Univerzitet u Beogradu
Elektrotehnički fakultet

Analiza i predikcija saobraćajnih nesreća

MASTER RAD

Kandidat:
Marko Domić 2013/0240

Profesor:
prof. dr Boško Nikolić

Beograd
septembar 2018.

Sadržaj

1. UVOD

U većini struka kojim se čovečanstvo bavi, sve je više prisutniji porast samih podataka koje se koriste. Njihovom obradom i analizom, vrše se proračuni koji dovode do rezultata od velikog značaja za samu problematiku određene oblasti. Kako se skup tih podataka vremenom povećava, samom čoveku je sve teže da ih analizira bez pomoći određenih alata za njihovu obradu. Današnja brzina stvaranja novih informacija dovodi do izražaja nemogućnosti tražioca da se nosi sa nejednoznačnošću sintakse i semantike prirodnog jezika.[1]

Prateći rast količine podataka iz dana u dan, javlja se velika potreba za razvijanjem savremenih alata i drugih softverskih rešenja, kako bi se njihova obrada lakše i brže izvršavala. Trenutno postoji mnogo alata sa kojim možemo istraživati podatke, ali moramo naglasiti da ne postoji ni jedan univerzalan, koji će nam rešiti sve probleme. Svaki problem istraživanja podataka je novi izazov za koji ne znamo tačno kojim ćemo metodom ili alatom dobiti najbolji rezultat. Zbog toga se obično koriste više alata čije rezultate upoređujemo i biramo najbolje rešenje.[2]

Kao jedan od najčešće korišćenih alata za obradu velikih količina podataka koristi se **mašinsko učenje**, kao podoblast veštačke inteligencije. Ono je proisteklo iz proučavanja, prepoznavanja obrazaca i teorije računskog učenja, te korišćenja statistike u cilju učenja na osnovu prethodno dostupnih podataka.[3] Algoritmi mašinskog učenja su konstruisani da se adaptiraju na podatke nad kojim vrše treniranje, kako bi mogli ubuduće da se koriste kao alati za analizu i predikciju novih ishoda tih podataka. Detaljnije o samom mašinskom učenju će biti objašnjeno u drugoj glavi ovog rada. Objasniće se kako mašinsko učenje radi, njene podele i njihove namene.

U trećoj glavi biće nabrojani i detaljnije opisani tri regresivna algoritma mašinskog učenja koji će se kasnije iskoristiti za okvirnu predikciju lokacija saobraćanijh nezgoda u Beogradu pod određenim uslovima. Zbog prirode problema, kao i samih podataka koji se koriste prilikom predikcija, od koristi su samo regresivni algoritmi. Za rešavanje problematike ovog rada, korišćeni su sledeći algoritmi: *Linear regression, Decision tree regression* i *Random forest regression.*

Nakon toga, u četvrtoj glavi će se krenuti od detaljnijeg opisa samog problema saobraćajnih nezgoda u Beogradu, preko sređivanja ulaznih podataka pa do predikcije lokacija gde bi one mogle biti u određeno vreme. Nakon detaljnog opisa problema na samom početku, biće

objašnjeno prikupljanje ulaznih podataka vezane za saobraćajne nesreće (tačno vreme, lokacija, tip nezgode...), kao i njihovo prilagođavanje regresivnim algoritmima. Sređeni podaci se nakon toga koriste kao podloga za učenje algoritama opisanih u trećem poglavlju, kao i za predikciju koordinata određenog broja saobraćajnih nezgoda u određeno vreme. Rezultati se upoređuju i zaključuje se koji je najpogodniji za datu problematiku.

U petoj glavi, autor će iskazati svoj lični zaključak ovog rada, kao kratak rezime od postavljanja ciljeva do izračunavanja samih rezultata. Pored toga, objasniće prednosti i mane svog rešenja, kao i mogućnost njegovog poboljšanja ili unapređenja.

2. MAŠINSKO UČENJE

Mašinsko učenje je podoblast veštačke inteligencije čiji je cilj konstruisanje algoritama i računarskih sistema koji su sposobni da se adaptiraju na analogne nove situacije i uče na bazi iskustva. Razvijene su različite tehnike učenja za izvršavanje različitih zadataka.[4]

Mašinsko učenje omogućava sistemu da automatski uči i unapređuje se na osnovu iskustva dobijenih od podataka, bez ekplicitnog njihovog dodavanja programski. Ono se fokusira na razvoj softvera koji sam pristupa podacima, koristi ih i uči sam za sebe. Razvijene su različite tehnike učenja za izvršavanje različitih zadataka. Prve koje su bile predmet istraživanja, tiču se nadgledanog učenja za diskreciono donošenje odluka, nadgledanog učenja za kontinuirano predviđanje i pojačano učenje za sekvencionalno donošenje odluka, kao i nenadgledano učenje.[4]

Dosada najbolje shvaćen od svih navedenih zadataka je odlučivanje preko jednog pokušaja (engl. one-shot learning). Računaru je dat opis jednog objekta (događaja ili situacije) i od njega se očekuje da kao rezultat izbaci klasifikaciju tog objekta. Na primer, program za prepoznavanje alfanumeričkih znakova kao ulaznu vrednost ima digitalizovanu sliku nekog alfanumeričkog znaka i kao rezultat treba da izbaci njegovo ime. Izuzetno bitna distinkcija između mašinskog učenja i veštačke inteligencije nalazi se u cilju njihovog operisanja: dok veštačka inteligencija ima za cilj ne samo da imitira ljudsko razmišljanje kroz učenje, već i da to bude prožeto apstraktnim razmišljanjem, predstavljanjem znanja i rasuđivanjem, mašinsko učenje je samousmereno ka stvaranju softvera koji može da uči iz prošlih iskustava. Mašinsko učenje je znatno bliže data miningu i statističkoj analizi. Neki čak veruju da je napredovalo ispred statistike time što se oslanja na tačnost predviđanja, nasuprot čistom modelovanju podataka.[5]

Danas mnogi korisnici tehnologija mogu imati veliki benefit koristeći mašinsko učenje. Ono je implementirano u mnogim tehnologijama koje svakodnevno koristimo, kao što su:

- Skeniranja i prepoznavanja lica sa slika radi lakšeg nalaženja korisnika na društvenim mrežama;
- OCR (*Optical Character Recognition*) - pomaže prilikom skeniranja tekstualnih dokumenata sa papira;
- *Recommendation engines* – softveri koji koriste mašinsko učenje koje prati interesovanja korisnika i predviđa šta bi moglo da ga interesuje, od serija i filmova koje bi mogao da pogleda, do produkta za koje bi bio zainteresovan da ih kupi;
- U skorijoj budućnosti, samovozeći automobili, prilikom određivanje rute kretanja.

Primeri su brojni. Mašinsko učenje kao nauka se neprestano razvija. Kako vreme prolazi,

konstantan razvoj tehnologije utiče i na sam razvoj mašinskog učenja, unapređujući ga u jedno od najkorišćenijih i najtraženijih alata današnjice.

2.1. ISTORIJA MAŠINSKOG UČENJA

Pojam ”*mašinsko učenje*” se javlja još daleke 1959. godine u dotadašnjoj IBM kompaniji. Izmislio ga je tadašnji američki *pioneer* veštačke inteligencije gejming industrije, Arthur Samuel. Prilikom razvoja veštačke inteligencije, naučnici su hteli da mašina sama uči iz seta podataka. Kako bi to realizovali, pristupali su problemu na razne načine, a jedan od njih je implementacija neuralne mreže. Korišćeni su i linearni modeli statistike, kao i verovatnoća, pogotovu u automatizovanju postavljanja medicinskih dijagnoza.

Sve veći naglasak na logičkom pristupu baziranom na znanju, razlika između veštačke inteligencije i mašinskog učenja je sve više rasla. Do 1980. godine, ekspertski sistemi su se probili kao najznačajniji deo veštačke inteligencije, dok je statistika ostala u senci. Rad na simboličnom učenju (učenju zasnovanog na znanju) nastavio se u krugu veštačke inteligencije, što je dovodilo do induktivnog logičkog programiranja, a ispadanje statistike iz kruga za prepoznavanje uzoraka i pronalaženja informacija.

Mašinsko učenje i *data mining* često implementiraju iste metode (algoritme), pa se samim tim preklapaju u većini slučajeva. Ključna razlika između ova dva pojma je što mašinsko učenje predviđa rezultate na osnovu poznatih podataka koje je kroz trening naučio, dok kod *data mining*-a se otkrivaju nepoznati (prethodni, nestali, suprotno od predikcije) podaci unutar samog skupa podataka. Zarad postizanja željenog, *data mining* koristi dosta algoritama koje koristi mašinsko učenje, samo sa drugačijim ciljem. Slično tome, mašinsko učenje koristi algoritme *data mining*-a, najčešće prilikom unapređivanja samih podataka nad kojim će da istrenira svoju mrežu zarad veće preciznosti.

Statistika i mašinsko učenje su danas jako usko povezani. Po američkom naučniku Michael I. Jordan-u, mašinsko učenje, od metodoloških principa do teorijskih alata, ima dugačku istoriju usko povezanu sa statistikom. Takođe, on predlaže da bi bilo poželjno celu nauku nazvati *data science* (nauka o podacima).

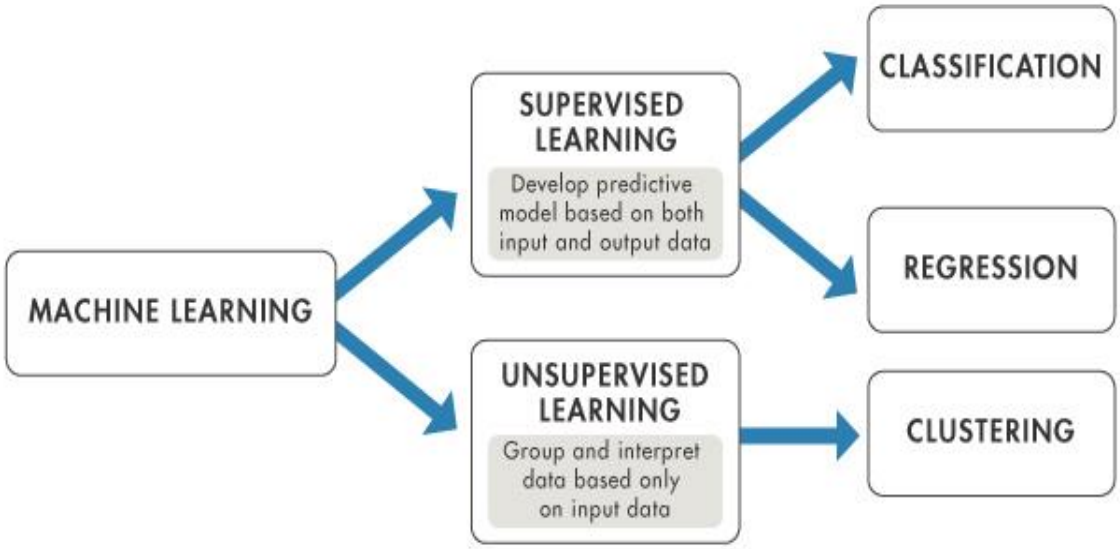
Leo Breiman, američki statističar, je striktno razdvojio dva modela rada: model podataka i model algoritama. U model algoritama spadaju algoritmi mašinskog učenja, kao što je *Random forest*.

Neki statističari su prisvojili metode mašinskog učenja i tako mešovitu oblast su nazvali *statističko učenje*.[6]

2.2. METODE MAŠINSKOG UČENJA

U zavisnosti od zadataka koje mašinsko učenje mora da izvrši, možemo ih klasifikovati u više kategorija. Postoje više metoda učenja koje se koriste u zavisnosti od strukture podataka nad kojima se ono vrši. Takođe, postoji i više kategorija koje grupišu same metode učenja, ali sledeće dve su najzastupljenije:

1. Nadgledano učenje (*Supervised machine learning algorithms*);
2. Nenadgledano učenje (*Unsupervised machine learning algorithms*);

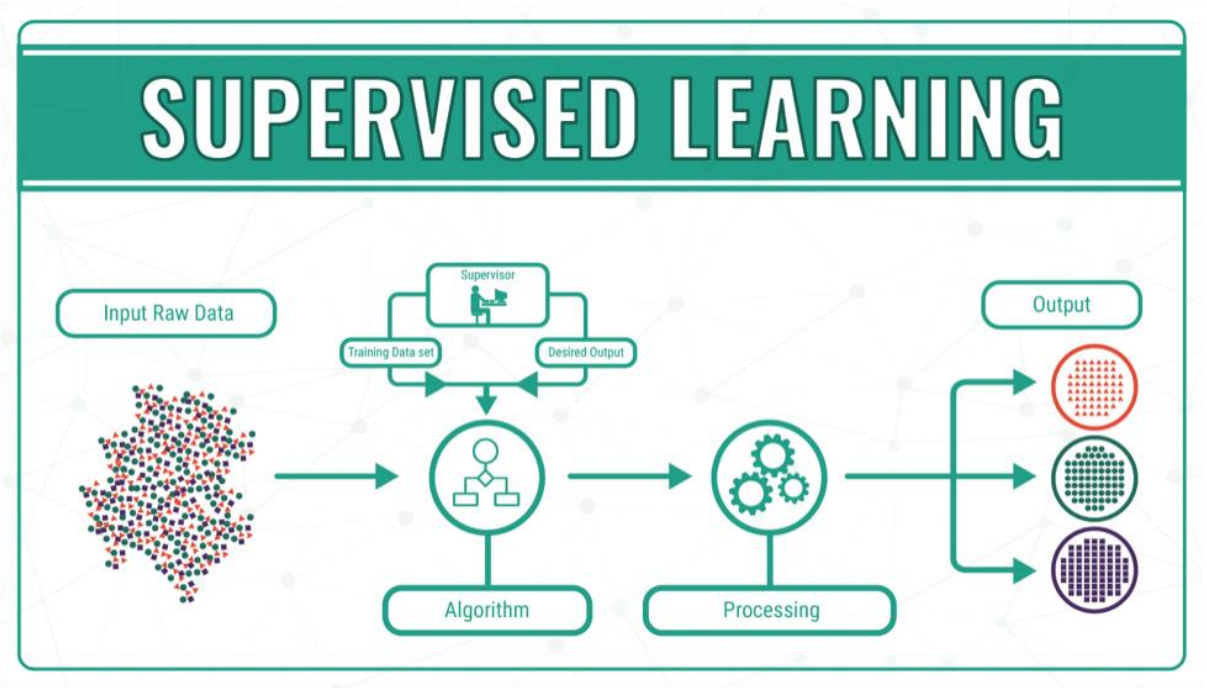


***Slika 2.2**: Podela algoritama mašinskog učenja. [7]*

2.2.1. Nadgledano učenje

Kod mašinskog učenja koji koristi algoritme **nadgledanog učenja** (*Supervised machine learning algorithms*), ono poseduje podatke koji su pravilno strukturirani i obeleženi (*labeled*). Prilikom učenja, svaki podatak u skupu nad kojim se vrši učenje, poseduje vrednosti koje služe kao ulazne vrednosti u algoritam i izlazne vrednosti koje bi trebalo algoritam da vrati (izračuna) nakon unetih ulaznih vrednosti. U većini slučajeva, čovek je taj koji obeležava podatke koje će algoritam koristiti i time definiše klasu kojoj podatak pripada. Za svaki obeleženi ulazni podatak, zna se njegov izlazni na osnovu kojeg se vrši samo učenje (po tome je ovaj oblik učenja dobio ime nadgledano, nadzorno učenje).

Na osnovu njih, algoritam mašinskog učenja trenira svoju mrežu tako što svoju izlaznu vrednost za određeni podatak upoređuje sa izlaznom izračunatom vrednošću. U zavisnosti od odstupanja od greške (razlika između tačne i predviđene vrednosti) prilikom upoređivanja, algoritam vrši korekciju svog rada, kako bi u narednoj predikciji sledeće odstupanje bilo što manje. Svakom sledećom iteracijom kroz podatke učenja (u daljem tekstu *dataset*), algoritam postaje precizniji nego pre. Nakon treninga, kada algoritmu postavimo samo obeležene ulazne podatke bez izlaznih, on će moći da ih predvidi na osnovu prethodnih iskustava nad kojim je istrenirao svoju mrežu.



***Slika 2.2.1**: Nadgledano učenje nad obeleženim skupom podataka. [8]*

U algoritmima nadgledanog mašinskog učenja su implementirane funkcije koje prave vezu između ulaznih i izlaznih obeleženih podataka. Broj različitih algoritama je obiman da bi se svaki od njih detaljno objasnio u ovom radu. U trećoj glavi će biti detaljnije opisani oni algoritmi nadgledanog učenja koji su korišćeni za problematiku ovog rada i time se upoznati sa osnovnim konceptima kako oni funkcionišu.

Na osnovu podataka za učenje, postoje dva osnovna problema koje oni mogu obrađivati. U pitanju su **regresija** i **klasifikacija**. Algoritmi koji se bave regresijom na osnovu ulaznih podataka koje uzimaju, predviđaju rezultat kontinuirane vrednosti. To znači da dobijena izlazna vrednost ne pripada skupu unapred definisanih vrednosti. Primera radi, u slučaju učenja nad cenama akcije skenirane sa berze, kao izlaza (predikciona) vrednost bi bila njena cena u određenom budućem vremenu. Vreme u ovom slučaju predstavlja ulazne vrednosti regresivnog algoritma, dok izlaz predstavlja cena. S obzirom da je izlaz izražen kao realan broj, njegova

vrednost nije prethodno definisana. Pošto je teško predvideti tačnu vrednost izlaza funkcije, merenje tačnosti algoritma se zasniva na proceni što manjeg odstupanja od greške. Kao najjednostavniji i najpopularniji algoritam regresije smatra se *linearna regresija*, o kojoj će biti više reči u 3. poglavlju.

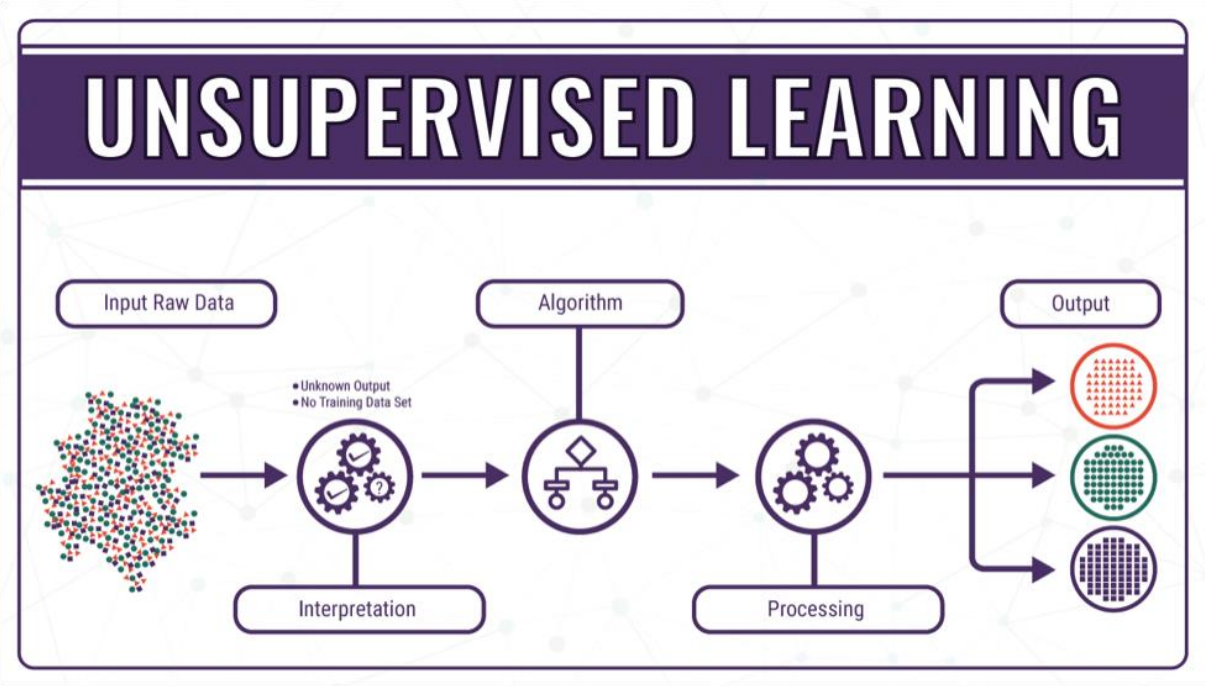
Skroz suprotno od regresivnih, izlazne vrednosti klasifikacionih algoritama su već prethodno definisane kroz dataset nad kojim je izvršen. Za rezultat možemo reći da određuje tačno kojoj klasi pripadaju ulazni podaci predikcije. Ovi algoritmi su korisni u slučajevima kada je neophodno razvrstati podatke na osnovu svojih obeleženih kriterijuma u zasebne grupe (klase). Dobar primer svakodnevne primene ovog algoritma je elektronska pošta (email) koju koriste veliki broj korisnika. Na osnovu tipa poruke i ostalih nekih poruka, klasifikacioni algoritmi mašinskog učenja raspoređuju poruke u posebne direktorijume, kao što su željene i neželjene poruke (*spam*). U zavisnosti od broja klasa koje mogu biti na izlatu, razlikujemo dve vrste klasifikacije:

1. Binarna - rezultat može biti jedna od dve klase, kao u primeru opisanom za klasifikaciju;
2. Višeklasna - rezultat može biti jedna od više klasa, kao npr. preopznavanje tipa nezgode koja se desila (postoji više od dve vrste nezgoda).

Kada se izgrađuje sistem koji će svoju primenu pronaći u rešavanju raznih stvarnih problema, tada se u većini slučajeva koristi višeklasna klasifikacija gde broj klasa može biti i do nekoliko stotina. Regresija za razliku od klasifikacije, kako je navedeno ranije u radu, ne svrstava podatke u klase već kao izlazni podatak predaje broj koji može predstavljati zaradu zaposlenog, procena zarade nekog preduzeća i sl. što je pogodno za skupove podataka u kojima ne postoje fiksno određene klase, nego vrednosti mogu varirati s obzirom na ulazne vrednosti.[7]

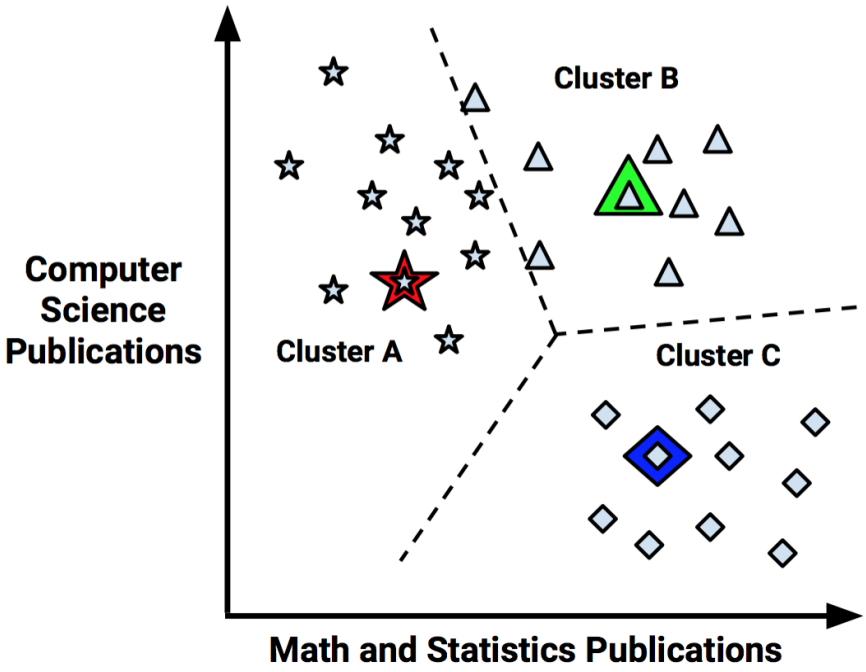
2.2.2. Nenadgledano učenje

Za razliku od skupova podataka koji su obeleženi i koriste se u algoritmima nadgledanih mašinskih učenja, nenadgledani algoritmi koriste podatke koje imaju samo ulazne vrednosti bez izlaznih sa kojim bi mogli biti povezani. Kako bi se mogla vršiti predikcija ovih algoritama, oni su u obavezi da prepoznaju određene pravilnosti (zakonitosti) u ulaznim podacima i na osnovu njih vrše predikcije. U većini slučajeva ulazni podaci se predstavljaju kao niz realnih brojeva.



***Slika 2.2.2.1**: Nenadgledano učenje nad neobeleženim skupom podataka. [8]*

Na osnovu predikcija koje su dali algoritmi, u zavisnosti od svojih vrednosti, izlazi se mogu svrstati u određene grupe. Takav vid obrade se naziva grupisanje podataka (engl. *clustering*). Dobar primer grupisanja u stvarnom svetu je svrstavanje kupaca u skupove s obzirom na njihove navike kupovanja. [9]



***Slika 2.2.2.2**: Grupisanje podataka. [10]*

3. REGRESIVNI ALGORITMI MAŠINSKOG UČENJA

Kako priroda problema nalaže, za izradu ovog rada korišćeni su regresivni algortimi mašinskog učenja. To se moglo zaključiti na osnovu samih ulaznih podataka nad kojim se učenje vrši, kao i prema vrednostima očekivanih predikcija. Kao što je pomenuto ranije, broj takvih algoritama je veliki i trebalo bi odabrati najpogodnije za dati problem, kako bi predikcija bila što preciznija. U ovom poglavlju će biti opisani oni algoritmi koji su dali relativno precizne predikcije.

3.1. LINEARNA REGRESIJA

Linearna regresija je jedna od najpoznatijih (ako ne i najpoznatiji) i najbolje razumljivih algoritama, kako u mašinskom učenju, tako i u statistici. Ovaj model se proučava i koristi poslednjih 200 godina, što dovodi do toga da postoji dosta sinonima koji asociranju na njega. Jedan od njih je i **linearni model** koji predstavlja linearnu vezu između ulaznih, nezavisnih, varijabli (x) i jedne izlazne, zavisne, varijable (y) koja je izračunata kombinacijom ulaznih vrednosti. Kada postoji samo jedan ulazni podatak, misli se na **jednostavnu linearnu regresiju**, u suprotnom se misli na **višestruku linearnu regresiju**. [11]

Različite tehnike se primenjuju za učenje i pripremu jednačine linearne regresije iz podataka. Najpoznatija je metoda najmanjih kvadrata (prosečna kvadratna greška), ali postoje i druge koje će biti objašnjene u ovom poglavlju. Iako je linearna regresija primarno stvorena u statističke svrhe, zarad razumevanja njenog rada nije neophodno poznavanje linearne algebre, niti detaljnije statistike.

3.1.1. Linearna jednačina

Linearna regresija je atraktivan i zanimljiv model zbog njene proste implementacije u odnosu na druge algoritme. U osnovi je linearna jednačina koja se kombinuje sa ulaznim vrednostima (jednom ili više) kako bi dobila rezultat (y) koji predstavlja predikciju. Zbog toga, i ulazni i izlani podaci jednačine moraju biti realni brojevi.

Za svaki ulazni parametar (x), vezuje se koeficijent (uglavnom se obeležava grčkim slobom Beta). Pored toga, dodaje se još jedan koeficijent koji se ne vezuje ni za jedan ulazni podatak,

već služi kao slučajna varijabla koja dodaje šum u linearnu relaciju između zavisne varijable i regresora. Drugim rečima, prilikom prikaza linearne funkcije u dvodimenzionalnom koordinatnom sistemu, ovaj slobodni koeficijent utiče na ugao pod kojim će kriva biti u odnosu na horizontalnu osu. Na osnovu opisanih parametra, formula za izračunavanje izlazne vrednosti je sledeća:

$$y = B_0 + B_1x_1 + B_2x_2 + \ldots + B_nx_n$$

gde je **y** rezultat predikcije, $\mathbf{x_{1..n}}$ ulazni podaci, $\mathbf{B_0}$ slobodni koeficijent i $\mathbf{B_{1..n}}$ koeficijenti koji se pridružuju ulaznim parametrima.

***Tabela 3.1.1**: Primeri jednačina linearne regresije*

Broj ulaznih parametra	Linearna funkcija
1	$y = B_0 + B_1x_1$
2	$y = B_0 + B_1x_1 + B_2x_2$
3	$y = B_0 + B_1x_1 + B_2x_2 + B_3x_3$
4	$y = B_0 + B_1x_1 + B_2x_2 + B_3x_3 + B_4x_4$
5	$y = B_0 + B_1x_1 + B_2x_2 + B_3x_3 + B_4x_4 + B_5x_5$

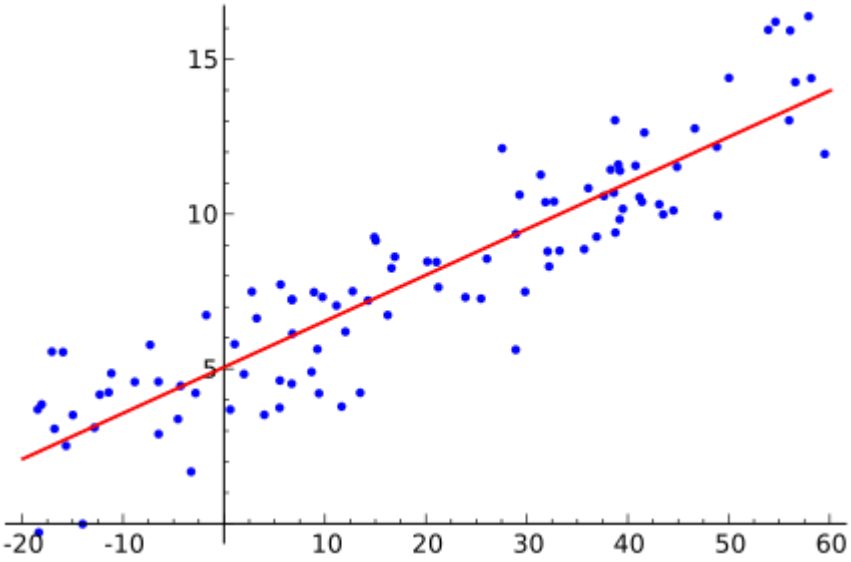
Kao što se može primetiti, sama implementacija algoritma je u stvari čista linearna jednačina sa više ulaznih vrednosti i koeficijentima koji utiču na njih. U slučajevima kada određeni koeficijent dođe do vrednosti 0, time on uklanja određenu ulaznu vrednost iz jednačine, što govori da vrednosti tih kolona su nepotrebne za učenje same mreže i mogu se ukloniti iz skupa ulaznih podataka.

3.1.2. Učenje modela

Termin učenja modela linearne regresije odnosi se na procenu vrednosti samih koeficijenata koji se koriste u linearnoj jednačini. Njihove procene se vrše na osnovu skeniranih ulaznih i izlaznih podataka dataset-a predviđenog za učenje. Pošto je linearna regresija jedan od najpoznatijih algoritama mašinskog učenja, danas postoji veliki broj tehnika koje modifikuju linearnu jednačinu, tačnije njene koeficijente, prilikom učenja.

3.1.2.1. PROSTA LINEARNA REGRESIJA

Kada je reč o prostoj linearnoj regresiji, odnosi se na učenja koja koriste podatke koji imaju samo jedan ulazni podatak i jedan izlazni. Za procenu vrednosti njihovih koeficijenata jednačina, koristi se statistika.



***Slika 3.1.2.1**: Prosta linearna regresija. [12]*

Formula jednačine za prostu linearnu regresiju je sledeća:

$$y_i = B_0 + B_1 x_i$$

gde je *i* redni broj iteracije prilikom čitanja dataset-a.

Posmatrajući sliku 3.1.2.1, može se zaključiti da je nemoguće naći vrednosti ulaza i izlaza koje mogu ispoštovati formulu. Samim tim, u formulu se uvodi novi parametar:

$$y_i = B_0 + B_1 x_i + e_i$$

gde vrednost e_i parametra označava razliku dobijene vrednosti ulaznih parametara sa koeficijentima od odgovarajuće izlazne vrednosti (odstupanje originalnih vrednosti). Cilj samog algoritma je pronaći idealne B_0 i B_1 koeficijente tako da jednačina bude što preciznija i koeficijent odstupanja od originalnih vrednosti što manji.

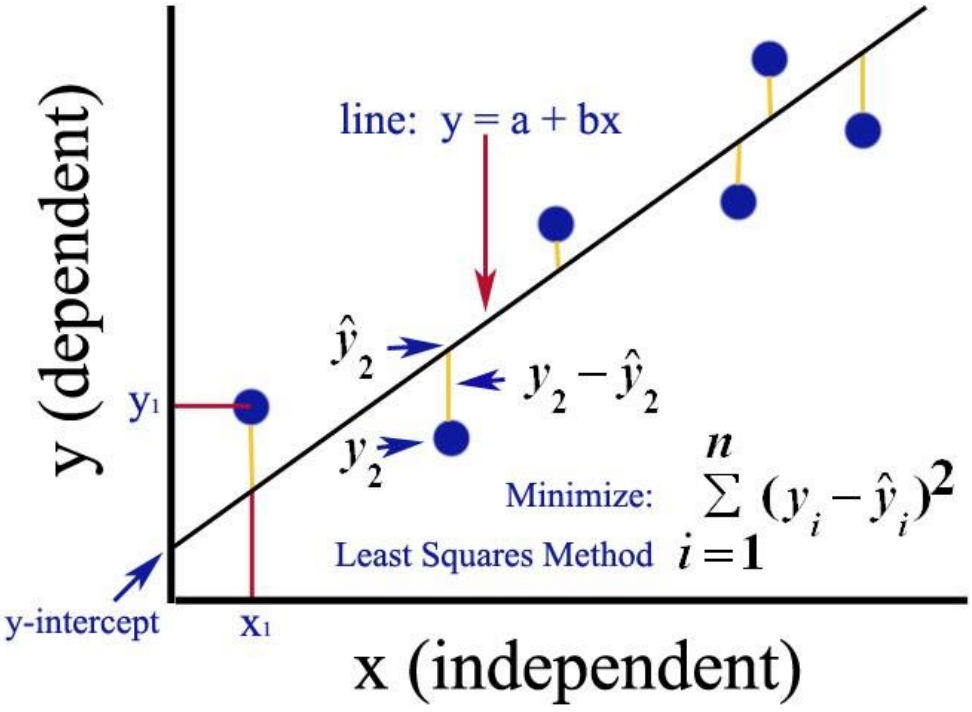
3.1.2.2. PROSEČNA KVADRATNA GREŠKA

U ovoj metodi imamo više od jedne ulazne vrednosti i neophodno je uraditi procenu za više koeficijenata jednačine. On pokušava da minimizuje zbir kvadratnih ostataka.

Kako sada postoji više od jedne ulazne vrednosti, koristeći linearne jednačine, dobija se više jednačina, a samim tim i više izlaza:

$$y_1 = B_0 + B_1 x_1$$
$$y_2 = B_0 + B_1 x_2$$
$$\vdots$$
$$\vdots$$
$$y_n = B_0 + B_1 x_n$$

gde je n broj ulaznih vrednosti. Svaki y od 1 do n odstupa od svoje tačne vrednosti koja je definisana kao očekivani izlaz funkcije. Kako bi se našla najefikasnija jednačina za predikciju izlaznih vrednosti, neophodno je naći takav y da je suma svih kvadrata razlike između tražene i dobijene vrednosti bude minimalna. Formula koja daje takve parametre takve jednačine, kao i sam grafik gde je ilustrovana cela metodologija je detaljno prikazana na slici 3.1.2.2. Promenljiva *n* označava broj ulaznih parametara u jednačini.



***Slika 3.1.2.2**: Prosečna kvadratna greška. [13]*

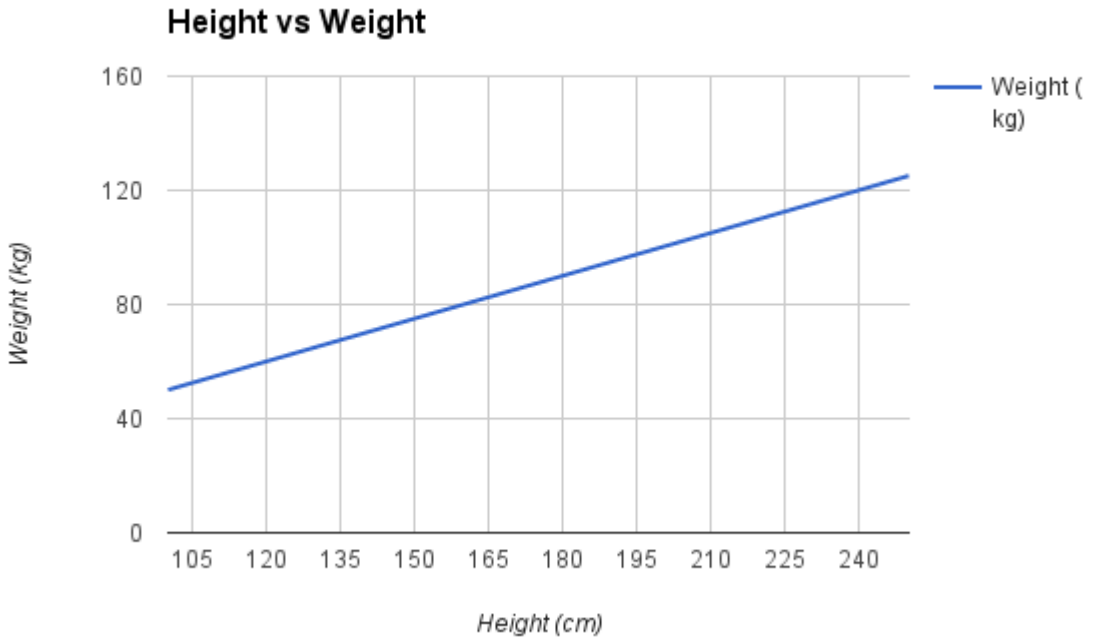
3.1.3. <u>Pravljenje predikcije</u>

Kako je u prethodnom podpoglavlju objašnjeno kako se treniraju linearne jednačine same linearne regresije, pravljenje same predikcije je lakši deo procesa. Sređena jednačina dobijena kroz proces učenja modela, sada se koristi za kalkulaciju izlaznih podataka na osnovu ulaznih. Ulazni podaci jednačine su ulazni podaci za kalkulaciju predikcije, a sam rezultat je predikcija.

Kroz konkretan primer je najbolje objašnjeno kako se vrši predikcija. Trebalo bi odrediti masu osobe (y u jednačini) na osnovu njene visine (ulazni podataka x). [11] Jednačina je sledeća:

$$y = B_0 + B_1 x$$

B_0 i B_1 su koeficijenti jednačine izračunati prethodnim učenjem proste linearne regresije nas skupom podataka (visina i masa osoba). Ako su dobijeni koeficijenti 0.1 i 0.5 respektivno, za ulazne (visine osoba) se uzimaju vrednosti od 100 do 250 (vrednosti su izražene u cm). Grafikon sa rezultatima je prikazan na slici 3.1.3.



***Slika 3.1.3**: Rezultat predikcije mase osoba na osnovu njihovih visina. [11]*

3.1.4. Priprema podataka za učenje linearne regresije

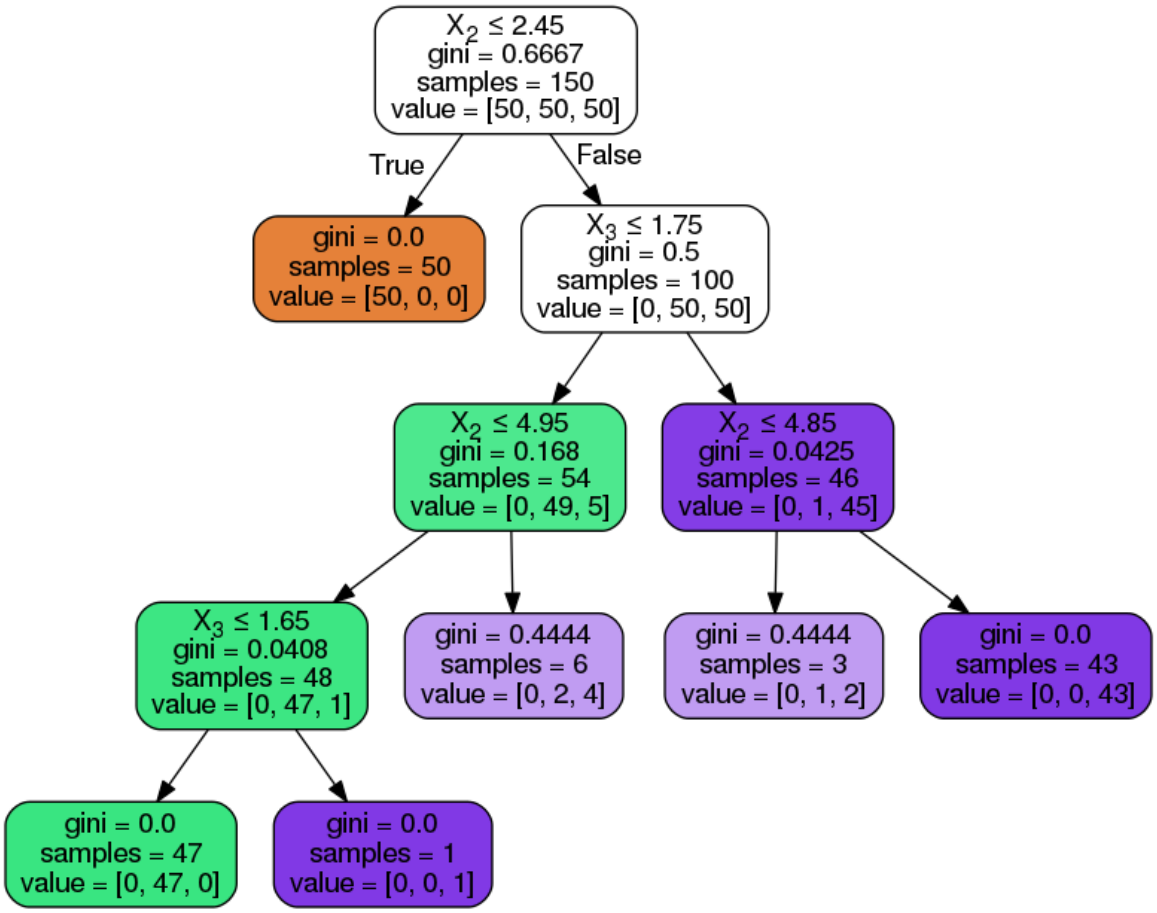
Kako bi mašinsko učenje radilo efikasno i precizno, neophodno je adekvatno prilagoditi podatke, kako bi ih algoritmi linearne regresije kvalitetno i precizno skenirali i obučili se pomoću njih. Postoje određena pravila za samu pripremu podataka koje mnogi naučnici prate zarad što bolje predikcije.

Za pripremu podataka postoje mnoge heruistike metode, kao što su:

- Linearna pretpostavka – vezu između ulaznih i izlaznih podataka linearna regresija smatra kao linearno zavisnu i samo kao takvu. Ovo je značajno u slučajevima velikog broja ulaznih parametara;
- Izbacivanje šumova – linearna regresija pretpostavlja da nema nepotrebnog šuma (razlika između izlazne i tačne vrednosti). Savetuje se da se koriste takvi podaci koji bi proizvodili što je manje moguće šuma;
- Izbacivanje zavisnosti – u slučaju zavisnosti između ulaznih podataka, može doći do *over-fit* podataka;
- Reskaliranje podataka – predikcije linearne regresije su pouzdanije ako su podaci nad kojim se vrši učenje reskalirani koristeći standardizaciju i normalizaciju.

3.2. DECISION TREE

Stablo odlučivanja (*Decision tree*) spada u grupu nadgledanih algoritama mašinskog učenja. Iako se u većini slučajeva koristi kao klasifikacioni algoritam, takođe je od koristi i za regresivne probleme. Kako ono pokriva i regresivne i klasifikacione situacije, ovaj algoritam je kao zajedničko ime dobilo CART (*Classification And Regression Tree*).



***Slika 3.2**. Primer stabla odlučivanja. [14]*

Struktura koja se koristi u srži ovog algoritma je stablo. Svaki čvor unutar tog stable predstavlja sam test nad ulaznim podatkom, svaka grana koja izlazi iz tog čvora prestavlja ishod tog testa, a list stabla predstavlja krajnji obeleženi rezultat predikcije. Prilikom svake odluke, kreće se od korenog čvora (root node). U analizi samih odluka i verdnosti, stablo odlučivanja može da posluži kao struktura pomoću koje je moguće vizuelno predstaviti rezultate.

Prednosti CART-a su brojna, kao što su:

- Samo razumevanje, implementacija i vizualizacija su krajnje jednostavne;
- Moguće je raditi sa varijabilnim ulazima, kao i sa kategoričnim;
- Laka priprema podataka što dovodi do uštede vremena i energije pred samu predikciju. Ulazne vrednosti bilo kakvog tipa, ne utiču na performanse rada algoritma.

Naravno, kao i svaki drugi, ni ovaj algoritam nije savšen. Postoje određene mane koje karatkerišu CART algoritam:

- Moguće je da algoritam napravi previše kompleksno stablo odlučivanja, što dovodi do toga da podaci nisu uvek generalizovani dobro. Ovakav slučaj se u mašinskom učenju naziva *Overfitting*.
- Stabla mogu biti nestabilna ako rade sa podacima gde se javljaju male varijacije između njih. To dovodi do generisanja kompletno pogrešnog stabla odlučivanja. Takvi problemi se uglavnom rešavaju tehnikama pakovanja (*Bagging)* i podsticanja (*Boosting*).

Zbog korišćenja pohlepnih algoritama (*Greedy algorithms*), oni ne mogu granatovati generisanje najoptimalnijeg stabla odlučivanja. Samim tim što izgenerišu u datom momentu optimalno rešenje, to ne znači da je ono najoptimalnije na globalnom nivou. Rešavanje ovakvih problema se postiže generisanje više vrsta različitih stabla i upoređivanje njihovih rezultata. Sledeći opisan algoritam se bavi upravom ovakvom problematikom.

3.2.1. Primeri CART-a

Sa problemima kakve ovaj algoritam rešava se čovek susreće svaki dan. Kada su klasifikacioni problemi u pitanju, mnoge kompanije danas koriste ovakve algoritme za predikcije u svojim poslovnim domenima. Na osnovu imovinskog, zdravstvenog i mnogih drugih stanja, osiguravajuće kuće procenjuju cene svojih polisa osiguranja. Moguće je i na osnovu visine i težine predvideti da li je osoba muškog ili zenskog pola.

Što se regresivnih problema tiče, moguće je predvideti statističke podatke preživljavanja

ljudi na bordu koji tone. Na osnovu ulaznih podataka kao što su pol osoba i njihova starost, kalkulacije algoritma mogu biti izražene šanse u procentima njihovog preživljavanja. Postoje primeri kao što su predviđanja cena nekretnina na osnovu njihovog stanja. Kao ulazni parametri bi se uzimali lokacija, tip nekretnine, površina, broj soba, itd.

3.2.2. Regresivna stabla odlučivanja

Kao što je pomenuto ranije, regresivna stabla se koriste kada su zavisne, ulazne vrednosti dataset-a kontinuirane (vrednosti nisu definisane i klasifikovane, uglavnom predstavljene u vidu realnih brojeva). Prilikom skeniranja podataka nad kojim se vrši učenje, dele se na manje podskupove, u vidu stabla gde su čvorovi **odluke**, a listovi **izlazne (predikcione) vrednosti**. Svaki čvor odluke ima dve ili više grane koje izlaze iz njega i predstavljaju rezultat njegove odluke. Koreni čvor (**root**) se smatra kao najbolji prediktor od svih čvorova odluka i zato se uvek kreće od njega. Krajnji čvorovi stabla (listovi) predstavljaju krajnju odluku stabla, tj. predikciju na osnovu ulaznih parametara, koja je u većini slučajeva regresivnih stabla odlučivanja numeričkog tipa.

Glavni algoritam koji služi za izradu stabla odlučivanja se zove **ID3**, kreiran od strane J. R. Quinlan-a, koji koristi od vrha nadole pohlepnu (*greedy*) pretragu kroz prostor mogućih grana bez povratka. ID3 algoritam se može iskoristiti prilikom izrade regresivnog stabla odlučivanja, korišćenjem tehnike standardnog smanjenja odstupanja (*Standard Deviation Reduction*). [15]

Kako izrada ide sa vrha nadole od korenskog čvora, ono podrazumeva podelu podataka u podskupove koji sadrže primerke sa sličnim vrednostima. Koristi se standardna devijacija za izračunavanje homogenosti (jednoznačnosti) numeričkog uzorka. Ako je uzorak potpuno homogen, njegova standardna devijacija je nula.

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$\boldsymbol{Count} = n = 14$$

$$\boldsymbol{Average} = \bar{x} = \frac{\sum x}{n} = 39.8$$

$$\boldsymbol{Standard\ Deviation} = S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}} = 9.32$$

$$\boldsymbol{Coeffeicient\ of\ Variation} = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

***Slika 3.2.2.1**: Standardna devijacija sa jednim atributom [15]*

$$S(T,X) = \sum_{c \in X} P(c)S(c)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = **P**(Sunny)***S**(Sunny) + **P**(Overcast)***S**(Overcast) + **P**(Rainy)***S**(Rainy)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

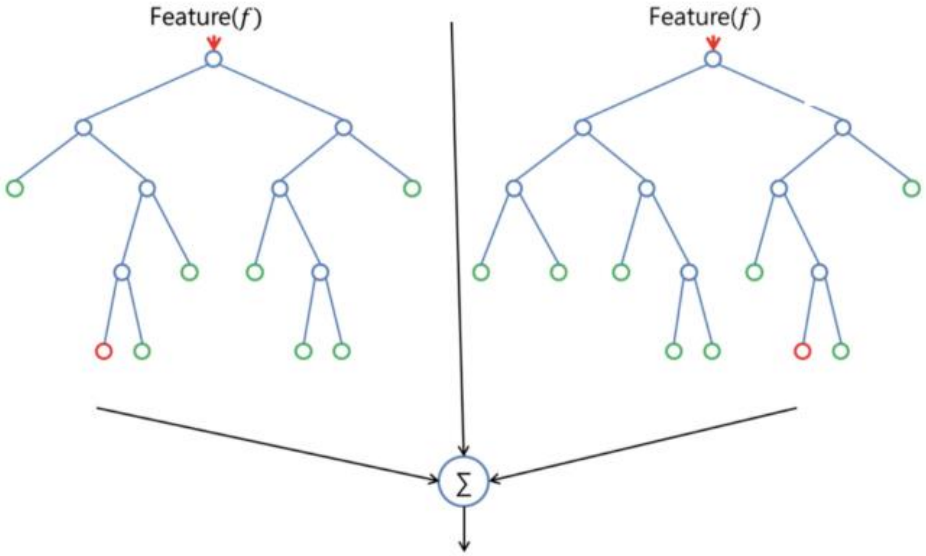
***Slika 3.2.2.2**: Standardna devijacija sa 2 atributa [15]*

3.3. RANDOM FOREST

Algoritam učenja u obliku slučajnih šuma (*Random forest*) je takođe jedan od lakših algoritama mašinskog učenja koji i bez preterano velikog podešavanja putem parametara može da da dobre rezultate u većini slučajeva. S obzirom da se može koristiti i nad klasifikacionim i nad regresivnim problemima, takođe je i jedan od najiskorišćenijih algoritama.

Kako mu samo ime kaže, algoritam koristi takozvanu „šumu“ podmetoda koje vrše predikcije za određene ulazne podatke. Predikcije se sumiraju i posebnim metodama se filtrira najbolja. Te podmetode su realizovane u vidu stabla odlučivanja (*Decision Trees*) istrenirane u većini slučajeva metodom pakovanja (*bagging*). Drugim rečima, algoritam slučajnih šuma se sastoji od više stabla odlučivanja koja vrše predikciju i spaja njihove rezultate kako bi dobio što preciznije i stablinije predikcije ulaznih vrednosti.

Što se tiče samih parametara, algoritam ima skoro pa iste parametre kao i algoritam stabla odlučivanja od koga se sastoji. Upotreba više stabla odlučivanja sa metodom pakovanja nisu više potrebni toliko, s obzirom da je ovakav koncept već implementiran unutar samog ovog algoritma.



***Slika 3.3**: Izled algoritma učenja slučajnih šuma sa više stabla odlučivanja [16]*

3.3.1. Primer iz realnog života

Najbolji način kako da se opiše rad algoritma je kroz primer sličan realnom životu. Ako određena osoba koja se zove Andrija, želi da odluči na koja mesta želi da otputuje tokom jednogodišnjeg odmora. On počinje da pita ljude koje zna za savet. Prvo odlazi kod prijatelja koji ga pita na koje destinacije je išao pre i da li su mu se svidele ili ne. Na osnovu odgovora, prijatelj mu daje savet.

Ovo je tipičan primer stabla odlučivanja. Andrijin prijatelj kreira pravila koja vode do njegove odluke o preporuci tako što koristi Andrijine odgovore.

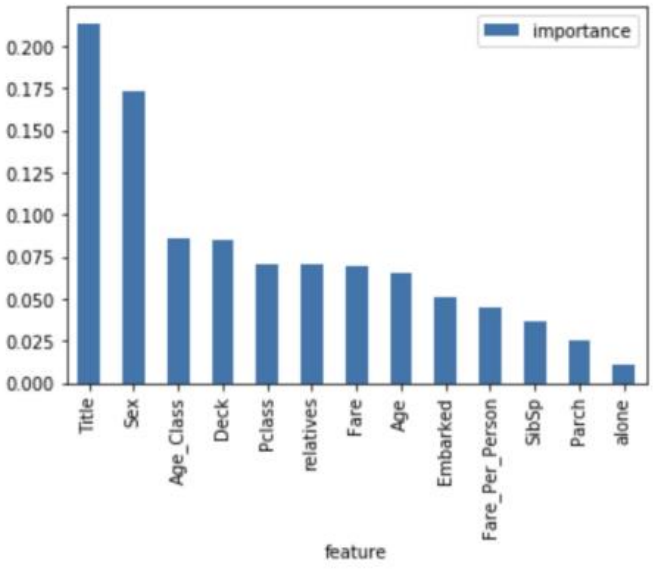
Potom, Andrija počinje da pita više prijatelja da ga posavetuju i oni ga takođe pitaju pitanja na osnovu kojih mogu da daju preporuku. Zatim on odabere destinaciju koja je predložena najviše, što je tipičan primer algoritma učenja slučajnih šuma.[16]

3.3.2. Prioriteti odluka

Jedna od dobrih opcija ovog algoritma je ta što je moguće izmeriti prioritet samih odluka predikcije unutar čvorova stabla odlučivanja. Korišćenjem Python biblioteke Sklearn, takođe iskorišćena prilikom izrade samog rada, moguće je izdvojiti metrike prioriteta odluka tako što gledaju koliko čvorova stabla koriste tu odluku. Metrike se računaju automatski posle treninga. U zbiru prioriteti moraju davati 1.

Na osnovu ovih metrika, moguće je utvrditi koji ulazni parametri su od koristi a koji ne, što znatno pomaže prilikom sređivanja dataset-a. Što je više podataka u samom dataset-u, to su veće šanse da model zahvati problem overfitting-a.

feature	importance
Title	0.213
Sex	0.173
Age_Class	0.086
Deck	0.085
Pclass	0.071
relatives	0.070
Fare	0.069
Age	0.065
Embarked	0.051
Fare_Per_Person	0.045
SibSp	0.037
Parch	0.025
alone	0.011



***Slika 3.3.2**: Primer metrike prioriteta ulaznih parametara [16]*

3.3.3. Parametri algoritma

Prilikom korišćenja *Random forest*-a, korisniku je omogućeno da modifikuje sam rad algoritma postavljanjem određenih parametara. Kombinacijom dobrih parametara u zavisnosti od samih ulaznih podataka i kako su oni organizovani, znatno utiče na generisanje boljih predikcija, kao i na sam rad algoritma.

Parametri koji se najčeće koriste prilikom podešavanja *Random forest* algoritma su:

1. Broj stabla odlučivanja – algoritmu se prosleđuje prirodan broj koji označava broj stabla koji će se generisati za vršenje predikcije. Što je broj stabla veći, to su predikcije stabilnije i preciznije. Samim tim se usporava kalkulacija rezultata. Obično se prosleđuje kao „*n_estimators*“ parametar.
2. Maksimalan broj odluka - algoritmu se prosleđuje prirodan broj koji označava maksimalan broj odluka koje algoritam može da ima po jednom stablu. Prosleđuje se kao „*max_features*“ parametar.
3. Minimalan broj listova – moguće je podesiti minimalan broj listova koliko stablo može da ima po jednom čvoru odluke. Za setovanje ove funkcionalnosti koristi se parametar „*min_sample_leaf*“.
4. Broj jezgara procesora – setovanjem parametra „*n_jobs*“ moguće je podesiti broj jezgara procesora koje će koristiti algoritam. Time se znatno povećava brzina njegovog izvršavanja jer omogućava rad u paraleli.
5. Setovanjem parametra „*oob_score*“ na *true* omogućava algoritmu da validira svoje učenje. Podaci nad kojim se vrši treniranje se dele na dva dela. Prvi deo služi za učenje, što predstavlja dve trećine celog dataset-a, a ostatak služi za validaciju predikcija učenja. Pre same podele, podaci se međusobno izmešaju kako bi se izbegle pravilnosti njihovog rasporeda.

3.3.4. Prednosti i mane algoritma

Glavna prednost algoritma učenja u obliku slučajnih šuma je ta što se može iskoristiti i za klasifikacione i za regresivne probleme. Takođe se pokazao kao relativno lak za korišćenje pomoću podešivih parametara na osnovu kojih je moguće poboljšati predikcije. Njihov broj nije veliki i relativno su laki za razumevanje.

Jedan od najčešćih problema sa kojim se susreću algoritmi mašinskog učenja je *overfitting*, što se retko dešava korišćenjem *Random forest* algoritma. Kako postoje i više nego dovoljno stabla odlučivanja u samom algoritmu, time se sprečava pojava samog problema.

Korišćenjem velikog broja stabla odlučivanja može dovesti do usporavanja rada algoritma, što predstavlja problem u situacijama kada se koristi za predikcije u realnom vremenu. Generalno, ovi algoritmi su brzi za učenje, ali spori za kreiranje predikcije posle učenja. Za preciznije predikcije zahteva se više stabla odlučivanja, što rezultira sporijem modelu. U mnogim današnjim aplikacijama je *Random forest* algoritam dovoljno brz, ali postoje situacije gde su performanse u realnom vremenu bitne, pa samim tim bi drugi pristupi problemu bili poželjniji. [16]

3.3.5. Primena algoritma u svakodnevnom životu

Danas se algoritam učenja u obliku slučajnih šuma koristi u mnogim različitim oblastima rada. U bankarstvu bi se mogla koristi kao detekcija klijenata koji koriste njene usluge češće nego ostali i dati im određene pogodnosti. Pored toga, koristi se i prilikom detekcija prevari klijenata koji žele da prevare samu banku. Takođe, može se primeniti prilikom učenja o ponašanju cena na berzi i davati predikciju ponašanja u budućnosti.

U medicinskom domenu se koristi prilikom utvrđivanja korektne kombinacije medicinskih komponenata i za analizu pacijentskih dijagnoza u prošlosti kako bi utvrdili bolest. [16]

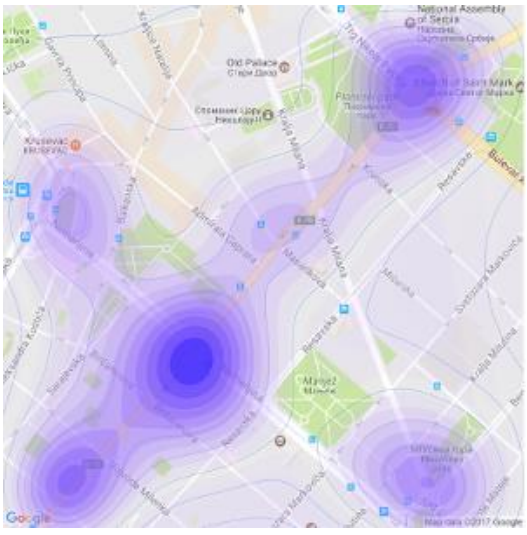
Kako broj korisnika na internetu raste, tako je i potreba za analizom njihovih interesovanja i radnji sve veća. Korišćenjem *Random forest* algoritma, može se utvrditi interesovanje korisnika na osnovu njegovih akcija i posećenih stranica na internetu, što pomaže *E-commerce* portalima prilikom reklamiranja njihovih produkta targetnoj grupi ljudi.

4. PROBLEM SAOBRAĆAJNIH NESREĆA

4.1. OPIS PROBLEMA

Naglim savremenim razvojem tehnike, tehnologije, informatike kao i ostalih nauka teži se svakodnevnom povećavanju životnog standarda, što sa jedne strane uslovljava povećavanje potreba za automobilima tj. njihovim posedovanjem, kao i posedovanjem dozvola (licenca) za upravljanje istih. Ogromno povećavanje potrebe za automobilima na drumovima širom sveta u poslednjem periodu pored pozitivnih posledica, prate i broj negativnih posledica. Vozač, vozilo i put predstavljaju jedne od osnovnih faktora bezbednosti saobraćaja. Udeo ovih faktora u bezbednosti saobraćaja je veoma veliki i on prati tendenciju porasta saobraćajnih nezgoda na našim putevima, što nam ukazuje na njen stalni rast i istovremeno da se ovim faktorima treba posvetiti što veća pažnja. [17]

Kao adekvatan primer primene algoritama mašinskog učenja, tema ovog rada bi bila predikcija lokacija saobraćajnih nezgoda na teritoriji grada Beograda. Kako je vremenom gužva u saobraćaju sve veća, rast nezgoda je neminovan. Na osnovu podataka koji su javno dostupni svima na Portalu otvorenih podataka Republike Srbije (https://data.gov.rs/sr/) mogu se preuzeti podaci korišćeni za izradu ovog rada. Koristeći ulazne parametre iz podataka o saobraćajnim nezgodama za 2015. 2016. 2017. i delom za 2018. godinu na teritoriji grada Beograda, neophodno je izvršiti predikcije koordinata saobraćajnih nesreća određenog tipa gde bi one mogle biti u određenom vremenskom trenutku i u određenim klimatskim uslovima.



***Slika 4.1**: Grafički prikaz nlokacija nezgoda na teritoriji grada Beograda. [18]*

4.2. PRIKUPLJANJE PODATAKA

U svakom projektu koji se bavi mašinskim učenjem, filtriranje i čišćenje podataka je uvek kritičan prvi korak pre nego što se krene sa korišćenjem algoritma. Pre svega je neophodno razumeti ceo skup podataka i značenje svakog polja, kako bi mogli korektno da ih iskoristimo u samim algoritmima učenja.

Na sajtu Portala otvorenih podataka, postoje javni podaci o saobraćajnim nezgodama koji su dostupni svima i koji su iskorišćeni za izradu ovog rada. U trenutku njihovog preuzimanja, na sajtu su bila dostupna četiri dataset fajla, saobraćajne nezgode na teritoriji Beograda u godinama 2015, 2016, 2017 i delimično 2018. godine (do kraja Juna). Prilikom njihovog skidanja, dobijeni fajlovi su bili ODS formata (*Open Document Spreadsheet*) što ih čini nepogodnim za rad nad njima. Korišćenjem Excel alata, fajlovi koji sadrže podatke za učenje su konvertovani u CSV fajl, što ih čini mnogo pogodnijim za radi i samo čitanje.

Struktura svakog fajla nije bila ista. Različit raspored kolona, kao i viška nekih u nekim fajlovima je trebalo prilagoditi kako bi se mogli gurpisati zajedno u jedan fajl. Kako su skupovi veliki, njihovo sređivanje je zahtevalo pisanje dodatnog koda. Zbog dostupnih biblioteka za čitanje i obradu podataka iz CSV fajla, za njihovo sređivanje se koristio projekat pisan u programskog jezika Java.

4.2.1. Kolone dataset-a

Iako im se ne poklapaju sve kolone, sva četiri dataset-a imaju ovih 7 zajedničkih kolona koje opisuju saobraćajnu nesreću:

ID nezgode	Datum i vreme	Longitude	Latitude	Nezgoda	Vrsta nezgode	Opis nezgode
1166606	01.01.2017,22:30	20.426063	44.792292	Sa mat.stetom	SN SA JEDNIM VOZILOM	Nezgode sa učešćem jednog vozila i preprekama na ili iznad kolovoza
1162877	01.01.2017,22:30	20.47217	44.79319	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću istim putem u suprotnim smerovima uz skre
1162787	01.01.2017,21:20	20.38684	44.82062	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – sudar pri uporednoj vožnji
1162726	01.01.2017,04:00	20.446809	44.822215	Sa mat.stetom	SN SA JEDNIM VOZILOM	Nezgoda sa jednim vozilom – silazak ulevo sa kolovoza na pravcu
1162760	01.01.2017,10:00	20.46225	44.82702	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Sudar sa parkiranim vozilom sa desne strane kolovoza
1162695	01.01.2017,01:10	20.47742	44.77239	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – sustizanje
1162918	01.01.2017,03:50	20.47681	44.75562	Sa povredjenim	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila – čeoni sudar
1163921	01.01.2017,11:00	20.46414	44.82877	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Ostali sudari sa parkiranim vozilom
1162765	01.01.2017,00:30	20.3764	44.4478	Sa mat.stetom	SN SA JEDNIM VOZILOM	Nezgoda sa jednim vozilom – silazak sa kolovoza u krivini
1164578	01.01.2017,04:00	20.44094	44.81437	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću istim putem u istom smeru uz skretanje, skr
1162764	01.01.2017,09:00	20.2596	44.3778	Sa mat.stetom	SN SA JEDNIM VOZILOM	Nezgoda sa jednim vozilom – silazak udesno sa kolovoza na pravcu
1162757	01.01.2017,10:40	20.4662512	44.789368	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – sustizanje
1162965	01.01.2017,04:20	20.47325	44.80637	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – sustizanje
1162729	01.01.2017,04:30	20.379359	44.849343	Sa mat.stetom	SN SA JEDNIM VOZILOM	Nezgoda sa jednim vozilom – silazak ulevo sa kolovoza na pravcu
1162731	01.01.2017,06:45	20.378596	44.843408	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – preticanje
1162734	01.01.2017,07:10	20.427782	44.764413	Sa povredjenim	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Ostali sudari sa najmanje dva vozila koja se kreću u istom smeru – ostalo
1164196	01.01.2017,02:00	20.29401	44.89601	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Sudar sa parkiranim vozilom sa leve strane kolovoza
1162786	01.01.2017,17:15	20.51824	44.84749	Sa povredjenim	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Ostale nezgode sa najmanje dva vozila – suprotni smerovi bez skretanja
1164151	01.01.2017,04:00	20.370789	44.847432	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću različitim putevima uz prolazak kroz raskrsni
1162742	01.01.2017,08:30	20.46737	44.80353	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Sudar sa parkiranim vozilom sa desne strane kolovoza
1162753	01.01.2017,17:59	20.47994	44.79011	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću istim putem u istom smeru uz skretanje, skr
1162758	01.01.2017,19:00	20.465962	44.716841	Sa povredjenim	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – sustizanje
1162956	01.01.2017,04:35	20.47683	44.81715	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću različitim putevima uz prolazak kroz raskrsni
1162921	01.01.2017,02:30	20.507894	44.823248	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – sudar pri uporednoj vožnji
1162778	01.01.2017,16:30	20.45944	44.80632	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću istim putem u istom smeru uz skretanje, pol
1162808	01.01.2017,22:55	20.45242	44.79928	Sa mat.stetom	SN SA JEDNIM VOZILOM	Nezgoda sa jednim vozilom na kolovozu
1163617	01.01.2017,03:45	20.47057	44.87312	Sa povredjenim	SN SA JEDNIM VOZILOM	Nezgoda sa jednim vozilom i prevrtanjem
1162745	01.01.2017,06:00	20.48062	44.80318	Sa povredjenim	SN SA PEŠACIMA	Prelazak pešaka zdesna, van raskrsnice, bez skretanja vozila
1162754	01.01.2017,20:30	20.46676	44.82002	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Ostali sudari sa parkiranim vozilom
1162879	01.01.2017,12:35	20.49373	44.79789	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Sudar sa parkiranim vozilom sa desne strane kolovoza
1162881	01.01.2017,15:00	20.52695	44.79313	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Sudar sa parkiranim vozilom sa desne strane kolovoza
1162724	01.01.2017,08:00	20.39123	44.82556	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK	Najmanje dva vozila koja se kreću različitim putevima uz prolazak kroz raskrsni
1162905	02.01.2017,22:45	20.39828	44.82133	Sa mat.stetom	SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA	Najmanje dva vozila koja se kreću u istom smeru – uključivanje u saobraćaj
1162889	02.01.2017,19:30	20.45762	44.75081	Sa mat.stetom	SN SA PARKIRANIM VOZILIMA	Sudar sa parkiranim vozilom sa leve strane kolovoza

***Slika 4.2.1**: Prikaz podataka o saobraćajnim nesrećama na teritoriji grada Beograda u toku 2017. godine*

1. ID nezgode – interni identifikator saobraćajne nesreće koja se desila;
2. Datum i vreme – tačno vreme kada se saobraćajna nesreća dogodila;
3. Longitude i latitude – tačne koordinate mesta saobraćajne nesreće (realni brojevi);
4. Nezgoda – naslov same nezgode. Vrednost ove kolone može;
    - Sa mat. stet;
    - Sa povredjenim;
    - Sa poginulim.
5. Vrsta nezgode – vrsta nezgode. Klasifikuje se u pet kategorija:
    - SN SA JEDNIM VOZILOM;
    - SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA;
    - SN SA NAJMANjE DVA VOZILA – SKRETANje ILI PRELAZAK;
    - SN SA PARKIRANIM VOZILOM;
    - SN SA PEŠACIMA.
6. Opis nezgode – kratak opis saobraćajne nesreće koja se dogodila.

4.2.2. Klimatski parametri

Kao što se može primetiti, u preuzetim dataset-ovima ne postoje parametri koji pokazaju klimatsko stanje u trenutku svake saobraćajne nesreće. S obzirom da su neophodni uslovi pod kojim se desila nezgoda kao ulazni parametri mašinskog učenja, vrednosti vremenskih prilika u danima nezgode nisu mogli da se nađu na istom portalu gde se nalaze i dataset-ovi.

S obzirom da je neophodno naći meteorološka stanja baš u trenucima saobraćajnih nezgoda, potrebno je naći portal koji može da vrati takve informacije. Idealan kandidat za to bio je *Time and Date* portal (https://www.timeanddate.com/). Na njihovom sajtu je moguće preuzeti tačna meteorološka stanja u različitim delovima dana. Koristeći njihove adekvatne HTTP zahteve, kao odgovor se dobija HTML stranica u kojoj se nalazi JSON vrednost klimatskih promena.

Nažalost, mogućnost za dovlačenjem traženih parametara nije moguće unošenjem konkretnog datuma i vremena. Jedini način koji je bio omogućen je dovlačenje JSON vrednosti svih klimatskih promena u jednom mesecu. Sortiranjem svih dataset-ova po datumu i vremenu, zaključuje se da je prva nezgoda bila u februaru 2014. godine (postoje zalutali podaci iz ranijih godina unutar dataset-a za 2015. godinu), a poslednja u junu 2018. S obzirom da ima malo nezgoda iz 2014. godine, sa sajta je skinuto ukupno 47 JSON fajlova koji sadrže meteorološke

vrednosti. Svaki fajl sadrži vrednosti za svaki dan odereñenog meseca, u određenom kvartalu dana.


```
{
  "units": {
    "temp": "°C",
    "prec": "mm",
    "wind": "km/h",
    "baro": "mbar"
  },
  "detail": [
    {
      "hl": true,
      "hls": "Fri, 1 Jun",
      "hlsh": "1 Jun",
      "date": 1527811200000,
      "ts": "00:00",
      "ds": "Friday, 1 June 2018, 00:00 — 06:00",
      "icon": 13,
      "desc": "Clear.",
      "temp": 20,
      "templow": 18,
      "baro": 1015,
      "wind": 3,
      "wd": 150,
      "hum": 62
    },
    {
      "date": 1527832800000,
      "ts": "06:00",
      "ds": "Friday, 1 June 2018, 06:00 — 12:00",
      "icon": 2,
      "desc": "Scattered clouds.",
      "temp": 29,
      "templow": 23,
      "baro": 1016,
      "wind": 2,
      "wd": 60,
      "hum": 50
    },
    {
      "date": 1527854400000,
      "ts": "12:00",
      "ds": "Friday, 1 June 2018, 12:00 — 18:00",
```


Slika 4.2.2: *Primer JSON fajla sa meteorološkim parametrima za 1. Jun 2018. Godine*

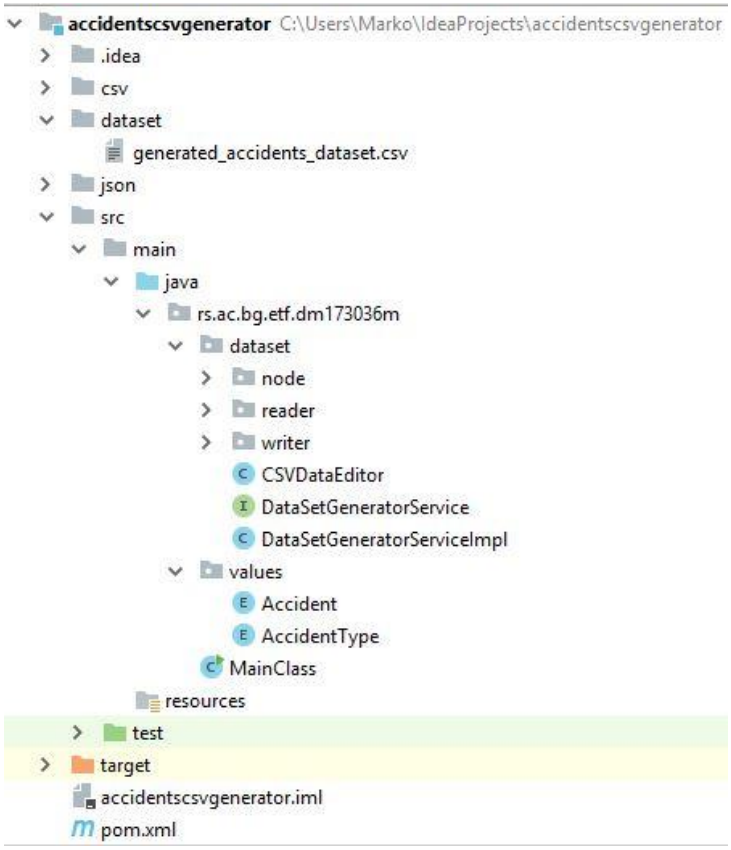
Parametri koji se uzimaju za mašinsko učenje su temperatura (temp), najniža temperatura

(templow), vazdušni pritisak (baro), brzina vetra (wind) i vlažnost vazduha (hum). Jedinice koje se koriste su naznačene na vrhu JSON fajla kao prvi element, ali nisu od značaja za mašinsko učenje i vršenje predikcije.

4.3. GENERISANJE SREĐENOG DATASET FAJLA

Kao što je napomenuto ranije, broj i raspored kolona se ne podudaraju u sva četiri fajla dataset-a kompletno, već sa malim razlikama. Ako se sagledaju sva četiri fajla, može se zaključiti da im većina kolona ima isto značenje, samo su na pogrešnim mestima u tabelama. Takođe neophodno je svakom podatku dodeliti meteorološke vrednosti u tom trenutku, kako bi uslovi pod kojim se desila nezgoda bili potpuni.

Za kreiranje sređenog dataset-a spremnog za mašinsko učenje, kreiran je novi Javin projekat pod nazivom *Accidents CSV generator*. Kreiran je korišćenjem *Maven build* alata pomoću kojeg je mnogo lakše uvoženje eksternih pomoćnih biblioteka za rad. Pored toga što kreira CSV fajl, ovaj projekat služi i za obradu inicijalnih podataka iz fajlova preuzetih sa Portala otvorenih podataka i njihovog spajanja sa meteorološkim vrednostima iz JSON fajlova.



***Slika 4.3**: Primer JSON fajla sa meteorološkim parametrima za 1. Jun 2018. godine*

4.3.1. Skeniranje vrednosti iz fajlova

Svi CSV fajlovi se nalaze unutar csv foldera u projektu. Kako bi se dobio krajnji CSV fajl sa sređenim vrednostima, neophodno je pročitati svaki red iz svih CSV fajlova unutar csv foldera. Svaki red u fajlu sadrži informacije o jednoj saobraćajnoj nesreći. Pošto su u pitanju CSV fajlovi, postoje određene biblioteke koje omogućavaju lako njihovo čitanje. U ovom projektu se koristi biblioteka *OpenCSV* koja je uvezena kao *dependency* naznačen unutar *pom.xml* fajla *Maven* alata.

S obzirom na to da je svaki CSV fajl drugačiji, čitanje svakog od njih je drugačije. Za svaki od njih kreirana je nova klasa koja čita iz njih vrednosti na poseban način. Svaka ta klasa implementira interfejs *CSVDataset* sa jednom metodom *readDataRows*. U implementaciji ove metode je definicija kako se čita CSV fajl za koji je ta klasa kreirana.

```
@Override
public List<CSVDataNode> readDataRows(List<CSVDataNode> dataNodes) {

    List<CSVDataNode> resultDataNodes = dataNodes;
    if (resultDataNodes == null) {
        resultDataNodes = new ArrayList<>();
    }

    try {
        String[] nextLine;

        System.out.println("Started reading " + ACCIDENTS_2018_URL + "file.");
        Date readingStarted = new Date();

        while ((nextLine = reader.readNext()) != null) {

            List<String> lineAsList = new ArrayList<>(Arrays.asList(nextLine));
            String accident = lineAsList.get(4);
            String dateTime = lineAsList.get(1);
            double longitude = Double.valueOf(lineAsList.get(2));
            double latitude = Double.valueOf(lineAsList.get(3));
            String accidentType = lineAsList.get(5);
            String accidentDescription = lineAsList.get(6);

            CSVDataNode readNode = new CSVDataNode(accident, dateTime, longitude,
                    latitude, accidentType, accidentDescription);

            if (!resultDataNodes.contains(readNode)) {
                resultDataNodes.add(readNode);
            }
        }

        Date readingFinished = new Date();
        System.out.println("Finished with reading " +
                ACCIDENTS_2018_URL + "file. Time: " +
                ((readingFinished.getTime() - readingStarted.getTime()) / 1000) + "s.");
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return resultDataNodes;
}
```

***Slika 4.3.1.1**: Implementacija metode readDataRows u klasi koja skenira CSV fajl sa podacima saobraćajnih nesreći 2018. godine.*

Pored ovih fajlova, neophodno je pročitati i JSON fajlove u kojima se nalaze meteorološke vrednosti. Svi pomenuti fajlovi se nalaze unutar json foldera u projektu i odatle se čitaju. Za olakšano čitanje JSON fajlova korišćena je biblioteka pod nazivom *JSONSimple*. Ona je takođe uvezena kao *Maven dependency* u projekat.

```
public List<WeatherNode> readWeatherFromDirectory() {

    List<WeatherNode> weatherNodes = new ArrayList<>();

    System.out.println("Started reading weathers from json files.");
    Date readingStarted = new Date();

    for (File subFile : weatherJsonsDirectory.listFiles()) {
        weatherNodes.addAll(readWeatherFromFile(subFile.getPath()));
    }

    Date readingFinished = new Date();
    System.out.println("Finished with reading weather json files. Time: " +
            ((readingFinished.getTime() - readingStarted.getTime()) / 1000) + "s.");

    return weatherNodes;
}

private List<WeatherNode> readWeatherFromFile(String jsonFilePath) {

    List<WeatherNode> weatherNodes = new ArrayList<>();
    JSONParser parser = new JSONParser();

    try {

        Object obj = parser.parse(new FileReader(jsonFilePath));
        JSONObject jsonObject = (JSONObject) obj;

        // loop array
        JSONArray msg = (JSONArray) jsonObject.get("detail");
        Iterator iterator = msg.iterator();
        while (iterator.hasNext()) {
            weatherNodes.add(WeatherNode.generateWeatherNode((JSONObject) iterator.next()));
        }
    } catch (ParseException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return weatherNodes;
}
```

***Slika 4.3.1.2**: Implementacija metode za čitanje JSON fajla sa meteorološkim vrednostima*

Klasa u kojoj se nalazi implementacija metode čitanja iz JSON fajla (*readWeatherFromDirectory*) se zove *WeatherJsonScanner*. Za razliku od više implementiranih klasa kao kod čitača CSV fajlova, ovde postoji samo jedna jer svi JSON fajlovi imaju istu strukturu.

Vrednosti koje su skenirane iz CSV i JSON fajlova se smeštaju u odgovarajuće klase. Vrednosti saobraćajnih nesreća se smeštaju u listu *CSVDataNode* objekata u kojima se nalaze osnovne informacije o jednoj saobraćajnoj nesreći.

```
public class CSVDataNode {

    private static final DateFormat formatDate = new SimpleDateFormat( pattern: "dd.MM.yyyy,HH:mm", Locale.ENGLISH);
    private static final DateFormat formatDateForPrintingDataset = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");

    private Accident accident;
    private Date dateTime;
    private String dateTimeValue;
    private double longitude;
    private double latitude;
    private AccidentType accidentType;
    private String accidentDescription;

    private String id;
```

***Slika 4.3.1.3**: Polja CSVDataNode klase.*

Vrednosti iz JSON fajlova se smeštaju u listu *WeatherNode* objekata, gde se nalaze osnovne informacije o meteorološkim vrednostima jednog kvartala dana.

```
public class WeatherNode {

    private static final DateFormat dateFormater = new SimpleDateFormat( pattern: "dd-MM-yyyy");

    private Date date;
    private Date startTime;
    private Date endTime;
    private String day;
    private Double temp;
    private Double tempLow;
    private Double baro;
    private Double wind;
    private Double hum;
```

***Slika 4.3.1.4**: Polja WeatherNode klase.*

Prilikom čitanja svake saobraćajne nezgode iz CSV fajlova, proverava se da li već postoji ista takva u listi već uneta pre. To se postiže tako što je *equals* metoda klase *CSVDataNode* pregažena novom definicijom gde se, osim reference objekata, proveravaju njihovi identifikatori da li su jednaki. Identifikator svakog *CSVDataNode* objekta se sastoji od svih vrednosti njegovih polja. S obzirom da će izgenerisani CSV fajl biti korišćen i za učenje i za testni deo, neophodno je izbegavati duplirane vrednosti u dataset-u kako se preciznost predikcija ne bi poremetila.

Na kraju čitanja svih podataka iz CSV fajlova u listu *CSVDataNode* objekata, raspored elemenata je sortiran po datumu jer su tako sortirani u samim fajlovima. Zbog toga se nakon čitanja, mešaju elementi liste pozivom metode *Collections.shuffle*, kako bi ispis tih podataka bio nasumičan i pravilan za algoritme mašinskog učenja.

4.3.2. Sređivanje i pisanje vrednosti u CSV fajl

Čitanjem svih dokumenata vezanih za saobraćajne nesreće i meteorološke prilike, vrednosti su sačuvane unutar listi objekata *CSVDataNode* i *WeatherNode*. Nakon toga je neophodno te vrednosti ispisati u novokreirani izlazni CSV fajl koji će odabrani algoritmi mašinskog učenja koristiti.

Međutim, prilikom pisanja u novi fajl, neophodno je voditi računa kako predstavljamo ulazne vrednosti dataset-a koji će algoritmi koristiti. Sa njihovo korišćenje će se koristiti algoritmi iz Python biblioteke *Scikit-learn* za mašinsko učenje. S obzirom da oni za ulazne vrednosti primaju samo vrednosti tipa realnih brojeva, neophodno je srediti ulazne podatke koji su tipa *Datetime* i *String*.

Pored toga, neophodno je naći za svaku saobraćajnu nesreću iz liste *CSVDataNode* objekata odgovarajuće meteorološke parametre za tačno vreme kada se dogodilo. Kako se ne bi usložnjavala struktura, sve ove obrade će se odvijati u toku generisanja nove linije izlaznog fajla.

4.3.2.1. IZBACIVANJE NEPOTREBNIH ATRIBUTA

S obzirom da su samo pojedini atributi (kolone) ključni za predikciju koordinata lokacija saobraćajnih nezgoda, u novogenerisanom dataset fajlu nije obavezno čuvati ostale atribute koji nisu od značaja. Čak šta više, preporučuje se njihovo izbacivanje jer time znatno smanjuje njegovu veličinu što ga čini lakšim za čitanje prilikom prikupljanja podataka za učenje.

Za samu predikciju, najbitniji su nam sledeći parametri saobraćajne nesreće:

- Temperatura;
- Vazdušni pritisak;
- Brzina vetra;
- Vlažnost vazduha;
- Datum i vreme nezgode;
- Naslov nezgode;
- Tip nezgode;
- Geografska širina;
- Geografska dužina.

Svi ostali parametri, kao što su interni identifikator, opis, ikona, dan u nedelji, najniža temperature i ostali, mogu se zanemariti jer se ne uzimaju u razmatranje prilikom mašinskog učenja. Samim tim, oni se neće upisati u novi izlazni izgenerisani CSV fajl.

4.3.2.2. REFAKTORISANJE DATUMA I VREMENA

Kako regresivni algoritmi *Scikit* biblioteke primaju i generišu samo realne brojeve, neophodno mu je priložiti i takav dataset. Zbog toga, obavezne ulazne vrednosti datum i vreme ne mogu se priložiti u izvornom formatu.

Način koji se preporučuje za ovakve situacije je transformacija zvana *1-of-K encoding*. Svaka vrednost datuma se predstavlja u više kolona sa vrednostima 0 ili 1. Za svaki element datuma i vremena postoji kolona. U tabeli postoje kolone 5 kolona za godine (2014, 2015, 2016, 2017 i 2018), 12 kolona za mesece (od Januara do Decembra), 31 kolona za dane u mesecu, 24 časa u danu i 60 minuta u satu, što sve ukupno čini 132 kolone. Na primer, za datum 10.8.2017. i vreme 20:00, u kolonama 2017, august, 10_day, 20_hour i 0_min bi bile setovane vrednosti 1, dok u ostalim vrednost 0.

```
private List<String> writeSplittedDate(Date timeRead) {

    List<String> splittedDateList = new ArrayList<>();

    Calendar calendar = Calendar.getInstance();
    calendar.setTime(timeRead);
    int month = calendar.get(Calendar.MONTH);
    int day = calendar.get(Calendar.DAY_OF_MONTH);
    int hour = calendar.get(Calendar.HOUR_OF_DAY);
    int minute = calendar.get(Calendar.MINUTE);

    boolean foundMonth = false;
    for (int i = 0; i < 12; i++) {
        if (i == month) {
            splittedDateList.add(String.valueOf(1));
            foundMonth = true;
        } else {
            splittedDateList.add(String.valueOf(0));
        }
    }

    if (!foundMonth) {
        throw new RuntimeException("Month not found in loop while generating dataset row. Month: " +
                month + "; Date: " + new SimpleDateFormat( pattern: "dd-MM-yyyy HH:mm").format(timeRead));
    }
```

***Slika 4.3.2.2.1**: Početak metode ispisa enkodovanog datuma.*

S obzirom da bi time imali mnogo kolona i fajl bi bio previše veliki, delovi datuma su generalizovani. Kolone sa godinama su izbačene iz razloga što za učenje je bitno u kom delu godine se desila nesreća, nevezano za nju. Takođe, umesto kolona za vreme (časove i minute), uvedene su samo 4 kolone (prva, druga, treća i četvrta četvrtina dana) koje se setuju u zavisnosti

od dela dana. Za primer od malopre, gde je vreme bilo bilo 20:00, samo 4. kolona bi bila setovana sa vrednošću 1, dok ostale 3 sa 0. Iako tačno vreme znači prilikom učenja algoritma, dataset je jako mali da bi tako detaljni parametri uticali na razrešavanje dilema. Umesto toga, znatno smo smanjili broj kolona (umesto 132, sada ima 47 kolona), a predikcije su znatno preciznije.

```
boolean foundQuarter = false;
for (int i = 0; i < 24; i += 6) {
    if (hour >= i && hour < i + 6) {
        splittedDateList.add(String.valueOf(1));
        foundQuarter = true;
    } else {
        splittedDateList.add(String.valueOf(0));
    }
}

if (!foundQuarter) {
    throw new RuntimeException("Quarter not found in loop while generating dataset row. Hour: " +
            hour + ", minute: " + minute + "; Date: " + new SimpleDateFormat( pattern: "dd-MM-yyyy HH:mm").format(timeRead));
}
```

***Slika 4.3.2.2.2**: Deo metode ispisa enkodovanog datuma za generisanje kvartala dana*.

4.3.2.3. REFAKTORISANJE NASLOVA I VRSTE NESREĆE

Slično kao i sa datumom, vrednosti za naslov i vrstu nesreće je takođe neophodno refaktorisati korišćenjem *1-of-K encoding* metode. Vrednosti naslova nesreće mogu biti „Sa mat.stetom“, „Sa poginulim“ i „Sa povredjenim“, što je ukupno 3 kolone.

```
private List<String> writeSplittedAccident(Accident accident) {

    List<String> splittedAccidentList = new ArrayList<>();

    if (accident.equals(Accident.SA_MAT_STETOM)) {
        splittedAccidentList.add(String.valueOf(1));
    } else {
        splittedAccidentList.add(String.valueOf(0));
    }

    if (accident.equals(Accident.SA_POGINULIM)) {
        splittedAccidentList.add(String.valueOf(1));
    } else {
        splittedAccidentList.add(String.valueOf(0));
    }

    if (accident.equals(Accident.SA_POVREDJENIM)) {
        splittedAccidentList.add(String.valueOf(1));
    } else {
        splittedAccidentList.add(String.valueOf(0));
    }

    return splittedAccidentList;
}
```

***Slika 4.3.2.3.1**: Metoda za ispisivanje 0 i 1 vrednosti u kolone za naslov*

Što se vrste nesreći tiče, njihove vrednosti mogu biti „SN SA JEDNIM VOZILOM“, „SN SA NAJMANjE DVA VOZILA – BEZ SKRETANjA“, „SN SA NAJMANjE DVA VOZILA – SKRETANjE ILI PRELAZAK“, „SN SA PARKIRANIM VOZILOM“, „SN SA PEŠACIMA“ i „“ (prazna kolona, nedefinisana), što je ukupno 6 kolona.

```
private List<String> writeSplittedAccidentType(AccidentType accidentType) {

    List<String> splittedAccidentTypeList = new ArrayList<>();
    boolean found = false;

    if (accidentType.equals(AccidentType.SN_SA_JEDNIM_VOZILOM)) {
        splittedAccidentTypeList.add(String.valueOf(1));
        found = true;
    } else {
        splittedAccidentTypeList.add(String.valueOf(0));
    }

    if (accidentType.equals(AccidentType.SN_SA_NAJMANJE_DVA_VOZILA_BEZ_SKRETANJA)) {
        splittedAccidentTypeList.add(String.valueOf(1));
        found = true;
    } else {
        splittedAccidentTypeList.add(String.valueOf(0));
    }

    if (accidentType.equals(AccidentType.SN_SA_NAJMANJE_DVA_VOZILA_SKRETANJE_ILI_PRELAZAK)) {
        splittedAccidentTypeList.add(String.valueOf(1));
        found = true;
    } else {
        splittedAccidentTypeList.add(String.valueOf(0));
    }

    if (accidentType.equals(AccidentType.SN_SA_PARKIRANIM_VOZILIMA)) {
        splittedAccidentTypeList.add(String.valueOf(1));
        found = true;
    } else {
        splittedAccidentTypeList.add(String.valueOf(0));
    }

    if (accidentType.equals(AccidentType.SN_SA_PESACIMA)) {
        splittedAccidentTypeList.add(String.valueOf(1));
        found = true;
    } else {
        splittedAccidentTypeList.add(String.valueOf(0));
    }

    if (accidentType.equals(AccidentType.DEFAULT) || !found) {
        splittedAccidentTypeList.add(String.valueOf(1));
    } else {
        splittedAccidentTypeList.add(String.valueOf(0));
    }

    return splittedAccidentTypeList;
}
```

***Slika 4.3.2.3.2**: Metoda za ispisivanje 0 i 1 vrednosti u kolone za vrste*

4.3.2.4. PISANJE IZLAZNIH VREDNOSTI

Nakon obrade vrednosti datuma, vremena, naslova i vrste nesreće, ne preostaje više ništa nego da se vrednosti ispišu u novokreirani CSV fajl. Prvo će se ispisivati vrednosti meteoroloških prilika u trenutku nesreće (temperatura, vazdušni pritisak, brzina vetra i vlažnost vazduha), zatim kolone koje predstavljaju enkodirane vrednosti datuma, naslova nesreće i vrste nesreće i na samom kraju vrednosti koordinata u 2 kolone. (geografska širina i dužina), što čini sve ukupno 62 kolone.

Izlazni fajl se kreira u folderu *dataset* projekta pod nazivom *generated_accidents_dataset*. Fajl ima ukupno 55824 redova vrednosti koje će koristiti algoritmi mašinskog učenja za predikciju lokacija saobraćajnih nezgoda.

4.4. PREDIKCIJE ALGORITAMA MAŠINSKOG UČENJA

U folderu *dataset* projekta *Accidents CSV Generator* se nalazi pripremljeni dataset koji je izgenerisan iz više koraka pod nazivom *generated_accidents_dataset.csv*. Taj fajl se koristi u novom projektu koji se zove *Accidents Predictions*. Smešta se u istoimeni *dataset* folder unutar novog projekta.

Accidents Predictions projekat je baziran na programskom jeziku Python, uz korišćenje biblioteke *Scikit-learn*. Kako je cilj ovog rada da se predvide koordinate lokacija gde bi mogle biti saobraćajne nesreće na osnovu ulaznih brojčanih parametara, neophodno je koristiti regresivne algoritme nadgledanog mašinskog učenja. Od velikog izbora takvih algoritama, za potrebe ovog rada su iskorišćena 3 najpoznatija: linearna regresija, stablo odlučivanja i algoritam na osnovu nasumičnih stabla.

4.4.1. Priprema podataka za učenje

Prva faza koja je svim algoritmima zajednička je čitanje dataset-a iz fajla i razdvajanje ulaznih i izlaznih podataka (vertikalna podela). Pored toga, neophodno je razdvojiti podatke tako da jedan, veći deo bude za treniranje samog algoritma, a drugi, manji za testiranje izvršenog učenja (horizontalna podela). U suštini, za taj manji deo će se vršiti predikcija, kako bi mogli odmah nakon toga da uporedimo te vrednosti sa pravim vrednostima i izračunamo prosečno odstupanje od tačne vrednosti. Za odstupanje se smatra rastojanje između dve tačke (izračunate i tačne) definisane koordinatima u kilometrima.

```
# Import data from dataset
importedPre = readValuesFromFileLeft()
importedPost = readValuesFromFileRight()

expectedPre = importedPre[-numOfLastRowsForTest:]
expectedPost = importedPost[-numOfLastRowsForTest:]

importedPre = importedPre[:-numOfLastRowsForTest]
importedPost = importedPost[:-numOfLastRowsForTest]
print('Data successfully imported.')

X_train, X_test, y_train, y_test = train_test_split(np.array(importedPre), np.array(importedPost),
                                                    test_size=testSizePercent)
```

***Slika 4.4.1.1**: Čitanje i sređivanje podataka iz dataset-a*

Kako ceo dataset ima 55824 saobraćajnih nezgoda, za učenje će se iskoristiti 54824 redova, dok preostalih 1000 će biti iskorišćeno za testiranje predikcija. Na slici 4.4.1.1 je prikazan isečak koda koji čita vrednosti iz dataset-a i deli ih horizontalno i vertikalno. Na početku se promenljivama dodeljuju dvodimenzionalni nizovi podataka, *importedPre* promenljivi niz sa 55824 redova sa 60 kolona (sve kolone osim koordinata), a *importedPost* niz od 55824 redova sa 2 kolone (samo koordinate lokacija saobraćajnih nezgoda). Varijablama *expectedPre* i *expectedPost* se dodeljuju elementi *importedPre* i *importedPost* nizova, tačnije poslednjih 1000 redova (vrednost *numOfLastRowsForTest* je 1000).

Nakon redefinisanja *importedPre* i *importedPost* varijabli, definišu se 4 varijable, *X_train*, *X_test*, *y_train* i *y_test*. Ove varijable predstavljaju delove podataka nad kojima će se izvršiti trening regresivnih algoritama. Varijabla *testSizePercent* ima vrednost 0.1 što znači da će se za učenje uzeti 90% ulaznih podataka, a poslednjih 10% služe za validaciju treninga. Nakon izračunavanja ova četiri parametra, prosleđuju se svakom regresivnom algoritmu za učenje i predikciju.

4.4.2. Rezultati linearne regresije

Osim četiri parametara koje smo dobili u prethodnom koraku korišćenjem *train_test_split* metode, za validaciju predikcije neophodni su nam parametri *expectedPre* i *expectedPost*. Algoritam nakon istanciranja, kreće sa učenjem tako što poziva metodu *fit* i prosleđuje joj 2 parametra, *X_train* i *y_train*. Nakon završenog učenja, omogućena je provera treninga tako što će se testirati nad varijablama *X_test* i *y_test* pozivom metode *score*, ali u ovom slučaju nama to nije neophodno. Odmah posle toga, računa se predikcija i smešta se u promenljivu *prediction.* Kako bi se testirala predikcija, poziva se metoda *calculateAverageDistance* koja računa prosečnu grešku udaljenosti predikcionih tačaka od tačnih vrednosti, izraženu u kilometrima.

```
print("-----------------------")
print("Linear Regression")

start = timer()

clf = LinearRegression()
clf.fit(X_train, y_train)
prediction = clf.predict(np.array(expectedPre))

end = timer()

print('{:16s}: {:5.3f}{:2s}'.format("AVERAGE DISTANCE", calculateAverageDistance(prediction, np.array(expectedPost)),
                                    "km"))
print('{:13s}: {:5.3f}{:1s}'.format("Finished in: ", end - start, "s"))
```

***Slika 4.4.2.1**: Kod za predikciju koordinata saobraćajnih nesreća korišćenjem linearne regresije*

Kada se pokrene algoritam napisan u programskom jeziku Python (slika 4.4.2.1), dobijaju se sledeći rezultati:

```
Data import.
Data successfully imported.
-----------------------
Linear Regression
AVERAGE DISTANCE: 7.339km
Finished in: : 0.291s

Process finished with exit code 0
```

***Slika 4.4.2.2**: Rezultati linearne regresije*

Kao rezultat dobija se da je prosečna greška, izražena kao prosečno rastojanje između predikcionih i tačnih koordinata, 7.339 kilometara. Algoritam je završio zajedno sa treningom i predikcijom za manje od trećine sekunde, što je relativno brzo s obzirom na to da je radio sa 54824 podataka i 62 kolone tokom treninga i poslednjih 1000 podataka tokom predikcije.

4.4.3. Rezultati stabla odlučivanja

Priprema podataka za trening i predikciju je identična kao i priprema za linearnu regresiju. U suštini, ceo kod algoritma je identičan kao i kod linearne regresije, osim instanciranja regresora. U ovo slučaju, koristi se *DecisionTreeRegressor*, koji radi drugačije od lineranor regresora.

```
print("-----------------------")
print("Decision Tree Regressor")

start = timer()

clf = DecisionTreeRegressor()
clf.fit(X_train, y_train)
prediction = clf.predict(np.array(expectedPre))

end = timer()

print('{:16s}: {:5.3f}{:2s}'.format("AVERAGE DISTANCE", calculateAverageDistance(prediction, np.array(expectedPost)),
                                    "km"))
print('{:13s}: {:5.3f}{:1s}'.format("Finished in: ", end - start, "s"))
```

***Slika 4.4.3.1**: Kod za predikciju koordinata saobraćajnih nesreća korišćenjem regresora stabla odlučivanja*

Puštanjem algoritma sa slike 4.4.3.1 dobijaju se sledeći rezultati:

```
Data import.
Data successfully imported.
-----------------------
Decision Tree Regressor
AVERAGE DISTANCE: 9.220km
Finished in: : 0.653s

Process finished with exit code 0
```

***Slika 4.4.3.2**: Rezultati regresora stabla odlučivanja*

Korišćenjem ovog regresora, dobija se za prosečnu grešku 9.22 kilometara, dok je vreme njegovog izvršavanja čitavih 0.653 sekunde.

4.4.4. Rezultati Random forest regresora

Kod Random forest regresora se takođe podaci pripremaju na identičan način kao i kod prethodna dva opisana algoritma. Regresor se instancira pozivom konstruktora *RandomForestRegressor()*. Za početak se uzima slučaj kada su svi podrazumevani parametri usvojeni za rad *Random forest* regresora, a oni su:

- n_estimators = 10 (broj stabla odlučivanja)
- max_features = auto (maksimalan broj odluka)

- min_sample_leaf = 1 (minimalan broj listova)
- n_jobs = 1 (broj jezgara procesora)

```
print("-----------------------")
print("Random Forest Regressor")

start = timer()

clf = RandomForestRegressor()
clf.fit(X_train, y_train)
prediction = clf.predict(np.array(expectedPre))

end = timer()

print('{:16s}: {:5.3f}{:2s}'.format("AVERAGE_DISTANCE", calculateAverageDistance(prediction, np.array(expectedPost)),
                                    "km"))
print('{:13s}: {:5.3f}{:1s}'.format("Finished in: ", end - start, "s"))
```

***Slika 4.4.4.1**: Kod za predikciju koordinata saobraćajnih nesreća korišćenjem Random forest regresora*

Pokretanjem *Random forest* regresora sa podrazumevanim parametrima dobijaju se sledeći rezultati:

```
Data import.
Data successfully imported.
-----------------------
Random Forest Regressor
AVERAGE_DISTANCE: 8.633km
Finished in: : 4.285s

Process finished with exit code 0
```

***Slika 4.4.4.2**: Rezultati Random forest regresora*

Rezultat je bolji u odnosu na regresor stabla odlučivanja, ali je znatno sporiji. S obzirom da uzima deset stabla odlučivanja i da radi na samo jednom jezgru procesora, ovi rezultati su sasvim opravdani.

Ako bismo povećali broj stabla odlučivanja *Random forest* regresoru, kao i broj jezgara koji mu je dopušten na korišćenje, kod bi izgledao ovako:

```
print("-------------------------------------")
print("Random Forest Regressor (with params)")

start = timer()

clf = RandomForestRegressor(n_estimators=150, n_jobs=4)
clf.fit(X_train, y_train)
confidence = clf.score(X_test, y_test)
prediction = clf.predict(np.array(expectedPre))

end = timer()

print('{:16s}: {:5.3f}{:2s}'.format("AVERAGE_DISTANCE", calculateAverageDistance(prediction, np.array(expectedPost)),
                                   "km"))
print('{:13s}: {:5.3f}{:1s}'.format("Finished in: ", end - start, "s"))
```

***Slika 4.4.4.3**: Kod za predikciju koordinata saobraćajnih nesreća korišćenjem Random forest regresora*

Puštanjem koda sa slike 4.4.4.3 dobijaju se sledeći rezultati:

```
Data import.
Data successfully imported.
-------------------------------------
Random Forest Regressor (with params)
AVERAGE_DISTANCE: 8.427km
Finished in: : 19.870s

Process finished with exit code 0
```

***Slika 4.4.4.4**: Rezultati Random forest regresora sa modifikovanim parametrima*

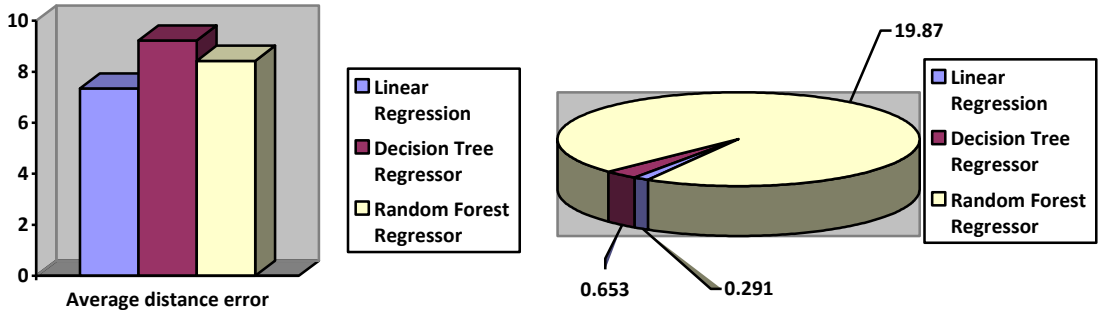
4.4.5. Analiza rezultata sva tri algoritma

Vrednosti prosečne greške udaljenosti i vreme izvršavanja sva tri algoritma su prikazana u tabeli 4.4.5, kao i na slici 4.4.5:

***Tabela 4.4.5**: Rezultati algoritama linearne regresije, stable odlučivanja i na osnovu nasumičnih šuma*

	Linear Regression	Decision Tree Regressor	Random Forest Regressor
Average distance error	7.339 km	9.22 km	8.427 km
Execution time	0.291s	0.653s	19.870s

Korišćenjem stabla odlučivanja, dobija se za prosečnu grešku 9.22 kilometara, što je za skoro 2 kilometara više od prosečne greške dobijene korišćenjem linearnog regresora, a skoro kilometar više od prosečne greške dobijene *Random forest* regresora. Iako je drastično brži od *Radnom forest* regresora, stablo odlučivanja je i dalje više od dva puta sporiji od linearnog regresora, što govori da ovaj algoritam nije najefikasniji za konkretan problem i njegove podatke.



***Slika 4.4.5**: Rezultati algoritama linearne regresije, stable odlučivanja i na osnovu nasumičnih šuma: a) sa leve, strane prosečna greška razdaljine u kilometrima po regresoru; b) sa desne strane, vreme izvršavanja u sekundama.*

Kako se broj stabla odlučivanja povećao na 150 i broj dozvoljenih jezgara na 4 u slučaju *Random forest* regresora sa parametrima, prosečna greška se smanjila za malo više od 200 metara, što je dobro. Iako je podešeno da radi na više jezgara, parametrizovani *Random forest* regresor je znatno sporiji nego onaj koji radi na jednom jezgru, što je opravdano time da se sad koristi 15 puta više stabla nego pre.

Iako je on dao mnogo bolje rezultate nego *Decision tree*, njegova prosečna greška je i dalje za kilometar veća od linearnog regresora. Pored toga, linearni regresor je drastično brži i od regresora stabla odlučivanja i od *Random forest* regresora (gleda se regresor sa najmanjom vrednošću u oba grafa na slici 4.4.5). Shodno tome, može se slobodno reći da od ova tri algoritma, linearni regresor daje najbolje rezultate u najkraćem mogućem roku.

5. ZAKLJUČAK

Mašinsko učenje je danas svakodnevnica. Vremenom postaje sve korisniji i traženiji alat koji može biti od pomoći čoveku u širokom domenu. Brojni su primeri gde ova tehnologija igra glavnu ulogu od kojih su neke od životne važnosti.

Velika prednost mašinskog učenja je ta što je konstruisana u pravcu da poseduje mogućnost samostalnog učenja bez eksplicitnog programiranja. Na taj način, softver je više uposlen i brže uči, što mnogo olakšava krajnjim korisnicima.

Pored mnogih alata danas za obradu velikih količina podataka koje iz dana u dan sve više i više raste, mašinsko učenje se pokazalo kao jedna od najboljih. Upravo ona ima sposobnost da korišćenjem adekvatnih algoritama uvidi bitne veze i korelacije između podataka i da na prikladan način reaguje na njih. Za razliku od čoveka i njegovog učenja gde su česte situacije zaboravljanja starih informacija, kod mašinskog učenja, prilikom dolaska novih informacija, one se nadograđuju na stare koje se i dalje čuvaju i koriste zarad novijih predikcija. Da bi mašinsko učenje bilo korisno čoveku, njeno znanje mora biti mnogo jače od njegovog, što je i razlog njenog kreiranja. Za poslove i podatke koje prevazilaze ljudski um, čovek je stvorio softver koji uči za njega i radi posao što predstavlja osnovu današnje veštačke inteligencije čija je glavna namena da pomogne čovečanstvu.

Kroz ovaj rad, detaljno su objašnjeni osnovni koncepti mašinskog učenja, njegove primene i način njegovog funkcionisanja. Kako je ova oblast dosta obimna, kroz njenu podelu na algoritme nadgledanog i nenadgledanog učenja, pa i kroz njihovu podelu dalje, čitaoci ovog rada mogu da steknu okvirni utisak kako sistem funkcioniše iznutra. Takođe, mogu se upoznati i sa bitnim terminima koji se koriste u ovoj oblasti, kao i da prepoznaju u kojim se slučajevima koji algoritmi koriste. Za svaki objašnjeni algoritam postoji primer iz realnog života koji na slikovit način pokušava da približi opis njegovog funkcionisanja čitaocima ovog rada.

Kao adekvatan primer primene mašinskog učenja u realnim situacijama, odabran je problem saobraćajnih nesreća na teritoriji grada Beograda. Podaci o nezgodama, preuzeti sa Portala otvorenih podataka, su i više nego dobro opisani, sa dosta korisnih informacija o njima. Kao dodatak, za svaku nesreću su preuzeti podaci o meteorološkim stanjima u trenucima događaja. Skup takvih podataka je sasvim solidan za treniranje algoritama mašinskog učenja.

Kako je za temu rada odabrano predviđanje lokacija nezgoda u zavisnosti od drugih parametara dataset-a, a lokacija je predstavljena kao par realnih brojeva (geografska širina i dužina), za njihovu predikciju odgovaraju samo regresivni algoritmi učenja. Korišćenjem *Scikit-learn Python* biblioteke, implementacije svih regresora zahtevaju realne brojeve i kao ulazne i kao izlazne parametre. S obzirom da vrednosti u preuzetom dataset-u nisu sve brojnog tipa, bilo

je neophodno refaktorisati vrednosti koje koriste algoritmi za predikciju kako bi bili izraženi u vidu realnih brojeva. Cela logika pripreme podataka za mašinsko učenje je implementirana unutar Javinog projekta pod nazivom *Accidents CSV generator*.

Nakon njihove pripreme, za predikciju koordinata uzeta su u obzir tri regresivna algoritma: linearna regresija, stablo odlučivanja i algoritam odlučivanja na osnovu nasumičnih šuma. Cilj je bio da se izvrše predikcije sva tri algoritma, izračuna prosečna greška udaljenosti lokacija, kao i vreme njihovog izvršavanja i videti koji je od njih najefikasniji za dati problem. Istraživanje je pokazalo da je linearni regresor najpogodniji za predikciju, ne samo zbog najmanje prosečne greške udaljenosti, nego i drastičnoj razlici u vremenu izvršavanja od preostala dva algoritma.

Kada se malo bolje sagleda dataset, uočava se prostor za predikcije i drugih vrednosti osim lokacije saobraćajnih nesreća. Otvorena je mogućnost predviđanja tipova nezgoda na osnovu ostalih parametara korišćenjem nekih od klasifikacionih algoritama, kao i mnogo drugih stvari. Takođe, moguće je primeniti i neke od nenadgledanih algoritama koji skeniraju sve podatke kao ulazne kako bi na osnovu njih napravili određene klastere.

Kao što se može primetiti, primeri su brojni različitih mogućnosti analize problema saobraćajnih nesreća. S obzirom na to da je ovo jedan u milion problema sa kojim se čovek danas susreće, jasno je koliko je važna uloga mašinskog učenja u rešavanju istih. U bliskoj budućnosti (ako ne i sadašnjosti), ono može biti od velikog značaja samom čovečanstvu.

6. LITERATURA

1. I. Krišto, *Primjena metoda strojnog učenja za poboljšanje pretraživanja dokumenata, diplomski rad,* Sveučilište u Zagrebu, fakultet elektrotehnike i računarstva, 2012. preuzeto u Avgustu 2018. link: http://ferko.fer.hr/ferko/EPortfolio!dlFile.action;jsessionid=42713DBCB23F1EF250DCDA50EAE46163.jvm1?id=297

2. A. Palkovač, *Istraživanje podataka uz pomoć softverskog alata Rapidminer, master rad,* Beograd, univerzitet Singidunum, Departman za poslediplomske studije, Savremene informacione tehnologije, Master studijski program, preuzeto u Avgustu 2017. link: https://www.google.rs/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=2ahUKEwi5sNSAmpDdAhVBiqQKHY6mBQUQFjADegQIBxAC&url=https%3A%2F%2Fsingipedia.singidunum.ac.rs%2Fpreuzmi%2F41760-istrazivanje-podataka-uz-pomoc-softverskog-alata-rapidminer%2F1839&usg=AOvVaw12BTIQ_ixaLbiSzopvP_d-

3. M. Gavrilov, *Šta je mašinsko učenje i kako menja poslovne softvere*, Startit, 31.07.2015 preuzeto u Avgustu 2018. link: https://startit.rs/sta-je-masinsko-ucenje-i-kako-menja-poslovne-softvere/

4. Vikipedija, *Mašinsko učenje*, preuzeto u Avgustu 2018. link: https://sr.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D1%81%D0%BA%D0%BE_%D1%83%D1%87%D0%B5%D1%9A%D0%B5

5. Studenti.rs, *Mašinsko učenje*, 22.01.2016. preuzeto u Avgustu 2018. link: http://studenti.rs/skripte/masinstvo/masinsko-ucenje/

6. Wikipedia, *Machine learning*, preuzeto u Avgustu 2018. link: https://en.wikipedia.org/wiki/Machine_learning#History_and_relationships_to_other_fields

7. adaptirano prema dokumentaciji MathWorks, preuzeto u Avgustu 2018. link: https://www.mathworks.com/help/stats/machine-learning-in-matlab.html

8. adaptirano prema *Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning, Ronald van Loon*, 05.02.2018. preuzeto u Avgustu 2018. link: http://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/

9. N. Komljenović, *Primjena metoda strojnog učenja u analizi korisničkih upita*, *diplomski rad,* Osijek, Sveučilište Josiša Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija, Sveučilišni studij, preuzeto u Avgustu 2018. link: https://repozitorij.unios.hr/islandora/object/etfos:1490/preview

10. adaptirano prema *Introduction to Clustering and Unsupervised Learning, Packt*, 23.02.2016. preuzeto u Avgustu 2018. link: https://hub.packtpub.com/introduction-clustering-and-unsupervised-learning/

11. J. Brownlee, *Linear Regression for Machine Learning, Machine Leargning Algorithms*, 25.03.2016. preuzeto u Avgustu 2018. link: https://machinelearningmastery.com/linear-regression-for-machine-learning/

12. adaptirano prema *Linear regression, Wikipedia*, preuzeto u Avgustu 2018. link: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

13. adaptirano prema *Ordinary Least Squares (OLS), Xavier Geerinck - Blog*, Xavier Geerinck, 17.05.2018. preuzeto u Avgustu 2018. link: https://xaviergeerinck.com/ordinary-least-squares

14. A. Brown, *Machine Learning With Decision Trees, AI Zone*, 14.02.2018. preuzeto u Avgustu 2018. link: https://dzone.com/articles/machine-learning-with-decision-trees-1

15. adaptirano prema *Decision Tree - Regression,* preuzeto u Avgustu 2018. link: http://www.saedsayad.com/decision_tree_reg.htm

16. N. Donges, *The Random Forest Algorithm*, 22.02.2018. preuzeto u Avgustu 2018. link: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

17. M. Stepić, *Statistika saobraćajnih nezgoda*, Kragujevac, Fakultet fakultet inženjerskih nauka u Kragujevcu, 2015. preuzeto u Avgustu 2018. link: http://www.cqm.rs/2015/cd3/pdf/papers/focus_1/54.pdf

18. adaptirano prema *Analiza podataka o saobraćajnim nezgodama u programskom jeziku R,* 20.3.2017. preuzeto u Avgustu 2018. link: https://data.gov.rs/sr/reuses/analiza-podataka-o-saobratshajnim-nezgodama-u-programskom-jeziku-r/