Korištenje Pythona u analizi sentimenta

Tokić, Anamaria

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Humanities and Social Sciences / Sveučilište u Zagrebu, Filozofski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:131:524660

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

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

Filozofski fakultet
Sveučilišta u Zagrebu

Repository / Repozitorij:

ODRAZ - open repository of the University of Zagreb
Faculty of Humanities and Social Sciences

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2021./2022.

Anamaria Tokić

Korištenje Pythona u analizi sentimenta

Završni rad

Mentorica: prof.dr.sc. Tomislava Lauc

Zagreb, kolovoz 2022.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Sadržaj

Sadržaj ii
1. Uvod 1
2. Analiza sentimenta 2
2.1. Analiza sentimenta na razini dokumenta 3
2.2. Analiza sentimenta na razini rečenice 4
2.3. Analiza sentimenta na razini entiteta i aspekta 4
3. Klasifikacija teksta 5
3.1. Nadzirana klasifikacija 6
3.2. Naivni Bayesov klasifikator 7
3.2.1. Naivni Bayesov klasifikator u Pythonu 8
3.3. Stablo odlučivanja 9
3.3.1. Stabla odlučivanja u Pythonu 9
3.4. Support Vector Machine 10
3.4.1. SVM u Pythonu 12
3.5. Logistička regresija 12
3.5.1. Logistička regresija u Pythonu 14
3.6. Usporedba logističke regresije i naivnog Bayesovog klasifikatora 14
4. Opis korištenog korpusa 15
5. Korišteni alati 17
5.1. pandas 17
5.2. NumPy 19
5.3. NLTK 20
5.4. Scikit-Learn 21
6. Opis dobivenih rezultata 23
7. Zaključak 30

8. Literatura 31
Popis slika 33
Popis formula 35
Popis tablica 36
Sažetak 37
Summary 38

1. Uvod

Razvoj *World Wide Weba* je zaslužan za ogromnu količinu podataka kojom raspolažemo danas. Zanimljiv je podatak da se trenutno dnevno na internetu kreira oko 2.5 kvintilijuna bytova novih podataka.[1] Nezamisliva je ručna obrada tolike količine podataka te su se za istu razvile računalne metode. Poseban je naglasak na razvoju analize podataka i strojnog učenja, odnosno na razvoju umjetne inteligencije. Jedna od zanimljivih primjena umjetne inteligencije je upravo u polju onoga što je karakteristično za ljude, a računala ne razumiju – u polju osjećaja, emocija i mišljenja. Ovaj će se rad baviti upravo jednim od načina računalne obrade podataka iz kojih se dobije opće mišljenje veće skupine ljudi o nekom proizvodu ili usluzi – analizom sentimenta.

U sklopu ovog završnog rada bit će prikazana analiza sentimenta pomoću Pythona. Bavit će se Python bibliotekama koje se koriste u obradi teksta – *pandas, NumPy, ScikitLearn* i *NLTK*. Prikazat će se njihove metode i funkcije, tj. alati koji se koriste u analizi sentimenta te svim koracima koji joj prethode. Prikazat će se nadzirana klasifikacija i neki od najčešćih klasifikatora, te će se, primjera radi, usporediti naivni Bayesov klasifikator i logistička regresija te opisati korišteni korpus osvrta s Amazona. Za kraj će se prikazati sveukupni proces analize sentimenta te dobiveni rezultati. Sveukupni cilj ovog rada je prikazati proces analize sentimente u programskom jeziku Python, što naravno ne znači da je to jedini način provođenja analize sentimenta niti da je Python jedini jezik u kojem je to moguće.

[1] https://earthweb.com/how-much-data-is-created-every-day/#:~:text=15.1)%20Related%20Reading-,Key%20Data%20Creation%20Statistics%202022,data%20is%20created%20every%20day

2. Analiza sentimenta

Analiza sentimenta (eng. *sentiment analysis*) ili rudarenje mišljenja (eng. *opinion mining*) se bavi analizom mišljenja ljudi, sentimenta, evaluacija, procjena, stavova i emocija prema entitetima poput proizvoda, usluga, organizacija, individualaca, problema, događaja, tema i njihovih atributa (Liu, 2012). „Podrazumijeva računalnu obradu prirodnog jezika (eng. *Natural language processing -NLP*) kojom se, analitičkim pristupom, može dobiti uvid u stavove i mišljenja pojedinca"(Jakopović i Preradović, 2016). Iako dodiruje skoro sve sfere računalne obrade prirodnog jezika, analiza sentimenta je jako ograničena što se tiče obrade prirodnog jezika budući da nije potrebno razumjeti semantiku svakog dokumenta već samo neke njegove aspekte, poput negativnog ili pozitivnog sentimenta. U ovom se slučaju pod dokumentom podrazumijeva jedan entitet koji može sadržavati jednu ili više riječi, odnosno rečenica.

Cilj analize sentimenta je odrediti opću prirodi i snagu mišljenja o nekom pitanju, primjerice što ljudi misle o nekom proizvodu na društvenim mrežama (Kitchin, 2014). Informacije poput tih su korisne tvrtkama, poput agencija za oglašavanje ili tvrtki koje se bave prodajom, koje žele na vrijeme primijetiti nove trendove i reagirati pravovremeno.

Ukratko, analiza sentimenta se bavi pronalaženjem statističkih i/ili lingvističkih uzoraka u tekstu koji bi otkrili nečiji stav (Yassenov i Misailovič, 2009). Baveći se analizom sentimenta bavimo se tekstom, a zahvaljujući sveprisutnosti interneta trenutno nam ne manjka tekstualnih podataka. Web sadrži ogromne količine teksta u obliku raznih objava na društvenim mrežama, e-mailova, blogova, web stranica, opisa proizvoda, komentara i slično.

Općenito, tekstualne informacije možemo podijeliti u dvije glavne domene: činjenice i mišljenja. Dok se činjenice fokusiraju na prijenos objektivnih podataka, mišljenja izražavaju sentiment svojih autora (Yassenov i Misailovič, 2009). Liu (2010) definira činjenice kao objektivne izraze o entitetima, događajima i njihovim svojstvima, dok mišljenja definira kao subjektivne izraze koji opisuju sentimente, procjene i osjećaje koje ljudi imaju prema entitetima, događajima i njihovim svojstvima. Općenito možemo reći da su rečenice koje nose mišljenja subjektivne, a one koje navode činjenice, objektivne. Ipak, valja primijetiti da rečenice poput: „Kupila sam novi mobitel prije dva dana, a sada ne radi.", iznose činjenice, ali i implicitno iznose stavove svojih autora.

Liu (2015) sentiment definira kao temeljni osjećaj, stav, evaluaciju ili emociju pridruženu nekom mišljenju, a formalno kao trojku (y, o, i), gdje je y tip sentimenta, o orijentacija sentimenta, a i intenzitet sentimenta. Tip sentimenta ovisi o domeni u kojoj se radi analiza, dok

orijentacija može biti *negativna, pozitivna* ili *neutralna*, o čemu će biti spomena kasnije ovomu radu. Intenzitet sentimenta ovisi o tipu sentimenta te ga je najlakše objasniti primjerom – naime, usporedbom riječi poput *dobro* i *izvanredno* ili *ne voljeti* i *mrziti*, postaje očito da nemaju isti intenzitet.

Najvažniji indikatori sentimenta su riječi koje nose sentiment (eng. *sentiment words*) ili riječi koje nose mišljenja (eng. *opinion words*) (Liu, 2012). To su riječi koje se koriste kako bi se izrazio pozitivan ili negativan sentiment. Riječi poput *dobro, divno* ili *nevjerojatno* izražavaju pozitivan sentiment, a riječi poput *loše, slabo* ili *užasno* negativan. Većina riječi koje izražavaju sentiment su pridjevi ili prilozi, ali to mogu biti i imenice, poput *smeće* i *užas*, ili glagoli, poput *voljeti* i *mrziti*. Također, fraze i idiomi, poput *skupo kao suho zlato* mogu biti iskorištene kako bi se iskazao sentiment. Te riječi i fraze zajedno čine leksikon sentimenta.

Liu (2015) mišljenja dijeli na četiri komponente: nositelja mišljenja, objekt, samo mišljenje i vrijeme kada je mišljenje izraženo. Nositelj mišljenja je ono što je izvor mišljenja – osoba, organizacija ili što drugo te izravno utječe na važnost mišljenja. Očito je da mišljenje neke važne javne osobe nema istu važnost kao mišljenje nekog „običnog“ građanina. Objekt je cilj mišljenja, odnosno ono o čemu je mišljenje izraženo. Samo mišljenje predstavlja pozitivan, negativan ili neutralan stav o određenom entitetu. Vremenska je komponenta napose važna jer mišljenje koje je izrečeno jučer ili danas nije jednako relevantno kao ono izrečeno prije nekoliko mjeseci ili godina. Liu (2012) samu analizu sentimenta dijeli na tri razine, koje su objašnjene u sljedećim poglavljima.

2.1. Analiza sentimenta na razini dokumenta

Klasifikacija sentimenta klasificira dokument koji nosi mišljenje na osnovu toga izražava li pozitivno ili negativno mišljenje te se taj zadatak često naziva i *klasifikacija sentimenta na razini dokumenta* jer smatra cijeli dokument osnovnom informacijskom jedinicom (Liu, 2010). Važno je naglasiti kako se pri tom pretpostavlja da dokument zaista i sadrži mišljenje. Radeći analizu sentimenta na razini dokumenta, klasificiramo sentiment cijelog dokumenta, u smislu sadrži li, kao cjelina, negativan ili pozitivan sentiment. Radi li se o osvrtu proizvoda, analiza sentimenta na razini dokumenta će, pretpostavljajući da se cijeli korpus sastoji od osvrta na samo jedan proizvod, iznjedriti negativan ili pozitivan opći stav kupaca o tom proizvodu.

2.2. Analiza sentimenta na razini rečenice

Kako svaku rečenicu možemo promatrati kao zaseban kraći dokument, nema značajnih razlika u analizi sentimenta na razini rečenice od analize sentimenta na razini dokumenta. Međutim, u ovom slučaju izostaje pretpostavka da rečenica sadrži mišljenje, pa se uvodi dodatan korak, odnosno prvo se rečenice klasificiraju na osnovu toga sadrže li mišljenje ili ne, nakon čega se radi analiza sentimenta onih rečenica koje su klasificirane da sadrže mišljenje, odnosno analizira se sentiment samo subjektivnih rečenica.

Liu (2012) definira problem analize sentimenta na razini rečenice na sljedeći način: ako se promatra rečenica x, odredi izražavali li x pozitivno, negativno ili neutralno mišljenje. Neutralno mišljenje, u većini slučajeva, znači da osoba koja je napisala rečenicu nema stav o entitetu koji promatra.

2.3. Analiza sentimenta na razini entiteta i aspekta

Konačno, analiza na razini entiteta i aspekta provodi najprecizniju analizu. Naime, prethodno spomenute analiza na razini dokumenta i analiza na razini rečenice se ne bave time što se točno ljudima sviđa ili ne. Analiza na razini entiteta i aspekta se ne bavi jezikom, već se bavi samim mišljenjem. Temelji se na ideji da se mišljenje sastoji od negativnog ili pozitivnog sentimenta te od objekta samog mišljenja, te Liu (2012) naglašava kako je uporaba mišljenja bez identifikacije njegovog objekta, ograničena. Primjerice, u rečenici „*Iako nema veliku memoriju, sviđa mi se ovaj mobitel.*“, ton je pozitivan, ali ne možemo reći da je cijela rečenica pozitivna. Naime, osoba izražava pozitivan stav prema samom mobilnom telefonu, ali negativan prema njegovoj memoriji. U ovom su slučaju objekt i mobilni telefon i njegova memorija.

3. Klasifikacija teksta

Klasifikacija teksta je zadatak dodjeljivanja oznake ili kategorije cijelom tekstu ili dokumentu (Jurafsky i Martin, 2019). Yassenov i Misailovič (2009) navode kako postoje dva glavna pristupa klasifikaciji: nadzirani i nenadzirani. Ti su termini preuzeti iz polja strojnog učenja. Metaforično, kod nadziranog bi učenja profesor studentima dao ishode učenja zajedno s primjerima iz kojih uče, dakle postoji poznati cilj do kojega studenti trebaju doći. Isti se skup primjera može koristi i u nenadziranom učenja, no u tom slučaju profesor studentima neće dati ishode učenja već će im ostaviti da sami formiraju vlastite zaključke o tome što je danim primjerima zajedničko, odnosno o tom što je cilj učenja.

Obično su klase u koje se podatci smještaju međusobno isključive. Neke od klasifikacija su: Bayesova klasifikacija, stabla odlučivanja, umjetne neuronske mreže, *support vector machine* ili SVM i slično. Primjer klasifikacijskog pitanja bi mogao biti: „Među svim kupcima Amazona, koji će najvjerojatnije pristati na danu ponudu?“ te bi u ovomu primjeru dvije klase bile: „pristati će“ i „neće pristati.“ Klasifikacija teksta se može podijeliti u nekoliko koraka:

1. Podjela korpusa u skup za treniranje i u skup za testiranje
2. Odabir modela
3. Treniranje modela pomoću skupa za treniranje
4. Evaluacija performansi modela pomoću skupa za testiranje
5. Primjena modela na neviđenim podatcima i generiranje predikcija

U ovomu je radu fokus na jednom od zadataka klasifikacije teksta – analizi sentimenta. Klasifikacije sentimenta je obično formulirana kao klasifikacija s dvije klase - pozitivnom i negativnom (Liu, 2013). I podatci za učenje i podatci za testiranje su u ovom slučaju osvrti proizvoda. Ti osvrti uz tekstualni dio zadrže i još jedan način izražavanja mišljenja – putem brojeva, odnosno putem jedne do pet zvjezdica. Te zvjezdice uvelike olakšavaju fazu treniranja, odnosno osvrt s četiri ili pet zvjezdica se klasificira kao pozitivan osvrt, a osvrt s jednom ili dvije zvjezdice se smatra negativnim (Liu, 2010). Obično se osvrti s tri zvjezdice izostave, budući da su neutralni, odnosno uglavnom ne sadrže mišljenje. Isto će se napraviti i u ovom radu.

Tradicionalni klasifikatori teksta uglavnom klasificiraju dokumente po različitim temama, primjerice klasificiraju novinskih članaka u teme poput „politika“, „sport“ ili „tehnologija“. Kod takve klasifikacije su ključne riječi koje imaju veze s temom. No, u klasifikaciji sentimenta su važne riječi koje nose mišljenje, pozitivno ili negativno, poput *super, odlično, nevjerojatno,*

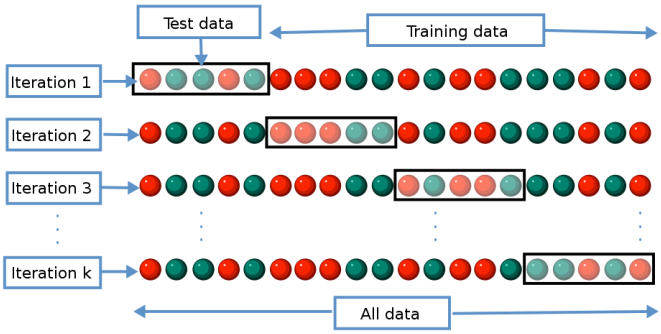
užasno, loše, najgore i slično. Kako je analiza sentimenta problem klasifikacije, bilo koja metoda nadziranog učenja može biti primijenjena.

3.1. Nadzirana klasifikacija

Nadzirana je klasifikacija zadatak odabira točne oznake klase za neki unos pri čemu se svaki unos gleda izolirano od svih ostalih unosa, a skup oznaka je definiran unaprijed (Bird, Klein i Loper., 2009). Jurafsky i Martin (2019) daju formalnu definiciju nadzirane klasifikacije: „uzmi ulaz x i fiksni skup izlaznih klasa $Y = y_1, y_2, \dots, y_m$ i vrati predviđenu klasu $y \in Y$."

Jedna od glavnih razlika nadzirane i nenadzirane klasifikacije leži u činjenici da se klasifikator kod nadzirane klasifikacije trenira na unaprijed označenim primjerima, dok tehnike nenadziranog učenja dodjeljuju oznake na osnovu samo unutarnjih razlika među podatcima (Yassenov i Misailovič, 2009). Kod nadziranog učenja model se trenira da spoji ulazne podatke s određenim poznatim izlaznim podatcima. Učenje je nadzirano u smislu da su podatci za učenje prisutni kako bi vodili proces učenja. U slučaju korpusa osvrta koji se koristi u ovom radu, osvrti su unaprijed označeni kao oni koji nose pozitivan ili negativan sentiment i to na temelju broja zvjezdica koje su korisnici ostavili uz svoje recenzije.

Tehnike nadziranog učenja dijele korpus podataka u dvije grupe – skup za treniranje i skup za testiranje. Skup za treniranje je skup podataka koji se koristi za treniranje novog klasifikatora, dok je skup za testiranje skup podataka koji se koristi za testiranje klasifikatora na podatcima koji se ne pojavljuju u skupu za treniranje, odnosno na neviđenim podatcima (Bird, Klein i Loper, 2009). Yassenov i Misailovič (2009) naglašavaju da je, u ovom koraku, najbolje raditi križnu validaciju (eng. *cross-validation*), kako bi se smanjila pristranost pri podjeli podataka na ta dva skupa. U križnoj validaciji se korpus podijeli na N skupova, a proces klasifikacije se ponavlja isto toliko puta tako da se podatci iz jedne grupe koriste za treniranje, a iz druge za testiranje. Na slici 1 se nalazi shema križne validacije s N grupa. Rezultata klasifikacije je srednja vrijednost rezultata jedne grupe, a klasifikacija je utoliko pouzdanija što rezultati svake grupe daju sličnije rezultate. U ostatku ovoga poglavlja će se pobliže prikazati neki od nadziranih klasifikatora – naivni Bayesov klasifikator, stablo odlučivanja, stroj potpornih vektora te logistička regresija.



Slika 1. Shema križne validacije (Izvor: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.svg)

3.2. Naivni Bayesov klasifikator

Naivni Bayesov klasifikator je statistički, odnosno probabilistički klasifikator. Ime je dobio po Thomasu Bayesu, engleskom svećeniku i matematičaru (Bramer, 2016). To da je probabilistički klasifikator znači da će za dokument d od svih klasa $c \in C$, izabrati onu klasu c koja ima najveću aposteriornu vjerojatnost (Jurafsky i Martin, 2019). Bayesov algoritam nam omogućava da jednom formulom izračunamo vjerojatnosti svake klase te odaberemo onu s najvećom vjerojatnošću (Bramer, 2016).

Naivni Bayesov klasifikator je baziran na Bayesovom teoremu:

$$p(y|X) = \frac{p(X|y) * p(y)}{p(X)}$$

Jednadžba 1. Bayesov teorem

Iz Bayesovog se teorema izvodi naivni Bayesov klasifikator, kojemu je cilj izabrati klasu y s najvišom vjerojatnošću. Operacija argmax, koja se može vidjeti u jednadžbi 2 pronalazi onaj argument koji daje maksimalnu vrijednost ciljne funkcije, a u ovom slučaju je cilj pronaći maksimalnu vrijednost varijable y:

$$y = argmax_y[P(y) * \prod_{i=1}^{n} P(x_i|y)]$$

Jednadžba 2. Naivni Bayesov klasifikator[2]

Riječ *naivna* u nazivu ove klasifikacije se referira na činjenicu da ova metoda pretpostavlja da je utjecaj koji jedan atribut ima na vjerojatnost neke klase neovisan o utjecajima drugih atributa (Bramer, 2016), što je nerazumno za pretpostaviti, posebno s obzirom na to da gotovo svi

[2] https://towardsdatascience.com/a-mathematical-explanation-of-naive-bayes-in-5-minutes-44adebcdb5f8

problemi iz stvarnoga svijeta sadrže osobine s raznim stupnjevima ovisnosti jednih o drugima (Bird, Klein i Loper, 2009). Ipak, mnogi autori navode da unatoč mnogim teoretskim slabostima, naivni Bayes daje dobre rezultate u praksi.

Naivni Bayesov klasifikator se oslanja na prikazivanje dokumenta kao *bag-of-words*, odnosno se oslanja na *princip vreće riječi*. Riječima se ignorira pozicija u rečenici, već se gleda samo njihovu frekvencija, odnosno broj pojavljivanja u dokumentu (Medium, 2018).

3.2.1. Naivni Bayesov klasifikator u Pythonu

Naivni Bayesov klasifikator se u Python može pozvati ili iz modula *nltk* ili iz modula *sklearn*. U nltk se podmodul zove *NaiveBayesClassifie*r, a u *sklearn naive_bayes*. Na slici 2 je prikazan primjer klasifikacije u *nltk*-u pomoću naivnog Bayesa. Metoda *.train()* na temelju skupa za treniranje uči klasifikator. Klasificira li se samo jedna instanca, koristi se metoda *.classify()*, a za klasifikaciju više instanci metoda *classify_many()*. Na slici 3 je prikazan primjer klasifikacije pozivanjem naivnog Bayesovog klasifikatora iz modula *sklearn*. Važno je primijetiti da je pri korištenju biblioteke *sklearn* potrebno prvo definirati klasifikator, u ovom slučaju kao *MultinomialNB()*. Osim *MutinomialNB() sklearn* sadrži još i *GaussianNB* - Gaussov naivni Bayesov algoritam za klasifikaciju; *BernoulliNB* – za podatke koji su distribuirani po Bernoullijevoj distribuciji; *CategoricalNB* – za podatke koji su kategorično distribuirani; i još nekolicinu implementacija naivnog Bayesovog klasifikatora. Metoda *.fit()* će prilagoditi model podatcima da omogući što bolja pouzdanost. Metoda *.predict()* predviđa oznake testnih podataka.

```
from nltk.classify import NaiveBayesClassifier
klasifikatorNB = NaiveBayesClassifier.train(skup_treniranje)
klasifikatorNB.classify(neoznacena_instanca) #za jednu neviđenu instancu
klasifikatorNB.classify_many(neoznacene_instance) #za više neviđenih instanci
```

Slika 2. Naivni Bayesov klasifikator u nltk

```
from sklearn import naive_bayes
klasifikatorNB = naive_bayes.MultinomialNB()
klasifikatorNB.fit(podatci_treniranje, oznake_treniranje)
oznake_predvidjene = klasifikatorNB.predict(podatci_test)
```

Slika 3. Naivni Bayesov klasifikator u sklearn

3.3. Stablo odlučivanja

Stablo odlučivanja je klasifikacijska metoda u kojoj se klasifikacijski proces modelira korištenjem skupa hijerarhijskih odluka na varijable osobina, raspoređeni u strukturu nalik stablu (Aggarwal, 2015). Stablo odlučivanja kreira klasifikator baziran na dijagramu toka. Na svakoj razini upotrebljava jednostavan klasifikator koji provjerava prisutnost određene osobine, a oznaka se dodjeljuje rečenici na lisnom čvoru stabla (Yassenov i Misailovič, 2009). Koristi se metafora obitelji za objašnjavanje veze između čvorova u stablu (Bird, Klein i Loper, 2009), te se korijenski čvorovi nazivaju „roditelji“, a listovi su „djeca“. Pri odabiranju oznake za ulaznu vrijednost počinje se na vrhu od inicijalnog čvora, odnosno korijenskog čvora, koji sadrži uvjet koji provjerava jednu od osobina ulazne vrijednosti i odabire granu ovisno o vrijednosti te osobine. Prateći tu granu prema dnu, dolazi se do novog čvora odluke, odnosno internog čvora, s novim uvjetom za osobine ulazne vrijednosti te nastavlja slijediti grane koje su odabrane uvjetom svakog čvora sve dok se ne dođe do lisnog čvora koji daje oznaku za ulaznu vrijednost. Cilj je napraviti model koji predviđa vrijednost ciljne varijable učeći jednostavna pravila odlučivanja koja su izvedena iz osobina podataka (scikit, 2022).

3.3.1. Stabla odlučivanja u Pythonu

U Pythonu klasa *DecisionTreeClassifier* ima mogućnost izvršavati više-klasnu klasifikaciju na skupu podataka. Kao ulaz uzima dva *array*-a, X i Y. *Array* X veličine [n_uzoraka, n_osobina] sadrži uzorke za treniranje, a *array* Y vrijednosti cijelih brojeva, veličine [n_uzoraka] sadrži oznake klasa za uzorke za treniranje (scikit, 2022). Sljedeći primjer na slici 4 prikazuje kako pozivamo stablo odluka u Python skriptu:

```
from sklearn import tree
klasifikator = tree.DecisionTreeClassifier()
klasifikator = klasifikstor.fit(podatci_treniranje, oznake_treniranje)
```

Slika 4. Stablo odlučivanja u sklearn

Nakon što model prilagodimo podatcima s *.fit()* metodom, možemo ga koristiti za predikcije klasa uzoraka i to pomoću *predict()* metode ili možemo predvidjeti vjerojatnost svake klase s *predict_proba()* metodom. *DecisionTreeClassifie*r može vršiti i binarnu klasifikaciju gdje su

oznake [-1,1] i klasifikaciju s više oznaka, gdje su oznake [0,…K-1] (scikit, 2020). Stablo odlučivanja možemo i grafički prikazati pomoću *plot_tree()* metode.

3.4. Support Vector Machine

Support vector machines ili SVM (hrv. metoda potpornih vektora) je klasifikacijski algoritam koji je izveden iz teorije statističkog učenja Vladimira Vapnika i njegovih kolega 1992. godine. SVM je diskriminativni klasifikator. Umjesto da modelira svaku klasu kao navni Bayesov klasifikator, SVM pronađe liniju, ili krivulju u slučaju dvije dimenzije, ili razdjelnicu u slučaju više dimenzija, koja odvaja klase jednu od druge (VanderPlas, 2017). SVM je prirodno definiran za binarnu klasifikaciju numeričkih podataka ali se taj problem binarnih klasa može poopćiti za slučajeve s više klasa (Aggarwal, 2015). Klasifikacijska funkcija SVM-a je bazirana na konceptu ravnina odlučivanja koje definiraju granice odlučivanja između klasa (Katardzic, 2011).

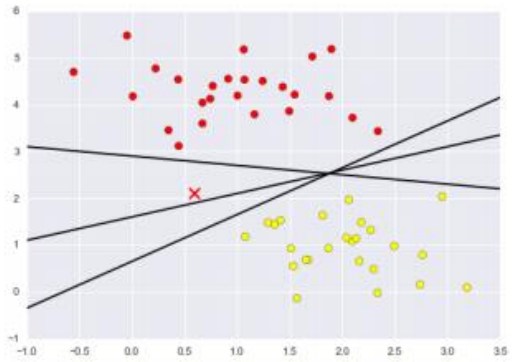
SVM podatke za učenje predstavlja kao točke u prostori tako da se podatci koji pripadaju različitim klasam mogu razdvojiti širokim razmakom, koji se zove hiper ravnina, a novi se podatci klasificiraju na osnovu toga na koju stranu te ravnine padnu (Sarkar, 2016). To je tipični proces linearne klasifikacije, no SVM može izvoditi i nelinearnu klasifikaciju, pomoću jezgrenih funkcija, što prelazi granice ovoga rada.

Sarkar (2016) daje matematičku definicija SVM-s:

> „Skup podataka za učenje se sastoji od n podatkovnih točaka $(\overrightarrow{x_1}, y_1)$, ...,$(\overrightarrow{x_n}, y_n)$ tako da je varijable klasa $y_i \in \{-1, 1\}$ gdje svaka vrijednost ukazuje na klasu koja odgovara točki $\overrightarrow{x_i}$. Svaka podatkovna točka $\overrightarrow{x_i}$ je vektor osobine, a cilj SVM algoritma je pronaći hiper ravninu maksimalne margine koja odvaja skup podataka koji imaju oznaku klase $y_i = 1$ od skupa podataka koji imaju oznaku klase $y_i = -1$, tako da je udaljenost od hiper ravnine i uzoraka podatkovnih točaka od najbliže klasa maksimalna. Ti uzorci podatkovnih točaka se onda nazivaju potpornim vektorima."

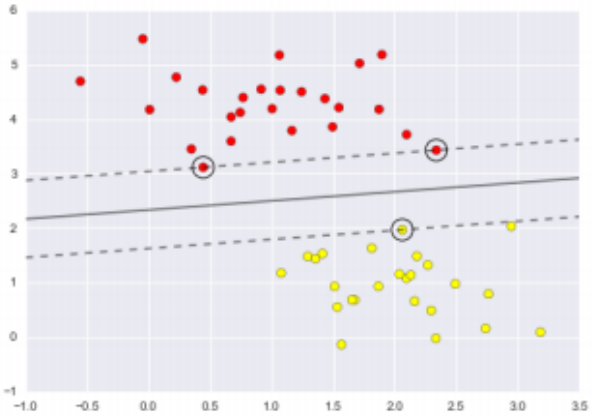
Na slici 5 su vidljivi linearno odvojivi podatci, odnosno najjednostavniji slučaj za korištenje SVM-a. Kod linearno odvojivih podataka moguće je konstruirati linearnu hiper ravninu koja jasno odvaja podatke u dvije klase (Katardzic, 2011), što je prilično neobičan slučaj, jer su podatci iz realnog svijeta rijetko potpuno odvojivi. Često se u literaturi termin hiper-ravnina koristi kako bi se referiralo na opću razdvajajuću površinu (eng. *general seperating surface*) ili

na generalizaciju linije ili ravnine. Granice odlučivanja (engl. *decision boundaries*) ili općenitije površine odlučivanja (eng. *decision* surfaces) su linije koje razdvajaju dva područja (Provost i Fawcet, 2013). Ipak, na slici 5 primjećujemo da postoji više načina na koje se može linearno odvojiti dvije klase podataka - na slici su dvije klase predstavljene kao crvene i žute točke, odnosno postoji beskonačno mnogo granica odluka te ovisno o tome koju liniju odaberemo novi će podatak - označen crvenim slovom x, dobiti drugu oznaku. Te linije nazivamo linearnim diskriminantama i, kao što je već spomenuto, postoji ih beskonačno mnogo između dvije klasa.



Slika 5. Linearno odvajanje dviju klasa (izvor: VanderPlas, 2017)

Međutim, umjesto samo povlačenja linija, oko svake linije možemo nacrtati marginu do najbližeg podatka. Hiper ravnina s maksimalnom marginom je ona koja razdvaja dvije klase na način da postoji veliko područje, odnosno margina, na svakoj strani granice u kojemu se ne nalazi niti jedan podatak za treniranje.



Slika 6. Margina (izvor: VanderPlas, 2017)

Na slici 6 se može primijetiti da neki podatci leže točno na margini - ti se podatci nazivaju potpornim vektorima (eng. *support vectors*). Ključ uspjeha ovog klasifikatora je u tome da je za prilagodbu modela podatcima važna samo pozicija potpornih vektora, a bilo koji podatak koji je na pravoj strani margine ne utječe na prilagodbu. Razlog tomu je u činjenici da ti podatci ne doprinose funkciji gubitka (engl. *loss function*) koja se koristi kako bi se model prilagodio podatcima, pa njihova poziciji i količina nisu važni sve dok ne prelaze marginu (VanderPlas, 2017). Funkcija gubitka određuje kolika bi penalizacija trebala biti dana instanci bazirano na grešci u vrijednosti koju je predvidio model, odnosno bazirano na njenoj udaljenosti od granice razdvajanja (Provost i Fawcet, 2013).

3.4.1. SVM u Pythonu

U Pythonu se SVM i njegove varijacije nalaze u *scikit learn* biblioteci. *SVC*, *NuSVC* i *LearnSVC* su varijacije SVM-a, odnosno klasifikatori sposobni za binarnu i više-klasnu klasifikaciju (scikit, 2020). Na početku se poziva, odnosno uvozi *svm* iz *sklearn* biblioteke kao što se vidi na slici 7. Također, pri korištenju SVM-a je potrebno definirati željeni klasifikator, pomoću primjerice *.SVC()* metode.

```
from sklearn import svm
clfrSVM=svm.SVC(kernel='linear', C=0.1)
clfrSVM.fit(train_data, train_labels)
predicted_labels=clfrSVM.predict(test_data)
```

Slika 7. SVM u sklearn

SVC i *NuSVC* su slične metode, ali uzimaju različiti skup parametara i imaju drugačije matematičke formulacije. *LinearSVC* je brža implementacija klasifikacije potpornih vektora (scikit, 2022). Kao i drugi klasifikatori, i ova tri kao ulaz uzimaju dva *array*-a: *array x* oblika (n_uzoraka, n_osobina) koji sadrži uzorke za treniranje, i *array y* oznaka klasa, oblika (n_uzoraka). Za predikcije oznaka novih instanci se koristi *predict()* metoda, a za informacije o svojstvima potpornih vektora, koriste se atributi *.support_vectors_*, *.support_* te *.n_support_*.

3.5. Logistička regresija

Logistička je regresija probabilistički klasifikator koji koristi nadzirano strojno učenje (Jurafsky i Martin, 2019). Ime *logistička regresija* bi se moglo navesti na zaključak da je

logistička regresija regresijski algoritam, no nije, ona je klasifikacijski algoritam. Predstavio ju je 1958. godine D.R. Cox, a razlog njenog pomalo zbunjujućeg imena leži u činjenici da se u polju statistike smatra regresijskim algoritmom, a polje strojnog učenja ju je preuzelo i počelo koristiti kao klasifikacijski algoritam (Skansi, 2018). Razlika regresije i klasifikacije leži u vrijednosti ciljne varijable – kod regresija je ona kategorička, a kod klasifikacije numerička. Naime, kod logističke regresije, model proizvede numeričku procjenu, ali su vrijednosti ciljne varijable u podatcima kategoričke. Provost i Fawcet (2013) naglašavaju da je debata zapravo samo akademska, a da je važnije razumjeti što zapravo logistička regresija radi. Ona procjenjuje vjerojatnosti pripadnosti klasi - numeričku količina, preko kategoričke klase. Može se koristiti s dvije klase – primjerice pozitivan i negativan sentiment, ili s više klasa, pa ju onda nazivamo multinomialnom logističkom regresijom. Ovaj se rad bavi samo binarnom logističkom regresijom, jer analiza sentimenta zahtjeva samo dvije klase.

Logistička regresija modelira vjerojatnost da se neki događaj dogodi, kao linearnu funkciju skupa varijabli prediktora (Kantardzic, 2011). Ona ne predviđa vjerojatnost varijabli, već procjenjuje vjerojatnost da varijable imaju određenu vrijednost. Za razliku od, primjerice, naivnog Bayesa koji procjenjuje pripada li neka instanca jednoj ili drugoj klasi, logistička regresija izravno izračunava vjerojatnost pripadnosti jednoj klase. Poput drugih klasifikatora ima dvije faze - fazu učenja u kojoj trenira sustav te fazu testiranju u kojoj se izračunava vjerojatnosti i vraća oznaku s najvećom vjerojatnošću.

Ovaj se rad bavi najjednostavnijim oblike logističke regresije, binarnom logističkom regresijom. Za nju se pretpostavlja da je varijabla klase binarna i iz skupa {-1, +1}. Aggarwal (2015) daje matematičku formalnu definiciju logističke regresije:

> „Neka je $\bar{\theta} = (\theta_0, \theta_1, \ldots, \theta_d)$ vektor $d + 1$ različitih parametara. i-ti parametar θ_i je koeficijent povezan s i-tom dimenzijom izvornih podataka, a θ_0 je parametar pomaka. Onda za zapis $\bar{X} = (x_1 \ldots x_d)$ vjerojatnost da varijabla klase C prima vrijednosti -1 ili +1 je modelirana pomoću logističke funkcije:
>
> $$P(C = +1|\bar{X}) = \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$
>
> $$P(C = -1|\bar{X}) = \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$
>
> Suma ovih dviju vrijednosti, $P(C = +1|\bar{X})$ i $P(C = -1|\bar{X})$ je 1.“

3.5.1. Logistička regresija u Pythonu

U Python se logistička regresija poziva iz modula *sklearn*, odnosno iz podmodula *linear_model*. Klasifikator se pozove funkcijom *LogisticRegression()* te je daljnji postupak uglavnom identičan onomu drugih klasifikatora.

```
from sklearn.linear_model import LogisticRegression
klasifikatorLR = LogisticRegression()
klasifikatorLR.fit(podatci_treniranje, oznake_treniranje)
```

Slika 8. Logistička regresija

3.6. Usporedba logističke regresije i naivnog Bayesovog klasifikatora

Kao što je prethodno već spomenuto, i naivni Bayes i logistička regresija su probabilistički klasifikatori. Njihova najveća razlika leži u činjenici da je logistička regresija diskriminatorni, a naivni Bayes generativni klasifikator (Jurafsky, Martin, 2019). To znači da dok naivni Bayesov klasifikator pretpostavlja specifičan oblik distribucije vjerojatnosti osobina za svaku klasu, logistička regresija izravno modelira vjerojatnosti pripadnosti klasi u smislu varijabli osobina s diskriminatornom funkcijom (Aggarwal, 2015). U analizi će sentimenta to značiti da naivni Bayes računa vjerojatnost pripadnosti neke instance klasama 1 i 0 te odabiru onu klasu s većom vjerojatnosti, dok logička regresija računa vjerojatnost da instanca pripada klasi 1, pa ako je vjerojatnost veća od 0.5 daje predikciju oznake 1, inače 0. Dakle, najveća razlika leži u samoj pretpostavci modeliranja.

4. Opis korištenog korpusa

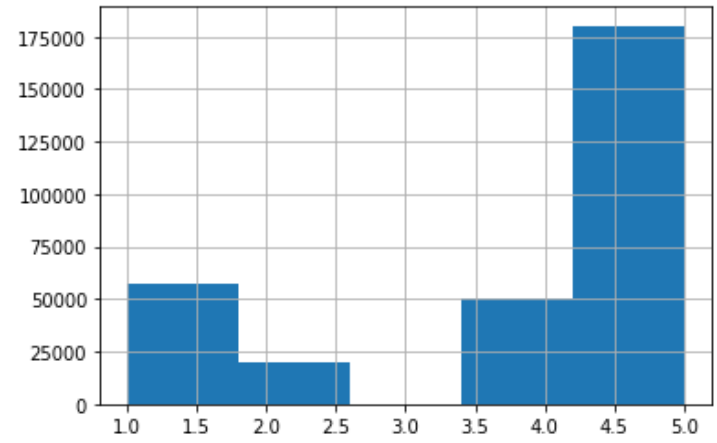
Korpus koji je korišten u ovome radu je korpus osvrta s jedne od najvećih svjetskih Internet trgovine, Amazona[3]. Korpus je u .csv formatu, odnosno u svakom se redu nalazi po jedan osvrt i njemu pripadajući podatci tako da su vrijednosti, koje bi se inače nalazile u posebnim stupcima, odvojene zarezom. Podatci o pojedinom proizvodu su podijeljeni u šest kategorija: 'Product Name', 'Brand Name', 'Price', 'Rating', 'Reviews', 'Review Votes'. Za ovaj su rad posebno važne kategorije 'Rating' i 'Reviews', a na osnovu vrijednosti u kategoriji 'Rating' je kasnije dodana i nova kategorija 'Positively Rated', koja je poprimila vrijednost 1, ako je vrijednost toga reda u stupcu 'Rating' 4 ili 5. Oni osvrti koji u stupcu 'Positively Rated' imaju vrijednost 1 se smatraju pozitivnima. Kategorija 'Positively Rated' poprima vrijednost 0, ako je vrijednost toga reda u stupca 'Rating' 1 ili 2, odnosno se u ovom slučaju radi o negativnim osvrtima. Oni osvrti koji u stupcu 'Rating' imaju vrijednost 3 su već na početku izbačeni. Razlog tomu je taj što se osvrti koji su ocijenjeni s ocjenom 3, odnosno s 3 zvjezdice, smatraju neutralnima. Korpus na početku broji 413826 osvrta, no izbačeni su osvrti u kojima nedostaje neki podatak te oni koji se smatraju neutralnima. Konačna se struktura korpusa može vidjeti u tablici 1.

Početni broj osvrta	Konačan broj osvrta	
413826	308277	
	Broj pozitivnih osvrta	Broj negativnih osvrta
	230674	77603

Tablica 1. Konačna struktura korpusa

U tablici 1 vidimo da je znatno više pozitivnih osvrta nego negativnih. Na histogramu na slici 9 se može vidjeti točna struktura korpusa po kategoriji Rating s izbačenim neutralnim osvrtima, a na slici 10 prvih pet osvrta u korpusu.

[3] Korpus je preuzet sa: https://www.coursera.org/learn/python-text-mining/resources/d9pwm



Rating	Product Name	Brand Name	Price	Reviews	Review Votes
1	57535	57535	57535	57535	57535
2	20068	20068	20068	20068	20068
4	50421	50421	50421	50421	50421
5	180253	180253	180253	180253	180253

Slika 9. Histogram

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	It works good but it goes slow sometimes but i...	0.0
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	Great phone to replace my lost phone. The only...	0.0

Slika 10. Prvih pet redova korpusa

5. Korišteni alati

Praktični je dio ovog završnog rada napisan u programskom jeziku Python. Python je opći i višenamjenski programski jezika koji sadrži velik broj standardnih biblioteka i modula za rješavanje raznih problema (Sarkar, 2016). Pripada skupini objektno orijentiranih programskih jezika, poput C++ ili Jave, ali i skupini skriptnih jezika, poput Perl-a, koji se mogu koristiti za brzo pisanje malih programa ili skripti kojima se automatiziraju drugi zadatci, no to ne znači da se Python i slični programski jezici ne mogu koristiti za stvaranje velikih software-a. Postoje dvije osnovne verzije programskog jezika – Python 2.X i 3.X. Python 2.X vremenom izlazi iz uporabe te je nedavno i verziji 2.7 prestala potpora. Ovaj je rad rađen u verziji 3.9 u distribuciji Anaconda[4], odnosno u Jupyter Notebook-u[5].

Anaconda je besplatna i otvorena distribucija programskih jezika Python i R za znanstveno računalstvo. Popularna je za korištenje jer jednom instalacijom donosi većinu alata koji se koriste u podatkovnoj znanosti i strojnom učenju. Anaconda sa sobom donosi i Jupiter Notebook – web aplikaciju otvorenog koda koja omogućava kreiranje i dijeljenje dokumenata koji sadrže kod, jednadžbe, vizualizaciju i narativni tekst. Jupyter Notebook kod sprema u datoteke s ekstenzijom .ipynb, odnosno u tekstualnu datoteku koja opisuje sadržaj bilježnice koja sadrži kod u formatu JSON.

Kombinacija podrške za biblioteke poput *Pandas* ili *Scikit-learn* i općenito snage u razvoju *software*-a za opće namjene, je učinila Python odličnom opcijom za razvoj podatkovnih aplikacija (McKinney, 2018). Pored standardnih biblioteka, na Internetu možemo pronaći niz biblioteka treće strane. Biblioteke treće strane koje Python podržava se nalaze u repozitoriju *Python Package Index* (PyPI)[6]. Tamo se mogu pronađi i sve biblioteke koje se obrađuju u ovome radu. Python ima stotine biblioteka treće strane, specijaliziranih softverskih paketa koji mu proširuju primjenu. Ovdje će se spomenuti samo neke biblioteke te će se objasniti one njihove metode i funkcije koje su korištene.

5.1. Pandas

Pandas[7] je biblioteka otvorenog koda koja omogućava visoke performanske, strukture podataka koje su lake za korištenje te alate za analizu podataka za programski jezik Python

[4] https://www.anaconda.com/products/individual
[5] https://jupyter.org/
[6] https://pypi.org/
[7] https://pandas.pydata.org/docs/index.html#

(pandas, 2022). Razvijena je u AQR Capital Management. Njen glavni tvorac Wes McKinney (2018) navodi da je ime *pandas* derivirano iz *panel data*, što je termin iz ekonometrije za multidimenzionalne strukturirane skupove podataka, te igra frazom *Python data analysis. Pandas* financijski podupire NumFOCUS, koji naglašava da je misija *pandas* da postane najmoćniji i najfleksibilniji alat za analizu podataka otvorenog koda dostupna za primjenu u bilo kojem programskom jeziku (pandas, 2022). McKinney (2018) u svojoj knjizi *Python for Data Analysis* objašnjava je u *pandas* spojio ideje *NumPy*-a o visokim performansama i izračunima, sa sposobnostima fleksibilne manipulacije podatcima koju posjeduju proračunske tablice i relacijske baze podataka, poput SQL-a, a sve u svrhu omogućavanja „sofisticirane funkcionalnosti indeksiranja, lake preoblike, podjele i odabira podskupova podatak“.

Ključna struktura u pandas je objekt *DataFrame* – dvodimenzionalna podatkovna struktura sa stupcima potencijalno drugačijih tipova (pandas, 2022). Možemo ga promatrati kao proračunsku tablicu ili rječnik objekata tipa *Series*. *Series* je jednodimenzionalni array koji može sadržavati bilo koji tip podataka. O *array*-ima će biti više spomena kasnije, u poglavlju koje se bavi s *numPy*-em.

Pristup podatcima je nužan prvi korak za korištenje većine alata., a *pandas* posjeduje niz funkcija za čitanje tabličnih podataka kao *DataFrame* objekata. Više je načina za stvaranje *DataFrame*-a, jedan od kojih je korišten i u ovome radu. Naime, jedan od načina na koji se stvara *DataFrame* je učitavanje .csv datoteke pomoću *pandas*, na taj se način pohranjuje i može čitati pomoću *.read_csv()* metode, što se može vidjeti i na slici 12

```
import pandas as pd
df = pd.read_csv(r'C:\Users\Ana\Desktop\Amazon_Unlocked_Mobile.csv')
df.head(3)
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0

Slika 11. Učitavanje korpusa u Python pomoću pandas

Metoda *.head()* s argumentom 3 ispisuje prva tri reda korpusa u obliku tablice sa stupcima i redovima, iako se ti podatci u našem korpusu nalaz odvojeni zarezom u redovima. To je zasluga metode *.read_csv()* koja učitava podatke iz datoteke te ih razlomi na mjestima gdje se nalazi zarez. Također je važno za napomenuti, *.read_csv()* nije jedina metoda za učitavanje podataka, *.read_excel()* učitava podatke iz Excel tablice, *.read_html* čita sve tablice iz html dokumenta i slično.

Podatci koji nedostaju (engl. *missing data)* su česta pojava pri analizi podataka, a jedan o ciljeva *pandas* je učiniti rad s takvim podatcima što bezbolnijim (McKinney, 2018). Radi li se o numeričkim podatcima, *pandas* koristi *floating-point* vrijednost *NaN* (engl. *Not a Number*) za prikazivanje podataka koji nedostaju. Također, u *pandas* je usvojena konvencija iz programskog jezika R da se na *podatke koji nedostaje* referira kao *NA*, što stoji za engl. *not available*. Više je funkcija koje se koriste za takve podatke, primjerice *.dropna()* koja omogućava korisnicima da analiziraju i ispuste redove ili stupce s *Null* vrijednostima na različite načine ili *fillna* koja ispunjava mjesta gdje nedostaje neki podataka s nekom vrijednošću. Također, da bi olakšala detektiranje vrijednosti koje nedostaju, *pandas* sadrži metode *.isna()* i *.notna()* koje vraćaju *True* ili *False* ovisno o tome jesu li ili nisu detektirali takvu vrijednost. Metoda za rad s podatcima koji nedostaju je naravno više, no ovaj rad koristi samo jednu *.dropna()* i to s parametrom *inplace=True*, koji je inače po defaultu *False,* a kada mu je vrijednost *True*, izvrši operaciju „na licu mjesta“, odnosno na ovaj smo način prošli kroz cijeli korpus i izbacili one redove u kojima nedostaje neka vrijednost bez da smo promijenili originalni korpus.

5.2. NumPy

NumPy[8] je kratica za *Numeric Python* i odavno je temelj mnogih brojčanih operacija u Pythonu (McKinney, 2018). Najnovija verzija je 1.23 te je izišla 22. lipnja 2022. *NumPy* omogućava multidimenzonalne array objekte visokih performansi te alate za rad s tim arrayima. *NumPy*-eva glavna struktura, odnosno objekt je *ndarray* - brz i efikasan n-dimenzionalni *array* objekt, odnosno tablica elemenata, svih istog tipa, koji su indeksirani torkom pozitivnih cijelih brojeva. U *NumPy*-u se dimenzije nazivaju osima (eng. *axes*). Glavne informacije koje definiraju N-dimenzionalni *array* su oblik *array*-a i vrsta stavki od kojih se sastoji (Oliphant, 2006). Oblik *array*-a je skup cijelih brojeva većih od nula – po jedan za svaku dimenziju, koji pružaju informaciju o tome koliko daleko indeks može varirati u toj dimenziji. Svi su *array*-i indeksirani počevši od 0 i završavajući sa n-1, što je i inače konvencija indeksiranja listi u Pythonu.

Za početak je važno, kao i svaku drugu biblioteku, pozvati *numpy* pomoću metode *import*. Nakon toga možemo početi koristiti njegove metode i funkcije. Kao što se može vidjeti na slici 12 pomoću *pandas* je učitan korpus, te izbačeni oni redovi u kojima nedostaje neka vrijednost,

[8] https://numpy.org/

kao i oni u kojima je osvrtu pridodana ocjena od tri zvjezdice, odnosno izbacili smo redove koji sadrže neutralan sentiment. Zatim je, dosta jednostavno, pomoću metode *.where()* označen korpus na način da je dodan stupac 'Positively Rated' gdje je upisano 0, ako se u stupcu 'Rating' nalazi 1 ili 2, ili 1, ako se u stupcu 'Rating' nalazi 4ili 5.

```
import pandas as pd
import numpy as np
df = pd.read_csv(r'C:\Users\Ana\Desktop\Amazon_Unlocked_Mobile.csv')
df.dropna(inplace= True)
df = df[df['Rating'] != 3]
df['Positively Rated'] = np.where(df['Rating'] > 3, 1, 0)
df.head(3)
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Positively Rated
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0	1
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	1
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0	1

Slika 12. Priprema podataka pomoću numpy-a

5.3. NLTK

Natural Language Toolkit ili skraćeno *NLTK*[9] je biblioteka otvorenog koda za obradu prirodnog jezika. Pruža sučelje za preko 50 korpusnih i leksičkih resursa, poput *WordNet*-a. Pruža podršku za mnoge zadatke obrade prirodnog jezika poput klasifikacije, tokenizacije, označavanja vrsta riječi i slično. Tvorci su mu Steven Bird, Ewen Klein i Edward Loper. Slike 13, 14, 15, 16, 17 pokazuju tipične zadatke za koje koristimo *NLTK* u Pythonu te njihove rezultate.

```
from nltk.tokenize import word_tokenize
unos = "I feel so LUCKY to have found this used phone."
pojavnica = nltk.word_tokenize(unos)
print(pojavnica)
```

```
['I', 'feel', 'so', 'LUCKY', 'to', 'have', 'found', 'this', 'used', 'phone', '.']
```

Slika 13. Tokenizacija

```
from nltk.probability import FreqDist
fdist = FreqDist(pojavnica)
fdist
```

```
FreqDist({'I': 1, 'feel': 1, 'so': 1, 'LUCKY': 1, 'to': 1, 'have': 1, 'found': 1, 'this': 1, 'used': 1, 'phone': 1, ...})
```

Slika 14. Frekvencijska distribucija

[9] https://www.nltk.org/

```
from nltk.tokenize import word_tokenize
unos = "I feel so LUCKY to have found this used phone."
pojavnica = nltk.word_tokenize(unos)
print(pojavnica)
```

```
['I', 'feel', 'so', 'LUCKY', 'to', 'have', 'found', 'this', 'used', 'phone', '.']
```

Slika 15. Korjenovanje s PorterStemmer-om

```
lematizator = nltk.WordNetLemmatizer()
korjen2 = [lematizator.lemmatize(p) for p in pojavnica]
print(korjen2)
```

```
['I', 'feel', 'so', 'LUCKY', 'to', 'have', 'found', 'this', 'used', 'phone', '.']
```

Slika 16. Korjenovanje s wordNet lematizatorom

```
print(nltk.pos_tag(token))
```

```
[('I', 'PRP'), ('feel', 'VBP'), ('so', 'RB'), ('LUCKY', 'NNP'), ('to', 'TO'), ('have', 'VB'), ('found', 'VBN'), ('this', 'DT'),
('used', 'JJ'), ('phone', 'NN'), ('.', '.')]
```

Slika 17. Označavanje vrste riječi

5.4. Scikit-Learn

Scikit-learn[10] je biblioteka strojnog učenja izgrađena na temeljima *NumPy*-a, *SciPy*-a i *Matplotlib*-a, koja pruža jednostavne i efikasne alate za česte zadatke u analizi podataka poput klasifikacije, regresije, odabira modela i slično (Igual, Segui, 2017). Projekt je započeo 2007. godine kao projekt na *Google summer of code*, no u javnost je izišao tek 2010. godine. Po uzoru na *NumPy* ulazni podatci su prezentirani kao *numpy array*-i te se zbog toga gotovo bez ikakvih problema integriraju s drugim bibliotekama Pythona. Uključuje module za SVM, logističku regresiju, stablo odluka i slično za zadatke klasificiranja, module za križnu validaciju, metriku i slično za zadatke odabira modela, module za normalizaciju i ekstrakciju osobina za zadatke koji prethode samoj obradi podataka.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df['Reviews'], df['Positively Rated'], random_state = 0)
print('x_train prvi unos:', x_train[0])
print('x_train oblik:', x_train.shape)
```

```
x_train prvi unos: I feel so LUCKY to have found this used (phone to us & not used hard at all), phone on line from someone who
upgraded and sold this one. My Son liked his old one that finally fell apart after 2.5+ years and didn't want an upgrade!! Than
k you Seller, we really appreciate it & your honesty re: said used phone.I recommend this seller very highly & would but from t
hem again!!
x_train oblik: (231207,)
```

Slika 18. Podjela korpusa na skup za treniranje i skup za testiranje pomoću *sklearn*

[10] https://scikit-learn.org/stable/index.html#

sklearn nudi jako korisnu metodu *train_test_split*. Ona dijeli korpus na skup za treniranje i skup za testiranje na način da ga slučajnim odabirom podijeli na particije koje smješta u ta dva skupa. Korištenje ove funkcije znači da nije potrebno ručno dijeliti korpus na skup za testiranje i skup za testiranje. Na slici 18 se vidi primjer podjele korpusa te, primjera radi, prvi primjer u skupu za treniranje. Također, pomoću metode *.shape* je moguće saznati koliko se dokumenata nalazi u skupu *x_train*.

Kako bismo prikazali korpus kao *bag-of-words,* koristit ćemo metode metoda *CountVectorizer* - za konvertiranje skupa podataka u vektor broja pojavnica, te *.transform()* – za pretvaranje tog vektora u matricu termina dokumenta (eng. *document term matrix*).

Metoda *TfidfVectorizer()* procjenjuje „težine" riječi, odnosno koliko su važne za dokument te se kao njen argument može staviti *min_df*, odnosno definirati u koliko se minimalno dokumenata riječ mora pojaviti da uđe u vokabular. TFIDF je produkt TF-a i IDF-a (Provost i Fawcet, 2013). TF je frekvencija termina u dokumentu, a IDF je inverzna frekvencija dokumenta te mjeri rijetkost termina. Veća se težina daje onim riječima koje se često pojavljuju u nekom dokumentu, ali ne i u cijelom korpusu. Manja težina znači da se riječ malo koristi ili u cijelom korpusu ili u jednom dužem dokumentu.

Problem *bag-of-words* pristupa leži u činjenici da ignorira poredak riječi odnosno razlama fraze poput „not working" . U tom je slučaju rješenje je uvođenje n-grama (Provost i Fawcet, 2013), jer „not working" nosi veće značenje od zasebnih riječi „not" i „working". Već spomenuta metoda *CountVectorizer()* u svrhu rješavanja ovog problem može primiti argument *ngram_range().* Sada su nabrojane neke od najvažnijih metoda *scikit-learn*-a, ali i ostalih biblioteka korištenih u analizi sentimenta. Sljedeće će se poglavlje baviti rezultatima koji se dobiju kada se te metode i funkcije primjene na opisan korpus.

6. Opis dobivenih rezultata

U ovom poglavlju će se prikazati ukupan kod potreban za izvođenje analize senimenta, objasniti dobiveni rezultati, izračunati točnost modela te analizirati sentiment na nekoliko primjera. Za početak je potrebno u Python pozvati biblioteke *numpy* i *pandas* te učitati korpus. Prvih pet redova korpusa se nalazi na slici 19.

```
import pandas as pd
import numpy as np
df = pd.read_csv(r'C:\Users\Ana\Desktop\Amazon_Unlocked_Mobile.csv')
df.head()
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	It works good but it goes slow sometimes but i...	0.0
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	Great phone to replace my lost phone. The only...	0.0

Slika 19. Učitavanje korpusa te prvih pet redova

Budući da nam redovi koji sadrže prazne vrijednosti ne koriste, isto kao ni osvrti koji imaju ocjenu 3, odnosno su neutralni, sljedeći je korak izbrisati takve retke iz korpusa. Nadalje, stvoriti ćemo novi stupac imena 'Positively Rated' koji će imati vrijednost 1 za one osvrte koji su ocijenjeni ocjenom 4 ili 5, te vrijednost 0 za one osvrte ocijenjene s 1 ili 2. Na ovaj ćemo načini podijeliti korpus na dvije klase – pozitivne i negativne osvrte. Pozitivni osvrt se nalaze u klasi 1, a negativni u klasi 0. Na slici 20 se može vidjeti kako korpus izgleda nakon što je "očišćen" od nekorisnih podataka.

```
df.dropna(inplace=True)
df = df[df['Rating']!=3]
df['Positively Rated'] = np.where(df['Rating'] >3, 1, 0)
df.head()
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Positively Rated
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0	1
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	1
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0	1
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	It works good but it goes slow sometimes but i...	0.0	1
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	Great phone to replace my lost phone. The only...	0.0	1

Slika 20. Čišćenje korpusa

Pogledamo li srednju vrijednost stupca ‘Positively Rated’ pomoću metode *.mean()* dobit ćemo rezultat od 0.748, što znači da je 74.8% korpusa je pozitivno ocijenjeno, odnosno klase koje imamo su neuravnotežene. Metodom *train_test_split* ćemo podijelit korpus u skup za testiranje i skup za treniranje. Metoda *.shape* će nam reći koliko se dokumenata nalazi u skupu za treniranje. Slika 21 prikazuje prvi redak stupca ‘Reviews’ te stupca ‘Positively Rated’ u korpusu ze treniranje te podatak da se u korpusu za treniranje nalazi 231207 osvrta, odnosno dokumenata.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(df['Reviews'], df['Positively Rated'], random_state=0)
print('x_train first entry: \n\n',x_train[0])
print('\n y_train first entry: \n\n',y_train[0])
print('\n\n x_train shape:',x_train.shape)
```

```
x_train first entry:

 I feel so LUCKY to have found this used (phone to us & not used hard at all), phone on line from someone who upgraded and sold
this one. My Son liked his old one that finally fell apart after 2.5+ years and didn't want an upgrade!! Thank you Seller, we r
eally appreciate it & your honesty re: said used phone.I recommend this seller very highly & would but from them again!!

 y_train first entry:

 1


 x_train shape: (231207,)
```

Slika 21. Podjela korpusa na skup ze treniranje i skupa za testiranje

Te ćemo dokumente morati pretvoriti u numerički prikaz koji *scikit-learn* može koristiti. Već spomenuti *bag-of-words* pristup jednostavan je i često korišten način predstavljanja teksta koji zanemaruje strukturu i samo broji koliko se često svaka riječ pojavljuje. *CountVectorizer* nam omogućuje korištenje *bag-of-words* pretvaranjem zbirke tekstualnih dokumenata u matricu broja pojavnica. Prvo, instanciramo *CountVectorizer* i prilagodimo ga našim podatcima za treniranje pomoću *.fit()*. Prilagodba *CountVectorizera* sastoji se od tokenizacije, odnosno pretvaranja u pojavnice, podataka za treniranje te izgradnje vokabulara. *CountVectorizer* tokenizira svaki dokument pronalaženjem svih nizova znakova od najmanje dva slova ili broja odvojenih znakom razmaka. Pretvara sve u mala slova te gradi rječnik koristeći te pojavnice. Rječnik ćemo dobiti pomoću metode *get_feature_names*. Primjera radi, na slici 22 je prikazan svaki tritisućiti unus u rječniku te možemo vidjeti kako rječnik otprilike izgleda.

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer().fit(x_train)
vect.get_feature_names()[::3000]
```

```
['00',
 '99303',
 'assignment',
 'bullets',
 'condishion',
 'definitely',
 'esteem',
 'fusion2',
 'human',
 'kinds',
 'microsaudered',
 'oldy',
 'poori',
 'rediculoius',
 'send',
 'storecons',
 'trace',
 'waiste']
```

Slika 22. Svaki tritisućiti unos u riječniku pojavnica

Vidimo da rječnik izgleda prilično neuredno, odnosno uključuje riječi s brojevima kao i pogrešno napisane riječi ili samo brojeve. Pomoću naredbe *len(vect.get_feature_names())* ćemo dobiti podatak da baratamo s 53216 pojavnica. Nadalje, pomoću metode *.transform(),* što je prikazano na slici 23, dokument u skupu *x_train* transformiramo u matricu pojavnica te na taj način dobijemo reprezentaciju *x_traina* u obliku *bag-of-words*. U toj matrici svaki redak odgovara dokumentu, a svaki stupac riječi iz našeg vokabulara za treniranje. Unosi u ovoj matrici predstavljaju broj pojavljivanja svake riječi u svakom dokumentu. Budući da je broj riječi u vokabularu puno veći od broja riječi koje bi se mogle pojaviti u jednom osvrtu, većina unosa ove matrice je nula.

```
x_train_vectorized = vect.transform(x_train)
x_train_vectorized
```

```
<231207x53216 sparse matrix of type '<class 'numpy.int64'>'
        with 6117776 stored elements in Compressed Sparse Row format>
```

Slika 23. Metoda .transform()

Nadalje ćemo upotrijebiti ovu matricu pojavnica *x_ train_vectorized* za treniranje modela. U ovom ćemo slučaju koristiti logističku regresiju, kako je prikazano na slici 24, međutim je moguće koristiti i druge, gore navedene klasifikatore.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train_vectorized,y_train)
```

Slika 24. Treniranje modela logističke regresije

Idući je korak napraviti predviđanja koristeći *x_test* te izračunati površinu ispod krivulje, odnosno *roc_auc_score*. *ROC AUC* je površina ispod *ROC* krivulje, odnosno površina dijagrama koji predstavlja odnos količine stvarnih pozitivnih rezultata i količine lažnih pozitivnih rezultata te se koristi kao mjera kvalitete klasifikatora. Što je *roc_auc_score* bliži 1 to je klasifikator bolji, što je bliži 0.5 to je veća nasumičnost kojom su odabirane klase pri klasifikaciji. Transformirat ćemo *x_test* koristeći vektorizator koji je prilagođen podacima za treniranje. Važno je napomenuti da će riječi u *x_testu* koje se nisu pojavile u *x_trainu* jednostavno biti zanemarene.

```
from sklearn.metrics import roc_auc_score
predictions = model.predict(vect.transform(x_test))
print('AUC score:', roc_auc_score(y_test, predictions))
```

```
AUC score: 0.9205848532221643
```

Slika 25. roc_auc_score

Pogledamo li koeficijente iz modela, odnosno deset najmanjih i deset najvećih koeficijenata koji se nalaze na slici 26, možemo vidjeti da je model povezao riječi kao što su “worst”, “garbage” i “junk” s negativnim recenzijama, a riječi kao što su “excellent”, “loves” i “awesome” s pozitivnih recenzija.

```
feature_names = np.array(vect.get_feature_names())
sorted_coef_index = model.coef_[0].argsort()
print('Smallest coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest coefs:\n{}\n'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest coefs:
['worst' 'garbage' 'junk' 'unusable' 'false' 'worthless' 'useless'
 'crashing' 'disappointing' 'awful']

Largest coefs:
['excelent' 'excelente' 'exelente' 'loving' 'loves' 'perfecto' 'excellent'
 'complaints' 'awesome' 'buen']
```

Slika 26. Prikaz 10 najmanjih i najvećih koeficijenata

Sljedeći je korak izračunati *Tf-Idf*, odnosno *Term frequency-inverse document frequency*, koja nam omogućava da brojčano izrazimo koliko je pojavnica važna za dokument. Velika težina pridodaje se pojavnicama koje se često pojavljuju u određenom dokumentu, ali se ne pojavljuju često u korpusu. Pojavnice s manjim *tf–idf*-om ili se često koriste u svim dokumentima ili se rijetko koriste i pojavljuju se samo u dugim dokumentima. Slično kao kad smo koristili *CountVectorizer*, instancirat ćemo *tf–idf* vektorizator i prilagoditi ga podacima za treniranje.

Budući da *tf–idf* vektorizator prolazi kroz isti početni proces tokenizacije dokumenta, možemo očekivati da će vratiti isti broj značajki.

Međutim, *tf–idf* vectorizor kao argument može uzeti *min_df* koji nam omogućava da specificiramo minimalni broj dokumenata u kojima se pojavnica treba pojaviti da bi postala dio vokabulara. To nam pomaže da uklonimo one riječi koje će se pojaviti samo u nekolicini dokumenata i vjerojatno neće biti korisni prediktori. Na primjer, ovdje ćemo unijeti *min_df* = 5, što će ukloniti sve riječi iz našeg vokabulara koje se pojavljuju u manje od pet dokumenata. Time ćemo smanjenjiti broj pojavnica što pak može pomoći u poboljšanju performansi modela.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vect = TfidfVectorizer(min_df=5,).fit(x_train)
len(vect.get_feature_names())
```

```
17951
```

Slika 27. Slika 27. tf–idf vektorizator

Možemo vidjeti da smo smanjili broj pojavnica s preko 35000 na 17951. Ponovimo li gornji proces te ponovno izračunamo *Area Under Curve* ćemo vidjeti da nema poboljšanja u *AUC* rezultatu, ali smo uspjeli dobiti isti rezultat koristeći mnogo manje pojavnica. Pogledajmo na slici 28 koje pojavnice imaju najmanji i najveći *tf–idf*:

```
feature_names = np.array (vect.get_feature_names())
sorted_tfidf_index = x_train_vectorized.max(0).toarray()[0].argsort()
print('Smallest tfidf:\n{}\n'.format(feature_names[sorted_tfidf_index[:10]]))
print('Largest tfidf:\n{}\n'.format(feature_names[sorted_tfidf_index[:-11:-1]]))
```

```
Smallest tfidf:
['commenter' 'pthalo' 'warmness' 'storageso' 'aggregration' '1300'
 '625nits' 'a10' 'submarket' 'brawns']

Largest tfidf:
['defective' 'batteries' 'gooood' 'epic' 'luis' 'goood' 'basico'
 'aceptable' 'problems' 'excellant']
```

Slika 28. Pojavnice s najvećim i najmanjim tf-idf-om

```
sorted_coef_index = model.coef_[0].argsort()
print('Smallest coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest coefs:\n{}\n'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest coefs:
['not' 'worst' 'useless' 'disappointed' 'terrible' 'return' 'waste' 'poor'
 'horrible' 'doesn']

Largest coefs:
['love' 'great' 'excellent' 'perfect' 'amazing' 'awesome' 'perfectly'
 'easy' 'best' 'loves']
```

Slika 29. Novi izračun koeficijenata

Pogledamo li najmanji i najveći koeficijent iz novog modela na slici 29, ponovno možemo vidjeti koje je riječi naš model povezao s negativnim i pozitivnim recenzijama. Jedan problem s pristupom *bag-of-words* je zanemarivanje reda riječi. Odnosno, rečenica „not an issue, phone is working“ će se klasificirati isto kao „an issue, phone is not working“, u ovom slučaju kao negativnu recenziju, kao što možemo vidjeti na slici 30.

```
print(model.predict(vect.transform(['not an issue, phone is working', 'an issue, phone is not working'])))
```

```
[0 0]
```

Slika 30. Primjer problema s bag-of-words pristupom

Jedan od načina na koji možemo dodati određeni kontekst je korištenje n-grama, odnosno nizova riječi. Na primjer, bi-gram, koji daj parove susjednih riječi, bi nam mogao dati nizove kao što je “is working” ili “not working”. A tri-grami, koji nam daju trojke susjednih riječi, mogli bi nam dati značajke poput “is not an issue.” Da bismo stvorili n-gram, koristit ćemo naredbu *ngram_range* čije vrijednosti odgovaraju minimalnoj duljini i maksimalnoj duljini sekvenci.U ovom ćemo slučaju koristiti torku (1, 2).

```
vect = CountVectorizer(min_df=5, ngram_range=(1,2)).fit(x_train)
x_train_vectorized = vect.transform(x_train)
len(vect.get_feature_names())
```

```
198917
```

Slika 31. n-gram

Iako su n-grami korisni, mogu uzrokovati veliko povećanje broja pojavnica, kao što vidimo na gornjoj slici 31, broj pojavnica je sada gotovo 200 000. Nakon traniranja modela logističke regresije i dodavanja novih pojavnica, dodavanjem bigrama, uspjeli smo poboljšati *AUC* rezultat na 0,967. Ako na slici 32 pogledamo koje su karakteristike našeg modela povezane s

negativnim recenzijama, možemo vidjeti da sada imamo bigrame kao što su „no good“ i „not happy“, dok za pozitivne osvrte imamo „not bad“i „no problems“. Ako opet pokušamo klasificirati „not an issue, phone is working“ i „an issue, phone is not working“, možemo vidjeti da ih najnoviji model sada ispravno identificira kao pozitivnu, odnosno negativnu recenzije.

```
feature_names = np.array (vect.get_feature_names())
sorted_coef_index = model.coef_[0].argsort()
print('Smallest coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest coefs:\n{}\n'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest coefs:
['no good' 'not happy' 'not worth' 'worst' 'junk' 'not satisfied'
 'garbage' 'not good' 'terrible' 'defective']

Largest coefs:
['excelent' 'excelente' 'not bad' 'excellent' 'exelente' 'perfect'
 'awesome' 'no problems' 'no issues' 'perfecto']
```

Slika 32. Koeficijenti nakon korištenja bigrama

Pokušamo li s još nekoliko primjera, vidjeti ćemo da je model dosta točan, međutim, očekivano, ima problema s onim primjerima gdje je potrebno poznavanje semantike, odnosno s onim primjerima gdje jezik nije sveden samo na niz riječi poput “not bad, but cheaply made”. Kao što je moguće vidjeti na slici 33 ovaj je primjer klasificiran kao pozitivan. Za kraj je pomoću *accuracy_score* provjerena preciznost modela te je dobiven rezultat od 97.3%.

```
print(model.predict(vect.transform(['this phone is garbage', 'this phone is epic', 'bad'])))
```

```
[0 1 0]
```

```
print(model.predict(vect.transform(['phone is the worst', 'EPIC!'])))
```

```
[0 1]
```

```
print(model.predict(vect.transform(['not bad, but cheaply made'])))
```

```
[1]
```

Slika 33. Primjeri klasifikacije

7. Zaključak

U ovom su radu opisani odabrani postupci obrade prirodnog jezika i strojnog učenja u analizi sentimenta.

Razmišljamo li o mogućem korištenju analize sentimenta, dolazimo do zaključka da analiza sentimenta ima široku primjenu. Idemo li kino zanimat će nas mišljenje drugih ljudi o filmu koji želimo gledati. Tvrtku koja se bavi proizvodnjom zanimat će što kupci misle o njenim proizvodima, odnosno koliko su pozitivni ili negativni prema njima. Političke stranke zanima podupiru li ljudi njihov program ili ne. Društvene organizacije zanima opće mišljenje ljudi o trenutnim debatama, problemima i slično. Budući da iznošenje mišljenja na internetu ljudi još uvijek povezuju s anonimnošću, slobodniji su se izraziti tim putem. Ogromna količina podataka koja je svakodnevno generirana na internetu ovim putem, otvara vrata prikupljanju tih podataka u svrhu analiziranja istih. Korist će u podatcima pronaći svi.

U radu je prikazana analiza sentimenta pomoću programskog jezika Python na korpusu osvrta s Amazona, što se pokazalo vrlo uspješnim - toj je uspješnosti svakako pridonijela i činjenica da su korišteni osvrti na engleskom jeziku.

8. Literatura

1. Aggarwal, C. (2015) Data Mining: The Textbook. New York: Springer.
2. An Introduction to Bag-of-Words in NLP (2018). Pristupljeno: 20.08.2022. https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428
3. Bing, L. (2015) Sentiment Analysis: Mining Opinions, Sentiments and Emotions. New York: Cambridge University Press.
4. Bird, S., Klein, E. i Loper, E. (2019) Natural Language Processing with Python. Sabastopol, CA: O'Reillly.
5. Bramer, M. (2016) Principles of Data Mining. London: Springer.
6. Decision Trees (n.d.) Pristupljeno: 21.08.2022. https://scikit-learn.org/stable/modules/tree.html
7. Igual, L. i Segui, S. (2017) Introduction to Dana Science: A Python Approach to Concepts, Texhniques and Applications. Switzerland: Springer.
8. Jakopović, H. i Mikelić Preradović N. (2016) Identifikacija online imidža organizacija temeljem analite sentimenta korisnički generiranog sadržaja na hrvtskim portalima. *Medijaska istraživanja*, 22(2), str. 63-82.
9. Jurafsky, D. i Martin, J. (2019) Speech and Language Processing. New Jersey: Prentice-Hall, Inc.
10. Kantardzic, M. (2011) Data Mining: Concepts, Models, Methods, and Algorithms. New Jersey: John Wiley & Sons, Inc.
11. Kitchin, R. (2014) The Data Revolution. Los Angeles: Sage
12. Liu, B. (2010) Sentiment Analysis and Subjectivity. U: Indurkhya N. i Damerau F.J., ur. *Handbook of Natural Language Processing*. United States of America: Chapman & Hall/CRC. Str. 627-666.
13. Liu, B. (2012) Sentiment Analysis and Opinion Mining. San Rafael, CA: Morgan & Claypool Publishers.
14. McKinney, W. (2018) Python for dana analysis: dana wrangling with Pandas, Numpy and IPython. Sebastopol: O'Reilly.
15. Oliphant T. (2006) Guide to NumPy.

16. pandas documentation (2022). Pristupljeno: 21.08.2022. https://pandas.pydata.org/docs/index.html#
17. pandas package overview (2022). Pristupljeno: 20.08.2022. https://pandas.pydata.org/docs/getting_started/overview.html?highlight=numfocus
18. Pros and cons of various Machine Learning algorithms (2020). Pristupljeno: 21.08.2022. https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6
19. Provost, F. i Fawcet, T. (2013) Dana science for business: what you need to know about dana mining and dana-analytic thinking. Sabastopol, CA: O'Reillly.
20. Sarkar, D. (2016) Text Analytics with Paython. Banglore: apress.
21. Shin, T. (2020) A Mathematical Expalantion of Naive Bayes in 5 Minutes. Pristupljeno: 20.08.2022. https://towardsdatascience.com/a-mathematical-explanation-of-naive-bayes-in-5-minutes-44adebcdb5f8
22. Skansi, S. (2018) Intriduction to Deep Learning: From Logical Calculus to Artificial Inteligence. Springer.
23. sklearn.linear_model.LogisticRegression (n.d.) Pristupljeno: 20.08.2022. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
24. sklearn.tree.DecisionTreeClassifier (n.d.) Pristupljeno: 20.08.2022. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision%20tree#sklearn.tree.DecisionTreeClassifier
25. Support Vector Machines (n.d) Pristupljeno: 21.08.2022. https://scikit-learn.org/stable/modules/svm.html
26. VanderPlas, J. (2017) Python Dana Science Handbook. Sebastopol, CA: O'Reilly.
27. Wise. J. (2022) HOW MUCH DATA IS CREATED EVERY DAY IN 2022?. Pristupljeno: 21.08.2022. https://earthweb.com/how-much-data-is-created-every-day/
28. Yassenov, K. i Misailovič, S. (2009) Sentimenyt Analysis of Movie Review Comments. *International Conference on Data Mining Workshops.* United States Of America: International Conference on Data Mining Workshops

Popis slika

Slika 1. Shema križne validacije (Izvor: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.svg)
Slika 2. Naivni Bayesov klasifikator u nltk
Slika 3. Naivni Bayesov klasifikator u sklearn
Slika 4. Stablo odlučivanja u sklearn
Slika 5. Linearno odvajanje dviju klasa (izvor: VanderPlas, 2017)
Slika 6. Margina (izvor: VanderPlas, 2017)
Slika 7. SVM u sklearn
Slika 8. Logistička regresija
Slika 9. Histogram
Slika 10. Prvih pet redova korpusa
Slika 11. Učitavanje korpusa u Python pomoću pandas
Slika 12. Priprema podataka pomoću numpy-a
Slika 13. Tokenizacija
Slika 14. Frekvencijska distribucija
Slika 15. Korjenovanje s PorterStemmer-om
Slika 16. Korjenovanje s wordNet lematizatorom
Slika 17. Označavanje vrste riječi
Slika 18. Podjela korpusa na skup za treniranje i skup za testiranje pomoću sklearn
Slika 19. Učitavanje korpusa te prvih pet redova
Slika 20. Čišćenje korpusa
Slika 21. Podjela korpusa na skup ze treniranje i skupa za testiranje
Slika 22. Svaki tritisućiti unos u riječniku pojavnica
Slika 23. Metoda .transform()
Slika 24. Treniranje modela logističke regresije
Slika 25. roc_auc_score
Slika 26. Prikaz 10 najmanjih i najvećih koeficijenata
Slika 27. Slika 27. tf–idf vektorizator
Slika 28. Pojavnice s najvećim i najmanjim tf-idf-om
Slika 29. Novi izračun koeficijenata
Slika 30. Primjer problema s bag-of-words pristupom
Slika 31. n-gram

Slika 32. Koeficijenti nakon korištenja bigrama
Slika 33. Primjeri klasifikacije

Popis formula

Jednadžba 1. Bayesov teorem

Jednadžba 2. Naivni Bayesov klasifikator

Popis tablica

Tablica 2. Konačna struktura korpusa

Korištenje Pythona u analizi sentimenta

Sažetak

Mišljenja, naša vlastita, ali i ona drugih ljudi, uvelike utječu na naše ponašanje i donošenje odluka. Želimo li, na primjer, kupiti novi proizvod prvo ćemo posegnuti za osvrtima, tj. mišljenjima drugih ljudi o njemu. Analiza sentimenta ili dubinsko pretraživanje mišljenja se odnosi na računalnu obradu prirodnog jezika kojom se analiziraju stavovi i mišljenja pojedinaca o nekom proizvodu, temi, usluzi i slično. Ovaj će se završni rad baviti primjenom programskog jezika Python u analizi sentimenta. Posebno će se koristiti biblioteke Pandas, NLTK i Scikit-learn te njihove metode i funkcije. Također, bit će prikazani klasifikatori, s posebnim osvrtom na Naivni Bayesov klasifikator te logističku regresiju. Korpus koji će biti korišten u primjerima je korpus osvrta s Amazona koji je preuzet s poveznice: https://www.coursera.org/learn/python-text-mining/resources/d9pwm

Ključne riječi: analiza sentimenta, Python, pandas, NLTK, scikit, klasifikacija

Using Python for sentiment analysis

Summary

Opinions, our own and those of other people, influence our behavior and decision-making. If, for example, we want to buy a new product, we will first look for reviews, i.e. other people's opinion about it. Sentiment analysis or opinion mining refers to computer processing of natural language, which analyzes the attitudes and opinions of individuals about e.g. a product, topic or a service. This paper will deal with the application of the Python programming language in sentiment analysis. In particular with Pandas, NLTK and Scikit-learn libraries and their methods and function. Also, classifiers will be presented, with a special focus on the Naive Bayes classifier and logistic regression. The corpus that will be used in the examples is the review corpus from Amazon downloaded from the links: https://www.coursera.org/learn/python-text-mining/resources/d9pwm

Key words: seniment analysis, Python, pandas, NLTK, scikit, clasification