

**УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ**

Драган Зрилић

**РАЗВОЈ РЈЕШЕЊА ЗА ПРЕДВИЂАЊЕ И
НАДЗОР ПОВИШЕНОГ ВОДОСТАЈА**

дипломски рад

Бања Лука, октобар 2019.

Тема: **РАЗВОЈ РЈЕШЕЊА ЗА ПРЕДВИЂАЊЕ И
НАДЗОР ПОВИШЕНОГ ВОДОСТАЈА**

Кључне ријечи:
Машинско учење
Машине са векторима носачима
Градијентно појачавање
Неуралне мреже

Комисија: **проф. др Славко Марић, предсједник**
проф. др Зоран Ђурић, ментор
Александар Келеч, ма, члан

Кандидат:
Драган Зрилић

УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
КАТЕДРА ЗА РАЧУНАРСТВО И ИНФОРМАТИКУ

Предмет: Методи вјештачке интелигенције

Тема: РАЗВОЈ РЈЕШЕЊА ЗА ПРЕДВИЂАЊЕ И НАДЗОР
ПОВИШЕНОГ ВОДОСТАЈА

Задатак: Машинско учење (МЛ). Типови алгоритама и типични представници. Описати типичне проблеме који се јављају у процесу анализе података и припреме скупова података за тренирање, валидацију и тестирање (нпр. корелација, балансираност, означавање итд.). Анализирати велику количину података о висини водостаја ријеке Врбас, висини снијега, температури и протоку на мјерним станицама у сливу ријеке Врбас. Анализирати МЛ алгоритме погодне за тренирање модела за предвиђање повишеног водостаја. Изабрати један од анализираних алгоритама, извршити анализу доступних података, извршити припрему скупова података за тренирање, валидацију и тестирање, те извршити тренирање одговарајућег модела. Интегрисати софтвер за управљање дроном са излазом добијеног МЛ рјешења у сврху надзора подручја за које је предвиђен повишен водостај. Извршити анализу перформанси добијеног модела.

Ментор: проф. др Зоран Ђурић

Кандидат: Драган Зрилић (1132/15)

Бања Лука, октобар 2019.

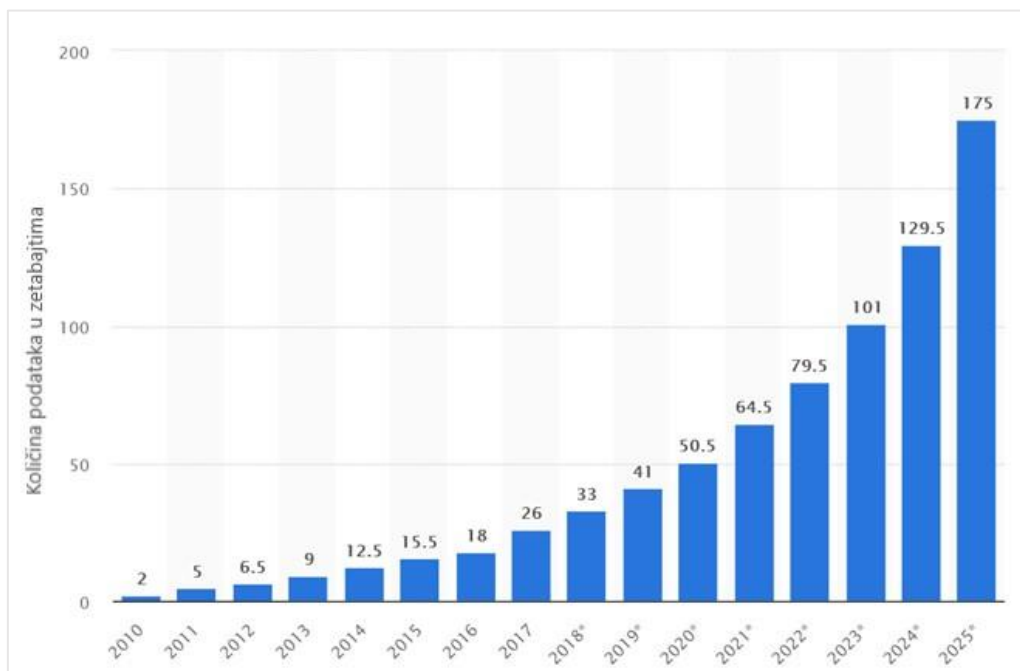
Садржај

| | | |
|-----------|---|-----------|
| 1. | УВОД..... | 1 |
| 2. | МАШИНСКО УЧЕЊЕ..... | 3 |
| 2.1. | Надгледано учење | 3 |
| 2.1.1. | Класификација | 5 |
| 2.1.2 | Регресија..... | 6 |
| 2.2. | Ненадгледано учење | 7 |
| 2.3. | Учење уз подстицаје..... | 9 |
| 3. | ПРОЦЕС МАШИНСКОГ УЧЕЊА..... | 11 |
| 3.1. | Прикупљање података | 12 |
| 3.2. | Припрема података..... | 13 |
| 3.3. | Анализа података | 19 |
| 3.4. | Избор алгорита..... | 24 |
| 3.5. | Обучавање модела..... | 24 |
| 3.6. | Евалуација модела | 26 |
| 3.7. | Тестирање модела | 29 |
| 3.8. | Примјена модела | 30 |
| 4. | АЛГОРИТМИ ЗА КЛАСИФИКАЦИЈУ И РЕГРЕСИЈУ | 32 |
| 4.1. | SVM (Support Vector Machines)..... | 32 |
| 4.2. | LightGBM | 37 |
| 4.3. | Неуралне мреже | 40 |
| 4.3.1 | Конволуционе неуралне мреже..... | 43 |
| 4.3.2. | Рекурзивне неуралне мреже | 46 |
| 5. | ПРАКТИЧНИ РАД..... | 50 |
| 6. | РЕЗУЛТАТИ..... | 61 |
| 7. | ЗАКЉУЧАК..... | 67 |
| | ЛИТЕРАТУРА..... | 68 |

Уз рад је приложен CD.

1. УВОД

Данас, у дигиталном свијету, постоје на располагању огромне количине података. Само у 2017. години произведено је 27 зетабајта података ($1\text{ZB}=10^{21}\text{B}$).^{1 2} Поређења ради, 1 зетабајт података је еквивалентан 250 милијарди DVD-ова. Иако количина података која се производи сваке године расте експоненцијалном брзином (слика 1.1), мали проценат бива искоришћен³. Велике компаније и државне институције улажу огромна средства у развој рјешења која ће омогућити лакшу и ефикаснију обраду велике количине, првенствено неструктурисаних, података. Разлог за то је што подаци у себи носе одређене информације из којих се стичу нова знања која омогућавају квалитетније услуге и пословање. Данас, већина таквих рјешења базирана је на употреби алгоритама из области вјештачке интелигенције.



Слика 1.1. Количина података произведена у свијету од 2010. године¹

Вјештачка интелигенција је грана рачунарских наука која се бави проучавањем и обликовањем рачунарских система који показују неки облик интелигенције [1]. Предност употребе алгоритама из области вјештачке интелигенције се огледа у уштеди времена и новца кроз аутоматизоване рутине и задатке у оквиру компанија, минимизацији грешака у раду, ефикаснијој обради података и њиховом бољем разумијевању.

¹ <https://www.statista.com/statistics/871513/worldwide-data-created/>

² <https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/>

³ <https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-age-of-analytics-competing-in-a-data-driven-world>

Данас, у већини развијених земаља, вјештачка интелигенција је постала саставни дио свакодневнице. Примјери употребе укључују: прилагођавање резултата претраге у *web* читачима, дигиталне асистенте (Google Assistant⁴, Amazon Alexa⁵, Apple Siri⁶), аутоматизовану доставу пакета, означавање особа на фотографијама, аутономна возила итд.

Један од циљева овог рада јесте и реализација рјешења базираног на машинском учењу. Идеја је била примијенити алгоритме машинског учења на проблем предвиђања ријечног водостаја у циљу одбране од поплава. Проблем је посматран из двије перспективе:

1. Предвиђање висине ријечног водостаја и
2. Класификација ријечног водостаја.

За реализацију су искоришћени алгоритми који се данас често користе у пракси: SVM (*Support Vector Machines*), GB (*Gradient Boosting*) и неуралне мреже (енг. *Neural Networks*). Модели су тренирани, њихове перформансе су поређене, а најбољи модел је искоришћен у реализацији одговарајућег софтвера за праћење ријечног водостаја. Додатно је имплементирана подршка за снимање подручја за које је предвиђен повишен водостај. Снимање је реализовано помоћу дрона DJI Mavic Air⁷.

У другом поглављу је описана област вјештачке интелигенције која се назива машинско учење, укључујући историјски развој машинског учења, те типове алгоритама и њихове типичне представнике. У трећем поглављу су описани кораци развоја пројекта базираног на машинском учењу, са освртом на то како је сваки од тих корака изгледао у току израде овог рада. У четвртном поглављу описани су алгоритми коришћени за израду модела, док пето поглавље садржи практичну реализацију рада. У шестом поглављу су приказани и образложени резултати поређења перформанси тренираних модела. На крају рада дат је закључак.

⁴ <https://assistant.google.com>

⁵ <https://developer.amazon.com/en-US/alexa>

⁶ <https://www.apple.com/siri/>

⁷ <https://www.dji.com/mavic-air>

2. МАШИНСКО УЧЕЊЕ

Машинско учење (енг. *Machine Learning*) је област вјештачке интелигенције која се бави дизајном алгоритама којима се „тренирају“ модели који су у стању да врше предвиђања и моделују законитости у подацима. Машинско учење представља тренутно једну од најпопуларнијих области рачунарства, како у академским круговима, тако и у индустрији. Машинско учење се данас користи од стране свих водећих свјетских компанија.

Иако у први план избија током 2000-их година, ова област има дугу историју развоја. Замишљена у радовима Алена Тјуринга, четрдесетих година прошлог вијека, активно се развија од педесетих када је конструисан перцептрон, први систем који учи једноставне законитости и представља претечу модерних неуронских мрежа. Неуронске мреже се, уз успоне и падове, развијају до деведесетих, када примат узимају метод вектора носача и други методи засновани на кернелима. За данашњи успон машинског учења заслужан је период развоја модела базираних на дубоком учењу (енг. *Deep Learning*) крајем прве и почетком друге деценије 21. вијека. Основу модела дубоког учења чине неуралне мреже. Резултати које су ови модели остварили⁸ довело је до тога да се данас вјештачка интелигенција и машинско учење поистовјећују са неуронским мрежама [1].

Неки од проблема на које је машинско учење успјешно примијењено су препознавање различитих објеката на сликама и видеу, препознавање разних облика тумора на медицинским снимцима, аутономна вожња аутомобила, аутономно летење, играње игара, класификација текста, машинско превођење, препознавање и синтеза говора, итд.

Иако су методе које се користе за рјешавање ових проблема веома разноврсне, можемо их подијелити на основу природе проблема учења. Разликујемо три групе проблема машинског учења, а то су проблеми надгледаног учења (енг. *supervised learning*), проблеми ненадгледаног учења (енг. *unsupervised learning*) и проблеми учења уз подстицаје (енг. *reinforcement learning*).

2.1. Надгледано учење

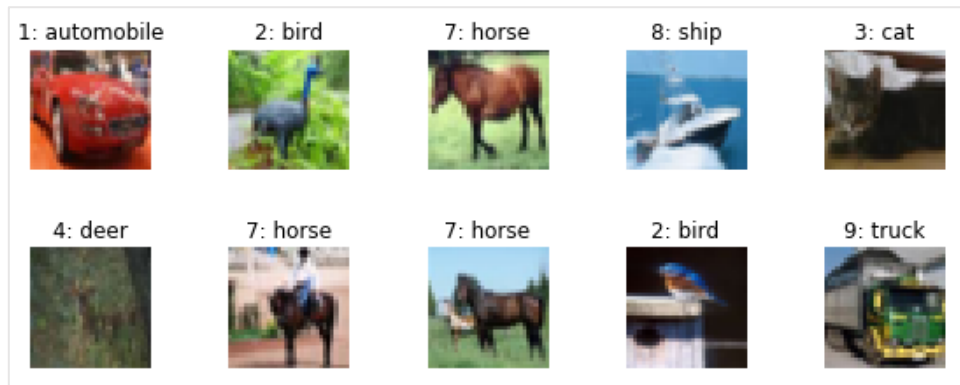
Основна карактеристика надгледаног учења је да се подаци састоје из парова (x, y) гдје x описује оно на основу чега се учи (улаз), а y представља оно што је потребно научити (излаз). Мотивација за овакав назив долази из аналогije са процесом учења при којем учитељ ученику задаје задатке, али му након његових одговора, ради поређења, даје и тачна рјешења [1].

Модел треба да научи како да на основу улаза генерише одређени излаз. Формално, модел је функција $f : X \rightarrow Y$, која дати узорак $x \in X$ пресликава у скуп $\hat{y} = f(x) \in Y$. Циљ је одредити такав модел да важи $y \approx \hat{y}$. Одступање излаза модела

⁸<https://towardsdatascience.com/the-5-deep-learning-breakthroughs-you-should-know-about-df27674ccdf2>

од стварне вриједности (поређење ученикових одговора са тачним рјешењима) је одређено функцијом грешке L (енг. *loss function*) [2].

На примјер, код проблема препознавања објеката на сликама, свака слика представља улаз, док излаз модела представља назив објекта (слика 2.1).



Слика 2.1. Подаци коришћени код проблема препознавања објеката⁹

Дакле, улаз модела чине подаци. Ти подаци се могу јавити у структурисаном, неструктурисаном и полуструктурисаном облику.

Подаци који се могу сачувати, којима се може приступити, те који се могу обрадити у облику фиксног формата, називају се структурисаним подацима. Примјери структурисаних података су каталози, записи у бази података (слика 2.2) , бројеви телефона, итд.

| Employee_ID | Employee_Name | Gender | Department | Salary_In_lacs |
|-------------|-----------------|--------|------------|----------------|
| 2365 | Rajesh Kulkarni | Male | Finance | 650000 |
| 3398 | Pratibha Joshi | Female | Admin | 650000 |
| 7465 | Shushil Roy | Male | Admin | 500000 |
| 7500 | Shubhojit Das | Male | Finance | 500000 |
| 7699 | Priya Sane | Female | Finance | 550000 |

Слика 2.2. Примјер структурисаних података

Сви подаци са непознатим обликом или структуром се могу класификовати као неструктурисани подаци. За разлику од структурисаних података, добијање вриједности из оваквих података је велики изазов. Примјер неструктурисаних података су текстуалне датотеке, слика, аудио, видео запис, резултати на *webu*, (слика 2.3.) итд.

⁹ <https://corochann.com/cifar-10-cifar-100-dataset-introduction-1258.html>



Слика 2.3. Примјер неструктурисаних података¹⁰

Полуструктурисани подаци могу садржати оба претходно наведена облика података. Могуће их је приказати у структурисаном облику, али за разлику од структурисаних података, немају фиксну структуру. Примјери полуструктурисаних података су: *e-mail*, XML (*Exstensible Markup Language*) шеме (слика 2.4), итд.

```
<rec><name>Prashant Rao</name><sex>Male</sex><age>35</age></rec>
<rec><name>Seema R.</name><sex>Female</sex><age>41</age></rec>
<rec><name>Satish Mane</name><sex>Male</sex><age>29</age></rec>
<rec><name>Subrato Roy</name><sex>Male</sex><age>26</age></rec>
<rec><name>Jeremiah J.</name><sex>Male</sex><age>35</age></rec>
```

Слика 2.4. Примјер полуструктурисаних података

Иако улазна вриједност модела може бити произвољног облика, излазна вриједност модела може бити категоричка (категоричка промјенљива) или реалан број (континуална промјенљива). Уколико је излаз модела категоричка промјенљива, тада модел рјешава проблем класификације, а у супротном проблем регресије.

2.1.1. Класификација

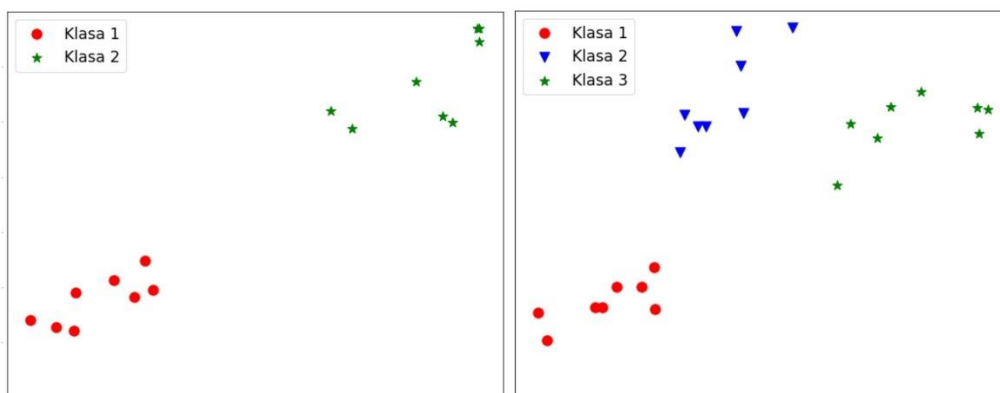
Задатак класификације је да се на основу обиљежја (енг. *features*) улазног узорка одреди класа којој узорак припада. Скуп класа је предефинисан и садржи коначно много елемената. Уколико скуп класа садржи само два елемента, ради се о бинарној класификацији (енг. *binary classification*), а уколико садржи више од два елемента онда се ради о класификацији у више класа (енг. *multiclass classification*) (слика 2.5) [2]. Неки од примјера класификације су [2]:

- Детекција нежељених порука (енг. *spam*) у електронској пошти – улаз модела је порука, а излаз су класе: нежељена порука, легитимна порука
- Препознавање говора – улаз модела је аудио снимак, а класе су изговорене ријечи
- Медицинска дијагностика – улаз је радиолошки/хистопатолошки снимак, а класе су могуће патолошке промјене

¹⁰ <https://www.laserfiche.com/ecmblog/4-ways-to-manage-unstructured-data-with-ecm/>

- Класификација музике по жанровима – улаз је аудио снимак, а класе су музички жанрови.

Уколико су класе међусобно дисјунктне, тада се сваки узорак сврстава у тачно једну класу. Ако класе нису међусобно дисјунктне онда се ради о проблему означавања (енг. *tagging*), тада се сваком узорку може додијелити више ознака класа.



Слика 2.5. Примјер бинарне класификације и класификације у више класа

Представници класификационих алгоритама су:

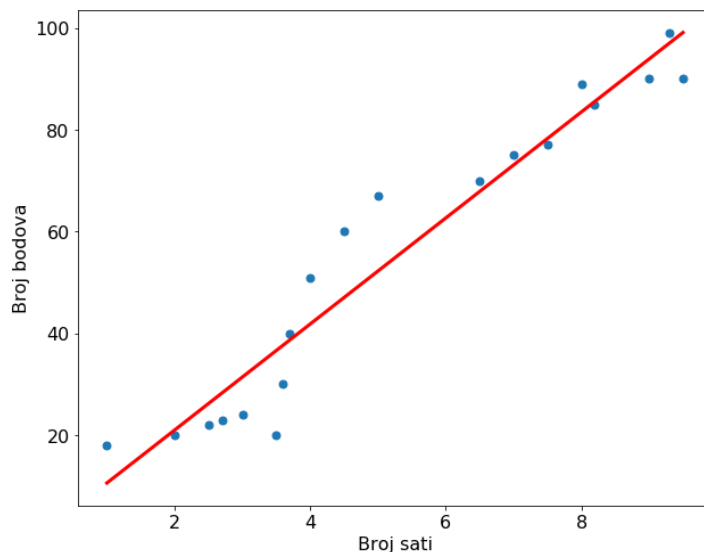
1. Логистичка регресија (енг. *Logistic Regression*)
2. Наивни Бајесов класификатор (енг. *Naïve Bayes Classifier*)
3. Најближи сусједи (енг. *Nearest Neighbor*)
4. SVM
5. Стабла одлучивања (енг. *Decision Trees*)
6. *Boosting*
7. Насумичне шуме (енг. *Random Forest*)
8. Неуралне мреже.

У четвртом поглављу су описани класификациони алгоритми који су искоришћени за израду практичног дијела рада.

2.1.2 Регресија

Задатак регресије је да на основу обиљежја улазног узорака предвиди вриједност излаза, при чему та вриједност представља реалан број (континуална промјенљива). Регресиони модели се могу подијелити на основу броја обиљежја улазног узорака:

1. Једноставни (Прости) регресиони модел (енг. *Simple Regression Model*) – улазни узорак се састоји од само једног обиљежја
2. Сложени регресиони модел (енг. *Multiple Regression Model*) – улазни узорак се састоји од више обиљежја.



Слика 2.6. Регресиони проблем – предвиђање броја бодова на тесту

Примјери регресионих проблема су:

- Предвиђање нивоа водостаја на основу историјских података о водостају
- Предвиђање броја бодова на тесту на основу утрошеног времена за учење (слика 2.6.)
- Предвиђање цијена акција на берзи на основу берзанских индекса.

2.2. Ненадгледано учење

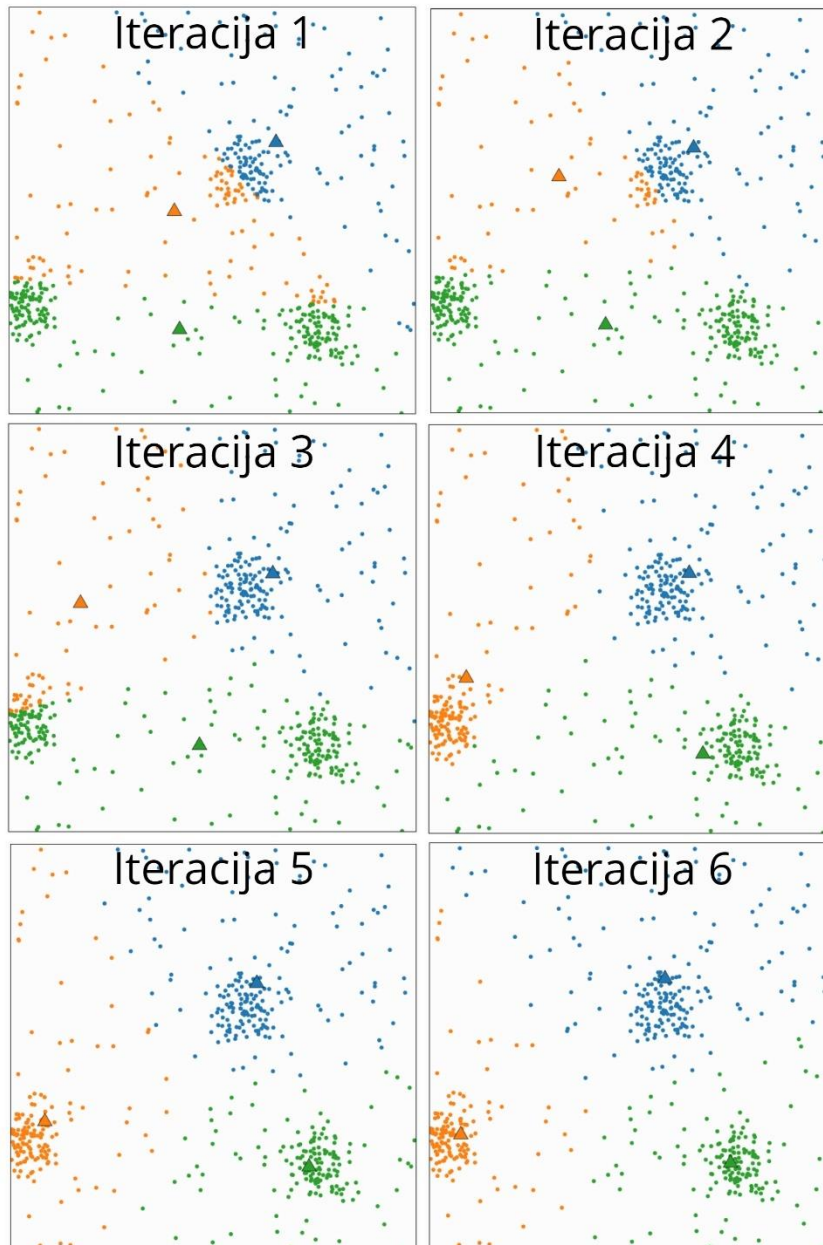
Ненадгледано учење се карактерише недостатком информације о томе шта је потребно научити, тј. не постоји информација о жељеној излазној вриједности. Мотивација за назив долази из тога што у овом случају не постоји „учитељ са тачним рјешењима, већ ученици сами морају да науче тачне одговоре“. Због тога, код оваквих метода постоји недостатак општости [1].

Кластеризовање (енг. *Clustering*) представља једну врсту проблема налажења структуре у подацима. Кластеризовање има за циљ да на основу улазних података пронађе структуре група (слика 2.7) [1].

Неки од примјера кластеризовања су [1]:

- Груписање текстова
- Идентификација заједница у друштвеним мрежама
- Проналажење сличних слика
- Детекција ткива на медицинским снимцима.

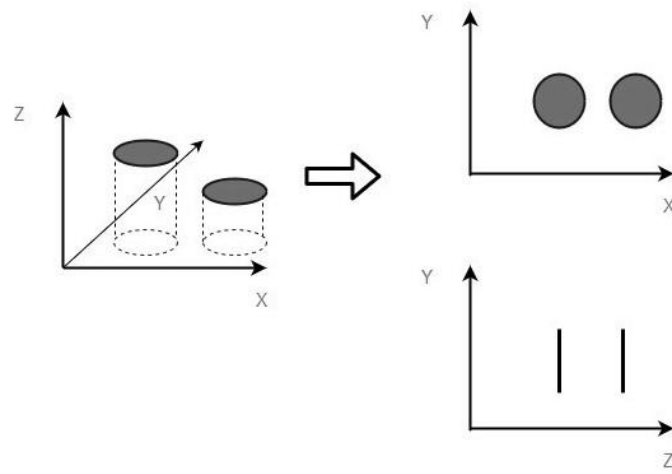
Кластеризовање се често користи у комбинацији са методама надгледаног учења у виду обраде података. Захваљујући кластеризовању могуће је читаве групе података замијенити са једним представником из те групе. Иако на тај начин долази до губитка информација, рачунарске и меморијске перформансе се знатно побољшавају [1].



Слика 2.7. Примјер кластеризовања података

Друга врста ненадгледаног учења је смањење димензионалности (енг. *Dimensionality Reduction*). Циљ ових алгоритама је да високо-димензионалне податке „претворе“ у податке ниже димензионалности (слика 2.8). Као и са кластеризовањем, ови алгоритми се често користе у комбинацији са алгоритмима надгледаног учења. Наиме, подаци често садрже шум (енг. *noise*) који знатно деградира перформансе

модела. Смањењем димензионалности долази до уклањања шума и извлачења скривених информација у подацима.



Слика 2.8. Смањење димензионалности података¹¹

Алгоритми који спадају у ову врсту учења су :

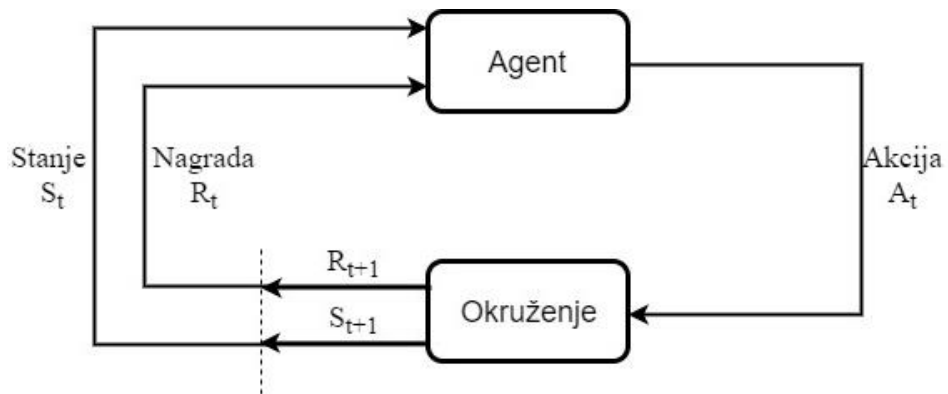
1. *K Means*
2. PCA (енг. *Principal Component Analysis*)
3. t-SNE (енг. *t-Distributed Stochastic Neighbor Embedding*)
4. Правило удруживања (енг. *Association rule*).

Трећа врста ненадгледаног учења је детекција вансеријских узорака (енг. *outlier detection*). Код ове врсте учења излаз модела је реалан број који представља разлику улазног узорка од „типичног“ узорка у скупу података.

2.3. Учење уз подстицаје

Учење уз подстицаје се користи када је потребно ријешити проблем извршавањем низа акција, чијим се заједничким утицајем долази до рјешења проблема. Агент (програм) дјелује на окружење (енг. *Environment*) извршавањем низа акција. Те акције утичу на стање окружења (енг. *State*), које повратно утиче на агента са повратним информацијама које могу бити „награде“ или „казне“ у виду нумеричких вриједности (енг. *Reward*) [1].

¹¹ <https://www.geeksforgeeks.org/dimensionality-reduction/>



Слика 2.9. Илустрација агента који учи уз подстицаје

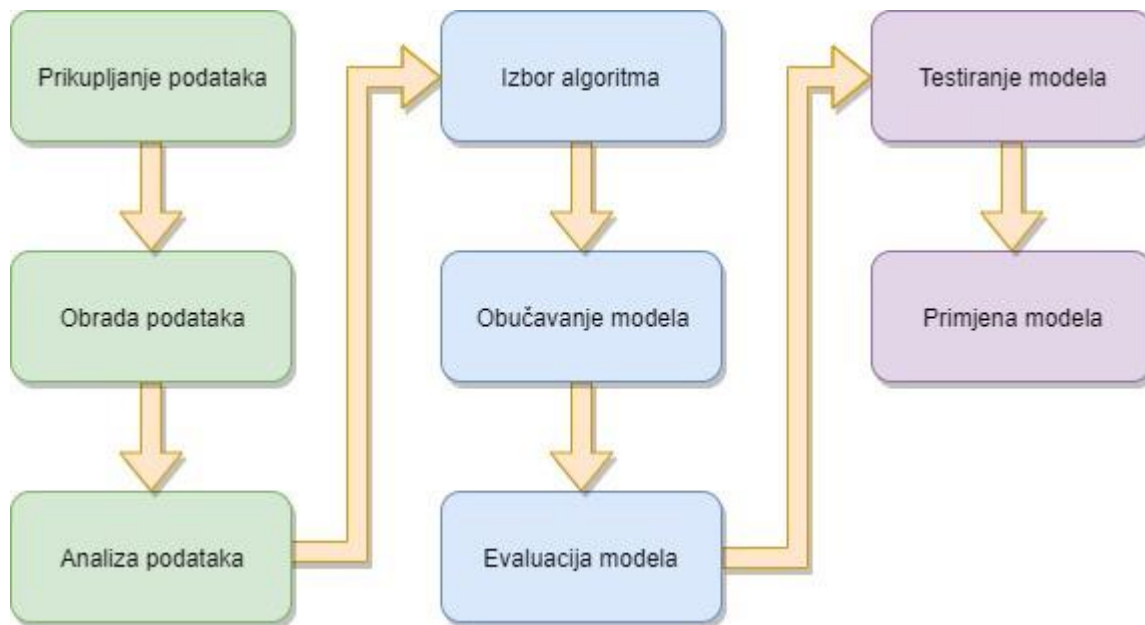
Исход учења је оптимална политика (енг. *optimal policy*), тј. пресликавање из стања окружења у низ акција које максимизирају награду или минимизују казну (слика 2.9). У пракси, максимална награда не постоји или је превише „скупа“, па се због тога обично узима награда која је довољна за рјешавање проблема [1].

Овакав облик учења је доминантан у индустрији игара и роботичи.

3. ПРОЦЕС МАШИНСКОГ УЧЕЊА

Развој модела машинског учења почиње са прикупљањем података за обучавање модела. Прикупљени подаци пролазе кроз фазу обраде у оквиру које се врши чишћење и трансформација података. Након ове фазе врши се анализа података у оквиру које је могуће додатно унаприједити скуп података кроз селекцију и/или трансформацију обиљежја. На основу резултата анализе података врши се избор алгоритма. Затим се врши обучавање модела. Обучени модел пролази кроз фазу евалуације перформанси. У овој фази се подешавају хиперпараметри¹² модела. На основу резултата евалуације врши се избор алгоритма и прелази се на његово тестирање. Резултати добијени у овој фази представљају перформансе модела. Уколико добијене перформансе задовољавају одређене норме, модел је спреман да се примијени у реалном окружењу [3] [4]. Кораци процеса машинског учења су приказани на слици 3.1.

Да би се одабрао најбољи модел, потребно је кроз одређене фазе проћи више пута. На тај начин се добијају различити модели истог типа алгоритма те на основу евалуације сваког од њих се одређује најбољи.



Слика 3.1. Кораци процеса машинског учења

У наредним потпоглављима описани су кораци машинског учења уз објашњење како је сваки од тих корака изгледао приликом израде практичног дијела рада.

¹² Параметри који се дефинишу прије обучавања и чије вриједности није могуће мијењати за вријеме обучавања

3.1. Прикупљање података

Подаци су неопходни за тренирање (обучавање), валидацију и тестирање модела. На интернету постоје јавно доступне колекције података¹³. Већина колекција је означена и припремљена што олакшава процес машинског учења (након прикупљања података прелази се на избор алгоритма). Поред тога, постоје подаци доступни посредством *web API*-а¹⁴, као и тржиште за куповину података¹⁵. Остали извори података укључују базе података, датотеке, сензоре и сл. Ови подаци се називају сировим подацима (енг. *row data*).

| Rijeka Vrbas HS Banja Luka vodostaj 8.9.2016-24.1.2019 | | | | | | | |
|--|----------|-----|----|--|--|--|--|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| 09-08-16 | 13:30:00 | 107 | cm | | | | |
| 09-08-16 | 14:00:00 | 107 | cm | | | | |
| 09-08-16 | 14:30:00 | 107 | cm | | | | |
| 09-08-16 | 15:00:00 | 106 | cm | | | | |
| 09-08-16 | 15:30:00 | 107 | cm | | | | |
| 09-08-16 | 16:00:00 | 108 | cm | | | | |
| 09-08-16 | 16:30:00 | 108 | cm | | | | |
| 09-08-16 | 17:00:00 | 107 | cm | | | | |
| 09-08-16 | 17:30:00 | 104 | cm | | | | |
| 09-08-16 | 18:00:00 | 102 | cm | | | | |
| 09-08-16 | 18:30:00 | 101 | cm | | | | |
| 09-08-16 | 19:00:00 | 100 | cm | | | | |

Слика 3.2. Подаци о ријечном водостају са хидрометеоролошке станице у Бањалуци

Подаци коришћени за израду рада прикупљени су са хидрометеоролошких станица у сливу ријеке Врбас од стране хидрометеоролошког савеза Републике Српске¹⁶.

¹³ Kaggle (<https://www.kaggle.com/>), UCI Machine Learning Repository (<http://mlr.cs.umass.edu/ml/>)

¹⁴ ProgrammableWeb (<https://www.programmableweb.com>)

¹⁵ Qlik DataMarket (<https://www.qlik.com/us/products/qlik-data-market>)

¹⁶ <https://rhmmzrs.com>

| Rijeka Vrbas HS Banja Luka protok 8.9.2016-24.1.2019 | | | | | | | |
|--|----------|-----------|---------|-----------------|--------|-----------------|------|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| 09-08-16 | 60,367 | 1,328,070 | 132,000 | 23:30:00 A30P30 | 28,450 | 19:30:00 A30P30 | m3/s |
| 09-08-16 | 13:30:00 | 34,720 | m3/s | | | | |
| 09-08-16 | 14:00:00 | 34,480 | m3/s | | | | |
| 09-08-16 | 14:30:00 | 34,360 | m3/s | | | | |
| 09-08-16 | 15:00:00 | 34,180 | m3/s | | | | |
| 09-08-16 | 15:30:00 | 34,300 | m3/s | | | | |
| 09-08-16 | 16:00:00 | 34,800 | m3/s | | | | |
| 09-08-16 | 16:30:00 | 35,000 | m3/s | | | | |
| 09-08-16 | 17:00:00 | 34,640 | m3/s | | | | |
| 09-08-16 | 17:30:00 | 32,460 | m3/s | | | | |
| 09-08-16 | 18:00:00 | 30,410 | m3/s | | | | |

Слика 3.3. Подаци о протоку воде са хидрометеоролошке станице у Бањалуци

Добијени су у виду Excel табеле са информацијама о термину мјерења, нивоу ријечног водостаја, протоку воде (слике 3.2. и 3.3).

3.2. Припрема података

Припрема података представља један од најважнијих корака у процесу машинског учења. Без овог корака модел би имао велике потешкоће приликом обучавања и веома лоше перформансе приликом евалуације, тј. модел би био неупотребљив. Из тог разлога је потребно доста времена уложити у овај корак како би се подаци што боље припремили за процес обучавања.

Проблеми који се јављају у процесу припреме података су:

1. Недостајући подаци (енг. *Missing data*)
2. Подаци са великом количином шума (енг. *Noisy data*)
3. Неконзистентни подаци (енг. *Inconsistent data*).

| Rijeka Vrbas HS Banja Luka protok 8.9.2016-24.1.2019 | | | | | | | |
|--|----------|-----------|---------|-----------------|--------|-----------------|------|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| 09-08-16 | 60,367 | 1,328,070 | 132,000 | 23:30:00 A30P30 | 28,450 | 19:30:00 A30P30 | m3/s |
| 09-08-16 | 13:30:00 | 34,720 | m3/s | | | | |
| 09-08-16 | 14:00:00 | 34,480 | m3/s | | | | |

Слика 3.4. Примјер ирелевантних вриједности

Припрема података у склопу практичног дијела рада је обухватала сљедеће активности:

- Форматирање података у облик погодан за обраду
 - Уклањање редова са ирелевантним вриједностима (слика 3.4)
 - Форматирање датума (слика 3.5)
 - Извлачење вриједности измјереног протока из различитих колона у оквиру Excel табеле (слика 3.5)
- Допуњавање недостајућих вриједности

| | | | | |
|------------|----------|--------|--------|------|
| 11/20/2018 | 09:00:00 | 28,650 | m3/s | |
| 11/20/2018 | 09:30:00 | 28,880 | m3/s | |
| 11/20/2018 | 10:00:00 | 30,050 | m3/s | |
| 09-08-16 | 13:30:00 | A30P30 | 34,720 | m3/s |
| 09-08-16 | 14:00:00 | A0P0 | 34,480 | m3/s |
| 09-08-16 | 14:30:00 | A30P30 | 34,360 | m3/s |

Слика 3.5. Примјер неконзистентног формата датума и различитих позиција ријечног протока

Недостајући подаци

Недостајући подаци представљају веома озбиљан проблем у процесу припреме података. Приликом обраде таквих података потребно је водити рачуна о томе да ли су подаци намјерно изостављени или не. На примјер, особе са високим приходима често не наводе своје износе у анкетама, жене често не наводе своју тежину или године старости и сл. Са друге стране, аутоматске хидрометеоролошке станице прате одређене параметре сваких 30 минута и уколико се у записима појави мјерење којем недостаје одређени параметар, могуће је разматрати начине како да се недостајући податак обради.

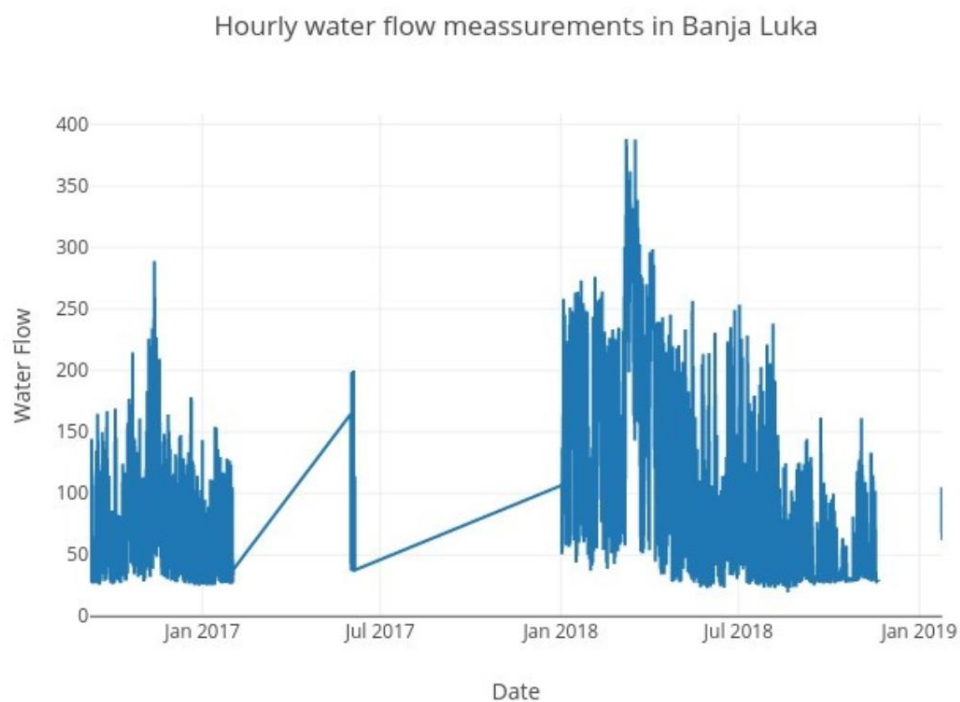
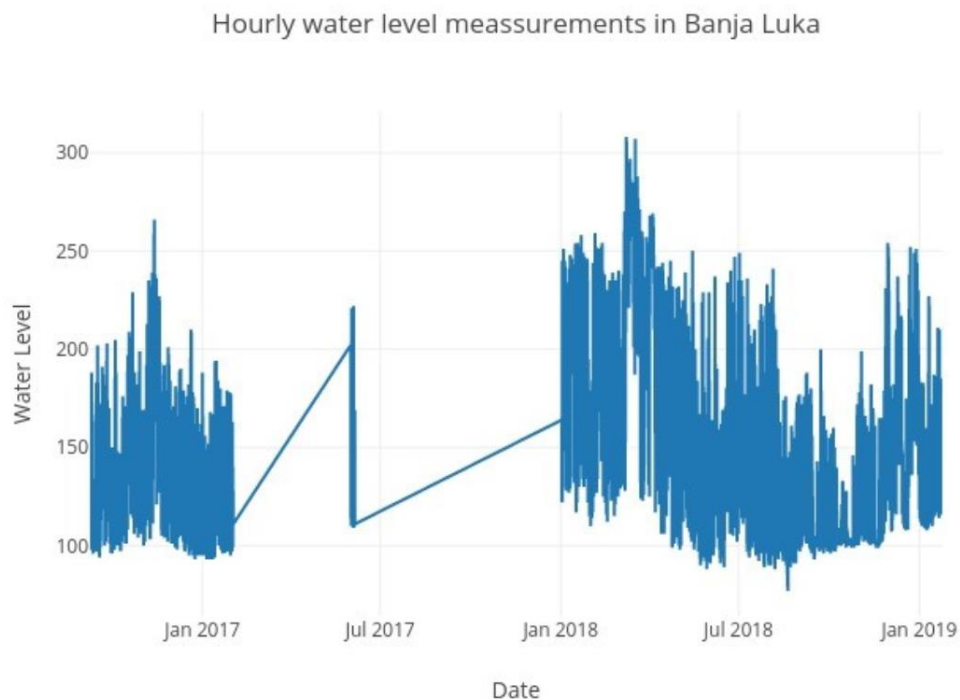
Начин обраде зависи од врсте проблема. Уколико постоји зависност између мјерења (нпр. временска зависност), тада би уклањање таквих записа утицало на перформансе модела. Уколико не постоји зависност између записа, онда је брисање записа један од начина да се он обради. Међутим, уколико у скупу података постоји велики број узорака са недостајућим подацима, тада би њихово уклањање знатно смањило величину скупа података, што би резултовало лошијим перформансама модела. Алтернатива брисању је допуњавање недостајућих података. Постоји велики број метода за допуњавање недостајућих података чија употреба такође зависи од врсте проблема који модел треба да ријеши (нпр. да ли је вриједност нумеричка/категоричка/ординална, колико је тај податак повезан са проблемом који се рјешава и сл) [5].

Подаци између којих постоји временска зависност се називају временска серија (енг. *Time series*).

Методе допуњавања карактеристичне за временску серију су [5]:

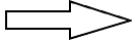
- Задња опсервација пропагирана унапријед / Сљедећа опсервација пропагирана уназад (енг. *Last observation carried forward – LOCF / Next observation carried backward - NOCB*) подразумева да се недостајући податак замијени са задњом/првом наредном опсервацијом
- Линеарна интерполација
- Сезонско прилагођавање и линеарна комбинација
- Допуњавање вриједности на основу аритметичке средине/медијане/мода.

Подаци са хидрометеоролошких станица представљају временску серију. У добијеним подацима постоји велики број недостајућих вриједности. На слици 3.6. је приказан проблем недостајућих вриједности, гдје за период од јануара 2017. до јануара 2018. постоји мали број мјерења. Такође, са слике се види да мјерења за ријечни проток нису забиљежена у јануару 2019. године.

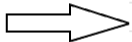


Слика 3.6. Водостај и проток ријеке Врбас прије допуњавања

Да би ријешили проблем недостајућих вриједности искоришћене су методе LOCF и NOCB (слике 3.7. и 3.8), при чему је пропагација урађена само за једну недостајућу вриједност како би грешка проузрокована допуњавањем била што мања. Међутим, број недостајућих података је и даље био велики. Више од 50% узорака је недостајало.

| | | | | | | |
|---------------------|-----|------|---|---------------------|-----|------|
| 1971-12-29 00:00:00 | 134 | 65.8 |  | 1971-12-29 00:00:00 | 134 | 65.8 |
| 1971-12-30 00:00:00 | 130 | 61 | | 1971-12-30 00:00:00 | 130 | 61 |
| 1971-12-31 00:00:00 | 121 | 51.1 | | 1971-12-31 00:00:00 | 121 | 51.1 |
| 1972-01-01 00:00:00 | 136 | | | 1972-01-01 00:00:00 | 136 | 51.1 |

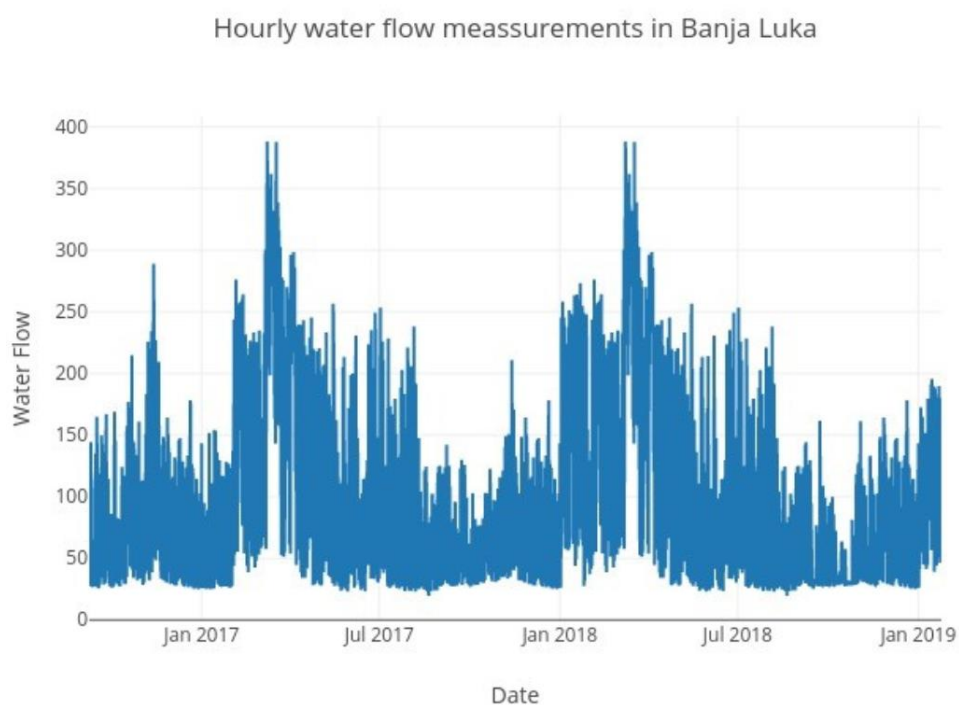
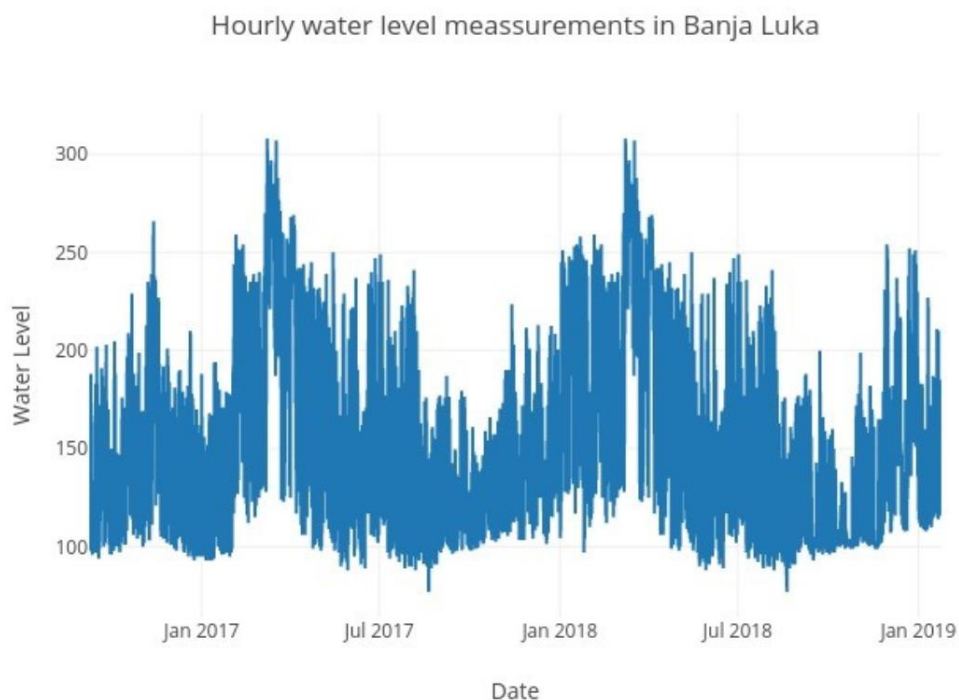
Слика 3.7. Задња опсервација пропагирана унапријед

| | | | | | | |
|---------------------|-----|------|---|---------------------|-----|------|
| 1971-12-22 00:00:00 | 144 | |  | 1971-12-22 00:00:00 | 144 | |
| 1971-12-23 00:00:00 | 150 | | | 1971-12-23 00:00:00 | 150 | 73 |
| 1971-12-24 00:00:00 | 140 | 73 | | 1971-12-24 00:00:00 | 140 | 73 |
| 1971-12-25 00:00:00 | 146 | 81.4 | | 1971-12-25 00:00:00 | 146 | 81.4 |
| 1971-12-26 00:00:00 | 141 | 74.4 | | 1971-12-26 00:00:00 | 141 | 74.4 |

Слика 3.8. Сљедећа опсервација пропагирана уназад

Како би се ријешо проблем недостајућих вриједности урађено је сљедеће:

Прво су допуњени недостајући термини, а затим, под претпоставком да је вриједност протока ријеке и нивоа водостаја у тренутку t слична вриједностима у тренуцима $t \pm n$, гдје n представља период од годину дана, недостајуће вриједности су допуњене средњом вриједношћу у односу на доступна мјерења. На примјер, вриједност протока 07.07.2017. године у 12:00 часова је непозната, допуњена вриједност је средња вриједност протока за 07.07.2016. године у 12:00 часова и 07.07.2018. године у 12:00 часова. Затим су над таквим подацима примијењене методе LOCF и NOCB. Резултат допуњавања је приказан на слици 3.9.



Слика 3.9. Водостај и проток ријеке Врбас- након допуњавања

Са слике се види да су допуњени подаци попримили тренд¹⁷ (енг. *trend*) и сезоналност¹⁸ (енг. *seasonality*) који карактерише период од јануара 2018. до јануара

¹⁷ <https://stats.oecd.org/glossary/detail.asp?ID=6692>

¹⁸ <https://stats.oecd.org/glossary/detail.asp?ID=6695>

2019. С обзиром да су за допуњавање искоришћена мјерења из 2016. и 2018. године, као посљедица се уочавају одступања забиљежена у марту 2018. која су пренесена на мјерења у марту 2017. На тај начин је уведена одређен ниво грешке у податке, који ће резултовати лошијим перформансама модела када се искористи у продукцији.

3.3. Анализа података

Анализа података је процес откривања образаца, уочавања аномалија, тестирања хипотезе и провјере претпоставки, уз помоћ статистичких алата и визуелне репрезентације података.

У оквиру практичног дијела рада извршена је сљедећа анализа:

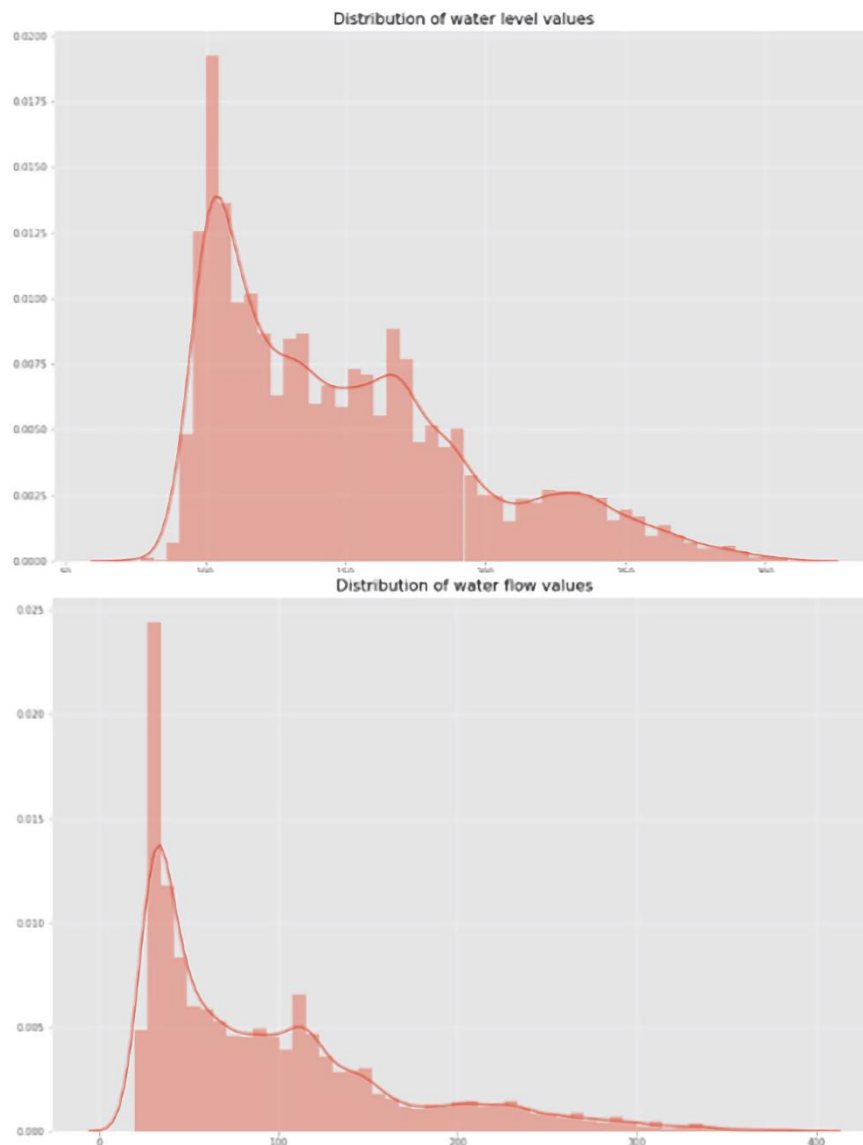
- Статистичка анализа појединачних обиљежја
- Утврђивање корелације између обиљежја
- Утврђивање балансираности скупа података.

Статистичка анализа подразумијева увид у скуп података кроз различите статистичке методе (слика 3.10 и 3.11).

| | Hour | Minute | DayOfWeek | Quarter | Month | Year | DayOfYear | DayOfMonth | WeekOfYear | WaterLevel | WaterFlow | FloodLevelCategory |
|-------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------------|
| count | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41338.000000 | 41337.000000 |
| mean | 11.502274 | 14.998549 | 2.992670 | 2.636606 | 6.913977 | 2017.341647 | 195.089990 | 15.805748 | 28.186584 | 151.158631 | 94.794391 | 0.545710 |
| std | 6.919273 | 15.000181 | 1.997499 | 1.159687 | 3.623225 | 0.738458 | 111.020123 | 8.728045 | 15.796892 | 47.048714 | 70.844054 | 0.731158 |
| min | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 2016.000000 | 1.000000 | 1.000000 | 1.000000 | 77.000000 | 19.540000 | 0.000000 |
| 25% | 6.000000 | 0.000000 | 1.000000 | 2.000000 | 4.000000 | 2017.000000 | 96.000000 | 8.000000 | 14.000000 | 111.000000 | 36.460000 | 0.000000 |
| 50% | 12.000000 | 0.000000 | 3.000000 | 3.000000 | 7.000000 | 2017.000000 | 207.000000 | 16.000000 | 30.000000 | 141.000000 | 73.260000 | 0.000000 |
| 75% | 17.000000 | 30.000000 | 5.000000 | 4.000000 | 10.000000 | 2018.000000 | 294.000000 | 23.000000 | 42.000000 | 178.000000 | 126.000000 | 1.000000 |
| max | 23.000000 | 30.000000 | 6.000000 | 4.000000 | 12.000000 | 2019.000000 | 366.000000 | 31.000000 | 52.000000 | 308.000000 | 388.200000 | 3.000000 |

Слика 3.10. Статистичка анализа података

Са слике 3.11. видимо да је средња вриједност за водостај и проток већа од медијане (50% - 50. перцентил) и да постоји велика разлика између 75. перцентиала и максималне вриједности. На основу тога можемо закључити да у датом скупу података постоје вансеријски узорци (енг. *outliers*). Ти узорци представљају поплаве.



Слика 3.11. Расподјела вриједности за водостај и проток

Основни проблем података је недовољна количина обиљежја у односу на циљну промјенљиву (ниво водостаја). Недостатак информација у подацима се на процес учења одражава високом грешком током евалуације и тестирања. Из тог разлога се током анализе одређује матрица корелације (енг. *Correlation Matrix*) како би се утврдила веза између обиљежја. Уколико обиљежје није јако корелисано са циљном промјенљивом мале су шансе да ће оно имати значаја током предвиђања. У случају линеарног модела користи се Пирсонов коефицијент корелације¹⁹, док се за нелинеарне моделе користи Спирманов коефицијент корелације²⁰.

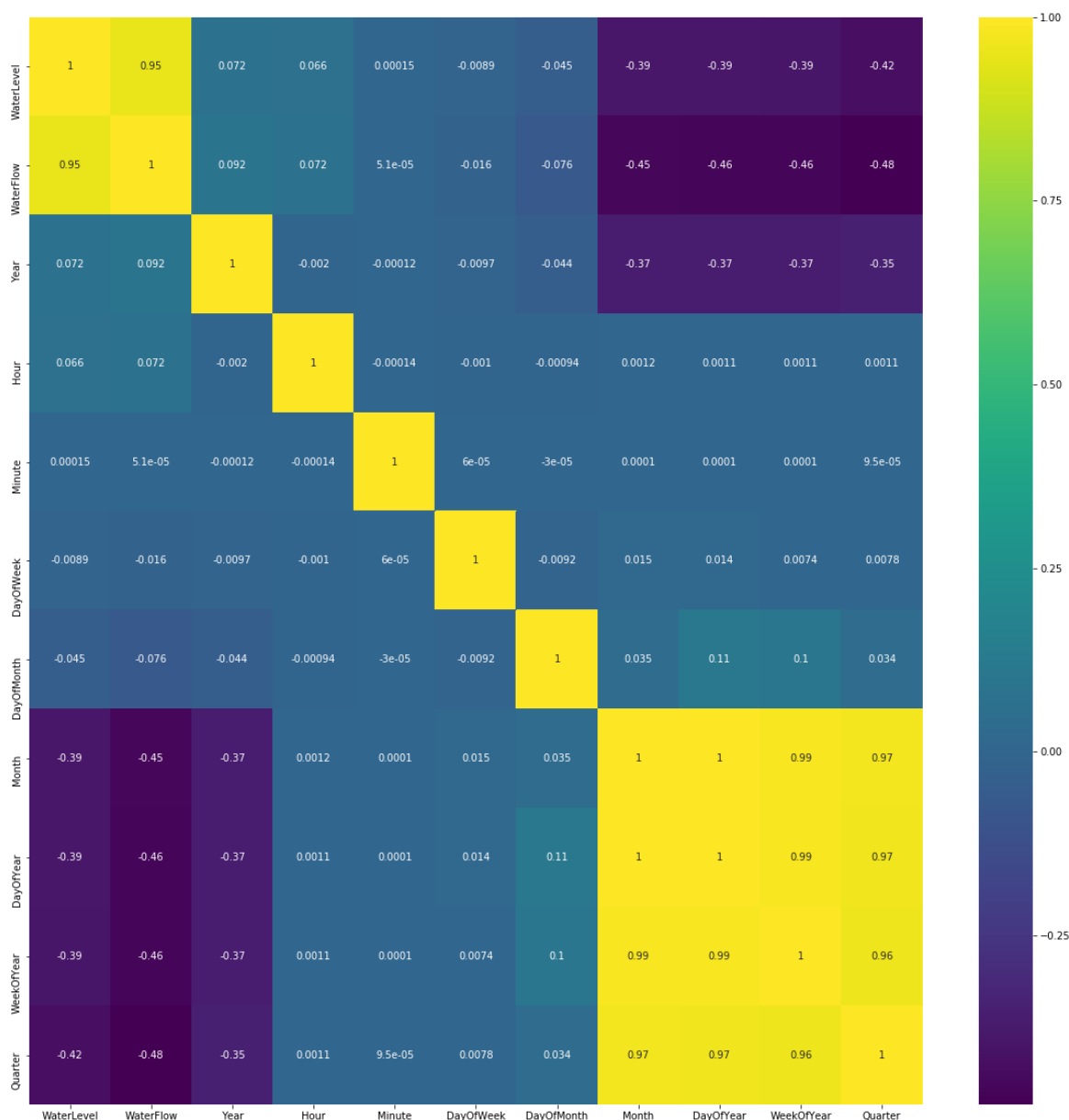
¹⁹ https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

²⁰ https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

На слици 3.12. је приказана корелација између обиљежја у односу на водостај. Сваки запис у матрици представља корелацију два обиљежја у односу на водостај. Бројеви и боје представљају степен корелације. Уколико између два обиљежја постоји јака корелација њена вриједност тежи јединици или минус јединици у зависности од тога да ли је она позитивна или негативна. Нијансе жуте боје у матрици корелације указују на јаку позитивну корелацију између обиљежја, док нијансе љубичасте боје указују на јаку негативну корелацију.

Са слике се види да постоји јака корелација између протока и водостаја, док је корелација године, минута, дана у седмици и дана у години са водостајем занемарујућа. То значи да та обиљежја можемо уклонити. Иако је степен корелације водостаја и сата мали, ово обиљежје је остављено јер је показано да оно има утицај на резултате модела (слика 6.2).

У случају класификације, корисно је провјерити балансираност класа. Уколико су класе небалансиране, често долази до грешака током евалуације модела. Метрика која се често користи за утврђивање перформанси класификатора је тачност. Тачност представља удио тачно класификованих узорака у односу на укупан број узорака. Међутим, употреба ове метрике у случају небалансираног скупа података даје погрешан увид у перформансе модела. На примјер, уколико једној класи припада 95% узорака, а другој 5%, тачност од 0.95 која на први поглед изгледа импресивна може бити постигнута тако што класификатор научи да све узорке третира као да припадају већинској класи. У таквом случају класификатор није ни потребан. Додатни проблем се јавља уколико је битно да класификатор тачно класификује узорке из мањинске класе. У таквим случајевима се умјесто тачности користе метрике као што су одзив и F1 мјера (обе метрике су објашњене у потпоглављу „Евалуација модела“) које узимају у обзир и број нетачно класификованих узорака мањинске класе, те на тај начин дају бољу интерпретацију модела. Како би се одабрала метрика која адекватно приказује перформансе модела потребно је посебну пажњу посветити анализи података.



Слика 3.12. Матрица корелације у односу на водостај

Методe које се често користе за рјешавање проблема небалансираних класа су:

1. Прикупљање додатних узорака мањинске класе
2. Промјена метрике за евалуацију (више о метрикама за евалуацију је дато у потпоглављу „Евалуација модела“)
3. Реизбор (енг. *Resampling*)
 - а. Повећање броја узорака мањинске класе (енг. *oversampling*)
 - б. Смањење броја узорака већинске класе (енг. *undersampling*)
4. Употреба модела са казном (енг. *Penalized Models*).

Прикупљање нових узорака представља најбоље рјешење, јер тада није потребно примјењивати друге методе које могу нарушити перформансе модела. Проблем је у

томе што често додатни узорци нису доступни. Примјер оваког проблема је рад са временским серијама у којима није могуће поновити мјерење због утицаја временске компоненте. У таквим случајевима се морају користити алтернативне методе допуњавања.

Реизбор представља добро рјешење у случају да не постоји међузависност између узорака. *Oversampling* се ради тако што се узорци мањинске класе копирају или креирају нови узорци, чиме се изједначава однос мањинске и већинске класе. *Undersampling* се ради тако што се одређен број узорака из већинске класе уклања. Узорци се могу уклањати насумично или помоћу напреднијих техника или користећи алгоритме за кластеризацију, при чему треба водити рачуна да број уклоњених узорака не буде превелик, јер то може довести до недовољно прилагођеног модела (енг. *underfitting*).

Употреба модела са казном подразумева да се, приликом тренинга, грешка услед нетачно класификованог узорака додатно кажњава. На тај начин модел посвећује више „пажње“ на узорке из мањинске класе.

У оквиру практичног дијела рада било је потребно обучити класификатор за ријечни водостај. У почетку је проблем посматран као бинарна класификација, да ли се поплава догодила (1) или није (0). С обзиром да су поплаве ријетки догађаји, број узорака класе 0 је био далеко већи од броја узорака класе 1.

Први корак је била употреба модела са казном. Додијељене тежине су израчунате тако да је тежина за сваку класу обрнуто пропорционална фреквенцији појављивања у скупу података:

$$C[i] = \frac{N}{nM[i]}$$

gdje su:

$C[i]$ – тежина за i -ту класу

N – број узорака

n – број класа

$M[i]$ – број појављивања i -те класе.

Међутим, перформансе модела са тежинама након евалуације су биле лоше. Како би се постигле боље перформансе извршена је подјела већинске класе у три подкласе (слика 3.13) којима су додијељене тежине на претходно описан начин. Овакав приступ је дао доста боље резултате (резултати добијени овим приступом су приказани у шестом поглављу).

```
0.0    24310
1.0    11890
2.0     4743
3.0      394
Name: FloodLevelCategory, dtype: int64
```

Слика 3.13. Број узорака сваке класе након подјеле већинске класе

3.4. Избор алгоритма

Избор алгоритма зависи од врсте проблема који се рјешава (класификација/регресија), карактеристика скупа обиљежја (тип обиљежја, степен хомогености типова и опсега вриједности обиљежја, степен међузависности - корелације обиљежја) и обима података који су на располагању.

Узевши у обзир наведене факторе, одабрани су сљедећи модели: SVM, LightGBM (алгоритам базиран на градијентном појачавању) и неуралне мреже. Наведени модели су искоришћени за предвиђање ријечног водостаја и за класификацију нивоа водостаја.

3.5. Обучавање модела

Прије него што се пређе на фазу обучавања, неопходно је подијелити податке у три дисјунктна подскупа: *тренинг скуп*, *валидациони скуп* и *тестни скуп*.

Тренинг скуп се користи за обучавање модела, валидациони скуп се користи за одређивање хиперпараметара модела у циљу побољшања перформанси, док се тестни скуп користи за коначну евалуацију модела. Веома је важно да ова три скупа буду дисјунктна, јер би у супротном модел током обучавања могао да „запамти“ узорке који се појављују у тренинг и тест скуповима и на тај начин добије боље резултате током евалуације перформанси.

Подјела зависи од броја узорака. У пракси се подаци често дијеле према омјеру 60/20/20% на тренинг, валидациони и тестни скуп, респективно. Уколико је број узорака велики, реда стотина хиљада и више, тренинг скуп обично чини 90-99% од укупног броја узорака.

У општем случају, прије подјеле је потребно насумично промијенити редослијед података. На тај начин се смањује вјероватноћа да модел постане превише прилагођен (енг. *overfitting*). Другим ријечима, модел ће бити у стању да боље генерализује. Уколико се не изврши промјена редослиједа узорака постоји велика вјероватноћа да ће скупови након подјеле имати различите расподеле података, што ће дати лоше резултате у току евалуације. На примјер, класификатор који је обучен за класификацију објеката на сликама ће имати лоше перформансе у току евалуације уколико се објекти који се јављају у тренинг скупу не појаве у валидационом/тестном скупу, и обрнуто. Насумични избор узорака осигурава да тренинг, валидациони и тестни скуп задрже расподелу података која карактерише читав скуп.

Међутим, уколико је скуп података временска серија овакав приступ није могућ, јер би то значило да не постоји међузависност између узорака. Умјесто тога, подјела података се врши узимајући у обзир временску компоненту.

Након припреме и анализе, скуп података се састоји од 41338 узорака и представља мјерења у период од 08.09.2016. до 24.01.2019. године. Подаци су подијељени у три скупа према односу 80/10/10%.

Хиперпараметри за моделе базиране на SVMу и LightGBMу су одређени помоћу *Hyperopt*-а²¹. *Hyperopt* је *Python* библиотека за оптимизацију хиперпараметара. Базирана је на Бајесовој оптимизацији²² (енг. *Bayesian optimization*). За њено коришћење потребно је дефинисани простор претраживања хиперпараметара, функцију циља, као и алгоритам који ће се користити за претрагу. Резултат претраживања је *Python dictionary* објекат који се састоји од кључева који представљају назив хиперпараметра и његову оптималне вриједности.

```
Hyperopt estimated optimum {'C': 73.07203333068946, 'epsilon': 1.0048809046572094}
Hyperopt estimated optimum {'C': 0.016018759178962227}
```

Слика 3.14. Хиперпараметри SVM модела (Класификација, Регресија)

Хиперпараметри добијени помоћу ове библиотеке су приказани на сликама 3.14. и 3.15.

| | |
|--|---|
| <pre>{'device': 'cpu', 'is_unbalance': (True,), 'learning_rate': 0.09031673772304524, 'min_data_in_leaf': 512, 'num_class': (4,), 'num_leaves': 511, 'objective': 'multiclassova'}</pre> | <pre>{'bagging_fraction': 0.8650449752569586, 'device': 'cpu', 'feature_fraction': 0.7878056863611946, 'learning_rate': 0.05, 'max_bin': 498, 'min_data_in_leaf': 20, 'num_iterations': 100.0, 'num_leaves': 4095, 'objective': 'regression'}</pre> |
|--|---|

Слика 3.15. Хиперпараметри LightGBM модела (Класификација, Регресија)

Архитектура неуралне мреже је инспирисана радовима [6] [7], док су хиперпараметри одређени помоћу *HParams*²³ библиотеке чија је употреба слична употреби *HyperOpt* библиотеке. Оптимални хиперпараметри су приказани на слици 3.16.

```
# ===== CLASSIFICATION =====
NUM_CLASSES = len(np.unique(yc_train.values))
BATCH_SIZE = 120
DENSE_LAYERS = 3
DROPOUT_RATE = 0.037878
# ===== REGRESSION =====
CONV_LAYERS = 5
CONV_KERNEL_SIZE = 9
LSTM_LAYERS = 2
DENSE_LAYERS = 1
BATCH_SIZE = 120
WINDOW_SIZE = 96
SHIFT = 1
PADDING = "CAUSAL"
ACTIVATION = "RELU"
```

Слика 3.16. Хиперпараметри модела неуралних мрежа (Класификација, Регресија)

²¹ <https://github.com/jaberg/hyperopt>

²² <https://arxiv.org/pdf/1807.02811.pdf>

²³ https://www.tensorflow.org/api_docs/python/tf/contrib/training/HParams?hl=en

3.6. Евалуација модела

Евалуација модела треба да покаже колико добро научени модел генерализује када му улаз представљају подаци које до тада није „видио“. Постоје различите технике за евалуацију као што су: евалуација помоћу скупа за тестирање, К-слојна унакрсна валидација (енг. *K-fold cross-validation*), евалуација помоћу скупа за тестирање (енг. *Hold-out validation*), евалуација помоћу скупова за валидацију и тестирање, угњевдена унакрсна валидација (енг. *Nested cross-validation*) [1].

Осим одабира технике за евалуацију, неопходно је одабрати и метрику за евалуацију. Метрика за евалуацију служи за квантификацију перформанси модела, тј. омогућава нам да перформансе модела изразимо у виду нумеричких вриједности. Избор метрике је веома важан корак, јер лоше одабрана метрика може приказати потпуно погрешно стање модела (примјер класификатора са 95% узорака из једне класе). Метрике за надгледано учење се могу подијелити у двије категорије:

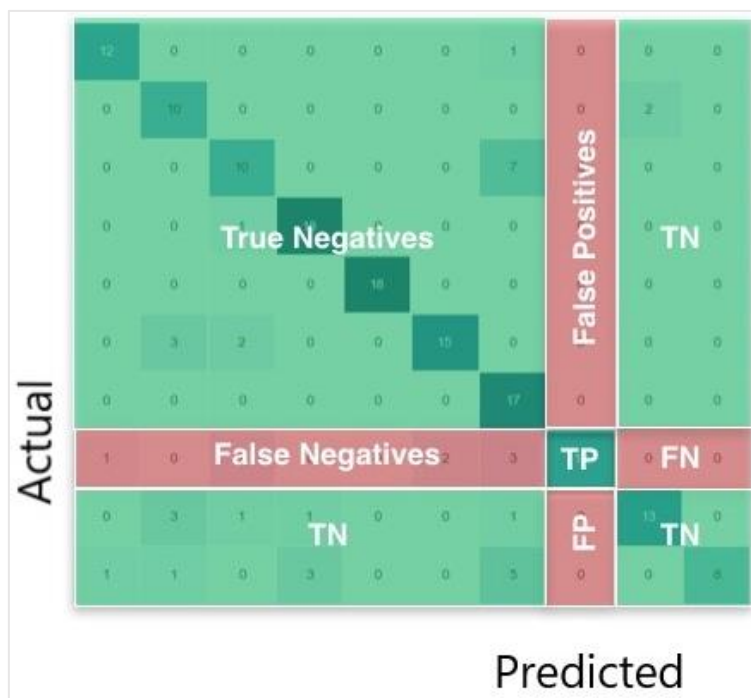
- Метрике за класификацију:
 - Тачност, Прецизност, Одзив (енг. *Classification Accuracy, Precision, Recall*)
 - Матрица конфузије (енг. *Confusion Matrix*)
 - Површина испод криве (енг. *Area Under the Curve*)
 - Ф-Мјера (енг. *F-Score*)
- Метрике за регресију:
 - Средњеквадратна грешка и њен коријен (енг. *Mean Square Error, Root Mean Square Error*)
 - Средња апсолутна грешка (енг. *Mean Absolute Error*)

Метрике за класификацију су добијене на основу матрице конфузије. То је матрица C чији елемент c_{ij} представља број елемената класе i који су класификовани у класу j . Дијагонални елементи означавају тачне класификације, док елементи који нису на дијагонали означавају грешке. У случају бинарне класификације, обично се једна класа назива позитивном, а друга негативном (слика 3.17).

| | Pozitivna (Predviđena klasa) | Negativna (Predviđena klasa) |
|--------------------------------------|---|---|
| Pozitivna (Stvarna klasa) | Stvarno pozitivna (TP) | Lažno negativna (FN) |
| Negativna (Stvarna klasa) | Lažno pozitivna (FP) | Stvarno negativna (TN) |

Слика 3.17. Матрица конфузије у случају бинарне класификације

Матрица конфузије у случају класификације у више класа је приказана на слици 3.18.



Слика 3.18. Матрица конфузије у случају класификације више класа²⁴

Стварно позитивни узорци су узорци које је модел класификовао као позитивне. Број таквих узорака је означен са TP (енг. *True Positive*). Стварно негативни узорци су узорци које је модел класификовао као негативне. Број таквих узорака је означен са TN (енг. *True Negative*). Лажно позитивни узорци су негативни узорци које је модел класификовао као позитивне. Број таквих узорака је означен са FP (енг. *False Positive*). Лажно негативни узорци су позитивни узорци које је модел класификовао као негативне. Број таквих узорака је означен са FN (енг. *False Negative*).

Тачност класификатора представља број тачно класификованих узорака у односу на укупан број узорака. У случају бинарне класификације тачност је дефинисана на следећи начин:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Одзив представља удио тачно позитивних узорака, тј. представља однос броја тачно класификованих позитивних узорака и укупног броја позитивних узорака:

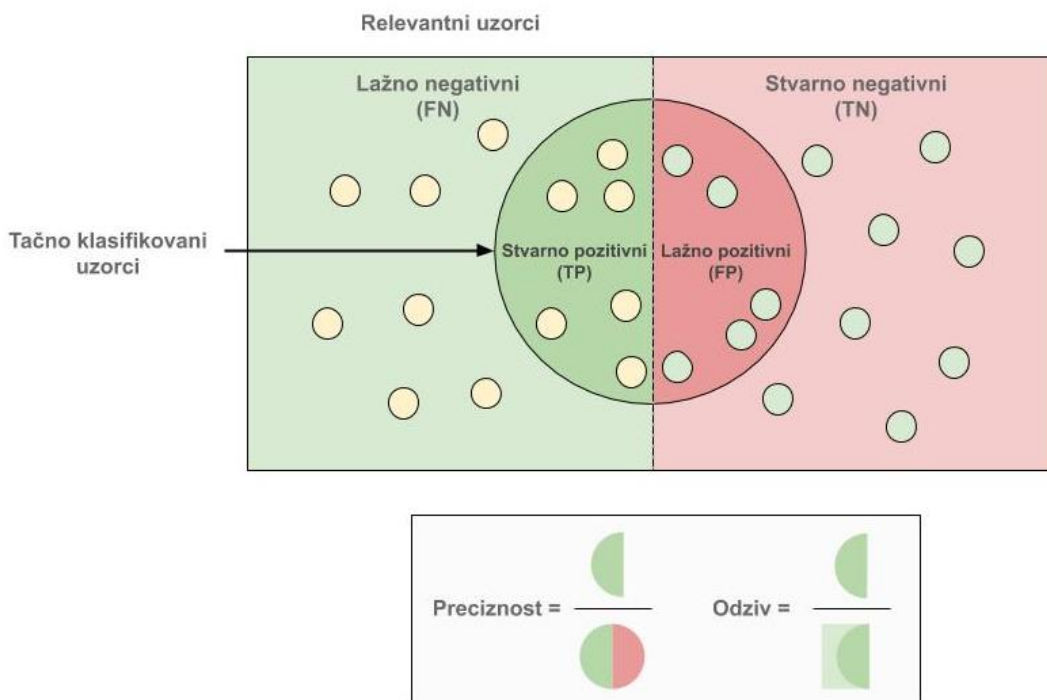
$$R = \frac{TP}{TP + FN}$$

²⁴<https://stackoverflow.com/questions/31324218/scikit-learn-how-to-obtain-true-positive-true-negative-false-positive-and-fal>

Прецизност представља однос броја тачно класификованих позитивних узорака и укупног броја узорака класификованих у позитивну класу:

$$P = \frac{TP}{TP + FP}$$

Прецизност и одзив су илустровани на слици 3.19.



Слика 3.19. Илустрација прецизности и одзива

Ф1-Мјера представља хармонијску средину прецизности и одзива:

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{R + P}$$

Метрика која се често користи у случају бинарне класификације са небалансираним класама је површина испод ROC криве. Површина испод ROC криве је удио парова узорака из двије класе, таквих да је узорку из негативне класе придружен мањи скор него узорку из позитивне класе, у укупном броју парова узорака из различитих класа [1].

Порвшина испод ROC криве се дефинише на следећи начин:

$$AUC = \frac{1}{N_1 N_2} \left(\sum_{i \in C_2} r(i) - \frac{N_2(N_2 + 1)}{2} \right)$$

Гдје су:

C_1, C_2 скупови узорака из различитих класа

$N_1 = |C_1|$

$N_2 = |C_2|$

$r(i)$ функција која сваком узорку i додјељује ранг у сортираном низу узорака у односу на вриједност модела f

Средња квадратна грешка (MSE) се рачуна као сума квадрата разлике између стварне вриједности и излаза модела у односу на број узорака:

$$MSE = \frac{1}{N} \sum_{i=0}^N (y_i - \hat{y}_i)^2$$

Ова метрика је корисна када се у подацима појављују вансеријски узорци који су релевантни, тј. нису последица грешке/шума у подацима. Да би се ова метрика изразила на истој скали, као и циљна промјенљива, рачуна се њен коријен (RMSE).

Због јаког утицаја вансеријских узорака на ову грешку и због потребе да се грешка изрази у односу на вриједност циљне промјенљиве, користи се средња апсолутна грешка:

$$MAE = \frac{1}{N} \sum_{i=0}^N |y_i - \hat{y}_i|$$

Приликом израде практичног дијела рада, за евалуацију перформанси су одабране следеће метрике:

- Класификација: тачност, прецизност, одзив, F1-мјера
- Регресија: средња квадратна грешка и средња апсолутна грешка.

3.7. Тестирање модела

Након одабира оптималних хиперпараметара, модел се обучава на тренинг и валидационим скупом, а затим се тестира на тестном скупу података. Добијени резултати представљају стварне перформансе модела.

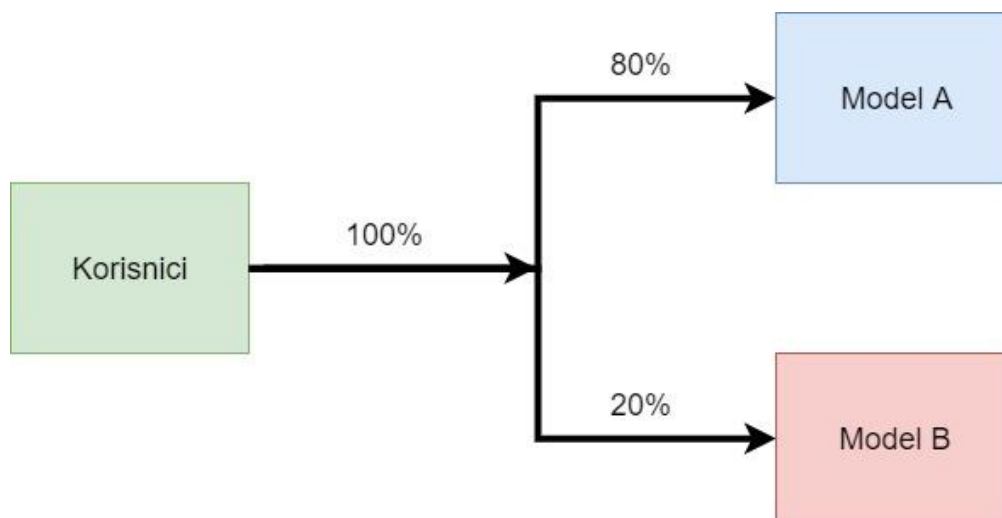
Резултати модела коришћених у овом раду су приказани у шестом поглављу.

3.8. Примјена модела

Примјена модела подразумијева његову употребу у стварном свијету. У овом кораку модел врши предикције на основу података добијених у реалном времену. На примјер, уколико је модел истрениран за детекцију преваре приликом банкарских трансакција, модел врши предикцију на основу записа трансакција које су се догодиле у датом тренутку.

Са стране наручиоца, ово представља најважнији корак, јер се у њему показује да ли (нови) модел пружа корист у виду квалитетнијих услуга, већих прихода, и сл. Међутим, метрике кориштене за вријеме евалуације често не одговарају кључним индикаторима успјешности (енг. *Key Performance Indicator – KPI*)²⁵. Због тога се за евалуацију перформанси модела у пракси користе методе као што су А/Б тестирање (енг. *A/B Testing*).

А/Б тестирање подразумијева поређење тренутног модела – модел А (енг. *controlled model*) са новим – модел Б (енг. *challenger model*). Поређење се врши на основу статистичких тестова у којима се нулта хипотеза (енг. *null hypothesis*) пореди са алтернативном хипотезом (енг. *alternative hypothesis*). Нулта хипотеза подразумијева да модел Б не доприноси промјени кључне метрике (KPI), док алтернативна хипотеза подразумијева да модел Б доприноси промјени кључне метрике. Циљ А/Б тестирања је да се одреди да ли постоји статистички значајна промјена у кључној метрици. За вријеме тестирања, корисници су подијељени у двије групе при чему једна група користи модел А, а друга модел Б (слика 3.20), врши се праћење кључних метрика и на основу статистичког теста доноси закључак [8] [9].



Слика 3.20. Примјер А/Б тестирања

Примјена модела у оквиру практичног дијела рада подразумијева преузимање података о тренутном нивоу водостаја и ријечног протока за одређену станицу, на

²⁵KPI су кључни показатељи успјеха (или неуспјеха) и мјерила која подржавају пословну стратегију компаније и успјешно пословање и одражавају мјеру пословног успјеха и одрживости пословне стратегије компаније

основу којих модел врши предикцију нивоа водостаја у периоду од наредних шест сати. За сваку предикцију је одређен ниво опасности. У случају да модел предвиди опасан ниво водостаја, заказује се емитовање путем *YouTube*-а. За емитовање је искоришћен дрон DJI Mavic Air који је снимао станицу за коју је предвиђена опасност.

С обзиром да дрон не посједује подршку за аутоматско заказивање лета, тај корак није било могуће имплементирати у изradi. Умјесто тога, дефинисана је путања летења коју је било потребно учитати ручно у апликацију за управљање дроном, те је на тај начин имплементирано емитовање надзора подручја за које је предвиђен повишен водостај.

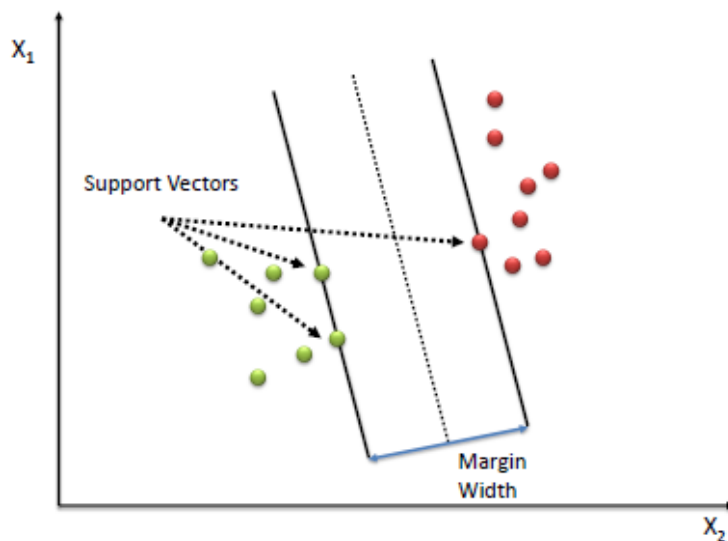
4. АЛГОРИТМИ ЗА КЛАСИФИКАЦИЈУ И РЕГРЕСИЈУ

У овом поглављу су описани алгоритми коришћени приликом израде практичног дијела рада. Описане су основне карактеристике сваког алгоритма као и његова употреба у регресионим, односно класификационим проблемима.

4.1. SVM (Support Vector Machines)

SVM припадају класи линеарних класификатора чија је граница одлучивања у линеарном простору обиљежја [10].

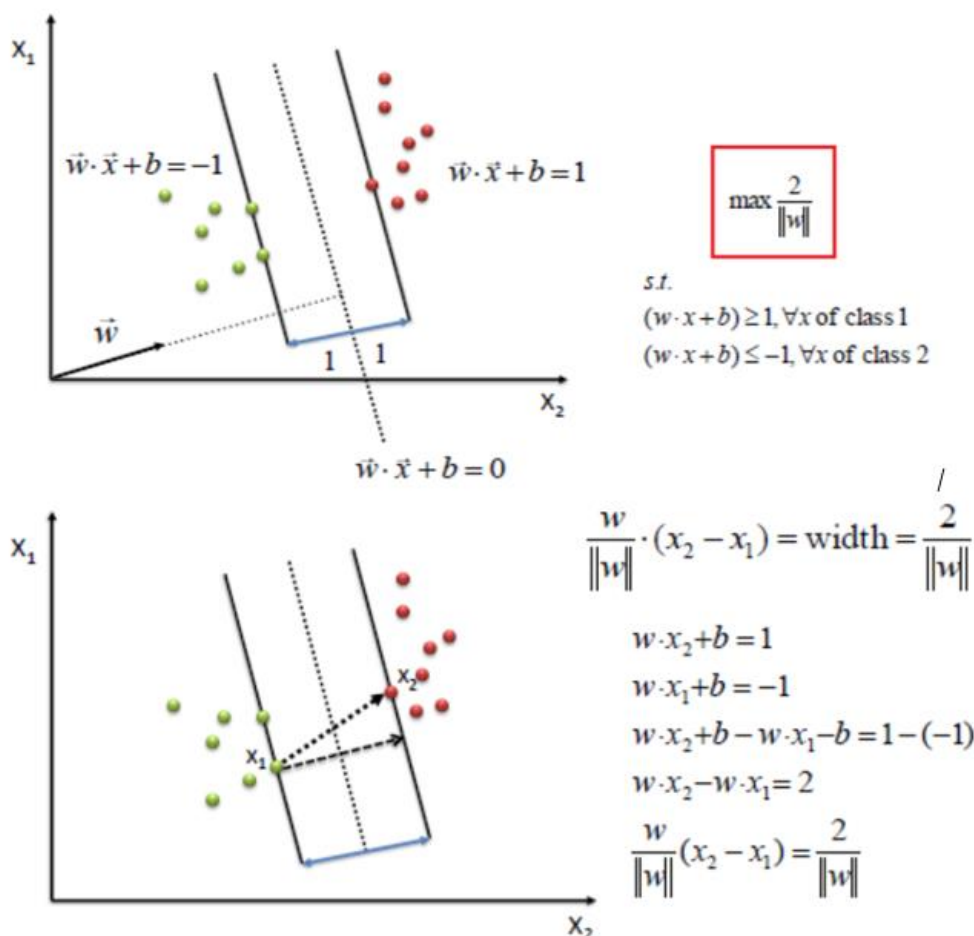
SVM модел класификује узорке тражећи хиперраван која максимизује маргину између двије класе. Маргина представља растојање између најближих узорака супротној класи, при чему се ти узорци називају вектори носачи (енг. *support vectors*) (слика 4.1). Вектори носачи служе за одређивање хиперравни која је једнако удаљена од вектора носача (тзв. граница одлучивања) [2].



Слика 4.1. Вектори носачи и маргина²⁶

За одређивање оптималне хиперравни потребно је максимизирати ширину маргине (слика 4.2).

²⁶ https://www.saedsayad.com/support_vector_machine.htm



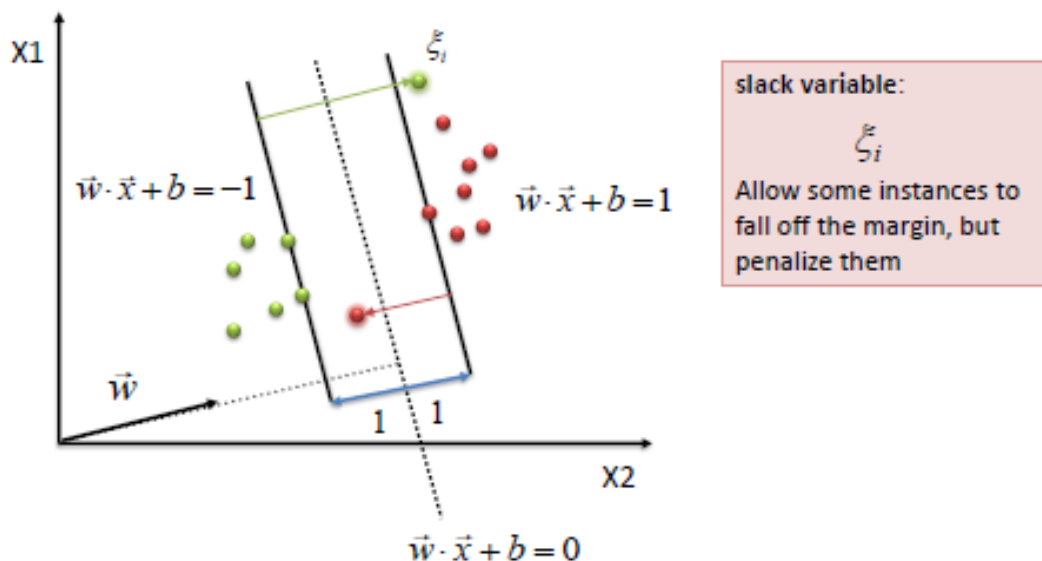
Слика 4.2. Формулација проблема максимизације маргине²⁷

Ширина маргине (w) и граница одлучивања (b) се одређују рјешавајући следећу једначину помоћу метода квадратног програмирања:

$$\min \frac{1}{2} \|\vec{w}\|^2 : y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \forall \vec{x}_i$$

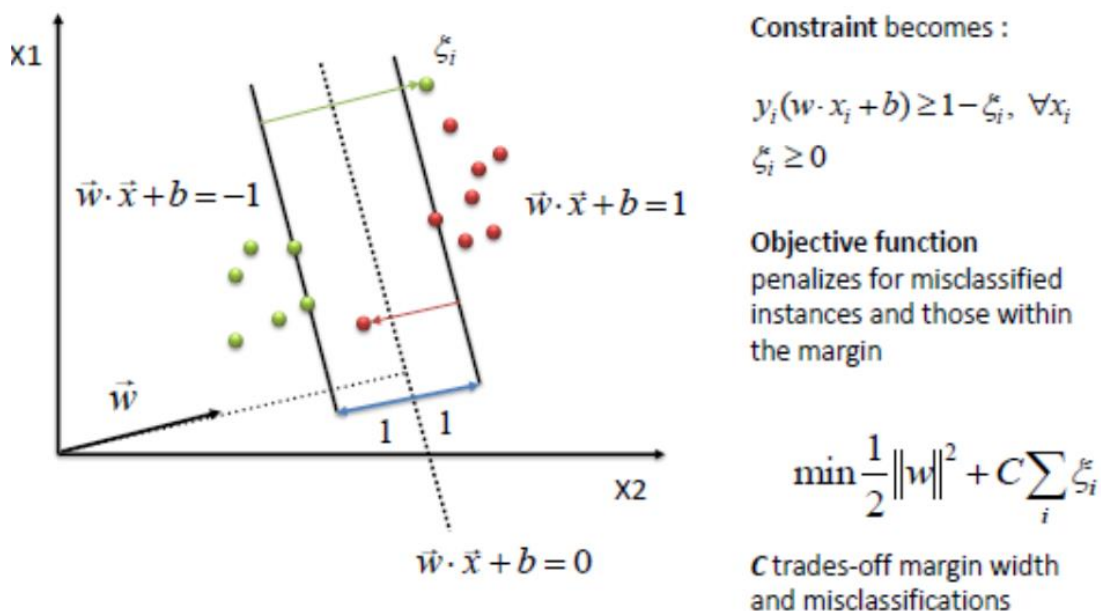
У идеалном случају, рјешавајући наведену једначину, добија се хиперраван која у потпуности раздваја узорке. Међутим, у већини проблема то није случај, те је због тога уведена корекциона промјенљива (енг. *slack variable*) која дозвољава погрешне класификације, али уз одређену цијену (слика 4.3).

²⁷ https://www.saedsayad.com/support_vector_machine.htm



Слика 4.3. SVM са корекционом промјенљивом²⁸

У овом случају, алгоритам максимизује маргину, при чему настоји да корекционе промјенљиве буду (приближно) једнаке нули, минимизирајући суму удаљености погрешно класификованих узорака од хиперравни (слика 4.4).



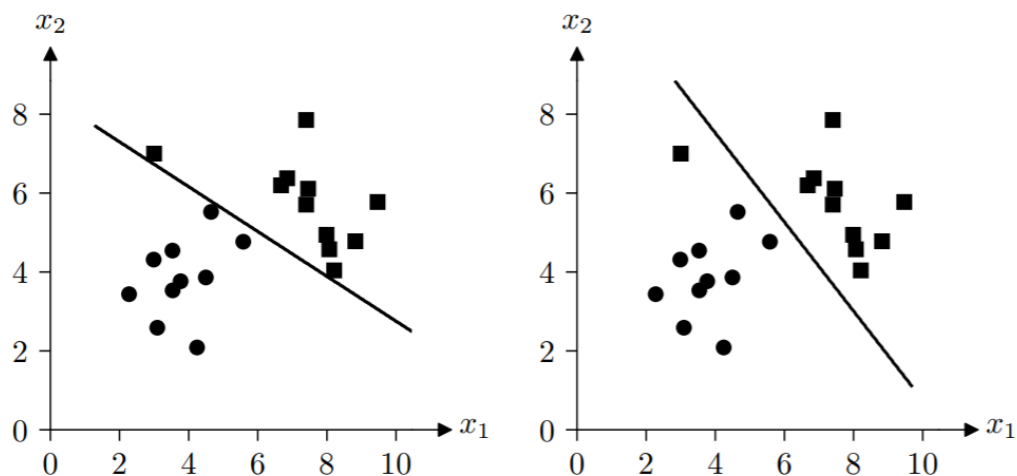
Слика 4.4. Формулација проблема максимизације маргине са корекционим промјенљивим²⁹

Параметар C представља регулациони члан којим се контролише способност генерализације. Уколико је вриједност параметра C превелика, тачност класификације ће се побољшати, али ће то довести до слабије генерализације, док премале

²⁸ https://www.saedsayad.com/support_vector_machine.htm

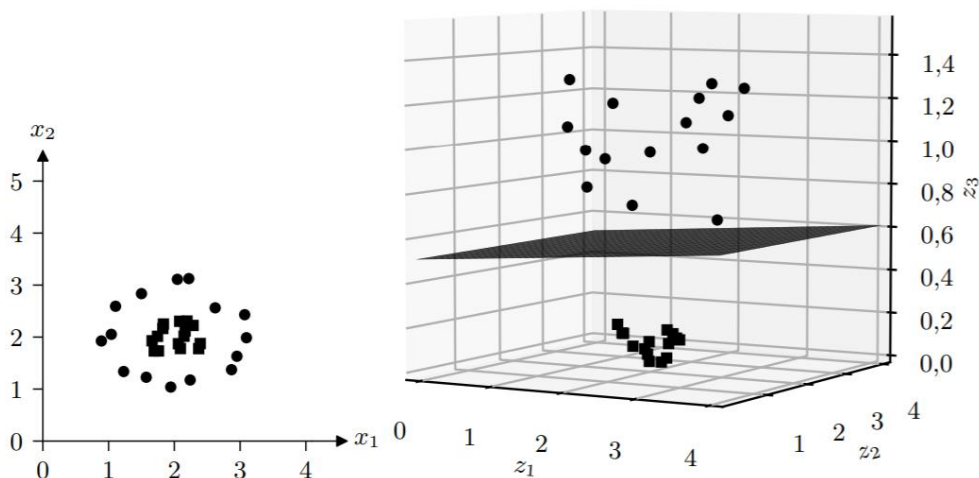
²⁹ https://www.saedsayad.com/support_vector_machine.htm

вриједности параметра C могу довести до премало прилагођеног модела. Утицај параметра C је приказан на слици 4.5 [2].



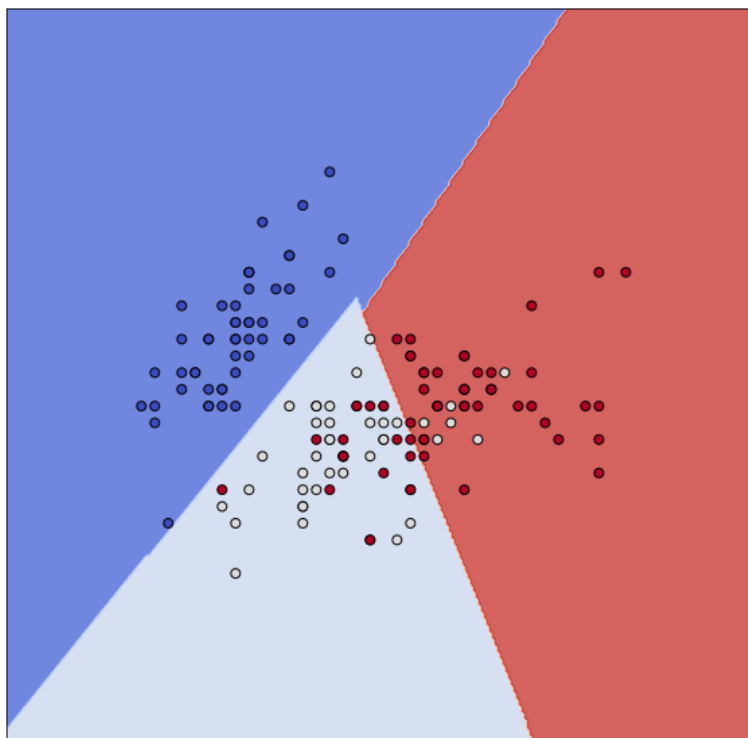
Слика 4.5. Граница одлучивања за $C=100$ и $C=1$ [2]

У случају линеарно недјеливих класа, SVM користи кернел функцију како би узорке у линеарно недјеливом простору пресликао у вишедимензиони простор, у којем може пронаћи оптималну хиперраван (Слика 4.6). Често коришћене кернел функције су: линеарни кернел, полиномијални кернел и Гаусов кернел (радијална базна функција) [2].



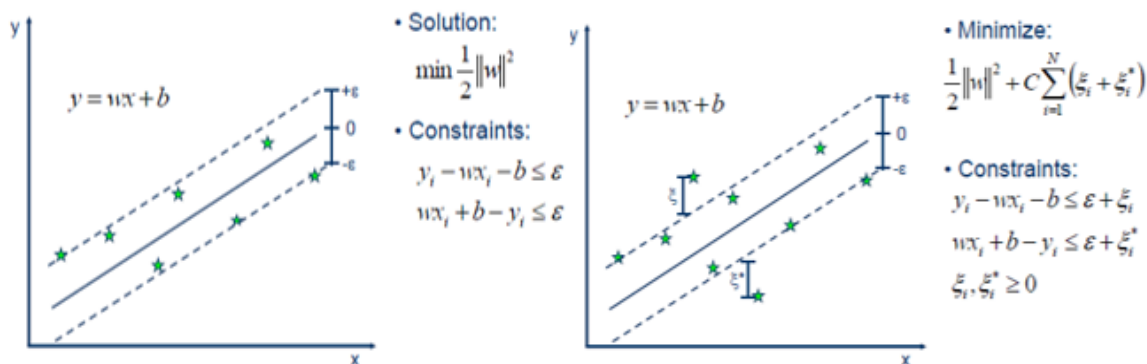
Слика 4.6. Мапирање линеарно недјеливих класа у вишедимензиони простор обиљежја у којем класе постају линеарно сепарабилне [2]

У случају класификације у више класа, врши се обучавање онолико класификатора колико постоји класа, при чему се узорци у тренинг скупу из те класе означавају позитивним, а остали узорци негативним. Затим се на тестне узорке примијени сваки од обучених класификатора и узорак се додјељује оној класи којој одговара највећа сигурност класификације. Овакав класификатор се назива линеарна машина [2]. Границе одлучивања за овакав класификатор су приказане на слици 4.7.



Слика 4.7. Границе одлучивања у случају класификације више класа

SVM се може искористити и за рјешавање регресионих проблема (слика 4.8).



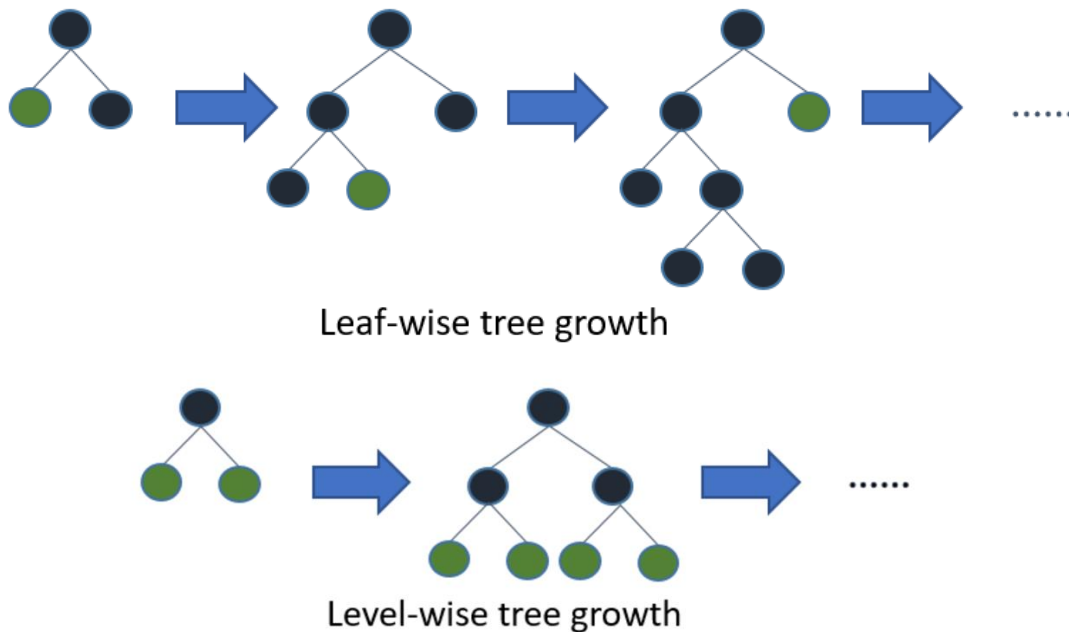
Слика 4.8. Формулација проблема и функција коштања код SVR³⁰

Базира се на истом принципу као и класични SVM, уз неколико мањих разлика. С обзиром да је циљ регресије одредити континуалну промјенљиву, веома је тешко предвидјети тачну вриједност, с обзиром да постоји бесконачно много рјешења. Из тога разлога SVM за регресију (SVR) уводи маргину толеранције (епсилон). Осим тога, идеја је иста као и код класификације, тј. потребно је минимизирати грешку и пронаћи хиперраван са максималном маргином, узимајући у обзир да се дио грешака толерише (грешке чија је удаљеност мања од удаљености епсилон маргина од хиперравни).

³⁰ https://www.saedsayad.com/support_vector_machine_reg.htm

4.2. LightGBM

LGBM (представља брзи, дистрибуирани *framework*³¹ са високим перформансама, базиран на градијентном појачавању (енг. *gradient boosting*) и стаблима одлуке. За разлику од других алгоритама базираних на стаблима одлуке, код LGBM-а стабла расту у односу на лист, док код других алгоритама (XGBoost³², AdaBoost³³) стабла расту по нивоима. Нова стабла се додају на лист са максималним делта губитком (енг. *delta loss*), што омогућава мање губитке у односу на метод ширења по нивоима (слика 4.9) [11].



Слика 4.9. Развијање стабла по листовима и по нивоима³⁴

Стабла одлучивања базирана на градијентном појачавању (енг. *Gradient Boosting Decision Trees GDBT*) представљају ансамбл³⁵ стабала одлуке тренираних у секвенци (*boosting* метод). У свакој итерацији GDBT поправља тренутно рјешење оптимизационог проблема, додајући негативну вриједност градијента функције која се минимизује. Након што се модел обучи, сљедећи задатак представља проналазак оптималних тачака гранања (енг. *split points*) [1].

У сваком чвору стабла, осим листова, налази се по један тест. Сваки тест има више од једног исхода. Сваком исходу одговара само једна грана стабла која води до сљедећег чвора. Листови су означени вриједностима које представљају предвиђања

³¹ Софтвер који пружа одређен скуп функционалности и који се може изнова користити за развој апликација, производа и рјешења

³² <https://xgboost.ai/about>

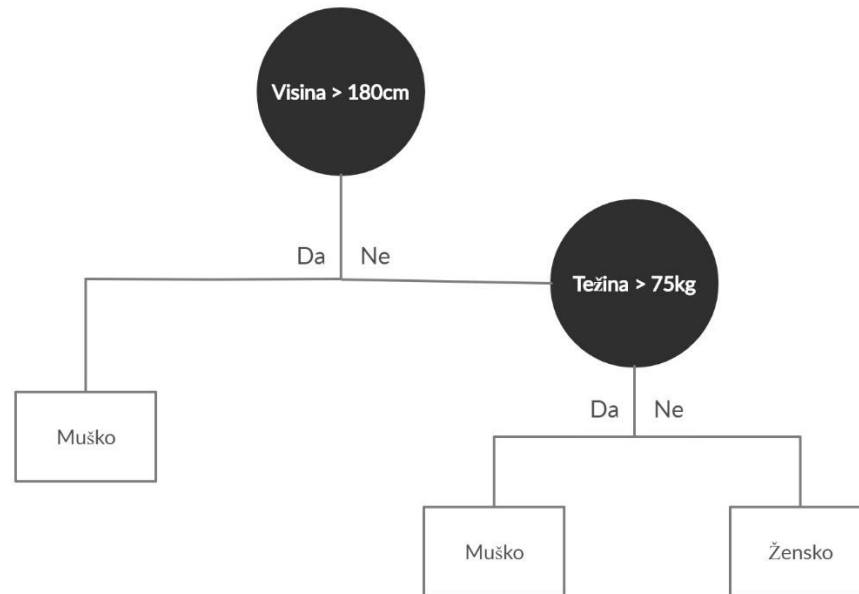
³³ <https://en.m.wikipedia.org/wiki/AdaBoost>

³⁴ <https://github.com/Microsoft/LightGBM/blob/master/docs/Features.rst>

³⁵ Скуп већег броја модела који заједнички доносе одлуке у циљу минимизације грешке

стабла. Тачке гранања представљају вриједности обиљежја на основу којих се врши гранање у чвору стабла. Слика 4.10. представља примјер стабла одлуке гдје се гранање врши у чворовима висина и тежина. Оптимално гранање се бира из простора кандидата гранања који је одређен на основу тренутног знања модела. Другим ријечима, бирају се тачке гранања из којих се добија највише информација [1].

Одређивање вриједности које одговарају листовима, у случају регресије, обично се израчунава као просјек вриједности које су придружене датом листу, а у случају класификације, као већинска класа међу тим узорцима [1].



Слика 4.10. Примјер стабла одлуке

LGBM одређује оптималне тачке гранања на основу алгоритма базираног на хистограму (енг. *Histogram based algorithm*). Алгоритам ради тако што за вријеме обучавања групише вриједности обиљежја у дискретне ступце (енг. *bins*) како би конструисао хистограм обиљежја [11].

LGBM настоји да убрза вријеме потребно за изградњу хистограма пододмјеравањем скупа података и груписањем обиљежја користећи GOSS (*Gadient Based One Side Sampling*) и EFB (*Exclusive Feature Bundling*) алгоритме [11].

Псеудо кођ за наведене алгоритме је приказан на сликама 4.11 и 4.12.

```

Input:  $I$ : training data,  $d$ : iterations
Input:  $a$ : sampling ratio of large gradient data
Input:  $b$ : sampling ratio of small gradient data
Input:  $loss$ : loss function,  $L$ : weak learner
models  $\leftarrow \{\}$ , fact  $\leftarrow \frac{1-a}{b}$ 
topN  $\leftarrow a \times \text{len}(I)$ , randN  $\leftarrow b \times \text{len}(I)$ 
for  $i = 1$  to  $d$  do
    preds  $\leftarrow$  models.predict( $I$ )
     $g \leftarrow loss(I, \text{preds})$ ,  $w \leftarrow \{1, 1, \dots\}$ 
    sorted  $\leftarrow$  GetSortedIndices(abs( $g$ ))
    topSet  $\leftarrow$  sorted[1:topN]
    randSet  $\leftarrow$  RandomPick(sorted[topN:len( $I$ )],
    randN)
    usedSet  $\leftarrow$  topSet + randSet
     $w[\text{randSet}] \times = \text{fact}$   $\triangleright$  Assign weight  $fact$  to the
    small gradient data.
    newModel  $\leftarrow L(I[\text{usedSet}], -g[\text{usedSet}],$ 
     $w[\text{usedSet}])$ 
    models.append(newModel)

```

Слика 4.11. Псеудо код GOSS алгоритма [11]

GOSS задржава узорке са великим градијентима док пододмјеравање врши над узорцима са малим градијентима на насумичан начин (енг. *random sample*).

EFB се састоји из два корака:

1. Одређивање обиљежја који се могу груписати
2. Груписање одабраних обиљежја.

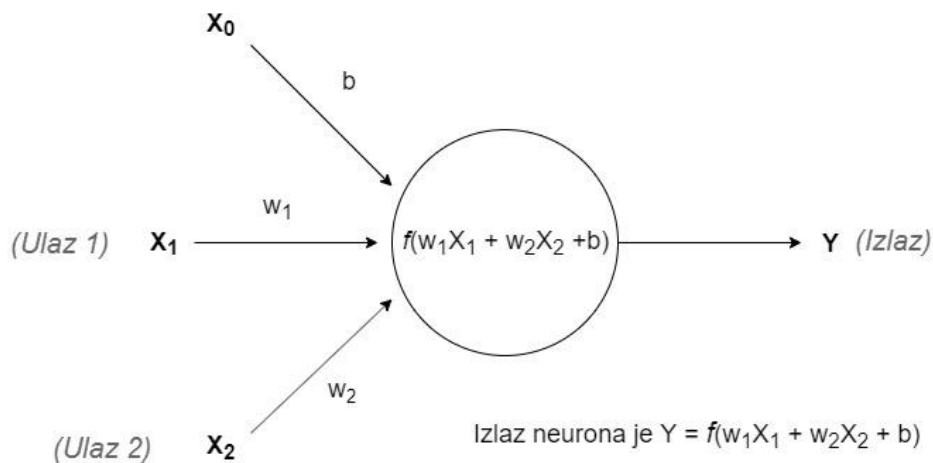
На тај начин се добија на уштеди времена током обучавања модела.

| | |
|--|---|
| <pre> Input: F: features, K: max conflict count Construct graph G searchOrder $\leftarrow G.\text{sortByDegree}()$ bundles $\leftarrow \{\}$, bundlesConflict $\leftarrow \{\}$ for i in searchOrder do needNew \leftarrow True for $j = 1$ to len(bundles) do cnt \leftarrow ConflictCnt(bundles[j], $F[i]$) if cnt + bundlesConflict[j] $\leq K$ then bundles[j].add($F[i]$), needNew \leftarrow False break if needNew then Add $F[i]$ as a new bundle to bundles Output: bundles </pre> | <pre> Input: numData: number of data Input: F: One bundle of exclusive features binRanges $\leftarrow \{0\}$, totalBin $\leftarrow 0$ for f in F do totalBin += f.numBin binRanges.append(totalBin) newBin \leftarrow new Bin(numData) for $i = 1$ to numData do newBin[i] $\leftarrow 0$ for $j = 1$ to len(F) do if $F[j].\text{bin}[i] \neq 0$ then newBin[i] $\leftarrow F[j].\text{bin}[i] + \text{binRanges}[j]$ Output: newBin, binRanges </pre> |
|--|---|

Слика 4.12. Псеудо код EFB алгоритма [11]

4.3. Неуралне мреже

Модел неуралних мрежа је базиран на принципу рада неурона у људском мозгу. Основна јединица која чини неуронску мрежу је неурон (слика 4.13).



Слика 4.13. Илустрација неурона

Неурон прима податке на улазу, обрађује их и резултат обраде просљеђује на излаз. Сваком улазу се додјељују тежине (енг. *weights*). Поред тежина, на сваки улаз се додаје слободан члан - помјерај (енг. *bias*). Тежине и помјерај су параметри које модел учи. Унутар сваког неурона рачуна се функција f као тежинска сума улаза. Ова функција се назива активациона функција (енг. *activation function*) и њена улога је да уведе нелинеарност у излаз неурона. Разлог за то је што је већина улазних података нелинеарна, те је на тај начин омогућено да мрежа научи нелинеарне репрезентације података. Свака активациона функција као улаз прима број и над њим се примјењује одређена математичка операција [1] [2].

Активационе функције које се данас често користе су:

- Сигмоидна

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

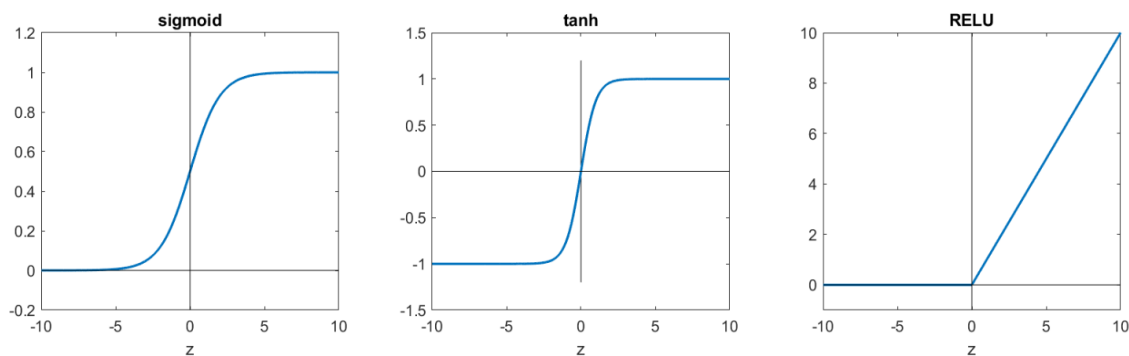
- Тангенс хиперболички

$$\tanh(x) = 2\sigma(2x) - 1$$

- Исправљачка линеарна функција (енг. *Rectified Linear Unit – ReLU*)

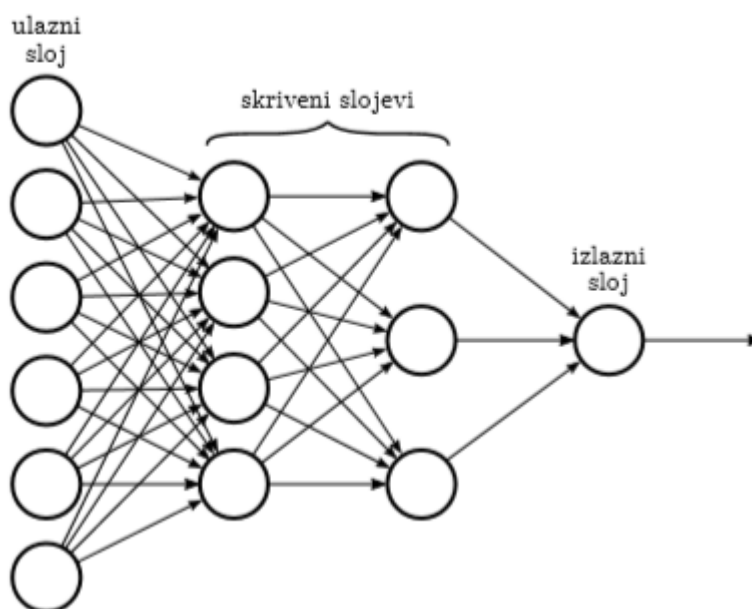
$$f(x) = \max(0, x)$$

Њихов графички приказ је дат на слици 4.14.



Слика 4.14. Активационе функције³⁶

Модел који се састоји од само једног неурона назива се перцептрон и то је уједно и најједноставнији модел неуронских мрежа. Такав модел није претјерано употребљив. Вишеслојне неуронске мреже (слика 4.15) се састоје од неурона који су груписани у слојеве. Неурони у првом слоју су улазни неурони, они имају само један улаз и вриједност на излазу је једнака вриједности на улазу. Неурони у посљедњем слоју су излазни неурони. Због тога се овај слој назива излазни слој. Слојеви који се налазе између улазног и излазног се називају скривени слојеви. Улази скривених слојева представљају излазе из претходног слоја [1].



Слика 4.15. Примјер потпуно повезане неуронске мреже [2]

Број скривених слојева и број неурона у сваком скривеном слоју представљају хиперпараметре мреже. Хиперпараметри мреже дефинишу њену архитектуру и утичу на алгоритам обучавања. Архитектура мреже се најчешће одређује емпиријски.

Циљ обучавања неуралне мреже је одредити скуп тежина и помјерај који минимизују функцију коштања. С обзиром да се неуронска мрежа састоји од великог броја тежина и да је функција цијене нелинеарна функција тежина, проблем

³⁶ <https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2019/www/hwnotes/HW1p1.html>

минимизације није могуће ријешити аналитичким путем. Умјесто тога, за минимизацију се користи метод градијентног спуста (енг. *gradient descent*). Почевши од излазног слоја па до улазног, алгоритам у свакој итерацији обавља сљедеће [1]:

- Проширује до тада израчунати парцијални извод изводом активационе функције по правилу рачунања извода сложене функције
- Израчунава вриједност градијента по параметрима чворова на текућем нивоу (тежине и слободни члан), а до тада израчунати парцијални извод се множи улазима чворова које ти параметри множе
- Проширује до тада израчунати парцијални извод изводом линеарне комбинације по улазима по правилу рачунања извода сложене функције.

Неуралне мреже се могу искористити за рјешавање регресионих и класификационих проблема. У случају регресионог проблема, у чворовима излазног слоја активациона функција се изоставља, док се, у случају класификационог, примјењује функција меког максимума (енг. *softmax*):

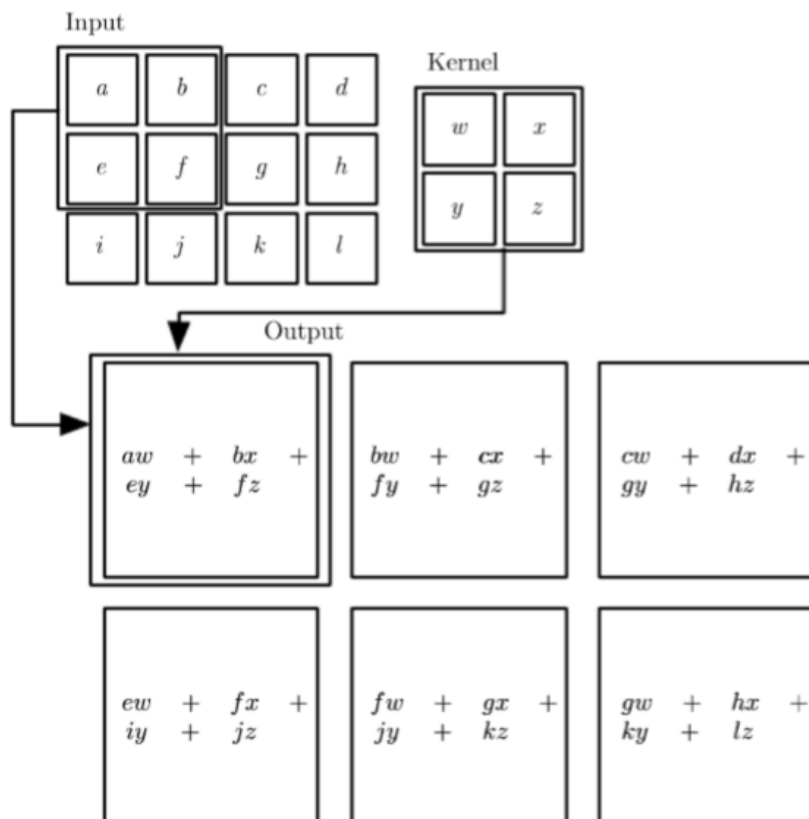
$$softmax(x) = \left(\frac{e^{x_1}}{\sum_{i=1}^C e^{x_i}}, \dots, \frac{e^{x_C}}{\sum_{i=1}^C e^{x_i}} \right)$$

У овом случају, излаз представља вектор димензије C , при чему C представља број класа. С обзиром да се вриједности сумирају на 1, резултат ове функције се може користити као расподела вјероватноћа. Индекс елемента са највећом вјероватноћом се узима као предикција класе.

Примјер са слике 4.15. представља потпуно повезану неуралну мрежу. Мрежа је добила назив по томе што је излаз једног слоја (осим посљедњег) повезан са свим улазима у наредном слоју. Осим тога, постоје и друге врсте, као што су конволуционе неуралне мреже и рекурзивне неуралне мреже.

4.3.1 Конволуционе неуралне мреже

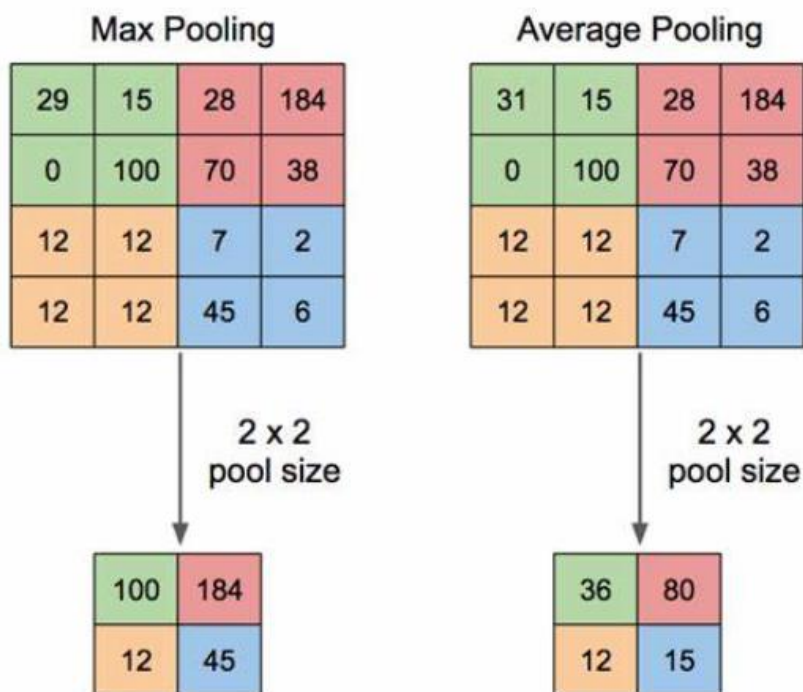
Конволуционе неуралне мреже (енг. *Convolutional neural networks* - *CNN*) се користе у обради сигнала, као што је звук, слика, временска серија. Назив су добиле због тога што уче филтере (кERNELЕ) чијом конволуционом примјеном детектују одређена својства сигнала, као што су ивице, контуре, дијелови објеката и сл (слика 4.16.). Овакви слојеви се називају конволуциони слојеви (енг. *convolution layers*) [1].



Слика 4.16. Конволуција улазне слике са филтером [1]

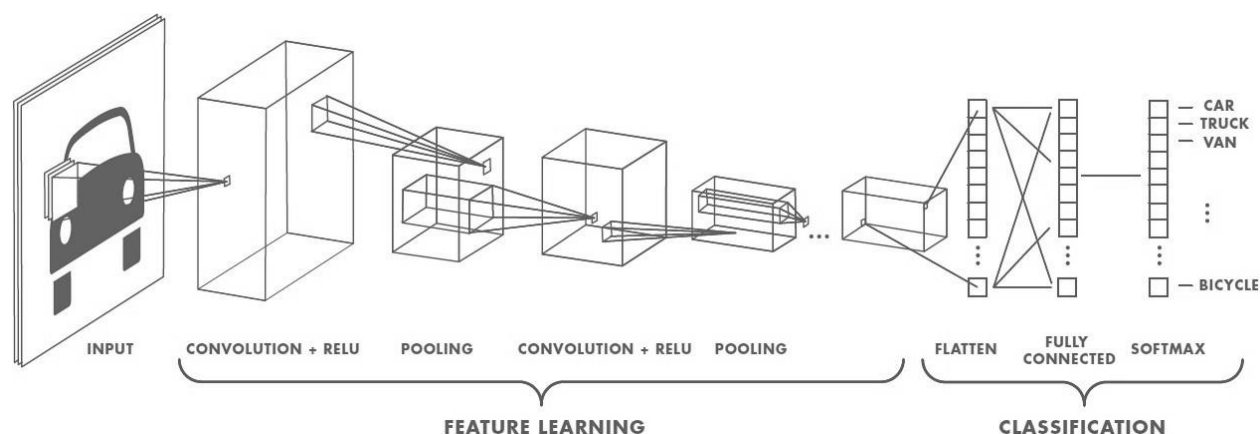
Након конволуције примјењује се нелинеарна активациона функција (најчешће ReLu). Излаз конволуционог слоја је мапа обиљежја (енг. *feature map*) која се може користи као улаз наредног конволуционог слоја. Филтери у сваком слоју представљају детекторе одређених узорака чије присуство/одсуство има значај за модел.

Поред конволуционих слојева, у конволуционим неуронским мрежама се користе и слојеви за агрегацију (енг. *pooling layer*). Агрегациони слојеви слиједе након конволуционих. За слој је потребно дефинисати величину прозора. Агрегација се реализује тако што се прозор помјера по мапи обиљежја, при чему она одређује највећу или средњу вриједност узорака обухваћених прозором (слика 4.17). Дакле, употребом слојева за агрегацију, смањује се количина података, а тиме и број параметара које мрежа треба да научи што резултује бржем обучавању. Осим тога, слојеви за агрегацију обезбјеђују одређени степен инваријантности на трансформације обиљежја [1] [2].



Слика 4.17. Примјер агрегације максималне и средње вриједности (број параметара је смањен са 16 на 4)³⁷

Архитектура конволуционе неуронске мреже се обично састоји од неколико конволуционих и агрегационих слојева, након којих слиједе потпуно повезани слојеви (слика 4.18).

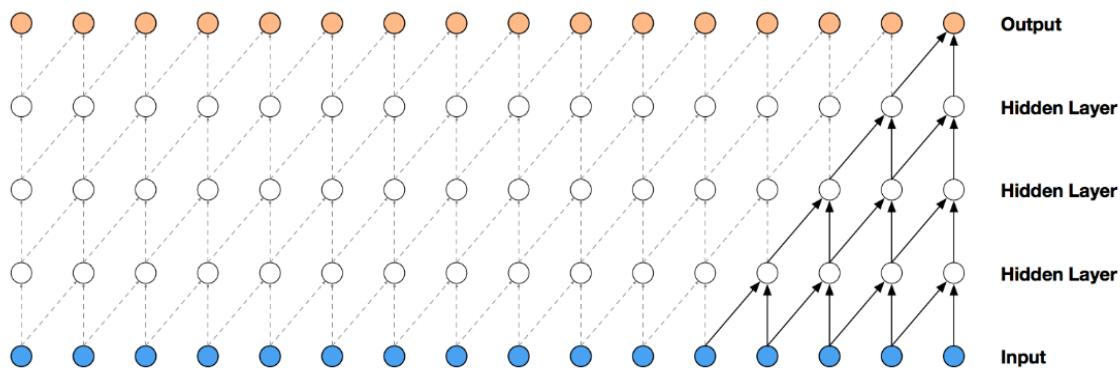


Слика 4.18. Примјер конволуционе неуронске мреже за класификацију бројева са слике³⁸

Уколико се користе за обраду временских серија, неопходно је ограничити узорке над којима се конволуција примјењује. У таквим случајевима конволуција се одвија каузално (енг. *causal*). Примјер каузалне конволуције је приказан на слици 4.19.

³⁷https://www.researchgate.net/publication/333593451_Application_of_Transfer_Learning_Using_Convolutional_Neural_Network_Method_for_Early_Detection_of_Terry%27s_Nail/figures?lo=1

³⁸https://gigazine.net/gsc_news/en/20170808-what-is-deep-learning



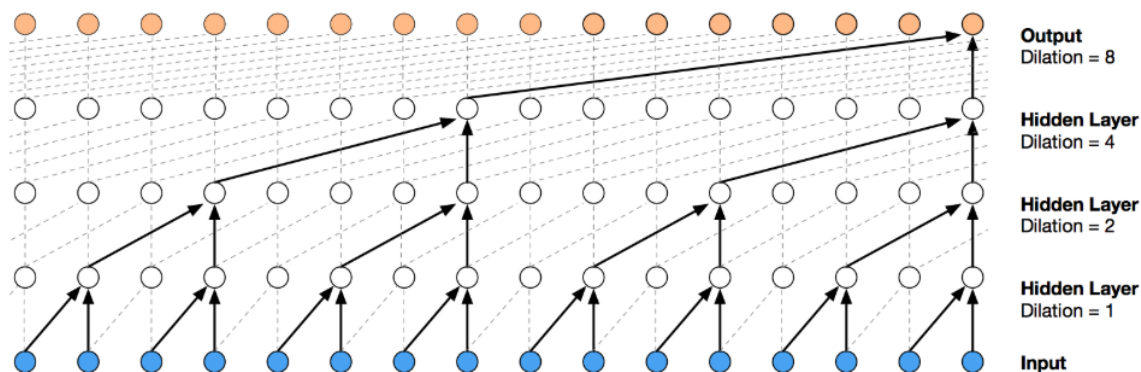
Слика 4.19. Каузална конволуција [6]

Каузална конволуција не узима у обзир будуће узорке, тако да, у случају излаза у тренутку t , улази конволуције представљају узорци у тренуцима $t - n, t + 1 - n, \dots, t$ (n = величина филтера).

Иако каузална конволуција искључује будуће узорке, да би се узеле у обзир дугорочне зависности између узорака, неопходно је повећати рецептивно поље филтера. Један начин је да се повећа величина филтера и број конволуционих слојева, међутим, такав приступ би резултовао превише сложеним моделом, како рачунарски тако и статистички. Другим ријечима, модел би био неупотребљив [6] [12].

Да би се ријешило проблем дугорочних зависности користе се проширене конволуције (енг. *dilated convolutions*) које експоненцијално проширују рецептивно поље повећавањем броја слојева. У проширеном конволуционом слоју, филтери се не примјењују тако што између сваког улазног узорка постоји одређен корак (енг. *dilation rate*). Корак се повећава како број проширених конволуционих слојева расте. Величина корака је 2^l , гдје l представља број проширеног конволуционог слоја [6] [12].

На слици 4.20. је приказан примјер блока проширених конволуционих слојева.



Слика 4.20. Проширени конволуциони слојеви [6]

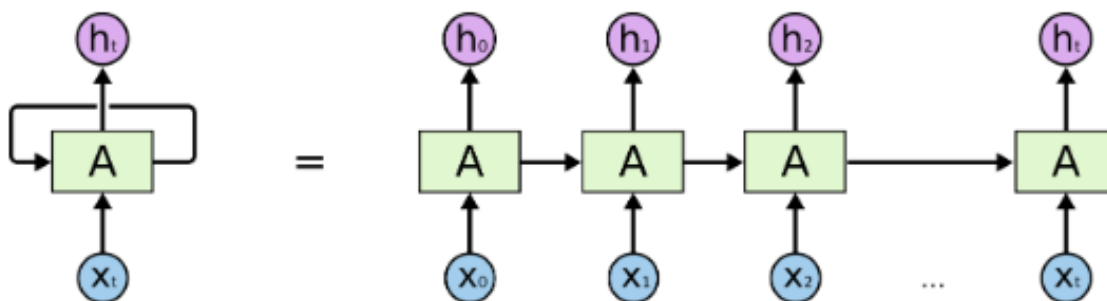
Постоји неколико разлога зашто конволуционе мреже имају широку примјену за обраду сигнала у односу на потпуно повезане мреже.

Сваки чвор је повезан са мањим бројем чворова из претходног слоја, тиме је смањен укупан број параметара за тренирање. Затим, конволуционе мреже користе дијељене параметре дефинисане филтером канала. Конволуционе мреже су неосјетљиве на транслације што им омогућава да пронађу одређени сигнал било гдје, без обзира како је он транслиран на слици. Специјализоване су за топологију сигнала, тј. узимају у обзир распоред интензитета сигнала (на примјер, пиксели на слици) [1].

Са друге стране, конволуционе мреже су осјетљиве на друге трансформације, као што су ротација и скалирање. Додатни проблем је чињеница да конволуциони слојеви морају произвести излаз тачно одређених димензија [1].

4.3.2. Рекурзивне неуралне мреже

Рекурзивне неуралне мреже представљају неуралне мреже специјализоване за обраду секвенцијалних података, као што су звук, реченице природног језика³⁹ и временске серије. Рекурзивне мреже омогућавају моделовање зависности између узорака (нпр. врста ријечи у реченици обично зависи од врста претходних неколико ријечи), тако што садрже скривено стање (h_t) које чува информацију о елементима секвенце, обрађеним у претходним корацима (слика 4.21.) [1].



Слика 4.21. Различите репрезентације рекурзивне неуралне мреже. Компактна репрезентација и репрезентација „разматравање кроз вријеме“ [13]

Параметри мреже одређују како се скривено стање мијења из корака у корак на основу претходног стања и тренутних улаза и како се генерише излаз мреже, у зависности од тренутног стања.

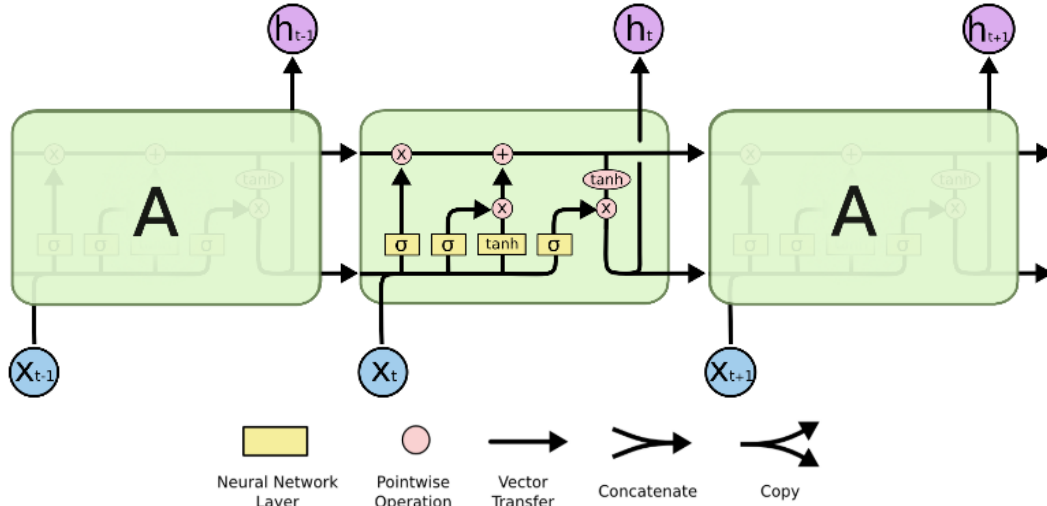
У пракси, класичне рекурзивне мреже се веома ријетко сусрећу. Умјесто њих се користи дуга краткотрајна меморија (енг. *long short-term memory LSTM*). LSTM представља врсту рекурзивних мрежа са специфичном структуром, која омогућава контролу читања и уписа у јединицу.

Основна идеја ових мрежа је постојање *ћелије* која чува скривено стање уз контролу писања, читања и заборављања (слика 4.22). Контрола се врши помоћу капија (енг. *gates*).

³⁹ https://sh.wikipedia.org/wiki/Prirodni_jezik

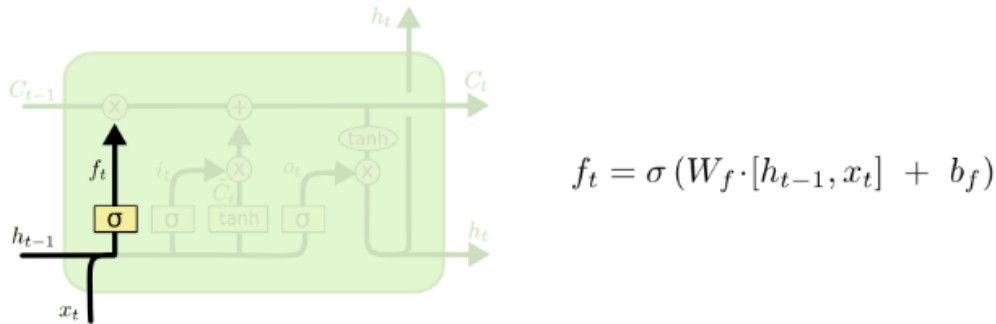
У оквиру LSTM јединице постоје три врсте капија:

1. Улазна капија i_t (енг. *Input gate*)
2. Капија заборављања f_t (енг. *Forget gate*)
3. Излазна капија o_t (енг. *Output gate*).



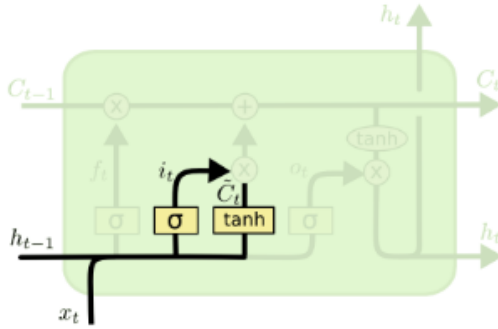
Слика 4.22. Структура LSTM јединице [13]

Први корак представља одлуку које информације ће се одбацити из ћелије. Одлука се доноси на основу капије заборављања и зависи од тренутног улаза x_t и скривеног стања из претходног корака h_{t-1} . Излаз капије заборављања је резултат сигмоидне активационе функције и представља вриједност између 0 и 1, која се множи са вриједностима из претходне ћелије C_{t-1} . Уколико је излаз капије заборављања 0, стање претходне ћелије се у потпуности одбацује, односно, памти уколико је излаз капије 1 (слика 4.23) [13].



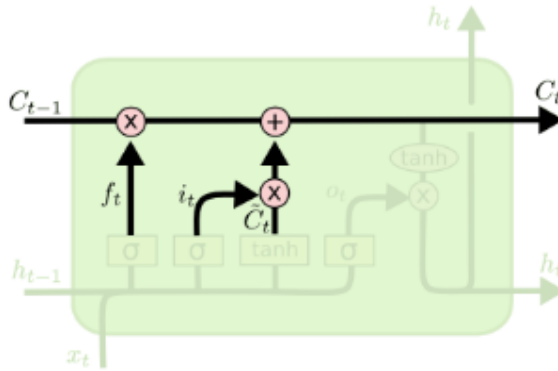
Слика 4.23. Капија заборављања [13]

Наредни корак представља одлуку о чувању нових информација у ћелији. Овај корак се састоји из два дијела. Први дио чини улазна капија која одлучује које вриједности ће се ажурирати, а други дио чини \tanh слој чији резултат представља кандидата за ново стање \tilde{C}_t (слика 4.24) [13].



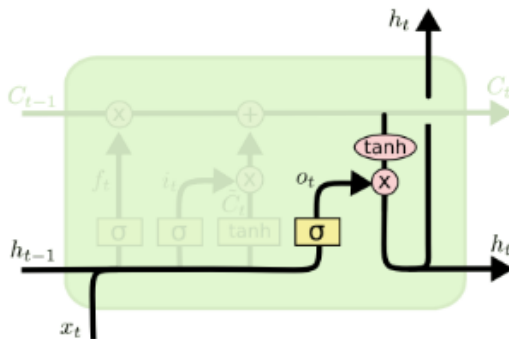
Слика 4.24. Ажурирање стања ћелије (Корак 1) [13]

Ново стање се одређује тако што се претходно стање C_{t-1} множи са капијом заборављања, а затим се дода производ улазне капије и кандидата за ново стање. Множењем кандидата и улазне капије се одређује за колико је LSTM јединица одлучила да ажурира претходно стање (слика 4.25) [13].



Слика 4.25. Ажурирање стања ћелије (Корак 2) [13]

Последњи корак представља одлуку о томе које информације ће бити прослијеђене на излаз. Одлука се доноси на основу излазне капије и зависи од тренутног улаза x_t и скривеног стања из претходног корака h_{t-1} (као и код улазне капије). Вриједност капије заборављања представља излаз сигмоидне функције. Стање ћелије се пропушта кроз \tanh слој и множи са излазном капијом како би се одредио излаз ћелије h_t (слика 4.26) [13].



Слика 4.26. Излаз LSTM јединице [13]

Захваљујући томе што моделују прелазе из стања у стање, умјесто директне зависности излаза од претходних улаза, рекурзивне мреже су у стању да представе дубоке структурне зависности и да обрађују секвенце промјенљиве велике дужине. LSTM јединице омогућавају моделовање дугорочних зависности, као што је временска серија. Међутим, због велике дужине секвенце и броја параметара, LSTM јединице су теже за обучавање у односу на потпуно повезане и конволуционе мреже [1].

5. ПРАКТИЧНИ РАД

Практични рад се састоји од *web* апликације која путем REST APIa⁴⁰ добија излазе модела за предвиђање повишеног водостаја. Web апликација је реализована помоћу *VueJS*⁴¹ frameworka, REST API је реализован помоћу *Flask frameworka*, а за креирање модела су искоришћене *LightGBM*, *scikit-learn*⁴² и *TensorFlow*⁴³ библиотеке.

У наставку је приказана реализација корака процеса машинског учења, REST API-а и *web* апликације.

Први у низу корака је био форматирање података. Форматирање података је извршено тако што је приликом учитавања *Excel* табеле прослијеђен конвертор за сваку колону табеле. Примјер учитавања је приказан на слици 5.1. заједно са реализацијом конвертора за колону „Проток“.

```
27
28 def water_flow_converter(cell_value):
29     cell_value_str = str(cell_value)
30     whole_part, decimal_part = cell_value_str[:-3], cell_value_str[-3:]
31     try:
32         return float(whole_part + "." + decimal_part)
33     except:
34         if str(cell_value) in ["A0P0", "A30P30"]:
35             return FLAG
36         else:
37             return np.NaN

[ ] 1 water_level = pd.read_excel(xlsx, sheet_name=bl_sheets[0], converters={0:date_converter, 1:time_converter})
    2 water_flow = pd.read_excel(xlsx, sheet_name=bl_sheets[1], converters={0:date_converter, 1:time_converter, 2:water_flow_converter, 3:water_flow_converter})
```

Слика 5.1. Учитавање и форматирање података за станицу у Бањој Луци

Након форматирања услиједило је уклањање редова са недостајућим вриједностима и редова који се понављају у табели (слике 5.2 и 5.3).

⁴⁰ https://en.wikipedia.org/wiki/Representational_state_transfer

⁴¹ <https://vuejs.org>

⁴² <https://scikit-learn.org/stable/index.html>

⁴³ <https://www.tensorflow.org>

```
[ ] 1 def drop_na(df):
2     shape_1 = df.shape
3     print(f"{shape_1} ->", end=" ")
4     df = df.dropna()
5     shape_2 = df.shape
6     print(f"{shape_2} | dropped {shape_1[0] - shape_2[0]} rows")
7     if shape_1 != shape_2:
8         df = df.reset_index(drop=True)
9     return df

[ ] 1 water_level = drop_na(water_level)
2     water_flow = drop_na(water_flow)

↳ (25637, 3) -> (25634, 3) | dropped 3 rows
   (137805, 3) -> (111707, 3) | dropped 26098 rows
```

Слика 5.2. Уклањање редова са недостајућим вриједностима

Тим кораком је завршено чишћење података. С обзиром да је велики број редова у табели био уклоњен, неопходно је било извршити њено допуњавање.

```
[ ] 1 def drop_duplicates(df, subset="DateTime"):
2     shape_1 = df.shape
3     print(f"{shape_1} ->", end=" ")
4     df = df.drop_duplicates(subset=subset)
5     shape_2 = df.shape
6     print(f"{shape_2} | dropped {shape_1[0] - shape_2[0]} duplicates")
7     return df

[ ] 1 water_level = drop_duplicates(water_level)
2     water_flow = drop_duplicates(water_flow)

↳ (25634, 2) -> (25634, 2) | dropped 0 duplicates
   (111707, 2) -> (22533, 2) | dropped 89174 duplicates
```

Слика 5.3. Уклањање дупликата

Допуњавање табеле (слика 5.4.) је извршено на следећи начин: прво су допуњени недостајући термини мјерења, а затим су вриједности водостаја и протока за недостајуће термине одређене помоћу функције *fill_nans* чија реализација је приказана на слици 5.5.

```
[ ] 1 resampled = water_level.set_index("DateTime").resample("30T").mean().reset_index()
    2 resampled.info()
    3 water_level = resampled.apply(fill_nans,axis=1)
    4 water_level.info()
    5 print(f"\nFilled with {resampled['WaterLevel'].isnull().sum() - water_level['WaterLevel'].isnull().sum()} values")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41662 entries, 0 to 41661
Data columns (total 2 columns):
DateTime      41662 non-null datetime64[ns]
WaterLevel     25634 non-null float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 651.0 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41662 entries, 0 to 41661
Data columns (total 2 columns):
DateTime      41662 non-null datetime64[ns]
WaterLevel     41300 non-null object
dtypes: datetime64[ns](1), object(1)
memory usage: 651.0+ KB

Filled with 15666 values
```

Слика 5.4. Допуњавање табеле

Претходно описани кораци су извршени одвојено над водостајем и протоком. Сљедећи корак је представљао спајање табела.


```

1 def get_past_next_years_values(row, key="WaterLevel"):
2     """ Gets key values in the available search space based on the year from the row
3     This function will search all available key values from the past and from the future, accumulate them into a list and return the list
4     Example:
5         If the row["DateTime"] value is 2017-01-01,
6         The for loop will try to access key values from 2016-01-01, 2018-01-01 and 2019-01-01
7     """
8     min_year = resampled["DateTime"].min().year
9     max_year = resampled["DateTime"].max().year
10    curr_year = row["DateTime"].year
11    values = []
12
13    for i in range(1, max_year - min_year + 1):
14        minus_dt = row["DateTime"].replace(year=curr_year - i)
15        plus_dt = row["DateTime"].replace(year=curr_year + i)
16
17        if min_year <= minus_dt.year:
18            minus = resampled[resampled["DateTime"] == minus_dt]
19            if minus.shape[0] > 0:
20                # Get key value from minus year
21                minus_key = minus[key].iloc[0]
22                if not pd.isnull(minus_key):
23                    values.append(minus_key)
24
25        if plus_dt.year <= max_year:
26            plus = resampled[resampled["DateTime"] == plus_dt]
27            if plus.shape[0] > 0:
28                # Get key value from plus year
29                plus_key = plus[key].iloc[0]
30                if not pd.isnull(plus_key):
31                    values.append(plus_key)
32
33    return values
34
35
36 def fill_nans(row):
37     """ Calculates missing wLvl & wFlow as the mean of the measurements for all past and next years
38     Example: DateTime wLvl wFlow
39             row: 2017-07-07 nan nan -> 2017-07-07 20 20
40             minus: 2016-07-07 10 10
41             plus: 2018-07-07 30 30
42     """
43     WL = "WaterLevel"
44     WF = "WaterFlow"
45
46     if WL in row.index and pd.isnull(row[WL]):
47         wl = get_past_next_years_values(row, key="WaterLevel")
48         if len(wl) > 0:
49             row[WL] = float(np.mean(wl).item())
50
51     if WF in row.index and pd.isnull(row[WF]):
52         wf = get_past_next_years_values(row, key="WaterFlow")
53         if len(wf) > 0:
54             row[WF] = float(np.mean(wf).item())
55     return row

```

Слика 5.5. Функција за одређивање недостајућих вриједности за водостај и проток

Спајање табела је извршено на основу термина мјерења по правилу *outer join*⁴⁴. Као резултат добијена је табела са унијом вриједности појединачних табела (слика 5.6).

⁴⁴ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html>

```
1 sheets = [water_level, water_flow]
2 key = sheets[0].columns[0]
3 %time sheet = sheets[0].merge(sheets[1], on=key, how="outer")
4 sheet = sheet.sort_values([key])
5 sheet = sheet.reset_index(drop=True)
6 # sheet = sheet.dropna(thresh=2)

CPU times: user 22.6 ms, sys: 10 µs, total: 22.6 ms
Wall time: 26.8 ms
```

Слика 5.6. Спајање табела

С обзиром да су и даље постојали термини са недостајућим мјерењима (случај када не постоје претходна и будућа мјерења за недостајући термин), извршено је допуњавање методама задња опсервација пропагирана унапријед и сљедећа опсервација пропагирана уназад (слика 5.7). Тим кораком је завршена припрема података.

```
[ ] 1 display(sheet.info())
2 sheet = sheet.fillna(method="ffill", limit=1)
3 display(HTML("<br/><br/>"))
4 display(sheet.info())
5 sheet = sheet.fillna(method="bfill", limit=1)
6 display(HTML("<br/><br/>"))
7 display(sheet.info())
8 display(HTML("<br/><br/>"))
9 sheet = drop_na(sheet)
10 sheet["WaterLevel"] = sheet["WaterLevel"].astype(float)
11 sheet["WaterFlow"] = sheet["WaterFlow"].astype(float)
12 display(sheet.info())
```

Слика 5.7. Допуњавање недостајућих мјерења методама задња опсервација пропагирана унапријед и сљедећа опсервација пропагирана уназад

Припремљени подаци су затим подијељени у тренинг, валидациони и тестни скуп према односу 80/10/10 (слика 5.8).

```
[ ] 1 def train_test_split(x,y,split=0.8):
2     """Split x,y DataFrames into train and test subsets
3
4     Parameters:
5     -----
6     x,y : pandas.DataFrame
7         Dataframes which will be split based on train_size
8     split : float, optional
9         Should be between 0.0 and 1.0 and represents the proportion of the dataset to include in the train split
10
11     Returns:
12     -----
13     x_train, x_test, y_train, y_test: pandas.DataFrame
14         List of DataFrames containing train-test split of input DataFrames.
15     """
16     data_size = x.shape[0]
17     train_size = int(data_size * split)
18     x_train, x_test = x.iloc[:train_size], x.iloc[train_size:]
19     y_train, y_test = y.iloc[:train_size], y.iloc[train_size:]
20     return x_train, x_test, y_train, y_test

[ ] 1 # Split data into train and test set
2 x_train, x_test, y_train, y_test = train_test_split(x,y)
3 x_valid, x_test, y_valid, y_test = train_test_split(x_test,y_test,split=0.5)
4 print_shape([x_train,x_valid,x_test,y_train,y_valid,y_test])
```

Слика 5.8. Подјела података у тренинг, валидациони и тестни скуп

За одређивање хиперпараметара модела искоришћене су библиотеке *HyperOpt* и *HParams*. Примјер употребе *HyperOpt* библиотеке је приказан на сликама 5.9-5.11.

```
[ ] space = {
    "objective": "regression",
    "device": "cpu",
    "max_bin": hp.quniform("max_bin",255,512,1),
    "num_leaves": hp.choice("num_leaves",[15, 31, 63, 127, 255, 511, 1023, 2047, 4095]),
    "num_iterations " : hp.quniform("num_iterations ",1e2,1e3,1e2),
    "min_data_in_leaf": hp.quniform("min_data_in_leaf",20,255,1), #512,
    'feature_fraction': hp.uniform('feature_fraction', 0.75, 1.0),
    'bagging_fraction': hp.uniform('bagging_fraction', 0.75, 1.0),
    "learning_rate": 0.05
}

space

{ 'bagging_fraction': <hyperopt.pyll.base.Apply at 0x7f2aaa44d5c0>,
  'device': 'cpu',
  'feature_fraction': <hyperopt.pyll.base.Apply at 0x7f2aaa44d908>,
  'learning_rate': 0.05,
  'max_bin': <hyperopt.pyll.base.Apply at 0x7f2aaa623fd0>,
  'min_data_in_leaf': <hyperopt.pyll.base.Apply at 0x7f2aaa44dd30>,
  'num_iterations ': <hyperopt.pyll.base.Apply at 0x7f2aa9751eb8>,
  'num_leaves': <hyperopt.pyll.base.Apply at 0x7f2aaa6239e8>,
  'objective': 'regression' }
```

Слика 5.9. Примјер дефинисања простора претраживања

Први корак приликом рада са *HyperOpt* библиотеком јесте дефинисање простора претраживања (слика 5.9). Након што се дефинише простор претраживања потребно имплементирати функцију циља (слика 5.10). Функција циља као аргумент прима објекат који у себи садржи узорке из простора претраживања који се користи приликом креирања модела. Након што се модел креира, врши се његово обучавање и евалуација. Функција циља као повратну вриједност враћа резултат евалуације.

Повратна вриједност омогућава *HyperOpt*-у да чува најбољи скуп хиперпараметара за вријеме оптимизације.

```
[ ] def objective_r(space):
    space["num_leaves"] = int(space["num_leaves"])
    space["min_data_in_leaf"] = int(space["min_data_in_leaf"])
    space["max_bin"] = int(space["max_bin"])
    # Train
    lgbm_trained = lgbm.train(space,
                              lgbm_train,
                              valid_sets=lgbm_eval,
                              early_stopping_rounds=5,
                              verbose_eval=False)
    # Predict
    y_pred = lgbm_trained.predict(x_valid, num_iteration=lgbm_trained.best_iteration)
    # Evaluate
    score = metrics.mean_squared_error(y_valid.values.ravel(), y_pred)
    scores.append((score, space))
    return score
```

Слика 5.10. Примјер функције циља

Одређивање оптималних хиперпараметара се врши позивом *fmin* функције из *HyperOpt* библиотеке која као аргументе прима функцију циља, простор претраживања, врсту алгоритма претраге и максималан број евалуација. Резултат позива ове функције је *Python dictionary* објекат са оптималним вриједностима хиперпараметара (слика 5.11).

```
[ ] scores=[]
    best_lgbm = fmin(fn=objective_r,
                    space=space,
                    algo=ALGO,
                    max_evals=MAX_EVALS)

[ ] best_lgbm

[ ] {'bagging_fraction': 0.8650449752569586,
    'feature_fraction': 0.7878056863611946,
    'max_bin': 498.0,
    'min_data_in_leaf': 20.0,
    'num_iterations': 100.0,
    'num_leaves': 8}
```

Слика 5.11. Примјер оптимизације хиперпараметара

Оптимални хиперпараметри су искоришћени за обучавање модела, обучени модел је евалуиран и сачуван како би се могао употребити за поређење са другим моделима (слика 5.12). Приликом обучавања овог модела кориштен је „нови“ тренинг скуп којег чине узорци тренинг и валидационог скупа из претходног корака.

```
[ ] x_train_append = x_train.append(x_valid,ignore_index=True)
    y_train_append = y_train.append(y_valid,ignore_index=True)

    lgbm_train = lgbm.Dataset(x_train_append,y_train_append)

[ ] booster_r = lgbm.train(best_lgbm,lgbm_train)

[ ] y_pred = booster_r.predict(x_test)

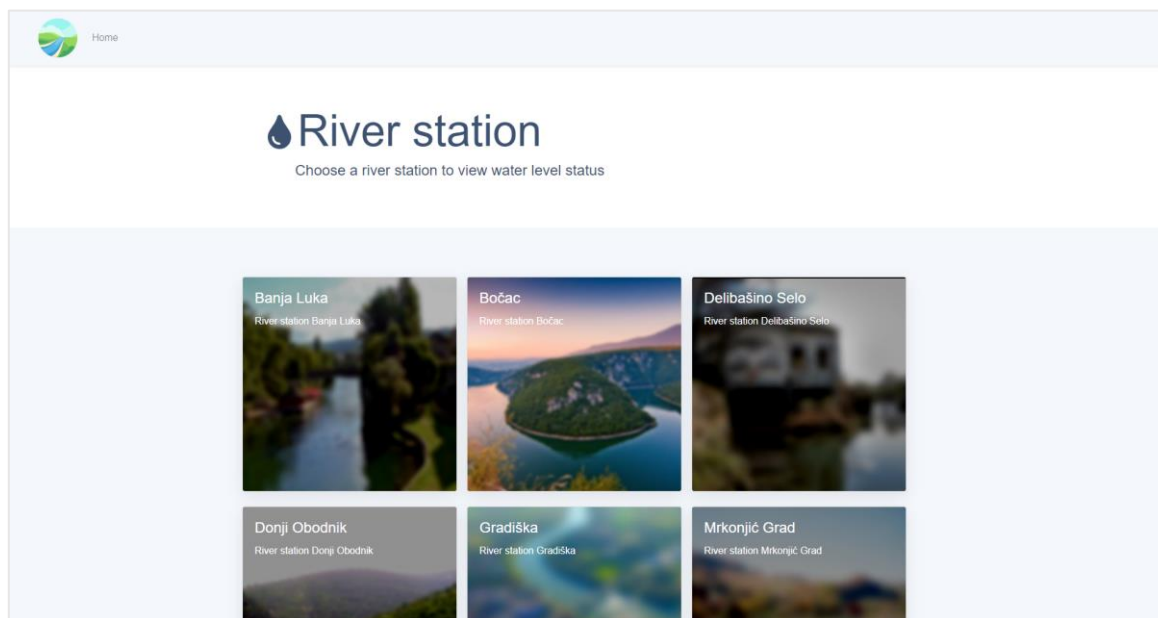
[ ] print(f"MSE: {metrics.mean_squared_error(y_test.values.ravel(), y_pred)}")
    print(f"MAE: {metrics.mean_absolute_error(y_test.values.ravel(), y_pred)}")

↗ MSE: 113.04589511165942
   MAE: 7.2256834339264255
```

Слика 5.12. Примјер обучавања модела са оптималним хиперпараметрима

Након што је одређен оптималан модел за сваки од алгоритама извршена је њихова евалуација. Након евалуације модела, најбољи модел је искоришћен за предвиђање нивоа водостаја у оквиру *web* апликације.

Web апликација се састоји од почетне стране на којој су приказане станице у сливу ријеке Врбас (слика 5.13).



Слика 5.13. Web апликација

Корисник, након што одабере станицу, добија приказ информација о станици (слике 5.14). Информације станице укључују тренутно стање водостаја заједно са термином мјерења (1), обавјештење о повишеном водостају и термину за који је он предвиђен (2), графички приказ стања водостаја у периоду од 24 сата (3), графички приказ предвиђеног нивоа водостаја у интервалу од 6 сати (4), снимак подручја за које је предвиђен повишен водостај (5). Снимање подручја се приказује само у случају ако је предвиђен повишен водостај. Кориснику је омогућено да путем *e-maila* добије обавјештење о почетку емитовања снимка на страници.



Слика 5.14. Демонстрација снимања станице у случају предвиђеног повишеног водостаја

Снимање станице је реализовано помоћу дрона DJI Mavic Air (слика 5.15) и *Red Waypoint*⁴⁵ андроид апликације.

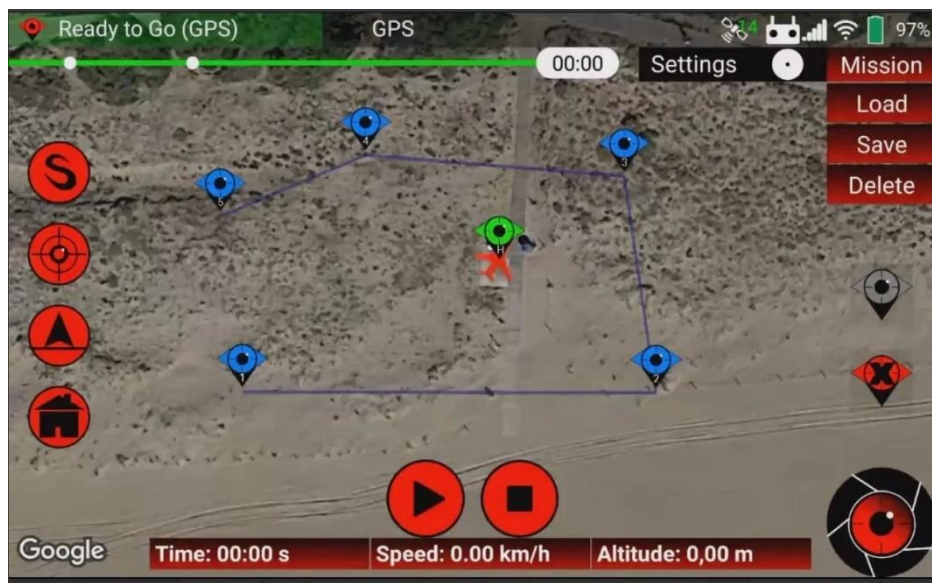
⁴⁵ <http://redwaypoint.com/>



Слика 5.15. DJI Mavic Air⁴⁶

Red Waypoint представља апликацију за управљање DJI дронова са подршком за аутономно летење помоћу кориснички дефинисаних путања.

Да би се омогућило аутономно летење потребно је дефинисати путању летења. Путања се дефинише помоћу путних тачака (енг. *waypoints*) (слика 5.16.) при чему је за сваку тачку могуће дефинисати параметре као што су брзина, висина и акција коју дрон треба да изврши (почетак/крај снимања, повратак на почетну тачку и сл).



Слика 5.16. Примјер дефинисане путање⁴⁷

Након дефинисања, путању је потребно сачувати и повезати дрон са апликацијом. Након повезивања потребно је учитати сачувану путању и покренути аутономно летење одабиром одговарајуће опције у оквиру апликације.

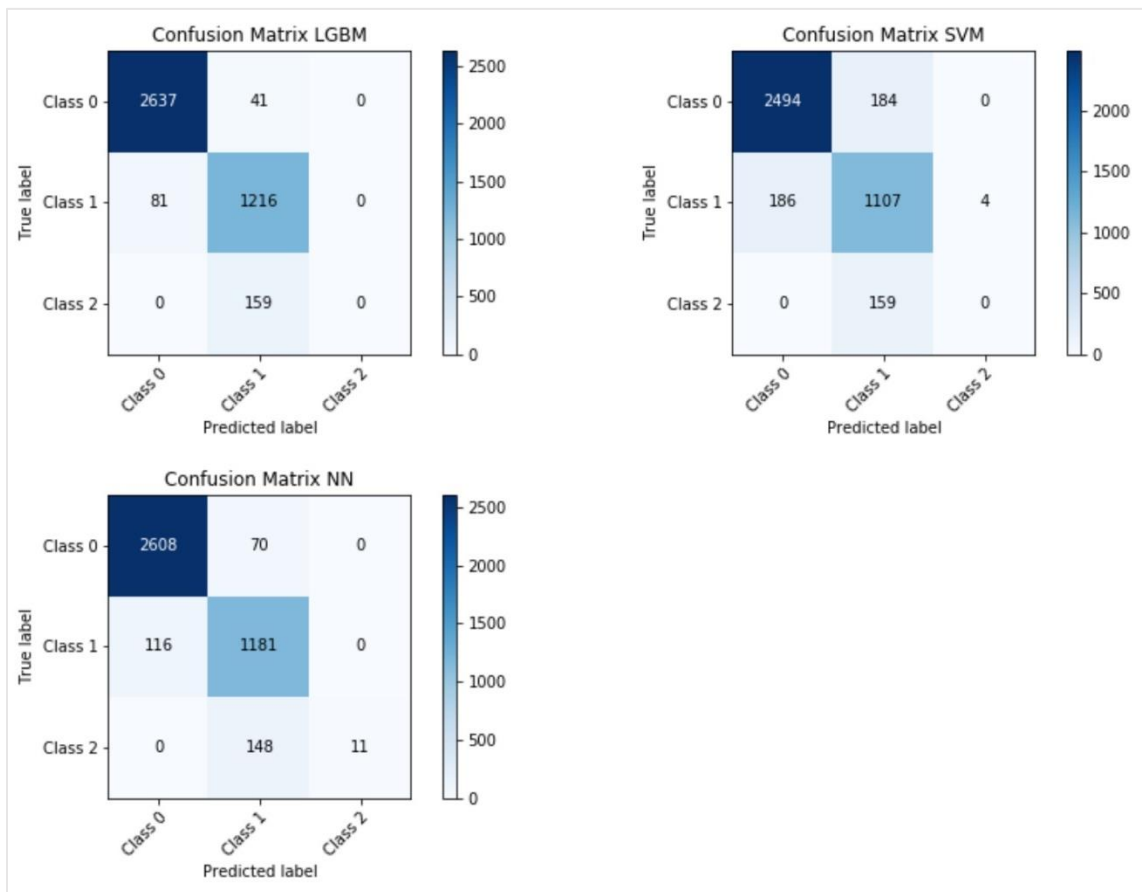
⁴⁶ <https://www.dji.com/mavic-air>

⁴⁷ https://play.google.com/store/apps/details?id=red.waypoint.RedWaypointPro&hl=en_US

Добијени снимак је постављен на YouTube и налази се на следећем линку:
<https://youtu.be/f8eUeAV62YI>.

6. РЕЗУЛТАТИ

Као што је објашњено у потпоглављу „Евалуација модела“ матрица конфузије омогућава бољи увид у перформансе модела. На слици 6.1. су приказане матрице конфузија за моделе који су базирани на алгоритмима из четвртог поглавља. С обзиром да су матрице димензија 3×3 може се закључити да тестни скуп не садржи догађаје који су означени као поплаве (класа 3). Број на позицији i, j представља број узорака класе i које је модел класификовао као узорке класе j .



Слика 6.1. Матрица конфузије LGBM, SVM, NN

У табели 6.1. је приказан удио тачно позитивних (TP), лажно позитивних (FP), лажно негативних (FN) и тачно негативних (TN) узорака, док је у табели 6.2. приказан њихов удио у односу на сваку класу.

На основу резултата из табеле можемо закључити да је сваки од класификатора успио тачно да класификује већину узорака, што је и доказано њиховом евалуацијом у табели 6.3.

| Model | True Positives (TP) | False Positive (FP) | False Negative (FN) | True Negative (TN) |
|-----------------|---------------------|---------------------|---------------------|--------------------|
| LightGBM | 3853 | 281 | 281 | 7987 |
| SVM | 3601 | 533 | 533 | 7735 |
| Neural Networks | 3800 | 334 | 334 | 7934 |

Табела 6.1. Удио тачно позитивних (TP), лажно позитивних (FP), лажно негативних (FN) и тачно негативних (FN) узорака

На основу резултата из табеле 6.3. може се закључити да је LGBM класификатор најбољи. Међутим, циљ обучавања је био да се добије класификатор који тачно класификује ријетке догађаје (поплаве). У табели 6.2. се види да је NN класификатор једини био у стању да тачно класификује ријетке догађаје (класа 2).

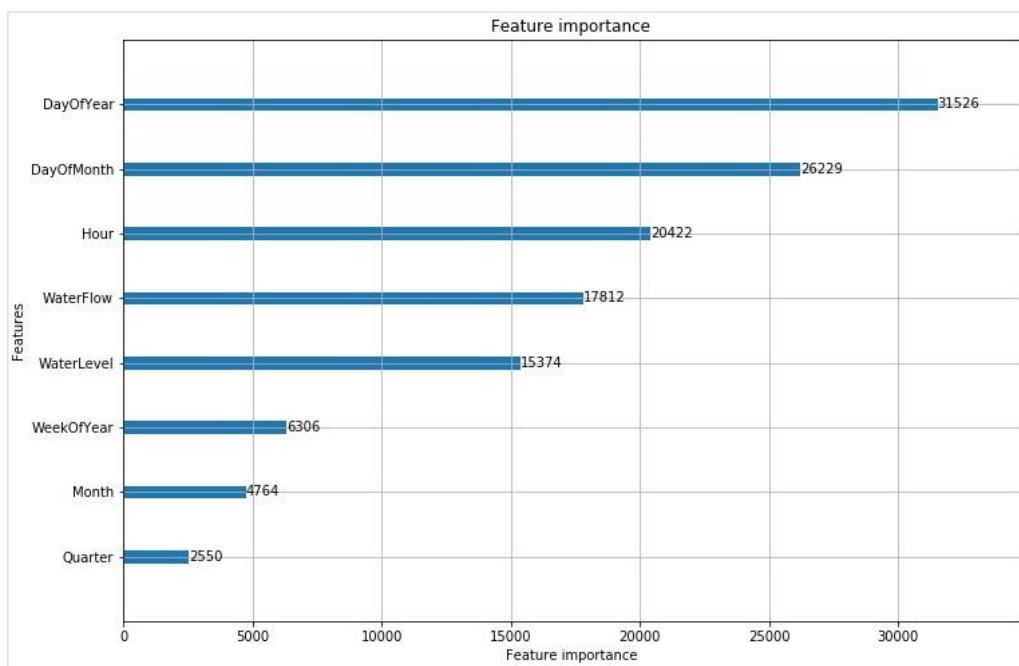
| Model | Class | True Positives (TP) | False Positive (FP) | False Negative (FN) | True Negative (TN) |
|-----------------|-------|---------------------|---------------------|---------------------|--------------------|
| LightGBM | 0 | 2637 | 81 | 41 | 1375 |
| | 1 | 1216 | 200 | 81 | 2637 |
| | 2 | 0 | 0 | 159 | 3975 |
| SVM | 0 | 2494 | 186 | 184 | 1270 |
| | 1 | 1107 | 343 | 190 | 2494 |
| | 2 | 0 | 4 | 159 | 3971 |
| Neural Networks | 0 | 2608 | 116 | 70 | 1340 |
| | 1 | 1181 | 218 | 116 | 2619 |
| | 2 | 11 | 0 | 148 | 3975 |

Табела 6.2. Удио тачно позитивних (TP), лажно позитивних (FP), лажно негативних (FN) и тачно негативних (FN) узорака за сваку класу

С обзиром да је од 159 узорака који представљају ријетке догађаје у тестном скупу NN класификатор успио да тачно класификује само 11 можемо закључити да овакав класификатор не би био погодан у пракси.

| Model | Accuracy | Recall | Precision | F1-Score |
|-----------------|----------|--------|-----------|----------|
| LightGBM | 0.93 | 0.93 | 0.93 | 0.93 |
| SVM | 0.87 | 0.87 | 0.87 | 0.87 |
| Neural Networks | 0.92 | 0.92 | 0.92 | 0.92 |

Табела 6.3. Перформансе класификатора



Слика 6.2. Значај обиљежја за LGBM модел

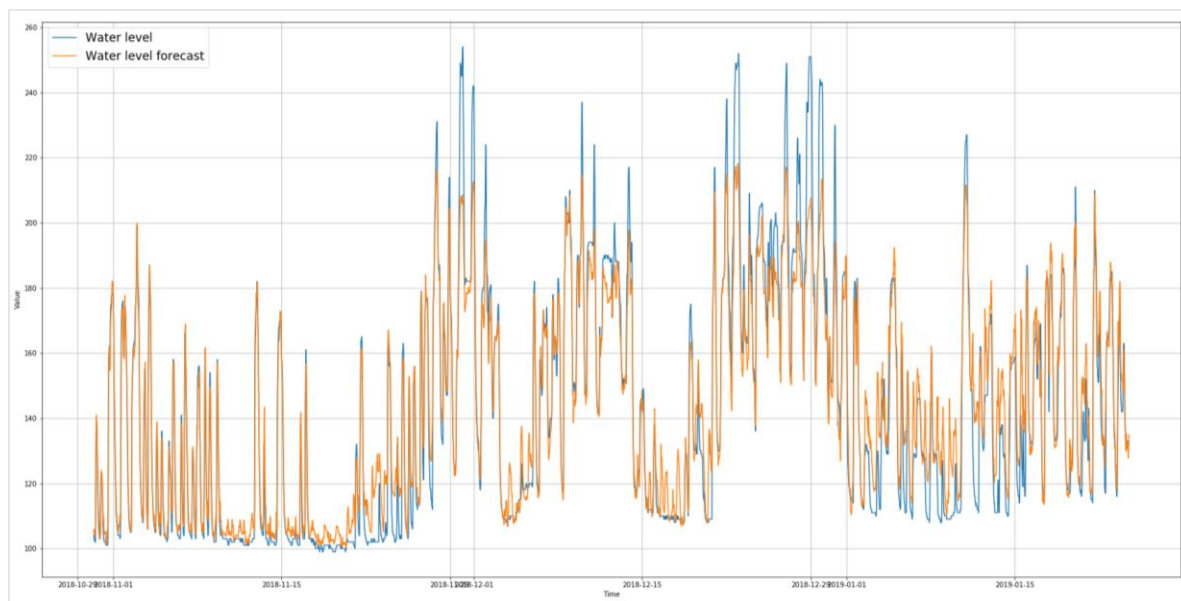
На слици 6.2. је дат графички приказ значаја обиљежја LGBM модела. С обзиром да обиљежје „Cat“ има велики утицај на LGBM модел оно није уклоњено у фази анализе података, иако је њен степен корелације са ријечним водостајем мали (слика 3.12).

Табела 6.4. садржи евалуацију перформанси модела обучених за предвиђање ријечног водостаја, док је на сликама 6.3 – 6.5. приказано предвиђање ријечног водостаја у односу на стварне вриједности за сваки од модела.

На основу резултата из табеле и приказаних предвиђања можемо закључити да је SVM модел најбољи.

| Model | Mean Squared Error (MSE) | Mean Absolute Error (MAE) |
|-----------------|--------------------------|---------------------------|
| LightGBM | 113.05 | 7.23 |
| SVM | 23.72 | 3.05 |
| Neural Networks | 730 | 18 |

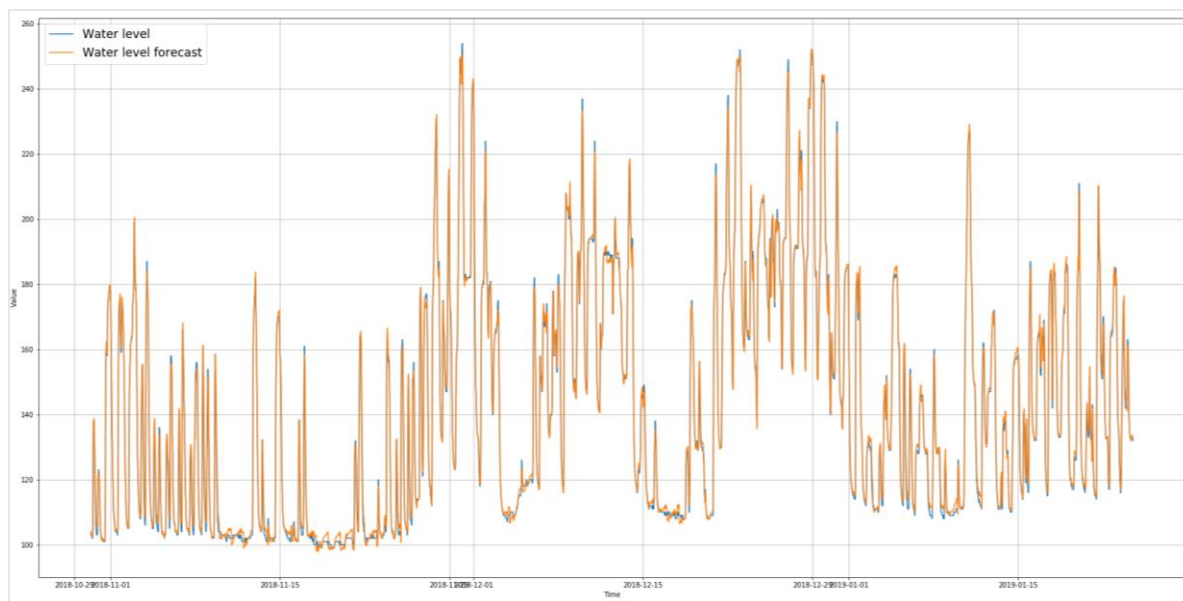
Табела 6.4. Перформансе модела за предвиђање ријечног водостаја



Слика 6.3. Предвиђања ријечног водостаја у односу на стварне вриједности LGBM

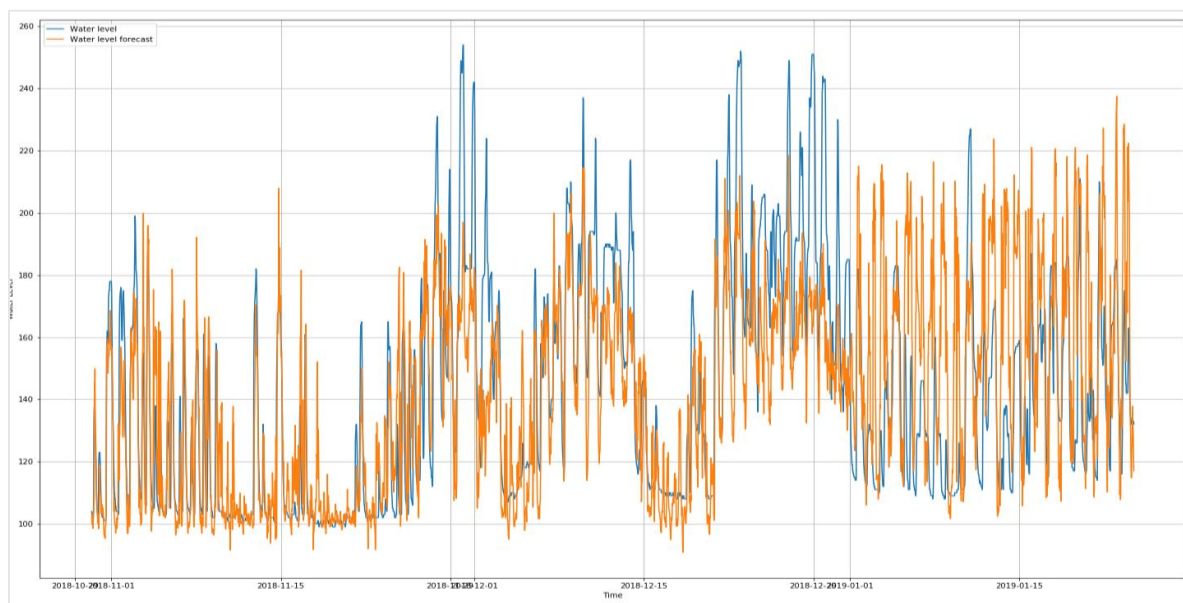
Међутим, треба узети у обзир чињеницу да су допуњени подаци добијени на основу вриједности будућих мјерења, што је омогућило моделима да „виде“ тестне податке. С обзиром на то, резултате из табеле 6.4. треба узети са опрезношћу, те не би требало очекивати да било који од обучених модела да приближне резултате у пракси. Додатни разлог зашто је SVM модел дао „одличне“ резултате је знатно мањи број хиперпараметара у односу на остала два модела што је омогућило једноставнију претрагу простора хиперпараметара у оквиру *HyperOpt* скрипте.

Такође, тестни скуп не садржи узорке који представљају поплаве за разлику од тренинг скупа што је омогућило моделима да добију знатно мање вриједности грешака у току евалуације него што је то био случај за вријеме обучавања и евалуације.



Слика 6.4. Предвиђања ријечног водостаја у односу на стварне вриједности SVM

Модел базиран на неуралним мрежама се на основу резултата из табеле 6.4. показао као најлошији што се може видјети и на слици 6.5. Разлог за то је превише мален скуп података. Познато је да модели дубоког учења дају одличне резултате када је скуп података велики.



Слика 6.5. Предвиђања ријечног водостаја у односу на стварне вриједности NN

Један од хиперпараметара за вријеме обучавања модела је била и архитектура мреже, међутим и са једноставнијом архитектуром мрежа није била у стању да оствари значајније резултате. Разлог за то је што LSTM ћелије нису у стању да памте дугорочне зависности. Такође ни изостављање LSTM слојева није довело до бољих резултата јер проширени конволуциони блокови, иако превазилазе проблем

дугорочних зависности, због малог скупа података нису били у стању да науче зависности између обиљежја и излаза као LGBM и SVM модел.

7. ЗАКЉУЧАК

Алгоритми машинског учења су у стању да ријеше сложене проблеме који су превише тешки или одузимају превише времена за човјека (нпр. филтрирање нежељених порука, предвиђање стања на берзи и сл.).

Дипломски рад је базиран на рјешавању проблема предвиђања ријечног водостаја (регресија) и класификације нивоа водостаја.

Приликом развоја рјешења базираног на машинском учењу пројекат пролази кроз 7 фаза: прикупљање података, обрада података, анализа података, избор алгоритма, обучавање модела, евалуација модела, тестирање модела и примјена модела.

Иако је сваки од описаних корака веома битан, посебно треба нагласити зависност модела од података. Уколико се корацима везаним за рад са подацима не посвети довољно времена, модел неће бити у стању да генерализује.

Приликом избора модела треба водити рачуна о величини скупа података и доступним ресурсима (процесор, графичка карта, меморија, итд.). Неуралне мреже данас представљају де факто стандард приликом избора модела. Међутим, приликом избора модела, треба узети у обзир њихове потребе за великом количином података и процесној моћи. У раду је показано да постоје и други модели који нису толико „скупи“ за обучавање, а дају боље резултате.

Лоше одабрана метрика може дати потпуно погрешну репрезентацију модела. Адекватна метрика се може одабрати само уколико је анализа података извршена исправно.

Да би се утврдиле перформансе модела неопходно је податке подијелити у три дисјунктна скупа. У супротном, модел би био у стању да „запамти“ резултате у току обучавања, те да, приликом евалуације, постигне одличне резултате, а у примјени катастрофалне. Тренинг скуп се користи за обучавање, валидациони скуп за избор хиперпараметара, а тестни скуп за евалуацију (стварних) перформанси модела.

Уколико је модел након тестирања постигао довољно добре резултате онда је спреман да се примијени у пракси за рјешавање проблема у реалном времену.

На основу резултата добијених у дипломском раду може се закључити да нити један од модела није употребљив у пракси. Главни разлог за то је велики број недостајућих вриједности у скупу података као и мали број обиљежја из којих су модели требали да науче зависности са излазом.

ЛИТЕРАТУРА

- [1] М. Николић, А. Зечевић *Машино учење*, Београд, 2019.
- [2] В. Рисојевић *Мултимедијални системи*, Универзитет у Бањој Луци: Електротехнички факултет, 2018.
- [3] F. Giasson, "A *Machine Learning Workflow*," 10.03.2017. <http://fgiasson.com/blog/index.php/2017/03/10/a-machine-learning-workflow/>, посјећено 24.10.2019.
- [4] A. Pant, "Workflow of a *Machine Learning project*," 11.01.2019. <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>, посјећено 24.10.2019.
- [5] C. University, *Missing-data imputation*, Columbia University .
- [6] A. van den Oord, S. Dieleman, H. Zen, "Wavenet: A generative model for raw audio", <https://arxiv.org/pdf/1609.03499.pdf>, посјећено: 24.10.2019. године
- [7] M.Patel, A.Patel, Dr. R.Ghosh, "Precipitation Nowcasting: Leveraging bidirectional LSTM and 1D CNN", <https://arxiv.org/ftp/arxiv/papers/1810/1810.10485.pdf>, посјећено: 24.10.2019. године
- [8] Ruslana Dalinina, Jean-René Gauthier, and Pramit Choudhary, "Testing Predictive Models in Production," <https://www.oracle.com/a/ocom/docs/oracle-ds-testing-predictive-models-in-production.pdf>, посјећено: 24.10.2019. године
- [9] A. Zheng, *Evaluating Machine*, Sebastopol: O'Reilly Media, Inc. , 2015.
- [10] C. Cortes and V. Vapnik, "Support Vector Networks", http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf, посјећено: 24.10.2019. године
- [11] G.Ke, Q.Meng, T.Finley, T.Wang, W.Chen, W.Ma, Q.Ye, T-Y.Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," Neural Information Processing Systems Conference
- [12] S.Bai, J.Z.Kolter, V.Koltun, , "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", <https://arxiv.org/pdf/1803.01271.pdf>. посјећено: 24.10.2019. године
- [13] C.Olah, "Understanding LSTM Networks," 27.08.2015. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. посјећено: 24.10.2019. године