Klasifikacija teksta primjenom algoritma k najbližih susjeda

Benić, Martina

Sveučilište u Rijeci, Fakultet za matematiku
Diskretna matematika i primjene

Martina Benić

Klasifikacija teksta primjenom algoritma k najbližih susjeda

Diplomski rad

Rijeka, rujan 2024.

Sveučilište u Rijeci, Fakultet za matematiku
Diskretna matematika i primjene

Martina Benić

Klasifikacija teksta primjenom algoritma k najbližih susjeda

Mentor: doc. dr. sc. Sanda Bujačić Babić

Diplomski rad

Rijeka, rujan 2024.

Sadržaj

Sažetak

Tema diplomskog rada je klasifikacija teksta primjenom algoritma k najbližih susjeda, odnosno primjenom kNN algoritma. U radu će se navedeni algoritam opisati kroz prizmu strojnog učenja, uz prethodno uvođenje vrlo detaljne matematičke podloge koja je temelj ovog, ali i mnogih drugih alata strojnog učenja. Jedan od glavnih uvjeta efikasnosti kNN algoritma je odabir parametra k pa će se kroz rad opisati razne metode i pristupi koji argumentiraju njegov izbor. Osim osnovnog kNN algoritma, definirat će se i dvije njegove inačice, algoritam k težinski najbližih susjeda i algoritam k najbližih susjeda s lokalnom srednjom vrijednošću. U okviru primjene kNN algoritma opisat će se problem klasifikacije teksta te procesi koje je potrebno provesti prije klasifikacije. Neki od njih su obrada teksta, indeksiranje riječi i izraza te proces smanjenja dimenzije koji obuhvaća vrijednost informacijskog dobitka i proces dekompozicije singularnih vrijednosti. Nakon navedenog, opisuje se upotreba poboljšanog kNN algoritma specificiranog upravo za svrhu klasifikacije teksta. Posljednji dio rada uključuje programsku realizaciju klasifikacije teksta korištenjem oba uvedena algoritma.

Ključne riječi: algoritam k najbližih susjeda, klasifikacijski problem, klasifikacijska stopa točnosti, algoritam k težinski najbližih susjeda, algoritam k najbližih susjeda s lokalnom srednjom vrijednošću, Booleovo ponderiranje, ponderiranje učestalosti riječi, tf-idf ponderiranje.

1 Uvod

Ljudska rasa smatra se najinteligentnijom vrstom na Zemlji. Tijekom svake faze vrlo turbulentne evolucijske povijesti ljudske vrste iznimno se ističe ljudska želja za neprestanim usavršavanjem, otkrivanjem određenih zakonitosti te instrumenata kojima je zadaća olakšati funkcioniranje ljudi. Upravo po tome se ljudska vrsta razlikuje od svih postojećih živućih vrsta na Zemlji. Kroz povijest su poznate četiri industrijske revolucije, a posljednju, četvrtu, svi imamo prilike osjetiti. Ona nam donosi uzlet umjetne inteligencije, robotike, napredne autonomne autoindustrije, 3D printanje, nanotehnologiju te kvantna računala. Jedna od najznačajnijih karakteristika četvrte industrijske revolucije jest gomilanje ogromne količine podataka koji se skupljaju i nastoje koristiti u daljnje svrhe, posebno kao trening podaci za poboljšavanje performansi algoritama strojnog učenja. Upravo zato što postoje ogromne količine podataka koje mogu biti korisne u daljnjem razvoju algoritama, potrebno je razviti "inteligentne" računalne sustave koji će moći imitirati ljudske postupke te na taj način olakšati funkcioniranje ljudima i usmjeriti ih na kompliciranije zadaće koje računala nisu (još) u mogućnosti odraditi na zadovoljavajući način. Razvojem navedenih računalnih sustava bavi se umjetna inteligencija (eng. artificial intelligence). Jedno od područja umjetne inteligencije je strojno učenje čiji je cilj osmisliti algoritme koji na temelju dobivenih podataka predviđaju vrijednosti novih podataka. Jedan od algoritama strojnog učenja je algoritam k najbližih susjeda, odnosno kNN algoritam koji je zbog svoje efikasnosti i jednostavnosti stekao veliku popularnost. Naime, kNN algoritam koristi se u financijama, medicini, poljoprivredi i mnogim drugim područjima, a vrlo je popularan prilikom analize teksta u rješavanju problema klasifikacije teksta. U okviru rada promatrat ćemo razne metrike koje se mogu koristiti te ćemo diskutirati izbor vrijednosti parametra k o kojem uvelike ovisi efikasnost kNN algoritma. Dodatno, u radu opisujemo osnovni kNN algoritam te njegovu inačicu koja je specifično namijenjena klasifikaciji teksta.

2 Osnovni pojmovi u strojnom učenju

Strojno učenje jedno je od najbrže rastućih područja umjetne inteligencije. Zadaća mu je osmisliti algoritme koji će na temelju dobivenih podataka, prirodnih uzoraka ili veza među podacima, moći odlučiti ili predvidjeti vrijednosti novih podataka. U strojnom učenju se u svrhu lakšeg razumijevanja klasičnih procesa definira jedan od osnovnih pojmova, učenje, odnosno

procedura koja asocira na učenje živih bića iz postojećih podataka i iskustva. Strojno učenje nalazi mnogobrojne primjene u praksi pa tako njegove rezultate nerijetko možemo naći u različitim znanostima, od informacijsko-komunikacijskih tehnologija do filozofije i teorije znanosti. Kompliciranih problemi koje ne možemo riješiti niti opisati na jednostavan način, velike količine podataka (rudarenje) te potreba za modifikacijom sustava kroz vrijeme, samo su neki od razloga velikog uzleta u istraživanju i razvoju strojnog učenja.

U strojnom učenju uvode se i definiraju neki standardni pojmovi koji sudjeluju u svakom klasičnom problemu strojnog učenja. Za početak potrebno je izabrati ili izgraditi skup podataka na kojem ćemo provoditi određeni algoritam strojnog učenja. Svi podaci koje koristimo moraju biti standardizirani i obrađeni prije implementacije algoritma strojnog učenja. U brojnim izvorima o strojnom učenju nastoji se formalno definirati osnovne pojmove strojnog učenja, odnosno odgovoriti na pitanje što je učenje u kontekstu strojnog učenja, što je trening, a što testni skup i sl. U [2] se, na primjer, učenje definira na ovakav način: "Računalni program uči iz iskustva E, s obzirom na klasu zadataka T i mjeru uspješnosti P, ako se njegova izvedba na zadacima iz T, mjerena pomoću P, poboljšava s iskustvom E.". Kako bi se izgradio model strojnog učenja, cjelokupni skup podataka potrebno je podijeliti na dva dijela. Jedan dio najčešće sadrži 80% podataka te se na njemu gradi model strojnog učenja, a zove se trening skup. Preostali dio podataka kojeg nismo koristili za treniranje, već ga koristimo u svrhu testiranja, odnosno ispitivanja efikasnosti kreiranog model, zove se testni skup.

Definicija 2.1 *Par $(x^{(i)}, y^{(i)})$ naziva se **trening primjer**, a skup podataka $\{(x^{(i)}, y^{(i)}) \mid i = 1, ..., n\}$, koji koristimo za treniranje/učenje, naziva se **trening skup**.*

Definicija 2.2 ***Značajka** ili **atribut** je bitna karakteristika/svojstvo primjera.*

Definicija 2.3 ***Vektor obilježja** nekog primjera je vektor $x = (x_1, x_2, ..., x_n)$ gdje su $x_i, i = 1, ..., n$ pojedine značajke tog primjera te je n ukupan broj značajki.*

Definicija 2.4 *Skup podataka $\{(x^{(i)}, y^{(i)}) \mid i = 1, ..., m\}$, koji koristimo za procjenu točnosti modela, naziva se **testni skup**.*

Primjer 2.5 *Neka je zadan primjer procjene cijene nekretnine na temelju njezinih značajki. Trening primjer je par $(x^{(i)}, y^{(i)})$, gdje je $x^{(i)}$ vektor obilježja nekretnine i, a $y^{(i)}$ cijena nekretnine i. Neke od karakteristika pojedine*

nekretnine su površina, godina izgradnje i boja. Nisu sve navedene karakteristike jednako bitne prilikom procjene cijene: dok površina i godina izgradnje uvelike utječu na cijenu nekretnine, njezina boja nije toliko bitna stavka. Stoga, površina i godina izgradnje čine bitne karakteristike te su one značajke, dok boja nekretnine nije značajka u ovom primjeru. Kako su površina i godina izgradnje značajke, vektor obilježja $x^{(i)}$ sastoji se od tih dviju značajki, odnosno $x^{(i)} = (x_1, x_2)$, pri čemu x_1 označava površinu, a x_2 godinu izgradnje. Neka je sveukupno zadano N nekretnina. Skup $\{(x^{(i)}, y^{(i)}) \mid i = 1, ..., n\}$, pri čemu je $n < N$, čini trening skup te se on koristi za treniranje, dok skup $\{(x^{(i)}, y^{(i)}) \mid i = n+1, ..., N\}$, koji je disjunktan s trening skupom, čini testni skup i on se koristi za procjenu točnosti modela.

Strojno učenje, obzirom na način učenja, možemo podijeliti u tri kategorije:

- nadzirano učenje (eng. supervised learning),
- nenadzirano učenje (eng. unsupervised learning),
- podržano učenje (eng. reinforcement learning).

Nadzirano učenje je vrsta strojnog učenja u kojem algoritam dobiva ulazne podatke koji su organizirani tako da je definirano što su značajke (nezavisne varijable) ili svojstva te što su izlazne vrijednosti podataka (zavisna varijabla). Cilj algoritama nadziranog učenja je odrediti funkciju koja za nove nezavisne varijable, određuje, odnosno predviđa izlazne vrijednosti, to jest vrijednost zavisne varijable. Dva problema koja se rješavaju primjenom nadziranog učenja su: klasifikacija i regresija. Ukoliko očekujemo da varijabla čiju vrijednost želimo procijeniti bude realan broj i ujedno element nekog intervala ili segmenta realnih brojeva, radi se o regresijskom problemu. U suprotnom, ako izlazna varijabla može poprimiti samo konačan broj unaprijed očekivanih realnih vrijednosti, radi se o klasifikacijskom problemu [16]. Primjerice, ako na temelju svojstava nekretnine, kao što su njezina površina i godina izgradnje, procjenjujemo njezinu vrijednost, riječ je o regresijskom problemu jer procjena cijene može poprimiti realan broj unutar nekog intervala ili segmenta skupa $\mathbb{R}_+$. Primjer klasifikacijskog problema je problem prepoznavanja lica, pri čemu je ulazni podatak slika lica te je potrebno odrediti kojoj od unaprijed definiranih klasa lica ljudi iz baze slika ta slika pripada. U nenadziranom učenju na raspolaganju imamo podatke iz kojih treba izvesti određene zaključke, pri čemu se dani podaci ne dijele na zavisne i nezavisne varijable. Potrebno je odrediti pravilnost u uzorku, odnosno odrediti koji se uzorci javljaju češće od nekih drugih, koji se slučajevi nikad ne javljaju

i slično. Nenadzirano učenje možemo podijeliti u dvije kategorije: grupiranje (eng. clustering) i smanjivanje dimenzije (eng. dimensionality reduction). Primjer nenadziranog učenja je problem grupiranja rukom pisanih znamenki. Dani skup znamenki potrebno je podijeliti u grupe na način da se postigne što veća sličnost između znamenki unutar iste grupe te što manja sličnost između znamenki koji pripadaju različitim grupama [6].
U podržanom učenju program "uči" kako se "ponašati" u okolini kroz iskustvo. Ova vrsta učenja ima tri komponente: agenta, okolinu i akcije. Program dobiva povratnu informaciju o svojim akcijama u obliku nagrade te na taj način određuje koje su akcije najbolje. Izlazni proces je niz akcija pri čemu jedna akcija sama po sebi nije ključna, već niz akcija u pravilnom redoslijedu. Primjer ove vrste učenja je robotski agent čija je okolina labirint u kojem se nalazi. Akcije koje agent ima na raspolaganju su kretanje gore, dolje, lijevo i desno te je cilj agenta pronaći izlaz iz labirinta. Također, u problem podržanog učenja spada i primjer "učenja" računala da igra određenu igru. Kao i u prethodno navedenom primjeru, pojedinačna akcija neće biti ključna, već je ključan niz akcija koji dovodi do pobjede [17].

3 Problem klasifikacije

Problem klasifikacije jedan je od problema nadziranog učenja u okviru kojeg je potrebno odrediti funkciju koja ulaznim vrijednostima dodjeljuje izlazne vrijednosti, pri čemu je skup izlaznih vrijednosti diskretan skup. Traženu funkciju nazivamo klasifikator. Označimo s $\mathcal{X}$ skup ulaznih vrijednosti i s $\mathcal{Y}$ skup izlaznih vrijednosti. Za dani trening skup potrebno je odrediti klasifikator, odnosno funkciju $h : \mathcal{X} \to \mathcal{Y}$, pri čemu želimo da funkcija h što bolje procjenjuje podatke testnog skupa. Parametar kojim mjerimo efikasnost učinka klasifikatora je klasifikacijska stopa točnosti (eng. classification accuracy rate).

Definicija 3.1 ***Klasifikacijska stopa točnosti*** *je postotak točno klasificiranih podataka iz testnog skupa.*

Neki od algoritama nadziranog strojnog učenja koji se bave klasifikacijom su sljedeći: linearni klasifikator, logistička regresija, naivni Bayesov klasifikator, metoda potpornih vektora, neuronske mreže, stabla odlučivanja, slučajne šume, algoritam k najbližih susjeda.

Primjer 3.2 *Tvrtke prilikom objavljivanja natječaja za posao dobivaju veliki broj prijava. Između ukupnog broja prijava, samo neki od kandidata ulaze u*

uži izbor za zaposlenje i budu pozvani na razgovor za posao. Na temelju podataka iz prijava koji su relevantni za obavljanje posla, procjenjuje se hoće li navedeni kandidat biti pozvan na dodatni razgovor. Neki od tih podataka su stupanj obrazovanja, prethodno radno iskustvo na sličnim poslovima, učestalost mijenjanja poslova i slično. Naime, ukoliko tvrtka ima iskustvo da kandidati koji učestalo mijenjaju poslove nisu dugo ostali na nekoj od pozicija u njihovoj tvrtki, u budućnosti žele izbjeći takva iskustva i ne zapošljavati takve kandidate. Na taj način, model na temelju prošlih iskustava, odlučuje pozivaju li se takvi kandidati na daljnji razgovor ili ne.

Ovo je primjer klasifikacijskog problema s dvije klase: klijenti koji su pozvani na razgovor i klijenti koji nisu pozvani na razgovor. Pomoću podataka o klijentu definira se klasifikator, pri čemu su podaci o kandidatu ulazne vrijednosti, dok izlaznu vrijednost čini jedna od dviju klasa.

Nakon treninga na podacima prošlih kandidata, može se definirati klasifikacijsko pravilo. Označimo s a "godine radnog iskustva na sličnim poslovima" i s b "broj promijenjenih poslova unazad godine dana".

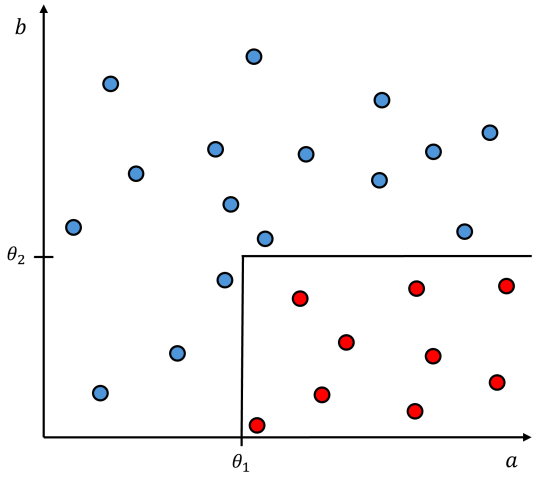
Klasifikacijsko pravilo možemo definirati na sljedeći način:

$$\begin{aligned} &IF\ a > \theta_1\ \ AND\ b < \theta_2\ \ THEN \\ &\qquad pozvati\ na\ razgovor \\ &ELSE \\ &\qquad ne\ pozvati\ na\ razgovor, \end{aligned}$$

za odgovarajuće vrijednosti parametara θ_1 i θ_2. U navedenom klasifikacijskom pravilu, kao podatke o kandidatu promatramo samo godine njegovog prethodnog radnog iskustva i učestalost mijenjanja poslova, odnosno broj promijenjenih poslova u prethodnoj godini. Pomoću klasifikacijskog pravila, iz podataka novih klijenata, određujemo pripadaju li oni klasi kandidata koje treba pozvati na razgovor ili klasi kandidata koje ne treba pozvati na razgovor.

Na Slici 1 grafički je prikazano prethodno navedeno klasifikacijsko pravilo. Crvenom bojom označeni su kandidati koji su pozvani na razgovor, dok su plavom bojom označeni kandidati koji nisu pozvani na razgovor.

Na ovaj način se izrazito olakšalo zaposlenicima u sektoru ljudskih resursa određene tvrtke jer je algoritam odradio posao klasifikacije koji bi inače trebali odraditi zaposlenici tvrtke. Naglasimo da se naknadno algoritam uvijek može revidirati tako da se analiziraju rubni primjeri pojedinih klasa, no to je u svakom slučaju puno manji opseg posla od onog koji nije uključivao klasifikacijski algoritam i koji se odnosi na detaljno pregledavanje svake prijave nekog zaposlenika iz sektora ljudskih resursa.



Slika 1: Klasifikacija prijavljenih kandidata

.

4 Algoritam k najbližih susjeda (kNN algoritam)

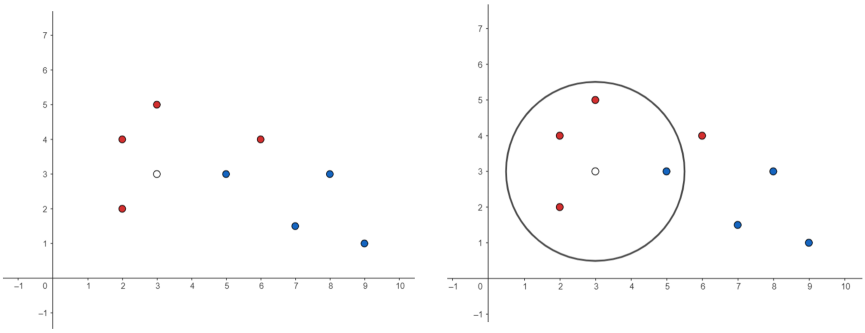
Algoritam k najbližih susjeda (eng. k nearest neighbor algorithm), odnosno kNN algoritam, jedan je od popularnijih algoritama strojnog učenja. Razlog djelomično leži u tome što je algoritam vrlo jednostavno objasniti pa je on u konačnici vrlo predočljiv i razumljiv. U usporedbi s ostalim kompliciranijim algoritmima njegova točnost je u većini slučajeva približno jednaka ili veća.

4.1 Opis kNN algoritma

kNN algoritam pronalazi grupu od k elemenata koji se u trening skupu nalaze najbliže elementu čiju je izlaznu vrijednost potrebno odrediti. Prilikom određivanja udaljenosti između elemenata koriste se razne metrike od kojih su najčešće Euklidska i Manhattan metrika. kNN algoritam elementu pridružuje vrijednost najmnogobrojnije klase unutar grupe od k najbližih elementa. Efikasnost kNN algoritma uvelike ovisi o odabiru parametra k

te postoje razne metode odabira optimalnog parametra k. Također, postoje razne varijante kNN algoritma kao što su algoritam k težinski najbližih susjeda i algoritam k najbližih susjeda s lokalnom srednjom vrijednošću.

Primjer 4.1 *U koordinatnom sustavu nalaze se podaci koji su prikazani u obliku kružića koji su klasificirani u dvije klase: crveni i plavi kružići. Zadatak je odrediti kojoj od navedenih klasa pripada novi podatak koji je označen kao bijeli kružić. Ako pretpostavimo da je $k = 4$, tražimo četiri kružića koja se nalaze najbliže bijelom kružiću te promatramo kako su oni klasificirani.*



Slika 2: Određivanje klase bijelog kružića kNN algoritmom

Na desnoj slici su označeni oni primjeri koji se nalaze najbliže novom podatku. Uočimo da jedan podatak pripada klasi plavih kružića dok preostala tri podatka pripadaju klasi crvenih kružića. kNN algoritam će u ovom slučaju odrediti da je novi podatak dio klase označene crvenim kružićima.

kNN algoritam općenito karakterizira:

i) skup označenih objekata koji se koriste za procjenu klase testnog objekta,

ii) metrika, odnosno funkcija kojom se određuje udaljenost između objekata,

iii) parametar k koji predstavlja broj najbližih susjeda,

iv) metoda koja se koristi za određivanje klase zadanog objekta na temelju klasa i udaljenosti k najbližih susjeda.

Najjednostavniji oblik kNN algoritma uključuje dodjeljivanje klase objektu njegovog najbližeg susjeda ili klasu većine njegovih susjeda, što je prikazano u Primjeru 4.1.

Algoritam 1 Osnovni kNN algoritam [19]

Input: trening skup D,
testni objekt z,
skup klasa L

Output: klasa objekta z, $c_z \in L$

foreach $\mathbf{y} \in D$ **do**
odredi udaljenost između $\mathbf{z}$ i $\mathbf{y}$, $d(\mathbf{z}, \mathbf{y})$;
end
Odaberi skup $N \subseteq D$ od k najbližih susjeda objekta z;
$c_z = \arg\max_{v \in L} \sum_{y \in N} I(v = c_y)$,
gdje je I indikatorska funkcija koja vraća 1 ukoliko je njezin argument true, odnosno ako je klasa od y jednaka klasi v, i 0 inače.

Za dani trening skup D, skup klasa L i testni objekt z, koji je vektor obilježja nepoznate klase, algoritam računa udaljenost između objekta z i svih objekata iz D te određuje listu od k najbližih susjeda. Zatim objektu z dodjeljuje najmnogobrojniju klasu iz liste od k najbližih susjeda.
Objasnimo ovaj algoritam na Primjeru 4.1. Trening skup D predstavljaju podaci u obliku crvenih i plavih kružića. Testni objekt z je bijeli kružić, dok skup L sadrži dvije klase: crveni i plavi kružići. Potrebno je odrediti kojoj od dviju navedenih klasa pripada objekt z. Za svaki objekt iz trening skupa, $y \in D$, određujemo njegovu udaljenost od objekta z. Zatim odredimo podskup N skupa D koji sadrži četiri objekta koja se nalaze najbliže objektu z. Za svaku klasu skupa L, gledamo koliko objekata skupa N pripada toj klasi. Klasi crvenih kružića pripadaju tri od četiri elementa skupa N, dok plavoj klasi pripada samo jedan element skupa N. Više objekata pripada crvenoj klasi pa kNN algoritam objektu z dodjeljuje crvenu klasu.

4.2 Metrike kNN algoritma

Neke od metrika koje se mogu koristiti za mjerenje udaljenosti između objekata su Euklidska i Manhattan metrika.

Definicija 4.2 ***Euklidska metrika*** *je funkcija* $d : \mathbb{R}^n \to \mathbb{R}$ *definirana na sljedeći način:*

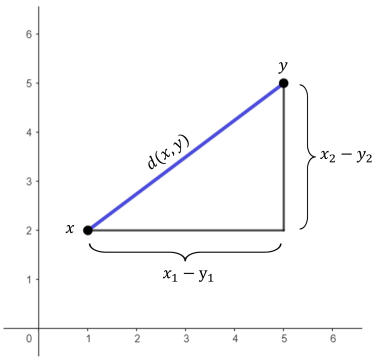
$$d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}, \quad x, y \in \mathbb{R}^n.$$

Definicija 4.3 ***Manhattan metrika*** *je funkcija* $d : \mathbb{R}^n \to \mathbb{R}$ *definirana na sljedeći način:*

$$d(x, y) = \sum_{k=1}^{n} |x_k - y_k|, \quad x, y \in \mathbb{R}^n.$$

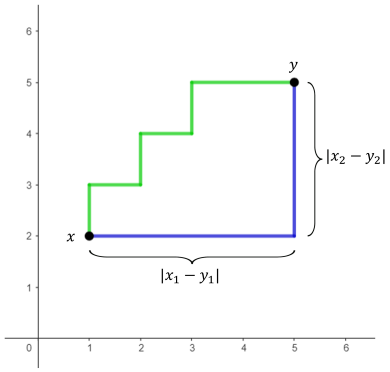
Primjer 4.4 *Neka su* $x = (1, 2)$ *i* $y = (5, 5)$*. Odredimo Euklidsku i Manhattan metriku između* x *i* y*.*

Euklidska metrika: $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
$= \sqrt{(1 - 5)^2 + (2 - 5)^2}$
$= \sqrt{25}$
$= 5.$



Slika 3: Euklidska metrika

Manhattan metrika: $d(x, y) = |x_1 - y_1| + |x_2 - y_2|$
$= |1 - 5| + |2 - 5|$
$= 4 + 3$
$= 7.$

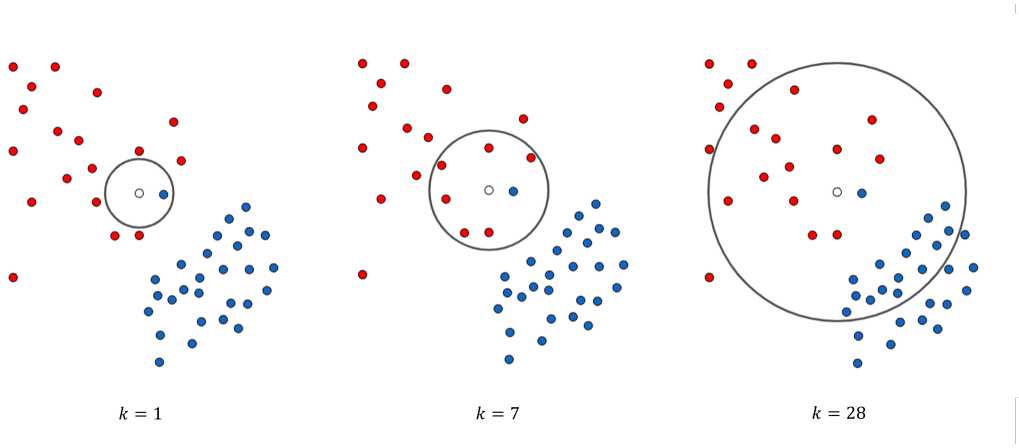


Slika 4: Manhattan metrika

Manhattan metrika prikazana je i plavom i zelenom bojom što nisu jedine mogućnosti. Razlog tome je taj što prvi pribrojnik označuje udaljenost koju trebamo prijeći po prvoj koordinati, a drugi pribrojnik udaljenost koju je potrebno prijeći po drugoj koordinati. Pri tome nije bitno kojim redoslijedom prelazimo te udaljenosti pa zbog toga imamo više načina na koje možemo prikazati Manhattan udaljenost.

4.3 Odabir parametra k

Jedan od glavnih problema kNN algoritma je odrediti vrijednost parametra k tako da algoritam što točnije klasificira podatke. Na Slici 5 možemo vidjeti kako klasifikacija novog podatka uvelike ovisi o odabiru vrijednosti parametra k. Kao i u Primjeru 4.1, zadane su dvije klase označene kružićima crvene i plave boje te je potrebno utvrditi kojoj klasi pripada novi podatak reprezentiran bijelim kružićem. Promatrajući slučaj kada je $k = 1$, tražimo podatak iz trening skupa koji je najbliži novom podatku. Uočimo kako je plavi kružić podatak iz trening skupa koji je najbliže novom podatku te zaključujemo kako novi podatak pripada plavoj klasi. Ukoliko odaberemo slučaj $k = 7$, u sedam podataka trening skupa koji se nalaze najbliže novom

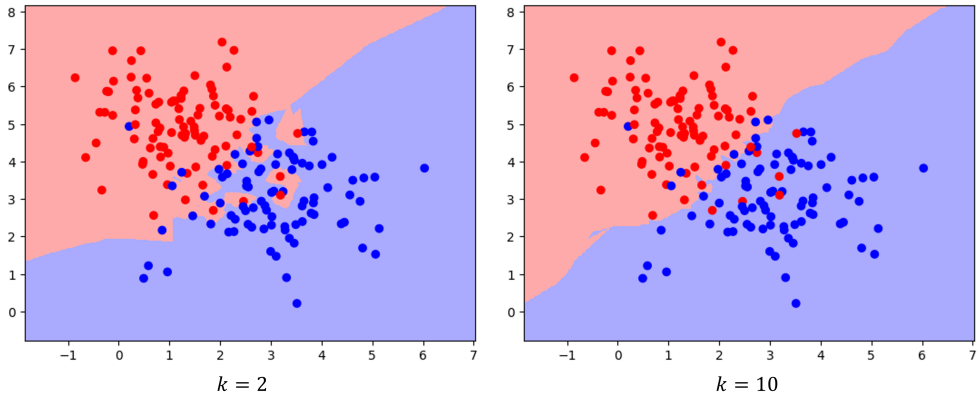


Slika 5: Različite vrijednosti parametra k

podatku, nalazi se šest crvenih i jedan plavi kružić. Kako je crvena klasa mnogobrojnija, zaključujemo da je novi podatak crvene klase. Promotrimo sada slučaj kada je $k = 28$. U trening skupu, u 28 najbližih susjeda novog podatka nalazi se 12 crvenih i 16 plavih kružića. Na temelju toga zaključujemo

da novi podatak pripada plavoj klasi. Na temelju ovog primjera, vidimo kako je u različitim slučajevima odabira parametra k na istom primjeru, testnom primjeru dodijeljena i različita klasa. Prilikom odabira parametra k, bitno je da ne odaberemo ni premalo ni preveliko susjedstvo testnog objekta. Iz Slike 5 očigledno zaključujemo da je drugi slučaj, $k = 7$, najbolji odabir parametra k te da je zbog rasporeda crvenih i plavih kružića, najveća vjerojatnost da bijeli kružić pripada crvenoj klasi.

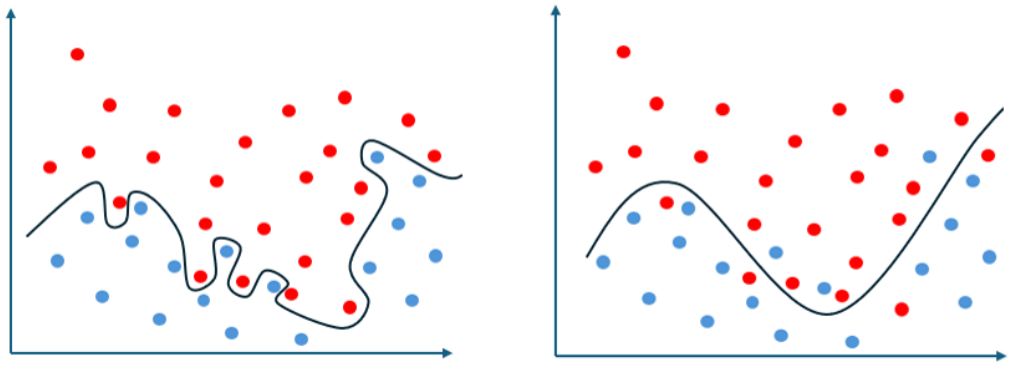
Za zadani parametar k moguće je napraviti granice između klasa. Granica postaje glađa s povećanjem vrijednosti k. Na Slici 6 možemo uočiti kako je granica između dviju klasa glađa u slučaju kada je $k = 10$.



Slika 6: Granica između klasa

Prilikom odabira vrijednosti parametra k može doći do prenaučenosti (eng. overfitting). Za model kažemo da je prenaučen ukoliko daje točna predviđanja za podatke iz trening skupa, ali ne predviđa točno nove podatke. Neki od razloga zbog kojih dolazi do prenaučenosti su premalen trening skup koji ne sadrži dovoljan broj reprezentativnih podataka cijelog skupa te podaci koji sadrže puno nepotrebnih i nevažnih informacija koje nisu potrebne za klasifikaciju. Na Slici 7 lijevo možemo vidjeti kako izgleda prenaučen model, dok na slici desno možemo vidjeti kako izgleda jedna varijanta optimalno izabrane vrijednosti parametra k koja ne rezultira prenaučenošću. Iako su na lijevoj slici svi podaci točno klasificirani, to ne uvjetuje točno predviđanje novih podataka. Veća je vjerojatnost da će nove podatke bolje klasificirati desni model te je on poželjniji. Naime, ukoliko neki podatak odstupa od ostalih podataka iz svoje klase, ponekad je bolje ne gledati njegova svojstva kao glavna obilježja klase. Ako neki crveni kružić odskače od ostalih crvenih kružića te ima svojstva sličnija plavim kružićima, ukoliko se njegova svojstva

promatraju kao obilježja crvene klase, teže će se odrediti razlika između tih dviju klasa te će se prilikom klasifikacije novih podataka teže klasificirati podatak s glavnim obilježjima neke klase.



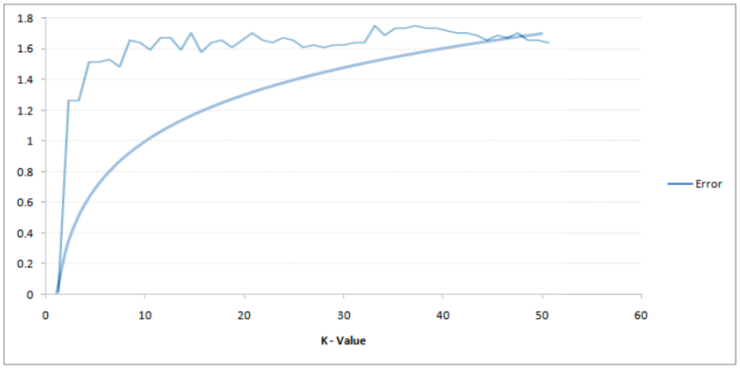
Slika 7: Prenaučen i regularan model

4.4 Greške modela

Stopa greške je postotak netočno klasificiranih podataka. Ukoliko promatramo *stopu greške trening skupa* (eng. training error rate), riječ je o postotku netočno klasificiranih podataka iz trening skupa, dok je *validacijska stopa greške* (eng. validation error rate) postotak netočno klasificiranih podataka testnog skupa. Promatrajući stopu greške za trening skup u ovisnosti o parametru k, promatramo odstupanje između točne i predviđene vrijednosti podataka iz trening skupa u ovisnosti o odabiru parametra k.
Neka je $\mathcal{X}$ skup ulaznih vrijednosti, L skup izlaznih vrijednosti, odnosno klasa te neka je $D = \{(x_i, y_i) \mid i = 1, ..., n\}$ trening skup i $h : \mathcal{X} \to L$ kNN klasifikator. Grešku trening skupa, ϵ_{tr} [19], definiramo na sljedeći način:

$\epsilon_{tr} = 0,$
$\forall (x_i, y_i) \in D$ takav da vrijedi $h(x_i) \neq y_i$, $\epsilon_{tr} = \epsilon_{tr} + 1.$

Greška ϵ_{tr} će za trening skup uvijek biti 0 ukoliko je $k = 1$. Razlog tome je taj što je svakom objektu iz trening skupa najbliži on sam. Na Slici 8 prikazane su greške trening skupa dva različita modela [22] te možemo uočiti kako se stopa greške povećava s povećavanjem parametra k.

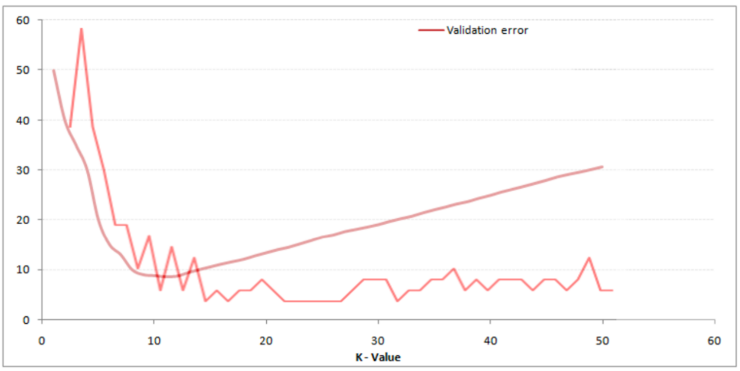


Slika 8: Greška trening skupa

Promotrimo sada stopu greške testnog skupa, odnosno validacijsku stopu greške. Neka je $\mathcal{X}$ skup ulaznih vrijednosti, L skup izlaznih vrijednosti, odnosno klasa te neka je $D = \{(x_i, y_i) \mid i = 1, ..., n\}$ trening skup, $T = \{(x_i, y_i) \mid i = 1, ..., m\}$ testni skup i $h : \mathcal{X} \to L$ kNN klasifikator.
Grešku testnog skupa, ϵ_{te} [19], definiramo na sljedeći način:

$$\epsilon_{te} = 0,$$
$$\forall (x_i, y_i) \in T \text{ takav da vrijedi } h(x_i) \neq y_i,\ \epsilon_{te} = \epsilon_{te} + 1.$$

Na Slici 9 prikazane su validacijske greške dva različita modela [24]. Možemo uočiti kako validacijska greška ϵ_{te} u početku pada te nakon određene vrijednosti parametra k počinje rasti.



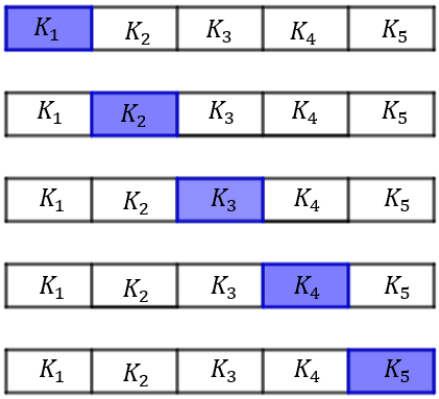
Slika 9: Greška testnog skupa

Prilikom izgradnje modela, želimo da taj model što bolje procjenjuje nove podatke. Odnosno, nakon obuke modela na trening skupu, cilj je da validacijska stopa greška bude što manja.

4.5 Metode odabira parametra k

Budući da parametar k ima važnu ulogu u efikasnosti kNN algoritma, postavlja se pitanje na koji se način odabire njegova optimalna vrijednost. Jedan od načina odabira vrijednosti parametra k je metoda pokušaja i pogreški dok se ne pronađe vrijednost za koju algoritam postiže najbolji rezultat. Sljedeći način je da se za odabir parametra k uzima vrijednost $\sqrt{n}$ zaokružena na najbliži cijeli broj, pri čemu je n broj podataka u trening skupu. Slično, za vrijednost parametra k može se uzeti vrijednost $n^{(2/8)}$ ili $n^{(3/8)}$ zaokružena na cijeli broj, pri čemu je n ponovno broj podataka u trening skupu [11].
U ovisnosti o vrijednosti parametra k, moguć je slučaj da se unutar k najbližih elemenata nalazi jednak broj elemenata svake klase. Naime, ukoliko se podaci klasificiraju u dvije klase te je vrijednost parametra k paran broj, moguće je da se među k najbližih elemenata nalazi jednak broj elemenata jedne i druge klase. Pogledajmo Primjer 4.1 gdje je $k = 4$. U slučaju da su podaci drukčije raspoređeni, unutar četiri najbliža podatka mogla su se nalaziti dva crvena i dva plava kružića i tada bi obje klase imale jednak broj elemenata te kNN algoritam ne može po principu mnogobrojnosti dodijeliti klasu. Ukoliko je u istom primjeru vrijednost parametra k neparan broj, unutar k najbližih elemenata sigurno će jedna od klasa biti mnogobrojnija te će kNN algoritam dodijeliti tu klasu. Ako u primjeru imamo n klasa, tada se problem s jednakim brojem elemenata unutar $m \leq n$ klasa može pojaviti ukoliko je k višekratnik od m. U slučaju da dvije ili više klasa imaju jednak broj pojavljivanja, neki od načina dodjeljivanja klase su da se objektu dodjeljuje jedna od tih klasa na nasumičan način ili klasa koja se najčešće javlja u trening skupu.
Optimalnu vrijednost parametra k moguće je odrediti promatranjem grešaka modela na način da minimiziramo grešku, odnosno tražimo k za koji je greška najmanja. Nećemo promatrati grešku trening skupa jer je ona uvijek jednaka nuli za $k = 1$. Iz Slike 5 možemo vidjeti da ne vrijedi da je optimalna vrijednost uvijek $k = 1$. Stoga promatramo validacijsku grešku. Problem određivanja optimalne vrijednosti parametra k je problem optimizacije te je potrebno odrediti minimalnu vrijednost krivulje greške testnog skupa. Navedeni postupak naziva se *metoda lakta* jer krivulja greške testnog skupa podsjeća na lakat.
Jedan od učinkovitih načina za pronalaženje optimalne vrijednosti parametra k je metoda unakrsne provjere (eng. cross validation). U ovoj metodi podaci se dijele na trening skup i testni skup više puta. Dva najčešće korištena oblika ove metode su metoda K-kratne unakrsne provjere (eng. K-fold cross-validation) te metoda izostavljanja jednog uzorka kod unakrsne provjere (eng. leave-one-out cross-validation).

U *metodi K-kratne unakrsne provjere* (K-FCV) potrebno je podijeliti podatke u K međusobno disjunktnih skupova koji čine particiju cjelokupnog skupa podataka, pri čemu je najjednostavniji slučaj kada su svi skupovi jednake veličine. Zatim se $K-1$ skup koristi za treniranje skupa podataka te se na temelju toga procjenjuju podaci iz preostalog test skupa. Navedeni postupak se zatim ponavlja za svih K mogućih izbora za skupove koji predstavljaju test skup. [4]
Promotrimo opisanu metodu na jednom primjeru. Neka je $K = 5$. Potrebno je podatke podijeliti u pet međusobno disjunktnih skupova K_1, K_2, K_3, K_4, K_5 koje čine particiju početnog skupa podataka. Plavom bojom označen je skup koji se koristi kao testni skup, dok unija preostalih skupova u retku predstavljaju trening skup.



Slika 10: Podjela podataka u K-FCV

Za skupove u svakom retku potrebno je primijeniti kNN algoritam za različite vrijednosti parametra k. U Tablici 1 prikazan je postupak za vrijednosti parametra $k = 1$, $k = 2$ i $k = 3$. Promotrimo slučaj kada je $k = 1$ te promatrajmo različite mogućnosti testnog skupa. Neka $K_2 \cup K_3 \cup K_4 \cup K_5$ čini trening skup i neka je K_1 testni skup. Primijenimo kNN algoritam za $k = 1$ te izračunamo klasifikacijsku stopu točnosti koju označimo s A_{11}. Zatim kao testni skup uzmimo K_2 dok preostali skupovi predstavljaju trening skup. Ponovno primijenimo kNN algoritam za $k = 1$ te izračunamo A_{12}. Nastavljamo postupak za $k = 1$ pri čemu su testni skupovi K_3, K_4 i K_5 te izračunamo redom A_{13}, A_{14} i A_{15}. Uočimo kako smo iskoristili sve mogućnosti testnih skupova te je potrebno odrediti konačnu vrijednost klasifikacijske stope točnosti za $k = 1$.

k vrijednost	Trening skup ($\cup$)	Test skup	Točnost
1	K_2, K_3, K_4, K_5	K_1	A_{11}
1	K_1, K_3, K_4, K_5	K_2	A_{12}
1	K_1, K_2, K_4, K_5	K_3	A_{13}
1	K_1, K_2, K_3, K_5	K_4	A_{14}
1	K_1, K_2, K_3, K_4	K_5	A_{15}
2	K_2, K_3, K_4, K_5	K_1	A_{21}
2	K_1, K_3, K_4, K_5	K_2	A_{22}
2	K_1, K_2, K_4, K_5	K_3	A_{23}
2	K_1, K_2, K_3, K_5	K_4	A_{24}
2	K_1, K_2, K_3, K_4	K_5	A_{25}
3	K_2, K_3, K_4, K_5	K_1	A_{31}
3	K_1, K_3, K_4, K_5	K_2	A_{32}
3	K_1, K_2, K_4, K_5	K_3	A_{33}
3	K_1, K_2, K_3, K_5	K_4	A_{34}
3	K_1, K_2, K_3, K_4	K_5	A_{35}

Tablica 1: Postupak K-FCV

To ćemo napraviti tako da odredimo aritmetičku sredinu prethodno izračunatih klasifikacijskih stopa točnosti, odnosno

$$A_1 = \frac{A_{11} + A_{12} + A_{13} + A_{14} + A_{15}}{5}.$$

Sada promatramo slučaj kada je $k = 2$ i različite mogućnosti testnih skupova te dobijemo klasifikacijske stope točnosti $A_{21}, A_{22}, A_{23}, A_{24}, A_{25}$. Izračunamo klasifikacijsku stopu točnosti za $k = 2$:

$$A_2 = \frac{A_{21} + A_{22} + A_{23} + A_{24} + A_{25}}{5}.$$

Na analogan način odredimo klasifikacijsku stopu točnosti A_3 za $k = 3$. Potrebno je usporediti vrijednosti A_1, A_2 i A_3 te je optimalna vrijednost parametra k, u ovako postavljenom slučaju, onaj broj čija je klasifikacijska stopa točnosti najveća.
Poseban slučaj metode K-kratne unakrsne provjere je *metoda izostavljanja jednog uzorka kod unakrsne provjere* (LOO-CV). U ovom slučaju je $K = n$, pri čemu je n ukupan broj podataka što znači da podatke dijelimo u n skupova i svaki skup ima samo jedan podatak.

4.6 Vrste kNN algoritma

Osim osnovnog i najjednostavnijeg oblika kNN algoritma, postoje njegove brojne varijante. Neke od njih su algoritam k težinski najbližih susjeda (eng. distance-weight k nearest neighbor) i algoritam k najbližih susjeda s lokalnom srednjom vrijednošću (eng. local mean k nearest neighbor algorithm).

Algoritam k težinski najbližih susjeda, odnosno WkNN algoritam, inačica je kNN algoritma u kojoj podacima dodjeljujemo težine. To radimo na način da najbliži podatak ima najveću težinu te se težina smanjuje s povećanjem udaljenosti. Neka je z testni podatak. Udaljenost nekog podatka x iz skupa podataka do testnog podatka z je $d(z, x)$. Definiramo težinu podatka x s obzirom na podatak z, što označavamo s w_x, na sljedeći način:

$$w_x = \frac{1}{d(z, x)}.$$

Uočimo da najbliži podatak poprima najveću težinu jer je njegova udaljenost od testnog podatka najmanja. Za zadani testni objekt z potrebno je odrediti udaljenost od svih objekata iz trening skupa D te odrediti skup N koji sadrži k najbližih susjeda objekta z, kao u kNN algoritmu. Objektima skupa N potrebno je dodijeliti težine i zbrojiti težine objekata unutar iste klase. Testnom objektu z dodjeljuje se ona klasa čiji je zbroj težina najveći.

Algoritam 2 WkNN algoritam [18]

Input: trening skup D,
testni objekt z,
skup klasa L
Output: klasa objekta z, $c_z \in L$
foreach $\mathbf{y} \in D$ **do**
odredi udaljenost između $\mathbf{z}$ i $\mathbf{y}$, $d(\mathbf{z}, \mathbf{y})$;
end
Odaberi skup $N \subseteq D$ od k najbližih susjeda objekta z;
foreach $\mathbf{y} \in N$ **do**
odredi težinu objekta $\mathbf{y}$ s obzirom na $\mathbf{z}$, w_y;
end
$c_z = \arg\max_{v \in L} \sum_{y \in N} I(v = c_y) w_y$, gdje je I indikatorska funkcija.

Algoritam k najbližih susjeda s lokalnom srednjom vrijednošću, odnosno LMkNN algoritam, oblik je kNN algoritma u kojem računamo srednju vrijednost svake klase. Neka je dan testni objekt z, trening skup D i skup klasa L. Potrebno je odrediti udaljenosti objekta z od svih objekata trening skupa D te odrediti

skup N koji sadrži do k najbližih objekata svake klase. Za svaku od klasa skupa L, potrebno je izračunati prosjek pojavljivanja u skupu N. Neka je v klasa iz L, izračunajmo njezin prosjek na sljedeći način:

$$w_v = \frac{1}{k} \sum_{y \in N} I(v = c_y).$$

Algoritam 3 LMkNN algoritam [18]

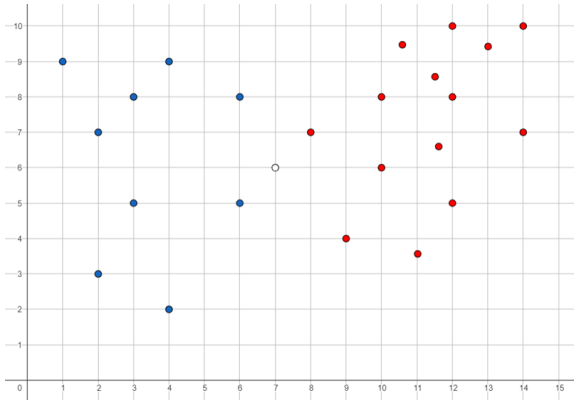
Input: trening skup D,
testni objekt z,
skup klasa L

Output: klasa objekta z, $c_z \in L$

foreach $\mathbf{y} \in D$ **do**
odredi udaljenost između $\mathbf{z}$ i $\mathbf{y}$, $d(\mathbf{z}, \mathbf{y})$;
end
Odaberi skup $N \subseteq D$ koji od svake klase sadrži najviše k najbližih susjeda objekta z;
foreach $\mathbf{v} \in L$ **do**
odredi prosjek $w_v = \frac{1}{k} \sum_{y \in N} I(v = c_y)$;
end
$c_z = \arg\max_{v \in L} w_v$.

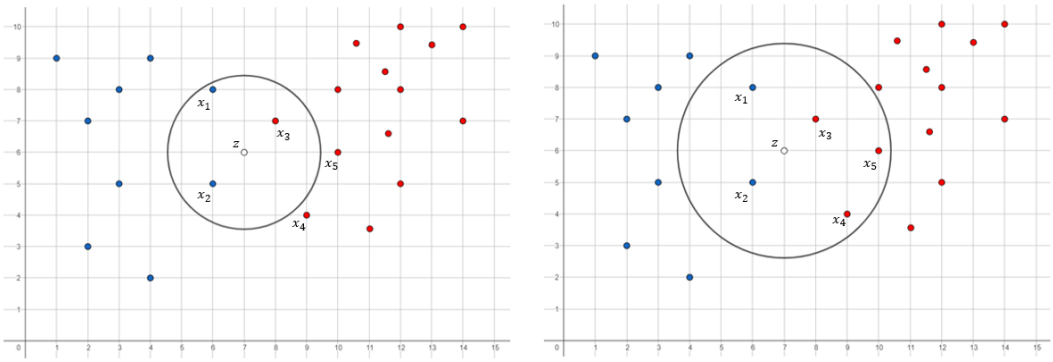
Primijetimo da k u kNN algoritmu označava broj najbližih susjeda, dok u LMkNN algoritmu označava najveći broj najbližih susjeda svake klase. Ukoliko je $k = 1$, LMkNN algoritam odgovara kNN algoritmu.

Primjer 4.5 *Neka je zadan testni objekt koji je u koordinatnom sustavu prikazan u obliku bijelog kružića, $z = (7, 6)$. Trening skup D čine svi plavi i crveni kružići vidljivi u koordinatnom sustavu, dok se skup klasa L sastoji od dviju klasa: crveni i plavi kružići. Pomoću kNN, WkNN i LMkNN algoritma, potrebno je odrediti kojoj od dviju klasa pripada bijeli kružić, pri čemu je $k = 3$. Prilikom računanja udaljenosti koristit će se Euklidska metrika.*



Slika 11: Raspored podataka

Za sva tri algoritma najprije je potrebno odrediti udaljenosti od testnog objekta do svih objekata iz trening skupa. Određivanjem tri najbliža susjeda, kNN algoritam bijelom kružiću dodjeljuje plavu klasu jer iz Slike 12 (lijevo) vidimo da se u tri najbliža susjeda nalaze dva plava i jedan crveni kružić.



Slika 12: Dodjeljivanje klase kNN, WkNN i LMkNN algoritmom

Dodijelimo sada objektu z klasu pomoću WkNN algoritma. Kao i u KNN algoritmu, promatramo tri kružića koja se nalaze najbliže bijelom kružiću. Radi jednostavnosti, označimo: $x_1 = (6, 8), x_2 = (6, 5)$ i $x_3 = (8, 7)$. Odredimo njihove težine s obzirom na objekt z. Vrijedi:

$$w_{x_1} = \frac{1}{d(z, x_1)} = \frac{1}{\sqrt{5}},$$

$$w_{x_2} = \frac{1}{d(z, x_2)} = \frac{1}{\sqrt{2}},$$
$$w_{x_3} = \frac{1}{d(z, x_3)} = \frac{1}{\sqrt{13}}.$$

Potrebno je zbrojiti težine unutar iste klase. Kako su x_1 *i* x_2 *objekti plave klase, njihove težine zbrajamo. Vrijedi:*

$$w_p = w_{x_1} + w_{x_2} = \frac{1}{\sqrt{5}} + \frac{1}{\sqrt{2}},$$

$$w_c = w_{x_3} = \frac{1}{\sqrt{5}}.$$

Vrijedi da je $w_p > w_c$ *pa WkNN algoritam objektu* z *dodjeljuje plavu klasu. Preostaje još odrediti klasu objekta* z *korištenjem LMkNN algoritma. Potrebno je odrediti najviše tri najbliža susjeda plave boje te najviše tri susjeda crvene boje. Na Slici 12 (desno) označeni su primjeri koji zadovoljavaju navedeno. Dva objekta su plave boje dok su tri objekta crvene boje. Primijetimo kako nismo mogli promatrati skup koji sadrži tri plava kružića jer bi tada skup sadržavao četiri crvena kružića što nije moguće zbog* $k = 3$*. Označimo* $x_4 = (9, 4)$ *i* $x_5 = (10, 6)$ *te odredimo prosjek svake klase. Vrijedi:*

$$w_p = \frac{1}{3} \sum_{y \in N} I(p = c_y) = \frac{2}{3},$$

$$w_c = \frac{1}{3} \sum_{y \in N} I(c = c_y) = \frac{3}{3} = 1.$$

Vrijedi da je $w_c > w_p$ *te LMkNN algoritam objektu* z *dodjeljuje crvenu klasu. Uočimo kako sva tri algoritma nisu dodijelila istu klasu objektu* z*. Naime, kNN i WkNN algoritam objektu* z *dodjeljuju plavu klasu, dok LMkNN algoritam objektu* z *dodjeljuje crvenu klasu. Jedan od razloga tome je različito značenje parametra* k*. U kNN algoritmu* k *označava broj najbližih susjeda objekta* z*, dok u LMkNN algoritmu označava najveći broj najbližih susjeda svake klase. Također, u WkNN algoritmu objektima dodjeljujemo težinu, dok u LMkNN algoritmu računamo prosjek pojavljivanja svake klase.*

4.7 Primjena kNN algoritma

Zbog svoje efikasnosti i jednostavnosti u odnosu na druge algoritme, kNN algoritam ima primjenu u raznim područjima iz svakodnevnog života. Koristi se u analizi teksta gdje je jedan od najpopularnijih algoritama za klasifikaciju teksta, u okviru kojeg je potrebno odrediti kojem području pripada određeni tekst (medicina, kultura, sport, ...). kNN algoritam ima brojne primjene u financijama. Koristi se u predviđanju cijena dionica na temelju ekonomskih podataka i uspješnosti tvrtki, u određivanju tečaja valute te određivanju kreditnog rizika. Naime, važno je da banka može unaprijed predvidjeti rizik vezan uz kredit, odnosno vjerojatnost da klijent ne plati i ne vrati cijeli iznos. Na taj se način osigurava da će banka ostvariti profit te da iznos kredita ne premašuje financijske mogućnosti klijenta [2]. kNN algoritam se u financijama koristi prilikom analize podataka o klijentima radi uočavanja navika klijenata te isticanja potencijalnih rizika i prijevara. kNN algoritam se koristi i u medicini za predviđanje i dijagnozu raznih bolesti te otkrivanje čimbenika koji izazivaju neku bolest. Neki od primjera su dijagnosticiranje dijabetesa, predviđanje hoće li pacijent nakon srčanog udara doživjeti i drugi srčani udar, pronalaženje rizičnih čimbenika raka prostate i slično. kNN algoritam koristi se i u poljoprivredi prilikom procjene dnevnih oborina ili nekih drugih vremenskih uvjeta te za prognoziranje klime i za analizu šumskih resursa nekog područja što uključuje procjenu broja i vrste drveća, njihove starosti, prisutnost životinjskih vrsta i slično [7].
Osim primjene u rješavanju navedenih problema, kNN algoritam može se koristiti i u problemima grupiranja gdje je potrebno odrediti neke pravilnosti u uzorku. Ovaj problem, u tom slučaju, ne spada u problem nadziranog učenja, već u problem nenadziranog učenja te se u tom slučaju može reći da je kNN algoritam nenadziranog učenja.

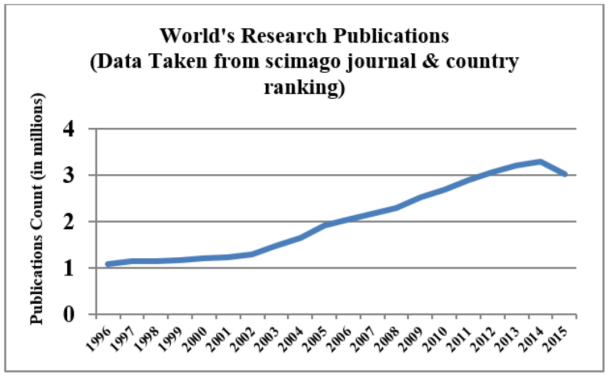
4.8 Svojstva kNN algoritma

Kao i svi algoritmi strojnog učenja, kNN algoritam ima svoje prednosti i mane. Glavna prednost ovog algoritma je njegova jednostavna implementacija i razumljivost, odnosno mogućnost jasnog predočenja što algoritam u kojem trenutku radi i kako funkcionira. Kao algoritam nadziranog učenja, nije namijenjen za rješavanje samo jedne vrste problema, već je njime moguće rješavati i klasifikacijske i regresijske probleme. kNN algoritam ne zahtijeva prethodne pretpostavke o podacima kao što je distribucija podataka, što znači da algoritam neće biti efikasniji ukoliko vrijedi neka pretpostavka o podacima. Na primjer, efikasnost neće ovisiti o tome jesu li podaci normalno

distribuirani ili ne. Zbog toga kažemo da je kNN neparametarski algoritam. Još jedna njegova prednost je visoka razina točnosti u usporedbi s ostalim, kompliciranijim algoritmima. Osim navedenih prednosti, kNN algoritam ima i mane. Jedna od njih je potrošnja velike količine memorije. Razlog tome je pohrana svih podataka trening skupa te je zbog toga kNN "lijeni" algoritam strojnog učenja. Također, svaka nova klasifikacija ili predviđanje zahtijeva ponovni pregled cijelog skupa podataka pa se javlja problem vremenske složenosti pri korištenju velikog broja podataka. Trening proces je vrlo brz jer se sastoji samo od spremanja podataka, no suprotno tome, vrijeme testiranja može biti jako dugo zbog prolaska kroz cijeli trening skup i računanja udaljenosti svakog novog podatka od svih elemenata trening skupa. Jedan od problema koji se može javiti prilikom korištenja kNN algoritma je odabir optimalne vrijednosti parametra k, a prilikom odabira male vrijednosti parametra k, algoritam može biti osjetljiv na šum podataka. Zbog navedenog, možemo zaključiti kako je kNN vrlo koristan algoritam ukoliko se problem sastoji od manjeg broja podataka ili ukoliko na raspolaganju imamo veću količinu memorije i vremena.
Prilikom klasifikacije ili predviđanja vrijednosti novog podatka, kNN prolazi cijelim trening skupom pa se smatra da algoritam ne "uči" te se često postavlja pitanje spada li kNN algoritam u algoritme strojnog učenja.

5 Problem klasifikacije teksta

Jedan od vrlo poznatih i gorućih problema je problem klasifikacije teksta. Zadnjih godina je broj tekstualnih dokumenata jako porastao te se pojavio problem svrstavanja dokumenata u skupine sličnog sadržaja. Na Slici 13 prikazan je porast broja izdanih tekstualnih dokumenata od 1960. do 2015. godine.



Slika 13: Rast broja izdanih tekstualnih dokumenata [3]

Obrada prirodnog jezika (eng. Natural Language Processing) ili kraće NLP, područje je umjetne inteligencije koje se bavi računalnim proučavanjem i razumijevanjem prirodnog jezika, pri čemu pojam prirodni jezik obuhvaća jezik koji se svakodnevno koristi za međusobnu komunikaciju između ljudi [5]. Zbog svakodnevne upotrebe prirodnog jezika, čovjek razumije jezik te većinu zadataka vezanih uz jezik vrlo prirodno odrađuje, što nije slučaj s računalima pa je cilj NLP-a omogućiti računalima razumijevanje prirodnog jezika. Neki od zadataka NLP-a su sažimanje teksta, prevođenje teksta, razdvajanje riječi na morfeme, a jedan od popularnih i čestih zadataka obrade prirodnog jezika je klasifikacija teksta u okviru koje je potrebno odrediti kojoj od unaprijed definiranih kategorija pripada zadani tekst i to na temelju sličnosti s tekstovima koji već pripadaju toj kategoriji. Navedeni problem dodatno otežava činjenica da svaki tekstualni dokument može pripadati jednoj ili više kategorija.
Tekstualni dokumenti mogu se klasificirati u kategorije iz raznih područja kao što su matematika, sport, glazba i slično, a još jedan primjer koji spada u problem klasifikacije teksta je problem razvrstavanja recenzija i komentara. Na primjer, u velikim tvrtkama, poslodavcima je bitno zadovoljstvo radnika i klijenata. Potrebno je analizirati veliki broj anketa i recenzija te ih klasificirati na pozitivne i negativne. Nakon razvrstavanja, moguće je, na primjer, analizirati samo negativne kritike te brže uočiti probleme koji se javljaju. Isto vrijedi za recenzije proizvoda, filmova, usluga te prepoznavanje govora mržnje u komentarima na raznim društvenim mrežama.
Za klasifikaciju teksta primjenjuju se brojne tehnike strojnog učenja kao što su regresijski modeli, stabla odlučivanja, algoritam k najbližih susjeda, neuronske mreže, metoda potpornih vektora i Bayesov klasifikator.

5.1 Obrada teksta i indeksiranje

Neka je zadano K različitih kategorija teksta, $c_1, ..., c_K$, te neka je M ukupan broj dokumenata koje je potrebno klasificirati. Na temelju danih podataka potrebno je odrediti pravilo klasifikacije na temelju kojeg se klasificiraju novi tekstualni dokumenti.

Prije samog procesa klasifikacije teksta, dokumente koji se sastoje od niza znakova potrebno je pretvoriti u oblik pogodan za algoritam učenja. Također, provodi se postupak tokenizacije koji obuhvaća proces razdjeljivanja tekstualnog sadržaja na riječi (tokene), fraze, izraze koji se sastoje od više riječi i slično. Prilikom obrade teksta uobičajeno se koriste sljedeći procesi i transformacije [9]: uklanjanje HTML ili drugih oznaka, uklanjanje čestih stop-riječi te izvođenje korijena riječi. Stop-riječi su one riječi koje se opće-

nito javljaju u svim kategorijama te ne pridonose razlici između pojedinih kategorija pa ih se prilikom obrade teksta redovito izostavlja. Na primjer, riječ "je" javljat će se u svim kategorijama te na temelju nje nećemo moći procijeniti o kojoj se vrsti teksta radi. Izvođenje korijena riječi bitno je radi jednostavnije analize teksta te riječi istog korijena promatramo kao istu riječ. Na primjer, ukoliko se u tekstu pojavljuju glagoli "misli" i "misliti", smatrat ćemo da se dva puta javila ista riječ jer je "misli" korijen riječi "misliti". Iz teksta je potrebno ukloniti oznake kao što su zaglavlje, podnožje, navodnici i slično jer oni ne čine razliku između kategorija. Također, prilikom klasifikacije ne promatramo samo riječi zasebno već i fraze i izraze koji se javljaju unutar teksta.

Nakon navedenih procesa i transformacija, svaki dokument potrebno je prikazati u obliku vektora koji sadrži podatke o riječima koje se javljaju u tom dokumentu. Što se određena riječ češće javlja u nekom dokumentu, to je ona relevantnija za taj dokument. Označimo s f_{ik} frekvenciju riječi i u dokumentu k te neka je n_i broj dokumenata u kojima se pojavljuje riječ i. Potrebno je odrediti težinu riječi i u dokumentu k, što označavamo s a_{ik}. Prilikom određivanja vrijednosti a_{ik}, moguće je koristiti više metoda, a neke od njih su Booleovo ponderiranje, ponderiranje učestalosti riječi i tf-idf ponderiranje [9].
Korištenjem *Booleovog ponderiranja* (eng. Boolean weighting), a_{ik} poprima vrijednost 1 ukoliko se riječ i javlja u dokumentu k, a 0 inače:

$$a_{ik} = \begin{cases} 1, & f_{ik} > 0 \\ 0, & \text{inače} \end{cases} .$$

Ponderiranjem učestalosti riječi (eng. word frequency weighting), težina a_{ik} jednaka je vrijednosti frekvencije f_{ik}, odnosno vrijedi: $a_{ik} = f_{ik}$.
Jedan od najčešće korištenih načina određivanja vrijednosti a_{ik} je tf-idf *ponderiranje* (eng. term frequency-inverse document frequency weighting). Glavna ideja ove metode je da su ključne one riječi koje se češće javljaju u dokumentu, dok se u preostalim dokumentima javljaju rjeđe. Naime, ukoliko se neka riječ često javlja u dokumentu, možemo pretpostaviti kako je ona bitna za taj dokument, no ukoliko se ista riječ često javlja i u preostalim dokumentima, ona neće biti reprezentativna za određeni dokument. Navedeno će vrijediti za stop-riječi jer na temelju njih nećemo moći klasificirati dokumente te ih iz tog razloga nastojimo ukloniti iz dokumenata. Vrijednost tf označava broj pojavljivanja pojma u dokumentu, odnosno $tf = f_{ik}$, dok vrijednost idf označava inverzan broj pojavljivanja riječi u ostalim dokumentima. Drugim riječima, smanjuje se težina riječi koja se često javlja

u drugim dokumentima te vrijedi: $idf = \log\left(\frac{M}{n_i}\right)$. Vrijednost a_{ik} biti će proporcionalna frekvenciji riječi i u dokumentu k i obrnuto proporcionalna broju dokumenata koji sadrže riječ i. Odnosno, kako bi odredili vrijednost a_{ik}, potrebno je pomnožiti vrijednosti tf i idf, dakle vrijedi: $a_{ik} = tf \cdot idf$. Tada je:

$$a_{ik} = f_{ik} \log\left(\frac{M}{n_i}\right).$$

Nakon određivanja težina riječi i izraza koji se nalaze u dokumentima, dokumente prikazujemo u modelu vektorskog prostora. Neka u danih M dokumenata ima ukupno n različitih riječi i izraza. Svaki dokument k prikazujemo u obliku vektora težina:

$$d_k = (a_{1k}, a_{2k}, ..., a_{nk}),\ k = 1, ..., M,$$

gdje je a_{ik} težina riječi i u dokumentu k, pri čemu je $i = 1, ..., n$.

Primjer 5.1 *Neka je dano $M = 10$ tekstualnih dokumenata koje je potrebno klasificirati. Promatrajmo riječ i te odredimo njezinu težinu a_{ik} u dokumentu k, pri čemu pretpostavimo da je frekvencija riječi i u dokumentu k, $f_{ik} = 130$ te da se riječ i pojavljuje u $n_i = 8$ dokumenata.*
Korištenjem Booleovog ponderiranja, a_{ik} će poprimit vrijednost 1 jer vrijedi da je $f_{ik} = 130 > 0$.
Ponderiranjem učestalosti riječi, vrijednost a_{ik} jednaka je vrijednosti frekvencije. Odnosno, $a_{ik} = f_{ik} = 130$.
Ukoliko koristimo tf-idf ponderiranje, potrebno je odrediti vrijednosti tf i idf. Vrijedi:

$$tf = f_{ik} = 130,$$
$$idf = \log\left(\frac{M}{n_i}\right) = \log\left(\frac{10}{8}\right) = \log(1.25) = 0.0969.$$

Vrijednost a_{ik} jednaka je umnošku vrijednosti tf i idf, odnosno vrijedi:

$$a_{ik} = tf \cdot idf = 130 \cdot 0.0969 = 12.597.$$

5.2 Smanjenje dimenzije

Radi poboljšanja učinkovitosti klasifikacije i smanjenja računske složenosti, potrebno je ukloniti riječi koje ne daju bitne informacije za klasifikaciju. Kako bi znali koje je riječi potrebno ukloniti promatramo nekoliko

vrijednosti koje karakteriziraju svaku riječ. Jedna od navedenih vrijednosti je broj dokumenata u kojima se riječ i pojavljuje, odnosno vrijednost n_i. Unaprijed je potrebno definirati vrijednost praga koja predstavlja najmanji broj dokumenata u kojima se riječ mora pojavljivati kako ne bi bila uklonjena. U suprotnom, ako je broj dokumenata u kojima se riječ pojavljuje manji od vrijednosti praga, riječ je potrebno ukloniti te je ne promatramo u procesu klasifikacije. Sljedeća vrijednost pomoću koje odlučujemo zadržavamo li ili uklanjamo riječ, informacijski je dobitak (eng. information gain), u oznaci IG. On označava informaciju namijenjenu za predviđanje kategorije teksta na temelju prisustva ili odsustva riječi iz dokumenta. Informacijski dobitak riječi i dan je sljedećom formulom [21]:

$$IG(i) = \sum_{j=1}^{K} P(c_j)\log(P(c_j)) + P(i)\sum_{j=1}^{K} P(c_j \mid i)\log(P(c_j \mid i)) + P(\bar{i})\sum_{j=1}^{K} P(c_j \mid \bar{i})\log(P(c_j \mid \bar{i})).$$

Vrijednost $P(c_j)$ označava udio dokumenata koji pripadaju klasi c_j, dok se s $P(i)$ označava udio dokumenata u kojima se riječ i pojavljuje, a s $P(\bar{i})$ udio dokumenata u kojima se riječ i ne pojavljuje. Udio dokumenata iz klase c_j koji sadrže riječ i označava se s $P(c_j|i)$, a s $P(c_j|\bar{i})$ označen je udio dokumenata iz klase c_j koji ne sadrže riječ i.
U ovom slučaju uklonit ćemo riječi čiji je informacijski dobitak manji od prethodno definirane vrijednosti praga.

5.2.1 Dekompozicija singularnih vrijednosti

Dekompozicija singularnih vrijednosti (eng. Singular Value Decomposition), odnosno SVD, postupak je kojim se iz nekog skupa podataka, koji je prikazan u matričnom obliku, izdvajaju najvažniji podaci, dok se ostali podaci zanemaruju.
Neka je $\mathbb{F}$ polje i $M_{nm}(\mathbb{F})$ skup matrica reda $n \times m$ nad poljem $\mathbb{F}$.

Definicija 5.2 *Neka je* $A = [a_{ij}] \in M_{nm}(\mathbb{F})$, $i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$. ***Transponirana matrica*** *matrice* A *je matrica* $A^T = [a_{ji}] \in M_{mn}(\mathbb{F})$, $i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$.

Definicija 5.3 *Neka je* $A = [a_{ij}] \in M_{nm}(\mathbb{C})$, $i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$. ***Konjugirana matrica*** *matrice* A *je matrica* $\overline{A} = [\overline{a_{ij}}] \in M_{nm}(\mathbb{C})$, $i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$, *pri čemu za* $a_{ij} = x + iy \in \mathbb{C}, x, y \in \mathbb{R}$, *vrijedi:* $\overline{a_{ij}} = x - iy \in \mathbb{C}$.

Definicija 5.4 *Za matricu $A \in M_n(\mathbb{F})$ kažemo da je* ***dijagonalna*** *ako vrijedi da je $a_{ij} = 0$ za $i \neq j$.*

Definicija 5.5 *Za matricu $A \in M_n(\mathbb{F})$ kažemo da je* ***regularna*** *(invertibilna) ako postoji matrica $X \in M_n(\mathbb{F})$ takva da vrijedi $AX = XA = I$. Tada je X inverzna matrica matrice A te pišemo $X = A^{-1}$. Ukoliko matrica X ne postoji, kažemo da je A* ***singularna matrica****.*

Definicija 5.6 *Za matricu $A \in M_n(\mathbb{F})$ kažemo da je* ***ortogonalna*** *ako vrijedi $A^T A = AA^T = I$.*

Napomena 5.7 *Svaka ortogonalna matrica je regularna te za ortogonalnu matricu A vrijedi: $A^{-1} = A^T$.*

Definicija 5.8 *Neka je $A \in M_{mn}(\mathbb{C})$.* ***Adjungirana matrica*** *matrice A je matrica $A^* = \overline{A}^T = \overline{A^T}$.*

Definicija 5.9 *Za matricu $A \in M_n(\mathbb{C})$ kažemo da je* ***unitarna*** *ako vrijedi $AA^* = A^*A = I$.*

Definicija 5.10 *Neka je $A \in M_{mn}(\mathbb{R})$.* ***Svojstveni vektor*** *matrice A je vektor $\vec{x} \in \mathbb{R}^n$, $\vec{x} \neq \vec{0}$, takav da za neki skalar $\lambda \in \mathbb{R}$ vrijedi $A \cdot \vec{x} = \lambda \cdot \vec{x}$. Skalar λ zove se* ***svojstvena vrijednost*** *matrice A.*

Definicija 5.11 *Neka je $A \in M_{mn}(\mathbb{R})$ te neka je λ svojstvena vrijendost matrice $A^T A$.* ***Singularna vrijednost*** *matrice A je $\sigma = \sqrt{\lambda}$.*

Definicija 5.12 ***Norma*** *na $\mathbb{R}^n$ je preslikavanje $|| \cdot || : \mathbb{R}^n \to \mathbb{R}$ za koje vrijedi:*

i) $||x|| \geq 0, \forall x \in \mathbb{R}^n$,

ii) $||x|| = 0 \Leftrightarrow x = 0$,

iii) $||\lambda x|| = |\lambda| ||x||, \forall \lambda \in \mathbb{R}, \forall x \in \mathbb{R}^n$,

iv) $||x + y|| \leq ||x|| + ||y||, \forall x, y \in \mathbb{R}^n$.

Sljedećim teoremom opisana je SVD dekompozicija [15].

Teorem 5.13 *Neka su m i n ($m \geq n$) prirodni brojevi te A proizvoljna $m \times n$ realna matrica. Tada postoji dekompozicija $A = UDV^T$, gdje je U unitarna matrica reda m, V unitarna matrica reda n i $D = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$, pri čemu vrijedi $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$.*

Stupci matrice $U = [u_1, ..., u_n]$ nazivaju se lijevi singularni vektori, dok se stupci matrice $V = [v_1, ..., v_n]$ nazivaju desni singularni vektori. Vrijednosti $\sigma_i,\ i = 1, ..., n$, nazivaju se singularne vrijednosti matrice A.
Za navedene matrice vrijedi sljedeće:

$$A = UDV^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T.$$

Napomena 5.14 *Ukoliko je $m < n$, tada se SVD dekompozicija primjenjuje na matricu A^T.*

Vrijedi:

$$A^T A = (UDV^T)^T (UDV^T) = (V^T)^T D^T U^T U D V^T = V D^2 V^T,$$

$$AA^T = (UDV^T)(UDV^T)^T = UDV^T (V^T)^T D^T U^T = U D^2 U^T.$$

Teorem 5.15 *Svojstva SVD-a [15]*
Neka je $m \geq n$ te neka je $A = UDV^T$ dekompozicija matrice A na singularne vrijednosti. Vrijede sljedeće tvrdnje:

i) Svojstvene vrijednosti simetrične matrice $A^T A$ su σ_i^2. Desni singularni vektori v_i su pripadni ortonormirani [1] svojstveni vektori.

ii) n svojstvenih vrijednosti simetrične matrice AA^T su σ_i^2, dok je preostalih $m - n$ svojstvenih vrijednosti jednako nula. Desni singularni vektori u_i su pripadni ortonormirani svojstveni vektori za svojstvene vrijednosti σ_i^2.

iii) Neka je $A_k = UD_kV^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, gdje je $D_k = diag(\sigma_1, ..., \sigma_k, 0, ..., 0)$ i $k < n$. Tada matrica A_k ima rang [2] k te je ona najbolja aproksimacija matrice A među svim matricama ranga k.

Primjer 5.16 *Odredimo SVD dekompoziciju matrice A, gdje je*

$$A = \begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

[1] Ortonormirani vektori su ortogonalni vektori čija je duljina jednaka 1.
[2] Rang matrice je broj linearno nezavisnih redaka u matrici.

Potrebno je odrediti svojstvene vrijednosti matrica A^TA i AA^T pomoću kojih određujemo matrice V i U.

$$A^TA = \begin{bmatrix} -1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

Za svojstvene vrijednosti vrijedi: $\lambda_1 + \lambda_2 = 4$,
$\lambda_1 \cdot \lambda_2 = \det(A^TA) = 3$.
Slijedi: $\lambda_1 = 3,\ \lambda_2 = 1$.
Odredimo singularne vrijednosti: $\sigma_1 = \sqrt{\lambda_1} = \sqrt{3}$,
$\sigma_2 = \sqrt{\lambda_2} = \sqrt{1} = 1$.
Odredimo sada matricu V.

$$(A^TA - \lambda_1 I)x_1 = 0 \Rightarrow \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow x_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$||x_1|| = \sqrt{(-1)^2 + 1^2} = \sqrt{2} \Rightarrow v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Euklidska norma vektora $x = (x_1, ..., x_n) \in \mathbb{R}^n$ definirana je na sljedeći način: $||x|| = \sqrt{x_1^2 + ... + x_n^2}$.

$$(A^TA - \lambda_2 I)x_2 = 0 \Rightarrow \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow x_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$||x_2|| = \sqrt{1^2 + 1^2} = \sqrt{2} \Rightarrow v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Slijedi:

$$V = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = V^T.$$

Radi određivanja matrice U, potrebno je odrediti svojstvene vrijednosti matrice AA^T.

$$AA^T = \begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Svojstvene vrijednosti su: $\lambda_1 = 3,\ \lambda_2 = 1$ *i* $\lambda_3 = 0$.
Odredimo sada matricu U.

$$(AA^T - \lambda_1 I)x_1 = 0 \Rightarrow \begin{bmatrix} -1 & 1 & -1 \\ 1 & -2 & 0 \\ -1 & 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow x_1 = \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix}$$

$$||x_1|| = \sqrt{(-2)^2 + (-1)^2 + 1^2} = \sqrt{6} \Rightarrow u_1 = \frac{1}{\sqrt{6}} \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix}$$

$$(AA^T - \lambda_2 I)x_2 = 0 \Rightarrow \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_{21} \\ x_{22} \\ x_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow x_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$||x_2|| = \sqrt{1^2 + 1^2} = \sqrt{2} \Rightarrow u_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$(AA^T - \lambda_3 I)x_3 = 0 \Rightarrow \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow x_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$||x_2|| = \sqrt{1^2 + (-1)^2 + 1^2} = \sqrt{3} \Rightarrow u_3 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Slijedi:

$$U = \begin{bmatrix} -\frac{1}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \end{bmatrix}.$$

Matrica U reda je 3 dok je matrica V reda 2. Zaključujemo da je D matrica reda 3×2 pa zadnji redak nadopunimo s nulama.
Slijedi:

$$D = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Konačno, SVD dekompozicija matrice A je:

$$A = UDV^T = \begin{bmatrix} -\frac{1}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \end{bmatrix} \cdot \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Kako bi SVD metodu mogli primijeniti na problem klasifikacije teksta, potrebno je formirati matricu H koja će sadržavati informacije o tekstu. Matricu ćemo formirati pomoću vektora težina svakog dokumenta na način da su redci matrice riječi koje se pojavljuju u dokumentima, dok su stupci matrice vektori težine svakog dokumenta kojeg treba klasificirati. Odnosno, elementi matrice H su vrijednosti $a_{ik},\ i = 1, ..., n,\ k = 1, ..., M$.

$$H = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1k} & ... & a_{1M} \\ a_{21} & a_{22} & ... & a_{2k} & ... & a_{2M} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & ... & a_{ik} & ... & a_{iM} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{n1} & a_{n2} & ... & a_{nk} & ... & a_{nM} \end{bmatrix}$$

SVD dekompozicijom dobijemo:

$$H = UDV^T,$$

pri čemu je U unitarna matrica reda n, V unitarna matrica reda M te D dijagonalna matrica reda $n \times M$, $D = diag(\sigma_1, ..., \sigma_n)$. Kako bi smanjili dimenziju problema, iz skupa singularnih vrijednosti potrebno je izbaciti male singularne vrijednosti, odnosno koristiti ćemo samo k, $k < n$, najvećih singularnih vrijednosti, pri čemu za ostalih $n - k$ singularnih vrijednosti vrijedi $\sigma_{k+1} = ... = \sigma_n = 0$. Na taj način formiramo matricu $H_k = UD_kV^T$ koja je najbolja aproksimacija matrice H ranga k. Tako je smanjena dimenzija problema, dok su zadržane sve bitne informacije koje služe za klasifikaciju.
SVD dekompozicija standardna je procedura koja se koristi u svrhu smanjenja dimenzije i pojednostavljivanja problema raznih vrsta.

5.3 Klasifikacija teksta primjenom kNN algoritma

kNN je klasifikacijski algoritam strojnog učenja koji se koristi za klasifikaciju teksta, odnosno za određivanje pripadnosti odabranog teksta unaprijed definiranoj kategoriji i to na temelju sličnosti s tekstovima koji već pripadaju toj kategoriji. Nakon prethodno opisanih postupaka obrade teksta, indeksiranja riječi i izraza te smanjenja dimenzije, dokumente je potrebno klasificirati u zadane kategorije.
Neka je M ukupan broj dokumenata koje je potrebno klasificirati u K različitih kategorija, $L = \{c_1, \ldots, c_K\}$. Trening skup se sastoji od ukupno s dokumenata čije kategorije znamo, $D = \{D_1, ..., D_s\}$. Uzmimo dokument

$d_k, k \in \{1, ..., M\}$ i odredimo kategoriju kojoj on pripada. Procesom indeksacije svakoj riječi i izrazu u dokumentima određena je njihova težina u određenom dokumentu. Stoga dokument d_k možemo prikazati kao vektor težina $d_k = (a_{1k}, a_{2k}, ..., a_{nk})$, pri čemu je ukupan broj različitih riječi i izraza koji se pojavljuju u svim dokumentima jednak n. U procesu klasifikacije, potrebno je odrediti udaljenost vektora d_k od vektora svih dokumenata iz trening skupa te odrediti k najbližih vektora, odnosno dokumenata. Možemo primijeniti Algoritam 1, pri čemu za određivanje udaljenosti možemo koristiti Euklidsku metriku:

$$d(d_k, D_j) = \sqrt{\sum_{i=1}^{n}(a_{ik} - a_{ij})^2}, \;\; j = 1, ..., s.$$

Sada algoritam određuje skup $N \subseteq D$ od k najbližih dokumenata dokumenta d_k te dokumentu d_k dodjeljuje kategoriju koja broji najviše elemenata. Odnosno,

$$c_{d_k} = \arg\max_{v \in L} \sum_{y \in N} I(v = c_y).$$

Osim Euklidske metrike, prilikom odabira k najbližih dokumenata može se koristiti funkcija sličnosti. Sličnost je funkcija $sim : \mathbb{R}^n \to \mathbb{R}$ definirana na sljedeći način [21]:

$$sim(d_k, d_j) = \frac{\sum_{i=1}^{n} a_{ik} \cdot a_{ij}}{\sqrt{(\sum_{i=1}^{n} a_{ik})^2} \cdot \sqrt{(\sum_{i=1}^{n} a_{ij})^2}},$$

pri čemu su $d_k = (a_{1k}, a_{2k}, ..., a_{nk}), d_j = (a_{1j}, a_{2j}, ..., a_{nj}) \in \mathbb{R}^n$.
Nakon izračunavanja sličnosti, određuje se skup N od k najsličnijih dokumenata dokumentu d_k, odnosno k dokumenata s najvećom vrijednosti funkcije sličnosti. Korištenjem elemenata skupa N, za svaku kategoriju zasebno određuje se vjerojatnost da dokument d_k pripada upravo toj kategoriji [21]:

$$P(d_k, c_i) = \sum_{d_j \in N} sim(d_k, d_j) \cdot f(d_j, c_i),$$

pri čemu je $f : D \times L \to \{0, 1\}$ funkcija definirana na sljedeći način:

$$f(d_i, c_j) = \begin{cases} 1, & d_i \in c_j \\ 0, & d_i \notin c_j \end{cases}.$$

kNN algoritam tada dokumentu d_k dodjeljuje kategoriju čija je vjerojatnost najveća. Odnosno, vrijedi:

$$c_{d_k} = \arg\max_{v \in L} P(d_k, v).$$

5.3.1 Poboljšani kNN algoritam za klasifikaciju teksta

Zbog smanjenja složenosti pri klasifikaciji teksta primjenom kNN algoritma, koristi se poboljšani kNN algoritam za klasifikaciju teksta [21]. On se temelji na kombinaciji algoritma grupiranja i kNN algoritma. Naime, podaci se najprije grupiraju te se zatim za klasifikaciju koristi kNN algoritam.
Neka je zadano K različitih kategorija, $L = \{c_1, \ldots, c_K\}$, te neka se trening skup D sastoji od ukupno s dokumenata prikazanih u obliku vektora težina čiju kategoriju znamo. Neka je N_{c_i} broj dokumenata koji pripadaju kategoriji c_i, $i \in \{1, ..., K\}$ te označimo s $d_{i1}, d_{i2}, ..., d_{iN_{c_i}}$ dokumente koji pripadaju klasi c_i. Tada vrijedi da je $\sum_{i=1}^{K} N_{c_i} = s$.
Za svaku od K kategorija, potrebno je odrediti centar te kategorije, O_i, $i \in \{1, ..., K\}$. Za izračunavanje j-te koordinate n-dimenzionalnog vektora O_i, koristi se sljedeća formula:

$$O_{ij} = \frac{1}{N_{c_i}} \sum_{l=1}^{N_{c_i}} d_{ilj}.$$

Nakon određivanja centra svake kategorije, potrebno je odrediti Euklidsku udaljenost od centra do svih dokumenata iz te kategorije. Odredimo udaljenost centra kategorije c_i od svih dokumenata koji se nalaze u toj kategoriji na sljedeći način:

$$d(O_i, d_{im}) = \sqrt{\sum_{j=1}^{n} (d_{imj} - O_{ij})^2},\ m \in \{1, ..., N_{c_i}\}.$$

Nakon određivanja centra svake kategorije, uklanjaju se dokumenti koji se nalaze blizu granice između različitih kategorija. Za te dokumente često će vrijediti da se nalaze bliže dokumentima neke druge kategorije nego dokumentima kategorije kojoj pripadaju. Označimo s d_{O_i} vrijednost praga kategorije c_i. Ukoliko za dokument d_{im}, $1 \leq m \leq N_{C_i}$, vrijedi: $d(O_i, d_{im}) > d_{O_i}$, potrebno je ukloniti dokument d_{im}.
Unutar svake kategorije potrebno je provesti postupak grupiranja, odnosno dokumenti unutar iste kategorije grupiraju se u više skupina. Jedan od najpopularnijih algoritama grupiranja je *K-means algoritam*, odnosno algoritam K sredina koji je prikazan Algoritmom 4.

Algoritam 4 K-means algoritam [1]

Input: vektori $x_1, ..., x_n$,
broj skupina, K
Output: središte svake skupine μ_k, $k \in \{1, ..., K\}$
foreach $i \in \{1, ..., n\}$ **do**
$r_i = [0, 0, ..., 0]$;
$k' = RandomInteger(1, K)$;
$r_{ik'} = 1$;
end
while svi r_i ostanu nepromijenjeni
foreach $k \in \{1, .., K\}$ **do**
$N_k = \sum_{i=1}^{n} r_{ik}$;
$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{ik} x_i$;
end
foreach $i \in \{1, ..., n\}$ **do**
$r_i = [0, 0, ..., 0]$;
$k' = \arg\min_k ||x_i - \mu_k||^2$;
$r_{ik'} = 1$;
end

Promatrajmo kategoriju c_i, $i \in \{1, ..., K\}$. Korištenjem K-means algoritma, grupiramo kategoriju c_i u S_i skupina te dobijemo S_i središta, $\{W_{i1}, W_{i2}, ..., W_{iS_i}\}$, koji se koriste kao predstavnici te kategorije. Neka se u svakoj skupini nalazi redom Num_{ij}, $j \in \{1, ..., S_i\}$ dokumenata.
Umjesto početnog trening skupa, kao novi trening skup D' koriste se središta svih kategorija, odnosno $D' = \{W_{11}, W_{1S_1}, W_{21}, ..., W_{2S_2}, ..., W_{K1}, ..., W_{KS_K}\}$. Korištenjem kNN algoritma, određuje se skup $N \subseteq D'$ od k najbližih dokumenata dokumenta d_k te se za svaku kategoriju c_j, $j \in \{1, ..., K\}$, izračunava vjerojatnost da dokument d_k pripada upravo toj kategoriji:

$$P(d_k, c_j) = \sum_{W_{ij} \in N} sim(d_k, W_{ij}) \cdot f(W_{ij}, c_j),$$

pri čemu je $f : D' \times L \to \{0, 1\}$ funkcija definirana na sljedeći način:

$$f(W_{ij}, c_j) = \begin{cases} \frac{Num_{ij}}{N_{c_i}}, & W_{ij} \in c_j \\ 0, & W_{ij} \notin c_j \end{cases}.$$

Dokumentu d_k dodjeljuje se kategorija čija je vjerojatnost najveća. Odnosno, vrijedi:

$$c_{d_k} = \arg\max_{v \in L} P(d_k, v).$$

Algoritmom 5 prikazan je poboljšani kNN algoritam za klasifikaciju teksta.

Algoritam 5 kNN algoritam za klasifikaciju teksta [21]

Input: trening skup D,
testni dokument z, koji je prikazan kao vektor težina,
skup klasa $L = \{c_1, \ldots, c_K\}$,
broj skupina svake kategorije S_i, $i \in \{1, ..., K\}$

Output: kategorija dokumenta z, $c_z \in L$

foreach $i \in \{1, ..., K\}$ **do**
odredi centar O_i;
ukloni dokumente za koje vrijedi $d(O_i, d_{im}) > d_{O_i}$, $1 \leq m \leq N_{C_i}$,
pri čemu je d_{O_i} vrijednost praga;
grupiraj dokumente kategorije C_i primjenom K-means algoritma
te odredi S_i središta $\{W_{i1}, W_{i2}, ..., W_{iS_i}\}$;
odredi broj elemenata u svakoj skupini, Num_{ij}, $j \in \{1, ..., S_i\}$;
središta dodaj u novi trening skup D';

end
Odaberi skup $N \subseteq D'$ od k najbližih susjeda dokumenta z;
foreach v $\in L$ **do**
odredi vjerojatnost $P(z, v) = \sum_{W_{ij} \in N} sim(z, W_{ij}) \cdot f(W_{ij}, v)$;
end
$c_z = \arg\max_{v \in L} P(z, v)$.

6 Programska realizacija problema klasifikacije teksta kNN algoritmom

Zadana je baza tekstualnih dokumenata koja se sastoji od ukupno 2225 dokumenta koji pripadaju pet različitih kategorija: politika(0), sport(1), tehnologija(2), zabava(3), posao(4) [23]. Navedenu bazu u Python dokumentu spremimo pod nazivom "data" te prvi stupac baze, koji sadrži tekstualne dokumente, spremimo pod nazivom "tekst", a drugi, koji sadrži informaciju o tome kojoj kategoriji tekst pripada, spremimo pod nazivom "y".

```
1  import numpy as np
2  import pandas as pd
3
4  import nltk
   ↪  #paket za obradu prirodnog jezika
5  nltk.download('stopwords')
   ↪  #preuzimanje liste stop-riječi
6  nltk.download('punkt')
   ↪  #preuzimanje interpunkcijskih znakova
7  nltk.download('wordnet')
   ↪  #leksička baza podataka (sadrži npr. sinonime i antonime) -
   ↪  potrebno za izvođenje korijena riječi
8  from nltk.tokenize import word_tokenize
   ↪  #preuzimanje funkcije koja dijeli rečenicu na listu tokena
9  from nltk.corpus import stopwords
   ↪  #nltk.corpus sadrži funkcije za čitanje i obradu tekstualnih
   ↪  podataka iz različitih korpusa (zbirka teksta), a na ovaj
   ↪  način pristupamo zbirci stop-riječi
10
11 from sklearn.feature_extraction.text import TfidfVectorizer
12 from nltk.stem import WordNetLemmatizer
13
14 data=pd.read_csv("/content/df_file.csv")
15 tekst=data["Text"]
16 y=data["Label"]
```

Prije klasifikacije, tekstualne dokumente potrebno je obraditi. Potrebno je iz teksta ukloniti znakove i stop-riječi, podijeliti tekst na tokene, izvesti korijen riječi i slično. U Pythonu postoje brojne gotove funkcije koje se koriste u NLP-u, a neke od njih su `word_tokenize()` i `WordNetLemmatizer()` koje služe za podjelu teksta na tokene i izvođenje korijena riječi. Navedena obrada definirana je unutar funkcije "obrada" koju primijenimo na tekstualne dokumente (linije koda od 19 do 37).

```
17 import re
   ↪  #modul za rad s regularnim izrazima
18 stop_rijeci = stopwords.words("english")
   ↪  #lista engleskih stop riječi (npr. end, and...)
19 def obrada(Tekst):
20     Tekst = Tekst.lower()
       ↪  #cijeli tekst piše malim slovima (npr. Sport = sport)
```

```
21      Tekst = Tekst.replace("\n"," ").replace("\t"," ")
        ↪  #oznake za novi red i veće razmake (tabulaciju)
        ↪  zamjenjuju se s jednim razmakom
22      Tekst = re.sub(r'\d+', '', Tekst)
        ↪  #zamjena brojeva s razmakom
23      Tekst = re.sub(r'[^\w\s]', '', Tekst)
        ↪  #uklanjanje svih znakova koji nisu slova, brojevi ili
        ↪  razmaci; r - gledamo i posebne znakove, tj. # se ne bi
        ↪  gledao kao komentar, ^ - negacija, w -slovo, broj ili
        ↪  donja crta, s - razmak
24      Tekst = re.sub("\s+"," ",Tekst)
        ↪  #u tekstu zamjenjuje jedan (\s) ili više (+) razmaka sa
        ↪  samo jednim razmakom
25
26      tokens = word_tokenize(Tekst)
        ↪  #podjela teksta na tokene - zasebne riječi
27
28      data = [i for i in tokens if i not in stop_rijeci]
        ↪  #uklanjanje stop riječi
29
30      # Izvođenje korijena riječi (lemmatization)
31      korijen = WordNetLemmatizer()
32      novi_tekst = []
33      for i in data:
34          rijec = korijen.lemmatize(i)
            ↪  #izvođenje korijena riječi
35          novi_tekst.append(rijec)
36
37      return " ".join(novi_tekst)
        ↪  #spajanje svih korijena u jedan string u kojem su izrazi
        ↪  odvojeni razmakom
38
39  X_tekst = tekst.apply(obrada)
40  X_tekst
```

Napomena 6.1 *Pogledajmo kako obrada teksta izgleda na jednoj proizvoljnoj rečenici. Promatrajmo sljedeću rečenicu: "The ball has a circumference of 68 - 70 cm \n and weighs 410 - 450 g.".*

```
recenica = 'The ball has a circumference of 68 - 70 cm \n and
↪  weighs 410 - 450 g.'
recenica = recenica.lower()
recenica = recenica.replace("\n"," ").replace("\t"," ")
recenica = re.sub(r'\d+', '', recenica)
recenica = re.sub(r'[^\w\s]', '', recenica)
recenica = re.sub("\s+"," ",recenica)
```

Primjenom `.lower()`*, u rečenici će sva velika slova postat mala te će ona izgledati: "the ball has a circumference of 68 - 70 cm \n and weighs 410 - 450 g.". Na izmijenjenu rečenicu primijenimo* `.replace("\n"," ").replace("\t"," ")` *kao bi uklonili znak za novi red pa rečenica sad izgleda: "the ball has a circumference of 68 - 70 cm and weighs 410 - 450 g.". Sljedećom naredbom,* `re.sub(r'\d+', '', recenica)`*, uklanjamo brojeve iz rečenice, odnosno dobijemo rečenicu: "the ball has a circumference of - cm and weighs - g.". Zatim je iz rečenice potrebno ukloniti znakove koji nisu slova, brojevi ili razmaci. Korištenjem naredbe* `re.sub(r'[^\w \s]', '', recenica)` *dobijemo sljedeću rečenicu: "the ball has a circumference of cm and weighs g". Primijetimo kako u rečenici više nama znaka "-" i točke na kraju rečenice. Za kraj još preostaje ukloniti nepotrebne razmake iz rečenice, a to je moguće korištenjem naredbe* `re.sub("\s+"," ",recenica)`*. Nakon obrade teksta rečenica izgleda: "the ball has a circumference of cm and weighs g".*

Na obrađeni tekst potrebno je primijeniti *tf-idf* ponderiranje te na taj način indeksirati riječi unutar teksta. Pomoću parametara funkcije za *tf-idf* ponderiranje moguće je odabirati veličinu izraza koji će se promatrati te odbaciti riječi i izraze koji se prečesto ili prerijetko javljaju te nisu značajni za klasifikaciju.

```
tfidf = TfidfVectorizer(ngram_range=(1,5),max_df=0.95,
↪  max_features=15000)

#ngram_range(1,5) - gledaju se zasebno jedna riječ, dvije
↪  riječi, ..., 5 riječi zajedno
#max_df=0.95 - ako se izraz pojavljuje u više od 95% dokumenata,
↪  ne promatramo ga
#max_features=15000 -koristi se samo 15 000 najvažnijih značajki

X = tfidf.fit_transform(X_tekst)
```

Kada je tekst preoblikovan u oblik pogodan za klasifikaciju, skup tekstualnih dokumenata potrebno je podijeliti na trening skup i testni skup, pri čemu definiramo da se trening skup sastoji od 80% podataka. Na trening skup primijenimo kNN algoritam, pri čemu možemo mijenjati parametar k. Nakon treniranja modela, predviđamo klasu testnog skupa i uspoređujemo predviđene i stvarne vrijednosti testnog skupa kako bi izračunali klasifikacijsku stopu točnosti (linije koda od 48 do 57).

```
48 from sklearn.model_selection import train_test_split
49 X_train, X_test, y_train, y_test =
   ↪  train_test_split(X,y,train_size=0.8,random_state=0)
50
51 from sklearn.neighbors import KNeighborsClassifier
52 knn = KNeighborsClassifier()
53 knn.fit(X_train, y_train)
54 predicted = knn.predict(X_test)
55
56 from sklearn.metrics import accuracy_score
57 accuracy_score(y_test, predicted)
```

Klasifikacijska stopa točnosti primjenom kNN algoritma iznosi 95%.

6.1 Programska realizacija korištenjem poboljšanog kNN algoritma

Prilikom korištenja poboljšanog kNN algoritma, nakon obrade tekstualnih dokumenata, formiramo novu bazu "podaci" koja sadrži obrađene tekstualne dokumente i njihove klase. Dokumente iz navedene baze potrebno je grupirati po klasama te na taj način dobijemo 5 klasa. Na svaku klasu potrebno je primijeniti tf-idf ponderiranje.

```
1 podaci = pd.DataFrame({'tekst':X_tekst,'klasa':y})
2 podaci_klase = podaci.groupby('klasa')   #grupiramo po klasama
3 klasa_0 = podaci_klase.get_group(0)
4 klasa_1 = podaci_klase.get_group(1)
5 klasa_2 = podaci_klase.get_group(2)
6 klasa_3 = podaci_klase.get_group(3)
7 klasa_4 = podaci_klase.get_group(4)
8
9 tfidf = TfidfVectorizer(ngram_range=(1,5),max_df=0.95,
  ↪  max_features=15000)
```

```
klasa0 = tfidf.fit_transform(klasa_0["tekst"])
klasa1 = tfidf.fit_transform(klasa_1["tekst"])
klasa2 = tfidf.fit_transform(klasa_2["tekst"])
klasa3 = tfidf.fit_transform(klasa_3["tekst"])
klasa4 = tfidf.fit_transform(klasa_4["tekst"])
```

Proizvoljno odaberimo broj skupina svake klase, na primjer želimo svaku klasu grupirati u $K = 28$ skupina. Na svaku skupinu primijenimo K-means algoritam te zatim odredimo središta svake klase.

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=28, random_state=1)
model.fit(klasa0)
centri0 = model.cluster_centers_
model.fit(klasa1)
centri1 = model.cluster_centers_
model.fit(klasa2)
centri2 = model.cluster_centers_
model.fit(klasa3)
centri3 = model.cluster_centers_
model.fit(klasa4)
centri4 = model.cluster_centers_
```

Kako bi mogli primijeniti kNN algoritam, dobivena središta spajamo zajedno pod nazivom "pod". Svakom središtu moramo pridružiti njegovu klasu pa u "kl" spremamo po 28 oznaka svake klase. Sada baza podataka na koju primjenjujemo kNN algoritam sadrži ukupno 140 tekstualnih dokumenata.

```
pod = np.vstack([centri0,centri1,centri2,centri3,centri4])
values = np.arange(5)
kl = np.concatenate([np.full(28, value) for value in values])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
↪  train_test_split(pod,kl,test_size=0.8,random_state=0)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

knn.fit(X_train, y_train)
predicted = knn.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predicted)
```

Klasifikacijska stopa točnosti primjenom poboljšanog kNN algoritma iznosi 80%. Možemo primijetiti kako se klasifikacijska stopa točnosti smanjila u odnosu na klasifikacijsku stopu točnosti primjenom kNN algoritma, no primijetimo kako se baza tekstualnih dokumenata smanjila s 2225 na 140, što znači da je smanjena vremenska složenost jer je smanjen skup podataka koji se pregledava prilikom svake nove klasifikacije. Možemo zaključiti da i značajnim smanjenjem veličine baze, klasifikacijska stopa točnosti ostaje vrlo visoka.

7 Zaključak

U ovom radu opisan je kNN algoritam i dvije njegove inačice: algoritam k težinski najbližih susjeda i algoritam k najbližih susjeda s lokalnom srednjom vrijednošću. Jedan od glavnih uvjeta efikasnosti algoritma je odabir optimalne vrijednosti parametra k te su u radu opisane metode odabira parametra k, a neke od njih su: metoda lakta, metoda K-kratne unakrsne provjere i metoda izostavljanja jednog uzorka kod unakrsne provjere.
Algoritam k najbližih susjeda ili kNN algoritam popularan je zbog svoje jednostavne implementacije i predočljivosti. Pri usporedbi s drugim algoritmima, njegova točnost je približno jednaka ili čak i veća. Također, on je neparametarski algoritam pa za njegovu upotrebu nije potrebna prethodna pretpostavka o podacima. Iako je vrlo popularan i često korišten, neke njegove mane su potrošnja velike količine memorije i problem vremenske složenosti pa je zbog tih razloga kNN algoritam bolje koristiti na manjem broju podataka. kNN algoritam primjenjuje se u raznim područjima iz svakodnevnog života kao što su financije i medicina, a vrlo je važan u analizi teksta prilikom rješavanja problema klasifikacije teksta. Prije same klasifikacije, dane tekstualne dokumente potrebno je obraditi i indeksirati kako bi bili pogodni za primjenu algoritma te se zatim primjenjuje kNN algoritam. Osim osnovnog kNN algoritma, u radu je opisan i poboljšani kNN algoritam koji je namijenjen specijalno za klasifikaciju teksta. U navedenom algoritmu podatke je potrebno grupirati u skupine te se zatim centri svake skupine koriste kao trening skup za kNN algoritam. U radu je prikazana programska realizacija rješavanja problema klasifikacije teksta pomoću kNN algoritma i poboljšanog kNN algoritma. Oba algoritma dala su visoku klasifikacijsku stopu točnosti te, iako je klasifikacijska stopa točnosti prilikom korištenja poboljšanog kNN algoritma nešto manja, njegovom primjenom smanjila se veličina baze podataka pa je time smanjena i vremenska složenost te verzije algoritma.

Literatura

[1] Adams, R. P., *K-Means Clustering and Related Algorithms*, Princeton University, New Jersey, SAD, 2018.

[2] Alpaydin, E., *Introduction to Machine Learning*, London, England, 2009.

[3] Azam, M., Ahmed, T., Sabah, F., Hussain, M. I., *Feature Extraction based Text Classification using K-Nearest Neighbor Algorithm* , International Journal of Computer Science and Network Security, VOL.18 No.12, 2018.

[4] Bishop, C. M., *Pattern Recognition and Machine Learning*, Cambridge, UK, 2006.

[5] Chopra, A., Prashar, A., Sain, C., *Natural Language Processing*, Gurgaon ,India, 2013.

[6] Ghahramani, Z., *Unsupervised Learning*, London, UK, 2004.

[7] Imandoust, S. B., Bolandraftar, M., *Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background* , Department of Economics, Payame Noor University, Tehran, Iran, 2013.

[8] Jabri, S., Dahbi, A., Gadi, T., Bassir, A., *Ranking of Text Documents using TF-IDF Weighting and Association Rules mining*, Hassan 1st University, Settat, Morocco, 2018.

[9] Krishnakumar, A., *Text categorization: Building a kNN classifier for the Reuters-21578 collection*, University of California, Santa Cruz, SAD, 2007.

[10] Lane, H., Howard, H., Hapke, C., *Natural Language Processing in Action*, Shelter Island, New York, SAD, 2019.

[11] Maleki, M., Manshouri, N., Kayikcioglu, T., *A Novel Simple Method to Select Optimal k in k-Nearest Neighbor Classifier*, Trabzon, Turkey, 2017.

[12] Manning, C. D., Raghavan, P., Schütze, H., *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, England, 2008.

[13] Markov, Z., Larose, D. T., *Data-mining the Web: uncovering patterns in Web content, structure, and usage*, Central Connecticut State University, New Britain, SAD, 2007.

[14] Mitchell, T. M., *Machine Learning*, McGraw – Hill, New York, SAD, 1997.

[15] Novak, A., Pavlović, D., *Dekompozicija matrice na singularne vrijednosti i primjene*, Hrvatski matematički elektronički časopis, 2013.

[16] Osisanwo, F., Y., Akinsola, J., E., T., Awodele, O., Hinmikaiye, J., O., Olakanmi, O., Akinjobi, J., *Supervised Machine Learning Algorithms: Classification and Comparison*, Ogun State, Nigeria, 2017.

[17] Sutton, R., S., Barto, A,. G., *Reinforcement Learning: An Introduction*, Cambridge, England, 2014.

[18] Syaliman, K., U., Nababan, E., B., Sitompul, O., S., *Improving the accuracy of k-nearest neighbor using local mean based and distance weight*, Universitas Sumatera Utara, Medan, Indonesia, 2017.

[19] Wu, X., Kumar, V., *The Top Ten Algorithms in Data Mining*, University of Minnesota, Minnesota, SAD, 2009.

[20] Yiang, S., Pang, G., Wu, M., Kuang, L., *An impr An improved K-nearest-neighbor algorithm for text categorization est-neighbor algorithm for text categorization*, Singapore Management University , Singapore, 2012.

[21] Yong, Z., Youwen, L., Shixiong, X., *An Improved KNN Text Classification Algorithm Based on Clustering*, China University of Mining and Technology, China, 2016.

[22] Analytic Vidhya, *Guide to K-Nearest Neighbors Algorithm in Machine Learning*, URL: `https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/` (28.08.2024.)

[23] kaggle, URL: `https://www.kaggle.com/datasets/sunilthite/text-document-classification-dataset` (13.08.2024.)

[24] Medium, *Machine Learning basics: K Nearest Neighbors*, URL: `https://medium.com/@pingsubhak/machine-learning-basics-k-nearest-neighbors-9e8e2d46db75` (28.08.2024.)

Popis slika