

Elektronski fakultet Niš
Univerzitet u Nišu

Predmet: Medicinski informacioni sistemi

TEMA: Obrada podataka o anamnezama iz
niškog Doma zdravlja i njihova klasifikacija
korišćenjem Naive Bayes algoritma

Mentor: prof. dr. Dragan Janković

Student: Aleksandra Stanojević 918

Maj 2020.

Sadržaj

Uvod	2
Prikupljanje i format podataka	3
Prečišćavanje podataka	4
Korak 1. - Mala slova	5
Korak 2. - Tokenizacija	5
Korak 3. - Brisanje znakova interpunkcije	6
Korak 4. - Uklanjanje stop reči	6
Korak 5. - Normalizacija - Skraćivanje na koren reči	7
Korak 6. - Nepotrebne reči	7
Korak 7. - Spajanje reči u rečenice	8
Brisanje praznih redova	8
Model “Bag Of Words”	9
Retke matrice	9
Principal Component Analysis	10
Podela na podatke za treniranje i test podatke	10
Algoritam Naive Bayes	10
Rezultati	12
Zaključak	13
Literatura	14

Uvod

Upravljanje nestruktuiranim podacima predstavlja ogroman problem za industriju informacionih tehnologija u zdravstvu. Jedan od razloga je taj što se nestrukturirani podaci ne mogu transformisati u podesive informacije na način na koji to mogu strukturirani podaci. Jedno rešenje je pretvaranje nestrukturiranih podataka u strukturirane podatke, ali to je jako složena i skupa aktivnost.

Prema rečima Peter J. Embi-a, dr. med. predsednika i izvršnog direktora instituta Regenstrief: “Osamdeset procenata kliničkih podataka zaključano je u nestrukturiranim beleškama lekara koje EHR¹ ne mogu da pročitaju i tako im ne može pristupiti napredna podrška za donošenje odluka i aplikacije za poboljšanje kvaliteta“. Nažalost, više od 95 posto zdravstvenih sistema ne može iskoristiti ove vredne kliničke podatke, jer je potrebna analitika za pristup teška, skupa i zahteva napredni skup tehničkih veština i infrastrukture. [4]

U ovom radu opisan je pristup obradi nestruktuiranih podataka iz niškog Doma zdravlja upotrebom određenih NLP² pristupa, tako da se oni mogu koristiti za klasifikaciju pomoću algoritma Naive Bayes. Najpre se kreira model Bag Of Words nakon obrade podataka, a kako podaci sadrže veliki broj klasa, vrši klasifikacija na veći broj klasa korišćenjem Bajesovih mreža. U nastavku će biti opisan postupak obrade podataka, Naive Bayes algoritam i prikazani i diskutovani rezultati klasifikacije.

¹ **EHR** (*engl. Electronic Health Record*) - Elektronski zdravstveni podaci predstavljaju sistematizovanu kolekciju elektronskih skladištenih zdravstvenih podataka pacijenata i populacije u digitalnom formatu.

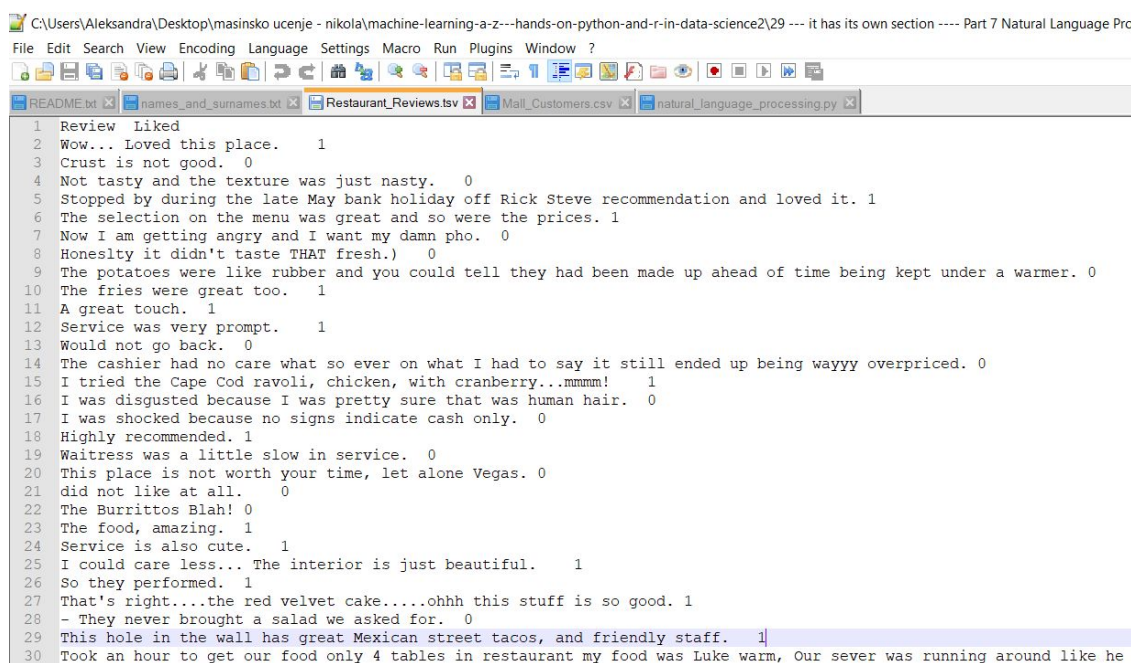
² **NLP** (*engl. Natural Language Processing*) - Podoblast lingvistike, računarstva, informacionih tehnologija i veštačke inteligencije koja se bavi interakcijom između računara i ljudskih (prirodnih jezika), posebno načinima za obradu i analizu velikih količina podataka pisanih prirodnim jezikom.

Prikupljanje i format podataka

Kao što je već napomenuto u uvodnom delu rada, više od 80% zdravstvenih podataka čine nestruktuirani podaci, koje moramo najpre obraditi da bismo nad njima mogli vršiti napredne analize i upotrebili aplikacije za podršku u donošenju odluka i povećanju kvaliteta. [2]

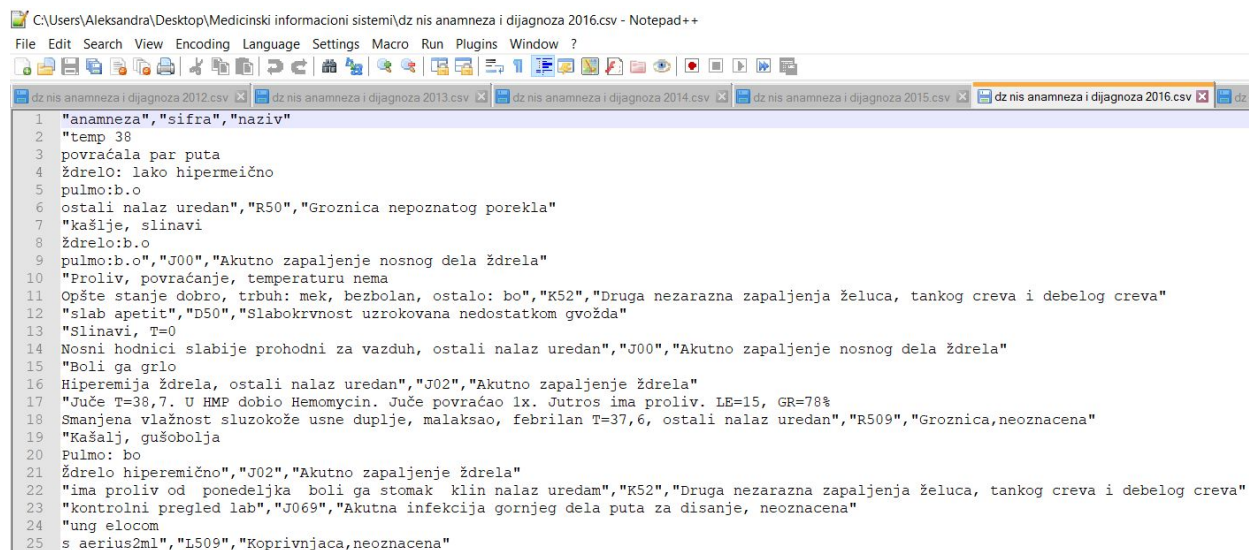
Najpre je važno razmotriti najbolji format prikupljenih podataka. Prvo je važno da postoje dve kolone, jedna koja sadrži text i druga koja sadrži ono što je ključni atribut naše analize. Ključni atribut analize može biti, ukoliko se recimo radi o komentarima, tip komentara, gde bi 0 predstavljala negativan, a 1 pozitivan komentar. Ukoliko posmatramo dve opcije .csv i .tsv fajlova, najlošiji pristup je .csv file kod koga su sadržaji ovih dveju kolona odvojeni zarezom (zarez predstavlja delimiter), jer će nam u tom slučaju biti jako teško prepoznati gde je kraj prve kolone, ako tekst u njoj takođe sadrži zarez. Dva bolja pristupa su sledeća:

1. **.tsv fajl** kod koga je delimiter tab taster kao u primeru sa *Slika 1*. - razlog je taj što se mnogo retko dešava da se među nestruktuiranim podacima nađe ovaj specijalni karakter, te je najveća verovatnoća da ćemo u ovom slučaju imati dobro prepoznate kolone;
2. **.csv file** kod koga je delimiter zarez, ali su sadržaji kolona odvojeni znacima navoda - kako je veća verovatnoća da se u nestruktuiranim podacima javi ovaj specijalni karakter, tako je manja verovatnoća da ćemo dobro prepoznati sadržaje kolona u podacima, ali u zavisnosti od samih podataka može predstavljati i dobar pristup.



Slika 1. Podaci formatirani kao .tsv

U slučaju podataka iz niškog Doma zdravlja koji se u ovom radu analiziraju i obrađuju u pitanju je 2. pristup formatiranja podataka. Podaci su anamneze lekara opšte prakse u vidu nestruktuiranih podataka u prvoj koloni. Anamneze sadrže komentare lekara i simptome pacijenta, kao i tok pregleda i rezultate nalaza. Ključni atribut analize je dijagnoza koja je podeljena na dve kolone - šifru dijagnoze i njen naziv, od kojih možemo koristiti samo jednu a drugu po potrebi odrediti na osnovu prve ili obe, u zavisnosti kako odlučimo. Prikaz formata podataka prikupljenih u niškom Domu zdravlja dat je na *Slika 2*.



```

1 "anamneza","sifra","naziv"
2 "temp 38"
3 "povraćala par puta"
4 "ždrela: lako hipermeično"
5 "pulmo:b.o"
6 "ostali nalaz uredan","R50","Groznicna nepoznatog porekla"
7 "kašlje, slinavi"
8 "ždrela:b.o"
9 "pulmo:b.o","J00","Akutno zapaljenje nosnog dela ždrela"
10 "Proliv, povraćanje, temperaturu nema"
11 "Opšte stanje dobro, trbuh: mek, bezbolan, ostalo: bo","K52","Druga nezarazna zapaljenja želuca, tankog creva i debelog creva"
12 "slab apetit","D50","Slabokrvnost uzrokovana nedostatkom gvožđa"
13 "Slinavi, T=0"
14 "Nosni hodnici slabije prohodni za vazduh, ostali nalaz uredan","J00","Akutno zapaljenje nosnog dela ždrela"
15 "Boli ga grlo"
16 "Hiperemija ždrela, ostali nalaz uredan","J02","Akutno zapaljenje ždrela"
17 "Juče T=38,7. U HMP dobio Hemomycin. Juče povraćao 1x. Jutros ima proliv. LE=15, GR=78%"
18 "Smanjena vlažnost sluzokože usne duplje, malaksao, febrilan T=37,6, ostali nalaz uredan","R509","Groznicna, neoznacena"
19 "Kašalj, gušobolja"
20 "Pulmo: bo"
21 "ždrela hipermeično","J02","Akutno zapaljenje ždrela"
22 "ima proliv od ponedeljka boli ga stomak klin nalaz uredan","K52","Druga nezarazna zapaljenja želuca, tankog creva i debelog creva"
23 "kontrolni pregled lab","J069","Akutna infekcija gornjeg dela puta za disanje, neoznacena"
24 "ung elocom"
25 "s aerius2ml","L509","Koprivnjaca, neoznacena"

```

Slika 2. Podaci iz niškog Doma zdravlja u .csv formatu

Prečišćavanje podataka

Nad nestruktuiranim podacima iz niškog Doma zdravlja biće vršeno pročišćavanje korišćenjem NLP (*engl. Natural Language Processing*) alata. Ova obrada podataka podrazumeva niz koraka koji vodi od teksta do modela Bag Of Words koji će biti rezultat prečišćavanja.

Na Slika 3. prikazan je metod za prečišćavanje reči čiji će svaki korak biti pojedinačno razmatran u nastavku ovog poglavlja. Metod, kao i čitava obrada podataka napisana je u programskom jeziku Python³ upotrebom programskog okruženja Spyder⁴.

³ **Python** je programski jezik visokog nivoa opšte namene koji podržava imperativni, objektno-orijentisani funkcionalni stil programiranja.

⁴ **Spyder** je kros-platformsko razvojno okruženje otvorenog koda (*engl. Integrated Development Environment*) za naučno programiranje na programskom jeziku Python.

```

def clean_text(raw_text):

    # Convert to lower case
    text = raw_text.lower()

    # Tokenize
    tokens = nltk.word_tokenize(text)

    # Remove punctuation
    token_words = [w for w in tokens if w.isalpha()]

    # Remove stop words
    cleaned_words = [w for w in token_words if not w in stops_words]

    # Stemming
    stemmed_words = [stem(w) for w in cleaned_words]

    # Remove unnecessary words
    cleaned_words = [w for w in stemmed_words if not w in stemmed_unnecessary_words]

    # Connect tokens into a sentence
    cleaned_text = ' '.join(cleaned_words)

    return cleaned_text

```

Slika 3. Metoda za prečišćavanje teksta

Korak 1. - Mala slova

Kako su podaci koje obrađujemo komentari napisani korišćenjem i malih i velikih slova azbuke, ne želimo da iste reči ne budu iz ovog razloga prepoznate u algoritmima koje koristimo kao različite. Iz tog razloga ćemo čitav tekst pre dalje obrade najpre prevesti tako da sva slova budu mala slova azbuke. Za pomenuto prevođenje iskorišćena je metoda za rad sa string podacima `lower()` programskog jezika Python.

```

# Convert to lower case
text = raw_text.lower()

```

Slika 4. Prevođenje svih slova u podacima u mala slova azbuke

Korak 2. - Tokenizacija

U daljoj obradi nestruktuiranih podataka algoritmi će de izvršavati nad rečima, te se iz tog razloga u ovom koraku vrši tokenizacija. Tokenizacija je razdvajanje rečenice koja je u vidu string podatka na listu čiji je svaki element jedna reč iz rečenice. U ovom slučaju korišćena je funkcija za tokenizaciju `word_tokenize(text)` biblioteke NLTK⁵.

⁵ **NLTK** (*engl. Natural Language Toolkit*) je vodeća platforma za izgradnju programa u programskom jeziku Python za rad sa podacima napisanim u prirodnim jeziku.

```
# Tokenize
tokens = nltk.word_tokenize(text)
```

Slika 5. Tokenizacija reči

Korak 3. - Brisanje znakova interpunkcije

Anamneze koje predstavljaju nestruktuirane podatke koje obrađujemo mogu sadržati slova azbuke, brojeve, specijalne karaktere i znakove interpunkcije. Mi želimo da zadržimo samo one podatke koji su nam od važnosti, a to su brojevi i slova azbuke, dok ćemo ostale obrisati.

```
# Remove punctuation
token_words = [w for w in tokens if w.isalpha()]
```

Slika 6. Brisanje znakova interpunkcije iz podataka

Korak 4. - Uklanjanje stop reči

Obavezni korak prilikom obrade nestruktuiranih podataka je uklanjanje stop reči. Stop reči su frekventne reči, kao što su do, ka, takođe, ipak i mnoge druge, koje želimo da filtriramo iz dokumenta pre dalje obrade. Ove reči uglavnom imaju malo leksičkog sadržaja i njihovo prisustvo u tekstu nije od značaja, jer nam uglavnom ne daje informacije koje možemo upotrebiti kasnije prilikom analize korišćenjem bilo kog od algoritama mašinskog učenja. [1]

```
# Remove stop words
cleaned_words = [w for w in token_words if not w in stops_words]
```

Slika 7. Uklanjanje stop reči iz podataka

Problem koji se javlja u ovom koraku je taj što je za obradu teksta napisanog na **srpskom jeziku** neophodno uvoženje datoteke koja sadrži stop reči za isti jezik. Postoji nekoliko biblioteka koje se mogu koristiti u pajtonu za NLP, a neke od njih su Natural Language Toolkit [NLTK], TextBlob, spaCy[5], a samo poslednja navedena nudi podršku i za srpski jezik. Međutim, još jedna prepreka bila je i činjenica da su stop reči za srpski jezik u ovoj biblioteci bile napisane na ćirilici, dok su podaci koji se obrađuju napisani u latinici. Ovaj problem rešen je prevođenjem liste stop reči na ćirilici, na reči u latinici korišćenjem online alata Lexilogos. [14] Ovako prevedene reči zapamćene su u datoteci i smeštene u datoteku unutar biblioteke NLTK gde su smeštene i stop reči drugih jezika koje ova biblioteka podržava.

Korak 5. - Normalizacija - Skraćivanje na koren reči

U tekstu koji obrađujemo se jako često javljaju reči takve da ista može naći u svojim različitim oblicima, odnosno padežima i vremenima. Kako želimo da na kraju obrade podataka imamo što manje različitih reči, što je važno kod kreiranja modela Bag Of Words, potrebno je da skratimo reči. Naime, što imamo više reči to ćemo kasnije imati manje dimenzija kada budemo kreirali retku matricu, o kojoj će kasnije biti više reči. Normalizacijom teksta smanjuje se broj dimenzija, tj. smanjuje se retkost, a jedna od tehnika koje se koriste u NLP je skraćivanje na koren reči. Skraćivanje na koren reči koristi niz pravila (ili modela) da iseče sve oblike jedne iste reči tako da se dobije samo jedna reč. [3] U srpskom jeziku pogotovo, ali i u slučaju drugih jezika vrlo često reči koje su rezultat skraćivanja deluju i zvuče besmisleno, no svakako to nije od važnosti, dokle god je normalizacija pravilno izvršena.

Za srpski jezik postoje dva alata za skraćivanje na koren reči, a slobodnom procenim izabran je alat Serbian stemmer koji je napisao Dragan Vidaković i koji je otvorenog koda. [6] Alat je nadogradnja na hrvatski alat napisan na Filozofskom fakultetu Univerziteta u Zagrebu, jer sam alat podrazumeva jako dobro poznavanje jezičkih pravila i najbolje je da bude napisan od strane stručnjaka. Kako ovaj alat takođe vrši i uklanjanje stop reči, ali mnogo manjeg broja nego što se otklanja u prethodnom koraku (Korak 4.), ovaj deo koda je obrisao iz alata.

```
# Stemming
stemmed_words = [stem(w) for w in cleaned_words]
```

Slika 8. Skraćivanje reči na njihov koren

Korak 6. - Nepotrebne reči

Nešto kasnije u obradi podataka će reči koje se najmanje javljaju u podacima kao što su recimo pogrešno napisane reči, vlastita imena i prezimena biti izbačene. Međutim, uočeno je da u podacima ima nekih reči koje se jako često ponavljaju, a nisu od značaja za proces određivanja dijagnoza pacijenata, što je krajnji cilj ovog rada. Izdvojeno je nešto manje od 100 reči koje su smatrane sigurnim viškom, kao “th” (skraćenica za “terapija”), “rp” (skraćenica za “redovni pregled”), kontrola, savet, doktor, uput, pregled i druge.

```
#Remove unnecessary words
cleaned_words = [w for w in stemmed_words if not w in stemmed_unnecessary_words]
```

Slika 9. Uklanjanje irelevantnih reči iz podataka

Ovaj korak je napisan specifično za uklanjanje irelevantnih podataka iz anamneza pacijenata i moguće ga je mnogo unaprediti uz konsultacije sa stručnim licima (lekarima). Pregledom podataka moguće je utvrditi koje su to reči koje se često ponavljaju, a pritom ne mogu biti simptom bolesti i samim tim se mogu dodati u listu nepotrebnih reči i obrisati iz podataka.

Korak 7. - Spajanje reči u rečenice

Ovaj korak predstavlja kraj NLP obrade podataka i podrazumeva spajanje izdvojenih relevantnih reči u rečenice, sa obzirom na to da su one tokenizovane u koraku 2. i predstavljaju svaka poseban string podatak.

```
#Connect tokens into a sentence
cleaned_text = ' '.join(cleaned_words)
```

Slika 10. Spajanje tokena u jedan string podatak

Brisanje praznih redova

Nakon pročišćavanja podataka primećeno je da su neke od anamneza postale potpuno prazni stringovi, jer su sve reči koje su u istoj bile napisane detektovane kao irelevantne. Potrebno je redove u podacima čije su anamneze prazni stringovi obrisati u potpunosti, a da bi to postigli potrebno je obaviti dva koraka:

1. Potrebno je proveriti za svaki red podataka da li u koloni koja sadrži vrednost anamneze nema podataka, odnosno da li se radi o praznom stringu. Ako je u pitanju prazan string on će biti zamenjen specijalnom vrednošću u programskom jeziku Python.
2. Zatim će svi redovi koji u koloni koja sadrži podatke o anamnezi imaju vrednost None⁶ biti izuzeti iz podataka.

```
# Replacing empty string with None
dataset['anamneza'] = dataset['anamneza'].apply(lambda y: np.nan if len(y)==0 else y)
# Removing rows with None in column ['anamneza']
dataset = dataset.dropna(axis = 0, subset = ['anamneza'])
```

Slika 11. Brisanje praznih redova iz podataka

⁶ **None** - specijalna vrednost u programskom jeziku Python koja označava da su podaci prazni, odnosno da nema podataka.

Model “Bag Of Words”

Ovaj model je najjednostavniji i najčešće se primenjuje u slučaju kada želimo da nestruktuirane podatke prilagodimo tako da se nad njima mogu primenjivati algoritmi mašinskog učenja.

Ovaj model podrazumeva kreiranje vektora za tekst, gde taj vektor ima dužinu koliko ima reči u rečenici, a vrednost predstavlja broj ponavljanja određene reči u datom tekstu. [7]

Ukoliko kreiramo ovaj model na osnovu podataka koje obrađujemo dobićemo niz vektora, odnosno matricu. Ta matrica će imati onoliko kolona koliko ima različitih reči u svim redovima podataka koje obrađujemo, a redova koliko i redova podataka koje obrađujemo, dok će vrednosti u matrici biti broj ponavljanja reči u tekstu.

Ova matrica će imati na mnogo više mesta u tabeli nultu vrednost, nego bilo koju drugu, a takva matrica se zove retka matrica i veoma je značajna u oblasti mašinskog učenja. Mnogo algoritama koji se koriste u mašinskom učenju prilagođeno je da može kao ulazni parametar da prihvati i retku matricu.

Retke matrice

Matrice koje sadrže uglavnom nulte vrednosti nazivaju se retke matrice i razlikuju se od matrica kod kojih većina vrednosti nije nula, odnosno gustih matrica. Velike retke matrice uobičajne su generalno, a posebno u primenjenom mašinskom učenju, kao što su podaci koji sadrže brojeve, kodiranje podataka koji mapiraju kategorije u brojeve, pa čak i u celim potpoljima mašinskog učenja, kao što je obrada prirodnog jezika. Rad sa retkim matricama je jako skup, a veliko poboljšanje performansi može se postići korišćenjem reprezentacija i operacija koje specifično upravljaju sa retkom matricom. [8]

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 5000)
# X = sparse_matrix
X = cv.fit_transform(dataset['anamneza']).toarray()
y = dataset.iloc[:,1].values
```

Slika 12. Kreiranje retke matrice

Prilikom kreiranja retke matrice moguće je smanjiti njenu dimenziju (broj kolona) zadržavanjem samo određenog broja reči (5000 u primeru na Slika 12.) i to prvih toliko reči po njihovoj frekvenciji pojavljivanja u podacima. Jednostavnije rečeno, biće obrisane one reči koje se retko ponavljaju u podacima. Primer iz podataka koji se obrađuju je ime i prezime lekara, koje se verovatno samo jednom pojavljuje i biće obrisano u ovom koraku.

Principal Component Analysis

Postoji još načina smanjenja dimanzija retke matrice, a to su tehnike za smanjenje dimenzionalnosti. Jedna od tehnika je Principal Component Analysis - PCA koja kreira nove kolone tako da one pokrivaju neki procenat prethodnih kolona i korisnik (programer) može da oceni koliko mu je procenata pokrivenosti dovoljno za dalju obradu. [9]

Za primere čiji će rezultati biti prikazani u jednom od narednih poglavlja uziman je broj novih kolona takav da pokriva između 30 i 50% prethodnih kolona.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 100)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
```

Slika 13. Implementacija tehnike PCA

Podela na podatke za treniranje i test podatke

Nakon potpuno završene NLP obrade podataka i kreiranja modela Bag Of Words moguće je nad podacima iskoristiti neki od algoritama mašinskog učenja za klasifikaciju podataka. Za klasifikaciju podataka izabran je algoritam Naive Bayes, koji pripada algoritmima nadgledanog učenja. Ovo znači da je prvo potrebno podatke podeliti na podatke za treniranje i test podatke. Prvi će se najpre koristiti da bi algoritam naučio o podacima i klasama, a zatim će na test podacima biti testirano koliko je predviđanje korektno.

Preporučena količina test podataka je 10 - 25%, te je prema tome količina podataka za treniranje 75 - 90%. Najbolje je testirati nekoliko kombinacija i na osnovu rezultata klasifikacije utvrditi da koja je podela najadekvatnija za konkretne podatke. [3]

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=0)
```

Slika 14. Podela na podatke za treniranje i test podatke

Algoritam Naive Bayes

Bajesova teorema je pojam iz verovatnoće, koji se koristi pri računu sa uslovljenom verovatnoćom. Ime je dobio po matematičaru Tomasu Bajesu (*engl. Thomas Bayes*), a predstavljena je sledećom formulom.

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}$$

- $p(c | d)$ - verovatnoća da će primer d biti u klasi c , ovo pokušavamo da izračunamo;
- $p(d | c)$ - verovatnoća generisanja instance d date klase c ;
- $p(c)$ - verovatnoća pojave klase c , odnosno učestalost klase c u posmatranoj bazi podataka;
- $p(d)$ - verovatnoća da će se dogoditi slučaj d . [11]

Na Sliku 15. prikazan je kod primene Naive Bayes algoritma nad podacima o anamnezama pacijenata. Algoritam je najpre izvršen i njegova predviđanja su upisana u poseban niz podataka, a zatim je radi dobijanja rezultata odnosno procenta tačnosti predviđanja izvršeno poređenje stvarnih rezultata i predviđenih rezultata.

```
# Fitting Naive Bayes Classification to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

Slika 15. Primena algoritma Naive Bayes nad podacima

Razlog korišćenja baš Naive Bayes algoritma za klasifikaciju podataka o anamnezama pacijenata niškog Doma zdravlja je taj što je ovaj algoritam prilagođen za rad sa retkim matricama i takođe može vršiti klasifikaciju kada je prisutno više od dva klase u podacima. Oba navedena bila su slučaj podataka o anamnezama, te je izabran pristup klasifikaciji korišćenjem ovog algoritma.

Rezultati

Rezultati obrade i klasifikacije podataka pri izmenama različitih parametara date su u Tabela 1.

Number of rows	Number of columns	PCA	Accuracy
4455	3000	/	0.16610549943883277
4455	3500	/	0.16722783389450055
4455	3750	/	0.16722783389450055
4455	4000	/	0.16722783389450055
4455	/	/	0.16610549943883277
4455	/	1000	0.13804713804713806
4455	/	500	0.1335578002244669
4455	/	100	0.12570145903479238
51730	4000	/	0.11231393775372124
1358045	4000	/	ERROR

Tabela 1. Rezultati klasifikacije

Možemo zaključiti da je najbolja kombinacija prema testiranim kada ne koristimo PCA tehniku i imamo 3500 kolona u retkoj matrici. Korektnost predviđenih podataka smanjuje se sa korišćenjem PCA tehnike što je razumljivo znajući način njenog funkcionisanja. Do smanjenja dolazi sa povećanjem količine podataka, a razlog je taj što svaka nova količina podataka donosi i nove klase, odnosno dijagnoze. Da bi se poboljšala korektnost klasifikacije potrebno je da podaci (redovi) sa istom klasom (dijagnozom) budu prisutni i u test i u trening podacima i to u većoj meri u slučaju trening podataka.

Problem koji se javlja kada pokušamo program da izvršimo nad svim podacima o anamnezama pacijenata iz 2018. godine dolazi do greške koja nastaje usled nedostatka memorije za smeštanje toliko velike retke matrice. Jedan od mogućih unapređenja ovog softvera je i proučavanje i primena algoritama za veliku količinu podataka (Big Data) koji potencijalno mogu rešiti ovaj problem ili jednostavno drugi načini i algoritmi za smanjenje dimenzionalnosti.

Podaci nad kojima je vršena obrada su anamneze iz niškog Doma zdravlja iz 2018. godine, tako što su uzimani prvih 4455 i prvih 52730, a zatim i svi redovi.

Zaključak

U ovom radu su obrađivani podaci iz niškog Doma zdravlja iz 2018. godine. Najpre je vršena NLP obrada podataka od prevođenja svih slova u mala slova azbuke, tokenizacije, izbacivanja znakova interpunkcije, stop reči i nepotrebnih reči do svođenja reči na njihov koren i spajanja tokena u string podatke. Zatim je napravljen od liste string podataka, koja predstavlja rezultat prečišćavanja podataka, model Bag Of Words, odnosno retka matrica. Retka matrica je skup podataka koji smo zatim iskoristili za klasifikaciju korišćenjem Naive Bayes algoritma mašinskog učenja. Rezultati u rasponu korektnog predviđanja od 11 do 17% su jako malo zadovoljavajući pogotovo sa obzirom na to da se povećanjem podataka ne dobijaju bolji već sve gori rezultati.

Poboljšanja algoritma bi podrazumevala upotrebu drugih algoritama za klasifikaciju podataka iz oblasti mašinskog učenja i njihovo poređenje, kao i testiranja kombinacija parametara različitih koraka algoritma. Ova poboljšanja mogu uticati na bolje rezultate klasifikacije, ali ne nužno. Naime, najveći problem su sami podaci koji u velikoj količini ne sadrže relevantne informacije. Problem uočen kod podataka iz niškog Doma zdravlja su irelevantni podaci koji u nekim slučajevima uopšte ne predstavljaju anamnezu i simptome pacijenata.

Najvažnije za dobre rezultate, bilo koje od obrada podataka algoritmima mašinskog učenja, su dobro prikupljeni i formatirani podaci. Neke od ideja su ograničenje unošenja samo simptoma u prozoru za unos anamneze i to razdvojeni zarezom, dok je za pisanje liste simptoma morali da se konsultuju stručnjaci (medicinski radnici). Takođe, je važno sprečiti unos reči sa greškama u spelovanju i nepostojećih reči.

Oblast upotrebe algoritama mašinskog učenja u oblasti zdravstva još uvek je nedovoljno istražena i radi se na njenom unapređenju, te se očekuje još mnogo novina, algoritama i alata u skoroj budućnosti. Te novime će pomoći kreiranju korektnijih sistema predlaganja dijagnoza na osnovu unosa simptoma i još puno drugih aplikacija.

Literatura

- [1] Bird, S., Klein, E., Loper, E., (2009) *Natural Language Processing with Python*. 1st edn. United States of America: O'Reilly Media.
- [2] Ciampa, M., Revels, M., (2012) *Introduction to Healthcare Information Technology*, 1st edn. United States of America: Cengage Learning.
- [3] Benjamin Bengfort, Rebecca Bilbro, Tony Ojeda, (2018) *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*, 1st edn. United States of America: O'Reilly Media.
- [4] Regenstrief Institute, Inc. (2017) *nDepth* Available at: <https://www.regenstrief.org/implementation/ndepth/> (Accessed 25 April 2020)
- [5] Matthew Honnibal (2015) *spaCy* Available at: <https://spacy.io/> (Accessed 22 March 2020)
- [6] Dragan Vidaković (2017) *Serbian Stemmer* Available at: <https://github.com/vdragan1993/serbian-stemmer> (Accessed 24 March 2020)
- [7] Michael Allen (2018) *104: Using free text for classification – ‘Bag of Words’* Available at: <https://pythonhealthcare.org/2018/12/15/104-using-free-text-for-classification-bag-of-words/> (Accessed 3 April 2020)
- [8] Jason Brownlee (2018) *A Gentle Introduction to Sparse Matrices for Machine Learning* Available at: <https://machinelearningmastery.com/sparse-matrices-for-machine-learning/> (Accessed 3 April 2020)
- [9] Ian T. Jolliffe and Jorge Cadima (2016) Principal component analysis: a review and recent developments Available at: <https://doi.org/10.1098/rsta.2015.0202> (Accessed 10 May 2020)
- [10] Michael Allen (2018) Python for healthcare modelling and data science Available at: <https://pythonhealthcare.org/category/natural-language-processing/> (Accessed 20 April 2020)

- [11] Dan Jurafsky - Stanford University (2011) *Text classification and Naive Bayes* Available at: <https://web.stanford.edu/class/cs124/lec/naivebayes.pdf> (Accessed 10 March 2020)
- [12] David Cournapeau and Matthieu Brucher (2007) scikit-learn Available at: <https://scikit-learn.org/> (Accessed 4 March 2020)
- [13] Python Software Foundation (2001-2020) *Python* Available at: <https://docs.python.org/3/> (Accessed 4 March 2020)
- [14] Xavier Negre (2002-2020) *Serbian Cyrillic-Latin conversion* Available at: https://www.lexilogos.com/keyboard/serbian_conversion.htm (Accessed 22 March 2020)