Izveštaj

Luka Arambašić

Jun 2023

Contents

1 Uvod

Projekat je radjen na *diamonds* dataset-u iz biblioteke *tensorflow_datasets*.

Projekat za cilj ima da se prikazu razlicite tehnike analize i pretprocesiranja nad skupom podataka i demonstracije modela klasifikacije, klasterovanja i pravila pridruzivanja.

Opste o skupu podataka:

Skup podataka sadrzi fizicke atribute i cene 53 940 dijamanata.
Atributi(10):

- price: Cena u US dolarima. (326 USD–18,823 USD)
- cut: Kvalitet reza. (Fair, Good, Very Good, Premium, Ideal)
- color: Boja dijamanta. od D (najbolje) do J (najgore)
- clarity: Cistoca dijamanta. (I1 (najgore), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (najbolje))
- x: Duzina u mm. (0–10.74)
- y: Sirina u mm. (0–58.9)
- z: Dubina mm. (0–31.8)
- depth: 100 * z / mean(x, y). (43–79)
- table: Sirina vrha dijamanta u odnosu na najsiru tacku. (43–95)

Homepage: https://ggplot2.tidyverse.org/reference/diamonds.html
Source code: tfds.structured.diamonds.Diamonds
Versions: 1.0.0 (default): Initial release.
Download size: 2.64 MiB
Dataset size: 13.01 MiB

Klasifikacija: ovaj skup podataka se obicno koristi u svrhe problema linearne regresije i predvidjanja cene na osnovu atributa. Kako bismo problem regresije sveli na problem klasifikacije, mozemo kontinualni atribut cene podeliti na opsege cena koji ce nam ciniti klase za predvidjanje. S obzirom da vec imamo informaciju da je najvisa cena u skupu 18 823USD, napravicemo cetiri klase: [0, 5000], (5000, 10000], (10000, 15000] i 15000+

Klasterovanje: cilj je uociti da li u skupu podataka postoje grupacije instanci i izvuci dodatne informacije koje nam mogu pomoci u modelovanju.

Pravila pridruzivanja: cilj je uociti da li u podacima postoje neke pravilnosti koje se veoma cesto ponavljaju i uslovljavaju neki specificni rezultat.

2 Opis modela

Pretprocesovanje

S obzirom da je ovo jedan od popularnih skupova podataka koji se koristi u svrhe vezbanja masinskog ucenja, sam skup podataka je vec donekle preprocesovan i spreman za modelovanje. *Nedostajuce vrednosti* ne postoje. *Kodiranje kategorickih atributa* je vec izvrseno i celobrojne vrednosti se vec nalaze u kolonama atributa "clarity", "color" i "cut".

Inicijalno, svi atributi su imali prefiks "features/" koji smo odstranili radi lakseg rukovanja skupom podataka.

Outlier-i su tehnikom *z-score* odstranjeni ako im je skor veci od 3. Prethodno je primenjeno na sve atribute osim atributa "price" koji zelimo da predvidjamo. Razlog za to je jer postoji mogucnost da su potencijalni outlier-i samo kombinacija visoko kvalitetnih atributa i da to utice na odvajanje u ceni. Nakon sto se otklone outlier-i za atribute, broj outlier-a se u ceni promenio sa 1206 na 1152 sto najverovatnije ukazuje da je pretpostavka tacna.

Dalje, napravljena je nova kolona *class* koja sadrzi jednu od cetiri klase opsega cene za predvidjanje. Nakon ovoga je uklonjena kolona *price* jer vise nije od znacaja.

Moze se uociti da skup podataka sadrzi 3 atributa koji opisuju dimenziju dijamanta, x, y i z, kao i 2 atributa koji opisuju odnos tih dimenzija, depth i table. Depth nam daje neke podatke o obliku dijamanta (100 * z / mean(x, y)), a table nam daje podataka o odnosu gornje povrsine i najsireg dela dijamanta. Iz ovih atributa je napravljen jedan atribut koji ce bolje uspeti da predstavi informaciju o dimenziji dijamanta, tako sto su pomnozene vrednosti x, y i z. Taj atribut je nazvan *volume* i predstavlja opisan kvadar oko dijamanta. Posmatrajuci oblik dijamanta, bilo bi racunski jako zahtevno izracunati stvarnu zapreminu. Bez obzira na taj nedostatak, nadamo se da ce algoritmi uspeti da uhvate informaciju i da dobiju "sliku" o zapremini dijamanta. Istu zapreminu mozemo dobiti za razlicite vrednosti x, y i z, a oblik dijamanta i odnos strana nisu zanemarljivi, bice ostavljeni atributi *depth, table, x, y, z* i prepustiti algoritmu da izabere sta mu najvise odgovara.

Svi atributi osim *carat* su normalizovani. *Carat* necemo dirati jer svakako ne odskace mnogo od ostatka i vecina je u opsegu od 0 do 1.

Na osnovu analize skupa uvidjeno je da postoji velika neuravnotezenost klasa. Najzastupljenija klasa cini 70%, dok najmanje zastupljena cini svega 3% S obzirom da imamo preko 50 000 instanci u skupu, 70% cini 35 000 instanci. Koriscenje tehnike poput SMOTE-a (Synthetic Minority Oversampling Technique) bi nam dovelo dataset na preko 140 000 instanci sto bi veoma povecalo vreme treniranja modela. Takodje, instanci u klasi "15000+" ima svega 3% a ona sadrzi najveci deo outliera i visoke cene, nesto sto bismo smatrali "ekstremnim vrednostima". Oversamplovanje ovakve klase sa 1500 instanci na 35 000

dovodi u rizik da nam se smanji konzistentnost skupa podataka, pogotovo ako novogenerisane instance ne bi bile dobra reprezentacija inicijalne distribucije. Nadalje cemo koristiti tehniku *weighting*, tamo gde bude potrebno. Ona podrazumeva da kao argument nasem klasifikacionom modelu prosledimo tezinu klase kao merilo paznje koju ce joj posvetiti.

2.1 Klasifikacija

Modeli korisceni za klasifikaciju su: *DecisionTreeClassifier, KNeighborsClassifier, MLPClassifier, CategoricalNB, RandomForestClassifier i XGBClassifier.*

DecisionTreeClassifier Drveta odlucivanja su hijerarhijske strukture koje se sastoje od cvorova i grana. Svaki cvor predstavlja testiranje odredjene osobine, dok grane predstavljaju moguce vrednosti te osobine. Na osnovu rezultata testiranja, model se krece kroz drveta sve do listova, gde se donose konacne odluke o klasifikaciji. Klasifikacija novih instanci se vrsi prolaskom kroz model drveta, pri cemu se na svakom cvoru primenjuje odgovarajuci test i prelazi se na sledeci cvor sve dok se ne stigne do lista. Na kraju se vrsi klasifikacija na osnovu odluke u listu.

Prednosti su u tome sto drveta odlucivanja lako razumljiva i mogu se vizualizovati u hijerarhijskoj strukturi. Takodje, pruzaju meru znacajnosti atributa, pokazujuci koji atributi najvise uticu na predvidjanja. Ove informacije mogu pomoci pri odabiru atributa. Drvo odlucivanja moze modelirati kompleksne nelinearne veze izmedju osobina i ciljne promenljive. Mogu uhvatiti interakcije i pragove, sto ih cini pogodnim za probleme u kojima linearne metode mozda nece dati dobre rezultate. Jos jedna od pogodnosti ovog modela je to sto je manje osetljiv na ekstremne vrednosti i nedostajuce vrednosti u poredjenju s nekim drugim algoritmima. Odluke se donose na osnovu serije podela, pa ekstremne vrednosti ili nedostajuce vrednosti ne uticu toliko na konacni rezultat. (Ranije je pomenuto da imamo outlier-e i da smo neke odlucili da zadrzimo.)

Prvobitno je odradjen obican *test-train-split* da bismo uvideli kako se model ponasa nad nasim skupom podataka. Napravljene su pomocne funkcije *model_report* i *feature_importance* za vizualizaciju evaluacije modela i znacajnosti atributa.

S obzirom da imamo neuravnotezenost klasa, a algoritam je osetljiv na tu neuravnotezenost, koristicemo parametar *class_weight* kako bismo dodelili tezine klasama. One su izracunate tako sto je za svaku klasu izracunat procenat koji zauzima u skupu a zatim reciprocnu vrednost toga pomnozimo sa 100.

Nakon ovoga, istreniran je prvi model i izvrsena evaluacija modela i znacajnosti atributa. Zatim je koriscen *GridSearchCV* kako bi bili pronadjeni najbolji parametri za model.

Tokom analize skupa podataka uvidjena je visoka korelacija nekih atributa, a i na osnovu analize znacajnosti atributa tokom treniranja modela, ocigledno je da se tehnikom *PCA (Principal Component Analysis)* moze smanjiti dimenzionalnost skupa podataka i mozda napraviti bolja predikcija. Pretpostavka

se ispostavila tacnom jer kumulativna suma 4 komponente iznosi preko 0.95. Ponovo je primenjena pretraga parametara kako bismo nasli najbolji model i uporedili ga sa rezultatima ostalih modela.

KNeighborsClassifier Ovaj algoritam se oslanja na slicnost izmedju instanci kako bi donosio odluke o klasifikaciji. Klasifikacija se vrsi na osnovu bliskosti izmedju instance koja se predvidja i njenih K najblizih suseda. Bliskost se obicno meri putem euklidskog rastojanja ili drugih metrika udaljenosti. Takodje, ovaj algoritam se moze primeniti na razlicite vrste podataka i dobro se nosi sa sumovima i outlier-ima.

Motivacija za ovaj model lezi u njegovoj jednostavnosti ali i efikasnosti. Naime, algoritam ne zahteva pretpostavke o distribuciji podataka ili obliku veza izmedju osobina i ciljne promenljive. *KNeighborsClassifier* se oslanja na lokalnost podataka. Klasifikacija se vrsi na osnovu slicnosti izmedju instance koja se predvidja i njenih najblizih suseda. Ovo je posebno korisno u problemima gde su slicne instance sklonije pripadati istoj klasi. Pored ovoga, algoritam pruza informaciju o verovatnoci pripadnosti svakoj klasi. Skalabilnost je takodje jedna od prednosti, u smislu velicine trening skupa podataka. Moze se lako primeniti na velike skupove podataka bez gubitka performansi.

Inicijalno je koriscen obican *train-test-split* kako bismo uvideli ponasanje na skupu podataka. Od pomocnih funkcija implementirana je *model_report* za evaluaciju modela. Za ovaj model je od velikog znacaja skaliranje atributa s obzirom da se mere rastojanja izmedju vrednosti. U preprocesovanju je izvrsena normalizacija vrednosti. Napravljeno je vise modela i primenje su tehnike *PCA i GridSearchCV* i svi modeli su uporedjeni kako bi se nasao onaj sa najboljim rezultatima.

MLPClassifier Algoritam se zasniva na neuronskim mrezama, posebno viseslojnim perceptronima, kako bi naucio kompleksne veze izmedju ulaznih osobina i ciljnih klasa. MLPClassifier koristi metodu propagacije unapred (feed-forward) kako bi preneo ulazne podatke kroz mrezu i generisao izlazne vrednosti. Zatim se koristi metoda propagacije unazad (backpropagation) kako bi se prilagodili parametri mreze na osnovu razlike izmedju predvidjenih izlaza i stvarnih vrednosti ciljnih klasa. Ovaj proces se iterativno ponavlja kako bi se minimizovala greska i optimizovali parametri mreze.

MLPClassifier je mocan alat za modelovanje kompleksnih veza izmedju ulaznih osobina i ciljnih klasa. Neuronske mreze imaju sposobnost nauciti i predstaviti nelinearne veze koje drugi linearni modeli ne mogu. S obzirom da su nam atributi uglavnom kontinualnog tipa, nadamo se da ce mreza uspeti da "uhvati" vezu izmedju atributa i klasa. Algoritam je takodje zadovoljavajuce otporan na sumove. Takodje, visoka prilagodljivost strukture same mreze je od velikog znacaja jer je mozemo prilagoditi nasem skupu podataka. Algoritam pokazuje visoke performanse kako sa kategorickim, tako i sa kontinualnim atributima.

Koristimo train-test-split kao i do sada kako bismo uvideli osnovne karak-

teristike modela. Od pomocnih funkcija tu su *model_report i plot_learning_curve* koja iscrtava krivu ucenja modela na trening skupu. Za model_report su korisceni *precision, recall i f1-score* kao i accuracy. Pored njih, funkcija i vizualizuje matricu konfuzije. Koriscene su tehnike *PCA i GridSearchCV*. U potrazi za najboljim parametrima isprobano je nekoliko jednoslojnih i viseslojnih arhitektura mreze kao i razlicite aktivacione funkcije. Nakon poredjenja svih modela i rezultata, ispostavlja se da se najbolje pokazala prvobitna konfiguracija parametara.

CategoricalNB Ovaj algoritam se zasniva na Naivnom Bajesovom pristupu i prilagodjen je za rad sa podacima koji imaju diskretne kategorije umesto kontinuiranih vrednosti. Naivni Bajesov pristup pretpostavlja nezavisnost izmedju ulaznih osobina, sto moze biti korisno kada je ta pretpostavka blizu istine. CategoricalNB primenjuje Naivni Bajesov pristup na podatke koji sadrze kategoricke osobine, kao sto su razlicite kategorije, labele ili oznake. Koriste se statisticke metode, posebno verovatnoce Bayesove teoreme.

CategoricalNB je pogodan za probleme klasifikacije gde se koriste kategoricki podaci. Takodje, poznat je po svojoj brzini izvrsavanja i otpornosti na sum u podacima. Buduci da se zasniva na radu sa verovatnocama, algoritam moze pravilno modelirati raspodelu kategorickih vrednosti, cak i u prisustvu suma ili nedostajucih vrednosti.

Od pomocnih funkcija koriste se vec opisana *model_report* i dodata je **plot_predicted_probabilities** s obzirom da nam algoritam nudi mogucnost da prikazemo verovatnocu da instanca pripada odredjenoj klasi. Ona iscrtava grafikon na kome su predstavljene verovatnoce. Istrenirano je vise modela, primenjene su tehnike PCA i GridSearchCV i uporedjeni su dobijeni rezultati. Ono sto se moglo odmah uociti je da u poredjenju sa prethodnim algoritmima, ovaj zaostaje po svim metrikama evaluacije koje smo izabrali. Razlog za to je najverovatnije visoka korelacija izmedju atributa i to sto imamo samo 3 kategoricka atributa dok su ostali kontinualni. Iako rezultat nije toliko los, opet je u velikom deficitu u poredjenju sa ostalim algoritmima.

RandomForestClassifier Algoritam se zasniva na konceptu ensemble metoda i kombinuje vise stabala odlucivanja kako bi donosio odluke o klasifikaciji kombinovanjem predvidjanja svakog stabla. Odluka se obicno vrsi vecinskim glasanjem, gde se klasa koja je najcesca medju svim stablima smatra konacnim predvidjanjem. Svako stablo se trenira na razlicitom podskupu podataka dobijenih metodom slucajnog izbora. Takodje, tokom konstrukcije svakog stabla, vrsi se slucajan izbor atributa koji se koriste za donosenje odluka na svakom cvoru.

Ovaj algoritam ima nekoliko prednosti, ukljucujuci visoku tacnost predvidjanja, otpornost na preprilagodjavanje i sposobnost rukovanja sa velikim skupovima podataka. Paralelizacija treninga stabala omogucava brzu obradu velikog broja instanci ili osobina, cime se stedi vreme obuke i predikcije. Neuravnotezenost klasa ne predstavlja problem, jer se algoritam zasniva na stablima odlucivanja, te se mogu pripisati tezine klasama. Kombinacija vise stabala od-

lucivanja smanjuje varijabilnost i greske koje se mogu pojaviti u pojedinacnim stablima, sto vodi do pouzdanih rezultata klasifikacije. Kao i algoritam stabla odlucivanja, može pružiti informacije o značajnosti atributa za klasifikaciju. Kombinacija vise stabala odlucivanja omogucava modelu da izgradi stabilne predikcije, uz minimalan uticaj suma ili nedostajucih vrednosti.

Od pomocnih funkcija se koriste *model_report* za evaluaciju modela, *plot_learning_curve* za vizualizaciju ucenja algoritma na trening skupu i *feature_importance* koja prikazuje znacaj atributa. Tezine klasa su identicno napravljene kao za stabla odlucivanja, reciprocnom vrednoscu procenta klase pomnozenu sa 100. Kao i za ostale, koriscene su tehnike PCA i GridSearchCV kako bismo evaluirali razlicite modele i kasnije ih uporedili.

Veoma zanimljiva informacija se moze uociti nakon prikazivanja znacaja atributa. Naime, novogenerisani atribut *volume* ima najvecu ocenu znacaja sa preko 18%. Time se pokazalo da je pravljenje te kolone veoma dobro uticalo na skup podataka. Takodje, jedan od modela je zabelezio, za sada, najvisi skor evaluacije modela.

XGBClassifier Ovaj algoritam pripada porodici gradijentnog boostinga i koristi ekstremni gradijentni boosting (XGBoost) kao osnovni mehanizam za poboljsanje tacnosti klasifikacije.

XGBClassifier koristi *ensemble* pristup i takodje je zasnovan na kombinovanju vise stabala odlucivanja, kako bi donosio odluke o klasifikaciji. Svako drvo odlucivanja se trenira iterativno, pri cemu se svaka iteracija fokusira na korekciju gresaka prethodne iteracije. Na taj nacin, model se postepeno poboljsava kako bi dao preciznije predvidjanje.

Poznati po svom efikasnom upravljanju velikim skupovima podataka, visokim tacnostima predvidjanja, robustnosti na sum i sposobnosti rukovanja sa razlicitim vrstama atributa, *Gradient booster*-i su najkorisceniji modeli za resavanje realnih problema. Takodje, XGBClassifier moze pruziti informacije o znacajnosti atributa za klasifikaciju jer se zasniva na stablima odlucivanja.

Vec od najosnovnijeg modela se mogla uociti visoka performantnost. Jedina mana ovog algoritma je sto je vreme treniranja obicno dosta dugo, pogotovo kada se koristi GridSearchCV za pretragu parametara. Takodje, moze se uociti da PCA tehnika ne poboljsava rezultate. XGBClassifier je sposoban da uoci i najsitnije korelacije izmedju atributa i da ih modeluje na pravi nacin. Jos jednom, nakon analize znacajnosti atributa, pokazano je da je kreiranje atributa *volume* bila jako dobra ideja.

2.2 Klasterovanje

Zeleli bismo da znamo da li u nasem skupu pdoataka postoje neka grupisanja instanci. Te informacije nam mogu pomoci u klasifikacionom problemu, na primer za odredjivanje klasa koje mozemo predvideti. Ako su klasteri jasno definisani i ako bismo klase pravili po klasterima, verovatno je da bismo poboljsali predvidjanja modela.

Skup podataka cemo obraditi tako sto cemo ga undersemplovati kako bismo izbalansirali klase, ali i da bismo ubrzali proces treniranja modela. Klase ce biti iste, kao za problem klasifikacije. Nakon obrade imamo nesto vise od 6 000 instanci sto je i dalje dovoljno za pouzdano treniranje modela, s obzirom da je selektovan skup reprezentativan. Ono sto nas najvise zanima je, da li je nasa podela na 4 klase dobra ili mozda postoji neka druga konfiguracija tih klasa koja bi dala bolje rezultate. Za reprezentaciju dobijenih rezultata bice korisceni atributi *carat* i *price*.

Modeli korisceni za klasterovanje su: *KMeans, AgglomerativeClustering, DBSCAN i SpectralClustering*

KMeans za cilj ima pronalazenje K klastera, pri cemu je K unapred odredjen broj klastera koje zelimo da formiramo. Koristimo ga zbog jednostavnosti implementacije i brzine izvrsavanja. Medjutim, treba imati na umu da KMeans moze biti osetljiv na inicijalnu postavku centroida klastera.

Kako bismo to resili, uvescemo pomocnu funkciju *plot_search* koja ce nam pokrenuti vise modela za zadati opseg (0, k]. Za meru evaluacije modela koristicemo *silhouette_score*. Pored ove funkcije imacemo jos jednu, *plot_centroids* koja ce nam sluziti za vizualizaciju klastera i centroida dobijenih iteriranjem modela. Uvedena je i jos jedna funkcija, preuzeta sa *scikit-learn* sajta, koja nam crta *silhouette diagram*.

AgglomerativeClustering pocinje sa svakom instancom podataka kao jednim klasterom. Zatim se iterativno spajaju parovi klastera koji su najblizi jedan drugom na osnovu definisane metrike udaljenosti, pri cemu se formira hijerarhijska struktura klastera. Ovaj proces se ponavlja sve dok se svi klasteri ne spoje u jedan veliki klaster ili dok se ne postigne neki unapred definisani kriterijum zaustavljanja. Na kraju procesa, rezultat moze biti predstavljen kao dendrogram koji vizualizuje hijerarhijsko povezivanje klastera. Prednosti aglomerativnog klasterovanja ukljucuju jednostavnost implementacije i mogucnost rukovanja razlicitim metrikama udaljenosti. Takodje, ovaj algoritam omogucava fleksibilnost u izboru broja klastera, buduci da mozemo odabrati zeljeni broj klastera posmatrajuci rezultujuci dendrogram. Od pomocnih funkcija koriste se, vec opisane, *plot_search* i funkcija za crtanje *silhouette diagram*-a.

DBSCAN je algoritam klasterovanja zasnovan na gustini koji se koristi za grupisanje podataka na osnovu njihove prostorne raspodele. Ovaj algoritam je sposoban da identifikuje klasterovanje u skupu podataka bez potrebe za unapred definisanim brojem klastera. DBSCAN razlikuje tri vrste tacaka:

pogodjene tacke (core points), susedne tacke (border points) i tacke suma (noise points). Pogodjene tacke se nalaze unutar klastera, susedne tacke su u blizini klastera, dok tacke suma ne pripadaju nijednom klasteru. DBSCAN ima nekoliko prednosti, ukljucujuci sposobnost otkrivanja klastera proizvoljnih oblika, otpornost na sum i sposobnost rukovanja promenljivim gustinama klastera. Od pomocnih funkcija tu je samo *plot_search*.

SpectralClustering je algoritam klasterovanja koji se koristi za grupisanje podataka na osnovu spektralne analize matrice slicnosti. Ovaj algoritam ima sposobnost otkrivanja slozenih struktura i nelinearnih veza medju podacima. Vazno je napomenuti da spektralno klasterovanje zahteva odabir odgovarajucih parametara, kao sto su broj klastera i metoda konstrukcije matrice slicnosti. Pravilan izbor ovih parametara je kljucan za postizanje kvalitetnih rezultata klasterovanja. Mana je to sto spektralno klasterovanje moze biti komputaciono zahtevno za velike skupove podataka. Nakon *plot_search*-a i analize *silhouette* koeficijenta za razlicit broj klastera, pazljivo su razmatrani svi rezultati i kako predstavljeni pomocu funkcije za crtanje *silhouette diagram*-a.

2.3 Pravila pridruzivanja

Apriori je algoritam za otkrivanje asocijativnih pravila u skupovima podataka. Algoritam funkcionise na principu pronalazenja cestih uzoraka u podacima. Asocijativna pravila su izrazi u obliku "A -¿ B", gde A i B predstavljaju skupove stavki, a strelica predstavlja implikaciju. Ova pravila se zasnivaju na merama *support* i *confidence*, koje se koriste za procenu znacajnosti i pouzdanosti pravila.

Modelovanje i obrada skupa podatka su radjeni u IBM-ovom SPSS-u. S obzirom da ovaj algoritam radi samo sa kategorickim atributima, odbaceni su kontinualni atributi. Skup podataka je takve prirode da nije podoban za pravila pridruzivanja jer se uglavnom koristi za regresiju (Cilj je predvidjanje cene dijamanata). Ali, pokusacemo da uvidimo da li mozda postoje neka pravila izmedju kategorickih atributa *clarity, color, cut* sa klasama koje smo predvidjali u klasifikaciji. Instancama je ponovo dodeljen novi atribut *class* koji predstavlja opseg cene u kom se cena instance nalazi.

3 Rezultati

Rezultati klasifikacije

Od svih istreniranih modela selektovani su najbolji predstavnici i uporedjeni su njihovi rezultati na nasem skupu podataka. Metrike za evaluaciju modela koje su koriscene su: *accuracy, precision, recall i f1-score*. Na kraju, uporedjena su vremena treniranja modela na skupu podataka radi ocene skalabilnosti.

Procenat tacnosti (*accuracy_score*):

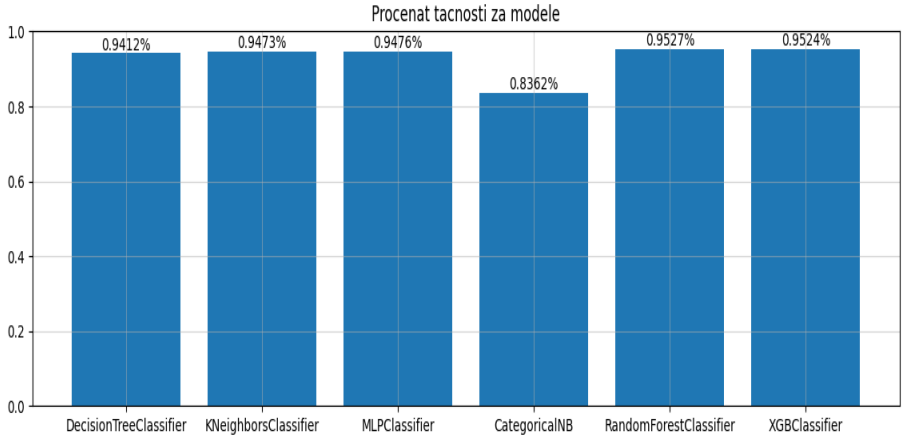


Figure 1: Accuracy_score za modele.

Svi modeli daju zadovoljavajuce rezultate osim CategoricalNB. Moze se uociti da najbolje rezultate daju RandomForestClassifier i XGBClassifier, sto je i ocekivano od ensemble modela.

Preciznost je metrika koja predstavlja odnos tačno pozitivno klasifikovanih instanci prema zbiru tačno pozitivno i lažno pozitivno klasifikovanih instanci. Rezultati za preciznost (*precision_score*):

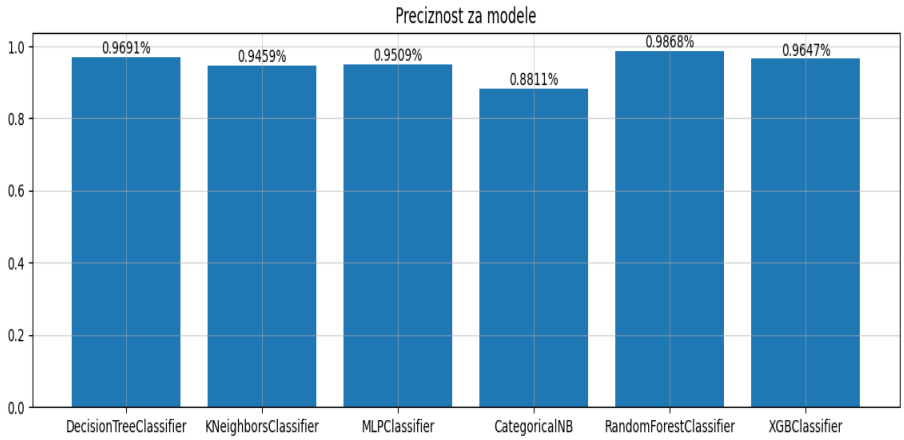


Figure 2: Precision_score za modele.

Odziv predstavlja odnos tačno pozitivno klasifikovanih instanci prema zbiru tačno pozitivno klasifikovanih i lažno negativno klasifikovanih instanci. Odziv meri sposobnost modela da pravilno identifikuje sve pozitivne instance. Rezultati za odziv (*recall_score*):

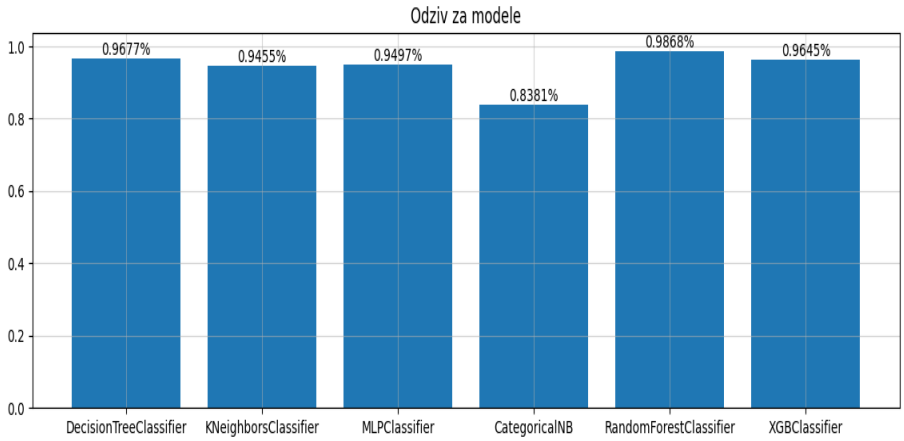


Figure 3: Recall_score za modele.

Mozemo primetiti da su accuracy_score i recall_score jednaki za svaki od 6 modela. Ako su rezultati preciznosti i odziva jednaki za model, to znaci da je model postigao istu performansu u pogledu tacnog identifikovanja pozitivnih

instanci i ukupne tacnosti svojih predvidjanja.

F1-skor je harmonična sredina preciznosti i odziva. On kombinuje informacije o tačnosti modela u klasifikaciji pozitivnih i negativnih instanci. Rezultati za F1-skor (*f1_score*):

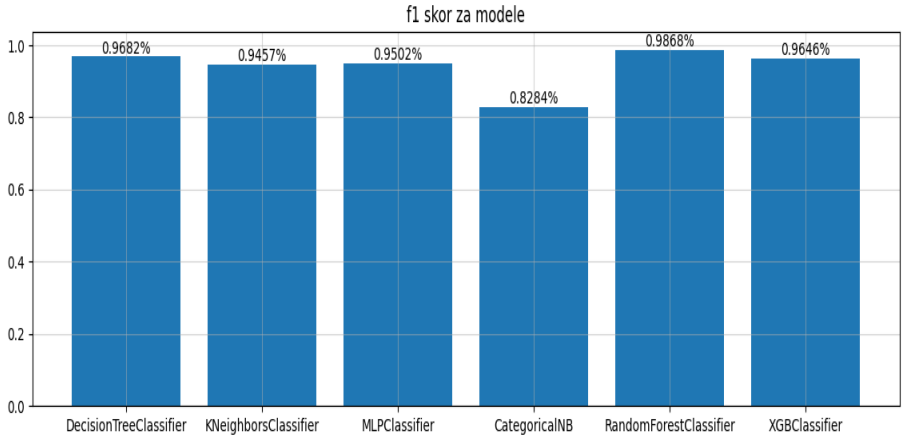


Figure 4: F1_score za modele.

Od svih modela **RandomForestClassifier** pokazuje najbolje rezultate. Za njim je odmah **XGBClassifier**. Ocekivano od *ensemble* modela. Ostali modeli daju zadovoljavajuce rezultate osim Bayesa. Razlog za to je najverovatnije visoka linearna korelacija izmedju kontinualnih atributa. Mada ni njegovi rezultati nisu mnogo losi.

S obzirom na generalno visoke performanse svih modela, ne bismo pogresili da za nas problem izaberemo bilo koji od njih ali kada govorimo o realnim problemima u obzir se mora uzeti i vreme potrebno da se svaki model istrenira i njegova mogucnost za rad na jako velikim skupovima podataka. S toga smo izmerili vreme treniranja modela i uporedili im rezultate. Potrebno je odluciti da li je ta mala razlika u preciznosti pogadjanja vredna u odnosu na razliku vremenskih cena modela.

Ako bismo birali prostije modele i dvoumili se, na primer, izmedju KNeighborsClassifier-a ili MLPClassifer-a, ova statistika nam u tome moze pomoci:

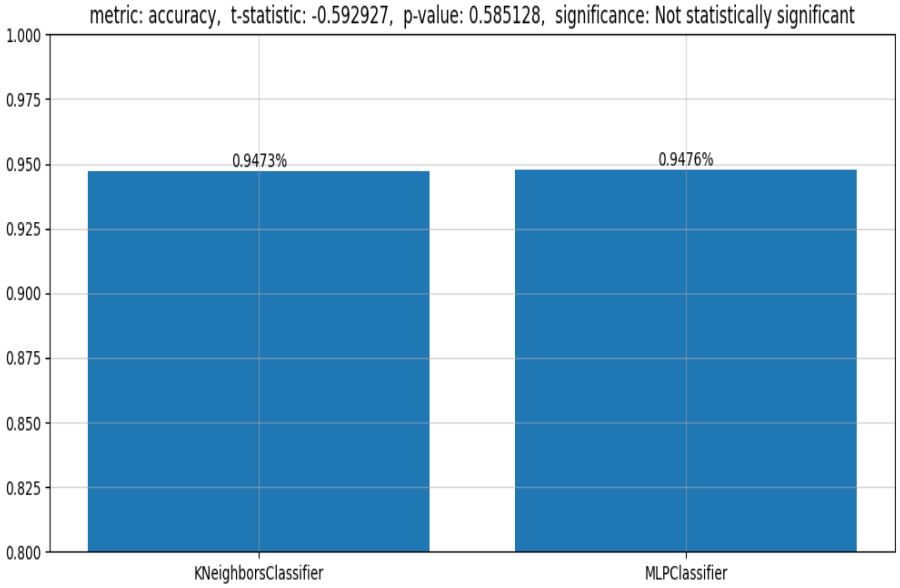
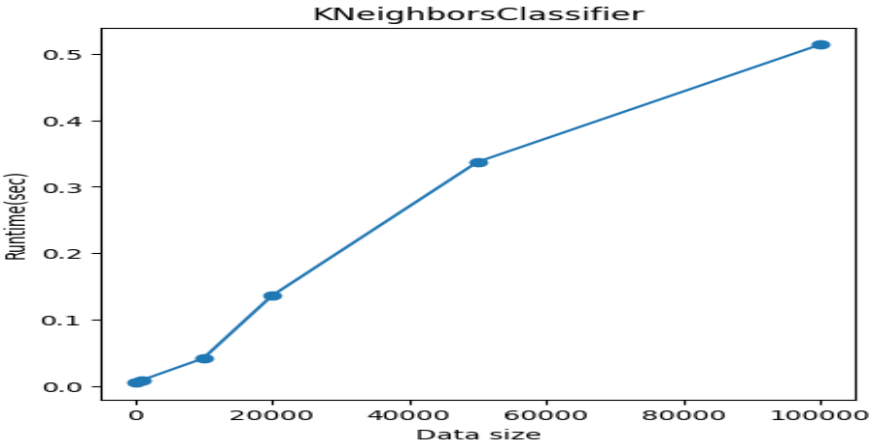


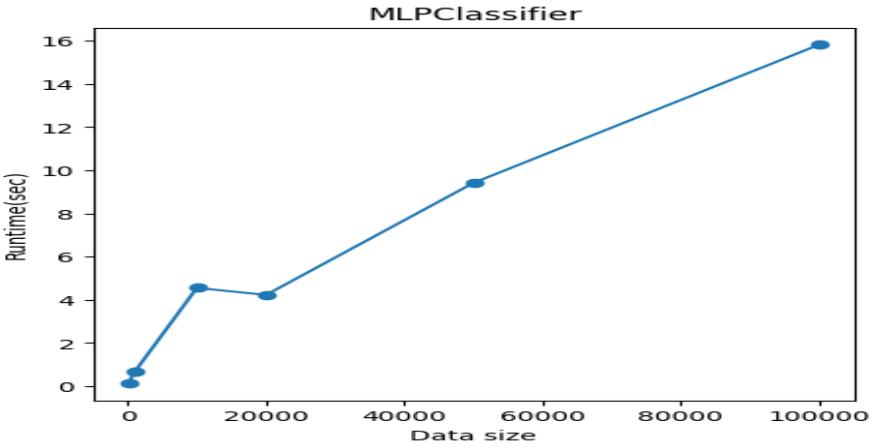
Figure 5: KNN vs MLP

MLPClassifier jeste model koji daje malo bolje rezultate. Ali mozemo videti da ta razlika je tek 0.003% u tacnosti predvidjanja. Da li je tih 0.0003% vredno vremenske cene potrebne da se istrenira cela neuronska mreza?

Pogledajmo sada vreme izvrsavanja za KNeighborsClassifier:



Pogledajmo sada vreme izvrsavanja za MLPClassifier:



Mozemo uociti da razlika u vremenu nije zanemarljivo mala. KNeighborsClassifier se na skupu podataka od 100 000 instanci istrenira za 0.5 sekundi dok je, za isti skup podataka, MLPClassifier-u potrebno cak 16 sekundi. Namece se pitanje sta bismo radili da imamo milion instanci u skupu podataka? U ovakvom slucaju bismo "zrtvovali" 0.0003% tacnosti zarad 32 puta brze treniranje modela.

Ako se ipak odlucimo za kompleksnije *ensemble* modele poput XGBClassifier-a ili RandomForestClassifier-a, trebalo bi uzeti u obzir da njihova visoka moc predvidjanja i modeliranje kompleksnih veza izmedju atributa, dolazi sa visokom vremenskom cenom potrebnom da se te veze uoce.

Ako uporedimo njihove ocene tacnosti videcemo da imamo slicnu situaciju, jedan model je bolji ali je takav inkrement statisticki neznacajan:

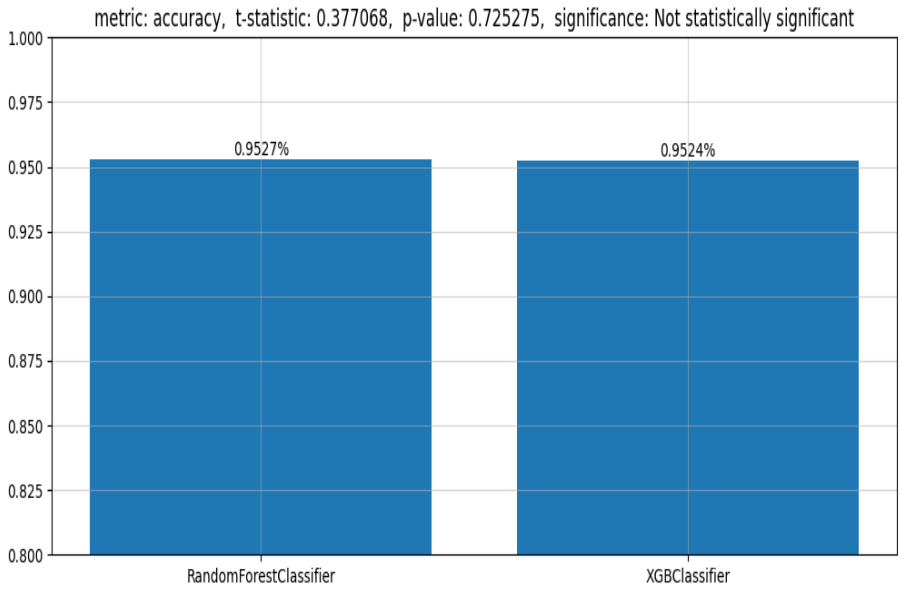
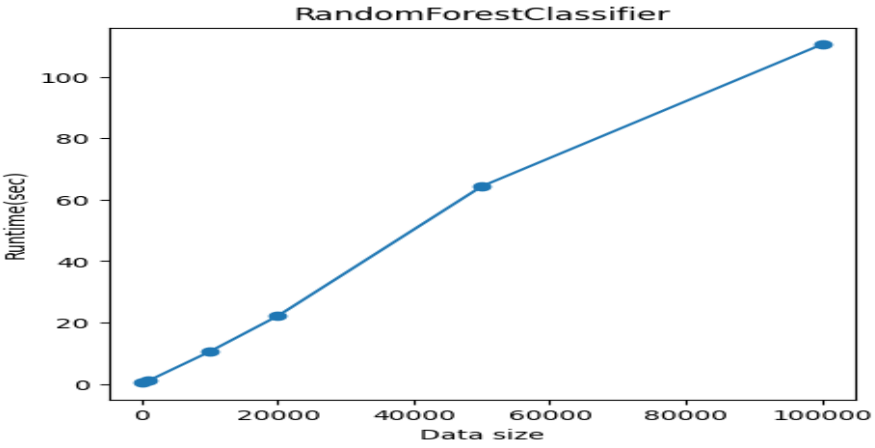


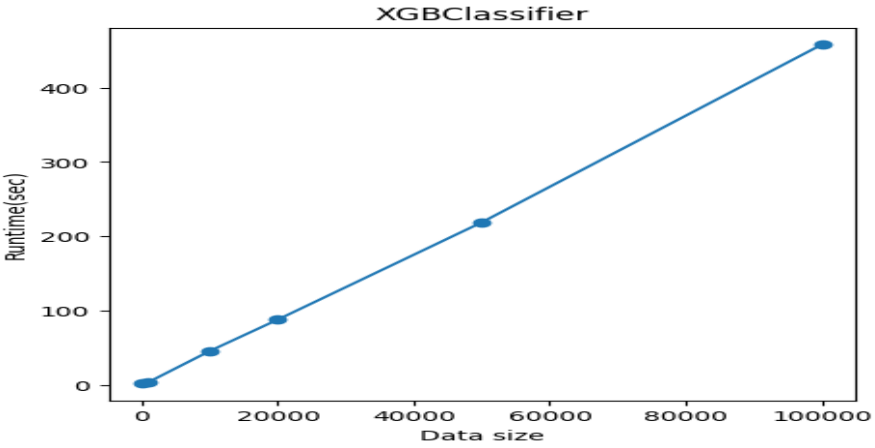
Figure 6: RFC vs XGB

Ponovo imamo istu razliku od 0.0003%. Ponovo nam se namece pitanje da li je to vredno razlike u vremenima potrebnim za treniranje modela. Od *ensemble* modela generalno ocekujemo duze vreme treniranja pa ta razlika moze biti od ogromnog znacaja.

Pogledajmo sada vreme izvrsavanja za RandomForestClassifier:



Pogledajmo sada vreme izvrsavanja za XGBClassifier:



Posle pazljivog razmatranja, uoceno je da je RandomForestClassifier cak 4 puta brzi od XGBClassifiera. Ta razlika na ovoj razmeri je veoma bitna. Svakako je RandomForestClassifier i tacniji model.

Nakon uzimanja u obzir svih metoda evaluacije modela koji su korisceni, ocigledno je zakljuciti da se najbolje pokazao **RandomForestClassifier**.

Rezultati klasterovanja

Metrike za evaluaciju modela klasterovanja koje su koriscene su: *Silhouette score, Dunn index, Rand index i Intra-cluster distances.* Razmatrani modeli korisceni za klasterovanje su: *KMeans, AgglomerativeClustering, DBSCAN i SpectralClustering.*

Silueta skor je metrika koja meri koliko dobro je svaki objekat dodeljen odgovarajucem klasteru, uzimajuci u obzir gustinu i razdvajanje klastera. Vrednosti Silhouette koeficijenta se krecu u opsegu od -1 do 1. Vece vrednosti silueta koeficijenta ukazuju na bolje klasterovanje, pri cemu vrednost bliska 1 ukazuje na dobro definisane i odvojene klastera, dok vrednost bliska -1 ukazuje na objekte koji su pogresno dodeljeni klasterima. Silueta skor za modele:

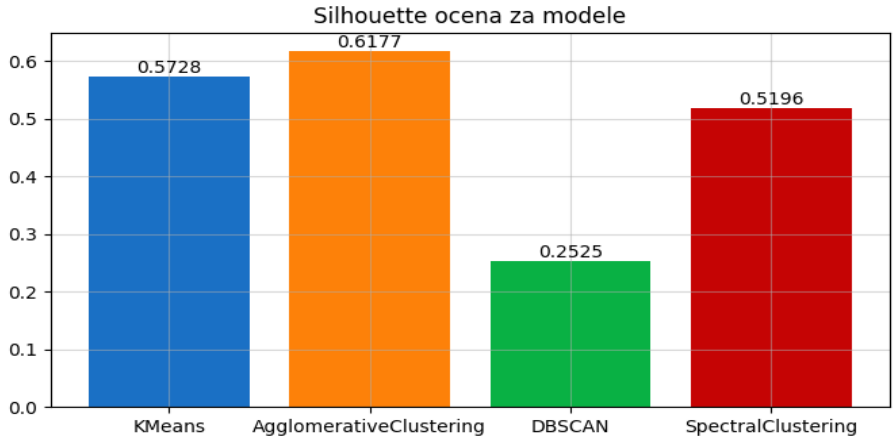


Figure 7: Silhouette score

AgglomerativeClustering daje najbolje rezultate za svoja 2 klastera.

Dunn index je metrika koja meri razdvajanje izmedju klastera i gustocu objekata unutar klastera. Visoke vrednosti Dunn indeksa ukazuju na dobro odvojene klastera i dobru gustinu unutar klastera, dok niske vrednosti ukazuju na manje jasno definisane klastere i vecu rasprsenost objekata.

Dunn index za modele:

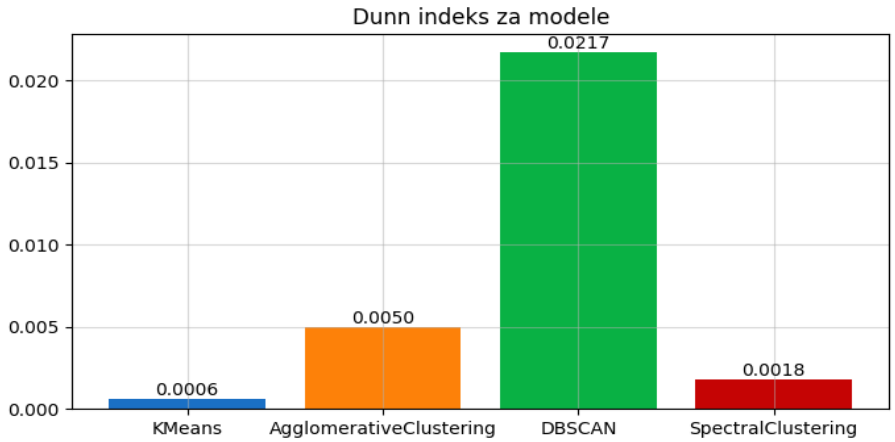
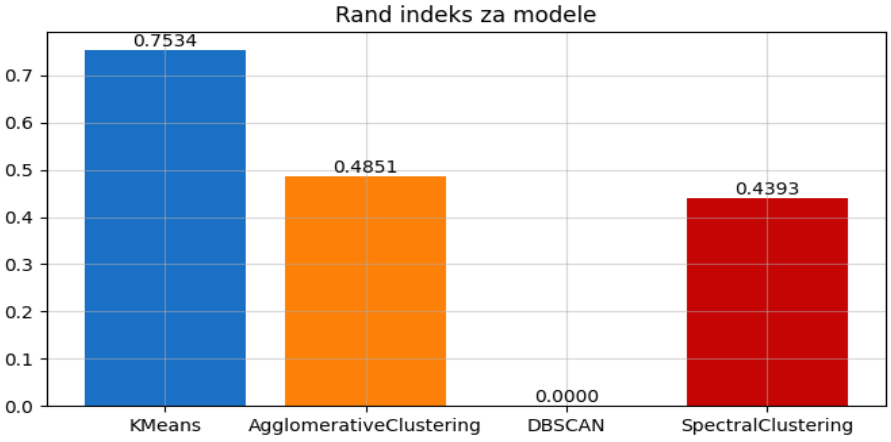


Figure 8: Dunn index

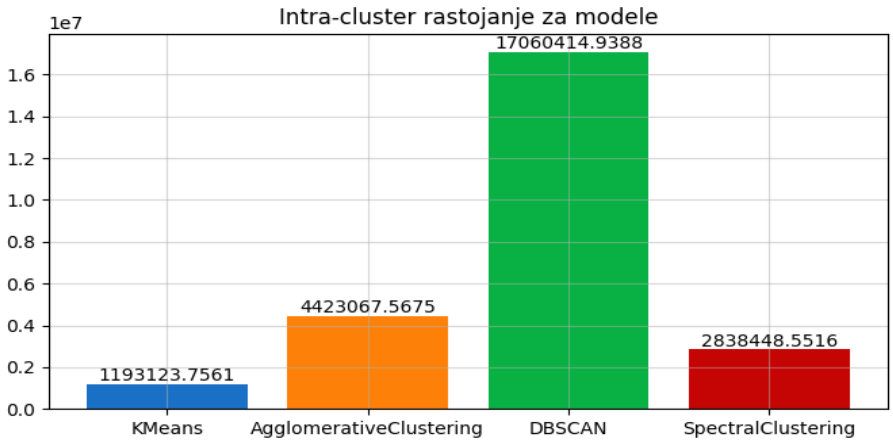
U prvoj metrici bismo odmah otpisali DBSCAN, ali mozemo videti da po ovoj metrici daje daleko bolje rezultate nego ostali. To ne treba da nas zavara jer kada uzmemo u obzir sta meri ova metrika i prisetimo se koji su klasteri odredjeni (Slika 15), jasno nam je zasto je ovakav rezultat.

Rand indeks je metrika koja se koristi za merenje slicnosti izmedju predvidjenih klastera i stvarnih klasa (ukoliko su dostupne). Ova metrika ocenjuje koliko su tacno klasteri modela uskladjeni sa pravim klasama. Vrednosti *Rand indeksa* se krecu u opsegu od 0 do 1. Rand indeks za modele:



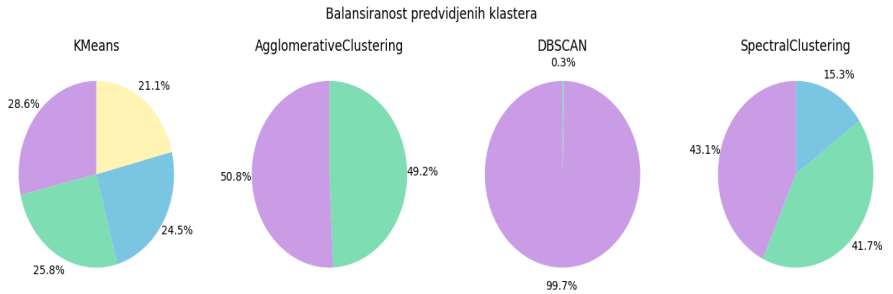
Sada mozemo uociti da se KMeans najbolje pokazao. DBSCAN ima najnizu mogucu ocenu.

Intra-klaster rastojanje je metrika koja se koristi za merenje prosecne udaljenosti izmedju objekata unutar istog klastera. Ova metrika ocenjuje kompaktnost i gustinu klastera tako sto racuna prosecno rastojanje izmedju svih parova objekata unutar istog klastera. Manje vrednosti intra-klaster rastojanja ukazuju na bolje klasterovanje, jer ukazuju na to da su objekti unutar klastera blizi jedni drugima. Intra-klaster rastojanje za modele:



S obzirom na prirodu ove metrike, mozemo uociti da je KMeans ponovo najbolji dok je DBSCAN daleko losiji od ostalih.

Sada cemo pogledati predvidjene klastere i njihovu balansiranost:



Na osnovu svega ovoga, moze se zakljuciti:
KMeans nam pokazuje da postoje 4 solidno balansirana klastera. Odgovara nasem klasifikacionom problemu, sto potvrdjuje da je podela na 4 opsega cene dobra odluka.

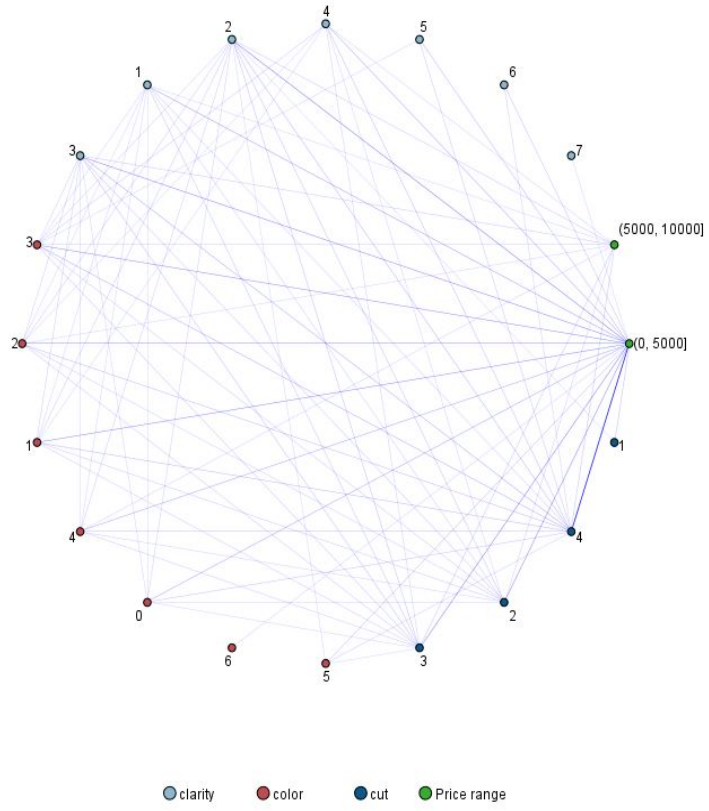
AgglomerativeClustering nam pokazuje dobru podelu na dva klastera cija je odlicna balansiranost.
DBSCAN nam pokazuje da ipak postoji velika neuravnotezenost po odredjenom atributu. Taj atribut je carat.
SpectralClustering nam pokazuje da podela na 3 klastera nije losa ideja

Rezultati pravila pridruzivanja

Apriori algoritam nam je pokazao sledece rezultate za kategoricke atribute *clarity, color, cut* i njihovu vezu sa *class* atributom:

Consequent	Antecedent	Support %	Confidence %	Lift
Price range = (0, 5...	color = 1	18.163	82.382	1.133
Price range = (0, 5...	color = 0	12.56	81.063	1.115
Price range = (0, 5...	cut = 4	39.954	76.869	1.057
Price range = (0, 5...	color = 2	17.69	75.959	1.045
Price range = (0, 5...	cut = 2	22.399	72.331	0.995
Price range = (0, 5...	clarity = 2	24.221	71.496	0.983
Price range = (0, 5...	clarity = 4	15.148	71.068	0.977
Price range = (0, 5...	color = 3	20.934	70.262	0.966

Izdvojeno je 8 pravila. Sva pravila predvidjaju da se instance nalaze u prvom opsegu. Granice za *support* i *confidence* su 10% i 70%. Ako pogledamo i *lift*, mozemo zakljuciti da nam nijedno pravilo nije preterano interesantno, te na nasem skupu podataka nema uocljivih pravila od nekog velikog znacaja. Maksimalan support je nesto manje od 40% a maksimalan confidence je nesto vise od 82%. Jedino pravilo od nekog znacaja je prvo, da iz color=1 sledi pripadanje klasi (0, 5000]. Ni graf mreze nam ne pruza neke zanimljive podatke:



4 Zakljucak

Nakon primenjivanja razlicitih tehnika istrazivanja podataka i masinskog ucenja na skupu podataka *diamonds* iz biblioteke *tensorflow_datasets*, istrenirano je oko 10 modela i preispitane su karakteristike svakog od njih.

Sto se skupa podataka tice, podaci ispunjavaju kvantitativne i kvalitativne mere te se mogu koristiti za pouzdano predvidjanje cenovnih klasa dijamanata, u opsegu do 20 000USD.

Klasifikacija nam je pokazala da nam je najbolji model za nas skup podataka **RandomForestClassifier**.

Klasterovanje nam je pokazalo da postoji vise mogucih podela kontinualnog opsega cene i da mozemo dobiti dobre rezultate za **2, 3 ili 4** klase.

Pravila pridruzivanja su nam samo potrvdila ocigledno, da nisko kvalitetni dijamanti teze tome da kostaju manje.