FINAL PROJECT PHASE IV

Table of Contents

1. Introduction
   a. Background and Context
   b. Research Question
   c. Data Collection and Cleaning Summary
2. Data Description
3. Preregistration Statement
4. Data Analysis
   a. Data Exploration
   b. Research Question Models - Evaluation of Significance - Interpretation and Conclusions
5. Limitations
6. Source Code
7. Acknowledgements

1. Introduction

a. Background and Context

In this project, we examined the dynamics of the FIFA World Cup, specifically in seeking to understand how the pressure of the tournament's progression impacts the nature of play. By analyzing the dataset of match statistics, we focus on expected goals (xG) to infer whether the transition from group stage to knockout rounds influences the attacking intensity of the teams. The ability to predict match outcomes through such statistical measures is not just compelling for its application in sports analytics but also as a reflection of human behavior in competitive scenarios.

By pairing the dataset with robust statistical methods, we aim to investigate the strategic adaptations and team performance. The findings from our study have the potential to contribute to a more nuanced understanding of football tactics and to spark further study into the effects of psychological and environmental factors on high-level competition.

b. Research Question:

Question: What are the most influential statistical indicators for predicting the outcomes of FIFA World Cup matches, and how does their predictive accuracy vary across different tournament stages such as group matches, knockout rounds, and the final?

Rationale:

We aim to investigate the key statistical indicators that significantly influence the outcomes of FIFA World Cup matches. We are particularly interested in understanding how these indicators contribute to predicting match outcomes at various stages of the tournament, including group matches, knockout rounds, and the final. By identifying and analyzing these influential indicators, we intend to enhance our understanding of the factors that play a crucial role in determining match results throughout the different phases of this prestigious football event.

```
# imports and settings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import duckdb

from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split, KFold,
cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error,
mean_absolute_percentage_error, accuracy_score, precision_score,
recall_score, f1_score, precision_recall_curve, roc_auc_score,
roc_curve, auc
from sklearn import preprocessing

from scipy import stats
from sklearn.metrics import confusion_matrix

#read csv
wc_df = pd.read_csv("./wc_2022.csv", encoding='unicode_escape')
wc_df.head()

   match_no day_of_week       date   hour
venue  \
0         1         Sun  20-Nov-22  17:00                Al Bayt
Stadium
1         2         Mon  21-Nov-22  14:00  Khalifa International
Stadium
2         3         Mon  21-Nov-22  17:00             Al Thumama
Stadium
3         4         Mon  21-Nov-22  20:00          Ahmed bin Ali
Stadium
4         5         Tue  22-Nov-22  11:00          Lusail Iconic
Stadium

                         referee    group              1             2
\
0                Daniele Orsato  Group A          QATAR       ECUADOR

1                 Raphael Claus  Group B        ENGLAND          IRAN

2                Wilton Sampaio  Group A        SENEGAL   NETHERLANDS

3  Abdulrahman Ibrahim Al Jassim  Group B  UNITED STATES         WALES
```

```
4                  Slavko Vincic  Group C      ARGENTINA  SAUDI ARABIA


   attendance  ...  1_panelties_scored  2_panelties_scored  
1_goal_prevented  \
0       67372  ...                   0                   1  
6   
1       45334  ...                   0                   1  
8   
2       41721  ...                   0                   0  
9   
3       43418  ...                   0                   1  
7   
4       88012  ...                   1                   0  
4   

   2_goal_prevented  1_own_goal  2_own_goal 1_forced_turnovers  \
0                 5           0           0                 52   
1                13           0           0                 63   
2                15           0           0                 63   
3                 7           0           0                 81   
4                14           0           0                 65   

   2_forced_turnovers  1_defensive_pressure_applied  \
0                  72                           256   
1                  72                           139   
2                  73                           263   
3                  72                           242   
4                  80                           163   

   2_defensive_pressure_applied  
0                           279  
1                           416  
2                           251  
3                           292  
4                           361  

[5 rows x 59 columns]
```

c. Data Filtering and Column Concatenation

In this section, we perform data preprocessing to prepare our dataset for analysis. Specifically, we remove irrelevant columns and concatenate two existing columns into a new one.

1. **Removing Irrelevant Columns:**
   - We eliminate the following columns from the dataset:
     - `day_of_week`: This information is not needed for our analysis.
     - `hour`: The hour of the match is not relevant to our current study.
     - `referee`: The referee's name is not a focus of our analysis.

        - `1_forced_turnovers`: This column is not necessary for our current analysis.
        - `2_forced_turnovers`: Similarly, this column is not required for our analysis.
2. **Concatenating Columns:**
    - We create a new column called "Match," which combines the content of "team in 1" and "team in 2." This column will represent each match as "team in 1 vs. team in 2."

These data preparation steps ensure that our dataset is streamlined and ready for the analysis phase.

```
# Remove specified columns and create a new 'Match' column
clean_df = wc_df.drop(columns=['day_of_week', 'hour', \
                               'referee', '1_forced_turnovers', \
                               '2_forced_turnovers'])
conc_col = clean_df['1'] + ' vs ' + clean_df['2']

# conc_col=conc_col.to_list()
clean_df.insert(loc = 4,
          column = 'match',
          value = conc_col)

print(clean_df.shape)

#insert a column for the winner of each match, first create
def calculate_result(row):
    """
    Determine the result of a soccer match based on goal difference.

    Parameters:
    - row (pd.Series): A row from clean_df containing '1_goals' and
'2_goals'.

    Returns:
    - str: The result of the match. It can be one of three values:
      - Team 1's name, if team 1 wins.
      - Team 2's name, if team 2 wins.
      - 'Draw', if the match is a tie.
      """
    diff = row['1_goals'] - row['2_goals']
    if diff < 0:
        return row['2']
    elif diff > 0:
        return row['1']
    else:
        return 'Draw'

clean_df['Result'] = clean_df.apply(calculate_result, axis=1)
```

```
#replace some draw in games with results from penalty shoot-out,
#first create a dictionary of matches with the shoot-out winners
shoot_out_winners = {
    'JAPAN vs CROATIA': 'CROATIA',
    'MOROCCO vs SPAIN': 'MOROCCO',
    'CROATIA vs BRAZIL': 'CROATIA',
    'NETHERLANDS vs ARGENTINA': 'ARGENTINA',
    'ARGENTINA vs FRANCE': 'ARGENTINA'}

#update the 'Result' column based on the shoot-out results
for match, winner in shoot_out_winners.items():
    match_condition = (clean_df['match'] == match) &
(clean_df['Result'] == 'Draw')
    clean_df.loc[match_condition, 'Result'] = winner

#print updated results to check if the shoot-out winners are updated
print(clean_df[clean_df['match'].isin(shoot_out_winners.keys())]
[['match', 'Result']])


(64, 55)
                       match     Result
52          JAPAN vs CROATIA    CROATIA
54          MOROCCO vs SPAIN    MOROCCO
56         CROATIA vs BRAZIL    CROATIA
57  NETHERLANDS vs ARGENTINA  ARGENTINA
63       ARGENTINA vs FRANCE  ARGENTINA
```

1. **Grouping Teams into Stages**

- We group the columns by different stages of the World Cup with goals attempted and shots ontarget in order to showcase the frequnecy attempts and precision of the attempts in each match throughout the game

```
duckdb.sql(
"""
SELECT "group", "match", "1_attempts", "2_attempts", "1_ontarget",
"2_ontarget"
FROM clean_df
ORDER BY "group"
""").df()

          group                        match  1_attempts  2_attempts  \
0         Final          ARGENTINA vs FRANCE          21          10

1       Group A         NETHERLANDS vs QATAR          13           6

2       Group A       SENEGAL vs NETHERLANDS          14           9

3       Group A             QATAR vs ECUADOR           5           6
```

```
4       Group A            ECUADOR vs SENEGAL            8           15
..          ...                           ...          ...          ...
59  Round of 16              FRANCE vs POLAND           16           11
60  Round of 16  NETHERLANDS vs UNITED STATES           11           18
61  Round of 16        ARGENTINA vs AUSTRALIA           14            5
62   Semi-Final          ARGENTINA vs CROATIA           10           12
63   Semi-Final            FRANCE vs MOROCCO            14           13

    1_ontarget  2_ontarget
0            9           5
1            4           4
2            3           3
3            0           3
4            4           5
..         ...         ...
59           7           3
60           6           7
61           5           2
62           7           3
63           2           1

[64 rows x 6 columns]
```

2. Data Description: 2022 World Cup Results Dataset

What are the observations (rows) and the attributes (columns)?

- **Observations (rows)**:
  - Each observation represents a unique match from the 2022 World Cup.
- **Attributes (columns)**:
  - The dataset details the match number, day of the week, date, hour, venue, referee, group, and various statistics related to the performance of both teams.

Why was this dataset created?

- The dataset was created to capture detailed statistics and results of each match during the 2022 World Cup for potential insights into team performances, game dynamics, and other tournament aspects.

Who funded the creation of the dataset?

- Technique used for data collection is web scrapping and was done by Shrikrishna Parab.

What processes might have influenced what data was observed and recorded and what was not?

- Significance of specific statistics in understanding a football match.
- Availability and accuracy of technology for tracking specific statistics.
- Demand from stakeholders (e.g., coaches, teams, analysts) for specific data.

What preprocessing was done, and how did the data come to be in the form that you are using?

- Preprocessing steps
- Data cleaning to handle inaccuracies.
- Aggregating data from multiple sources.
- Handling missing data or imputing values.

If people are involved, were they aware of the data collection, and what purpose did they expect the data to be used for?

- In professional football, players and coaches typically expect data collection for performance analysis, broadcasting insights, and public dissemination.

Where can your raw source data be found, if applicable?

- Data can be found on Github https://github.com/shreeparab1890/Fifa-WC-2022-Qatar-Data-Analysis-EDA/blob/main/Fifa_WC_2022_Match_data.csv

Dataset Column Descriptions:

1. `match_no`: Match number in the tournament.
2. `day_of_week`: Day of the week when the match took place.
3. `date`: Date of the match.
4. `hour`: Time of day when the match started.
5. `venue`: Location where the match was held.
6. `referee`: Name of the referee officiating the match.
7. `group`: Group of teams to which the match belongs.
8. `1`: Team 1 identifier or name.
9. `2`: Team 2 identifier or name.
10. `attendance`: Number of spectators present at the match.
11. `1_xg`: Expected goals for Team 1.
12. `2_xg`: Expected goals for Team 2.
13. `1_poss`: Possession percentage for Team 1.
14. `2_poss`: Possession percentage for Team 2.
15. `1_goals`: Goals scored by Team 1.
16. `2_goals`: Goals scored by Team 2.
17. `score`: Final score of the match.
18. `1_attempts`: Total attempts by Team 1.

19. `2_attempts`: Total attempts by Team 2.
20. `1_conceded`: Goals conceded by Team 1.
21. `2_conceded`: Goals conceded by Team 2.
22. `1_goal_inside_penalty_area`: Goals scored by Team 1 from inside the penalty area.
23. `2_goal_inside_penalty_area`: Goals scored by Team 2 from inside the penalty area.
24. `1_goal_outside_penalty_area`: Goals scored by Team 1 from outside the penalty area.
25. `2_goal_outside_penalty_area`: Goals scored by Team 2 from outside the penalty area.
26. `1_ontarget`: On-target attempts by Team 1.
27. `2_ontarget`: On-target attempts by Team 2.
28. `1_offtarget`: Off-target attempts by Team 1.
29. `2_offtarget`: Off-target attempts by Team 2.
30. `1_attempts_inside_penalty_area`: Attempts by Team 1 inside the penalty area.
31. `2_attempts_inside_penalty_area`: Attempts by Team 2 inside the penalty area.
32. `1_attempts_outside_penalty_area`: Attempts by Team 1 outside the penalty area.
33. `2_attempts_outside_penalty_area`: Attempts by Team 2 outside the penalty area.
34. `1_yellow_cards`: Yellow cards received by Team 1.
35. `2_yellow_cards`: Yellow cards received by Team 2.
36. `1_red_cards`: Red cards received by Team 1.
37. `2_red_cards`: Red cards received by Team 2.
38. `faul_against_1`: Number of fouls against Team 1.
39. `faul_against_2`: Number of fouls against Team 2.
40. `1_offsides`: Offsides committed by Team 1.
41. `2_offsides`: Offsides committed by Team 2.
42. `1_passes`: Total passes made by Team 1.
43. `2_passes`: Total passes made by Team 2.
44. `1_passes_compeletd`: Total passes completed by Team 1.
45. `2_passes_compeletd`: Total passes completed by Team 2.
46. `1_corners`: Total corner kicks awarded to Team 1.
47. `2_corners`: Total corner kicks awarded to Team 2.
48. `1_free_kicks`: Total free kicks awarded to Team 1.
49. `2_free_kicks`: Total free kicks awarded to Team 2.
50. `1_panelties_scored`: Penalties scored by Team 1.
51. `2_panelties_scored`: Penalties scored by Team 2.
52. `1_goal_prevented`: Goals prevented by Team 1 (e.g., by goalkeeper or defense).
53. `2_goal_prevented`: Goals prevented by Team 2 (e.g., by goalkeeper or defense).
54. `1_own_goal`: Own goals scored by Team 1.
55. `2_own_goal`: Own goals scored by Team 2.
56. `1_forced_turnovers`: Forced turnovers by Team 1. Forced turnovers indicate instances where possession is lost as a result of pressure from an opposing player.

57. `2_forced_turnovers`: Forced turnovers by Team 2. Forced turnovers indicate instances where possession is lost as a result of pressure from an opposing player.
58. `1_defensive_pressure_applied`: Defensive pressure applied by Team 1.
59. `2_defensive_pressure_applied`: Defensive pressure applied by Team 2.
60. `Result`: Winning country in each match. If the game ended in a draw, the value is "Draw".
61. `1_pass_success`: Proportion of successful passes from total number of passes for Team 1.
62. `2_pass_success`: Proportion of successful passes from total number of passes for Team 2.
63. `winning_pass_success`: The pass success rate of the winning team.
64. `match`: The two countries competing in the game.
65. `Team_1_Goals_Diff`: The goals of Team 1 minus their expected goals.
66. `Team_2_Goals_Diff`: The goals of Team 2 minus their expected goals.

3. Preregistration statements

Hypothesis 1: Higher possession statistics (e.g., `1_poss` and `2_poss`) predict the match outcomes

- **Analysis**: Use a logistic regression model to analyze the effect of possession percentages (`1_poss` and `2_poss`) on match outcomes (`Result`). This analysis will be performed across matches, and the strength and significance of the possession coefficients in each of these stages will be compared to determine if their influence varies across matches.

Hypothesis 2: Matches in the knockout stages (Quarter-final, Semi-Final, and Final) have a higher average xG (expected goals) value for teams, indicating that these matches are more aggressive in nature as compared to the group stages.

- **Analysis**: To determine whether knockout stage matches in the World Cup are more attack-oriented, we'll compare their average expected goals (xG) to those in group stage matches. By using a t-test, we'll check for a statistically significant difference. If our p-value falls below 0.05 (we use an alpha level of 5%), it suggests knockout matches might indeed be more aggressive.

4. Data Analysis

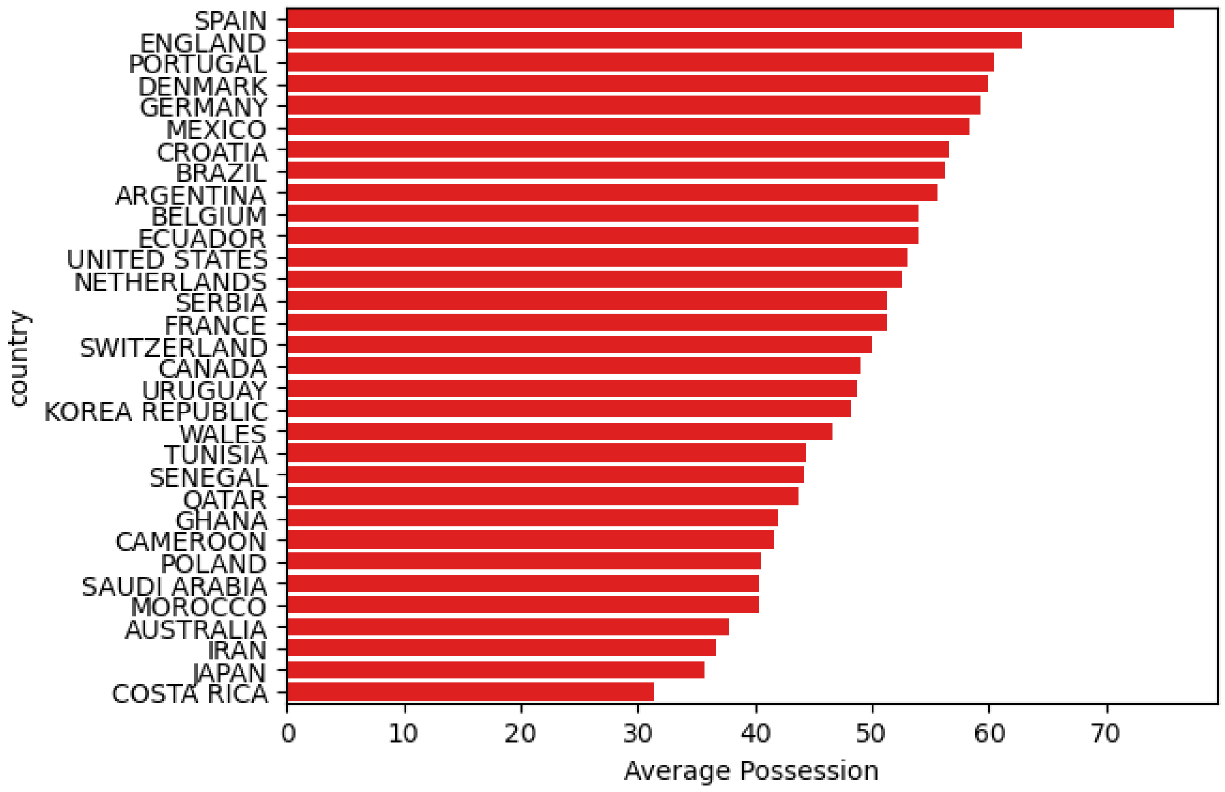
a. Data explorations

In this section we explored some interesting ways to evaluate the 2022 World Cup, including calculating the average possession per team in each match, the ranking of goals scored by each nation throughout the stage, the passing success rate of each team, team's actual performance vs. their expected goals, yellow cards, and red cards of the teams throughout the match. The purpose of these explorations is to give us inspirations on how to proceed with more refined analysis later on.

1: Calculating the average possession

```
#got all the games in the home
home = duckdb.sql("""
    SELECT DISTINCT  "1_poss" AS "possession", "1" AS "country"
    FROM wc_df
""").df()
#got all the games in the away
away = duckdb.sql("""  SELECT DISTINCT "2_poss" AS "possession" ,"2"
AS "country"
    FROM wc_df
 """).df()

#combining both of the data frames in order to calculate
games= pd.concat([home, away])
print(all)

Average_possesion = duckdb.sql("""select AVG(possession) AS "Average
Possession", country
           from games
           group by "country"
           order by "Average Possession" desc """).df()
sns.barplot(Average_possesion ,x="Average Possession", y="country",
color="r")
plt.show()
```

```
<built-in function all>
```



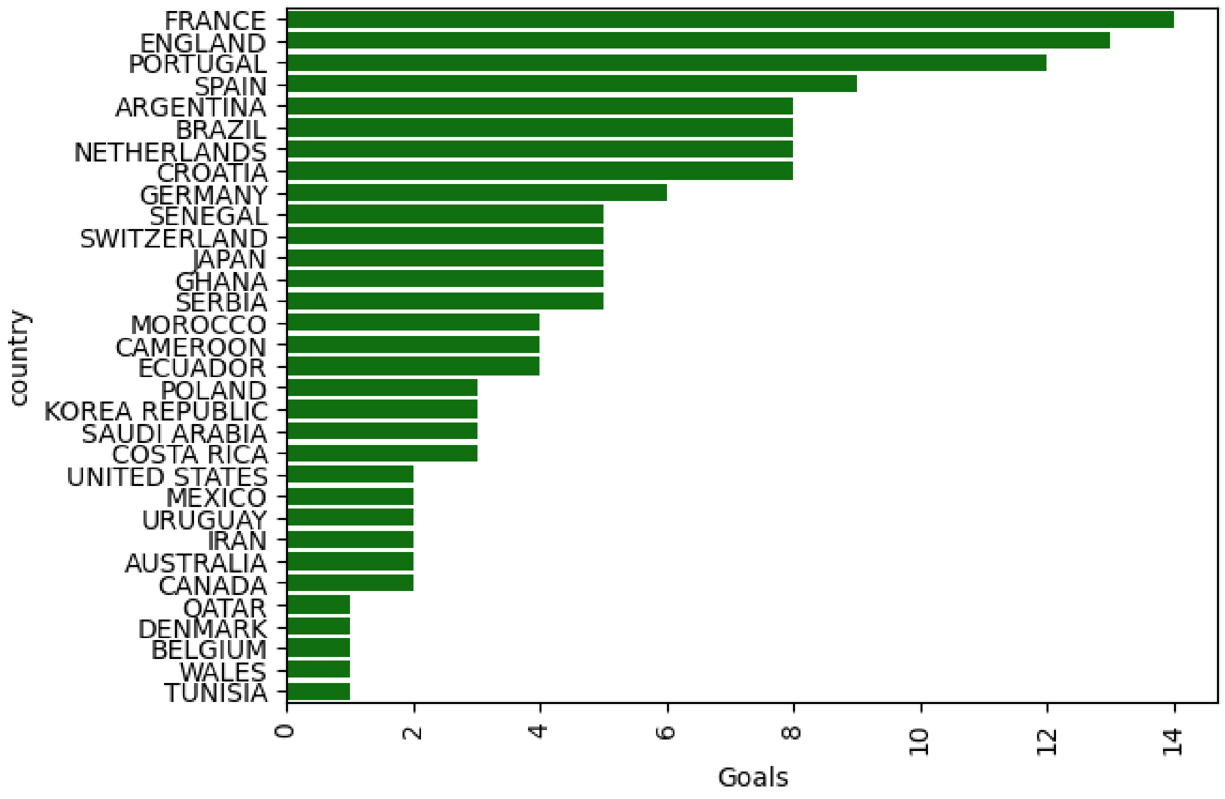
Interpertation for the graph above

The bar chart graphs the average possession percentages of teams in the World Cup 2022. Spain tops the chart with the highest average possession, indicating their playing style is likely characterized by maintaining control of the ball. The data descends through countries like England, Germany, down to Japan and Costa Rica, which have the lowest average possession, suggesting a more defensive or counter-attacking style.

2: Ranking the total goals scored by each country throughout the stages

```
#got all the games in the home
home_goals = duckdb.sql("""
    SELECT DISTINCT  "1_goals" AS "goals", "1" AS "country"
    FROM wc_df
""").df()

#got all the games in the away
away_goals = duckdb.sql("""  SELECT DISTINCT "2_goals" AS "goals" ,"2"
AS "country"
    FROM wc_df
 """).df()

#combining both of the data frames in order to calculate
goals= pd.concat([home_goals, away_goals])
```

```
total_goals = duckdb.sql("""select sum(goals) AS "Goals", country
            from goals
            group by "country"
            order by "Goals" desc """).df()

sns.barplot(total_goals ,x="Goals", y="country", color="g")
plt.xticks(rotation=90)
plt.show()
```



Interpertation for the graph above

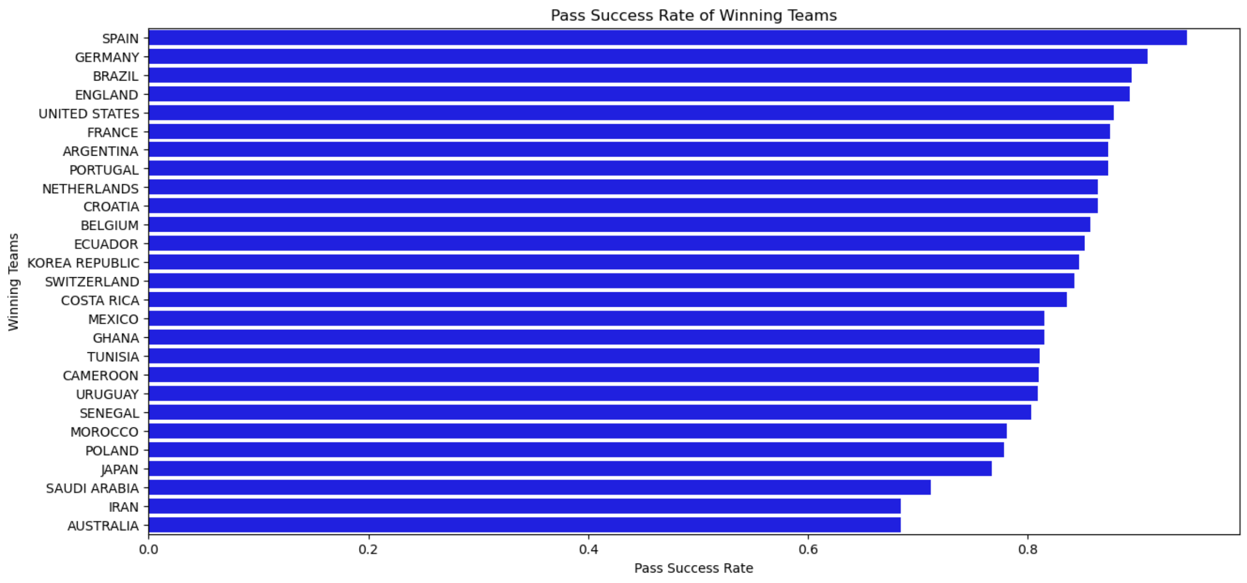
The bar chart illustrates the total goals scored by each country, presumably in a football tournament context. France leads with the highest number of goals, indicating a strong offensive performance. The chart shows a descending order of goals scored, with countries like England, Portugal, and Argentina also showcasing significant offensive prowess. In contrast, countries like Wales and Tunisia are at the bottom, indicating fewer goals scored during the matches considered. The uniform green color of the bars allows for a direct comparison of goal-scoring between each country, suggesting a correlation between a country's position on the chart and its attacking effectiveness during the tournament.

3: Comparing the success rate of the passes of the winning team with the losing team

```
#compare the success rate of the passes of the winning team with the
losing team
#compare the success rate of the passes of each round

clean_df["1_pass_success"] = clean_df["1_passes_compeletd"] /
clean_df["1_passes"]
clean_df["2_pass_success"] = clean_df["2_passes_compeletd"] /
clean_df["2_passes"]


#defining and performing questions
def winning_team_pass_success(row):
    """
    Calculate the pass success rate of the winning team for a given
match.

    Parameters:
    - row: A row from clean_df containing the columns 'Result',
      '1_pass_success', and '2_pass_success', representing the result
of the match
      and pass success rates of teams 1 and 2, respectively.

    Returns:
    - float: The pass success rate of the winning team. It can be one
of:
      - Team 1's pass success rate, if team 1 wins.
      - Team 2's pass success rate, if team 2 wins.
      - None, if the match is a tie."""

    if row['Result'] == row['1']:
        return row['1_pass_success']
    elif row['Result'] == row['2']:
        return row['2_pass_success']
    else:  # For draws
        return None

clean_df['winning_pass_success'] =
clean_df.apply(winning_team_pass_success, axis=1)

# Drop rows with NaN values in the winning_pass_success column
df1 = clean_df.dropna(subset=['winning_pass_success'])

# Group by 'Result' and calculate the mean of 'winning_pass_success'
df1 = duckdb.sql("""SELECT Result, AVG(winning_pass_success) AS
winning_pass_success
                    FROM df1
                    GROUP BY Result
```

```
                     ORDER BY winning_pass_success DESC;
                     """).df()

# Plot the data
plt.figure(figsize=(15, 7))
sns.barplot(data=df1, y="Result", x="winning_pass_success", color="b",
ci=None, orient='h')
plt.title('Pass Success Rate of Winning Teams')
plt.xlabel('Pass Success Rate')
plt.ylabel('Winning Teams')
plt.show()
```

```
/var/folders/49/nd093vm131sg2qd1qv__yfl00000gn/T/
ipykernel_88900/96339419.py:45: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(data=df1, y="Result", x="winning_pass_success",
color="b", ci=None, orient='h')
```

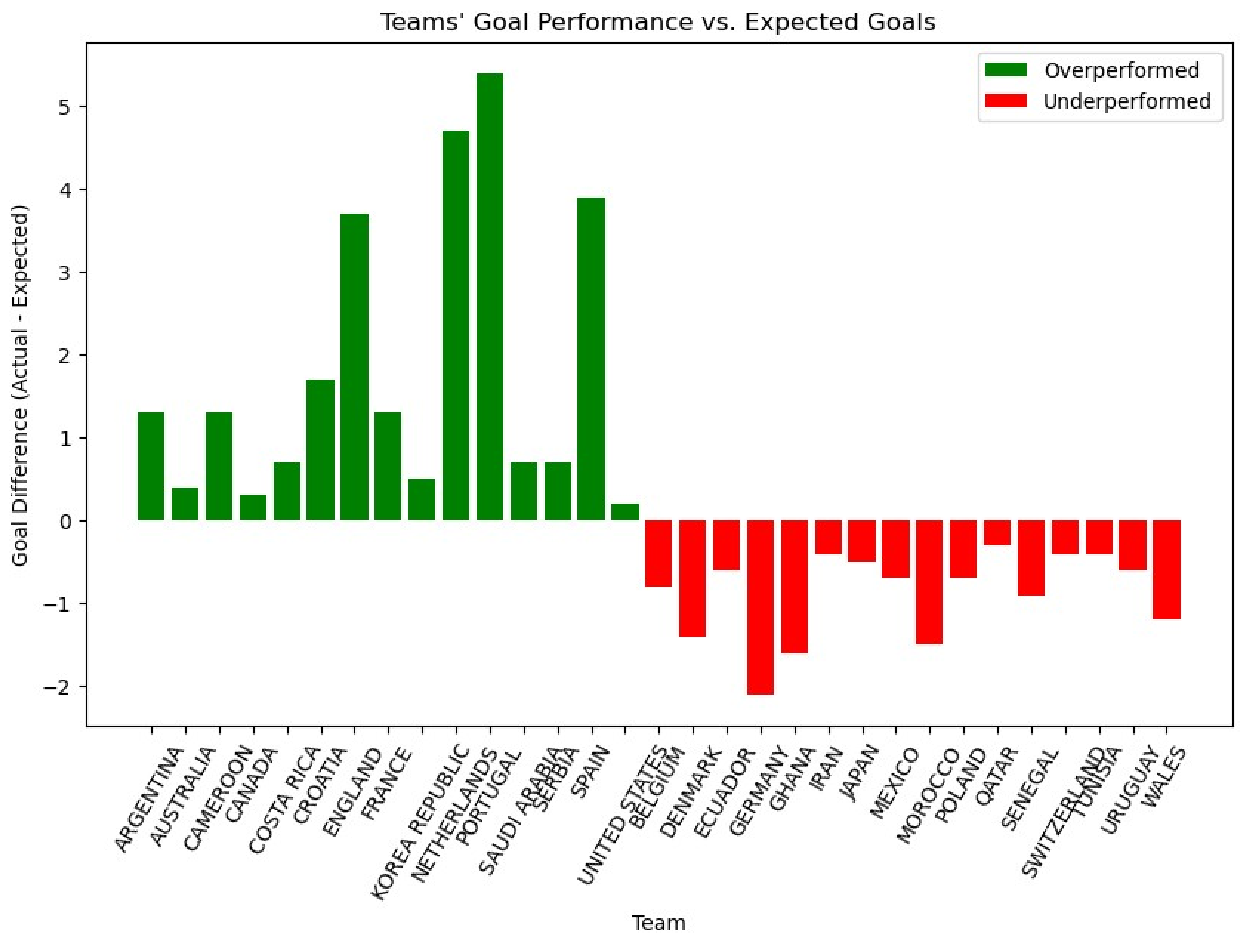


Interpertation for the graph above

The bar chart visualizes the pass success rate of winning football teams, with the teams listed on the y-axis and their corresponding pass success rates on the x-axis. Spain is shown to have the highest pass success rate, which suggests a strong correlation between accurate passing and winning matches for them. Other top teams, like Germany and Brazil, also have high pass success rates, reinforcing the importance of pass accuracy in successful outcomes. The graph illustrates a downward trend, with the teams at the bottom, such as Iran and Australia, having lower success rates, implying a potential area for improvement to enhance their chances of winning.

4: Teams' Goal Performance vs. Expected Goals

```
clean_df["Team_1_Goals_Diff"] = clean_df["1_goals"] - clean_df["1_xg"]
clean_df["Team_2_Goals_Diff"] = clean_df["2_goals"] - clean_df["2_xg"]

# Calculate the total overperformance or underperformance for each
team
team_1_overperformance = clean_df.groupby("1")
["Team_1_Goals_Diff"].sum()
team_2_overperformance = clean_df.groupby("2")
["Team_2_Goals_Diff"].sum()

# Identify the teams that have overperformed or underperformed
overperforming_teams = team_1_overperformance[team_1_overperformance >
0].index
underperforming_teams = team_1_overperformance[team_1_overperformance
< 0].index

# bar plot to visualize overperforming and underperforming teams
plt.figure(figsize=(10, 6))
plt.bar(overperforming_teams,
team_1_overperformance[overperforming_teams], label="Overperformed",
color="green")
plt.bar(underperforming_teams,
team_1_overperformance[underperforming_teams], label="Underperformed",
color="red")
plt.xlabel("Team")
plt.ylabel("Goal Difference (Actual - Expected)")
plt.title("Teams' Goal Performance vs. Expected Goals")
plt.legend()
plt.xticks(rotation=60)
plt.show()
```

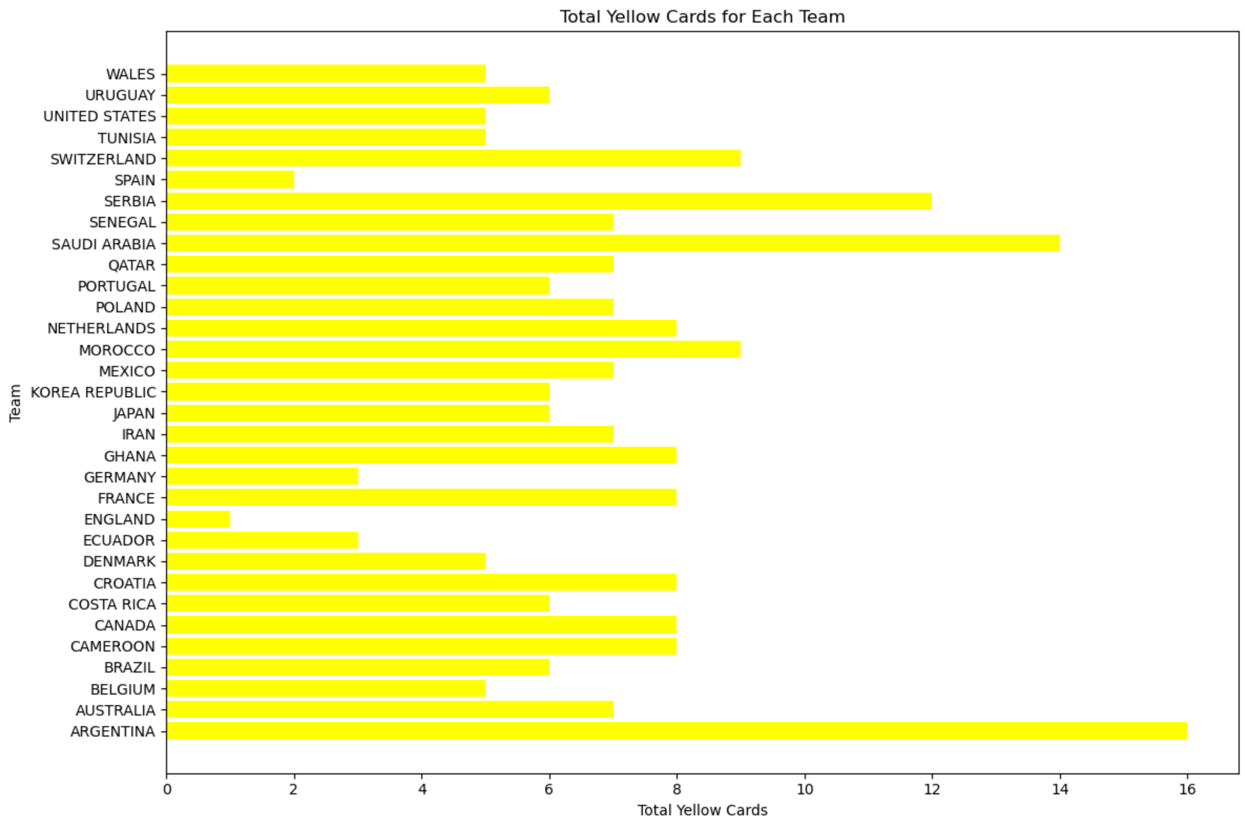


Teams' Goal Performance vs. Expected Goals

The graph illustrates the deviation of actual goals scored by teams from their expected goals (xG) in the World Cup 2022. Teams that have scored more than their xG are labeled as 'Overperformed' and are represented by green bars, indicating a positive goal difference. Conversely, teams with fewer goals than expected are categorized as 'Underperformed', denoted by red bars, reflecting a negative goal difference. This visual representation underscores the efficiency or lack thereof in converting goal-scoring opportunities. Teams like the Republic of Korea and Serbia significantly exceeded their xG, hinting at a high level of finishing precision or perhaps a streak of favorable outcomes. On the other hand, teams such as Uruguay and Wales underperformed relative to their xG, which might suggest missed opportunities or strong defensive opposition.

5: Yellow Cards of all teams

```
plt.figure(figsize=(12, 8)) # Adjust the figure size to fit your
display
plt.barh(total_yellow_cards['Team'],
total_yellow_cards['yellow_Cards'], color='yellow') # Note the 'barh'
for horizontal bar
plt.ylabel("Team")
plt.xlabel("Total Yellow Cards")
plt.title("Total Yellow Cards for Each Team")
plt.tight_layout() # Adjust the layout to make room for the y-labels
# Show the plot
plt.show()
```



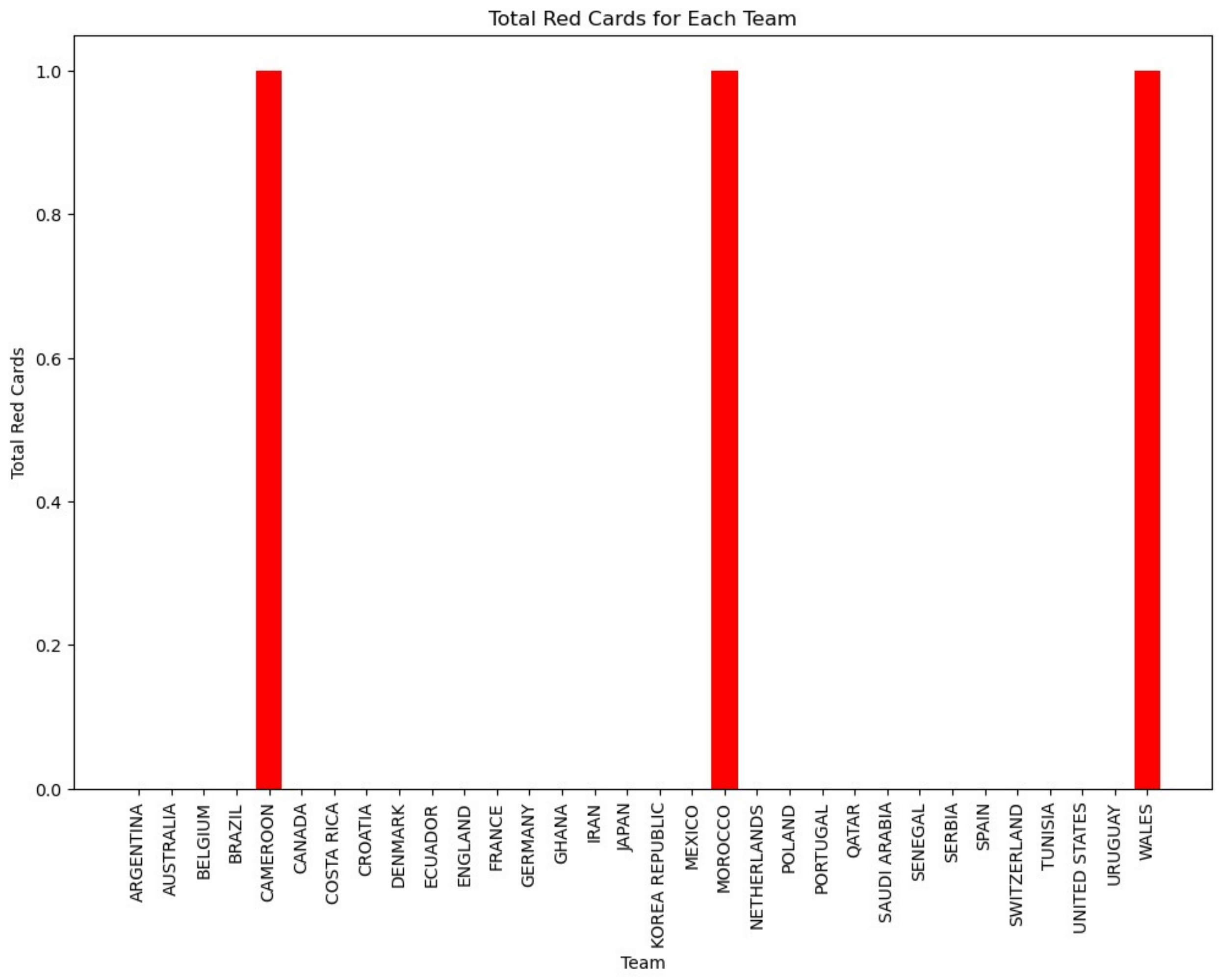
Yellow Cards Graph Interpertation

The bar chart provides a straightforward visualization of the total number of yellow cards received by each team during the World Cup 2022. The length of the bars corresponds to the number of yellow cards each team accumulated throughout the tournament, with Argentina receiving the highest number, indicative of a more aggressive style of play or a stricter interpretation of fouls by referees. In contrast, teams like Wales received the fewest, which could point towards a more disciplined approach to tackling and contesting for the ball.

6: Red Cards of all teams

```
team_1_red_cards = clean_df[['1', '1_red_cards']]
team_2_red_cards = clean_df[['2', '2_red_cards']]

# Rename columns to 'Team' and 'Red_Cards'
team_1_red_cards.columns = ['Team', 'Red_Cards']
team_2_red_cards.columns = ['Team', 'Red_Cards']
```

```
# Concatenate the two DataFrames
team_red_cards = pd.concat([team_1_red_cards, team_2_red_cards],
ignore_index=True)


total_red_cards = team_red_cards.groupby('Team')
['Red_Cards'].sum().reset_index()


plt.figure(figsize=(12, 8))
plt.bar(total_red_cards['Team'],total_red_cards['Red_Cards'],
color='red')
plt.xlabel("Team")
plt.ylabel("Total Red Cards")
plt.title("Total Red Cards for Each Team")
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



Red Cards Interpertation

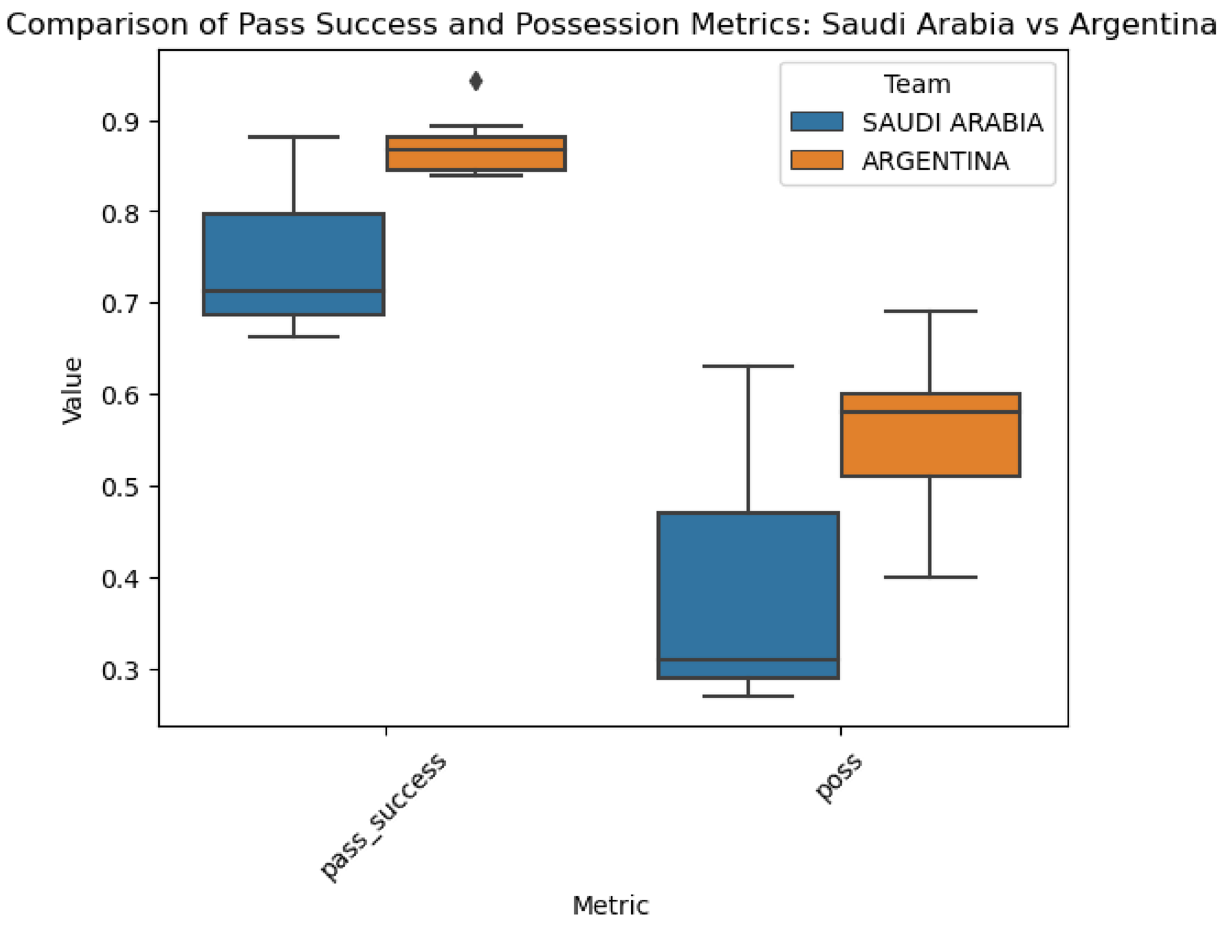
This bar chart displays the total red cards received by each team during the World Cup 2022. The majority of teams did not receive any red cards, maintaining a clean disciplinary record. However, a few teams, as indicated by the red bars, did have players sent off. The chart highlights the relatively infrequent occurrence of red cards in the tournament, reflecting either the teams' disciplined play or effective refereeing that managed to keep play fair without needing to resort to the most severe on-field sanctions.

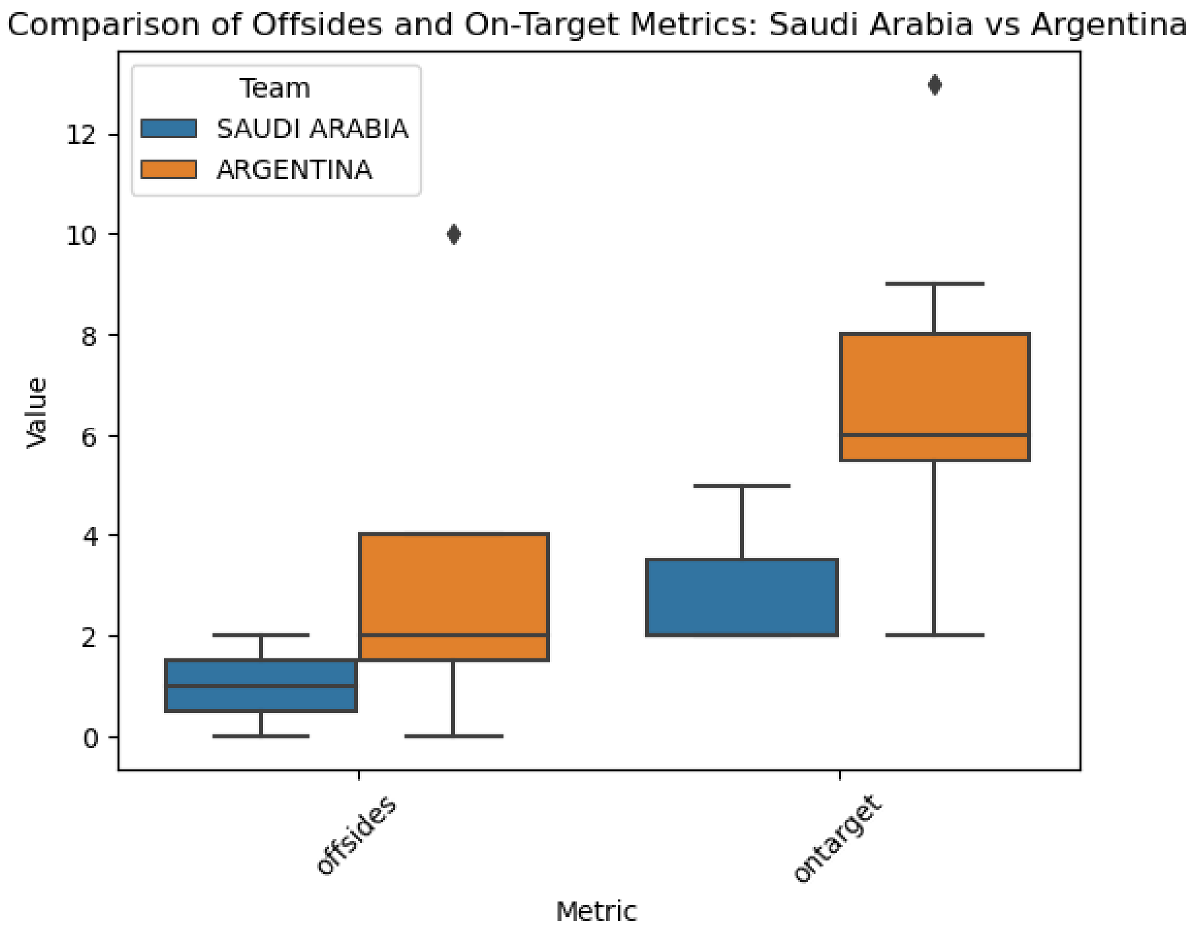
7: Comparing the successful pass rate and possession of Saudi Arabia and Argentina

Comparing offsides and shots ontarget of Saudi Arabia and Argentina

```
#pick out one game, comparing the average Saudi Arabian performance
metrics with
#their actual performance in the Argentina vs. Saudi Arabia match

clean_df["1_poss"] = clean_df["1_poss"] / 100
clean_df["2_poss"] = clean_df["2_poss"] / 100

# Retrieve metrics for all Saudi Arabia matches
saudi_metrics = duckdb.sql("""
    SELECT
        CASE
            WHEN "1" = 'SAUDI ARABIA' THEN "1_pass_success"
            WHEN "2" = 'SAUDI ARABIA' THEN "2_pass_success"
        END AS pass_success,
        CASE
            WHEN "1" = 'SAUDI ARABIA' THEN "1_offsides"
            WHEN "2" = 'SAUDI ARABIA' THEN "2_offsides"
        END AS offsides,
        CASE
            WHEN "1" = 'SAUDI ARABIA' THEN "1_ontarget"
            WHEN "2" = 'SAUDI ARABIA' THEN "2_ontarget"
        END AS ontarget,
        CASE
            WHEN "1" = 'SAUDI ARABIA' THEN "1_poss"
            WHEN "2" = 'SAUDI ARABIA' THEN "2_poss"
        END AS poss
    FROM clean_df
    WHERE "1" = 'SAUDI ARABIA' OR "2" = 'SAUDI ARABIA'
""").df()

# Retrieve metrics for all Argentina matches
argentina_metrics = duckdb.sql("""
    SELECT
        CASE
            WHEN "1" = 'ARGENTINA' THEN "1_pass_success"
            WHEN "2" = 'ARGENTINA' THEN "2_pass_success"
        END AS pass_success,
        CASE
            WHEN "1" = 'ARGENTINA' THEN "1_offsides"
            WHEN "2" = 'ARGENTINA' THEN "2_offsides"
        END AS offsides,
        CASE
            WHEN "1" = 'ARGENTINA' THEN "1_ontarget"
            WHEN "2" = 'ARGENTINA' THEN "2_ontarget"
        END AS ontarget,
        CASE
            WHEN "1" = 'ARGENTINA' THEN "1_poss"
            WHEN "2" = 'ARGENTINA' THEN "2_poss"
        END AS poss
    FROM clean_df
    WHERE "1" = 'ARGENTINA' OR "2" = 'ARGENTINA'
""").df()

# Add a 'team' column to distinguish between Saudi Arabia and
Argentina metrics
saudi_metrics['team'] = 'SAUDI ARABIA'
argentina_metrics['team'] = 'ARGENTINA'

# Concatenate the two dataframes
combined_df = pd.concat([saudi_metrics, argentina_metrics])

# Melt the dataframe for box plots
melted_df = pd.melt(combined_df, id_vars=['team'],
value_vars=['pass_success', 'poss', 'offsides', 'ontarget'],
                    var_name='metric', value_name='value')

# Possession and Pass Success Metrics
poss_pass_df = melted_df[melted_df['metric'].isin(['pass_success',
'poss'])]

# Offsides and On-target Metrics
offsides_ontarget_df = melted_df[melted_df['metric'].isin(['offsides',
'ontarget'])]

# Plotting
# First boxplot for pass_success and poss
sns.boxplot(data=poss_pass_df, x='metric', y='value', hue='team')
plt.title('Comparison of Pass Success and Possession Metrics: Saudi
Arabia vs Argentina')
plt.ylabel('Value')
plt.xlabel('Metric')
plt.legend(title='Team')
plt.xticks(rotation=45)
plt.show()
```

```
# Second boxplot for offsides and on-target
sns.boxplot(data=offsides_ontarget_df, x='metric', y='value',
hue='team')
plt.title('Comparison of Offsides and On-Target Metrics: Saudi Arabia
vs Argentina')
plt.ylabel('Value')
plt.xlabel('Metric')
plt.legend(title='Team')
plt.xticks(rotation=45)
plt.show()
```





Interpertation for the graph above

The two boxplot graphs compare performance metrics between Saudi Arabia and Argentina in a particular match or set of matches. The first graph compares pass success rates and possession, showing that Argentina has a tighter interquartile range for pass success and consistently higher possession values than Saudi Arabia, indicating more controlled gameplay.

The second graph compares offsides and on-target shots, revealing that Argentina has a higher median of on-target shots but also a greater number of offsides, which could suggest a more aggressive attacking strategy. Saudi Arabia, with fewer offsides and on-target shots, may have played more conservatively. These visuals serve to highlight the contrasting styles of play between the two teams, with Argentina taking a more aggressive approach and Saudi Arabia showing a potentially more measured strategy.

b. Research question models & testing

Hypothesis 1 Analysis

In our logistic regression analysis examining whether football match outcomes can be predicted using team possession statistics ('1_poss' and '2_poss'), we relied on the F1 score and ROC-AUC score to evaluate the model. The F1 score, a balance between precision and recall, was 74%, indicating a relatively high ability to predict accurately and consistently. The ROC-AUC score, reflecting the model's capability to distinguish between classes across all thresholds, was included for assessing model performance holistically.

The findings suggest possession statistics alone are insufficient predictors of football match results. The moderate F1 score points to a need for more nuanced models incorporating a broader array of factors, highlighting the complexity of predicting football outcomes and the limitations of relying on single metrics like possession percentages.

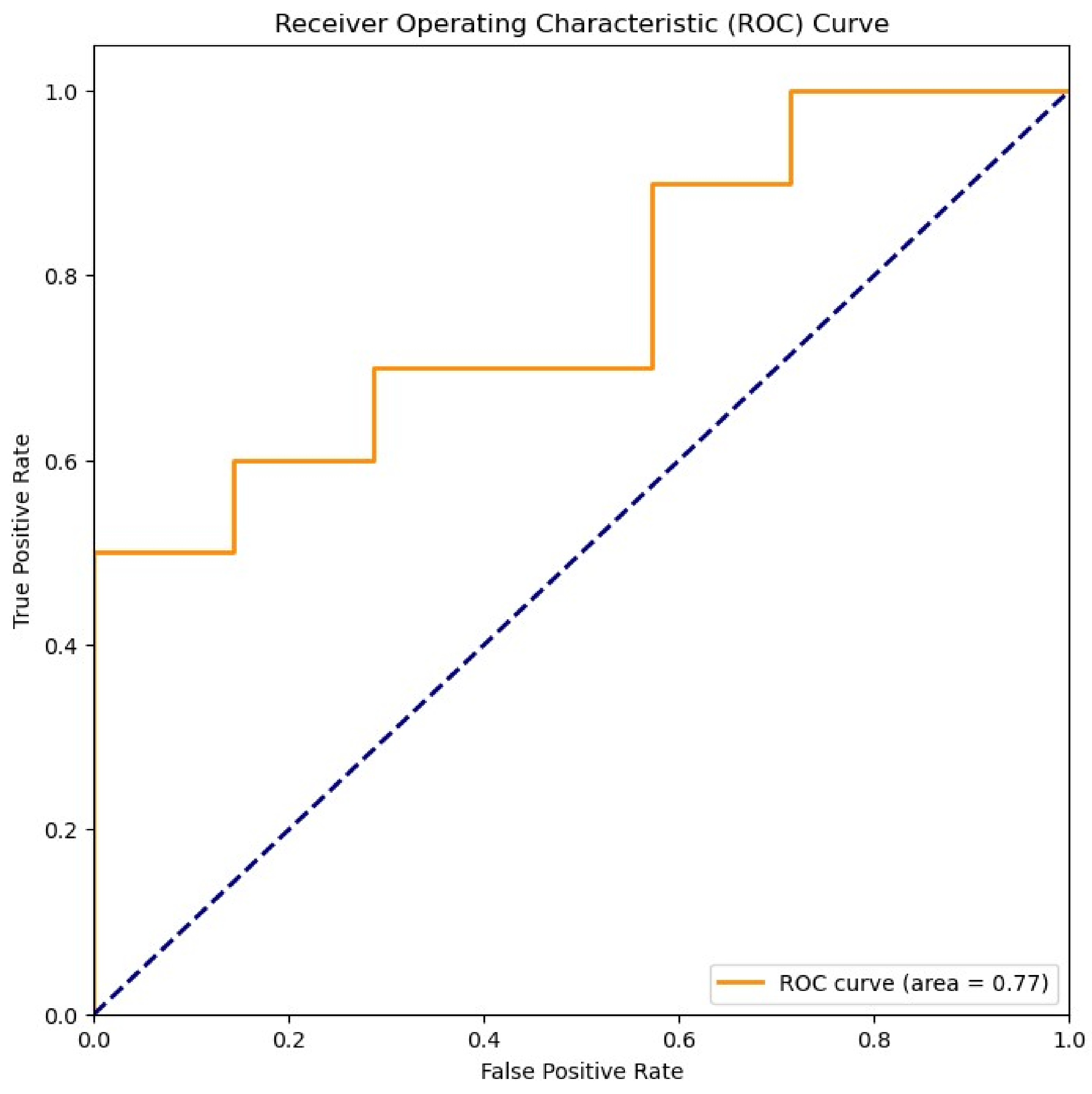
Overall, the analysis reveals the limitations of using possession percentages as sole indicators for predicting football match results. It points towards the need for more comprehensive models that incorporate a wider range of factors beyond simple possession metrics.

```
data=clean_df

# Dropping rows where the match result was a draw
data_no_draws = data[data['Result'] != 'Draw']

# Encoding the 'Result' as binary (1 for Team 1 win, 0 otherwise)
data_no_draws['Result_encoded'] = (data_no_draws['Result'] ==
data_no_draws['1']).astype(int)

X = data_no_draws[['1_pass_success', '2_pass_success']]
y = data_no_draws['Result_encoded']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=2950)

log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, y_train)
y_pred = log_reg_model.predict(X_test)

# Evaluating F1 score
f1 = f1_score(y_test, y_pred)

# Compute ROC curve and ROC area
y_prob = log_reg_model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
coefficients = log_reg_model.coef_[0]

# Calculating odds ratios for the coefficients
odds_ratio_1_pass_success = np.exp(coefficients[0])
odds_ratio_2_pass_success = np.exp(coefficients[1])

# Displaying the results
print(f"F1 Score: {f1:.2f}")
print(f"ROC-AUC Score: {roc_auc:.2f}")
print(f"Coefficient for '1_poss': {coefficients[0]:.4f}")
print(f"Coefficient for '2_poss': {coefficients[1]:.4f}")
print(f"Odds ratio for '1_poss': {odds_ratio_1_pass_success:.2f}")
print(f"Odds ratio for '2_poss': {odds_ratio_2_pass_success:.2f}")
```

```
F1 Score: 0.74
ROC-AUC Score: 0.77
Coefficient for '1_poss': 0.0558
Coefficient for '2_poss': 0.1899
Odds ratio for '1_poss': 1.06
Odds ratio for '2_poss': 1.21
```

```
/var/folders/49/nd093vm131sg2qd1qv__yfl00000gn/T/
ipykernel_88900/2539528740.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_no_draws['Result_encoded'] = (data_no_draws['Result'] ==
data_no_draws['1']).astype(int)
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc


# Plot ROC curve
plt.figure(figsize=(8, 8))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



Interpreting the ROC curve

The ROC curve provided indicates a model with a good ability to distinguish between the positive and negative outcomes of football matches, as shown by the area under the curve (AUC) of 0.77. This is considerably better than a random guess (an AUC of 0.5, shown by the dashed line).

The shape of the curve suggests that the model achieves a strong true positive rate without a corresponding increase in the false positive rate up to a certain point. Overall, the model demonstrates a relatively high degree of predictive power, but there may be room for improvement, especially in the threshold selection, whhich could be optimized to balance between sensitivity and specificity.

Evaluation of Significance

Interpretation of the relationship between '1_poss', '2_poss' and 'Result' Our model estimates that, all else equal, for each percentage point increase in Team 1's pass success rate ('1_poss'), the odds of Team 1 winning are multiplied by 1.06 (or an increase of 6% in odds). This suggests that better pass success for Team 1 has a positive but modest impact on their chances of winning, which aligns with the general understanding that effective possession can contribute to better match outcomes.

Our model also estimates that, all else equal, for each percentage point increase in Team 2's pass success rate ('2_poss'), the odds of Team 1 winning are multiplied by 1.21 (or an increase of 21% in odds). This counterintuitive finding implies that higher pass success rates for Team 2 might also increase Team 1's chances of winning, indicating a more nuanced or complex relationship between pass success rates and match outcomes than initially hypothesized.

Hypothesis 2 Analysis

- **Null Hypothesis**: There is no difference in the combined xG between group stage and knockout stage matches.
- **Alternative Hypothesis**: There is a difference in the combined xG between group stage and knockout stage matches.

Evaluation of Significance

In the OLS regression results:

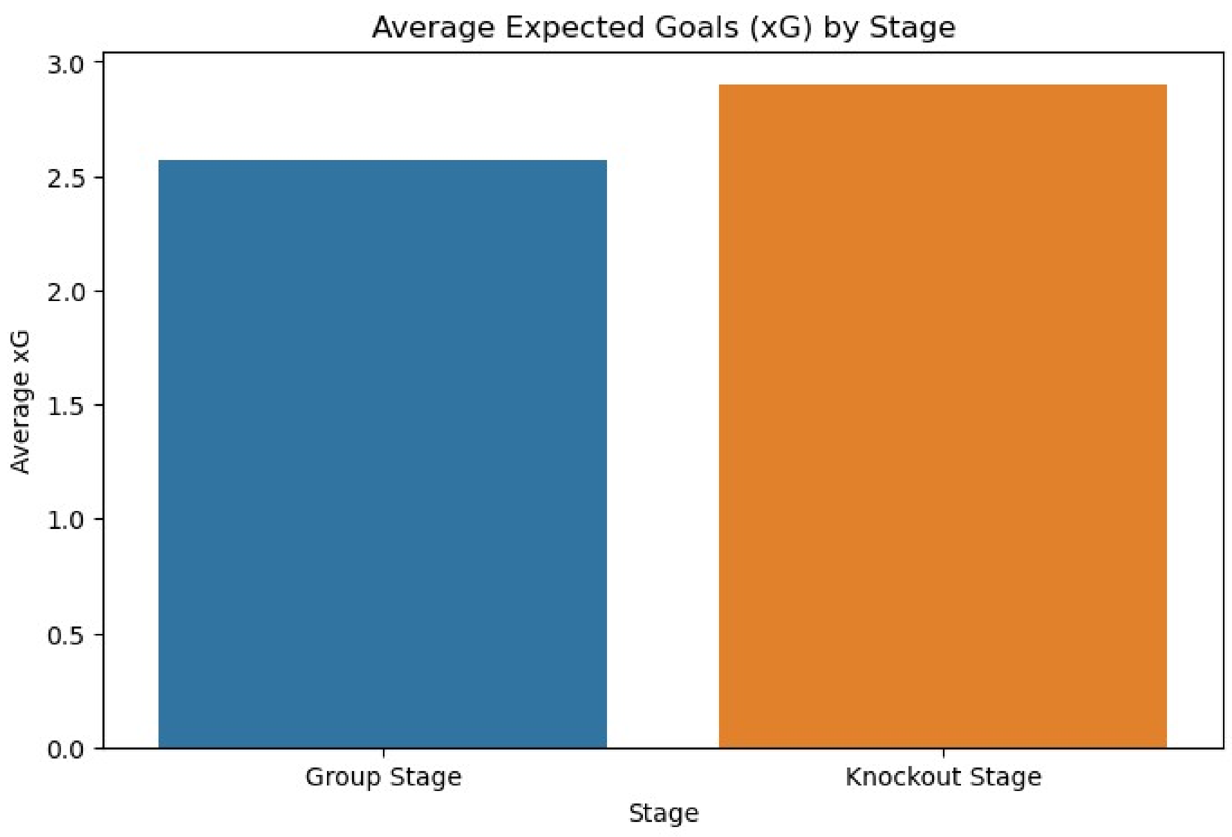
- The **Adjusted R-squared** is **0.001**, indicating that the model explains only 0.1% of the variance in the combined expected goals (xG), which is extremely low.
- The **F-statistic** is **1.032**, and the corresponding **p-value** is **0.314**.

The high p-value (0.314) associated with the F-statistic indicates that the results are not statistically significant at 0.05 alpha level. This means we do not have sufficient evidence to reject the null hypothesis. In other words, the analysis does not support the alternative hypothesis that knockout stage matches have a higher average xG compared to group stage matches. Instead, it suggests that the stage of the match (group vs. knockout) does not significantly influence the combined xG, implying similar levels of attacking play in both stages of the tournament.

```
import statsmodels.api as sm

data = clean_df
data['Stage_Type'] = data['group'].apply(lambda x: 0 if 'Group' in x
else 1)

# The dependent variable is the combined xG
data['Combined_xG'] = data['1_xg'] + data['2_xg']

# Fit an OLS model
model = sm.OLS(data['Combined_xG'],
sm.add_constant(data['Stage_Type']))
```

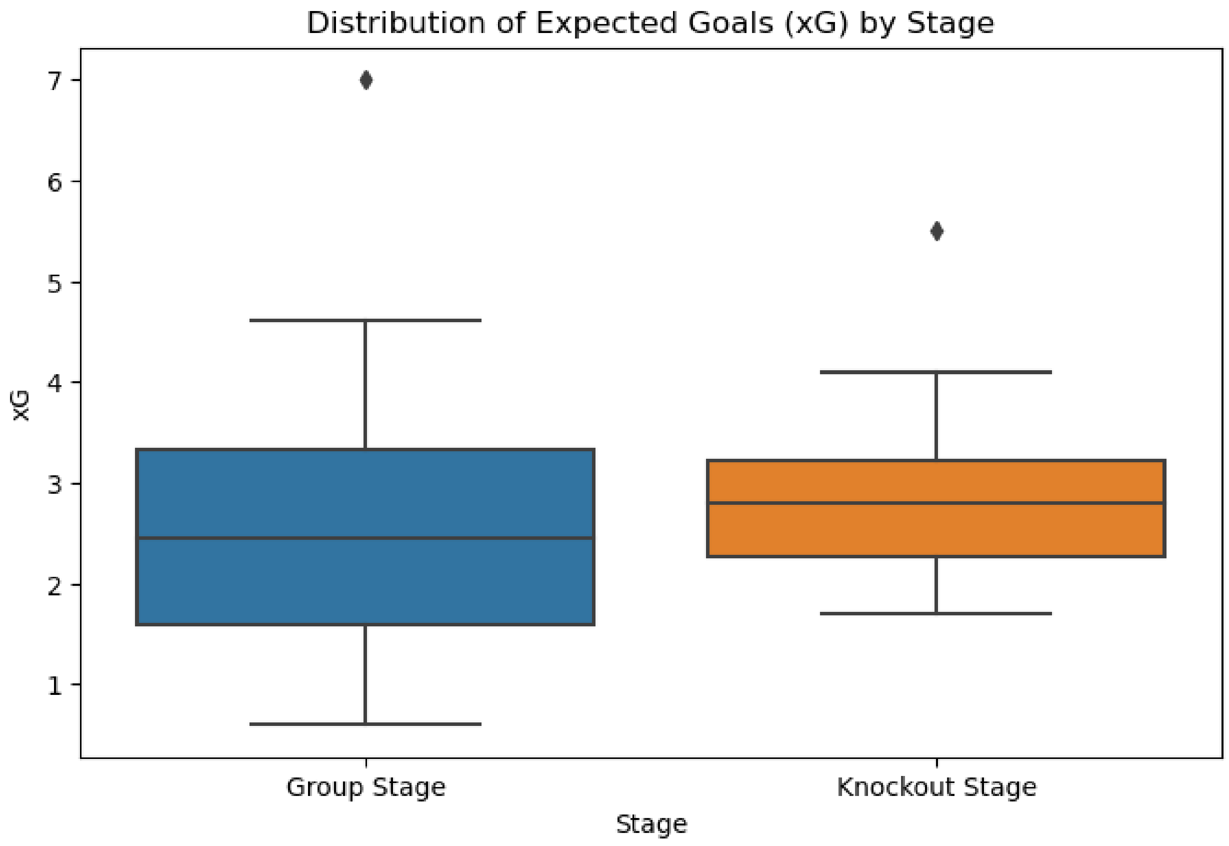
```
results = model.fit()

# Print the summary table
print(results.summary())

                            OLS Regression Results                            

==============================================================================
========
Dep. Variable:            Combined_xG   R-squared:                       
0.016
Model:                            OLS   Adj. R-squared:                  
0.001
Method:                 Least Squares   F-statistic:                     
1.032
Date:                Sun, 03 Dec 2023   Prob (F-statistic):              
0.314
Time:                        21:44:09   Log-Likelihood:                  
-97.979
No. Observations:                  64   AIC:                             
200.0
Df Residuals:                      62   BIC:                             
204.3
Df Model:                           1                                         

Covariance Type:            nonrobust                                         

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
--------
const          2.5667      0.164     15.648      0.000       2.239       
2.895
Stage_Type     0.3333      0.328      1.016      0.314      -0.322       
0.989
==============================================================================
========
Omnibus:                       18.751   Durbin-Watson:                   
1.992
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               
27.586
Skew:                           1.091   Prob(JB):                     
1.02e-06
Kurtosis:                       5.363   Cond. No.                         
2.48
==============================================================================
========
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
# Data preparation
clean_df['Stage_Type'] = clean_df['group'].apply(lambda x: 0 if
'Group' in x else 1)
clean_df['Combined_xG'] = clean_df['1_xg'] + clean_df['2_xg']

# Calculating average xG for group and knockout stages
avg_xg_group = clean_df[clean_df['Stage_Type'] == 0]
['Combined_xG'].mean()
avg_xg_knockout = clean_df[clean_df['Stage_Type'] == 1]
['Combined_xG'].mean()
avg_xg = [avg_xg_group, avg_xg_knockout]

# Defining stages for visualization
stages = ['Group Stage', 'Knockout Stage']

# Visualization - Bar Plot of Average xG
plt.figure(figsize=(8, 5))
sns.barplot(x=stages, y=avg_xg)
plt.title('Average Expected Goals (xG) by Stage')
plt.ylabel('Average xG')
plt.xlabel('Stage')
plt.show()
```



Bar Plot Interpretation: The bar plot displays the average expected goals (xG) for the group and knockout stages of the tournament, with the group stage shown in blue and the knockout stage in orange. While the average xG for the knockout stage appears slightly higher than for the group stage, the difference is not huge. The height of each bar corresponds to the mean xG value, allowing for a straightforward comparison. This visual suggests that while there might be a trend towards higher xG in the knockout stages, the difference is not large enough to draw definitive conclusions about the aggressiveness of play without more analysis.

```
# Preparing data for Box Plot visualization
group_stage = clean_df[clean_df['Stage_Type'] == 0]
knockout_stage = clean_df[clean_df['Stage_Type'] == 1]

# Creating combined xG for both stages
combined_xg = pd.concat([group_stage['Combined_xG'],
knockout_stage['Combined_xG']])
stage_labels = ['Group Stage'] * len(group_stage) + ['Knockout Stage']
* len(knockout_stage)

# Visualization - Box Plot of xG Values
plt.figure(figsize=(8, 5))
sns.boxplot(x=stage_labels, y=combined_xg)
plt.title('Distribution of Expected Goals (xG) by Stage')
plt.ylabel('xG')
plt.xlabel('Stage')
plt.show()
```



Box Plot Interpretation: The box plot illustrates the distribution of expected goals (xG) across group and knockout stages. The central line in each box indicates the median xG, and the box length represents the interquartile range (IQR), which shows the middle 50% of scores for each stage. Both stages show a similar median value, but the knockout stage displays a slightly wider IQR, suggesting more variability in the xG values. Outliers are present in both stages, indicated by diamonds, reflecting matches with exceptionally high xG values that fall outside the typical range. The similarity in the spread and central tendency between the two stages implies that the overall attacking intensity, as measured by xG, is quite consistent across the stages.

5: Data Limitations

- The data lacks comprehensive coverage of a game or season, with notable omissions such as injury reports, player transfers, and off-field events. These omissions can significantly impact the accuracy of analysis.
- The data might exhibit bias stemming from factors like home-field advantage, referee decisions, or player behavior, potentially distorting the interpretation of results.

- Certain events, like fouls and bookings (yellow and red cards), hinge on subjective referee decisions, introducing potential bias into the dataset.
- The games may have been influenced by unaccounted factors such as weather conditions, pitch quality, and crowd atmosphere, which, despite their significant impact on the game, are often missing from the datasets.
- National teams participating in the World Cup can display varying levels of talent and experience, creating fluctuations in the strength of competition from match to match. Considering this context is crucial for proper analysis.

Specific World Cup Considerations

- The knockout stages of the World Cup introduce single-elimination matches, increasing the unpredictability of outcomes.
- Qatar's potential home-field advantage could sway results.
- The World Cup's four-year cycle and the relatively limited number of matches may lead to increased variability in statistics due to smaller sample sizes.

6: Sources used

- https://matplotlib.org/stable/users/index.html
- https://matplotlib.org/stable/tutorials/index.html
- https://numpy.org/doc/
- https://seaborn.pydata.org/tutorial.html
- https://duckdb.org/docs/
- https://www.geeksforgeeks.org/numpy-tutorial/
- https://www.datacamp.com/tutorial/seaborn-python-tutorial

7: Acknowledge

- Github: we are very thankful for Shrikrishna Parab for scraping the 2022 FIFA World Cup, in which his introduction can be found here: https://github.com/shreeparab1890/Fifa-WC-2022-Qatar-Data-Analysis-EDA/blob/main/README.md