Primjena metoda strojnog učenja u analizi korisničkih upita

Komljenović, Nikola

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:679105

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PRIMJENA METODA STROJNOG UČENJA U ANALIZI KORISNIČKIH UPITA

Diplomski rad

Nikola Komljenović

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 22.09.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Nikola Komljenović
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 820 R, 09.10.2015.
OIB studenta:	74920464278
Mentor:	Doc.dr.sc. Ratko Grbić
Sumentor:	
Sumentor iz tvrtke:	Ivan Vučak
Predsjednik Povjerenstva:	Doc.dr.sc. Emmanuel-Karlo Nyarko
Član Povjerenstva:	Doc.dr.sc. Josip Job
Naslov diplomskog rada:	Primjena metoda strojnog učenja u analizi korisničkih upita
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U okviru diplomskog rada potrebno je napraviti pregled trenutno aktualnih metoda strojnog učenja za analizu tekstualnih dokumenata. Izgraditi skup podataka dohvaćanjem korisničkih upita iz različitih izvora te izgraditi modele nadziranim i nenadziranim učenjem koji omogućuju grupiranje i klasifikaciju dokumenata. Model implementirati u odgovarajuću informatičku platformu (npr. Apache Hadoop). (sumentor Ivan Vučak, Sedam IT, Zagreb)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	22.09.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 05.10.2017.

Ime i prezime studenta:	Nikola Komljenović
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 820 R, 09.10.2015.
Ephorus podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Primjena metoda strojnog učenja u analizi korisničkih upita**

izrađen pod vodstvom mentora Doc.dr.sc. Ratko Grbić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
2. KORISNIČKI UPITI I DOHVAĆANJE PODATAKA 3
2.1. Korisnički upiti na društvenim mrežama 3
2.1.1. Facebook 3
2.1.2. Twitter 4
2.2. Dohvaćanje podataka 6
2.2.1. Dohvaćanje podataka koristeći API 6
2.2.2. Dohvaćanje podataka koristeći Quintly 7
3. STROJNO UČENJE 9
3.1. Vrste strojnog učenja 10
3.1.1. Nadzirano učenje 10
3.1.2. Nenadzirano učenje 11
3.2. Klasifikacija i regresija 11
3.3. Obrada prirodnog jezika 12
3.4. Standardni algoritmi strojnog učenja 12
3.4.1. Naivni Bayesov klasifikator 13
3.4.2. Strojevi s potpornim vektorima 14
4. OBRADA PODATAKA 16
4.1. Klasifikacija korisničkih upita 16
4.2. Python biblioteke 18
4.2.1. Pandas 18
4.2.2. NLTK 19
4.2.3. scikit-learn 20
4.3. Obrada upita 20
4.3.1. Normalizacija i interpunkcijski znakovi 20
4.3.2. Stop riječi i tokenizacija 21

4.4. Morfološka normalizacija 23
4.4.1. Korjenovanje 23
5. IZGRADNJA I IMPLEMENTACIJA KLASIFIKATORA 25
5.1. Naivni Bayes i SVM model klasifikatora 25
5.1.1. Skup riječi 26
5.1.2. Vektorizacija 27
5.2. Evaluacija modela 27
5.3. Implementacija klasifikatora u Web okruženje 28
5.3.1. Flask Microframework 29
5.3.2. Implementacija API-ja 30
5.3.3. Python Anywhere 31
6. ZAKLJUČAK 34
LITERATURA 35
SAŽETAK 38
ABSTRACT 39
ŽIVOTOPIS 40
PRILOZI 41

1. UVOD

Strojno učenje je postalo središnja tehnologija koja se u današnje vrijeme pokušava progurati u sve pore Interneta. Njegovu primjenu možemo vidjeti u svim tražilicama, internetskim trgovinama gdje nam se uz pomoć strojnog učenja pokušava ponuditi baš ono što u tom trenutku želimo. Uz sve te primjene kroz ovaj rad želi se pokazati kako primijeniti strojno učenje u obradi prirodnog jezika. Jezik je vrlo kompleksna stvar kada je u pitanju računalna obrada. Tekst je potrebno najprije pretvoriti u računalu razumljiv oblik. Ovaj rad obrađuje mogućnost primjene i implementacije računalnih algoritama dobivenih strojnim učenjem za potrebe određivanja klase upita kojeg je korisnik uputio pružatelju telekomunikacijskih usluga. Algoritam će biti implementiran u obliku web aplikacije u *cloud* servisu. Izazov ovoga rada upravo je pokušaj primjene takvih algoritama tj. klasifikatora koristeći upite na hrvatskom jeziku. Većina izrađenih klasifikatora odnosi se na engleski jezik koji je najzastupljeniji u današnjem svijetu. Postoji nekoliko izvedbi klasifikatora na hrvatskom jeziku koji su dali dobre rezultate u postupku klasifikacije, ali je i dalje jako mala zainteresiranost za razvoj sustava koji obrađuju hrvatski jezik.

Kako je Internet postao središte komunikacije tako su se i svi oblici korisničke podrške preselili na Internet. Danas je gotovo nezamislivo da neki od pružatelja telekomunikacijskih usluga nemaju profil na nekoj od društvenih mreža ili mogućnost komunikacije putem razgovora na svojoj internetskoj stranici. Korisnici se sve više okreću tom načinu komunikacije gdje ne moraju imati izravan telefonski kontakt s operaterom koji im pruža pomoć nego sve žele obaviti sa svog računala ili mobilnog telefona u obliku tekstualne poruke. Kako se povećava broj korisnika koji na ovaj način žele komunicirati tako i pružatelji usluga moraju osigurati da svaki korisnik u što kraćem vremenu dobije odgovor na svoj upit. Automatska (računalna) klasifikacija svakog korisnikovog upita, uz osnovne informacije koje bi on zajedno sa upitom dostavio, omogućile bi da operateri efikasnije dodjeljuju upite tehničkoj/korisničkoj službi te na taj način povećaju kvalitetu cjelokupnog sustava podrške. Kroz ovaj diplomski rad nastoji se pokazati primjenjivost strojnog učenja kod ovakvih sustava tj. problema.

Rad je strukturiran na sljedeći način. U drugom poglavlju prikazan je način na koji korisnici komuniciraju sa pružateljima usluga, primjeri upita te koje su sve društvene mreže korištene za dohvat podataka koji će se obrađivati u ovom radu. Uz to opisan je način dohvaćanja podataka sa društvenih mreža te koje su se tehnologije i servisi koristili za to.

Treće poglavlje opisuje općenito što je to strojno učenje, kako se primjenjuje na obradu prirodnog jezika. Definirani su klasifikatori koji će se koristiti u radu te teorijska podloga nadziranog i nenadziranog učenja. U četvrtom poglavlju opisano je kojim klasama i na koji način su korisnički upiti ručno klasificirani i pripremljeni za izgradnju klasifikatora. Opisan je i programski jezik Python i svi njegovi alati i moduli koji će služiti za obradu upita. Detaljan opis postupaka dan je u ovom poglavlju sa svim programskim kodovima koji su korišteni u tom procesu. Peto poglavlje detaljno opisuje način izgradnje klasifikatora, korištene algoritme za klasifikaciju i njihovu implementaciju u programskom kodu. Implementacija klasifikatora u web aplikaciju korištenjem API-ja (engl. *Application Programming Interface*) je također opisana u ovom poglavlju kao i sve tehnologije koje su korištene u tim koracima. Šesto poglavlje, koje je i rezimirajuće, opisuje dobivene rezultate i ostvarene ciljeve. Zaključuju se nova saznanja koja su nastala nakon izrade ovog diplomskog rada. Opisuje mogućnosti unaprjeđenja izgrađenog sustava.

2. KORISNIČKI UPITI I DOHVAĆANJE PODATAKA

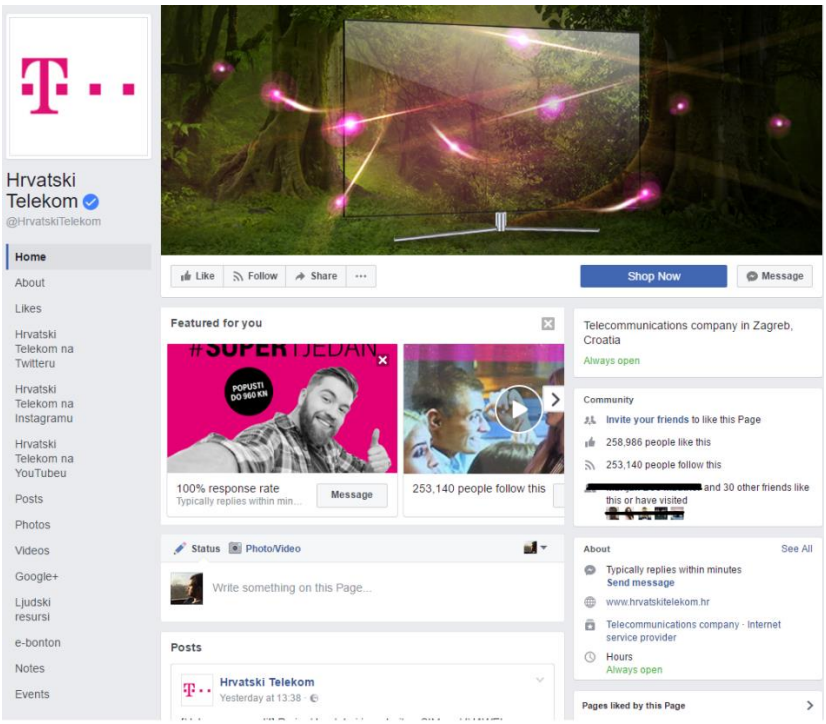
Internet postaje glavno središte svih oblika komunikacije pa tako i korisnika sa pružateljima usluga poput Interneta, televizije ili fiksne telefonije. Danas se sve više korisnika obraća pružatelju usluga putem društvenih mreža ukoliko postoje poteškoće sa internetskim pristupom ili nekom od ostalih usluga. Korisnici i pružatelji usluga sve više izbjegavaju telefonski kontakt te se žele iskoristiti sve moderne tehnologije i njihove mogućnosti. S druge pak strane javno dostupni upiti korisnika predstavljaju značajan izvor informacija pa se ovi podaci mogu iskoristiti za izgradnju različitih računalnih algoritama koji mogu poboljšati određene Internet servise.

2.1. Korisnički upiti na društvenim mrežama

Teško je danas sresti osobu koja barem jednom nije vidjela ili koristila društvene mreže kao što su Facebook i Twitter. One su izvor svih važnih informacija, događaji se putem njih prenose u stvarnom vremenu te je i komunikacija na njima moguća u svakom potrebnom trenutku. Ako želite kontaktirati agenta podrške, dovoljno je samo ostaviti upit na stranici ili se javiti putem razgovora i u svega nekoliko minuta stiže odgovor na problem. Na taj način izbjegavaju se duža čekanja na javljanje agenta kao kada se kontaktira podrška putem telefona.

2.1.1. Facebook

Facebook je društvena mreža osnovana 2004. godine. Danas broji više od milijardu aktivnih korisnika te je trenutno najveća društvena mreža na svijetu. Na slici 2.1. prikazana je Facebook stranica Hrvatskog Telekoma. Kroz svoj sustav za razgovore Facebook omogućava trenutnu komunikaciju što je iskorišteno kako bi se moglo pomoći korisnicima koji imaju poteškoća sa uslugama.



Sl. 2.1. *Facebook stranica pružatelja telekomunikacijskih usluga*

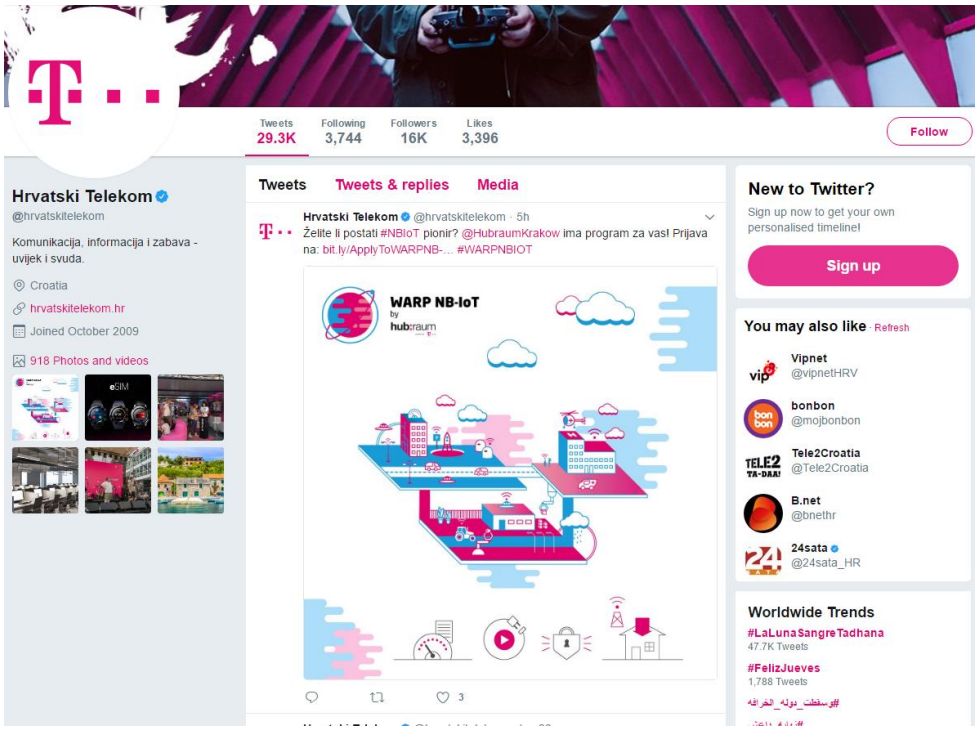
U ovom radu se za potrebe kreiranja baze upita koriste se neke od objava koje su korisnici ostavili na ovoj stranici. Objave su javne i kao takve dostupne svima da ih pročitaju. Primjer objave prikazan je na slici 2.2. te pružatelj usluge može komentarom odgovoriti na upit.



Sl. 2.2. *Upit korisnika pružatelju telekomunikacijskih usluga putem Facebooka*

2.1.2. Twitter

Twitter je nešto drugačija društvena mreža od Facebook-a. Temelji se na pisanju *tweetova* koji mogu biti dugački maksimalno 140 znakova na koje korisnici mogu odgovarati, proslijediti ih na svoj profil ili ih samo označiti oznakom sviđanja.



Sl. 2.3. *Twitter profil pružatelja telekomunikacijskih usluga*

Slika 2.3. prikazuje Twitter stranicu pružatelja usluga. Kada korisnik želi ostaviti poruku drugom korisniku tada je dovoljno da uz znak @ napiše korisničko ime onoga kome želi poslati *tweet*. Upravo na taj način korisnici mogu komunicirati sa pružateljima usluga putem Twittera što je vidljivo na slici 2.4., te pružatelj može direktno odgovoriti korisniku na taj postavljeni upit.



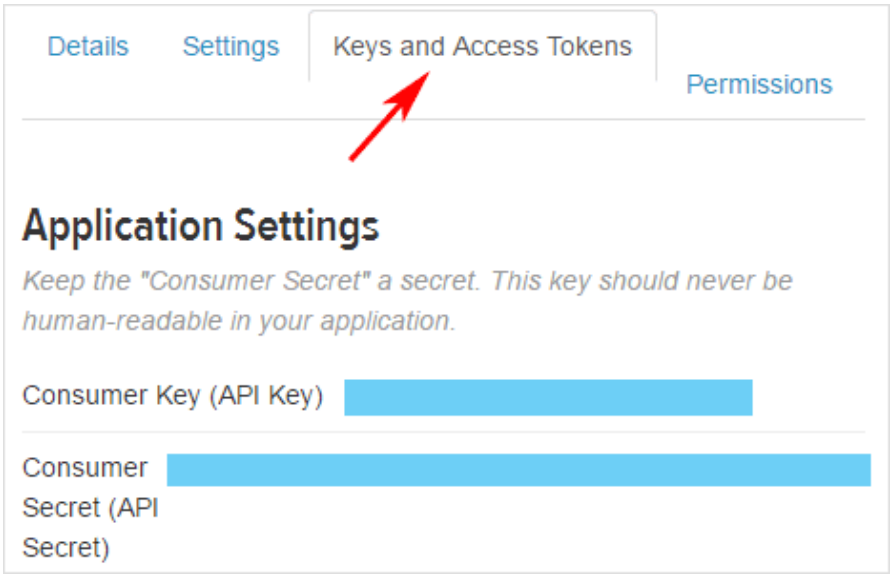
Sl. 2.4. *Upit korisnika pružatelju telekomunikacijskih usluga putem Twittera*

2.2. Dohvaćanje podataka

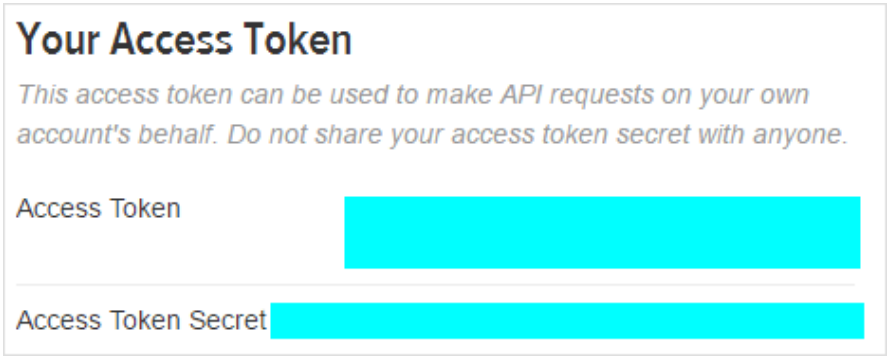
Zbog velike količine podataka koje korisnici ostavljaju na društvenim mrežama njihovo ručno dohvaćanje bio bi zamoran i dugotrajan posao. Kako bi se taj posao olakšao upotrebljava se API pristup za automatsko dohvaćanje podataka sa društvenih mreža. API je softver koji omogućuje da dvije aplikacije međusobno komuniciraju. Kada neka tvrtka ili osoba kreira softver oni često naprave i javno dostupan API kojeg onda mogu koristiti svi koji žele iskoristiti funkcionalnosti tog softvera u svom proizvodu.

2.2.1. Dohvaćanje podataka koristeći API

Za dohvaćanje podataka sa Twittera koristi se Twitter REST API. Skripta za dohvat podataka pisana je u programskom jeziku Python uz korištenje biblioteke Tweepy koja omogućava korištenje Twitter API-a, te PIP alat za upravljanje Python paketima. Prije korištenja API-a potrebno je kreirati Twitter aplikaciju gdje se dobiva API *Key*, API *Secret*, *Consumer Key* i *Consumer Secret* što je prikazano na slici 2.5. i 2.6. prema [7], te nam to koristi kako bi se mogli autorizirati. Ovaj način dobar je za kontinuirano dohvaćanje podataka sa Twittera u određenim vremenskim trenucima.



Sl. 2.5. *Privatni podaci Twitter aplikacije za autorizaciju*



Sl. 2.6. *Podaci Twitter aplikacije za autorizaciju*

Nakon što je sve postavljeno potrebno je upisati te podatke u programski kod kako bi se nakon pokretanja skripte mogla izvršiti autorizacija te započeti dohvaćanje podataka s Twitter društvene mreže. Podaci se spremaju u CSV (engl. *Comma Separated Value*) datoteku koja se sprema u lokalnu memoriju računala. Programski kod je vidljiv u prilogu P.2.1. te je najznačajniji dio ove skripte dan je u tablici 2.1. gdje se upisuju podaci za autorizaciju i pojam koji se želi pretražiti na Twitteru i na osnovu kojega se dohvaćaju podaci.

Tab. 2.1. Podaci za Twitter autorizaciju

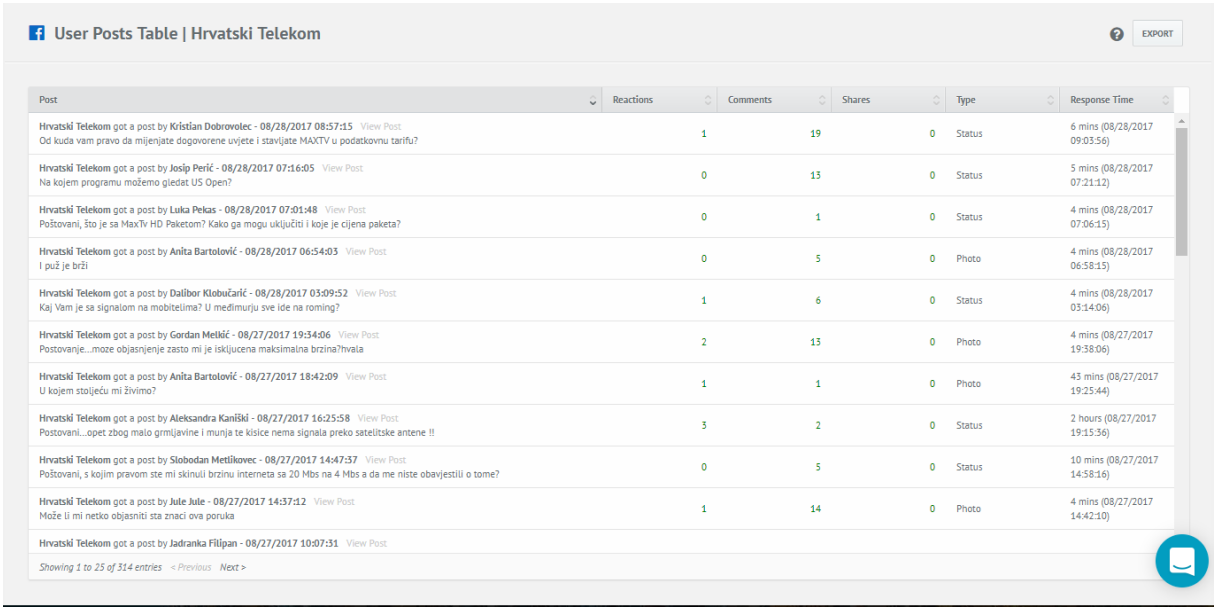
Podatak	**Opis**
consumer_key	Ključ za API pristup
consumer_secret	Tajni ključ za API pristup, ne smije biti javno dostupan.
access_token	Omogućava API zahtjeve i vezan je uz korisnički račun koji je kreirao aplikaciju.
access_token_secret	Tajni ključ za API zahtjeve.
accountvar	Pojam koji želimo pretražiti i dohvatiti

Kada su svi podaci upisani skripta je spremna za dohvaćanje podataka te ju je potrebno samo pokrenuti na računalu i pričekati neko vrijeme dok se proces ne završi i zatim pokrenuti kreiranu CSV datoteku te tako vršiti daljnju obradu.

2.2.2. Dohvaćanje podataka koristeći Quintly

Zbog zaštite privatnosti korisnika dohvaćanje podataka ponekad nije jednostavan posao. Facebook zbog svojih postavki privatnosti ne dozvoljava direktno dohvaćanje javno postavljenih objava korisnika putem API-ja. Qunitly je alat za analizu podataka sa društvenih mreža koji nudi mogućnost praćenja događanja na odabranim profilima društvenih mreža. Omogućava uvid u objave, statistiku posjećenosti, čitanja objava te dohvaćanje mnoštva

različitih podataka sa društvenih mreža. Za potrebe ovog diplomskog rada dohvaćeni su upiti korisnika sa društvenih mreža različitih pružatelja telekomunikacijskih usluga. Na slici 2.7. prikazana je tablica upita koju je moguće preuzeti u XLSX formatu koji je pogodan za daljnju obradu i pretvaranje u CSV format.



Sl. 2.7. *Korisnički upiti na Quintly-ju*

3. STROJNO UČENJE

Od samih početaka stvaranja inteligentnih sustava težilo se tome da se uz pomoć računala automatizira rješavanje različitih vrsta problema. Prvi pokušaji stvaranja inteligentnih sustava počeli su od toga da se uz pomoć ručno napisanih naredbi i uvjeta dolazi do krajnjeg rezultata ili se donese odluka na osnovu unosa korisnika. Takav pristup je primjenjiv ako se stvara sustav čiji model i proces čovjek može shvatiti i prenijeti u programski kod. Međutim, mnogi današnji problemi ne mogu se riješiti na klasičan algoritamski način. Alternativa je primjena strojnog učenja koji predstavlja drugi pogled na rješavanje problema gdje se ne pokušava postaviti fiksna pravila koja će dovest do rješenja već se uz pomoć statističke analize pokušavaju otkriti značajke koje će pomoći u učenju iz dobivenih podataka i dovesti do rješenja problema.

Gotovo je nemoguće u današnjem svijetu naprednih tehnologija zamisliti sustav koji ne koristi strojno učenje. Poznate stranice poput Facebooka, Amazona, Netflixa za svoje prijedloge proizvoda ili objava koriste jedan ili više modela strojnog učenja. Tako pri svakoj posjeti neke online trgovine dobivamo prijedloge što kupiti na osnovu onoga što smo do sada pretraživali ili kupovali, kada poželimo pronaći film koji nam odgovara tada tu također pomaže strojno učenje i predlaže ono što je najbolje za nas. Uz komercijalnu primjenu, strojno učenje je prisutno i u znanosti i istraživanju te se tako koristi u istraživanju strukture DNA, otkrivanju novih čestica te stvaranje personaliziranih terapija za liječenje raka.

Razlikujemo dvije vrste strojnog učenja, to su nadzirano i nenadzirano učenje. Osnovni pojmovi su ulazne i izlazne varijable. Ulazne varijable su primjerice uzorci teksta ili neke druge mjerljive veličine na temelju kojih se nastoji odrediti vrijednost izlazne varijable. Nadzirano učenje temelji se na tome da se algoritmu učenja predaju ulazne i izlazne vrijednosti varijabli tzv. podatkovni primjeri. Rezultat učenja je matematički model koji aproksimira ovisnost ulaznih varijabli i izlazne varijable. Kada su na raspolaganju samo vrijednosti ulaznih veličina, tada se koristi nenadzirano učenje. Princip nenadziranog učenja je pokušati na što bolji način prepoznati pravilnosti u podacima s obzirom na značajke koje su im zajedničke.

3.1. Vrste strojnog učenja

Prethodno u radu opisane su vrste strojnog učenja. Svaka od njih pogodna je za rješavanje određenih vrsta problema pa je potrebno odabrati jednu od njih koja će najuspješnije riješiti problem.

3.1.1. Nadzirano učenje

Nadzirano učenje je najčešći oblik učenja u primjeni strojnog učenja u stvarnim sustavima. Za primjenu ovog načina učenja potrebno je imati označene podatke tj. vrijednosti ulaznih i izlaznih veličina. Podatke u većini slučajeva označava čovjek koji pokušava što točnije odrediti klasu kojoj pripada podatak. Funkcija koja povezuje ulazne i izlazne varijable nakon procesa učenja omogućava predviđanje izlazne varijable za novi podatak koji želimo predvidjeti. Najjednostavniji oblik funkcije prikazan je jednadžbom (3-1) prema [10].

$$Y = f(X) \tag{3-1}$$

gdje Y predstavlja izlaznu varijablu, X su ulazne varijable, dok je $f(X)$ funkcija koja povezuje ulaz i izlaz. Učenje se naziva nadzirano zbog toga što se za svaki ulazni podatak zna koja je točno vrijednost izlaza. Dva su osnova problema koje algoritmi nadziranog učenja mogu obrađivati, a to su klasifikacija i regresija. Kod klasifikacije izlazna varijabla je diskretna vrijednost odnosno klasa kojoj pripada ulazni podatak. Primjer klasifikacije je rješavanje problema razvrstavanja elektroničke pošte na željenu i neželjenu gdje se svaka nova poruka može dodijeliti jednoj od tih klasa. Kod regresijskih problema izlazna varijabla ima kontinuirane vrijednosti kao na primjer godišnja zarada u kunama. Prema [9] primjer upotrebe regresije je procjena vrijednosti automobila. Ulazne varijable su glavni podaci o automobilu kao što su marka, godina, prijeđen kilometraža i sl., na osnovu čega algoritam pokušava odrediti okvirnu cijenu automobila.

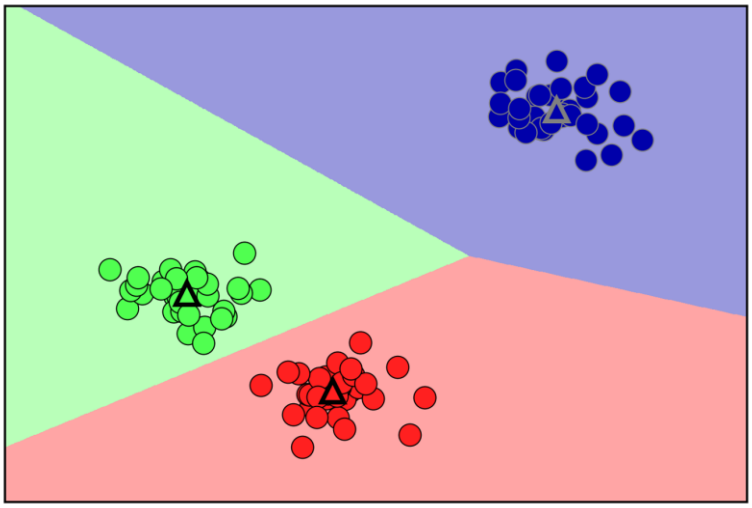
Kroz ovaj diplomski rad koristit će se klasifikacija u kojoj će ulazni podaci biti korisnički upiti upućeni pružatelju telekomunikacijskih usluga te je primjer takvih upita prikazan na slici 3.1., dok će izlazne varijable biti klase kao što su Internet, televizija, fiksne usluge, infrastruktura, mobitel, račun, ugovor, kritika, pohvala i nepoznat upit. Svaki novi upit pokušat će se dodijeliti jednoj od navedenih klasa.

Mozete mi reci kako je moguce potrositi internet u par dana a stalno koristi vifi
Sto se tocno mjenja u Zakon tarifi?Ostaju li i mb /SMS i poziviili samo sada imamo mogućnost mb?
A gdje su one zvijezdice za dobar los zao ? REVIEWS

Sl. 3.1. *Primjeri korisničkih upita*

3.1.2. Nenadzirano učenje

Za razliku od nadziranog, nenadzirano učenje nema saznanja o izlaznim varijablama koje bi mogao povezati sa ulaznim pa se upravo zbog toga zove nenadzirano učenje. Kod problema nenadziranog učenja potrebno je prepoznati pravilnosti u ulaznim podacima pa je jedan od osnovnih problema u nenadziranom učenju problem grupiranja (engl. *clustering*) podataka. Primjer grupiranja dan je na slici 3.2. prema [1, str. 170] gdje je vidljiva podjela podataka u tri grupe. Kada pristigne novi ulazni podatak on se dodjeljuje jednoj od grupa ovisno o značajkama koje posjeduje. Prema [10] najbolji primjer grupiranja u stvarnom svijetu je svrstavanje kupaca u skupine s obzirom na njihove navike kupovanja.



Sl. 3.2. *Grupiranje podataka nenadziranim učenjem*

Drugi način rješavanja problema korištenjem nenadziranog učenja je pravilo udruživanja gdje se u ulaznim podacima pronalaze pravila koja dobro opisuju dijelove podataka koje predajemo sustavu. Prema [10] primjer ovakvog načina rješavanja problema je kada želimo napraviti procjenu da li će osoba koja kupuje stvar X također ima tendenciju da kupi stvar Y. Uz sve ovo ovi se principi mogu koristiti kako bi se obradili podaci koji će se kasnije predati kao ulazi sustavu koji koristi nadzirano učenje.

3.2. Klasifikacija i regresija

Kako je već spomenuto prethodno u radu klasifikacija je problem nadziranog učenja gdje izlazna varijabla poprima diskretne vrijednosti. Podatke je u većini slučajeva potrebno ručno klasificirati prije nego ih se preda nekom od algoritama učenja. Postoje dvije vrste klasifikacije:

- Binarna – klasificiranje podataka u dvije klase, dobar primjer je neželjena elektronička pošta gdje se odlučuje da li je elektronička pošta željena ili neželjena
- Višeklasna – može imati i više od dvije klase kao npr. prepoznavanje osoba na slici.

Kada se izgrađuje sustav koji će svoju primjenu pronaći u rješavanju raznih stvarnih problema, tada se u većini slučajeva koristi višeklasna klasifikacija gdje broj klasa može biti i do nekoliko stotina. Regresija za razliku od klasifikacije, kako je navedeno prethodno u radu, ne svrstava podatke u klase već kao izlazni podatak predaje broj koji može predstavljati zaradu zaposlenika, procjenu zarade nekog poduzeća i sl. što je pogodno za skupove podataka u kojim ne postoje fiksno određene klase nego vrijednosti mogu varirati s obzirom na ulazne podatke. Regresija ima svoju domenu primjene pa se u ovome radu teško može primijeniti upravo zbog toga što je na ulazni upit potrebno odrediti točnu klasu kojoj on pripada.

3.3. Obrada prirodnog jezika

U današnjem svijetu sve je veća težnja za tim da se dobije sustav koji će u potpunosti moći razumjeti jezik i uspješno komunicirati sa osobom ili nekim drugim uređajem koji koristi sustav. Najbolji primjeri su virtualni asistenti Google Assistant, Siri i Cortana koji su dio mobilnih operacijskih sustava Android, iOS i Windows Mobile te oni spadaju u grupu sustava koji odgovaraju na upite (engl. *Question Answering Systems*). Uz njih, najpoznatija primjena obrade prirodnog jezika (engl. *Natural Language Processing*) su programi za prevođenje kao što je Google Translate koji svakodnevno postaje sve bolji u prepoznavanju konteksta i svega ostalog što obuhvaća lingvistika. Klasifikacija izraza ili dokumenata koja je također dio obrade prirodnog jezika jedan je od dobrih načina kako riješiti problem klasifikacije koji se analizira u ovom diplomskom radu.

3.4. Standardni algoritmi strojnog učenja

Prilikom izrade ovog rada koristit će se algoritmi strojnog učenja koji koriste nadzirano učenje za izgradnju klasifikatora. Danas postoje mnogi algoritmi koje je moguće primijeniti ali su za potrebe ovog rada izabrani oni koji najbolje odgovaraju problemu kojeg se želi riješiti a to je klasifikacija teksta te će se u ovom poglavlju dati teorijske osnove primijenjenih klasifikatora.

3.4.1. Naivni Bayesov klasifikator

Naivni Bayesov klasifikator koristi nadzirano učenje u izgradnji klasifikatora. Kako navode A.C., Muller i S., Guido u svojoj knjizi: „Naivni Bayesov klasifikator je dio obitelji klasifikatora koji su iznimno slični linearnim klasifikatorima koji su navedeni u poglavljima prije. Međutim, oni često znaju biti i puno brži u procesu treniranja. Cijena ove efikasnosti i brzine je da naivni Bayesovi modeli često pružaju lošije rezultate izvedbe za razliku od linearnih klasifikatora.“ [1]. Linearni modeli su često korišteni u praksi i vrše predikciju uz korištenje linearne funkcije. Za binarnu linearnu klasifikaciju koristi se jednadžba (3-2) prema [1, str. 56]

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b > 0 \tag{3-2}$$

gdje $\hat{y}$ predstavlja rezultat klasifikacije koji je u ovom slučaju linearna funkcija koja vizualno rastavlja klase. Oznake w su parametri modela, dok su x ulazne veličine.

Neke od vrsta naivnih Bayesovih modela su: *Gaussian, Bernoulli, Multinominal*. Za klasifikaciju teksta, najviše se koriste *Bernoulli* i *Multinominal* modeli. Oba ova modela koriste statistiku kao podlogu za izračunavanje rezultata, te su to modeli sa samo jednim parametrom alfa koji određuje kompleksnost modela. *Bernoulli* i *Multinominal* modeli Bayesovog klasifikatora daju vrlo dobre rezultate prilikom klasifikacije teksta.

Temelj cijelog Bayesovog klasifikatora je Bayesov teorem koji uz pomoć jednadžbe (3-3) prema [2, str. 57] izračunava uvjetnu vjerojatnost temeljenu na prethodnom znanju kojeg imamo o problemu.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3-3}$$

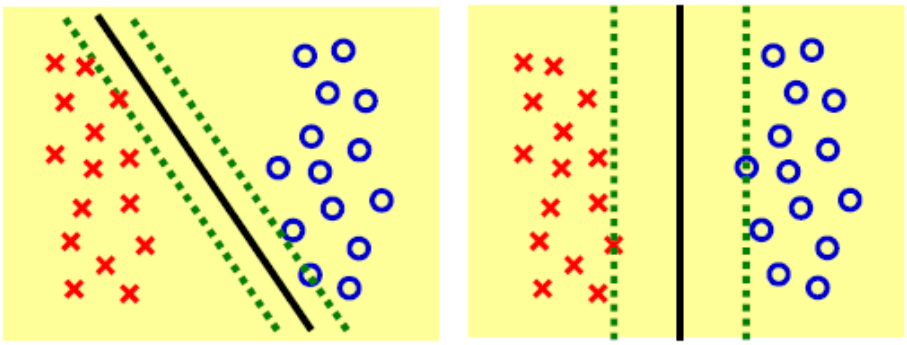
Oznaka $P(A|B)$ predstavlja tu uvjetnu vjerojatnost pojavljivanja događaja, te A i B predstavljaju događaje. $P(A)$ predstavlja vjerojatnost pojavljivanja događaja A, dok $P(B)$ predstavlja vjerojatnost pojavljivanja događaja B. Ono što Bayesov algoritam izvršava prilikom klasifikacije je izračunavanje vjerojatnosti za određenu klasu s obzirom na korisnikov upit.

Zbog prirode rada Bayesovog klasifikatora najbolji način za predavanje podataka klasifikatoru je takav da on može uočavati učestalost pojavljivanja pojedinih pojmova u upitu te na taj način može naučiti i stvoriti klasifikator koji će uspješno dobivene podatke analizirati po frekvenciji pojavljivanja određenih pojmova.

3.4.2. Strojevi s potpornim vektorima

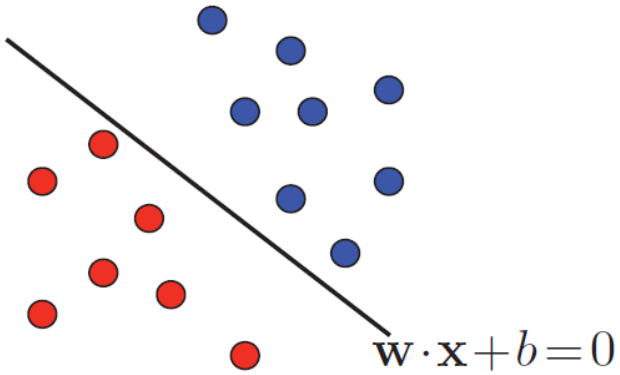
Strojevi s potpornim vektorima (engl. *Support Vector Machine*) pripadaju skupini linearnih algoritama za klasifikaciju koji koriste nadzirano učenje prilikom izgradnje klasifikatora. Temelji se na linearnoj regresiji i sadržava hiperravnine odnosno plohe koje razdvajaju klase. Prema [8, str. 337] osnova ovog algoritma je korištenje potpornih vektora koji se temelje na principu maksimalizacije margine. Što se točke klasifikacije nalaze bliže hiperravnini to je veća vjerojatnost da taj podatak graniči između više klasa pa je tako veća nesigurnost kojoj klasi pripada. Slika 3.3. prema [8, str. 304] pokazuje različite vrste margina s obzirom na to koliko su pojedini podaci blizu hiperavnini koja razdvaja klase pa tako imamo malu i veliku marginu.

Prema [8, str. 305] crna linija predstavlja granicu odluke kojoj klasi će pripasti element te ona prolazi kroz sredinu dvaju razdvojenih klasa. Prostor koji se nalazi između isprekidanih zelenih linija naziva se ploha razdvajanja. Podaci se većinom nalaze u konveksnim ljuskama koje grupiraju pojedinačne klase.



Sl. 3.3. *Mala i velika margina*

Klasifikator može biti linearni separabilan ili neseparabilan, gdje neseparabilan predstavlja situaciju kada nije moguće razdvojiti podatke u klase već dolazi do pojavljivanja određenog pojma u klasi kojoj on ne pripada. Ukoliko je problem koji se stavi pred klasifikator neseparabilan tada se želi postići što je moguće bolje razdvajanje podataka u klase koristeći standardnu proceduru koja koristi plohe razdvajanja. Sustav koji se temelji na linearnoj klasifikaciji kao pretpostavku koristi to da uzorci za treniranje koji pripadaju skupu S mogu biti idealno razdijeljeni hiperravninom što je prikazano na slici 3.4. prema [11, str. 64]. To su primjeri vrlo jednostavnih ili ponekad i binarnih klasifikacija gdje je moguće napraviti podjelu na dvije klase. Čest je slučaj da u stvarnom životu nemamo samo dvije klase te je tada ovakav klasifikator neučinkovit prilikom rješavanja postavljenog problema.

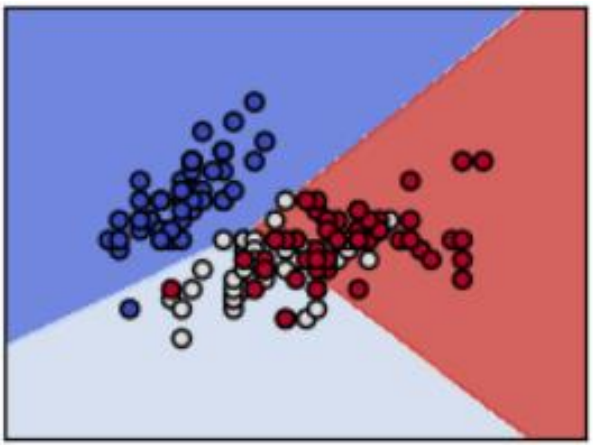


Sl. 3.4. *Razdvajanje klasa hiperravninom*

Jednadžba (3-4) prema [11, str. 64] opisuje definiciju linearnog klasifikatora gdje H predstavlja hipotezu funkcije koja mapira funkcije koje izvršavaju klasifikaciju. Hipoteza forme $x \rightarrow sign(w \cdot x + b)$ omogućava označavanje jedne klase podataka te ih stavlja na jednu stranu hiperravnine $w \cdot x + b$ dok ostale tada prelaze na suprotnu stranu.

$$H = \{x \rightarrow sign(w \cdot x + b) : w \epsilon \mathbb{R}^N, b \epsilon \mathbb{R}\} \tag{3-4}$$

Za potrebe ovog rada potrebno je koristiti klasifikator koji će uspješno izvršavati klasifikaciju gdje je potrebno vršiti klasifikaciju na osnovu više određenih klasa. Linearni klasifikator ima mogućnost obrade više klasa gdje se pokušava razdvojiti klase. Slika 3.5. prema [10] prikazuje primjer višeklasne podjele podataka koristeći linearni klasifikator koji koristi metodu potpornih vektora. Prednost ove metode je što koristi potporne vektore koji zauzimaju relativno malo memorijskog prostora pa su zbog toga pogodni za velike količine podataka.



Sl. 3.5. *Višeklasna podjela podataka*

4. OBRADA PODATAKA

Jedan od važnih koraka prilikom izgradnje klasifikatora korisničkih upita je predobradba dostupnih podataka na temelju kojih se izgrađuje klasifikator. Podatke, koje su dohvaćeni s Interneta, potrebno je predobraditi kako bi bili pogodni za računalnu obradu. Predobrada obuhvaća normalizaciju, uklanjanje interpunkcijskih znakova, rastavljanje rečenica na riječi te pronalazak korijena riječi.

4.1. Klasifikacija korisničkih upita

U svijetu koji se sve više oslanja na komunikaciju putem interneta vrlo je važno da pružatelji telekomunikacijskih usluga svojim korisnicima omoguće stabilan pristup internetu bez čestih prekida i uz to stabilnu telefonsku liniju. Izgrađeni klasifikator se može upotrijebiti za automatsku klasifikaciju korisničkih upita te se na taj način može ubrzati cijeli postupak podrške od strane operatera. Da bi izgradio klasifikator, potrebno je najprije imati podatkovne primjere, tj. korisničke upite koji su labelirani tj. svakom dohvaćenom korisničkom upitu pridružena je klasa. Klasifikacija dohvaćenih korisničkih upita obavljena je ručno na način da se svakom korisničkom upitu dodijelila klasa na temelju sadržaja upita. Analizom dohvaćenih podatka zaključilo se da su upiti uglavnom vezani uz probleme s ugovornom obvezom pretplatnika, probleme s fiksnim uslugama općenito, internet, televizijom, mobitelom i računima za određene usluge. Osim toga, neki od upita predstavljaju kritike ili pohvale na račun operatera. U konačnici definirano je deset klasa pri čemu je jedna klasa "Nepoznato" kada se upit na temelju teksta ne može klasificirati ni u jednu od navedenih klasa. Rezultat ove obrade spremljen je u CSV datoteku. Klase u koje su podijeljeni korisnički upiti su sljedeće prikazani su u tablici 4.1. sa danim opisom pojedine klase.

Tab. 4.1. Klase korisničkih upita.

Klasa	**Opis**
Ugovor	Korisnik ima primjedbu ili pitanje vezano za svoj pretplatnički ugovor na uslugu.
Fiksne usluge	Postoji problem sa telefonskom linijom ili je upit vezan uz neku od usluga vezanu uz telefon.
Internet	Korisnik nema internetskog pristupa, problem sa propusnošću veze i sl.
Televizija	Problemi sa IPTV uređajem, krivim prikazom slike na televizoru ili želja za aktivacijom dodatnih paketa.
Mobitel	Poteškoće vezane uz mobilne usluge i mobilni Internet.
Račun	Upiti vezani uz probleme sa računima koji ili su preveliki ili ne stižu na adresu korisnika.

Pohvala	Korisnici ostave pružatelju usluga pozitivnu povratnu informaciju gdje izražavaju zadovoljstvo sa uslugom ili otklonjenom poteškoćom u kratko vrijeme.
Kritika	Korisnik ostavlja kritiku zbog nezadovoljstva uslugom ili dugim rokom za otklanjanje poteškoće.
Infrastruktura	Problemi vezani uz infrastrukturu koju održava pružatelj usluga.
Nepoznato	Upiti za koje nije moguće odrediti klasu.

Konkretan primjer ručne klasifikacije upita prikazan je na slici 4.1. gdje su upiti prikupljeni na društvenim mrežama, točnije na profilu Hrvatskog Telekoma. Ručno klasificiranih upita ima ukupno 1462, te je u tablici 4.2. prikazano koliki je broj upita pridružen jednoj od klasa.

Tab. 4.2. Broj upita po klasama

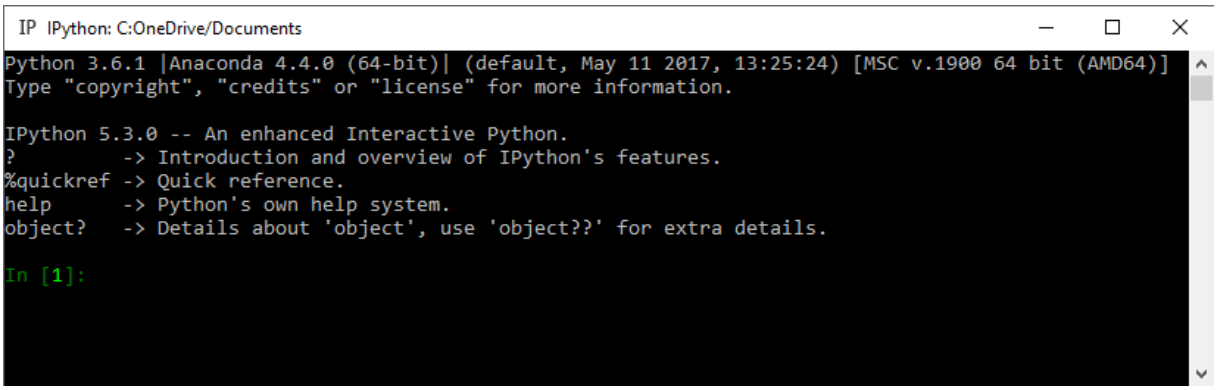
Klasa	**Broj upita**
Mobitel	354
Kritika	236
Internet	192
Nepoznato	176
Televizija	129
Fiksne usluge	117
Račun	86
Infrastruktura	79
Ugovor	54
Pohvala	39

pageName	userName	message	classification
Hrvatski Telekom	Krunoslav Kovač	Kako da ja dodem do vašeg tehničara? Evo dolazi već 3 dana, ali nikako da stigne..	Nepoznato
Hrvatski Telekom	Lidija Nikolić	ima veze sa zdravim razumom, kako nešto što biste trebali obaviti za 10-15 dana se može	Kritika
Hrvatski Telekom	Til Da	Zanima me zašto mi ne želite izdati valjani račun? Naime, raskinula sam ugovor s HT-om prije vi	Račun
Hrvatski Telekom	Mirabel Jaska	pozdrav može li objašnjenje zašto još nije isporučen novi max tv uređaj iako je trebao doći prij	Televizija
Hrvatski Telekom	Josip Perić	Zašto od 7 kanala SportKluba vi imate 6, a onaj jedan niste uvrstili u ponudu? Možete nama poj	Televizija
Hrvatski Telekom	Dino Balaško	Sramota. Dali je ovo internet u 21st.? Hmmmmm pa toga nema ni u šumi. Da ne govorim o prob	Internet
Hrvatski Telekom	Goran William Rose	Pozdrav, da li postoji način da se na mobitelu blokiraju svi privatni dolazni pozivi?	Mobitel
Hrvatski Telekom	Tatjana Ana Zarijević	Poštovani, upravo sam maloprije opet i ponovo kontaktirala vašu korisničku službu. Naime, u s	Fiksne usluge
Hrvatski Telekom	Ira Bulic	Poštovani, Od nedavno (petka 23.6.) koristim Vaše usluge. Korisnik sam postala tako da mi je m	Internet
Hrvatski Telekom	Jasminka Kolić	Postovani ...Zadnje vrijeme nisam zadovoljna vasim uslugama , sve cesce ostanemo bez interne	Račun
Hrvatski Telekom	Ksenija Gortan	Kada možemo očekivati popravak kvara? Bez interneta, MaxTV-a i telefona kao da sam na pust	Fiksne usluge
Hrvatski Telekom	Ana Pejaković	Zanima me dali su vaši službenici glupi, gluhi ili ne razumiju hrvatski? na 6 puta ponovljeno pit	Ugovor
Hrvatski Telekom	Hrvoje Plus	Ako uzmem maxtv to go uslugu i nemam vise na racunu, istekne mi i internet paket da li u tom	Televizija
Hrvatski Telekom	Milena Rakin	Šaljem pritužbu Povjerenstvu za pritužbe u drugostupanjskom postupku putem Odsjeka za upr	Nepoznato

Sl. 4.1. *Ručno klasificirani korisnički upiti*

4.2. Python biblioteke

Programski jezik Python postao je glavni alat za razvoj aplikacija koje koriste strojno učenje. A.C., Muller i S., Guido u svojoj knjizi navode: „On kombinira snagu programskih jezika opće namjene i jednostavnost uporabe koju imaju skriptni programski jezici specifične domene kao što su MATLB i R.“ [1]. Zbog velikog broja biblioteka koje olakšavaju obradu podataka i izgradnju modela metodama strojnog učenja sve više programera koristi ovaj programski jezik kako bi jednostavno razvili ponekad složene aplikacije u vrlo kratkom vremenu. Sve potrebne biblioteke za izradu klasifikatora korisničkih upita sadržane su u Anaconda distribuciji Python-a koja sadrži biblioteke kao što su NumPy, SciPy, pandas i druge. Kroz svoj interaktivni *interpreter* koji je prikazan na slici 4.2. ovaj programski jezik omogućuje direktno upisivanje liniju po liniju koda i trenutno izvršavanje. Python se zbog svega navedenog pokazao kao idealan jezik za primjenu u ovom diplomskom radu.



Sl. 4.2. *IPython interpreter*

4.2.1. Pandas

Pandas je biblioteka koja služi za lakšu manipulaciju podacima koje se dohvaćaju iz baza podataka, CSV datoteka, Excel datoteka i svih datoteka koje imaju tabličnu strukturu. Kako bi se lakše obrađivali takvi podaci koriste se *dataframe* strukture podataka koje su dio pandas biblioteke. Kako W., McKinney navodi u svojoj knjizi: „Pandas omogućava sofisticirane mogućnosti indeksiranja kako bi preoblikovanje, rezanje i oblikovanje učinio što lakšim i omogućio sakupljanje i dohvaćanje pod skupova podataka.“ [14]. Za potrebe ovog rada u pandas *dataframe* se učitala CSV datoteka ručno klasificiranih korisničkih upita i kroz ostale postupke sva se manipulacija sa podacima radila kroz *dataframe* koji je prikazan na slici 4.3. iz kojeg je vidljiva struktura podataka koja je uvelike slična onoj koja je bila u CSV datoteci što omogućava jednostavnu manipulaciju i obradu.

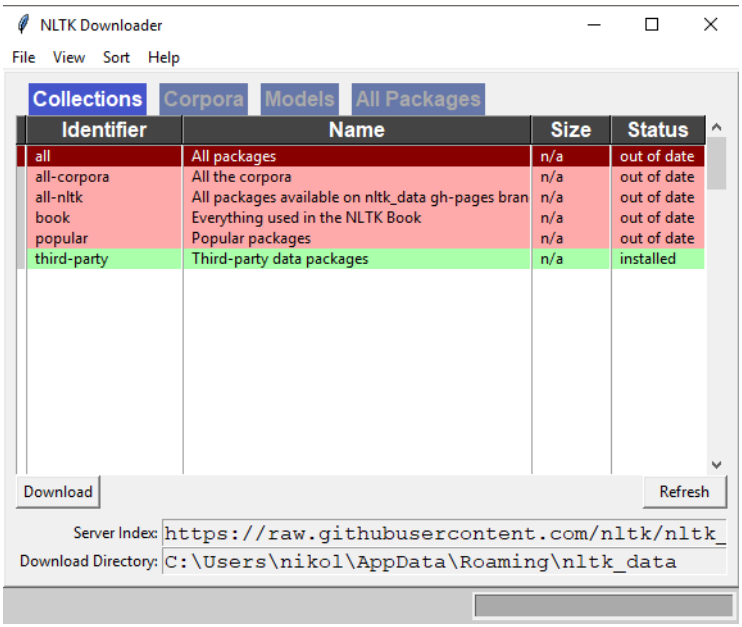
	message	classification
0	Kako da ja dodem do vašeg tehničara? Evo dolaz...	Nepoznato
1	Poštovani, može li mi itko objasniti,ovdje poš...	Kritika
2	Zanima me zašto mi ne želite izdati valjani ra...	Račun
3	pozdrav može li objašnjenje zašto još nije isp...	Televizija
4	Zašto od 7 kanala SportKluba vi imate 6, a ona...	Televizija
5	Sramota. Dali je ovo internet u 21st.? Hmmmmm ...	Internet
6	Pozdrav, da li postoji način da se na mobitelu...	Mobitel
7	Poštovani, upravo sam maloprije opet i ponovo ...	Fiksne usluge
8	Poštovani, Od nedavno (petka 23.6.) koristim V...	Internet
9	Postovani ...Zadnje vrijeme nisam zadovoljna v...	Račun
10	Kada možemo očekivati popravak kvara? Bez inte...	Fiksne usluge
11	Zanima me dali su vaši službenici glupi, gluhi...	Ugovor
12	Ako uzmem maxtv to go uslugu i nemam vise na r...	Televizija
13	Šaljem pritužbu Povjerenstvu za pritužbe u dru...	Nepoznato
14	Poštovani, javljam Vam se zbog jednog detalja ...	Infrastruktura
15	Da krenem s poštovani ne mogu,vas ne poštujem....	Kritika
16	Zašto se kod vas svi operateri zovu Borna? I z...	Kritika
17	crko internet. opet. složite mi je*eni interne...	Internet
18	opet fejs miče postove?????	Nepoznato
19	Samo bih molio da se presluša razgovor početko...	Nepoznato
20	sad ovu vidim ali onu di sam napisao da mi na ...	Nepoznato
21	Di je moja objava, zašto je ne vidim? ostale v...	Nepoznato
22	7miseci cekam paricu,dali je to normalno u 21s...	Infrastruktura
23	Mi mozete pomoc oko interneta? Vec jedo 2 sata...	Internet
24	Postovani, Posaljem Vam smrtni list Vase koris...	Nepoznato
25	Imam prigovor na rad ovlaštenog servisera (MOB...	Mobitel

Sl. 4.3. *Dataframe korisničkih upita*

4.2.2. NLTK

NLTK (engl. *Natural Language Toolkit)* je skup biblioteka unutar programskog jezika Python koje služe za obradu prirodnog jezika. Da bi se mogle koristiti pojedine značajke potrebno je preuzeti zbirke i modele u NLTK alatu za preuzimanje dodataka prikazanom na slici 4.4. koji se pokreće uz pomoć naredbi prikazanih niže.

```
1. import nltk
2. nltk.download()
```



Sl. 4.4. *NLTK alat za preuzimanje dodataka*

Za potrebe ovog diplomskog rada instalirana je zbirka *stopwords* koja će služiti za uklanjanje riječi koje nisu relevantne za analizu značenja korisničkog upita. Uz nju koriste se još i *word_tokenize* koji služi za rastavljanje rečenice u pojedine riječi te *stem* paket koji izvodi korijene riječi. Isti moduli i zbirke se koriste za obradu ručno klasificiranih upita i novih upita.

4.2.3. scikit-learn

Još jedan važan skup alata koji se mogu koristit u Python programskom jeziku je scikit-learn. Ovaj projekt otvorenog koda u sebi sadrži razne alate za izgradnju modela za strojno učenje i vizualizaciju rezultata. Anaconda distribucija već unutar sebe sadrži ovaj alat pa nisu potrebne dodatne instalacije unutar Pythona. Postoje razni modeli klasifikatora koje scikit-learn podržava, a neki od njih su:

- Strojevi sa potpornim vektorima
- Naivni Bayesov klasifikator
- Random forest.

Za potrebe izgradnje Naivnog Bayesovog klasifikatora i SVM klasifikatora korištene su funkcije iz ove biblioteke.

4.3. Obrada upita

Nakon ručne klasifikacije teksta potrebno je dodatno pregledati upite korisnika. Kako se često u pisanju događaju pravopisne greške ili izostavi dio slova u riječi potrebno je ispraviti što je više moguće takvih pogrešaka kako bi kasnije imali što "čišće" podatke za izgradnju klasifikatora. Rečenice koje su prešle u novi redak predstavljaju problem pri obradi skupa podataka te ih je također potrebno spojiti sa prethodnim rečenicama. Daljnja procedura uključuje postupke koji će normalizirati upite i svesti ih na oblik prilagođen za izgradnju modela klasifikatora. Upiti su obrađeni uz pomoć programskog jezika Python koristeći niz biblioteka iz NLTK-a.

4.3.1. Normalizacija i interpunkcijski znakovi

Prvi korak pri obradi upita je osnovna normalizacija teksta s kojom se uklanja pojava velikih slova na početku rečenica i općenito bilo gdje u rečenici. Algoritmi koji kreiraju model klasifikatora riječi kao što su *Postojanje* i *postojanje* smatraju kao dvije različite riječi zbog velikog početnog slova. Klasificirani upiti koji su učitani u *dataframe* normaliziraju se uz pomoć ugrađene funkcije *str.lower()* koja prolazi kroz svaki red kolone i zamjenjuje veliko

slovo u riječi s odgovarajućim malim slovom. Na sličan način se normalizira i korisnički upit kada se model koristi za klasifikaciju novih korisničkih upita gdje se koristi ugrađena funkcija *lower()* koja se primjenjuje na *string* tipu podatka, detalji se mogu vidjeti u prilogu P.4.1..

Interpunkcijski znakovi također predstavljaju podatak koji najčešće nije relevantan za izradu klasifikatora pa ih je potrebno ukloniti iz upita. Regularni izrazi su jednostavno i efikasno rješenje te za njih postoji funkcija koja se učitava uz pomoć modula za regularne jezike. Funkcija jednako uklanja znakove iz skupa korisničkih upita i iz pojedinačnog upita kojeg korisnik preda klasifikatoru i to na način da prođe kroz upit i svaki interpunkcijski znak zamjeni s praznim poljem te se detalji oko način izvedbe u kodu mogu vidjeti u prilogu P.4.1. Slika **4.5** prikazuje normalizirane upite bez interpunkcijskih znakova.

	message	classification
0	kako da ja dodem do vašeg tehničara evo dolazi...	nepoznato
1	poštovani može li mi itko objasnitiovdje pošto...	kritika
2	zanima me zašto mi ne želite izdati valjani ra...	račun
3	pozdrav može li objašnjenje zašto još nije isp...	televizija
4	zašto od 7 kanala sportkluba vi imate 6 a onaj...	televizija
5	sramota dali je ovo internet u 21st hmmmm pa ...	internet
6	pozdrav da li postoji način da se na mobitelu ...	mobitel
7	poštovani upravo sam maloprije opet i ponovo k...	fiksne usluge
8	poštovani od nedavno petka 236 koristim vaše u...	internet
9	postovani zadnje vrijeme nisam zadovoljna vasi...	račun
10	kada možemo očekivati popravak kvara bez inter...	fiksne usluge

Sl. 4.5 *Normalizirani upiti bez interpunkcija*

4.3.2. Stop riječi i tokenizacija

Nisu sve riječi jednako važne za značenje unutar rečenice. Riječi koje se često pojavljuju su na primjer biti, evo, ej, eno, gdje, kada i slične riječi. Zatim u istu skupinu spadaju veznici i negacije. Sve te riječi potrebno je ukloniti iz upita kako bi na kraju samo ostale riječi na temelju kojih je moguće donijeti zaključak o značenju upita. Za provedbu ovog postupka preuzet je rječnik hrvatskih stop riječi koji je učitan u *stopwords* korpus koji je dio NLTK modula unutar Pythona. Programski kod prikazan niže pokazuje na koji način je izveden postupak uklanjanja stop riječi skupa korisničkih upita u *dataframe* tipu podatka što je detaljno prikazano u prilogu P.4.1.

```
1. #Uklanjanje stop words-a
2. stop = stopwords.words('croatian')
3. df['message'] =
   df['message'].apply(lambda x: [item for item in x if item not in stop
   ])
```

Petlja prolazi kroz *dataframe* i uz pomoć lambda funkcije vrši izmjenu riječi koje se nalaze u *stopwords* rječniku s praznim prostorom u rečenici. Na jednak način se obrađuju i

upiti korisnika koji se klasificiraju. Pošto se radi o tipu podatka *string* koristi se *join()* funkcija koja također uz pomoć lambda funkcije filtrira sve riječi koje se nalaze u rječniku *stopwords* što je vidljivo u kodu koji slijedi, a kompletan kod može se pronaći u prilogu P.4.2.

```
1. stop = stopwords.words('croatian')
2.     user_input = " ".join(filter(lambda word: word not in stop,
   user_input.split()))
```

Tokenizacija je postupak razdvajanja rečenice u riječi te je to jedan od zadnjih koraka obrade upita. NLTK modul ima ugrađenu funkciju *word_tokenize()* s kojom se vrši rastavljanje rečenice u riječi. Kada se taj postupak radi na *dataframe-u* tada je potrebno napisati funkciju koja će prolaziti kroz *dataframe* i svaku rečenicu koju obradi spremiti u polje koje će kasnije proslijediti natrag u *dataframe* što je vidljivo u kodu koji slijedi i koji je detaljno prikazan u prilogu P.4.1.

```
1. #Tokenizacija - razbijanje teksta u riječi
2. def apwords(words):
3.     filtered_sentence = []
4.     words = word_tokenize(words)
5.     for w in words:
6.         filtered_sentence.append(w)
7.     return filtered_sentence
8. addwords = lambda x: apwords(x)
9. df['message'] = df['message'].apply(addwords)
```

Kod obrade korisničkog upita koji se preda u klasifikator kako bi se dobio odgovor, programski kod je nešto jednostavniji te se koristi ugrađena funkcija *split()* koja rastavi rečenicu upita na riječi što je detaljno vidljivo u prilogu P.4.2. Slika 4.6. prikazuje upite iz kojih su uklonjene stop riječi i izvršena tokenizacija.

```
user_input = user_input.split()
```

	message	classification
0	[dodem, vašeg, tehničara, evo, dolazi, 3, dana...	nepoznato
1	[poštovani, može, itko, objasnitiovdje, pošto,...	kritika
2	[zanima, zašto, želite, izdati, valjani, račun...	račun
3	[pozdrav, može, objašnjenje, zašto, isporučen,...	televizija
4	[zašto, 7, kanala, sportkluba, imate, 6, onaj,...	televizija
5	[sramota, dali, ovo, internet, 21st, hmmmm, t...	internet
6	[pozdrav, postoji, način, mobitelu, blokiraju,...	mobitel
7	[poštovani, upravo, maloprije, opet, ponovo, k...	fiksne usluge
8	[poštovani, nedavno, petka, 236, koristim, usl...	internet
9	[postovani, zadnje, vrijeme, zadovoljna, vasim...	račun
10	[možemo, očekivati, popravak, kvara, bez, inte...	fiksne usluge

Sl. 4.6. *Uklonjene stop riječi i izvršena tokenizacija*

4.4. Morfološka normalizacija

Složenost obrade jezika u sustavima koji koriste strojno učenje uvelike ovisi o tome kolika je morfološka složenost sustava. Kako navodi A., Pirkola u svome radu: „Morfološka složenost jezika određena je učestalošću afiksacije u jeziku te složenošću postupka segmentacije afiksa u pojedinim oblicima riječi.“ [16]. Fleksija nastaje tako da se dodaju gramatički morfemi osnovi riječi. Afiksi se nadovezuju na osnovu riječi te su dio fleksije prilikom stvaranja nove riječi. Kako navodi J., Šnajder u svome radu: „Primjer fleksije u hrvatskom jeziku je kuća-kućom ili bogat-najbogatiji. Predmet poučavanja derivacijske morfologije jest tvorba riječi iz postojećih uporabom derivacijskih afiksa.“ [15]. Morfološke varijacije predstavljaju problem kod klasifikacije teksta jer se jedan pojam može pojaviti u raznim oblicima koji će u sustavu predstavljati različite pojmove iako bi oni za potrebe klasifikacije imali isto značenje.

U svrhu izbjegavanja navedenih problema potrebno je povesti morfološku normalizaciju gdje postoje dva osnovna pristupa, a to su korjenovanje i lematizacija. Korjenovanje je proces gdje se uklanja afiks iz riječi kako bi se dobio njen korijen koji je zajednički za sve morfološke varijante te riječi. Neke od vrsta korjenovanja su korjenovanje temeljeno na pravilima, hibridno korjenovanje gdje uz standardno odsijecanje se koristi i rječnik koji provjerava i sprječava da se normaliziraju slične riječi u istu riječ. Lematizacija je proces s kojim se pronalazi ispravan kanonski oblik riječi. To ponekad nije jednostavan proces zbog razine složenosti pojedinog jezika te je potreban veliki napor kako bi se stvorio lingvistički leksikon. Kako navodi J., Šnajder: „Rječnička normalizacija teoretski nudi apsolutnu lingvističku točnost, no izgradnja lingvističkog leksikona iziskuje veliko lingvističko znanje i ogroman ljudski napor.“ [15].

4.4.1. Korjenovanje

Postupak korjenovanja (engl. *stemming*) u ovom radu izveden je koristeći korjenovanje temeljeno na pravilima. Ovaj jednostavan algoritam pronalazi korijen riječi tako da ukloni prefikse i sufikse iz riječi uz pomoć ručno kodiranih pravila. Kako hrvatski jezik nije jednostavan za provedbu korjenovanja zbog morfološke složenosti jezika u kojem ima dosta glasovnih promjena, stvaranje algoritma za korjenovanje u tom slučaju može biti zahtjevan proces. Postoje razni algoritmi za korjenovanje razvijeni za engleski jezik kao što su Lovinsov algoritam, Porterov algoritam.

Za hrvatski jezik postoji algoritam za korjenovanje razvijen na Filozofskom fakultetu Sveučilišta u Zagrebu od strane grupe za obradu prirodnog jezika odjela za informacijske znanosti. Prema [18] algoritam za korjenovanje provodi niz transformacija koje su definirane i koje provode i nadziru morfološke promjene, te nakon tog koristi i niz definiranih pravila koji uklanjaju sufikse i tako izvode korjenovanje. Sličan postupak se provodi i za korisničke upite s kojima se gradio klasifikator i koji se nalaze unutar *dataframe*-a, te i za pojedine upite koje korisnik proslijedi klasifikatoru. Postupak za oba slučaja je vidljiv u kodovima koji slijede i koji su detaljno opisani u prilozima P.4.1., P.4.2.

```
1. #Stemming - pretvaranje rijeci u njihove korijene
2. df['message']=df['message'].apply(lambda x
   : [stemmer.stem(y) for y in x])

   user_input = [[stemmer.stem(word) for word in sentence.split("
   ")] for sentence in user_input]
```

Obrada upita u *dataframe*-u se odvija tako da se uz pomoć lambda funkcije izvršava funkcija *stem* objekta *stemmer* koja je vidljiva u prilogu P.4.2. i koja prolazi kroz upite koristeći *for* petlju i obrađuje riječ po riječ te ih takve sprema nazad u *dataframe*. Kada korisnik želi predati klasifikatoru upit tada se taj postupak djelomično razlikuje jer se više ne koristi lambda funkcija nego se s *for* petljom prolazi kroz rečenicu i analizira se svaka riječ te se predaje funkciji *stem* koja provodi postupak i zapisuje rezultat u varijablu. Završeni postupak obrade upita prikazan je na slici 4.7. i takvi upiti su sada spremni za korištenje u okviru postupka učenja određenog tipa klasifikatora što je opisano u narednom poglavlju.

	message	classification
0	[dod, vašeg, tehničar, ev, dolaz, 3, dan, nika...	nepoznato
1	[poštovan, može, itk, objasnitiovdj, pošt, nig...	kritika
2	[zan, zašt, želite, izda, valjan, račun, naim,...	račun
3	[pozdrav, može, objašnjenj, zašt, isporučen, n...	televizija
4	[zašt, 7, kanal, sportklub, imat, 6, ona, jeda...	televizija
5	[sramot, dal, ov, internet, 21st, hmmmm, tog,...	internet
6	[pozdrav, postoj, način, mobitel, blokiraj, pr...	mobitel
7	[poštovan, uprav, maloprij, opet, ponov, konta...	fiksne usluge
8	[poštovan, nedavn, petk, 236, korist, uslug, k...	internet
9	[postovan, zadnj, vrijem, zadovoljn, vasi, usl...	račun
10	[možemo, očekiva, popravak, kvar, bez, interne...	fiksne usluge

Sl. 4.7. *Korisnički upiti nakon potpune obrade*

5. IZGRADNJA I IMPLEMENTACIJA KLASIFIKATORA

Glavni izazov ovog diplomskog rada je izgraditi klasifikator koji će imati što veću preciznost prilikom klasifikacije korisničkih upita. Ručna klasifikacija i obrada upita koja je obrađena u prethodnim poglavljima rada u velikoj mjeri utječe na to koliko će klasifikator uspješno odrediti klasu nakon što primi novi upit od korisnika. Izgrađeni klasifikatori su Naivni Bayes i stroj sa potpornim vektorima koji će biti implementirani u web okruženje koje će omogućiti API pristup.

5.1. Naivni Bayes i SVM model klasifikatora

Implementacija klasifikatora u Python programskom jeziku je vrlo jednostavna uz korištenje scikit-learn skupa alata koji sadrži algoritme za klasifikaciju. Za izradu klasifikatora koristi se skup podataka od 1462 ručno klasificirana upita koji su podijeljeni na dio za trening koji je veličine 1100 upita, dok je dio za testiranje odnosno evaluaciju veličine 362 upita. Prvi korak u pripremi podataka za klasifikator je da se svi upiti koji se trenutno nalaze u *dataframe* tipu podatka pretvore u polje podataka kako bi se lakše mogla izvesti vektorizacija upita koja će biti objašnjena u daljnjem radu.

Da bi se ispravno provelo klasificiranje potrebno je dohvatiti značajke (engl. *features*) koje su najčešće riječi koje se pojavljuju u ručno klasificiranim upitima. Nakon toga se stvara rječnik koji se dobiva iz popisa značajki s kojim će se moći vršiti prebrojavanje riječi u određenom korisnikovom upitu. Cijeli taj postupak je priprema za uvrštavanje značajki u model skupa riječi (engl. *bag of words*). Broj značajki ne smije biti prevelik jer će se tada pojaviti riječi koje nisu relevantne za klasifikaciju, pa će s time opasti efikasnost klasifikatora. Odabrani broj značajki u ovom diplomskom radu je 1100 riječi. Prikazan je programski kod koji sadržava funkcije za izradu Naivnog Bayesovog klasifikatora i klasifikatora koji koristi metodu potpornih vektora koji je detaljno prikazan u prilogu P.4.1.

```
1. #Naive Bayes klasifikator
2. classifier_nb = MultinomialNB(alpha = 1.8, fit_prior = False)
3. classifier_nb.fit(train_data_features, targets)
4.
5. #SVM klasifikator
6. classifier_svm = svm.LinearSVC(multi_class = 'crammer_singer')
7. classifier_svm.fit(train_data_features, targets)
```

Korišten je *Multinominal* klasifikator jer se razmatra problem višeklasne klasifikacije, te se on općenito koristi u klasifikaciji teksta. *Alpha* parametar prema [20] predstavlja parametar

uglađivanja koji je ostavljen na vrijednost 1.8 radi boljeg rezultata klasificiranja. *Crammer_singer* parametar SVM klasifikatora predstavlja vrstu višeklasne strategije klasifikacije. Proces klasificiranja novog upita kojeg korisnik pošalje klasifikatoru sastoji se od toga da se upit koji korisnik pošalje proslijedi funkciji *classification()* koja obavi sav postupak opisan prethodno u radu i kao rezultat vrati klasu kojoj pripada taj pojam što je prikazano u prilogu P.5.1.

5.1.1. Skup riječi

Vrlo jednostavan model koji se koristi za klasifikaciju teksta je skup riječi. Prema [1] bitni koraci u stvaranju skupa riječi su:

- Tokenizacija – razbijanje rečenica u skup riječi.
- Stvaranje rječnika – sakupljanje svih značajki skupa s kojima će se kodirati pojedini upit.
- Kodiranje – za svaki upit se prebrojava koliko puta se pojavila određena riječ iz rječnika.

Kada se prođe kroz sve korake dobiva se polje koje se sastoji od brojeva koji predstavljaju broj pojavljivanja riječi u rečenici. Neka je rečenica 1: Imaš li pojam o čemu se tu radi?, a neka je rečenica 2: Sam pojam o tome ti je jako stran. Iz obje rečenice možemo izuzeti riječi koje će predstavljati rječnik, pa će on izgledati ovako [imaš, pojam, čemu, radi, sam, tome, jako, stran]. Kada rečenice rastavimo u riječi i prebrojimo pojavljivanje riječi iz rječnika u pojedinoj rečenici tada dobijemo sljedeća polja prikazana u tablici 5.1.

Tab. 5.1. Primjer skupa riječi

Rječnik	Imaš	Pojam	Čemu	Radi	Sam	Tome	Jako	Stran
Rečenica 1	1	0	1	1	0	0	0	0
Rečenica 2	0	1	0	0	1	1	1	1

Iz primjera je vidljivo kako se neke riječi pojavljuju u obje rečenice, dok nekih ima samo u jednoj od njih. Na većim skupovima podataka veličina rječnika raste i do nekoliko tisuća pojmova. Ovaj jednostavan model pokazao se vrlo efikasnim u ovom diplomskom radu gdje je s njegovom primjenom uspješno izvršena klasifikacija.

5.1.2. Vektorizacija

Vektorizacija je ugrađena funkcija unutar scikit-learn skupa alata. Ona je način da se skup riječi izvede u programskom jeziku Python. Kada se upit spremi u polje tada se on predaje funkciji *fit_transform* koja je vezana uz prethodno kreiran objekt *vectorizer* gdje je definirano koja je vrsta funkcije koja će raditi obradu, te koliko će se značajki uzeti iz skupa svih riječi. Zatim se stvara skup značajki *train_data_features* koji se kasnije predaje klasifikatoru. Uz sve to stvara se i rječnik koji će služiti prilikom klasifikacije. Kod niže prikazuje izvedbu vektorizatora u programskom jeziku Pyhton koji je detaljno prikazan u prilogu P.4.1.

```
1. #Kreiranje vektorizer-a
2. vectorizer = CountVectorizer(tokenizer=lambda doc: doc,
   lowercase=False, max_features = 1100)
3.
4. train_data_features = vectorizer.fit_transform(train_inqury)
5.
6. train_data_features = train_data_features.toarray()
7.
8. vocab = vectorizer.get_feature_names()
9.
10.   dist = np.sum(train_data_features, axis=0)
```

5.2. Evaluacija modela

Nakon izvršene klasifikacije svih upita koji se nalaze u skupu za testiranje može se izvesti zaključak koliko je postupak pripreme podataka i izgradnje klasifikatora bio dobar. Dan je primjer korištenja izgrađenog klasifikatora prikazan na slici 5.1. te su u ovom slučaju oba klasifikatora dala točan odgovor, odnosno klase dobivene klasifikacijom podudaraju se sa klasama dodijeljenim tijekom ručne klasifikacije.

```
Unesite vaš upit ili upišite kraj za izlazak: Internet jako sporo radi, ne mogu otvoriti ni jednu stranicu,
kada će te to riješiti?
[['internet', 'jak', 'spor', 'rad', 'mogu', 'otvori', 'jedn', 'stranic', 'riješit']]
Klasa iz Naive Bayes klasifikatora je:  ['internet']
Klasa iz SVM klasifikatora je:  ['internet']
```

Sl. 5.1. *Rezultat klasifikacije upita*

Evaluacija klasifikatora provedena je na testnom skupu na način da su uspoređeni izlazi klasifikatora za svaki korisnički upit iz testnog skupa podataka sa stvarnom klasom upita. Postotak pogođenih korisničkih upita za Naivni Bayes je 60,94 %, dok je za SVM 58,72 %. Klasifikatori su vrlo slični u postotku ispravno klasificiranih upita što potvrđuje da su oba jednako dobri u klasifikaciji teksta.

Kada postoji više od dvije klase u postupku klasifikacije dobro je izvesti evaluaciju modela koristeći matrice zabune (engl. *confusion matrix*). Prema [3, str. 240] matrica zabune je tablica u kojoj svaka ćelija [*i*,*j*] sadržava broj koliko je puta oznaka klase *j* bila jednaka oznaci klase *i* koja je sadržana u ručno klasificiranim upitima testnog skupa podataka. Dijagonala matrice predstavlja upite koji su ispravno klasificirani, dok svi ostali elementi matrice predstavljaju grešku u klasifikaciji upita. Za Naivni Bayesov klasifikator matrica zabune prikazana je na slici 5.2., dok slika 5.3. prikazuje matricu zabune za SVM klasifikator i iz kojih je vidljivo koliki je broj upita ispravno klasificiran.

	pohvala	kritika	internet	televizija	mobitel	ugovor	račun	fiksne usluge	nepoznato	infrastruktura
pohvala	4	1	1	0	0	1	0	0	0	0
kritika	2	37	3	1	4	1	3	6	3	1
internet	1	1	32	2	3	0	0	11	2	0
televizija	0	0	0	29	3	0	0	7	0	0
mobitel	0	4	8	1	65	1	0	3	2	1
ugovor	0	0	0	0	1	11	3	3	0	0
račun	0	0	0	0	0	2	11	10	1	0
fiksne usluge	0	2	1	0	0	0	1	10	0	0
nepoznato	0	10	1	0	7	2	1	5	12	2
infrastruktura	1	1	4	0	5	0	0	1	0	9

Sl. 5.2. *Matrica zabune Naivnog Bayesovog klasifiktora*

	pohvala	kritika	internet	televizija	mobitel	ugovor	račun	fiksne usluge	nepoznato	infrastruktura
pohvala	5	0	1	0	0	0	0	0	1	0
kritika	1	29	1	2	7	1	0	5	14	1
internet	0	2	35	3	5	0	1	4	0	2
televizija	0	2	0	24	1	0	2	4	3	3
mobitel	0	1	9	2	61	0	2	3	6	1
ugovor	0	4	0	1	2	7	2	1	1	0
račun	1	2	2	0	1	1	8	5	3	1
fiksne usluge	0	2	1	0	1	0	1	7	0	2
nepoznato	0	3	0	1	4	0	0	2	30	0
infrastruktura	0	4	2	1	5	0	0	1	2	6

Sl. 5.3. *Matrica zabune SVM klasifikatora*

Iz prikaza matrica se vidi kako je većina klasificiranih upita smještena na dijagonali matrice što predstavlja ispravno klasificirane upite. Ponešto upita nalazi se izvan dijagonale i to su pogreške prilikom klasifikacije kojih u ovom slučaju nema previše. Primjećuje se kako se često dogodi da se klasa kritika zamijeni sa klasom nepoznato što se događa zbog pomalo sarkastičnog način pisanja kritika od strane korisnika.

5.3. Implementacija klasifikatora u Web okruženje

Izgrađeni klasifikator svoju primjenu može pronaći u raznim aplikacijama pa je odlučeno da se za potrebe demonstracije izradi web aplikacija koja koristi izrađeni API. Web aplikacija pristupa bazi podataka u koju se spremaju pristigli upiti koje korisnik preda putem obrasca.

Svaki uneseni upit se klasificira pomoću izgrađenih modela te se korisniku ujedno i prikaže rezultat klasifikacije. Tehnologije koje se koriste su Foundation *frontend framework* uz pomoć kojeg se kreira korisničko sučelje aplikacije, zatim Python Anywhere koji omogućava pokretanje koda i izradu web aplikacije u oblaku, a u ovom slučaju služi za implementaciju izgrađenog klasifikatora.. Pristup funkcionalnostima aplikacije omogućen je implementacijom API-ja uz pomoć Flask *microframeworka* putem kojeg web aplikacija komunicira sa Python skriptom.

5.3.1. Flask Microframework

Flask je *microframework* koji omogućava izgradnju web aplikacija koje unutar sebe mogu sadržavati Python kod. *Microframework* označava da Flask ne ovisi o vanjskim bibliotekama. Instalacija Flask-a jednaka je kao i kod ostalih Python modula. U ovom radu korišten je za izradu API pristupa aplikaciji te se uz pomoć njega izgradila web aplikacija koja je komunicirala sa Python skriptom. Kako bi ispravno pokretao stranice, Flask ima određena pravila kako rasporediti datoteke koje sadrže programski kod. Datoteke kojima korisnik treba pristupiti kada posjeti web aplikaciju smještaju se u *static* mapu, pa se tako u ovom radu u toj mapi nalaze Javascript i jQuery kodovi koji omogućavaju dinamičko osvježavanje sadržaja na stranici, CSS (engl. *Cascading Style Sheets*) datoteke, te razne funkcionalnosti Foundation *frameworka*. Mapa template sadrži sve HTML (engl. *HyperText Markup Language*) datoteke, što je vidljivo u prilogu P.5.2., te je ona zadužena za prikaz stranice. Slika 5.4. prikazuje općenitu strukturu Flask datoteka.

```
$ tree hello_flask/
hello_flask/
|-- static
`-- templates
```

Sl. 5.4. *Struktura Flask datoteka*

Struktura koda drugačija je nego kod klasične izrade web aplikacije. HTML kod se poziva unutar Python skripte, te je za ispravno prikazivanje potrebno koristiti *template* Flask dodatak. Ono što taj dodatak još omogućava je da ne postoji potreba za mijenjanjem kompletnog koda kada je potrebno stvoriti novu stranicu sa nešto drugačijim izgledom nego da se izmjene samo elementi od interesa, a ostali se zadrže od prethodno izrađenih stranica. Osnovni kod za prikaz stranice prikazan je niže. Iz njega se vidi da svaka stranica mora sadržavati svoju rutu koja govori kojoj stranici se pristupa kada se izvodi kod. Nakon rute slijedi funkcija koja se izvodi

i veže uz stranicu pa se unutar funkcije izvodi Python kod koji na kraju ima povratnu vrijednost.

```
1. from flask import Flask
2. app = Flask(__name__)
3.
4. @app.route("/")
5. def hello():
6.     return "Hello World!"
```

Implementacija ovog rada izvedena je uz pomoć dvije rute i funkcije gdje prva *make_predict()* funkcija prima korisnički upit, normalizira i provodi klasifikaciju te kao povratnu vrijednost vraća upit i klase iz oba klasifikatora u JSON (engl. *JavaScript Object Notation*) obliku što se detaljno može vidjeti u prilogu P.5.2. JSON je dio Javascripta koji omogućava jednostavan način razmjene podataka između sustava. Zbog svoje jednostavne sintakse vrlo je jednostavno dobiti potreban podatak iz pristiglog zahtjeva. Druga funkcija *homepage()* zadužena je za prikazivanje sučelje stranice na kojoj se unosi upit i za ispisivanje klase upita dobivene pomoću implementiranog klasifikatora. Povratna vrijednost ove funkcije je HTML stranica pohranjena u template mapu.

5.3.2. Implementacija API-ja

Pristup funkcionalnostima aplikacije izveden je uz pomoć API pristupa. Korisnik sa web stranice može uputiti zahtjev za klasifikacijom svog upita koji mora biti u JSON obliku. U ovom slučaju dovoljno je unutar zahtjeva poslati upit koji će sustav preuzeti, klasificirati ga i u JSON obliku vratiti dobivene klase pomoću implementiranog Naivnog Bayesa te SVM klasifikatora.. Zbog načina izvedbe web sustava uz upit će se predati ime i prezime osobe koja šalje upit i pružatelj usluga kojemu je upućen. Primanje i slanje zahtjeva odvija se uz pomoć *request* Flask modula koji može primati razne vrste zahtjeva i spremati njihove vrijednosti u varijablu. Za potpunu funkcionalnost potreban je i *jsonify* modul koji je također dio Flask-a i koji prima vrijednost uz pomoć funkcije *get_json*. Pokazni primjer pristupa prikazan je u kodu niže gdje se vidi da je potrebno navesti poveznicu ili putanju servera gdje se želi pristupiti koja je spremljena u *url* varijablu. Nakon što korisnik unese upit on se sprema u data varijablu u JSON obliku koji se zatim šalje POST vrstom zahtjeva kojem se predaje putanja do sustava i podatak koji se šalje.

```
1. import requests, json
2.
3. url = "http://localhost:5000"
4. while 1:
```

```
5.      user_input = input("Unesite vaš upit: ")
6.      if user_input == 1:
7.          break;
8.      data = json.dumps({'upit':user_input})
9.      r = requests.post(url, data)
10.        print (r.json())
```

Primjer poslanog zahtjeva koji sadržava upit i odgovor u JSON obliku prikazani su na slici 5.5. koja jasno prikazuje oblik odgovora koji odgovara formatu koji je i zahtjevan. Takav primljeni zahtjev lako je spremiti u bazu podataka, ispisati na web stranici ili predati nekom drugom sustavu koji će nastaviti obradu.

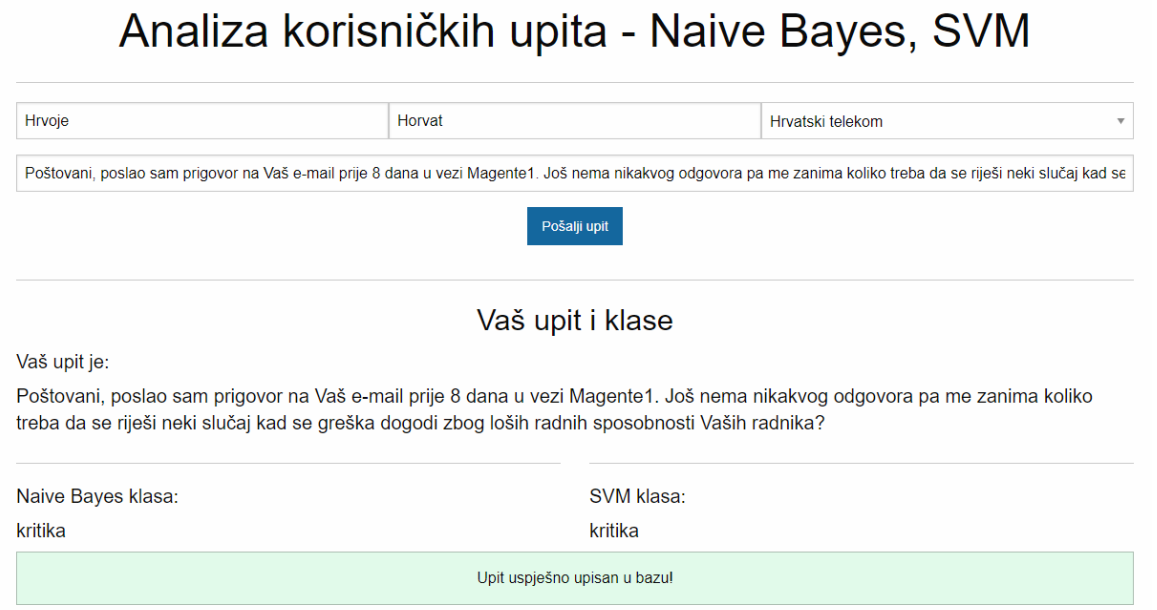
```
Unesite vaš upit: Poštovani, poslao sam prigovor na Vaš e-mail prije 8 dana u vezi Magente1. Još
nema nikakvog odgovora pa me zanima koliko treba da se riješi neki slučaj kad se greška dogodi
zbog loših radnih sposobnosti Vaših radnika?
{'results': ['kritika', 'kritika']}
```

Sl. 5.5. *Upit poslan API zahtjevom*

5.3.3. Python Anywhere

Konačna implementacija svih funkcionalnosti izvedena je u *online* sustavu Python Anywhere. Kroz ovaj sustav omogućeno je pokretanje Python skripti te pristup terminalu Linux operacijskog sustava putem kojeg se instaliraju svi potrebni moduli. Mogućnost izgradnje baze podataka kojoj će aplikacija pristupati također je dio ovog online sustava. Za ispravno izvođenje koda potrebno je instalirati sve potrebne module koji su detaljno navedeni u prilogu P.5.2., te koji su učitani u kod kako bi se mogle iskoristiti sve njihove funkcionalnosti.

Na slici 5.6. prikazan je izgled sučelja web aplikacije s kojom se upiti predaju klasifikatoru. Upit se zajedno sa imenom, prezimenom i operaterom kojem se upućuje upit šalje pozadinskoj skripti koja ga obrađuje i vraća zajedno sa klasama u JSON obliku koji se zatim dinamički prikazuje na istoj stranici na kojoj je i postavljen upit. Dinamičko prikazivanje sadržaja moguće je korištenjem AJAX-a koji omogućava asinkrono razmjenjivanje podataka sa serverom i njegovo dinamičko postavljanje na stranici bez potrebe da ju ponovno učitava te su detalji koda prikazani u prilogu P.5.3.



Sl. 5.6. *Prikaz sučelja web aplikacije*

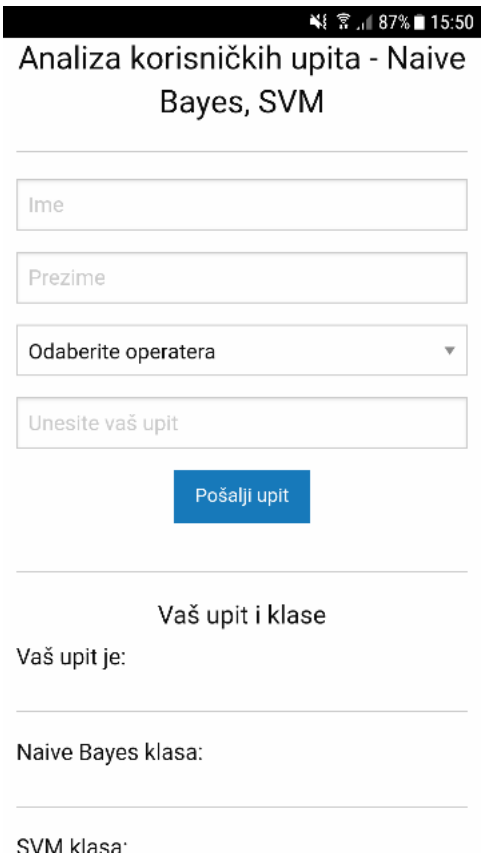
Upite je nakon klasifikacije potrebno pohraniti u bazu podataka. Sustav daje mogućnost izrade MySQL baza podataka sa željenim brojem tablica te se toj bazi može pristupiti iz web aplikacije. Za potrebe ovog diplomskog rad kreirana je jedna tablica baze podataka koja je prikazana u tablici 5.2. i u nju se upisuju svi klasificirani upiti kako bi se mogla vršiti daljnja analiza klasifikatora.

Tab. 5.2. Tablica upita u bazi podataka

Naziv	**Tip podatka**
id	INT
firstname	VARCHAR
lastname	VARCHAR
operater	VARCHAR
upit	TEXT
klasa_naive	VARCHAR
klasa_svm	VARCHAR

Web aplikacija može se posjetiti na poveznici http://nkomljenovic.pythonanywhere.com/ gdje je moguće predati novi upit po želji i isprobati klasifikaciju u stvarnom vremenu. Foundation *framework* omogućava izradu web aplikacija prilagođenih za mobilne uređaje, pa

je tako i ovoj aplikaciji moguće pristupiti putem mobilnog uređaja te je izgled aplikacije prikazan na slici 5.7. gdje se vidi da su svi elementi prilagođeni prikazu na mobilnom uređaju.



Sl. 5.7. *Izgled aplikacije u mobilnom pregledniku*

6. ZAKLJUČAK

Kroz izradu rada stečena su mnoga znanja na području strojnog učenja i njegove primjene u obradi prirodnog jezika. Ovo vrlo kompleksno područje iziskivalo je mnogo istraživanja na polju jezika, morfologije, obrade riječi i općenito svega vezanog uz obradu prirodnog jezika. Cilj ovog diplomskog rada je uz korištenje algoritama za obradu prirodnog jezika i klasifikaciju izraditi matematički model tj. klasifikator koji će biti u mogućnosti klasificirati korisnički upit u jednu od deset predefiniranih klasa. Za potrebe izgradnje klasifikatora dohvaćeni su korisnički upiti s društvenih mreža što je bilo otežano zbog zatvorenosti društvenih mreža koje ne dopuštaju dohvaćanje objava koje su korisnici ostavili na profilu pružatelja telekomunikacijskih usluga pa je bilo potrebno uz korištenje API pristupa koristiti i online sustave za analitiku društvenih mreža. Kroz obradu podataka postignuto je izvršavanje normalizacije korisničkog upita i njegovo čišćenje i rastavljanje na pojedine riječi što je važan korak u pripremi upita za klasifikaciju. Predobradom podataka načinjeni su skupovi za treniranje i testiranje klasifikatora. Uspješnom izgradnjom dvaju klasifikatora i njihovom implementacijom u *cloud* okruženje postignuto je to da se korisnički upit može klasificirati korištenjem jednostavnog API-a. Za potrebe demonstracije izrađena je web aplikacija za testiranje rada klasifikatora u stvarnom vremenu. Dobiveni rezultati zadovoljavajući su te modeli klasifikatora uspješno klasificiraju čak i zahtjevnije upite korisnika. Mjesta za napredak svakako ima te bi u daljnjem razvoju ovog sustava bilo dobro primijeniti kompleksnije algoritme za klasifikaciju, primijeniti i razviti bolje metode za obradu teksta i primijeniti naprednije metode označavanja podataka od one koja je primijenjena u ovom radu.

LITERATURA

[1] A.C., Muller, S., Guido, *Introduction to Machine Learning with Python*, O'Reilly Media, Inc., Sebastopol USA, 2016.

[2] J., Pustejovsky, A., Stubs, *Natural Language Annotation for Machine Learning*, O'Reilly Media, Inc., Sebastopol USA, 2012.

[3] S., Bird, E., Klein, E., Loper, *Natural Language Processing with Python*, O'Reilly Media, Inc., Sebastopol USA, 2009.

[4] *Company info: Stats*, Facebook Inc., USA, dostupno na: https://newsroom.fb.com/company-info/ [29.06.2017.]

[5] D., Roos, *How to Leverage an API for Conferencing: What is an API?*, How Stuff Works, Venice CA USA, 2007, dostupno na: http://money.howstuffworks.com/business-communications/how-to-leverage-an-api-for-conferencing1.htm [29.06.2017.]

[6] *REST APIs*, Twitter Inc., USA, dostupno na: https://dev.twitter.com/rest/public [29.06.2017.]

[7] P., Onesphory, *How to extract Twitter tweets data and followers to Excel*, Simplified Webscraping, Dubai UAE, 2017., dostupno na: https://nocodewebscraping.com/extract-twitter-tweets-followers-excel/ [29.06.2017.]

[8] M., Sugiyama, *Introduction to Statistical Machine Learning*, Elsevier Inc., Waltham USA, 2016.

[9] E., Alpaydin, *Introduction to Machine Learning*, The MIT Press, London England, 2004.

[10] J., Brownlee, *Supervised and Unsupervised Machine Learning Algorithms*, Machine Learning Mastery, Vermont Victoria Australia, 2016., dostupno na: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ [16.09.2017.]

[11] M., Mohri, A. Rostamizadeh, A., Talwalkar, *Foundations of Machine Learning*, The M.I.T. Press, Massachusetts USA, 2012.

[12] *Support Vector Machine: Classification*, Scikit-Learn developers, dostupno na: http://scikit-learn.org/stable/modules/svm.html#svm-classification [27.08.2017.]

[13] R., Basili, *Learning to classify text using Support Vector Machines: Methods, Theory, and Algorithms*, Computational Linguistics, br. 4, sv. 29, str. 656-664, Siječanj 2003.

[14] W., McKinney, *Python for Data Analysis*, O'Reilly Media, Inc., Sebastopol USA, 2012.

[15] J., Šnajder, *Postupci morfološke normalizacije u pretraživanju informacija i klasifikaciji teksta*, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, Zagreb Hrvatska, 2008.

[16] A., Pirkola, *Morphological typology of languages for IR*, Journal of Documentation,, no. 3, sv. 57, pp. 330–348, 2001.

[17] N., Ljubešić, D., Boras, O., *Kubelka, Retrieving information in Croatian: Building a simple and efficient rule-based stemmer*, Digital information and heritage. Zagreb: Odsjek za informacijske znanosti Filozofskog fakulteta u Zagrebu, pp. 313–320, 2007.

[18] N., Ljubešić, I., Pandžić, *Stemmer for Croatian: A simple rule-based stemmer for Croatian*, Filozofski fakultet Sveučilišta u Zagrebu, Zagreb Hrvatska, 2007., dostupno na: http://nlp.ffzg.hr/resources/tools/stemmer-for-croatian/ [26.08.2017.]

[19] J., Shaikh, *Machine learning, NLP: Text Classification using scikit-learn, python and NLTK*, Medium, USA, 2017. dostupno na: https://medium.com/towards-data-science/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a [31.08.2017.]

[20] *MultinominalNB*, Scikit-Learn developers, dostupno na: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB [27.08.2017.]

[21] *LinearSVC*, Scikit-Learn developers, dostupno na: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html [27.08.2017.]

[22] *CountVectorizer*, Scikit-Learn developers, dostupno na: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html [29.08.2017.]

[23] *How to use word_tokenize in data frame*, Stack Overflow, dostupno na: https://stackoverflow.com/questions/33098040/how-to-use-word-tokenize-in-data-frame [29.08.2017.]

[24] *Introduction to Flask*: *What is flask?*, Kushal Das, dostupno na: http://pymbook.readthedocs.io/en/latest/flask.html [10.09.2017.]

[25] *Flask Documentation*, dostupno na: http://flask.pocoo.org/docs/0.12/ [11.09.2017.]

[26] J., Lengstorf, *JSON: What It Is, Hot It Works, & How to Use It*, Copter Labs, 2009., dostupno na: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/ [15.09.2017.]

[27] *jQuery AJAX Methods*, w3schools, dostupno na: https://www.w3schools.com/jquery/jquery_ref_ajax.asp [15.09.2017.]

[28] *Foundation Documentation*, Foundation, dostupno na: https://foundation.zurb.com/sites/docs/ [15.09.2017.]

[29] *AJAX_Forms_jQuery_Flask*, Github, dostupno na: https://github.com/PrettyPrinted/AJAX_Forms_jQuery_Flask [15.09.2017.]

[30] *MySQLdb User's Guide*, Source Forge, dostupno na: http://mysql-python.sourceforge.net/MySQLdb.html [15.09.2017.]

SAŽETAK

Naslov: Primjena metoda strojnog učenja u klasifikaciji korisničkih upita

Sažetak: U ovom radu dan je pregled metoda za klasifikaciju korisničkih upita i obradu prirodnog jezika korištenjem strojnog učenja. Dohvaćeni su podaci sa društvenih mreža koji su zatim ručno klasificirani, obrađeni te korišteni za izgradnju i evaluaciju klasifikatora. Korišteni algoritmi za klasifikaciju su Naivni Bayesov algoritam i strojevi sa potpornim vektorima. Rezultati evaluacije dobiveni su u obliku matrice zabune i općenitog postotka ispravnog klasificiranja upita u testnom skupu. Upiti se uspješno klasificiraju te sustav pokazuje dobre rezultate i na zahtjevnijim upitima korisnika. Model je implementiran unutar web aplikacije u koju korisnik unosi upit i osnovne podatke, aplikacija klasificira upit kroz implementirane modele te konačan rezultat zajedno sa upitom i podacima sprema u bazu podataka i ispisuje u web aplikaciji.

Ključne riječi: strojno učenje, korisnički upiti, klasifikacija, obrada prirodnog jezika

ABSTRACT

Title: Machine learning in user inquiry analysis

Abstract: This paper reviews the methods for classifying user queries and natural language processing using machine learning. Data from social networks has been fetched, which are then manually classified, processed and used to construct and evaluate the classifier. Used classification algorithms are Naive Bayes algorithm and support vector machines. The test set of data was submitted to classifier, and an evaluation of the obtained classes was conducted for each query. Evaluation results were obtained in the form of confusion matrix and the overall percentage of correct classification of queries in the test set. Queries are successfully classified, and the system shows good results even on more demanding user queries. Model is implemented within a web application where user inputs the query and personal data, application classifies query through the implemented models, and final result along with query and data saves it in database and prints it in the web application.

Keywords: machine learning, user inquiry, classification, natural language processing

ŽIVOTOPIS

Nikola Komljenović rođen je 08.03.1992. godine u Županji. Nakon završene osnovne škole 2007. godine upisuje Tehničku školu Ruđera Boškovića u Vinkovcima gdje stječe zvanje arhitektonskog tehničara. Po završetku srednje škole upisuje Elektrotehnički fakultet u Osijeku, stručni studij elektrotehnike, smjer informatika. Trenutno je student druge godine diplomskog studija računalnog inženjerstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Uz redovno obrazovanje, pohađao je i privatnu glazbenu školu Sonatina gdje je polazio nastavu iz klavira i gitare kroz 6 godina. Od 2016. godine stipendist je tvrtke Sedam IT iz Zagreba.

Potpis:

_____

PRILOZI

Prilog P.2.1. [7]:

```
1. import tweepy
2. import json
3. import csv
4. import urllib
5.
6. # Autentifikacija
7. consumer_key = "Your consumer key goes here"
8. consumer_secret = "Your consumer secret goes here"
9. access_token = "Your access key goes here"
10.   access_token_secret = "Your access secret goes here"
11.
12.   accountvar = "#Football"  #Pojam za pretrazivanje
13.   outputfilecsv = accountvar+"_stream.csv"
14.   fc = csv.writer(open(outputfilecsv, 'wb'))
15.   fc.writerow(["created_at","screen_name","tweet_text","time_zone"])
16.
17.   # Primanje podataka
18.   class StdOutListener(tweepy.StreamListener):
19.       def on_data(self, data):
20.           # Povratna vrijednost u JSON formatu
21.
22.           decoded = json.loads(data)
23.           try:
24.               print decoded['entities']['media'][0]['media_url']
25.               link = decoded['entities']['media'][0]['media_url']
26.               filename = link.split('/')[-1]
27.               urllib.urlretrieve(decoded['entities']['media'][0]['me
    dia_url'],filename)
28.
29.               for media in decoded['extended_entities']['media'][0][
    'video_info']['variants']:
30.                   if 832000 in media.values():
31.                       link = media['url']
32.                       filename = link.split('/')[-1]
33.                       urllib.urlretrieve(media['url'],filename)
34.
35.
36.           except (NameError, KeyError,AttributeError):
37.
38.               pass
39.
40.           print '%s @%s: %s
    %s\n' % (decoded['created_at'],decoded['user']['screen_name'],decoded
    ['text'].encode('ascii', 'ignore'),decoded['user']['time_zone'])
41.           fc.writerow([decoded['created_at'],decoded['user']['screen
    _name'],decoded['text'].encode('ascii', 'ignore'),decoded['user']['ti
    me_zone']])
42.
43.            #Dodavanje vrijednosti u izlaznu vrijednost
44.
45.           # Konverzija u ASCII format
```

```
46.
47.             #print ''
48.             return True
49.
50.         def on_error(self, status):
51.             print status
52.
53.     if __name__ == '__main__':
54.         l = StdOutListener()
55.         auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
56.         auth.set_access_token(access_token, access_token_secret)
57.
58.         print "Showing all new tweets for %s"%accountvar
59.
60.         https://dev.twitter.com/docs/streaming-apis
61.         stream = tweepy.Stream(auth, l)
62.         stream.filter(track=[accountvar])
```

Prilog P.4.1. [17, 18, 19]:

```
1. # -*- coding: utf-8 -*-
2. from nltk import word_tokenize
3. from nltk.corpus import stopwords
4. from nltk.stem import CroatianStemmer
5. from sklearn.feature_extraction.text import CountVectorizer
6. from sklearn.naive_bayes import MultinomialNB
7. from sklearn import svm
8. from sklearn.metrics import confusion_matrix
9. from sklearn.metrics import accuracy_score
10.   import pandas as pd
11.   import numpy as np
12.   from inqury_functions import inqury_normalization
13.
14.   #Kreiranje objekta za stemmer
15.   stemmer = CroatianStemmer()
16.
17.   #Ucitavanje CSV datoteke
18.   df = pd.read_csv('Customer_data_onlyclassandmessage.csv', header=0
   , error_bad_lines=False)
19.   #---OBRADA TEKSTA---
20.
21.   #Normalizacija (mala slova)
22.   df["message"]= df["message"].str.lower()
23.   df["classification"]= df["classification"].str.lower()
24.
25.   #Uklanjanje interpunkcijskih znakova
26.   df['message'] = df['message'].str.replace('[^\w\s]','')
27.
28.   #Tokenizacija - razbijanje teksta u rijeci
29.   def apwords(words):
30.       filtered_sentence = []
31.       words = word_tokenize(words)
32.       for w in words:
33.           filtered_sentence.append(w)
34.       return filtered_sentence
35.   addwords = lambda x: apwords(x)
```

```
36.   df['message'] = df['message'].apply(addwords)
37.
38.   #Uklanjanje stop words
39.   stop = stopwords.words('croatian')
40.   df['message'] =

    df['message'].apply(lambda x: [item for item in x if item not in
   stop])

41.
42.   #Stemming - pretvaranje riječi u njihove korijene
43.   df['message']=df['message'].apply(lambda
   x : [stemmer.stem(y) for y in x])
44.
45.   # Broj upita za treniranje
46.   num_inqury = 1100
47.
48.   # Prazna lista za spremanje upita
49.   train_inqury = []
50.
51.   #Spremanje upita u listu
52.   for i in range( 0, num_inqury ):
53.       train_inqury.append(df["message"][i])
54.
55.   #Kreiranje vektorizer-a
56.   vectorizer = CountVectorizer(tokenizer=lambda
   doc: doc, lowercase=False, max_features = 1100)
57.
58.
59.   train_data_features = vectorizer.fit_transform(train_inqury)
60.
61.   train_data_features = train_data_features.toarray()
62.
63.   vocab = vectorizer.get_feature_names()
64.
65.   dist = np.sum(train_data_features, axis=0)
66.
67.   #Naive Bayes klasifikator
68.   classifier_nb = MultinomialNB(alpha = 1.8, fit_prior = False)
69.   targets = df['classification'][:1100]
70.   classifier_nb.fit(train_data_features, targets)
71.
72.   #SVM klasifikator
73.   classifier_svm = svm.LinearSVC(multi_class = 'crammer_singer')
74.   classifier_svm.fit(train_data_features, targets)
75.
76.   test_all = []
77.   train_nb = []
78.   train_svm = []
79.   for i in range(1101, 1462):
80.       test_all.append(df['classification'][i])
81.       train_inqury = [df['message'][i]]
82.       train_nb_counts = vectorizer.transform(train_inqury)
83.       train_nb.append(classifier_nb.predict(train_nb_counts))
84.       train_svm_counts = vectorizer.transform(train_inqury)
85.       train_svm.append(classifier_svm.predict(train_svm_counts))
86.
```

```
87.   print("Konfuzijska matrica za Naive Bayes:")
88.   print(confusion_matrix(test_all, train_nb, labels=['pohvala', 'kri
   tika', 'internet', 'televizija', 'mobitel','ugovor', 'fiksne
   usluge', 'nepoznato', 'infrastruktura']))
89.   print("Score za Naive Bayes:")
90.   print(accuracy_score(test_all, train_nb))
91.   print("----------------------------")
92.   print("Konfuzijska matrica za SVM:")
93.   print(confusion_matrix(test_all, train_svm, labels=['pohvala', 'kr
   itika', 'internet', 'televizija', 'mobitel','ugovor', 'fiksne
   usluge', 'nepoznato', 'infrastruktura']))
94.   print("Score za SVM:")
95.   print(accuracy_score(test_all, train_svm))
```

Prilog P.4.2.[17, 18, 19]

```
1. import re
2. from nltk.corpus import stopwords
3. import pickle
4. from nltk.stem import CroatianStemmer
5. from sklearn.feature_extraction.text import CountVectorizer
6.
7. stemmer = CroatianStemmer()
8.
9. #Funkcija koja obrađuje i normalizira upit, kao argument prima string
   sa upitom
10.   def inqury_normalization(user_input):
11.       user_input = user_input.lower()
12.       user_input = re.sub(r'[^\w\s]','',user_input)
13.       stop = stopwords.words('croatian')
14.       user_input = " ".join(filter(lambda word: word not in stop,
   user_input.split()))
15.       user_input = user_input.split()
16.       user_input = [[stemmer.stem(word) for word in sentence.split("
   ")] for sentence in user_input]
17.       user_input = [l[0] for l in user_input]
18.       x = []
19.       x.append(user_input)
20.       return(x)
```

Prilog P.5.1.[17, 18, 19, 20, 21, 22]

```
1. def classification(user_input):
2.     #Ucitavanje CSV datoteke
3.     df = pd.read_csv('..\Data\Customer_data_onlyclassandmessage.csv',
   header=0, error_bad_lines=False)
4.
5.     #---OBRADA TEKSTA---
6.
7.     #Normalizacija (mala slova)
8.     df["message"]= df["message"].str.lower()
9.     df["classification"]= df["classification"].str.lower()
10.
11.       #Uklanjanje interpunkcijskih znakova
12.       df['message'] = df['message'].str.replace('[^\w\s]','')
```

```
13.
14.       #Tokenizacija - razbijanje teksta u rijeci
15.       def apwords(words):
16.           filtered_sentence = []
17.           words = word_tokenize(words)
18.           for w in words:
19.               filtered_sentence.append(w)
20.           return filtered_sentence
21.       addwords = lambda x: apwords(x)
22.       df['message'] = df['message'].apply(addwords)
23.
24.       #Uklanjanje stop words
25.       stop = stopwords.words('croatian')
26.       df['message'] =
   df['message'].apply(lambda x: [item for item in x if item not in stop
   ])
27.
28.       #Stemming - pretvaranje rijeci u njihove korijene
29.
30.       df['message']=df['message'].apply(lambda x
   : [stemmer.stem(y) for y in x])
31.
32.       #Bag of words
33.
34.       # Broj upita za treniranje
35.       num_inqury = 1462
36.
37.       # Prazna lista za spremanje upita
38.       train_inqury = []
39.
40.       #Spremanje upita u listu
41.       for i in range( 0, num_inqury ):
42.           train_inqury.append(df["message"][i])
43.
44.
45.       vectorizer = CountVectorizer(tokenizer=lambda doc: doc,
   lowercase=False, max_features = 1100)
46.
47.       train_data_features = vectorizer.fit_transform(train_inqury)
48.
49.       train_data_features = train_data_features.toarray()
50.
51.       vocab = vectorizer.get_feature_names()
52.
53.       dist = np.sum(train_data_features, axis=0)
54.
55.       #Naive Bayes klasifikator
56.       classifier_nb = MultinomialNB(alpha = 1.8, fit_prior = False)
57.       targets = df['classification'][:1462]
58.       classifier_nb.fit(train_data_features, targets)
59.
60.       #SVM klasifikator
61.
62.       classifier_svm = svm.LinearSVC(multi_class = 'crammer_singer')
63.       classifier_svm.fit(train_data_features, targets)
64.
65.       example_counts = vectorizer.transform(user_input)
```

```
66.       predictions_nb = classifier_nb.predict(example_counts)
67.       print("Klasa iz Naive Bayes klasifikatora je: ",
   predictions_nb)
68.       predictions_svm = classifier_svm.predict(example_counts)
69.       print("Klasa iz SVM klasifikatora je: ", predictions_svm)
70.
71.       return(predictions_nb, predictions_svm)
```

Prilog P.5.2. [25, 26, 27]

```
1. # -*- coding: utf-8 -*-
2.
3. from flask import Flask, render_template, request, jsonify
4. import re
5. import dill
6. from nltk.corpus import stopwords
7. from nltk.stem import CroatianStemmer
8. from sklearn.feature_extraction.text import CountVectorizer
9. import MySQLdb
10.
11.   db=MySQLdb.connect(host='Nkomljenovic.mysql.pythonanywhere-
   services.com', user='Nkomljenovic',
   passwd='a6G9E36P',db='Nkomljenovic$user_inqury', charset='utf8')
12.   db.ping(True)
13.   x = db.cursor()
14.
15.   stemmer = CroatianStemmer()
16.
17.   app = Flask(__name__)
18.
19.   @app.route('/')
20.   def homepage():
21.       return render_template("main.html")
22.
23.   @app.route('/predict', methods=['POST'])
24.   def make_predict():
25.       def inqury_normalization(user_input):
26.           user_input = user_input.lower()
27.           user_input = re.sub(r'[^\w\s]','',user_input)
28.           stop = stopwords.words('croatian')
29.           user_input = " ".join(filter(lambda word:
   word not in stop, user_input.split()))
30.           user_input = user_input.split()
31.           user_input
   = [[stemmer.stem(word) for word in sentence.split("
   ")] for sentence in user_input]
32.           user_input = [l[0] for l in user_input]
33.           x = []
34.           x.append(user_input)
35.           return(x)
36.
37.       def classification(normalized_input):
38.           svm_load= dill.load(open('svm_new', 'rb'))
39.           naive_load= dill.load(open('naive_bayes_new', 'rb'))
40.           vectorizer= dill.load(open('vectorizer_new', 'rb'))
41.           vocab = dill.load(open('vocab_new', 'rb'))
```

```
            vectorizer = CountVectorizer(vocabulary=vocab,
   tokenizer=lambda doc: doc, lowercase=False, max_features = 1100)
            vectorizer._validate_vocabulary()
            example_counts = vectorizer.transform(normalized_input)
            predictions_naive = naive_load.predict(example_counts)
            str_naive = ''.join(predictions_naive)
            predictions_svm = svm_load.predict(example_counts)
            str_svm = ''.join(predictions_svm)
            return(str_naive, str_svm)

        upit_ispis = request.form['upit_ispis']
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        operater = request.form['operater']
        user_inqury = inqury_normalization(upit_ispis)
        inqury_classification = classification(user_inqury)
        naive_bayes_class = inqury_classification[0]
        svm_class = inqury_classification[1]

        try:
            x.execute(

     """INSERT INTO upiti (firstname, lastname, operater, upit,
   klasa_naive, klasa_svm) VALUES (%s,%s,%s,%s,%s,%s)""",(firstname,
   lastname, operater, upit_ispis, naive_bayes_class, svm_class))

            db.commit()
            return (jsonify({'upit' : upit_ispis,'klasa_naive' :
   naive_bayes_class, 'klasa_svm' : svm_class}))
        except:
            return (jsonify({'error': 'Upit nije upisan u bazu
   podataka!'}))

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

Prilog P.5.3. [28, 29, 30]

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
   scale=1.0">
    <title>ML_Inqury_Analyze</title>
    <link rel="stylesheet" href="{{url_for('static',
   filename='css/foundation.css')}}">
    <link rel="stylesheet" href="{{url_for('static',
   filename='css/app.css')}}">
</head>
  <body>
      <div class="grid-container">
        <div class="grid-x grid-padding-x">
        <div class="large-12 text-center cell">
          <h1>Analiza korisnickih upita - Naive Bayes, SVM</h1>
          <hr>
```

```
16.         </div>
17.         <div class="cell">
18.         <form>
19.         <div class="grid-x">
20.         <div class="large-4 cell">
21.           <input type="text" id="firstname" placeholder="Ime" requir
    ed/>
22.         </div>
23.         <div class="large-4 cell">
24.           <input type="text" id="lastname" placeholder="Prezime" req
    uired/>
25.         </div>
26.         <div class="large-4 cell">
27.             <select id="select_operater">
28.               <option value="#">Odaberite operatera</option>
29.               <option value="Hrvatski telekom">Hrvatski
    telekom</option>
30.               <option value="Vipnet">Vipnet</option>
31.               <option value="Optima telekom">Optima telekom</option>
32.               <option value="Iskon">Iskon</option>
33.             </select>
34.         </div>
35.         </div>
36.         <div class="large-12 text-center cell">
37.           <input type="text" id="inqury" placeholder="Unesite vaš
    upit" required/>
38.           <input class="button" type="submit" value="Pošalji upit">
39.         </div>
40.         </form>
41.         </div>
42.         <div class="cell text-center">
43.           <hr>
44.           <h3>Vaš upit i klase</h3>
45.         </div>
46.         <div class="cell">
47.           <h5>Vaš upit je: </h5>
48.           <h5 id="upit_ispis"></h5>
49.         </div>
50.         <div class="large-6 cell">
51.           <hr>
52.           <h5>Naive Bayes klasa:</h5>
53.           <h5 id="naive_bayes_ispis"></h5>
54.         </div>
55.         <div class="large-6 cell">
56.           <hr>
57.           <h5>SVM klasa:</h5>
58.           <h5 id="svm_ispis"></h5>
59.         </div>
60.         <div class="cell text-center">
61.             <div class="callout
    success" id="uspjesan_upis" style="display:none;">
62.                 <h6>Upit uspješno upisan u bazu!</h6>
63.             </div>
64.             <div class="callout
    alert" id="neuspjesan_upis" style="display:none;">
65.                 <h6>Upit nije upisan u bazu podataka! Pokušajte
    ponovno.</h6>
```

```
66.                 </div>
67.             </div>
68.         </div>
69.     </div>
70.         <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.
    0/jquery.min.js"></script>
71.         <script src="{{url_for('static', filename='js/vendor/what-
    input.js')}}"></script>
72.         <script src="{{url_for('static',
    filename='js/vendor/foundation.js')}}"></script>
73.         <script src="{{url_for('static',
    filename='js/app.js')}}"></script>
74.         <script type="text/javascript">
75.         $(document).ready(function(){
76.             $('form').on('submit', function(event){
77.                 $.ajax({
78.                     data : {
79.                         firstname: $('#firstname').val(),
80.                         lastname: $('#lastname').val(),
81.                         operater: $('#select_operater').val(),
82.                         upit_ispis: $('#inqury').val()
83.                     },
84.                     type : 'POST',
85.                     url : '/predict'
86.                 })
87.                 .done(function(data){
88.                     if (data.error){
89.                         $('#neuspjesan_upis').show();
90.                         $('#uspjesan_upis').hide();
91.                     }else{
92.                         $('#upit_ispis').text(data.upit);
93.                         $('#naive_bayes_ispis').text(data.klasa_naive;
94.                         $('#svm_ispis').text(data.klasa_svm);
95.                         $('#uspjesan_upis').show();
96.                         $('#neuspjesan_upis').hide();
97.                     }
98.                 });
99.                 event.preventDefault();
100.            });
101.        });
102.        </script>
103.    </body>
104.  </html>
```