دانشگاه فردوسی مشهد
Ferdowsi University of Mashhad

Regression Project – Expected Goal

[**Department**: Department of Industrial Engineering]

[**Students:** Amirali Khatib; Amirali Bagherzadeh]

[**Student id:** 4032315014; 4031315278]

[**Professor:** Dr. Alireza Shadman]

Summer 2025

Contents

Abstract 1
1. Introduction 1
2. Related work 2
3. Proposed framework 2
4. Experiments 3
4.1. Dataset 3
4.2. Feature Engineering 4
4.2.1. Goal distance 4
4.2.2. Shot angle 4
4.2.3. Number of opponents and teammates separately in a shot's triangular area 5
4.3. Exploratory Data Analysis 6
4.3.1. Response Variable 6
4.3.2. Numerical Features 7
4.3.3. Categorical Features 9
4.4. Feature selection 9
4.4.1 Forward Stepwise Selection 10
4.4.2 Principal Component Analysis 10
4.4.3. Partial Least Squares 10
4.5. Methodology 10
4.5.1. K-Nearest Neighbors 10
4.5.2. Bagging 11
4.5.3. Random Forest 11
4.5. Hyperparameter Tuning 12
4.6. Final Result 13
5. Conclusion and Future works 14
Appendix .... a
Appendix 1 .... a
Appendix 2 .... a
Appendix 3 .... b
Appendix 4 c
Appendix 5 .... d
References .... e

Figures

Figure 1- Hudl-Statsbomb logo III
Figure 2- Difference between our xG model and the one for Hudl-Statsbomb 3
Figure 3 - Illustration for better understanding of the shot angle's trigonometry 4
Figure 4 – Vertices of a shot triangular zone 5
Figure 5 - Distribution of xG 6
Figure 6 - Distribution of Numeric Features 7
Figure 7 - Scatter plot of Numerical Features by xG 8
Figure 8 - Correlation plot for numerical features of xG dataset 8
Figure 9 - Frequency of classes for categorical variables 9
Figure 10 - Violin plot of categorical features by xG 9
Figure 11 - xG Forward Stepwise Selection 10
Figure 12- Illustration of Bagging Ensemble Method 11
Figure 13- Illustration of Random Forest Ensemble Method 12
Figure 14 – 10-Fold Cross Validation workflow 12
Figure 15 - Illustration of the work flow of training SVM (Reduced 5-Fold Cross Validation) 13
Figure 16- A screenshot from the raw form of the event data .... a
Figure 17 - xG components explained variance in PCA method .... b
Figure 18 - xG components MSE in PLS method .... b

Tables

Table 1- size of data set 3
Table 2 – xG's data description 3
Table 3 – xG's final data description 6
Table 4 - xG dataset's summarization of response variable 6
Table 5 – xG dataset's summarization of numerical features 7
Table 6- Results of top 3 models 14
Table 7- Models' Hyperparameter Space in xG .... b
Table 8- Hyperparameter tuning results in xG c
Table 9- Final result of predicting test in xG .... d

Preface

In recent years, the exponential growth of data has led to a transformative shift in the technological landscape. Business owners have allocated significant budgets to data-driven decision-making strategies to improve their organizations' efficiency and productivity. Football, as an industry, is known for its substantial financial turnover. Many renowned companies actively serve as sponsors for football clubs. So, every team try to be successful in their matches to satisfy its fans and sponsors. For this purpose, they require approaches to understand their opponents' playing strategy to make the most efficient efforts.

Data analytics has become an indispensable tool for football clubs, enabling them to analyze opponents' playing styles and performance statistics effectively. Put all these facts together, help us realize the main reason that football teams recruit football data analyst. Alternatively, many of these clubs outsource the data gathering and analytics procedures to a second company by signing a contract.

One of the most recently famous companies in the field of football data analytics is **Hudl Statsbomb** (Founded in 2017). It is a leading sports analytics company specializing in advanced football data and analytics tools. It provides detailed event data with over 3,400 events per match and unique metrics like pressures, On-Ball Value (OBV), and xG (expected goals). Their flagship platform, **Statsbomb IQ**, helps clubs, leagues, and organizations in player scouting, opposition analysis, and strategic decision-making. With over 280 professional clients worldwide, Statsbomb plays a crucial role in modernizing football analytics.

hudlstatsbomb

Figure 1- Hudl-Statsbomb logo

Huld-statsbomb, fortunately, has provided considerably large open free event-data which has encouraged many passionate researchers around the world to run a study on the basis of this available event-data.

Primarily, this project aims to conduct a study using Hudl-Statsbomb event data. Estimation of the xG measure is what we aim for throughout this project.

xG is measured by using surprisingly regression approaches to directly predict the probability of shot success concerning many factors (will be discussed in the sections ahead).

Regression Project – Predicting Expected Goal

Abstract

Predicting the probability that a football shot will result in a goal—known as Expected Goals (xG)—is a critical task for performance analysis and decision support in modern football analytics. Given the limitations of the available dataset and the absence of key variables, we devoted significant effort to feature engineering. In this project, we develop a regression-based xG model by combining shot-specific features (e.g., distance, angle, shot type), contextual variables (e.g., game state, player status). We cleaned and normalized the dataset, then split it 80/20 into training and testing sets. We performed hyperparameter tuning for each model using a k-fold cross-validation approach. Model performance is assessed via mean squared error (MSE), and R-squared (R^2) on test data. Also, we used a new metric to check xG prediction interval. Experimental results demonstrate that tree-based models have the best performance. The top-performing regressor was Bagging, which achieved to R^2 of 0.878.

1. Introduction

In the domain of football analytics, the xG metric has become one of the most important tools for evaluating team and player performance. This metric represents the probability of a shot resulting in a goal, considering factors such as shot location, angle, distance from the goal, shot type (i.e. header, direct shot, and etc.), the level of defensive pressure. xG helps coaches and analysts assess not only the final outcome of a game but also the quality of chances created and the efficiency of teams and players on the pitch. It provides deep insights into the efficiency of attacking plays and highlights areas for improvement.

The concept of Expected Goals (xG) has evolved significantly over time, driven by advancements in statistical modeling and machine learning techniques. While the exact origins of the xG metric are unclear, early references suggest that its roots may lie in studies related to shot outcomes in other sports, such as ice hockey (Macdonald, 2012). Green (2012) also contributed to the early discussions surrounding the metric. The xG statistic fundamentally represents a classification problem, assessing the probability of a shot resulting in a goal, which necessitates the application of various machine learning and statistical methods (Bandara et al., 2024).

2. Related work

Initially, xG models relied heavily on logistic regression and simple statistical analyses, focusing primarily on shot characteristics such as location and distance from the goal (Goddard, 2005). These early models laid the groundwork for understanding scoring probabilities, with findings indicating that the number of attempted shots correlates with the number of goals scored (Brechot & Flepp, 2020). Subsequent studies confirmed that winning teams typically attempt more shots than losing teams, reinforcing the importance of shot volume in performance evaluation (Bandara et al., 2024).

Recent literature has explored various directions to enhance the accuracy of xG models. Studies such as Rathke (2017) and Spearman (2018) confirmed the importance of spatial features (distance and angle), while Brechot and Flepp (2020) emphasized the value of contextual variables like shot type and match situation. Other researchers have attempted to incorporate more sophisticated inputs, such as player ability—often using proxies like FIFA ratings (Eggels et al., 2020)—or match importance metrics based on league dynamics (Bedford & Schembri, 2018).

As the field progressed, researchers began to incorporate more sophisticated modeling techniques, including gradient boosting, neural networks, and support vector machines (Herbinet, 2018).

3. Proposed framework

In this report, we aim to investigate and predict xG values using machine learning models, but there is a main difference between our xG models and the other typical ones. Commonly, xG models are designed on the basis of classification algorithms while we aim to use regression models on the case study. The main reason behind this seemingly irrational decision is that we have xG-score for each shot in our features. Hence, there is no need to train a binary classification on the instances with GOAL or NO GOAL labels.

Be aware of the fact that we want to fit our proposed regression model on the instances within which the xG-score is already exist. These xG-scores are computed by Hudl-Statsbomb’s binary classification model. In addition to the available features in our given dataset, they also have access to higher-level data which is inaccessible for us. They store this data from the players’ GPS tracking vests and the computer vision instruments that record factors such as ball speed, ball spin, and the other factors at the moment. As we illustrate the difference between our proposed workflow and their model in Figure 2, we take their sigmoid resulted probability as target variable xG-score.

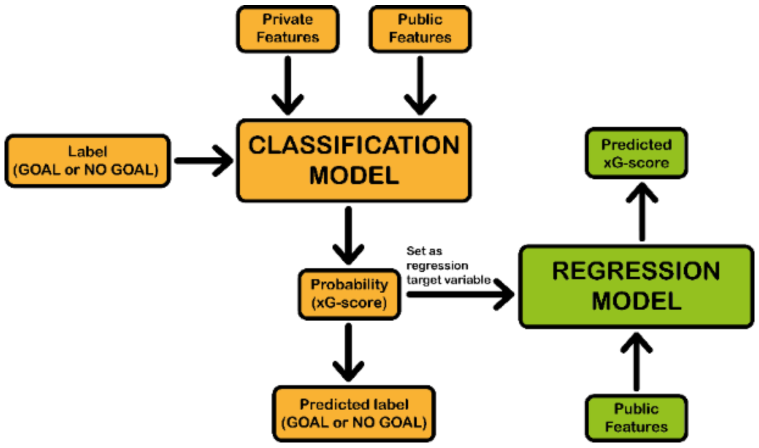


Figure 2- Difference between our xG model and the one for Hudl-Statsbomb

4. Experiments

In this section, we discuss the experimental procedure.

4.1. Dataset

The size of the dataset is reported in Table 1. The dataset contains no missing values.

Table 1- size of data set

Data Set	Size
xG	87,111

The data gathering process is detailed in Appendix 1. Our collected dataset contains the shot event data of the more than 3000 football matches. Before going through more detailed steps, we describe our initial dataset's features in Table 2 so as to get an initial understanding from it.

Table 2 – xG's data description

Feature name	Description	Data type
Play-pattern	Name of the play pattern relevant to this event (i.e. Regular Play, From Corner, From Free Kick, and etc.)	Categorical
Start-loc	The x and y coordinates at which the shot is executed.	List
Under-Pressure	Indicating whether action was performed while being pressured by an opponent.	Boolean
Follows-dribble	Indicating whether the shot comes after a successful dribble by the shooter.	Boolean
First-time	Indicating whether the shot is the player's first touch	Boolean
Technique	Name of the technique used for the shot.	Categorical
Body-part	Name of the body part used for the shot.	Categorical
Open-goal	Indicating whether the goalkeeper is in the goal or not.	Boolean
Shot-type	Name for the attribute option specifying the type of shot.	Categorical
Deflected	Shot was redirected by another player's touch.	Boolean
Teammate360	The list of the cartesian coordinates of the surrounding teammates	Nested list
Opponent360	The list of the cartesian coordinates of the surrounding opponents	Nested list

Feature name	Description	Data type
xG	The Hudl-Statsbomb's xG-score calculated for the shot.	Numerical

4.2. Feature Engineering

In order to improve our model performance, it is needed to develop alternative features. We take the advantages of 360 freeze frame of the players' positional status per event and the other given coordinates to develop the new features. The sub-sections below introduce those features and the approaches we followed to generate them.

4.2.1. Goal distance

The first feature that can be derived is the goal distance. Goal distance indicates the distance between the shot executer's position at the time of a shot and the center of the goal. This distance is a critical feature in football analytics as it significantly influences the likelihood of scoring. It is calculated by using the Euclidean distance between the shot's coordinates and the coordinates of the goal center.

$$\text{Goal distance} = \sqrt{\left(X_{shot} - X_{goal\ center}\right)^2 + \left(Y_{shot} - Y_{goal\ center}\right)^2}$$

4.2.2. Shot angle

The second thing to consider is the view that the player has of the goal or the shot angle. This feature is so critical since the more you can see, the better your chance of scoring.

We calculated the shot angle based on **Cosine Law Formula**. Considering Figure 3, we calculate the vectors of the player's distance from both goal posts and the distance between the goal posts that are respectively equal to $\vec{a}$, $\vec{b}$, and $\vec{c}$. Here is the formula of calculating the angle based on the previously mentioned law.

$$\theta = Arccos\left(\frac{\|\vec{a}\|^2 + \left\|\vec{b}\right\|^2 - \|\vec{c}\|^2}{2\|\vec{a}\|\left\|\vec{b}\right\|}\right)$$

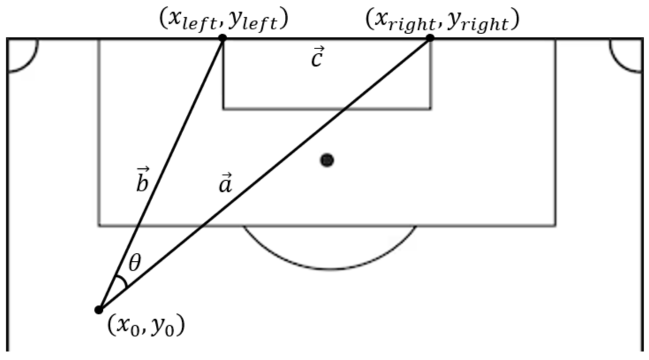


Figure 3 - Illustration for better understanding of the shot angle's trigonometry

4.2.3. Number of opponents and teammates separately in a shot's triangular area

As it is obvious, there is a triangular area that explain the vision of the player behind the shot. And also scattered points in this triangular area represent the players between the ball and the goal. Counting the number of players either teammates or opponents provides additional features for our regression model.

In order to realize whether a player is present in the shot event's corresponding triangular zone, we go through a linear algebraic procedure named **Sign rule method**. Initially, take the Figure 4 and its notations into consideration.

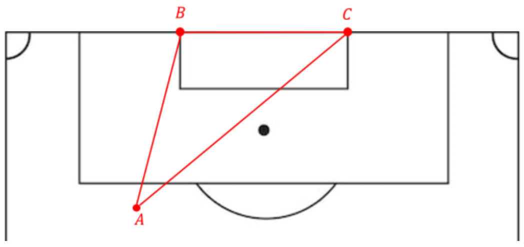


Figure 4 – Vertices of a shot triangular zone

As you see in the Figure 4, we take the vertices A, B, and C respectively as the shot location, left goal post, and the right goal post. Additionally, we should define a point as P to check if it lies on the zone. To determine that, we use Sign rule method in the following two steps below.

1. For each triangle edge ($A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$), calculate the cross product of vectors:
   - $v_1 = B - A;\ v_2 = P - A$
   - $v_3 = C - B;\ v_4 = P - B$
   - $v_5 = A - C;\ v_6 = P - C$

   Cross products:

   $$z_1 = v_1 \times v_2\ ;\ z_2 = v_3 \times v_4\ ;\ z_3 = v_5 \times v_6$$

2. All the z_i should have the same sign (all positive or all negative) for P to lie inside the triangle. If a z_i differs in sign, P lies outside.

Finally, for every instance in our dataset, we count separately the number of opponents and teammates which are present in the zone.

After augmenting our data set by adding four new features, you can see the resulting features here in Table 3.

Table 3 – xG's final data description

Feature name	Description	Data type
Play-pattern	Name of the play pattern relevant to this event (i.e. Regular Play, From Corner, From Free Kick, and etc.)	Categorical
Under-Pressure	Indicating whether action was performed while being pressured by an opponent.	Boolean
Follows-dribble	Indicating whether the shot comes after a successful dribble by the shooter.	Boolean
First-time	Indicating whether the shot is the player's first touch	Boolean
Technique	Name of the technique used for the shot.	Categorical
Body-part	Name of the body part used for the shot.	Categorical
Open-goal	Indicating whether the goalkeeper is in the goal or not.	Boolean
Shot-type	Name for the attribute option specifying the type of shot.	Categorical
Deflected	Shot was redirected by another player's touch.	Boolean
Number of teammates in triangular zone	The list of the cartesian coordinates of the surrounding teammates	Numerical
Number of opponents in triangular zone	The list of the cartesian coordinates of the surrounding opponents	Numerical
Goal distance	Distance between shot executer and the center of the goal.	Numerical
Shot angle	Using Cosine law to calculate the angle between the shot executer and the goal posts.	Numerical
xG	The Hudl-Statsbomb's xG-score calculated for the shot.	Numerical

4.3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) uncovers patterns, trends, and relationships in data. It focuses on understanding data, identifying key features, and examining variable interactions. By visualizing distributions and correlations, EDA offers insights for further analysis and model development. This section discusses the features and their distributions.

4.3.1. Response Variable

First, Response variable must be analyzed. Table 4 shows the summarization of xG.

Table 4 - xG dataset's summarization of response variable

Feature	mean	std	min	25%	50%	75%	max
xG	0.106297	0.149382	0.00018	0.027673	0.05479	0.11006	0.995122

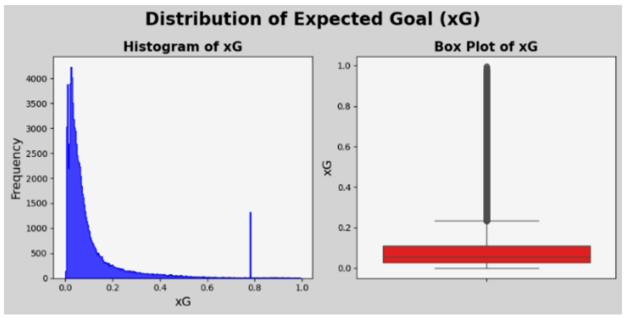


Figure 5 - Distribution of xG

Based on the displayed chart in Figure 5, it is observed that the average xG is approximately 0.13. Additionally, most of the xG values are below 0.2, which is clearly evident from the box plot, where values greater than around 0.22 are identified as outliers. Furthermore, the histogram shows a right skew, with most xG values being small and close to 0.

4.3.2. Numerical Features

The Numerical Features of the datasets are summarized and visualized in Table 5 and Figure 6.

Table 5 – xG dataset's summarization of numerical features

Feature	mean	std	min	25%	50%	75%	max
Goal-distance	19.20253	8.752675	0.4	12.01062	18.4456	25.5256	92.80
angle	25.38471	15.78733	0	15.0218	19.7265	31.4640	180
teammates in triangle	0.184466	0.474959	0	0	0	0	5
opponents in triangle	1.709841	1.053728	0	1	1	2	11

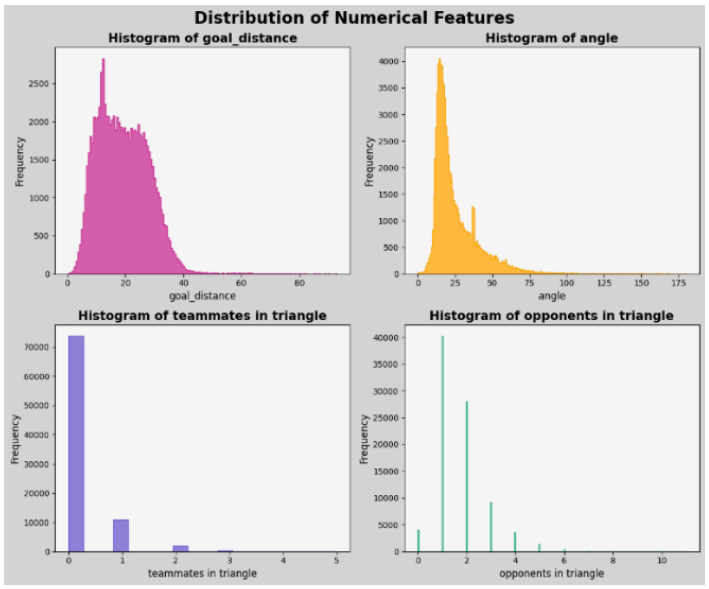


Figure 6 - Distribution of Numeric Features

The provided histograms reveal the patterns in the data. Most features exhibit skewness, with goal distance and shot angel being positively skewed, indicating that the majority of values cluster at lower ranges, while a few outliers extend into higher ranges. In addition, the distribution of teammates in triangle is heavily left-skewed, suggesting that most observations have no teammates in the triangle, reflecting limited offensive support in those instances. opponents in triangle shows a more balanced distribution but still leans slightly toward lower values, highlighting the concentration of defensive players near the action. Figure 7 also displays the relationship between numerical variables and the response variable xG.

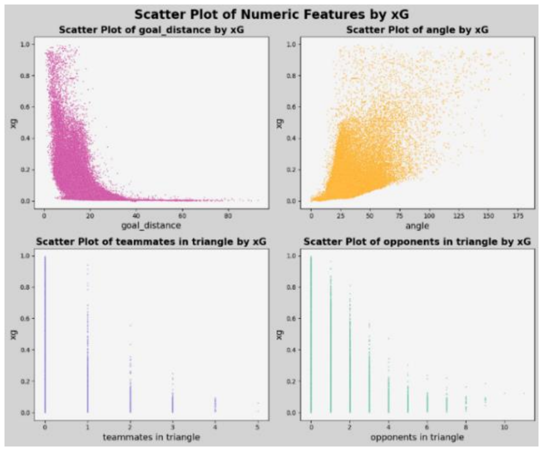
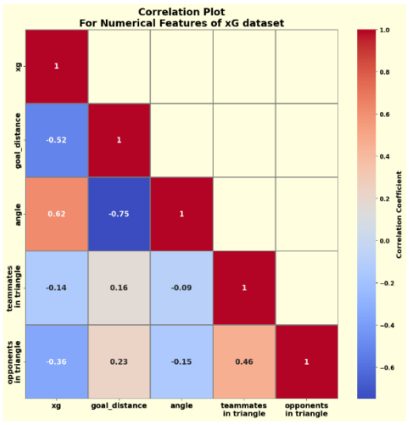


Figure 7 - Scatter plot of Numerical Features by xG

It seems that these two variables follow a quadratic relationship. Additionally, for the variable shot angle, it is evident that as the value of angle increases, the value of xG also increases. This observation is logical, as a higher angle indicates a better position, leading to a higher chance of scoring. It is also observed that for the variables of teammates in triangle and opponents in triangle, as these values increase, the value of xG decreases.

Another analysis that is so inevitable in this section is discussing the correlation between numerical features. The correlation heatmap is provided in Figure 8.



	xg	goal_distance	angle	teammates in triangle	opponents in triangle
xg	1				
goal_distance	-0.52	1			
angle	0.62	-0.75	1		
teammates in triangle	-0.14	0.16	-0.09	1	
opponents in triangle	-0.36	0.23	-0.15	0.46	1

Figure 8 - Correlation plot for numerical features of xG dataset

According to Figure 8, angle has the most intensive correlation with the response variable xG, and its value is positive. It shows the essence of angle in numerical features. Following that, goal-distance has the next most intensive correlation with the response variable, but it is negative. A positive correlation means that as one variable increases, the other also tends to increase, while a negative correlation indicates that as one variable increases, the other tends to decrease.

Generally speaking, it is ideal for us to have independent features which have lowest absolute value of correlation with each other while having high absolute value of correlation with the target variable.

4.3.3. Categorical Features

In this section, we aim to explore of categorical variables and see their relationships with the response variable. These are provided in Figure 9 and Figure 10.

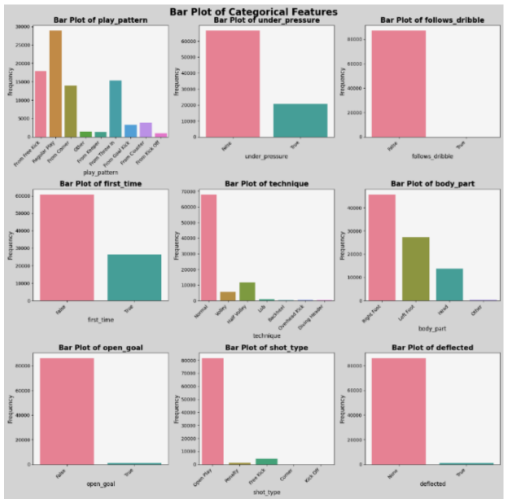


Figure 9 - Frequency of classes for categorical variables

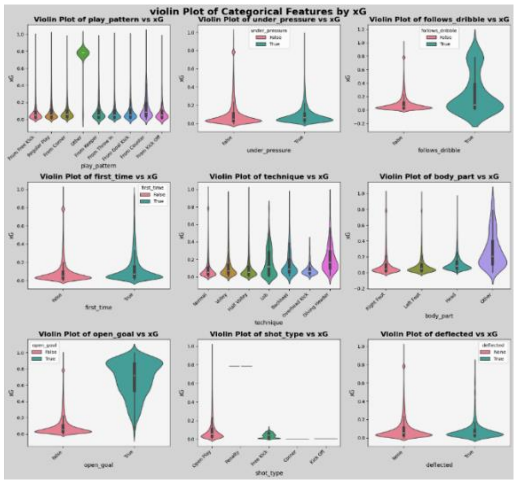


Figure 10 - Violin plot of categorical features by xG

4.4. Feature selection

Choosing a fewer number of features from the initial set of features will improve the generalization of the model while giving the same or higher performance. We implement three feature selection strategies (FSS, PCA, and PLS) in subsections ahead. We, optionally, decided choose forward stepwise selection strategy as our alternative feature selection approaches beside choosing all the features.

4.4.1 Forward Stepwise Selection

We decided to take the features obtained until the 16th step of Forward Stepwise algorithm. As you can see in Figure 11, There is no significant difference between the resulting metrics in this step and the ones ahead. There are also the selected feature names on the RSS variation plot.

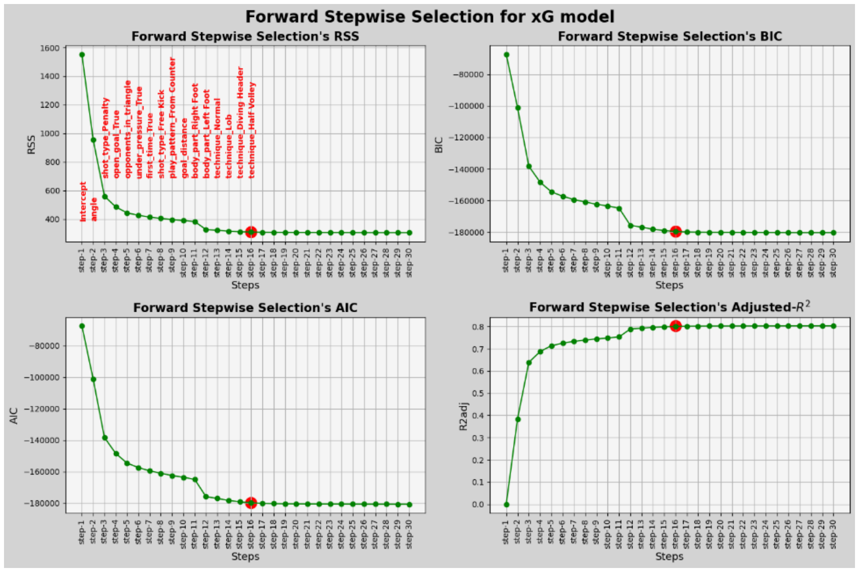


Figure 11 - xG Forward Stepwise Selection

4.4.2 Principal Component Analysis

Given the report's length constraints, the section has been moved to Appendix 2.

4.4.3. Partial Least Squares

Given the report's length constraints, the section has been moved to Appendix 2.

4.5. Methodology

In this section, we focused on providing the brief theorical explanation of further three outperforming models in the result section.

4.5.1. K-Nearest Neighbors

The K-Nearest Neighbors (KNN) Regressor is a simple yet powerful non-parametric machine learning algorithm used for predicting continuous outcomes. Unlike traditional regression models that learn a fixed function from the training data, k-NN works by identifying the k closest data points (neighbors) to a given input based on a distance metric, typically Euclidean distance. The predicted value for the input is then calculated as the average (or sometimes weighted average) of the target values of these neighbors. Because it relies directly on the training data for predictions, k-NN is considered a lazy learner, meaning it has no explicit training phase and stores the entire dataset. While it is intuitive and easy to implement, k-NN regression can be sensitive to the choice of k, the distance metric, and the presence of noisy or high-dimensional data. In the next section, which deals with hyperparameter tuning, the best choice of k should be determined.

4.5.2. Bagging

The Bagging Decision Tree Regressor is an ensemble learning technique that improves the performance and stability of regression models by combining multiple decision trees through a method called bootstrap aggregating, or bagging. In this approach, several decision trees are trained independently on different random subsets of the original training data, generated with replacement. Each tree learns to predict the target variable, and the final prediction is obtained by averaging the outputs of all the trees. This process helps to reduce variance and mitigate overfitting, which are common issues with individual decision trees. The Bagging Regressor is particularly effective when using high-variance base models like decision trees and performs well on a wide range of regression tasks, offering better accuracy and robustness than a single tree. We depicted this model's common training strategy in Figure 12.

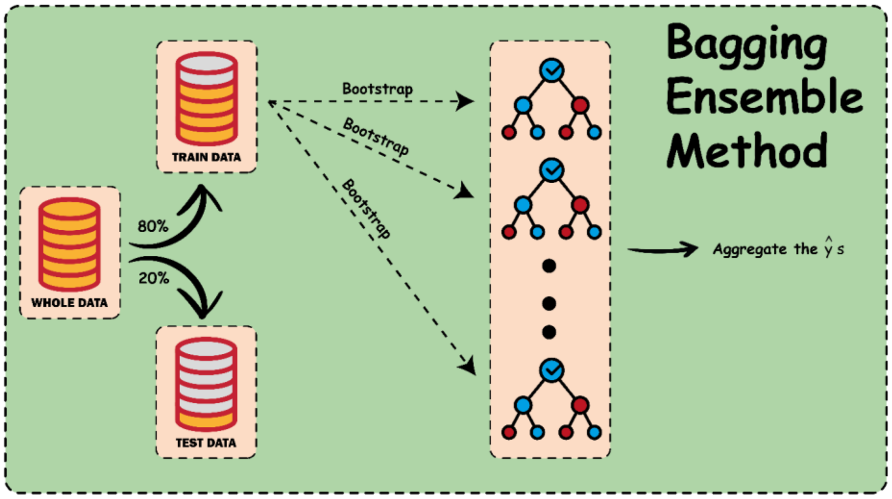


Figure 12- Illustration of Bagging Ensemble Method

4.5.3. Random Forest

The Random Forest Regressor is an ensemble learning method used for predicting continuous outcomes by combining the predictions of multiple decision trees. It builds a forest of decision trees during training, where each tree is trained on a random subset of the data and a random subset of features, a technique known as bootstrap aggregating or bagging. During prediction, the Random Forest takes the average of the outputs from all individual trees to produce a final result, which helps reduce overfitting and improve generalization compared to a single decision tree. This method is robust to noise, handles both linear and nonlinear relationships well, and is capable of capturing complex patterns in the data. However, it can be computationally intensive and less interpretable than simpler models. Here, in Figure 13, the Random Forest training strategy is depicted.

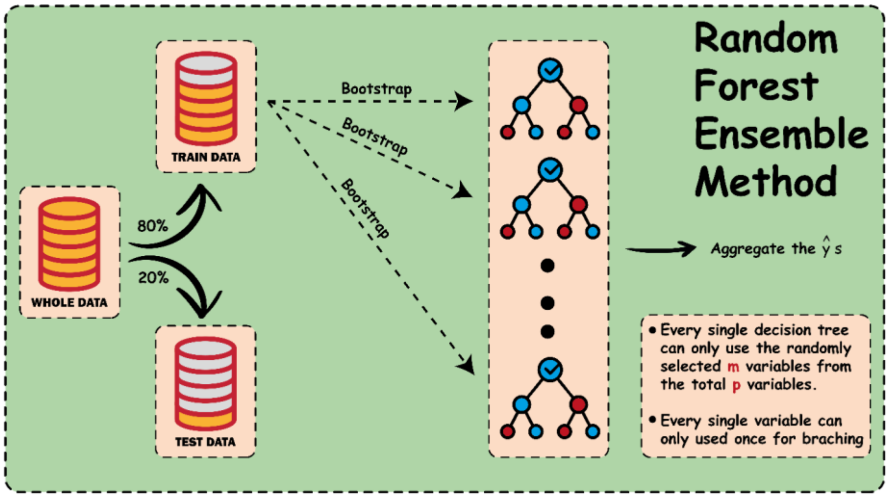

Random Forest Ensemble Method
WHOLE DATA
80%
20%
TRAIN DATA
TEST DATA
Bootstrap
Bootstrap
Bootstrap
Aggregate the $\hat{y}$ s
• Every single decision tree can only use the randomly selected m variables from the total p variables.
• Every single variable can only used once for braching

Figure 13- Illustration of Random Forest Ensemble Method

4.5. Hyperparameter Tuning

Before going directly through tuning the models' hyperparameters, we should set a training strategy to increase the robustness of the resulting performance measures. Hence, we choose the K-Fold Cross Validation as our proposed strategy. As an example, Figure 14 obviously shows our train-test split strategy and the following 10-Fold Cross Validation in one picture. Be noticed that some of our models follow other resampling method

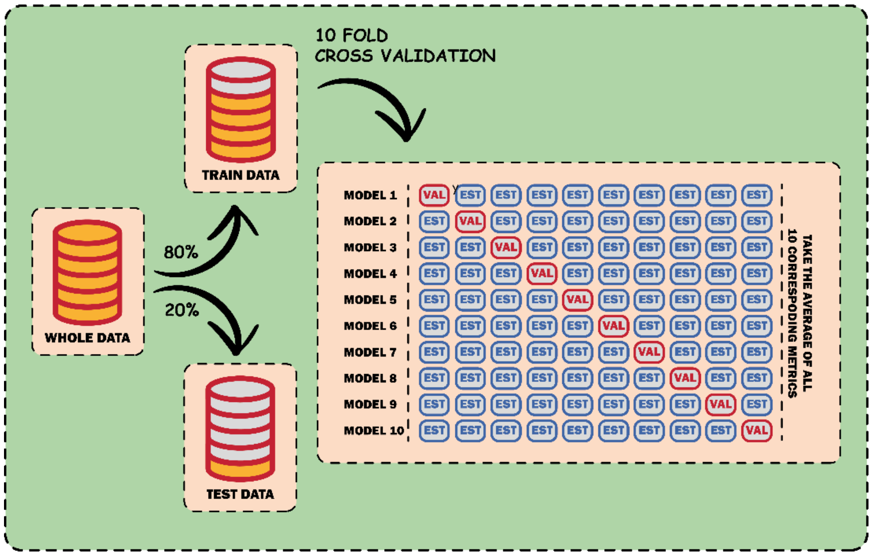


Figure 14 – 10-Fold Cross Validation workflow

The number of folds is determined based on the learning time of the models. In most cases, 10-Fold Cross Validation, and in a few of them with 5-Fold Cross Validation is implemented. Next, we will deal with different models to find their outstanding hyperparameters. The combinations of hyperparameters that we want to search into them are provided in Table 7 in Appendix 3. we provided the combinations of hyperparameters that we want to search into them for each of the chosen classification models. Additionally, the outperforming hyperparameters for each model and feature selection strategy are shown in Table 8 in Appendix 4 in conjunction with their resulting accuracies.

There is also an exception on the case of the evaluation and training strategy in hyperparameter tuning phase. It belongs to SVR model. As all we know, SVR is too computationally sensitive on the matter of the number of observations. To circumvent this challenge and make the implementation as easy as possible, we iteratively take relatively small number of samples from the train set and feed it into 5-Fold Cross Validation (We coined this procedure name as Reduced 5-Fold Cross Validation). Figure 15 shows this specific procedure in a more obvious way.

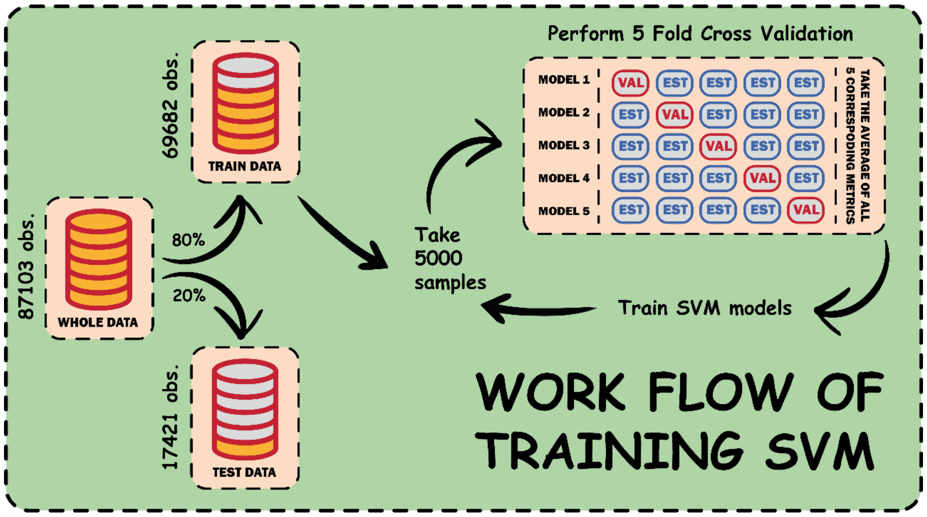

Perform 5 Fold Cross Validation
MODEL 1 VAL EST EST EST EST
MODEL 2 EST VAL EST EST EST
MODEL 3 EST EST VAL EST EST
MODEL 4 EST EST EST VAL EST
MODEL 5 EST EST EST EST VAL
TAKE THE AVERAGE OF ALL 5 CORRESPODING METRICS
69682 obs.
TRAIN DATA
87103 obs.
WHOLE DATA
80%
20%
17421 obs.
TEST DATA
Take 5000 samples
Train SVM models
WORK FLOW OF TRAINING SVM

Figure 15 - Illustration of the work flow of training SVM (Reduced 5-Fold Cross Validation)

Each model adjustment should be compared to themselves with the other hyperparameters. The reason behind is related to the models' resampling method difference.

4.6. Final Result

After completing the hyperparameter tuning process and identifying the optimal model configurations, we will proceed to predict the test dataset in this section. To evaluate the models effectively, we need to introduce a new metric specific to this problem, called **Out of Bound (OOB)**. Since xG represents a probability, any prediction exceeding 1 or falling below 0 is considered invalid. The OOB metric calculates the percentage of predictions that fall outside the acceptable range of (0,1).

$$\text{invalid:} \qquad 0 < \hat{y}_i \text{ } or \text{ } \hat{y}_i > 1$$

$$\text{Out of Bound(OOB)} = \frac{N_{invalid}}{N_{all}} * 100$$

Based on the results of hyperparameter tuning, we selected the optimal set of hyperparameters and features selection strategies that yielded the lowest MSE to train the model and make predictions on the test set. The final results of the top 3 models are reported in Table 6. Also, final results of all model are reported in Table 9 in Appendix 5.

Table 6- Results of top 3 models

Regression model	Feature set	Hyperparameters	MSE	R-squared	OOB
KNN	Forward Stepwise	k = 18	0.002928	0.869158	0
Random Forest	Forward Stepwise	max-depth = 13 n-estimator = 40	0.002727	0.878126	0
Bagging	Forward Stepwise	max-depth = 13 n-estimator = 40	0.002726	0.878155	0

As you see in Table 6, the best-performing regressor is Bagging, which achieved the lowest MSE and an Out-of-Bound (OOB) error of 0. Tree-based models demonstrated superior performance. Additionally, tree-based models have a good ability to preserve the true range of the response variable, which can be one of their advantages compared to other models.

5. Conclusion and Future works

In this study, we developed and evaluated a regression-based model for predicting Expected Goal by integrating both shot-specific and contextual features. Through rigorous data cleaning, normalization, and feature engineering—necessitated by limitations in the original dataset—we trained and tested multiple models using an 80/20 split and optimized hyperparameters via K-Fold Cross Validation. Performance metrics on the hold-out set, including mean squared error and R^2, alongside a novel prediction-interval metric, consistently highlighted the superiority of tree-based approaches. In particular, the Bagging regressor achieved an R^2 of 0.878, underscoring its robustness in capturing the complex interactions that influence shot outcomes.

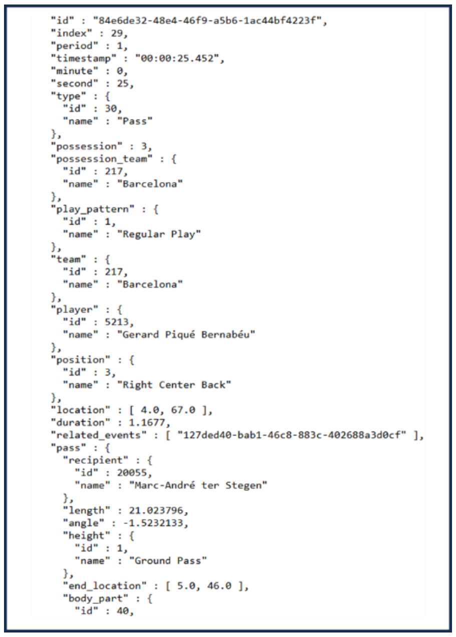
For future works, the predictive power of the xG model could be enhanced by incorporating additional contextual and dynamic features, such as player fatigue levels, game state, and teammate positioning changes just before the shot. These features can offer deeper insight into the underlying conditions affecting shot success.

Moreover, we can take the advantages of incorporating the extracted features of sequential actions preceded by the shot execution. Moreover, defining the neural network model that can be trained on the tabular features and the images of the player positional features can be beneficial to capture more information inside the model.

Appendix

Appendix 1

Due to the highly detailed characteristics of the data, it is not possible to store them in tabular form. Instead, they are stored in nested dictionary structured data as Json[1] files that connected to each other by their corresponding primary keys hierarchically (Tournament, Match, Event, and freeze frame). In Figure 16, there is a screen shot from Hudl-Statsbomb GitHub repository that they keep the data in it.



```
"id" : "84e6de32-48e4-46f9-a5b6-1ac44bf4223f",
"index" : 29,
"period" : 1,
"timestamp" : "00:00:25.452",
"minute" : 0,
"second" : 25,
"type" : {
  "id" : 30,
  "name" : "Pass"
},
"possession" : 3,
"possession_team" : {
  "id" : 217,
  "name" : "Barcelona"
},
"play_pattern" : {
  "id" : 1,
  "name" : "Regular Play"
},
"team" : {
  "id" : 217,
  "name" : "Barcelona"
},
"player" : {
  "id" : 5213,
  "name" : "Gerard Piqué Bernabéu"
},
"position" : {
  "id" : 3,
  "name" : "Right Center Back"
},
"location" : [ 4.0, 67.0 ],
"duration" : 1.1677,
"related_events" : [ "127ded40-bab1-46c8-883c-402688a3d0cf" ],
"pass" : {
  "recipient" : {
    "id" : 20055,
    "name" : "Marc-André ter Stegen"
  },
  "length" : 21.023796,
  "angle" : -1.5232133,
  "height" : {
    "id" : 1,
    "name" : "Ground Pass"
  },
  "end_location" : [ 5.0, 46.0 ],
  "body_part" : {
    "id" : 40,
```

Figure 16- A screenshot from the raw form of the event data

We transformed this complicated Json datatype to a csv[2] data frame for the ease of data pre-processing (You can have an access to the Data gathering Python code via the link in footnote[3]).

Appendix 2

Through PCA approach, we transformed our data dimensionality into the non-interpretable and reduced number of features. As we did in Figure 17, to choose the proper number of components, it is needed to accumulate the explained variance of components and choose the minimum number of components which meet the condition of having accumulate explained variance more than 95%.

[1] JavaScript Object Notation
[2] Comma-separated values
[3] Data gathering

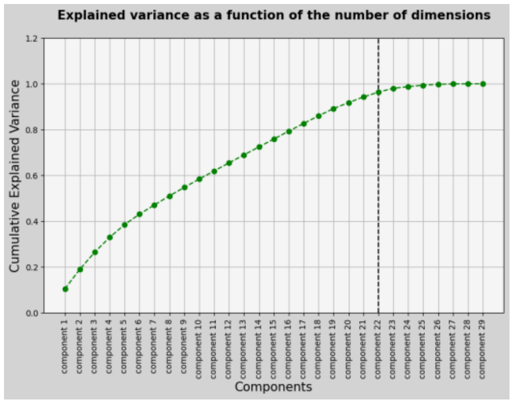


Figure 17 - xG components explained variance in PCA method

Then the resulting simple linear regression model validation MSE on these reduced features is **0.00595**.

In PLS, it constructs components through supervised approach. In Figure 18, we decided to use Elbow method to determine the number of components in our study.

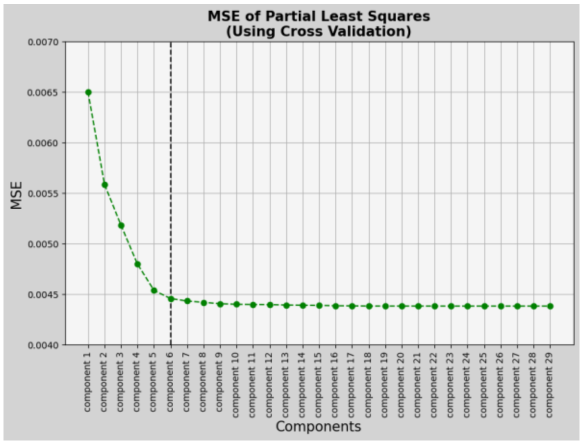


Figure 18 - xG components MSE in PLS method

Appendix 3

In this part, you can see Models' Hyperparameter Space in Table 7.

Table 7- Models' Hyperparameter Space in xG

Regression models	Hyperparameters' Space	Resampling method

Linear Regression	degree: {1,2,3}
	only-interaction: {True, False}
		10-Fold CV

KNN	k: {1, 2, 3, …, 25}	10-Fold CV
GAM[4]	spline: {cubic spline, natural spline} df: {3, 4, 5, 6, 7, 8, 9}	10-Fold CV
Ridge	λ: $\{10^{-5}, 10^{-4}, 10^{-3}, \ldots, 10^{5}\}$	10-Fold CV
Lasso	λ: $\{10^{-5}, 10^{-4}, 10^{-3}, \ldots, 10^{5}\}$	10-Fold CV

[4] Splines are only implemented on the numerical feature length. The remaining categorical features are treated as dummy variables.

Elastic Net classifier	λ: $\{10^{-5}, 10^{-4}, 10^{-3}, \ldots, 10^{5}\}$ α: {0.5}	10-Fold CV
Decision Tree	max-leaf-nodes: {10, 30, 50, … , 400}	10-Fold CV

Random Forest	max-depth: {5, 6, 7, …, 15}
	n-estimator: {10, 20, 30, 40}
		10-Fold CV

Bagging	max-depth: {5, 6, 7, …, 15}
	n-estimator: {10, 20, 30, 40}
		5-Fold CV

Gradient Boosting	max-depth: {1, 2}
	n-estimator: {10, 50, 100}
	learning rate: {0.05, 0.1, 0.5, 0.6}
		5-Fold CV

SVM	C: {0.1, 1, 10}
	kernel function: {linear, poly, rbf}
		Reduced 5-Fold CV

MLP Neural Network	activation function: {Tanh, Relu}
	size of hidden layer's neurons: {16, 32, 64}
	number of hidden layers: {1, 2}
	number of epochs: {30, 60}
	learning rate: {0.001, 0.01, 0.1}
		5-Fold CV

Appendix 4

In this section, you can see the results of Hyperparameter Tuning in Table 8.

Table 8- Hyperparameter tuning results in xG

Regression model	Feature set	Hyperparameters	MSE	R-squared

Regression	All	degree = 2
		only-interaction = False
			0.003189	0.856831

Regression	Forward Stepwise	degree = 2
		only-interaction = False
			0.003147	0.858724

KNN	All	k = 8	0.003203	0.855665
KNN	Forward Stepwise	k = 18	0.002822	0.872855
Ridge	All	λ = 1.0	0.004383	0.798886
Ridge	Forward Stepwise	λ = 1.0	0.004384	0.798886
Lasso	All	λ = 0.00001	0.004384	0.798878
Lasso	Forward Stepwise	λ = 0.00001	0.004384	0.798878
Elastic Net	All	λ = 0.00001 $\alpha = 0.5$	0.004384	0.798882
Elastic Net	Forward Stepwise	λ = 0.00001 $\alpha = 0.5$	0.004384	0.798882

GAM	All	spline = cubic spline
		df = 9
			0.003782	0.830191

GAM	Forward Stepwise	spline = cubic spline
		df = 9
			0.00383	0.827943

MLP	All	activation function = Relu size of hidden layer's neurons = 64 number of hidden layers = 2 number of epochs = 60 learning rate = 0.01	0.003238	0.854596
MLP	Forward Stepwise	activation function = Relu	0.003074	0.861945

		size of hidden layer's neurons = 32 number of hidden layers = 1 number of epochs = 60 learning rate = 0.01		
Decision Tree	All	max-leaf-nodes = 190	0.00306	0. 862554
Decision Tree	Forward Stepwise	max-leaf-nodes = 190	0.00306	0.862554

Random Forest	All	max-depth = 13
		n-estimator = 40
			0.00261	0. 882688

Random Forest	Forward Stepwise	max-depth = 13
		n-estimator = 40
			0.00261	0.882688

Bagging	All	max-depth = 13
		n-estimator = 40
			0.002629	0. 881915

Bagging	Forward Stepwise	max-depth = 13
		n-estimator = 40
			0.002629	0.881915

Gradient Boost	All	max-depth = 2
		n-estimator = 100
		learning rate = 0.6
			0.002887	0. 870354

Gradient Boost	Forward Stepwise	max-depth = 2
		n-estimator = 100
		learning rate = 0.6
			0.002887	0.870354

SVM	All	C = 1.0
		kernel = rbf
			0.006068	0.729391

SVM	Forward Stepwise	C = 1.0
		kernel = rbf
			0.004677	0.790781

Appendix 5

This section presents the final prediction results on the test set using the optimal configuration of all models.

Table 9- Final result of predicting test in xG

Regression model	**Feature set**	**Hyperparameters**	**MSE**	**R-squared**	**OOB**
Regression	Forward Stepwise	degree = 2 only-interaction = False	0.003233	0.855533	3.943516
KNN	Forward Stepwise	k = 18	0.002928	0.869158	0
Ridge	All	λ = 1.0	0.004430	0.802017	8.880087
Lasso	Forward Stepwise	λ = 0.00001	0.004482	0.799702	8.707881
Elastic Net	Forward Stepwise	λ = 0.00001 $\alpha = 0.5$	0.004483	0.799704	8.707881
GAM	All	spline = cubic spline df = 9	0.003834	0.828674	9.878881

MLP	Forward Stepwise	activation function = Relu size of hidden layer's neurons = 32 number of hidden layers = 1 number of epochs = 60 learning rate = 0.1	0.003128	0.860196	3.426898
Decision Tree	Forward Stepwise	max-leaf-nodes = 190	0.003069	0.862882	0
Random Forest	Forward Stepwise	max-depth = 13 n-estimator = 40	0.002727	0.878126	0
Bagging	Forward Stepwise	max-depth = 13 n-estimator = 40	0.002726	0.878155	0
Gradient Boost	Forward Stepwise	max-depth = 2 n-estimator = 100 learning rate = 0.6	3.426898	0.863916	1.647437
SVM	Forward Stepwise	C = 1.0 kernel = rbf	0.004269	0.809256	9.000631

References

[1] Brian Macdonald. An Expected Goals Model for Evaluating NHL Teams and Players. In 2012 MIT Sloan Sports Analytics Conference, 2012.

[2] Sam Green. Assessing The Performance of Premier League Goalscorers, 2012. Accessed 09-08-2021.

[3] Goddard John. Regression models for forecasting goals and match results in association football. International Journal of Forecasting, 21(2):331–340, 4 2005. https://doi.org/10.1016/j.ijforecast.2004.08.002

[4] Brechot Marc and Flepp Raphael. Dealing With Randomness in Match Outcomes: How to Rethink Performance Evaluation in European Club Football Using Expected Goals. Journal of Sports Economics, 21(4):335–362, 5 2020. https://doi.org/10.1177/1527002519897962

[5] Herbinet C. Predicting Football Results Using Machine Learning Techniques. Technical report, Imperial College London, 2018.

[6] Ishara Bandara and Sergiy Shelyag. Predicting goal probabilities with improved xG models using event sequences in association football. School of IT, Deakin University of Melbourne.

[7] Rathke Alex. An examination of expected goals and shot efficiency in soccer. Journal of Human Sport and Exercise, 12(Proc2), 2017.

[8] William Spearman and William Spearman Hudl. Beyond Expected Goals. In 2018 MIT Sloan Sports Analytics Conference, 2018.

[9] Brechot Marc and Flepp Raphael. Dealing With Randomness in Match Outcomes: How to Rethink Performance Evaluation in European Club Football Using Expected Goals. Journal of Sports Economics, 21(4):335–362, 5 2020. https://doi.org/10.1177/1527002519897962.

[10] Bedford Anthony and Schembri Adrian J. A probability based approach for the allocation of player draft selections in Australian rules football. ©Journal of Sports Science and Medicine, 5:509–516, 2006.