VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

PREDICT THE WORLD CUP 2022

MEMBERS:
Nguyen Viet Anh - ITDSIU18027 - 0915111492
Chung Huu Tin - ITDSIU20097 - 0379004801

Ho Chi Minh City, Vietnam
2022-2023

Index

1. Topic Motivation

Football is known as the king of sports! First and foremost, football is easy. This is a fantastic illustration of the saying "simplicity is beautiful" in action. One goal for all 22 players: to put a ball into the net without touching it by hand. Although there are countless approaches a team might use to accomplish that goal, the fundamentals are still the same.
Second, football offers enormous scope for personal expression and originality. Each particular move, whether it be a lovely pass, a deft dribble, or a potent hit, may be distinguished by a particular element of flair. One merely needs to contrast Messi's and Neymar's dribbling techniques. While Messi and Neymar are both fantastic footballers, Neymar favors extremely artistic touches and plays, while Messi excels at rapid, short dribbles. There are various ways that each player may

use their unique abilities and influence the game. Every second on the field may be customized to your liking, whether it is through tackling, heading, or simple movement.
Thirdly, football is a mental as well as a physical struggle. Compromises, playing methods, tactical formations—the list is endless. Managers are essential to a team's performance because of how they execute tactics, make replacements, and coach their players. A team can have some highly talented players, but if there is no overarching plan that unifies them and inspires them to achieve, your team's efficacy will be constrained.
Last but not least, football is adored worldwide. Its widespread use is evidence of people and our remarkable skills. The game is played by billions of people, and it has brought joy and hope to countless individuals all over the world. (Except, of course, if your team consistently loses).

2. Abbreviations

(Google)Colab: Colaboratory
Rand: random forest
NA: Not Available
Corr: correlation
WC: World Cup
Euro: Europe

3. Pre-processing and processed collection data.

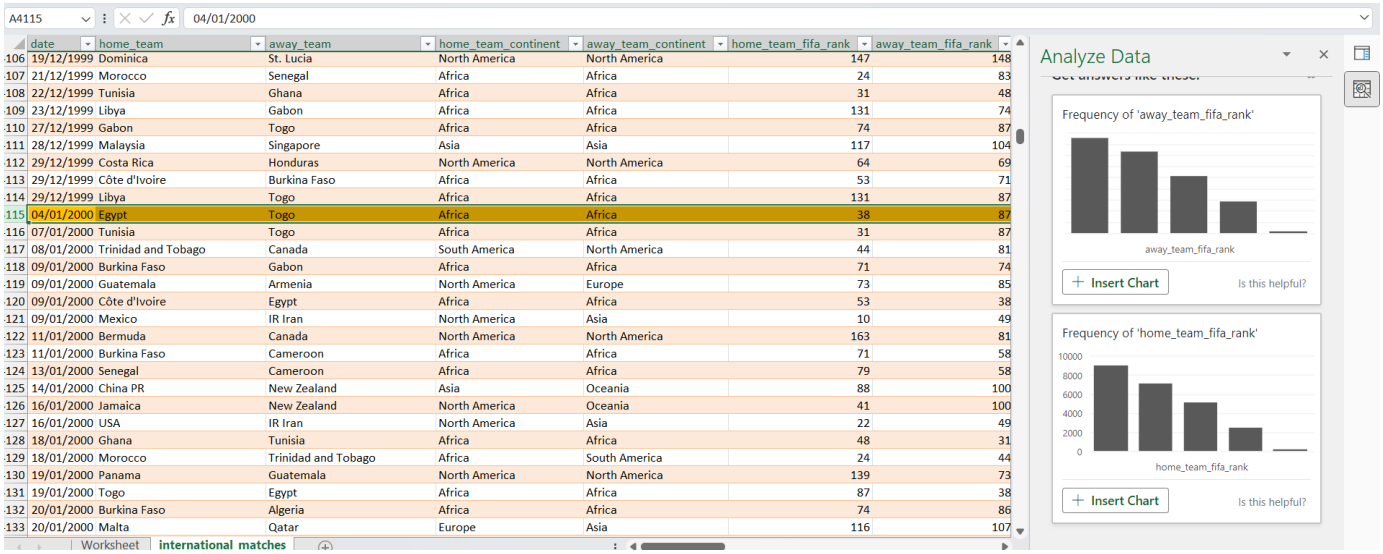
Use excel to fix and choose a range for the desired prediction:

We decided to choose the range from the year 2000 to 2022 to see the prediction base on the last 22 years' data. Identify future trends based on historical data. The reason behind this decision was between 1993 to 2000:

- Globalization hasn’t affected many countries around the globe at the time, football constitutes one of the most dynamic, sociologically illuminating

domains of globalization. Ref: [The globalization of football: a study in the glocalization of the 'serious life']

- Secondly, the world cup started in 1930, and the data was recorded back in 1993. So 2000 is simply a number that our group decided to work on, and the result turns out doesn't affect too much.



A4115 | 04/01/2000

	date	home_team	away_team	home_team_continent	away_team_continent	home_team_fifa_rank	away_team_fifa_rank
106	19/12/1999	Dominica	St. Lucia	North America	North America	147	148
107	21/12/1999	Morocco	Senegal	Africa	Africa	24	83
108	22/12/1999	Tunisia	Ghana	Africa	Africa	31	48
109	23/12/1999	Libya	Gabon	Africa	Africa	131	74
110	27/12/1999	Gabon	Togo	Africa	Africa	74	87
111	28/12/1999	Malaysia	Singapore	Asia	Asia	117	104
112	29/12/1999	Costa Rica	Honduras	North America	North America	64	69
113	29/12/1999	Côte d'Ivoire	Burkina Faso	Africa	Africa	53	71
114	29/12/1999	Libya	Togo	Africa	Africa	131	87
115	04/01/2000	Egypt	Togo	Africa	Africa	38	87
116	07/01/2000	Tunisia	Togo	Africa	Africa	31	87
117	08/01/2000	Trinidad and Tobago	Canada	South America	North America	44	81
118	09/01/2000	Burkina Faso	Gabon	Africa	Africa	71	74
119	09/01/2000	Guatemala	Armenia	North America	Europe	73	85
120	09/01/2000	Côte d'Ivoire	Egypt	Africa	Africa	53	38
121	09/01/2000	Mexico	IR Iran	North America	Asia	10	49
122	11/01/2000	Bermuda	Canada	North America	North America	163	81
123	11/01/2000	Burkina Faso	Cameroon	Africa	Africa	71	58
124	13/01/2000	Senegal	Cameroon	Africa	Africa	79	58
125	14/01/2000	China PR	New Zealand	Asia	Oceania	88	100
126	16/01/2000	Jamaica	New Zealand	North America	Oceania	41	100
127	16/01/2000	USA	IR Iran	North America	Asia	22	49
128	18/01/2000	Ghana	Tunisia	Africa	Africa	48	31
129	18/01/2000	Morocco	Trinidad and Tobago	Africa	South America	24	44
130	19/01/2000	Panama	Guatemala	North America	North America	139	73
131	19/01/2000	Togo	Egypt	Africa	Africa	87	38
132	20/01/2000	Burkina Faso	Algeria	Africa	Africa	74	86
133	20/01/2000	Malta	Qatar	Europe	Asia	116	107

Worksheet | international matches

Transform back to file CSV to use pandas and import into Jupiter notebook:

international_matches_processed.csv

- In colab import through google drive to easily share and work online

```
[3] from google.colab import drive
    drive.mount('/content/drive')
```

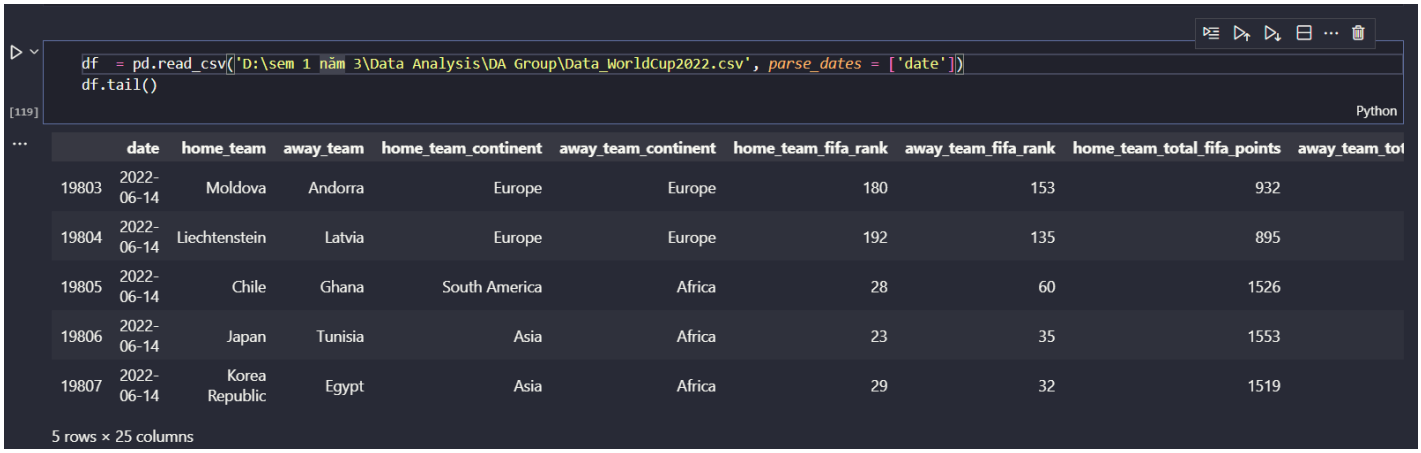
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df = pd.read_csv('/content/drive/MyDrive/DA group /Data_WorldCup2022.csv', parse_dates = ['date'])
df.tail()
```

	date	home_team	away_team	home_team_continent	away_team_continent	home_team_fifa_rank	away_team_fifa_rank	home_team_total_fifa_points	away_team_total_fifa_points	home_team_score	...	shoot_
19803	2022-06-14	Moldova	Andorra	Europe	Europe	180	153	932	1040	2	...	
19804	2022-06-14	Liechtenstein	Latvia	Europe	Europe	192	135	895	1105	0	...	
19805	2022-06-14	Chile	Ghana	South America	Africa	28	60	1526	1387	0	...	
19806	2022-06-14	Japan	Tunisia	Asia	Africa	23	35	1553	1499	0	...	
19807	2022-06-14	Korea Republic	Egypt	Asia	Africa	29	32	1519	1500	4	...	

5 rows × 25 columns

- In the local Jupiter notebook



```
df  = pd.read_csv('D:\sem 1 năm 3\Data Analysis\DA Group\Data_WorldCup2022.csv', parse_dates = ['date'])
df.tail()
```

[119] Python

	date	home_team	away_team	home_team_continent	away_team_continent	home_team_fifa_rank	away_team_fifa_rank	home_team_total_fifa_points	away_team_tot
19803	2022-06-14	Moldova	Andorra	Europe	Europe	180	153	932	
19804	2022-06-14	Liechtenstein	Latvia	Europe	Europe	192	135	895	
19805	2022-06-14	Chile	Ghana	South America	Africa	28	60	1526	
19806	2022-06-14	Japan	Tunisia	Asia	Africa	23	35	1553	
19807	2022-06-14	Korea Republic	Egypt	Asia	Africa	29	32	1519	

5 rows × 25 columns

---

4. Tool Used in the Prediction

1. Excel 

To process from raw data to data cleaning

2. Google Colaboratory Notebook/Jupyter Notebook(ipynb file)

Programing Language: Python

Use the popular libraries in python to do the following processes;
-Import, clean, tidy, and visualize data
-Build model: Decision tree, Random forest, and draw conclusions for Worldcup prediction

5. Code process

5.1 Data

Import the library and read the data

```
# Importing the standard libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Library for working with dates
import datetime as dt

# To display the line plot
%matplotlib inline

# Identify dataset trend
from scipy import stats

# Class is used to select no more than N intervals at nice locations
from matplotlib.ticker import MaxNLocator

# Importing the libraries to train the model
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.linear_model import LogisticRegression

# Importing the metrics libraries
from sklearn.metrics import  accuracy_score
from sklearn.metrics import classification_report
```

Analyze to find the top rank and mean top rank for 3 roles in football

1. Top 10 teams based on FIFA rank

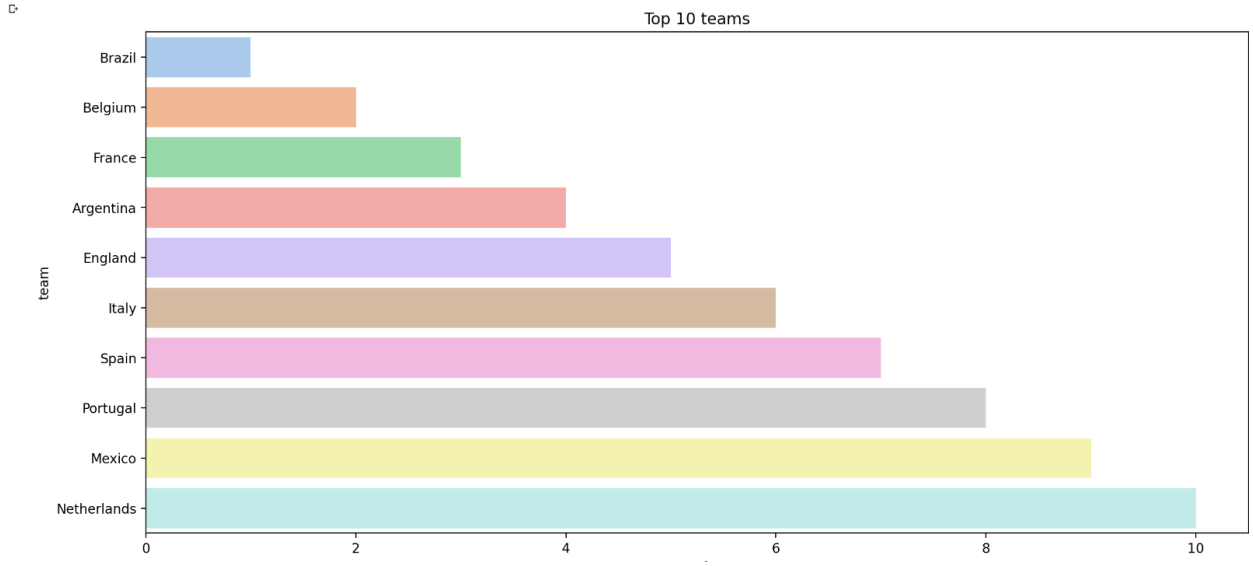
Indexing `home_team_fifa_rank', 'away_team_fifa_rank,` and into `fifa_rank`. Subset away_team and home_team with home_team_fifa_rank and away_team_fifa_rank and change back to team and fifa_rank to see the top teams according to their rank. Finally, merge 2 data frames together to get the `fifa_rank` data frame to visualize the bar plot below

```
fifa_rank = df[['date','home_team','away_team','home_team_fifa_rank', 'away_team_fifa_rank']]
home = fifa_rank[['date','home_team','home_team_fifa_rank']].rename(columns={"home_team":"team","home_team_fifa_rank":"rank"})
away = fifa_rank[['date','away_team','away_team_fifa_rank']].rename(columns={"away_team":"team","away_team_fifa_rank":"rank"})
fifa_rank = pd.concat([home,away])
```

```
[8] #Select each country latest match
    fifa_rank = fifa_rank.sort_values(['team','date'],ascending=[True,False])
    fifa_rank_top10 = fifa_rank.groupby('team').first().sort_values('rank',ascending=True)[0:10].reset_index()
    fifa_rank_top10
```

	team	date	rank
0	Brazil	2022-06-06	1
1	Belgium	2022-06-14	2
2	France	2022-06-13	3
3	Argentina	2022-06-05	4
4	England	2022-06-14	5
5	Italy	2022-06-14	6
6	Spain	2022-06-12	7
7	Portugal	2022-06-12	8
8	Mexico	2022-06-14	9
9	Netherlands	2022-06-14	10

```
plt.figure(figsize=(15,7),dpi=200)
sns.barplot(data=fifa_rank_top10,y='team',x='rank',palette='pastel',dodge=False)
plt.title("Top 10 teams");
```



2. Top 10 teams with the highest offense point

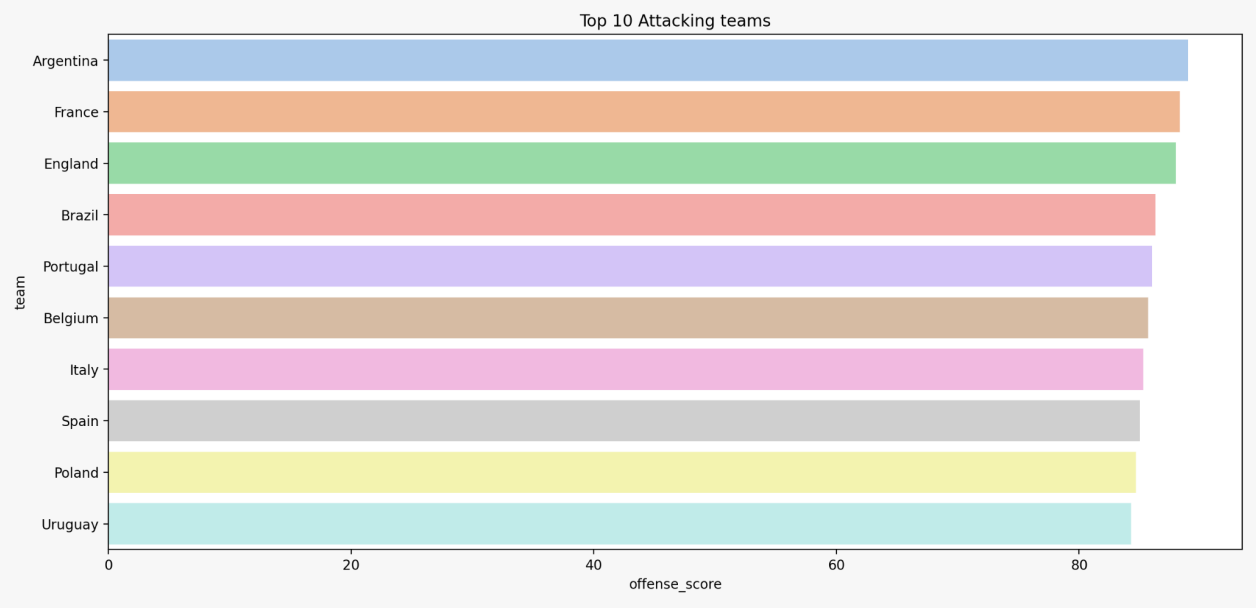
The reason behind this comparison with different rankings is:
In sports, especially football, each member has their own role, and is important to compare strengths among other teams.

Therefore before starting a prediction, we diagnose the data trends of which team was winning the highest scores in the tournament, the highest saves, highest midfield score using 6 attributes: `'home_team_mean_defense_score', 'home_team_mean_offense_score', 'home_team_mean_midfield_score', 'away_team_mean_defense_score', 'away_team_mean_offense_score', 'away_team_mean_midfield_score'`

```
print(df[['home_team_mean_defense_score','home_team_mean_offense_score','home_team_mean_midfield_score']].tail(3))
print(df[['away_team_mean_defense_score','away_team_mean_offense_score','away_team_mean_midfield_score']].tail(3))
```

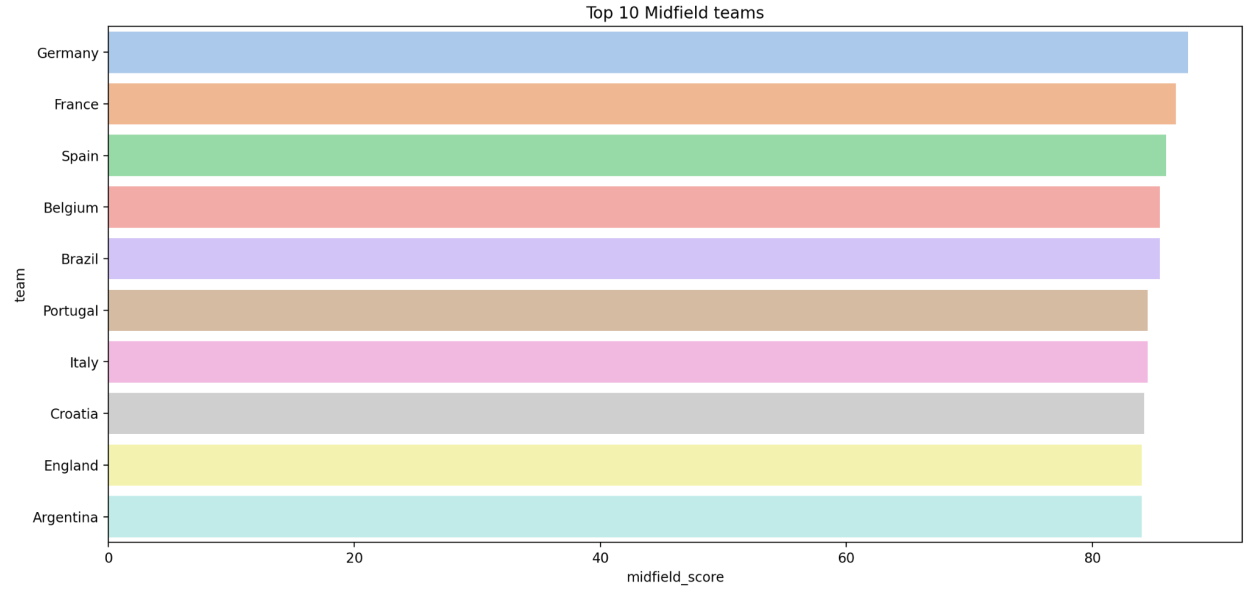
```
       home_team_mean_defense_score  home_team_mean_offense_score  \
19805                          76.0                          77.0
19806                          75.0                          75.0
19807                          73.0                          80.0

       home_team_mean_midfield_score
19805                           78.0
19806                           78.0
19807                           74.0
       away_team_mean_defense_score  away_team_mean_offense_score  \
19805                          76.0                          76.0
19806                          71.0                          72.0
19807                          70.0                          79.0

       away_team_mean_midfield_score
19805                           78.0
19806                           74.0
19807                           71.0
```

Just like in the previous barplot, we start to subsetting from the df with the attributes we need. Then subset the new data frame into 2 separate data frames with the same attribute but name as home_… and away_….Rename columns **home_team__mean…** and **away_team_mean_…** into ‘team’ and convert back to 3 attributes which are vital to plot the top 10 teams which each attribute above. Finally, concatenate them together to have data that can be compared by using sort_values on the ‘team’ attribute, and then group by on ‘team’ and sort_values on ‘offense_score’ this time to plot the top 10 attacking teams `fifa_offense_top10`



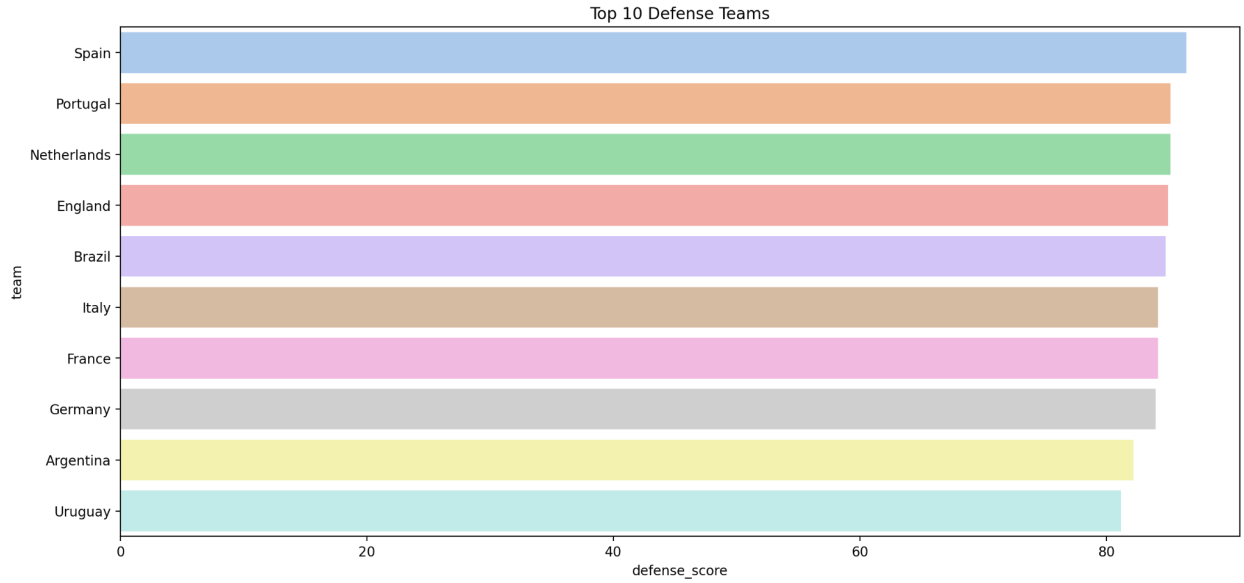
3. Top 10 teams with the highest midfield points

Do the same steps in the analysis, subset, and plotting df except change the columns `home_team_mean_offense_score', 'away_team_mean_offense_score'`
For: `'home_team_mean_midfield_score','away_team_mean_midfield_score'`



4. Top 10 teams with the highest offense point

The same for the highest offense points team



5. Winning Change of top 10 teams

Brazil in top 4 `offense_score` and top 5 in `defense_score,midfield_score`
Yet Brazil has the record for the most World Cup titles in soccer history with five, taking home the trophy in 1958, 1962, 1970, and 1994 and, most recently, in 2002. Maybe because we exclude the data from before the year 2000. Which is when Brazil's team was in its prime. Plus the data only got recorded in 1993, so the Brazil team was not as strong as they were supposed to be. This should be accountable when we make our prediction of which team/country would win the 2022 World Cup title

Dealing with missing values

Analyze and fill nas

```
[19] df.isnull().sum()
```

0 giây

```
date                                0
home_team                           0
away_team                           0
home_team_continent                 0
away_team_continent                 0
home_team_fifa_rank                 0
away_team_fifa_rank                 0
home_team_total_fifa_points         0
away_team_total_fifa_points         0
home_team_score                     0
away_team_score                     0
tournament                          0
city                                0
country                             0
neutral_location                    0
shoot_out                           0
home_team_result                    0
home_team_goalkeeper_score      11429
away_team_goalkeeper_score      11713
home_team_mean_defense_score    12021
home_team_mean_offense_score    11298
home_team_mean_midfield_score   11646
away_team_mean_defense_score    12244
away_team_mean_offense_score    11496
away_team_mean_midfield_score   11829
dtype: int64
```

As mentioned briefly earlier, all the missing values shown in the picture above were only missing till 2004-09-03. That equates to 5 years of missing values in the range our group chooses to analyze, and the scoring was also inconsistent leading to more than 11 thousand missing values in each of the 'score' attributes.

First, we fill NAN values with mean in Pandas using the fillna() method from the pandas' library. Using groupby to grouby `home_team` with `ome_team_goalkeeper_score` and apply df.transform to fill NAN for the missing

value. After fina() for all the columns with each missing more than 11k values. This is the tidy version of the data frame

```
df.isnull().sum()
```

```
date                            0
home_team                       0
away_team                       0
home_team_continent             0
away_team_continent             0
home_team_fifa_rank             0
away_team_fifa_rank             0
home_team_total_fifa_points     0
away_team_total_fifa_points     0
home_team_score                 0
away_team_score                 0
tournament                      0
city                            0
country                         0
neutral_location                0
shoot_out                       0
home_team_result                0
home_team_goalkeeper_score   4991
away_team_goalkeeper_score   5511
home_team_mean_defense_score 6940
home_team_mean_offense_score 6116
home_team_mean_midfield_score 6036
away_team_mean_defense_score 7217
away_team_mean_offense_score 6260
away_team_mean_midfield_score 6159
dtype: int64
```

dex	home_team_goalkeeper_score	away_team_goalkeeper_score	home_team_mean_defense_score	home_team_mean_offense_score	away_team_mean_defense_score	away_team_mean_offense_score	away_team_mean_midfield_score
0	70.0	69.0	70.0	74.0	62.0	72.0	68.0
1	65.0	69.0	72.0	71.0	62.0	72.0	68.0
2	66.0	72.0	65.0	71.0	69.0	71.0	73.0
3	71.0	67.0	67.0	70.0	68.0	72.0	73.0
4	67.0	63.0	50.0	67.0	50.0	70.0	73.0
5	70.0	70.0	77.0	81.0	70.0	74.0	72.0
6	79.0	69.0	77.0	79.0	69.0	71.0	70.0
7	57.0	72.0	50.0	62.0	69.0	71.0	73.0
8	71.0	78.0	67.0	70.0	78.0	79.0	77.0
9	73.0	78.0	78.0	80.0	78.0	79.0	77.0

We can see that most NaN is now gone and the one that is still missing is the team was not in the FIFA game itself. Their lack of appearance does not mean they were./are less than the team the in the raw data. Simply set all missing values after transforming to 50 before we move on to go to the. …

```
df.fillna(50,inplace=True)
df.isnull().sum()A
```

```
date                           0
home_team                      0
away_team                      0
home_team_continent            0
away_team_continent            0
home_team_fifa_rank            0
away_team_fifa_rank            0
home_team_total_fifa_points    0
away_team_total_fifa_points    0
home_team_score                0
away_team_score                0
tournament                     0
city                           0
country                        0
neutral_location               0
shoot_out                      0
home_team_result               0
home_team_goalkeeper_score     0
away_team_goalkeeper_score     0
home_team_mean_defense_score   0
home_team_mean_offense_score   0
home_team_mean_midfield_score  0
away_team_mean_defense_score   0
away_team_mean_offense_score   0
away_team_mean_midfield_score  0
dtype: int64
```

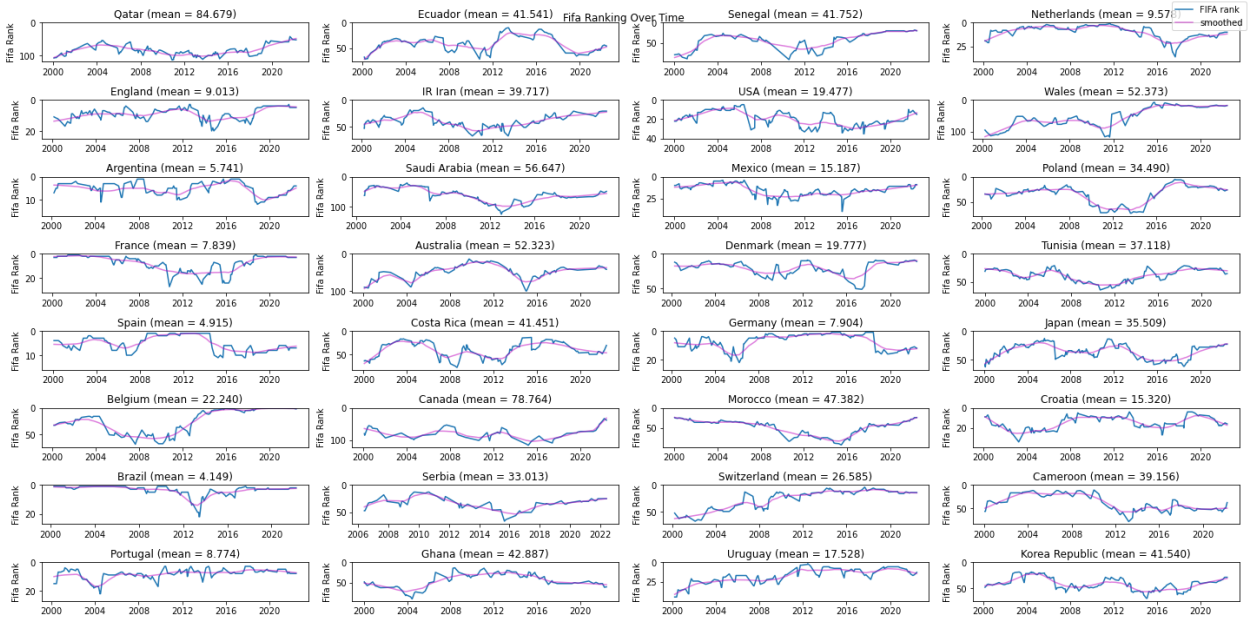
Top 10 teams in Worldcup 22 with individual winning percentage

World Cup teams and group draw in full

- **Group A:** Qatar, Ecuador, Senegal, Netherlands
- **Group B:** England, Iran, USA, Wales
- **Group C:** Argentina, Saudi Arabia, Mexico, Poland
- **Group D:** France, Australia, Denmark, Tunisia
- **Group E:** Spain, Costa Rica, Germany, Japan
- **Group F:** Belgium, Canada, Morocco, Croatia
- **Group G:** Brazil, Serbia, Switzerland, Cameroon
- **Group H:** Portugal, Ghana, Uruguay, South Korea

The picture above shows 32 teams in the FIFA World Cup 2022. Subset the data with 32 qualified teams. We define a function to plot a bar chart showing the rank of a country over time, and run it through the list containing 32 countries in the picture above. We can see a trend that a lot of teams that are attending the world cup have gone up in rank(Whereas a lower rank indicates a better team).

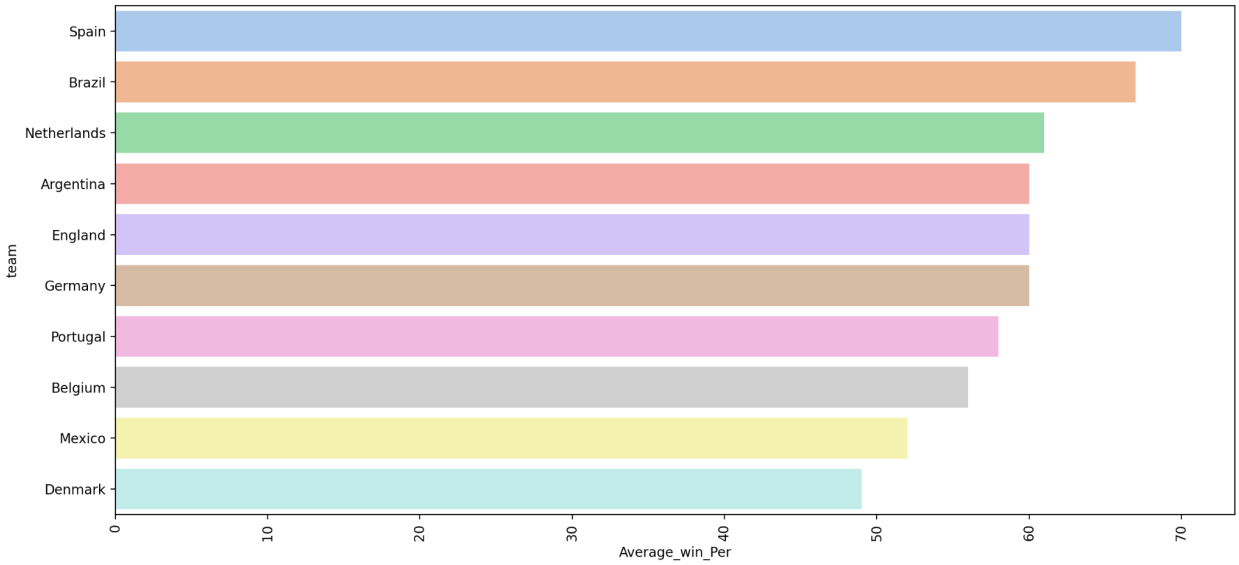
- With 7 teams ranked lower than 10: Brazil, Portugal, Germany, Spain, France, Argentina, and England
- All the 21 rankings mean over time are all in the range 55(Saudi Arabia approximately 56.647,). The outliers clearly are Canada and Qatar rank up 78 and 84 in the FIFA ranking mean since 2000 till now.



We repeat the steps from earlier: subset into 2 different data frames start home_… and away_…Then rename 2 the columns back to team and rank. Finally, concatenate and sort values by team, and use group by method to get the `rank_top10`
Data frame with rank ascending order.

This time instead of plotting existing values in the data. We define a function to see the **average win rate** based on the nominal attribute `home_team_result`

```
[36] def home_percentage(team):
         score = len(df[(df['home_team'] == team) & (df['home_team_result'] == "Win")]) / len(df[df['home_team'] == team]) * 100
         return round(score)
     def away_percentage(team):
         score = len(df[(df['away_team'] == team) & (df['home_team_result'] == "Lose")]) / len(df[df['away_team'] == team]) * 100
         return round(score)
```



The barplot is the plot with Average_win_Per to see the top 10 teams, Using the function our group defines that return the score. Using the function on home_team to create 2 new columns `Home_win_Per` and `Away_win_Per.` Take the average of each team whether they are home or away teams and call it `Average_win_Per` column.

Then plot the new data frame of 32 qualified teams to see the top team with the highest win rate. This data should come close to who will be in the top 10 of the 2022 World Cup since their performance was good from 2000.

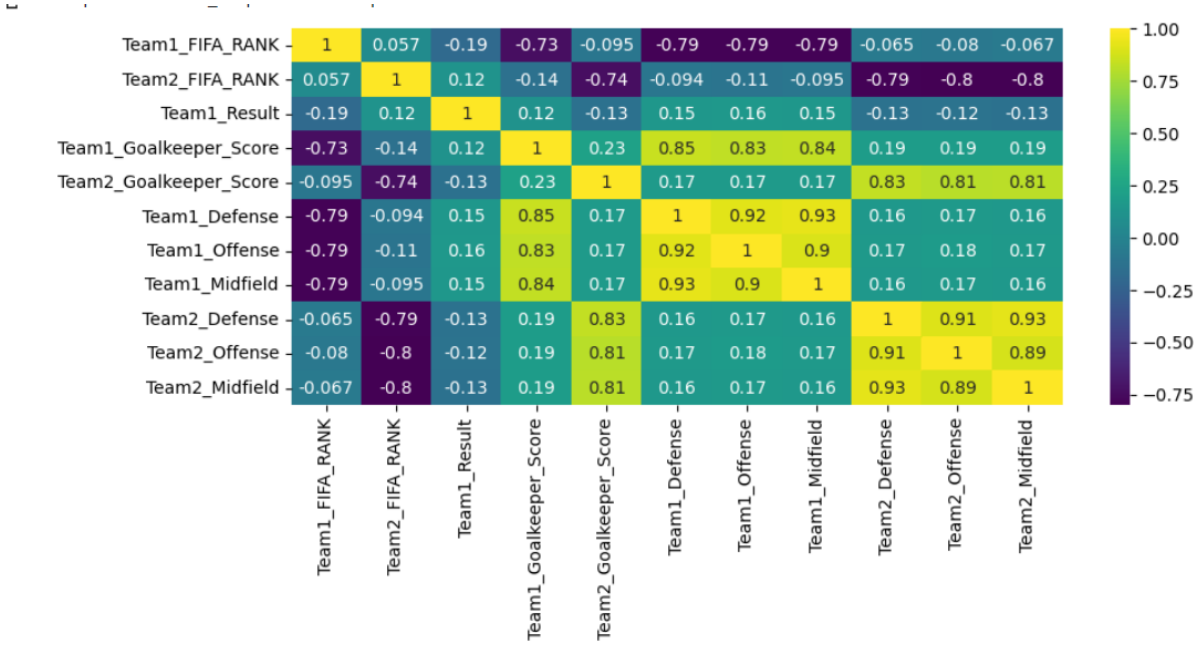
Correlation Analysis

After seeing the rank distribution of 32 teams that qualified for the 2022 WorldCup tournament

```
print(final_df.corr())
```

```
Output exceeds the size limit. Open the full output data in a text editor
                      Team1_FIFA_RANK  Team2_FIFA_RANK  Team1_Result  \
Team1_FIFA_RANK              1.000000         0.045458     -0.192660
Team2_FIFA_RANK              0.045458         1.000000      0.122823
Team1_Result                -0.192660         0.122823      1.000000
Team1_Goalkeeper_Score      -0.735506        -0.123008      0.123016
Team2_Goalkeeper_Score      -0.082718        -0.736754     -0.130074
Team1_Defense               -0.793681        -0.081750      0.157862
Team1_Offense               -0.793355        -0.094779      0.166237
Team1_Midfield              -0.793984        -0.085577      0.157549
Team2_Defense               -0.055427        -0.794802     -0.127226
Team2_Offense               -0.067151        -0.800565     -0.121054
Team2_Midfield              -0.057569        -0.802470     -0.132665

                      Team1_Goalkeeper_Score  Team2_Goalkeeper_Score  \
Team1_FIFA_RANK                   -0.735506               -0.082718
Team2_FIFA_RANK                   -0.123008               -0.736754
Team1_Result                       0.123016               -0.130074
Team1_Goalkeeper_Score             1.000000                0.217527
Team2_Goalkeeper_Score             0.217527                1.000000
Team1_Defense                      0.850622                0.161113
Team1_Offense                      0.829850                0.162335
Team1_Midfield                     0.836981                0.160769
Team2_Defense                      0.179132                0.832317
Team2_Offense                      0.175584                0.807502
Team2_Midfield                     0.176955                0.815808
...
Team1_Midfield              0.147076        0.155312        0.147617
Team2_Defense               1.000000        0.907733        0.924558
Team2_Offense               0.907733        1.000000        0.887597
Team2_Midfield              0.924558        0.887597        1.000000
```

A better way to see that is graphically using a heat map as shown below.



The diagonal is equal to 1 since it represents the correlation of the same attribute. Team1_Goalkeepe_score seems to be highly correlated with Team2 Defense, Offense, Midfield, and vice visa.
Meaning the 4 score types of the team is an almost direct relation to each other, but when comparing the scores of team1 to team2 it is a different story. While the rank order of the team literally has the opposite correlation, which is expected because the higher the score of the team the lower the rank.

```
#Holdout another DF for pipeline
pipe_DF = final_df
#Create dummies for categorical columns
final_df = pd.get_dummies(final_df)
```

```
pd.get_dummies(final_df)
```

	Team1_FIFA_RANK	Team2_FIFA_RANK	Team1_Result	Team1_Goalkeeper_Score	Team2_Goalkeeper_Score	Team1_Defense	Team1_Offense	Team1_Midfield	Team2_Defense	Team2_Offense
1	31	87	1	65.0	69.0	72.0	71.0	73.0	62.0	72.0
2	44	81	2	66.0	72.0	65.0	71.0	67.0	69.0	71.0
6	10	49	1	79.0	69.0	77.0	79.0	78.0	69.0	71.0
7	163	81	0	57.0	72.0	50.0	62.0	50.0	69.0	71.0
8	71	58	2	71.0	78.0	67.0	70.0	71.0	78.0	79.0
...	...	...	...	...	...	...	...	...	...	...
19795	10	18	1	81.0	74.0	85.0	83.0	84.0	75.0	73.0
19796	26	2	0	87.0	89.0	75.0	85.0	76.0	81.0	86.0
19805	28	60	0	79.0	74.0	76.0	77.0	78.0	76.0	76.0
19806	23	35	0	73.0	64.0	75.0	75.0	78.0	71.0	72.0
19807	29	32	1	75.0	70.0	73.0	80.0	74.0	70.0	79.0

7147 rows × 400 columns

We used dummies to convert categorical variables to numeric variables that are used to represent categorical data. Converting win to 1, draw to 2, and lose to 0 in team1_results.

```
#split X and y and train test split (For League Matches)
X = final_df.drop('Team1_Result',axis=1)
y = final_df['Team1_Result']
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)
#Validtion and test set
X_hold_test, X_test, y_hold_test, y_test = train_test_split(X_val, y_val, test_size=0.5, random_state=42)
```

```
[ ] #Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_hold_test = scaler.transform(X_hold_test)
```

```
from sklearn.metrics import classification_report,ConfusionMatrixDisplay
def metrics_display(model):
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test,y_pred))
    ConfusionMatrixDisplay.from_predictions(y_test,y_pred,cmap=plt.cm.PuBu)
```

We look for a pattern that accurately and efficiently describes all the data points while training a machine-learning model. There are two frequent mistakes made in the process. These are overfitting and underfitting.

Used to optimize model parameters is the validation set.
Test Set: to obtain a fair assessment of the performance of the final model.

Create metrics constructor

5.2 Machine Learning

Decision tree
Random forest
XGBoost
Pipeline

6) Requirement project results

- Data points for analysis

Date,
Home/Away teams,
Home/Away team total FIFA points,
Home/Away team result,
Home/Away win percentage,
Average win percentage,
Home/Away team goalkeeper score,
Mean of home/away team defense score,
Mean of home/away team offense score,
Mean of home/away team midfield score,
Ranking

- Errors/problems of findings lead to impact on accuracy

  Having missing values in data (home/away team goalkeeper score, home/away team defense score, home/away team midfield score, home/away team offense score).

  The results from many years ago, the prediction will be more accurate when we use the results from 9 or 10 years up to now, so we use the data from 2000 up to now.

- Anomalies or unexpected behavior

  In the top 10 rankings, Italy is in 6th. However, they were out in the play-off. So, the ranking is not a prerequisite for strong teams to

advance to the next round. It also depends on the form and performance of each player in each team.

Figure out the ranking of the top 10 football team

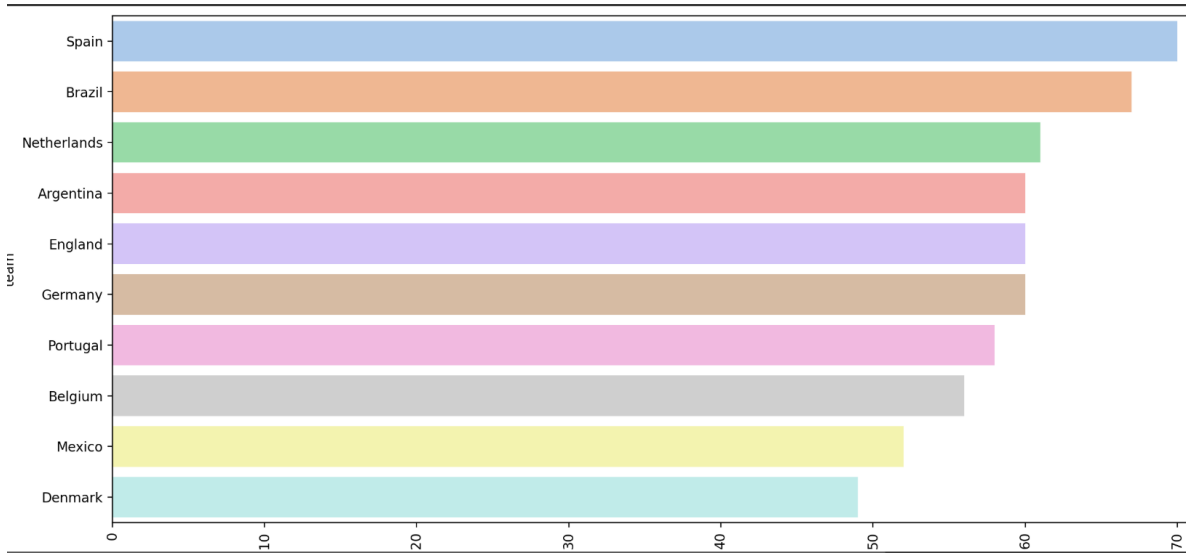
```
#Select each country latest match
fifa_rank = fifa_rank.sort_values(['team','date'],ascending=[True,False])
fifa_rank_top10 = fifa_rank.groupby('team').first().sort_values('rank',ascending=True)[0:10].reset_index()
fifa_rank_top10
```

	team	date	rank
0	Brazil	2022-06-06	1
1	Belgium	2022-06-14	2
2	France	2022-06-13	3
3	Argentina	2022-06-05	4
4	England	2022-06-14	5
5	Italy	2022-06-14	6
6	Spain	2022-06-12	7
7	Portugal	2022-06-12	8
8	Mexico	2022-06-14	9
9	Netherlands	2022-06-14	10

- The key takeaway is based on the data visualization chart

Predicted value and chart

	team	date	rank	Home_win_Per	Away_win_Per	Average_win_Per
6	Spain	2022-06-12	7	80	61	70.0
0	Brazil	2022-06-06	1	74	60	67.0
9	Netherlands	2022-06-14	10	65	57	61.0
3	Argentina	2022-06-05	4	69	52	60.0
4	England	2022-06-14	5	66	53	60.0
11	Germany	2022-06-14	12	62	58	60.0
7	Portugal	2022-06-12	8	65	51	58.0
1	Belgium	2022-06-14	2	63	49	56.0
8	Mexico	2022-06-14	9	62	43	52.0
10	Denmark	2022-06-13	11	53	45	49.0



The actual results

According to the data information, strong teams like Mexico, Belgium, Denmark, and Germany were out of the league instead the USA, Senegal, Australia, Japan, and Morocco joined in the next round. So, the prediction of the World Cup should be checked on the performance of each football player and the history of head-to-head. Moreover, the element of surprise is a point that we cannot predict so the prediction results will have errors. What we need to do is to minimize that error as little as possible.

- The types of analysis: prediction

Precision: The ratio of True Positives to All Positives is known as precision.

$$Precision = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)}$$

Recall: The recall serves as a gauge of how well our algorithm detects True Positives

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

Accuracy: The proportion of all right forecasts to all predictions is known as accuracy.

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{(True Positive + False Positive + True Negative + False Negative)}}$$

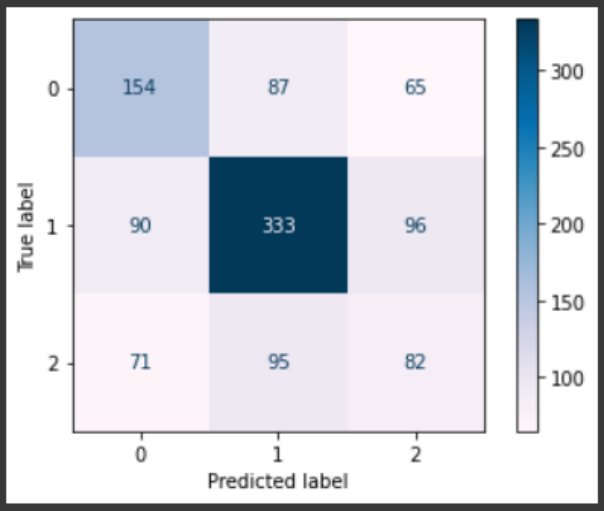
F1-score: The Harmonic mean of the Precision and Recall

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Model

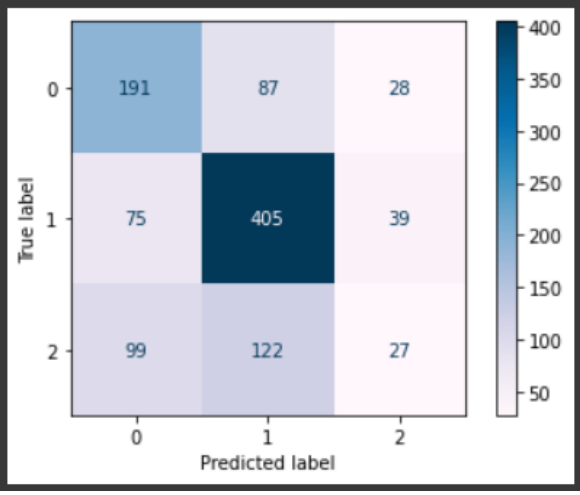
```python
#Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
metrics_display(DecisionTreeClassifier())
```

```
              precision    recall  f1-score   support

           0       0.49      0.50      0.50       306
           1       0.65      0.64      0.64       519
           2       0.34      0.33      0.33       248

    accuracy                           0.53      1073
   macro avg       0.49      0.49      0.49      1073
weighted avg       0.53      0.53      0.53      1073
```



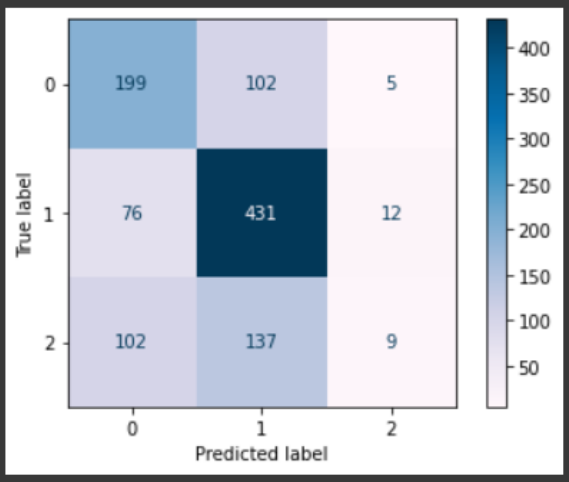
```python
#Random Forest
from sklearn.ensemble import RandomForestClassifier
metrics_display(RandomForestClassifier())
```

	precision	recall	f1-score	support
0	0.52	0.62	0.57	306
1	0.66	0.78	0.71	519
2	0.29	0.11	0.16	248
accuracy			0.58	1073
macro avg	0.49	0.50	0.48	1073
weighted avg	0.53	0.58	0.54	1073



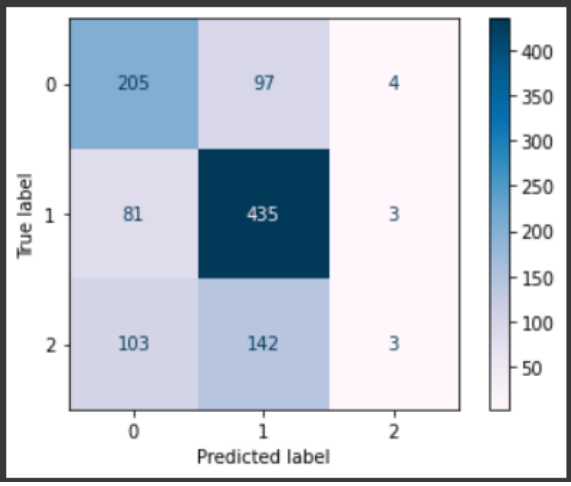
```
#Ada Boost Classifier
from sklearn.ensemble import AdaBoostClassifier
metrics_display(AdaBoostClassifier())
```

	precision	recall	f1-score	support
0	0.53	0.65	0.58	306
1	0.64	0.83	0.72	519
2	0.35	0.04	0.07	248
accuracy			0.60	1073
macro avg	0.51	0.51	0.46	1073
weighted avg	0.54	0.60	0.53	1073



```
[ ]  #XGB Boost
     from xgboost import XGBClassifier
     metrics_display(XGBClassifier(use_label_encoder=False))
```

	precision	recall	f1-score	support
0	0.53	0.67	0.59	306
1	0.65	0.84	0.73	519
2	0.30	0.01	0.02	248
accuracy			0.60	1073
macro avg	0.49	0.51	0.45	1073
weighted avg	0.53	0.60	0.53	1073



Precision and recall are both equally crucial: For example, in our model, if the data indicates that strong teams would advance to the next round, but the actual outcomes are inaccurate, it shows that the teams are influenced by factors other than the data obtained. If other variables influence performance, advantage, etc., we will strive for not only rapid recovery but also excellent precision.

In such circumstances, we employ a technique known as F1-score. This is easier to work with since, instead of balancing accuracy and recall, we can simply strive for a high F1 score, which indicates high Precision and high Recall numbers.

The accuracy of the random forest is higher than the decision tree (58% > 48%). , we use the XGBoost algorithm to attempt to

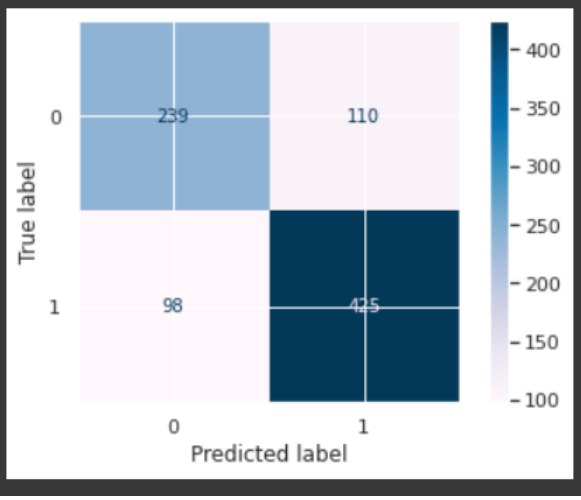
accurately predict a target variable by combining the estimates of a set of simpler, weaker models. The accuracy is boosted to 61%.

We will predict the knockout stage!

```
[240] knock_df = pipe_DF[pipe_DF['Team1_Result'] != 2]

[241] pipe_knock_df = knock_df
      knock_df = pd.get_dummies(knock_df)
      X = knock_df.drop('Team1_Result',axis=1)
      y = knock_df['Team1_Result']

      X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)
      X_hold_test, X_test, y_hold_test, y_test = train_test_split(X_val, y_val, test_size=0.5, random_state=42)

[242] metrics_display(RandomForestClassifier())
```

```
metrics_display(RandomForestClassifier())
```

	precision	recall	f1-score	support
0	0.71	0.68	0.70	349
1	0.79	0.81	0.80	523
accuracy			0.76	872
macro avg	0.75	0.75	0.75	872
weighted avg	0.76	0.76	0.76	872

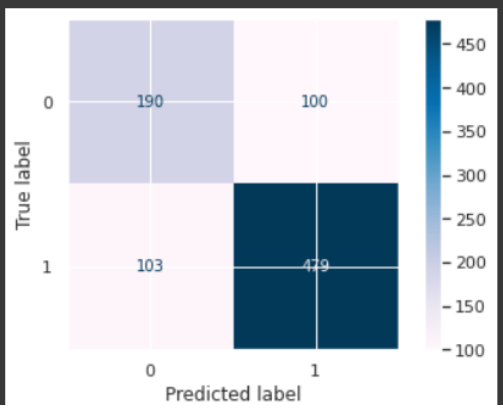


The accuracy of the knockout stage is higher than in previous models (58% - 76%) because we removed the draw results.

The random forest is giving good accuracy, so testing with a holdout test and finalizing the model.

```
model = RandomForestClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_hold_test)
print(classification_report(y_hold_test,y_pred))
ConfusionMatrixDisplay.from_predictions(y_hold_test,y_pred,cmap=plt.cm.PuBu);
```

```
              precision    recall  f1-score   support

           0       0.65      0.66      0.65       290
           1       0.83      0.82      0.83       582

    accuracy                           0.77       872
   macro avg       0.74      0.74      0.74       872
weighted avg       0.77      0.77      0.77       872
```



XGB model is finalized, so creating a pipeline for it.

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
column_trans = make_column_transformer(
    (OneHotEncoder(),['Team1', 'Team2']),remainder='passthrough')

pipe_X = pipe_DF.drop('Team1_Result',axis=1)
pipe_y = pipe_DF['Team1_Result']

from sklearn.pipeline import make_pipeline
pipe_League = make_pipeline(column_trans,StandardScaler(with_mean=False),XGBClassifier(use_label_encoder=False))
pipe_League.fit(pipe_X,pipe_y)
```

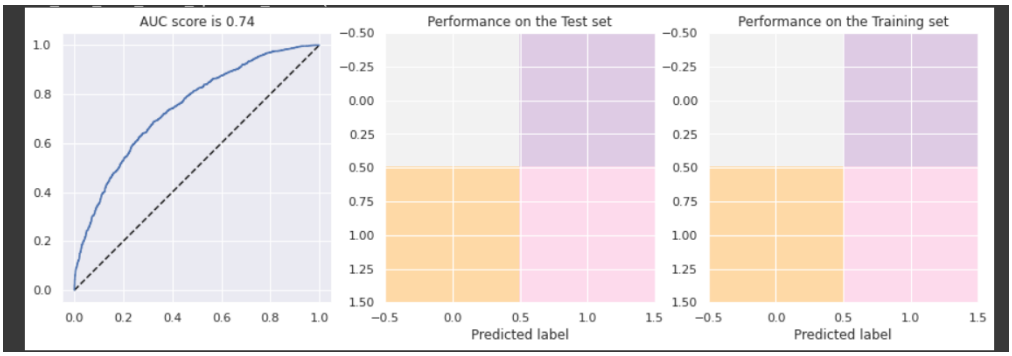
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(),
                                                  ['Team1', 'Team2'])])),
                ('standardscaler', StandardScaler(with_mean=False)),
                ('xgbclassifier',
                 XGBClassifier(objective='multi:softprob',
                               use_label_encoder=False))])
```

```
pipe_X = pipe_knock_df.drop('Team1_Result',axis=1)
pipe_y = pipe_knock_df['Team1_Result']
pipe_knock = make_pipeline(column_trans,StandardScaler(with_mean=False),RandomForestClassifier())
pipe_knock.fit(pipe_X,pipe_y)
```

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(),
                                                  ['Team1', 'Team2'])])),
                ('standardscaler', StandardScaler(with_mean=False)),
                ('randomforestclassifier', RandomForestClassifier())])
```

7)Simulation

```
#X, y = df.loc[:,['average_rank', 'rank_difference', 'point_difference','is_stake']], df['is_won']
X, y = df.loc[:,['average_rank', 'rank_difference', 'point_difference']], df['is_won']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

logreg = linear_model.LogisticRegression(C=1e-5)
features = PolynomialFeatures(degree=2)
model = Pipeline([
    ('polynomial_features', features),
    ('logistic_regression', logreg)
])
model = model.fit(X_train, y_train)

# figures
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
plt.figure(figsize=(15,5))
ax = plt.subplot(1,3,1)
ax.plot([0, 1], [0, 1], 'k--')
ax.plot(fpr, tpr)
ax.set_title('AUC score is {0:0.2}'.format(roc_auc_score(y_test, model.predict_proba(X_test)[:,1])))
ax.set_aspect(1)

ax = plt.subplot(1,3,2)
cm = confusion_matrix(y_test, model.predict(X_test))
ax.imshow(cm, cmap='Pastel1', clim = (0, cm.max()))

ax.set_xlabel('Predicted label')
ax.set_title('Performance on the Test set')

ax = plt.subplot(1,3,3)
cm = confusion_matrix(y_train, model.predict(X_train))
ax.imshow(cm, cmap='Pastel1', clim = (0, cm.max()))
ax.set_xlabel('Predicted label')
ax.set_title('Performance on the Training set')
pass
```



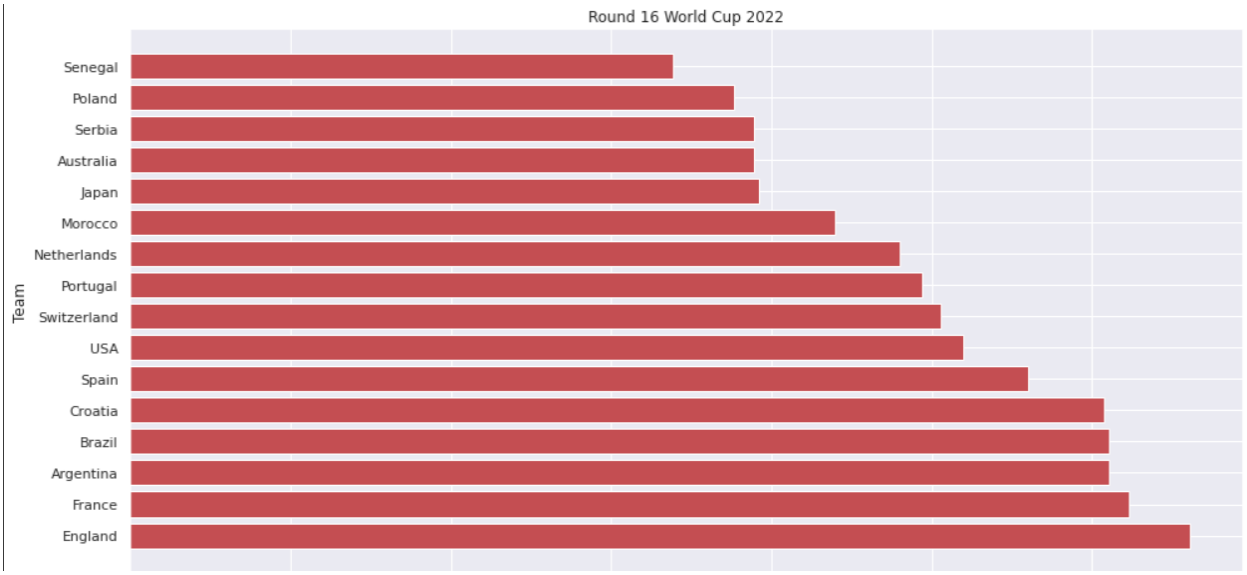
The model of simulation on the vertical is true positive rate (TPR), and on the horizontal is false positive rate (FPR). The AUC score is 0.74, so the model is quite good.

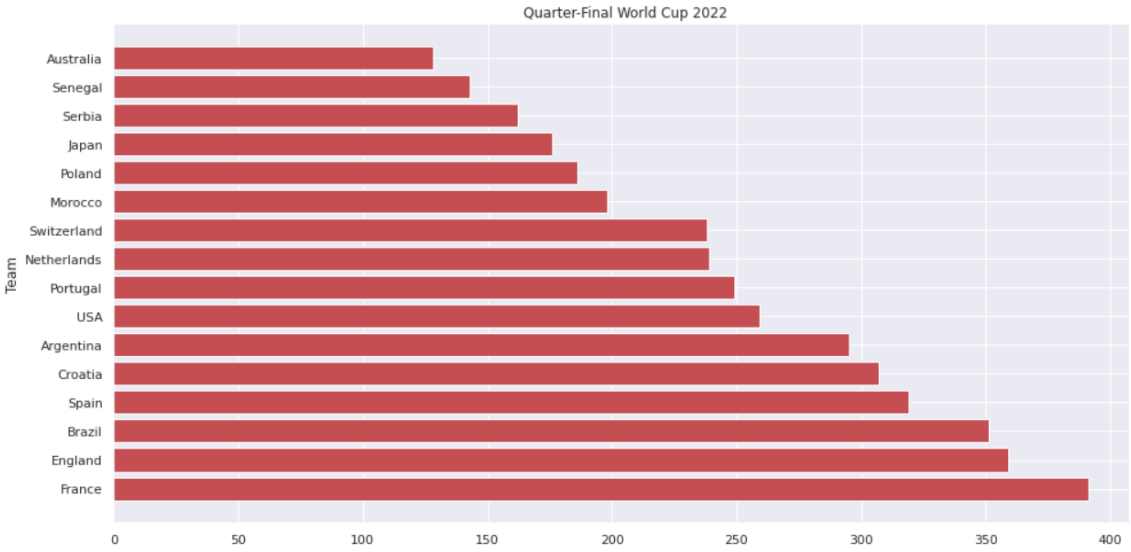
Test

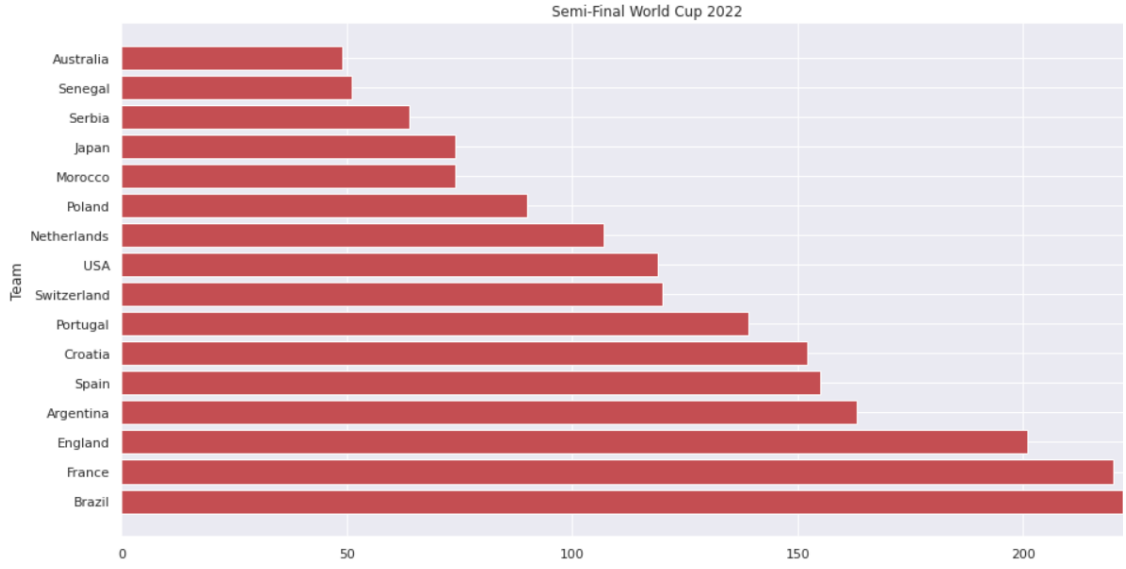
```
[248] world_cup_rankings_home = df[['home_team','home_team_fifa_rank','home_team_total_fifa_points']].loc[df['home_team'].isin(list_2022) & (df['date']>'2021-01-01')] #Get last 2 year
      world_cup_rankings_away = df[['away_team','away_team_fifa_rank','away_team_total_fifa_points']].loc[df['away_team'].isin(list_2022)& (df['date']>'2021-01-01')]
      world_cup_rankings_home = world_cup_rankings_home.set_index(['home_team'])

[249] #The idea is to separete the performance of each Team as Home or Away.
      world_cup_rankings_home = world_cup_rankings_home.groupby('home_team').mean()
      world_cup_rankings_away = world_cup_rankings_away.groupby('away_team').mean()

      world_cup_rankings_away.head(13)
      round(world_cup_rankings_away)
```

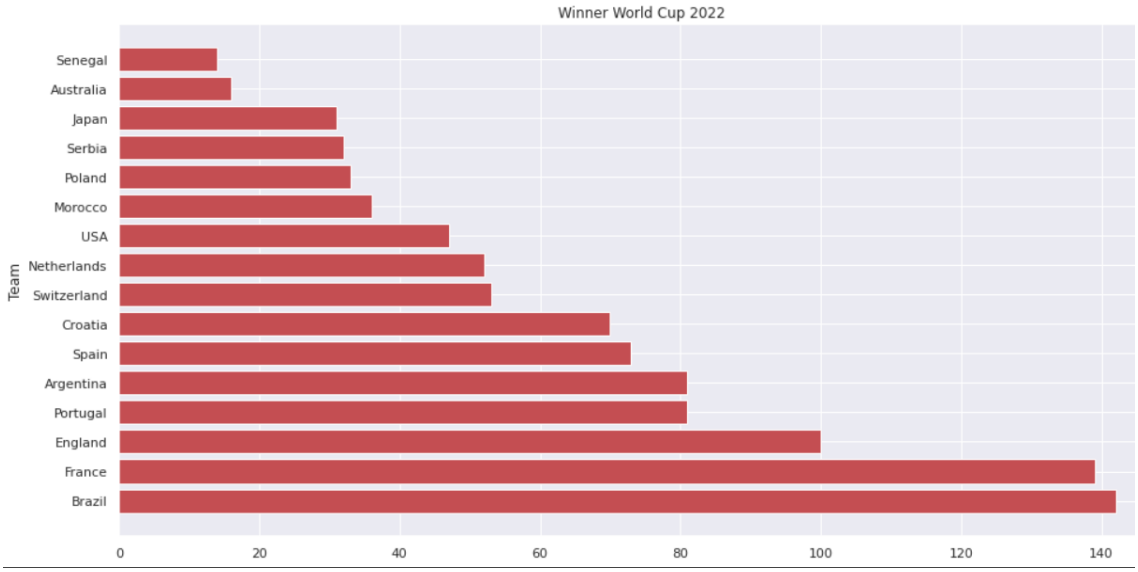
away_team		
Argentina	6.0	1718.0
Australia	36.0	1478.0
Belgium	1.0	1811.0
Brazil	2.0	1810.0
Cameroon	49.0	1432.0

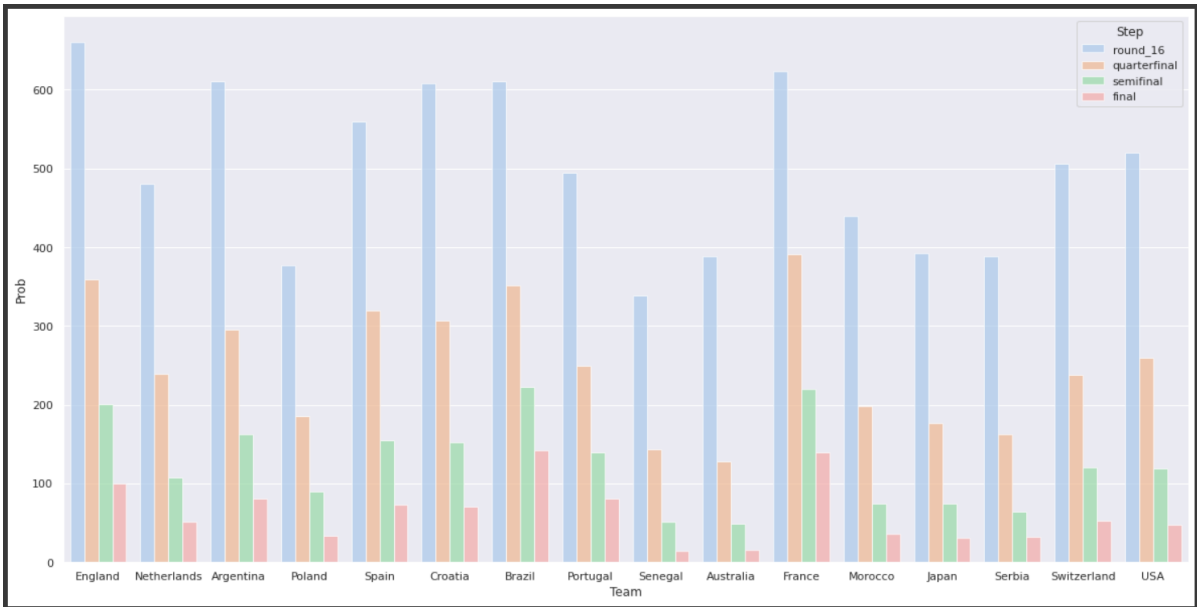
We test the data by splitting into each team home and away. Besides that, we round the FIFA rank points. Finally, we use the barplot to help the viewers observe easily the probability of each team joining in the next round.




Quarter-Final World Cup 2022
Team
Australia
Senegal
Serbia
Japan
Poland
Morocco
Switzerland
Netherlands
Portugal
USA
Argentina
Croatia
Spain
Brazil
England
France
0
50
100
150
200
250
300
350
400


Semi-Final World Cup 2022
Team
Australia
Senegal
Serbia
Japan
Morocco
Poland
Netherlands
USA
Switzerland
Portugal
Croatia
Spain
Argentina
England
France
Brazil
0
50
100
150
200





So, Brazil has a high opportunity to become a champion.
Argentina, Portugal, England, and France are four big challenges to Brazil.

Brazil had five World Cups (the country is known to win the most trophies in history).

WC 2002: Brazil champion
Euro 2006: Italy champion
WC 2010: Spain champion
Euro 2014: German champion
WC 2018: France champion
Euro 2020: Italy champion