Automatska klasifikacija tekstualnih dokumenata temeljem teme

Šebalj, Domagoj

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:088042

Rights / Prava: Attribution-NonCommercial-ShareAlike 3.0 Unported/Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 3.0

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

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Domagoj Šebalj

Algoritmi klasifikacije u strojnom učenju

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Domagoj Šebalj

Matični broj: 44856/16-R

Studij: Informacijski sustavi

Algoritmi klasifikacije u strojnom učenju

ZAVRŠNI RAD

Mentor/Mentorica:

Prof. dr. sc. Jasminka Dobša

Varaždin, rujan 2020.

Domagoj Šebalj

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

.Područje ovog rada obuhvaća algoritme klasifikacije strojnog i dubokog učenja s naglaskom na klasifikaciju tekstualnih dokumenata. . Cilj rada je na intuitivan način objasniti algoritme klasifikacije i njihove prednosti, odnosno nedostatke. U uvodu je riječ o potrebi za strojnim učenjem i klasifikacijom podataka. Zatim su objašnjene vrste učenja te vrste algoritama nadgledanog učenja na primjerima algoritama klasifikacije. Poslijednji dio rada obuhvaća izradu i analizu modela za klasifikaciju kompleksnog skupa podataka.

Ključne riječi: duboko učenje; strojno učenje; klasifikacija tekstualnih podataka; analiza osjećaja; procesuiranje prirodnog jezika; znanost o podacima

Sadržaj

1. Uvod 1

2. Algoritmi strojnog učenja 2

2.1. Nadgledani način učenja 2

2.2. Nenadgledani način učenja 2

2.3. Polunadgledani način učenja 3

2.4. Učenje podrškom 3

3. Algoritmi klasifikacije 4

3.1. Logistička regresija (engl. logistic regression) 4

3.2. K-najbližih susjeda (engl. K-Nearest Neighbors, KNN) 6

3.3. Metoda potpornih vektora (engl. Support Vector Machines, SVM) 7

3.4. Naivni Bayes-ov algoritam (engl. Naive Bayes algorithm, NB) 9

3.5. Klasifikacija pomoću stabla odlučivanja (engl. decision tree classification) 11

3.6. Klasifikacija slučajnom šumom (engl. Random forest classification) 13

3.7. Duboko učenje (engl. deep learning) 15

3.7.1. Unaprijedna neuronska mreža (engl. Feed – Forward Neural Network) 17

3.7.2. Modeli bazirani na povratnim neuronskim mrežama (engl. Recurrent neural networks) 18

3.7.3. Modeli bazirani na konvolucijskim neuronskim mrežama (engl. Convolutional Neural Network – Based models, CNN – Based models) 19

4. Primjena 20

4.1. Okruženje 20

4.1.1. Biblioteke 20

4.2. Skup podataka 21

4.3. Model 25

4.4. Evaluacija modela 27

5. Zaključak 30

Popis literature 31

Popis ilustracija 33

1. Uvod

Svijet se danas nalazi u informacijskom dobu. Svakog se dana stvori enormna količina podataka od kojih je većina nestrukturirana. Klasifikacija tih podataka omogućuje veće razumijevanje potreba ljudske populacije. Od ciljanog marketinga koji spušta cijene za tvrtke te većeg pristupa manjih i novih tvrtki kupcima. Znati tko želi što kupiti postalo je iznimno važno te omogućilo probijanje manjih tvrtki na svjetsko tržište, a kupcima omogućilo pristup novim, boljim i jeftinijim proizvodima.

Na jednak je način prediktivno modeliranje omogućilo praćenje navika kupaca te samim time omogućilo bankama bolju i bržu reakciju na kupnje koje nisu karakteristične za određenu osobu, povećavajući pritom sigurnost novčanih sredstava svojih kupaca.

Internet servisi postaju sve češći i sve više personiziraniji. Netflix, Spotify i slične aplikacije omogućuju pristup audiovizualnom sadržaju koji najbolje odgovara nekoj osobi. Pametni sustavi prate i smanjuju potrošnju električne energije u domovima ljudi. Navigacijski sustavi prate, ne samo najkraće puteve do odredišta, već i količinu prometa, brzinu vožnje i mnoge druge uvjete na cesti. Sve to čini život lakšim i jednostavnijim, a omogućeno je razumijevanjem svih podatka koje na dnevnoj razini kao populacija proizvedemo.

Stoga, grana strojnog učenja, može očekivati velik porast u nadolazećim godinama. International Data Corporation predviđa da će količina proizvedenih podataka do 2025. narasti na gotovo 175 zettabajta. Imati ljude koji će analizirati sve te podatke jednostavno nije izvedivo, a i ljudski mozak ima svoja ograničenja kod prepoznavanja uzoraka. Tako da, ako se želi nastaviti trend rasta ugodnosti života i jednostavnosti pristupa informacijama i uslugama, potrebno je razviti modele koji će automatizirati cijeli proces, omogućiti nam razumijeti potrebe ljudi i pomoći nam dostaviti ljudima ono što im je potrebno. Klasifikacija podataka je samo početak toga.

2. Algoritmi strojnog učenja

Postoje različiti načini na koje je moguće klasificirati algoritme strojnog učenja. Jedan od najčešćih je klasificiranje po načinu učenja. Po tom kriteriju algoritmi su podijeljeni na algoritme nadgledanog načina učenja (supervised learning), nenadgledanog načina učenja (unsupervised learning) te učenje podrškom (reinforcement learning), a u nekim se izvorima također spominje polunadgledani način učenja (semi-supervised learning). (Brownlee, 30.09.2020.)

2.1. Nadgledani način učenja

U ovom slučaju ulazni se podaci nazivaju podacima za treniranje (x) i svaki ima svoju oznaku ili rezultat (Y). Tijekom procesa treniranja, model pokušava usavršiti funkciju (f(x)) kojom svakom ulaznom podatku dodjeljuje oznaku ili rezultat. Usavršavanje funkcije se izvršava iterativno, na način da model prvo dobije podatke za treniranje, zatim stvara predviđanje bazirano na funkciji razvijenoj od prijašnjih podataka te, ako je predviđanje pogrešno, mijenja parametre funkcije kako bi približio predviđanje stvarnoj oznaci ili rezultatu.

Takav način strojnog učenja je moguće lakše shvatiti kao aritmetičke zadatke s odgovorima, ali izbrisanim aritmetičkim operacijama. Model stoga pokušava shvatiti koje operacije koristiti kako bi brojevi bili smisleni.

Ovu skupinu algoritama je moguće dodatno podijeliti na algoritme klasifikacije, gdje je izlazni podatak predviđena klasa podataka (npr. boja: crvena, zelena), te na algoritme regresije, gdje je izlazni podatak općenito realni broj (npr. novčana vrijednost, težina)

2.2. Nenadgledani način učenja

Algoritmi koji uče na ovaj način su okarakterizirani nedostatom oznaka, odnosno rezultata, u podacima za treniranje. Model u ovom slučaju ima slobodu sam tražiti uzorke u podacima za treniranje grupiranjem podataka za koje smatra da su povezani što predstavlja jednu od najvećih prednosti ovakvih modela. Cilj stoga nije klasifikacija u smislu nadgledanog načina učenja, već analiza osnovne strukture podataka kako bi se uočili nepoznati uzorci. (Brownlee, 2020.)

Ovu je skupinu također moguće dodatno podijeliti na algoritme klasteriranja, kojima se želi otkriti svojstvene grupacije podataka (npr. grupiranje kupaca po navikama kupovine), te na algoritme asocijacije, kojima se žele otkriti pravila koja opisuju veliku količinu podataka (npr. kupci koji kupe proizvod X također kupuju proivod Y)

2.3. Polunadgledani način učenja

Problemi u kojima postoji velika količina ulaznih podataka (x), a manji dio tih podataka ima svoju oznaku (Y) se rješavaju koristeći polunadgledani način učenja. Oni se nalaze između nadgledanog i nenadgledanog načina. Velik dio problema današnjice spada u ovu kategoriju dijelom zbog visoke cijene procesa označavanja svih podataka, a drugim dijelom zbog velike dostupnosti i niske cijene nestrukturiranih podataka. Ono se koristi za otkrivanje i učenje strukture ulaznih podataka.

Dobar primjer ovakvog problema je skup slika koje je potrebno klasificirati, ali samo dio tih slika ima oznaku (npr. pas, mačka, osoba).

2.4. Učenje podrškom

Učenje podrškom podrazumijeva treniranje modela za donošenje sekvence odluka pokušavajući postići neki cilj u kompleksnom okruženju. Može se zamisliti da je stroju predstavljeno okruženje nalik računalnoj igrici gdje za svaku odluku može biti nagrađen ili penaliziran. Cilj takvog stroja je maksimizirati dobivene nagrade određene od strane programera na sličan način na koji se kod ljudi oslobađa dopamin prilikom postizanja nekog cilja. (Osiński, Budek, 2018.)

Iako programer definira nagrade za određene događaje, on stroju ne daje naputke ili prijedloge već stroj mora sam otkriti kako riješiti neki problem kako bi maksimizirao nagradu. To se postiže isprva nasumično, zatim otkrivanjem sofisticiranijih taktika. Danas je to najbliže što stroj može doći ljudskoj kreativnosti, ali prednost je stroja što može skupiti iskustvo iz tisuće igranja istovremeno pod uvjetom da ima dovoljno jake računalne komponente.

Primjer ovakvog načina moguće je uočiti u samovozećim automobilima. Razlog tome je što su situacije na cesti nepredvidive i nemoguće je napisati toliko if-else selekcija. Model stoga može naučiti da prilikom vožnje uvijek mora staviti sigurnost na prvo mjesto, zatim minimizirati vrijeme vožnje, pratiti prometne znakove i slično. Takvi prioriteti se određuju sustavom nagrada.

3. Algoritmi klasifikacije

Kako je u ovom radu naglasak na klasifikaciji tekstualnih podataka radi se o nadlgedanom načinu učenja. Takva klasifikacija je klasični problem obrade prirodnog jezika (engl. Natural Language Processing - NLP) kojoj je cilj pridodati odgovarajuću oznaku tekstualnim podacima kao što su rečenice i odlomci. Primjena klasifikacije tekstualnih podataka ima veoma široku primjenu u automatiziranom odgovaranju na pitanja, detekciji spam mailova, kategorizaciji vijesti, moderiranju sadržaja, klasifikaciji namjere korisnika te analizi osjećaja koja je fokus ovoga rada.

Pristup klasifikaciji teksta mogu biti (prema: Minaee, Kalchbrenner, Cambria, Nikzad, Chenaghlu, Gao, 2020.): metode bazirane na pravilima (engl. rule-based methods), metode bazirane na strojnom učenju (machine learning based methods) te hibridne metode.

Karim i Rahman (2013.) predlažu da bi ta podjela trebala uključivati metode bazirane na: statistici, udaljenosti, stablima odlučivanja, neuronskim mrežama, te pravilima.

Metode bazirane na pravilima tekst klasificiraju u različite kategorije pomoću skupa unaprijed definiranih pravila. Na primjer, bilo koji dokument u kojem se nalaze riječi „nogomet“ ili „košarka“ će biti svrstani u kategoriju sportova. Ovakve metode zahtjevaju dobro poznavanje domene zadatke i teške su za održavanje. Metode strojnog učenja se razlikuju od ovakvih metoda po tome što koriste unaprijed označene primjere na kojima model može pronaći poveznice između teksta i kategorije kojoj pripada. Dakle strojnim se učenjem mogu uočiti skriveni uzorci u podacima. Takvi su sustavi skalabilniji, jednostavniji za održavanje i može ih se koristiti za različite zadatke. Hibridne metode, kako i samo ime govori koriste kombinaciju metoda baziranih na pravilima i strojnog učenja kako bi napravili predviđanje. (Minaee, Kalchbrenner, Cambria, Nikzad, Chenaghlu, Gao, 2020.)

3.1. Logistička regresija (engl. logistic regression)

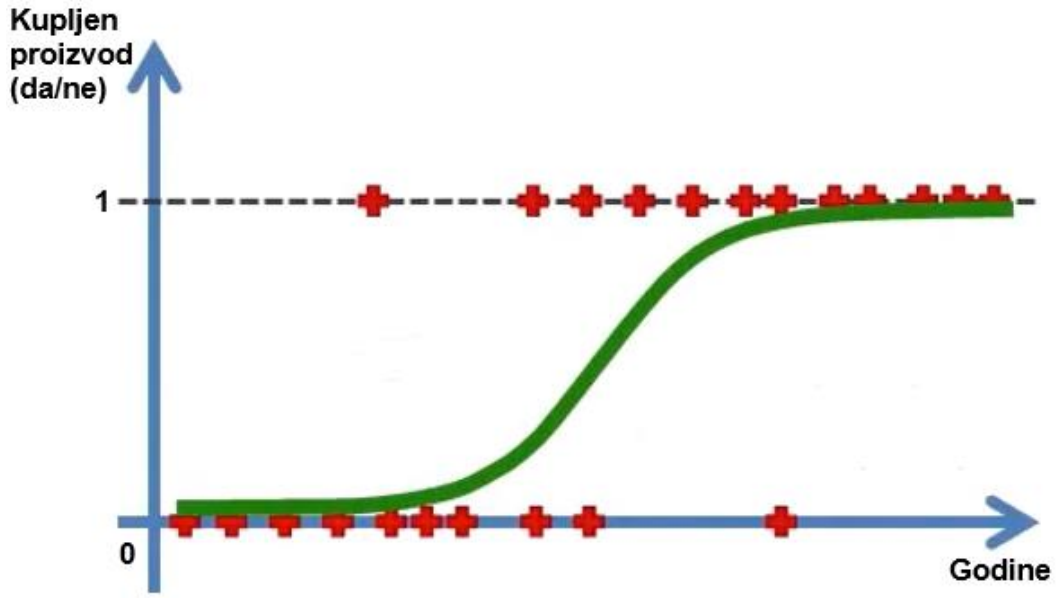
Regresija je statistička metoda u kojoj je jedna varijabla objašnjena temeljem jedne ili više drugih varijabli. Varijable koje ovom metodom objašnjavamo nazivaju se zavisne, a one kojima ju objašnjavamo nezavisne. Stoga je cilj logičke regresije je razumjeti binarni ili proporcionalni odgovor (zavisnu varijablu) temeljem jednog ili više prediktora. (Hilbe, 2020.)

Za bolje razumijevanje logičke regresije može se zamisliti situacija u kojoj neku tvrtku zanima je li kupac koji je primio neku posebnu ponudu kupio taj proizvod ili ne u ovisnosti o njegovoj/njezinoj dobi. Zamislimo da mlađi kupci većinom nisu kupili proizvod, a stariji kupci većinom jesu.

Na slici 1. kupci su prikazani crvenom bojom, a funkcija kojom se može približno odrediti trend je prikazana zelenom. Na X osi su prikazane godine, a na Y osi informacija o tome je li osoba kupila proizvod ili ne. Funkcija se naziva sigmoidalna funkcija te je određena jednadžbom:

$$P = \frac{1}{1 + e^{-x}}$$

P u ovom slučaju predstavlja vjerojatnost da je osoba kupila proizvod, a x je parametar modela (godine kupca). Ovaj model stoga kao izlaznu vrijednost ne daje jednostruk odgovor, već vjerojatnost da je kupac kupio proizvod što se može interpretirati na različite načine, a prikazano je na Y osi. Primjerice, ako je vjerojatnost veća od 50% pretpostavit ćemo da je osoba kupila proizvod, a ako je manja od 50% pretpostavljamo da osoba nije kupila proizvod



Slika 1: Model logističke regresije

Prednost ovakvog modela je upravo u vjerojatnostima. Svaku novu osobu moguće je postaviti na grafu te odrediti vjerojatnost da će kupiti taj proizvod što nam omogućuje rangiranje kupaca te ciljano slanje ponuda onima za koje se smatra da će kupiti proizvod. Takvo rangiranje može uštediti značajne količine novčanih sredstava tvrtke.

3.2. K-najbližih susjeda (engl. K-Nearest Neighbors, KNN)

Algoritam K-najbližih susjeda radi na principu udaljenosti između poznatih podataka i novog neoznačenog podatka. Atributi nekog podatka se predstavljaju pomoću brojeva te se svaki atribut prikazuje kao jedna dimenzija prostora. Stoga svaki podatak ima svoj položaj u prostoru.

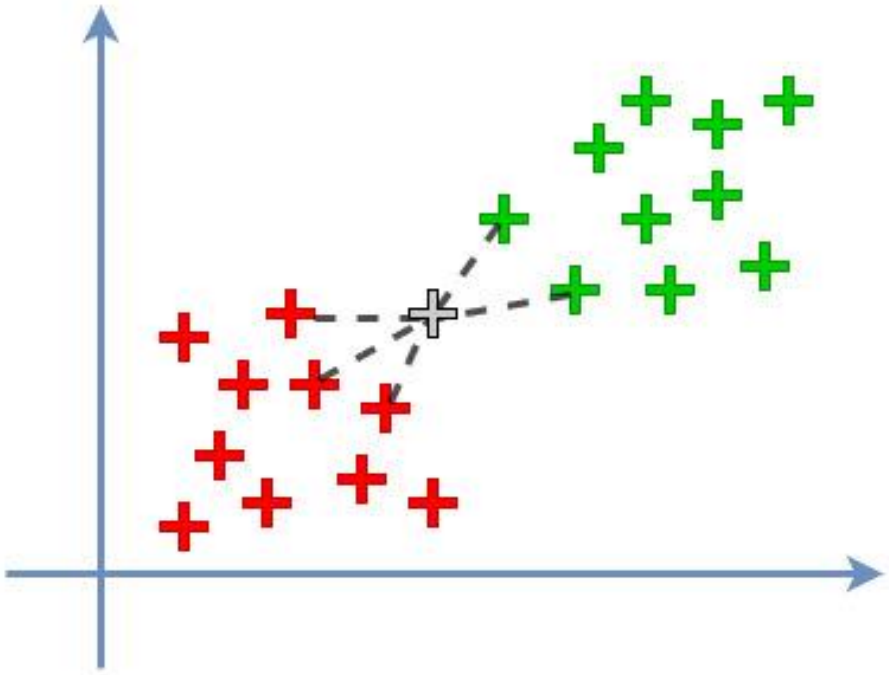
Postoje dva bitna koncepta koje je potrebno objasniti kada se radi s ovim algoritmom: varijabla K i metoda računanja udaljenosti. Varijabla K u imenu algoritma označava broj najbližih susjeda koje provjeravamo te je proizvoljne vrijednosti, a kao najčešća vrijednost se uzima broj 5 iako neki autori predlažu da bi K trebao biti jednak korijenu broju elemenata podataka za treniranje. (Zhang, 2016.)

Drugi bitan koncept je računanje udaljenosti. U slučaju na slici 2., kada podatak ima 2 atributa radi se o koordinatnom sustavu u ravnini te je udaljenost moguće izračunati na više načina. Problem nastaje prilikom rada sa stvarnim podacima koji mogu imati deset, dvadeset ili više atributa te se radi o prostoru koji je ljudima nezamisliv. Stoga standarni KNN algoritam koristi jednadžbu euklidske udaljenosti:

$$D(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

gdje su p i q elementi čiju udaljenost želimo izračunati, a n broj dimenzija odnosno atributa. Postoje također druge metode računanja udaljenosti od kojih vrijedi spomenuti Manhattan udaljenost. (Zhang, 2016.)

Na slici 2. se može vidjeti prikaz modela algoritma KNN. U ovom je slučaju, radi pojednostavljivanja prikaza, postavljeno da postoje dvije klase podataka (označene crvenom i zelenom bojom) koje imaju dva atributa (X i Y os). Novi je podatak (siva boja) potrebno klasificirati u crvenu ili zelenu skupinu. Algoritam KNN stoga računa udaljenost između tog novog podatka i ostalih elemenata te, u ovom slučaju, uzima 5 najbližih susjeda. Od tih 5 susjeda, 3 pripada crvenoj klasi, a 2 zelenoj stoga algoritam zaključuje da novi podatak pripada crvenoj klasi.



Slika 2: Prikaz modela KNN

3.3. Metoda potpornih vektora (engl. Support Vector Machines, SVM)

Metoda potpornih vektora bazirana je na principu minimizacije strukturnog rizika iz teorije strojnog učenja. Ideja minimizacije strukturnog rizika je pronaći hipotezu h za koju možemo sa sigurnošću tvrditi da ima minimalnu istinsku pogrešku. Istinska pogreška je vjerojatnost da će h napraviti grešku na nepredviđenom i slučajno odabranom testnom primjeru. (Joachims, 1998.)

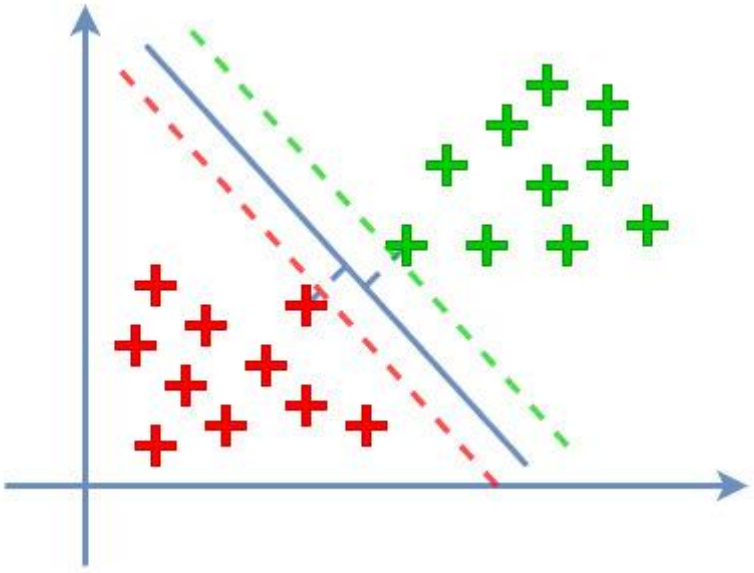
Važno svojstvo SVM-a je njihova mogućnost da učenje čine neovisnim o dimenzijama svojstvenog prostora jer kao i kod KNN-a podaci se postavljaju u prostoru gdje je broj dimenzija jednak broju atributa podataka. Glavna razlika u odnosu na KNN, gdje se računa udaljenost od k najbližih susjeda, je postavljanje optimalne granice između klasa podataka u svojstvenom prostoru koja se naziva hiperravnina maksimalne margine (engl. maximum margin hyperplane). Položaj takve granice u prostoru je određen graničnim podacima koje nazivamo potporni vektori kao što je prikazano na slici 3. Granični podaci su oni podaci najsličniji drugoj klasi.

Na primjer, možemo zamisliti problem klasifikacije jabuka i naranča prikazano na slici 3. Najreprezentativniji element svake klase nalazi se u donjem lijevom, za crvenu klasu odnosno jabuke, ili gornjem desnom za zelenu klasu, odnosno naranče. U praksi bi to značilo da se dolje lijevo, na primjer, nalazi jabuka savršenog oblika, crvene boje, optimalne veličine itd. U gornjem

desnom kutu bi se stoga nalazila naranča optimalnog oblika i veličine te narančaste boje.

Elemente bliže granici možemo onda zamisliti kao jabuku narančaste boje ili naranču zelene boje te takve elemente nazivamo potpornim vektorima. Razlog takvog naziva je što u dvodimenzionalnom prostoru kao na slici ti elementi jesu točke, ali u prostorima s nekoliko tisuća dimenzija ti elementi su opisani preko vektora. Granica između ovih dviju klasa je stoga određena jednakom udaljenošću između takvih potpornih vektora.

Prilikom unosa podatka za klasifikaciju on se pozicionira u tom n-dimenzionalnom prostoru te se provjerava s koje strane granice se nalazi. Ako se, u primjeru na slici 3, novi podatak nađe na donjem lijevom dijelu, podatak će biti svrstan u crvenu klasu.



Slika 3: Prikaz modela SVM

Joachims (Kako ga Sebastiani parafrazira, 2002.) tvrdi da postoje dvije važne prednosti za korištenje SVM-a za klasifikaciju tekstualnih podataka:

- odabir pojmova nije potreban jer SVM-ovi imaju tendenciju otpornosti na prenaučenost i moguće ih je skalirati na velik broj dimenzija;
- nema potrebe za ljudskim ili strojnim podešavanjem validacijskog seta podataka, jer postoje teorijski motivirane standardne postavke parametara koje su pokazale da pružaju maksimalnu efikasnost

3.4. Naivni Bayes-ov algoritam (engl. Naive Bayes algorithm, NB)

Multinominalni Naive Bayes (NB) model je metoda učenja zasnovana na vjerojatnosti. Iz perspektive klasifikacije tekstualnih dokumenata ta metoda predlaže vjerojatnost da dokument d pripada klasi c po formuli:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

gdje $P(c|d)$ predstavlja uvjetnu vjerojatnost da se pojam t_k pojavljuje u dokumentu klase c. $P(t_k|c)$ se interpretira kao mjera koliko dokaz t_k pridonosi tvrdnji da je c ispravna klasa. $P(c)$ je apriorna vjerojatnost pojavljivanja dokumenta u klasi c. Ako pojmovi u dokumentu ne daju jasni odabir klase, odabiremo onu klasu s većom a priori vjerojatnosti. Oznake $\langle t_1, t_2, \ldots, t_{n_d} \rangle$ su tokeni u d koji su dio vokabulara koji koristimo za klasifikaciju, a n_d broj takvih tokena u dokumentu. Na primjeru dokumenta s jednom rečenicom *Beijing and Taipei join the WTO,* oznake $\langle t_1, t_2, \ldots, t_{n_d} \rangle$ mogu biti $\langle Beijing, Taipei, join, WTO \rangle$ uz $n_d = 4$, ako riječi *and* i *the* gledamo kao stop riječi. (Manning, Raghavan, Schütze, 2009, str. 258.)

Ovaj se algoritam može lakše shvatiti intuitivno pomoću slike 4. Zamislimo ponovo dvije klase, crvenu s 10 elemenata i zelenu s 30 elemenata. Pretpostavimo također da X – os predstavlja godine, a Y – os godišnju plaću. Crvena boja stoga može predstavljati osobe koje hodaju do radnog mjesta, a zelena one koji se voze automobilom. Unosom novog podatka, označenog sivom bojom, zanima nas pripada li on skupini koja hoda ili koja vozi.

Pomoću Bayesovom teorema, možemo izračunati kolika je vjerojatnost da ta osoba pripada crvenoj klasi. Formula kojom ćemo to izračunati je sljedeća:

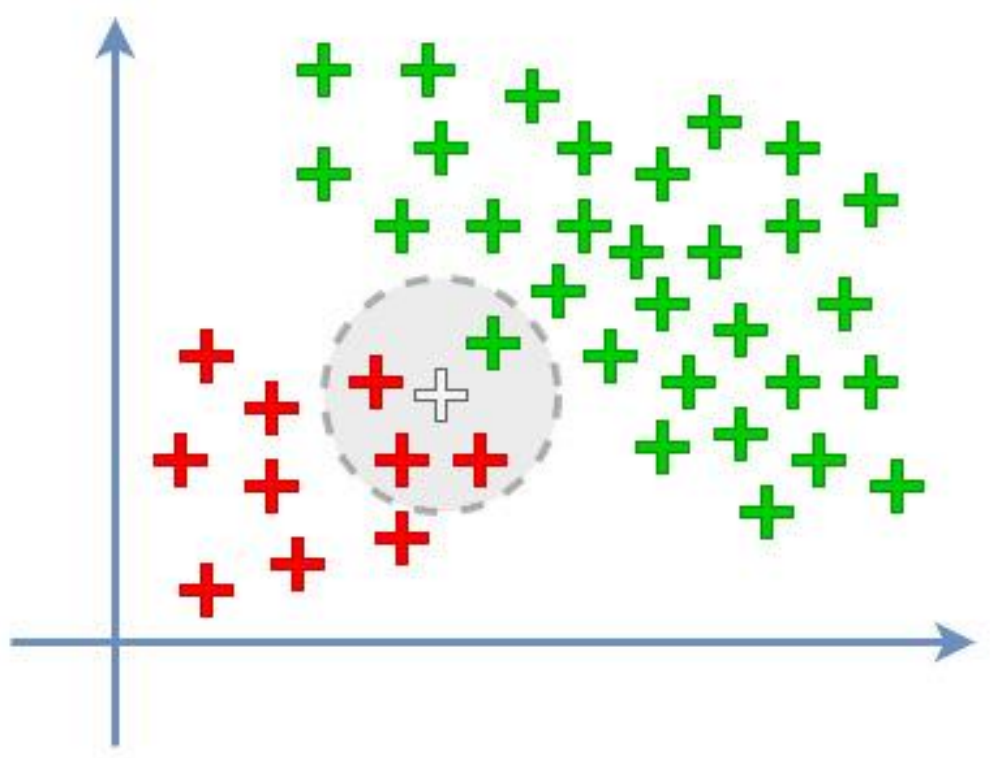
$$P(Hoda|X) = \frac{P(X \mid Hoda) * P(Hoda)}{P(X)}$$

$P(Hoda|X)$ u ovom slučaju predstavlja vjerojatnost da osoba s nekim vrijednostima atributa X hoda do svog radnog mjesta te se naziva aposteriorna vjerojatnost. $P(Hoda)$ se naziva apriorina vjerojatnost, predstavlja vjerojatnost da bilo koja osoba hoda te ju možemo izračunati tako da podijelimo broj osoba koje hodaju s brojem ukupnih zapažanja što je u ovom slučaju jednako $\frac{10}{40}$. $P(X)$ se zove marginalna vjerojatnost, a računamo ju tako da prvo oko novog podatka

odredimo radijus, na slici 4. iscrtana linija. Radijus predstavlja raspon atributa. Na primjer, sve osobe između 25 i 35 godina s plaćom između 60 i 80 tisuća kuna godišnje. Marginalna vjerojatnost stoga predstavlja kolika je vjerojatnost da novi element pripada toj skupini. Računamo ju na način da zbrojimo elemente u tom radijusu te ih podijelimo s ukupnim brojem zapažanja što je u slučaju na slici 4 jednako $\frac{4}{40}$. $P(X \mid Hoda)$ predstavlja vjerodostojnost, odnosno vjerojatnost da netko tko hoda ima vrijednosti atributa jedanke X. Znači kolika je vjerojatnost da neka osoba koja hoda ima vrijednosti atributa označenih iscrtanom linijom. U ovom primjeru ta vjerojatnost je jednaka $\frac{3}{10}$. Sada možemo izračunati aposteriornu vjerojatnost $P(Hoda|X)$:

$$P(Hoda|X) = \frac{\frac{3}{10} * \frac{10}{40}}{\frac{4}{40}} = 0.75$$

Na kraju zaključujemo da je vjerojatnost da novi element pripada crvenoj klasi 75% te ga zbog toga klasificiramo kao crveni element, odnosno kao osobu koja hoda do svog radnog mjesta. Vjerojatnost pripadanja zelenoj klasi nije potrebno računati jer kada postoje samo dvije klase zbroj vjerojatnosti pripadanja je uvijek jednak 1. U slučaju da postoji više od dvije klase potrebno je izračunati vjerojatnost za svaku od njih.



Slika 4: Prikaz modela NB algoritma

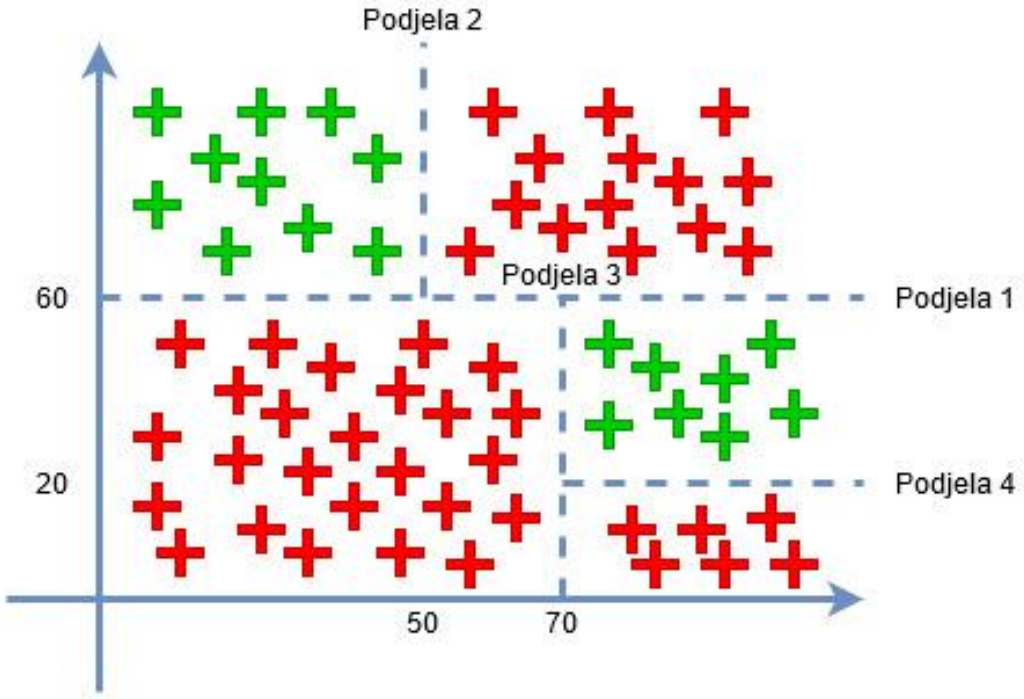
Manning, Raghavan i Schütze (2009) također smatraju da je efikasnost jedan od razloga popularnosti NB algoritma za klasifikaciju tekstualnih datoteka. Razlog efikasnosti je linearnost kompleksnosti treniranja i testiranja modela prilikom prolaska kroz podatke, a zato što se kroz podatke mora proći barem jednom, predlažu da NB algoritam ima optimalnu vremensku kompleksnost.

3.5. Klasifikacija pomoću stabla odlučivanja (engl. decision tree classification)

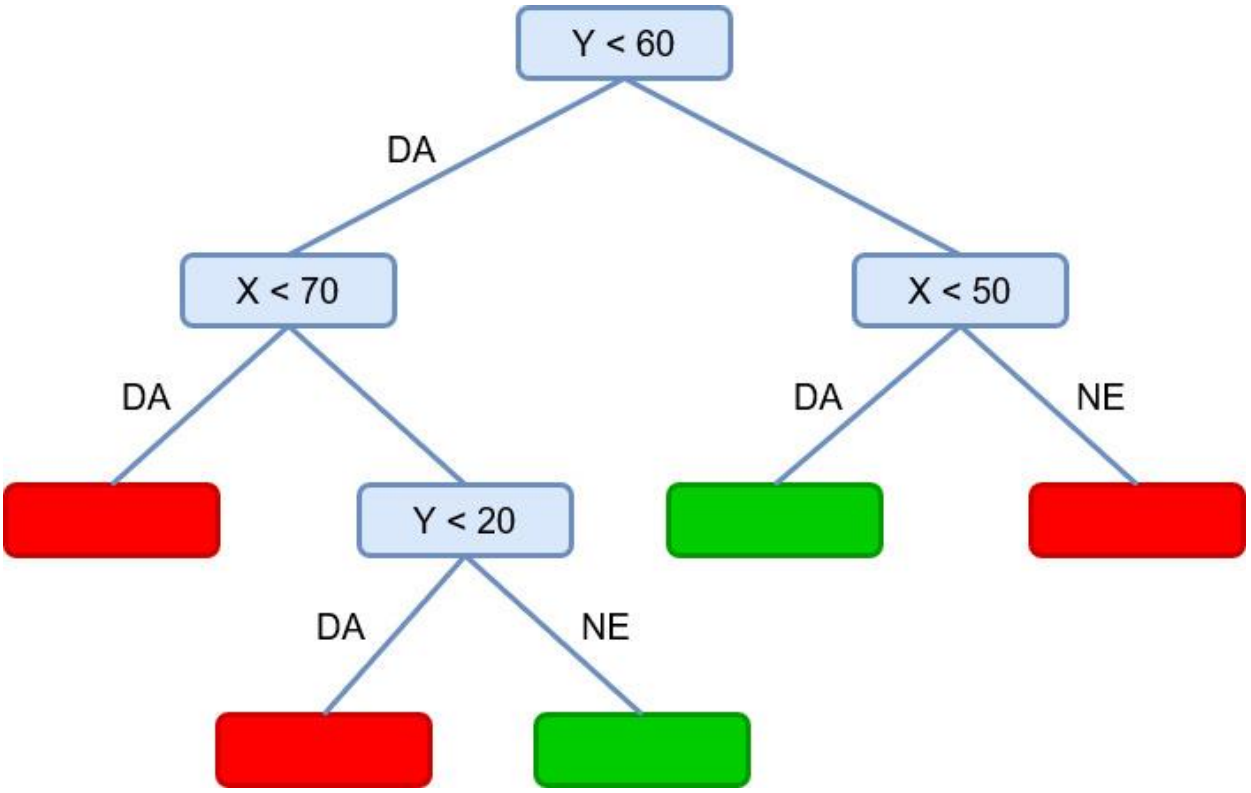
Mitchell (Kako ga parafrazira Sebastiani, 2002.) opisuje klasifikator pomoću stabla odlučivanja kao stablo u kojem su unutarnji čvorovi označeni pojmovima, grane koje izlaze iz tih čvorova su označene testovima na ponderima koje pojam ima u testnom dokumentu, a listovi su označeni kategorijama. Takav klasifikator kategorizira dokument d_j putem rekurzivnog testiranja pondera, koji pojmovi označavajući unutarnje čvorove imaju u vektoru d_j, dok ne naiđu na čvor lista čija se oznaka zatim dodjeljuje d_j. Većina takvih klasifikatora koriste binarnu reprezentaciju dokumenata, te se stoga sastoje od binarnih stabala.

Karim i Rahman (2013) jednostavnijim rječnikom objašnjavaju klasifikator pomoću stabla odlučivanja kao algoritam rudarenja podataka koji kreira upute koje korak po korak određuju pravila za odlučivanje izlazne vrijednosti nove instance podataka. Stablo koje stvara je upravo to: stablo u kojem svaki čvor predstavlja mjesto gdje se mora donjeti odluka bazirana na ulaznom podatku, nakon toga se kreće na idući čvor sve dok se ne dođe do lista koji govori predviđenu izlaznu vrijednost.

Intuitivno, ovaj algoritam prvo dijeli set podataka u nekoliko iteracija, kao što je prikazano na slici 5, maksimizirajući pritom broj elemenata neke kategorije u svakom nastalom prostoru. Na bazi tih podjela se kasnije stvara stablo odluke na način da je prva podjela prva odluka u stablu, prikazano na slici 6. Na ovom primjeru prva je podjela razdijelila set podataka po Y – osi u broju 60, stoga je prva odluka prikazana u stablu na slici 7: je li novi podatak manji od 60. Ako jest krećemo se lijevo, a ako nije krećemo desno. Na taj se način kreće po stablu dok se ne dođe do nekog od čvorova lista označenih crvenom i zelenom bojom. Kada se dođe do čvora lista njegova oznaka postaje oznaka novog podatka.



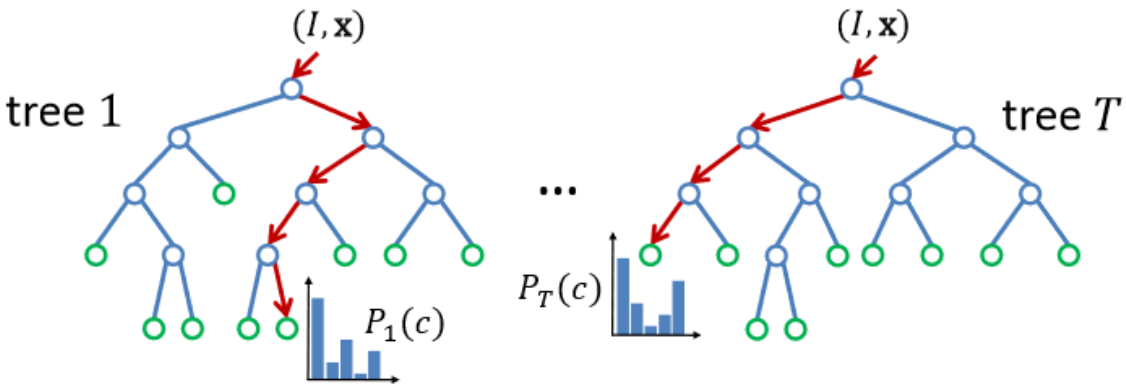
Slika 5: Prikaz podjele skupa podataka kod stabla odluke



Slika 6: Prikaz stabla odluke

3.6. Klasifikacija slučajnom šumom (engl. Random forest classification)

Shotton, Fitzgibbon, Cook, Sharp, Finocchio, Moore, Kipman i Blake (2016.) u svom radu, referencirajući mnoge autore, navode da su se slučajna stabla odluke i šume pokazali kao brzi i efektivni višeklasni klasifikatori te da mogu biti implementirani efikasno na GPU-u. Kao što je prikazano na slici 7, šuma je ansambl T stabala odluke od kojih se svaki sastoji od čvorova odluka, označenih plavom bojom, i listova,označenih zelenom bojom. Crvena strelica pokazuje različite putanje kojim svako stablo dolazi do predviđanja za neki podatak. Za donošenje konačne predikcije svako stablo kao izlazni podatak donosi neko svoje predviđanje, a na kraju glasanjem donose konačnu odluku.



Slika 7: Prikaz slučajne šume (preuzeto iz rada Shotton et al. (2016))

Liaw i Wiener (2002.) kažu da se u poslijednje vrijeme povećava interes za metodama koje generiraju nekoliko klasifikatora te agregiraju njihove rezultate, odnosno ansamblima. Dvije vrlo poznate metode su boosting i bagging stabla odluke. Kod boostinga, uzastopna stabla daju dodatnu težinu točakama pogrešno predviđenim od strane prijašnjih prediktora. Na kraju, težinsko glasanje se poduzima za konačno predviđanje. Kod bagginga, uzastopna stabla ne ovise jedna o drugima, svako je nezavisno konstruirano koristeći bootstrap uzorak skupa podataka. Na kraju, jednostavna većina glasova stabala odluke je potrebna za predviđanje.

Algoritam slučajnih šuma izradio je Breiman u radu *Random Forests* (2001.) te se bazira na verziji bagging metode. Algoritam gradi velik broj različitih stabala koja glasaju u slučaju klasifikacije ili pronalaze srednju vrijednost u slučaju regresije. Iz skupa za učenje se dobiva novi uzorak koji je po veličini jednak originalnom. Zatim se nad svakim stablom izrađuje stablo odluke

prema pseudokodu:

1. Za b=1 do B , gdje je B broj stabala

    a) Izvadi bootstrap uzorak Z* veličine N iz skupa za učenje

    b) Izgradi stablo slučajne šume trenirano na uzorku bootstrap, rekurzivno ponavljajući sljedeće korake za svaki čvor u stablu

        i.) Izaberi m slučajnih varijabli od p mogućih.

        ii.) Pronađi najbolju varijablu i vrijednost za podjelu među odabranih m.

        iii.) Podijeli čvor u lijevi i desni.

2. Spremi skup stabala $\{T_b\}_1^B$.

3. Prilikom predviđanja

    a.) Za regresiju : $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(X)$

    b.) Za klasifikaciju:

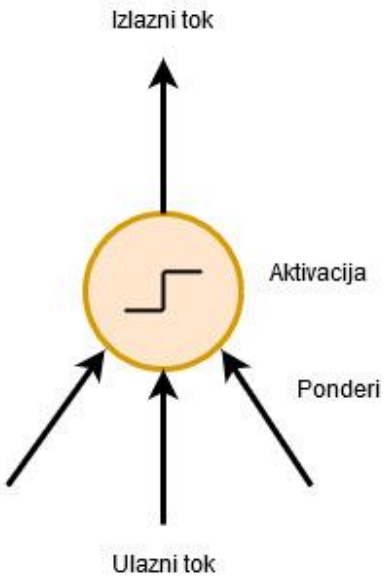
$$\hat{c}_{rf}^B(x) = najviše\ glasova\ \{\hat{c}_b(x)\}_1^B,\ gdje\ je\ \hat{c}_b(x)\ predikcija\ pojedinog\ stabla$$

3.7. Duboko učenje (engl. deep learning)

Duboko učenje je grana strojnog učenja koja proučava kako jednostavni modeli bioloških mozgova mogu biti korišteni za rješavanje teških računalnih problema kao što je prediktivno modeliranje. Cilj nije kreirati realistični model mozga već algoritme i strukture podataka koji se mogu koristiti za rješavanje kompleksih zadataka, a koji su motivirani modelom bioloških mozgova. (Brownlee, 2020)

Brownlee (2020) također navodi da snaga neuronskih mreža dolazi iz sposobnosti učenja reprezentacije skupa podataka za treniranje i kako najbolje takav skup podataka povezati sa željenom izlaznom vrijednošću.

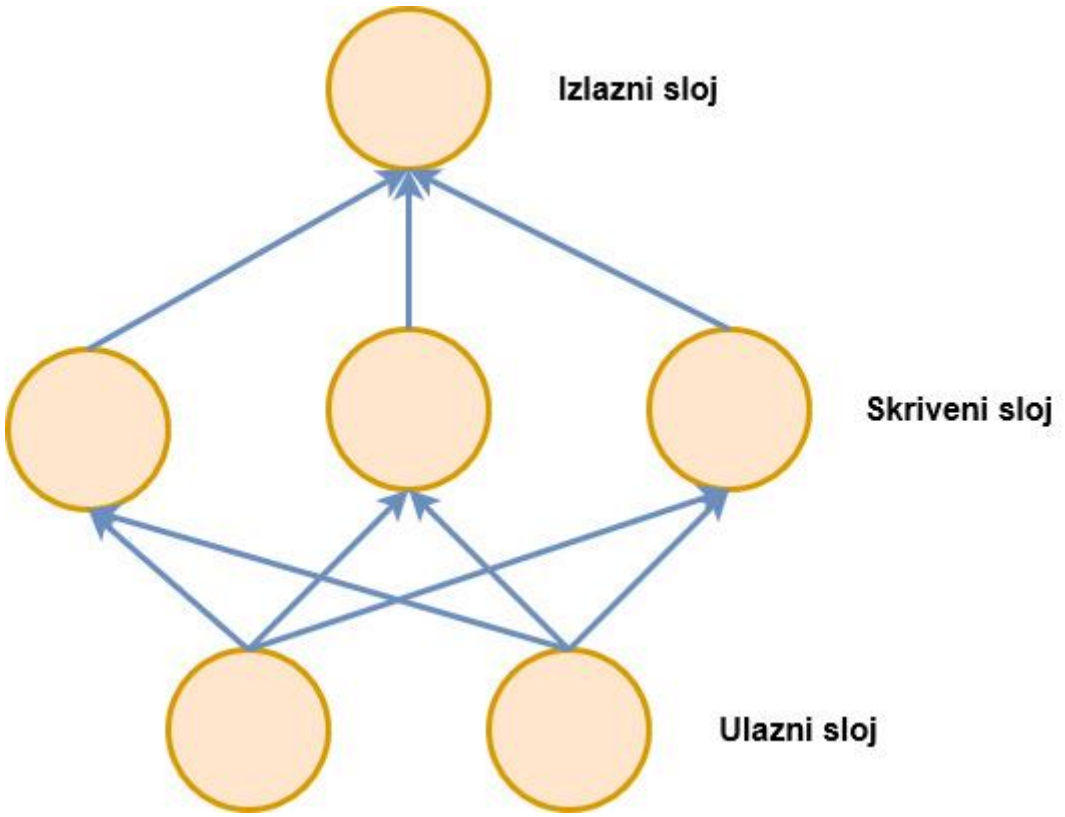
Svaka neuronska mreža se bazira na neuronu koji je u suštini jednostavna računska jedinica sa ponderiranim ulaznim signalima iz kojih stvara izlazni signal pomoću funkcije aktivacije, prikazano na slici 8. Ta funkcija aktivacije određuje hoće li se neuron aktivirati, odnosno u kojoj mjeri će se aktivirati, i poslati izlazni signal. Brownlee (2020) navodi da je poslijednje vrijeme ispravljačka aktivacijska funkcija (engl. *rectifier activation function*) pokazala najbolje rezultate.



Slika 8: Prikaz neurona

Svaka se neuronska mreža sastoji od tri vrste slojeva, prikazanih na slici 9. Ulazni sloj ili vidljivi sloj je prvi koji prima podatke, a najčešći broj neurona u tom sloju je jednak broju stupaca, odnosno atributa, u skupu podataka. Ti se neuroni razlikuju od onih ranije opisanih jer njihova je zadaća samo primiti podatke i slati ih idućem sloju. Skriveni sloj sadrži neurone u točno onom

ranije opisanom smislu. Zadnji je izlazni i njegova je zadaća poslati izlaznu vrijednost cijele mreže. Dizajn ovog sloja uvelike ovisi o željenom obliku izlaznih podataka. Primjerice za binarnu klasifikaciju je potreban samo jedan neuron, a za višeklasnu klasifikaciju je potrebno onoliko neurona koliko ima klasa podataka. Dok je za regresiju potreban samo jedan neuron bez funkcije aktivacije.



Slika 9: Neuronska mreža

Ulazni podaci za neuronske mreže moraju biti numeričke vrijednosti. Kvalitativni atributi, na primjer muško ili žensko, se enkodiraju na način da se za svaku vrijednost tog atributa doda po jedan stupac, 2 stupca u slučaju spola, te se u stupac koji označava željenu vrijednost stavlja 1, a ostali stupci imaju vrijednost 0. Na jednak se način enkodiraju vrijednosti klasa kojima se označavaju ulazni podaci. Ostale se numeričke vrijednosti standardiziraju (engl.standardization), odnosno poprimaju vrijednosti između -1 i 1, ili normaliziraju (engl. normalization), tj. skaliraju u raspon od 0 do 1.

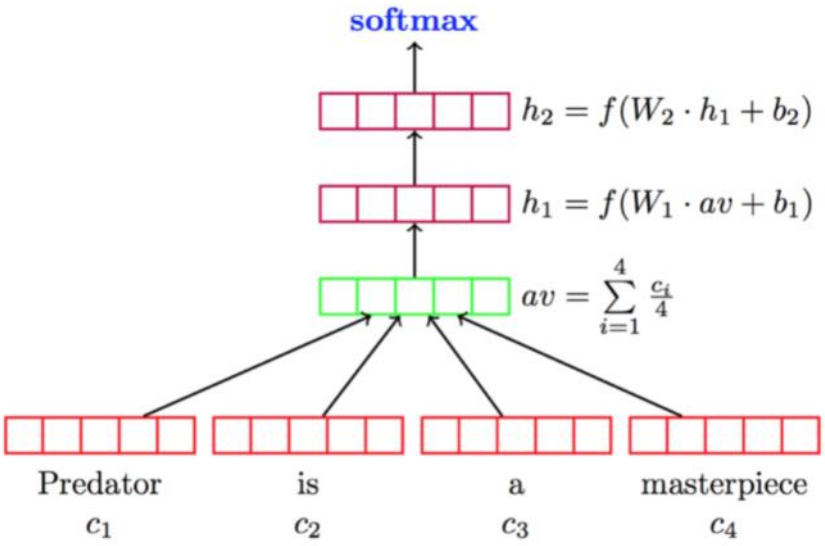
Brownlee (2020) navodi da se klasični te također i danas preferiran način treniranja neuronskih mreža zove stohastički gradijentni spust (engl. stochastic gradient descent). Cilj *stohastičkog gradijentnog spusta* je pronaći globalni minimum funkcije troška kako bi se povećala točnost mreže. Neuronska mreža učitava podatke redak po redak aktivirajući pritom neurone

mreže, izlazna se vrijednost zatim uspoređuje sa očekivanom vrijednošću i računa se greška. Kako ta promjena ne bi utjecala samo na izlazni sloj, ona se propagira kroz neuronsku mrežu unazad, sloj po sloj, te se ponderi ažuriraju ovisno o razini sudjelovanja u greški. Taj se algoritam naziva *Backpropagation* algoritam. Proces se ponavlja za sve primjere u skupu podataka za treniranje te se jedan takav prolaz kroz sve podatke naziva epoha. Mreža može biti trenirana proizvoljnim brojem epoha.

3.7.1. Unaprijedna neuronska mreža (engl. Feed – Forward Neural Network)

Minaee, Kalchbrenner, Cambria, Nikzad, Chenaghlu i Gao (2020), unaprijedne neuronske mreže spominju kao jedne od najjednostavnijih modela dubokog učenja za tekstualnu reprezentaciju. Unatoč tome, dalje navode, postigli su veoma veliku preciznost na mnogim referentnim mjerilima. Takvi modeli na tekst gledaju kao vreću riječi (engl. bag of words, BoW). Za svaku riječ oni uče vektorsku reprezentaciju koristeći embedding model poput *word2vec* ili *Glove*, uzimaju sumu ili prosjek vektora kao reprezentaciju teksta, šalju ih kroz jedan ili više unaprijednih slojeva (engl. Feed forward layers), znanih također kao višeslojni perceptori (engl. Multi-layer Perceptrons, MLP), te na kraju vrše klasifikaciju na zadnjem sloju koristeći klasifikator poput logističke regresije, Naive Bayes-a ili potpornih vektora.

Minaee et al. (2020) navode kao primjer ovakvih modela tzv. *Deep Average Network* (DAN), čiju arhitekturu prikazuju na slici 10. Unatoč jednostavnosti, DAN pokazuje preformanse bolje od mnogih drugih veoma sofisticiranih modela koji su eksplicitno naučeni sadržaju teksta. Na primjer, DAN ima bolje preformanse od sintaktičkih modela na skupovima podataka s velikom sintaktičkom varijancom. Referencirajući se na rad A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, i T. Mikolov, “Fasttext. zip: Compressing text classification models” (2016), gdje je predložen jednostavan i efikasan tekstualni klasifikator zvan fastText. Poput DAN-a, fastText gleda riječi kao BoW. Ali za razliku od DAN-a, fastText koristi metodu skupa n-grama (engl. Bag of n-grams) kako bi se uvele dodatne značajke koje nose informacije o poretku riječi u tekstu. Takav se pristup pokazao veoma efikasan u praksi te je ostvario rezultate usporedive s metodama koje eksplicitno koriste poredak riječi.



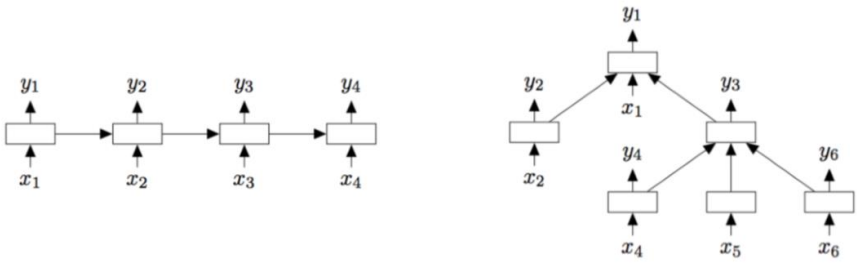
Slika 10: Arhitektura DAN-a (preuzeto iz rada Minaee et al. (2020))

3.7.2. Modeli bazirani na povratnim neuronskim mrežama (engl. Recurrent neural networks)

Modeli bazirani na povratnim neuronskim mrežama (RNN) na tekst gledaju kao sekvencu riječi i namjera im je pohvatati zavisnosti među riječima i strukturu teksta za klasifikaciju. Minaee et al. (2020) navode da takvi standardni modeli ne rade dobro i često imaju niže preformanse od unaprijednih neuronskih mreža. Među mnogim varijantama RNN-a, metoda Ćelije s dugoročnom memorijom (engl. Long Short-Term Memory, LSTM) je najpopularnija metoda koja je dizajnirana da bolje pohvata dugoročne zavisnosti. Takve su mreže uvele memorijsku ćeliju kako bi zapamtile vrijednosti kroz arbitrarne intervale vremena, te tri propusnice (novog ulaza, izlaza, zaboravljanja) kako bi se regulirao tok informacija u i van ćelije.

Tai, Socher i Manning (2015) zbog limitacija dvosmjerne i višeslojne LSTM arhitekture, koje mogu primati striktno sekvencijalno propagiranje informacija, predlažu arhitekture bazirane na stablima, tzv. *Child-Sum Tree-LSTM* i *N-ary Tree LSTM*. Obje varijante, kako navode, dozvoljavaju bogatiju mrežnu topologiju gdje svaka jedinica LSTM mreže može inkorporirati informacije iz višestrukih jedinica djece.

Grafička usporedba tradicionalne lančane i stablaste LSTM mreže sa arbitrarnim faktorom grananja može se vidjeti na slici 11.



Slika 11: Lančana i stablasta arhitektura LSTM mreže (preuzeto iz rada Minaee et al. (2020))

3.7.3. Modeli bazirani na konvolucijskim neuronskim mrežama (engl. Convolutional Neural Network – Based models, CNN – Based models)

Le Cun, Bottou, Bengio i Haffner (1998) navode da su RNN modeli trenirani prepoznati uzorke kroz vrijeme dok CNN modeli uče prepoznati uzorke u prostoru. Minaee et al. (2020) tvrde da RNN modeli dobro funkcioniraju u slučajevima označavanja vrsta riječi u korpusu (engl. POS tagging) ili u slučaju QA gdje je potrebno razumijavanje dalekosežne semantike, dok CNN funkcionira izvrsno na slučajevima gdje je potreno detektirati lokalne uzorke s nepromjenjivim pozicijama. Ti uzorci mogu biti ključne fraze koje izražavaju određenu emociju, primjerice *„želim“* ili temu poput *„ugrožene vrste“*. Stoga CNN modeli su postali jedan od najpopularnijih modela za klasifikaciju teksta.

Kim (2014.) je predložio nekoliko vrsta CNN mreža te izvršio njihovu usporedbu. Nasumični CNN (engl. CNN-rand) je njegov referentni model gdje su sve riječi nasumično generirane te zatim modificirane tokom treniranja. Statički CNN (engl. CNN-static) je model s unaprijed treniranim vektorima od word2vec. Sve su riječi, uključujući one nepoznate koje su nasumično inicijalizirane, statične i samo ostali parametri modela su naučeni. Nestatički CNN (engl. CNN-non-static) je isti kao i statički, a jedina je razlika prilagodba unaprijed treniranih vektora svakom zadatku. Zadnji je multikanalni CNN (engl. CNN-multichannel), koji predstavlja model s dva skupa vektora riječi gdje se svaki skup tretira kao kanal te se svaki filter primjenjuje na oba kanala, ali je gradijent propagiran unazad samo kroz jedan od kanala. Stoga model prilagođava samo jedan skup vektora dok drugi ostaje statičan. Oba su kanala inicijalizirana pomoću word2vec. Ti su modeli po Minaee et al.(2020) unaprijedili već veoma napredne tehnike analize emocija i klasifikacije pitanja.

4. Primjena

Programski je primjer izrađen za klasifikaciju prema osjećajima na kompleksnom skupu podataka sa različitim tipovima varijabli. Kod je izrađen u programskom jeziku Python u online okruženju Google Colaboratory koji omogućuje pokretanje koda na oblaku (engl. cloud). Za izradu modela korištene su biblioteke Tensorflow i Keras, uz numpy za rad na velikim nizovima i matricama, pandas za manipulaciju podacima te scikit-learn zato što sadrži mnogobrojne algoritme za klasifikaciju.

Kod je dostupan na lokaciji :
https://colab.research.google.com/drive/1HtC1wA3zMJ0QTqu55YjA_cxDeneFStwm#scrollTo=LCAHha5U9UkO

4.1. Okruženje

Google Colaboratory je besplatno online okruženje koji omogućuje pokretanje Python koda na oblaku uz pristup Google-ovim GPU-ima te dolazi s instaliranim bibliotekama poput Tensorflowa, Kerasa, PyTorcha i OpenCV. Okruženje je replicira Jupyter notebook, dobro poznat alat mnogim machine learning developerima, toliko da i sprema kod s ektenzijom .ipynb. Uz otvoreni intenet preglednik ostaje aktivan do 12 sati nakon pokretanja. Iz tih je razloga ovaj programski primjer izrađen pomoću Google Colaboratoy-ja.

4.1.1. Biblioteke

Tensorflow je biblioteka otvorenog koda za numeričko računanje i strojno učenje. Sadrži mnoštvo modela i algoritama strojnog i dubokog učenja. Jedna od glavnih prednosti je to što koristi efikasnost C++ - a u kojem je napisan, a jednostavan je za korištenje jer se poziva pomoću Python koda.

Keras je također biblioteka otvorenog koda napisana u Pythonu, a pokreće se preko Tensorflow-a. Razvoj Kerasa je baziran na omogućavanju brzog eksperimentiranja. Njime se minimizira količina koda koji je potrebno napisati što ga čini veoma laganim za naučiti. Njegovu je primjenu moguće primjetiti kod Netflixa, Ubera, Yelpa i mnogih drugih poznatih tvrtki.

Numpy je jedna od osnovnih Python biblioteka za strojno učenje koja omogućuje višedimenzionalne objekte, mnoge izvedene objekte poput matrica i skrivenih nizova, te mnoge operacije za rad na takvim objektima poput manipulacije oblika, matematičke operacije, logičke operacije, sortiranje, statističke operacije i mnoge druge.

Pandas je biblioteka za manipulaciju i analizu podataka. Veoma je optimiziran za pokretanje jer su bitniji djelovi koda napisani u programskom jeziku C. Sadrži brz i efikasan DataFrame objekt za manipulaciju podacima sa integriranim indeksiranjem, alate za čitanje i zapisivanje u raznim formatima, integrirano rukovanje podacima koji nedostaju te mnoge druge korisne alate.

Scikit-learn je besplatna biblioteka koja sadrži razne algoritme klasifikacije, regresije i klasteriranja poput potpornih vektora i slučajne šume. Poput pandasa, kritičniji djelovi koda su napisani u programskom jeziku C što scikit-learn čini veoma efikasnim.

4.2. Skup podataka

Za treniranje modela korišteni su podaci Internacionalnog istraživanja o prethodnicama emocija i reakcijama (engl. International survey on emotional antecedents and reactions, ISEAR). Podaci tog istraživanja su za model podijeljeni u sljedeće kategorije: kvalitativni atributi, ordinalni atributi, diskretni numerički atribut, tekstualni atribut i oznaka emocije.

U kvalitativne atribute u ovom slučaju ubrajamo: spol subjekta, njegovo religijsko opredjeljenje, prakticira li subjekt religiju, područje obrazovanja subjekta te država i grad stanovanja. Za te atribute pretpostavljamo da nemaju direktan utjecaj na način izražavanja ili osjećanja neke emocije, stoga su u modelu predstavljeni pomoću OneHotEncoder-a iz biblioteke scikit-learn. Ovom se metodom dodaje po jedan stupac za svaku moguću kategoriju atributa te se vrijednost definira oznakom 1 u stupcu određenom za tu specifičnu vrijednost kako je već objašnjeno u poglavlju 3.7. ovog rada.

Diskretni numerički atributi su oni koji mogu poprimiti samo konačno ili prebrojivo mnogo vrijednosti. U slučaju ovog skupa podataka, toj skupini atributa pripadaju godine subjekta te su one normalizirane, odnosno skalirane u rasponu od 0 do 1.

Ordinalni su atributi u svrhu ovog rada podijeljeni u dvije skupine, one koje imaju direktan utjecaj na neku emociju i one koji nemaju. U one koje nemaju se ubrajaju: očevo i majčino zanimanje, kada se dogodila ta emocija, koliko je osoba dugo osjećala tu emociju i koliko je

emocija bila intenzivna te se s njima postupa na jedak način kao i sa kvalitativnim atributima.

Ordinalni atributi s direktnim utjecajem na emocije su oni koji direktno opisuju stanje u kojemu se subjekt našao. Po ISEAR istraživanju rekacije su podijeljene u dvije skupine. Prva je skupina psiholoških simptoma u koje se ubrajaju:

- ergotropne reakcije (promjene u disanju, brži otkucaji srca, stezanje ili drhtanje mišića, znojni dlanovi) s ocjenama od 0 do 4,
- trofotropne reakcije (knedla u grlu, želučane tegobe, plakanje) s ocjenama od 0 do 3,
- te osjet temperature (osjećanje hladnoće i drhtavica, osjećanje topline i ugode, osjećaj vrućine) s ocjenama od -1 do 2 gdje 0 označava nikakve promjene.

Druga skupina ordinalnih atributa s direktnim utjecajem na predviđenu emociju su oni koji opisuju ekspresivno ponašanje subjekta u trenutku pojavljivanja neke emocije. U njih ubrajamo:

- pokrete tijela (povlačenje ili kretanje prema ljudima ili stvarima),
- neverbalne aktivnosti (smijanje, plakanje, druge promjene u izrazu lica, vikanje i druge promjene u glasu, te ostale promjene u govoru tijela),
- paralingvističke aktivnosti (melodičnost pričanja, poteškoće u govoru, promjene tempa pričanja),
- verbalne aktivnosti (moguće kategorije: tišina, kratke izjave, jedna ili dvije rečenice, duge izjave),
- pokušaj skrivanja ili kontroliranja emocija (1 – ne, 3 – puno, 0 – nije primjenjivo),
- je li situacija bila očekivana (1 – ne, 3 – poprilično, 0 – nije primjenjivo),
- je li događaj bio ugodan ili neugodan (1 – ugodan, 3 – neugodan, 0 – nije primjenjivo),
- koliko je događaj koji je izazvao emociju bio bitan za ostvarivanje neki želja, potreba ili ciljeva u trenutku kada se odvio, odnosno je li pomogao ili spriječio ostvarivanje istih (1 – pomogao, 3 – spriječio, 0 nije primjenjivo)
- je li po subjektivnom mišljenju situacija koja je izazvala emociju bila nezaslužena ili nepravedna (1 – ne uopće, 3 – jako nepravedna ili nezaslužena, 0 – nije

primjenjivo),

- tko je odgovoran za nastanak situacije (1 – subjekt, 2 – bliske osobe, 3 – ostale osobe, 4 – nepersonalni uzrok)
- kako bi subjekt evaluriao vlastitu sposobnost nošenja sa sitaucijom, odnosno njenim posljedicama u trenutku kada se situacija odvila (1 – nije mislio/la da je potrebno išta poduzimati, 2 – mislio/la je da može pozitivno utjecati na posljedice, 3 – mislio/la je da može izbjeći situaciju i izbjeći negativne posljedice, 4 – pravio/la se da se ništa bitno nije dogodilo i pokušao/la je misliti o nečem drugom, 5 – gledao/ la je na sebe kao bespomoćnu osobu dominiranu od strane događaja i njegoih posljedica)
- ako je situacija izazvana vlastitim ili tuđim ponašanjem, bi li subjektovi poznanici gledali na nju kao neprimjerenu ili nemoralnu (1 – ne, 3 – jako, 0 – nije primjenjivo)
- kako je taj događaj utjecao na subjektove osjećaje prema sebi, primjerice na samopoštovanje ili samopouzdanost (1 – negativno, 3 – pozitivno, 0 – nije primjenjivo)
- kako je događaj utjecao na odnose s ljudima koji su u njemu sudjelovali (1 – negativno, 3 – pozitivno, 0 – nije primjenjivo)

Iz opisa ovih ordinalnih atributa može se zaključiti da brojčane vrijednosti predstavljaju neku skalu, ali sva pitanja nemaju jednak raspon brojčanih vrijednosti odgovora. Kako bi modelu olakšali računanje i spriječili mogućnost da neko pitanje ima veću težinu zbog većeg raspona brojčanih vrijednosti, tu ćemo skupinu ordinalnih atributa skalirati kao i numerički atribut.

Podjele atributa su navedene u kodu na slici 12, a enkodiranje, odnosno sklairanje na slici 13.

```
# Podjela stupaca u kategorije: cat_cols - kategoričke vrijednosti(dalje podijeljeno na stupce
#                               koji se enkodiraju u vektore i integer stupce koji se skaliraju)
#                               num_cols - numerički stupac koji se skalira (godine subjekta)
#                               txt_cols - tekstualni stupac
#                               y_cols - stupac s kategorijama

enc_cols = ['CITY', 'COUN', 'SEX', 'RELI', 'PRAC', 'FOCC', 'MOCC', 'FIEL', 'WHEN', 'LONG', 'INTS',]
int_cols =['ERGO', 'TROPHO', 'TEMPER', 'EXPRES', 'EXP1', 'EXP2', 'EXP10', 'MOVE', 'PARAL', 'CON',
           'EXPC', 'PLEA', 'PLAN', 'FAIR', 'CAUS', 'COPING', 'MORL', 'SELF', 'RELA', 'VERBAL']
cat_cols = enc_cols + int_cols
num_cols = ['AGE']
txt_cols = ['SIT']
y_cols = ['EMOT']
cols = cat_cols + num_cols + txt_cols
```

Slika 12: Podjela atributa

```
# Skaliranje stupca godina između 0 i 1
ns = MinMaxScaler()
trainNum = ns.fit_transform(train[num_cols])
testNum = ns.transform(test[num_cols])

# Enkodiranje u vektore stupce koji nemaju međusobne povezanosti
ohe = OneHotEncoder().fit(df[enc_cols])
trainCat = ohe.transform(train[enc_cols]).toarray()
testCat = ohe.transform(test[enc_cols]).toarray()

# Skalira između 0 i 1 stupce sa integer vrijdnostima koji direktno utječu na predviđanje
ns2 = MinMaxScaler()
trainCat = ns2.fit_transform(train[int_cols])
testCat = ns2.transform(test[int_cols])

# Spaja numeričke i kategorijske stupce
trainX = np.hstack([trainNum, trainCat])
testX = np.hstack([testNum, testCat])

# Enkodira Y vrijednosti
catEncoder = LabelBinarizer()
trainY = catEncoder.fit_transform(y_train[y_cols])
testY = catEncoder.transform(y_test[y_cols])
```

Slika 13: Predobrada podataka

Kao što je opisano u poglavlju 3.7., ulazni podaci neuronske mreže moraju biti brojčane vrijednosti. Da bi tekst pretvorili u vrijednosti koje neuronska mreža može čitati, prvo ga moramo počistiti od svih neželjenih znakova, poput punktuacije, kako bi ostala samo velika i mala slova. Zatim sva slova pretvaramo u mala te koristeći *PorterStemmer* izdvajamo sve nastavke riječi da bi koristili samo korijen riječi. Istovremeno, zanemarujemo sve riječi koje ne nose značenje (engl. stopwords) poput *the, a, an*, *in,* itd. Kada je tekst počišćen, poziva se metoda *adapt* sloja TextVectorization koja na temelju cijelog skupa tekstualnih atributa stvara riječnik analizirajući frekvenciju pojavljivanja riječi. U ovom slučaju zbog manje količine tekstualnih atributa stvaramo riječnik od 20 000 riječi koji se primjenjuje na svakom elementu skupa podataka. Svaki element skupa atributa nakon toga postaje vektor sa 128 vrijednosti, gdje svaka vrijednost predstavlja cjeli broj određen tom riječi iz riječnika. Razlog odabira vrijednosti 128 je to što se u podaciama pojavljuju samo kraći tekstualni opisi te ova vrijednost omogućuje zadržavanje cjelovitosti teksta, a istovremeno smanjuje vrijeme potrebno za treniranje. Na primjer, s veličinom 300 vrijeme potrebno za jednu epohu je oko jedne minute, a s veličinom 128 to trajanje pada na otprilike 30 sekundi po epohi. Prikaz koda se nalaz na slici 14.

```
#Čišćenje teksta
for i in range (0, len(df)):
    sit = re.sub('[^a-zA-Z]', ' ', df['SIT'][i])
    sit = sit.lower()
    sit = sit.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    sit = [ps.stem(word) for word in sit if not word in set(all_stopwords)]
    sit = ' '.join(sit)
    corpus.append(sit)

#Kreiranje riječnika
vectorize_layer = TextVectorization(
    max_tokens=max_features,
    output_mode="int",
    output_sequence_length=sequence_length,
)

#Prilagodba riječnika na cijelom skupu podataka
vectorize_layer.adapt(corpus)

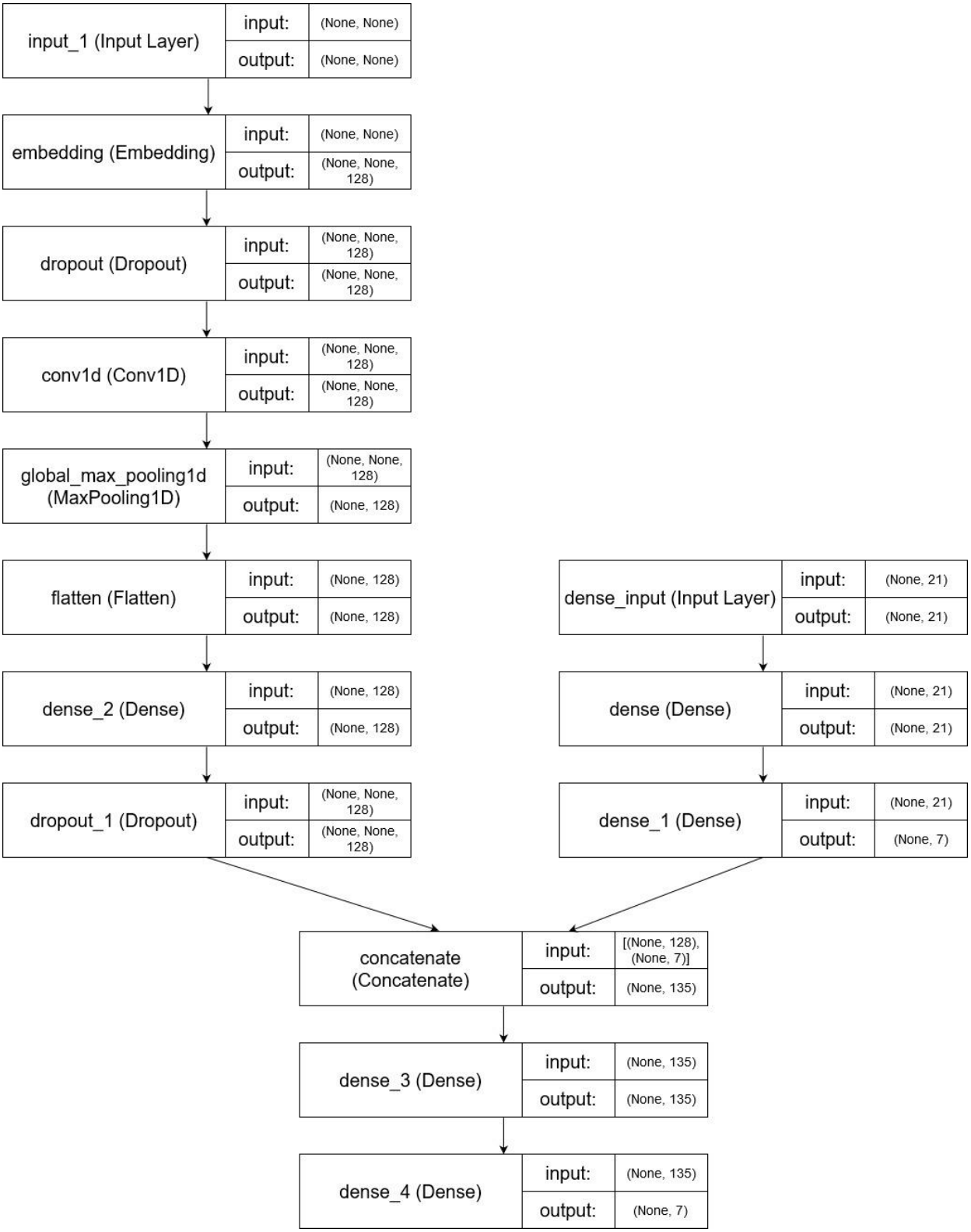
#Podjela na set za treniranje i testiranje
train, test, y_train, y_test, trainSit, testSit = train_test_split(df, y_df, corpus, test_size=0.25)

#Vektoriziranje skupova za treniranje i testiranje
trainSit = vectorize_layer(trainSit)
testSit = vectorize_layer(testSit)
```

Slika 14: Predobrada teksta

4.3. Model

Ideja ovog modela dolazi iz načina na koji ljudi percipiraju svijet. Ovaj skup podataka se može shvaćati kao komunikacija s drugom osobom. U svakom dijalogu ljudi svjesno percipiraju informacije koje druga osoba u dijalogu prenosi te podsvjesno neverbalne znakove te osobe. Stoga, ranijim objašnjenjem podataka u ovom radu, možemo doći do zaključka da su numerički i kategorički atributi dio neverbalne, a sam opis situacije dio verbalne komunikacije. Također, ljudski mozak sadrži odvojene centre za procesuiranje zvuka i slike te odvojeni centar za donošenje odluka. Zbog tih se razloga ovaj model sastoji od 3 dijela: jednostavni MLP za procesuiranje numeričkih i kategoričkih vrijednosti, CNN za procesuiranje teksta te dio koji prima izlazne vrijednosti prijašnja dva dijela te daje konačno predviđanje. Grafički prikaz modela može se vidjeti na slici 15, a objašnjenje slojeva na slici 16.



Slika 15: Grafički prikaz modela

```
Model: "functional_3"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, None)]            0
_________________________________________________________________________________________________
embedding (Embedding)        (None, None, 128)         2560000     input_1[0][0]
_________________________________________________________________________________________________
dropout (Dropout)            (None, None, 128)         0           embedding[0][0]
_________________________________________________________________________________________________
conv1d (Conv1D)              (None, None, 128)         49280       dropout[0][0]
_________________________________________________________________________________________________
global_max_pooling1d (GlobalMax (None, 128)            0           conv1d[0][0]
_________________________________________________________________________________________________
dense_input (InputLayer)     [(None, 21)]              0
_________________________________________________________________________________________________
flatten (Flatten)            (None, 128)               0           global_max_pooling1d[0][0]
_________________________________________________________________________________________________
dense (Dense)                (None, 21)                462         dense_input[0][0]
_________________________________________________________________________________________________
dense_2 (Dense)              (None, 128)               16512       flatten[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, 7)                 154         dense[0][0]
_________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0           dense_2[0][0]
_________________________________________________________________________________________________
concatenate (Concatenate)    (None, 135)               0           dense_1[0][0]
                                                                   dropout_1[0][0]
_________________________________________________________________________________________________
dense_3 (Dense)              (None, 135)               18360       concatenate[0][0]
_________________________________________________________________________________________________
dense_4 (Dense)              (None, 7)                 952         dense_3[0][0]
=================================================================================================
Total params: 2,645,720
Trainable params: 2,645,720
Non-trainable params: 0
```

Slika 16: Prikaz modela

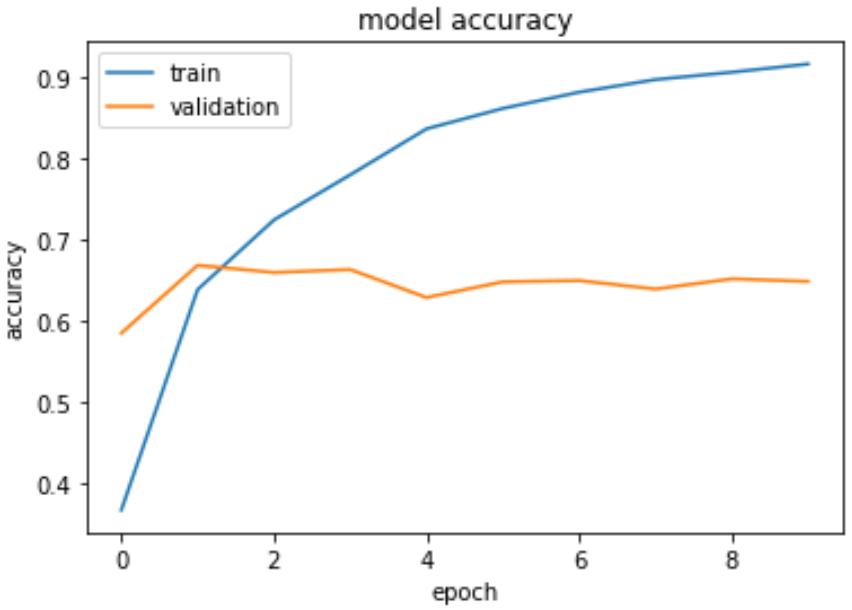
4.4. Evaluacija modela

Postoji više načina evaluiranja modela. Za evaluciju ovog modela biti će korištena točnost (engl. accuracy) kao jedna od najjednostavnijih i najčešće korištenih metrika. Ona predstavlja koliko je model puta pogodio kojoj klasi pripadaju testni podaci. Model je treniran u 10 epoha, a detalji su dostupni na slici 17.

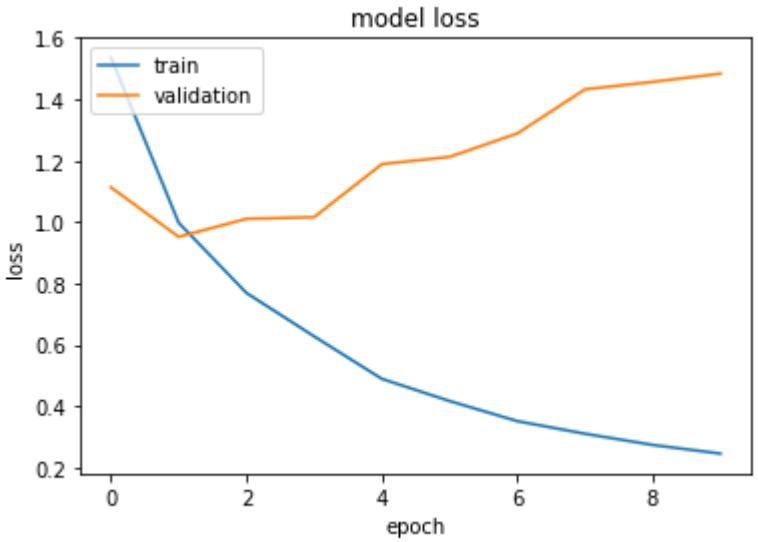
```
Epoch 1/10
719/719 [==============================] - 29s 40ms/step - loss: 1.5360 - accuracy: 0.3679 - val_loss: 1.1130 - val_accuracy: 0.5853
Epoch 2/10
719/719 [==============================] - 28s 39ms/step - loss: 0.9976 - accuracy: 0.6389 - val_loss: 0.9519 - val_accuracy: 0.6688
Epoch 3/10
719/719 [==============================] - 28s 39ms/step - loss: 0.7690 - accuracy: 0.7245 - val_loss: 1.0105 - val_accuracy: 0.6599
Epoch 4/10
719/719 [==============================] - 28s 39ms/step - loss: 0.6275 - accuracy: 0.7800 - val_loss: 1.0159 - val_accuracy: 0.6635
Epoch 5/10
719/719 [==============================] - 28s 39ms/step - loss: 0.4890 - accuracy: 0.8365 - val_loss: 1.1887 - val_accuracy: 0.6291
Epoch 6/10
719/719 [==============================] - 29s 40ms/step - loss: 0.4167 - accuracy: 0.8615 - val_loss: 1.2124 - val_accuracy: 0.6484
Epoch 7/10
719/719 [==============================] - 29s 40ms/step - loss: 0.3512 - accuracy: 0.8815 - val_loss: 1.2886 - val_accuracy: 0.6500
Epoch 8/10
719/719 [==============================] - 28s 40ms/step - loss: 0.3101 - accuracy: 0.8970 - val_loss: 1.4326 - val_accuracy: 0.6395
Epoch 9/10
719/719 [==============================] - 28s 39ms/step - loss: 0.2736 - accuracy: 0.9061 - val_loss: 1.4569 - val_accuracy: 0.6521
Epoch 10/10
719/719 [==============================] - 28s 39ms/step - loss: 0.2453 - accuracy: 0.9162 - val_loss: 1.4835 - val_accuracy: 0.6489
```

Slika 17: Pregled treniranja modela po epohama

Kao što je brzo moguće primjetiti, model u drugoj epohi dolazi do točnosti na podacima za testiranje od 66,88% no nakon toga raste samo točnost na podacima za treniranje. Ta se pojava naziva prenaučenost, a ono što možemo zaključiti iz toga jest da veći broj epoha neće pozitivno djelovati na točnost modela. Isto nam govori i vrijednost funkcije gubitka koja svoj minimum na testnim podacima dostiže u drugoj epohi te nakon toga nastavlja rasti, dok njena vrijednost na podacima za treniranje tijekom svih epoha nastavlja padati. Stoga možemo zaključiti da model pokazuje optimalne rezultate na ovom skupu podataka nakon 2 epohe. Grafički se prikaz navedenoga nalazi na slikama 18 i 19.



Slika 18: Grafički prikaz točnosti modela kroz epohe



Slika 19: Grafički prikaz vrijednosti funkcije gubitka kroz epohe

Za detaljan prikaz predviđanja modela se koristi matrica zbunjivanja (engl. confusion matrix). Ona za svaku klasu prikazuje koliko je model puta za tu klasu predvidio točno, a koliko puta bilo koju drugu klasu. Za ovaj model matrica zbunjivanja se nalazi na slici 20 gdje svaki redak predstavlja točnu klasu, a stupac predviđene klase. Iz matrice ovog modela može se isčitati da model najmanje poteškoća, odnosno najviše točnih predviđanja, ima kod klase koja predstavlja osjećaj radosti. Ta je pojava veoma razumljiva jer od 7 klasa u ovom skupu podataka, radost je jedina emocija koja se može klasificirati kao pozitivna. Drugi redak predstavlja osjećaj straha, treći ljutnju, četvrti tugu, peti gađenje, šesti sram te poslijednji krivnju. Jednak je poredak i na stupcima slijeva nadesno.

	Joy	Fear	Anger	Sadness	Disgust	Shame	Guilt
Joy	[[282	1	5	4	3	4	3]
Fear	[7	197	24	19	15	13	15]
Anger	[5	15	163	23	29	26	22]
Sadness	[9	16	19	190	20	11	11]
Disgust	[3	13	33	16	153	23	7]
Shame	[11	22	26	11	18	130	32]
Guilt	[6	16	37	26	13	41	129]]

Slika 20: Matrica zbunjivanja modela

5. Zaključak

Prilikom izrade ovog modela isprobane su razne razine kompleksnosti. Zaključak je da povećanje kompleksnosti modela, naročito konvolucijske grane, ne utječe dovoljno pozitivno na postotak točnosti da bi opravdalo povećanje vremena koje je potrebno za treniranje. Veća kompleksnost jedino dovodi do veće otpornosti modela na prenaučenost. Stoga se može pretpostaviti da kako bi povećali točnost modela, potreban je veći skup podatka. To je najviše moguće primjetiti na zadnje dvije klase, krivnja i sram, jer matrica zbunjivanja pokazuje da model najviše griješi kod razlikovanja tih dviju emocija.

Sama točnost, s maksimalnom vrijednošću od 66,88% je zadovoljavajuća uzevši u obzir da se radi o 7 klasa. Nju se može usporediti s radom Hamisha Dicksona (2018.) gdje je implementirao modele Yoon Kimovog (2014) istraživanja na sličnom skupu podataka. Dickson postiže točnost u rasponu od 45,39% do 50,74% prilikom višeklasnog klasificiranja s pet klasa, a ta točnost raste na 87,27% kod binarnog klasificiranja.

Model izrađen prilikom pisanja ovog rada može imati primjenu u omogućavanju komunikacije stroja i čovjeka. Potrebno je samo zamisliti da umjesto zapisanih vrijednosti kategoričkih atributa, stroj ima modul za vizualnu percepciju okoline te sam, koristeći na primjer algoritam slučajne šume opisan u radu Shotton et al. (2016) (razvijen za promatranje položaja tijela za Xbox), određuje vrijednosti tih atributa. Tekstualni atribut u tom slučaju može zamijeniti modul za sluh, kao što imaju virtualne pomoćne tehnologije poput Amazonove Alexe, Appleove Siri, itd. Za govor bi naravno bio potreban još jedan modul, no to nije sadržaj ovog rada.

Daljnjim razvojem ovakvih tehnologija omogućujemo dodatnu automatizaciju i pristupačnost usluga poput onih emocionalne potpore koje su danas još uvijek veoma skupe. Danas se već na tržištu nalazi nekoliko aplikacija koje omogućuju komuniciranje s agentom umjetne inteligencije za emocionalnu potporu poput Woebot-a. Iako on ima omogućenu samo tekstualnu komunikaciju, te je mogućnosti moguće proširiti.

Popis literature

Jason Brownlee (12.08.2019.): *A Tour of Machine Learning Algorithms*. Pristupano 30.08.2020. preko https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Jason Brownlee (15.08.2020.): *Crash Course On Multi-Layer Perceptron Neural Networks*. Pristupano 04.09.2020. preko https://machinelearningmastery.com/neural-networks-crash-course/

Margaret Rouse (2020.): *Unsupervised learning.* Pristupano 30.09.2020. preko https://searchenterpriseai.techtarget.com/definition/unsupervised-learning

Błażej Osiński, Konrad Budek (05.07.2018.): *What is reinforcement learning? The complete guide* pristupano (30.08.2020.) preko https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

Shervin Minaee, Nal Kalchbrenner, Eric Cambria, Narjes Nikzad, Meysam Chenaghlu, Jianfeng Gao (06.04.2020.): *Deep Learning Based Text Classification: A Comprehensive Review. Preuzeto 02.09.2020. sa* arxiv.org/abs/2004.03705, Cornell University

Joseph M. Hilbe (2009.): *Logistic Regression Models.* Boca Raton: Taylor & Francis Group preuzeto 01.09.2020. sa:
https://books.google.hr/books?id=tmHMBQAAQBAJ&lpg=PP1&ots=YgabAs6JcB&dq=logistic%20regression&lr&hl=hr&pg=PR5#v=onepage&q=logistic%20regression&f=false

Zhongheng Zhang (2016.): *Introduction to machine learning: k-nearest neighbors* Pristupano 01.09.2020. preko https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4916348/

Joachims T. (1998): *Text categorization with Support Vector Machines: Learning with many relevant features.* U: Nédellec C., Rouveirol C. (eds) Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1398. Springer, Berlin, Heidelberg. Preuzeto 01.09.2020. sa https://link.springer.com/content/pdf/10.1007%2FBFb0026683.pdf

Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze (2009.): *An Introduction to Information Retrieval.* Cambridge, England: Cambridge University Press.

Wenliang Du, Zhijun Zhan (2002): *Building Decision Tree Classifier on Private Data*. Preuzeto 02.09.2020. sa https://surface.syr.edu/cgi/viewcontent.cgi?article=1007&context=eecs

Masud Karim, Rashedur M. Rahman (2013.): *Decision Tree and Naïve Bayes Algorithm for Classification and Generation of Actionable Knowledge for Direct Marketing.* Journal of Software Engineering and Applications 6(4).

Fabrizio Sebastiani (2002*) Machine Learning in Automated Text Categorization, ACM Computing Surveys, Vol. 34, No. 1, March 2002, pp. 1–47*

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, Andrew Blake (2016.): *Real-Time Human Pose Recognition in Parts from Single Depth Images.* Microsoft Research Cambridge & Xbox Incubation. Preuzeto 03.09.2020. sa https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf

Leo Breiman (2001): *Random Forests*. Machine Learning, Volume 45, Issue 1, pp. 5–32.

Gary Marcus (2017): *Deep Learning: A Critical Appraisal.* Preuzeto 04.09.2020. sa https://arxiv.org/ftp/arxiv/papers/1801/1801.00631.pdf

Kai Sheng Tai, Richard Socher, Christopher D. Manning (2015): *Improved Semantic Representations FromTree-Structured Long Short-Term Memory Networks.* Preuzeto 05.09.2020. sa https://arxiv.org/pdf/1503.00075.pdf

Y. Le Cun, L. Bottou, Y. Bengio, P. Haffner (1998): *Gradient based learning applied to document recognition*. Proceedings of the IEEE, vol.86, no.11, pp.2278–2324.

Yoon Kim (2014): *Convolutional Neural Networks for Sentence Classification.* Preuzeto 06.09.2020. sa https://arxiv.org/pdf/1408.5882.pdf

Hamish Dickson (2018): *CNN for Sentence Classification by Yoon Kim.* Pristupljeno 10.09.2020. sa: https://www.kaggle.com/hamishdickson/cnn-for-sentence-classification-by-yoon-kim

Popis ilustracija

Slika 1: Model logističke regresije 5
Slika 2: Prikaz modela KNN 7
Slika 3: Prikaz modela SVM 8
Slika 4: Prikaz modela NB algoritma 10
Slika 5: Prikaz podjele skupa podataka kod stabla odluke 12
Slika 6: Prikaz stabla odluke 12
Slika 7: Prikaz slučajne šume (preuzeto iz rada Shotton et al. (2016)) 13
Slika 8: Prikaz neurona 15
Slika 9: Neuronska mreža 16
Slika 10: Arhitektura DAN-a (preuzeto iz rada Minaee et al. (2020)) 18
Slika 11: Lančana i stablasta arhitektura LSTM mreže (preuzeto iz rada Minaee et al. (2020)) 19
Slika 12: Podjela atributa 23
Slika 13: Predobrada podataka 24
Slika 14: Predobrada teksta 25
Slika 15: Grafički prikaz modela 26
Slika 16: Prikaz modela 27
Slika 17: Pregled treniranja modela po epohama 28
Slika 18: Grafički prikaz točnosti modela krooz epohe 28
Slika 19: Grafički prikaz vrijednosti funkcije gubitka kroz epohe 29
Slika 20: Matrica zbunjivanja modela 29

Sve su slike, osim ako nije drugačije navedeno, izrađene samostalno alatom draw.io, a slike koda su snimke zaslona.