**Matematički fakultet u Beogradu**

**Seminarski rad iz Istraživanje Podataka 1**

Analiza skupa podataka   
„Data Mining Amazon reviews Dataset”

**Nikola Veselinović 200/2015**

Sadržaj

[Uvod 1](#_Toc137832506)

[Analiza podataka 2](#_Toc137832507)

[Predprocesiranje podataka 3](#_Toc137832508)

[Klasifikacija 5](#_Toc137832509)

[Stablo odlučivanja 5](#_Toc137832510)

[Slučajna šuma 7](#_Toc137832511)

[K Najbližih Suseda (KNN) 9](#_Toc137832512)

[Naivni Bajes 11](#_Toc137832513)

[Poređenje modela 12](#_Toc137832514)

[Klasterovanje 15](#_Toc137832515)

[Zaključak 15](#_Toc137832516)

# Uvod

Cilj ovog rada je detaljna analiza podataka iz skupa podataka „Data Mining Amazon reviews Dataset”[[1]](#footnote-1) kao i konstrukcija modela koji mogoćava predviđanje autora komentara na osnovu samog sadržaja komentara.

„Data Mining Amazon reviews Dataset” je derivat skupa podataka „customer reviews“ sa sajta Amazona koji je namenjen za identifikaciju autora komentara. Identifikovano je 50 najaktivnijih korisnika (sa jedinstvenim identifikacionim brojem i korisničkim imenom) na ovoj platformi. Za svakog korisnika je prikupljeno po 30 komentara. Za atribute koriste se 10000 različitih tekstualnih struktura koje su se najčešće pojavljivale u komentarima. Atributi uključuju informacije o stilu pisanja autora, kao što je upotreba cifara, interpunkcije, dužina reči i rečenica, kao i učestalost reči i slično.

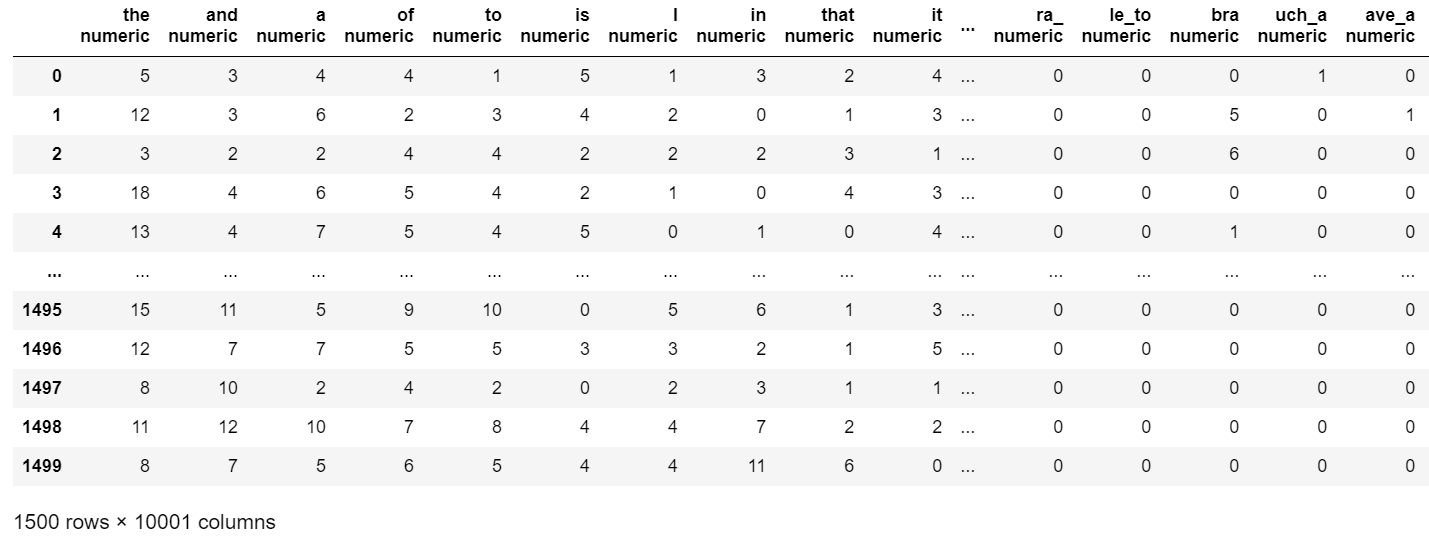
|  |  |  |  |
| --- | --- | --- | --- |
| **Karakteristike skupa podataka** | Više-parametarski, tekst, teorija domena | **Karakteristike atributa** | Celobrojni |
| **Predviđen zadatak** | Klasifikacija | **Izostavljene vrednosti** | Nema |
| **Broj instanci** | 1500 | **Broj parametara** | 10000 |

# Analiza podataka

Pre početka rada sa podacim potrebno je upoznati se sa samom bazom podataka. Podatci su upakovani u .arff fajlu koji je formatiran tako da se segmenti odvajaju sa @segment iza čega idu podatci. Bitni segmenti su **@relation**: koji prati informacija o tome na šta se odnosi ovaj fajl, **@attribute**: koji prati ime i tip atributa (npr. @attribute have numeric), ovaj segment se ponavlja onoliko puta koliko imamo atributa, pored konkretnog tipa atributima možemo proslediti listu mogućih vrednosti, što i koristimo u ovom modelu da proseldimo listu autora, i za kraj **@data** koji je indicator da posle njega svaki red predstavlja jedan red iz tabele podataka, a kolone u redu su razdvojene zarezima.

U našem skupu podataka postoji 10001 kolona, od kojih prvih 10000 su numeričkog tipa, a poslednja kolona ima ime autora kao svoju vrednost, i 1500 redova.

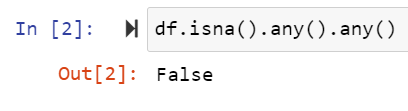
**@attribute class** *{Agresti,Ashbacher,Auken,Blankenship,Brody,Brown,Bukowsky,CFH,Calvinnme,Chachra,Chandler,Chell,Cholette,Comdet,Corn,Cutey,Davisson,Dent,Engineer,Goonan,Grove,Harp,Hayes,Janson,Johnson,Koenig,Kolln,Lawyeraau,Lee,Lovitt,Mahlers2nd,Mark,McKee,Merritt,Messick,Mitchell,Morrison,Neal,Nigam,Peterson,Power,Riley,Robert,Shea,Sherwin,Taylor,Vernon,Vision,Walters,Wilson}*



Slika 1 Prikaz podataka

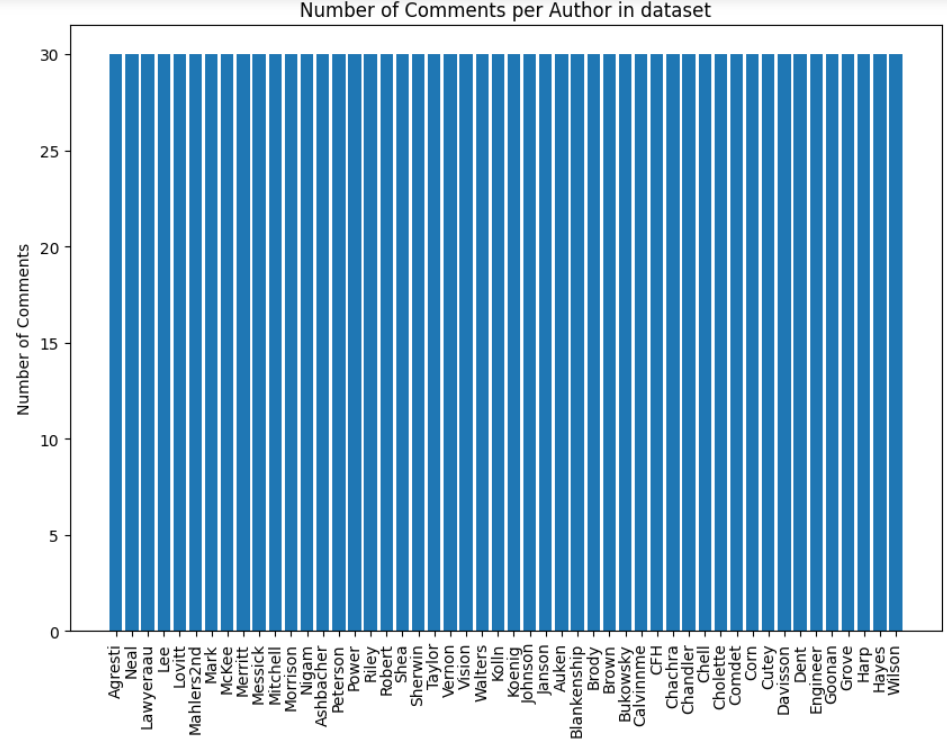
Za svaku od klasa (autora) imamo po 30 redova u skupu podataka.

# Predprocesiranje podataka

Posle analize podataka potrebno je obraditi podatke tako da budu spremni za formiranje modela klasifikacije ili za klasterovanje. Ovo se dešava iz nekoliko koraka. Prvo ćemo proveriti da li su nam sve vrednosti adekvatne u skupu podataka, zatim ćemo vrštiti podelu na test i terning skupove i standardizaciju podataka ili pronaći najuticajnije atribute i izdvojiti ih za klasterovanje.

U ovom skupu podataka nema nedostajućih vrednosti, nema dupliranih imena atributa, i nema dupliranih klasa. Funkcija koju smo napravili za učitavanje podataka ima implementiranu proveru da li postoje duplirani atributi i u slučaju da postoje ona ih spaja (i javlja da postoji takav slučaj i za koji argument). Da ovo nije slučaj mi smo mogli da postavimo nedostajuće vrednosti na određenu vrednost (npr. 0, što bi za ovaj skup podataka najviše imalo smisla), ili na srednju vrednost svih vrednosti kolone ili na srednju vrednost vrednosti kolone koje pripadaju istoj klasi.

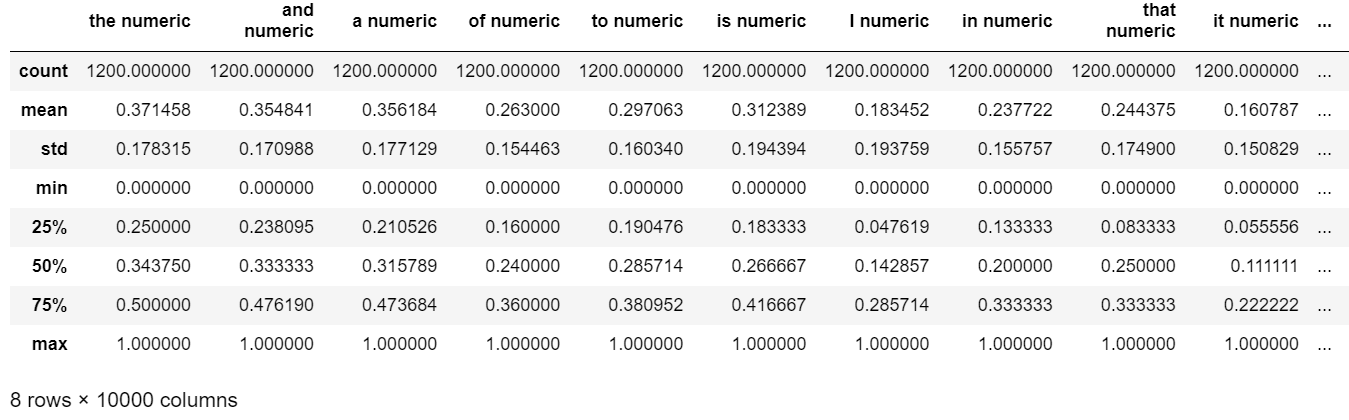
Slika 2 Provera za nedostajuće vrednosti



Slika 3 Broj redova po klasi

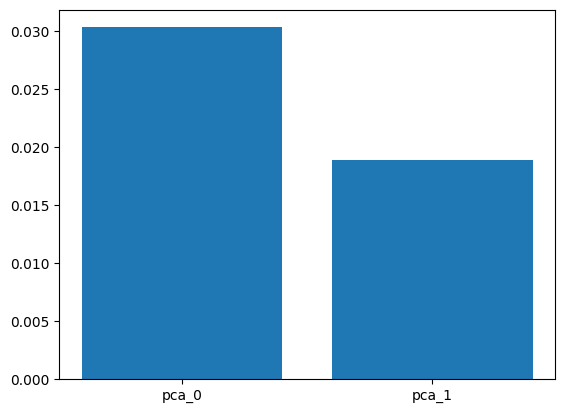
Kao priprema za klasifikaciu mi delimo skup na test i trening slupove, tako da *80%* skupa se odvaja za treniranje, a *20%* za testiranje. Ovo se postiže korišćenjem funkcije *train\_test\_split* iz biblioteke *sklearn.model\_selection*. Nešto što je bitno za rezultate u našem slučaju je postavljanje atributa *stratify=Y*, sto se stara o tome da procentualno jednak broj redova svake klase se nađe u svakom od skupova. Kada to ne bi uradili moglo bi da dodje do situacije gde za neku klasu svi redovi ili većina njih se nalazi u jednom od skupova što dovodi do toga da ili nema dovoljno informacija za trening ili nema dovoljno testnih primera za pouzdano ocenjivanje validnosti dobijenog modela.

Nešto što može bitno da utiče na rezultat kreiranje modela je razlika u broju pojavljivanja određenih reči u rečenicama. Ovo se odnosti na to da rečice ili veznici se daleko češće koriste nego individualne imenice ili glagoli. Zato mi treba da skaliramo podatke tako da nijedan atribut ne bude značajno bitiniji nego neki drugi samo zato što su njegove vrednosti veće dimenzije nego kod drugih. Ovo postižemo skaliranjem podataka. Izabrali smo da koristimo MinMaxSkaliranje jer sve vrednosti u skupu podataka (sem klase) su numeričke vrednosti iz skupu N0. Ovo skaliranje sve vrednosti jedne kolone skalira tako da se skup [mink,maxk] preslikava u skup [0,1]. Za ovo koristimo funkciju *MinMaxScaler* iz biblioteke *sklearn.preprocessing*.

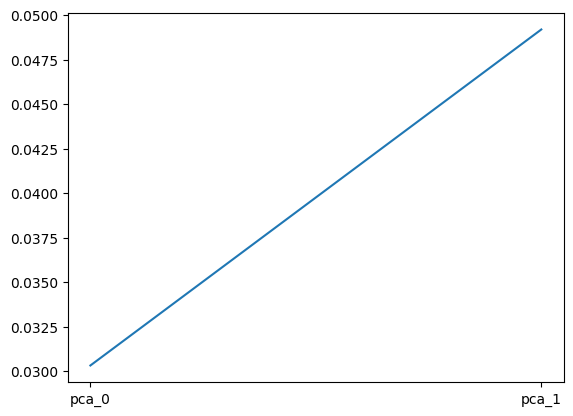


Slika 4 Statistika skaliranih vrednosti

Za klasifikaciju prvo koristimo *OrdinalEncoder* nad celim skupom podataka da promenimo sve tekstualne vrednosti (u našem slučaju imena autora) u jedinstvene celobrojne vrednosti. Koristimo MinMaxScaler da normalizujemo vrednosti. Zatim koristimo *PCA (Principal Component Analysis)* da smestimo nše podatke u 2D prostor.



Slika 5 Nivo važnosti pca vrednosti



Slika 6 Suma važnosti odabranih psa komponenti

# Klasifikacija

U ovom segmentu ćemo pokazati 3 različita algoritma za klasifikaciju i uporediti njihove rezultate.

Algoritmi koje koristimo:

1. Stablo odlučivanja
2. Slučajna šuma
3. K najbližih suseda
4. Naivni Bajes

Kako bi pronašli što optimalnije parametre za algoritme koristimo *GridSearch* iz biblioteke *sklearn.model\_selection*. Prosleđujemo vrednosti i model koji se zatim testiraju i upoređuju.

## Stablo odlučivanja

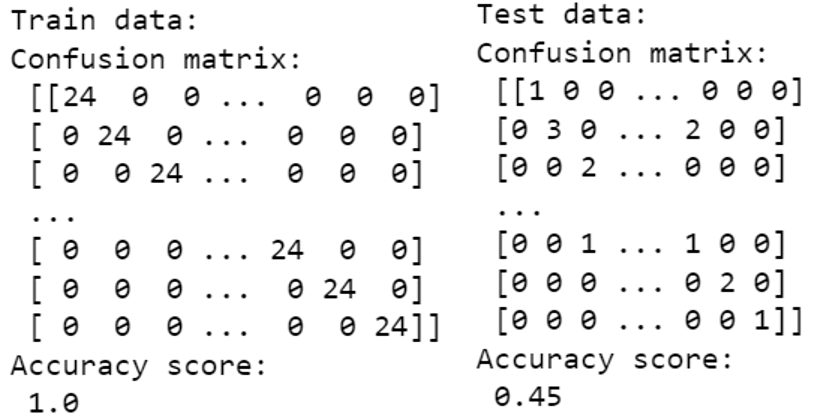
Kao prvi algoritam koji ćemo da razmatramo uzimamo Stablo odlučivanja (*Decision Tree*). On se zasniva na razdvajanju podataka na osnovu atributa kako bi se došlo do konačne klasifikacije. Algoritam radi na principu izgradnje hijerarhijske strukture stabla, gde svaki čvor vrši testiranje atributa, dok grane odgovaraju ishodima testa. Proces izgradnje stabla odlučivanja se odvija iterativno. Na početku, algoritam bira atribut koji najbolje deli podatke na različite klase. Ovaj izbor atributa se vrši na osnovu nekog kriterijuma, kao što je Gini indeks ili entropija, koji meri koliko dobro atribut razdvaja podatke.

Nakon izbora atributa, stablo se grana na osnovu mogućih vrednosti tog atributa. Postupak se rekurzivno ponavlja za svako podstablo sve dok se ne dostigne krajnji čvor stabla ili se postigne određeni uslov zaustavljanja, kao što je dostizanje maksimalne dubine stabla ili nedovoljan broj podataka za dalje grananje.

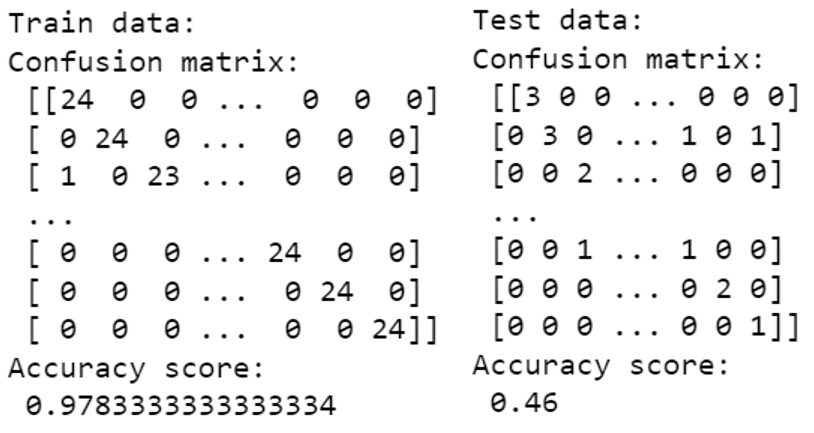
Kada se stablo izgradi, klasifikacija novih podataka se vrši prolaskom kroz stablo od korena do listova, gde list u kome se završi predstavlja klasu koju dodeljujemo novom podatku.

Prvo ćemo probati da iskoristimo *DecisionTreeClassifier* iz biblioteke sklearn.tree bez modifikacije parametara kako bi dobili base line rezultate. Ono sto možemo da primetimo je da tokom treninga imamo 100% tačnost, a tokom testiranja imamo 45% tačnost.

Koristićemo GridSearch i proslediti mu sledeće parametre: *'criterion': ['gini', 'entropy'], 'max\_depth': [20, 30, 35, 40, 45, 50, 60, 100, 120, 200].* Za maksimum smo stavili 200 jer se pokazalo da je optimum oko 40, pa smo smanjili zbog dužine izvršavanja. On će pokrenuti DecisionTreeClassifier za svaku kombinaciju prosleđenih parametara i onda možemo izvući najbolju kombinaciju. Primetićemo da u ovom slučaju naše odabrane vrednosti su pokazale i u najboljem slučaju malo bolje rezultate od inicijalno postavljenih. Najbolji parametri: *{'criterion': 'gini', 'max\_depth': 40}.*



Slika 7 Statistika Stabla odlučivanja bez modifikacije parametara



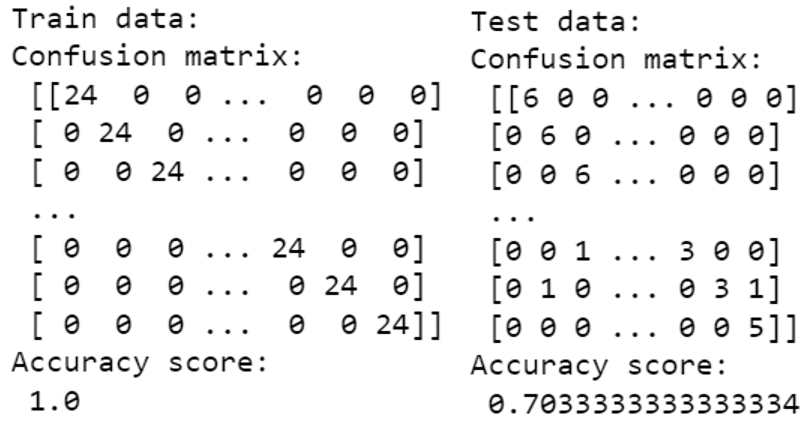
Slika 8 Rezultati za najbolju kombinaciju prosleđenih parametara za Stabla odlučivanja

## Slučajna šuma

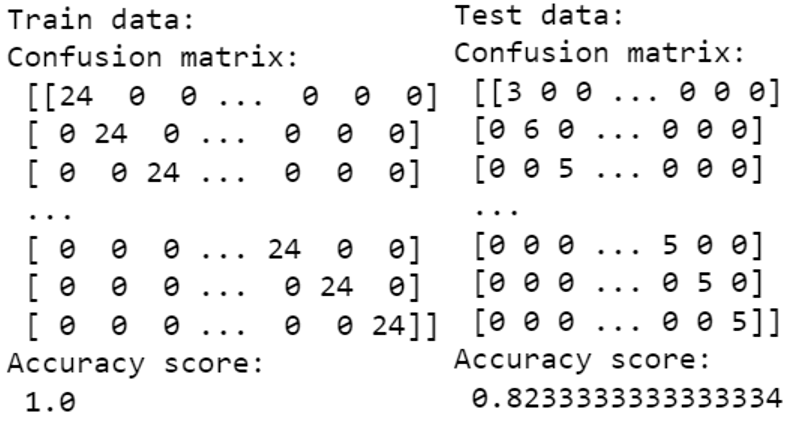
Slučajna šuma (Random Forest) je ansambla stabala odlučivanja, gde se više stabala odlučivanja kombinuje kako bi se postigla bolja klasifikacija. Svako stablo se trenira na nasumično izabranim podskupovima podataka, a klasifikacija se vrši glasanjem više stabala. Ovaj korak smanjuje preprilagođenost, povećava stabilnost, daje viši nivo generalizacije, bolji rad sa velikom količinom podataka i atributa, i višu otpornot na nedostajuće vrednosti.

Koristimo RandomForestClassifier iz biblioteke sklearn.ensemble. Vec po prvom rolazu možemo da vidimo bolje rezulate nego kod običnog stable odlučivanja sa čak ~25% boljim rezultatima u testnoj fazi. U trening fazi možemo da vidimo 100% tačnosti dok u test fazi imamo čak ~70% tačnosti.

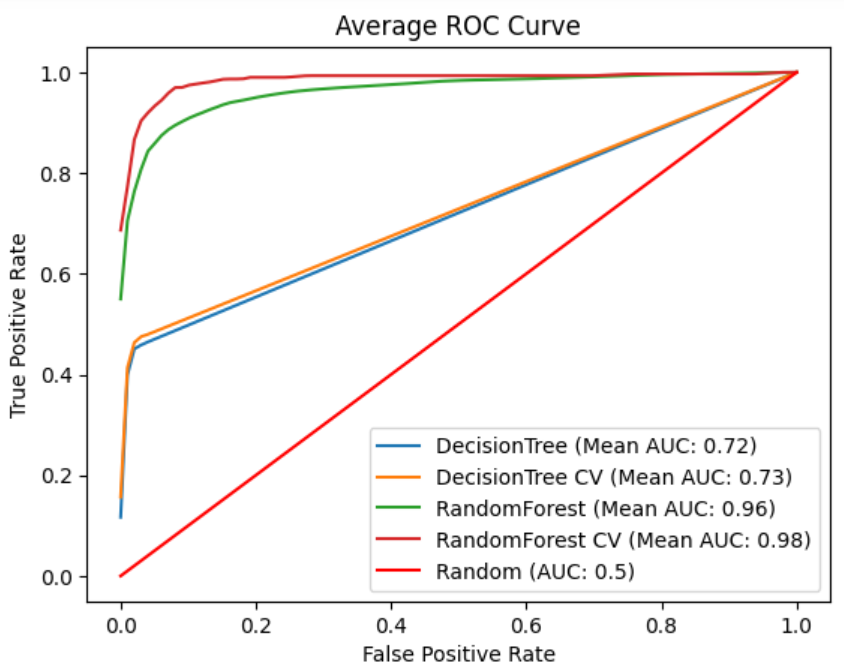
Kako bi pokušali da dobijemo što bolje rezultate i ovde ćemo primeniti GridSearch sa sledećim parametrirma: *'criterion': ['gini', 'entropy', 'log\_loss'], 'n\_estimators': [50, 100, 200, 300, 500, 1000].* Rezultati bi potencijalno bili bolji sa još većom dubinom stable ali zbog dugog trajanja obrade postavili smo maksimalnu dubinu sa kojom radimo na 1000. Možemo videte da je ovde menjanje parametara zaista imalo veliki uticaj na rezultate sa čak ~82% tačnosti u fazi testiranja za model sa parametrima: {'criterion': 'gini', 'n\_estimators': 1000}.



Slika 9 Rezultati algoritma slučajne šume



Slika 10 Rezultat algoritma slučajne šume za najbolje izabrane parameter



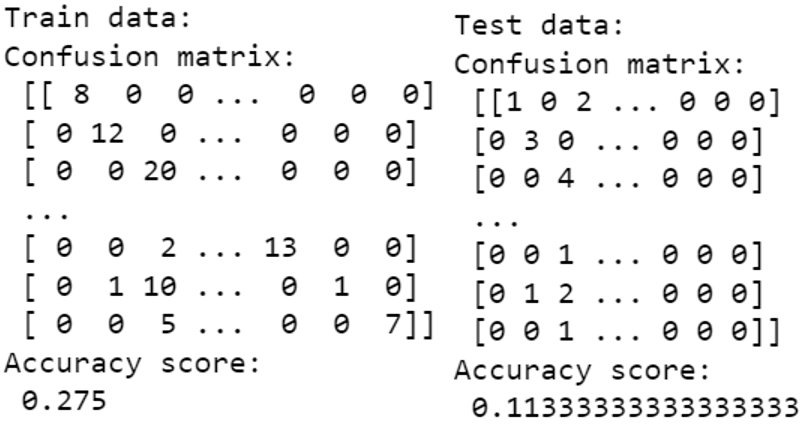
Slika 11 ROC kriva za algoritme Stabla odlučivanja i Slučajne šume

## K Najbližih Suseda (KNN)

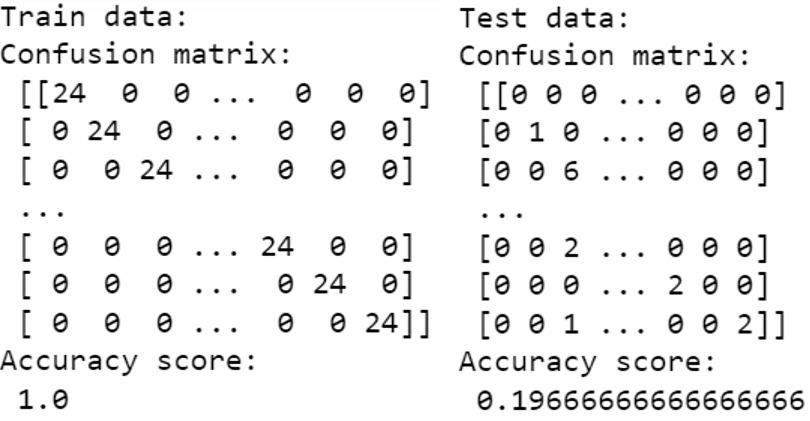
K najbližih susesda (K Nearest Neighbors) algoritam kao što mu ime kaže dodeljuje klasu čvoru tako što posmatra koje su klase većina od njegovih K najbližih suseda. KNN može biti manje efikasan zbog potrebe za skladištenjem celog skupa podataka i računanja rastojanja za svaku instancu, i mi očekujemo da će to biti sada slučaj. Veliki uticaj na reultate ima odabir parametra K, bilo da je preveliki ili premali, može da ima negativne rezultate na tačnost našeg modela.

Prvo ćemo probati bez menjanja parametara sa funkcijom *KNeighborsClassifier* iz biblioteke *sklearn.neighbors*. I kao što smo očekivali rezultati nisu idealni. Tokom treninga imamo svega *~27%* tačnosti, a tokom testiranja samo *~11%.*

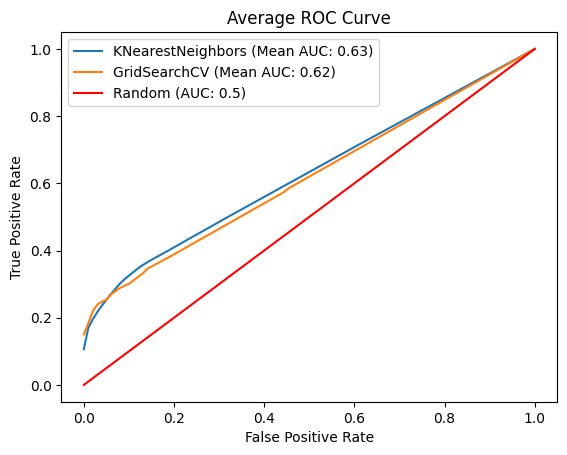
Primenjujemo GridSearch sa parametrima: *{'n\_neighbors': range(3,[], 3), 'weights': ['uniform', 'distance'], 'p': [1, 2]}.* Možemo primetiti znatno poboljšanje u rezultatima, sa *100%* tokom treninga i *~20%* tokom testiranja. Parametri za najbolje rezultate: *{'n\_neighbors': 3, 'p': 2, 'weights': 'distance'}.*



Slika 12 Rezultati KNN bez promene argumenata



Slika 13 Rezultati za KNN sa najboljim izabranim parametrima



Slika 14 ROC kriva za KNN sa i bez izmene argumenata

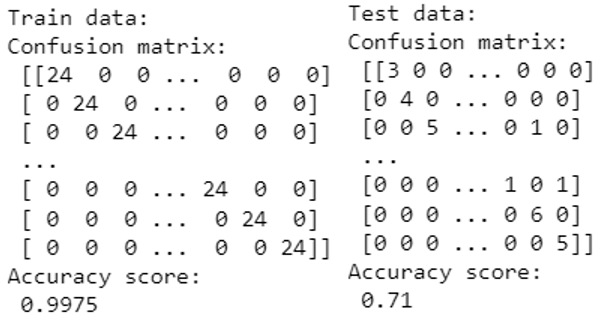
## Naivni Bajes

Naivni Bajes je algoritam koji se zasniva na Bajesovoj teoremi sa pretpostavkom nezavisnosti atributa, zbog čega se i naziva „naivni“. Algoritam radi na principu izračunavanja verovatnoća pripadnosti određenoj klasi za dati ulazni vektor atributa. Prvo se izračunava apriorna verovatnoća svake klase na osnovu raspodele klasa u trening skupu podataka. Zatim se za svaku klasu izračunava verovatnoća da ulazni vektor atributa pripada toj klasi. Za numeričke atribute, verovatnoća se računa na osnovu raspodele atributa u klasi, obično koristeći normalnu raspodelu. Nas zanimaju isključivo numerički atributi jer su nam takvi svi podaci u skupu sem klase koje treba da dobijemo. Za klasu biramo onu koja ima najveću verovatnoću da je tačna.

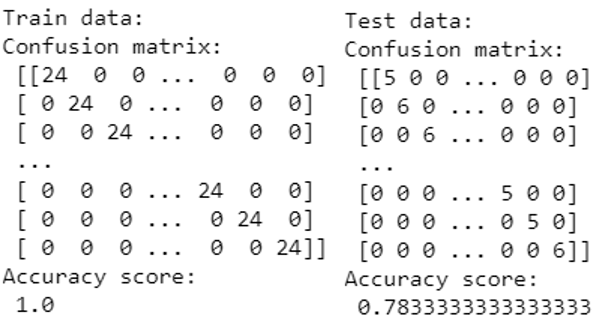
Multinomijalni Naivni Bajesov algoritam je specifična varijanta Naivnog Bajesovog algoritma koja je prilagođena za rad sa diskretnim atributima koji predstavljaju brojanje ili frekvenciju pojavljivanja određenih događaja, i mi ćemo njega koristiti.

Koristimo funkciju *MultinomialNB* iz biblioteke *sklearn.naive\_bayes.* Ako ga upotrebimo bez menjanja argumenata dobijama tačnost od *~100%* pri treningu i *~71%* pri testiranju.

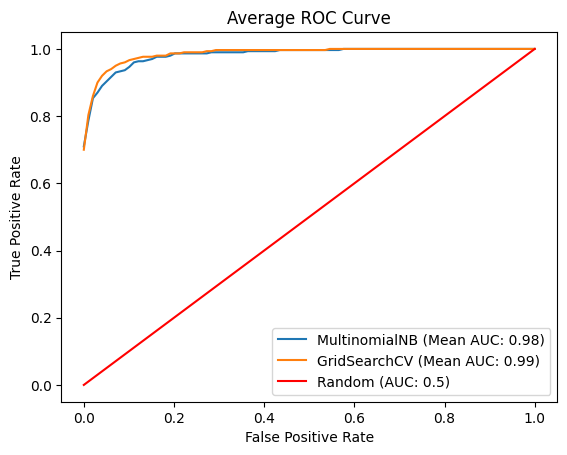
Kada primenimo GridSearch sa parametrima: *{'alpha': [0.1, 0.2, 0.4, 0.5, 0.8, 1.0], 'fit\_prior': [True, False]}.* Dobijamo trening tačnost od *100%* i tačnost pri testiranju od *~78%* za parametre *{'alpha': 0.1, 'fit\_prior': True}.*



Slika 15 Rezultati Multinomialnog Bajesovog Naivnog algoritma



Slika 16 Rezultati Multinomijalnog Naivnog Bajesovog algoritma sa najboljim odabranim parametrima

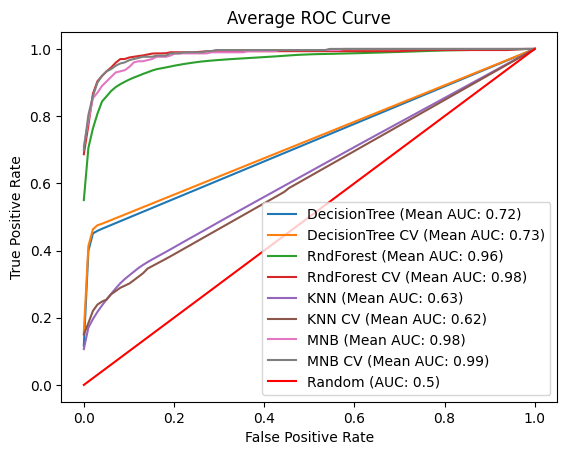


Slika 17 ROC kriva za Multinomijalni Naivni Bajes

## Poređenje modela

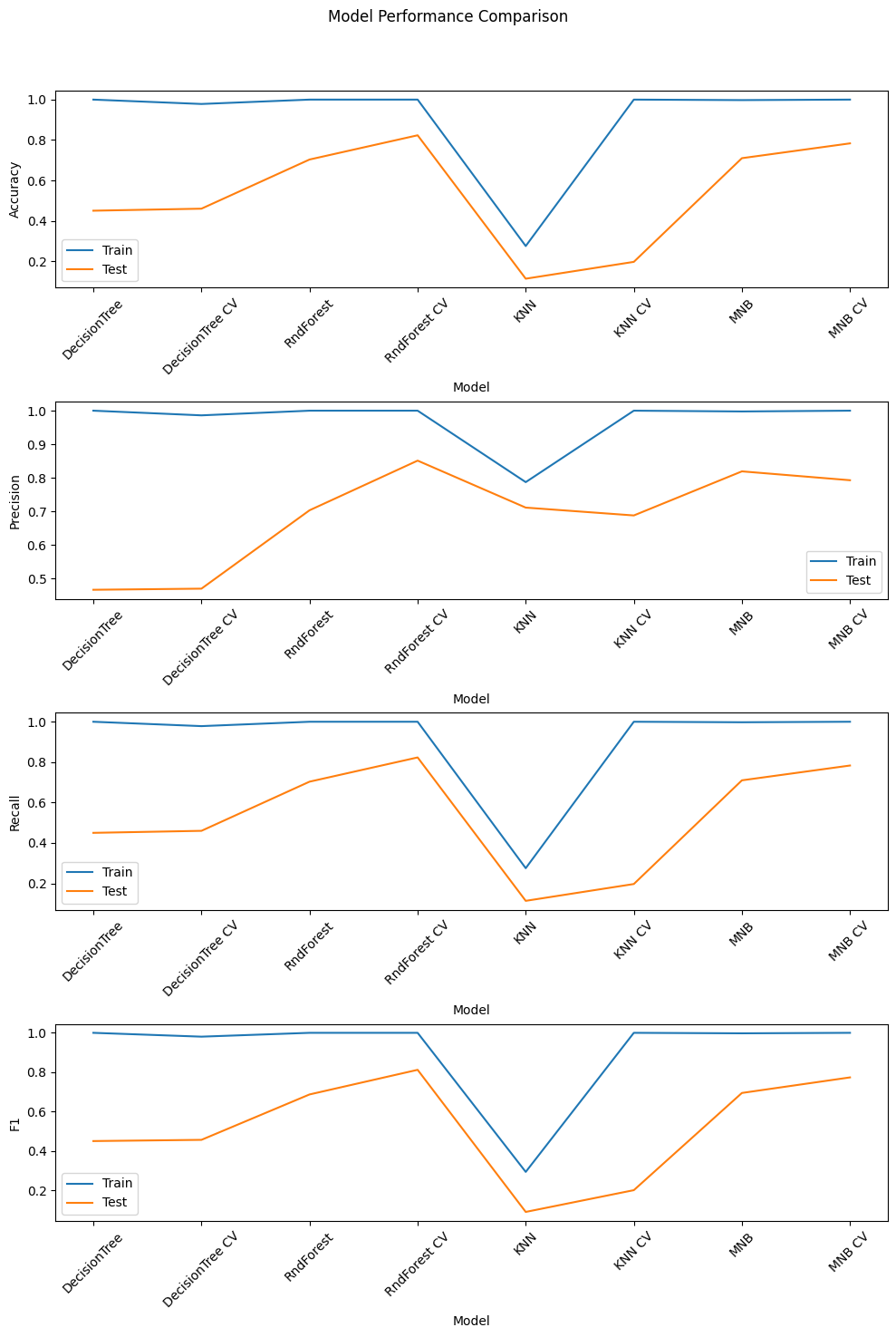
Sada kada imamo sve modele izgrađene možemo ih i uporediti. Za to koristimo ROC krivu (Receiver Operating Characteristic curve) čime prikazujemo odnos između stope lažno pozitivnih (FPR) i tope tačno pozitivnih (TPR). Idealna ROC kriva ide ka gornjem levom uglu što znači da je TPR visok, a FPR nizak. Površina ispod krive (AUC – Area Under the Curve) koristi se kao mera performansi modela. Što je veći AUC to je bolji model. Za AUC imamo model koji nasumično pogađa klasu .

Možemo da vidimo da je Multinomijalni Naivni Bajes najbolji model za ovaj skup podataka, ali je blisko praćen modelom Slučajne šume, čija varijacija sa odabranim parametrima čak pokazuje i bolje rezultate nego Multinomijalni Naivni Bajes, dok su KNN i Stablo odlučivanja dosta lošiji ali idalje bolji od nasumičnog pogađanja. Za razliku od raultata najbolje 2 metode koji su približni, ovde još jednom možemo da vidimo jasnu razliku između KNN i Stabla odlučivanja, sa znatno lošijim performansama za KNN.



Slika 18 ROC kriva i poređenje svih modela

Kako bi dobili jasniju sliku izdvajamo i na grafovima poredimo metrike svih korišćenih modela. Metrike koje posmatramo su: tačnost (Accuracy), preciznost (Precision), osetljivost (Recall, Sensitivity, True Positive Rate) i F1 skor(F1 score). Iako nijedan model nije dao preterano dobre rezultate (uzrok čega je verovatno relativno mali skup podataka) idalje možemo da istaknemo metode Multinomijalnog Naivnog Bajesa i Slučajne šume. Možemo takođe primetiti da se ističu i KNN i Stablo odlučivanja sa svojim znatno lošijim rezultatima, gde je KNN sem u preciznosti ubedljivo najbolji model za ovaj skup podataka.



Slika 19 Grafovi koji porede individualne metrike modela

# Klasterovanje

Cilj klasterovanja podataka je identifikacija inherentnih obrazaca, struktura ili grupa unutar skupa podataka. Klasterovanje je tehnika nenadgledanog mašinskog učenja koja ima za cilj grupisanje sličnih podataka na osnovu njihovih intrinzičkih karakteristika ili blizine u prostoru karakteristika.

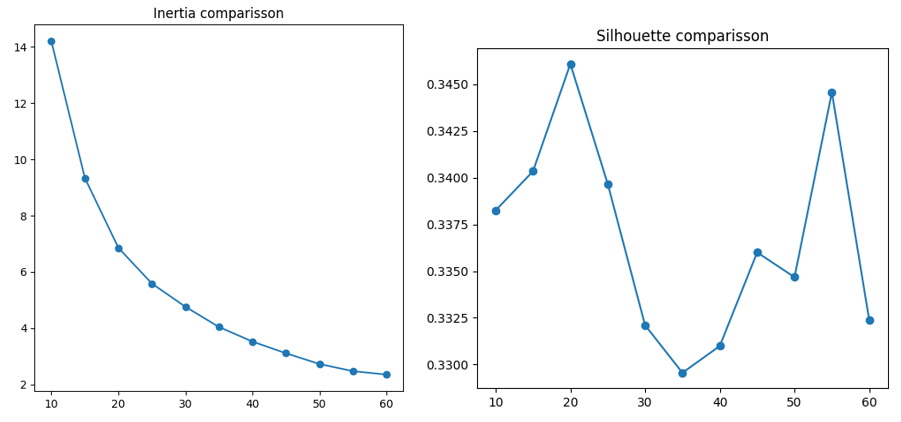
U ovom segmentu ćemo pokazati primenu 2 različita modela klasterovanja:

1. KSredina
2. DBSCAN

## KSredina

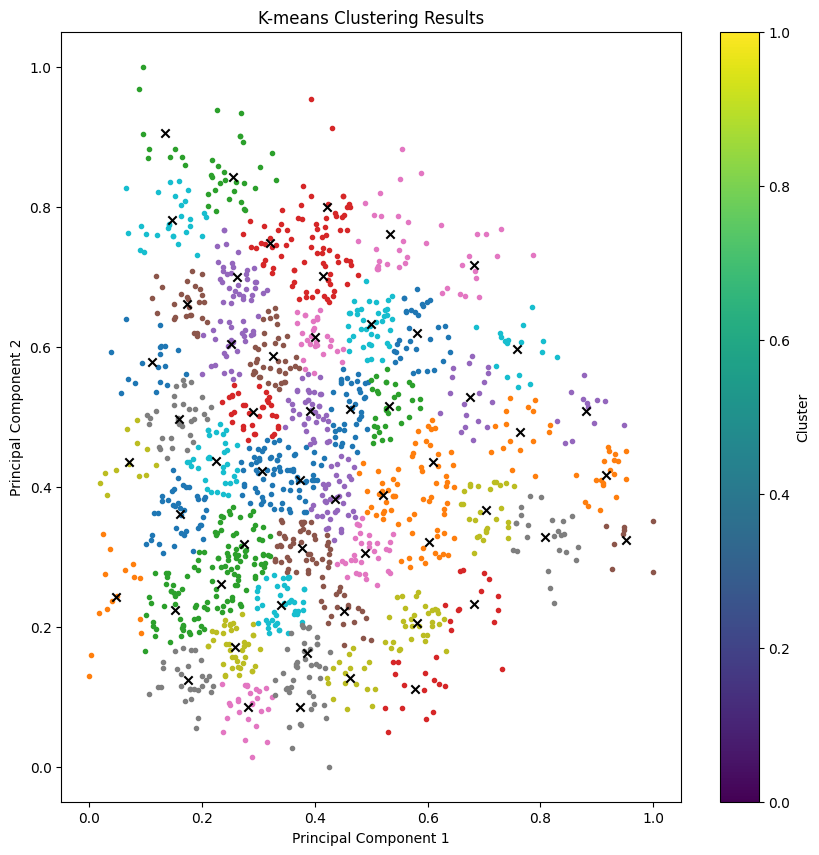
Algoritam K-Sredina (K-Means) je metoda klasteriranja podataka koja grupiše podatke u predefinisan broj K klastera. Radi tako što iterativno ažurira pozicije centroida u svakom klasteru, minimizujući sumu kvadratnih udaljenosti između tačaka i centroida. Svaki podatak je dodeljen klasteru čijem centroidu je najbliži. Algoritam se ponavlja dok se centroidi stabilizuju ili dostigne predefinisan broj iteracija. Krajnji rezultat je grupisanje podataka u klasterima na osnovu sličnosti.

Za izbor najboljih parametara za K-Sredina koristiti dve mere, siluetu (Silhouette) i inerciju (Inertia), čiji odnos vrednosti možemo videte na slikama ispod.



Slika 20 Silueta i Inercija – Kmeans

Sa grafika mozemo da vidimo da Silueta ima 2 maksimuma za broj klastera 20 i 55, ali kada se konsultujemo sa grafikom inercije vidimo da ona naglo opada i kod 55 skoro pa da se više i ne menja. Naš model ima 50 klasa pa ovaj rezultat i nije toliko loš s obzirom da umesto 1500 koristimo samo 2 parametra.

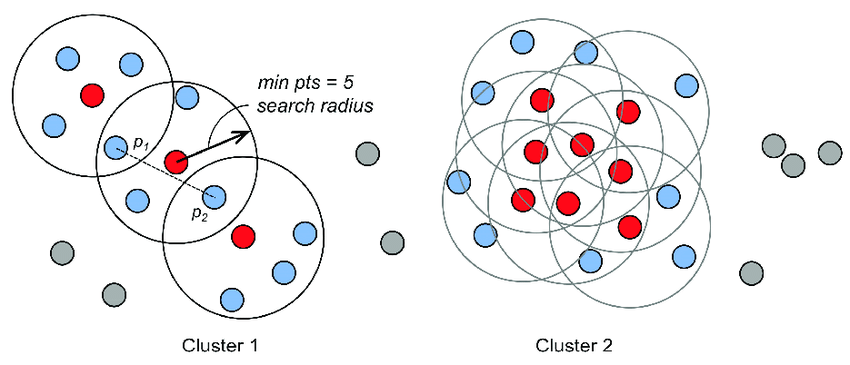


Slika 21 Prikaz klasterovanja

Ovde možemo da vidimo da zaista ima smisla kako je izvršeno klasterovanje pa čak bi bilo teško smisleno ukloniti 5 od određenih klastera.

## DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) je algoritam nenadgledanog učenja koji se koristi za identifikaciju klastera u skupovima podataka. Zasniva se na gustini tačaka u prostoru.

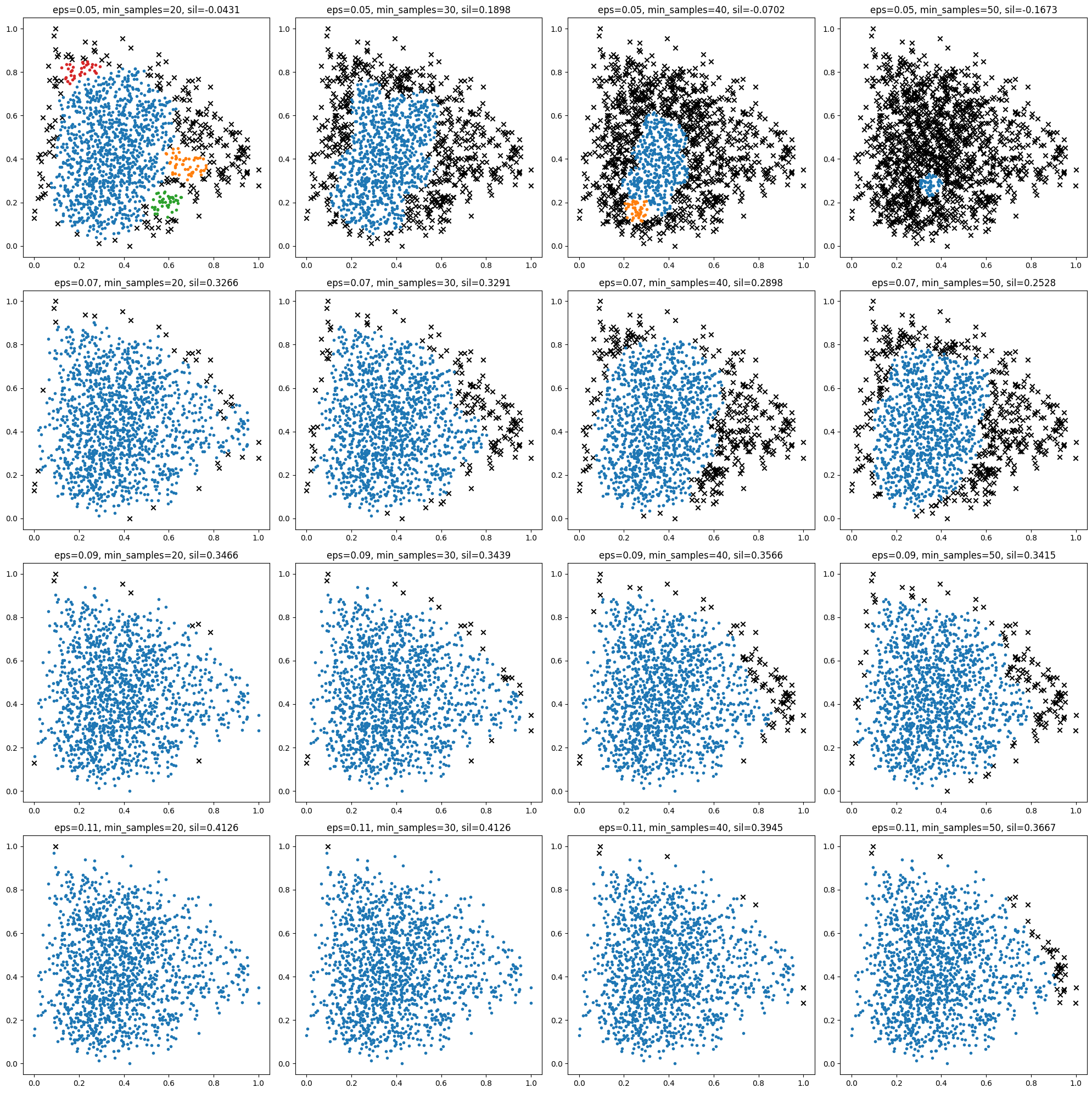


Slika 22 DBSCAN[[2]](#footnote-2)

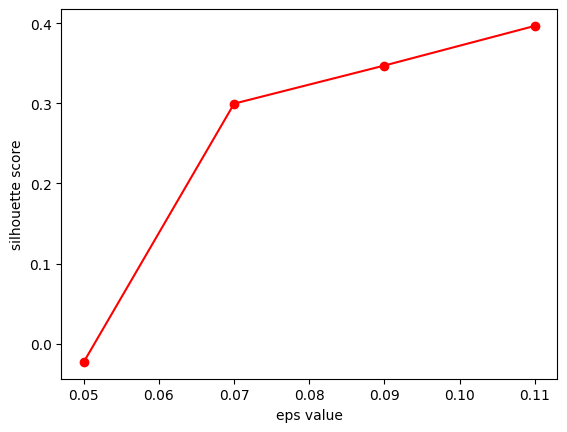
Na slici možemo videti da postoje 3 vrste tačaka, crvene su ključne (tačke koje zadovoljavaju uslove klastera), plave tačke koje su krajične (tačke koje ne zadovoljavaju uslov ali se nalaze na dohvat ključnih tačaka) i sive tačke koje su šum (nisu nijedne od prethodna dva tipa). Ovaj vid klasifikacije tačaka je razlog zašto je ovaj algoritam ume da se snađe i kada postoji šum u podacima. Ali ovaj algoritam ima manu, a to je da se ne snalazi sa skupovima podataka visokih dimenzija ili šarenolikih gustina.

Epsilon vrednost predstavlja razdaljinu ili radijus koja definiše poluprečnik unutar kojeg se traže susedne tačke. Što je epsilon veći, znači da će veći broj manjih klastera pripastijednom koji će se prikazivati kao jedan veći.

Očekujemo relativno slabe rezultate od ovog algortima sa našim skupom podataka ali daće nam svakako dobar uvid u povezanost glavna 2 parametra našeg skupa. Kako bi odabrali parametre za građenje modela prosledićemo mu više kombinacija parametara i izabraćemo onu koja izgleda najbolje.

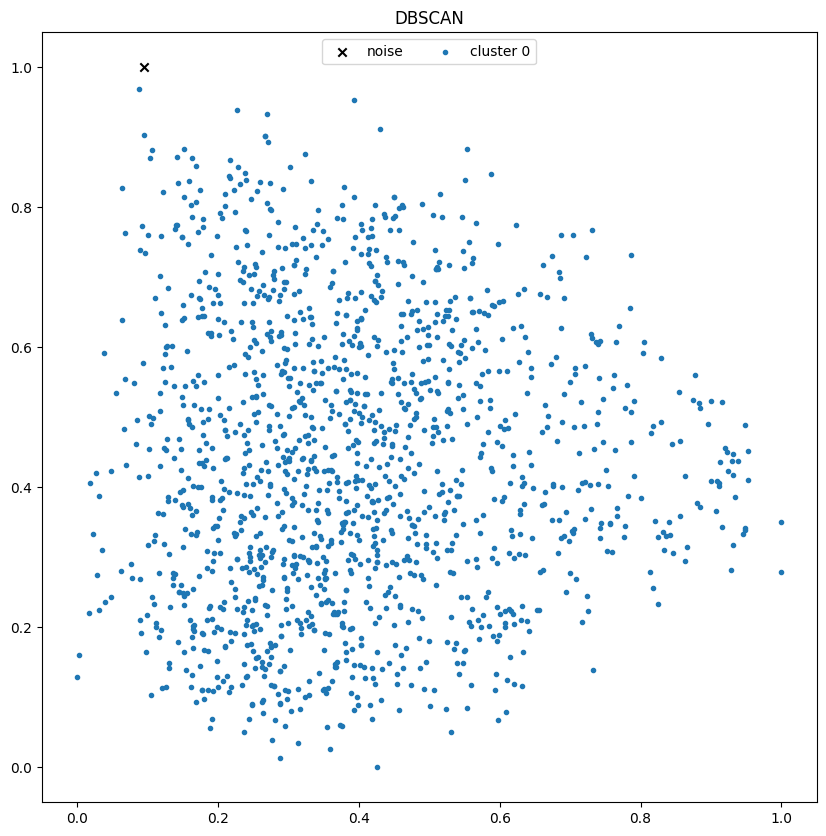


Slika 23 DBSCAN za različite kombinacije parametara



Slika 24 EPS vrednost i njen uticaj na siluetu

Gledajući ove dijagrame vidimo da su našti podatci uglavnom grupisani na sličnoj udaljenosti i u okolini jednih drugih. Najmanji broj izostavljenih čvorova, tj najvišu vrednost za siluetu imamo za parametre eps=0.11 I min\_samples=30.



Slika 25 DBSCAN za esp=0.11 i min\_samples=30

# Pravila asocijacije

Pravila pridruživanja je tehnika istraživanja podataka koja se koristi za otkrivanje zanimljivih veza ili asocijacija među elementima u velikim skupovima podataka. Fokusira se na pronalaženje obrazaca ili pravila koja opisuju zajedničko pojavljivanje ili zavisnost između različitih elemenata u transakcionoj bazi podataka ili skupu transakcija.

# Zaključak

1. “Data Mining Amazon reviews Dataset” – Zhi Liu (2019). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Amazon+Commerce+reviews+set]. National Engineering Research Center for E-Learning, Hubei Wuhan, China. [↑](#footnote-ref-1)
2. https://www.researchgate.net/figure/The-DBSCAN-algorithm-and-two-generated-clusters-There-are-three-types-of-points-as\_fig2\_342082665 [↑](#footnote-ref-2)