Analiza skupa podataka CS:GO Round Winner Classification

Seminarski rad u okviru kursa Istraživanje podataka 1

Matematički fakultet

Bogdan Stojadinović
mi20073@alas.matf.bg.ac.rs

Maj 2023.

UVOD

U ovom seminarskom radu prikazan je proces rada sa podacima iz skupa preuzetog sa sajta Kaggle i možete im pristupiti pomoću linka. Skup podataka sadrži informacije o rundama igrice CS:GO, a prikupljeni su od strane Skybox-a kao deo njihovog CS:GO AI Challenge-a. Ukupno ima 122.411 instanci, svaka koja je opisana sa 97 atributa.

Cilj ovog rada je provesti detaljnu analizu skupa podataka i primeniti različite tehnike mašinskog učenja kako bismo napravili model koji može predvideti pobednika runde u CS:GO-u. Ukoliko želite da saznate nešto više o igrici CS:GO to možete učiniti na sledećem linku.

EKSPLORATIVNA ANALIZA PODATAKA

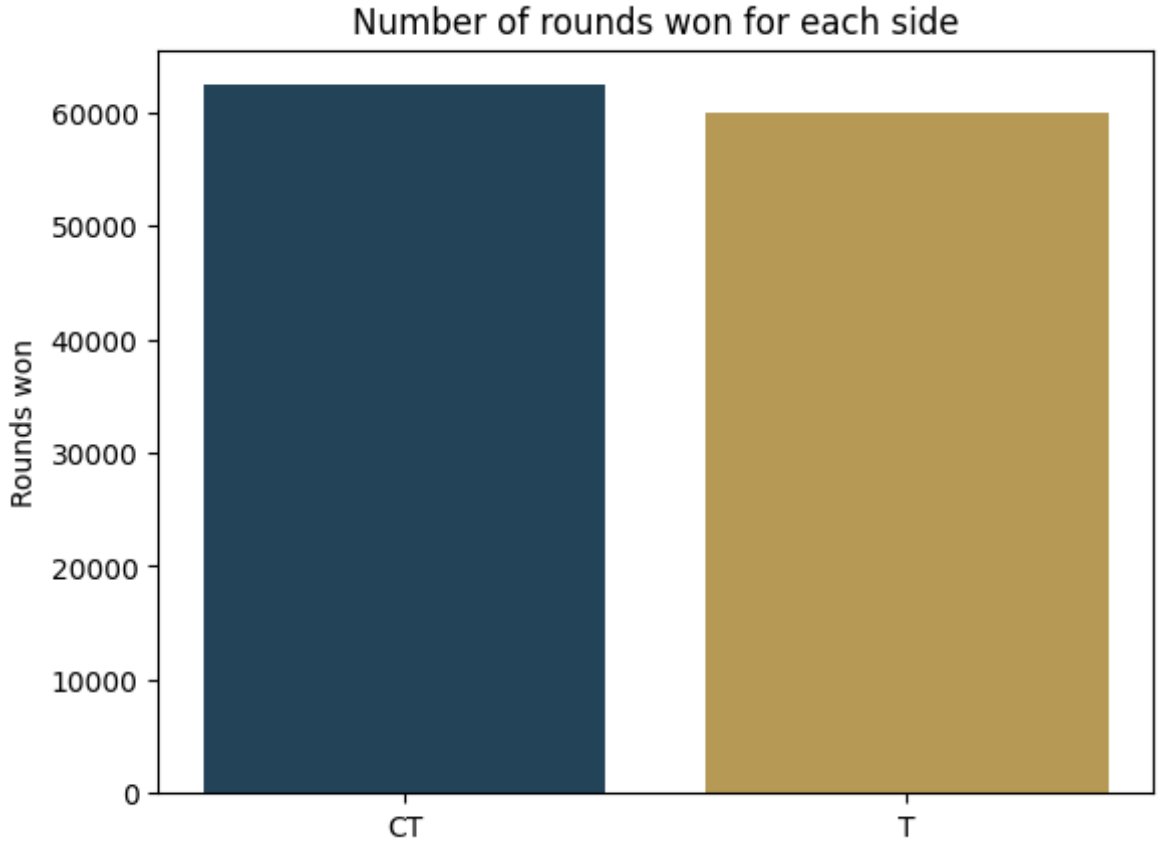
Pre početka izgradnje modela potrebno je izvršiti detaljnu eksplorativnu analizu podataka kako bismo se bolje upoznali i razumeli podatke sa kojima radimo. Imamo 97 atributa koji opisuju razne elemente runde u CS:GO-u, vreme preostalo do kraja runde, broj igrača na obe strane, oružja koja igrači koriste... Kompletna lista atributa:

- time_left — Vreme preostalo do kraja trenutne runde
- ct_score — Broj rundi koje je pobedila CT strana
- t_score — Broj rundi koje je pobedila T strana
- map — Mapa na kojoj se igra runda
- bomb_planted — Indikator da li je bomba postavljena
- ct_health — Ukupan broj životnih poena na CT strani
- t_health — Ukupan broj životnih poena na T strani
- ct_armor — Ukupan broj oklopnih poena na CT strani
- t_armor — Ukupan broj životnih poena na T strani
- ct_money — Ukupna količina novca na CT strani
- t_money — Ukupna količina novca na T strani
- ct_helmets — Ukupan broj kaciga na CT strani
- t_helmets — Ukupan broj kaciga na T strani
- ct_defuse_kits — Ukupan broj kompleta za deaktiviranje bombe na CT strani
- ct_players_alive — Ukupan broj živih igrača na CT strani
- t_players_alive — Ukupan broj živih igrača na T strani
- ct_weapon_X — Ukupan broj za svako od oružja X na CT strani
- t_weapon_X — Ukupan broj za svako od oružja X na T strani
- ct_grenade_X — Ukupan broj za svaku od granata X na CT strani
- t_grenade_X — Ukupan broj za svaku od granata X na T strani
- round_winner — Pobednik runde

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...
0	175.00	0.0	0.0	de_dust2	False	500.0	500.0	0.0	0.0	4000.0	...
1	156.03	0.0	0.0	de_dust2	False	500.0	500.0	400.0	300.0	600.0	...
2	96.03	0.0	0.0	de_dust2	False	391.0	400.0	294.0	200.0	750.0	...
3	76.03	0.0	0.0	de_dust2	False	391.0	400.0	294.0	200.0	750.0	...
4	174.97	1.0	0.0	de_dust2	False	500.0	500.0	192.0	0.0	18350.0	...

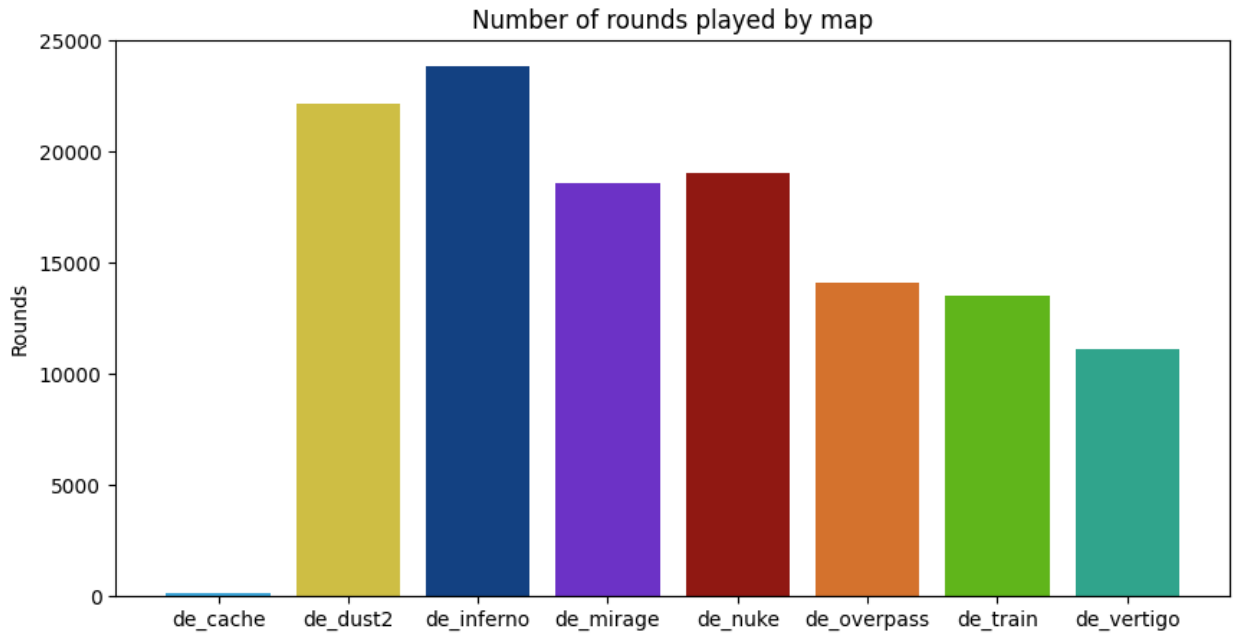
Slika 1: Prikaz skupa podataka

Koristeći atribut *round_winner* možemo da proverimo koja strana je više puta pobedila i da vidimo da li je jedna strana možda dominantnija od druge.



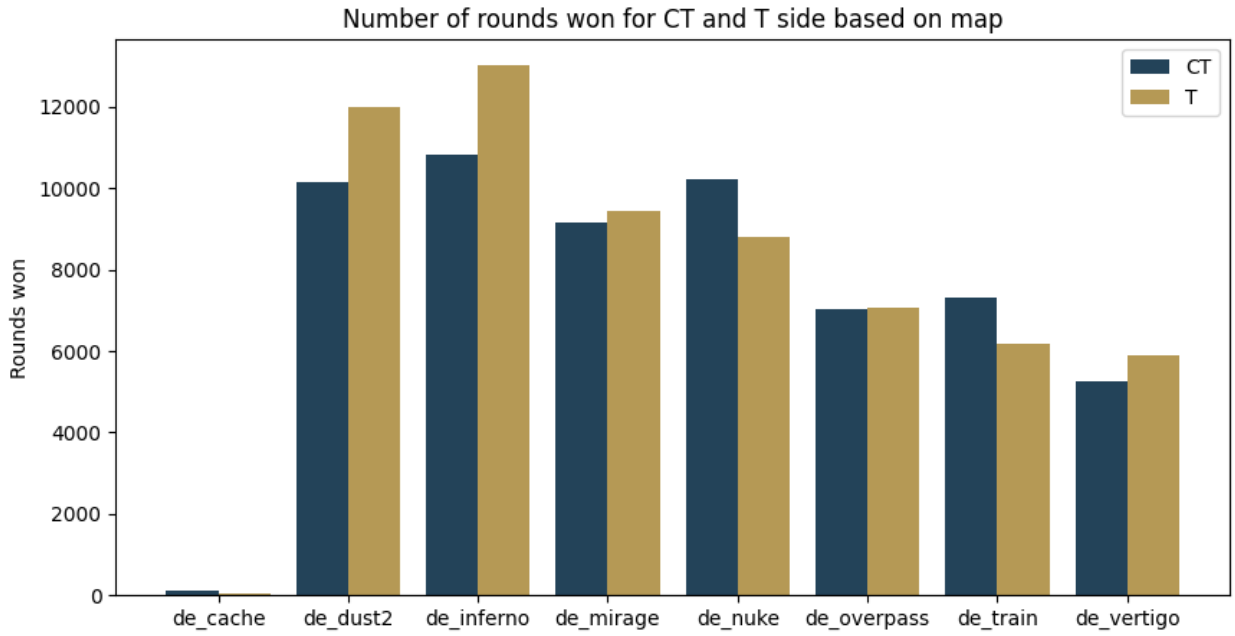
Slika 2: Broj pobeđenih rundi za svaki tim

Sa slike 2 možemo da primetimo da je broj pobeda podjednako raspoređen na obe strane, sa minimalnom prednošću za CT. Ovo nam odgovara kod problema klasifikacije zato što imamo balansirane klase i neće biti potrebe za uzorkovanjem.



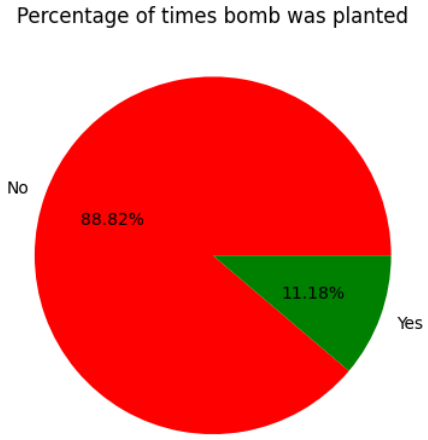
Slika 3: Broj odigranih rundi u zavisnosti od mape

Na slici 3 su prikazane sve mape na kojima su odigrane runde i možemo videti raspodelu broja rundi za svaku od njih. Sada se postavlja pitanje, da li mapa utiče na ishod runde tj da li neka strana ima prednost u zavisnosti od mape na kojoj se igra. Ako pogledamo sliku 4 vidimo da su mape dust2, inferno i vertigo više naklonjene T strani, nuke, train i cache više CT strani dok su mirage i overpass poprilično balansirane sa blagim naklonom ka T strani.

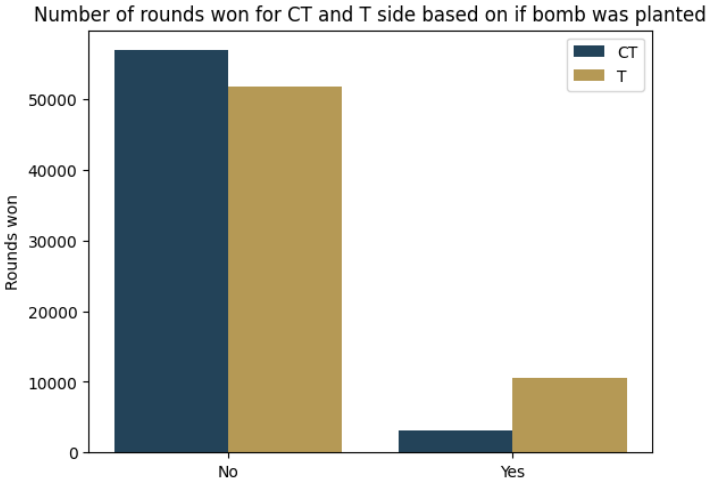


Slika 4: Broj pobeda za CT i T stranu u zavisnosti od mape

Još jedna stvar koja znatno može da utiče na krajnji ishod runde je to da li je T strana postavila bombu. Ukoliko jesu to znatno povećava njihove šanse jer otežava posao CT strani koja umesto odbrane sad mora da krene u napad. Na slici 5 vidimo da u velikoj većini slučajeva T strana nije uspela da postavi bombu ali na slici 6 vidimo da su ipak uspeli da pobede veliki broj rundi bez obzira na to. Sa druge strane, u 11% slučajeva kada jesu uspeli da postave bombu imaju znatno veći broj pobeda nego CT strana.

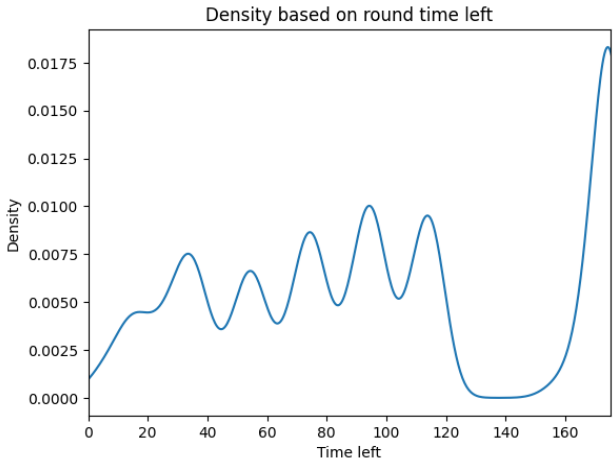


Slika 5: Da li je bomba postavljena

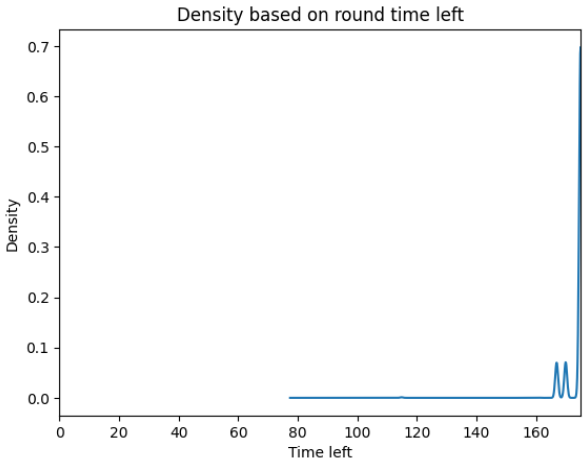


Slika 6: Broj pobeda za CT i T u zavisnosti od toga da li je bomba postavljena

Pored ovih, još jedan od bitnih atributa čini skup oružja koje igrači u timu koriste. Svaki igrač ima priliku da na početku runde kupi oružje koje želi i koje može da priušti. Ukoliko bi želeli da prikažemo koja su to oružja koja se najčešće kupuju na početku runde nailazimo na problem. Naime naš skup sadrži podatke iz raznih perioda tokom runde, kao što možemo videti na slici 7.

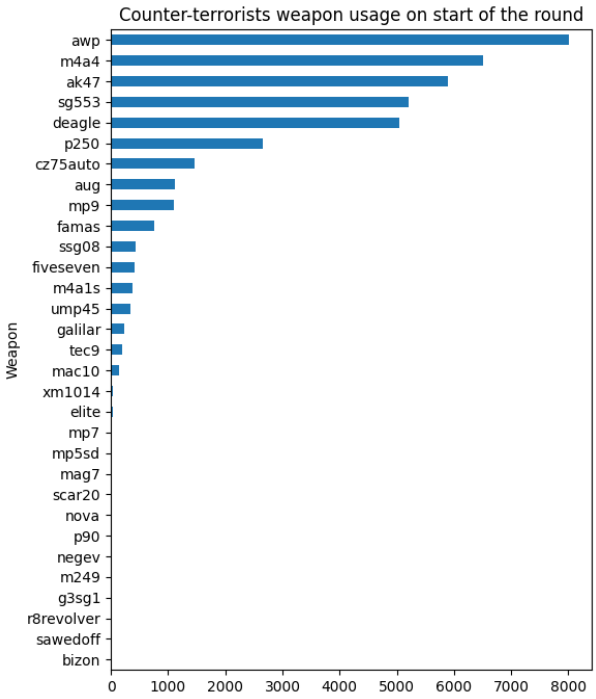


Slika 7: Gustina podataka zasnovana na atributu time_left (original)

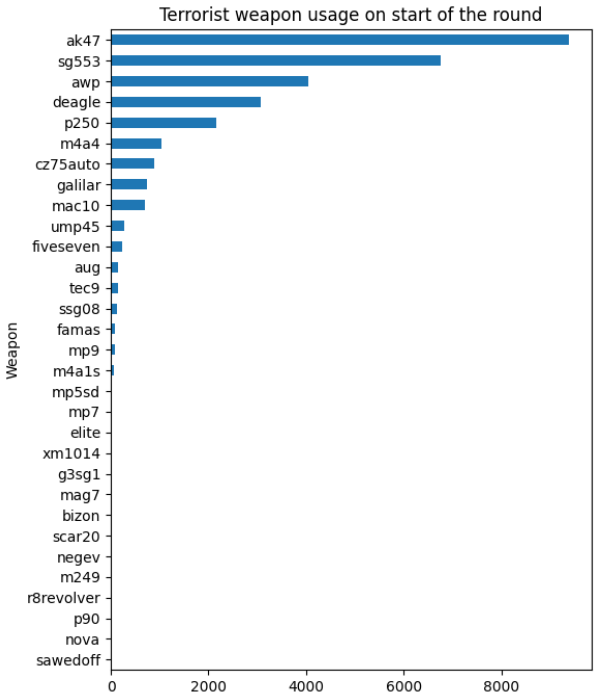


Slika 8: Gustina podataka zasnovana na atributu time_left (nakon obrade)

Ukoliko izvršimo obradu skupa tako da izbacimo sve instance koje predstavljaju jednu istu rundu samo u drugim vremenskim trenucima, ali tako da ostavimo onu instancu koja je najranija u rundi, dobićemo podatke sa početka runde kada svaki igrač bira koje oružje želi. Takođe, svaki igrač dobija besplatan pištolj na početku svake runde, glock za T stranu, dok CT strana ima priliku da bira između usps ili p2000 (odabir se dešava pre početka partije). Zbog toga izbacujemo kolone koje predstavljaju ta oružja pošto bi ona dominirala i ne bi dobili realne podatke. Nakon svih ovih obrada konačno možemo da proverimo za šta su se igrači najviše opredeljivali. Slike 9 i 10 prikazuju taj odabir za CT odnosno T stranu.

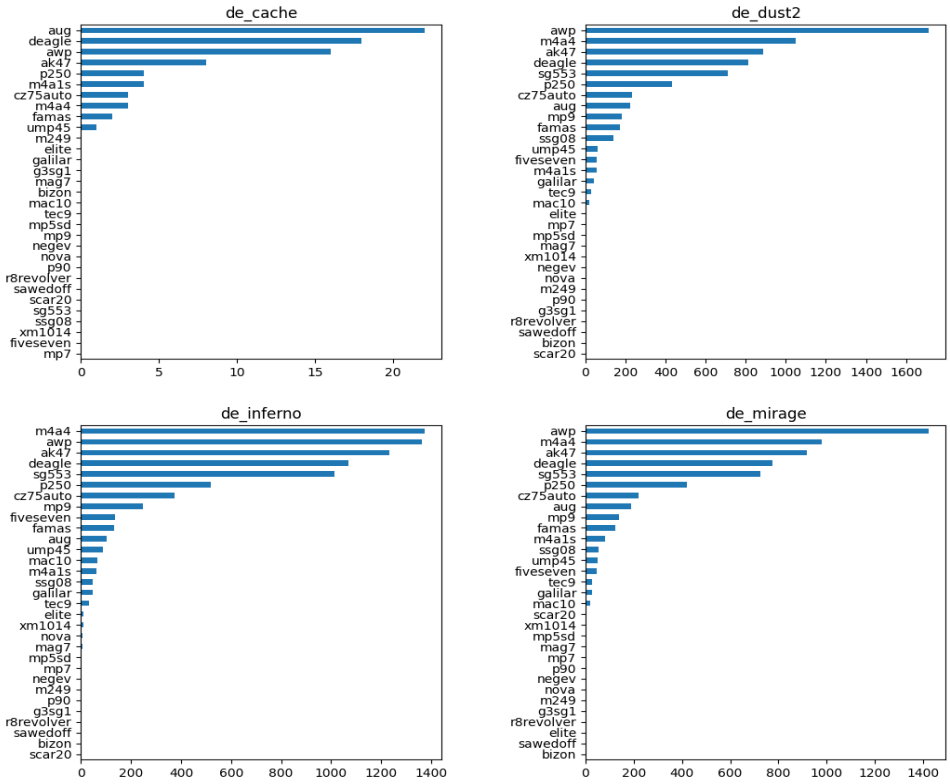


Slika 9: Najkorišćenija oružja na CT strani

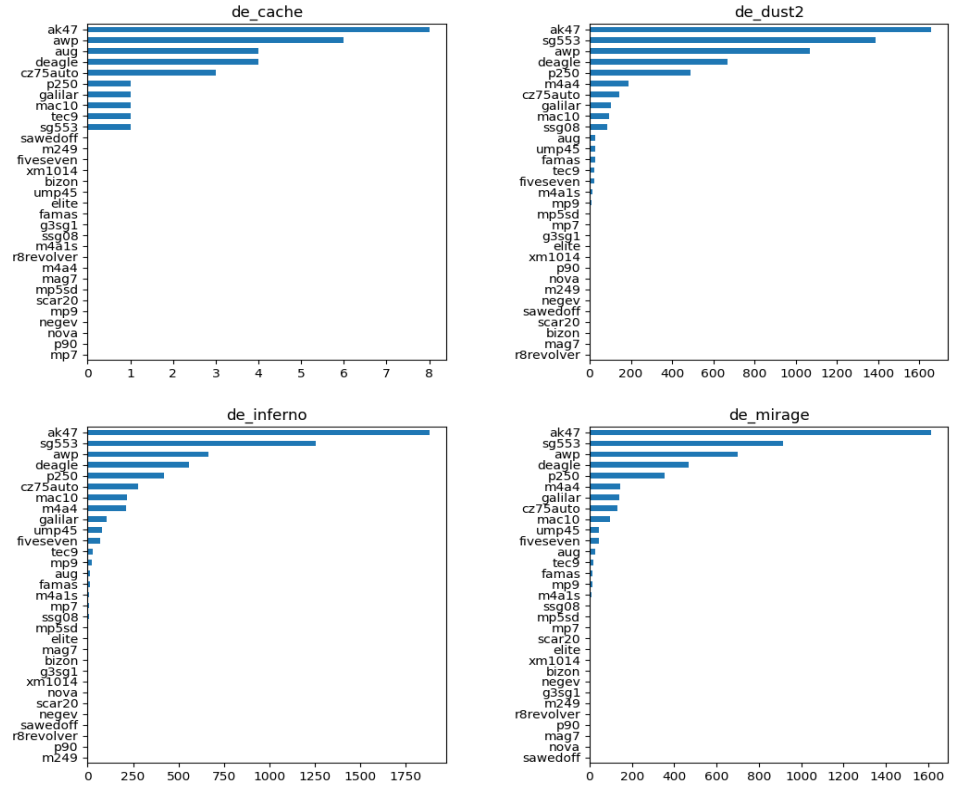


Slika 10: Najkorišćenija oružja na T strani

Sada, možemo da crtamo iste ovakve grafike, ali da ih podelimo za svaku mapu. Time primećujemo kako se neka oružja više koriste na određenim mapama. Rezultate možemo pogledati na slikama 11 i 12.



Slika 11: Najkorišćenije oružje na CT strani za neke mape



Slika 12: Najkorišćenije oružje na T strani za neke mape

PRETPROCESIRANJE PODATAKA

Sledeći korak je priprema podataka za dalju obradu. Prva stvar koju proveravamo je da li u našem skupu podataka postoje nedostajuće vrednosti. Primećujemo da u našem skupu ne postoji nijedno polje koje nije popunjeno podacima. Zbog toga nema potrebe za nikakvom obradom podataka u ovom trenutku. Od ranije znamo da su klase podataka balansirane (pogledati sliku 2 za podsetnik) pa možemo da preskočimo i tu obradu. Zatim proveravamo elemente van granica, autlajere. Pošto za svaki atribut znamo u kom opsegu su njegove moguće vrednosti (broj živih igrača u timu je od 0 do 5, ukupan broj životnih poena je od 0 do 500, ukupan broj oružja/granata je od 0 do 5...) možemo da izbacimo sve instance u kojima se vrednost nekog atributa nalaze izvan ovih opsega.

Sada imamo konačan skup podataka koje ćemo koristiti za treniranje modela, ali ostalo nam je još par koraka da bismo mogli da ih primenimo na konkretne algoritme. Odvojićemo kolonu *round_winner* jer nam ona predstavlja ciljnu promenljivu koju želimo da predviđamo kod problema klasifikacije, a nije nam uopšte potrebna kod problema klasterovanja. Potrebno je da kategoričke atribute pretvorimo u numeričke, u našem slučaju to je samo kolona *map*. S obzirom da postoji 8 različitih mapa kodiraćemo ime svake mape kao jedan broj u opsegu od 0 do 7. Nakon ovih obrada podaci su spremni za dalju pripremu za konkretne probleme klasifikacije i klasterovanja.

Za klasifikaciju nam je potrebno da podelimo skup na train skup i test skup, korišćenjem `train_test_split` funkcije iz biblioteke `sklearn`. Koristićemo 80% podataka za trening a preostalih 20% za testiranje. Za neke algoritme neophodna je normalizacija podataka. To možemo postići korišćenjem funkcije `MinMaxScaler` iz biblioteke `sklearn`, vrednosti za min i max ćemo izabrati na osnovu podataka iz trening skupa, a zatim ćemo transformaciju primeniti nad trening i test skupovima. Odabir ove tehnike normalizacije zasnovan je na tome što naš skup nema autlajere, a i svaki atribut ima određene gornje i donje granice. Sada ovako podeljene i standardizovane podatke možemo sačuvati uz pomoć biblioteke `joblib` i kasnije učitavati i koristiti za klasifikaciju.

Kod pripreme podataka za klasterovanje nema potrebe podeliti podatke na trening i test, pošto nemamo tačno zadato rešenje problema i nemamo šta da testiramo. Podatke zato skaliramo nad celim skupom i onda su spremni za čuvanje i dalje korišćenje.

KLASIFIKACIJA

DECISION TREES

Prvi algoritam koji ćemo koristiti jesu stabla odlučivanja (eng. Decision Trees). Ukratko, algoritam radi na principu razdvajanja podataka u različite grupe na osnovu svojstava ili atributa. Počinje s postavljanjem pitanja o karakteristikama podataka i gradi stablo s granama koje odgovaraju različitim odgovorima na ta pitanja. Svaki čvor stabla predstavlja neki uslov, a primeri podataka se dele na osnovu tih uslova. Nakon izgradnje stabla, primeri se klasifikuju putujući od korena do listova, gde svaki list predstavlja konačnu klasifikaciju primera.

Koristeći `DecisionTreeClassifier` bez ikakvih podešavanja hiperparametara dobijamo rezultate sa slike 13. Primećujemo da su rezultati na trening skupu skoro savršeni dok na test skupu uspešno pogađamo samo ~83%. Očigledno je da je došlo do preprilagođavanja i poželjno je da ponovo izvršimo algoritam uz dodatno podešavanje hiperparametara. Za to cemo koristiti unakrsnu validaciju `GridSearchCV`. Nakon što izaberemo koje hiperparametre želimo da testiramo, `GridSearchCV` će odabrati kombinaciju hiperparametara koja daje najbolje rezultate na osnovu unakrsne validacije. Na slici 14 vidimo nove rezultate i primećujemo da nam unakrsna validacija nije pomogla.

```
Train data:

Confusion matrix:
 [[47918   114]
 [  181 49697]]
Accuracy score:  0.9969870289040956
Precision score:  0.9977113488988376
Recall score:  0.9963711455952524
F1 score:  0.9970407968782915
```

```
Test data:

Confusion matrix:
 [[ 9812  2152]
 [ 2118 10396]]
Accuracy score:  0.8255576435983332
Precision score:  0.8284985655084476
Recall score:  0.8307495604922487
F1 score:  0.829622536110446
```

Slika 13: Rezultati klasifikacije korišćenjem stabla odlučivanja

```
Train data:                                  Test data:

Confusion matrix:                            Confusion matrix:
 [[47687   345]                               [[ 9776  2188]
 [  221 49657]]                               [ 2077 10437]]
Accuracy score:  0.9942191808804004          Accuracy score:  0.8257619086526677
Precision score:  0.9931002759889604         Precision score:  0.8266930693069307
Recall score:  0.9955691888207225            Recall score:  0.8340258910020777
F1 score:  0.9943331998398078                F1 score:  0.8303432913003699
```

Slika 14: Rezultati klasifikacije korišćenjem stabla odlučivanja sa unakrsnom validacijom

RANDOM FOREST

Algoritam klasifikacije slučajne šume (eng. Random Forest) je ansambl metoda koja kombinuje više stabala odlučivanja. Svako stablo se trenira na nasumično izabranim podskupovima podataka, a klasifikacija se vrši glasanjem više stabala. Ovo povećava stabilnost, smanjuje preprilagodljivost i poboljšava generalizaciju modela. Odluke se donose na osnovu većine glasova iz stabala, pružajući pouzdanu klasifikaciju.

Odmah na prvi pogled sa slike 15 vidimo da je ovaj algoritam bolji od običnih stabala odlučivanja, jer smo bez podešavanja hiperparametara dobili bolje rezultate na test podacima. Test skup sada uspešno pogađa ~88% instanci, dok je trening opet skoro perfektan. Problem preprilagođavanja je i dalje prisutan, što će se ispostaviti kao konstantan problem kod svakog modela. Unakrsna validacija nam opet nije pomogla, podešavanjem boljih hiperparametara dobijamo malo bolje rezultate (slika 16)

```
Train data:                                  Test data:

Confusion matrix:                            Confusion matrix:
 [[47880   152]                               [[10598  1366]
 [  143 49735]]                               [ 1626 10888]]
Accuracy score:  0.9969870289040956          Accuracy score:  0.8777677914862325
Precision score:  0.9969531140377252         Precision score:  0.8885261955279908
Recall score:  0.9971330045310558            Recall score:  0.8700655266101965
F1 score:  0.99704305117025                  F1 score:  0.8791989664082687
```

Slika 15: Rezultati klasifikacije korišćenjem slučajnih šuma

```
Train data:                                  Test data:

Confusion matrix:                            Confusion matrix:
 [[47877   155]                               [[10641  1323]
 [  140 49738]]                               [ 1558 10956]]
Accuracy score:  0.9969870289040956          Accuracy score:  0.8823024756924586
Precision score:  0.9968933517727938         Precision score:  0.8922550696310775
Recall score:  0.9971931512891455            Recall score:  0.8754994406264983
F1 score:  0.9970432289943971                F1 score:  0.8837978461662567
```

Slika 16: Rezultati klasifikacije korišćenjem slučajnih šuma sa unakrsnom validacijom

KNN

Algoritam klasifikacije k-najbližih suseda (eng. K Nearest Neighbors, KNN) je jednostavan i intuitivan algoritam koji se koristi za klasifikaciju. Radi na principu pronalaženja k najbližih suseda (instanci) u skupu podataka na osnovu njihove udaljenosti od novog primera koji treba klasifikovati. Klasifikacija se vrši na osnovu većine klasa suseda, gde se novi primer dodeljuje najčešćoj klasi među susedima. Kod KNN jako je bitno kako izaberemo parametar k. Ako uzmemo preveliko k naš model će uvek predviđati klasu koja ima veći broj elemenata, što ne želimo.

Bez podešavanja hiperparametara naš model konačno nije preprilagođen, sada imamo ~89% na trening skupu, dok je test skup na ~82% (slika 17). Ipak nismo zadovoljni ovim rezultatima pa se ponovo okrećemo ka `GridSearchCV`. Unakrsnom validacijom dobili smo da najbolje rezultate daje k = 25. Trening skup se ponovo nalazi u poznatoj teritoriji od preko 99%, dok je test napredovao čak 2% do neverovatnih ~84% (slika 18).

```
Train data:                                  Test data:

Confusion matrix:                            Confusion matrix:
 [[42912  5120]                               [[ 9816  2148]
 [ 5445 44433]]                               [ 2288 10226]]
Accuracy score:  0.8920947809212542          Accuracy score:  0.8187760437944277
Precision score:  0.8966762859968115         Precision score:  0.8264102149668661
Recall score:  0.8908336340671238            Recall score:  0.8171647754514944
F1 score:  0.8937454113908138                F1 score:  0.8217614914818387
```

Slika 17: Rezultati klasifikacije korišćenjem k najbližih suseda

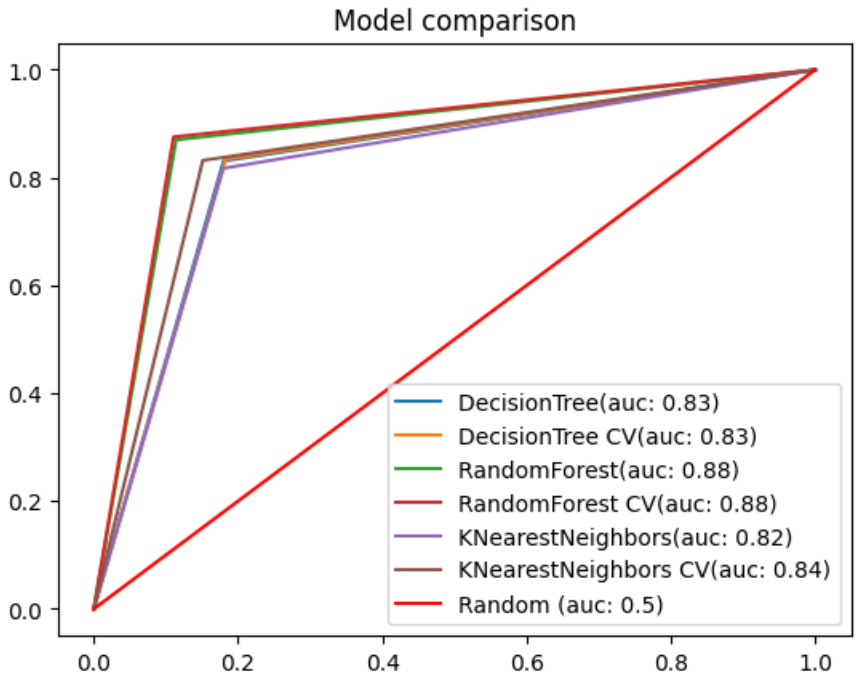
```
Train data:                                  Test data:

Confusion matrix:                            Confusion matrix:
 [[47932   100]                               [[10156  1808]
 [  197 49681]]                               [ 2104 10410]]
Accuracy score:  0.9969666019814115          Accuracy score:  0.8401830214886837
Precision score:  0.9979912014624054         Precision score:  0.8520216074643968
Recall score:  0.9960503628854405            Recall score:  0.8318683074956049
F1 score:  0.9970198376463741                F1 score:  0.8418243571081999
```

Slika 18: Rezultati klasifikacije korišćenjem k najbližih suseda sa unakrsnom validacijom

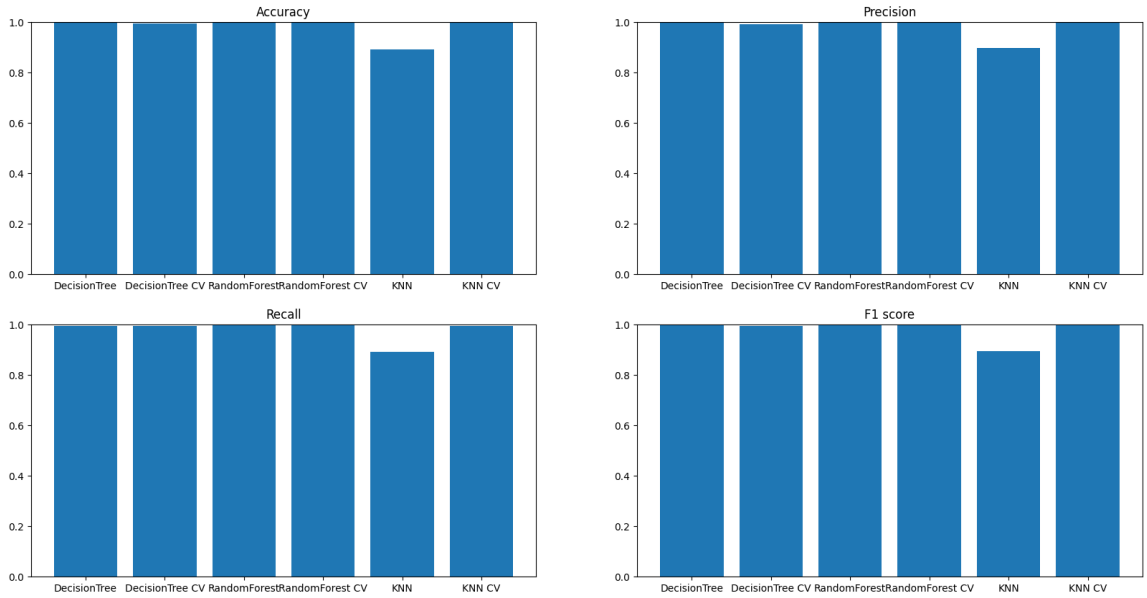
POREĐENJE MODELA

Sada kada imamo istrenirane sve modele možemo međusobno da ih uporedimo i vidimo koji daje najbolje rezultate. Za to ćemo koristiti ROC krivu (eng. Receiver Operating Characteristic curve). Prikazuje odnos između stopa lažno pozitivnih (FPR) i stopa istinito pozitivnih (TPR). Idealna ROC kriva ide prema gornjem levom uglu, što znači visok TPR i nizak FPR. Površina ispod ROC krive (AUC - Area Under the Curve) koristi se kao mera performansi modela, gde veća vrednost AUC ukazuje na bolji model. Za AUC = 0.5 imamo model koji nasumično pogađa klase.

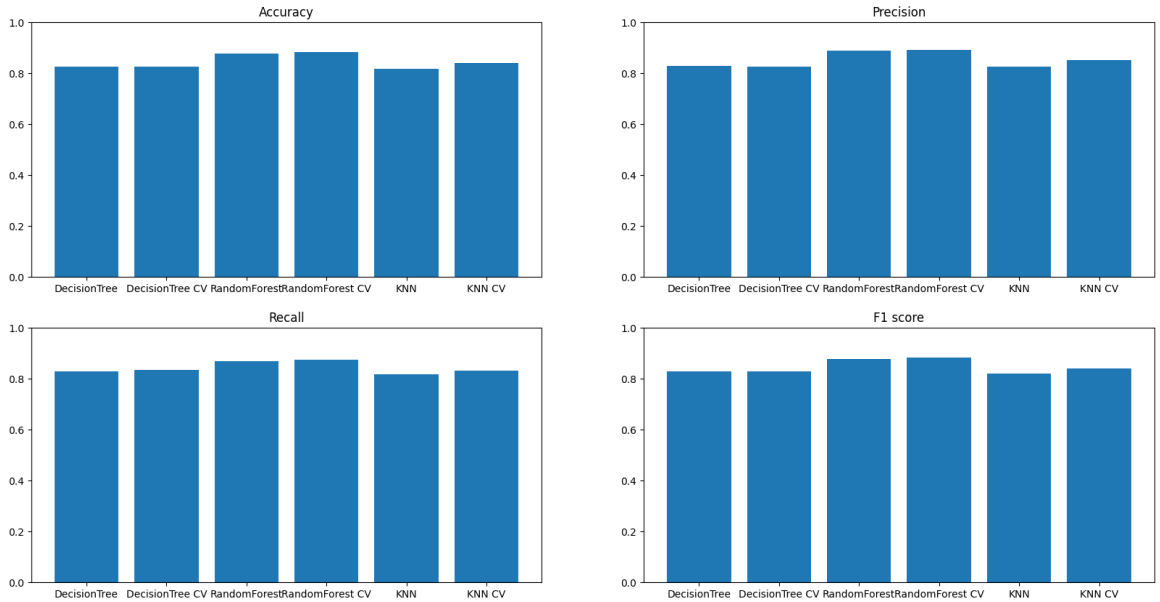


Slika 19: Poređenje modela korišćenjem ROC krive

Na slici 20 su grafički prikazani rezultati za četiri od najkorišćenijih metrika kod klasifikacije izračunate nad trening podacima: tačnost, preciznost, odziv i f1 skor. Jedino se KNN model sa podrazumevanim hiperparametrima izdvaja zbog svojih lošijih rezultata, dok su ostali komfortno u 99-om percentilu. Ipak, ovi podaci su nam manje bitni, značajnija nam je slika 21 koja sadrži rezultate nad test podacima. Nijedan model nije dao preterano dobre rezultate ali se metod slučajnih šuma najbolje pokazao. Stabla odlučivanja i KNN su približno isti, a primećujemo i da nam unakrsna validacija skoro uopšte nije pomogla da dođemo do boljih rezultata klasifikacije.



Slika 20: Poređenje različitih metrika nad trening skupom



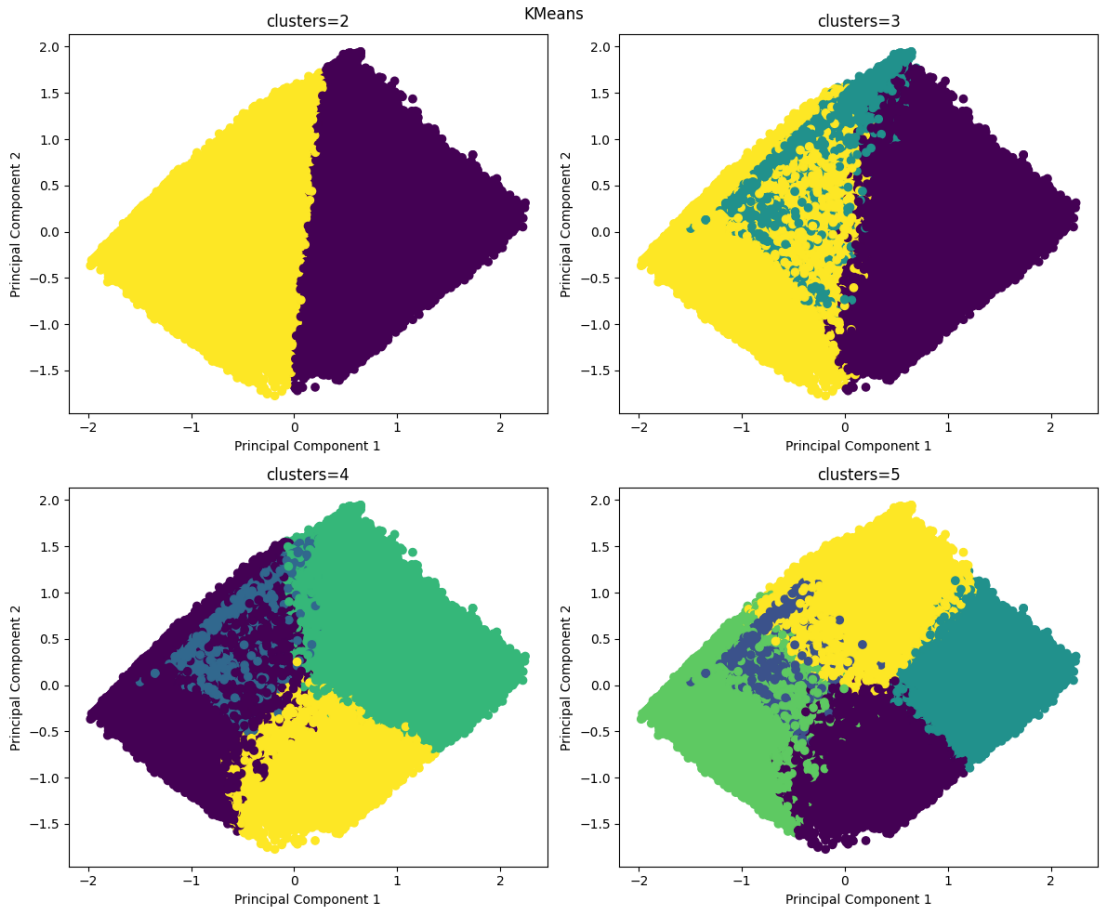
Slika 21: Poređenje različitih metrika nad test skupom

KLASTEROVANJE

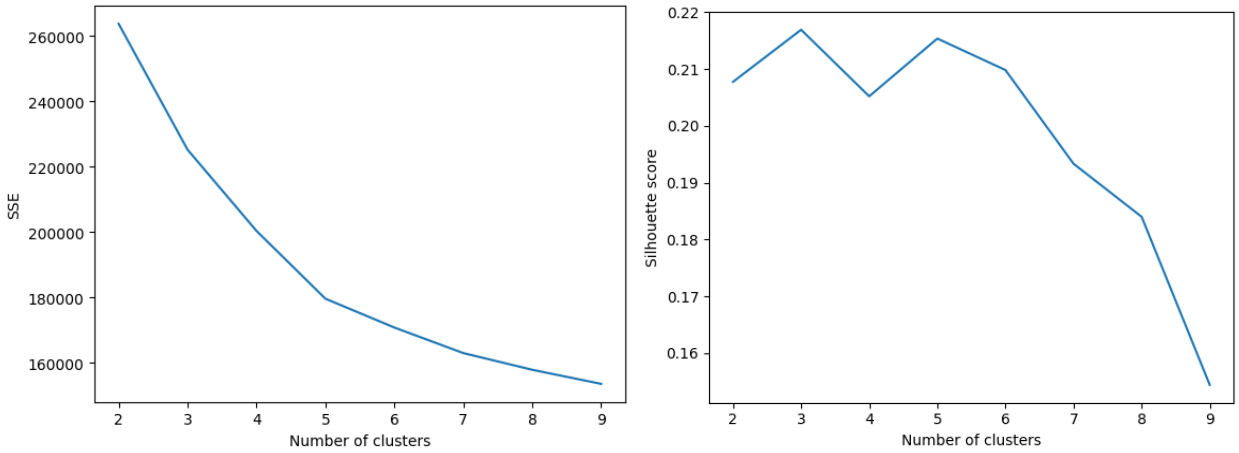
KMEANS

K-means je jedan od najpoznatijih algoritama za klasterovanje podataka. Algoritam počinje nasumičnim odabirom K centara klastera, a zatim iterativno pridružuje svaki podatak najbližem centroidu i ažurira njegove pozicije kako bi se minimizirala udaljenost unutar klastera. Proces se ponavlja sve dok se centroidi ne stabilizuju. K-means je brz i jednostavan algoritam, ali može biti osetljiv na početni odabir centroida i nepravilne oblike klastera.

Koristeći PCA (Principal Component Analysis), smestili smo naše podatke u 2D prostor i sada možemo da vizuelizujemo rezultate K-means. Pokretanjem ovog algoritma više puta za različite vrednosti parametra *k* koji predstavlja broj klastera koje tražimo, dobijamo raznolike rezultate. Tokom izvršavanja računaćemo i pamtićemo vrednosti za SSE (Sum of Squares Errors) i silhouette score (slika 23).

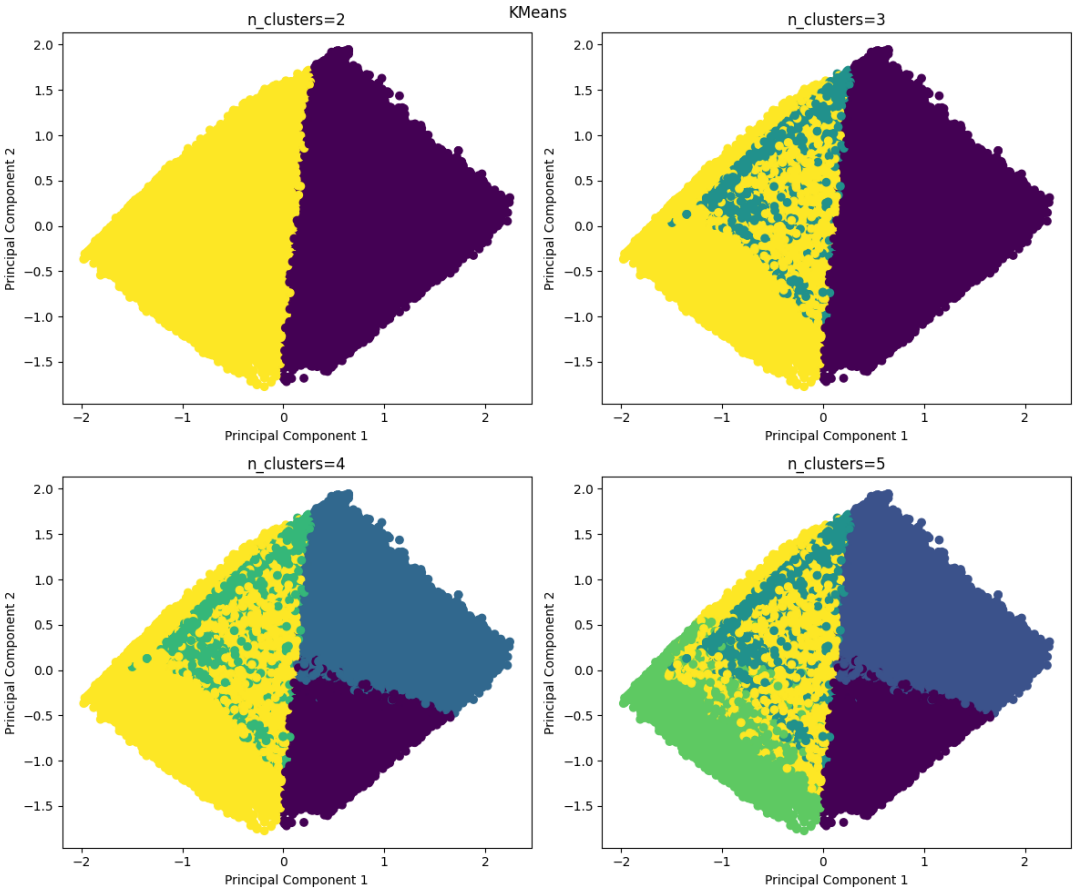


Slika 22: Prikaz klastera dobijenih primenom K-means za neke vrednosti k



Slika 23: Metrike ze procenu kvaliteta K-means algoritma

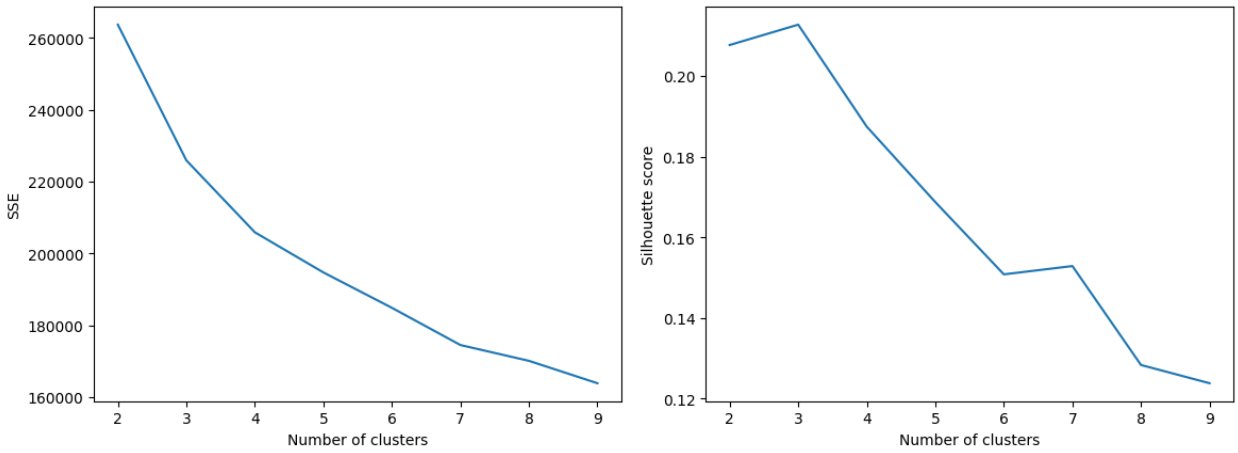
Na osnovu datih rezultata deluje kao da naš skup najbolje reaguje kada ga podelimo na 3 ili 5 klastera, što je kontradiktorno sa onim što mi znamo. Pokušaćemo da primenimo varijaciju K-means algoritma, Bisecting K-means.



Slika 24: Prikaz klastera dobijenih primenom Bisecting K-means za neke vrednosti k

BISECTING KMEANS

Ovaj algoritam započinje sa jednim klasterom koji sadrži sve podatke i iterativno ga deli na dva manja. Deljenje se vrši tako da se trenutni klaster podeli na dva manja koristeći K-means algoritam. Zatim se odabere onaj koji ima više elemenata ili veću grešku i ponavlja se postupak sve dok ne dobijemo željeni broj podela.



Slika 25: Metrike za procenu kvaliteta Bisecting K-means algoritma

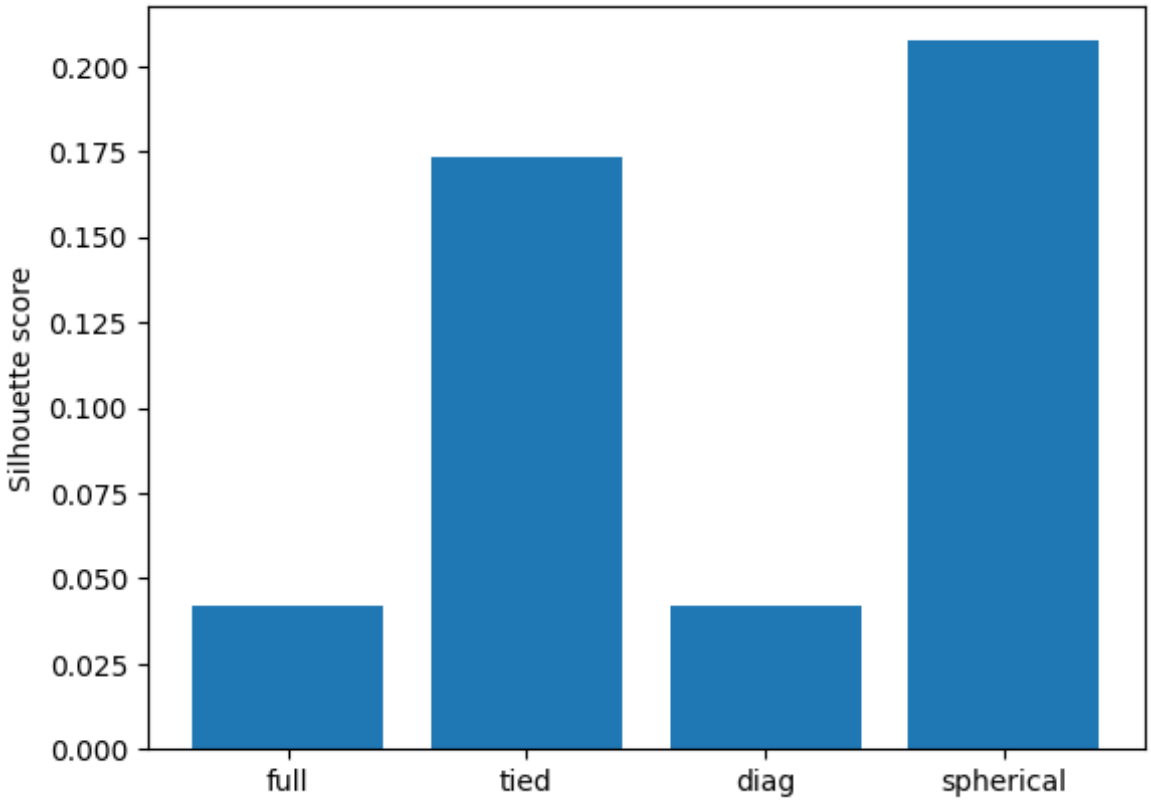
Kod ove tehnike primećujemo lošije rezultate za veće brojeve klastera, dok za k = 2 i k = 3 dobijamo veoma slične rezultate kao i za običan K-means i stoga ovaj algoritam odbacujemo.

GAUSSIAN MIXTURE

Gaussian Mixture Model (GMM) je algoritam za klasterovanje koji se zasniva na kombinaciji više Gausovih (normalnih) distribucija. Svaki klaster u GMM modelu predstavlja jednu Gausovu distribuciju sa svojim parametrima, kao što su srednja vrednost i kovarijansa. GMM modeluje podatke kao kombinaciju tih Gausovih distribucija i koristi algoritam očekivanja i maksimizacije za određivanje najboljih parametara. Klasifikacija novih podataka se vrši na osnovu verovatnoće pripadnosti svakom klasteru. Ovaj algoritam je koristan za klasterovanje podataka s različitim oblicima i gustinama, ali može biti osetljiv na odabir početnih parametara i zahteva pravilno određivanje broja klastera. Takođe može identifikovati podskupove podataka koji se preklapaju ili su složeni.

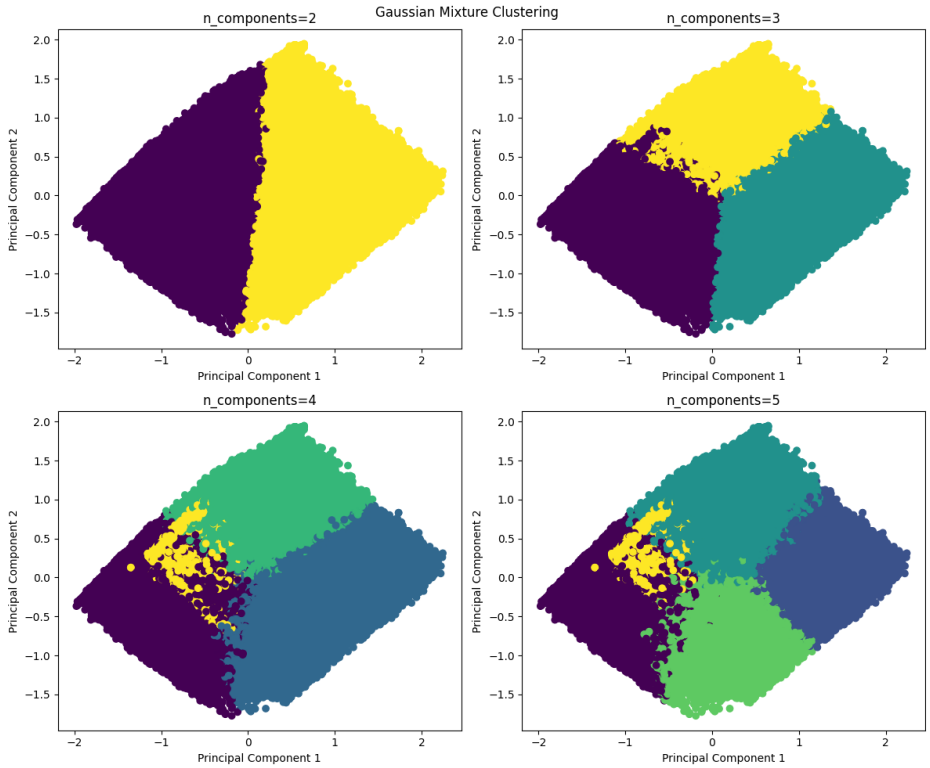
Imamo 4 različite vrste kovarijanse koje možemo da koristimo:

- 'full': svaka komponenta ima svoju opštu matricu kovarijanse
- 'tied': sve komponente dele istu opštu matricu kovarijanse
- 'diag': svaka komponenta ima svoju dijagonalnu matricu kovarijanse
- 'spherical': svaka komponenta ima svoju pojedinačnu varijansu.

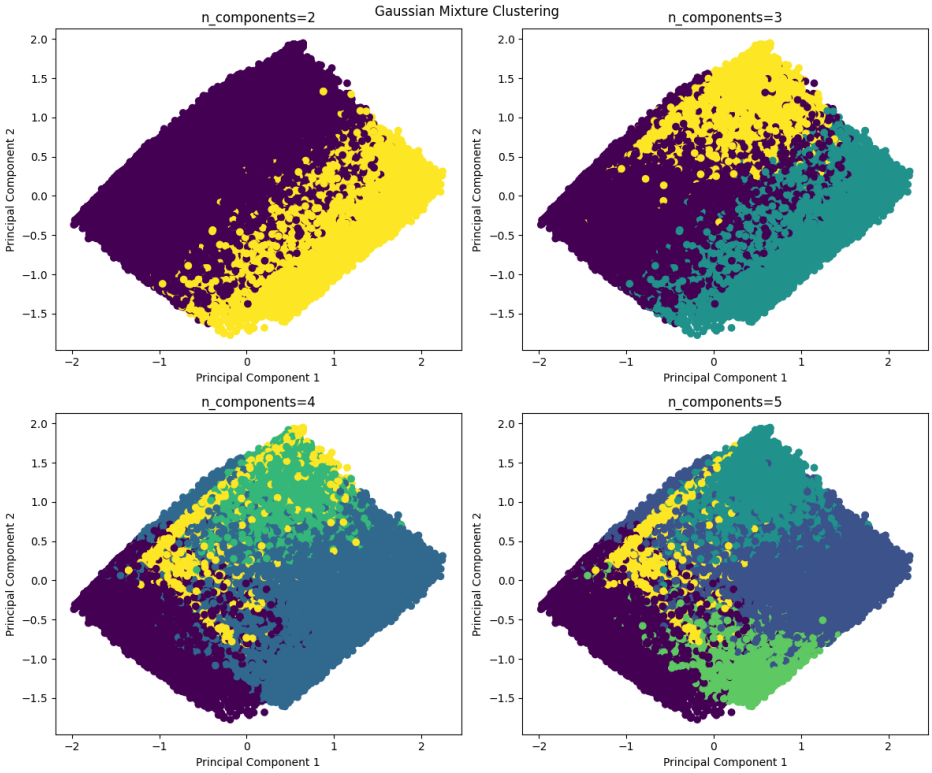


Slika 26: Izbor tipa kovarijanse

Ukoliko izaberemo fiksan broj klastera koji tražimo i pokrenemo algoritam za svaku od ove 4 mogućnosti, poređenjem po silhouette score-u na slici 26 možemo primetiti da full i diag daju dosta slabe rezultate i zato ćemo koristiti samo tied i spherical.



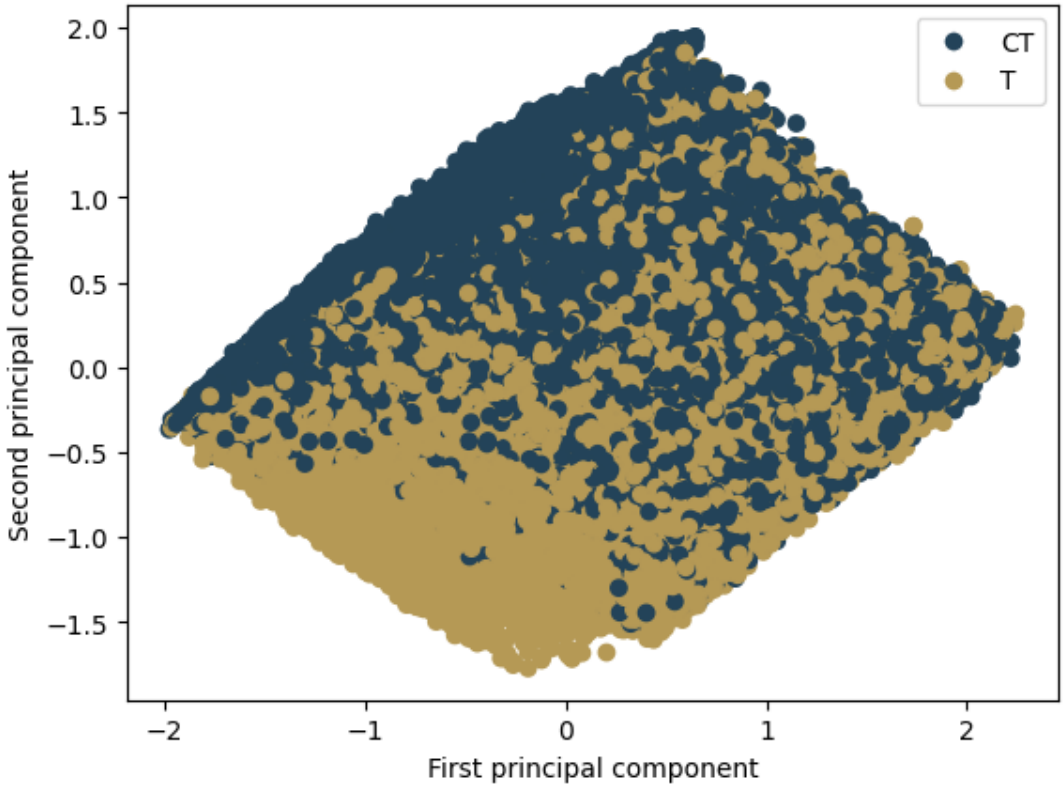
Slika 27: Prikaz klastera dobijenih primenom GMM sa covariance_type='spherical' za neke vrednosti k



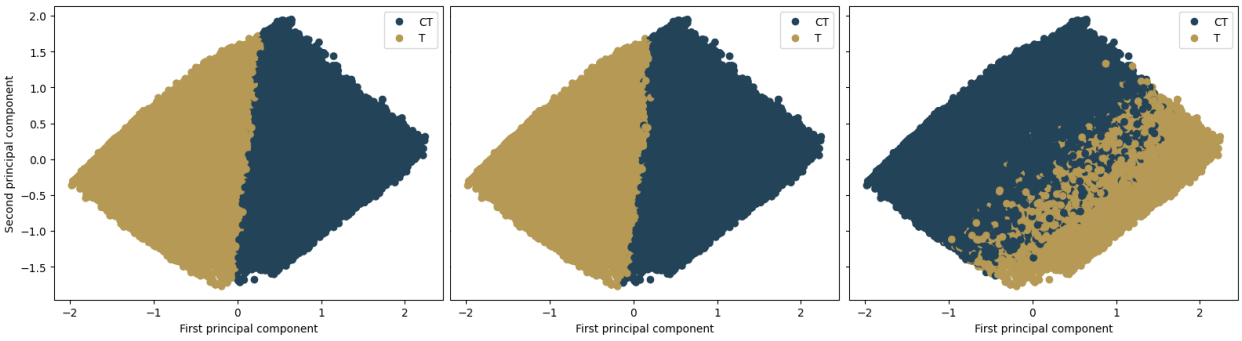
Slika 28: Prikaz klastera dobijenih primenom GMM sa covariance_type='tied' za neke vrednosti k

POREĐENJE MODELA

S obzirom da je naš skup podataka prvenstveno napravljen za klasifikaciju, nama je poznato da postoje dve klase (CT i T). Zbog toga možemo da upoređujemo naše modele sa tačnom podelom i tako da vidimo koliko su oni zapravo (ne)precizni. Na slici 29 vidimo kako su podaci raspoređeni i primećujemo da su dosta isprepletani i da ne postoji nekakvo pravilo koje ih lepo grupiše.



Slika 29: Originalna podela podataka na klase CT i T



Slika 30: Kmeans

Slika 31: GMM (spherical)

Slika 32: GMM (tied)

Odmah na prvi pogled vidimo da nijedan algoritam nije uspeo da pronađe klastere podjednake onima u originalnoj podeli na klase, što je i bilo za očekivati. Naizgled Gaussian Mixture Model se izdvaja od ostalih i čini se da je on najpribližniji pravom skupu podataka. Ukoliko jedan klaster koji nam algoritmi vrate posmatramo kao CT skup, a drugi kao T skup, možemo da napravimo funkciju koja nam pokazuje koliki je procenat poklapanja. Kada to izračunamo dobijemo za K-means da ima preklapanje od 50.48%, GMM sa spherical kovarijansom ima samo 49.38%, a GMM sa tied kovarijansom ima neverovatnih 54.65% . Ovo je u potpunosti suprotno silhouette score-u kod koga je GMM sa tied kovarijansom bio najgori. Postoje razni drugi algoritmi i modeli koji možda uspeju da daju bolje rezultate ali sam limitiram tehnologijom svog vremena (i budžeta) pa ćemo morati da se zadovoljimo ovakvim izuzetnim rezultatima.

PRAVILA PRIDRUŽIVANJA

Ponovo, pošto je skup prvenstveno napravljen za klasifikaciju, ne možemo primeniti pravila pridruživanja bez nekakvih izmena podataka. Izvršićemo dodatnu obradu i filtriranje tako da dobijemo skup nad kojim možemo primeniti neki od algoritama za pravila pridruživanja. Primećujemo da se u igrici kupuju oružja, granate, oklop itd. Možemo izdvojiti samo te podatke i posmatrati da li korišćenje nekog oružje podstiče kupovinu neke granate ili neka slična pravila. Takođe, podelićemo podatke u dva skupa, jedan za CT i jedan za T. Ovime garantujemo da nećemo imati situacije da kupovina oružja na jednoj strani utiče na drugu, i pritom možemo i da uporedimo da li se neka pravila ponavljaju.

APRIORI

Za pronalaženje pravila pridruživanja koristićemo algoritam apriori, a primenićemo ga u SPSS Modeleru. Na slikama 33 i 34 vidimo neka od najboljih pravila koje nam je vratio apriori. CT strana ima znatno više generisanih pravila od T strane, 4127 protiv 1480. Pored toga čini se i da su pravila za CT bolja od ovih na T. Sortiranje smo izvršili po vrednostima za *lift*, koja nam govori koliko puta češće se pojavljuje pravilo nego što bi se ono očekivalo ako su događaji nezavisni.

Sort by: Lift 4127 of 4127

Consequent	Antecedent	Support %	Confidence %	Lift
ct_grenade_incen...	ct_grenade_hegre... ct_weapon_awp ct_weapon_m4a4 ct_grenade_smok... ct_defuse_kits	22,526	89,824	2,061
ct_grenade_incen...	ct_grenade_hegre... ct_weapon_awp ct_weapon_m4a4 ct_grenade_smok... ct_helmets	23,068	89,202	2,046
ct_grenade_incen...	ct_grenade_hegre... ct_weapon_awp ct_weapon_m4a4 ct_grenade_smok... ct_grenade_flash...	23,888	88,985	2,041
ct_grenade_incen...	ct_grenade_hegre... ct_weapon_awp ct_grenade_smok... ct_defuse_kits ct_grenade_flash...	25,393	88,984	2,041
ct_grenade_incen...	ct_grenade_hegre... ct_weapon_awp ct_grenade_smok... ct_defuse_kits ct_helmets	25,003	88,842	2,038

Slika 33: Rezultati apriori algoritma nad podacima za CT stranu

Sort by: Lift 1480 of 1480

Consequent	Antecedent	Support %	Confidence %	Lift
t_grenade_moloto...	t_grenade_hegre... t_weapon_awp t_weapon_ak47 t_grenade_smoke... t_grenade_flashb...	10,72	96,388	1,861
t_grenade_moloto...	t_grenade_hegre... t_weapon_awp t_grenade_smoke... t_grenade_flashb... t_helmets	12,202	95,99	1,853
t_grenade_moloto...	t_grenade_hegre... t_weapon_awp t_grenade_smoke... t_grenade_flashb... t_armor	12,203	95,983	1,853
t_grenade_moloto...	t_grenade_hegre... t_weapon_awp t_grenade_smoke... t_grenade_flashb...	12,22	95,862	1,851
t_grenade_moloto...	t_grenade_hegre... t_weapon_awp t_weapon_ak47 t_grenade_smoke... t_armor	10,93	95,725	1,848

Slika 34: Rezultati apriori algoritma nad podacima za T stranu

ZAKLJUČAK

Iz priloženih rezultata možemo da zaključimo da predviđanje pobednika runde u CS:GO nije ni malo naivan izazov. Postoji veliki broj faktora koji nisu opisani u 97 atributa koje poseduje ovaj skup podataka, a najbitniji od njih je čovek koji igra. U zavisnosti od toga ko sedi za kompjuterom i kontroliše karaktera, rezultati raznih scenarija se mogu drastično razlikovati. Postoje igrači koji su jednostavno bolji od ostalih, zbog boljeg ciljanja, poznavanja i razumevanja igre ili iskustva u stresnim i naizgled nemogućim situacijama. Ako je na početku svake runde lako pogoditi koji tim će da pobedi, gde je tu zabava. Dinamičnost igrice i strategija koja se konstantno menja u toku igranja su samo neki od razloga zašto je ova igrica toliko popularna i 11 godina nakon izlaska. U budućnosti možda bude prilike da se ponovo upustim u izazov predviđanja pobednika runde, ali to će biti u nastavku ove igrice - CS2, možda tada budem imao više sreće.

REFERENCE

1. https://github.com/MATF-istrazivanje-podataka-1/2023_Data_Mining_Go_winner_Dataset
2. https://scikit-learn.org/stable/index.html
3. https://matplotlib.org/stable/plot_types/index.html