

```

import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import networkx as nx
from networkx.drawing.nx_pydot import graphviz_layout
import seaborn as sns
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from operator import itemgetter

```

```
df = pd.read_csv("results.csv")
```

```
df["date"] = pd.to_datetime(df["date"])
```

```
df.isna().sum()
```

```

date            0
home_team       0
away_team       0
home_score      0
away_score      0
tournament      0
city            0
country         0
neutral         0
dtype: int64

```

```
df.dropna(inplace=True)
```

```
df.dtypes
```

```

date            datetime64[ns]
home_team       object
away_team       object
home_score      int64
away_score      int64
tournament      object
city            object
country         object
neutral         bool
dtype: object

```

```
df.sort_values("date").tail()
```

	date	home_team	away_team	home_score	away_score	\
46411	2024-03-26	Tajikistan	Saudi Arabia	1	1	
46412	2024-03-26	Jordan	Pakistan	7	0	
46413	2024-03-26	Bahrain	Nepal	3	0	

46415	2024-03-26	Bangladesh	Palestine	0	1
46441	2024-03-26	Finland	Estonia	2	1

		tournament	city	country	neutral
46411	FIFA World Cup	qualification	Dushanbe	Tajikistan	False
46412	FIFA World Cup	qualification	Amman	Jordan	False
46413	FIFA World Cup	qualification	Riffa	Bahrain	False
46415	FIFA World Cup	qualification	Dhaka	Bangladesh	False
46441		Friendly	Helsinki	Finland	False

```
df = df[(df["date"] >= "2020-7-11")].reset_index(drop=True)
```

```
df.sort_values("date").tail()
```

	date	home_team	away_team	home_score	away_score	\
3606	2024-03-26	Tajikistan	Saudi Arabia	1	1	
3607	2024-03-26	Jordan	Pakistan	7	0	
3608	2024-03-26	Bahrain	Nepal	3	0	
3610	2024-03-26	Bangladesh	Palestine	0	1	
3636	2024-03-26	Finland	Estonia	2	1	

		tournament	city	country	neutral
3606	FIFA World Cup	qualification	Dushanbe	Tajikistan	False
3607	FIFA World Cup	qualification	Amman	Jordan	False
3608	FIFA World Cup	qualification	Riffa	Bahrain	False
3610	FIFA World Cup	qualification	Dhaka	Bangladesh	False
3636		Friendly	Helsinki	Finland	False

```
df.home_team.value_counts()
```

```
home_team
United States    44
Mexico           44
Bahrain          42
Qatar            40
Morocco          37
..
Cook Islands     1
Elba Island      1
Aymara           1
New Caledonia    1
Tibet            1
Name: count, Length: 243, dtype: int64
```

```
rank = pd.read_csv("fifa_ranking-2024-04-04.csv")
```

```
rank["rank_date"] = pd.to_datetime(rank["rank_date"])
rank = rank[(rank["rank_date"] >= "2020-7-11")].reset_index(drop=True)
```

```
rank["country_full"] = rank["country_full"].str.replace("USA", "United States")
```

```

rank = rank.set_index(['rank_date']).groupby(['country_full'],
group_keys=False).resample('D').first().ffill().reset_index()

df_ranked = df.merge(rank[["country_full", "total_points",
"previous_points", "rank", "rank_change", "rank_date"]],
left_on=["date", "home_team"], right_on=["rank_date",
"country_full"]).drop(["rank_date", "country_full"], axis=1)

df_ranked = df_ranked.merge(rank[["country_full", "total_points",
"previous_points", "rank", "rank_change", "rank_date"]],
left_on=["date", "away_team"], right_on=["rank_date", "country_full"],
suffixes=("_home", "_away")).drop(["rank_date", "country_full"],
axis=1)

df_ranked[(df_ranked.home_team == "Brazil") | (df_ranked.away_team ==
"Brazil")].tail(10)

```

	date	home_team	away_team	home_score	away_score	\
2281	2023-06-17	Brazil	Guinea	4	1	
2336	2023-06-20	Brazil	Senegal	2	4	
2439	2023-09-08	Brazil	Bolivia	5	1	
2507	2023-09-12	Peru	Brazil	0	1	
2544	2023-10-12	Brazil	Venezuela	1	1	
2637	2023-10-17	Uruguay	Brazil	2	0	
2685	2023-11-16	Colombia	Brazil	2	1	
2789	2023-11-21	Brazil	Argentina	0	1	
2998	2024-03-23	England	Brazil	0	1	
3061	2024-03-26	Spain	Brazil	3	3	

	tournament	city	country	neutral
2281	Friendly	Barcelona	Spain	True
2336	Friendly	Lisbon	Portugal	True
2439	FIFA World Cup qualification	Belém	Brazil	False
2507	FIFA World Cup qualification	Lima	Peru	False
2544	FIFA World Cup qualification	Cuiabá	Brazil	False
2637	FIFA World Cup qualification	Montevideo	Uruguay	False
2685	FIFA World Cup qualification	Barranquilla	Colombia	False
2789	FIFA World Cup qualification	Rio de Janeiro	Brazil	False
2998	Friendly	London	England	False
3061	Friendly	Madrid	Spain	False

	total_points_home	previous_points_home	rank_home
rank_change_home \			
2281	1834.21	1840.77	3.0
2.0			
2336	1834.21	1840.77	3.0
2.0			
2439	1828.27	1828.27	3.0
0.0			
2507	1561.20	1561.20	21.0
0.0			
2544	1837.61	1828.27	3.0
0.0			
2637	1626.51	1633.13	17.0
1.0			
2685	1626.60	1629.60	17.0
1.0			
2789	1812.20	1837.61	3.0
0.0			
2998	1800.05	1800.05	3.0
0.0			
3061	1732.64	1732.64	8.0
0.0			

	total_points_away	previous_points_away	rank_away
rank_change_away			
2281	1305.92	1290.47	79.0
-4.0			
2336	1613.21	1603.98	18.0
-1.0			
2439	1295.09	1295.09	83.0
0.0			
2507	1828.27	1828.27	3.0
0.0			
2544	1422.83	1417.23	53.0
-4.0			
2637	1837.61	1828.27	3.0
0.0			
2685	1812.20	1837.61	3.0
0.0			
2789	1861.29	1851.41	1.0
0.0			
2998	1784.09	1784.09	5.0
0.0			
3061	1784.09	1784.09	5.0
0.0			

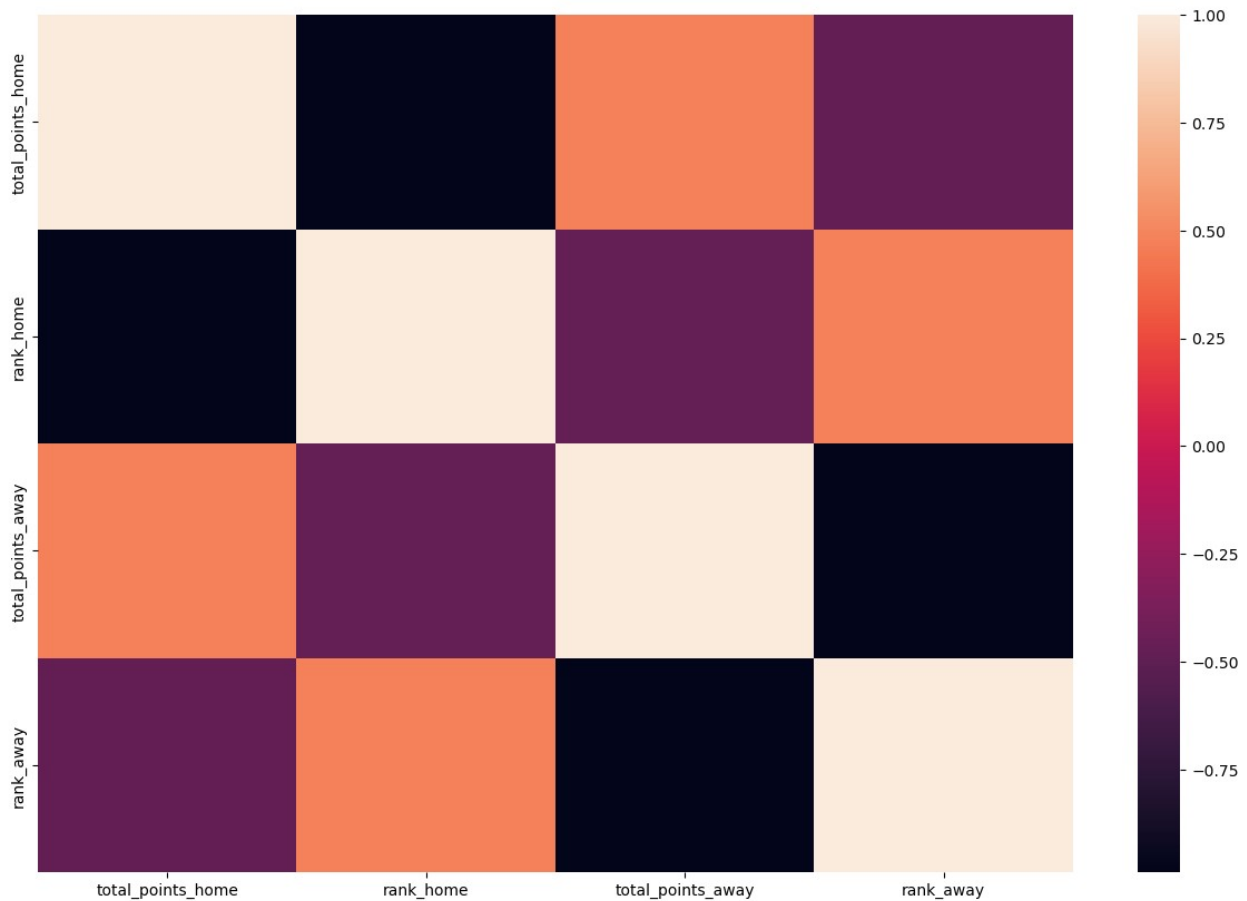
df = df_ranked

```
def result_finder(home, away):
    if home > away:
        return pd.Series([0, 3, 0])
    if home < away:
        return pd.Series([1, 0, 3])
    else:
        return pd.Series([2, 1, 1])

results = df.apply(lambda x: result_finder(x["home_score"],
x["away_score"]), axis=1)

df[["result", "home_team_points", "away_team_points"]] = results

plt.figure(figsize=(15, 10))
sns.heatmap(df[["total_points_home", "rank_home", "total_points_away",
"rank_away"]].corr())
plt.show()
```



```
df["rank_dif"] = df["rank_home"] - df["rank_away"]
df["sg"] = df["home_score"] - df["away_score"]
df["points_home_by_rank"] = df["home_team_points"]/df["rank_away"]
df["points_away_by_rank"] = df["away_team_points"]/df["rank_home"]
```

```

home_team = df[["date", "home_team", "home_score", "away_score",
"rank_home", "rank_away", "rank_change_home", "total_points_home",
"result", "rank_dif", "points_home_by_rank", "home_team_points"]]

away_team = df[["date", "away_team", "away_score", "home_score",
"rank_away", "rank_home", "rank_change_away", "total_points_away",
"result", "rank_dif", "points_away_by_rank", "away_team_points"]]

home_team.columns = [h.replace("home_", "").replace("_home",
 "").replace("away_", "suf_").replace("_away", "_suf") for h in
home_team.columns]

away_team.columns = [a.replace("away_", "").replace("_away",
 "").replace("home_", "suf_").replace("_home", "_suf") for a in
away_team.columns]

team_stats = pd.concat([home_team, away_team])

team_stats_raw = team_stats.copy()

stats_val = []

for index, row in team_stats.iterrows():
    team = row["team"]
    date = row["date"]
    past_games = team_stats.loc[(team_stats["team"] == team) &
(team_stats["date"] < date)].sort_values(by=['date'], ascending=False)
    last5 = past_games.head(5)

    goals = past_games["score"].mean()
    goals_l5 = last5["score"].mean()

    goals_suf = past_games["suf_score"].mean()
    goals_suf_l5 = last5["suf_score"].mean()

    rank = past_games["rank_suf"].mean()
    rank_l5 = last5["rank_suf"].mean()

    if len(last5) > 0:
        points = past_games["total_points"].values[0] -
past_games["total_points"].values[-1]#qtd de pontos ganhos
        points_l5 = last5["total_points"].values[0] -
last5["total_points"].values[-1]
    else:
        points = 0
        points_l5 = 0

    gp = past_games["team_points"].mean()
    gp_l5 = last5["team_points"].mean()

```

```

gp_rank = past_games["points_by_rank"].mean()
gp_rank_l5 = last5["points_by_rank"].mean()

stats_val.append([goals, goals_l5, goals_suf, goals_suf_l5, rank,
rank_l5, points, points_l5, gp, gp_l5, gp_rank, gp_rank_l5])

stats_cols = ["goals_mean", "goals_mean_l5", "goals_suf_mean",
"goals_suf_mean_l5", "rank_mean", "rank_mean_l5", "points_mean",
"points_mean_l5", "game_points_mean", "game_points_mean_l5",
"game_points_rank_mean", "game_points_rank_mean_l5"]

stats_df = pd.DataFrame(stats_val, columns=stats_cols)

full_df = pd.concat([team_stats.reset_index(drop=True), stats_df],
axis=1, ignore_index=False)

home_team_stats = full_df.iloc[:int(full_df.shape[0]/2),:]
away_team_stats = full_df.iloc[int(full_df.shape[0]/2):,:]

home_team_stats.columns[-12:]

Index(['goals_mean', 'goals_mean_l5', 'goals_suf_mean',
'goals_suf_mean_l5',
'rank_mean', 'rank_mean_l5', 'points_mean', 'points_mean_l5',
'game_points_mean', 'game_points_mean_l5',
'game_points_rank_mean',
'game_points_rank_mean_l5'],
dtype='object')

home_team_stats = home_team_stats[home_team_stats.columns[-12:]]
away_team_stats = away_team_stats[away_team_stats.columns[-12:]]

home_team_stats.columns = ['home_'+str(col) for col in
home_team_stats.columns]
away_team_stats.columns = ['away_'+str(col) for col in
away_team_stats.columns]

match_stats = pd.concat([home_team_stats,
away_team_stats.reset_index(drop=True)], axis=1, ignore_index=False)

full_df = pd.concat([df, match_stats.reset_index(drop=True)], axis=1,
ignore_index=False)

full_df.columns

Index(['date', 'home_team', 'away_team', 'home_score', 'away_score',
'tournament', 'city', 'country', 'neutral',
'total_points_home',
'previous_points_home', 'rank_home', 'rank_change_home',
'total_points_away', 'previous_points_away', 'rank_away',
'rank_change_away', 'result', 'home_team_points',
'away_team_points',

```

```

        'rank_dif', 'sg', 'points_home_by_rank', 'points_away_by_rank',
        'home_goals_mean', 'home_goals_mean_l5', 'home_goals_suf_mean',
        'home_goals_suf_mean_l5', 'home_rank_mean',
'home_rank_mean_l5',
        'home_points_mean', 'home_points_mean_l5',
'home_game_points_mean',
        'home_game_points_mean_l5', 'home_game_points_rank_mean',
        'home_game_points_rank_mean_l5', 'away_goals_mean',
        'away_goals_mean_l5', 'away_goals_suf_mean',
'away_goals_suf_mean_l5',
        'away_rank_mean', 'away_rank_mean_l5', 'away_points_mean',
        'away_points_mean_l5', 'away_game_points_mean',
        'away_game_points_mean_l5', 'away_game_points_rank_mean',
        'away_game_points_rank_mean_l5'],
        dtype='object')

def find_friendly(x):
    if x == "Friendly":
        return 1
    else: return 0

full_df["is_friendly"] = full_df["tournament"].apply(lambda x:
find_friendly(x))

full_df = pd.get_dummies(full_df, columns=["is_friendly"])

full_df.columns

Index(['date', 'home_team', 'away_team', 'home_score', 'away_score',
      'tournament', 'city', 'country', 'neutral',
'total_points_home',
      'previous_points_home', 'rank_home', 'rank_change_home',
      'total_points_away', 'previous_points_away', 'rank_away',
      'rank_change_away', 'result', 'home_team_points',
'away_team_points',
      'rank_dif', 'sg', 'points_home_by_rank', 'points_away_by_rank',
      'home_goals_mean', 'home_goals_mean_l5', 'home_goals_suf_mean',
      'home_goals_suf_mean_l5', 'home_rank_mean',
'home_rank_mean_l5',
      'home_points_mean', 'home_points_mean_l5',
'home_game_points_mean',
      'home_game_points_mean_l5', 'home_game_points_rank_mean',
      'home_game_points_rank_mean_l5', 'away_goals_mean',
      'away_goals_mean_l5', 'away_goals_suf_mean',
'away_goals_suf_mean_l5',
      'away_rank_mean', 'away_rank_mean_l5', 'away_points_mean',
      'away_points_mean_l5', 'away_game_points_mean',
      'away_game_points_mean_l5', 'away_game_points_rank_mean',
      'away_game_points_rank_mean_l5', 'is_friendly_0',

```



```
'is_friendly_1'],
      dtype='object')
```

```
base_df = full_df[["date", "home_team", "away_team", "rank_home",
"rank_away", "home_score", "away_score", "result", "rank_dif",
"rank_change_home", "rank_change_away", 'home_goals_mean',
      'home_goals_mean_l5', 'home_goals_suf_mean',
'home_goals_suf_mean_l5',
      'home_rank_mean', 'home_rank_mean_l5', 'home_points_mean',
      'home_points_mean_l5', 'away_goals_mean', 'away_goals_mean_l5',
      'away_goals_suf_mean', 'away_goals_suf_mean_l5',
'away_rank_mean',
      'away_rank_mean_l5', 'away_points_mean',
'away_points_mean_l5', 'home_game_points_mean',
'home_game_points_mean_l5',
      'home_game_points_rank_mean',
'home_game_points_rank_mean_l5', 'away_game_points_mean',
      'away_game_points_mean_l5', 'away_game_points_rank_mean',
      'away_game_points_rank_mean_l5',
      'is_friendly_0', 'is_friendly_1']]
```

```
base_df.tail()
```

	date	home_team	away_team	rank_home	rank_away
home_score \					
3058	2024-03-26	Scotland	Northern Ireland	34.0	74.0
0					
3059	2024-03-26	Senegal	Benin	17.0	98.0
1					
3060	2024-03-26	Slovenia	Portugal	55.0	7.0
2					
3061	2024-03-26	Spain	Brazil	8.0	5.0
3					
3062	2024-03-26	Finland	Estonia	60.0	123.0
2					

	away_score	result	rank_dif	rank_change_home	...	\
3058	1	1	-40.0	-2.0	...	
3059	0	0	-81.0	-3.0	...	
3060	0	0	48.0	1.0	...	
3061	3	2	3.0	0.0	...	
3062	1	0	-63.0	1.0	...	

	home_game_points_mean	home_game_points_mean_l5	\
3058	1.609756	0.4	
3059	2.130435	2.6	
3060	1.809524	2.0	
3061	1.979167	2.4	
3062	1.282609	1.2	

	home_game_points_rank_mean	home_game_points_rank_mean_l5	\
3058	0.047218	0.007393	
3059	0.046507	0.034018	
3060	0.035370	0.065393	
3061	0.097396	0.046042	
3062	0.057781	0.010899	

	away_game_points_mean	away_game_points_mean_l5	\
3058	0.923077	1.4	
3059	1.160000	0.4	
3060	2.239130	3.0	
3061	2.195122	0.8	
3062	0.930233	0.2	

	away_game_points_rank_mean	away_game_points_rank_mean_l5	\
3058	0.020086	0.038922	
3059	0.012331	0.002920	
3060	0.099223	0.057056	
3061	0.116011	0.203774	
3062	0.009899	0.001786	

	is_friendly_0	is_friendly_1
3058	False	True
3059	False	True
3060	False	True
3061	False	True
3062	False	True

[5 rows x 37 columns]

base_df.isna().sum()

date	0
home_team	0
away_team	0
rank_home	0
rank_away	0
home_score	0
away_score	0
result	0
rank_dif	0
rank_change_home	0
rank_change_away	0
home_goals_mean	88
home_goals_mean_l5	88
home_goals_suf_mean	88
home_goals_suf_mean_l5	88
home_rank_mean	88
home_rank_mean_l5	88
home_points_mean	0

home_points_mean_l5	0
away_goals_mean	102
away_goals_mean_l5	102
away_goals_suf_mean	102
away_goals_suf_mean_l5	102
away_rank_mean	102
away_rank_mean_l5	102
away_points_mean	0
away_points_mean_l5	0
home_game_points_mean	88
home_game_points_mean_l5	88
home_game_points_rank_mean	88
home_game_points_rank_mean_l5	88
away_game_points_mean	102
away_game_points_mean_l5	102
away_game_points_rank_mean	102
away_game_points_rank_mean_l5	102
is_friendly_0	0
is_friendly_1	0
dtype: int64	

```
base_df_no_fg = base_df.dropna()
```

```
df = base_df_no_fg
```

```
def no_draw(x):
    if x == 2:
        return 1
    else:
        return x
```

```
df = df.copy()
df["target"] = df["result"].apply(lambda x: no_draw(x))
```

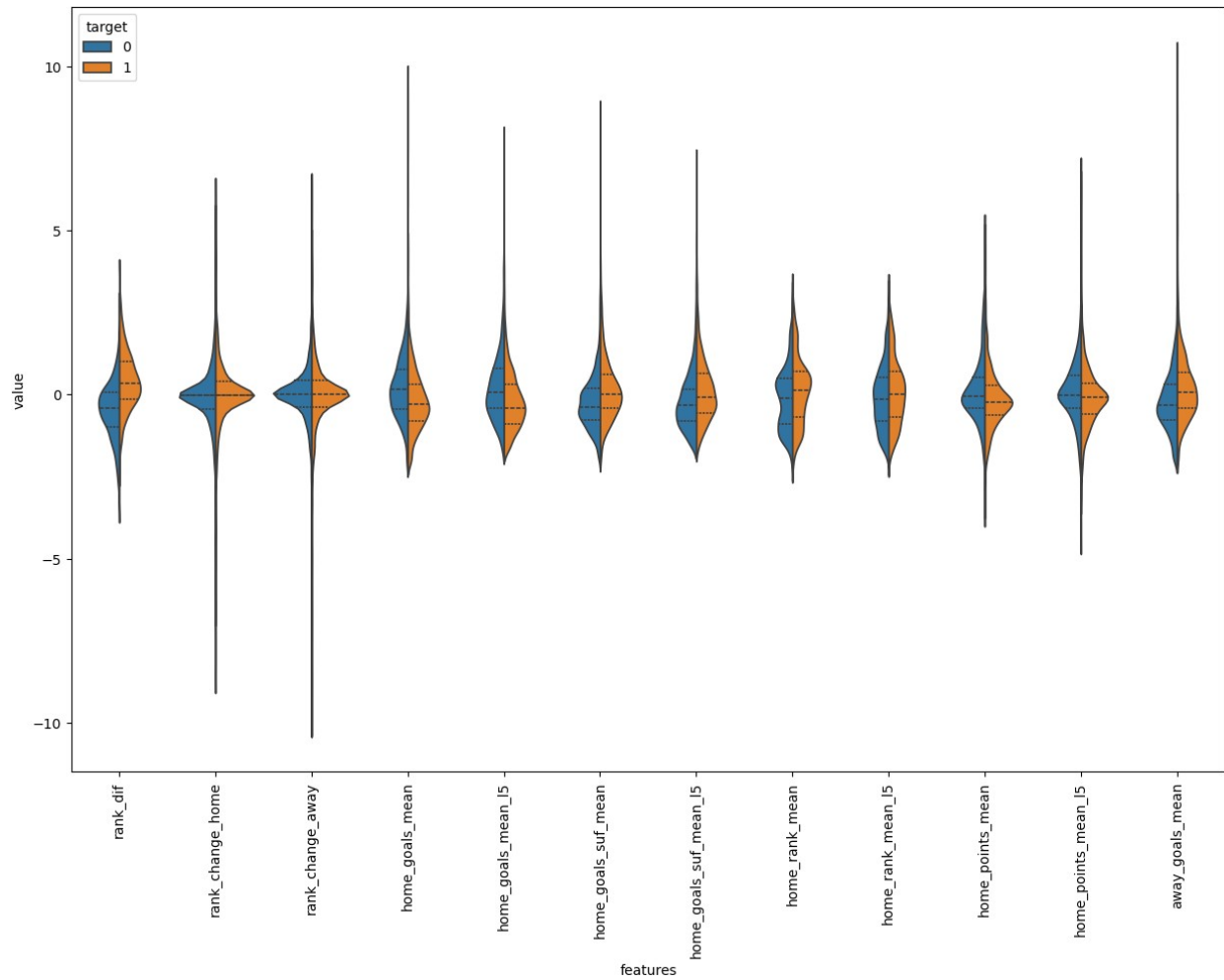
```
data1 = df[list(df.columns[8:20].values) + ["target"]]
data2 = df[df.columns[20:]]
```

```
scaled = (data1[:-1] - data1[:-1].mean()) / data1[:-1].std()
scaled["target"] = data1["target"]
violin1 = pd.melt(scaled, id_vars="target", var_name="features",
value_name="value")
```

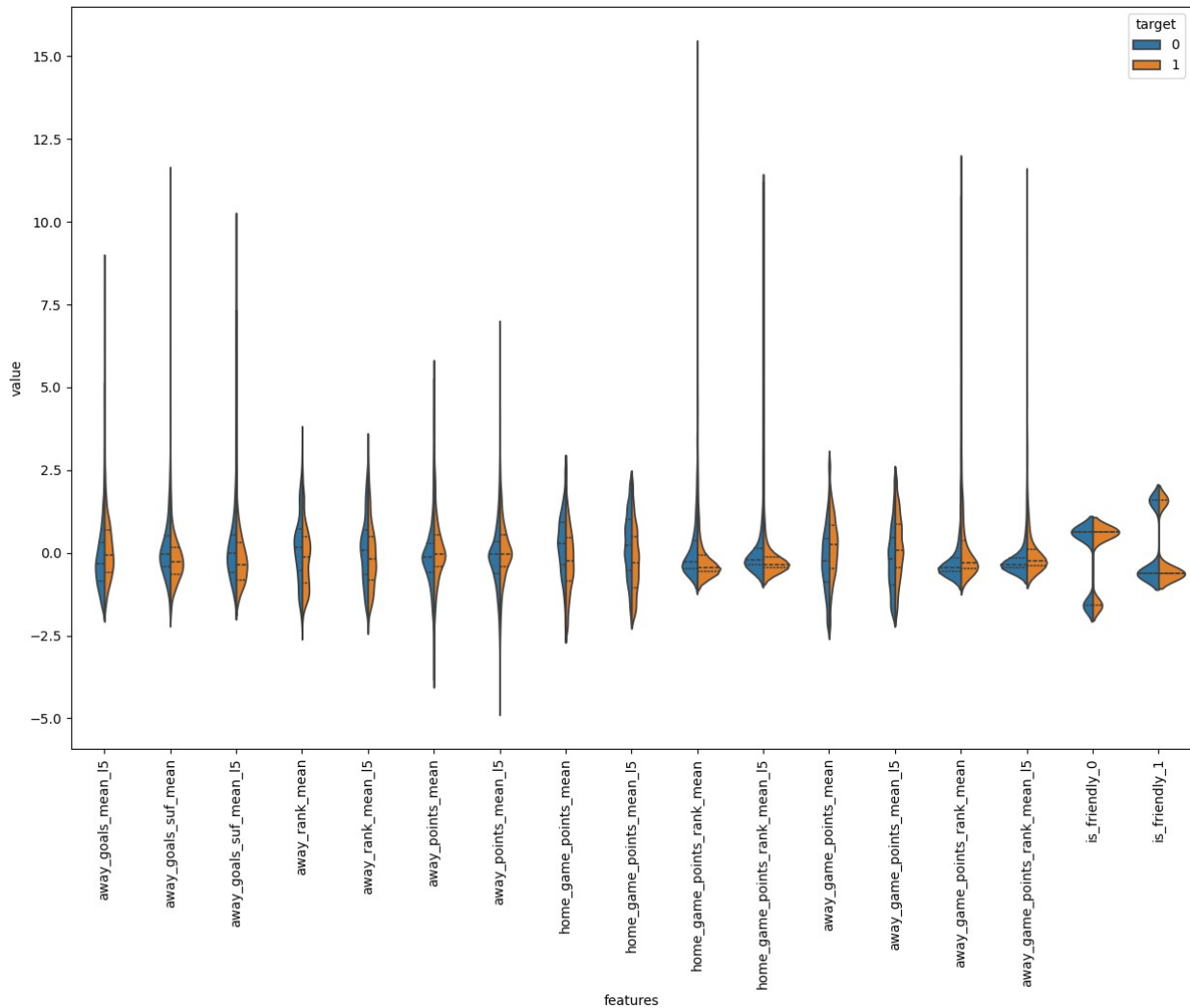
```
scaled = (data2[:-1] - data2[:-1].mean()) / data2[:-1].std()
scaled["target"] = data2["target"]
violin2 = pd.melt(scaled, id_vars="target", var_name="features",
value_name="value")
```

```
plt.figure(figsize=(15,10))
sns.violinplot(x="features", y="value", hue="target",
data=violin1,split=True, inner="quart")
```

```
plt.xticks(rotation=90)
plt.show()
```



```
plt.figure(figsize=(15,10))
sns.violinplot(x="features", y="value", hue="target",
data=violin2,split=True, inner="quart")
plt.xticks(rotation=90)
plt.show()
```



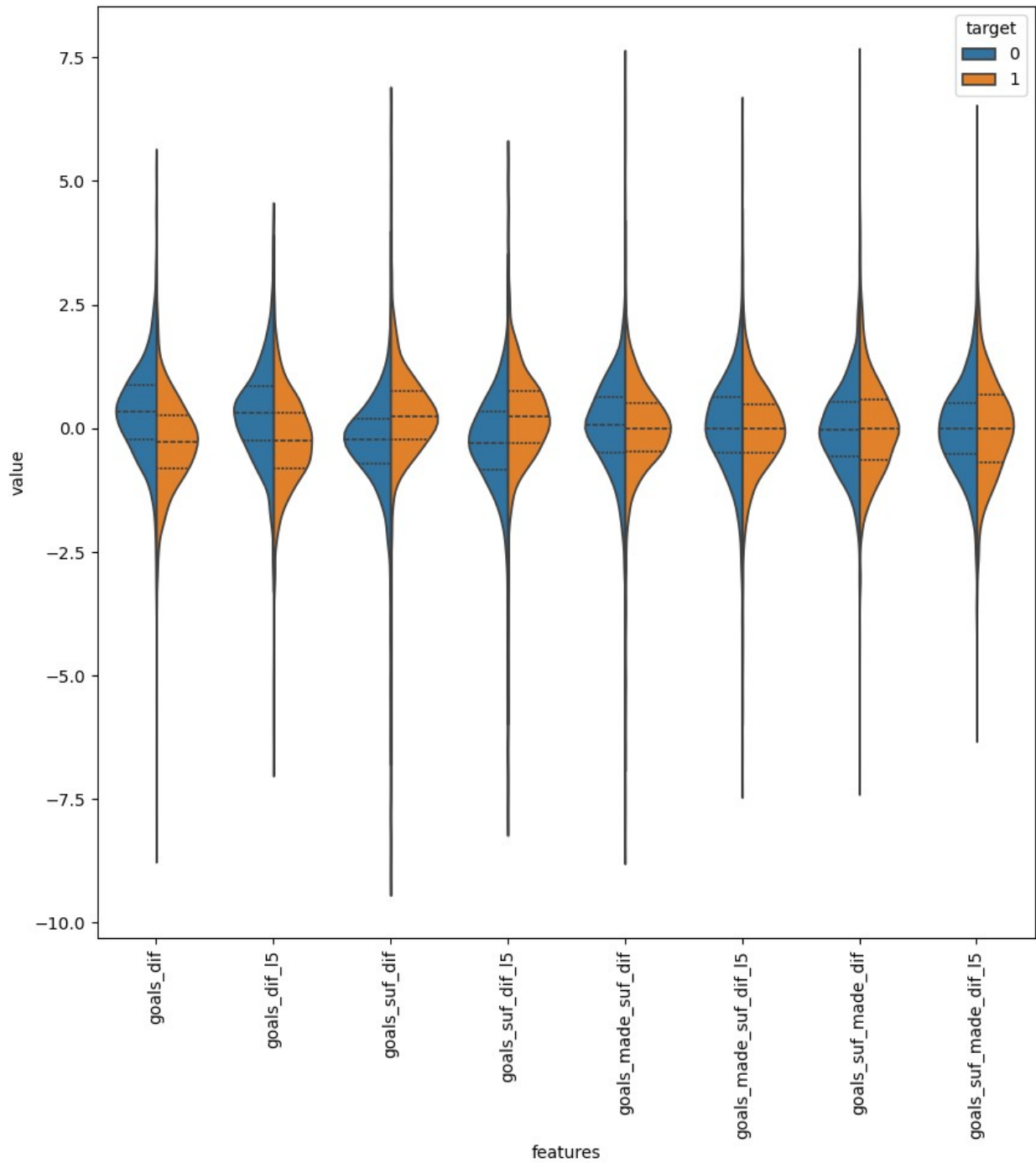
```

dif = df.copy()
dif.loc[:, "goals_dif"] = dif["home_goals_mean"] -
dif["away_goals_mean"]
dif.loc[:, "goals_dif_l5"] = dif["home_goals_mean_l5"] -
dif["away_goals_mean_l5"]
dif.loc[:, "goals_suf_dif"] = dif["home_goals_suf_mean"] -
dif["away_goals_suf_mean"]
dif.loc[:, "goals_suf_dif_l5"] = dif["home_goals_suf_mean_l5"] -
dif["away_goals_suf_mean_l5"]
dif.loc[:, "goals_made_suf_dif"] = dif["home_goals_mean"] -
dif["away_goals_suf_mean"]
dif.loc[:, "goals_made_suf_dif_l5"] = dif["home_goals_mean_l5"] -
dif["away_goals_suf_mean_l5"]
dif.loc[:, "goals_suf_made_dif"] = dif["home_goals_suf_mean"] -
dif["away_goals_mean"]
dif.loc[:, "goals_suf_made_dif_l5"] = dif["home_goals_suf_mean_l5"] -
dif["away_goals_mean_l5"]

```

```
data_difs = dif.iloc[:, -8:]
scaled = (data_difs - data_difs.mean()) / data_difs.std()
scaled["target"] = data2["target"]
violin = pd.melt(scaled, id_vars="target", var_name="features",
value_name="value")

plt.figure(figsize=(10,10))
sns.violinplot(x="features", y="value", hue="target",
data=violin, split=True, inner="quart")
plt.xticks(rotation=90)
plt.show()
```



```

dif.loc[:, "dif_points"] = dif["home_game_points_mean"] -
dif["away_game_points_mean"]
dif.loc[:, "dif_points_l5"] = dif["home_game_points_mean_l5"] -
dif["away_game_points_mean_l5"]
dif.loc[:, "dif_points_rank"] = dif["home_game_points_rank_mean"] -
dif["away_game_points_rank_mean"]
dif.loc[:, "dif_points_rank_l5"] =

```

```

dif["home_game_points_rank_mean_l5"] -
dif["away_game_points_rank_mean_l5"]

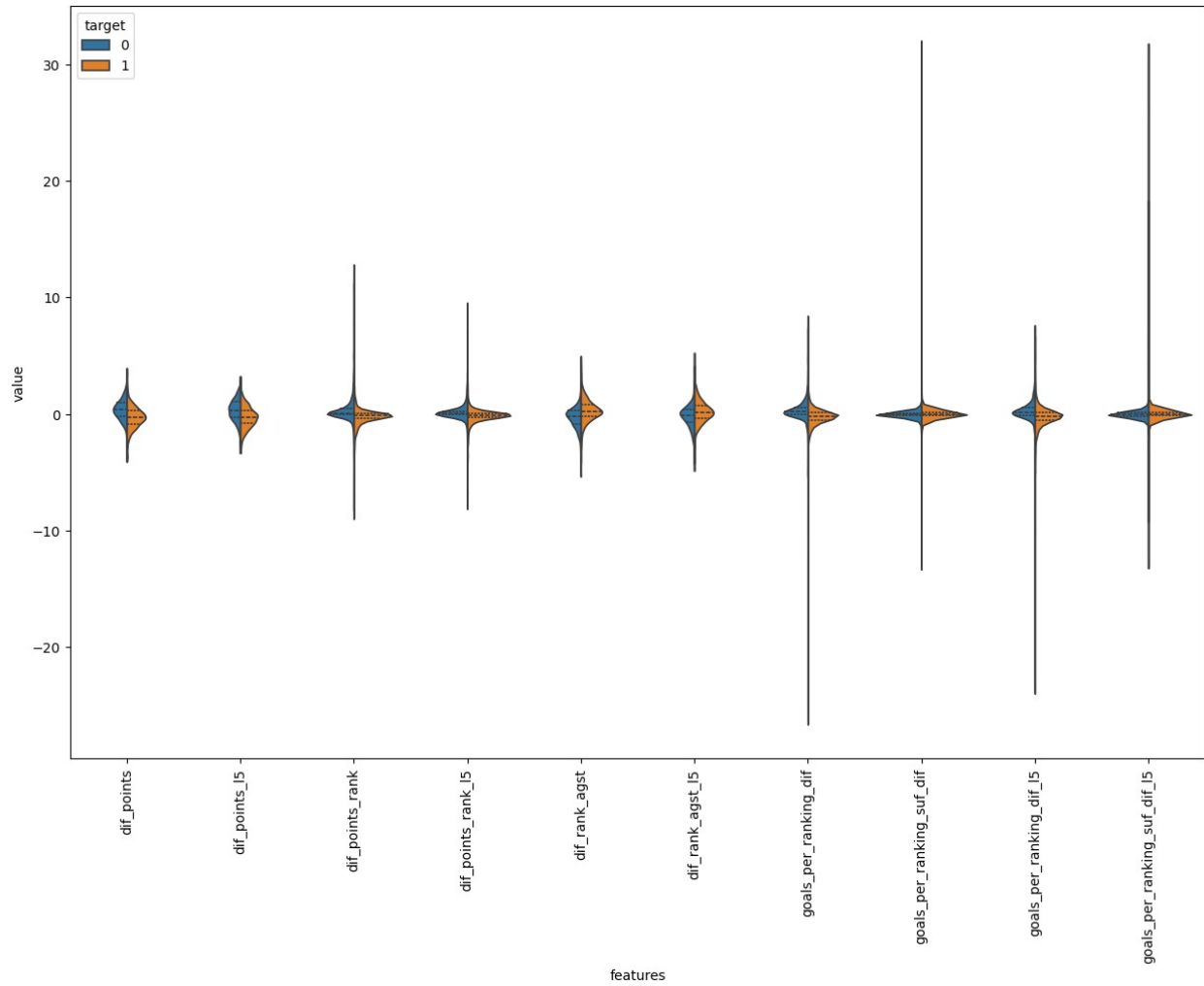
dif.loc[:, "dif_rank_agst"] = dif["home_rank_mean"] -
dif["away_rank_mean"]
dif.loc[:, "dif_rank_agst_l5"] = dif["home_rank_mean_l5"] -
dif["away_rank_mean_l5"]

dif.loc[:, "goals_per_ranking_dif"] = (dif["home_goals_mean"] /
dif["home_rank_mean"]) - (dif["away_goals_mean"] /
dif["away_rank_mean"])
dif.loc[:, "goals_per_ranking_suf_dif"] = (dif["home_goals_suf_mean"]
/ dif["home_rank_mean"]) - (dif["away_goals_suf_mean"] /
dif["away_rank_mean"])
dif.loc[:, "goals_per_ranking_dif_l5"] = (dif["home_goals_mean_l5"] /
dif["home_rank_mean"]) - (dif["away_goals_mean_l5"] /
dif["away_rank_mean"])
dif.loc[:, "goals_per_ranking_suf_dif_l5"] =
(dif["home_goals_suf_mean_l5"] / dif["home_rank_mean"]) -
(dif["away_goals_suf_mean_l5"] / dif["away_rank_mean"])

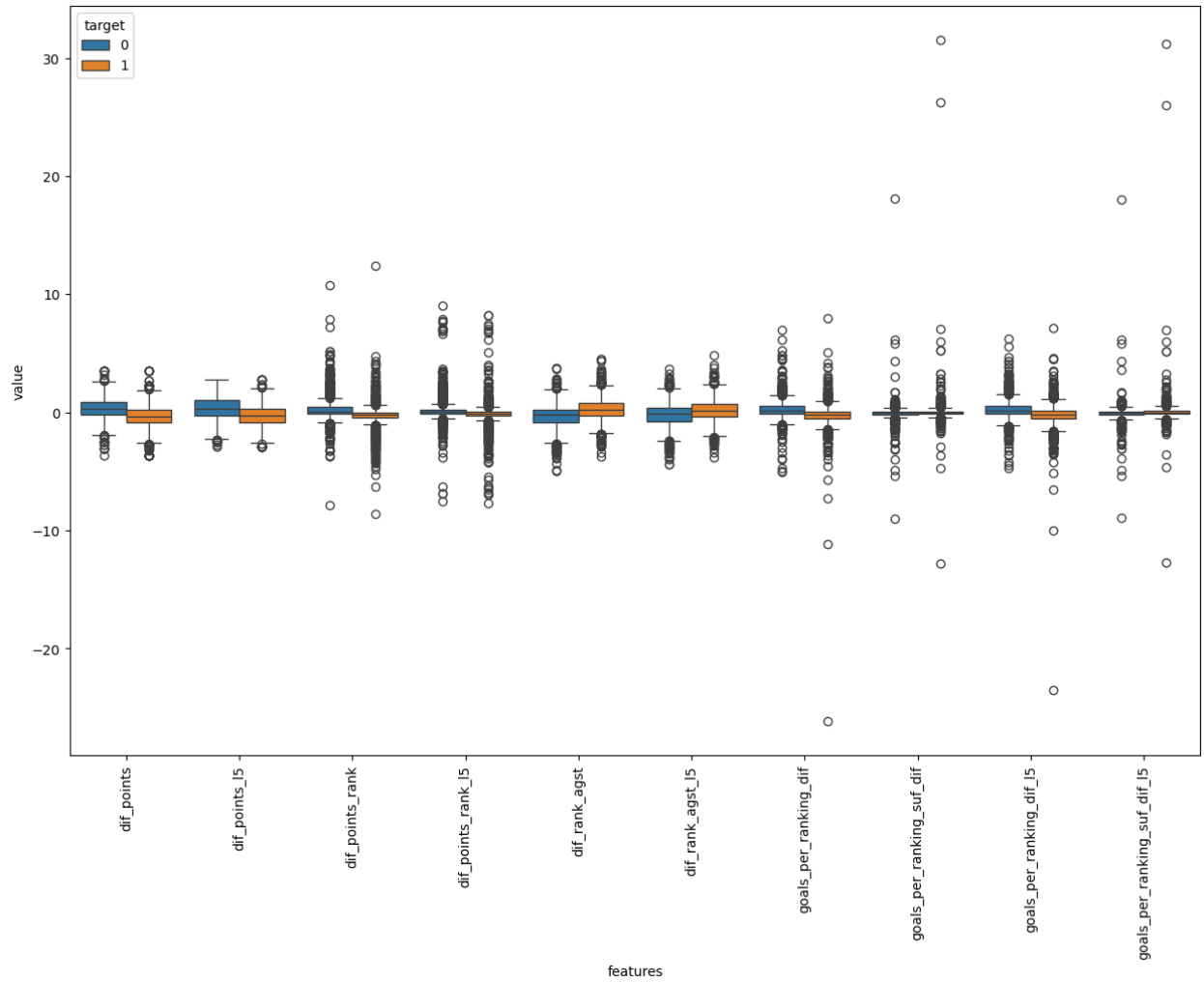
data_difs = dif.iloc[:, -10:]
scaled = (data_difs - data_difs.mean()) / data_difs.std()
scaled["target"] = data2["target"]
violin = pd.melt(scaled, id_vars="target", var_name="features",
value_name="value")

plt.figure(figsize=(15,10))
sns.violinplot(x="features", y="value", hue="target",
data=violin, split=True, inner="quart")
plt.xticks(rotation=90)
plt.show()

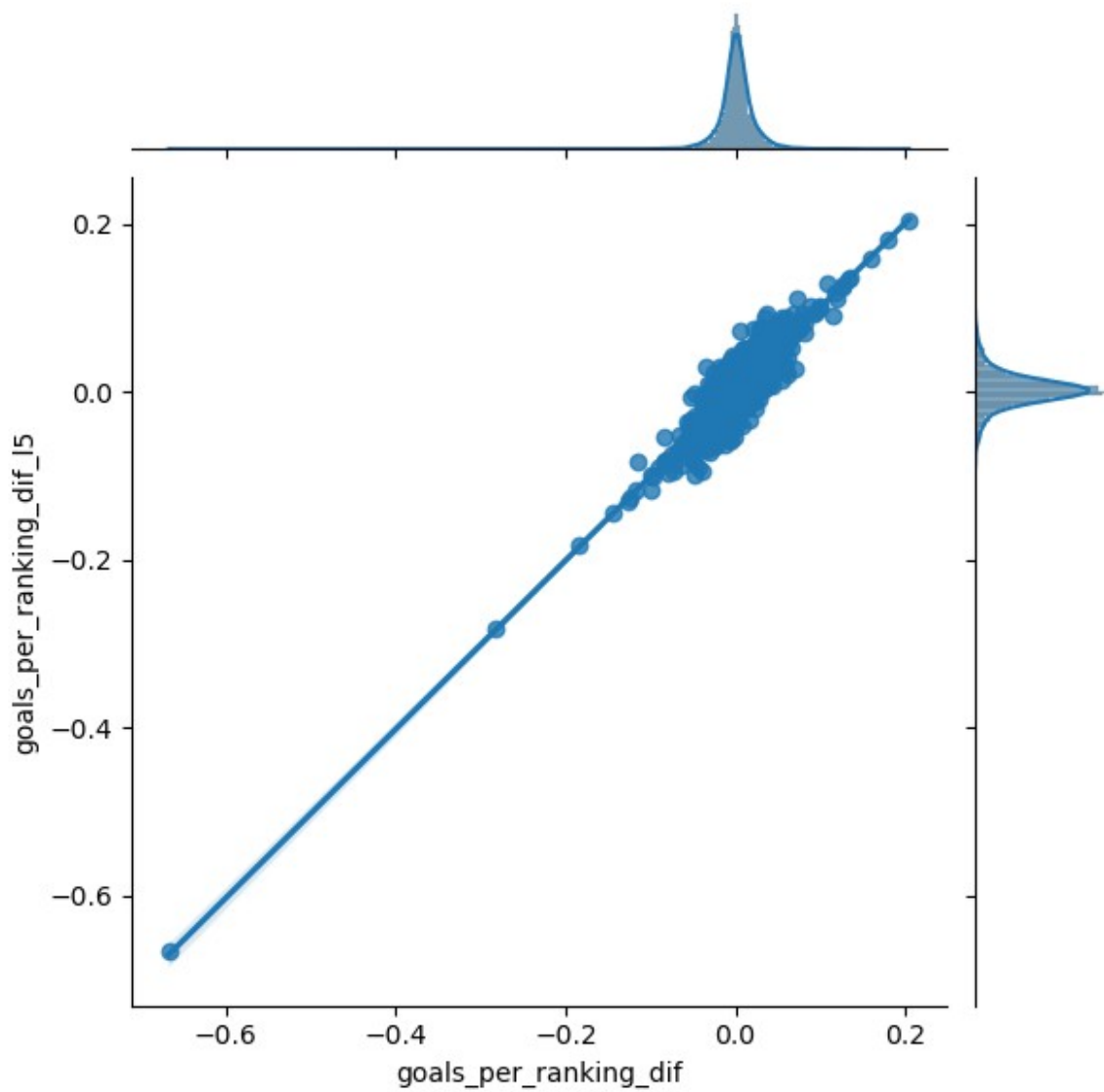
```

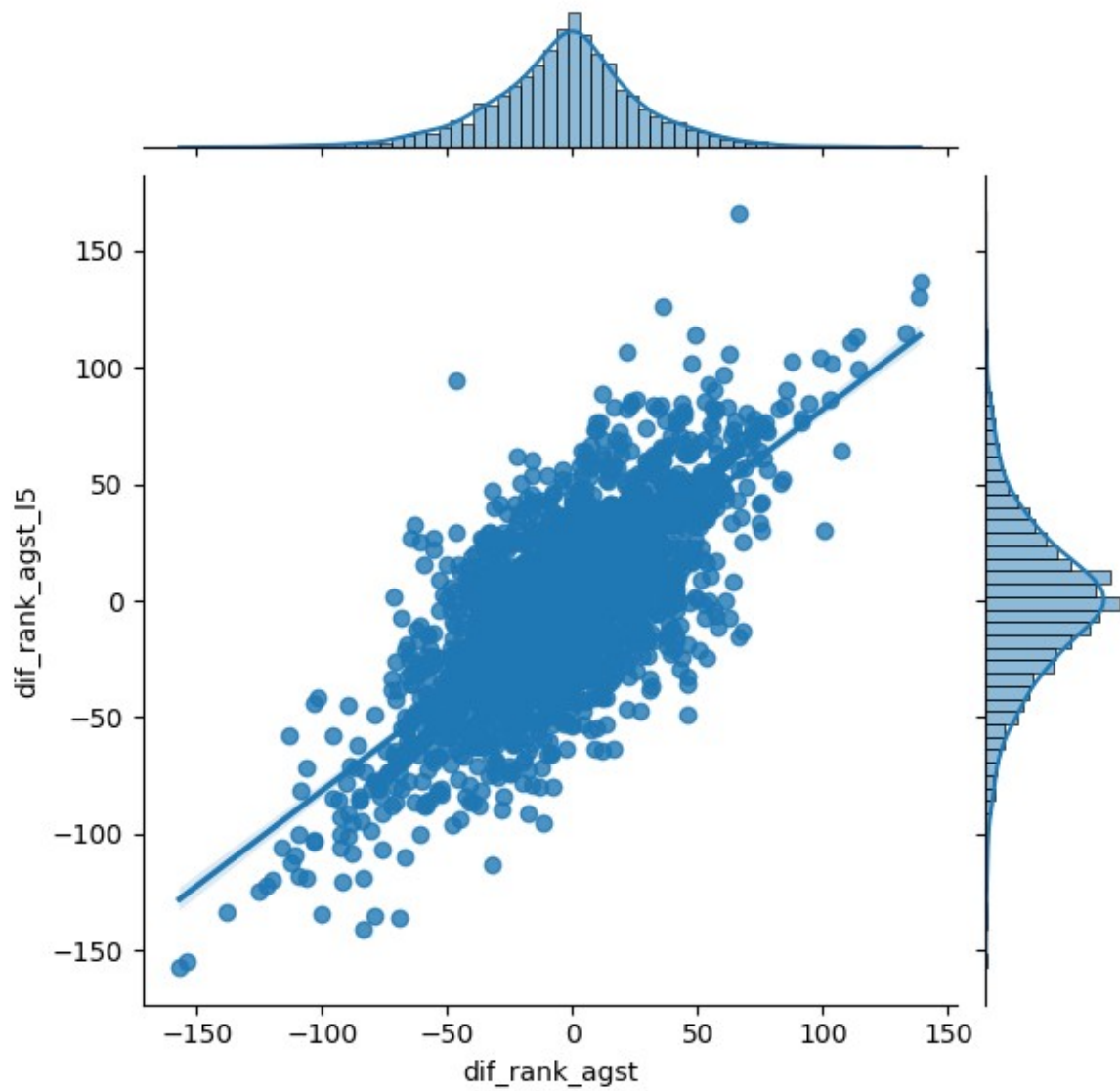
```
plt.figure(figsize=(15,10))
sns.boxplot(x="features", y="value", hue="target", data=violin)
plt.xticks(rotation=90)
plt.show()
```



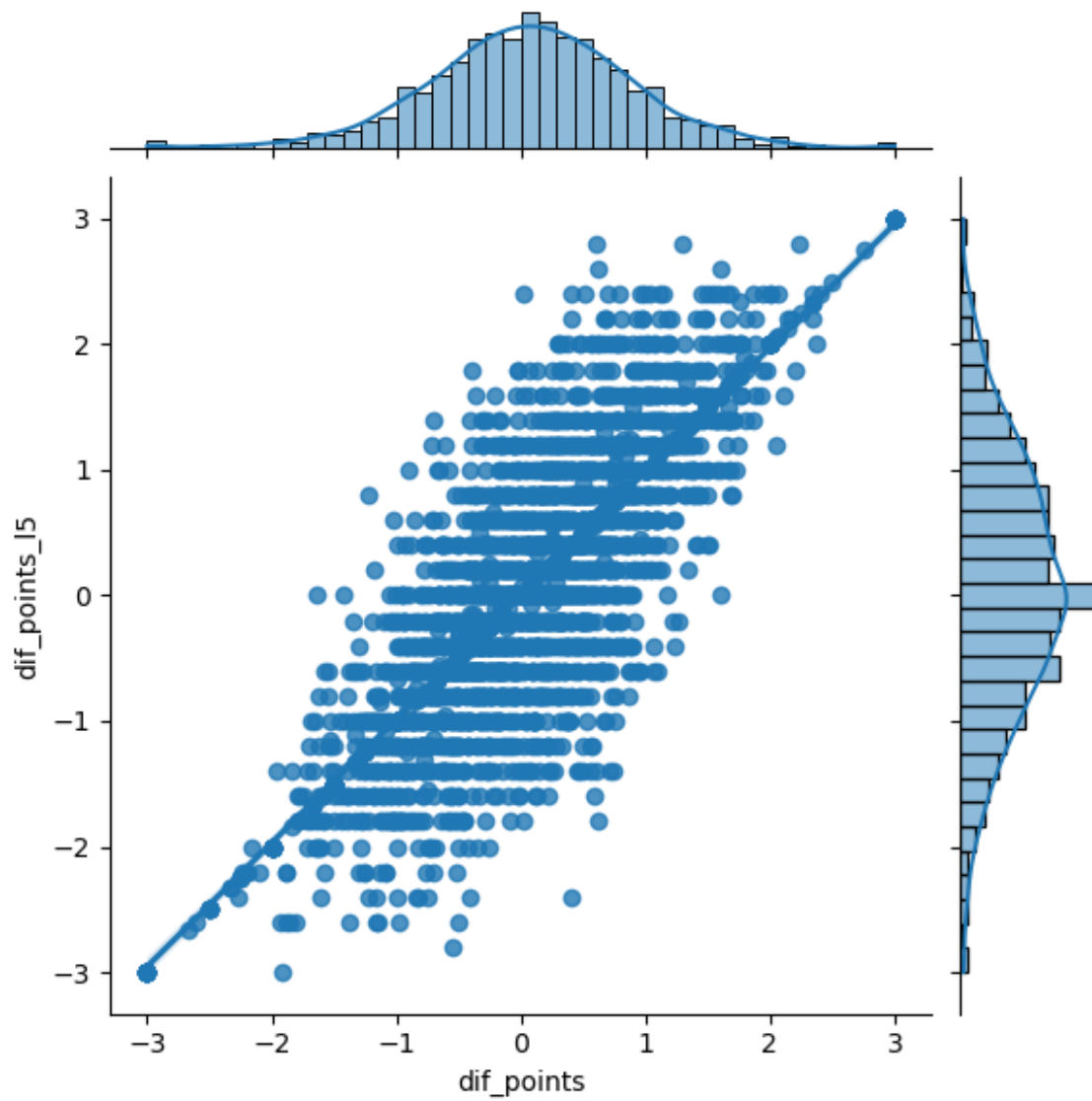
```
sns.jointplot(data = data_difs, x = 'goals_per_ranking_dif', y =
'goals_per_ranking_dif_l5', kind="reg")
plt.show()
```



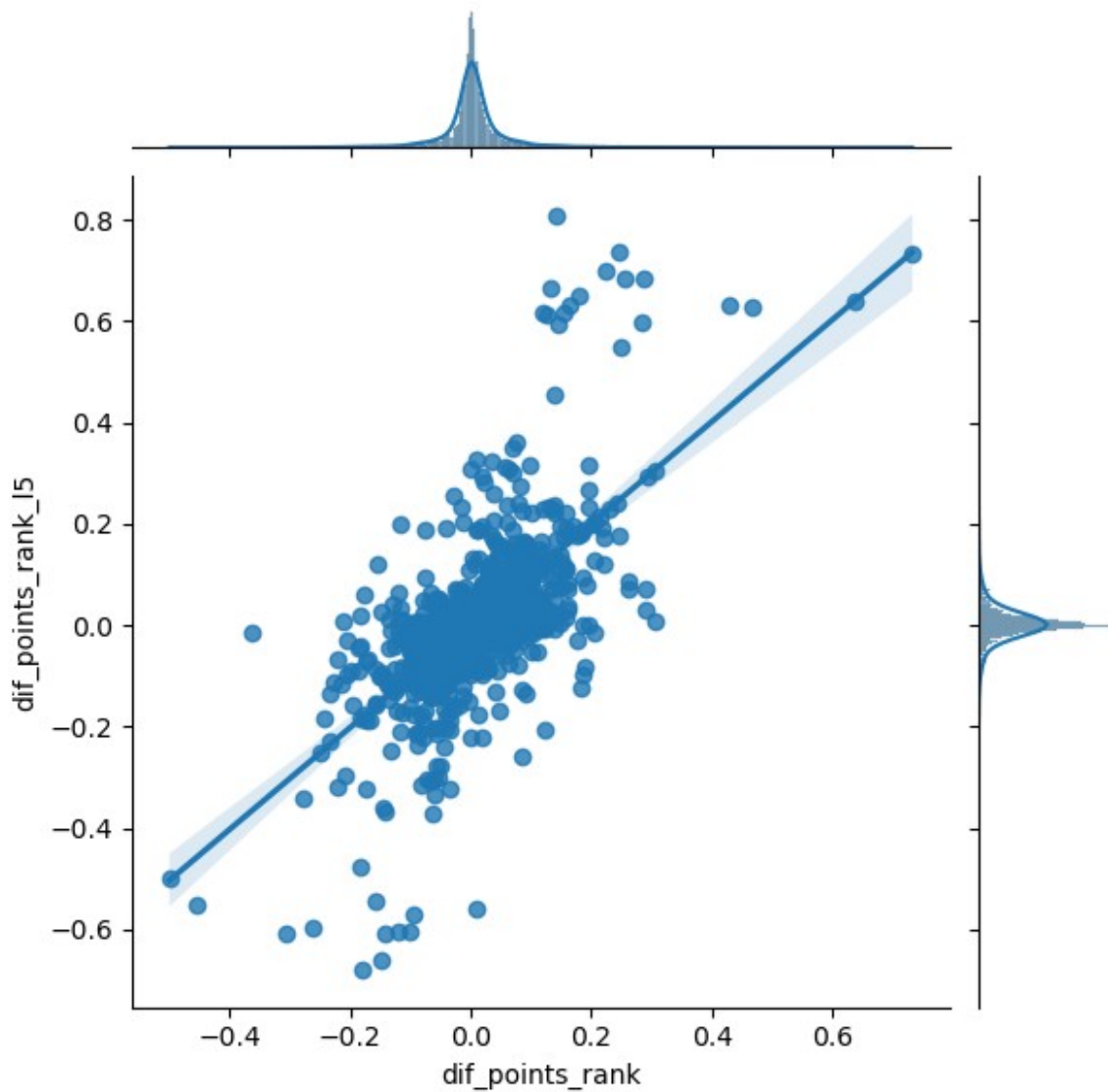
```
sns.jointplot(data = data_difs, x = 'dif_rank_agst', y =  
'dif_rank_agst_l5', kind="reg")  
plt.show()
```



```
sns.jointplot(data = data_difs, x = 'dif_points', y = 'dif_points_l5',  
kind="reg")  
plt.show()
```



```
sns.jointplot(data = data_difs, x = 'dif_points_rank', y =  
'dif_points_rank_l5', kind="reg")  
plt.show()
```



```
def create_db(df):
    columns = ["home_team", "away_team", "target", "rank_dif",
               "home_goals_mean", "home_rank_mean", "away_goals_mean",
               "away_rank_mean", "home_rank_mean_l5", "away_rank_mean_l5",
               "home_goals_suf_mean", "away_goals_suf_mean", "home_goals_mean_l5",
               "away_goals_mean_l5", "home_goals_suf_mean_l5",
               "away_goals_suf_mean_l5", "home_game_points_rank_mean",
               "home_game_points_rank_mean_l5", "away_game_points_rank_mean",
               "away_game_points_rank_mean_l5", "is_friendly_0", "is_friendly_1"]

    base = df.loc[:, columns]
    base.loc[:, "goals_dif"] = base["home_goals_mean"] -
base["away_goals_mean"]
    base.loc[:, "goals_dif_l5"] = base["home_goals_mean_l5"] -
base["away_goals_mean_l5"]
```

```

    base.loc[:, "goals_suf_dif"] = base["home_goals_suf_mean"] -
base["away_goals_suf_mean"]
    base.loc[:, "goals_suf_dif_l5"] = base["home_goals_suf_mean_l5"] -
base["away_goals_suf_mean_l5"]
    base.loc[:, "goals_per_ranking_dif"] = (base["home_goals_mean"] /
base["home_rank_mean"]) - (base["away_goals_mean"] /
base["away_rank_mean"])
    base.loc[:, "dif_rank_agst"] = base["home_rank_mean"] -
base["away_rank_mean"]
    base.loc[:, "dif_rank_agst_l5"] = base["home_rank_mean_l5"] -
base["away_rank_mean_l5"]
    base.loc[:, "dif_points_rank"] =
base["home_game_points_rank_mean"] -
base["away_game_points_rank_mean"]
    base.loc[:, "dif_points_rank_l5"] =
base["home_game_points_rank_mean_l5"] -
base["away_game_points_rank_mean_l5"]

    model_df = base[["home_team", "away_team", "target", "rank_dif",
"goals_dif", "goals_dif_l5", "goals_suf_dif", "goals_suf_dif_l5",
"goals_per_ranking_dif", "dif_rank_agst", "dif_rank_agst_l5",
"dif_points_rank", "dif_points_rank_l5", "is_friendly_0",
"is_friendly_1"]]
    return model_df

```

```
model_db = create_db(df)
```

```
model_db
```

	home_team	away_team	target	rank_dif	goals_dif	\
26	Spain	Ukraine	0	-16.0	-1.000000	
27	Switzerland	Germany	1	-3.0	0.000000	
28	Hungary	Russia	1	14.0	-2.000000	
29	Serbia	Turkey	1	0.0	1.000000	
30	Wales	Bulgaria	0	-36.0	0.000000	
...	
3058	Scotland	Northern Ireland	1	-40.0	0.490306	
3059	Senegal	Benin	0	-81.0	0.612174	
3060	Slovenia	Portugal	0	48.0	-1.039337	
3061	Spain	Brazil	1	3.0	0.086890	
3062	Finland	Estonia	0	-63.0	0.291709	

	goals_dif_l5	goals_suf_dif	goals_suf_dif_l5	
goals_per_ranking_dif				\
26	-1.0	0.000000	0.0	-
0.100000				
27	0.0	1.000000	1.0	-
0.083333				
28	-2.0	-1.000000	-1.0	-
0.068966				

29	1.0	2.000000	2.0
0.026316			
30	0.0	-1.000000	-1.0
0.012170			
...
...			
3058	0.0	-0.111320	1.8
0.012210			
3059	0.8	-0.322609	-1.4
0.008433			
3060	-2.0	0.228778	0.2
0.034331			
3061	1.2	0.077744	-0.6
0.012174			
3062	1.0	-0.481800	-0.4
0.009597			

	dif_rank_agst	dif_rank_agst_l5	dif_points_rank	
dif_points_rank_l5 \				
26	3.000000	3.0	-0.183333	-
0.183333				
27	16.000000	16.0	-0.125000	-
0.125000				
28	0.000000	0.0	0.000000	
0.000000				
29	-14.000000	-14.0	0.000000	
0.000000				
30	24.000000	24.0	0.022312	
0.022312				
...	
...				
3058	-13.773609	-51.2	0.027132	-
0.031528				
3059	-14.836522	-21.6	0.034176	
0.031098				
3060	33.734990	-5.8	-0.063854	
0.008337				
3061	8.534045	39.4	-0.018616	-
0.157732				
3062	-20.936299	50.8	0.047881	
0.009113				

	is_friendly_0	is_friendly_1
26	True	False
27	True	False
28	True	False
29	True	False
30	True	False
...

3058	False	True
3059	False	True
3060	False	True
3061	False	True
3062	False	True

[2937 rows x 15 columns]

```
X = model_db.iloc[:, 3:]
y = model_db[["target"]]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.2, random_state=1)
```

```
gb = GradientBoostingClassifier(random_state=5)
```

```
params = {"learning_rate": [0.01, 0.1, 0.5],
          "min_samples_split": [5, 10],
          "min_samples_leaf": [3, 5],
          "max_depth": [3, 5, 10],
          "max_features": ["sqrt"],
          "n_estimators": [100, 200]
        }
```

```
gb_cv = GridSearchCV(gb, params, cv = 3, n_jobs = -1, verbose = False)
```

```
gb_cv.fit(X_train.values, np.ravel(y_train))
```

```
GridSearchCV(cv=3,
estimator=GradientBoostingClassifier(random_state=5),
          n_jobs=-1,
          param_grid={'learning_rate': [0.01, 0.1, 0.5],
                      'max_depth': [3, 5, 10], 'max_features':
['sqrt'],
                      'min_samples_leaf': [3, 5],
                      'min_samples_split': [5, 10],
                      'n_estimators': [100, 200]},
          verbose=False)
```

```
gb = gb_cv.best_estimator_
```

```
gb
```

```
GradientBoostingClassifier(learning_rate=0.01, max_depth=5,
max_features='sqrt',
                        min_samples_leaf=3, min_samples_split=5,
                        n_estimators=200, random_state=5)
```

```
params_rf = {"max_depth": [20],
             "min_samples_split": [10],
             "max_leaf_nodes": [175],
```

```

        "min_samples_leaf": [5],
        "n_estimators": [250],
        "max_features": ["sqrt"],
    }

rf = RandomForestClassifier(random_state=1)

rf_cv = GridSearchCV(rf, params_rf, cv = 3, n_jobs = -1, verbose =
False)

rf_cv.fit(X_train.values, np.ravel(y_train))

GridSearchCV(cv=3, estimator=RandomForestClassifier(random_state=1),
n_jobs=-1,
            param_grid={'max_depth': [20], 'max_features': ['sqrt'],
                        'max_leaf_nodes': [175], 'min_samples_leaf':
[5],
                        'min_samples_split': [10], 'n_estimators':
[250]}},
            verbose=False)

rf = rf_cv.best_estimator_

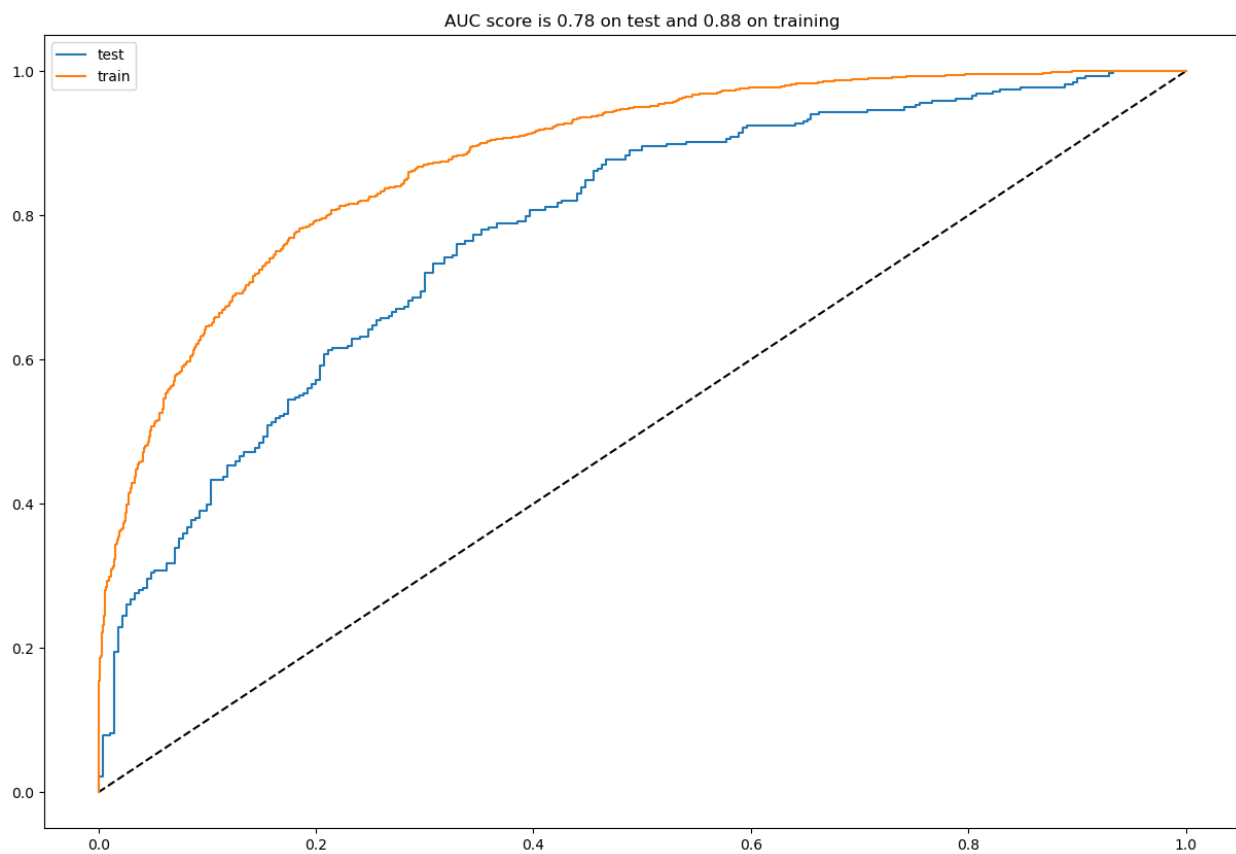
def analyze(model):
    fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test.values)
[:,1]) #test AUC
    plt.figure(figsize=(15,10))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr, label="test")

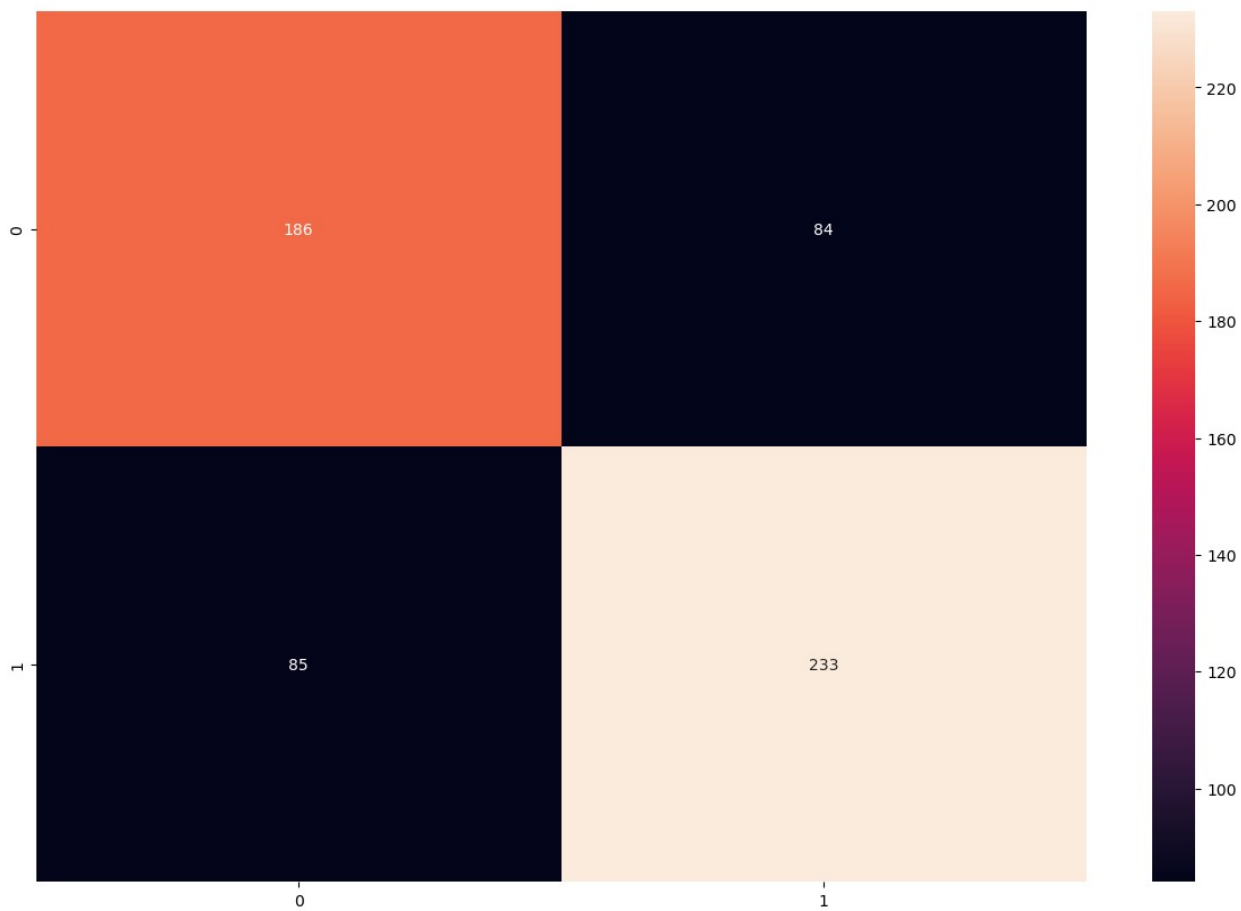
    fpr_train, tpr_train, _ = roc_curve(y_train,
model.predict_proba(X_train.values)[:,1]) #train AUC
    plt.plot(fpr_train, tpr_train, label="train")
    auc_test = roc_auc_score(y_test,
model.predict_proba(X_test.values)[:,1])
    auc_train = roc_auc_score(y_train,
model.predict_proba(X_train.values)[:,1])
    plt.legend()
    plt.title('AUC score is %.2f on test and %.2f on training'%
(auc_test, auc_train))
    plt.show()

    plt.figure(figsize=(15, 10))
    cm = confusion_matrix(y_test, model.predict(X_test.values))
    sns.heatmap(cm, annot=True, fmt="d")

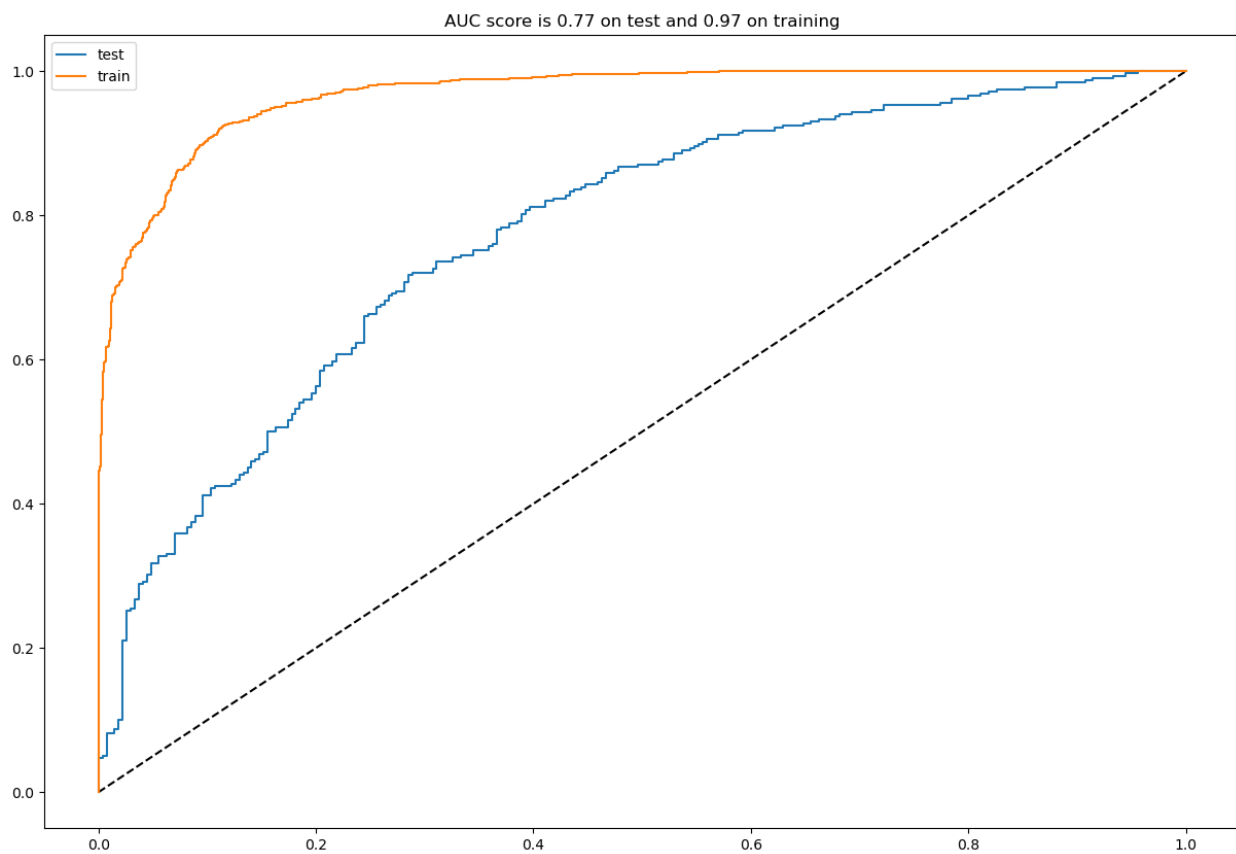
analyze(gb)

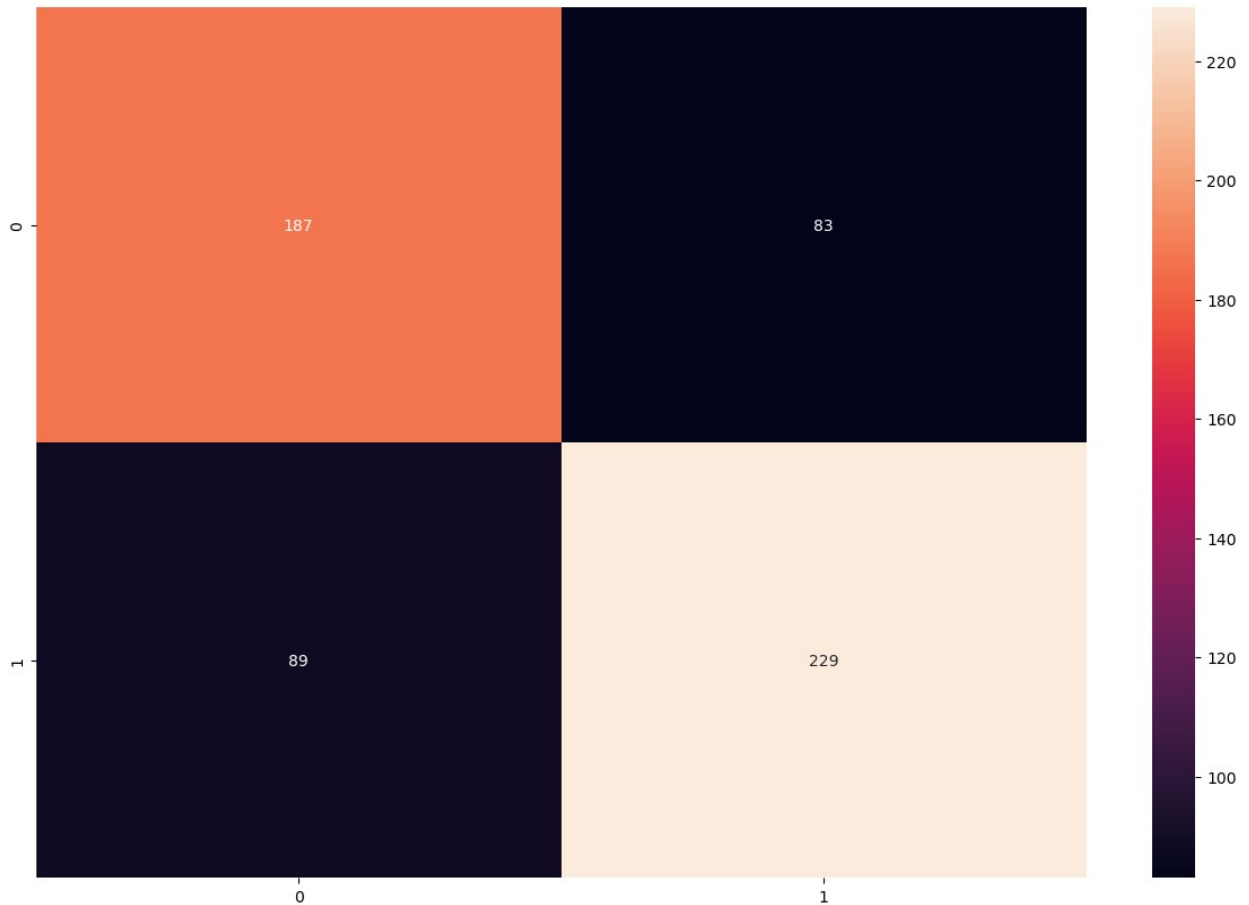
```





`analyze(rf)`





```
def create_table_with_points(teams):
    groups = ['A', 'B', 'C', 'D']
    table = {group: [[team, 0, []] for team in teams[group]] for group
in groups}
    return table

teams = {
    'A': ['Argentina', 'Peru', 'Chile', 'Canada'],
    'B': ['Mexico', 'Ecuador', 'Venezuela', 'Jamaica'],
    'C': ['United States', 'Uruguay', 'Panama', 'Bolivia'],
    'D': ['Brazil', 'Colombia', 'Paraguay', 'Costa Rica']
}

table = create_table_with_points(teams)

table
{'A': [['Argentina', 0, []],
      ['Peru', 0, []],
      ['Chile', 0, []],
      ['Canada', 0, []]],
 'B': [['Mexico', 0, []],
```

```

    ['Ecuador', 0, []],
    ['Venezuela', 0, []],
    ['Jamaica', 0, []]],
    'C': [['United States', 0, []],
    ['Uruguay', 0, []],
    ['Panama', 0, []],
    ['Bolivia', 0, []]],
    'D': [['Brazil', 0, []],
    ['Colombia', 0, []],
    ['Paraguay', 0, []],
    ['Costa Rica', 0, []]]}

matches = []
for group, teams in table.items():
    for i in range(len(teams)):
        for j in range(i+1, len(teams)):
            matches.append((group, teams[i][0], teams[j][0]))

matches[:5]

[('A', 'Argentina', 'Peru'),
 ('A', 'Argentina', 'Chile'),
 ('A', 'Argentina', 'Canada'),
 ('A', 'Peru', 'Chile'),
 ('A', 'Peru', 'Canada')]

base_df = team_stats

def find_stats(team_1):
    past_games = team_stats_raw[(team_stats_raw["team"] ==
team_1)].sort_values("date")
    last5 = team_stats_raw[(team_stats_raw["team"] ==
team_1)].sort_values("date").tail(5)

    team_1_rank = past_games["rank"].values[-1]
    team_1_goals = past_games.score.mean()
    team_1_goals_l5 = last5.score.mean()
    team_1_goals_suf = past_games.suf_score.mean()
    team_1_goals_suf_l5 = last5.suf_score.mean()
    team_1_rank_suf = past_games.rank_suf.mean()
    team_1_rank_suf_l5 = last5.rank_suf.mean()
    team_1_gp_rank = past_games.points_by_rank.mean()
    team_1_gp_rank_l5 = last5.points_by_rank.mean()

    return [team_1_rank, team_1_goals, team_1_goals_l5,
team_1_goals_suf, team_1_goals_suf_l5, team_1_rank_suf,
team_1_rank_suf_l5, team_1_gp_rank, team_1_gp_rank_l5]

def find_features(team_1, team_2):
    rank_dif = team_1[0] - team_2[0]
    goals_dif = team_1[1] - team_2[1]

```

```

goals_dif_l5 = team_1[2] - team_2[2]
goals_suf_dif = team_1[3] - team_2[3]
goals_suf_dif_l5 = team_1[4] - team_2[4]
goals_per_ranking_dif = (team_1[1]/team_1[5]) -
(team_2[1]/team_2[5])
dif_rank_agst = team_1[5] - team_2[5]
dif_rank_agst_l5 = team_1[6] - team_2[6]
dif_gp_rank = team_1[7] - team_2[7]
dif_gp_rank_l5 = team_1[8] - team_2[8]

    return [rank_dif, goals_dif, goals_dif_l5, goals_suf_dif,
goals_suf_dif_l5, goals_per_ranking_dif, dif_rank_agst,
dif_rank_agst_l5, dif_gp_rank, dif_gp_rank_l5, 1, 0]

import numpy as np
from operator import itemgetter

advanced_group = []
last_group = ""
thresh = 0.05

for k in table.keys():
    for t in table[k]:
        t[1] = 0
        t[2] = []

for teams in matches:
    draw = False
    team_1 = find_stats(teams[1])
    team_2 = find_stats(teams[2])

    features_g1 = find_features(team_1, team_2)
    features_g2 = find_features(team_2, team_1)

    probs_g1 = gb.predict_proba([features_g1])
    probs_g2 = gb.predict_proba([features_g2])

    team_1_prob_g1 = probs_g1[0][0]
    team_1_prob_g2 = probs_g2[0][1]
    team_2_prob_g1 = probs_g1[0][1]
    team_2_prob_g2 = probs_g2[0][0]

    team_1_prob = (probs_g1[0][0] + probs_g2[0][1]) / 2
    team_2_prob = (probs_g2[0][0] + probs_g1[0][1]) / 2

    if ((team_1_prob_g1 > team_2_prob_g1) & (team_2_prob_g2 >
team_1_prob_g2)) | (
        (team_1_prob_g1 < team_2_prob_g1) & (team_2_prob_g2 <
team_1_prob_g2)):
        draw = True

```



```

        for i in table[teams[0]]:
            if i[0] == teams[1] or i[0] == teams[2]:
                i[1] += 1

    elif team_1_prob > team_2_prob:
        winner = teams[1]
        winner_proba = team_1_prob
        for i in table[teams[0]]:
            if i[0] == teams[1]:
                i[1] += 3

    elif team_2_prob > team_1_prob:
        winner = teams[2]
        winner_proba = team_2_prob
        for i in table[teams[0]]:
            if i[0] == teams[2]:
                i[1] += 3

    for i in table[teams[0]]:
        if i[0] == teams[1]:
            i[2].append(team_1_prob)
        if i[0] == teams[2]:
            i[2].append(team_2_prob)

    if last_group != teams[0]:
        if last_group != "":
            print("\n")
            print("Group %s advanced: " % (last_group))

            for i in table[last_group]:
                i[2] = np.mean(i[2])

            final_points = table[last_group]
            final_table = sorted(final_points, key=itemgetter(1, 2),
reverse=True)
            advanced_group.append([final_table[0][0], final_table[1]
[0]])

            for i in final_table:
                print("%s ----- %d" % (i[0], i[1]))
            print("\n")
            print("-" * 10 + " Starting Analysis for Group %s " %
(teams[0]) + "-" * 10)

        if draw == False:
            print("Group %s - %s vs. %s: Winner %s with %.2f probability"
% (teams[0], teams[1], teams[2], winner,
winner_proba))
        else:
            print("Group %s - %s vs. %s: Draw" % (teams[0], teams[1],

```

```

teams[2]))
    last_group = teams[0]

print("\n")
print("Group %s advanced: " % (last_group))

for i in table[last_group]:
    i[2] = np.mean(i[2])

final_points = table[last_group]
final_table = sorted(final_points, key=itemgetter(1, 2), reverse=True)
advanced_group.append([final_table[0][0], final_table[1][0]])
for i in final_table:
    print("%s ----- %d" % (i[0], i[1]))

```

```

----- Starting Analysis for Group A -----
Group A - Argentina vs. Peru: Winner Argentina with 0.78 probability
Group A - Argentina vs. Chile: Winner Argentina with 0.79 probability
Group A - Argentina vs. Canada: Winner Argentina with 0.73 probability
Group A - Peru vs. Chile: Draw
Group A - Peru vs. Canada: Winner Canada with 0.62 probability
Group A - Chile vs. Canada: Winner Canada with 0.64 probability

```

```

Group A advanced:
Argentina ----- 9
Canada ----- 6
Peru ----- 1
Chile ----- 1

```

```

----- Starting Analysis for Group B -----
Group B - Mexico vs. Ecuador: Draw
Group B - Mexico vs. Venezuela: Winner Mexico with 0.65 probability
Group B - Mexico vs. Jamaica: Winner Mexico with 0.69 probability
Group B - Ecuador vs. Venezuela: Winner Ecuador with 0.73 probability
Group B - Ecuador vs. Jamaica: Winner Ecuador with 0.70 probability
Group B - Venezuela vs. Jamaica: Draw

```

```

Group B advanced:
Ecuador ----- 7
Mexico ----- 7
Venezuela ----- 1
Jamaica ----- 1

```

```

----- Starting Analysis for Group C -----
Group C - United States vs. Uruguay: Draw

```

Group C - United States vs. Panama: Winner United States with 0.73 probability
Group C - United States vs. Bolivia: Winner United States with 0.81 probability
Group C - Uruguay vs. Panama: Winner Uruguay with 0.67 probability
Group C - Uruguay vs. Bolivia: Winner Uruguay with 0.74 probability
Group C - Panama vs. Bolivia: Winner Panama with 0.66 probability

Group C advanced:
United States ----- 7
Uruguay ----- 7
Panama ----- 3
Bolivia ----- 0

----- Starting Analysis for Group D -----
Group D - Brazil vs. Colombia: Winner Brazil with 0.58 probability
Group D - Brazil vs. Paraguay: Winner Brazil with 0.77 probability
Group D - Brazil vs. Costa Rica: Winner Brazil with 0.80 probability
Group D - Colombia vs. Paraguay: Winner Colombia with 0.72 probability
Group D - Colombia vs. Costa Rica: Winner Colombia with 0.77 probability
Group D - Paraguay vs. Costa Rica: Draw

Group D advanced:
Brazil ----- 9
Colombia ----- 6
Paraguay ----- 1
Costa Rica ----- 1

advanced = advanced_group

advanced

```
[['Argentina', 'Canada'],  
 ['Ecuador', 'Mexico'],  
 ['United States', 'Uruguay'],  
 ['Brazil', 'Colombia']]
```

```
playoffs = {"Quarter-Final": [], "Semi-Final": [], "Final": []}
```

```
for p in playoffs.keys():  
    playoffs[p] = []
```

```
actual_round = ""  
next_rounds = []
```

```
for p in playoffs.keys():  
    if p == "Quarter-Final":
```

```

control = []
for a in range(0, len(advanced)*2, 1):
    if a < len(advanced):
        if a % 2 == 0:
            control.append((advanced*2)[a][0])
        else:
            control.append((advanced*2)[a][1])
    else:
        if a % 2 == 0:
            control.append((advanced*2)[a][1])
        else:
            control.append((advanced*2)[a][0])

playoffs[p] = [[control[c], control[c+1]] for c in range(0,
len(control)-1, 1) if c%2 == 0]

for i in range(0, len(playoffs[p]), 1):
    game = playoffs[p][i]

    home = game[0]
    away = game[1]
    team_1 = find_stats(home)
    team_2 = find_stats(away)

    features_g1 = find_features(team_1, team_2)
    features_g2 = find_features(team_2, team_1)

    probs_g1 = gb.predict_proba([features_g1])
    probs_g2 = gb.predict_proba([features_g2])

    team_1_prob = (probs_g1[0][0] + probs_g2[0][1])/2
    team_2_prob = (probs_g2[0][0] + probs_g1[0][1])/2

    if actual_round != p:
        print("-"*10)
        print("Starting simulation of %s"%(p))
        print("-"*10)
        print("\n")

    if team_1_prob < team_2_prob:
        print("%s vs. %s: %s advances with prob %.2f"%(home,
away, away, team_2_prob))
        next_rounds.append(away)
    else:
        print("%s vs. %s: %s advances with prob %.2f"%(home,
away, home, team_1_prob))
        next_rounds.append(home)

    game.append([team_1_prob, team_2_prob])
    playoffs[p][i] = game

```

```

        actual_round = p
    else:
        playoffs[p] = [[next_rounds[c], next_rounds[c+1]] for c in
range(0, len(next_rounds)-1, 1) if c%2 == 0]
        next_rounds = []
        for i in range(0, len(playoffs[p])):
            game = playoffs[p][i]
            home = game[0]
            away = game[1]
            team_1 = find_stats(home)
            team_2 = find_stats(away)

            features_g1 = find_features(team_1, team_2)
            features_g2 = find_features(team_2, team_1)

            probs_g1 = gb.predict_proba([features_g1])
            probs_g2 = gb.predict_proba([features_g2])

            team_1_prob = (probs_g1[0][0] + probs_g2[0][1])/2
            team_2_prob = (probs_g2[0][0] + probs_g1[0][1])/2

            if actual_round != p:
                print("-"*10)
                print("Starting simulation of %s"%(p))
                print("-"*10)
                print("\n")

            if team_1_prob < team_2_prob:
                print("%s vs. %s: %s advances with prob %.2f"%(home,
away, away, team_2_prob))
                next_rounds.append(away)
            else:
                print("%s vs. %s: %s advances with prob %.2f"%(home,
away, home, team_1_prob))
                next_rounds.append(home)
            game.append([team_1_prob, team_2_prob])
            playoffs[p][i] = game
            actual_round = p

```

```

-----
Starting simulation of Quarter-Final
-----

```

```

Argentina vs. Mexico: Argentina advances with prob 0.67
United States vs. Colombia: Colombia advances with prob 0.54
Canada vs. Ecuador: Ecuador advances with prob 0.59
Uruguay vs. Brazil: Brazil advances with prob 0.55
-----

```

Starting simulation of Semi-Final

Argentina vs. Colombia: Argentina advances with prob 0.62

Ecuador vs. Brazil: Brazil advances with prob 0.72

Starting simulation of Final

Argentina vs. Brazil: Argentina advances with prob 0.56

```
plt.figure(figsize=(15, 10))
```

```
G = nx.balanced_tree(2, 2)
```

```
labels = []
```

```
for p in playoffs.keys():
    for game in playoffs[p]:
        label = f"{game[0]}({round(game[2][0], 2)}) \n {game[1]}
({round(game[2][1], 2)})"
        labels.append(label)
```

```
labels_dict = {}
```

```
labels_rev = list(reversed(labels))
```

```
for l in range(len(list(G.nodes))):
    labels_dict[l] = labels_rev[l]
```

```
pos = graphviz_layout(G, prog='twopi')
```

```
labels_pos = {n: (k[0], k[1] - 0.08 * k[1]) for n, k in pos.items()}
```

```
center = pd.DataFrame(pos).mean(axis=1).mean()
```

```
nx.draw(G, pos=pos, with_labels=False, node_color=range(len(G.nodes)),
edge_color="#bbf5bb", width=10,
```

```
font_weight='bold', cmap=plt.cm.Greens, node_size=5000)
```

```
nx.draw_networkx_labels(G, pos=labels_pos,
                        bbox=dict(boxstyle="round,pad=0.3",
fc="white", ec="black", lw=.5, alpha=1),
                        labels=labels_dict)
```

```
texts = ["Quarter \n Final", "Semi \n Final", "Final\n"]
```

```
pos_y = pos[0][1] + 55
```

```
for text in reversed(texts):
    pos_x = center
    pos_y -= 65
    plt.text(pos_x, pos_y, text, fontsize=18)
```

```
plt.axis('equal')
```

```
plt.show()
```

