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](https://www.kaggle.com/datasets/christianlillelund/csgo-round-winner-classification). 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](https://en.wikipedia.org/wiki/Counter-Strike:_Global_Offensive).

# 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



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.



Sa slike 2 možemo da primetimo da je broj pobeda pođednako 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.



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.



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.



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.



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.



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.





# 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.





## 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)





## 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).





## 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.



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.





# 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).





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.



## 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.



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.



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.





## 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.





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.





# 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>

# 