Scientific Computing, Modeling & Simulation (SCMS)
Savitribai Phule Pune University (SPPU)

Master of Technology (M.Tech.)
Programme in Modeling and Simulation

Internship Project Report

Modeling Shot Quality using Expected Goals Metric

Nirmitk Tripathi
CMS1924

Academic Year 2020-2021

Scientific Computing, Modeling & Simulation (SCMS)
Savitribai Phule Pune University (SPPU)

Certificate

This is certify that this report, titled

Modeling Shot Quality using Expected Goals Metric,

authored by

Nirmitk Tripathi (CMS1924),

describes the project work carried out by the author under our supervision during the period from July 2021 to January 2022. This work represents the project component of the Master of Technology (M.Tech.) Programme in Modeling and Simulation at the Center for Modeling and Simulation, Savitribai Phule Pune University.

Mihir Arjunwadkar, Director
SCMS-SPPU
Pune 411007 India

Scientific Computing, Modeling & Simulation (SCMS)
Savitribai Phule Pune University (SPPU)

Author's Declaration

This document, titled

Modeling Shot Quality using Expected Goals Metric,

authored by me, is an authentic report of the project work carried out by me as part of the Master of Technology (M.Tech.) Programme in Modeling and Simulation at the Center for Modeling and Simulation, Savitribai Phule Pune University. In writing this report, I have taken reasonable and adequate care to ensure that material borrowed from sources such as books, research papers, internet, etc., is acknowledged as per accepted academic norms and practices in this regard. I have read and understood the University's policy on plagiarism (`http://unipune.ac.in/administration_files/pdf/Plagiarism_Policy_University_14-5-12.pdf`).

Nirmitk Tripathi
CMS1924

Abstract

The most important statistic in a game of football is the number of goals scored. So it becomes imperative for everyone involved to find ways to improve their goal scoring abilities. Any technique, or piece of insight that can give them an edge over others, is a topic of intense research and inquest. One of the most important of such pieces of research is the Expected Goals Metric. **Expected goals (xG)** are the number of goals that can be expected to be scored based on where and how a shot was taken. An xG of 0.93 implies that 93 times out of 100, that particular shot would have been a goal. The objective of this project is to build a Machine Learning model from scratch, including webscraping requisite data required to build such a model and use the scraped data to quantify and hence measure and rate the quality of any arbitrary shot taken in a football match given the location of the shot taken and the body part with which it was taken. Such quantification of a football shot will not only be incredibly useful for footballers and coaches in order to assess and improve their performances, but will also help scouts and clubs as a measure to judge a player's shot quality and make an educated decision on player recruitment.

Acknowledgements

I am indebted to **Dr. Bhalachandra Pujari** for his support on this project. I am thankful to Dr. Pujari for his help and support which made my work much easier than it would have otherwise been. Special thanks to **Prof. David J. Sumpter** for his path-breaking lectures on football data analysis and his insightful book on statistical and probabilistic modeling of soccer, **Soccermatics**. I am also thankful to him for sharing his own wonderful reference articles in public, which I have referenced and duly cited whenever necessary in this report, allowing me to conduct and complete my project in such a niche topic as sports data science and modeling.

I would like to convey my utmost gratitude to Department of Scientific Computing, Modeling and Simulation, Savitribai Phule Pune University (SCMS-SPPU) for supporting my project and providing me with a wonderful working environment.

I would also like to thank **Sushant Rao** for his workshop in **Mad About Sports** as well as continuous help and guidance after the workshop whenever I called upon him. His help was certainly a boon for me.

Last but not the least I would like to thank my friends and CMS colleagues Gaurav Tikhe, and Jayesh Patil, for giving me this idea of doing a project in football data analytics in the first place and then motivating me to undertake it further. I would also like to further thank Sandeep Lengare, Akshay Ghatage and Sanket Kotkar for their encouragement and help during this duration of project and helping me out whenever I needed their help.

Contents

Chapter 1

Introduction

In the era of big data and artificial intelligence, many different domains such as healthcare, finance, banking, retail, travel, social media and many more have started employing mathematical models on the available "big data" to analyze and improve their existing systems. Sports analytics is one such branch which has been attracting increasing interest due to availability of advanced sensing technologies for data acquisition which make the data available. This, coupled with the ability of mathematics and statistics to model anything within a given set of rules, makes mathematical modeling of sports, such as football, an interesting topic of intrigue to dive into. Football and maths start from the same point- "laws of the game". For football, these laws are decided by the International Football Association Board. For maths, these laws are defined by nature/axioms. Football has randomness and uncertainty and Maths can explain a lot about randomness and uncertainty.

1.1 Uncertainty

> *"Football is like chess, but with dice."*
>
> – Peter Krawietz(Liverpool Football Club coach)[5]

Football is also like life- it's random. In 2015, Cristian Tomasetti, an applied mathematician, and Bert Vogelstein, a medical doctor, used a statistical argument to come to a controversial conclusion that about 2/3rd of cancer cases can be attributed to just pure "bad luck". Yes, certain life choices do correlate highly with certain types of cancer, like smoking and lung cancer, but this is only one side of the coin.[14]

Sports are similar as well. Lot of the results can be attributed to pure randomness and luck, albeit, some more than the others. Football is a bit different from all other sports though. It is decided by the number of goals scored which are extremely rare events. Hence, it is extremely unpredictable and luck and randomness has far greater affect in football matches, final results and league standings, than it has in other sports[5]. In fact, football is one of the most random sports to be played with the bookmakers picking the correct favourite team less successfully than in other sports. If we compare betting odds across different sports, we will find out that just over 50% of favourites win their matches in football as compared to 60% in other sports, making it only as good as "a coin-toss game", a 50/50 proposition at best [4].

This uncertainty in football makes it harder for us to make correct decisions. The role of football analytics, thus, is to increase certainty and decrease risk by providing informed analysis.

The first foray in such a direction of soccer/football data analytics was undertaken by Richard Pollard and Charles Reep in 1997 where they were the first to introduce the concept of measuring the quality of a team possession by calculating the probability of the possession resulting in a goal being scored minus the probability of the possession resulting in a goal being conceded, using a series of recorded on-the-ball data events. [11] With a lot of hardwork and a little bit of creativity, and inspiration, both footballers and mathematicians seek to reach their respective goals. It is this particular area of modeling football, where this project will be headed towards.

1.2 Expected Goals

The modern concept of *"Expected Goals"* in its current form, aka **xG**, was first introduced by Sam Green in 2014, where he explained a metric to calculate a measure of quality of a chance or shot if taken. This metric calculated the probability of a shot being a goal on average and was a much better metric to assess a performance as compared to purely shots taken, because they can be misleading, or number of goals scored, which as explained earlier, are inherently rare and hence more prone to randomness [4]. Expected Goals, thus, attempts to strip out the uncertainty in goal scoring.

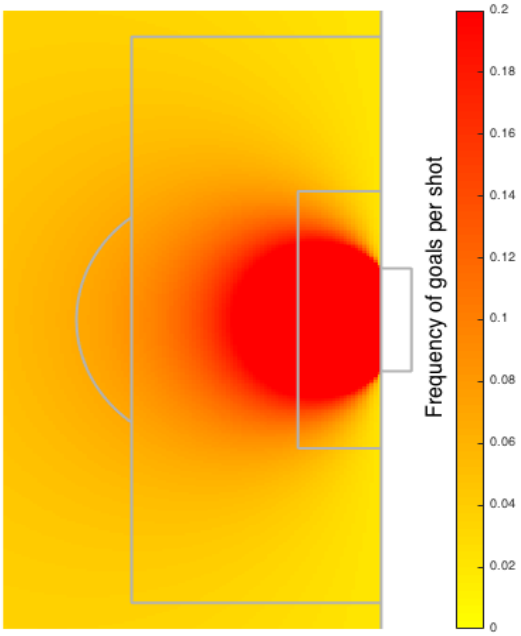

Figure 1.1: Goals to shot taken ratio w.r.t. to location on the pitch [3]

This is incredibly useful for footballers and coaches in order assess and improve their performances as well as scouts and clubs as a measure to judge a player's shot quality and make an educated decision on player recruitment. For example, by comparing a striker's cumulative xG with the number of goals he scored, a prospective recruiter can see whether the said striker is actually creating high quality chances and is just unlucky (if his goals tally is less) or whether the player in a hot streak of goal-scoring form, is just getting lucky (if his goals tally is significantly greater than his cumulative xG) or actually deserves the goals he's scoring. Accordingly,

they can realise whether or not they are overpaying for a player based on a lucky hot streak of goals or can also identify players who are unlucky but can create high-quality chances.

1.3 Data Acquisition

Historically speaking, one of the biggest hindrances in doing any kind of sports data analytics was (and still is) the lack of data availability. The fundamental difficulty in attempting to make an objective study of team performance at soccer is a lack of regularly recorded quantitative data. The first requirement, therefore, is the availability of this data and how to obtain it.

One of the earliest forays into statistical analysis and modeling of football was made by Charles Reep in 1968, where he was trying to analyse the relationship between number of passes, possession and goals scored. In his pioneering work, he utilized the data he acquired through the 1950s to model goal scoring probability on the number of passes in a pass chain [12].

He defined and designed a notational technique which enabled footballing data from a match to be recorded on paper. The real-time continuous stream of actions of the game are broken down in a series of on-the-ball discrete events like pass, shot, tackle, foul etc. and recorded. Furthermore, a detailed categorization of these events is also done, adding up more detailed information related to each such event. For instance, a pass event recorded is further categorized by additional features like the length, the direction, the outcome and the height of the particular pass [12]. 60 years ago, this cumbersome process was done by hand on pen and paper. Nowadays, there are dedicated professionals and even automated softwares (under human supervision) that record football events as such and report it to huge databases created, managed and utilized by some of the very prominent sporting organizations and companies. Since 2012, many such data acquisition and statistical and mathematical analysis companies have cropped up. Presently, the 3 most prominent organisations are Statsbomb, Wyscout and Opta.

One of the major issue faced by modern day data analysts and modelers is the issue of proprietorship of this data by those data acquiring organisations, making on-the-ball event data sets almost impossible to be accessed by anyone other than those who pay for it, basically creating a pay-wall between amateur data analysts and sports enthusiasts.

However in 2019, events data for various matches in various competitions for different seasons was made available by Statsbomb but not for all the matches in the season. For example, events data for only Barcelona's matches for all the seasons from 2007/08 to present was available for public access by Statsbomb.

Wyscout is another organization that keeps track of events data for every team and every match across various first tier leagues around the world for several seasons. They have made the complete set of events data across the Top 5 European Football Leagues (English Premier League, Spanish La Liga, Italian Serie A, German Bundesliga and French Ligue 1) for all matches for the entire 2017-18 season, freely available for public access to be utilized in football data analysis.

However for building the machine learning model to measure the quality of shot taken in this project, web scraped shot data from understat has been, primarily because the amount of data extracted from understat is way higher than that available from wyscout or statsbomb.

2

Theoretical Concepts

This section summarizes the theoretical background of the thesis. Firstly modelling randomness through Poisson distribution is covered. Then, the concept of "*Expected Goals*" and shot probability is explained in detail, followed by a discussion on supervised machine learning and the machine learning methods implemented in the project. Lastly imbalanced dataset and its handling- SMOTE technique are covered.

2.1 Modeling randomness: Poisson distribution

Consider a game of football as a dice throwing competition between 2 teams, where the dice score is assigned as the final scoreline. This can be treated as a very rudimentary model to predict the scoreline of a football match. However, it will not be an accurate model because of 2 major reasons-

1. Firstly, such a throw of dice modeling will mean that scorelines like 8-2, 9-0, or even a 0-0 will not be possible because the dice have faces numbered only from 1 to 6.

2. Secondly, the fact that in this method a 5-5 scoreline is as probable as a 1-1 scoreline which is wrong since it is common knowledge that a 1-1 scoreline is a common one while a 5-5 game is incredibly rare.

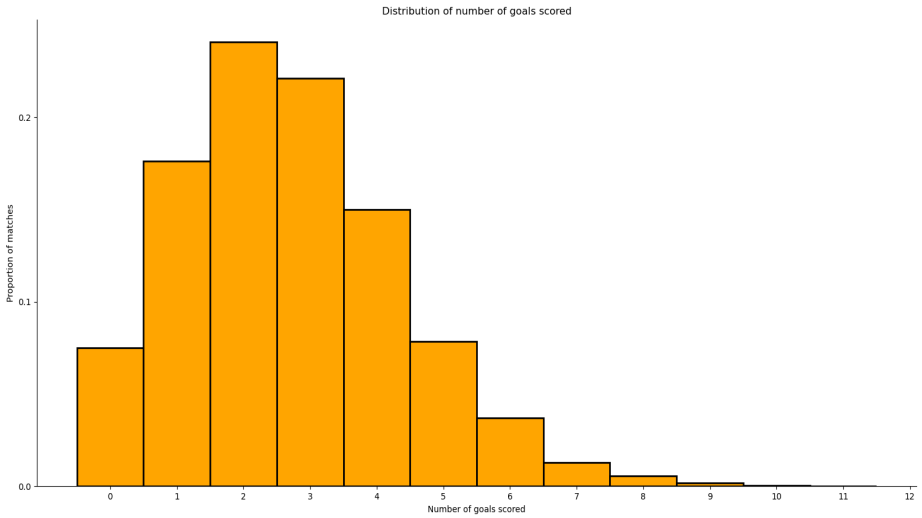

Figure 2.1: Histogram of total number of goals scored in a match for all top 5 leagues from 2014/15 to 2020/21 season.

From the histogram above one can clearly observe that a 0-0 scoreline is much more common than a 5-5 scoreline (which forms a tiny part of the bar at number 10, since there can be multiple scorelines having 10 total goals in a match- 5-5;8-2;9-1 etc.).

Therefore it will be obvious to conclude that such a model does not work for football. Football games are not random like a dice throw.

But football games are random in a different way. As mentioned in the previous section, a football match is highly unpredictable- you can be watching a game of football and just look away for a few seconds and realise that one of the teams has already scored a goal. As a modeler, this tells us something important.

> *"Luck plays a much greater role in football that we would like to admit: it's even quantifiable. If Jürgen Klopp had known as much, he might never become coach of Liverpool FC."*
>
> – Ian Graham, the director of research for Liverpool Football Club.[5]

A goal is just as likely to be scored now as it is to be scored at any other time in a match.

There could be a multitude of factors determining the rate at which a team scores, but the timing of those goals being scored remains, more or less, random.

2.1.1 Poisson Distribution

Consider a counting process that describes the occurrences of a certain type of events of interest in a unit time interval subject to three simplifying assumptions (discussed below). In

our case, imagine a football game as 90 single-minute slots, in each of which there's an equal chance of a goal being scored (or not being scored). This complete *Dichotomous experiment* is termed as a *"Bernoulli Trial"*.

Let X be the random variable describing the probability distribution of number of goals being scored in a match. We wish to know the probability that there are goals being scored in a single match. We will start with Binomial Distribution as a starting point and an approximation of $P(X = k)$. Later we will limit N (number of trials) $\to \infty$ and see how this Binomial distribution translates into a Poisson distribution.

Suppose we have access to the data of a number of matches and as a result, we know that the average number of goals scored in a match is $E(X) = \alpha$. The simplifying assumptions mentioned earlier area as follows:

1. The numbers of goals being scored in non-overlapping time intervals are independent.
2. The probability of one goal being scored in a very short time interval of length h is αh.
3. The probability of having more than one goals being scored in a very short time interval is essentially 0.

Assumption 1 implies that number of goals being scored in a random time interval does not depend on or affect the number of goals being scored in some other time period. In other words, large number of goals being scored in one interval does not mean fewer goals will be scored in the following interval.

Assumption 2 implies that the rate of goals being scored is directly proportional to only the length of the time interval and not on the exact location/value of time in that interval. For example, the rate of goals being scored for a time interval of $0 - 10$ minutes is same as that for a time interval of $35 - 45$ minutes or $60 - 70$ minutes but less than that for a $0 - 15$ minutes or $30 - 45$ minutes time interval. This also implies that non-overlapping short intervals have same probability of success.

Assumption 3 implies that in a single time interval, there can only be 2 outcomes of a trial- a success or a failure. This fulfills the criteria of a Dichotomous experiment. There can only either be an occurrence of a single goal in a given time interval or a non-occurrence of it. 2 or more goals are not allowed to happen in a single time interval slot.

In the beginning, we can assume a football match to be composed of 90 single minute slots (90 Bernoulli trials). Therefore the binomial distribution of X=k goals being scored during the match with n=90 and p=$\alpha/90$ based on our assumptions above will be:

$$P(X = k) \approx \binom{90}{k} (\frac{\alpha}{90})^k (1 - \frac{\alpha}{90})^{90-k} \quad k = 1, 2, 3, \ldots 90 \tag{2.1}$$

We can divide up the match into n equal subintervals, each of length = $1/n$. Assumption 3 ensures that each subinterval is a *Bernoulli trial* (either it is a success or a failure; one goal being scored or no goal being scored). Assumption 2 ensures that the non-overlapping subintervals all have the same probability of success. Assumption 1 tells us that all the n subintervals are independent. So breaking up the match into n moments and counting the number of moments

that are successes will result in a binomial distribution with parameters n and $p = \frac{\alpha}{n}$. So we are ready to proceed with the following approximation to our probability distribution:

$$P(X = k) = \binom{n}{k}(\frac{\alpha}{n})^k(1 - \frac{\alpha}{n})^{n-k} \quad k = 1, 2, 3, \ldots n \tag{2.2}$$

As we increase the number of trials/granularity (number of time interval slots), i.e., as N (number of trials) $\to\alpha$, our Binomial distributions starts translating into a Poisson distribution:

$$P(X = k) = \lim_{n\to\infty} \binom{n}{k}(\frac{\alpha}{n})^k(1 - \frac{\alpha}{n})^{n-k} = \frac{e^{-\alpha}\alpha^k}{k!} \quad k = 1, 2, 3, \ldots n \tag{2.3}$$

The proof for the above equation is given here: A

2.1.2 Modeling the randomness

From the data available from the top 5 leagues in Europe from 2014/15 to 2020/21 seasons we calculated that on an average, 2.70 goals are being scored in a match in total. This implies that the probability that there is a goal in any one of these slots will be, $2.7/90 = 0.03$. In other words, the chance of us seeing a goal in any randomly chosen minute is about 1 in 33.

Using this model we simulate 90 minutes of a football match with the probability of a goal being scored during each 1 minute interval is 0.03, several times and find out the frequency of the total number of goals scored in a match. The result is shown in the figure below.

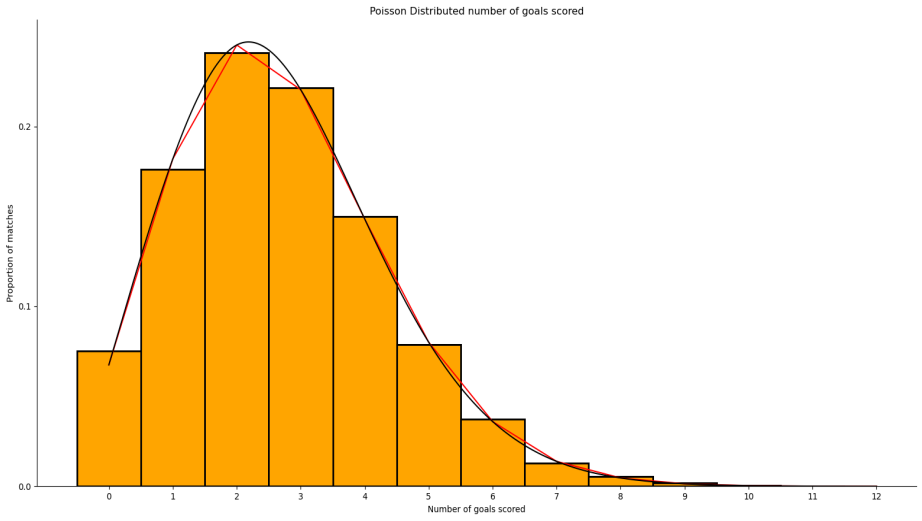

Figure 2.2: Poisson Distribution of number of goals scored in a match

The graph plotted below is known as the Poisson distribution. Such a distribution arises whenever the timing of occurrence of previous events has no effect on the occurrence of future events. This is exactly what happens in football. The probability that a goal will be scored between nth and (n+1)th minute is same as, and does not depend on, the probability of a goal being scored (n-1)th and nth minute. Neither the number of goals already scored so far in the match, nor the amount of time played, influences the probability of another goal being scored.

As a result, the resulting Poisson curve does remarkably well in capturing the overall shape and trend of the goal histogram.

It is this randomness in football which makes its every minute unpredictable and pops out Poisson distribution.

2.2 Expected Goals

The emergence of *Poisson Distribution* in goal-scoring and the subsequent randomness in modeling goal-scoring in football matches raises 2 questions in front of us:

1. How far into the future can we predict a football match?
2. What is the probability whether a shot taken will be a goal or not?

It is the second question that is answered by ***Expected Goals***

What is xG? Very simply, xG (or expected goals) is the probability that a shot will result in a goal based on the characteristics of that shot and the events leading up to it. Some of these characteristics/variables include:

1. **Location of shooter:** How far was it from the goal and at what angle on the pitch?
2. **Body part:** Was it a header or off the shooter's foot?
3. **Type of pass:** Was it from a through ball, cross, set piece, etc?
4. **Type of attack:** Was it from an established possession? Was it off a rebound? Did the defense have time to get in position? Did it follow a dribble? [1]

The context of a scoring opportunity is precisely what informs its xG rating. A rebound falling to a player in front of an open goal six yards out will have a high xG score, but a shot taken from 35 yards at a narrow angle will have a low xG score. If you see that chance is described as having an xG rating of 0.35 that means a player would be expected to score from the chance 35 per cent of the time- a one in three chance. If a chance is described as 0.5 xG it should be scored 50% of the time and so on.

2.2.1 Relationship between Expected Goals and Shot Location

The biggest parameter to affect the probability of a shot being a goal is its location, i.e., its X and Y co-ordinates on the football pitch. This relationship between the xG value of a shot with its location can be translated to a correlation between xG value with the distance of the location from goal line **(aka Shot Distance)** as well as the angle made by the shot location with the goal mouth **(aka Shot Angle)**

The rules for shooting are mathematical, even if we don't always notice it. The figure shows three different shooting positions and the angles between the goal posts.

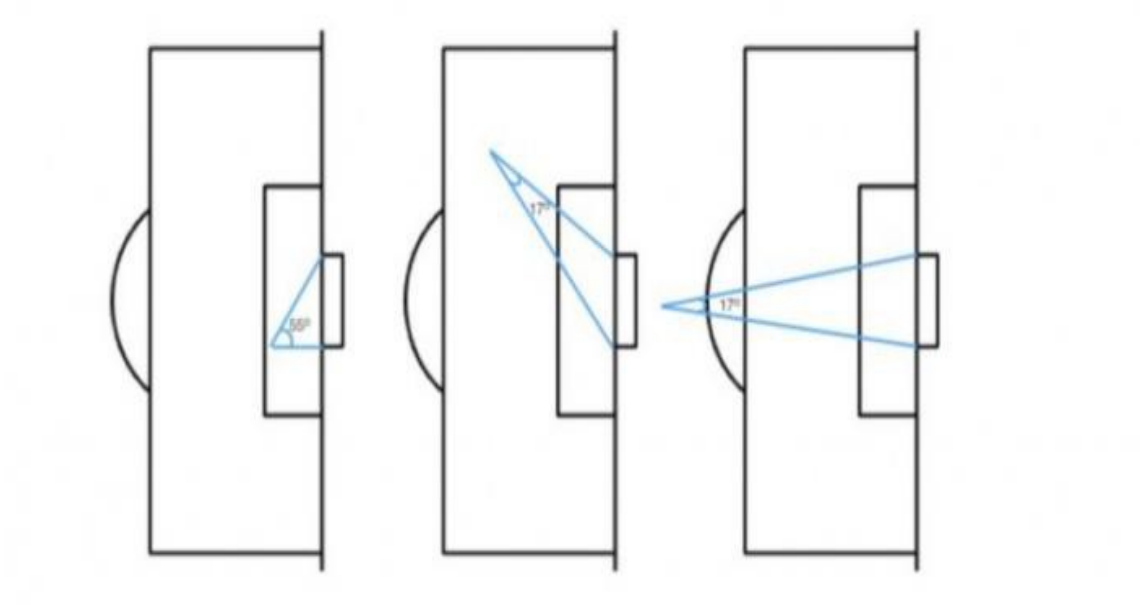

Figure 2.3: The Geometry of Shooting[3]

In the left-hand example the angle is 55 degrees, giving a very good chance of a goal from here. In the middle and right-hand figures the angles are both 17 degrees, providing much narrower chances. From here we can clearly observe a positive correlation between shot angle and goal scoring chances. Greater the angle formed by the goal mouth with the shot location, greater the chances of goal scoring from that shot (aka "xG") From here we can also observe a negative correlation between shot distance and goal scoring chances. Closer the shot location from the goal mouth, greater the chances of goal scoring from that shot (aka "xG"). This "common sense" observation was further reinforced by the regression curves for the xG values of shots taken with respect to Shot Distance and Shot Angle.

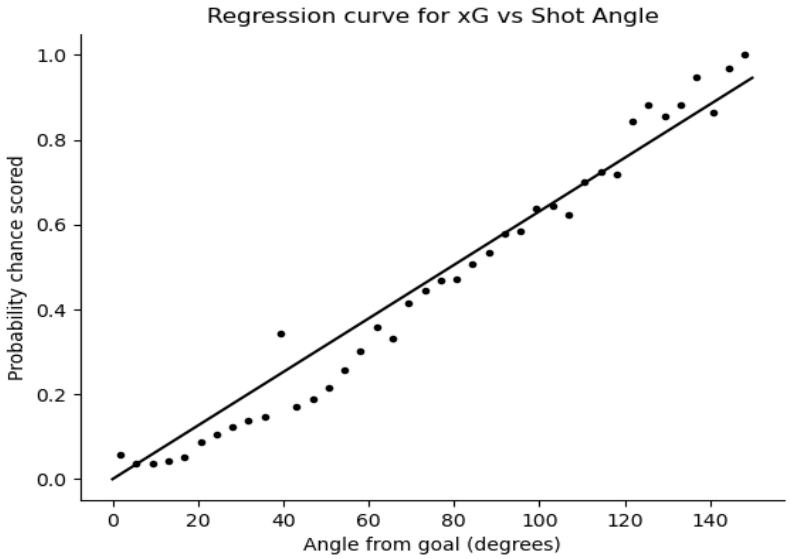

Figure 2.4: Regression curve for xG vs Shot Angle

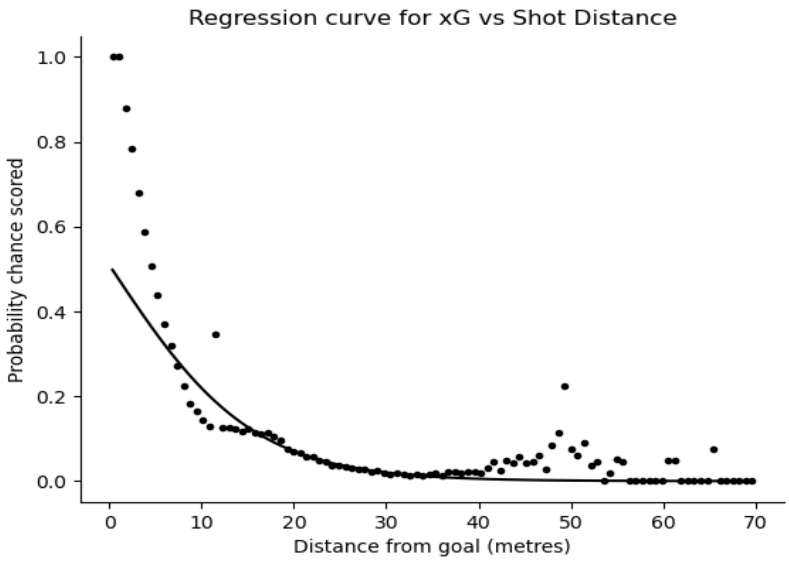

Figure 2.5: Regression curve for xG vs Shot Distance

The most simplistic probabilistic concept to calculate ***"xG"*** value of a shot from a particular location uses the simple ratio between the total goals scored from that particular location to the total shots taken from that location. Using the nglish Premier League data for the 2016/17 season, this particular approach was found to be flawed.

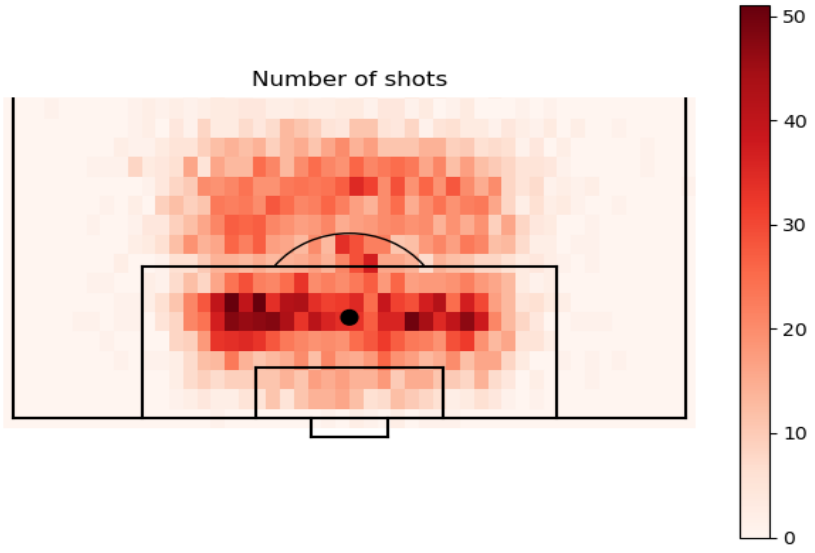

Figure 2.6: Concentration of number of Shots during 2016/17 season

As we can see from the figures, few locations far off from the goal have ***"xG"*** values higher than those near the penalty area. This is a paradoxical situation since the probability of a shot taken from that far out has to be less than that from inside the penalty area. This particular

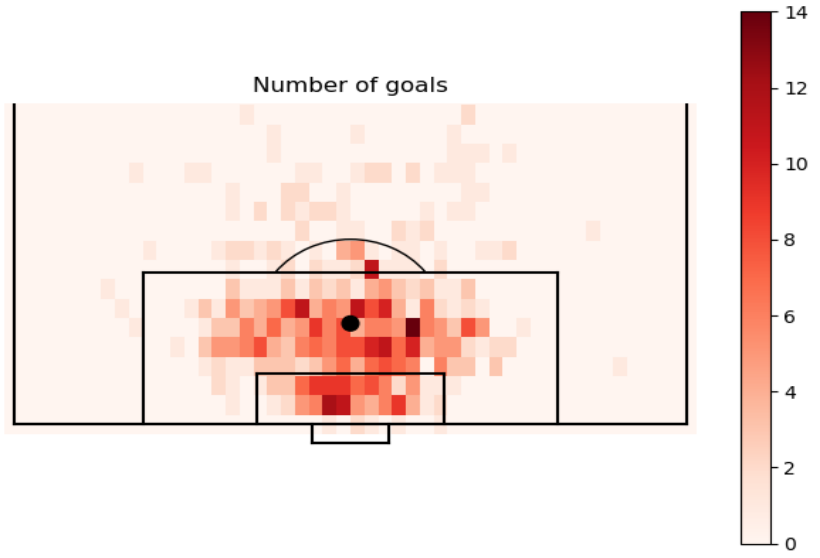

Figure 2.7: Concentration of number of Goals during 2016/17 season

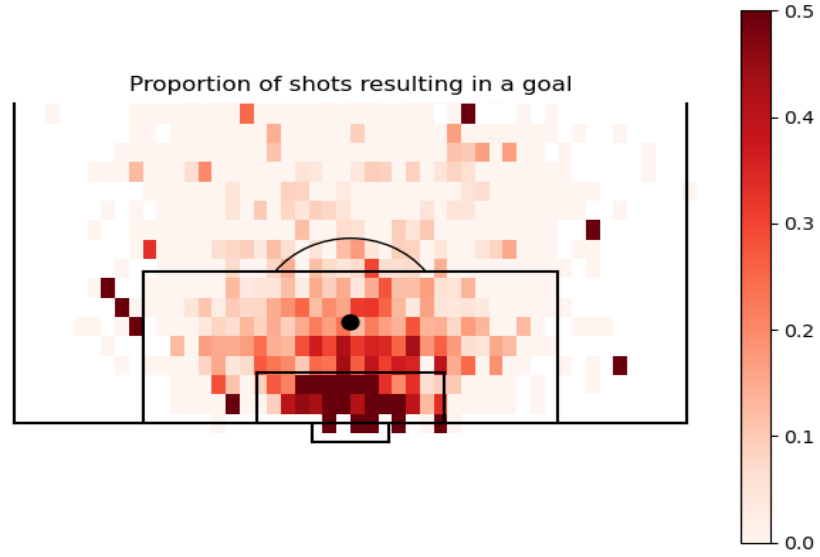

Figure 2.8: Probability of a shot being a goal during the 2016/17 season

anomaly arises because of luck. During the course of a season, only a few shots are taken from far out and a few of those end up being goals due to quality of the shot taker or due to a goalkeeping mistake. This in turn increases the ***"xG"*** value of a shot taken from that location.

2.2.2 Shot Distance and Shot Angle

Shot Distance from goal is calculated using some high-school mathematics, by using the 2D distance formula as we know the X and Y coordinates of the shot location. Keeping center of goal mouth as the origin $(0, 0)$

$$Shot\,Distance = \sqrt{(X^2 + Y^2)} \tag{2.4}$$

Similarly, since the shot location as well as the width of the goal mouth is known, cosine law of triangle is utilised to derive the angle made by the shot with the goal mouth aka Shot Angle.

$$Shot\,Angle = \arccos{(a^2 + b^2 - 8^2)/(2 * a * b)} \tag{2.5}$$

where:

$$a = \sqrt{X^2 + (Y + 4)^2} \tag{2.6}$$

$$b = \sqrt{X^2 + (Y - 4)^2} \tag{2.7}$$

$$Width\,of\,goal\,mouth = 8 \tag{2.8}$$

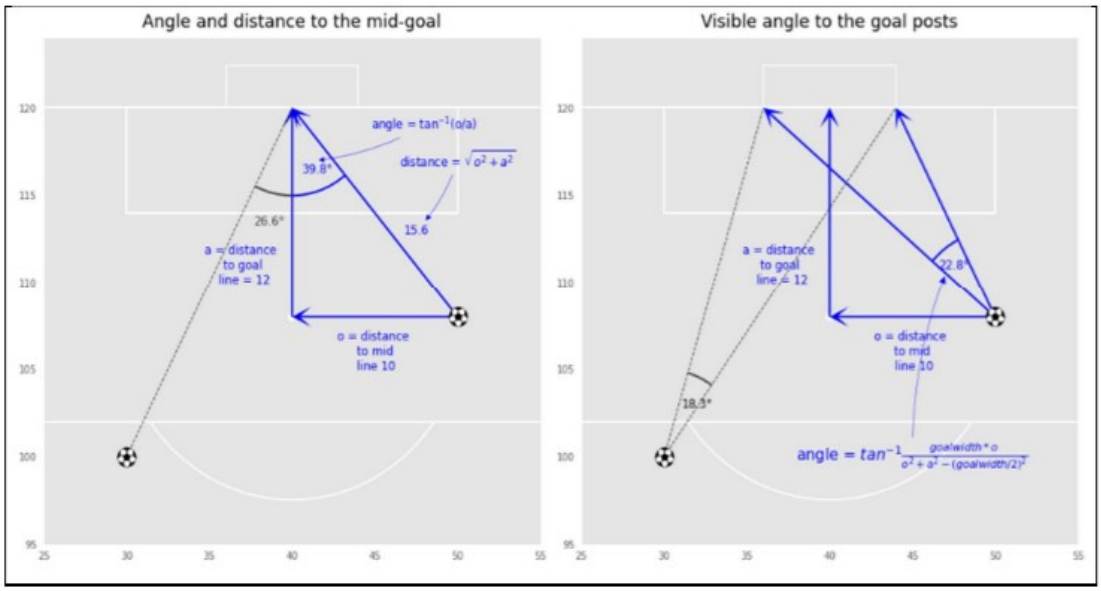

Figure 2.9: Expected Goals: calculating the angles and distances.[3]

2.3 Supervised Machine Learning

Supervised Machine Learning is a technique to learn the relationship between a set of pairwise input and output values: (x_i, y_i)[13]. It is used whenever we want to predict a certain outcome y

from a given input x, and we have examples of input/output pairs (x_i, y_i). We build a machine learning model from these input/output pairs, which comprise our training set. Our goal is to make accurate predictions for new, never-before-seen data.

There are two major types of supervised machine learning problems, called **classification** and **regression**.

In classification problems, the task is to correctly predict and classify the output depending on the input, out of a set of probable output values. In regression problems, the task is to correctly predict a numerical output depending on the input, out of a set of infinite probable real number values. The difference between the 2 is in the type of output y. If it is a *categorical* variable (out of a finite set of probable values), it is a ***classification*** problem. If it is a *numerical* variable (out of an infinite set of probable real number values), it is a ***regression*** problem.

2.3.1 Logistic Regression

It is one of the most common type of linear classification model used primarily for binary classification. It is basically an extension of Linear Regression into the genre of classification. In logistic regression, a prediction is made using the following formula:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b \tag{2.9}$$

where w denotes the set of coefficients of parameters and b is the intercept

The formula looks very similar to the one for linear regression, but instead of just returning the weighted sum of the features, we threshold the predicted value at 0.5 by default. If the function is smaller than 0.5, we predict class 0; if it is larger than 0.5, we predict class 1. This prediction rule is common to all linear models for classification [10]. This mapping of output values from $(-\infty, +\infty)$ in the case of linear regression to (0, 1) in case of logistic regression is done with the help of the *logit* or *sigmoid* function:

$$sigmoid(W.x) = \frac{1}{1 + e^{-W.x}} \tag{2.10}$$

The output of the above binary predictor equation, the value in the range (0, 1), is commonly interpreted as the probability of a sample belonging to the positive class, which is labelled 1 [13]. The value of the threshold value is what determines the classification of unknown data. If r is the threshold value of the predictor function the predictor class of sample x be $\hat{y}(x)$, then:

$$\hat{y} = \begin{cases} -1 & sigmoid(W.x) < \text{r} \\ 1 & sigmoid(W.x) \geq \text{r} \end{cases} \tag{2.11}$$

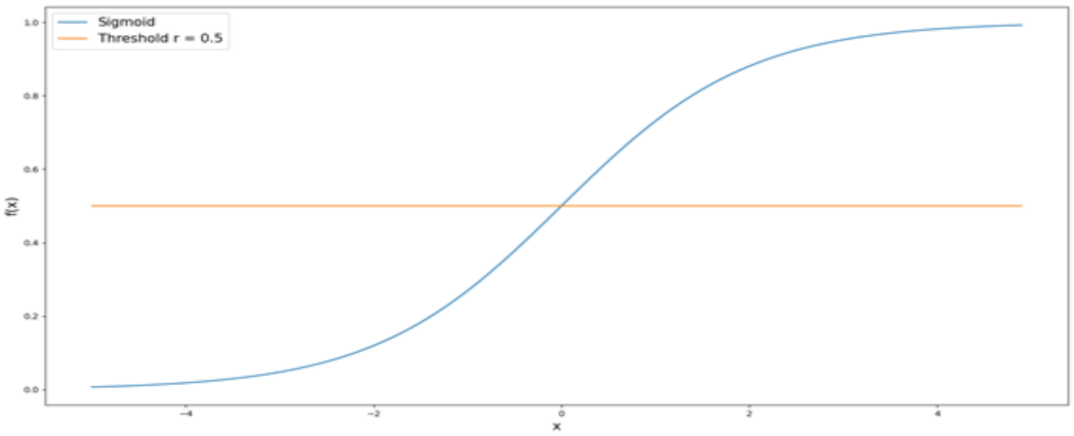

Figure 2.10: Plot of the sigmoid function. The threshold r = 0.5 is marked on the vertical axis. To predict a sample as belonging to the positive class, labeled 1, the output of the model must be larger than 0.5. This corresponds to the argument of the sigmoid function being larger than 0. For logistic regression, the argument passed to the sigmoid function is the dot product between the models weights and the input variables.

The Expected Goals metric is a classic supervised learning classification problem. The algorithm takes in certain parameters containing the contextual knowledge and information of the shot at the time it was taken, as inputs, to classify whether the shots resulted in a goal (Class 1) or not (Class 0).

2.3.2 Decision Tree Methods and Random Forest

One drawback of a logistic regression model is that domain knowledge and feature engineering are needed to encode non-linear relationships. Linear models do not handle the non-linear relationships between xG and pitch location data well (x and y coordinates).

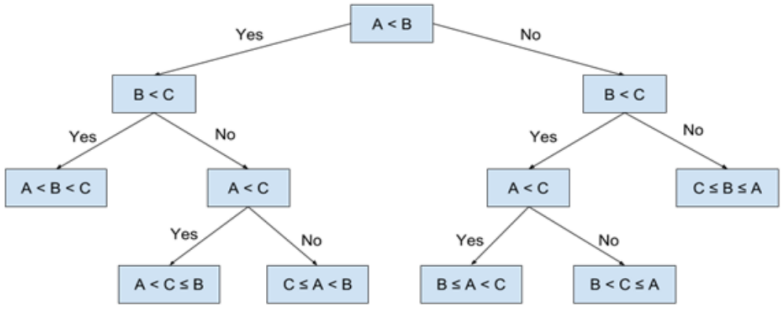

Figure 2.11: Decision Trees use a set of hierarchical if/else questions with binary outcomes until the root nodes (class labels) are reached

An alternative to linear models Logistic Regression is to use Decision Trees model to classify whether a shot is a goal or not. Decision trees essentially learn *"a hierarchy of multiple if/else questions, leading to a decision"* [10]. These questions are known as tests. The *"algorithm searches over all possible search space and finds the one that is most informative"* about whether the shot is a goal.

A random forest classifier extends decision trees, which are prone to overfitting the training data i.e. not generalising well to new examples. Random forests build many decision trees models, but "inject some randomness into the tree building to ensure that each tree is different" [10]. A prediction is then made by taking the average of the decision trees in the random forest. A random forest estimator is usually better than a single decision tree because its variance is reduced [2].

The main benefits of decision tree methods over logistic regression are:

1. They can handle categorical features without using one-hot encoding to encode features since several tests can combine to split a categorical feature

2. They can model non-linear relationships (e.g. shot location data) without any feature engineering because the trees can naturally create interactions by combining several tests to partition the data (e.g. headed shots from a cross).

2.4 Class Imbalance

Imbalanced classification involves developing predictive models on classification datasets that have a severe class imbalance. The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on the minority class, although typically, it is performance on the minority class that is most important. Binary classification is one of the simplest and most common applications of Supervised ML algorithms. The most commonly used parameter to evaluate the performance of a given model is the Percentage Accuracy parameter ($Percentage\,Accuracy = \frac{Total\,correct\,precdictions}{Number\,of\,samples\,predicted}$). However in many real world problems, the distribution of class labels is heavily lopsided and skewed. As a result, the percentage accuracy parameter to evaluate a model performance becomes problematic.

This skewness in classes can occur because of biased sampling techniques or due to error while sampling. However, such class imbalance occurs mostly when we are trying to predict and classify rare events, for example, fraudulent credit card transactions or, in our case, which shots in a football match will result in goals. Depending on the type of data we are dealing with, the distribution of the majority and minority classes can range for only slightly bias (65-35 ratio) to heavily skewed (99-1 ratio). The more common class in a dataset is termed as the majority class and the other as the minority class.

Predicting rare events is difficult and standard classifiers such as logistic regression, support vector machines and decision trees are mainly suitable for balanced datasets [7]. On imbalanced datasets, these standard predictors often perform unsatisfactorily on the minority class without any data manipulation [9]. Another challenge is that common performance measures, such as the accuracy of a predictor, introduces a bias towards the majority class [8]. For instance, consider a set of 50,000 of Covid-19 samples. Out of those several thousands, only a few hundred (500) of these samples will be positive. This is a classic case of class imbalance because the skewness between the 2 classes is at a ratio of 1:100, i.e., for every Covid +ve test result there

will be 100 Covid –ve tests results. Now a simple classifier may predict all the samples from the majority class, Class 0, and still have an outcome of 99% accuracy in predicting the majority class and 0% accuracy in predicting the minority class, Class 1, because it has failed to learn anything from the training dataset. Hence, the predictive performance on the minority class is not well captured by this metric. Another challenge with imbalanced data is the possibility of the learning algorithm treating the minority samples as noise [9].

Many different machine learning approaches have been designed to properly handle imbalanced classification [7].

1. **Oversampling:** One approach to addressing imbalanced datasets is to oversample the minority class.
2. **Undersampling:** The other approach is to synthesize minority class data points from the existing examples.
3. **SMOTE:** This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

2.4.1 SMOTE (Synthetic Minority Oversampling Technique):

Under this technique, the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbours. Depending upon the amount of over-sampling required, neighbours from the k nearest neighbours are randomly chosen. By default, the implementation currently uses five nearest neighbours. For instance, if the amount of over-sampling needed is 200%, only two neighbours from the five nearest neighbours are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbour. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general [6].

Figures below show the actual class distribution in the shot data over which the project is done. Only 10.47% of the total shots taken actually ended up being goals. Almost 9 times of the shots were not goals. In other words, the class imbalance in actual data is a 9:1 ratio. Hence **SMOTE** technique of oversampling is used to overcome this class imbalance. After **SMOTE** oversampling, the ratio between the 2 classes of data is a more comparable 44.32% of minority class as compared to 10.47% in original case.

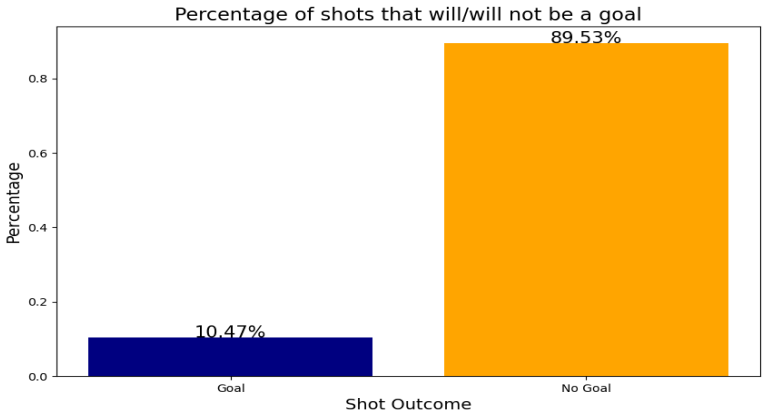

Figure 2.12: Distribution of classes before SMOTE oversampling.

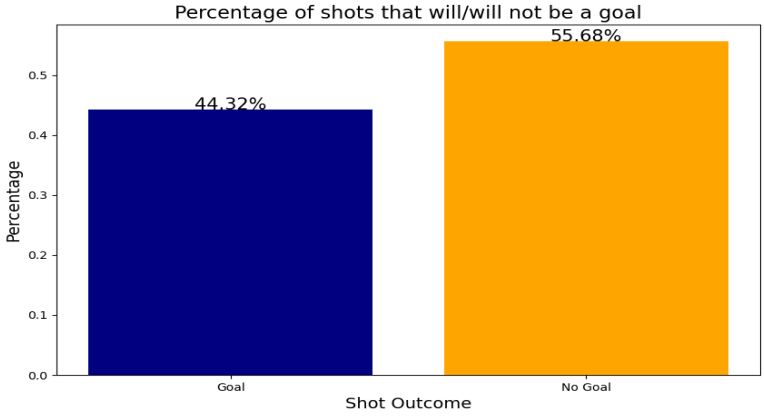

Figure 2.13: Distribution of classes after SMOTE oversampling.

3

Results and Discussion

3.1 Data Acquisition

As mentioned in section 1.3, the biggest obstacle encountered in doing any kind of sports data analytics is the lack of data availability. Situation has improved a lot during the past decade with many data recording organizations propping up. However, the issue of data proprietorship and availability of free data is still a very pertinent issue.

In order to circumvent the data proprietorship issue, requisite open-source data was web scraped from understat.com

3.1.1 Web-scraping

Apart from Stasbomb and Wyscout data, certain valuable events data like shots taken are available on certain websites like understat.com from where these shot events data can be web-scraped and utilised. This can be achieved by utilising data scraping techniques in python. Some of the important libraries to be imported are:

1. ***numpy-*** fundamental package for scientific computing with Python
2. ***pandas-*** library providing high-performance, easy-to-use data structures and data analysis tools
3. ***requests-*** is the only Non-GMO HTTP library for Python, safe for human consumption. (quoted from official docs)
4. ***BeautifulSoup-*** a Python library for pulling data out of HTML and XML files.
5. ***json-*** Python library to handle and work with JSON format data.

3.1.2 Website research and structure of data

In any web scraping project first thing that needs to be done is researching the web-page one wants to scrape and understand the working behind it. That's fundamental. On the home page we can notice that the site has data for 6 European Leagues:

- **EPL (English Premier League)**
- **La Liga (Spanish Division 1)**
- **Bundesliga (German Top Division)**

- **Serie A (Italian Premier League)**
- **RFPL (Russian Division 1)**

Also from the website, we can see that shots data from the season 2014-15 till present season are available.

Following the website research for player data it was found that the URL structure for player shots data is given as: ***'https://understat.com/' + 'player name/' + 'player id.***

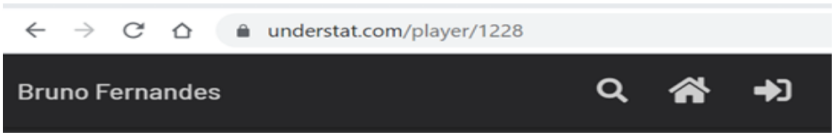

Figure 3.1: Understat.com URL structure for player shot data

The existing issue here is the fact that we need to know the "player_ids" of every player in order to scrape each and every player data. For example, ***Bruno Fernandes*** here has a **player_id** ***1228***. In order to overcome this particular problem, It was decided to create a dictionary of players with player names and their ids as key-value pairs. For that I understood that the title of the webpage holds the player name for the player-id mentioned in the URL:

This information was enough to manufacture a code to get a player name-player id dictionary for first 100 players as a test case. However, it was seen that the code took a significant amount of time to execute this code for just 100 players. So it was decided to time the code in order to get the average time the code takes to execute for 100 players.

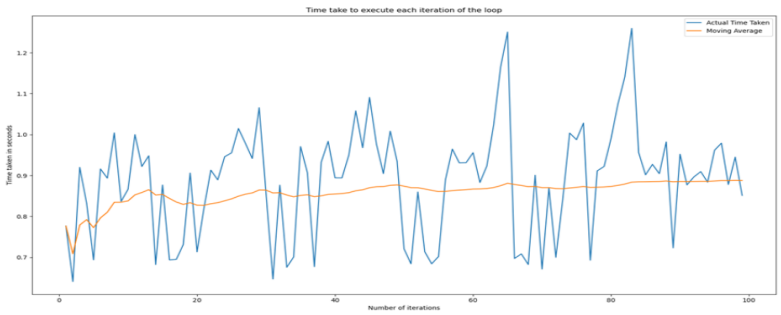

Figure 3.2: Moving Average time taken to scrap shot data for 100 players

The average time taken for the code to create a dictionary of 100 player name-player id key-value pairs was 0.8875 seconds/iteration, or 88.75 seconds in total. For 100 or even 1000 players, this is a reasonable time duration, but for higher number of players (which can even be as high as 10000), we need to implement multithreading and/or multiprocessing techniques, as opposed to regular python processing techniques which execute the codes sequentially but asynchronously.

3.1.3 Multithreading and Multiprocessing

Multiprocessing refers to the ability of the system to divide a task into multiple processes and then execute those processes on multiple processors (usually = No. of cores in the system's CPU). Several processes/tasks are performed simultaneously by the CPU, with each task being executed by a separate processor.

Just like multiprocessing, multithreading is also a way of achieving multitasking, by utilising the concept of threads. A **thread** is a subset of a process that is a sequence of such instructions within a program that can be executed independently of other code. **Multithreading** is, therefore, defined as the ability of a processor to execute such multiple "threads" synchronously, by making use of a technique called **context switching**. *In context switching, the state of a thread is saved and state of another thread is loaded whenever any interrupt (due to I/O or manually set) takes place. Context switching takes place so frequently that all the threads appear to be running in parallel.*

Multithreading increases code execution speed if the code is getting slowed down due to is I/O or peripheral or network bottlenecks with a significant number of interrupts and does not help much in CPU heavy computation tasks. Multiprocessing increases the speed regardless of the code being CPU heavy or due to I/O or network bottlenecks, by dividing the processes into number of processors thereby multiplying the load that the CPU can handle simultaneously.

Since, the present problem is that of network interrupt and bottleneck and less of a computational heavy-load, it was concluded that multithreading will be a better technique to go forward with, in order to hasten the execution of this code. Therefore, I ran the code and time its execution for multithreading.

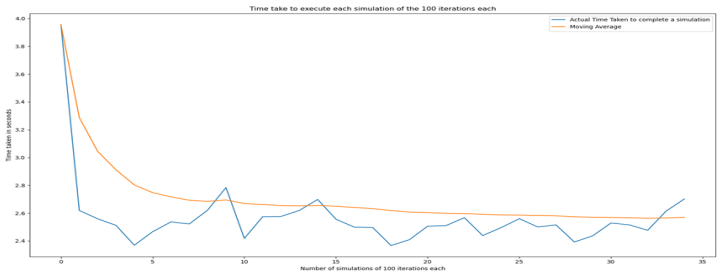

Figure 3.3: Moving Average time taken to scrap shot data for 100 players over 100 simulations

In fact, when the code was timed, it was observed that, on one hand, asynchronously running the code for creating a player_dict for 100 players was taking 88.75 seconds (mentioned previously), while on the other hand, it was possible to repeat the same process (called as simulations in the plot above) 35 times within 88.75 seconds using multithreading. This meant that the given code to scrap player names-player ids dictionary from the understat site was made 35 times faster using multithreading. The moving average came out to be 2.57 seconds, i.e, it took only 2.57 seconds, on average, for the code to create the player_dict dictionary of 100 players *(Asynchronously it took 88.75 seconds, and 88.75/2.57 = 34.5- no. of times the code was faster using multithreading).*

After a bit of brute-force, it was eventually found out that the understat website had player data for 10359 players, and player stats webpages do not exist from player_id 10360 onwards.

3.2 Expected Goals "xG" Classification ML Model with default parameters

The entire dataset has a total of 373,322 instances of shots taken over a duration of years from 2014/15 to 2019/20 season from first division leagues of England, Italy, France, Germany, Spain and Russia. The test-train split is taken to be at 20% while the trained and validated models are further tested over entirely new dataset of 93,331 shots selected randomly from 2014/15 to 2021/22 seasons at the time of model testing.

As mentioned under the section the occurrence of a goal from a shot is a rare event akin to a fraud credit card transaction or a COVID transmission. As illustrated in 2.12, the distribution of minority class to majority class is a 1:9 ratio which leads to a heavy class imbalance. As a result, **SMOTE** technique for oversampling minority class instances is undertaken which results in a distribution as illustrated in 2.13 with a minority to majority class ratio of approximately 9:11.

The correlation matrix between xG values and the 3 parameters is shown as follows:

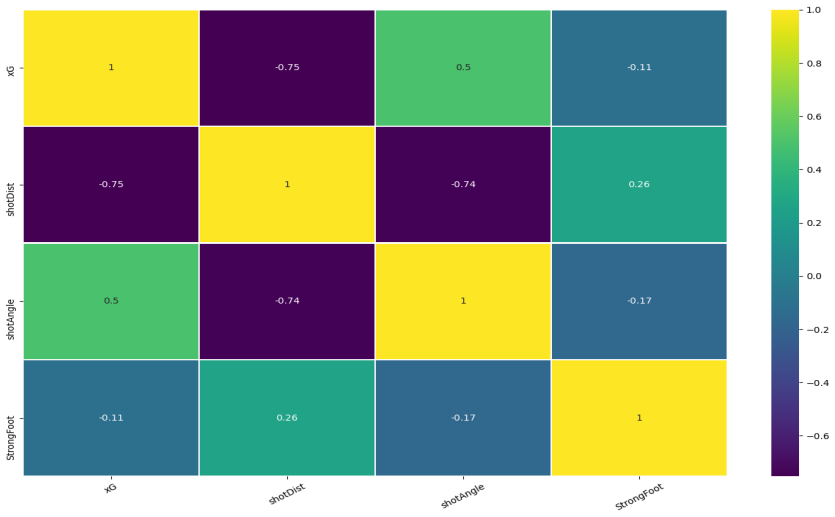

Figure 3.4: Correlation Matrix between xG values of shots taken and the 3 parameters

Once shot distance and shot angle parameters have been engineered out of shot location as well as the strong foot/weak foot/header parameter is label encoded, the dataset is good to go for model training.

3.2.1 Logistic Regression

The data was first trained on Logistic Regression algorithm and the resultant trained model was then used to classify the 93,331 shots data selected randomly from 2014/15-2021/22 season, a span of 7 years. No hyperparameter optimization was employed in this particular set classification modeling. The figures below show the result and performance of the trained model under Logistic Regression algorithm with default hyperparameters:

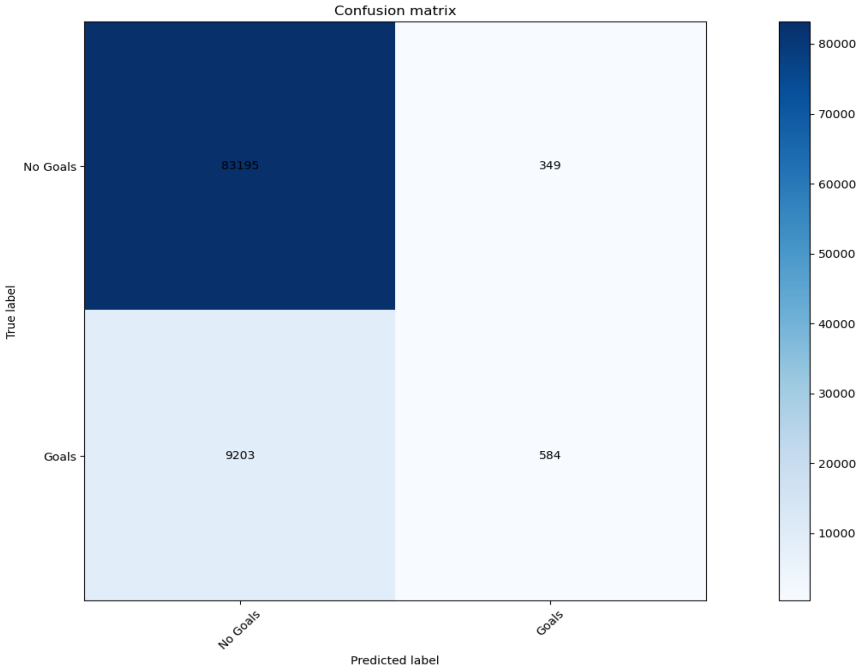

Figure 3.5: Confusion Matrix: Logistic Regression with default hyperparameters

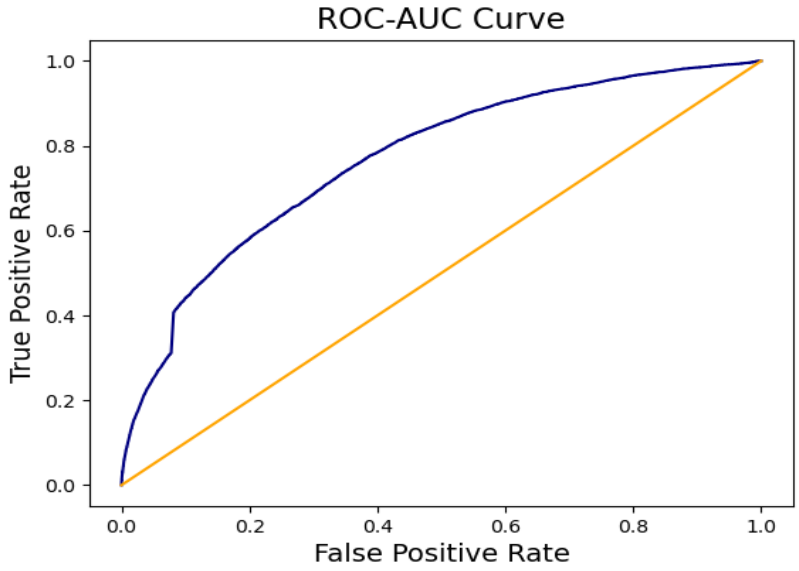

Figure 3.6: ROC-AUC Curve: Logistic Regression with default hyperparameters

	Precision	Recall	f1-score
No Goal	90.04%	99.58%	94.57%
Goal	62.59%	05.97%	10.89%
Accuracy	89.77%	89.77%	89.77%

Table 3.1: Precision, Recall and f1-score values for Logistic Regression model.

3.2.2 Random Forest

The same data was then trained on Random Forest algorithm with an aim to classify the shots with better results as ensemble algorithms have a few advantages over Logistic Regression as already mentioned n 2.3.2and the resultant trained model was then used to classify the 93,331 shots data selected randomly from 2014/15-2021/22 season, a span of 7 years. No hyperparameter optimization was employed in this particular set of classification modeling as well. The figures below show the result and performance of the trained model under Random Forest algorithm with default hyperparameters:

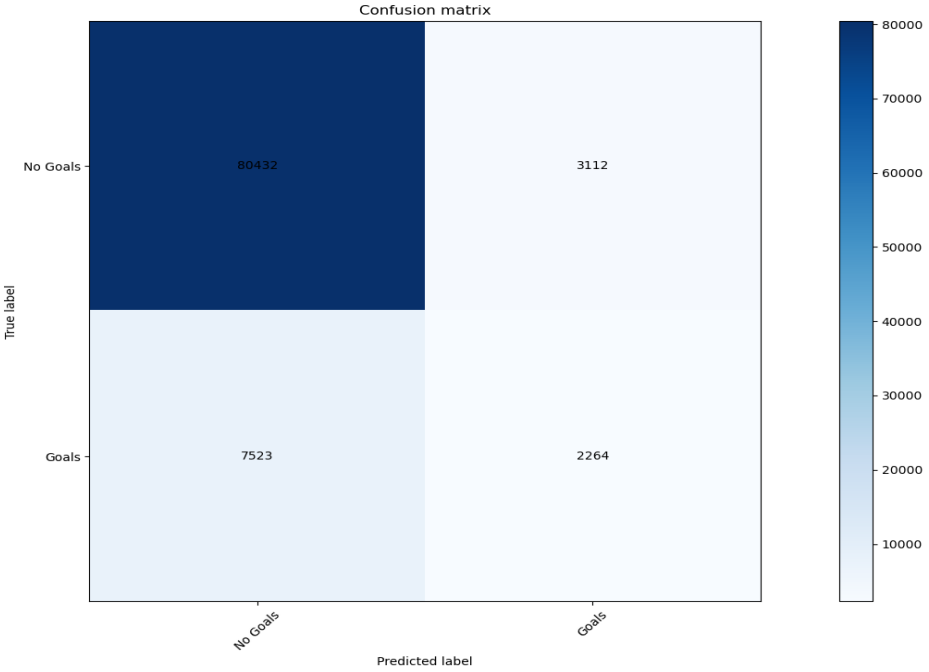

Figure 3.7: Confusion Matrix: Random Forest with default hyperparameters

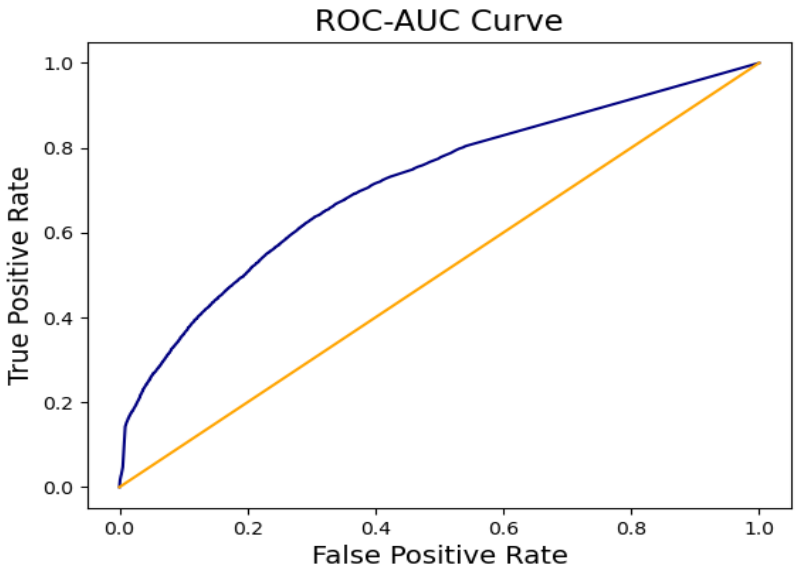

Figure 3.8: ROC-AUC Curve: Random Forest with default hyperparameters

	Precision	Recall	f1-score
No Goal	91.44%	96.27%	93.8%
Goal	42.11%	23.13%	29.86%
Accuracy	88.61%	88.61%	88.61%

Table 3.2: Precision, Recall and f1-score values for Random Forest model.

3.2.3 xGBoost

Finally, the shot dataset was trained on xGBoost algorithm with default hyperparameters and the results are illustrated in the Confusion Matrix and the ROC-AUC curve below:

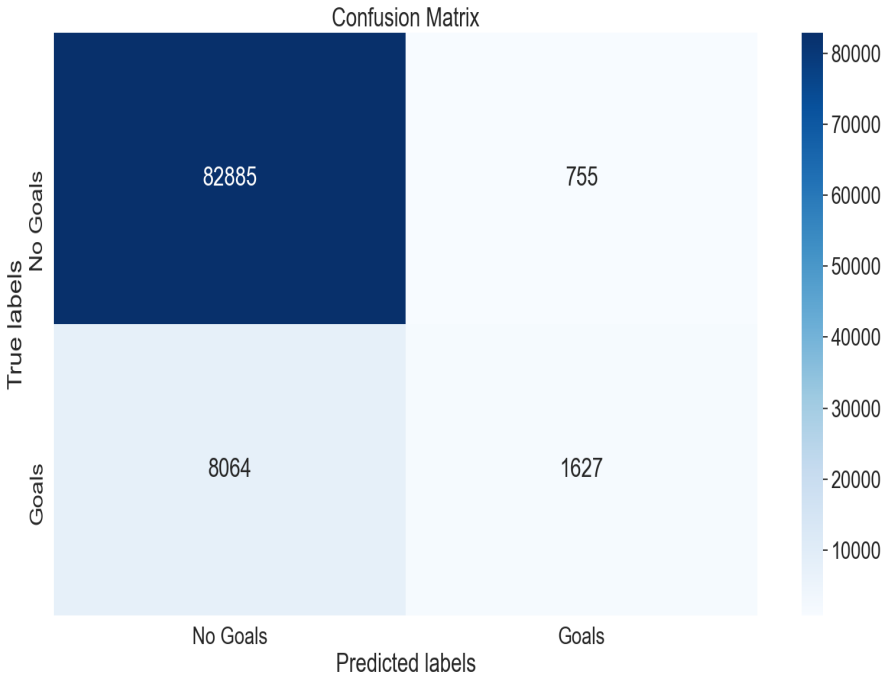

Figure 3.9: Confusion Matrix: xGBoost with default hyperparameters

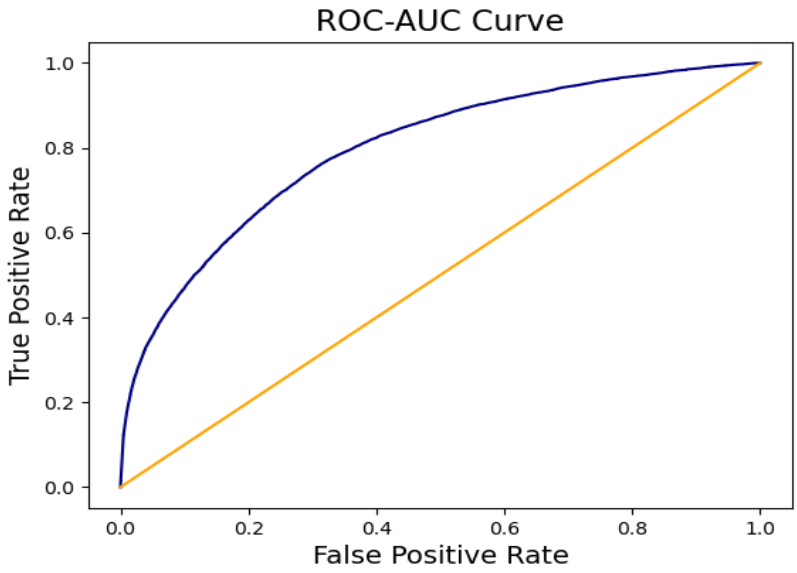

Figure 3.10: ROC-AUC Curve: xGBoost with default hyperparameters

	Precision	Recall	f1-score
No Goal	91.13%	99.1%	94.95%
Goal	68.3%	16.79%	26.95%
Accuracy	90.55%	90.55%	90.55%

Table 3.3: Precision, Recall and f1-score values for xGBoost model.

3.3 Discussion

As shown in the tables and figures above, the performance of the Random Forest model is very much comparable to that of xGBoost model. The overall accuracy of xGBoost model (90.49%) is objectively better than the accuracy of Random Forest model(88.61%). However, on closer examination we can see that the f1-score for prediction of class *Goal* is higher for Random Forest model (29.86%) as compared to the f1-score of the same class for xGBoost model (26.95%). This gives us an important inference in the process of more accurate prediction of a goal being scored from a shot being taken. The higher overall accuracy of xGBoost model can be attributed to the higher f1-score for prediction of class *No Goal* for xGBoost model (94.95%) as compared to the f1-score for the prediction of the same class for Random Forest model (93.8%)

This better performance by Random Forest in prediction of Goals from dataset as compared to that of xGBoost leads us to choose the Random Forest algorithm to delve a bit deeper into. In other words, the Random Forest model was now being trained on the same set of data, with hyperparameter optimization operation applied to further explore any maxima beyond the default hyperparameters.

3.3.1 Expected Goals "xG" Classification ML Model using Random Forest with Bayesian Hyperparameter Optimization

Bayesian Optimization is an automated hyperparameter optimization technique which uses probability to find the minimum of a function. The final aim is to find the input value to a function which can gives us the lowest possible output value. It usually performs better than random,grid and manual search providing better performance in the testing phase and reduced optimization time.

Using **Hyperopt**, Bayesian Optimization can be implemented giving 3 three main parameters to the function fmin:

1. *Objective Function* = defines the loss function to minimize.
2. *Domain Space* = defines the range of input values to test (in Bayesian Optimization this space creates a probability distribution for each of the used Hyperparameters).
3. *Optimization Algorithm* = defines the search algorithm to use to select the best input values to use in each new iteration.

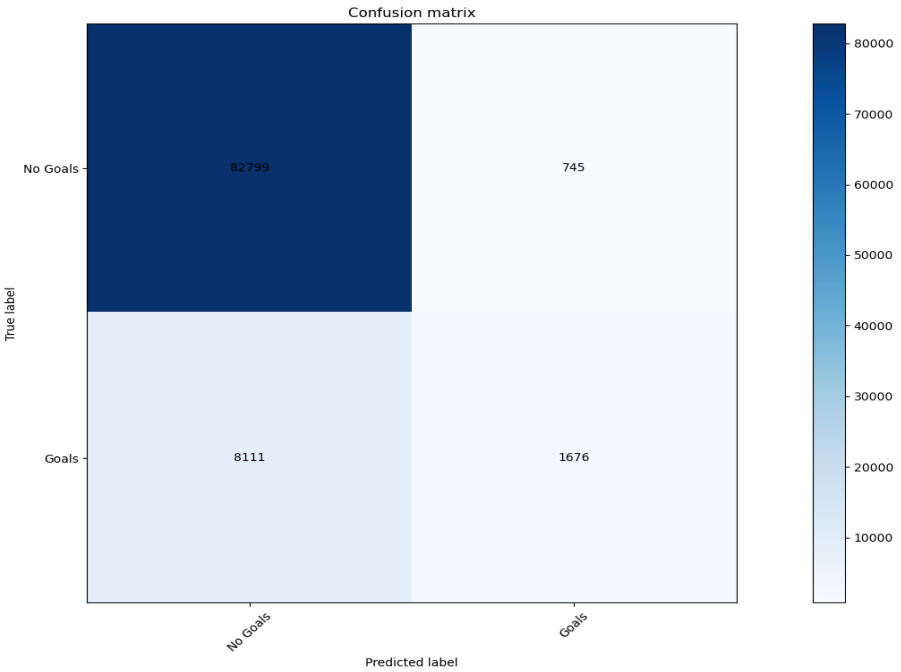

Figure 3.11: Confusion Matrix: Random Forest with Bayesian Hyperparameter Optimisation

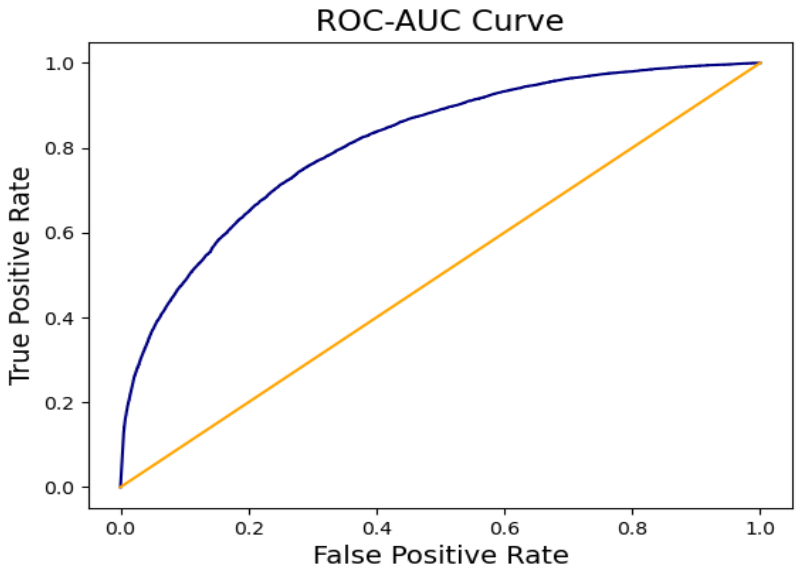

Figure 3.12: ROC-AUC Curve: Random Forest with Bayesian Hyperparameter Optimisation

	Precision	Recall	f1-score
No Goal	91.14%	99.1%	94.95%
Goal	68.45%	16.88%	27.08%
Accuracy	90.56%	90.56%	90.56%

Table 3.4: Precision, Recall and f1-score values for Random Forest model (with Bayesian HPO).

There was a mixed performance in the performance of the Random Forest algorithm after hyperparameter optimization. A set of hyperparameters were obtained at which the classification model gave a much better value of overall accuracy as compared to the accuracy obtained with default hyperparameters of Random Forest algorithm, and even a slight improvement from xGBoost algorithm accuracy as well, but the f1-score with respect to prediction of a Goal being scored was found to be lower than the value obtained with Random Forest algorithm with default hyperparameters.

4

Summary and Conclusion

As mentioned in the in 1.3, availability of open source data for sports data analysis in general, and football/soccer data analysis in particular, is one of the major hindrances into comprehensive football data analysis. This project, firstly and foremostly, gives a novel technique to webscrap a comprehensive and large enough shot dataset from sing multithreading that greatly reduces the time complexity of the data acquisition process.

Once the data was collected, it was used in training, validating and testing of 4 machine learning models. The first 3 models used 3 different algorithms- Logistic Regression, Random Forest, and xGBoost without any hyperparameter optimisation and with default hyperparameters. Logistic Regression had the worst performance amongst the 3 while Random Forest and xGBoost had comparable results. The overall accuracy of Random Forest algorithm was slightly poorer than that of xGBoost algorithm. However, the performance of Random Forest in prediction of goals being scored was slightly better than that of xGBoost. The last model used Random Forest with Bayesian Hyperparameter Optimisation and had the best overall accuracy, slightly bettering its own performance without any hyperparameter optimisation and minutely bettering the xGBoost performance. However, surprisingly, the performance of this model in predicting a goal was found to be poorer than without any hyperparameter optimsation, although it was still found to be better than the xGBoost algorithm's performance.

Possibilities for future work include the following:

1. To estimate a set of hyperparameters to better the performance of this model, using different hyperparameter optimisation techniques.

2. Use a different set of data that involves more parameters that can be utilised in classification of the shots. Preferably developing a model that involves a greater use of contextual information from a match situation. For example, whether the event preceding the shot was a pass, or a cross, or a dribble, or whether it was a dead-ball (free-kick) situation.

3. Using results found in point 1 with a smaller data set and comparing it with the results of this project and inferring the importance of size of data set with that of size of parameters/attributes set; whether it is important to have a larger data set or whether it is important to have more attributes in the data set for a better performance in classification.

4. Use Neural networks on this data set and compare the results with those found in this project as well those from point 1.

Bibliography

[1] FBref xG Explained.

[2] Scikit-learn:Ensemble Methods.

[3] David J. Sumpter, *The Geometry of Shooting*.

[4] C. Anderson and D. Sally. *The Numbers Game: Why Everything You Know About Football is Wrong*. Penguin Books Limited, 2013.

[5] C. Biermann. *Football Hackers: The Science and Art of a Data Revolution*. Bonnier Books Limited, 2019.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002.

[7] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[8] O. Loyola-González, J. F. Martínez-Trinidad, J. Carrasco-Ochoa, and M. García-Borroto. Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases. *Neurocomputing*, In press, 04 2015.

[9] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.

[10] A. Muller and S. Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Incorporated, 2018.

[11] R. Pollard and C. Reep. Measuring the effectiveness of playing strategies at soccer. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 46:541 – 550, 01 2002.

[12] C. Reep, R. Pollard, and B. Benjamin. Skill and chance in ball games. *Journal of the Royal Statistical Society. Series A (General)*, 134:623, 01 1971.

[13] S. J. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Pearson, 3 edition, 2009.

[14] C. Tomasetti and B. Vogelstein. Variation in cancer risk among tissues can be explained by the number of stem cell divisions. *Science*, 347(6217):78–81, 2015.

Appendix A

Appendix

In the derivation of 2.3, we need the following two mathematical tools. The statement A.1 is one of the definitions of the mathematical constant e. In the statement A.2, the integer n in the numerator is greater than the integer k in the denominator. It says that whenever we work with such a ratio of two factorials, the result is the product of n with the smaller integers down to $(n-(k-1))$. There are exactly k terms in the product.

$$\lim_{n\to\infty}(1+\frac{x}{n})^n = e^x \tag{A.1}$$

$$\frac{n!}{(n-k)!} = n(n-1)(n-2)...(n-k+1)\ k > n \tag{A.2}$$

The following is the derivation of 2.3

$$P(X=k) = \lim_{n\to\infty}\binom{n}{k}(\frac{\alpha}{n})^k(1-\frac{\alpha}{n})^{n-k} \tag{A.3}$$

$$= \lim_{n\to\infty}\frac{n!}{k!(n-k)!}(\frac{\alpha}{n})^k(1-\frac{\alpha}{n})^{n-k} \tag{A.4}$$

$$= \lim_{n\to\infty}\frac{n(n-1)(n-2)...(n-k+1)}{n^k}(\frac{\alpha^k}{k!})(1-\frac{\alpha}{n})^n(1-\frac{\alpha}{n})^{-k} \tag{A.5}$$

$$= (\frac{\alpha^k}{k!})[\lim_{n\to\infty}\frac{n(n-1)(n-2)...(n-k+1)}{n^k}][\lim_{n\to\infty}(1-\frac{\alpha}{n})^n)\lim_{n\to\infty}(1-\frac{\alpha}{n})^{-k} \tag{A.6}$$

$$= \frac{e^{-\alpha}\alpha^k}{k!}] \quad k=1,2,3,\dots n \tag{A.7}$$

In A.6 $\lim_{n\to\infty}\frac{n(n-1)(n-2)\cdots(n-k+1)}{n^k} = 1$ The reason being that the numerator is a polynomial where the leading term is n^k. Upon dividing by n^k and taking the limit, we get 1. Based on A.1, we have e $\lim_{n\to\infty}\left(1-\frac{\alpha}{n}\right)^n = e^{-\alpha}$. For the last limit in the derivation we have $\lim_{n\to\infty}\left(1-\frac{\alpha}{n}\right)^{-k} = 1$.