Univerzitet u Nišu
Elektronski fakultet

Katedra za računarstvo

Ana Tonić

Analiza sentimenata Twitter postova

Mentor:

doc. dr Miloš Bogdanović

Student:

Ana Tonić 1566

Niš, 2023

Sadržaj

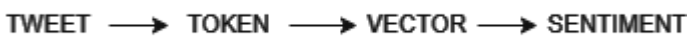
Uvod

Obrada prirodnih jezika (eng. Natural Language Processing) predstavlja skup hardverskih i softverskih alata čija je namena razumevanje i generisanje teksta pisanog prirodnim jezikom. Pod prirodnim jezicima se podrazumevaju jezici koje koriste ljudi za međusobnu komunikaciju. NLP je oblast veštačke inteligencije čiji je cilj razumevanje i generisanje nestruktuiranih tekstualnih dokumenata. Neki od zadataka obrade prirodnih jezika su sledeći: mašinsko prevođenje, sistemi za odgovaranje na pitanja, prepoznavanje govora, izdavanje informacija, analiza sentimenata itd. Prilikom kreiranja sistema za obradu prirodnih jezika, programeri se susreću sa nekim od sledećih problema: nejednoznačne rečenice, višeznačnosti reči, sinonimi itd [1]. U ovom radu predstavljen je problem klasifikacije dokumenata. Klasifikuju se tvitovi na one koji nose pozitivan i one koji nose negativan sentiment.

U ovom projektu postoje sledeće celine:

1. Pregled skupa podataka – u ovom delu izvučene su korisne informacije o podacima i vizuelizovani su podaci radi intuitivnijeg prikaza informacija i uvida u balansiranost skupa podataka
2. Normalizacija teksta – u ovom delu uklonjene su suvišne reči i znakovi, izvršena je tokenizacija i korenovanje reči
3. Vektorizacija teksta – u ovom delu izvršena je transformacija tokena u vektore brojeva kako bi se dobila adekvatna forma za ulaz u model mašinskog učenja
4. Modeliranje – u ovom delu izvršeno je treniranje modela logističke regresije na odabranom skupu podataka, evaluacija modela, i primer predviđanja na do tada neviđenom tvitu

Na slici 1 vizuelno je prikazana šema koraka u analizi sentimenata nad skupom podataka sa tvitovima.



Slika 1 Koraci u razvoju sistema za analizu sentimenata

U ovom radu su korišćene sledeće biblioteke:

- pandas – biblioteka koja služi za rad sa skupovima podataka
- numpy – biblioteka koja omogućava rad sa vektorima, matricama i drugim reprezentacijama brojeva
- matplotlib – biblioteka koja se koristi za vizuelizaciju podataka
- wordcloud – biblioteka koja prikazuje najčešće prisutne reči u pozitivnim ili negativnim tvitovima

- re – biblioteka koja omogućava formiranje regex izraza
- emoji – biblioteka koja omogućava prevođenje emotikona u tekstualnu reprezentaciju emocije koju oni predstavljaju
- contractions – biblioteka koja vraća u osnovni oblik sažmane (skraćene) reči u engleskom jeziku
- nltk – biblioteka bogata paketima za vršenje zadataka obrade prirodnih jezika kao što su tokenizacija, izbacivanje stop reči, stemovanje
- string – biblioteka koja sadrži skup znakova interpunkcije koje treba ukloniti iz tvitova
- sklearn – biblioteka koja sadrži razne module za obavljanje zadataka mašinskog učenja i nlp-a kao što su podela na trening i test skup podataka, modeli mašinskog učenja, funkcije za evaluaciju modela, vektorizacija korpusa
- seaborn – biblioteka za vizuelizaciju podataka

1. Pregled skupa podataka

Na početku rada je urađena kraća deskriptivna analiza podataka kako bi se upoznali sa podacima iz skupa podataka i zaključili koji koraci predobrade teksta su neophodni.

Skup podataka sadrži 18728 instanci.

Postoje tri kolone za svaku instancu, a to su:

- textID – jedinstveni identifikator svakog posta
- tweet_text – sam sadržaj svakog tweet-a
- sentiment – labela koja označava da li je sentiment za taj tweet pozitivan ili negativan. U ovom skupu podataka su prisutni samo pozitivni i negativni sentimenti, ne i neutralni.

Pregled prvih 10 instanci u skupu podataka dat je na slici 2.

	textID	tweet_text	sentiment
3183	1961376747	@inflight1 that sucks mama	negative
9718	1694053450	Ah, the bank holiday shift at work. What fun	negative
8991	1966125705	No waterfront anymore faccia luna and claren...	negative
7286	1964838903	Well school's finally over and idk but I'm sad...	negative
3561	1961658672	@jhaywardbenzal ouch. I hate it when that happ...	negative
3370	1961516915	I saw the sun..... but then I blink and it was...	negative
17142	1753363934	Happy mother's day everyone!	positive
12880	1695846172	LOVING the hot weather forecast for the rest o...	positive
6217	1963909495	I feel slightly sick now #BGT	negative
17903	1753617473	@bugmum oh and twas my very brilliant idea if ...	positive

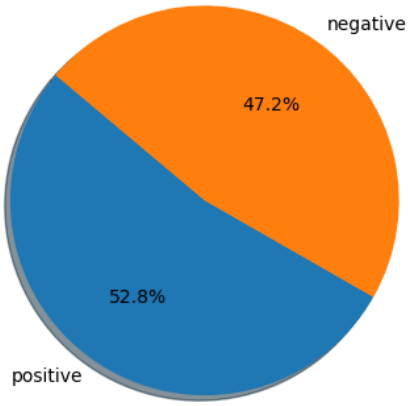
Slika 2 Prvih 10 instanci u skupu podataka

1.1 Vizuelizacija podataka

Vizuelizacija podataka nam može pomoći pri deskriptivnoj analizi podataka. Metodama vizuelizacije, podaci se predstavljaju na intuitivan način i pruža se jasan uvid u balansiranost i distribuciju podataka.

Za vizuelizaciju podataka najpre je korišćena biblioteka *matplotlib*. Ova biblioteka je jedna od najrasprostranjenijih biblioteka za vizuelizaciju i pruža veliki skup funkcija i dijagrama za vizuelizaciju. Prilikom izgradnje modela mašinskog učenja, za određene algoritme, važno je da skup podataka bude balansiran, kako bi treniranje bilo što kvalitetnije. Na slici 3 je prikazana

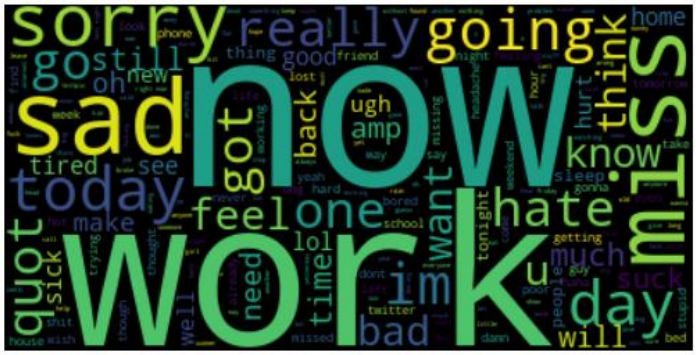
balansiranost pozitivnih u odnosu na negativne tvitove, i može se zaključiti da je ovaj skup podataka izbalansiran. U skupu podataka ima 9897 tvitova sa pozitivnim sentimentom i 8830 tvitova sa negativnim sentimentom.



Slika 3 Vizuelizacija odnosa pozitivnih i negativnih tvitova

Paket wordcloud je veoma zgodan da se brzo stekne uvid koje reči se najčešće ponavljaju u korpusu koji se obrađuje [2]. Na slici 4 je prikazano koje reči se najčešće javljaju u pozitivnim tvitovima, a na slici 5 koje u negativnim.

Slika 4 Frekventne reči u pozitivnim tvitovima



Slika 5 Frekventne reči u negativnim tvitovima

2 Normalizacija teksta

Sledeći korak u procesiranju prirodnih jezika jeste normalizacija teksta. Odstranjivanje reči koje ne donose dodatnu informaciju o sentimentu, kao i formatiranje postojećih reči spada u korak koji se naziva Normalizacija teksta.

2.1 Uklanjanje suvišnih Twitter karakteristika

Prilikom pregleda tvitova može se zaključiti da tvitovi poseduju neke reči i znakove koji ne dodaju dodatnu informaciju kao što su:

1. Oznaka za retweet
2. „@“ znak pre imena korisnika twitter-a
3. Emotikoni
4. Linkovi do web stranica
5. Oznaka za heštag

Da bismo uklonili suvišne reči i znakove potrebno je najpre da ih pronađemo u tekstu. Pronalaženje reči u tekstu vršićemo kreiranjem regularnih izraza (regex). U python-u je omogućeno kreiranje regex izraza korišćenjem biblioteke *re* [3]. Pomoću regex-a ćemo ukloniti retweet tag koji je u tvitovima prisutan kao „RT“, user tag označen sa „@“, url adresu i hashtag.

U tvitovima mogu biti prisutni razni emotikoni, koji mogu dati dodatnu informaciju i sentimentu u tvitu, ali ne u izvornom obliku u kom se nalaze. Potrebno je emotikone na neki način prevesti u tekst kako bi se dobila ta informacija koju oni sadrže. U python-u je ovo moguće izvesti pomoću paketa *emoji*. Pozivom funkcije *demojize(tweet)* se svaki emoji u prosleđenom tvitu konvertuje u svoju tekstualnu reprezentaciju.

2.2 Uklanjanje suvišnih karakteristika reči

Takođe, postoji reformatiranje koje nije vezano konkretno za tvitove već predstavlja standardne korake prilikom pripreme teksta za sledeći korak u nlp obradi. Ti standardni koraci obuhvataju sledeće:

1. Uklanjanje velikih slova
2. Ponavljanje slova
3. Ponavljanje znakova interpunkcije
4. Uklanjanje sažimanja reči, što je prisutno u engleskom jeziku (I'm -> I am)

U python-u, smanjenje velikih slova u mala, vrši se pozivom funkcije *lower*(). *Lower*() je ugrađena funkcija u python-u i koristi se za rukovanje sa stringovima. Problem ponovljenih slova i znakova interpunkcije rešava se pomoću regex-a.

U engleskom jeziku prisutna je kontrakcija odnosno sažimanje više reči u jednu reč. Osnovni primer za ovaj pojam jeste sažimanje izraza „I am" u izraz „I'm". Kako bi se rešio problem sažimanja reči potrebno je najpre instalirati paket *contranctions*. Paket contractions sadrži dictionary *contractions_dict* u kome se nalaze mapiranja za sva sažimanja u engleskom jeziku. U *contractions_dict* nalaze se parovi skraćen izraz:originalni izraz koji se koriste za oslobađanje od sažimanja u korpusu rečenica.

2.3 Tokenizacija

Tokenizacija je način da se tekst podeli u manje celine koje se nazivaju chunks. Može se zamisliti da je osnova tokenizacije podela rečenice na reči. Tokenizacija je dodatna prilika da se odabere koje reči i dodatni znakovi će se zadržati a koji neće. Prilikom tokenizacije se obično razmatra da li će se zadržati ili odbaciti reči iz sledećih grupa:

1. Znakovi interpunkcije
2. Stop reči – reči koje se svakodnevno koriste u jeziku a nemaju neko posebno značenje kao što su na primer: I, am, for…
3. Brojevi.

Ove tri grupe reči nekada treba odbaciti, a nekada treba zadržati, zavisi od slučaja do slučaja [4].

2.3.1 Implementacija tokenizacije

Za implementaciju tokenizacije u okviru programskog jezika python korišćena je *nltk* biblioteka. Nltk biblioteku nije potrebno prethodno instalirati, već je dovoljno samo importovati je. U sastavu nltk biblioteke nalazi se *nltk.tokenize* paket koji sadrži različite metode za tokenizaciju. Jedna od tih metoda je *word_tokenize(text)* koja je korišćena u ovom projektu. Ova

metoda razbija tekst na pojedinačne reči, uzimajući u obzir razmake i interpunkcijske znakove. Svaka reč postaje zaseban token.

2.3.1.1 Uklanjanje znakova interpunkcije

Svi znakovi interpunkcije mogu se pronaći u unapred definisanoj konstanti u Python-ovom modulu *string*. Ta konstanta je *string.punctuation*. Nakon primene *word_tokenize(text)* funkcije za svaki token iz rezultata se proverava da li se nalazi u nizu *string.punctuation*, i ukoliko se pronađe u nizu izbacuje se iz liste tokena koji pripadaju obrađenom tvitu.

2.3.1.2 Uklanjanje stop reči

Stop words su reči koje same po sebi ne nose informaciju. To su reči poput "the", "is", "and", "a" i slične. Procenjuje se da čine 20-30% reči u bilo kom korpusu. Ne postoji univerzalna stop words lista. U ovom projektu korišćen je paket *nltk.corpus* koji je deo *nltk* biblioteke. *nltk.corpus* sadrži kolekciju *stopwords* u kojoj se nalaze stop reči za različite jezike među kojima je i engleski jezik.

Među stop rečima u engleskom jeziku, nalazi se i reč *not*. Međutim ta reč nosi informaciju kada se vrši zadatak analize sentimenata u obradi prirodnih jezika. Stoga je iz liste stop reči koja se koristi u ovom projektu uklonjena reč stop. Nakon toga, prolaskom kroz svaki token iz tvita vrši se provera da li je ispitivana reč stop word, i ako jeste, uklanja se iz liste tokena za ispitivani tvit.

2.3.1.3 Uklanjanje brojeva i znakova koji nisu u alfabeti

U analizi sentimenata se može smatrati da brojevi ne nose informaciju o sentimentu, kao ni znakovi koji nisu delovi alfabete. Za uklanjanje ovih suvišnih znakova iz stringova korišćena je funkcija *isalpha()*. Ako string sadrži bilo koji karakter koji nije slovo, kao što su brojevi, interpunkcijski znakovi ili razmaci, funkcija *isalpha()* će vratiti False.

2.4 Stemovanje – korenovanje

Korenovanje predstavlja tehniku za svođenje reči na njihove korenske oblike. Ovaj korak je značajan u zadacima obrade prirodnog jezika jer se smanjivanjem broja reči mogu redukovati dimenzije vektora fičera, bez značajnog uticaja na kvalitet predikcije modela. Korenovanje je vrlo brza metoda, ali reči koje su rezultat korenovanja nisu nužno reči koje imaju semantičko značenje, odnosno ne dobijaju se uvek reči iz rečnika. Upravo je to razlika između korenovanja i

lematizacije [5]. Da li će za sjedinjavanje biti upotrebljeno korenovanje ili lematizacija, zavisi od konkretnog problema. U slučaju problema predikcije sentimenata, preporučuje se upotreba korenovanja.

2.4.1 Implementacija korenovanja u python-u

Što se tiče implementacije korenovanja u python-u, korišćenjem nltk biblioteke obezbeđena su tri različita algoritma za korenovanje:

- *PorterStemmer*
- *LancasterStemmer*
- *SnowballStemmer*

SnowballStemmer se smatra novijom i poboljšanom verzijom *PorterStemmer*-a, dok je *LancasterStemmer* poznat kao agresivniji algoritam koji odseca veće delove reči. Iz tog razloga će u ovom projektu biti korišćen *SnowballStemmer*.

2.5 Lematizacija

U ovom radu korišćeno je korenovanje, ali je dat kratak osvrt na lematizaciju zbog značaja i česte upotrebe u zadacima obrade prirodnoj jezika.

Lematizacija je proces koji, kao i korenovanje, služi za sažimanje sintaksno sličnih reči u jednu reč koja predstavlja osnovu sličnih reči. Ono što razlikuje ove dve tehnike jeste to što korenovanje ne daje reči sa jasnim semantičkim značenjem, a lematizacija daje. Takođe, druga razlika je u tome što lematizacija uzima u obzir kontekst u kome se reč nalazi i ne skraćuje reči uvek kao korenovanje. Uzimanje konteksta u obzir, usporava rad ovog algoritma, pa je sporiji u odnosu na korenovanje.

3 Reprezentacija teksta

Kada smo došli do ovog koraka, ono što imamo jesu reči, koje su razdvojene jedna od druge, odnosno imamo tokene. Ali, kompjuteri rade sa brojevima, a ne sa slovima. Zbog toga je esencijalni korak prilikom analize sentimenata transformacija tokena u vektore brojeva. Neke od najkorišćenijih metoda za vektorizaciju tokena su:

- Positive/Negative Word Frequencies
- Bag-of-Words
- TF-IDF

Cilj ovih metoda jeste da predstave tekst u oblik koji je razumljiv računarima i algoritmima mašinskog učenja. Cilj je ekstraktovati feature-e iz tekstova i predstaviti ih kao vektore brojeva.

3.1 Pretprocesiranje teksta

Pre vektorizacije teksta, potrebno je da izvršimo pretprocesiranje teksta. Za pretprocesiranje teksta, korišćena je biblioteka *Scikit-Learn*. *Scikit-Learn* podržava različite algoritme za klasifikaciju i klasterizaciju. Takođe, pruža mnogo korisnih alata za pretprocesiranje. Ideja jeste izmeniti skup podataka koji se koristi dodavanjem nove kolone koja sadrži tokene odgovarajućeg tvit-a. Ovaj korak je neophodan kako bi se u nastavku obrade koristila samo ova kolona. Još jedan od koraka u pretprocesiranju skupa podataka jeste kreiranje kolone koja nosi informaciju o sentimentu u binarnom obliku, umesto u tekstualnom. Nakon ovih koraka pretprocesiranja, prikazano je kako izgleda skup podataka, štampanjem prvih 10 kolona u skupu podataka, što je prikazano na slici 6.

	textID	tweet_text	sentiment	tokens	tweet_sentiment
0	1956967666	Layin n bed with a headache ughhhh...waitin o...	negative	[layin, n, bed, headach, call]	0
1	1956967696	Funeral ceremony...gloomy friday...	negative	[funer, friday]	0
2	1956967789	wants to hang out with friends SOON!	positive	[want, hang, friend, soon]	1
3	1956968477	Re-pinging @ghostridah14: why didn't you go to...	negative	[not, go, prom, bf, not, like, friend]	0
4	1956968636	Hmmm. http://www.djhero.com/ is down	negative	[hmm]	0
5	1956969035	@charviray Charlene my love. I miss you	negative	[charlen, love, miss]	0
6	1956969172	@kelcouch I'm sorry at least it's Friday?	negative	[sorri, least, friday]	0
7	1956969531	Choked on her retainers	negative	[choke, retain]	0
8	1956970047	Ugh! I have to beat this stupid song to get to...	negative	[ugh, beat, stupid, song, get, next, rude]	0
9	1956970424	@BrodyJenner if u watch the hills in london u ...	negative	[watch, hill, london, realis, tourtur, week, w...	0

Slika 6 Prvih 10 kolona nakon pretprocesiranja

Sledeći korak u pretprocesiranju jeste konverzija teksta u listu, zbog toga što su funkcije iz *Scikit-Learn* biblioteke prilagođene radu sa listama. Kreirane su dve liste, jedna za tokene, i druga za sentimente.

3.2 TF – IDF

TF – IDF je skraćenica od Term Frequency – Inverse Document Frequency. Ovo je jedna od najšire rasprostranjenih tehnika za reprezentaciju teksta.

3.2.1 Term Frequency

Prvi deo ove tehnike se predstavlja frekvenciju svake reči u svakom dokumentu u korpusu. U našem slučaju dokumenti su rečenice. Računa se kao odnos broja ponavljanja određene reči u dokumentu i ukupnog broja reči u dokumentu [6].

$$tf_{w,d} = \frac{n_{w,d}}{\sum_k n_{w,d}}$$

Primer izračunavanja TF mere je dat na slici 7. Tako je TF mera za reč „I“ u dokumentu d1 jednaka ¼, jer se pojavljuje jednom u rečenici koja ima ukupno 4 reči. Isti slučaj je i sa rečima w2, w4 i w5. Reči w3 i w6 se ne pojavljuju u d1, i zato je vrednost njihove TF mere za d1 jednaka 0.

	features		d1	d2
TWEET 1 I like my cat	I	w1	1/4	1/4
	like	w2	1/4	0
	love	w3	0	1/4
	my	w4	1/4	1/4
TWEET 2 I love my dog	cat	w5	1/4	0
	dog	w6	0	1/4

Slika 7

3.2.2 Inverse Document Frequency

Inverzna učestalost dokumenta se koristi za izračunavanje težine reči u svim dokumentima. Ovaj element ima za cilj da ograniči uticaj reči koje se često pojavljuju u dokumentima. Okreće fokus ka rečima koje se pojavljuju ređe, a koje možda nose veću informaciju o sentimentu. Računa se kao logaritam odnosa ukupnog broja dokumenata i broja dokumenata gde se određena reč ponavlja. Idf se računa na nivou reči, a ne na nivou dokumenata kao tf. Primer računanja IDF mere je dat na slici 8.

$$idf_w = \log(\frac{N}{df_w})$$

	features		IDF
TWEET 1 I like my cat	I	w1	Log(2/2) = 0
	like	w2	Log(2/1) = 0.3
	love	w3	Log(2/1) = 0.3
TWEET 2 I love my dog	my	w4	Log(2/2) = 0
	cat	w5	Log(2/1) = 0.3
	dog	w6	Log(2/1) = 0.3

Slika 8 Primer izračunavanja TF mere

Pomoću TF i IDF mera, njihovim množenjem, kreiramo konačnu TF-IDF meru, a primer je prikazan na slici 9. Dobijena TF-IDF matrica se može koristiti kao ulaz za različite modele mašinskog učenja ili druge zadatke obrade prirodnog jezika.

			TF		* IDF =	TF - IDF	
	features		d1	d2		d1	d2
TWEET 1 I like my cat	I	w1	1/4	1/4	Log(2/2) = 0	0	0
	like	w2	1/4	0	Log(2/1) = 0.3	0.075	0
	love	w3	0	1/4	Log(2/1) = 0.3	0	0.075
TWEET 2 I love my dog	my	w4	1/4	1/4	Log(2/2) = 0	0	0
	cat	w5	1/4	0	Log(2/1) = 0.3	0.075	0
	dog	w6	0	1/4	Log(2/1) = 0.3	0	0.075

Slika 9 Primer izračunavanja IDF mere

3.2.3 Primena TF-IDF mere u python-u

U okviru scikit learn biblioteke postoji paket *sklearn.feature_extraction.text* koji sadrži klasu *TfidfVectorizer* koji ćemo koristiti za izvlačenje karakteristika(feature-a) iz teksta i transformaciju teksta u vektore brojeva na osnovu tih karakteristika koje predstavljaju TF-IDF mere reči u rečenicama. Kreiraćemo *fit_tfidf* funkciju pomoću koje će biti kreiran TF-IDF vektorizator za zadati korpus dokumenata. Klasa *TfidfVectorizer* nudi metode za pretprocesiranje i tokenizaciju, koje u ovom slučaju neće biti korišćene jer će ovoj funkciji biti prosleđena lista već pretprocesiranih i tokenizovanih rečenica.

Ovako dobijena matrica koristiće se kao ulaz u model logističke regresije iz *Scikit-Learn* biblioteke. Cilj modela je da kreira sigmoidnu krivu određujući njenu jednačinu minimizacijom cost funkcije pomoću algoritma gradijentnog spusta. Izlaz iz modela jeste vektor parametara W i b. Dovođenjem novog tvita u obliku tf-idf vektora pomoću tih parametara određuje se položaj tog vektora na sigmoidnoj krivi, a taj položaj određuje verovatnoću da je tvit pozitivnog ili negativnog sentimenta. Detaljan opis modela logističke regresije je dat u sledećem poglavlju.

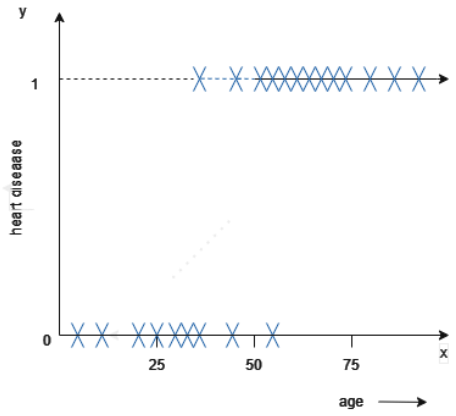
4 Model mašinskog učenja

4.1 Logistička regresija

Logistička regresija je model mašinskog učenja koji se koristi za binarnu klasifikaciju. Binarna klasifikacija je vrsta klasifikacije kod koje izlazne predikcije mogu da uzmu jednu od dve moguće vrednosti. Na primer 0/1 ili True/False. Neke primene logističke regresije u realnom svetu su:

- Da li je e-mail spam ili nije
- Da li je transakcija prevara ili nije
- Da li osoba ima neku bolest ili ne
- Da li je tumor maligni ili ne

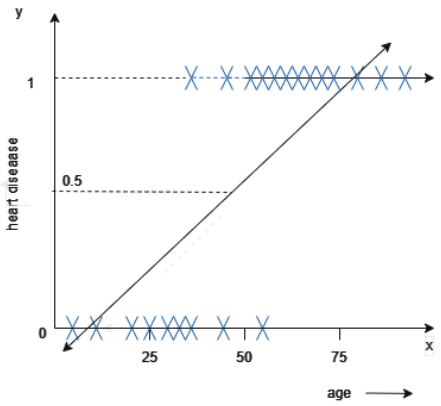
Ovo može biti razumljivije na osnovu sledećeg primera: Uzmimo primer u kome treba predvideti da li osoba ima bolest srca ili ne, na osnovu ulaznog parametra *starost(age)*. Grafik bolesti srca u odnosu na starost je ilustrovan na slici 10.



Slika 10 Grafik bolesti srca u odnosu na starost

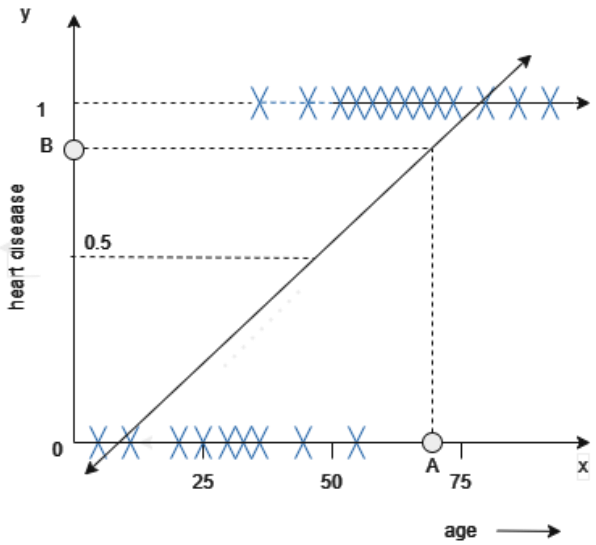
Cilj klasifikacije jeste napraviti nove predikcije na osnovu datog skupa podataka. Jedna stvar koja se može uraditi jeste da se nacrta prava linija koja odgovara ovom skupu podataka. Ta

prava linija se posmatra kao verovatnoća da li osoba ima bolest srca ili nema. Grafik na kome je iscrtana regresiona linija je predstavljen na slici 11.



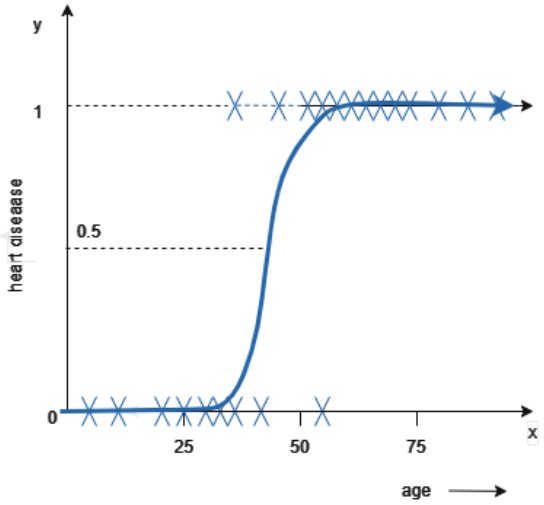
Slika 11 Regresiona linija nad posmatranim skupom podataka

Na osnovu ove prave linije može da se predvidi da li osoba ima bolest srca ili ne kada dovedemo nove ulaze. Neka je novi ulaz koji je na slici 12 predstavljen tačkom A. Vrednost koja odgovara ovoj tački na y osi je predstavljena tačkom B i to je verovatnoća da osoba ima bolest srca.



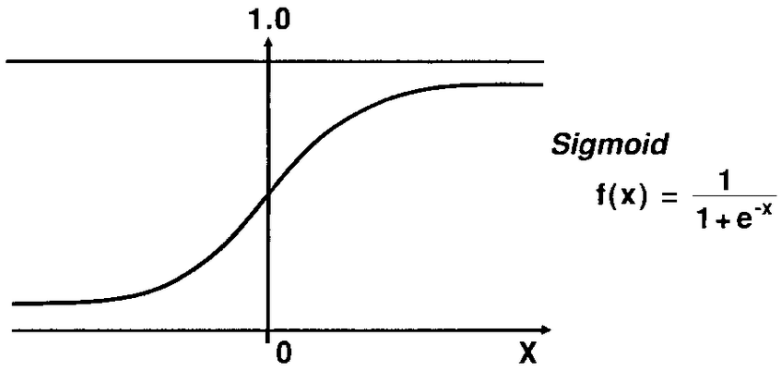
Slika 12 Primer određivanja verovatnoće bolesti srca na osnovu broja godina

Ako je verovatnoća veća 0.5 onda će izlaz klasifikatora biti 1 ili ako je verovatnoća ispod 0.5, onda će izlaz klasifikatora biti 0. Mana ovog pristupa jeste to što je mnogo podataka udaljeno od linije odnosno ovakva linija ne može da se prilagodi dobro podacima. Ovi problemi se prevazilaze korišćenjem krive koja mnogo više odgovara skupu podataka i mnogo je verovatnije da će dati tačno predviđanje. Slika krive predstavljena je na slici 13.



Slika 13 Sigmoidna kriva za posmatrani skup podataka

Ova kriva se naziva sigmoidna kriva. Izgled sigmoidne krive i sigmoidna funkcija su dati na slici 14.



Slika 14 Izgled sigmoidne funkcije i jednačine sigmoidne funkcije

Logistička regresija je u osnovi linearna regresija nad kojom je primenjena logistička funkcija. Kada se nad formulom za linearnu regresiju primeni sigmoidna funkcija dobijamo jednačinu sigmoidne krive. Matematički zapisano:

- Linearna regresija: $y = m * x + c$
- Logistička regresija: $\sigma(m * x + c)$

4.1.1 Višestruka logistička regresija

Prethodna razmatranja se odnose samo na regresiju u slučaju da imamo jednu nezavisnu promenljivu. U slučaju da imamo više atributa, jednačina višestruke linearne regresije je data sledećom formulom:

$$y = w_n x_n + w_{n-1} x_{n-1} + \ldots + w_2 x_2 + w_1 x_1 + b$$

w – težine
x – ulazni feature-i

Jednačina višestruke logističke regresije je stoga:

$$y = \sigma(w_n x_n + w_{n-1} x_{n-1} + \ldots + w_2 x_2 + w_1 x_1 + b)$$

Primenom sigmoidne funkcije nad ovom jednačinom možemo da vršimo predikcije nad bilo kojim skupom podataka sa bilo kojim brojem atributa. Takođe, prethodnu jednačinu za logističku regresiju možemo da predstavimo u formi matrice:

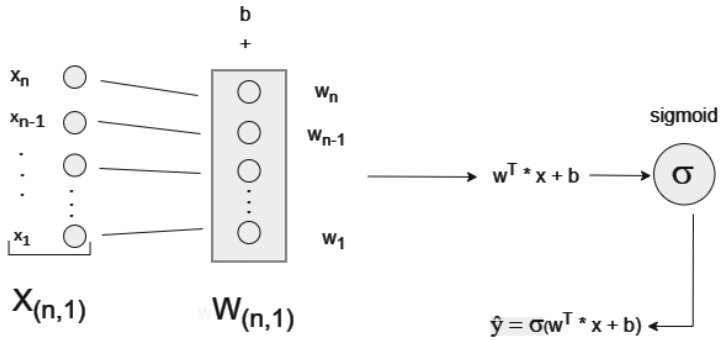
$$Y = W^T * X + b$$

U prethodnoj jednačini, W predstavlja matricu vrednosti svih atributa. i oblika je (n, 1). Izgled:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_n \end{bmatrix}_{nx1}$$

X će biti reprezentovano u formi matrice, svih ulaznih atributa što će biti izmnoženo sa W^T i sabrano sa b da bi se dobilo W^T * X + b. Sledeći korak je prosleđivanje izraza kroz sigmoidnu funkciju, čiji će rezultat biti konačne predikcije. Konačna vrednost predstavlja verovatnoću da li osoba ima bolest srca ili nema. Na slici 15 su grafički predstavljeni prethodno opisani koraci.

if $\hat{y}$ >= 0.5 then 1
if $\hat{y}$ <= 0.5 then 0



Slika 15 Višestruka linearna regresija

Sledeći korak jeste da treniranje ulaznih težina odnosno parametara tako da kriva najbolje odgovara ulaznom skupu podataka. Da bi se vršilo treniranje potrebno je koristiti cost funkciju, koja će biti opisana u sledećem poglavlju.

4.1.2 Cost funkcija kod logističke regresije

Cost funkcija u mašinskom učenju je:

- Reprezentacija greške u mašinskom učenju
- Pokazuje koliko dobro model vrši predikcije vršeći poređenje sa vrednostima u trening skupu podataka
- Cilj treniranja modela je da minimizacija vrednosti cost funkcije

Greška se može predstaviti kao apsolutna vrednost razlike predviđene i originalne vrednosti.

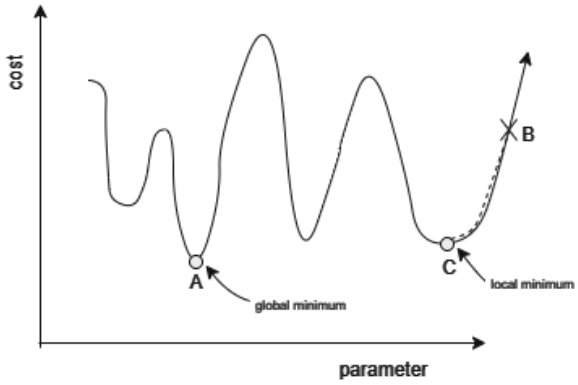
$$|y_{pred} - y|$$

Prethodni izraz predstavlja grešku za samo jednu obzervaciju. Recimo da imamo *m* opservacija. Cost funkcija predstavlja prosek vrednosti greške za sve opservacije.

$$cost = \frac{1}{m}\sum_{i=0}^{m}|y_{pred} - y|$$

Ova funkcija ima više problema. Ako je izlaz klasifikatora kontinualna vrednost onda je pogodna ova formula. Na primer izlazne vrednosti za linearnu regresiju su kontinualne vrednosti. Ali logistička regresija se koristi za binarnu klasifikaciju. Ono što se desi kada se koristi gore navedena funkcija jeste da se treniranje završi u lokalnom minimumu umesto u globalnom minimumu. Posmatrajmo cost funkciju datu grafikom na slici 16. Prikazan je grafik funkcije *cost vs parameters*. Kao što se vidi na grafiku minimalna vrednost cost funkcije se nalazi u tački označenoj sa A. Recimo da je inicijalni početak u tački B, onda je vrlo verovatno da će kao

minimalna vrednost cost funkcije biti pronađena tačka C koja predstavlja lokalni minimum umesto da završimo u tački A koja predstavlja globalni minimum. To je razlog zbog koga ova funkcija ne radi u slučaju logističke regresije.



Slika 16 Funkcija greške

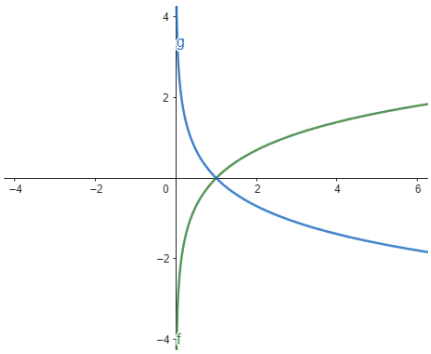
Zbog toga se koristi druga formula kao cost funkcija. Ta formula je: σ

$$cost = -\frac{1}{m}\sum_{i=0}^{m}[y * log(y_{pred}) + (1-y) * log(1-y_{pred})]$$

$$y_{pred} = \sigma(w^T * x + b)$$

4.1.2.1 Objašnjenje cost funkcije

Na grafiku na slici 17 je plavom linijom označena funkcija log(x), a pošto funkcija greške ima u izrazu ispred i znak minus, onda je predstavljena i funkcija g=-log(x), označena zelenom bojom.



Slika 17 Grafik funkcija log(x)- plava, i -log(x) zelena boja

U logističkoj regresiji y može da uzme 0 ili 1 kao svoju vrednost.

Npr. $y = 0 \mid cost = -1 * log(1 - y_{pred})$

Vrednost koju može da uzme y_{pred} je između 0 i 1. Uzmimo da je y_{pred} blizu 0, što je blizu do prave vrednosti, onda će 1-y_{pred} biti blizu do 1. I svaka vrednost logaritma od vrednosti koja je blizu do 1 ima nižu vrednost što znači da će greška biti manja. U slučaju da je y_{pred} blizu do 1, što je daleko od naše prave vrednosti, onda će 1-y_{pred} biti blizu 0, a greška će biti velika.

Za $y = 1 \mid cost = -log(y_{pred})$

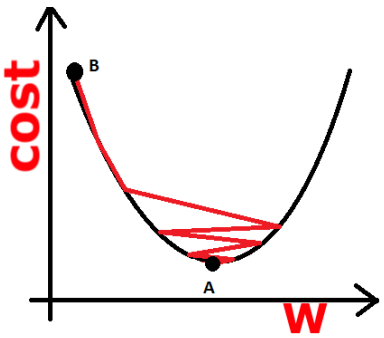
U ovom slučaju, ukoliko je y blizu do 1, što je zapravo blizu do prave vrednosti, onda će vrednost greške na osnovu jednačine biti manja. Sa druge strane ako je y_{pred} blizu do 0, što je daleko od prave vrednosti, onda će vrednost greške biti visoka.

Ovo je način na koji funkcioniše ova funkcija koja predstavlja grešku za jednu observaciju. Sledeći korak jeste minimizacija cost funkcije, a da bi seto postiglo potrebno je koristiti algoritam koji se zove algoritam gradijentnog spusta.

4.1.3 Metod gradijentnog spusta

Kako cost funkcija predstavlja reprezentaciju greške, potrebno je minimizovati grešku, tako da se tačnost poveća. Da bi se minimizovala greška, potrebno je da koristiti algoritam gradijentnog spusta.

Recimo, da ako grafički predstavimo zavisnost cost funkcije od parametara, dobijemo grafik prikazan na slici 18. Minimum funkcije leži u tački A, ali recimo da na početku algoritma krećemo od tačke B. Potrebno je da vrednost konvergira u tački minimuma, označenoj sa A. Ovo možemo postići krećući se malim koracima i eventualno oscilirati oko minimuma. Treba ažurirati w na taj način da uvek konvergira. Ako smo sa leve strane od minimuma onda se w povećava, i idemo ka minimumu, a ako smo sa desne strane onda w treba da se smanjuje da bismo došli do minimuma funkcije.



Slika 18 Zavisnost cost funkcije od parametara

To se postiže ponavljanjem koraka datim sledećim jednačinama:

$$W = W - \alpha \frac{\partial cost}{\partial w}$$

α – veoma mala vrednost pozitivne konstante koja se naziva stopa učenja

$\frac{\partial cost}{\partial w}$ – predstavlja nagib (slope) grafa

Ako smo na levom delu od minimuma nagib je negativan, tako da množimo negativnu vrednost drugom negativnom vrednošću koju dodajemo na W. A ako smo na desnoj strani od minimuma onda će nagib biti pozitivan. Tako da smanjujemo W za malu vrednost, stoga W će se smanjivati i pomeraćemo se ka minimumu i eventualno ćemo oscilirati u oblasti veoma blizu do lokalnog minimuma.

Slično, potrebno je da ažuriramo parametar b.

$$b = b - \alpha * \frac{\partial cost}{\partial b}$$

Ažuriranje oba ova parametra će voditi do minimalne vrednosti cost funkcije. Vrednosti izvoda cost funkcije po W i b su dati sledećim formulama, a kako se dolazi do njih objašnjeno je u narednom poglavlju [7].

$$\frac{\partial COST}{\partial W} = (\hat{Y} - Y) * X$$

$$\frac{\partial COST}{\partial b} = (\hat{Y} - Y)$$

4.1.4 Izvod cost funkcije za logističku regresiju

Prilikom implementacije algoritma gradijentnog spusta kod logističke regresije potrebno je da izračunamo sledeće dve stvari:

$$\frac{\partial COST}{\partial W} = ?$$

$$\frac{\partial COST}{\partial b} = ?$$

Da bismo izračunali $\frac{\partial COST}{\partial W}$, možemo da izračunamo: $\frac{\partial L}{\partial W1}$, $\frac{\partial L}{\partial W2}$, pa da onda odatle izračunamo $\frac{\partial COST}{\partial W}$., gde ∂L predstavlja izvod greške za jednu opservaciju. Zbog jednostavnosti, $\frac{\partial L}{\partial W1}$ ćemo zapisivati kao $\frac{\partial L}{\partial W}$. Da bismo izračunali $\frac{\partial L}{\partial W}$ predstavićemo ovaj izraz kao izvod složene funkcije:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial w}$$

$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w}$$

=====>

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w}$$

Prvi korak jeste izračunati $\frac{\partial L}{\partial a}$:

$$\frac{\partial L}{\partial a} = \frac{\partial(-[y * log(a) + (1-y) * log(1-a)])}{\partial a}$$

$$\frac{\partial L}{\partial a} = -\frac{y}{a} - \frac{(1-y)}{(1-a)} * (-1)$$

$$\frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$$

Drugi korak jeste izračunati $\frac{\partial a}{\partial z}$:

$$\frac{\partial a}{\partial z} = \frac{\partial((1+e^{-z})^{-1})}{\partial z}$$

$$\frac{\partial a}{\partial z} = -1 * (1 + e^{-z})^{-2} * (-1) * e^{-z}$$

$$\frac{\partial a}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2}$$

$a^2 = \frac{1}{(1+e^{-z})^2}$

$e^{-z} + 1 = \frac{1}{a}$

$e^{-z} = \frac{1}{a} - 1$

$e^{-z} = \frac{(1-a)}{a}$

$$\frac{\partial a}{\partial z} = \frac{1-a}{a} * a^2$$

$$\frac{\partial a}{\partial z} = a * (1-a)$$

Treći korak jeste izračunati $\frac{\partial z}{\partial w}$:

$$\frac{\partial z}{\partial w} = \frac{\partial(w * x + b)}{\partial w}$$

$$\frac{\partial z}{\partial w} = x + 0 = x$$

Konačno:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w}$$

$$\frac{\partial L}{\partial w} = \left[-\frac{y}{a} + \frac{(1-y)}{(1-a)} \right] * a * (1-a) * x$$

$$\frac{\partial L}{\partial w} = \frac{-y + a * y + a - a * y}{a * (1-a)} * a * (1-a) * x$$

$$\frac{\partial L}{\partial w} = \frac{a-y}{a * (1-a)} * a * (1-a) * x$$

$$\frac{\partial L}{\partial w} = (a-y) * x$$

$$\frac{\partial L}{\partial w1} = (a-y) * x$$

$$\frac{\partial L}{\partial w1} = (a-y) * x$$

.

.

$$\frac{\partial L}{\partial w1} = (a-y) * x$$

$$\frac{\partial L}{\partial W} = (A-Y) * X$$

$$\frac{\partial cost}{\partial W} = (A-Y) * X$$

Sledeće što treba izračunati je $\frac{\partial COST}{\partial b}$. Taj izraz se može preformulisati u sledeći:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial b}$$

$\frac{\partial L}{\partial a}$ i $\frac{\partial a}{\partial z}$ su poznati, tako da treba izračunati samo $\frac{\partial z}{\partial b}$.

$$\frac{\partial z}{\partial b} = \frac{\partial (w * x + b)}{\partial b}$$

$$\frac{\partial z}{\partial b} = 1$$

Iz ovoga sledi:

$$\frac{\partial L}{\partial b} = \frac{a - y}{a(1-a)} * a(1-a) * 1$$

$$\frac{\partial L}{\partial b} = a - y$$

Prethodna formula je za jednu obzervaciju. Za sve observacije:

$$\frac{\partial cost}{\partial b} = A - Y$$

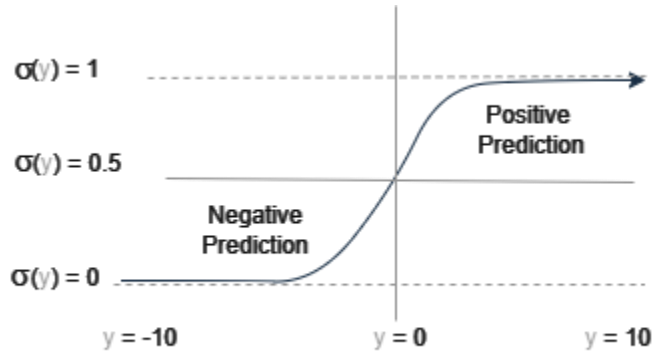
Konačni izvodi:

$$\frac{\partial cost}{\partial W} = (A - Y) * X$$

$$\frac{\partial cost}{\partial b} = A - Y$$

4.2 Primena modela

Na slici 19 je dat izgled sigmoidne funkcije a ispod slike osnovne formule logističke regresije koje su detaljno objašnjene u prethodnom poglavlju.

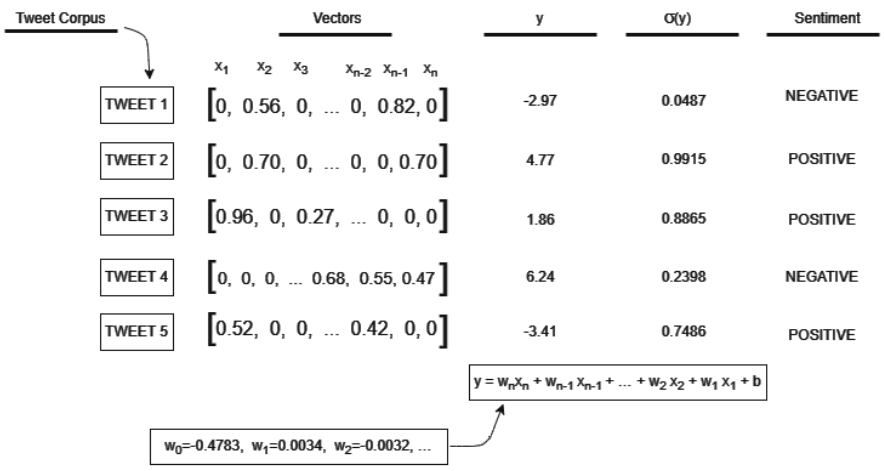


Slika 19 Primer izgleda sigmoidne krive

$$y = w_n x_n + w_{n-1} x_{n-1} + \ldots + w_2 x_2 + w_1 x_1 + b$$

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

Recimo da imamo korpus tvitova predstavljen na slici 20, gde za svaki tvit iz korpusa pokušavamo da prediktujemo sentiment. Neki tvitovi imaju pozitivan, a neki negativan sentiment. Prvi korak je da se očiste tvitovi i transformišu u vektorsku reprezentaciju. Dobijeni vektori predstavljaju set objašnjavajućih varijabli (fičera) koji mogu biti iskorišćeni za predikciju sentimenta kod tvita. Kako bi se od vektora dobila promenljiva, računa se linearna kombinacija fičera sa naučenim parametrima, označena sa y. Kada se dobije ta vrednost, onda se ona prosleđuje sigmoidnoj funkciji $\sigma(y)$. Izlaz sigmoidne funkcije predstavlja verovatnoću da se događaj desio, odnosno u našem slučaju predstavlja verovatnoću da je tvit pozitivan.



Slika 20 Predikcija sentimenata tvita nakon naučenih težina

4.2.1 Deljenje skupa podataka na trening i test skup

Pre primene modela logističke regresije, podelićemo skup podataka na trening i test podskup podataka tako da trening skup uzima 80 procenata početnog skupa, a test skup ostatak. Trening skup podataka će se koristiti kako bi se što bolje odredili parametri logističke regresije minimizacijom cost funkcije primenom algoritma gradijentnog spusta, kako je to opisano u poglavlju 4.1, a test podskup će se zatim koristiti za evaluaciju tog modela. Podela skupa podataka na ta dva podskupa ostvarena je korišćenjem *train_test_split* modula iz biblioteke scikit learn. Ono što radi ova funkcija jeste da kreira 4 nove varijable: X_train, x_test, y_train i y_test. X_train i y_train se koriste tokom trening faze, a x_test i y_test se koriste tokom test faze primene modela [8]. Nakon podele, trening skup podataka ima 14981 instancu, a test skup ima 3746 instanci podataka. Na slici 21 je dat primer kako izgleda 1 tvit iz trening skupa podataka.

```
Train tweet: ['woke', 'hour', 'sleep', 'feel', 'better']
Sentiment: 1
```

Slika 21 Primer tvita nakon pretprocesiranja i tokenizacije

4.2.2 Treniranje modela

Nakon podele na trening i test skup, cilj je istrenirati model kako bi kasnije mogao da bude korišćen za predviđanje sentimenata kod novih, do tada neviđenih tvitova. Kao što je ranije u tekstu rečeno, model logističke regresije je pogodan za probleme binarne klasifikacije kao što je analiza sentimenata. Za treniranje modela korišćen je LogisticRegression model iz *Scikit-Learn* biblioteke.

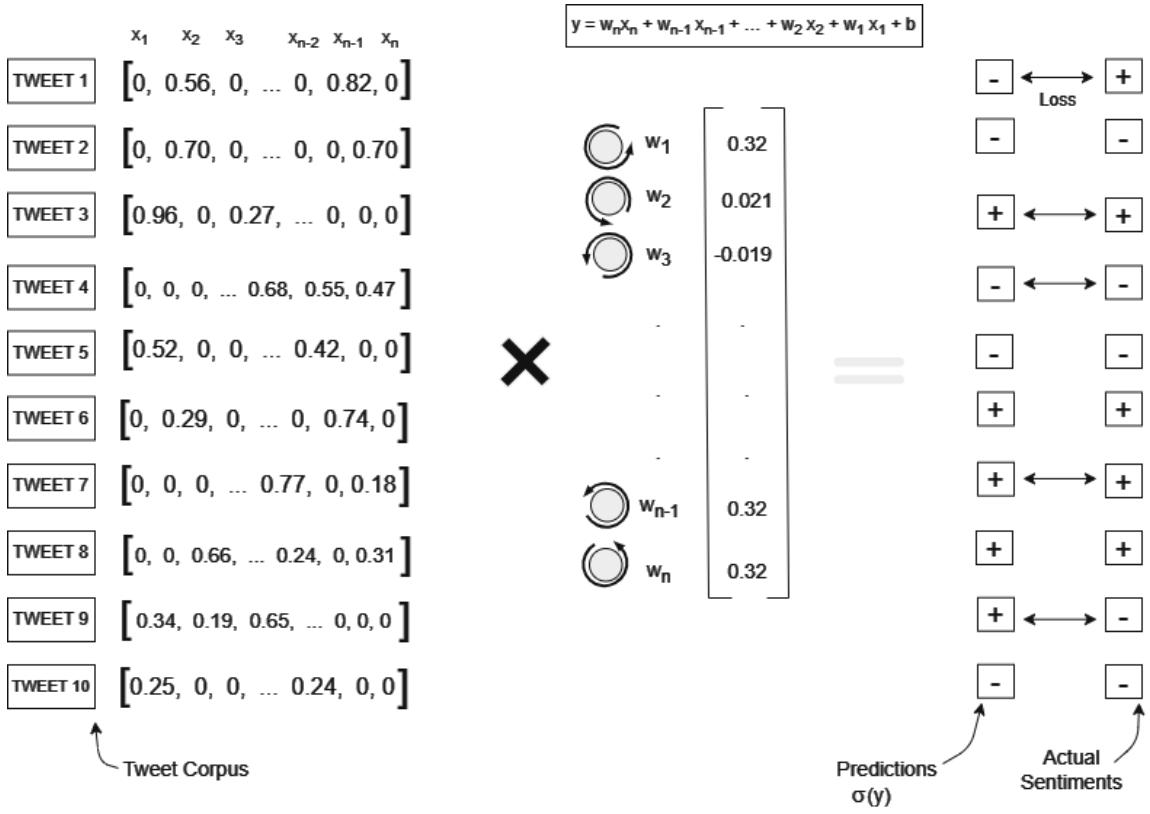
Za treniranje modela kreirana je funkcija *fit_lr(X_train, y_train)* u kojoj se kreira model i poziva treniranje modela nad trening skupom podataka prosleđenim preko parametara funkcije. Cilj treniranja je pronaći optimalne težine odnosno vrednosti parametara W i b kako bi veza između X_train i y_train bila opisana što tačnije moguće. Celokupan proces treniranja modela logističke regresije je detaljno opisan u poglavlju 4.1.

Trening skup podataka je pomoću vektorizatora transformisan u matricu oblika 14981*11744, gde prva dimenzija predstavlja broj dokumenata(rečenica),a druga dimenzija predstavlja broj jedinstvenih reči u početnom skupu podataka. Oblik ulazne matrice predstavljen je promenljivom *X_train*, vrednosti parametara vektorom *W* i promenljivom *b*, čijom linearnom kombinacijom nastaje vektor Y koji se prosleđuje sigmoidnoj funkciji

$$X_{train} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}_{nxm} \qquad W = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ w_n \end{bmatrix}_{nx1} \qquad b = single\ weight/parameter$$

$$Y = [\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot]_{1xm}$$

Na slici 22 prikazan je pipeline treniranja modela za problem klasifikacije sentimenata. Tvitove iz trening skupa podataka, za koje je poznat sentiment, transformišemo u vektore karakteristika. Sledeći korak je pronalazak optimalnih vrednosti w parametara kako bi se cost funkcija minimizovala. Minimizacija funkcije vrši se metodom gradijentnog spusta koja je objašnjena u poglavlju 4.1.3.



Slika 22 Pipeline treniranja modela za problem klasifikacije sentimenata

Nakon treniranja modela, i određivanja težina, možemo da odredimo koji tokeni imaju najveću težinu, odnosno koji tokeni najviše doprinose predikciji da je tvit pozitivan. Na slici 23 je prikazano top 10 reči koje je model prepoznao kao najznačajnije. Upoređujući te vrednosti sa vrednostima iz wordcloud sa slike 24 koji prikazuje najčešće prisutne reči u pozitivnim tvitovima mogu se na obe slike videti reči love, thank, good, hope,… što znači da je naš model pronašao očekivane reči kao najznačajnije za nošenje informacije o pozitivnom sentimentu.

```
array(['glad', 'amaz', 'nice', 'happi', 'great', 'hope', 'awesom', 'good',
       'thank', 'love'], dtype=object)
```

Slika 23 Top 10 reči koje je model prepoznao kao najznačajnije

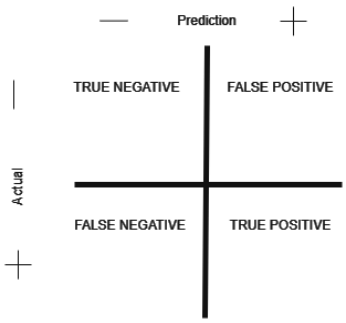
Slika 24 Frekventne reči u tvitovima sa pozitivnim sentimentom

4.2.3 Evaluacija modela

Za evaluaciju modela postoji više različitih tehnika. Ovde su korišćene matrica konfuzije i tačnost modela.

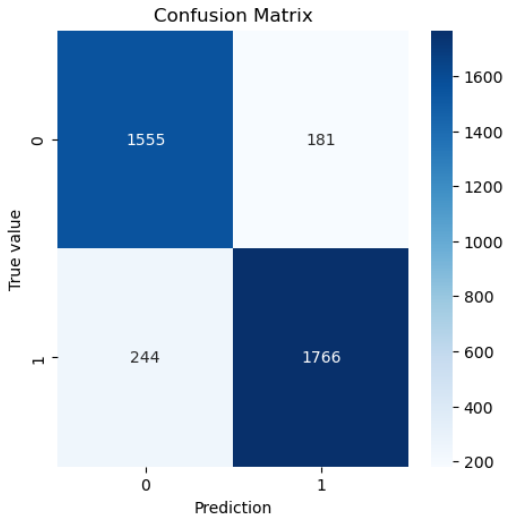
4.2.3.1 Matrica konfuzije

Matrica konfuzije predstavlja vizuelni prikaz koji je podeljen na četiri regiona na osnovu vrednosti predikcije i prave vrednosti sentimenta. Gornji levi deo predstavlja broj instanci koje su zaista negativne i za koje je model dao negativnu vrednost predikcije. Taj deo se naziva True Negative. Donji desni deo predstavlja broj instanci koje su zaista pozitivne i za koje je model dao pozitivnu vrednost predikcije. Taj deo se naziva True Positive. Zbir vrednosti ove dve kategorije predstavlja broj tačno predviđenih instanci. Donji levi ugao prestavlja oblast u kojoj se nalaze instance za koje je model dao negativnu predikciju a koje su zapravo pozitivne, i taj deo se naziva False Negative. Ovakva greška se naziva Type Two Error u statistici. Konačno, u gornjem desnom delu je predstavljen broj instanci koje su zapravo negativne, a predviđene su kao pozitivne i taj deo se naziva False Positive. U statistici, ovaj tip greške se naziva Type One Error [9]. Na slici 25 prikazana je vizuelizacija prethodno objašnjenih regiona u matrici konfuzije.



Slika 25 Izgled matrice konfuzije

Za izračunavanje matrice konfuzije u python-u korišćena je metoda confusion_matrix iz sklearn.metrics modula. Za plotovanje matrice korišćena je funkcija *heatmap* iz biblioteke *seaborn*. Na slici 26 prikazana je matrica konfuzije za test skup podataka. Sa slike se može zaključiti da model dobro predviđa kako pozitivne tako i negativne tvitove.



Slika 26 Izgled matrice konfuzije za test skup podataka

4.2.3.2 Tačnost modela

Druga mera koju ćemo koristiti za evaluaciju modela je tačnost modela. Tačnost predstavlja procenat ispravno prediktovanih tvitova od strane modela. Tačnost modela (accuracy) se definiše sledećom jednačinom:

$$Acc = \frac{TP + TN}{TP + TN + FN + FP}$$

Iako je tačnost široko rasprostranjena mera evaluacije kod problema binarne klasifikacije kao što je analiza sentimenata, ona je relevantna samo ako se radi sa izbalansiranim skupom podataka. Na primer, ako imamo skup podataka sa 90% pozitivnih tvitova i 10% negativnih tvitova. Model može lako postići visoku tačnost ako dobro predviđa pozitivne sentimente, a skoro uvek da pogrešnu predikciju za negativan sentiment. U takvim slučajevima mnogo je bolje koristiti metriku odziva (recall) umesto tačnosti. Postoji i metrika F1 score koja kombinuje tačnost i odziv u jednoj metrici [10].

Za izračunavanje tačnosti modela u python-u korišćena je funkcija *accuracy_score* iz biblioteke *sklearn.metrics*. Tačnost modela za naš test skup podataka iznosi 88.65%. U mašinskom učenju dobra tačnost je subjektivna stvar. U skladu sa industrijskim standardima, mera tačnosti između 70% i 90% nije samo idealna, već je i realna [11].

4.2.4 Predikcija novih ulaza

Da bi se vršila predikcija novih tvitova kreirana je nova funkcija *predict_tweet(tweet)* koja obavlja sledeće zadatke:

- Pretprocesiranje tvita uz pomoć ranije kreirane process_tweet(tweet) funkcije koja čisti tvit od suvišnih reči i znakova, a onda vrši tokenizaciju i steming tokena.
- Transformaciju u vektor pomoću ranije kreiranog tf vektorizatora, kako bi tvit bio konvertovan u odgovarajući oblik koji model prepoznaje
- Predikcija sentimenta tvita uz pomoć prethodno istreniranog modela logističke regresije.

Novi tvit koji je unešen je sledeći: "'RT @ColdHearted: Absolutely loved the new movie! The storyline was captivating, the acting was superb, and the visuals were stunning... Highly recommend it! #movie #entertainment'"

Izlaz iz funkcije jeste da je tvit pozitivan – slika27.

```
1 predict_tweet(tweet)
```

```
'Prediction is positive sentiment'
```

Slika 27 Određivanje sentimenta za potpuno novi tvit

Zaključak

Programski jezik python je obogaćen nizom biblioteka koje olakšavaju rad sa skupovima podataka i izvršavanje zadataka u procesu razvoja sistema za obradu prirodnog jezika. Pored biblioteka koje omogućavaju pretprocesiranje celokupnog skupa podataka, pretprocesiranje pojedinačnih tvitova, omogućene su i biblioteke koje vrše stemovanje, lematizaciju, tokenizaciju i vektorizaciju. Stoga je python pravi izbor što se tiče programskog jezika. Odabir metode za transformaciju tvitova u vektorsku reprezentaciju je takođe jedan od glavnih zadataka prilikom kreiranja modela obrade prirodnih jezika. U ovom projektu odabrano je vršenje vektorizacije korišćenjem tf-idf mere na nivou unigrama, i uz pomoć takve reprezentacije je istreniran model logističke regresije koji je dao visoku preciznost.

U ovom radu primenjena je logistička regresija kao model za analizu sentimenata Twitter postova. Cilj je bio predvideti da li je sentiment nekog posta pozitivan ili negativan. Rezultat pokazuje da je model logističke regresije postigao visoku tačnost od skoro 90% pri predviđanju sentimenta. U obrađivanom skupu podataka su prisutni tvitovi sa pozitivnim i negativnim sentimentima, stoga je za rešavanje problema binarne klasifikacije pogodan algoritam logističke regresije. U slučaju da su u skupu podataka prisutni i negativni sentimenti, ovaj algoritam ne bi bio adekvatan, te treba razmisliti o mogućnostima primene drugih algoritama mašinskog ili dubokog učenja.

Ova visoka tačnost ukazuje na efikasnost logističke regresije u razumevanju i klasifikaciji sentimenata u Twitter postovima. Model je bio u stanju da ispravno identifikuje i razlikuje između pozitivnih i negativnih sentimenta na osnovu karakteristika tekstualnih podataka. Ova sposobnost modela može biti od koristi u različitim scenarijima, kao što su praćenje javnog mnjenja, analiza korisničkih reakcija, ili prepoznavanje trendova u stvarnom vremenu. Međutim, važno je napomenuti da analiza sentimenta nije potpuno precizna nauka i da i dalje postoje izazovi u tumačenju konteksta, sarkazma, ili emocija koje mogu biti prisutne u tekstualnim podacima. Stoga, iako je postignuta visoka tačnost, dalja istraživanja i poboljšanja mogu biti potrebna kako bi se postigao još veći nivo preciznosti u analizi sentimenta.

U zaključku, potvrđeno je da logistička regresija može biti efektivan model za analizu sentimenta Twitter postova, sa skoro 90% tačnosti. Ovi rezultati pružaju osnovu za dalja istraživanja i primenu ovog modela u različitim oblastima gde je analiza sentimenta od značaja.

Literatura

[1] „Procesiranje prirodnih jezika,“ Elektronski fakultet Niš, [Na mreži]. Available: https://cs.elfak.ni.ac.rs/nastava/course/view.php?id=101. [Poslednji pristup 4 7 2023].

[2] D. Vu, „Generating WordClouds in Python Tutorial,“ datacamp, 2 2023. [Na mreži]. Available: https://www.datacamp.com/tutorial/wordcloud-python. [Poslednji pristup 7 7 2023].

[3] „re — Regular expression operations,“ Python, [Na mreži]. Available: https://docs.python.org/3/library/re.html. [Poslednji pristup 8 7 2023].

[4] A. Pai, „What is Tokenization in NLP? Here’s All You Need To Know,“ Analytics Vidia, 4 5 2023. [Na mreži]. Available: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/. [Poslednji pristup 9 7 2023].

[5] „stemming,“ TechTarget, [Na mreži]. Available: https://www.techtarget.com/searchenterpriseai/definition/stemming#:~:text=Stemming%20is%20the%20process%20of,natural%20language%20processing%20(NLP).. [Poslednji pristup 6 7 2023].

[6] K. Ganesan, „What Is Term Frequency?,“ Opinosis Analtics, [Na mreži]. Available: https://www.opinosis-analytics.com/knowledge-base/term-frequency-explained/#:~:text=TF(t)%20%3D%20(Number,of%20terms%20in%20the%20document).. [Poslednji pristup 10 7 2023].

[7] J. Patel, „Logistic Regression Machine Learning,“ Coding Lane, 5 4 2021. [Na mreži]. Available: https://www.youtube.com/playlist?list=PLuhqtP7jdD8Chy7QIo5U0zzKP8-emLdny. [Poslednji pristup 13 7 2023].

[8] C. Albon, Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning, O’Reilly Media, 2018.

[9] S. Narkhede, „Understanding Confusion Matrix,“ Towsrds Data Science, 9 5 2018. [Na mreži]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Poslednji pristup 9 7 2023].

[10] „Accuracy,“ Hasty, 17 5 2023. [Na mreži]. Available: https://hasty.ai/docs/mp-wiki/metrics/accuracy#:~:text=The%20Accuracy%20score%20is%20calculated,formula%20is%20the%20following%20one.&text=As%20you%20can%20see%2C%20Accuracy,False%20Positive%2C%20and%20False%20Negative.. [Poslednji pristup 9 7 2023].

[11] K. Barkved, „How To Know if Your Machine Learning Model Has Good Performance,“ obviously.ai, 9 3 2022. [Na mreži]. Available: https://www.obviously.ai/post/machine-learning-model-performance#:~:text=Good%20accuracy%20in%20machine%20learning,also%20consistent%20with%20industry%20standards.. [Poslednji pristup 23 8 2023].