|  |  |  |
| --- | --- | --- |
| Unigrb | UNIVERZITET U NIŠU  ELEKTRONSKI FAKULTET | logo_1960_4 |

Danica Đorđević

**Upoređivanje različitih metoda klasifikacije**

**Tehnike i metode analize podataka**

Student: Danica Đorđević 1121

Niš, 2021. god.

Sadržaj

[1. UVOD 3](#_Toc719371876)

[2. DATA MINING 5](#_Toc1646803365)

[2.1. Preprocesiranje podataka 6](#_Toc24014251)

[2.1.1. Čišćenje podataka 7](#_Toc1452989120)

[2.1.2. Transformacija podataka 9](#_Toc1365420822)

[2.1.3. Redukcija podataka 10](#_Toc626529809)

[2.2. Mašinsko učenje 11](#_Toc1611695325)

[2.2.1. Nenadgledano učenje 13](#_Toc485464545)

[2.2.2. Nadgledano učenje 16](#_Toc2130558550)

[2.2.2.1. Regresija 16](#_Toc1764254749)

[2.2.2.2. Klasifikacija 18](#_Toc1368929587)

[2.3. Stabla odluke 19](#_Toc1793502717)

[2.4. Naive Bayes algoritam 19](#_Toc393121377)

[2.5. K-najbliži susedi algoritam 19](#_Toc365465154)

[2.6. Poređenje algoritama 19](#_Toc1254877415)

[3. IMPLEMENTACIJA 20](#_Toc894375302)

[4. ZAKLJUČAK 30](#_Toc779644607)

[5. LITERATURA 31](#_Toc1959796091)

# UVOD

Tehnološki razvoj je doveo do digitalizacije i automatizacije velikog broja procesa. Mnoga ljudska zanimanja su prethodnih decenija zamenile mašine, kompjuteri i različiti softveri. Kako bi mašina mogla da zameni čoveka, neophodno je da poseduje ljudske veštine i mogućnost donošenja zaključka na osnovu trenutnog stanja u okruženju. Ono što je potrebno mašinama jeste, zapravo, ljudska inteligencija. Zbog potreba automatizacije procesa, stvorila se nova grana u inženjerstvu - veštačka inteligencija. Veštačka inteligencija je vrsta inteligencije koju demonstriraju mašine, za razliku od prirodne inteligencije koju iskazuju ljudi i životinje, koja obuhvata i svest i emotivnost. Ovo znači da mašine poseduju neki nivo inteligencije i sposobne su da samostalno donose zakljuške i rešavaju probleme.

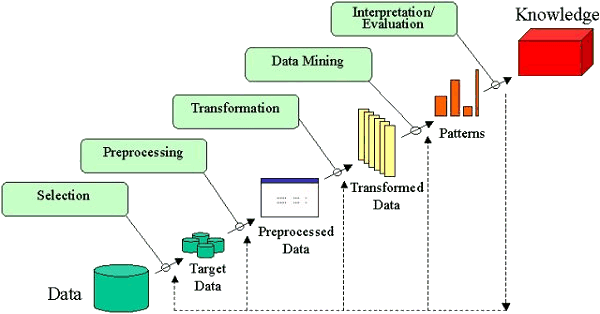
Razvoj veštačke inteligencije je počeo 50ih godina XX veka, tada je i termin “veštačka inteligencija” prvi put iskorišćen. Tada nastupa i “zlatno doba” veštačke inteligencije (1956-1974). Programi razvijani u ovom periodu su za većinu ljudi bili fascinantni. Kompjuteri su rešavali probleme iz algebre, dokazivali teoreme iz geometrije i učili da govore Engleski jezik. Nakon ovog “zlatnog” perioda nastupa prva “zima” veštačke inteligencije (1974-1980). U ovom periodu je veštačka inteligencija bila predmet mnogih kritičara, a imala je i brojne finansijske neuspehe. Istraživači koji su se bavili veštačkom inteligencijom nisu uspeli da shvate težinu problema sa kojima su se suočavali. Ogroman optimizam podigao je veoma visoka očekivanja, a kada se obećani rezultati nisu ostvarili, finansiranje veštačke inteligencije je nestalo. Uprkos poteškoćama sa javnom percepcijom umetne inteligencije krajem 70-ih, nove ideje su istraživane u logičkom programiranju, zdravom razumu i mnogim drugim oblastima. Osamdesetih godina XX veka, veštačka inteligencija kreće da se razvija ponovo. U ovom periodu (1980-1987) su nastali takozvani “ekspert” sistemi, a znanje je postalo fokus svih glavnih istraživanja. Ovi sistemi predstavljaju program koji odgovara na pitanja ili rešava probleme u vezi sa odreženim domenom znanja, koristeći logička pravila koja proizilaze iz znanja stručnjaka u tim domenima. Ekspertski sistemi ograničili su se na mali domen specifičnog znanja (čime su izbegli zdravorazumski problem znanja), a njihov jednostavan dizajn relativno je olakšao izgradnju i modifikaciju programa nakon što su postojali. Nakon ovog perioda dolazi još jedna “zima” u razvoju veštačke inteligencije (1987-1993), u kome su finansiranja za njen razvoj još jednom predstavljala problem. Ponovni razvoj, polje veštačke inteligencije je postiglo krajem XX i početkom XXI veka (1993-2011). Područje veštačke inteligencije, sada staro više od pola veka, konačno je postiglo neke od svojih najstarijih ciljeva. Počeo je da se uspešno koristi u tehnološkoj industriji. U ovom periodu kreće razvoj naprednog mašinskog učenja, koje će znatno uticati na dalju razvoj ovog polja.

U nastavku rada će biti više reči o tehnikama i metodama analize podataka (*eng. data mining*), preprocesiranju i čišćenju podataka, mašinskom učenju, podeli mašinskog učenja, kao i o klasifikaciji. Biće izvršeno i upoređivanje dva algoritma za klasifikaciju: Algoritma baziranog na stablima odlučivanja i Naiv Bajesovog algoritma.

# DATA MINING

Data mining je proces pronalaženja anomalija, obrazaca i korelacija u velikim skupovima podataka radi predviđanja ishoda. Koristeći širok spektar tehnika, mogu se koristiti ove informacije za povećanje prihoda, smanjenje troškova, poboljšanje odnosa sa kupcima, smanjenje rizika i još mnogo toga. Širok proces pronalaženja znanja u podacima i primena određenih metoda data mining-a na visokom nivou je naziva procesom otkrivanja znanja. Ovim procesom se bave istraživači mašinskog učenja, prepoznavanja obrazaca, baza podataka, statistike, veštačke inteligencije, sticanja znanja za ekspertske sisteme i vizualizacije podataka. Cilj procesa otkrivanja znanja je izdvajanje znanja iz podataka u kontekstu velikih baza podataka. To se realizuje korišćenjem metoda (algoritama) za rukovanje podacima da bi se izdvojilo (identifikovalo) ono što se smatra znanjem, u skladu sa specifikacijama mera i granicama, koristeći bazu podataka zajedno sa potrebnom transformacijom ili preprocesingom te baze podataka.

* Celokupni proces pronalaženja i tumačenja obrazaca iz podataka uključuje ponovljenu primenu sledećih koraka:
* Razumevanje domena aplikacije, ciljeva krajnjih korisnika i relevantnog predznanja,
* Stvaranje ciljnog skupa podataka (*eng. dataset*) - što uključuje odabir skupa podataka ili fokusiranje na određeni podkup promenljivih ili uzoraka podataka na kojima treba izvršiti otkrivanje,
* Preprocesiranje podataka - što uključuje uklanjanje odstupanja (*eng. outlier*) i šumova, razvijanje strategije za rukovanjem nedostajućim vrednostima polja podataka, itd.,
* Redukcija podataka i projekcija - ovaj korak uključuje pronalaženje korisnih karakteristika za predstavljanje podataka u zavisnosti od cilja zadatka, korišćenje metoda za smanjenje ili transformaciju dimenzionalnosti,
* Izbor metode data mining-a koja će se primeniti - to mogu biti regresija, klasifikacija, klasterizacija, itd.,
* Izbor data mining algoritma - što uključuje izvor metoda koje će se koristiti za traženje obrazaca u podacima, odabir prikladnih modela i parametara,
* Obučavanje - uključuje traženje obrazaca od interesa u određenom reprezentativnom obliku ili skupu takvih reprezentacija, kao što su pravila klasifikacije, regresija, klasterovanje, itd.,
* Interpretacija rezultata,
* Utvrđivanje otkrivenog znanja.



U nastavku će biti više reči o preprocesiranju podataka, kao i o data minig metodama.

# Preprocesiranje podataka

Preprocesiranje podataka je u praksi veoma važno. To je data mining tehnika koja transformiše sirove podatke u razumljivije, korisnije i efikasnije formate. Data mining vodi računa o kvalitetu podataka, dok mašinsko učenje ne uzima u obzir kvalitet podataka. Podaci koji se prikupljaju sa različitih izvora mogu biti nepouzdani, pa je neophodno da se ti problemi otklone pre primenjivanja algoritma za mašinsko učenje, u suprotnom će donošenje odluka biti takođe nepouzdano.

Preprocesiranje podataka je često neophodno jer podaci mogu biti nepotpuni, nekonzistentni, duplicirani i puni šumova. Koraci u preprocesiranju podataka su sledeći:

* **Čišćenje podataka,**
* **Transformacija podataka,**
* **Redukcija podataka.**

# Čišćenje podataka

U ovom koraku se vrši rukovanje nedostajućim podacima i šumovima. Do nedostatka vrednosti u poljima dataset-a 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 srednjom vrednošću, koja predstavlja srednju vrednos svih poznatih vrednosti u toj koloni,
* **Srednja vrednost klase** - ova metoda popunjava nedostajuća polja srednjom vrednošću, koja predstavlja srednju vrednos 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 srednom vrednošću gornjeg i dodnjeg 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, prihodi Mark Zuckerberg-a će biti daleko veći od prihoda ostalih ljudi. Da bi se šumovi iz podataka uklonili, neophodno je izvršiti izglađivanje podataka (*eng. data smoothing*). Postoje tri glavne strategije u uklanjanju šumova, a to su:

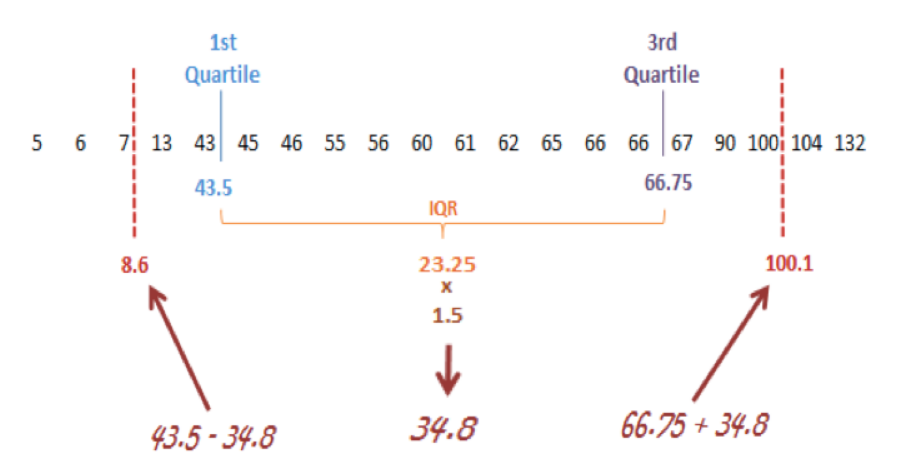
* **Bining** (*eng. binning*) - ova metoda radi nad sortiranim podacima. Svi podaci su podeljeni na segmente jednake veličine, a zatim se izvode različite metode za izglađivanje. Svaki segmentirani se obrađuje odvojeno. Svi podaci u segmentu mogu se zameniti srednjim vrednostima ili se granične vrednosti mogu koristiti za dovršavanje zadatka.
* **Regresija** - kod ove metode se podaci mogu izgladiti njihovim uklapanjem u regresionu funkciju. Korišćena regresija može biti linearna (koja ima jednu nezavisnu promenljivu) ili višestruka (koja ima više nezavisnih promenljivih).
* **Klasterizacija** - ova metoda grupiše slične podatke u klastere. Odstupanja mogu ostati neotkrivena ili se mogu detektovati ako ispadnu iz granica klastera.

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.

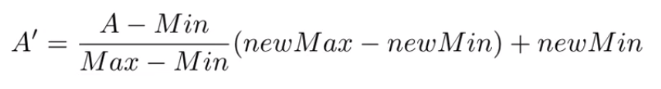
Gde Q1 predstavlja prvi kvartil, a Q3 treći kvartil. Cela metoda pronalaženja odstupanja se zasniva na kvartilima podataka. Prvi kvartil Q1 je vrednost ≥ 1/4 podataka, drugi kvartil ili medijana je vrednost ≥ 1/2 podataka, a treći kvartil K3 je vrednost ≥ 3/4 podataka. Interkvartilni opseg (IQR) predstavlja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



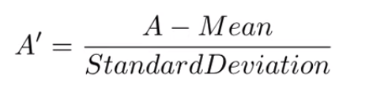
# Transformacija podataka

Ovaj korak se preduzima da bi se podaci transformisali u odgovarajuće forme, pogodne za data mining. Transformacija podataka može uključivate sledeće metode:

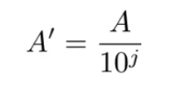
* **Normalizacija** - radi se nad numeričkim tipovima podataka, radi skaliranja vrednosti podataka u određenom opsegu (najčešće u opsegu od 0 do 1 ili od -1 do 1). Najčešće se koristi *Min-Max* tehnika, koja koristi formulu na slici, gde A’ predstavlja normalizovanu vrednost, a A originalnu vrednost,



* **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 prikazanoj na slici, gde A’ predstavlja normalizovanu vrednost, a A originalnu vrednost,



* **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, gde A’ predstavlja normalizovanu vrednost, a A originalnu vrednost,



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

# 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 smanjenja podataka. Cilj redukcije podataka je povećati efikasnost skladištenja i smanjiti troškove skladištenja i analize podataka. Neki od koraka u redukciji podataka su:

**Agregacija podataka** - ova tehnika podrazumeva kombinovanje dva ili više atributa (ili objekata) u jedan atribut (objekat). Podaci se smanjuju primenom OLAP operacija poput *slice*, *dice*, *rollup*. Koristi najmanji nivo neophodan za rešavanje problema,

**Redukcija dimenzionalnosti** - u ovoj metodi se atributi ili dimenzije podataka smanjuju. Nisu svi atributi potrebni za data mining. Najprikladniji podskup atributa bira se tehnikama poput PCA (Principal Componenet Analysis) i Wavelet-ove transformacije,

**Brojno smanjenje** (eng. Numerosity reduction) - ova tehnika smanjuje obim podataka odabirom manjih obrazaca za reprezentaciju podataka. Brojno smanjenje se može izvršiti pomoću histograma, klasterovanja ili semplovanja podataka. Brojno smanjenje je ponekad neophodno jer je obrada celokupnog skupa podataka skupa i dugotrajna.

**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.

# Mašinsko učenje

Mašinsko učenje predstavlja deo veštačke inteligencije i proučava računarske algoritme, koji se progresivno poboljšavaju kroz iskustvo. Algoritmi mašinskog učenja grade model zasnovan na uzroku podataka, poznatom kao trening podaci, kako bi radio predikciju ili donosio odluke, a da za to nije izričito programiran. Algoritmi mašinskog učenja koriste se u širokom spektru aplikacija, poput filtriranja e-pošte i računarskog vida, gde je teško ili neizvodljivo razviti konvencionalne algoritme za izvršavanje potrebnih zadataka. Podskup mašinskog učenja usko je povezan sa računarskom statistikom koja se fokusira na predviđanje pomoću računara. Ali nije svo mašinsko učenje statističko učenje. Studija matematičke optimizacije obezbeđuje metode, teoriju i domene primene u polje mašinskog učenja. Mining podataka je srodno područje proučavanja, fokusirajući se na istraživačku analizu podataka kroz nenadgledano učenje. U svojoj primeni na poslovne probleme, mašinsko učenje se naziva i prediktivnom analitikom.

Mašinsko učenje obuhvata računare koji otkrivaju kako mogu da izvršavaju zadatke, a da za to nisu specifično programirani. Uključuje računare koji uče na osnovu obezbeđenih podataka, kako bi kasnije izvršavali određene zadatke. Za jednostavne zadatke dodeljene računarima, moguće je programirati algoritme koji mašini govore kako da izvrši sve korake potrebne za rešavanje datog problema, to znači da mašini nije potrebno učenje jer ima sve moguće korake koji joj pomažu da reši problem. Napredniji zadaci mogu predstavljati izazov za čoveka, jer on ručno kreira potrebne algoritme. U praksi se može ispostaviti efikasnije pomoći mašini da razvije sopstveni algoritam, umesto da ljudski programeri preciziraju svaki potreban korak. Disciplina mašinskog učenja koristi različite pristupe za podučavanje računara da izvršavaju zadatke tamo gde nije dostupan potpuno zadovoljavajući algoritam. U slučajevima kada postoji ogroman broj potencijalnih odgovora, jedan od pristupa je označavanje nekih tačnih odgovora kao validnih. To se zatim može koristiti kao podaci za obuku računara za poboljšanje algoritma koji koristi za utvrđivanje tačnih odgovora. Postoje različiti pristupi u mašinskom učenju, koji se dele u tri kategorije:

* **Nenadgledano učenje** - Algoritmu učenja nisu dostupne vrednosti željenih izlaza, a cilj je otkrivanje skrivenih obrazaca u podacima i učenje karakterisika.
* **Nadgledano učenje** - Računaru su dostupni ulazi i željeni izlazi, a cilj je naučiti opšte pravilo koje mapira ulaze na izlaze.
* **Pojačano učenje** - Računarski program komunicira sa dinamičkim okruženjem u kojem mora da postigne određeni cilj (npr. vožnja vozilom ili igranje igre protiv protivnika). Dok se kreće kroz problem, programu se daju povratne informacije koje su analogne nagradama, koje pokušava da maksimizira.

Mašinsko učenje i analiza podataka se koriste da bi iz neke količine podataka izvukle znanje. Izdvajanje znanja je proces prikupljanja informacija iz strukturiranih i nestrukturiranih izvora kako bi se stvorila baza znanja za identifikovanje smislenih i korisnih obrazaca iz velikih i semantički nejasnih (*eng. fuzzy*) skupova podataka. Nejasni skupovi podataka su skupovi čiji elementi imaju određeni nivo članstva. Stepen članstva definisan je funkcijom članstva koja se vrednuje između 0 i 1. Izdvojeno znanje se ponovo koristi, zajedno sa izvornim podacima, da bi se stvorilo nabrajanje obrazaca koji se vraćaju u bazu znanja. Proces otkrivanja znanja uključuje programsko istraživanje velike količine podataka za obrasce koji se mogu nabrojati kao znanje. Stečeno znanje je predstavljeno kao više modela prema kojima se mogu postaviti određena ispitivanja, po potrebi. Otkrivanje znanja spaja koncepte računarske nauke i mašinskog učenja (kao što su baze podataka i algoritmi) sa statistikama radi rešavanja korisnički orijentisanih upita i problema. Znanje se može opisati u različitim oblicima, kao što su klase učesnika, modeli pridruživanja atributa i zavisnosti. Otkrivanje znanja u velikim podacima koristi osnovne mašinske algoritme koji su dizajnirani za klasifikaciju, klasterizaciju, smanjenje dimenzionalnosti i zajedničko filtriranje, kao i skalabilne distribuirane sisteme.

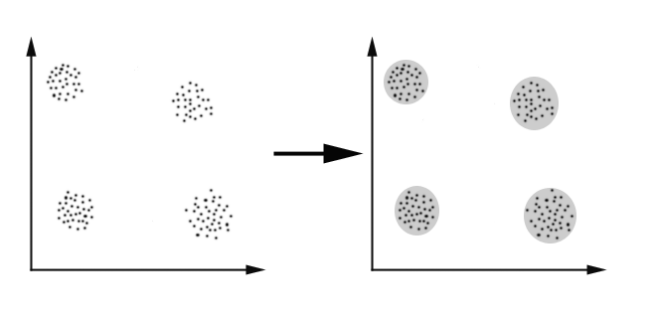
U nastavku će biti reči o nenadgledanom i nadgledanom učenju.

# Nenadgledano učenje

Nenadgledano učenje je vrsta algoritma koja uči obrasce iz neobeleženih podataka. U nekim problemima sa prepoznavanjem obrazaca, trening podaci se sastoje od skupa ulaznih vektora *x* bez odgovarajućih ciljnih vrednosti. Cilj takvih nenadgledanih problema učenja može biti otkrivanje grupa sličnih primera unutar podataka, gde se to naziva grupisanjem, ili utvrđivanje načina na koji se podaci distribuiraju u prostoru, poznatom kao procena gustine. Jednostavnije rečeno, za n-uzorkovani prostor *x1* do *xn*, prave oznake klasa nisu predviđene za svaki uzorak, što je poznato kao učenje bez “nastavnika”. Nenadgledano učenje je teže u poređenju sa zadacima nadgledanog učenja. Kod ovog oblika učenja je teže proceniti da li su rezultati od značaja, s obzirom da nisu dostupne realne vrednosti rešenja tj. podaci su neoznačeni. Za procenu rezultata neophodno je da stručnjak pogleda i analizira rezultate, a to se naziva eksterno vrednovanje. Pored eksternog vrednovanja, neophodno je definisati i ciljnu funkciju klasterovanja tj. uraditi internu evaluaciju. Iako je nenadgledano učenje teže, ono ima i svoje prednosti. Obeležavanje velikih datasetova je vrlo skupo, a ručno obeležavanje može biti dugotrajan proces. Stoga je lakše raditi sa neoznačenim podacima, kao što je to slučaj prilikom prepoznavanja govora. Postoje i slučajevi kada ne znamo na koliko klasa su podaci podeljeni, što je slučaj kod mining-a podataka. Takođe, možda ima potrebe za korišćenjem klasterizacije da bi se stekao određeni uvid u strukturu podataka pre dizajniranja klasifikatora. Nenadgledano učenje se može podeliti u dve kategorije:

* **Parametarsko nenadgledano učenje** - U ovom slučaju pretpostavljamo parametarsku distribuciju podataka. Pretpostavlja se da podaci uzorka potiču iz populacije koja sledi raspodelu verovatnoće na osnovu fiksnog skupa parametara. Teoretski, u normalnoj porodici distribucija, svi članovi imaju isti oblik i parametrišu se srednjom i standardnom devijacijom. To znači da ako znate srednju i standardnu devijaciju i da je distribucija normalna, znate verovatnoću bilo kog budućeg posmatranja. Parametarsko nenadzirano učenje uključuje izgradnju Gaussovih modela mešavine i korišćenje algoritma očekivanja-maksimizacije za predviđanje klase predmetnog uzorka. Ovaj slučaj je mnogo teži od standardnog učenja pod nadzorom, jer ne postoje oznake odgovora, pa stoga ne postoji tačna mera tačnosti za proveru rezultata.
* Neparametarsko nenadgledano učenje - U neparametrizovanoj verziji učenja bez nadzora, podaci su grupisani u klastere, gde svaki klaster (nadamo se) govori nešto o kategorijama i klasama prisutnim u podacima. Ova metoda se obično koristi za modeliranje i analizu podataka sa malim veličinama uzoraka. Za razliku od parametarskih modela, neparametrijski modeli ne zahtevaju od modelara bilo kakve pretpostavke o raspodeli populacije, pa se tako ponekad nazivaju metodom bez raspodele.

Klasterovanje se može smatrati najvažnijim problemom nenadglednaog učenja. Kao i svaki drugi problem ove vrste, i ovaj problem se bavi pronalaženjem strukture u kolekciji neobeleženih podataka. Oskudna definicija klasterizacije može biti proces organizovanja objekata u klastere čiji su članovi na neki način slični. Klaster je stoga kolekcija objekata koji su slični među sobom i koji se razlikuju od objekata koji pripadaju drugim klasterima.



Skup tačaka koje su na raspolaganju sa pojmom rastojanja između tačaka se grupišu u određeni broj klastera, tako da:

* **interno rastojanje** tačaka unutar klastera bude malo, što znači da su objekti unutar klastera međusobno slični,
* **eksterno rastojanje** tačaka između klastera treba da bude veliko, što znači da su objekti unutar tih klastera međusobno različiti.

Cilj klasterizacije je odrediti interno grupisanje u skupu neobeleženih podataka. Ali kako odlučiti šta predstavlja dobro grupisanje? Može se pokazati da ne postoji apsolutni „najbolji“ kriterijum koji bi bio nezavisan o konačnom cilju klasterisanja. Shodno tome, korisnik je taj koji treba da pruži ovaj kriterijum na takav način da rezultat klasterisanja odgovara njihovim potrebama.

Nadledano i nenadgledano učenje se često zajedno spominju zajedno. Za razliku od učenja pod nadzorom, učenje bez nadzora koristi neobeležene podatke. Iz tih podataka otkriva obrasce koji pomažu u rešavanju problema klastera ili udruživanja. Ovo je posebno korisno kada stručnjaci za predmet nisu sigurni u uobičajena svojstva u skupu podataka. Uobičajeni algoritmi klasterovanja su hijerarhijski, k-srednji i Gaussovi modeli smeša.

Do polunadgledanog učenja dolazi kada je označen samo deo zadatih ulaznih podataka. Nenadgledano i polunadgledano učenje mogu biti privlačnije alternative, jer može biti dugotrajno i skupo osloniti se na stručnost domena za označavanje podataka na odgovarajući način za nadzirano učenje. U nastavku će biti više reči o nadgledanom učenju.

# Nadgledano učenje

Nadgledano učenje je najčešće korišćena podgrana mašinskog učenja danas. Tipično, novi praktičari mašinskog učenja započinju svoje putovanje nadgledanim algoritmima učenja.

Nadgledani algoritmi mašinskog učenja dizajnirani su za učenje na primerima. Naziv „nadgledano“ učenje potiče od ideje da je obuka ove vrste algoritma kao da nastavnik nadgleda čitav proces.

Kada se trenira algoritam nenadgledanog učenja, podaci o obuci sastojaće se od ulaza uparenih sa tačnim izlazima. Tokom treninga, algoritam će tražiti obrasce u podacima koji su u korelaciji sa željenim rezultatima. Nakon treninga, nadgledani algoritam učenja unosiće nove neviđene ulaze i određivaće na osnovu kojih podataka će biti označeni novi ulazi na osnovu podataka iz prethodne obuke. Cilj nadziranog modela učenja je predvideti tačnu oznaku za novo predstavljene ulazne podatke. U svom najosnovnijem obliku, nadgledani algoritam učenja može se napisati jednostavno kao:



Gde je *Y* predviđeni izlaz koji je određen funkcijom mapiranja koja dodeljuje klasu ulaznoj vrednosti *x*. Funkciju koja se koristi za povezivanje ulaznih karakteristika sa predviđenim izlazom stvara model mašinskog učenja tokom treninga. Nadgledano učenje se može podeliti u dve potkategorije:

* klasifikacija i
* regresija.

# Regresija

Regresija je prediktivni statistički proces gde model pokušava da pronađe važnu vezu između zavisnih i nezavisnih promenljivih. Cilj algoritma regresije je predvideti kontinuirani broj kao što su prodaja, prihod i rezultati testova. Jednačina za osnovnu linearnu regresiju može se napisati na sledeći način:

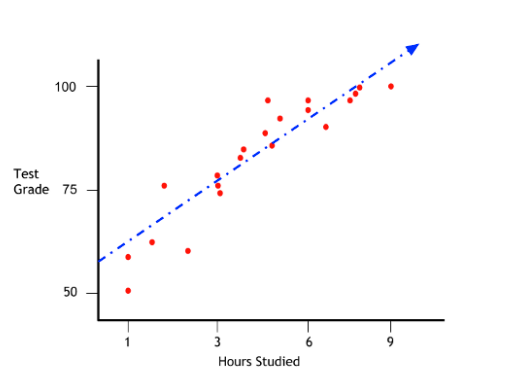


Gde je x[i] karakteristika(x) za podatke, a gde su w[i] i b parametri koji se razvijaju tokom treninga. Za jednostavne modele linearne regresije sa samo jednom osobinom u podacima, formula izgleda ovako:



Gde je w nagib, x je pojedinačna karakteristika, a b je presek na y osi. Ova jednačina zapravo predstavlja jednačinu prave. Za jednostavne regresione probleme kao što je ovaj, predviđanja modela predstavljena su linijom koja najbolje odgovara. Za modele koji koriste dve karakteristike koristiće se ravan. A za model koji koristi više od dve osobine koristiće se hiperravan.

Recimo da je zadatak predvideti ocenu studenta na testu u zavisnosti od toga koliko je sati učio pred taj test. Pretpostavimo da nacrtani podaci sa linijom koja najbolje odgovara izgledaju ovako:



Postoji jasna pozitivna korelacija između količine sati koje je student proveo učeći za test (nezavisna promenljiva) i konačnog rezultata testa studenta (zavisna promenljiva). Kroz tačke podataka može se povući linija koja najbolje odgovara, kako bi se prikazala predviđanja modela kada se dobije novi ulaz. Recimo da treba da se sazna koliko dobro će student proći na testu, ako je za njega učio pet sati. Može se koristiti linija koja najbolje odgovara za predviđanje rezultata testa na osnovu performansi drugih učenika. Sa slike se može videti da student sa 5 sati učenja može da očekuje da na testu osvoji 80 poena. Postoji mnogo različitih vrsta algoritama regresije. Tri najčešća su:

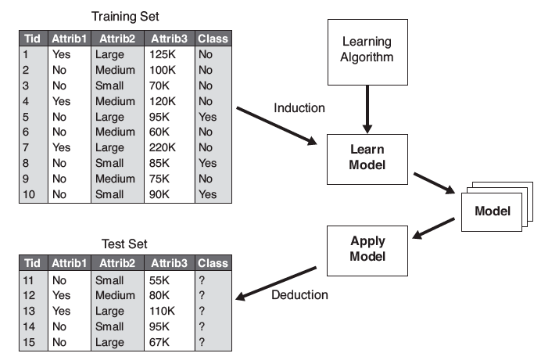
* Linearna regresija
* Logistička regresija
* Polinomska regresija

# 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 na dat na sledećoj slici:



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 je prikazan 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 kao neželjena pošta, a ne neželjena 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 neželjenoj pošti moraju se koristiti kao podaci o obuci. Kada se klasifikator pravilno obuči, može se koristiti za otkrivanje nepoznate e-pošte.

Klasifikacija spada u kategoriju nadzora pod nadzorom gde su ciljevi takođe dati sa ulaznim podacima. Mnogo je primena u klasifikaciji u mnogim domenima, poput odobrenja kredita, medicinske dijagnoze, ciljnog marketinga itd.

Postoje dve vrste učenika koji se svrstavaju u:

* **Lenje učesnike** - oni jednostavno 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 podacima o obuci. U poređenju sa željnim učenicima, lenji učenici imaju manje vremena za obuku, ali više vremena za predviđanje. Primeri: K-najbliži sused, rezonovanje zasnovano na slučaju.
* **Željne učesnike** - oni konstruišu model klasifikacije na osnovu datih podataka o obuci pre nego što dobiju podatke za klasifikaciju. Mora biti u stanju da se posveti jednoj hipotezi koja pokriva čitav prostor instance. Zbog konstrukcije modela, željnim učenicima treba puno vremena za treniranje i manje vremena za predviđanje. Primeri: stablo odlučivanja, Naive Bayes, veštačke neuronske mreže

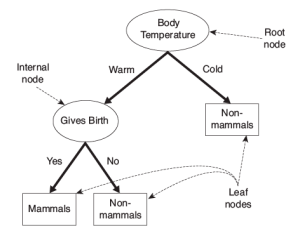
Danas je dostupno mnogo algoritama za klasifikaciju, ali nije moguće zaključiti koji je superiorniji od drugog. U nastavku će biti obrađena tri algoritma za klasifikaciju: algoritam baziran na stablima odluke, algoritam K najbližih suseda i Naiv Bajesov algoritam.

# 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 hijararhiju i sastoji se od tipa čvora:

* **Koren** - čvorovi koji ne poseduju ulazne grane, a imaju više ili nijednu izlaznu granu,
* **Interni čvorovi** - čvorovi koji ima 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 je prikazano stablo odluke prilikom klasifikacije 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 odluka. Ovi algoritmi najčešće koriste pohlepnu (*eng. greedy*) strategiju za odabir atributa na kome će da se vrši podela. Jedan takav algoritam se zove Hantov algoritam, koji prestavlja 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 Dt skup trening podataka, koji su povezani sa čvorom t i neka je y={y1, y2, ..., yc}skup klasnih atributa. U nastavku je data rekurzivna definicija Hantovog algoritma:

**Korak 1**: Ako svi objekti u skupu Dt pripadaju istoj klasi yt, onda je čvor t list, označen sa yt klasom.

**Korak 2**: Ako skup Dt sadrži objekte koji pripadaju više klasama, testirajući uslov atributa je selektovan radi particionisanja objekta na manje delove. Čvor potomak (*eng. child node*) se kreira za svaki ishod testirajućeg uslova i objekati u skupu Dt se distibuiraju 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:

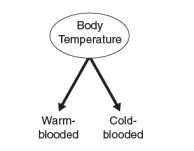
* Moguće je da su neki od čvorova potomaka, kreiranih u koraku 2, prazni tj. ne postoje objekti povezani sa ovim čvorovima. To se može dogoditi ako nijedan objekat u trening skupu nema kombinaciju vrednosti atributa povezanih sa takvim čvorovima. U ovom slučaju, čvor se proglašava čvorom lista sa istom oznakom klase kao i većina klasa koje imaju objekti trening skupa povezani sa njegovim roditeljskim čvorom.
* U koraku 2, ako svi objekti povezani sa Dt 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:

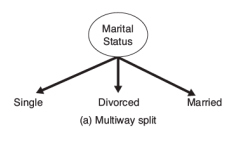
**Kako bi trebalo podeliti trening skup?** - svaki rekurzivni korak u procesu rasta stabla odluke mora odabrati uslov ispitivanja atributa da bi se zapisi podelili na manje podskupove. Da bi primenio ovaj korak, algoritam mora da obezbedi metod za specifikaciju uslova testa za različite tipove atributa, 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.

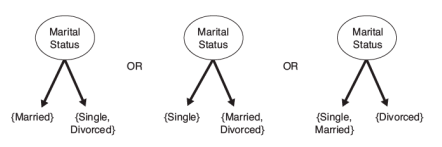
Algoritmi indukcije stabla odluka moraju da obezbede metodu za izražavanje uslova ispitivanja atributa i njegovih odgovarajućih ishoda za različite tipove atributa. Uslov testa za binarni atribut generiše dva potencijalna ishoda, kao što je prikazano na slici:



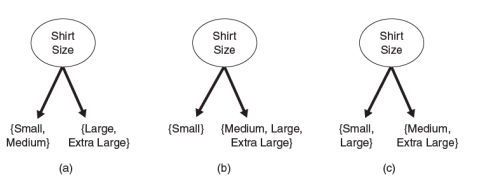
S obzirom da nominalni atribut može imati mnogo vrednosti, njegovo probno stanje može se izraziti na dva načina. Za višestruko razdvajanje, broj ishoda zavisi od broja različitih vrednosti za odgovarajući atribut. Na primer, ako atribut kao što je bračni status ima tri različite vrednosti: samac, oženjen ili razveden - njegovo testno stanje će proizvesti trosmernu podelu. Ova podela je prikazana na sledećoj slici:



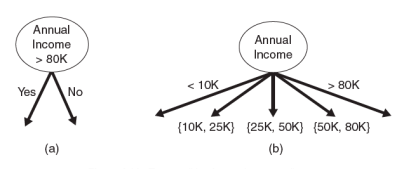
S druge strane, neki algoritmi stabla odlučivanja, poput CART, proizvode samo binarne podele razmatrajući sve 2k-1 - 1 načina stvaranja binarne particije vrednosti k atributa. Slika ispod ilustruje tri različita načina grupisanja vrednosti atributa za bračno stanje u dva podskupa.



Ordinalni atributi takođe mogu proizvesti binarne ili višestruke podele. Redovne vrednosti atributa mogu se grupisati sve dok grupisanje ne krši svojstvo redosleda vrednosti atributa. Slika 4.10 ilustruje različite načine razdvajanja zapisa o treningu na osnovu atributa veličina košulje. Grupisanja prikazana na slikama 4.10 (a) i (b) čuvaju redosled vrednosti atributa, dok grupisanje prikazano na slici 4.10 (c) krši ovo svojstvo jer kombinuje vrednosti atributa Small i Large u istu particiju dok je Medium i Extra Large se kombinuju u drugu particiju.



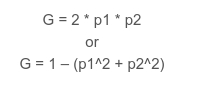
Za kontinuirane atribute, uslov testa može se izraziti kao test upoređivanja (A < *υ*) ili (A ≥ *υ*) sa binarnim ishodima ili upit opsega sa ishodima oblika *υi* ≤ A < *υi+1*, za *i* = 1 , ..., *k*. Razlika između ovih pristupa prikazana je na slici 4.11. Za binarni slučaj, algoritam stabla odlučivanja mora uzeti u obzir sve moguće podeljene položaje v i bira onaj koji daje najbolju particiju. Za višestruko razdvajanje, algoritam mora uzeti u obzir sve moguće opsege kontinuiranih vrednosti. Jedan od pristupa je primena strategija diskretizacije opisanih u odeljku 2.3.6 na stranici 57. Nakon diskretizacije, svakom rednom intervalu će se dodeliti nova redna vrednost. Susedni intervali se takođe mogu agregirati u šire opsege sve dok je svojstvo naloga sačuvano.



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 poređane i različite tačke podele se isprobavaju i testiraju pomoću funkcije troškova. Bira se podela sa najboljim troškovima (najniži trošak jer troškove minimiziramo). 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“ (koliko su mešani trening podaci dodeljeni svakom čvoru). Gini indeks se definiše kao:



Gde je G Ginijev indeks za sve klase, pk je procenat primera treninga 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:



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:



Gde je Ginijev indeks za tačku razdvajanja, g1\_1 je procenat primeraka u grupi 1 za klasu 1, g1\_2 za klasu 2, g2\_1 za grupu 2 i klasu 1, g2\_2 grupa 2 klase 2, ng1 i ng2 su ukupan broj slučajeva u grupi 1 i 2 i n su 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 podacima o treningu. Najčešći postupak zaustavljanja je korišćenje minimalnog broja broja primera obuke dodeljenih svakom čvoru lista. Ako je brojanje manje od nekog minimuma, razdvajanje se ne prihvata i čvor se uzima kao završni čvor lista. Broj članova treninga prilagođen je skupu podataka, npr. 5 ili 10. Definiše koliko će drvo biti specifično za podatke o obuci. Previše specifično (npr. Brojanje 1) i stablo će prekomerno opremiti podatke o treningu i verovatno će imati loše performanse na skupu testova.

Kriterijum zaustavljanja je važan jer snažno utiče na performanse vašeg drveta. 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. Rezidba smanjuje složenost konačnog klasifikatora, a time i poboljšava tačnost predviđanja smanjenjem prekomerne opreme. Složenost stabla odluke definiše se kao broj cepanja u stablu. Poželjnija su jednostavnija stabla. Lako ih je razumeti (možete ih odštampati i pokazati stručnjacima za predmet), a manja je verovatnoća da će prekomerno opremiti vaše podatke. Najbrža i najjednostavnija metoda obrezivanja je proći kroz svaki čvor lista na drvetu i procijeniti efekat uklanjanja pomoću skupa testova za zadržavanje. Čvorovi lista uklanjaju se samo ako to rezultira padom funkcije ukupnih troškova na celom skupu testova. Prestajete da uklanjate čvorove kada ne mogu da se naprave dodatna poboljšanja. Mogu se koristiti sofisticiranije metode obrezivanja, poput obrezivanja složenosti troškova (takođe se naziva obrezivanje najslabije veze), gde se parametar učenja (alfa) koristi za vaganje 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 cešto je vreme za obuku modela duže.

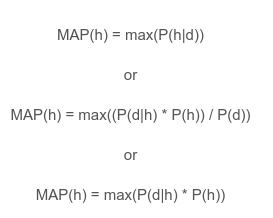
# Naive Bayes algoritam

Naive Bayes je jednostavan, ali iznenađujuće moćan algoritam za prediktivno modeliranje. U mašinskom učenju često smo zainteresovani za odabir najbolje hipoteze (h) datih podataka (d). U klasifikacionom problemu, naša hipoteza (h) može biti klasa koju treba dodeliti za novu instancu podataka (d). Jedan od najjednostavnijih načina izbora najverovatnije hipoteze s obzirom na podatke koje imamo i koje možemo koristiti kao prethodno znanje o problemu. Baiesova teorema pruža način na koji možemo izračunati verovatnoću hipoteze s obzirom na naše prethodno znanje. Bajesova teorema navodi se kao:



Gde je P (h | d) verovatnoća hipoteze h s obzirom na podatke d. To se naziva zadnja 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).

Možete videti da nas zanima izračunavanje zadnje verovatnoće P (h | d) iz prethodne verovatnoće p (h) sa P (D) i P (d | h). Nakon izračunavanja posteriorne verovatnoće za niz različitih hipoteza, možete odabrati hipotezu sa najvećom verovatnoćom. Ovo je maksimalno verovatna hipoteza i formalno se može nazvati maksimumom a posteriori (MAP) hipoteza. Procena maksimalne posteriori verovatnoće (MAP) je procena nepoznate veličine koja je jednaka načinu zadnje raspodele. MAP se može koristiti za dobijanje bodovne procene neopažene veličine na osnovu empirijskih podataka. Usko je povezan sa metodom procene maksimalne verovatnoće (ML), 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:



P (d) je normalizujući pojam koji nam omogućava izračunavanje verovatnoće. Možemo je odustati kada nas zanima najverovatnija hipoteza, jer je konstantna i koristi se samo za normalizaciju. Da se ​​vratimo na klasifikaciju, ako u podacima o treningu imamo paran broj primeraka u svakoj klasi, verovatnoća svake klase (npr. P (h)) biće jednaka. Ponovo, ovo bi bio konstantan pojam u našoj jednačini i mogli bismo ga ispustiti tako da završimo sa:



Naive Baies je algoritam klasifikacije za binarne (dvorazredne) i višerazredne probleme klasifikacije. Tehniku je najlakše razumeti kada se opisuje pomoću binarnih ili kategoričkih ulaznih vrednosti. Nazvan je naivnim Baies-om ili idiotskim Baies-om, jer je proračun verovatnoće za svaku hipotezu pojednostavljen kako bi njihov proračun bio izvediv. Umesto da pokušavaju da izračunaju vrednosti svake vrednosti atributa P (d1, d2, d3 | h), pretpostavlja se da su uslovno nezavisne s obzirom na ciljnu vrednost i izračunate kao P (d1 | h) \* P (d2 | H) i uskoro. Ovo je vrlo snažna pretpostavka koja je malo verovatna u stvarnim podacima, tj. Da atributi ne međusobno deluju. Ipak, pristup iznenađujuće dobro radi na podacima za koje ova pretpostavka ne važi.

Reprezentacija za Naivnog Baiesa je verovatnoća. Lista verovatnoća se čuva u evidenciji za naučeni naivni Baiesov model. Ovo uključuje:

* Verovatnoća klase: Verovatnoće svake klase u skupu podataka o obuci.
* Uslovne verovatnoće: Uslovne verovatnoće svake ulazne vrednosti date za svaku vrednost klase.

Učenje naivnog Baies-ovog modela iz vaših podataka o treningu je brzo.

Trening je brz jer treba izračunati samo verovatnoću svake klase i verovatnoću svake klase s obzirom na različite ulazne (k) 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) = brojanje (klasa = 1) / (brojanje (klasa = 0) + brojanje (klasa = 1))

U najjednostavnijem slučaju svaka klasa bi imala verovatnoću od 0,5 ili 50% za binarni problem klasifikacije sa istim brojem slučajeva 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) = brojanje (slučajevi sa vremenom = sunčano i klasa = izlazak) / brojanje (slučajevi sa klasom = izlazak)

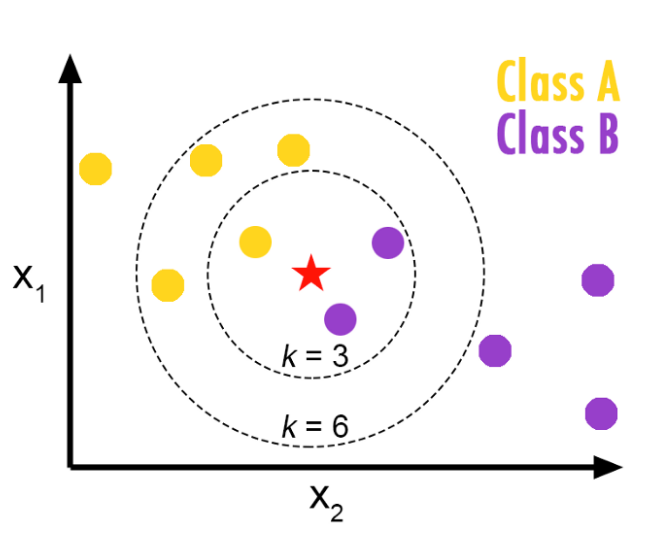
P (vreme = sunčano | klasa = boravak kod kuće) = brojanje (slučajevi sa vremenom = sunčano i klasa = boravak kod kuće) / brojanje (slučajevi sa nastavom = boravak kod kuće)

P (vreme = kiša | klasa = izlazak) = brojanje (slučajevi sa vremenom = kiša i klasa = izlazak) / brojanje (slučajevi sa klasom = izlazak)

P (vreme = kiša | klasa = boravak kod kuće) = brojanje (slučajevi sa vremenom = kiša i klasa = boravak kod kuće) / brojanje (slučajevi sa nastavom = boravak kod kuće)

# K-najbliži susedi algoritam

Algoritam K-najbližih 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. Euklidska ili Manhattanska itd. Zatim se bira K najbližih tačka, gde K može biti bilo koji ceo broj. Na kraju, 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.



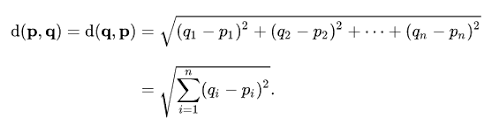
Koraci u KNN algoritmu su sledeći:

* Učitati podatke
* Inicijalizovati K
* Za svaku tačku u podacima odraditi:
* Izvračunati 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 tačku podataka sa testom čija je vrednost (2, 1, 3). Naša test tačka okružena je žutim i plavim tačkama koje predstavljaju naše 2 klase. Sada saznajemo udaljenost od naše test tačke do svake tačke na grafikonu. Pošto ima 10 tačaka, dobijamo 10 rastojanja. Utvrđujemo najnižu udaljenost i predviđamo da pripada istoj klasi najbližeg suseda. Ako je žuta tačka najbliža, onda predviđamo da je tačka podataka sa testom takođe žuta tačka. Ali, u nekim slučajevima možete dobiti i dve razdaljine koje su tačno jednake. Ovde uzimamo u obzir treću tačku podataka i izračunavamo njezinu udaljenost od naših test podataka. Na gornjem dijagramu, naši podaci o ispitivanju nalaze se između žute i plave tačke. Razmotrili smo udaljenost od treće tačke podataka i predvideli da su naši test podaci klase PLAVI.

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 koristimo jednostavnu formulu Euklidove distance.



Gornja formula uzima n broj dimenzija ili ih ovde možemo reći kao naše osobine u mašinskom učenju. 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.

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 udajenosti velike i osetljiv je na odstupanja (*eng. outlier*) i nedostajuće podatke.

# Poređenje algoritama

# IMPLEMENTACIJA

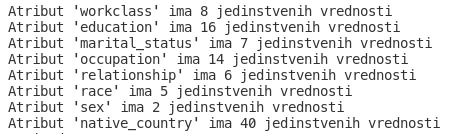
U ovom radu je odrađeno preprocesiranje podataka, kao i upoređivanje tri algoritama za klasifikaciju. Algoritmi koji su korišćeni za klasifikaciju su:

* Naive Bayes 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. Dataset ima 15 atributa, uključujući i atribut za koji se vrši predikcija. Ovaj dataset inicijalno sadrži 48,842 redova, ali za potrebe ovog projekta je izdvojeno 6204 reda. Svaki red sadrži sledeće atribute:

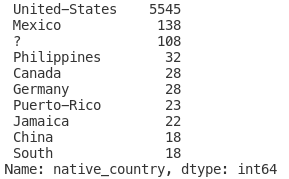
* **age** - atribut kazuje koliko godina ima pojedinac,
* **workclass** - atribut predstavlja u kom radnom odnosu je pojedinac. Ovaj atribut ima 8 jedinstvenih vrednosti,
* **fnlwgt** - atribut predstavlja finalnu težinu, koju cenzus određuje,
* **education** - atribut predstavlja nivo obrazovanja koji 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 jedinstvenig vrednosti,
* **income** - labela koja govori da li osoba zarađuje manje ili više od 50 hiljada godišnje.

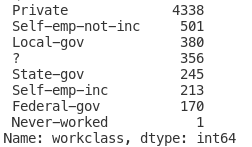


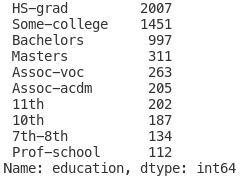
Prvo je izvršeno učitanje dataseta korišćenjem *pandas* biblioteke. Ova biblioteka učitane fajlove predstavlja u obliku DataFrame-a. DataFrame predstavlja tabelarnu reprezentaciju podataka. Podaci imaju dve dimenzije: redove i kolone. Zatim je izvršena podela tog DataFrame-a na dva dela:

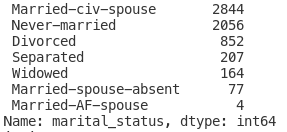
* ***X*** - predstavlja redove sa svim karakteristikama, bez kolone za koju se vrši predikcija tj. bez *income* kolone,
* ***y*** - predstavlja kolonu za koju se vrši predikcija i sadrži vrednosti kolone *income*.

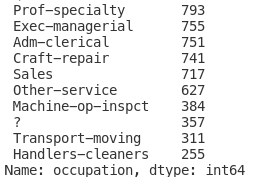
Za dalju obradu podataka, neophodno je da se uoče distribucije vrednosti kako bi se videlo koji atributi se mogu potencijalno modifikovati. Distribucije kategoričkih atributa su date na sledećim slikama:

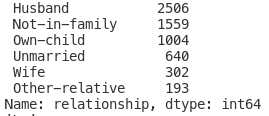


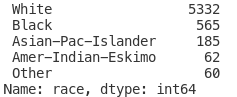








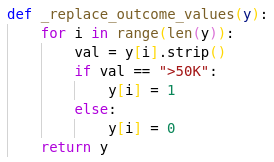






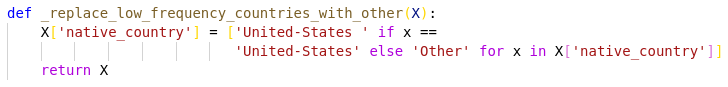


Pošto atribut income ima nenumeričke vrednosti, neophodno je te vrednosti prevesti u numeričke tipa int. Ovo je urađeno u funkciji prikazanoj na sledećoj slici:

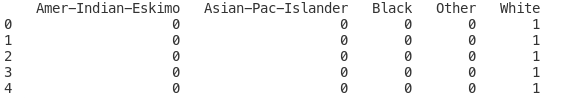


Ova funkcija prolazi kroz niz *y* i vrši zamenu vrednosti “>50K” jedinicom, a vrednosti “<=50K” nulom.

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 sa 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:

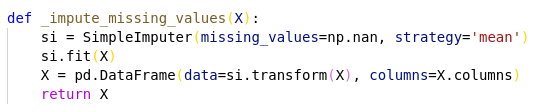


Ovaj dataset poseduje veliki broj kategoričkih atributa tj. atributa koji nemaju numeričku vrednost. Pošto modlei rade samo sa numeričim vrednostima, neophodno je ove atribute prevesti iz nenumeričke u numeričke oblike. Biblioteka *pandas* nudi rešenje za ovaj problem, korišćenjem *dummies* atributa. 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 sledećoj slici:

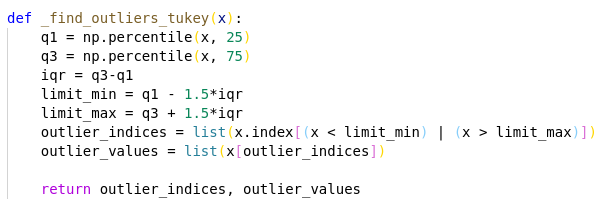


U ovom primeru se mogu videti 5 novih kolona, gde vrednost 1 u koloni označava da pojedinac pripada toj rasi, a vrednost 0 označava da pojedinac ne pripada toj rasi.

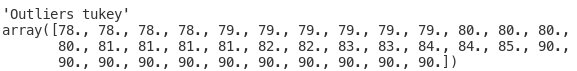
Korišćeni dataset ima nedostajućih vrednosti, pa je neophodno popuniti ta polja određenim vrednostima, u zavisnosti od korišćene strategije. Za popunjavanje nedostajućih vrednosti je korišćena strategija srednje vrednosti, što znači da su se nedostajuće vrednosti u kolonama popunjavale srednjom vrednošću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



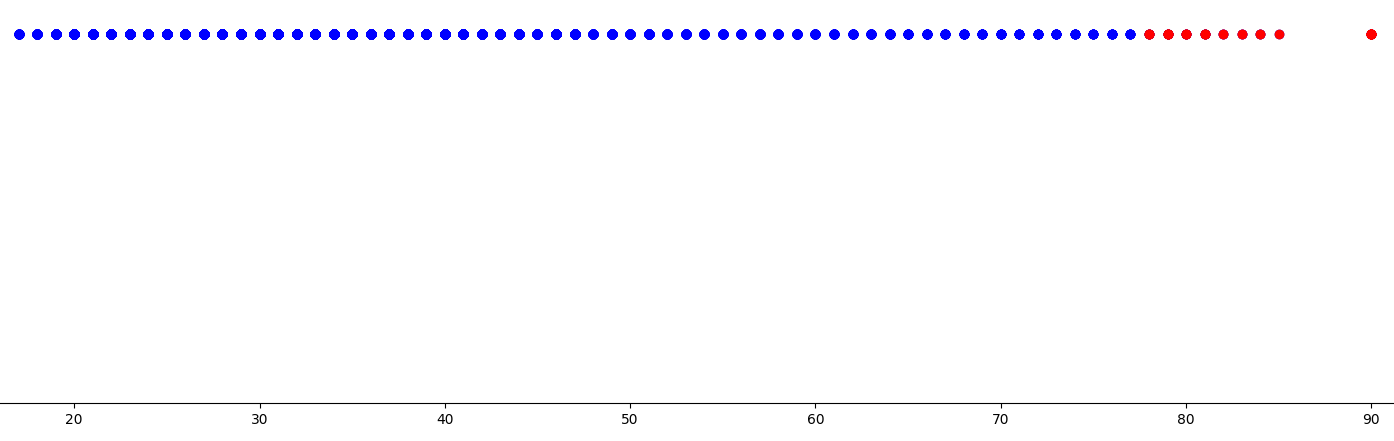
Zatim je izvršena detekcija outlier-a za određene kolone. Nije vršeno uklanjanje istih, već samo detekcija. Detekcija je izvršena *tukey* metodom u funkciji prikazanoj na sledećoj slici:



Vrednosti outlier-a kolone *age* su prikazani na sledećoj slici:

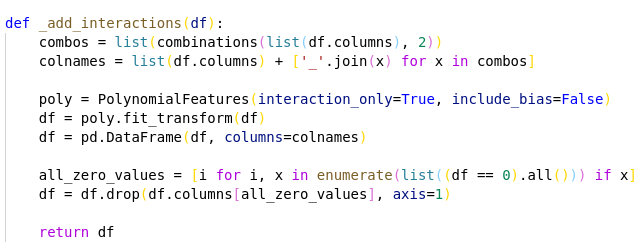


Vizuelni prikaz distribucije vrednosti outlier-a i svih vrednosti kolone *age* su prikazani na sledećoj slici:

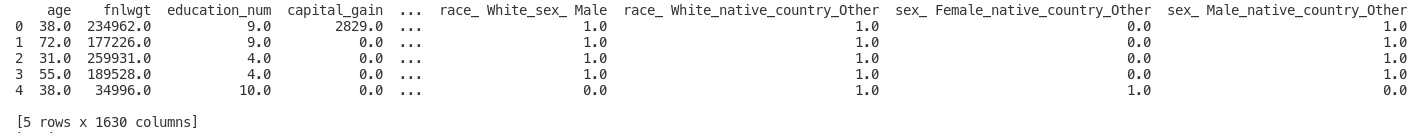


Na ovoj slici se mogu videti vrednosti izvan granica i one su prikazane crvenom bojom, dok su vrednosti koje ne odstupaju od uobičajenih vrednosti prikazane plavom bojom.

Često ulazne karakteristike interaguju na neočekivan i nelineralni način prilikom preditivnog modeliranja. Te interakcije mogu se 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 polinominalnih karakteristika. To je urađeno u funkciji prikazanoj na sledećoj slici:



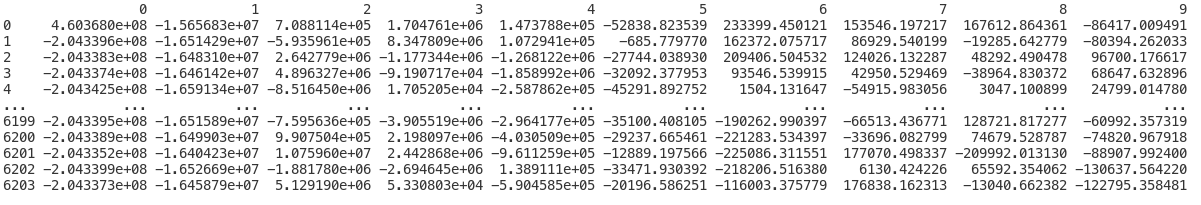
Dodate interakcije znatno povećavaju broj kolona. Prvih 5 redova dataframe-a sa dodatim interakcijama je prikazan na sledećoj slici:



Na ovoj slici se može videti da dataframe sada ima 1630 kolona, što je znatno više od početnih 15. Procesiranje toliko 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:

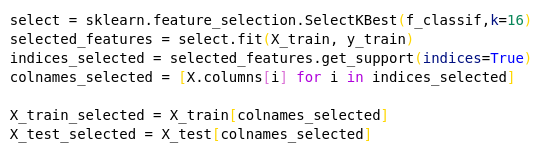


Dataframe nakon redukcije dimenzionalnosti ima 10 kolona, ali te kolone nisu više povezane sa prethodnim, već nove kolone dobijaju nove numeričke nazive. Izgled dataframe-a nakon redukcije dimenzionalnosti korišćenjem PCS metode je prikazan na sledećoj slici:

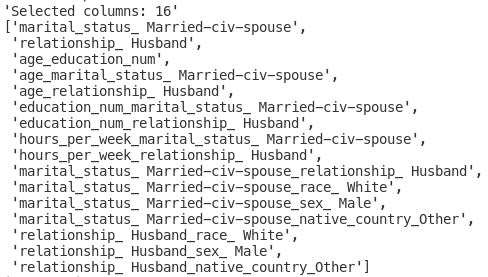


Na slici se može videti da imena novih kolona nemaju nikakvo značenje, a i vrednosti u kolonama je teško razumeti. Zbog teškog tumačenja atributa i njihovih vrednosti, odlučeno je da se u implementaciji ne koristi PCA metoda za redukciju dimenzionalnosti.

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:



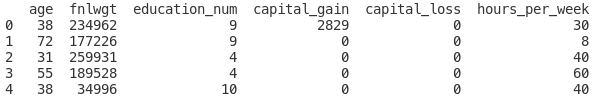
Imena selektovanih kolona su prikazana na sledećoj slici:



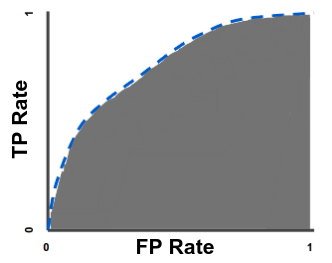
Takođe, izvršena je i kros validacija skupa podataka na trening i test skup. Veličina trening skupa predstavlja 90% celokupnog skupa podataka. Kros validacija je izvršena korišćenjem biblioteke *sklearn*, a kod je prikazan na sledećoj slici:



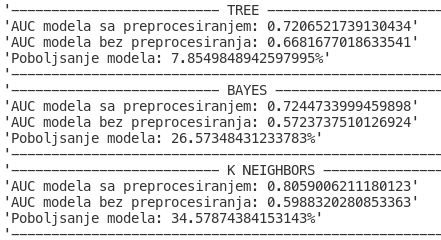
Ovim je korak preprocesiranja podataka završen. Zatim se prelazi na treniranje modela i upoređivanje rezultata pre i nakon preprocesiranja podataka. Iz skupa neprocesiranih podataka su izbačene kolone sa nenumeričkim podacima, kao i redovi sa nedostajućim vrednostima. Izgled skupa neprocesiranih podataka je dat na sledećoj slici:



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:



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 sledećoj slici:



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.

# ZAKLJUČAK

Nadglednao učenje je najjednostavnija potkategorija mašinskog učenja i služi kao uvod u mašinsko učenje mnogim praktičarima mašinskog učenja. Učenje pod nadzorom je najčešće korišćeni oblik mašinskog učenja i pokazalo se kao odličan alat u mnogim poljima.

# LITERATURA

1. <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" \l "The_birth_of_artificial_intelligence_1952%E2%80%931956)
5. <https://en.wikipedia.org/wiki/Timeline_of_machine_learning>
6. <https://en.wikipedia.org/wiki/Machine_learning>
7. <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>
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" \l ":~: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>