6CCS3PRJ Final Year Individual Project Report Title

Final Project Report

Author: Tihomir Stefanov

Supervisor: Jeroen Keppens

Student ID: 20051726

April 6, 2023

Abstract

Football(Soccer) predictions have been around for a long time. However, since the digitisation of the industry began in the early 2000s, the data being collected on a football pitch has grown exponentially with more and more sophisticated models trying to predict the best outcomes based on the many metrics that are being taken every game. In recent years many of the well-run clubs across the world started hiring more and more Data Scientists to help them plan ahead and be able to predict the best possible outcome of a game in their favour.
The aim of this project is to create such a football prediction model one using Regression and one using Classification methods for the Premier League season but using already established machine learning techniques. It will incorporate both in-game data and team ratings which are being generated using past results from those teams.

The two models will be chosen based on careful evaluation and comparison of each of the tested models using appropriate performance metrics like accuracy and root mean squared to mention a few. Then all the models will also be compared with benchmark methods that have been used in the past for football predictions.

In this project, we create team form metrics which are based on the teams' attacking and defensive strengths as well as the overall home advantage. These ratings are dynamic and will update after each round of fixtures in order to more accurately represent the performance of each team. We combine those metrics with pre-existing in-game statistics and evaluate the different machine-learning models on that dataset.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Tihomir Stefanov

April 6, 2023

Acknowledgements

I would like to thank my supervisor, Prof. Dr. Jeroen Keppens. The supervision and support that he provided helped guide the process of creating this piece of work. The assistance is much appreciated. Thanks to the Department of Informatics staff and IT services at King's College London.
Also thanks to my family for the encouragement and moral support throughout the project.

Contents

Chapter 1

Introduction

Football is the most watched sport in the world. Many people watch the games for fun or are not as engulfed by them, but there are also hardcore fans that go to each game, watch each game and break down what has happened after each game. The rivalry between clubs has been a long-standing tradition and many people want to predict the results with their friends and family before anyone has kicked a ball. In the past, those discussions were only local due to the limited capabilities of any social media and based on many assumptions and biased visual observations.

However, the two major factors that changed the perspective of football predictions taking it to a whole new level are:

- The introduction of Football Analytics in the early 2000s
- The introduction of social media and stream-based news-feed

The use of data science and machine learning in football betting has significantly grown over the past 20 years. With the availability of vast amounts of data, from match results to player statistics, teams, and individual performances, betting companies have leveraged machine learning algorithms to analyze and make sense of this data to make better predictions and ultimately improve their profits.

Here are some of the ways data science and machine learning have impacted the football betting business:

- Better predictions: By analyzing historical data, betting companies can use machine

learning algorithms to identify patterns, trends, and correlations between different variables. This enables them to make more accurate predictions about match outcomes, player performance, and other related factors that can impact the game.

- Personalized betting experiences: With data science, betting companies can use customer data to personalize the betting experience for each user. For example, they can analyze a user's betting history, preferences, and behaviour to recommend relevant matches, bets, and odds that are most likely to appeal to them.

- Risk management: Data science has also enabled betting companies to manage their risks better. By analyzing past betting patterns and identifying trends, companies can adjust their odds and betting limits to minimize their exposure to risk and maximize their profits.

- In-game analysis: With real-time data analytics, betting companies can also analyze in-game events to make informed decisions about live betting odds. This includes analyzing player statistics, injuries, substitutions, and other relevant factors that can impact the outcome of the game.

Furthermore, prediction systems are also used by the teams themselves. Liverpool for example has been using data analytics, AI and neuroscience to predict which players they should sign that could suit their system and to also analyse the opposition's tactics to exploit vulnerabilities.[26] Arsenal invested north of 2$ million dollars buying the company statDNA[28] in 2014 in order to integrate statistical analysis into the scouting of players based on numerous data points. This has directly helped them gradually develop a football squad that is currently top of the league and one of the favourites to win the Premier League.

1.1 Motivations for the project

My main motivation for this project stems from the sport itself. It's been really intriguing how data science contributes to the sport allowing smaller teams to compete with more financially capable teams because of statistical analysis and accurate prediction systems.

Another motivation of mine was my inexperience in the field of Machine Learning and after taking the Introduction to Artificial Intelligence module[39] during my second year at university, I decided that I want to learn more about the topic and what better way than to apply the learned knowledge into something that you are passionate about.

Since I am still a novice in the field of Machine Learning I wanted to start with a familiar topic so that the learning curve is not unnecessarily steep. However, the idea behind the project is to continue developing the model and expanding into other areas which have prediction-based algorithms in use such as in the fin-tech industry and stock price predictions.

1.2 Scope of the project

The scope of the project will be to create a machine-learning pipeline that will accurately evaluate multiple machine-learning models and compare their performance. I decided to focus on the Premier League since it has a vast amount of data that has been collected over the past 15 to 20 years and focusing on a single league would allow me to have more consistent results when implementing different machine learning models. The data will be collected from public resources on the Internet but also self-generated using past football results to do so.

1.3 Aim of the project

The primary goal of this individual project is to predict the outcome of football games using different Machine Learning techniques based on a multitude of features such as:

- Teams' attacking and defensive strengths
- In-game specific data points such as number of shots on and off target, number of yellow and red cards

In order to achieve the main goal of the project we need to accomplish the following objectives:

- To identify a complete and suitable dataset containing both team and game-specific data points to improve the accuracy of the machine learning model.
- Implement robust and easily maintainable Machine Learning training and testing pipeline
- To implement various ML classification and regression algorithms that are capable of predicting the outcome of football games based on the available dataset.
- To compare the performance of the various ML classification and regression algorithms to find the best two models from the two categories.
- To compare the accuracy of the models evaluated to other benchmark models.

Chapter 2

Background and Past Research

2.1 Machine Learning

Machine Learning is one of the three main pillars in the field of Artificial intelligence. Since the area is too broad to discuss in length in this paper I will focus mainly on how it is related to this project.

Machine Learning splits into two main types:

- Model-Based - where we define an encoding of what the world looks like and we define how we can change the world and then we search for good strategies on how to change the world the way we want to.
- Data-Driven - where we have a model that computes data and gives us the correct answer based on the information it has been provided from historic data.

For the purposes of this project, we will focus on Data-Driven machine learning techniques, especially of the Supervised learning type.

2.1.1 Supervised Learning

In Supervised learning, the program is 'trained' on a given set of examples and it reaches an accurate conclusion when given new data. Regression and Classification are the two main types of supervised Learning techniques.

2.1.2 Regression

The Regression machine learning model uses an already-existent dataset of values. Such a model assumes that there is a relationship between the input variable x and the output variable y so that you can predict y from the x variables. Hence correlations need to be established between those two variables. In the project context, we will look at specific regression models that are still widely used for football predictions.

Poisson Regression

Poisson Regression is part of a bigger group of models called GLM(Generalised Linear Models). They are a flexible class of statistical models that can handle a wide range of response variables and distributional assumptions. GLMs are an extension of linear regression models, which assume a linear relationship between the response variable and the predictor variables.[38]
In contrast, GLMs allow for non-linear relationships between the response variable and the predictors, and they can model a wide range of probability distributions. We will focus on the Poisson distributions specifically.
The general formula for GLMs can be expressed as follows:

$$g(E(Y)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

In the formula:

- Y is the response variable
- $X1, X2, ..., Xk$ are the predictor variables
- $g()$ is the link function
- $E()$ represents the expected value of Y

There are three main components of a GLM:

- Random Component - specifies the probability distribution of the response variable. In our case, the response variable will be a cont, and the random component would specify a Poisson distribution. The random component helps model the relationship between the response variable and the predictor variables.

- Systematic Component - consists of the predictor variables and their coefficients, which together form a linear predictor. The systematic component helps us to model the relationship between the response variable and the predictor variables. The coefficients represent the magnitude and direction of the relationship between each predictor variable and the response variable.
- Link function - used to relate the expected value of the response variable to the linear predictor. Different link functions can be used depending on the distributional assumption of the response variables. For the Poisson regression model, the link function used is the log link function because it maps the mean of the Poisson distribution to the entire real line, which is the range of the linear predictor.

The general form of a Poisson regression model with a log link function can be expressed as follows: $log(\mu_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}$

Where:

- Random component - The distribution of Y is Poisson with the mean λ
- Systematic component - x is the explanatory variable (continuous or discrete) and is linear in the parameters. There can be multiple variables of such kind
- Link function - the log link function

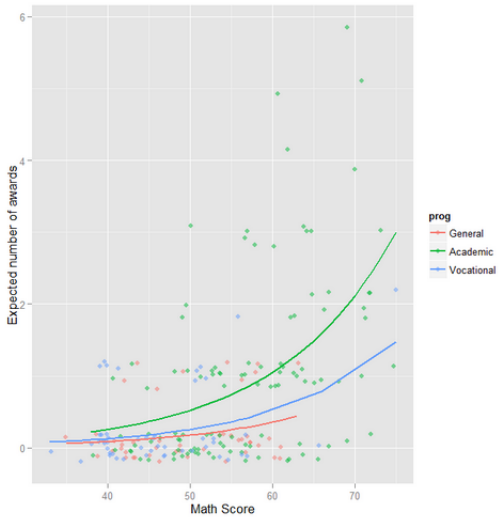

Figure 2.1: Example of Poisson Regression model [4]

Poisson Regression is a very popular technique because it is easy to implement, it handles count data such as non-negative integer values. Poisson Regression does not require any assumptions about the normality or line.

Although Poisson Regression does take into account over-dispersion, it assumes that the variable is equal to its mean. This means when there are cases where the variance of the response variable is larger than its mean, over-dispersion happens. This could cause biased estimates. Another disadvantage is the assumption of independence between the different observations. In some cases, observations may be correlated.

Artificial Neural Networks

Artificial Neural Networks[25] or ANN for short are part of the Deep Learning Machine Learning models in which the algorithms are based on the structure of the human brain. They use layers of neurons (nodes) which act as the processing unit of the network.

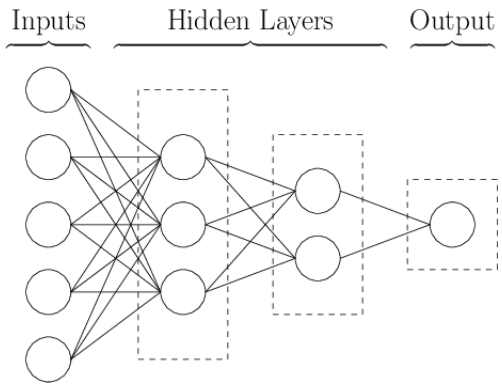

Figure 2.2: Example of a simple Neural Network [19]

As you can see from Figure 2.2 the neurons are split into layers of which there is one input and one output layer. The layers in between are called hidden layers. Inside the hidden layer, we perform most of the computations required by the network.

The initial inputs are fed into the input layer from the dataset and assigned a weight. Then neurons of one layer are connected to the next layer through channels. Each of those channels has assigned a numerical value also known as "weight". Since multiple neurons can be connected to a proceeding neuron in the next layer the sum of all the weights from all connected neurons is sent to the next neuron:

$$\sum_{i=1}^{n} w_i$$

Where n is the number of input neurons and w are the corresponding weights coming from each input neuron.

Then to the sum is added a numerical value called the "bias" which is different for each of

the neurons in the network. Finally, we pass the value through a threshold function also known as the activation function which as it is in its name determines whether this neuron has the necessary value to pass the threshold and be activated. So the final output would be this:

$$o = a(\sum_{i=1}^{n} w_i + b_j)$$

Where:

- b_j This is the bias value for neuron j
- $a()$ This is the activation function applied to the sum of the weights and the bias value
- o This is the output value from the activation function

Depending on whether the output value has surpassed the threshold the neuron gets activated. An activated neuron transmits data to the next layer of neurons. This is called forward propagation. In the output layer, the neuron with the highest value determines the final outcome in terms of probability. Doing this process once won't yield very accurate results, however, this is where backpropagation comes into play. After each iteration of forward propagation, the predicted outcome is compared to the actual outcome from the model and the error in prediction is calculated (also known as the loss function). The magnitude of the error whether positive or negative is then transferred all the way back through the network to the input layer. The way this is accomplished is by adjusting the weights of the channels between the neurons. Here is the full equation:

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

Where:

- $\frac{\partial C}{\partial w_{ij}}$ represents the sensitivity of the cost function to changes in the weight.
- η represents the learning rate determining how quickly weights are updated.
- $w_{ij}(t)$ represents the current weight w between nodes i and j for the past training attempt t
- $w_{ij}(t+1)$ represents the updated weight w between nodes i and j for the next training attempt t

In terms of activation functions, there are many that can be used for different purposes with one of the most commonly used ones being the sigmoidal function:

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

Where x is the input in our case the sum of all the weights from the input neurons. That way we can model non-linear relationships which is often the case in real-life problems to determine complex patterns in data.

Overall, Neural Networks have many advantages such as the ability to adapt to new data situations, they are well-suited to non-linear relationships between inputs and outputs and parallel processing making them very fast, efficient and accurate machine learning models especially when large amounts of data have to be processed.

However, the disadvantage of such a model is the high complexity of the model making it very difficult to interpret and analyse the decision-making of the model. Another disadvantage is the need for large amounts of data in order to optimise the real capabilities of the model.

Support Vector Regression (SVR)

Support Vector Regression[36] is part of a bigger family of Machine Learning models called Support Vector Machines or SVM for short. In SVR we are trying to fit a curve or a hyperplane (if there are multiple dimensions) that shows the best possible relationship between the input features and the target value. The best way to achieve this is to find a hyperplane that passes through as many of the data points as possible whilst minimising the error.

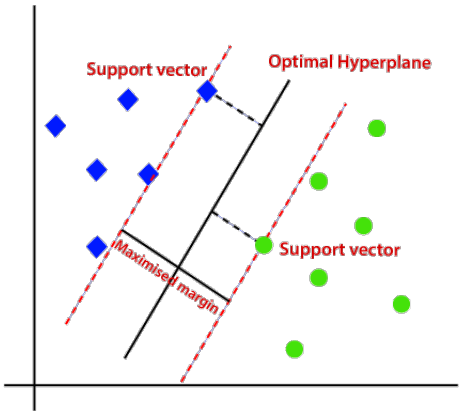

Figure 2.3: Example of a 2d hyperplane for an SVR [6]

From Figure 2.3 we can see that the overall idea of SVRs is to identify the hyperplane that will best separate the different sets of data. The Support Vectors are the extreme points in the dataset and having the distance between the support vectors and the hyperplane as far apart as possible will yield us the best possible accuracy. Mathematically this can be represented as

follows:

$$m = n1 + nk$$

Where:

- n is the distance between the hyperplane and the support vector
- k is the number of support vectors depending on the complexity of the data or the margin width
- m is the distance between the k number of support vectors also known as the distance margin

In order to transform the data points to be used from a low to a high dimensional space, Kernel functions are used. The way this is achieved is by computing the similarity between data points. This can be done by computing the dot product between the points in a new feature space which can have a higher number of dimensions. This allows for a more accurate representation of the data and the subsequent relationships that can be identified between it. There are many different kernel functions one of the most common ones is the Polynomial kernel. It is used for non-linear relationships between data. The formula for the Polynomial kernel function can be expressed as:

$$K(x, y) = (x^T \cdot y + c)^d$$

Where:

- x and y are input vectors
- c a constant that controls the influence of the lower or higher order polynomials.
- d is the degree of the polynomial. The higher the d the more complex nonlinear relationships can be captured by the kernel, but if not careful this could lead to over-fitting.

The main advantages of SVRs are that they can work well with small datasets even where the number of samples is less than the number of features. Similarly to ANN it can handle non-linear data due to the application of the kernel function and applying the hyperplane to high number of dimensions which is the case for most real-life problems.

Disadvantages with SVRs reside around it being computationally expensive due to the fact that they require solving a quadratic problem with a number of constraints. This is especially

evident with a large dataset. Another disadvantage could be the limitation of a single-output regression which means that there has to be only a single output/target parameter otherwise a new SVR needs to be created for each separate target parameter which is costly and time-consuming.

XG Boost Regressor

XG Boost stands for extreme gradient boosted trees. It is under the family of ensemble machine learning methods which combines the predictions of many decision trees and makes a final prediction. It works in a similar fashion to a Random Forest Machine Learning algorithm with the major difference being that it optimises each tree after the other learning from its mistakes instead of each tree running its classification independently and then having a majority voting on the final outcomes from each tree (Random Forest will be discussed in more detail in the Classification Machine Learning model)

Since each tree can have multiple strengths and weaknesses in its classifiers XG Boost uses a technique called ”boosting” which makes an ensemble of decision trees that work well together making more accurate predictions. It trains each new tree on examples that the previous tree struggled with before improving the overall accuracy of the model.

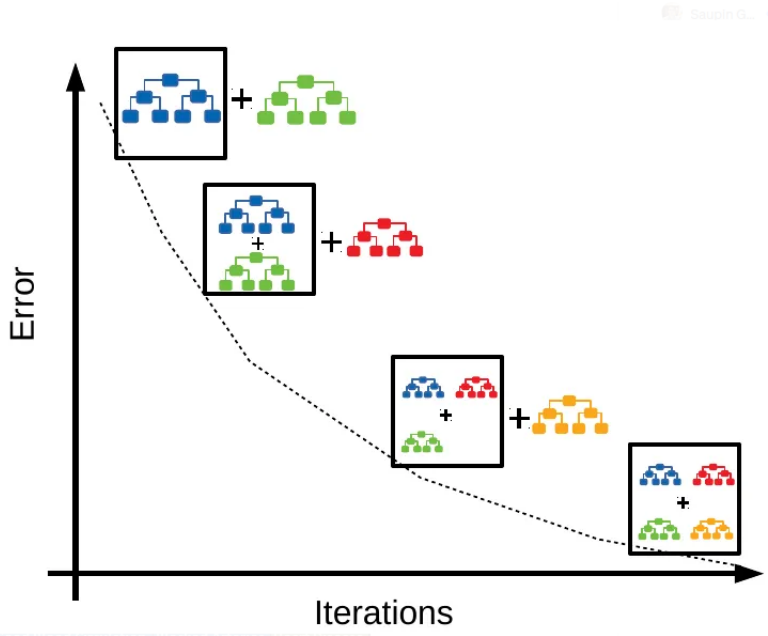

Figure 2.4: Example of how XGBoost reduces error [2]

The main advantages of XGBoost are that it has regularised boosting, so it prevents overfitting. It can handle missing values automatically and it has parallel processing. It can be used with

small datasets and uses a feature called Tree Pruning which creates trees with great depth but then it prunes them (removes unnecessary and irrelevant branches). This improves the overall accuracy of the tree.

The disadvantage to XGBoosting is that it contains a lot of hyperparameters which have to be finely tuned in order to be able to obtain the best possible performance out of the model. XG Boosting can be computationally expensive and quite a complex model which is hard to interpret.

2.1.3 Classification

K Nearest Neighbour

K Nearest Neighbour is part of a different subgroup of machine learning models called Lazy Learning models. Lazy learning is a type of machine learning algorithm that doesn't actually "learn" a model from the training data. Instead, it simply stores the training data in memory and makes predictions based on the similarity between the new input data and the stored training data.

The basic idea behind lazy learning is that it doesn't try to generalize from the training data to make predictions on new data. Instead, it uses the actual training examples as the basis for prediction. This means that lazy learning algorithms can be very flexible and adaptable, as they can adjust to new situations without requiring re-training.

One of the most popular lazy learning algorithms is the k-nearest neighbour (k-NN). This algorithm stores the entire training dataset in memory. When it receives new data, it finds the k-closest training examples to the new data point in terms of some distance metric (usually Euclidean distance). It then uses the labels of these k-closest examples to predict the label of the new data point.

Calculating the Euclidean distance is done using the following formula:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

where $\mathbf{p}$ and $\mathbf{q}$ are n-dimensional vectors representing two points in the Euclidean space. The distance $d(\mathbf{p}, \mathbf{q})$ is the Euclidean distance between the two points. The summation symbol $\sum$ indicates that we need to add up the squared differences between each coordinate of the two vectors, and the square root function $\sqrt{}$ is applied to the sum of the squared differences to obtain the final Euclidean distance.

For the K-nearest neighbour, it uses the k-nearest training examples in the feature space and outputs the most common class among the k neighbours for the classification version of this algorithm.

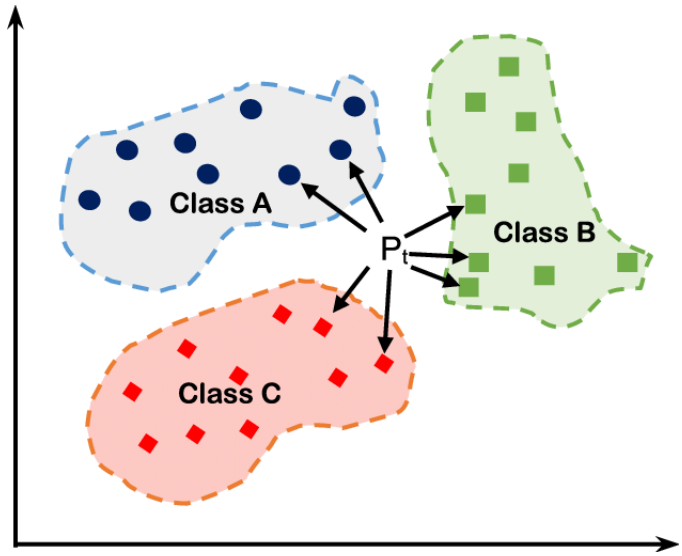

Figure 2.5: Example of KNN classifier [35]

Overall, lazy learning algorithms can be very useful in situations where the underlying data distribution is complex and difficult to model with a traditional learning algorithm. However, they may be less efficient than traditional learning algorithms because they require searching through the entire training dataset for each new prediction.

Random Forest

Random Forest is a Machine Learning technique which is part of the Decision Trees family. It constructs multiple Decision Trees during the training phase each one producing its own outcome. All of the Decision Trees are part of a Random Forest and the output from each one is collected. The most common outcome from the trees is chosen as the final decision.

A Decision Tree is a diagram in the form of a tree that has multiple branches. The branches are determined based on specific criteria in order to classify the data into different branches.

There are several factors that are taken into consideration when constructing a tree. The first one is the entropy or the measure of randomness or unpredictability in the dataset. High entropy means high randomness. Therefore by branching off the data into multiple branches we split the data into different sets which will decrease the entropy factor. Decision nodes determine under which branch the data should fall. They will have two or more branches. A leaf node is a node with no branches which carries the classification. The root node also known

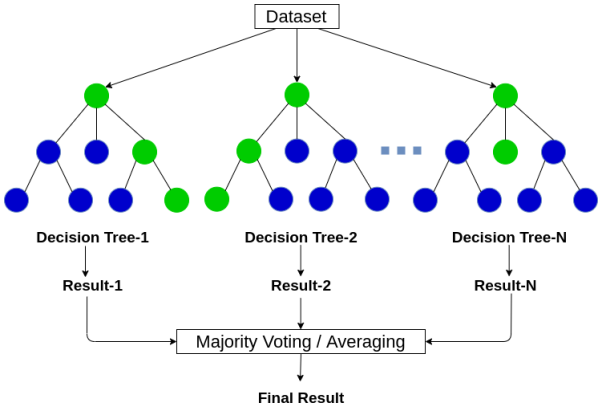

Figure 2.6: Example of a decision tree for animal classification [5]

as the topmost decision node is where all of the data is stored and the first decision has to be made on splitting it up.

The decisions are based on the features we provide in the training data and we decide which feature will be used at what stage of the decision tree based on the highest possible gain. The gain is measured in terms of the decrease in entropy, the higher the decrease the higher the gain.

Random forests amplify the process of decision trees and are very useful when we have high-dimensional data with various numbers of features that we have to observe since they can handle over-fitting pretty well. Another advantage of Random forests is that they handle records with missing data pretty well since even if there are missing features as long as there are some features present, through majority voting an accurate outcome is still possible. Finally, feature importance is another positive since Random Forests can determine which feature has the biggest effect on the outcome.

The main disadvantage of Random Forests is that it is considered a black-box model or in other words difficult to interpret easily. Another issue is that it is a computationally-intensive model especially when you have a large number of trees, hence it is going to be hard to implement for large datasets.

Naive Bayes

Naive Bayes is a probabilistic machine-learning model that belongs to the Generative models family. Models under this family understand the underlying distribution of the data and use it to generate new data points that follow the same distribution.

The model is stated as Naive since it assumes that the features of a data point are in-

dependent of each other making the algorithm computationally efficient. It uses the Bayes theorem[30] to calculate the probability of a statement happening based on historical evidence. In the model, the statement would be the target variable/variables and the historical evidence would be the feature points. Mathematically it is expressed as follows:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Where:

- $P(y|x)$ is the conditional probability of the class label y given the features x.
- $P(x|y)$ is the conditional probability of the features x given the class label y.
- $P(y)$ is the probability of the class label y.
- $P(x)$ is the probability of the features x.

In order to make a prediction, the model uses the Bayes Theorem to calculate the likelihood of each class label happening given the features. The choice is the class with the highest probability class label.

The Naive Bayes classifier is very easily interpretable and easy to implement, does not require large datasets to train the model to a high-performing standard, and can be used for real-time predictions since it is a very time-efficient algorithm and can be applied to a situation where there can be more than two outcome variables when predicting the data. Although it works well with small data it can be susceptible to overfitting in cases where the features are very correlated. It is also sensitive to feature distribution since it assumes that the features are normally distributed.

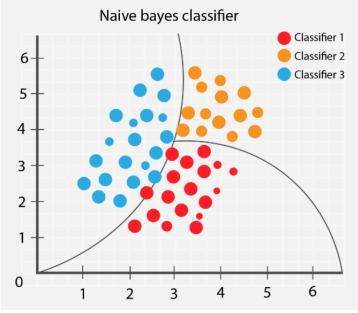

Figure 2.7: Example of Naive Bayes for three classifiers [3]

2.1.4 K-Fold Cross Validation

In order to evaluate the performance of machine learning models a method called K Fold Cross Validation is used. The idea behind the method is to split the data into two parts:

- a training set used to train the model
- a test set used to evaluate the model's performance

k-fold cross-validation, the data is divided into k equally sized folds. The model is trained on k-1 of the folds and tested on the remaining fold. This process is repeated k times, with each fold serving as the test set exactly once.

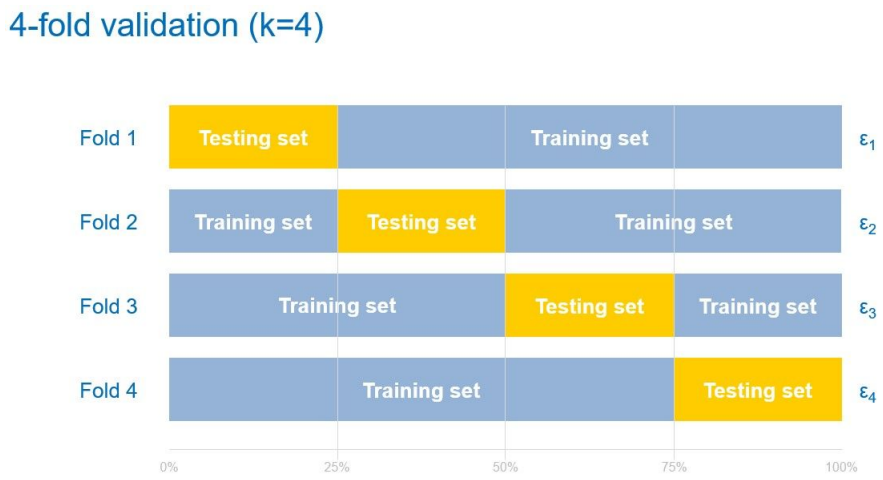

Figure 2.8: Example of k-fold cross validation [7]

In order to get the final accuracy of the model we would calculate the average accuracy across all k folds:

$$Accuracy = \frac{1}{K} \sum_{i=1}^{K} Acc_i \tag{2.1}$$

In this formula, we take the average of the accuracy scores obtained for each fold to obtain the overall accuracy score for the model. This formula provides an estimate of the model's performance that is more robust than using a single train-test split, as it takes into account the variability in performance that may occur when different subsets of the data are used for training and testing.[18]

Time Series K-Fold

Applying K-Fold cross validation is a good method to evaluate your models, however you cannot apply the standard K-Fold to time dependent data. The reason as to why is because, in time dependent data the chronological order of observations matters and the standard k-fold procedure might lead to over-fitting or underestimation of the model's performance.

One commonly used technique to mitigate those issues is the "forward chaining"[33] approach, where the data is again divided into k-folds sequentially and the model is trained on the first k-1 folds and evaluated on the kth fold. However, the difference between "forward chaining" and the normal k-fold is that initially the training set is limited to only the first or the first x number of folds sequentially and the test fold is x+1 and then trained on the model. With each iteration, the training set is shifted forward by one step, and the test set fold always stays as the x+1 where x is the last trainable fold chronologically, which ensures that the test set only contains future observations.

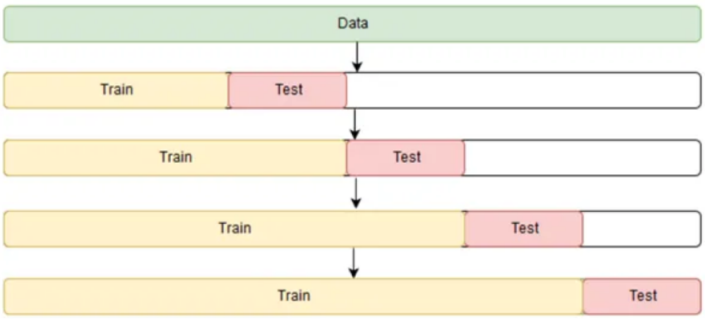

Figure 2.9: Example of k-fold cross validation using forward chaining[35]

2.2 Past applications of Machine Learning techniques for football predictions

This section of the project will focus on the chronological collection of valuable information regarding the topic and justify potential use within the machine learning model that is going to be developed.

2.2.1 Time-Independent Poisson Regression Model

From the research gathered, there are many football prediction models using Regression techniques. The one which seemed to be amongst the most used and popular was the Time-

Independent Poisson Regression Model. This model issues the start of the development of prediction models that have been implemented since the middle of the 20th century from Moroney (1951).[24]

2.2.2 Maher

However, the first major breakthrough for football predictions came from Maher in 1982[29]. He proposed a Poisson model for predicting football match outcomes based on the assumption that the number of goals scored by each team follows a Poisson distribution. The model uses the historical goal-scoring data of the two teams and calculates the expected number of goals that each team will score in the upcoming match. The predicted outcome of the match is then determined based on the difference between the expected number of goals scored by the two teams. The model has been shown to be a simple yet effective method for predicting the outcomes of football matches, particularly for low-scoring games. However, it does not account for factors such as home advantage, team form, and player injuries or suspensions, which can also impact match outcomes.

2.2.3 Dixon and Coles

Building on Maher, Dixon and Coles (1997)[34] created a model which is deemed overall successful which uses the Poisson Distribution model, but in comparison to Maher who used to generate the expected number of goals each team will score, they opted to use it to take into account the home advantage and the correlation between the number of goals scored by each team in a match. The model uses a bivariate Poisson distribution to model the number of goals scored by each team and incorporates a weighting factor to adjust for the home advantage. It also includes a time decay factor to account for the diminishing impact of historical data as the match approaches.

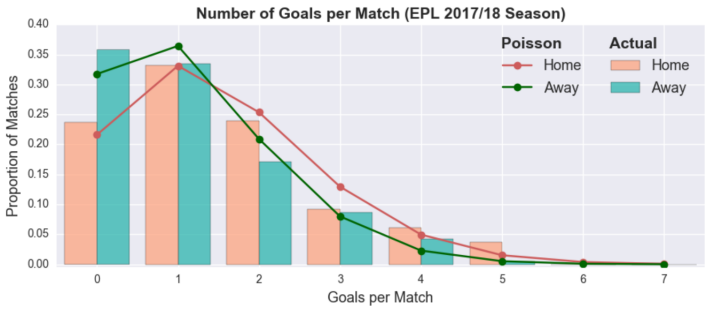

Figure 2.10: Dixon and Coles Poisson distribution for the 2017/18 Premier League season [17]

In addition, based on the past results from each of the teams it generates attacking and defensive strength coefficients for both the home and the away teams.

The formula used is:

$$P(X = k) = \frac{e^{-\lambda} \cdot \lambda^k}{k!}$$

Where:

- $P(X = k)$ is the number of goals scored
- e is the mathematical constant approximately equal to 2.71828
- λ is the average rate at which the events occur in this case the expected number of goals in the game
- $k!$ is the factorial of k, which means the product of all positive integers up to k in our case all the goals up to k

To use the formula, we need to know the average number of goals in a game (λ) and the number of goals we want to calculate the probability for (k). We can then substitute these values into the formula to get the probability of k goals occurring.[40]

The Dixon and Coles model has been shown to outperform the standard Poisson model in predicting football match outcomes, particularly for matches with high-scoring outcomes. However, like the Poisson model, it does not account for all possible factors that can impact match outcomes, such as player injuries or suspensions, and should be used in conjunction with other methods and expert knowledge.

2.2.4 Incorporating Domain Knowledge to the models.

In the 2017 Soccer Prediction Challenge Berrar, Lopes and Dubitzky[20] created a team rating system based on 5 different characteristics, attacking strength for the home team, defensive strength for the home team, attacking strength for the away team, defensive strength for the away team and home advantage. The values were based on goals scored by each team using recent past results from both teams to calculate predictions. In the end, they trained the different models using the generated ratings in order to create predictions. The two most successful models were K-nearest Neighbour and XG Boost, but the key observations that they gathered were that the main challenge in producing an accurate machine learning model is

being able to incorporate domain-specific data into the dataset which is not readily available in many leagues across the world. The dataset used by them was restricted only to goals scored and teams played, but it did not incorporate in-game statistics such as shots taken, yellow cards, red cards etc something that we will take into account when creating our own dataset for the model.

Another key observation they made was the importance of form and time in football. One of their aims was to identify what would be the optimal number of past results based on which they can calculate the 5 different data points mentioned above. They have identified that the accuracy of the model decreases when predicting games at the beginning of the season. This is due to the fact that teams may have strengthened or weakened depending on injuries, player transfers, ownership changes etc. Those are factors that the model cannot take into consideration and at least initially at the beginning of the season cannot be accounted for in the results. One way they resolved this problem is to use the data as one continuous season where for example data from two consecutive seasons is combined into one super season in which the results for the first x matches of the actual season are going to be considered the next x matches of the previous season. Therefore, no data is lost at the beginning of the season. One issue with this approach is the promoted and relegated teams which change every year, but this was addressed by giving the average ratings of the bottom three teams to the newly promoted teams.

2.2.5 Application of Machine Learning Techniques in Team sports

In 2022 Bunker and Susnjak[32] investigated the most commonly used machine learning models used for the most famous team sports in the world. They identified that for football(soccer) the most frequently used models were ANN Artificial Neural Networks. However, they further state that although those were the most frequently used models they are not easy to parameterise and often over-fit especially when the dataset is not sufficiently large enough. In, addition they are not the easiest to interpret and other algorithms such as the second most frequent model Decision Trees are much faster to train and interpret the data.

Although not in the top 2 most accurate models, there was also an inclination towards the use of Bayesian algorithms like Naive Bayes and BNs due to their ability to generate classifiers that do not overfit the data and they deem useful when using small datasets something that will be taken into consideration linking it with the recency feature discussed earlier by Berrar, Lopes and Dubitzky where having very outdated data points will affect the results negatively

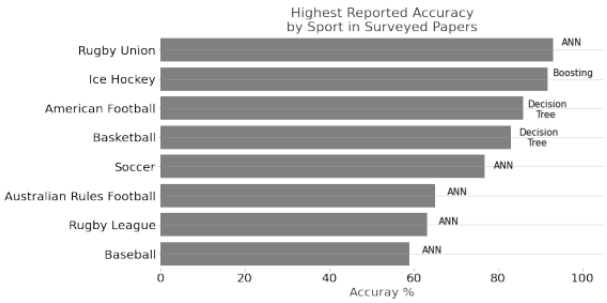

Figure 2.11: The highest recorded accuracy models for each team sport and with what machine learning model [19]

instead of contribute to the accuracy of the model hence having a smaller dataset with more recent data could be a more viable option instead.

In addition to model accuracies, the paper also focused on feature selection. Specifically for football(soccer), they have identified that it is the sport in which are used the most features and that number has increased from around 30 features in 2005 to around 65 in 2020. This is also due to the more accessible and readily available data online however, this does not comply with the accuracy of the model over time.

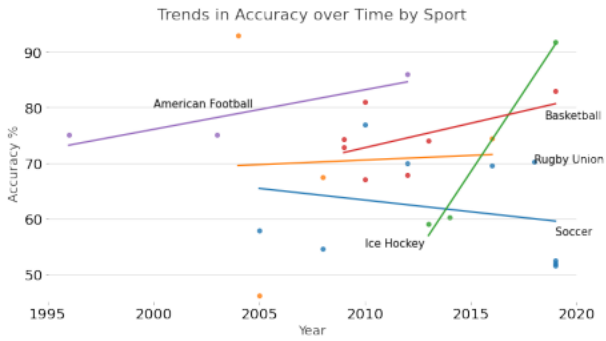

Figure 2.12: Sport accuracies over time [19]

As you can see from Figure 3.3 above, between 2005 and 2022 the accuracy of prediction decreased from around 65% to around 60%. This again supports the idea that having a larger number of matches does not necessarily lead to higher accuracy. The accuracy seems to increase when more features are included per football game to analyse.

One final observation that was discussed in this paper was with regard to the training and validation of the data. For the training of the data successful studies are accomplished either by training the data on a certain number of historical seasons and then testing it on a future season or in the case of using only one season of data available, the model is trained on a

number of competition rounds and tested on the future round within the same season. We will take into account both ways of training the data and produce a hybrid solution which can incorporate both methods together.

With regards to validation, cross-validation was one of the most commonly mentioned methods in a number of studies, but taking into account the time series problem which can be resolved using forward chaining in k-fold cross-validation as explained in section "2.3.4 K-Fold Cross Validation".

Overall, we will use the gathered information from past research on the topic and try to incorporate a Machine Learning Pipeline which includes techniques to sanitise and decide on a dataset that is going to be appropriate, then apply the most common and applicable machine learning models over the years with the use of proven validation techniques to increase the robustness and accuracy of the models produced.

Chapter 3

Requirements and Specification

Now that we have looked at what research has been done in the past both regarding machine learning techniques and their application in football prediction models, in this chapter we will focus on the domain of the project, what is its purpose and state the requirements in a level of priority.

3.1 Football the game

Here we will present a quick introduction to football. We will be focusing on the aspects of the sport that are related to the project itself:
Here is a brief summary of the rules of football/soccer:

- The game is played with two teams of 11 players each.
- The objective is to score more goals than the opposing team by kicking the ball into the opponent's goal.
- Players can use any part of their body except their arms and hands to control the ball (except for the goalkeeper within their penalty area).
- The game is divided into two halves of 45 minutes each, with a 15-minute break at halftime.
- The team with the most goals at the end of the game wins. If the teams have the same number of goals at the end of the game it is deemed as a draw.
- Fouls include any physical contact or dangerous play, and free kicks or penalty kicks may be awarded to the opposing team as a result.

- The ball is out of play if it goes over the touchline or the goal line, and play is restarted with a throw-in, corner kick, or goal kick depending on the place from which the ball went out.

Premier League competition format:

- 20 teams, each playing the other twice, once as the home team at their own stadium and once as the away team in the stadium of the opponent
- A team receives 3 points for a win, 1 point for a draw and 0 points for a loss
- The team with the most points at the end of the season is the champion
- The bottom 3 teams at the end of the season get relegated to the Championship/lower division
- The top 2 teams and the winner of the playoff game from teams 3rd till 6th in the Championship get promoted to replace the relegated teams for the upcoming season.

Main game events:

- Goals scored - a goal is scored when the ball at the back of the net
- Fouls committed - when a player obstructs the play
- Possession - The percentage of time each team has had a hold of the ball throughout the game
- Shots - when a player hits a shot towards the goal of the opposition
- Shots on target - similar to the shots statistics but it only counts shots that were going to be on target
- Yellow Cards - given for less serious offences. If a player receives 2 yellow cards, it turns into a red and he is sent off
- Red Cards - given either as a consequence of two yellow cards or due to a serious offence
- Corners - awarded when the whole of the ball passes over the goal line, on the ground or in the air, having last touched a player of the defending team, and a goal is not scored.
- Crosses - an attempted/accurate pass from a wide position to a central attacking area
- Penalties and Free-Kicks - a penalty kick is awarded if the offence occurred inside the kicker's penalty area unless the kicker was the goalkeeper in which case an indirect free kick is awarded.

3.2 Statistical Football Predictions

Statistical Football Prediction is a method used in sports betting, to predict the outcome of football matches by means of statistical tools[42]. The goal of this process is to outperform predictions set by bookmakers who set the odds on the outcome of football matches. Current methods of statistical analysis include football rankings. The Federation International Football Association (FIFA) assigns those rankings to each of the club and international teams worldwide based on their past game results and in the end the highest rank is assigned to the strongest team. While the ranking system is used worldwide by bookmakers to generate the odds of team A winning against team B, this method has its drawbacks:

- The ranking does not take into account the differences in attacking and defensive strengths
- The main purpose behind ranking the teams is not to predict the results of football games but to sort the teams according to average strength.

However, there are other methods of football prediction known as rating systems. In comparison to the ranking system, rating systems take into account a multitude of factors which include home team advantage, attacking team strength, defensive team strength and the skill set and ability of each player. In order to create an unbiased and accurate prediction model a type of rating system would be required since it allows for much more flexibility and opens up different aspects of the game other than just overall team rank based on current form.

3.3 Dataset

As discussed above, having a rich and meaningful set of historical data will improve the chances of a more accurate prediction model.

Since my focus will be on Premier League predictions I decided to use Football-data [9] since I can download static datasets of a vast number of data points that have happened across games from the 2002/03 season to the current 2022/2023 season. The data also comes with a note file which explains what each data point means inside the CSV file. Since each season's data is collected inside an individual excel spreadsheet I downloaded all the files. I combined it into one big file from which chosen data points will be used for training and testing the machine learning model. A snapshot of the collected data can be visible in the Appendix.

In addition, since the data was not ordered by the round of fixtures, I will also use "fixture-download"[8] to sanitise and reorder the data in a way so that it can be manipulated when

additional values such as attacking and defensive strength need to be added from generated values. This is explained in detail in the Design section of the project.
The data that is included in the dataset include:

- Data regarding fixture itself - venue, referee, home team, away team etc...
- In-game statistics - Home team shots, away team shots, full-time score, yellow cards, red cards, free kicks conceded, corners conceded, shots on target, etc...
- Betting statistics from multiple different betting companies - odds both for home and away win.

For our model, we will be using the data that has been collected from the two most recent complete seasons of the Premier League, 2020/2021 and the 2021/2022 season. The reason behind that is team form and recency bias have a huge impact on team performances and since football is a very competitive sport, team performances shift every season which makes using data from over 2 years old harmful rather than helpful to our accuracy predictions. This gives us 720 football games with 20 different teams. The specific data points that will be used as the features and the target variables for the training and testing of the different machine learning models are discussed in the Design chapter.

3.4 Project Requirements

Based on the project objectives set in section 1.3 (Aim of the project) of the report the following project requirements were established:

3.4.1 Dataset requirements

- D1 - The dataset should include data from a sufficient number of past football games.
- D2 - The features that will be included in the dataset should consist of both in-game statistics and season-form statistics of different football teams.
- D3 - The data should be in a structured format and appropriately sanitised for any outliers or incompatible data types.
- D4 - The data should be appropriately labelled to identify the target and feature variables.

3.4.2 Machine Learning pipeline requirements

- M1 - The pipeline should be able to successfully calculate the attacking and defensive strengths of the home and away teams for each round of fixtures based on the historical data provided as well as the home team advantage.
- M2 - The pipeline should provide a platform for easily extendable and maintainable ways of training and testing different machine learning models.
- M3 - The pipeline should provide some sort of validation for the trained models.
- M4 - The pipeline should provide a way of optimising the different machine-learning models.
- M5 - The pipeline should provide appropriate data manipulation of the provided dataset for the needs of the different machine learning models.

3.4.3 Evaluation requirements

- E1 - The software should provide appropriate evaluation metrics based on which the performance of the different Machine Learning models can be compared.
- E2 - The software should provide a clear representation of the results obtained during the evaluation of the different models.
- E3 - The software should include additional evaluation of the models in comparison to existing benchmark models.

3.5 Requirements specification

For each of the aforementioned requirements, we will now establish their specification and level of priority. There are three levels - low, medium and high. Some requirements are dependent on others due to the nature of a machine-learning project. For example, adding potential optimisation techniques for the different machine learning models depends on the completion of the ability of the software to apply training and testing for each of them. Therefore, the requirements which focus on the core features for the dataset manipulation, pipeline construction and evaluation process will be of High priority and any Medium or Low priority features will be built on top of them.

Requirement ID	Requirement	Specification	Priority
D1	The dataset should include data from a sufficient number of past football games.	The dataset consists of 720 games from the 2020/21 and 2021/22 Premier League seasons across 20 different teams.	High
D2	The features that will be included in the dataset should consist of both in-game statistics and season-form statistics of different football teams.	For in-game statistics the dataset takes into account Home/Away team: names, goals, shots, shots on target, fouls committed, corners, yellow cards, red cards, full-time result. For team form statistics the dataset takes into account: Home/Away team: attacking strength, defensive strength, overall home advantage and fixture round.	High
D3	The data should be in a structured format and appropriately sanitised for any outliers or incompatible data types.	It uses a Label encoder to transform any string values in the dataset such as team names to be of integer type. For sanitisation of the data, it uses a Min-Max scaler to reduce the risk of overfitting and underfitting the modules.	Medium
D4	The data should be appropriately labelled to identify the target and feature variables.	The feature variables will be all the in-game and team form data points explained in requirement D2, but the target variable will be the full-time result value label encoded as 1 for a home team win, 0 for a draw and -1 for away team win.	High

M1	The pipeline should be able to successfully calculate the attacking and defensive strengths of the home and away teams for each round of fixtures based on the historical data provided as well as the home team advantage.	For this purpose the Poisson distribution model will be used to calculate the different parameters for each team per fixture round.	High
M2	The pipeline should provide a platform for easily extendable and maintainable ways of training and testing different machine learning models.	An inheritance OOP Architecture is introduced to classify the different models based on Regression and Classification type of models	Medium
M3	The pipeline should provide some sort of validation for the trained models.	A time series K-fold Cross-validation functionality is introduced to evaluate the performance of the models across the full dataset	High
M4	The pipeline should provide a way of optimising the different machine-learning models.	Applies hyperparameter tuning for each of the models by applying GridSearch to find the optimal combination of values from each of the hyperparameters	Low
M5	The pipeline should provide appropriate data manipulation of the provided dataset for the needs of the different machine learning models.	Concatenation of the in-game data points with the calculated team form ratings as well as ordering the data based on the round of fixtures and not date using Pandas.	High
E1	The software should provide appropriate evaluation metrics based on which the performance of the different Machine Learning models can be compared.	Accuracy and Precision metrics for Classification models and Root Mean Squared Error and Mean Absolute Error for Regression models	High

E2	The software should provide a clear representation of the results obtained during the evaluation of the different models.	Uses Matplotlib to represent the comparison performance metrics for the results in tabular format for each of the compared machine learning models.	High
E3	The software should include additional evaluation of the models in comparison to existing benchmark models.	Applies three benchmark models: Dixon and Coles, Random pick and Home team wins only probability	Medium

Table 3.1: Requirements Specification Table

Chapter 4

Design

In this chapter, we will delve into the overall design of our model describing each section of the constructed pipeline and the reason behind those design choices. The design undertook a couple of iterations throughout which it expanded its overall structure from a basic comparison between the Poisson Regression model and Knn classifier to a more maintainable and flexible pipeline which can sanitise and construct a viable dataset, validate the accuracy of different machine learning models that will use the aforementioned dataset, make a final evaluation to identify the most accurate Classification and Regression models and then evaluate its accuracy with already existing betting company results.

4.1 Design Structure

The diagram in Figure 4.1 shows the whole pipeline structure. The pipeline is divided into three main parts:

- Part 1 - Generating the dataset
- Part 2 - Training, Testing and Validating Machine Learning models
- Part 3 - Evaluation stage

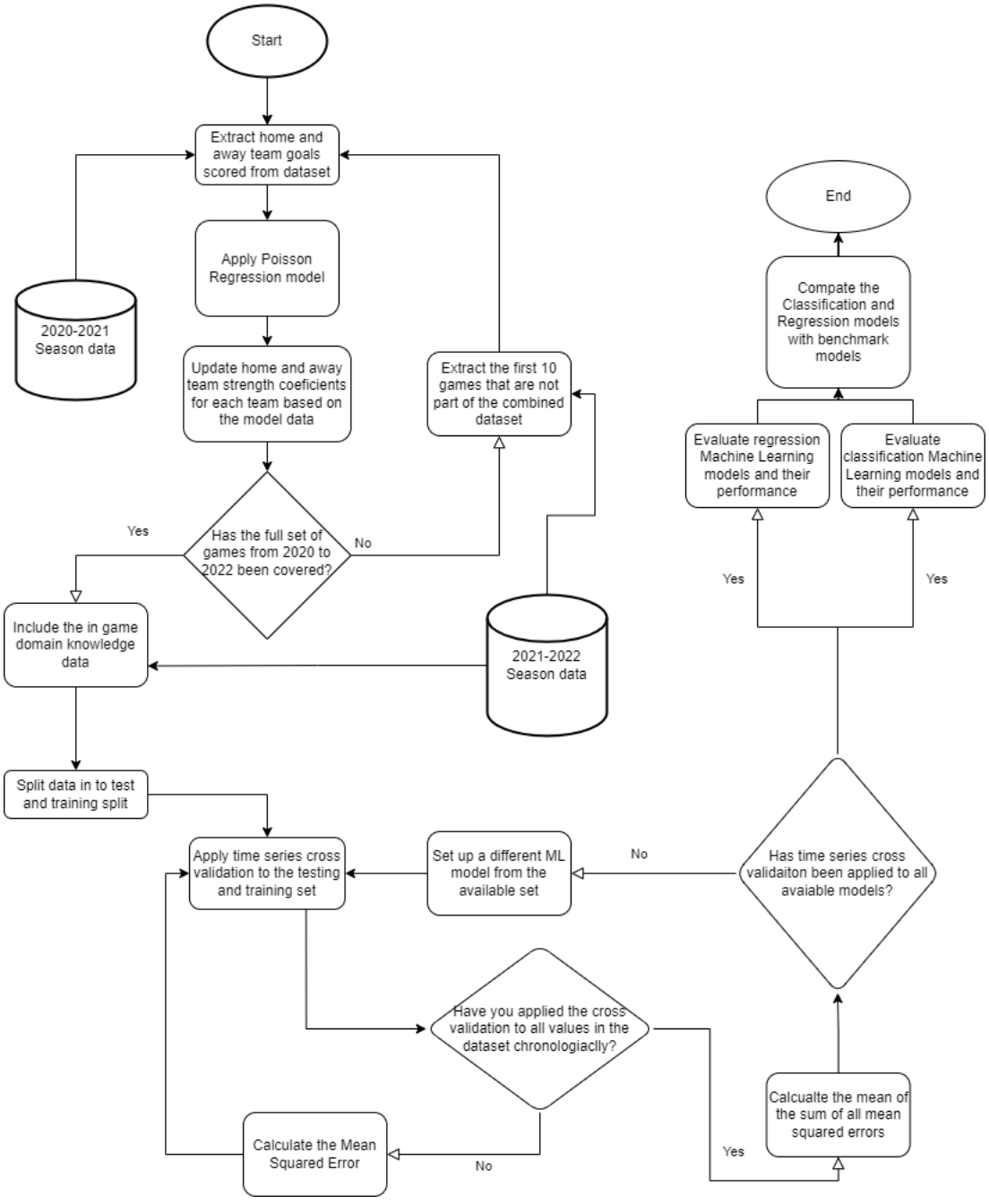

Figure 4.1: Overall design of the Machine Learning pipeline

4.2 Generating the dataset

In order to build a robust dataset that will be used to train and test the different machine-learning models we will incorporate multiple features that will be split into two categories:

- In-game statistics
- Team ratings

4.2.1 In game statistics

The in-game statistics will be used to take into account unpredictable in-match events such as red cards or shots on target. Those values are important since this allows us to incorporate domain-specific information about each individual game and that way we will not rely solely on team pre-match performance. Here are all the in-game statistics that will be used and the reason behind them:

1. Date - this feature will be used for sorting purposes since all the fixtures in the dataset are sorted by date of play.
2. Home/Away team Yellow cards - the more yellow cards a team receives the riskier it becomes for them to receive a red card hence being one man down. A study on the influence of red and yellow cards on team performance in 2022[27] showed that teams with more yellow cards become more conservative and tend to get dominated by the opposition because the players on yellow cards will not be as confident going into tackles.
3. Home/Away team Red Cards - they are a pivotal point in any football game since the opposition will have more players on the pitch which more often than not means they have a much higher chance of winning the game.
4. Home/Away team Possession - teams with more possession of the ball throughout the game tend to be the more dominant force, however not always the most threatening, since many teams play on the counterattack. - Not in the dataset - could be concatenated from other datasets!
5. Home/Away team shots - teams with the higher number of shots tend to have a better result at the end of the game simply because they have made more attempts on goal.

6. Home/Away team shots on target - teams with the higher number of shots on target tend to have a better result at the end of the game simply because they have made more accurate attempts on goal and have managed to test the goalkeeper more often.

7. Home/Away team corners - Corners throughout the game give great opportunities to score a goal. Many teams in the past have utilised this part of the game over the year and have built their goals scoring strategies upon them.

All of those statistics are readily available and do not have to be generated using any algorithm

4.2.2 Team ratings

Team ratings are used to include a grasp of the overall form of each team going into the fixture taking into account all the games that they have played in the past. This will allow us to have a 2-dimensional look at the performance of the home and away teams by considering both in-game statistics and past performance-related statistics. The following team ratings will be generated:

- Home team attacking strength
- Home team defensive strength
- Away team attacking strength
- Away team defensive strength

In addition, a home advantage coefficient will also be generated which will calculate the overall league advantage that home teams have up to the fixture that is being played. That way we take into consideration the home advantage in football which is a very significant parameter.

4.2.3 Data construction

As you can see in figure 4.2, the process begins with the extraction of the Premier league data from the 2020/2021 season. The only parameters that will be extracted from this dataset are the home and away teams, and the goals scored by both teams. Once the data is extracted the Poisson regression model will be applied. The objective of the regression model is to train the collected data and generate values for the attacking and defensive strength of each team as well as the current home team strength value. Once the values are generated they will be applied to the latest round of fixtures for the 2021/2022 season. For this purpose, before we

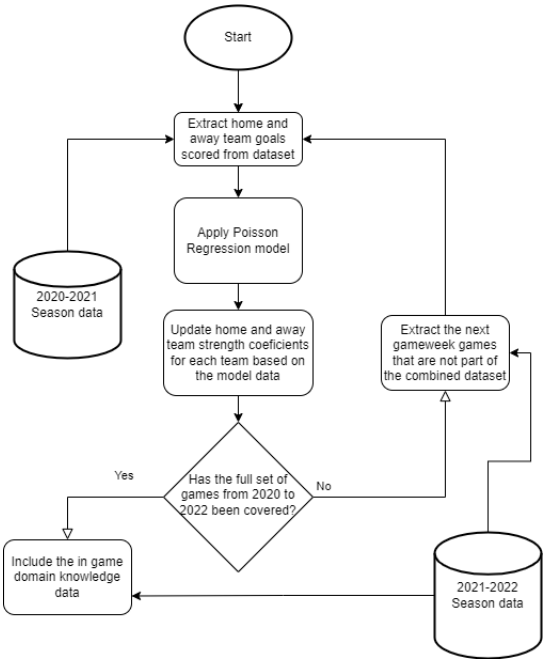

Figure 4.2: Data extraction and generation

begin iterating through the 2021/2022 season dataset, we will sort the fixtures based on game week. Each game week consists of all the teams playing in one game, so for example for the Premier League there are 20 teams, so 1 round of games will consist of 10 games in total. Once the dataset is sorted the algorithm will extract the next round of fixtures and concatenate them to the previous rounds of fixtures which also include the data from the previous season. Then with the updated dataset, the Poisson Regression model is retrained and the new coefficients are generated to be applied to the next round of fixtures. Once all the rounds of fixtures have been covered the values are concatenated with the in-game statistics. The dataset is ready for training and testing on multiple machine-learning models.

In the implementation section, the full process will be explained with provided examples of the generated coefficients and the necessary software that is being used.

4.3 Training, Testing and Validating Machine Learning models

Now we are going to focus on the second section of our pipeline in which we run cross-validation on multiple models estimating the mean evaluation metrics both for Regression and Classification models.

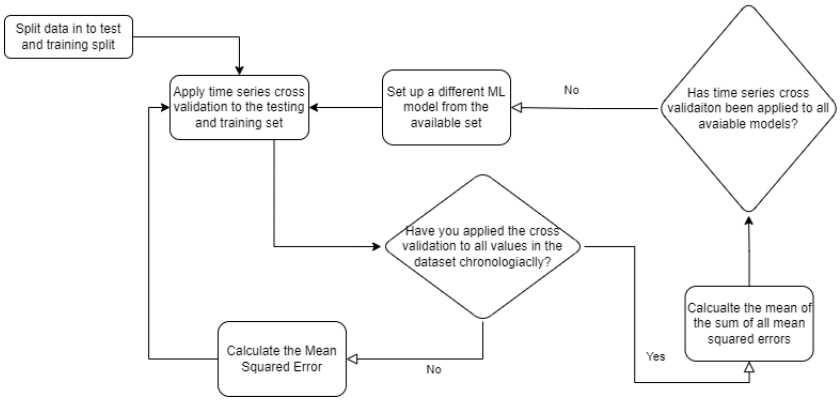

Figure 4.3: Training, testing and Validation

4.3.1 Cross Validation

Once the full dataset is combined from Section 1, the data is split into training and testing data. Since we are going to apply Time Series cross-validation on the data, the training and testing data will be split into x number of folds depending on how many are set based on the date of the game and more importantly the round of fixtures. The cross-validator will apply forward chaining and update the training and the testing data appropriately so that with every new fold larger and larger percentage of the dataset is being covered with the final fold covering all of the datasets with a specific ratio split between how much of it is training and how much testing data. Then a mean value out of all the folds will be calculated for each of the evaluation metrics used to evaluate the models during the cross-validation. Since we are cross-validating both classification and regression models each one will have different evaluation metrics because specific metrics are more accurate assessors for Regression and Classification types of models. The metrics will be explained in detail during the Evaluation section of the pipeline. Once all the models have been cross-validated and their mean metrics have been extracted the values are sent for evaluation to determine the most accurate Classification and Regression model.

4.3.2 Design of Machine Learning models

As shown in Figure 4.4 the way we will carry out all the machine learning processes for each model will be through the use of this inheritance class diagram. The models have a superclass called Model which contains the scale, train and predict methods since they do not change throughout any of the models. The second level of the inheritance tree adds the specific type of the model as well as the specific evaluation metrics that will be stored. In the last level of the inheritance, the tree will consist of the individual classes for each specific model. The

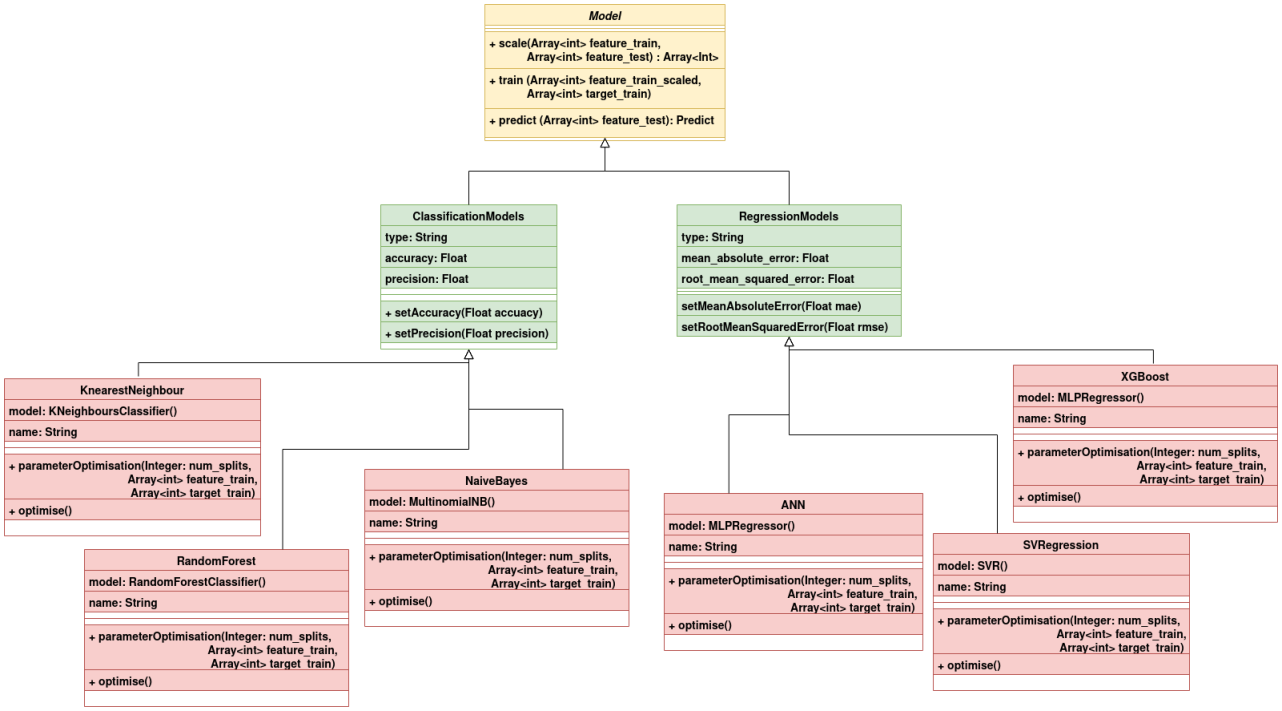

Figure 4.4: Class diagram of the Machine Learning Model Structure

features that will be stored here are an instance of the specific class for the model in skilearn library and the specific name of the model. In addition, each of the classes contains the methods parameterOptimisation and optimise. The first one carries out the GridSearch time series cross-validation to find the most optimal combination of hyperparameters specific to that model. The optimise() methods set up the aforementioned hyperparameters for the specific model. Having this type of structure will improve the maintainability of the code and allow for the quick and easy integration of new additional models into the pipeline.

Now that we have established the overall design of the classes we will focus on the different hyperparameters we will optimise for each of the chosen models.

K-nearest Neighbour

- k number or the number of nearest neighbours - determines the number of nearest data points to be considered for a prediction. For example, if k=3 that means that KNN will use the three nearest data points and depending on the majority of the points classified under a specific label, that data point will also be classified with the same label.
- weight - determines the weight given to each of the k neighbours. There are two types, ”weighted” which assigns a higher weight to values which are closer neighbours than others and there is ”uniform” which gives equal weight to each of the k neighbours.

- ”p” value also known as the power variable - determines what distance is going to be measured between the data points. There are two ways to measure the distance, using ”Manhattan” or ”Euclidean” distance.

Artificial Neural Network

- hidden layer sizes - this determines the number of hidden layers and for each hidden layer the number of neurons that will be used.
- activation - this is the hyperparameter which determines what activation function will be applied to the sum of the weightings and the bias of the node. The activation function allows the model to model complex non-linear relationships between the data points. The most common ones are ”relu”, ”logistic”, ”tanh”.
- solver - the solver determines the algorithm that is going to be used during backpropagation. The three solvers we will run our model on, are ”sgd”, ”adam”, ”lbfgs”.
- max iter - this determines the maximum number of iterations that the model will undertake. An iteration consists of a forward and backpropagation cycle. The number of iterations does not necessarily change positively the error rate and could lead to overfitting of the data if overiterated.

Random Forest

- n estimators - this parameter is used to determine the number of decision trees that will be used in the model. The aim is to find the balance where the number of trees is optimal so that it does not lead to under or overfitting.
- random state - It controls the selection of random samples and features for each split in the tree.

Supported Vector Regression

- kernel - this is the kernel function that is going to be applied to the data points to transform them into a higher-dimensional feature space. There are several different kernel functions, but we will evaluate our model with the following two:
  - linear - usually used for linear data it computes the dot product between the input features.

  - rbf - usually used for non-linear data it calculates the Euclidean distance between input feature vectors

- C - it is used to penalise the misclassifying training examples. A smaller value for C will allow for more training examples to be misclassified. However, larger values will result in fewer misclassification. Balancing between having a low training error and a high margin would improve the performance of the model.

- gamma - it determines the shape of the kernel and the level of influence each training example has on the model. It controls the flexibility of the model and the degree of overfitting.

Naive Bayes

- alpha - Since the Bayes classifier estimates the probability of each feature based on the number of occurrences it has in comparison to the total number of features, in the cases where this feature is non-existent in a specific class Bayes gives it a 0 value. This can be problematic when we want to calculate the posterior probability. Therefore the alpha hyperparameter helps with that by adding a set number which is bigger than 0 that replaces the features with the initially estimated probability of 0. This smoothens the curve and can reduce overfitting, however, if we choose too big of an alpha value this could lead to underfitting.

XG Boost

- learning rate - adjusts the weights on each step of training.

- number of estimators - It determines the number of decision trees that are going to be trained.

- maximum depth - it gives a maximum depth of each of the decision trees. Preventing a bigger depth limits the model from memorising the training data.

4.3.3 Evaluation Process

Once the cross-validation for each of the models is completed and the evaluation metrics for each model are collected, we will carry out two comparisons.

The first comparison is between the different types of models, so all Regression models will be compared with their evaluation metrics and the same will be done with the Classification

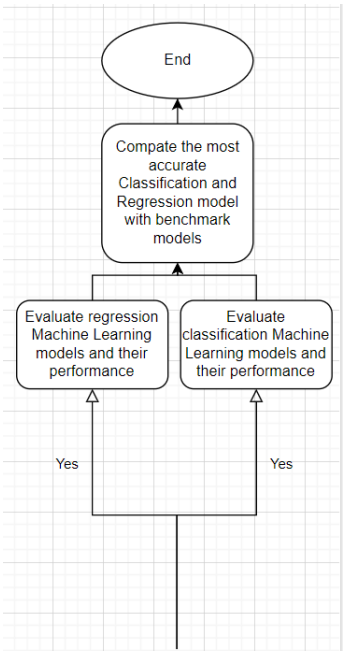

Figure 4.5: Evaluation section of the pipeline

models. They will initially be compared in their default settings without any hyperparameter tuning but in the second comparison, they will be optimised to identify the difference in performance before and after optimisation and to also see which models benefited the most.

The second comparison will consist of comparing the chosen Regression and Classification models to chosen benchmark models. The benchmark models would provide us with an external look as to how the models we have chosen to train and test our dataset compare in performance to work that has been done in the past. In particular, we will implement a version of the Dixon and Coles model (for more information regarding the model you can go back to the Literature Review Chapter). In addition, we will also compare it with some common naive approaches such as Random probability and Home team wins only.

We will now talk about the various evaluation metrics that will be applied to each of the models. Different metrics will be used for Regression and Classification models since they give a more accurate evaluation of the model.

Regression models evaluation metrics

- Mean Absolute Error MAE - it measures the average magnitude of errors. It calculates the average absolute difference between the actual and predicted values:

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

Where:

- $\hat{y}_i$ is the predicted value from the regression model
- y is the corresponding actual value
- n is the number of testing records available

A lower MAE indicates a lower error rate hence the predicted values are closer to the actual values. Although useful this metric is quite sensitive to outliers because it gives equal weight to all errors. If there are outliers this can significantly affect the final result.

- Root Mean Squared Error RMSE - Unlike MAE, RMSE takes into consideration both the direction and the magnitude of the errors hence it is more sensitive to outliers. Essentially, it calculates the square root of the average of the squared difference between the actual and the predicted variable:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where:

- n is the number of samples from the dataset
- $\hat{y}_i$ is the predicted value
- y_i is the actual outcome
- i is the sample index

RMSE has its drawbacks too such as being harder to interpret than MAE and having the assumption that data is normally distributed whereas MAE does not.

Overall, RMSE and MAE complement each other very well with both providing different perspectives on the performance of the model. RMSE can provide a more thorough analysis of any outlier cases whereas MAE will show the overall accuracy of the model.

Classification model evaluation metrics

- Accuracy - Accuracy is the most common evaluation metric for classification machine learning models. It measures the proportion of correct predictions in comparison to the total number of predictions made. The most common accuracy formula is for binary class

classification problems, however in our case since we can have three possible outcomes loss, win and a draw we will use a multi-class version of the formula as shown below:

$$Accuracy = \frac{\sum_{i=1}^{K} ConfusionMatrix_{ii}}{\sum_{i=1}^{K} \sum_{j=1}^{K} ConfusionMatrix_{ij}}$$

Where:

- $ConfusionMatrix_{ii}$ The number of instances that belong to a specific class i(for example draw) that are correctly classified.
- $ConfusionMatrix_{ij}$ The number of instances that should be of class i but are classified as class j (in our case that could be either a win or a loss)

Figure 4.6: Example of a Confusion Matrix[1]

The formula states that we sum the number of correctly classified instances inside the Confusion Matrix which would mean all the diagonal values inside the matrix and then we divide this sum by the sum of all the values in the matrix as shown in Figure 5.5.

The accuracy evaluation technique is very easy to interpret, however, it ignores the cost of misclassification and treats each such case indifferently, so in the cases where the cost of false positives for example can be much higher than the other outcomes, but it is given the same weight as the others.

- Precision - Precision measures the number of true positives out of all the instances predicted as positive. Since this is a classification model that consists of more than 2 classes the precision formula is shown as the average of the true positive values from each class.

This is also known as "macro-averaged precision" as stated by Richter in his paper in 2021 about "Machine learning in sports science"[31]:

$$Precision_{macro} = \frac{1}{k} \sum_{i=1}^{k} \frac{TP_i}{TP_i + FP_i}$$

Where:

- k is the number of classes
- i is the index of a specific class
- TP_i are the true positive instances in class i
- FP_i are the false positive instances in class i

Precision is a more specific measure than accuracy and that makes it more robust in imbalanced datasets. However, it cannot be used on its own to evaluate a model since it does not take into account false negatives and it will not give a complete picture of the model's performance.

Therefore, complimenting precision with accuracy would give us a more complete evaluation of the classification models.

Chapter 5

Implementation

We are now going to go over the implementation of the different sections of the pipeline. This will be a continuation of the design process discussed in chapter 4, but we will delve deeper into how each component is implemented and what software has been used to both support and produce each algorithm. Since it is an extension of the design process we will follow the same 3 part structure going from generating the data to the evaluation stage.

5.1 Resources used

In order to construct the pipeline the following libraries were used:

- Statsmodels[17] - A Python module that performs statistical modelling and analysis. We used it to train the Poisson regression model using the generalised linear model under the Poisson family.
- Pandas[14] - Used to convert all the datasets from various types into data frames which can then be easily manipulated and trained.
- Python[15] - the programming language that was used to construct the whole pipeline. Python was chosen because of its extensive Machine Learning libraries and extensions as well as its ease of use and speed in training and testing the machine learning models.
- Numpy[12] - Used to support the data manipulation that is performed during each iteration of the coefficient generation for the next round of fixtures since it supports multi-dimensional arrays and high-level mathematical functions

- Scipy[16] - Used to interpret the probability mass function in order to calculate the probability of the set number of goals being scored by each team
- Matplotlib[11] - Used to visualise the generated data from the Poisson Regression model
- fixtureDownload[8] - Used this dataset to sort all the fixtures based on the round number and not the date of play.
- GitHub[22] - Used for version control and as a backup for the source code repository.

5.2 Generating the dataset

5.2.1 Calculating the overall league statistics

The first thing that we need to do is to calculate those overall statistics for all the teams in the league and all games played:

- **H** - Total number of goals scored by each team since the start of the season at home
- **A** - Total number of goals scored by each team since the start of the season away from home.
- **G** - Total number of games played in the league by each team.

After those three initial values are collected we want to calculate the average goals scored by teams at home and away:

$$AH = \frac{H}{G}$$

$$AA = \frac{A}{G}$$

Where:

- AH Average number of goals scored by the home team
- AA Average number of goals scored by the away team

5.2.2 Home and Away Team statistics

Next, we need to calculate the same values but for the chosen home team H and chosen away team A:

- **H** - Total number of goals scored by home/away team since the start of the season at home.
- **A** - Total number of goals conceded by home/away team since the start of the season at home.
- **G** - Total number of games the home/away team played in the league.

Once those values are collected we calculate the averages again:

- Once for the home team:

$$HG = \frac{H}{G}$$

$$HC = \frac{A}{G}$$

- Once for the away team:

$$AG = \frac{H}{G}$$

$$AC = \frac{A}{G}$$

In the end, we should have a table resembling something like this if the two teams were Arsenal and Man United:

	Goals scored at home by all Premier League teams	Away goals scrored by all Premier League teams	Home goals scored by Arsenal	Home goals conceded by Arsenal	Away goals scored by Man Utd	Away goals conceded by Man Utd
Goals	582	436	57	12	34	12
Matches	380	380	19	19	19	19
Goals/Matches	1.532	1.147	3	0.63	1.78	0.63

Figure 5.1: Goals statistics for Arsenal and Man United[21]

5.2.3 Attack Strength and Defence Strength and Goal expectancy

Based on the averages collected we want to calculate the attack and defence strengths of the home and away teams using the following equations:

$$ASH = \frac{HS}{ASH}$$

$$DSH = \frac{HC}{ACH}$$

Where:

- ASH Attack Strength Home Team
- HS Average home team goals scored
- ASH Average goals scored Home from each team in the league
- DSH Defence Strength Home Team
- HC Average home team goals conceded
- AC Average goals conceded Home from each team in the league

and we do the same for the away team

$$ASA = \frac{AS}{ASA}$$

$$DSA = \frac{HC}{ACA}$$

Where:

- ASA Attack atrength away team
- AS Average away team goals scored
- AS Average goals scored away from each team in the league
- DSA Defence strength away team
- HC Average Away team goals conceded
- AC Average goals conceded away from each team in the league

Then we calculate the home team and the away team goal expectancy:

- $HTGE = ASH \times DSA \times AHGS$
- $ATGE = ATAS \times HTDS \times AAGS$

Where:

- $HTGE$ Home team goal expectancy
- ASH Attack strength home team

- DSA Defence strength sway team
- $AHGS$ Average home goals scored
- $ATGE$ Away team goal expectancy
- $ATAS$ Away team attacking Strength
- $HTDS$ Home team defensive strength
- $AAGS$ Average away goals scored

5.2.4 Applying the Poisson Distribution model

In our case, λ will be the home team and away team goal expectancy and we are going to run the Poisson distribution model for both the home and away teams independently. The interval (k) will be the number of goals each team can score. Since football rarely has a high number of goal-scoring games we can set the highest number to 5 for now. That means that when we run the Poisson distribution model we will get a breakdown of the likelihood of each of the possible results happening.

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{k!}$$

Note: The value of (e) is the Euler constant:

$$e = 2.71828...$$

An example of the application of the model with home team Arsenal and away team Manchester United and received the following results:

	20.97 % goals: 0	32.75 % goals: 1	25.59 % goals: 2	13.32 % goals: 3	5.2 % goals: 4	1.63 % goals: 5
26.25% goals: 0	5.5	7.36	4.92	2.2	0.73	0.2
35.11% goals: 1	8.6	11.5	7.69	3.43	1.15	0.31
23.48% goals: 2	6.72	8.98	6.01	2.68	0.9	0.24
10.47% goals: 3	3.5	4.68	3.13	1.39	0.47	0.13
3.5% goals: 4	1.37	1.83	1.22	0.54	0.18	0.05
0.94% goals: 5	0.43	0.57	0.38	0.17	0.06	0.02

Figure 5.2: Goals statistics for Arsenal and Man United

The values in yellow represent the probability of the score being a draw separated into the 5 possible outcomes. The values in red are for victories of the home side and the ones in green

are for victories of the away side. What the Poisson distribution model did is calculate the probability of how many goals the home team and the away team will score based on the value of their expected goals. What the table shows is a breakdown of the probabilities of all possible results up to 5 goals.

When we add up all the values for an away win, draw and home win we get the following as our final result:

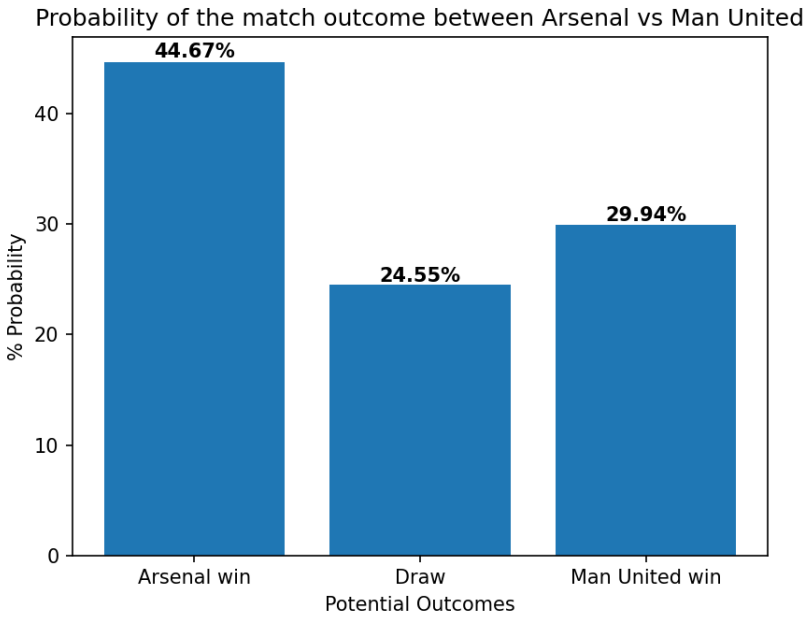

Figure 5.3: Goals statistics for Arsenal and Man United

5.2.5 Training the Poisson Regression model

The results we have achieved thus far were based on statistical modelling but we weren't actually training the model to achieve such results but instead just applying the distribution model on the goal expectancy to the home and away team. Now that we've established what we need for the model we can train it with the data from the dataset. For this purpose, we first need to establish the goal expectancy of each team at home and away by training the model. In addition to the home and away team goal expectancy, we will also train the model on the expected home team advantage since this is another major factor in many football games. Out of all of the values shown in the graphic above the one which interests us is the "coef". The coefficient shows us the strength of each team both attacking and defensively. This value is

Generalized Linear Model Regression Results

Dep. Variable:	goals	No. Observations:	1500
Model:	GLM	Df Residuals:	1460
Model Family:	Poisson	Df Model:	39
Link Function:	Log	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-2165.6
Date:	Wed, 01 Mar 2023	Deviance:	1701.3
Time:	17:12:58	Pearson chi2:	1.52e+03
No. Iterations:	5	Pseudo R-squ. (CS):	0.1864
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	0.2051	0.146	1.400	0.161	-0.082	0.492
attackStrength[T.Aston Villa]	-0.0897	0.135	-0.665	0.506	-0.354	0.175
attackStrength[T.Brentford]	-0.5507	0.154	-3.569	0.000	-0.853	-0.248
attackStrength[T.Brighton]	-0.3720	0.145	-2.557	0.011	-0.657	-0.087
attackStrength[T.Burnley]	-0.5573	0.154	-3.610	0.000	-0.860	-0.255
attackStrength[T.Chelsea]	0.1137	0.128	0.890	0.373	-0.137	0.364
attackStrength[T.Crystal Palace]	-0.2364	0.140	-1.686	0.092	-0.511	0.038
attackStrength[T.Everton]	-0.2709	0.142	-1.907	0.057	-0.549	0.008
attackStrength[T.Leeds]	-0.0983	0.136	-0.725	0.469	-0.364	0.168
attackStrength[T.Leicester]	0.1169	0.128	0.912	0.362	-0.134	0.368
attackStrength[T.Liverpool]	0.3090	0.122	2.527	0.011	0.069	0.549
attackStrength[T.Man City]	0.4337	0.119	3.645	0.000	0.200	0.667
attackStrength[T.Man United]	0.0856	0.129	0.663	0.507	-0.167	0.339
attackStrength[T.Newcastle]	-0.2602	0.142	-1.838	0.066	-0.538	0.017
attackStrength[T.Norwich]	-0.8229	0.169	-4.859	0.000	-1.155	-0.491
attackStrength[T.Southampton]	-0.2417	0.141	-1.713	0.087	-0.518	0.035
attackStrength[T.Tottenham]	0.1535	0.127	1.213	0.225	-0.095	0.402
attackStrength[T.Watford]	-0.5326	0.154	-3.450	0.001	-0.835	-0.230
attackStrength[T.West Ham]	0.0233	0.131	0.178	0.859	-0.233	0.280
attackStrength[T.Wolves]	-0.4464	0.149	-2.997	0.003	-0.738	-0.154
defenceStrength[T.Aston Villa]	0.1271	0.149	0.854	0.393	-0.165	0.419
defenceStrength[T.Brentford]	0.3169	0.142	2.230	0.026	0.038	0.595
defenceStrength[T.Brighton]	0.0275	0.152	0.181	0.856	-0.270	0.325
defenceStrength[T.Burnley]	0.1966	0.146	1.348	0.178	-0.089	0.482
defenceStrength[T.Chelsea]	-0.1988	0.162	-1.225	0.220	-0.517	0.119
defenceStrength[T.Crystal Palace]	0.2445	0.145	1.687	0.092	-0.039	0.529
defenceStrength[T.Everton]	0.2730	0.144	1.899	0.058	-0.009	0.555
defenceStrength[T.Leeds]	0.4117	0.140	2.938	0.003	0.137	0.686
defenceStrength[T.Leicester]	0.2565	0.145	1.769	0.077	-0.028	0.541
defenceStrength[T.Liverpool]	-0.2034	0.163	-1.248	0.212	-0.523	0.116
defenceStrength[T.Man City]	-0.3571	0.171	-2.083	0.037	-0.693	-0.021
defenceStrength[T.Man United]	0.1721	0.148	1.165	0.244	-0.118	0.462
defenceStrength[T.Newcastle]	0.3370	0.142	2.374	0.018	0.059	0.615
defenceStrength[T.Norwich]	0.4326	0.139	3.116	0.002	0.161	0.705
defenceStrength[T.Southampton]	0.4302	0.139	3.089	0.002	0.157	0.703
defenceStrength[T.Tottenham]	0.0236	0.154	0.153	0.878	-0.278	0.325
defenceStrength[T.Watford]	0.5544	0.136	4.083	0.000	0.288	0.821
defenceStrength[T.West Ham]	0.1261	0.149	0.846	0.398	-0.166	0.418
defenceStrength[T.Wolves]	0.0877	0.150	0.585	0.559	-0.206	0.381
home	0.0689	0.044	1.563	0.118	-0.017	0.155

Figure 5.4: attacking and defensive strengths of each team for one round of fixtures

extracted as the exponent of the parameter values:

$$e^x$$

The higher the value of "attackingStrength" the more goals are expected to be scored by that team and vice versa. For example, for the home advantage, it can be seen from the data that there is a 0.0689 swing towards scoring more goals. If you convert this into the probability then we have:

$$e^{0.0689} = 1.071$$

This captures the fact that the home team has a 1.071 times more likelihood to score a goal than their opponent. In that case, it is a comparison between any home and away team from the training data used. In contrast to the attacking strength, the away team strength is higher if the coefficient is more negative. For example, from the training data, we can see that Man City's defence strength has a -0.3571 coefficient which is the lowest out of all of the teams which means that they do not concede many goals so it is expected that it will be the hardest to score against them.

5.2.6 Additional challenges

The process of training the model is applied after each round of fixtures and it updates the dataset with the new coefficients. However, there were three main challenges with producing this model:

- The beginning of season problem discussed in the Chapter 2.
- Each season three new teams are promoted to the league replacing the relegated sides.
- The GLM model takes one of the variables for the attacking and defensive strength as baseline variables.

The beginning of season problem

The way this problem was mitigated was by initially calculating the team coefficients from last season's games, so we can have some base parameters we can work with of the most recent performances from each of the teams in the league. Then as each round iterates the coefficient update with the most recent games played developing a rating system that will gradually adapt to the current form of the team.

Each season three new teams are promoted to the league replacing the relegated sides

This issue was resolved by assigning the coefficients of the attacking and defensive strengths of the three relegated teams to the newly promoted teams. Although there might be some potential differences between the promoted and the relegated teams that way we do not waste data from the beginning of the season and are able to apply it to the training of the model. As the season progresses round after round the coefficients will align to the exact performance of the promoted teams.

The GLM model takes one of the variables for the attacking and defensive strength as baseline variables

This issue arises due to the fact that GLM takes the alphabetically first team(in this case Arsenal) and omits it from the model parameters. The algorithm instead rewrites the attacking and defensive values for Arsenal in terms of the other 19 teams, so that the model does not have to fit 20 features, but 19 instead. Therefore, in order to identify the attacking and defensive strength of Arsenal we need to apply the following formula:

$$x = -1(k1 + k2 + ... + kn)$$

Where:

- k each of the attacking/defensive strength coefficients
- n the number of features
- x either the attacking or the defensive strength of the team

Since we are calculating the defensive and the attacking strengths separately we need to apply the formula twice, using the defensive coefficients to infer the defensive strength and the same for the attacking strength.

5.3 Training and Cross-Validation

Following our design structure we will now delve into more detail explaining the cross-validation process. It will cover the different machine-learning models we are going to be using. We will also talk about the different hyperparameters that will be tuned for each model, what they

are used for and how they can improve the performance of each of the models. Note that the results from the cross-validation evaluations for both the non-optimised and optimised models will be shown in detail in the Evaluation chapter of the report since in the implementation we want to only describe the process to achieving those results, so there will be no data points on display in this chapter.

5.3.1 Cross-Validation

Two cross-validators were created one for the Regression and one for the Classification models. The reason behind that decision was that different measurement metrics will be applied to evaluate the different types of models, so a separation was required. Inside each of the Cross Validators, those are the methods used:

- ”preprocess data” - This method is currently used for encoding the different teams from the ”HomeTeam” and ”AwayTeam” columns, so they can be converted to integer values and noted as categorical variables. For the process was used LabelEncoder which is a skilearn preprocessing tool.

- ”validate(num folds)” - This function carries out the whole validation process. It first splits the data into Time specific splits, so that the model is not being evaluated with football games that have happened before games that are part of the training data. The data is then split into the feature and target variables and the cross-validation method is applied. The method is flexible since it allows the user to set the number of folds it wants to split the data on. For each fold, the data is first scaled to reduce the chance of over-fitting and then trained with the scaled training data. Once trained, the model is tested on each evaluation metric and the subsequent results are stored in the subsequent model that it is evaluating.

5.3.2 Machine Learning models applied

Parameter Optimisation

Each of the machine learning models had a unique parameter optimisation method which was specifically used to identify the combination of hyperparameters that contained the best performance. For this purpose, we used a GridSearch technique that completes an exhaustive search for all the different combinations of hyperparameters possible by storing them in a matrix that can be iterated upon. In addition, to fully inspect the data we applied a Time

Series K-Fold cross-validation that uses the same number of folds as the normal cross-validator that is used for evaluating the final models. For the Regression machine learning models the optimisation was based on the mean absolute error and root mean squared error. For Classification models, accuracy and precision were used as evaluation metrics. The method returns the best combination of hyperparameters and the outcome it has achieved based on the metrics given.

5.4 Evaluation

After each of the models has been optimised and trained we implemented the benchmark models which will be compared to the chosen Regression and Classification models.

5.4.1 Dixon and Coles

```
def train(self,features_train,features_test):

    attack_strength, defense_strength, league_average_goals = self.calculate_attack_defence_strength(\
                                                                features_train)
    expected_home_goals,expected_away_goals = self.calculate_expected_goals(features_train,\
                                                attack_strength, defense_strength, league_average_goals)

    max_goals = 10
    scorelines = np.arange(0, max_goals + 1)
    home_probs = np.zeros((len(features_test), len(scorelines)))
    away_probs = np.zeros((len(features_test), len(scorelines)))

    for i in range(len(features_test)):
        home_probs[i, :] = poisson.pmf(scorelines, expected_home_goals[i])
        away_probs[i, :] = poisson.pmf(scorelines, expected_away_goals[i])

    expected_home_goals = np.sum(home_probs * scorelines, axis=1)
    expected_away_goals = np.sum(away_probs * scorelines, axis=1)

    predictions = np.zeros(len(features_test))
    predictions[expected_home_goals > expected_away_goals] = 1
    predictions[expected_home_goals < expected_away_goals] = -1

    return predictions
```

Figure 5.5: The method used to train the Dixon and Coles model

For the Dixon and Coles model, we had to first limit the dataset to only use the ”HomeTeam”, ”AwayTeam”, ”HomeTeamGoalsScored”, ”AwayTeamGoalsScored” and ”FullTime Result” data points upon which the target variable will again be the Full Time Result. As we can see from Figure 5.5 the model then uses those data points to calculate the attack and defence strengths of each team and based on that calculate the expected goals for each round of fixtures. During the training of the model the poisson distribution is applied to the expected

home and away team goals that were calculated earlier producing the probability of each of the three possible outcomes happening. The outcome with the highest probability is taken as the output from the model. The process is completed iteratively for all fixtures in the training set.

Chapter 6

Legal, Social, Ethical and Professional Issues

In this chapter, we will discuss potential ethical, legal, social and professional issues that could arise from the use of machine learning models in football predictions.

6.1 Ethical Issues

Ethical issues related to football predictions could arise from their use in betting and gambling activities. Having a machine learning model that has a high accuracy of predicting the correct outcome for football games could lead to potential unfair advantages in betting and gambling. On the other hand, the model could provide a platform for more responsible bets being placed which are not based on team biasness or luck. However, for the model to be legitimate it has to be trained on an unbiased dataset that could reflect the most accurate representation of how teams can perform in their upcoming fixtures so that it does not provide intentionally misleading information for personal gains.

During the creation of the project such measures were taken into account with all dataset resources used being publically available and no intentional bias practices have taken place when determining the contents of the dataset.

6.2 Professional Issues

The machine learning models developed/used, trained and tested have to be evaluated rigorously to adhere to best practices in machine learning, so that no misleading results are produced.

Applying Time Series K-fold cross-validation, evaluation of the optimisation of the hyperparameters using GridSearch and the use of appropriate performance metrics such as Accuracy and Mean Absolute Error during the process of training and testing of the models ensure that the models adhere to good machine learning practices.

6.3 British Computing Society Code of Conduct and Code of Good Practice

The (BSC)British Computing Society's[37] code of conduct sets out professional and ethical standards that are expected of its members, who are computing professionals in the UK and around the world. During the creation of this project me and my supervisor were fully aware of BSC's code of good practice and it was followed throughout the development process. Due to the small scale of the project, a small portion of the guidelines was to be taken into account and followed which resided around Professional Competence and Integrity and the Duty to Relevant Authority which in this case would be my supervisor Dr Keppens and King's College London.

Chapter 7

Evaluation

In this chapter, we will focus on the findings from the carried out evaluation experiments. Those experiments consist of comparing different models we have applied to our modified dataset. Then the chosen Regression and Classification models will be compared against some benchmark methods.

Since we are trying to pick the best possible model out of the ones we have applied to our dataset we are not focusing on specific accuracy metrics that we have to achieve, however comparing the best Classification and Regression model to other benchmarks will show us how using a combination of team form data and in-game metrics compares to other established models in the past.

7.1 Comparison of models without optimisation of the hyperparameters

In this section, we are going to show and explain the results gathered from both the regression and classification models without any hyperparameter optimisations being made. This will give us an indication as to how tuning specific hyperparameters improved the performance of our ML models.

7.1.1 Regression models

Model Name	mean_absolute_error	root_mean_squared_error
ANN	0.5889	**0.7092**
SVR	**0.5753**	0.7241
XG Boost	0.6134	0.793

Figure 7.1: Evaluation metrics for non-optimised Regression models

From figure 7.1 we can see that the mean absolute error for Support Vector Machines is the lowest with a 0.57 error rate whereas, for the root mean squared error, the results are quite similar with ANN and SVR having much lower error rates in comparison to XG Boost.

7.1.2 Classification models

Model Name	accuracy	precision
K Nearest Neighbour	0.5235	**0.5562**
Random Forest	0.5471	0.5429
Naive Bayes	**0.5676**	0.5449

Figure 7.2: Evaluation results for non-optimised Classification models

From figure 7.2 we can see that Naive Bayes has the highest accuracy out of all of the models at 0.57, but K's nearest Neighbour has the highest precision at 0.56. Random Forest sits in the middle with quite a similar accuracy and precision. Currently, Naive Bayes has the best overall parameters and would be the choice for the highest-performing model. We will see whether that is the case after we optimise the hyperparameters.

So, before the optimisation of hyper-parameters for each of the models, Random Forest and Support Vector Machines are the highest-performing ones. We will now look by optimising the hyper-parameters on those models how much if at all their performance improves.

7.2 Comparison of models with optimising their hyperparameters

7.2.1 Regression optimised models

Model Name	mean_absolute_error	root_mean_squared_error
ANN	0.5892	**0.7061**
SVR	**0.5868**	0.7147
XG Boost	0.589	0.7463

Figure 7.3: Evaluation results for optimised Regression models

From figure 7.3 we can see that there was an improvement in the Mean Absolute Error of ANN with the error rate going below 0.6.The same can be said for XG Boost from being into the low 0.6 moving to the upper boundary of 0.5. SVR did not make any significant improvements after being optimised by the hyperparameters. After the optimisation all three models have close accuracy results. XG Boost will not be included though because its RMSE is much higher than ANN and SVR. We will choose ANN for our Regression model because although both ANN and SVR have almost identical Mean Absolute Errors, ANN has a slightly lower root mean squared error with 0.706 in comparison to 0.715 for SVR.

7.2.2 Classification optimised models

Model Name	accuracy	precision
K Nearest Neighbour	**0.5735**	0.5464
Random Forest	0.5676	**0.5899**
Naive Bayes	0.5676	0.5449

Figure 7.4: Evaluation results for optimised Classification models

From figure 7.4 we can see that there was an improvement in the accuracy of K Nearest Neighbour from being into the low 0.5 moving to the upper boundary of 0.5 accuracies and overtaking Random Forest and Naive Bayes who did not make any significant improvements after being optimised by the hyperparameters. Although K's nearest neighbour has the highest accuracy we will pick Random Forest as our Classification model because the difference in accuracies of KNN and Random Forest is smaller than the difference between their precisions with Random Forest still with the highest precision.

7.3 Hyperparameter evaluation

Now we will talk about the results from cross-validating the different hyperparameters for each of the models and what combination of hyperparameters produced the most accurate results. A snapshot of the GridSearch applied to see what combination of hyperparameters produces the best outcome is visible in Appendix A.

7.3.1 K-nearest Neighbour

- k number or the number of nearest neighbours - we ran a subset of all the odd numbers between 12 and 31. Initially, we applied the formula $\sqrt{n}$ in which n is the number of features used to train the dataset since that was deemed a good starting point from previous studies and the k number was 19. However, we expanded the range by adding values before and after 19 to check for better accuracy. From the achieved results the k number which was chosen was 29 as the most optimal one for this dataset.

- weight - the "distance" parameter was chosen for our non-linear dataset having the different data points weighted. The performance of KNN also favoured the distance-weighted hyperparameter.

- "p" value also known as the power variable - Surprisingly the p-value that performed better was for the "manhattan" distance instead of the "Euclidean". Therefore p was set to 1.

7.3.2 Artificial Neural Network

- hidden layer sizes - Multiple variations were used. Having a single hidden layer with a high number of neurons (40+), having more than one hidden layer with a low number of neurons (10) or a compromise between the two extremes. In the end a hidden layer consisting of two layers each one with 20 neurons was the most optimal choice for this hyperparameter.

- solver - Out of the three different solvers "adam" was the most optimal solver.

- activation - Out of the three different activation functions tested on the model, the most optimal one was 'tanh'.

- max iter - We tested the model on a range between 100 and 500 iterations with the maximum number of iterations that the model was the most optimised being 300.

7.3.3 Random Forest

- n estimators - a range of 3 to 15 decision trees was tested with the most optimal number of trees being 11, so overall the model prefers a moderate number of decision trees.
- random state - tests were carried out in the range of 0 to 30 and the most optimal value for the random state was 0. This could well be because the data used is time-dependent and the outcomes of future games heavily depend on the outcome of past games and adding randomness in the order of games would decrease the performance of the model rather than improve it.

7.3.4 Supported Vector Regression

- kernel - Out of the two kernel functions applied to the model the ”rbf” function produced higher performance than ”linear” as expected due to the non-linearity of the data and the need for multiple dimensions to establish the vector distances between the data points and the hyperplane.
- C value - The model was tested with a range of penalties from 0.05 to 0.2 with the most optimal results coming from having a higher penalty of 0.2 which meant that a narrower margin of tolerance around the true function produced better results.
- gamma - the models were tested on low gamma values between 0.1 and 1 and a scaled option. The outcome was that a value of 0.1 was the most suitable for the gamma value. This means that the kernel is wider and the decision boundary is less sensitive to small variations in the training data.

7.3.5 Naive Bayes

- alpha - A range of values was tested from 0 to 10.0 as the alpha value and the most optimal results were achieved when the alpha value was zero which suggests that most if not all of the features are observed in the different classes therefore there is no need for additional support for the probability estimate for all the features.

7.3.6 XG Boost

- learning rate - Since the default value for the learning rate is estimated to be around 0.3 and is recommended that the value be lower than 0.3 when trying to optimise it, we

tested it with values in the region 0.05 - 0.2 and the most optimal outcome was 0.2. This means that having a more aggressive boosting strategy could lead to better performance.

- number of estimators - The model was tested on a range of estimators between 50 and 200 with the most optimal results coming from a lower number of estimators or decision trees being introduced. Therefore, this means that although having a high number of trees should improve the performance in this case it leads to overfitting.

- maximum depth - trees were tested with depths of 3 4 and 5. The most optimal depth was 3 which suggests that the model leads to overfitting if the depth is quite high, so keeping it at a low level improves the overall performance.

7.3.7 Dixon and Coles

- maximum number of goals - In comparison to the rest of the models which produced their highest performance metrics using a maximum number of goals per game being 5, for the Dixon and Coles model that was not the case. The most optimal results came when the maximum number of goals was set to 10 instead. This could suggest that Dixon and Coles takes more into account any anomaly results of games with high to very high scoring games.

7.4 Comparison to benchmark models

Now we are going to compare the different performance metrics for our chosen Regression and Classification models with different benchmark methods.

7.4.1 Dixon and Coles

Model Name	accuracy	precision
K Nearest Neighbour	**0.5735**	0.5464
Random Forest	0.5676	**0.5899**
Naive Bayes	0.5676	0.5449
Dixon and Coles	0.2735	0.4233

Figure 7.5: Evaluation results for optimised Classification models

From figure 7.5 we can see that the Dixon and Coles model has a significantly lower accuracy of 0.274 and a precision of 0.423. The other models are a significant improvement on it with the highest accuracy of 0.575 coming from K-nearest Neighbour and the highest precision of 0.590 coming from Random Forest.

Model Name	mean_absolute_error	root_mean_squared_error
ANN	0.5898	**0.7091**
SVR	**0.5868**	0.7147
XG Boost	0.589	0.7463
Dixon and Coles	0.8853	1.0939

Figure 7.6: Results for optimised Regression models in comparison to Dixon and Coles

From figure 7.6 we can see that the Dixon and Coles model has a mean absolute error of 0.885 and a root mean squared error of 1.094. This shows that the regression models produce significantly better results than the Dixon and Coles benchmark model with the lowest root mean squared being 0.709 coming from Artificial Neural Networks and the lowest mean absolute error of 0.587 coming from Support Vector Regression.

From those results we can conclude that depending only on goals scored and past performance does not yield as accurate results as including domain-specific in-game statistics such as shots were taken, yellow and red cards etc in addition to the team form statistics. Dixon and Coles is a model which relies solely on goals scored from both sides and determines how to calculate its attacking and defensive strengths based on that.

7.4.2 Naive benchmarks

There are two naive benchmarks that we are going to use to compare the model's performance to our model's performance.

The first one is randomness, which indicates that we chose one of the three outcomes at random. This would give us an accuracy and precision of 0.33. That would mean a model that is lower than 0.33 is going to have a smaller chance than a random choice. In our case, the only model that has an accuracy below 0.33 is Dixon and Coles which in general is surprisingly low, however, it can be explained due to the low number of games in the dataset because one of the limitations of the model is the need for a sufficient number of past results in order for it to calculate the team's attacking and defensive strengths as well as home advantage accurately. The models we have evaluated assure us that the random factor is taken into account and their outcomes show much more accurate predictions.

The second naive benchmark we are going to use is the accuracy and precision of a model in which all the game outcomes are predicted to be wins for the home team. I chose this to be the outcome for all games because this is usually the most common outcome for football games due to the home advantage factor. The accuracy and precision of this outcome were 0.428 which is higher than the accuracy of the Dixon and Coles model and Random outcome

naive benchmark, however, the precision of the Dixon and Coles model is slightly higher at 0.430, but, those metrics are significantly lower than the outcomes that were gathered from the Classification models that were trained on the model.

7.5 Strengths and Weaknesses of the project

Based on our aims and objectives at the beginning of the project and the carried out experiments, these are the established strengths and weaknesses in our project:

7.5.1 Strengths

- We were able to generate a set of data features which represented the attacking and defensive strengths of each team as well as an overall home team advantage coefficient which was recalculated on a football round basis.
- We were able to create a machine learning pipeline that is maintainable and extendable for future models to be added and compared as well as any additional evaluation techniques to be added.
- We were able to explore a wide range of Machine Learning models for both Regression and Classification methods as well as optimise them based on a number of specific hyper-parameters for each.

7.5.2 Weaknesses

- We limited the model to only use data from the past 2 seasons of the Premier League. Including more domestic leagues and teams could make the model more robust and give a better generalisation of the data whilst keeping the data used on the model as recent as possible.
- There wasn't a significant improvement in the accuracy of the Classification models as well as in the error rate of the Regression models. This could be due to the hyper-parameter tests where we were able to identify only the local optimum, but maybe not the global optimum for those modules and it would require more computational power to achieve that.
- The model makes a couple of assumptions such as associating the attacking and defensive strengths of the promoted teams with the ratings of the teams that were relegated the

season before at the state of the final round, this fixes the issue of having ratings that are extremely high due to their results in the lower division last season, but it also does not fully reflect the true performance of those teams at the begging of the season.

Chapter 8

Conclusion and Future Work

8.1 Summary

The main aim of this project was to build a football game prediction model that is based on various machine-learning techniques for both Classification and Regression models. It involved the collection of both in-game and season-form data that concatenated together would give us a valid dataset to train the models upon.

We managed to achieve that by identifying an in-game dataset that provided us with the most influential game statistics such as possession, and shots on target for each fixture. We then also used the game results from the previous 2 seasons in the Premier League to calculate the attacking and defensive strengths of each team in the league using the Poisson Regression model in order to give us a dataset that is multidimensional and focuses on both the in-game aspect but also on the home and away team's form going into that game.

We achieved another one of our main objectives by creating an easy-to-use and maintainable machine-learning pipeline that can be extended to include additional machine-learning models to be evaluated against each other using appropriate time series cross-validation techniques and hyperparameter optimisation for better results.

We trained our data on models that were fundamentally different in approach even though they resided in the Classification or Regression category. For example, K Nearest Neighbour uses the Lazy Learning approach whereas Random Forest uses Decision Trees to establish the different classifiers (as explained in detail in the Background and Research chapter). This allowed us to compare results from a variety of techniques and establish the one which would most suit our

dataset. Another criterion upon which the models were chosen was based on past research, so the set of most effective models was chosen for evaluation.

8.2 Project Challenges

8.2.1 Finding appropriate data for the evaluation

A key challenge that was encountered early on was identifying the right dataset both in terms of the time frame and specific data to be used. Initially, much of the publicly available data was from past seasons which were up to the 2016/17 season which in a competitive sport like football was not going to give us very appropriate results. However, after deeper research, the dataset identified in FootballData was the most up-to-date containing the 2020/2021 and 2021/2022 seasons and covered the other main criteria which were the all-important in-game statistics that were going to allow our model to establish correlations between important actions that happen during the game and how they affect the outcome. Further ideas of player-related data came into mind as well such as calculating the player form and their injury record, but this was not very publicly available data and did not contain all the players in the Premier League for the seasons that the model was trained upon.

8.2.2 Parameter choices

As an extension to the previous challenge, this one was related to extending the currently available dataset to not only rely on in-game statistics but also form related data that is not concerned with the in-game dynamics itself. A couple of suggestions were thought of starting with the expected number of shots and player form, but since there wasn't readily available data for those two statistics, we decided to stick to team-specific performance metrics which included calculating the attacking and defensive strengths of teams as well as the overall league home advantage.

8.2.3 Identifying an efficient hyperparameter optimisation method

Initially, the only method of robustness that we had implemented when evaluating the performance of the models was through time series k-fold cross-validation, however, we took it a step further and included hyperparameter tuning which runs using a GridSearch method. Although models like SVM and Random Forest did not significantly improve in performance there were

others that did, for example, K-Nearest neighbour which by increasing its k number to 29 from 19 managed to increase its accuracy by 3-4%.

8.3 Potential future work on the project

There are a number of future directions that this project can undertake. We are going to focus on the ones which with more time and resources could be achieved.

8.3.1 Including a larger set of football domain-knowledge features into the dataset

In the current version of the dataset used we have incorporated domain-specific data such as shots on target and yellow cards as an example, however, taking this a step further would mean adding data that is not easily accessible online. Data of such sorts could be player statistics, ratings after each game, morale and injury records. In many teams, specific individuals perform better than others and help the team propel forward. An interesting development of the model would be in a similar fashion to the team form to calculate and predict player form going into a certain game. Each player can have a dynamic rating which could dictate how they might perform against the opposition hence that could affect the overall performance of the team and those factors could be incorporated into the model. A good website where such statistics are present but not freely available in a sanitised format is WhoScored.com[41].

8.3.2 Implementing a Football score application predictor

In the modern world, there are many sports mobile and web applications such as Livescore[10] that present the results from sports games in an easy and accessible way as well as the post-match statistics. Some of them even include betting odds and outcome percentages from the game itself. Those results however are always skewed to favour the betting companies and allow them to profit from the overall outcome of all placed bets. A study from Kaplan[23] identified that there is a specific bias in bookmakers where they intentionally lower the winning odds or make more favourable odds on the opposite outcome in order to cover any potential profit losses. Therefore the accuracy of those models purely looking from a results perspective might not be the most accurate measure.

Therefore, creating or integrating the current machine-learning model into an existing application could be more beneficial to the normal user since the algorithm will not take into

account betting biases to skew the shown probabilities for the game hence giving them a higher chance of having a successful bet if he decides to make such decision.

8.3.3 Optimisation of the feature parameters

In this project we focused on optimising the hyperparameters for each of the evaluated models, however, those optimisations can be taken a step further by optimising the dataset parameters based on different in-game events. For example, the weight of the attacking and defensive strength of both teams if one of the teams has one less player on the pitch. That should, in theory, increase the chances of the opposition winning and the scales for the other features need to be adjusted to favour that. In order to make such adjustments the OpenMOLE[13] software can be used. It uses parallel processing to handle multiple parameter tests. The tests are produced based on ranges that were initially defined by the user for each of the parameters and then use a chosen from the user model to run a multitude of randomly generated tests from the software. It then progressively discards tests that produce undesirable results and sustains the ones which do which in turn optimises the parameter values. Having more optimised parameters should add another layer of sanitation to the dataset and increase the accuracy of the models.

References

[1] Confusion matrix: How to use it and interpret results. https://www.v7labs.com/blog/confusion-matrix-guide.

[2] Diagram for gradient boosting. https://medium.com/swlh/gradient-boosting-trees-for-classification-a-beginners-guide-596b594a14ea.

[3] Diagram for naive bayes theorem. https://kdagiit.medium.com/naive-bayes-algorithm-4b8b990c7319.

[4] Diagram for poisson regression. https://stats.oarc.ucla.edu/r/dae/poisson-regression/.

[5] Diagram for random forests. https://ai-pool.com/a/s/random-forests-understanding.

[6] Diagram for support vector machine algorithm. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.

[7] Diagram for time series cross validation. https://www.mathworks.com/discovery/cross-validation.html.

[8] Fixture download. https://fixturedownload.com/.

[9] Football data website. https://football-data.co.uk/englandm.php/.

[10] Livescore - live football games. https://www.livescore.com/en/.

[11] Matplotlib library. https://matplotlib.org/.

[12] Numpy library. https://numpy.org/.

[13] Openmole - model exploration software. https://openmole.org/.

[14] Pandas libarary. https://pandas.pydata.org/.

[15] Python. https://www.python.org/.

[16] Scipy library. https://scipy.org/.

[17] Statsmodels library. https://www.statsmodels.org/stable/index.html.

[18] Daniel Berrar. Cross-validation. https://www.researchgate.net/profile/Daniel-Berrar/publication/324701535_Cross-Validation/links/5cb4209c92851c8d22ec4349/Cross-Validation.pdf.

[19] Lucas Cinelli, Gabriel Chaves, and Markus Lima. Vessel classification through convolutional neural networks using passive sonar spectrogram images, 05 2018. Diagram for Neural Networks.

[20] Werner Dubitzky Daniel Berrar, Philippe Lopes. Incorporating domain knowledge in machine learning for soccer outcome prediction. https://link.springer.com/article/10.1007/s10994-018-5747-8.

[21] Mohammed A. M.Sadeeq Rizgar R. Zebari Gheyath Mustafa Zebari, Subhi R. M. Zeebaree. Predicting football outcomes by using poisson model: Applied to spanish primera división. researchgate.net/profile/Subhi-Zeebaree/publication/357359461_Predicting_Football_Outcomes_by_Using_Poisson_Model_Applied_to_Spanish_Primera_Division/links/61caf740e669ee0f5c6c0204/Predicting-Football-Outcomes-by-Using-Poisson-Model-Applied-to-Spanish-Primera-Division.pdf.

[22] GitHub. Github repository. https://github.com/StefTih/FootballPredictionModel, 2023.

[23] Michael Kaplan. The secret betting strategy that beats online bookmakers. https://arxiv.org/abs/1710.02824, 2017.

[24] W. Koevoets. *Goals per match*. 2019. Chapter 5, available at www.goals-per-match.net.

[25] Anders Krogh. What are artificial neural networks? *Nature biotechnology*, 26(2):195–197, 2008.

[26] Ulrich Lichtenthaler. Liverpool fc: Extending data analytics and ai with neuroscience. https://tinyurl.com/4yj8darv, 2022.

[27] Carlos Lago-Peñas Martí Casals Llorenç Badiella, Pedro Puig. Influence of red and yellow cards on team performance in elite soccer. 2022.

[28] James McNicholas. What buying statdna means for arsenal and their moneyball approach to football. https://bleacherreport.com/articles/2269196-what-buying-statdna-means-for-arsenal-and-their-moneyball-approach-to-football, 2014.

[29] M.J.Maher. Modelling association football scores. http://www.90minut.pl/misc/maher.pdf, 1982.

[30] Kevin P Murphy. Machine learning: a probabilistic perspective. https://machinelearningmastery.com/bayes-theorem-for-machine-learning/, 2012.

[31] Chris Richter, Martin O'Reilly, and Eamonn Delahunt. Machine learning in sports science: challenges and opportunities. *Sports Biomechanics*, pages 1–7, 2021.

[32] Teo Susnjak Rory Bunker. The application of machine learning techniques for predicting match results in team sport: A review. https://www.jair.org/index.php/jair/article/download/13509/26786/30289, 2022.

[33] Pradip Samuel. Cross-validation in time series model. https://medium.com/@pradip.samuel/cross-validation-in-time-series-model-b07fbba65db7, 2020.

[34] David Sheehan. Predicting football results with statistical modelling: Dixon-coles and time-weighting. https://dashee87.github.io/football/python/predicting-football-results-with-statistical-modelling-dixon-coles-and-time-weighting.

[35] Soumya Shrivastava. Cross validation in time series. https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4, 2018.

[36] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14:199–222, 2004.

[37] British Computing Society. https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/#:~:text=avoid\%20injuring\%20others\%2C\%20their\%20property,of\%20bribery\%20or\%20unethical\%20inducement.

[38] Penn State. Introduction to glms. https://online.stat.psu.edu/stat504/lesson/6/6.1.

[39] Tommy Thompson. Introduction to artificial inteligence. https://keats.kcl.ac.uk/course/view.php?id=93530, 2022.

[40] Shaun Turney. Poisson distributions definition, formula examples. https://www.scribbr.com/statistics/poisson-distribution, 2022.

[41] WhoScored. WhoScored.com - Football Statistics. https://www.whoscored.com.

[42] Wikipedia. Statistical association football predictions. https://en.wikipedia.org/wiki/Statistical_association_football_predictions, 2022.

Appendix A

Additional information

A.1 Example of the Dataset

1	HomeTear	AwayTean	FTHG	FTAG	FTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR	HomeTear	HomeTear	AwayTean	AwayTean	HomeTear	Round
2	2	0	2	0	1	8	22	3	4	12	8	2	5	0	0	0	0	-0.98904	-0.98904	0.087895	0.006661	0.007813	1
3	12	8	5	1	1	16	10	8	3	11	9	5	4	1	2	0	0	0.28911	0.28911	0.135693	0.135693	0.007813	1
4	4	3	1	2	-1	14	14	3	8	10	7	7	6	2	1	0	0	-0.49558	-0.49558	-0.31198	-0.31198	0.007813	1
5	5	6	3	0	1	13	4	6	1	15	11	5	2	0	0	0	0	0.050158	0.050158	-0.26671	-0.26671	0.007813	1
6	7	15	3	1	1	14	6	6	3	13	15	6	8	2	0	0	0	-0.14834	-0.14834	-0.12761	-0.12761	0.007813	1
7	9	19	1	0	1	9	17	5	3	6	10	5	4	1	2	0	0	0.224206	0.224206	-0.41145	-0.41145	0.007813	1
8	17	1	3	2	1	13	11	7	2	18	13	2	4	3	1	0	0	-0.41499	-0.41499	0.007191	0.007191	0.007813	1
9	14	10	0	3	-1	14	19	3	8	4	14	3	11	1	1	0	0	-0.69861	-0.69861	0.215832	0.215832	0.007813	1
10	13	18	2	4	-1	17	8	3	9	4	3	7	6	1	0	0	0	-0.15545	-0.15545	0.128376	0.128376	0.007813	1
11	16	11	1	0	1	13	18	3	4	11	8	3	11	2	1	0	0	0.218963	0.218963	0.405324	0.405324	0.007813	1
12	18	9	4	1	1	13	5	7	1	8	8	10	0	0	1	0	1	0.180373	0.180373	0.234635	0.234635	0.022961	2
13	0	5	0	2	-1	6	22	3	5	10	4	9	8	3	0	0	0	0.110833	0.010511	0.08047	0.08047	0.022961	2
14	19	16	0	1	-1	25	8	6	6	9	7	5	4	1	4	0	0	-0.41813	-0.41813	0.22466	0.22466	0.022961	2
15	15	12	1	1	0	8	15	3	4	12	10	7	7	2	3	0	0	-0.11423	-0.11423	0.286692	0.286692	0.022961	2
16	3	17	2	0	1	13	10	3	1	11	6	7	2	4	0	0	0	-0.28499	-0.28499	-0.42041	-0.42041	0.022961	2
17	1	13	2	0	1	10	9	2	1	8	18	3	4	3	4	0	0	0.027492	0.027492	-0.16102	-0.16102	0.022961	2
18	8	7	2	2	0	17	17	4	8	6	13	8	5	2	4	0	0	0.160452	0.160452	-0.1167	-0.1167	0.022961	2
19	6	2	0	0	0	7	14	2	3	12	9	3	5	3	1	0	0	-0.28134	-0.28134	-1.00581	-1.00581	0.022961	2
20	10	4	2	0	1	27	9	9	3	6	12	8	4	0	0	0	0	0.233768	0.233768	-0.49953	-0.49953	0.022961	2
21	11	14	5	0	1	16	1	4	0	13	7	6	1	1	0	0	0	0.453252	0.453252	-0.6953	-0.6953	0.022961	2

Figure A.1: Example of the dataset which includes both in-game statistics and team form coeficients

A.2 Optimisation outcome results

```
################################################
Model K Nearest Neighbour:
Parameters: {'metric': 'manhattan', 'n_neighbors': 29}
Score: 0.5529411764705883
################################################
################################################
Model ANN:
Parameters: {'activation': 'relu', 'hidden_layer_sizes': (10, 10), 'max_iter': 500, 'solver': 'adam'}
Score: -0.5882970231043831
################################################
################################################
Model SVR:
Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
Score: -0.5136387756944607
################################################
################################################
Model Random Forest:
Parameters: {'n_estimators': 9, 'random_state': 0}
Score: 0.5588235294117647
################################################
################################################
Model Naive Bayes:
Parameters: {'alpha': 0}
Score: 0.538235294117647
################################################
################################################
Model XG Boost:
Parameters: {'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 50}
Score: -0.5887472425888786
################################################
```

Figure A.2: Example of some of the hyperparameter tuning results achieved