



UNIVERZITET U NIŠU  
ELEKTRONSKI FAKULTET



Danica Đorđević

# **Komparativna analiza algoritama klasifikacije**

**Tehnike i metode analize podataka**

Mentor: Suzana Stojković

Student: Danica Đorđević 1121

Niš, 2021. god.

# Sadržaj

<b>1. UVOD .....</b>	<b>3</b>
<b>2. PREPROCESIRANJE PODATAKA .....</b>	<b>4</b>
<b>2.1. ČIŠĆENJE PODATAKA .....</b>	<b>4</b>
<b>2.2. TRANSFORMACIJA PODATAKA .....</b>	<b>5</b>
<b>2.3. REDUKCIJA PODATAKA .....</b>	<b>6</b>
<b>3. KLASIFIKACIJA .....</b>	<b>8</b>
<b>3.1. STABLA ODLUKE.....</b>	<b>9</b>
<b>3.2. NAIVNI BAJESOV ALGORITAM .....</b>	<b>12</b>
<b>3.3. K-NAJBЛИŽI SUSEDI ALGORITAM .....</b>	<b>16</b>
<b>3.4. POREĐENJE ALGORITAMA .....</b>	<b>18</b>
<b>4. IMPLEMENTACIJA.....</b>	<b>18</b>
<b>5. ZAKLJUČAK.....</b>	<b>29</b>
<b>6. LITERATURA .....</b>	<b>30</b>

# 1. UVOD

Pojedinci i organizacije svakodnevno generišu veliku količinu podataka. Podaci se zatim skladište u bazama podataka i kasnije obrađuju, izvlače korisne informacije, čitaju ili analiziraju. Danas bi manuelna obrada podataka predstavljala veoma dug i zahtevan proces, koji bi bio sklon greškama. Usled toga, zadatak obrade, pronalaženja obrazaca i analiziranja podataka obavljaju softveri. Softveri koriste mašinsko učenje kako bi radili predikciju ili donosili odluke, a da za to nisu izričito programirani. Algoritmi mašinskog učenja koriste se u širokom spektru aplikacija, poput filtriranja elektronske pošte i računarskog vida, gde je teško ili neizvodljivo razviti konvencionalne algoritme za izvršavanje potrebnih zadataka.

Jedan od procesa, koje danas obaljaju softveri, je klasifikacija. Velika količina podataka često zahteva klasifikaciju, kako bi podaci bili pristupačniji, ali i kako bi se izvukle korisne informacije o podacima.

Pre klasifikacije podataka, korisno je te podatke prvo preprocesirati. To obuhvata uklanjanje nepostojećih vrednosti, nevalidnih podataka, i mnogo toga. Preprocesiranje može poboljšati performanse klasifikatora, te se preporučuje kao obavezan korak pre obrade podataka.

U nastavku rada će biti više reči o preprocesiranju i čišćenju podataka, kao i o klasifikaciji. Biće izvršeno upoređivanje tri algoritma za klasifikaciju: algoritma baziranog na stablima odlučivanja, algoritma K-najbliži suseda i Naivnog Bajesovog algoritma.

## 2. PREPROCESIRANJE PODATAKA

Preprocesiranje podataka je u praksi veoma važno. To je tehnika koja transformiše neobrađene (*eng. raw*) podatke u razumljivije, korisnije i efikasnije formate. Podaci koji se prikupljaju sa različitih izvora mogu biti nepouzđani, pa je neophodno da se ti problemi otklone pre primenjivanja algoritma mašinskog učenja, u suprotnom će donošenje odluka biti, takođe, nepouzđano. Koraci u preprocesiranju podataka su sledeći:

- Čišćenje podataka,
- Transformacija podataka,
- Redukcija podataka.

### 2.1. Čišćenje podataka

U ovom koraku se vrši rukovanje nedostajućim podacima i šumovima. Do nedostatka vrednosti u poljima skupa podataka (*eng. dataset*) može doći zbog brisanja te vrednosti usled nerazumevanja ili nekonzistentnosti, zbog smatranja da je taj podatak nerelevantan u trenutku prikupljanja podataka, zbog lošeg rada programa ili aplikacije, itd. Postoje razne strategije za rešavanje ovog problema. neke od njih su:

- **Brisanje redova koji imaju nedostajuće podatke** - ovim se može znatno smanjiti veličina dataseta i time ugroziti pouzdanost donošenja zaključaka,
- **Ponovno unošenje nedostajućih podataka** - ova metoda je u većini slučajeva neefikasna jer je proces ponovnog unošenja svih nedostajućih vrednosti često previše dug i zahtevan,
- **Korišćenje globalne konstantne** - kod ove metode se nedostajućim poljima mogu dodeliti vrednosti globalno definisanih promenljivih, poput *`Unknown`* ili *0*,
- **Srednja vrednost kolone** - ova metoda popunjava nedostajuća polja vrednošću, koja predstavlja srednju vrednost svih poznatih vrednosti u toj koloni,
- **Srednja vrednost klase** - ova metoda popunjava nedostajuća polja vrednošću, koja predstavlja srednju vrednost svih poznatih vrednosti te klase,
- **Medijana** - ova metoda popunjava nedostajuća polja medijanom te kolone,
- **Nasumična dodela vrednosti** - kod ove metode se poljima koja nemaju unetu vrednost dodeljuje nasumična vrednost nekog polja unutar te kolone,
- **Linearna interpolacija** - kod ove metode se nedostajuća vrednost popunjava srednjom vrednošću gornjeg i donjeg suseda unutar iste kolone,
- **Najučestalija vrednost** - kod ove metode se poljima koja nemaju unetu vrednost dodeljuje najučestalija vrednost unutar te kolone.

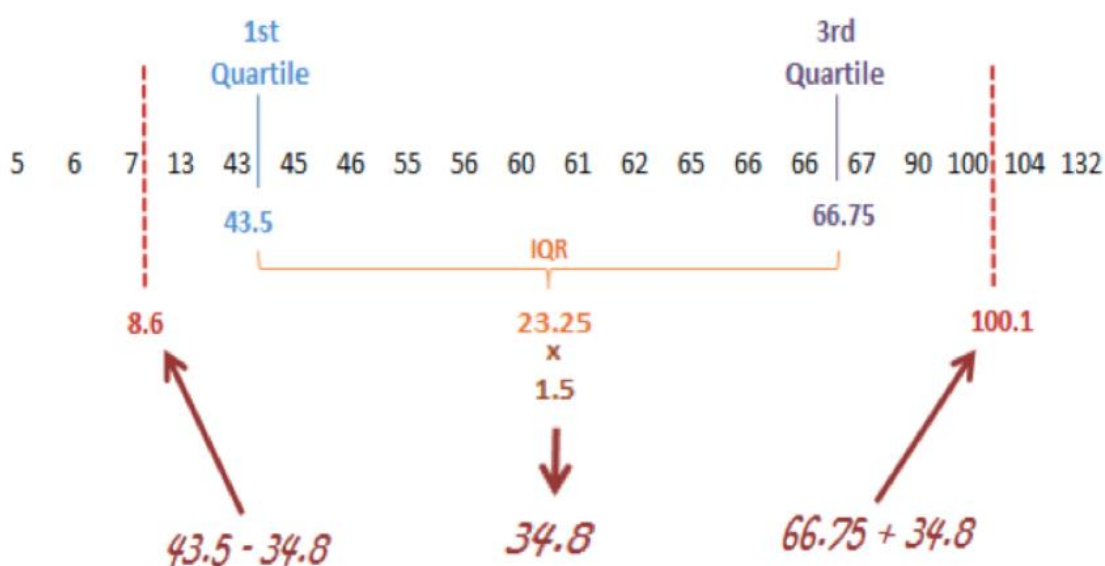
Šumovi predstavljaju podatke koji odstupaju od ostalih vrednosti ili su modifikovani. Šumovi nastaju zbog neispravnog prikupljanja podataka, grešaka pri unosu podataka, itd.

Takođe, šumovi mogu nastati i prirodnim putem. Na primer, ako se prikupljaju podaci o prihodima ljudi, prihod Mark Zuckerberg-a će biti daleko veći od prihoda ostalih ljudi i njegov prihod će se prikazivati kao šum.

U ovom radu je izvršena detekcija vrednosti koje odstupaju (*eng. outlier*), korišćenjem *Tukey* metode. Outlieri su definisani kao:

- $Q1 - 1.5(Q3 - Q1)$  - za vrednosti ispod granice,
- $Q3 + 1.5(Q3 - Q1)$  - za vrednosti iznad granice.

Gde  $Q1$  predstavlja prvi kvartil, a  $Q3$  treći kvartil. Cela metoda pronalaženja odstupanja se zasniva na kvartilima podataka. Prvi kvartil  $Q1$  je vrednost  $\geq 1/4$  podataka, drugi kvartil ili medijana je vrednost  $\geq 1/2$  podataka, a treći kvartil  $K3$  je vrednost  $\geq 3/4$  podataka. Interkvartilni opseg (IQR) predstavlja razliku trećeg i prvog kvartila ( $Q3 - Q1$ ). Kvantili i interkvartilni opseg su prikazani na slici broj 2.



Slika 2. Kvantili i interkvartilni opseg

## 2.2. Transformacija podataka

U ovom koraku se podaci transformišu u odgovarajuće forme, pogodne za data mining. Transformacija podataka može uključivati sledeće metode:

- **Normalizacija** - radi se nad numeričkim tipovima podataka, radi skaliranja vrednosti podataka u određenom opsegu (najčešće u opsegu od 0 do 1 ili od -1 do 1). Najčešće se koristi *Min-Max* tehnika, koja koristi formulu na slici 3, gde  $A'$  predstavlja normalizovanu vrednost, a  $A$  originalnu vrednost,

$$A' = \frac{A - Min}{Max - Min} (newMax - newMin) + newMin$$

Slika 3. Formula koja se koristi u Minmax tehnici

- **Standardizacija** - radi se nad numeričkim tipovima podataka, radi skaliranja vrednosti podataka u zavisnosti od standardne normalne distribucije (nakon standardizacije, srednja vrednost je 0, a standardna devijacija 1). Najčešće se koristi *z-score* tehnika, koja koristi formulu prikazanu na slici 4, gde  $A'$  predstavlja normalizovanu vrednost, a  $A$  originalnu vrednost,

$$A' = \frac{A - Mean}{StandardDeviation}$$

Slika 4. Formula koja se koristi u Z-score tehnici

- **Decimalno skaliranje** - radi se nad numeričkim tipovima podataka, radi skaliranja vrednosti podataka. Kod ove metode se decimalni zarez vrednosti  $A$  pomera za  $j$  pozicija, gde  $j$  predstavlja minimalni broj pozicija tako da apsolutni maksimum uzima vrednosti u opsegu  $[0,1]$ . Ova metoda koristi formulu sa slike 5, gde  $A'$  predstavlja normalizovanu vrednost, a  $A$  originalnu vrednost,

$$A' = \frac{A}{10^j}$$

Slika 5. Formula koja se koristi kod decimalnog skaliranja

- **Selekcija atributa** - kod ovog metoda se novi atributi izvedu iz datog skupa atributa, kako bi pomogli procesu data mining-a.
- **Diskretizacija** - ovde se vrši transformacija kontinualnih atributa u diskretne,
- **Generisanje kocepta hijerarhije** - ovde se atributi pretvaraju iz nižeg nivoa u viši nivo u hijerarhiji. Na primer, atribut "grad" se može pretvoriti u "država".

## 2.3. Redukcija podataka

Data mining je tehnika koja se koristi za rukovanje ogromnom količinom podataka. Prilikom rada sa ogromnom količinom podataka, analiza je sve teža. Da bi analiza bila lakša za obavljanje, koristi se tehnika za smanjenje podataka. Cilj redukcije podataka je povećati efikasnost skladištenja i smanjiti troškove analize i skladištenja podataka. Neki od koraka u redukciji podataka su:

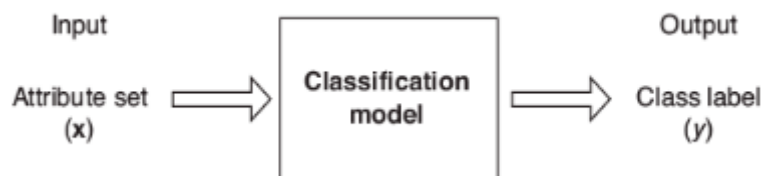
- **Agregacija podataka** - ova tehnika podrazumeva kombinovanje dva ili više atributa u jedan atribut. Podaci se smanjuju primenom *OLAP* operacija poput *slice*, *dice*, *rollup*.
- **Redukcija dimenzionalnosti** - u ovoj metodi se atributi ili dimenzije podataka smanjuju. Nisu svi atributi neophodni za data mining. Najprikladniji podskup atributa bira se

tehnikama poput *PCA* (eng. *Principal Component Analysis* - *PCA*) i *Wavelet*ove transformacije,

- **Brojno smanjenje** (eng. *Numerosity reduction*) - ova tehnika smanjuje obim podataka odabirom manjeg broja podataka. Brojno smanjenje se može izvršiti pomoću histograma, klasterovanja ili semplovanja podataka. Brojno smanjenje je ponekad neophodno jer je obrada celokupnog skupa podataka skupo i dugotrajno.
- **Selekcija podskupa atributa** - kod ove metode se relevantni atributi koriste, a nerelevantni odbacuju. Za obavljanje izbora atributa može se koristiti nivo značajnosti i *p-vrednost* atributa. Atribut koji ima *p-vrednost* veću od nivoa značajnosti može se odbaciti.

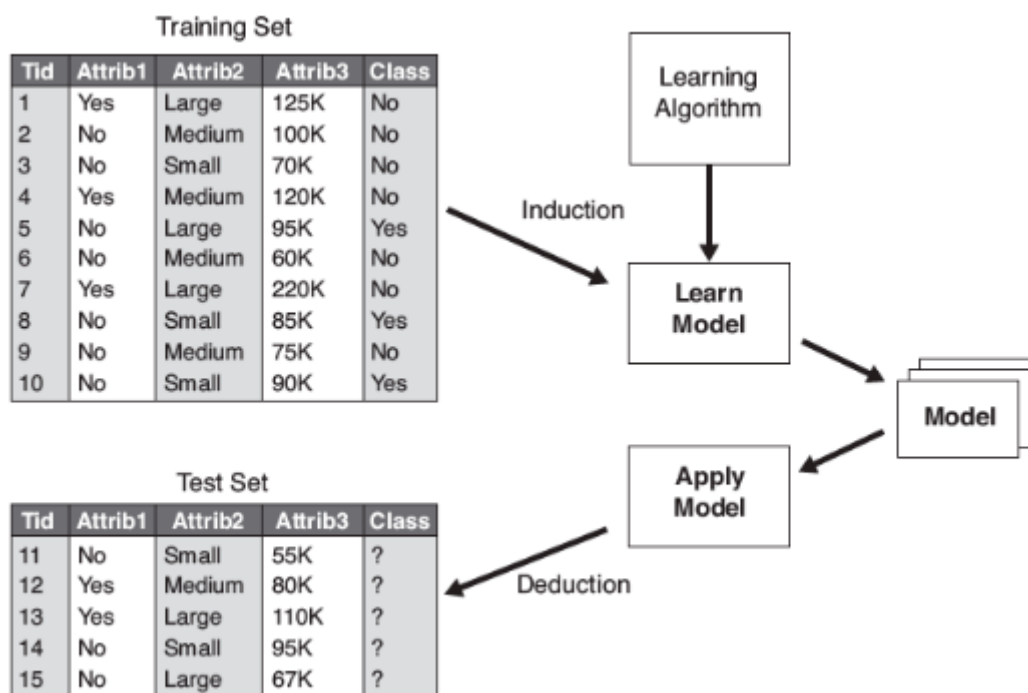
### 3. KLASIFIKACIJA

Klasifikacija je postupak predviđanja klase zadatih podataka. Klase se ponekad nazivaju ciljnim atributima ili karakteristikama. Klasifikaciono prediktivno modeliranje je zadatak približavanja funkcije mapiranja ( $f$ ) od ulaznih promenljivih ( $X$ ) do diskretnih izlaznih promenljivih ( $y$ ). Prikaz rada klasifikacionog modela je dat na sledećoj slici (slika 8):



Slika 8. Rad klasifikacionog modela

Sve tehnike klasifikacije koriste algoritme učenja da bi pronašli model koji najbolje opisuje vezu između atributa i klasne labele trening podataka. Model bi trebalo da vrši predikciju i nepoznatih podataka, koji se nazivaju test podaci. Na sledećoj slici (slika 9) je prikazan proces izgradnje modela za klasifikaciju:



Slika 9. Proces izgradnje modela za klasifikaciju

Na primer, otkrivanje neželjene pošte kod dobavljača usluga e-pošte može se identifikovati kao problem klasifikacije. Ovo je binarna klasifikacija, jer postoje samo 2 klase: neželjena pošta i željena pošta. Klasifikator koristi neke trening podatke da bi razumeo kako se date ulazne promenljive odnose na klasu. U ovom slučaju, poznati podaci o neželjenoj i željenoj pošti se moraju koristiti kao trening podaci. Kada se klasifikator pravilno obuči, može se koristiti za otkrivanje nepoznate e-pošte.

Mnogo je primena klasifikacije u mnogim domenima, poput odobrenja kredita, medicinske dijagnoze, ciljnog marketinga itd. Postoje dve vrste učenika koji se svrstavaju u:



- **Lenje učešnike** - oni čuvaju podatke o treningu i čekaju dok se ne pojave podaci o testiranju. Kada se to dogodi, klasifikacija se vrši na osnovu podataka koji su najviše povezani u uskladištenim trening podacima. U poređenju sa željnim učenicima, lenji učenici imaju manje vremena za obuku, ali više vremena za predikciju. Primeri: K-najbližih suseda, rezonovanje zasnovano na slučaju.
- **Željne učešnike** - oni konstruišu model klasifikacije na osnovu datih trening podataka, pre nego što dobiju podatke za klasifikaciju. Zbog konstrukcije modela, željnim učenicima treba puno vremena za treniranje i manje vremena za predikciju. Primeri: stablo odlučivanja, naivni Bajesov algoritam, veštačke neuronske mreže.

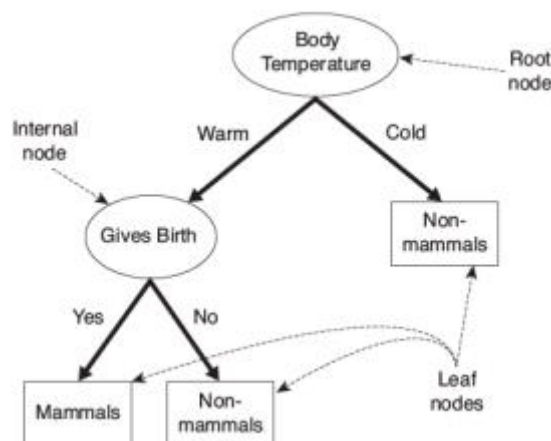
Danas je dostupno mnogo algoritama za klasifikaciju, ali nije moguće zaključiti koji je superiorniji od drugog. U nastavku će biti obrađena tri algoritma za klasifikaciju: algoritam baziran na stablima odluke, algoritam K najbližih suseda i naivni Bajesov algoritam.

### 3.1. Stabla odluke

Klasifikacija korišćenjem stabla odluke je jednostavna i često korišćena tehnika prilikom klasifikacije podataka. Ovakav klasifikator rešava problem tako što postavlja niz pitanja u vezi atributa, kako bi došao do zaključka. Svaki odgovor na postavljeno pitanje sledi podpitanje, sve dok se ne dođe do zaključka tj. sve dok se ne odredi kojoj klasi objekat pripada. Postavljena pitanja i odgovori na ista se mogu organizovati u vidu stabla odluke. Takvo stablo ima određenu hijerarhiju i sastoji se od tipa čvora:

- **Koren** - čvorovi koji ne poseduju ulazne grane, a imaju više ili nijednu izlaznu granu,
- **Interni čvorovi** - čvorovi koji imaju tačno jednu ulaznu granu, a dve ili više izlaznih,
- **Listovi** - čvorovi koji imaju tačno jednu ulaznu granu, a nijednu izlaznu.

U stablu odluke se svakom listu dodeljuje klasna labela. Neterminalni čvorovi, koji uključuju koren i ostale interne čvorove, imaju testirajuće uslove atributa, koji razdvajaju različite karakteristike objekata. Na sledećoj slici (slika 10) je prikazano stablo odluke prilikom klasifikacije sisara:



### Slika 10. Stablo odluke pri klasifikaciji sisara

U zavisnosti od dobijenog seta atributa, može se konstruisati mnogo različitih stabla odluke. Neka od tih stabla vrše precizniju klasifikaciju od drugih. Pronalaženje optimalnog stabla je računski neizvodljivo zbog eksponencijalne veličine prostora za pretragu. Uprkos tome, efikasni algoritmi su uspešno razvijeni da se u razumnom vremenskom intervalu indukuje razumno tačno, mada suboptimalno stablo odluke. Ovi algoritmi najčešće koriste pohlepnu (*eng. greedy*) strategiju za odabir atributa nad koji će se vršiti podela. Jedan takav algoritam se zove Hantov algoritam, koji predstavlja osnovu mnogih drugih algoritama kao što su *CART*, *C4.5* i *ID3*.

U Hantovom algoritmu se stablo odluke rekurzivno gradi, particionisanjem trening skupa na sukcesivne podskupove. Neka je  $D_t$  skup trening podataka, koji su povezani sa čvorom  $t$  i neka je  $y = \{y_1, y_2, \dots, y_c\}$  skup klasnih atributa. U nastavku je data rekurzivna definicija Hantovog algoritma:

- **Korak 1:** Ako svi objekti u skupu  $D_t$  pripadaju istoj klasi  $y_t$ , onda je čvor  $t$  list, označen  $y_t$  klasom.
- **Korak 2:** Ako skup  $D_t$  sadrži objekte koji pripadaju više klasama, testirajući uslov atributa je selektovan radi particionisanja objekta na manje delove. Čvor potomak se kreira za svaki ishod testirajućeg uslova i objekti u skupu  $D_t$  se distribuiraju potomcima u zavisnosti od ishoda. Algoritam se, zatim rekurzivno primenjuje na svaki čvor potomak.

Hantov algoritam će raditi ako je svaka kombinacija vrednosti atributa prisutna u trening podacima i ako svaka kombinacija ima jedinstvenu oznaku klase. Ove pretpostavke su suviše stroge za upotrebu u većini praktičnih situacija. Potrebni su dodatni uslovi za rešavanje sledećih slučajeva:

- Moguće je da su neki od čvorova potomaka, kreiranih u koraku 2, prazni tj. ne postoje objekti povezani sa ovim čvorovima. To se može dogoditi ako nijedan objekat u trening skupu nema kombinaciju vrednosti atributa povezanih sa takvim čvorovima. U ovom slučaju, čvor se proglašava čvorom lista sa istom oznakom klase kao i većina klasa koje imaju objekti trening skupa povezani sa njegovim roditeljskim čvorom.
- U koraku 2, ako svi objekti povezani sa  $D_t$  imaju identične vrednosti atributa (osim oznake klase), tada te zapise nije moguće dalje deliti. U ovom slučaju, čvor je proglašen čvorom lista sa istom oznakom klase kao i većina klasa koje imaju objekti trening skupa povezani sa ovim čvorom.

Algoritam učenja za indukciju stabla odluke mora da obradi sledeća dva pitanja:

- **Kako bi trebalo podeliti trening skup?** - svaki rekurzivni korak u procesu rasta stabla odluke mora odabrati uslov ispitivanja atributa da bi se zapisi podelili na manje podskupove. Da bi primenio ovaj korak, algoritam mora da obezbedi metod za specifikaciju uslova testa za različite tipove atributa, kao i objektivnu meru za procenu kvaliteta svakog testa.
- **Kako bi procedura deljenja trebalo biti zaustavljena?** - Uslov zaustavljanja je potreban da bi se zaustavio proces rasta stabla odluke. Moguća strategija je nastavak širenja čvora sve dok svi objekti ne pripadaju istoj klasi ili svi objekti imaju identične vrednosti atributa. Iako su oba uslova dovoljna da zaustave bilo koji algoritam indukcije stabla odluka, mogu se nametnuti i drugi kriterijumi koji će omogućiti da se postupak rasta stabala ranije završi.

Stvaranje binarnog stabla odluka zapravo je proces podele ulaznog prostora. Pohlepni (*eng. greedy*) pristup se koristi za podelu prostora koji se naziva rekurzivno binarno cepanje. Ovo je numerički postupak u kojem su sve vrednosti poredane i različite tačke podele se isprobavaju i testiraju pomoću funkcije troška. Bira se podela sa najboljim troškovima (najniži trošak jer troškove treba minimizirati). Sve ulazne promenljive i sve moguće tačke razdvajanja procenjuju se i biraju na pohlepan način (npr. najbolja tačka podele se bira svaki put). Za klasifikaciju se koristi funkcija Gini indeksa koja daje pokazatelj koliko su čvorovi lista „čisti” tj. on meri stepen ili verovatnoću da je određena promenljiva pogrešno klasifikovana kada je slučajno izabrana. Gini indeks se definiše kao:

$$G = \sum p_k(1 - p_k)$$

Gde je G Ginijev indeks za sve klase,  $p_k$  je procenat trening primera sa klasom k u pravougaoniku od interesa. Čvor koji ima sve klase istog tipa (savršena čistoća klase) imaće  $G = 0$ , dok će kao G koji ima 50-50 podela klasa za binarni problem klasifikacije (najlošija čistoća) imati  $G = 0,5$ . Za rešavanje binarnog problema, prethodna formula se može napisati kao:

$$G = 2 \cdot p_1 \cdot p_2$$

ili

$$G = 1 - (p_1^2 + p_2^2)$$

Vrednost Gini indeksa za svaki čvor procenjen je ukupnim brojem instanci u roditeljskom čvoru. Ginijev rezultat za odabranu tačku podele u binarnom problemu klasifikacije izračunava se na sledeći način:

$$G = (1 - (g_{11}^2 + g_{12}^2)) \frac{n_{g1}}{n} + (1 - (g_{21}^2 + g_{22}^2)) \frac{n_{g2}}{n}$$

Gde je G Ginijev indeks za tačku razdvajanja,  $g_{11}$  je procenat primeraka u grupi 1 za klasu 1,  $g_{12}$  za klasu 2,  $g_{21}$  za grupu 2 i klasu 1,  $g_{22}$  grupa 2 klase 2,  $n_{g1}$  i  $n_{g2}$  su ukupan broj slučajeva u grupi 1 i 2, i n je ukupan broj instanci koje pokušavamo da grupišemo od roditeljskog čvora.

Gore opisani postupak rekurzivnog binarnog cepanja mora da zna kada treba da zaustavi cepanje dok prolazi kroz stablo sa trening podacima. Najčešći postupak zaustavljanja je korišćenje minimalnog broja trening primera dodeljenih svakom čvoru lista. Ako je broj manji od nekog minimuma, razdvajanje se ne prihvata i čvor se uzima kao završni čvor lista. Broj članova treninga prilagođen je skupu podataka. Definiše koliko će drvo biti specifično za trening podatke. Previše specifično (npr. Ako je broj jednak jedinici) i stablo će prekomerno opremiti trening podatke i verovatno će imati loše performanse na skupu testova.

Kriterijum zaustavljanja je važan jer znatno utiče na performanse stabla odluke. Jedna od tehnika koja se koristi za poboljšanje performansa je *pruning*. Pruning je tehnika kompresije podataka u algoritmima mašinskog učenja i pretraživanja koja smanjuje veličinu stabala odluka uklanjanjem delova stabla koji su nekritični i suvišni za klasifikaciju instanci. Pruning smanjuje složenost konačnog klasifikatora, a time i poboljšava tačnost predviđanja smanjenjem prekomerne opremljenosti (*eng. overfitting*). Složenost stabla odluke definiše se kao broj cepanja u stablu. Poželjnija su jednostavnija stabla. Lako ih je razumeti, a manja je verovatnoća da će prekomerno opremiti podatke. Najbrža i najjednostavnija metoda pruninga

je proći kroz svaki čvor lista na drvetu i proceniti efekat uklanjanja pomoću skupa testova za zadržavanje. Čvorovi lista uklanjaju se samo ako to rezultuje padom funkcije ukupnih troškova na celom skupu testova. Prestaje se sa uklanjanjem čvorova kada ne mogu da se naprave dodatna poboljšanja. Mogu se koristiti sofisticiranije metode pruninga, poput obrezivanja složenosti troškova (takođe se naziva pruning najslabije veze), gde se parametar učenja (alfa) koristi za odluku da li se čvorovi mogu ukloniti na osnovu veličine podstabla.

Dobre strane ovog algoritma jesu to što nije neophodno pripremati, skalirati i normalizovati podatke pre primene algoritma, nedostajuće vrednosti ne utiču značajno na proces izgradnje stabla odluke i ovaj algoritam je veoma lako razumeti. Loše strane su to što mala promena podataka može prouzrokovati veliku promenu u strukturi stabla i može izazvati nestabilnost modela. Sračunavanja koja se vrše u ovom algoritmu ponekad mogu biti mnogo složenija u poređenju sa ostalim algoritmima i često je vreme za obuku modela duže.

## 3.2. Naivni Bajesov algoritam

Naivni Bajesov algoritam je jednostavan, ali iznenađujuće moćan algoritam za prediktivno modeliranje. U mašinskom učenju često postoji zainteresovanost za odabir najbolje hipoteze ( $h$ ) datih podataka ( $d$ ). U klasifikacionom problemu, hipoteza ( $h$ ) može biti klasa koju treba dodeliti za novu instancu podataka ( $d$ ). Bajesova teorema daje način na koji se može izračunati verovatnoća hipoteze, s obzirom na prethodno znanje. Bajesova teorema navodi se kao:

$$P(h|d) = \frac{P(d|h) \cdot P(h)}{P(d)}$$

Gde je  $P(h|d)$  verovatnoća hipoteze  $h$ , s obzirom na podatke  $d$ . To se naziva posterior verovatnoća.  $P(d|h)$  je verovatnoća podataka  $d$ , obzirom da je hipoteza  $h$  bila tačna.  $P(h)$  je verovatnoća da je hipoteza  $h$  tačna (bez obzira na podatke). To se naziva prethodnom verovatnoćom  $h$ .  $P(d)$  je verovatnoća podataka (bez obzira na hipotezu).

Od interesa je izračunavanje posterior verovatnoće  $P(h|d)$  iz prethodne verovatnoće  $p(h)$  sa  $P(d)$  i  $P(d|h)$ . Nakon izračunavanja posteriorne verovatnoće za niz različitih hipoteza, može se odabrati hipoteza sa najvećom verovatnoćom. Ovo je maksimalno verovatna hipoteza i formalno se može nazvati maksimum a posteriori (MAP) hipotezom. Procena maksimalne posteriori verovatnoće (MAP) je procena nepoznate veličine koja je jednaka modu posteriorne raspodele. MAP se može koristiti za dobijanje bodovne procene neopažene veličine na osnovu empirijskih podataka. Usko je povezan sa metodom procene maksimalne verodostojnosti (*eng. maximum likelihood*), ali koristi prošireni cilj optimizacije koji uključuje prethodnu raspodelu (koja kvantifikuje dodatne informacije dostupne putem prethodnog znanja o povezanom događaju) nad količinom koju neko želi da proceni. Procena MAP-a se stoga može smatrati regularizacijom procene maksimalne verovatnoće. Ovo se može zapisati kao:

$$\begin{aligned} \text{MAP}(h) &= \max (P(h|d)) \\ &\text{ili} \\ \text{MAP}(h) &= \max \left( \frac{P(d|h) \cdot P(h)}{P(d)} \right) \\ &\text{ili} \\ &12 \end{aligned}$$

$$\text{MAP}(h) = \max (P(d|h) \cdot P(h))$$

$P(d)$  je normalizujući pojam koji omogućava izračunavanje verovatnoće. Može se izostaviti kada nas zanima najverovatnija hipoteza, jer je konstantna i koristi se samo za normalizaciju. Ako u trening podacima postoji paran broj primeraka u svakoj klasi, verovatnoća svake klase (npr.  $P(h)$ ) biće jednaka. Ovo bi bio konstantan pojam u jednačini i  $P(h)$  bi se mogao izostaviti, tako da jednačina sada izgleda ovako:

$$\text{MAP}(h) = \max (P(d|h))$$

Naivni Bajes je algoritam klasifikacije za binarne i višeklasne probleme klasifikacije. Tehniku je najlakše razumeti kada se opisuje pomoću binarnih ili kategoričkih ulaznih vrednosti. Nazvan je naivnim Bajesom ili idiotskim Bajesom, jer je proračun verovatnoće za svaku hipotezu pojednostavljen kako bi njihov proračun bio izvodljiv. Umesto da pokušavaju da izračunaju vrednosti svake vrednosti atributa  $P(d_1, d_2, d_3 | h)$ , pretpostavlja se da su uslovno nezavisne s obzirom na ciljnu vrednost i izračunate kao  $P(d_1 | h) * P(d_2 | H)$  itd. Ovo je vrlo moćna pretpostavka koja je malo verovatna u stvarnim podacima, tj. malo je verovatno da atributi ne interaguju međusobno. Ipak, pristup iznenađujuće dobro radi na podacima za koje ova pretpostavka ne važi.

Reprezentacija za naivni Bajesov algoritam je verovatnoća. Lista verovatnoća se čuva u evidenciji za naučeni naivni Bajesov model. Ovo uključuje:

- **Verovatnoću klase:** Verovatnoće svake klase u skupu trening podataka.
- **Uslovne verovatnoće:** Uslovne verovatnoće svake ulazne vrednosti date za svaku vrednost klase.

Trening je brz jer treba izračunati samo verovatnoću svake klase i verovatnoću svake klase s obzirom na različite ulazne vrednosti. Postupcima optimizacije nije potrebno podešavati koeficijente.

Verovatnoće klase su jednostavno učestalost instanci koje pripadaju svakoj klasi podeljena ukupnim brojem instanci. Na primer, u binarnoj klasifikaciji verovatnoća instance koja pripada klasi 1 izračunava se kao:

$$P(klasa = 1) = \frac{broj(klasa = 1)}{broj(klasa = 0) + broj(klasa = 1)}$$

U najjednostavnijem slučaju svaka klasa bi imala verovatnoću od 0,5 ili 50% za binarni problem klasifikacije sa istim brojem instanci u svakoj klasi.

Uslovne verovatnoće su učestalost svake vrednosti atributa za datu vrednost klase podeljena učestalošću slučajeva sa tom vrednošću klase. Na primer, ako atribut „vreme“ ima vrednosti „sunčano“ i „kišovito“, a atribut klase ima vrednost klase „izlazak“ i „ostani kod kuće“, tada su uslovne verovatnoće svake vremenske vrednosti za svaku klasu vrednost se može izračunati kao:

$$P(vreme = sunčano | klasa = izlazak) = \frac{broj(instance\ sa\ vremenom = sunčano\ i\ klasa = izlazak)}{broj(instance\ sa\ klasom = izlazak)}$$

$$P(vreme = sunčano | klasa = boravak\ kod\ kuće) = \frac{broj(instance\ sa\ vremenom = sunčano\ i\ klasa = boravak\ kod\ kuće)}{broj(instance\ sa\ nastavom = boravak\ kod\ kuće)}$$

$$P(vreme = kiša | klasa = izlazak) = \frac{broj(instance\ sa\ vremenom = kiša\ i\ klasa = izlazak)}{broj(instance\ sa\ klasom = izlazak)}$$

$$P(vreme = kiša | klasa = boravak\ kod\ kuće) = \frac{broj(instance\ sa\ vremenom = kiša\ i\ klasa = boravak\ kod\ kuće)}{broj(instance\ sa\ nastavom = boravak\ kod\ kuće)}$$

S obzirom na naivni Bajesov model, može se vršiti predviđanje novih podataka korišćenjem Bajesove teoreme:

$$MAPA(h) = \max(P(d | h) \cdot P(h))$$

Koristeći gornji primer, ako postoji nova instanca sa sunčanim vremenom, može se izračunati:

$$\begin{aligned} izlazak &= P(vreme = sunčano | klasa = izlazak) \cdot P(klasa = izlazak) \\ ostati\ kod\ kuće &= P(vreme = sunčano | klasa = ostati\ kod\ kuće) \cdot P(klasa = ostati\ kod\ kuće) \end{aligned}$$

Može se odabrati klasa koja ima najveću izračunatu vrednost. Ove vrednosti se mogu pretvoriti u verovatnoće normalizacijom na sledeći način:

$$\begin{aligned} P(izlazak | vreme = sunčano) &= \frac{izlazak}{izlazak + ostati\ kod\ kuće} \\ P(ostati\ kod\ kuće | vreme = sunčano) &= \frac{ostati\ kod\ kuće}{izlazak + ostati\ kod\ kuće} \end{aligned}$$

Ako bi postojalo više ulaznih promenljivih gornji primer bi se mogao proširiti. Na primer, ako se doda atribut „automobil“ sa vrednostima „radi“ i „pokvaren“. Ovu verovatnoću treba pomnožiti u jednačinu.

U proračunu ispod je prikazana jednačina za oznaku klase „izlazak“ sa dodatkom ulazne promenljive automobila podešene na „radi“:

$$izlazak = P(vreme = sunčano | klasa = izlazak) \cdot P(automobil = radi | klasa = izlazak) \cdot P(klasa = izlazak)$$

Iznad su izračunate verovatnoće za ulazne vrednosti za svaku klasu korišćenjem frekvencije. Pomoću unosa sa stvarnom vrednošću može se izračunati srednja i standardna devijacija ulaznih vrednosti za svaku klasu da bi se sumirala raspodela. To znači da pored verovatnoće za svaku klasu, moramo sačuvati i srednju vrednost i standardna odstupanja za svaku ulaznu promenljivu za svaku klasu. Ovo je jednostavno izračunavanje srednjih vrednosti i vrednosti standardne devijacije svake ulazne promenljive (k) za svaku vrednost klase.

$$srednja\ vrednost(k) = \frac{1}{n} \sum (k)$$

Gde je n broj slučajeva, a k vrednosti za ulaznu promenljivu u trening podacima. Standardnu devijaciju možemo izračunati pomoću sledeće jednačine:

$$standardna\ devijacija(k) = \sqrt{\frac{1}{n} \sum (x_i - srednja\ vrednost(k))^2}$$

Ovo je kvadratni koren prosečne kvadratne razlike svake vrednosti k od srednje vrednosti k, gde je n broj primeraka,  $x_i$  je specifična vrednost k promenljive za i-tu instancu, srednja vrednost (k) je navedena u formuli iznad.

Verovatnoće novih k vrednosti izračunavaju se pomoću Gausove funkcije gustine verovatnoće (eng. *Gaussian Probability Density Function - PDF*). Prilikom predviđanja ovi parametri se mogu uključiti u Gausov PDF sa novim ulazom za promenljivu, a zauzvrat Gausov PDF će dati procenu verovatnoće te nove vrednosti unosa za tu klasu.

$$PDF(x, mean, sd) = \frac{1}{\sqrt{2\pi} sd} e^{-\frac{x-mean^2}{2sd^2}}$$

Gde je pdf(k) Gausov PDF, mean i sd su srednja vrednost i standardna devijacija izračunate gore, PI je numerička konstanta, e je numerička konstanta ili Ojlerov broj podignut na stepen i k je ulazna vrednost za ulaznu promenljivu. Tada možemo uključiti verovatnoće u gornju jednačinu da bismo predviđali ulaze sa stvarnom vrednošću. Na primer, prilagođavanje jednog od gornjih proračuna numeričkim vrednostima za vreme i automobil:

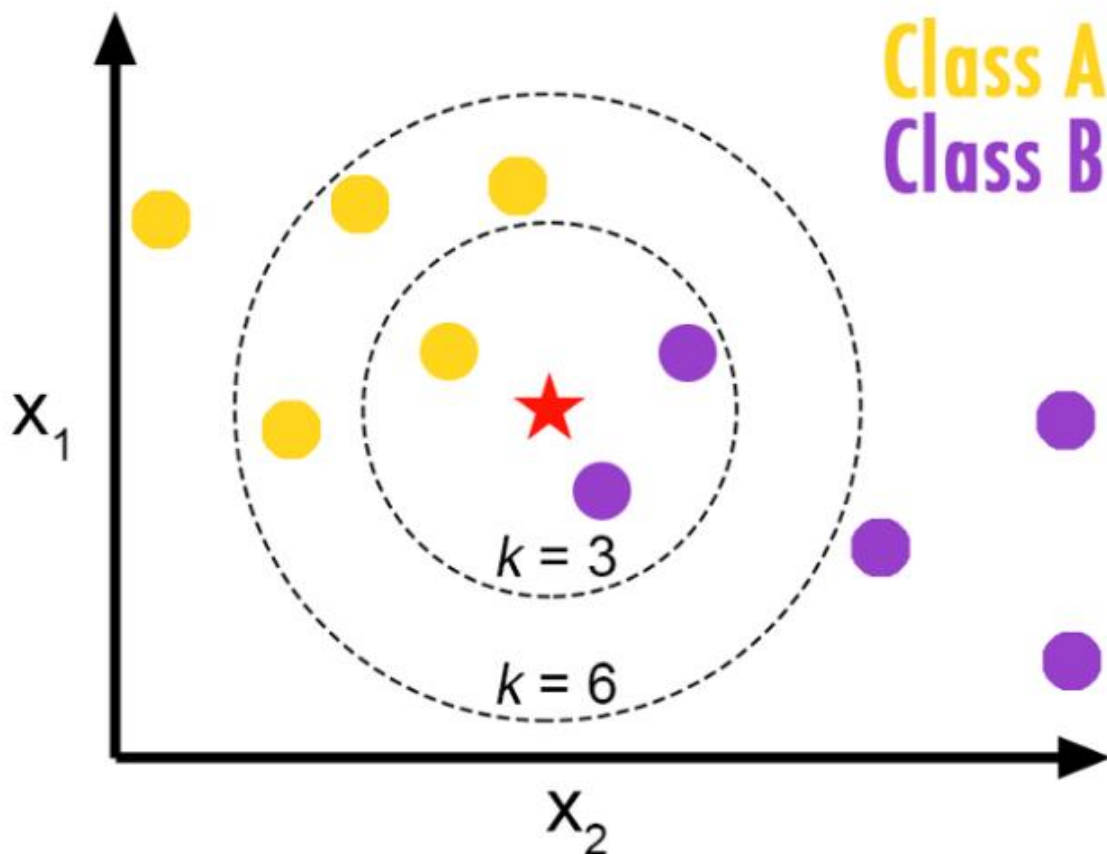
$$izlazak = P(pdf(vreme) | klasa = izlazak) * P(pdf(automobil) | klasa = izlazak) * P(klasa = izlazak)$$

Ovaj algoritam je dobar jer zahteva malu količinu trening podataka da bi procenio test podatke tj. ima kraći period obuke. Takođe, lako ga je implementirati, ali loša strana je to što naivni Bajesov algoritam pretpostavlja da su svi atributi međusobno nezavisni. U stvarnom

životu gotovo je nemoguće da dobijemo skup atributa koji su potpuno nezavisni. Jos jedna loša strana ovo algoritma je to što će svi objekti sa kategoričkim promenljivama, koje nisu zabeležene prilikom treninga, a prisutne su u test skupu, imati verovatnoći jednaku nuli. Ovo je često poznato kao nulta frekvencija.

### 3.3. K-najbliži susedi algoritam

Algoritam K-najbliži suseda (KNN) je jednostavan algoritam, koji se primenjuje kod nadgledanog učenja za rešavanje problema klasifikacije i regresije. Klasifikacioni problem vrši predikciju diskretnih vrednosti, a regresioni vrši predikciju realnih vrednosti. Ovaj algoritam pretpostavlja da se slične stvari nalaze u neposrednoj blizini. Algoritam izračunava udaljenost nove tačke podataka do svih ostalih tačaka trening podataka. Udaljenost može biti bilo koje vrste, npr. Euklidsko ili Manhattansko rastojanje itd. Zatim se bira K najbližih tačka, gde K može biti bilo koji ceo broj. Na kraju algoritam, tačku dodeljuje klasi kojoj pripada većina tačaka K podataka. Ovaj algoritam spada u grupu neparametarskih i budući da je takav, često je uspešan u situacijama klasifikacije kada je granica odluke vrlo nepravilna.



Slika 16. Primer primene algoritma K-najbližih suseda

Koraci u KNN algoritmu su sledeći:



- Učitati podatke
- Inicijalizovati K
- Za svaku tačku u podacima odraditi:
  - Izračunati udaljenost između upitne tačke i trenutne tačke
  - Dodati rastojanje i indeks u uređenu kolekciju
- Sortirati uređenu kolekciju rastojanja i indeksa u rastućem redosledu, prema rastojanjima
- Izabrati prvih K elemenata iz sortirane kolekcije
- Izdvojiti labele izabranih K elemenata
- Vratiti labelu koja se najviše pojavljuje u tih K elemenata

Rad ovog algoritma se može opisati sa gornje slike. Crvena zvezda predstavlja test tačku čija je vrednost (2, 1, 3). Test tačka okružena je žutim i ljubičastim tačkama koje predstavljaju podatke, koji pripadaju dvema klasama. Zatim se izračunava udaljenost od test tačke do svake tačke na grafikonu. Pošto ima 10 tačaka, dobijaju se 10 rastojanja. Utvrđuje se najniža udaljenost i predviđa da pripada istoj klasi najbližeg suseda. Ako je žuta tačka najbliža, onda predviđamo da je test tačka takođe žuta tačka. Ali, u nekim slučajevima mogu se dobiti i dve razdaljine koje su tačno jednake. Onda se uzima u obzir treća tačka podataka i izračunava njena udaljenost od test tačke (test podataka). Na gornjem dijagramu (slika 16), test tačka se nalazi između žute i ljubičaste tačke. Tada se razmatra udaljenost od treće tačke podataka i predviđa da test tačka pripada ljubičastoj klasi. U ovom primeru je za K uzeta vrednost 3.

Da bi se izabralo K koje najviše odgovara izabranim podacima, pokreće se KNN algoritam nekoliko puta sa različitim vrednostima K i bira K koje smanjuje broj grešaka na koje se nailazi, zadržavajući sposobnost algoritma da tačno pravi predviđanja kada dobije podatke koje nema i koje nije video ranije. Treba imati na umu da sa porastom vrednosti K, predviđanja postaju stabilnija.

KNN algoritam izračunava udaljenost između tačaka podataka. Za ovo se koristi jednostavna formula Euklidovog rastojanja. Formula je prikazana na slici 17.

$$\begin{aligned}
 d(\mathbf{p}, \mathbf{q}) &= d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \\
 &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.
 \end{aligned}$$

Slika 17. Formula za izračunavanje Euklidskog rastojanja

Gornja formula uzima n broj dimenzija tj. u mašinskom učenju n predstavlja broj atributa. Podrazumeva se da tačka podataka koja se nalazi na minimalnoj udaljenosti od ispitne tačke pripada istoj klasi. Gornja formula deluje isto u n broju dimenzija i stoga se može koristiti sa n brojem karakteristika tj. atributa.

Dobre strane ovog algoritma jesu jednostavna implementacija i to što se novi podaci, koji neće uticati na tačnost algoritma, mogu neprimetno dodavati, s obzirom da ovaj algoritam ne zahteva nikakav trening pre predviđanja. Loše strane jesu te što je neophodno izvršiti standardizaciju i normalizaciju podataka pre primene KNN algoritma na bilo koji skup podataka, ne radi dobro sa velikim dimenzijama jer sa velikim brojem dimenzija postaje teško izračunati udaljenost u svakoj dimenziji, ne radi dobro sa prevelikim skupovima podataka jer su troškovi izračunavanja udaljenosti velike i osetljiv je na odstupanja (*eng. outlier*) i nedostajuće podatke.

### 3.4. Poređenje algoritama

Sva tri algoritma imaju svoje prednosti i mane. Nijedan od njih nije superiorniji od drugih u svim pogledima. Svaki od njih će davati različite rezultate nad istim skupom podataka. Rezultati tj. performanse algoritma zavise od brojnih faktora, kao što su: struktura podataka, međusobna zavisnost između atributa, tip podataka, itd. Na primer, ako postoji skup podataka u kome su atributi međusobno povezani, korišćenje naivnog Bajesovog algoritma najverovatnije nije dobra odluka. Ovaj algoritam pretpostavlja da su svi atributi međusobno nezavisni i kao takav, neće davati najbolje rezultate u ovom primeru, te se treba okrenuti drugim alternativnim algoritmima. Sa druge strane, ako postoji skup podataka u kome su atributi međusobno nezavisni, naivni Bajesov algoritam bi bio najbolja opcija.

Algoritam K-najbližih suseda je veoma lak za implementaciju i lako razumljiv. Kako je ovde proces testiranja zanemarljiv, ovaj algoritam je dobra opcija ako je cilj uštedeti vreme koje je potroši na testiranje. Sa druge strane, ako skup podataka ima veliki broj atributa, onda ovaj algoritam treba zaobići jer je vreme, koje je potrebno za sračunavanje rastojanja, preveliko. Takođe, ovaj algoritam zahteva prethodno skaliranje podataka, što predstavlja dodatni zahtev prilikom njegovog korišćenja.

Mnogi smatraju da su algoritmi bazirani na stablima odluka najmoćniji prilikom klasifikacije. Klasifikator predviđa kojoj od klasa pripada nova tačka podataka na osnovu stabla odlučivanja. Ovaj algoritam je lako razumeti, radi i na linearnim i na nelinearnim problemima. A kao njegova najveća prednost se uzima to što nije neophodno izvršiti nikakvo skaliranje podataka pre upotrebe algoritma. U poglavlju 3 se može videti da preprocesiranje podataka ima najmanji uticaj na performanse algoritma baziranog na stablima odluke. To znači da ovaj algoritam može biti pouzdan iako podaci nisu prethodno modifikovani. Loša strana ovog algoritma je to što može imati loše rezultate nad vrlo malim skupovima podataka i lako može doći do overfitting-a. Ako je cilj obraditi podatke bez nekog velikog preprocesiranja, onda je ovaj algoritam dobra opcija, jer preprocesiranje ne utiče znatno na njegove performanse.

## 4. IMPLEMENTACIJA

U ovom radu je odrađeno preprocesiranje podataka, kao i upoređivanje tri algoritama za klasifikaciju. Algoritmi koji su korišćeni za klasifikaciju su:

- Naivni Bajesov algoritam,
- CART algoritam,
- K-najbližih suseda algoritam.

Iskorišćeni su algoritmi implementirani u biblioteci *sklearn*. Pre propuštanja podataka kroz algoritme, odrađeno je preprocesiranje podataka u cilju poboljšanja performansi modela.

U projektu je korišćen dataset koji je izvučen iz popisnog biroa. Treba predvideti da li osoba godišnje zaradi preko 50 hiljada dolara. Dataset ima 15 atributa, uključujući i atribut za koji se vrši predikcija tj. uključujući i klasni atribut. Ovaj dataset inicijalno sadrži 48,842 redova, ali za potrebe ovog projekta je izdvojeno 6204 reda. Svaki red sadrži sledeće atribute:

- **age** - atribut kazuje koliko godina ima pojedinac,
- **workclass** - atribut predstavlja u kom radnom odnosu je pojedinac. Ovaj atribut ima 8 jedinstvenih vrednosti,
- **fnlwgt** - atribut predstavlja finalnu težinu, koju cenzus određuje,
- **education** - atribut predstavlja nivo obrazovanja koji je pojedinac postigao. Ovaj atribut ima 16 jedinstvenih vrednosti,
- **education\_num** - atribut predstavlja stepen obrazovanja,
- **marital\_status** - atribut predstavlja bračni stanje pojedinca. Ovaj atribut ima 7 jedinstvenih vrednosti,
- **occupation** - atribut predstavlja čime se pojedinac bavi. ovaj atribut ima 14 jedinstvenih vrednosti,
- **relationship** - atribut koji predstavlja šta je pojedinac u odnosu na druge. Ovaj atribut ima 6 jedinstvenih vrednosti,
- **race** - atribut koji predstavlja rasu pojedinca. Ovaj atribut ima 5 jedinstvenih vrednosti,
- **sex** - atribut koji kazuje da li je osoba muškog ili ženskog pola (atribut ima 2 jedinstvene vrednosti),
- **capital\_gain** - atribut koji predstavlja stečeni kapital osobe,
- **capital\_loss** - atribut koji predstavlja izgubljeni kapital osobe,
- **hours\_per\_week** - atribut koji predstavlja broj radnih sati pojedinca na nedeljnom nivou,
- **native\_country** - atribut koji kazuje iz koje države potiče pojedinac. Ovaj atribut ima 40 jedinstvenih vrednosti,
- **income** - labela koja govori da li osoba zarađuje manje ili više od 50 hiljada godišnje.

```

Atribut 'workclass' ima 8 jedinstvenih vrednosti
Atribut 'education' ima 16 jedinstvenih vrednosti
Atribut 'marital_status' ima 7 jedinstvenih vrednosti
Atribut 'occupation' ima 14 jedinstvenih vrednosti
Atribut 'relationship' ima 6 jedinstvenih vrednosti
Atribut 'race' ima 5 jedinstvenih vrednosti
Atribut 'sex' ima 2 jedinstvenih vrednosti
Atribut 'native_country' ima 40 jedinstvenih vrednosti

```

Slika 18. Prikaz broja jedinstvenih vrednosti za svaki od atributa

Prvo je izvršeno učitavanje dataseta korišćenjem *pandas* biblioteke. Ova biblioteka učitane fajlove predstavlja u obliku DataFrame-a. DataFrame predstavlja tabelarnu reprezentaciju podataka. Podaci imaju dve dimenzije: redove i kolone. Zatim je izvršena podela tog DataFrame-a na dva dela:

- *X* - predstavlja redove sa svim karakteristikama, bez kolone za koju se vrši predikcija (bez klasnog atributa) tj. bez *income* kolone,
- *y* - predstavlja kolonu za koju se vrši predikcija i sadrži vrednosti kolone *income*.

Za dalju obradu podataka, neophodno je da se uoče distribucije vrednosti kako bi se videlo koji atributi se mogu potencijalno modifikovati. Distribucije kategoričkih atributa su date na slikama 19, 20, 21, 22, 23, 24, 25, 26 i 27:

```

United-States    5545
Mexico           138
?                108
Philippines      32
Canada           28
Germany          28
Puerto-Rico     23
Jamaica          22
China            18
South            18
Name: native_country, dtype: int64

```

Slika 19. Distribucija vrednosti atributa *native\_country*

```

Private          4338
Self-emp-not-inc 501
Local-gov        380
?                356
State-gov        245
Self-emp-inc     213
Federal-gov      170
Never-worked     1
Name: workclass, dtype: int64

```

Slika 20. Distribucija vrednosti atributa *workclass*

HS-grad	2007
Some-college	1451
Bachelors	997
Masters	311
Assoc-voc	263
Assoc-acdm	205
11th	202
10th	187
7th-8th	134
Prof-school	112

Name: education, dtype: int64

Slika 21. Distribucija vrednosti atributa *education*

Married-civ-spouse	2844
Never-married	2056
Divorced	852
Separated	207
Widowed	164
Married-spouse-absent	77
Married-AF-spouse	4

Name: marital\_status, dtype: int64

Slika 22. Distribucija vrednosti atributa *marital\_status*

Prof-specialty	793
Exec-managerial	755
Adm-clerical	751
Craft-repair	741
Sales	717
Other-service	627
Machine-op-inspct	384
?	357
Transport-moving	311
Handlers-cleaners	255

Name: occupation, dtype: int64

Slika 23. Distribucija vrednosti atributa *occupation*

Husband	2506
Not-in-family	1559
Own-child	1004
Unmarried	640
Wife	302
Other-relative	193

Name: relationship, dtype: int64

Slika 24. Distribucija vrednosti atributa *relationship*

```

White          5332
Black          565
Asian-Pac-Islander 185
Amer-Indian-Eskimo 62
Other          60
Name: race, dtype: int64

```

Slika 25. Distribucija vrednosti atributa *race*

```

Male          4143
Female        2061
Name: sex, dtype: int64

```

Slika 26. Distribucija vrednosti atributa *sex*

```

<=50K        4721
>50K         1483
Name: income, dtype: int64

```

Slika 27. Distribucija vrednosti atributa *income*

Pošto atribut *income* ima nenumeričke vrednosti, neophodno je te vrednosti prevesti u numeričke tipa *int*. Ovo je urađeno u funkciji prikazanoj na sledećoj slici (slika 28):

```

def _replace_outcome_values(y):
    for i in range(len(y)):
        val = y[i].strip()
        if val == ">50K":
            y[i] = 1
        else:
            y[i] = 0
    return y

```

Slika 28. Prikaz funkcije koja vrši prevođenje tipa klasnog atributa u tip *int*

Ova funkcija prolazi kroz niz *y* i vrši zamenu vrednosti ">50K" jedinicom, a vrednosti "<=50K" nulom.

Na slici distribucije atributa *native\_country* se može videti da je većina ljudi (skoro 90%) iz Sredinjenih Američkih Država, pa se modifikacija može uraditi baš na ovom atributu. Cilj je smanjiti broj jedinstvenih vrednosti za ovaj atribut i umesto četrdeset jedinstvenih vrednosti, imati dve. Treba zameniti vrednosti sa niskom frekventnošću pojavljivanja labelom *Other*. Tako da sada postoje samo dve jedinstvene vrednosti za atribut *native\_country*: *Other* i *United-States*. Zamena je izvršena u funkciji prikazanoj na sledećoj slici (slika 29):

```

def _replace_low_frequency_countries_with_other(X):
    X['native_country'] = ['United-States' if x ==
                           'United-States' else 'Other' for x in X['native_country']]
    return X

```

Slika 29. Funkcija koja vrši zamenu vrednosti sa niskom frekvencom pojavljivanja labelom *Other*

Ovaj dataset poseduje veliki broj kategoričkih atributa tj. atributa koji nemaju numeričku vrednost. Pošto modeli rade samo sa numeričkim vrednostima, neophodno je ove attribute prevesti iz nenumeričke u numeričke oblike. Biblioteka *pandas* nudi rešenje za ovaj problem, korišćenjem *dummies* atributa. Ovaj metod za svaku jedinstvenu vrednost koju kolona poseduje, pravi nove kolone, a staru kolonu briše. Tako, na primer, ako se primeni ova metoda na kolonu *race*, dobiće se 5 novih kolona sa binarnim vrednostima, dok će stara *race* kolona biti izbrisana. Primer primenjivanja ove metode na kolonu *race* je dat na slici 30:

	Amer-Indian-Eskimo	Asian-Pac-Islander	Black	Other	White
0	0	0	0	0	1
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1

Slika 30. Primenjivanje *dummies* metode na kolonu *race*

U ovom primeru se mogu videti 5 novih kolona, gde vrednost 1 u koloni označava da pojedinac pripada toj rasi, a vrednost 0 označava da pojedinac ne pripada toj rasi.

Korišćeni dataset ima nedostajućih vrednosti, pa je neophodno popuniti ta polja određenim vrednostima, u zavisnosti od korišćene strategije. Za popunjavanje nedostajućih vrednosti je korišćena strategija srednje vrednosti, što znači da su se nedostajuće vrednosti u kolonama popunjavale srednjom vrednošću tih kolona. Za popunjavanje polja je korišćen *SimpleImputer* iz *sklearn* biblioteke. Ovo je implementirano u funkciji prikazanoj na slici 31:

```
def _impute_missing_values(X):
    si = SimpleImputer(missing_values=np.nan, strategy='mean')
    si.fit(X)
    X = pd.DataFrame(data=si.transform(X), columns=X.columns)
    return X
```

Slika 31. Funkcija koja vrši zamenu nedostajućih vrednosti

Zatim je izvršena detekcija outlier-a za određene kolone. Nije vršeno uklanjanje istih, već samo detekcija. Detekcija je izvršena *tukey* metodom u funkciji prikazanoj na slici 32:

```
def _find_outliers_tukey(x):
    q1 = np.percentile(x, 25)
    q3 = np.percentile(x, 75)
    iqr = q3-q1
    limit_min = q1 - 1.5*iqr
    limit_max = q3 + 1.5*iqr
    outlier_indices = list(x.index[(x < limit_min) | (x > limit_max)])
    outlier_values = list(x[outlier_indices])

    return outlier_indices, outlier_values
```

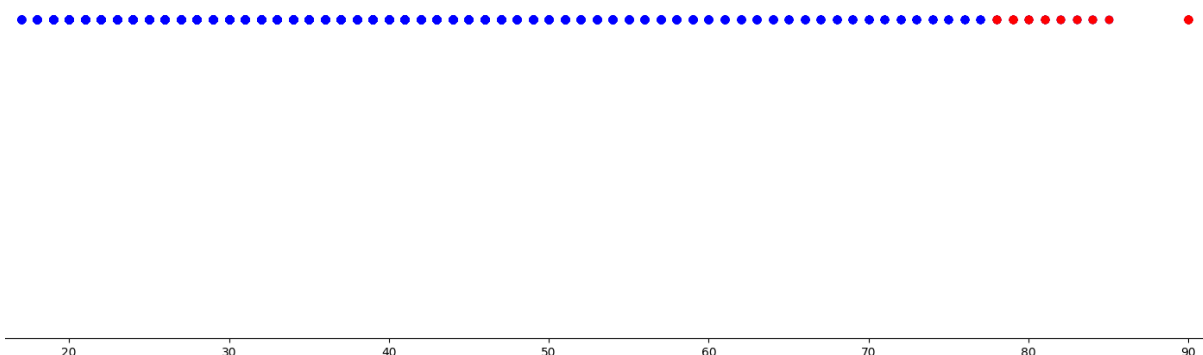
Slika 32. Funkcija koja vrši detekciju outlier-a

Vrednosti outlier-a kolone *age* su prikazani na sledećoj slici (slika 33):

```
'Outliers tukey'
array([78., 78., 78., 78., 79., 79., 79., 79., 79., 79., 80., 80., 80.,
       80., 81., 81., 81., 81., 82., 82., 83., 83., 84., 84., 85., 90.,
       90., 90., 90., 90., 90., 90., 90., 90., 90.])
```

Slika 33. Odstupanja tj. outlier-i za kolonu *age*

Vizuelni prikaz distribucije vrednosti outlier-a i svih vrednosti kolone *age* su prikazani na slici 34:



Slika 34. Vizuelni prikaz odstupanja vrednosti kolone *age*

Na ovoj slici se mogu videti vrednosti izvan granica i one su prikazane crvenom bojom, dok su vrednosti koje ne odstupaju od uobičajenih vrednosti prikazane plavom bojom.

Često ulazne karakteristike interaguju na neočekivan i nelinearni način prilikom prediktivnog modeliranja. Te interakcije se mogu identifikovati i modelirati pomoću algoritma za učenje. Drugi pristup je osmišljavanje novih karakteristika koje uočavaju ove interakcije i utvrđivanje da li poboljšavaju performanse modela. Pored toga, transformacije poput podizanja ulaznih promenljivih na stepen mogu pomoći u boljem uočavanju važnih odnosa između ulaznih promenljivih i ciljne promenljive tj. promenljive za koju se vrši predikcija. Ove karakteristike nazivaju se interakcije i polinomne karakteristike i omogućavaju upotrebu jednostavnijih algoritama za modeliranje, jer se tumačenja složenih



ulaznih promenljivih i njihovih odnosa pomeraju u fazu pripreme podataka. Ponekad ove karakteristike mogu poboljšati performanse modeliranja, iako po cenu dodavanja hiljada ili čak miliona dodatnih ulaznih promenljivih. U implementaciji je izvršeno dodavanje interakcija i polinomialnih karakteristika. To je urađeno u funkciji prikazanoj na sledećoj slici (slika 35):

```
def _add_interactions(df):
    combos = list(combinations(list(df.columns), 2))
    colnames = list(df.columns) + ['_'.join(x) for x in combos]

    poly = PolynomialFeatures(interaction_only=True, include_bias=False)
    df = poly.fit_transform(df)
    df = pd.DataFrame(df, columns=colnames)

    all_zero_values = [i for i, x in enumerate(list((df == 0).all())) if x]
    df = df.drop(df.columns[all_zero_values], axis=1)

    return df
```

Slika 35. Funkcija koja vrši dodavanje interakcija

Dodate interakcije znatno povećavaju broj kolona. Prvih 5 redova dataframe-a sa dodatim interakcijama je prikazan na sledećoj slici (slika 36):

```
0  age  fnlwtg  education_num  capital_gain  ...  race_ White_sex_ Male  race_ White_native_country_Other  sex_ Female_native_country_Other  sex_ Male_native_country_Other
1  72.0  177226.0  9.0  2829.0  ...  1.0  1.0  0.0  1.0
2  31.0  259931.0  4.0  0.0  ...  1.0  1.0  0.0  1.0
3  55.0  189528.0  4.0  0.0  ...  1.0  1.0  0.0  1.0
4  38.0  34996.0  10.0  0.0  ...  0.0  1.0  1.0  0.0

[5 rows x 1630 columns]
```

Slika 36. Izgled prvih 5 redova dataframe-a nakon dodavanja interakcija

Na ovoj slici se može videti da dataframe sada ima 1630 kolona, što je znatno više od početnih 15. Procesiranje tolikog broja kolona nije efikasno, pa je neophodno izvršiti i redukciju broja kolona tj. neophodno je izvršiti redukciju dimenzionalnosti. To je može uraditi korišćenjem PCA(Principal Component Analysis) metode iz biblioteke *sklearn*. Prikaz funkcije koja implementira ovu funkcionalnost je dat na sledećoj slici (slika 37):

```
def _pca(X):
    pca = PCA(n_components=10)
    X_pca = pd.DataFrame(pca.fit_transform(X))
    return X_pca
```

Slika 37. Implementacija PCA metode

Dataframe nakon redukcije dimenzionalnosti ima 10 kolona, ali te kolone nisu više povezane sa prethodnim, već nove kolone dobijaju nove numeričke nazive. Izgled dataframe-a nakon redukcije dimenzionalnosti korišćenjem PCA metode je prikazan na slici 38:

	0	1	2	3	4	5	6	7	8	9
0	4.603680e+08	-1.565683e+07	7.088114e+05	1.704761e+06	1.473788e+05	-52838.823539	233399.450121	153546.197217	167612.864361	-86417.009491
1	-2.043396e+08	-1.651429e+07	-5.935961e+05	8.347809e+06	1.072941e+05	-685.779770	162372.075717	86929.540199	-19285.642779	-80394.262033
2	-2.043383e+08	-1.648310e+07	2.642779e+06	-1.177344e+06	-1.268122e+06	-27744.038930	209406.504532	124026.132287	48292.490478	96700.176617
3	-2.043374e+08	-1.646142e+07	4.896327e+06	-9.190717e+04	-1.858992e+06	-32092.377953	93546.539915	42950.529469	-38964.830372	68647.632896
4	-2.043425e+08	-1.659134e+07	-8.516450e+06	1.705205e+04	-2.587862e+05	-45291.892752	1504.131647	-54915.983056	3047.100899	24799.014780
...	...	...	...	...	...	...	...	...	...	...
6199	-2.043395e+08	-1.651589e+07	-7.595636e+05	-3.905519e+06	-2.964177e+05	-35100.408105	-190262.990397	-66513.436771	128721.817277	-60992.357319
6200	-2.043389e+08	-1.649903e+07	9.907504e+05	2.198097e+06	-4.030509e+05	-29237.665461	-221283.534397	-33696.082799	74679.528787	-74820.967918
6201	-2.043352e+08	-1.640423e+07	1.075960e+07	2.442868e+06	-9.611259e+05	-12889.197566	-225086.311551	177070.498337	-209992.013130	-88907.992400
6202	-2.043399e+08	-1.652669e+07	-1.881780e+06	-2.694645e+06	1.389111e+05	-33471.930392	-218206.516380	6130.424226	65592.354062	-130637.564220
6203	-2.043373e+08	-1.645879e+07	5.129190e+06	5.330803e+04	-5.904585e+05	-20196.586251	-116003.375779	176838.162313	-13040.662382	-122795.358481

Slika 38. Izgled kolona nakon redukcije PCA metodom

Na slici se može videti da imena novih kolona nemaju nikakvo značenje, a i vrednosti u kolonama je teško razumeti. Zbog teškog tumačenja atributa i njihovih vrednosti, odlučeno je da se u implementaciji ne koristi PCA metoda za redukciju dimenzionalnosti.

Za redukciju dimenzionalnosti je iskorišćena metoda *SelectKBest* iz biblioteke *sklearn*. Ova metoda pronalazi k najrelevantnijih atributa. Rangiranje se vrši korišćenjem *f\_classif* algoritma. Ovaj algoritam je nezavisan od prediktivnog metoda koji se koristi. Ovom metodom su izdvojeni samo 16 najbitnijih kolona. Kod koji vrši selekciju najbitnijih atributa je prikazan na sledećoj slici (slika 39):

```
select = sklearn.feature_selection.SelectKBest(f_classif,k=16)
selected_features = select.fit(X_train, y_train)
indices_selected = selected_features.get_support(indices=True)
colnames_selected = [X.columns[i] for i in indices_selected]

X_train_selected = X_train[colnames_selected]
X_test_selected = X_test[colnames_selected]
```

Slika 39. Selekcija k najrelevantnijih atributa

Imena selektovanih kolona su prikazana na slici 40:

```
'Selected columns: 16'
['marital_status_Married-civ-spouse',
 'relationship_Husband',
 'age_education_num',
 'age_marital_status_Married-civ-spouse',
 'age_relationship_Husband',
 'education_num_marital_status_Married-civ-spouse',
 'education_num_relationship_Husband',
 'hours_per_week_marital_status_Married-civ-spouse',
 'hours_per_week_relationship_Husband',
 'marital_status_Married-civ-spouse_relationship_Husband',
 'marital_status_Married-civ-spouse_race_White',
 'marital_status_Married-civ-spouse_sex_Male',
 'marital_status_Married-civ-spouse_native_country_Other',
 'relationship_Husband_race_White',
 'relationship_Husband_sex_Male',
 'relationship_Husband_native_country_Other']
```

Slika 40. Rezultat izvršene selekcije k najrelevantnijih atributa

Takođe, izvršena je i kros validacija skupa podataka na trening i test skup. Veličina trening skupa predstavlja 90% celokupnog skupa podataka. Kros validacija je izvršena korišćenjem biblioteke *sklearn*, a kod je prikazan na slici 41:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.9, random_state=1)
```

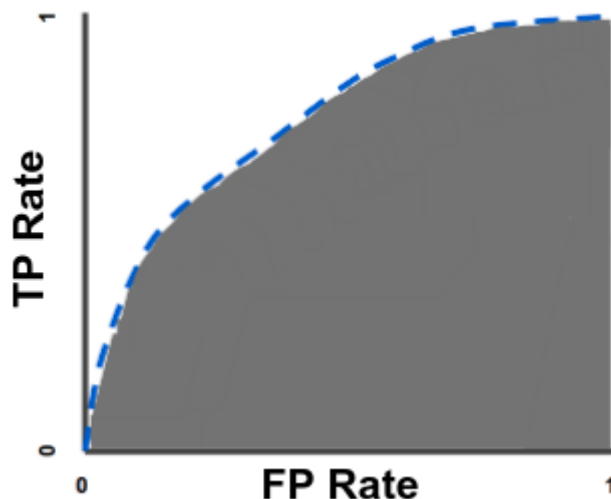
Slika 41. Kros validacija

Ovim je korak preprocesiranja podataka završen. Zatim se prelazi na treniranje modela i upoređivanje rezultata pre i nakon preprocesiranja podataka. Iz skupa neprocesiranih podataka su izbačene kolone sa nenumeričkim podacima, kao i redovi sa nedostajućim vrednostima. Nenumeričke podatke je neophodno izbaciti, da bi primenjivanje bilo kog algoritma bilo moguće. Izgled skupa neprocesiranih podataka je dat na sledećoj slici (slika 42):

	age	fnlwtg	education_num	capital_gain	capital_loss	hours_per_week
0	38	234962	9	2829	0	30
1	72	177226	9	0	0	8
2	31	259931	4	0	0	40
3	55	189528	4	0	0	60
4	38	34996	10	0	0	40

Slika 42. Izgled prvih 5 redova neprocesiranih podataka

Izvršena je klasifikacija korišćenjem tri algoritama. Za evaluaciju algoritama je korišćena AUC (*Area Under the ROC Curve*) mera. AUC meri celokupno dvodimenzionalno područje ispod cele ROC krive od (0,0) do (1,1). Izgled područja ispod ROC krive je prikazano na sledećoj slici (slika 43):



Slika 43. Područje ispod ROC krive

Jedan od načina tumačenja AUC je verovatnoća da model rangira slučajni pozitivni primer više od slučajnog negativnog primera. AUC se kreće u opsegu od 0 do 1. Model čija su predviđanja 100% pogrešna ima AUC od 0,0. Onaj model čija su predviđanja 100% tačna

ima AUC 1,0. Rezultati klasifikacije pre i nakon preprocesiranja podataka su prikazani na slici 44:

```
'----- TREE -----'  
'AUC modela sa preprocesiranjem: 0.7206521739130434'  
'AUC modela bez preprocesiranja: 0.6681677018633541'  
'Poboljsanje modela: 7.8549848942597995%'  
'-----  
'----- BAYES -----'  
'AUC modela sa preprocesiranjem: 0.7244733999459898'  
'AUC modela bez preprocesiranja: 0.5723737510126924'  
'Poboljsanje modela: 26.57348431233783%'  
'-----  
'----- K NEIGHBORS -----'  
'AUC modela sa preprocesiranjem: 0.8059006211180123'  
'AUC modela bez preprocesiranja: 0.5988320280853363'  
'Poboljsanje modela: 34.57874384153143%'  
'-----
```

Slika 44. Performanse sva tri algoritma pre i nakon preprocesiranja

Sa slike se vidi da su performanse preprocesiranih podataka u sva tri slučaja bolje. Najveće poboljšanje performansi je uočeno kod algoritma K najbližih suseda. Kod ovog algoritma je klasifikator takoreći nagađao vrednosti ciljnog atributa (atribut *income*) sa performansama od 0.598. Dok su se nakon preprocesiranja podataka performanse znatno poboljšale na 0.8. Ovi rezultati ukazuju na veliki značaj koji ima preprocesiranje podataka, kao i izbor algoritma za kreiranje modela.

## 5. ZAKLJUČAK

U radu su najpre objašnjeni pojmovi data mining-a i procesa otkrivanja znanja. Detaljnije su opisane tehnike preprocesiranja podataka, kao i pojmovi nadgledano i nenadgledano učenje. Zatim su obrađena i tri algoritma za klasifikaciju. Klasifikacija predstavlja jednu od moćnijih metoda nadgledanog učenja, koja se često koristi u praksi. Na primer, model klasifikacije može se koristiti za identifikovanje podnosilaca zahteva za kredit kao niske, srednje ili visoke kreditne rizike. Različite metode klasifikacije daju različite rezultate. Ne postoji “savršen” model, kao ni model koji je superiorniji u odnosu na ostale. Klasifikacioni modeli se koriste u zavisnosti od reprezentacije podataka koji se koriste, vrste problema koji se rešava, vrste izlaznih rezultata i mnogih drugih faktora. U zavisnosti od tih faktora će modeli imati manju ili veću preciznost u radu. Stoga je dobra praksa sistematično procenjivati skup različitih algoritama kandidata i otkrivati šta dobro ili najbolje deluje na podatke. Poznato je da neki algoritmi rade lošije ako postoje ulazne promenljive koje su irelevantne. Postoje i algoritmi na koje će negativno uticati ako su dve ili više ulaznih promenljivih u visokoj korelaciji. U tim slučajevima možda treba identifikovati i ukloniti nebitne ili visoko korelirane promenljive ili koristiti alternativne algoritme.

Kao veoma važan faktor u procesu klasifikacije, se nalazi i preprocesiranje podataka. Kao što je u radu i dokazano, preprocesiranje može znatno poboljšati performanse modela i u praksi predstavlja veoma važan korak. Kolone skupa podataka mogu imati različite tipove atributa, te tako neke promenljive mogu biti numeričke, kao što su: celi brojevi, vrednosti sa pokretnom zarezom, procenti, itd. Ostale promenljive mogu biti imena, kategorije ili oznake predstavljene znakovima ili rečima, a neke mogu biti binarne, predstavljene sa 0 i 1 ili Tačno i Netačno. Problem je što algoritmi mašinskog učenja u svojoj osnovi rade na numeričkim podacima. Brojevi uzimaju kao ulaz, a broj predviđaju kao izlaz. Svi podaci se vide kao vektori i matrice, koristeći terminologiju iz linearne algebre. Kao takvi, neobrađeni podaci moraju se promeniti pre treninga, procene i upotrebe modela mašinskog učenja. Ponekad promenama podataka može interno upravljati algoritam mašinskog učenja. Čak i ako neobrađeni podaci sadrže samo brojeve, često je potrebna neka priprema podataka.

Postoji međusobna zavisnost između podataka i izbora algoritama. Algoritmi prvenstveno nameću očekivanja podacima, a poštovanje tih očekivanja zahteva da podaci budu na odgovarajući način pripremljeni. Suprotno tome, oblik podataka može pomoći u odabiru algoritama za procenu za koje postoji veća verovatnoća da će biti efikasni.

## 6. LITERATURA

- [1] [https://en.wikipedia.org/wiki/Supervised\\_learning](https://en.wikipedia.org/wiki/Supervised_learning)
- [2] <https://archive.ics.uci.edu/ml/datasets/Adult>
- [3] <https://www.ibm.com/cloud/learn/machine-learning>
- [4] [https://en.wikipedia.org/wiki/History\\_of\\_artificial\\_intelligence#The\\_birth\\_of\\_artificial\\_intelligence\\_1952%E2%80%931956](https://en.wikipedia.org/wiki/History_of_artificial_intelligence#The_birth_of_artificial_intelligence_1952%E2%80%931956)
- [5] [https://en.wikipedia.org/wiki/Timeline\\_of\\_machine\\_learning](https://en.wikipedia.org/wiki/Timeline_of_machine_learning)
- [6] [https://en.wikipedia.org/wiki/Machine\\_learning](https://en.wikipedia.org/wiki/Machine_learning)
- [7] [https://en.wikipedia.org/wiki/Dimensionality\\_reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction)
- [8] <https://www.guru99.com/unsupervised-machine-learning.html>
- [9] <https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a>
- [10] [https://en.wikipedia.org/wiki/Artificial\\_intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence)
- [11] <https://www.ibm.com/cloud/learn/supervised-learning>
- [12] <https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590>
- [13] <https://www.guru99.com/supervised-vs-unsupervised-learning.html>
- [14] [http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1\\_kdd.html#:~:text=The%20terms%20knowledge%20discovery%20and,of%20what%20qualifies%20as%20knowledge.](http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html#:~:text=The%20terms%20knowledge%20discovery%20and,of%20what%20qualifies%20as%20knowledge.)
- [15] <https://www.geeksforgeeks.org/kdd-process-in-data-mining/>
- [16] <https://www.mygreatlearning.com/blog/what-is-regression/>
- [17] <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- [18] <http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-knn.html>
- [19] <https://equipintelligence.medium.com/k-nearest-neighbor-classifier-knn-machine-learning-algorithms-ed62feb86582>
- [20] <https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a>
- [21] <https://machinelearningmastery.com/naive-bayes-for-machine-learning/>
- [22] [https://en.wikipedia.org/wiki/Maximum\\_a\\_posteriori\\_estimation](https://en.wikipedia.org/wiki/Maximum_a_posteriori_estimation)
- [23] <https://machinelearningmastery.com/data-preparation-is-important/>
- [24] <https://www.kdnuggets.com/2020/05/guide-choose-right-machine-learning-algorithm.html>
- [25] [https://www.researchgate.net/figure/Advantage-and-Disadvantage-of-Variou-Classification-Techniques\\_tbl1\\_258285203](https://www.researchgate.net/figure/Advantage-and-Disadvantage-of-Variou-Classification-Techniques_tbl1_258285203)

- [26] <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-different-classification-models/>
- [27] [https://cs.elfak.ni.ac.rs/nastava/pluginfile.php/9517/mod\\_resource/content/1/Introduction-to-Data-Mining.pdf](https://cs.elfak.ni.ac.rs/nastava/pluginfile.php/9517/mod_resource/content/1/Introduction-to-Data-Mining.pdf)
- [28] <https://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of-naive.html?fbclid=IwAR0dQRHvv-DRZDKTpBT2Ub9REXuR3wGT6BqVZq3FXP-XlBa5mYvQKbSH37A>
- [29] [https://www.researchgate.net/figure/Advantage-and-Disadvantage-of-Variou-Classification-Techniques\\_tbl1\\_258285203](https://www.researchgate.net/figure/Advantage-and-Disadvantage-of-Variou-Classification-Techniques_tbl1_258285203)
- [30] <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-different-classification-models/>