Klasterovanje Olimpijskih igara

Seminarski rad u okviru kursa
Istraživanje podataka 1
Matematički fakultet

Mladen Canović
mladen.canovic@gmail.com

3. septembar 2019

Sažetak

Olimpijske igre predstavljaju najveći sportski događaj na svetu, koji okuplja sportiste, koji se nadmeću u raznim disciplinama, prethodne 123 godine. Sa druge strane, razvoj tehnologije doveo je do naglog razvoja jedne neolimpijske discipline - istraživanja podataka. Ovaj rad ima za cilj da analizira, obradi i iznese određene zaključke o podacima sa Olimpijskih igara korišćenjem metoda istraživanja podataka. Tehnikama klasterovanja pokušava da odgovori na pitanja o razvoju Olimpijskih igara, broju učesnika, osvajačima medalja, šansama učesnika na osnovu određenih parametara i mnoga druga.

Sadržaj

1 Uvod

Amerikanac Džejms Konoli je pobedom u olimpijskoj disciplini troskok, 6. aprila 1896. postao prvi olimpijski šampion nakon više od 1500 godina. Tačnije, od Olimpijskih igara 393. godine nove ere, nakon čega je rimski car Teodosije I zabranio održavanje Olimpijskih igara. Ponovno pokretanje Olimpijskih igara bila je ideja Barona Pjera de Kubertena. Prve moderne Olimpijske igre održane su u Atini, 1896. godine i na njima je učestvovalo 245 muških takmičara iz 14 država [2, 1]. Od 1896. godine do danas, održano je 23 zimskih i 28 letnjih olimpijskih igara u 43 različita grada.

U poglavlju 2 će ukratko biti pobrojan skup tehnologija i alata, koji će biti korišćen u okviru ovog seminarskog rada. U poglavlju 3 biće akcenat na podacima, biće izvršena analiza i pretprocesiranje. Takođe, biće izvršena vizuelizacija podataka i delimično ispitivanje međusobne zavisnosti atributa. U trenucima kada je, po mišljenju autora rada, neophodno dati kratak istorijski osvrt, to će biti učinjeno. U poglavlju 4 će biti izvršeno klasterovanje podataka i analiza dobijenih rezultata. U poglavlju 5 će biti dat osvrt na rad i predloženi dalji koraci u istraživanju.

2 Skup tehnologija i alata

Širok je spektar programskih jezika i programskih alata, koji olakšava analizu i obradu podataka kao i primenjivanje metoda istraživanja podataka nad njima. Skup tehnologija i alata koji će biti korišćen u ovom seminarskom radu je sledeći:

- Programski jezik *Python*;
- Biblioteka *pandas* za manipulaciju podacima;
- Biblioteka *numpy* za manipulaciju matematičkim strukturama;
- Biblioteka *sklearn*, koja olakšava korišćenje raznih metoda istraživanja podataka;
- Biblioteka *matplotlib* za vizuelizaciju podataka i odnosa među njima.

3 Analiza i pretprocesiranje podataka

Osnovu svakog projekta koji se bavi istraživanjem podataka čini dobar početni skup podataka. Podatke je potrebno analizirati, izvršiti pretprocesiranje i, na kraju, koristiti ih za pronalaženje određenih zakonitosti i donošenje konkretnih zaključaka [4].

Podaci, koji će biti korišćeni u nastavku rada sadrže informacije o učesnicima olimpijskih igara počev od 1896. godine i o nacionalnim olimpijskim komitetima (NOK) (eng. *National Olympic Committee (NOC)*), koje su ti učesnici predstavljali[1].

3.1 Analiza podataka

Podaci su razdvojeni u dve *CSV* (*Comma-separated values*) datoteke. U jednoj datoteci se nalaze podaci o prethodnim i trenutnim imenima nacionalnih komiteta. Svaki red ove tabele čine 3 kolone - skraćenica

[1] Podaci su preuzeti sa sledećeg linka: https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results.

komiteta, ime države (regiona) koju zastupa komitet, dodatna informacija ukoliko je komitet sa istom skraćenicom na različitim olimpijskim igrama predstavljao državu (region) različitog imena zbog izvesnih geopolitičkih promena. Prikazaćemo nekoliko redova, koji će pomoći da se stekne jasna slika o podacima iz ovog skupa podataka, u listingu 1.

```
noc_regions = pd.read_csv('../data/noc_regions.csv')
print(noc_regions.iloc[[26, 95, 147, 168, 227]])

Out>>
     NOC             region                     notes
26   BOH     Czech Republic                   Bohemia
95   IRL            Ireland                       NaN
147  NFL             Canada              Newfoundland
168  ROT                NaN      Refugee Olympic Team
227  YUG             Serbia                 Yugoslavia
```

Listing 1: Podaci iz skupa podataka o regionima

Proverićemo da li, osim olimpijskog tima izbeglica, postoji još redova sa nedostajućim vrednostima atributa *region* u listingu 2. U nastavku rada ćemo, prilikom spajanja sa drugom datotekom u listingu 4, razrešiti sve nedostajuće vrednosti.

```
noc_regions = pd.read_csv('../data/noc_regions.csv')
print("Redovi sa nedostajucim vrednostima za atribut '
    region':")
print(noc_regions[noc_regions['region'].isna()])

Out>>
Redovi sa nedostajucim vrednostima za atribut 'region':
     NOC region                         notes
93   IOA    NaN  Individual Olympic Athletes
168  ROT    NaN         Refugee Olympic Team
208  TUV    NaN                       Tuvalu
213  UNK    NaN                      Unknown
```

Listing 2: Pronalaženje nedostajućih vrednosti za atribut *region*

Što se tiče trećeg reda iz rezultata, došlo je do greške i potrebno je upisati ime države Tuvalu u polje *region*. Na osnovu poslednjeg reda iz rezultata vidimo da za neke učesnike nije poznato odakle dolaze. Kod preostala dva reda, u pitanju su timovi koji su formirani kako bi se omogućio nastup učesnicima, koji to nisu mogli da učine pod zastavom postojećeg olimpijskog komiteta.

Timu nezavisnih učesnika (eng. *Individual Olympic Athletes*) su pripadali učesnici čiji je nacionalni olimpijski komitet bio suspendovan iz političkih razloga. Ovi timovi su nastupali pod zastavom olimpijade. Prvi put je formiran ovaj tim 1992. godine i činili su ga takmičari iz Savezne Republike Jugoslavije i Severne Makedonije[2]. Olimpijski tim izbeglica (eng. *Refugee Olympic Team*) je formiran 2016. kada su pojedini takmičari tokom velike izbegličke krize ostali bez mogućnosti da se takmiče pod zastavama svojih olimpijskih komiteta. Ovo je učinjeno i da bi se podigla svest o ovom velikom problemu današnjice.

Zarad lakše analize, vrednosti iz kolone *notes* za ova 3 reda ćemo ubaciti u polje *region*. U poslednjoj koloni ćemo vrednost postaviti na string *None* na mesto svih nedostajućih vrednosti. Razlog za veliki broj

[2]Na ovom takmičenju je tim nezavisnih učesnika osvojio 3 medalje u streljaštvu. Jasna Šekarić je osvojila srebrnu medalju, a Aranka Binder i Stevan Pletikosić bronzanu.

nedostajućih vrednosti u poslednjoj koloni je to što je većina komiteta, od svog osnivanja, predstavljala entitet, koji se ni na koji način nije menjao.

Druga tabela predstavlja učesnike olimpijskih igara i njihove nastupe na olimpijskim igrama. Sadrži 15 atributa (kolona) i 271.116 redova (listing 5). Svaki red predstavlja učešće jednog sportiste na jednom događaju i njegov uspeh, tj. odličje koje je osvojio, ukoliko je osvojio. Jedna olimpijska disciplina može imati različite događaje, koji se razlikuju po određenim parametrima (dužina staze kod trčanja, visina skakaonice kod skokova u vodu, itd). Učesnik koji svoju državu predstavlja na više događaja, na istoj olimpijadi, nalazi se u više redova, od kojih svaki red odgovara jednom događaju. Atribute ćemo izlistati u listingu 3, a zatim prikazati značenje svakog atributa u tabeli 1.

```
print(data.columns.tolist())

Out>> ['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight',
       'Team', 'NOC', 'Games', 'Year', 'Season', 'City',
       'Sport', 'Event', 'Medal']
```

Listing 3: Ispis atributa

Ime atributa	**Značenje**
ID	ID učesnika
Name	Ime i prezime učesnika
Sex	Pol učesnika
Age	Starost učesnika
Height	Visina učesnika
Weight	Težina učesnika
Team	Tim učesnika [3]
NOC	NOK pod čijom zastavom učesnik nastupa
Games	Godina i sezona olimpijade (npr: *1984 Winter*)
Year	Godina održavanja olimpijade
Season	Sezona održavanja olimpijade (letnja/zimska)
City	Grad u kome se održala olimpijada
Sport	Naziv sporta
Event	Događaj (npr. *Trčanje 200m sprint*)
Medal	Osvojena medalja (*NA* - bez medalje)

Tabela 1: Nazivi atributa i njihovo značenje

Implementiraćemo funkciju u listingu 4, koju ćemo pozivati za učitavanje podataka. Ovo će nam olakšati učitavanje podataka tokom analize i testiranja, jer će se ista procedura učitavanja podataka, sa različitim parametrima, koristiti više puta.

```
def get_data(fields=None,
             noc=None,
             merge_noc=True,
             printable=False):

    if fields is None:
        # ukoliko ne izaberemo atribute, podrazumevano
        # ce izvuci kompletne redove
```

[3] Jedan NOK može kod nekih sportova imati više timova na jednom događaju

```
        athlete_events =
            pd.read_csv('../data/athlete_events.csv',
                        error_bad_lines=False)
    else:
        if noc is not None:
            # ukoliko je izabran NOK po kome zelimo
            # filtrirati potrebno je dodati i taj
            # atribut u spisak atributa
            fields.append('NOC')

        athlete_events =
            pd.read_csv('../data/athlete_events.csv',
                        error_bad_lines=False,
                        usecols=fields)

    df = pd.DataFrame(athlete_events)

    if noc is not None:
        nationality_filter = athlete_events['NOC'] == noc
        df = df[nationality_filter]

    if merge_noc is True:
        noc_regions =
            pd.read_csv('../data/noc_regions.csv')
        df = pd.merge(df, noc_regions,
                      on='NOC', how='left')

        df['region'].fillna(noc_regions['notes'],
                            inplace=True)
        df['notes'].fillna('None', inplace=True)
        df['region'].fillna('None', inplace=True)

    # ispisivanje dobijenih podataka ukoliko je trazeno
    if printable is True:
        pd.set_option('display.max_rows', None)
        pd.set_option('expand_frame_repr', False)
        print(df)

    return df
```

Listing 4: Funkcija za učitavanje podataka

Primetimo da smo u funkciji izvršili spajanje ove dve tabele po atributu koji označava nacionalni olimpijski komitet, kako bismo redu pridružili atribut, koji označava naziv države (regiona, organizacije) za koju učesnik nastupa. Učitaćemo podatke korišćenjem implementirane funkcije, a zatim broj atributa i redova novoformirane tabele izračunati u listingu 5.

```
data = get_data(printable=False)
print(data.shape)

Out>> (271116, 17)
```

Listing 5: Učitavanje podataka i ispis dimenzije

U listingu 6 prikažimo broj takmičara za svaki od olimpijskih komiteta, koji su na početku imali nedostajuće vrednosti u polju *region*.

```
print("IOA count: ", data[data['NOC'] == 'IOA'].shape[0])
print("ROT count: ", data[data['NOC'] == 'ROT'].shape[0])
print("TUV count: ", data[data['NOC'] == 'TUV'].shape[0])
print("UNK count: ", data[data['NOC'] == 'UNK'].shape[0])
```

```
Out>>
IOA count:  76
ROT count:  10
TUV count:  6
UNK count:  2
```

Listing 6: Broj takmičara za komitete sa početnim nedostajućim vrednostima polja *region*

Pogledajmo osnovne statističke informacije učitanih podataka, koristeći funkciju *describe* iz biblioteke *pandas*. Podatke ćemo prikazati u listingu 7 uz zaokruživanje na 2 decimale.

```
print(data.describe().round(2))

Out>>
                ID         Age      Height      Weight        Year
count   271116.00  261642.00  210945.00  208241.00  271116.00
mean     68248.95      25.56     175.34      70.70    1978.38
std      39022.29       6.39      10.52      14.35      29.88
min          1.00      10.00     127.00      25.00    1896.00
25%      34643.00      21.00     168.00      60.00    1960.00
50%      68205.00      24.00     175.00      70.00    1988.00
75%     102097.25      28.00     183.00      79.00    2002.00
max     135571.00      97.00     226.00     214.00    2016.00
```

Listing 7: Statističke informacije učitanih podataka

Dakle, prosek godina je 25.56, prosečna visina 175.34m, a težina 70.70kg itd. Možemo primetiti dve zanimljive ekstremne vrednosti - najstariji učesnik u istoriji olimpijskih igara imao je 97 godina, a najmlađi učesnik 10. Takođe, iz ovih podataka možemo zaključiti koliko je redova sa nedostajućim vrednostima po svakoj koloni od navedenih. Na primer, nedostaje nam informacija o starosti 9474 (271116 − 261642) učesnika. Broj redova iz cele tabele, koji sadrže nedostajuće vrednosti u bilo kojoj koloni, dobijamo u listingu 8.

```
null_rows = data[data.isna().any(axis=1)]
print(null_rows.shape[0])

Out>> 240935
```

Listing 8: Broj redova sa nedostajućim vrednostima

Nakon što smo pronašli broj redova sa nedostajućim vrednostima potrebno je utvrditi u kojim kolonama se nedostajuće vrednosti javljaju:

```
print(data.columns[data.isna().any()].tolist())

Out>> ['Age', 'Height', 'Weight', 'Medal']
```

Listing 9: Kolone koje sadrže nedostajuće vrednosti

Odokativnom analizom možemo doći do zaključka da najveći broj nedostajućih vrednosti dolazi iz činjenice da je vrednost u koloni koja predstavlja osvojenu medalju, učesnicima koji nisu osvojili nijedno odličje na događaju, označena sa *NA*. U polje koje označava osvojenu medalju, učesnicima koji nisu osvojili nijedno odličje na navedenom događaju, izmenićemo u string *None* (listing 10).

```
data['Medal'].fillna('None', inplace=True)
```

Listing 10: Ažuriranje polja koja označavaju osvojenu medalju

Kao što smo primetili, najstariji učesnik olimpijade imao je 97 godina. Proverimo ispravnost ovog podatka, pritom smanjujući starosnu granicu na 85 godina (listing 11).

```
age_filter = data['Age'] > 85
data = data[age_filter]
print(data[['Name', 'Age', 'Games', 'Sport']])

Out>>
                                Name   Age        Games              Sport
60861    Thomas Cowperthwait Eakins  88.0  1932 Summer  Art Competitions
60862    Thomas Cowperthwait Eakins  88.0  1932 Summer  Art Competitions
60863    Thomas Cowperthwait Eakins  88.0  1932 Summer  Art Competitions
98118                 Winslow Homer  96.0  1932 Summer  Art Competitions
257054      John Quincy Adams Ward  97.0  1928 Summer  Art Competitions
```

Listing 11: Učesnici stariji od 85 godina

Svi učesnici stariji od 85 godina su učestvovali na takmičenjima iz oblasti umetnosti[4]. Možemo pretpostaviti da su i u opštem slučaju učesnici koji su se takmičili u ovoj disciplini, među najstarijim učesnicima. Dodatnim smanjivanjem granice i analizom dobijenih podataka, potvrđujemo ovu pretpostavku. Discipline koje su imale najstarije učesnike, pored umetnosti, su streljaštvo, streličarstvo, jahanje, jedrenje (ovo će biti pokazano u radu). Broj učesnika iz oblasti umetnosti ćemo izračunati i prikazati u listingu 12.

```
sport_filter = data['Sport'] == 'Art Competitions'
data = data[sport_filter].drop_duplicates(
                subset=['ID'], keep='first')
num_of_artists = data.shape
print("Number of art competitors: ", num_of_artists[0])

Out>> Number of art competitors:  1814
```

Listing 12: Broj učesnika u umetničkim takmičenjima

Sličnim postupkom možemo pronaći i podatke o najmlađem učesniku u istoriji olimpijskih igara. U pitanju je Dimitrios Loundras, koji je kao najmlađi učesnik u istoriji došao i do bronzane medalje (listing 13).

```
age_filter = data['Age'] == 10
data = data[age_filter]
print(data[['Name', 'Age', 'Games', 'Sport', 'Medal']])

Out>>
                     Name  Age        Games      Sport   Medal
142882 Dimitrios Loundras 10.0 1896 Summer Gymnastics  Bronze
```

Listing 13: Najmlađi učesnik u istoriji

Kao što smo videli pomoću funkcije iz listinga 8, u početnom skupu podataka ima 240.935 redova sa nedostajućim vrednostima. Ove vrednosti ne smemo odbaciti, već ih je neophodno ispitati i obraditi na odgovarajući način. Nakon što smo izmenili vrednosti atributa *Medal* sa *NA* u string *None*, preostao je ukupno 64.951 red sa nedostajućim vrednostima, što možemo proveriti korišćenjem istog postupka iz listinga 8. Redove kod kojih atribut *Age* ima nedostajuće vrednosti, to polje ćemo zameniti srednjom vrednosti starosti svih takmičara za koje imamo ovu informaciju.

[4]Iako je malo poznato, takmičenja iz oblasti umetnosti su bila deo olimpijskih igara od 1912. do 1948. godine. U takmičenje su ulazila dela inspirisana sportom, a grane umetnosti koje su bile zastupljene bile su arhitektura, slikarstvo, književnost, vajarstvo i muzika. Ova olimpijska disciplina je ukinuta, jer je smatrano da su svi umetnici koji su se prijavljivali, profesionalci i da ne bi trebalo da učestvuju na olimpijadi. Osnovna ideja olimpijskih igara je bila da se nadmeću amateri. Sredinom dvadesetog veka, modernizacija društva i samog sporta, kao i političke promene, uticale su na promenu stava međunarodnog olimpijskog komiteta (eng. *International Olympic Committee (IOC)*) i počelo se sa omogućavanjem učešća profesionalcima [3, 2]

Za atribute *Height* i *Weight* ćemo koristiti modu svih redova za svaki od atributa. Postupak i rezultat njegovog izvršavanja dat je u listingu 14. Možemo primetiti da smo se uspešno rešili svih nedostajućih vrednosti iz polaznog skupa podataka.

```
def fill_na_attr_rows(data):
    data['Medal'].fillna('None', inplace=True)

    age_mean = int(data['Age'].mean())
    data['Age'] = data['Age'].fillna(age_mean)
    data['Age'].isna().sum()

    height_mode = data['Height'].mode()[0]
    data['Height'] = data['Height'].fillna(height_mode)
    data['Height'].isna().sum()

    weight_mode = data['Weight'].mode()[0]
    data['Weight'] = data['Weight'].fillna(weight_mode)
    data['Weight'].isna().sum()

    return data


data = fill_na_attr_rows(data)
# proveravamo da li postoji jos redova sa nedostajucim
# vrednostima
print(data[data.isna().any(axis=1)].shape[0])

Out>> 0
```

Listing 14: Izmena nedostajućih vrednosti srednjim

Nakon što smo otklonili sve nedostajuće vrednosti iz početnog skupa podataka, novi skup podataka ćemo sačuvati u novu datoteku. Postupak je prikazan u listingu 15.

```
data.to_csv(r'../data/full.csv')
```

Listing 15: Čuvanje ažuriranih podataka u novi dokument

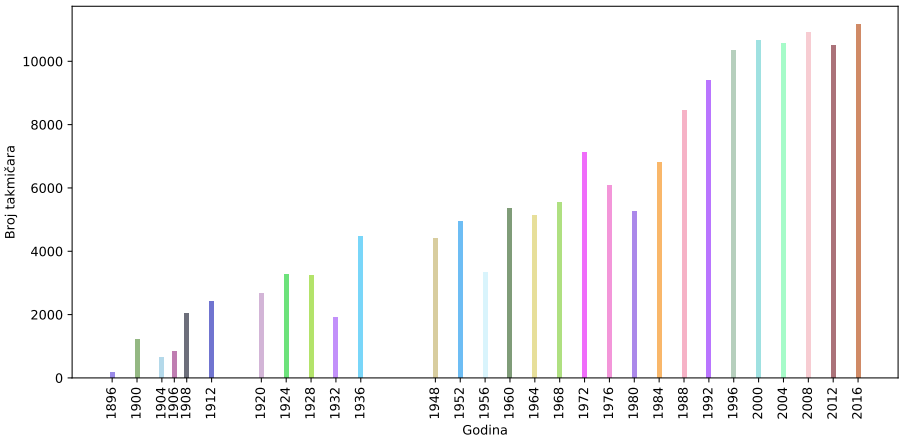
3.2 Dodatna analiza i vizualizacija podataka

Kako bismo još bolje razumeli podatke, dodatno ćemo ih analizirati i vizualizovaćemo ih korišćenjem biblioteke *matplotlib*. Učitaćemo podatke iz novoformirane datoteke i prikazaćemo neke zavisnosti njihovih atributa. Prvo što ćemo prikazati je broj učesnika na svakoj od letnjih olimpijada kroz istoriju. Isto ćemo učiniti odvojeno za muške, odnosno ženske učesnike. U listingu 16 je prikazan postupak izračunavanja za sve takmičare. Dodavanjem filtera za pol, možemo izračunati broj ženskih, odnosno muških takmičara na svakoj olimpijadi (zbog sličnosti sa prethodnim, u ova dva slučaja neće biti navođen listing već samo plot). Na slici 1 prikazan je broj svih učesnika na svakoj od olimpijada kroz istoriju. Na slici 2 prikazan je broj muških, a na slici 3 broj ženskih učesnika kroz istoriju.

```
# ucitavanje podataka iz nove datoteke
data = pd.read_csv('../data/full.csv')

# izdvajanje redova sa letnjih olimpijada sa cuvanjem
# jednog reda za svakog takmicara
filtered_data = data[(data['Season'] == 'Summer')].
                    drop_duplicates(['Year', 'ID'])
years = filtered_data[['Year']]
# broj pojavljivanja svake godine odrzavanja i
```

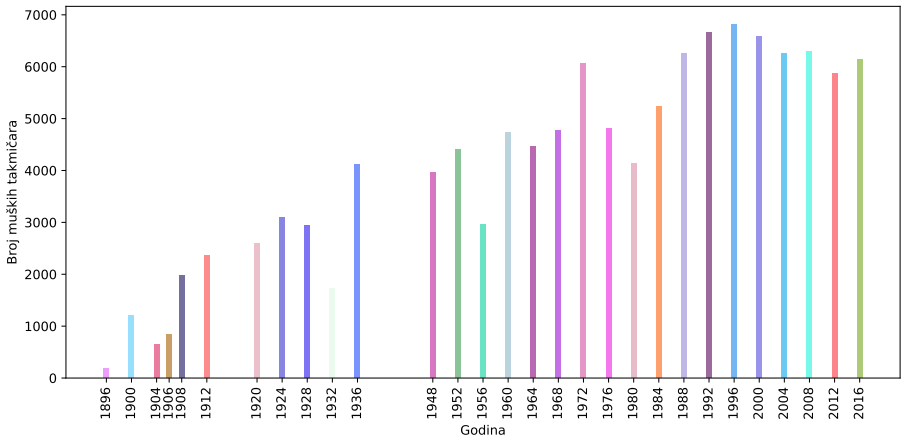
```
# sortiranje po godinama
contestants_count = pd.Series([x for x in years['Year']])
                        .value_counts()
contestants_count = contestants_count.sort_index()

# kreiranje DataFrame-a od prethodno izracunatih
# vrednosti
contestants_count_df =
    pd.DataFrame(contestants_count,
                 columns=['Count']).reset_index()

# godine na x-osu, broj ucesnika na y-osu
x = contestants_count_df['index']
y = contestants_count_df['Count']

f = plt.figure(figsize=(10, 5))
plt.plot(x, y, linestyle='None')
# napravimo listu odgovarajuce duzine sa nasumicno
# napravljenim bojama u heksadekadnom formatu
number_of_colors = contestants_count_df.count(axis=0)[0]
colors = ["#"+''.join([random.choice('0123456789ABCDEF')
                       for j in range(6)])
          for i in range(number_of_colors)]

plt.bar(x, y, alpha=0.6, align='center', color=colors,
        linewidth=10)
plt.xlabel("Godina")
plt.ylabel("Broj takmicara")
plt.xticks(x)
plt.tick_params(axis='x', which='major', rotation=90)
plt.show()
```

Listing 16: Učitavanje podataka i računanje broja učesnika za svaku olimpijadu

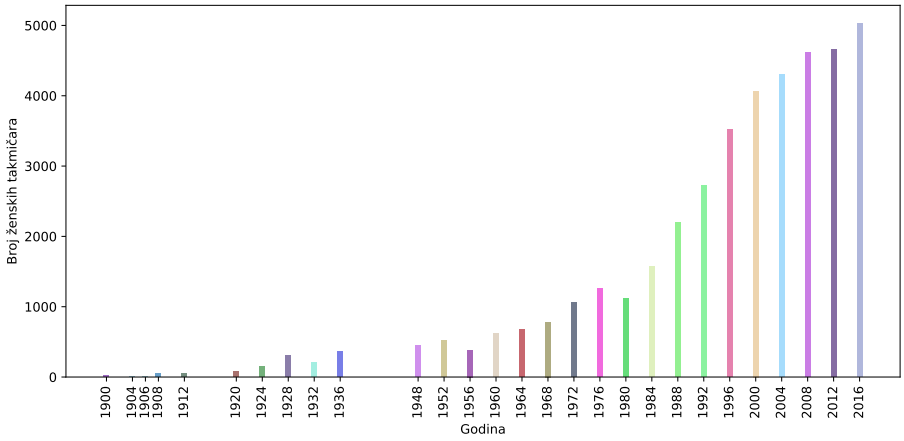


Slika 1: Broj učesnika kroz istoriju

Iz prikazanih plotova vidimo da je broj učesnika na Letnjim olimpijskim igrama u skoro neprestanom porastu, sa izuzetkom nekoliko olimpijada kada je broj učesnika opadao. Možemo videti i da na prvoj olimpijadi nije bilo ženskih takmičara. Decenijama nakon toga, broj ženskih takmičara je bio neuporedivo manji od kolega suprotnog pola. Tek krajem



Slika 2: Broj muških učesnika kroz istoriju



Slika 3: Broj ženskih učesnika kroz istoriju

sedamdesetih i početkom osamdesetih dolazi do povećanog rasta učesnica i popularizacije ženskog sporta na Olimpijskim igrama.

Na plotovima još vidimo da nekoliko olimpijada "nedostaje". U pitanju su godine u kojima su Olimpijske igre bile otkazivane zbog Prvog svetskog rata, a kasnije i Drugog svetskog rata (1916, 1940, 1944). Poslednja letnja olimpijada pre Drugog svetskog rata je održana u Trećem rajhu (Nacistička Nemačka). Još jedna godina koja ispada iz obrasca održavanja Olimpijskih igara na svake četiri godine je 1906. godina. Tada su održane prve i jedine Olimpijske međuigre (eng. *1906 Intercalated Games*), u Atini. Medalje i uspehe ostvarene na ovoj olimpijadi Međunarodni olimpijski komitet ne ubraja u zvanične [2]. Iz ovog razloga, ovu godinu ćemo obrisati iz našeg skupa podataka, postupkom prikazanim u listingu 17.

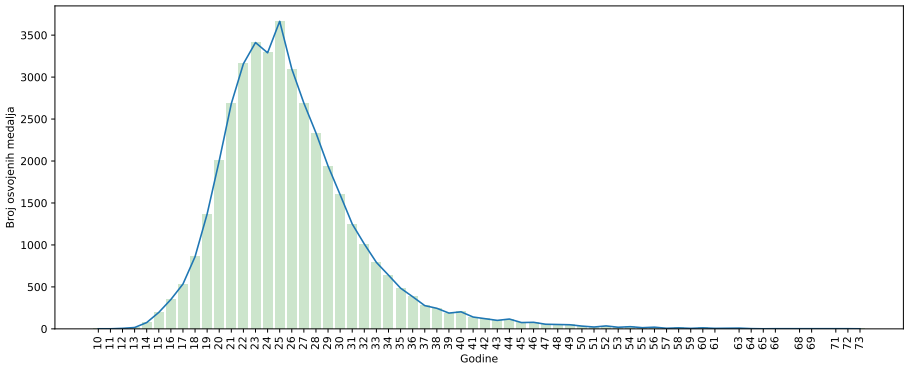
```
1 data = data[data['Year'] != 1906]
```

Listing 17: Brisanje rezultata iz nezvanične godine

Sledeća stvar koju ćemo ispitati je broj osvojenih medalja za sve uzraste takmičara da bismo ispitali zavisnost starosti učesnika i osvajanja medalje. Postupak je prikazan u listingu 18, a plot na slici 4.

```
# izdvajanje redova sa ucesnicima, koji su osvojili neku
# medalju i izdvajanje samo kolona 'Age' i 'Medal'
medals = data[(data['Medal'] != 'None')]
medals = medals[['Age', 'Medal']]
medal_count = pd.Series([x for x in medals['Age']]).
                value_counts()
medal_count = medal_count.sort_index()

medal_df = pd.DataFrame(medal_count, columns=['Count']).
                reset_index()

x = medal_df['index']
y = medal_df['Count']
plt.figure(figsize=(12, 5))
plt.plot(x, y)
plt.bar(x, y, alpha=0.2, align='center', color='green',
        linewidth=10)
plt.xlabel("Godine")
plt.ylabel("Broj osvojenih medalja")
plt.xticks(x)
plt.tick_params(axis='x', which='major', rotation=90)
plt.show()
```

Listing 18: Učitavanje podataka i računanje medalja za sve uzraste



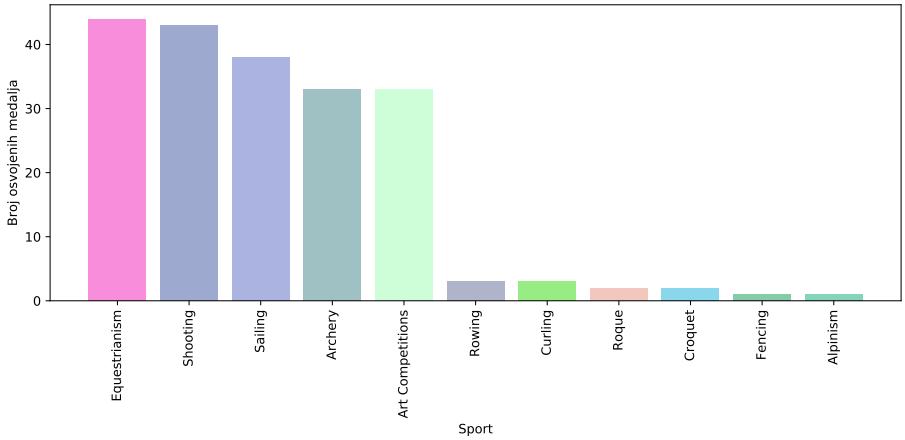
Slika 4: Broj osvojenih medalja po uzrastima takmičara

Vidimo da su medalje osvajali učesnici starosti od 10 do 73 godine, sa izuzetkom uzrasta od 62, 67 i 70 godina. Možemo primetiti i da su medalje najčešće osvajali učesnici starosti između 23 i 25 godina. Takođe, vidimo da je nezanemarljiv broj takmičara starijih od 50 godina osvojio medalju. Izračunajmo broj medalja za svaki sport za koji je to slučaj, na sličan način kao u prethodnoj analizi (listing 19) i prikažimo plot na slici 5.

```
older_medalists_sports = data[(data['Medal'] != 'None') &
                              (data['Age'] > 50)]
older_medalists_sports = older_medalists_sports[['Sport'
    ]]
sports_medal_count = pd.Series([x for x in
    older_medalists_sports['Sport']]).value_counts()
```

```
sports_medal_df = pd.DataFrame(sports_medal_count,
    columns=['Count']).reset_index()

x = sports_medal_df['index']
y = sports_medal_df['Count']

f = plt.figure(figsize=(10, 5))
plt.plot(x, y, linestyle='None')

number_of_colors = sports_medal_df.count(axis=0)[0]
colors = ["#"+''.join([random.choice('0123456789ABCDEF')
                       for j in range(6)])
          for i in range(number_of_colors)]

plt.bar(x, y, alpha=0.5, align='center', color=colors,
        linewidth=10)
plt.xlabel("Sport")
plt.ylabel("Broj osvojenih medalja")
plt.xticks(x)
plt.tick_params(axis='x', which='major', rotation=90)
plt.show()
```

Listing 19: Računanje broja medalja po sportu za takmičare starije od 50 godina



Slika 5: Broj osvojenih medalja za svaki sport za koji je takmičar stariji od 50 godina osvojio neko od odličja

Sa plota na slici 5 vidimo da se izdvaja pet sportova u kojima su takmičari stariji od 50 godina posebno uspešni. U pitanju su, po zastupljenosti medalja: jahanje, streljaštvo, jedriličarstvo, streličarstvo, umetnost.

3.3 Izvođenje novih atributa

Podatke, koje smo do sada obradili, ćemo iskoristiti za kreiranje nove tabele u kojoj će se nalaziti atributi, koji su izvedeni iz postojećih podataka. Novoformiranu tabelu ćemo kasnije koristiti prilikom klasterovanja.

Postupak formiranja nove tabele će biti prikazan u listingu 20. Nakon učitavanja podataka, u tabelu sačinjenu od početnih podataka, dodaćemo novo polje, koje će u slučaju da je medalja osvojena imati vrednost *1*, a

inače *0*. Nazvaćemo ga *MedalWon*. Grupišimo redove po atributu *region* i izračunajmo ukupan broj medalja na svim olimpijadama tih zemalja. Ovu kolonu ćemo nazvati *MedalsCount*. Dalje, izračunaćemo ukupan broj učesnika na svim olimpijadama svake države na sličan način. Ovaj atribut ćemo nazvati *ParticipantsCount*. Sledeći atribut koji ćemo izračunati ćemo nazvati *GamesCount* i on će sadržati broj igara na kojima je država učestvovala. Atribut *PartPerGamesScaled* će sadržati broj svih učesnika podeljen sa brojem igara na kojima je država učestvovala, skaliran na 1 (1 odgovara državi koja je imala najviše učesnika). Atribut *SuccessRate* određuje procenat uspešnosti države, skaliran na 1 (1 odgovara državi koja ima najveći procenat uspešnosti). Izračunaćemo ga tako što ćemo broj svih osvojenih medalja podeliti sa brojem učesnika.

```
# ucitavanje podataka i dodavanje indikatorske kolone -
# da li je medalja osvojena ili ne
data = pd.read_csv('../data/full.csv')
data['MedalWon'] = np.where(
                        data.loc[:, 'Medal'] == 'None',
                        0, 1)

# izracunavanje broja osvojenih medalja za svaku drzavu
# ucesnicu grupisanjem po atributu 'region' i
# sumiranjem prethodno napravljene kolone
country_medals_won_athletes =
    data.groupby(['region'])['MedalWon'].
        agg('sum').reset_index()
country_medals_won_athletes =
    country_medals_won_athletes.sort_values(
        'MedalWon', ascending=False)
country_medals_won_athletes.rename(
    columns={'MedalWon': 'MedalsCount'},
    inplace=True)

# izracunavanje ukupnog broja predstavnika na svim
# olimpijadama za svaku drzavu
athletes_count = data.drop_duplicates(['ID', 'Year'])
athletes_count = athletes_count.groupby(
    ['region'])['ID'].count().reset_index()
athletes_count.rename(
    columns={'ID': 'ParticipantsCount'}, inplace=True)

# izracunavanje ukupnog broja olimpijada na kojima
# je svaka drzava ucestvovala
games_count = data.drop_duplicates(['region', 'Games'])
games_count = games_count.groupby(['region'])['Year'].
    count().reset_index()
games_count.rename(columns={'Year': 'GamesCount'},
                   inplace=True)

# spajanje prethodno izracunatih vrednosti u DataFrame,
# po atributu 'region'
country_data = pd.DataFrame(country_medals_won_athletes)
country_data = pd.merge(country_data,
                        athletes_count,
                        on='region')
country_data = pd.merge(country_data,
                        games_count,
                        on='region')

# racunanje broja ucesnika po igrama i efikasnosti
# drzave, i njihovo skaliranje na 1
```

```
country_data['PartPerGamesScaled'] =
    country_data['ParticipantsCount'] /
    country_data['GamesCount']
country_data['SuccessRate'] =
    country_data['MedalsCount'] /
    country_data['ParticipantsCount']

x = country_data[['PartPerGamesScaled', 'SuccessRate']].
    values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
country_data[['PartPerGamesScaled', 'SuccessRate']] =
    x_scaled

# ispisivanje prvih 10 redova novoformirane tabele
# MC    - MedalsCount
# PC    - ParticipantsCount
# GC    - GamesCount
# PPGS  - PartPerGamesScaled
# SR    - SuccessRate
print(country_data.head(10))

Out>>
       region     MC     PC   GC       PPGS        SR
0         USA   5637  12851   50   1.000000  0.900179
1      Russia   3947   8100   36   0.874540  1.000000
2     Germany   3756  10653   46   0.900349  0.723555
3          UK   2068   8489   51   0.645134  0.499933
4      France   1777   8280   51   0.629077  0.440428
5       Italy   1637   7226   51   0.548101  0.464910
6      Sweden   1536   5414   50   0.417209  0.582225
7      Canada   1352   6464   49   0.509828  0.429233
8   Australia   1349   5534   48   0.444682  0.500254
9     Hungary   1135   4109   49   0.321515  0.566862
```

Listing 20: Kreiranje nove tabele od postojećih atributa

Korišćenjem postupka prikazanom u listingu 21 ćemo napraviti novu *CSV* datoteku, koju ćemo koristiti prilikom klasterovanja.

```
data.to_csv(r'../data/country_stats.csv')
```

Listing 21: Čuvanje novoformirane tabele u *CSV* dokument

4 Klasterovanje

U prethodnom poglavlju smo se rešili nedostajućih vrednosti, dodatno se upoznali sa podacima i vizualizovali ih, zarad sticanja još bolje slike o njihovim osobinama. Takođe, u poglavlju 3.3 smo podatke izdvojili u novu tabelu, koju ćemo u nastavku koristiti. U ovom poglavlju ćemo se baviti klasterovanjem (klaster analizom). Cilj klasterovanja je pronalaženje grupa objekata sličnih karakteristika, odnosno pronalaženje grupa objekata, koje se razlikuju u odnosu na sve druge grupe objekata [4].

Postoji više tehnika klasterovanja. U ovom radu koristićemo sledeće tehnike:

- **K-sredine** - Predstavlja tehniku, koja se zasniva na centroidama i particionisanju elemenata (tačaka), tako da te tačke pripadaju klasteru čijoj centroidi su najbliži. Slovo K u imenu K-sredine predstavlja broj klastera, odnosno broj centroida, na koliko želimo razdvojiti

tačke. Centroide predstavljaju centralne tačke klastera, koje su na početku izabrane nasumično. U svakom koraku primene algoritma za određivanje K klastera, za svaku tačku se određuje kom klasteru pripada na osnovu udaljenosti od centroide. Zatim se računa nova centroida za svaki klaster na osnovu vrednosti tačaka, koje pripadaju ažuriranom klasteru. Sa primenom se staje nakon koraka u kojem nema promena centroida. Bliskost dveju tačaka se računa korišćenjem Euklidskog rastojanja, kosinusnog rastojanja ili drugih mera bliskosti [4].

- **Hijerarhijsko klasterovanje** - Postoje dva glavna tipa hijerarhijskog klasterovanja. Kod prvog, početni broj klastera u ovoj metodi odgovara broju tačaka za koje se klasterovanje vrši. U svakom koraku se dva najbliža klastera spajaju, sve dok ne nastane jedan, sveobuhvatni klaster. Ovaj tip, koji ćemo i koristiti u ovom radu, se naziva klasterovanje spajanjem (eng. *agglomerative clustering*). Drugi pristup polazi od sveobuhvatnog klastera i u svakom koraku se ovaj klaster deli dok ne ostanu samo pojedinačne tačke kao klasteri ili dok ne ostane K klastera. Ovaj tip se naziva klasterovanje razdvanjanjem (eng. *divisive*). Podaci se vizuelizuju korišćenjem dendrograma[5] [4].

4.1 K-sredine

Prvo ćemo primeniti tehniku K-sredina. Da bismo odredili optimalan broj klastera koristićemo *elbow* metod. Rezultat primene ovog metoda je kriva (eng. *elbow curve*) sa koje možemo pročitati optimalan broj klastera. Tu vrednost čitamo na y-osi na mestu gde se kriva prelama i prestaje sa rastom. Ta stagnacija znači da nam veći broj klastera ne bi značajno doprineo u raspoređivanju podataka u klastere [4].

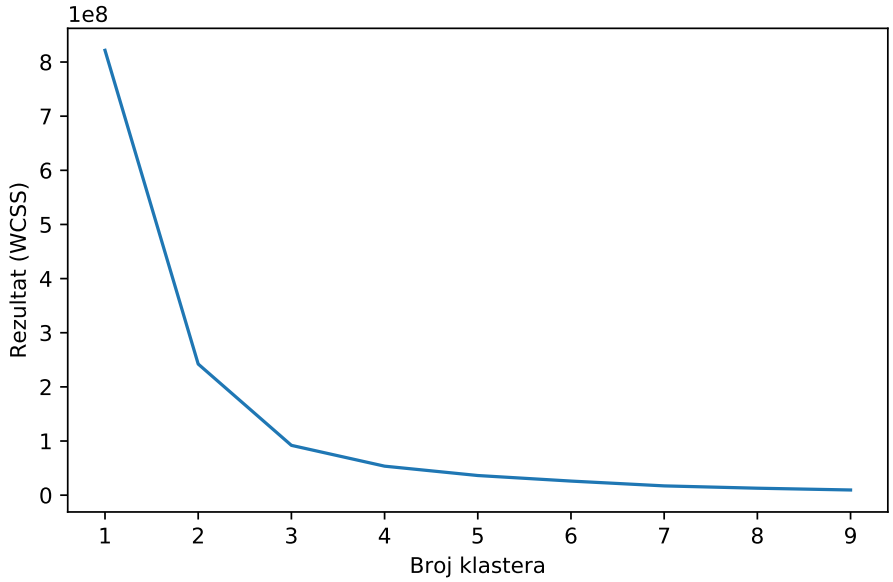
Pokušaćemo da klasterujemo podatke korišćenjem nekoliko kolona. Prvo ćemo to učiniti korišćenjem novoformiranih atributa *MedalsCount* i *ParticipantsCount*. Za početak, korišćenjem gore navedenog *elbow* metoda pronađimo optimalan broj klastera. Postupak određivanja vrednosti, koje će formirati *elbow* krivu prikazan je u listingu 22. Rezultat izvršavanja ovog listinga prikazan je na slici 6

```
# biramo podskup podataka, koji zelimo iskoristiti za
# klasterovanje
cluster_data = country_data[['ParticipantsCount',
                             'MedalsCount']]

# biramo raspon klastera za koji zelimo proveravati
# kvalitet i koristimo metod K-sredina za svaku
# od opcija za K
ec_cluster_options = range(1, 10)
kmeans_res = [KMeans(n_clusters=i, init='k-means++')
              for i in ec_cluster_options]
# wcss - within-cluster sum of square
wcss = [kmeans_res[i].fit(cluster_data).inertia_
        for i in range(len(kmeans_res))]

f_ec = plt.figure(figsize=(6, 4))
plt.plot(range(1, 10), wcss)
plt.xlabel('Broj klastera')
plt.ylabel('Rezultat (WCSS)')
```

[5]Dendrogram predstavlja dijagram, koji pokazuje hijerarhijski odnos između objekata.

```
plt.show()
```

Listing 22: Postupak određivanja *elbow* krive



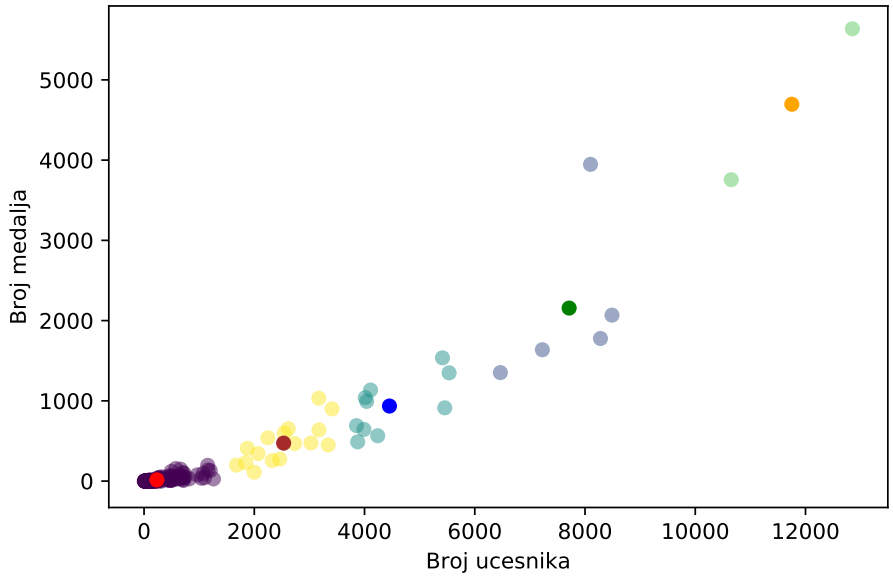
Slika 6: *Elbow* kriva za određivanje broja klastera za atribute *MedalsCount* i *ParticipantsCount*

Sa plota *elbow* krive vidimo da vrednost počinje da minimalno opada nakon broja 5. To je broj koji ćemo izabrati za broj klastera, koji ćemo koristiti za klasterovanje. U listingu 23 prikazan je postupak, kojim će se dobiti plot na kojem možemo videti dobijene klastere. Plot je prikazan na slici 7. Tačke obojene u jarko crveno, narandžasto, plavo, braon i zeleno predstavljaju centroide.

```
# biramo podskup podataka za klasterovanje
cluster_data = country_data[['ParticipantsCount',
                             'MedalsCount']]

kmeans = KMeans(n_clusters=5, init='k-means++')
y_kmeans = kmeans.fit_predict(cluster_data)

centroids = kmeans.cluster_centers_

# izracunate klastere dodajemo u tabelu kao atribut
# 'Cluster'
cluster_data['Cluster'] = y_kmeans
# srednje vrednosti za oba atributa za svaki klaster
kmeans_mean_cluster =
    pd.DataFrame(round(cluster_data.groupby('Cluster').
    mean(),
                 1))

f_clusters = plt.figure(figsize=(6, 4))
plt.xlabel('Broj ucesnika')
plt.ylabel('Broj medalja')
plt.scatter(cluster_data['ParticipantsCount'],
```

```
            cluster_data['MedalsCount'], c=kmeans.labels_
    .astype(float), alpha=0.5)

plt.scatter(centroids[:, 0], centroids[:, 1],
            c=['red', 'green', 'blue',
               'orange', 'brown'])
plt.show()

print(kmeans_mean_cluster)

Out>>
Cluster  ParticipantsCount  MedalsCount CentroidColor
0                    233.0         15.5           red
1                   7711.8       2156.2         green
2                   4452.2        935.5          blue
3                  11752.0       4696.5        orange
4                   2532.2        473.1         brown
```

Listing 23: Klasterovanje podataka po atributima *MedalsCount* i *ParticipantsCount*



Slika 7: Dobijeni klasteri za atribute *MedalsCount* i *ParticipantsCount*

Možemo primetiti da su klasteri i elementi unutar njih uglavnom linearno raspoređeni. Odatle vrlo jasno možemo zaključiti da postoji linearna veza između broja učesnika i broja osvojenih medalja. Činjenica je da što je veći broj takmičara, šanse za medalju rastu. Takođe, možemo pretpostaviti da što je jedna država uspešnija i broj njenih takmičara će na budućim olimpijadama biti u porastu. Prikažimo dve najuspešnije države iz klastera iz gornjeg desnog ugla u listingu 24. Vidimo da su najbliže po broju učesnika.

```
print(countries_data[
    ['region', 'ParticipantsCount', 'MedalsCount']
                    ].sort_values(
```

```
                'ParticipantsCount', ascending=False).
            head(2))

Out>>
    region  ParticipantsCount  MedalsCount
0      USA              12851         5637
2  Germany              10653         3756
```

Listing 24: Dve najuspešnije države ukoliko klasterujemo po atributima *MedalsCount* i *ParticipantsCount*

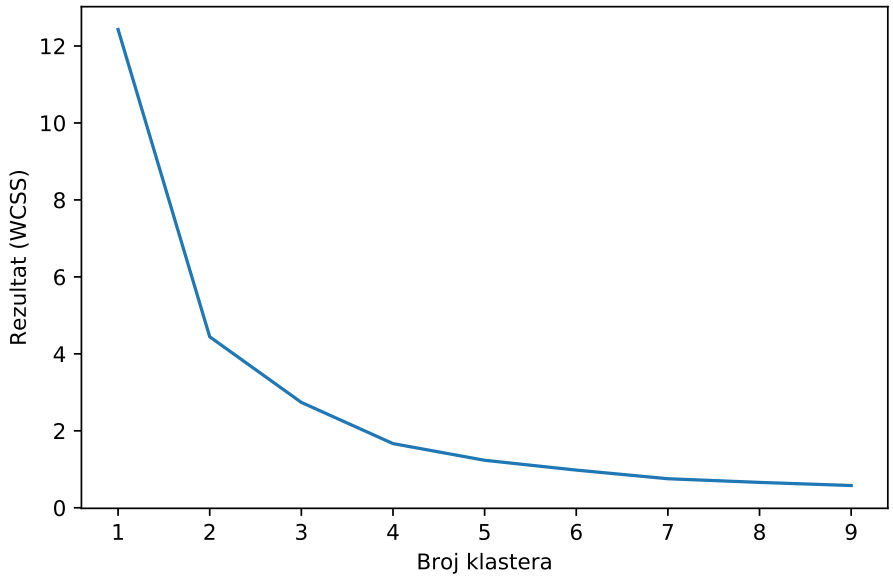
U prethodnom primeru zanemarili smo činjenicu da broj takmičara zavisi i od toga na koliko Olimpijskih igara je ta država učestvovala. Sa više učešća i broj takmičara je mogao biti veći, pa samim tim i broj medalja (kao što smo u prethodnom primeri videli, postoji linearna zavisnost ova dva atributa). U narednom primeru ćemo iskoristiti kolonu *PartPerGamesScaled*, koja će nam pomoći da ne favorizujemo države, koje su na olimpijadi učestvovale više puta od ostalih. Kolona *SuccessRate* će biti korišćena kao parametar uspeha učesnika svake od država. Postupak je analogan prethodnom, pa neće biti detaljno navođen. Dobija se *elbow* kriva prikazana na slici 8. Broj klastera će ponovo biti 5. Klasteri su prikazani na slici 9. Srednje vrednosti klastera i koji klaster odgovara kojoj boji prikazano je u listingu 25.

```
print(kmeans_mean_cluster)

Out>>
Cluster  PartPerGamesScaled  SuccessRate CentroidColor
0                       0.0          0.0           red
1                       0.3          0.5         green
2                       0.9          0.9          blue
3                       0.1          0.2        orange
4                       0.5          0.4         brown
```

Listing 25: Srednje vrednosti klastera

Plot na slici 9 nam daje bolju predstavu o efikasnosti takmičara jedne države. Klaster u gornjem desnom uglu predstavlja 3 najuspešnije države na olimpijadi - Nemačku, Rusiju i SAD. Najveću efikasnost osvajanja odličja imaju takmičari iz Rusije. Iza njih su SAD, a zatim Nemačka, koja je imala oko 2500 učesnika više, ali i oko 200 medalja manje od Rusije. Žuti klaster sa braon centroidom predstavlja države, koje u proseku imaju slabiju efikasnost, dok svetlo plavi klaster sa zelenom centroidom predstavlja države, koje u proseku imaju dobru efikasnost. Država, koja se izdvaja po efikasnosti u klasteru sa zelenom centroidom je Norveška. Prikažimo najboljih 15 država, ukoliko se posmatra njihova efikasnost osvajanja medalja u listingu 26. Vidimo da se Srbija nalazi na 15. mestu najefikasnijih država. Ovaj uspeh treba pripisati velikim uspesima srpskih sportista u timskim sportovima, te je broj odlikovanih takmičara veći.

```
top_sr = country_data.sort_values(
            ['SuccessRate'],
            ascending=False).reset_index()
print(top_sr[['region', 'SuccessRate']].head(15))

Out>>
        region SuccessRate MedalsCount ParticipantsCount
0       Russia    1.000000        3947              8100
1          USA    0.900179        5637             12851
2      Germany    0.723555        3756             10653
```



Slika 8: *Elbow* kriva za određivanje broja klastera za atribute *PartPerGames-Scaled* i *SuccessRate*

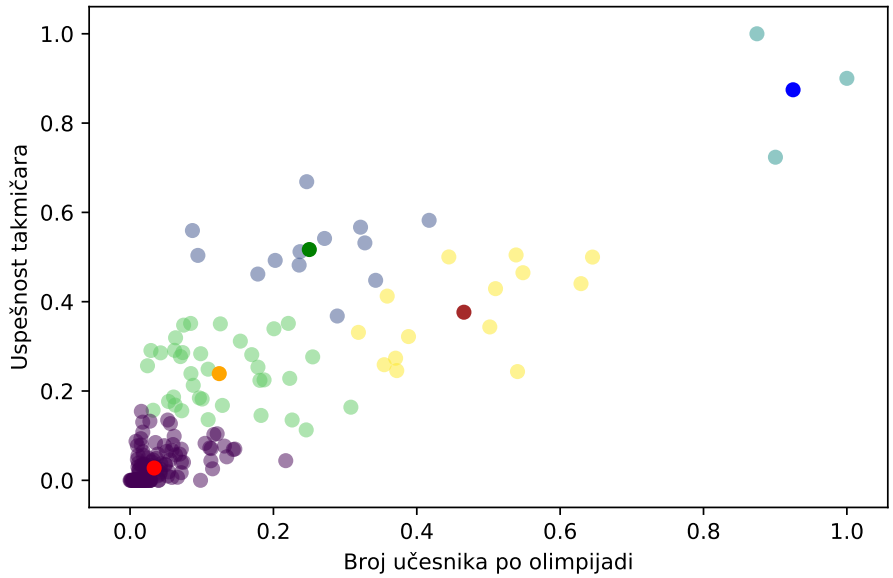
3	Norway	0.668743	1033	3170
4	Sweden	0.582225	1536	5414
5	Hungary	0.566862	1135	4109
6	Jamaica	0.559365	157	576
7	Finland	0.541634	900	3410
8	Netherlands	0.531709	1040	4014
9	Romania	0.512068	653	2617
10	China	0.504662	993	4038
11	Pakistan	0.503682	121	493
12	Australia	0.500254	1349	5534
13	UK	0.499933	2068	8489
14	Serbia	0.492489	539	2246

Listing 26: 15 najefikasnih država na olimpijadi

4.2 Hijerarhijsko klasterovanje

Sada ćemo primeniti metodu hijerarhijskog klasterovanja i uporediti sa rezultatima dobijenim metodom K-sredina. Za određivanje broja klastera, koristićemo dendrogram. Kreirajmo dendrogram za naše podatke, za atribute *ParticipantsCount* i *MedalsCount*. Postupak njegovog kreiranja će biti prikazan u listingu 27, a dendrogram će biti prikazan na slici 10. U pozivu funkcije za kreiranje dendrograma ćemo koristiti argument *truncate_mode='lastp'*, pa ćemo sa *p=20* videti poslednjih 20 odsecanja. Ovo koristimo da bismo smanjili gustinu dendrograma, odnosno da bismo povećali preglednost. Takođe, koristimo parametar *show_contracted=True* kako bismo na plotu iscrtali tačke, čija visina će nam označavati visinu klastera, koji nisu iscrtani. Izvršićemo odsecanje na visini *3000*, koristeći parametar *max_d*.

```
from scipy.cluster.hierarchy import dendrogram,
```



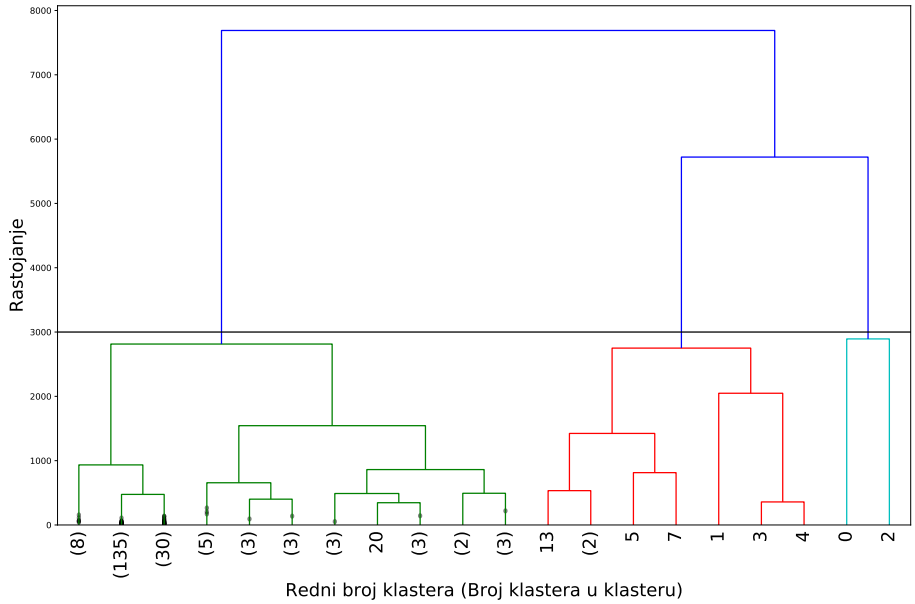
Slika 9: Dobijeni klasteri za atribute *PartPerGamesScaled* i *SuccessRate*

```
                                        linkage

cluster_data = country_data[['ParticipantsCount',
                             'MedalsCount']]

f_dendro = plt.figure(figsize=(15, 10))
# izracunamo rastojanja izmedju elemenata i kreiramo
# dendrogram
Z = linkage(cluster_data, method='average')
dendrogram = dendrogram(Z, truncate_mode='lastp',
                        p=20, leaf_rotation=90,
                        leaf_font_size=21,
                        show_contracted=True)

# vrednost na y-osi na kojoj cemo vrsiti odsecanje
max_d = 3000

plt.axhline(y=max_d, c='k')
plt.xlabel('Broj ucesnika', fontsize=21)
plt.ylabel('Broj osvojenih medalja', fontsize=21)
plt.show()
```

Listing 27: Kreiranje dendrograma za atribute *ParticipantsCount* i *MedalsCount*

Broj na x-osi na slici 10, koji se nalazi u zagradi predstavlja broj elemenata u tom klasteru. Sa ove slike možemo videti da je prava *y=3000*, koju smo odabrali za odsecanje, presekla 3 vertikalne linije. Zbog ovoga će nam broj klastera biti 3. U listingu 28 prikažimo postupak za iscrtavanje klastera, a na slici 11 prikažimo te klastere.

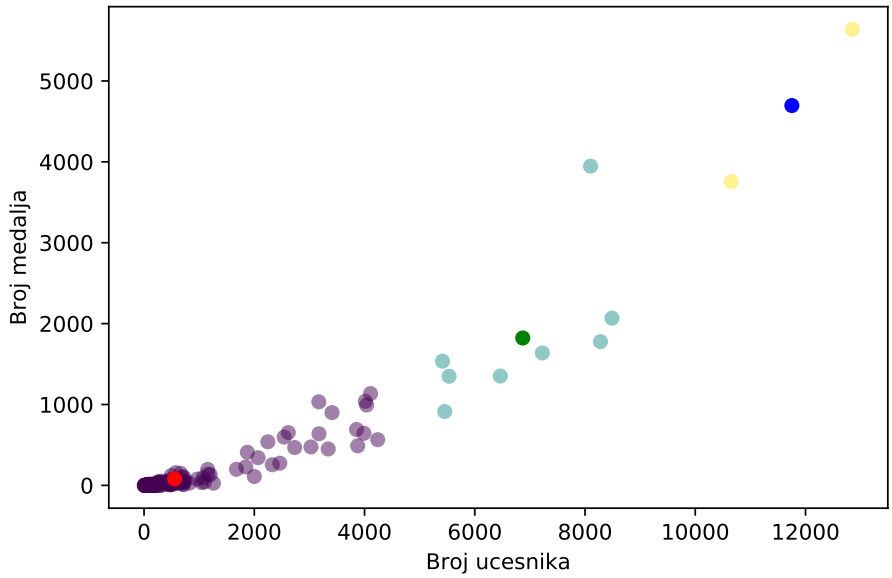
```
from scipy.cluster.hierarchy import linkage, fcluster

cluster_data = country_data[['ParticipantsCount',
```



Slika 10: Dendrogram za atribute *ParticipantsCount* i *MedalsCount*

```
                                ’MedalsCount’]]

clusters = fcluster(Z, 3000, criterion=’distance’)
# oduzimamo 1 da bi brojanje klastera krenulo od 0
# ovo radimo samo da bismo mogli da trazimo
# ispravne indekse iz liste sa bojama
cluster_data[’Cluster’] = clusters - 1

colors = [’red’, ’green’, ’blue’]

cluster_means = pd.DataFrame(
    round(cluster_data.groupby(’Cluster’).mean(), 1))
centroids = np.array(cluster_means)
cluster_means[’CentroidColor’] = pd.DataFrame(colors)

plt.figure(figsize=(6, 4))
plt.scatter(cluster_data[’ParticipantsCount’],
            cluster_data[’MedalsCount’],
            c=clusters, alpha=0.5)
plt.scatter(centroids[:, 0], centroids[:, 1], c=colors)
plt.xlabel(’Broj ucesnika’)
plt.ylabel(’Broj medalja’)
plt.show()

# stampamo srednje vrednosti klastera i boje centroida
print(cluster_means)

Out>>
Cluster  ParticipantsCount  MedalsCount CentroidColor
0                    555.8         80.7           red
1                   6870.2       1822.4         green
2                  11752.0       4696.5          blue
```

Listing 28: Kreiranje klastera za atribute *ParticipantsCount* i *MedalsCount*



Slika 11: Klasteri za atribute *ParticipantsCount* i *MedalsCount*

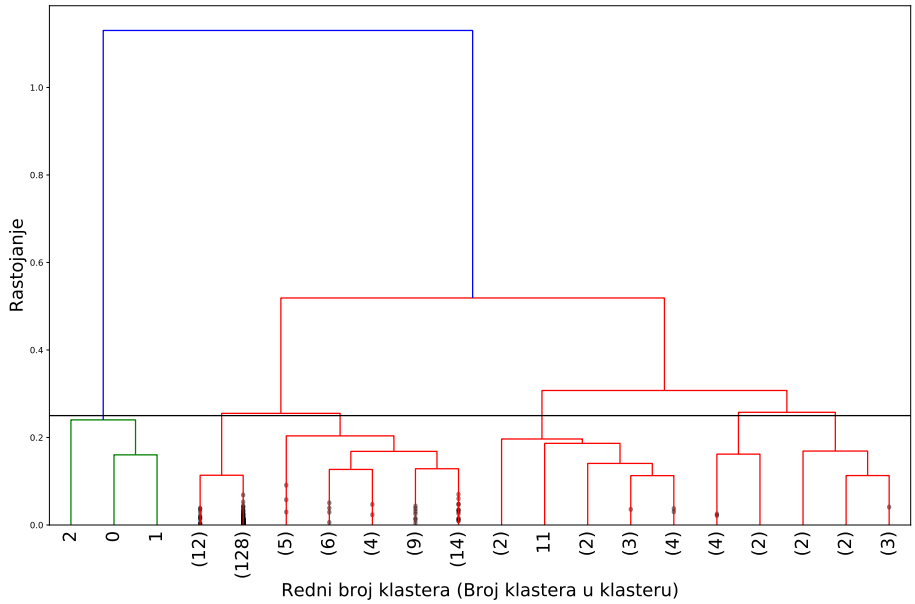
U odnosu na klasterovanje K-sredina, sa parametrima koje smo izabrali, sada imamo 3 klastera, koji opet, vrlo jasno prikazuju određenu linearnu vezu broja učesnika i broja medalja jedne države. Ponovo se izdvajaju dve najuspešnije države gledajući ove atribute - SAD i Nemačka. Osam država se nalazi u klasteru sa zelenom centroidom, koji predstavlja uspešnije države po ovim parametrima, dok se u klasteru sa crvenom centroidom nalaze manje uspešne države.

Sledeći primer, koji ćemo analizirati je hijerarhijsko klasterovanje podataka na osnovu atributa *PartPerGamesScaled* i *SuccessRate*. Prikažimo dendrogram za ove atribute na slici 12, a klastere na slici 13. Srednje vrednosti klastera i koji klaster odgovara kojoj boji prikazano je u listingu 29.

```
print(cluster_means)

Out>>
Cluster  PartPerGamesScaled  SuccessRate CentroidColor
0                       0.9          0.9           red
1                       0.0          0.0         green
2                       0.1          0.2          blue
3                       0.2          0.5        orange
4                       0.4          0.3          pink
5                       0.5          0.5         black
```

Listing 29: Srednje vrednosti klastera za atribute *PartPerGamesScaled* i *SuccessRate*

Sa slike 12 zaključujemo da ćemo ovaj put koristiti 6 klastera. Na slici 13 vidimo 6 klastera, koji nešto bolje razdvajaju države po uspešnosti gledajući dva parametra, koja su korišćena. Ponovo imamo iste 3 najuspešnije države, dok su manje uspešne države jasnije razdvojene, tako da sada imamo klaster sa narandžastom centroidom, koji predstavlja države koje su izuzetno uspešne uprkos manjem broju takmičara. Sa druge strane, u klasteru sa crnom centroidom nalaze se države, koje sa prosečno



Slika 12: Dendrogram za atribute *PartPerGamesScaled* i *SuccessRate*

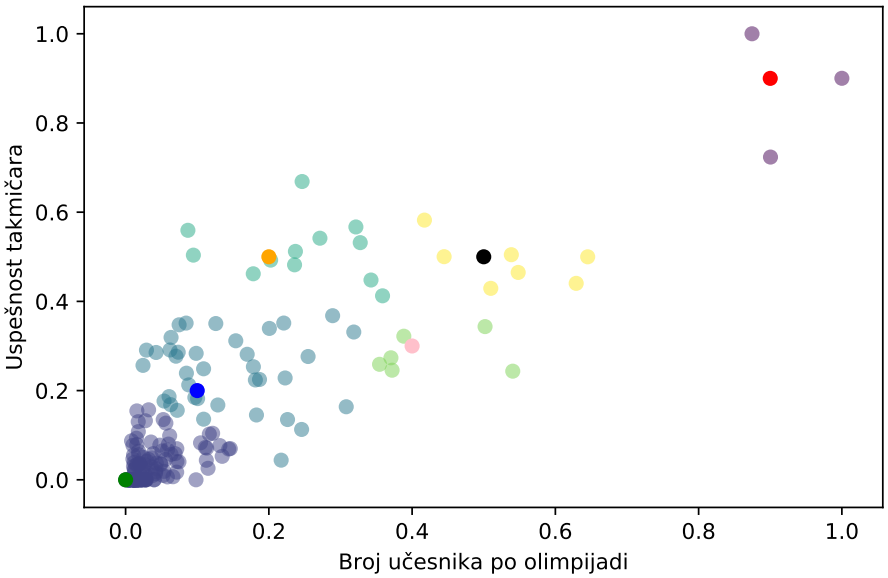
istim brojem učesnika ostvaruju u proseku iste rezultate, kao države iz narandžastog klastera. Države u klasteru sa plavom centroidom predstavljaju države koje u proseku imaju pozitivan učinak gledajući ova dva parametra. Roze klaster predstavlja šest država, koje imaju u proseku veliki broj učesnika, ali slab učinak što se tiče osvojenih medalja. Prikažimo ove države u listingu 30.

```
least_efficient =
    country_data[
        (country_data['PartPerGamesScaled'] > 0.35) &
        (country_data['SuccessRate'] < 0.4)]
print(least_efficient[['region',
                       'PartPerGamesScaled',
                       'SuccessRate']])

Out>>
     region  PartPerGamesScaled  SuccessRate
13    Japan            0.501845     0.343474
20   Poland            0.370429     0.273529
22    Spain            0.354540     0.258907
23   Brazil            0.388423     0.321926
31  Ukraine            0.540514     0.243522
35  Belarus            0.372032     0.245697
```

Listing 30: Države sa negativnim odnosom učesnika i osvojenih medalja

4.3 Klasterovanje učesnika

Poslednje što ćemo ispitati jeste uticaj visine, težine i godina takmičara na osvajanje medalje. Za početak, izvući ćemo podatke o visinama i težinama za sve takmičare, normalizovati ih, a zatim sabrati normalizovana polja za visinu i težinu. Zbir ova dva atributa će činiti novi atribut



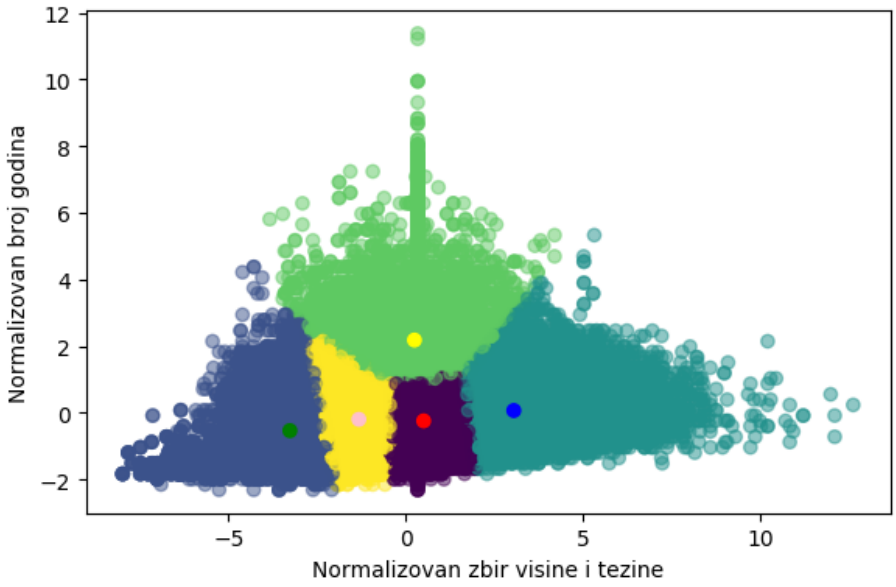
Slika 13: Klasteri za atribute *PartPerGamesScaled* i *SuccessRate*

SizeNorm, koji predstavlja dimenzije takmičara. Takođe, izvući ćemo broj godina takmičara i normalizovati ga. Ovaj podatak će činiti kolonu *AgeNorm*. Postupak za računanje odgovarajućih podataka će biti prikazan u listingu 31. Postupak određivanja i iscrtavanja *elbow* krive i klastera neće biti ispisan, već će biti prikazan samo plot, koji je rezultat klasterovanja. Nakon pozivanja *elbow* metoda odabrano je da će biti pet klastera. Na slici 14 su prikazani klasteri, koji su dobijeni za učesnike bez medalja, a na slici 15 klasteri za učesnike, koji su osvojili neku od medalja.

```
data['HeightNorm'] =
  (data['Height'] - data['Height'].mean()) /
    np.std(data['Height'], axis=0)
data['WeightNorm'] =
  (data['Weight'] - data['Weight'].mean()) /
    np.std(data['Weight'], axis=0)
data['SizeNorm'] =
  data['HeightNorm'] + data['WeightNorm']
data['AgeNorm'] =
  (data['Age'] - data['Age'].mean()) / np.std(data['Age'
    ], axis=0)

size_age_medal = data[data['MedalWon'] == 1]
size_age_no_medal = data[data['MedalWon'] == 0]
size_age_medal = size_age_medal[['SizeNorm', 'AgeNorm']]
size_age_no_medal =
  size_age_no_medal[['SizeNorm', 'AgeNorm']]

print(size_age_medal.head)

Out>>
    SizeNorm   AgeNorm
3   0.339734  1.347182
37  0.339734  0.710409
```

```
38   0.339734   0.710409
40   1.954303   0.392023
41  -0.664815   0.392023
```

Listing 31: Postupak za računanje normalizovanih parametara

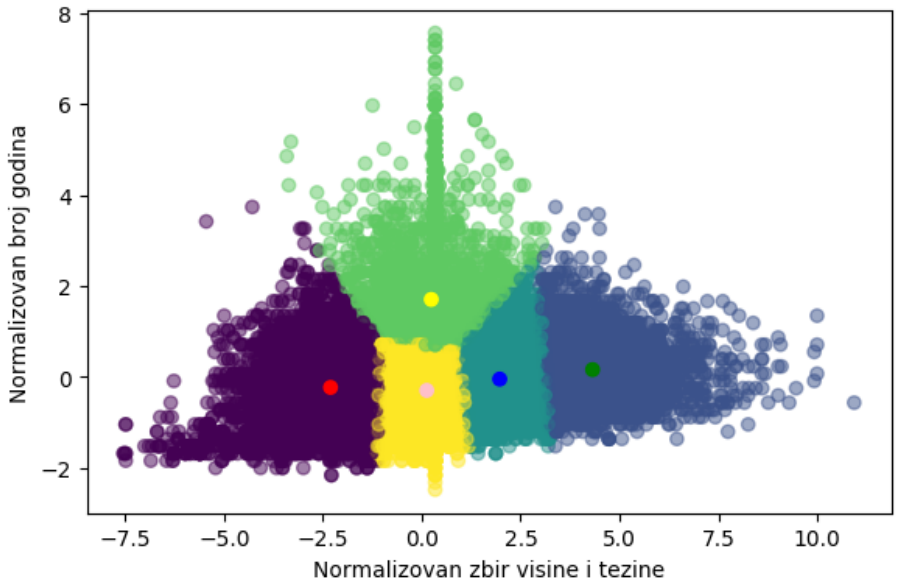


Slika 14: Klasteri za normalizovane atribute *SizeNorm* i *AgeNorm* za učesnike, koji nisu osvojili medalju

Ako pogledamo sliku 14 i sliku 15, možemo zaključiti da se dobijaju skoro potpuno isti klasteri, te da učesnici bilo kog uzrasta, visine ili težine, imaju podjednaku šansu za osvajanje medalja.

5 Zaključak

U ovom radu su primenjeni i prikazani koraci pojedinih osnovnih metoda istraživanja podataka na skupu podataka o takmičarima sa svih modernih Olimpijskih igara. Izvršena je analiza i obrada podataka, a zatim primenjene tehnike klasterovanja i upoređeni su rezultati dobijeni različitim tehnikama. Doneseni su određeni zaključci o olimpijadi i njenim učesnicima: broj učesnika je u porastu; broj ženskih takmičara je u velikom porastu; takmičari svih visina, težina i uzrasta imaju podjednake šanse za uspeh; najstariji osvajač medalje je od najmlađeg stariji 63 godine, itd. Sledeći korak u istraživanju bi bilo dodatno istraživanje postojećih, ali i uvođenje novih podataka, koji bi doneli dodatnu informaciju o državama učesnicama (bruto domaći proizvod, populacija, prosečna primanja, itd) ili o samim takmičarima.



Slika 15: Klasteri za normalizovane atribute *SizeNorm* i *AgeNorm* za učesnike, koji su osvajali medalju

Literatura

[1] Allen. Guttmann. *The Olympics : a history of the modern games / Allen Guttmann.* University of Illinois Press Urbana [Ill.], 1992.

[2] William Mellors Henry and Patricia Henry Yeomans. *An approved history of the Olympic games.* Los Angeles, Calif Southern California Committee for the Olympic Games, 1984 ed edition, 1984.

[3] Richard Stanton. *The Forgotten Olympic Art Competitions.* Victoria: Trafford Publishing, 2001.

[4] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Always learning. Pearson Addison Wesley, 2006.