축구 선수의 기록과 몸값의 상관 관계

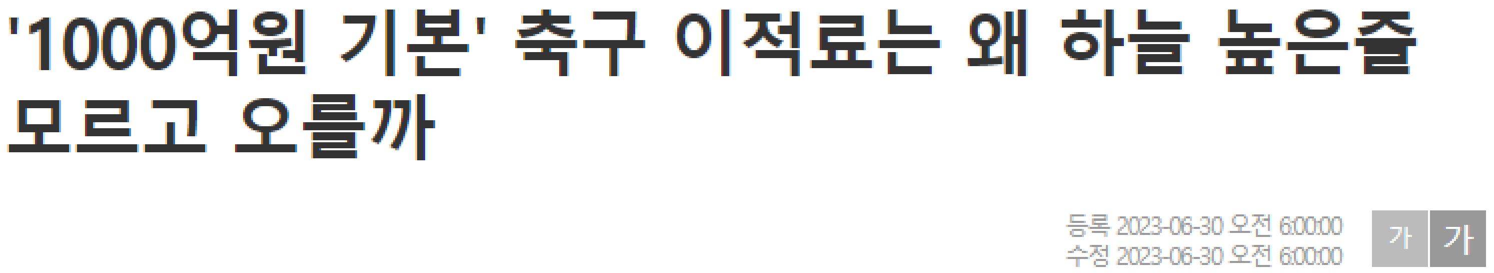
제주대학교
컴퓨터공학전공
2018108257 남기윤

목차

1. 개요 및 필요성

- 평소 축구에 관심의 많은 저의 호기심

- 천정부지로 오르는 축구 이적료 왜 오르고, 먹튀를 가릴 수 있나?



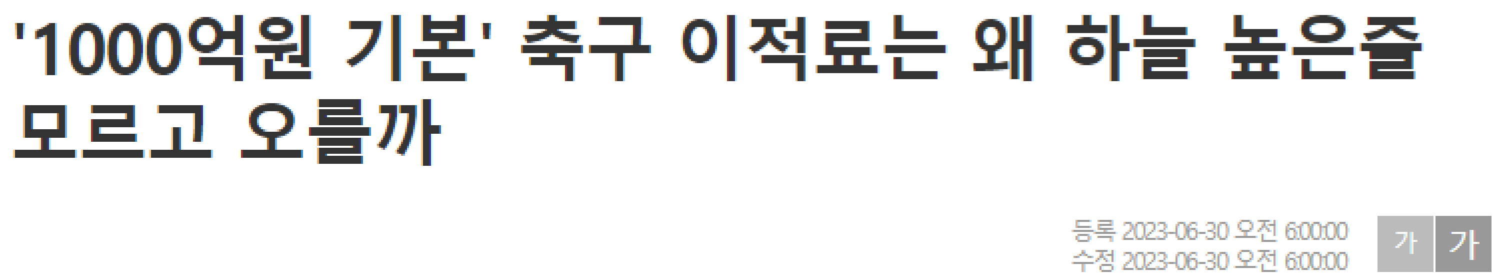
'1000억원 기본' 축구 이적료는 왜 하늘 높은줄 모르고 오를까

등록 2023-06-30 오전 6:00:00
수정 2023-06-30 오전 6:00:00

1. 개요 및 필요성

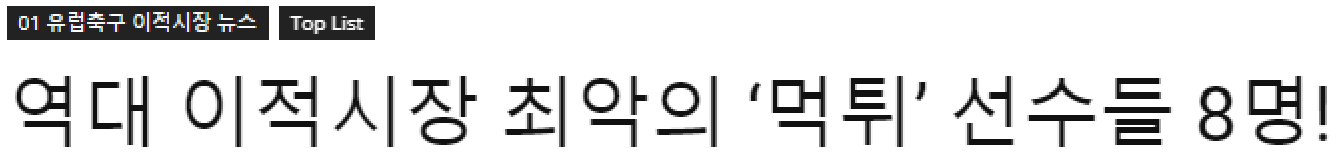
- 평소 축구에 관심의 많은 저의 호기심

- 천정부지로 오르는 축구 이적료 왜 오르고, 먹튀를 가릴 수 있나?



'1000억원 기본' 축구 이적료는 왜 하늘 높은줄 모르고 오를까

등록 2023-06-30 오전 6:00:00
수정 2023-06-30 오전 6:00:00



01 유럽축구 이적시장 뉴스 Top List

역대 이적시장 최악의 '먹튀' 선수들 8명!

2. 관련 내용

‘축구 이적료는 왜 오를까?’에 대한 기사 내용 중 일부

오늘날 세계 축구를 이끄는 ‘축구의 신’ 리오넬 메시(인터 마이애미)는 생각보다 이적가치가 낮다. 독일 통계전문사이트 ‘트랜스퍼마크트’가 평가한 예상 이적료는 3500만유로(약 502억원).

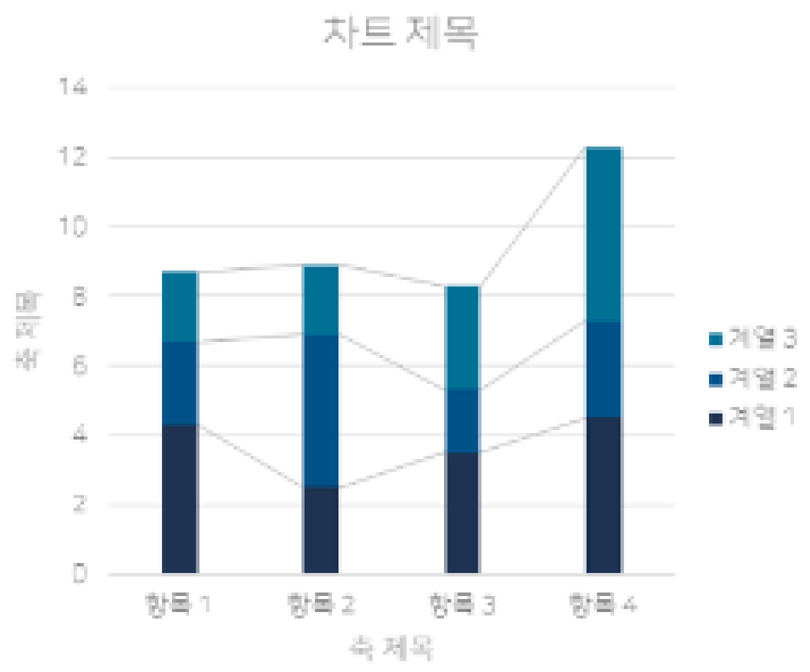
결코 적은 금액은 아니지만, 메시의 실력이나 이름값에는 걸맞지 않다. 만 36살에 이르는 나이 때문이다. 같은 실력이라도 30살 선수보다 22살 선수가 훨씬 이적가치가 높다.

2. 관련 내용

이적료 먹튀에 대한 기사 내용 중 일부

해마다 이적시장의 규모는 해가 다르게 커져가고 있고, 선수들의 이적료는 하늘 높은줄 모르고 치솟고 있다. 거액의 이적료를 지불하고 영입한 선수들은 팬들이 열렬한 환영을 받곤 하지만, 그렇게 거대한 이적료가 때때로는 선수들의 어깨를 짓누르는 부담으로 작용하기도 한다. 그런 부담이 부진으로 이어지는 경우도 있고, 거기에 자칫 부상이라도 입는다면 그를 영입한 클럽은 막대한 손해를 입기도 한다.

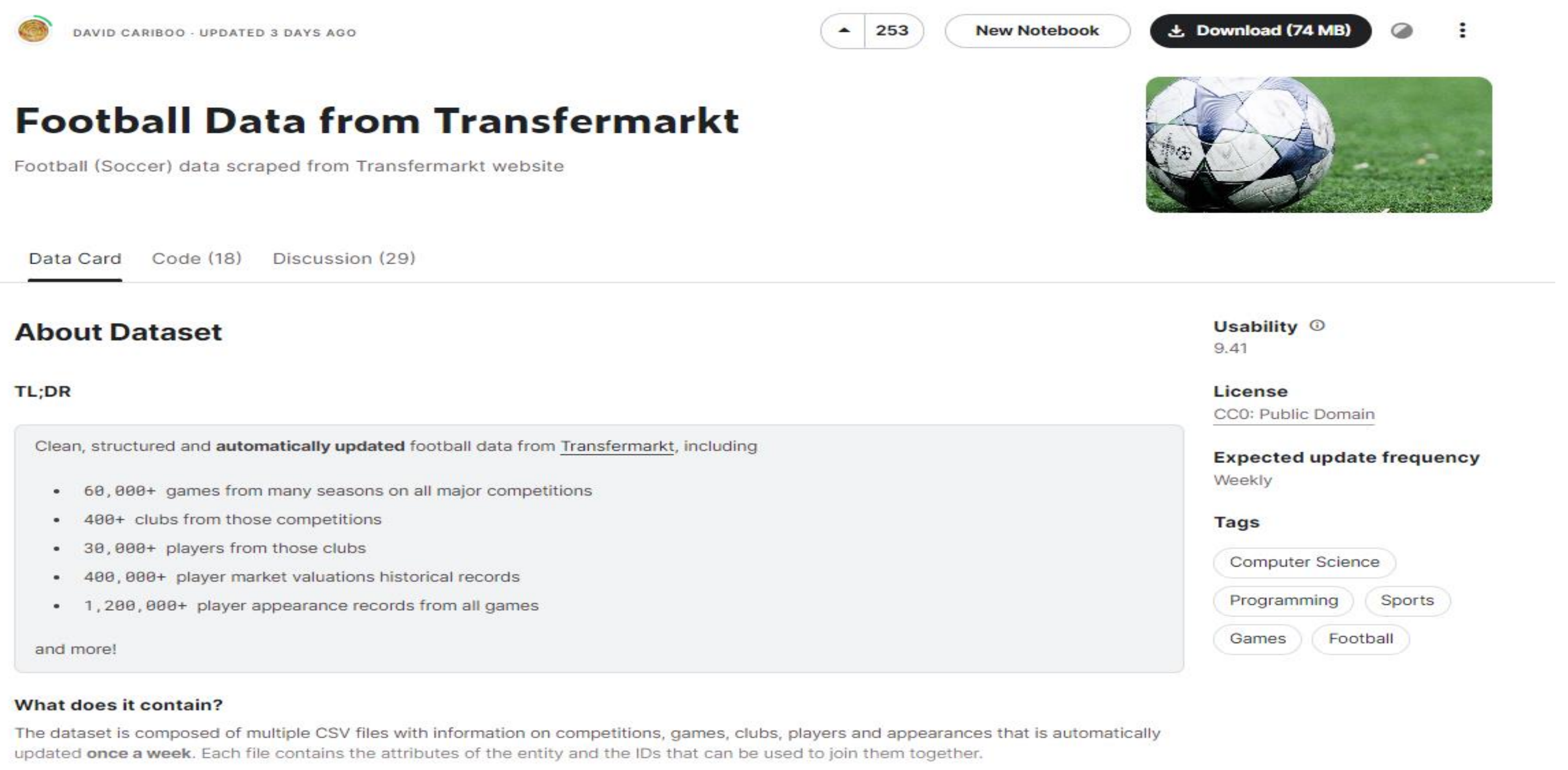
3. 기대 효과



사적인 호기심 해결 가능

팀 관리 및 스카우팅에 유용

4. 데이터셋



Football Data from Transfermarkt

Football (Soccer) data scraped from Transfermarkt website

About Dataset

TL;DR

Clean, structured and **automatically updated** football data from Transfermarkt, including

- `60,000+` games from many seasons on all major competitions
- `400+` clubs from those competitions
- `30,000+` players from those clubs
- `400,000+` player market valuations historical records
- `1,200,000+` player appearance records from all games

and more!

What does it contain?

The dataset is composed of multiple CSV files with information on competitions, games, clubs, players and appearances that is automatically updated **once a week**. Each file contains the attributes of the entity and the IDs that can be used to join them together.

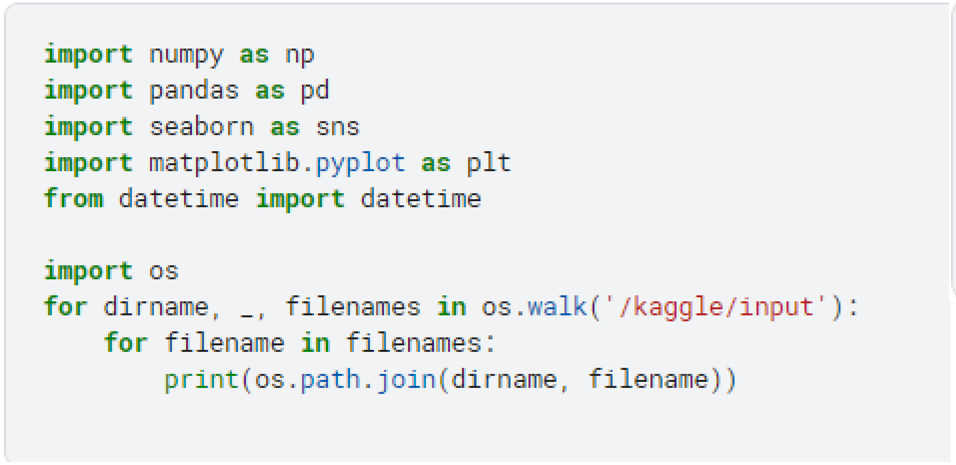
Usability
9.41

License
CC0: Public Domain

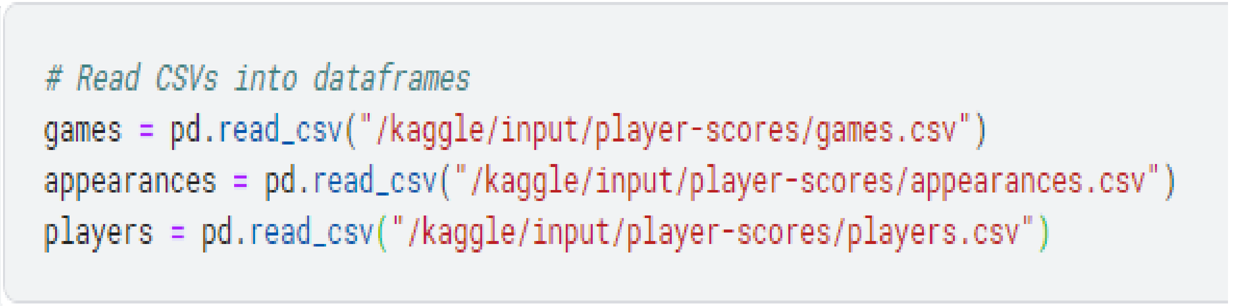
Expected update frequency
Weekly

Tags

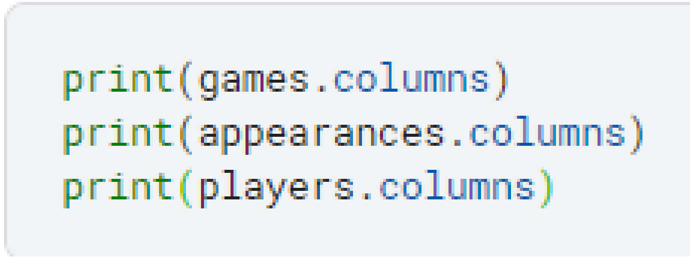
5. 데이터 분석



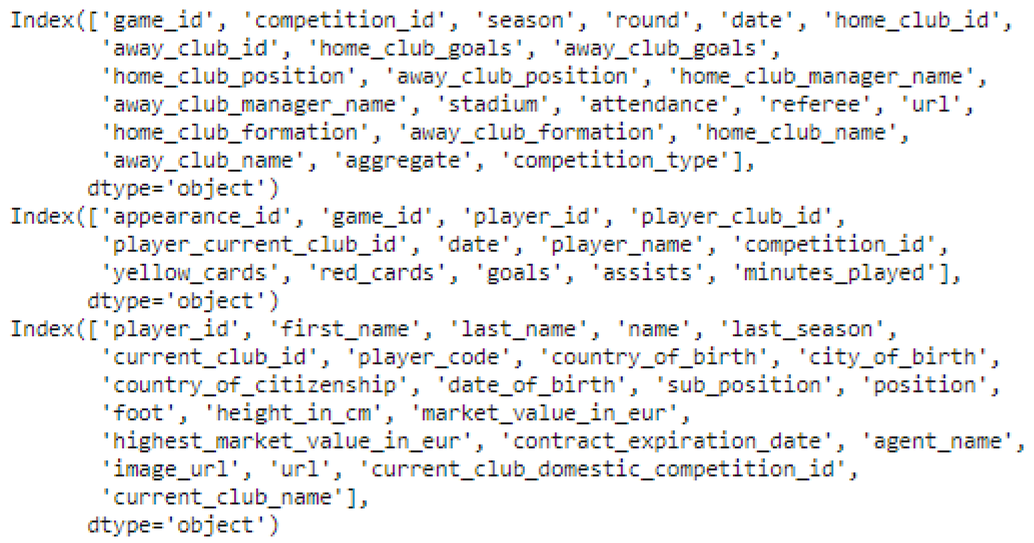
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```



```python
# Read CSVs into dataframes
games = pd.read_csv("/kaggle/input/player-scores/games.csv")
appearances = pd.read_csv("/kaggle/input/player-scores/appearances.csv")
players = pd.read_csv("/kaggle/input/player-scores/players.csv")
```

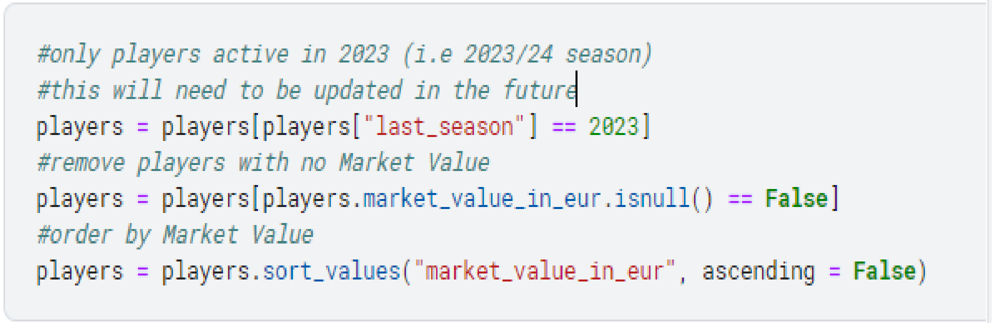


```python
print(games.columns)
print(appearances.columns)
print(players.columns)
```

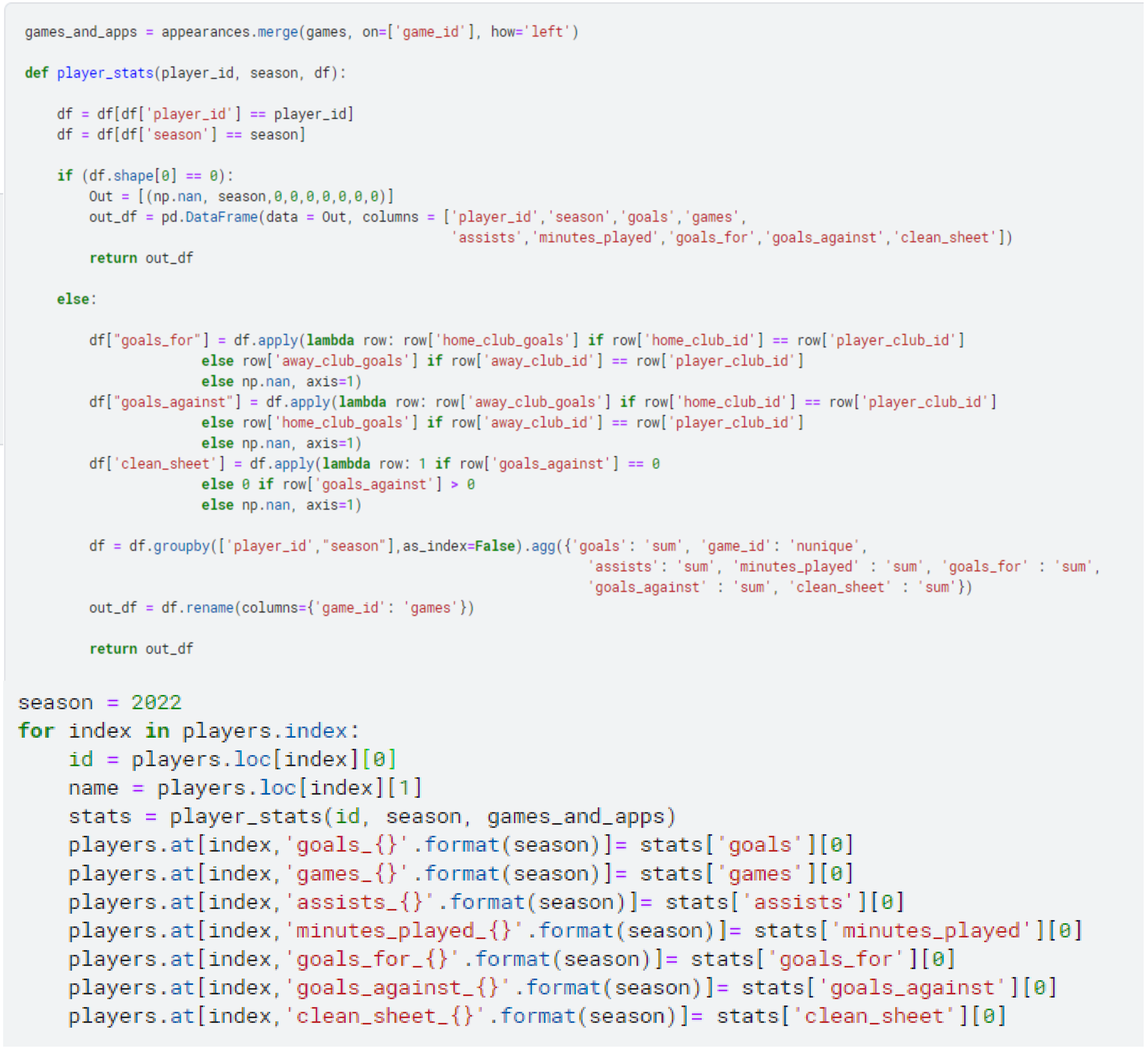


```
Index(['game_id', 'competition_id', 'season', 'round', 'date', 'home_club_id',
       'away_club_id', 'home_club_goals', 'away_club_goals',
       'home_club_position', 'away_club_position', 'home_club_manager_name',
       'away_club_manager_name', 'stadium', 'attendance', 'referee', 'url',
       'home_club_formation', 'away_club_formation', 'home_club_name',
       'away_club_name', 'aggregate', 'competition_type'],
      dtype='object')
Index(['appearance_id', 'game_id', 'player_id', 'player_club_id',
       'player_current_club_id', 'date', 'player_name', 'competition_id',
       'yellow_cards', 'red_cards', 'goals', 'assists', 'minutes_played'],
      dtype='object')
Index(['player_id', 'first_name', 'last_name', 'name', 'last_season',
       'current_club_id', 'player_code', 'country_of_birth', 'city_of_birth',
       'country_of_citizenship', 'date_of_birth', 'sub_position', 'position',
       'foot', 'height_in_cm', 'market_value_in_eur',
       'highest_market_value_in_eur', 'contract_expiration_date', 'agent_name',
       'image_url', 'url', 'current_club_domestic_competition_id',
       'current_club_name'],
      dtype='object')
```

5. 데이터 분석



```python
#only players active in 2023 (i.e 2023/24 season)
#this will need to be updated in the future
players = players[players["last_season"] == 2023]
#remove players with no Market Value
players = players[players.market_value_in_eur.isnull() == False]
#order by Market Value
players = players.sort_values("market_value_in_eur", ascending = False)
```



```python
games_and_apps = appearances.merge(games, on=['game_id'], how='left')

def player_stats(player_id, season, df):

    df = df[df['player_id'] == player_id]
    df = df[df['season'] == season]

    if (df.shape[0] == 0):
        Out = [(np.nan, season,0,0,0,0,0,0,0)]
        out_df = pd.DataFrame(data = Out, columns = ['player_id','season','goals','games',
                                                     'assists','minutes_played','goals_for','goals_against','clean_sheet'])
        return out_df

    else:

        df["goals_for"] = df.apply(lambda row: row['home_club_goals'] if row['home_club_id'] == row['player_club_id']
                       else row['away_club_goals'] if row['away_club_id'] == row['player_club_id']
                       else np.nan, axis=1)
        df["goals_against"] = df.apply(lambda row: row['away_club_goals'] if row['home_club_id'] == row['player_club_id']
                       else row['home_club_goals'] if row['away_club_id'] == row['player_club_id']
                       else np.nan, axis=1)
        df['clean_sheet'] = df.apply(lambda row: 1 if row['goals_against'] == 0
                       else 0 if row['goals_against'] > 0
                       else np.nan, axis=1)

        df = df.groupby(['player_id',"season"],as_index=False).agg({'goals': 'sum', 'game_id': 'nunique',
                                                                     'assists': 'sum', 'minutes_played' : 'sum', 'goals_for' : 'sum',
                                                                     'goals_against' : 'sum', 'clean_sheet' : 'sum'})
        out_df = df.rename(columns={'game_id': 'games'})

        return out_df
```

```python
season = 2022
for index in players.index:
    id = players.loc[index][0]
    name = players.loc[index][1]
    stats = player_stats(id, season, games_and_apps)
    players.at[index,'goals_{}'.format(season)]= stats['goals'][0]
    players.at[index,'games_{}'.format(season)]= stats['games'][0]
    players.at[index,'assists_{}'.format(season)]= stats['assists'][0]
    players.at[index,'minutes_played_{}'.format(season)]= stats['minutes_played'][0]
    players.at[index,'goals_for_{}'.format(season)]= stats['goals_for'][0]
    players.at[index,'goals_against_{}'.format(season)]= stats['goals_against'][0]
    players.at[index,'clean_sheet_{}'.format(season)]= stats['clean_sheet'][0]
```

5. 데이터 분석

```python
# Calculate the age of each player

players['date_of_birth'] = pd.to_datetime(players['date_of_birth'])
# drop players with no date of birth (2 records only)
players = players[players['date_of_birth'].isnull() == False]
now = datetime.now()
players['age'] = (now - players['date_of_birth']).apply(lambda x: x.days) / 365.25
players['age'] = players['age'].round().astype(int)
```

```python
players.head(5)
```

	player_id	first_name	last_name	name	last_season	current_club_id	player_code	country_of_birth	city_of_birth	country_of_citizenship	...	current_club_domestic_competition_id	current_club_name	goals_2022	games_2022	assists_2022	minutes_played_2022	goals_for_2022	goals_against_2022	clean_sheet_20
4606	342229	Kylian	Mbappé	Kylian Mbappé	2023	583	kylian-mbappe	France	Paris	France	...	FR1	Paris Saint-Germain	36.0	42.0	9.0	3474.0	95.0	46.0	1
12249	418560	Erling	Haaland	Erling Haaland	2023	281	erling-haaland	England	Leeds	Norway	...	GB1	Manchester City	51.0	51.0	9.0	4024.0	131.0	39.0	2
10293	371998	NaN	Vinicius Junior	Vinicius Junior	2023	418	vinicius-junior	Brazil	São Gonçalo	Brazil	...	ES1	Real Madrid	23.0	53.0	21.0	4549.0	114.0	54.0	1
23354	581678	Jude	Bellingham	Jude Bellingham	2023	418	jude-bellingham	England	Stourbridge	England	...	ES1	Real Madrid	14.0	42.0	7.0	3557.0	93.0	50.0	1
29959	401923	Victor	Osimhen	Victor Osimhen	2023	6195	victor-osimhen	Nigeria	Lagos	Nigeria	...	IT1	SSC Napoli	31.0	39.0	5.0	3022.0	85.0	33.0	1

5 rows × 31 columns

```python
players.shape
```

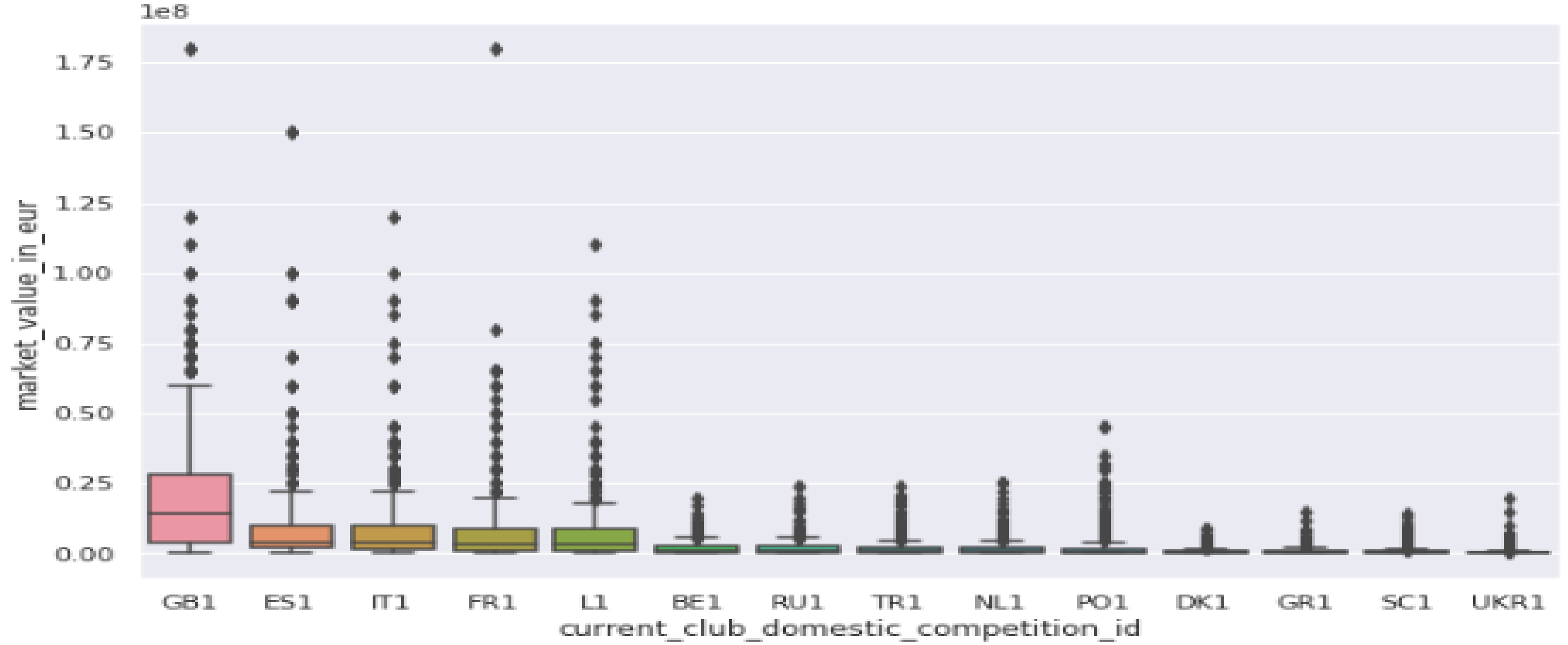
```
(6492, 31)
```

6. 데이터 시각화

리그

```
grouped = players.loc[:,['current_club_domestic_competition_id', 'market_value_in_eur']] \
    .groupby(['current_club_domestic_competition_id']) \
    .median() \
    .sort_values(by='market_value_in_eur', ascending=False)

sns.set(rc={'figure.figsize':(10,6)})
sns.boxplot(x=players.current_club_domestic_competition_id, y=players.market_value_in_eur, order=grouped.index)
```

```
<AxesSubplot:xlabel='current_club_domestic_competition_id', ylabel='market_value_in_eur'>
```

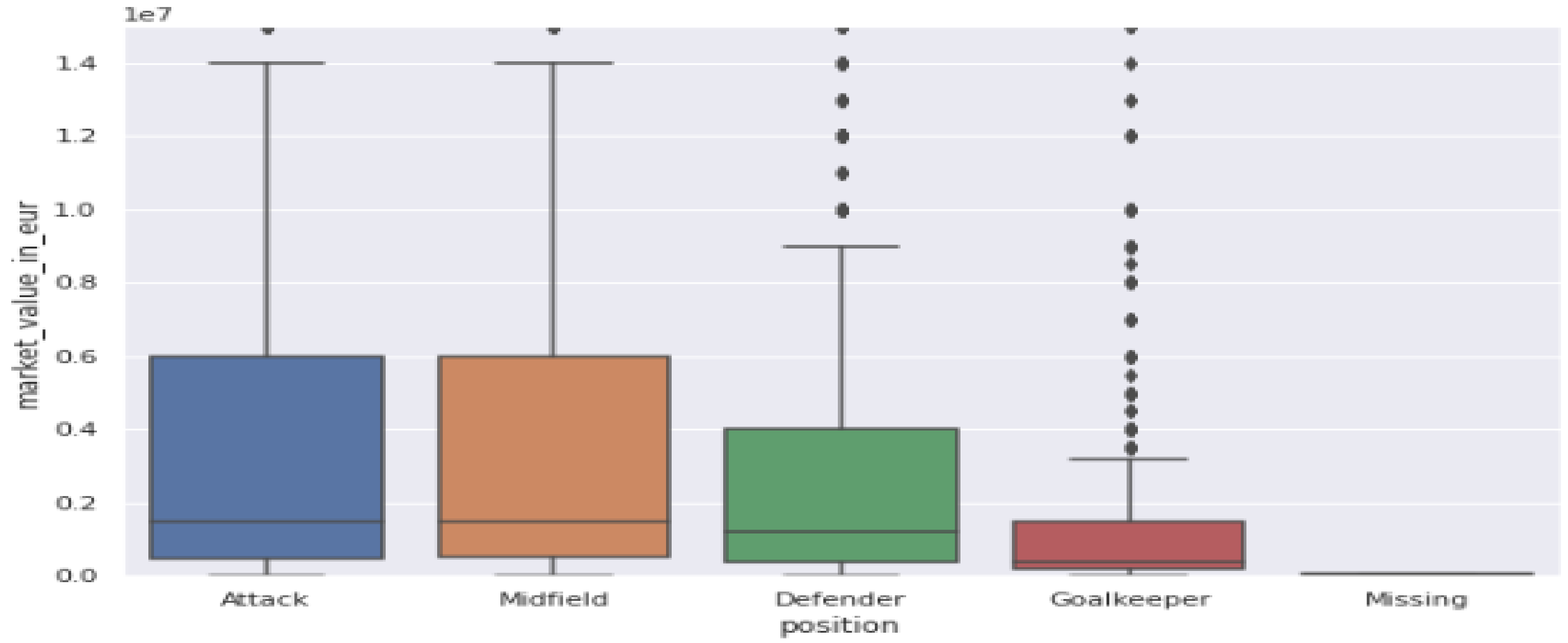


6. 데이터 시각화

포지션

```
grouped = players.loc[:,['position', 'market_value_in_eur']] \
    .groupby(['position']) \
    .median() \
    .sort_values(by='market_value_in_eur', ascending=False)

sns.set(rc={'figure.figsize':(10,6)})
plt.ylim(0, 15000000)
sns.boxplot(x=players.position, y=players.market_value_in_eur, order=grouped.index)
```

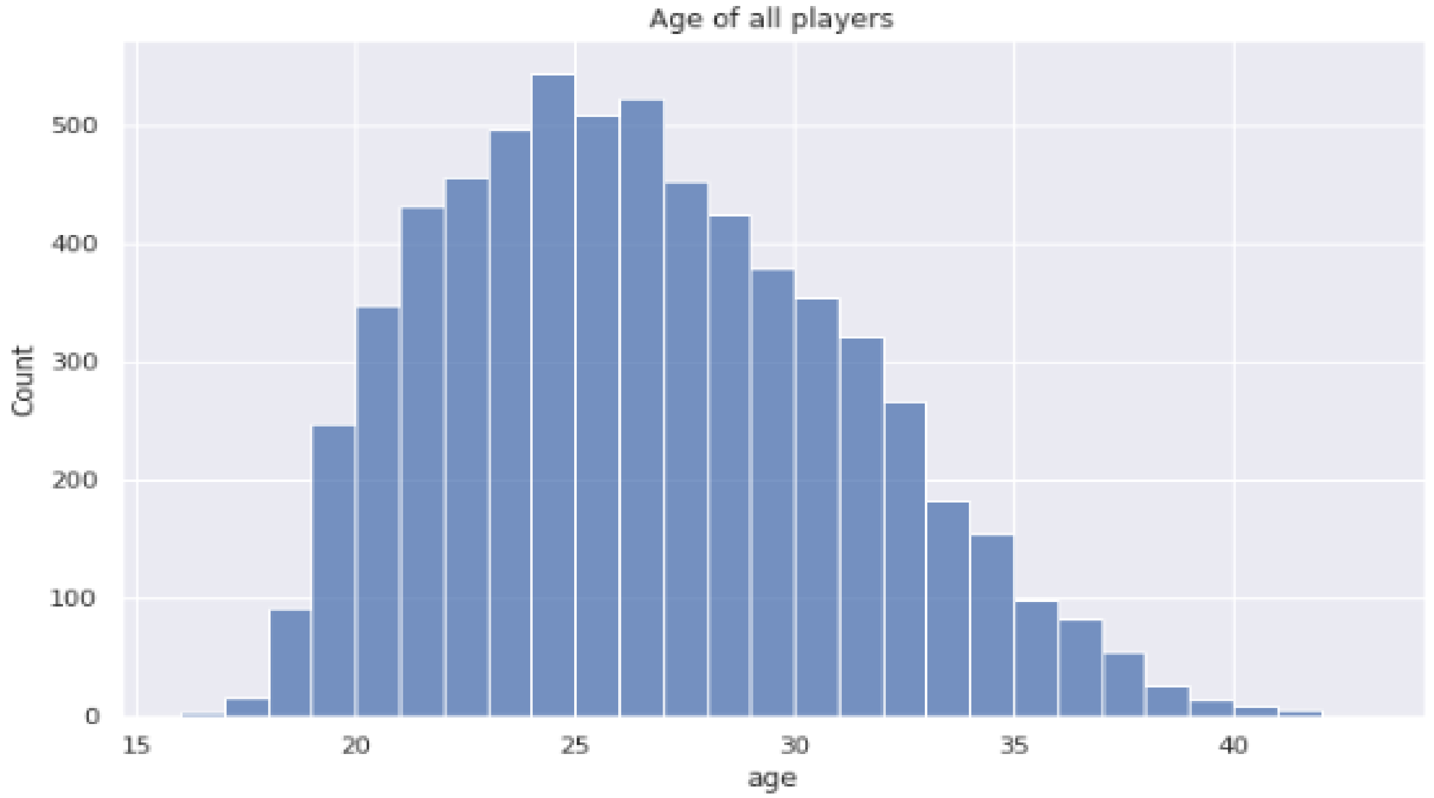
```
<AxesSubplot:xlabel='position', ylabel='market_value_in_eur'>
```



6. 데이터 시각화

나이

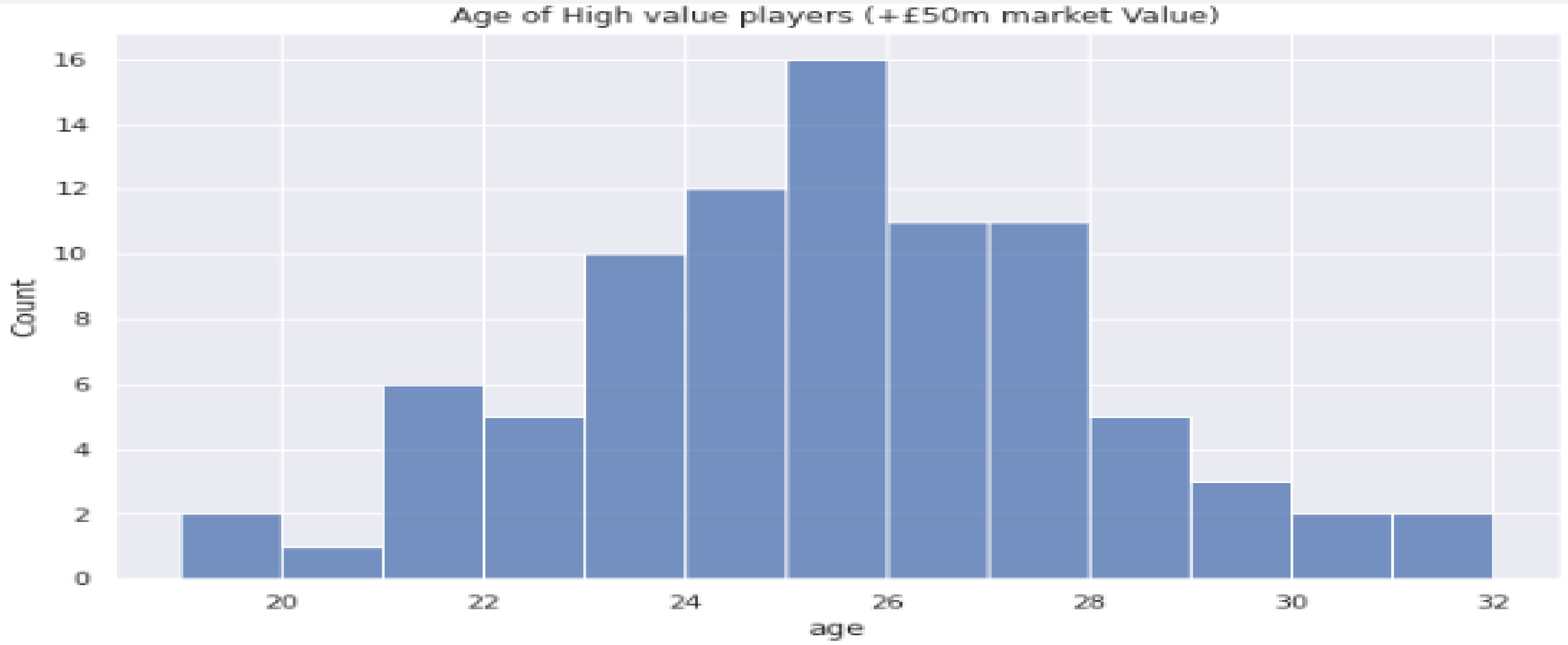
```
plt.title("Age of all players")
sns.histplot(x='age',data=players, binwidth=1)
plt.show()

print("Mean player age : ", players['age'].mean())
print("Median player age : ", players['age'].median())
```



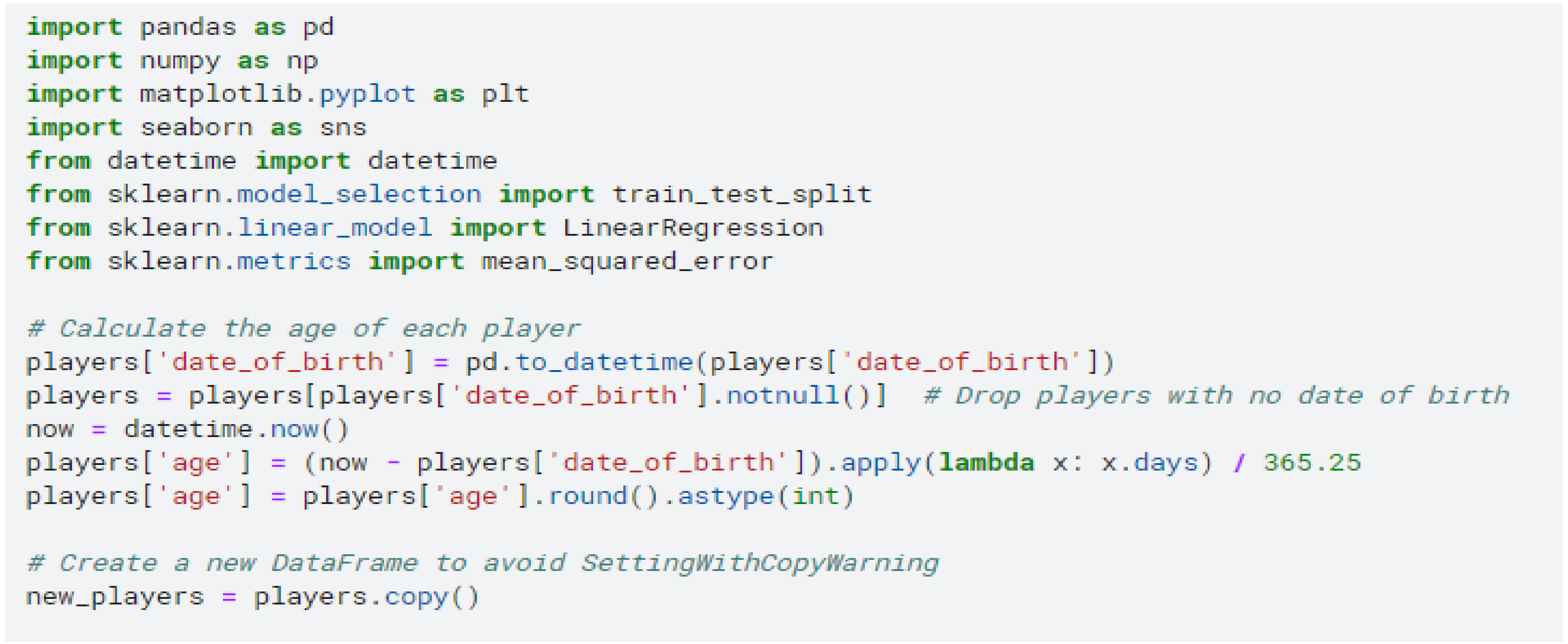
6. 데이터 시각화

가치+ 나이

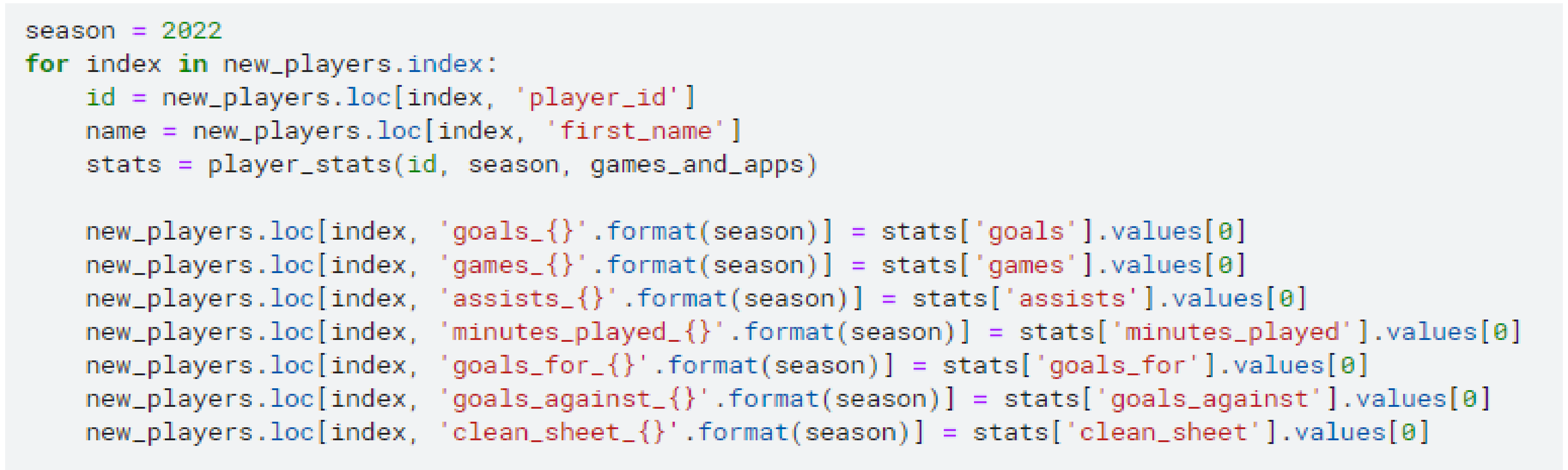
```
High_value_players = players[players['market_value_in_eur']>50000000]
print(High_value_players[['name', 'market_value_in_eur']])
plt.title("Age of High value players (+£50m market Value)")
sns.histplot(x='age',data=High_value_players, binwidth=1 )
plt.show()

print("Mean player age (High value) : ", High_value_players['age'].mean())
print("Median player age (High value) : ", High_value_players['age'].median())
```



7. 모델 학습(선형 회귀 모델)

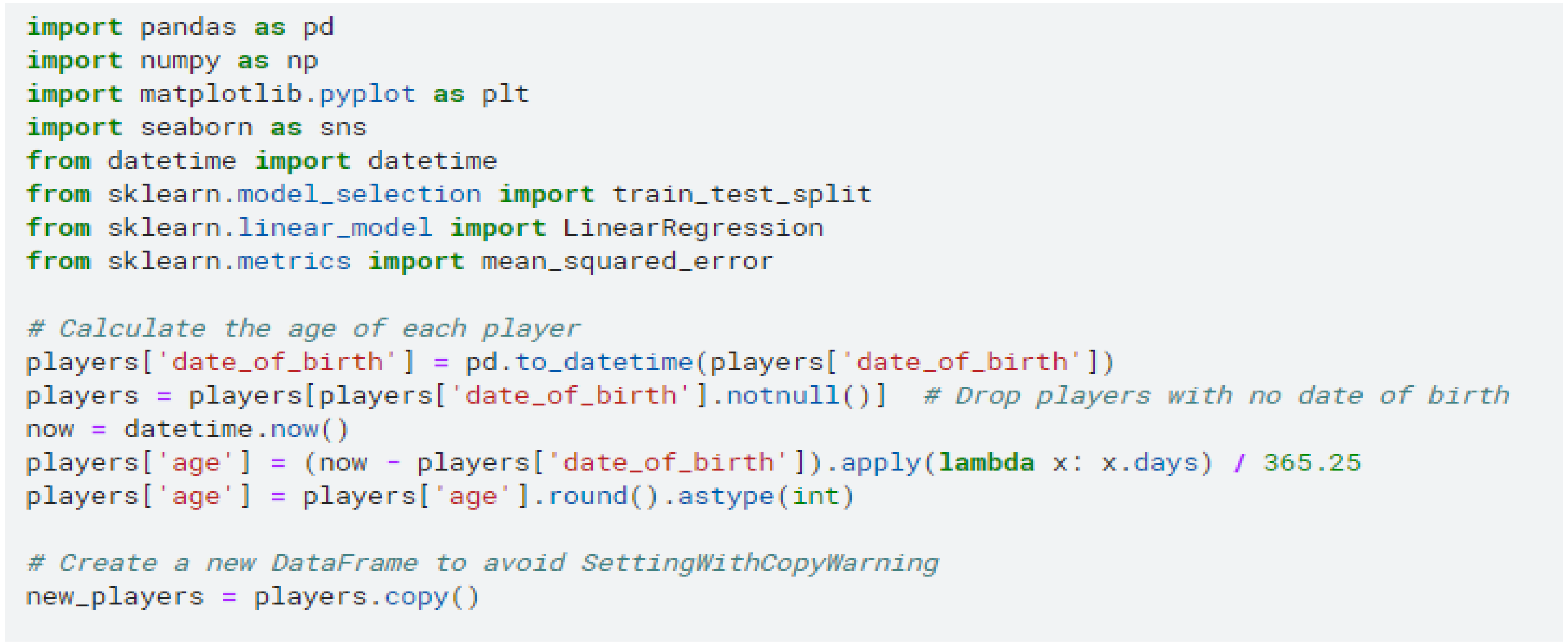


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Calculate the age of each player
players['date_of_birth'] = pd.to_datetime(players['date_of_birth'])
players = players[players['date_of_birth'].notnull()]  # Drop players with no date of birth
now = datetime.now()
players['age'] = (now - players['date_of_birth']).apply(lambda x: x.days) / 365.25
players['age'] = players['age'].round().astype(int)

# Create a new DataFrame to avoid SettingWithCopyWarning
new_players = players.copy()
```

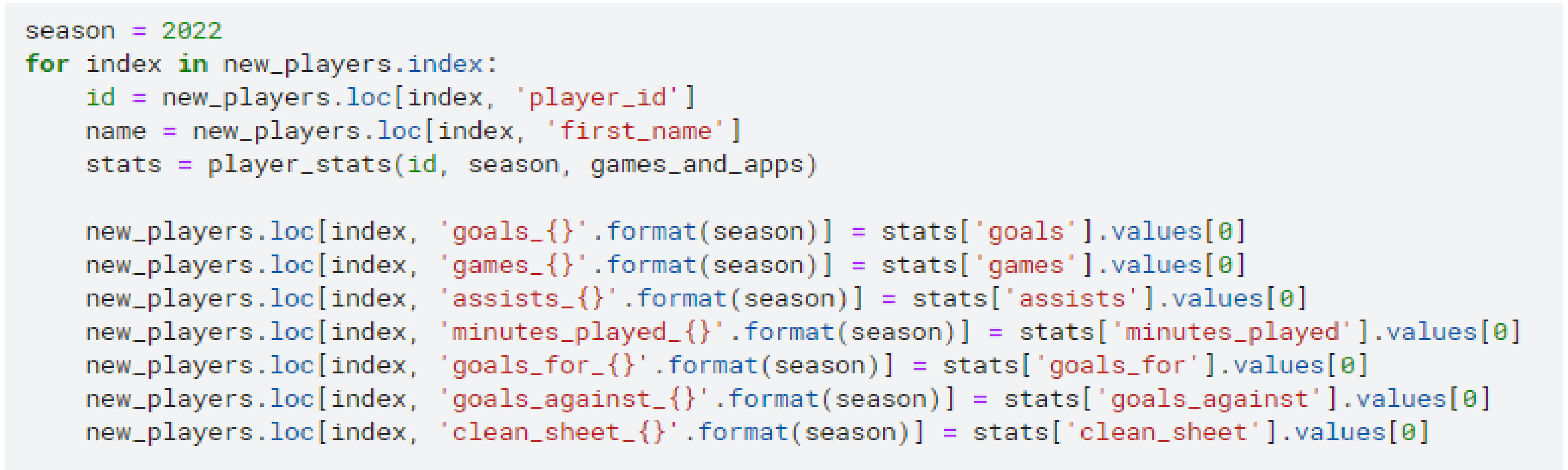


```python
season = 2022
for index in new_players.index:
    id = new_players.loc[index, 'player_id']
    name = new_players.loc[index, 'first_name']
    stats = player_stats(id, season, games_and_apps)

    new_players.loc[index, 'goals_{}'.format(season)] = stats['goals'].values[0]
    new_players.loc[index, 'games_{}'.format(season)] = stats['games'].values[0]
    new_players.loc[index, 'assists_{}'.format(season)] = stats['assists'].values[0]
    new_players.loc[index, 'minutes_played_{}'.format(season)] = stats['minutes_played'].values[0]
    new_players.loc[index, 'goals_for_{}'.format(season)] = stats['goals_for'].values[0]
    new_players.loc[index, 'goals_against_{}'.format(season)] = stats['goals_against'].values[0]
    new_players.loc[index, 'clean_sheet_{}'.format(season)] = stats['clean_sheet'].values[0]
```

7. 모델 학습(선형 회귀 모델)

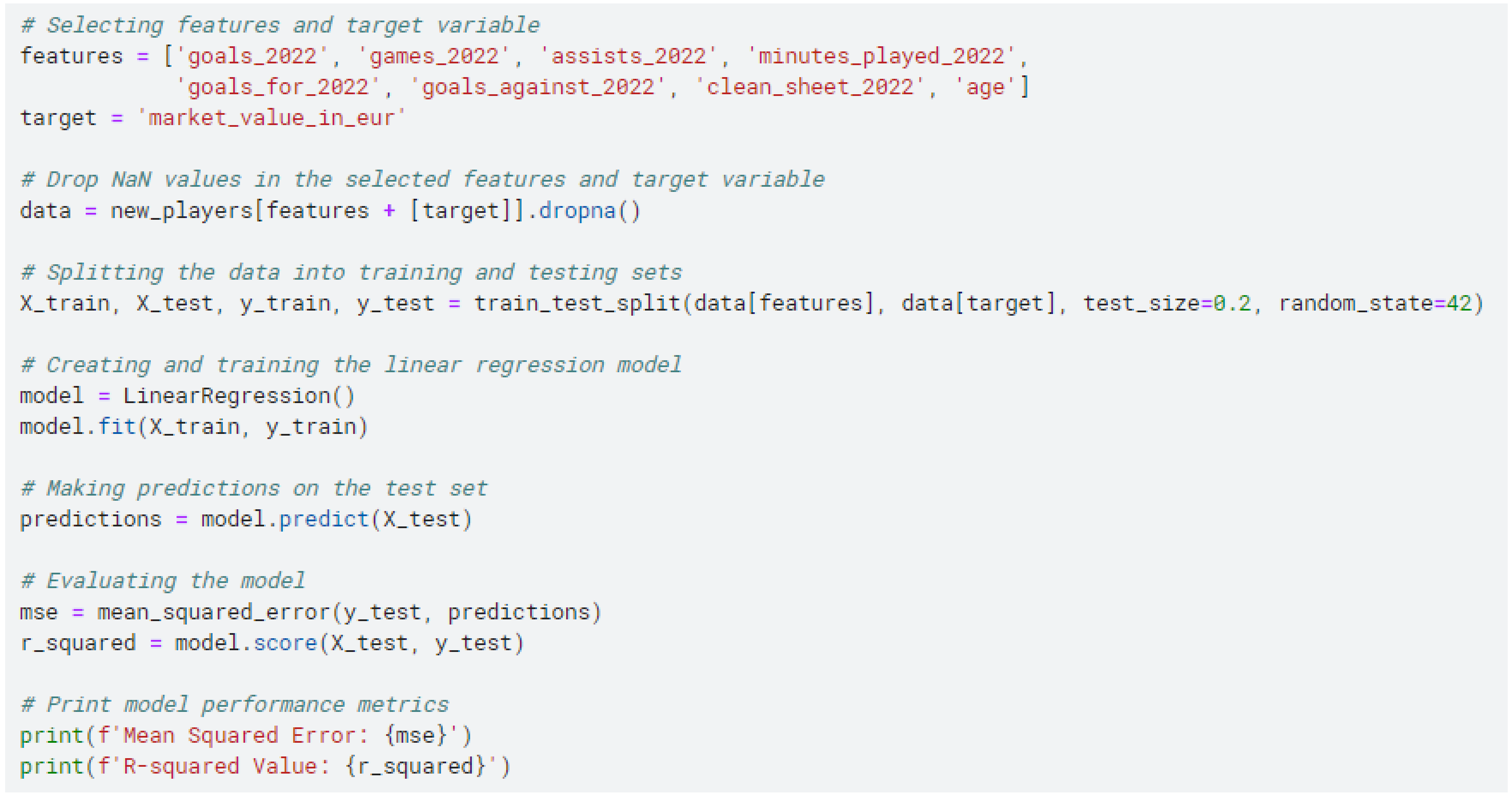


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Calculate the age of each player
players['date_of_birth'] = pd.to_datetime(players['date_of_birth'])
players = players[players['date_of_birth'].notnull()]  # Drop players with no date of birth
now = datetime.now()
players['age'] = (now - players['date_of_birth']).apply(lambda x: x.days) / 365.25
players['age'] = players['age'].round().astype(int)

# Create a new DataFrame to avoid SettingWithCopyWarning
new_players = players.copy()
```

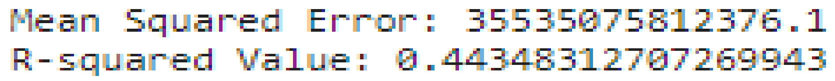


```python
season = 2022
for index in new_players.index:
    id = new_players.loc[index, 'player_id']
    name = new_players.loc[index, 'first_name']
    stats = player_stats(id, season, games_and_apps)

    new_players.loc[index, 'goals_{}'.format(season)] = stats['goals'].values[0]
    new_players.loc[index, 'games_{}'.format(season)] = stats['games'].values[0]
    new_players.loc[index, 'assists_{}'.format(season)] = stats['assists'].values[0]
    new_players.loc[index, 'minutes_played_{}'.format(season)] = stats['minutes_played'].values[0]
    new_players.loc[index, 'goals_for_{}'.format(season)] = stats['goals_for'].values[0]
    new_players.loc[index, 'goals_against_{}'.format(season)] = stats['goals_against'].values[0]
    new_players.loc[index, 'clean_sheet_{}'.format(season)] = stats['clean_sheet'].values[0]
```

7. 모델 학습(선형 회귀 모델)

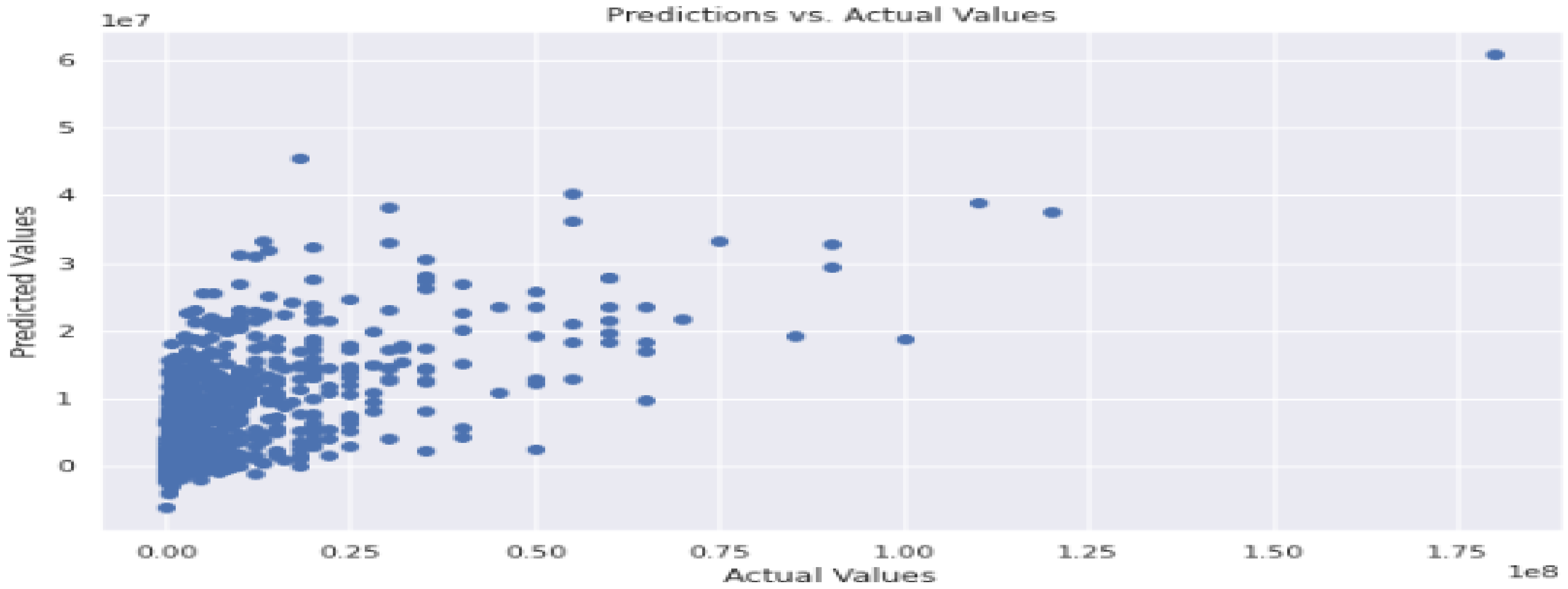


```python
# Selecting features and target variable
features = ['goals_2022', 'games_2022', 'assists_2022', 'minutes_played_2022',
            'goals_for_2022', 'goals_against_2022', 'clean_sheet_2022', 'age']
target = 'market_value_in_eur'

# Drop NaN values in the selected features and target variable
data = new_players[features + [target]].dropna()

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data[features], data[target], test_size=0.2, random_state=42)

# Creating and training the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Making predictions on the test set
predictions = model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(y_test, predictions)
r_squared = model.score(X_test, y_test)

# Print model performance metrics
print(f'Mean Squared Error: {mse}')
print(f'R-squared Value: {r_squared}')
```

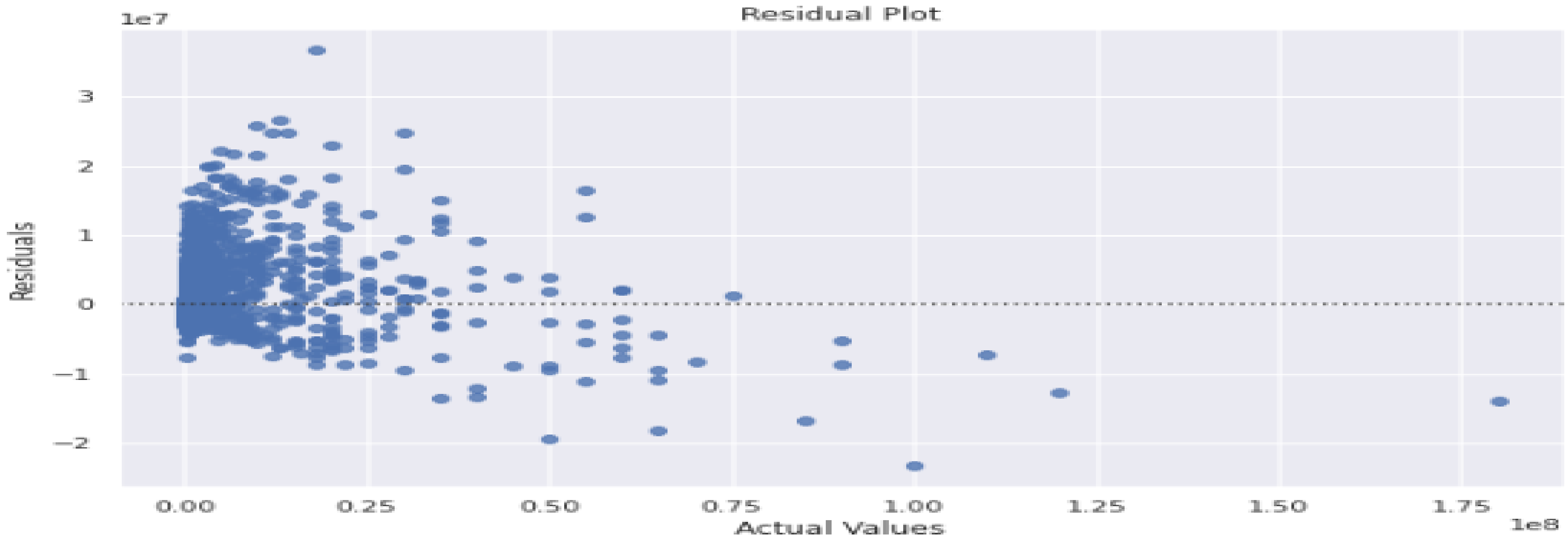


```
Mean Squared Error: 35535075812376.1
R-squared Value: 0.4434831270726994З
```

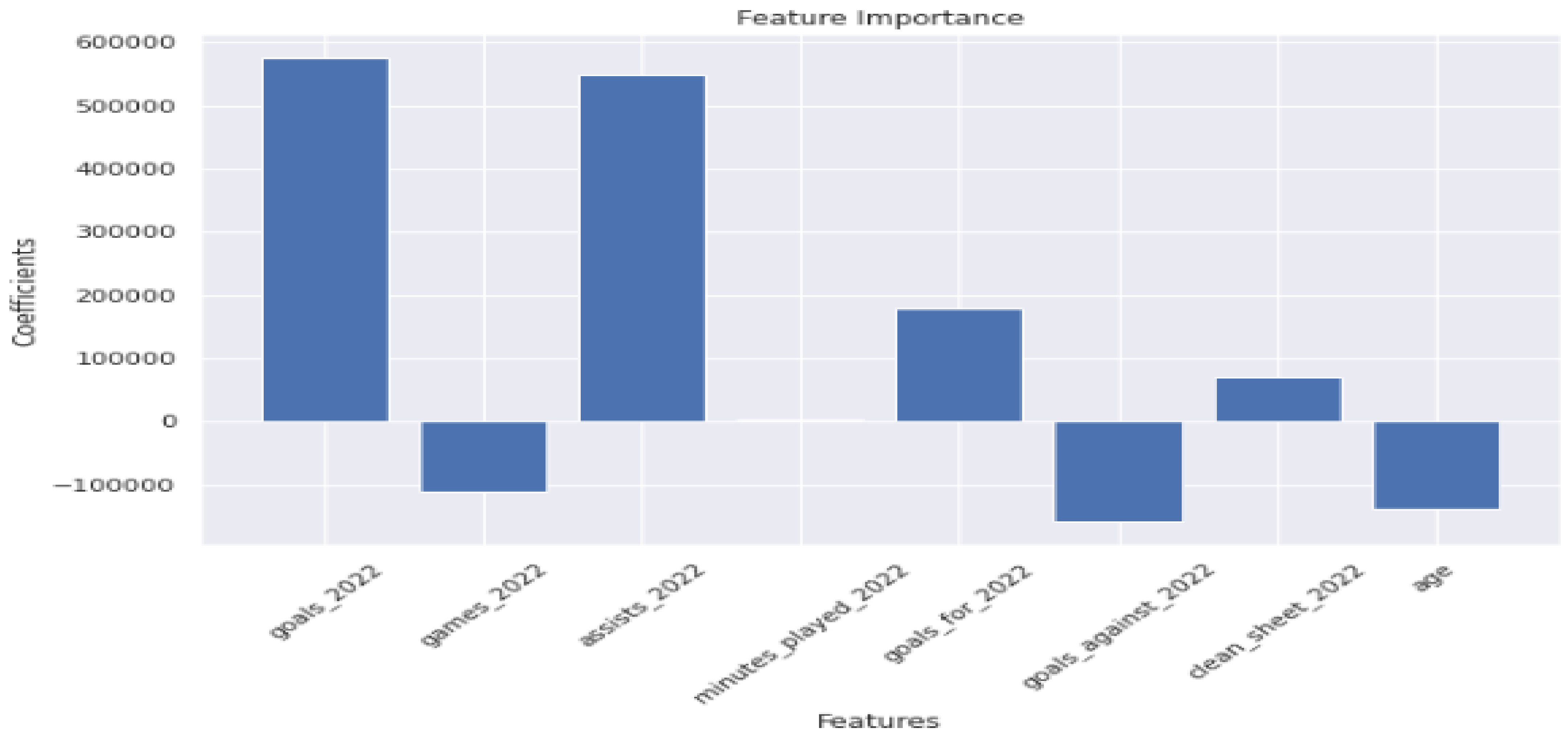
7. 모델 학습(선형 회귀 모델)



7. 모델 학습(선형 회귀 모델)



7. 모델 학습(선형 회귀 모델)



8. 소감

- 개인적으로 호기심이 있는 주제로 데이터를 분석하고 머신러닝을 할 수 있는 좋은 기회였다.

- 하지만 데이터셋에 바로 원하는 자료들이 없고, 모델 학습도 시켜본 사람이 없어 상당히 불편해 잘 되어 있는 것을 찾는 것이 중요하다고 느꼈다.

- 학습의 정확도가 떨어지고 단순히 보더라도 의구심이 드는 부분이 있었는데 시간이 부족해 더 여러가지 시도를 하지 못한 부분이 아쉬웠다.