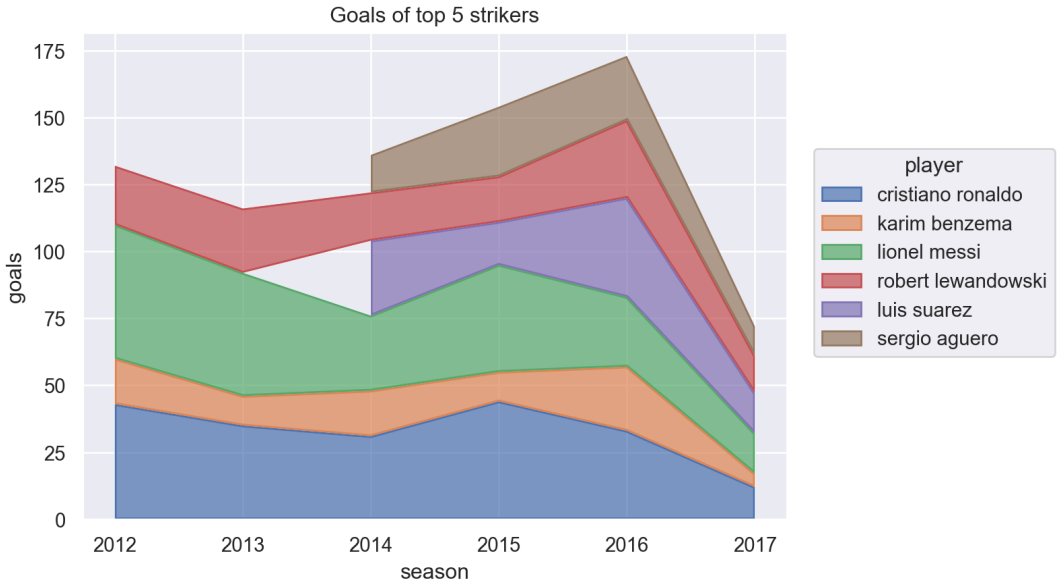
CS5764: Term Project Submission

Information Visualization of Football Events Dataset



Author : Prem Banker
School: Virginia Tech
Instructor: Dr. Reza Jafari
Date: 06/12/2023
Course: Information Visualization (CS5764)
Project URL: https://dashapp-mfdvcv47tq-nn.a.run.app/

Abstract:

This information visualization project explores the intricate dynamics of football events through an immersive and interactive platform. Leveraging a comprehensive dataset encompassing diverse football events, our project employs advanced visualization techniques to provide users with intuitive tools for uncovering patterns, trends, and key moments within the realm of football. Through interactive graphs, timelines, and player performance metrics, this project aims to enhance the understanding and appreciation of football analytics, catering to both enthusiasts and analysts alike.

Introduction:

Football, as a global phenomenon, generates a wealth of data during each match, capturing the essence of the game through a multitude of events such as goals, fouls, and player interactions. This project delves into the vast landscape of football events, seeking to transform raw data into meaningful visual narratives. By harnessing the power of information visualization, our goal is to create an engaging platform that not only simplifies the exploration of football data but also deepens the understanding of the sport's nuances.

The project's scope encompasses the development of visualizations that span the entirety of football events—ranging from Individual players stats to team stats and how some teams are faring and how others are doing. The major key aims of the project are to understand:

- What have been the team records in general ? What are the splits of wins, losses and draws
- How do the teams fare at home vs at away grounds ?
- Has football gotten any more attacking or less attacking ?
- Who have been the best teams and most attacking teams ?
- Who have been the best teams and most attacking teams ?
- Which players have been the most dominant throughout?

The aim of the project is to not only answer these but also just represent the data in a manner which suits both enthusiasts and analysts. The aim is to also deploy this application on the cloud hence it would be available to everyone across the globe alike.

Through this exploration, we aim to bridge the gap between raw data and insightful comprehension, fostering a new level of engagement with football analytics. Our interactive platform invites users to embark on a visual journey, uncovering the stories embedded within the events that define the beautiful game.

Description of the Dataset:

The Dataset is called the Football Events Dataset which has been sourced from Kaggle. The dataset involves data about different types of events in football namely goals, assist, saves, fouls etc. Additionally it also provides top level stats like game scores, and match odds as well. The dataset is divided into two files:

- Events.csv : Consists of >900,000 records which contains details of granular events in football like passes, saves, fouls, substitutions, etc.
- Ginf.csv : Consists of >10,000 records which contains the top-level game stats like away team, home team, away team scores. Etc.

While the dataset contains a large number of variables, a lot of these variables are not necessary for my study. The rejected variables would be discussed later on. The important variables in Events.csv include the follows:

- Event Type: The type of the event (goal, save, pass, through ball, etc.)
- Player: Who performed the action
- Player 2: Who assisted in the action
- Is_goal: Determining if the event resulted in a goal
- Event_team: The team who performed the action
- Opponent: The opposing team

The important variables in ginf.csv include the following:

- League: The league in which the game was played
- Season: The year in which the game was played
- Ht: The home team of the game
- At: The away team of the game
- Fthg: The number of goals scored by the home team
- Ftag: The number of goals scored by the away team

Pre Processing Dataset:

While my dataset included a lot of Null and Nan values because of the nature of the data (For e.g. a column called ‘player_out’ which indicates that a player was taken out due to substitution is irrelevant to almost 99% percent of the dataset), the actual cleaning part was rather unconventional for me. Instead of removing the dataset row by row, I eliminated the columns and thereby morphing my dataset into multiple dataframes useful for different use cases. The first major transformation was creating a TeamSeason dataframe from the Ginf.csv where the non-relevant columns were filtered out and appropriate columns were created from the dataset. The newly processed data frame looked as follows:

	name	season	wins	losses	draws	goals_scored	goals_conceded
0	Borussia Dortmund	2012	25	3	6	80	25
1	Hamburg SV	2012	8	14	12	35	57
2	FC Augsburg	2012	8	12	14	36	49
3	SC Freiburg	2012	10	14	10	45	61
4	Werder Bremen	2012	11	14	9	49	58

Fig1: Displaying the head of one of the morphed dataframe

The columns like **'id_odsp', 'link_odsp', 'adv_stats', 'date'** were eliminated from ginf.csv and the above data frame was created. Apart from that, since working on a large dataset which had a lot of non-helpful rows, the approach mainly relied on morphing the data into leaner chunks according to the requirement of the visualization that I wanted to achieve. Another such visualization included creating a player stat data frame which highlighted the individual player statistics for year by year. The class for the creation looked as follows:

```python
class TeamSeason:
    def __init__(self, name,season, wins=0, losses=0, draws=0, goalsScored=0, goalsConceded=0):
        # Instance variables
        self.name = name
        self.season = season
        self.wins = wins
        self.losses = losses
        self.draws = draws
        self.goalsScored = goalsScored
        self.goalsConceded = goalsConceded

    def addWin(self):
        self.wins+=1

    def addLoss(self):
        self.losses+=1

    def addDraw(self):
        self.draws+=1

    def getTotalGames(self):
        return self.wins + self.losses + self.draws

    def addGoalsScored(self,n):
        self.goalsScored+=n

    def addGoalsConceded(self,n):
        self.goalsConceded+=n
```

Fig2: Shows the class used for forming dataframe in Fig1

Outliers:

While removal of outliers was not necessary for this specific dataset, there were few interesting data points which were discovered by doing outlier analysis. The first was the distribution of away and home team goals. It was observed that more goals were scored by the home teams whereas away teams scored a little less goals and hence, the outliers vary in them. For instance, no away team had scored **>7 goals** in a game whereas there were instances where the home team had reached that target.

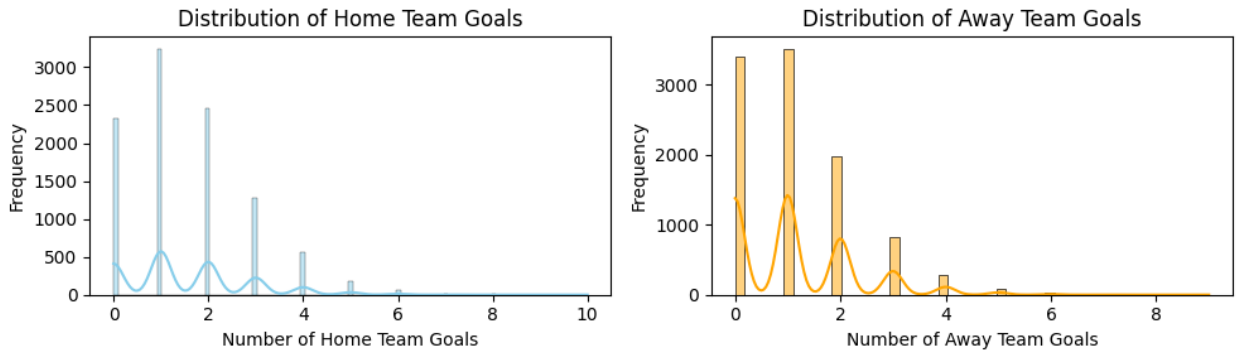


Fig3: Showing the distribution of goals by home and away teams

Another interesting observation done while observing the outliers for goals scored. Surprisingly, the outlier range of goals scored was very less. It was observed that a player who scored more than **>8.5 goals was considered an outlier**. While at first glance, the observation seems untrue as an average for a striker is about 15 goals a season. However after further analysis, the 8.5 outlier range made more sense as the dataset also included **defenders, midfielders and not just attackers** which would drive the average down and hence outlier range would also lower down. The observations were made by using IQR and outliers were classified by 1.5*IQR added to Q1 and Q3.

```
Q1 and Q3 of the goals by players is 1.0 to 4.0

IQR for goals scored is 3.0

Any goals scored more than 8.5 or less than -3.5  is an outlier
```

Fig4: Showing the outlier range for goals scored

Principal Component Analysis:

For this project, PCA was not required as predictions and concepts of dependent and independent variables were not required.

Normality Test:

Several normality tests were run on different columns of the data frame we used. The results for the normality test are as follows:

1) Wins:

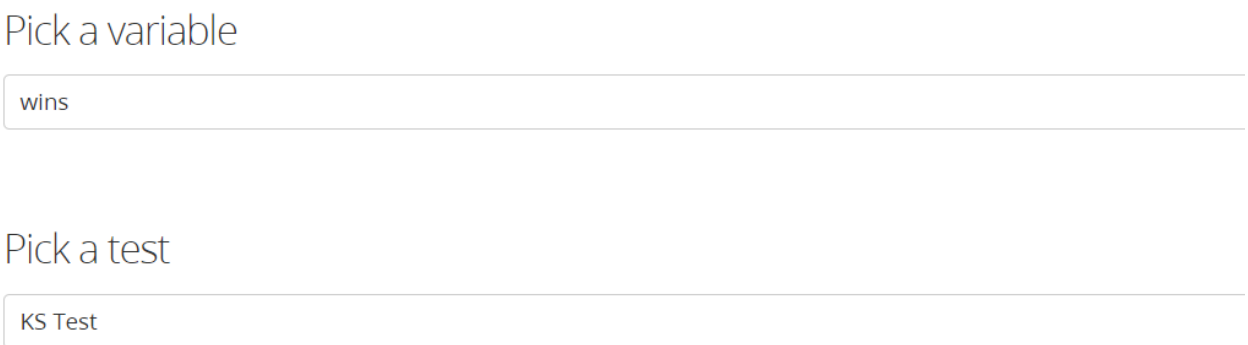


Fig5: Indicating the K-S Test output of wins showing that the column is normally distributed

2) Losses:

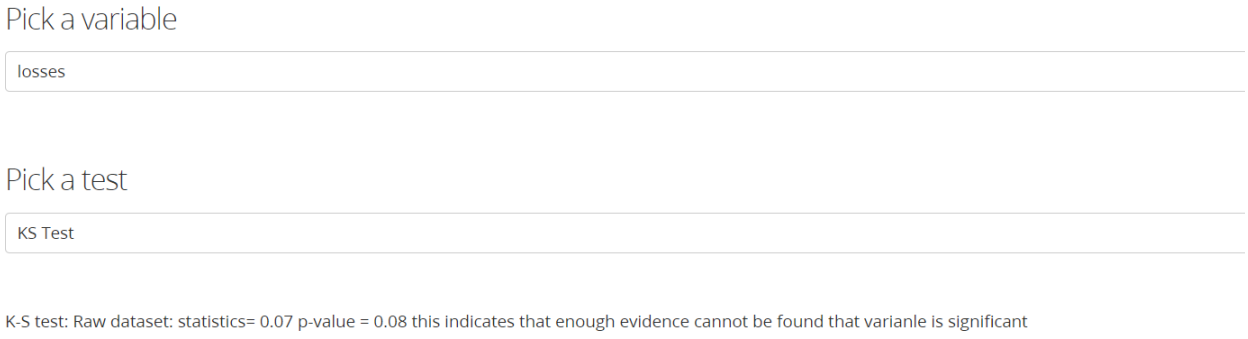


Fig6: Indicating the K-S Test output for losses which shows that column is not normally distributed.

Similarly the other columns were observed and they passed the normality test. So, losses was the column which did not pass the normality test.

Heatmap and Pearson correlation coefficient matrix.

The following was the output of the correlation coefficient matrix.

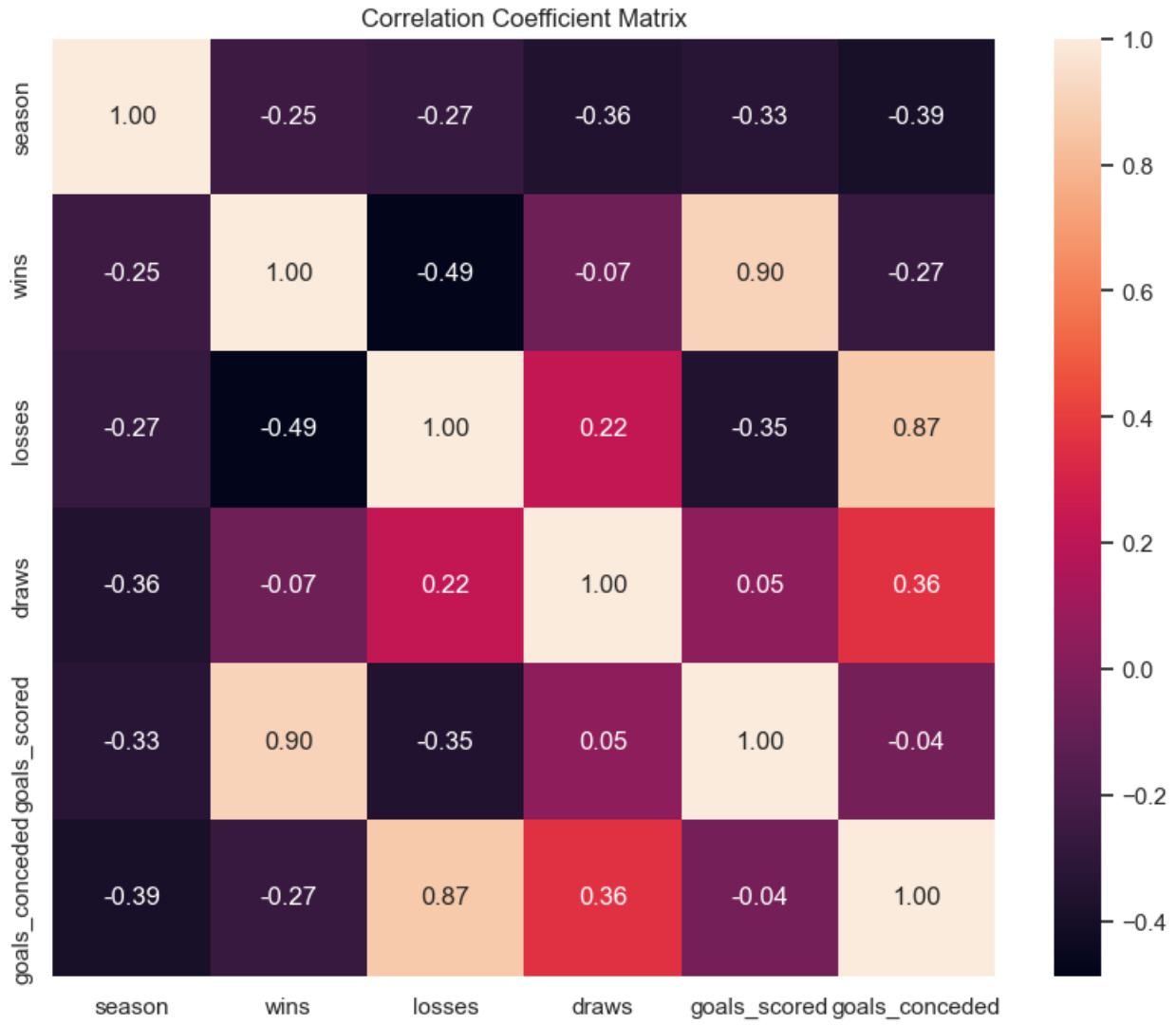


Fig7: Correlation coefficient matrix for the Team Seasons dataframe

Some interesting observations can be made from the above information:

- Wins and losses are strongly correlated in a negative direction. This is expected as the more the team wins, the less it looses
- Goals scored and games won has a very **strong correlation with 0.9**

- Similarly games lost and goals conceded have a very strong correlation equal to **0.87**
- Another interesting observation was wins and goals_conceded not having an extremely strong correlation (-0.27). It can be attributed to the fact that teams usually outscore each other more and worry less about conceding goals.

Statistics:

Here is a multivariate Boxen Plot for analyzing different columns:

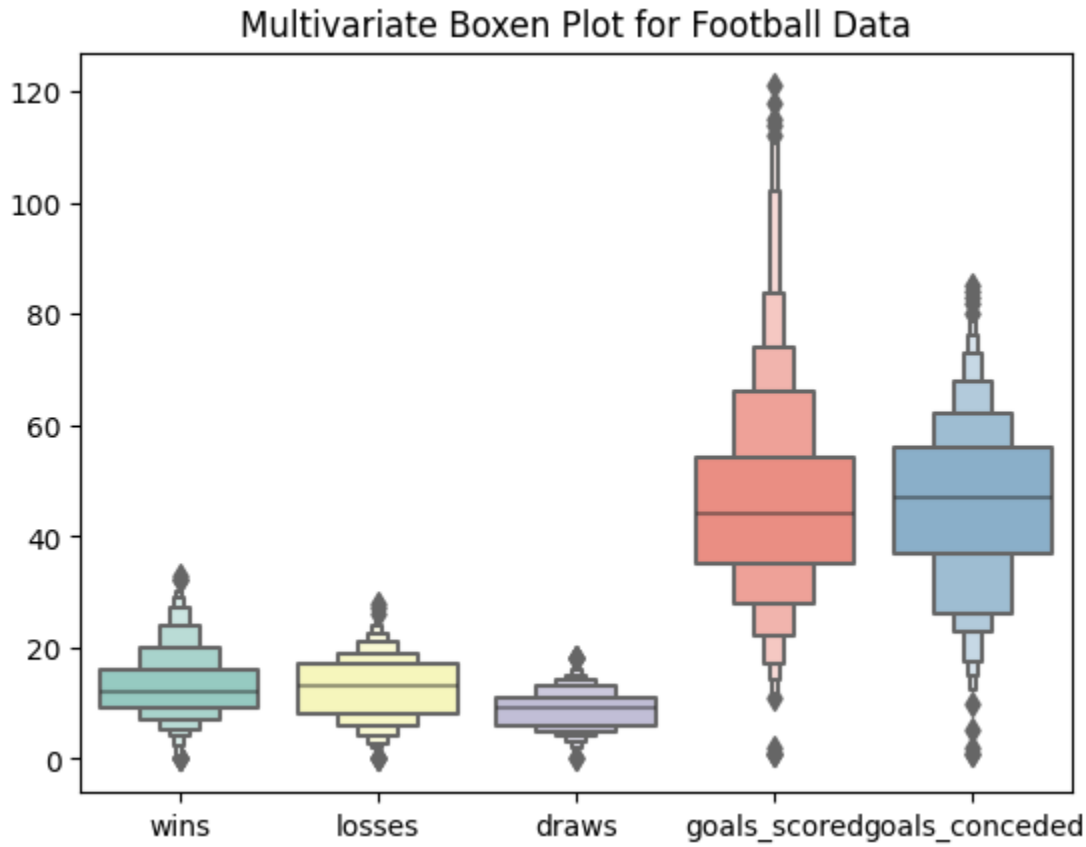


Fig8: Multivariate Boxplot analyzing different variables

As observed, goals_scored and goals_conceded have considerably higher mean than wins, losses and draws. This can be attributed to the fact that there are multiple goals scored throughout the game.

The following table describes the numeric values of the above seen graph

	season	wins	losses	draws	goals_scored	goals_conceded
count	592.00	592.00	592.00	592.00	592.00	592.00
mean	2014.50	12.73	12.73	8.69	46.17	46.17
std	1.71	6.05	5.56	3.32	18.09	15.03
min	2012.00	0.00	0.00	0.00	1.00	1.00
25%	2013.00	9.00	8.00	6.00	35.00	36.75
50%	2014.50	12.00	13.00	9.00	44.00	47.00
75%	2016.00	16.00	17.00	11.00	54.00	56.00
max	2017.00	33.00	28.00	18.00	121.00	85.00

Fig9: Numeric values of the Fig8

An interesting view here is the number of **wins maximum is 33** which is way out of the outlier range. This was the peak of the Barcelona team which had arguably the best team of all time recording 33 wins during the 2015 season.

Data Visualization:

1) Bar plot:

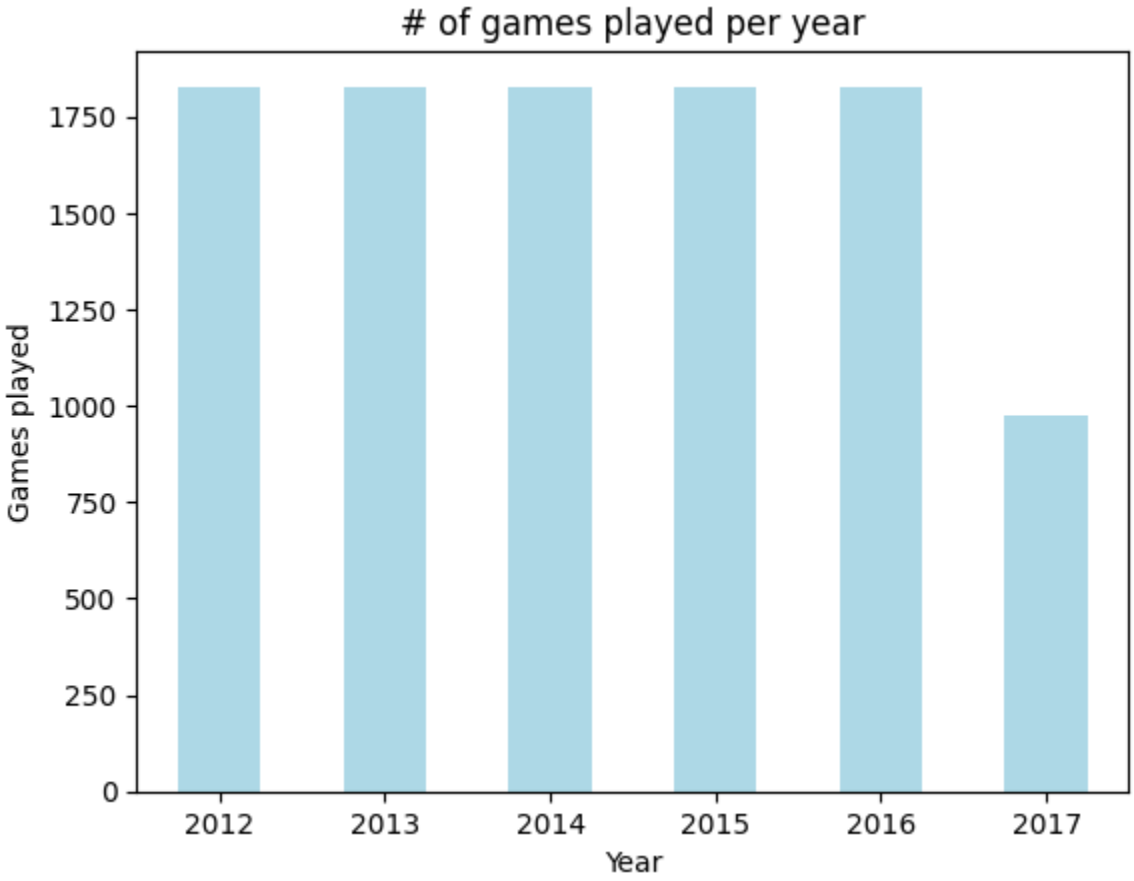


Fig10: Displaying the number of games per year

This was an elementary graph plotted to understand the time period of the dataset we have. We are dealing with data from 2012 till midway of 2017. As we only have partial games data available for 2017, the plot is shorter there.

2) Line plots:

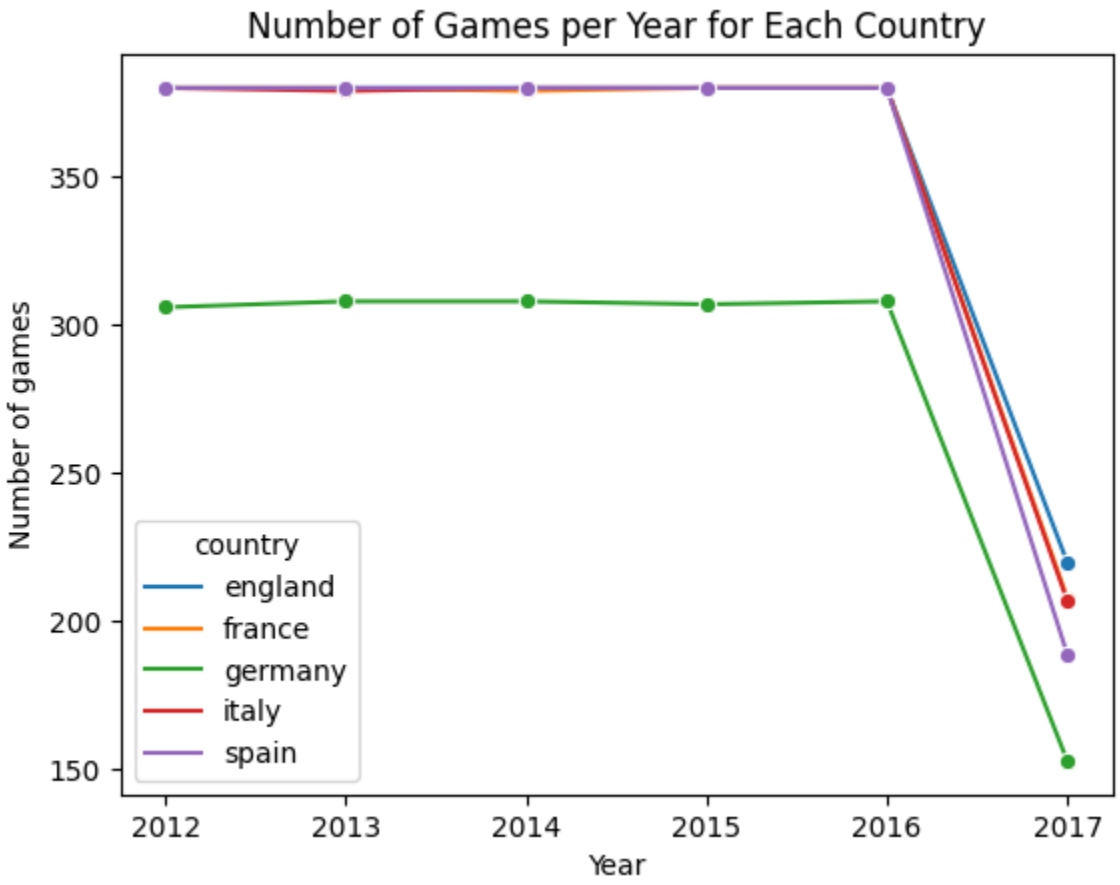


Fig11: Shows the line plot of games played per league

As we have data of different leagues, this plot was made to ensure we do **not have bias for one league.** As seen here, Germany consistently averages less games. That is because **Germany has 18 teams** whereas **other countries have 20 teams**.

3. Stacked Bar plot:

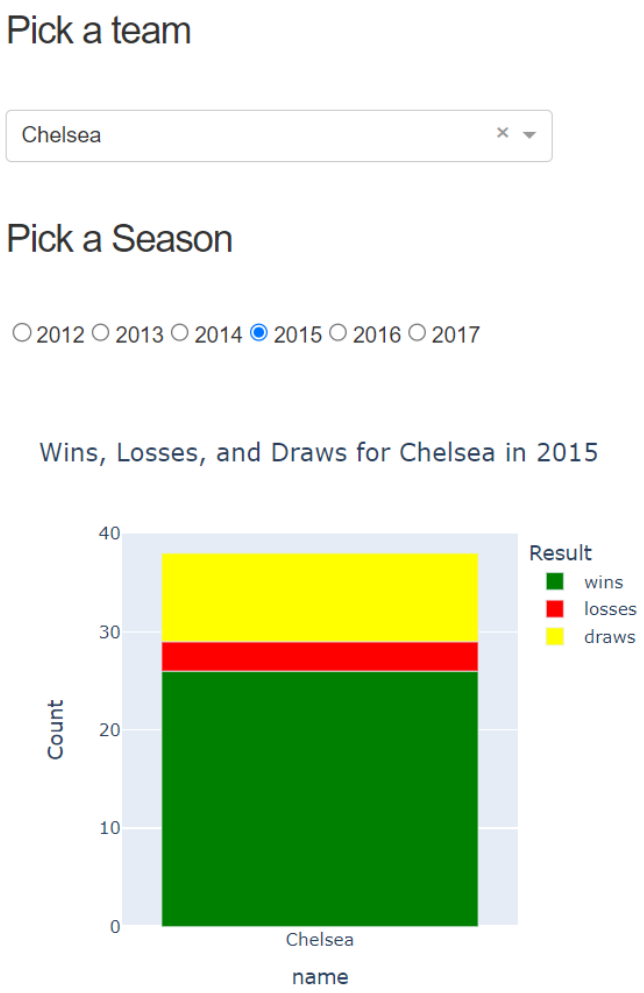


Fig12: Showing deployed app stacked bar graph

It Was part of my phase 2 visualization. This was to see how many wins, losses and draws different teams achieve during a particular season. Appropriate usage of radio buttons was done to ensure that users can view information of different teams for different years. The stacked bar plot was chosen here to indicate on a quick glance how often the team is winning as compared to them losing. Appropriate colors were used to indicate loss, draw and win.

4. Pie chart:

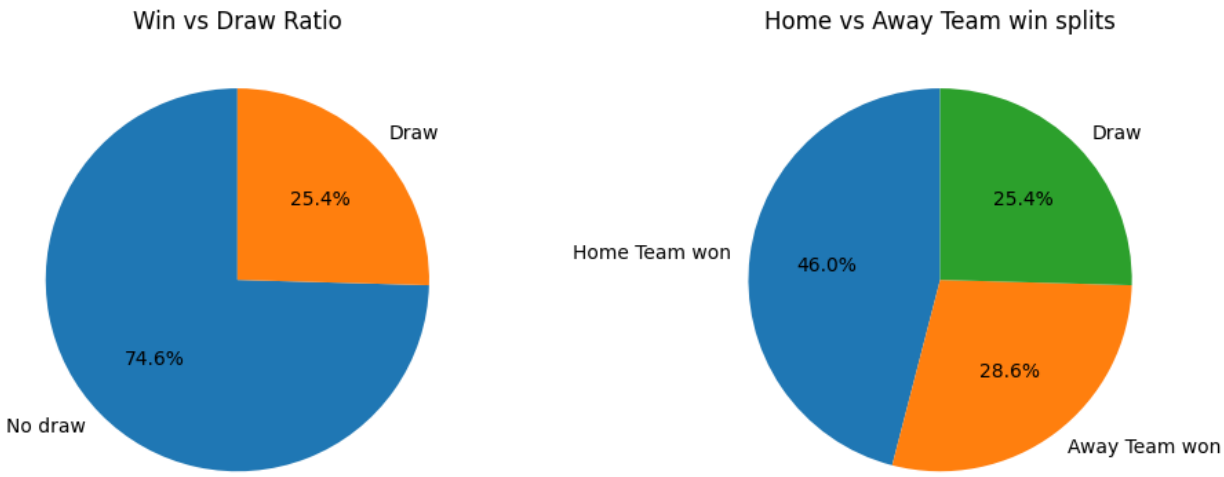


Fig13: Displaying pie chart for the win and draw ratio as well as home and away team splits

The pie chart gave an excellent visual representation. For instance approximately **75%** of the matches had a non-neutral outcome whereas 25% of the matches ended in a draw.

While talking about plot2, it was noted that Home Team won more often than Away teams with splits being **46 to 28** in favor of the home team. This observation helps conclude that **home ground advantage** is a real thing

5) Bar plot with counts

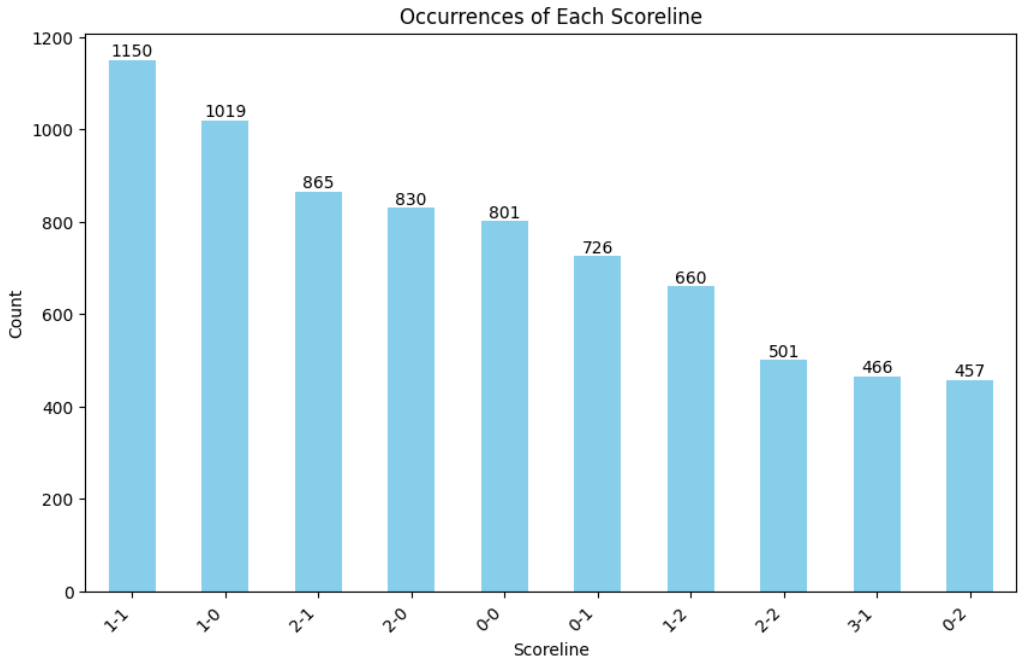


Fig14: Displaying the occurrences of the scorelines

As shown in the figure, the most common scoreline was **1-1**. This came in as rather surprising considering in the previous graph, we observed that only 25% of games ended in draw. Nevertheless, again it displays the home team advantage as **the first 4 plots indicate a positive result (win or draw) for the home team**.

6) Dist Plot

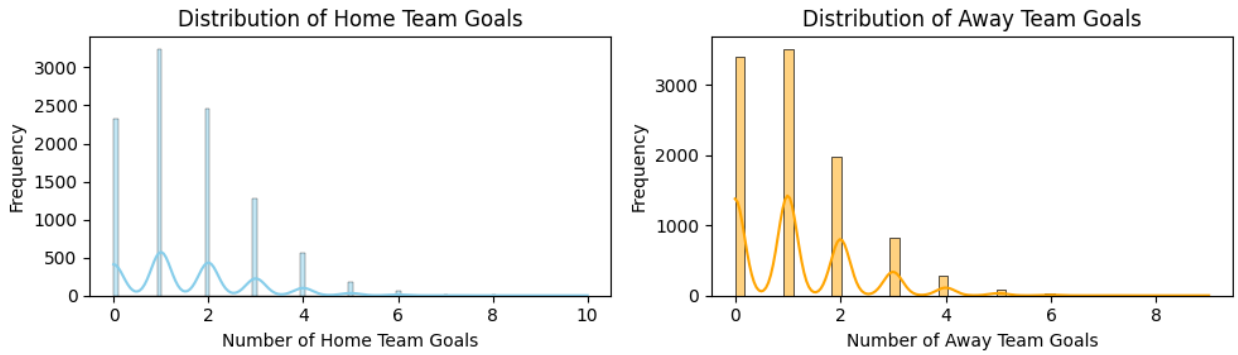


Fig15: Displaying the distribution of goals scored

As discussed in the Statistic section, the dist plot above shows the distribution of home team vs away team goals. As seen, there is a higher probability of the away team not scoring then the home team.

7) Pair plot:

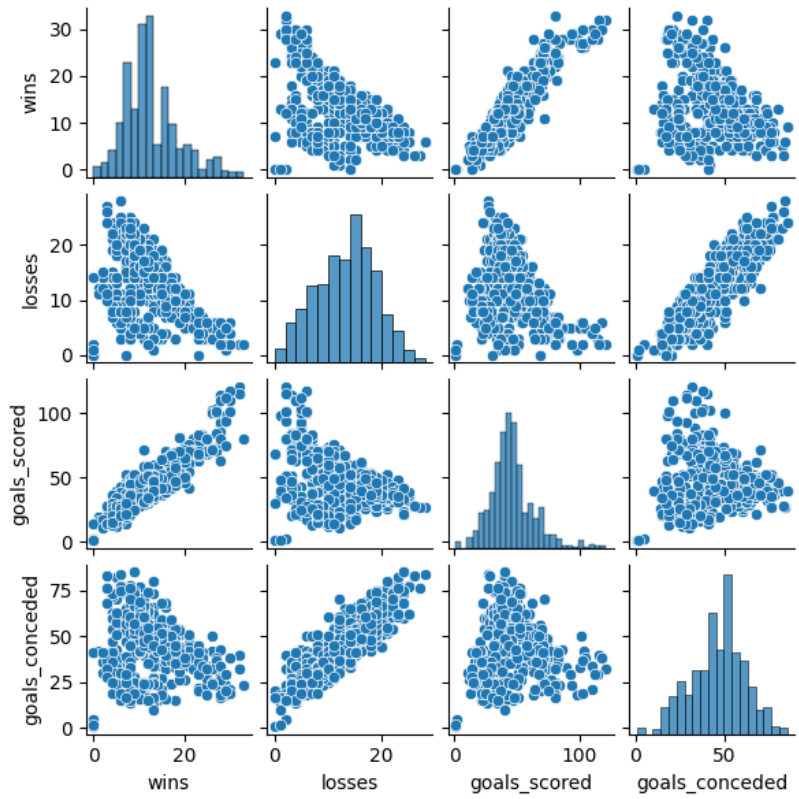


Fig16: Pair plot of team seasons dataframe

As evident from the graph, wins and goals_scored are directly correlated. Similarly, losses and goals_conceded are also strongly correlated. However, an interesting observation is wins and goals_conceded not showing a very high correlation which can be attributed to the fact that teams usually outscore more.

8) Heatmap with Cbar:

Fig17: Heatmap of goals conceded and scored by Real Madrid

For this example, the team Real Madrid were taken as a case study. As shown in the figure, Real Madrid are a consistently top performing team averaging more than **>100 goals** every season whereas conceding **<35 goals on average.** This helps in understanding why Madrid are such a great team.

9) Histogram plot with KDE:

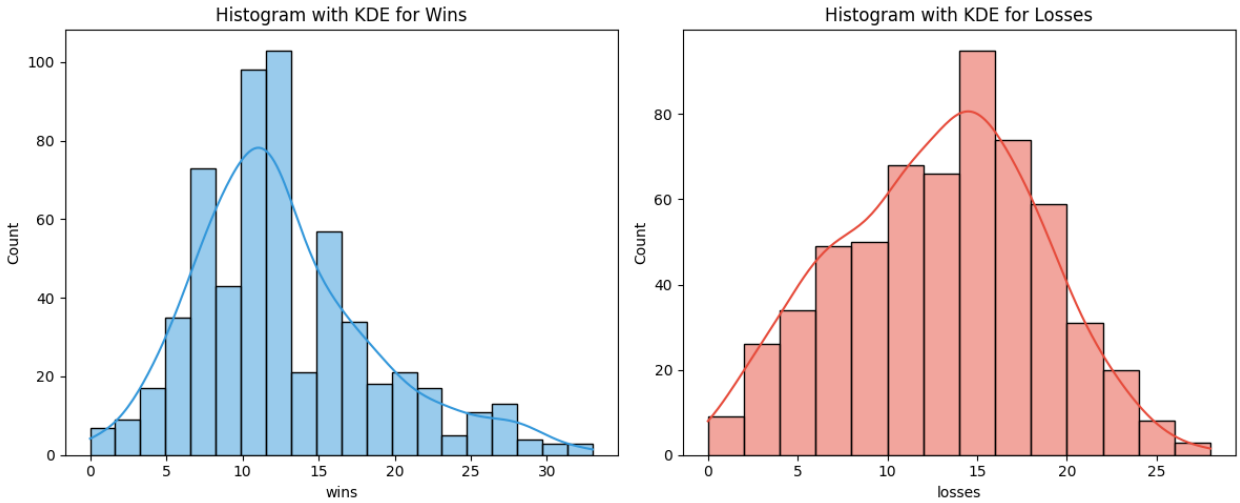


Fig18: Shows Histplot with KDE for wins and losses

As shown in the hist plot, it is clear that the mean **of losses is higher than the mean of wins.** This can be attributed to the fact that there are less teams in the leagues which are extremely great and the rest of the teams can be put into an average category. This also means that there are less good-teams than there are moderate teams.

10) QQ- plot, KDE plot and Reg plot:

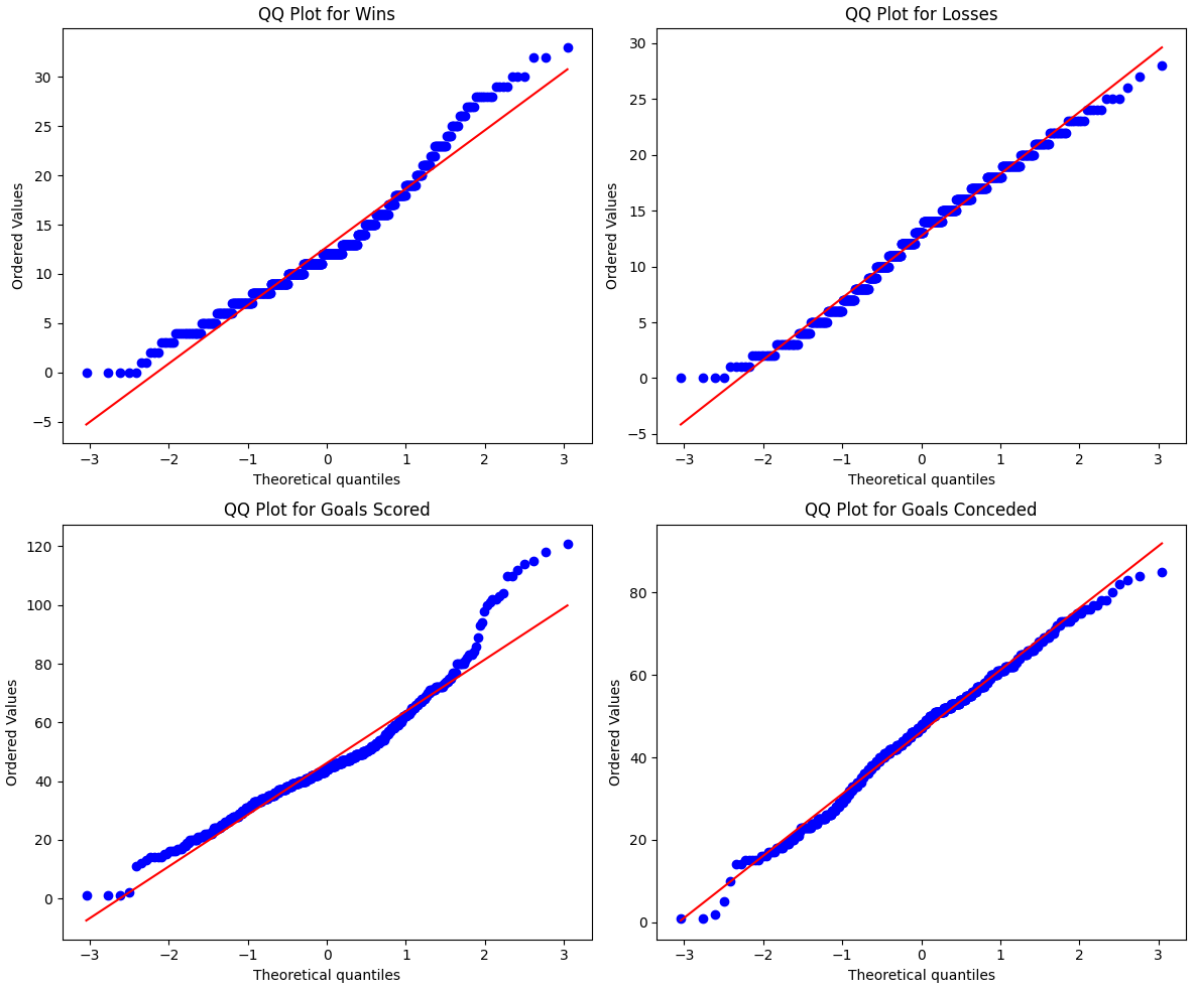


Fig19: Shows QQ Plot for wins, losses, goals scored and conceded.

QQ-Plot helps in understanding how normal the dataset is. The more the values deviate from the reference line, the less normal it is. As shown, except the outlier, the data points for each column seem to be normally distributed. A further proof of normality can be seen by the KDE plot of Goals Scored

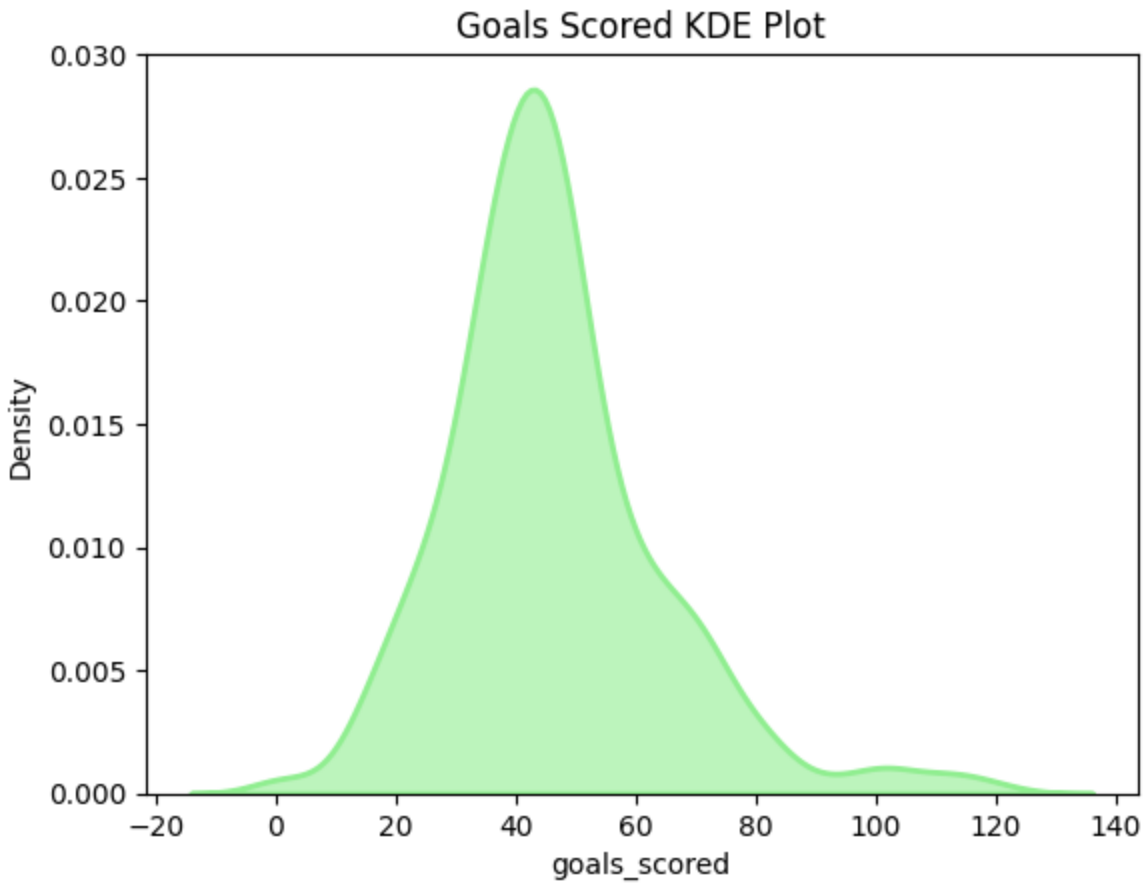


Fig20: KDE of Goals Scored

Apart from a little skewness on the right (**due to outliers like Real Madrid discussed earlier**) The data seems to be fairly normally distributed. Similarly the goals conceded showed a similar normality by the figure below.

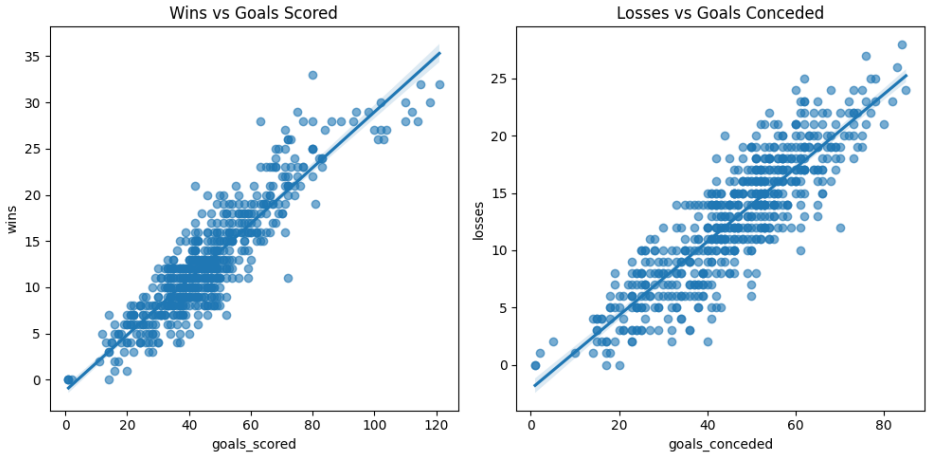


Fig21: Reg plot of goals scored and conceded

11) Area plot:

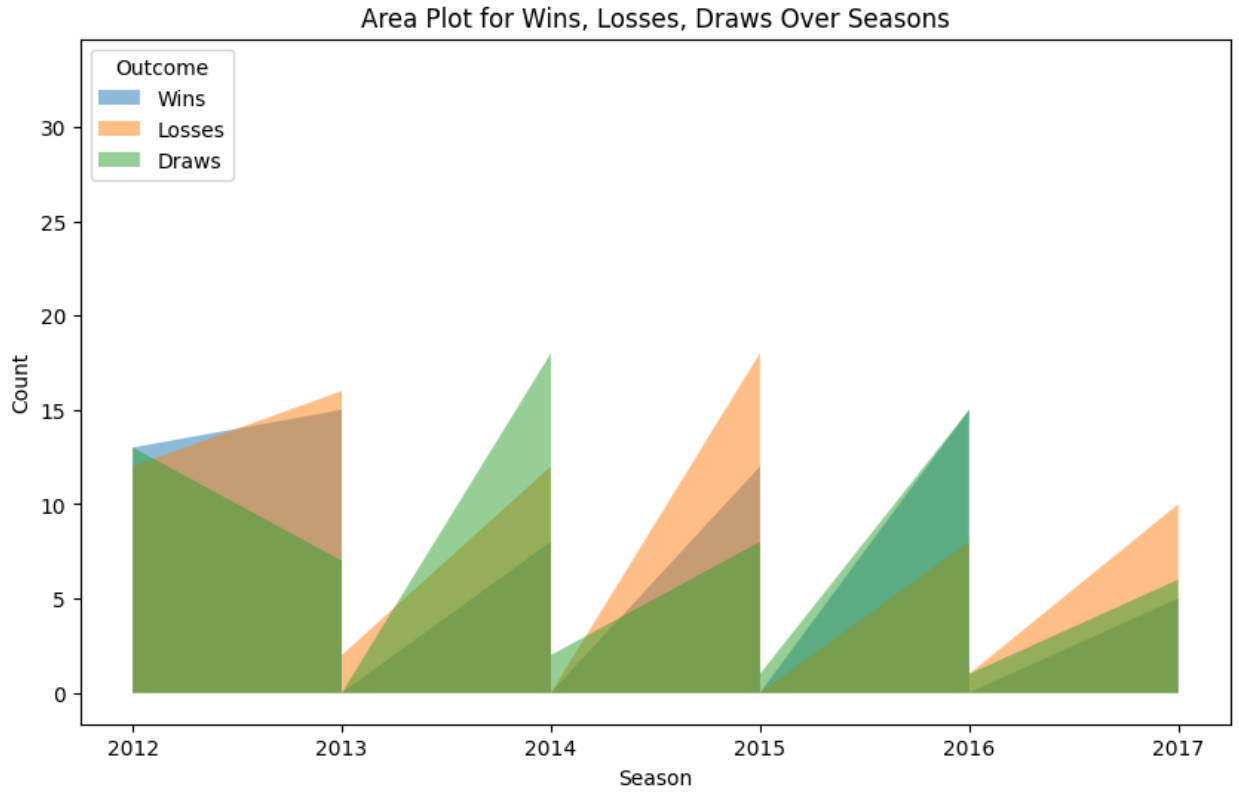


Fig22: Area plot of the match outcomes

This shows the distribution of wins and losses over the year. Interestingly, **2013** is the only year where Draws is taking a majority of the total games. In that sense, it can be considered an outlier year.

Another Area plot example was the plot of the top 6 strikers in the world and their respective goals scored. As seen from the figure, **Cristiano Ronaldo** and **Lionel Messi** remain untouched to the others whereas Luis Suarez comes close in 2016.

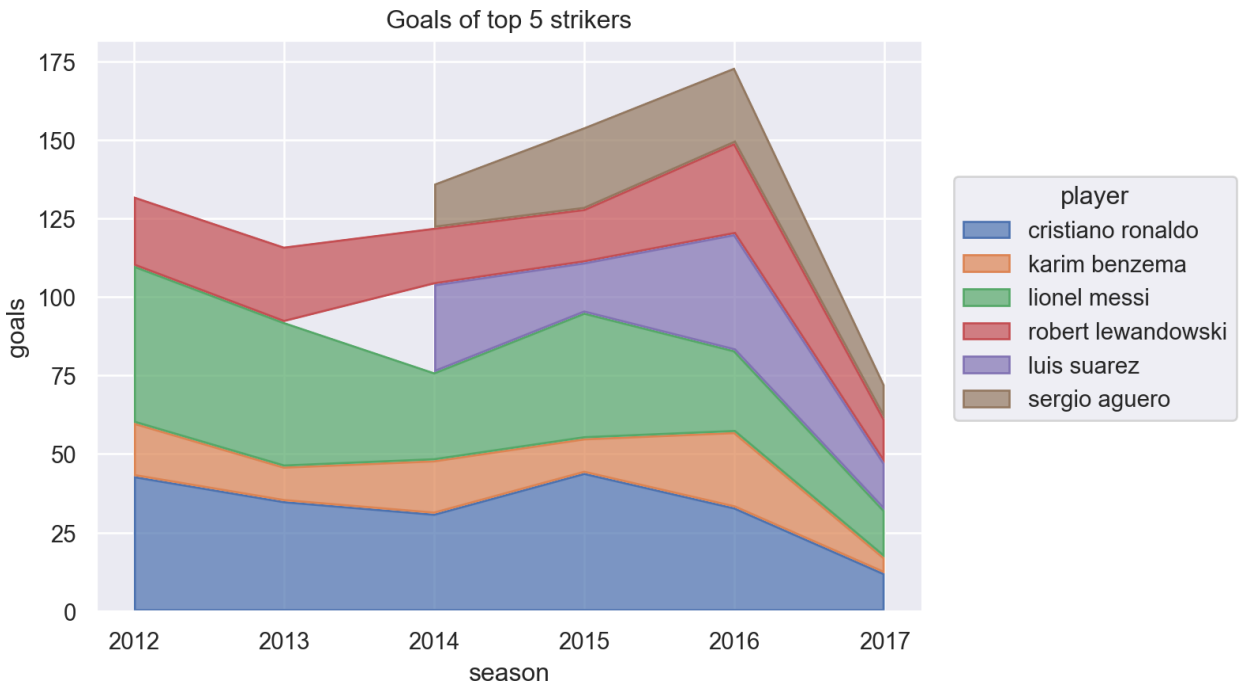


Fig23: Goals scored by top strikers per season

12) Violin Plot:

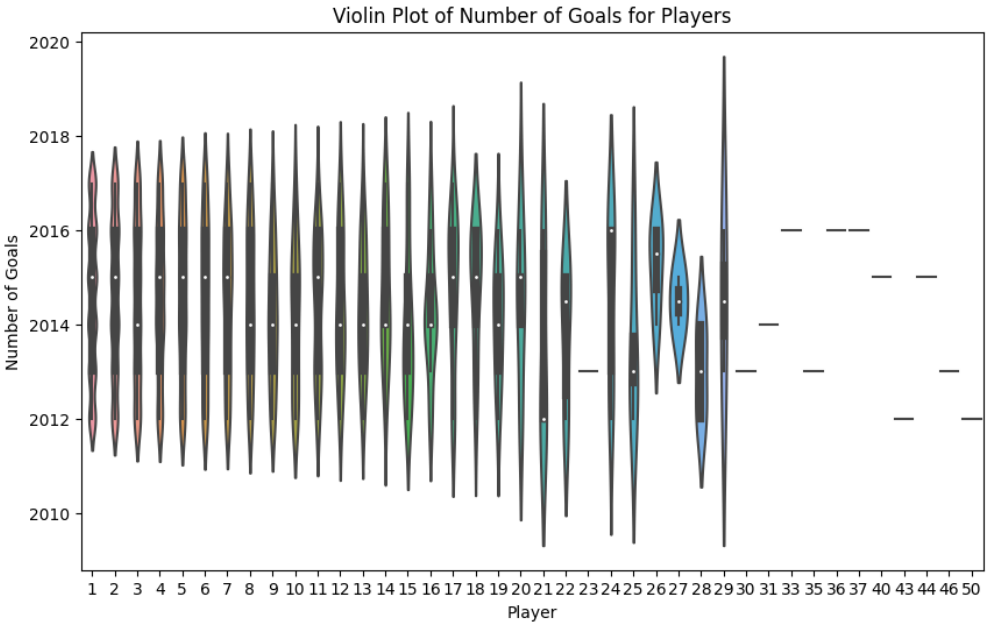


Fig24: Violin plot for the goals scored by players

As seen from the violin plot, most of the frequency of scores are distributed for **less than 22 goals**. Only a handful of players have managed to score more than 30 goals which is indicated by there not being any **violin like contour in region >30 goals.**

13) Joint plot and Scatter Representation:

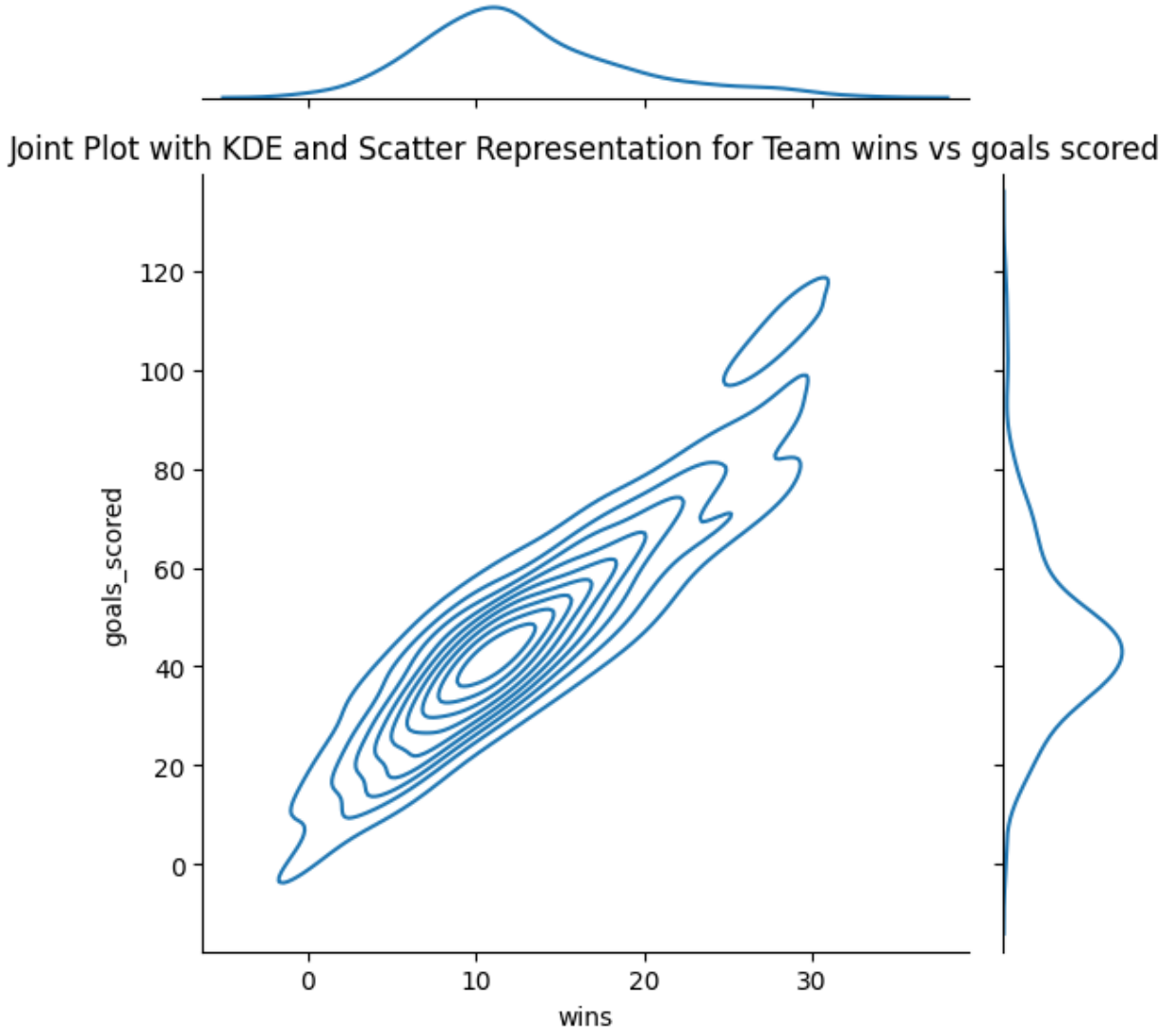


Fig25: Joint plot of goals_scored vs win

As discussed previously, goals scored and wins show a direct relationship and a strong one at that. The average wins are about **12** whereas average goals scored are about **50.**

14) Rug plot and Cluster map:

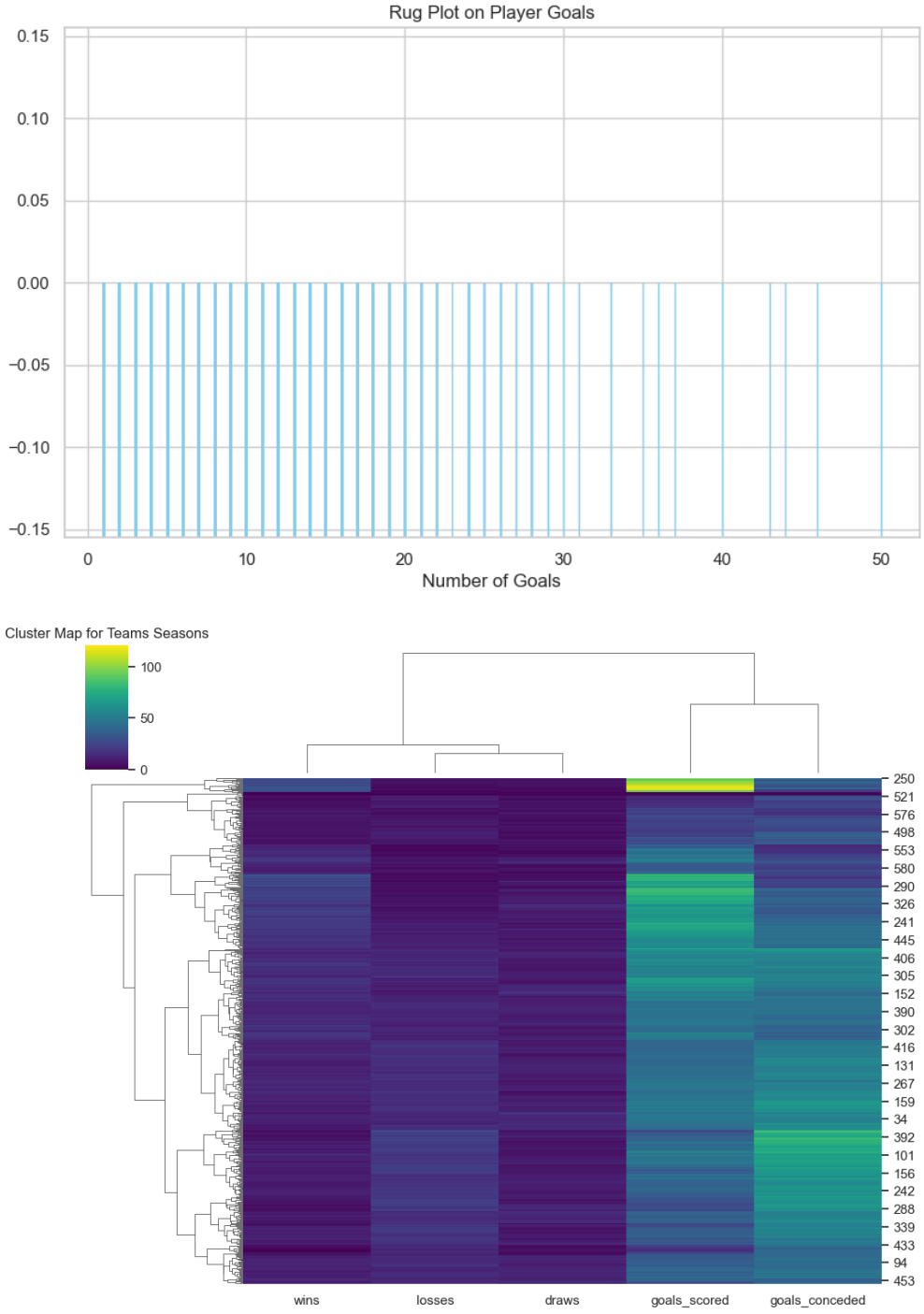


Fig26: Rug plot and Cluster map for goals and Team seasons

The Rug plot gives an estimate of the number of goals scored by players. The outliers are clearly visible like Messi with 50 goals and Ronaldo with 48 goals. Whereas in the Cluster map, clusters of data points are shown which shows the probability of the range of values occurring together.

15) Hexbin plot:

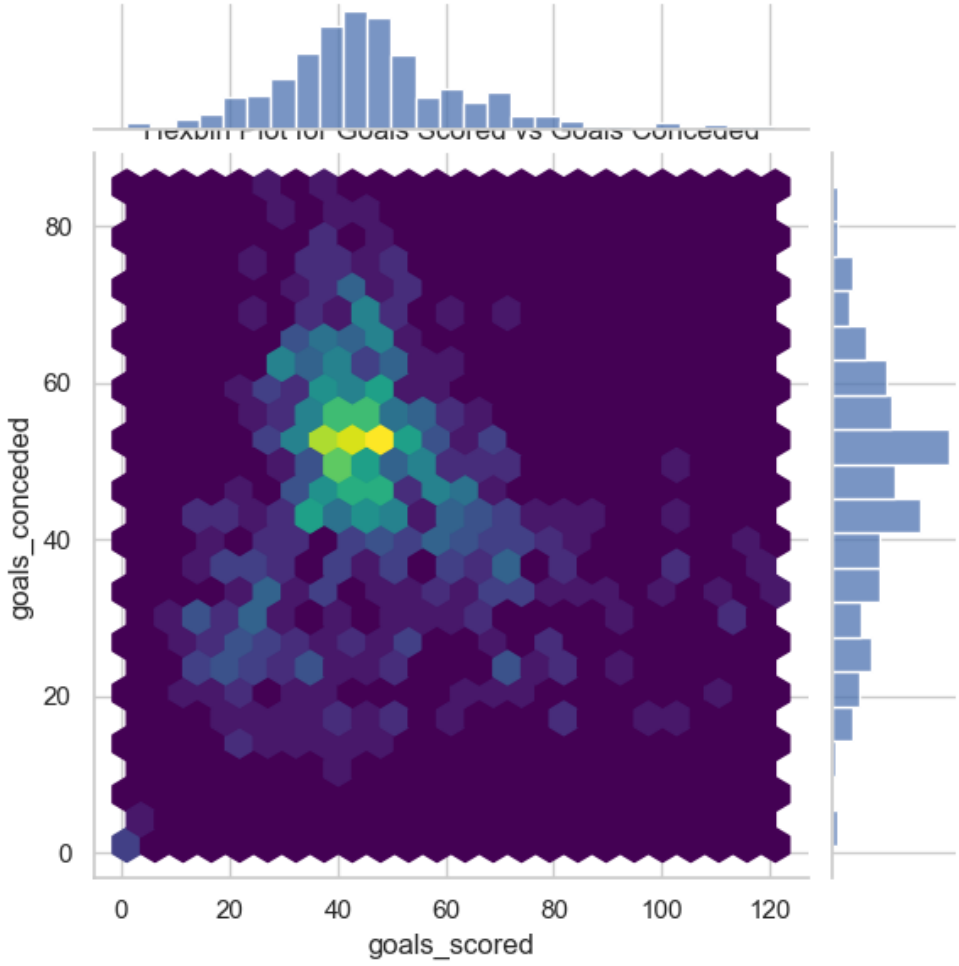


Fig27: Hexbin of goals scored vs conceded

As shown, there is not a strong correlation between goals scored and goals conceded. The most common occurrence for both lie in the range of 50 goals.

16) Strip plot

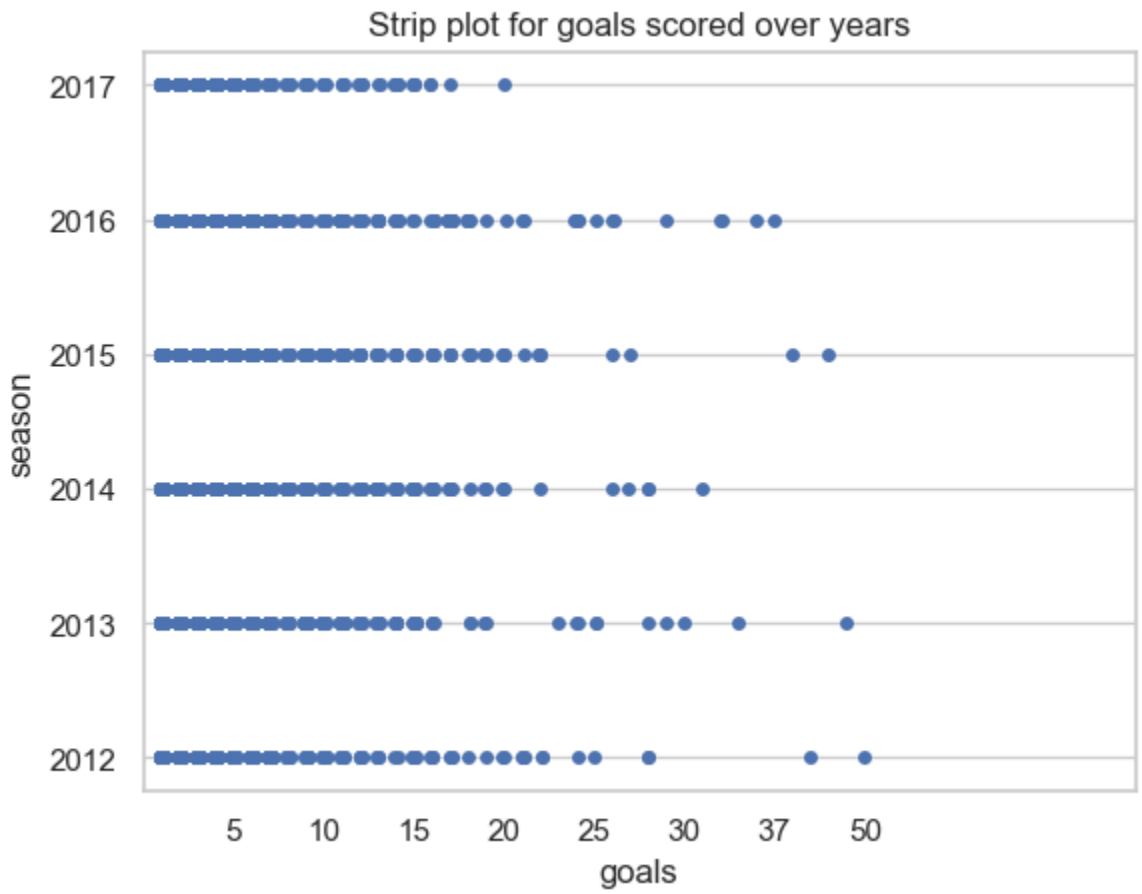


Fig28: Strip plot of goals scored by players across different years

As shown in the figure, the range of score is very less in 2017 because the number of games are very less. After that, 2014 has the least variation as **31 was the highest goals that season** which were scored by **Luis Suarez** in his final Liverpool season.

17) Swarm Plot:

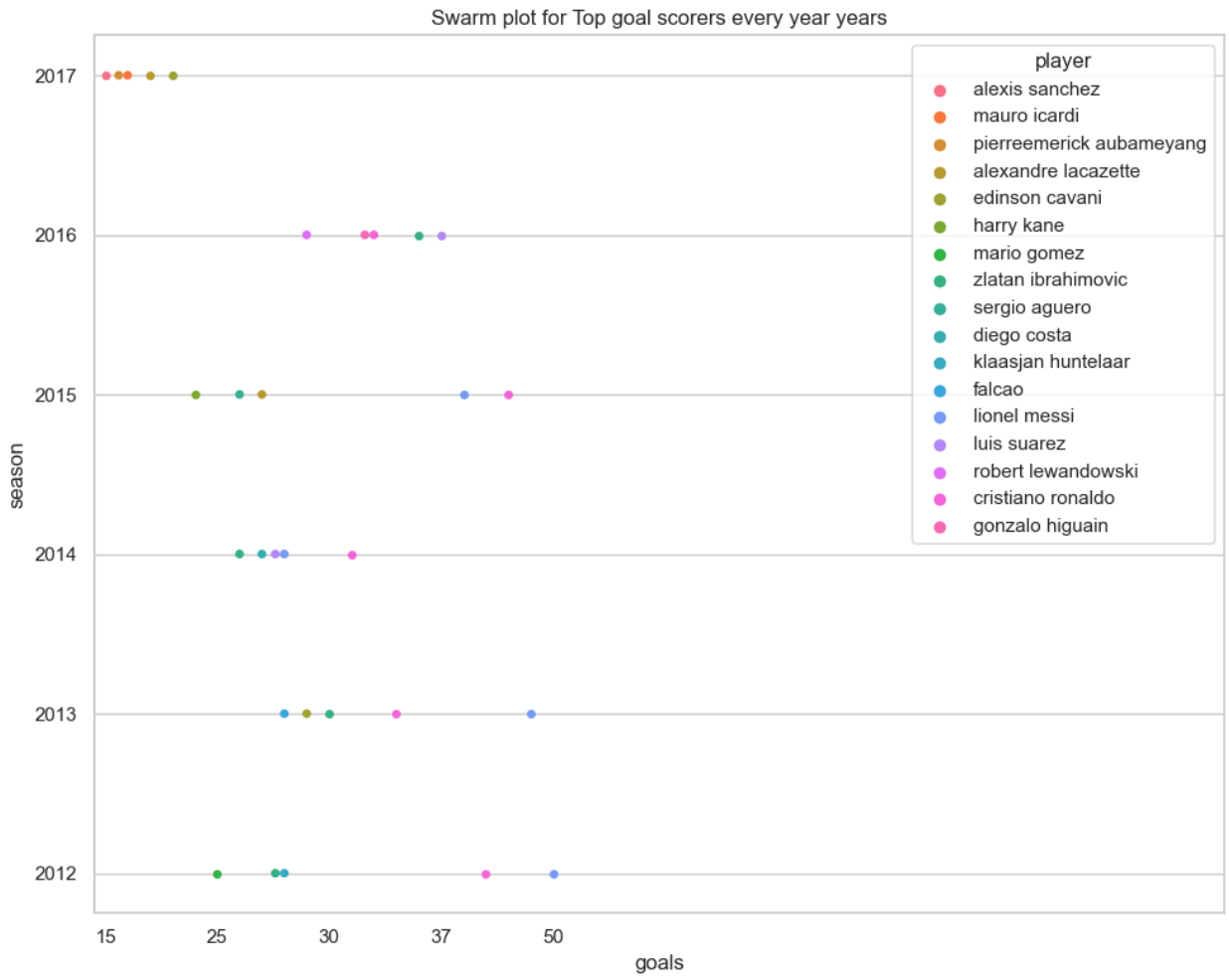


Fig29: Shows the swarm plot of top 5 scorers every year

This plot shows the top **5 scorers** for every year. This graph is particularly interesting as it shows the dominance of Mess and Ronaldo as they are visible every year.

Phase 2 Plots:

1) Pie and Bar plot:

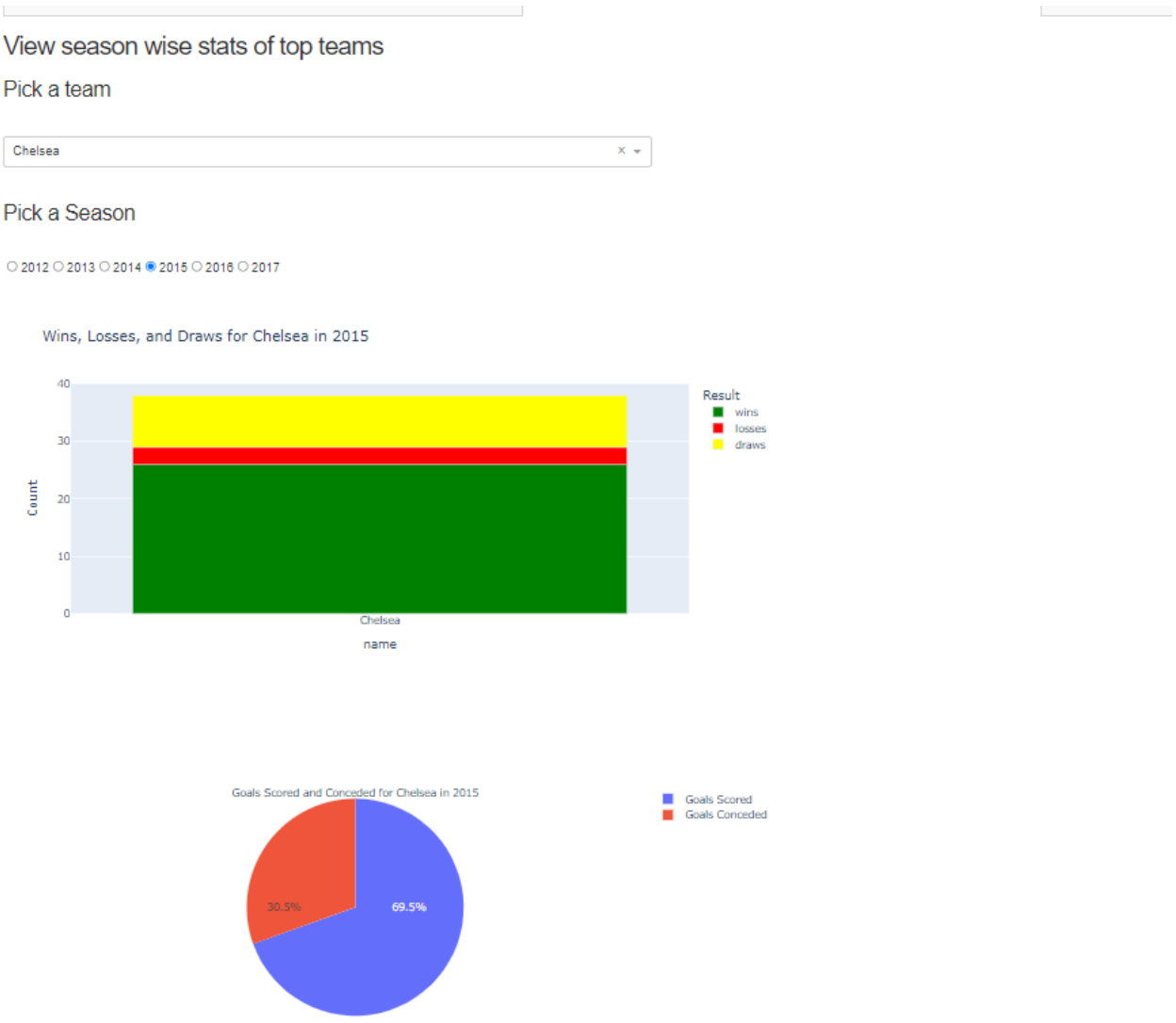


Fig30: Shows the phase 2 graph plot

It shows the bar and pie chart for the team stats for the particular year. As seen, dropdown is implemented along with use of Radio buttons

2) Line plot for player stats:

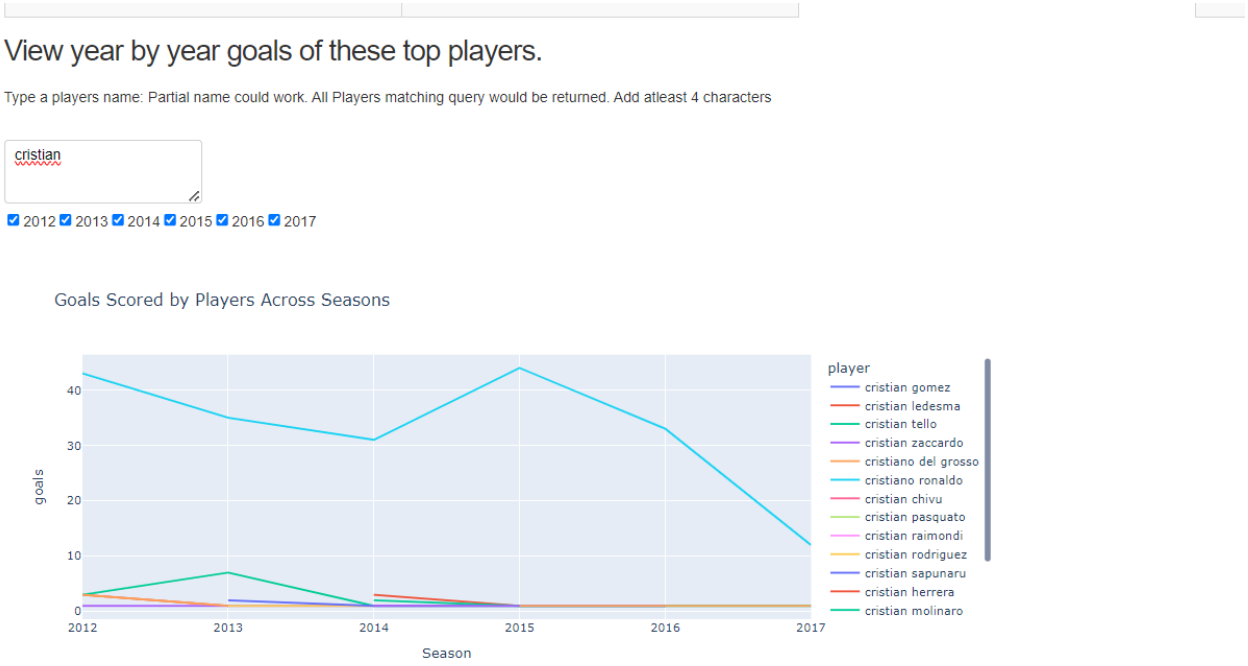
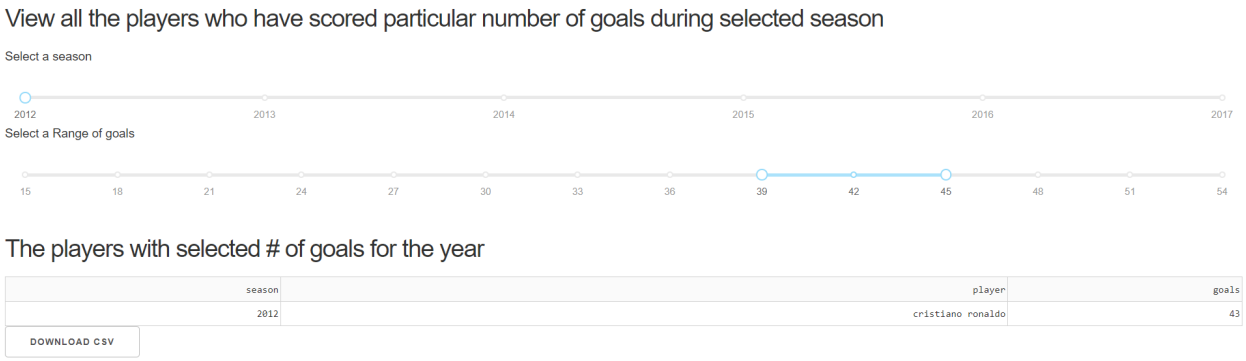


Fig31: Shows the line plot for player information

This uses search and checkbox functionality. In this, you can add prefix of the player name and all the players matching that would be displayed in the graph.

3) Data Table:



A Data table with the download functionality was also provided which helps in filtering according to the year and goal range. Sliders and Download functionality are implemented and loading is also present.

Conclusion

In conclusion, our Information Visualization Project has successfully transformed raw football event data into a visually compelling narrative, offering users an immersive and insightful exploration of the dynamics within the sport. The amalgamation of advanced visualization techniques, geospatial maps, timelines, and player-centric metrics has not only streamlined the analysis of football events but has also provided a user-friendly platform for enthusiasts and analysts to engage with the intricacies of the game.

Through this project, we have addressed the challenge of making complex data accessible, fostering a deeper understanding of football analytics. The project's interactive features empower users to unravel hidden patterns, observe trends, and relive key moments in a match, creating a bridge between statistical insights and the visceral experience of the sport.

As we reflect on the journey of this project, it becomes evident that information visualization serves as a powerful tool not only for data analysis but also for storytelling within the context of sports analytics. The success of this project lies not just in its technical achievements but in its ability to enhance the appreciation and comprehension of football events.

Looking ahead, the visualization platform created through this project stands as a testament to the potential of merging data science with sports analytics, offering a blueprint for future projects aimed at unraveling the intricacies of various domains. As the field of information visualization continues to evolve, our project stands as a contribution to the ongoing dialogue, inviting users to delve deeper into the beautiful game and sparking curiosity in the broader landscape of data exploration.

In essence, this project represents a step forward in leveraging data visualization to make the beautiful game even more captivating, bridging the gap between data and narrative, and offering a window into the nuanced world of football analytics.

Appendix:

1) Supporting Python Code: (utility.py). Other Submission files are attached in the Submission

```
# making a class to deal with team seasons easily
class TeamSeason:
    def __init__(self, name,season, wins=0, losses=0, draws=0,
goalsScored=0, goalsConceded=0):
        # Instance variables
        self.name = name
        self.season = season
        self.wins = wins
        self.losses = losses
        self.draws = draws
        self.goalsScored = goalsScored
        self.goalsConceded = goalsConceded

    def addWin(self):
        self.wins+=1

    def addLoss(self):
        self.losses+=1

    def addDraw(self):
        self.draws+=1

    def getTotalGames(self):
        return self.wins + self.losses + self.draws

    def addGoalsScored(self,n):
        self.goalsScored+=n

    def addGoalsConceded(self,n):
        self.goalsConceded+=n

    def export(self):
        return {
```

```
            'name':self.name,
            'season': self.season,
            'wins': self.wins,
            'losses': self.losses,
            'draws': self.draws,
            'goals_scored': self.goalsScored,
            'goals_conceded': self.goalsConceded
        }
```