Sports Analytics (Soccer)

Team Members:

19BDS0098 Lohit Jayaprakash

19BDS0125 Nikhitha Perapola

19BDS0109 Atluri Bhumika

Report submitted for the

Project Review of

Course Code: CSE3045

Predictive Analysis

Slot: A1

Professor: Dr. Ilanthenral Kandasamy

1. Introduction:

Football Analytics is a rapidly changing field, with the increase in computational power, parallel processing, advancements in computer vision the range of soccer analytics has boomed exponentially. The most important part of analytics is data and along with other advancements data has had the most important role in the advancements of this field. Two decades ago, only basic stats like goal count and possession percentage were around and not much information was juiced out of such trivial data.

The next wave of football analytics came into picture when Event Data had come up. It is basically the record of each event which had happened during a game and provides an extra dimension to gaining insights for applying it into coaching. This kind of data helps us to analyze much more advanced metrics like passing models, possession chains. But this kind of data still has limitations, as mentioned by FC Barcelona lead data analyst the data provides us the activities which happened during the match but for coaching purposes, context is very important when analyzing what a player did in a certain event. In current times, tracking data, which is currently the epitome of football data, is the collection of all the positions of the 22 players and the ball which is taken many times every second.

This helps us find tactical insights for the coach. This kind of data helps us find the context, like how a player with less ball contact can still help the team drastically by running into tactical positions and other tactics used in football. In this project we are dwelling in to soccer analytics and understanding and presenting what are the kinds of different data which can be found, and also certain metrics used in t football and how it can help gain insights and help a certain team perform better. We will be using different datasets and see how they compare using EDA and understand what value they provide.

2. Literature Review Summary Table:

Authors and Year (Reference)	*Title (Study)*	*Concept / Theoretical model/ Framework*	*Methodology used/ Implementation*	*Dataset details/ Analysis*	*Relevant Finding*	*Limitations/ Future Research/ Gaps identified*
Tom Decroos, Jesse Davis February 2020	Interpretable Prediction of Goals in Soccer	Three feature sets were trained using logistic regression, XGBoost and generalized additive model classifiers. .	The following datasets have been trained using the three classifiers and the and were evaluated using the Normalized Brier Score **The location only** feature set considers only the (x, y)-coordinates of the last action in a game state S. **VAEP** This considers the set of 151 features used in the original VAEP model. **Top-10** This considers the 10 most important features from VAEP feature set	Data set consists of event stream data of 760 matches from seasons 2017/18 and 2018/19 of the English Premier League.Dataset has been taken from StatsBomb.The data is divided into **Simple features** which consists of type, player, team, result, bodypart, time, start location, and end location. **Complex Features** include the distance and angle to the goal for both the action's start and end locations, and the distance covered during the action in both the x and y directions. **Game context features** the number of goals scored in the game by the team possessing the ball after action , the number of goals scored in the game by the defending team after action , and the goal difference after action. Data has a total of 151 features.	1)Only considering location is insufficient. 2)Having a model that captures non-linearities helps. 3)A small feature set can yield excellent performance.	XGBoost offers no intuitive explanations on why a given game state produced a higher or lower chance of scoring
Sunil Srinivas Sukumar 2019	Moneyball or Moneyfall? Current State of Analytics in Soccer and What Future Holds	Regression and correlation to understand the significance of factors.	multiple regression models would be used to test the significance of these variables at a smaller scale with the data mentioned	**Data**:Secondary(Collected from an already existing source) Type : **Source**: Worldfootball.net **Volume**: 2017-2018 Season of english premier league The dependent variable is the outcome of the games. Independent variables include, number of goals scored home/away,player attributes,goals conceded,clean sheets etc.	Quality, current form of the teams are important factors in the analytics models • Player fitness, possession, playing sequence & style also contribute to the models	predictive analytics models in soccer can be made better with the help of descriptive models that help understand the data well.

Wuhuan Deng* , Eric Zhong 2020	Analysis and Prediction of Soccer Games: An Application to the Kaggle European Soccer Database	Numbers of goals for both away team and home team in each of the matches are collected in order to study the distribution of goals. Two Poisson distribution models are built to fit the goal distribution. As a result, because the two variables, home goal and away goal, have Poisson distribution s, the difference between them, or net goal, should have a Skellam distribution.	The prediction in this work is to foretell the match results based on pre-game bet radios and in-game performance. Four models are built: Logistic Regression, Decision Tree, Random Forest and Deep Neural Network.	Open dataset for study in this work is acquired from www.kaggle.com, built by Hugo Mathien.soccer match data of 11 European countries with their lead championship from seasons 2008 to 2016. The database covers 24,637 matches and over 10,000 players. Detailed match events like goal types, possession, corner, cross, fouls, cards are also recorded in the database. The dataset is a SQL database file contains 7 tables of “Country”, “League”, “Match”, “Player”, “Player_Attribute”, “Team”, “Team_Attributes”. For the match table, 134 attributes are recorded.	the Deep Neural Network model gives the highest accuracy of 0.99. The Logistic Regression, Decision Tree and Random Forest models results in 0.95, 0.91 and 0.84. 4 most important features including possession, total shot, shot efficiency and goal efficiency are compared. Each of the 4 features is solely applied for the prediction for the 5 leagues.the best offensive and defensive teams are successfully evaluated, and the Poisson and Skellam Distribution is verified for fitting the goals. The results show that our feature engineering and designed Neural Network are successful in achieving the desired match result.	-

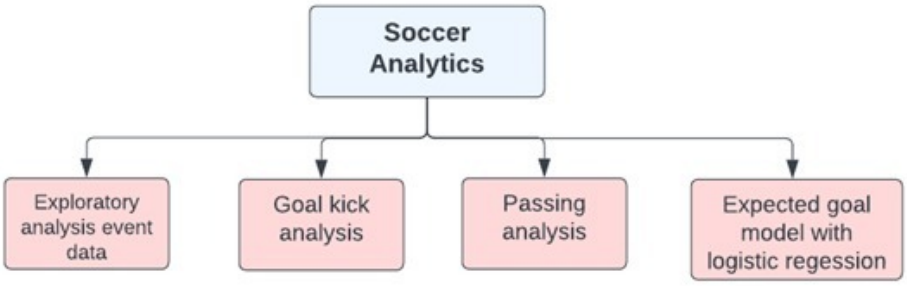
Enes Eryarsoy and Dursun Delen 2019	Predicting the Outcome of a Football Game: A Comparative Analysis of Single and Ensemble Analytics Methods	Compared their performances against each other: Naïve Bayes, Decision Trees, and Ensemble models.	Win/Loss/Draw Prediction: The game could easily end with any of the three outcomes. However, as it the middle ground between "win" and "loss, "draw" intuitively is the most difficult to predict. They also performed missing value imputation (single imputation, using mean and mode) to be able to run Neural Networks (2 hidden-layers, 10 nodes per layer), SVMs (with RBF), and kNN (k=5) Points/NoPoint Prediction: This makes the prediction task easier by reducing the possible number of outcomes to two: "Points" and "No Points". This prediction task is also relevant for soccer bettingthe performance difference between Gradient Boosting and Random Forest ensemble methods was not found to be statistically different at 0.05 alpha level for either of the "Win/Loss/Draw" and "Points/NoPoint" problems (α=0.11.2, and α=0.12 respectively).	The sample data for our study are collected from the Turkish Super League, using a variety of sources and means, including hand collection. The initial dataset included 3,060 game-level items of data, from 33 teams spanning a complete 10 seasons (2007-2017). The dataset has 50 variables and their description is mentioned clearly. The data is properly preprocessed and the missing values are handled. In exception, because of the non-random nature of the missing values in several of the critical independent variables, we could neither impute the missing values nor we could exclude them from the dataset.	The prediction results of the modeling techniques suggest ensemble tree models performed significantly better. However, the prediction accuracy for the "draw" class is proven to be more difficult and 2 visualizes this phenomenon using Gradient Boosted Trees Regression results. They were able to achieve over 74% accuracy in "Win/Loss/Draw", and over 86% accuracy in "Points/NoPoint" type of classification problems.	From this paper we can understand that a more carefully collected dataset (i.e. with fewer missing values), as well as a richer dataset in terms of variables, will help in predicting outcomes more accurately.

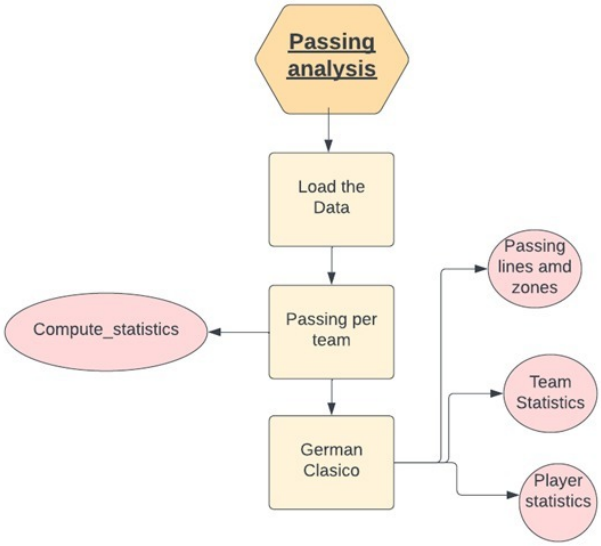
3. Objective of the project: We aim to take multiple datasets providing different kinds of data (match data, player data, event data, tracking data) and perform EDA to understand what insights each kind provides.

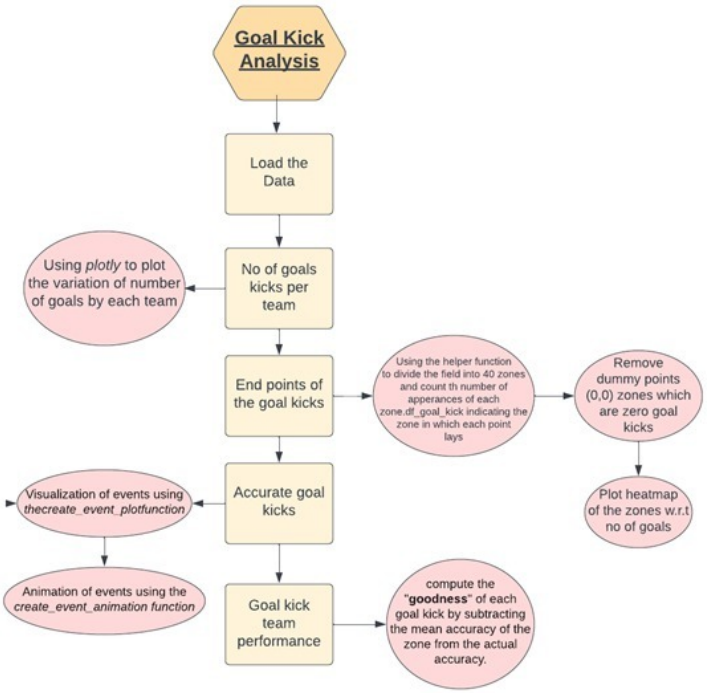
Next step is to understand and implement certain performance metrics in soccer like goal kick analysis, passing analysis and passing probability which is relatively close to current research in football analytics

4. Innovation component in the project:

Analyzing goal play and passing play. Finding a suitable model and implementing.







5. Implementation:

a. Methodology adapted:

b. Dataset used:

The dataset utilized is Wyscout (https://wyscout.com/) largest open collection of soccer-logs ever released, which contains all the spatio-temporal events (passes, shoots, fouls, etc.) that occur throughout all matches of a full season of seven competitions (La Liga, Serie A, Bundesliga, Premier League, Ligue 1, FIFA World Cup 2018, UEFA Euro Cup 2016). The position, time, outcome, player, and features of a match event are all recorded in this dataset.

The preprocessed data is sourced from a Github repository.

We will be dealing with a total of five different data sources.

→ Player data

→ Team data

→ Match data

→ Formation data

→ Event data

c. Tools to be used:

Python:

Panda Profiling , libs: pandas for data pipeline, scikit learn for modeling, matplotlib for basic visualization

d. Screenshot and Demo along with Visualization (Preprocessing:

```
def get_team_view(matches, team):

    opp_team = 1 if team == 0 else 0

    # save relevant information in data frame
    df_team_view = pd.DataFrame()

    df_team_view["matchId"] = [match["wyId"] for match in matches]
    df_team_view["gameweek"] = [match["gameweek"] for match in matches]
    df_team_view["dateutc"] = [match["dateutc"] for match in matches]

    # get the 2 teams with their side (home / away) and score
    df_team_view["teamId"] = [
        match["teamsData"][list(match["teamsData"])[team]]["teamId"]
        for match in matches
    ]
    df_team_view["side"] = [
        match["teamsData"][list(match["teamsData"])[team]]["side"] for match in matches
    ]
    df_team_view["score"] = [
        match["teamsData"][list(match["teamsData"])[team]]["score"] for match in matches
    ]
    df_team_view["oppTeamId"] = [
        match["teamsData"][list(match["teamsData"])[opp_team]]["teamId"]
        for match in matches
```

```
    df_team_view["oppTeamId"] = [
        match["teamsData"][list(match["teamsData"])[opp_team]]["teamId"]
        for match in matches
    ]
    df_team_view["oppScore"] = [
        match["teamsData"][list(match["teamsData"])[opp_team]]["score"]
        for match in matches
    ]

    return df_team_view


def get_all_formations(matches):
    """
    Function to compute a data frame with the formation for all the matches in *matches*
    :param matches: (dict) Dictionary containing all matches and the formations
    :return: pd.DataFrame containing all formations of all matches
    """

    lst_formations = list()
    for match in matches:

        match_id = match["wyId"]

        # loop through the two teams
        for team in [0, 1]:
            team = match["teamsData"][list(match["teamsData"])[team]]
            team_id = team["teamId"]

            # get all players that started on the bench
            player_bench = [player["playerId"] for player in team["formation"]["bench"]]
            df_bench = pd.DataFrame()
            df_bench["playerId"] = player_bench
            df_bench["lineup"] = 0
```

```python
def cleanse_wyscout_match_data(country):
    """
    Function to cleanse the wyscout match data and save it in the data folder
    :param country: (str) Country for which the event data should be cleansed
    :return: None
    """

    logging.info(f"Cleansing wyscout match data for {country}")

    # read the JSON file with matches
    matches = io.read_data("match_data", league=country, data_folder="raw_data_wyscout")

    # save relevant information in data frame
    df_matches = pd.concat(
        [get_team_view(matches, 0), get_team_view(matches, 1)], axis=0
    )

    # attach the points per team
    df_matches["points"] = np.where(
        df_matches["score"] > df_matches["oppScore"],
        3,
        np.where(df_matches["score"] == df_matches["oppScore"], 1, 0),
    )

    df_matches["dateutc"] = pd.to_datetime(df_matches["dateutc"])

    df_matches["scoreDiff"] = df_matches["score"] - df_matches["oppScore"]

    df_matches.sort_values(["matchId", "side"], ascending=[True, False], inplace=True)
    io.write_data(df_matches, "match_data", league=country.lower())

    df_formations = get_all_formations(matches)
    io.write_data(df_formations, "formation_data", league=country.lower())
```

e. Models used :

Logistic Regression

f. Screenshot and Demo along with Visualization:

Includes results with visualization

Player data :

Taking a look at a sample of the player dataset :

```
df_players.head()
```
[35] ✓ 0.3s

	playerId	playerStrongFoot	playerName	playerPosition
0	32777	right	H. Tekin	GK
1	393228	left	M. Sarr	DF
2	393230	unknown	O. Mandanda	GK
3	32793	right	A. N'Diaye	MD
4	393247	right	I. Konaté	DF

Checking for duplicate values :

```
print(f"Number of unique playerIds: {len(df_players['playerId'].unique())}")
```
[37] ✓ 0.2s

```
Number of unique playerIds: 3603
```

Finding out the number of players for each position :

```
df_players.groupby("playerPosition").size().sort_values()
```
[38] ✓ 0.2s

```
playerPosition
GK     426
FW     720
DF    1200
MD    1257
dtype: int64
```

Using Panda profiling to attain the following details about the datasets to analyze more effectively :

- Uniqueness

- Missing values

- Histograms

- Min / max / mean

- Column type

Overview | Reproduction | Warnings 1

Dataset statistics

Number of variables	4
Number of observations	3603
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	708.2 KiB
Average record size in memory	201.3 B

Variable types

CAT	3
NUM	1

Team data :

Taking a look at a sample of the team dataset :

```
df_teams
```
[41] ✓ 0.1s

	position	teamId	teamName	matches	goals	concededGoals	goalsDiff	points
4	1	2444	Bayern München	34	92	28	64	84
6	2	2449	Schalke 04	34	53	37	16	63
12	3	2482	Hoffenheim	34	66	48	18	55
3	4	2447	Borussia Dortmund	34	64	47	17	55
2	5	2446	Bayer Leverkusen	34	58	44	14	55
17	6	2975	RB Leipzig	34	57	53	4	53
5	7	2445	Stuttgart	34	36	36	0	51
15	8	2462	Eintracht Frankfurt	34	45	45	0	49

Match data :

Taking a look at a sample of the team dataset :

```
df_matches.head()
```
[43] ✓ 0.2s Python

	matchId	gameweek	dateutc	teamId	side	score	oppTeamId	oppScore	points	scoreDiff
305	2516739	1	2017-08-18 18:30:00	2444	home	3	2446	1	3	2
305	2516739	1	2017-08-18 18:30:00	2446	away	1	2444	3	0	-2
300	2516740	1	2017-08-19 13:30:00	2482	home	1	2443	0	3	1
300	2516740	1	2017-08-19 13:30:00	2443	away	0	2482	1	0	-1
301	2516741	1	2017-08-19 13:30:00	2457	home	2	2445	0	3	2

The first matchId appears in both , the first and the second row , this is beacuse the match is split into two perspectives , the home team adnd away team.

Considering the first match:

1. In the first line it says that teamId 2444 played at home against teamId 2446, scored 3 goals itself and the opponent scored 1 goal, e.g.2444 won 3-1 against 2446. This is teamId 2444's perspective.

2. In the second line it says that teamId 2446 played away against teamId 2444, scored 1 goal itself and the opponent scored 3 goals, e.g. teamId 2446 lost 1-3 at 2444. This is teamId 2446's perspective.

Running Panda's profilling :

Overview | Reproduction | Warnings 3

Dataset statistics

Number of variables	11
Number of observations	612
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	82.0 KiB
Average record size in memory	137.2 B

Variable types

NUM	8
CAT	2
DATE	1

Formation data :

Taking a look at a sample of the team dataset :

```
df_formations.head()
```

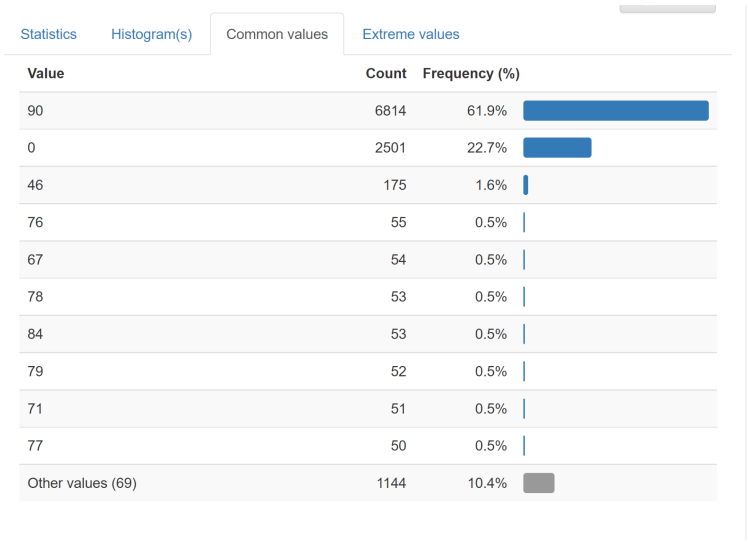
[47] ✓ 0.1s Python

	playerId	lineup	matchId	teamId	substituteIn	substituteOut	minuteStart	minuteEnd	minutesPlayed
0	209091	1	2517036	2444	0	1	0.0	46.0	46.0
1	14732	1	2517036	2444	0	1	0.0	68.0	68.0
2	14726	1	2517036	2444	0	1	0.0	71.0	71.0
3	14736	1	2517036	2444	0	0	0.0	90.0	90.0
4	14817	1	2517036	2444	0	0	0.0	90.0	90.0

There are 4 different groups of players:

1. Those that started the match (*lineup* = 1) and did not get substituted (*substituteOut* = 0)
2. Those that started the match (*lineup* = 1) and did get substituted (*substituteOut* = 1)
3. Those that started on the bench (*lineup* = 0) but did get substituted in (*substituteIn* = 1)
4. Those that started on the bench (*lineup* = 0) but did not get substituted in (*substituteIn* = 0)

Using Panda profiling :



Statistics | Histogram(s) | Common values | Extreme values

Value	Count	Frequency (%)
90	6814	61.9%
0	2501	22.7%
46	175	1.6%
76	55	0.5%
67	54	0.5%
78	53	0.5%
84	53	0.5%
79	52	0.5%
71	51	0.5%
77	50	0.5%
Other values (69)	1144	10.4%

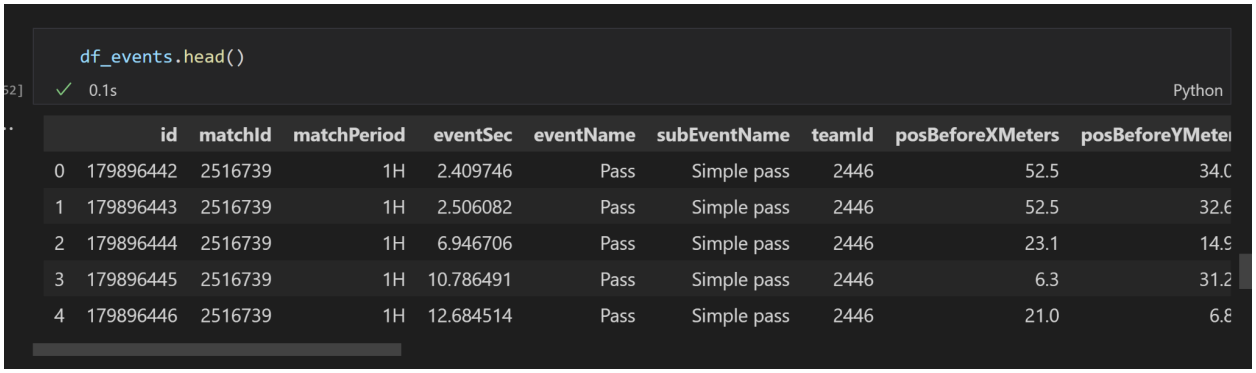
- Most substitutions happen between minute 70 and 80

- The minute with the most substitutions is the 46th (not surprisingly :-))

- The first substitutions already happened after 8 minutes

Event data :

Taking a look at a sample of the team dataset :



```
df_events.head()
```

✓ 0.1s Python

	id	matchId	matchPeriod	eventSec	eventName	subEventName	teamId	posBeforeXMeters	posBeforeYMeter
0	179896442	2516739	1H	2.409746	Pass	Simple pass	2446	52.5	34.0
1	179896443	2516739	1H	2.506082	Pass	Simple pass	2446	52.5	32.6
2	179896444	2516739	1H	6.946706	Pass	Simple pass	2446	23.1	14.9
3	179896445	2516739	1H	10.786491	Pass	Simple pass	2446	6.3	31.2
4	179896446	2516739	1H	12.684514	Pass	Simple pass	2446	21.0	6.8

Step 4 : Analysing The Data

Goal Kick Analysis:

In order to determine which teams are better at shooting and defending goal kicks using ,

→ **How many goal kicks a team scored** : Calculating the mean goals per team.

```
df_goal_kick = df_events[df_events["subEventName"] == "Goal kick"].copy()

print(f"Number of goal kicks per team and match: {(len(df_goal_kick) / df_goal_kick['matchId'].nunique() / 2):.1f}")
```

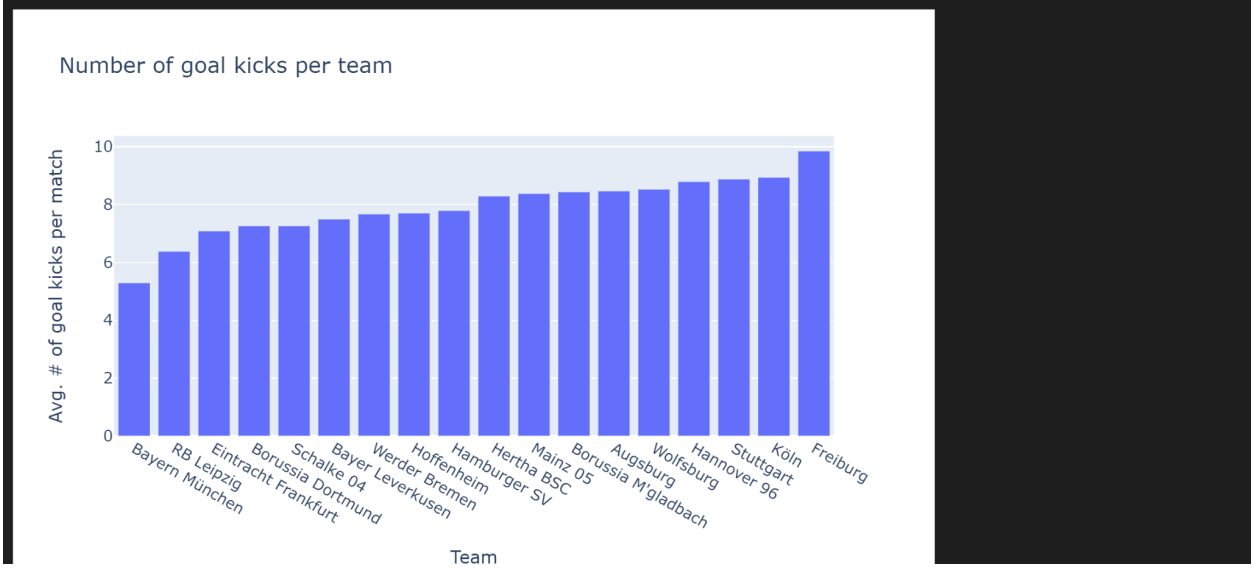
✓ 0.1s Python

```
Number of goal kicks per team and match: 7.9
```

The amount of goal kicks differs between the different teams :

```
# get total number of goal kicks per team
df_nb_gk = df_goal_kick.groupby("teamId").agg(nbGoalKicks=("teamId", "count")).reset_index().sort_values("nbGoalKicks")

# merge information of team such as team name and number of matches
df_nb_gk = pd.merge(df_nb_gk, df_league, on="teamId")

# compute number of goal kicks per match
df_nb_gk["goalKicksPerMatch"] = df_nb_gk["nbGoalKicks"] / df_nb_gk["matches"]

# plot bar chart with number of goal kicks per team
fig = px.bar(df_nb_gk, x="teamName", y="goalKicksPerMatch",
             labels={"goalKicksPerMatch":"Avg. # of goal kicks per match", "teamName": "Team"},
             title="Number of goal kicks per team")
fig.show()
```

✓ 1.6s



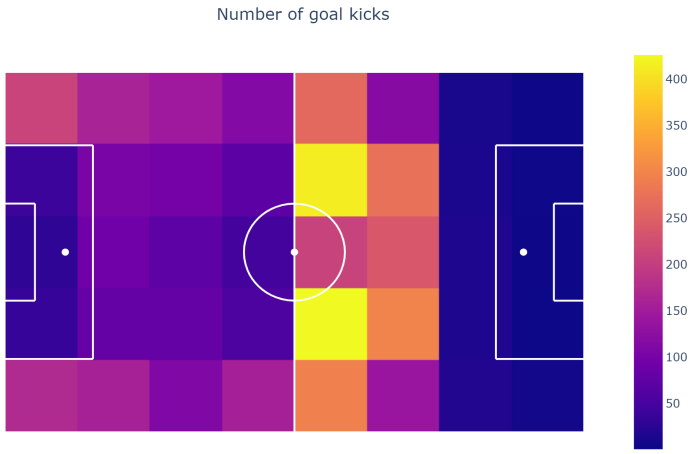
→ **End points of goal kicks** : How were these goal kicks made , the distance and position at which these goal kicks were successful .

Start by filtering the goal kicks that do not end up in the top left or the bottom right point of the field and compute the goals ending in each zone

splitting the field into zones and producing a heatmap to show how many goal kicks ended in each zone.We can divide the field into 40 zones and count the number of appearances for each zone.

```
nb_goal_kick
✓ 0.9s
array([[236, 160, 145, 114, 262, 117,  11,   2],
       [ 41, 104,  98,  72, 414, 273,  13,   3],
       [ 28,  93,  73,  47, 208, 237,  15,   2],
       [ 36,  81,  81,  56, 426, 297,  15,   1],
       [172, 158, 111, 156, 294, 139,  18,  38]], dtype=int64)
```

This can give us the total amount of goal kicks that have landed in each zone , the center point of the x and y zone.



→ **Accurate goal kicks** : The accuracy of these goal kicks is to be determined. This can be predicted by tracking as to what happens after the kick is made.

Taking a sample goal made by Ulreich and fetching all the necessary data :

	eventSec	subEventName	playerName	playerPosition	teamId	posBeforeXMeters	posBeforeYMeters	posAfterXMeters	posAfterYMeters
1046	678.384382	Goal kick	S. Ulreich	GK	2444	5.25	34.00	60.90	20.40
1047	682.102811	Air duel	A. Vidal	MD	2444	60.90	20.40	75.60	11.56
1048	682.132346	Air duel	D. Kohr	MD	2446	44.10	47.60	29.40	56.44
1049	685.392092	Simple pass	A. Dragović	DF	2446	29.40	56.44	7.35	38.76
1050	689.194186	Simple pass	B. Leno	GK	2446	7.35	38.76	30.45	53.04
1051	691.500076	Simple pass	J. Tah	DF	2446	30.45	53.04	32.55	51.68
1052	692.670984	Simple pass	C. Aránguiz	MD	2446	32.55	51.68	14.70	63.92
1053	693.139654	Simple pass	A. Dragović	DF	2446	14.70	63.92	26.25	65.28
1054	696.506104	Simple pass	A. Mehmedi	FW	2446	26.25	65.28	28.35	59.16

If the goal kicker's team is in possession of the ball following the goal kick, the goal kick should be considered accurate. A goal kick can also be considered accurate if the goal kicker's team made the first pass, shot, or free kick after the goal kick.

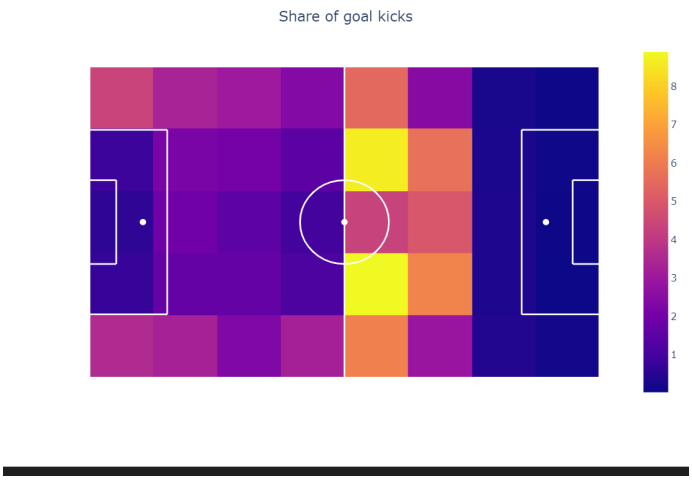
The main prediction is based on the first 20 secs after the goal kick. This would give us an idea as to exactly what happened after the kick and pave a way to future analysis.

```
print(f"Share of accurate goal kicks: {df_goal_kick['accurate'].mean()*100:.1f}%")
```

✓ 0.6s

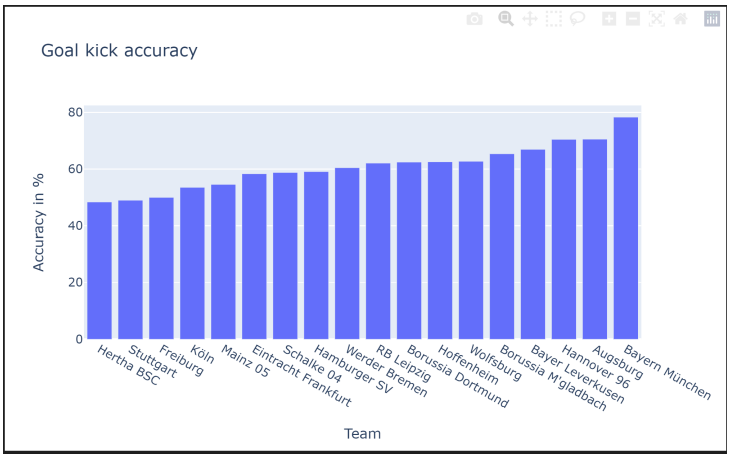
```
Share of accurate goal kicks: 60.2%
```

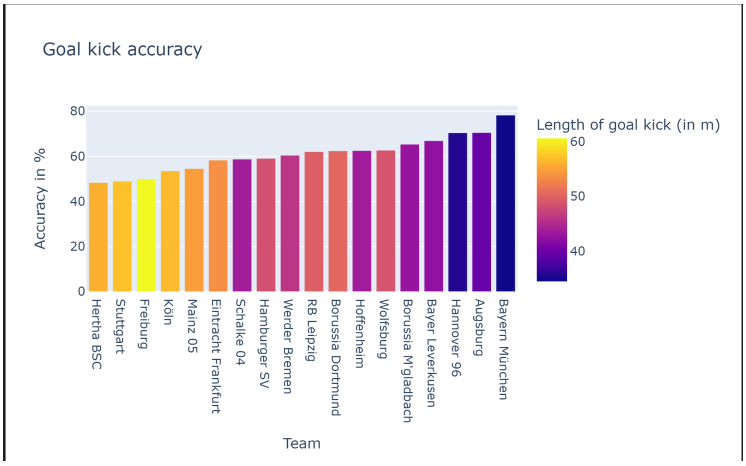
All the accurate goal kicks are extracted and the number of accurate goal kicks per zone is calculated. After which we can create a heatmap which shows the zones with the highest number of accurate goal kicks.



→ **Team performance** : Analyzing the performance of all the kicks made by all the teams and understanding the skill behind the teams that performed well.

Looking at the Goal kick Analysis for each team :





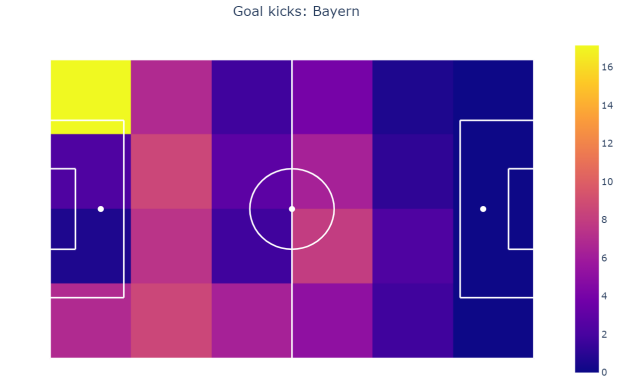
While Bayern only has ~35m on average, Freiburg averages ~60m. Moreover, we can tell that not surprisingly teams with longer goal kicks have less accuracy.

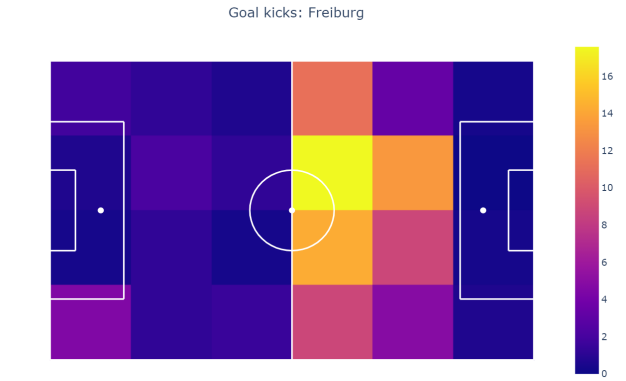
However:

1. Schalke seems to be relativly bad in goal kicks as the length is their length is rather short but they still only have an accuracy of ~59%

2. Wolfsburg on the other hand has long goal kicks with still an acceptable accuracy (~63%)

In order to understand exactly how Bayren and Freiburg's goal kick differ from one another :



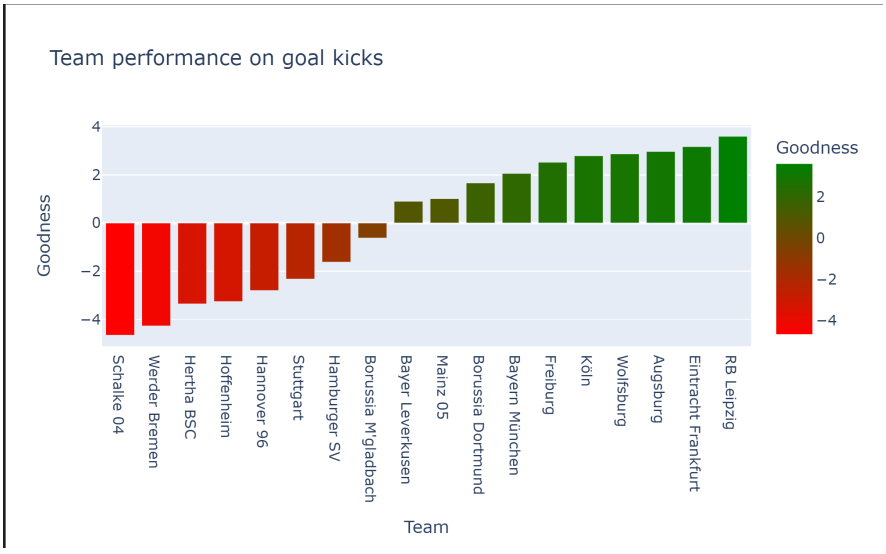


Bayern likes to play from the back while Freiburg likes to put the goal kicks shortly behind the center line.

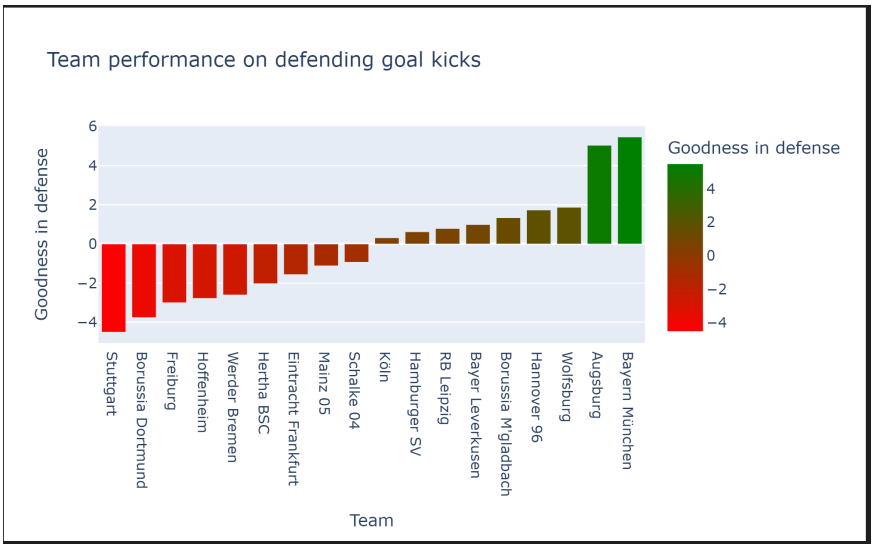
Starting with comparing the accuracy for each of the teams and understanding the different possible scenarios that could lead to a goal. Someone might lose the ball before gaining it back and scoring a goal.Some team might have a high accuracy in passing which would lead to an accurate goal , so the average length of the goal kick is used.

Different teams have players that comprise different players with varied playing traits. Someone might shoot short with a high accuracy , someone might shoot long with a high accuracy and vice versa.

The goodness of a team is the percentage point uplift a team has on goal kicks compared to the other teams.



A team is doing well in defending if the opponent is doing bad in their goal kick. We therefore multiply the goodness by -1 to get to the defending goodness.

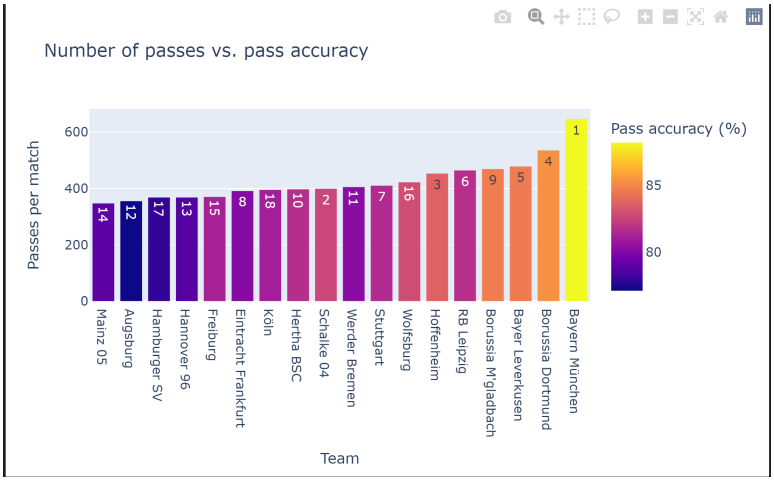


Passing Analysis:

For Computing statistics efficiently, we will be using the compute_statistics function. The position plot is used to indicate a player's position and their respective passing lines. Polar plots are passed to read the game setup. Gini coefficient is used to measure the team coordination and balance among the team players during the match.

Passing Per Team :

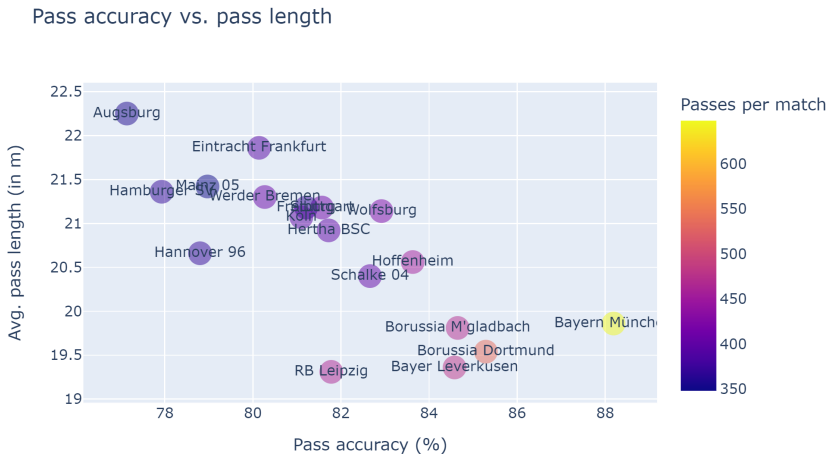
We now plot the number of passes together with the pass accuracy. We also add the league position of each team to the chart to see, if there is a correlation between teams doing well and the number of passes.



And there is definitely some correlation between the good teams, the number of passes and their accuracy. Nevertheless, there are some teams that are quite surprising. Take Augsburg, for example. They ended up 12th in league but were second worst in number of passes and worst in accuracy.

Maybe it has something to do with the length of the passes. Meaning, Augsburg is doing relatively bad in number of passes and accuracy because they play longer passes on average.

using a scatter plot in which we plot the accuracy vs. the pass length. This shows us quite nicely the negative correlation between pass length and accuracy.



Augsburg does indeed have a higher average pass length than the other teams. And also Frankfurt, which performs relatively bad on number of passes and accuracy compared to their league performance (see table above) plays relatively long passes on average. But, Schalke which finished second in the league and is neither extremely good in accuracy nor do they play a lot of passes.

Taking the game of German clasico, i.e. the home game of Borussia Dortmund against Bayern München.

Fetching player and team statistics:

	teamId	nbMatches	totalPasses	shareAccuratePasses	meanPassLength	totalShots	totalGoals	totalDuels	teamName
0	2444	1	485	87.42	20.68	10	3	220	Bayern München
1	2447	1	465	87.53	19.34	14	1	220	Borussia Dortmund

	playerId	playerName	playerPosition	teamId	nbMatches	totalPasses	shareAccuratePasses	meanPassLength	totalShots	totalGoals
0	3335	Bartra	DF	2447	1	46	82.61	24.55	1.0	1
1	3345	Thiago Alcântara	MD	2444	1	53	94.34	18.71	1.0	0
2	3416	Javi Martínez	MD	2444	1	31	93.55	18.03	0.0	0
3	14687	S. Papastathopoulos	DF	2447	1	13	100.00	28.20	0.0	0
4	14718	Rafinha	DF	2444	1	12	75.00	17.74	0.0	0

Measuring how well balanced a team is w.r.t. to the number of passes of each player by using the Gini coefficient of passes:

Two main points that we have to consider before measuring any compactness are:

1. Not all players played for 90 minutes; we should clean for that by rescaling the passes to 90 minutes
2. The goalie does have a special role; we should not consider him

We can only guess who will win a match by looking at the distribution of passes amongst the field players. We also have no idea how many passes, shots, or free kicks were made in total.

We calculate the number of shots for each team in each match, total number of shots for each game and finally calculate the average number of points of each team to find which one is higher.

Predictive Model :

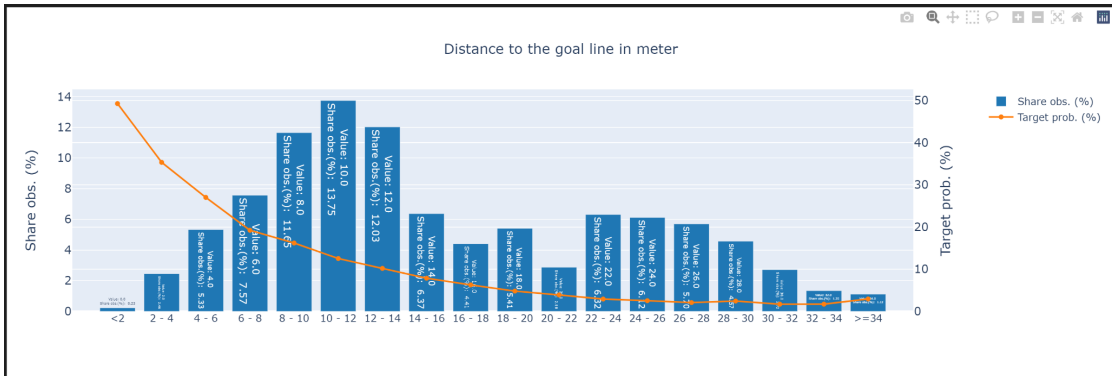
```
total_nb_shots = len(df_shots)
print(f"Total shots: {total_nb_shots}")
print(f"Expected goal when shooting: {df_shots['goal'].mean()*100:.2f}%")
```

```
Total shots: 40461
Expected goal when shooting: 10.56%
```

```
df_train, df_test, _, _ = train_test_split(df_shots, df_shots["goal"], test_size=0.25, random_state=42)
df_train.head()
```

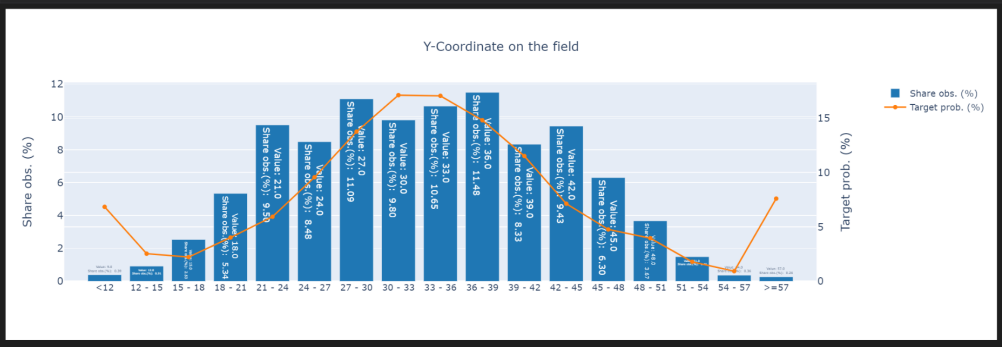
	id	matchId	matchPeriod	eventSec	eventName	subEventName	teamId	posBeforeXMeters	posBeforeYMeters	posAfterXMeters	posAfterYMeters	playerId	playerNan
372611	220884257	2499939	1H	1927.767965	Shot	Shot	1609	76.65	32.64	105.0	68.0	49876	G. Xha
501603	241111849	2565852	1H	776.495085	Shot	Shot	675	95.55	31.28	0.0	0.0	8278	G. Ba
596382	247255947	2500071	1H	2057.544092	Shot	Shot	1673	92.40	29.92	0.0	0.0	214654	S. Mour
199285	211492829	2516858	1H	1358.771285	Shot	Shot	2462	91.35	22.44	105.0	68.0	69616	A. Reb
425765	231574469	2565806	2H	388.734609	Shot	Shot	680	96.60	38.08	105.0	68.0	70125	Noli

```
df_train = df_train.copy()
df_train["distToGoalLine"] = 105 - df_train["posBeforeXMeters"]



df_train["distToGoalLineBuckets"] = ml_help.build_buckets(df_train, "distToGoalLine",
                                                          step_size=2,
                                                          min_val=1,
                                                          max_val=34)

fig, df_group = ml_help.create_univariate_variable_graph(df=df_train,
                                                         col="distToGoalLineBuckets",
                                                         target_col="goal",
                                                         binned_cols=True,
                                                         title_name="Distance to the goal line in meter")
fig.show()
```

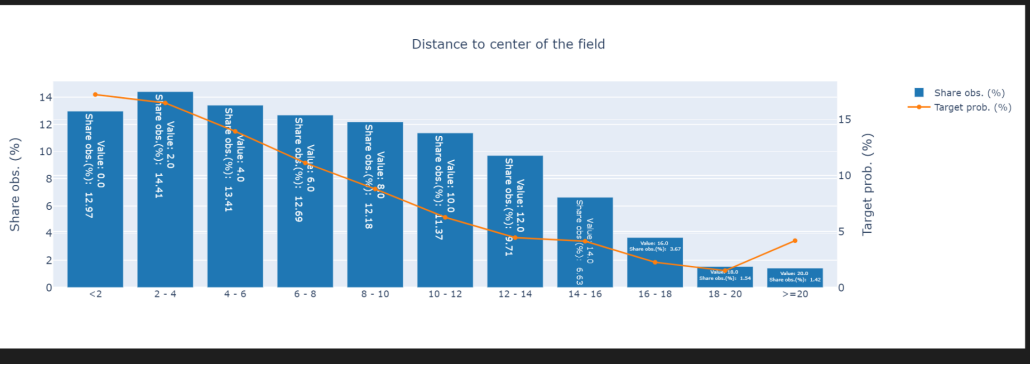



```
Y position of the shot
```

```
df_train["posBeforeYMetersBuckets"] = ml_help.build_buckets(df_train, "posBeforeYMeters",
                                                            step_size=3,
                                                            min_val=10,
                                                            max_val=58)

fig, df_group = ml_help.create_univariate_variable_graph(df=df_train,
                                                         col="posBeforeYMetersBuckets",
                                                         target_col="goal",
                                                         binned_cols=True,
                                                         title_name="Y-Coordinate on the field")
fig.show()
```




```
df_train["distToCenter"] = np.abs(34 - df_train["posBeforeYMeters"])

df_train["distToCenterBuckets"] = ml_help.build_buckets(df_train, "distToCenter",
                                                        step_size=2,
                                                        max_val=20)

fig, df_group = ml_help.create_univariate_variable_graph(df=df_train,
                                                         col="distToCenterBuckets",
                                                         target_col="goal",
                                                         binned_cols=True,
                                                         title_name="Distance to center of the field")
fig.show()
```



```
df_train.groupby("bodyPartShot").size()
```

```
bodyPartShot
head/body     4869
leftFoot     10011
rightFoot    15465
dtype: int64
```

```
df_train.groupby("playerStrongFoot").size()
```

```
playerStrongFoot
both          71
left        7452
right      22812
unknown        7
dtype: int64
```

```
df_train_trans = transformation_univariate(df_train_raw)
df_test_trans = transformation_univariate(df_test_raw)
```

```
features = ["distToGoalLine", "distToCenter", "head/body", "strongFoot", "weakFoot", "counterAttack"]
print(f"Transformation done correctly: {ml_help.check_column_match(df_train, df_train_trans, features)}")
```

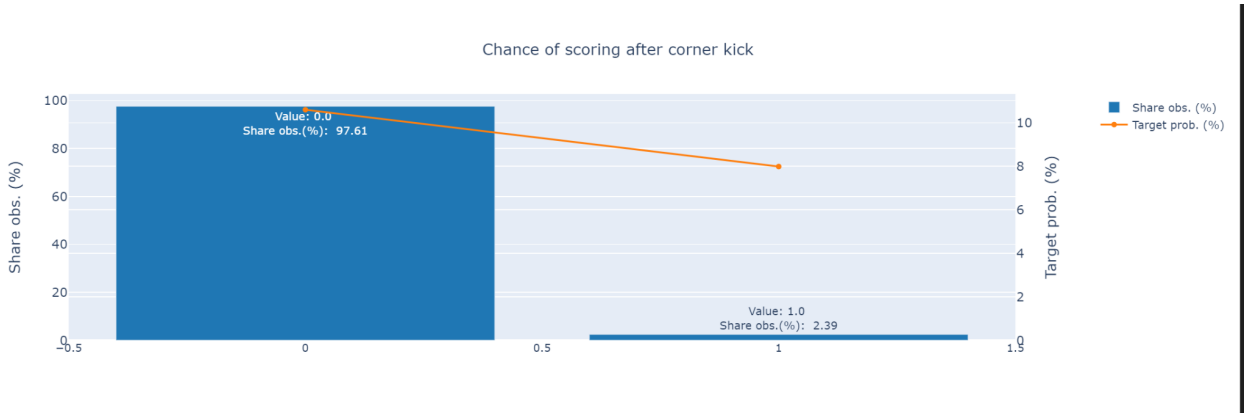
```
Transformation done correctly: True
```

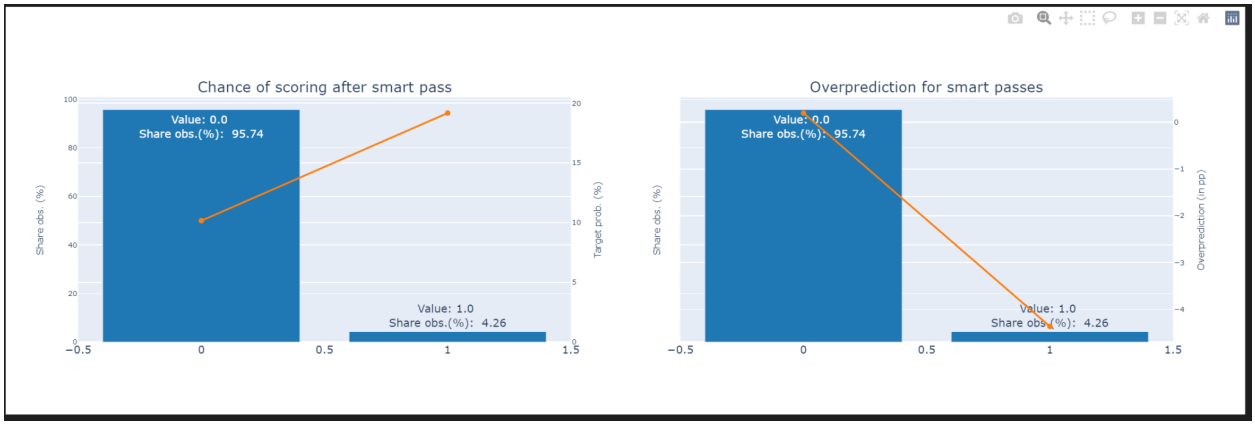
```
# training of the logistic regression on the train set
reg_uni_a = LogisticRegression(random_state=42)
reg_uni_a.fit(df_train_trans[features], np.array(df_train_trans["goal"]).ravel())

# prediction on the test set
pred_probs = reg_uni_a.predict_proba(df_test_trans[features])[:,1]
print(f"Log loss on test set after univariate analysis part A: {sk_metrics.log_loss(df_test_trans['goal'], pred_probs):.5f}")
```

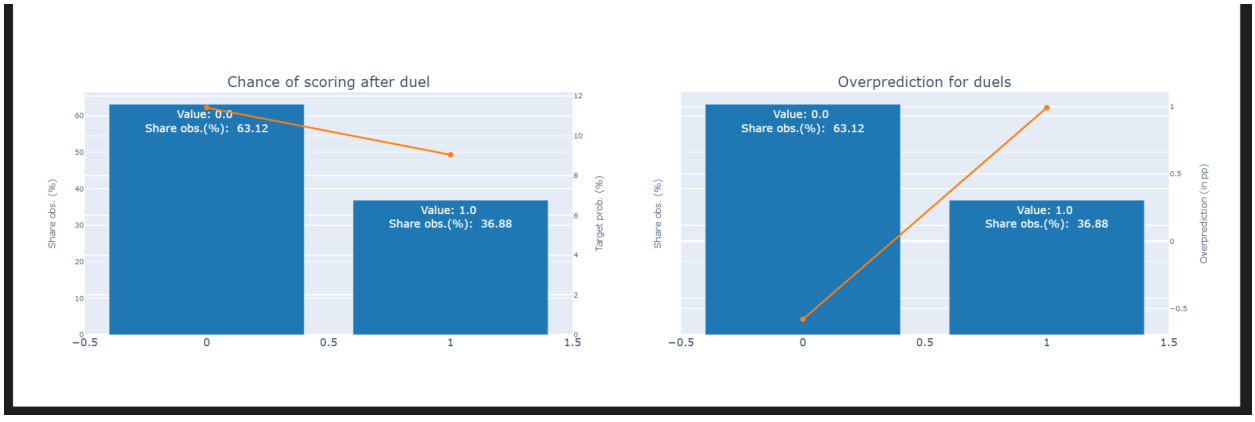
```
Log loss on test set after univariate analysis part A: 0.28791
```

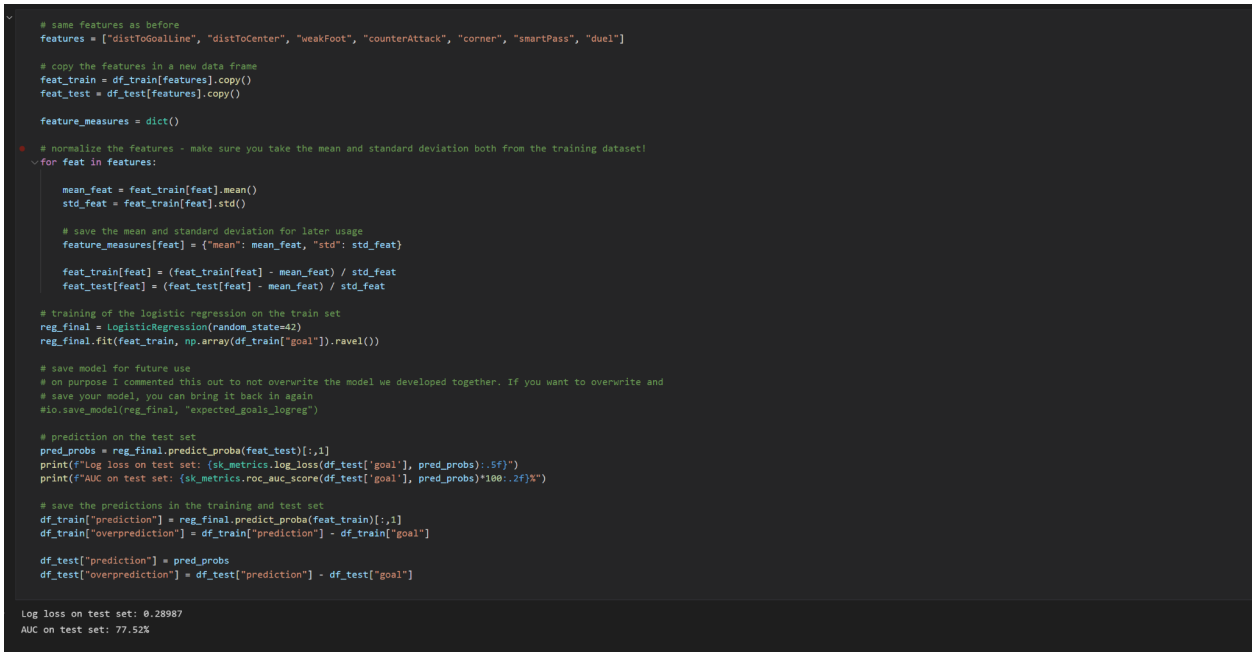
```
pred_probs_train = reg_uni_a.predict_proba(df_train_trans[features])[:,1]
df_train_trans["prediction"] = pred_probs_train
print(f"Log loss on train set after univariate analysis part A: {sk_metrics.log_loss(df_train_trans['goal'], pred_probs_train):.5f}")
print(f"AUC on test set after univariate analysis part A: {sk_metrics.roc_auc_score(df_test_trans['goal'], pred_probs)*100:.2f}%")
```

```
Log loss on train set after univariate analysis part A: 0.28931
AUC on test set after univariate analysis part A: 77.79%
```


Chance of scoring after corner kick
Share obs. (%)
Target prob. (%)
Value: 0.0
Share obs.(%): 97.61
Value: 1.0
Share obs.(%): 2.39


Chance of scoring after smart pass
Value: 0.0
Share obs.(%): 95.74
Value: 1.0
Share obs.(%): 4.26
Overprediction for smart passes
Overprediction (in pp)


Chance of scoring after duel
Value: 0.0
Share obs.(%): 63.12
Value: 1.0
Share obs.(%): 36.88
Overprediction for duels



```
# same features as before
features = ["distToGoalLine", "distToCenter", "weakFoot", "counterAttack", "corner", "smartPass", "duel"]

# copy the features in a new data frame
feat_train = df_train[features].copy()
feat_test = df_test[features].copy()

feature_measures = dict()

# normalize the features - make sure you take the mean and standard deviation both from the training dataset!
for feat in features:

    mean_feat = feat_train[feat].mean()
    std_feat = feat_train[feat].std()

    # save the mean and standard deviation for later usage
    feature_measures[feat] = {"mean": mean_feat, "std": std_feat}

    feat_train[feat] = (feat_train[feat] - mean_feat) / std_feat
    feat_test[feat] = (feat_test[feat] - mean_feat) / std_feat

# training of the logistic regression on the train set
reg_final = LogisticRegression(random_state=42)
reg_final.fit(feat_train, np.array(df_train["goal"]).ravel())

# save model for future use
# on purpose I commented this out to not overwrite the model we developed together. If you want to overwrite and
# save your model, you can bring it back in again
#io.save_model(reg_final, "expected_goals_logreg")

# prediction on the test set
pred_probs = reg_final.predict_proba(feat_test)[:,1]
print(f"Log loss on test set: {sk_metrics.log_loss(df_test['goal'], pred_probs):.5f}")
print(f"AUC on test set: {sk_metrics.roc_auc_score(df_test['goal'], pred_probs)*100:.2f}%")

# save the predictions in the training and test set
df_train["prediction"] = reg_final.predict_proba(feat_train)[:,1]
df_train["overprediction"] = df_train["prediction"] - df_train["goal"]

df_test["prediction"] = pred_probs
df_test["overprediction"] = df_test["prediction"] - df_test["goal"]

Log loss on test set: 0.28987
AUC on test set: 77.52%
```

6. Comparison/Conclusion:

This research demonstrated that data analytics can add value in the decision making process in sport settings such as the NFL. It also demonstrated that humans are central to the effective implementation of sport analytics,Coaches and sport managers are central in the collection of data, in the analysis of that data, and in the application of that data analysis.Logistic regression is easier to implement, interpret, and very efficient to train. If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.The heavy dataset of soccer analytics makes this an effective model with high accuracy.

7. References - IEEE std.

[1]
https://tomdecroos.github.io/reports/interpret_vaep.pdf
[2]
https://dr.lib.iastate.edu/server/api/core/bitstreams/f37d0926-6637-4686-b009-593a6513c940/content
[3]
https://www.researchgate.net/publication/346862578_Analysis_and_Prediction_of_Soccer_Games_An_Application_to_the_Kaggle_European_Soccer_Database
[4]

https://pdfs.semanticscholar.org/4f5b/70cb4a50eea46980a8d8ba9e105a83ab642f.pdf?_ga=2.262993740.1750646625.1647273350-1065461782.1646011745