SVEUČILIŠTE U RIJECI
FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA

Projektni zadatak iz kolegija

UVOD U ANALIZU I VIZUALIZACIJU PODATAKA

ANALIZA PERFORMANSI NOGOMETAŠA

Autor: **Andreas Prgić**
Mentor: **Doc. dr. sc. Lucia Načinović Prskalo**

U Rijeci, lipanj 2025.

Sadržaj

1. Uvod

U sklopu ovog projektnog zadatka, biti će opisan kompletan sadržaj triju projektnih zadataka izrađenih tokom semestra. Glavni cilj jest skupiti sve podatke te rezultate iz navedenih projekata, a zatim ih opisati kao cjelinu.

Projekt je u cijelosti rađen u programskome jeziku *Python*, a tri glavne biblioteke istoimenoga jezika koje su najviše upotrebljavane su: *Numpy, Pandas te MatPlotLib.* Skup podataka koji je odabran za potrebe naziva se *Football Data from Transfermarkt*, a preuzet je sa stranice *Kaggle* (https://www.kaggle.com/datasets/davidcariboo/player-scores).

Opis će sadržavati sve informacije o izrađenim projektima, od prikupljanja podataka, čišćenja podataka pa sve do analize rezultata te vizualizacije podataka. Osim toga sadržavati će detaljan opis kompletnog koda napisanog u okviru projektnih zadataka kako bi pojedini dijelovi kompletnog koda, koji su možda sažetije objašnjeni, bili jasniji.

2. Opis skupa podataka

Skup podataka se sastoji od ukupno 10 različitih tablica (.csv datoteka): *appearances*, *club_games*, *clubs*, *competitions*, *game_events*, *game_lineups*, *games*, *player_valuations*, *players* te *trasfers*. Glavni fokus projekata bio je na tablicama *players*, *clubs*, *games* te po potrebi su korištene tablice *competitions i appearances*.

Navedene tablice možemo protumačiti kao „glavne“ tablice skupa podataka iz razloga što sadrže one najbitnije podatke s navedene web stranice te iz razloga što upotrebom istih može se provesti kvalitetna analize, ali se i mogu dobiti gotovo svi najbitniji rezultati vezani uz nogometaše, klubove te rezultate utakmica.

2.1. Tablica *players*

Tablica *players,* sadrži informacije o raznim nogometašima iz različitih klubova koju sudjeluju na nekom natjecanju.

Atributi koji se nalaze unutar tablice su:

1. **player_id –** identifikacijski broj igrača – **int**
2. **first_name –** ime igrača – **string**
3. **last_name –** prezime igrača – **string**
4. **name –** puno ime igrača – **string**
5. **last_season –** zadnja odigrana sezona (godina odlaska u mirovinu) - **int**
6. **current_club_id –** identifikacijski broj kluba u kojemu je igrač trenutno (u slučaju mirovine, posljednji klub u kojemu je igrač igrao) - **int**
7. **player_code –** oznaka igrača unutar sustava *Transfermarkt* - **string**
8. **country_of_birth –** država rođenja igrača - **string**
9. **city_of_birth –** grad rođenja igrača – **string**
10. **country_of_citizenship –** državljanstvo igrača - **string**
11. **date_of_birth –** datum rođenja igrača – **datetime**
12. **sub_position –** sekundarna pozicija igrača – **string**
13. **position –** pozicija igrača – **string**
14. **foot –** ljevak/dešnjak – **string**
15. **height_in_cm –** visina igrača u centimetrima – **int**
16. **contract_expiration_date –** istek ugovora igrača – **datetime**
17. **agent_name –** ime menadžera – **string**
18. **image_url –** slika igrača (link - *Transfermarkt*) – **string**

19. **url –** igrač (url - *Transfermarkt*) – **string**
20. **current_club_domestic_competition_id –** identifikacijski broj lige u kojoj se trenutni klub igrača natječe – **string**
21. **current_club_name –** ime trenutnog kluba igrača – **string**
22. **market_value_in_eur –** vrijednost igrača u eurima – **int**
23. **highest_market_value_in_eur –** najveća vrijednost igrača u eurima – **int**

2.2. Tablica *clubs*

Tablica *clubs* prikazuje informacije o raznim klubovima unutar sustava *Transfermarkt.*

Atributi unutar tablice su:

1. **club_id –** identifikacijski broj kluba - **int**
2. **club_code –** oznaka kluba unutar sustava *Transfermarkt* – **string**
3. **name –** ime kluba – **string**
4. **domestic_competition_id –** oznaka lige u kojoj se klub nalazi – **string**
5. **total_market_value –** ukupna vrijednost kluba - **int**
6. **squad_size –** broj članova u ekipi - **int**
7. **average_age –** prosječna dob članova – **int**
8. **foreigners_number –** broj članova tima koji su strani državljani - **int**
9. **foreigners_percentage –** postotak članova tima koji su strani državljani – **int**
10. **national_team_players –** broj igrača koji igraju za reprezentaciju - **int**
11. **stadium_name –** ime stadiona kluba – **string**
12. **stadium_seats –** kapacitet stadiona - **int**
13. **net_transfer_record –** neto saldo transfera – **string**
14. **coach_name –** ime trenera – **string**
15. **last_season –** zadnja aktivna sezona kluba – **int**
16. **filename –** ime datoteke (s putanjo datoteke) unutar sustava *Transfermarkt* - **string**
17. **url –** klub (url – *Transfermarkt)* – **string**

2.3. Tablica *games*

Tablica *games* u sebi sadrži podatke o svim odigranim utakmicama koje su zabilježene unutar sustava *Transfermarkt.*

Atributi koje tablica sadrži su:

1. **game_id –** identifikacijski broj utakmice – **int**
2. **competition_id –** identifikacijski broj natjecanja – **string**
3. **season –** sezona u kojoj je utakmica odigrana – **int**
4. **round –** kolo u kojem je odigrana utakmica **- string**
5. **date –** datum kada je utakmica odigrana **- date**
6. **home_club_id –** identifikacijski broj kluba (domaći klub) **- int**
7. **away_club_id –** identifikacijski broj kluba (strani klub) **- int**
8. **home_club_goals –** broj golova domaćeg kluba **- int**
9. **away_club_goals –** broj golova gostujućeg kluba – **int**
10. **home_club_position –** pozicija domaćeg kluba na tablici **- int**
11. **away_club_position –** pozicija gostujućeg kluba na tablici – **int**
12. **home_club_manager_name –** ime trenera domaćeg kluba – **string**
13. **away_club_manager_name –** ime trenera gostujućeg kluba – **string**
14. **stadium –** ime stadiona na kojemu je odigrana utakmica – **string**
15. **attendance –** broj gledatelja **- int**
16. **referee –** ime sudca – **string**
17. **url –** utakmica (url – *Transfermarkt*) **- string**
18. **home_club_formation –** formacija domaćeg kluba – **string**
19. **away_club_formation –** formacija gostujućeg kluba – **string**
20. **home_club_name –** ime domaćeg kluba **- string**
21. **away_club_name –** ime gostujućeg kluba – **string**
22. **aggregate –** rezultat utakmice – **datetime**
23. **competition_type –** vrsta natjecanja – **string**

2.4. Tablica *appearances*

Tablica *appearances* sadrži podatke o odigranim susretima, odnosno o igračima koji su prisustvovali odigranoj utakmici.

Atributi unutar tablice su:

1. **appearance_id –** identifikacijski broj susreta – **string**
2. **game_id –** identifikacijski broj utakmice – **int**
3. **player_id –** identifikacijski broj igrača **– int**
4. **player_club_id –** identifikacijski broj kluba u kojem je igrač igrao za vrijeme odigranog susreta **- int**
5. **player_current_club_id –** identifikacijski broj kluba u kojemu je igrač trenutno – **int**
6. **date –** datum odigranog susreta – **int**
7. **player_name –** puno ime igrača – **string**
8. **competition_id –** identifikacijska oznaka natjecanja/lige – **string**

9. **yellow_cards –** broj žutih kartona - **int**
10. **red_cards –** broj crvenih kartona – **int**
11. **goals –** broj golova – **int**
12. **assists –** broj asistencija – **int**
13. **minutes_played –** broj odigranih minuta – **int**

2.5. Tablica *competitions*

Tablica *competitions* sadrži podatke o svim natjecanjima koja se nalaze unutar skupa podataka.

Atributi unutar tablice su:

1. **competition_id** – identifikacijska oznaka natjecanja/lige – **string**
2. **competition_code** – kod natjecanja – **string**
3. **name** – puni naziv natjecanja – **string**
4. **sub_type** – podtip natjecanja – **string**
5. **type** – tip natjecanja – **string**
6. **country_id** – identifikacijski broj države – **int**
7. **country_name** – naziv države – **string**
8. **domestic_league_code** – kod domaće lige – **string**
9. **confederation** – naziv konfederacije – **string**
10. **url** – službeni URL natjecanja – **string**
11. **is_major_national_league** – oznaka je li natjecanje glavna nacionalna liga – **int**

3. Prvi projektni zadatak

U sklopu prvog projektnog zadatka, osim što je zadatak bio pronaći i opisati skup podataka, bilo je potrebno izvesti operacije iz *NumPy* te *Pandas* biblioteke, a zatim i izračunati statističke mjere nad skupom podataka te pronaći vrijednosti koje bi mogle biti u korelaciji.

3.1. Operacije – NumPy

```python
# Učitavanje podataka
appearances = pd.read_csv('igraci/appearances.csv')
endl = "\n"
# 4.zadatak

# 1. Kreiranje NumPy polja s brojem golova
goals = appearances['goals'].dropna().to_numpy()

# 2. Izračunavanje prosječnog broja golova po nastupu
average_goals = np.mean(goals)
print(f"Prosječan broj golova po nastupu: {average_goals:.2f}",endl)

# 3. Standardizacija broja golova (normalizacija na raspon 0-1)
normalized_goals = (goals - np.min(goals)) / (np.max(goals) - np.min(goals))

# 4. Pronalaženje minimalnog i maksimalnog broja golova u jednoj utakmici
min_goals = np.min(goals)
max_goals = np.max(goals)
print(f"Minimalno golova u nastupu: {min_goals}, Maksimalno: {max_goals}",endl)

# 5. Izračunavanje varijance broja golova
variance_goals = np.var(goals)
print(f"Varijanca golova po nastupu: {variance_goals:.2f}",endl)
```

Slika 1 Kod vezan uz NumPy operacije

Unutar priloženoga koda možemo primijetiti korištenje raznih operacija iz bilblioteke NumPy. Početno, spremamo sadržaj .csv datoteke (appearances.csv) u novu varijablu *appearances*. Nakon toga stupac *goals* iz tablice pretvaramo u NumPy polje kako bismo mogli koristiti brze i vektorske operacije, koje nam nudi biblioteka NumPy.

U nastavku su navedene ostale operacije prikazane na slici:

- `np.mean(goals)`

- Koristi se za izračun prosječnog broja golova po nastupu te služi za uvid u ukupnu efikasnost igrača

- `(goals - np.min(goals)) / (np.max(goals) - np.min(goals))`
  - Omogućuje usporedbu među igračima bez obzira na ekstremne vrijednosti

- `np.var(goals)`
  - Služi za izračun varijacija golova među igračima jer ako ima više varijacija dolazi do veće oscijalcije u učinkovitosti igrača

- `np.min(goals) i np.max(goals)`
  - Pomoću min i max izračunavamo minimalni i maksimalni broj golova kako bi nam kasnije bilo lakše rangirati igrače

- `goals=appearances['goals'].dropna().to_numpy()`
  - Ovom linijom koda pretvaramo Pandas serije u NumPy polje kako bi lakše i brže obrađivali podatke

3.2. Operacije – Pandas

```python
# 5.zadatak

# 1. Grupiranje - ukupni golovi, asistencije, minute i broj nastupa
player_stats = appearances.groupby(['player_id', 'player_name']).agg({
    'goals': 'sum',
    'assists': 'sum',
    'minutes_played': 'sum',
    'appearance_id': 'count'  # broj nastupa
}).reset_index()

player_stats = player_stats.rename(columns={'appearance_id': 'appearances'})

# 2. Filtriranje - samo ozbiljni igrači
filtered_player_stats = player_stats[
    (player_stats['minutes_played'] > 500) |
    (player_stats['goals'] > 0) |
    (player_stats['assists'] > 0)
].copy()

# 3. Novi stupac - prosjek minuta po nastupu
filtered_player_stats['avg_minutes_per_appearance'] = filtered_player_stats['minutes_played'] / filtered_player_stats['appearances']

# 4. Sortiranje po golovima
sorted_by_goals = filtered_player_stats.sort_values(by='goals', ascending=False)

# 5. Top 10 asistenta
top10_assists = filtered_player_stats.sort_values(by='assists', ascending=False).head(10)
```

Slika 2 Kod vezan uz Pandas operacije

Idući zadatak bio je vezan uz operacije iz biblioteke Pandas. Kako bismo demonstrirali operacije, također je korištena ranije spomenuta varijabla appearances, koja predstavlja pandas DataFrame nastao učitavanjem podataka iz CSV datoteke (odnosno: `appearances = pd.read_csv('igraci/appearances.csv')`).

U nastavku su objašnjeni ostali dijelovi programskoga koda:

- `groupby([...]).agg({...})`
  - Kako bi izračunali statistike po igraču potrebno je grupirati podatke po igračima i agregirati vrijednosti
- `DataFrame[uvjet]`
  - Filtriranje redaka prema uvjetima kako bi makli nepoželjne igrače iz analize
- `sort_values(by='goals', ascending=False)`
  - Kako bi najlakše prepoznali najefikasnije igrače potrebno je sortirati podatke po broju golova u silaznom redoslijedu
- `assign()` ili `DataFrame['novi stupac'] = ...`
  - Izračunom i dodavanjem novog stupca je korisno u kreiranju dodatnih metrika i uvida

3.3. Izračun statističkih mjera

```
# 6.zadatak

# 1. Prosječan broj ukupnih golova po igraču
mean_goals = filtered_player_stats['goals'].mean()
print(f"Prosječan broj ukupnih golova po igraču: {mean_goals:.2f}",endl)

# 2. Medijan ukupnog broja asistencija po igraču
median_assists = filtered_player_stats['assists'].median()
print(f"Medijan ukupnog broja asistencija po igraču: {median_assists}",endl)

# 3. Standardna devijacija ukupno odigranih minuta po igraču
std_minutes = filtered_player_stats['minutes_played'].std()
print(f"Standardna devijacija ukupno odigranih minuta: {std_minutes:.2f}",endl)
```

Slika 3 Kod vezan uz operacije izračuna triju statističkih mjera

U priloženome kodu prikazan je proces izračuna triju statističkih mjera – aritmetičke sredine (`mean()`), medijana (`median()`) te standardne devijacije (`std()`). Sve navedene funkcije nalaze se unutar Pandas biblioteke.

U nastavku su navedene funkcije detaljnije opisane:

- `mean_goals = filtered_player_stats['goals'].mean()`
  - Računanje aritmetičke sredine ukupnih postignutih golova kako bi imali bolji uvid u opću razinu efikasnosti među igračima
- `median_assists = filtered_player_stats['assists'].median()`
  - Određuje se srednja vrijednost asistencija kako bi izbjegli neravnomjernu raspoređenost podataka
- `std_minutes = filtered_player_stats['minutes_played'].std()`
  - Mjeri broj koliko se odigranih minuta razlikuje od prosjeka što nam pomaže u otkrivanju razlika između standarnih i ekstremnih igrača

3.4. Korelacija

```
# 7.zadatak

# 1. Korelacija između minuta i golova
correlation_minutes_goals = filtered_player_stats['minutes_played'].corr(filtered_player_stats['goals'])
print(f"Korelacija između odigranih minuta i postignutih golova: {correlation_minutes_goals:.2f}",endl)

# 2. Korelacija između minuta i asistencija
correlation_minutes_assists = filtered_player_stats['minutes_played'].corr(filtered_player_stats['assists'])
print(f"Korelacija između odigranih minuta i asistencija: {correlation_minutes_assists:.2f}",endl)

# 3. Korelacija između golova i asistencija
correlation_goals_assists = filtered_player_stats['goals'].corr(filtered_player_stats['assists'])
print(f"Korelacija između golova i asistencija: {correlation_goals_assists:.2f}",endl)
```

Slika 4 Kod vezan uz korelaciju

U prikazanome kodu možemo primijetiti korištenje funkcija za izračun korelacije između različitih varijabli.

U nastavku su detaljnija objašnjenja:

- `correlation_minutes_goals = filtered_player_stats['minutes_played'].corr(filtered_player_stats['goals'])`

- Računanje korelacije između odigranih minuta i postignutih golova kako bi se ustanovilo postoji li povezanost između broja minuta na terenu i efikasnosti u postizanju golova među igračima

- `correlation_minutes_assists = filtered_player_stats['minutes_played'].corr(filtered_player_stats['assists'])`
  - Određuje se povezanost između odigranih minuta i broja asistencija kako bi vidjeli koliko igrači koji više igraju doprinose i u stvaranju šansi za golove

- `correlation_goals_assists = filtered_player_stats['goals'].corr(filtered_player_stats['assists'])`
  - Mjeri se povezanost između broja golova i asistencija kako bi se utvrdilo jesu li igrači koji češće postižu golove i oni koji češće daju asistencije isti, odnosno koliko su te dvije uloge povezane unutar ekipe

4. Drugi projektni zadatak

U sklopu drugog projektnog zadatka, glavni cilj bio je provesti detaljnu analizu podataka kroz čišćenje, manipulaciju, deskriptivnu statistiku, inferencijalnu statistiku te naprednu statističku analizu. Zadatak je bio podijeljen u nekoliko ključnih dijelova koji su nam omogućili dublje razumijevanja skupa podataka o nogometašima.

4.1. Učitavanje i provjera podataka

```
# ZADATAK 1
# 1.a

clubs = pd.read_csv('clubs.csv')
games = pd.read_csv('games.csv')
players = pd.read_csv('players.csv')
competitions = pd.read_csv('competitions.csv')
```

Slika 5 Kod za učitavanje podataka iz odabranih tablica

Kao što se može vidjeti na priloženoj slici br. 5, koristili smo podatke iz četiriju odabranih tablica: **clubs, games, players** i **competitions**.

4.2. Analiza nedostajućih vrijednosti

```
# NEDOSTAJAĆE VRIJEDNOSTI

print("Nedostajuće vrijednosti - clubs: ")
print(clubs.isnull().sum().sum(), endl)
print("Nedostajuće vrijednosti - games: ")
print(games.isnull().sum().sum(), endl)
print("Nedostajuće vrijednosti - players: ")
print(players.isnull().sum().sum(), endl, endl)
```

Slika 6 Kod za ispisivanje nedostajućih vrijednosti

Za tri tablice, proveli smo analizu nedostajućih vrijednosti. Rezultat ovog koda bio je da smo u tablici **clubs** imali 965 nedostajućih vrijednosti, u tablici **games** 95508 nedostajućih vrijednosti te u tablici **players** imali smo 43874 nedostajućih vrijednosti.

Objašnjenje funkcija:

- `isnull()` – idenitificira NaN vrijednosti u DataFrame-u
- `sum().sum()` – prvi `sum()` broji nedostajuće vrijednosti po stupcima, drugi `sum()` računa ukupan broj

Analiza dupliciranih vrijednosti

```
# DUPLICIRANE VRIJEDNOSTI
print("Duplicirane vrijednosti - clubs: ")
print(clubs.duplicated().sum().sum(), endl)
print("Duplicirane vrijednosti - games: ")
print(games.duplicated().sum().sum(), endl)
print("Duplicirane vrijednosti - players: ")
print(players.duplicated().sum().sum(), endl, endl)
```

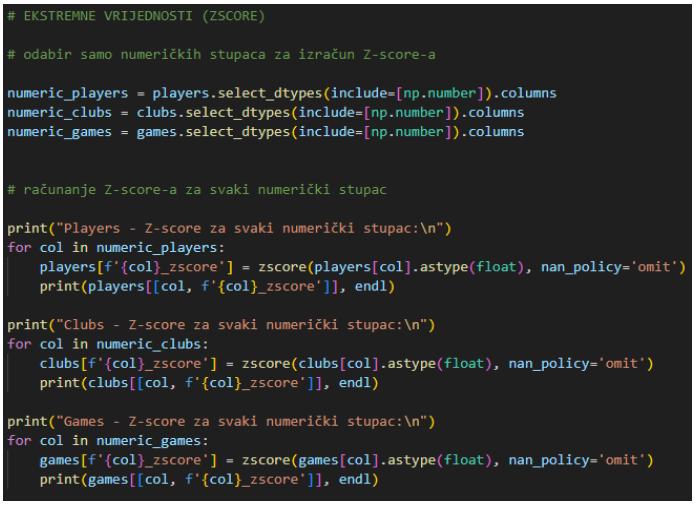
Slika 7 Kod za ispis dupliciranih vrijednosti

Osim što smo proveli analizu nedostajućih vrijednosti za već spomenute tri tablice, proveli smo i analizu dupliciranih vrijednosti. Rezultat ove analize je bio taj da nije bilo dupliciranih vrijednosti u tablicama odnosno, nula je bila rezultat ispisa za svaku tablicu.

Objašnjenje funkcija:

- `duplicated()` – identificira potpuno identične redove
- `sum()` – broji broj dupliciranih redova

4.3. Analiza ekstremnih vrijednosti pomoću Z-score



```
# EKSTREMNE VRIJEDNOSTI (ZSCORE)

# odabir samo numeričkih stupaca za izračun Z-score-a

numeric_players = players.select_dtypes(include=[np.number]).columns
numeric_clubs = clubs.select_dtypes(include=[np.number]).columns
numeric_games = games.select_dtypes(include=[np.number]).columns


# računanje Z-score-a za svaki numerički stupac

print("Players - Z-score za svaki numerički stupac:\n")
for col in numeric_players:
    players[f'{col}_zscore'] = zscore(players[col].astype(float), nan_policy='omit')
    print(players[[col, f'{col}_zscore']], endl)

print("Clubs - Z-score za svaki numerički stupac:\n")
for col in numeric_clubs:
    clubs[f'{col}_zscore'] = zscore(clubs[col].astype(float), nan_policy='omit')
    print(clubs[[col, f'{col}_zscore']], endl)

print("Games - Z-score za svaki numerički stupac:\n")
for col in numeric_games:
    games[f'{col}_zscore'] = zscore(games[col].astype(float), nan_policy='omit')
    print(games[[col, f'{col}_zscore']], endl)
```

Slika 8 Kod za izračun ekstremnih vrijednosti

Objašnjenje Z-score funkcija:

- `select_dtypes(include=[np.number])` - izdvaja samo numeričke stupce
- `zscore()` - računa Z-score vrijednost za svaki element
- `nan_policy='omit'` - ignorira nedostajuće vrijednosti prilikom računanja
- Vrijednosti s |Z-score| > 3 smatraju se ekstremnim vrijednostima

4.4. Čišćenje i uređivanje podataka

Rukovanje nedostajućim vrijednostima u tablici Players

```
# PLAYERS
# contract_expiration_date - ako nema datuma isteka ugovora, Dodajemo "Expired"

players['contract_expiration_date'] = players['contract_expiration_date'].fillna('Expired')
print(players[['name', 'contract_expiration_date']].head(10), endl)

# agent_name - ako nema imena agenta, Dodajemo "No agent"
players['agent_name'] = players['agent_name'].fillna('No agent')
print(players[['name', 'agent_name']].head(10), endl)


# sve ostale nedostajuće vrijednosti zamjenjujemo s "No data"
players = players.fillna('No data')
# ispis prvih 10 redaka uređene tablice
print(players.head(10), endl)
```

Slika 9 Kod za uređivanje nedostajućih vrijednosti u tablici Players

Obzirom da je tablica Players imala nedostajućih vrijednosti, odlučili smo urediti podatke kako ne bi bilo praznih redova u tablici.

Objašnjenje pristupa čišćenja podataka:

- `fillna('Expired')` – logično je pretpostaviti da nedostajući datum znači istekli ugovor
- `fillna('No agent')` - mnogi igrači nemaju agente, posebno mlađi ili manje poznati
- `fillna('No data')` - generička oznaka za sve ostale nedostajuće informacije

Formatiranje podataka u tablici Games

```
# GAMES
# aggregate - promijeni format rezultata iz npr. 3:03:00 u 03:03

def format_aggregate(val):
    if pd.isna(val) or str(val).strip() == '':
        return ''
    parts = str(val).split(':')
    if len(parts) < 2:
        return val
    hour = parts[0].zfill(2)
    minute = parts[1].zfill(2)
    return f"{hour}:{minute}"

games['aggregate'] = games['aggregate'].astype(str).apply(format_aggregate)
print(games['aggregate'].head(10))

# sve ostale nedostajuće vrijednosti zamjenjujemo s "No data"
games = games.fillna('No data')

# ispis prvih 10 redaka uređene tablice
print(games.head(10), endl)
```

Slika 10 Kod za formatiranje podataka u tablici Games

Objašnjenje formatiranja:

- Standardizacija prikaza rezultata iz formata "3:03:00" u "03:03"
- `zfill(2)` - dodaje nulu na početak ako je broj jednoznamenkast
- `apply()` - primjenjuje funkciju na svaki element stupca

4.5. Izračun ukupne tržišne vrijednosti klubova

```
# CLUBS
# izračun total_market_value kao zbroj vrijednosti svih igrača u tablici players
# pretvorba u numerički format
players['market_value_in_eur'] = pd.to_numeric(players['market_value_in_eur'], errors='coerce')

# Zamjena NaN s 0
players['market_value_in_eur'] = players['market_value_in_eur'].fillna(0).astype(int)

# Grupiranje i sumiranje
club_values = players.groupby('current_club_id')['market_value_in_eur'].sum()

# Spajanje s klubovima
clubs['total_market_value'] = clubs['club_id'].map(club_values).fillna(0).astype(int)

# Ispis rezultata
print(clubs[['name', 'total_market_value']].head(10))

# sve ostale nedostajuće vrijednosti zamjenjujemo s "No data"
clubs = clubs.fillna('No data')

# ispis prvih 10 redaka uređene tablice
print(clubs.head(10), endl, endl)
```

Slika 11 Kod za izračun ukupne tržišne vrijednosti klubova

Objašnjenje operacija:

- `pd.to_numeric(errors='coerce')` - pretvara u brojeve, neispravne vrijednosti postavlja na NaN
- `groupby().sum()` - grupira po klubu i računa zbroj vrijednosti svih igrača
- `map()` - mapira izračunate vrijednosti nazad u glavnu tablicu klubova

4.6. Manipulacija podacima

```
# ZADATAK 2
# 2.a
# ispitivanje dimenzija
print("Players: ", endl)
print("Dimenzije:", players.shape)
print("Ukupan broj elemenata:", players.size)
print("Broj dimenzija:", players.ndim)
print("Nazivi osi:", players.axes)
print("Nazivi stupaca:", players.columns, endl)

print("Clubs: ", endl)
print("Dimenzije:", clubs.shape)
print("Ukupan broj elemenata:", clubs.size)
print("Broj dimenzija:", clubs.ndim)
print("Nazivi osi:", clubs.axes)
print("Nazivi stupaca:", clubs.columns, endl)

print("Games: ", endl)
print("Dimenzije:", games.shape)
print("Ukupan broj elemenata:", games.size)
print("Broj dimenzija:", games.ndim)
print("Nazivi osi:", games.axes)
print("Nazivi stupaca:", games.columns, endl)
```

Slika 12 Kod za manipulaciju podacima

Objašnjenje fukncija:

- `shape`: Vraća n-torke s brojem redaka i stupaca
- `size`: Ukupan broj elemenata (redovi × stupci)
- `ndim`: Broj dimenzija DataFrame-a
- `axes`: Lista objekata koji predstavljaju osi
- `columns`: Index objekt s nazivima stupaca

4.7. Izdvajanje podskupova podataka

```
# 2.b
# izdvajanje podskupa po kriteriju
# izdvajanje igrača koji su trenutno u klubu Barcelona te čija je vrijednost veća od 10 milijuna te čiji ugovor nije istekao
igraci = players[
    (players['current_club_name'] == "Futbol Club Barcelona") &
    (players['market_value_in_eur'] > 10000000) &
    (players['contract_expiration_date'] != 'Expired')
]

print("Igrači FC Barcelone čija je vrijednost veća od 10 milijuna i čiji ugovor nije istekao: ", endl)
print(igraci[['name', 'current_club_name', 'market_value_in_eur', 'contract_expiration_date']], endl)
```

Slika 14 Kod za izdvajanje podskupa podataka

Objašnjenje:

- Kombiniranje više uvjeta pomoću logičkih operatora &
- Filtriranje igrača FC Barcelone čija je vrijednost viša od 10 milijuna eura s važećim ugovorom
- Demonstracija kompleksnog logičkog filtriranja

4.8. Kreiranje novog stupca

```python
# kreiranje novog stupca na temelju postojećih podataka
# kreiranje stupca in_domestic_club koji označava je li igrač u domaćem klubu (True/False)
# za potrebe ovoga zadatka koristiti će se dodatna tablica competitions, putem koje možemo doznati državu iz koje je klub
print("Novi stupac in_domestic_club: ", endl)
competitions = pd.read_csv('competitions.csv')


# spajanje tablica players i clubs na temelju current_club_id i club_id
igraci_klubovi = players.merge(
    clubs[['club_id', 'domestic_competition_id']],
    left_on='current_club_id',
    right_on='club_id',
    how='left'
)
# spajanje s tablicom competitions na temelju domestic_competition_id i competition_id
igraci_drzava_kluba = igraci_klubovi.merge(
    competitions[['competition_id', 'country_name']],
    left_on='domestic_competition_id',
    right_on='competition_id',
    how='left'
)

# kreiranje novog stupca in_domestic_club koji označava je li igrač u domaćem klubu (True/False)
players['in_domestic_club'] = igraci_drzava_kluba['country_name'] == players['country_of_birth']

# ispis prvih 10 redaka s novim stupcem
print(players[['name', 'country_of_birth', 'in_domestic_club']].head(10), endl)
```

Slika 15 Kod za kreiranje novoga stupca

Objašnjenje:

- Postupno spajanje triju tablica za dobivanje potrebnih informacija
- Kreiranje boolean varijable koja označava igra li igrač u domaćoj ligi
- Demonstracija složenih operacija spajanja (merge)

4.9. Grupiranje podataka

Broj igrača po zemlji rođenja

```python
# 2.c - grupiranje podataka

# broj igrača po zemlji rođenja
igraci_po_drzavi = players.groupby('country_of_birth')['player_id'].count().reset_index(name='broj_igraca')
igraci_po_drzavi_sorted = igraci_po_drzavi.sort_values(by='broj_igraca', ascending=False)
print("Broj igrača po zemlji rođenja (Top 10):", endl)
print(igraci_po_drzavi_sorted.head(10), endl)
```

Slika 16 Kod za ispis broja igrača po zemlji rođenja

Prosječna tržišna vrijednost igrača po klubu

```
# prosječna tržišna vrijednost igrača po klubu
vrijednost_po_klubu = players.groupby('current_club_name')['market_value_in_eur'].mean().reset_index(name='prosjecna_vrijednost')
vrijednost_po_klubu_sorted = vrijednost_po_klubu.sort_values(by='prosjecna_vrijednost', ascending=False)
vrijednost_po_klubu_sorted['prosjecna_vrijednost'] = vrijednost_po_klubu_sorted['prosjecna_vrijednost'].apply(lambda x: f"{x/1_000_000:.1f} M EUR")
print("Prosječna tržišna vrijednost igrača po klubu (Top 10):", endl)
print(vrijednost_po_klubu_sorted.head(10), endl)
```

Slika 17 Kod za izračunavanje prosječne tržišne vrijednosti igrača po klubu

Ukupan broj utakmica po natjecanju

```
# ukupan broj utakmica po natjecanju (nakon spajanja s tablicom competitions)
games_competitions = games.merge(
    competitions[['competition_id', 'name']],
    on='competition_id',
    how='left'
)

utakmice_po_natjecanju = games_competitions.groupby('name')['game_id'].count().reset_index(name='broj_utakmica')
utakmice_po_natjecanju_sorted = utakmice_po_natjecanju.sort_values(by='broj_utakmica', ascending=False)
print("Ukupan broj utakmica po natjecanju (Top 10):", endl)
print(utakmice_po_natjecanju_sorted.head(10), endl)
```

Slika 18 Kod za ispis ukupnog broja utakmica po natjecanju

Broj klubova po zemlji

```
# broj klubova po zemlji
clubs_merged = clubs.merge(
    competitions[['competition_id', 'country_name']],
    left_on='domestic_competition_id',
    right_on='competition_id',
    how='left'
)

klubovi_po_drzavi = clubs_merged.groupby('country_name')['club_id'].count().reset_index(name='broj_klubova')
klubovi_po_drzavi_sorted = klubovi_po_drzavi.sort_values(by='broj_klubova', ascending=False)
print("Broj klubova po zemlji (Top 10):", endl)
print(klubovi_po_drzavi_sorted.head(10), endl)
```

Slika 19 Kod za ispis broja klubova po zemlji

Objašnjenje grupiranja:

- `groupby()`: Grupira podatke prema određenoj varijabli
- `count()`: Brojač za vrijednosti koje nisu null
- `mean()`: Računa aritmetičku sredinu

- `reset_index()`: Pretvara indeks natrag u stupac
- `sort_values()`: Sortira podatke prema određenom stupcu

Podaci su grupirani po logički povezanim varijablama kako bi dobili agregirane uvide. Grupiranje po državi rođenja igrača omogućuje nam uvid u zastupljenost nacionalnosti u skupu. Grupiranje po klubovima i ligama otkriva gdje su koncentrirani najvrjedniji igrači ili najaktivnija natjecanja.

4.10. Sažete statistike po grupama

Ukupna tržišna igrača po klubu

```
# ukupna tržišna vrijednost igrača po klubu
ukupna_vrijednost_po_klubu = players.groupby('current_club_name')['market_value_in_eur'].sum().reset_index(name='ukupna_vrijednost')
ukupna_vrijednost_po_klubu_sorted = ukupna_vrijednost_po_klubu.sort_values(by='ukupna_vrijednost', ascending=False)
ukupna_vrijednost_po_klubu_sorted['ukupna_vrijednost'] = ukupna_vrijednost_po_klubu_sorted['ukupna_vrijednost'].apply(lambda x: f"{x/1_000_000:.1f} M EUR")
print("Ukupna tržišna vrijednost igrača po klubu (Top 10):", endl)
print(ukupna_vrijednost_po_klubu_sorted.head(10), endl)
```

Slika 20 Kod za izračunavanje ukupne tržišne vrijednosti igrača po klubu

Prosječna starost igrača po zemlji rođenja

```
# prosječna starost igrača po zemlji rođenja
if 'date_of_birth' in players.columns:
    players['date_of_birth'] = pd.to_datetime(players['date_of_birth'], errors='coerce')
    today = pd.to_datetime('today')
    players['age'] = (today - players['date_of_birth']).dt.days // 365
else:
    print("Stupac 'date_of_birth' ne postoji u players tablici.")
prosjecna_starost_po_drzavi = players.groupby('country_of_birth')['age'].mean().reset_index(name='prosjek_starosti')
prosjecna_starost_po_drzavi_sorted = prosjecna_starost_po_drzavi.sort_values(by='prosjek_starosti', ascending=False)
print("Prosječna starost igrača po zemlji rođenja (Top 10):", endl)
print(prosjecna_starost_po_drzavi_sorted.head(10), endl)
```

Slika 21 Kod za izračunavanje prosječne starosti igrača prema zemlji rođenja

```
# broj klubova po zemlji
clubs_merged = clubs.merge(
    competitions[['competition_id', 'country_name']],
    left_on='domestic_competition_id',
    right_on='competition_id',
    how='left'
)
broj_klubova_po_drzavi = clubs_merged.groupby('country_name')['club_id'].nunique().reset_index(name='broj_klubova')
broj_klubova_po_drzavi_sorted = broj_klubova_po_drzavi.sort_values(by='broj_klubova', ascending=False)
print("Broj klubova po zemlji (Top 10):", endl)
print(broj_klubova_po_drzavi_sorted.head(10), endl)
```

Slika 22 Kod za ispis broja klubova po zemlji

Broj utakmica po godini

```
# broj utakmica po godini
games['year'] = pd.to_datetime(games['date'], errors='coerce').dt.year
broj_utakmica_po_godini = games.groupby('year')['game_id'].count().reset_index(name='broj_utakmica')
broj_utakmica_po_godini_sorted = broj_utakmica_po_godini.sort_values(by='broj_utakmica', ascending=False)
print("Broj utakmica po godini:", endl)
print(broj_utakmica_po_godini_sorted.head(10), endl)
```

Slika 23 Kod za ispis broja utakmica po godini

Podatke smo grupirali po relevantnim varijablama te su izračunate ključne sažete statistike: ukupna tržišna vrijednost klubova, prosječna starost igrača po zemlji, broj klubova po državama i broj utakmica po godinama. Ove informacije daju nam uvid u financijsku moć klubova, starosnu strukturu po nacijama i vremensku dinamiku nogometnih događaja.

4.11. Deskriptivna statistika

Srednja vrijednost tržišne vrijednosti igrača

```
# srednja vrijednost tržišne vrijednosti
mean_value = players['market_value_in_eur'].mean()
print(f"Srednja tržišna vrijednost igrača: {mean_value:,.0f} EUR", endl)
```

Slika 24 Kod za izračun srednje vrijednosti tržišne vrijednosti igrača

Standardna devijacija tržišne vrijednosti igrača

```
# standardna devijacija tržišne vrijednosti
std_value = players['market_value_in_eur'].std()
print(f"Standardna devijacija tržišne vrijednosti: {std_value:,.0f} EUR", endl)
```

Slika 25 Kod za izračunavanje standardne devijacije tržišne vrijednosti igrača

Koeficijent varijacije tržišne vrijednosti

```
# koeficijent varijacije tržišne vrijednosti
cv_value = std_value / mean_value
print(f"Koeficijent varijacije tržišne vrijednosti: {cv_value:.2f}", endl)
```

Slika 26 Kod za izračunavanje koeficijenta varijacije tržišne vrijednosti

Izračunate mjere i objašnjenja:

- **Aritmetička sredina:** Prosječna tržišna vrijednost svih igrača
- **Standardna devijacija:** Mjera raspršenosti vrijednosti oko prosjeka
- **Koeficijent varijacije:** Relativna mjera varijabilnosti (σ/μ)

Starost igrača

```
# srednja starost
if 'age' in players.columns:
    mean_age = players['age'].mean()
    std_age = players['age'].std()
    cv_age = std_age / mean_age

    print(f"Srednja starost igrača: {mean_age:.2f} godina", endl)
    print(f"Standardna devijacija starosti: {std_age:.2f}", endl)
    print(f"Koeficijent varijacije starosti: {cv_age:.2f}", endl)
```

Slika 27 Kod za deskriptivnu statistiku starosti igrača

Visina igrača

```
# visina igrača
if 'height_in_cm' in players.columns:
    players['height_in_cm'] = pd.to_numeric(players['height_in_cm'], errors='coerce')
    mean_height = players['height_in_cm'].mean()
    std_height = players['height_in_cm'].std()
    cv_height = std_height / mean_height

    print(f"Srednja visina igrača: {mean_height:.2f} cm", endl)
    print(f"Standardna devijacija visine: {std_height:.2f} cm", endl)
    print(f"Koeficijent varijacije visine: {cv_height:.2f}", endl)
```

Slika 28 Kod za deskriptivnu statistiku visine igrača

Objašnjenje koeficijenta varijacije:

- CV = σ/μ × 100%
- Omogućuje usporedbu varijabilnosti različitih varijabli
- Vrijednosti < 0.1 ukazuju na nisku varijabilnost
- Vrijednosti > 0.3 ukazuju na visoku varijabilnost

Deskriptivna statistika koristi se za dobivanje osnovnih uvida u numeričke podatke. Izračunali smo osnovne mjere poput srednje vrijednosti, standardne devijacije i koeficijenta varijacije. Koeficijent varijacije nam daje mogućnost usporedbe varijabilnosti između različitih podataka bez obzira na jedinice mjere.

4.12. Inferencijalna statistika

Interval pouzdanosti za tržišnu vrijednost igrača

```
from scipy import stats

# 4. inferencijalna statistika

# interval pouzdanosti za tržišnu vrijednost igrača
vrijednosti = players['market_value_in_eur']
vrijednosti = vrijednosti[vrijednosti > 0]

mean = vrijednosti.mean()
std = vrijednosti.std()
n = len(vrijednosti)
confidence = 0.95

margin_error = stats.t.ppf((1 + confidence) / 2, df=n-1) * (std / (n ** 0.5))

lower = mean - margin_error
upper = mean + margin_error

print(f"95% interval pouzdanosti za prosječnu tržišnu vrijednost: {lower:,.0f} EUR - {upper:,.0f} EUR", endl)
```

Slika 29 Kod - interval pouzdanosti za tržišnu vrijednost igrača

Objašnjenje:

- Korištenje t-distribucije zbog nepoznate populacijske varijance
- 95% interval pouzdanosti znači da s 95% sigurnošću možemo reći da se populacijski prosjek nalazi u tom intervalu
- stats.t.ppf(): *objasniti*

T-test za hipotezu je li prosječna tržišna vrijednost veća od 5 milijuna €

```
# t-test – testiramo je li prosječna tržišna vrijednost veća od 5 milijuna
test_value = 5_000_000
t_stat, p_value = stats.ttest_1samp(vrijednosti, test_value)

print(f"t-statistika: {t_stat:.2f}", endl)
print(f"p-vrijednost: {p_value:.4f}", endl)

if p_value < 0.05:
    if mean < test_value:
        print("Značajna razlika: tržišna vrijednost je manja od 5 milijuna EUR (na razini od 5%)", endl)
    else:
        print("Značajna razlika: tržišna vrijednost je veća od 5 milijuna EUR (na razini od 5%)", endl)
else:
    print("Nema značajne razlike ", endl)
```

Slika 30 Kod za T-test hipoteze o tržišnoj vrijednosti

Hipoteze:

- H_0: $\mu = 5{,}000{,}000$ EUR (prosječna tržišna vrijednost jednaka je 5 milijuna €)
- H_1: $\mu \neq 5{,}000{,}000$ EUR (prosječna tržišna vrijednost različita je od 5 milijuna €)

Objašnjenje:

- **t-test:** Mjeri koliko se uzorak razlikuje od hipotetičke vrijednosti
- **p-vrijednost:** Vjerojatnost dobivanja takvog ili ekstremnog rezultata ako je H_0 istinita
- **Odluka:** Ako $p < 0.05$, odbacujemo H_0

4.13. Statistička analiza

ANOVA test - razlike u tržišnoj vrijednosti po pozicijama

```python
# 5. Statistička analiza

from scipy.stats import f_oneway, chi2_contingency, pearsonr
import matplotlib.pyplot as plt

print("ZADATAK 5 - Statistička analiza", endl)

# 5.a ANOVA - Postoji li značajna razlika u tržišnoj vrijednosti igrača po pozicijama?
if 'position' in players.columns:
    top_positions = players['position'].value_counts().head(3).index.tolist()
    grupa_vrijednosti = [
        players[players['position'] == poz]['market_value_in_eur'].dropna()
        for poz in top_positions
    ]
    anova_stat, anova_p = f_oneway(*grupa_vrijednosti)
    print(f"ANOVA test tržišnih vrijednosti između top 3 pozicije ({', '.join(top_positions)}):")
    print(f"F-statistika: {anova_stat:.2f}, p-vrijednost: {anova_p:.4f}", endl)
    if anova_p < 0.05:
        print("Postoji statistički značajna razlika između tržišnih vrijednosti tih pozicija.", endl)
    else:
        print("Nema statistički značajne razlike između tržišnih vrijednosti tih pozicija.", endl)
```

Slika 31 Kod za ANOVA test

Objašnjenje ANOVA testa:

- Testira postojanje značajnih razlika između tri ili više grupa
- H_0: Sve grupe imaju jednake prosjeke
- H_1: Barem jedna grupa ima različit prosjek
- F-statistika predstavlja omjer varijance između grupa i varijance unutar grupa

Chi-kvadrat test nezavisnosti

```python
# 5.a Chi-kvadrat test - Ima li povezanosti između pozicije i činjenice je li igrač u domaćem klubu?
if 'position' in players.columns and 'in_domestic_club' in players.columns:
    kont_tablica = pd.crosstab(players['position'], players['in_domestic_club'])
    chi2_stat, chi2_p, _, _ = chi2_contingency(kont_tablica)
    print("Chi-kvadrat test povezanosti između pozicije i statusa domaćeg kluba:")
    print(f"Chi2 statistika: {chi2_stat:.2f}, p-vrijednost: {chi2_p:.4f}", endl)
    if chi2_p < 0.05:
        print("Postoji značajna povezanost između pozicije i domaćeg statusa igrača.", endl)
    else:
        print("Nema značajne povezanosti između pozicije i domaćeg statusa igrača.", endl)
```

Slika 32 Kod za Chi-kvadrat test

- H_1: Postoji povezanost između pozicije i statusa domaćeg kluba
- Koristi kontingentnu tablicu za analizu

Korelacijska analiza

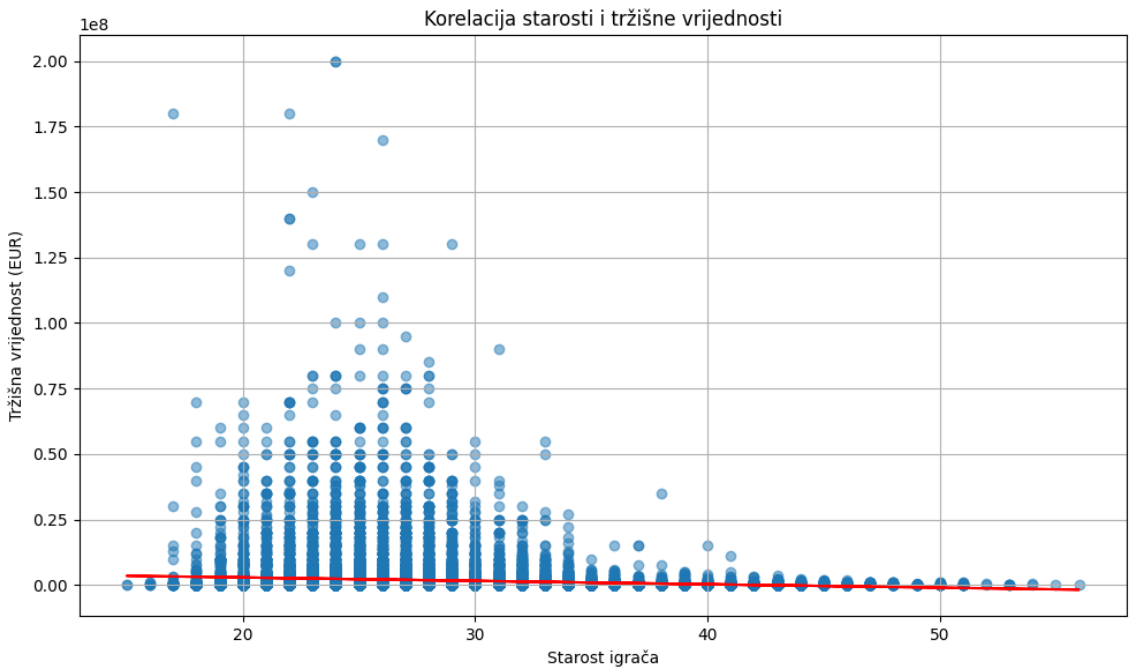
```
# 5.b Korelacija između tržišne vrijednosti i starosti
if 'market_value_in_eur' in players.columns and 'age' in players.columns:
    valid_data = players[['market_value_in_eur', 'age']].replace([np.inf, -np.inf], np.nan).dropna()
    korelacija, korel_p = pearsonr(valid_data['market_value_in_eur'], valid_data['age'])
    print("Korelacija između tržišne vrijednosti i starosti igrača:")
    print(f"Pearsonov koeficijent: {korelacija:.2f}, p-vrijednost: {korel_p:.4f}", endl)
    if korel_p < 0.05:
        print("Korelacija je statistički značajna.", endl)
    else:
        print("Korelacija nije statistički značajna.", endl)
```

Slika 33 Kod za korelacijsku analizu

Objašnjenje Pearsonove korelacije:

- Mjeri linearnu povezanost između dviju kontinuiranih varijabli
- Koeficijent korelacije (r) kreće se između -1 i +1
- $|r| < 0.3$: slaba korelacija
- $0.3 \leq |r| < 0.7$: umjerena korelacija
- $|r| \geq 0.7$: jaka korelacija

Provedena je inferencijalna analiza tržišne vrijednosti igrača. Izračunali smo 95% interval pouzdanosti za srednju vrijednost, što nam daje raspon u kojem se s velikom sigurnošću nalazi prava populacijska sredina. Također, koristili smo t-test za testiranje hipoteze da je prosječna tržišna vrijednost igrača veća od 5 milijuna eura. Dobivena p-vrijednost pokazuje postoji li značajna statistička razlika.



Slika 34 Korelacija starosti i tržišne vrijednosti

5. Treći projektni zadatak

Treći projektni zadatak iz kolegija *Uvod u analizu i vizualizaciju podataka* usmjeren je na primjenu znanja iz vizualizacije podataka pomoću alata Python i Tableau. Cilj zadatka bio je analizirati odabrani skup podataka te izraditi informativne i vizualno kvalitetne prikaze pomoću različitih vrsta grafova. Projekt se sastoji od tri dijela: pripreme podataka, izrade statičkih vizualizacija u Pythonu (koristeći Matplotlib i Seaborn), te izrade interaktivnih vizualizacija u alatu Tableau.

5.1. Priprema podataka

```python
import pandas as pd

# Popis izvornika i pripadajućih novih imena
csv_files = ["clubs.csv", "competitions.csv", "games.csv", "players.csv"]

for file in csv_files:
    # Učitaj CSV datoteku
    df = pd.read_csv(file)
    # Uzmi prvih 50 redova
    df_first50 = df.head(50)
    # Izradi novo ime datoteke (npr. games.csv -> games_uredeno.csv)
    new_filename = file.replace('.csv', '_uredeno.csv')
    # Spremi u novi CSV
    df_first50.to_csv(new_filename, index=False)
```

Slika 35 Kod za rezanje podataka

Funkcionalnost koda:

- **Uvoz pandas biblioteke** za rad s CSV datotekama
- **Definira se lista** izvornih datoteka koje želimo obraditi: clubs.csv, competitions.csv, games.csv, players.csv
- Za svaku datoteku u listi:
    - Datoteka se **učitava pomoću** pd.read_csv()
    - **Izvlači se prvih 50 redova** pomoću .head(50)
    - Kreira se **novo ime datoteke** gdje se .csv zamjenjuje s _uredeno.csv

- Novi uzorak se **sprema u novi CSV** koristeći to_csv(), bez indeksnog stupca (index=False)

5.2. Matplotlib

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from PIL import Image
import requests
from io import BytesIO

# Učitavanje podataka
clubs = pd.read_csv('clubs.csv')
players = pd.read_csv('players.csv')
competitions = pd.read_csv('competitions.csv')
games = pd.read_csv('games.csv')
```

Slika 36 Učitavanje podataka

Ovaj dio koda postavlja osnovu za daljnju analizu i vizualizaciju podataka:

- Prvo se uvoze sve biblioteke koje će omogućiti manipulaciju i prikaz podataka te rad sa slikama.
- Zatim se CSV datoteke učitavaju i pretvaraju u DataFrame-ove, koji se dalje koriste za izradu grafova, analiza i vizualizacija (npr. u Seaborn, Matplotlib ili Tableau).

```
igraci_klubovi = players.merge(
    clubs[['club_id', 'domestic_competition_id']],
    left_on='current_club_id',
    right_on='club_id',
    how='left'
).merge(
    competitions[['competition_id', 'country_name']],
    left_on='domestic_competition_id',
    right_on='competition_id',
    how='left'
)
```

Slika 37 Spajanje tablica da dobijemo zemlju kluba za svakog igrača

Ovo spajanje je ključno za daljnju analizu i vizualizaciju jer omogućuje, primjerice:

- prikaz broja domaćih i stranih igrača po zemlji
- analizu tržišne vrijednosti igrača po ligama ili državama
- usporedbe između država i klubova

```
# Kreiranje stupca za zemlju kluba i status igrača
igraci_klubovi['status'] = np.where(
    igraci_klubovi['country_of_birth'] == igraci_klubovi['country_name'],
    'Domaći igrač',
    'Strani igrač'
)
```

Slika 38 Kreiranje stupca za zemlju kluba i status igrača

Ovaj atribut (status) se kasnije koristi za vizualizaciju:

- Omogućuje **razdvajanje** domaćih i stranih igrača po ligama i zemljama
- Pomaže u analizi trendova – npr. koje zemlje imaju više stranih igrača u svojim klubovima

```
# Grupiranje po zemlji kluba i statusu
count_df = igraci_klubovi.groupby(['country_name', 'status']).size().unstack(fill_value=0)
count_df = count_df[['Domaći igrač', 'Strani igrač']]

# Filtriranje zemalja s više od 1000 igrača
count_df['Ukupno'] = count_df.sum(axis=1)
count_df_filtered = count_df[count_df['Ukupno'] > 1000].drop('Ukupno', axis=1)
```

Slika 39 Grupiranje i filtriranje podataka

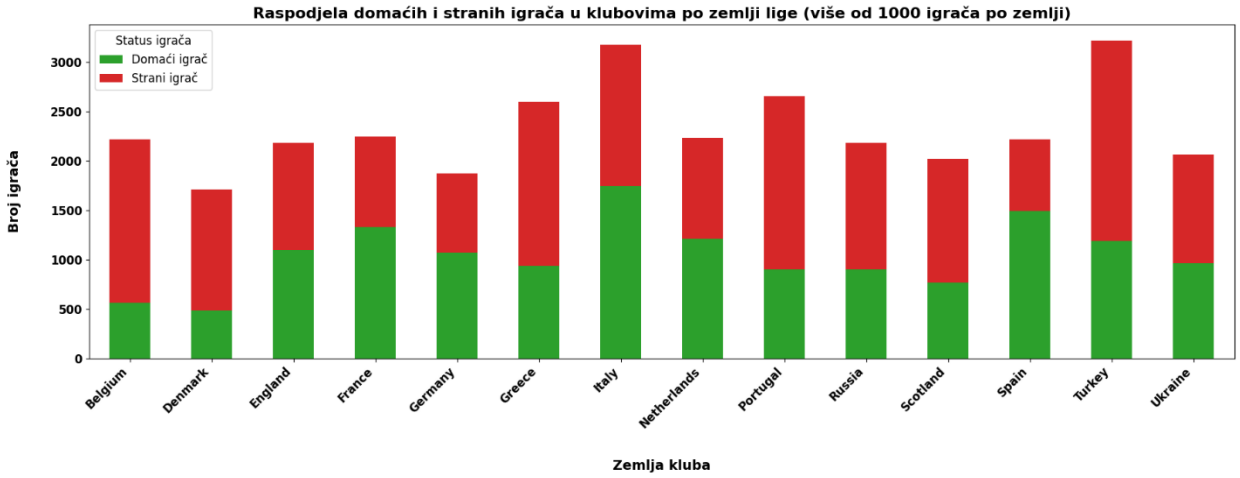
- **groupby(['country_name', 'status'])** – grupira podatke po zemlji lige i statusu igrača (Domaći igrač ili Strani igrač).
- **.size()** – broji koliko ima igrača u svakoj kombinaciji (npr. koliko domaćih u Njemačkoj, koliko stranih u Engleskoj itd.).
- **.unstack(fill_value=0)** – transformira "status" iz redaka u stupce za lakšu obradu i popunjava praznine s 0.
- Zatim se eksplicitno uzimaju samo stupci "Domaći igrač" i "Strani igrač" radi preglednosti.

- **`count_df.sum(axis=1)`** – zbraja domaće i strane igrače za svaku zemlju.
- Rezultat se sprema u novi stupac `"Ukupno"`.
- **`count_df[count_df['Ukupno'] > 1000]`** – filtrira samo one zemlje koje imaju **više od 1000 igrača ukupno**.
- **`.drop('Ukupno', axis=1)`** – nakon filtriranja, pomoćni stupac `"Ukupno"` se briše jer više nije potreban.

```
# Vizualizacija
ax = count_df_filtered.plot(
    kind='bar',
    stacked=True,
    color=['#2ca02c', '#d62728'],
    figsize=(18, 7)
)

plt.xlabel('Zemlja kluba', fontsize=14, fontweight='bold', labelpad=30)
plt.ylabel('Broj igrača', fontsize=14, fontweight='bold', labelpad=30)
plt.title('Raspodjela domaćih i stranih igrača u klubovima po zemlji lige (više od 1000 igrača po zemlji)', fontsize=16, fontweight='bold')
plt.legend(title='Status igrača', title_fontsize=12, fontsize=12, loc='upper left')
plt.xticks(rotation=45, ha='right', fontsize=12, fontweight='bold')
plt.yticks(fontsize=12, fontweight='bold')
plt.tight_layout()
plt.show()
```

Slika 40 Vizualizaciju

- plot(kind='bar') – kreira **stupčasti grafikon**
- stacked=True – domaći i strani igrači prikazani su kao **zbrojeni stupci**, tj. stacked bars (za bolju usporedbu ukupnog broja)
- color=['#2ca02c', '#d62728'] – definira boje: zelena za domaće, crvena za strane igrače
- figsize=(18, 7) – dimenzije slike su proširene za bolju čitljivost
- xlabel i ylabel definiraju oznake osi X i Y s povećanim fontom i podebljanim tekstom.
- labelpad dodaje prostor između oznake i osi.
- Naslov grafikona jasno opisuje sadržaj.
- Legenda objašnjava boje (domaći i strani igrač) i smještena je gore lijevo.
- xticks su rotirane za 45° kako bi se tekst država bolje prikazao.
- fontsize i fontweight poboljšavaju čitljivost.
- tight_layout() automatski prilagođava raspored elemenata kako bi se sve lijepo prikazalo.
- plt.show() prikazuje grafikon.



Slika 41 Raspodjela domaćih i stranih igrača

```
players['market_value_in_eur'] = pd.to_numeric(players['market_value_in_eur'], errors='coerce')

# Zamjena NaN s 0
players['market_value_in_eur'] = players['market_value_in_eur'].fillna(0).astype(int)

# Grupiranje i sumiranje
club_values = players.groupby('current_club_id')['market_value_in_eur'].sum()

# Spajanje s klubovima
clubs['total_market_value'] = clubs['club_id'].map(club_values).fillna(0).astype(int)

# sve ostale nedostajuće vrijednosti zamjenjujemo s "No data"
clubs = clubs.fillna('No data')
```

Slika 42 Spajanje i grupiranje

- Stupac market_value_in_eur u tablici igrača sadrži vrijednosti koje mogu biti tekstualne ili s greškama
- pd.to_numeric(..., errors='coerce') pretvara vrijednosti u brojeve, a sve što nije broj pretvara u NaN (nema vrijednosti)
- fillna(0) zamjenjuje NaN vrijednosti s nulom – važno za točne izračune
- astype(int) pretvara vrijednosti u cijele brojeve
- Grupira se po current_club_id (ID kluba u kojem igrač trenutno igra)
- Zbrajaju se tržišne vrijednosti svih igrača u tom klubu
- Rezultat: **serija s ukupnim tržišnim vrijednostima po klubovima**

- `map(club_values)` dodaje izračunatu tržišnu vrijednost iz `club_values` u tablicu `clubs`, temeljem `club_id`
- `fillna(0)` ponovno osigurava da klubovi bez dostupnih igrača imaju tržišnu vrijednost 0
- `astype(int)` pretvara sve vrijednosti u cijele brojeve.
- .fillna() - Ova naredba zamjenjuje sve NaN vrijednosti u preostaloj clubs tablici s tekstom "No data" – čistoća podataka za prikaz i analizu

```
# Odabir najvrijednijih klubova
top10_clubs = clubs.sort_values('total_market_value', ascending=False).head(10)
top10_ids = top10_clubs['club_id'].astype(float).tolist()
club_names = top10_clubs.set_index('club_id')['name'].to_dict()

# Prikupljanje golova za svaki klub
def get_goals_for_club(club_id):
    home_goals = games[games['home_club_id'] == club_id]['home_club_goals']
    away_goals = games[games['away_club_id'] == club_id]['away_club_goals']
    return pd.concat([home_goals, away_goals])
```

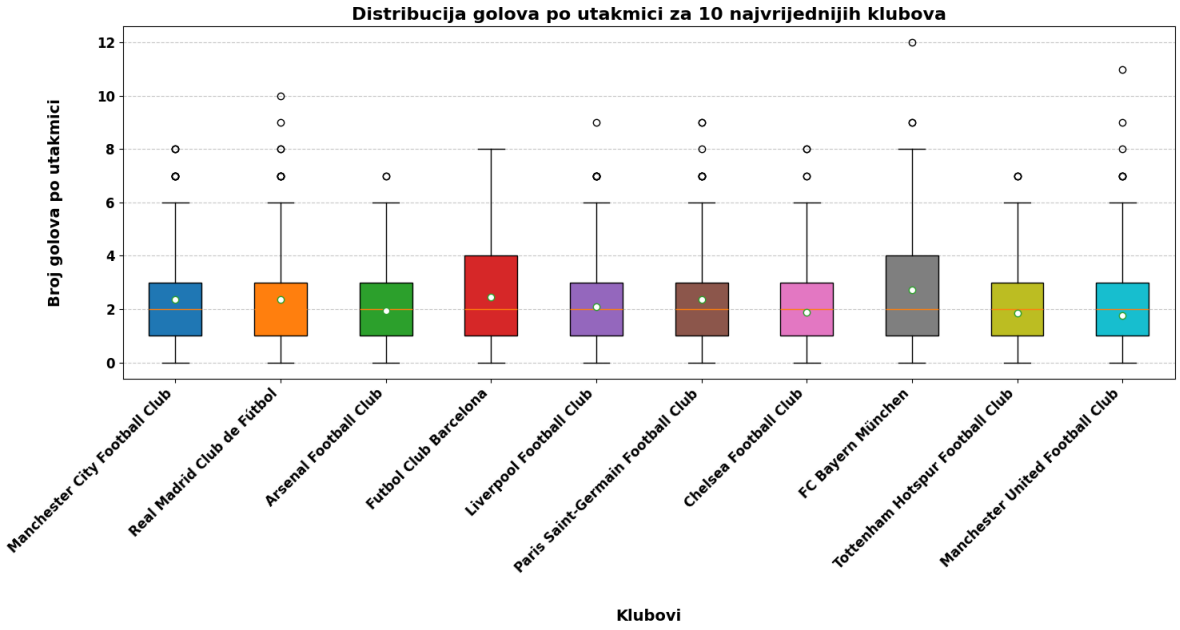
Slika 43 Obrada podataka

- Sortira klubove po stupcu total_market_value silazno (od najveće do najmanje vrijednosti).
- head(10) uzima prvih 10 – dakle, **10 najvrjednijih klubova**.
- Izvlače se club_id vrijednosti tih klubova.
- astype(float) osigurava kompatibilnost s podacima u drugim tablicama.
- .tolist() pretvara seriju u Python listu ([ID1, ID2, ...]).
- Postavlja club_id kao indeks i izdvaja imena klubova.
- to_dict() pretvara to u rječnik oblika {club_id: club_name}.
- Ovo se koristi za brzu identifikaciju klubova kod kasnijih analiza ili prikaza.
- Funkcija get_goals_for_club prima club_id kao argument.
- Pronalazi sve utakmice u kojima je taj klub bio domaćin i izvlači broj postignutih golova (home_club_goals).
- Zatim pronalazi utakmice u kojima je taj klub bio gost i izvlači away_club_goals.
- pd.concat([...]) spaja te dvije serije u jednu.

```
# Priprema podataka za boxplot
plt.figure(figsize=(15, 8))
data = []
labels = []

for club_id in top10_ids:
    goals = get_goals_for_club(club_id)
    if not goals.empty:
        data.append(goals)
        labels.append(club_names[club_id])
```

Slika 44 Priprema podataka

- plt.figure(figsize=(15, 8)) kreira prostor za vizualizaciju s definiranim dimenzijama (širina 15, visina 8).
- data – lista koja će sadržavati nizove golova po klubovima.
- labels – lista koja će sadržavati **nazive klubova** (za X-os dijagrama).
- za svaki klub u top10_ids poziva prethodno definirana funkcija get_goals_for_club, koja vraća sve golove koje je klub postigao (kod kuće i u gostima).
- Provjerava se je li goals serija prazna (goals.empty).
- Ako nije prazna:
    - Golovi se dodaju u data listu.
    - Naziv kluba se dodaje u labels listu (za prikaz ispod svakog boxa).



Slika 45 Golovi po 10 najvrijednijih klubova

```
# Filtriranje glavnih liga
glavne_lige = competitions[competitions['is_major_national_league'] == True]

# Spajanje klubova s ligama
klubovi_lige = clubs.merge(
    glavne_lige[['competition_id', 'name']].rename(columns={'name': 'league_name'}),
    left_on='domestic_competition_id',
    right_on='competition_id'
)

# Spajanje igrača s klubovima i ligama
igraci_lige = players.merge(
    klubovi_lige[['club_id', 'league_name']],
    left_on='current_club_id',
    right_on='club_id'
)
```

Slika 46 Filtriranje i spajanje

- Izdvaja se samo **one lige koje su označene kao "glavne nacionalne lige"**.
- U stupcu is_major_national_league nalaze se logičke vrijednosti (True/False), a filtriranjem se zadržavaju samo one koje su True.
- Spajaju se klubovi s prethodno filtriranim ligama pomoću **vanjskog ključa**:
- clubs.domestic_competition_id povezuje se s competitions.competition_id.
- Prije spajanja, stupac name iz competitions se preimenuje u league_name radi jasnoće.

- Dobiva se nova tablica klubovi_lige u kojoj svaki klub sada ima pridruženo ime lige u kojoj igra.
- Igrači se povezuju s informacijama o ligama, preko kluba za koji trenutno igraju.
- players.current_club_id povezuje se s klubovi_lige.club_id.
- Time se svakom igraču može pridružiti ime lige (league_name) u kojoj njegov klub igra.

```
# Pronalaženje najvrijednijeg igrača po ligi
najvrijedniji_po_ligi = igraci_lige.groupby('league_name').apply(
    lambda x: x.nlargest(1, 'market_value_in_eur')
).reset_index(drop=True)

# Pretvaranje vrijednosti na način da originalnunu vrijednost podijelimo s 1.000.000 radi preglednosti
najvrijedniji_po_ligi['market_value_millions'] = najvrijedniji_po_ligi['market_value_in_eur'] / 1_000_000

# Veliko slovo na početku imena lige
najvrijedniji_po_ligi['league_name'] = najvrijedniji_po_ligi['league_name'].str.capitalize()

# Generiranje random boja za svaki bar
np.random.seed(np.random.randint(0, 100))
colors = np.random.rand(len(najvrijedniji_po_ligi), 3)

plt.figure(figsize=(15, 8))
bars = plt.bar(
    najvrijedniji_po_ligi['league_name'],
    najvrijedniji_po_ligi['market_value_millions'],
    color=colors
)
```

Slika 47 Obrada podataka

- Grupa se po league_name – dakle, po svakoj ligi.
- x.nlargest(1, ...) vraća **igrača s najvećom tržišnom vrijednošću** unutar svake lige.
- .reset_index(drop=True) poništava višestruke indekse nastale grupiranjem i vraća čist DataFrame.
- Umjesto npr. "120000000 €", vrijednosti se pretvaraju u **milijune eura** – npr. "120.0".
- Poboljšanje izgleda vizualizacije – nazivi liga počinju velikim slovom (npr. "Premier league" → "Premier League").
- Generira se slučajni "seed" za boje – svako pokretanje daje različitu kombinaciju.
- np.random.rand(n, 3) stvara RGB vrijednosti za n različitih boja (koliko ima liga).
- Kreira se bar dijagram (`plt.bar`) gdje je:
  - X-os: naziv lige
  - Y-os: tržišna vrijednost najskupljeg igrača (u milijunima €)
  - Boje: slučajno generirane

```python
# Funkcija za dodavanje slike igrača
def dodaj_sliku(url, pozicija):
    try:
        response = requests.get(url, timeout=5)
        img = Image.open(BytesIO(response.content))
        img = img.resize((100, 100))
        return OffsetImage(img, zoom=0.4)
    except:
        return None
```

Slika 48 Dodavanje slika igrača

- url: poveznicu na sliku igrača (najčešće iz web izvora),
- pozicija: koordinatu gdje će slika biti prikazana u grafu (nije direktno korišteno unutar funkcije, ali se koristi kasnije pri pozicioniranju slike).

- Slika se preuzima s interneta pomoću requests.get.

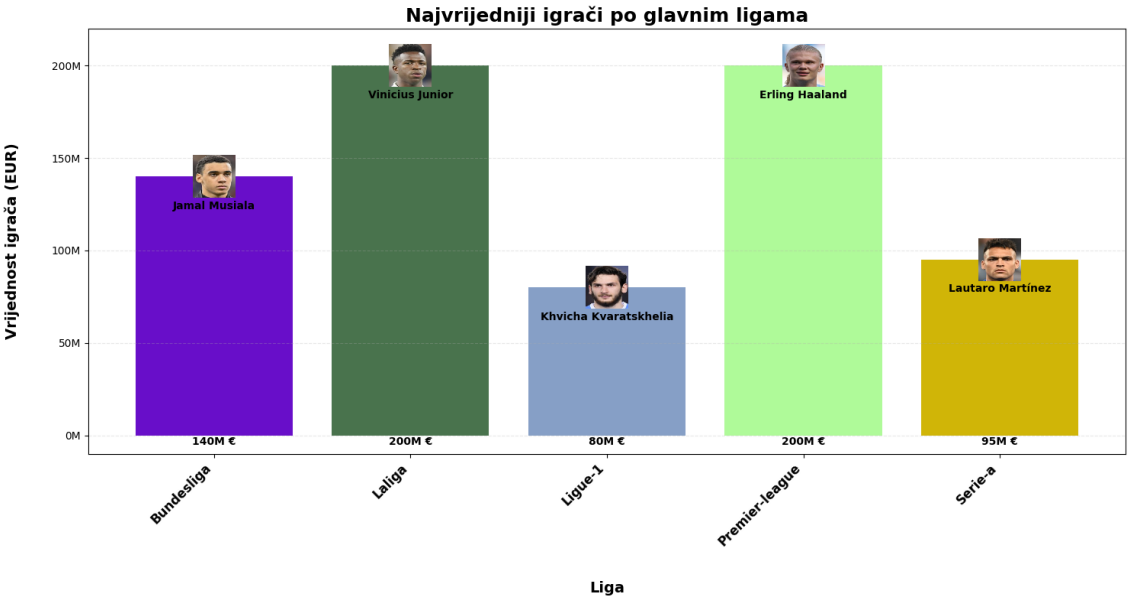
- timeout=5 znači da će pokušaj dohvaćanja trajati najviše 5 sekundi – radi sigurnosti da se ne "zaglavi" kod.

- Dohvaćeni sadržaj (response.content) se pretvara u sliku pomoću PIL.Image biblioteke.
- Slika se **skalira** na veličinu 100x100 piksela – dovoljno da se vidi, a da ne zauzima previše prostora u grafu.
- Funkcija vraća OffsetImage objekt, koji omogućava da se slika **grafički postavi točno na željenu lokaciju u matplotlib grafu**.
- Ako preuzimanje slike **ne uspije** (npr. zbog lošeg URL-a, problema s mrežom ili formatom slike) – funkcija **ne vraća ništa** (None) kako ne bi prekinula izvršavanje koda.

```
# Dodavanje slika i imena igrača
for bar, (_, row) in zip(bars, najvrijedniji_po_ligi.iterrows()):
    image = dodaj_sliku(row['image_url'], bar.get_x())
    if image:
        plt.gca().add_artist(AnnotationBbox(
            image,
            (bar.get_x() + bar.get_width()/2, bar.get_height()),
            frameon=False
        ))


    plt.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height() - 13,
        row['name'],
        ha='center',
        va='top',
        color='black',
        fontweight='bold',
        fontsize=10
    )

    # Vrijednost na početku bara (iznad x-osi)
    plt.text(
        bar.get_x() + bar.get_width()/2,
        -5,
        f'{row["market_value_millions"]:.0f}M €',
        ha='center',
        color='black',
        fontweight='bold'
    )
```

Slika 49 Dodavanje slika i imena

- bars: svaki stupac na grafu,
- row: red u tablici najvrijedniji_po_ligi, koji sadrži podatke o igraču (ime, URL slike, tržišna vrijednost itd.).
- Funkcija dodaj_sliku dohvaća i priprema sliku iz URL-a (ranije definirano).
- Ako je uspješno dohvaćena, dodaje se na graf preko AnnotationBox.
- Pozicionira se točno **iznad sredine stupca** (X-koordinata) i na vrh stupca (Y-koordinata).
- plt.text dodaje **tekstualni element** s imenom igrača.
- Pozicija je **ispod slike**, tj. malo ispod vrha stupca (-13 piksela).
- Tekst je centriran i podebljan (bold), crne boje.
- Prikazuje se vrijednost igrača u milijunima eura (npr. 85M €).
- Postavlja se na X-sredinu bara, ali na fiksnoj visini y = 5, tj. tik iznad baze.
- Font je crn i podebljan.



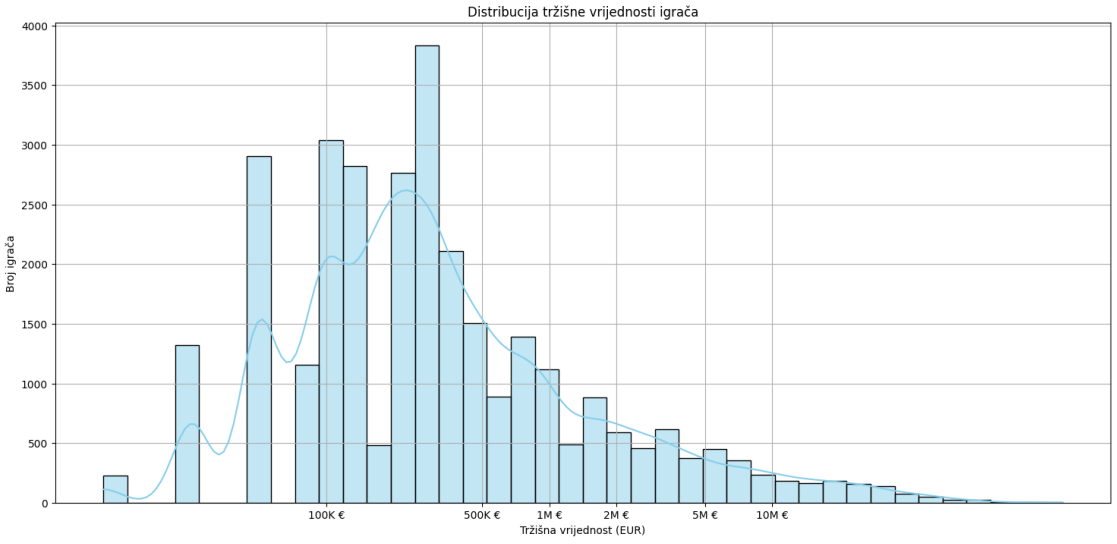
Slika 50 Najvrijedniji igrači po glavnim ligama

5.3. Seaborn

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.ticker as mticker


vrijednosti = players['market_value_in_eur']
vrijednosti = vrijednosti[vrijednosti > 0]

log_vrijednosti = np.log10(vrijednosti)

tick_values = [5, 5.7, 6, 6.3, 6.7, 7]  # odgovara 100k, 500k, 1M, 2M, 5M, 10M
tick_labels = ['100K €', '500K €', '1M €', '2M €', '5M €', '10M €']

plt.figure(figsize=(10, 6))
sns.histplot(log_vrijednosti, kde=True, bins=40, color='skyblue')

plt.title('Distribucija tržišne vrijednosti igrača')
plt.xlabel('Tržišna vrijednost (EUR)')
plt.ylabel('Broj igrača')

plt.xticks(tick_values, tick_labels)

plt.grid(True)
plt.tight_layout()
plt.show()
```

Slika 51 Seaborn

- Uzima se stupac market_value_in_eur iz tablice players i uklanjaju se sve vrijednosti koje su 0 ili manje.
- Tržišne vrijednosti variraju od 100.000 € do 100.000.000 €, što je **veliki raspon**. Log-transformacija (log10) se koristi kako bi:
- normalizirala distribuciju,
- bolje prikazala podatke u histogramu.
- X-osi se dodjeljuju vrijednosti koje odgovaraju:
    - 10^5 = 100.000 €
    - 10^6 = 1.000.000 €
    - 10^7 = 10.000.000 €
- Koristi se sns.histplot za crtanje histograma:
- kde=True dodaje **glatku krivulju gustoće** preko histograma,
- bins=40 znači da se koristi 40 stupaca,
- color='skyblue' određuje boju ispune.
- Naslovi osi i grafa jasno prikazuju što vizualizacija predstavlja.
- Prikazuje oznake na x-osi (koja je log-transformirana) u **čitljivom formatu** (100K €, 1M €, itd.). Dodaje mrežu radi bolje čitljivosti i prikazuje finalni graf.

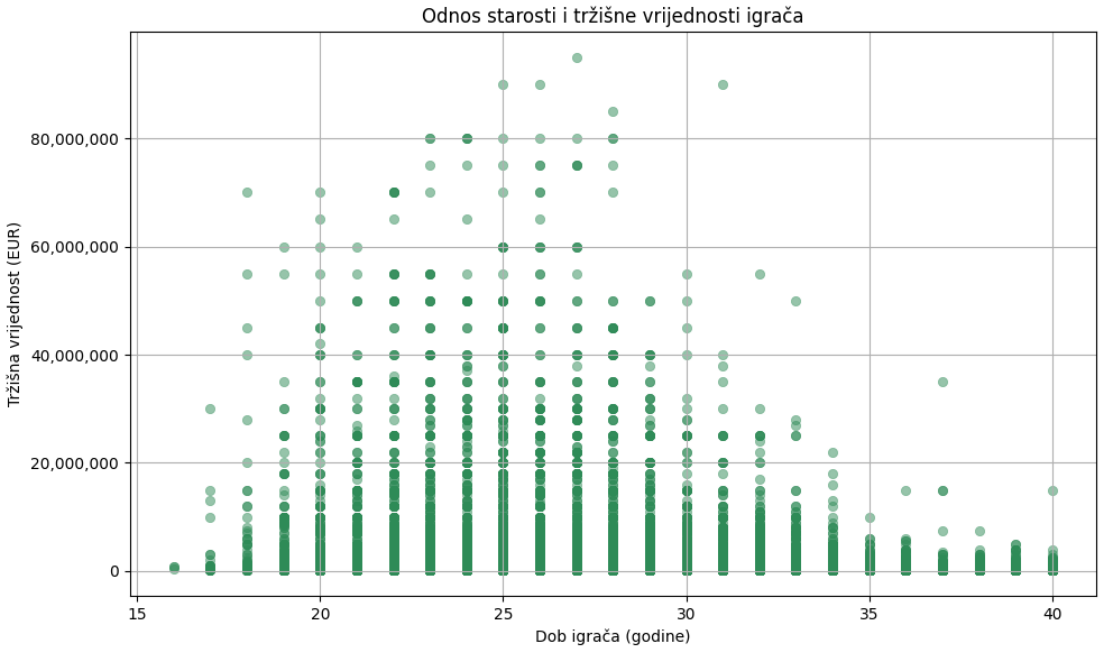


Slika 52 Distribucija tržišne vrijednosti igrača

```
players['date_of_birth'] = pd.to_datetime(players['date_of_birth'], errors='coerce')
today = pd.to_datetime('today')
players['age'] = (today - players['date_of_birth']).dt.days // 365

df_filtered = players[
    (players['market_value_in_eur'] > 100_000) &
    (players['market_value_in_eur'] < 100_000_000) &
    (players['age'] >= 16) &
    (players['age'] <= 40)
]

plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df_filtered,
    x='age',
    y='market_value_in_eur',
    alpha=0.5,
    edgecolor=None,
    color='seagreen'
)

plt.title('Odnos starosti i tržišne vrijednosti igrača')
plt.xlabel('Dob igrača (godine)')
plt.ylabel('Tržišna vrijednost (EUR)')
plt.ticklabel_format(style='plain', axis='y')
plt.gca().yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x):,}'))
plt.grid(True)
plt.tight_layout()
plt.show()
```

Slika 53 Odnos starosti i tržišne vrijednosti

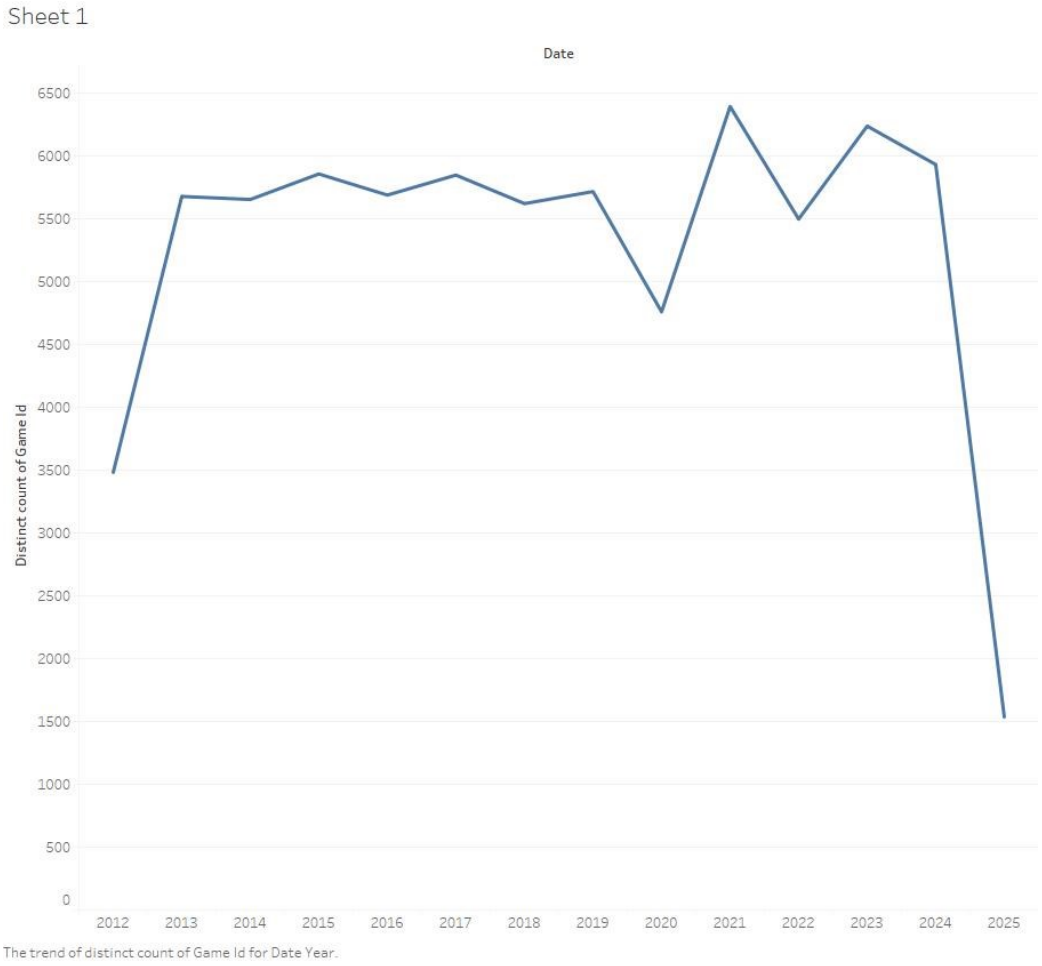
- Preračunava se date_of_birth u starost u godinama:
    - Koristi se trenutni datum (today),
    - Starost se izračunava kao razlika u danima / 365,
    - Rezultat se sprema u novi stupac 'age'.
- Igrači čija je tržišna vrijednost između **100.000 € i 100.000.000 €**
- Dob između **16 i 40 godina** – uobičajeni raspon profesionalnih nogometaša.
- Scatterplot koristi:
    - x = age (dob)
    - y = market_value_in_eur (tržišna vrijednost)
    - providnost (alpha=0.5) smanjuje “preklapanje” točaka
    - zelena boja (seagreen) bez obruba (edgecolor=None)
- Jasan naslov i označene osi.
- Tržišna vrijednost se prikazuje s tisućicama (npr. 1,000,000 umjesto 1000000), što poboljšava čitljivost.



Slika 54 Odnos starosti i tržišne vrijednosti igrača

5.4. Tableau

5.4.1. Linijski dijagram



Slika 55 Broj jedinstvenih utakmica po godini

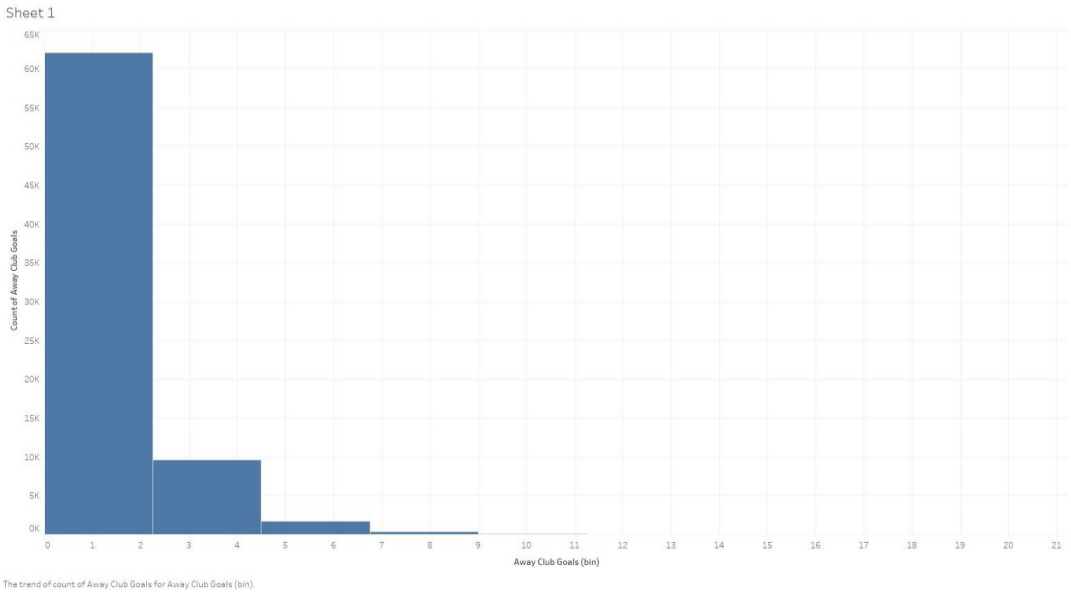
Vizualizacija: Linijski dijagram
Opis:
Ova vizualizacija prikazuje broj jedinstvenih utakmica (distinct count of Game Id) po godini (Date).

- **x-osa**: Godina (od 2012. do 2025.).
- **y-osa**: Broj različitih utakmica odigranih u svakoj godini.

Zaključak:

- Od 2013. do 2024. broj utakmica je relativno stabilan, s manjim oscilacijama.
- Najveći broj utakmica zabilježen je 2021.
- Godina 2025. pokazuje drastičan pad jer najvjerojatnije podaci za tu godinu još nisu kompletirani (npr. sezona je još u tijeku).

5.4.2. Histogram



Slika 56 Histogram postignutih golova gostojućih ekipa

Vizualizacija: Histogram
Opis:
Ova vizualizacija prikazuje distribuciju broja postignutih golova od strane gostujućih ekipa (Away Club Goals), podijeljeno u binove (grupe vrijednosti)

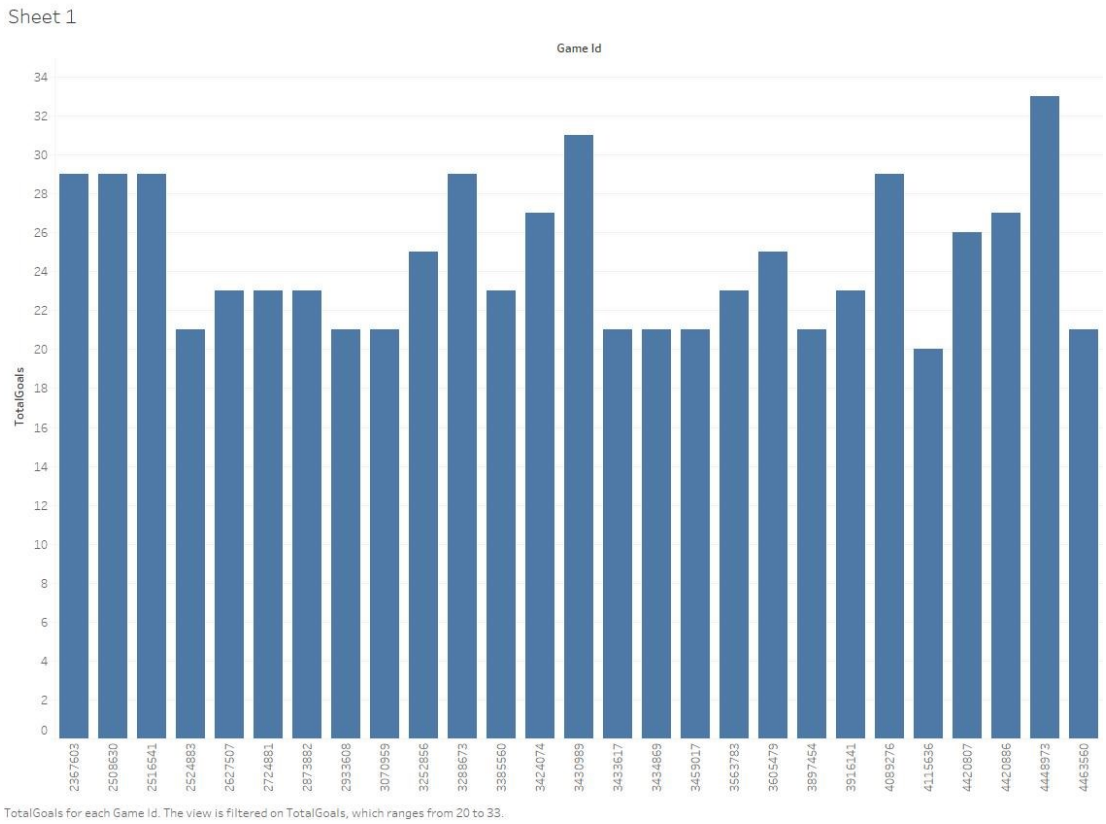
- **x-osa**: Broj postignutih golova u utakmici (npr. 0, 1, 2...)
- **y-osa**: Broj utakmica u kojima su gostujuće ekipe postigle navedeni broj golova

Zaključak:

- Većina utakmica završava s 0 do 2 gola postignutih od strane gostujuće ekipe

- Vrlo je rijetko da gostujuće ekipe postignu više od 4-5 golova, što je vidljivo iz naglog pada broja pojavljivanja za više golova

5.4.3. Vertikalni stupičasti dijagram



Slika 57 Broj golova po utakmici

Vizualizacija: Vertikalni stupčasti dijagram
Opis:
Ova vizualizacija prikazuje broj ukupnih golova postignutih u pojedinačnim utakmicama, gdje je svaka traka (stupac) označena jedinstvenim identifikatorom utakmice (Game Id).

- Na **x-osi** nalaze se ID-jevi utakmica.
- Na **y-osi** se prikazuje broj golova (TotalGoals) za svaku utakmicu.
- Filtrirano je tako da uključuje samo utakmice koje su imale **između 20 i 33 gola**, što sugerira fokus na najefikasnije utakmice.

Zaključak:
Ova analiza pomaže u identifikaciji najefikasnijih utakmica s najviše postignutih golova, što može biti korisno za analizu napadačkih performansi timova.

6. Zaključak

Kroz ova tri projektna zadatka detaljno smo analizirali skup podataka *Football Data from Transfermarkt.* Navedeni skup podataka izrazito je detaljan, što nam je omogućilo da bez problema odaberemo koje ćemo dijelove analizirati. Rješavanjem zadataka saznali smo koje sve mogućnosti nudi programski jezik *Python* zajedno s svojim bibliotekama kao što su *Numpy, Pandas, Matplolib* i drugi. Pomoću funkcija iz navedenih biblioteka uspješno smo prikupili, pročistili, proanalizirali te na kraju vizualizirali podatke. Na kraju smo izvršili analizu rezultata same analize te smo zaključili kako svi dobiveni rezultati imaju smisla te smo se dodatno uvjerili kako su točni.

Smatramo kako znanje stečeno na ovome kolegiju je izrazito korisno, ali i važno, ne samo iz perspektive programera, već i iz perspektive analitičara. Posebno nam se dojmilo što smo imali priliku birati skup podataka koji nam je blizak i zanimljiv, što je učinilo proces svladavanja gradiva dodatno zanimljivijim te učinkovitijim. Česti susreti s vizualizacijom podataka (grafovi) te s upotrebom statističkih metoda i ispitivanja svakako će nam biti korisni tokom profesionalne karijere.

7. Popis slika