57	12,52
	Б	9,30	7,09
MAPE (%)	А	**14,85**	10,74
	Б	6,58	**4,86**
MAE (%)	А	12,29	9,26
	Б	6,10	4,75
AR^2 (%)	А	63,53	70,52
	Б	95,89	96,88

У случају треће поделе на кластере приметна је велика разлика у бројности ове две групе потрошача. Више од 95% потрошача у домаћинствима значи и већу масовност у кластеру, а самим тим у другом кластеру са знатно мање података долази до потешкоћа у предикцији што се директно одражава и на посматране метрике. Отуда и јасна разлика у вредностима приказане грешке која је виша у кластеру за недомаћинства али и даље, поредећи моделе међусобно, модел *Б* је знатно успешнији од модела *А*.

4.4.4. Четврти тип кластеровања

Четврти тип кластеровања прави разлику између потрошача који живе у градској и оних који су у приградској средини. Обзиром да се и у овој подели уочава велика разлика у бројности потрошача (преко 70% потрошача живи у градској средини) очекивано је да и резултати предикције буду слични онима из претходног типа кластеровања.

Табела 8 показује оцене предвиђања оба модела (*А* и *Б*) кроз четири метрике (*RMSE(%), MAPE, MAE* и AR^2) добијене над скупом података из последње године посматраног периода за кластере са потрошачима из приградске и градске средине.

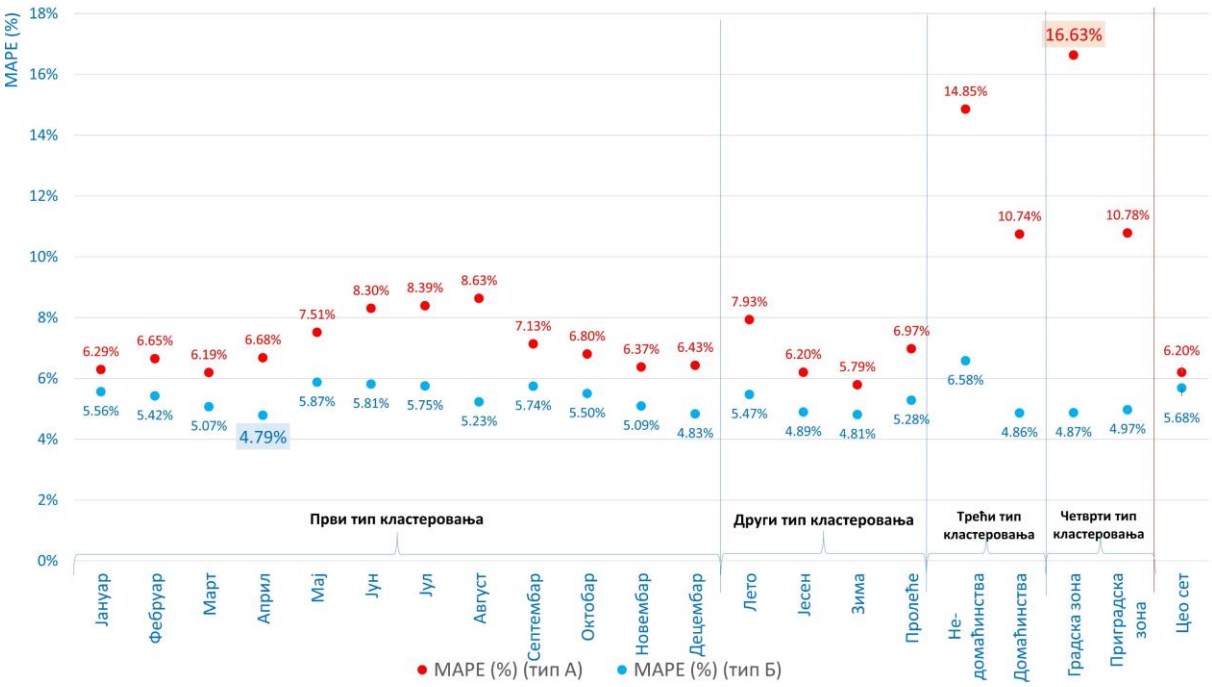
Резултати поново истичу модел *Б* као знатно успешнији. Поредећи резултате у кластерима међусобно, за кластер потрошача у градској средини модел типа *А* даје лошије резултате док је истовремено модел *Б* за исти тај кластер знатно прецизнији.

Табела 8: Поређење метрика за четврти тип кластеровања [54]

МЕТРИКА	ANN ТИП	ГРАДСКА ЗОНА	ПРИГРАДСКА ЗОНА
RMSE (%)	А	17,64	13,79
	Б	6,93	7,53
MAPE (%)	А	**16,63**	10,78
	Б	**4,87**	4,97
MAE (%)	А	13,26	9,84
	Б	4,70	4,92
AR^2 (%)	А	91,44	91,23
	Б	96,88	96,61

4.4.5. Прецизност предложеног модела

Јасно је да предложени модел (модел *Б*) има велику прецизност у предвиђању месечних потрошњи а како би резултати били јаснији, дијаграм на следећој слици (Слика 4-43) илуструје добијене вредности средње апсолутне грешке.

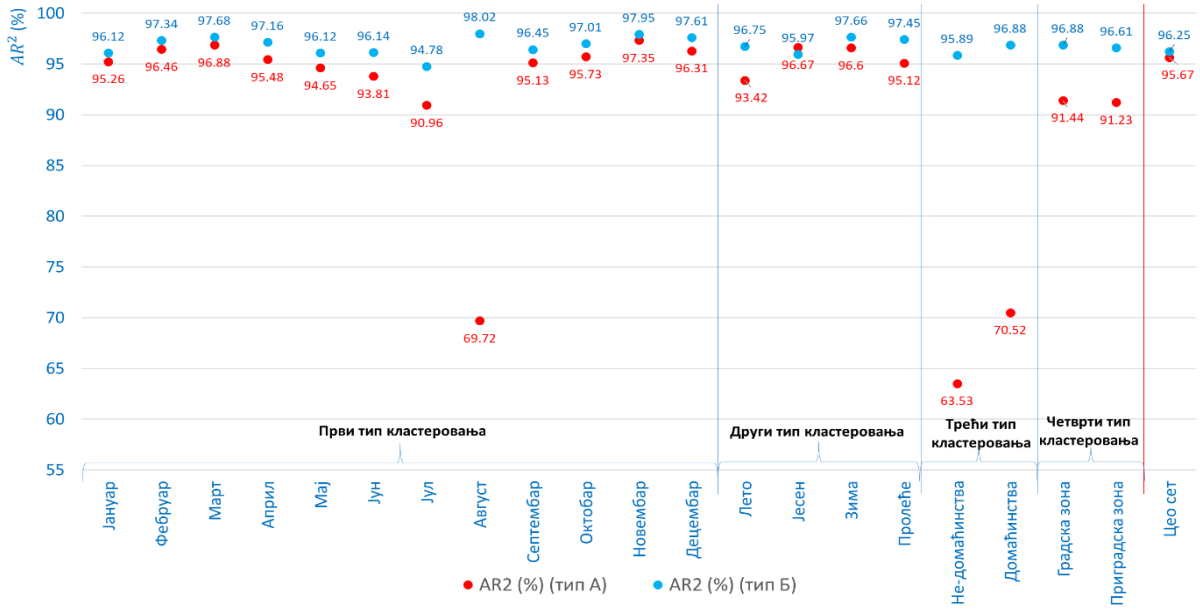


Слика 4-43: Средње вредности MAPE(%) за све потрошаче у тест скупу (упоредо за сваки кластер и скуп у целости), добијене помоћу модела типа *А* и модела типа *Б* [54]

На дијаграму, црвеним тачкама су обележени резултати добијени моделом ANN типа *А*, док плаве представљају вредности добијене мрежом типа *Б*. На *x* оси су ознаке кластера, а на *y* оси процентуална средња грешка. Крајње десно на дијаграму су резултати добијени моделима у базном скупу података.

Приметно је да је модел типа *Б* прецизнији али и сигурнији у односу на модел *А*, дајући мирнију расподелу тачака на графику.

Поред просечне грешке значајан показатељ успешности модела пружа AR^2 коефицијент показујући ефикасност особина потрошача у објашњавању потрошње електричне енергије. Следећа слика (Слика 4-44) илуструје вредности коефицијента детерминације за све кластере и оба модела.



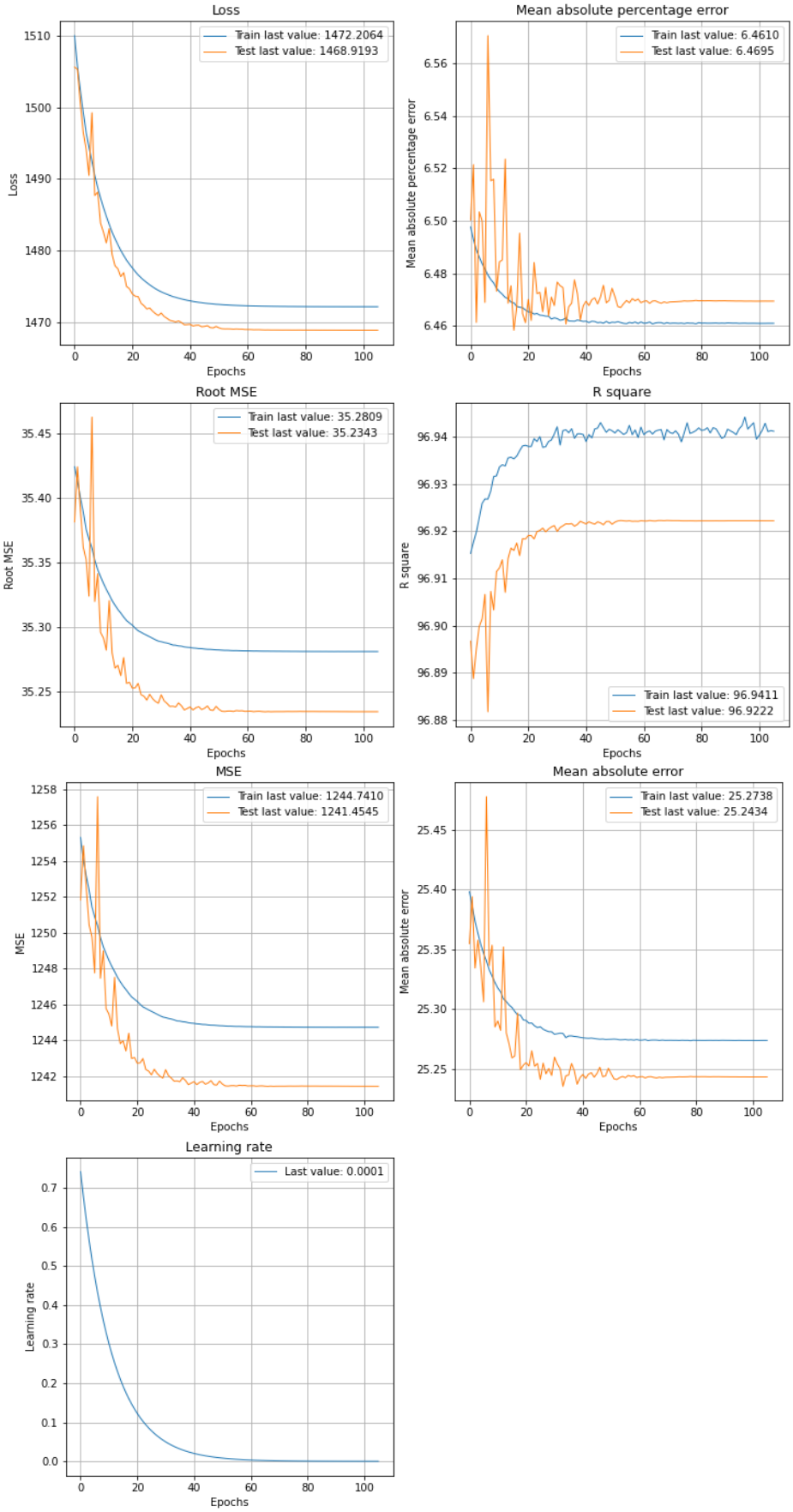
Слика 4-44: Прилагођени коефицијент детерминације (AR^2) за све кластере и моделе (*А* и *Б*)

Поново, црвене тачке представљају вредности коефицијента које постиже модел *А*, а плаве модел *Б*. У складу са раније приказаним резултатима, вредности добијене моделом *Б* поново се истичу као повољније.

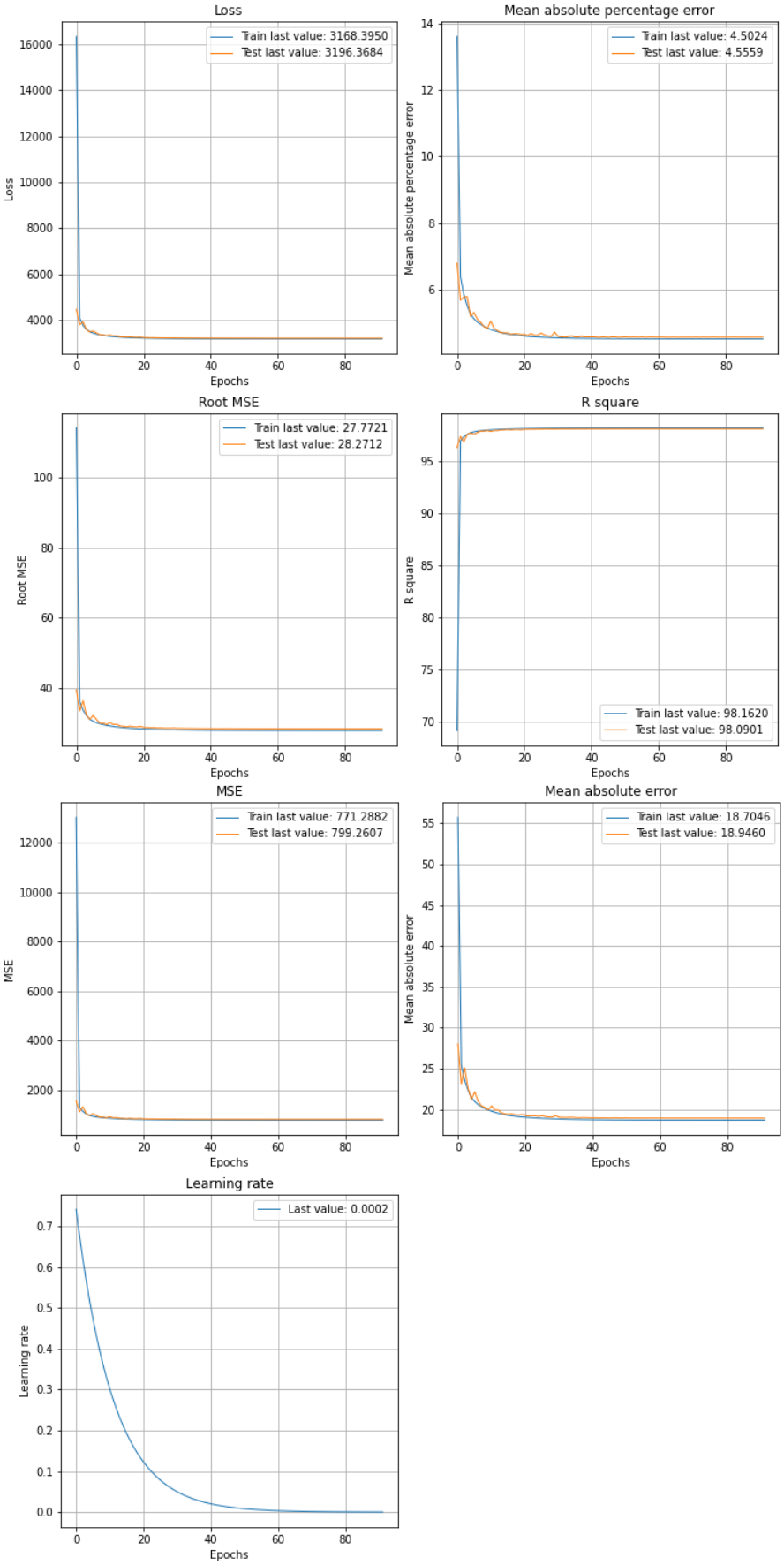
Ово потврђује раније наметнут закључак да је предложени модел (модел *Б*) не само успешнији у предвиђању у погледу прецизности исказану кроз висине грешке већ и по питању његове конзистентности и сигурности. У прилог успешности модела *Б* говори и крива сваке посматране метрике оцене квалитета модела (Слика 4-45 и Слика 4-46) праћене током једног процеса тренинга модела *А* и модела *Б*.

Поново је приметна разлика у понашању дијаграма који илуструју ова два модела. Можда и више него раније јасна је несигурност модела *А* (Слика 4-45) у свакој од метрика, а посебно када дође у контакт са новим, до тада непознатим подацима.

Насупрот томе, модел *Б* (Слика 4-46) показује сигурност и конзистентност за сваку од посматраних метрика уз готово потпуно поклапање линија криве која описује вредности метрика на тренинг и валидационом скупу података.

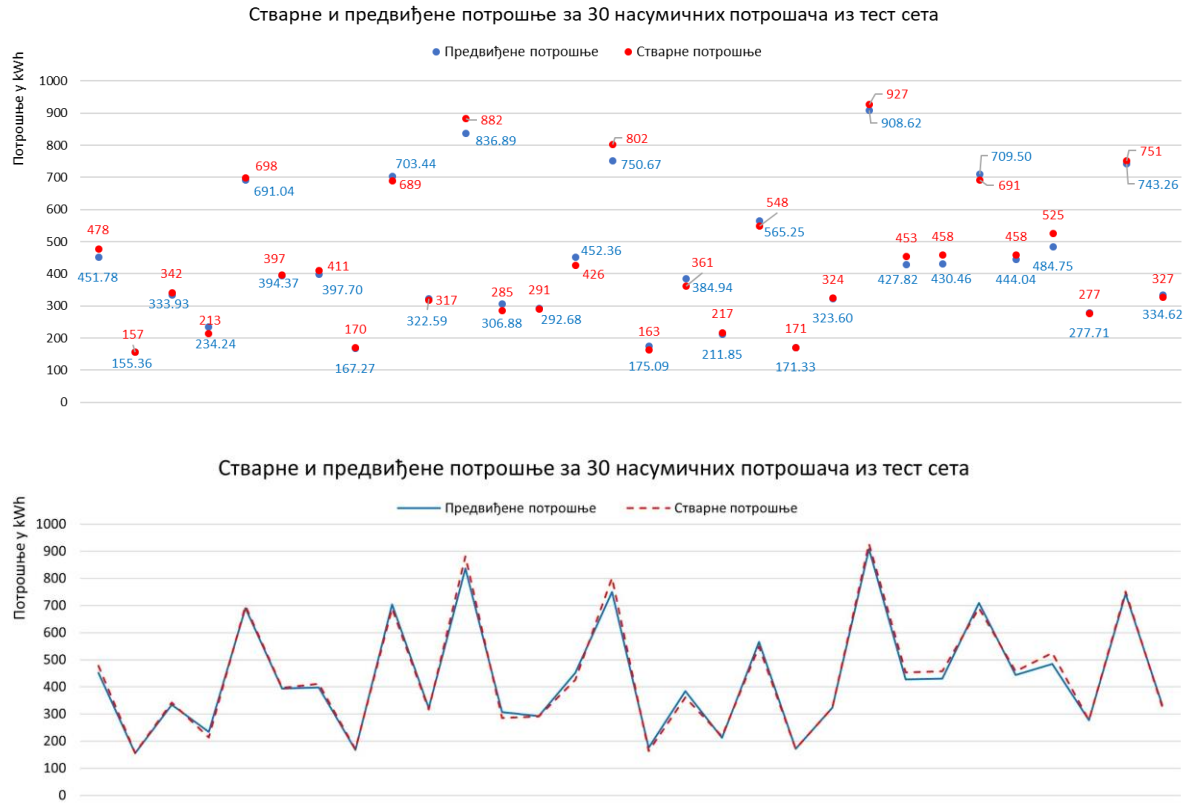


Слика 4-45: Дијаграми праћених метрика током процеса обучавања модела *A*



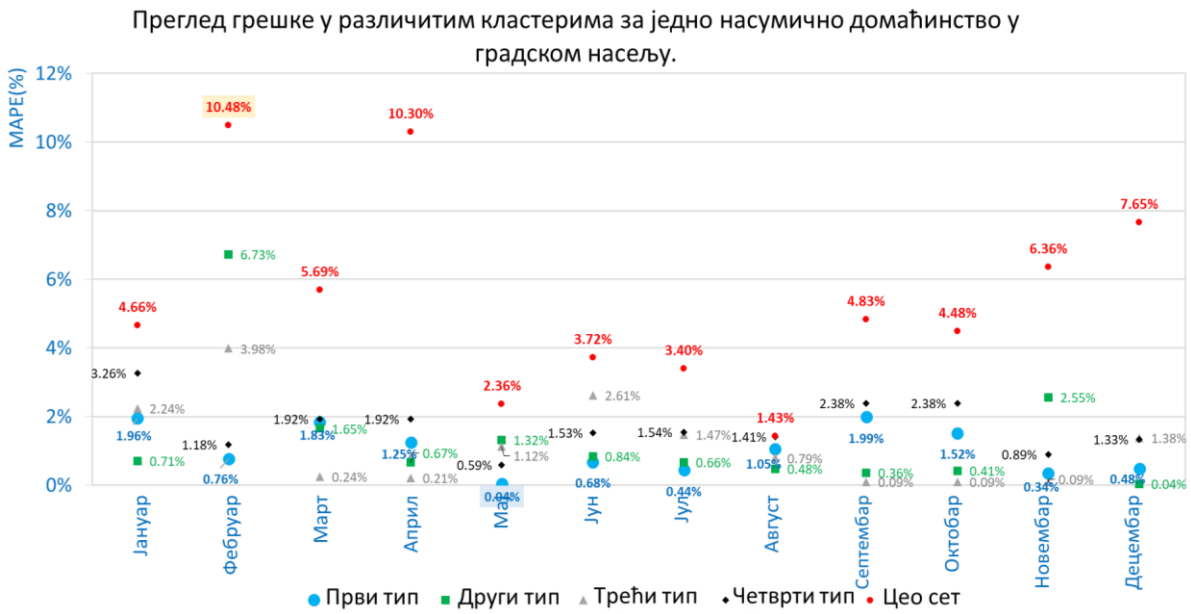
Слика 4-46: Дијаграми праћених метрика током процеса обучавања модела *Б*

Да су вредности које предвиђа модел заиста толико близу стварним потрошњама показује и упоредни дијаграм стварних и предикованих вредности дат на следећој слици (Слика 4-47). Црвене тачке на дијаграму представљају стварне, измерене потрошње на бројилима потрошача док плаве (респективно) представљају вредности по моделу. Иако дијаграм показује тек мали број потрошача (30 у односу на преко 300.000 потрошача у тест сету), јасно илуструје да су одступања модела минимална, а код појединих потрошача и неприметна. Све то потврђује велику прецизност предложеног модела у предвиђању месечних потрошњи електричне енергије и указује на оправданост његове примене.



Слика 4-47: Упоредни приказ стварних потрошњи и потрошњи по моделу за 30 насумично одабраних потрошача

Како ће у реалном систему често постојати потреба за предвиђањем потрошње електричне енергије појединачног потрошача, а не само као групе потрошача, на следећој слици (Слика 4-48), приказан је управо такав случај.



Слика 4-48: Средње вредности MAPE(%) за једног насумичног потрошача, добијене помоћу модела типа *Б*, за сваки тип кластеровања и скуп у целости [54]

Слика 4-48 илуструје вредности грешке за једног, насумично одабраног потрошача из категорије домаћинстава које се налази у градској зони. Тачке обележене црвеном бојом представљају апсолутне грешке у предикцији потрошњи за конкретног потрошача, добијене тренирањем и тестирањем модела типа *Б* над читавим скупом. Приметно је да су те вредности углавном на највишим позицијама на графику за сваки месец у години.

Јасно се види да сваки тип кластеровања (на графику представљени симболима плаве, зелене, сиве и црне боје), омогућује прецизније прогнозе у односу на оне у читавом скупу („цео сет“). При томе, у готово сваком месецу, први тип кластеровања (светло плави маркер) даје најбоље резултате. Вредности грешке које показују ови маркери падају испод 2%, а чак у половини месеци су ниже од 1%. Овим је још једном потврђено да предложени модел (модел *Б*) најмање греши при примени првог типа кластеровања, што га чини погодним не само у маси, већ и у случају предикција за индивидуалног потрошача.

Резултати представљени у овом поглављу потврђују пету посебну хипотезу по којој се формирањем хомогенијих група потрошача могу превазићи проблеми настали као последица хетерогености потрошача електричне енергије на неком подручју.

Кôд развоја модела А и модела В дати су у виду прилога (Прилог 9 и Прилог 10).

4.4.6. Поређење резултата ANN модела са другим ML моделима

Иако су ANN у више наврата показале супериорност над другим поменутим техникама ML у наставку ће бити приказани резултати које ове технике постижу за сваки од типова кластеровања. Табеле 9 до 12 дају упоредне вредности сваке од посматраних метрика (*RMSE(%), MAPE, MAE и* AR^2) за пет раније поменутих алгоритама ML у односу на вредности истих метрика постигнуте помоћу ANN модела.

Први тип кластеровања

Табела 9: Поређење резултата других техника ML са резултатима предложеног модела за први тип кластеровања

	МЕТРИКА	ЈАН	ФЕБ	МАР	АПР	МАЈ	ЈУН	ЈУЛ	АВГ	СЕП	ОКТ	НОВ	ДЕЦ
RMSE (%)	Линеарна регресија	8,47	-	9,26	9,21	12,02	11,84	11,08	-	11,65	10,69	10,26	-
	Градијент буст (XGBoost)	6,8	7,23	7,01	7,15	8,63	8,51	8,41	7,80	8,57	7,86	6,99	7,19
	Линеарни SVR	8,45	9,45	9,26	9,25	11,95	12,15	11,11	40,00	11,87	10,63	10,09	9,41
	Хубер регресија	8,54	9,41	9,27	9,26	11,96	12,09	11,13	40,15	11,85	10,69	10,05	9,45
	Стабла одлуке	9,54	10,49	10,08	10,01	12,02	11,97	11,72	10,78	11,80	11,47	9,83	10,48
	Предложени модел ANN	3,50	3,32	4,96	4,52	5,00	4,63	3,65	2,95	4,05	4,61	4,37	4,07
MAPE (%)	Линеарна регресија	6,69	-	7,25	7,26	9,67	9,56	9,27	-	9,33	8,51	8,69	-
	Градијент буст (XGBoost)	4,71	4,19	4,97	6,91	6,15	6,12	6,17	5,59	5,91	5,58	5,05	5,14
	Линеарни SVR	6,65	7,33	7,20	7,26	9,35	9,83	9,36	41,64	9,33	8,39	7,68	7,55
	Хубер регресија	6,76	7,33	7,23	7,28	9,38	9,77	9,35	41,86	9,29	8,49	7,73	7,58
	Стабла одлуке	6,27	7,08	6,62	6,25	8,13	8,07	8,17	7,36	7,75	7,52	6,64	7,09
	Предложени модел ANN	2,57	2,40	3,54	3,22	3,27	2,90	2,58	1,90	2,73	3,33	2,93	3,00
MAE (%)	Линеарна регресија	6,34	-	6,72	6,72	8,74	8,62	8,34	-	8,42	7,75	7,52	-
	Градијент буст (XGBoost)	4,74	5,03	4,87	4,94	5,98	5,94	5,93	5,51	5,83	5,45	4,81	4,99
	Линеарни SVR	6,31	6,79	6,67	6,71	8,55	12,16	8,36	38,45	8,48	7,66	7,08	6,89
	Хубер регресија	6,37	6,78	6,69	6,73	8,57	8,76	8,38	38,59	8,45	7,72	7,08	6,91
	Стабла одлуке	6,38	6,98	6,50	6,67	8,01	7,99	8,01	7,32	7,74	7,53	6,45	6,97
	Предложени модел ANN	2,27	2,15	3,25	3,00	3,12	2,80	2,46	1,79	2,60	3,07	2,70	2,72
AR^2 (%)	Линеарна регресија	95,07	-	96,38	95,25	9,14	92,53	90,98	-	93,32	94,81	96,56	-
	Градијент буст (XGBoost)	96,83	97,76	97,93	97,14	96,46	96,14	94,3	93,96	96,39	97,20	98,40	97,61
	Линеарни SVR	95,1	96,18	96,38	95,21	93,22	92,14	90,05	-	93,08	94,87	96,68	95,90
	Хубер регресија	96,83	97,76	97,93	97,14	96,46	96,14	94,3	93,96	96,39	97,20	98,40	97,61
	Стабла одлуке	93,75	95,29	95,77	94,41	93,13	92,37	88,92	88,47	93,16	94,03	96,84	94,23
	Предложени модел ANN	99,43	99,49	98,94	98,88	98,78	98,93	99,11	99,46	99,46	99,01	99,26	99,14

Други тип кластеровања

Табела 10: Поређење резултата других техника ML са резултатима предложеног модела за други тип кластеровања

	МЕТРИКА	ЛЕТО	ЈЕСЕН	ЗИМА	ПРОЛЕЋЕ
RMSE (%)	Линеарна регресија	11,91	9,85	9,15	11,06
	Градијент буст (XGBoost)	8,08	6,99	6,85	7,81
	Линеарни SVR	12,49	9,93	9,23	11,16
	Хубер регресија	12,47	9,91	9,23	11,18
	Стабла одлуке	11,07	9,98	9,53	11,17
	Предложени модел ANN	5,12	5,34	5,38	5,49
MAPE (%)	Линеарна регресија	9,17	7,83	7,13	8,91
	Градијент буст (XGBoost)	5,68	5,03	4,84	5,62
	Линеарни SVR	9,51	7,78	7,17	8,98
	Хубер регресија	9,49	7,82	7,18	8,99
	Стабла одлуке	7,30	6,60	6,18	7,28
	Предложени модел ANN	3,41	3,73	3,77	3,85
MAE (%)	Линеарна регресија	8,42	7,12	6,65	8,06
	Градијент буст (XGBoost)	5,55	4,84	4,75	5,46
	Линеарни SVR	8,69	7,12	6,68	8,11
	Хубер регресија	8,68	7,12	6,68	5,11
	Стабла одлуке	7,19	6,49	6,19	7,28
	Предложени модел ANN	3,30	3,48	3,51	3,65
AR^2 (%)	Линеарна регресија	92,19	96,22	96,25	93,88
	Градијент буст (XGBoost)	96,41	98,09	97,90	96,95
	Линеарни SVR	91,41	96,17	96,19	93,77
	Хубер регресија	96,40	98,09	97,91	96,95
	Стабла одлуке	93,26	96,12	95,94	93,75
	Предложени модел ANN	98,51	98,70	98,66	98,39

Трећи тип кластеровања

Табела 11: Поређење резултата других техника ML са резултатима предложеног модела за трећи тип кластеровања

	МЕТРИКА	НЕДОМАЋИНСТВА	ДОМАЋИНСТВА
RMSE (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	10,02	6,85
	Линеарни SVR	13,49	15,77
	Хубер регресија	13,39	15,86
	Стабла одлуке	13,86	9,01
	Предложени модел ANN	7,25	6,64
MAPE (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	3,64	4,77
	Линеарни SVR	14,67	17,05
	Хубер регресија	14,53	17,13
	Стабла одлуке	5,19	5,37
	Предложени модел ANN	5,15	4,55
MAE (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	3,74	4,70
	Линеарни SVR	11,11	13,49
	Хубер регресија	11,04	13,57
	Стабла одлуке	5,53	5,43
	Предложени модел ANN	4,85	4,46
AR^2 (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	96,45	98,05
	Линеарни SVR	93,57	89,70
	Хубер регресија	96,45	98,05
	Стабла одлуке	93,21	96,64
	Предложени модел ANN	98,05	97,89

Четврти тип кластеровања

Табела 12: Поређење резултата других техника ML са резултатима предложеног модела за четврти тип кластеровања

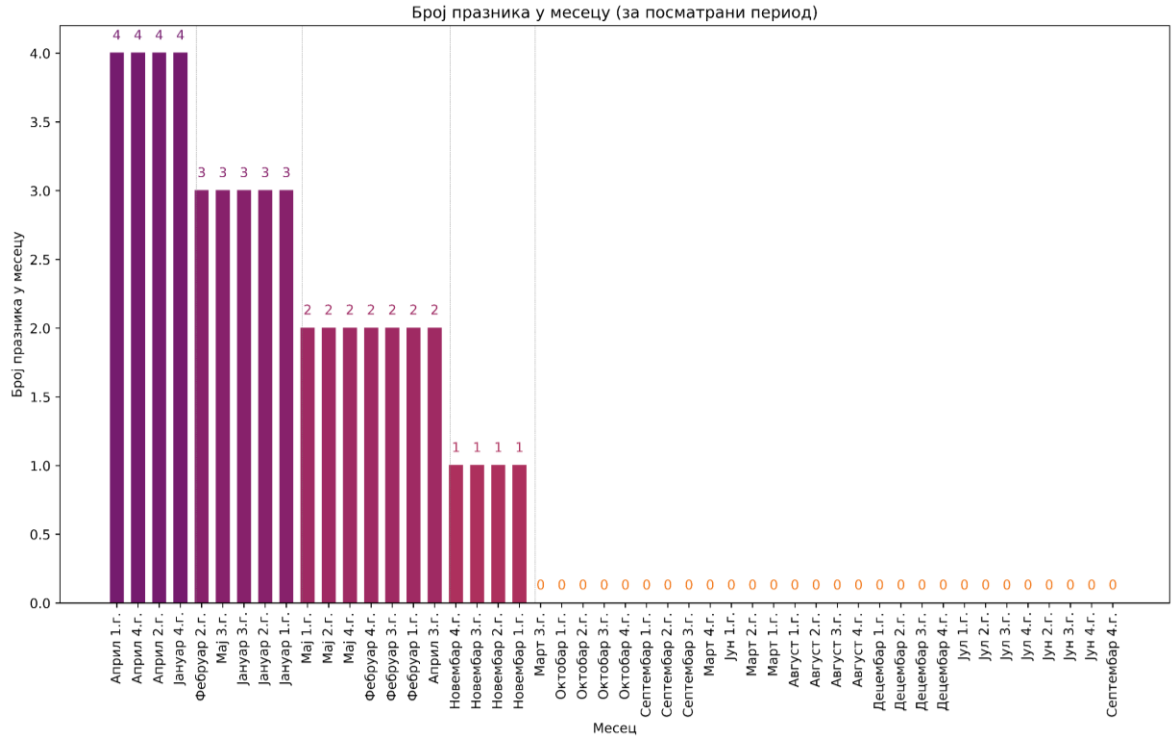
	МЕТРИКА	ГРАДСКА ЗОНА	ПРИГРАДСКА ЗОНА
RMSE (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	6,62	6,61
	Линеарни SVR	14,85	15,37
	Хубер регресија	14,86	15,39
	Стабла одлуке	8,75	9,03
	Предложени модел ANN	6,45	6,96
MAPE (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	4,71	4,58
	Линеарни SVR	15,82	16,23
	Хубер регресија	15,55	16,24
	Стабла одлуке	5,28	5,12
	Предложени модел ANN	4,56	4,52
MAE (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	4,85	4,50
	Линеарни SVR	12,59	12,89
	Хубер регресија	12,43	12,91
	Стабла одлуке	5,30	5,19
	Предложени модел ANN	4,41	4,59
AR^2 (%)	Линеарна регресија	-	-
	Градијент буст (XGBoost)	98,14	98,33
	Линеарни SVR	90,64	91,01
	Хубер регресија	98,14	98,33
	Стабла одлуке	96,74	96,89
	Предложени модел ANN	97,91	97,84

Посматрајући све приказане резултате у претходним табелама (Табела 9 - Табела 12) јасно се истиче модел ANN као супериорнији у односу на друге тестиране технике ML. Само у једном посматраном кластеру – *недомаћинства, трећи тип кластеровања,* Градијент буст (XGBoost) истиче као најбољи (Табела 11). Овај један, изоловани случај, не значи његову потпуну успешност. Посматрано у глобалу, предложени модел ANN је доста стабилнији што оправдава његову примену у било ком типу кластеровања.

4.4.7. Утицај празника на потрошње електричне енергије

Раније је приказан утицај неколико различитих особина потрошача и амбијента на месечну потрошњу електричне енергије. Постоји претпоставка да се током празника повећана флуктуација становништва може одразити на висину потрошње електричне енергије. Како се посматра градско подручје очекивано је да су потрошње током ових периода смањује. Постоји претпоставка да део становништва током ових дана не борави у граду, као и да део индустрије (потрошача из групе недомаћинства) делимично или потпуно престаје са радом. Тако је, са становишта висине месечне потрошње електричне енергије, занимљиво посматрати утицај празника на овај феномен.

У овом поглављу истражује се утицај празника (нерадних дана) на потрошњу електричне енергије и тестира хипотеза о могућности коришћења техника ML, конкретно, раније предложеног модела ANN, за њено предвиђање и током ових периода. Под термином празника с најпре подразумевају већи верски празници попут православног Божића и Ускрса. Такође, овде се убрајају и државни празници – нерадни дани на нивоу Републике Србије, које људи користе за одмор. Иако не за све, за велики део становништва ово су и званично нерадни дани. Слика 4-49 илуструје број таквих дана током свих месеци у посматраном периоду од четири године.

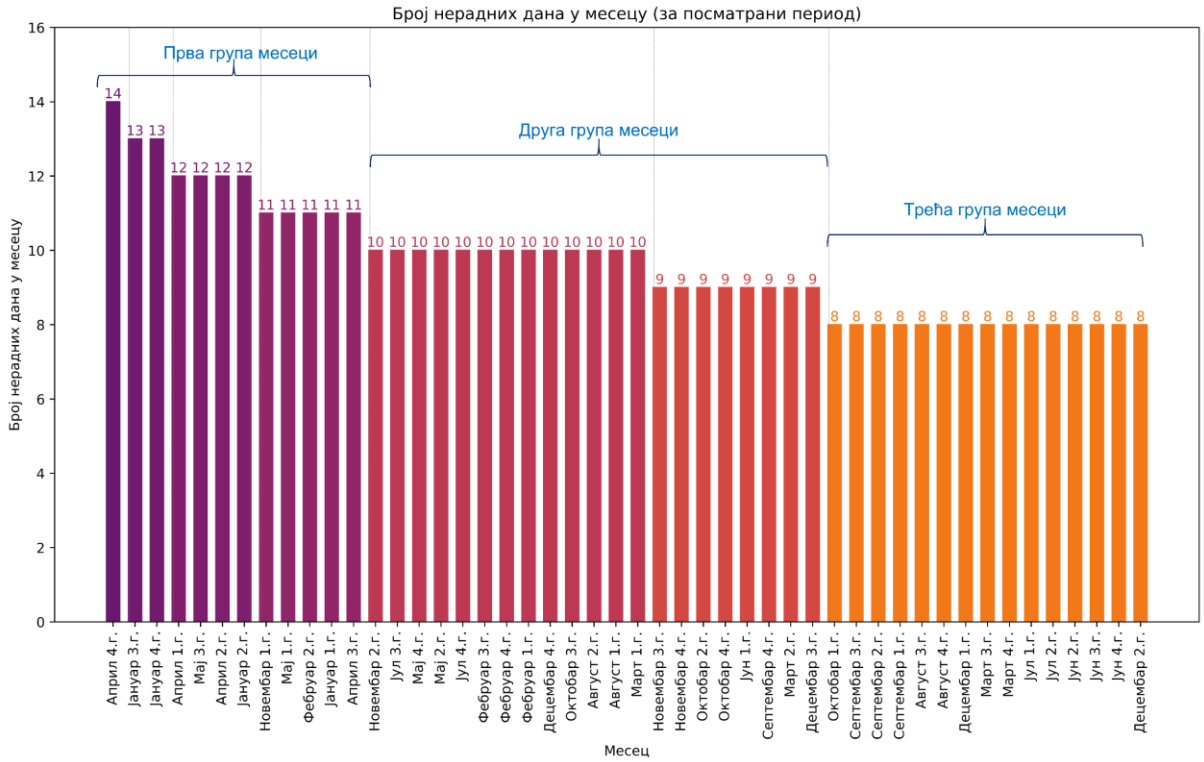


Слика 4-49: Број дана празника по месецима, за посматрани период од четири године

Април месец је карактеристичан као месец са највише празника због православног Ускрса када се празнује од петка до понедељка док јануар има нешто мање нерадних дана зависно од дана у недељи када пада православни Божић, итд.

Приметно је да постоји велики број месеци који немају ни један државни нити верски празник па су, у циљу што бољег сагледавања утицаја нерадних дана на потрошњу електричне енергије, празницима прикључени и дани викенда у сваком месецу. Критеријум је постављен на основу календара радних и нерадних дана доступног на

[137], за сваки месец у четири посматране године респективно. Тако је формирано ново обележје које раније није било присутно у скупу података („Нерадни дани“) и биће коришћено уместо раније присутних обележја за празнике и дане викенда. Слика 4-50 илуструје број нерадних дана по месецима узимајући у обзир и дане викенда.



Слика 4-50: Укупан број нерадних дана по месецима, за посматрани период од четири године

Поредећи претходне две слике (Слика 4-49 и Слика 4-50) примећује се да додати дани викенда не утичу пропорционално на све месеце па се добија другачија расподела нерадних дана. За анализу су занимљиви месеци са највише односно најмање нерадних дана. Међутим, како постоји само један месец са највећим бројем нерадних дана (април у првој посматраној години има чак 14 нерадних дана по датом критеријуму) док неколико месеци на другој страни има једнак, најмањи број (8 нерадних дана) потребно је успоставити равнотежу у погледу обимности месеци који ће бити подвргнути тестирању. У том циљу, а како би био узет у обзир што већи број потрошача, биће извршена подела на три велике групе:

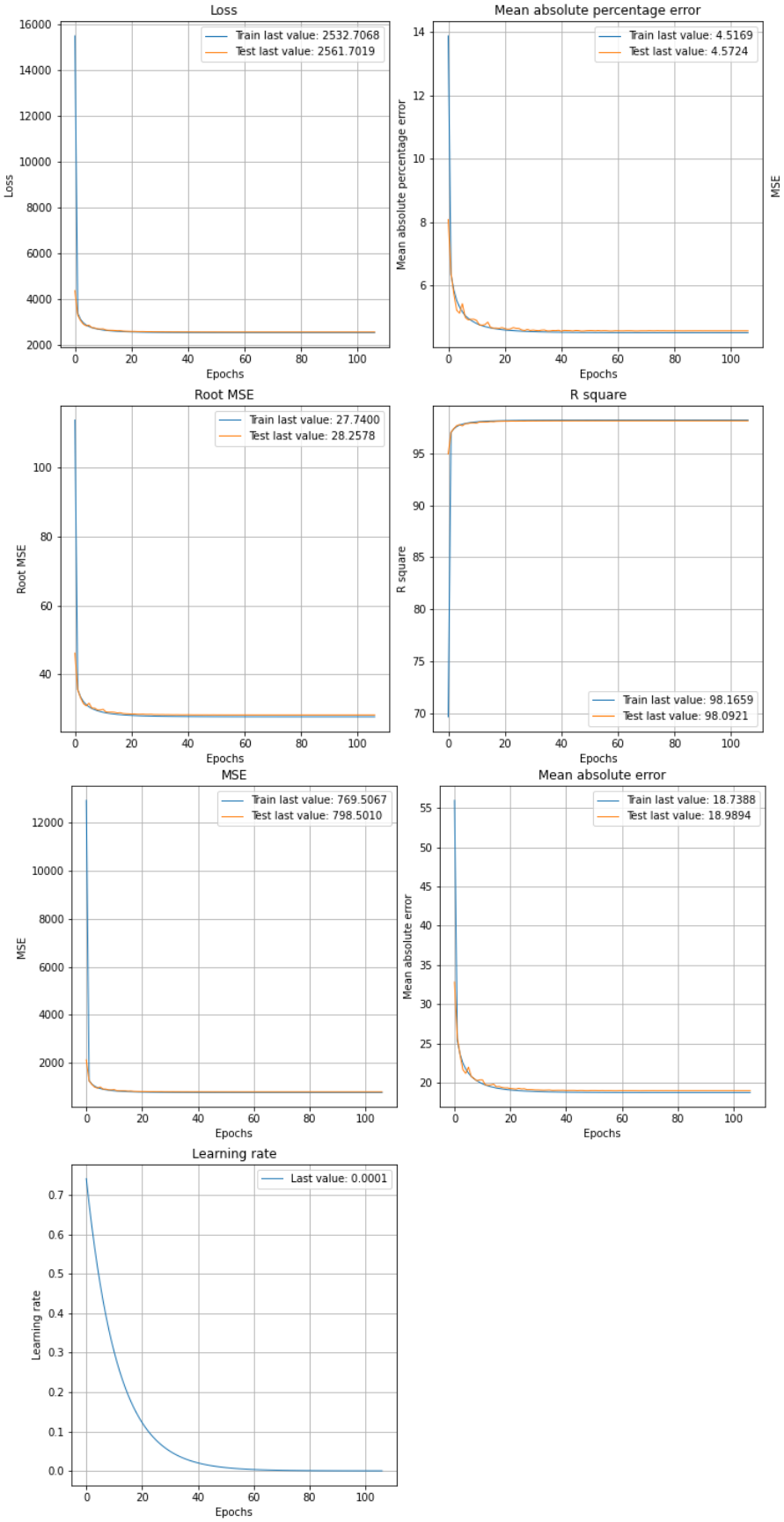
- прва група - месеци са највише нерадних дана (више од десет),
- друга група - месеци који имају десет или девет нерадних дана и
- трећа група - месеци са најмање нерадних дана (мање од девет).

Оваквом поделом се постиже најбоља равнотежа и самим тим очекују најреалнији резултати. За тестирање утицаја нерадних дана искоришћен је исти, раније предложен модел ANN са идентичним параметрима па се архитектура модела у овом поглављу неће посебно истицати.

Тестирање утицаја празника на све потрошаче и месеце

У циљу испитивања утицаја новоформираног обележја, најпре су, на следећој слици приказани резултати добијени узимајући у обзир све потрошаче и све месеце у години (Слика 4-51).

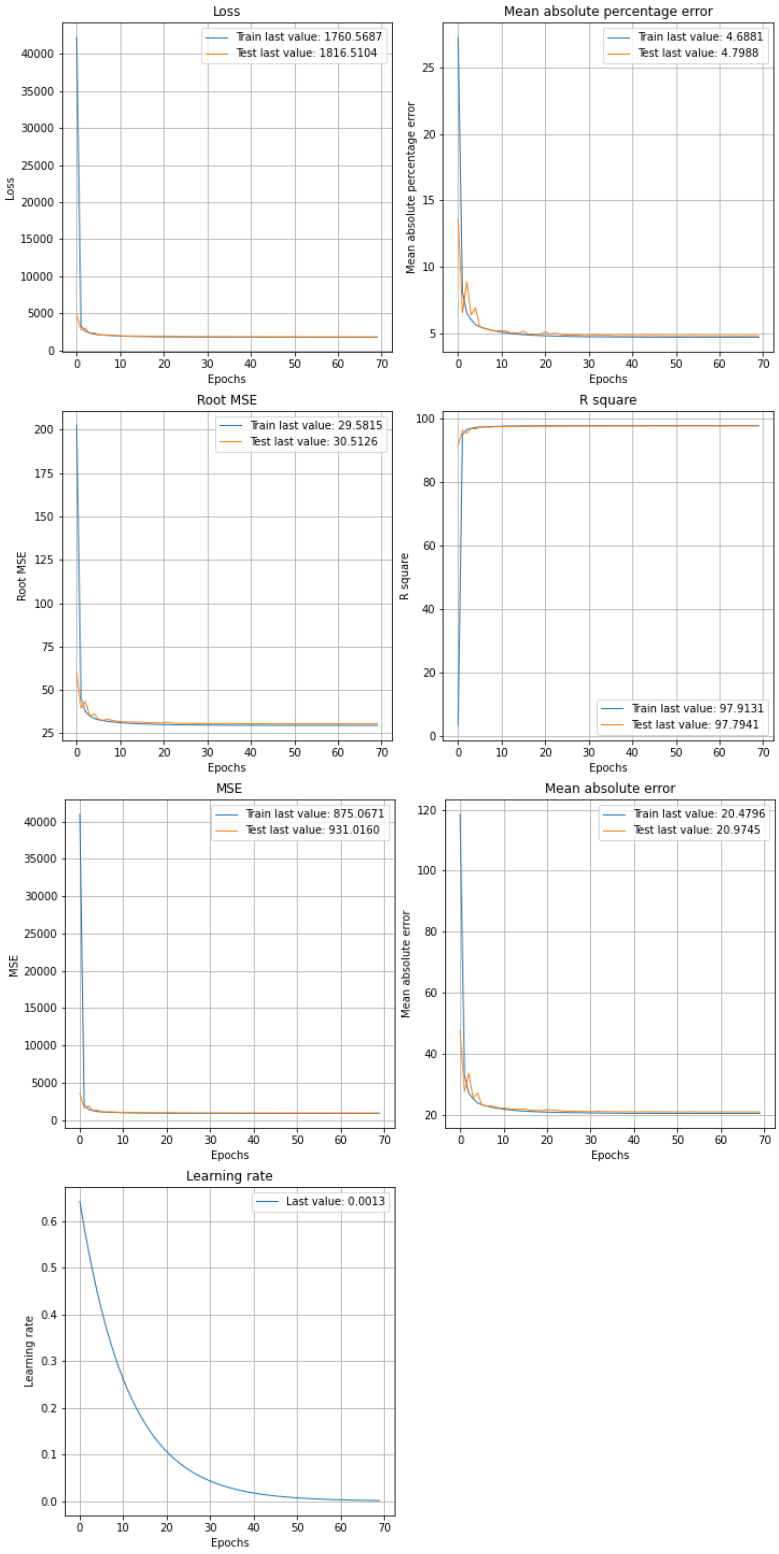
Поредећи добијене резултате са раније датим резултатима у оквиру поглавља 4.3 и 4.4, приметно је да се тежња модела ка минимуму и његова конзистентност одржала. Просечна апсолутна процентуална грешка сада износи 4,57% и коефицијент детерминације (R^2) је нешто већи од 98%, што представља одређен напредак у односу на резултате раније предложеног модела (4,63% за средњу процентуалну грешку и 97,87% за R^2). Овај благи напредак би се могло приписати утицају новоформираног обележја што ће бити тестирано на три групе месеци, раније формиране према броју нерадних дана.



Слика 4-51: Дијаграми праћених метрика предложеног ANN модела над свим подацима, са додатним обележјем („Нерадни дани“)

Тестирање утицаја празника током месеци са највише нерадних дана (прва група - више од 10 нерадних дана)

Резултати добијени предикцијом потрошњи електричне енергије посматрајући месеце са највећим бројем нерадних дана дати су на следећој слици (Слика 4-52).



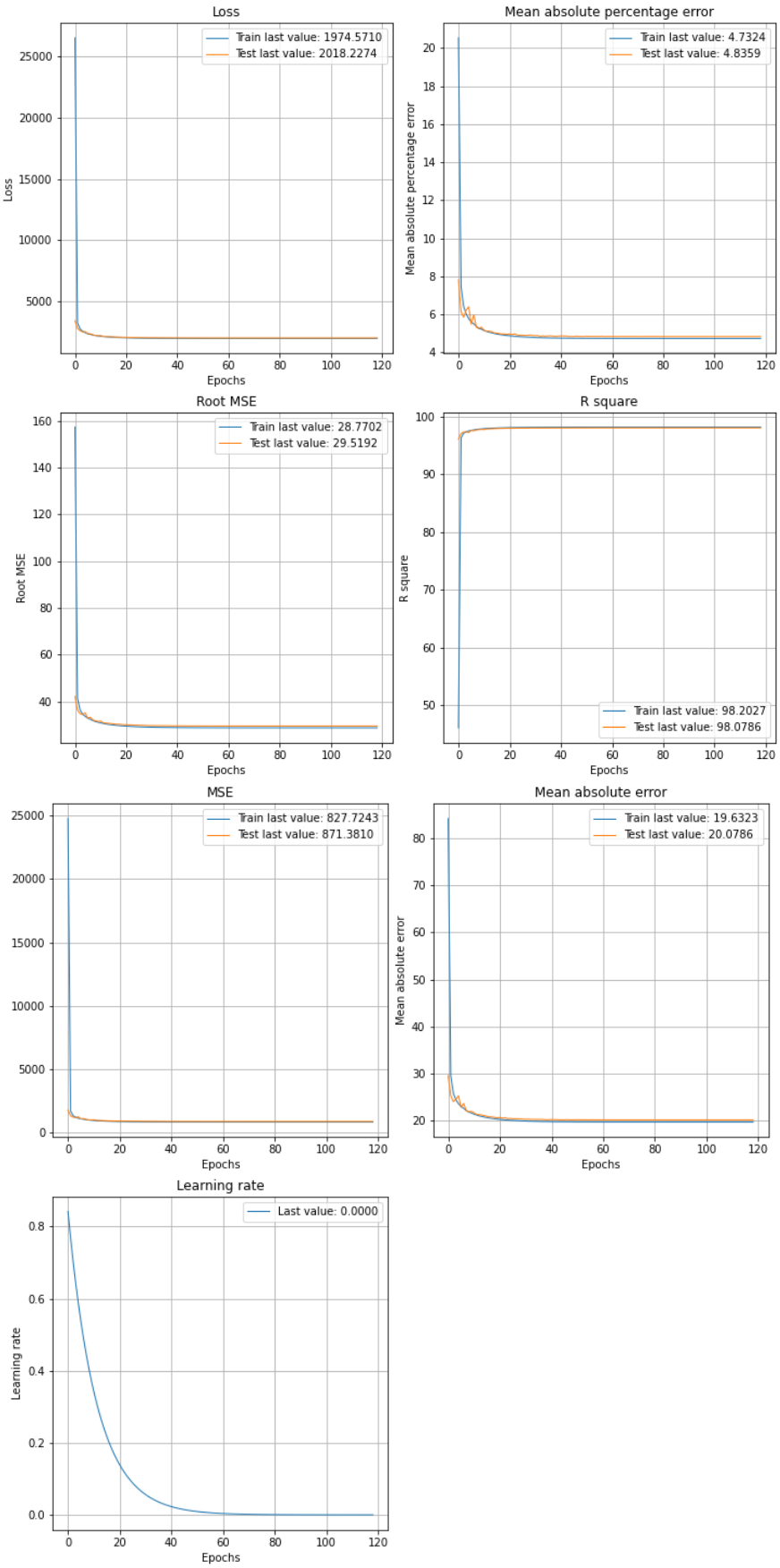
Слика 4-52: Дијаграми праћених метрика за прву групу месеци

Као и раније, модел показује изузетну конзистентност у тежњи ка минималној грешци која за прву групу месеци са десет нерадних дана износи 4,79% уз прецизност модела одређену коефицијентом детерминације од 97,9%.

Тестирање утицаја празника током месеци са 9 или 10 нерадних дана (друга група)

Резултати добијени предикцијом потрошњи електричне енергије посматрајући другу групу месеци (месеце са девет и десет нерадних дана) дати су на следећој слици (Слика 4-53).

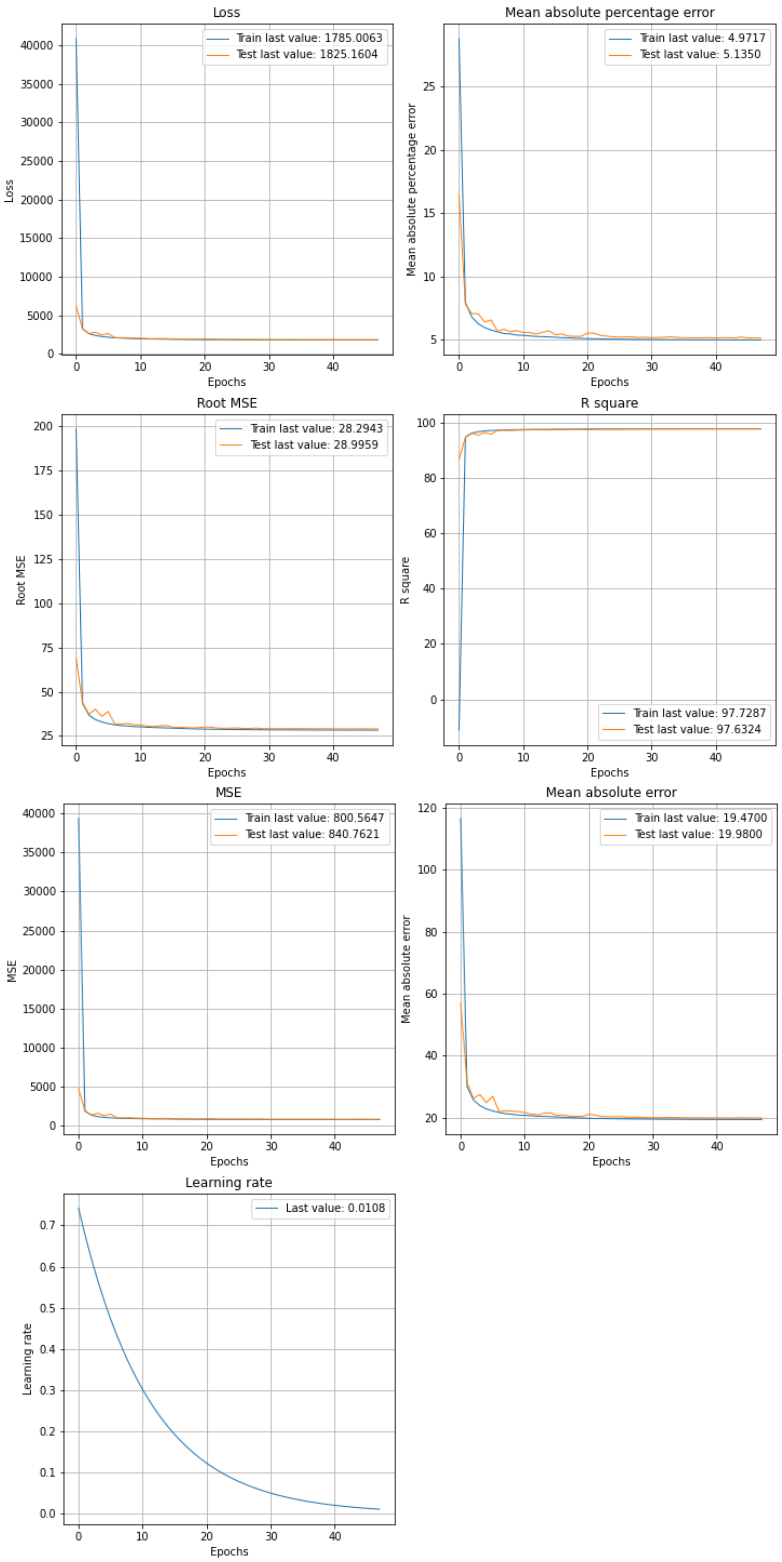
Поново модел показује изузетну конзистентност у тежњи ка минималној грешци која за прву групу месеци са највише нерадних дана износи 4,83% уз прецизност модела одређену коефицијентом детерминације од 98%.



Слика 4-53: Дијаграми праћених метрика за другу групу месеци

Тестирање утицаја празника током месеци са најмање нерадних дана (трећа група - мање од 9 нерадних дана)

Резултати добијени предикцијом потрошњи електричне енергије посматрајући месеце са најмањим бројем нерадних дана дати су на следећој слици (Слика 4-54).



Слика 4-54: Дијаграми праћених метрика за трећу групу месеци

Као и раније, предложени модел показује изузетну конзистентност у тежњи ка минималној грешци која за прву групу месеци са највише нерадних дана износи 5,15% уз прецизност модела одређену коефицијентом детерминације од 97,72%.

Табела 13: Резултати испитивања утицаја броја нерадних дана изражени преко метрика евалуације ANN модела

Скуп података	RMSE (%)	MAE (%)	MAPE (%)	AR^2 (%)
Тестирање за прву групу месеци - са највише нерадних дана (више од 10)	6,88	4,73	**4,79**	97,91
Тестирање за другу групу месеци - са 9 и 10 нерадних дана	6,90	4,69	**4,83**	98,07
Тестирање за трећу групу месеци - са најмање нерадних дана (мање од 9)	7,21	4,98	**5,15**	97,72
Тестирање за цео (базни) скуп података	6,69	4,49	**4,58**	97,87

Резултати кључних метрика приказани на претходним сликама (Слика 4-52, Слика 4-53, Слика 4-54) и у Табели 13 потврђују да се утицај броја празника (нерадних дана) у месецу може употребити у корист способности модела да предвиђа месечне потрошње електричне енергије и у скупу хетерогених потрошача какви су на подручју које је предмет истраживања. Тиме је потврђена друга посебна хипотеза по којој се техникама машинског учења, а превасходно помоћу ANN могу предвидети потрошње електричне енергије и за дане празника (нерадних дана). Истина, добијени резултати потенцијално могу унапредити додатним прилагођавањем параметара модела или другачијом архитектуром модела, али је циљ овог рада да се развије универзалан модел који може решавати различите проблеме повезане са предикцијом потрошње електричне енергије.

Кôд развоја модела за одређивање утицаја празника дат је у виду прилога (Прилог 11).

4.5. ANN модел за подршку класификацији потрошача електричне енергије

4.5.1. Модел за одређивање зоне потрошње

Неретко се јавља потреба и о класификацији потрошача на једном простору и то по различитим критеријумима. У складу са претходно представљеним моделом за подршку предвиђању потрошње електричне енергије у kWh употребом слојева нормализације тежина и нормализације активација у слоју (WNL и LNL слојева) природно је очекивати да се исти слојеви могу имплементирати и у модел за класификацију потрошача. У том погледу, занимљиво је бавити се предвиђањем зоне потрошње за сваког потрошача, одређену у зависности од остварених потрошњи [56].

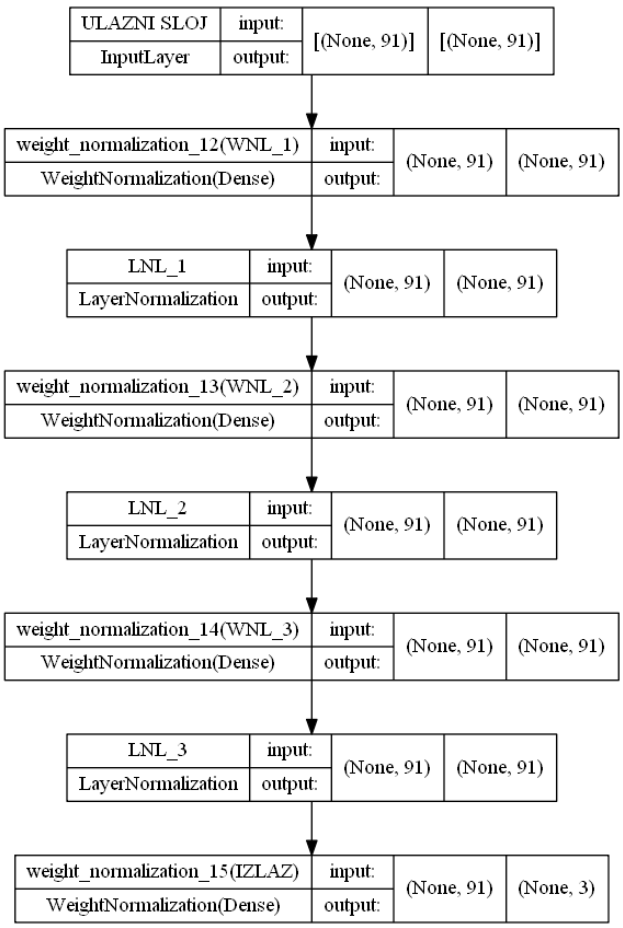
На посматраном подручју постоје три зоне у складу са висином остварене потрошње [129]:

- зелена зона – до 350 kWh,
- плава зона – од 350 до 1500 kWh,
- црвена зона – преко 1500 kWh.

Истраживањем скупа података (како је и раније приказано у поглављу 4.2) утврђено је да веома мали број потрошача има потрошње у највишој зони, а највећи број потрошача месечно потроши између 350 и 1500 kWh (Слика 4-55).

Слика 4-55: Потрошачи према зонама (висини) потрошње

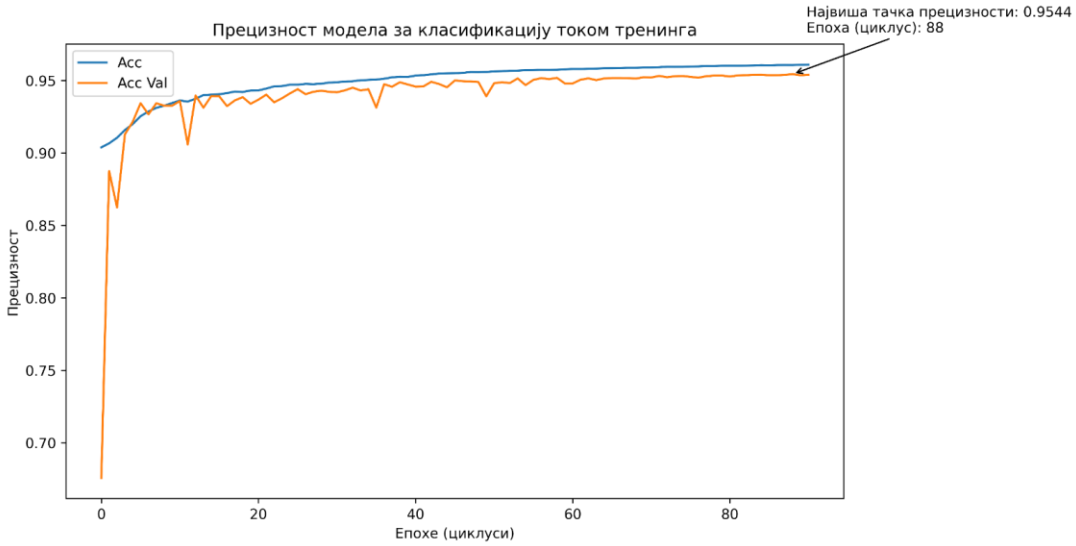
У складу са претходним, развијен је модел за предвиђање зоне потрошње сачињен од идентичних слојева као модел *Б* о коме је било речи у претходном поглављу (Слика 4-56).



Слика 4-56: Програмска штампа архитектуре модела за предвиђање зоне потрошње

За разлику од раније приказаног регресионог проблема, код кога се активациона функција не примењује на излазни слој, већ се линеарна комбинација улаза користи као резултат модела, у случају класификације на излазу из мреже ова функција је неопходна за додељивање вероватноћа припадности свакој од категорија. Тако је, у складу са тим одабрана *softmax* активациона функција за активацију неурона у свим скривеним али и излазном слоју.

Резултати модела потврђују очекивања да се моделом са слојевима са нормализацијом тежина може са великим успехом извршити и предвиђање потрошачке класе у коју ће се потрошач уклопити на основу познатих параметара. Слика 4-57 показује тренд прецизности модела кроз циклусе обучавања (епохе) приказујући прецизност модела на тренинг скупу (плава линија - *Acc*) и на валидационом скупу (црвена линија – *Acc Val*), а Слика 4-58 даје програмску штампу осталих резултата које постиже модел (прецизност, опозив и F_1 меру).



Слика 4-57: Дијаграм прецизности модела на тренинг и валидационом скупу

	precision	recall	f1-score	support
0	0.95	0.96	0.96	105032
1	0.95	0.94	0.95	81131
2	0.00	0.00	0.00	21
accuracy			0.95	186184
macro avg	0.64	0.63	0.64	186184
weighted avg	0.95	0.95	0.95	186184

Слика 4-58: Програмска штампа резултата предикције потрошачке зоне

На укупну прецизност модела највише утиче трећа класа, тј. потрошачи који припадају црвеној зони (класа са ознаком „2“, Слика 4-58). Вредности F_1 мере за сваку од класа показују да модел има потешкоћа са идентификацијом управо ове класе. Мали број потрошача са високим потрошњама који се нашао у скупу података након одстрањивања свих екстремних и аутлајер вредности не дају моделу довољно информација да би научио њихово понашање током тренинга модела, па у фази тестирања сви ови потрошачи бивају погрешно класификовани.

Резултати прецизности дати кроз матрицу конфузије (енгл. *Confusion Matrix*) (Слика 4-59) показују да је највећи број потрошача тачно класификован, у одговарајућу зону.

Слика 4-59: Матрица конфузије за предвиђање зоне потрошње

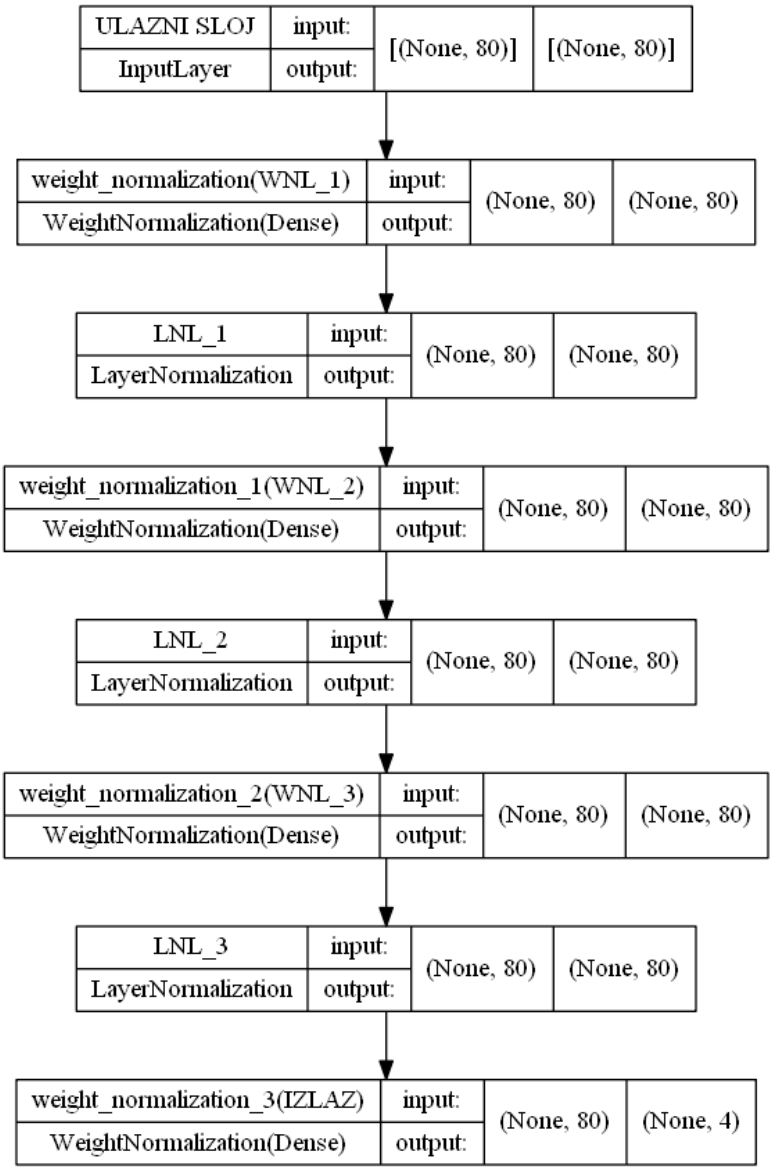
Матрица показује да модел има благе недоумице сврставајући одређен број потрошача (≈2%) у зелену зону док они заправо припадају плавој и обратно у износу од ≈2,5%. Све потрошаче који припадају категорији „Црвена зона“, а нашли су се у тест скупу, модел је погрешно препознао као потрошаче плаве зоне што је засигурно последица малобројности потрошача припадника црвене зоне (Слика 4-55).

На основу вредности приказаних у матрици конфузије и на листингу датом на слици - Слика 4-58, могу се израчунати и друге битне мере, дате у потпоглављу 2.2.1, као што су TPR, TNR и FPR: TPR = 0,946, TNR = 0,96, FPR = 0,038. Високе вредности прве две мере (TPR, TNR) указују на задовољавајућу способност модела да класификује потрошаче у одговарајућу потрошачку зону према њиховим особинама и потрошњама из ранијег периода што потврђује и ниска вредност последње мере (FPR). Вредност TPR од 0,946 указује да је модел способан да идентификује стварно позитивне примерке у класи, TNR са такође високом вредношћу показује да је у стању да идентификује и стварне негативне инстанце у класи, док FPR од 0,038 указује на то да модел ретко прави грешку идентификације негативних инстанци као позитивних.

Кôд развоја модела за одређивање зоне потрошње дат је у виду прилога (Прилог 12).

4.5.2. Модел за одређивање сезонских промена

И раније је показано да сезонске промене (смена годишњих доба) имају осетног утицаја на висине потрошњи (поглавље 4.2, Слика 4-16, Слика 4-17, али и Слика 4-35). Имајући у виду осцилације потрошњи проузроковане овим променама јавља се претпоставка да се раније приказани модел може употребити и за предвиђање сезонских промена. Слика 4-60 показује програмску штампу архитектуре модела за предвиђање сезонских промена.



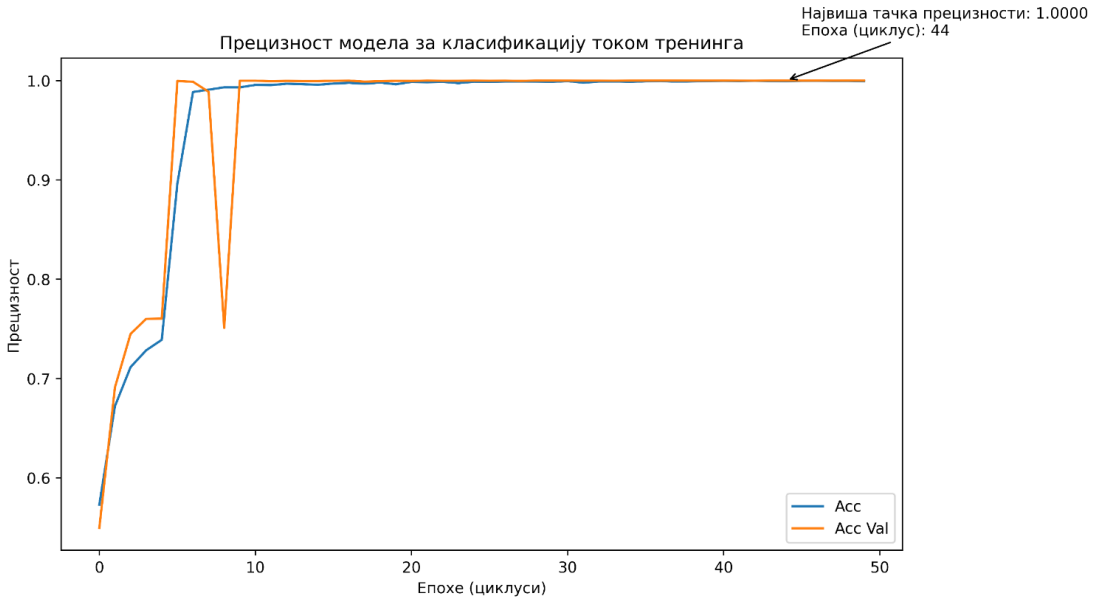
Слика 4-60: Програмска штампа (шема) архитектуре модела за предвиђање сезонских промена

Резултати које даје модел (Слика 4-61) и у овом случају поново потврђују очекивања да се предложеним моделом може прецизно предвидети промена сезоне (годишњег доба).

	precision	recall	f1-score	support
0	1.00	1.00	1.00	49372
1	1.00	1.00	1.00	47133
2	1.00	1.00	1.00	45048
3	1.00	1.00	1.00	44631
accuracy			1.00	186184
macro avg	1.00	1.00	1.00	186184
weighted avg	1.00	1.00	1.00	186184

Слика 4-61: Програмска штампа резултата предикције сезонских промена (годишњих доба)

Слика 4-62 показује тренд прецизности модела кроз циклусе обучавања (епохе) на тренинг скупу (плава линија) и на валидационом скупу (наранџаста линија). Јасно је да је модел брзо постигао високу прецизност како на тренинг тако и на валидационом скупу података. Тачан распоред предикција по класама дат је у матрици конфузије (Слика 4-63).



Слика 4-62: Дијаграм прецизности модела на тренинг и валидационом скупу

Према подацима из матрице конфузије (Слика 4-63) у читавом тест скупу модел је нетачно класификовао само две инстанце што још једном потврђује његову велику моћ да предвиди сезонске промене.

Слика 4-63: Матрица конфузије за предвиђање сезонских промена (годишњих доба)

Како би се додатно утврдила прецизност модела, поново, на основу вредности из матрице конфузије могу се израчунати и друге битне мере. Вредности TPR и TNR у износу од 1 (прецизније 0,99999) и FPR=0 говоре о невероватно великој способности модела да препозна сезонске промене на основу карактеристика потрошача доступних у скупу података.

Кôд развоја модела за одређивање сезонских промена дат је у виду прилога (Прилог 13).

Приказана прецизност модела при одређивању зоне потрошње и одређивања сезоне (годишњег доба) у синергији са раније приказаним резултатима о предвиђању висине потрошње електричне енергије у појединим деловима године (месецима или годишњим добима) или под утицајем празника током месеца, још једном потврђује на почетку постављену хипотезу по којој се техникама ML, а превасходно помоћу ANN могу предвидети периоди повећаних потрошњи електричне енергије какви најчешће настају услед промена годишњих доба. Висок, готово 100% степен прецизности у предвиђању сезонских промена и готово исто толико у одређивању потрошачке зоне може помоћи дистрибутивним компанијама у унапређењу планирања производње и дистрибуције електричне енергије у складу са овим променама. Такође, имајући у виду успешност предложених модела за класификацију раније поменута апликација би могла бити проширена и класификационим модулом.

4.6. Апликација за предвиђање потрошњи, намењена крајњем кориснику

Прегледом доступне научне литературе и референтних радова, није пронађен развијен систем који би могао да одговори на све предочене специфичности проблема предвиђања електричне енергије. Имајући то у виду заједно са важношћу овог истраживања и резултата приказаних у досадашњем излагању, намеће се идеја о развоју система (апликације, десктоп и/или мобилне) за предвиђање месечне потрошње електричне

енергије. Предложени систем би био заснован на обученом моделу који се кроз резултате овог истраживања показао као најуспешнији (модел типа *Б, Слика 4-37*). У самој апликацији би уз примену неког од предложених типова кластеровања крајњем кориснику било омогућено да на врло једноставан начин добије информацију о процењеној количини потрошене енергије за одређени месец у години.

Једна таква апликација - *Систем за предвиђање потрошње електричне енергије* (енгл. *Electricity Consumption Prediction System, ECPS*), представљена и у [54], би пружила крајњем кориснику значајне информације о обиму очекиване потрошње узимајући у обзир неколико познатих карактеристика потрошача и преузимањем дела података из јавно доступних извора на интернету (о температури, влажности ваздуха, потрошњи из претходног месеца итд).

Слика 4-64 даје груби приказ архитектуре апликације за предвиђање потрошње електричне енергије која се у основи састоји из три главна модула:

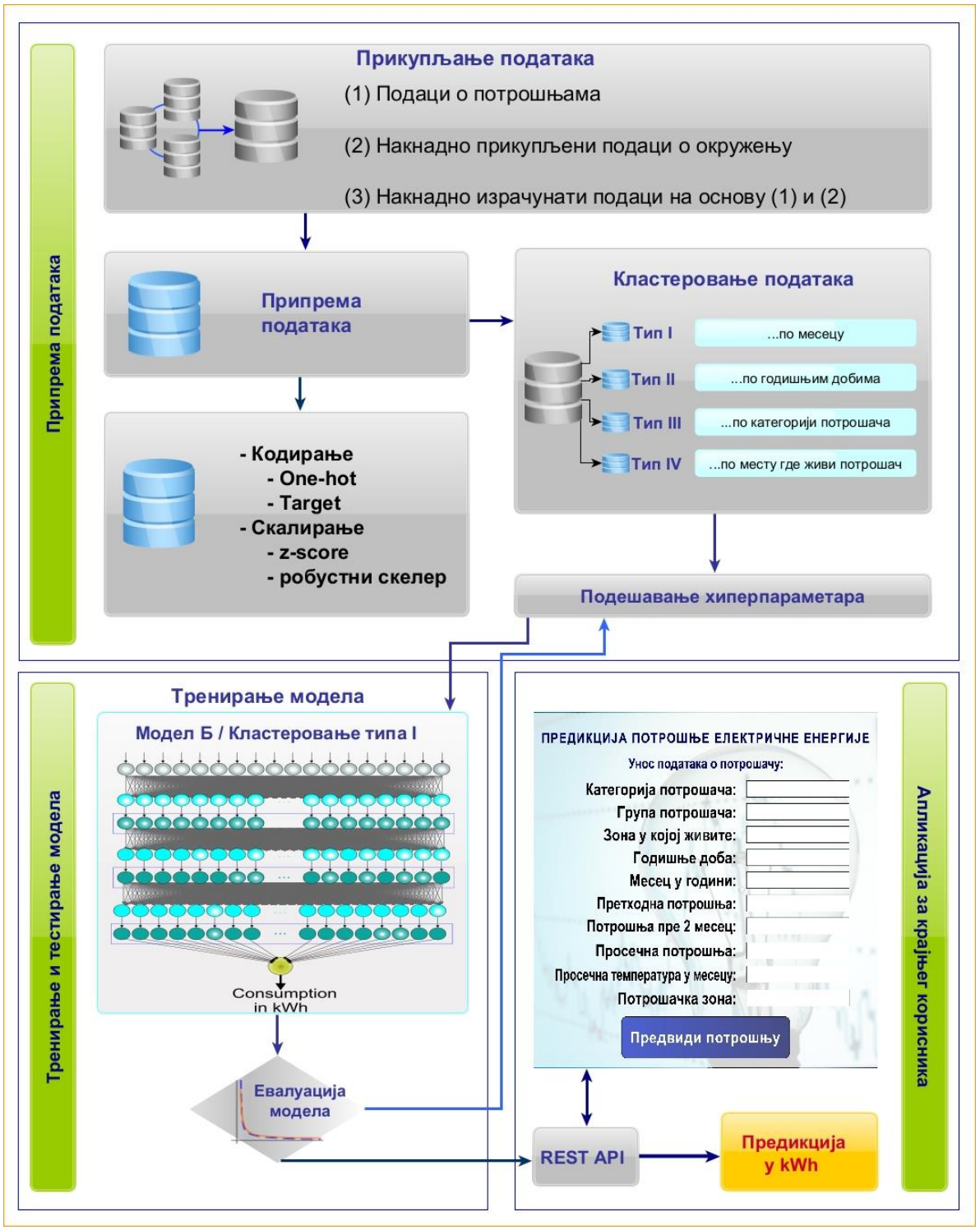
1. препроцесинг података,
2. одабир модела и типа кластеризације,
3. интерактивни интерфејс за корисника.

Први модул се односи на фазу прикупљања података, односно, фазу када се уносе нови подаци који ће учествовали у тренирању мреже, како би већ обучен модел био у прилици да научи и из нових података. У овој фази се врши и одабир типа кластеровања. Кориснику се може оставити могућност да ипак сам одлучи који од типова кластеровања ће одабрати уз сугестију на онај који се до сада истакао као најбољи. Поред тога, може се оставити и могућност за сталну надградњу новим типовима кластеровања. Затим се подаци кодирају и скалирају и на крају следи постављање вредности хиперпараметара, одабиром предложених или постављањем потпуно нових вредности.

Други модул се односи на модел ANN и одабир истог, што би подразумевало модел типа *Б* уколико не постоји други, бољи модел. Затим се врши евалуација и штампа прогнозираних потрошњи.

Трећи модул представља комуникацију са корисником кроз форму са пољима где се уносе карактеристике потрошача на основу којих ће бити извршено предвиђање.

Развој овакве апликације потврђује и у директној је вези са последњом, шестом посебном хипотезом по којој се применом резултата предикционог модела вишеслојне ANN може извршити планирање обезбеђења потребних количина електричне енергије чиме се омогућује и успостављање стабилног, несметаног снабдевања електричном енергијом на одређеном подручју.



Слика 4-64: Груби приказ архитектуре апликације за предвиђање потрошње електричне енергије [54]

5. ДИСКУСИЈА УСПЕШНОСТИ ПРЕДЛОЖЕНОГ МОДЕЛА

Како је већ раније поменуто, скуп података коришћен за тренинг и тестирање предложеног ANN модела садржи податке о потрошњама електричне енергије потрошача на територији града Ужица. Скуп садржи преко милион записа о потрошњама прикупљеним у периоду од четири посматране године, за преко 40.000 мерних места.

Предложеним поступком, расположиви, иницијални скуп података је ажуриран и допуњен одговарајућим карактеристикама потрошача и амбијента. Укључујући и метеоролошке одлике али и формирањем нових обележја на основу постојећих калкулативним методама, формиран је финални скуп који моделу пружа довољно информација за врло прецизно предвиђање. За формирани скуп обележја утврђена је њихова међузависност и важност за постизање циља - успешно предвиђање месечних потрошњи електричне енергије како за појединачног потрошача тако и за групе.

За разлику од истраживања у којима се користе колекције две или више техника ML (као на пример у [47] и [44]), и насупрот изричитим тврдњама аутора у [45] нужности њихове употребе, модел приказан у овом истраживању базиран је искључиво на ANN, а претходно поглавље показује његову супериорност у решавању конкретног проблема у односу на неколико најпопуларнијих техника ML.

Развијен је оригинални модел заснован на ANN који је у стању да предвиђа месечне потрошње у скупу хетерогених потрошача какви су на територији једног града, са свим специфичностима које једно слично подручје може да има.

Модел је обучаван, а затим и тестиран над хронолошки поређаним подацима, као што је то урађено у [147]. У овом случају записи о потрошњама прве три године се користе за тренинг модела док се подаци из последње календарске године користе за његово тестирање.

Уопштено говорећи, предложени модел (модел *Б*) истиче се знатно испред класичног модела са густим слојевима, истих вредности хиперпараметара. Успешност модела се огледа како у његовој коначној прецизности тако и по конзистентности и стабилности. Ово је приметно не само у погледу обраде хетерогеног (читавог) скупа података већ посебно у појединачним кластерима.

Предложеном методологијом поделе свих потрошача на хомогеније кластере постижу се прецизнија предвиђања од оних добијених у читавом скупу података, насупрот закључцима до којих су 2018. дошли аутори Зекић и Сушац у [60] али у складу са закључцима приказаним у [50] из исте године. Показало се да предложени поступак издвајања хомогенијих група потрошача или посматрање мањег временског оквира за тренинг и тестирање модела, доприносе убрзању процеса и чак подизању степена прецизности модела.

У случају предложеног кластеровања *типа III* постигнути су бољи резултати за потрошаче из категорије недомаћинства у односу на оне приказане у [15]. Аутори у [15] посматрају једну хомогенију групу потрошача, искључиво индустријске потрошаче, са значајно сличним понашањима у погледу биланса потрошњи и потрошачких навика у односу на оне у циљном скупу ове дисертације.

Резултати приказани у овој дисертацији показују да се и на простору са разноврсним потрошачима какав је на територији града Ужица могуће и то са великом тачношћу предвидети потрошње на месечном нивоу, што је насупрот тврдњама аутора у [13] да се

потрошња електричне енергије може предвидети само ако се посматра један тип потрошача.

Не само да је то могуће, већ је апсолутна грешка коју даје предложени модел у износу од ≈5% за све потрошаче у читавом скупу и ≈1% за појединачног насумичног потрошача чак неколико пута нижа од најниже коју су аутори постигли у истраживању приказаном у [53].

Ако се као мера успешности модела узме коефицијент детерминације, предложени модел у овој дисертацији (модел *Б*) је успешнији од модела предложеног од стране аутора у [32]. Аутори у [32] тестирају предложени модел на скупу сличних потрошача, смештеним на ограниченом простору у оквиру универзитетског насеља и постижу тачност на тестном скупу између 95% и 99% (изражену кроз коефицијент R^2). Тачност предложеног модела ове дисертације изражена кроз исти коефицијент износи 97,8% такође на тестном скупу. Треба нагласити да је резултат од 97,8% постигнут на веома хетерогеном скупу потрошача, односно, узимајући у обзир све потрошаче на територији града. У хомогенијим скуповима, применом модела на појединачне кластере потрошача, ови резултати су још бољи. Чак и поредећи коефицијент AR^2 који због своје природе (узимања у обзир броја предиктора) најчешће има ниже вредности од R^2, предложени модел у појединачним кластерима има веома високе вредности које се крећу од 97,8% до чак 99,5%.

Тестирајући хипотезу о утицају броја нерадних дана (празника) у току месеца на висину потрошње електричне енергије и увођењем броја нерадних дана као новог параметра, добијају се нешто прецизнији резултати. Посебно је значајно што за ове потребе није развијан нови модел већ је коришћена комплетна архитектура и хиперпараметри раније развијеног модела који се показао као најуспешнији.

Целокупни резултати добијени предложеним моделом потврђују да се ANN могу са великим успехом употребити за предвиђање потрошњи електричне енергије, што је у складу са закључцима које су још 2007. године у свом истраживању дали Tso и Jy (*Tso u Yau*) у [14] поредећи ANN са стаблима одлучивања и регресионим моделима. До сличних закључака касније долазе и аутори у [47].

Када се посматрају резултати ANN и њихова успешност у класификацији потрошача електричне енергије у одговарајућу зону потрошње, поново је неоспорна њихова успешност. Прецизност у предвиђању зоне потрошње преко 95% просечно за све три зоне наговештава да се са малим прилагођавањима модел може још усавршити и довести до знатно бољих резултата, али и да на резултате највише утиче малобројност класе потрошача који припадају зони са највишим потрошњама.

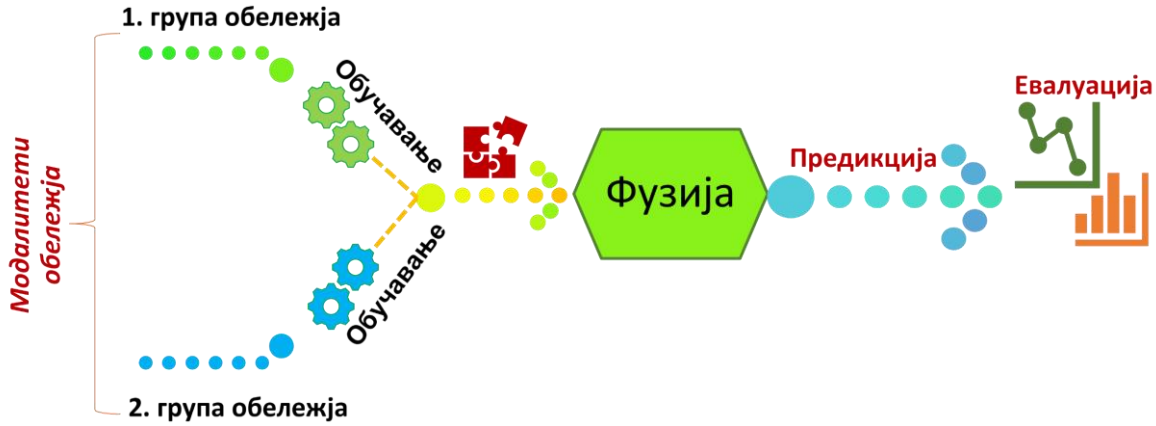
Још више успеха остварује модел у одређивању сезонских промена (промена годишњих доба) где са свега две погрешко класификоване инстанце постиже готово 100% успешности.

Развијено решење и флексибилност модела представљеног у овој дисертацији пружа могућност за његову примену како у погледу одређивања класе тако и вредности зависног обележја повезаног са потрошњом електричне енергије, али и на друге проблеме сличне природе.

6. МУЛТИМОДАЛНИ, ЕКСПЕРИМЕНТАЛНИ МОДЕЛ ANN КАО ПОЛАЗНА ТАЧКА БУДУЋИХ ИСТРАЖИВАЊА

Иако прилично успешан, предложени модел за предвиђање потрошњи електричне енергије се може надограђивати и усавршавати у разним правцима. Један од праваца адаптације приказаног модела може се остварити увођењем модалитета на самом улазу у мрежу. Као највећа предност и кључна карактеристика мултимодалних модела ANN истиче се (бар на теоријској основи), њихова способност интеграције и анализе више врста података истовремено како би се боље разумео контекст и обавила сложенија анализа. Таква, мултимодална мрежа, би могуће била у стању да брже и лакше учи законитости у потрошњама електричне енергије, интегришући знање стечено у обуци појединачних грана модела.

Мултимодалне мреже се најчешће користе за обраду различитих врста сигнала, слика и текста, аудио и видео садржаја, итд [148], [149]. У том смислу би се могло размишљати о различитим принципима за класификацију улазних обележја у овом истраживању. На пример, на улазу би се могли раздвојити улази различитог типа – модалитета (нумерички од категоријских обележја, метеоролошки од осталих података, обележја из иницијалног скупа од додатних, итд.). У овом поступку, гране улаза у мрежу ће, свака одвојено, процесуирати засебно кодиране податке, све до тренутка фузије ових грана у једној тачки пре добијања излаза из мреже (Слика 6-1).



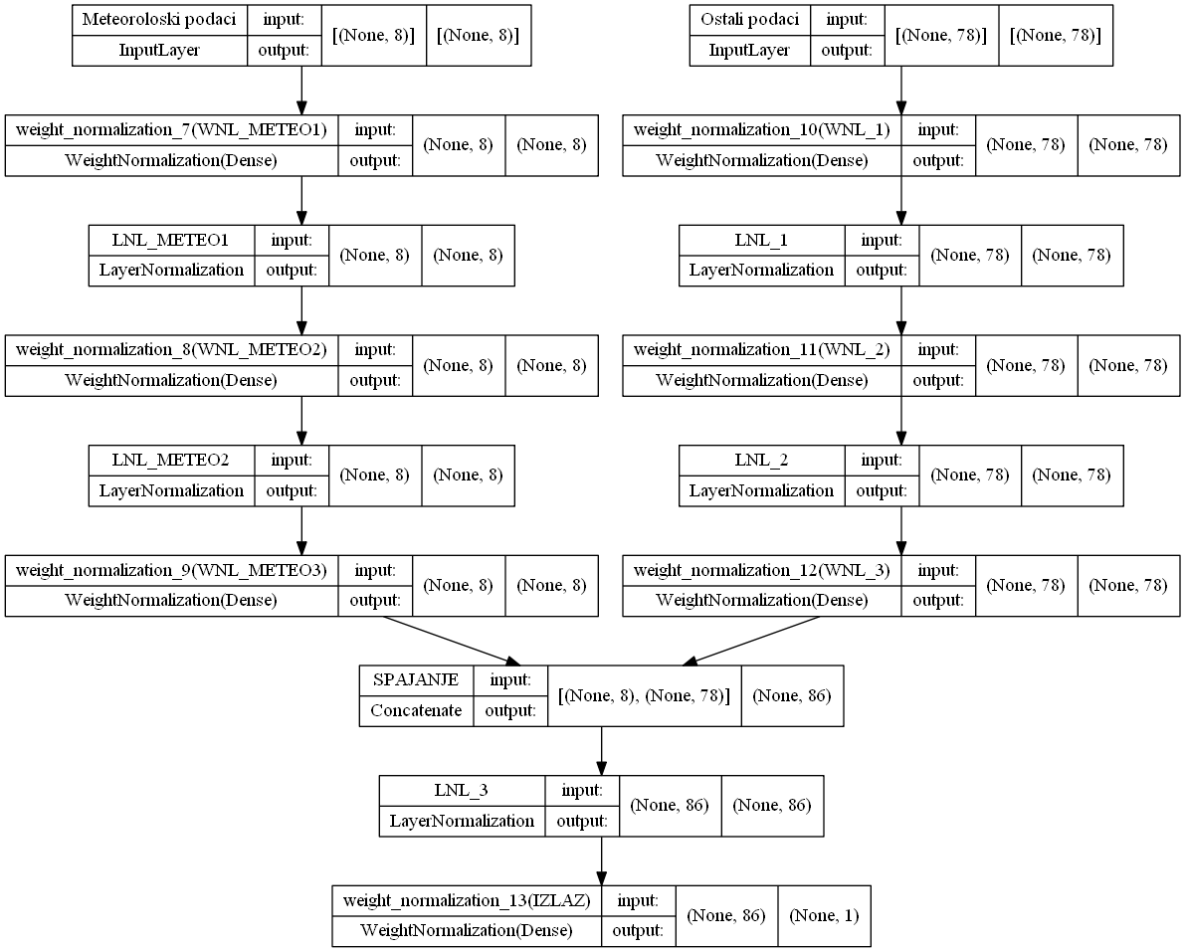
Слика 6-1: Илустрација модела ANN заснованог на принципу модуларности

Одлука о постављању тачке фузије у мултимодалним ANN углавном зависи од типа проблема и поставља се хеуристички, иако аутори у [150] долазе до закључка о бенефитима постављања ове тачке у раној фази модела (енгл. *An early fusion strategy*).

У наставку ће бити тестирана способност мултимодалног модела ANN кроз два случаја заснована на два принципа модуларности. Број скривених слојева и хиперпараметри модела су, у циљу поређења са раније формираним најуспешнијим моделом, исти. Након улазних слојева у обе гране модела следе по два слоја са нормализацијом тежина и нормализацијом активација слоја након чега следи тачка фузије, до тада одвојено тренираних параметара. Након фузије параметара обе гране, врши се нормализација активације слоја пре израчунавања излаза из мреже.

6.1. Мултимодалност заснована на метеоролошким и неметеоролошким карактеристикама потрошача

Као полазна основа у погледу развоја једног мултимодалног модела ANN у циљу предикције месечних потрошњи електричне енергије, послужиће подела обележја на метеоролошке и остале податке. Тачка фузије која се показала најбоље у решавању овог проблема налази се ближе излазном слоју. Архитектура овог модела дата је на следећој слици (Слика 6-2).



Слика 6-2: Програмска штампа (шема) архитектуре модела са две улазне гране метеоролошких и других обележја

Шема архитектуре модела (Слика 6-2) илуструје ANN са два одвојена инпута, чији процес обуке тече одвојено и паралелно све до тренутка спајања, након трећег слоја са нормализацијом тежина („WNL_METEO3" у левој грани и „WNL_3" у десној грани) након чега следи последњи LNL слој пре излазног слоја. Лева улазна грана модела садржи метеоролошке податке: просечну, минималну и максималну месечну температуру, број ведрих и облачних дана у току месеца, укупан број сунчаних сати у току месеца, просечан број сунчаних сати по дану и просечан број сати дневне светлости. Десна грана садржи сва остала обележја. Приметно је да грана са метеоролошким подацима има свега 8 неурона који су у овом случају нумерички, док се у десној грани налази преосталих 78 неурона носећи углавном категоријска обележја.

Иако на први поглед предложени модел делује доста рогобатно и у поређењу са раније предложеним моделом (модел *Б*) неупоредиво опширнији, број параметара подложних тренирању је идентичан у оба случаја што показује и следећа слика (Слика 6-3) у односу на програмску штампу модела *Б* (Слика 4-42).

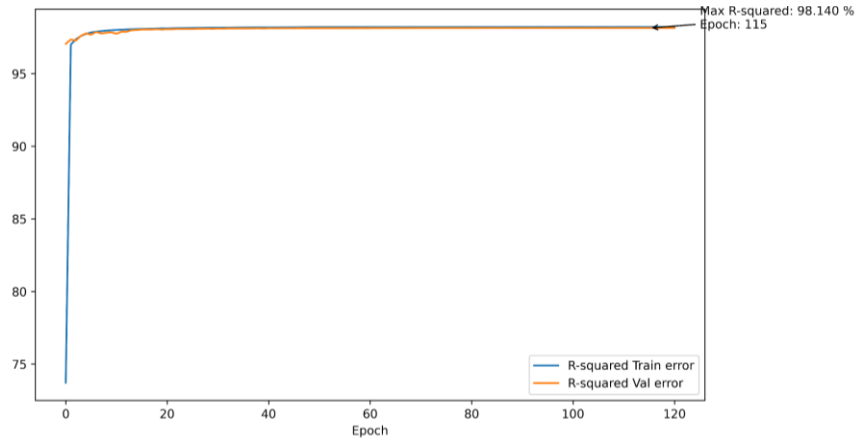
Layer (type)	Output Shape	Param #	Connected to
Meteoroloski podaci (InputLaye r)	[(None, 8)]	0	[]
Ostali podaci (InputLayer)	[(None, 78)]	0	[]
weight_normalization_7 (Weight Normalization)	(None, 8)	153	['Meteoroloski podaci[0][0]']
weight_normalization_10 (Weigh tNormalization)	(None, 78)	12403	['Ostali podaci[0][0]']
LNL_METEO1 (LayerNormalization)	(None, 8)	8	['weight_normalization_7[0][0]']
LNL_1 (LayerNormalization)	(None, 78)	156	['weight_normalization_10[0][0]']
weight_normalization_8 (Weight Normalization)	(None, 8)	153	['LNL_METEO1[0][0]']
weight_normalization_11 (Weigh tNormalization)	(None, 78)	12403	['LNL_1[0][0]']
LNL_METEO2 (LayerNormalization)	(None, 8)	8	['weight_normalization_8[0][0]']
LNL_2 (LayerNormalization)	(None, 78)	156	['weight_normalization_11[0][0]']
weight_normalization_9 (Weight Normalization)	(None, 8)	153	['LNL_METEO2[0][0]']
weight_normalization_12 (Weigh tNormalization)	(None, 78)	12403	['LNL_2[0][0]']
SPAJANJE (Concatenate)	(None, 86)	0	['weight_normalization_9[0][0]', 'weight_normalization_12[0][0]']
LNL_3 (LayerNormalization)	(None, 86)	172	['SPAJANJE[0][0]']
weight_normalization_13 (Weigh tNormalization)	(None, 1)	176	['LNL_3[0][0]']

Total params: 38,344
Trainable params: 19,548
Non-trainable params: 18,796

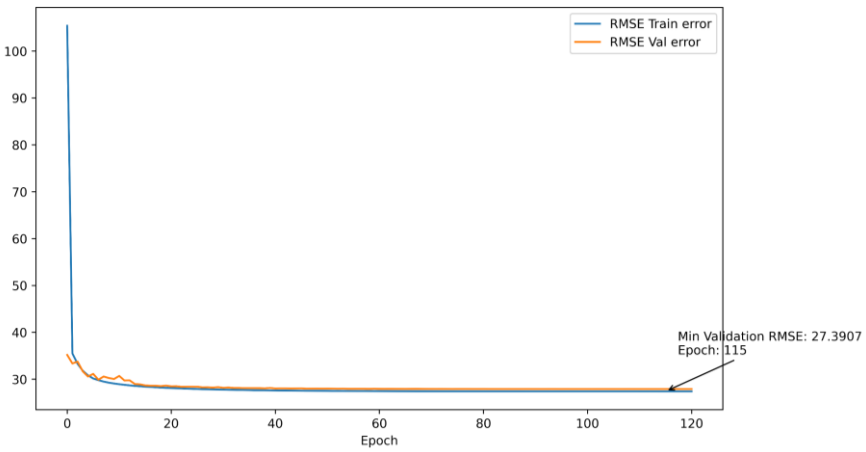
Слика 6-3: Програмска штампа параметара модела за две одвојене улазне гране

Пратећи задате метрике за оцену квалитета закључује се да и овај модел добро препознаје шаблоне у подацима. Чак у преко 98% случајева модел тачно предвиђа вредности таргет варијабле што показује коефицијент детерминације на следећој слици (Слика 6-4). Слика 6-5 и Слика 6-6 приказују кретање корена средње квадратне грешке (RMSE у kWh) и апсолутне вредности средње процентуалне грешке (MAPE) одакле се поново јасно може препознати завидна сигурност и конзистентност модела који стабилно тежи ка

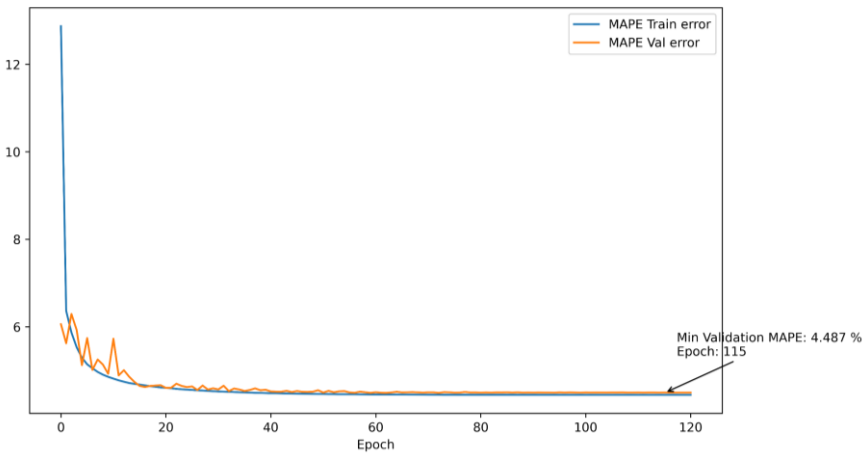
минимуму, постижући грешку од 4,48%, односно ≈27 kWh. Ово не представља значајан напредак у односу на раније представљени модел *Б*, али може бити добар основ за даље правце истраживања, посебно ако би се у обзир узели још неки модалитети података као улаза у ANN.



Слика 6-4: Коефицијент детерминације (R^2) за модел са две улазне гране



Слика 6-5: Корен средње квадратне грешке (RMSE у kWh) за модел са две улазне гране



Слика 6-6: Средња просечна апсолутна грешка (MAPE) за модел са две улазне гране

Успешност и овог модела најбоље се види из упоредног приказа стварних и предикованих вредности. Слика 6-7 даје такав приказ: стварне вредности - црвена линија дијаграма и вредности по моделу - плава линија дијаграма. Због превеликог броја тачака у тест скупу дијаграм предикције потрошњи свих потрошача не би био јасан, па је на слици (Слика 6-7) приказано поређење вредности за 100 насумично одабраних потрошача.

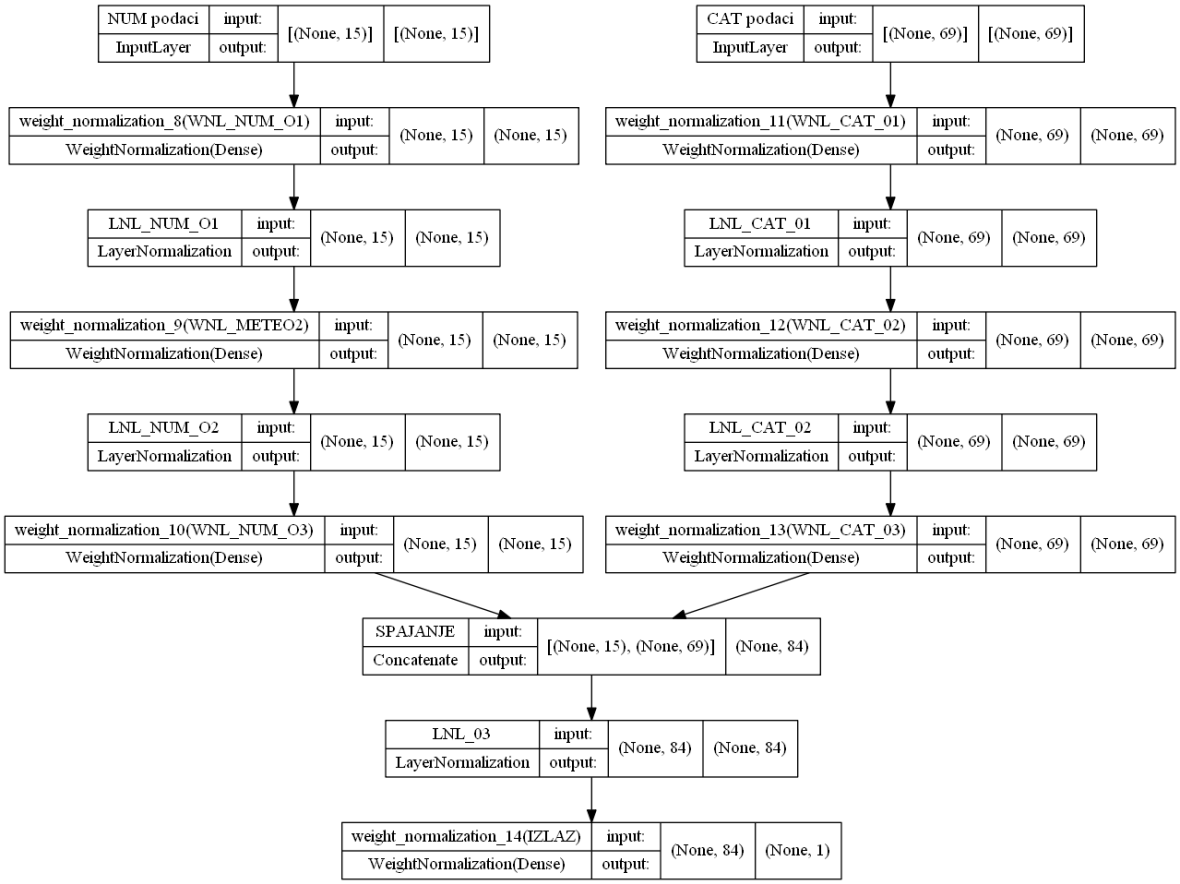
Слика 6-7: Дијаграм стварних и вредности по моделу (модел са паралелним слојевима, две гране и два улаза) – за 100 потрошача из скупа насумично одабраних

Готово потпуно поклапање црвене и плаве линије на дијаграму јасно показује да и модел са две улазне гране, два инпута постиже завидну прецизност, потврђујући способност и овог модела да предвиди потрошње најразличитијих потрошача у таргет скупу. Ово свакако отвара нова поглавља у даљем истраживању која се могу фокусирати на формирање више улазних грана користећи различите модалитете поделе обележја. То је посебно значајно када се узму у обзир различити начини прикупљања података као што је то приказано на самом почетку ове дисертације.

Кôд мултимодалног модела заснованог на метеоролошким и неметеоролошким карактеристикама дат је у виду прилога (Прилог 14).

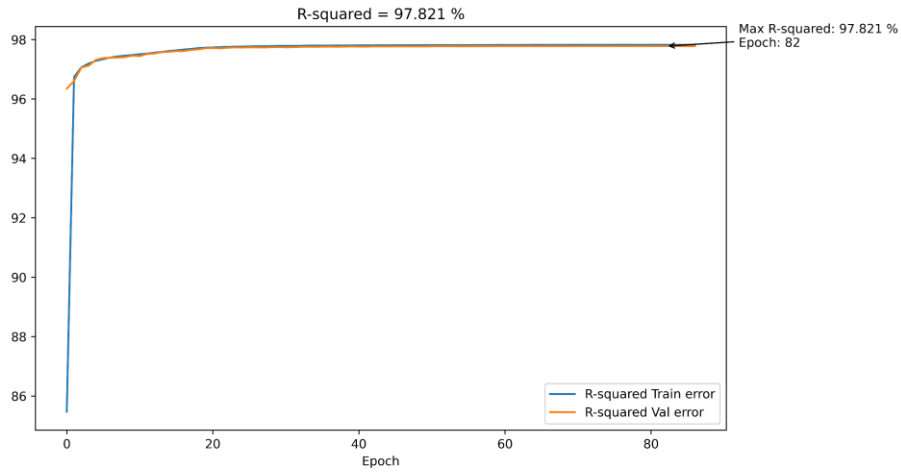
6.2. Мултимодалност заснована на нумеричким и категоријским обележјима потрошача

У наставку је приказан још један значајан тест успешности мултимодалних ANN у предвиђању потрошњи електричне енергије, узимајући као критеријум тип улазних параметара. На следећој слици (Слика 6-8) приказана је архитектура модела који као критеријум за мултимодалност има разлику између нумеричких и категоријских обележја.

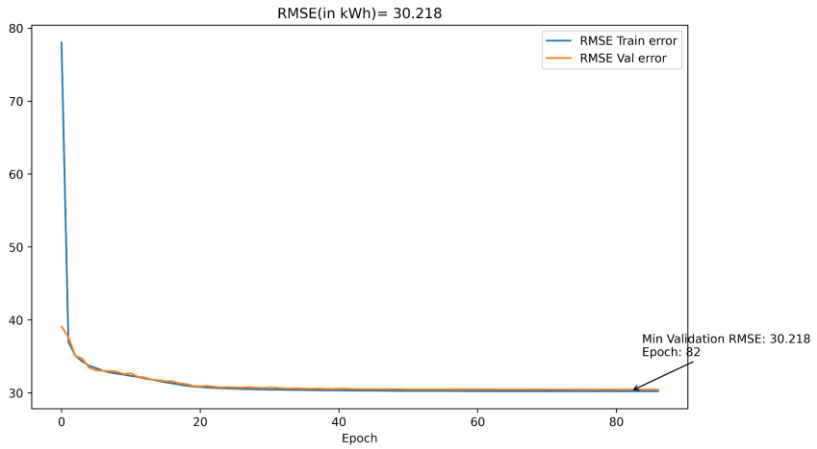


Слика 6-8: Програмска штампа (шема) архитектуре модела са две улазне гране нумеричких и категоријских обележја

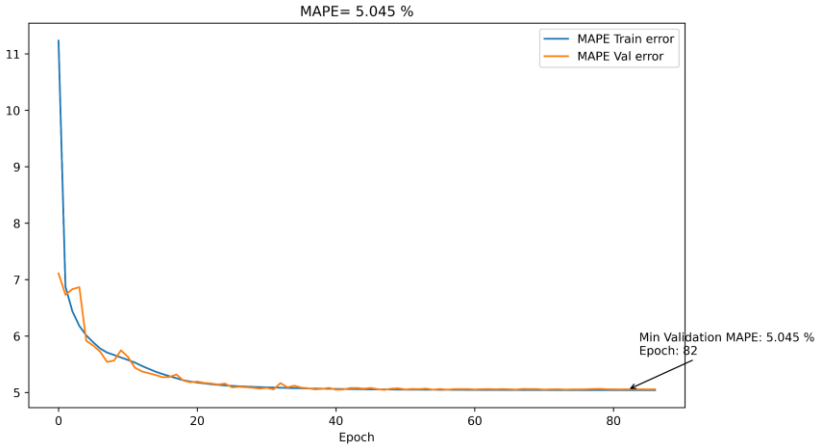
За разлику од претходне модуларности, ова прави нешто другачију расподелу обележја, сврставајући сва нумеричка у једну групу, која тако чине 15 улаза у једну грану мреже и категоријска која након кодирања чине 69 улаза у другу грану. Резултати које даје овај мултимодални модел приказана су на следећим сликама (Слика 6-9 - Слика 6-11).



Слика 6-9:Коефицијент детерминације (R^2) за модел са две улазне гране (нумеричких и категоријских обележја)



Слика 6-10: Корен средње квадратне грешке (RMSE у kWh) за модел са две улазне гране (нумеричких и категоријских обележја)



Слика 6-11: Средња просечна апсолутна грешка (MAPE) за модел са две улазне гране (нумеричких и категоријских обележја)

Кроз претходне дијаграме показана је конзистентност и тежња модела ка минимуму што обећава да се и овај тип модуларности, као и претходни, може искористити за процену потрошње електричне енергије на месечном нивоу и у скупу са хетерогеним потрошачима какав је коришћен за тестирање модела у овом истраживању.

Предности оваквог једног модела се огледају кроз више аспеката, а можда се највећа предност истиче већ у фази прикупљања и припреме улаза у мрежу. Ово је посебно значајно јер се често дешава да подаци морају бити прикупљени из више извора што их чини различитим како по облику у коме се чувају, типу података, али и по разним другим карактеристикама.

Мултимодалност омогућује истраживачима да користе истовремено више извора података али и да формирају процедуре за обраду сродних група података што даље пружа разне могућности. У том смислу би се у будућим истраживањима могли користити модалитети података који потичу из иницијалног скупа података одвојено од обележја која су накнадно укључена у конкретан скуп из других извора и обележја која су добијена калкулативним методама. Кôд мултимодалног модела приказаног у овом потпоглављу дат је у виду прилога (Прилог 15).

7. ЗАКЉУЧНА РАЗМАТРАЊА И БУДУЋИ ПРАВЦИ ИСТРАЖИВАЊА

Целокупно истраживање инспирисано је потребом за проналажењем решења проблема планирања производње и потрошње електричне енергије као енергента за којим се потреба временом не смањује, већ напротив.

Аутори су и раније користили разне технике машинског учења бавећи се предвиђањем потрошњи електричне енергије, али углавном базирајући се на краткорочне прогнозе. Искорак начињен у овој дисертацији, односи се превасходно на временски оквир за који се врши предвиђање али и структуру потрошача на које се то предвиђање односи.

Кроз неколико поступака тестирано је више алгоритама машинског учења и њихови резултати поређени са резултатима које даје предложени ANN модел.

Предложено решење и приступ приказани у овој дисертацији могу бити од великог значаја а разлози за то су вишеструки. Не само да су обухваћени сви сегменти потрошача на територији града, већ се предвиђање врши на месечном нивоу, управо онако како се врши и њен обрачун и наплата. Охрабрујући резултати, изражени кроз веома ниске грешке и високу предиктивну способност, истичу важност предложеног решења која се може сагледати са неколико аспеката:

- са аспекта произвођача електричне енергије (у циљу производње и обезбеђења оптималне количине електричне енергије);
- са аспекта дистрибутера (каналисања произведене електричне енергије до потрошача различитих категорија);
- са аспекта крајњег потрошача (утврђивање потребне количине електричне енергије за његово оптимално функционисање).

Решавајући дефинисани проблем, предложен је модел заснован на ANN, тестиран на великом броју записа о стварним потрошњама електричне енергије потрошача на нивоу једног града. Обука и тестирање модела употребом реалних података пружају сигурност у добијене резултата, потврђујући способност модела да пронађе голим оком невидљиве обрасце у понашању потрошача. Тако препознати обрасци се даље врло ефикасно могу користити за прогнозе будућих потрошњи. Ово је посебно важно јер су на подручју посматраног града заступљене све могуће категорије и групе потрошача. Шаблони понашања ових разноврсних потрошача доста варирају што додатно отежава сам процес моделовања и тренинга ANN која би била у стању да прати сваког појединца у маси.

Већ у најранијој фази истраживања, постаје јасно да предвиђање потрошњи електричне енергије извршено само на основу обележја из иницијалног скупа, не даје довољно прецизне прогнозе на којима би се могле заснивати даље одлуке. За превазилажење овог проблема, предложен је поступак адаптације скупа обележја, из којих модел црпи драгоцена знања и постиже високе тачности при тестирању на непознатом, разноврсном и обимном скупу потрошача, са грешком од свега ≈±4%.

Корак даље начињен је развојем модела адаптивне структуре, способним да прилагођава број неурона у скривеним слојевима обиму података и величини скупа обележја. Значај аспекта адаптивности предложеног модела истиче се при сегментацији или кластеровању потрошача приликом формирања једноставнијих, хомогенијих група. Просечна апсолутна грешка потрошача добијена на тај начин, у појединачним месецима је углавном нижа од грешке просечне грешке добијене у базном скупу, за све групе и категорије потрошача. Прецизност постаје још уочљивија када се посматра један издвојени потрошач. Поредећи грешке које модел постиже за једног издвојеног

потрошача у различитим кластерима јасно указује на значај кластеровања по основу месеци у години. Просечна грешка појединачног пада испод 2% а у половини месеци у години је чак испод 1%.

Сви приказани резултати добијени су тестирањем хипотеза постављених на почетку истраживања. У том смислу, општа хипотеза мора бити одбачена, јер, на основу података којима располажу дистрибутивне компаније није могуће извршити прецизне прогнозе будућих потрошњи електричне енергије на територији једног града.

Такође, потпуно супротно очекивањима, показало се да временске прилике, иако имају одређеног утицаја, нису кључне за процену потрошње електричне енергије на овом простору. Стога, прва посебна хипотеза, такође мора бити одбачена.

Све друге хипотезе су потврђене кроз раније приказане резултате што отвара врата практичној примени предложеног модела. Једна таква примена је могућа кроз развој кориснички оријентисане апликације, превасходно намењене крајњем кориснику, у циљу планирања рационалније индивидуалне потрошње. Осим тога, апликација би могла бити корисна и дистрибутивним компанијама као помоћ у планирању и оптимизацији дистрибуције, а самим тим и успостављања оптималног система снабдевања електричном енергијом.

Такође, незанемарљиви су и резултати постигнути при препознавању зоне потрошње или годишњег доба. Успешност у одређивању зоне потрошње у износу од 95% и готово потпуна прецизност у одређивању годишњег доба, наглашавају да предложени модел може бити полазна тачка у решавању разних класификационих проблема повезаних са потрошњама електричне енергије.

Иако је модел већ постигао велику прецизност у реалним условима, тестиран над стварним потрошачима, може се размишљати и о његовим евентуалним ограничењима. Може се појавити такав потрошач који ће својим карактеристикама значајно одударати од образаца понашања потрошача који су учествовали у тренирању мреже. Разлози за то могу бити различити али и непредвидиви па се може размишљати и у правцу укључивања додатних обележја која би указивала на те специфичности потрошача.

Планира се и даљи рад у правцу формирања хомогенијих скупова потрошача, фокусирајући се на одређену категорију или групу потрошача, одређену зона града, итд. У циљу још прецизнијих резултата, може се размишљати о иновативнијим, нестандардним моделима, такође заснованим на ANN, попут мултимодалног, представљеног у последњем поглављу дисертације. Увођењем модалитета улазних обележја, њиховим засебним припремама и тренирањем већим делом у независним гранама ANN, може се доћи до прецизних прогноза месечних потрошњи електричне енергије за све потрошаче на подручју града, о чему говоре просечне апсолутне грешке од ≈4,5% односно ≈5% у зависности од модалитета. Ово отвара могућност да скупови података и обележја прикупљени различитим техникама из различитих извора буду процесуирани у једној истој мрежи али на различите начине.

Флексибилност коју испољавају ANN у прилагођавању различитим стварним проблемима подстиче на њихову примену и у решавању оних повезаних са предикцијом потрошње електричне енергије, а резултати то и потврђују. Било да се проблем посматра са становишта једног, групе или великог броја потрошача, прецизности постигнуте предложеним моделом и приказане у овој дисертацији недвосмислено истичу потенцијал ANN у том процесу.

На основу свега раније реченог, са сигурношћу се може рећи да потенцијалне позитивне импликације, произашле из овог истраживања, досежу и даље од потреба индивидуалних корисника и дистрибутивних компанија. Добијени закључци могу допринети ширем пољу управљања и планирања развоја паметних градова и оптимизације њихових енергетских мрежа као вида подршке одрживих енергетских пракси.

Показало се да је могуће успоставити поуздан систем, заснован на техникама машинског учења, способан да предвиди будуће потрошње електричне енергије и у једном хетерогеном систему о чему говоре и резултати тестирања над реалним подацима о потрошњама свих потрошача на подручју града Ужица приказани у овој дисертацији.

КОРИШЋЕНА ЛИТЕРАТУРА

[1] H. L. Willis, *Power distribution planning reference book*, CRC press, 1997.

[2] "Elektroprivreda Srbije", Доступно на: http://www.eps.rs/cir Приступљено: 18. септембра 2023.

[3] A. Pavić and K. Trupinić, "Gubici električne energije u distribucijskoj mreži," *J. Energy Energ.*, vol. 56, no. 2, pp. 182–215, 2007.

[4] N. Arghira, L. Hawarah, S. Ploix, and M. Jacomino, "Prediction of appliances energy use in smart homes," *Energy*, vol. 48, no. 1, pp. 128–134, Dec. 2012, doi: 10.1016/j.energy.2012.04.010.

[5] A. Mosavi and A. Bahmani, "Energy consumption prediction using machine learning; a review," *Energies*, pp. 1–63, 2019.

[6] U. F. I. Abdulrahman, M. Asante, and J. B. Hayfron-Acquah, "A survey of machine learning's electricity consumption models," *Commun. Appl. Electron. CAE*, vol. 7, no. 21, pp. 6–10, 2018.

[7] J. Walther and M. Weigold, "A systematic review on predicting and forecasting the electrical energy consumption in the manufacturing industry," *Energies*, vol. 14, no. 4, p. 968, 2021.

[8] A. Mosavi, M. Salimi, S. Faizollahzadeh Ardabili, T. Rabczuk, S. Shamshirband, and A. R. Varkonyi-Koczy, "State of the Art of Machine Learning Models in Energy Systems, a Systematic Review," *Energies*, vol. 12, no. 7, Art. no. 7, Jan. 2019, doi: 10.3390/en12071301.

[9] Y.-C. Hu, "Electricity consumption prediction using a neural-network-based grey forecasting approach," *J. Oper. Res. Soc.*, vol. 68, no. 10, pp. 1259–1264, Oct. 2017, doi: https://doi.org/10.1057/s41274-016-0150-y.

[10] X. Song, G. Liang, C. Li, and W. Chen, "Electricity consumption prediction for Xinjiang electric energy replacement," *Math. Probl. Eng.*, vol. 2019, pp. 1–11, 2019, doi: https://doi.org/10.1155/2019/3262591.

[11] M. Mordjaoui, S. Haddad, A. Medoued, and A. Laouafi, "Electric load forecasting by using dynamic neural network," *Int. J. Hydrog. Energy*, vol. 42, no. 28, pp. 17655–17663, 2017, doi: 10.1016/j.ijhydene.2017.03.101.

[12] R. Jovanović and I. Božić, "Primena metoda veštačke inteligencije u obnovljivim izvorima energije i energetskoj efikasnosti," *Zb. Medjunarodnog Kongresa O Procesnoj Ind.*, vol. 31, no. 1, pp. 63–81, 2018.

[13] T. Le, M. T. Vo, B. Vo, E. Hwang, S. Rho, and S. W. Baik, "Improving electric energy consumption prediction using CNN and Bi-LSTM," *Appl. Sci.*, vol. 9, no. 20, p. 4237, 2019.

[14] G. K. F. Tso and K. K. W. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007, doi: https://doi.org/10.1016/j.energy.2006.11.010.

[15] S. A. Sarswatula, T. Pugh, and V. Prabhu, "Modeling Energy Consumption using Machine Learning," *Front. Manuf. Technol.*, vol. 2, 2022.

[16] M. Kumar, V. Shenbagaraman, R. N. Shaw, and A. Ghosh, "Predictive data analysis for energy management of a smart factory leading to sustainability," in *Innovations in electrical and electronic engineering*, Springer, 2021, pp. 765–773.

[17] R. R. Rathod and R. D. Garg, "Regional electricity consumption analysis for consumers using data mining techniques and consumer meter reading data," *Int. J. Electr. Power Energy Syst.*, vol. 78, pp. 368–374, 2016, doi: https://doi.org/10.1016/j.ijepes.2015.11.110.

[18] K. Kavaklioglu, "Modeling and prediction of Turkey's electricity consumption using Support Vector Regression," *Appl. Energy*, vol. 88, no. 1, pp. 368–375, 2011.

[19] K. Kavaklioglu, H. Ceylan, H. K. Ozturk, and O. E. Canyurt, "Modeling and prediction of Turkey's electricity consumption using artificial neural networks," *Energy Convers. Manag.*, vol. 50, no. 11, pp. 2719–2727, 2009, doi: https://doi.org/10.1016/j.enconman.2009.06.016.

[20] M. J. Kargar and D. Charsoghi, "Predicting annual electricity consumption in Iran using artificial neural networks (NARX)," *Indian J Sci Res*, vol. 5, no. 1, pp. 231–242, 2014, doi: doi: 10.1080/15567240802706734.

[21] D. Knežević and M. Blagojević, "Classification of electricity consumers using artificial neural networks," *Facta Univ. Ser. Electron. Energ.*, vol. 32, no. 4, pp. 529–538, 2019, doi: 10.2298/FUEE1904529K.

[22] D. Knežević, M. Blagojević, and A. Ranković, "Prediction of electricity consumption using artificial neural networks," presented at the Knowledge management in informatics, Kopaonik, 2020.

[23] A. Nugaliyadde, U. Somaratne, and K. W. Wong, "Predicting electricity consumption using deep recurrent neural networks," *ArXiv Prepr. ArXiv190908182*, 2019.

[24] M. Beccali, M. Cellura, V. L. Brano, and A. Marvuglia, "Short-term prediction of household electricity consumption: Assessing weather sensitivity in a Mediterranean area," *Renew. Sustain. Energy Rev.*, vol. 12, no. 8, pp. 2040–2065, 2008, doi: https://doi.org/10.1016/j.rser.2007.04.010.

[25] P. Tiwari, "Architectural, demographic, and economic causes of electricity consumption in Bombay," *J. Policy Model.*, vol. 22, no. 1, pp. 81–98, 2000.

[26] J. Milojković, V. Litovski, E. fakultet u Nišu, and M. Milić, "Predvidjanje mesečne potrošnje električne energije na nivou prigradske trafostanice," presented at the Naučno-stručni simpozijum Energetska efikasnost, Banja Luka: ENEF 2015, Sep. 2015.

[27] J. Milojković and V. Litovski, "Novi modeli predvidjanja potrošnje električne energije zasnovani na VNM," in *54th ETRAN Conference*, Donji Milanovac, Jun. 2010.

[28] M. K. Kim, J. Cha, E. Lee, V. H. Pham, S. Lee, and N. Theera-Umpon, "Simplified neural network model design with sensitivity analysis and electricity consumption prediction in a commercial building," *Energies*, vol. 12, no. 7, p. 1201, 2019, doi: DOI: 10.3390/en12071201.

[29] M. Zekić-Sušac, R. Scitovski, and A. Has, "Cluster analysis and artificial neural networks in predicting energy efficiency of public buildings as a cost-saving approach," *Croat. Rev. Econ. Bus. Soc. Stat.*, vol. 4, no. 2, pp. 57–66, 2018.

[30] M. K. M. Shapi, N. A. Ramli, and L. J. Awalin, "Energy consumption prediction by using machine learning for smart building: Case study in Malaysia," *Dev. Built Environ.*, vol. 5, p. 100037, 2021.

[31] H. Cai, S. Shen, Q. Lin, X. Li, and H. Xiao, "Predicting the energy consumption of residential buildings for regional electricity supply-side and demand-side management," *IEEE Access*, vol. 7, pp. 30386–30397, 2019, doi: doi: 10.1109/ACCESS.2019.2901257.

[32] J. Yuan, C. Farnham, C. Azuma, and K. Emura, "Predictive artificial neural network models to forecast the seasonal hourly electricity consumption for a University Campus," *Sustain. Cities Soc.*, vol. 42, pp. 82–92, 2018.

[33] K. McElwee, "Using neural nets to predict tomorrow's electric consumption," Medium. Доступно на: https://towardsdatascience.com/using-neural-nets-to-predict-tomorrows-electric-consumption-cc1ae3ae7cc2, Приступљено: 18. септембар 2023.

[34] M. Irfan *et al.*, "Multi-region electricity demand prediction with ensemble deep neural networks," *PLOS ONE*, vol. 18, no. 5, p. e0285456, May 2023, doi: 10.1371/journal.pone.0285456.

[35] S. Aman, M. Frincu, C. Charalampos, U. Noor, Y. Simmhan, and V. Prasanna, "Empirical comparison of prediction methods for electricity consumption forecasting," *Univ. South. Calif. Tech Rep*, pp. 14–942, 2014.

[36] J.-S. Chou and D.-S. Tran, "Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders," *Energy*, 2018, doi: 10.1016/J.ENERGY.2018.09.144.

[37] Y. T. Chae, R. Horesh, Y. Hwang, and Y. M. Lee, "Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings," *Energy Build.*, vol. 111, pp. 184–194, 2016.

[38] T. A. Nakabi and P. Toivanen, "An ANN-based model for learning individual customer behavior in response to electricity prices," *Sustain. Energy Grids Netw.*, vol. 18, p. 100212, Jun. 2019, doi: 10.1016/j.segan.2019.100212.

[39] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 44–55, 2001, doi: doi: 10.1109/59.910780.

[40] A. Selakov, D. Cvijetinović, and S. Ilić, "Short term load forecasting using support vector machines for different consumer types," *J. Process. Energy Agric.*, vol. 16, no. 3, pp. 128–131, 2012.

[41] S. A. Ilić, S. M. Vukmirović, A. M. Erdeljan, and F. J. Kulić, "Hybrid artificial neural network system for short-term load forecasting," *Therm. Sci.*, vol. 16, no. suppl. 1, pp. 215–224, 2012, doi: doi: 10.2298/TSCI120130073I.

[42] N. Amjady and F. Keynia, "Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm," *Energy*, vol. 34, no. 1, pp. 46–57, 2009, doi: doi: https://doi.org/10.1016/j.energy.2008.09.020.

[43] S. Ilić, "Kratkoročno predviđanje potrošnje električne energije u velikim elektroenergetskim sistemima," Univerzitet u Novom Sadu, 2013.

[44] R. Banik, P. Das, S. Ray, and A. Biswas, "Prediction of electrical energy consumption based on machine learning technique," *Electr. Eng.*, vol. 103, no. 2, pp. 909–920, Apr. 2021, doi: 10.1007/s00202-020-01126-z.

[45] S. Shan, B. Cao, and Z. Wu, "Forecasting the Short-Term Electricity Consumption of Building Using a Novel Ensemble Model," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2925740.

[46] K. Li, J. Tian, W. Xue, and G. Tan, "Short-term electricity consumption prediction for buildings using data-driven swarm intelligence based ensemble model," *Energy Build.*, 2020, doi: 10.1016/j.enbuild.2020.110558.

[47] K. Chen, J. Jiang, F. Zheng, and K. Chen, "A novel data-driven approach for residential electricity consumption prediction based on ensemble learning," *Energy*, vol. 150, 2018, doi: 10.1016/J.ENERGY.2018.02.028.

[48] A. Azadeh, M. Saberi, V. Nadimi, M. Iman, and A. Behrooznia, "An integrated intelligent neuro-fuzzy algorithm for long-term electricity consumption: cases of selected EU countries," *Acta Polytech. Hung.*, vol. 7, no. 4, pp. 71–90, 2010, doi: doi 10.1109/ems.2010.56.

[49] A. Azadeh, S. F. Ghaderi, S. Tarverdian, and M. Saberi, "Integration of artificial neural networks and genetic algorithm to predict electrical energy consumption," *Appl. Math. Comput.*, vol. 186, no. 2, pp. 1731–1741, 2007, doi: doi: 10.1109/IECON.2006.348098.

[50] S. G. Yoo and H.-Á. Myriam, "Predicting residential electricity consumption using neural networks: A case study," in *Journal of Physics: Conference Series*, IOP Publishing, 2018, p. 012005. doi: doi: 10.1088/1742-6596/1072/1/012005.

[51] W. Jiang, X. Wu, Y. Gong, W. Yu, and X. Zhong, "Holt–Winters smoothing enhanced by fruit fly optimization algorithm to forecast monthly electricity consumption," *Energy*, vol. 193, 2020, doi: 10.1016/j.energy.2019.116779.

[52] E. A. Frimpong and P. Y. Okyere, "Monthly Energy Consumption Forecasting Using Wavelet Analysis and Radial Basis Function Neural Network," *J. Sci. Technol. Ghana*, vol. 30, no. 2, pp. 157–164, 2010, doi: 10.4314/just.v30i2.60541.

[53] R. F. Berriel, A. T. Lopes, A. Rodrigues, F. M. Varejao, and T. Oliveira-Santos, "Monthly energy consumption forecast: A deep learning approach," *2017 Proc. Int. Jt. Conf. Neural Netw.*, no. May, pp. 4283–4290, 2017.

[54] D. Knežević, M. Blagojević, and A. Ranković, "Electricity Consumption Prediction Model for Improving Energy Efficiency Based on Artificial Neural Networks," *Stud. Inform. Control*, vol. 32, no. 1, pp. 69–79, Mar. 2023, doi: 10.24846/v32i1y202307.

[55] A. Banga, R. Ahuja, and S. Sharma, "Stacking machine learning models to forecast hourly and daily electricity consumption of household using internet of things," *J. Sci. Ind. Res.*, vol. Vol. 80, no. 2021, pp. 894–904, 2021.

[56] D. Knežević, M. Blagojević, and A. Ranković, "Classification of electricity consumers using model based on neural network," in *International Multidisciplinary Conference "Challenges of Contemporary Higher Education*, Conference of Academies for Applied Studies in Serbia (CAASS), 2024, pp. 403–408.

[57] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv Prepr. ArXiv160706450*, 2016.

[58] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 901–909, 2016, doi: https://doi.org/10.48550/arXiv.1602.07868.

[59] S. Xiang and H. Li, "On the effects of batch and weight normalization in generative adversarial networks," *Prepr. ArXiv170403971*, 2017.

[60] M. Zekić-Sušac, R. Scitovski, and A. Has, "Cluster analysis and artificial neural networks in predicting energy efficiency of public buildings as a cost-saving approach," *Croat. Rev. Econ. Bus. Soc. Stat.*, vol. 4, no. 2, pp. 57–66, 2018, doi: 10.2478/crebss-2018-0013.

[61] J. Parraga-Alava, J. D. Moncayo-Nacaza, J. Revelo-Fuelagán, P. D. Rosero-Montalvo, A. Anaya-Isaza, and D. H. Peluffo-Ordóñez, "A data set for electric power consumption forecasting based on socio-demographic features: Data from an area of southern Colombia," *Data Brief*, vol. 29, 2020, doi: 10.1016/j.dib.2020.105246.

[62] M. Alanbar, A. Alfarraj, and M. Alghieth, "Energy Consumption Prediction Using Deep Learning Technique," *Int. J. Interact. Mob. Technol. IJIM*, vol. 14, no. 10, pp. 166–177, 2020.

[63] M. Nikolić and A. Zečević, *Mašinsko učenje*. Beograd: Matematički fakultet, 2019.

[64] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[65] Y. LeCun, Y. Bengio, G. Hinton, and others, "Deep learning nature," vol. Nature 521, pp. 436–444, 2015, doi: https://doi.org/10.1038/nature14539.

[66] M. Kulin, T. Kazaz, E. De Poorter, and I. Moerman, "A survey on machine learning-based performance improvement of wireless networks: PHY, MAC and network layer," *Electronics*, vol. 10, no. 3, p. 318, 2021.

[67] V. Zocca, G. Spacagna, D. Slater, and P. Roelants, *Python deep learning*. Packt Publishing Ltd, 2017.

[68] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[69] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data Second Edition*. 2011. doi: 10.1007/978-3-642-19460-3.

[70] P. Ditthakit, S. Pinthong, N. Salaeh, J. Weekaew, T. Tran, and Q. Pham, "Comparative study of machine learning methods and GR2M model for monthly runoff prediction," *Ain Shams Eng. J.*, vol. 14, Sep. 2022, doi: 10.1016/j.asej.2022.101941.

[71] P. J. Huber, "Robust Estimation of a Location Parameter," *Ann. Math. Stat.*, vol. 35, no. 1, pp. 73–101, 1964.

[72] D. Draxler, "Generalized Huber Regression," Medium, Доступно на: https://towardsdatascience.com/generalized-huber-regression-505afaff24c, Приступљено: 2. октобар 2023.

[73] Tracyrenee, "Is sklearn's Huber Regressor better than LinearRegression?," Medium, Доступно на: https://tracyrenee61.medium.com/is-sklearns-huber-regressor-better-than-linearregression-a8dc4ee2ea66, Приступљено: 2. октобар 2023.

[74] V. A. S. Hernández, R. Monroy, M. A. Medina-Pérez, O. Loyola-González, and F. Herrera, "A Practical Tutorial for Decision Tree Induction: Evaluation Measures for Candidate Splits and Opportunities," *ACM Comput. Surv.*, vol. 54, no. 1, p. 18:1-18:38, 2021, doi: 10.1145/3429739.

[75] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *Stata J.*, vol. 20, no. 1, pp. 3–29, 2020.

[76] G. Biau and E. Scornet, "A Random Forest Guided Tour." arXiv, Nov. 18, 2015. Доступно на: http://arxiv.org/abs/1511.05741, Приступљено: 2. октобар 2023.

[77] D. R. Yehoshua, "Random Forests," Medium, Доступно на: https://medium.com/@roiyeho/random-forests-98892261dc49, Приступљено: 22. август 2023.

[78] J. Bhattacharyya, "Understanding XGBoost Algorithm In Detail," Analytics India Magazine. Доступно на: https://analyticsindiamag.com/xgboost-internal-working-to-make-decision-trees-and-deduce-predictions/, Приступљено: 2. септембар 2023.

[79] L. Demajo, "Explainable AI for Interpretable Credit Scoring," Thesis, University of Malta, 2020. Доступно на: http://skr.rs/zNKq, Приступљено: 4. јул 2023.

[80] Е. Н. Горбачевская, "Классификация нейронных сетей," *Вестник Волжского Университета Им ВН Татищева*, no. 2 (19), pp. 128–134, 2012.

[81] N. Stanković, "Faktori učenja i predviđanje uspešnosti u programiranju primenom veštačkih neuronskih mreža," University of Kragujevac, Faculty of Technical Sciences Čačak, 2021.

[82] F. Vázquez, "Towards Data Science," A "weird" introduction to Deep Learning, Приступљено: 17. јул 2023.

[83] S. Haykin, "Neural networks and learning machines, Hoboken." NJ, USA: Pearson, 2008.

[84] R. Rojas, *Neural Networks-A Systematic Introduction*. Berlin: Springer, 1996.

[85] C. Gershenson, "Artificial neural networks for beginners," *ArXiv Prepr. Cs0308031*, 2003.

[86] "What is Neuroscience? Why is it Important?," CUSABIO, Доступно на: https://www.cusabio.com/Neuroscience.html Приступљено: 2. август 2023.

[87] P. V. and H. Singh, "Deep Learning based Brain Tumour Segmentation," *WSEAS Trans. Comput.*, vol. 19, pp. 234–241, Jan. 2021, doi: 10.37394/23205.2020.19.29.

[88] J. M. Nazzal, I. M. El-Emary, S. A. Najim, A. Ahliyya, and others, "Multilayer perceptron neural network (MLPs) for analyzing the properties of Jordan oil shale," *World Appl. Sci. J.*, vol. 5, no. 5, pp. 546–552, 2008.

[89] U. Orhan, M. Hekim, and M. Ozer, "EEG signals classification using the K-means clustering and a multilayer perceptron neural network model," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13475–13481, 2011, doi: doi: https://doi.org/10.1016/j.eswa.2011.04.149.

[90] J. Kauffmann, M. Esders, L. Ruff, G. Montavon, W. Samek, and K.-R. Müller, "From clustering to cluster explanations via neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022.

[91] D. Ruppert, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," *J. Am. Stat. Assoc.*, 2004, doi: 10.1198/jasa.2004.s339.

[92] Chabacano, *Diagram showing overfitting of a classifier*. 2008. Доступно на: https://commons.wikimedia.org/wiki/File:Overfitting.svg, Приступљено: 29. август 2023.

[93] Miroslava J. Pavlović, "Programski okvir zasnovan na mašinskom učenju za automatizaciju obrade rezultata fotoakustičnih merenja," Doktorska disertacija, Novi Sad, Novi Sad, 2020.

[94] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *28th International Conference on International Conference on Machine Learning*, 2011, pp. 265–272.

[95] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," *Artif. Intell. Rev.*, vol. 39, no. 3, pp. 251–260, 2013.

[96] S. S. Haykin and others, "Neural networks and learning machines/Simon Haykin." New York: Prentice Hall, 2009.

[97] M. I. Jordović-Pavlović, M. M. Stanković, M. N. Popović, Ž. Ćojbašić, S. Galović, and D. D. Markushev, "The application of artificial neural networks in solid-state photoacoustics for the recognition of microphone response effects in the frequency domain," *J. Comput. Electron.*, vol. 19, no. 3, pp. 1268–1280, 2020.

[98] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review.," *Artif. Intell. Rev.*, vol. 39, no. 3, pp. 1–10, 2013.

[99] Y. Kocyigit, A. Alkan, and H. Erol, "Classification of EEG recordings by using fast independent component analysis and artificial neural network," *J. Med. Syst.*, vol. 32, no. 1, pp. 17–20, 2008.

[100] A. Subasi, "EEG signal classification using wavelet feature extraction and a mixture of expert model," *Expert Syst. Appl.*, vol. 32, no. 4, pp. 1084–1093, 2007.

[101] E. D. Übeyli, "Combined neural network model employing wavelet coefficients for EEG signals classification," *Digit. Signal Process.*, vol. 19, no. 2, pp. 297–308, 2009.

[102] M. B. Simonović, "Primena veštačkih neuronskih mreža za kratkoročno predviđanje i analizu sistema daljinskog grejanja," Универзитет у Нишу, Машински факултет, 2016. Доступно на: https://nardus.mpn.gov.rs/handle/123456789/7064, Приступљено: 29. мај 2023.

[103] S. Aleksić, A. Pantić, and D. Pantić, "High electric field stress model of n-channel VDMOSFET based on artificial neural network," *J. Comput. Electron.*, vol. 17, no. 3, pp. 1210–1219, 2018, doi: doi: 10.1007/s10825-018-1167-z.

[104] B. Huval *et al.*, "An empirical evaluation of deep learning on highway driving," *ArXiv Prepr. ArXiv150401716*, 2015.

[105] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," *J. Cheminformatics*, vol. 9, no. 1, pp. 1–13, 2017.

[106] B. S, P. A, and D. Murugan, *A review of Deep learning Techniques for COVID-19 identification on Chest CT images*. 2022.

[107] N. Adaloglou, "In-layer normalization techniques for training very deep neural networks," AI Summer. Доступно на: https://theaisummer.com/normalization/, Приступљено: 15. јун 2023.

[108] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *ArXiv Prepr. ArXiv12125701*, 2012.

[109] K. (Yi) Li, "How to Choose a Learning Rate Scheduler for Neural Networks," neptune.ai. Доступно на: https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler, Приступљено: 14. април 2023.

[110] E. Zvornicanin, "Relation Between Learning Rate and Batch Size | Baeldung on Computer Science.", Доступно на: https://www.baeldung.com/cs/learning-rate-batch-size, Приступљено: 6. јул 2023.

[111] S. Barreto, "Why Mini-Batch Size Is Better Than One Single 'Batch' With All Training Data | Baeldung on Computer Science," Доступно на: https://www.baeldung.com/cs/mini-batch-vs-single-batch-training-data, Приступљено: 21. јун 2023.

[112] Z. Zhuang, "Adaptive Strategies in Non-convex Optimization," University of Science and Technology of China, 2016. Доступно на: http://arxiv.org/abs/2306.10278, Приступљено: 21. јул 2023.

[113] A. Y. Ng, "Feature selection, L 1 vs. L 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[114] CFI Team, "Elastic Net," A regression method that performs variable selection and regularization simultaneously. Доступно на: http://skr.rs/zNKp, Приступљено: 2. јун 2023.

[115] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034. doi: 10.48550/arXiv.1502.01852.

[116] X. Li and J. Zhou, "Short-term power load fitting/forecasting based on composite modified fractal interpolation approaches," *Int. Trans. Electr. Energy Syst.*, vol. 31, no. 1, pp. 1–22, 2021, doi: 10.1002/2050-7038.12709.

[117] A. M. Ranković and D. N. Ćetenović, "Modeling of photovoltaic modules using a gray-box neural network approach," *Therm. Sci.*, vol. 21, no. 6, pp. 2837–2850, 2017, doi: 10.2298/TSCI160322023R.

[118] Y. Xiang, L. Gou, L. He, S. Xia, and W. Wang, "A SVR–ANN combined model based on ensemble EMD for rainfall prediction," *Appl. Soft Comput. J.*, vol. 73, pp. 874–883, 2018, doi: 10.1016/j.asoc.2018.09.018.

[119] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining*, vol. 72. Cham, Switzerland: Springer International Publishing, 2015.

[120] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data Preprocessing for Supervised Learning," *Int. J. Comput. Sci.*, vol. 1, pp. 111–117, Jan. 2006.

[121] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," *ACM SIGKDD Explor. Newsl.*, vol. 3, no. 1, pp. 27–32, 2001.

[122] F. Pargent, F. Pfisterer, J. Thomas, and B. Bischl, "Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features," *Comput. Stat.*, no. 37, pp. 2671–2692, 2022, doi: https://doi.org/10.1007/s00180-022-01207-6.

[123] M. Larionov, "Sampling techniques in bayesian target encoding," *ArXiv Prepr. ArXiv200601317*, 2020.

[124] L. Al Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A preprocessing engine," *J. Comput. Sci.*, vol. 2, no. 9, pp. 735–739, 2006.

[125] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *ArXiv Prepr. ArXiv150306462*, 2015.

[126] A. Singh Chadha, "Becoming Human: Artificial Intelligence Magazine," What Do Normalization and Standardization Mean? When to Normalize Data and When to Standardize Data? Доступно на: https://medium.com/,

[127] I. F. Ilyas and X. Chu, "Outlier detection," in *Data Cleaning*, New York, NY, USA: Association for Computing Machinery, 2019. Доступно на: https://doi.org/10.1145/3310205.3310208, Приступљено: 18. мај 2023.

[128] K. Li, "On integrating information visualization techniques into data mining: A review," *ArXiv Prepr. ArXiv150300202*, 2015.

[129] "Elektrodistribucija Srbije," Tarifni sistem za obračun električne energije. Доступно на: https://elektrodistribucija.rs/propisi/tarifni_sistem.pdf

[130] S. Grafberger, J. Stoyanovich, and S. Schelter, "Lightweight Inspection of Data Preprocessing in Native Machine Learning Pipelines.," in *CIDR*, 2021.

[131] S. Grafberger, S. Guha, J. Stoyanovich, and S. Schelter, "MLINSPECT: A Data Distribution Debugger for Machine Learning Pipelines," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2736–2739.

[132] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Anal.*, vol. 1, no. 1, pp. 1–22, 2016.

[133] S. Raschka, *Python machine learning*. Packt publishing ltd, 2015.

[134] Sadhvi Anunaya, "Analytics Vidhya," Data Preprocessing in Data Mining – A Hands On Guide (Updated 2023). Доступно на: http://skr.rs/zNK5/, Приступљено: 18. мај 2023.

[135] "Popis stanovništva 2022." Доступно на: https://popis2022.stat.gov.rs/sr-Latn/, Приступљено: 15. јул 2023.

[136] "РХМЗ - Републички хидрометеоролошки завод Србије" Доступно на: http://www.hidmet.gov.rs/latin/meteorologija/klimatologija_godisnjaci.php, Приступљено: 15. април 2023.

[137] "Календар са државним празницима," CEKOS IN, Доступно на: http://www.cekos.rs/kalendar/kalendar-sa-dr%C5%BEavnim-praznicima/1/2014, Приступљено: 19. септембар 2022.

[138] "O podeli na godišnja doba u meteorologiji – Meteologos." Доступно на: http://www.meteologos.rs/o-podeli-na-godisnja-doba-u-meteorologiji/, Приступљено: 19. септембар 2022.

[139] Republički propisi, "Metodologija za određivanje cene električne energije za garantovano snabdevanje." Доступно на: https://www.pravno-informacioni-sistem.rs/SlGlasnikPortal/eli/rep/sgrs/drugidrzavniorganiorganizacije/odluka/2018/99/4, Приступљено: 19. септембар 2022.

[140] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy: IEEE, Dec. 2008, pp. 413–422. doi: 10.1109/ICDM.2008.17.

[141] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data TKDD*, vol. 6, no. 1, pp. 1–39, 2012, doi: https://doi.org/10.1145/2133360.2133363.

[142] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proc. Vol.*, vol. 46, no. 20, pp. 12–17, 2013, doi: https://doi.org/10.3182/20130902-3-CN-3020.00044.

[143] D. Knežević, "The visualisation of electricity consumers and electricity consumption in an urban area," in *Science and Higher Education in Function of Sustainable Development*, Oct. 2021.

[144] D. Knežević, M. Blagojević, and A. Ranković, "Monthly electricity consumption prediction: integrating artificial neural networks and calculated attributes," *J. Sci. Ind. Res. JSIR*, vol. 83, no. 1, 2024, doi: https://doi.org/10.56042/jsir.v83i1.3523.

[145] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 7700 LECTU, pp. 437–478, 2012.

[146] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks.," *J. Mach. Learn. Res.*, vol. 10, no. 1, 2009.

[147] M. Mohammadigohari, "Energy consumption forecasting using machine learning," Master's Thesis, Rochester Institute of Technology, Rochester, NY, USA, Rochester, NY, USA, 2021.

[148] G. Goh *et al.*, "Multimodal neurons in artificial neural networks," *Distill*, vol. 6, no. 3, p. e30, 2021.

[149] L. Ma, Z. Lu, L. Shang, and H. Li, "Multimodal Convolutional Neural Networks for Matching Image and Sentence," in *2015 IEEE International Conference on Computer*

Vision (ICCV), Santiago, Chile: IEEE, Dec. 2015, pp. 2623–2631. doi: 10.1109/ICCV.2015.301.

[150] K. Gadzicki, R. Khamsehashari, and C. Zetzsche, "Early vs Late Fusion in Multimodal Convolutional Neural Networks," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, Jul. 2020, pp. 1–6. doi: 10.23919/FUSION45008.2020.9190246.

ПРИЛОГ 1 - Препроцесирање

[ ]:
```
import pandas as pd
import plotly.express as px
from sklearn.ensemble import IsolationForest
from sklearn.linear_model import LinearRegression
import numpy as np
baza="EPSdataset INITIAL"
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
df.loc[(df['Month'] == 'january') & (df['Year'] == 2014), 'MAXtemp'] = 13.8
df.loc[(df['Month'] == 'january') & (df['Year'] == 2014), 'MINtemp'] = -10.4

df.loc[(df['Month'] == 'february') & (df['Year'] == 2014), 'MAXtemp'] = 19.5
df.loc[(df['Month'] == 'february') & (df['Year'] == 2014), 'MINtemp'] = -9

df.loc[(df['Month'] == 'march') & (df['Year'] == 2014), 'MAXtemp'] = 20.3
df.loc[(df['Month'] == 'march') & (df['Year'] == 2014), 'MINtemp'] = -2.4

df.loc[(df['Month'] == 'april') & (df['Year'] == 2014), 'MAXtemp'] = 21.3
df.loc[(df['Month'] == 'april') & (df['Year'] == 2014), 'MINtemp'] = -0.7

df.loc[(df['Month'] == 'may') & (df['Year'] == 2014), 'MAXtemp'] = 25.4
df.loc[(df['Month'] == 'may') & (df['Year'] == 2014), 'MINtemp'] = 0.5

df.loc[(df['Month'] == 'june') & (df['Year'] == 2014), 'MAXtemp'] = 28
df.loc[(df['Month'] == 'june') & (df['Year'] == 2014), 'MINtemp'] = 7

df.loc[(df['Month'] == 'july') & (df['Year'] == 2014), 'MAXtemp'] = 28.4
df.loc[(df['Month'] == 'july') & (df['Year'] == 2014), 'MINtemp'] = 8

df.loc[(df['Month'] == 'august') & (df['Year'] == 2014), 'MAXtemp'] =30.4
df.loc[(df['Month'] == 'august') & (df['Year'] == 2014), 'MINtemp'] =8.6

df.loc[(df['Month'] == 'september') & (df['Year'] == 2014), 'MAXtemp'] =23.4
df.loc[(df['Month'] == 'september') & (df['Year'] == 2014), 'MINtemp'] = 3.7

df.loc[(df['Month'] == 'october') & (df['Year'] == 2014), 'MAXtemp'] = 25.2
df.loc[(df['Month'] == 'october') & (df['Year'] == 2014), 'MINtemp'] = -1.8
```

```
df.loc[(df['Month'] == 'november') & (df['Year'] == 2014), 'MAXtemp'] = 18.2
df.loc[(df['Month'] == 'november') & (df['Year'] == 2014), 'MINtemp'] = -2.6

df.loc[(df['Month'] == 'december') & (df['Year'] == 2014), 'MAXtemp'] = 15
df.loc[(df['Month'] == 'december') & (df['Year'] == 2014), 'MINtemp'] = -13.5

#-------------------------------------------------------------------------
df.loc[(df['Month'] == 'january') & (df['Year'] == 2015), 'MAXtemp'] = 13
df.loc[(df['Month'] == 'january') & (df['Year'] == 2015), 'MINtemp'] = -13.5

df.loc[(df['Month'] == 'february') & (df['Year'] == 2015), 'MAXtemp'] = 13
df.loc[(df['Month'] == 'february') & (df['Year'] == 2015), 'MINtemp'] = -10

df.loc[(df['Month'] == 'march') & (df['Year'] == 2015), 'MAXtemp'] = 18
df.loc[(df['Month'] == 'march') & (df['Year'] == 2015), 'MINtemp'] = -4

df.loc[(df['Month'] == 'april') & (df['Year'] == 2015), 'MAXtemp'] = 23
df.loc[(df['Month'] == 'april') & (df['Year'] == 2015), 'MINtemp'] = -4.5

df.loc[(df['Month'] == 'may') & (df['Year'] == 2015), 'MAXtemp'] = 30
df.loc[(df['Month'] == 'may') & (df['Year'] == 2015), 'MINtemp'] = 4.9

df.loc[(df['Month'] == 'june') & (df['Year'] == 2015), 'MAXtemp'] = 29
df.loc[(df['Month'] == 'june') & (df['Year'] == 2015), 'MINtemp'] = 6.5

df.loc[(df['Month'] == 'july') & (df['Year'] == 2015), 'MAXtemp'] = 32.6
df.loc[(df['Month'] == 'july') & (df['Year'] == 2015), 'MINtemp'] = 11.8

df.loc[(df['Month'] == 'august') & (df['Year'] == 2015), 'MAXtemp'] =33
df.loc[(df['Month'] == 'august') & (df['Year'] == 2015), 'MINtemp'] =11.5

df.loc[(df['Month'] == 'september') & (df['Year'] == 2015), 'MAXtemp'] =35
df.loc[(df['Month'] == 'september') & (df['Year'] == 2015), 'MINtemp'] = 6.4

df.loc[(df['Month'] == 'october') & (df['Year'] == 2015), 'MAXtemp'] = 23
df.loc[(df['Month'] == 'october') & (df['Year'] == 2015), 'MINtemp'] = 1.1

df.loc[(df['Month'] == 'november') & (df['Year'] == 2015), 'MAXtemp'] = 8
df.loc[(df['Month'] == 'november') & (df['Year'] == 2015), 'MINtemp'] = -3

df.loc[(df['Month'] == 'december') & (df['Year'] == 2015), 'MAXtemp'] = 10
df.loc[(df['Month'] == 'december') & (df['Year'] == 2015), 'MINtemp'] = -8.6
#-------------------------------------------------------------------------
df.loc[(df['Month'] == 'january') & (df['Year'] == 2016), 'MAXtemp'] = 15
df.loc[(df['Month'] == 'january') & (df['Year'] == 2016), 'MINtemp'] = -15

df.loc[(df['Month'] == 'february') & (df['Year'] == 2016), 'MAXtemp'] = 19.5
```

```
df.loc[(df['Month'] == 'february') & (df['Year'] == 2016), 'MINtemp'] = -3.6

df.loc[(df['Month'] == 'march') & (df['Year'] == 2016), 'MAXtemp'] = 22.1
df.loc[(df['Month'] == 'march') & (df['Year'] == 2016), 'MINtemp'] = -3

df.loc[(df['Month'] == 'april') & (df['Year'] == 2016), 'MAXtemp'] = 26.2
df.loc[(df['Month'] == 'april') & (df['Year'] == 2016), 'MINtemp'] = -1.2

df.loc[(df['Month'] == 'may') & (df['Year'] == 2016), 'MAXtemp'] = 27.3
df.loc[(df['Month'] == 'may') & (df['Year'] == 2016), 'MINtemp'] = 1

df.loc[(df['Month'] == 'june') & (df['Year'] == 2016), 'MAXtemp'] = 29.2
df.loc[(df['Month'] == 'june') & (df['Year'] == 2016), 'MINtemp'] = 9.6

df.loc[(df['Month'] == 'july') & (df['Year'] == 2016), 'MAXtemp'] = 31.1
df.loc[(df['Month'] == 'july') & (df['Year'] == 2016), 'MINtemp'] = 9.7

df.loc[(df['Month'] == 'august') & (df['Year'] == 2016), 'MAXtemp'] =28.1
df.loc[(df['Month'] == 'august') & (df['Year'] == 2016), 'MINtemp'] =9

df.loc[(df['Month'] == 'september') & (df['Year'] == 2016), 'MAXtemp'] =26.2
df.loc[(df['Month'] == 'september') & (df['Year'] == 2016), 'MINtemp'] = 6.6

df.loc[(df['Month'] == 'october') & (df['Year'] == 2016), 'MAXtemp'] = 22.6
df.loc[(df['Month'] == 'october') & (df['Year'] == 2016), 'MINtemp'] = 0.4

df.loc[(df['Month'] == 'november') & (df['Year'] == 2016), 'MAXtemp'] = 18.4
df.loc[(df['Month'] == 'november') & (df['Year'] == 2016), 'MINtemp'] = -6.9

df.loc[(df['Month'] == 'december') & (df['Year'] == 2016), 'MAXtemp'] = 16.1
df.loc[(df['Month'] == 'december') & (df['Year'] == 2016), 'MINtemp'] = -9.4

#------------------------------------------------------------------------

df.loc[(df['Month'] == 'january') & (df['Year'] == 2017), 'MAXtemp'] = 9.2
df.loc[(df['Month'] == 'january') & (df['Year'] == 2017), 'MINtemp'] = -19.2

df.loc[(df['Month'] == 'february') & (df['Year'] == 2017), 'MAXtemp'] = 17.1
df.loc[(df['Month'] == 'february') & (df['Year'] == 2017), 'MINtemp'] = -7

df.loc[(df['Month'] == 'march') & (df['Year'] == 2017), 'MAXtemp'] = 22.2
df.loc[(df['Month'] == 'march') & (df['Year'] == 2017), 'MINtemp'] = -1.5

df.loc[(df['Month'] == 'april') & (df['Year'] == 2017), 'MAXtemp'] = 23.4
df.loc[(df['Month'] == 'april') & (df['Year'] == 2017), 'MINtemp'] = -3.5

df.loc[(df['Month'] == 'may') & (df['Year'] == 2017), 'MAXtemp'] = 25.4
```

```
df.loc[(df['Month'] == 'may') & (df['Year'] == 2017), 'MINtemp'] = 2

df.loc[(df['Month'] == 'june') & (df['Year'] == 2017), 'MAXtemp'] = 31
df.loc[(df['Month'] == 'june') & (df['Year'] == 2017), 'MINtemp'] = 8.5

df.loc[(df['Month'] == 'july') & (df['Year'] == 2017), 'MAXtemp'] = 34.1
df.loc[(df['Month'] == 'july') & (df['Year'] == 2017), 'MINtemp'] = 10.6

df.loc[(df['Month'] == 'august') & (df['Year'] == 2017), 'MAXtemp'] =36
df.loc[(df['Month'] == 'august') & (df['Year'] == 2017), 'MINtemp'] =10.6

df.loc[(df['Month'] == 'september') & (df['Year'] == 2017), 'MAXtemp'] =32
df.loc[(df['Month'] == 'september') & (df['Year'] == 2017), 'MINtemp'] = 4.5

df.loc[(df['Month'] == 'october') & (df['Year'] == 2017), 'MAXtemp'] = 25
df.loc[(df['Month'] == 'october') & (df['Year'] == 2017), 'MINtemp'] = -0.5

df.loc[(df['Month'] == 'november') & (df['Year'] == 2017), 'MAXtemp'] = 16.4
df.loc[(df['Month'] == 'november') & (df['Year'] == 2017), 'MINtemp'] = -4.5

df.loc[(df['Month'] == 'december') & (df['Year'] == 2017), 'MAXtemp'] = 15.6
df.loc[(df['Month'] == 'december') & (df['Year'] == 2017), 'MINtemp'] = -6.8

#-------------------------------------------------------------------------

df.loc[(df['Month'] == 'january') & (df['Year'] == 2018), 'MAXtemp'] = 14.5
df.loc[(df['Month'] == 'january') & (df['Year'] == 2018), 'MINtemp'] = -17.6

df.loc[(df['Month'] == 'february') & (df['Year'] == 2018), 'MAXtemp'] = 13.6
df.loc[(df['Month'] == 'february') & (df['Year'] == 2018), 'MINtemp'] = -14.2

df.loc[(df['Month'] == 'march') & (df['Year'] == 2018), 'MAXtemp'] = 20.6
df.loc[(df['Month'] == 'march') & (df['Year'] == 2018), 'MINtemp'] = -15

df.loc[(df['Month'] == 'april') & (df['Year'] == 2018), 'MAXtemp'] = 26
df.loc[(df['Month'] == 'april') & (df['Year'] == 2018), 'MINtemp'] = -0.2

df.loc[(df['Month'] == 'may') & (df['Year'] == 2018), 'MAXtemp'] = 26.4
df.loc[(df['Month'] == 'may') & (df['Year'] == 2018), 'MINtemp'] = 6.2

df.loc[(df['Month'] == 'june') & (df['Year'] == 2018), 'MAXtemp'] = 30.2
df.loc[(df['Month'] == 'june') & (df['Year'] == 2018), 'MINtemp'] = 6.6

df.loc[(df['Month'] == 'july') & (df['Year'] == 2018), 'MAXtemp'] = 28
df.loc[(df['Month'] == 'july') & (df['Year'] == 2018), 'MINtemp'] = 10

df.loc[(df['Month'] == 'august') & (df['Year'] == 2018), 'MAXtemp'] =28.1
```

```
df.loc[(df['Month'] == 'august') & (df['Year'] == 2018), 'MINtemp'] =11
#-----------------------------**********----------------------------------------------------

# IZVOR ZA DaylightHours  / SunshineHours
#  https://www.aladin.info/sr/srbija/uzice-klima#daylight_sunshine
df.loc[(df['Month'] == 'january') , 'DaylightHours'] = 9.4
df.loc[(df['Month'] == 'february') , 'DaylightHours'] =10.5
df.loc[(df['Month'] == 'march') , 'DaylightHours'] =12
df.loc[(df['Month'] == 'april') , 'DaylightHours'] =13.5
df.loc[(df['Month'] == 'may') , 'DaylightHours'] =14.8
df.loc[(df['Month'] == 'june') , 'DaylightHours'] =15.4
df.loc[(df['Month'] == 'july') , 'DaylightHours'] =15.1
df.loc[(df['Month'] == 'august') , 'DaylightHours'] =13.9
df.loc[(df['Month'] == 'september') , 'DaylightHours'] =12.5
df.loc[(df['Month'] == 'october') , 'DaylightHours'] =11
df.loc[(df['Month'] == 'november') , 'DaylightHours'] =9.7
df.loc[(df['Month'] == 'december') , 'DaylightHours'] =9
#----------
df.loc[(df['Month'] == 'january') , 'SunshineHours'] = 4.4
df.loc[(df['Month'] == 'february') , 'SunshineHours'] =4.1
df.loc[(df['Month'] == 'march') , 'SunshineHours'] =6.2
df.loc[(df['Month'] == 'april') , 'SunshineHours'] =8.9
df.loc[(df['Month'] == 'may') , 'SunshineHours'] =9.4
df.loc[(df['Month'] == 'june') , 'SunshineHours'] =9.9
df.loc[(df['Month'] == 'july') , 'SunshineHours'] =11
df.loc[(df['Month'] == 'august') , 'SunshineHours'] =11.3
df.loc[(df['Month'] == 'september') , 'SunshineHours'] =9
df.loc[(df['Month'] == 'october') , 'SunshineHours'] =7
df.loc[(df['Month'] == 'november') , 'SunshineHours'] =6
df.loc[(df['Month'] == 'december') , 'SunshineHours'] =4.9
```

[ ]:
```
df.loc[(df['CalculationPeriod'] == 'january 2014.'), 'MAXtemp'] = 13.8
df.loc[(df['CalculationPeriod'] == 'january 2014.'),  'MINtemp'] = -10.4

df.loc[(df['CalculationPeriod'] == 'february 2014.') , 'MAXtemp'] = 19.5
df.loc[(df['CalculationPeriod'] == 'february 2014.') , 'MINtemp'] = -9

df.loc[(df['CalculationPeriod'] == 'march 2014.'), 'MAXtemp'] = 20.3
df.loc[(df['CalculationPeriod'] == 'march 2014.'), 'MINtemp'] = -2.4

df.loc[(df['CalculationPeriod'] == 'april 2014.') , 'MAXtemp'] = 21.3
df.loc[(df['CalculationPeriod'] == 'april 2014.') , 'MINtemp'] = -0.7

df.loc[(df['CalculationPeriod'] == 'may 2014.'), 'MAXtemp'] = 25.4
df.loc[(df['CalculationPeriod'] == 'may 2014.') , 'MINtemp'] = 0.5

df.loc[(df['CalculationPeriod'] == 'june 2014.') , 'MAXtemp'] = 28
```

```
df.loc[(df['CalculationPeriod'] == 'june 2014.') , 'MINtemp'] = 7

df.loc[(df['CalculationPeriod'] == 'july 2014.'), 'MAXtemp'] = 28.4
df.loc[(df['CalculationPeriod'] == 'july 2014.') , 'MINtemp'] = 8

df.loc[(df['CalculationPeriod'] == 'august 2014.'), 'MAXtemp'] =30.4
df.loc[(df['CalculationPeriod'] == 'august 2014.'), 'MINtemp'] =8.6

df.loc[(df['CalculationPeriod'] == 'september 2014.'), 'MAXtemp'] =23.4
df.loc[(df['CalculationPeriod'] == 'september 2014.'), 'MINtemp'] = 3.7

df.loc[(df['CalculationPeriod'] == 'october 2014.'), 'MAXtemp'] = 25.2
df.loc[(df['CalculationPeriod'] == 'october 2014.') , 'MINtemp'] = -1.8

df.loc[(df['CalculationPeriod'] == 'november 2014.'), 'MAXtemp'] = 18.2
df.loc[(df['CalculationPeriod'] == 'november 2014.') , 'MINtemp'] = -2.6

df.loc[(df['CalculationPeriod'] == 'december 2014.') , 'MAXtemp'] = 15
df.loc[(df['CalculationPeriod'] == 'december 2014.') , 'MINtemp'] = -13.5

#-------------------------------------------------------------------------
df.loc[(df['CalculationPeriod'] == 'january 2015.'), 'MAXtemp'] = 13
df.loc[(df['CalculationPeriod'] == 'january 2015.') , 'MINtemp'] = -13.5

df.loc[(df['CalculationPeriod'] == 'february 2015.') , 'MAXtemp'] = 13
df.loc[(df['CalculationPeriod'] == 'february 2015.') , 'MINtemp'] = -10

df.loc[(df['CalculationPeriod'] == 'march 2015.') , 'MAXtemp'] = 18
df.loc[(df['CalculationPeriod'] == 'march 2015.') , 'MINtemp'] = -4

df.loc[(df['CalculationPeriod'] == 'april 2015.') , 'MAXtemp'] = 23
df.loc[(df['CalculationPeriod'] == 'april 2015.') , 'MINtemp'] = -4.5

df.loc[(df['CalculationPeriod'] == 'may 2015.') , 'MAXtemp'] = 30
df.loc[(df['CalculationPeriod'] == 'may 2015.') , 'MINtemp'] = 4.9

df.loc[(df['CalculationPeriod'] == 'june 2015.') , 'MAXtemp'] = 29
df.loc[(df['CalculationPeriod'] == 'june 2015.') , 'MINtemp'] = 6.5

df.loc[(df['CalculationPeriod'] == 'july 2015.') , 'MAXtemp'] = 32.6
df.loc[(df['CalculationPeriod'] == 'july 2015.') , 'MINtemp'] = 11.8

df.loc[(df['CalculationPeriod'] == 'august 2015.') , 'MAXtemp'] =33
df.loc[(df['CalculationPeriod'] == 'august 2015.') , 'MINtemp'] =11.5

df.loc[(df['CalculationPeriod'] == 'september 2015.') , 'MAXtemp'] =35
df.loc[(df['CalculationPeriod'] == 'september 2015.') , 'MINtemp'] = 6.4
```

```
df.loc[(df['CalculationPeriod'] == 'october 2015.') , 'MAXtemp'] = 23
df.loc[(df['CalculationPeriod'] == 'october 2015.') , 'MINtemp'] = 1.1

df.loc[(df['CalculationPeriod'] == 'november 2015.') , 'MAXtemp'] = 8
df.loc[(df['CalculationPeriod'] == 'november 2015.') , 'MINtemp'] = -3

df.loc[(df['CalculationPeriod'] == 'december 2015.') , 'MAXtemp'] = 10
df.loc[(df['CalculationPeriod'] == 'december 2015.') , 'MINtemp'] = -8.6
#-------------------------------------------------------------------------
df.loc[(df['CalculationPeriod'] == 'january 2016.') , 'MAXtemp'] = 15
df.loc[(df['CalculationPeriod'] == 'january 2016.') , 'MINtemp'] = -15

df.loc[(df['CalculationPeriod'] == 'february 2016.') , 'MAXtemp'] = 19.5
df.loc[(df['CalculationPeriod'] == 'february 2016.') , 'MINtemp'] = -3.6

df.loc[(df['CalculationPeriod'] == 'march 2016.') , 'MAXtemp'] = 22.1
df.loc[(df['CalculationPeriod'] == 'march 2016.') , 'MINtemp'] = -3

df.loc[(df['CalculationPeriod'] == 'april 2016.') , 'MAXtemp'] = 26.2
df.loc[(df['CalculationPeriod'] == 'april 2016.') , 'MINtemp'] = -1.2

df.loc[(df['CalculationPeriod'] == 'may 2016.') , 'MAXtemp'] = 27.3
df.loc[(df['CalculationPeriod'] == 'may 2016.') , 'MINtemp'] = 1

df.loc[(df['CalculationPeriod'] == 'june 2016.') , 'MAXtemp'] = 29.2
df.loc[(df['CalculationPeriod'] == 'june 2016.') , 'MINtemp'] = 9.6

df.loc[(df['CalculationPeriod'] == 'july 2016.') , 'MAXtemp'] = 31.1
df.loc[(df['CalculationPeriod'] == 'july 2016.') , 'MINtemp'] = 9.7

df.loc[(df['CalculationPeriod'] == 'august 2016.') , 'MAXtemp'] =28.1
df.loc[(df['CalculationPeriod'] == 'august 2016.') , 'MINtemp'] =9

df.loc[(df['CalculationPeriod'] == 'september 2016.') , 'MAXtemp'] =26.2
df.loc[(df['CalculationPeriod'] == 'september 2016.') , 'MINtemp'] = 6.6

df.loc[(df['CalculationPeriod'] == 'october 2016.') , 'MAXtemp'] = 22.6
df.loc[(df['CalculationPeriod'] == 'october 2016.') , 'MINtemp'] = 0.4

df.loc[(df['CalculationPeriod'] == 'november 2016.') , 'MAXtemp'] = 18.4
df.loc[(df['CalculationPeriod'] == 'november 2016.') , 'MINtemp'] = -6.9

df.loc[(df['CalculationPeriod'] == 'december 2016.') , 'MAXtemp'] = 16.1
df.loc[(df['CalculationPeriod'] == 'december 2016.') , 'MINtemp'] = -9.4

#-------------------------------------------------------------------------
```

```
df.loc[(df['CalculationPeriod'] == 'january 2017.') , 'MAXtemp'] = 9.2
df.loc[(df['CalculationPeriod'] == 'january 2017.') , 'MINtemp'] = -19.2

df.loc[(df['CalculationPeriod'] == 'february 2017.') , 'MAXtemp'] = 17.1
df.loc[(df['CalculationPeriod'] == 'february 2017.') , 'MINtemp'] = -7

df.loc[(df['CalculationPeriod'] == 'march 2017.') , 'MAXtemp'] = 22.2
df.loc[(df['CalculationPeriod'] == 'march 2017.') , 'MINtemp'] = -1.5

df.loc[(df['CalculationPeriod'] == 'april 2017.') , 'MAXtemp'] = 23.4
df.loc[(df['CalculationPeriod'] == 'april 2017.') , 'MINtemp'] = -3.5

df.loc[(df['CalculationPeriod'] == 'may 2017.') , 'MAXtemp'] = 25.4
df.loc[(df['CalculationPeriod'] == 'may 2017.') , 'MINtemp'] = 2

df.loc[(df['CalculationPeriod'] == 'june 2017.') , 'MAXtemp'] = 31
df.loc[(df['CalculationPeriod'] == 'june 2017.') , 'MINtemp'] = 8.5

df.loc[(df['CalculationPeriod'] == 'july 2017.') , 'MAXtemp'] = 34.1
df.loc[(df['CalculationPeriod'] == 'july 2017.') , 'MINtemp'] = 10.6

df.loc[(df['CalculationPeriod'] == 'august 2017.') , 'MAXtemp'] =36
df.loc[(df['CalculationPeriod'] == 'august 2017.') , 'MINtemp'] =10.6

df.loc[(df['CalculationPeriod'] == 'september 2017.') , 'MAXtemp'] =32
df.loc[(df['CalculationPeriod'] == 'september 2017.') , 'MINtemp'] = 4.5

df.loc[(df['CalculationPeriod'] == 'october 2017.') , 'MAXtemp'] = 25
df.loc[(df['CalculationPeriod'] == 'october 2017.') , 'MINtemp'] = -0.5

df.loc[(df['CalculationPeriod'] == 'november 2017.') , 'MAXtemp'] = 16.4
df.loc[(df['CalculationPeriod'] == 'november 2017.') , 'MINtemp'] = -4.5

df.loc[(df['CalculationPeriod'] == 'december 2017.') , 'MAXtemp'] = 15.6
df.loc[(df['CalculationPeriod'] == 'december 2017.') , 'MINtemp'] = -6.8

#-------------------------------------------------------------------------

df.loc[(df['CalculationPeriod'] == 'january 2018.') , 'MAXtemp'] = 14.5
df.loc[(df['CalculationPeriod'] == 'january 2018.') , 'MINtemp'] = -17.6

df.loc[(df['CalculationPeriod'] == 'february 2018.') , 'MAXtemp'] = 13.6
df.loc[(df['CalculationPeriod'] == 'february 2018.') , 'MINtemp'] = -14.2

df.loc[(df['CalculationPeriod'] == 'march 2018.'), 'MAXtemp'] = 20.6
df.loc[(df['CalculationPeriod'] == 'march 2018.') , 'MINtemp'] = -15
```

```
df.loc[(df['CalculationPeriod'] == 'april 2018.') , 'MAXtemp'] = 26
df.loc[(df['CalculationPeriod'] == 'april 2018.') , 'MINtemp'] = -0.2

df.loc[(df['CalculationPeriod'] == 'may 2018.') , 'MAXtemp'] = 26.4
df.loc[(df['CalculationPeriod'] == 'may 2018.') , 'MINtemp'] = 6.2

df.loc[(df['CalculationPeriod'] == 'june 2018.') , 'MAXtemp'] = 30.2
df.loc[(df['CalculationPeriod'] == 'june 2018.') , 'MINtemp'] = 6.6

df.loc[(df['CalculationPeriod'] == 'july 2018.') , 'MAXtemp'] = 28
df.loc[(df['CalculationPeriod'] == 'july 2018.') , 'MINtemp'] = 10

df.loc[(df['CalculationPeriod'] == 'august 2018.') , 'MAXtemp'] =28.1
df.loc[(df['CalculationPeriod'] == 'august 2018.') , 'MINtemp'] =11
#---------------------------***********-----------------------------------------------

#   DNEVNA SVETLOST PROSECNO
df.loc[(df['CalculationPeriod'] == 'january 2014.'), 'NoOfClearDays'] =3
df.loc[(df['CalculationPeriod'] == 'january 2014.'),  'NoOfCloudyDays'] =14
df.loc[(df['CalculationPeriod'] == 'january 2014.'), 'InsolationInHours'] =94.1

df.loc[(df['CalculationPeriod'] == 'february 2014.') , 'NoOfClearDays'] = 3
df.loc[(df['CalculationPeriod'] == 'february 2014.') , 'NoOfCloudyDays'] = 7
df.loc[(df['CalculationPeriod'] == 'february 2014.'), 'InsolationInHours'] =115.
 ↪1

df.loc[(df['CalculationPeriod'] == 'march 2014.'), 'NoOfClearDays'] = 7
df.loc[(df['CalculationPeriod'] == 'march 2014.'), 'NoOfCloudyDays'] = 11
df.loc[(df['CalculationPeriod'] == 'march 2014.'), 'InsolationInHours'] =180.6

df.loc[(df['CalculationPeriod'] == 'april 2014.') , 'NoOfClearDays'] = 0
df.loc[(df['CalculationPeriod'] == 'april 2014.') , 'NoOfCloudyDays'] = 12
df.loc[(df['CalculationPeriod'] == 'april 2014.'), 'InsolationInHours'] =109.8

df.loc[(df['CalculationPeriod'] == 'may 2014.'), 'NoOfClearDays'] = 2
df.loc[(df['CalculationPeriod'] == 'may 2014.') , 'NoOfCloudyDays'] = 14
df.loc[(df['CalculationPeriod'] == 'may 2014.'), 'InsolationInHours'] =189.7

df.loc[(df['CalculationPeriod'] == 'june 2014.') , 'NoOfClearDays'] = 5
df.loc[(df['CalculationPeriod'] == 'june 2014.') , 'NoOfCloudyDays'] = 8
df.loc[(df['CalculationPeriod'] == 'june 2014.'), 'InsolationInHours'] =225.2

df.loc[(df['CalculationPeriod'] == 'july 2014.'), 'NoOfClearDays'] = 2
df.loc[(df['CalculationPeriod'] == 'july 2014.') , 'NoOfCloudyDays'] =4
df.loc[(df['CalculationPeriod'] == 'july 2014.'), 'InsolationInHours'] =278.4
```

```
df.loc[(df['CalculationPeriod'] == 'august 2014.'), 'NoOfClearDays'] =7
df.loc[(df['CalculationPeriod'] == 'august 2014.'), 'NoOfCloudyDays'] =3
df.loc[(df['CalculationPeriod'] == 'august 2014.'), 'InsolationInHours'] =268.6

df.loc[(df['CalculationPeriod'] == 'september 2014.'), 'NoOfClearDays'] =3
df.loc[(df['CalculationPeriod'] == 'september 2014.'), 'NoOfCloudyDays'] =11
df.loc[(df['CalculationPeriod'] == 'september 2014.'), 'InsolationInHours'] =125

df.loc[(df['CalculationPeriod'] == 'october 2014.'), 'NoOfClearDays'] = 8
df.loc[(df['CalculationPeriod'] == 'october 2014.') , 'NoOfCloudyDays'] =13
df.loc[(df['CalculationPeriod'] == 'october 2014.'), 'InsolationInHours'] =137.7

df.loc[(df['CalculationPeriod'] == 'november 2014.'), 'NoOfClearDays'] = 5
df.loc[(df['CalculationPeriod'] == 'november 2014.') , 'NoOfCloudyDays'] =13
df.loc[(df['CalculationPeriod'] == 'november 2014.'), 'InsolationInHours'] =94.6

df.loc[(df['CalculationPeriod'] == 'december 2014.') , 'NoOfClearDays'] = 7
df.loc[(df['CalculationPeriod'] == 'december 2014.') , 'NoOfCloudyDays'] = 16
df.loc[(df['CalculationPeriod'] == 'december 2014.'), 'InsolationInHours'] =66.4

#-------------------------------------------------------------------------
df.loc[(df['CalculationPeriod'] == 'january 2015.'), 'NoOfClearDays'] = 3
df.loc[(df['CalculationPeriod'] == 'january 2015.') , 'NoOfCloudyDays'] =17
df.loc[(df['CalculationPeriod'] == 'january 2015.') , 'InsolationInHours'] =80.8

df.loc[(df['CalculationPeriod'] == 'february 2015.') , 'NoOfClearDays'] = 6
df.loc[(df['CalculationPeriod'] == 'february 2015.') , 'NoOfCloudyDays'] = 12
df.loc[(df['CalculationPeriod'] == 'february 2015.') , 'InsolationInHours']␣
 ↪=101.4

df.loc[(df['CalculationPeriod'] == 'march 2015.') , 'NoOfClearDays'] = 3
df.loc[(df['CalculationPeriod'] == 'march 2015.') , 'NoOfCloudyDays'] = 16
df.loc[(df['CalculationPeriod'] == 'march 2015.') , 'InsolationInHours'] =108.1

df.loc[(df['CalculationPeriod'] == 'april 2015.') , 'NoOfClearDays'] = 6
df.loc[(df['CalculationPeriod'] == 'april 2015.') , 'NoOfCloudyDays'] = 6
df.loc[(df['CalculationPeriod'] == 'april 2015.') , 'InsolationInHours'] =210.6

df.loc[(df['CalculationPeriod'] == 'may 2015.') , 'NoOfClearDays'] = 4
df.loc[(df['CalculationPeriod'] == 'may 2015.') , 'NoOfCloudyDays'] = 10
df.loc[(df['CalculationPeriod'] == 'may 2015.') , 'InsolationInHours'] =206.2

df.loc[(df['CalculationPeriod'] == 'june 2015.') , 'NoOfClearDays'] = 4
df.loc[(df['CalculationPeriod'] == 'june 2015.') , 'NoOfCloudyDays'] = 7
df.loc[(df['CalculationPeriod'] == 'june 2015.') , 'InsolationInHours'] =242.8

df.loc[(df['CalculationPeriod'] == 'july 2015.') , 'NoOfClearDays'] = 11
```

```
df.loc[(df['CalculationPeriod'] == 'july 2015.') , 'NoOfCloudyDays'] = 2
df.loc[(df['CalculationPeriod'] == 'july 2015.') , 'InsolationInHours'] =325.3

df.loc[(df['CalculationPeriod'] == 'august 2015.') , 'NoOfClearDays'] =12
df.loc[(df['CalculationPeriod'] == 'august 2015.') , 'NoOfCloudyDays'] =4
df.loc[(df['CalculationPeriod'] == 'august 2015.') , 'InsolationInHours']=269.3

df.loc[(df['CalculationPeriod'] == 'september 2015.') , 'NoOfClearDays'] =5
df.loc[(df['CalculationPeriod'] == 'september 2015.') , 'NoOfCloudyDays'] =12
df.loc[(df['CalculationPeriod'] == 'september 2015.') , 'InsolationInHours']␣
 ↪=185.2

df.loc[(df['CalculationPeriod'] == 'october 2015.') , 'NoOfClearDays'] = 0
df.loc[(df['CalculationPeriod'] == 'october 2015.') , 'NoOfCloudyDays'] = 12
df.loc[(df['CalculationPeriod'] == 'october 2015.') , 'InsolationInHours'] =123.
 ↪7

df.loc[(df['CalculationPeriod'] == 'november 2015.') , 'NoOfClearDays'] = 12
df.loc[(df['CalculationPeriod'] == 'november 2015.') , 'NoOfCloudyDays'] = 7
df.loc[(df['CalculationPeriod'] == 'november 2015.') , 'InsolationInHours']␣
 ↪=154.1

df.loc[(df['CalculationPeriod'] == 'december 2015.') , 'NoOfClearDays'] = 15
df.loc[(df['CalculationPeriod'] == 'december 2015.') , 'NoOfCloudyDays'] = 7
df.loc[(df['CalculationPeriod'] == 'december 2015.') , 'InsolationInHours'] =163

#-------------------------------------------------------------------------
df.loc[(df['CalculationPeriod'] == 'january 2016.') , 'NoOfClearDays'] = 2
df.loc[(df['CalculationPeriod'] == 'january 2016.') , 'NoOfCloudyDays'] = 14
df.loc[(df['CalculationPeriod'] == 'january 2016.') , 'InsolationInHours'] =80.5

df.loc[(df['CalculationPeriod'] == 'february 2016.') , 'NoOfClearDays'] = 1
df.loc[(df['CalculationPeriod'] == 'february 2016.') , 'NoOfCloudyDays'] = 7
df.loc[(df['CalculationPeriod'] == 'february 2016.') , 'InsolationInHours'] =91.
 ↪8

df.loc[(df['CalculationPeriod'] == 'march 2016.') , 'NoOfClearDays'] = 4
df.loc[(df['CalculationPeriod'] == 'march 2016.') , 'NoOfCloudyDays'] = 14
df.loc[(df['CalculationPeriod'] == 'march 2016.') , 'InsolationInHours'] =106.8

df.loc[(df['CalculationPeriod'] == 'april 2016.') , 'NoOfClearDays'] = 7
df.loc[(df['CalculationPeriod'] == 'april 2016.') , 'NoOfCloudyDays'] = 10
df.loc[(df['CalculationPeriod'] == 'april 2016.') , 'InsolationInHours'] =190.3

df.loc[(df['CalculationPeriod'] == 'may 2016.') , 'NoOfClearDays'] = 4
df.loc[(df['CalculationPeriod'] == 'may 2016.') , 'NoOfCloudyDays'] = 8
df.loc[(df['CalculationPeriod'] == 'may 2016.') , 'InsolationInHours'] =192.2
```

```
df.loc[(df['CalculationPeriod'] == 'june 2016.') , 'NoOfClearDays'] = 2
df.loc[(df['CalculationPeriod'] == 'june 2016.') , 'NoOfCloudyDays'] = 5
df.loc[(df['CalculationPeriod'] == 'june 2016.') , 'InsolationInHours'] =225.7

df.loc[(df['CalculationPeriod'] == 'july 2016.') , 'NoOfClearDays'] = 10
df.loc[(df['CalculationPeriod'] == 'july 2016.') , 'NoOfCloudyDays'] = 3
df.loc[(df['CalculationPeriod'] == 'july 2016.') , 'InsolationInHours'] =280.2

df.loc[(df['CalculationPeriod'] == 'august 2016.') , 'NoOfClearDays'] =8
df.loc[(df['CalculationPeriod'] == 'august 2016.') , 'NoOfCloudyDays'] =6
df.loc[(df['CalculationPeriod'] == 'august 2016.') , 'InsolationInHours'] =235.8

df.loc[(df['CalculationPeriod'] == 'september 2016.') , 'NoOfClearDays'] =5
df.loc[(df['CalculationPeriod'] == 'september 2016.') , 'NoOfCloudyDays'] =6
df.loc[(df['CalculationPeriod'] == 'september 2016.') , 'InsolationInHours'] =␣
 ↪230.9

df.loc[(df['CalculationPeriod'] == 'october 2016.') , 'NoOfClearDays'] = 0
df.loc[(df['CalculationPeriod'] == 'october 2016.') , 'NoOfCloudyDays'] = 15
df.loc[(df['CalculationPeriod'] == 'october 2016.') , 'InsolationInHours'] =106.
 ↪5

df.loc[(df['CalculationPeriod'] == 'november 2016.') , 'NoOfClearDays'] = 5
df.loc[(df['CalculationPeriod'] == 'november 2016.') , 'NoOfCloudyDays'] = 14
df.loc[(df['CalculationPeriod'] == 'november 2016.') , 'InsolationInHours']␣
 ↪=128.2

df.loc[(df['CalculationPeriod'] == 'december 2016.') , 'NoOfClearDays'] = 12
df.loc[(df['CalculationPeriod'] == 'december 2016.') , 'NoOfCloudyDays'] = 7
df.loc[(df['CalculationPeriod'] == 'december 2016.') , 'InsolationInHours']␣
 ↪=131.4

#-------------------------------------------------------------------------

df.loc[(df['CalculationPeriod'] == 'january 2017.') , 'NoOfClearDays'] = 6
df.loc[(df['CalculationPeriod'] == 'january 2017.') , 'NoOfCloudyDays'] = 14
df.loc[(df['CalculationPeriod'] == 'january 2017.') , 'InsolationInHours'] = 97.
 ↪4

df.loc[(df['CalculationPeriod'] == 'february 2017.') , 'NoOfClearDays'] = 5
df.loc[(df['CalculationPeriod'] == 'february 2017.') , 'NoOfCloudyDays'] = 12
df.loc[(df['CalculationPeriod'] == 'february 2017.') , 'InsolationInHours']␣
 ↪=100.3

df.loc[(df['CalculationPeriod'] == 'march 2017.') , 'NoOfClearDays'] = 7
```

```
df.loc[(df['CalculationPeriod'] == 'march 2017.') , 'NoOfCloudyDays'] = 8
df.loc[(df['CalculationPeriod'] == 'march 2017.') , 'InsolationInHours'] = 185.1

df.loc[(df['CalculationPeriod'] == 'april 2017.') , 'NoOfClearDays'] = 3
df.loc[(df['CalculationPeriod'] == 'april 2017.') , 'NoOfCloudyDays'] = 12
df.loc[(df['CalculationPeriod'] == 'april 2017.') , 'InsolationInHours'] =150.6

df.loc[(df['CalculationPeriod'] == 'may 2017.') , 'NoOfClearDays'] = 5
df.loc[(df['CalculationPeriod'] == 'may 2017.') , 'NoOfCloudyDays'] = 5
df.loc[(df['CalculationPeriod'] == 'may 2017.') , 'InsolationInHours'] =211.7

df.loc[(df['CalculationPeriod'] == 'june 2017.') , 'NoOfClearDays'] = 6
df.loc[(df['CalculationPeriod'] == 'june 2017.') , 'NoOfCloudyDays'] = 5
df.loc[(df['CalculationPeriod'] == 'june 2017.') , 'InsolationInHours'] = 282.5

df.loc[(df['CalculationPeriod'] == 'july 2017.') , 'NoOfClearDays'] = 14
df.loc[(df['CalculationPeriod'] == 'july 2017.') , 'NoOfCloudyDays'] = 2
df.loc[(df['CalculationPeriod'] == 'july 2017.') , 'InsolationInHours'] =318.2

df.loc[(df['CalculationPeriod'] == 'august 2017.') , 'NoOfClearDays'] =17
df.loc[(df['CalculationPeriod'] == 'august 2017.') , 'NoOfCloudyDays'] =3
df.loc[(df['CalculationPeriod'] == 'august 2017.') , 'InsolationInHours'] =300

df.loc[(df['CalculationPeriod'] == 'september 2017.') , 'NoOfClearDays'] =3
df.loc[(df['CalculationPeriod'] == 'september 2017.') , 'NoOfCloudyDays'] = 6
df.loc[(df['CalculationPeriod'] == 'september 2017.') , 'InsolationInHours']␣
 ↪=189

df.loc[(df['CalculationPeriod'] == 'october 2017.') , 'NoOfClearDays'] = 11
df.loc[(df['CalculationPeriod'] == 'october 2017.') , 'NoOfCloudyDays'] = 5
df.loc[(df['CalculationPeriod'] == 'october 2017.') , 'InsolationInHours'] =␣
 ↪204.3

df.loc[(df['CalculationPeriod'] == 'november 2017.') , 'NoOfClearDays'] = 6
df.loc[(df['CalculationPeriod'] == 'november 2017.') , 'NoOfCloudyDays'] = 14
df.loc[(df['CalculationPeriod'] == 'november 2017.') , 'InsolationInHours']␣
 ↪=102.8

df.loc[(df['CalculationPeriod'] == 'december 2017.') , 'NoOfClearDays'] = 3
df.loc[(df['CalculationPeriod'] == 'december 2017.') , 'NoOfCloudyDays'] = 19
df.loc[(df['CalculationPeriod'] == 'december 2017.') , 'InsolationInHours'] =71.
 ↪1

#------------------------------------------------------------------------

df.loc[(df['CalculationPeriod'] == 'january 2018.') , 'NoOfClearDays'] = 6
df.loc[(df['CalculationPeriod'] == 'january 2018.') , 'NoOfCloudyDays'] = 11
```

```
df.loc[(df['CalculationPeriod'] == 'january 2018.') , 'InsolationInHours'] =112.
↪8

df.loc[(df['CalculationPeriod'] == 'february 2018.') , 'NoOfClearDays'] = 0
df.loc[(df['CalculationPeriod'] == 'february 2018.') , 'NoOfCloudyDays'] = 17
df.loc[(df['CalculationPeriod'] == 'february 2018.') , 'InsolationInHours'] =58.
↪9

df.loc[(df['CalculationPeriod'] == 'march 2018.'), 'NoOfClearDays'] = 0
df.loc[(df['CalculationPeriod'] == 'march 2018.') , 'NoOfCloudyDays'] =18
df.loc[(df['CalculationPeriod'] == 'march 2018.') , 'InsolationInHours'] =102.3

df.loc[(df['CalculationPeriod'] == 'april 2018.') , 'NoOfClearDays'] = 9
df.loc[(df['CalculationPeriod'] == 'april 2018.') , 'NoOfCloudyDays'] = 6
df.loc[(df['CalculationPeriod'] == 'april 2018.') , 'InsolationInHours'] =218

df.loc[(df['CalculationPeriod'] == 'may 2018.') , 'NoOfClearDays'] = 5
df.loc[(df['CalculationPeriod'] == 'may 2018.') , 'NoOfCloudyDays'] = 4
df.loc[(df['CalculationPeriod'] == 'may 2018.') , 'InsolationInHours'] = 275.1

df.loc[(df['CalculationPeriod'] == 'june 2018.') , 'NoOfClearDays'] = 0
df.loc[(df['CalculationPeriod'] == 'june 2018.') , 'NoOfCloudyDays'] = 11
df.loc[(df['CalculationPeriod'] == 'june 2018.') , 'InsolationInHours'] =165.4

df.loc[(df['CalculationPeriod'] == 'july 2018.') , 'NoOfClearDays'] = 3
df.loc[(df['CalculationPeriod'] == 'july 2018.') , 'NoOfCloudyDays'] = 8
df.loc[(df['CalculationPeriod'] == 'july 2018.') , 'InsolationInHours'] =217.9

df.loc[(df['CalculationPeriod'] == 'august 2018.') , 'NoOfClearDays'] =14
df.loc[(df['CalculationPeriod'] == 'august 2018.') , 'NoOfCloudyDays'] =4
df.loc[(df['CalculationPeriod'] == 'august 2018.') , 'InsolationInHours'] =280.4
```

[ ]:
```
dictionary={'january 2014.':'4',
'february 2014.':'5.6',
'march 2014.':'7.1',
'april 2014.':'9.2',
'may 2014.':'12.3',
'june 2014.':'16.9',
'july 2014.':'18.9',
'august 2014.':'18.7',
'september 2014.':'14.1',
'october 2014.':'10.7',
'november 2014.':'7.6',
'december 2014.':'1.6',
'january 2015.':'1.4',
'february 2015.':'0.7',
'march 2015.':'3.4',
```

```
'april 2015.':'8.6',
'may 2015.':'15.4',
'june 2015.':'17.2',
'july 2015.':'22.3',
'august 2015.':'22.1',
'september 2015.':'17.1',
'october 2015.':'9.7',
'november 2015.':'8.1',
'december 2015.':'4.1',
'january 2016.':'0.9',
'february 2016.':'6.7',
'march 2016.':'4.8',
'april 2016.':'12.3',
'may 2016.':'12.8',
'june 2016.':'18.1',
'july 2016.':'20',
'august 2016.':'17.8',
'september 2016.':'15.9',
'october 2016.':'8.3',
'november 2016.':'5.5',
'december 2016.':'0.2',
'january 2017.':'-5.2',
'february 2017.':'3.6',
'march 2017.':'8.2',
'april 2017.':'8.2',
'may 2017.':'14.2',
'june 2017.':'19.9',
'july 2017.':'21.4',
'august 2017.':'22.5',
'september 2017.':'14.9',
'october 2017.':'11.2',
'november 2017.':'5.2',
'december 2017.':'1.9',
'january 2018.':'2.8',
'february 2018.':'-1',
'march 2018.':'3.8',
'april 2018.':'14.5',
'may 2018.':'16.6',
'june 2018.':'17.3',
'july 2018.':'18.6',
'august 2018.':'20.8',
           }
           }
df['AVG_TEMPERATURE'] = df['CalculationPeriod'].map(dictionary)
def enkoder(kolona):
    Meann = df.groupby([kolona])['ConsumptionInkwh'].mean().to_dict()
    return Meann
```

```
df['AVGconsumption'] = df['CONSUMER'].map(enkoder('CONSUMER'))
# df.to_csv("sa vremenskim kolonama.csv",index=False)
print("OK")


letnjiMeseci=['april 2014.', 'april 2015.', 'april 2016.', 'april 2017.',␣
 ↪'april 2018.','may 2014.', 'may 2015.', 'may 2016.', 'may 2017.', 'may 2018.
 ↪',
              'june 2014.', 'june 2015.', 'june 2016.', 'june 2017.', 'june␣
 ↪2018.','july 2014.', 'july 2015.', 'july 2016.', 'july 2017.', 'july 2018.',
              'august 2014.', 'august 2015.', 'august 2016.', 'august 2017.',␣
 ↪'august 2018.','september 2014.', 'september 2015.', 'september 2016.',␣
 ↪'september 2017.',
              'october 2014.', 'october 2015.', 'october 2016.', 'october 2017.
 ↪']
zimskiMeseci=['january 2014.', 'january 2015.', 'january 2016.', 'january 2017.
 ↪', 'january 2018.','february 2014.', 'february 2015.', 'february 2016.',␣
 ↪'february 2017.', 'february 2018.',
              'march 2014.', 'march 2015.', 'march 2016.', 'march 2017.',␣
 ↪'march 2018.','november 2014.', 'november 2015.', 'november 2016.',␣
 ↪'november 2017.',
              'december 2014.', 'december 2015.', 'december 2016.', 'december␣
 ↪2017.']
df.loc[(df['CalculationPeriod'].isin(letnjiMeseci)), 'Summer/WinterTime'] =␣
 ↪'Daylight saving time'
df.loc[(df['CalculationPeriod'].isin(zimskiMeseci)), 'Summer/WinterTime'] =␣
 ↪'Standard time'

kolone=['CONSUMER', 'CONSUMER_CATEGORY', 'ZONE', 'PERIOD_OD', 'PERIOD_DO',
       'CONSUMER_GROUP', 'STANJE_MT', 'PRETHODNO_STANJE_MT', 'STANJE_VT',
       'PRETHODNO_STANJE_VT', 'NISKApot', 'VISOKApot',
       'NOdays', 'Year', 'Month', 'CalculationPeriod', 'Seasons', 'timeidx',
       'Trends',  'MAXtemp', 'MINtemp', 'NoOfClearDays','Summer/WinterTime',
       'NoOfCloudyDays', 'InsolationInHours', 'DaylightHours', 'SunshineHours',
       'AVG_TEMPERATURE','razlikaOdMAX', 'razlikaOdMIN', 'AVGconsumption',
       'ConsumptionZone1MB','ConsumptionZone',
        '1monthBefore', '2monthBefore', '3monthBefore',  'ConsumptionInkwh']
df = df.reindex(kolone, axis=1)
decimals = pd.Series([3,3,0,0,0,3,0,0,3,0],␣
 ↪index=['scores','Trends','1monthBefore','2monthBefore',
                                                    ␣
 ↪'3monthBefore','AVGconsumption','razlikaOdMAX',
                                                    ␣
 ↪'razlikaOdMIN','3monthAvg','ConsumptionInkwh'])
```

```
df.round(decimals).to_csv('dopunjen set.csv',index=False)
```

[ ]:
```
df.drop(index=df[df['CONSUMER_GROUP'] =='Low voltage'].index, inplace=True)
df.drop(index=df[df['CONSUMER_GROUP'] =='Medium voltage 10KV'].index,␣
 ↪inplace=True)  df['ConsumptionInkwh'] = df['ConsumptionInkwh'].
 ↪astype('float64')
df.drop(['Year','PRETHODNO_STANJE_MT', 'STANJE_VT','STANJE_MT',␣
 ↪'PRETHODNO_STANJE_VT',
        'potVT', 'potMT'], axis='columns', inplace=True)
print(df.info())
#KREIRANJE NOVOG DF OD KATEGORICKIH KOLONA
kategorickeKolone = df.select_dtypes(include=['object']).copy()
print('\n\nkategoricke\n',kategorickeKolone.columns)
numericke=df.select_dtypes(include=['int64','float64']).copy()
print('\n\nnumericke\n',numericke.columns)
X=df.select_dtypes(include=['int64','float64']).copy()
y=df.loc[:,'ConsumptionInkwh']
iforest = IsolationForest(n_estimators=100, max_samples='auto',
                          contamination='auto', #max_features=2,
                          bootstrap=True, n_jobs=-1, random_state=5, verbose=1)
pred= iforest.fit_predict(X)
df['scores']=iforest.decision_function(X)
df[['scores']] = df[['scores']].round(3)
df['anomaly_label']=pred
df['anomaly']=df['anomaly_label'].apply(lambda x: 'outlier' if x==-1  else␣
 ↪'inlier')
fig=px.histogram(df,x='scores',color='anomaly')
fig.show()
df.drop(index=df[df['anomaly_label'] ==-1].index, inplace=True)
df.drop(['anomaly_label', 'anomaly'], axis='columns', inplace=True)
```

[ ]:
```
grupisanoPoPotrosacu=df.groupby('CONSUMER')
def brojanje(POTROSAC):
    broj = grupisanoPoPotrosacu['CONSUMER'].count()
    return broj

df.drop(['Trends' ], axis='columns', inplace=True)
df['timeidx'] = df.groupby(['CONSUMER']).cumcount()
def linear_model(dfrejm):
    y = dfrejm[['ConsumptionInkwh']].values
    X = dfrejm[['timeidx']].values
    return np.squeeze(LinearRegression().fit(X, y).coef_)

trends_by_group = df.groupby('CONSUMER').apply(linear_model).
 ↪to_frame(name='Trends').reset_index()
rezultat = df.merge(trends_by_group, how='left', on='CONSUMER')
df=rezultat
```

[ ]:
```
def MIN(POTROSAC):
    razlikaMIN = df.groupby([POTROSAC])['ConsumptionInkwh'].min().to_dict()
    return razlikaMIN

def MAX(POTROSAC):
    razlikaMAX = df.groupby([POTROSAC])['ConsumptionInkwh'].max().to_dict()
    return razlikaMAX

df['razlikaOdMAX'] = df['CONSUMER'].map(MAX('CONSUMER'))
df['razlikaOdMIN'] = df['CONSUMER'].map(MIN('CONSUMER'))

df['razlikaOdMAX'] = df['razlikaOdMAX']-df['ConsumptionInkwh']
df['razlikaOdMIN'] =df['ConsumptionInkwh']-df['razlikaOdMIN']

def enkoder(kolona):
    Meann = df.groupby([kolona])['ConsumptionInkwh'].mean().to_dict()
    return Meann
df['AVGconsumption'] = df['CONSUMER'].map(enkoder('CONSUMER'))  # OVA KOLONA␣
 ↪DAJE ISTE REZULTATE KAO PRVA KOLONA CONSUMER
df[['Trends', 'scores','AVGconsumption']] = df[['Trends', 'scores',␣
 ↪'AVGconsumption']].round(3)
```

[ ]:
```
df['indexLag'] = df.groupby(['CONSUMER']).cumcount()
def enkoder(potrosac):
    Meann = df.groupby([potrosac])['ConsumptionInkwh'].mean().to_dict()
    #print(Meann)
    return Meann

df['kolonaMEANpoPOT'] = df['CONSUMER'].map(enkoder('CONSUMER'))
# print(df.head())
# Lagg = df.groupby(['CONSUMER'])['ConsumptionInkwh'].mean().to_dict()

df['1monthBefore'] = df.groupby(['CONSUMER'])['ConsumptionInkwh'].shift(1)
df['2monthBefore'] = df.groupby(['CONSUMER'])['ConsumptionInkwh'].shift(2)
df['3monthBefore'] = df.groupby(['CONSUMER'])['ConsumptionInkwh'].shift(3)

df=df.fillna(0)
df.loc[df['1monthBefore'] == 0, "1monthBefore"] =df['kolonaMEANpoPOT']
df.loc[df['2monthBefore'] == 0, "2monthBefore"] =df['kolonaMEANpoPOT']
df.loc[df['3monthBefore'] == 0, "3monthBefore"] =df['kolonaMEANpoPOT']

df.drop(['kolonaMEANpoPOT'], axis=1, inplace=True)
df.drop(['indexLag'], axis=1, inplace=True)

df.loc[df['1monthBefore'] <=350, "ConsumptionZone1MB"] ='GREEN ZONE'
df.loc[(df['1monthBefore'] >350) & (df['1monthBefore'] <=1600) ,␣
 ↪'ConsumptionZone1MB'] = 'BLUE ZONE'
```

```
df.loc[df['1monthBefore'] >1600, "ConsumptionZone1MB"] ='RED ZONE'

df.loc[df['ConsumptionInkwh'] <=350, "ConsumptionZone"] ='GREEN ZONE'
df.loc[(df['ConsumptionInkwh'] >350) & (df['ConsumptionInkwh'] <=1600) ,␣
 ↪'ConsumptionZone'] = 'BLUE ZONE'
df.loc[df['ConsumptionInkwh'] >1600, "ConsumptionZone"] ='RED ZONE'
```

ПРИЛОГ 2 - Метеоролошки подаци за 2014. годину

РЦ УЖИЦЕ ширина 43 ° 53’ дужина 19° 50’ висина 833m 2014

Месец	Ваздушни притисак (mb)				Температура ваздуха (° C)								Екстреми			
	7	14	21	ср	мах	мин	амп	мин 5 cm	7	14	21	ср	мах	дан	мин	дан
1	917,4	917,2	917,9	917,5	8,2	0,9	7,2	0,3	2,6	6,6	3,5	4,0	13,8	19	-10,4	27
2	918,7	918,2	918,7	918,6	10,2	1,4	8,8	0,1	3,4	8,7	5,1	5,6	19,5	19	-9,0	3
3	919,2	918,8	919,3	919,1	12,2	2,6	9,6	1,7	4,4	10,4	6,8	7,1	20,3	18	-2,4	11
4	916,3	916,4	916,4	916,4	14,3	5,3	9,0	4,6	7,4	11,7	8,9	9,2	21,3	4	-0,7	17
5	918,4	918,4	918,8	918,5	17,5	7,9	9,6	6,4	10,7	15,0	11,7	12,3	25,4	23	0,5	15
6	921,5	921,5	921,8	921,6	21,5	13,0	8,6	10,7	15,5	19,7	16,1	16,9	28,0	10	7,0	1
7	919,9	919,6	920,0	919,8	23,7	14,3	9,4	12,3	17,4	22,2	18,1	18,9	28,4	21	8,0	2
8	921,1	920,8	921,2	921,0	23,7	14,3	9,4	12,4	16,6	22,3	17,8	18,7	30,4	14	8,6	29
9	921,4	921,9	922,2	921,8	17,9	10,6	7,3	9,1	12,7	16,5	13,6	14,1	23,4	19	3,7	24
10	923,4	923,7	924,1	923,7	14,8	7,2	7,6	5,1	8,9	13,4	10,2	10,7	25,2	13	-1,8	26
11	921,5	921,3	922,0	921,6	11,2	4,4	6,8	2,0	5,9	10,1	7,2	7,6	18,2	5	-2,6	26
12	921,4	921,4	921,8	921,5	5,0	-0,9	5,9	-1,6	0,4	3,3	1,3	1,6	15,0	24	-13,5	31
год	920,0	919,9	920,3	920,1	15,0	6,8	8,3	5,3	8,8	13,4	10,0	10,6	30,4	8	-13,5	12

Месец	Напон водене паре (mb)				Релативна влажност (%)				Ветар (m/s)			Инсолација	Облачност у десетинама				Падавине (mm)			Снег (cm)	
	7	14	21	ср	7	14	21	ср	мин	ср	>6Б	>8Б	(h)	7	14	21	ср	сума	мах	дан	У	Н
1	6,3	7,1	6,4	6,6	82	72	80	78	46	1,5	2	0	94,1	7,5	6,6	5,6	6,6	19,1	4,8	25	6	3
2	6,4	7,0	6,6	6,7	80	64	76	73	30	2,1	3	0	115,1	6,9	7,1	3,9	6,0	12,3	4,6	15	3	2
3	6,6	7,5	7,0	7,0	80	63	73	72	29	2,0	5	1	180,6	5,8	6,0	5,8	5,9	83,3	15,2	6	3	2
4	8,8	9,6	9,5	9,3	84	72	82	79	31	1,8	3	0	109,8	8,1	8,5	6,7	7,8	155,5	28,7	25	21	4
5	10,5	11,4	10,9	10,9	80	68	79	76	37	2,3	6	2	189,7	6,0	6,9	6,1	6,3	218,0	65,3	15	-	0
6	13,1	14,5	13,2	13,6	75	65	72	71	43	1,6	2	1	225,2	4,6	6,7	5,2	5,5	124,7	27,4	26	-	-
7	15,1	15,9	14,9	15,3	76	59	72	69	43	2,0	1	1	278,4	3,9	6,2	5,5	5,2	172,9	80,1	26	-	-
8	14,6	16,9	15,3	15,6	78	63	75	72	38	1,4	1	0	268,6	3,8	5,4	4,4	4,5	118,8	46,5	1	-	-
9	12,6	14,0	13,6	13,4	85	74	85	82	45	1,7	0	0	125,0	6,7	7,4	5,5	6,5	158,8	29,5	5	-	-
10	9,3	11,3	10,4	10,4	81	73	81	79	39	1,3	2	0	137,7	6,2	6,3	5,7	6,1	46,1	19,3	23	1	1
11	7,9	8,8	8,3	8,3	83	71	80	78	43	1,3	3	0	94,6	7,0	6,5	5,6	6,3	16,2	3,4	20	-	0
12	5,5	5,8	5,7	5,7	85	76	83	81	29	2,0	3	0	66,4	6,6	7,4	6,1	6,7	91,6	36,4	2	6	4
год	9,7	10,8	10,2	10,2	81	68	78	76	29	1,8	31	5	1885,2	6,1	6,7	5,5	6,1	1217,3	80,1	7	5	0

Месец	Број дана са													Појава							
	Тн ≤ -10	Тх <0	Тн <0	Тх ≥ 25	Тх ≥ 30	Тн ≥ 20	Облачност < 2	Облачност > 8	Падавине ≥ 0.1	Падавине ≥ 1	Падавине ≥ 10	К	Сн	Су	Кр	По	С	Г	Грм	≡	Сп
1	1	3	8	0	0	0	3	14	8	6	0	5	3	0	0	0	0	0	0	4	7
2	0	2	9	0	0	0	3	7	7	4	0	9	1	2	0	0	0	0	0	3	5
3	0	0	6	0	0	0	7	11	16	15	2	10	3	1	0	0	0	0	1	11	6
4	0	0	1	0	0	0	0	12	19	15	7	18	4	2	1	0	0	0	3	9	2
5	0	0	0	2	0	0	2	14	19	16	7	19	1	0	0	0	1	0	6	6	0
6	0	0	0	8	0	0	5	8	15	14	4	17	0	0	0	0	1	1	2	4	0
7	0	0	0	10	0	0	2	4	16	12	4	20	0	0	0	0	0	1	6	1	0
8	0	0	0	12	1	2	7	3	14	11	3	15	0	0	0	0	0	0	9	3	0
9	0	0	0	0	0	0	3	11	20	16	6	22	0	0	0	0	0	0	5	9	0
10	0	0	5	1	0	0	8	13	14	6	2	9	2	1	0	0	0	0	0	10	1
11	0	0	5	0	0	0	5	13	10	5	0	9	1	0	0	1	0	0	1	10	0
12	3	5	16	0	0	0	7	16	15	14	1	6	7	3	0	0	0	0	2	13	12
год	4	10	50	33	1	2	52	126	173	134	36	159	22	9	1	1	2	2	35	83	33

Месец	Честине праваца и средње брзине ветра (m/s)																
	N		NE		E		SE		S		SW		W		NW		C
	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	тихо
1	6	2,2	6	1,0	8	0,9	2	1,0	13	2,1	26	2,2	14	1,6	10	2,3	8
2	4	1,8	12	1,0	5	2,0	3	2,3	14	3,5	23	3,5	10	2,7	3	1,3	10
3	16	3,2	14	1,9	4	2,3	5	1,8	3	1,0	17	3,5	15	2,5	5	2,0	14
4	17	1,9	5	0,8	9	2,0	2	1,5	5	1,6	9	1,6	12	1,9	25	2,8	6
5	14	2,5	7	2,0	4	1,5	6	1,8	4	1,5	13	1,4	16	2,6	20	5,3	9
6	23	2,1	9	2,0	5	1,6	2	1,5	3	3,3	4	0,8	10	2,7	16	2,7	18
7	13	2,5	11	1,2	8	1,0	6	0,8	5	2,2	7	3,0	22	2,8	15	3,3	6
8	18	1,9	11	1,6	11	1,4	4	0,8	4	3,3	8	1,6	13	2,6	8	2,1	16
9	14	1,9	21	2,0	3	1,7	6	1,3	2	2,5	9	2,0	9	2,3	15	3,0	11
10	6	2,0	13	1,2	9	1,2	6	1,5	7	1,3	11	1,3	15	2,2	12	1,9	14
11	5	1,8	13	1,5	7	1,1	7	1,3	4	3,0	18	2,3	8	1,4	9	2,0	19
12	11	2,6	7	2,0	3	3,3	0	0,0	2	1,5	13	1,8	15	4,2	14	4,1	28
год	146	2,2	133	1,6	73	1,5	52	1,4	63	2,4	159	2,3	157	2,6	153	3,0	159

ПРИЛОГ 3 - Метеоролошки подаци за 2015. годину

РЦ УЖИЦЕ ширина 43 º 53’ дужина 19º 50’ висина 833m 2015

Месец	Ваздушни притисак (mb)				Температура ваздуха (° C)			мин					Екстреми			
	7	14	21	ср	мах	мин	амп	5 cm	7	14	21	ср	мах	дан	мин	дан
1	919,9	919,6	919,5	919,7	4,6	-2,0	6,6	-2,6	-0,1	3,2	1,2	1,4	13,0	11	-13,5	1
2	916,8	916,9	918,2	917,3	4,3	-2,4	6,6	-3,6	-1,1	3,1	0,5	0,7	13,0	24	-10,0	18
3	922,0	921,7	921,7	921,8	7,4	-0,1	7,5	-1,2	1,3	5,8	3,3	3,4	18,0	26	-4,0	8
4	922,2	921,7	922,3	922,1	14,0	3,9	10,1	1,8	5,9	11,9	8,2	8,6	23,0	16	-4,5	6
5	920,8	920,8	920,9	920,9	20,4	10,9	9,4	6,7	13,1	18,6	14,9	15,4	30,0	6	4,9	28
6	923,5	923,3	923,5	923,5	22,1	12,6	9,5	9,7	15,4	20,4	16,5	17,2	29,0	13	6,5	25
7	923,4	923,0	923,0	923,1	27,7	17,3	10,4	14,4	20,1	26,0	21,6	22,3	32,6	19	11,8	10
8	922,9	923,0	923,4	923,1	27,7	17,2	10,5	14,6	19,4	26,4	21,3	22,1	33,0	31	11,5	22
9	922,6	922,4	923,0	922,7	22,3	13,1	9,1	11,5	14,9	20,6	16,3	17,1	35,0	18	6,4	29
10	923,1	923,0	923,5	923,2	13,7	6,6	7,1	5,6	7,8	12,4	9,2	9,7	23,0	4	1,1	21
11	923,5	923,2	923,8	923,5	-	4,1	-	1,5	5,7	11,3	7,7	8,1	-	-	-3,0	28
12	932,5	932,0	932,6	932,4	-	0,8	-	-0,8	2,2	7,3	3,5	4,1	-	-	-8,6	31
год	922,8	922,6	923,0	922,8	-	6,9	-	4,9	8,8	14,0	10,4	10,9	-	-	-13,5	1

Месец	Напон водене паре (mb)				Релативна влажност (%)				Ветар (m/s)			Инсо-лација	Облачност у десетинама				Падавине (mm)			Снег (cm)	
	7	14	21	ср	7	14	21	ср	мин	ср	>6Б	>8Б	(h)	7	14	21	ср	сума	мах	дан	У	Н
1	5,2	5,7	5,2	5,3	83	74	78	78	35	3,1	7	0	80,8	7,6	7,3	6,6	7,1	39,8	8,6	6	10	4
2	5,0	5,6	5,1	5,2	85	75	79	80	28	1,7	4	0	101,4	6,6	6,6	5,4	6,2	62,1	13,4	7	13	4
3	5,3	5,9	5,5	5,6	80	67	72	73	34	2,9	8	0	108,1	7,8	7,4	5,9	7,1	102,2	31,7	6	26	5
4	6,7	6,8	6,6	6,7	73	50	61	61	26	3,4	7	0	210,6	4,3	6,3	4,0	4,9	52,8	18,6	6	7	4
5	11,0	12,0	11,1	11,3	73	57	66	65	32	2,0	3	0	206,2	5,7	6,9	5,3	6,0	52,0	18,2	23	-	-
6	12,7	13,7	12,8	13,1	73	58	69	67	34	2,5	4	0	242,8	4,5	6,1	5,6	5,4	101,7	35,1	18	-	-
7	15,9	16,1	15,1	15,7	67	49	58	58	28	1,2	1	0	325,3	2,0	4,0	2,3	2,8	11,6	10,1	14	-	-
8	14,8	15,5	14,2	14,8	68	47	59	58	28	1,5	0	0	269,3	3,5	4,7	2,7	3,7	37,1	16,7	21	-	-
9	11,5	12,4	12,6	12,2	71	55	69	65	21	1,9	1	0	185,2	5,5	6,1	6,0	5,9	91,9	39,2	25	-	-
10	9,4	10,4	9,8	9,9	88	73	84	81	31	1,5	2	0	123,7	7,7	6,5	5,9	6,7	74,3	22,4	11	-	-
11	6,8	7,9	7,2	7,3	75	61	70	69	29	1,8	3	0	154,1	4,7	4,6	3,3	4,2	50,5	21,8	22	8	3
12	5,7	6,5	6,1	6,1	80	66	77	74	27	1,2	0	0	163,0	4,0	3,9	3,5	3,8	4,5	3,1	20	2	1
год	9,2	9,9	9,3	9,5	76	61	70	69	21	2,1	40	0	2170,5	5,3	5,9	4,7	5,3	680,5	39,2	9	13	0

Месец	Број дана са																				
	Тн ≤ -10	Тх <0	Тн <0	Тх ≥ 25	Тх ≥ 30	Тн ≥ 20	Облачност < 2	Облачност > 8	Падавине ≥ 0.1	Падавине ≥ 1	Падавине ≥ 10	К	Сн	Појава Су	Појава Кр	Појава По	Појава С	Појава Г	Појава Грм	Појава ≡	Сп
1	2	9	19	0	0	0	3	17	18	11	0	12	11	1	0	0	0	0	0	5	21
2	1	5	21	0	0	0	6	12	16	12	1	5	11	0	0	0	0	0	0	12	21
3	0	2	16	0	0	0	3	16	16	10	5	5	11	0	0	0	0	0	0	6	16
4	0	0	7	0	0	0	6	6	13	9	2	7	6	1	0	0	0	0	0	1	8
5	0	0	0	6	1	0	4	10	11	8	2	11	0	0	0	0	0	0	4	1	0
6	0	0	0	10	0	0	4	7	14	13	4	14	0	0	0	0	0	1	3	2	0
7	0	0	0	23	11	7	11	2	3	1	1	5	0	0	0	0	0	0	5	1	0
8	0	0	0	25	13	8	12	4	9	6	1	9	0	0	0	0	0	0	3	4	0
9	0	0	0	12	8	5	5	12	10	6	3	10	0	0	0	0	0	1	2	5	0
10	0	0	0	0	0	0	0	12	14	8	4	12	0	0	0	0	0	0	0	14	0
11	0	-	10	-	-	0	12	7	6	5	2	1	3	2	0	0	0	0	0	5	7
12	0	-	11	-	-	0	15	7	3	1	0	1	1	0	0	0	0	0	0	8	3
год	3	-	84	-	-	20	81	112	133	90	25	92	43	4	0	0	0	2	17	64	76

Месец	Честине праваца и средње брзине ветра (m/s)																	
	N		NE		E		SE		S		SW		W		NW		C	
	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	тихо	
1	7	2,4	4	1,3	6	0,8	5	8,4	6	3,7	24	3,8	16	2,6	17	4,8	8	
2	11	2,9	15	1,6	4	3,0	5	2,2	3	2,7	7	3,2	9	2,8	5	4,8	25	
3	12	3,3	18	2,6	9	2,4	3	1,7	4	2,3	10	4,4	12	4,3	14	4,9	11	
4	9	3,9	5	1,8	1	2,0	1	2,0	4	5,5	20	3,8	17	3,6	30	3,8	3	
5	9	2,9	10	2,2	10	1,7	6	1,3	6	1,7	9	2,0	15	2,8	19	3,1	9	
6	16	3,3	11	2,5	10	2,2	5	1,4	3	3,7	4	3,5	18	2,6	15	3,9	8	
7	14	1,4	14	1,9	8	1,5	4	1,5	4	1,3	5	1,2	6	3,0	18	1,8	20	
8	13	1,8	19	1,8	10	1,8	9	2,0	3	3,3	8	2,4	9	2,4	7	1,6	15	
9	21	1,9	9	1,7	5	2,0	4	1,3	5	4,8	10	2,7	14	2,4	14	2,7	8	
10	9	2,2	30	1,6	7	1,9	9	1,3	2	1,0	7	1,1	9	2,2	12	2,0	8	
11	7	1,6	12	0,9	3	1,0	1	0,0	4	1,3	16	3,3	19	3,0	15	2,3	13	
12	6	1,0	3	2,0	2	1,0	4	0,8	2	2,0	15	2,1	21	2,7	12	2,0	28	
год	134	2,4	150	1,8	75	1,9	56	2,1	46	2,8	135	3,0	165	2,9	178	3,1	156	

ПРИЛОГ 4 - Метеоролошки подаци за 2016. годину

РЦ УЖИЦЕ ширина 43 º 53’ дужина 19º 50’ висина 822m 2016

Месец	Ваздушни притисак (mb)				Температура ваздуха (° C)								Екстреми			
	7	14	21	ср	мах	мин	амп	мин 5 cm	7	14	21	ср	мах	дан	мин	дан
1	918,3	918,3	919,1	918,6	4,3	-2,5	6,8	-3,5	-1,0	2,9	0,8	0,9	15,0	27	-15,0	4
2	917,9	917,9	918,6	918,1	11,9	2,4	9,4	2,0	4,4	9,4	6,4	6,7	19,5	15	-3,6	6
3	915,8	916,1	916,6	916,1	8,4	1,4	7,0	0,2	2,8	7,2	4,6	4,8	22,1	31	-3,0	15
4	917,6	917,2	917,6	917,5	17,9	7,3	10,6	5,4	9,8	16,1	11,7	12,3	26,2	18	-1,2	26
5	918,4	918,4	918,6	918,5	17,4	8,5	8,9	6,1	11,1	15,7	12,2	12,8	27,3	29	1,0	16
6	920,8	920,3	920,9	920,7	23,2	13,9	9,3	11,9	16,6	21,5	17,2	18,1	29,2	17	9,6	2
7	922,7	922,9	923,1	922,9	25,1	15,6	9,5	13,3	18,3	23,1	19,2	20,0	31,1	13	9,7	17
8	924,4	924,6	925,1	924,7	22,7	13,3	9,3	10,8	15,9	20,9	17,2	17,8	28,1	5	9,0	12
9	923,7	923,6	924,1	923,8	20,4	11,9	8,5	8,2	13,8	19,2	15,2	15,9	26,2	16	6,6	23
10	923,8	923,9	924,4	924,0	12,2	5,4	6,8	-	7,1	10,5	7,8	8,3	22,6	2	0,4	6
11	921,9	922,1	922,1	922,0	10,7	1,8	8,9	0,5	3,8	8,0	5,2	5,5	18,4	6	-6,9	30
12	929,9	929,7	930,1	929,9	4,3	-3,4	7,8	-5,6	-1,7	2,7	-0,2	0,2	16,1	11	-9,4	31
год	921,3	921,3	921,7	921,4	14,9	6,3	8,6	-	8,4	13,1	9,8	10,3	31,1	7	-15,0	1

Месец	Напон водене паре (mb)				Релативна влажност (%)				Ветар (m/s)			Инсо-лација	Облачност у десетинама				Падавине (mm)			Снег (cm)	
	7	14	21	ср	7	14	21	ср	мин	ср	>6Б	>8Б	(h)	7	14	21	ср	сума	мах	дан	У	Н
1	5,0	5,9	5,4	5,4	81	77	80	79	42	2,7	4	0	80,5	7,7	7,6	5,6	7,0	58,7	19,2	4	11	4
2	6,4	7,4	6,6	6,8	77	64	70	70	34	3,2	6	1	91,8	6,1	7,6	5,1	6,2	32,4	9,5	11	3	1
3	6,2	6,8	6,6	6,6	83	70	79	77	33	1,9	2	0	106,8	6,5	8,1	5,6	6,8	180,0	59,6	7	7	4
4	7,7	9,1	8,0	8,3	66	53	62	60	22	1,9	6	0	190,3	5,5	6,6	4,4	5,5	45,7	9,5	15	2	1
5	9,8	10,9	10,3	10,3	75	63	73	70	34	2,1	7	0	192,2	5,8	7,1	4,7	5,8	150,8	27,7	3	-	-
6	14,4	16,0	14,5	15,0	76	62	74	71	30	1,6	1	0	225,7	5,4	6,5	6,2	6,0	174,8	54,8	20	-	-
7	15,6	16,4	15,2	15,7	74	59	69	67	34	1,3	1	0	280,2	3,7	4,7	2,7	3,7	68,8	43,0	16	-	-
8	14,4	16,0	14,7	15,0	80	66	75	74	35	1,3	2	0	235,8	4,4	4,9	3,4	4,2	146,0	35,3	8	-	-
9	12,1	13,5	12,5	12,7	76	60	72	69	37	1,4	0	0	230,9	4,0	5,6	4,0	4,5	48,7	19,7	6	-	-
10	8,6	9,6	9,2	9,1	85	76	86	82	41	1,2	2	0	106,5	8,3	7,7	6,4	7,5	104,4	19,2	8	-	-
11	6,7	7,8	7,1	7,2	82	72	79	77	47	1,8	2	0	128,2	6,1	6,4	5,9	6,2	93,2	29,0	8	5	2
12	4,2	4,7	4,4	4,4	78	66	73	73	25	2,1	0	0	131,4	5,1	4,2	3,9	4,4	10,1	3,8	27	4	1
год	9,3	10,3	9,6	9,7	78	66	74	73	22	1,9	33	1	2000,3	5,7	6,4	4,8	5,6	1113,6	59,6	3	8	0

Месец	Број дана са													Правац							
	Тн ≤ -10	Тх <0	Тн <0	Тх ≥ 25	Тх ≥ 30	Тн ≥ 20	Облачност < 2	Облачност > 8	Падавине ≥ 0.1	Падавине ≥ 1	Падавине ≥ 10	К	Сн	Су	Кр	По	С	Г	Грм	≡	Сп
1	5	11	18	0	0	0	2	14	16	11	2	7	11	0	0	0	0	0	0	5	18
2	0	1	10	0	0	0	1	7	11	9	0	7	7	0	0	0	0	0	0	9	5
3	0	0	10	0	0	0	4	14	16	14	7	14	6	2	0	0	0	0	1	11	6
4	0	0	2	2	0	0	7	10	8	8	0	11	1	0	0	0	0	1	1	7	2
5	0	0	0	4	0	0	4	8	19	17	5	19	0	0	0	0	0	0	6	5	0
6	0	0	0	10	0	2	2	5	16	14	5	18	0	0	0	0	0	0	10	3	0
7	0	0	0	20	3	3	10	3	8	5	2	9	0	0	0	0	0	0	4	1	0
8	0	0	0	6	0	0	8	6	11	11	6	11	0	0	0	0	0	0	4	8	0
9	0	0	0	3	0	0	5	6	9	4	2	10	0	0	0	0	0	0	2	5	0
10	0	0	0	0	0	0	0	15	18	13	4	17	0	0	0	0	0	0	2	11	0
11	0	2	11	0	0	0	5	14	10	9	3	9	5	1	0	0	0	0	0	4	3
12	0	6	24	0	0	0	12	7	6	5	0	2	6	1	0	0	0	0	0	4	5
год	5	20	75	45	3	5	60	109	148	120	36	134	36	4	0	0	0	1	30	73	39

Месец	Честине праваца и средње брзине ветра (m/s)																	
	N		NE		E		SE		S		SW		W		NW		C	
	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	тихо	
1	6	2,3	3	1,7	2	1,5	2	1,0	4	1,5	27	2,7	28	3,4	20	2,9	1	
2	4	2,0	6	1,7	5	2,4	2	2,0	11	4,7	30	5,5	8	4,3	7	3,6	14	
3	19	1,9	10	1,4	6	1,5	2	3,5	7	2,3	15	1,9	14	2,7	13	2,9	7	
4	8	2,0	3	1,3	5	1,0	7	1,3	11	2,4	18	2,9	18	2,2	11	3,0	9	
5	8	2,3	7	1,4	10	1,0	4	0,8	6	2,8	20	2,8	14	2,4	18	3,2	6	
6	8	1,3	18	1,6	6	1,0	4	1,5	12	2,4	9	3,0	11	2,0	11	3,2	11	
7	14	1,9	14	1,5	9	1,0	3	0,7	4	1,3	5	1,2	12	1,3	17	2,7	15	
8	13	2,3	14	0,9	8	0,9	5	1,4	0	0,0	8	1,5	10	1,3	19	2,5	16	
9	15	1,8	19	1,5	8	2,8	4	1,8	0	0,0	7	2,1	10	2,1	11	2,4	16	
10	9	1,4	11	1,0	2	1,0	4	0,8	3	1,0	7	1,1	13	2,3	23	2,0	21	
11	2	1,0	6	0,8	1	1,0	2	1,5	4	1,5	32	2,6	10	3,0	18	2,5	15	
12	3	5,0	5	1,4	1	1,0	1	2,0	3	5,3	12	2,2	27	3,1	20	2,9	21	
год	109	2,0	116	1,4	63	1,3	40	1,5	65	2,7	190	2,9	175	2,6	188	2,7	152	

ПРИЛОГ 5 - Метеоролошки подаци за 2017. годину

РЦ УЖИЦЕ ширина 43 ° 53’ дужина 19° 50’ висина 822m 2017

Месец	Ваздушни притисак (mb)				Температура ваздуха (° C)								Екстреми			
	7	14	21	ср	мах	мин	амп	мин 5 cm	7	14	21	ср	мах	дан	мин	дан
1	921,7	921,7	922,1	921,8	-1,5	-8,8	7,3	-10,0	-7,1	-3,2	-5,3	-5,2	9,2	14	-19,2	8
2	921,9	922,3	922,6	922,2	8,1	0,0	8,0	-1,4	1,4	6,4	3,2	3,6	17,1	28	-7,0	12
3	920,1	920,2	920,9	920,4	12,8	3,9	8,9	1,8	6,0	11,2	7,7	8,2	22,2	22	-1,5	27
4	920,4	920,1	920,4	920,3	13,4	3,7	9,7	2,0	6,1	11,2	7,8	8,2	23,4	27	-3,5	20
5	920,6	920,7	920,9	920,7	19,3	9,6	9,6	6,8	12,0	17,9	13,4	14,2	25,4	31	2,0	10
6	921,8	921,8	921,5	921,7	24,9	14,5	10,5	10,3	17,6	23,5	19,3	19,9	31,0	30	8,5	8
7	921,8	921,6	921,8	921,7	27,3	15,9	11,4	12,7	18,4	25,9	20,7	21,4	34,1	23	10,6	26
8	923,8	923,2	923,2	923,4	28,8	16,6	12,2	12,6	19,2	27,2	21,8	22,5	36,0	6	10,6	23
9	921,1	920,7	921,5	921,1	20,9	10,7	10,2	8,0	12,5	19,2	14,0	14,9	32,0	1	4,5	30
10	924,0	923,8	924,2	924,0	16,4	6,9	9,6	3,9	8,7	15,0	10,5	11,2	25,0	18	-0,5	30
11	920,6	920,2	920,6	920,5	9,0	2,1	7,0	0,0	3,8	7,3	4,8	5,2	16,4	2	-4,5	28
12	919,8	920,2	920,8	920,3	5,4	-1,4	6,8	-1,8	1,3	3,4	1,5	1,9	15,6	12	-6,8	20
год	921,5	921,4	921,7	921,5	15,4	6,2	9,3	3,8	8,4	13,8	10,0	10,5	36,0	8	-19,2	1

Месец	Напон водене паре (mb)				Релативна влажност (%)				Ветар (m/s)			Инсолација	Облачност у десетинама				Падавине (mm)			Снег (cm)	
	7	14	21	ср	7	14	21	ср	мин	ср	>6Б	>8Б	(h)	7	14	21	ср	сума	мах	дан	У	Н
1	3,0	3,6	3,4	3,4	80	74	80	78	39	2,4	4	2	97,4	7,1	5,6	5,1	5,9	41,8	8,0	17	15	2
2	5,4	6,1	5,7	5,7	79	67	74	73	33	2,3	4	0	100,3	6,9	6,8	5,5	6,4	32,4	13,9	25	12	6
3	6,6	7,1	7,0	6,9	71	56	67	65	28	2,9	5	0	185,1	5,3	6,0	4,3	5,2	30,4	8,8	8	3	1
4	7,0	7,8	7,1	7,3	74	62	69	68	30	2,7	7	0	150,6	6,2	7,4	5,8	6,5	97,8	21,7	16	10	3
5	11,3	12,6	11,5	11,8	81	62	75	73	36	2,0	2	0	211,7	6,5	6,3	3,6	5,5	99,5	16,2	21	-	-
6	14,9	15,5	14,7	15,0	73	54	66	65	22	1,7	1	0	282,5	3,4	5,0	4,0	4,1	27,9	8,9	24	-	-
7	14,3	14,6	14,3	14,4	68	44	60	57	26	2,0	3	0	318,2	3,0	4,0	2,7	3,2	46,1	12,2	26	-	-
8	14,2	14,6	13,9	14,2	64	42	55	54	20	1,8	1	0	300,0	2,6	3,6	1,9	2,7	13,2	7,1	21	-	-
9	10,9	11,5	11,3	11,3	76	55	72	68	27	2,0	3	0	189,0	5,4	6,2	5,1	5,6	55,5	16,3	20	-	-
10	8,8	10,1	9,4	9,4	77	60	73	70	37	1,9	2	0	204,3	3,8	4,7	3,4	4,0	83,2	27,5	25	-	-
11	6,6	7,3	6,9	6,9	81	72	80	78	35	1,5	1	0	102,8	6,4	6,7	5,7	6,3	39,1	8,9	16	2	1
12	5,3	5,7	5,1	5,4	79	74	76	77	40	3,1	9	3	71,1	7,3	7,3	7,3	7,3	80,9	20,5	3	12	5
год	9,0	9,7	9,2	9,3	75	60	71	69	20	2,2	42	5	2213,0	5,3	5,8	4,5	5,2	647,8	27,5	10	13	0

Месец	Број дана са													Правац							
	Тн ≤ -10	Тх <0	Тн <0	Тх ≥ 25	Тх ≥ 30	Тн ≥ 20	Облачност < 2	Облачност > 8	Падавине ≥ 0.1	Падавине ≥ 1	Падавине ≥ 10	К	Сн	Су	Кр	По	С	Г	Грм	≡	Сп
1	9	16	31	0	0	0	6	14	15	11	0	0	14	0	0	0	0	0	0	8	31
2	0	6	15	0	0	0	5	12	5	4	2	2	5	0	0	0	0	0	0	8	9
3	0	0	3	0	0	0	7	8	10	8	0	9	3	1	0	0	0	0	0	0	1
4	0	0	4	0	0	0	3	12	15	11	3	11	6	1	0	0	2	0	2	10	4
5	0	0	0	1	0	0	5	5	18	15	1	16	0	0	0	0	4	0	4	4	0
6	0	0	0	15	5	2	6	5	8	5	0	9	0	0	0	0	0	0	2	0	0
7	0	0	0	22	10	5	14	2	7	6	2	8	0	0	0	0	0	0	3	2	0
8	0	0	0	24	16	9	17	3	5	4	0	7	0	0	0	0	0	0	1	2	0
9	0	0	0	8	4	1	3	6	17	10	2	15	0	0	0	0	0	0	3	4	0
10	0	0	2	1	0	0	11	5	8	8	2	7	0	0	0	0	0	0	0	3	0
11	0	1	7	0	0	0	6	14	12	8	0	9	4	1	0	0	0	0	0	12	3
12	0	7	21	0	0	0	3	19	13	10	4	5	9	0	0	0	0	0	0	7	19
год	9	30	83	71	35	17	86	105	133	100	16	98	41	3	0	0	6	0	15	60	67

Месец	Честине праваца и средње брзине ветра (m/s)																
	N		NE		E		SE		S		SW		W		NW		C
	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	ч	б	тихо
1	8	4,3	17	1,4	1	1,0	2	0,5	2	10,5	17	4,5	9	2,3	14	4,4	23
2	2	1,0	12	1,1	3	0,8	3	1,3	8	2,9	25	4,3	7	2,1	14	2,6	10
3	19	3,5	5	2,8	5	1,0	1	2,0	8	6,6	12	3,2	9	3,6	20	4,1	14
4	6	3,3	9	2,1	0	0,0	7	1,7	6	4,2	16	4,5	25	2,8	14	3,1	7
5	15	2,3	16	1,6	2	1,5	5	1,2	4	2,0	7	3,7	12	2,8	21	3,3	11
6	7	2,1	14	1,5	6	0,8	4	1,0	4	2,5	6	4,2	17	2,1	19	2,8	13
7	19	2,8	10	2,1	5	1,8	2	1,5	2	1,5	9	1,6	18	2,4	18	3,1	10
8	14	3,1	19	1,6	7	1,6	12	1,5	1	2,0	2	3,0	8	1,3	20	3,0	10
9	10	2,6	16	1,3	4	1,3	9	1,2	4	4,8	8	3,9	19	2,8	11	3,0	9
10	7	2,6	13	1,5	2	1,0	1	1,0	5	1,6	15	2,5	20	2,5	22	2,4	8
11	5	1,4	9	1,2	3	1,0	8	1,4	5	1,4	18	2,7	14	2,6	12	2,2	16
12	3	1,0	0	0,0	2	1,0	0	0,0	10	6,8	25	5,9	13	2,6	27	1,9	13
год	115	2,8	140	1,5	40	1,1	54	1,4	59	4,3	160	3,9	171	2,6	212	2,9	144

ПРИЛОГ 6 - Приступ I - Развој модела тип 1

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from termcolor import colored, cprint
from numpy import unique
from keras import metrics
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
print('OK')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)
enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
ss=StandardScaler()
loe=LeaveOneOutEncoder()
X=df
x_columns = X.columns.drop('ConsumptionInkwh')
x = X[x_columns].values
y = X['ConsumptionInkwh'].values
target_col=[ 'CONSUMER' ]

features_num = [
    'AVG_TEMPERATURE',
    'MAXtemp', 'MINtemp',
     'humidityIn%','precipitationIn_mm',
   'NOfDwithPrecipitation',
     'NoOfClearDays',     'NoOfCloudyDays',     'InsolationInHours',
    'DaylightHours',    'SunshineHours',
       'WorkDay',
    'Weekend',
     'Holiday',
    # 'Praznik',
     'scores', 'Trends',
    '1monthBefore','2monthBefore', '3monthBefore',
    'AVGconsumption', 'razlikaOdMAX',     'razlikaOdMIN',
    ]
```

```
features_cat = [
                'CONSUMER_CATEGORY',
                'CONSUMER_GROUP',
                  'CalculationPeriod',
                  'Seasons',
                'ConsumptionZone',
                'ConsumptionZone1MB',
                'ZONE',
                 'Summer/WinterTime',
                ]
x_train, x_test, y_train, y_test = train_test_split(
    df.drop('ConsumptionInkwh',axis=1),df.ConsumptionInkwh,
    test_size=0.20, random_state=56)

zaposle=pd.DataFrame(x_test)
num_pipe = Pipeline(steps=[
    ('robust', robust),
    # ('ss', ss),
    ])
target_pipe= Pipeline(steps=[
     ('target', enc),
  ('robust', robust),
    # ('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
     ('robust', robust),
    # ('ss', ss)
])

preprocessor = ColumnTransformer(transformers=[
     ('tar', target_pipe, target_col),
    ('cat', cat_pipe, features_cat),
    ('num', num_pipe, features_num),

    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)
x_train1=x_train
x_test1=x_test

x_train = pd.DataFrame(x_train)#,columns=x_columns)
x_test = pd.DataFrame(x_test)#,columns=x_columns)


input_shape = [x_train.shape[1]]
```

```
x=input_shape
print("Input shape: {}".format(input_shape))
x=x_train

winsound.Beep(940, 200)
print("OK")
class CustomCallback(Callback):
    def __init__(self):
        super(CustomCallback, self).__init__()
        self.min_val_mape = np.inf
        self.best_epoch = 0

    def on_epoch_end(self, epoch, logs=None):
        current_val_mape = logs.get('val_mean_absolute_percentage_error')
        if current_val_mape is not None and current_val_mape < self.
 ↪min_val_mape:
            self.min_val_mape = current_val_mape
            self.best_epoch = epoch
custom_callback = CustomCallback()
```

[ ]:
```
def plott_history(history):
    hist=pd.DataFrame(history.history)
    hist['epoch']=history.epoch
    plt.title(baza+' |   MAPE= {:0.3F}'.format(score[1])+" %")
    plt.xlabel('MAPE')
    plt.plot(hist['epoch'],hist['mean_absolute_percentage_error'],label='MAPE␣
 ↪Train Error')
    plt.plot(hist['epoch'], hist['val_mean_absolute_percentage_error'],␣
 ↪label='MAPE Val Error')
    plt.legend()
    plt.savefig(str(today)+' MAPE= {:0.3F} chart_regression.png'.
 ↪format(score[1]), dpi=600, bbox_inches='tight')
    min_val_loss = min(epochs_hist.history['val_loss'])
    min_val_loss_epoch = epochs_hist.history['val_loss'].index(min_val_loss) + 1
    plt.annotate(f'Min Validation Loss: {min_val_loss:.4f}\nEpoch:␣
 ↪{min_val_loss_epoch}',
                 xy=(min_val_loss_epoch, min_val_loss), xycoords='data',
                 xytext=(10, 30), textcoords='offset points',
                 arrowprops=dict(arrowstyle="->"))

    plt.grid(True)
    plt.show()
histories = []

initial_learning_rate = 0.74184
llr=[]
```

```
def lr_exp_decay(epoch, lr):
    k = 0.07
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,
                              decay=0.0001,
                              epsilon=1e-07)  #1e-07, )

ct1 = datetime.datetime.now()
print("\n\ncurrent time:-", ct1)

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5,min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',
                        min_delta=1e-3,
                        patience=5, verbose=2, mode='auto',
                        restore_best_weights=True)



def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100
metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse,
          r_square]

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
callbacks=[tensorboard_callback,LearningRateScheduler(lr_exp_decay, verbose=1),␣
 ↪monitor]
```

```
histories = []
#  M O D E L
initializer = initializers.he_normal
bias_initializer =initializers.he_normal
activation='relu'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.75, l2=0.75)
kernel_L1=keras.regularizers.l1(0.2105)
activity_regularizer=keras.regularizers.l2(1e-5)
#  M O D E L
# brneurona=x.shape[1]
brneurona=85
inputl =Input(shape=x.shape[1],name='ULAZNI SLOJ')

l1=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                 beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_1'))(inputl)

l2=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                          kernel_initializer=initializer,
                                          ␣
 ↪kernel_regularizer=kernel_regularizer,
                                          ␣
 ↪bias_initializer=bias_initializer,
                                          name='WNL_1')))(l1)

l3=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                          kernel_initializer=initializer,
                                          ␣
 ↪kernel_regularizer=kernel_regularizer,
                                          ␣
 ↪bias_initializer=bias_initializer,
                                          name='WNL_2')))(l2)

l4=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                ␣
 ↪beta_constraint=None,gamma_constraint=None,name='LNL_2'))(l3)
```

```
output = (tfa.layers.
 ↪WeightNormalization(Dense(1,activation='linear',bias_initializer␣
 ↪='zeros',name='IZLAZ')))(l4)
model = keras.Model(inputs=inputl, outputs=output,name="ModelTip1")

print(model.summary())

imemodela='WNL'
plot_model(model, to_file=baza +' model.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[ ]:
```
# PRVO POKRETANJE
histories = []

model.compile(optimizer=opt,loss='mean_squared_error', metrics=metricss)

history=model.fit(x_train,y_train,epochs=500,validation_data=(x_test, y_test),
                  verbose=2, batch_size= 512,
                  shuffle=True,callbacks=[callbacks,custom_callback])

best_epoch = custom_callback.best_epoch
print(f"\n\nNajniža val_mape postignuta u epohi: {best_epoch}")

histories.append(history)

history_dict=history.history
pred = model.predict([x_test])

score =mean_squared_error(pred,y_test)
print("\nMSE: {}".format(score))

score2 = np.sqrt(mean_squared_error(pred,y_test))
print("RMSE: {}\n".format(score2))

y_train_pred=model.predict(x_train)

r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,pred)
print("R2 score TRAIN SET: {:0.5F}".format(r2train))
print("R2 score TEST SET: {:0.5F}".format(r2test))


maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))

prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
```

```
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate(x_test, y_test, verbose = 1)
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
model.save(str(baza)+ " "+imemodela+"MAPE="+ str(score[1])+" model .h5",␣
 ↪overwrite=True,include_optimizer= True)
plot_history(history, style="-",  side= 5,graphs_per_row = 2,␣
 ↪single_graphs=False)
```

[ ]:
```
print('\nMSE: {:0.3F}'.format(score[2]))
print('RMSE: {:0.3F}'.format(score[4]))
rmsePer=(score[4]/prosekY)*100
print('RMSE%: {:0.3F}'.format(rmsePer)+' %')
print('MAPE: {:0.3F}'.format(score[1])+' %')
print('MAE: {:0.3F}'.format(score[3]))
print('MAE%: {:0.3F}'.format(maePer)+' %')
print('R2: {:0.3F}'.format(score[5])+' %')

r2=(score[5])/100
r2 = r2_score(y_test,pred)
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('Adjusted R2: {:0.3F}'.format(adj_rsquared*100)+" %")

winsound.Beep(440, 400)
```

ПРИЛОГ 7 - Приступ I - Развој модела тип 2

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from termcolor import colored, cprint
from numpy import unique
from keras import metrics
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
print('OK')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)
X=df
x_columns = X.columns.drop('ConsumptionInkwh')
x = X[x_columns].values
y = X['ConsumptionInkwh'].values

enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(with_centering=False, with_scaling=True, unit_variance=True)
ss=StandardScaler()
target_col=['CONSUMER']

features_num = ['AVG_TEMPERATURE', 'MAXtemp', 'MINtemp',
     'humidityIn%','precipitationIn_mm', 'NOfDwithPrecipitation',
     'NoOfClearDays', 'NoOfCloudyDays', 'InsolationInHours',␣
 ↪'DaylightHours','SunshineHours',
     'WorkDay','Weekend', 'Holiday',
    'scores', 'Trends',
   '1monthBefore','2monthBefore', '3monthBefore',
    'AVGconsumption', 'razlikaOdMAX',  'razlikaOdMIN']
features_cat = ['CONSUMER_CATEGORY',
                  'CONSUMER_GROUP',
                  'CalculationPeriod','Seasons',
                'ConsumptionZone','ConsumptionZone1MB',
                  'ZONE','Summer/WinterTime']

x_train, x_test, y_train, y_test = train_test_split(
    df.drop('ConsumptionInkwh',axis=1),df.ConsumptionInkwh,
```

```
    test_size=0.20, random_state=56)

num_pipe = Pipeline(steps=[
     # ('robust', robust),
     ('ss', ss),
])
target_pipe= Pipeline(steps=[
     ('target', enc),
      #('robust', robust),
   ('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
     # ('robust', robust),
   ('ss', ss),
])

preprocessor = ColumnTransformer(transformers=[
    ('tar', target_pipe, target_col),
    ('num', num_pipe, features_num),
    ('cat', cat_pipe, features_cat),
    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)

x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)

input_shape = [x_train.shape[1]]
x=input_shape
print("Input shape: {}".format(input_shape))
x=x_train
winsound.Beep(940, 200)
```

[ ]:
```
def plott_history(history):
    hist=pd.DataFrame(history.history)
    hist['epoch']=history.epoch
    plt.title(baza+' |   MAPE= {:0.3F}'.format(score[1])+" %")
    plt.xlabel('MAPE')
    plt.plot(hist['epoch'],hist['mean_absolute_percentage_error'],label='MAPE␣
 ↪Train Error')
    plt.plot(hist['epoch'], hist['val_mean_absolute_percentage_error'],␣
 ↪label='MAPE Val Error')
    plt.legend()
    plt.savefig(str(today)+' MAPE= {:0.3F} chart_regression.png'.
 ↪format(score[1]), dpi=600, bbox_inches='tight')
```

```
    plt.show()

    histories = []
initial_learning_rate = 0.74184
llr=[]

def lr_exp_decay(epoch, lr):
    k = 0.09
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,decay=0.0001,epsilon=1e-07)

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5,min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',
                        min_delta=1e-3,
                        patience=5, verbose=2, mode='auto',
                        restore_best_weights=True)

def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100

metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse,
          r_square]

callbacks=[LearningRateScheduler(lr_exp_decay, verbose=1), monitor]

winsound.Beep(940, 200)
ct1 = datetime.datetime.now()
activation='relu'
```

```
#  M O D E L
brneurona=85
# brneurona=x.shape[1]
inputl =Input(shape=x.shape[1],name='ULAZNI SLOJ')
l1=((Dense(brneurona, activation=activation,name='GUSTI_SLOJ_1')))(inputl)
l2=((Dense(brneurona, activation=activation, name='GUSTI_SLOJ_2')))(l1)
output = ((Dense(1,activation='linear',name='IZLAZNI')))(l2)
model = keras.Model(inputs=inputl, outputs=output,name="ModelTip2")
imemodela='DENSE'

model.summary()
plot_model(model, to_file=baza +' ModelTip2.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[ ]:
```
histories = []
print('ime baze:  ', baza)
model.compile(optimizer=opt,loss='mean_squared_error', metrics=metricss)

histories.append(model.fit(x_train,y_train,epochs=500,validation_data=(x_test,␣
 ↪y_test),
                  verbose=2, batch_size= 500,
                  shuffle=True,callbacks=callbacks))
pred = model.predict([x_test])

score =mean_squared_error(pred,y_test)
print("\nFinal score (MSE): {}".format(score))
score2 = np.sqrt(mean_squared_error(pred,y_test))
print("Final score (RMSE): {}\n".format(score2))
y_train_pred=model.predict(x_train)
r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,pred)
print("R2 score TRAIN SET: {:0.5F}".format(r2train))
print("R2 score TEST SET: {:0.5F}".format(r2test))

maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))

prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate(x_test, y_test, verbose = 1)
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
```

```
model.save(str(baza)+ " "+imemodela+"MAPE="+ str(score[1])+" model.h5",␣
 ↪overwrite=True,include_optimizer= True)

plot_history( histories, show_standard_deviation=False,
    show_average=True, path="DENSE slojevi dijagrami/"+baza+" zbirno/")
ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
```

[ ]:
```
print('\nMSE: {:0.3F}'.format(score[2]))
print('RMSE: {:0.3F}'.format(score[4]))
rmsePer=(score[4]/prosekY)*100
print('RMSE%: {:0.3F}'.format(rmsePer)+' %')
print('MAPE: {:0.3F}'.format(score[1])+' %')
print('MAE: {:0.3F}'.format(score[3]))
print('MAE%: {:0.3F}'.format(maePer)+' %')
print('R2: {:0.3F}'.format(score[5])+' %')

r2=(score[5])/100
r2 = r2_score(y_test,pred)
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('Adjusted R2: {:0.3F}'.format(adj_rsquared*100)+" %")

winsound.Beep(440, 400)
```

[ ]:
```
```

ПРИЛОГ 8 - Развој модела - друге технике ML

[ ]:
```
import numpy as np
from numpy import unique
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
```

```
from keras.layers.merge import concatenate
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from termcolor import colored, cprint
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)
enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
ss=StandardScaler()
loe=LeaveOneOutEncoder()
X=df
x_columns = X.columns.drop('ConsumptionInkwh')
x = X[x_columns].values
y = X['ConsumptionInkwh'].values
target_col=[ 'CONSUMER' ]

features_num = [
    'AVG_TEMPERATURE',
    'MAXtemp', 'MINtemp',
     'humidityIn%','precipitationIn_mm',
   'NOfDwithPrecipitation',
     'NoOfClearDays',     'NoOfCloudyDays',     'InsolationInHours',
    'DaylightHours',    'SunshineHours',
       'WorkDay',
    'Weekend',
     'Holiday',
    # 'Praznik',
     'scores', 'Trends',
    '1monthBefore','2monthBefore', '3monthBefore',
    'AVGconsumption', 'razlikaOdMAX',     'razlikaOdMIN',
    ]
features_cat = [
                'CONSUMER_CATEGORY',
                'CONSUMER_GROUP',
```

```
                  'CalculationPeriod',
                  'Seasons',
                'ConsumptionZone',
                'ConsumptionZone1MB',
                'ZONE',
                 'Summer/WinterTime',
                ]
x_train, x_test, y_train, y_test = train_test_split(
    df.drop('ConsumptionInkwh',axis=1),df.ConsumptionInkwh,
    test_size=0.20, random_state=56)

zaposle=pd.DataFrame(x_test)
num_pipe = Pipeline(steps=[
    ('robust', robust),
    # ('ss', ss),
    ])
target_pipe= Pipeline(steps=[
     ('target', enc),
  ('robust', robust),
    # ('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
     ('robust', robust),
    # ('ss', ss)
])

preprocessor = ColumnTransformer(transformers=[
     ('tar', target_pipe, target_col),
    ('cat', cat_pipe, features_cat),
    ('num', num_pipe, features_num),

    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)
x_train1=x_train
x_test1=x_test

x_train = pd.DataFrame(x_train)#,columns=x_columns)
x_test = pd.DataFrame(x_test)#,columns=x_columns)


input_shape = [x_train.shape[1]]
x=input_shape
print("Input shape: {}".format(input_shape))
x=x_train
```

```
winsound.Beep(940, 200)
print("OK")
class CustomCallback(Callback):
    def __init__(self):
        super(CustomCallback, self).__init__()
        self.min_val_mape = np.inf
        self.best_epoch = 0

    def on_epoch_end(self, epoch, logs=None):
        current_val_mape = logs.get('val_mean_absolute_percentage_error')
        if current_val_mape is not None and current_val_mape < self.
 ↪min_val_mape:
            self.min_val_mape = current_val_mape
            self.best_epoch = epoch
custom_callback = CustomCallback()
```

[ ]:
```
# LINEARNA REGRESIJA
from sklearn.linear_model import LinearRegression
linear_regression_model = LinearRegression()
print("Kreiranje modela...")
linear_regression_model.fit(x_train, y_train)
print("Kreiranje završeno.")
y_pred = linear_regression_model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
prosekY=statistics.mean(y_test)
maePer=(mae/prosekY)*100
rmsePer=(rmse/prosekY)*100
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
print("\nLinearna regresija:")
print('RMSE% : {:0.3F}'.format(rmsePer))#+' %')
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print('MAE%: {:0.3F}'.format(maePer))#+' %')
print("RMSE:", rmse)
text = colored('MAPE%: {:0.3F}'.format(mape), 'red', attrs=['reverse', 'blink'])
print(text)
r2 = linear_regression_model.score(x_test, y_test)
print('R-squared: {:0.3F}'.format(r2*100)+" %")
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
```

```
print('Adjusted R2: {:0.3F}'.format(adj_rsquared*100)+" %")
print('\n')
```

[ ]:
```
#  DecisionTree
ct1 = datetime.datetime.now()
print("\ncurrent time:-", ct1)
from sklearn.tree import DecisionTreeRegressor

print("Kreiranje Decision Tree modela...")
max_features=x
decision_tree_model = DecisionTreeRegressor(criterion='squared_error',
                                            splitter='best',
                                            max_depth=None,min_samples_split=2,
                                            min_samples_leaf=1,␣
 ↪min_weight_fraction_leaf=0.0,
                                            max_features=None, random_state=42,␣
 ↪max_leaf_nodes=None,
                                            min_impurity_decrease=0.0,␣
 ↪ccp_alpha=0.0)

train_rmse_history = []
test_rmse_history = []
decision_tree_model.fit(x_train, y_train)

y_pred_decision_tree = decision_tree_model.predict(x_test)
mse = mean_squared_error(y_test, y_pred_decision_tree)

rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred_decision_tree)
r2_decision_tree = r2_score(y_test, y_pred_decision_tree)
prosekY=statistics.mean(y_test)
maePer=(mae/prosekY)*100
rmsePer=(rmse/prosekY)*100

mape = np.mean(np.abs((y_test - y_pred_decision_tree) / y_test)) * 100

print("\nDecision Tree:\n")
print('RMSE% : {:0.3F}'.format(rmsePer))#+' %')
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print('MAE%: {:0.3F}'.format(maePer))#+' %')
print("Root Mean Squared Error:", rmse)
text = colored('MAPE%: {:0.3F}'.format(mape), 'red', attrs=['reverse', 'blink'])
print(text)
r2 = metrics.r2_score(y_test,y_pred_decision_tree)
print('R-squared: {:0.5F}'.format(r2*100)+" %")
p=x_test.shape[1]
```

```
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('Adjusted R2: {:0.3F}'.format(adj_rsquared*100)+" %")
print('\n')
winsound.Beep(440, 100)
```

[ ]:
```
#SVR, LinearSVR
ct2 = datetime.datetime.now()
print("\ncurrent time:-", ct2)
from sklearn.svm import SVR, LinearSVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

print("Kreiranje LSVR modela...")
model = LinearSVR(epsilon=0.0, tol=1e-4, C=1.0, loss='epsilon_insensitive',␣
 ↪random_state=42)

model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Evaluacija L SVR modela...")

mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mse)
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

prosekY=statistics.mean(y_test)
maePer=(mae/prosekY)*100
rmsePer=(rmse/prosekY)*100

print("L SVR:\n")
print('RMSE% : {:0.3F}'.format(rmsePer))#+' %')
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print('MAE%: {:0.3F}'.format(maePer))#+' %')
print("Root Mean Squared Error:", rmse)
# print("Mean Absolute Percentage Error:", mape)
text = colored('MAPE%: {:0.3F}'.format(mape), 'red', attrs=['reverse', 'blink'])
print(text)
# print("R-squared:", r2)
r2 = metrics.r2_score(y_test,y_pred)
print('R-squared: {:0.5F}'.format(r2*100)+" %")
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
```

```
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('Adjusted R2: {:0.3F}'.format(adj_rsquared*100)+" %")
print('\n')

score = model.score(x_train, y_train)
print("R-squared TRAIN:", score)
score2 = model.score(x_test, y_test)
print("R-squared TEST:", score2)
plt.figure(figsize=(14, 8))
winsound.Beep(440, 100)
ct2 = datetime.datetime.now()
print("\ncurrent time:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
```

[ ]:
```
#  XGBoost Regression
import xgboost as xgb
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split

print("Kreiranje XGBoost Regressor modela...")
model = XGBRegressor(objective='reg:squarederror', random_state=10)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

prosekY=statistics.mean(y_test)
maePer=(mae/prosekY)*100
rmsePer=(rmse/prosekY)*100

print("\nXGBoost Regressor:\n")
print('RMSE% : {:0.3F}'.format(rmsePer))#+' %')
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print('MAE%: {:0.3F}'.format(maePer))#+' %')
print("Root Mean Square Error (RMSE):", rmse)
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
text = colored('MAPE%: {:0.3F}'.format(mape), 'red', attrs=['reverse', 'blink'])
print(text)
print('R-squared: {:0.3F}'.format(r2*100)+" %")
n = len(x_test)  # Broj instanci u test skupu
```

```
p = x_test.shape[1]  # Broj atributa
adjusted_r2 = 1 - (1 - r2) * ((n - 1) / (n - p - 1))
print('Adjusted R2: {:0.3F}'.format(adjusted_r2*100)+" %")

score = model.score(x_train, y_train)
print("R-squared TRAIN:", score)
score2 = model.score(x_test, y_test)
print("R-squared TEST:", score2)
```

[ ]:
```
# HuberRegressor
from sklearn.linear_model import HuberRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

print("Kreiranje Huber regresionog modela...")
model = HuberRegressor(epsilon=1.35)

model.fit(x_train, y_train)
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

prosekY=statistics.mean(y_test)
maePer=(mae/prosekY)*100
rmsePer=(rmse/prosekY)*100

print("\nHuber Regresija:\n")
print('RMSE% : {:0.3F}'.format(rmsePer))#+' %')
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print('MAE%: {:0.3F}'.format(maePer))#+' %')
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

print("RMSE:", rmse)
text = colored('MAPE%: {:0.3F}'.format(mape), 'red', attrs=['reverse', 'blink'])
print(text)
print("R-squared: {:0.3F}" .format(score2*100)+" %")

n = len(x_test)
p = x_test.shape[1]
adjusted_r2 = 1 - (1 - score2) * ((n - 1) / (n - p - 1))
print('Adjusted R2: {:0.3F}'.format(adjusted_r2*100)+" %")
```

[ ]:
```

```

ПРИЛОГ 9 - Приступ II - model A

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from termcolor import colored, cprint
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)
X=df
x_columns = X.columns.drop('ConsumptionInkwh')
x = X[x_columns].values
y = X['ConsumptionInkwh'].values

enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(with_centering=False, with_scaling=True, unit_variance=True)
ss=StandardScaler()
target_col=['CONSUMER']

features_num = ['AVG_TEMPERATURE', 'MAXtemp', 'MINtemp',
     'humidityIn%','precipitationIn_mm', 'NOfDwithPrecipitation',
     'NoOfClearDays', 'NoOfCloudyDays', 'InsolationInHours',␣
 ↪'DaylightHours','SunshineHours',
     'WorkDay','Weekend', 'Holiday',
    'scores', 'Trends',
   '1monthBefore','2monthBefore', '3monthBefore',
    'AVGconsumption', 'razlikaOdMAX',  'razlikaOdMIN']
features_cat = ['CONSUMER_CATEGORY',
                  'CONSUMER_GROUP',
                  'CalculationPeriod','Seasons',
                'ConsumptionZone','ConsumptionZone1MB',
                  'ZONE','Summer/WinterTime']

x_train, x_test, y_train, y_test = train_test_split(
    df.drop('ConsumptionInkwh',axis=1),df.ConsumptionInkwh,
    test_size=0.20, random_state=56)

num_pipe = Pipeline(steps=[
     # ('robust', robust),
```

```
     ('ss', ss),
])
target_pipe= Pipeline(steps=[
     ('target', enc),
      #('robust', robust),
   ('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
     # ('robust', robust),
   ('ss', ss),
])

preprocessor = ColumnTransformer(transformers=[
    ('tar', target_pipe, target_col),
    ('num', num_pipe, features_num),
    ('cat', cat_pipe, features_cat),
    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)

x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)

input_shape = [x_train.shape[1]]
x=input_shape
print("Input shape: {}".format(input_shape))
x=x_train
winsound.Beep(940, 200)
```

[ ]:
```
def plott_history(history):
    hist=pd.DataFrame(history.history)
    hist['epoch']=history.epoch
    plt.title(baza+' |   MAPE= {:0.3F}'.format(score[1])+" %")
    plt.xlabel('MAPE')
    plt.plot(hist['epoch'],hist['mean_absolute_percentage_error'],label='MAPE␣
 ↪Train Error')
    plt.plot(hist['epoch'], hist['val_mean_absolute_percentage_error'],␣
 ↪label='MAPE Val Error')
    plt.legend()
    plt.savefig(str(today)+' MAPE= {:0.3F} chart_regression.png'.
 ↪format(score[1]), dpi=600, bbox_inches='tight')
    plt.show()

    histories = []
initial_learning_rate = 0.74184
```

```
llr=[]

def lr_exp_decay(epoch, lr):
    k = 0.09
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,decay=0.0001,epsilon=1e-07)

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5,min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',
                        min_delta=1e-3,
                        patience=5, verbose=2, mode='auto',
                        restore_best_weights=True)

def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100

metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse,
          r_square]

callbacks=[LearningRateScheduler(lr_exp_decay, verbose=1), monitor]

winsound.Beep(940, 200)
ct1 = datetime.datetime.now()
activation='relu'

#  M O D E L
# brneurona=85
brneurona=x.shape[1]
```

```
inputl =Input(shape=x.shape[1],name='ULAZNI SLOJ')
l1=((Dense(brneurona, activation=activation,name='GUSTI_SLOJ_1')))(inputl)
l2=((Dense(brneurona, activation=activation, name='GUSTI_SLOJ_2')))(l1)
l3=((Dense(brneurona, activation=activation,name='GUSTI_SLOJ_3')))(l2)
output = ((Dense(1,activation='linear',name='IZLAZNI')))(l3)
model = keras.Model(inputs=inputl, outputs=output,name="ModelTipA")
imemodela='DENSE'

model.summary()
plot_model(model, to_file=baza +' ModelTip2.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[ ]:
```
histories = []
print('ime baze:  ', baza)
model.compile(optimizer=opt,loss='mean_squared_error', metrics=metricss)

histories.append(model.fit(x_train,y_train,epochs=500,validation_data=(x_test,␣
 ↪y_test),
                  verbose=2, batch_size= 500,
                  shuffle=True,callbacks=callbacks))
pred = model.predict([x_test])

score =mean_squared_error(pred,y_test)
print("\nFinal score (MSE): {}".format(score))
score2 = np.sqrt(mean_squared_error(pred,y_test))
print("Final score (RMSE): {}\n".format(score2))
y_train_pred=model.predict(x_train)
r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,pred)
print("R2 score TRAIN SET: {:0.5F}".format(r2train))
print("R2 score TEST SET: {:0.5F}".format(r2test))

maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))

prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate(x_test, y_test, verbose = 1)
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)

model.save(str(baza)+ " "+imemodela+"MAPE="+ str(score[1])+" model.h5",␣
 ↪overwrite=True,include_optimizer= True)
```

ПРИЛОГ 10 - Приступ II - Модел B

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from termcolor import colored, cprint
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
ss=StandardScaler()

X=df
x_columns = X.columns.drop('ConsumptionInkwh')
x = X[x_columns].values
y = X['ConsumptionInkwh'].values

target_col=[ 'CONSUMER']

features_num = [ 'AVG_TEMPERATURE','MAXtemp', 'MINtemp',
     'humidityIn%','precipitationIn_mm', 'NOfDwithPrecipitation', 'Holiday',
     'NoOfClearDays',     'NoOfCloudyDays',     'InsolationInHours',
    'DaylightHours',    'SunshineHours',  'WorkDay',  'Weekend',
   # 'Praznik',
     'scores', 'Trends',
    '1monthBefore','2monthBefore', '3monthBefore',
    'AVGconsumption', 'razlikaOdMAX',     'razlikaOdMIN']
features_cat = ['CONSUMER_CATEGORY',
                'CONSUMER_GROUP','CalculationPeriod',
                 'Seasons','ConsumptionZone',
                'ConsumptionZone1MB','ZONE',
                 'Summer/WinterTime'  ]
x_train, x_test, y_train, y_test = train_test_split(
    df.drop('ConsumptionInkwh',axis=1),df.ConsumptionInkwh,
    test_size=0.20, random_state=56)

num_pipe = Pipeline(steps=[
    ('robust', robust),
   # ('ss', ss),
```

```
])
target_pipe= Pipeline(steps=[
     ('target', enc),
  ('robust', robust),
    #('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
     ('robust', robust),
    #('ss', ss),
])

preprocessor = ColumnTransformer(transformers=[
     ('tar', target_pipe, target_col),
    ('cat', cat_pipe, features_cat),
    ('num', num_pipe, features_num),
    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)
x_train1=x_train
x_test1=x_test

x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)

input_shape = [x_train.shape[1]]
x=input_shape
x=x_train

class CustomCallback(Callback):
    def __init__(self):
        super(CustomCallback, self).__init__()
        self.min_val_mape = np.inf
        self.best_epoch = 0

    def on_epoch_end(self, epoch, logs=None):
        current_val_mape = logs.get('val_mean_absolute_percentage_error')
        if current_val_mape is not None and current_val_mape < self.
 ↪min_val_mape:
            self.min_val_mape = current_val_mape
            self.best_epoch = epoch
custom_callback = CustomCallback()
```

[ ]:
```
def plott_history(history):
    hist=pd.DataFrame(history.history)
    hist['epoch']=history.epoch
```

```
    plt.title(baza+' |   MAPE= {:0.3F}'.format(score[1])+" %")
    plt.xlabel('MAPE consumption')
    plt.plot(hist['epoch'],hist['mean_absolute_percentage_error'],label='MAPE␣
 ↪Train Error')
    plt.plot(hist['epoch'], hist['val_mean_absolute_percentage_error'],␣
 ↪label='MAPE Val Error')
    plt.legend()
    plt.savefig(str(today)+' MAPE= {:0.3F} chart_regression.png'.
 ↪format(score[1]), dpi=600, bbox_inches='tight')
    plt.show()

histories = []
initial_learning_rate = 0.74
llr=[]
def lr_exp_decay(epoch, lr):
    k = 0.09
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,
                              decay=0.0001,
                              epsilon=1e-07)  #1e-07, )

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5,min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',
                        min_delta=1e-3,
                        patience=5, verbose=2, mode='auto',
                        restore_best_weights=True)

def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100
```

```
metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse, r_square ]

callbacks=[LearningRateScheduler(lr_exp_decay, verbose=1),␣
 ↪monitor]#,model_checkpoint_callback]
#  M O D E L
initializer = initializers.he_normal
bias_initializer =initializers.he_normal
activation='relu'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.75, l2=0.75)
kernel_L1=keras.regularizers.l1(0.2105)
activity_regularizer=keras.regularizers.l2(1e-5)

brneurona=x.shape[1]
inputl =Input(shape=x.shape[1])
l1=(tfa.layers.WeightNormalization(Dense(brneurona, activation='relu',
           ␣
 ↪kernel_initializer=initializer,kernel_regularizer=kernel_regularizer,
               bias_initializer=bias_initializer)))(inputl)

l2=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                              ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                               beta_constraint=None,gamma_constraint=None))(l1)

l3=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                         kernel_initializer=initializer,
                                         ␣
 ↪kernel_regularizer=kernel_regularizer,
                                         ␣
 ↪bias_initializer=bias_initializer)))(l2)

l4=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                              ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                               beta_constraint=None,gamma_constraint=None))(l3)

l5=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                         kernel_initializer=initializer,
                                         ␣
 ↪kernel_regularizer=kernel_regularizer,
                                         ␣
 ↪bias_initializer=bias_initializer)))(l4)
```

```
l6=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                beta_constraint=None,gamma_constraint=None))(l5)

l5=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                                    kernel_initializer=initializer,
                                                   ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                   ␣
 ↪bias_initializer=bias_initializer)))(l4)

l6=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                beta_constraint=None,gamma_constraint=None))(l5)

output = (tfa.layers.
 ↪WeightNormalization(Dense(1,activation='linear',bias_initializer␣
 ↪='zeros',name='IZLAZ')))(l6)
model = keras.Model(inputs=inputl, outputs=output,name="PristupIImodelB")
imemodela='WNL'
plot_model(model, to_file=baza +' model.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[ ]:
```
#prvi trening
histories = []
model.compile(optimizer=opt,loss='mean_squared_error', metrics=metricss)
history=model.fit(x_train,y_train,epochs=500,validation_data=(x_test, y_test),
                  verbose=2, batch_size= 512,␣
 ↪shuffle=True,callbacks=[callbacks,custom_callback])
pred = model.predict([x_test])
best_epoch = custom_callback.best_epoch
histories.append(history)
history_dict=history.history

score =mean_squared_error(pred,y_test)
print("\nFinal score (MSE): {}".format(score))
score2 = np.sqrt(mean_squared_error(pred,y_test))
print("RMSE: {}\n".format(score2))

y_train_pred=model.predict(x_train)
r2train=r2_score(y_train,y_train_pred)
```

```
r2test=r2_score(y_test,pred)
print("R2 score on TRAIN SET: {:0.5F}".format(r2train))
print("R2 score on TEST SET: {:0.5F}".format(r2test))

maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))

prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate(x_test, y_test, verbose = 1)
print('\nMAPE: {:0.3F}'.format(score[1]),'%')

model.save(str(baza)+ " "+imemodela+"MAPE="+ str(score[1])+" SACUVAN model.h5",␣
 ↪overwrite=True,include_optimizer= True)

plot_history( histories,show_standard_deviation=False,
    show_average=True, path="Folder/"+baza+"/")
```

[ ]:
```
NOEPOCH=best_epoch # NASTAVAK obuke

score1 = metrics.mean_squared_error(pred,y_test)
print("\n\nFinal score (MSE): {}".format(score1))
score2 = np.sqrt(metrics.mean_squared_error(pred,y_test))
print("Final score (RMSE): {}\n".format(score2))
score = model.evaluate(x_test, y_test, verbose = 1)
print('\nMAPE: {:0.3F}'.format(score[1]))

histories.append(model.fit(x_train,y_train,initial_epoch=NOEPOCH,epochs=500,
                  validation_data=(x_test, y_test),verbose=2, batch_size=128,
                  shuffle=True,callbacks=callbacks))

pred = model.predict([x_test])

score =mean_squared_error(pred,y_test)
print("\nFinal score (MSE): {}".format(score))
score2 = np.sqrt(mean_squared_error(pred,y_test))
print("Final score (RMSE): {}\n".format(score2))
y_train_pred=model.predict(x_train)

r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,pred)
print("R2 score on TRAIN SET: {:0.5F}".format(r2train))
print("R2 score on TEST SET: {:0.5F}".format(r2test))
```

```
maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))
prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate(x_test, y_test, verbose = 1)
print('\nMAPE: {:0.3F}'.format(score[1]),'%')

model.save("SACUVAN model", overwrite=True,include_optimizer= True)

plot_history(
    histories,
    show_standard_deviation=False,
    show_average=True,
    title="nastavljeno")
```

[ ]:
```
#  NASTAVAK, LOAD MODEL
imemodela='SACUVAN model.h5'
model = load_model(imemodela, custom_objects={"r_square": r_square})
history=model.fit(x_train,y_train,epochs=500,
                  validation_data=(x_test, y_test),verbose=2, batch_size=500,
                  shuffle=True,callbacks=callbacks)

history_dict=history.history
loss_values=history_dict['loss']
val_loss_values=history_dict['val_loss']
rmseloss=history_dict['root_mean_squared_error']
rmsevallos=history_dict['val_root_mean_squared_error']

pred = model.predict(x_test)

score1 = metrics.mean_squared_error(pred,y_test)
print("\nMSE: {}".format(score1))
score2 = np.sqrt(metrics.mean_squared_error(pred,y_test))
print("RMSE: {}".format(score2))
y_train_pred=model.predict(x_train)
y_test_pred=model.predict(x_test)

score = model.evaluate(x_test, y_test, verbose = 1)
print("\nMAPE: ",' {:0.3F}'.format(score[1]),"%")
model.save(str(baza)+ " "+imemodela+"MAPE="+ str(score[1])+" ponovno cuvanje␣
 ↪modela.h5",
           overwrite=True,include_optimizer= True)
plot_history(history)
```

```
plot_history( histories, show_standard_deviation=False,
    show_average=True, path="DENSE slojevi dijagrami/"+baza+" zbirno/")
ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
```

[ ]:
```
print('\nMSE: {:0.3F}'.format(score[2]))
print('RMSE: {:0.3F}'.format(score[4]))
rmsePer=(score[4]/prosekY)*100
print('RMSE%: {:0.3F}'.format(rmsePer)+' %')
print('MAPE: {:0.3F}'.format(score[1])+' %')
print('MAE: {:0.3F}'.format(score[3]))
print('MAE%: {:0.3F}'.format(maePer)+' %')
print('R2: {:0.3F}'.format(score[5])+' %')

r2=(score[5])/100
r2 = metrics.r2_score(y_test,pred)
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('Adjusted R2: {:0.3F}'.format(adj_rsquared*100)+" %")

winsound.Beep(440, 400)
```

ПРИЛОГ 11 - Испитивање утицаја празника

[ ]:
```
import numpy as np
from numpy import unique
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
```

```
from keras.layers.merge import concatenate
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from termcolor import colored, cprint
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)
enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
ss=StandardScaler()
X=df
x_columns = X.columns.drop('ConsumptionInkwh')
x = X[x_columns].values
y = X['ConsumptionInkwh'].values
df['Praznik']=df['Weekend']+df['Holiday']
target_col=[ 'CONSUMER' ]
features_num = [
'AVG_TEMPERATURE',
'MAXtemp', 'MINtemp',
'humidityIn%','precipitationIn_mm',
'NOfDwithPrecipitation',
'NoOfClearDays', 'NoOfCloudyDays', 'InsolationInHours',
'DaylightHours', 'SunshineHours',
'WorkDay',
 'Praznik',
'scores', 'Trends',
'1monthBefore','2monthBefore', '3monthBefore',
'AVGconsumption', 'razlikaOdMAX', 'razlikaOdMIN',
]
features_cat = [
'CONSUMER_CATEGORY',
'CONSUMER_GROUP',
'CalculationPeriod',
'Seasons',
'ConsumptionZone',
```

```
'ConsumptionZone1MB',
'ZONE',
'Summer/WinterTime',
]

x_train, x_test, y_train, y_test = train_test_split(
    df.drop('ConsumptionInkwh',axis=1),df.ConsumptionInkwh,
    test_size=0.20, random_state=56)

num_pipe = Pipeline(steps=[
    ('robust', robust),
    # ('ss', ss),
])
target_pipe= Pipeline(steps=[
     ('target', enc),
  ('robust', robust),
    # ('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
    ('robust', robust),
    # ('ss', ss),
])

preprocessor = ColumnTransformer(transformers=[
     ('tar', target_pipe, target_col),
    ('cat', cat_pipe, features_cat),
    ('num', num_pipe, features_num),
    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)
x_train1=x_train
x_test1=x_test
x_train = pd.DataFrame(x_train)#,columns=x_columns)
x_test = pd.DataFrame(x_test)#,columns=x_columns)

input_shape = [x_train.shape[1]]
x=input_shape
print("Input shape: {}".format(input_shape))
x=x_train

winsound.Beep(940, 200)
print("OK")
class CustomCallback(Callback):
    def __init__(self):
        super(CustomCallback, self).__init__()
```

```
        self.min_val_mape = np.inf
        self.best_epoch = 0
    def on_epoch_end(self, epoch, logs=None):
        current_val_mape = logs.get('val_mean_absolute_percentage_error')
        if current_val_mape is not None and current_val_mape < self.
 ↪min_val_mape:
            self.min_val_mape = current_val_mape
            self.best_epoch = epoch
custom_callback = CustomCallback()
```

[ ]:
```
def plott_history(history):
    hist=pd.DataFrame(history.history)
    hist['epoch']=history.epoch
    plt.title(baza+' |   MAPE= {:0.3F}'.format(score[1])+" %")
    plt.xlabel('MAPE')
    plt.plot(hist['epoch'],hist['mean_absolute_percentage_error'],label='MAPE␣
 ↪Train Error')
    plt.plot(hist['epoch'], hist['val_mean_absolute_percentage_error'],␣
 ↪label='MAPE Val Error')
    plt.legend()
    plt.savefig(str(today)+' MAPE= {:0.3F} chart_regression.png'.
 ↪format(score[1]), dpi=600, bbox_inches='tight')
    plt.show()

histories = []
initial_learning_rate = 0.74184
llr=[]
def lr_exp_decay(epoch, lr):
    k = 0.09
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,
                              decay=0.0001,
                              epsilon=1e-07)  #1e-07, )


ct1 = datetime.datetime.now()
print("\n\ncurrent time:-", ct1)
class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        #print('lr=',K.eval(lr))
        decay = self.model.optimizer.decay
        #print('decay=',K.eval(decay))
        iterations = self.model.optimizer.iterations
```

```
        #print('iterations=',K.eval(iterations))
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5,min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',min_delta=1e-3,
                        patience=5, verbose=2, mode='auto',
                        restore_best_weights=True)

def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100

metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse,r_square]

callbacks=[LearningRateScheduler(lr_exp_decay, verbose=1),␣
 ↪monitor]#,model_checkpoint_callback]
winsound.Beep(940, 200)
print("OK")

#  M O D E L
initializer = initializers.he_normal
bias_initializer =initializers.he_normal
activation='relu'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.75, l2=0.75)
kernel_L1=keras.regularizers.l1(0.2105)
activity_regularizer=keras.regularizers.l2(1e-5)
brneurona=x.shape[1]

inputl =Input(shape=x.shape[1],name='ULAZNI SLOJ')

l1=(tfa.layers.WeightNormalization(Dense(brneurona, activation='relu',
             kernel_initializer=initializer,#.GlorotNormal(),
                kernel_regularizer=kernel_regularizer,
                bias_initializer=bias_initializer,
               name='WNL_1')))(inputl)

l2=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
```

```
                                  ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                   beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_1'))(l1)
l3=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                                  kernel_initializer=initializer,
                                                 ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                 ␣
 ↪bias_initializer=bias_initializer,
                                                 name='WNL_2')))(l2)
l4=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                  ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                  ␣
 ↪beta_constraint=None,gamma_constraint=None,name='LNL_2'))(l3)
l5=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                                  kernel_initializer=initializer,
                                                 ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                 ␣
 ↪bias_initializer=bias_initializer,
                                                 name='WNL_3')))(l4)

l6=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                  ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                  ␣
 ↪beta_constraint=None,gamma_constraint=None,name='LNL_3'))(l5)

output = (tfa.layers.
 ↪WeightNormalization(Dense(1,activation='linear',bias_initializer␣
 ↪='zeros',name='IZLAZ')))(l6)
model = keras.Model(inputs=inputl, outputs=output,name="Praznici")
imemodela='WNL'
plot_model(model, to_file=baza +' model.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[ ]:
```
histories = []
model.compile(optimizer=opt,loss='mean_squared_error', metrics=metricss)

history=model.fit(x_train,y_train,epochs=500,validation_data=(x_test, y_test),
                  verbose=2, batch_size= 512,  #500,#250
                  shuffle=True,callbacks=[callbacks,custom_callback])
```

```
best_epoch = custom_callback.best_epoch
print(f"\n\nNajniža val_mape postignuta u epohi: {best_epoch}")
histories.append(history)
history_dict=history.history

pred = model.predict([x_test])
score =mean_squared_error(pred,y_test)
print("\nFinal score (MSE): {}".format(score))
score2 = np.sqrt(mean_squared_error(pred,y_test))
print("Final score (RMSE): {}\n".format(score2))
y_train_pred=model.predict(x_train)
r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,pred)
print("R2 score on TRAIN SET: {:0.5F}".format(r2train))
print("R2 score on TEST SET: {:0.5F}".format(r2test))
maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))

prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate(x_test, y_test, verbose = 1)

text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
model.save(str(baza)+ " "+imemodela+"MAPE="+ str(score[1])+" model.h5",␣
 ↪overwrite=True,include_optimizer= True)
plot_history(history, style="-",  side= 5,graphs_per_row = 2, path="sa␣
 ↪praznicima/"+baza+" grupno/", single_graphs=False)  # da ih sacuva posebno
```

[ ]:
```
selected_metrics = [
    'mse',
    'root_mean_squared_error',
    'mean_absolute_error',
    'mean_absolute_percentage_error',
    'lr',
    'r_square']
```

```
epochs = range(1, len(history.history['loss']) + 1)

num_plots = len(selected_metrics)
fig, axes = plt.subplots(num_plots, 1, figsize=(8, 5 * num_plots))

for i, metric in enumerate(selected_metrics):
    ax = axes[i]
    train_values = history.history[metric]

    # Ako je metrika "lr", nema "val_" vrednosti
    if metric == 'lr':
        test_values = None
    else:
        test_values = history.history['val_' + metric]

    if test_values is not None:
        if metric == 'r2':
            best_epoch = test_values.index(max(test_values))
        else:
            best_epoch = test_values.index(min(test_values))

    ax.plot(epochs, train_values, 'b', label=f'Training {metric}')

    if test_values is not None:
        ax.plot(epochs, test_values, 'orange', label=f'Test {metric}')
        ax.set_title(f'{metric} with Best Test Value at Epoch {best_epoch + 1}')

    ax.set_xlabel('Epochs')
    ax.set_ylabel(metric)
    ax.legend()

   # Anotacija za najpovoljniju vrednost
    if test_values is not None:
        if metric == 'r2':
            best_value = max(test_values)
        else:
            best_value = min(test_values)
        ax.annotate(f'Najbolja vrednost: {best_value:.4f}', xy=(best_epoch + 1,␣
 ↪best_value), xytext=(best_epoch + 1, best_value + 0.1),
                    arrowprops=dict(arrowstyle='->',␣
 ↪connectionstyle='arc3,rad=0.5'))

plt.tight_layout()
plt.show()
```

ПРИЛОГ 12 - Класификација - Зона потрошње

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from termcolor import colored, cprint
from numpy import unique
from numpy import argmax
from keras import losses
from keras import optimizers
from keras import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
X=df
# x_columns = X.columns.drop('Seasons')
x_columns = X.columns.drop('ConsumptionZone')
x = X[x_columns].values
x= pd.DataFrame(x,columns=x_columns)
# y = X['Seasons'].values
y = X['ConsumptionZone'].values
yLabel='ConsumptionZone'
# yLabel='Seasons'
x= pd.DataFrame(x)
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
ss=StandardScaler()
y = LabelEncoder().fit_transform(y)
n_class = len(unique(y))

target_col=['CONSUMER']
features_num = [
    'AVG_TEMPERATURE',
    'MAXtemp', 'MINtemp',
   #   'humidityIn%','precipitationIn_mm',
   # 'NOfDwithPrecipitation',
```

```
    'NoOfClearDays',     'NoOfCloudyDays',     'InsolationInHours',     ␣
 ↪'DaylightHours',    'SunshineHours',
    #'WorkDay','Weekend', 'Holiday',
    'scores', 'Trends',
    #'1YBefore','2YBefore','3YBefore',
    '1monthBefore','2monthBefore', '3monthBefore',
    'AVGconsumption',    'razlikaOdMAX',     'razlikaOdMIN']
features_cat = [
                'CONSUMER_CATEGORY',
                'CONSUMER_GROUP',
                'CalculationPeriod',
                  'Seasons',
                 #'ConsumptionZone',
                'ConsumptionZone1MB',
                'ZONE',
                 'Summer/WinterTime',
                ]

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.20,␣
 ↪random_state=56) #10

num_pipe = Pipeline(steps=[
  ('robust', robust),
   ('ss', ss),
])
target_pipe= Pipeline(steps=[
     ('target', enc),
     ('robust', robust),
      ('ss', ss)])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder())])

preprocessor = ColumnTransformer(transformers=[
    ('tar', target_pipe, target_col),
    ('num', num_pipe, features_num),
    ('cat', cat_pipe, features_cat),
    ], remainder='drop' )
winsound.Beep(940, 200)
print("OK")
x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)

x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)

input_shape = [x_train.shape[1]]
x=input_shape
```

```
print("Input shape: {}".format(input_shape))
x=x_train

winsound.Beep(940, 200)
print("OK")
```

[ ]:
```
histories = []
#  M O D E L
initializer = initializers.he_normal
bias_initializer =initializers.GlorotUniform()
activation='softmax'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.55, l2=0.55)
kernel_L1=keras.regularizers.l1(0.2105)
activity_regularizer=keras.regularizers.l2(1e-5)
brneurona=x.shape[1]
inputl =Input(shape=x.shape[1],name='ULAZNI SLOJ')
l1=(tfa.layers.WeightNormalization(Dense(brneurona, activation='relu',
             kernel_initializer=initializer,#.GlorotNormal(),
                kernel_regularizer=kernel_regularizer,
                bias_initializer=bias_initializer,
               name='WNL_1')))(inputl)

l2=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                  beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_1'))(l1)
l3=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                                  kernel_initializer=initializer,
                                                ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                ␣
 ↪bias_initializer=bias_initializer,
                                                name='WNL_2')))(l2)
l4=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                ␣
 ↪beta_constraint=None,gamma_constraint=None,name='LNL_2'))(l3)
l5=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                                  kernel_initializer=initializer,
                                                ␣
 ↪kernel_regularizer=kernel_regularizer,
```

```
                                                    ␣
 ↪bias_initializer=bias_initializer,
                                                    name='WNL_3')))(l4)
l6=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                  ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                  ␣
 ↪beta_constraint=None,gamma_constraint=None,name='LNL_3'))(l5)

output = (tfa.layers.
 ↪WeightNormalization(Dense(n_class,activation='softmax',bias_initializer␣
 ↪='zeros',name='IZLAZ')))(l6)
model = Model(inputs=inputl, outputs=output)
imemodela='WNL klasifikacija'
plot_model(model, to_file=baza +' klasifikacija model.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[ ]:
```
initial_learning_rate = 0.74184
llr=[]
def lr_exp_decay(epoch, lr):
    k = 0.09
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)
opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,
                              decay=0.0001,
                              epsilon=1e-07)  #1e-07, )

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        llr.append(lr_with_decay)
        print("Learning Rate = ", K.eval(lr_with_decay))

rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5, min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',
                        min_delta=1e-3,
                        patience=15, verbose=2, mode='auto',
                        restore_best_weights=True)

ct1 = datetime.datetime.now()
```

```
callbacks=[monitor,LearningRateScheduler(lr_exp_decay, verbose=1), monitor]

def f1score(precision, recall):
    _f1score = ( 2 * recall * precision) / (recall + precision+ K.epsilon())
    return _f1score

metricsF = tfa.metrics.F1Score(num_classes=n_class, threshold=0.7)


losss=keras.losses.SparseCategoricalCrossentropy()
metrics=[keras.losses.SparseCategoricalCrossentropy(), keras.metrics.
 ↪SparseCategoricalAccuracy()]

model.compile(optimizer=opt,  loss=losss,  metrics=["acc"])

history=model.fit(x_train,y_train,epochs=500,
                  validation_data=(x_test, y_test),
                  verbose=2, batch_size=512,shuffle=True,
                  callbacks=callbacks)
history_dict=history.history

pred = model.predict(x_test)
pred = argmax(pred, axis=-1).astype('int')
acc = accuracy_score(y_test, pred)
print('\n\nAccuracy: %.3f  \n\n' % acc)

ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
score = model.evaluate(x_test,y_test, verbose=1)
print('\n Accuracy:', score)

confusion_mtx = [y_test, pred]
print("\n\n",confusion_mtx,"\n")

conf_matrix = confusion_matrix(y_true=y_test, y_pred=pred)
fig, ax = plt.subplots(figsize=(10,10))
ax.matshow(conf_matrix, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center',␣
 ↪size='xx-large')

plt.xlabel('Predictions', fontsize=14)
plt.ylabel('Actuals', fontsize=14)
plt.title('Confusion Matrix for '+(yLabel), fontsize=18)
```

```
plt.show()
print(pred[0])
print(np.sum(pred[0]))
matrix = confusion_matrix(y_test, pred)
print(matrix)

print(classification_report(y_test, pred))
from IPython.display import display, HTML
```

[ ]:
```
conf_matrix = confusion_matrix(y_test, pred)
tp = conf_matrix[1, 1]  # True Positive
fn = conf_matrix[1, 0]  # False Negative
tn = conf_matrix[0, 0]  # True Negative
fp = conf_matrix[0, 1]  # False Positive

tpr = tp / (tp + fn)  # True Positive Rate (Recall)
fpr = fp / (fp + tn)  # False Positive Rate
tnr = tn / (tn + fp)  # True Negative Rate

f1_scores = f1_score(y_test, pred, average=None)
for i, f1 in enumerate(f1_scores):
    print(f"F1-score za klasu {i}: {f1:.3f}")
print(f"TPR (True Positive Rate): {tpr:.3f}")
print(f"TNR (True Negative Rate): {tnr:.3f}")
print(f"FPR (False Positive Rate): {fpr:.3f}")
```

[ ]:
```
# F1-score za sve klase
from sklearn.metrics import f1_score
f1_scores = f1_score(y_test, pred, average=None)
# Kreirajte DataFrame koristeći pandas
data = {'Klasa': np.unique(y_test),
        'TPR': tpr,
        'FPR': fpr,
        'TNR': tnr,
        'F1': f1_scores}

df = pd.DataFrame(data)
# Ispisivanje tabele
print(df)
```

[ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(history.history['acc'], label='Acc')
plt.plot(history.history['val_acc'], label='Acc Val')
plt.title('                              ')
plt.xlabel('    (    )')
plt.ylabel('      ')
plt.legend()
```

```
max_acc = max(history.history['val_acc'])
max_acc_epoch = history.history['val_acc'].index(max_acc) #+ 1
plt.annotate(f'               : {max_acc:.4f}\n    (    ): {max_acc_epoch}',
             xy=(max_acc_epoch, max_acc), xycoords='data',
             xytext=(10, 30), textcoords='offset points',
             arrowprops=dict(arrowstyle="->"))
plt.grid(True)
plt.show()
```

[ ]:
```
import seaborn as sns
group_counts = ["{0:0.0f}".format(value) for value in
                conf_matrix.flatten()]

group_percentages = ["{0:.2%}".format(value) for value in
                     conf_matrix.flatten()/np.sum(conf_matrix)]

labels = [f"{v1}\n{v2}\n" for v1, v2 in
          zip(group_counts,group_percentages)]

labels = np.asarray(labels).reshape(3,3)
ax = sns.heatmap(conf_matrix, annot=labels, fmt='', cmap='Blues')
ax.set_title("             '"+(yLabel)+"'\n", fontsize=18);
ax.set_xlabel('\n         ')
ax.set_ylabel('       \n');
ax.xaxis.set_ticklabels(['   ', '    ', '    '])
ax.yaxis.set_ticklabels(['   ', '    ', '    '])
plt.savefig(str(today)+' matrica konfuzije KLASIFIKACIJA    .png', dpi=600,␣
 ↪bbox_inches='tight')
plt.show()
```

[ ]:
```
print('Test loss:{:0.2F}'.format(score[0]) )
print('CAT ACC: {:0.2F}'.format(score[1]*100)+" %")
print('ACCURACY: {:0.2F}'.format(score[1]*100)+" %")
from sklearn.metrics import f1_score
print(f1_score(y_test, pred, average=None))
print(f1_score(y_test, pred, average='micro') )
print(f1_score(y_test, pred, average="macro"))
print(f1_score(y_test, pred, average='weighted'))
```

ПРИЛОГ 13 - Класификација - Годишња доба

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from termcolor import colored, cprint
from numpy import unique
from numpy import argmax
from keras import losses
from keras import optimizers
from keras import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
X=df
x_columns = X.columns.drop('Seasons')
x = X[x_columns].values
x= pd.DataFrame(x,columns=x_columns)
y = X['Seasons'].values
# y = X['ConsumptionZone'].values
# yLabel='ConsumptionZone'
yLabel='Seasons'
x= pd.DataFrame(x)
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
ss=StandardScaler()
y = LabelEncoder().fit_transform(y)
n_class = len(unique(y))

target_col=['CONSUMER']
features_num = [
    'AVG_TEMPERATURE',
    'MAXtemp', 'MINtemp',
     'humidityIn%','precipitationIn_mm',
   'NOfDwithPrecipitation',
```

```
    'NoOfClearDays',     'NoOfCloudyDays',     'InsolationInHours',     ␣
 ↪'DaylightHours',    'SunshineHours',
    'WorkDay','Weekend', 'Holiday',
   'scores', 'Trends',
   #'1YBefore','2YBefore','3YBefore',
   '1monthBefore','2monthBefore', '3monthBefore',
   'AVGconsumption',    'razlikaOdMAX',     'razlikaOdMIN']
features_cat = [
                'CONSUMER_CATEGORY',
                'CONSUMER_GROUP',
                'CalculationPeriod',
                 # 'Seasons',
                 'ConsumptionZone',
                'ConsumptionZone1MB',
                'ZONE',
                 'Summer/WinterTime',
                ]

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.20,␣
 ↪random_state=56)

num_pipe = Pipeline(steps=[
  ('robust', robust),
   ('ss', ss),
])
target_pipe= Pipeline(steps=[
     ('target', enc),
     ('robust', robust),
      ('ss', ss)])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder())])

preprocessor = ColumnTransformer(transformers=[
    ('tar', target_pipe, target_col),
    ('num', num_pipe, features_num),
    ('cat', cat_pipe, features_cat),
    ], remainder='drop' )
winsound.Beep(940, 200)
print("OK")
x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)

x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)

input_shape = [x_train.shape[1]]
x=input_shape
```

```
print("Input shape: {}".format(input_shape))
x=x_train
winsound.Beep(940, 200)
print("OK")
```

[ ]:
```
histories = []
#  M O D E L
initializer = initializers.he_normal
bias_initializer =initializers.GlorotUniform()
activation='softmax'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.55, l2=0.55)
kernel_L1=keras.regularizers.l1(0.2105)
activity_regularizer=keras.regularizers.l2(1e-5)
brneurona=x.shape[1]
inputl =Input(shape=x.shape[1],name='ULAZNI SLOJ')
l1=(tfa.layers.WeightNormalization(Dense(brneurona, activation='relu',
             kernel_initializer=initializer,#.GlorotNormal(),
                kernel_regularizer=kernel_regularizer,
                bias_initializer=bias_initializer,
               name='WNL_1')))(inputl)

l2=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                               ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_1'))(l1)
l3=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                          kernel_initializer=initializer,
                                          ␣
 ↪kernel_regularizer=kernel_regularizer,
                                          ␣
 ↪bias_initializer=bias_initializer,
                                          name='WNL_2')))(l2)
l4=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                               ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                               ␣
 ↪beta_constraint=None,gamma_constraint=None,name='LNL_2'))(l3)
l5=(tfa.layers.WeightNormalization(Dense(brneurona, activation=activation,
                                          kernel_initializer=initializer,
                                          ␣
 ↪kernel_regularizer=kernel_regularizer,
                                          ␣
 ↪bias_initializer=bias_initializer,
```

```
                                                    name='WNL_3')))(l4)
l6=(layers.LayerNormalization(axis=-1,epsilon=0.
  ↪01,center=True,scale=True,beta_initializer="zeros",
                                 ␣
  ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                 ␣
  ↪beta_constraint=None,gamma_constraint=None,name='LNL_3'))(l5)

output = (tfa.layers.
  ↪WeightNormalization(Dense(n_class,activation='softmax',bias_initializer␣
  ↪='zeros',name='IZLAZ')))(l6)
model = Model(inputs=inputl, outputs=output)
imemodela='WNL klasifikacija'
plot_model(model, to_file=baza +' klasifikacija model.png', show_shapes=True,␣
  ↪show_layer_names=True)
```

[ ]:
```
initial_learning_rate = 0.74184
llr=[]
def lr_exp_decay(epoch, lr):
    k = 0.09
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)
opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,
                              decay=0.0001,
                              epsilon=1e-07)  #1e-07, )

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        llr.append(lr_with_decay)
        print("Learning Rate = ", K.eval(lr_with_decay))

rlrop = ReduceLROnPlateau(monitor='val_loss' ,
                          factor=0.2, patience=5, min_lr=0.00008,verbose=2)

monitor = EarlyStopping(monitor= 'val_loss',
                        min_delta=1e-3,
                        patience=15, verbose=2, mode='auto',
                        restore_best_weights=True)

ct1 = datetime.datetime.now()
callbacks=[monitor,LearningRateScheduler(lr_exp_decay, verbose=1), monitor]
```

```
def f1score(precision, recall):
    _f1score = ( 2 * recall * precision) / (recall + precision+ K.epsilon())
    return _f1score

metricsF = tfa.metrics.F1Score(num_classes=n_class, threshold=0.7)


losss=keras.losses.SparseCategoricalCrossentropy()
metrics=[keras.losses.SparseCategoricalCrossentropy(), keras.metrics.
 ↪SparseCategoricalAccuracy()]

model.compile(optimizer=opt,  loss=losss,  metrics=["acc"])

history=model.fit(x_train,y_train,epochs=500,
                  validation_data=(x_test, y_test),
                  verbose=2, batch_size=512,shuffle=True,
                  callbacks=callbacks)
history_dict=history.history

pred = model.predict(x_test)
pred = argmax(pred, axis=-1).astype('int')
acc = accuracy_score(y_test, pred)
print('\n\nAccuracy: %.3f  \n\n' % acc)


ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
score = model.evaluate(x_test,y_test, verbose=1)
print('\n Accuracy:', score)

confusion_mtx = [y_test, pred]
print("\n\n",confusion_mtx,"\n")

conf_matrix = confusion_matrix(y_true=y_test, y_pred=pred)
fig, ax = plt.subplots(figsize=(10,10))
ax.matshow(conf_matrix, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center',␣
 ↪size='xx-large')

plt.xlabel('Predictions', fontsize=14)
plt.ylabel('Actuals', fontsize=14)
plt.title('Confusion Matrix for '+(yLabel), fontsize=18)
plt.show()
```

```
print(pred[0])
print(np.sum(pred[0]))
matrix = confusion_matrix(y_test, pred)
print(matrix)
print(classification_report(y_test, pred))
from IPython.display import display, HTML
```

[ ]:
```
conf_matrix = confusion_matrix(y_test, pred)
# TPR, FPR i TNR
tp = conf_matrix[1, 1]  # True Positive
fn = conf_matrix[1, 0]  # False Negative
tn = conf_matrix[0, 0]  # True Negative
fp = conf_matrix[0, 1]  # False Positive

tpr = tp / (tp + fn)  # True Positive Rate (Recall)
fpr = fp / (fp + tn)  # False Positive Rate
tnr = tn / (tn + fp)  # True Negative Rate
# F1-score za sve klase
f1_scores = f1_score(y_test, pred, average=None)
for i, f1 in enumerate(f1_scores):
    print(f"F1-score za klasu {i}: {f1:.3f}")
print(f"TPR (True Positive Rate): {tpr:.3f}")
print(f"TNR (True Negative Rate): {tnr:.3f}")
print(f"FPR (False Positive Rate): {fpr:.3f}")
```

[ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(history.history['acc'], label='Acc')
plt.plot(history.history['val_acc'], label='Acc Val')
plt.title('                              ')
plt.xlabel('    (    )')
plt.ylabel('      ')
plt.legend()

max_acc = max(history.history['val_acc'])
max_acc_epoch = history.history['val_acc'].index(max_acc) #+ 1
plt.annotate(f'               : {max_acc:.4f}\n    (    ): {max_acc_epoch}',
             xy=(max_acc_epoch, max_acc), xycoords='data',
             xytext=(10, 30), textcoords='offset points',
             arrowprops=dict(arrowstyle="->"))
plt.grid(True)
plt.show()
```

[ ]:
```
import seaborn as sns
group_counts = ["{0:0.0f}".format(value) for value in
                conf_matrix.flatten()]

group_percentages = ["{0:.2%}".format(value) for value in
```

```
                    conf_matrix.flatten()/np.sum(conf_matrix)]

labels = [f"{v1}\n{v2}\n" for v1, v2 in
          zip(group_counts,group_percentages)]

labels = np.asarray(labels).reshape(4,4)

ax = sns.heatmap(conf_matrix, annot=labels, fmt='', cmap='Blues')
ax.set_title("             '"+(yLabel)+"'\n", fontsize=18);
ax.set_xlabel('\n          ')
ax.set_ylabel('        \n');

ax.xaxis.set_ticklabels(['  ', '    ', '  ', '   '])
ax.yaxis.set_ticklabels(['  ', '    ' ,'  ', '   '])
plt.savefig(str(today)+' matrica konfuzije KLASIFIKACIJA godisnja doba.png',␣
 ↪dpi=600, bbox_inches='tight')
plt.show()
```

[ ]:
```
print('Test loss:{:0.2F}'.format(score[0]) )
print('CAT ACC: {:0.2F}'.format(score[1]*100)+" %")
print('ACCURACY: {:0.2F}'.format(score[1]*100)+" %")
from sklearn.metrics import f1_score
print(f1_score(y_test, pred, average=None))
print(f1_score(y_test, pred, average='micro') )
print(f1_score(y_test, pred, average="macro"))
print(f1_score(y_test, pred, average='weighted'))
```

[ ]:
```

```

ПРИЛОГ 14 - Мултимодалност - Метеоролошки и неметеоролошки подаци

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from termcolor import colored, cprint
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)

X=df
x_columns = X.columns.drop([
     'AVG_TEMPERATURE',
    'MAXtemp', 'MINtemp',#'Praznik',
    'humidityIn%','precipitationIn_mm', 'NOfDwithPrecipitation',
    'NoOfClearDays', 'NoOfCloudyDays', 'InsolationInHours',␣
 ↪'DaylightHours','SunshineHours'])

x = X[x_columns].values
x=pd.DataFrame(x,columns=[x_columns])
x1_columns = ['AVG_TEMPERATURE',
    'MAXtemp', 'MINtemp',
    'humidityIn%','precipitationIn_mm', 'NOfDwithPrecipitation',
    'NoOfClearDays', 'NoOfCloudyDays', 'InsolationInHours',␣
 ↪'DaylightHours','SunshineHours']

x1 = X[x1_columns].values
x1=pd.DataFrame(x1,columns=[x1_columns])
y = X['ConsumptionInkwh'].values
y_columns=['ConsumptionInkwh']
winsound.Beep(940, 200)
print('OK')
ydf=pd.DataFrame(y,columns=[y_columns])

enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
ss=StandardScaler()
```

```
target_col=[('CONSUMER',)]
features_num = [ (      'Praznik',),
            (            'scores',),
            (            'Trends',),
            (      '1monthBefore',),
            (      '2monthBefore',),
            (      '3monthBefore',),
            (    'AVGconsumption',),
            (      'razlikaOdMAX',),
            (      'razlikaOdMIN',)]

features_cat = [( 'CONSUMER_CATEGORY',),
            (              'ZONE',),
            (    'CONSUMER_GROUP',),
            ( 'CalculationPeriod',),
            (           'Seasons',),
            ( 'Summer/WinterTime',),
            (   'ConsumptionZone',),
            ('ConsumptionZone1MB',),]

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.20,␣
 ↪random_state=56)
x_train1, x_test1, y_train1, y_test1 = train_test_split(x1,y,test_size=0.20,␣
 ↪random_state=56)

num_pipe = Pipeline(steps=[
  ('robust', robust),
  # ('ss', ss)
])
target_pipe= Pipeline(steps=[
     ('target', enc),
      ('robust', robust),
     # ('ss', ss)
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
     ('robust', robust),
  #  ('ss', ss)
])

preprocessor = ColumnTransformer(transformers=[
    ('tar', target_pipe, target_col),
      ('num', num_pipe, features_num),
    ('cat', cat_pipe, features_cat),
    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
```

```
x_test = preprocessor.transform(x_test)

x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)

features_num1 = [(      'AVG_TEMPERATURE',),
            (              'MAXtemp',),
            (              'MINtemp',),
            # (          'humidityIn%',),
            # (   'precipitationIn_mm',),
            # ('NOfDwithPrecipitation',),
            (        'NoOfClearDays',),
            (       'NoOfCloudyDays',),
            (    'InsolationInHours',),
            (        'DaylightHours',),
            (        'SunshineHours',)
                ]

preprocessor1 = ColumnTransformer(transformers=[
    ('num1', num_pipe, features_num1),
    ], remainder='drop')

x_train1 =preprocessor1.fit_transform(x_train1,y_train1)
x_test1 = preprocessor1.transform(x_test1)

x_train1 = pd.DataFrame(x_train1)
x_test1 = pd.DataFrame(x_test1)

input_shape = [x_train.shape[1]]
input_shape1 = [x_train1.shape[1]]
print("Input shape: {}".format(input_shape))
print("Input shape1: {}".format(input_shape1))
x=x_train
x1=x_train1
winsound.Beep(940, 200)
print("OK")
```

[ ]:
```
initializer = initializers.he_normal
bias_initializer =initializers.he_normal
activation='relu'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.5105, l2=0.5105)
kernel_L1=keras.regularizers.l1(0.2105)

input0 =Input(shape=x.shape[1],name="Ostali podaci") #
input1=Input(shape=x1.shape[1],name="Meteoroloski podaci")  # 8  #
```

```
tfalevo = (tfa.layers.WeightNormalization(Dense((x1.shape[1]),␣
↪activation=activation,
                                                  kernel_initializer=initializers.
↪GlorotNormal(),
                                                 ␣
↪kernel_regularizer=kernel_regularizer,
                                                ␣
↪bias_initializer=bias_initializer, name='WNL_METEO1')))(input1)
lnlevo =(layers.LayerNormalization(axis=-1,epsilon=0.
↪01,center=False,scale=True,beta_initializer="zeros",
                                    ␣
↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                     beta_constraint=None,gamma_constraint=None,␣
↪name='LNL_METEO1'))(tfalevo)
tfalevo2 = (tfa.layers.WeightNormalization(Dense((x1.shape[1]),␣
↪activation=activation,
                                                  kernel_initializer=initializers.
↪GlorotNormal(),
                                                 ␣
↪kernel_regularizer=kernel_regularizer,
                                                ␣
↪bias_initializer=bias_initializer, name='WNL_METEO2')))(lnlevo)
ln1 =(layers.LayerNormalization(axis=-1,epsilon=0.
↪01,center=False,scale=True,beta_initializer="zeros",
                                    ␣
↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                     beta_constraint=None,gamma_constraint=None,␣
↪name='LNL_METEO2'))(tfalevo2)
tfalevo3 = (tfa.layers.WeightNormalization(Dense((x1.shape[1]),␣
↪activation=activation,
                                                  kernel_initializer=initializers.
↪GlorotNormal(),
                                                 ␣
↪kernel_regularizer=kernel_regularizer,
                                                ␣
↪bias_initializer=bias_initializer, name='WNL_METEO3')))(ln1)
tfacentar = (tfa.layers.WeightNormalization(Dense((x.shape[1]),␣
↪activation=activation,
                                                   ␣
↪kernel_initializer=initializer,
                                                  ␣
↪kernel_regularizer=kernel_regularizer,
                                                 ␣
↪bias_initializer=bias_initializer, name='WNL_1'
                                                  )))(input0)
```

```
lncentar =(layers.LayerNormalization(axis=-1,epsilon=0.
  ↪01,center=True,scale=True,beta_initializer="zeros",
                                                    ␣
  ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                                    ␣
  ↪beta_constraint=None,gamma_constraint=None, name='LNL_1'))(tfacentar)
tfacentar2 = (tfa.layers.WeightNormalization(Dense((x.shape[1]),␣
  ↪activation=activation,kernel_initializer=initializer,
                                                          ␣
  ↪kernel_regularizer=kernel_regularizer,
                                                       ␣
  ↪bias_initializer=bias_initializer, name='WNL_2')))(lncentar)
ln2 =(layers.LayerNormalization(axis=-1,epsilon=0.
  ↪01,center=True,scale=True,beta_initializer="zeros",
                                                    ␣
  ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                  beta_constraint=None,gamma_constraint=None,␣
  ↪name='LNL_2'))(tfacentar2)
tfacentar3 = (tfa.layers.WeightNormalization(Dense((x.shape[1]),␣
  ↪activation=activation,kernel_initializer=initializer,
                                                          ␣
  ↪kernel_regularizer=kernel_regularizer,
                                                       ␣
  ↪bias_initializer=bias_initializer, name='WNL_3'
                                                       )))(ln2)
merge = concatenate([tfalevo3,tfacentar3], name='SPAJANJE')
lnhiden=(layers.LayerNormalization(axis=-1,epsilon=0.
  ↪01,center=True,scale=True,beta_initializer="zeros",
                                ␣
  ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                    beta_constraint=None,gamma_constraint=None,␣
  ↪name='LNL_3'))(merge)
output = (tfa.layers.
  ↪WeightNormalization(Dense(1,activation='linear',bias_initializer ='zeros',␣
  ↪name='IZLAZ')))(lnhiden)

model = Model(inputs=[input0,input1], outputs=output,name='Meteo')
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='meteoroloski model_plot.png', show_shapes=True,␣
  ↪show_layer_names=True)
```

[ ]:
```
histories = []
ct1=datetime.datetime.now()
print("\nSADASNJE VREME:-", ct1)
initial_learning_rate = 0.9
llr=[]
```

```
def lr_exp_decay(epoch, lr):
    k = 0.1
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,
                              rho=0.99,decay=0.00001,epsilon=1e-07)

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,factor=0.2, patience=5,min_lr=0.
 ↪00008,verbose=2)
monitor = EarlyStopping(monitor= 'val_loss',min_delta=1e-4,patience=5,␣
 ↪verbose=2, mode='auto',restore_best_weights=True)

def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100

metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse,r_square]


log_dir = "logs/fit/" + "pok" + str(1)
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
callbacks=[tensorboard_callback,LearningRateScheduler(lr_exp_decay, verbose=1),␣
 ↪monitor]

model.compile(optimizer=opt,loss="mse", metrics=metricss )

history=model.
 ↪fit([x_train,x_train1],y_train,epochs=500,validation_data=([x_test,x_test1],␣
 ↪y_test),
                  verbose=2, batch_size=512,shuffle=True,callbacks=callbacks)
histories.append(history)
history_dict=history.history
```

```
pred = model.predict([x_test,x_test1])
pred=pred.astype(int)

score =mean_squared_error(pred, y_test)
print("\nFinal score (MSE): {}".format(score))

score2 = np.sqrt(mean_squared_error(pred, y_test))
print("Final score (RMSE): {}\n".format(score2))

y_train_pred=model.predict([x_train,x_train1])
y_test_pred=model.predict([x_test,x_test1])
y_train_pred=y_train_pred.astype(int)
y_test_pred=y_test_pred.astype(int)

r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,y_test_pred)
print("R2 score on TRAIN SET: {:0.5F}".format(r2train))
print("R2 score on TEST SET: {:0.5F}".format(r2test))

# ZA plots
loss_values=history_dict['loss']
val_loss_values=history_dict['val_loss']

mapelos=history_dict['mean_absolute_percentage_error']
mapevallos=history_dict['val_mean_absolute_percentage_error']

rmseloss=history_dict['root_mean_squared_error']
rmsevallos=history_dict['val_root_mean_squared_error']

maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))
prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate([x_test,x_test1], y_test, verbose = 1)
ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
print("trajanje:-", trajanje)
model.save(str(today)+str(baza)+"model.h5", overwrite=True,include_optimizer=␣
 ↪True)
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
```

```
winsound.Beep(440, 200)
winsound.Beep(500, 200)
winsound.Beep(500, 200)
plot_history(history, style="-",  side= 5,graphs_per_row = 2,␣
 ↪single_graphs=False)
```

[ ]:
```
print('ime baze:  ', baza,'\n ')
print('RMSE: {:0.2F}'.format(score[4]))
rmsePer=(score[4]/prosekY)*100
print('RMSE%: {:0.2F}'.format(rmsePer)+' %')
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
print('MAE%: {:0.2F}'.format(maePer)+' %')
print('R2: {:0.2F}'.format(score[5])+" %")
r2=(score[5])/100
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('AR2: {:0.2F}'.format(adj_rsquared*100)+" %")
print("\nSADASNJE VREME:-", datetime.datetime.now())
winsound.Beep(440, 100)
```

ПРИЛОГ 15 - Мултимодалност - Нумеричка и ненумеричка обележја

[ ]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.metrics import *
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import *
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import plotly.express as px
import math
import time
import datetime
ct = datetime.datetime.now()
import statistics
from tensorflow import keras
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import callbacks
from keras.callbacks import ReduceLROnPlateau, Callback
from tensorflow.keras import initializers
import keras.backend as K
from keras.regularizers import l2
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras import layers
from category_encoders import *
import winsound
from datetime import date
today = date.today()
from sklearn import linear_model
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers.merge import concatenate
```

```
from keras.models import Model, Input, load_model
from keras.models import save
from plot_keras_history import show_history, plot_history
from keras.utils.vis_utils import plot_model
from termcolor import colored, cprint
import os
import colorama
from colorama import Fore, Back, Style
import sys
```

[ ]:
```
baza='podaci'
df = pd.read_csv(baza +'.csv')
```

[ ]:
```
ct = datetime.datetime.now()
print("start current time:-", ct)
X=df
category_1=['CONSUMER', 'CONSUMER_CATEGORY',
                          'ZONE',
                'CONSUMER_GROUP',
             'CalculationPeriod',
                      'Seasons',
             'Summer/WinterTime',
               'ConsumptionZone',
            'ConsumptionZone1MB']
numerical_22= ['AVG_TEMPERATURE','MAXtemp','MINtemp',
               # 'humidityIn%', 'precipitationIn_mm', 'NOfDwithPrecipitation',
               'Praznik', 'NoOfClearDays', 'NoOfCloudyDays',␣
 ↪'InsolationInHours',
                'DaylightHours', 'SunshineHours',
               #'WorkDay', 'Weekend','Holiday',
                 'scores', 'Trends',
               '1monthBefore', '2monthBefore','3monthBefore',
               'AVGconsumption', 'razlikaOdMAX', 'razlikaOdMIN']

target_col=[('CONSUMER',) ]

cat = [      ('CONSUMER_CATEGORY',),
                         ( 'ZONE',),
                ('CONSUMER_GROUP',),
             ('CalculationPeriod',),
                       ( 'Seasons',),
            ( 'Summer/WinterTime',),
               ('ConsumptionZone',),
            ('ConsumptionZone1MB',),
      ]
x = X[category_1].values
x=pd.DataFrame(x,columns=[category_1])
```

```
x1 = X[numerical_22].values
x1=pd.DataFrame(x1,columns=[numerical_22])

y = X['ConsumptionInkwh'].values
y_columns=['ConsumptionInkwh']

enc = TargetEncoder()
norm = preprocessing.MinMaxScaler(feature_range=(0, 1))
power = PowerTransformer(method='yeo-johnson')
robust=RobustScaler(quantile_range=(25, 75),with_centering=True,␣
 ↪with_scaling=True, unit_variance=True)
ss=StandardScaler()

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.20,␣
 ↪random_state=56)
x_train1, x_test1, y_train1, y_test1 = train_test_split(x1,y,test_size=0.20,␣
 ↪random_state=56)

num_pipe = Pipeline(steps=[
     ('robust', robust),
   #('ss', ss),
])
target_pipe= Pipeline(steps=[
        ('target', enc),
        ('robust', robust),
      #('ss', ss),
])
cat_pipe = Pipeline(steps=[
    ('encode', OneHotEncoder()),
      ('robust', robust),
    # ('ss', ss),#(with_mean=False)),
])

preprocessor = ColumnTransformer(transformers=[
    ('tar', target_pipe, target_col),
    ('cat', cat_pipe, cat),
    ], remainder='drop')

x_train =preprocessor.fit_transform(x_train,y_train)
x_test = preprocessor.transform(x_test)

x_train = pd.DataFrame(x_train)#,columns=x_columns)
x_test = pd.DataFrame(x_test)#,columns=x_columns)

#-------------------------------------
num = [('AVG_TEMPERATURE',),
 # ('humidityIn%',),
```

```
 # ('precipitationIn_mm',),
 # ('NOfDwithPrecipitation',),
 ('MAXtemp',),
 ('MINtemp',),
       ('Praznik',),
 ('NoOfClearDays',),
 ('NoOfCloudyDays',),
 ('InsolationInHours',),
 # ('DaylightHours',),
 # ('SunshineHours',),
 # ('WorkDay',),
 # ('Weekend',),
 # ('Holiday',),
 ('scores',),
 ('Trends',),
 ('1monthBefore',),
 ('2monthBefore',),
 ('3monthBefore',),
 ('AVGconsumption',),
 ('razlikaOdMAX',),
 ('razlikaOdMIN',),
      ]
preprocessor1 = ColumnTransformer(transformers=[  #NUMERICKA GRANA
      ('num', num_pipe, num),
    ], remainder='drop')

x_train1 =preprocessor1.fit_transform(x_train1,y_train1)
x_test1 = preprocessor1.transform(x_test1)

x_train1 = pd.DataFrame(x_train1)
x_test1 = pd.DataFrame(x_test1)

ydf=pd.DataFrame(y,columns=[y_columns])
input_shape = [x_train.shape[1]]
input_shape1 = [x_train1.shape[1]]
print("Input shape CAT: {}".format(input_shape))
print("Input shape NUM: {}".format(input_shape1))
x=x_train
x1=x_train1
winsound.Beep(940, 200)
print("OK")
```

[ ]:
```
# M O D E L
initializer = initializers.he_normal
bias_initializer =initializers.he_normal
activation='relu'
kernel_regularizer=keras.regularizers.l1_l2(l1=0.5105, l2=0.5105)
```

```
kernel_L1=keras.regularizers.l1(0.2105)
input0 =Input(shape=x.shape[1],name="CAT podaci")
input1=Input(shape=x1.shape[1],name="NUM podaci")

tfalevo = (tfa.layers.WeightNormalization(Dense((x1.shape[1]),␣
 ↪activation=activation,
                                                  kernel_initializer=initializer,
                                                 ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                 ␣
 ↪bias_initializer=bias_initializer, name='WNL_NUM_O1')))(input1)
lnlevo =(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=False,scale=True,beta_initializer="zeros",
                                   ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                    beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_NUM_O1'))(tfalevo)
tfalevo2 = (tfa.layers.WeightNormalization(Dense((x1.shape[1]),␣
 ↪activation=activation,
                                                    kernel_initializer=initializer,
                                                 ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                 ␣
 ↪bias_initializer=bias_initializer, name='WNL_METEO2')))(lnlevo)
ln1 =(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=False,scale=True,beta_initializer="zeros",
                                   ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                    beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_NUM_O2'))(tfalevo2)
tfalevo3 = (tfa.layers.WeightNormalization(Dense((x1.shape[1]),␣
 ↪activation=activation,
                                                    kernel_initializer=initializer,
                                                 ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                 ␣
 ↪bias_initializer=bias_initializer, name='WNL_NUM_O3')))(ln1)
tfacentar = (tfa.layers.WeightNormalization(Dense((x.shape[1]),␣
 ↪activation=activation,
                                                     ␣
 ↪kernel_initializer=initializer,
                                                 ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                 ␣
 ↪bias_initializer=bias_initializer, name='WNL_CAT_O1'
                                                  )))(input0)
```

```
lncentar =(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                          ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                           beta_constraint=None,␣
 ↪gamma_constraint=None, name='LNL_CAT_01'))(tfacentar)
tfacentar2 = (tfa.layers.WeightNormalization(Dense((x.shape[1]),␣
 ↪activation=activation,
                                                    ␣
 ↪kernel_initializer=initializer,
                                                   ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                   ␣
 ↪bias_initializer=bias_initializer, name='WNL_CAT_02'
                                                   )))(lncentar)
ln2 =(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                                        ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                    beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_CAT_02'))(tfacentar2)
tfacentar3 = (tfa.layers.WeightNormalization(Dense((x.shape[1]),␣
 ↪activation=activation,
                                                    ␣
 ↪kernel_initializer=initializer,
                                                   ␣
 ↪kernel_regularizer=kernel_regularizer,
                                                   ␣
 ↪bias_initializer=bias_initializer, name='WNL_CAT_03'
                                                   )))(ln2)
merge = concatenate([tfalevo3,tfacentar3], name='SPAJANJE')
lnhiden=(layers.LayerNormalization(axis=-1,epsilon=0.
 ↪01,center=True,scale=True,beta_initializer="zeros",
                       ␣
 ↪gamma_initializer="ones",beta_regularizer=None,gamma_regularizer=None,
                                    beta_constraint=None,gamma_constraint=None,␣
 ↪name='LNL_03'))(merge)
output = (tfa.layers.
 ↪WeightNormalization(Dense(1,activation='linear',bias_initializer ='zeros',␣
 ↪name='IZLAZ')))(lnhiden)
model = Model(inputs=[input0,input1], outputs=output,name='NumCat')
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='KAT NUM model_plot.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

```
[ ]: histories = []
ct1=datetime.datetime.now()
print("\nSADASNJE VREME:-", ct1)
initial_learning_rate = 0.9
llr=[]
def lr_exp_decay(epoch, lr):
    k = 0.1
    llr.append(lr)
    return initial_learning_rate * math.exp(-k*epoch)

opt=keras.optimizers.Adadelta(learning_rate=initial_learning_rate,  #0.001 ---␣
 ↪ne pomaze,
                              rho=0.99,decay=0.00001,epsilon=1e-07)

class MyCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.lr
        decay = self.model.optimizer.decay
        iterations = self.model.optimizer.iterations
        lr_with_decay = lr / (1. + decay * K.cast(iterations, K.dtype(decay)))
        print("Learning Rate = ", K.eval(lr_with_decay))

print_rl = MyCallback()
rmse=keras.metrics.RootMeanSquaredError()
rlrop = ReduceLROnPlateau(monitor='val_loss' ,factor=0.2, patience=5,min_lr=0.
 ↪00008,verbose=2)
monitor = EarlyStopping(monitor= 'val_loss',min_delta=1e-4,patience=5,␣
 ↪verbose=2, mode='auto',restore_best_weights=True)

def r_square(y_test, pred):
    SS_res =  K.sum(K.square(y_test - pred))
    SS_tot = K.sum(K.square(y_test - K.mean(y_test)))
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )*100

metricss=[(keras.metrics.MeanAbsolutePercentageError(
    name="mean_absolute_percentage_error", dtype=None)),
          'mse','mean_absolute_error',rmse,r_square]

log_dir = "logs/fit/" + "pok" + str(1)
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
callbacks=[tensorboard_callback,LearningRateScheduler(lr_exp_decay, verbose=1),␣
 ↪monitor]

model.compile(optimizer=opt,loss="mse", metrics=metricss )
```

```
history=model.
 ↪fit([x_train,x_train1],y_train,epochs=500,validation_data=([x_test,x_test1],␣
 ↪y_test),
                  verbose=2, batch_size=512,shuffle=True,callbacks=callbacks)
histories.append(history)
history_dict=history.history

pred = model.predict([x_test,x_test1])
pred=pred.astype(int)

score =mean_squared_error(pred, y_test)
print("\nFinal score (MSE): {}".format(score))

score2 = np.sqrt(mean_squared_error(pred, y_test))
print("Final score (RMSE): {}\n".format(score2))

y_train_pred=model.predict([x_train,x_train1])
y_test_pred=model.predict([x_test,x_test1])
y_train_pred=y_train_pred.astype(int)
y_test_pred=y_test_pred.astype(int)

r2train=r2_score(y_train,y_train_pred)
r2test=r2_score(y_test,y_test_pred)
print("R2 score on TRAIN SET: {:0.5F}".format(r2train))
print("R2 score on TEST SET: {:0.5F}".format(r2test))

# ZA plots
loss_values=history_dict['loss']
val_loss_values=history_dict['val_loss']

mapelos=history_dict['mean_absolute_percentage_error']
mapevallos=history_dict['val_mean_absolute_percentage_error']

rmseloss=history_dict['root_mean_squared_error']
rmsevallos=history_dict['val_root_mean_squared_error']

maeMODEL=mean_absolute_error(y_test,pred)
print('mae MODEL: {:0.3F}\n'.format(maeMODEL))
prosekY=statistics.mean(y_test)
maePer=(maeMODEL/prosekY)*100
print('MAE% : {:0.3F}'.format(maePer)+' %')
rmsePer=(score2/prosekY)*100
print('RMSE% : {:0.3F}'.format(rmsePer)+' %\n')
score = model.evaluate([x_test,x_test1], y_test, verbose = 1)
ct2 = datetime.datetime.now()
print("\nSADASNJE VREME:-", ct2)
trajanje=ct2-ct1
```

```
print("trajanje:-", trajanje)
model.save(str(today)+str(baza)+"model.h5", overwrite=True,include_optimizer=␣
 ↪True)
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
plot_history(history, style="-",  side= 5,graphs_per_row = 2,␣
 ↪single_graphs=False)
```

[ ]:
```
print('ime baze:  ', baza,'\n ')
print('RMSE: {:0.2F}'.format(score[4]))
rmsePer=(score[4]/prosekY)*100
print('RMSE%: {:0.2F}'.format(rmsePer)+' %')
text = colored('\nMAPE: {:0.3F}'.format(score[1]), 'red', attrs=['reverse',␣
 ↪'blink'])
print(text)
print('MAE%: {:0.2F}'.format(maePer)+' %')
print('R2: {:0.2F}'.format(score[5])+" %")
r2=(score[5])/100
p=x_test.shape[1]
N=x_test.shape[0]
rx = (1-r2)
ry = (N-1) / (N-p-1)
adj_rsquared = (1 - (rx * ry))
print('AR2: {:0.2F}'.format(adj_rsquared*100)+" %")
print("\nSADASNJE VREME:-", datetime.datetime.now())
winsound.Beep(440, 100)
```

Биографија кандидата

Драгана (Милан) Кнежевић, рођена је у Бајиној Башти, 28. марта 1987. године, одрасла на Тари - Калуђерске Баре.

Основно и средње образовање завршила у Бајиној Башти са одличним успехом.

Високо образовање (основне и специјалистичке студије) завршила на Високој пословно-техничкој школи у Ужицу (сада Академија струковних студија Западна Србија, Одсек Ужице, скраћено АССЗС) након чега 2014. уписује Факултет техничких наука у Чачку, Универзитет у Крагујевцу, студијски програм Информационе технологије. Дипломирала 2015. године код ментора др Владимира Младеновића, са оценом 10, и просеком 8.79 у току студија.

Годину дана по завршетку основних студија уписује мастер студије Информационе технологије и исте завршава 2017. године, одбраном мастер рада, код ментора др Марије Благојевић са оценом 10 и просечном оценом на мастер студијама 9.75.

Докторске студије уписала 2017. године на Факултету техничких наука у Чачку, Универзитет у Крагујевцу, студијски програм Информационе технологије.

Од почетка основних студија на АССЗС, ангажована као студент демонстратор на више наставних предмета.

Од 2013-2017. године запослена на АССЗС, Одсек Ужице, у звању Сарадник у настави за научну област Рачунарске науке.

Од 2019. године запослена на АССЗС, Одсек Ужице у звању Асистент, научна област Рачунарско инжењерство и информатика.

Од момента првог запослења ангажована као члан неколико тимова: члан организационог одбора научног скупа СЕД, тима за промоцију Одсека, тима за самовредновање, тима за одржавање и ажурирање сајта Одсека Ужице и учесник на неколико успешно реализованих пројеката.

Аутор и коаутор више научних радова у категоријама М23 (2 рада), М24 (1 рад) и М33 (15 радова).

Коаутор једног практикума и једне збирке задатака.

Удата и поносна мајка једне Софије.

Списак објављених научних и стручних радова

Категорија М23 - Рад у међународном часопису

[1] **D. Knežević,** M. Blagojević, and A. Ranković, "Monthly electricity consumption prediction: integrating artificial neural networks and calculated attributes," J. Sci. Ind. Res. JSIR, vol. 83, no. 1, 2024, doi: https://doi.org/10.56042/jsir.v83i1.3523.

[2] **D. Knežević**, Marija Blagojević, Aleksandar Ranković, Electricity Consumption Prediction Model for Improving Energy Efficiency Based on Artificial Neural

Networks, *Studies in Informatics and Control*, ISSN 1220-1766, vol. 32(1), pp. 69-79, 2023, doi: https://doi.org/10.24846/v32i1y202307.

Категорија М24 – Рад у националном часопису међународног значаја

[1] **D. Knežević,** Marija Blagojević, Classification of electricity consumers using artificial neural networks, *Facta Universitatis,* (2019), Vol. 32, No. 4, pp. 529-538;

Категорија М33 - Саопштење са међународног скупа штампано у целини

[1] **D. Knežević**, M. Blagojević, A. Ranković, "Classification of electricity consumers using model based on neural network", *International Multidisciplinary Conference CCHE*, Conference of Academies for Applied Studies in Serbia (CAASS), 2024, стр. 403-408, ISBN-978-86-82744-02-3; ISBN-978-86-82744-00-9;

[2] M. Murić, **D. Knežević**, A. Petrović, "Masters of vectors: comparative exploration of CorelDRAW, Adobe Illustrator, and Affinity Designer 2", *International Multidisciplinary Conference CCHE*, Conference of Academies for Applied Studies in Serbia (CAASS), 2024, стр. 70-75, ISBN-978-86-82744-02-3; ISBN-978-86-82744-00-9;

[3] **D. Knežević**, I. Stanišević, M. Murić, A. Petrović, "Teaching programming to undergraduate students of applied studies: challenges faced by both teachers and students in recent years", *International Multidisciplinary Conference CCHE*, Conference of Academies for Applied Studies in Serbia (CAASS), 2024, стр. 263-268, ISBN-978-86-82744-02-3; ISBN-978-86-82744-00-9;

[4] A. Petrović, P. Melenec, M. Milivojević, M. Murić, **D. Knežević**, T. Komlenović, "Developing software and hardware of a device for measuring the sample temperature in a viscometer", *International Multidisciplinary Conference CCHE*, Conference of Academies for Applied Studies in Serbia (CAASS), 2024, стр. 284-291, ISBN-978-86-82744-02-3; ISBN-978-86-82744-00-9;

[5] **Dragana Knežević,** Ljubica Diković, (2023), „The Internet of Things in healthcare: advantages, applications, and future research directions", *SED 2023*, Užice 2023, ISBN 978-86-82078-18-0;

[6] **Dragana Knežević,** Srđan Obradović, Milorad Murić, (2023), „Effect of social media on edu website traffic", *SED 2023*, Užice 2023, ISBN 978-86-82078-18-0;

[7] **Dragana Knežević,** Srđan Obradović, Milovan Milivojević, (2023), „Using DBMS and Shiny web framework to develop software for student survey data collection and analysis", *SED 2023*, Užice 2023, ISBN 978-86-82078-18-0;

[8] Milorad Murić, **Dragana Knežević**, Aleksandar Milovanović, (2023), „Mobile application for exam registration for master's studies", *SED 2023*, Užice 2023, ISBN 978-86-82078-18-0;

[9] M. Murić, N. Stefanović, M. Milanović, **D. Knežević**, (2022), „E-invoicing – Case Study in Serbia", *TIE 2022*, стр. 326-333, Čačak 2022. DOI: 10.46793/TIE22.326M;

[10] **D. Knežević,** (2021), The visualisation of electricity consumers and electricity consumption in an urban area, *SED 2021*, Užice 2021; ISBN 978-86-82078-11-1;

[11] **D. Knežević,** M. Blagojević, A. Ranković, (2019), „Prediction of electricity consumption using artificial neural networks", *KMI*, Kopaonik 2020; ISBN 978-86-6211-123-4;

[12] **D. Knežević,** „The correlation between national e-learning standards of Serbia and its neighboring countries and international standards", *TIE 2018*, ISBN: 978-86-7776-226-1, Čačak 2018;

[13] **D. Knežević,** M. Blagojević, I. Marinković, „Using data mining techniques to discover the correlation between high schools finished and higher education programmes chosen by students“, *SED 2017*, ISBN: 978-86-83573-90-5, Užice 2017;

[14] M. J. Pavlović, **D. Knežević,** S. Petrović, „Data visualization and exploration of student database“, *SED 2017*, ISBN: 978-86-83573-90-5, Užice 2017;

[15] **D. Knežević,** „3d multimedia content creation using sophisticated software“, *SED 2016*, ISBN: 978-86-83573-82-0, Užice 2016.

Уџбеници:

1. Помоћни уџбеник: Мурић Милорад, **Кнежевић Драгана**, *Корак по корак кроз алате графичког дизајна: (Corel и Photoshop)*, Академија струковних студија Западна Србија, Одсек Ужице, ИСБН: 978-86-82078-20-3, 2023.
2. У поступку штампе: Помоћни уџбеник – збирка задатака: **Кнежевић Драгана**, Мурић Милорад, *Збирка задатака за десктоп и андроид апликације: програмски језици C, C++ и JAVA*.

Образац 1

ИЗЈАВА АУТОРА О ОРИГИНАЛНОСТИ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Изјављујем да докторска дисертација под насловом:

РАЗВОЈ МОДЕЛА ЗА ПРОЦЕНУ МЕСЕЧНЕ ПОТРОШЊЕ ЕЛЕКТРИЧНЕ ЕНЕРГИЈЕ ЗАСНОВАНОГ НА ТЕХНИКАМА МАШИНСКОГ УЧЕЊА

представља *оригинално ауторско дело* настало као резултат *сопственог истраживачког рада*.

Овом Изјавом такође потврђујем:

- да сам *једини аутор* наведене докторске дисертације,
- да у наведеној докторској дисертацији *нисам извршио/ла повреду* ауторског нити другог права интелектуалне својине других лица,

У Чачку, 8. март 2024. године,

Драгана Киезевић

потпис аутора

Образац 2

ИЗЈАВА АУТОРА О ИСТОВЕТНОСТИ ШТАМПАНЕ И ЕЛЕКТРОНСКЕ ВЕРЗИЈЕ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Изјављујем да су штампана и електронска верзија докторске дисертације под насловом:

РАЗВОЈ МОДЕЛА ЗА ПРОЦЕНУ МЕСЕЧНЕ ПОТРОШЊЕ ЕЛЕКТРИЧНЕ ЕНЕРГИЈЕ ЗАСНОВАНОГ НА ТЕХНИКАМА МАШИНСКОГ УЧЕЊА

истоветне.

У Чачку, 8. март 2024. године,

Драгана Кнежевић

потпис аутора

Образац 3

ИЗЈАВА АУТОРА О ИСКОРИШЋАВАЊУ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Ја, Драгана Кнежевић,

☑ дозвољавам

☐ не дозвољавам

Универзитетској библиотеци у Крагујевцу да начини два трајна умножена примерка у електронској форми докторске дисертације под насловом:

РАЗВОЈ МОДЕЛА ЗА ПРОЦЕНУ МЕСЕЧНЕ ПОТРОШЊЕ ЕЛЕКТРИЧНЕ ЕНЕРГИЈЕ ЗАСНОВАНОГ НА ТЕХНИКАМА МАШИНСКОГ УЧЕЊА

и то у целини, као и да по један примерак тако умножене докторске дисертације учини трајно доступним јавности путем дигиталног репозиторијума Универзитета у Крагујевцу и централног репозиторијума надлежног министарства, тако да припадници јавности могу начинити трајне умножене примерке у електронској форми наведене докторске дисертације путем *преузимања*.

Овом Изјавом такође

☑ дозвољавам

☐ не дозвољавам[1]

припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од следећих *Creative Commons* лиценци:

1) Ауторство

2) Ауторство - делити под истим условима

3) Ауторство - без прерада

4) Ауторство - некомерцијално

[1] Уколико аутор изабере да не дозволи припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од *Creative Commons* лиценци, то не искључује право припадника јавности да наведену докторску дисертацију користе у складу са одредбама Закона о ауторском и сродним правима.

5) Ауторство - некомерцијално - делити под истим условима

6) Ауторство - некомерцијално - без прерада[2]

У Чачку, 8. март 2024. године,

потпис аутора

[2] Молимо ауторе који су изабрали да дозволе припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од *Creative Commons* лиценци да заокруже једну од понуђених лиценци. Детаљан садржај наведених лиценци доступан је на: http://creativecommons.org.rs/