

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

<b>Uvod</b>	<b>2</b>
<b>Prikupljanje i format podataka</b>	<b>3</b>
<b>Balansiranje podataka</b>	<b>4</b>
<b>Prečišćavanje podataka</b>	<b>6</b>
Korak 1. - Mala slova	6
Korak 2. - Tokenizacija	7
Korak 3. - Brisanje znakova interpunkcije	7
Korak 4. - Uklanjanje stop reči	7
Korak 5. - Normalizacija - Skraćivanje na koren reči	8
Korak 6. - Nepotrebne reči	9
Korak 7. - Spajanje reči u rečenice	9
Brisanje praznih redova	9
<b>Model “Bag Of Words”</b>	<b>10</b>
Retke matrice	10
Principal Component Analysis	11
<b>Podela na podatke za treniranje i test podatke</b>	<b>11</b>
<b>Algoritam Naive Bayes</b>	<b>12</b>
<b>Rezultati</b>	<b>13</b>
<b>Zaključak</b>	<b>15</b>
<b>Literatura</b>	<b>17</b>

# 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 EHR<sup>1</sup> 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]

Podaci koji će se u ovom radu obrađivati su anamneze svih bolesti koje su detektovane u niškom Domu zdravlja u toku 2018. godine. U pitanju su podaci od 1 334 138 redova sa 5 523 klasa odnosno dijagnoza bolesti od “Površinske povrede ručja i šake”, “Akutan bol u trbuhu”, “Akutno zapaljenje bešike” do “Zloćudni tumor zadnjeg creva”, “Zloćudni tumor mozga” i mnogih drugih.

U ovom radu opisan je pristup obradi nestruktuiranih podataka iz niškog Doma zdravlja upotrebom određenih NLP<sup>2</sup> 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.

---

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

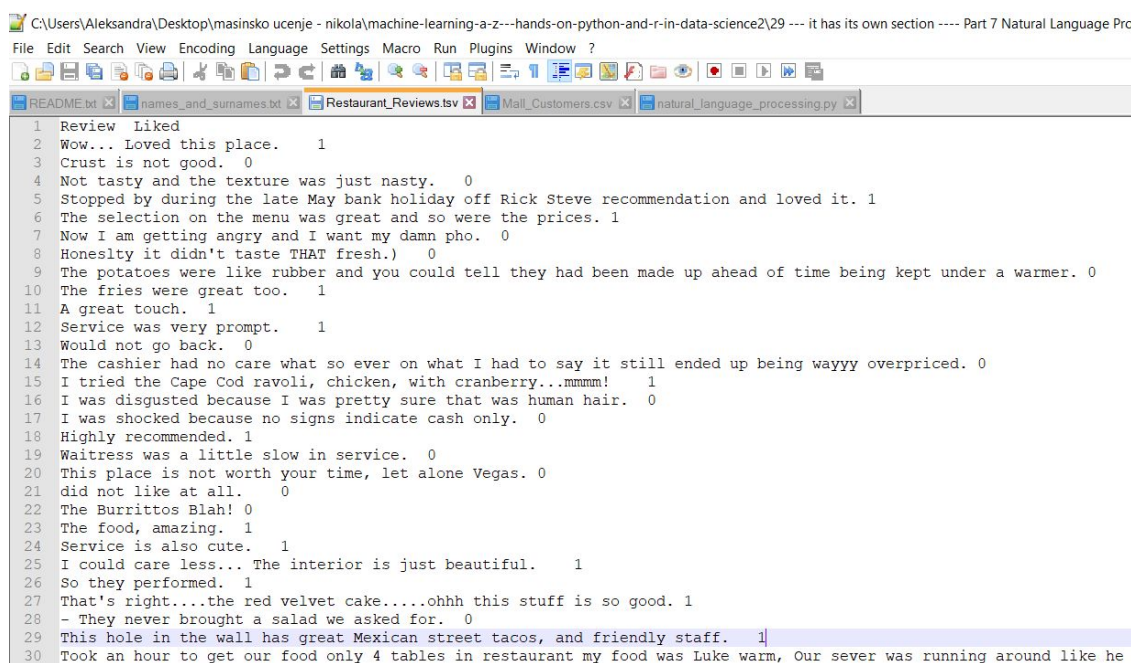
<sup>2</sup> **NLP** (*engl. Natural Language Processing*) - Podoblast lingvistike, računarstva, informacionih tehnologija i veštačke inteligencije koja se bavi interakcijom između računara i ljudskih (prirodnih jezika), posebno načinima za obradu i analizu velikih količina podataka pisanih prirodnim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



```
C:\Users\Aleksandra\Desktop\masinsko ucenje - nikola\machine-learning-a-z---hands-on-python-and-r-in-data-science2\29 --- it has its own section ---- Part 7 Natural Language Proc
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
names_and_surnames.txt Restaurant_Reviews.tsv Mail_Customers.csv natural_language_processing.py
1 Review Liked
2 Wow... Loved this place. 1
3 Crust is not good. 0
4 Not tasty and the texture was just nasty. 0
5 Stopped by during the late May bank holiday off Rick Steve recommendation and loved it. 1
6 The selection on the menu was great and so were the prices. 1
7 Now I am getting angry and I want my damn pho. 0
8 Honeslty it didn't taste THAT fresh.) 0
9 The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer. 0
10 The fries were great too. 1
11 A great touch. 1
12 Service was very prompt. 1
13 Would not go back. 0
14 The cashier had no care what so ever on what I had to say it still ended up being wayyy overpriced. 0
15 I tried the Cape Cod ravoli, chicken, with cranberry...mmmm! 1
16 I was disgusted because I was pretty sure that was human hair. 0
17 I was shocked because no signs indicate cash only. 0
18 Highly recommended. 1
19 Waitress was a little slow in service. 0
20 This place is not worth your time, let alone Vegas. 0
21 did not like at all. 0
22 The Burritos Blah! 0
23 The food, amazing. 1
24 Service is also cute. 1
25 I could care less... The interior is just beautiful. 1
26 So they performed. 1
27 That's right....the red velvet cake....ohhh this stuff is so good. 1
28 - They never brought a salad we asked for. 0
29 This hole in the wall has great Mexican street tacos, and friendly staff. 1
30 Took an hour to get our food only 4 tables in restaurant my food was Luke warm, Our sever was running around like he
```

*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, temperatura 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 hiperemično","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*

## Balansiranje podataka

Podaci koji će se u nastavku obrađivati su jako veliki, sadrže 1334138 redova i predstavljaju anamneze prikupljenje u niškom Domu zdravlja za ukupno 5523 različitih bolesti. Ovolika količina podataka ne može se koristiti ni u jednom algoritmu mašinskog učenja za klasifikaciju, pa je potrebno da se oni smanje a zatim i prečiste.

Od ukupno 5523 klasa postoje i one koje imaju po jednu ili dve anamneze iz čitave godine, te je klasifikacija kod takvih klasa apsolutno besmislena, te će zbog toga sve klase koje u podacima imaju manje od 50 anamneza biti odbačene. Na ovaj način smo smanjili podatke za nekih 70000 redova.

Između preostalih 1296 klasa uočena je velika razlika u količini anamneza za svaku klasu, naime neke imaju 50 anamaneza dok druge imaju i preko 100000 anamneza. Ovakvi podaci su neuravnoteženi..

Neuravnoteženi skup podataka je onaj u kome postoji ozbiljan nedostatak u distribuciji klasa. Ova pristranost u skupu podataka može uticati na mnoge algoritme mašinskog učenja, a neki od njih u potpunosti ignorišu manjinsku klasu.

Jedan pristup rešavanju problema neuravnoteženosti je nasumično prilagođavanje skupa podatak. Dva glavna pristupa za nasumično preraspodelu neuravnoteženog skupa podataka su brisanje primera iz većinske klase (*engl. undersampling*) i dupliranje primera iz manjinske klase (*engl. oversampling*). [15]

```
#importing the data
dataset = pd.read_csv("dz_nis_anamneza_i_dijagnoza_2018.csv")

#counting number of rows for each unique value from column 'sifra'
list = dataset.groupby(['sifra']).size().reset_index(name='counts')

#removing rows containing less or equal than 50 in column 'counts'
list = list.drop(list[list.counts < 50].index)

#removing all rows from data which contain a value in column 'sifra' that is in list['sifra']
dataset = dataset[dataset['sifra'].isin(list['sifra'])]

#preparing data for undersampling
X = dataset.iloc[:, [0]].values
y = dataset.iloc[:, [1]].values

#define undersample strategy
undersample = RandomUnderSampler(sampling_strategy='not minority')

#fit and apply the transform
X_over, y_over = undersample.fit_resample(X, y)
X_over = X_over.tolist()
y_over = y_over.tolist()

#turn lists to dataframe
dataset = pd.DataFrame(
    {'anamneza': X_over,
     'sifra': y_over
    })
```

*Slika 3. Balansiranje podataka korišćenjem tehnike undersampling*

Na Slika 3. prikazan je postupak balansiranja podataka primenom pristupa brisanja iz većinske klase (*engl. undersampling*). Izabrana strategija brisanja je “not minority” koja podrazumeva svođenje broja redova svake klase na broj redova koliko ima klasa sa najmanje redova. U primeru sa 1296 klasa, klasa sa najmanje redova ima 50 redova, te kada se iz svih ostalih klasa koje imaju više redova nasumično obrišu redovi tako da ih ostane 50 dobijamo 64800 redova podataka. Ova količina je i dalje jako velika, ali se oni ipak mogu obraditi za razliku od količine koju smo imali pre balansiranja, a pritom balansiranje značajno doprinosi i samim algoritmima klasifikacije.

## 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 texta do modela Bag Of Words koji će biti rezultat prečišćavanja.

Na Sliku 4. 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 Python<sup>3</sup> upotrebom programskog okruženja Spyder<sup>4</sup>.

```
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 4. 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.

---

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

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



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

*Slika 5. 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 NLTK<sup>5</sup>.

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

*Slika 6. 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 7. 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]

---

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



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

*Slika 8. 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 9. 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”, na engleskom “therapy”), “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 10. 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 nepotrebni 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 11. 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 None<sup>6</sup> 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 12. Brisanje praznih redova iz 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]

---

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

```

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 13. Kreiranje retke matrice*

Prilikom kreiranja retke matrice moguće je smanjiti njenu dimenziju (broj kolona) zadržavanjem samo određenog broja reči (5000 u primeru na Slika 13.) 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 14. 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 15. 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 Slika 16. 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 16. 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.

Broj redova	Broj klasa	Broj kolona	Balansirani podaci	NLP obrada	PCA	Preciznost
1334138	5523	5000	ne	ne	/	ERROR
64800	1296	5000	da	ne	/	0.086033950 61728394
64800	1296	5000	da	da	/	0.116880513 23175622
6000	18	5000	ne	ne	/	0.150833333 33333335
6000	18	5000	da	ne	/	0.327777777 7777778
6000	18	5000	da	da	/	0.423076923 0769231
6000	18	5000	da	da	500	0.230769230 76923078
6000	18	5000	da	da	300	0.287878787 8787879
6000	18	5000	da	da	100	0.323076923 0769231
6000	18	1000	da	da	/	0.378787878 7878788
6000	18	500	da	da	/	0.343511450 3816794

**Tabela 1.** Rezultati klasifikacije

Problem koji se javlja kada pokušamo da izvršimo klasifikaciju bez prethodne obrade podataka nad svim podacima o anamnezama pacijenata iz 2018. godine dolazi do greške koja nastaje usled

nedostatka memorije za smeštanje toliko velike retke matrice. Naime kako imamo 1334138 redova podataka i 5523 klasa, ovakve podatke bez prethodne obrade ne možemo upotrebiti u svrhe klasifikacije.

U Tabela 1. posmatrane su dve situacije, odnosno svi podaci i prvih 6000 redova podataka. Tehnike čije promene parametra su nam od najvećeg značaja su broj kolona, balansiranje i NLP obrada.

Prema rezultatima vidimo da je i u slučaju velike količine podataka i manje količine podataka dobijen najbolji rezultat klasifikovanja korišćenjem Naive Bayes algoritma kada su bili primenjeni i balansiranje i NLP obrada.

Kada je u pitanju parametar broj kolona, vidimo da on može dovesti do poboljšanja, ali i pogoršanja u rezultatima, te ga je potrebno ručno odabrati.

Korišćenje PCA tehnike smanjuje korektnost klasifikacije, ali je znajući način njenog funkcionisanja i uloge takav ishod očekivan.

Ono što je odmah uočljivo je da što imamo manje klasa to je korektnost predviđanja veća, a i sami algoritmi za klasifikaciju podataka najbolje rezultate daju u slučaju klasifikacije na dve klase, te ovo treba imati na umu još prilikom prikupljanja podataka.



## Zaključak

U ovom radu su obrađivani podaci iz niškog Doma zdravlja iz 2018. godine. U pitanju su podaci od 1 334 138 redova sa 5 523 klasa. Kako se u medicinskoj informatici teži kreiranju sistema koji bi na osnovu unešenih simptoma mogli dati dijagnozu, odnosno detektovati bolest, idealni slučaj bi bio sistem sa visokom preciznošću klasifikacije koji se može koristiti za sve klase (moguće bolesti).

Sa obzirom na količinu podataka, ukoliko pokušamo njihovu klasifikaciju za 5523 klasa bez prethodne obrade podataka dolazi do greške koja nastaje usled nedostatka memorije za smeštanje toliko velike retke matrice. Da bismo mogli primeniti algoritam klasifikacije potrebno je da pre toga obradimo ove podatke.

Najpre je izvršeno balansiranje podataka, tako što je izdvojeno 1296 klasa, odnosno onih koje se javljaju kao dijagnoza u najmanje 50 redova. Tehnikom undersampling-a nasumično je izdvojeno po 50 redova za svaku od 1296 klasa i tako dobijeno 64 800 redova podataka.

Zatim da bi se smanjio broj različitih reči u anamnezama vršena je NLP obrada podataka od prevođenja svih slova u mala slova azbuke, tokenizacije, izbacivanja znakova interpunkcije, stop reči i nepotrebnih reči do svodenja reči na njihov koren. Zatim je od prečišćenih podataka kreiran model Bag Of Words, odnosno retka matrica koju smo zatim iskoristili za klasifikaciju korišćenjem Naive Bayes algoritma mašinskog učenja.

Rezultati korektnog predviđanja, odnosno klasifikovanja, od 11% su jako malo zadovoljavajući, te je radi komparacije izdvojeno prvih 6000 redova podataka i nad njima izvršena ista obrada a zatim i klasifikacija na 18 klasa (svih klasa sa najmanje 50 redova). Uočeno je da u slučaju manje količine klasa imamo znatno bolje rezultate klasifikacije i to 42% korektnih predviđanja.

Krenuvši od podataka nad kojima nije bilo moguće primeniti nijedan od algoritama klasifikacije, obradom se došlo do klasifikacije na 1296 klasa, međutim ovakva klasifikacija je dala loše rezultate. Sledeći koraci u obradi bi bili različiti algoritmi za smanjenje dimenzionalnosti matrice, što bi moglo značajno ubrzati proces klasifikacije koji je za 64 800 podataka trajao oko 50 minuta. A što se tiče korektnosti predviđanja 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 primenom algoritama 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. Takođe, je važno sprečiti unos reči sa greškama u spelovanju i nepostojećih reči.

Sa prethodno navedenim unapređenjima potencijalno bi se mogao konstruisati sistem koji bi vršio klasifikaciju na preko 1000 klasa sa visokom korektnošću, no za tako nešto potrebno je dosta testiranja, proučavanja i konsultacije sa stručnim licima (medicinskim osobljem).

# 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](https://www.lexilogos.com/keyboard/serbian_conversion.htm) (Accessed 22 March 2020)
- [15] Jason Brownlee (2020) *Random Oversampling and Undersampling for Imbalanced Classification* Available at: <https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/> (Accessed 10 March 2020)