**UNIVERZITET U SARAJEVU**

**ELEKTROTEHNIČKI FAKULTET SARAJEVO**

# DOMAĆA ZADAĆA 3

# MAŠINSKO UČENJE

**Odsjek: Računarstvo i Informatika**
**Datum: 05.01.2020**

**Studenti:**

- **Mašović Haris, 1689/17993**
- **Muminović Amir, 1661/17744**

**Napomena:**

- ovaj isti dokument je prebacen kao drugi pdf fajl radi IEEE uslova u zadaci, te je ovaj isti ostavljen radi jedostavnijeg formata i citanja

# Izbor dataseta i definisanje problema

Izabrali smo sljedeci dataset: https://www.kaggle.com/karangadiya/fifa19

Fifa 19 je igrica koja simulira igranje fudbala u elektronskom smilu. Prethodni dataset predstavlja podatke u svakom igracu unutar te igrice, te njegove karakteristike u igrici, ukljucujuci overall rating igraca.

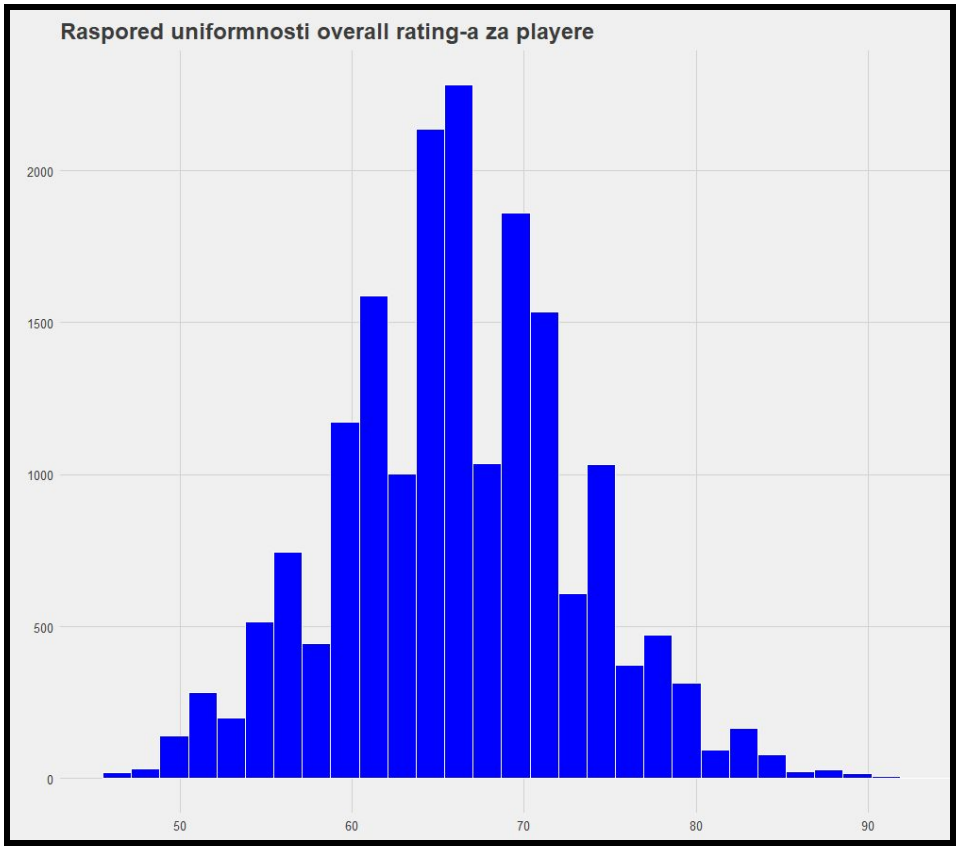
Ova igrica kao jednu od opcija ima tkv. Manager mode, u manager mode-u igrac moze da sebe stavi u ulogu manager-a, te da bira igrace koje zeli da ima u svom timu uz ogranicenu kolicinu para (virtuelne pare unutar igrice). Shodo tome mozemo definisati nas sljedeci problem:

Ukoliko manager zeli nekog igraca u svom timu, a taj igrac mu nije na raspolaganju ili je previse skup za trenutni budzet, cilj mu je naci igrace “slicne” onom kojeg on zeli uz odgovarajucu +/- epsilon vrijednost za njihovu cijenu, za datu poziciju igraca. Shodo tome, kada bi manager morao rucno izanalizirati igrace koji su veoma “slicni” morao bi proci kroz dosta varijabli te dosta igraca, te bi mu trebalo dosta vremena da to odradi. Cilj je napraviti takav clustering (nad pozicijama igraca koje igraju) kada manager bude trazio igrace “slicne” igracem kojeg inicijalno zeli, da dobije zeljenog igraca koji je dovoljno dobar u rangu kriterija managera, za poziciju koju trazi, a taj odgovor moze dobiti na osnovu naseg clusteringa.

# Analiza dataseta po pitanju pogodnosti klasteringa

Izabrani set podataka sadrzi i numericke i kategoricke varijable, te razlog zasto je pogodan za klastering predstavlja to sto svaki igrac ima svoje podatke i atribute koje se koriste u igrici za racunanje njegovog overall ratinga, pored toga svaki igrac ima svoju poziciju/pozicije koju/koje igra (3 moguce pozicije), koliko vrijedi i koja je starosna dob igraca. Set podataka sadrzi 84 varijable, te u ovom radu nece biti prikazana analiza svih ti varijabli, vec mozemo reci da svaka varijabla predstavlja odredjenu karakteristiku za svakog igraca kao sto je npr Aggression, Positioning, Club itd.

Shodno time, mozemo zakljuciti da sam igrac ima dosta karakteristika koje su vezane za njega, a ono sto cini ovaj dataset pogodan za nas clustering jeste da se igraci mogu grupisati u odredjene pozicije, a da pozicija igraca u tim clusterima ce zavisiti od navedenih iznad parametara, a takva raspodjela nam odgovora za nas postavljeni problem koji je opisan u prethodnom poglavlju. Posto je dataset vecinski prvenstveno organizovan bez velikih nedostajucih vrijednosti i nije potrebno dosta preprocessinga, ipak je bilo potrebno za clustering uraditi odredjeni preprocessing. Taj preprocessing moze se naci u poslanom .R fajlu do trenutka kada se plott-a uniformnost overall ratinga za sve igrace. Kao rezultat toga imamo sljedeci graf:




Prethodna slika predstavlja po x-osi overall rating za sve igrace u igrici tj. skinutom dataset-u, dok y-osa predstavlja broj koliko igraca ima takve overall vrijednosti. Mozemo predpostaviti da pocetni dataset nema clustering tendenciju, jer karakteristike igraca su napravljene tako da je overall rating (koji se ujedno i najvise gleda) napravljen tako da bude uniformno rasporedjen (slika iznad) sa svim karakteristikama. To mozemo provjeriti i hopkinsovom statiskom:

```
hopkins(data_cluster, 50)
```

```
> hopkins(data_cluster, 50)
$H
[1] 0.2127662
```

Medjutim, povod za kreiranje clusteringa za ovaj dataset predstavlja grupisanje igraca koje igraju iste pozicije (uradjeno prije poziva algoritama clusteringa sa 4 moguce podjele (GK, DEF, MID, FWD)) i tek onda na osnovu karakteristika igraca kojeg ne mozemo imati, mozemo naci razne druge igrace koji imaju veoma slicne karakterstike onom kojeg mi trazimo.

U nastavku su prikazani algoritmi, objasnjenja tih algoritama, validacije i svrha koristenja tog specificnog algoritma za prethodno opisani problem.

# K-means (Prototip-bazirani) - Masovic Haris

## Opis algoritma

K-means je jednostavan iterativni algoritam koji dati set podataka dijeli u korisnički specificiran broj klastera. Naziv potiče od funkcije distance koja se primjenjuje - mean (srednja vrijednost) i broja klastera koji se formiraju - k. Srednja vrijednost objekata u klasteru se uzima kao centar (centroid) klastera i pripadnost objekta nekom klasteru se procjenjuje na osnovu udaljenosti tog objekta od centroida svih formiranih klastera. Svaka se tačka se dodjeljuje klasteru koji je na osnovu te udaljenosti najbliži.

U izvršenju k-means algoritma kao i ostalih klastering algoritama cilj je da se slični objekti svrstaju u isti klaster, a da se razlika objekata koji nisu u istom klasteru maksimizira. Za mjeru distance objekata se najčešće uzima Euklidova distanca. Pseudo kod za algoritam je tad u nastavku:

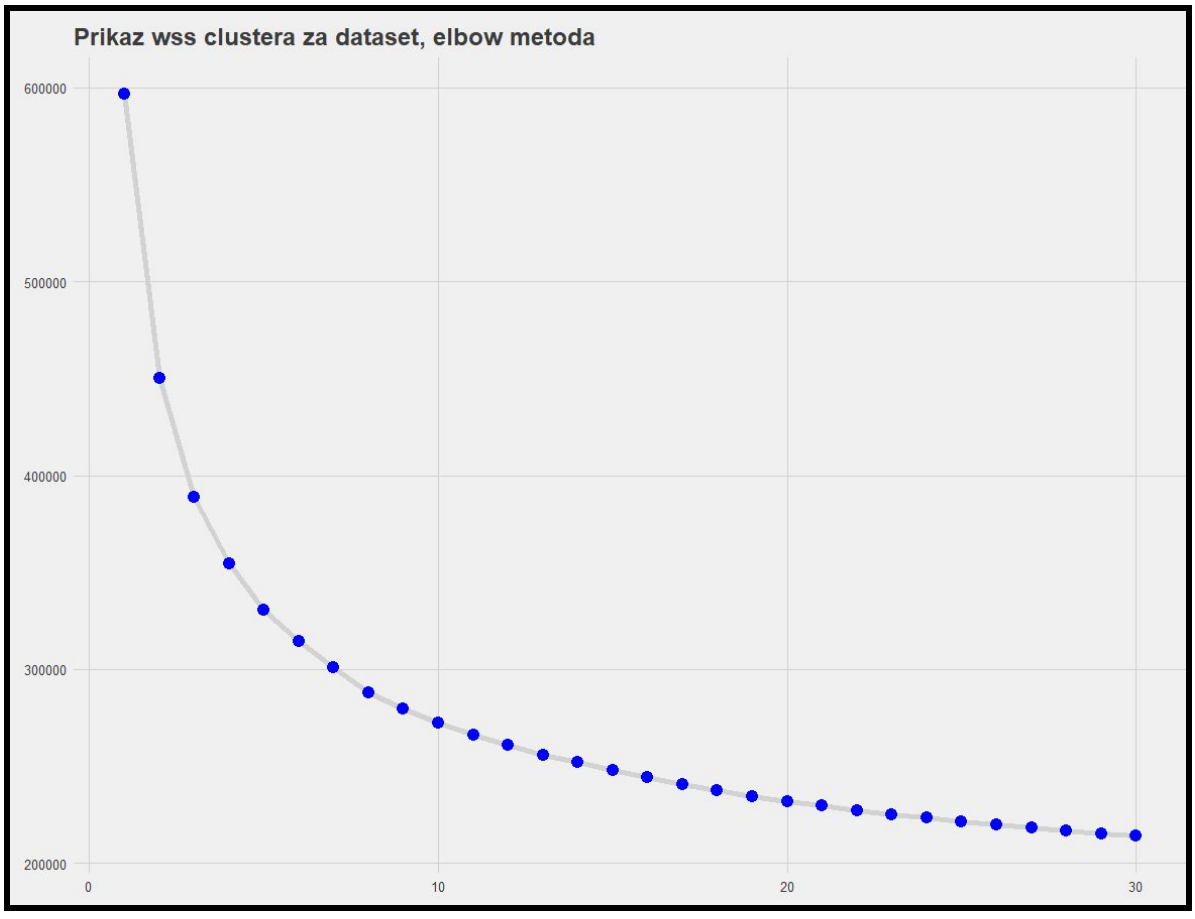
$D$ - n-dimenzionalni prostor u kojem se razmatra problem;
$\boldsymbol{x}$ - tačka u prostoru $D$ data kao $\boldsymbol{x} = \{\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_n\}$ ;
$D_N$ - set podataka koji se analiziraju dat kao $N$ tačaka u prostoru $D$, to je skup tačaka $\boldsymbol{x}_i$ gdje $i = 1,..,N$;
$k$ - broj klastera koji su formirani;
$\boldsymbol{m}$ - centroid klastera dat kao tačka u prostoru $D$;
$C$ - klastering podataka kao niz uređenih parova $\{(C_1,\boldsymbol{m}_1), (C_2,\boldsymbol{m}_2),...,(C_k,\boldsymbol{m}_k)\}$, gdje je $C_j$ identifikator klastera, $\boldsymbol{m}_j$ centroid tog klastera, gdje $j = 1,..,k$;
$D_{Nk}$ - vektor pripadnosti tačaka iz $D_N$ klasterima $C$ dat kao uređeni parovi $D_{Nk} = \{(\boldsymbol{x}_1,C_j),(\boldsymbol{x}_2,C_j),..., (\boldsymbol{x}_N,C_j)\}$

```
Ulaz   : broj klastera k
Izlaz  : klasterizacija - set k klastera predstavljenih vektorom centroida C
 i vektorom pripadnosti tačaka DNk

(1)   izaberi k tačaka iz D kao početne centroide klastera i formiraj vektor C
(2)   repeat
(3)       dodijeli svaki objekat xi iz DN klasteru kojem je najbliži (najsličniji) na osnovu
          udaljenosti-distance objekta od centroida klastera mj i zabilježi to u DNk
(4)       ažuriraj vrijednosti mj (centroid) kao srednju vrijednost tačaka koje u datoj
          iteraciji pripadaju klasteru Cj
(5)       izračunaj vrijednost E (funkcije cilja)
(6)   until nema promjene u vrijednosti E
```

## Implementacija algoritma za definisani problem

Prije nego sto implementiramo k-means algoritam za nas definisan problem, prvo je pametno odrediti broj k, odnosno broj clustera k, koji je optimalan broj za formiranje clustera. Shodno s tim formirat cemo 30 clustera elbow metodom, i nad 30 iteracija zapamtiti wss i prikazati graf za k klustera respektivno od 1 do 30. Shodno dosad opisanim formirati cemo graf na osnovu wss vrijednosti clustera i vidjeti najoptimalniju vrijednost odnosno najbolju vrijednost k tj. koliko clustera je potrebno napraviti za nas dataset/problem. Kod se nalazi na pocetku implementacije algoritma k-means u .R fajlu. Kao rezultat dobijamo:




Najbolje k za nas dataset,  se moze ocitati iz prethodnog grafa, i to predstavlja mjesto gdje se lakat grafa pocinje da “savija”, a to predstavlja oko vrijednosti 8, odnosno za nase optimalno k uzecemo vrijednost k = 8.

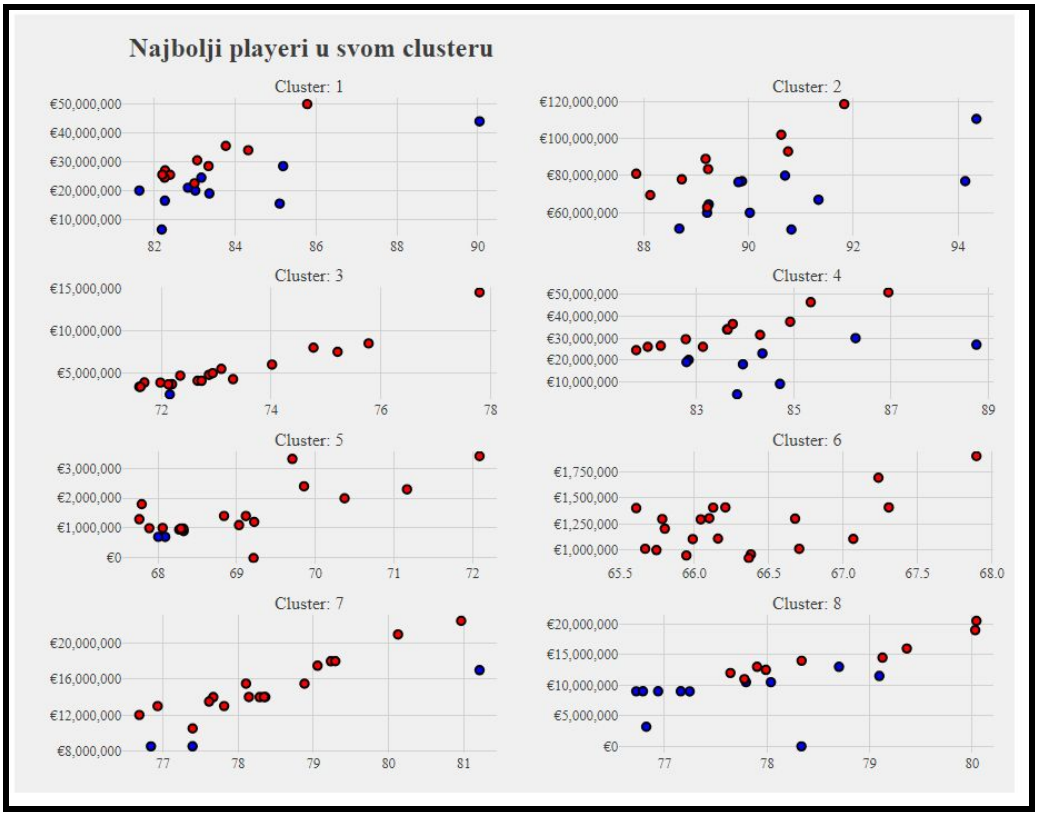
Shodno tome izvrsiti cemo k-means algoritam sa 8 clustera za nas dataset. Nastavak koda u .R fajlu omogucava da podijelimo igrace u clustere i svakom igracu dodijelimo cluster u kojem se

nalazi i formiramo finalni izgled dataseta koji nama treba. Shodno tome mozemo prikazati odnos igraca u clusterima koju dobijemo izvrsavanjem dalje koda:

```
> table(cluster_analysis$Cluster, cluster_analysis$PositionGroup)

     DEF  FWD  MID
1 1130   39 1695
2  139  304  648
3 1487   43 1352
4 1614    3  162
5 1453    6  160
6   24  983 1047
7   18  713 1622
8    1 1327  152
```

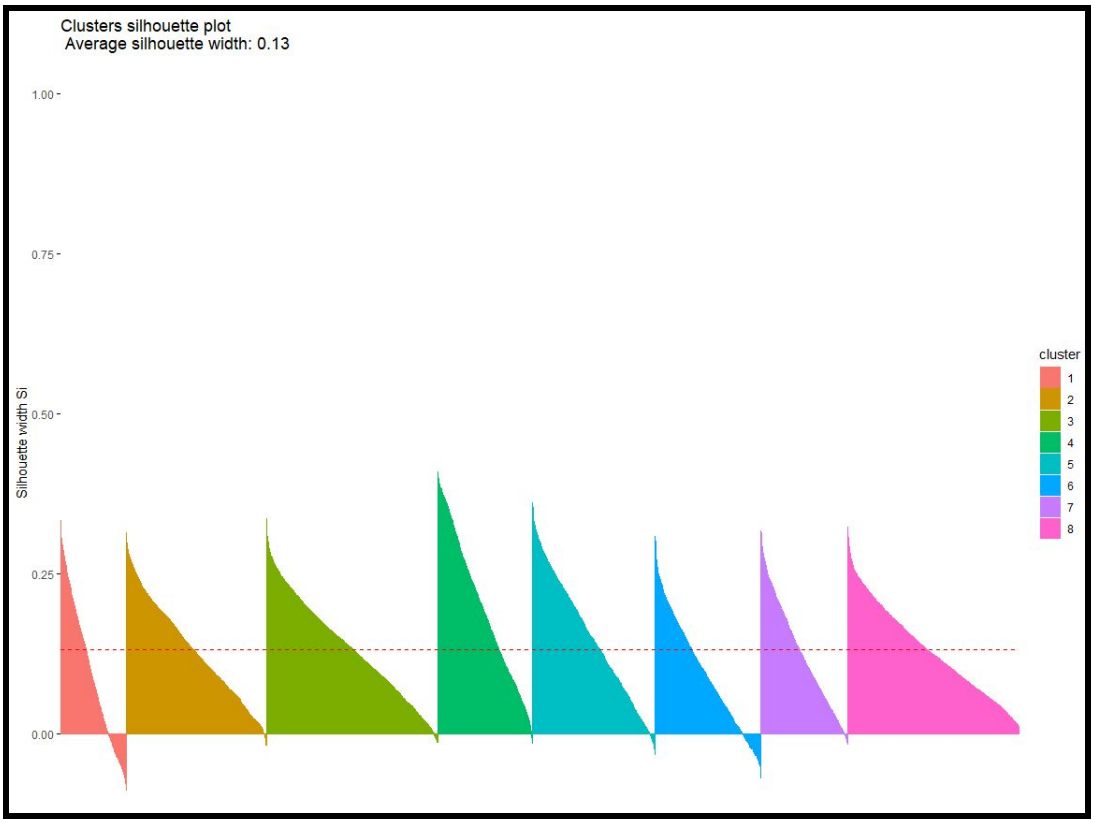
U clusteru 1 imamo prikaz igraca koji su defanzivno orijentisani i ujedno koji su orijentisani ka sredini vise tj. imaju napadackog potencijala iz odbrane, cluster 2 predstavlja igrace koji su orijentisani napadacki, a ujedno igraci koji mogu igrati na sredini terena (mjesavina, mix toga), cluster 3 pokazuje istu skupinu kao cluster 1, ali pod drugim karakteristima u odnosu na cluster 1. Cluster 4, 5 predstavlaju defanzivne igrace, naravno oba clustera po razlicitim karakteristikama. Cluster 6, 7 predstavljaju istu skupinu kao cluster 2, samo razlicite karakteristike i cluster 8 predstavlja suhe igrace koji igraju samo napad. Mozemo predstaviti najboljih 20 u svakom clusteru (kod unutar .R fajla), a rezultat tog plota je dat na sljedecoj slici:



# Validacija clustering-a (interni i eksterni kriteriji validacije)

## Internal validation

Na osnovu prethodnih poglavlja, mozemo zakljuciti da imamo veoma blizu podatke u dataset-u odnosno da nije izrazena prevelika tendencija za clustering. Internu validaciju cemo uraditi pomocu silhouette indexa, odnosno izvrsavanjem koda pod dijelom interne validacije, kao rezultat cemo dobiti sljedeci grafik. Kao rezultat dobijen je sljedeci graf:




Vidimo da su u clusteru 1 i 6 izrazen najveci broj pogresaka kada je u pitanju dodjela igraca odgovarajucem clusteru, te takoder je prikazan relativo mal silhoutte index, sto je bilo za ocekivano na osnovu opisanog dataseta, ali mozemo zakljuciti da je vecina igraca u clusteru u kojem trebaju biti, te odredjena tacnost je zadovoljena.

## External validation

Externu validaciju cemo uraditi pomocu mjera slicnosti Rand indeks. U .R fajlu se nalazi kod pomocu kojeg vrsimo ovu validaciju. Validacija provjerava da li se igrac nalazi u dvije kategorije od moguce tri, jer dosta igraca obicno zna igrati poziciju izmedju sredine i napada i recimo

izmedju odbrane i sredine. Shodno tome gleda se ⅔ u odnosu na postavljeni set (kada bi se gledala striktno pozicija ne bi imalo smisla). Kao rezultat poziva rand index vrijednosti imamo sljedece rezultate:

```
>
> randIndex(cluster_analysis$Cluster, cluster_analysis$positionResult)
      ARI
0.7750665
>
```

## Zakljucak

Na kraju da bi rjesili prvobitni problem, napisana je funkcija vratiSlicne koja kao parametre prima id igraca, max igraca koje zeli da ispise funckija i eps vrijednost za cijenu. Ova funkcija iz clustera u kojoj se nalazi igrac na osnovu kojeg trazimo druge igrace, vraca slicne igrace tom igracu. Na osnovu svih karakteristika igraca, ne samo pozicije uzimaju se igraci koji su po cijeni veoma blizu kao igrac pod kojim vrsimo pretgragu. Primjer pretrage slicnih igraca Seadu Kolasincu je dat u nastavku:

```
C:/Users/User/Desktop/mudz3/
> cluster_analysis[cluster_analysis$ID == 207993,]
       ID         Name    Club Age PositionGroup Overall Cluster ValueNumeric_pounds
536 207993 S. Kolašinac Arsenal  25           DEF      79       1            13000000
> vratiSlicne(207993, 10, 0.05)
      ID         Name                    Club Age PositionGroup Overall Cluster ValueNumeric_pounds
1 181786   V. Corluka        Lokomotiv Moscow  32           DEF      82       1            12500000
2 194229   Hugo Mallo               RC Celta  27           DEF      80       1            13500000
3 186547 M. Musacchio                  Milan  27           DEF      80       1            12500000
4 210455        Jonny Wolverhampton Wanderers  24           DEF      79       1            13500000
5 201143     A. Mandi              Real Betis  26           DEF      79       1            13000000
6 235212    A. Hakimi       Borussia Dortmund  19           DEF      78       1            12500000
7 229984  B. Chilwell          Leicester City  21           DEF      78       1            12500000
8 221342 Pablo Maffeo           VfB Stuttgart  20           DEF      78       1            13000000
>
```

Na kraju iako dataset na pocetku nije pokazao clustering tendenciju, ipak mozemo zakljuciti da ovaj algoritam vraca dobre podatke  na osnovu interne validacije (silhouette index) i na osnovu externe validacije (rand index).

Takodjer, ukoliko analiziramo dobivene igrace za prethodni primjer, mozemo zakljuciti da su ovi napadacki orijentisani defanzivni igraci (kao Sead Kolasinac), te na overall rating igraca je veoma slican njegovom, i njihove karakteristike se mogu porediti, pa manager u ovom slucaju moze pogledati jos 8 opcija ukoliko nije S. Kolasinca moguce dovesti u klub.

Naravno uvijek se moze poboljsati clustering tako sto se skaliraju pocetni podaci dodatno (pored odredjenih skaliranja koja su uradjena), te pokusati napraviti da se clusteri sto bolje razdvoje odnosno da je veca razlika u karakteristikama igraca ili eventualno uvesti vise generalnih pozicija od 3 koje su koristene ovdje (+1 ako racunamo golmane koji su davno izolovani).

# DBSCAN (klastering zasnovan na gustoći) - Muminović Amir

## Opis algoritma

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) je algoritam koji pripada metodama koje pronalaze klastere koristeći informacije o gustoći tačaka u prostoru. DBSCAN je popularan algoritam zbog činjenice da ne zahtjeva predefinisani broj klastera te dobro filtira šum ako je prisutan u podacima.

Algoritam prima dva parametra: parametar distance ε i minimalni broj tačaka neophodnih za formiranje klastera MinPts. ε određuje radijus koji će biti uključen u pretragu susjedstva. Rad algoritma je veoma osjetljiv na izbor ovih parametara. Shodno tome, njihov izbor mora biti pažljivo određen.

U prvom koraku, obavlja se nasumični odabir tačaka. Za odabrane tačke pronalazimo susjedstvo, čija je širina određena parametrom ε. Obavlja se provjera da li se u području nalazi više tačaka od parametra algoritma MinPts. Granične i tačke jezgre se uključuju u klaster dok se nepodobne tačke odbacuju. Postupak se ponavlja sve tačke nisu dio klastera ili su odbačene (prikazane kao nepodobnosti). Pseudokod za algoritam je tad u nastavku:

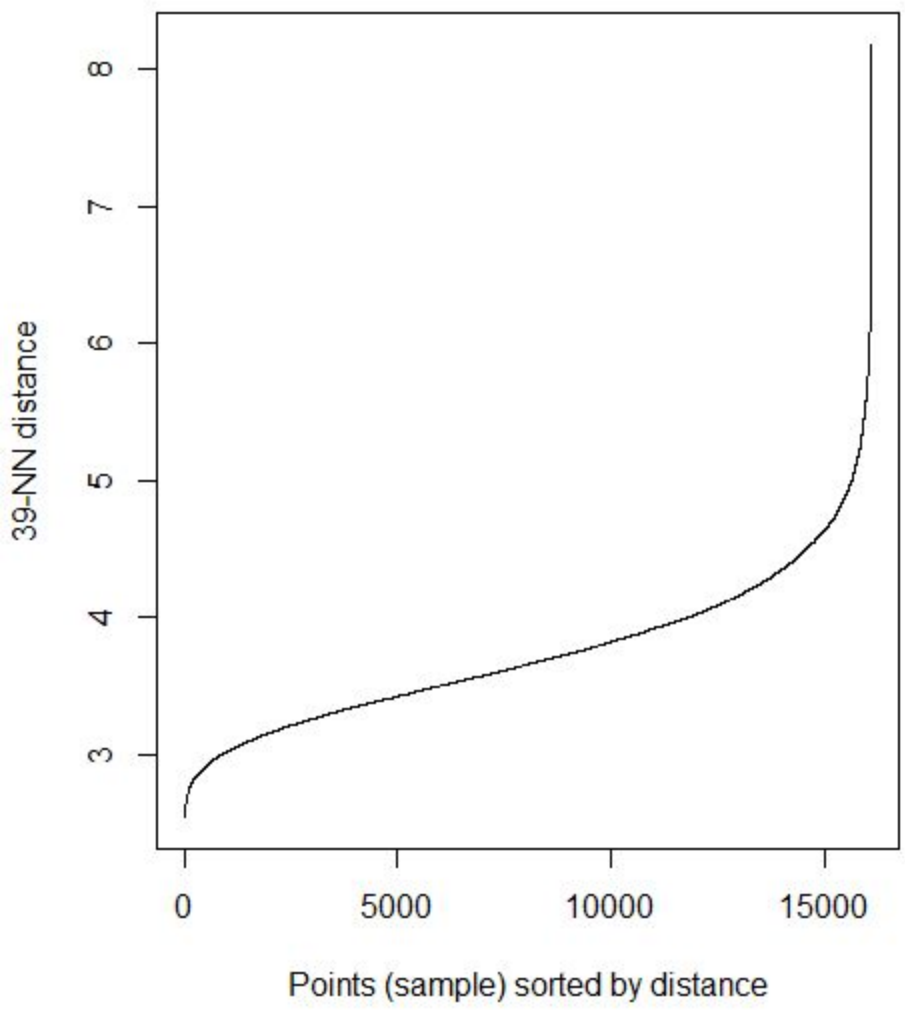
```
Ulaz: Ɛ radijus okoline objekta, MinPts - minimalan broj objekata u
        Ɛ -okolini tačke da bi ona bila jezgro
Izlaz: klasterizacija - set k klastera

(1)  tačku p iz D_N na osnovu MinPts i Ɛ označi kao jezgro, graničnu tačku ili nepodobnost
(2)  eliminiraj iz D_N sve tačke označene kao nepodobnosti
(3)  poveži granom sva jezgra koja su jedna drugim u Ɛ -okolini
(4)  svaku grupu povezanih jezgri označi kao zaseban klaster
(5)  svaku graničnu tačku svrstaj u klaster kojem pripada njena tačka jezgre

Metoda za označavanje tačaka (potreban u koraku 1):
(1)  repeat
(2)      uzmi tačku p iz D_N koja nije obilježena kao analizirana
(3)      if broj tačaka u Ɛ -okolini od p >= MinPts then
(4)          označi p kao jezgro
(5)          repeat
(6)              uzmi sljedeću tačku q iz Ɛ -okoline od p
(7)              if q nije označeno kao jezgro then
(8)                  označi q kao graničnu tačku
(9)          until sve tačke iz Ɛ -okoline nisu provjerene
(10)     else
(11)         označi p kao nepodobnost (šum, noise)
(12)     označi da je p analizirana
(13) until sve tačke iz D_N nisu analizirane
```

## Implementacija algoritma za definisani problem

Implementacija algoritma se sastoji od tri funkcije. Prva funkcija, nadjiSusjede, pronalazi sve tačke iz skupa podataka D koji su udaljeni manje od vrijednosti epsilona, u poređenju sa zadanom tačkom. Kao mjera distance koristi se Euklidova distanca.
Druga funkcija, povecajCluster se poziva u situaciji kada obavljamo ekspanziju postojećeg klastera. Nakon identifikacije jednog regiona kao klastera, poziva se povećajKlaster nad svim elementima unutar tog klastera. Ako te tačke označene kao nepodobne, dodjeljuje im se klasa klastera. Ako nisu obrađene, dobijaju klasu C ali također pronalazimo susjedstvo novih tačaka i uključujemo ga u pretragu.
Funkcija dbsearch kreira listu labela za svaki element u skupu podataka te u svakoj iteraciji provjerava po jednu tačku iz skupa podataka. Pronalazi njeno susjedstvo te broji broj tačaka u susjedstvu. Ako je manji od minPts, označava tačku kao nepodobnu. Ako je tačka pogodna, poziva se povećajCluser nad novom tačkom. Na kraju vraća listu labela dodijeljenu svakom elementu skupa.
Kao što je navedeno u prošlom dijelu, kvalitet dobijenih rezultata znatno ovisi od vrijednosti odabranih za parametre ε i MinPts. Optimalna vrijednost ε se može naći skiciranjem kNNdist grafika te identifikacijom tačke naglog porasta na tom grafiku. Pozivanjem funkcije kNNdistplot iz biblioteke dbscan, dobiju se sljedeći rezultati:




Grafik n: Grafik dobijen nakon primjene kNNdistplot funkcije, pri čemu se za optimalno ε može uzeti 4

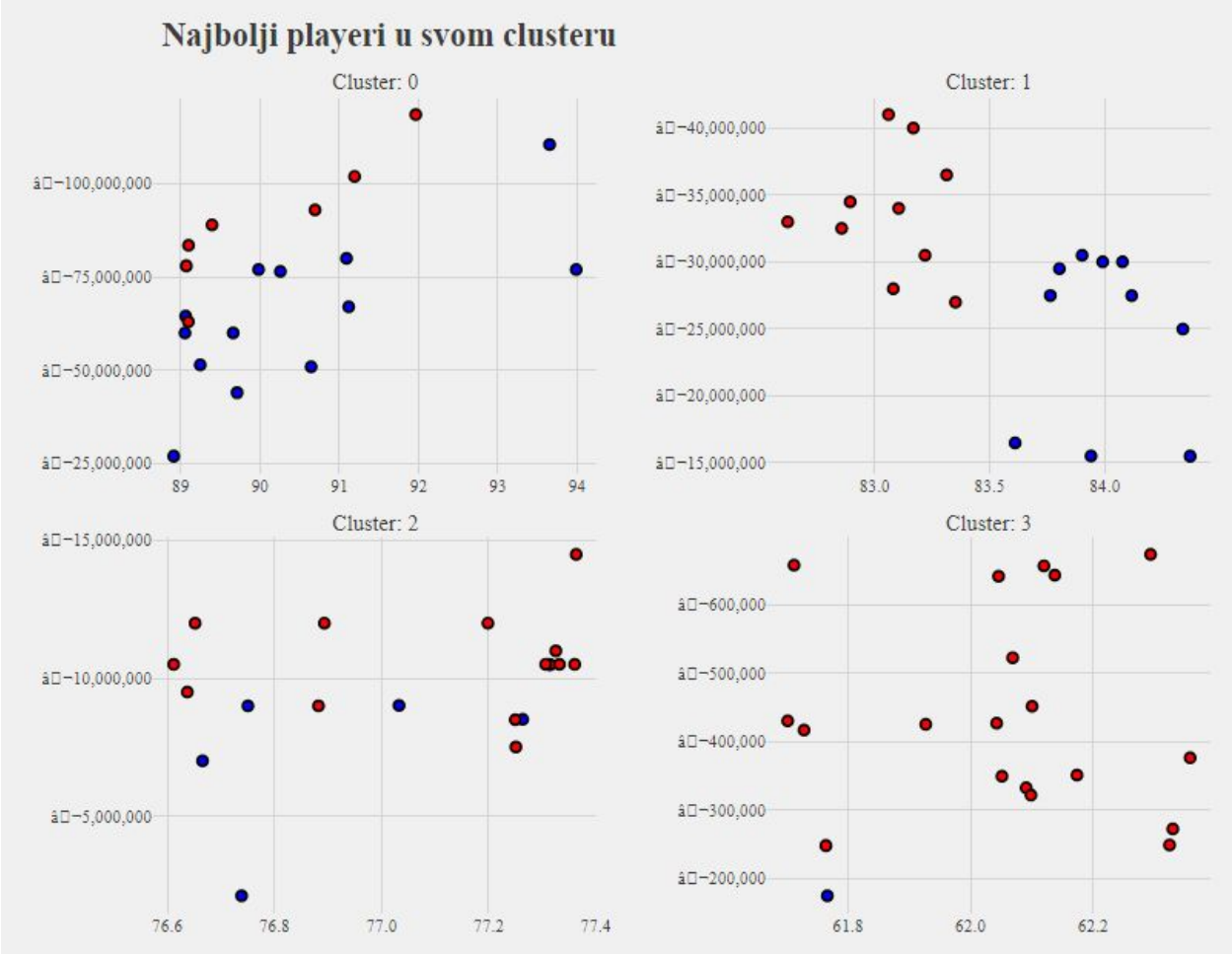
Na osnovu dobijenog grafika možemo zaključiti da je tačka najbržeg rasta ε = 4. Izbor parametra MinPtr se najčešće obavlja eksperimentiranjem ili korištenjem domenskog znanja. Razni testovi su pokazali da se dobiju dobri rezultati sa vrijednosti 150. Primjenom algoritma s tim vrijednostima dobiju se sljedeći rezultati:

```
   0    1    2    3
2390  664 9209 3859
```

2390 tačaka je označeno kao nepodobne. Ostale tačke su podijeljene u tri klastera. Ako obavimo podjelu po pozicijama koje igrači zauzimaju, dobijemo sljedeću tabelu:

| | DEF | FWD | MID |
|---|---|---|---|
| 0 | 1841 | 334 | 215 |
| 1 | 146 | 157 | 361 |
| 2 | 2673 | 2028 | 4508 |
| 3 | 1206 | 899 | 1754 |

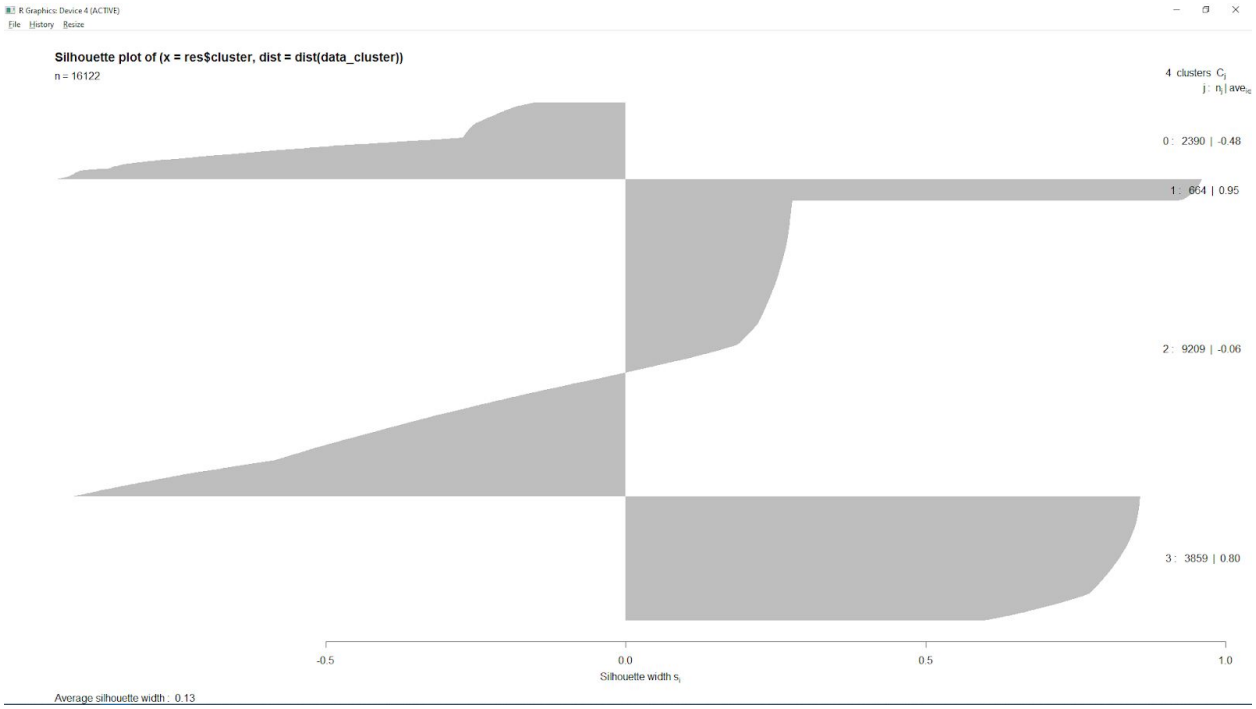
Možemo primjetiti da se veliki broj defanzivnih igrača nije dodjelilo klasteru. Gledajući dobijene rezultate vidimo također da klastering nije obavljen po poziciji koju igrač igra, već po drugim parametrima. Za bolji rezultat, neophodno je uključiti samo atribute u klastering koji su blisko vezani sa pozicijom koju igrač igra. Analogno kao u opisu prvog algoritma, možemo predstaviti top 20 igrača unutar novodobijenih klastera:



# Validacija clustering-a (interni i eksterni kriteriji validacije)

## Internal validation

Možemo obaviti internu validaciju dobijenih rezultata koristeći funkciju silluete za pronalazak silluete indexa. Nakon poziva funkcije, možemo skicirati rezultate i dobiti sljedeći graf.




Možemo vidjeti da prvi klaster (oznaka 1) ima veliku širinu u jednom regionu iako ima poprilično dobre rezultate u drugim dijelovima. Klaseri 2 i 3 također imaju poprilično veliku širinu siluete što je znak da se klastering model treba popraviti.

## External validation

Eksterna validacija se obavlja analogno kao u prvom algoritmu. Nakon ponavljanja iste procedure dobije se sljedeći rezultat:

```
> randIndex(cluster_analysis$Cluster, cluster_analysis$positionResult)
ARI
  1
```

Dobijena je vrijednost 1 za ARI što znači da je klastering dobro urađen i da svi elementi pripadaju tačno jednoom klasteru.

# Zaključak

Algoritam je pokazao poprilično dobar u kreiranju deterministčkih klastera, što je pokazala eksterna validacija. Interna validacija je pokazala da ima poprilično široke siluete što je znak da treba napraviti drugačiji odabir polaznih parametara eps i MinPtr. Dodatna indikacija da odabir parametara nije optimalan je činjenica da je skoro 20% dataseta klasificirano sa nepodobnim tačkama.

Bitno je naglasiti da kreirani algoritam ima veoma dugo vrijeme izvršavanja (prosječno 90 minuta) za trenutni set podataka (~16000 elemenata). Za efikasniju upotrebnu neophodno je obaviti brojne optimizacije kako bi se rezultati mogli dobivati brže.

Rezultati dobiveni klasterizacijom sa DBSCAN se mogu poboljšati ako bi se obavila selekcija atributa koji imaju veći uticaj na atribut pozicije igrača.