Универзитет у Београду

Математички факултет

Мастер рад

Имплементација додатка за Интернет прегледач за препознавање лажних вести

Студент:
Александар Шука

Ментор:
др Александар Картељ

Ментор:

др Александар Картељ, доцент

Математички факултет, Универзитет у Београду

Чланови комисије:

др Филип Марић, вандредни професор

Математички факултет, Универзитет у Београду

др Јелена Граовац, доцент

Математички факултет, Универзитет у Београду

Садржај

1 Увод

Живот се у великој мери преселио на Интернет. Људи су одлучили да путем Интернета једноставније и комфорније обављају своје послове и споредне животне навике. Једна од основних животних потреба код модерног човека је информисање, и природно је да се и начин информисања због приступачности премести на Интернет. Најбитнија карактеристика неке информације је њена истинитост или валидност. Поставља се питање какав утицај на истинитост има овакав једноставнији, комфорнији и приступачнији начин информисања.

Некада су једини начини информисања били писаним путем или жива реч. Када се каже писаним путем онда се обично мисли на вести кроз новинске чланке. За објављивање једног новинског чланка одговорна је новинска агенција где је укључено више новинара, уредника, под-уредника и на крају главни уредник. Поред осталих ствари, њихова улога је проверавање валидности чланака које желе да објаве.

Када је жива реч у питању, где информацију добија уживо од других људи, човек искуствено на основу карактеристика саговорника може до неке мере да одреди ниво валидности информације коју добија. Тако човек одлучује да је валидности информације коју добија од блиског пријатеља, неког ко има карактеристике особе која говори истину или овлашћеног лица, већа од валидности информације коју добије од особе која има сумњиве карактеристике.

Данас због експанзије информација на Интернету људи све ређе купују новине, а ширењем и прихватањем мобилних уређаја људима је врста информисања преко Интернета постала приступачнија него жива реч. Појединац у друштву још увек нема осећај како да приступа оваквим вестима, иако је Интернет званично настао 1970-тих година он тек сада постаје доступан на свим друштвеним нивоима.

Овакво окружење је постало погодоно за разне политичке, друштвене и комерцијалне манипулације. Оне су постале толико распрострањене да је термин лажна вест (енгл. *fake news)* прихваћен као синоним за дезинформације које се врше путем Интернета.

Спекулише се да су лажне вести биле један од кључних фактора који је утицао на резултате избора у Сједињеним Америчким Државама 2016. године [1]. Моћан утицај лажних вести постаје још опаснији када се на оглашавање примене технике машинског учења, где се сваком појединцу сервирају оне информације које ће га учинити више ангажованим и натерати га да више времена проведе читајући вести. Ово доводи до великих ефеката на поларизовање у друштву које историјски гледано може да има бурне реакције [2].

Лажне вести утичу на појединца па самим тим и на групу људи или популацију и на њихове међусобне односе. Са тим у виду, јако је битно развити технике које би помогле човеку да оцени валидност пласираних информација. Претходно наведено је да се технике

машинског учења користе како би се појединцу представиле информације најинтересантније њему. Технике машинског учења се могу искористити и за препознавање оваквог садржаја, оне су погодне за праћење фреквенције речи и врста речи, могу разликовати у тексту лична мишљења наспрам чињеница (лична мишљења теже да имају мању валидност у односу на чињенице). У наредним поглављима су детаљно описане технике машинског учења које се уз помоћ датог текста, без додатних информација, користе за решавање предложеног проблема класификације лажних вести на Интернету.

2 Преглед техника за решавање проблема

За овај рад су изабране и упоређене две технике машинског учења, једна једноставна, линеарна регресија и једна сложенија, метод подржавајућих вектора. У наредним поглављима описан је општи поглед на машинско учење и детаљно су описане обе технике. Такође, описане су и методе које се користе у обради природних језика као и предложена имплементација у програмском језику Python заједно са два различита скупа података.

2.1 Основне дефиниције машинског учења

Машинско учење је подобласт вештачке интелигенције која омогућава софтверској апликацији да постане прецизнија у предвиђању исхода, а да није експлицитно испрограмирана. То најчешће подразумева моделовање компликованих функција, а најчешћа варијанта је такозвано надгледано учење, у којем се на основу тренинг података врши одлучивање или предвиђање података за које није познат циљ.

Коришћење машинског учења ради одлучивања или предвиђања би требало да пронађе скривене обрасце и везе, и тиме ухвати генеричка правила која важе међу подацима. Овакав вид генерализације је битан јер је карактеристичан за природну интелигенцију и скоро га је немогуће експлицитно испрограмирати.

Подаци над којима се врши предвиђање су најчешће представљени као вектори. Сваки вектор садржи вредности променљивих које се називају атрибутима и квантитативно су мерљиви, тако неки од атрибута могу бити висина, број година, просечна плата, итд. Код слика атрибути могу бити вредности пиксела за ту слику. У случају када су подаци категорички, лако се могу бинаризовати и на тај начин применити исти приступ. Вектор појединачних улазних података се обично обележава $\boldsymbol{x} \in \mathbb{R}^n$, а i-ти атрибут тог податка као x_i. Скуп свих улазних податак $\boldsymbol{x}$ обележава се са $\boldsymbol{X}$, у том случају са x_i^j се означава i-ти атрибут j-тог улазног податка.

Резултат предвиђања је циљна променљива, обично се означава са y, скуп свих циљних променљивих је $\boldsymbol{Y}$, док се као y^j означава j-та циљна променљива. Постоје различити типови предвиђања као што су класификација, регресија, детектовање аномалија

и друге, у зависности у којем контексту се врши предвиђање. Класификација је врста предвиђања код које је циљна променљива категоричка. Модел покушава да предвиди којој од категорија податак припада, на пример да ли је тумор на основу података бенигни или малигни, да ли је дати текст истинит или лажан, препознавање саобраћајних знакова или цифара на слици. У случају да је број циљних класа два, онда се ради о бинарној класификацији, у случају да је број класа већи од два онда је то вишекласна класификација. За разлику од класификације, код регресије циљна променљива је нумеричка и користи се приликом проблема као што су предвиђање цена деоница на берзи, рачунање оптималне брзине возила.

Процес конструкције функције предвиђања се популарно назива учење. По начину на којем се моделовање врши проблеми се угрубо могу поделити на проблеме надгледаног учења, ненадгледаног учења и учења поткрепљивањем. Надгледано учење, предиктивно учење, је проблем код кога се модел тренира применом скупа података, тренинг података, за које су познате вредности атрибута, као и вредности циљне променљиве. Код проблема ненадгледаног учења није задата вредност циљне променљиве, већ модел треба да уочи везе, шаблоне, у оквиру задатих атрибута. Код учења поткрепљивањем је реч о раду интелигентних агената у датком окружењу, који своје акције прилагођавају тако да максимизују кумулативну награду у одређеном временском периоду.

2.2 Надгледано учење

Један од начина да се формализује надгледано учење је путем теорије о апроксимацији функција [3]. Дакле, за задати вектор атрибута $\boldsymbol{x}$ и циљну променљиву y, постоји функција f таква да је $y = f(\boldsymbol{x})$. Циљ је апроксимирати функцију f неком функцијом f', тако да се за формирање функције f' користе расположиви подаци. Притом је потребно да f' добро уопштава, то јест да f' добро апроксимира функцију f, односно предвиђа вредности циљне променљиве и на подацима који нису расположиви. Ово се обично постиже тако што се скуп свих података подели на два дела, један део над којим се модел тренира, тренинг подаци, и други део над којима ће се проверавати квалитет добијеног модела, тест подаци.

Како је тренинг скуп ограничен и у пракси је обично тешко наћи јасну везу између атрибута и циљне променљиве, проблем се може посматрати и из угла вероватноће, где је однос између атрибута и циљне променљиве је задат заједничком расподелом вероватноће $p(\boldsymbol{x}, y)$.

За проналажење f' које најбоље предвиђа потребно је дефинисати меру квалитета предвиђања. Потребно је да $y \sim f'(\boldsymbol{x})$, односно да је разлика праве вредности циљне променљиве и њене апроксимације што мања. За то се користи фукција губитка (енгл. *loss function*), $L(y, f'(\boldsymbol{x}))$ која представља разлику између праве вредности и предвиђене.

С обзиром да није довољно испитати грешку само на једном пару праве и предвиђене вредности, да би се добила информација какав је модел, потребно је проћи кроз што већи

скуп таквих парова. Ако се узме цео скуп расположивих података $D = (\boldsymbol{x}^j, y^j)$, где $j = 1, \ldots, N$, онда се грешка може дефинисати као $E(f', D) = \frac{1}{N} \sum_{j=1}^{N} L(y^j, f'(\boldsymbol{x}^j))$. Као што је већ раније наглашено, ово још увек није довољно да модел добро уопштавао, па је потребно некако „симулирати“ нерасположиве, будуће, податке. Ово се решава поделом скупа D на два дисјунктна дела, први подскуп D_{train} и други подскуп D_{test}. Грешка над тест скупом се још назива и грешка уопштавања, или грешка генерализације.

За потребе побољшања грешке уопштавања може се користити и техника валидације. Она подразумева да се одвоји један део тренинг података и додатно подели на тренинг и валидациони скуп. Модел се тренира над тренинг подацима и приликом тренирања проверава његов квалитет над валидационим скупом података. Валидациони подаци „глуме“ будући тест скуп. Такође, као још софистицираније техника се може користити техника унакрсне валидације. Идеја је да се скуп података подели на K подскупова, да се тренинг врши над $K - 1$ скупу, а K-ти скуп да се користи као тест скуп, овај корак се понавља K пута за сваки од K подскупова. Када се грешка израчуна за сваки од K подскупова, она се упросечи и резултат је грешка која се касније користи као квалитет модела. Мана оваквог приступа је што корак тренирања се понавља K пута, и може бити временски захтевно.

2.2.1 Линеарна регресија

Већина техника које се користе за апроксимацију функција у надгледаном учењу имају параметре (тежине) које је могуће мењати како би се прилагођавао модел. Најједноставнији такав модел је линеарна регресија. Параметри се обично озачавају са $\boldsymbol{w}$ или $\boldsymbol{\theta}$. Функција линеарне регресије има следећи облик:

$$F_w(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Обично се вектор атрибута $\boldsymbol{x}$ прошири са 1, које се додаје на почетак вектора. У векторском запису изгледа:

$$F_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x}$$

Модуће је дефинисати модел који је линеарна комбинација нелинеарних функција по x. Одабрана функција (нелинеарна) обично се назива базична функција:

$$F_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \Phi(\boldsymbol{x})$$

Коришћење оваквог модела омогућава моделовање сложенијих скупова података, при чему још увек важе једноставности у израчунавању које допушта линеаран модел по

параметру w. И овакав модел наилази на ограничења, наиме модел зависи од избора функције Φ, одабир функције Φ се врши пре него што се погледа скуп података који се моделује.

Неки од избора базичне функције су:

- Полиномијална функција, $\Phi(x) = x^t$
- Сигмоидна функција, $\Phi(x) = \frac{1}{(1+e^{-t})}$

Нелинеарност се у неким ситуацијама може увести и накнадно:

$$F_{\boldsymbol{w}}(\boldsymbol{x}) = \Phi(\boldsymbol{w}^T \cdot \boldsymbol{x})$$

Техником максималне веродостојности се може показати да се тражење оптималних вредности вектора $\boldsymbol{w}$ своди на минимзацију средње квадратне грешке, дате следећим изразом [4].

$$\min \sum_{j=1}^{N} \left(y^j - F_{\boldsymbol{w}}(\boldsymbol{x}^j) \right)^2$$

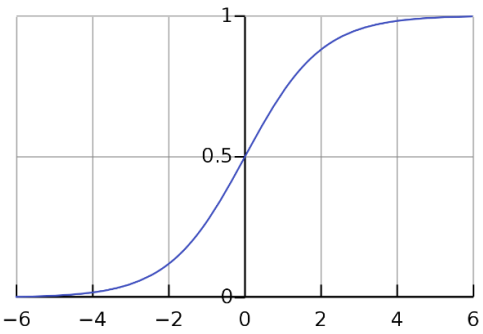
2.2.2 Логистичка регресија

Логистичка регресија је линеаран модел и користи се у случају када циљна променљива узима вредности из скупа $y \in \{0, 1\}$. Иако циљна променљива код логистичке регресије има бинарну вредност, регресија у називу стоји да би се нагласила сличност са линеарном регресијом.

Функција која узима вредности из опсега $0 \leq \Phi(x) \leq 1$ и користи се за логистичко регресију је сигмоидна функција, још се назива и логистичка функција.

$$\Phi(x) = \frac{1}{1 + e^{-t}}$$

Приказана је на слици:



Слика 2.1 Приказ сигмоидне функције

И у случају логистичке регресије, методом максималне веродостојности може се доћи до израза који је потребно минимозовати како би се одредили коефицијенти $\boldsymbol{w}$.

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \sum_{j=1}^{N} \left(y^j - \Phi\left(\boldsymbol{w} * \boldsymbol{x}^j\right)\right) * \boldsymbol{x}^j$$

2.2.3 Метод подржавајућих вектора

Код претходно наведених метода циљ је био одредити параметре модела коришћењем технике максималне веродостојности, са предпоставком да циљна променљива за одређене податке долази из нормалне расподеле. Метод подржавајућих вектора (енгл. *Support Vector Machine – SVM*) је техника машинског учења која спада у групу линеарно дискриминативних метода са додатним условом о максималности маргине [3].

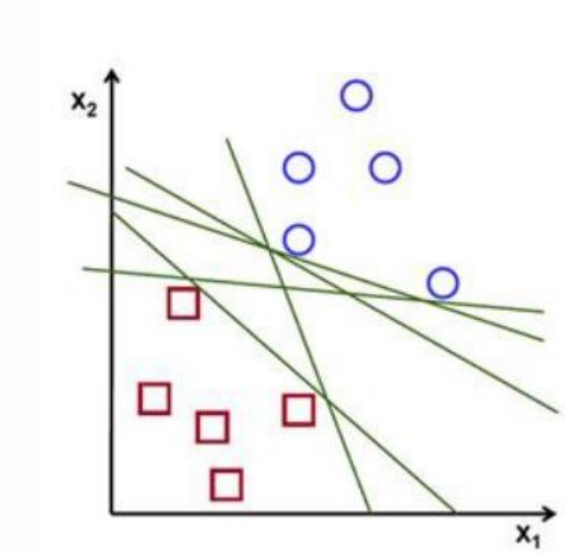
Општа форма линеарне дискриминативне фунцкије је:

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Где су $w_i \in \mathbb{R}$, $i = 1, \ldots, N$ тежине додељене атрибутима вектора, док је w_0 слободан члан. Функција f дефинише хиперраван (маргину) која раздваја области припадности две класе. Уобичајена функција одлучивања је дата са:

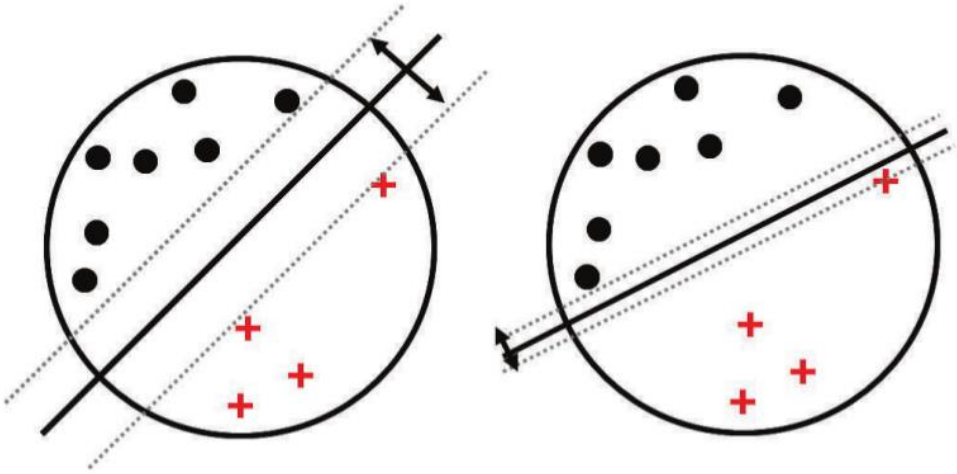
$$c(\boldsymbol{x}) = \begin{cases} -1, & f(\boldsymbol{x}) < 0 \\ 1, & f(\boldsymbol{x}) \geq 0 \end{cases}$$

У случају да су подаци из тренинг скупа линеарно сепарабилни, могуће је наћи такав вектор коефицијената $\boldsymbol{w} = (w_0, \ldots, w_N)$ за који важи да, након што се уврсти у функцију f, функција одлучивања даје конкретне резултате за све тренинг податке. Како овај метод узима у разматрање само линеарну раздвојивост података, за оптимизациони проблем коефицијената може се искористити градијентни спуст или Њутнове метода.



Слика 2.2 Приказ хиперравни (маргина) које коректно раздвајају линеарно сепрабилне податке.

Јасно је да вектор коефицијената није јединствен, па је циљ пронаћи ону раздвајајућу хиперраван која је максимално удаљена од свих тренинг података (вектора) са обе стране. Ово својство је кључно за метод подржавајућих вектора.



Слика 2.3 Лево је приказ хиперравни (маргине) која је максимално удаљена од најближе тачке са обе стране.

Нека је дат скуп података D_{train} сачињен од парова облика $(\boldsymbol{x}^j, y^j), j = 1, \dots, N_{train}$, где $x^j \in \mathbb{R}^N$ представља N-димензионални вектор вредности по сваком од улазних атрибута, а $y^j \in \{-1, 1\}$ његову припадајућу класу. SVM користи тренинг скуп D_{train} у циљу проналажења раздвајајуће хиперравни $\boldsymbol{w} * \boldsymbol{x} + w_0 = 0$, која је максимално удаљена од свих тренинг података са обе стране и да за све податке важи $y^j * (\boldsymbol{w} * \boldsymbol{x}^j + w_0) \geq 1$.

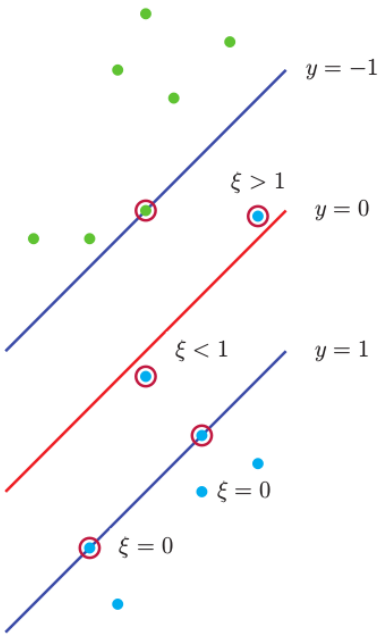
Како су линеарно раздвојиви подаци врло ретки у пракси, постоји потреба за допуштањем одређене грешке по питању раздвојивости података. Мање стриктна варијанта је базирана на мекој маргини (енгл. *soft margin*), која дозвољава подацима (векторима) да се налазе са погрешне стране, али ипак довољно близу. Уводи се ненегативна променљива ξ_j која представља грешку преласка $\boldsymbol{x}^j$ на погрешну страну, $y^j * (\boldsymbol{w} * \boldsymbol{x}^j + w_0) \geq 1 - \xi_j$. Оптималне вредности за $\boldsymbol{w}$ и w_0 могу бити пронађене решавањем следећег оптимизационог проблема:

$$\min\left(\frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + C\sum_{j=1}^{N_{train}}\xi_j\right),$$

уз ограничења:

$$y^j * (\boldsymbol{w} * \boldsymbol{x}^j + w_0) \geq 1 - \xi_j, j = 1, \dots, N_{train}$$
$$\xi_j \geq 0, j = 1, \dots, N_{train}$$

C је регуларизациони параметар који контролише утицај грешке прелажења.



Слика 2.4 Приказ хиперравни која раздваја податке уз допуштање грешке.

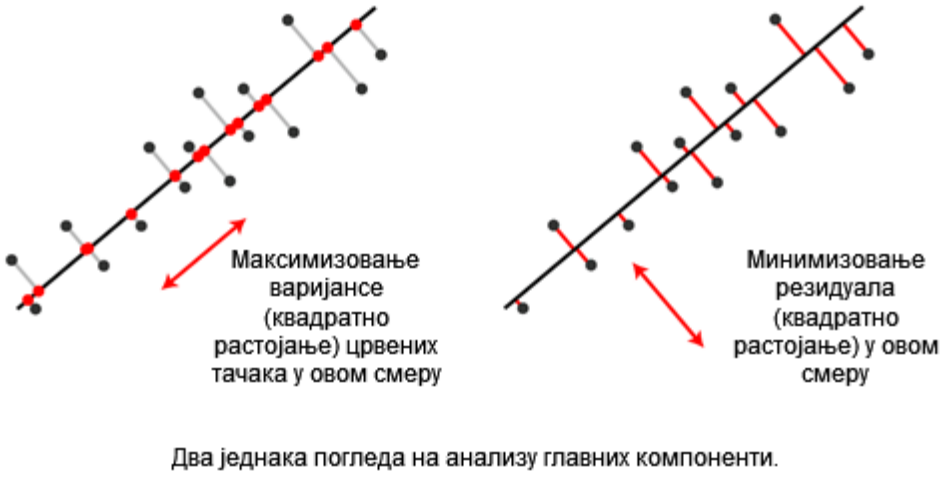
За решавање проблема у случају када су подаци потпуно нелинеарни користи се кернелска варијанта *SVM*-а [4].

2.3 Ненадгледано учење

Ненадгледано учење се односи на коришћење алгоритама вештачке интелигенције за проналажење образаца у скуповима података за које није позната вредност циљне промењиве. Алгоритмима је стога дозвољено да класификују, озаче или групишу податке без икаквих спољашњих смерница. Другим речима, ненадгледано учење омогућава систему да сам идентификује обрасце унутар скупова података.

2.3.1 Анализа главних компоненти

Анализа главних компоненти (енгл. *Principal Component Analysis*) се заснива на претпоставци да подаци услед линеарних зависности или корелације, па и саме природе података, не попуњавају униформно цео простор атрибута, већ се пружају дуж неког његовог линеарног подпростора мање димензије од полазног. Када би се подаци представили као ортогоналне пројекције на тај подпростор и уколико би се даље радило са тим репрезентацијама, могли би се умањити негативни ефекти коју висока димензионалност и корелисаност атрибута имају на методе машинског учења [7]. Циљ је пронаћи ортогоналну пројекцију у нови потпростор којом се губи што мање информација. Информација се чува тако што се максимизује варијанса између података након пројекције или аналогно томе минимизује средњеквадратно растојање између података и њихових пројекција.



Слика 2.2 Приказ ортогоналне пројекције података из простора 2 димензије на подпростор димензије 1.

Да би се подаци пројектовали на простор мање димензије, и притом максимизовала варијанса између пројектованих података, узима се скуп података $\boldsymbol{X} = \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^N\}$, док је $\boldsymbol{x}^j$

димензионалности K, $\boldsymbol{x}^{\mathrm{j}} = \{x_1^j, \dots, x_K^j\}$. Почиње се од тога што се врши пројекција на једнодимензиони простор чији смер се може дефинисати јединичним D-димензионим вектором $\boldsymbol{u}_1$, за који важи $\boldsymbol{u}_1^T\boldsymbol{u}_1 = 1$. У том случају свака тачка се слика у скалар $\boldsymbol{u}_1^T\boldsymbol{x}^j$. Средња вредност пројектованих података је $\boldsymbol{u}_1^T\overline{\boldsymbol{x}}$, где је $\overline{\boldsymbol{x}}$ задато једначином:

$$\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{x}^j$$

Док је варијанса у том случају:

$$\frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{u}_1{}^T\boldsymbol{x}^j - \boldsymbol{u}_1{}^T\overline{\boldsymbol{x}}) = \boldsymbol{u}_1{}^T\boldsymbol{S}\,\boldsymbol{u}_1$$

где је $\boldsymbol{S}$ матрица коваријансе:

$$\boldsymbol{S} = \frac{1}{N}\sum_{j=1}^{N}\left(\boldsymbol{x}^j - \overline{\boldsymbol{x}}\right)\left(\boldsymbol{x}^j - \overline{\boldsymbol{x}}\right)^T.$$

Ово је оптимизациони проблем, у којем се максимизује $\boldsymbol{u}_1{}^T\boldsymbol{S}\,\boldsymbol{u}_1$ по $\boldsymbol{u}_1$. Након што се изврши оптимизација по $\boldsymbol{u}_1$, она се фиксира постаје такозвана прва компонента. Следеће компоненте се добијају итеративним поступком у којем се поново максимизује варијанса уз задржавање претходно израчунатих компоненти. На крају се добија потпростор описан скупом компоненти, то јест векторима $\boldsymbol{u}_1, \dots, \boldsymbol{u}_m$.

3 Обрада природних језика

Обрада природних језика је подобласт вештачке интелигенције која се бави изучавањем техника којима рачунар може да опонаша, разуме или моделује људски говор, писање и комуникацију [8].

Први корак у обради неког језика је нормализација текста, односно претворити текст у формат са којим машина може што боље да барата. Како је немогуће од једном радити са читавим текстом, а неефикасно је радити са појединачним карактерима, први корак у нормализацији текста је токенизација текста. У већини природних језика токени су заправи речи које су одвојене размаком или знаком интерпункције. Код језика као што је јапански или кинески, где речи нису одвојене размаком, процес токенизације бива доста компликованији. Само идентични токени из угла рачунара представљају исте речи, иако се зна да на пример речи као што су “play”, “plays”, “playing” имају скоро исто значење. Поред тога што се сличне речи из угла рачунара гледају као различити токени, такође проблем прави и увећавање броја токена чиме се увећава димензија улаза.

Претходно наведени проблем се превазилази техником нормализације која се назива стемовање (енгл. *stemming*). Ова техника коришћењем регуларних израза проналази корени облик токена, а затим све сличне речи пресликава у базни токен. У претходном примери речи “play”, “plays”, “playing” слика у токен “play”. Напреднија техника којом се утврђује да две речи имају исти токен, упркос површним разликама, назива се лематизација. На пример речи “sang”, “sung”, “sings” су све облизи речи “sing”. Реч “sing” је основна лема за ове речи, а лематизатор пресликава све њих у “sing” [10].

Овако нормализован текст је улаз за даље технике обраде природних језика. У следећој глави „Претпроцесирања података“ је дат детаљан опис техника које се најчешће користе код проблема класификовања текстова.

3.1 Претпроцесирање текста

У случајевима класификовања података, где су подаци представљени као скуп текстова, методе машинског учења предложене у поглављу „Преглед техника за решавање проблема“ постају неупотребљиве, јер као улаз очекују вектор. Циљ претпроцесирања података је да податке дате као скуп текстова преслика у векторе. Свако пресликавање које текст претвара у нумеричке атрибуте се може искористити као техника претпроцесирања. Међутим неке технике имају више смисла од других у зависности од тога колико добро раздвајају класе. Технике претпроцесирања такође спадају у обраду природних језика, а у наставку биће детаљно описане технике претпроцесирања искоришћене приликом имплементације овог рада.

3.1.1 Технике претпроцесирања текста засноване на скупу речи и фреквенцији појављивања речи

Један интуитиван начин како скуп текстова (докумената) претворити у вектор је креирањем корпуса речи – *Bag of Words* или кратко BOW. Добијени вектор на i-тој позицији садржи информацију колико се пута i-та реч (према некој нумерацији) појавила у скупу текстова. Зарад ефикасног приступа траженој речи, то јест њеном броју појављивања, за похрањивање информација о броју појављивања речи се користи структура података речник. Нека речник L садржи N речи. Претварањем M скупа докумената D у вектор добија се матрица величине $M \times N$, где су редови документи, а колоне речи.

Следећа матрица одговара документима (текстовима): D_1: "Joe woke up late, so Joe was late for train", D_2: "Joe arrived late at office".

	Joe	Woke	up	Late	So	Was	For	Train	arrived	at	Office
D_1	2	1	1	2	1	1	1	1	0	0	0
D_2	0	0	0	1	0	0	1	0	1	1	1

Табела 3.1 Речник L, који представља бекторски приказ текстова D_1 и D_2 из документа D на основу броја појављивања речи.

Предност ове технике је што је једноставна, и полази од претпоставке да документи који имају сличне речи морају имати слично значење.

Недостатак ове технике је што у реалним применама матрица постаје ретка, где речник има велики број речи, а у документима се појављују само неке речи, тако да број појављивања већине речи је 0 и подаци попуњавају само мали део векторског простора.

Још један недостатак ове технике је тај што за меру сличности текстова узима у обзир и речи које се често појављују у свим документима, као што су везници, прилози, помоћни глаголи, и слично, а оне нису занимљиве јер ако је реч честа у свим документима онда та реч нема велики утицај на стварно класификовање докумената.

Технике претпроцесирања текста засноване на фреквенцији појављивања речи

Начин који се у пракси показује као бољи је начин заснован на фреквенцији речи и инверзној фреквенцији докумената ($tfidf$). Како није довољно да се прати само појављивање речи у неком документу због недостатка претходно наведене технике, потребно је пратити и појављивање речи у осталим документима. Фреквенција докумената df [9] за неку реч у документу се дефинише као број докумената у којима се појављује та реч. Ако је укупан број докумената N, инверзна фреквенција документа idf неке речи t се дефинише као:

$$idf_t = \log \frac{N}{df_t}$$

Из претхофне формуле види се да је у случају ретких речи вредност инверзне фреквенције документа велика, док је у случају честих речи вредност мала.

Нека се укупан број речи у неком документу d озачава са T и број појављиваха речи t у документу d означава са $n_{t,d}$, фреквенција неке речи t у документу d дефинише се као:

$$tf_{t,d} = \frac{n_{t,d}}{T}$$

Узимајући у обзир претходне формуле дефинише се мера $tfidf$ којом се документ претвара у вектор на следећи начин:

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

Другим речима $tfidf$ додељује вредности речи t у документу d тако да:

1. *Вредност $tfidf$ је највећа за речи t које се појављују много пута у малом броју докумената (то значи да такве речи носе велику информацију за описивање тог скупа докумената).*
2. *Вредност $tfidf$ је мала за речи t које се појављују мало пута у документу или се појављују у много докумената.*

У пракси број најфреквентнијих речи које ће се пратити бира се валидацијом као један од хипер-параметара модела.

Недостатак ове технике је што са њеном применом губи се редослед речи у реченици, па самим тим и део семантике.

Једно од унапређења ове технике је праћење и фреквенције појављивања N-грама. N-грам је скуп речи најчешће дужине 2 (биграм) или 3 (триграм), и фреквенција појављивања се прати на исти начин као праћење фреквенције појављивања једне речи, где би N у овом случају имало вредност 1 (униграм). Праћењем више речи у склопу термина чува се контекст у коме се реч појављује и тиме чува један део семантике.

3.1.2 Технике претпроцесирања текста засноване на плиткој синтакси

Још један начин како документ претворити у вектор је праћењем фреквенције појављивања врста речи, на енглеском *part-of-speech* или кратко *POS*. Основне врсте речи су именице, глаголи, придеви, прилози, али такође се могу пратити и фреквенције предлога, везника, речца, бројева, заменица. Ова техника помаже у отклањању препрека из претходне

технике, технике засноване на праћењу фреквенција појављивања речи, где се исте речи појављују у различитим контекстима. На пример:

I like you.
I am like you.

Обе реченице имају сличан скуп речи, међутим врста речи "like" се разликује, у првој реченици врста речи "like" је глагол, док у другој је предлог.

У пракси процес додељивања врсте речи одређеним речима назива се POS означавање (енгл. *POS tagging*).

Постоје различите технике за POS означавање [9]:

1. Методе засноване на лексичкој анализи – одређеној речи A се додељује врста речи V, ако се врста речи V најчешће појављивала уз A у тренинг скупу.
2. Методе засноване на правилима – постоје правописна правила која могу да одреде врсту речи, у енглеском језику речи које се завршавају са "ed" или "ing" су обично глаголи. Ово правило се користи упоредо са лексичком анализом како би се доделила врста речи оним речима које се не појављују у тренинг скупу.
3. Пробабилистичке методе – технике засноване на скривеним Марковљевим моделима и условним случајним пољима (енгл. *Conditional Random Fields*) који за низ врста речи додељују вероватноћу да се тај низ појави.
4. Методи засновани на добоком учењу – на основу означеног тренинг скупа формира се модел заснован на рекурентним неуронским мрежама.

3.1.3 Остале технике претпроцесирања текста

Поред техника заснованих на фреквенцији појављивања речи и плитке синтаксе постоје још неки параметри који се користе како би се пронашли скривени обрасци и везе међу текстовима, као што су број правописних грешака, број знакова интерпункције, просечна дужина реченице, број речи код којих су сва слова писана великим словом [11], [12].

3.1.4 Утицај техника претпроцесирања такста на препознавање лажних вести

Техника као што је плитка синтакса налази своје примене у препознавању ауторства текстова. Узима се у обзир претпоставка да аутори остављају свој потпис кроз стил писања. То значи да два различита аутора која пишу о истој теми користе сличан скуп речи, али њихов стил писања се разликује. Техника је предложена као другачији, нови пут у праћењу стила писања [13]. Са друге стране код два текста која говоре о различитим темама, и на нивоу скупа речи највећим делом се разликују, плитка синтакса и остале технике

претпроцесирања могу да препознају правила која на нивоу испод саме фреквенције речи повезују ова два документа. Једно од таквих правила у случају препознавања лажних вести је квалитет текста, где су аутори лажних текстова су често непрофесионални и код таквих текстова се очекује одређени број и образац у погледу правописних грешака, знакова интерпункције, просечне дужине реченице. Код аутора валидних текстова очекује се непристрасност, догађаји у тексту су описани чињеницама без субјективног мишљења, што ће се одразити у одређеном обрасцу броја именица, глагола, прилога и осталих атрибута плитке синтаксе.

4 Предложена метода за препозавање лажних вести

Као део рада представљена је имплементација решења, које коришћењем техника описаних у претходним поглављима, покушава да предвиди да ли је дати новински чланак истинит или лажан.

Са циљем да се имплементира решење, а имајући у виду да већина информација које човек добија долази са Интернета, имплементиран је и додатак (енгл. *extension*) за Chrome веб прегледач који се користи за класификовање новинских чланака на Интернету, детаљна имплементација додатка је описана у секцији „*Chrome* додатак за препознавање лажних вести“.

Да би имплементација која класификује текстове могла да закључи валидност чланка, било је потребно сакупити означене новинске чланке. У наставку су наведене коришћене технике и детаљи имплементације, као и примењени начини прикупљања података.

4.1 Коришћене технологије и библиотеке

Python

Jezik *Python* је интерпретабилан објектно орјентисани програмски језик високог нивоа *[14]*. Иако интуитиван и једноставан за коришћење, због тога што је интерпретабилан и језик високог нивоа, такође спада у групу језика који се споро извршавају. Његова популарност је довела до стварања заједнице која се бави развијањем наменских библиотека, између осталих и за потребе машинског учења.

Pycharm

Као радно окружење за језик *Python*, за овај рад изабран је *Pycharm*. *Pycharm* је развила компанија *JetBrains*, која се бави развијањем алата за програмирање. У даљим поглављима биће приказани делови кода написани у овом окружењу.

NumPy

NumPy је основни пакет за научно израчунавање u *Python* језику *[15]*. *NumPy* је *Python* библиотека која пружа објекте као што су вишедимензиони низови, матрице, и има функције за брзе операције над низовима, укључујући и операције основне линеарне алгебре, статистичке операције, фуријеове трансформације, и остало.

У основи *NumPy* библиотеке је објекат *ndarray*, који енкапсулира *n*-димензиони низ хомогених података, где већина операција бива извршена у компајлираном коду због побољшања перформанси. Постоји неколико разлика између *NumPy* низова и обичних низова:

- *NumPy* низови имају фиксну величину за разлику од *Python* низова који могу динамички да расту. Мењање величине *ndarray* објекта ће креирати нови низ и обрисати стари.
- Елементи *NumPy* низа морају бити истог типа, па ће у меморији бити исте величине.
- *NumPy* низови олакшавају напредне математичке и друге операције над великим бројем података. Обично се такве операције извршавају ефикасније и са мање написаног изворног кода помоћу уграђених *Python* операција.

Значај претпоставке о фиксној величини низа, из које произилази брзина извршавања операција и једноставност кода, најбоље се види на једноставном примеру множења два једнодимензиона низа.

```
c = []
for i in range(len(a)):
    c.append(a[i] * b[i])
```

Слика 4.1 Пример кода који множи два једнодимензиона низа у језику Python.

Додавање елемента на крај низа (*append()* у језику *Python*) има амортизовану служеност $O(1)$. Неефикасност овог приступа лежи у томе што, у тренутку када се попуни резервисана величина низа, низ мора интерно да се преалоцира у низ дупло веће димензије ($O(n)$).

```
for(i = 0; i < rows; i++) {
    c[i] = a[i] * b[i];
}
```

Слика 4.2 Пример кода који множи два једнодимензиона низа у језику C.

Иако је ефикасније, *C* не пружа широк спектар библиотека као *Python*. Такође, једноставност кода је мања са повећањем димензије низа, то се види на примеру множења два дводимензиона низа, код којих се множе елементи на одговарајућим позицијама:

```
for(i = 0; i < rows; i++) {
    for(j = 0; j < columns; j++) {
        c[i][j] = a[i][j] * b[i][j];
    }
}
```

Слика 4.3 Пример кода који множи два дводимензиона низа у језику C.

У наставку се види једноставност кода писаног коришћењем *NumPy* библиотеке.

```
c = a * b
```

Слика 4.4 Пример кода који множи два низа у језику Python коришћењем библиотеке *NumPy*.

Основне операције са *ndarray* објектима се имплицитно извршавају над елементима. Такође, операције над елементима извршавају се ефикасно, јер се у позадини користи прекомпајлирани *C* код.

Scikit-learn

Све већи број научних и математичких пакета заснованих на *Python* програмском језику користи *NumPy* низове. *Scikit-learn* је библиотека за подршку у писању модела машинског учења. Она користи следеће *Python* технологије *NumPy, SciPy* (такође користи: *NumPy*, ефикасни алгоритам за линеарну алгебру, баратање матрицама, статистичке функције), *Cython* (језик за комбиновање *C* и *Python* кода) [16]. *Scikit-learn* интегрише широк спектар алгоритама машинског учења за надгледано и ненадгледано учење. Нагласак је стављен на једноставност употребе, перформансе, документованост, доследност интерфејса и портабилност због једноставне лиценце. Ове технологије могу се једноставно преузети са Интернета.

Централни објекат је *estimator*, који имплементира *fit()* методу. Метода *fit()* прима као аргументе улазне податке (векторе атрибута) и опционо циљну променљиву у случају надгледаног учења. Објекат естиматор за надгледано учење као што је *SVM* класификатор имплементира метод *predict()*, који се користи за предвиђање циљне променљиве датог улазног податка (или датих улазних података).

```
from sklearn import svm

_svm = svm.SVC()
_svm.fit(train_x, train_y)
predicted = _svm.predict(test_x)
```

Слика 4.5 Пример кода који креира SVM естиматор у језику Python, позива метод fit(), затим позива метод predict() над тестним подацима и враћа предвиђање.

Неки *estimator* објекти имају другачију улогу. На пример код *PCA, estimator* објекат имплементира *transform()* методу, враћајући измењене улазне податке.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.6,svd_solver='full')
pca.fit(features)
transformed_features = pca.transform(features)
```

Слика 4.6 Пример кода који креира PCA естиматор објкеат у језику Python, позива метод fit() са улазним аргументом који представља атрибуте података, затим позива метод transform() над атрибутима и враћа измењене атрибуте применом технике PCA.

Објекат естиматор може такође имати *score()* методу, која враћа нумеричку оцену квалитета модела на основу логаритма веродостојности или функције грешке. Други важан метод је *cross_val_score()*, који се користи за технику унакрсне валидације.

```
from sklearn import svm
from sklearn.model_selection import cross_val_score

estimator = svm.SVC()
scores = cross_val_score(estimator, train_x, train_y, cv=5)
```

Слика 4.7 Пример кода који креира PCA модел у језику Python и евалуира квалитет коришћењем унакрсне валидације. Метода cross_val_score() прихвата модел, улазне податке, циљну променљиву и број који говори на колико подскупова поделити улазне податке.

Pandas

Pandas је библиотека напсана за *Python*, која олакшава баратање табеларним подацима, као што су табеларно записани подаци (енгл. *spreadsheet*) или базу података. Такође, нуди операције за манипулисање нумеричким табелама и временским серијама. Назив *Pandas* потиче од израза "*panel data*". Најчешће коришћен објекат је *DataFrame* који нуди апстракцију над 2-димензионалном означеном структуром података, са колонама потенцијално различитих типова. *DataFrame* прима различите улазе као што су: речник над 1-*D ndarray*, листе, речнике, 2-*D ndarray*, други *DataFrame* објекат.

Nltk

Netural Language Toolkit или *nltk* је *Python* библиотека која се користи за писање програма за рад са природним језицима [17]. Библиотека пружа више од 50 корпуса текстова, између осталих и *WordNet* [18]. Такође, пружа скуп метода за обраду текста, као што су токенизација, нормализација, стемовање, означавање, парсирање, семантичко резоновање, и друге.

Enchant

Enchant је библиотека која садржи широк спектар метода за правопис, укључујући и подршку за различите језике *[19]*. Провера правописа врши се помоћу објекта *Dict*, који представља речник. На основу речника методом *check()* проверава се да ли је реч исправно написана, док методом *suggest()* добија се листа предложених тачно написаних речи.

Newspaper

Newspaper је библиотека за језик *Python* која служи за издвајање и парсирање новинских чланака. Користи напредне алгоритме за обраду страница са Интернета како би извукла сав корисни текст са Интернет странице.

Неке од функционалности ове библиотеке су [20]:

- окружење за вишенитно скидање чланака,
- идентификација *URL*-а вести,
- издвајање текста из *HTML* странице,
- издвајање слика из *HTML* странице,
- издвајање кључних речи из текста,
- издвајање резимеа из текста,
- издвајање аутора из текста,
- издвајање *Google trends* појмова,
- иодршка за више од десет језика (енглески, кинески, немачки, арапски и други).

```
from newspaper import Article

article = Article(string_url)
article.download()
article.parse()
print(article.text)
```

Слика 4.8 Пример кода у језику Python који коришћењем објекта Article из библиотеке Newspaper учитава чланак са Интернет странице, издваја текст и исписује га на стандардни излаз функцијом print().

Pickle

Pickle модул имплементира протокол за серијализацију и десеријализацију у бинарни облик објеката у језику *Python* [21]. *Pickling* је процес којим се хијерархија *Python* објеката претвара у ток бајтова, а *unpickling* је инверзна операција, код које се ток бајтова конвертује назад у хијерархију објеката. Тако серијализовани објекти се могу чувати на диску за каснију поновну употребу. Конкретно у машинском учењу овај модул се користи за чување претходно формираних модела, како би се избегло њихово тренирање приликом поновног покретања програма.

4.2 Прикупљање података

У раду су коришћена два скупа података, где оба скупа података представљају податке на енглеском језику.

Први је настао са циљем да се обезбеди скуп који саджи две врсте текстова, истинитих текстова и лажних текстова. У датом тренутку, постојали су само засебни скупови са лажним текстовима. Тако је први скуп података настао обједињавањем лажног и истинитог скупа, сакупљених из различитих извора. За лажан скуп текстова у овом раду изабран је *"Getting Real about Fake News"* скуп лажних текстова [22]. Овај скуп лажних текстова садржи текст и метаподатке са 244 Интернет локације и има укупно 12999 текстова. Текстови су сакупљени током 2016. године обрадом страница које су у том тренутку биле означене као странице са лажним садржајем.

uuid	ord	author	published	title	text	language	crawled	site_url	cou	domain_	thread_title	spam	main_img_	replies	parti	likes	comm	shares	type
6a17	0	Barracu	2016-10-2	Muslir	Print	english	2016-10	100perc	US	25689	Muslims BUS	0	http://bb4	0	1	0	0	0	bias
2bdc	0	reasoni	2016-10-2	Re: W	Why	english	2016-10	100perc	US	25689	Re: Why Did	0	http://bb4	0	1	0	0	0	bias
c70e	0	Barracu	2016-10-3	BREAK	Red	english	2016-10	100perc	US	25689	BREAKING: V	0	http://bb4	0	1	0	0	0	bias
7cf7c	0	Fed Up	2016-11-0	PIN DI	Email K	english	2016-11	100perc	US	25689	PIN DROP SP	0.07	http://100	0	0	0	0	0	bias
0206	0	Fed Up	2016-11-0	FANTA	Email	english	2016-11	100perc	US	25689	FANTASTIC!	0.87	http://100	0	0	0	0	0	bias
8f30f	0	Barracu	2016-11-0	Hillary	Print	english	2016-11	100perc	US	25689	Hillary Goes	0	http://bb4	0	1	0	0	0	bias
d3cc	0	Fed Up	2016-11-0	BREAK	BREAKI	english	2016-11	100perc	US	25689	BREAKING! N	0.7	http://100	0	0	0	0	0	bias
b4bb	0	Fed Up	2016-11-0	WOW	BREAKI	english	2016-11	100perc	US	25689	WOW! WHIS	0.19	http://100	0	0	0	0	0	bias
f54d8	0	Fed Up	2016-11-0	EVIL H	Email	english	2016-11	100perc	US	25689	EVIL HILLARY	1	http://100	0	0	0	0	0	bias
fd2c	0	EdJenne	2016-11-1	NOT K	Studen	english	2016-11	100perc	US	25689	NOT KIDDIN	0	http://con	0	1	0	0	0	bias
c8d4	0	Fed Up	2016-11-1	BOOM	Email	english	2016-11	100perc	US	25689	BOOM! MAT	0	http://100	0	0	0	0	0	bias
f0b2	0	Fed Up	2016-11-1	BOOM	Copyrig	english	2016-11	100perc	US	25689	BOOM! This	0	http://100	0	0	0	0	0	bias
a9efb	0	EdJenne	2016-11-1	TRUM	Go to A	english	2016-11	100perc	US	25689	TRUMP SUPF	0	http://con	0	1	0	0	0	bias
13e3	0	Fed Up	2016-11-1	TOMI	Copyrig	english	2016-11	100perc	US	25689	TOMI LAHRE	0.27	http://100	0	0	0	0	0	bias

Слика 4.9 Пример података који долазе из скупа података "Getting Real about Fake News", скуп представља скуп искључиво лажних вести.

Поред тога што скуп лажних вести *"Getting Real about Fake News"* саджи и неке друге атрибуте, као што су аутор, коментари на Интернет објави, број свиђања, дељења и учесника на објави, у овом раду коришћен је само атрибут који представља текст чланка. Коришћен је само атрибут који представља текст чланка из разлога што модел машинског

учења не треба да рачуна на атрибуте који нису доступни у општем случају претраге Интернета.

У наставку су приказани неки од лажних чланака из скупа података *"Getting Real about Fake News"*:

- *"There is plenty of proof the machines are rigged whic also means trhe polls are rigged to match the machines so they must all know about it . Ask yourself how can a poll of a few thousand asked questions where they have no option possible tell you what hundreds of millions people who havent been polled think ?? Its impossible and has been the greatest con of the elite . And using past eletion results which were rigged to ppredict future results is also a scam .Soros Expects "Landslide Popular Vote Victory" For Trump But President Clinton Is A "Done Deal" Zero Hedge Done deal alright , they have been rinning this gig forever ."*

- *"Trump has an excuse now to audit any vote with these machines , trust the UN to be involved . He neds them to take a photo of their vote with phones , and be onto the polling booths for the bus loads of illegal voters going from booth to booth , he can bust this whole scam wide open if he is smart . Soros gave 10 million to Clinton , this must be ilegal surely ..."*

- *"He has got to go after him , he is the one causing al the trouble around the worl and is 100% proof that what Trump says about crooked and corruption is true . Conflict of interest ?? You bet it is funded by the state dept to run riot causing huge cost to USA with as Trump says ,wars they dont need amd shouldnt be involved in and making profit along the way . Have a look at his funding to all these non profits on the Gov site , its an outrage he is involved with the Voting machines . Soros Criminal Conviction Exposes "Human Rights" Scam Soros leverages "human rights" for personal gain - as does his global NGO empire. http://landdestroyer.blogspot.com.au/2012/03/surpise-soros-is-convicted-criminal.html"*

За скуп валидних чланака искоришћен је скуп *"BBC News Archive"* [23]. Овај скуп података садржи 2225 докумената сакупљених са званичне *BBC* Интернет странице. Документи представљају чланке из 5 различитих области, бизнис, забава, политика, спорт, технологија, објављених у преиоду између 2004 и 2005 године.

У наставку су приказани неки од истинитих чланака из скупа података *"BBC News Archive"*:

- *"Labour plans maternity pay rise Maternity pay for new mothers is to rise by £1,400 as part of new proposals announced by the Trade and Industry Secretary Patricia Hewitt. It would mean paid leave would be increased to nine months by 2007, Ms Hewitt told GMTV's Sunday programme. Other plans include letting maternity pay be given to fathers and extending rights to parents of older children. The Tories dismissed the maternity pay plan as "desperate", while the Liberal Democrats said it was misdirected. Ms Hewitt said: "We have already doubled the length of maternity pay, it was 13 weeks when we were elected, we have already taken it up to 26 weeks. "We are going to extend the pay to nine months by 2007 and the aim is to get it right up to the full 12 months by the end of the next Parliament." She said new mothers were already entitled to 12 months leave, but that many women could not take it as only six of those months were paid. "We have made a firm commitment. We will definitely extend the maternity pay, from the six months where it now is to nine months, that's the extra £1,400." She said ministers would consult on other proposals that could see*

fathers being allowed to take some of their partner's maternity pay or leave period, or extending the rights of flexible working to carers or parents of older children. The Shadow Secretary of State for the Family, Theresa May, said: "These plans were announced by Gordon Brown in his pre-budget review in December and Tony Blair is now recycling it in his desperate bid to win back women voters." She said the Conservatives would announce their proposals closer to the General Election. Liberal Democrat spokeswoman for women Sandra Gidley said: "While mothers would welcome any extra maternity pay the Liberal Democrats feel this money is being misdirected." She said her party would boost maternity pay in the first six months to allow more women to stay at home in that time. Ms Hewitt also stressed the plans would be paid for by taxpayers, not employers. But David Frost, director general of the British Chambers of Commerce, warned that many small firms could be "crippled" by the move. "While the majority of any salary costs may be covered by the government's statutory pay, recruitment costs, advertising costs, retraining costs and the strain on the company will not be," he said. Further details of the government's plans will be outlined on Monday. New mothers are currently entitled to 90% of average earnings for the first six weeks after giving birth, followed by £102.80 a week until the baby is six months old."

- *"Ad sales boost Time Warner profit Quarterly profits at US media giant TimeWarner jumped 76% to $1.13bn (£600m) for the three months to December, from $639m year-earlier. The firm, which is now one of the biggest investors in Google, benefited from sales of high-speed internet connections and higher advert sales. TimeWarner said fourth quarter sales rose 2% to $11.1bn from $10.9bn. Its profits were buoyed by one-off gains which offset a profit dip at Warner Bros, and less users for AOL. Time Warner said on Friday that it now owns 8% of search-engine Google. But its own internet business, AOL, had has mixed fortunes. It lost 464,000 subscribers in the fourth quarter profits were lower than in the preceding three quarters. However, the company said AOL's underlying profit before exceptional items rose 8% on the back of stronger internet advertising revenues. It hopes to increase subscribers by offering the online service free to TimeWarner internet customers and will try to sign up AOL's existing customers for high-speed broadband. TimeWarner also has to restate 2000 and 2003 results following a probe by the US Securities Exchange Commission (SEC), which is close to concluding. Time Warner's fourth quarter profits were slightly better than analysts' expectations. But its film division saw profits slump 27% to $284m, helped by box-office flops Alexander and Catwoman, a sharp contrast to year-earlier, when the third and final film in the Lord of the Rings trilogy boosted results. For the full-year, TimeWarner posted a profit of $3.36bn, up 27% from its 2003 performance, while revenues grew 6.4% to $42.09bn. "Our financial performance was strong, meeting or exceeding all of our full-year objectives and greatly enhancing our flexibility," chairman and chief executive Richard Parsons said. For 2005, TimeWarner is projecting operating earnings growth of around 5%, and also expects higher revenue and wider profit margins. TimeWarner is to restate its accounts as part of efforts to resolve an inquiry into AOL by US market regulators. It has already offered to pay $300m to settle charges, in a deal that is under review by the SEC. The company said it was unable to estimate the amount it needed to set aside for legal reserves, which it previously set at $500m. It intends to adjust the way it accounts for a deal with German music publisher Bertelsmann's purchase of a stake in AOL Europe, which it had reported as advertising revenue. It will now book the sale of its stake in AOL Europe as a loss on the value of that stake."*

Други скуп података *"News Data Set – Fake OR Real"* објављен је крајем Јула 2020. године [24]. Овај скуп података је згодан јер садржи оба типа текста, лажне и истините, и намењен је баш за тренирање модела који се користе за класификовање лажних вести. Скуп података садржи 6060 текстова код којих је однос истинитих и лажних текстова 50%-50%.

	title	text	type
8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fellow at the Freedom Center, is a New York writer focusing on radical Islam.	FAKE
10294	Watch The Exact Moment Paul Ryan Committed Politica	Google Pinterest Digg Linkedin Reddit Stumbleupon Print Delicious Pocket Tumblr	FAKE
3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Monday that he will stop in Paris later this week, amid criticism that no top American	REAL
10142	Bernie supporters on Twitter erupt in anger against the D	" Kaydee King (@KaydeeKing) November 9, 2016 The lesson from tonight's Dem losses: Time for Democrats to start listening to	FAKE
875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners Hillary Clinton and Donald Trump are leading in the polls.	REAL
6903	Tehran, USA		FAKE
7341	Girl Horrified At What She Watches Boyfriend Do After H	Share This Baylee Luciani (left), Screenshot of what Baylee caught on FaceTime (right)	FAKE
95	"Britain's Schindler" Dies at 106	A Czech stockbroker who saved more than 650 Jewish children from Nazi Germany has died at the age of 106. Dubbed â€œBritain	REAL
4869	Fact check: Trump and Clinton at the 'commander-in-chi	Hillary Clinton and Donald Trump made some inaccurate claims during an NBC â€œcommander-in-chiefâ€ forum on military and	REAL
2909	Iran reportedly makes new push for uranium concession	Iranian negotiators reportedly have made a last-ditch push for more concessions from the U.S. and five other world powers as	REAL
1357	With all three Clintons in Iowa, a glimpse at the fire that	CEDAR RAPIDS, Iowa had one of the most wonderful rallies of my entire career right here in 1992, Bill Clinton said by way of	REAL
988	Donald Trump's Shockingly Weak Delegate Game Someh	Donald Trump's organizational problems have gone from bad to worse to flat-out embarrassing. Hereâ€™s Politico with the play-	REAL
7041	Strong Solar Storm, Tech Risks Today | S0 News Oct.26.2	Click Here To Learn More About Alexandra's Personalized Essences Psychic Protection Click Here for More Information on	FAKE
7623	10 Ways America Is Preparing for World War 3	October 31, 2016 at 4:52 am	FAKE
1571	Trump takes on Cruz, but lightly	Killing Obama administration rules, dismantling Obamacare and pushing through tax reform are on the early to-do list.	REAL
4739	How women lead differently	As more women move into high offices,Â they often bring a style and approach that is distinct from men. But do they make	REAL
7737	Shocking! Michele Obama & Hillary Caught Glamorizing D	Shocking! Michele Obama & Hillary Caught Glamorizing Date Rape Promoters First lady claims moral high ground while	FAKE
8716	Hillary Clinton in HUGE Trouble After America Noticed SI	0	FAKE
3304	What's in that Iran bill that Obama doesn't like?	Washington (CNN) For months, the White House and Congress have wrangled over a bill that would give lawmakers a greater say	REAL

Слика 4.10 Пример података који долазе из скупа података "Getting Real about Fake News".

У наставку су приказани неки од истинитих и лажних чланака из скупа података *"News Data set – Fake OR Real"*:

- *"Minister Laurent Fabius and President Francois Hollande, then return to Washington. The visit by Kerry, who has family and childhood ties to the country and speaks fluent French, could address some of the criticism that the United States snubbed France in its darkest hour in many years. The French press on Monday was filled with questions about why neither President Obama nor Kerry attended Sunday's march, as about 40 leaders of other nations did. Obama was said to have stayed away because his own security needs can be taxing on a country, and Kerry had prior commitments. Among roughly 40 leaders who did attend was Israeli Prime Minister Benjamin Netanyahu, no stranger to intense security, who marched beside Hollande through the city streets. The highest ranking U.S. officials attending the march were Jane Hartley, the ambassador to France, and Victoria Nuland, the assistant secretary of state for European affairs. Attorney General Eric H. Holder Jr. was in Paris for meetings with law enforcement officials but did not participate in the march. Kerry spent Sunday at a business summit hosted by India's prime minister, Narendra Modi. The United States is eager for India to relax stringent laws that function as barriers to foreign investment and hopes Modi's government will act to open the huge Indian market for more American businesses. In a news conference, Kerry brushed aside criticism that the United States had not sent a more senior official to Paris as quibbling a little bit." He noted that many staffers of the American Embassy in Paris attended the march, including the ambassador. He said he had wanted to be present at the march himself but could not because of his prior commitments in India. But that is why I am going there on the way home, to make it crystal clear how passionately we feel about the events that have taken place there," he said. And I don't think the people of France have any doubts about America's understanding of what happened, of our personal sense of loss and our deep commitment to the people of France in this moment of trauma."*

- *"Google Pinterest Digg Linkedin Reddit Stumbleupon Print Delicious Pocket Tumblr. There are two fundamental truths in this world: Paul Ryan desperately wants to be president. And Paul Ryan will never be president. Today proved it. In a particularly staggering example of political cowardice, Paul Ryan re-re-re-reversed course and announced that he was back on the Trump Train after all. This was an aboutface from where he was a few weeks ago. He had previously declared he would not be supporting or defending Trump after a tape was made public in which Trump bragged about assaulting women. Suddenly, Ryan was appearing at a pro-Trump rally and boldly declaring that he already sent in his vote to make him President of the United States. It was a surreal moment. The*

figurehead of the Republican Party dosed himself in gasoline, got up on a stage on a chilly afternoon in Wisconsin, and lit a match. . @SpeakerRyan says he voted for @realDonaldTrump : "Republicans, it is time to come home" https://t.co/VyTT49YvoE pic.twitter.com/wCvSCg4a5I ” ABC News Politics (@ABCPolitics) November 5, 2016 The Democratic Party couldn't have asked for a better moment of film. Ryan's chances of ever becoming president went down to zero in an instant. In the wreckage Trump is to leave behind in his wake, those who cravenly backed his campaign will not recover. If Ryan's career manages to limp all the way to 2020, then the DNC will have this tape locked and loaded to be used in every ad until Election Day. The ringing endorsement of the man he clearly hates on a personal level speaks volumes about his own spinelessness. Ryan has postured himself as a "principled" conservative, and one uncomfortable with Trump's unapologetic bigotry and sexism. However, when push came to shove, Paul Ryan “ like many of his colleagues “ turned into a sniveling appeaser. After all his lofty tak about conviction, his principles were a house of cards and collapsed with the slightest breeze. What's especially bizarre is how close Ryan came to making it through unscathed. For months the Speaker of the House refused to comment on Trump at all. His strategy seemed to be to keep his head down, pretend Trump didn't exist, and hope that nobody remembered what happened in 2016. Now, just days away from the election, he screwed it all up. If 2016's very ugly election has done any good it's by exposing the utter cowardice of the Republicans who once feigned moral courage. A reality television star spit on them, hijacked their party, insulted their wives, and got every last one of them to kneel before him. What a turn of events. Featured image via Twitter”.

4.3 Припрема података

Процес припреме података своди се на примену, у претходним поглављима описаних, техника нормализације и претпроцесирања у језику *Python*. Излаз процеса припреме података представља улаз за фазу обуке (тренирања модела) или улаз за фазу предвиђања (примене модела). Модел очекује да се на податке примени исти низ техника припреме података у фази обуке и у фази предвиђања.

У случају фазе обуке излаз претпроцесирања је матрица података, димензије $M \times N$, где је N број текстова, а сваки текст је претворен у низ атрибута дужине M.

Код фазе предвиђања, која се врши над једним текстом, излаз је низ атрибута дужине M који описују тај текст у простору атрибута.

Сваки корак припреме података представља по једну технику претпроцесирања. Низ атрибута дужине M је надаовезани излаз сваке од техника претпроцесирања.

У наставку су приказани делови кода који имплементирају коришћене технике претпроцесирања.

Прва примењена техника је техника нормализације текста, која улазни текст претвара у низ речи, токена. Речи претвара у речи писане малим словима, како би се избегло разликовање између речи писаним великим и малим словима. Из скупа речи избацује речи које не носе значење у погледу техника за обраду природних језика коришћењем скупа речи из библиотеке nltk.

```
def textToWords(self, inputString):
    letters_only = re.sub("[^a-zA-Z]", " ", str(inputString))
    words = letters_only.lower().split()
    stops = set(nltk.corpus.stopwords.words("english"))
    meaningful_words = [w for w in words if not w in stops]
    return (" ".join(meaningful_words))
```

Слика 4.10 Део кода, односно функција textToWords(), која врши токенизацију и нормализацију текста.

Овако нормализован текст је даље улаз у технику засновану на фреквенцији појављивања речи. За потребе модуларности имплементирана је класа BagOfWords и њена имплементација дата у коду 4.11. Класа BagOfWords користи објекат *TfidfVectorizer* из библиотеке scikit-learn (односно sklearn). Објекат *TfidfVectorizer* је кључан за рачунање фреквенција појављивања речи докумената, параметар *ngram_range* говори од колико речи токен треба да се садржи, у овом случају прате се појављивања униграма и биграма вредношћу (1,2). Параметри *max_df* и *min_df* користе се као филтер који игнорише речи које се појављују више од *max_df* пута или мање од *min_df* пута. Параметар *max_features* је број речи чију фреквенцију желимо да пратимо у текстовима, он је такође димензија излазног низа из корака BagOfWords. У фази тренинга модел који рачуна фреквенције појављивања речи се чува коришћењем библиотеке *pickle*, како би се исти модел користио касније у фази предвиђања.

```
def transformData(self, data, trainModel=True, transformerOutputPath=None):
    num_texts = data.size
    clean_train_texts = []

    for i in range(0, num_texts):
        clean_train_texts.append(self.textToWords(data[i]))

    if (trainModel == True):
        vectorizer = TfidfVectorizer(analyzer='word',
                                     sublinear_tf=True,
                                     strip_accents='unicode',
                                     ngram_range=Consts.TFIDF_NGRAM_RANGE,
                                     max_df=Consts.TFIDF_MAX_DF,
                                     min_df=Consts.TFIDF_MIN_DF,
                                     max_features=Consts.TFIDF_MAX_FEATURES)
        tfidf = vectorizer.fit(clean_train_texts)

        if (transformerOutputPath is not None):
            pickle.dump(tfidf, open(transformerOutputPath, "wb"))

        data_features = vectorizer.transform(clean_train_texts).toarray()
        self.vectorizer = vectorizer
    else:
        data_features = self.vectorizer.transform(clean_train_texts).toarray()

    self.data_features = data_features
    self.feature_names = self.vectorizer.get_feature_names()
```

Код 4.11 Имплементација класе BagOfWords.

Наредни пример је приказ класе *Punctations* која броји знакове интерпункције у датом тексту. Како се за дуже текстове очекује да имају већи број знакова интерпункције,

број се дели са дужином текста, такође се примењују и хипер параметри како би се побољшале нумеричке операције у наредним фазама.

```
class Punctations:
    def transformData(self, data):
        num_texts = data.size
        punctuation_freq = []
        count = lambda l1, l2: sum([1 for x in l1 if x in l2])
        for i in range(0, num_texts):
            count_punctation = count(data[i],set(string.punctuation))
            text_punctuation_freq = count_punctation / len(data[i])
            text_punctuation_freq = Consts.PC_MUL*text_punctuation_freq**Consts.PC_POW
            punctuation_freq.append(text_punctuation_freq)

        self.punctuation_freq = pd.DataFrame(np.array(punctuation_freq),
                                             columns=['punctuation_frequency'])
```

Код 4.12 Приказ имплементације класе Punctations.

Наредни пример је приказ класе *SpellCheck* која рачуна просечан број правописних грешака по тексту. Класа користи речник из библиотеке *enchant* како би одредила да ли је реч правилно написана, број правописних грешака се касније дели са укупном дужином текста како би се избегло да дугачки текстови имају већи број правописних грешака од краћих текстова. Математичке операцију су примењене како би се побољшала нумеричка својства.

```
class SpellCheck:
    def transformData(self, data):
        num_texts = data.size
        misspelled_freq = []
        d = enchant.Dict("en_US")
        for i in range(0, num_texts):
            letters_only = re.sub("[^a-zA-Z]", " ", str(data[i]))
            words = letters_only.lower().split()
            ms_count = sum([1 for x in words if d.check(x) is False])
            freq = ms_count / len(words)
            if (ms_count != 0):
                freq = ms_count / len(words)
                freq = Consts.SC_MUL * freq ** Consts.SC_POW
            else:
                freq = 0
            misspelled_freq.append(freq)
        self.misspelled_freq = pd.DataFrame(np.array(misspelled_freq),
                                            columns=['misspelled_freq'])
```

Код 4.13 Приказ имплементације класе SpellCheck.

Наредни пример је приказ функције која рачуна просечну дужину речи и реченица.

```
class AverageLength:
    def transformData(self, data):
        num_texts = data.size
        avg_sent_len = []
        avg_word_len = []
        for i in range(0, num_texts):
            filtered = ''.join(filter(lambda x: x not in Consts.PUNCT_MARKS,
                                      str(data[i])))
            words = [word for word in filtered.split() if word]
            word_len = sum(map(len, words))
            avg_word_len.append((word_len / len(words))**Consts.AL_POW)

            sents = str(data[i]).split('.')
            filtered_sents = []
            for sent in sents:
                sent = ''.join(filter(lambda x: x not in Consts.PUNCT_MARKS,
                                      sent))
                filtered_sents.append(sent)
            sent_len = sum(map(len, filtered_sents))
            avg_sent_len.append((sent_len / len(filtered_sents))**Consts.AL_POW)

        self.average_word_len = pd.DataFrame(np.array(avg_word_len),
                                             columns=['average_word_len'])
        self.average_sentence_len = pd.DataFrame(np.array(avg_sent_len),
                                                 columns=['average_sentence_len'])
```

Код 4.13 Приказ имплементације класе AvarageLength.

Наредни пример је приказ класе *CapitalizedWords* које рачуна број речи писаних великим словима у тексту. Посебна пажња се даје текстовима који немају ни једну реч писану великим словима, такви текстови добијају негативну фреквенцију да би се избегло баратање нула вредностима.

```
class CapitalizedWords:
    def transformData(self, data):
        cap_freq = []
        num_texts = data.size
        for i in range(0, num_texts):
            filtered = ''.join(filter(lambda x: x not in Consts.PUNCT_MARKS,
                                      str(data[i])))
            filtered = filtered.split()
            cw_len = len([word for word in filtered
                          if word.isupper() and len(word)>Consts.CW_MIN_WORD_LEN])
            freq = cw_len / len(filtered)
            if(freq == 0):
                freq = Consts.CW_NEG_MUL * len(filtered)
            else:
                freq = Consts.CW_MUL * freq ** Consts.CW_POW
            cap_freq.append(freq)

        self.capitalized_words_freq = pd.DataFrame(np.array(cap_freq),
                                                   columns=['capitalized_words_freq'])
```

Код 4.14 Приказ имплементације класе CapitalizedWords.

Наредни пример је приказ класе *ShallowSyntax* која примењује технику плитке синтаксе. Коришћењем скупа *upenn_tagset* из библиотеке *nltk*, добија се речник понуђених врста речи, затим функцијом *pos_tag()* одређују се врсте речи за речи из датог текста. Функцијом *FreqDist()* се рачунају фреквенције датих врста речи у тексту. Како врста речи у скупу има 36, да би се избегло праћење сувишних врста речи, примењена је техника анализе главних компоненти, која има улазни параметар *n_components* број главних компоненти које се задржавају. У фази тренинга модел који врши анализу главних компоненти се чува коришћењем библиотеке *pickle*, како би се исти модел касније користио у фази предвиђања.

```
class ShallowSyntax:
    def transformData(self, data, trainModel=True, pcaOutputPath = None):
        num_texts = data.size
        columns = []
        tagdict = nltk.load('help/tagsets/upenn_tagset.pickle')
        for key in tagdict.keys():
            columns.append(str(key))

        df = pd.DataFrame(columns=columns)

        for i in range(0, num_texts):
            new_row = pd.DataFrame(index = [i],columns = columns)
            for key in tagdict.keys():
                new_row.set_value(i, key, 0)

            text = nltk.tokenize.word_tokenize((str(data[i]).replace(r'"(.*?)"', '')))
            tagged_text = nltk.pos_tag(text)
            tag_fd = nltk.FreqDist(tag for (word, tag) in tagged_text)
            for (key,value) in tag_fd.items():
                if(key in tagdict.keys()):
                    new_row.set_value(i,
                                      key,
                                      Consts.SS_MUL*(value/tag_fd.N())**Consts.SS_POW)
            df = pd.concat([df,new_row])

        if(trainModel == True):
            pca = PCA(n_components=Consts.PCA_COMPONENTS,svd_solver='full')
            pca.fit(df.values)
            if(pcaOutputPath is not None):
                pickle.dump(pca, open(pcaOutputPath, "wb"))
            transformed_features = pca.transform(df.values)
            self.pca = pca
        else:
            transformed_features = self.pca.transform(df.values)

        self.shallow_syntax_features = pd.DataFrame(transformed_features)
```

Код 4.15 Приказ имплементације класе ShallowSyntax.

Имплементација такође подржава динамичко подешавање програма помоћу улазних датотеки које садрже команде, тако да се без мењања изворног кода могу генерисати модели који користе и комбинују различите врсте техника претпроцесирања. Циљ је изабрати онај скуп техника који даје најбоље резултате. Преглед упоређивања различитих модела дат је у

поглављу „Експериментални резултати“. Једна улазна датотека може да садржи команде за више модела, па је излаз из једног тренинга скуп модела са одговарајућим резултатима.

У наставку је дат приказ једне датотеке команди. Сваки ред у датотеци генерише по један модел са одређеним техникама претпроцесирања података. Редом у коду 4.16, први ред генерише модел који користи BagOfWords метод, други плитку синтаксу, трећи генерише модел који користи фреквенцију правописних грешака, и тако даље. Такође, је могуће комбиновати више техника, тако осми ред генерише модел који користи технике BagOfWords; фреквенцију знакова интерпункције и просечну дужину речи и реченице.

```
bow
shallow_syntax
misspelled_words
punctation
sentence
capitalized_words
bow shallow_syntax
bow punctation sentence
shallow_syntax misspelled_words punctation sentence
bow shallow_syntax misspelled_words punctation sentence capitalized_words
```

Код 4.16 Приказ једне датотеке команди.

4.4 Метод подржавајућих вектора и логистичка регресија

У раду су искоришћена два приступа: метод подржавајућих вектора, *Support Vector Machine* и логистичка регресија. Ова два алгоритма су одабрана како би био упоређен један једноставан приступ као што је логистичка регресија и један сложенији као што је метод подржавајућих вектора.

Оба алгоритма су део библиотеке *Scikit-learn*. У приложеној имплементацији, током фазе тренинга улаз у алгоритам је матрица атрибута настала претпроцесирањем скупа текстова, док је излаз прилагођен модел и оцена добијена унакрсном валидацијом. Модел се чува помоћу библиотеке *pickle* и касније користи приликом предвиђања.

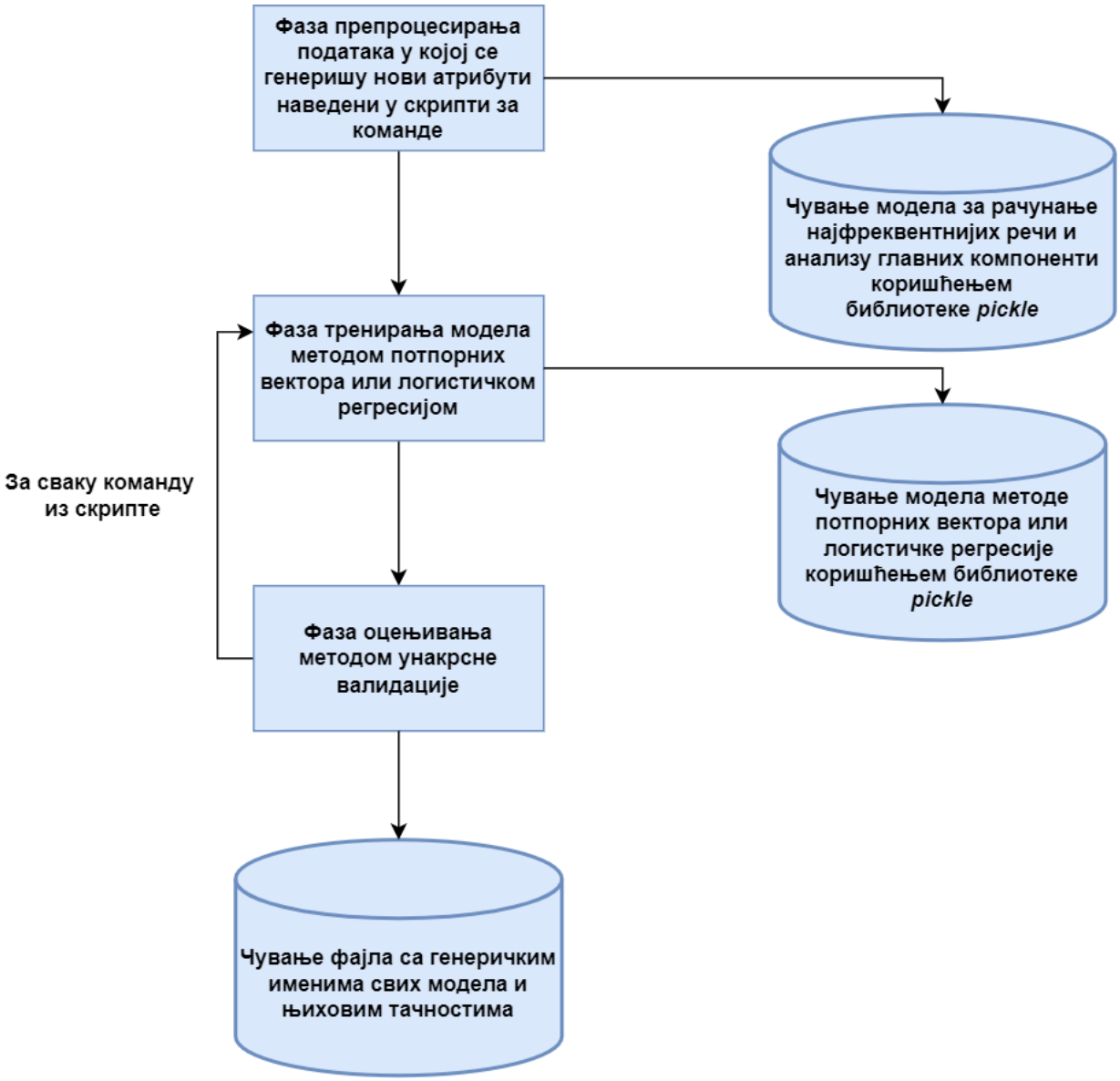
У фази предвиђања модел се учитава помоћу библиотеке *pickle*. Затим коришћењем метода *predict()* над низом атрибута, добијеним препоцесирањем појединачног текста, добија се предвиђање за тај текст.

Имплементација метода *ScoreModel()* из креиране класе *FeatureWrapper* приказан је у коду 4.17. Метода се користи за тренирање модела. Као улаз прима аргумент *data_train* и *data_test*. Тренирање модела врши се унакрсном валидацијом коришћењем *data_train* скупа, док се тачност модела рачуна коришћењем *data_test* скупа. Остали параметри су *commandsFile* путања са које ће се учитати датотека команди, *useSVM* који говори да ли се користи метод подржавајућих вектора или логистичка регресија, параметар *cv* говори да ли се користи унакрсна валидација, параметар *outFile* који је путања до текстуалне датотеке где ће бити сачувана генеричка имена модела и њихова тачност, *saveModelPath* путања до датотеке где ће *pickle* библиотека сачувати моделе, *pcaOutputPath* путања где ће *pickle*

библиотека сачувати модел за анализу главних компоненти, *transformerOutputPath* путања где ће *pickle* библиотека сачувати модел за рачунање фреквенција појављивања речи.

```
class FeatureWrapper:
    def ScoreModel(self, data_train, data_test, commandsFile, useSVM=True,
                   outFile=None, saveModelPath=None, pcaOutputPath=None,
                   transformerOutputPath=None):
        X_train = data_train['text']
        y_train = data_train['type']
        X_test = data_test['text']
        y_test = data_test['type']

        # Part of code for generating attributes
        # .
        # .
        # .
        # For each command in command file execute below code

        if (useSVM == True):
            clf = svm.SVC()
            clf.fit(new_features_train, y_train)
        else:
            clf = LogisticRegression(random_state=Consts.LR_RANDOM_STATE,
                                     max_iter=Consts.LR_MAX_ITER
                                    ).fit(new_features_train, y_train)

        if (saveModelPath is not None):
            dump(clf, saveModelPath + '/' + title_string + '.joblib')

        scores = cross_val_score(clf, new_features_train, y_train, cv=Consts.CV_FOLDS)
        acc_score = clf.score(new_features_test, y_test)

        score_string = "cross_val: {d:.3f}".format(d=scores.mean()) \
                       + " / {d:.3f}".format(d=scores.std()) \
                       + "\n" \
                       + "acc_test {d:.3f}:".format(d=acc_score) \
                       + "\n"
        out_string += "-----------------\n"
        out_string += title_string + "\n"
        out_string += score_string + "\n"
        out_string += "-----------------\n"

        if outFile is not None:
            with open(outFile, 'w') as f:
                f.write(out_string)
```

Код 4.17 Приказ функције ScoreModel из креиране класе FeatureWrapper која се користи за тренирање модела.



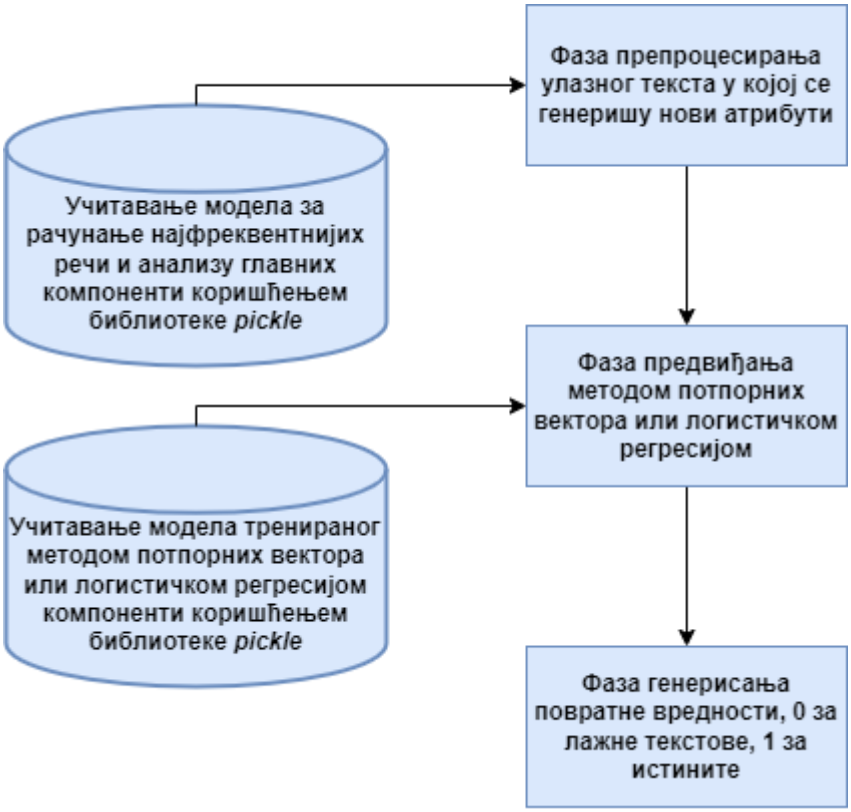
Слика 4.19 Приказ делова фазе учења.

Имплементација метода *ScoreText()* из креиране класе *FeatureWrapper* која се користи за предвиђање истинитости једног текста дата је у коду 4.20. Као улаз прима аргумент *input_text*, текст над којим се врши предвиђање, аргумент *commands* који се користи за одабир претходно сачуваног модела који ће се сада користити за предвиђање, аргумент *saveModelPath* путања до директоријума где се налазе сачувани модели, аргумент *transformModelPath* представља путању до модела који се користи за рачунање најфреквентнијих речи, аргумент *pcaModelPath* представља путању до модела кои се користи за анализу главних компоненти. Повратна вредност метода *ScoreText()* је

променљива *predicted* која представља предвиђање учитаног подела, узима вредности 0 за лажне текстове и 1 за истините.

```
def ScoreText(self, input_text, commands,
              modelPath, transformModelPath, pcaModelPath):

    df_data_text = pd.DataFrame([input_text], columns=['text'])
    new_features = None
    commands_split = commands.split()
    for command in commands_split:
        if (command == Consts.CM_BOW):
            bow = BagOfWords()
            bow.transformText(df_data_text['text'], transformModelPath)
            new_features = pd.concat([new_features,
                                      pd.DataFrame(bow.data_features)],
                                     axis=1)
        elif (command == Consts.CM_SS):
            ss = ShallowSyntax()
            ss.transformText(df_data_text['text'], pcaModelPath)
            new_features = pd.concat([new_features,
                                      pd.DataFrame(ss.shallow_syntax_features)],
                                     axis=1)
        elif (command == Consts.CM_MW):
            ms = SpellCheck()
            ms.transformData(df_data_text['text'])
            new_features = pd.concat([new_features,
                                      pd.DataFrame(ms.misspelled_freq)],
                                     axis=1)
        elif (command == Consts.CM_PC):
            punc = Punctations()
            punc.transformData(df_data_text['text'])
            new_features = pd.concat([new_features,
                                      pd.DataFrame(punc.punctuation_freq)],
                                     axis=1)
        elif (command == Consts.CM_SL):
            sent = AverageLength()
            sent.transformData(df_data_text['text'])
            new_features = pd.concat([new_features,
                                      pd.DataFrame(sent.average_word_len),
                                      pd.DataFrame(sent.average_sentence_len)],
                                     axis=1)
        elif (command == Consts.CM_CW):
            cw = CapitalizedWords()
            cw.transformData(df_data_text['text'])
            new_features = pd.concat([new_features,
                                      pd.DataFrame(cw.capitalized_words_freq)],
                                     axis=1)
        else:
            print("Unkown command: ", command)

    clf = load(modelPath + '/' + commands + '.joblib')
    predicted = clf.predict(new_features)
    return predicted
```

Код 4.20 Приказ функције ScoreText из креиране класе FeatureWrapper која се користи за предвиђање истинитости једног текста.



Слика 4.21 Приказ делова фазе предвиђања.

5 Експериментални резултати

Експериментални резултати обухватају поређење између два алгоритма, метод подржавајућих вектора и логистичке регресије, над два различита скупа података описаних у једном од претходних поглавља. Први скуп података настао коришћењем лажних вести из скупа података *"Getting Real about Fake News"* и коришћењем истинитих вести из скупа података *"BBC News Archive"*. Овако обједињен скуп података, ради краћег записа, у раду назван је *"real_fake_dataset"*. Други скуп података је *"News Data Set – Fake OR Real"* који је у раду, због краћег записа, назван *"news_dataset"*.

Експериментални резултати такође обухватају резултате модела који користе различите комбинације атрибута. Због великог броја различитих комбинација атрибута, приказани су само они који дају најбоље резултате. У наставку су наведене скраћенице коришћених техника претпроцесирања.

- *BagOfWords – bow*.
- Плитка синтакса – *Shallow Syntax – ss*.
- Фреквенција правописних грешака – *Misspelled Words – mw*.

- Фреквенција знакова интерпункције – *Punctation – pc.*
- Фреквенција речи писаних великим словом – *Capitalized Words – cw.*
- Просечна дужина речи и реченица – *Avarage Sentence Length – sl.*

Такође, приказани су и резултати модела добијених различитим подешавањима хиперпараметара претпроцесирања *BagOfWords* технике, као што су 1-грами, 2-грами, њихово комбиновање као и различити број најфреквентинијих n-грама, 1000, 1500 и 3000 најфреквентнијих.

У наставку су приказане табеле експерименталних резултата, где су резултати представљени као просечна тачност над тренинг подацима са стандардном девијацијом унакрсне валидације са 5 подскупова, и тачност над тест подацима (однос тренинг и тест података је 0.80 према 0.20).

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	**0.989**	**0.002**	**0.991**
ss	0.55	0.006	0.563
mw	0.504	0.001	0.485
pc	0.765	0.018	0.737
sl	0.525	0.012	0.485
cw	0.696	0.023	0.694
bow + ss	0.988	0.002	0.991
bow + ss + mw	0.988	0.003	0.988
bow + ss + pc	0.985	0.004	0.989
bow + ss + cw	0.988	0.003	0.989

Слика 5.1 "real_fake_dataset" скуп података, логистичка регресија, где BoW користи 3000 најфреквентнијих 1-грама и 2-грама.

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	**0.992**	**0.001**	**0.992**
ss	0.56	0.006	0.561
mw	0.62	0.010	0.604
pc	0.826	0.003	0.789
sl	0.691	0.007	0.689
cw	0.525	0.006	0.515
bow + ss	0.980	0.004	0.98
bow + ss + mw	0.625	0.01	0.609

Слика 5.2 "real_fake_dataset" скуп података, метод подржавајућих вектора, где BoW користи 3000 најфреквентнијих 1-грама и 2-грама.

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	0.908	0.006	0.902
ss	0.607	0.011	0.629
mw	0.563	0.029	0.575
pc	0.498	0.003	0.516
sl	0.55	0.021	0.591
cw	0.585	0.033	0.547
bow + ss	0.911	0.007	0.91
bow + ss + mw	0.911	0.007	0.91
bow + ss + pc	**0.91**	**0.007**	**0.913**
bow + ss + cw	0.911	0.007	0.911

Слика 5.3 "news_dataset" скуп података, логистичка регресија, где BoW користи 1000 најфреквентнијих 1-грама и 2-грама.

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	0.917	0.008	0.924
ss	0.607	0.011	0.629
mw	0.563	0.029	0.575
pc	0.498	0.003	0.516
sl	0.55	0.021	0.591
cw	0.585	0.033	0.547
bow + ss	0.925	0.008	0.936
bow + ss + mw	**0.926**	**0.009**	**0.936**
bow + ss + pc	0.918	0.011	0.932
bow + ss + cw	0.924	0.009	0.937

Слика 5.4 "news_dataset" скуп података, логистичка регресија, где BoW користи 3000 најфреквентнијих 1-грама и 2-грама.

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	**0.883**	**0.012**	**0.88**
ss	0.608	0.007	0.625
mw	0.563	0.015	0.579
pc	0.605	0.014	0.622
sl	0.529	0.018	0.537
cw	0.523	0.009	0.529
bow + ss	0.879	0.009	0.88
bow + ss + mw	0.567	0.014	0.592
bow + ss + pc	0.605	0.014	0.625
bow + ss + cw	0.537	0.021	0.544

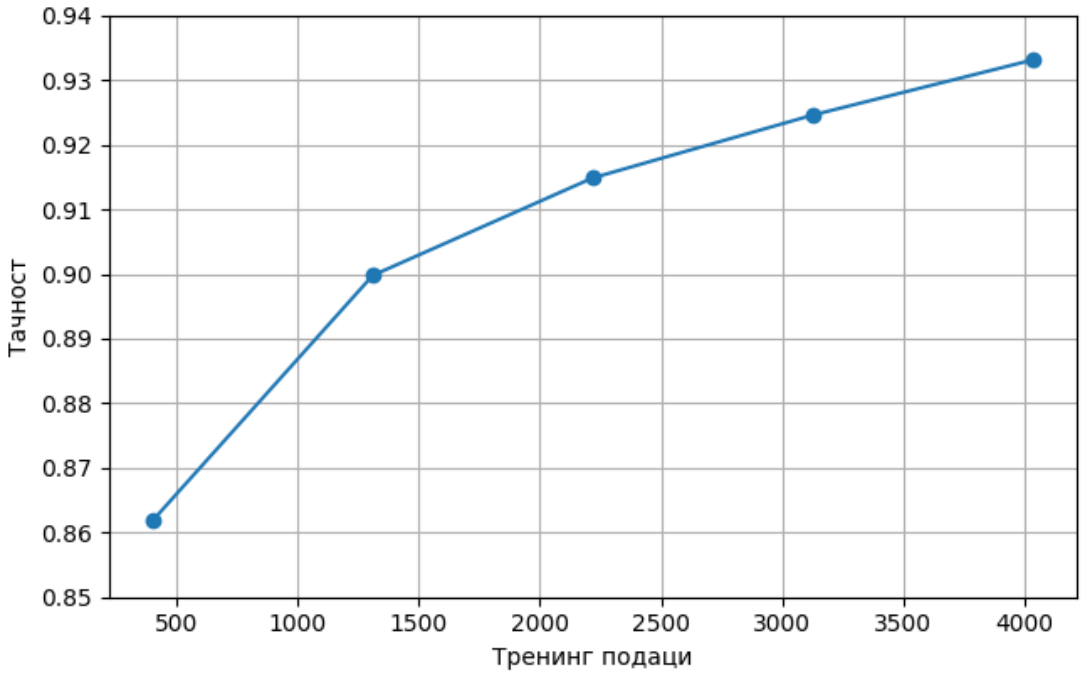
Слика 5.5 "news_dataset" скуп података, метод подржавајућих вектора, где BoW користи 1000 најфреквентнијих 1-грама и 2-грама.

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	**0.933**	**0.005**	**0.934**
ss	0.608	0.007	0.625
mw	0.563	0.015	0.579
pc	0.605	0.014	0.622
sl	0.529	0.018	0.537
cw	0.523	0.009	0.529
bow + ss	0.918	0.006	0.921
bow + ss + mw	0.567	0.014	0.592
bow + ss + pc	0.605	0.014	0.625
bow + ss + cw	0.537	0.021	0.544

Слика 5.6 "news_dataset" скуп података, метод подржавајућих вектора, где BoW користи 1500 најфреквентнијих 1-грама и 2-грама.

Ознака модела	Тачност унакрсном валидацијом	Стандардна девијација	Тачност на тест скупу
bow	**0.933**	**0.006**	**0.941**
ss	0.608	0.007	0.625
mw	0.563	0.015	0.579
pc	0.605	0.014	0.622
sl	0.529	0.018	0.537
cw	0.523	0.009	0.529
bow + ss	0.918	0.003	0.931
bow + ss + mw	0.567	0.014	0.592
bow + ss + pc	0.605	0.014	0.625
bow + ss + cw	0.537	0.021	0.544

Слика 5.7 "news_dataset" скуп података, метод подржавајућих вектора, где BoW користи 3000 најфреквентнијих 1-грама и 2-грама.



Слика 5.8 Приказ тачности на тренинг подацима кроз итерације унакрсне валидације за метод подржавајућих вектора, где BoW користи 3000 најфреквентнијих 1-грама и 2-грама, на "news_dataset" скупу података.

Представљени резултати показују да техника која најбоље раздваја позитивне од негативних истанци је *Bag-Of-Words*, такође у неким случајевима се може видети како остале технике, као што је плитка синтакса, у комбинацији са BOW могу да побољшају тачност. Највећу оцену тачност на тест подацима, 0.941, и тачност валидације у експерименталним резултатима, 0.933 са стандардном девијацијом 0.003, има модел који користи метод подржавајућих вектора, у претпроцесирању користи технику *Bag-Of-Words* са 3000 најфреквентнијих 1-грама и 2-грама. Због мале разлике у оцени, а велике разлике у

броју улазних параметара, у имплементацији овог рада искоришћен је модел који користи метод подржавајућих вектора и прати 1500 најфреквентнијих 1-грама и 2-грама, овај модел има оцену 0.933 и стандардну девијацију 0.002.

Треба узети у обзир да, у реалном свету, различитих вести и начина писања има много. То значи да се не може очекивати да преложени модел добро ради за све стилове писања који се могу наћи на Интернету. Тако модел трениран на скупу података *"news_dataset"* са тачношћу од 0.933, када се примени на податке из скупа *"real_fake_dataset"* даје тачност 0.554.

6 *Chrome* додатак за препознавање лажних вести

Додатак или екстензија за прегледач је мали програм који се ивршава у Интернет прегледачу и користе се за мењање, манипулисање или контролисање елемената странице [25]. Овакви програми су обично имплементирани у познатим веб технологијама као што су *HTML, JavaScript, CSS*. Даље ће бити детаљније објашњена имплементација додатка за *Google Chrome* прегледач.

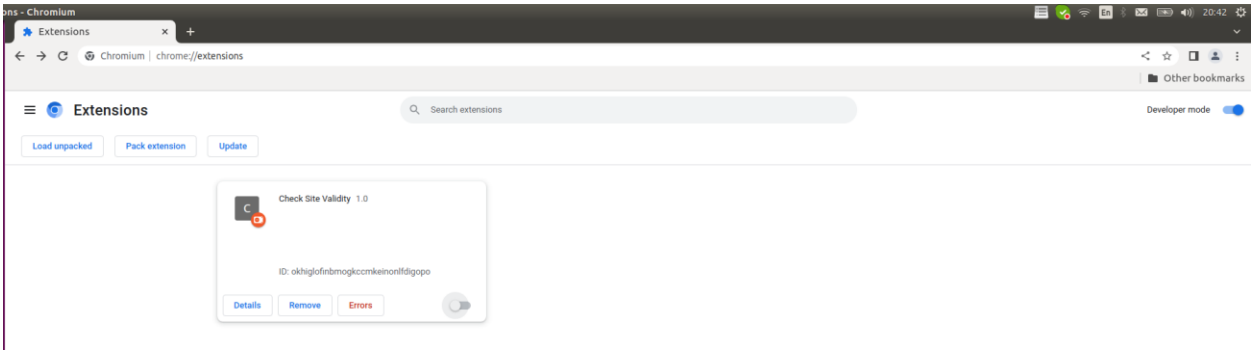
Прва компонента екстензије је *manifest* датотека, која је *JSON* формата, и садржи својства, информације и подешавања додатка за прегледач. Као основне информације у *manifest* датотеци наводе се верзија *manifesta* која се користи (тренутно актуелна верзија је 2), име екстензије, иконица и верзија екстензије која ће бити видљива корисницима. Такође, у *manifest* датотеци се наводе и остале компоненте од којих је додатак за прегледач сачињен, као и њихове дозволе. Неке од компоненти које се користе су *content_scripts, background, browser_action* и *permissions* компонента.

- *content_scripts* компонента, дефинише информације кроз датотеке писане у *JavaScript* језику. Сврха ових скрипти је да ослушкују догађаје и реагују са одговарајући начин. Догађаји могу бити иницирани од стране *content_scripts* компоненте, веб странице или другог додатка за прегледач, док инструкције могу извршавати познате *JavaScript* методе или приступати тренутним елементима странице и мењати их.

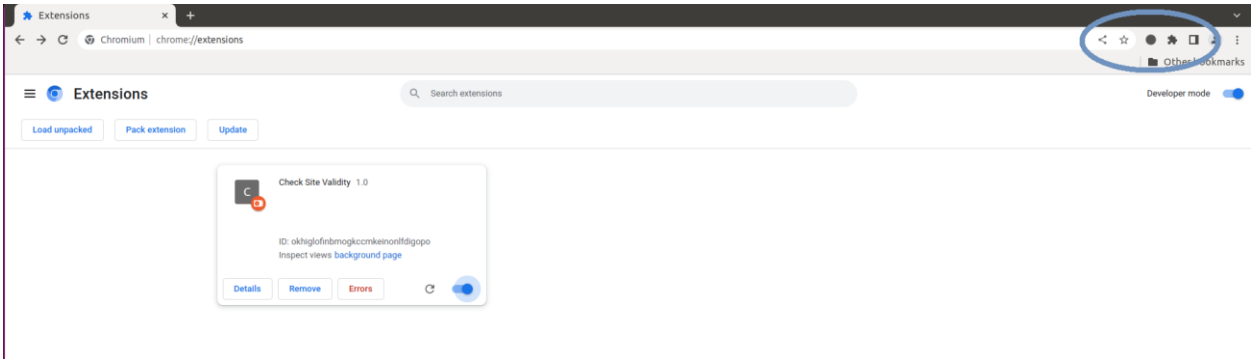
- *browser_action* компонента, дефинише информације и датотеке такође писане у *JavaScript* језику. Ова компонента се користи за прилагођавање самог изгледа додатка за прегледач који се налази у горњем десном углу прозора. Може се мењати иконица додатка, као и додавање самих акција приликом интеракције са интеракције са иконицом додатка.

- *permissions* компонента, дефинише низ регуларних израза који представљају шаблоне имена страница над којима је додатак применљив.

Учитавање додатка у сам прегледач врши се са странице *chroma://extensions/*. Довољно је унети путању до датотеке у којој се налази *manifest* датотека жељеног додатка. После чега ће се он наћи у листи подржаних додатака, где се дугметом контролише да ли је додатак активан или не.



Слика 6.1 Приказ листе додатака за прегледач (приказ неактивираног додатка за прегледач).



Слика 6.2 Приказ активираног додатка за прегледач.

6.1 *Flask* сервер

Flask је веб окружење за писање *Python* апликација написано у језику *Python*, које је засновано на *WSGI* сету алата. *WSGI* (енгл. *Web Server Gateway Interface)* је спецификација која описује како веб сервер комуницира са веб апликацијама писаним у језику *Python*. Језик *Python* није нативни језик за писање веб апликација, па у том случају коришћењем *WSGI* интефејса, веб север као што је *Apache*, комуницира са *Python* језиком као да је нативни језик. Основно *Flask* језгро је једноставно, али прошириво, тако да иницијално нема подржане неке од захтева које имају веб апликације, као што су унапред изабрана база података, начин провере образаца, технологије аутентификације и друге, него даје слободу

ономе ко развија апликације да сам изабере и укључи неке од већ развијених *Python* библиотека [26].

У наставку је приказан један пример једноставне *"Hello World" Flask* апликације која се извршава од стране сервера.

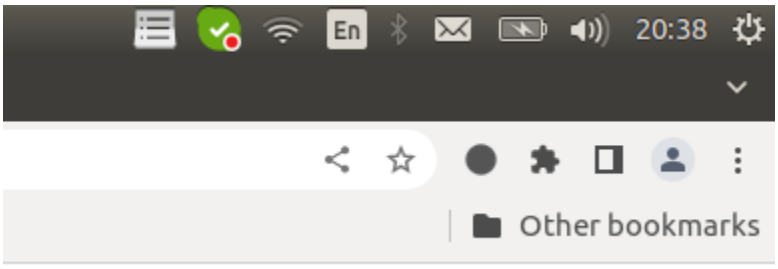
```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
   return "Hello World"


if _name_ == "__main__":
   app.run(debug=False)
```

Код 6.3 Пример једноставне "Hello World" Flask апликације.

Ова апликација је довољна да се пресретне захтев http://host_address:port/ и врати стринг *"Hello World"*. Иако језик *Python* није нативни језик за писање веб апликација, из примера се види да због једноставног *Flask* језгра писање веб апликација у језику *Python* постаје једноставно.

6.2 Комуникација додатка за прегледач и *Flask* сервера

Да би се успешно класификовала страница на Интернету, потребно је садржај странице искористити као улаз у један од модела машинског учења, а затим у зависности од излаза модела сигнализирати кориснику да ли је садржај странице истинит или лажан.

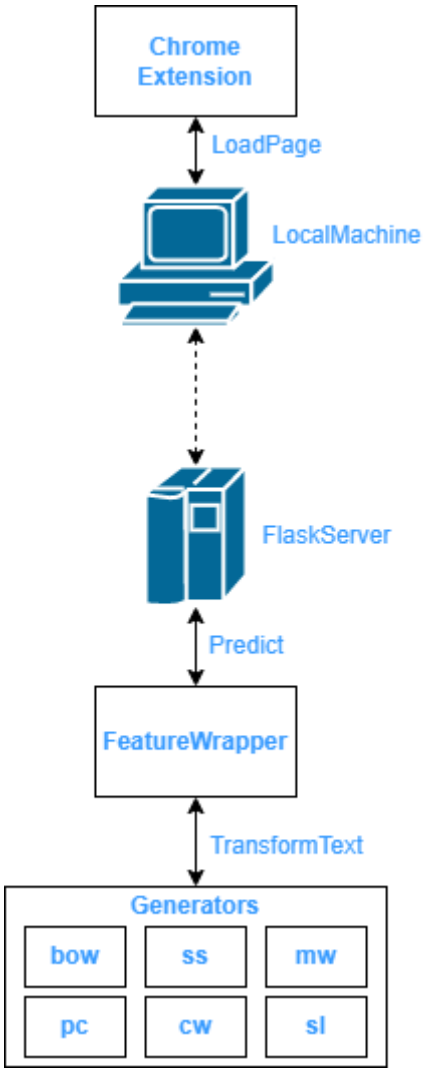


Слика 6.4 Приказ неутралног стања индикатора сиви кружић (током учитавања странице).

Претходно је речено да додатак за прегледач може да контролише елементе странице, па самим тим и да приступи текстуалном садржају странице. Такође је описан начин имплементације додатка за прегледач. Имплементација се састоји из дела који описује додатак кроз *manifest* датотеку и дела који се извршава од стране прегледача, писан у *JavaScript* програмском језику. Како се за предвиђање у машинском учењу користе

библиотеке у језику *Python*, текст странице је потребно да се обради у делу кода који је писан у језику *Python*.

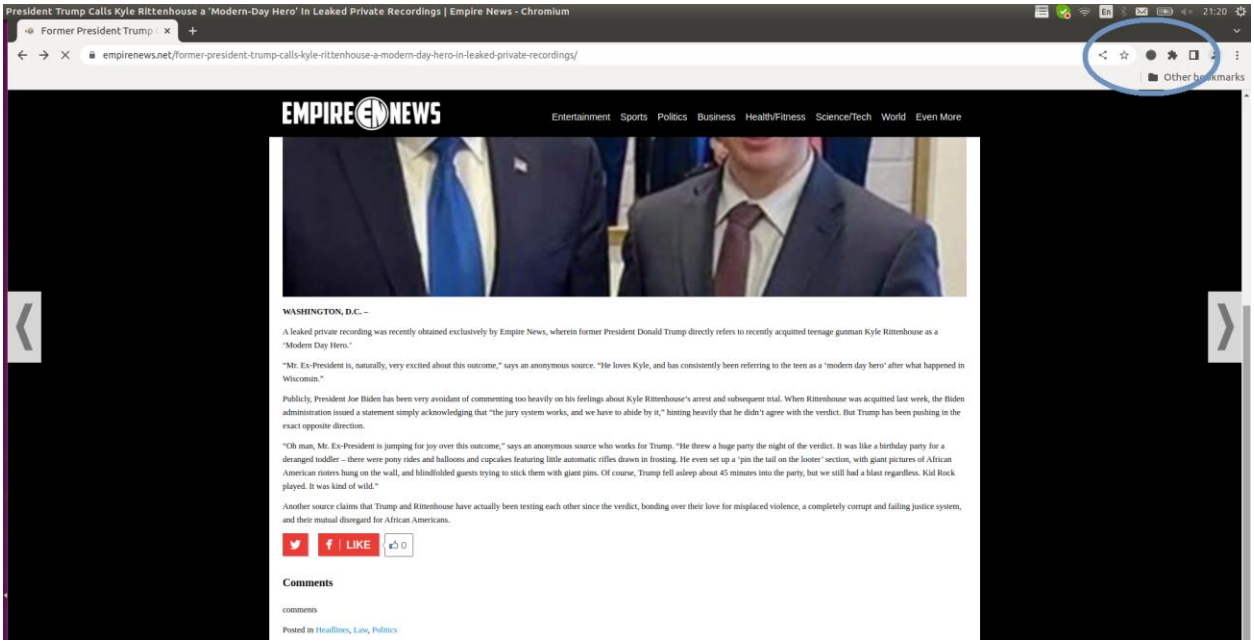
Постоји више начина да додатак за прегледач буде написан или да извршава делове кода писане у језику *Python*. Неки од њих су, коришћење окружења за писање клијентских веб апликација *Brython* [27], или компајлирањем *Python* кода у *JavaScript* код коришћењем разних решења као што је *Transcrypt* [28]. Недостатак ових техника је тај што не могу да укључе библиотеке за машинско учење (*scikit-lean*, *nltk*, и друге). Зато ће у наставку бити описан предложени приступ како овај недостатак може да се реши коришћењем *Flask* сервера.



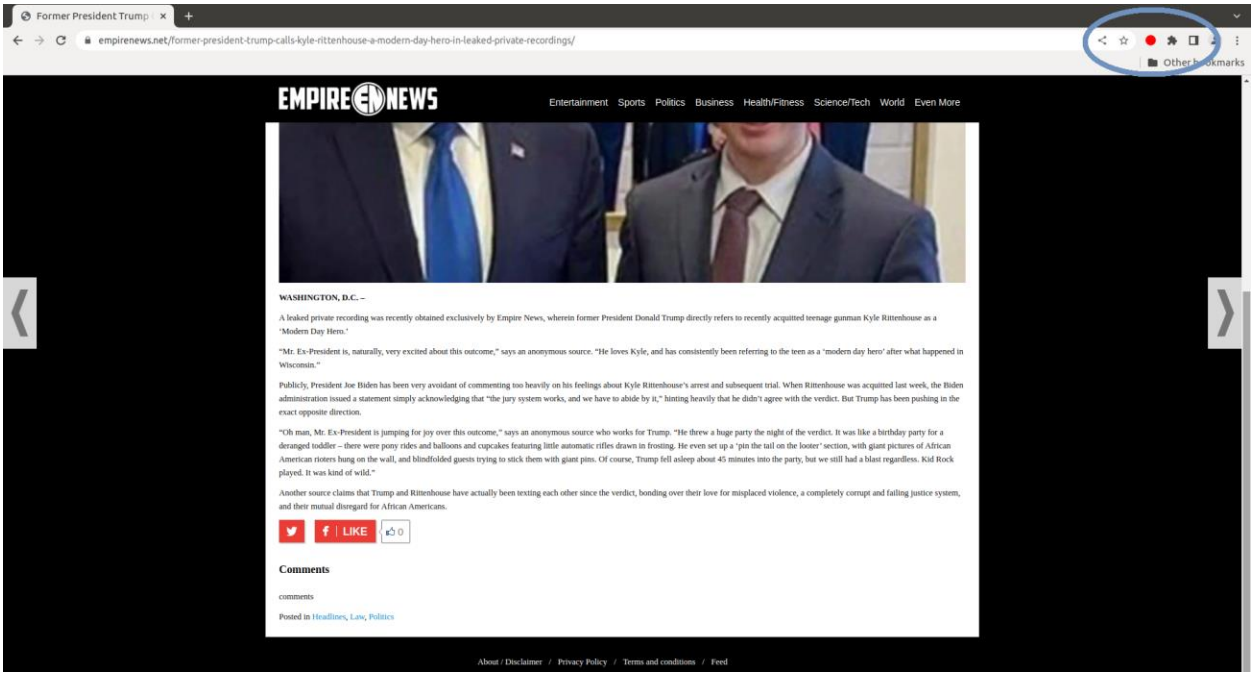
Слика 6.5 Поглед високог нивоу комуникације између додатка за прегледач и Python апликације која врши предвиђање.

- Додатак за прегледач, унутар *content_script* компоненте, ослушкује учитавање нове странице.
- Када се нова страница учита, догађај се шаље *background* компоненти додатка за прегледач.
- Додатак за прегледач коришћењем *background* компоненте шаље веб захтев, преко *Flask* сервера, *Python* апликацији.
- *Flask* сервер путем WSGI интерфејса даје могућност веб серверу да веб захтев (послат од стране додатка за прегледач) проследи *Python* апликацији, која затим обрађује захтев и враћа веб одговор. Сама *Python* апликација може да укључи било коју од *Python* библиотека, као и библиотеке за машинско учење.

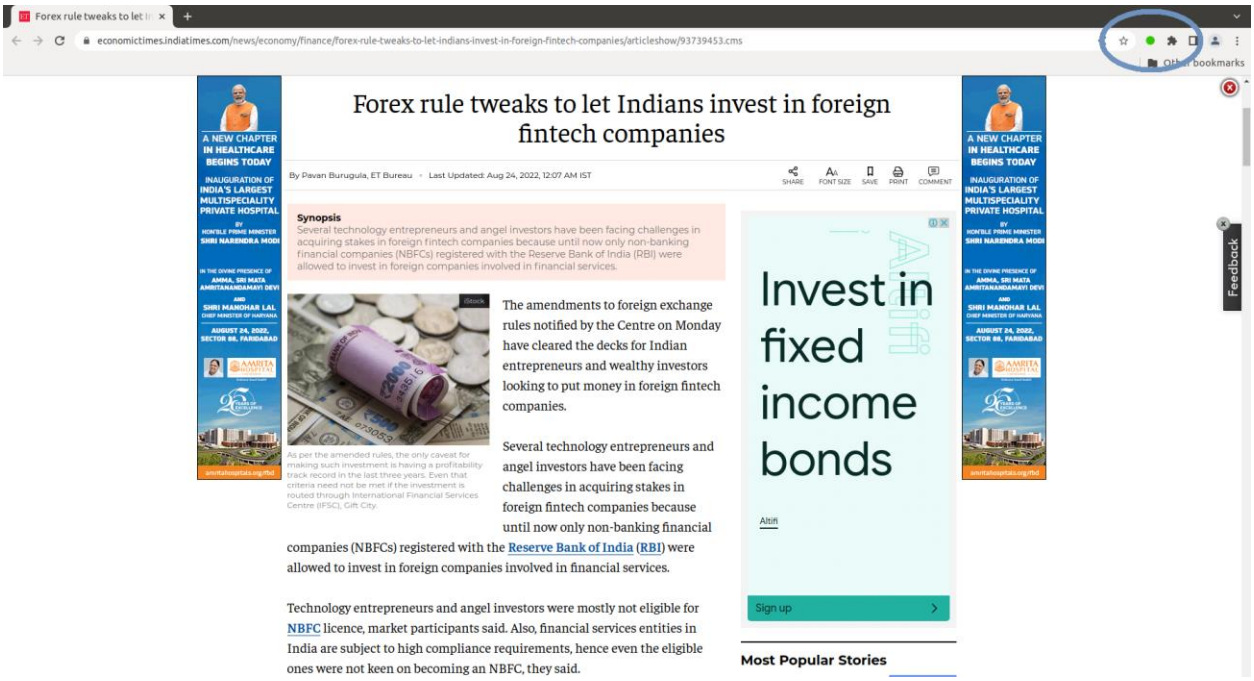
У захтеву који шаље додатак за прегледач мора да се нађе текст над којим ће се вршити предвиђање. Веб страница, поред текста новинског чланка, садржи текстуалне податке који се не тичу самог новинског чланка, већ се тичу неке друге странице, текстове које се тичу реклама, описне текстове саме странице, елементе који садрже видео садржаје или слике, и друге. Издвајање текста новинског чланка из *HTML* странице је компликован проблем, зато је предложено коришћење *Newspaper Python* библиотеке која служи за издвајање и парсирање новиниских чланака. *Newspaper* библиотека такође не може да се учита у додатак за прегледач. Зато додатак за прегледач, унутар захтева *Python* апликацији уместо текста странице, шаље изворни *URL* странице над којом се врши предвиђање, после чега *Python* апликација, уз помоћ *Newspaper* библиотеке, издваја користан текст новинског чланка из *HTML* странице. Издвојен текст чланка (енгл. *article*) користи се као улаз у претходно истрениран модел. Резултат предвиђања је индикатор да ли је дати новински чланак истинит или лажан, који *Flask* сервер враћа прегледачу путем веб одговора. На крају веб прегледач на основу веб одговара приказује кориснику да ли је тренутна страница истинита или лажна, у облику индикатора који мења боју, црвена за лажне текстове или зелена за истините текстове.



Слика 6.6 Приказ неутралног индикатора током учитавања странице, сива боја.



Слика 6.7 Приказ црвеног индикатора који сигнализира лажну страницу.



Слика 6.8 Приказ зеленог индикатора који сигнализира истиниту страницу.

У наставку је дат пример кода *manifest* датотеке додатка за прегледач, унутар којег је дефинисано име “Check Site Validity”, верзија манифест датотеке, почетна иконица индикатора боје која је у овом случају сива, листа датотека који дефинишу *background* и *content_script* компоненту и на крају права приступа додатка за прегледач који се у овом случају примењује на било коју веб страницу.

```
{
    "name": "Check Site Validity",
    "version": "1.0",
    "manifest_version": 2,

    "icons": {
        "38": "neutral.png"
    },
    "background": {
        "scripts": ["jquery.min.js", "background.js"],
   "persistent" : true
    },

    "content_scripts" : [
        {
            "matches" : ["<all_urls>"],
            "js": ["ContentScript.js"],
            "run_at": "document_end"
        }
    ],
    "browser_action": {
    },

    "permissions": [
        "*://*/*", "http://*/*", "https://*/*"
    ]
}
```

Код 6.9 Део кода manifest датотеке додатка за прегледач.

У наставку је дат пример кода додатка за прегледач унутар *content_script* компоненте који ослушкује учитавање странице. На акцију учитавања странице *content_script* компонента шаље *URL* странице *background* компоненти.

```
window.onload = function(){
    chrome.runtime.sendMessage({message: "listeners",
                                type: "background",
                                content: window.location.href},
                               function(response) {}
    );
};
```

Код 6.10 Део кода додатка за прегледач унутар content_script компоненте.

У наставку је дат пример кода додатка за прегледач унутар *background* компоненте која прима од компоненте *content_script* поруку са *URL*-ом тренутне странице, затим поставља тренутну боју индикатора на сиву, после чега захтев са *URL*-ом тренутне странице шаље на предвиђање *Python* апликацији, на крају обрађује одговор *Python* aplikacije, у случају да је одговор 1 поставља боју индикатора на зелену у супротном на црвену.

```
var serverURL = "http://127.0.0.1:5000"

chrome.runtime.onMessage.addListener(
    function(request, sender, sendResponse) {
        if (request.message == "listeners" && request.type == "background") {
         chrome.browserAction.setIcon({path: "neutral.png"});
         $.ajax({
            url: serverURL + "/_predict/",
            type: "POST",
            data: { arg1: request.content} ,
            success: function(resp){
              console.log(resp);
                        if (resp.data[1] == "1") {
                 chrome.browserAction.setIcon({path: "true.png"});
              } else {
                 chrome.browserAction.setIcon({path: "false.png"});
              }
            },
            error: function(e, s, t) {
              console.log("ERROR OCCURRED");
              console.log(e);
              console.log(s);
              console.log(t);
            }

         });
          return true;
      }
   }
);
```

Код 6.11 Део кода додатка за прегледач унутар background компоненте.

У наставку је приказан део кода Python апликације која покреће *Flask* сервер и обрађује захтев који стиже *POST* методом на адреси http://127.0.0.1:5000/_predict/, врши предвиђање и враћа као одговор 0 или 1 у зависности да ли је текст истинит или лажан. За предвиђање *Python* апликација користи метод подржавајућих вектора, фреквенцију 1500 најфреквентинијих 1-грама и 2-грама.

```
from newspaper import Article
import Consts
from flask import Flask, render_template, jsonify, request
from flask_cors import CORS, cross_origin
from FeatureWrapper import FeatureWrapper

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
features = FeatureWrapper()
@app.route('/_predict/', methods=['POST'])
def _predict():
    article = Article(request.form['arg1'])
    article.download()
    article.parse()
    predicted = features.ScoreText(article.text,
                                   Consts.INPUT_COMMANDS,
                                   Consts.INPUT_DIR_PATH,
                                   pcaModelPath=Consts.INPUT_DIR_PATH
                                                + Consts.PCA_PICKLE_PATH,
                                   transformModelPath=Consts.INPUT_DIR_PATH
                                                      + Consts.TFIDF_PICKLE_PATH)

    return jsonify({'data' : str(predicted)})

if __name__ == "__main__":
    app.run(debug=True)
```

Слика 6.12 Део кода Python апликације која покреће Flask сервер.

7 Закључак

Главни циљ овог рада је да детаљно опише и представи једно решење имплементације алгоритма за класификовање лажних вести на Интернету. Представљено решење или његови делови могу бити врло лако интегрисани у реалан пројекат. Поред детаљног описа два алгоритма машинског учења, једног једноставног као што је линеарна регресија и једног сложенијег као што је метод подржавајућих вектора, описана је и имплементација најчешће коришћених техника за претпроцесирање података у обради природних језика, нормализација, *Bag-Of-Words*, фреквенција знакова интерпункције, фреквенција правописних грешака, просечна дужина речи и реченица, фреквенција речи писаних скроз великим словом и плитка синтакса. Такође, за контролу димензије улазних података дат је опис технике анализе главних компоненти. Представљен је је и детаљан опис интеграције *Flask* сервера и имплементације додатка за прегледач. Даље унапређење могло би се огледати у интегрисању моћне технике за обраду природних језика *word2vec* [29].

8 Референце

[1] Social Media and Fake News in the 2016 Election, Hunt Allcott and Matthew Gentzkow.
[2] Is the Internet causing political polarization? Evvidence from demographics, Levi Boxell, Matthew Gentzkow, Jesse M. Shapiro.
[3] Примене метахеуристике засноване на електромагнетизму у решавању проблема класификације, Александар А. Картељ
[4] Machine Learning: A Probabilistic Perspective, Kevin P. Murphy.
[5] Pattern Recognition and Machine Learning, Christopher M. Bishop.
[6] Constrained Optimization and Lagrange Lagrange Multiplier Methods, Dimitri P. Bertsekas.
[7] Mašinsko Učenje, Mladen Nikolić, Andelka Zečević
[8] Natural Language Processing with Python, Steven Bird, Ewan Klein and Edward Loper.
[9] An Introduction to Information Retrieval, Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf
[10] Natural Language Processing with Python. O'Reilly Media Inc., Bird Steven, Edward Loper and Ewan Klein (2009)
[11] We Built a Fake News & Click-bait Filter:What Happened Next Will Blow Your Mind! https://www.acl-bg.org/proceedings/2017/RANLP%202017/pdf/RANLP045.pdf
[12] Combining Machine Learning with Knowledge Engineering to detect Fake News in Social Networks-a survey
https://www.researchgate.net/publication/344327978_Combining_Machine_Learning_with_Knowledge_Engineering_to_detect_Fake_News_in_Social_Networks-a_survey
[13] Shallow Text Analysis and Machine Learning for Authorship Attribtion.
https://www.researchgate.net/publication/220778266_Shallow_Text_Analysis_and_Machine_Learning_for_Authorship_Attribtion
[14] Python Documentation, https://www.python.org/doc/
[15] NumPy user guide, https://numpy.org/doc/stable/user/
[16] Scikit-learn: Machine Learning in Python,
https://jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
[17] Natural Language Processing with Python. O'Reilly Media Inc., Bird Steven, Edward Loper and Ewan Klein (2009).
[18] WordNet Documentation, https://wordnet.princeton.edu/documentation
[19] Enchant by Dom Lachowicz, http://www.abisource.com/enchant/
[20] Newspaper3k: Article scraping & curation,
https://newspaper.readthedocs.io/en/latest/index.html
[21] pickle — Python object serialization, https://docs.python.org/3/library/pickle.html
[22] Getting Real about Fake News, Meg Risdal, https://www.kaggle.com/mrisdal/fake-news
[23] BBC News Archive, https://www.kaggle.com/datasets/hgultekin/bbcnewsarchive

[24] News Data Set – Fake OR Real https://www.kaggle.com/datasets/vikasukani/news-data-set-fake-news-with-python
[25] What are extensions? Chrome Developers, https://developer.chrome.com/docs/extensions/mv2/overview/
[26] Flask User`s Guide, https://flask.palletsprojects.com/en/2.2.0/
[27] Brython, https://brython.info/static_doc/en/intro.html
[28] Transcrypt, Python in Browser https://www.transcrypt.org/documentation
[29] Speech and Language Processing, Daniel Jurafsky and James H. Martin.

Биографија аутора

Александар Шука рођен је 28.06.1991. године у Београду. Смер информатика на Математичком факултету је уписао 2010. године, а завршио 2015. године. Након тога је уписао мастер студије информатика на истом факултету.

Септембра 2015. године почео је са сталним запослењем на позицији млађег *C/C++* софтверског инжењера у начном институту *RT-RK*. Овде ради на развоју софтвера за *set-top box* уређаје у дигиталној телевизији. Као члан тима од десет људи успешно је завршио пројекат за индијску муштерију, *Nagra Vision*, која је произвела седам милиона *RTOS* базираних *set-top box* уређаја са *RT-RK* софтверским решењем.

Јула 2017. године прелази на позицију средњег софтвер инжењера, где је развијао софтвер за *set-top box* уређаје на бази *Android* оперативног система за јапанску муштерију, *NTT Docomo*. Због потреба пројекта проводи једну годину у Токију, Јапан. Пројекат је успешно завршио као члан тима од петнаест људи и муштерија је произвела петсто хиљада *Android* базираних *set-top box* уређаја са *RT-RK* софтверским решењем.

Јула 2019. прелази на позицију старијег софтвер инжењера за индијску муштерију, *Jio Reliance*. Овде ради на развијању софтвера за *Android* базиране *set-top box* уређаје. Због потреба пројекта проводи девет месеци у Мумбају, Индија.

Априла 2020. године прелази на позицију вође програма. Почиње да води више пројеката за муштерију *Jio Reliance* са тимом већим од тридесет и пет људи. Муштерија је произвела пет милиона *Android* базираних *set-top box* уређаја са *RT-RK* софтверским решењем.

Фебруара 2021. године научни институт *RT-RK* отвара канцеларију *Three Meadows* у Бангалору, Индија, са циљем да оствари своје присуство у Азији. Александар фебруара 2021. бива постављен за инжењеринг директора *Three Meadows* канцеларије, због чега одлази у Бангалор, Индија. Овде је директно одговоран за формирање и раст канцеларије на више од осамдесет људи.