Imperial College
London

INDIVIDUAL PROJECT REPORT

DEPARTMENT OF COMPUTING

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

Predicting Football Results Using Machine Learning Techniques

Author:
Corentin HERBINET

Supervisor:
Dr. Jonathan
PASSERAT-PALMBACH

June 20, 2018

Submitted in partial fulfillment of the requirements for the Joint Mathematics and Computing MEng of Imperial College London

Abstract

Many techniques to predict the outcome of professional football matches have traditionally used the number of goals scored by each team as a base measure for evaluating a team's performance and estimating future results. However, the number of goals scored during a match possesses an important random element which leads to large inconsistencies in many games between a team's performance and number of goals scored or conceded.

The main objective of this project is to explore different Machine Learning techniques to predict the score and outcome of football matches, using in-game match events rather than the number of goals scored by each team. We will explore different model design hypotheses and assess our models' performance against benchmark techniques.

In this project, we developed an 'expected goals' metric which helps us evaluate a team's performance, instead of using the actual number of goals scored. We combined this metric with a calculation of a team's offensive and defensive ratings which are updated after each game and used to build a classification model predicting the outcome of future matches, as well as a regression model predicting the score of future games. Our models' performance compare favourably to existing traditional techniques and achieve a similar accuracy to bookmakers' models.

Acknowledgments

Firstly, I would like to thank my supervisor, Dr. Jonathan Passerat-Palmbach, for his constant support and help throughout the duration of the project. His confidence in my work encouraged me to do my best.

Secondly, I would like to thank my second marker, Prof. William Knottenbelt, for his expertise and for taking the time to discuss my ideas and to give me interesting insight.

Finally, I would also like to thank my family, for always believing in me, my friends, for their fun and support, Paul Vidal, for your friendship throughout our JMC adventure, and Astrid Duval, for being there when I needed it the most.

Contents

Chapter 1

Introduction

1.1 Data Science for Football

As one of the most popular sports on the planet, football has always been followed very closely by a large number of people. In recent years, new types of data have been collected for many games in various countries, such as play-by-play data including information on each shot or pass made in a match.

The collection of this data has placed Data Science on the forefront of the football industry with many possible uses and applications:

- Match strategy, tactics, and analysis
- Identifying players' playing styles
- Player acquisition, player valuation, and team spending
- Training regimens and focus
- Injury prediction and prevention using test results and workloads
- Performance management and prediction
- Match outcome and league table prediction
- Tournament design and scheduling
- Betting odds calculation

In particular, the betting market has grown very rapidly in the last decade, thanks to increased coverage of live football matches as well as higher accessibility to betting websites thanks to the development of mobile and tablet devices. Indeed, the football betting industry is today estimated to be worth between 300 million and 450 million pounds a year [1].

1.2 Motivation

A particularly important element of Data Science in football is the ability to evaluate a team's performance in games and use that information to attempt to predict the result of future games based on this data.

Outcomes from sports matches can be difficult to predict, with surprises often popping up. Football in particular is an interesting example as matches have fixed length (as opposed to racket sports such as tennis, where the game is played until a player wins). It also possesses a single type of scoring event: goals (as opposed to a sport like rugby where different events score a different number of points) that can happen an infinite amount of times during a match, and which are all worth 1 point.

The possible outcomes for a team taking part in a football match are *win*, *loss* or *draw*. It can therefore seem quite straightforward to predict the outcome of a game. Traditional predictive methods have simply used match results to evaluate team performance and build statistical models to predict the results of future games.

However, due to the low-scoring nature of games (less than 3 goals per game on average in the English Premier League in the past 15 years) (Fig.1.1), there is a random element linked to the number of goals scored during a match. For instance, a team with many scoring opportunities could be unlucky and convert none of their opportunities into goals, whereas a team with a single scoring opportunity could score a goal. This makes match results an imperfect measure of a team's performance and therefore an incomplete metric on which to predict future results.

Figure 1.1: Average number of goals scored per game in the English Premier League [2]

A potential solution to this problem can be explored by using in-game statistics to dive deeper than the simple match results. In the last few years, in-depth match statistics have been made available, which creates the opportunity to look further

than the match result itself. This has enabled the development of 'expected goals' metrics which estimate the number of goals a team should have been expected to score in a game, removing the random element of goalscoring.

The emergence of new Machine Learning techniques in recent years allow for better predictive performance in a wide range of classification and regression problems. The exploration of these different methods and algorithms have enabled the development of better models in both predicting the outcome of a match and the actual score.

1.3 Objectives

This project aims to extend the state of the art by combining two popular and modern prediction methods, namely an expected goals model as well as attacking and defensive team ratings. This has become possible thanks to the large amount of data that is now being recorded in football matches.

Different Machine Learning models will be tested and different model designs and hypotheses will be explored in order to maximise the predictive performance of the model.

In order to generate predictions, there are some objectives that we need to fulfill: Firstly, we need to find good-quality data and sanitize it to be used in our models. In order to do so, we will need to find suitable data sources. This will allow us to have access to a high number of various statistics to use, compared to most of the past research that has been done on the subject where only the final result of each match is taken into account.

The main approach we will take is to build a model for expected goal statistics in order to better understand a team's performance and thus to generate better predictions for the future. To build this model, we will test different Machine Learning techniques and algorithms in order to obtain the best possible performance. We will be able to use data for shots taken and goals scored such as the location on the pitch or the angle to goal to estimate how many goals a team would have been expected to score during the game, and reduce the impact of luck on the final match result.

In parallel to this, we will generate and keep track of team ratings as football matches are played to take into account the relative strength of the opposition, in a similar way to the popular ELO rating system. This will allow us to better gauge a team's current level and in consequence to generate better predictions for future games. These two key project objectives are presented in Fig.1.2.

An important part of this project will be to build a suitable Machine Learning training and testing pipeline to be able to test new algorithms, with new features, and

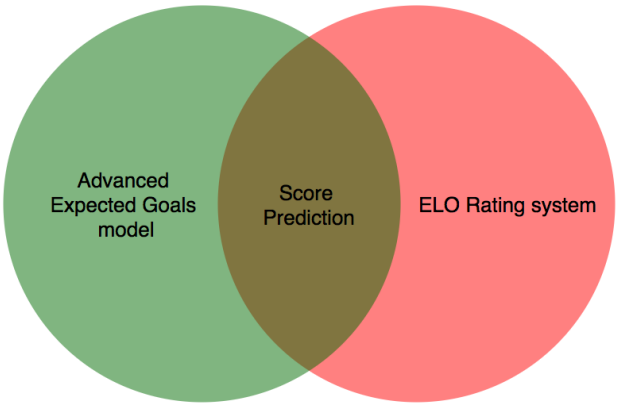

Figure 1.2: Key project objectives to generate predictions

compare it to other models with relative ease, which will result in a general project workflow illustrated in Fig.1.3.

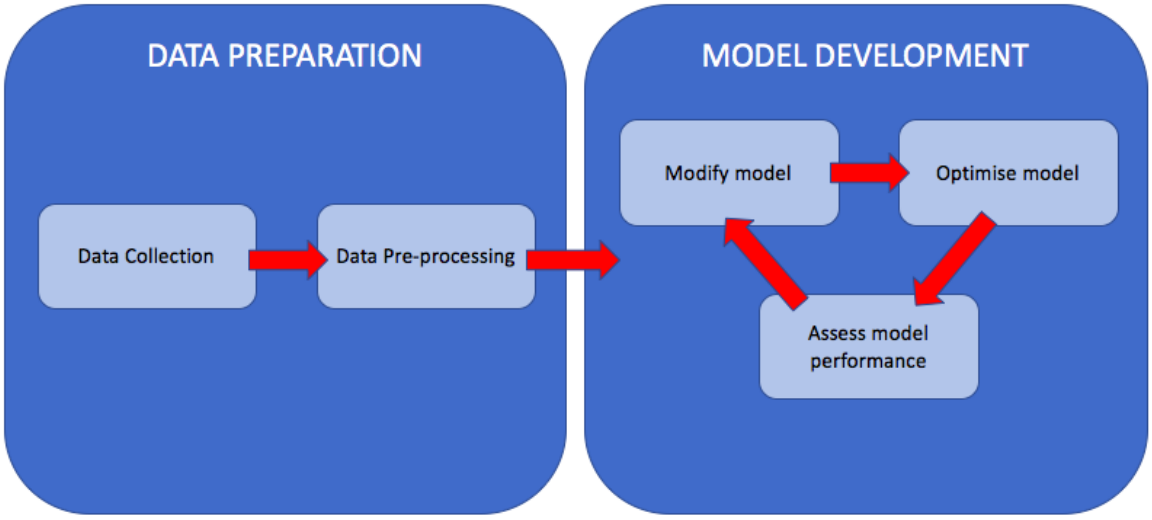

Figure 1.3: Project Workflow

Finally, our models will be assessed against benchmark predictive methods including bookmakers' odds, using different performance metrics. A successful outcome for the project would be the creation of both a classification model capable of predicting a future game's outcome, and a regression model capable of predicting a future game's score, whose predictive performance compare favourably to different benchmark methods.

1.4 Challenges

We face a number of challenges on the path to achieving the objectives we have set out:

- **Data availability & quality**: Finding a public database of football data with the necessary statistical depth to generate expected goals metrics is an essential part of the project. However, the leading football data providers do not make their information publicly available. We will need to scour various public football databases to find one that is suitable for us to use. The alternative approach in the case where we do not find a suitable database would be to find websites displaying the required data and using web scraping techniques to create our own usable database.
- **Research and understanding of prediction landscape**: In order to design our models and test different hypotheses, we will need to undertake a thorough background research of prediction techniques and develop a mathematical understanding of various Machine Learning algorithms that can be used for our predictions.
- **Testing different models and parameters**: An important challenge will be to make the model training and testing tasks as quick and easy as possible, in order to test and compare different models. A robust pipeline will have to be built to enable us to find the best possible models.

1.5 Contributions

- An exploration of the Machine Learning landscape as well as past research in sports predictions, presented in Chapter 2.
- A prepared and usable dataset containing all necessary statistics to generate expected goal metrics and calculate team ELO ratings, presented in Chapter 3.
- A model training and testing pipeline allowing us to quickly and easily generate performance metrics for different models and hypotheses, presented in Chapter 5.
- A classification model to predict match outcomes and a regression model to predict match results, presented in Chapters 4 and 6.
- An assessment of our models' performance against benchmark methods, presented in Chapter 7.

Chapter 2

Background

2.1 Football Rules & Events

In this section, we will quickly present the rules of football, different match events and possible outcomes.

- **Simplified football rules**:
  - Football is a game played with two opposing teams of 11 players and a ball, during 90 minutes.
  - On each side of the pitch, a team has a target called the goal in which they attempt to put the ball.
  - Scoring a goal gives a team 1 point.
  - The team with the largest number of points at the end of the game wins the match.
  - If both teams have scored the same number of goals, the game ends in a draw.
- **Domestic leagues competition format**:
  - Each European country usually has a domestic league where clubs play against each other.
  - There are usually 20 teams in each league, with each team playing the others twice, once at their stadium ('home' match) and once at the opposing team's stadium ('away' match).
  - Winning a match gives a team 3 points, a draw gives each team 1 point.
  - The team with the highest number of points at the end of the season wins the league.
- **Main match events**:
  - **Goals**: a goal is scored when the ball enters the opposing team's goal.

- **Shots**: a shot is when a player hits the ball towards the opposing team's goal with the intention of scoring. There are many different kinds of shots, hit with different parts of the body or from different distances and angles.
- **Passes**: a pass is when a player hits the ball towards another player of his team.
- **Crosses**: a cross is when a player hits the ball from the side of the pitch towards the opposing team's goal with the intention of passing the ball to one of his teammates.
- **Possession**: the possession represents the fraction of the time that a team controls the ball in the match.
- **Penalties & Free Kicks**: free kicks happen when a foul is committed by the opposing team on the pitch. In that case, the team that conceded the foul can play the ball from where the foul has happened. If the foul happens inside the penalty box (the zone near the goal), a penalty is awarded: the team that conceded the foul can shoot at goal from close range without anyone from the opposing team around.
- **Cards**: cards are awarded whenever the referee deems a foul to be suitably important. Yellow cards are awarded for smaller fouls and do not have a direct consequence. However, two yellow cards collected by the same player result in a red card. If a player collects a red card, they have to leave the pitch, leaving their team with one less player. Red cards can also be directly obtained if a dangerous foul is committed or in other specific circumstances.
- **Corners**: corners are awarded to the opposing team when a team hits the ball out of the pitch behind their goal. In this case, the ball is placed on the corner of the pitch and can be hit without any other player around.

2.2 Machine Learning Techniques

In this section, we will present an overview of popular supervised Machine Learning techniques, for its subsets of classification and regression.

Supervised learning is the task of learning a function that maps input data to output data based on example input-to-output pairs. Classification happens when the output is a category, whereas regression happens when the output is a continuous number. In our case, we want to predict the outcome category (home win/draw/away win) or the number of goals scored by a team (continuous number), so we are only interested in the supervised learning landscape of Machine Learning. We have summarised some popular supervised learning methods in Fig.2.1, which we will now cover in more detail.

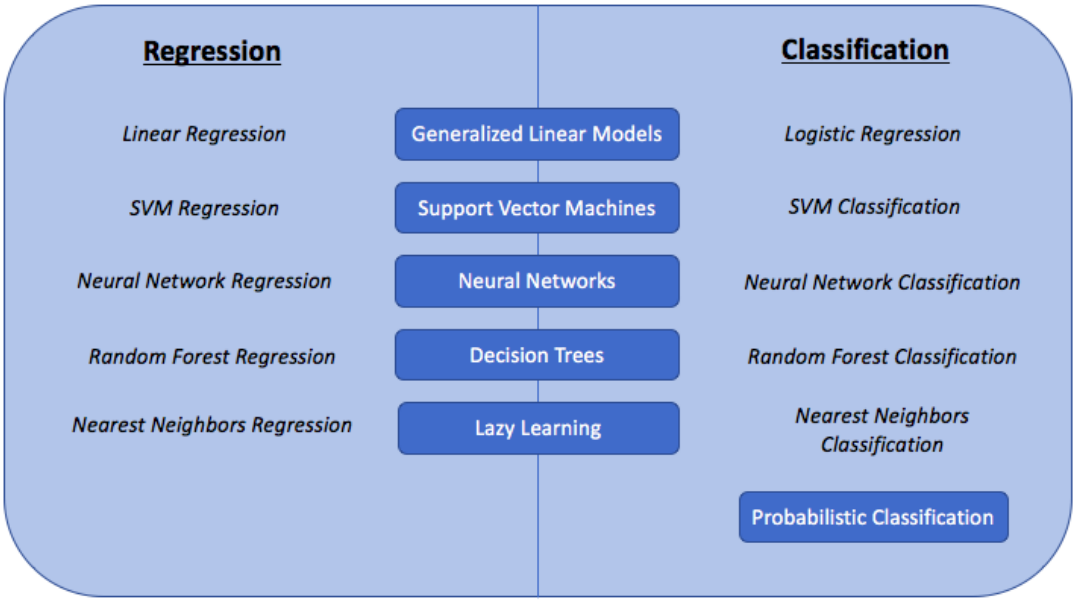

Figure 2.1: Overview of supervised Machine Learning techniques

2.2.1 Generalized Linear Models

Generalized Linear Models are a set of regression methods for which the output value is assumed to be a linear combination of all the input values (Fig.2.2).

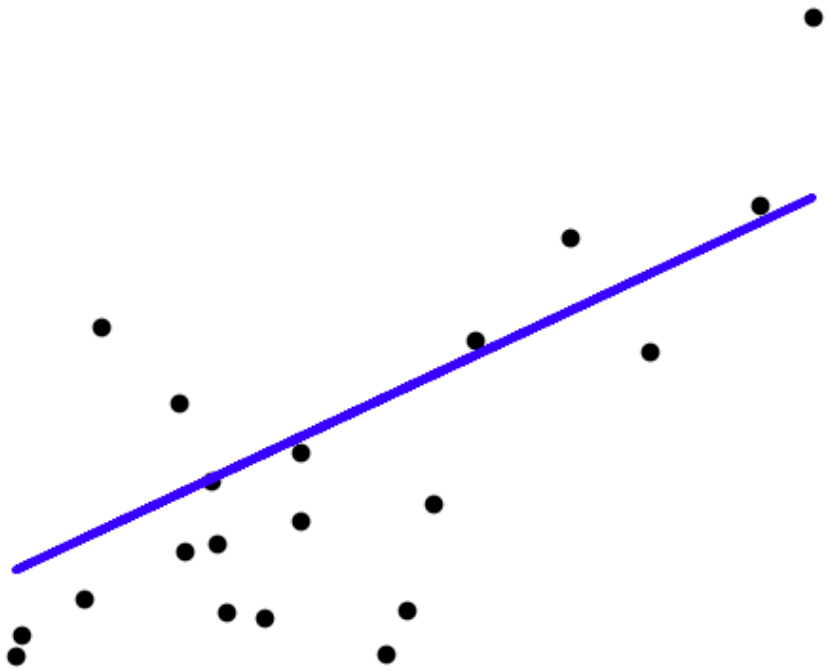

Figure 2.2: Linear Regression technique [3]

The mathematical formulation of the regression problem is:

$y = \beta_0 + \beta_1 x_1 + ... + \beta_m x_m + \epsilon$

where:

- y is the observed value/dependent variable
- the x_k are the input values/predictor variables
- the β_k are the weights that have to be found
- ϵ is a Gaussian-distributed error variable

Linear Regression fits a linear model with weights $\hat{\beta} = (\hat{\beta_1}, ..., \hat{\beta_m})$ by minimizing the residual sum of squares between the actual responses Y in the dataset and the predicted responses $X\beta$ where X is the matrix of input variables in the dataset. This is the Ordinary Least Squares method:

$\hat{\beta} = \min_\beta \|X\beta - Y\|^2$

Once the model has been fitted with the best possible weight β, the Linear Regression model can generate a continuous output variable given a set of input variables.

On the other hand, Logistic Regression is a classification model for a binary output variable problem. Indeed, the output variable of the model represents the log-odds score of the positive outcome in the classification.

$\ln(\frac{p(x)}{1-p(x)}) = \beta_0 + \beta_1 x_1 + ... + \beta_m x_m$

The log-odds score is a continuous variable which is used as input in a logistic function (Fig.2.3):

$\sigma(z) = \frac{1}{1+e^{-z}}$

The logistic function outputs a number between 0 and 1 which is chosen to be the probability of the positive classification outcome given the input variables. Setting and optimizing a cut-off probability allows for classification decisions to be made when presented with new input variables.

Generalized Linear Models are a popular technique as they are easy to implement and, in many classification or regression problems, assuming linearity between predictor variables and the outcome variable is sufficient to generate robust predictions. In addition to this, the fitted coefficients are interpretable, meaning that we can understand the direction and magnitude of association between each predictor variable and the outcome variable.

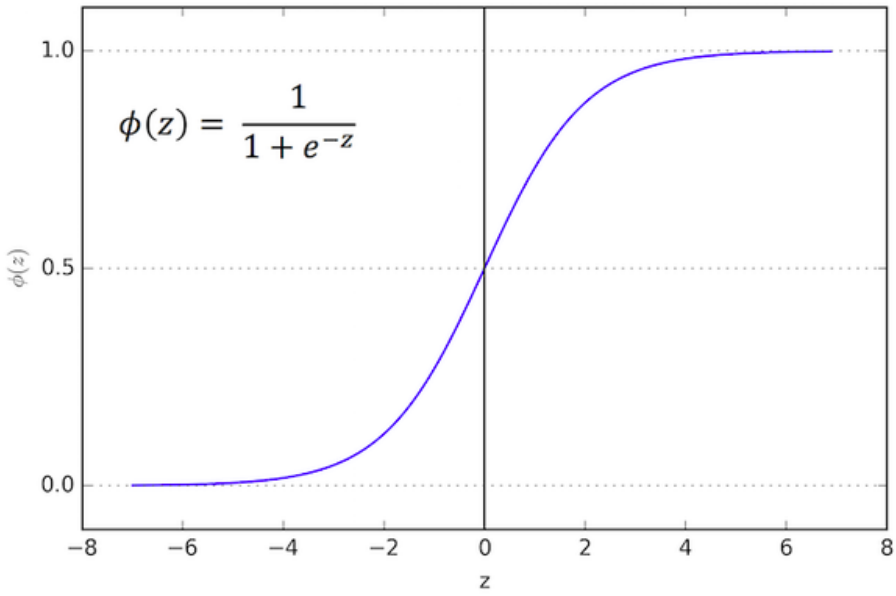

Figure 2.3: Logistic function [4]

However, using Generalized Linear Models can also have some disadvantages: the assumption that the input variables and output variable are linearly connected does not always fit the problem and can be too simple. Furthermore, if the input variables are highly-correlated, the performance of the model can be quite poor.

2.2.2 Decision Trees

Decision Trees are a popular Machine Learning technique to link input variables, represented in the tree's branches and nodes, with an output value represented in the tree's leaves. Trees can both be used in classification problems, by outputting a category label, or in regression problems, by outputting a real number. Decision Trees can be fitted using different algorithms, including the CART or ID3 decision tree algorithms which are the most popular. These algorithms use a mix of greedy searching and pruning so that the tree both fits and generalizes the data to new input/output pairs.

Decision Trees have the advantage that they scale very well with additional data, they are quite robust to irrelevant features and they are interpretable: the choices at each node allow us to understand the impact of each predictor variable towards the outcome. However, decision trees can often become inaccurate, especially when exposed to a large amount of training data as the tree will fall victim to overfitting. This happens when the model fits the training data very well but is not capable of generalizing to unseen data, thus resulting in poor predictive performance.

Random Forests operate by building a large amount of decision trees during training, taking a different part of the dataset as the training set for each tree. For classifi-

cation problems, the final output is the mode of the outputs of each decision tree, whereas for regression problems, the mean is taken. This technique is illustrated in Fig.2.4.

This results in a model with much better performance compared to a simple decision tree, thanks to less overfitting, but the model is less interpretable as the decisions at the nodes of the trees are different for each tree.

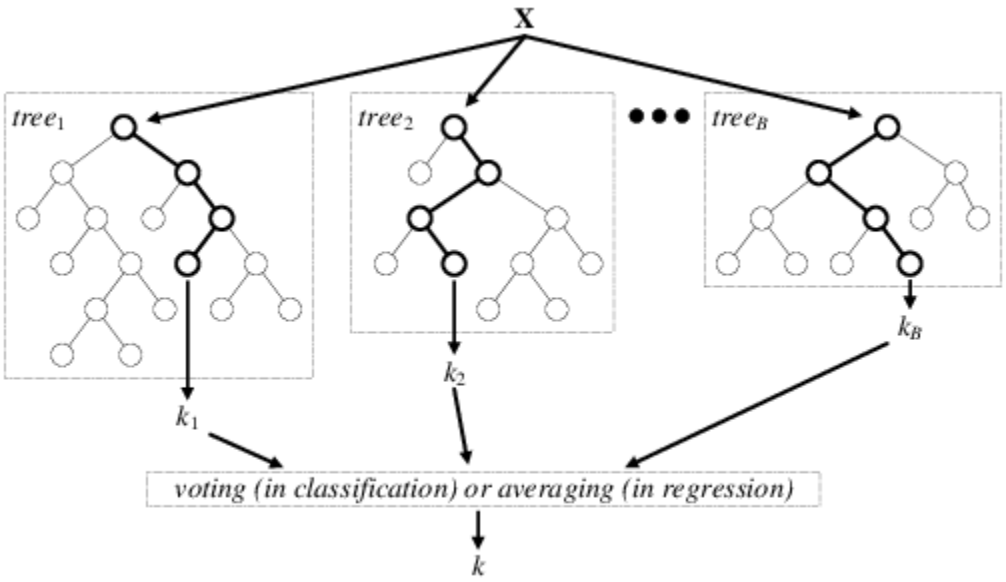

Figure 2.4: Random Forest technique [5]

2.2.3 Probabilistic classification

Probabilistic classifiers are Machine Learning methods capable of predicting the probability distribution over classes given input variables.

One of the most popular probabilistic classifiers is the Naive Bayes classifier. It is a probabilistic classification method based on the 'naive' assumption that every two different features are independent, thus enabling the use of Bayes' Theorem:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1,...,x_n|y)}{P(x_1,...,x_n)}$$

where:

- y is the target class
- the x_i are the predictor (input) variables

Using the independence assumption, this leads to the following:

$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$

MAP (Maximum A Posteriori) estimation enables the creation of a decision rule (rule to choose the class to return as output) by choosing the most probable hypothesis.

For Gaussian Naive Bayes, the likelihood $P(x_i|y)$ of the features is assumed to follow a Gaussian distribution, as illustrated in Fig.2.5:

$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2})$

where the σ_y and μ_y parameters are estimated using Maximum Likelihood Estimation.

The advantages of using Naive Bayes classifiers is that they are highly scalable when presented with large amounts of data: indeed, they take an approximately linear time to train when adding features. However, even though the classifier can be robust enough to ignore the naive assumption, the predicted probabilities are known to be somewhat inaccurate.

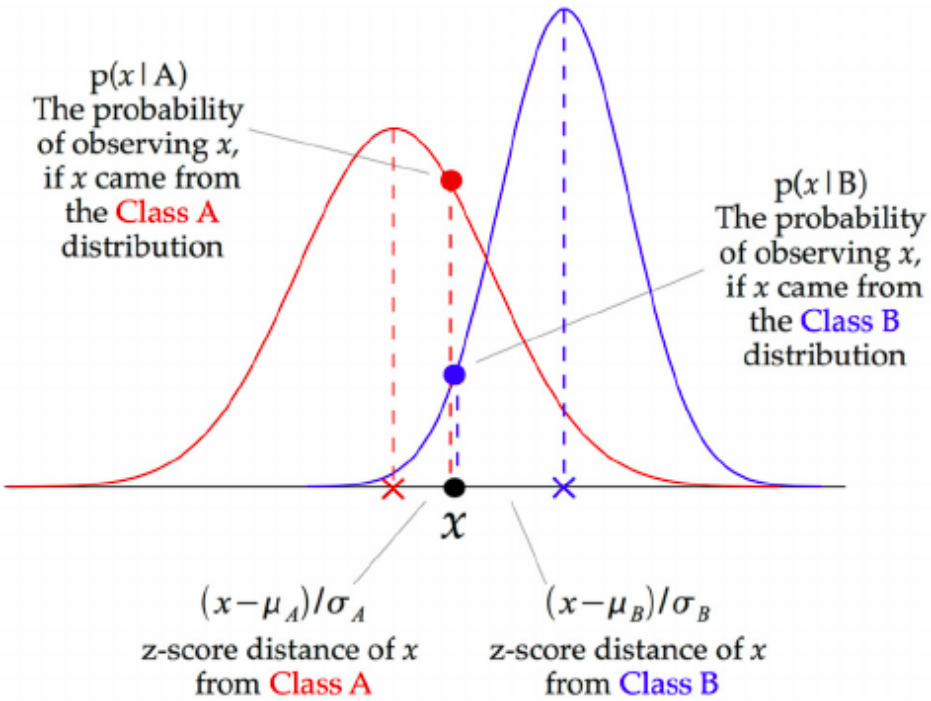

Figure 2.5: Gaussian Naive Bayes method [6]

2.2.4 Lazy learning

Lazy learning is a Machine Learning technique for which no model is actually built but the training data is generalized when new inputs are given. They are known to be best for large sets of data with a small number of features.

The K-nearest-neighbors algorithm is a lazy learning method for both classification and regression that takes the k nearest training examples in the feature space and outputs:

- for classification: the most common class among the k neighbors
- for regression: the average of the values for the k neighbors

The best choice for the k parameter depends on the training data set. Indeed, a higher value reduces the effect of noise in the data but makes the approximation less local with regards to other training data points. A classification example using the k-Nearest Neighbors method is illustrated in Fig.2.6.

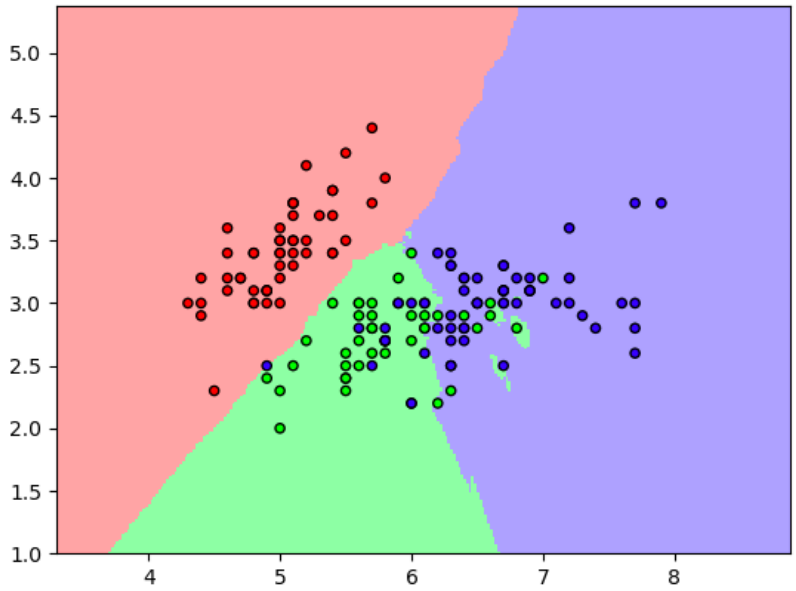

Figure 2.6: k-Nearest-Neighbors classification [7]

The main advantage of lazy learning methods is that the target function is approximated locally, which means that it is sensitive to the local structure of the training data. This allows these methods to easily deal with changes in the underlying data classification or regression distributions. However, these methods come with some drawbacks: space is required to store the training dataset as the algorithm will run through all training data examples to find those that are closest to the input values. This makes these techniques quite slow to evaluate when testing.

2.2.5 Support Vector Machines

Support Vector Machines (SVMs) are Machine Learning models for both classification and regression. An SVM model represents the training data as points in space so that examples falling in different categories are divided by a hyperplane (see Fig.2.7) that is as far as possible from the nearest data point.

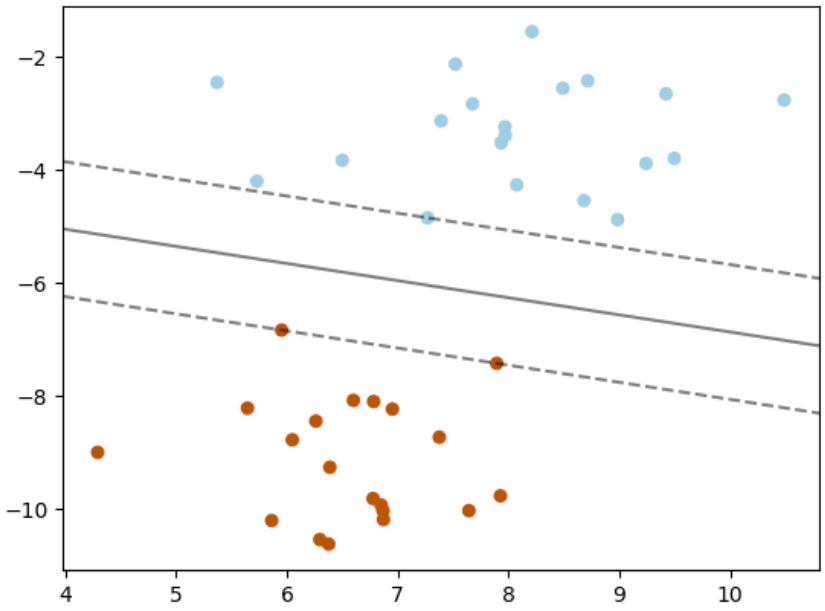

Figure 2.7: SVM Hyperplane for classification [8]

New inputs are mapped in the same way as the training data and classified as the category they fall into (which side of the hyperplane). When the data is not linearly separable, the kernel trick can be used, by using different possible kernel functions such as Radial Basis Functions (RBF) or polynomial functions, in order to map the data into high-dimensional feature spaces and find a suitable high-dimensional hyperplane.

The above classification problem can be extended to solving regression problems in a similar way, by depending only on a subset of the training data to generate a regression prediction.

Advantages for using Support Vector Machines include that they are effective in high-dimensional spaces, that they are memory efficient thanks to the use of a subset of training points in the decision function, and finally that they are versatile through the use of different possible kernel functions. On the other hand, using SVMs can have some disadvantages: they do not directly provide probability estimates for classification problems, and correctly optimising the kernel function and regularization term is essential to avoid overfitting.

2.2.6 Neural Network models

Neural Networks, also known as Artificial Neural Networks (ANNs), are systems that are based on a collection of nodes (neurons) that model at an algorithmic level the links between neurons in the human brain.

Each neuron can receive a signal from neurons and pass it on to other neurons. Two neurons are connected by an edge which has a weight assigned to it, which models the importance of this neuron's input to the other neuron's output. A neural network is usually composed of an input layer, with one neuron per input variable for the model, an output layer, composed of a single neuron which will give the classification or regression result, and a number of hidden layers between the two, containing a variable number of neurons in each layer. Fig.2.8 illustrates the architecture of a neural network with one hidden layer.

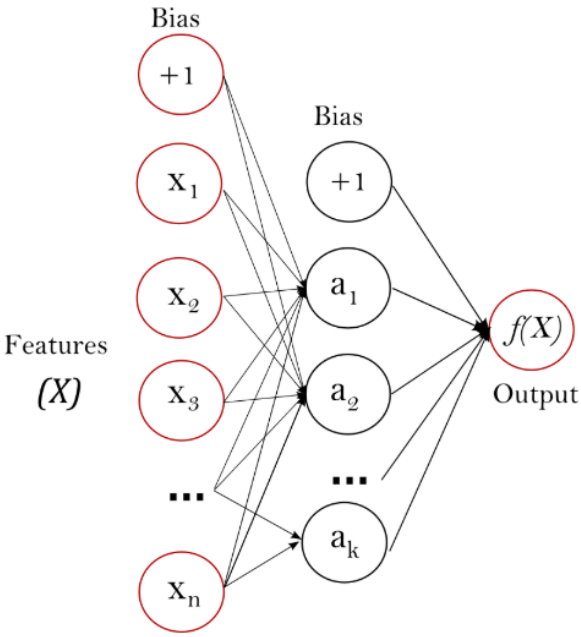

Figure 2.8: Neural Network diagram [9]

A neuron which receives an input p_j from another neuron then computes its activation value through its activation function f: $a_j = f(p_j)$. The neuron's output o_j is then generated through its output function f_o such that $o_j = f_o(a_j)$.

This leads us to the propagation function which calculates the input p_j that a neuron j receives in the network:

$p_j = \sum_i o_i w_{ij}$ where w_{ij} is the weight between neurons i and j

Training the Neural Network model involves setting the correct weight between each

two neurons in the system. This is done through the back-propagation algorithm which inputs a new training example, calculates gradient of the loss function (a function which quantifies the error between the Neural Network prediction and the actual value) with respect to the weights and updates the weights from the output layer all the way back to the input layer:

$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}}$

where:

- η is the learning rate and determines the magnitude of change for the weights
- C is the cost/loss function which depends on the learning type and neuron activation functions used

The advantages in using Neural Networks as classification or regression models are that they usually achieve a high level of predictive accuracy compared to other techniques. However, they require a very large amount of training data to optimise the model. In addition to this, neural networks are not guaranteed to converge to a single solution and therefore are not deterministic. Finally, Neural Networks are not interpretable: indeed, there are in general too many layers and neurons to understand the direction and magnitude of association of each input variable with the output variable through the different weights.

2.3 Past Research

We have separated the past research on football predictions in three categories. Firstly, we will look at the general landscape of football predictions and the Machine Learning techniques that have previously been used. Secondly, we will concentrate on research linked to using team ratings to improve predictions. Finally, we will talk about the papers that present models estimating the expected goals that a team is estimated to have obtained.

2.3.1 Football prediction landscape

Generating predictions for football scores has been an important research theme since the middle of the 20th century, with the first statistical modelling approaches and insights coming from Moroney (1956) [10] and Reep (1971) [11] who used both the Poisson distribution and negative binomial distribution to model the amount of goals scored in a football match, based on past team results.

However, it was only in 1974 that Hill proved that match results are not solely based on chance, but can be modeled and predicted using past data [12].

The first breakthrough came from Maher [13] in 1982 who used Poisson distributions to model home and away team attacking and defensive capabilities, and used

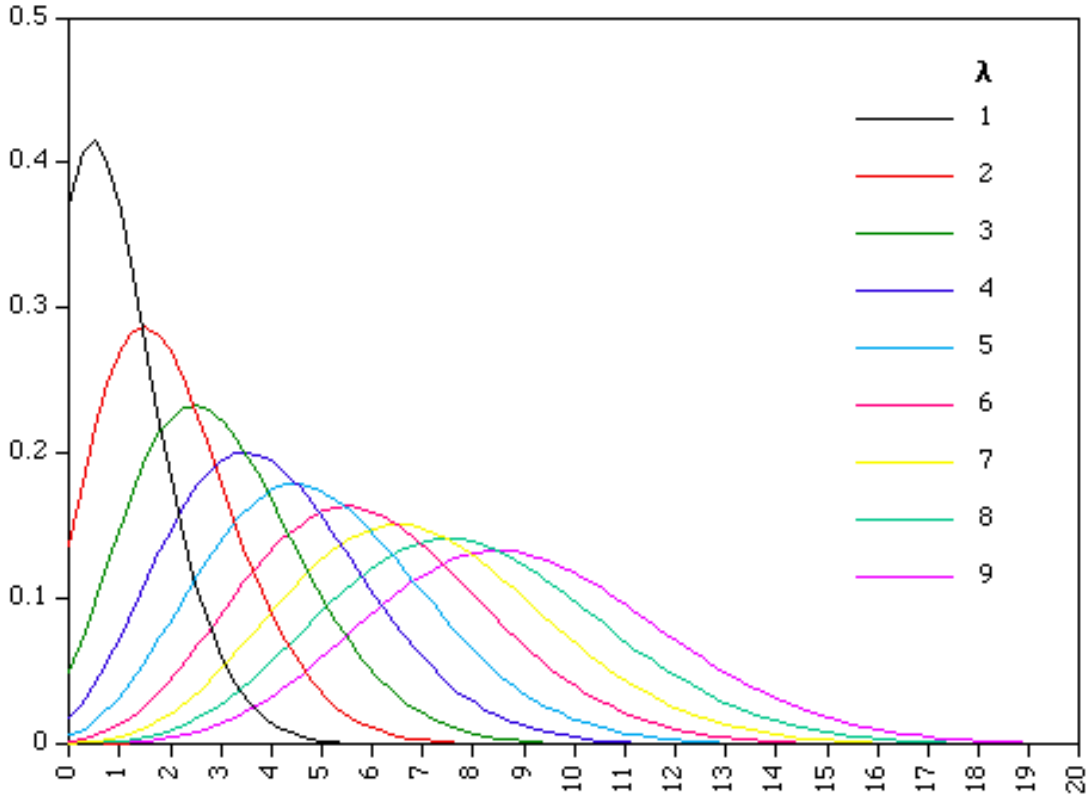

Figure 2.9: Poisson distribution for values of λ and k [15]

this to predict the mean number of goals for each team. Following this, Dixon and Coles [14] (1997) were the first to create a model capable of outputting probabilities for match results and scores, again following a Poisson distribution.

The Dixon and Coles model is still seen as a traditionally successful model, and we will use it as a benchmark against the models that we will be creating.

The Dixon and Coles model is based on a Poisson regression model, which means that an expected number of goals for each team are transformed into goal probabilities following the Poisson distribution (illustrated in Fig.2.9):

$P(k \text{ goals in match}) = e^{-\lambda}\frac{\lambda^k}{k!}$

where λ represents the expected number of goals in the match.

The Poisson distribution enables the calculation of the probability of scoring a certain number of goals for each team, which can then be converted into score probabilities and finally into match outcome probabilities.

Based on past results, the Dixon and Coles model calculates an attacking and de-

fensive rating for each team by computing Maximum Likelihood estimates of these ratings on past match results and uses a weighting function to exponentially down-weight past results based on the length of time that separates a result from the actual prediction time.

Rue and Salveson [16] (2000) chose to make the offensive and defensive parameters vary over time as more results happen, then using Monte Carlo simulation to generate predictions. In 2002, Crowder et al. [17] followed up on their work to create a more efficient update algorithm.

At the beginning of the 21st century, researchers started to model match results (win/draw/loss) directly rather than predicting the match scores and using them to create match result probabilities. For example, Forrest and Simmons (2000) [18] used a classifier model to directly predict the match result instead of predicting the goals scored by each time. This allowed them to avoid the challenge of interdependence between the two teams' scores.

During the same year, Kuypers [19] used variables pulled from a season's match results to generate a model capable of predicting future match results. He was also one of the first to look at the betting market and who tried to generate profitable betting strategies following the model he developed.

We have therefore seen that past research have tried to predict actual match scores as well as match results. It would be interesting in this project to look at the performance of generating a classification model for match outcome against a regression model for match scores.

We will now take a look at more recent research done on the subject, with the use of modern Machine Learning algorithms that will be interesting for us to investigate when trying different predictive models.

In 2005, Goddard [20] tried to predict football match results using an ordered probit regression model, using 15 years of results data as well as a few other explanatory variables such as the match significance as well as the geographical distance between the two teams. It was one of the first papers to look at other variables than actual match results. He compared the model predictions with the betting odds for the matches and found that there was the possibility of a positive betting return over time. Like Goddard, we will want to use other explanatory variables in our model than only match results, which will allow us to use different sets of features to try obtaining the best model possible.

It is also interesting to look at the algorithms used for predictions in other team sports: for example, in 2006, Hamadani [21] compared Logistic Regression and SVM with different kernels when predicting the result of NFL matches (American Football).

More recently, Adam (2016) [22] used a simple Generalised Linear Model, trained using gradient descent, to obtain match predictions and simulate the outcome of a tournament. He obtained good results, even with a limited set of features, and recommends to add more features and to use a feature selection process, which is something that will be interesting for us to do in this project considering the number of different features that will be available to us.

Tavakol (2016) [23] explored this idea even further: again using a Linear Model, he used historical player data as well as historical results between the two teams going head to head in order to generate predictions. Due to the large number of features available, he used feature extraction and aggregation techniques to reduce the number of features to an acceptable level to train a Linear Model.

There are multiple ways to reduce the number of features to train a Machine Learning model: for instance, Kampakis [24] used a hierarchical feature design to predict the outcome of cricket matches. On the other hand, in 2015, Tax et al. [25] combined dimensionality reduction techniques with Machine Learning algorithms to predict a Dutch football competition. They came to the conclusion that they obtained the best results for the PCA dimensionality reduction algorithm, coupled with a Naive Bayes or Multilayer Perceptron classifier. It will be interesting for us to try different dimensionality reduction techniques with our Machine Learning algorithms if we have a large number of features we choose to use. This also shows us that a large amount of data might not be required to build a Neural Network model and achieve interesting results.

Bayesian networks have been tested in multiple different recent research papers for predicting football results. In 2006, Joseph [26] built a Bayesian Network built on expert judgement and compared it with other objective algorithms, namely Decision Tree, Naive Bayesian Network, Statistics-based Bayesian Network and K-nearest-neighbours. He found that he obtained a better model performance for the Network built on expert judgement, however expert knowledge is needed and the model quickly becomes out of date.

Another type of Machine Learning technique that has been used for a little longer is an Artificial Neural Network (ANN). One of the first studies on ANN was made by Purucker in 1996 [27] to predict NFL games, who used backpropagation to train the network. One of the limitations of this study was the small amount of features used to train the network. In 2003, Kahn [28] extended the work of Purucker by adding more features to train the network and achieved much better results, confirming the theory that Artificial Neural Networks could be a good choice of technique to build sports predictive models.

Hucaljuk et al. (2011) [29] tested different Machine Learning techniques from multiple algorithm families to predict football scores:

- Naive Bayes (probabilistic classification)
- Bayesian Networks (probabilistic graphical model)
- LogitBoost (boosting algorithm)
- K-nearest-neighbours (lazy classification)
- Random Forest (decision tree)
- Artificial Neural Networks

They observed that they obtained the best results when using Artificial Neural Networks. This experiment is espacially interesting to us as we will want to test different algorithms in the same manner, to obtain the one that works the best for our data and features.

By looking at the past research done on the subject of building models to predict football match outcomes, we have been able to gain interesting information on the different techniques we should try using as well as the potential pitfalls we should avoid.

2.3.2 ELO ratings

We will now look at two areas of research that are especially important considering the objective of our project: firstly, we will want to look at how team ratings have previously been used to improve predictions.

The most famous rating system, still used today, was invented in 1978 by Elo [30] to rate chess players. The name "ELO rating" was kept for future uses, such as Buchdahl's 2003 paper [33] on using ELO ratings for football to update teams' relative strength. These types of rating have also been used for other sports predictions, such as tennis. Indeed, Boulier and Stekler (1999) [31] used computer-generated rankings to improve predictions for tennis matches, while Clarke and Dyte (2000) [32] used a logistic model with the difference in rankings to estimate match probabilities.

In 2010, Hvattuma used an ELO system [34] to derive covariates used in regression models in the goal of predicting football match outcomes. He tested two different types of ELO ratings, one taking in account the match result (win/loss/draw) and another only taking the actual score in account. He used different testing benchmarks to evaluate his predictions, which enabled him to see that he obtained better results using ELO ratings than the other benchmarks he ran his model against. Relative team strength ratings are very important to us in this project to encode past information and continuously update each team's relative strength as new results are added to the model.

We will want to explore different possibilities of keeping track and updating these team ratings over time. Multiple approaches have already been explored: to generate predictions for the football Euro 2016 tournament, Lasek (2016) [35] compared using ordinal regression ratings and using least squares ratings, combined with a Poisson distribution, to generate predictions and simulate the tournament a large number of times using Monte Carlo simulations. Viswanadha et al. (2017) [36] used player ratings rather than team ratings to predict the outcome of cricket matches using a Random Forest Classifier. Using player data to calculate relative player strengths and improve our model could be a potential extension to this project. Finally, in 2013, Fenton [37] came up with a new type of team rating, called pi-rating. This rating dynamically rates teams by looking at relative differences in scores over time. There are clearly different ways of evaluating relative team strength and weakness, and we will need to try different techniques for this project to try and obtain the best predictions we can.

2.3.3 Expected goals (xG) models

The second theme that is important to us for this project is building an expected goals model. This is a quite modern concept that aims to analyse match data to understand how many goals a team should have scored considering the statistics that have been observed during the match. Using an expected goals model enables us to eliminate some of the randomness that is associated with actual goals scored and get a better picture of a team's performance, and therefore strength.

In 2012, MacDonald [38] created an expected goals model to evaluate the performance of NHL (Hockey) matches, using two metrics:

- The Fenwick rating (shots and missed shots)
- The Corsi rating (shots, missed shots and blocked shots)

This enabled the evaluation of a team's performance to understand if, for example, they were wasteful with their goal opportunities, or if they did not manage to create enough goalscoring opportunities. Very good results were obtained for this expected goals model, with more efficient estimates for future results. A possible extension for this paper would be to use the opponent's data to calculate the expected goals. That is exactly what we will be trying to do with football: using both teams' data to understand how many goals each team were expected to score in a match, and use this value to improve future predictions.

In 2015, Lucey [40] used features from spatio-temporal data to calculate the likelihood of each shot of being a goal. This includes shot location, defender proximity, game phase, etc. This allows us to quantify a team's effectiveness during a game and generate an estimation for the number of goals they would have been expected to score. This example is particularly interesting for us, as we have geospatial data for shots and goals scored in matches. We will use that data to build an advanced expected goals model to better understand how a team has actually performed during

the game.

Similarly, Eggels (2016) [39] used classification techniques to classify each scoring opportunity into a probability of actually scoring the goal, using geospatial data for the shot as well as which part of the body was used by the player. Difference classification techniques were tested including logistic regression, decision trees and random forest. We will also need in this project to test different methods of quantifying the probability of scoring a goal for each opportunity.

Finally, one of the most interesting examples for out project comes from a website named FiveThirtyEight [41], who combine an expected goals model with a team rating system to predict the outcome of future football matches. In essence, they keep an offensive and defensive rating for each time depending on an average of goals scored and expected goals that are recorded in each game. This allows them to forecast future results and run Monte Carlo simulations to try and predict competition winners. It is interesting to note that the weight given to the expected goals decrease if a team is leading at the end of a match, which would be an hypothesis for us to explore.

For our project, we will take inspiration from these methods, but we will improve the expected goals model by testing different Machine Learning algorithms and adding features in order to try and output the best possible predictions.

Chapter 3

Dataset

3.1 Data origin

We have obtained a dataset from the Kaggle Data Science website called the 'Kaggle European Soccer Database' [42]. This database has been made publicly available and regroups data from three different sources, which have been scraped and collected in a usable database:

- Match scores, lineups, events: http://football-data.mx-api.enetscores.com/
- Betting odds: http://www.football-data.co.uk/
- Players and team attributes from EA Sports FIFA games: http://sofifa.com/

It includes the following:

- Data from more than 25,000 men's professional football games
- More than 10,000 players
- From the main football championships of 11 European countries
- From 2008 to 2016
- Betting odds from various popular bookmakers
- Team lineups and formations
- Detailed match events (goals, possession, corners, crosses, fouls, shots, etc.) with additional information to extract such as event location on the pitch (with coordinates) and event time during the match.

We will only be using 5 leagues over two seasons as they possess geographical data for match events that we will need to build our expected goals models:

- English Premier League
- French Ligue 1

- German Bundesliga
- Spanish Liga
- Italian Serie A

We will only be using data from the 2014/2015 as well as 2015/2016 seasons as they are the most recent seasons available in the database and the only ones containing the data that we need.
This gives us usable dataset of:

- 3,800 matches from the top 5 European leagues
- 88,340 shots to analyse
- More than 100 different teams

3.2 Data pre-processing

An important step before building our model is to analyse and pre-process the data to make sure that it is in a usable format for us to use when training and testing different models.

Three pre-processing steps were taken in order to achieve this:

- Part of the data that we needed, namely all match events such as goals, possession, corners, etc. was originally in XML format in the database. We therefore built a script in R to extract this data and store it in new tables, linked to the 'Matches' table thanks to a foreign key mapping to the match ID. An extract of the XML for a goal in one match is presented below:

```
<goal>
   <value>
      <comment>n</comment>
      <stats>
         <goals>1</goals>
         <shoton>1</shoton>
      </stats>
      <event_incident_typefk>406</event_incident_typefk>
      <coordinates>
         <value>18</value>
         <value>67</value>
      </coordinates>
      <elapsed>35</elapsed>
      <player2>35345</player2>
      <subtype>header</subtype>
      <player1>26777</player1>
```

```
        <sortorder>1</sortorder>
        <team>9826</team>
        <id>3647567</id>
        <n>200</n>
        <type>goal</type>
        <goal_type>n</goal_type>
    </value>
</goal>
```

- Some data elements were set to NULL, which led to us deleting some unusable rows and in other cases to us imputing values to be able to use the maximum possible amount of data. For instance, the possession value was missing for some games, so we entered a balanced value of 50% possession for each team in this case.
- Finally, having extracted the geographical coordinates for each shot in the dataset, we generated distance and angle to goal values which we added to our goals and shots database tables.
  The following formula was used to generate the distance to goal of a shot, where the coordinates of the goal are (lat=0, lon=23):

  $$D(lat, lon) = \sqrt{lat^2 + (lon - 23)^2}$$

  The following formula was used to generate the angle of a shot:

  $$A(lat, lon) = tan^{-1}(\frac{|lon-23|}{lat})$$

3.3 Data features

A simplified diagram of the database structure and features is presented in Fig.3.1. We will now present the different tables and features that we have in our database and that we can use in our models:

- Matches table
  - ID
  - League ID
  - Season
  - Date
  - Home team ID
  - Away team ID

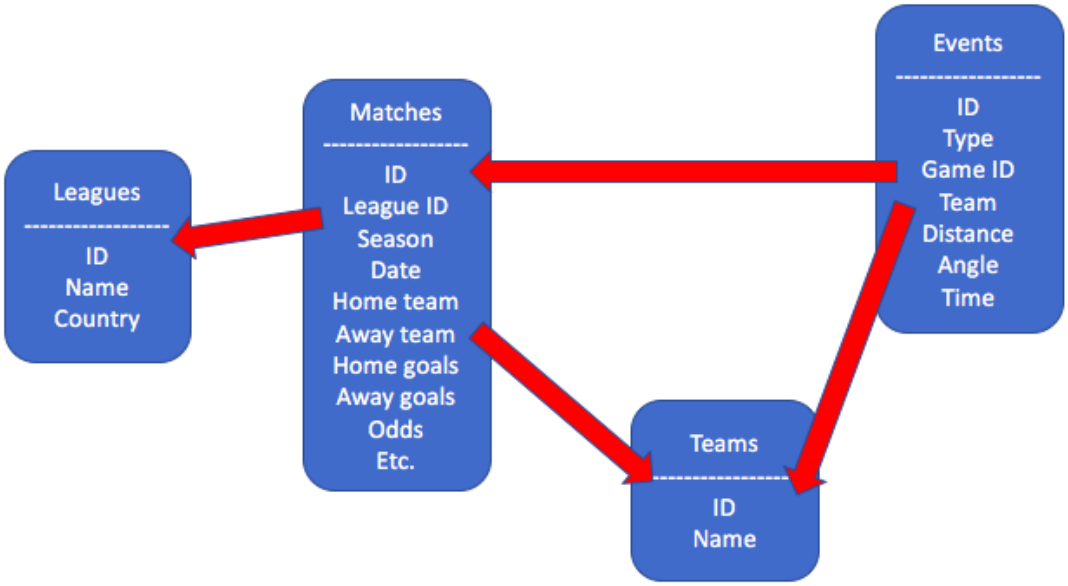

Figure 3.1: Structure of the Database

  - Home team goals scored
  - Away team goals scored
  - Home team possession
  - Away team possession
  - Home win odds
  - Draw odds
  - Away win odds

- Events tables:
  Here is a list of the different match events tables that we have extracted:

  - Goals
  - Shots on target
  - Shots off target
  - Corners
  - Crosses
  - Cards
  - Fouls

  For each of these match event tables, we have the following features:

  - ID
  - Type

- Subtype
- Game ID
- Team ID
- Player ID
- Distance to goal (only for goals and shots)
- Angle to goal (only for goals and shots)
- Time elapsed

Extracts of the database and row values examples are available in the Appendix.

Chapter 4

Design

In this chapter, we will present the general design of our model and the choices we have made.

Our model is a mixture of multiple regression and classification algorithms used to generate different metrics that are finally used as inputs for our classification model (for match outcomes) and regression model (for match scores).

4.1 Model components

In this section, we will introduce the different components of our model and explain their role.

A diagram of our different components and how they are linked is presented in Fig.4.1.

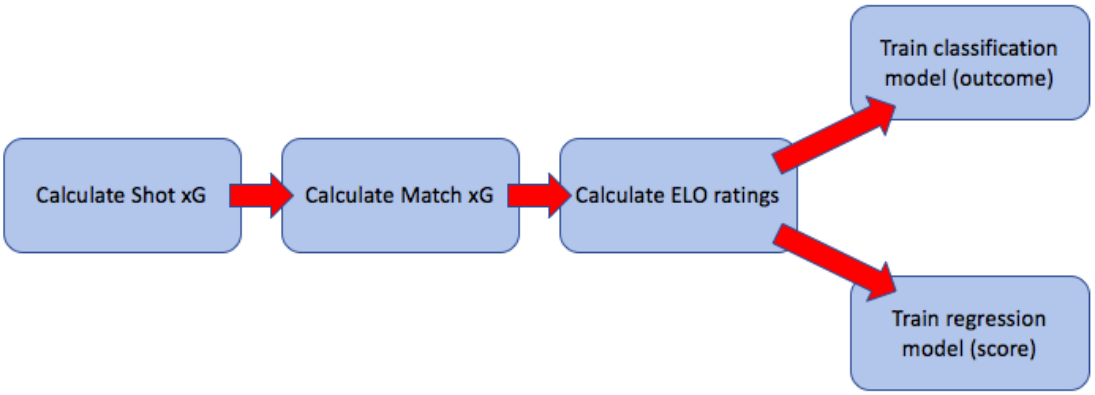

Figure 4.1: Diagram of model components

We have five main model components which we will present one by one:

- **Shot xG generation**:
  This component's objective is to generate an expected goal value for each shot representing the probability that the shot results in a goal, with some adjustments to reflect specific match situations.
- **Match xG generation**:
  This component's objective is to generate a shot-based expected goals value for each match by looking at the expected goals values for each shot during that match. In addition to this, a non-shot-based expected goals value is generated using match information other than shots.
- **ELO calculation**:
  This component's objective is to generate offensive and defensive team ELO ratings after each match using expected goals values and the actual performance. ELO ratings are recalculated after each match and the team ratings are stored for use in our predictive classification and regression models.
- **Classification model training**:
  This component's objective is to train and test a classification model capable of taking two teams' ELO ratings and generating a prediction for a match between these two teams between a home team win, a draw and an away team win.
- **Regression model training**:
  This component's objective is to train and test a regression model capable of taking two teams' ELO ratings and generating a prediction for the expected number of goals each team will score. These values are then used to generate a prediction for the match outcome.

4.2 Model choices

We will now dive into more detail for each component of the model, presenting the choices we have made and explaining how each value is obtained.

- **Shot xG generation**:
  - To generate an expected goals value for each shot, we firstly take all shots in our database, including goals, shots on target that did not result in a goal (shots that were stopped on their way to goal by another player) and shots off target (shot attempts that do not go in the direction of the goal.
  - We then separate each shot into a specific category depending on the type of shot:
    * 'Normal' shots: lobs, shots from distance, deflected shots, blocked shots, etc.

    - Volleys (shots hit with the foot when the ball is still in the air)
    - Headers (shots hit with the head)
    - Direct free-kicks (shots that result from a free kick that can be directly shot at goal)
    - Indirect free-kicks (shots that result from a free kick that cannot be directly shot at goal)
    - Bicycle kicks (shots taken above the level of the shoulders)
    - Own goals (goals scored by players into their own net)
    - Penalties
- Next, we assign an outcome value of 1 to each shot that resulted in a goal, and 0 to each shot that did not result in a goal.
- This allows us to build a separate classification model for each shot type, taking as predictor variables the distance to goal and the angle to goal, with the outcome variable reflecting if the shot resulted in a goal or not. We have chosen to build a separate classification model for each shot type as we make the assumption that each type of these goals have different probabilities of resulting in a goal if taken from the same exact location on the pitch.
- For each shot, we calculate the probability of scoring as an expected goals value using the suitable classification model, entering as input the shot distance and angle to goal, as illustrated by Fig.4.2.

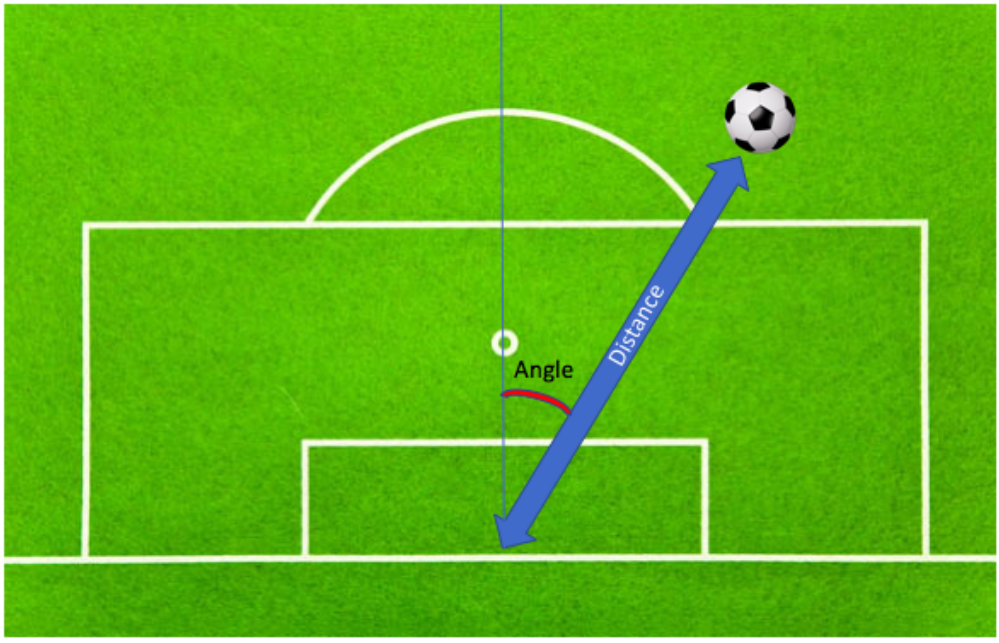

Figure 4.2: Diagram of shot distance and angle used to predict xG

– Once we have our expected goals value for a shot, we proceed with an adjustment based on if the team is already leading and by how many goals. We decide to make different adjustments if a team is winning by a single goal, with the match final outcome still in the balance, and when a team is winning by two goals or more, making the lead more comfortable for the side that is winning. Adjusting the expected goals value is done to reflect the fact that a team that is losing will become more attacking to try and equalise or gain a result from the match, thus making it easier for the winning team to attack and score goals, and making the shot worth less than if the teams had the same number of goals for example.

  We therefore decide to set a time t in the match after which the value of a shot will decrease linearly until the end of the game towards a weight k between 0 and 1. We will later try different times for which to start decreasing shot values, with the hypothesis that the time where substitutions are usually made could be a good starting point as managers often bring on a more attacking player for a more defensive one in order to catch up in the score. We will also try different values k if a team is leading by 1 goal or by 2 goals and more. Fig.4.3 illustrates our hypothesis by showing the value of a goal as the game is progressing.

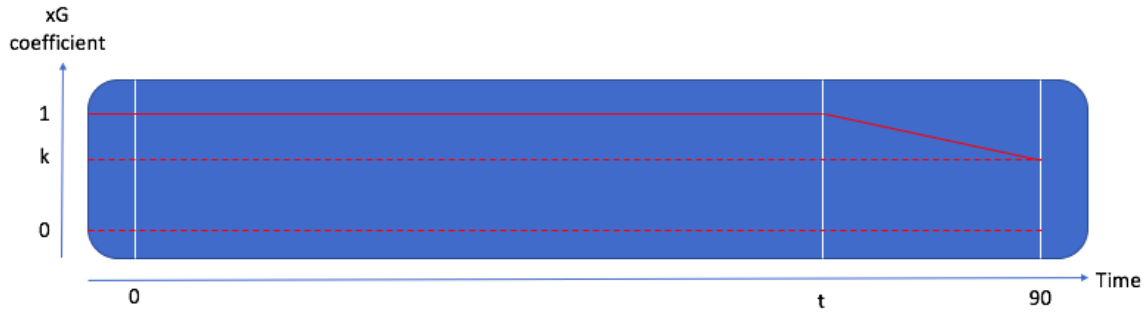

Figure 4.3: Diagram of value of shot xG over time when team is leading

– Finally, we proceed with a final adjustment, which is to decrease the value of a shot by a certain coefficient if the player's team has a higher number of players on the pitch due to a red card given to one of the opposing team's players. This is due to the fact that a team with less players will have on average a worse performance than if the two teams have the same number of players, making it easier to attack and score for the opponent. We also decide to increase the value of a shot by another coefficient if the player's teams have a lower number of players on the pitch, for the same reason as mentioned above.

- **Match xG generation**:
  - Firstly, for each match, we sum of each team's shot expected goals values to generate a shot-based expected goals value for each team at the end of a match.
  - We also decide to build a non-shot-based expected goals metric for each team by evaluating other information than shots:
    * possession
    * number of corners
    * number of crosses
    * number of yellow cards
    * number of yellow cards for the opponent
    * number of fouls
    * number
  - We therefore decide to train a regression model which takes as input these previously mentioned statistics for each team and takes as output the number of actual goals the team has scored. We take this decision as dangerous situations on the pitch that can result in goals can often not lead to a shot being taken. We believe that using other information allows us to better assess a team's performance and their likelihood to score goals.
  - Once we have built this regression model, we look at each match and generate a non-shot-based expected goals metric for the home team and for the away team depending on their in-match statistics.
  - Both the shot-based and non-shot-based expected goal values will be used to recalculate ELO ratings after each match.
- **ELO calculation**:
  - Our main method for ELO calculation is to store team ratings and modify the ratings after each match has been completed, in order to keep a metric of team strength and weakness.
  - Each team will possess 6 different ratings that will be stored and updated:
    * General offensive rating
    * General defensive rating
    * Home offensive rating
    * Away offensive rating
    * Home defensive rating
    * Away defensive rating
  - Each offensive rating represents the number of goals the team would be expected to score against an average team at that point in time, while each defensive rating represents the number of goals the team would be expected to conceded against an average team at that point in time.

- For each match played at home, a team will have their general and home ratings updated, and for each match played away, a team will have their general and home ratings updated. This allows us to keep track of specific strengths, for instance a team might be extremely strong at home but very poor away, which will result in strong home ratings but average general ratings.
- The ratings are recalculated using the following formula:

  $$ELO(t+1) = ELO(t) + \eta(actual(t) - expected(t))$$

  In other words, we update an ELO rating after each match by taking the pre-match ELO rating, and adding a weighted difference between the actual number of expected goals and the expected amount of expected goals. If the actual number of expected goals is greater than the expected amount, then the ELO rating will increase, otherwise it will decrease. This works for both offensive and defensive ratings, by simply taking the opponent's expected goals values to recalculate the defensive ratings. This formula works in a similar way to a back-propagation algorithm which adjusts a value at each iteration using a coefficient, similar to a learning rate.

  The actual number of expected goals value is a weighted average of a team's shot-based expected goals value and non-shot-based expected goals value recorded in the match, which were generated previously. The weights given to each metric will be optimised when training the model.

  The expected number of expected goals value is a weighted average of:
  - a team's general offensive rating
  - this team's home/away-specific offensive rating
  - the opposing team's general defensive rating
  - the opposing team's home/away-specific defensive rating

  The weights used for this calculation will be optimised when training the model.
- We had to decide how to set the ELO ratings for the first time, before the start of the 2014/2015 season for which we started to generate expected goals values. We decided to take set each team's ELO ratings to the average number of goals scored/conceded during the previous season (2013/2014). For promoted teams (teams that played in the lower division in the previous season), we took an average of the relegated teams' average number of goals scored/conceded. This follows the assumption that promoted teams are in general part of the teams that have the most difficulties when arriving in the country's top football league.
- Finally, we adjust every team's ratings between two seasons towards the mean of the league by a certain coefficient that we will attempt to optimise for our models.

  - Having recalculated the ELO ratings of each team after each match, we link them to the following match that a team is playing so that the ratings can be used to predict the result of the following match.

- **Classification model training**:
  - Now that we have generated our ELO ratings for each team and each match, we can train a classification model to predict the outcome (home win/draw/away win) of a match.
  - We use both teams' offensive and defensive ratings, both general and home/away-specific, as well as an interaction term of each team's general offensive rating multiplied by the opponent's defensive rating, as predictor variables, and the actual outcome of the match as dependent variable to train our model.
  - We use cross-validation to train our model multiple times using different training sets, and generating performance metrics on the test set for which we then take the average for all cross-validation training runs.

- **Regression model training**:
  - In parallel, we also train a regression model to predict the actual score of the match.
  - We first double the training data, to take our training examples' ELO ratings from the point of view of the home team attacking and then from the point of view of the away team attacking.
  - We add a variable to each data point to represent if the team is playing at home (1) or away (0) in order to model the home advantage in matches.
  - We use these ratings as predictor variables for our regression model, with the actual number of goals scored as outcome variable.
  - Again, we use cross-validation to train our model and generate regression testing metrics.
  - Finally, we use the predictions for the number of goals by each team to model the match score probabilities using a Poisson distribution, which we then sum to get the probability of each outcome in the match. We then use these probabilities to generate classification metrics and evaluate the classification performance of our regression model.

Chapter 5

Implementation

5.1 General pipeline

In this section, we will present the general pipeline that we have set up in order to easily be able to train and test different models and compare their performances.

The pipeline we have created is illustrated in Fig.5.1.

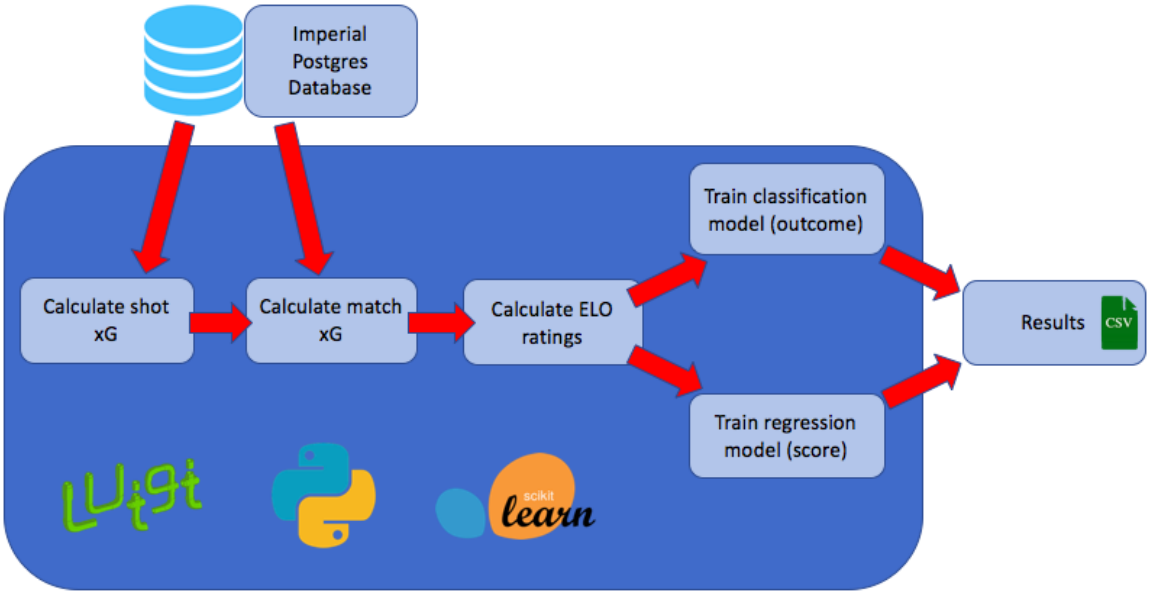

Figure 5.1: General project pipeline

The pipeline consists of the following elements:

- Our database tables are stored in the Imperial Postgres database which allows us to easily read and write to different tables while having access to a large amount of memory space without being limited. We have two versions of our pipeline: one of them writes the results to the database after each value generation (shot xG, match xG, etc.), which allows us to independently test

and tweak different parts of the model without having to recalculate values every time. The other version of the pipeline simply reads the data from the database but does not write to it, allowing us to run multiple possible versions of the model, with different parameters, in parallel. This proves to be especially useful when optimising the model parameters.

- We have used Python 2 to write out scripts for each component in the pipeline, using the Pandas library [43] which gives us many useful Data Science tools and allows us to extract data from the Postgres database and use DataFrames to manipulate the data throughout the pipeline.
- We used the Scikit-Learn [44] library for Python for their easy implementation of different Machine Learning algorithms which allowed us to try and optimise different classification and regression techniques that we wanted to test.
- We created a CSV file to which we could write after each training/testing run to store the model performance metrics and keep track of which models performed best.
- Finally, we used the Luigi [45] library for Python in order to link the different components of our pipeline together. This will be presented in more detail in the next section of the report.

5.2 Luigi

Luigi [45] is an open-source Python created by Spotify which allows its users to build pipelines of batch jobs, with an integrated visualiser page. We used Luigi to build our pipeline of model training and testing so that we could easily keep track of the different Machine Learning techniques and parameter choices we were going to test.

Luigi comes with an integrated dependency handler as well as error handling in case one component of our pipeline failed during model training and testing. The different tasks and errors can be visualised thanks to a local web server, as illustrated in Fig.5.2.

We did not use Luigi to its full potential in creating batch jobs with many different parameters as our approach was more incremental with only a few testing runs every time we tweaked our model. However, Luigi allowed us to test different parts of the pipeline independently, for instance testing different ELO calculation methods without having to recalculate expected goal values.

Fig.5.3 illustrates the dependency graph for an example where we wanted to recalculate all expected goals values and train new models. We can see the contrast with Fig.5.4 where we simply wanted to try new ELO calculation methods without having to recalculate xG metrics for each shot and match.

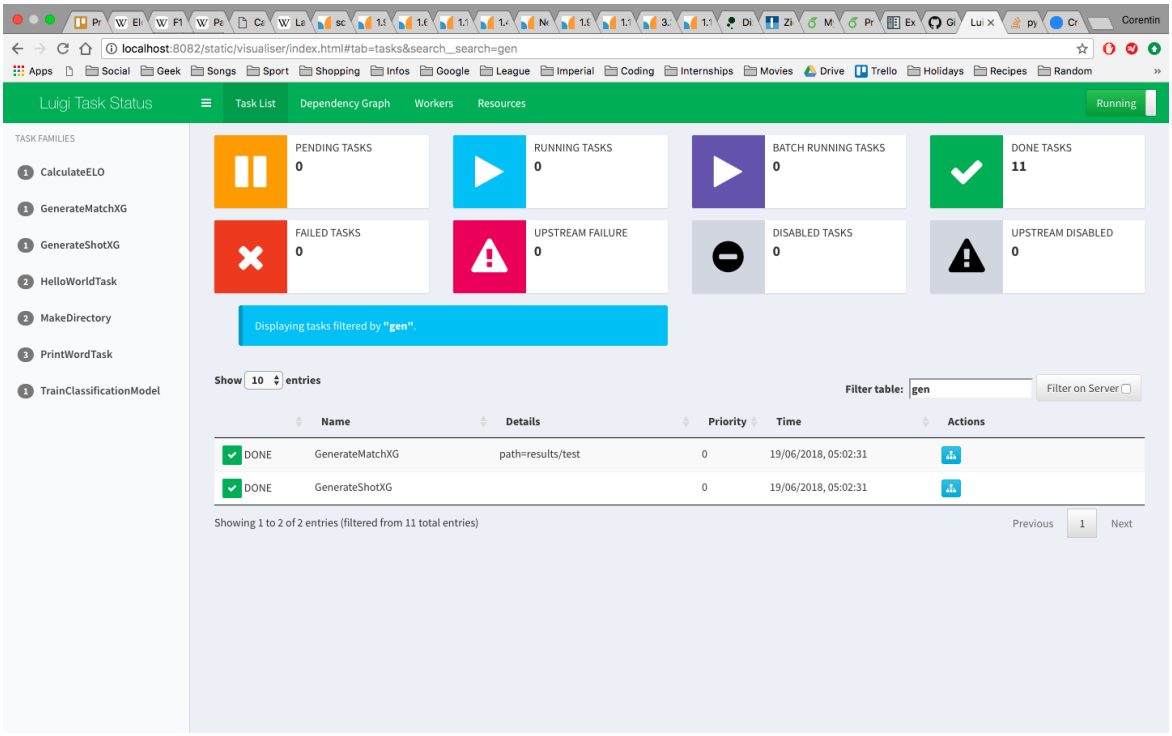

Figure 5.2: Luigi visualiser page

Figure 5.3: Luigi dependency graph

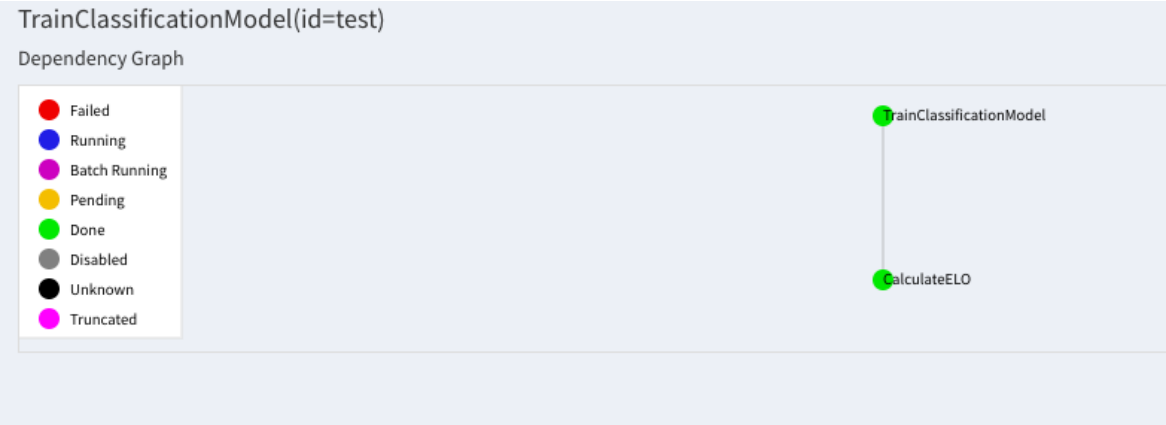

Figure 5.4: Luigi dependency graph without xG calculation

Chapter 6

Experimentation & Optimisation

In this chapter, we will present the experiments that we undertook to optimise our model and obtain the best possible performance.

6.1 Testing

In this section, we will look at different testing methods and metrics that we have used as a base to optimise our model.

6.1.1 Cross-validation testing

Cross validation is a method when training a model in order to avoid overfitting, which is the situation when the model fits the training data very well but cannot generalise to data that has not been seen before.

Cross-validation therefore uses a split between a training data set and a testing data set. The training set is used to train the model, whereas the predictive performance of the trained model is then tested on the test set. To keep a high number of samples with which to train the model, cross-validation runs the training and testing routines multiple times, with a different part of the dataset used as test data for each iteration. The model evaluation metrics are then averaged across all training iterations. This principle is illustrated in Fig.6.1.

The disadvantage of using cross-validation to train and test a new model is that the training time is multiplied by the number of cross-validation iterations. In general, though, it is worth sacrificing model training speed in order to reach a more robust final model that is less prone to overfitting.

6.1.2 Testing metrics

Different testing metrics have been used in order to evaluate the performance of our regression and classification models.

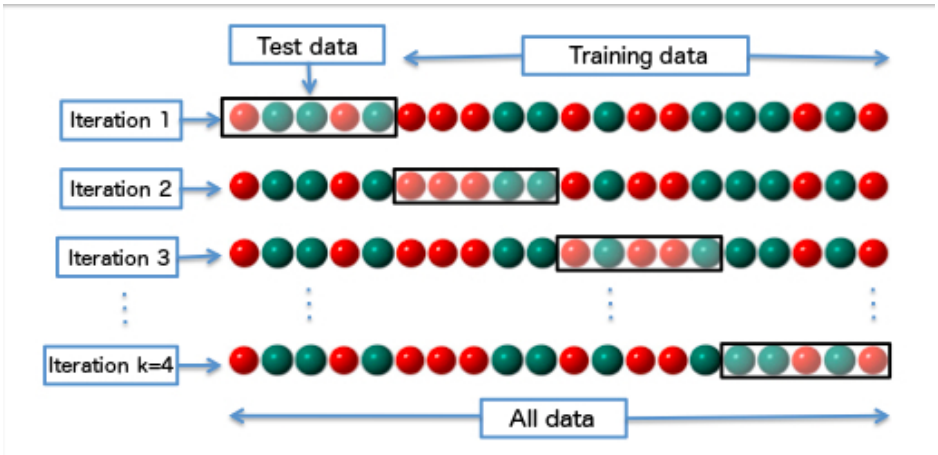

Figure 6.1: Diagram of the cross-validation model training technique [46]

For our regression model, we have looked at two different metrics:

- **Mean Absolute Error**:

  $$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

  where:

  - y_i is the actual value
  - $\hat{y}_i$ is the value predicted by the regression model

  Using the Mean Absolute Error allows us to interpret our model's performance as we know the average distance between the actual value and the model's predicted value. However, it is not always the best metric on which to make model decisions.

- **Root Mean Squared Error**:

  $$RMSE = \sqrt{(\hat{y}_i - y_i)^2}$$

  where:

  - y_i is the actual value
  - $\hat{y}_i$ is the value predicted by the regression model

  The Root Mean Squared Error is a metric that, compared to the Mean Absolute Error, is not interpretable. However, the RMSE gives a larger weight to predictions that are far away from the actual value. As we want our model to obtain a robust predictive performance for all possible examples, we will want to penalise these large errors and we have therefore chosen both RMSE and MAE as the testing metrics we want to minimise.

For our classification model, we have also looked at two different metrics:

- **Accuracy**:

  $Acc = \frac{n_true}{n_total}$

  where:

  - n_true is the number of examples that the classifier has correctly predicted
  - n_total is the total number of examples

  Accuracy is a simple metric that allows us to understand the performance of our classification model by seeing what proportion of examples it has correctly predicted. The accuracy is always between 0 and 1, and better performance is achieved for higher accuracy. However, the accuracy metric is missing some important information to quantify our classifier's performance.

- **F1 Score**:

  $F1 = 2\frac{precision*recall}{precision+recall}$

  where:

  - $precision = \frac{tp}{tp+fp}$
    tp is the number of true positive classifications (number of examples correctly classified as positive)
    fp is the number of false positive classifications (number of examples wrongly classified as positive)
  - $recall = \frac{tp}{tp+fn}$
    tp is the number of true positive classifications
    fn is the number of false negative classifications (number of examples wrongly classified as negative)

  The F1 score is less easy to understand and interpret compared to the accuracy, especially for a multilabel classification problem as we have here (three possible outcomes). In this case, the F1 score is calculated for each category and the average is taken as the final F1 score.

  The F1 score is in general recognised to be more useful than accuracy as it takes false positives and false negatives into account. It is especially useful for uneven class distribution, as we have here (home wins happen more often than draws or away wins). We will therefore use the F1 score as well as Accuracy as metrics to maximise in order to choose the classification model with the best performance.

6.2 Choice of models

In this section, we will look at the experiments undertaken to choose the regression and classification models we will be using in our final model at different stages of our pipeline.

- **Shot xG classification model**
  We have seen that we need a classification model for each shot type capable of predicting the probability of the shot resulting in a goal given the distance to goal and the angle to goal. We will therefore test and compare four different classification models capable of generating probability predictions:
  - Logistic Regression
  - k-Nearest-Neighbors
  - Gaussian Naive Bayes
  - Random Forest Classifier

  The results we obtained using the same parameters, keeping all other elements of the pipeline constant and training a new general classification model, are presented in Fig.6.2. We can see that we obtain the best accuracy by using the Gaussian Naive Bayes model, which we will choose in order to build our shot xG classification models.

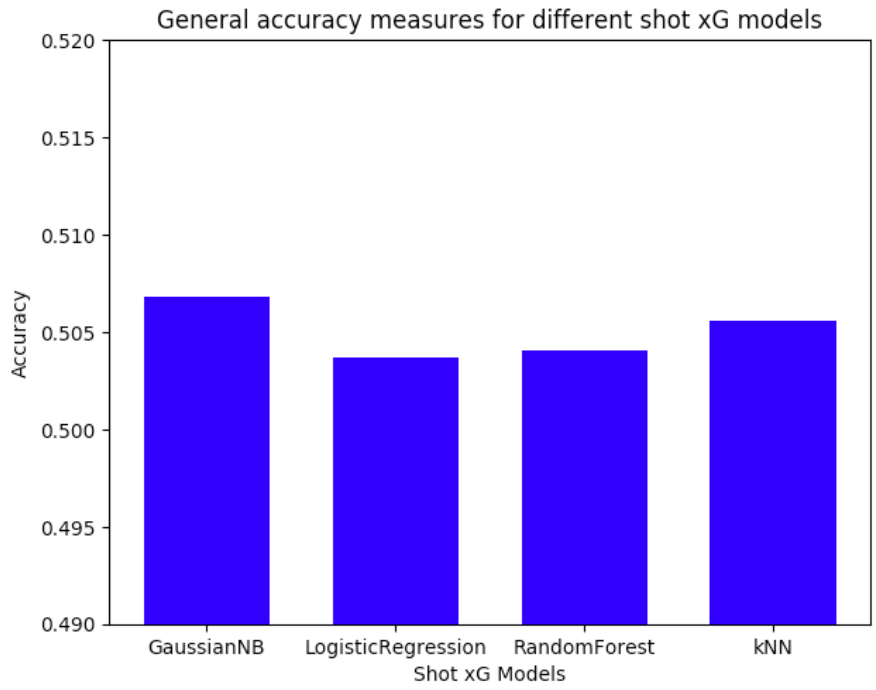

Figure 6.2: Accuracy for different shot xG classification models

- **Non-shot-based xG regression model**
  We have seen that we need a regression model capable of predicting an expected number of goals given the different in-game statistics excluding shots. We will therefore test and compare four different regression models:

  – Linear Regression
  – SVM Regressor with RBF kernel
  – SVM Regressor with Linear kernel
  – Random Forest Regressor (with a forest of 10 decision trees)

  The results we obtained using the same parameters, keeping all other elements of the pipeline constant and training a new general classification model, are presented in Fig.6.3. We can see that we obtain the best accuracy by using the Random Forest Regressor model with a forest size of 10 decision trees, which we will choose to implement in order to build our non-shot-based xG regression model.

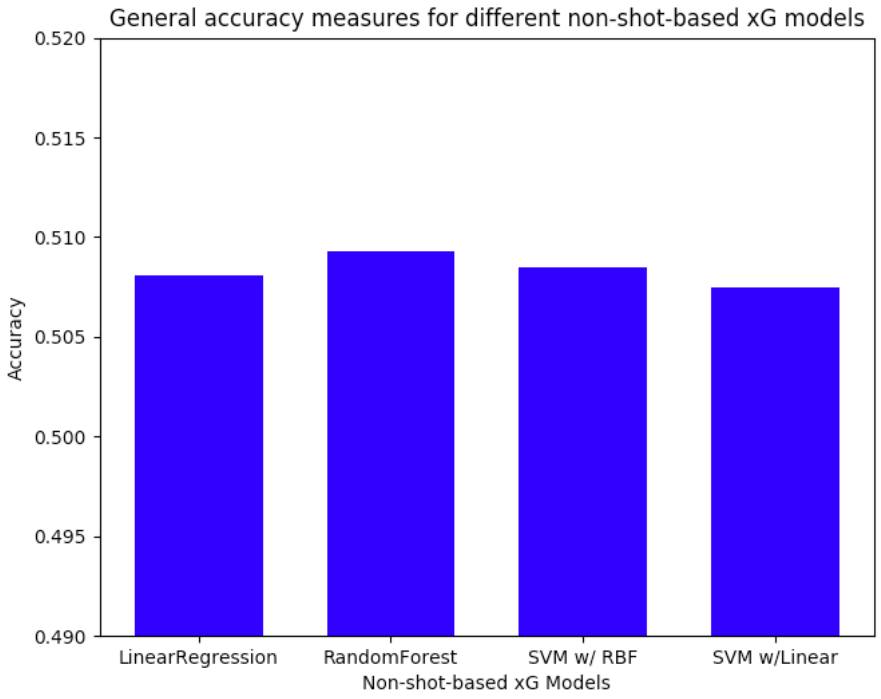

Figure 6.3: Accuracy for different non-shot-based xG regression models

- **General outcome classification model**
  We have seen that we need a classification model capable of predicting the outcome of a game given the different ELO ratings of the two teams playing against each other. We will therefore test and compare eight different classification models capable of generating probability predictions:

- Logistic Regression
- k-Nearest-Neighbors
- Gaussian Naive Bayes
- Random Forest Classifier
- SVM with RBF kernel
- SVM with polynomial kernel
- SVM with linear kernel
- Neural Network (Multilayer Perceptron with one hidden layer of 100 neurons: this has been optimised as the best Neural Network architecture for this task)

The results we obtained using the same parameters, keeping all other elements of the pipeline constant and training our match outcome classification model, are presented in Fig.6.4. We can see that we obtain the best accuracy by using the SVM model with a linear kernel function, which we will choose in order to build our match outcome classification models.

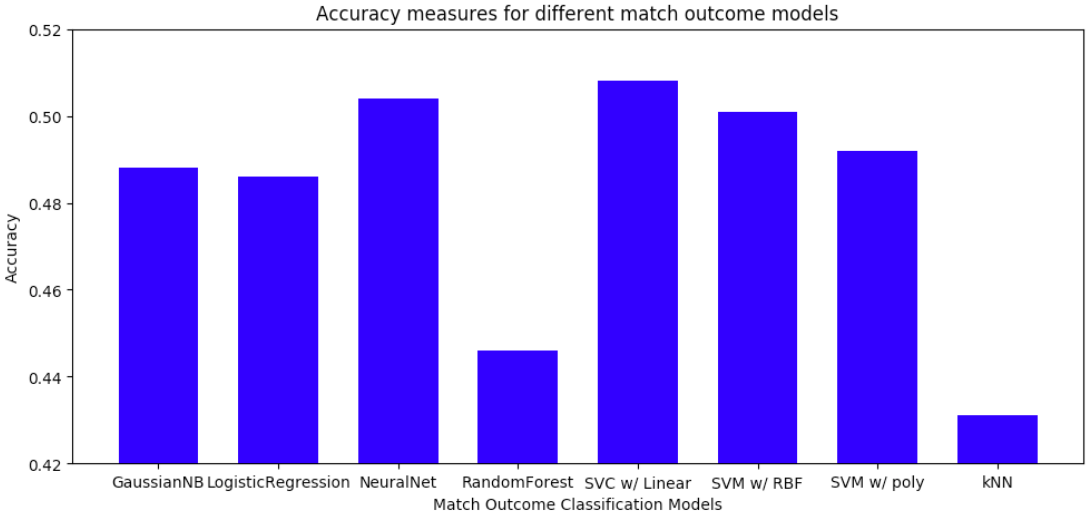

Figure 6.4: Accuracy for different match outcome classification models

- **Match score regression model**
  We have seen that we need a regression model capable of predicting expected number of goals for each team given offensive and defensive ELO ratings as well as home advantage. We will therefore test and compare six different regression models:

  - Linear Regression
  - SVM Regressor with RBF kernel
  - SVM Regressor with Linear kernel

- SVM Regressor with Polynomial kernel
- Random Forest Regressor
- Neural Network (Multilayer Perceptron with two hidden layers of 100 neurons: this has been optimised as the best Neural Network architecture for this task)

The results we obtained using the same parameters, keeping all other elements of the pipeline constant and training a new match score regression model, are presented in Fig.6.5. We can see that we obtain the lowest Mean Absolute Error by using the Neural Network model, which we will choose in order to build our match score regression model.

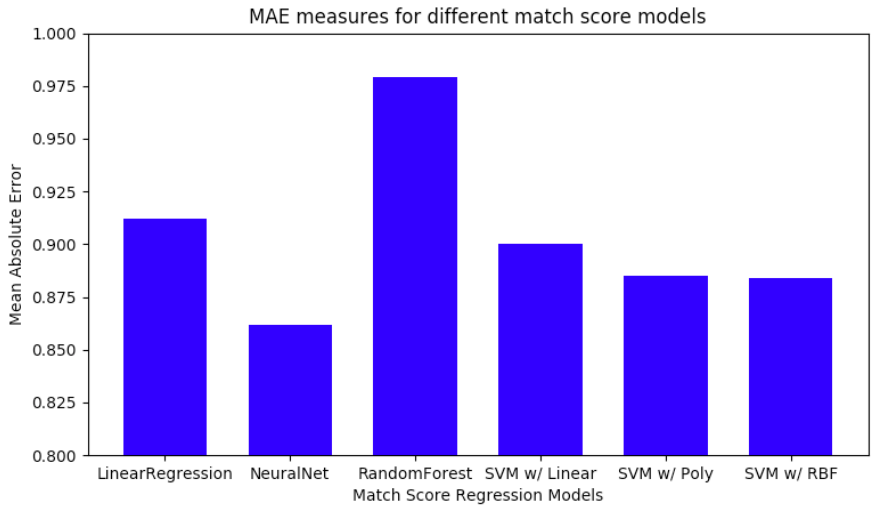

Figure 6.5: MAE for different match score regression models

6.3 Parameter optimisation

In this section, we will explain which parameters we wanted to optimise and the method that was used to do so.

6.3.1 OpenMOLE parameter optimisation

OpenMOLE [47] is a software which enables parameter optimisation through the use of Genetic Algorithms.

The software takes as input a model, a number of parameters to test as well as their ranges, and two metrics to minimize. It then runs multiple batches of parallel tests with random parameters for each run. The use of Genetic Algorithms enables the reproduction of the tests that generate the best results in order to converge towards

a minimum, while discarding parameter values that generate poor results.

The final output of this calibration method is a set of parameters called the Pareto frontier solutions for which one of the two metrics to minimise cannot be minimised without increasing the other metric. This principle is illustrated in Fig.6.6.

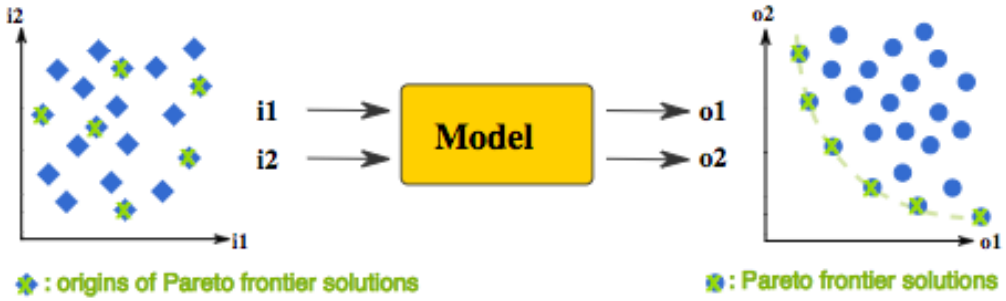

Figure 6.6: Diagram of the openMOLE parameter optimisation method [47]

6.3.2 Results

Two batches of optimisations were run for the same parameters: one to optimise the match score regression model, and the other to optimise the match outcome classification model. The optimisation ran for 6 hours, evaluating a total of 130,000 different parameter combinations.

The parameters to optimise and their ranges given to the software were:

- Weight on shot xG if the team has a player advantage: [0,1]
- Weight on shot xG if the team has a player disadvantage: [1,2]
- Weight k on shot xG at end of game if winning by 1 goal: [0.5,1]
- Weight k on shot xG at end of game if winning by 2 goals or more: [0.5,1]
- ELO ratings adjustment towards mean between seasons: [0,0.5]
- 'Learning rate' η for ELO ratings after each game: [0,0.25]
- Weight of shot-based xG compared to non-shot-based xG: [0,1]
- Weight of home/away-specific ELO ratings compared to general ratings in ELO recalculation: [0,1]

The metrics to optimise given to the software were:

- for the regression model: minimise RMSE, maximise Accuracy
- for the classifiction model: maximise F1 Score and Accuracy

The results of the parameter optimisation for the match score regression model are as follows:

- Weight on shot xG if the team has a player advantage: 0.95
- Weight on shot xG if the team has a player disadvantage: 1.9
- Weight k on shot xG at end of game if winning by 1 goal: 0.6
- Weight k on shot xG at end of game if winning by 2 goals or more: 0.85
- ELO ratings adjustment towards mean between seasons: 0.01
- 'Learning rate' η for ELO ratings after each game: 0.15
- Weight of shot-based xG compared to non-shot-based xG: 0.5
- Weight of home/away-specific ELO ratings compared to general ratings in ELO recalculation: 0

The results of the parameter optimisation for the match outcome classification model are as follows:

- Weight on shot xG if the team has a player advantage: 0.9
- Weight on shot xG if the team has a player disadvantage: 1.25
- Weight k on shot xG at end of game if winning by 1 goal: 0.52
- Weight k on shot xG at end of game if winning by 2 goals or more: 0.9
- ELO ratings adjustment towards mean between seasons: 0.135
- 'Learning rate' η for ELO ratings after each game: 0.12
- Weight of shot-based xG compared to non-shot-based xG: 0.6
- Weight of home/away-specific ELO ratings compared to general ratings in ELO recalculation: 0.2

6.3.3 Analysis of optimal parameters

We have very interestingly seen that the optimal parameters for the match score regression model and the match outcome classification are different. Although they approximately agree on most parameters, there are a few for which the difference is significant.

- Weight on shot xG if the team has a player advantage: both models agree that the weight should be close to 1 (0.9 and 0.95). We can interpret this parameter value meaning that teams with a player disadvantage will not attack as much as they normally would, and that they will concentrate on defending, thus making still difficult to score against them even if they are at a player disadvantage.

- Weight on shot xG if the team has a player disadvantage: both models are quite far apart for this parameter (1.25 and 1.95). They do agree on the fact that shots' xG value should be increased when playing with a player disadvantage as it is in general very difficult to create chances in this case.

- Weight k on shot xG at end of game if winning by 1 goal: both models agree that a shot's value should be decreased by nearly half (0.52 and 0.6) by the end of the game when leading by 1 goal (optimised for the shot xG value to start decreasing at the 70th minute). This shows us that a team chasing a one goal lead will take risks to equalise near the end of the game and therefore make it easier for the opposing team to attack and to create chances.

- Weight k on shot xG at end of game if winning by 2 goals or more: both models agree that a shot's value should be decreased by only a little (0.85 and 0.9) by the end of the game when leading by 2 goals or more (optimised for the shot xG value to start decreasing at the 70th minute). This shows us that a team chasing a two goal lead will not take that many risks to equalise near the end of the game as the outcome is difficult to change, and will only make it a little easier for the opposing team to attack and to create chances compared to if the teams had the same number of goals.

- ELO ratings adjustment towards mean between seasons: both optimised parameter are very far off, with the optimal parameter being of 0.01 for the regression model and 0.135 for the classification model. This suggests that the number of goals scored or conceded for a team at the beginning of a season is quite similar than for the end of the previous season. On the other hand, the probability of winning/drawing/losing is not quite the same as the end of the previous season, so it should be adjusted towards the league's mean.

- 'Learning rate' η for ELO ratings after each game: Both models agree on the fact that the ELO ratings learning rate should be approximately the same (0.12 and 0.15). A higher learning rate for the regression model suggests that scoring and conceding goals depends more on the form of recent games than winning and losing games, which is more stable over time.

- Weight of shot-based xG compared to non-shot-based xG: Both models agree that half or more (0.5 and 0.6) of a team's match xG metric should come from their shots compared to other in-game statistics. This does mean that non-shot-based xG is quite important in the model and does reflect a better chance of winning a game/scoring goals.

- Weight of home/away-specific ELO ratings compared to general ratings in ELO recalculation: Both models agree that the weight of home/away-specific ELO ratings does not should not affect a team's predicted amount of xG too much compared to general attacking and defensive ratings. For the model score regression model, home/away-specific ratings should not even be taken into

account, meaning that only looking at the general offensive and defensive ratings should give a good idea of the xG value a team is expected to generate in a match.

Chapter 7

Evaluation

In this chapter, we will evaluate our models in absolute terms as well as comparing their predictive performance against benchmark methods that have been used in past research on the subject. We will also be able to check our predictions against actual betting odds to see if we come close to professional bookmakers' complex models.

As we are looking to obtain the best model we can, there is no specific accuracy that we need to attain to be successful in this project. The comparison with other benchmarks will be the best way of knowing if using a combination of expected goals model and team ratings is better than simpler models that have been used in the past.

7.1 Absolute results

- **Match outcome classification model results**:
  For our final classification model, we obtain a predictive Accuracy of 0.511 and a F1 Score of 0.382.
  This means that our model is able to predict the correct outcome of a match in more than 50% of games.
  Also, when using actual goals instead of our expected goals metrics in the same model, we only obtain an accuracy of 0.496 and a F1 score of 0.361. We can see that using expected goals does not give us a large predictive advantage over using actual goals. However, due to the limitations in the data to estimate shot xG probability, we can say that this difference could only increase as better quality data is used to generate xG metrics.

- **Match score regression model results**:
  For our final regression model, we obtain a RMSE value of 1.153, a MAE value of 0.861 and, using the Poisson distribution to generate match outcome probabilities, an Accuracy of 0.446.
  This means that, on average, our model predicts a number of goals for a team that is 0.861 goals away from reality. This could seem to be quite large, how-

ever it is worth remembering that there exists a high number of outliers, such as very high number of goals, something that is nearly never predicted by the model, which tends to be more conservative.
We can also see that our Accuracy performance of 0.446 is quite poor compared to our classification model and other benchmark models (Fig.7.1). This could lead to the hypothesis that the Poisson distribution does not accurately model the distribution of goals in a game of football.
Finally, when using actual goals instead of our expected goal metrics in the same model, we obtain a worse RMSE value of 1.202 as well as a worse Accuracy of 0.427. We can again see that using expected goals rather than actual goals in our models has generated a better predictive performance, even though our 'actual goals' model performs quite well compared to other benchmark models (Fig.7.1).

7.2 Comparison with benchmarks

In this section, we will compare different performance metrics of our final models with benchmark models.

In Fig.7.1, we can see a comparison of the classification accuracy for our final models as well as benchmark models.

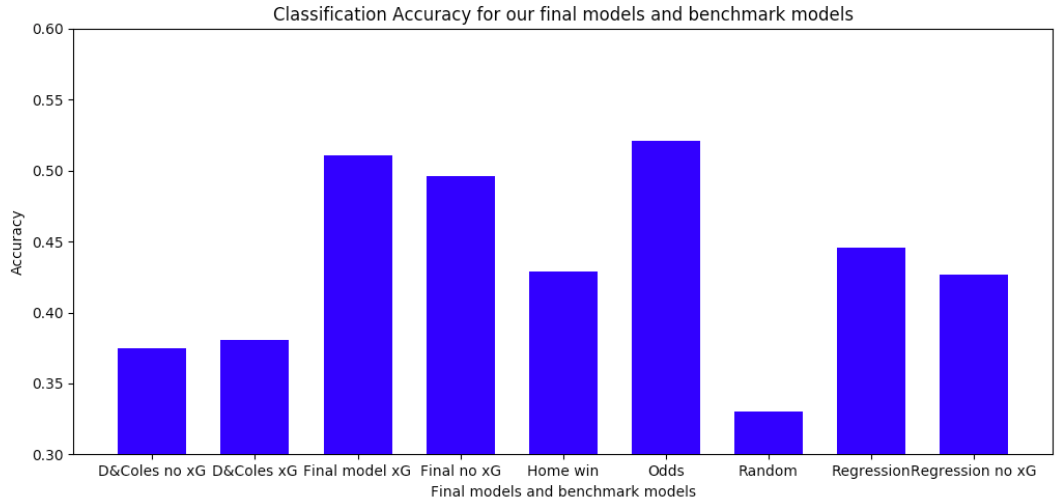

Figure 7.1: Accuracy for our final models and benchmark models

In Fig.7.2, we can see a comparison of the classification F1 score for our final models as well as benchmark models.

Finally, in Fig.7.3, we can see a comparison of the root mean squared error for our final models as well as benchmark models.

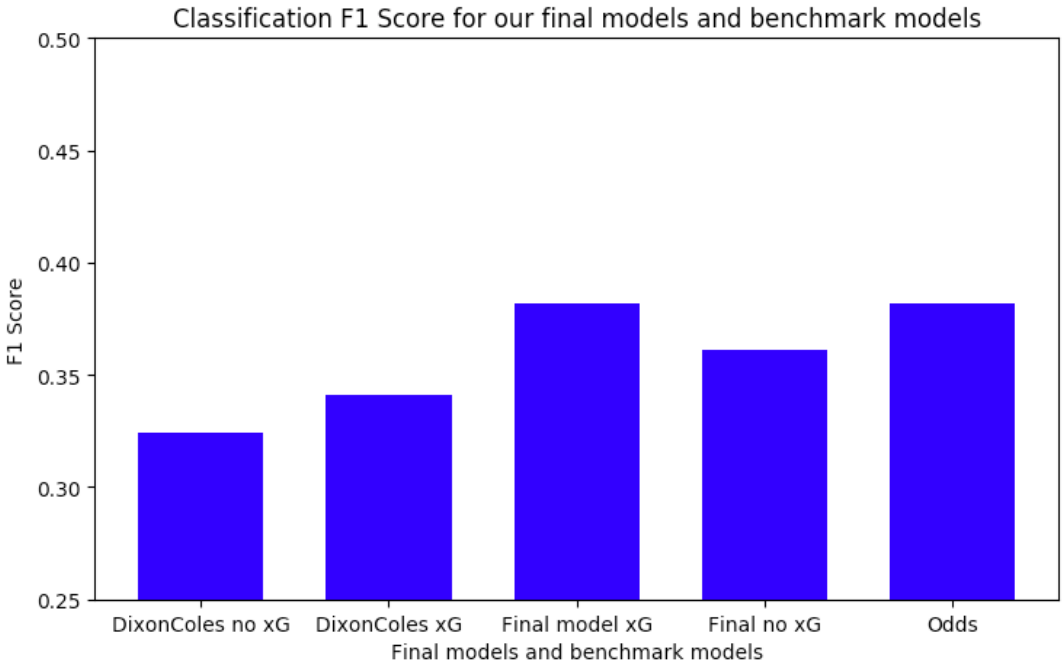

Figure 7.2: F1 Score for our final models and benchmark models

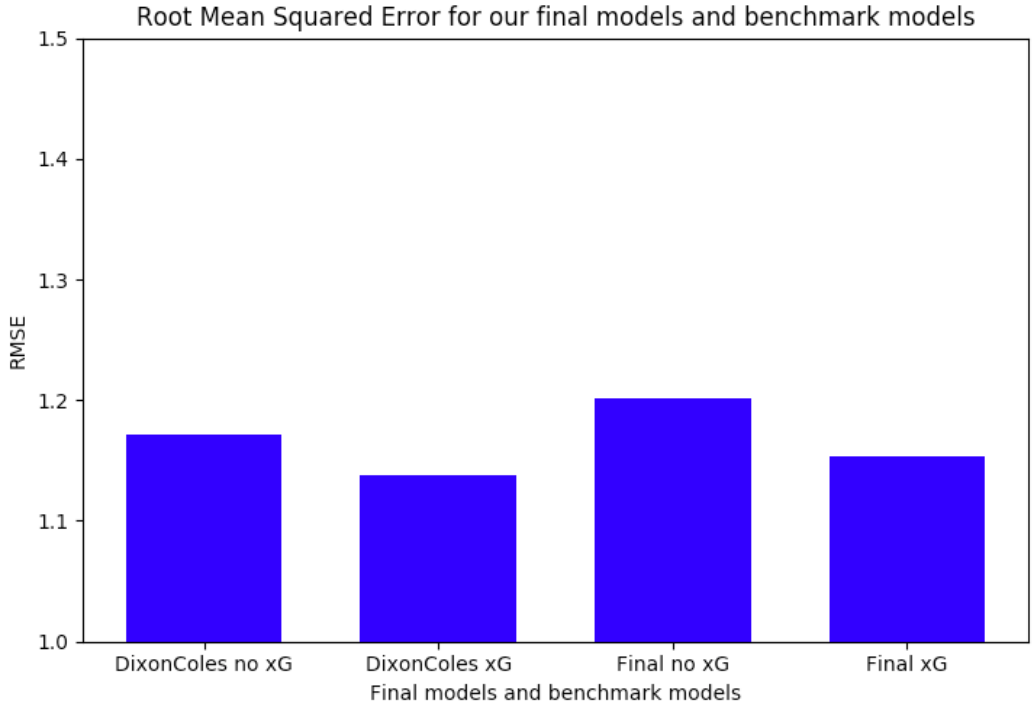

Figure 7.3: RMSE for our final models and benchmark models

7.2.1 Betting odds

Our dataset contained the betting odds from Bet365, one of the United Kingdom's most popular bookmakers, for each game that we attempted to predict.

Choosing the smallest odds value as the most probable outcome of a game for the bookmakers' model, we obtain an Accuracy of 0.521, which means that bookmakers manage to predict the correct outcome for 52.1% of matches in our dataset.

When comparing the performance metrics of the bookmaker's model to the classification model we created in this project, we can see that our final classification model has marginally worse classification accuracy (0.511 vs 0.521). However, we obtain the same F1 score as the bookmaker's model (0.382). This shows us that our final classification model performs quite well in general, as bookmakers use complex models trained on many seasons and have access to more data than we had in our database.

7.2.2 Dixon & Coles model

We implemented the Dixon and Coles model in Python in order to generate performance metrics and compare them to our own final models' performance.

Firstly, we decided to try implementing the classical Dixon and Coles model using actual match results, as well as implementing the same model but this time using expected goals instead of actual goals. As we can see in our comparison graphs (Figs.7.1, 7.2 and 7.3), the expected goals Dixon and Coles model consistently outperforms the actual goals model accross our performance metrics (Accuracy: 0.381 vs 0.375, F1 Score: 0.341 vs 0.324, RMSE: 1.138 vs 1.172).

When comparing our final models to our Dixon and Coles implementation, we find that our match outcome classifier outperforms the Dixon and Coles model in terms of Accuracy and F1 score by a consequent margin. Indeed, our Accuracy of 0.511 is much larger than the Dixon and Coles accuracy of 0.375, and our F1 score of 0.382 is again larger than the Dixon and Coles model F1 score of 0.324.

However, the Dixon and Coles model's Root Mean Squared Error in predicting model scores comes very close to our final regression model's RMSE score: 1.172 for the Dixon and Coles model against 1.153 for our final regression model. It is interesting to note that the best-performing model in terms of RMSE is the Dixon and Coles model using expected goals data, with a RMSE of 1.138.

7.2.3 Other naive benchmarks

One naive benchmark we can use is for our classifier to always predict a Home win as that is the most probable outcome on average.

This gives us an Accuracy of 0.429, which we can use as a lower bound for the expected accuracy of an intelligent match outcome classification model. We can see that our regression model, when used for classification by using the Poisson distribution to estimate outcome probabilities, achieves a marginally better accuracy of 0.446, which is quite disappointing.

Finally, another naive benchmark we can use is to randomly choose one of the three outcomes, which gives us a classification accuracy of 0.33. A model with lower accuracy can be said to perform worse than a random model. We can observe than the classification accuracy of the Dixon and Coles model is only slightly higher than that at 0.375. This is quite a low accuracy for the Dixon and Coles model and could be due to the small amount of data for the Dixon and Coles model to calculate each team's ratings using Maximum Likelihood.

7.3 Strengths & Weaknesses

The strengths of this project are:

- We have built an easy-to-use pipeline that can easily be reproduced and reused in other Machine Learning projects.
- Using ELO ratings allows for the model to be improved after each game by calculating expected goals and recalculating the ELO ratings, without the need to train the whole model again.
- We achieved a classification accuracy comparable to the bookmakers' complex models and higher than traditional models such as the Dixon & Coles model which suggests that expected goals as a metric is today an essential part of football modelling techniques in order to achieve higher predictive performance.
- The parameter optimisation step enabled us to understand the relative importance of some elements in the model when predicting the outcome or result of future matches.
- We have explored a wide range of different Machine Learning models and parameters to optimise our model's performance as much as we could.

However, the project also possesses some weaknesses:

- We hit an upper bound in classification accuracy when optimising the parameters and our choice of Machine Learning models. This could be due to the high variance of football matches results, which could explain why we could not achieve better performance, or to our model's design which could have disallowed us from achieving better classification accuracy.
- We only train our models using data from 2 seasons and 5 leagues: having more data available would help make the model more robust and better generalize training data.

- We did not have another dataset to train our model on, which means that the obtained results should be verified using another dataset.
- To use the model and generate a prediction, specific data is needed (e.g. coordinates of each shot in a game) which makes the model difficult to use without the right data.

Chapter 8

Conclusion & Future Work

8.1 Summary

Our main objective of building an expected goals model by exploring different Machine Learning techniques has been accomplished. Indeed, we used modern Machine Learning algorithms such as Neural Networks, Random Forest and Support Vector Machines techniques to generate match outcome and match score predictions.

We managed to find and improve a database containing enough information to generate expected goals metrics, through both shots and other in-game statistics, and ELO team ratings.

A model training and testing pipeline was built to quickly and easily tweak our model and try different hypotheses, using Luigi to link our different model components together.

We have also compared our predictions to benchmark methods in order to better understand our models' predictive performance. We have crucially found that our expected goals models achieve a similar performance to bookmakers' odds, and that using expected goals instead of actual goals in traditional models such as the Dixon & Coles model, helps achieve better predictive performance.

8.2 Challenges & Solutions

8.2.1 Finding data

One of the main challenges encountered during the project was to find suitable data to use to build an expected goals model. A lot of time was spent doing research to find public databases which enabled me to find the Kaggle database that we have used for this project.

However, I believed that I could find more interesting data to build better models so I set out to scrape data from the WhoScored.com website [48] which contains a very large amount of data for each game. After familiarising myself with some popular Web Scraping tools, I realised that the website had an anti-scraping protection that was near impossible to break, as they wanted to avoid their data being put into public databases. I spent a lot of time trying to find ways around this protection, however after some time we decided with my supervisor that it would be wiser to go to our fall-back position of using the Kaggle database and start to build our model.

8.2.2 Model & parameter choices

Difficult model choices had to be made to start of the project in terms of model design, which then led to an incremental approach using simple shot expected goals and ELO calculations, then adding elements that made the model more accurate.

The large number of parameters to optimise made it near impossible to find a set of best parameters by testing different values for each parameter step-by-step. I am very thankful that my supervisor, Dr. Jonathan Passerat-Palmbach, talked to me about the OpenMOLE parameter optimisation method using Genetic Algorithms, and helped me optimise my model parameters, which allowed the model to achieve better predictive performance.

8.3 Future extensions

There are many directions in which this project could be taken with more time and resources.

8.3.1 Improved data

Firstly, I believe that the model can definitely be improved with more interesting data regarding match events. For instance, one weakness of the shot expected goals estimation is that we do not know the position of the opposing team's players at the time of the shot. It is clear that, for instance, having a player between the ball and the goal would dramatically reduce the probability of the shot resulting in a goal.

Also, we had no information regarding the geographical location and type of passes that were made in the game. As many dangerous scoring positions do not result in a shot, having passes data would allow the model to better estimate the number of expected goals a team should have scored in a game.

Getting data from the WhoScored.com website would allow the collection of a very large amount of interesting data for each match. WhoScored.com is a popular football statistics website that presents very interesting data on a huge range of games from all the top championships worldwide since 2009. Its data is provided by Opta

[49], the largest sports data provider in the world today. WhoScored.com is especially interesting to use as it displays valuable match data such as:

- Possession, shots, dribbles, crosses, tackles, etc.
- Player formation and ratings
- Match events with their time of occurrence and position on the football pitch
- Shots data (zone from which the shot is taken, part of the body, from which game situation, ...) which is extremely valuable to build an expected goals model

8.3.2 Monte Carlo simulations to predict future events

Monte Carlo simulations are run a large number of times and rely on random sampling to generate predictions.

Now that we have created our expected goals model, we could run Monte Carlo simulations thousands of times on one season of a league to see which team has the highest probability of winning the championship, for example.

The advantage of Monte Carlo techniques is that they can be used to generate predictions into the future for competitions that have not started for example.

8.3.3 Player-centred models

With more time, we could add a player-centric element to our expected goals models by using player data to better understand how many goals a team was expected to score in a game.

For example, we could use a player's shooting accuracy or the knowledge about his strongest foot to better assign probabilities to each shot or match event that this player takes part in.

Instead of looking at team performance with expected goals, we could also look at individual player performance to better predict team performance using all the players taking part in a match, or generating team selections that have the highest probability of winning a match.

8.3.4 Team profiles

All the team data such as the teams' possession, number of crosses, number of tackles, number of headers, etc., could be used to classify each team into different categories of playing styles.

Models could then be built to understand the interaction between the playing styles of two different teams and help predict the outcome of a match where these two teams face each other.

This would add to the ELO offensive and defensive team ratings and could help us understand some of the outliers that cannot be predicted simply by using the ELO ratings.

8.3.5 Betting analysis

Another potential future extension to this project could be to look at betting odds to see whether our model could recommend good value bets and profitable betting strategies that generate a profit on the long run.

Other than looking at betting odds for traditional bookmakers, the predictions could be used to try and generate profitable strategies on Augur [50], a Blockchain-powered forecasting tool where people bet against each other rather than against a bookmaker. This could lead to better betting opportunities compared to bookmakers, who use a risk model in order to minimise potential losses.

Bibliography

[1] *Size of the betting industry* [http://www.bbc.co.uk/sport/football/24354124[1]]. pages 1

[2] *Average goals per match in the Premier League* [http://www.skysports.com/football/news/11095/10752408/the-evolution-of-full-backs-how-they-became-integral-to-success[2]]. pages 2

[3] *Linear Model diagram from scikit-learn website* [http://scikit-learn.org/stable/modules/linear_model.html[3]]. pages 8

[4] *Logistic Function plot* [http://www.thefactmachine.com/logistic-regression/[4]]. pages 10

[5] *Architecture of the Random Forest model* [https://www.researchgate.net/figure/Architecture-of-the-random-forest-model_fig1_301638643[5]]. pages 11

[6] *Diagram of the Gaussian Naive Bayes model* [https://www.researchgate.net/figure/Illustration-of-how-a-Gaussian-Naive-Bayes-GNB-classifier-works-For-each-fig1_255695722[6]]. pages 12

[7] *Diagram of the k-Nearest-Neighbors model* [http://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html[7]]. pages 13

[8] *Plot of the SVM Hyperplane* [http://scikit-learn.org/stable/modules/svm.html[8]]. pages 14

[9] *Diagram of the neural network model* [http://scikit-learn.org/stable/modules/neural_networks_supervised.html[9]]. pages 15

[10] M. J. Moroney. *Facts from figures, 3rd edn.*. Penguin: London, 1956. pages 16

[11] C. Reep. *Skill and chance in ball games*. Journal of the Royal Statistical Society Series A 131: 581-585, 1971. pages 16

[12] I.D. Hill. *Association football and statistical inference*. Applied Statistics 23: 203-208, 1974. pages 16

[13] M. J. Maher. *Modelling association football scores*. Statistica Neerlandica, 1982. pages 16

[14] M.J. Dixon, S.C. Coles. *Modelling association football scores and inefficiencies in the football betting market*. Applied Statistics, 1997. pages 17

[15] *Plot of the Poisson distribution* [https://www.umass.edu/wsp/resources/poisson/[10]]. pages 17

[16] H. Rue, O. Salvesen. *Prediction and retrospective analysis of soccer matches in a league*. Statistician, 2000. pages 18

[17] M. Crowder, M. Dixon, A. Ledford, M. Robinson. *Dynamic modelling and prediction of English Football League matches for betting*. Statistician, 2002. pages 18

[18] D. Forrest, R. Simmons. *Forecasting sport: The behaviour and performance of football tipsters*. International Journal of Forecasting, 2000. pages 18

[19] T. Kuypers. *Information and efficiency: An empirical study of a fixed odds betting market*. Applied Economics, 2000. pages 18

[20] J. Goddard. *Regression models for forecasting goals and match results in association football*. International Journal of Forecasting, 2005. pages 18

[21] B. Hamadani. *Predicting the outcome of NFL games using machine learning*. Stanford University, 2006. pages 18

[22] A. Adam. *Generalised linear model for football matches prediction*. KULeuven, 2016. pages 19

[23] M. Tavakol, H. Zafartavanaelmi and U. Brefeld. *Feature Extraction and Aggregation for Predicting the Euro 2016*. Leuphana University of Luneburg, 2016. pages 19

[24] S. Kampakis, W. Thomas. *Using Machine Learning to Predict the Outcome of English County twenty over Cricket Matches*. University College London. pages 19

[25] N. Tax, Y. Joustra. *Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach*. Transactions on Knowledge and Data Engineering, 2015. pages 19

[26] A. Joseph, N.E. Fenton, M. Neil. *Predicting football results using Bayesian nets and other machine learning techniques*. Knowledge-Based Systems, 2006. pages 19

[27] M.C. Purucker. *Neural network quarterbacking*. IEEE Potentials, 1996. pages 19

[28] J. Kahn. *Neural network prediction of NFL football games*. World Wide Web Electronic Publication, 2003. pages 19

[29] Hucaljuk, J.; Rakipovic, A. *Predicting football scores using machine learning techniques*. MIPRO, 2011 Proceedings of the 34th International Convention, 2011. pages 19

[30] A.E. Elo. *The rating of chessplayers, past and present*. Arco Publishing, 1978. pages 20

[31] B.L. Boulier, H.O. Stekler. *Are sports seedings good predictors? An evaluation*. International Journal of Forecasting, 1999. pages 20

[32] S.R. Clarke, D. Dyte. *Using official ratings to simulate major tennis tournaments*. International Transactions in Operational Research, 2000. pages 20

[33] J. Buchdahl. *Fixed odds sports betting: Statistical forecasting and risk management*. High Stakes, 2003. pages 20

[34] Lars Magnus Hvattuma, Halvard Arntzen. *Using ELO ratings for match result prediction in association football*. International Journal of Forecasting, 2010. pages 20

[35] J. Lasek. *Euro 2016 Predictions Using Team Rating Systems*. ECML/PKDD, 2016. pages 21

[36] Sasank Viswanadha, Kaustubh Sivalenkal, Madan Gopal Jhawar, Vikram Pudi. *Dynamic Winner Prediction in Twenty20 Cricket: Based on Relative Team Strengths*. Mahindra Ecole Centrale, Hyderabad, India, 2017. pages 21

[37] Anthony Costa Constantinou, Norman Elliott Fenton. *Determining the level of ability of football teams by dynamic ratings based on the relative discrepancies in scores between adversaries*. Journal of Quantitative Analysis in Sports, 2013. pages 21

[38] Brian Macdonald. *An Expected Goals Model for Evaluating NHL Teams and Players*. MIT Sloan Sports Analytics Conference, 2012. pages 21

[39] Harm Eggels, Ruud van Elk, Mykola Pechenizkiy. *Explaining soccer match outcomes with goal scoring opportunities predictive analytics*. Eindhoven University of Technology, 2016. pages 22

[40] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, Iain Matthews. *Quality vs Quantity: Improved Shot Prediction in Soccer using Strategic Features from Spatiotemporal Data*. MIT Sloan Sports Analytics Conference, 2015. pages 21

[41] *FiveThirtyEight football predictions* [`https://projects.fivethirtyeight.com/soccer-predictions/`[11]]. pages 22

[42] *Kaggle European Soccer Database* [`https://www.kaggle.com/hugomathien/soccer`[12]]. pages 23

[43] *Pandas Python library* [https://pandas.pydata.org/[13]]. pages 36

[44] *Scikit-Learn Python library* [http://scikit-learn.org/stable/index.html[14]]. pages 36

[45] *Luigi: Python module to build batch jobs pipeline* [https://github.com/spotify/luigi[15]]. pages 36

[46] *Diagram of the cross-validation model training technique* [https://en.wikipedia.org/wiki/Cross-validation_(statistics)[16]]. pages 40

[47] *openMOLE parameter optimisation method diagram* [https://next.openmole.org/Calibration.html[17]]. pages 45, 46

[48] *WhoScored.com football statistics website* [https://www.whoscored.com/[18]]. pages 57

[49] *Opta Sport data provider* [http://www.optasports.com/[19]]. pages 58

[50] *Augur: Decentralized prediction markets* [http://www.augur.net/[20]]. pages 59

Appendix: Dataset Examples

	id	ıtr	league_id	season	aç	date	ı_	home_team_api_id	away_team_api_id	home_team_goal	away_team_goal
	Filter		Filter	Filter		Filter		Filter	Filter	Filter	Filter
1	1	1	1	2008/2009	1	2008-08-17...		9987	9993	1	1
2	2	1	1	2008/2009	1	2008-08-16...		10000	9994	0	0
3	3	1	1	2008/2009	1	2008-08-16...		9984	8635	0	3
4	4	1	1	2008/2009	1	2008-08-17...		9991	9998	5	0
5	5	1	1	2008/2009	1	2008-08-16...		7947	9985	1	3
6	6	1	1	2008/2009	1	2008-09-24...		8203	8342	1	1
7	7	1	1	2008/2009	1	2008-08-16...		9999	8571	2	2
8	8	1	1	2008/2009	1	2008-08-16...		4049	9996	1	2
9	9	1	1	2008/2009	1	2008-08-16...		10001	9986	1	0
10	10	1	1	2008/2009		2008-11-01...		8342	8571	4	1
11	11	1	1	2008/2009		2008-10-31...		9985	9986	1	2
12	12	1	1	2008/2009		2008-11-02...		10000	9991	0	2
13	13	1	1	2008/2009		2008-11-01...		9994	9998	0	0
14	14	1	1	2008/2009		2008-11-01...		7947	10001	2	2
15	15	1	1	2008/2009		2008-11-01...		8203	9999	1	2
16	16	1	1	2008/2009		2008-11-01...		9996	9984	0	1
17	17	1	1	2008/2009		2008-11-01...		4049	9987	1	3
18	18	1	1	2008/2009		2008-11-02...		9993	8635	1	3
19	19	1	1	2008/2009		2008-11-08...		8635	9994	2	3
20	20	1	1	2008/2009		2008-11-08...		9998	9996	0	0
21	21	1	1	2008/2009		2008-11-09...		9986	8342	2	2
22	22	1	1	2008/2009		2008-11-07...		9984	10000	2	0
23	23	1	1	2008/2009		2008-11-08...		9991	7947	1	1
24	24	1	1	2008/2009		2008-11-08...		9999	4049	1	2
25	25	1	1	2008/2009		2008-11-08...		8571	8203	0	0
26	26	1	1	2008/2009		2008-11-08...		10001	9987	1	0

Figure 1: Screen capture of the Matches table in our database

	id	type	subtype1	ty	ye	ye	team	lon	lat	elapsed	ed	game_id
	Filter	Filter	Filter				Filter		Fil...	Filter		Filter
1	5623584	shoton	blocked_shot				8678	28	63	90	3	4702
2	5623515	shoton	shot				10260	26	10	62		4702
3	5623505	shoton	distance				10260	29	18	60		4702
4	5623500	shoton	blocked_shot				10260	28	9	59		4702
5	5623476	shoton	blocked_shot				8678	17	60	48		4702
6	5623471	shoton	blocked_shot				8678	26	60	47		4702
7	5623469	shoton	blocked_shot				10260	24	13	47		4702
8	5623445	shoton	blocked_shot				8678	25	60	38		4702
9	5623418	shoton	blocked_shot				8678	22	56	22		4702
10	5623405	shoton	blocked_shot				10260	14	10	13		4702
11	5617051	shoton	shot				208931	29	64	90	2	13206
12	5617031	shoton	blocked_shot				8535	34	54	89		13207
13	5617013	shoton	shot				8543	33	11	85		13207
14	5617010	shoton	big chance shot				8543	16	4	85		13207
15	5617004	shoton	shot				208931	16	67	85		13206
16	5616962	shoton	shot				8543	16	9	76		13207
17	5616956	shoton	blocked_shot				9876	24	52	75		13209
18	5616948	shoton	blocked_shot				9876	11	60	74		13209
19	5616900	shoton	distance				8543	30	18	65		13207
20	5616899	shoton	header				8600	22	4	65		13206
21	5616887	shoton	shot				8543	14	8	63		13207
22	5616886	shoton	blocked_shot				8543	20	12	62		13207
23	5616765	shoton	lob				8535	20	63	44		13207
24	5616756	shoton	blocked_shot				8600	28	15	43		13206
25	5616741	shoton	shot				8600	35	3	40		13206
26	5616711	shoton	shot				8535	26	60	34		13207

Figure 2: Screen capture of the Shots table in our database