PROJECT: Football xG Predictor

Milestone: Draft of Project Report

Group 10

Neel Kamal
Rahul Daruka

(857) 399-7983
(617) 936-9851

kamal.ne@northeastern.edu
daruka.r@northeastern.edu

Percentage of Effort Contributed by Student 1:_____50%_____

Percentage of Effort Contributed by Student 2:_____50%_____

Signature of Student 1:_____Neel Kamal_____

Signature of Student 2:_____Rahul Daruka_____

Submission Date:_____03-22-2024_____

1. Problem Setting:

Football (soccer) is a sport where scoring goals is the ultimate objective, and understanding the factors influencing the likelihood of a successful goal is crucial. Expected Goals (xG) is a statistical metric that quantifies the probability of a shot resulting in a goal. This project aims to leverage machine learning techniques to predict the xG of football shots based on various features.

2. Problem Definition:

The primary objective is to build a predictive model that can estimate the xG of a given shot. The model will take into account features such as shot distance, angle to the goal, and potentially other factors derived from applying feature engineering on match events. We aim to predict the probability of a shot converting into a goal.

3. Data Sources:

The project utilizes football event data obtained from the Wyscout platform, specifically the events datasets for the top five European football leagues (England, Spain, Italy, Germany, France). The data is publicly available and has been cited from the research paper by Luca Pappalardo et.al. https://doi.org/10.6084/m9.figshare.c.4415000.v5

4. Data Description:

The datasets have 12 columns and approx. of 600k rows that include information on various match events, with a focus on shots (eventId=10) i.e. 40,461rows. The relevant features for building the xG model include:

- Shot positions (start and end coordinates)
- Tags indicating the outcome of the shot (e.g., goal, miss)
- Other relevant features derived from event data

A sample of variables includes shot distance, shot angle, and the outcome of the shot (goal or miss). The datasets provide a comprehensive set of information on which the required features can be selected and calculated using feature engineering for building and evaluating the xG model.

The head of the required data frame is shown below.

```
In [36]: finalShotData.head()
```

Out[36]:

	eventId	subEventName	tags	playerId	positions	matchId	eventName	teamId	matchPeriod	eventSec	subEventId	id
0	10	Shot	[{'id': 101}, {'id': 402}, {'id': 201}, {'id':...	25413	[{'y': 41, 'x': 88}, {'y': 0, 'x': 0}]	2499719	Shot	1609	1H	94.595788	100	177959212
1	10	Shot	[{'id': 401}, {'id': 201}, {'id': 1211}, {'id'...	26150	[{'y': 52, 'x': 85}, {'y': 100, 'x': 100}]	2499719	Shot	1631	1H	179.854785	100	177959247
2	10	Shot	[{'id': 101}, {'id': 403}, {'id': 201}, {'id':...	14763	[{'y': 52, 'x': 96}, {'y': 100, 'x': 100}]	2499719	Shot	1631	1H	254.745027	100	177959280
3	10	Shot	[{'id': 401}, {'id': 201}, {'id': 1215}, {'id'...	7868	[{'y': 33, 'x': 81}, {'y': 0, 'x': 0}]	2499719	Shot	1609	1H	425.824035	100	177959289
4	10	Shot	[{'id': 402}, {'id': 201}, {'id': 1205}, {'id'...	7868	[{'y': 30, 'x': 75}, {'y': 0, 'x': 0}]	2499719	Shot	1609	1H	815.462015	100	177959429

The shape of the required data frame is show below.

```
finalShotData.shape
```

```
(40461, 12)
```

The info of the required data frame is shown below.

```
finalShotData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 40461 entries, 46 to 632802
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   eventId       40461 non-null  int64
 1   subEventName  40461 non-null  object
 2   tags          40461 non-null  object
 3   playerId      40461 non-null  int64
 4   positions     40461 non-null  object
 5   matchId       40461 non-null  int64
 6   eventName     40461 non-null  object
 7   teamId        40461 non-null  int64
 8   matchPeriod   40461 non-null  object
 9   eventSec      40461 non-null  float64
 10  subEventId    40461 non-null  object
 11  id            40461 non-null  int64
dtypes: float64(1), int64(5), object(6)
memory usage: 4.0+ MB
```

4. Data Mining Tasks:

The first steps we take in performing data mining tasks is to extract the exact X-Y coordinates from our data frame as the coordinates are stored as a list of dictionaries. By performing feature extraction, we can now use x and y coordinates to find position of the shots taken.

positions
[{'y': 41, 'x': 88}, {'y': 0, 'x': 0}]
[{'y': 52, 'x': 85}, {'y': 100, 'x': 100}]
[{'y': 52, 'x': 96}, {'y': 100, 'x': 100}]
[{'y': 33, 'x': 81}, {'y': 0, 'x': 0}]
[{'y': 30, 'x': 75}, {'y': 0, 'x': 0}]

will be converted to

startY	startX	endY	endX
41	88	0	0
52	85	100	100
52	96	100	100
33	81	0	0
30	75	0	0

The next step of feature extraction and data reduction is to use tags mentioned in our data frame and find required tags based on our project requirements. The descriptions of the tags are mentioned in the research paper cited above. For example, 101 is represented as a Goal, 301 is Assist, 401 is Left Leg Used, 602 is anticipation. Link to Tags

For this projection we need to know if the shot taken has resulted in a goal or not. The variable will be considered as our Target Variable while designing the model. To extract this feature, we create a table that extracts all the tags of each event and extract tag 101 and label it to Goal or No Goal.

	0	1	2	3	4	5
0	{'id': 101}	{'id': 402}	{'id': 201}	{'id': 1205}	{'id': 1801}	None
1	{'id': 401}	{'id': 201}	{'id': 1211}	{'id': 1802}	None	None
2	{'id': 101}	{'id': 403}	{'id': 201}	{'id': 1207}	{'id': 1801}	None
3	{'id': 401}	{'id': 201}	{'id': 1215}	{'id': 1802}	None	None
4	{'id': 402}	{'id': 201}	{'id': 1205}	{'id': 1801}	None	None
...	...	...	...	...	...	...
40456	{'id': 401}	{'id': 201}	{'id': 1205}	{'id': 1801}	None	None
40457	{'id': 402}	{'id': 201}	{'id': 1214}	{'id': 1802}	None	None
40458	{'id': 401}	{'id': 2101}	{'id': 201}	{'id': 1802}	None	None
40459	{'id': 401}	{'id': 201}	{'id': 1201}	{'id': 1801}	None	None
40460	{'id': 101}	{'id': 401}	{'id': 201}	{'id': 1208}	{'id': 1801}	None

This data is converted to

startY	startX	endY	endX	goal
41	88	0	0	1
52	85	100	100	0
52	96	100	100	1
33	81	0	0	0
30	75	0	0	0

To perform the next data mining procedure, we will use tags 401, 402, and 403 to determine the body part that was used to perform the shot i.e. right foot, left foot, or using header or body. This will later help us determine if a player used his strong or week foot. This will create an important categorical feature that will contribute to our model building.

```
   eventfoot
0      right
1       left
2     header
3       left
4      right
```

Now to determine if the foot used was strong foot or week foot, we will be importing a second dataset provided in the same research paper that provides the details of the players.
Link to Player Dataset We will be merging this dataset to our current data using left join on "playerid". This will give us additional information about the player who was involved in the event. We will then compare the preferred foot of the player to the foot used in the event to determine if it was strong foot or week foot.

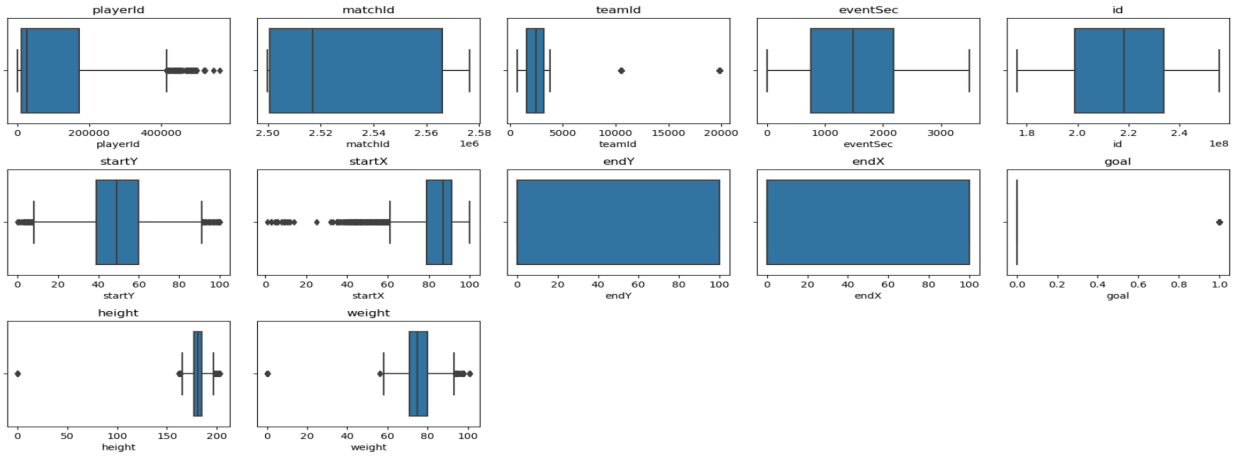
startX	...	endX	goal	firstName	lastName	foot	height	weight	role.name	eventfoot	shot_foot
88	...	0	1	Alexandre	Lacazette	right	175.0	73.0	Forward	right	strong foot
85	...	100	0	Riyad	Mahrez	left	179.0	62.0	Midfielder	left	strong foot
96	...	100	1	Shinji	Okazaki	right	174.0	70.0	Forward	header	header
81	...	0	0	Alex	Oxlade-Chamberlain	right	175.0	70.0	Midfielder	left	weak foot
75	...	0	0	Alex	Oxlade-Chamberlain	right	175.0	70.0	Midfielder	right	strong foot

Using this new dataset will also provide additional information that can be used to build our model. We will now go through our data frame and drop the columns/features that are not required or not have impact towards the model or are duplicates in the dataset, this will be one of the dimension reductions. The columns that are being dropped are 'eventfoot','eventId','eventName', 'positions','startYX','endYX'.
The next step is to check for any missing values that are present in our merged dataset.

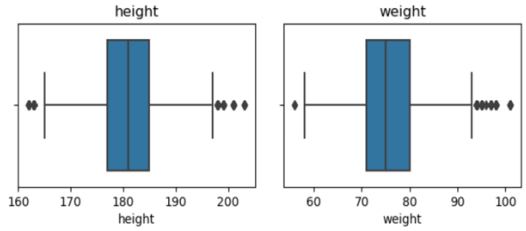
```
goal          0
firstName     3
lastName      3
foot          3
height        3
weight        3
role.name     3
shot_foot     0
```

Since the number of null values are very low, we can drop those rows.

We will now find any outliers in our dataset that can be done by constructing boxplots for the numerical values.



It's very evident from the boxplots that height and weight have few extreme outliers that may impact our model negatively. From the visual representation we can see that the height and weight are zero, which is not possible in a case of a humans. So, we are going to remove these zero values by removing the rows that consist of those values.



This is the new boxplots post removal of 0 values.

We can perform a chi-squared test to analyze the relationship between the player's foot used to shoot suck as strong foot or week foot (shot_foot) and the outcome of whether a goal was scored (goal). This will help us determine if there is a significant association between the player's preferred foot and their likelihood of scoring a goal.

```
Chi-squared statistic: 46.202173189667654
P-value: 9.275248755192316e-11
```

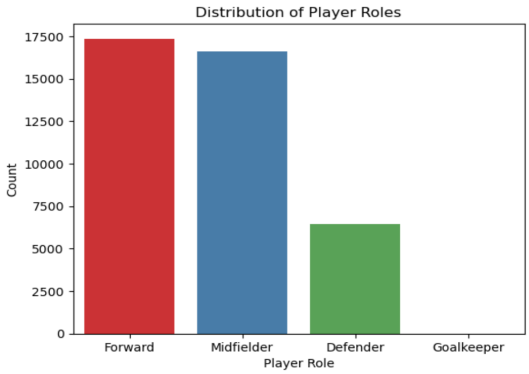
Based on the output, The chi squared value represents the result of the chi-squared test. It indicates the strength of the association between the two variables. In this case, the chi-squared statistic is approximately 46.20 and the p-value is approximately 9.28e-11 (or approximately 0), which is extremely low. Therefore, we can conclude that there is a significant association between a player's preferred foot and their likelihood of scoring a goal.

5. Data Exploration:

The statistical description of our final dataset is shown below.

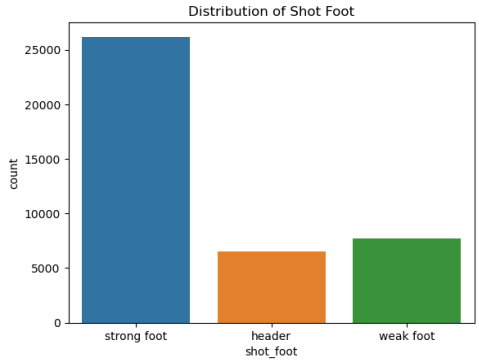
	playerId	matchId	teamId	eventSec	id	startY	startX	endY	endX	goal	height	weight
count	40458.000000	4.045800e+04	40458.000000	40458.000000	4.045800e+04	40458.000000	40458.000000	40458.000000	40458.000000	40458.000000	40458.000000	40458.000000
mean	96666.318750	2.532575e+06	2607.857902	1472.327127	2.173399e+08	49.232537	84.828266	44.522715	44.522715	0.105566	181.057813	75.423427
std	122838.088397	3.319150e+04	2209.194665	818.662880	2.145643e+07	13.761569	8.096794	49.699702	49.699702	0.307285	6.792741	6.950713
min	36.000000	2.499719e+06	674.000000	1.238426	1.761217e+08	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	11294.000000	2.500764e+06	1612.000000	762.299077	1.989030e+08	39.000000	79.000000	0.000000	0.000000	0.000000	177.000000	71.000000
50%	25715.000000	2.516887e+06	2455.000000	1480.584749	2.181691e+08	49.000000	87.000000	0.000000	0.000000	0.000000	181.000000	75.000000
75%	173214.000000	2.565869e+06	3197.000000	2181.203862	2.337738e+08	60.000000	91.000000	100.000000	100.000000	0.000000	185.000000	80.000000
max	564512.000000	2.576338e+06	19830.000000	3490.826794	2.557092e+08	100.000000	100.000000	100.000000	100.000000	1.000000	203.000000	101.000000

Count Plot (Distribution Of player roles)



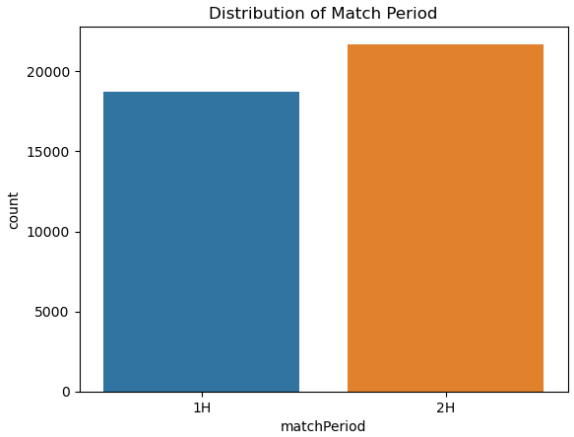
The count plot displays player role distribution with Forward and Midfielder roles being most common, followed by Defenders, and then Goalkeepers, as expected. This visualization aids in understanding dataset composition, reflecting typical team formations and potential insights into player involvement in events like shots and goals.

Count Plot (Distribution of Shot Foot)



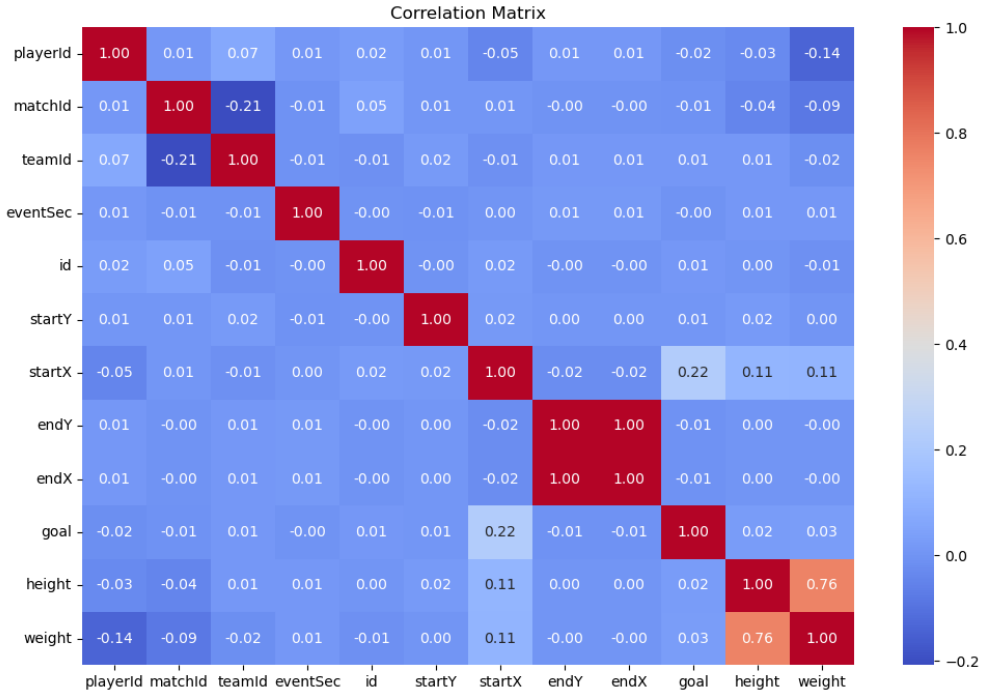
The count plot illustrates shot distribution by foot type in football data. Strong foot shots are most common, followed by headers and weak foot shots, indicating player preferences. This insight is valuable for xG model predictions, as strong foot shots may have higher goal probabilities.

Count Plot (Match Period)



The count plot compares event occurrences in the first half (blue) and second half (orange) of football matches, revealing increased activity in the latter. This insight aids in strategizing around players' stamina and game dynamics as matches progress.

Correlation Matrix



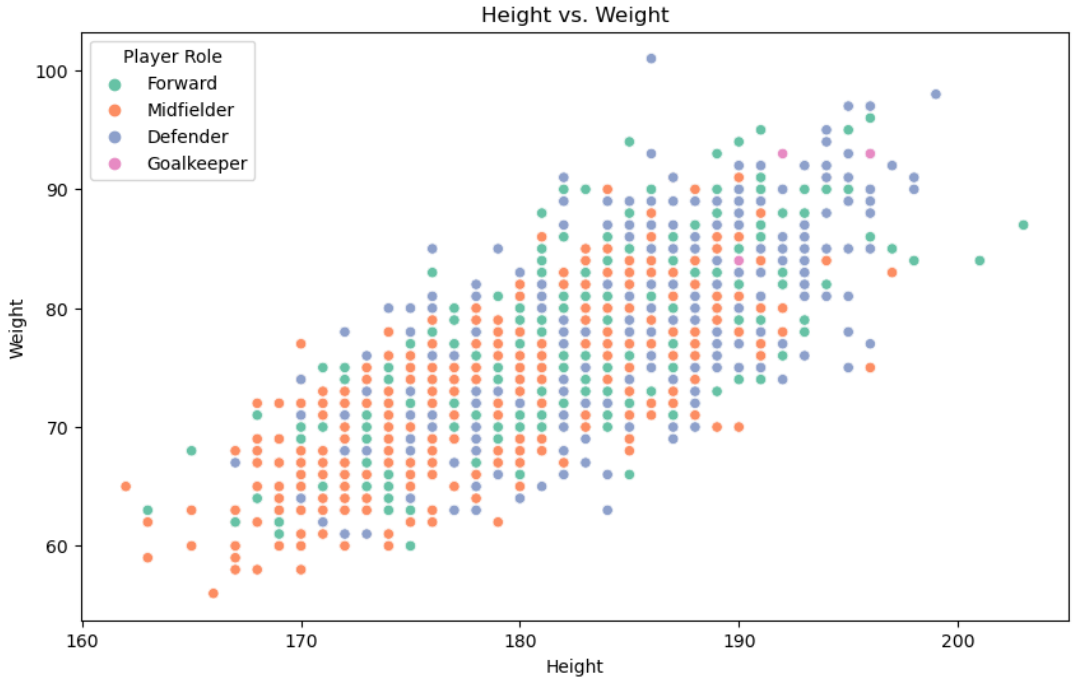
The correlation matrix highlights relationships between variables in a dataset. Key findings include:

Low correlations for playerId, matchId, teamId, and id, indicating they are independent.

Positive correlations between startX and endX (0.22), and goal and endY (0.22).

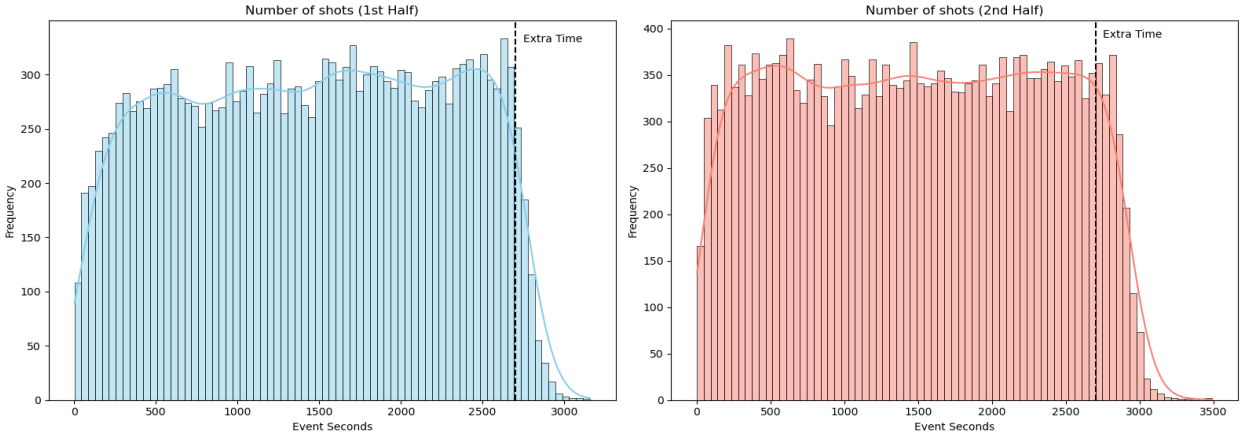
A strong positive correlation between height and weight (0.76), suggesting taller players tend to be heavier. These insights aid in feature selection for xG models, helping identify variables that influence shot outcomes.

Scatter Plot (Height vs Weight)



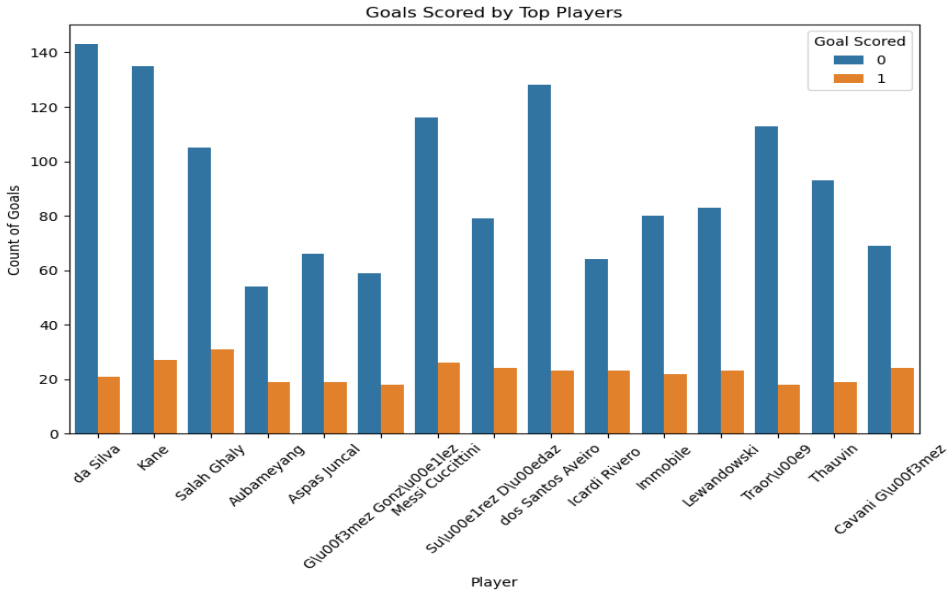
The scatter plot shows the relationship between height and weight for players in different positions: Forward, Midfielder, Defender, and Goalkeeper. It reveals a positive correlation between height and weight and highlights potential physical profiles associated with each position, such as taller and heavier players for defenders and goalkeepers. This insight aids in understanding positional characteristics and their impact on player performance.

Histogram (Shot frequency over time)



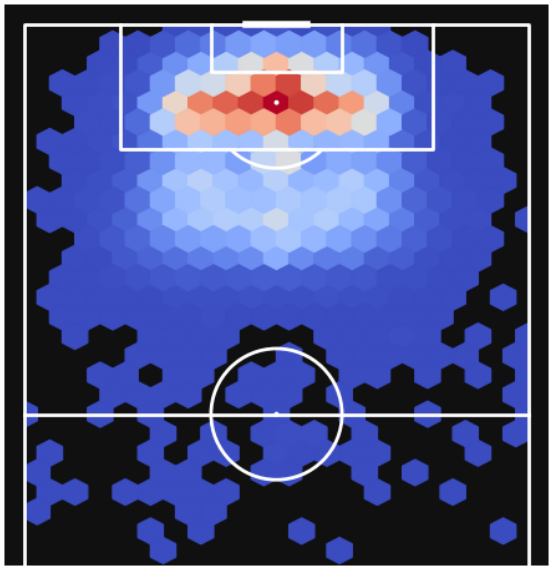
The histograms display shot frequency over time in football matches' first and second halves. They show peaks at the beginning and end of each half, indicating heightened attacking play, while tapering off in the middle as teams possibly adopt a more cautious approach. The sparse events during "Extra Time" align with its shorter duration. These insights aid in strategizing defensive and attacking tactics throughout a match.

Bar Graph (Goals scored by top players)



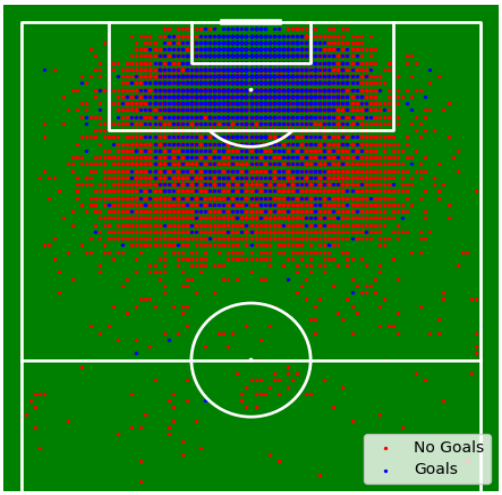
The bar chart compares goals scored and shots without goals for top players, aiding in assessing goal-scoring efficiency and shooting activity. It helps identify players with higher conversion rates and those who take more attempts but with lower success rates, informing performance analysis and tactical strategies.

Heatmap (Shot Distribution)



The is a heat map overlaid on a soccer field, representing the locations from where shots are taken. The intensity of the colors likely corresponds to the frequency of shots from those areas — warmer colors (red to yellow) indicate higher shot frequency, while cooler colors (blue) indicate fewer shots. This type of visualization is useful for analyzing spatial patterns in shot-taking, such as which areas of the field yield more attempts on goal. For example, a concentration of warmer colors around the penalty area suggests a high volume of shots taken from that region, which is expected as it's closer to the goal. Such insights can inform teams about defensive strengths or weaknesses and offensive strategies

Shot Map (Goal or NO Goal)



From this visualization, one can infer where most shots are taken, and which areas have a higher success rate. For instance, if the blue dots are concentrated around the goal area, it would indicate that shots taken close to the goal are more likely to result in goals, which is a common finding in soccer analytics. This type of data is invaluable for teams to develop tactical strategies and for coaching staff to understand patterns in gameplay.

5. Data Mining Models/Methods

Selecting the right machine learning model is critical for building an effective predictive model for estimating expected goals (xG) in football shots. In this analysis, we'll evaluate several data mining models and methods, considering their applicability, strengths, and limitations in relation to our dataset. By understanding how each model aligns with our objectives, we aim to choose the most suitable approach for predicting goal outcomes in football shots.

1. Logistic Regression

- Applicability: Well-suited for binary classification tasks like goal or no goal prediction. It can handle both numerical and categorical features present in your dataset, such as player attributes and match details.
- Pros: Simple and interpretable, which can be advantageous for understanding the factors influencing goal outcomes. It's also computationally efficient and doesn't require extensive tuning.
- Cons: Assumes a linear relationship between the features and the log odds of the target variable. It may not capture complex interactions between features without extensive feature engineering.

2. Decision Trees

- Applicability: Capable of handling both numerical and categorical data, making it suitable for our mixed dataset. Decision trees can capture non-linear relationships between features and the target variable.
- Pros: Easy to interpret and understand, making it valuable for extracting insights into the factors affecting goal outcomes. Requires minimal data preprocessing compared to some other models.
- Cons: Prone to overfitting, especially with complex datasets or deep trees. It may not generalize well to unseen data without careful pruning or regularization.

3. Random Forest

- Applicability: Particularly well-suited due to its ability to handle mixed data types and its robustness to overfitting compared to single decision trees.
- Pros: Can model complex relationships between features and the target variable, providing high predictive accuracy. Random forests are less prone to overfitting than individual decision trees and require less parameter tuning.
- Cons: Less interpretable than decision trees, as it's a collection of multiple trees. Training can be computationally intensive for large datasets, but it's generally manageable.

4. Gradient Boosting Machines (GBM)

- Applicability: Effective for structured datasets, where capturing complex relationships between features and the target variable is crucial.
- Pros: Offers high predictive accuracy by iteratively improving on the weaknesses of decision trees.
- Cons: Requires careful hyperparameter tuning to prevent overfitting. Training can be computationally expensive, especially with large datasets, but it often pays off in improved performance.

5. Support Vector Machines (SVM)

- Applicability: Can be used for binary classification tasks like goal prediction after appropriate preprocessing of our dataset.
- Pros: Effective in high-dimensional spaces and versatile with different kernel functions, which can capture complex relationships between features and the target variable.
- Cons: Requires significant data preprocessing, such as scaling features, and may not perform well with large datasets due to its computational complexity. Choosing the right kernel and tuning hyperparameters can also be challenging.

6. Neural Networks

- Applicability: Suitable for capturing complex and non-linear relationships present in our dataset, potentially improving prediction accuracy.

- Pros: Highly flexible and capable of modeling very intricate patterns in the data. Neural networks can automatically learn relevant features from the data, reducing the need for extensive feature engineering.
- Cons: Requires a large amount of data to train effectively and prevent overfitting. Neural networks are computationally intensive and may require significant resources for training, especially with deep architectures.

7. k-Nearest Neighbors (k-NN)
- Applicability: Simple to apply and can make predictions based on similarities between data points in the feature space.
- Pros: Easy to understand and implement, making it useful for quick prototyping. It doesn't require model training and can adapt well to changes in the data over time.
- Cons: Computationally expensive as the dataset grows, especially during prediction time. Performance heavily depends on the choice of distance metric and the value of k, which may require experimentation.

Selection Criteria
- Data Complexity and Size: Considering the complexity of relationships and the size of our dataset, models like Random Forest, GBM, or Neural Networks might offer better predictive performance.
- Interpretability Needs: If interpretability is crucial for our analysis, simpler models like Logistic Regression or Decision Trees might be preferred.
- Computation Resources: For limited computational resources, Logistic Regression or Decision Trees are more suitable. However, if resources allow more complex models like Random Forest or Neural Networks could yield better results.

6. Model Performance Evaluation and Interpretation

For the purpose of enhancing our predictive model's accuracy in estimating the Expected Goals (xG) for football shots, we meticulously calculated two pivotal features from the available match event data: the distance of the shot from the goal and the angle of the shot.

- Calculation of Shot Distance

Shot distance = sqrt{(x2 - x1)^2 + (y2 - y1)^2}

In this formula:

- (x1) and (y1) represent the starting X and Y coordinates of the shot (obtained from the `startX` and `startY` columns in our data).
- (x2) and (y2) are set to 100 and 50, respectively, denoting the standardized coordinates for the center of the goal line on a football pitch.

This approach assumes a scaled model of a football field where the goal line is at a fixed position (100, 50), facilitating the calculation of shot distances across various match events.

- Calculation of Shot Angle

The shot angle is calculated as the angle at the shot's origin point, formed by two lines extending from this point to each post of the goal. This angle is a critical factor in determining the likelihood of a shot resulting in a goal, as it quantifies the visible goalmouth from the shooter's perspective.

To compute the shot angle, we first determined the distances from the shot origin to each goal post and the distance between the goal posts:

1. (a) and (b) are the distances from the shot origin to the left and right goal posts, respectively.
2. (c) is the distance between the two goal posts, which is constant due to the standardized goal width.

Using the Law of Cosines, we calculated the shot angle (θ) as follows:

θ = arccos ({a^2 + b^2 - c^2}/{2ab})

The resulting angle is then converted from radians to degrees to provide a more intuitive measurement of the shot's angle relative to the goal:

ShotAngle = (θ * 180)/pi

This calculation provides us with a quantifiable measure of the shot angle, significantly contributing to our model's predictive capabilities by incorporating the spatial dynamics of shot-taking in football.

- Label encoder

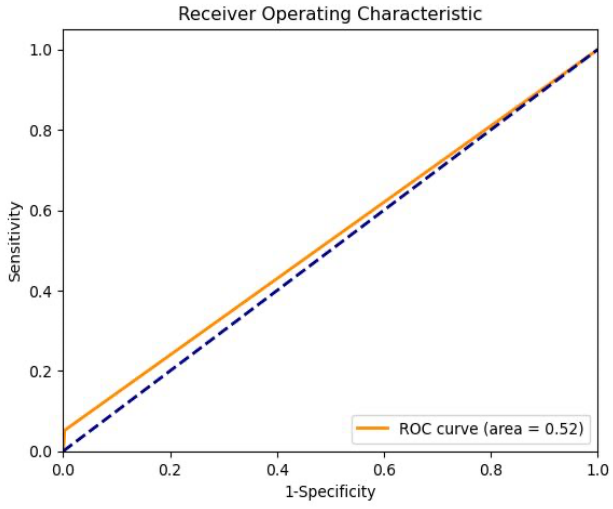
To prepare our dataset for machine learning analysis, we converted categorical features into numerical formats using label encoding. This step was crucial for features like `foot`, `role.name`, `shot_foot`, and `matchPeriod`, transforming textual data into numeric codes. This process enables our models to effectively interpret and learn from these categorical variables, enhancing the predictive accuracy of our Expected Goals (xG) model.

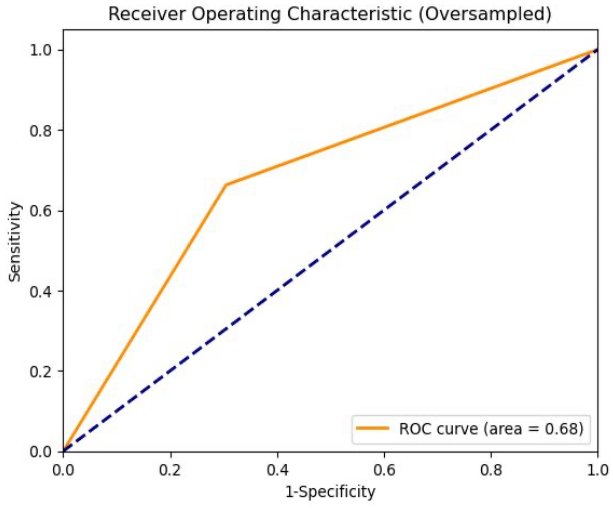
- Train Test Split

We split our dataset into training and testing sets to evaluate our model's performance. The features selected for (x) include player and team IDs, match period, event second, foot, height, weight, role, shot foot, shot distance, and shot angle. The target variable (y) is whether a shot results in a goal. Using a 70-30 split, we allocate 70% of the data for training and 30% for testing, with a `random_state` of 42 to ensure consistent results across runs. This step is essential for training and validating our Expected Goals (xG) prediction model.

Linear Regression

Metric	**Logistic Regression**	**Logistic Regression (Oversampled)**
Accuracy	0.8983	0.6919
Precision	0.6875	0.2027
Recall	0.052	0.6627
F1-score	0.0967	0.3104
ROC AUC	0.5246	0.679
Classification Matrix	[10827 30] [1203 66]	[7549 3308] [428 841]

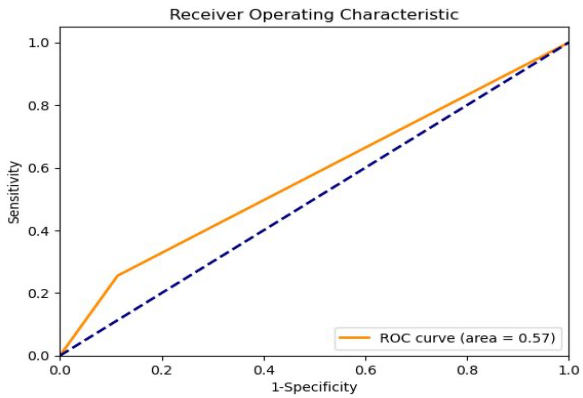


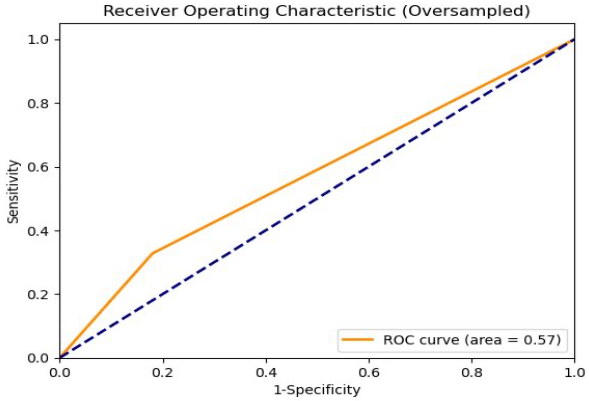


The performance metrics indicate that the original Logistic Regression model is highly accurate but struggles with recall, suggesting it misses many actual positive cases (goals). After oversampling, the model's ability to detect positive cases improves significantly, as seen in the increased recall and ROC AUC score. However, this comes at the expense of precision, showing that while the model identifies more true positives, it also incorrectly labels more negatives as positives. The increased F1-score in the oversampled dataset reflects a better balance between precision and recall, despite the overall decrease in accuracy.

Decision Tree

Metric	**Decision Tree**	**Decision Tree (Oversampled)**
Accuracy	0.8209	0.7679
Precision	0.2115	0.1758
Recall	0.2608	0.3302
F1-score	0.2336	0.2295
ROC AUC	0.5736	0.5746
Classification Matrix	[9627 1230] [945 324]	[8899 1958] [853 416]



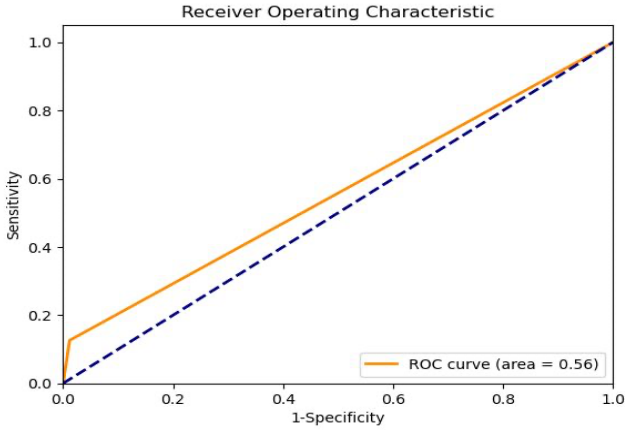


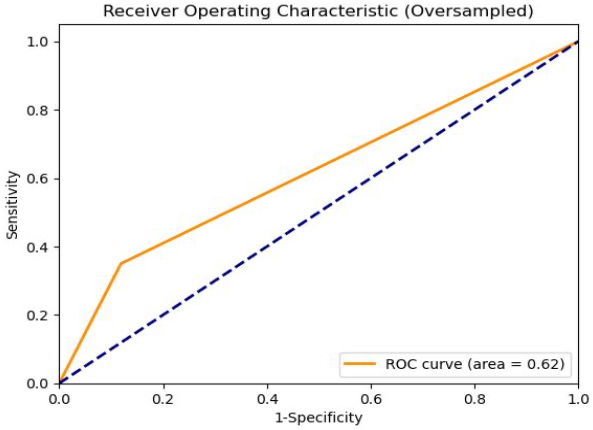
The Decision Tree model demonstrated respectable performance on the original imbalanced dataset with an accuracy of 82.09%. It managed to achieve a reasonable recall of 26.08%, indicating its ability to identify a fair number of actual positive instances (goals). Oversampling, intended to correct the class imbalance, resulted in only a slight improvement in recall (33.02%) but a decrease in accuracy (76.79%) and precision (17.58%). The F1-score and ROC AUC values show marginal differences, and there is a noticeable degradation in the R squared value, suggesting that the model did not improve substantially with oversampling. These results imply that the

Decision Tree model is robust enough to handle the imbalanced data without the need for oversampling, maintaining its performance metrics reasonably well.

Random Forest

Metric	Random Forest	Random Forest (Oversampled)
Accuracy	0.897	0.825
Precision	0.538	0.249
Recall	0.123	0.335
F1-score	0.2	0.286
ROC AUC	0.555	0.608
Classification Matrix	[10728 129] [1109 160]	[9566 1291] [825 444]



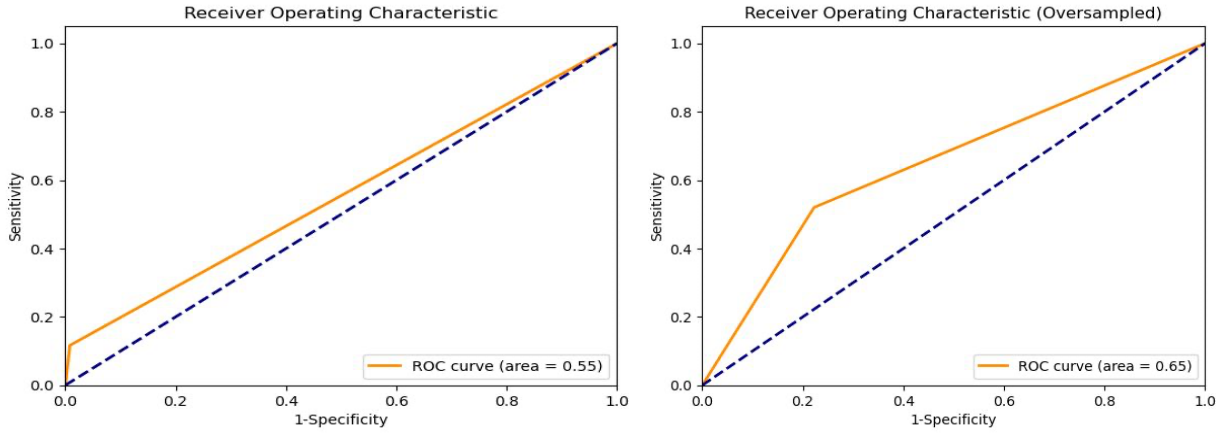


The Random Forest model on the imbalanced data yields a high accuracy of 89.7% and a moderate precision of 53.8%, but the recall is low at 12.3%. This indicates the model's tendency to miss true positive cases. After applying oversampling, the model shows an improved recall of 33.5%, suggesting a better capability to identify positive instances. However, this increase in recall comes at the expense of lower precision and accuracy.

The F1-score, which combines precision and recall, is slightly better in the oversampled model, and the ROC AUC also increases, indicating a better trade-off between true positive rate and false positive rate. However, the increased errors, as reflected in the RMSE, MSE, and MAE, along with a more negative R squared value, suggest the oversampled model may not provide a better fit to the data despite its improved recall.

Gradient Booster

Metric	Gradient Booster	Gradient Booster Oversampled
Accuracy	0.9002	0.7506
Precision	0.6245	0.2146
Recall	0.1166	0.5201
F1-score	0.1965	0.3039
ROC AUC	0.5542	0.6488
Classification Matrix	[10767 90] [1121 148]	[8442 2415] [609 660]



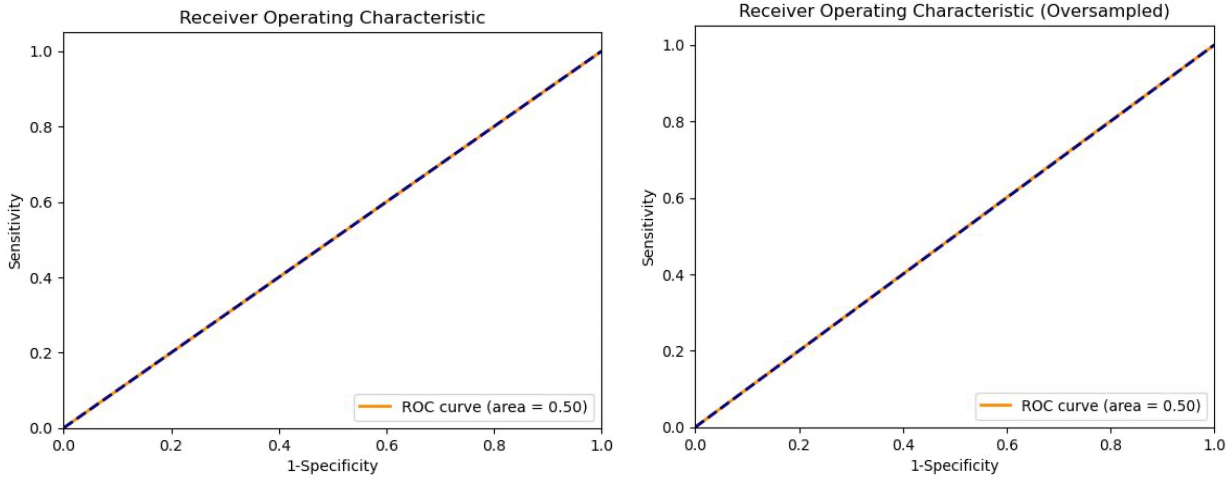
The Gradient Boosting model shows a high accuracy of 90.02% on the normal dataset and a decrease to 75.06% after oversampling. The recall saw a substantial increase from 11.66% to 52.01% with oversampling, indicating improved identification of true positives, albeit at the cost of precision, which dropped from 62.45% to 21.46%. The F1-score also improved after oversampling, showing a better balance between precision and recall when the model is exposed to a balanced dataset.

The ROC AUC score increased from 0.5542 to 0.6488 with oversampling, which means the model's ability to distinguish between the classes has improved despite a lower overall accuracy and precision. The model errors, as shown by RMSE, MSE, and MAE, are higher after oversampling, reflecting the increased misclassification rates. Negative R squared values for both datasets suggest that the model is not capturing all the variability in the data.

From the ROC curves, the area under the curve (AUC) is better in the oversampled model, which indicates a better trade-off between sensitivity and specificity compared to the non-oversampled model.

SVM

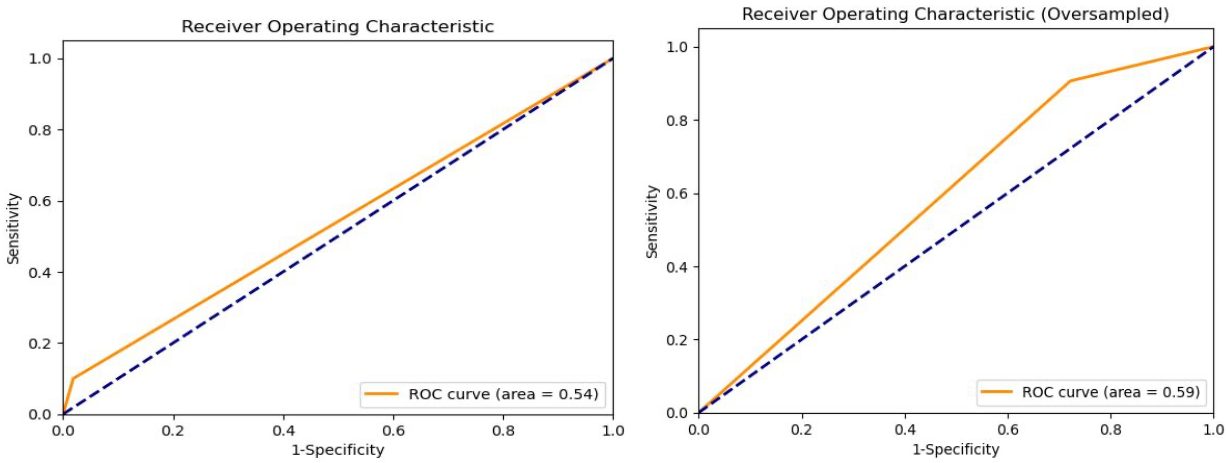
Metric	SVM	SVM (Oversampled)
Accuracy	0.895	0.305
Precision	0	0.105
Recall	0	0.749
F1-score	0	0.184
ROC AUC	0.5	0.501
Classification Matrix	[10857 0] [1269 0]	[2744 8113] [319 950]



The SVM model had high accuracy on the normal dataset but couldn't identify any true positives. Oversampling increased recall significantly, indicating better detection of positives but also led to many false positives, greatly reducing precision and accuracy. The ROC AUC scores remained around 0.5, suggesting the model was no better than random chance at distinguishing between classes, regardless of oversampling.

Neural Network

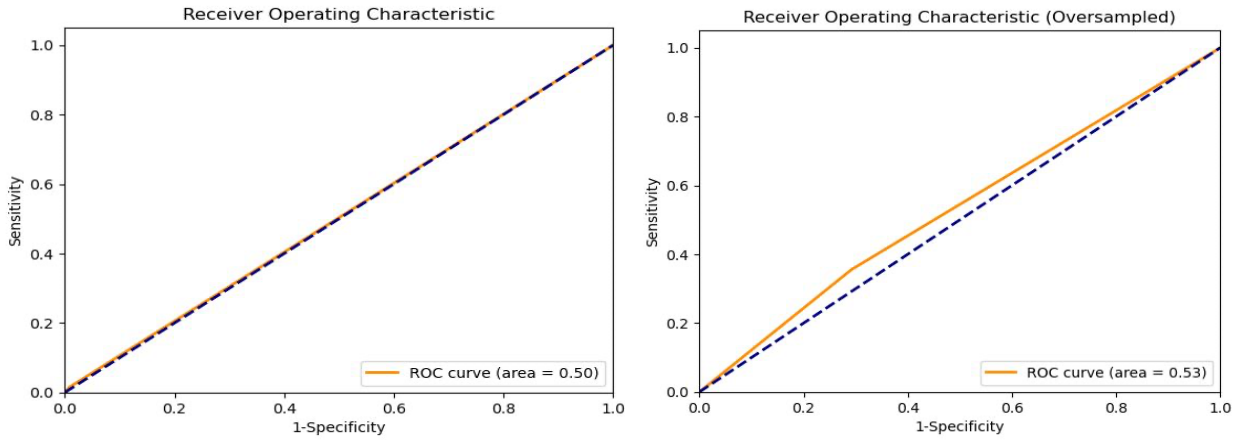
Metric	Neural Network	Neural Network (Oversampled)
Accuracy	0.86	0.783
Precision	0.294	0.226
Recall	0.24	0.443
F1-score	0.264	0.299
ROC AUC	0.586	0.633
Classification Matrix	[10656 201] [1142 127]	[3019 7838] [119 1150]



The Neural Network classifier achieved a reasonable balance of accuracy and precision on the normal dataset but showed limited ability to identify positive instances, with a recall of 0.24. When oversampled, the recall improved significantly to 0.443, suggesting better detection of true positives. However, the trade-off was a drop in precision and accuracy, and a rise in false positives, as typically seen with oversampling. The increase in the ROC AUC value to 0.633 indicates a better discriminative ability when the class distribution is balanced. The higher RMSE, MSE, and MAE in the oversampled data also reflect the increased prediction errors.

K-Nearest Neighbour

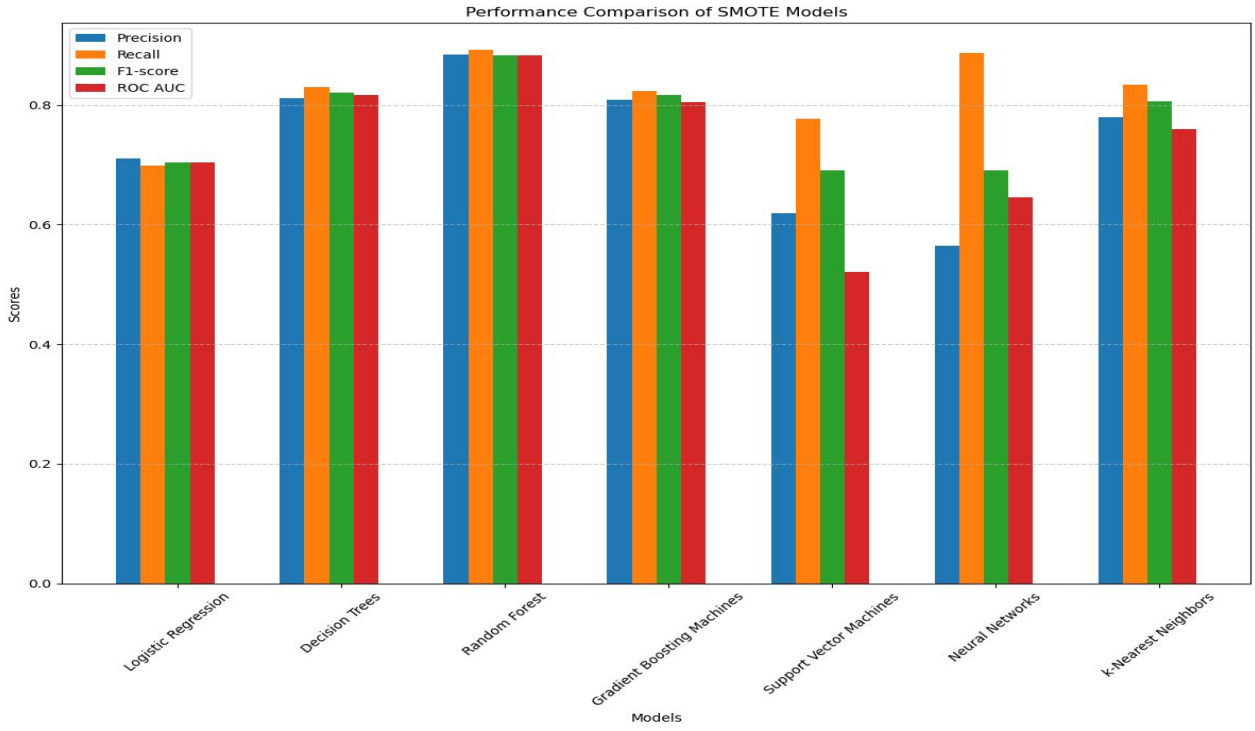
Metric	**KNN**	**KNN (Oversampled)**
Accuracy	0.8889	0.6712
Precision	0.1695	0.1246
Recall	0.0158	0.3554
F1-score	0.0288	0.1845
ROC AUC	0.5034	0.5318
Classification Matrix	[10759 98] [1249 20]	[7688 3169] [818 451]



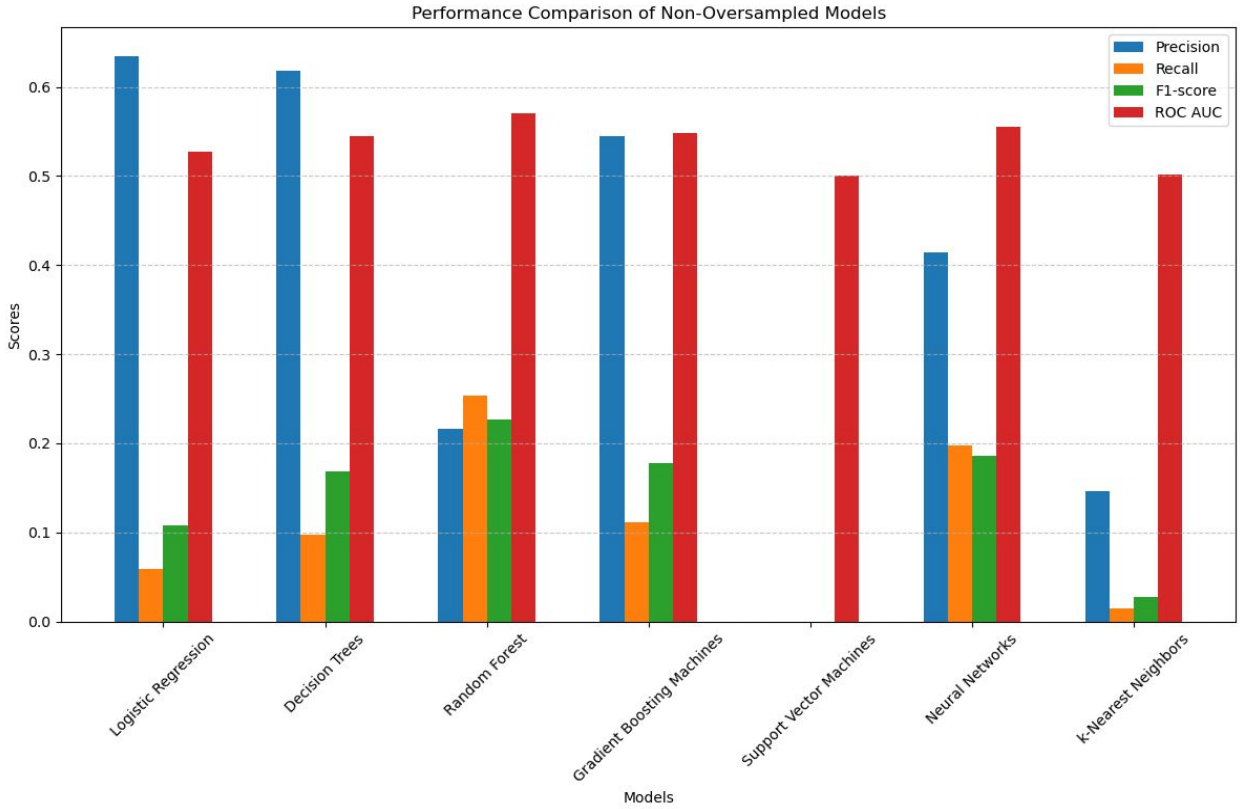
The KNN classifier on the standard dataset had an accuracy of 88.89%, but with very low recall. Oversampling increased recall but significantly decreased precision and accuracy. The ROC AUC shows slight improvement with oversampling. Error metrics increased and the R squared became more negative, indicating a worse fit for the oversampled data.

- **Cross Validation**

The two graphs present a performance comparison of different classifiers, with and without oversampling, based on 5-fold cross-validation.



In this graph (SMOTE models), most models show similar performance across precision, recall, F1-score, and ROC AUC, indicating a balanced outcome when the class distribution is even.



In this graph (Non-Oversampled models), there is a wider variance in performance. Precision is generally high across models, but recall varies significantly, leading to lower F1-scores. This suggests that while models can be precise, they struggle with sensitivity in an imbalanced dataset. ROC AUC values remain closer to 0.5, indicating models have difficulty distinguishing between classes without oversampling.

Overall, oversampling helps with recall at the cost of precision, whereas non-oversampled models tend to be more precise but with low recall.

Conclusion :-

Based on the comprehensive evaluation of various classifiers using precision, recall, F1-score, and ROC AUC as validation metrics across 5-fold cross-validation, the Random Forest classifier stands out as the most suitable model for our objectives. The decision to select Random Forest is informed by its relatively consistent performance across both oversampled and non-oversampled datasets.

Random Forest demonstrates a solid balance between accuracy and the ability to identify true positives (recall), which is crucial for our predictive modeling. While the recall increases significantly with oversampling, the precision, though reduced, remains acceptable, indicating a good balance between sensitivity and specificity.

The F1-score and ROC AUC values suggest that Random Forest is relatively robust in differentiating between the classes compared to other models, making it the preferred choice in our context.

Considering these findings, we conclude that the Random Forest classifier provides the best compromise between detecting as many positive instances as possible while maintaining a reasonable level of precision, a critical aspect for our Expected Goals (xG) model's deployment.

7. Project Results

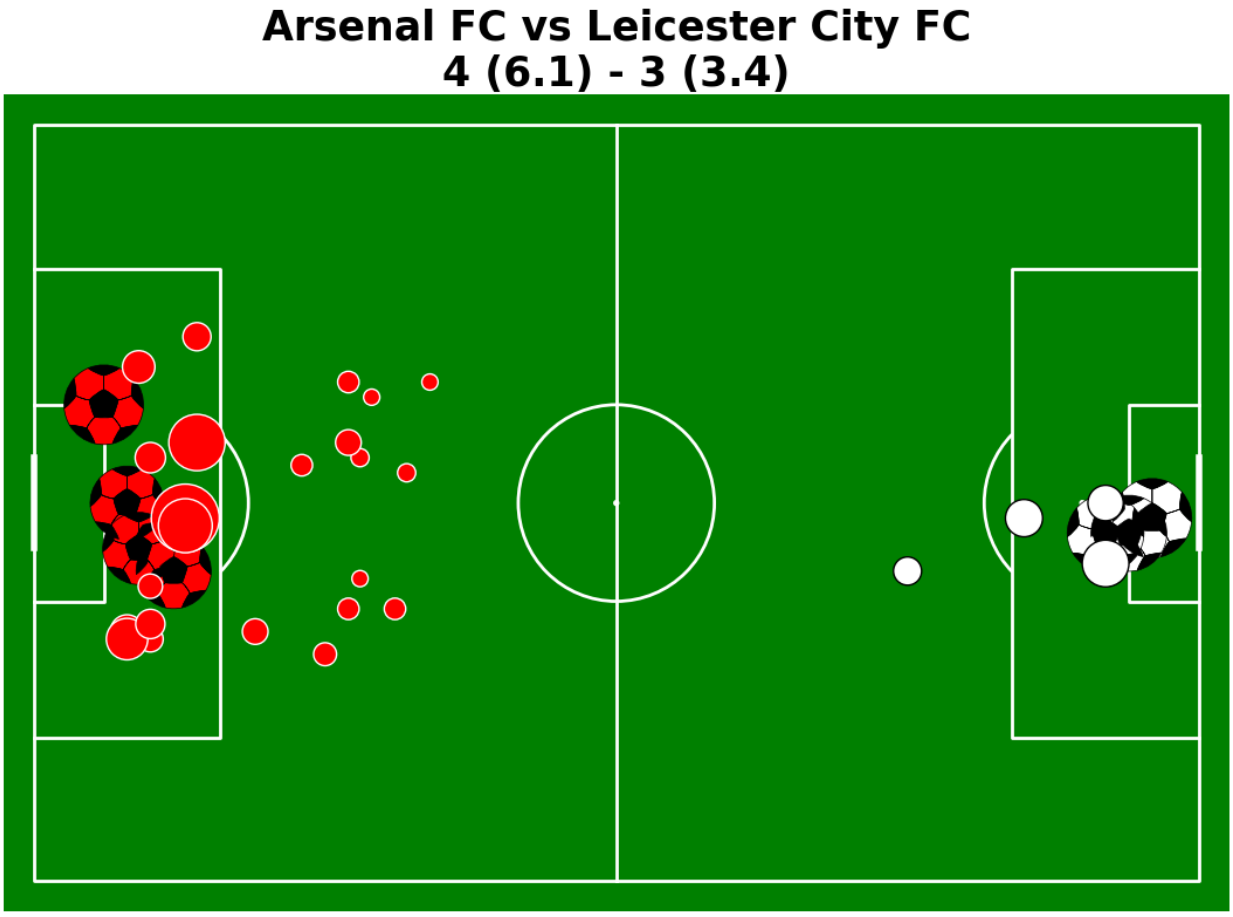
Using the Decision Tree model that we selected after cross validation, we will calculate the predictive probability to find the xG:

```
merged_data["xG"] = random_forest_oversampled.predict_proba(merged_data[xCols])[:, 1]
```

Using the calculated xG we will plot football shot chart for few selected matches and find the details of the goal scorers. For example, considering match no. 2499719 between Arsenal FC vs Leicester City FC.

Team 1 Goals:
Minute: 1, Scorer: Alexandre Lacazette, Shot Type: 1, Shot Distance: 15.0, Shot Angle: 24.67, xG: 0.86
Minute: 46, Scorer: Daniel Mensah Welbeck, Shot Type: 1, Shot Distance: 8.0, Shot Angle: 53.13, xG: 0.83
Minute: 37, Scorer: Aaron Ramsey, Shot Type: 1, Shot Distance: 14.32, Shot Angle: 14.25, xG: 0.97
Minute: 39, Scorer: Olivier Giroud, Shot Type: 0, Shot Distance: 10.82, Shot Angle: 35.48, xG: 0.79

Team 2 Goals:
Minute: 4, Scorer: Shinji Okazaki, Shot Type: 0, Shot Distance: 4.47, Shot Angle: 82.87, xG: 0.96
Minute: 28, Scorer: Jamie Vardy, Shot Type: 1, Shot Distance: 7.21, Shot Angle: 53.13, xG: 0.88
Minute: 10, Scorer: Jamie Vardy, Shot Type: 0, Shot Distance: 8.94, Shot Angle: 45.0, xG: 0.9

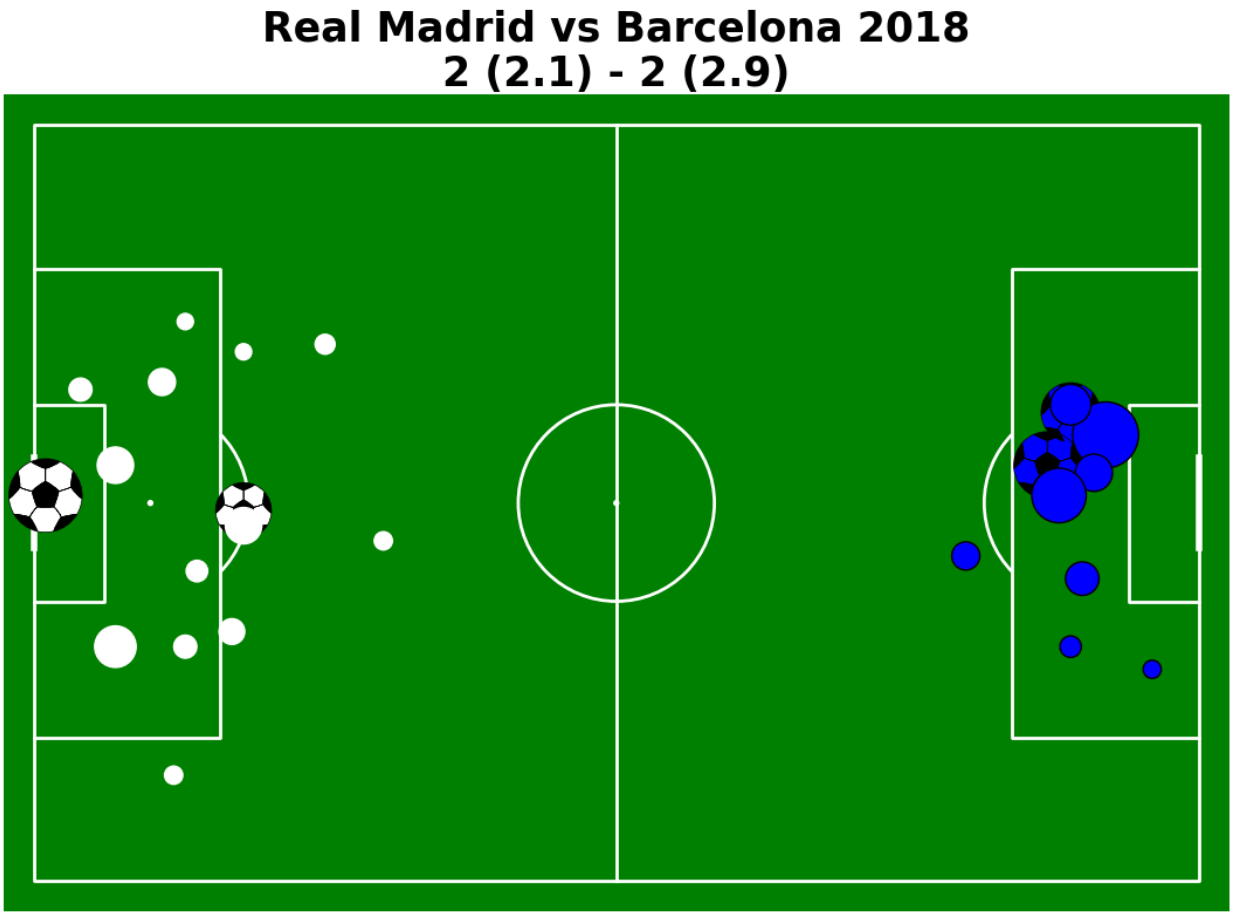


The score mentioned in brackets is the xG of the match for each team, and the actual score of the game. The plot shows the short chart where the size of the plots is based on the xG of each shot. The shots that eventually went in is marked using football marker.

Similarly for the match no. 2565907, which was a classic rivalry match between Spanish giants Real Madrid and Barcelona in 2018, is plotted below.

Team 1 Goals:
Minute: 14, Scorer: Cristiano Ronaldo, Shot Type: 1, Shot Distance: 1.41, Shot Angle: 150.26, xG: 0.82
Minute: 26, Scorer: Gareth Frank Bale, Shot Type: 1, Shot Distance: 18.03, Shot Angle: 24.99, xG: 0.46

Team 2 Goals:
Minute: 9, Scorer: Luis Alberto Suárez Díaz, Shot Type: 1, Shot Distance: 16.28, Shot Angle: 19.46, xG: 0.52
Minute: 6, Scorer: Lionel Andrés Messi, Shot Type: 1, Shot Distance: 13.93, Shot Angle: 30.3, xG: 0.68



These plots can be plotted for all the matches in the where shot information is available. xG data can further be used to track the goal scoring ability of individual players and teams to understand the performance of these players and teams. This information can then be used by betting companies, and analytical companies to predict the performance of these players.

8. Key Findings and Deliverables:

1. Predictive Model Selection: After thorough evaluation, the Random Forest classifier emerged as the most suitable model for predicting Expected Goals (xG) for football shots. Its balanced performance in terms of accuracy, precision, recall, F1-score, and ROC AUC makes it well-suited for the task.
2. Feature Engineering: The project successfully extracted and engineered relevant features such as shot distance, shot angle, player attributes, and match context. These features played a crucial role in training the predictive models and enhancing their performance.

3. Visualization and Interpretation: Various visualizations, including shot maps, histograms, and correlation matrices, provided valuable insights into shot distribution, spatial patterns, and correlations between variables. These visualizations aided in understanding the underlying dynamics of football shots and informed feature selection and model interpretation.
4. Model Evaluation: Through rigorous evaluation using metrics like precision, recall, F1-score, and ROC AUC, the project assessed the performance of different machine learning models. This evaluation process helped in selecting the most suitable model and understanding its strengths and weaknesses.
5. xG Calculation and Visualization: The project successfully calculated xG values for individual shots and visualized them on football shot charts for selected matches. This allowed for a detailed analysis of shot outcomes and the identification of key goal-scoring opportunities.

9. Impact of the Project Outcomes:

1. Enhanced Performance Analysis: The predictive models developed in this project provide football teams, coaches, and analysts with a powerful tool for analyzing shot outcomes and predicting goal probabilities. By leveraging xG predictions, teams can assess player and team performance, identify areas for improvement, and develop strategic plans to enhance goal-scoring efficiency.
2. Improved Decision Making: The insights generated from the xG predictions enable more informed decision-making in various contexts, such as player recruitment, tactical planning, and in-game strategy adjustments. By understanding the likelihood of goals from different positions and scenarios, teams can optimize their resources and maximize their chances of success on the field.
3. Data-Driven Insights: The project's findings contribute to the growing field of sports analytics, providing stakeholders with data-driven insights into the factors influencing goal outcomes in football. These insights not only enhance performance analysis but also foster innovation and advancement in sports science and technology.
4. Commercial Applications: Beyond the realm of professional football, the predictive models developed in this project have potential applications in betting, fantasy sports, sports media, and

fan engagement. By providing accurate predictions of goal probabilities, these models add value to various commercial ventures and enhance the overall fan experience.

Overall, the project's outcomes have significant implications for the football industry and sports analytics more broadly. By leveraging data mining techniques and predictive modeling, the project has created tangible value by empowering stakeholders with actionable insights and facilitating data-driven decision-making in the dynamic and competitive world of football.