```
#%%
### Contributions
# Tasks were distributed as follows:
# [Wenhan Zhan]: [Part 1 in EDA]
# [Ruoyao Yan]: [Part 1 in Data Cleaning, Part 4,7 in EDA, Part 5 in Advanced Analysis, Part 1,2,3 in Prediction]
# [Jun Sun]: [Part 2 in Data Cleaning, Part 1,2,3,5,6 in EDA, Part 1,2,3,4 in Advanced Analysis, Part 3 in Prediction]
```

```
#%% md
# 1. Data Cleaning
```

```
#%%
# Module: [Data Cleaning1]
# Contributor: [Ruoyao Yan]
import pandas as pd

df = pd.read_csv('Topic1_english_pl_dataset.csv')

#data cleaning
print("missing value statistics:")
print(df.isnull().sum())

numeric_cols = ['FullTimeHomeGoals', 'FullTimeAwayGoals', 'HalfTimeHomeGoals',
                'HalfTimeAwayGoals', 'HomeShots', 'AwayShots', 'HomeShotsOnTarget',
                'AwayShotsOnTarget', 'HomeCorners', 'AwayCorners', 'HomeFouls',
                'AwayFouls', 'HomeYellowCards', 'AwayYellowCards', 'HomeRedCards',
                'AwayRedCards']

df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
categorical_cols = ['FullTimeResult', 'HalfTimeResult', 'HomeTeam', 'AwayTeam']

for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])

```

```
#check check check check list
df.loc[df['FullTimeHomeGoals'] < 0, 'FullTimeHomeGoals'] = 0
df.loc[df['FullTimeAwayGoals'] < 0, 'FullTimeAwayGoals'] = 0
df.loc[df['HalfTimeHomeGoals'] < 0, 'HalfTimeHomeGoals'] = 0
df.loc[df['HalfTimeAwayGoals'] < 0, 'HalfTimeAwayGoals'] = 0


mask = df['HalfTimeHomeGoals'] > df['FullTimeHomeGoals']
df.loc[mask, 'HalfTimeHomeGoals'] = df.loc[mask, 'FullTimeHomeGoals']

mask = df['HalfTimeAwayGoals'] > df['FullTimeAwayGoals']
df.loc[mask, 'HalfTimeAwayGoals'] = df.loc[mask, 'FullTimeAwayGoals']

df['FullTimeHomeGoals'] = df['FullTimeHomeGoals'].astype(int)
df['FullTimeAwayGoals'] = df['FullTimeAwayGoals'].astype(int)
df['HomeShotsOnTarget'] = df['HomeShotsOnTarget'].astype(int)
df['HomeFouls'] = df['HomeFouls'].astype(int)
df['HalfTimeAwayGoals'] = df['HalfTimeAwayGoals'].astype(int)
df['AwayRedCards'] = df['AwayRedCards'].astype(int)

valid_results = ['H', 'D', 'A']
df['FullTimeResult'] = df['FullTimeResult'].apply(lambda x: x if x in valid_results else df['FullTimeResult'].mode()[0])


def check_result_consistency(row):
    if row['FullTimeHomeGoals'] > row['FullTimeAwayGoals'] and row['FullTimeResult'] != 'H
```

```
57 ':
58         return False
59     elif row['FullTimeHomeGoals'] < row['
   FullTimeAwayGoals'] and row['FullTimeResult'] != 'A
   ':
60         return False
61     elif row['FullTimeHomeGoals'] == row['
   FullTimeAwayGoals'] and row['FullTimeResult'] != 'D
   ':
62         return False
63     return True
64 
65 
66 df['Consistent'] = df.apply(
   check_result_consistency, axis=1)
67 inconsistent = df[~df['Consistent']]
68 
69 for idx, row in inconsistent.iterrows():
70     if row['FullTimeHomeGoals'] > row['
   FullTimeAwayGoals']:
71         df.at[idx, 'FullTimeResult'] = 'H'
72     elif row['FullTimeHomeGoals'] < row['
   FullTimeAwayGoals']:
73         df.at[idx, 'FullTimeResult'] = 'A'
74     else:
75         df.at[idx, 'FullTimeResult'] = 'D'
76 
77 df['HomeShots'] = df[['HomeShots', '
   HomeShotsOnTarget']].max(axis=1)
78 df['AwayShots'] = df[['AwayShots', '
   AwayShotsOnTarget']].max(axis=1)
79 
80 df.drop('Consistent', axis=1, inplace=True)
81 
82 df.to_csv('cleaned_football_data.csv', index=False)
83 print("data has been cleaned and saved! ")
84 print("\nMissing value statistics after cleaning:")
85 print(df.isnull().sum())
86 
87 #for ppt
88 print("\ncleaned data sample:")
```

```
 89 print(df.head())
 90 time_df = pd.DataFrame({
 91     'MatchDate': ['2000-08-19', '2001-09-20',
    '#######', '2003-07-01']
 92 })
 93
 94 time_df['MatchDate'] = pd.to_datetime(time_df['
    MatchDate'].replace('#######', pd.NaT), errors='
    coerce')
 95
 96 print(time_df)
 97
 98 def determine_outcome(home_goals, away_goals):
 99     if home_goals > away_goals:
100         return 'Home Win'
101     elif home_goals < away_goals:
102         return 'Away Win'
103     else:
104         return 'Draw'
105
106
107 df['MatchOutcome'] = df.apply(
108     lambda row: determine_outcome(row['
    FullTimeHomeGoals'], row['FullTimeAwayGoals']),
109     axis=1
110 )
111
112 print(df[['MatchDate', 'HomeTeam', 'AwayTeam', '
    FullTimeHomeGoals', 'FullTimeAwayGoals', '
    MatchOutcome']].head())
113
114 count_home = (df['FullTimeResult'] == 'H').sum()
115 print(f"Total home win: {count_home}")
116 count_away = (df['FullTimeResult'] == 'A').sum()
117 print(f"Total away win: {count_away}")
118 print("This shows that they have a higher winning
    rate play at home compare to play away")
119 num_teams = df['HomeTeam'].nunique()
120 print(f"total numbers of football teams: {
    num_teams}")
121 #%%
```

```
# Module: [Data Cleaning2]
# Contributor: [Jun Sun]
import pandas as pd
df = pd.read_csv('cleaned_football_data.csv')
df['HomeGoalDifference'] = df['FullTimeHomeGoals'] - df['FullTimeAwayGoals']
df['AwayGoalDifference'] = df['FullTimeAwayGoals'] - df['FullTimeHomeGoals']
df['GoalDifference'] = df['HomeGoalDifference']
df.to_csv('cleaned_football_data.csv', index=False)
import pandas as pd
df = pd.read_csv('cleaned_football_data.csv')
print(df.head())
```

```
#%% md
# 2. Basic Exploratory Data Analysis (EDA)
2.1 Trends in number of matches played (by season)
```

```
#%%
# Module: [EDA1]
# Contributor: [Wenhan Zhan] and [Junxiang Wang]
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("cleaned_football_data.csv")

df['MatchDate'] = pd.to_datetime(df['MatchDate'])

def get_season(date):
    if date.month >= 8:
        return f"{date.year}-{date.year+1}"
    else:
        return f"{date.year-1}-{date.year}"

df['Season'] = df['MatchDate'].apply(get_season)

season_counts = df['Season'].value_counts().sort_index()

plt.figure(figsize=(12,6))
sns.lineplot(x=season_counts.index,y=season_counts
```

```
158 .values,markers='o')
159 plt.xticks(rotation = 30)
160 plt.title('Number of Matches per Season')
161 plt.xlabel('Season')
162 plt.ylabel('Number of Matches')
163 plt.tight_layout()
164 plt.show()
165 #%% md
166 # Trends in number of matches played (by Months)
167 #%%
168 # Module: [EDA2]
169 # Contributor: [Jun Sun]
170 import pandas as pd
171 import matplotlib.pyplot as plt
172 from datetime import datetime
173
174 df = pd.read_csv(
175     'cleaned_football_data.csv')
176
177 def is_valid_date(date_str):
178     try:
179         datetime.strptime(date_str, "%Y-%m-%d")
    # Change format if needed
180         return True
181     except:
182         return False
183
184 valid_dates = df['MatchDate'].apply(is_valid_date)
185
186 df = df[valid_dates]
187
188 df['MatchDate'] = pd.to_datetime(df['MatchDate'])
189
190 df['Month'] = df['MatchDate'].dt.month_name()
191
192 month_order = ['August', 'September', 'October', '
    November', 'December',
193                'January', 'February', 'March', '
    April', 'May']
194
195 monthly_matches = df['Month'].value_counts().
```

```
reindex(month_order)

plt.figure(figsize=(12, 6))
monthly_matches.plot(kind='bar', color='blue')
plt.title('Total Number of Matches by Month (All Seasons Combined)')
plt.xlabel('Month')
plt.ylabel('Number of Matches')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
#%% md
## Distribution of home and away team goals (bar chart)
```

```
#%%
# Module: [EDA3]
# Contributor: [Jun Sun]
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv(
    'cleaned_football_data.csv')

home_goals = df['FullTimeHomeGoals'].value_counts().sort_index()
away_goals = df['FullTimeAwayGoals'].value_counts().sort_index()

max_goals = max(home_goals.index.max(), away_goals.index.max())
x_range = range(0, max_goals + 1)

home_goals = home_goals.reindex(x_range, fill_value=0)
away_goals = away_goals.reindex(x_range, fill_value=0)

fig, ax = plt.subplots(1, 2, figsize=(14, 5), sharey=True)
```

```
228
229 ax[0].bar(home_goals.index, home_goals.values,
    color='skyblue', edgecolor='black')
230 ax[0].set_title('Home Team Goals Distribution',
    fontsize=14)
231 ax[0].set_xlabel('Goals Scored', fontsize=12)
232 ax[0].set_ylabel('Number of Matches', fontsize=12)
233 ax[0].set_xticks(x_range)
234 ax[0].grid(axis='y', linestyle='--', alpha=0.5)
235
236 ax[1].bar(away_goals.index, away_goals.values,
    color='salmon', edgecolor='black')
237 ax[1].set_title('Away Team Goals Distribution',
    fontsize=14)
238 ax[1].set_xlabel('Goals Scored', fontsize=12)
239 ax[1].set_xticks(x_range)
240 ax[1].grid(axis='y', linestyle='--', alpha=0.5)
241
242 plt.tight_layout()
243 plt.show()
244
```

```
245 #%%
246 # Module: [EDA4]
247 # Contributor: [Ruoyao Yan]
248 import pandas as pd
249 import matplotlib.pyplot as plt
250 import seaborn as sns
251
252
253 df = pd.read_csv("cleaned_football_data.csv")
254 df['MatchDate'] = pd.to_datetime(df['MatchDate'])
255
256 def get_season(date):
257     if date.month >= 8:
258         return f"{date.year}-{date.year+1}"
259     else:
260         return f"{date.year-1}-{date.year}"
261
262 df['Season'] = df['MatchDate'].apply(get_season)
263 season_goals = df.groupby('Season')[['
    FullTimeHomeGoals', 'FullTimeAwayGoals']].sum().
```

```
sum(axis=1)
first_15_seasons = [s for s in season_goals.index if int(s.split('-')[0]) <= 2014]
last_10_seasons = [s for s in season_goals.index if 2015 <= int(s.split('-')[0]) <= 2024]


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

ax1.bar(range(len(first_15_seasons)),
        season_goals[first_15_seasons].values,
        color='skyblue', edgecolor='black')
ax1.set_title('Total Goals (2000-2015)')
ax1.set_xticks(range(len(first_15_seasons)))
ax1.set_xticklabels(first_15_seasons, rotation=45, ha='right')
ax1.set_ylim(0, 1300)
ax2.bar(range(len(last_10_seasons)),
        season_goals[last_10_seasons].values,
        color='salmon', edgecolor='black')
ax2.set_title('Total Goals (2015-2025)')
ax2.set_xticks(range(len(last_10_seasons)))
ax2.set_xticklabels(last_10_seasons, rotation=45, ha='right')
ax2.set_ylim(0, 1300)
plt.tight_layout()
plt.show()
```

```
#%% md
## Full-time/half-time result ratio (pie chart)
```

```
#%%
# Module: [EDA5]
# Contributor: [Jun Sun]
import matplotlib.pyplot as plt

result_order = ['H', 'D', 'A']

fulltime_counts = df['FullTimeResult'].value_counts().reindex(result_order, fill_value=0)
fulltime_labels = ['Home Win (H)', 'Draw (D)', 'Away Win (A)']
```

```
297 fulltime_colors = ['lightgreen', 'lightgray', '
    tomato']
298 
299 
300 halftime_counts = df['HalfTimeResult'].
    value_counts().reindex(result_order, fill_value=0)
301 halftime_labels = ['Home Lead (H)', 'Draw (D)', '
    Away Lead (A)']
302 halftime_colors = ['lightblue', 'lightgray', '
    orange']
303 
304 fig, ax = plt.subplots(1, 2, figsize=(14, 6))
305 
306 
307 ax[0].pie(fulltime_counts, labels=fulltime_labels
    , autopct='%1.1f%%',
308           colors=fulltime_colors, startangle=140,
    wedgeprops={'edgecolor': 'black'})
309 ax[0].set_title('Full-Time Match Result Proportion
    ', fontsize=14)
310 
311 # Half-Time Pie Chart
312 ax[1].pie(halftime_counts, labels=halftime_labels
    , autopct='%1.1f%%',
313           colors=halftime_colors, startangle=140,
    wedgeprops={'edgecolor': 'black'})
314 ax[1].set_title('Half-Time Match Result Proportion
    ', fontsize=14)
315 
316 plt.tight_layout()
317 plt.show()
318 
319 #%% md
320 ## Home vs Away Win Percentage (bar graph)
321 #%%
322 # Module: [EDA6]
323 # Contributor: [Jun Sun]
324 import matplotlib.pyplot as plt
325 
326 total_matches = len(df)
327 total_home_matches = df['HomeTeam'].count()
```

```
total_away_matches = df['AwayTeam'].count()

num_home_wins = (df['FullTimeResult'] == 'H').sum()
num_away_wins = (df['FullTimeResult'] == 'A').sum()

home_win_rate = num_home_wins / total_home_matches
away_win_rate = num_away_wins / total_away_matches

win_rate_data = {
    'Home Win Rate': home_win_rate,
    'Away Win Rate': away_win_rate
}


fig, ax = plt.subplots(figsize=(8, 5))
bars = ax.bar(win_rate_data.keys(), win_rate_data.values(), color=['royalblue', 'darkorange'])

for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 0.02,
            f'{height:.1%}', ha='center', va='bottom', fontsize=12)

ax.set_title('Home vs Away Win Rate', fontsize=14)
ax.set_ylabel('Win Rate', fontsize=12)
ax.set_ylim(0, 1)
ax.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

```
#%%
# Module: [EDA7]
# Contributor: [Ruoyao Yan]
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv('cleaned_football_data.csv')
```

```
364
365 # Filter Arsenal vs Brighton matches
366 AvsB = []
367 for index, row in df.iterrows():
368     home = row['HomeTeam']
369     away = row['AwayTeam']
370     if (home == 'Arsenal' and away == 'Brighton')
    or (home == 'Brighton' and away == 'Arsenal'):
371         AvsB.append(row)
372
373 arsenal_vs_brighton = pd.DataFrame(AvsB)
374
375 total_arsenal_wins = len(arsenal_vs_brighton[
376     ((arsenal_vs_brighton['HomeTeam'] == 'Arsenal'
    ) & (arsenal_vs_brighton['FullTimeResult'] == 'H'
    )) |
377     ((arsenal_vs_brighton['AwayTeam'] == 'Arsenal'
    ) & (arsenal_vs_brighton['FullTimeResult'] == 'A'
    ))
378 ])
379 total_brighton_wins = len(arsenal_vs_brighton[
380     ((arsenal_vs_brighton['HomeTeam'] == 'Brighton
    ') & (arsenal_vs_brighton['FullTimeResult'] == 'H'
    )) |
381     ((arsenal_vs_brighton['AwayTeam'] == 'Brighton
    ') & (arsenal_vs_brighton['FullTimeResult'] == 'A'
    ))
382 ])
383 total_draws = len(arsenal_vs_brighton[
    arsenal_vs_brighton['FullTimeResult'] == 'D'])
384
385 labels = ['Arsenal Wins', 'Brighton Wins', 'Draws'
    ]
386 counts = [total_arsenal_wins, total_brighton_wins
    , total_draws]
387 colors = ['red', 'blue', 'grey']
388
389 plt.figure(figsize=(6, 4))
390 plt.pie(
391     counts,
392     labels=labels,
```

```
    autopct='%1.2f%%',
    startangle=90,
    colors=colors,
)

plt.title('Arsenal vs Brighton Match Results', fontsize=14)
plt.tight_layout()
plt.show()
```

```
#%% md
```

```
# 3. Advanced Statistical Analysis
```

```
#%% md
```

```
Average shots per match, shots on target, and goal conversion rate
```

```
#%%
# Module: [Advanced Analysis1]
# Contributor: [Jun Sun]
df.fillna({
    'HomeShots': 0, 'AwayShots': 0,
    'HomeShotsOnTarget': 0, 'AwayShotsOnTarget': 0,
    'FullTimeHomeGoals': 0, 'FullTimeAwayGoals': 0
}, inplace=True)

df['TotalShots'] = df['HomeShots'] + df['AwayShots']
df['TotalShotsOnTarget'] = df['HomeShotsOnTarget'] + df['AwayShotsOnTarget']
df['TotalGoals'] = df['FullTimeHomeGoals'] + df['FullTimeAwayGoals']

average_shots = df['TotalShots'].mean()
average_shots_on_target = df['TotalShotsOnTarget'].mean()
average_goals = df['TotalGoals'].mean()

total_goals = df['TotalGoals'].sum()
total_shots = df['TotalShots'].sum()
goal_conversion_rate = total_goals / total_shots if total_shots > 0 else 0
```

```
426
427
428 print("Match Statistics Summary")
429 print("-" * 40)
430 print(f"Average total shots per match     : {
    average_shots:.2f}")
431 print(f"Average shots on target per match : {
    average_shots_on_target:.2f}")
432 print(f"Average goals per match           : {
    average_goals:.2f}")
433 print(f"Goal conversion rate (Goals/Shots): {
    goal_conversion_rate:.2%}")
434
435 #%% md
436 ## Average distribution of fouls, yellow cards,
    and red cards (Home vs Away)
437 #%%
438 # Module: [Advanced Analysis2]
439 # Contributor: [Jun Sun]
440 import matplotlib.pyplot as plt
441 import numpy as np
442
443 # Compute averages
444 categories = ['Fouls', 'Yellow Cards', 'Red Cards'
    ]
445 home_values = [
446     df['HomeFouls'].mean(),
447     df['HomeYellowCards'].mean(),
448     df['HomeRedCards'].mean()
449 ]
450 away_values = [
451     df['AwayFouls'].mean(),
452     df['AwayYellowCards'].mean(),
453     df['AwayRedCards'].mean()
454 ]
455
456 x = np.arange(len(categories))
457 bar_width = 0.35
458
459 fig, ax = plt.subplots(figsize=(10, 6))
460 bars1 = ax.bar(x - bar_width / 2, home_values,
```

```
bar_width, label='Home Team', color='skyblue')
bars2 = ax.bar(x + bar_width / 2, away_values,
    bar_width, label='Away Team', color='salmon')


ax.bar_label(bars1, fmt='%.2f', padding=3)
ax.bar_label(bars2, fmt='%.2f', padding=3)

ax.set_ylabel('Average per Match')
ax.set_title('Average Fouls and Cards per Match (
    Home vs Away)')
ax.set_xticks(x)
ax.set_xticklabels(categories)
ax.set_ylim(0, max(max(home_values), max(
    away_values)) + 2)
ax.grid(axis='y', linestyle='--', alpha=0.7)
ax.legend()

plt.tight_layout()
plt.show()
```

```
#%% md
# Correlation between corner kicks and win rate
```

```
#%%
# Module: [Advanced Analysis3]
# Contributor: [Jun Sun]
import pandas as pd


df = pd.read_csv("cleaned_football_data.csv")

df['HomeWin'] = df['FullTimeResult'].apply(lambda
    x: 1 if x == 'H' else 0)
df['AwayWin'] = df['FullTimeResult'].apply(lambda
    x: 1 if x == 'A' else 0)

home_correlation = df[['HomeCorners', 'HomeWin']].
    corr().iloc[0, 1]
away_correlation = df[['AwayCorners', 'AwayWin']].
    corr().iloc[0, 1]

print("Correlation between Home Corners and Home
```

```
Win:", home_correlation)
print("Correlation between Away Corners and Away Win:", away_correlation)
#%% md
# Analysis of the relationship between goal difference and match outcome
#%%
# Module: [Advanced Analysis4]
# Contributor: [Jun Sun]
import matplotlib.pyplot as plt
import pandas as pd

required_columns = ['FullTimeHomeGoals', 'FullTimeAwayGoals', 'FullTimeResult']
if not all(col in df.columns for col in required_columns):
    missing = [col for col in required_columns if col not in df.columns]
    raise ValueError(f"Missing required columns: {missing}")

df_clean = df.dropna(subset=required_columns).copy()

df_clean['GoalDifference'] = df_clean['FullTimeHomeGoals'] - df_clean['FullTimeAwayGoals']

result_mapping = {'H': 1, 'D': 0, 'A': -1}
df_clean['MatchOutcomeScore'] = df_clean['FullTimeResult'].map(result_mapping)


df_clean = df_clean.dropna(subset=['MatchOutcomeScore'])
gd_corr = df_clean[['GoalDifference', 'MatchOutcomeScore']].corr().iloc[0, 1]


print(f"   Correlation between goal difference and match result: {gd_corr:.3f}")
```

```
if abs(gd_corr) >= 0.7:
    print("→ Strong correlation: Goal difference is a good predictor of result.")
elif abs(gd_corr) >= 0.4:
    print("→ Moderate correlation: Goal difference has some predictive power.")
else:
    print("→ Weak correlation: Goal difference isn't strongly predictive of result.")


plt.figure(figsize=(10, 6))
plt.scatter(
    df_clean['GoalDifference'], df_clean['MatchOutcomeScore'],
    alpha=0.6, edgecolors='k', color='mediumseagreen'
)
plt.axhline(0, color='gray', linestyle='--', lw=1)
plt.axvline(0, color='gray', linestyle='--', lw=1)

plt.title('Goal Difference vs Match Outcome Score', fontsize=14)
plt.xlabel('Goal Difference (Home - Away)', fontsize=12)
plt.ylabel('Match Outcome Score (H=1, D=0, A=-1)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

```

```
#%% md
# Liverpool vs Mancity
```

```
#%%
# Module: [Advanced Analysis5]
# Contributor: [Ruoyao Yan]
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv(
    'cleaned_football_data.csv')
```

```
554
555 LvsM = []
556 for index, row in df.iterrows():
557     home = row['HomeTeam']
558     away = row['AwayTeam']
559     if (home == 'Liverpool' and away == 'Man City'
    ) or (home == 'Man City' and away == 'Liverpool'):
560         LvsM.append(row)
561
562 liverpool_vs_mancity = pd.DataFrame(LvsM)
563
564 total_liverpool_wins = len(liverpool_vs_mancity[((
    liverpool_vs_mancity['HomeTeam'] == 'Liverpool'
    ) & (liverpool_vs_mancity['FullTimeResult'] == 'H'
    )) | ((liverpool_vs_mancity['AwayTeam'] == '
    Liverpool') & (liverpool_vs_mancity['
    FullTimeResult'] == 'A'))])
565
566 total_mancity_wins = len(liverpool_vs_mancity[((
    liverpool_vs_mancity['HomeTeam'] == 'Man City'
    ) & (liverpool_vs_mancity['FullTimeResult'] == 'H'
    )) | ((liverpool_vs_mancity['AwayTeam'] == 'Man
    City') & (liverpool_vs_mancity['FullTimeResult'
    ] == 'A'))])
567
568 total_draws = len(liverpool_vs_mancity[
    liverpool_vs_mancity['FullTimeResult'] == 'D'])
569
570 labels = ['Liverpool Wins', 'Man city Wins', '
    Draws']
571 counts = [total_liverpool_wins, total_mancity_wins
    , total_draws]
572 colors = ['red', 'blue', 'grey']
573
574 plt.figure(figsize=(6, 4))
575 plt.pie(
576     counts,
577     labels=labels,
578     autopct='%1.2f%%',
579     startangle=90,
580     colors=colors,
```

```
)

plt.title('Liverpool vs Man City Match Results', fontsize=14)
plt.tight_layout()
plt.show()
```

```
#%% md
# Prediction
```

```
#%% md
Predict the numeber of goals scored by Liverpool as the home team
```

```
#%%
# Module: [Prediction1]
# Contributor: [Ruoyao Yan]
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt

liverpool_home = df[df['HomeTeam'] == 'Liverpool'].copy()

features = ['HomeShotsOnTarget', 'AwayShotsOnTarget']
X = liverpool_home[features]
y = liverpool_home['FullTimeHomeGoals']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

```

```
print("Predicted goals:", y_pred)
y_pred_rounded = np.round(y_pred).astype(int)
print("Predicted goals (rounded):", y_pred_rounded)
counter = Counter(y_pred_rounded)
most_common_pred, frequency = counter.most_common(1)[0]
print(f"Most frequently predicted goals: {most_common_pred} (appeared {frequency} times)")

print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R² score:", r2_score(y_test, y_pred))
print("Maximum predicted value:", y_pred.max())

plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual goals")
plt.ylabel("Predicted goals")
plt.title("Liverpool home goal prediction")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--')  # Ideal line
plt.ylim(0, 5)
plt.show()

total_goals_method1 = most_common_pred * 19
print(f"Season total goals: {total_goals_method1:.1f}")
```

```
#%% md
Predict Liverpool's specific winnning rate in home games
```

```
#%%
# Module: [Prediction2]
# Contributor: [Ruoyao Yan]
import pandas as pd
import numpy as np
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

```
646
647 liverpool_home = df[df['HomeTeam'] == 'Liverpool'
    ].copy()
648 liverpool_home['HomeWin'] = (liverpool_home['
    FullTimeResult'] == 'H').astype(int)
649 win_rate_by_shots = liverpool_home.groupby('
    HomeShotsOnTarget')['HomeWin'].mean().reset_index
    ()
650 win_rate_by_shots.columns = ['HomeShotsOnTarget',
    'WinRate']
651
652 X = win_rate_by_shots[['HomeShotsOnTarget']]
653 y = win_rate_by_shots['WinRate']
654 from sklearn.linear_model import LinearRegression
655
656 model = LinearRegression()
657 model.fit(X, y)
658
659 X_test = pd.DataFrame(np.arange(0, 15).reshape(-1
    , 1), columns=['HomeShotsOnTarget'])
660 y_pred = model.predict(X_test)
661 import matplotlib.pyplot as plt
662
663 plt.figure(figsize=(10, 6))
664 plt.scatter(X, y, color='blue', label='
    Actual_win_rate')
665 plt.plot(X_test, y_pred, color='red', label='
    Predicted_win_rate')
666 plt.xlabel('Liverpool home shots on target')
667 plt.ylabel('Predicted_win_rate')
668 plt.title('Liverpool home shots on target vs win
    rate')
669 plt.legend()
670 plt.grid(True)
671 plt.show()
672 next_season_shots = 4
673 predicted_win_rate = model.predict(pd.DataFrame({'
    HomeShotsOnTarget': [next_season_shots]}))
674 print(f"Projected average home shots on target for
     Liverpool next season {next_season_shots},
    Predicted win rate: {predicted_win_rate[0]:.1%}")
```

```
675
676 features = ['HomeShotsOnTarget', '
    AwayShotsOnTarget']
677 X = liverpool_home[features]
678 y = liverpool_home['FullTimeHomeGoals']
679
680 X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.2, random_state
    =42)
681
682 model = LinearRegression()
683 model.fit(X_train, y_train)
684
685
686 y_pred = model.predict(X_test)
687
688 print("Predicted goals:", y_pred)
689 y_pred_rounded = np.round(y_pred).astype(int)
690 print("Predicted goals (rounded):", y_pred_rounded
    )
691 counter = Counter(y_pred_rounded)
692 most_common_pred, frequency = counter.most_common(
    1)[0]
693 print(f"Most frequently predicted goals: {
    most_common_pred} (appeared {frequency} times)")
694
695 print("Mean Squared Error (MSE):",
    mean_squared_error(y_test, y_pred))
696 print("R² score:", r2_score(y_test, y_pred))
697 print("Maximum predicted value:", y_pred.max())
698 #%% md
699 Predicting goals using cards and corner kicks
700 #%%
701 # Module: [Prediction3]
702 # Contributor: [Ruoyao Yan] and [Jun Sun]
703 import pandas as pd
704 import numpy as np
705 import matplotlib.pyplot as plt
706 import seaborn as sns
707
708 from sklearn.model_selection import
```

```
train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,
    r2_score, mean_absolute_error


df = pd.read_csv('cleaned_football_data.csv')
df['TotalGoals'] = df['FullTimeHomeGoals'] + df['
    FullTimeAwayGoals']
df['TotalCorners'] = df['HomeCorners'] + df['
    AwayCorners']
df['TotalYellowCards'] = df['HomeYellowCards'] +
    df['AwayYellowCards']
df['TotalRedCards'] = df['HomeRedCards'] + df['
    AwayRedCards']

features = ['TotalCorners', 'TotalYellowCards', '
    TotalRedCards']
target = 'TotalGoals'

X = df[features]
y = df[target]

X_train, X_test, y_train, y_test =
    train_test_split(
        X, y, test_size=0.2, random_state=42
)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse:.2f}")
print(f"MAE: {mae:.2f}")
print(f"R²: {r2:.2f}")

```

```
coef_df = pd.DataFrame({
    'Feature': features,
    'Coefficient': model.coef_,
    'AbsoluteEffect': np.abs(model.coef_)
}).sort_values(by='AbsoluteEffect', ascending=False)

print("\nThe impact of each feature on the total number of goals (sorted by absolute value):")
print(coef_df)


plt.figure(figsize=(18, 5))

for i, feature in enumerate(features):
    plt.subplot(1, 3, i+1)
    sns.regplot(x=feature, y='TotalGoals', data=df, scatter_kws={'alpha': 0.4})
    plt.title(f"{feature} vs TotalGoals")
    plt.xlabel(feature)
    plt.ylabel("TotalGoals")

plt.tight_layout()
plt.show()

# example
new_data = pd.DataFrame([[12, 4, 1]], columns=features)  # 12角球、4黄牌、1红牌
predicted_goal = model.predict(new_data)[0]
print(f"\nPredict the number of goals in the match: {predicted_goal:.1f}")
```