Sveučilište Josipa Jurja Strossmayera

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Seminar iz kolegija Raspoznavanje uzoraka i strojno učenje:

SLUČAJNE ŠUME (RANDOM FORESTS)

Student:

Tomislav Rešicki , DRB

U Osijeku,

Rujan, 2016.

Sadržaj

1.UVOD **3**

2.STROJNO UČENJE **4**

2.1.Metode i algoritmi strojnog učenja **4**

2.2. Stablo odlučivanja i metode **5**

2.3.Nadgledano učenje **6**

2.4.Klasifikacija **7**

2.5.Regresija **7**

3.STABLO ODLUČIVANJA **8**

3.1.Struktura stabla odlučivanja **8**

3.2.Korištenje stabla i prednosti **9**

3.3.Izrada stabla odlučivanja **10**

4. SLUČAJNA ŠUMA (RANDOM FOREST) **11**

4.1.Općenito o slučajnim šumama **11**

4.2.Algoritam izrade slučajne šume **13**

4.3.Prednosti i nedostaci slučajne šume **14**

5.PROGRAMSKA IMPLEMENTACIJA STABLA ODLUČIVANJA I SLUČAJNE ŠUME **14**

6.ZAKLJUČAK **15**

7.LITERATURA **16**

1.UVOD

Strojnim učenjem možemo nazvati programiranje računala na taj način da optimiziraju neki kriterij uspješnosti na temelju podatkovnih primjera ili nekog iskustva od ranije [1]. Osnovni princip strojnog učenja opisuje pojave iz podataka. Rezultat mogu biti pravila, funkcije, odnosi, sustavi jednadžbi, distribucija vrijednosti ili slično. Strojno učenje ima naglasak na automatiziranju otkrivanja i korištenja pravilnosti u podacima, karakteriziranju podataka koje koristimo za učenje, kao i uvjete (metode) koje garantiraju kvalitetu naučenih modela. Raspolažemo modelom koji je definiran na neke parametre, a učenje se provodi optimizacijom parametara modela temeljem podataka. Postavlja se pitanje, zašto strojno učenje? Strojnim učenjem se rješavaju složeniji problemi (ne postoji ljudsko znanje o procesu ili ljudi ne mogu dati objašnjenje o procesu). Također, strojno učenje može raditi na ogromnim količinama podataka, otkrivaju znanja u skupovima podataka. Primjena strojnog učenja na velike baze podataka je kod dubinske analize podataka ili otkrivanja znanja u skupovima podataka. Strojno učenje usko je povezano i sa umjetnom inteligencijom (robotika, raspoznavanje uzoraka, govora, umjetne neuronske mreže, pretraživanje informacija i sl.) te statistikom.

Podjela strojnog učenja dijeli se na nadgledano (nadzirano) i nenadgledano (nenadzirano), te učenje uz podršku i ostala učenja (polu-nadzirano, relacijsko, transduktivno i sl.).

2.STROJNO UČENJE

Primjena strojnog učenja, osim u računarnim i upravljačkim sustavima zastupljena je i u medicini, biologiji, oblikovanju, inžinjerstvu, otkrivanja podataka u tekstu, glazbi ili slici. U strojnom učenju nemamo univerzalni algoritam za učenje ali su ipak osmisljeni efikasni algoritmi koji rješavaju određeni tip problema, kao i njihove uspješne implementacije.

2.1.Metode i algoritmi strojnog učenja

Postoje najčešći oblici primjene algoritama strojnog učenja i dijele se na : otkrivanje znanja u novim skupovima, programske implementacije koje nije moguće napraviti na klasičan način i programe koji se trebaju dinamički mijenjati i prilagođavati uvjetima. Metode i algoritmi strojnog učenja su:

1) Reprezentacija modela/funkcije -> linearna funkcija, polinom, nepoznate vrijednosti slobodnih parametara

2) Algoritam pretraživanja / optimizacije -> strojno učenje se poistovjećuje sa pretraživanjem / optimizacijskim problemom

3) Metoda procjene pogreške na neviđenim primjerima -> skalarna funkcija cilja kojom ćemo kvantificirati kako radi / generalizira naš naučeni model

Strojno učenje sa nadzorom dijeli se na regresiju i klasifikaciju a sa oba pojma ćemo se detaljno upoznati u nastavku. Algoritmi koji se koriste u strojnom učenju a vezani su za učenje sa nadzorom su : naivni Bayes, stabla odluke, štreber. Ukratko ćemo se upoznati sa svakim od njih:

Naivni Bayes -> zadatak mu je da jednostavno izračuna apriorne vjerojatnosti klasa i uvjetne vjerojatnosti klasa i svojstava. Nakon toga mu je zadatak odabrati klasu s najvećom aposteriornom vjerojatnošću.

Štreber -> najlakši algoritam za učenje svih primjera napamet. Ima mogućnost pohraniti u memoriju sve što je potrebno za naučiti. Radi na principu da ako dobije pitanje za klasu novog objekta, on gleda u memoriju. Ako objekt već postoji u memoriji, vraća se nova klasa a ukoliko ne postoji, vraća povratnu informaciju da ne zna.

Stabla odluke-> alternativni način prikazivanja i analize situacije odlučivanja. Omogućuju nam prikazivanje vjerojatnosti koje su povezane sa različitim pravcima za koje se ne možemo odlučiti.

2.2.Stablo odlučivanja i metode

Stabla odlučivanja (decision tree) su zapravo veoma moćne i popularne tehnike modeliranja za klasifikacijske i predikcijske probleme. Stabla su temeljena na metodi podjele, pri čemu je skup primjera podijeljen na određene podskupove i ta se podjela rekurzivno ponavlja dok se ne dođe do primjera koji izvršava klasifikaciju znanja.

Stablo odlučivanja primjenjuje se za :

1)Razvrstavanje (classification)

2)Predviđanje (prediction)

3)Procjenu vrijednosti (estimation)

4)Grupiranje (clustering)

5)Opisivanje podataka

6)Vizualizaciju

Podjela stabla odlučivanja ostvarena je po raznim kriterijima, pa se tako dijele na regresijska stabla više značajki, ukoliko je ciljna varijabla realan broj i stabla za razvrstavanje.

Standardna metoda izrade modela korištenjem stabla učenja je rekurzivno particioniranje (metoda kod koje izrada modela kreće od korijena stabla). Sam proces učenja započinje usporedbom mogućih podjela na temelju neke značajke. Kod realnih brojeva se uspoređuju podjele kod svake vrijednosti koja se pojavljuje u toj značajki. Za odabir najbolje podjele postoje razni kriteriji kojima se mjeri smanjenje varijabilnosti distribucije ciljne značajke u granama ispod u usporedbi sa granom na kojoj se radi podjela. Podjela u korijenu predstavlja podjelu ulaznog prostora na dva podprostora s granicom paralelnom jednoj ulaznoj (značajki). Ta podjela se tada ponavlja u svakoj slijedećoj grani dok sve grane ne postanu potpuno čiste ili dok se ne zadovolji neki od kriterija zaustavljanja. Moguće je da, pomoću ulaznih značajki koje su na raspolaganju, nije moguće postići potpunu čistoću podgrana.

2.3. Nadgledano učenje

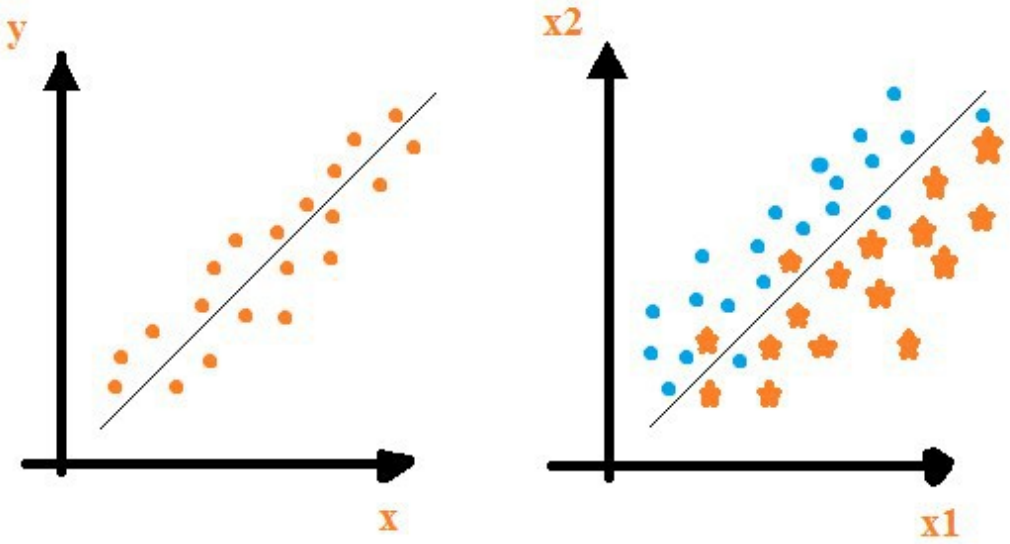
Nadgledano učenje je učenje u kojemu za svaki ulaz postoji odgovarajući izlaz. Kada dobijemo rješenje na izlazu, uspoređujemo ga sa točnim izlazom. Ovim postupcima se mogu rješavati dvije vrste problema, a to su klasifikacija i regresija. Razlika između klasifikacije i regresije je u tome što kod klasifikacije primjeru pridružujemo klasu kojoj primjer pripada, dok kod regresije pridružujemo neku kontinuiranu vrijednost. Kod klasifikacije je ciljna varijabla nominalna a kod regresije kontinuirana. Cilj nadgledanog učenja je minimizirati grešku između stvarnog izlaza (učenika) i željenog izlaza (učitelj).

2.4. Klasifikacija

Klasifikacija (raspoznavanje uzoraka) ima svrhu odrediti neku klasu C, kojoj pripada primjer x. Primjeri se mogu interpretirati kao točke u n-dimenzijskom prostoru koji nazivamo ulazni prostor ili prostor primjera. Postavimo skup X kao skup svih mogućih primjera. Svi algoritmi strojnog učenja imaju pretpostavku da su primjeri iz X uzrokovani nezavisno i iz zajedničke distribucije P(x,y) i tu pretpostavku nazivamo iid (independent and identically distributed). Kod nadgledanog učenja znamo oznaku klase y kojoj pripada primjer x iz skupa za učenje. Primjer klasifikacije je raspoznavanje lica, gdje imamo skup podataka za učenje u vidu slika a zadatak je prepoznati lice unatoč promjenama u pozi, osvjetljenju, šminki, frizuri i slično.

2.5. Regresija

Regresija bi se najjednostavnije mogla opisati kao učenje funkcija, a njena eksplicitna informacija o objektu je numerička (npr. cijena nečega, broj kilograma ili slično). Kao cilj regresije smatra se aproksimiranje nepoznate funkcije. Kao primjer regresije možemo uzeti input-output odnos nekog složenog sustava. Tako regresiju možemo primjetiti kod otkucaja srca (reakcija na različite podražaje).



Sl. 2.1. Primjer regresije i klasifikacije

3.STABLO ODLUČIVANJA

3.1. Struktura stabla odlučivanja

Sastoje se od čvorova i grana, gdje su roditeljski čvorovi povezani sa roditeljskim putem grana. Kako bi se to efikasnije objasnilo, kažemo da čvor koji nema roditelja postaje korijenski čvor, a čvorovi bez djece su listovi odnosno čvorovi odgovora (daju sva moguća rješenja problema), a ostale čvorove nazivamo čvorovima odluke. Rezulati odluke su "da" ili "ne" ili pak izbori između više ponuđenih vrijednosti. Možemo reći da je stablo odlučivanja klasifikacijski algoritam u formi stablaste strukture u kojemu razlikujemo dva tipa čvorova koji su povezni granama:

a) krajnji čvor (leaf node) -> njime završava određena grana stabla i oni definiraju klasu kojoj pripadaju primjeri koji zadovoljavaju uvjete na toj grani stabla

b) čvor odluke (decision node) -> definira određeni uvjet u obliku vrijednosti varijable iz kojeg izlaze grane koje zadovoljavaju vrijednosti atributa.

3.2. Korištenje stabla i prednosti

Stabla odlučivanja mogu se koristiti za klasifikaciju primjera i to na način da se krene od prvog čvora odlučivanja u korijenu stabla i kreće po onim granama stabla koja primjer sa svojim vrijednostima zadovoljava , sve do krajnjeg čvora. Krajnji čvor klasificira primjer u jednu od postojećih klasa problema. Osnovni preduvjeti za korištenje stabla odlučivanja:

- Značajan broj primjera – potrebno je što više primjera u skupu generiranja
- Diskretnost klasa – svaki primjer mora pripadati samo jednoj od klasa, dok klasa mora biti puno manje od primjera
- Definiran broj klasa – raspoređivanje primjera u kategorije mora biti unaprijed definirano, gdje te kategorije moraju imati konačan broj
- Opis u obliku atributa – podaci o primjeru moraju biti opisani u obliku konačnog broja atributa

Kako bi se lakše upoznali sa stablima odlučivanja, potrebno je znati i prednosti ovog načina pretraživanja a to su :

- Kompaktniji oblik – znanje je čitljivije nego kod pravila
- Mali zahtjevi za računalne resurse – vrijeme i memorija
- Mogućnost korištenja svih tipova atributa – kategorički i numerički
- Sposobnost generiranja razumljivih metoda
- Odražavanje važnosti pojedinih atributa za konkretni problem
- Lakša provjera pogrešaka
- Mogućnost raspodjele – ako radimo sa složenijim stablom možemo ga predstaviti kao odvojeno stablo što olakšava razvoj stabla

3.3. Izrada stabla odlučivanja

Da bismo se odlučili za stablo odlučivanja, moramo biti upoznati sa uvjetima za odlučivanje a neki od tih uvjeta su :

- Uvjet potpune sigurnosti (da sa 100% sigurnošću možemo znati što će se dogoditi u budućnosti)
- Uvjet nesigurnosti (što bi se moglo dogoditi ali ne znamo kolike su vjerojatnosti da će se neka od tih varijanti dogoditi
- Uvjet rizika (poznate varijante u budućnosti,kao i vjerojatnost da će se neka od njih i dogoditi)

Prije same konstrukcije stabla odlučivanja možemo zadati problem odlučivanja koji će služiti za izradu stabla odlučivanja. Taj model u sebi sadrži informacije o alternativama i vrijednostima koje će biti korištene u stablu. Sama izrada stabla odlučivanja dijeli se na 3 koraka a mi ćemo se ukratko upoznati sa svakim.

1. KORAK

Izgradnja logičkog modela

U prvom koraku svim čvorovima odluka , čvorovima posljedica , granama alternativnih akcija, kao i granama posljedičnih stanja pridružuju se vjerojatnosti pojave pojedine posljedice i parcijalan tok (npr. novca u nekoj udruzi ili firmi).

2. KORAK

Računanje očekivanih vrijednosti odluka postupkom unazad

Računanje unatrag započinje na krajnjim čvorovima stabla i kreće prema unatrag, do početnog čvora odluke. Svakom se čvoru dodaje ekvivalentna očekivana vrijednost i to na način:

- na završnom čvoru je izračunata konačna vrijednost te alternative
- na čvoru odluka očekivana pridružena vrijednost jednaka je najvećoj od algoritmom prethodno izračunatih očekivanih vrijednosti neposrednih sljedećih čvorova u stablu odlučivanja
- čvoru posljedica dodaju se očekivane vrijednosti izračunate kao :

$$EV_{i-1} = \sum_j p_j EV_i, i \in \{1,2,\dots,n\}, j \in \{1,2,..m\}$$

gdje je EV_{i-1} očekivana vrijednost u čvoru *i-1*

Ev_i je očekivana vrijednost u čvoru *i*

p_j je vjerojatnost grane *j* koja izlazi iz čvora posljedica *i-1*

3.KORAK

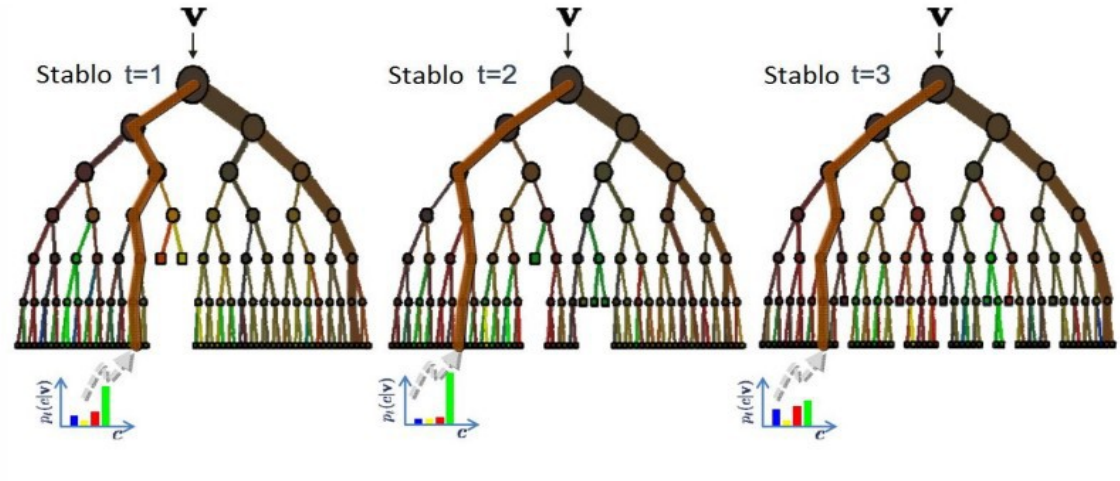
Pronalazak optimalnog puta

3. korak je pronalaženje optimalnog puta postupkom računanja prema naprijed. Nakon što su se u prethodnom koraku izračunale očekivane vrijednosti za svaki čvor, može se prepoznati optimalan put u stablu odlučivanja, računanjem prema naprijed, od početnog čvora odluke. Pridružena vrijednost početnom čvoru je jednaka očekivanoj vrijednosti grane koja se nalazi na optimalnom putu. Analogno se promatra sljedeći čvor, sve do završnog čvora.

4. SLUČAJNA ŠUMA (RANDOM FOREST)

4.1.Općenito o slučajnim šumama

Slučajna šuma je klasifikacijski algoritam koji su razvili Leo Breiman i Adele Cutler. Termin „slučajna šuma odlučivanja“ prvi je predložio Tin Kam Ho 1995. godine. Metoda slučajne šume ujedinjuje bagging tehniku koju je razvio Breiman i Ho-ovu metodu slučajnog odabira potprostora (engl. „random subspace method“) sa svrhom stvaranja skupa stabala odluke s kontroliranim varijacijama. Osnovna ideja je korištenje mnogo, pojedinačno slabih, klasifikatora (stabala odluke) na temelju čijih se izlaza donosi odluka o pripadnosti uzorka nekoj od klasa. Sastoje se od kolekcije nezavisnih stabala odlučivanja gdje svako stablo predstavlja jedan glas u većinskom donošenju odluke. Vizualni primjer prikazan je na slici 2.



Sl. 4.2. Slučajna šuma sa 3 stabla

Kao što je vidljivo, slučajna šuma je logična nadogradnja na model stabla odlučivanja koja rješava neke njegove nedostatke. Kao što se da zaključiti iz imena klasifikatora bitna stvar prilikom treninga stabla je izvor slučajnosti. Izvori slučajnosti prilikom treninga mogu biti različiti kao recimo nasumičan odabir podataka za treniranje, nasumično odabran podskup od odabranih podataka za treniranje pa čak i nasumičan odabir značajki po kojem će se raditi test funkcija u svakom čvoru. Ti izvori slučajnosti rješavaju problem velike varijance

pojedinačnog stabla odlučivanja jer je svako stablo trenirano na drugačijem podskupu podataka. Krajnja odluka se dobiva tako da se uzme histogram odluka svakog stabla i ona odluka sa najvećim brojem ponavljanja se uzima kao konačna. Sama konstrukcija šume kreće od *n* broja primjera, *k* broj stabala i *m* zadanog parametra. Za svako od *k* stabala uzimamo nasumično *n* primjera za učenje s preklapanjem. Stablo se konstruira tako da na svakom čvoru odaberemo *m* atributa i radimo najbolju podjelu po tih *m* atributa. Na taj način stablo raste do maksimalne veličine. Ako bismo htjeli klasificirati novi primjer tada pustimo primjer niz svako stablo, gdje svako stablo zasebno donosi odluku o klasi, a primjer će biti klasificiran klasom koja "bude imala najviše glasova". Pogreška slučajne šume može ovisiti o dvije veličine a to su kolinearnost stabala (što je veća, veća je i greška) te jakost svakog pojedinog stabla u šumi (jako stablo ima nisku pogrešku, a povećanjem jakosti svakog stabla smanjuje se greška cijele šume). U ovim odnosima važno je napomenuti da se i smanjenjem varijable *m* smanjuju i jakost i kolinearnost. Kako bismo dobili optimalan interval za *m* koristimo oob stopu pogreške. Tijekom rasta šume, procjenjuje se i pogreška, a svako stablo pojedinačno gradi drugačiji skup primjera. Svaki podatak se pušta kroz stablo da se izvrši klasifikacija. Na kraju izvođenja , postavljamo *j* kao klasu koja je dobila najviše glasova svaki puta kada je primjer *n* bio oob. Svaki omjer broja puta kada *j* nije jednak prvoj klasi i prosjek svih n-ova daju oob procjenu pogreške.

4.2.Algoritam izrade slučajne šume

1. Uzorkuje se T nezavisnih podskupova skupa podataka za učenje. Svako nezavisno sampliranje je osnova za učenje jednog stabla odlučivanja.

2. Izgradi se maksimalno duboka stabla (bez ograničenja dubine stabla). U izgradnji stabala za svaki čvor slučajno se odabere mali podskup značajki. Unutar toga podskupa odredi se najbolja značajka uobičajenom metodom.

3. Podskup stabala testira se na svim preostalim primjerima ili se svako stablo testira sa svojim preostalim primjerima. Pogreška se usrednjava preko svih stabala.

4. Za nepoznati primjer X prikupljaju se glasovi svih stabala i određuje većinski razred klasifikacije.

Broj stabala (T) se povećava dok se točnost klasifikacije na skupovima za evaluaciju ne ustali. Veličina podskupa slučajnih značajki za svaki čvor je √n gdje je n veličina cijelog skupa. Povećanje slučajnog podskupa povećava točnost klasifikacije svakog pojedinog stabla, ali i međusobnu korelaciju između stabala što smanjuje točnost klasifikacije cijele šume.

4.3.Prednosti i nedostaci slučajne šume

Slučajna šuma pruža velike mogućnosti u analizi podataka. Zbog nasumičnosti odabira podataka nije potrebna međuvalidacija. Slučajne se šume ne prilagođavaju podacima za učenje zbog toga jer je svako stablo naučeno na nekom drugom nasumičnom podskupu podataka tako da daje drugačije odluke. Također, povrđena je bolja točnost klasifikacije u usporedbi sa popularnim algoritmima. Ova metoda je zapravo veoma efikasna za velike baze podataka i podatke sa nepoznatim vrijednostima.

Kada govorimo o manama ovog načina analize podataka, problem je vremenska i prostorna složenost s obzirom na jednostavnije klasifikatore.

5.PROGRAMSKA IMPLEMENTACIJA STABLA ODLUČIVANJA I SLUČAJNE ŠUME

Programski paket koji je izabran je Python a na jednostavnijem primjeru prikazati će se generiranje slučajne šume.

```
from sklearn.ensemble import RandomForestClassifier7

from numpy import genfromtxt, savetxt

def main():
dataset = genfromtxt(open('Data/train.csv','r'), delimiter=',', dtype='f8')[1:]
target = [x[0] for x in dataset]
train = [x[1:] for x in dataset]
test = genfromtxt(open('Data/test.csv','r'), delimiter=',', dtype='f8')[1:]

rf = RandomForestClassifier(n_estimators=100)
rf.fit(train, target)
savetxt('Data/submission2.csv', rf.predict(test), delimiter=',', fmt='%f')
if __name__=="__main__":
  main()
```

Dakle, jednostavnim kodom prikazati ćemo način rada slučajnih šuma. U kodu su vidljive funkcije koje su definirane u Pythonu (RandomForestClassifier7 , fit, predict). Unutar maina (def main():) kreiramo trening i test podatke. Kako bismo otvorili naš csv file sa podacima za treniranje koristimo naredbu dataset i tu toj liniji koda učitavamo te podatke s kojima ćemo raditi. Na kraju ove linije koda vidimo [1:] , što bi značilo da uzimamo sve unose od 2. do posljednjeg.

Kada smo došli do učitavanja podataka, imamo dvije naredbe target i train. U naredbi target uzimamo prvi podatak iz dataseta [0:], dok u naredbi train spremamo sve ostale podatke [1:]. Slijedeća linija koda (test) radi identičnu stvar kao i dataset samo što uzima druge podatke (iz drugog file-a).

Nakon što smo učitali podatke za treninranje, potrebno je generirati slučajnu šumu i to radimo pomoću naredbe "rf", gdje se uz pomoć funkcije fit kreće u trening prema train i target podacima koje smo predali. Na kraju koda vidimo liniju sa početkom savetxt, što znači da nakon generiranja spremimo podatke u neki novi csv file.

6.ZAKLJUČAK

Cilj rada bio je upoznati se detaljnije sa algoritmom Random forest (slučajne šume), te prikazati njegovu korisnost i učinkovitost. Kako bismo lakše mogli razumjeti pojmove koji nam se nameću u terminologiji, potrebno je upoznati pozadinu alogritma. Upoznavanjem sa strojnim učenjem (metodama, podjelama i slično) lakše je shvatiti stablo odlučivanja a krajnje i slučajne šume. Kroz rad smo naučili koje su prednosti i nedostaci, princip funkcioniranja slučajnih šuma kako bismo lakše shvatili programsko rješenje dano za temu koja se obrađivala. Nakon temeljite teorijske podloge bilo je potrebno izabrati programski paket u kojemu će se prikazati jednostavniji kod koji bi prikazao funkcioniranje slučajnih šuma a mi smo se odučili za Python, pošto smo se s njime upoznali kroz kolegij Raspoznavanje uzoraka i strojno učenje. Na jednostavnom primjeru možemo vidjeti kako se u svega par linija koda može raditi sa velikim skupom podataka.

7.LITERATURA

[1] Materijali s kolegija Strojno učenje, FER Zagreb

Nastavni materijal s kolegija Raspoznavanje uzoraka i strojno učenje, FERITOS

Internetske stranice :

http://www.sas.com/en_us/insights/analytics/machine-learning.html

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html