|  |
| --- |
|  |
| Predicting altruism through free pizza |
| Matea Pešić, Martina Radenić, Dora Raštegorac, Petra Staroverški |

|  |
| --- |
| Sveučilište u Zagrebu Prirodoslovno – matematički fakultet Matematički odsjek  2020./2021. |

Sadržaj

[UVOD 2](#_Toc65163495)

[OPIS PROBLEMA 2](#_Toc65163496)

[MACHINE LEARNING 3](#_Toc65163497)

[PROVEDBA ALGORITMA 4](#_Toc65163498)

[BAYESOVE MREŽE 6](#_Toc65163499)

[USPOREDBA ALGORITAMA 7](#_Toc65163500)

[Još neke usporedbe mogućih implementacija 8](#_Toc65163501)

[ZAKLJUČAK 9](#_Toc65163502)

[LITERATURA 10](#_Toc65163503)

# UVOD

Problem kojim se bavimo u ovom projektu je predviđanje čina nesebičnosti kroz dobrotu nasumičnih stranaca. Želimo pronaći algoritam koji će predvidjeti kojem će korisniku društvenih mreža neki drugi korisnik kupiti pizzu. Usporedit ćemo dva načina implementacije.

Projekt je započeo od strane Tim Althoff-a na Kaggle-u  
Preko Reddit zajednice, korisnici mogu preko poruke predati zahtjev za besplatnom pizzom.  
Npr.: „Otpjevati ću pjesmu, odsvirati instrument, samo mi dajte pizzu!“  
Ako netko od drugih korisnika odluči da mu se zahtjev svidio te pošalje besplatnu pizzu, smatramo da je on bio uspješan.

# OPIS PROBLEMA

Predviđajući uspjeh nesebičnih zahtjeva za pizzu, naš zadatak je prepoznati pozitivne, negativne i neutralne zahtjeve te osjećaje koji su pri tom izneseni. Analiziramo vrste riječi koje se pojavljuju na temelju čega donosimo odluku o optimalnim zahtjevima s najvećom mogućnosti uspjeha. Pomoću danog teksta pokušavamo predvidjeti kako će ljudi reagirati na njega, koje će osjećaje izazvati te koje su šanse uspjeha.   
Za svaku riječ dajemo ocjenu njene „polarnosti“ koja može biti pozitivna, negativna i neutralna. Na primjer: „Volim ovaj žanr, ali mrzim ovaj film.“.  
Analizirajući dijelove ove rečenice dolazimo do zaključka:  
-*Volim ovaj žanr*, ima pozitivnu konotaciju zbog riječi volim *-Mrzim ovaj film,* ima negativnu konotaciju zbog riječi mrzim

Kao ulaz problema koristimo bazu podataka tekstualnih zahtjeva te za svaki od podataka izvlačimo informacije o njegovoj polarnosti. Cilj izlaza algoritma je na osnovu dobivenih informacija predvidjeti hoće li zahtjev biti uspješan ili ne.

# MACHINE LEARNING

Problemu ćemo pristupiti koristeći strojno učenje pomoću kojeg u više faza dolazimo do zaključaka te u konačnici i do traženog odgovora. Faze su opisane sljedećom slikom:

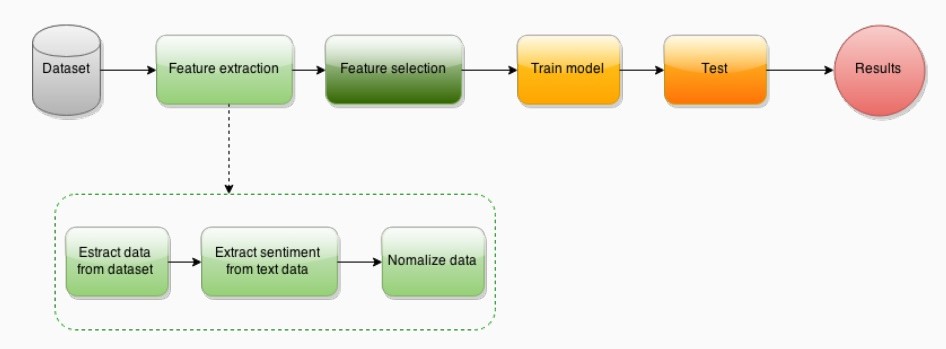


Figure 1: Faze izvođenja

Dakle, iz zadane baze podataka tekstualnih zahtjeva izvlačimo trenutni podatak na kojem radimo, iz njega izvedemo sve moguće komponente koje možemo proučavati i uspoređivati te na temelju odluke o polarnosti određujemo daljnji slijed operacija. Testiramo dane odluke i na temelju toga donosimo rezultat.

Radimo s bazom podataka koja se sastoji od 5671 tekstualnih zahtjeva za pizzu. Svaki od primjera zahtjeva sadrži nekoliko informacija:  
 -Naslov, tj. vrstu sadržaja koja je zahtjev, zahvala ili ponuda slanja pizze.   
 -Informacije o samom zahtjevu kao što su broj komentara, broj lajkova i slično.  
 -Opis korisnika koji podnosi zahtjev, npr. dob i vrijeme zahtjeva.  
 -Polje koje sadrži oznaku o tome je li zahtjev već ispunjen ili ne   
 (taj zahtjev nam daje najbitnije informacije za daljnji rad).

Tekstualna svojstva:  
 -Za svaki zahtjev najbitnije svojstvo su **naslov** i **vrsta** zahtjeva  
 - Sva svojstva koja promatramo transformiramo u numerički format  
 -Tako preoblikovane podatke dalje obrađujemo pomoću danog machine learning algoritma

Da bi preoblikovali tekst u podatke spremne za obradu potrebne su nam još dvije mjere:  
 -**Količina sentimenta**, tj. koliku količinu osjećajnih riječi korisnici koriste pri predaji zahtjeva.  
 -**Stopa uspjeha**, tj. kolika je bila uspješnost predanog zahtjeva gledajući ispunjene zahtjeve i količinu lajkova.

# PROVEDBA ALGORITMA

Dana baza podataka se nalazi u Json formatu, koristimo Python da bi izvukli potrebne podatke za daljnju obradu te NLTK-ov API da bi dobili polarnost zahtjeva.

NLTK (*Natural Language Tool Kit*) je vodeća platforma za rad s podatcima na ljudskom jeziku u izgradnji Python programa. Sadrži više od 50 Corpusa, leksičke resurse te paket biblioteka za obradu teksta za klasifikaciju, tokenizaciju te mnoge druge funkcije.

U našem algoritmu koristimo platformu za semantičko obrazloženje, tj. za dani tekst NLTK vraća jednu od tri vrste polarnosti: pozitivnu, negativnu ili neutralnu. Ako je vrijednost polarnosti neutralna, sentimentalna vrijednost je veća od 0.5 i zahtjev je označen kao neutralan. Inače je tekst označen kao veći od dviju varijabli, pozitivan ili negativan koje u sumi moraju davati jedan.  
Te tri vrijednosti se mogu prikazati jedinstvenom vrijednošću:  
Neka su pPos i PNeu NLTK vrijednosti koje označavaju pozitivni i negativni sentiment, tada je

SClabel = pPos \* sign(0.5 – pNeu)

gdje je funkcija sign zadana kao:

Ta jedinstvena varijabla čuva informacije o pozitivnosti i polarnosti zahtjeva.  
Npr. vrijednost varijable SClabel -0.7 nam označava da je zahtjev neutralan zbog negativnog predznaka te je njegova pozitivnost jednaka 0.7 (dakle negativnost iznosi 0.3).  
Izvlačimo novo svojstvo da bi saznali stopu uspjeha zahtjeva:  
U polju „Bag of Words“ se nalaze najčešće riječi koje rezultiraju uspješnom zahtjevu. Za svaku riječ pamtimo broj pojavljivanja te riječi u zahtjevu.  
Stopu uspjeha u konačnici računamo kao:

Npr. za rečenicu „*Home, sweet home*“ stopa uspjeha se računa kao:

Dakle, skupili smo podatke za matricu [5671 x 25] čiji redovi predstavljaju primjere zahtjeva, a stupci svojstva tih zahtjeva.  
Od tih 25 svojstava su 4 vezana za zahtjev, 17 za korisnika, 2 za sentiment (prethodno definirane SClabel[naslov] i SClabel[tekst]) te 2 vezana za uspjeh (stopaUspjeha[naslov] i stopaUspjeha[tekst])

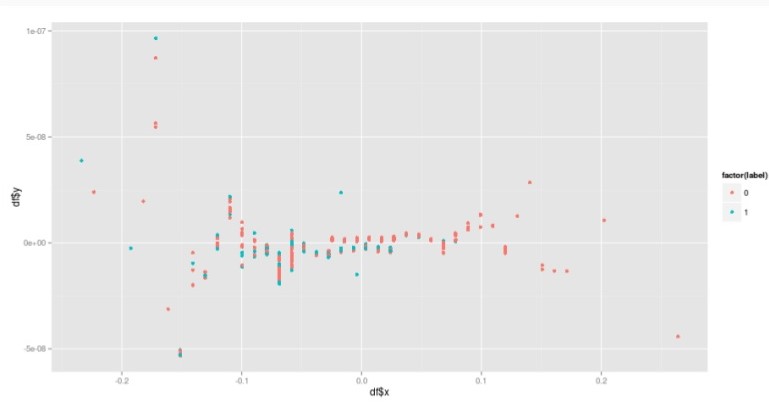


Figure 2: 2D vizualizacija baze podataka

Da bi uskladili rang svojstava koristimo formulu : Xnova =

Gdje je X stupac matrice podataka, μ je aritmetička sredina te je σ standardna devijacija.  
Podijelili smo podatke nasumičnim odabirom bez ponavljanja na sljedeći način:  
-set za treniranje [3884 x 25] 70% podataka  
-set za testiranje [1664 x 25] 30% podataka

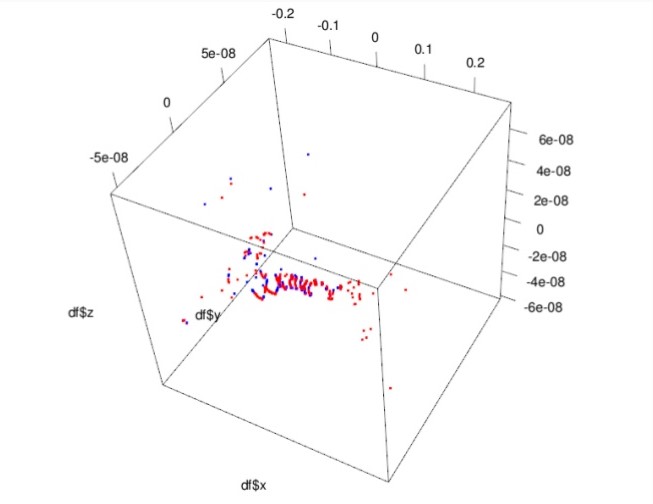


Figure 3: 3D vizualizacija baze podataka

# BAYESOVE MREŽE

Bayesova mreža je prikaz zajedničke distribucije vjerojatnosti strukturom usmjerenog acikličkog grafa i lokalnih vjerojatnosti. U mnogim slučajevima Bayesova mreža reducira broj parametara s eksponencijalnog u linearni.

Naivni Bayesovi klasifikatori su zbirka klasifikacijskih algoritama koji se temelje na Bayesovom teoremu. To nije jedan algoritam već obitelj algoritama gdje svi dijele zajedničko načelo, tj. svaki par obilježja koja se klasificiraju neovisan je jedan o drugom.

Bayesovo pravilo glasi:

gdje su A i B događaji.

U osnovi pokušavamo pronaći vjerojatnost događaja A, s obzirom da je događaj B istina. Dokaz je vrijednost atributa nepoznate instance (ovdje je to događaj B). P(A|B) je posteriori vjerojatnost za B, tj. vjerojatnost događaja nakon što se vide dokazi, a P(A) je priori A, odnosno vrijednost događaja prije nego što se vide dokazi.

Osnovna naivna Bayesova pretpostavka je da svaki atribut ima neovisan i jednak doprinos ishodu. U pravilu ove pretpostavke nisu točne u stvarnim situacijama, točnije pretpostavka o neovisnosti nikada nije točna, ali često dobro djeluje u praksi.   
Bayesov teorem možemo primijeniti na sljedeći način:

gdje je y varijabla klase, a X ovisni vektor obilježja,

Pri kreiranju modela klasifikatora nalazimo vjerojatnosti danog skupa ulaza za sve moguće vrijednosti varijable klase y i odabiremo izlaz sa najvećom vjerojatnošću. Ova metoda primjenjiva je za diskretne podatke. U slučaju kontinuiranih podataka potrebno je iznijeti neke pretpostavke u vezi distribucije vrijednosti svake značajke. Različiti naivni Bayesovi klasifikatori razlikuju se uglavnom prema pretpostavkama koje daju u vezi s distribucijom .

# USPOREDBA ALGORITAMA

Usporedba losiji\_pizza.py koda sa Bolji\_free\_pizza.py kodom:

Sličnosti:

* Oba koda koriste 'stopwords' - riječi koje nam ne daju nikakvo značenje o samoj rečenici te izbacivanjem istih ne mijenjamo strukturu i značenje te rečenica. Te riječi učitavamo pomoću naredbe 'from nltk.corpus import stopwords'.
* Također, kodovi svim riječima početna slova pretvaraju u mala te iz teksta miču dijelove koji nam nisu potrebni, npr. interpunkcijski znakovi
* Rade vektorizaciju skupova podataka
* Koriste naivni Bayesov model

Razlike:

* losiji\_pizza.py kod uz izbacivanje znakova koji nam ne trebaju u daljnjoj obradi i 'stopwords' riječi obavlja procese stemming-a i lematizacije. Stemming pokušava skratiti riječ koristeći standardna regex pravila, a lematizacija pokušava naći korijen riječi uz pomoć lingvističkih pravila (uz korištenje regex-a). Zbog tih procesa kod će se izvršavati duže nego Bolji\_free\_pizza.py
* Bolji\_free\_pizza.py kod, nakon što je isprobao točnost naivnog Bayesovog klasifikatora (koristimo MultinomialNB), ide u proces poboljšanja dobivenog modela, tj. traži vrijednosti koje maksimiziraju varijable 'roc' i 'auc' te onda s tim poboljšanim modelom trenira cijeli skup podataka, dok losiji\_pizza.py to ne radi
* Na kraju, kod predviđanja hoće li osoba, iz danog skupa podataka kojeg trebamo predvidjeti, dobiti pizzu Bolji\_free\_pizza.py koristi .predict\_proba() koji će vratiti vjerojatnost (decimalni broj) da će ta osoba dobiti pizzu, dok losiji\_pizza.py koristi .predict() koji će kao output za predviđanje vratiti ili 0 ili 1.

Zbog ovdje navedenih razloga je točnost koda Bolji\_free\_pizza.py (cca. 60%) bolja od točnosti koda losiji\_pizza.py (cca. 50%)

## Još neke usporedbe mogućih implementacija

Osim navedene dvije implementacije, naravno postoje još mnoge mogućnosti provedbe algoritma poput: algoritma k-najbližih susjeda, slučajne šume te metoda potpornih vektora. Točnost koju smo postigli našom implementacijom je oko 60% dok je točnost navedenih algoritama redom:   
 -Algoritam k-najbližih susjeda: poboljšava se sa što većim k, npr. za:   
 k=1 je točnost oko 40%,   
 k=15 je točnost oko 60%  
 k=25 je točnost oko 90%

-Slučajna šuma: točnost je oko 83%

-Metoda potpornih vektora: npr. točnost za:   
 Spline implementaciju je oko 52%  
 Polynomial implementacija je oko 78%   
 Gaussova implementacija je oko 64%

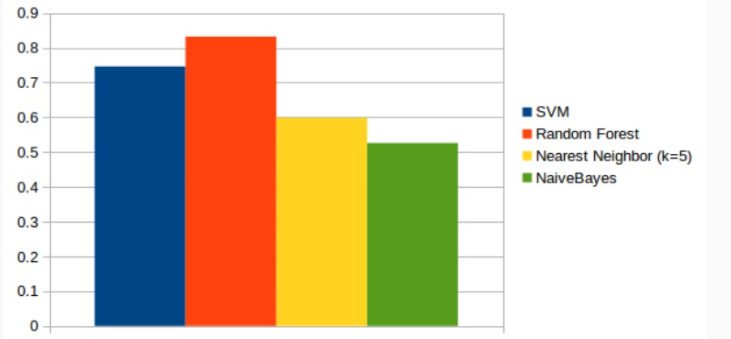


Figure 4: Prikaz točnosti različitih implementacija

# ZAKLJUČAK

Iako implementacija našeg algoritma, čija je točnost oko 60% nije najpreciznija (ne daje najbolje rezultate) daje nam strategiju odabira najboljih riječi za povećanje šansi ispunjenja zahtjeva.   
Koristeći neke riječi koje su se našim algoritmom pokazale najbolje, kao što su:  
helping, unemployed, rice, paycheck, tight, person šanse za dobitak besplatne pizze se povećavaju.  
Za razliku od njih, riječi poput: birthday, free, studying, friends, florida ne rezultiraju pretjeranim uspjehom.

Dakle, ako ste naumili dobiti besplatnu pizzu, sudeći po našem algoritmu Vaša objava bi trebala sadržavati neke od gore navedenih dobitnih riječi. Ne savjetujemo laganje, ali besplatna pizza je uvijek dobro došla. Sretno!

# LITERATURA

[1] <https://www.kaggle.com/c/random-acts-of-pizza>

[2] <https://www.reddit.com/r/Random_Acts_Of_Pizza/>

[3] <https://cs.stanford.edu/~althoff/raop-dataset/altruistic_requests_icwsm.pdf>

[4] <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

[5] <http://degiorgi.math.hr/~singer/ui/ui_1415/ch_14a.pdf>

[6] <http://degiorgi.math.hr/~singer/ui/ui_1415/ch_14b.pdf>