Seminarski rad u okviru kursa Istraživanje podataka I

Analiza skupa podataka

Orbit Classification For Prediction / NASA

Maja Milenković
mi19160@alas.matf.bg.ac.rs

Matematički fakultet,
Univerzitet u Beogradu

Sadržaj

1. Uvod

Ovaj projekat predstavlja detaljno istraživanje i implementaciju različitih metoda za klasterovanje i klasifikaciju baze podataka na skup podataka Orbit Classification For Prediction / NASA. Koristimo dataset koji sadrži podatke o orbitama 1748 asteroida, prikupljene u formatu pandas DataFrame, kako bismo dublje razumeli karakteristike ovih nebeskih tela i njihove orbitalne klase.

1.1 Eksplorativna analiza podataka

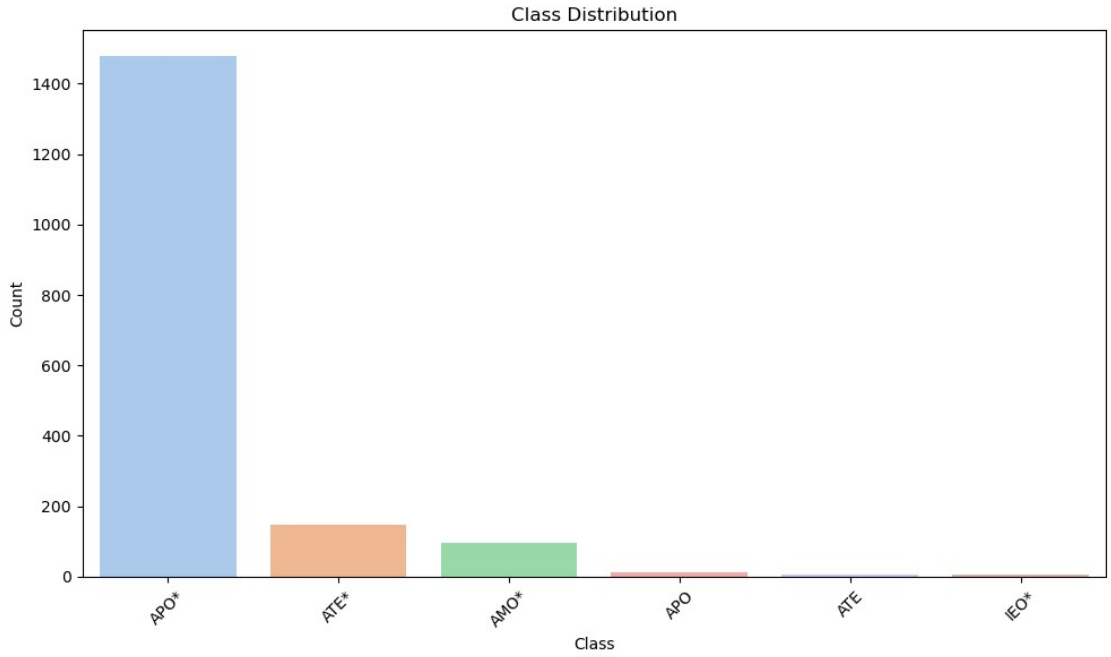
Skup podataka se sastoji od 12 atributa koji pružaju informacije o parametrima asteroidnih orbita:

- a (AU) - Polu-glavna osa orbite u astronomskim jedinicama (AU).
- e - Ekscentricitet orbite.
- i (deg) - Nagib orbite u odnosu na ekliptičku ravni i ekvinocij J2000 (J2000-Ecliptic) u stepenima.
- w (deg) - Argument periheliona (J2000-Ecliptic) u stepenima.
- Node (deg) - Geografska dužina uspona čvora orbite (J2000-Ecliptic) u stepenima.
- M (deg) - Srednja anomalija na epohi u stepenima.
- q (AU) - Perihelion udaljenost orbite u astronomskim jedinicama (AU).
- Q (AU) - Afelion udaljenost orbite u astronomskim jedinicama (AU).
- P (yr) - Orbitalni period u Julian godinama.
- H (mag) - Apsolutna V-magnituda.
- MOID (AU) - Minimalna udaljenost preseka orbite (minimalna udaljenost između oscilirajućih orbita NEO-a i Zemlje).
- class - Klasifikacija objekta

	a (AU)	e	i (deg)	w (deg)	Node (deg)	M (deg)	q (AU)	Q (AU)	P (yr)	H (mag)	MOID (AU)	class
0	1.078066	0.826854	22.825495	31.382966	88.010681	215.528772	0.1867	1.97	1.12	16.90	0.034507	APO*
1	1.245304	0.335342	13.337482	276.893024	337.207958	104.155607	0.8277	1.66	1.39	15.60	0.030669	APO*
2	1.470264	0.559922	6.352995	285.852564	35.736768	174.626213	0.6470	2.29	1.78	16.25	0.025795	APO*
3	1.776025	0.650141	39.832538	267.791993	356.903343	173.188556	0.6214	2.93	2.37	15.20	0.003551	APO*
4	1.874123	0.764602	1.326399	43.388048	349.694944	235.158622	0.4412	3.31	2.57	18.80	0.011645	APO*

1.1.1 Balansiranost klasa

Jedan od koraka u našem istraživanju je bio detaljan pregled balansa između o klasa u našem dataset-u. Provera ravnoteže između klasa je značajna jer velike razlike u broju instanci između klasa mogu imati negativan uticaj na performanse algoritama.



Udeo svake klase u odnosu na celokupan skup podataka jeL

APO*: 92.12%

ATE*: 9.29%

AMO*: 5.99%

APO: 0.87%

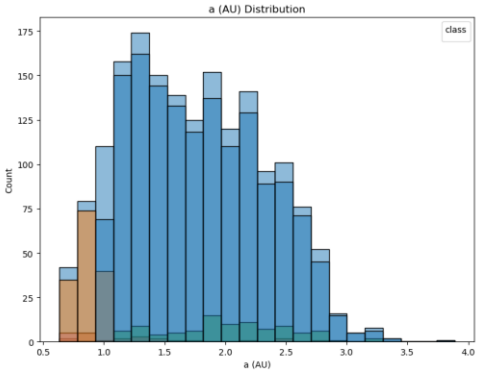
ATE: 0.44%

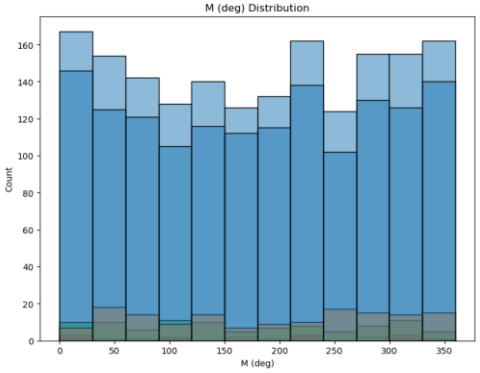
IEO*: 0.31%

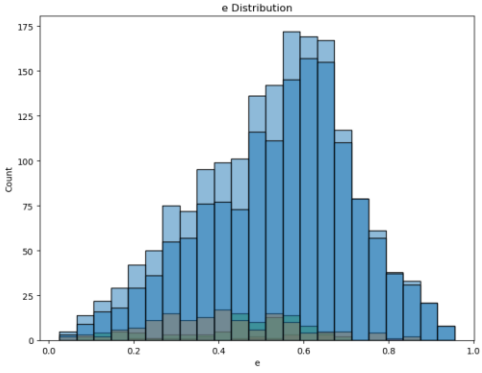
Nakon analize, primetili smo da postoji nebalansiranost između klasa u našem dataset-u. Kako bismo se nosili sa nebalansiranošću, planiramo da primenimo odgovarajuće tehnike za rad sa nebalansiranim klasama po potrebi.

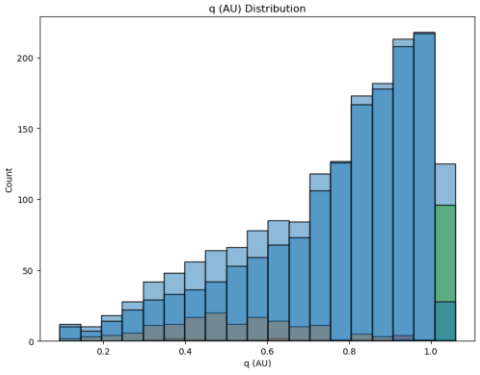
1.1.2 Histogrami atributa

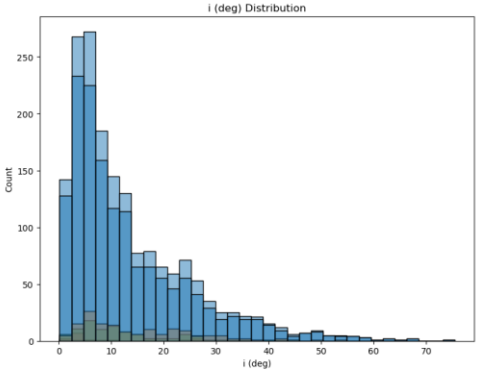
Na narednim slikama prikazani su histogrami svakog atributa u odnosu na klase. Oblik histograma nam daje informacije o raspodeli vrednosti atributa.

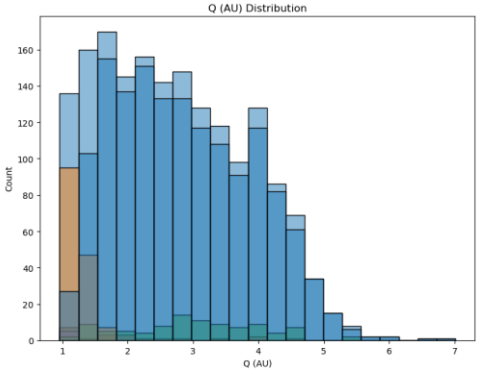

a (AU) Distribution
class
Count
a (AU)

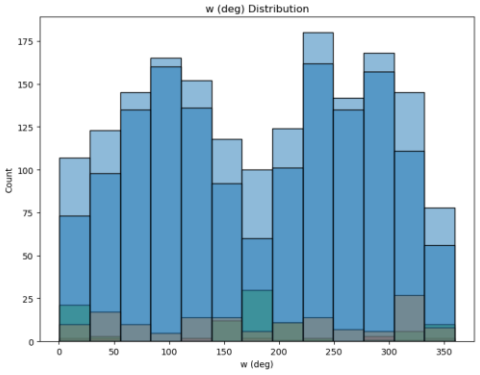

M (deg) Distribution
Count
M (deg)

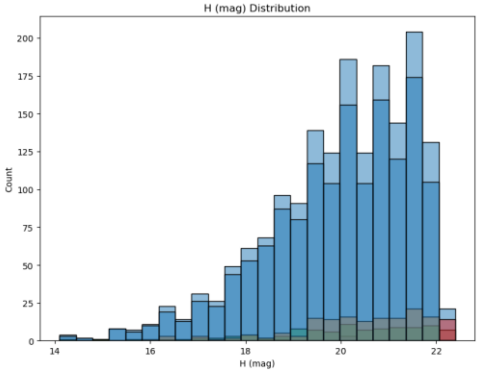

e Distribution
Count
e

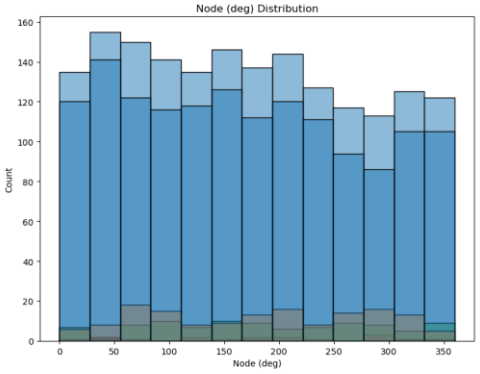

q (AU) Distribution
Count
q (AU)

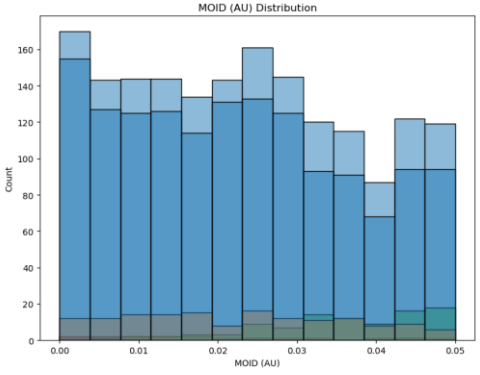

i (deg) Distribution
Count
i (deg)


Q (AU) Distribution
Count
Q (AU)


w (deg) Distribution
Count
w (deg)

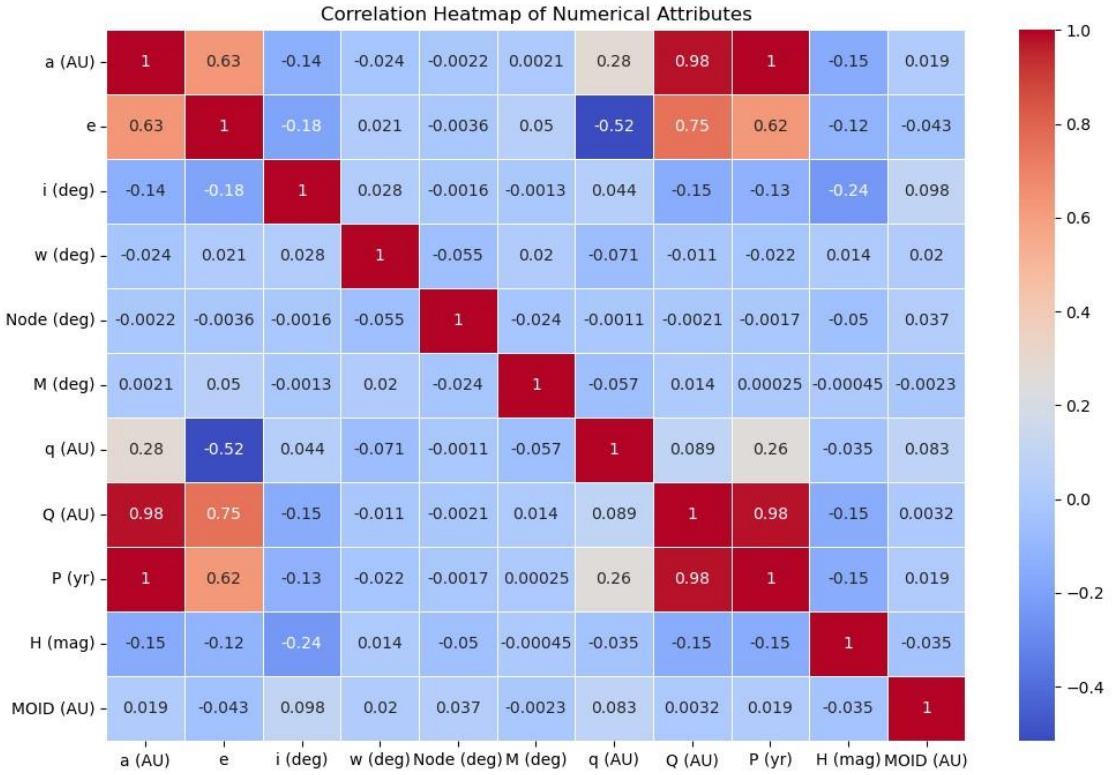

H (mag) Distribution
Count
H (mag)


Node (deg) Distribution
Count
Node (deg)


MOID (AU) Distribution
Count
MOID (AU)

1.1.3 Matrica korelacije

Matrica korelacije koristi skalu boja da prikaže snagu i smer korelacija. Tamnije boje, poput duboko plave ili crvene, ukazuju na jače korelacije, dok svetlije boje prikazuju slabije ili nepostojeće korelacije.



Uočavamo da postoji jaka korelacija između određenih atributa, što ukazuje na to da se jedan atribut može eliminisati iz analize, jer ne doprinosi dodatnim informacijama koje već nisu obuhvaćene drugim atributom.

1.2 Pretprocesiranje

1.2.1 Nedostajuće vrednosti

Kao prvi korak u pripremi podataka, važno je rukovati sa nedostajućim vrednostima. Srećom, kao što možemo videti na slici, u našem datasetu nema nedostajućih vrednosti.

```
data.isna().sum()
```

```
a (AU)         0
e              0
i (deg)        0
w (deg)        0
Node (deg)     0
M (deg)        0
q (AU)         0
Q (AU)         0
P (yr)         0
H (mag)        0
MOID (AU)      0
class          0
dtype: int64
```

1.2.2 Rukovanje autlajerima i skaliranje

Sledeći korak je rukovanje elementima van granica(outliers). Za identifikaciju autlajera koristili smo IQR (Interquartile Range) metod:

1.Prvo se računaju kvartili:

Prvi kvartil (Q1) predstavlja vrednost ispod koje se nalazi 25% najmanjih vrednosti u datasetu.

Treći kvartil (Q3) predstavlja vrednost ispod koje se nalazi 75% najmanjih vrednosti u datasetu.

2.Zatim se računa IQR (Interquartile Range), što je razlika između trećeg i prvog kvartila:

IQR = Q3 - Q1

3.Autlajeri se identifikuju kao vrednosti koje padaju izvan opsega:

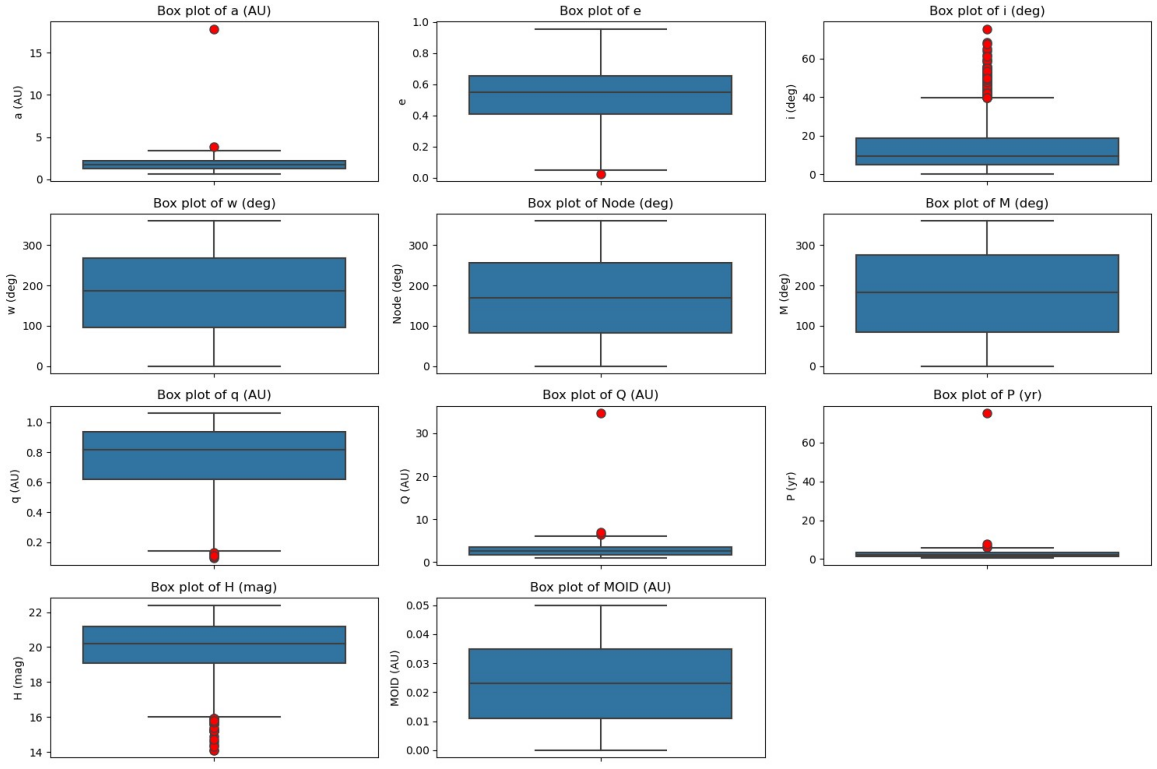
Donja granica: Q1 - 1.5 * IQR

Gornja granica: Q3 + 1.5 * IQR

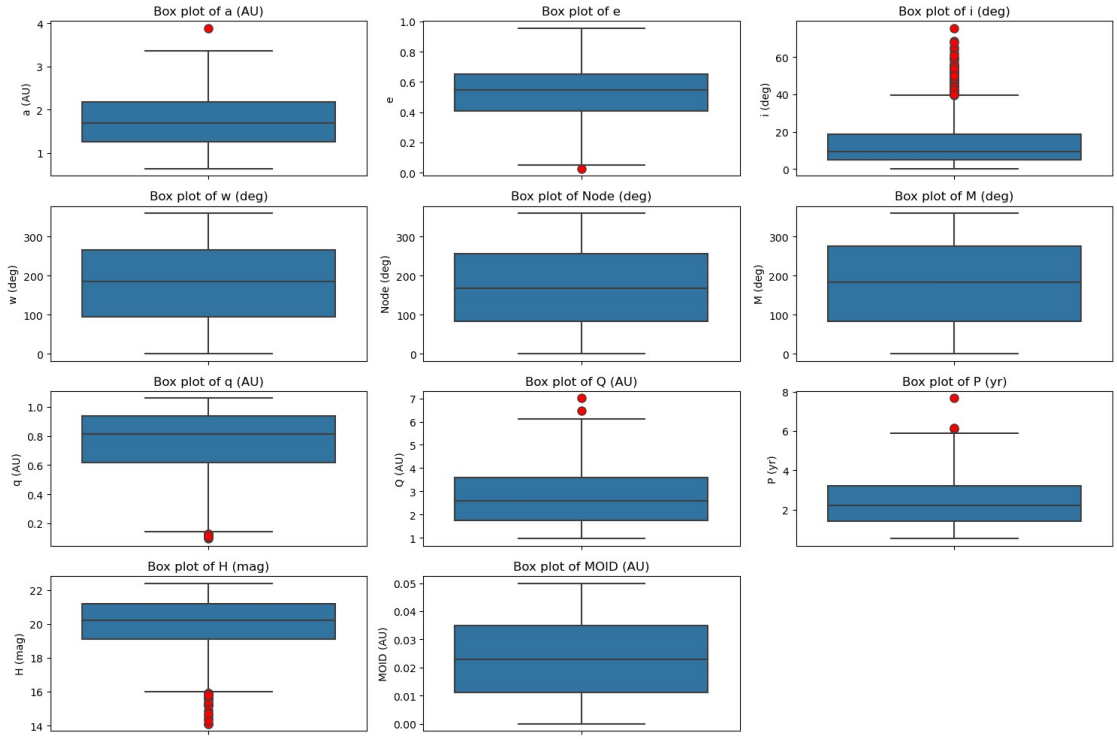
Vrednosti koje su manje od donje granice ili veće od gornje granice smatraju se autlajerima, a rezultati pokazuju da nema mnogo autlajera u našem datasetu.

	lower	min	num_lower	upper	max	num_upper	percantage
a (AU)	-0.104511	0.635223	0	3.546646	3.888719	1	0
e	0.041192	0.025425	1	1.020715	0.956042	0	0
i (deg)	-15.987849	0.146084	0	39.647466	75.412403	65	4
w (deg)	-160.797589	0.521838	0	523.014817	359.662669	0	0
Node (deg)	-178.426441	0.136042	0	517.340838	359.854602	0	0
M (deg)	-204.964619	0.052165	0	564.446926	359.825201	0	0
q (AU)	0.138375	0.092800	10	1.414575	1.060100	0	1
Q (AU)	-0.957500	0.960000	0	6.302500	7.010000	2	0
H (mag)	15.950000	14.100000	25	24.350000	22.400000	0	1
MOID (AU)	-0.024570	0.000010	0	0.070444	0.049987	0	0

Kako bismo bolje vizualizovali autlajere i uočili da li ima nekih koje se ističu, kreirali smo boxplotove za svaki atribut. Boxplotovi nam pružaju jasnu sliku o distribuciji vrednosti atributa i pomažu nam da identifikujemo potencijalne autlajere koji se izdvajaju kao ekstremne vrednosti u odnosu na ostale.



Mozemo primetiti da postoji jedan autlajer koji se značajno razlikuje od ostalih autlajera u datasetu. Uklonili smo taj ekstremni autlajer kako bismo on ne bi kompromitovao tačnost naših modela.



2. Klasifikacija

Klasifikacija predstavlja proces otkrivanja i modeliranja obrazaca u podacima kako bi se dodeljivali objekti ili instance određenim klasama ili kategorijama. Kao tehnike za klasifikaciju koristimo Stablo odlučivanja i K-najbližih suseda.

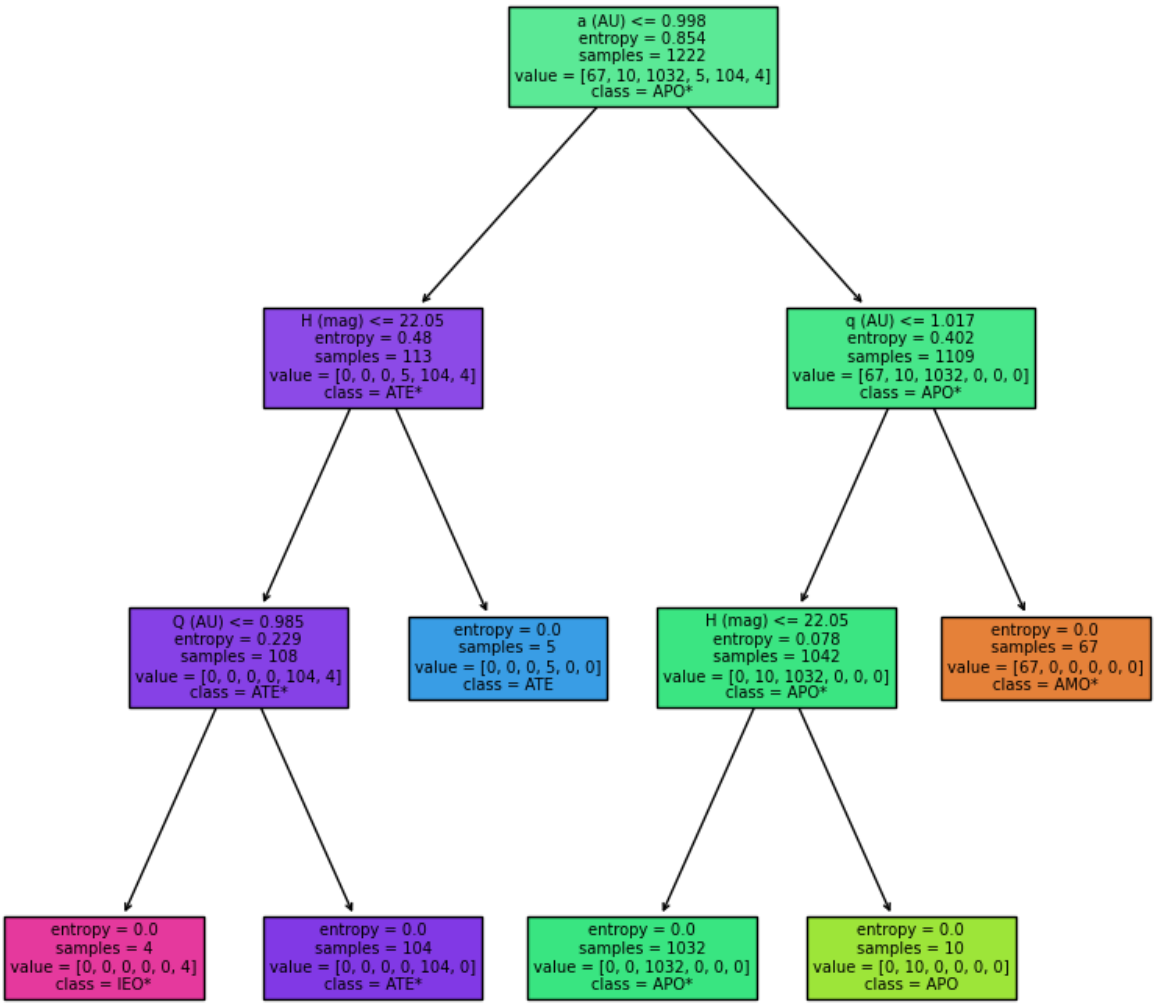
2.1 Stabla odlučivanja

Algoritam Stabla odlučivanja počinje sa celim dataset-om i bira atribut koji najbolje deli podatke na osnovu određenog kriterijuma za homogenost, kao što je Gini indeks ili entropija. Tada se dataset deli na manje grupe, odnosno grane stabla, na osnovu vrednosti tog atributa i postupak se rekurzivno ponavlja dok se ne ispune kriterijumi za zaustavljanje. Nije potrebna normalizacija i nije osetljiv na autlajere.

Nad modelom se može pozvati funkcija “feature_importances_” kako bismo dobili informacije o značaju atributa pri klasifikaciji. Ova funkcija nam omogućava da saznamo koji atributi doprinose najviše u donošenju odluka modela, što je korisno za razumevanje važnosti svakog atributa u procesu klasifikacije.

```
model.feature_importances_
```

```
array([0.52212145, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.36915616, 0.02280428, 0.08591811, 0.        ])
```



2.1.1 Odabir hiper-parametara

Za treniranje modela primenili smo GridSearch koji procesom automatskog odabira nalazi najbolje hiperparametre za model. Prvo smo definisali koje hiperparametre želimo optimizovati, kao što su dubina stabla ili kriterijum podela i zatim smo odredili raspon vrednosti za svaki hiperparametar. GridSearch je potom kros validacijom ispitao sve kombinacije ovih vrednosti kako bi pronašao najbolju kombinaciju hiperparametara koja daje optimalne performanse za naš model.

```
params = {'criterion': ['gini', 'entropy'],
          'max_depth': range(2,10),
          'min_samples_leaf' : [2,3,4,5],
         }
```

- **"criterion"**: određuje kriterijum koji se koristi za merenje kvaliteta podela u svakom čvoru stabla. Dva najčešće korišćena kriterijuma su "gini" i "entropy"
- **“max_depth”**: kontroliše maksimalnu dubinu stabla. Dubina stabla odnosi se na broj čvorova od korena do lista i time se može sprečiti preprilagođavanje modela na trening podacima
- **“min_samples_leaf”**: određuje minimalni broj instanci koje moraju biti u svakom listu stable i može pomoći u sprečavanju preprilagođavanja

```
estimator = GridSearchCV(DecisionTreeClassifier(), param_grid = params, scoring = 'accuracy', cv = 4, verbose = 4)
```

```
estimator.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 64 candidates, totalling 256 fits
```

Nakon postavljanja parametara za ispitivanje, zovemo GridSearch() kojoj prosleđujemo metodu klasifikacije, listu kombinacija parametara koje želimo ispitati, “scoring” - meru performansi koju želimo optimizovati tokom GridSearch procesa, “cv” - deli dostupne podatke na trening i validacione skupove više puta(broj koji je prosleđen kao njegov argument) i evaluira model na različitim kombinacijama ovih skupova., “verbose”- kontroliše nivo ispisivanja informacija tokom izvršavanja.

Sa parametrima koje smo mi izabrali, model trenira koristeći unakrsnu validaciju sa 4 preklopa za svaku od 64 različite kombinacije hiperparametara, što ukupno rezultuje u 256 testiranja (fitovanja) modela.

```
estimator.best_estimator_
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=2)
```

Kao povratnu vrednost funkcije GridSearch() dobijamo estimator nad kojim možemo koristiti funkciju **best_estimator_** kako bismo saznali koja je najbolja konfiguracija hiperparametara radi dobijanja modela sa najvećom tačnošću.

Kao alat za analizu performansi modela za klasifikaciju koristimo matricu konfuzije. Ova matrica prikazuje broj tačno klasifikovanih instanci za svaku klasu, kao i broj pogrešno klasifikovanih instanci. Matrica konfuzije omogućava analizu tačnosti i grešaka modela u klasifikaciji, što je korisno za evaluaciju njegove efikasnosti.

```
confusion_matrix(y_train, y_train_pred)
```

```
array([[  67,    0,    0,    0,    0,    0],
       [   0,   10,    0,    0,    0,    0],
       [   0,    0, 1032,    0,    0,    0],
       [   0,    0,    0,    5,    0,    0],
       [   0,    0,    0,    0,  104,    0],
       [   0,    0,    0,    0,    0,    4]])
```

2.1.2 RandomTree

Kao napredniju tehniku koristimo RandomForest za poboljšanje performansi modela za klasifikaciju. Ova tehnika radi tako što kombinuje više stabala odlučivanja (Decision Trees) u ansambl, pri čemu svako stablo daje svoju predikciju. Random Forest koristi tehniku "bagging" (Bootstrap Aggregating) kako bi kreirao različite podskupove trening podataka za svako stablo i time smanjio varijansu modela. Osim toga, Random Forest uvodi slučajnost u izbor atributa pri izgradnji svakog stabla, što dodatno doprinosi raznolikosti ansambla i smanjuje rizik od preprilagođavanja.

```
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(X_train, y_train)
```

```
random_forest_report(model, X_test, y_test)
```

```
Confusion Matrix:
[[ 29   0   0   0   0   0]
 [  0   4   0   0   0   0]
 [  1   0 443   0   0   0]
 [  0   0   0   2   0   0]
 [  0   0   1   0  44   0]
 [  0   0   0   0   0   1]]

Classification Report:
              precision    recall  f1-score   support

        AMO*       0.97      1.00      0.98        29
         APO       1.00      1.00      1.00         4
        APO*       1.00      1.00      1.00       444
         ATE       1.00      1.00      1.00         2
        ATE*       1.00      0.98      0.99        45
        IEO*       1.00      1.00      1.00         1

    accuracy                           1.00       525
   macro avg       0.99      1.00      0.99       525
weighted avg       1.00      1.00      1.00       525
```

2.2 K najbližih suseda

Algoritam K najbližih suseda (K-Nearest Neighbors ili skraćeno KNN) je jednostavan i efikasan algoritam za klasifikaciju i može se koristiti za različite tipove podataka. Algoritam radi na osnovu principa da su slični podaci često grupisani zajedno. Osnovni koraci ovog algoritma za klasifikaciju su:

- Izbor vrednosti K: Prvi korak u primeni KNN algoritma je izbor broja K, koji predstavlja broj najbližih suseda koji će se uzeti u obzir pri donošenju odluke.
- Računanje rastojanja: Za svaku instancu koju želimo klasifikovati, računaju se rastojanja između te instance i svih instanci u trening skupu podataka. Najčešće se koristi euklidsko rastojanje, ali mogu se koristiti i druge metrike rastojanja, u zavisnosti od problema.
- Sortiranje i izbor najbližih suseda: Nakon što se izračunaju rastojanja, instanci se dodeljuju klase na osnovu K najbližih suseda iz trening skupa.
- Donošenje odluke: Klasifikacija nove instance se vrši tako što se broji koliko puta se svaka klasa pojavljuje među K najbližih suseda. Klasa koja se najčešće pojavljuje postaje predviđena klasa za novu instancu.

Autlajeri, ili ekstremno odstupajuće vrednosti mogu značajno uticati na KNN jer se oslanja na rastojanja između instanci. Takođe, atributi sa većim numeričkim opsezima mogu dominirati nad atributima sa manjim opsezima i imati veći uticaj na rezultat. Da bismo efikasno nosili sa ovim izazovima, prvo skaliramo podatke pre primene KNN algoritma. Skaliranje podataka omogućava ravnotežu između atributa sa različitim opsezima.

Najpre ćemo primeniti K najbližih suseda (KNN) na nebalansiranim podacima kako bismo dobili osnovne rezultate. Nakon toga, koristićemo tehniku kombinovanog oversampling-a i undersampling-a, poznatu kao SMOTE-ENN (Synthetic Minority Over-sampling Technique with Edited Nearest Neighbors), kako bismo izbalansirali podatke i zatim ponovo primenili KNN algoritam.

Rezultati bez GridSearcha:

Na trening podacima:

```
Confusion matrix:
[[  31    0   36    0    0    0]
 [   0    0   10    0    0    0]
 [   3    0 1024    0    5    0]
 [   0    0    3    0    2    0]
 [   0    0   35    0   69    0]
 [   0    0    2    0    2    0]]
Accuracy: 0.9198036006546645
Precision: 0.9198036006546645
Recall: 0.9198036006546645
F1 score: 0.9198036006546645
```

Na test podacima:

```
Confusion matrix:
[[  2   0  27   0   0   0]
 [  0   0   4   0   0   0]
 [  4   0 436   0   4   0]
 [  0   0   1   0   1   0]
 [  0   0  28   0  17   0]
 [  0   0   0   0   1   0]]
Accuracy: 0.8666666666666667
Precision: 0.8666666666666667
Recall: 0.8666666666666667
F1 score: 0.8666666666666667
```

```
params = {
    'n_neighbors' : range(2,20),
    'weights' : ['uniform', 'distance'],
    'p' : [1,2]
}
```

```
estimator = GridSearchCV(KNeighborsClassifier(), params, cv = 4, verbose = 4)
```

- **“n_neighbors”**:određuje broj najbližih suseda koji će se uzeti u obzir pri donošenju odluke o klasi novih instanci.
- **“weights”**: Parametar koji kontroliše težine suseda. Može biti postavljen na "uniform" (svi susedi imaju istu težinu) ili "distance" (susedi imaju težine obrnuto proporcionalne rastojanju od nove instance).
- **“p”**: Parametar koji određuje koji tip rastojanja se koristi. Ako je postavljen na 1, koristi se Manhattan rastojanje (L1 norma), dok se za vrednost 2 koristi Euklidsko rastojanje (L2 norma).

```
estimator.best_estimator_
```

```
KNeighborsClassifier(n_neighbors=6, p=1, weights='distance')
```

Rezultati sa najboljim procenjenim hiperparametrima:

Na trening skupu:

```
Confusion matrix:
[[  67    0    0    0    0    0]
 [   0   10    0    0    0    0]
 [   0    0 1032    0    0    0]
 [   0    0    0    5    0    0]
 [   0    0    0    0  104    0]
 [   0    0    0    0    0    4]]
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 score: 1.0
```

Na test skupu:

```
Confusion matrix:
[[  7   0  22   0   0   0]
 [  0   0   4   0   0   0]
 [  4   0 438   0   2   0]
 [  0   0   1   0   1   0]
 [  0   0  18   0  27   0]
 [  0   0   0   0   1   0]]
Accuracy: 0.8990476190476191
Precision: 0.8990476190476191
Recall: 0.8990476190476191
F1 score: 0.8990476190476191
```

2.2.1 BaggingClassifier

Kao ansambl tehniku za unapređenje KNN možemo iskoristiti BaggingClassifier. Ovaj ansambl algoritam funkcioniše tako što generiše više podskupova (bootstrap uzoraka) od trening podataka, a zatim trenira više kopija osnovnog algoritma (u ovom slučaju, KNN) na svakom od ovih podskupova. Nakon toga, rezultati svih kopija se kombinuju kako bi se donela konačna klasifikaciona odluka.

```
Confusion matrix:
[[  22    0   45    0    0    0]
 [   0    0    9    0    1    0]
 [   0    0 1026    0    6    0]
 [   0    0    2    0    3    0]
 [   0    0   32    0   72    0]
 [   0    0    2    0    2    0]]
Accuracy: 0.9165302782324058
Precision: 0.9165302782324058
Recall: 0.9165302782324058
F1 score: 0.9165302782324058
--------------------------------
Confusion matrix:
[[  3   0  26   0   0   0]
 [  0   0   4   0   0   0]
 [  3   0 437   0   4   0]
 [  0   0   1   0   1   0]
 [  0   0  29   0  16   0]
 [  0   0   0   0   1   0]]
Accuracy: 0.8685714285714285
Precision: 0.8685714285714285
Recall: 0.8685714285714285
F1 score: 0.8685714285714285
```

Rezultati primene algoritma nad izbalansiranim podacima nakon primene SMOTEENN(iskorišćen je GridSearch zarad najboljih rezultata).

Nad trening skupom:

```
Confusion matrix:
[[1029    0    0    0    0    0]
 [   0 1032    0    0    0    0]
 [   0    0  802    0    0    0]
 [   0    0    0 1032    0    0]
 [   0    0    0    0 1032    0]
 [   0    0    0    0    0 1032]]
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 score: 1.0
```

Nad test skupom:

```
Confusion matrix:
[[ 12   0  16   0   1   0]
 [  0   0   4   0   0   0]
 [ 31   7 390   1  15   0]
 [  0   0   1   0   1   0]
 [  0   0  13   3  26   3]
 [  0   0   0   0   0   1]]
Accuracy: 0.8171428571428572
Precision: 0.8171428571428572
Recall: 0.8171428571428572
F1 score: 0.8171428571428572
```

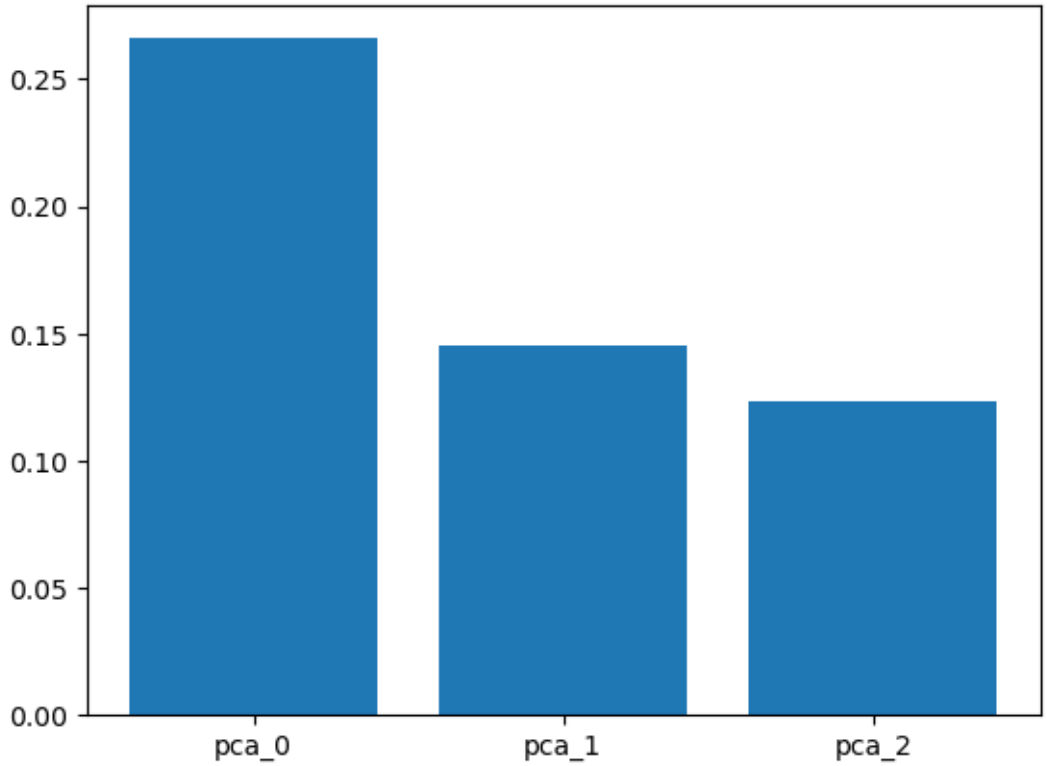
3. Klasterovanje

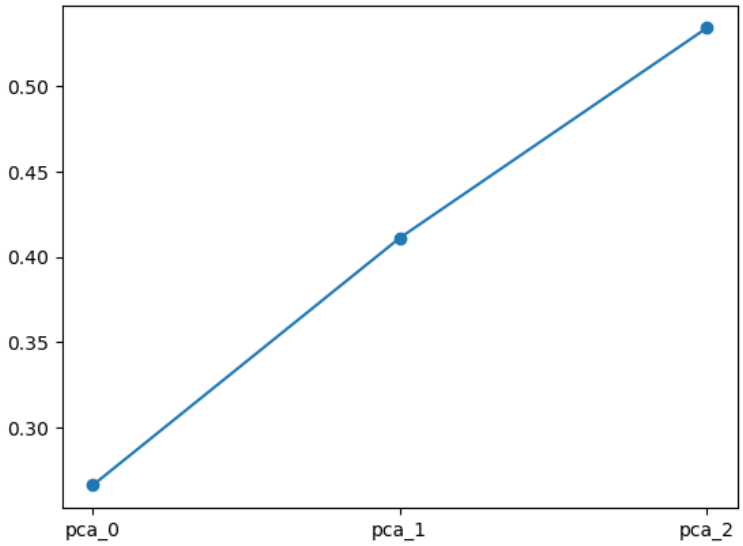
Klasterovanje je osnovona tehnika u oblasti mašinskog(nenadgledanog) učenja i analize podataka koja se koristi za grupisanje sličnih instanci ili tačaka u iste klastere dok klasteri treba da budu što različitiji. Ova tehnika nam pomaže u razumevanju prirodnih struktura i obrazaca u podacima. Dve prikazane metode za klasterovanje su "K-sredina" (K-means) i algoritam sakupljajućeg hijerarhijskog klasterovanja (agglomerative techniques).

Klasterizacijski algoritmi poput "K-sredina" i "algoritma sakupljajućeg hijerarhijskog klasterovanja" su osetljivi na prisustvo autlajera u podacima, kao i na neskalirane atribute koji mogu uticati na njihovu performansu. Stoga, pre primene ovih algoritama, sproveli smo skaliranje podataka kako bismo uskladili numeričke opsege atributa i osigurali da algoritmi ne budu previše uticajni na atribute sa većim vrednostima. Takođe smo koristili tehniku smanjivanja dimenzionalnosti PCA (Principal Component Analysis), kako bismo smanjili broj atributa i omogućili vizualizaciju podataka u nižoj dimenziji, čime smo olakšali razumevanje strukture podataka i identifikaciju klastera.

3.1 Analiza glavnih komponenti PCA

PCA radi tako što transformiše originalne atribute u novi skup nezavisnih atributa, nazvane glavne komponente, koje su linearna kombinacija originalnih atributa. Glavna svrha PCA je da smanji broj atributa, ali zadrži što više varijabilnosti iz podataka. Primenom PCA na skup podataka, smanjili smo dimenzionalnost podataka sa deset atributa na tri glavne komponente. Time smo zadržali ključne informacije iz originalnih atributa, dok istovremeno smanjujemo kompleksnost skupa podataka. PCA transformacija nam omogućava da zadržimo više od 50% varijanse originalnih podataka u tri glavne komponente.





3.2 K-sredina

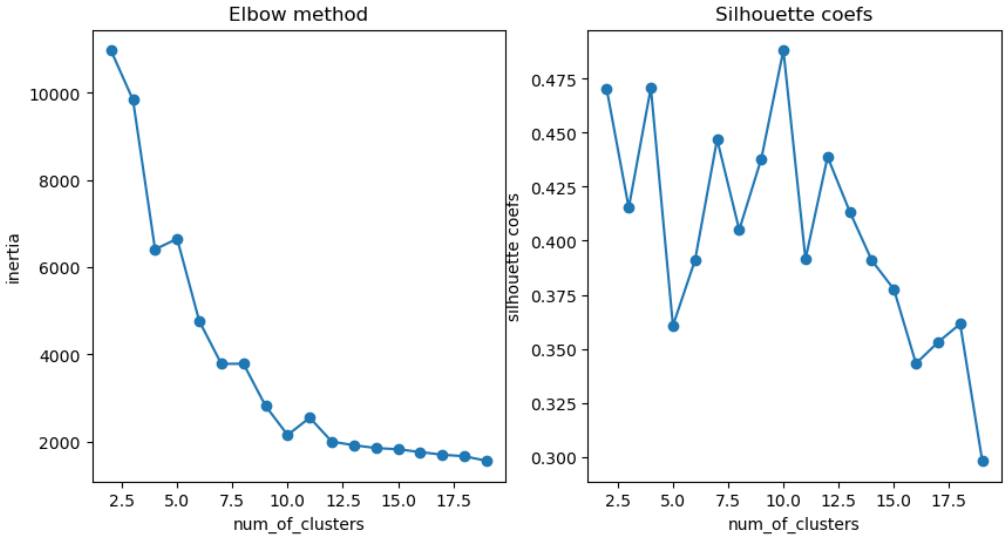
Algoritam K-sredina (K-means) je algoritam za klasterovanje koji funkcioniše tako što deli skup podataka u K klastera, gde je K unapred definisani broj klastera. Ovaj algoritam počinje sa nasumičnim izborom K centara (reprezentativnih tačaka) za svaki klaster. Zatim se iterativno izvršava sledećim koracima:

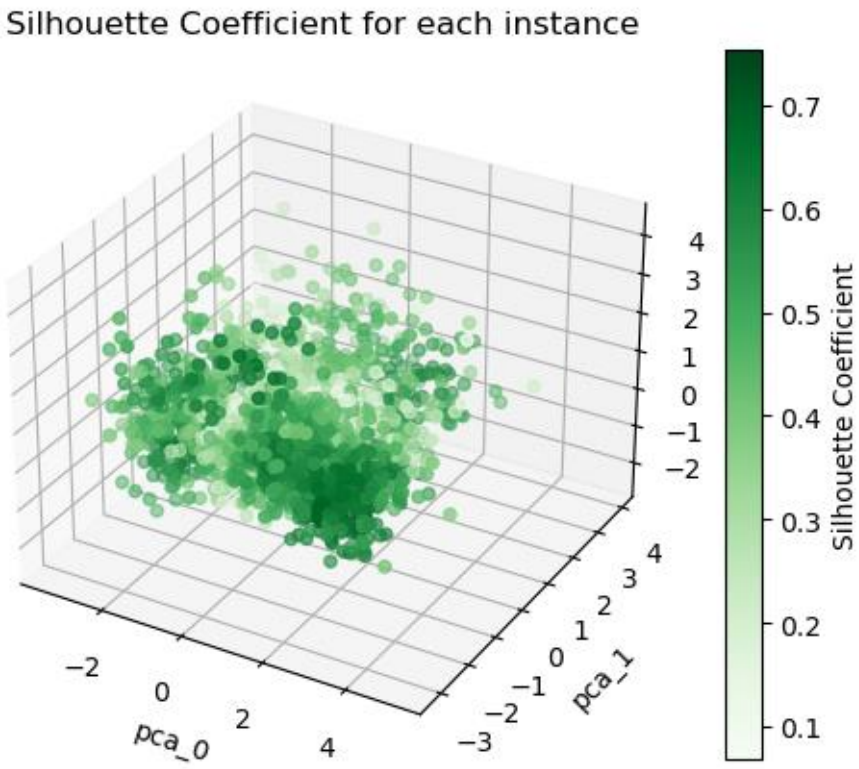
- Svaka tačka se dodeljuje najbližem centru, formirajući time klasterove
- Centri se ponovo računaju kao srednje vrednosti tačaka u svakom klasteru
- Postupak se ponavlja dok se centri ne stabilizuju ili dok ne bude ispunjen kriterijum zaustavljanja.

Konačni rezultat je podela podataka u K klastera, pri čemu se minimizuje suma kvadratnih udaljenosti između tačaka i centara klastera. K-sredina je brz i skalabilan algoritam, ali može biti osetljiv na inicijalnu nasumičnu selekciju centara i nije garantovan da će konvergirati ka globalnom optimumu.

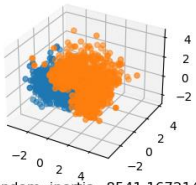
Biranje optimalnog broja klastera K u algoritmu K-sredina je ključno za dobijanje kvalitetnih klastera. Postoji nekoliko metoda za određivanje optimalnog K, a dve često korištene metrike su SSE (Sum of Squared Errors) i silhouette koeficijent.

- SSE (Sum of Squared Errors): SSE meri sumu kvadratnih udaljenosti svake tačke od njenog najbližeg centra klastera. Ideja je da se SSE minimizuje kako bi se postigla kompaktnost klastera, ali što je K veće, SSE će obično biti manje. Zato se često koristi "elbow metoda," gde se SSE meri za različite vrednosti K, a zatim se traži tačka na grafikonu gde se smanjenje SSE usporava, stvarajući izgled "laktanja" na grafiku. Ta tačka se smatra optimalnim K.
- Silhouette koeficijent: Silhouette koeficijent meri koliko je svaka tačka slična tačkama u svom klasteru u poređenju sa tačkama u drugim klasterima. Vrednost Silhouette koeficijenta varira između -1 (loše klasterovanje) i 1 (dobro klasterovanje). Da biste odabrali optimalni K pomoću Silhouette koeficijenta, možete izračunati ovu metriku za različite vrednosti K i odabrati K sa najvišim Silhouette koeficijentom.

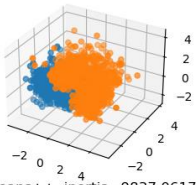

Elbow method
inertia
num_of_clusters
Silhouette coefs
silhouette coefs
num_of_clusters


Silhouette Coefficient for each instance
pca_0
pca_1
Silhouette Coefficient

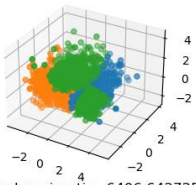
k=2, init=random, inertia=10976.654376412047



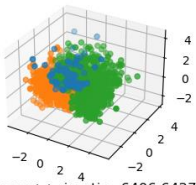
k=2, init=k-means++, inertia=10976.654376412049



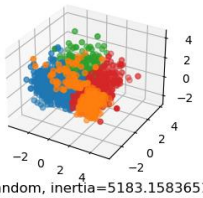
k=3, init=random, inertia=8541.167214570438



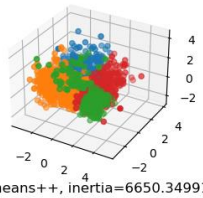
k=3, init=k-means++, inertia=9837.061764379194



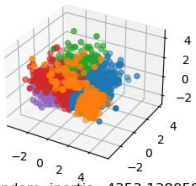
k=4, init=random, inertia=6406.642725114426



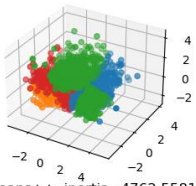
k=4, init=k-means++, inertia=6406.642725114426



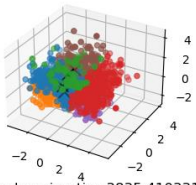
k=5, init=random, inertia=5183.158365191007



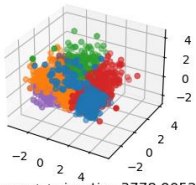
k=5, init=k-means++, inertia=6650.349919487733



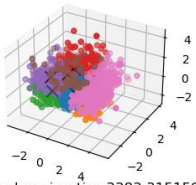
k=6, init=random, inertia=4353.138058027614



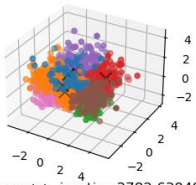
k=6, init=k-means++, inertia=4762.558174779405



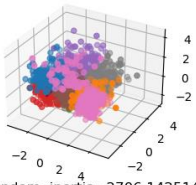
k=7, init=random, inertia=3835.410331381281



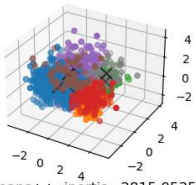
k=7, init=k-means++, inertia=3778.905232012551



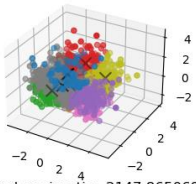
k=8, init=random, inertia=3383.315159471204



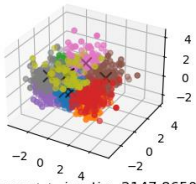
k=8, init=k-means++, inertia=3782.6394926969774



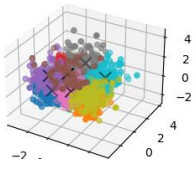
k=9, init=random, inertia=2706.142514507065



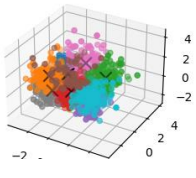
k=9, init=k-means++, inertia=2815.053520476047



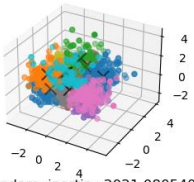
k=10, init=random, inertia=2147.8650699616105



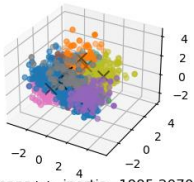
k=10, init=k-means++, inertia=2147.8650699616105



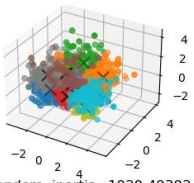
k=11, init=random, inertia=2049.420246397707



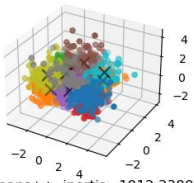
k=11, init=k-means++, inertia=2543.7525101968176



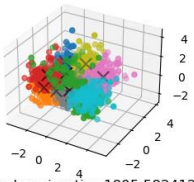
k=12, init=random, inertia=2021.0805408594106



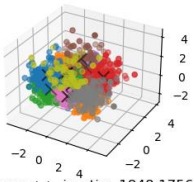
k=12, init=k-means++, inertia=1995.2070033273685



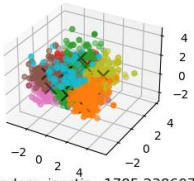
k=13, init=random, inertia=1929.493924847839



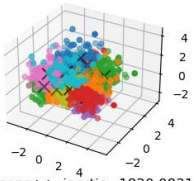
k=13, init=k-means++, inertia=1912.3380702032457



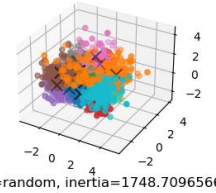
k=14, init=random, inertia=1895.5824125655333



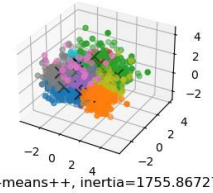
k=14, init=k-means++, inertia=1848.1756149193332



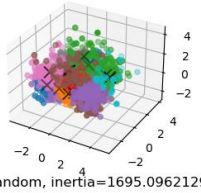
k=15, init=random, inertia=1785.2386073011162



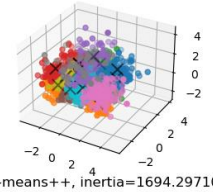
k=15, init=k-means++, inertia=1820.0831023783257



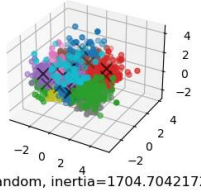
k=16, init=random, inertia=1748.709656659492



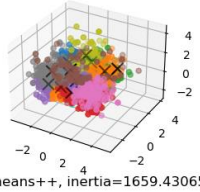
k=16, init=k-means++, inertia=1755.867275572985



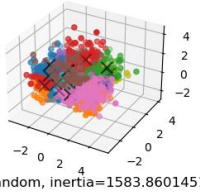
k=17, init=random, inertia=1695.0962129310842



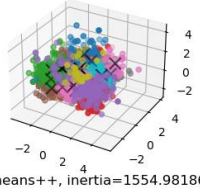
k=17, init=k-means++, inertia=1694.297108466414



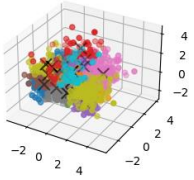
k=18, init=random, inertia=1704.7042172216898



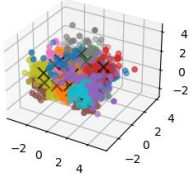
k=18, init=k-means++, inertia=1659.4306571173538



k=19, init=random, inertia=1583.8601451210395

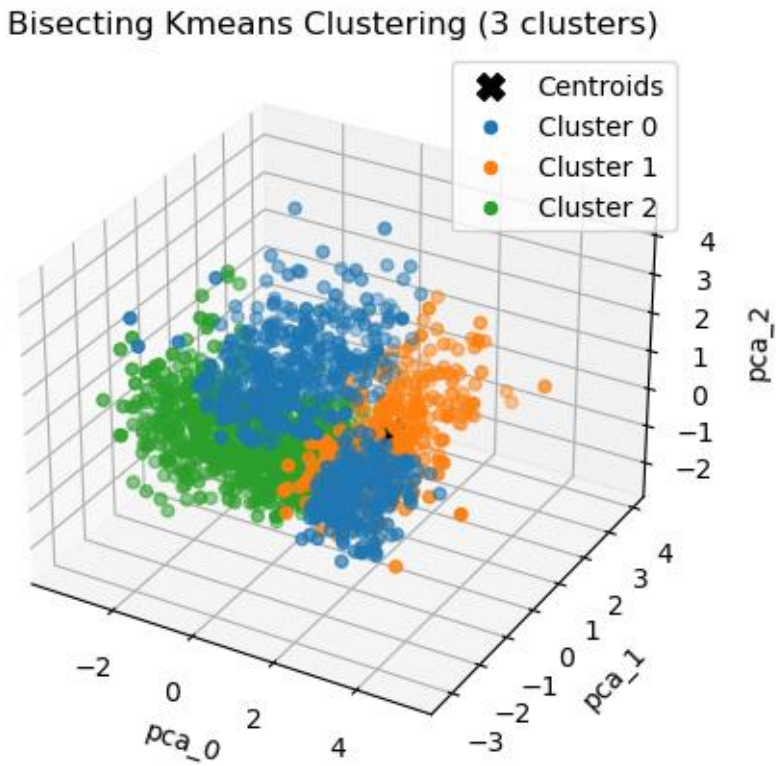


k=19, init=k-means++, inertia=1554.9818615417507



3.2.1 Bisekting K-Means

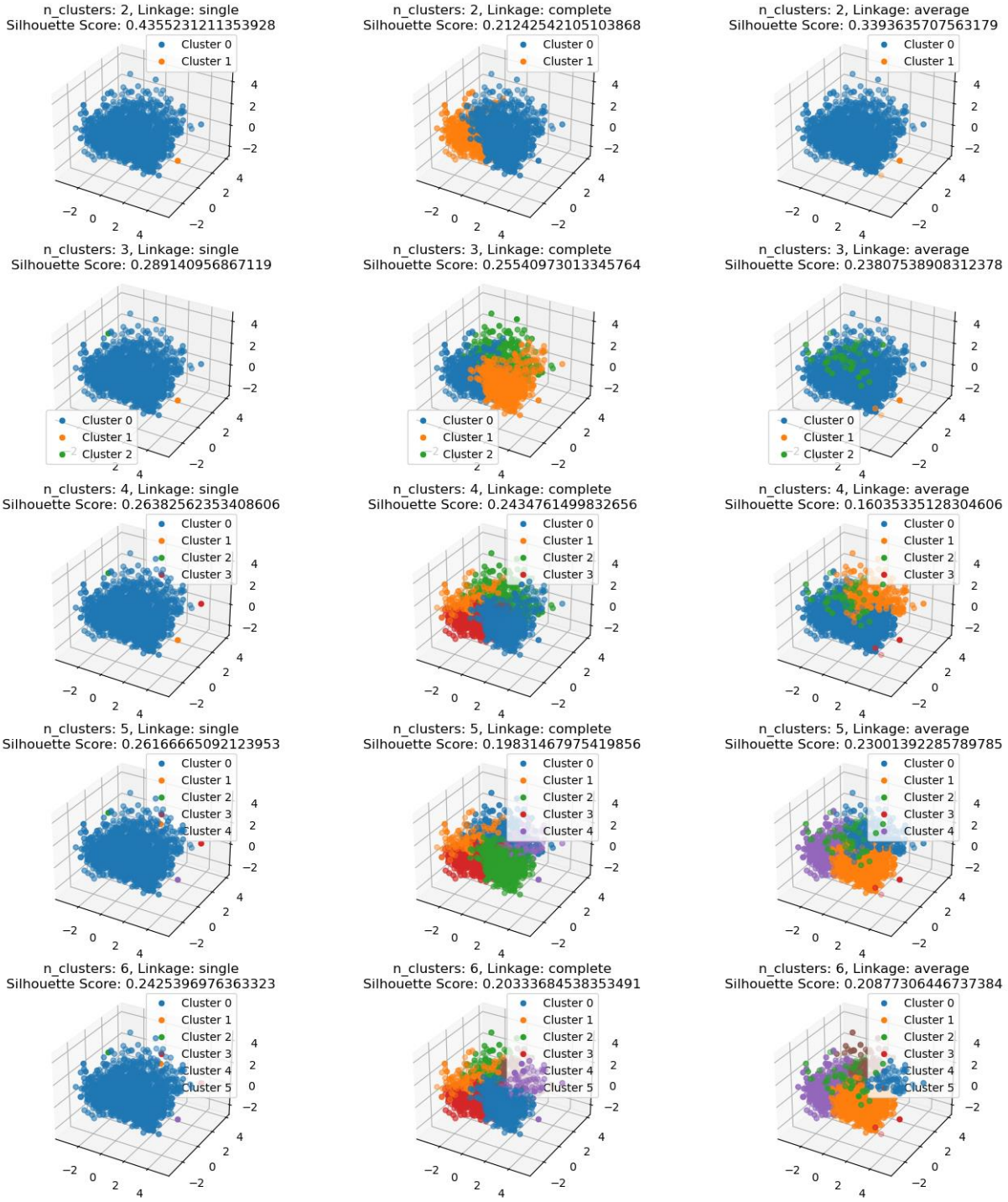
Bisekting K-Means (Bisektni K-sredina) je varijacija algoritma K-sredina koja se koristi za podelu podataka na K klastera putem hijerarhijskog pristupa. Ovaj algoritam počinje sa jednim klasterom koji obuhvata sve tačke i zatim iterativno deli klaster na dva manja klastera tako da se minimizira suma kvadratnih udaljenosti unutar svakog podklastera. Ovaj proces se ponavlja dok se ne dostigne unapred definisani broj klastera K ili se ispunjava određeni kriterijum zaustavljanja.



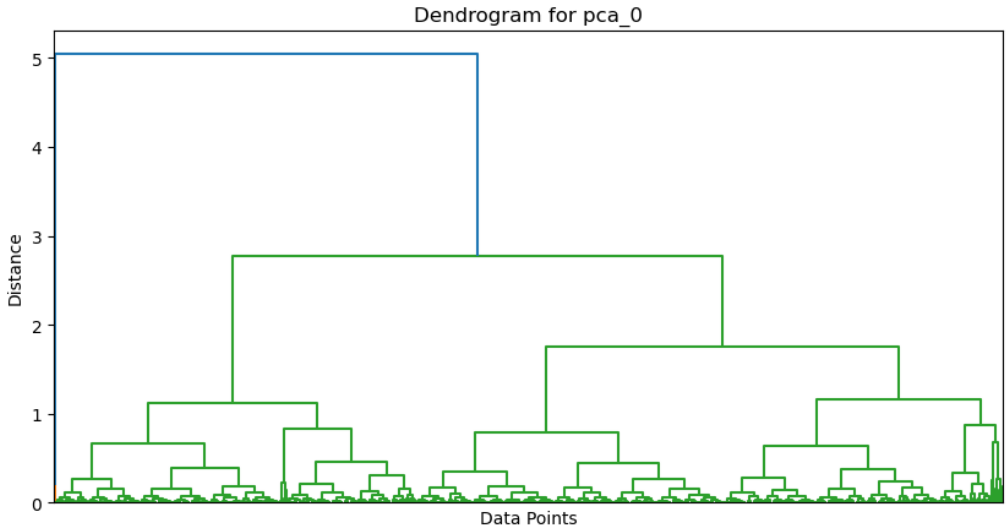
3.3 Algoritam sakupljajućeg hijerarhijskog klasterovanja

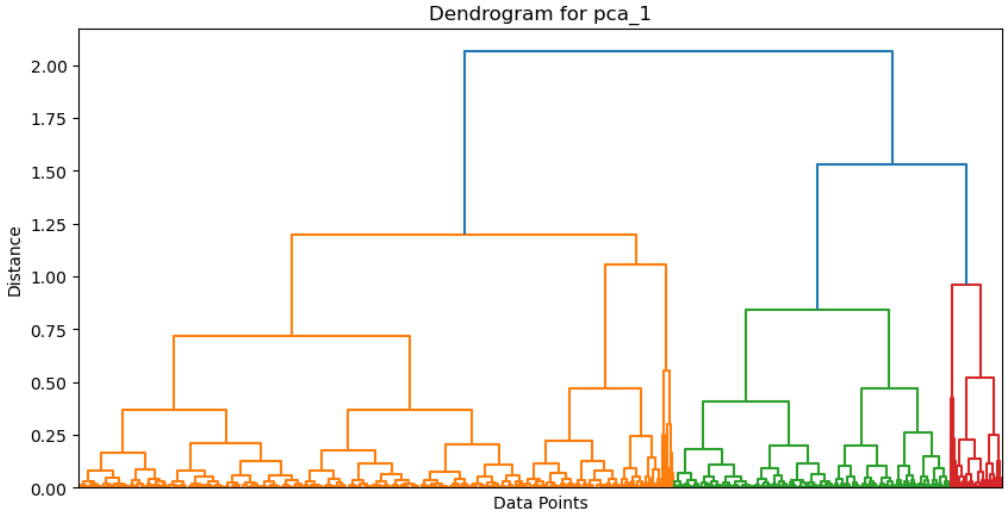
Algoritam sakupljajućeg hijerarhijskog klasterovanja je metoda klasterovanja koja počinje sa svakom tačkom kao zasebnim klasterom i iterativno spaja najbliže klasterove kako bi se formirali hijerarhijski klasteri. Ovaj proces se nastavlja sve dok se ne formira jedan krajnji klaster koji obuhvata sve tačke ili dok se dostigne unapred određeni broj klastera. Postoji nekoliko metoda za merenje udaljenosti, kao što su pojedinačno povezivanje (single linkage) - udaljenost najbližih instanci između dva klastera, kompletan povezivanje (complete linkage) – rastojanje između najudaljenijih instanci između dva klastera, i srednje povezivanje (average linkage) – rastojanje između centroida dva klastera, između ostalih. Rezultat ovog procesa je hijerarhijska struktura klastera koja omogućava analizu podataka na različitim nivoima granularnosti.

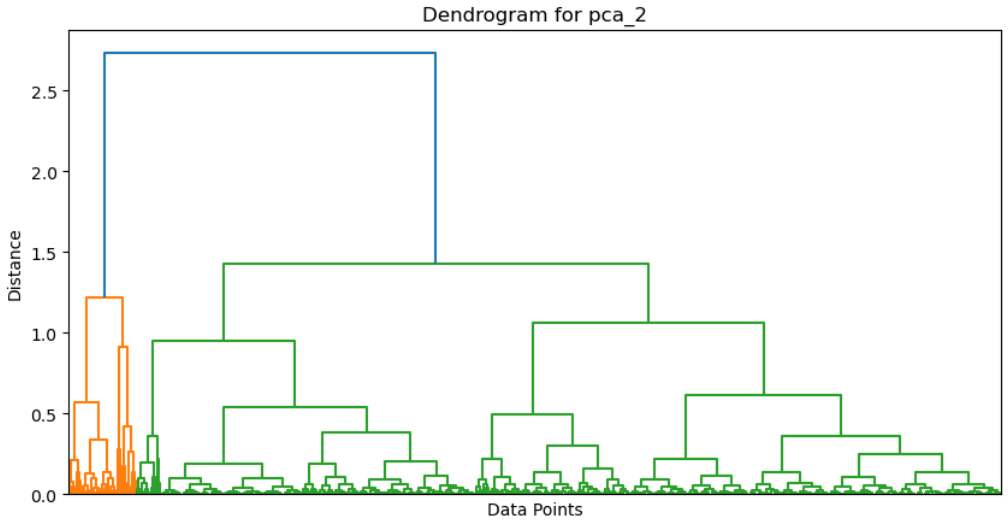
Na sledećoj slici je prikazano klasterovanje primenom ove tehnike za različite vrednosti parametara, a takođe i silhouette koeficijent za svaki od tih klastera.



Dendrogram je grafički prikaz hijerarhijske strukture klastera dobijene primenom tehnike klasterovanja sa sakupljajućim hijerarhijskim pristupom. Ovaj dijagram vizualizuje način na koji su tačke u skupu podataka grupisane i povezane hijerarhijski, počevši od svake tačke kao pojedinačnog klastera, pa sve do konačnog klastera koji obuhvata sve tačke.Na dendrogramu se na vertikalnoj osi obično prikazuje merilo udaljenosti između klastera ili tačaka.


Dendrogram for pca_0
Distance
Data Points
0
1
2
3
4
5


Dendrogram for pca_1
Distance
Data Points
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

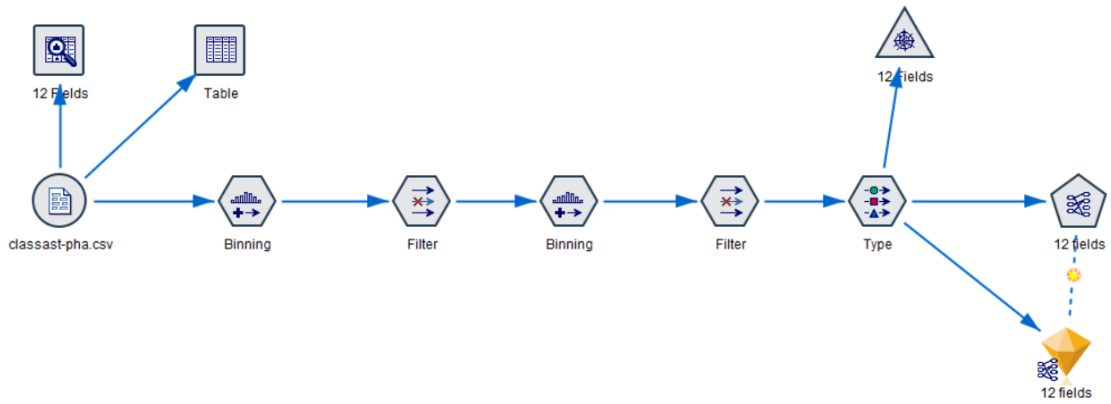

Dendrogram for pca_2
Distance
Data Points
0.0
0.5
1.0
1.5
2.0
2.5

4. Pravila pridruživanja

Pravila pridruživanja u okviru Data Mininga su tehnika analize podataka koja se koristi za identifikaciju čestih veza i obrazaca između stavki u velikim datasetima. Ova tehnika otkriva pravila u obliku "ako-onda," gde se "ako" deo odnosi na uslove ili kombinaciju stavki, dok se "onda" deo odnosi na posledice ili stavke koje su često povezane sa tim uslovima.

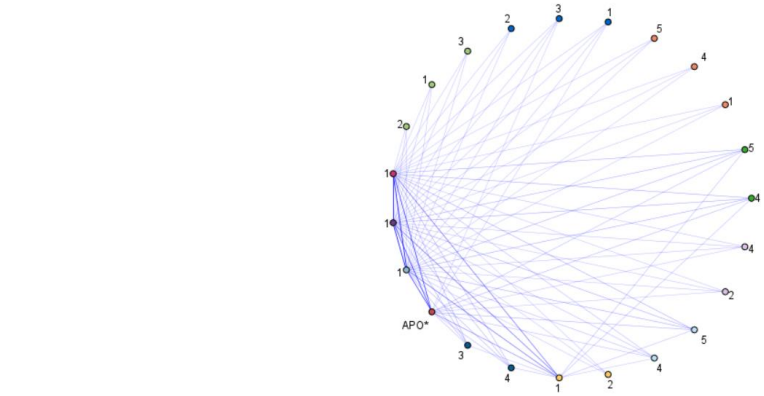
4.1 Apriori algoritam

Apriori algoritam je tehnika analize pravila pridruživanja koja funkcioniše tako što počinje sa jednočlanim stavkama koje se često pojavljuju u datasetu, računa koliko često se svaka stavka pojavljuje (podrška), a zatim postavlja prag podrške kako bi identifikovao česte stavkovne skupove. Zatim generiše nove kandidate za više stavki, ponavlja proces računanja podrške i eliminacije kandidata koji ne ispunjavaju prag podrške.



Read Values | Clear Values | Clear All Values

Field	Measurement	Values	Missing	Check	Role
a (AU)	Continuous	[0.635222...		Coerce	Both
e	Continuous	[0.025424...		Coerce	Both
i (deg)	Continuous	[0.146083...		Coerce	Both
w (deg)	Continuous	[0.521837...		Coerce	Both
Node (deg)	Continuous	[0.136041...		Coerce	Both
M (deg)	Continuous	[0.052165...		Coerce	Both
q (AU)	Continuous	[0.0928,1....		Coerce	Both
Q (AU)	Continuous	[0.96,34.68]		Coerce	Both
P (yr)	Continuous	[0.51,75.22]		Coerce	Both
H (mag)	Continuous	[14.1,22.4]		Coerce	Both
MOID (AU)	Continuous	[1.0E-5,0.0...		Coerce	Both
class	Nominal	AMO*,APO...		Coerce	Both



a (AU)_BIN class e_BIN H (mag)_BIN i (deg)_BIN M (deg)_BIN MOID (AU)_BIN Node (deg)_BIN P (yr)_BIN q (AU)_BIN Q (AU)_BIN w (deg)_BIN

Sort by: Confidence % 65 of 65

Consequent	Antecedent	Support %	Confidence %
class = APO*	q (AU)_BIN = 4 e_BIN = 4	12.357	99.537
class = APO*	w (deg)_BIN = 2 q (AU)_BIN = 4	10.469	97.268
class = APO*	q (AU)_BIN = 4 H (mag)_BIN = 4	11.67	96.569
class = APO*	e_BIN = 4 H (mag)_BIN = 4	12.3	95.349
class = APO*	q (AU)_BIN = 4 i (deg)_BIN = 1	20.195	95.184
class = APO*	q (AU)_BIN = 4	29.176	94.51
class = APO*	w (deg)_BIN = 2	22.769	94.221
class = APO*	e_BIN = 4 i (deg)_BIN = 1	26.087	94.079
class = APO*	w (deg)_BIN = 2 i (deg)_BIN = 1	15.675	93.796
class = APO*	e_BIN = 4	34.039	93.445
class = APO*	w (deg)_BIN = 4 i (deg)_BIN = 1	16.362	93.007
class = APO*	w (deg)_BIN = 4	24.886	92.874
class = APO*	q (AU)_BIN = 4 H (mag)_BIN = 5	11.041	91.71
class = APO*	e_BIN = 4 H (mag)_BIN = 5 i (deg)_BIN = 1	10.584	91.351
class = APO*	H (mag)_BIN = 3 i (deg)_BIN = 1	10.526	91.304
class = APO*	MOID (AU)_BIN = 1		