MACHINE LEARNING ALGORITHMS FOR SPORTS' RESULTS PREDICTION

ANDRES IGEA

This dissertation was submitted in part fulfilment of requirements for the degree of MSc Software Development

DEPT. OF COMPUTER AND INFORMATION SCIENCES
UNIVERSITY OF STRATHCLYDE

AUGUST 2019

Declaration

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself. Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research. – N/A

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive. (please tick) Yes [✔] No [ ]

I declare that the word count for this dissertation (excluding title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is 21,348 words.

I confirm that I wish this to be assessed as a Type 1 2 3 4 (5) Dissertation (please circle)

Signature:  Date: 25th of August 2019

Abstract

The forecasting of football matches' results has proved to be a difficult task for which many attempts have been made. The use of traditional statistical models and their results produced until now have been relatively poor. However, better predictive accuracies have been claimed by some authors using machine learning models built with informative features derived from previous matches.

Is it possible to predict the outcome of football-matches using machine learning just like some other authors claim to do? Is it possible to outperform those models?

An ambitious aim of this dissertation is to produce a model able to surpass the accuracies achieved by the sports betting operators. The positivism research paradigm is adopted in this study. There are a myriad of factors influencing the results of football games and this project has defined some of these important features and assessed them. It has been found that using the author's models, football players' ratings obtained from the EA Sports video game FIFA (Borjigin, 2019) show a higher forecasting ability than other more 'sport related' features derived from previous games such as the number of shots on target, corners, yellow cards, goals, etc.

Using pipelines that combine pre-processing of data, engineering and selection of features, as well as the selection of the best hyper-parameters for several machine learning algorithms, the model designed has been able to outperform the book-makers during the 2011/2012 and 2012/2013 seasons of the German Bundesliga. For the remaining seasons (2010/2011, 2013/2014, 2014/2015), the project's model was able to obtain an equal performance in the 2013/2014 season and a slightly inferior performance in the 2010/2011 and 2014/2015 to that of the sports betting operators.

The project also discusses that different predictabilities apply to different football leagues and seasons, and that for the season 2015/2016 of the German Bundesliga the author obtains a prediction accuracy of 51%. This performance is slightly lower than the five betting companies studied.

Acknowledgements

The author would like to express his gratitude to his supervisor Dr. Kostas Liaskos for his unconditional guidance, advice and support throughout the preparation of this project until its completion.

I gratefully acknowledge the friendly support that I received from all the staff members in the CIS support team as well as all the lecturers whom I was fortunate enough to learn from.

Special thanks to my family for their patience and encouragement during the completion of this work.

Table of Contents

List of illustrations

1 Introduction

1.1 The background of this project and the problem to solve

Several factors, such as the growing media coverage have recently increased the popularity of numerous sports. In particular, soccer (also called association football, and sometimes in this project simply football) is now followed by around half of the global population (B. Sawe, World Atlas, 2019).

Nowadays, the most popular sport in the world is soccer. The area of influence of this sport is global and 4 billion people follow the sport to some degree (B. Sawe, World Atlas, 2019). The 'Beautiful Game' as it was once described by the Brazilian footballer *Pelé*, is the sport most watched on television, with the most expensive television rights, the highest paid sportsmen and competitions, and the most popular teams on the social media (Highlights et al., 2019).

Several reasons may be behind the forecast of football matches' results, curiosity and economic reasons are often the motivating forces that impulse soccer fans to predict the matches' results. The soccer enthusiasts sometimes actuate propelled by the pursuit of economic benefit, betting either online or in the high street betting shop. At other times, the attempt to predict the results of the games simply correspond to an attempt to obtain information that allows for an interesting conversation with some friends or work peers after the weekend.

Machine learning (ML) algorithms may be used to forecast the results of soccer games. The ML algorithm is trained using a data set consisting of a significant number of data points (matches). For each match this training data includes features relevant to the games such as the players, the number of corners, penalties, yellow and red cards, etc. of the match. The results of each match are also part of this training data. Prediction models may be built using the training data. The models produced may be assessed comparing the results predicted by the model for a testing set of matches with the actual results of those matches.

The first successful attempts to predict the results of football matches by quantitative methods were made towards the middle of the last century. Moroney (1956) describes a statistic process to forecast soccer's results. As at that time, limited computer power was available, solving the problem was a very tedious and time-consuming task.

Using Data Science techniques to forecast results of soccer games is becoming increasingly popular. Football clubs now employ data analysts trying to gain a competitive edge. According to Ian Graham, the Liverpool's Director of Research, each football match comprehends thousands of players' actions, but the research department can only assess the downloadable ones included in the football stat sheets and forms. Graham I. adds that the data the research department manage is very limited, and that he is working in improving the mathematical model of the games using video tracking. As more and more features are added to the model, improved models of a soccer match can be created. More complete models will produce predictions of higher accuracy (Nytimes.com, 2019).

At this stage, in order to avoid a possible disappointment of the reader, it is important to point out that the prediction of results of specific matches constitutes a difficult challenge. This is due to two main reasons, one is the random component that is introduced in each game by the number of goals that each team scores, and as a consequence, the final result of the match. The other reason is the fact that three different results are possible (home win, draw and away win) with the prediction of draw being difficult issue. There is an apparent dichotomy. It is not hard to predict which teams will be successful by the end of the league but developing a model able to predict the result of a specific game proves to be a complex task. The accuracy that may be expected from the models developed for soccer will not reach the percentages of correct results obtained in other sports where the only possible outcomes are win or lose (Dixon and Coles, p.267, 1997).

Aoki, Assuncao and Vaz de Melo (2017) emphasize on the random component present in the results of the following sports: basketball, handball, soccer and volleyball. The authors suggest a probabilistic graphical model able to disentangle the relative components of the skills of the

teams and randomness during the match. Their results show that out of the four analyzed sports, soccer is the one with a greater component of 'luck' in the final results. They justify this fact due to the relatively small number of points scored during the soccer games (average per match 2.62). They also demonstrate that for the leagues from 2007-2008 to 2015-2016, if the teams Real Madrid and Barcelona were removed from the Spanish *Primera División*, the predictable results for the final positions of the teams in the *La Liga* ranking would be totally random.

1.2 The problem formulation

This project aims to develop and assess a Machine Learning model able to predict results of football matches.

The project also provides answers to the following research questions:

- Which features of the data available have more predicting ability?
- Can data collected from EA Sports (regarding player ratings) outperform real-world historical data in order to predict the outcome of football-matches?
- Which machine learning algorithms produce more accurate predictions? Which parameters should be used when using those algorithms to predict soccer match results?
- Do the models developed improve the predicting capacity that existing models claim to obtain?

1.3 The objectives of the project

This project was carefully conducted with all the knowledge acquired throughout the year and tries to adhere to the best software engineering practices. As per the proposal for this piece of research, the following objectives were set using the software development prioritization technique called MOSCOW. The objectives of the project were divided according to their obligatoriness: must haves, should haves, could haves and won't haves (Igea, 2019, p.5).

- **Must have:** Machine learning models able to predict the results of sport games should be built. The models' accuracies should be assessed. Answer the research questions shown in the problem formulation and evaluate them.
- **Should have:** a web page in which registered users can take advice about which bets may have a greater return of investment (bets for which greater discrepancies are found between predictions of owned model and betting agencies forecasts).
- **Could have:** A web page in which registered users can search information about betting odds and places (with links to them) where bets can be placed for their sports of interest.

During the unfolding of the project, using a variant of the Scrum Agile Methodology, it was found that the objectives shown above did not constitute a final contract between the author and the supervisor.

De facto, from the original set of objectives, due to the time constraints imposed on this dissertation and the setting of overly optimistic and ambitious objectives, only the first bullet point, that is the 'must have' is addressed in the text that follows.

1.4 The structure of this report

The project has been structured according to the following chapters:

In this first chapter, the 'Introduction' to the project is presented. In the second chapter, a review of the research about football results predictability prior to the commencement of this project, 'Literature Review', may be found. The chapter describes the previous related research works and contains an overview about the previous work on engineered features for soccer results prediction. Chapter three contains a brief overview of the 'theoretical background of the main machine learning algorithms', their advantages and disadvantages. The 'Methodology' followed during the project is presented in chapter four. In chapter five of the project, the 'Presentation and evaluation of results' may be found. Lastly, in chapter six, the project's conclusions are presented, further work and enhancement of this project is proposed.

1.5 Summary of the chapter

This first chapter of the project explains the initial motivation, background and problem to be solved. A description of the objectives and structure of the project is also provided.

The following chapter reviews the research about prediction of football matches' results prior to the commencement of this project.

2 Literature review

The purpose of this section is to review the research in the prediction of football matches' results prior to this project. The section is divided into two parts, the first will briefly describe the main documents of related work and the second has an emphasis in the previous work on engineered features applicable to machine learning for soccer results prediction.

2.1 Previous work: related work documents

The eldest reference found about modelling the results of a soccer game during this project's literature review, (Moroney, 1956), predicts the number of scores in a game using as input the number of goals of previous matches. The distribution of scores is initially fitted using a Poisson distribution, but the author sustains that improved results can be obtained using a negative binomial distribution.

Reep, Pollard and Benjamin (1971) use a negative binomial distribution to fit a model with the scores of the soccer game, but conclude that the game is dominated by a random component due to the significant amount of noise included in the data, and as a result, no successful predictions can be expected.

In spite of the above, Hill (1974) was able to demonstrate significant correlation when performing simple comparison tests for predictions of final football league placings. Hill's findings encouraged other researches to investigate the challenging field of predicting soccer results through statistical techniques.

An important step forward was taken by Maher (1982). The author used Poisson distributions to model independently home and away teams' scores from attack and defense parameters of the teams that were based on the soccer teams' performance in previous matches. The approach was able to provide estimates of maximum likelihood.

Dixon and Coles (1997) describe a parametric model able to produce maximum likelihood estimates for the English cup and league in the period 1992 to 1995. A Poisson regression model is enhanced by the introduction of time-dependent teams' performances and parameters optimization. Authors claim the model can be used to formulate a betting strategy in which they obtain positive return.

Rue and Salvesen (2000) propose a model to predict the results of the English Premier League and Division 1 of the season 1997-1998. The authors use the Markov Chain Monte Carlo technique to produce an iterative simulation that estimates the time dependent features of all the teams in the league simultaneously. Authors sustain their model suggested bets that produced significant pay-offs.

Joseph, Fenton and Neil (2006) compared the accuracy of the forecasts of a Bayesian net with the accuracies provided by the following Machine Learning techniques: K-nearest neighbour, decision trees (MC4), Naïve Bayesian and Data Driven Bayesian learners. According to the authors, Bayesian nets provided excellent accuracies (59.21%) predicting the results of the matches played by the Tottenham Hotspur Football Club in the period 1995-1997.

Another author that claims can beat the bookmakers' odds is Buursma (2011). The author describes in detail the betting strategies used and the performance of the different classifiers. Different football related features are used. This paper provides details of the list of features used in the author's investigation and shows the performances of the models he builds when using different selections of features.

The model built by Constantinou, Fenton and Neil (2012) generated predictions for the season 2010-2011 using training data from 1993 to 2010. The model uses four engineered features called strength, form, psychology and fatigue for both the home and the away teams. A deeper explanation of those four engineered features is provided in the next section. Using a Bayesian network model, and the four time-dependent engineered features (which include objective and subjective information) named above, the authors demonstrate that the subjective information

can improve the accuracy of the results' prediction and that their model generates profit via longshot bets.

Ulmer and Fernandez (2013) predict soccer results in the English Premier League. The authors trained several machine learning algorithms with the data of the seasons from 2002-2003 to 2011-2012. They predicted the results of the matches played during the seasons 2012-2013 and 2013-2014. The best accuracy (0.50) they obtained was using the Support Vector Machine model with a Gaussian kernel and parameters optimized using Grid Search. The paper explains how the authors solved several issues related to the use of a feature they defined to assess the 'streakiness' of a team.

Yezus (2014) uses the data of English Premier League and produces predictions using four different machine learning algorithms. The author claims that using a Random Forest model she was able to obtain an accuracy of 0.634. The author suggests several engineered features that will be analysed in the following section.

The problem of soccer match results prediction is approached by Shin and Gasparyan (2014) in an innovative manner. The paper of the authors used as features what they call 'virtual data' (33 features per player) collected from the video game FIFA 2015. The paper predicts the results of the football matches of the Spanish 'Primera Liga' using the 'virtual data' collected from the Sofifa website (Borjigin, 2019). Using several machine learning algorithms, the results found using as features the 'virtual data' are compared to the results found using as features 'real data' (24 parameters per match such as red cards, penalties, goals, corners, etc.). The three-class classification problem of predicting the result of a match (home win, draw, away win) is redefined as three bi-class classification problems. The authors claim they obtained accuracies of around 0.75 both when they used the 'virtual' and the 'real' data.

Tavakol, Zafartavanaelmi and Brefeld (2016) forecast the results of Euro 2016 tournament using a linear model that predicts probabilities of the three possible outcomes (win, lose or draw). Authors use as part of their data set the information that the FIFA provides about the players' ratings.

Razali et al. (2017) suggest a Bayesian Network to classify results in the categories home win, draw and away win. Outcomes of the matches of the English Premiere League from 2010 to 2013 are forecasted. Authors claim a predicting accuracy of 75.09%. Even though Razali et al. provide a list called 'Main Factors in Football Match Prediction' they do not inform about which engineered features were used for prediction if any. It seems possible that the surprisingly high accuracy of the model was perhaps due to the fact that the model might use some data of a specific match to predict the result of this same match. Another possible reason for the high accuracy may be that the authors predict results for each of the three seasons studied independently. This makes the teams' features to be more homogeneous within a season than they would be if the data of three seasons was used simultaneously.

Klyuchka et al. (2017) predicted the result of the match of the English Premier League between the Manchester United and the Tottenham Hotspur celebrated the 28th of October 2017. Three different methods were used: a method based on the Poisson distribution, a method based on the weighted sum of indicators and a method based on forecasting rules. Authors claim that they obtained accuracies of 83.5%, 77.6% and 70.4% respectively.

Dubitzky et al. (2018) explore the limits of the predictability using advanced machine learning. Their paper provides the results found during the 2017 Soccer Prediction Challenge. The rules of the challenge require to use as data the Open International Soccer Database. The method the authors use to assess the predictions is the rank probability score (RPS). Authors provide in this paper a very detailed explanation of the RPS concept and how it is calculated.

Wunderlich and Memmert (2018) build a model using data from English and foreign leagues of the seasons starting from 2007-2008 to 2016-2017. The model is based on the ELO rating system and betting information publicly available. According to the authors, this new ELO based model shows improved accuracy than the classic ELO rating models. The paper also provides a detailed explanation of the process of parameters' calibration required by the ELO models.

2.2 Previous work: engineered features for football results prediction

2.2.1 Strength, form, psychology and fatigue

These engineered features were suggested by Constantinou, Fenton and Neil (2012). The authors' model uses four engineered features called strength, form, psychology and fatigue for both the home and the away teams. The first feature (strength), due to its significant objective content is called objective component, whereas the last three features (form, psychology and fatigue) are called subjective components.

- The team strength is calculated using previous information (points obtained in the last five seasons, weighted higher for closer seasons) and current information (total points during the current season also weighted higher for closer matches). It is optional to also include a component of subjective information (the strength rating provided by an expert).
- The form of a team measures the actual performance of the team versus its expected performance during the last five matches. Closer matches are weighted higher. Higher values of form mean the team is performing better than expected and lower values of form mean that the team is performing worse than expected.
- The psychology feature of a team includes subjective concepts such as their team spirit, motivation, head to head biases and managerial issues.
- The fatigue feature of a team considers concepts such as the number of days from the previous match, how tough that match was, how many of the players that play a specific match rested in the previous match, etc.

2.2.2 Players' overall ratings and potential ratings

Tavakol, Zafartavanaelmi and Brefeld (2016) use the player Stats Database by FIFA (Fifaindex.com, 2019). This database provides players' overall ratings (OVR) and potential ratings (POT). Past data is available in the Kaggle European Soccer database (Mathien, Kaggle.com, 2019). The authors do not detail which engineered features were produced after the extraction of the players' ratings, but the information available allows the previously mentioned data to be

arranged to engineer ratings for the team attack, defense and global abilities. This time-dependent data can also infer some other indicators about the morale and team momentum that are otherwise very complex to gauge as they may require subjective assessment.

2.2.3 Team virtual features

Shin and Gasparyan (2014) suggest representing a team by the function team_virtual_features. Any player on the field is represented by a vector of 33 components. Each of these components is an integer in the range from 1 to 100. The 33 features are divided into the following six categories: attacking, defending, skill, movement, power, mentality and goalkeeping. Each category has five features, being the only exception, the category defending for which only three features are described in the Sofifa website (Borjigin, 2019). To build by aggregation the function team_virtual_features, for each category, only the top performers of the team for that specific category are chosen. The function is calculated following the eq.2.1 below.

$$Team\_Virtual\_Features = \begin{cases} \sum top\ 4\ Attacking\ , \in [0, 2000] \\ \sum top\ 4\ Defending\ , \in [0, 1200] \\ \sum top\ 1\ Goalkeeping\ , \in [0, 500] \\ \sum top\ 5\ Skill, Movement, Power, Mentality\ , \in [0, 2500] \end{cases} \quad \text{Eq.2.1}$$

2.2.4 Static and dynamic features

Yezus (2014) defines several engineered features for the teams. Some of the features she defines are static (features that do not depend on the rival teams) and other are dynamic features (they depend on both teams). All the suggested features are normalized (their value fluctuates between 0 to 1).

- **Static feature 'Form'**

The variable res_i represents the result of the i match. The values that the variable res_i takes are 0, 1 or 2 depending on the result: lose, draw or win respectively.

The value of $form$ is calculated according to the following equation:

$$Form = \frac{1}{10}\sum_{i=1}^{5} res_i \qquad \text{Eq.2.2}$$

- **Static feature 'motivation'**

A derby is a soccer game between local rivals. In this case $derby = 1$, otherwise $derby = 0$.

The key positions for motivation are the highest and lower positions in the league ranking: {1, 2, 3, 4, 5, 6, 17, 18}.

$dist$, is the distance to the closest 'key position'.

$left$, is the number of tours left to the end of the season.

$tour = 1\ if\ left < 6$, otherwise $tour = 0$

The value of $motivation$ is then defined as:

$$motivation = min(max\left(1 - \frac{dist}{3 * left}, derby, \frac{tour + dist}{2}\right), 1) \qquad \text{Eq.2.3}$$

- **Dynamic feature 'goal difference'**

The difference between goals scored by each team is called $diff$.

There is a match for which the value of the variable $diff$ takes a maximum value. That value is called $\max\ \_diff$.

The value of the feature 'goal difference' is defined by the following equation:

$$goal\ difference = \frac{1}{2} + \frac{diff}{2 * \max\ \_diff} \qquad \text{Eq.2.4}$$

2.2.5 ELO ratings of a team

The ELO system (Wunderlich and Memmert, 2018) was named after its creator A. Elo. It was initially used to assess the relative strength of chess players (but it may also be applied for assessing football teams). According to A. Elo, the difference of ELO ratings between two players can be used to predict the result of a chess match. In this system the ELO ratings of the players are updated after each match. Points are transferred after each match from the player who loses to the player who wins. The amount of points transferred from the loser of the match to the winner of the match depends on the difference of rates between them.

The eq.2.5 below was suggested by Leung and Joseph (2014):

$$New\_ELO\_player\_A = Old\_ELO\_player\_A + K\ (S_A - E_A) \qquad \text{Eq.2.5}$$

In the above equation the value of K normally taken is 32.

S_A is the actual score in the match (1 if the player A wins, 0.5 if the player A draws and 0 if the player A loses).

E_A is the expected result of a match. It is calculated using the difference between the ELO ratings before the match of both players with the eq.2.6 below.

$$E_A = \frac{1}{\left(1 + 10^{\frac{ELO_B - ELO_A}{400}}\right)} \quad \text{Eq.2.6}$$

Where ELO_B and ELO_A are the ELO ratings for players B and A respectively.

It can be noticed that when both players have the same ELO rating the expected outcome is 0.5, assuming the system a draw result. For significant positive differences of ELO ratings favorable to player A, expected outcomes significantly greater than 0.5 are obtained and when the player A has a lower ELO rating than his rival, expected outcomes are smaller than 0.5, meaning that the predicted result is the defeat of player A.

2.2.6 'Streakiness' of a team

Ulmer and Fernandez (2013) engineer a feature called by the authors $form$ that is used as an assessment parameter of the 'streakiness' of a team. The feature $form$ is related to the results the team obtained during the last x matches. When calculating the parameter, two different problems were found by the authors.

The first problem is to determine the value of the parameter $form$ that should be imputed to the first x matches of the season. For those matches there is not enough information to calculate the parameter $form$. The authors tested the errors found under two possible approaches, a first approach is not to include in the training data the first x matches (this was called in the paper the Ignore Procedure), an alternative approach is to attribute to the value of $form$ during the first x weeks values calculated using equations that consider the number of weeks with information actually available (Scaling Procedure).

The second problem authors faced was to define the value of weeks x to be considered for the calculation of the value of $form$.

Ulmer and Fernandez (2013) calculated the error rates of the algorithm when using different values of the number of weeks x during the calculations. Calculations were produced for both procedures (Ignore and Scaling). Figure 2.1 below shows the results they found when using the machine learning algorithm Support Vector Machines.

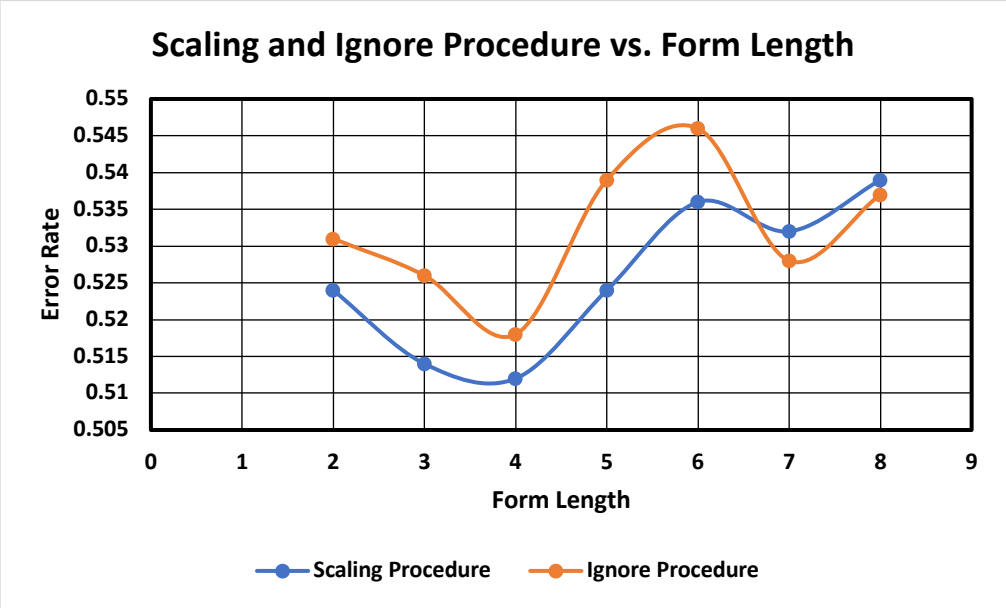

Fig.2.1: SVM algorithm error rates versus $form$ length x. Reproduced from Ulmer and Fernandez (2013).

As it may be seen in the Fig.2.1 above, authors found that ignoring the first 4 matches of the season and using the Scaling Procedure considering the last 4 matches led to smaller errors when using the SVM algorithm.

2.3 Project motivation

The main motivation of this project is to investigate the gaps found in the above literature review. For instance, it was found that some authors did not employ forward chaining as their cross-validation technique for determining the machine learning models hyper-parameters. This fact could have led to a sub-optimal choice of a machine learning's model accompanying parameters,

thus weakening or most likely overestimating its generalization performance and consequently negatively affecting the results claimed by these authors. What results can be obtained by applying forward chaining? (this is a more appropriate statistical technique for validating the results of the time-series domain problem of predicting the outcome of a football match).

On the other hand, other authors exclusively focused on a few select machine learning models. The natural question arises. Were the best models chosen for the explored problem?

2.4 Summary of the chapter

An intrinsic motivation towards this project was to determine whether in the allocated timeframe, a superior model that resulted in better accuracies achieved than those by the online gambling giants could be produced.

To be able to produce a high-quality predicting model it is necessary to understand the machine learning algorithms already available and their underlying theory. This chapter also reviews the features that have been previously used by researchers to produce models for predicting the results of football matches. It is only by having a full understanding of the existing models and features that the previously described task can be successfully attempted.

The following chapter contains a brief overview of the existing machine learning algorithms applicable for football matches' results forecasting.

3 Theoretical background - machine learning algorithms

3.1 Introduction to machine learning algorithms

Machine learning algorithms may be used to predict football match results. If a machine learning algorithm is provided with a large set of data from previous matches, it will be able to predict the result of matches that are not part of the originally provided data set.

The data provided to the algorithm is called the training set as it is used to train the algorithm that creates a model able to predict the results of matches that are not part of the training set.

The data set supplied to the machine learning algorithm includes two types of data: features and labels. In football, some relevant features are the ratings of the players of the match, the number of penalties, the number of corners, etc. The labels are the results of the matches.

Using mathematical language, as shown in eq.3.1 below, the features would be variables 'X' or inputs, the labels would be outputs 'y' (home win, draw, away win) and the machine learning algorithm builds the model or function 'f'.

$$y = f(X) \qquad \text{Eq.3.1}$$

Increasing the number of matches and the number of relevant features included on the training set generally increases the accuracy of the model 'f' when predicting results, but often leads to longer training and predicting times.

3.2 Learning paradigms

The most important Learning Paradigms are the Supervised, Unsupervised and Reinforcement algorithms. Each of these paradigms is used for a specific learning task.

3.2.1 Supervised learning

In supervised learning, the algorithm is trained using a set of data (training data-set) which includes both, the features and the results or labels to be predicted. Using those pairs of data, a model able to produce results for a new set of data defined only by their features is produced.

Using the mathematical language introduced in the previous section, in supervised learning the pairs (X, y) of the training data are known and the aim is to build a model whose function f minimizes the error of the model when it predicts for previously unseen data.

The machine learning algorithms used for predicting football results use data sets including inputs (features) and outputs (labels or results) therefore, they are supervised algorithms. To build a model able to predict outputs, the algorithms will be first trained with pairs of inputs and outputs.

Classification and regression algorithms

The supervised machine learning algorithms can be divided into two groups, the regression and the classification techniques. In football, a classification technique could predict a result such as home win, draw or away win (class results). A classification technique may also predict the number of goals (integers) scored by the home team and the away team, for example (2,0). However, a regression algorithm may predict as a result of the same football match a pair of rational numbers (1.9,0.3) for the goals scored by the home and away teams respectively and therefore a home win result.

Predicting the result of a football match is a multiclass (home win, draw, away win) classification problem with imbalance seen between the classes. This is because home wins, draws and away wins are not equally likely to occur as evidenced by fig.3.1 below.

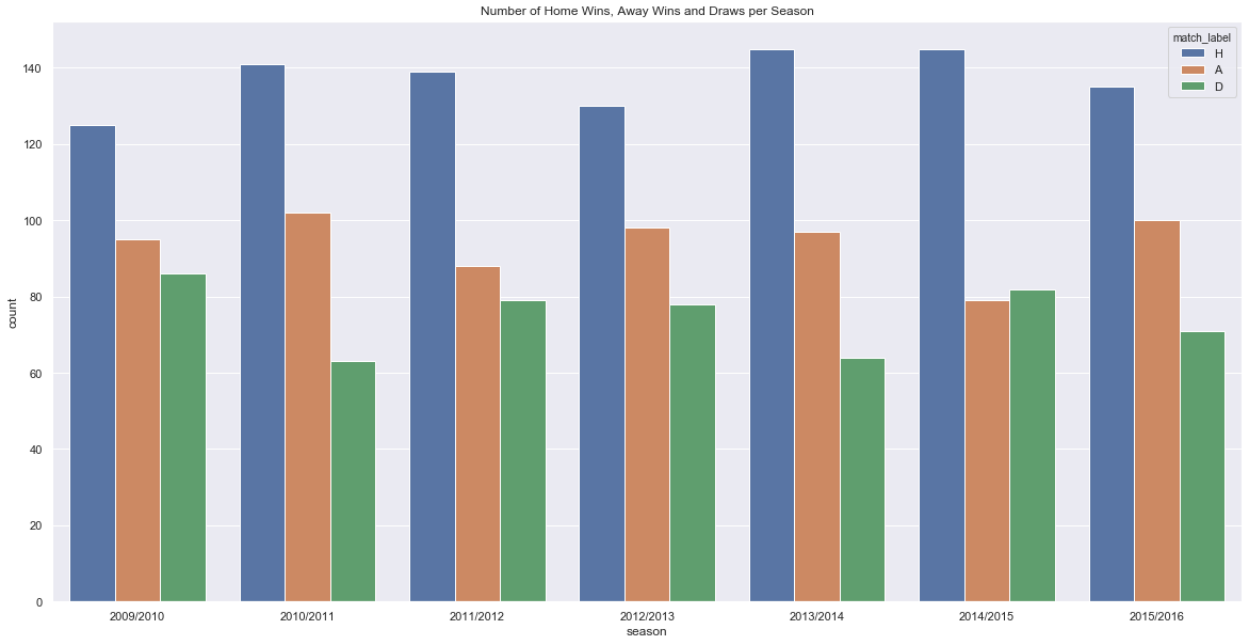

Fig.3.1: Number of home wins, away wins and draws in the German Bundesliga dataset.

3.2.2 Unsupervised learning

In unsupervised learning, the algorithm is built using only the features of a data set (input data), as there is no known output. Using the mathematical language introduced in the previous section, in unsupervised learning, only the features X of the training data are known and the algorithm is asked to provided knowledge out of the supplied data.

In the context of this project, unsupervised learning can be used for dimensionality reduction. Many features such as the rating of the twenty-two players of the game, the number of shots, the distances and angles of the shots, the number of penalties, fouls etc. may be available. Unsupervised learning may summarize the most important characteristics using a smaller number of features. This would help to reduce the training time of a supervised algorithm, and the predicting time for the model. Unsupervised learning in football may also be used to divide matches, teams or players into similar groups or clusters likely to behave in a similar manner.

3.2.3 Reinforcement learning (RL)

In this machine learning paradigm, an agent evaluates the current state of the environment, performs an optimal action with the intention of maximizing returns, and after, each action gets feedback from the environment (Vaibhavi, MarkTechPost, 2019). This process is called Markov Decision Process and it is illustrated in Fig.3.2 below.

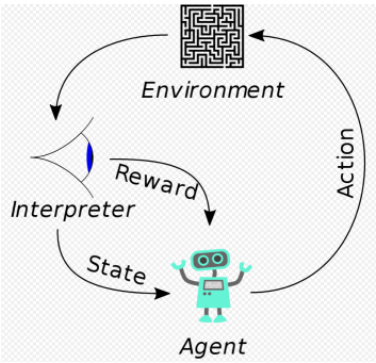

Fig.3.2: Reinforcement Learning Paradigm (En.wikipedia.org, Reinforcement Learning, 2019).

Some of the fields where the RL paradigm is successfully applied currently are: traffic light control, robotics, web system configuration, chemistry, personalized recommendations, bidding and advertising, games and deep learning (Medium, *Applications of Reinforcement Learning in Real World,* 2019).

In the context of this dissertation, a possible application of this paradigm could be devising a betting strategy with the intention of maximizing the return on bets placed.

3.3 Types of supervised machine learning algorithms

In this section, some important supervised machine learning algorithms are introduced.

3.3.1 Linear models

The linear algorithms assume that the labels or outputs are linked to the inputs or features through a linear function.

Using mathematical language in matrix notation (En.wikipedia.org, Linear Regression, 2019):

$$y = X\beta + \varepsilon \qquad \text{Eq.3.2}$$

In the above equation, for a training set of n matches, and p features:

- y is an n -dimensional vector with the results or data labels for the training data.
- X is a matrix of $p + 1$ columns and n rows in which the first column is a vector with all its elements equal to 1. The p columns after the first, gather one feature per each column (e.g. each element of a specific vector column is the number of corners shot by the away team in each team).
- β is a vector of dimension $p + 1$. The first column of this vector is the intercept, the next columns are the regression coefficients for each feature.
- ε is an n -dimensional vector called the error term. The error term collects the influences of other features that are not part of the matrix X.

The Linear Regression machine learning algorithm, using the least squares method, calculates the vector β from the data labels y and the matrix X that includes the values of the features.

The Linear Regression machine learning model is therefore defined by the vector β and it predicts the results of $X\beta$ for an input set X**.** The output of the linear regression model would be a vector y whose elements are real numbers.

In some classification problems, the output can only take two possible values (the output belongs or does not to a predefined class). A football prediction machine learning algorithm that would consider only two possible outcomes, i.e. home win ($y = 1$) versus home does not win ($y = 0$) (this would include away win and draw), would be a binary classification algorithm. Such a

problem may be solved building a Logistic Regression Classification model. Mathematically, the problem would be expressed as:

$$\begin{aligned} y &= 1 \qquad if \qquad X\beta + \varepsilon \\ y &= 0 \qquad else \end{aligned} \qquad \text{Eq.3.3}$$

And the construction of the model would consist in finding the β values that best fit the above equation. The output of the Logistic Regression model would be a vector y whose elements are categoric labels.

Advantages and disadvantages of the linear models may be found on the table 3.1 below.

Table 3.1: Linear Models, advantages and disadvantages (Muller and Guido, 2018, p.67)

Advantages	Disadvantages
Model easy to build. Once the coefficients of the model β are obtained, it becomes easy to interpret which features are more important and if they influence positively or negatively the outcome.	If there is no linearity between the input variables X and the outcome y in the model's working range, predictions of the model will be poor. The model performs poorly if several input variables are significantly correlated.

The library Scikit-learn includes the modules *LinearRegression* and *LogisticRegression* able to build linear machine learning classifiers.

3.3.2 Decision trees

Decision Trees are machine learning models used for both classification and regression problems. Using the training data, decision trees algorithms infer a hierarchy of *if* and *else* questions that lead to the outcome vector y.

The algorithm is called the Decision Tree as it may be graphically represented using a tree. The tree has nodes that split for the different values of a specific feature. The tree also has edges (branches) that are outcomes of a split and go to the next feature or nodes. The root is the node where the first split is performed and the tree also has leaves that are the terminal nodes which predict the outputs of the problem.

In the Random Forest algorithm (fig.3.3), the training-data is split in several subsets and each of these subsets is used to train a specific tree of a 'forest' made of a large number of trees. When the Random Forest is solving a classification problem, the final output is the class that has been found as the solution in the greatest number of trees. For regression problems, the final output is the mean of the outputs found for all the trees.

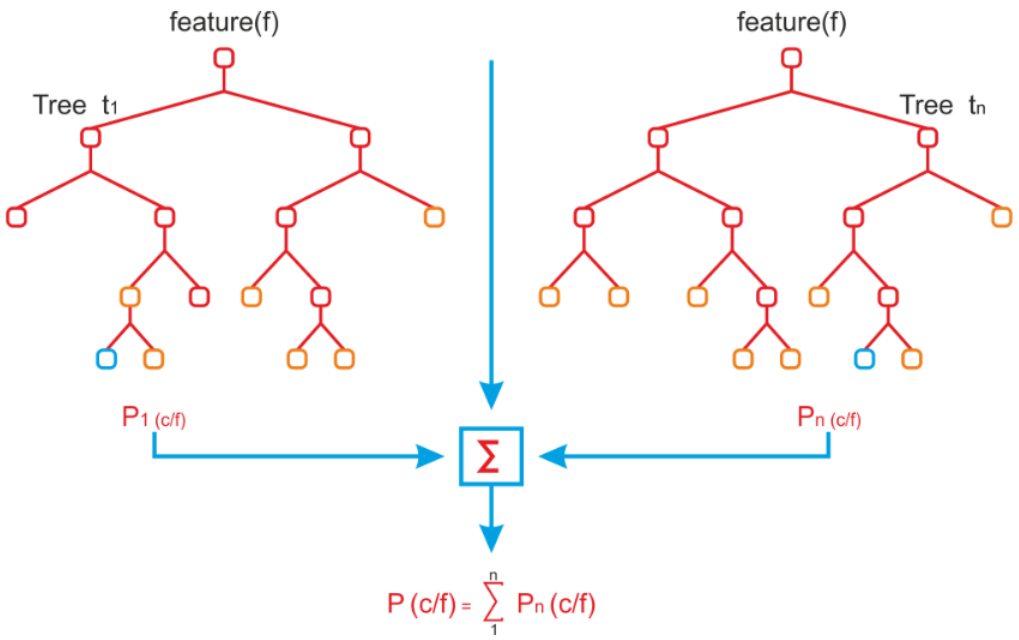

Fig.3.3: Random Forest and its assembling technique explained (O'Reilly, Scala Machine Learning Projects, 2019).

Advantages and disadvantages of decision trees based algorithms may be found on the table 3.2 below.

Table 3.2: Decision Trees, advantages and disadvantages (Muller and Guido, 2018, p.82-83)

Advantages	Disadvantages
Easy to interpret graphical results. The algorithm supplies an array called 'feature_importances_' with a rating in the range from 0 to 1 for each feature. A feature with 0 rating means that the tree does not consider it, and a feature with rating 1 means that it fully predicts the label. The Random Forest algorithm reduces the tendency to overfitting by averaging the outputs and injecting randomness into the ensemble of trees.	To avoid overfitting of the model: • before the decision tree gets too deep and complex, its growth has to be limited (pre-pruning). • after building the tree, it is often required to remove the nodes (features) with little information (post-pruning). Random Forest algorithms are slower than linear models and require more memory.

The library Scikit-learn includes the modules *DecisionTreeClassifier, DecisionTreeRegressor, RandomForestRegressor* and *RandomForestClassifier* able to build decision tree machine learning models.

3.3.3 Naïve Bayes classifiers

During the training process that the algorithm undertakes, the Naïve Bayes Classifiers defines parameters looking at each feature individually and it collects for each feature simple per-class statistics. The Naïve Bayes classifiers are applicable for continuous data and store the average values as well as the standard deviations of each feature for each class (Muller and Guido, 2018, p.69). An example in football prediction would be to have data for many shots (rows) and two features of the shots that may be the angle and distance to the goalposts. The algorithm collects statistics of the mean and standard deviations for shots that resulted in a goal and also for the

shots that did not result in a goal. Hence, the Naïve Bayes Classifier algorithm produces a model able to predict if a shot from a defined distance and angle results in a goal or not.

The advantages and disadvantages of the Naïve Bayes Classifiers may be found on the table 3.3 below.

Table 3.3: Naïve Bayes Classifiers, advantages and disadvantages (Muller and Guido, 2018, p.70)

Advantages	Disadvantages
Algorithm is very fast to train and prediction with the model is also fast. Good performance for large number of features (>100). For sparse data sets (where for plenty of the features the value is 0 most of the times) performance is high.	In general terms, the Naïve Bayes classifiers have lower generalization performance than the linear classifiers.

The library Scikit-learn includes the module *GaussianNB* able to build Naïve Bayes Classifying machine learning models.

3.3.4 The K-nearest-neighbours (KNN)

This machine learning algorithm builds a model by storing the training data supplied. Given the features of a new data point, the prediction for its label is performed by finding the labels of the closest points found in the training data.

The parameter k defines the number of closest neighbours to be taken in order to find the label of the input data. The K-nearest-neighbours algorithm may perform classification and regression. When performing classification, the label predicted for the input is the class label most common

among the k closest neighbours. When carrying out regression, the output predicted for the input is the average of the values of the k closest neighbours.

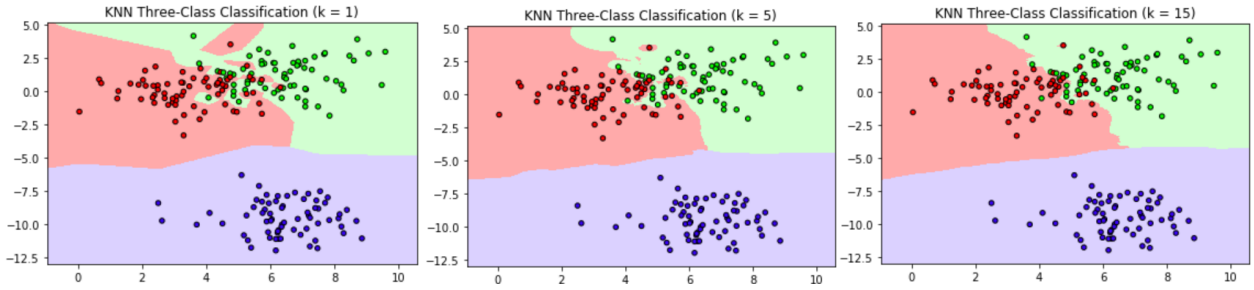

Fig.3.4: Decision boundaries for increasing number of neighbours.

Fig.3.4 above shows data points of three classes (represented by the red, green and blue colours) and the decision boundaries defined by a KNN algorithm. The decision boundaries separate the plot in three areas of different colors (pink, light green and light purple). It can be noticed that for $k = 1$ the boundaries followed are very close to the points, and when the number of neighbours k is increased, the decision boundary becomes smoother. The advantages and disadvantages of the K-nearest-neighbours models are illustrated on the table 3.4 below.

Table 3.4: K-nearest-neighbors, advantages and disadvantages (Muller and Guido, 2018, p.44)

Advantages	Disadvantages
Easy to understand. Building the model is fast, it just consists in storing the training data.	Predictions are slow if training-set is large either in number of features or in number of data points. For large number of features (>100) performance is poor. For sparse data sets (where for plenty of features the value is 0 most of the times) performance is poor.

The library Scikit-learn includes the modules *KNeighborsClassifier* and *KNeighborsRegressor* able to build k-nearest-neighbours machine learning models.

3.3.5 Support vector machines (SVMs)

The SVMs are supervised algorithms that can be used for both classification and regression.

Given a training data set where the data points are all labelled as belonging to one of two classes, the SVM algorithm produces a model that assigns the new points to one of those two classes. The SVM produces a non-probabilistic binary linear classifier.

The SVM models map data points in a multi-dimensional space (as many dimensions as there are features), and the points that belong to each class are separated by gaps that are as wide as possible. The decision boundaries are called hyperplanes. If the number of features is two, the hyperplane is a line, if it is three it is a plane, but if the number of features is four or more, it is not possible to visualize it.

The class of a new sample point is predicted by the side of the boundary it falls onto when it is plotted in the model's map. Fig.3.5 below shows the scatter plot of points that belong to two different classes (represented by red and yellow points). Data points have only two features (Feature 1 and Feature 2), therefore the hyperplane is a line (continuous grey line in the plot). This hyperplane becomes the decision boundary. The two yellow points circled define the vector of the direction of the hyperplane. The two grey dotted lines define the margins.

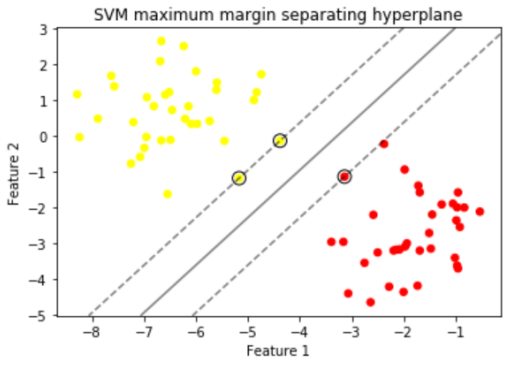

Fig.3.5: SVM model, two classes and the decision boundary.

Advantages and disadvantages of the Support Vector Machines may be found on the table 3.5 below.

Table 3.5: Support Vector Machines, advantages and disadvantages (Statinfer.com, 2019)

Advantages	Disadvantages
It works well for a high number of features. It also works well for data such as text and images.	If the training data-set is large, the training takes long. The final model is difficult to understand, and it is also difficult to tune some parameters. It does not give the probability of belonging to a class, it gives the predicted class as ouput.

The library Scikit-learn includes the modules *svm.SVC* (classifier) and *svm.SVR* (regressor). They are able to build Support Vector Machine models.

3.3.6 Artificial neural networks (ANNs)

Artificial Neural Networks (ANNs) are often used for problems within the reinforcement learning paradigm. Neural Networks are designed to emulate biological neural networks and learn in a manner similar to how human brains learn (reinforcement learning).

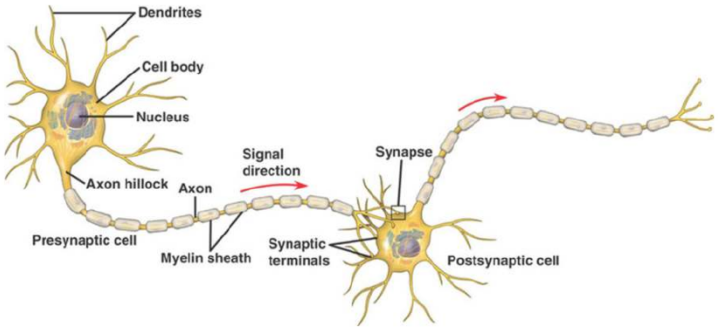

Fig.3.6: Two connected neurons (Anon, 2019).

The fig.3.6 above illustrates how a human brain works. The dendrites receive inputs and based on those inputs the neuron's cell body produces an electrical output that is transmitted through the axon to the dendrites of the next neuron. Neurons are connected forming a network with the axon of a neuron connected to the dendrites of several neurons.

Based on biological neural networks, computer experts have designed artificial neural networks to solve problems that may be considered simple for human beings but difficult for computers, such as image recognition.

The basic unit of the artificial neural network is the perceptron. Fig.3.7 below illustrates the functioning of a perceptron. The perceptron receives inputs (input 1 and input 2) in their dendrites. The different inputs and the bias first get weighted and thereafter added together in the preprocessor of the perceptron body. The addition of the weighted signals is passed through a nonlinear activation function in the processor, generating the output. This is called the feed-forward process.

The two different activation functions that are most often used are the hyperbolic tangent and the rectified linear unit. The rectified linear unit or *relu* function provides values of $y = 0$ for values of $x \leq 0$ and values of $y = x$ for values of $x > 0$ (Muller and Guido, 2018, p.107). Usual weights are in the range [-1,1]. The bias is introduced to avoid having outputs equal to zero if the input 1 and the input 2 are both equal to zero.

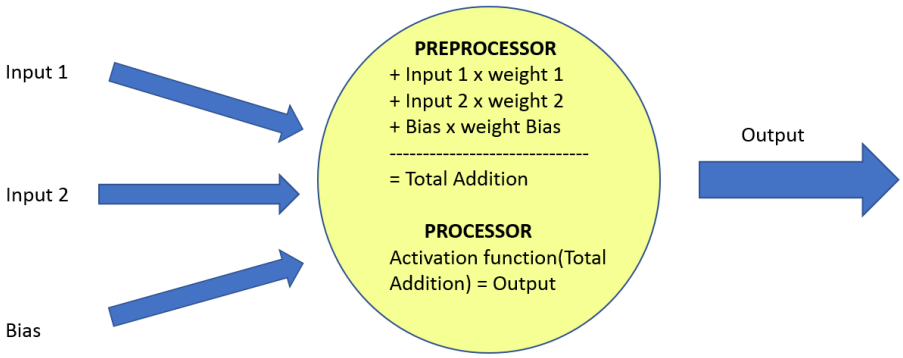

Fig.3.7: Perceptron: Preprocessor and Processor.

The training of the perceptron is carried out according to the following steps: the perceptron is given inputs for which the answer is known, an answer is required from the perceptron, the error (difference between known answer and answer provided by the perceptron) is calculated and weights are adjusted according to the error. This iterative process is repeated starting from the first step until the error decreases to reach a predefined value.

In a similar manner in which neurons in a human brain are connected to each other, outputs of some perceptrons can be connected to the inputs of other perceptrons forming artificial neural networks. Fig.3.8 below shows an artificial neural network built by linking four layers of perceptrons together. The inputs of the perceptrons on the left column (blue) are the features of the neural model and the output of the perceptron on the right column (green) represent the result. The fact that the activation function of the perceptrons is nonlinear allows the artificial neural ensemble to build more complicated models than a linear network could (Muller and Guido, 2018, p.106).

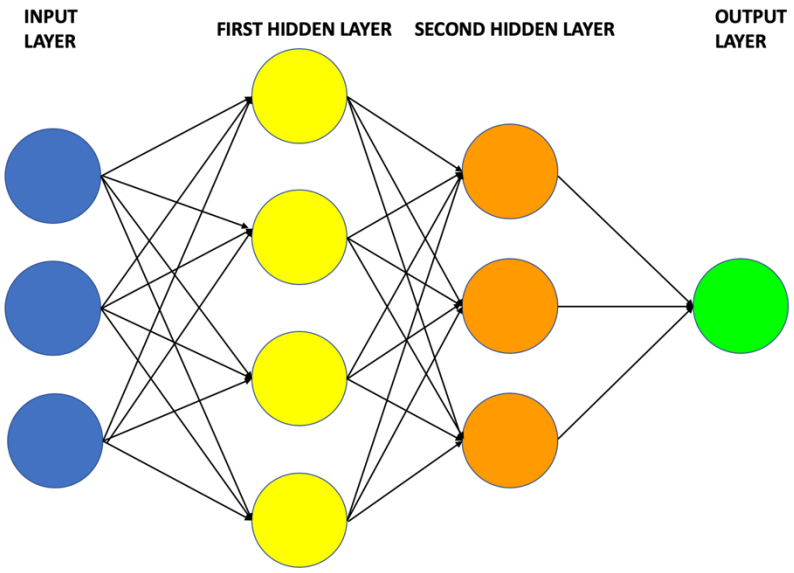

Fig.3.8: Artificial neural network with two hidden layers.

The whole artificial neural network can also be trained in a manner similar to the one in which perceptrons are trained. However, in this case, the error term is distributed through the layers by modifying the weights at the nodes (Nielsen, 2019). This process is called backpropagation.

Advantages and disadvantages of the Artificial Neural Networks are explained on the table 3.6 below.

Table 3.6: Artificial Neural Networks (ANNs) advantages and disadvantages (Muller and Guido, 2018, p.117-118)

Advantages	Disadvantages
Neural networks can be used to build very complex models. If enough data, training time, and computer power is available, results may be better than when other machine learning algorithms are used.	Some of these algorithms usually do not run on Windows machines. They require long training times. They work well with homogeneous data, that is, features of similar kinds, but if features are nonhomogeneous, then tree-based algorithms are preferable. Tuning of parameters such as number of perceptrons per layer and number of layers is complex.

The library Scikit-learn includes the modules *neural_network.MLPRegressor* and *neural_network.MLPClassifier* able to build Artificial Neural Networks machine learning models.

3.3.7 Gradient boosting algorithms

Gradient Boosting Algorithms can be used for both classification and regression problems. A gradient boosting algorithm is a powerful algorithm formed with an ensemble of other not so powerful algorithms (weak learners). Usually, the ensemble is made of individual decision trees, but this algorithm is different from the random forest algorithm. In the random forest algorithm,

the final output is the class that has been found as the solution in the greatest number of trees. For regression problems, the final output in the random forest algorithm is the mean of the values found for all the trees.

However, the gradient boosting algorithm builds a model assembling trees in series. In this ensemble, each tree is used to try to correct the mistakes of the previous tree (Muller and Guido, 2018, p.88).

The Random Forest algorithm injects randomness into the ensemble of trees by means like bootstrapping. In the Gradient Boosting Algorithm, by default, no randomness is injected, but significant pre-pruning is applied to avoid an ensemble of trees too deep and complex. The depth of trees built in gradient boosting algorithms is usually limited to a maximum of five leaf nodes, making the model fast and limiting the amount of memory required to run it.

Advantages and disadvantages of the gradient boosting algorithms may be found on the table 3.7 below.

Table 3.7: Gradient boosting algorithms, advantages and disadvantages (Muller and Guido, 2018, p.91-92)

Advantages	Disadvantages
The algorithm supplies an array called 'feature_importances_' with a rating in the range from 0 to 1 for each feature. A feature with 0 rating means that the tree does not consider it, and a feature with rating 1 means that it fully predicts the label. Predictions are fast. Memory consumption is small.	To avoid overfitting of the model: • before the decision tree is too deep and complex its growth has to be limited (pre-pruning) using the parameter max_depth. • The parameter called learning_rate has to be optimized. This parameter controls the strength with which each tree corrects the mistakes of the previous tree. • Training time may be long.

The library Scikit-learn includes the modules *ensemble.GradientBoostingRegressor and ensemble.GradientBoostingClassifier.* They are able to build gradient boosting machine learning models.

3.4 Summary of the chapter

In this chapter, it has been learned that predicting the outcome of a football match could be done either through a supervised machine learning regression algorithm (where the number of goals to be scored by each team is predicted) or a supervised machine learning multi-classification model (where the outcome of a match: home win, draw or away win is determined). In this dissertation a multiclassification domain problem is dealt with. However, it should be noticed that the problem can be also solved through a regression driven model to determine the number of goals scored by each team and therefore the final result of the football match (class label).

After analyzing several machine learning models in this section (the list presented above is by no means fully comprehensive), it was decided to use the following supervised machine learning models for the prediction of the outcome of a football match:

- K nearest neighbours.
- Logistic Regression.
- Gaussian Naïve Bayes.
- Decision trees.
- Random forests.
- Gradient boosted decision trees.
- Support vector machines.

The observant reader with a keen eye for detail will have noticed that despite their current popularity, artificial neural networks, are not among the chosen models. Not using neural networks was an explicit decision made. Neural networks, also referred as deep learning (a subfield of artificial intelligence) are a craft on their own, and optimizing their hyper-parameters is an art by itself (Medium. Is Optimizing your Neural Network a Dark Art?, 2019).

The chapter that follows discusses the various steps taken throughout the course of this project which led to predict the outcome of football matches occurring in the German Bundesliga 2015/2016 season.

4 Methodology

The methodology of a project are the processes and resources that are used to plan and deploy the project from the project definition to its termination. This section describes the methodology followed during this project.

4.1 Software engineering methodology

4.1.1 An Agile method: variant of Scrum

The project has followed an Agile Methodology. A variant of Scrum consisting of one-week sprints followed by meetings with the supervisor was used. In this variant, the supervisor acts as the customer and during the weekly meetings assesses what has been done, performs a gap analysis between what has been done and what should have been done, and provides priorities for the next week. The author of this project acts as a hybrid developer and scrum master, as he is responsible for the project. During these meetings, the author explained the weekly increments (achievements of the period), the difficulties faced and how they were overcome. Figure 4.1 below is used to illustrate the variant of the Scrum methodology used.

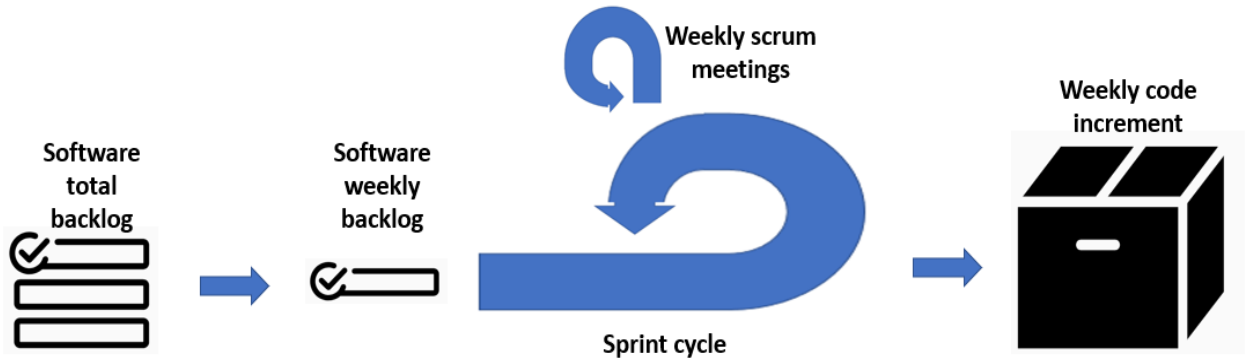

Fig.4.1: High level view of the scrum process. Icons downloaded from: (Noun Project, 2019).

4.1.2 An Agile method: variant of test-driven development

The software life cycle model is the set of activities performed during the software development process. To successfully complete this project, it is necessary to choose a suitable life cycle model. The life cycle model defines the sequence of software activities and transitions between them. Given the project's lead time, the interaction with my customer (supervisor), resources available and my level of expertise, an Agile type of software development methodology called Test Driven Development (TDD) was chosen (Igea, 2019, p.38).

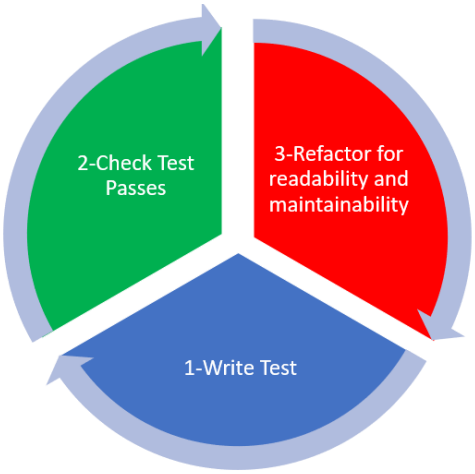

Fig.4.2: Test-driven development.

The process of test-driven development can be explained using the fig.4.2 above. The process is iterative and incremental. The process starts by writing a test that will be used to check a piece of code that has not been written yet. Afterwards, code that is able to pass the test is written. Following the successful completion of the test, refactoring of the code is carried out, so the code becomes more maintainable and readable. More loops are performed adding new requirements, and the iterations continue until the project is successfully completed (Igea, 2019, p.39).

During the project, the Pandas 'info' and the 'head', 'tail' and 'describe' methods were extensively used throughout the process of software development to ensure that the code written produced what the author intended to do.

4.1 The dataset

The main data-set used in this project was downloaded from the Kaggle dataset 'European Soccer Database' (Mathien, Kaggle.com, 2019). This data set was collected and refined mainly from the three following sources:

- Information from bookmakers about betting odds, http://www.football-data.co.uk/ (Football-data.co.uk, 2019).
- Information about the players, https://sofifa.com/ (Borjigin, 2019).
- Information about events, line-ups, scores: http://football-data.mx-api.enetscores.com/ (Fifaindex.com, 2019).

The dataset includes information for eleven countries' leagues ranging from 2008-2009 to 2015-2016 seasons. The data includes:

- more than ten thousand players with information that includes their ratings throughout the seasons.
- more than twenty-five thousand matches with their results, team line-ups, etc.
- betting odds from several bookmakers.
- events occurred during the matches such as number of goals per team, number of penalties, fouls, corners, shots. Events are time-stamped and the locations of the football ground where events happened are also available.

4.2 The software, libraries and tools employed

The high-level programming language used in this project is Python 3.6. Python and its libraries are frequently used to solve Data Analysis, Data Science and Machine Learning problems.

The notebook environment used is Jupyter Notebook. The Jupyter Notebooks may be easily shared after being saved as a 'ipynb' file. Jupyter Notebooks may also be shared, and version controlled using GitHub.

Python has its own standard library, but it may also be used to run many third-party libraries for solving numerous domain-driven problems. The main libraries used during the project were:

- NumPy: used for scientific computing.
- matplotlib: used for plotting figures.
- Pandas: for manipulation and analysis of data-frames.
- SciPy: scientific computing.
- Seaborn: visualization of graphics.
- Scikit-learn: for machine learning.
- FuzzyWuzzy: checking strings similarities.

The software called 'DB Browser for SQLite' was used to extract data-frames in the csv file format through SQL queries formulated to any data-sets that were stored in an SQL database.

4.3 The high-level overview of the machine learning workflow

The fig.4.3 below shows the high-Level overview of the methodology followed and the most important milestones throughout this project.

The following main steps were performed:

a) The referenced web pages are downloaded as either SQL databases or csv files.
b) The data is explored to determine which tables are useful for the project.
c) Using the 'DB Browser for SQLite', the tables of interest stored in an SQL database are converted to csv files.
d) The csv files are converted into Pandas data-frames in which further data-exploration and data wrangling is performed.
e) Feature Extraction and Selection takes place in order to represent the data in the best way possible for the machine learning models.
f) Different algorithms of Machine Learning are run.
g) Parameters are optimized.

h) Results for the 2015/2016 season (the testing data-set) are obtained.

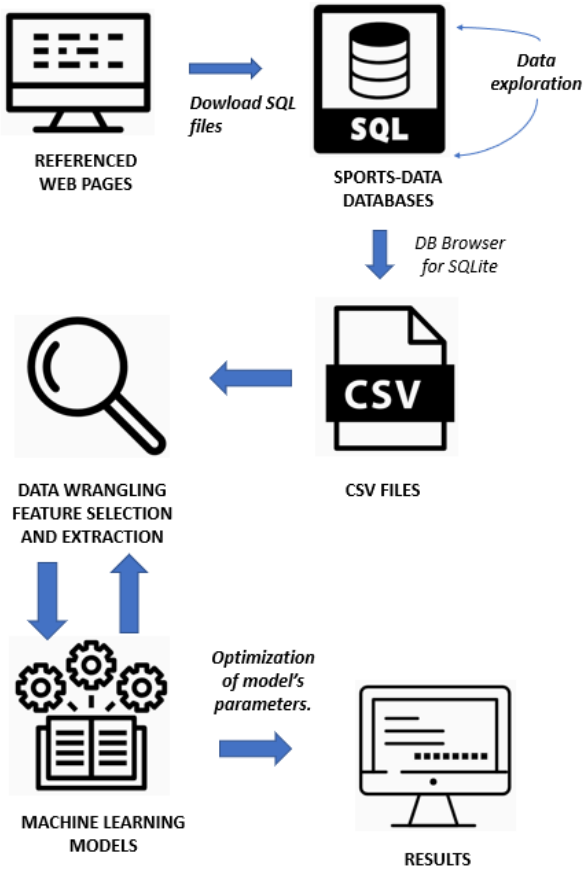

Fig.4.3: High-level overview of the machine learning workflow. Icons downloaded from: (Noun Project, 2019).

4.4 The pre-processing of the data

The steps included in the pre-processing of the data are: data wrangling, feature engineering and feature selection.

The data wrangling is performed to display the data-set information in a format that facilitates decision-making.

4.4.1 The first attempt at the data wrangling

The first attempt during the data-acquisition step of the data wrangling consisted of downloading various data-sets from the Kaggle website for Data Science. These datasets individually contained scattered information about the football matches and the players of each team. The aim was to build a rich dataset containing as much information (features) as possible.

The main European Football database is stored in an SQLite database. The software called 'DB Browser for SQLite' was used to extract the following csv files:

- 'main_matches.csv', with information about more than 25,000 football matches - seasons 2008 to 2016. (Mathien, Kaggle.com, 2019).
- 'main_players.csv', with information about more than 10,000 football players ratings - seasons 2008 to 2016. (Mathien, Kaggle.com, 2019).

The following datasets were already provided in csv format and were directly downloaded from Kaggle:

- 'fifa_players_17.csv', with information about more than 17,000 football players ratings - FIFA 2017. (Kaggle.com, Complete FIFA 2017 Player dataset (Global), 2019).
- 'fifa_players_18++.csv', with information about 185 sport parameters for every player - FIFA 2018. (Kaggle.com, FIFA 18 Complete Player Dataset, 2019).
- 'fifa_players_19.csv', with information about attributes for every player registered in the latest edition of FIFA 19 database. (Kaggle.com, FIFA 19 complete player dataset, 2019).
- 'new_matches.csv', this dataset contains information about Premier Leagues, *Primera Division*, *Lique 1*, Serie A and *Bundesliga* from 2004-2005 to 2018-2019, more than 28,000 games. (Kaggle.com, European Football Games, 2019).
- 'football_betting_odds.csv', with information about football-data.co.uk collected results, fixtures and market odds. (Kaggle.com, World Soccer - archive of soccer results and odds, 2019).

- ‘events.csv’, with information about event data about each football match. It also contains text commentaries scraped from: bbc.com, espn.com and onefootball.com. (Kaggle.com, Football Events, 2019).
- ‘ginf.csv’, with metadata and market odds about each soccer’s match. The odds are collected from oddsportal.com. (Kaggle.com, Football Events, 2019).

The above files were imported into the Jupyter Notebook environment as Pandas’ data-frames. Immediately after this, the assessment of each data-set started. The data-frames were visually assessed to get acquainted with the data. To get a deeper understanding of the data, the following checks were employed: visualizing the heads and tails of the data-frames, obtaining the number of non-null entries and the data types of each column as well as obtaining descriptive statistics of each numeric data-type column. The distribution of the null data (fig.4.4 below) was also visualized.

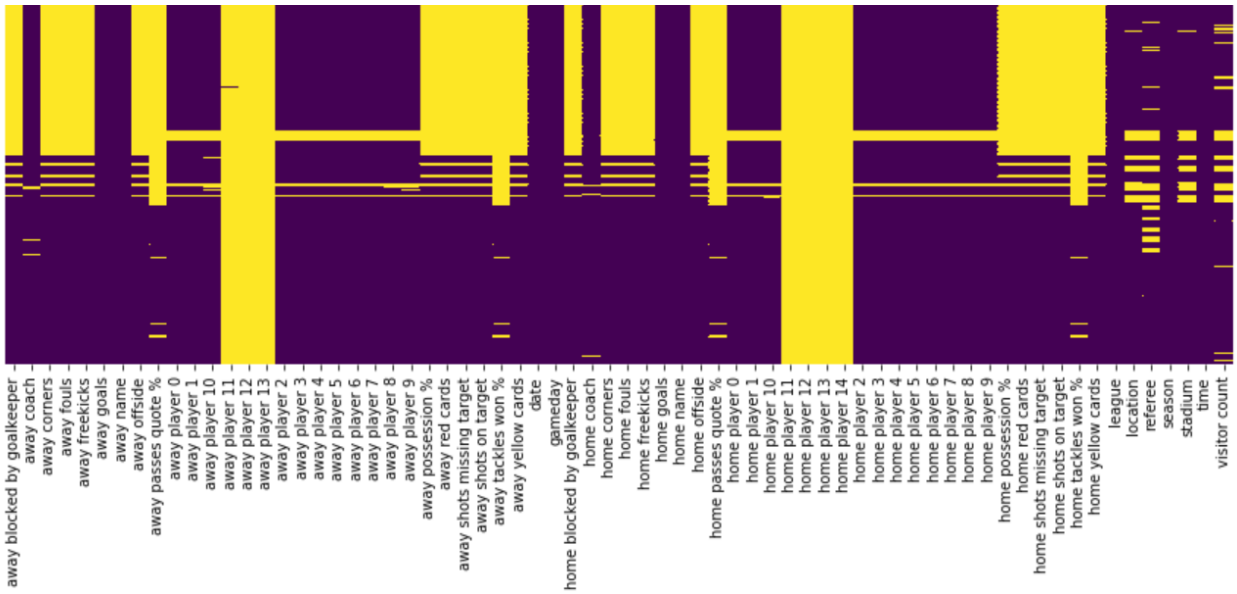

Fig.4.4: Overview of the data-frame: ‘new_matches’, in yellow: null data and in purple: non-null data.

It was also checked whether the data-frames contained duplicated rows and if in any given league and season the number of matches played for each team was the same.

For the data-frames ‘fifa_players_17’ and ‘fifa_players_18’ coming from the files ‘fifa_players_17.csv’ and ‘fifa_players_18++.csv’, it was found that some players’ names were repeated several times.

Quality issues

During the performance of the previous analysis, some quality issues were found for the ‘main_matches’ data-frame. The following list describes the most important ones:

1. Some column names were not descriptive of what they contain.

2. For certain columns, some missing values were found.

3. In-game events are found in XML format (goal, shoton, shotoff, foulcommit, card, cross, corner, possession) within one column.

4. The dates in the column 'date' are strings rather than datetime format.

5. There were missing games for the Belgium League Season 2013-2014.

6. Some inconsistent data was found for the following Poland seasons and teams:

   - Match season 2008-2009, team Polonia Bytom 30.
   - Match season 2010-2011, team Polonia Bytom 30.
   - Match season 2011-2012, team Widzew Łódź 30.

For the second data-frame with information about football matches, ‘new_matches’, some quality issues were also found. The most important ones found were number 1, 2 and 4 described in the list above for the ‘main_matches’ data-frame.

During the analysis, some quality issues were also found for the players' information. The list below describes the main ones:

- Some missing values were found for certain columns.
- The dates in columns 'date' and 'birthday' were strings rather than datetime format.

Tidiness issues

During the performance of the previous analysis, some tidiness issues were found for the data-frame 'main_matches'. The following list describes the most important ones:

- The in-game events were not atomic, they should be split into respective teams' events.
- Several columns were duplicated or unnecessary.

During the analysis, some tidiness issues were also found for the players' information. The list below describes the main ones:

- For the 'fifa_players_17.csv', there was a missing primary key when compared to the 'main_players' table.
- The column headings of the post 2017 player-ratings data-frames needed to be consistent with the 'main_players' data-frame in order to be able to concatenate all of them together.

Actions were required to depurate the quality and the tidiness issues programmatically. The following processes were run on the players' data-frames:

- a new column 'date' was added.
- the missing primary key was added.
- checking how many names are duplicated and dropping rows with duplicate names, as for these cases it is not possible to accurately perform an inambiguous join operation between data-frames based solely on the name.

- The players databases were then concatenated (stacked on top of each other in a chronological order).

Then, it was checked how many matches could actually be used from the 'new_matches' table. A football match may be saved if a direct match is found between the 'new_matches' data-frame football players' name to their corresponding ID in the master data-frame of football players.

After finding out that it was only slightly more than 20 matches that could be saved according to the above plan, another process was designed with the purpose of rescuing the other football-matches. Using the Python library FuzzyWuzzy to check for similarity of strings, a function called 'match_name' was written. This function finds the best match from a player's name on the matches data-frame to the player's name on the players' data-frame. After the execution of this function, a data-frame was created with the players' names and their corresponding best match. As the results of this method have some room for error, the data-frame was converted to an excel file and checked manually.

Eventually, this branch of the project was abandoned. It was learnt that while conducting further literature review, that the problem of predicting the outcome of a football match falls under the category of a 'time-series' problem (Yiannakis et al., 2006, p.96). Even though significant efforts were made trying to rescue the matches with missing player IDs, a large number of time-gaps without matches in between were found. If a model were to be created with the missing time-dependent data, it would not be able to accurately reflect the ground-truth of what happened during those missing matches up until the current match (e.g. cumulative shots on target would not represent the actual cumulative shots on target up until that match, so the validity of the model would become questionable due to the fact that there are missing matches in between).

4.4.2 The second attempt at the data wrangling

Even though the previously described branch of the project was abandoned for the reason outlined earlier, the experience that was acquired on the data-wrangling process proved to be

extremely useful for the next attempt of depuration of the data-set. Again, the starting point is the Kaggle European Soccer Database (Mathien, Kaggle.com, 2019).

The process started using the software 'DB Browser for SQLite' to extract from the SQLite database uploaded by Mathien the following csv files:

- 'main_matches.csv' with information about more than 25,000 football matches - seasons 2008 to 2016.
- 'main_players.csv' with information about more than 10,000 football players ratings - seasons 2008 to 2016.

The following was directly downloaded from Kaggle:

- 'football_betting_odds.csv' with football-data.co.uk collected results, fixtures and market odds as well as in-game statistics.

Those three files were imported into the Jupyter Notebook environment as Pandas' data-frames with the names 'main_matches', 'main_players' and 'football_data' respectively.

The same standard procedures that were utilized during the first attempt were employed again (acquainting the data, visual inspection etc.).

For the data-frame 'main_matches', the in-game events (goal, shoton, shotoff, foulcommit, card, cross, corner, possession) were still in XML format. These events are not atomic, so they should be split into their respective teams' events.

To obtain the in-match statistic data two possible approaches could be used. The first was to use an XML parser to extract the relevant in-game events. This approach was not followed. The approach followed was to obtain the in-match information from the 'football_data' data-frame.

The seasons and the leagues that the data-frames 'football_data' and 'main_matches' had in common were found. After further analysis, the following observations were gathered.

Leagues that do not have useful match statistics from the 'football_data' dataframe were:

1. Belgium Jupiler League.
2. Netherlands Eredivisie.
3. Portugal Liga ZON Sagres.

After conducting further research online, it was also found that the following leagues did not have useful match statistics (these leagues were not in the 'football_data' dataframe):

1. Poland Ekstraklasa.
2. Switzerland Super League.

As a result, the useful leagues to conduct the analysis were the following:

1. England Premier League.
2. France Ligue 1.
3. Germany 1. Bundesliga.
4. Italy Serie A.
5. Scotland Premier League.
6. Spain Liga BBVA.

Therefore, at that time, the data found in all the above six leagues was considered to be useful.

To be able to produce some important engineered features, it is necessary to know who were the players that played in each match. Consequently, the matches that did not contain the player id for all 22 players in a given match were dropped.

Comparing the number of matches per season for each league before and after dropping the matches that had missing some players' IDs, it was found that the most complete dataset was the one corresponding to the German Bundesliga. A data-frame called 'bundesliga' was built. Further work was then performed on this data-frame.

To obtain the important in-game statistics for the data-frame 'bundesliga', an initial merge with the data-frame 'football_data' was attempted on the columns date, home-team and away-team. However, the names of the teams did not coincide, therefore the merge was not successful at first. At this stage, the previous attempt of trying to match players by name using string matching algorithms became very useful, as the acquired knowledge helped to carry out this merge.

The algorithms based on the library FuzzyWuzzy to check similarity of strings were ran to obtain the best string match for the names of the teams in the 'bundesliga' data-frame to the names of the teams in the 'football_data' data-frame. The data-frame produced was exported to an excel file for a more thorough visual check. After several more processes of data preparation, the two tables were successfully merged to the 'main_matches' data-frame.

However, after all the previously explained efforts to identify the missing players, there were still some teams where the identity of some players remained unknown. The lack of unique IDs for some players caused a problem when obtaining the football players' most recent rating before the football match. The next section deals with solving the problem caused by the lack of ratings for some players.

At this stage, a last attempt to recover the player's FIFA IDs was made for the 72 Bundesliga matches. The process consisted in firstly, identifying the missing players' name and then looking it up in the main football players' data-frame. This operation was performed to avoid the imputation of data as preserving the authenticity of the data to the highest possible extent was a priority in order to have reliable results derived from the analysis. Nevertheless, these efforts resulted in vain as it was found out that plenty of the missing players for each match were not present in the players' rating table. As a result, a successful merge to obtain these players' ratings would have proven impossible, unless the players' rating table would first be updated with the missing information of that player.

Rating imputation

The rating of some players was not available in the original data-set (Mathien, Kaggle.com, 2019). This happened because Mathien transferred the data for each player from their Sofifa profiles (Borjigin, 2019) to the table of the matches from enetscores (Football-data.mx-api.enetscores.com, 2019), and to do this, the only common keys were the players' birthdays and names. For many players, as names and birthdays coming from both tables did not produce a perfect match, Mathien wrote a script to search in the internet the names in the football matches table to try to find the names used by the Sofifa website. However, this process was not able to produce results for all the players and that is why the ratings of some players were not available.

To deal with the problem of incomplete data for the players' ratings, two options were considered.

The first option was to drop the 72 matches from the Bundesliga, mainly part of the season 2008-2009. For all those matches the rating of at least one player was missing. This first option would impoverish the dataset and if possible, was to be avoided.

A second alternative option was to impute a value to the missing values of the ratings. However, before opting for this solution, some checks were performed.

A frequently used technique in statistics for imputation of missing values is called mean imputation or mean substitution. It consists of substituting the unknown values by the mean of the distribution of the known values.

To see whether it was reasonable to impute as overall rating of a player, its team's average (average rating of the known players), some statistical tests were completed.

The knowledge of the standard deviations of the distribution of the players' ratings within a team shed some light on whether the substitution made sense or not. Therefore, the following steps were taken:

1. segregation of the matches with full data
2. performing a join with the most relevant rating based on the nearest backwards looking date to the match date
3. calculating the average players' rating for both home_team and away_team, calculating the standard deviation for both home_team's and away_team's players' ratings
4. Running the 'describe' function to obtain summary descriptive statistics.

It was found that the average players' rating of a football team (with all 11 players present) based on the average of the team players' overall rating before a given match was 74.55.

As it can be seen in fig.4.5 below, it was also found that the mean of the standard deviations for the distribution of players' ratings for a football team was around 3.66.

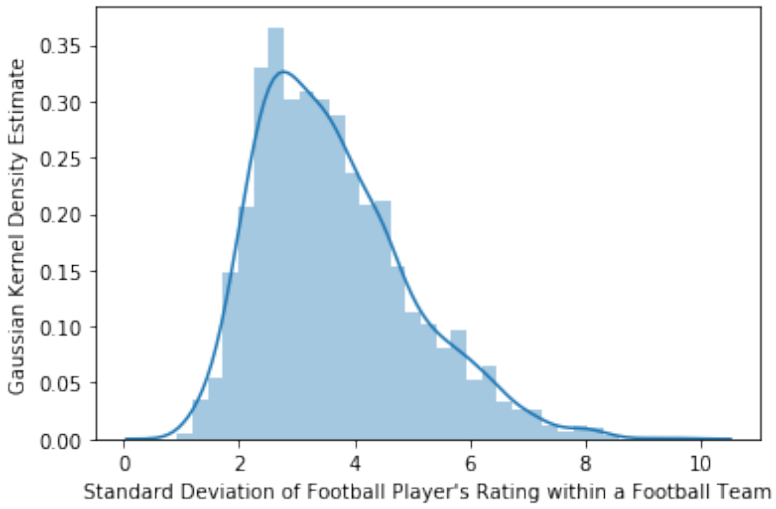

Fig.4.5: Standard deviations for the distribution of players' ratings within a football team.

The standard deviations show that the dispersion of data is moderately low, hence the imputation of missing football players' ratings with the average of the rest of the players' ratings present per team seemed as a strong option moving forward. The distribution of the standard deviation for the distribution of players' ratings seemed to follow a Poisson distribution with most values being skewed to the left.

It was also decided to run further statistical tests to check whether a football team's players ratings distribution was normal. If the statistical distribution of the ratings of the players of a team followed a Gaussian distribution, then some assumptions could be made. The Anderson Darling (AD) test was used to check the normality of the data. In the AD test, two hypotheses are considered: H_0 (data sampled from a Gaussian population) and H_A (data sampled from a non-Gaussian population). The AD test produces a parameter called the p-value. If the p-value is greater than 0.05, then the H_0 hypothesis is accepted. Normality of the distribution of the ratings of the players for a team was tested, by finding the p-values shown in the fig.4.6 below.

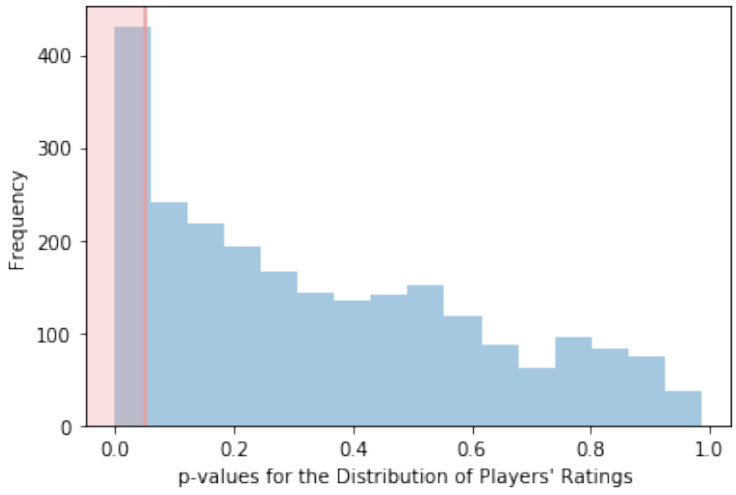

Fig.4.6: p-values for the distribution of players' ratings (Anderson-Darling test).

As it may be observed, many of the p-values are greater than 0.05 and it is therefore hard to reject the null hypothesis of the corresponding ratings following the Gaussian distribution. However, the number of teams with p-value smaller than 0.05 could not be ignored and therefore it was decided to circumvent the fact that no access to all the players' overall rating was available. This was achieved through smart feature extraction as shown in the section that follows.

4.4.3 Feature selection and engineering

One of the major challenges that data scientists face is the optimal representation of the data to best suit the needs of the intended application. It can be observed, for instance, in Kaggle competitions, that sophisticated feature engineering can make the difference between the winning and losing entries. That is to say, that the right extraction of features can make a big difference to the performance of the supervised machine learning model (often playing a greater role than the hyper-parameters optimization of the model).

The use of expert knowledge is often required to improve the forecasting ability of machine learning algorithms. Even though machine learning algorithms are sometimes used with the purpose of avoiding the use of complex rules designed by experts, the use of expert knowledge may be of significant help to identify or define highly informative features.

This section concentrates on the feature extraction (preparation of data to be fed into the Machine Learning models) of the project. Jupyter Notebooks were written to perform the following steps:

1. To ensure the data is clean.
2. To create functions that derive useful features from the original features
3. To test that the engineered features derived do not present any information leakage (as a time-series problem is being analysed).

Feature engineering for the 'virtual' Bundesliga

The 'virtual data' analysed was extracted and curated as explained in sections 4.4.1 and 4.4.2: Data Wrangling. This set of features is called 'virtual' as it was collected from the website (Borjigin, 2019) of the EA Sports video game FIFA. The data-set includes information of the players' attributes in the German Bundesliga seasons (from 2008 to 2016).

In the mentioned website, for each player, there is a set of 33 attributes available. Those 33 attributes are perodically updated. The value of any of those 33 parameters is an integer in the range from 1 to 100. An approach similar to the one taken by Shin and Gasparyan (2014) is followed. As suggested in their paper, the 33 parameters were split into 7 categories as shown by the table 4.1 below.

Table 4.1: Categories and features for 'virtual data' (Shin and Gasparyan, 2014)

Categories:	**Features:**	**# Features:**
Attacking	Crossing, Finishing, Heading Accuracy, Short passing, Volleys	5
Skill	Dribbling, Curve, Free Kick Accuracy, Long Passing, Ball Control	5
Movement	Acceleration, Sprint Speed, Agility, reactions, Balance	5
Power	Shot Power, jumping, Stamina, Strength, Long Shots	5
Mentality	Aggression, Interceptions, Positioning, Vision, Penalties	5
Defending	Marking, Standing Tackle, Sliding Tackle	3
Goalkeeping	GK Diving, GK Handling, GK Kicking, GK Positioning, GK Reflexes	5

To produce the engineered 'virtual' features, the steps defined below were followed:

1. The names of the columns in the data-frame 'bundesliga' were renamed to identify the player and the 33 features above. For example, a possible column name is: agility_home_player_3.
2. Thereafter, the function 'combine_features' which groups player's features into the 7 main skills or categories that make up a player was built. This function obtains the aggregate for each of the 7 main skills for each player before a football match, uses as parameters rows in the Pandas data-frame, and returns a new set of columns according to the description given above (columns): home_player_1_attacking,

home_player_1_skill, etc. The function was designed to be able to aggregate even when NaN values were found (as they were replaced with zeros).

3. It was tested that the function produced the expected results.
4. A function 'obtain_virtual_features' that obtains the 'virtual features' for each team was written. The inputs to this function are the rows of the 'bundesliga' data-frame. The function output for each team is the aggregation of the ratings for the best 4 attacking players' ratings, the best 5 skill players' ratings, the best 5 movement players' rating, the best 5 power players' rating, the best 5 mentality players' ratings, the best 4 defending players' ratings and the best goalkeeper player's rating (in all cases the goal-keeper). The function returns 14 new columns according to the definition of the function: away_top_5_movement_sum, home_top_5_movement_sum, away_top_5_power_sum, home_top_5_power_sum, etc. This was the key step to be able to circumvent the problem of facing missing values as only the best selection of 'n' players was chosen to represent a team according to the 7 categories that define a player (It should be noted that there were only at most 3 missing players per team per football-match and none of them were the goal-keeper).
5. The function was tested to ensure it produced the expected results.

The resulting features from the above steps were the ones shown in Eq 2.1 with the addition that they were all scaled from 1 to 100. Further to this, the difference between each feature of the home team and away team were computed and these resulted in the final set of virtual features (49 features in total). These are listed in table C.1 of Appendix C.

A function 'label_match' to obtain the match label (home win, draw or away win) was produced. The inputs of this function are the number of home and away goals. The function returns a string (H, D, A) that represents the result (home win, draw or away win). This last function was tested and the data-frame 'bundesliga_test' which now included the labels and the engineered features was saved as a csv file.

Feature engineering 'real' Bundesliga

This data-set includes the in-game statistics of the German Bundesliga seasons from 2008 to 2016. To extract meaningful features, some of the features contained in this data-set were found to have a detailed explanation in the dictionary (Football-data.co.uk, 2019).

Some auxiliary functions were designed to help the process of building the 'real' engineered features.

The table A.1 of Appendix A describes the name of those auxiliary functions, their description, the inputs required, and the outputs obtained.

The final set of features produced may be found in table C.2 of Appendix C.

An aggressive manual feature reduction was performed, after finding that these features provide a considerable increase in accuracy compared to the isolated set of features that would be obtained by simply applying the feature extraction functions presented in table A.1 of Appendix A (Hessels, 2018, p.22).

4.5 Machine learning

4.5.1 Automated machine learning

The software library, auto-sklearn, was initially used. This software library provides an automated solution to supervised machine learning problems.

Auto-sklearn uses Bayesian optimization for the fine-tuning of the various hyper parameters found in many machine learning algorithms.

Despite the various positive reviews and mentions in academic papers, the results obtained with this library did not seem promising after two trial runs.

The initial run consisted of a one-hour trial to see whether the software package was working as expected. After this one hour, auto-sklearn was able to achieve an accuracy of 50%. Following this successful trial, the full exploitation of this library started with a second trial where the running time was set to be 72 hours. Unfortunately, after 60 hours of initializing auto-sklearn, it ran into memory and storage difficulties and thus put at halt the optimization process for this trial. This information was displayed (fig.4.7) in an error log produced by the Jupyter notebook.

```
OSError: [Errno 28] No space left on device
Call stack:
  File "/usr/local/anaconda3/lib/python3.6/runpy.py", line 193, in _run_mo
dule_as_main
    "__main__", mod_spec)
  File "/usr/local/anaconda3/lib/python3.6/runpy.py", line 85, in _run_cod
e
    exec(code, run_globals)
  File "/usr/local/anaconda3/lib/python3.6/site-packages/ipykernel_launche
r.py", line 16, in <module>
    app.launch_new_instance()
  File "/usr/local/anaconda3/lib/python3.6/site-packages/traitlets/config/
application.py", line 658, in launch_instance
    app.start()
```

Fig.4.7: Error log produced for the second trial.

At that stage, the next logical step was to set the same parameters but this time rather than the optimization time being set at 72 hours it was reduced to 48 hours. This should have allowed the program to still optimize for a long enough time so that the parameters were fully optimized, but not too long so that the optimization process failed.

Therefore, the next 48-hour trial was set to begin. To the disappointment of the author, the accuracy obtained was that of 47% (table 4.2), much lower than in the previously achieved trial run with an optimization time of one hour (50%). At this stage, abandoning auto-sklearn seemed to be the best course of action.

Table 4.2:

Algorithm: Auto-sklearn				
Dataset: German Bundesliga season 2015/2016 (virtual)				
Classification report				
	precision	**Recall**	**f1 - score**	**support**
0	0.47	0.26	0.34	100
1	0.67	0.03	0.05	71
2	0.47	0.87	0.61	135
avg/total	0.52	0.47	0.39	306

4.5.2 Implementing machine learning models (one step lower in the level of abstraction)

Abstracting away from the optimization problem and choosing the best single machine learning model did not seem to be the best strategy to follow for the remainder of the project. Thus, it was decided to conduct an intensive period of study of machine learning for data scientists. This was conducted by the full reading of Muller and Guido (2018). The rationale behind this decision was that applying an algorithm to a dataset without truly understanding the logic behind the model and the meaning of its different parameters does not usually lead to attaining reasonable and understandable results as perhaps evidenced with the auto-sklearn library.

In the problem of predicting football matches results, the objective is to know for a given football match that has not occurred yet, if the game will result in a home win, a draw or an away win.

It is therefore required to find a model with forecasting abilities for new and unseen football matches.

It is at this point when the concept of underfitting and overfitting comes into play in the machine learning world. It is possible to create complex models able to achieve a 100% accuracy on the training set (the dataset that the machine learning model learns from) however, these complex models have the tendency of overfitting. These models adjust perfectly to the training data but

have a small capacity to generalize, and if a new set of data is used with the model, results may be poor. Using overfitting models is a common mistake that junior data scientists often fall into.

In general, the objective is not getting a model able to achieve 100% accuracy in the training set, but to be able to obtain high accuracies in predicting the outcome of new football matches. The rules that follow an overfitting model are too complex and are not able to generalize well.

The opposite problem is to choose an extremely simplistic model. This is called underfitting. Sometimes, to be able to provide an easy to explain model, the model is simplified to extreme barebones such that the accuracies of predictions which the model produces are poor. This is because the model is not able to capture all the interrelations among the features and the results.

To know whether a machine learning model will perform well in the future with unseen data it is required to evaluate it with a test set.

Hence, the key to the matter is to choose a model that will generalize well to new data. This is achieved by neither underfitting nor overfitting on the machine learning models. This is best portrayed in the below fig.4.8.

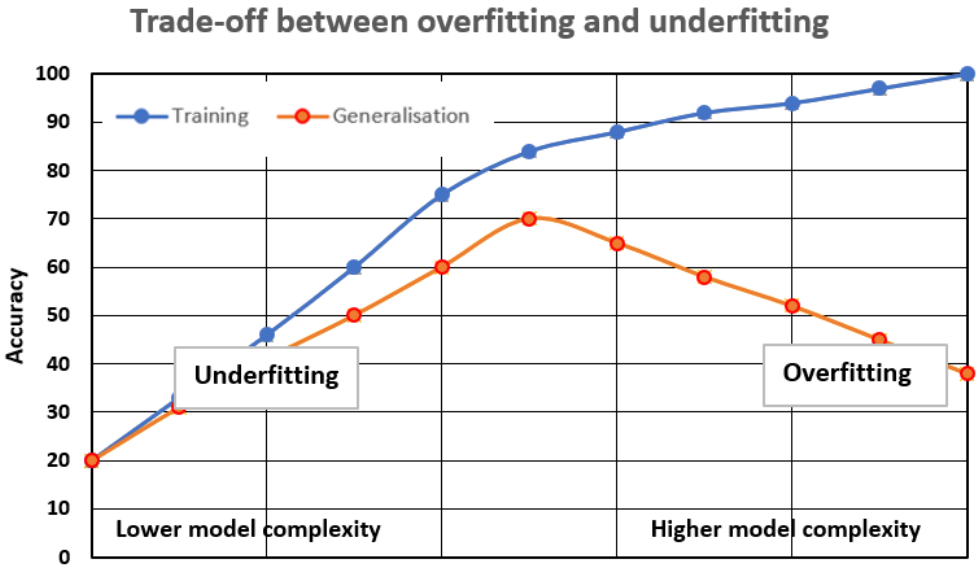

Fig.4.8: The trade-off between overfitting and underfitting.

In practice, this means that to obtain the sweet spot such that the machine learning model is able to generalize well, important parameters intrinsic to each classifier (machine learning model) have to be adjusted. For instance, with the nearest neighbour classifier, adjusting the number of neighbours and the distance measure between each datapoint control the model complexity of this classifier.

This is well portrayed in fig 4.9 where it can be observed that as the parameter 'number of neighbours' is increased, the accuracy achieved on the training set is sacrificed to obtain a better generalizing model (reflected on the higher accuracy achieved in the test data-set.)

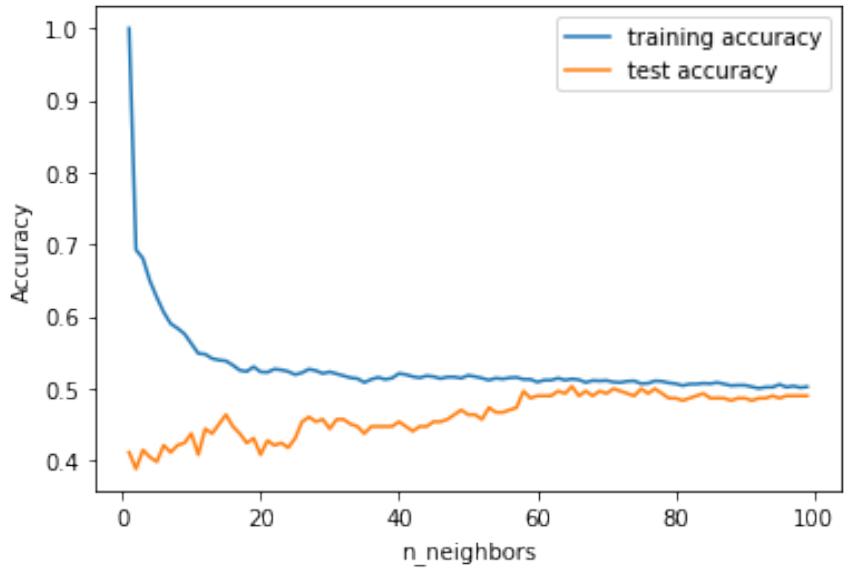

Fig.4.9: Controlling the complexity of the KNN classifier by varying the number of neighbours parameter.

It is worth noting that for some algorithms, due to the manner the scikit-learn library works, the preprocessing of the data (such as scaling the data) has to be performed first, so the scoring metric relevant to the problem in question can be optimized.

The supervised machine learning models mentioned in the summary section of Chapter 3 were employed for the prediction of the outcome of football matches.

4.5.3 Automatic Feature Selection

Contrary to the author's initial beliefs, a greater number of features does not necessarily mean a greater chance of maximizing accuracy. In fact, the number of features is highly correlated to the complexity of the model. It is therefore wise to avoid the temptation of increasing the dimensionality of the data. Then, the reader might wonder why an extensive feature engineering was carried out if increasing the dimensionality of the data leads to a higher chance of overfitting. At this stage, it is necessary to be reminded of the following concepts and facts:

1. It was necessary to do so, as the original set of features contained information which occurred during the match event.

2. The aim was to represent the data in the most optimal way to maximize the learning of the machine learning models.

Apart from the reasons mentioned above, an important step of the machine learning workflow is to reduce the number of features to the most descriptive ones, getting rid of the rest (which might be considered as noise in the model). This often produces simpler models, which tend to generalize better.

In the pipelines designed in this project (a concept which will be discussed in the penultimate subsection of this chapter), univariate statistics as well as iterative feature selection are employed. These reduce the number of features of the model and the chance of overfitting.

For the univariate feature selection, the SelectKBest class was used. This selects (as the name indicates) 'K' number of features with the lowest p-values (meaning that they are likely to be highly related to the match label).

A disadvantage of this approach towards feature selection is that it is completely independent of the machine learning model to be constructed. As a consequence of this, univariate tests are very fast to execute as they do not require training and testing of a machine learning model. In practice, the price to pay for this high speed is that the features chosen are more often than not a sub-optimal choice. Despite this, univariate feature selection may prove useful if there is a large number of features to build the model.

A more computationally expensive method employed was iterative feature selection. In this approach, recursive feature elimination with cross-validation was used. A model is built with all the original features, it is cross validated according to the cross-validation scheme and its performance is recorded. This is successively repeated by eliminating the least informative feature of the model, until the prespecified number of features are left. The model chosen is the one with the highest cross validation score. The concept of cross-validation will be revisited in the following subsection. Recursive feature elimination is illustrated in the fig.4.10.

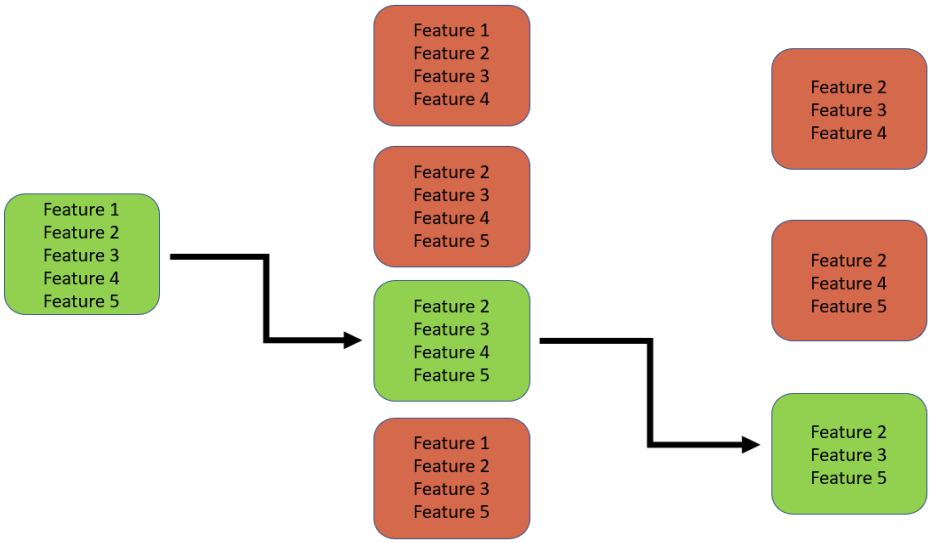

Fig.4.10: Recursive feature elimination (Reproduced from Medium, Feature Selection Methods in Machine Learning, 2019).

It should be noted that recursive feature elimination is a greedy procedure. This means that while it is a heuristic approach to feature reduction, it does not guarantee the optimal combination of

features that would yield the best performance of the scoring metric that it is being tried to maximise. This might be observed if it was a given fact that the combination of features 1, 2, 3 was the best combination that yielded the best accuracy for the given dataset in fig 4.10. This is reflected by the fact that the greedy recursive feature elimination selected feature 2, feature 3 and feature 5 as the best subset of features in fig 4.10 and did not obtain the optimal combination. The author understands that the only way to reach the optimal combinations of features for a given problem and data-set would be an exhaustive search (brute-force). Such approach is not normally used in practice due to its computationally and resource intensive nature.

In the below fig.4.11, it may be seen that according to recursive feature elimination with cross-validation, the best combination of features are just 2 features which provide the highest cross-validation score and hence the best generalization performance.

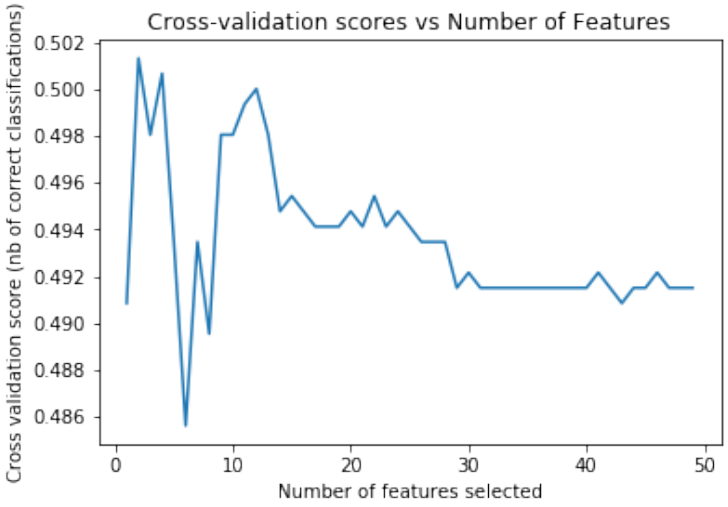

Fig.4.11: Recursive feature elimination with cross-validation (virtual dataset).

As it has been previously mentioned, one of the most important aspects of the machine learning workflow is the selection of features. Above, it has been discussed how to do so with scikit-learn feature selection tools. Now that it is understood how to best represent the data, the next

subsection describes how to select the appropriate hyper-parameter values also known as model fine-tuning.

4.5.4 Evaluating the performance of machine learning models

"Cross-validation is a statistical method of evaluating generalization performance that is more stable and thorough than using a split into a training and a test set." (Muller and Guido, 2018, p.252).

To cross-validate the model's results, different methods can be used to split the data. As mentioned by Yiannakis et al. (2006, p.96), predicting the outcome of a football match is a time-series domain problem. When evaluating a prediction task that belongs to this domain, it is interesting to learn from the past and predict for the future, as one might expect. Hence, traditional methods of cross-validation such as K fold validation, Stratified K-fold cross validation and leave-one-out cross validation are not the most appropriate statistical methods to validate the generalization performance of the models. Forward chaining is the most suitable method to use in this research. In forward chaining, as with all other cross-validation techniques, the dataset is split into the non-test set and the test set. The main difference between forward chaining and the other techniques mentioned is that the non-test set is further split into an actual training set (in which all data is used up to a certain date) where the model is fitted and a validation set (the remaining data past that date up to another date) where the initial parameters selection of the model is evaluated on.

In practical terms, this is how it would be reflected in the data-set: given the first season in the non-test set (training set), what is the expected outcome of the future matches in the following season (validation set)? Now, given the first two seasons in the non-test set (note that the last season of these two was the previous validation set), what is the expected outcome of the future matches in the following season (validation set)? This happens sequentially up until the last season in the non-test dataset is reached as the last validation set. This is best illustrated in the fig. 4.12 and 4.13 below.

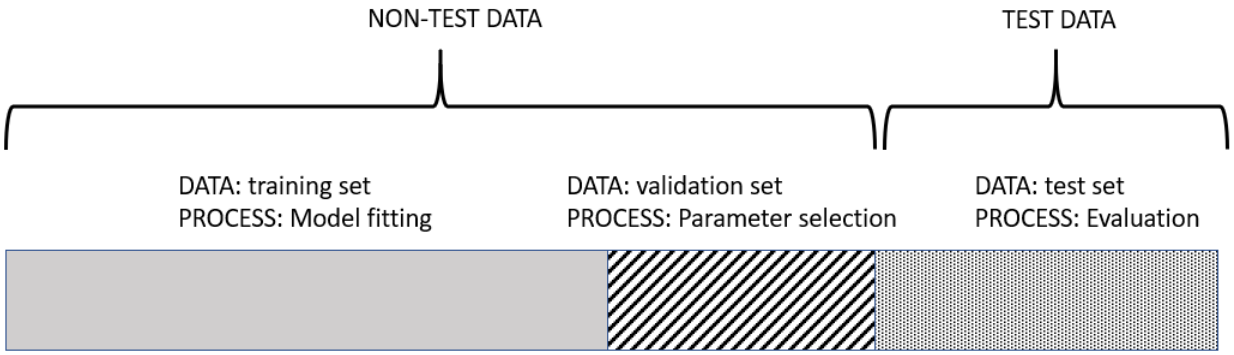

Fig.4.12: Test data and non-test data.

Using forward chaining (fig.4.13), it is required to train 'n' models (depending on how many splits the practitioner specifies) rather than one. Even though several models (with different training data) are trained, the purpose is to check the generalization performance of a machine learning model with a specific set of hyper-parameters. This results in a high computational cost. Again, in machine learning a trade-off situation is found, trying to achieve the fullest optimization is always accompanied by associated computational costs which usually translate into an increase in computational time.

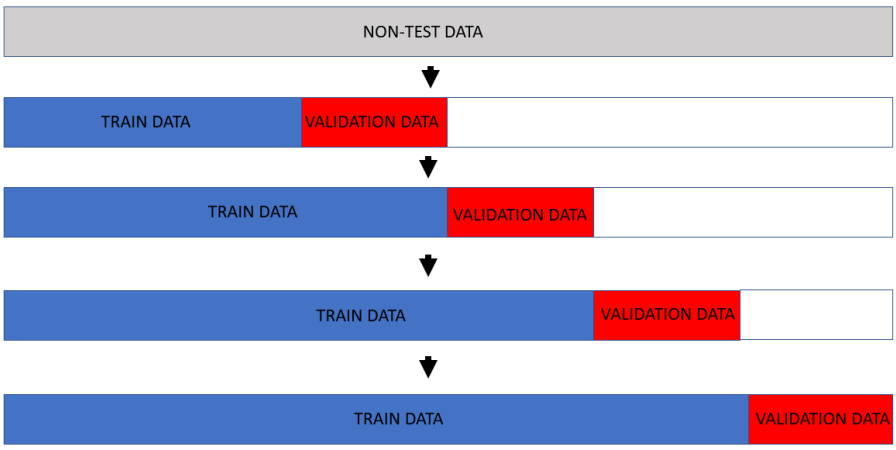

Fig.4.13: Forward chaining.

As outlined above, several models (with different training data and specific set of hyper-parameters) are trained with the purpose of observing their generalization performance.

However, this is only done for a specific set of hyper-parameters and there is no guarantee that those used parameters are the ones that would result in the best generalization performance. The question which naturally arises is about what can be done to find the values of these important hyper-parameters. The techniques called Grid Search and Random Search are used to tackle this issue. In basic terms, grid search is a brute force method where all the different combinations of values provided to the class are tried for the respective parameters of the machine learning model. Usually, trying all possible combinations of hyper-parameters is not a viable machine learning strategy, hence random search is also employed as an alternative. Random search selects random combinations from a predefined grid where the user specifies the limit of search iterations. Random searches are more efficient because not all hyper-parameters are equally important to tune (Bergstra and Bengio, 2012, p302-303).

Sci-kit learn provides an implementation that combines both concepts mentioned above (cross-validation and fine-tuning hyperparameters) and those are the GridSearchCV and RandomizedSearchCV classes. By instantiating the GridSearchCV class and calling the 'fit' method on this object, forward chaining for each combination of the machine learning models' hyper-parameters is performed. After the finalization of the search for the best parameters which resulted in the best forward chaining performance (the parameters which yielded the highest mean validation accuracy), a new model is fitted on the entire non-test dataset with the best parameters and it is stored within this object. The final evaluation is then conducted through the test set score. This score gives an indication of the generalization performance with the chosen parameters for data never seen before.

A similar approach may be carried out for the RandomizedSearchCV class where rather than performing forward chaining for each combination found in the parameter grid, there is an explicit limit of search iterations provided by the user.

4.5.5 Pipelines

With the sheer amount of information presented in the above subsections, it might be tempting to consider machine learning as an obscure art only reserved for the few. As it was mentioned

before, some models require preprocessing steps such as that of scaling. There are also usually rewards in reducing the dimensionality of the data (by either applying feature selection techniques or principal component analysis or even both). Moreover, the right combination of values for the hyper-parameters has to be chosen. All of these processing steps are required to obtain the best generalization performance and extract the full potential of the machine learning models.

To be able to chain together the various steps outlined earlier in this chapter, scikit-learn provides a class called Pipeline. Pipelines are able to encapsulate all of these steps into one single object. The grid search class can be applied to Pipelines to evaluate them. The grid search also allows to search not only for the relevant parameters of each model but also the several possibilities that exist within the previously mentioned preprocessing steps. It therefore allows to significantly amplify the search space for all the different combinations of sequential processing (i.e. scaling data), feature extraction/dimensionality reduction and hyper parameters selection. This class is one of the most convenient ones as it allows to run more processes with less code, thereby minimizing the likelihood of coding a sequence of steps with syntax or logical errors.

4.6 Summary of the chapter

In this chapter the methodology of the project is described. The software engineering methods used for the project's deployment (a variant of scrum and test-driven development) are illustrated. Detailed explanations about the software used, tools employed and machine learning workflow are produced. The steps of data pre-processing and data wrangling are shown and the quality and tidiness issues found in these processes were explained. Despite significant efforts in trying to enlarge the dataset to contain as many football matches as possible, as well as their corresponding features, the subject was settled with two parallel data sources (the virtual set and the real set).

The sub-sections of feature engineering and feature selection follows. With feature engineering, the aim was to represent the data in the most informative way to the machine learning models and this was performed carefully so as to avoid any data-leakage. With regards to feature

selection, it was discussed that having a high number of features does not necessarily mean greater generalization performance. In fact, as the number of features or dimensions grow, the amount of data needed to generalize accurately grows exponentially (Medium, The Curse of Dimensionality! 2019). This is known as the curse of dimensionality.
Univariate feature selection and iterative feature selection methods were introduced in order to tackle this curse (by reducing the features/dimensions of the data) as acquiring more data was an option that had been already explored with little success.

The penultimate part of this chapter discussed the machine learning models' performance evaluation relevant to the time-series problem arisen in this dissertation.

Lastly, constructing the pipelines (which combine all the pre-processing steps mentioned and the hyper-parameter tuning) and deploying them jointly with gridsearch or randomized search to search for the optimal values that yielded the highest generalization performance led to obtain the results seen in the chapter that follows.

5 Presentation and evaluation of results

Pipelines that ran several algorithms and optimized parameters and pre-processing steps were built and applied in parallel to the real and virtual datasets. The models were built using the data from the seasons 2009-2010 to 2014-2015 and tested with the data of the season 2015-2016.

This section provides answers to the following research questions posed in the section 1.2 (problem formulation):

- Which features of the data available have more predicting ability?
- Can data collected from EA Sports (regarding player ratings) outperform real-world historical data in order to predict the outcome of football-matches?
- Which machine learning algorithms produce more accurate predictions? Which parameters and pre-processing steps should be used when using those algorithms to predict football match results?
- Do the models developed improve the predicting capabilities that existing models claim to obtain?

5.1 Which features of the data available have more predicting ability?

The *feature importance* values were derived by training an extra trees classifier composed of 10,000 trees. The values of the feature importance fluctuate between 0 (not correlated with label) and 1 (predicts the label with total certainty). The addition of all the *feature importance* values is the unity.

5.1.1 Virtual set feature importance values

Figure 5.1 below shows for the virtual set, the top 10 most important features obtained. They may be seen in the figure that follows.

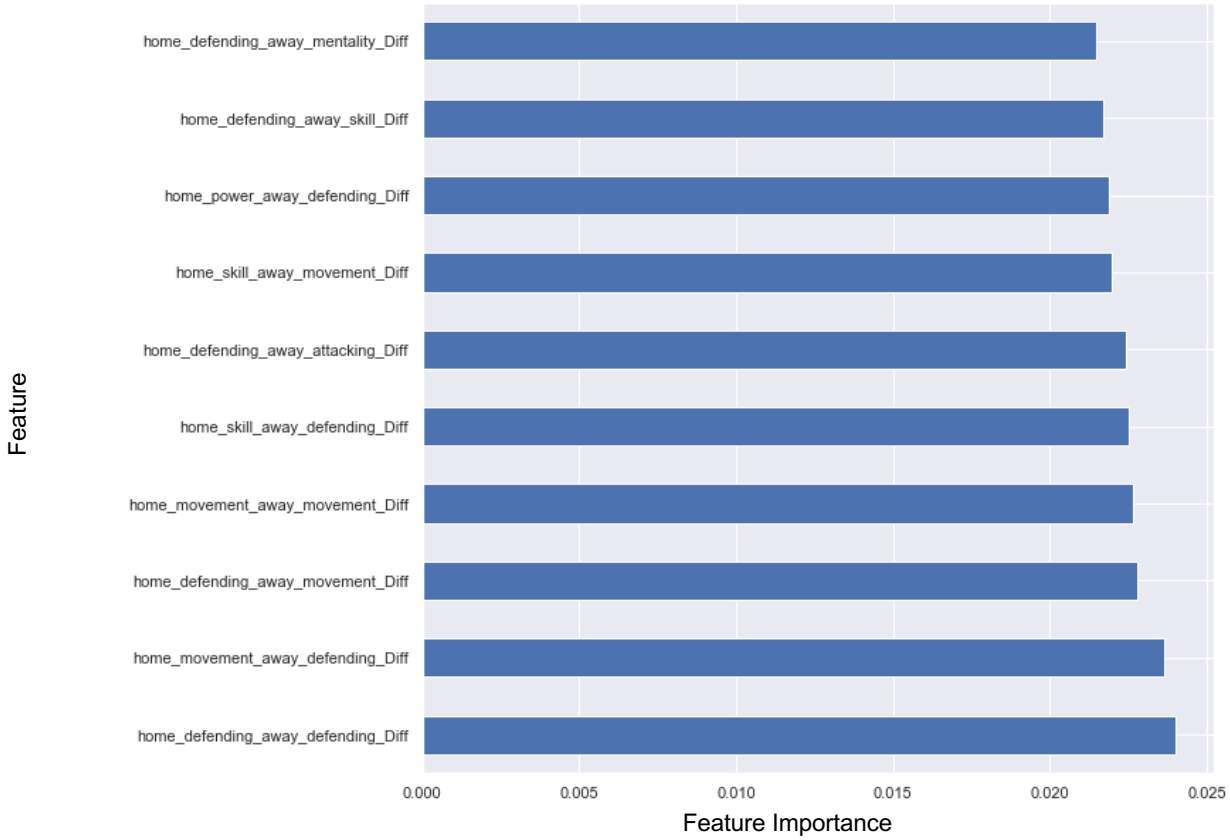

Fig.5.1: *Feature importance* values for the virtual dataset.

It can be first observed that the *Feature importance* values for the top 10 most important features are smaller than 0.025 and fluctuate in a very small range (approximately from 0.022 to 0.024). The values found illustrate the complexity of the problem of predicting football result predictions. The virtual dataset has 49 features, with an average importance of 1/49 = 0.020. As the features with highest *Feature importance* values fluctuate in a very small range (approximately from 0.022 to 0.024) and are very close to the average *Feature importance* (0.020), this means that there is not a strong dominant feature in the dataset that will help to predict the outcome of a football match.

This is further supported by carefully studying the correlation matrix (fig.5.2) for this data-set which shows positive high correlations amongst the features.

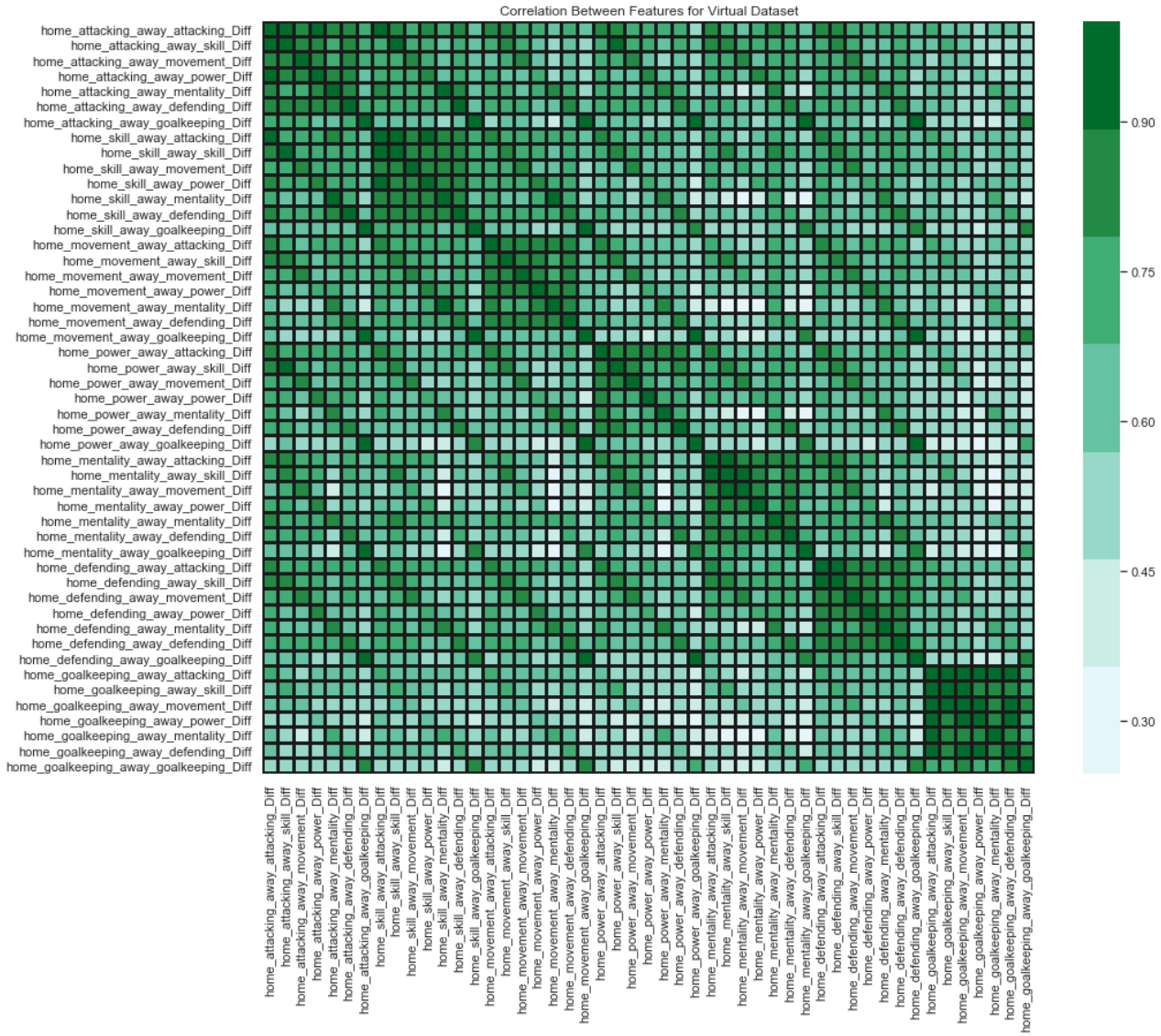

Fig 5.2: *Correlation matrix heatmap* for features in the virtual dataset.

As it may be deduced from the fig.5.1, the home defending and away defending feature difference plays the most significant role when predicting the outcome of a football match. It is said that the best offence (attack strategy) is achieved by having the best defense (having an unbreakable defense). An interesting highlight found was that the difference between the defense of the home team and the away team's attack is not part of the three most important features, it is actually ranked sixth from a total of 49 virtual features.

5.1.2 Real set feature importance values

Figure 5.3 below shows for the real set, the top 10 most important features obtained. They may be seen in the following figure.

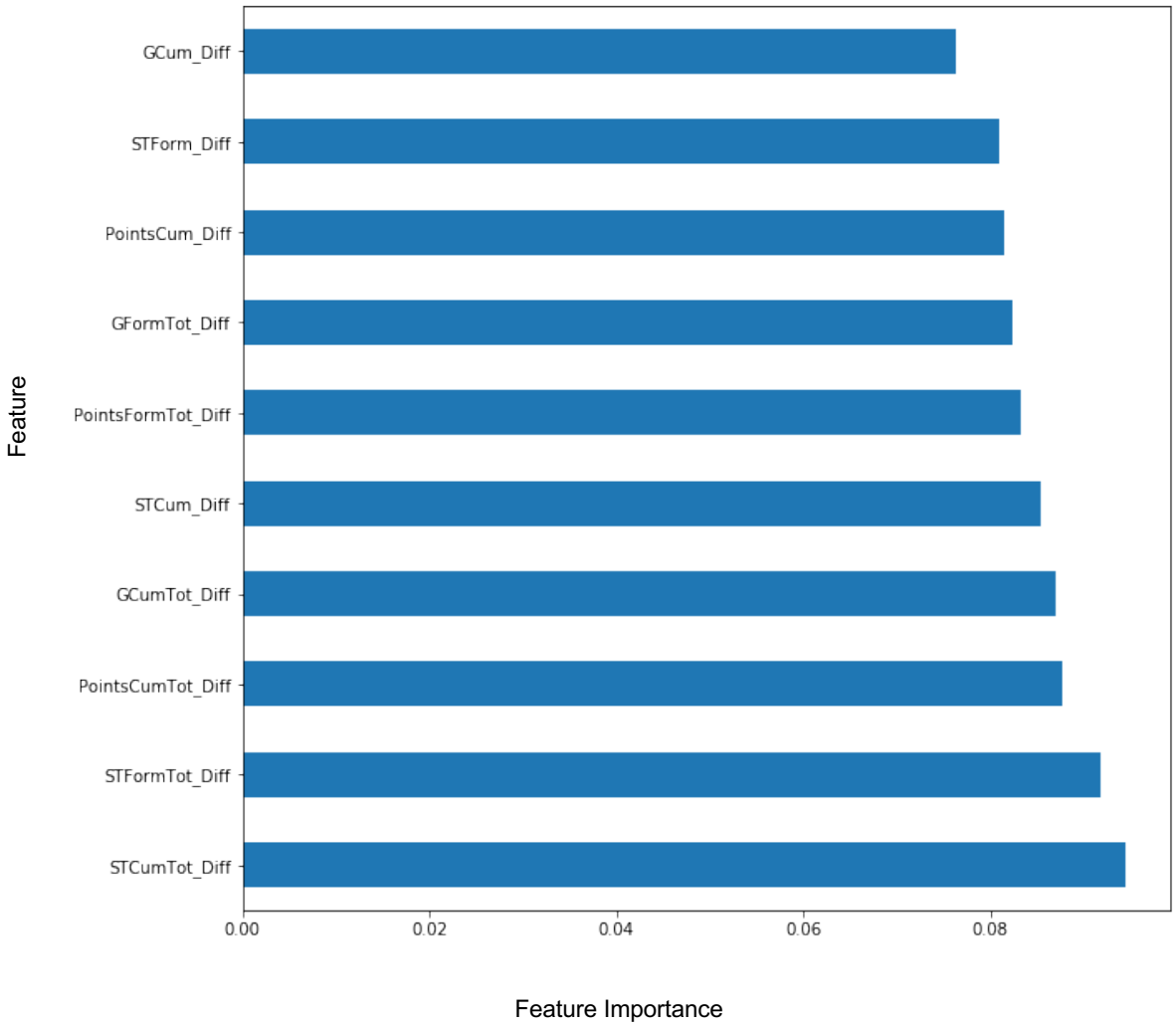

Fig.5.3: *Feature importance* values for the real dataset.

For the real dataset there are only 12 features, with an average importance of 1/12 = 0.083. For this real dataset, the ten features with the highest *Feature importance,* values fluctuate in a very small range (approximately from 0.075 to 0.095) and are also very close to the average *Feature importance* (0.083). This means that for the real dataset there is also not a strong dominant feature that will help to predict the outcome of a football match.

From the fig.5.3 it may be inferred that for the real dataset, the features related to 'Shots on Target' are among the most informative features, however, again, as for the case of the virtual data, no single feature may be considered highly superior to the rest of features.

For illustrative purposes, the correlation between features in the real dataset is shown below in fig 5.4.

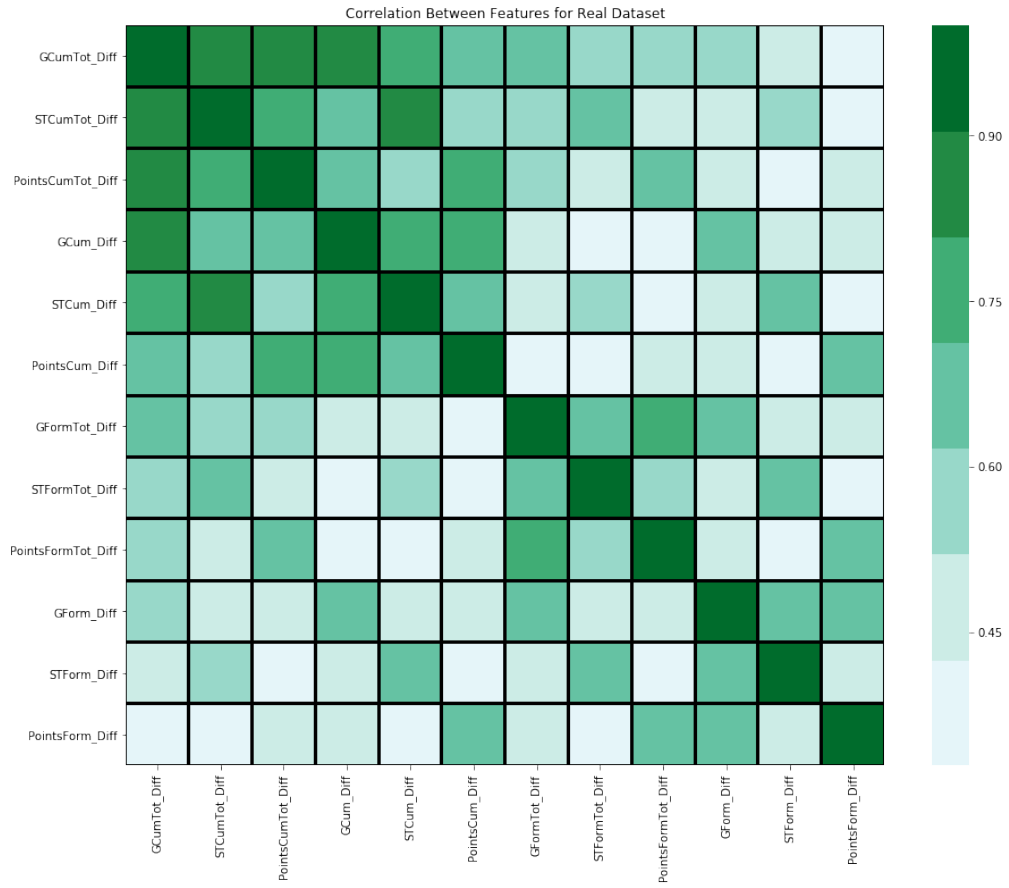

Fig 5.4: *Correlation matrix heatmap* for features in the real dataset.

5.2 Can data collected from EA Sports (regarding player ratings) outperform real-world historical data in order to predict the outcome of football-matches?

To answer this question properly, the concept of the confusion matrix would need to be first explained using an example.

In the table 5.1 below, 'A' means away team wins, 'D' means draw and 'H' means home team wins. There is a total of 306 matches per season, in this example (306 = 27 + 11 + 62 + 13 + 5 + 53 + 16 + 16 + 103). The number of matches properly predicted are 27 + 5 + 103 = 135 (the elements in the main diagonal). The rows represent actual classes, the columns represent the predicted classes. As an example, 62 matches were predicted as H, but they were actually A.

Table 5.1: Confusion matrix for the virtual dataset (random forest classifier).

	Predicted A	**Predicted D**	**Predicted H**
Actual A	27	11	62
Actual D	13	5	53
Actual H	16	16	103

Several metrics may be used to assess the machine learning models. These testing metrics are defined in the table 5.2 below coupled with an explicit example:

Table 5.2: Testing metrics for binary classification.

Recall for Away Wins	$\frac{True\ Positives}{True\ Positives + False\ Negatives} = \frac{27}{27 + (11 + 62)} = 0.27$	Eq.5.1
Precision for Away Wins	$\frac{True\ Positives}{True\ Positives + False\ Positives} = \frac{27}{27 + (13 + 16)} = 0.48$	Eq.5.2
F1-score for Away Wins	$\frac{Precision\ x\ Recall}{Precision + Recall}\ x\ 2 = \frac{0.48\ x\ 0.27}{0.48 + 0.27}\ x\ 2 = 0.35$	Eq.5.3
Accuracy or recall weighted avg	$\frac{Number\ of\ correctly\ predicted\ matches}{Total\ number\ of\ matches} = \frac{27 + 5 + 53}{306} = 0.44$	Eq.5.4

Since the problem of interest concerns the evaluation of multiclass classification, it is required to compute a weighted average over all the classes in order to obtain the weighted recall, weighted precision and the weighted f1-score.

In this project, the main metric used to assess the machine learning models built is the accuracy or weighted recall average.

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ matches}{Total\ number\ of\ matches} \qquad \text{Eq.5.5}$$

Results obtained for accuracy using the different machine learning pipelines (full names of pipelines may be read in the Appendix B) on the test dataset (German Bundesliga 2015/2016 season) have been plotted in fig.5.5 below.

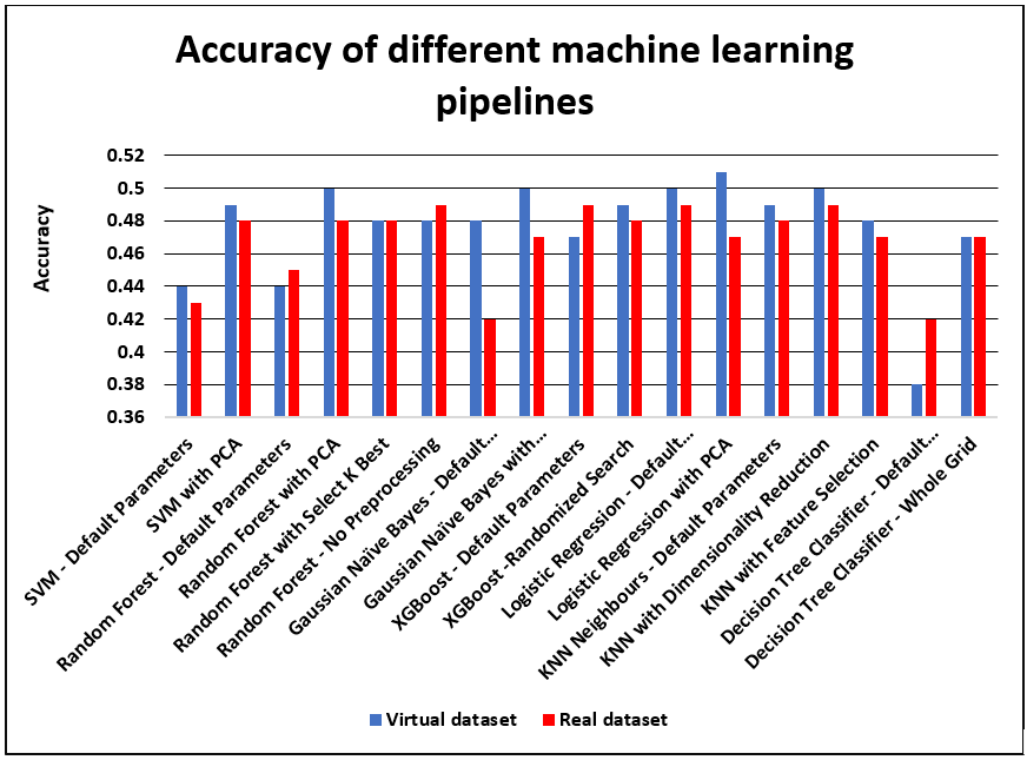

Fig.5.5: A visual summary of the accuracies obtained with the pipelines of machine learning models deployed in the 2015/2016 season of the German Bundesliga.

According to the results obtained, and as it may be seen in fig.5.5, it can be concluded that, in general terms, virtual data collected from EA Sports (blue) outperforms real-world historical data (red) when predicting the outcome of football-matches. In fact, the best accuracy result (0.51) is obtained using the Pipeline of Logistic Regression with PCA with the virtual dataset.

5.3 Which machine learning algorithms produce more accurate predictions? Which parameters should be used when using those algorithms to predict soccer match results?

According to the results obtained, the best pipeline and model was Logistic Regression with Principal Component Analysis with the virtual dataset. Table 5.3 below shows the confusion matrix and classification report obtained using this algorithm.

Table 5.3: Confusion matrix and classification report for the algorithm with the best accuracy

Algorithm: Logistic Regression with Principal Component Analysis				**Dataset:** Virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	43	1	56	**A**	0.51	0.43	0.47	100
Actual D	18	0	53	**D**	0.00	0.00	0.00	71
Actual H	23	0	112	**H**	0.51	0.83	0.63	135
				accuracy			0.51	306
				macro avg	0.34	0.42	0.37	306
				weighted avg	0.39	0.51	0.43	306

The following processing steps and hyper-parameters define the best classifier obtained:

1. Pre-processing steps applied to the virtual non-test data:

- Applying a MinMaxScaler with the values of features ranging from 0 to 1.
- Reducing Dimensionality of the data with Principal Component Analysis and a retained variance of 90%.

2. Hyper-parameters relevant to the Logistic Regression Classifier:

- C=1
- class_weight=None
- max_iter = 10,000
- multi_class= 'auto'
- penalty= 'l2'
- solver = 'newton-cg'

Almost equally well-performing pipelines are the K-Nearest neighbour classifiers (virtual and real datasets) with dimensionality reduction. For the virtual and real data-sets, they obtain an accuracy of 0.50 and 0.49 respectively (the 2nd and 3rd best values for accuracy obtained in the test dataset) respectively.

Surprisingly, XGBoost was not crowned as the championing algorithm, despite its popularity among the data science community to be a de facto superior model for the application of machine learning to tabular data (as is the case in this research).

Another interesting observation drawn is that in most cases, dimensionality reduction either through means of Principal Component Analysis or feature selection resulted in a better prediction performance than running the same set of algorithms without this crucial pre-processing step.

For greater completeness, the full set of results for all pipelines (confusion matrices and the testing metrics in the form of a classification report) may be found in Appendix B.

5.4 Do the models developed improve the predicting capacity that existing models claim to obtain?

Figure 5.6 below illustrates the accuracies obtained by previous researchers in the prediction of football match results and the author's accuracy for the 2015/2016 German Bundesliga Season.

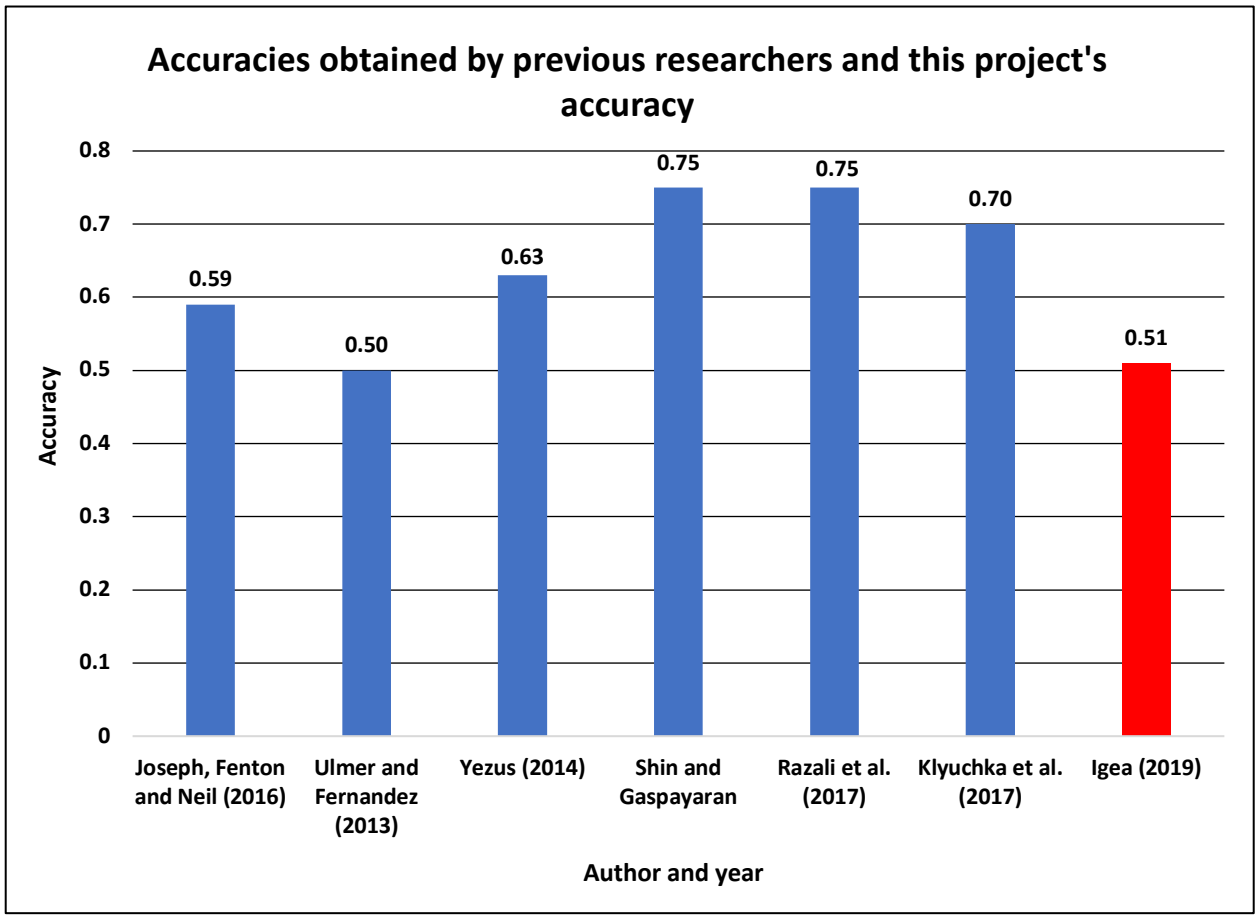

Fig.5.6: A visual summary of the accuracies obtained by previous researchers and the author's.

Shin and Gasparyan (2014) obtained an extremely high accuracy (0.75) (predicting the Spanish La Liga) due to the way the problem of predicting a football match is addressed by them. They state that '*we will have three binary classification problems where in each one we try to distinguish between one of the labels and the other two*'. This means that the accuracy they claim to achieve

is for a binary classification problem rather than the multi-classification problem tackled in this dissertation.

Razali et al. (2017) when predicting matches in the English Premier League state in their abstract that they use K-fold cross validation for testing the accuracy of their prediction model. It has previously been discussed that this method of cross-validation is not the most appropriate one for evaluating the generalization performance in a time-series domain problem. Razali et al. do not mention any works related to feature engineering. This may most likely imply that they were predicting the outcome of a football-match once the match had ended (and all the relevant statistics from the match generated). This practice has not been carried out in this dissertation as the aim was to predict the outcome before a match had begun and not from information available after the match had finished.

Yezus (2014) does not mention any statistical method of evaluating generalization performance so it leaves the reader wondering whether she did in fact split the data into a non-test set and a test set for the English Premier League matches scraped. If she did not, she would not have had an independent dataset (a dataset which was not used to create the model) to evaluate the generalization performance. This is a crucial step when evaluating results derived from applying machine learning methods as the results obtained by not following cross-validation tend to produce an overly optimistic representation of generalization performance.

Joseph, Fenton and Neil (2006) made use of an expert constructed Bayesian net. It was built almost exclusively with subjective judgment and their research only focused on predicting the football matches played by a single team, Tottenham Hotspur Football Club.

From the insights given above, it is thus difficult to compare and benchmark the results obtained in this dissertation with that of other authors due to the shortcomings presented by some authors in their methodology or the general unsuitability to do so (predicting football matches in other leagues).

Moreover, the high accuracies claimed by some authors for what may be thought to be very similar datasets (such as the English Premier league or the Spanish La Liga) to the one employed during this project (the German Bundesliga), have not been achieved despite using similar feature extraction techniques and feature selection methods. This may very well be due to an innate difference found across football-teams in different leagues.

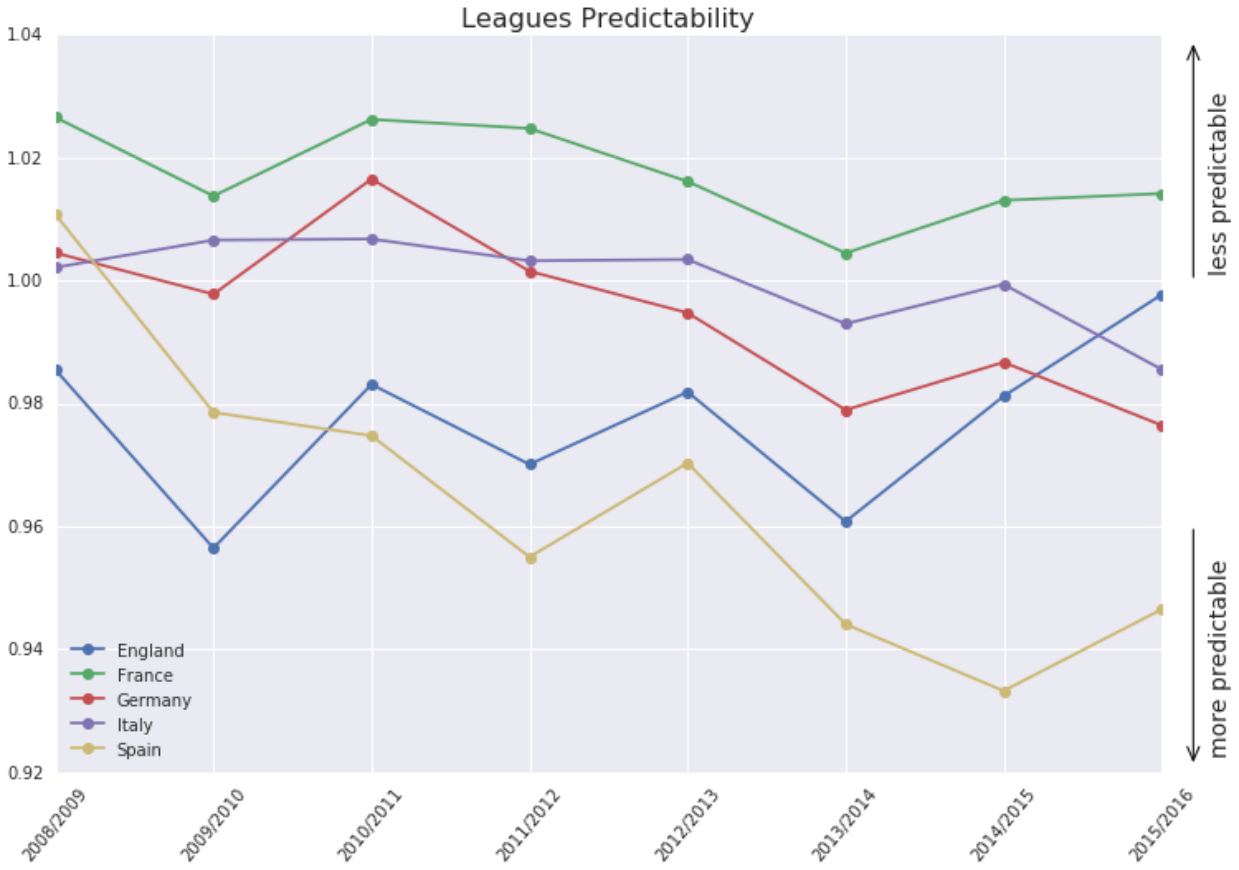

Fig.5.7: Football leagues' predictability. Reproduced from: (Kaggle.com, The Most Predictable League, 2019).

The above argument is clearly depicted by fig.5.7. Figure 5.7 illustrates the leagues' predictability across several seasons. The diagram was produced using book-makers' odds and the entropy concept. While several authors have analysed the English Premier league and they have also achieved higher accuracy results than the ones presented in this project, it is important to note that the German Bundesliga has been more difficult to predict than the English Premier League

for professional sports betting operators. Therefore, the high reported accuracies found in the literature review should not create disappointment to the reader of this project, as significant differences are to be expected as conveyed by the fig.5.7.

Hence, comparing accuracies non-pertaining to the same dataset may not be entirely appropriate to evaluate results since predictability is not only governed by the football league but it is also affected by the football season.

Therefore, what might be a more appropriate benchmark when analyzing the results would be to look at what the industry's sports betting operators' predictions were in the analyzed league during the 2015/2016 season (the used test-set). The confusion matrices and classification reports were obtained for the sports betting operators for this season (which may be found in Appendix D) and their accuracies were extracted from this to build fig.5.8 below.

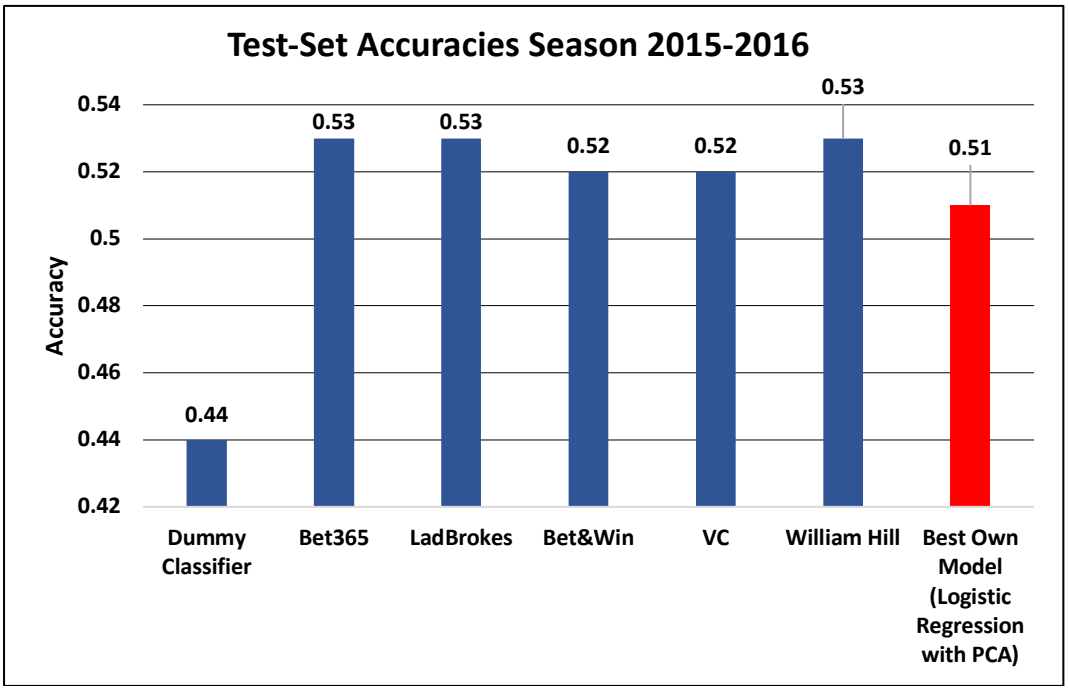

Fig.5.8: Test-set accuracies German Bundesliga season 2015-2016.

Another base-line to assess the results obtained in this dissertation is to determine how much better the produced models perform when compared to a "smart" dummy classifier (which

always predicts a home win, that is, the most frequently observed class in the dataset). This comparison is also shown in fig.5.8 (outermost left bar).

It can be observed that the best pipeline (in red) obtained was able to achieve a 7% positive difference compared to a dummy classifier that produces predictions on the most frequently observed class. This may be taken as an indication that the chosen metric to assess the results obtained (accuracy) is a sensible one for the problem of predicting the outcome of football-matches.

If the class imbalance in the data-set is contemplated, it could be worth considering the f1-score (the harmonic mean between accuracy and precision) as a suitable alternative for the scoring metric to maximise.

However, the fig.5.8 above does not provide a full picture as it only provides an indication of the generalization performance of the best pipeline produced. A more insightful perspective from which results can be analysed is the study of the performance of the obtained pipeline throughout the several seasons used in the non-test data-set (fig.5.9 below). As the cross-validation technique of forward chaining was followed, this is a perfectly acceptable analysis.

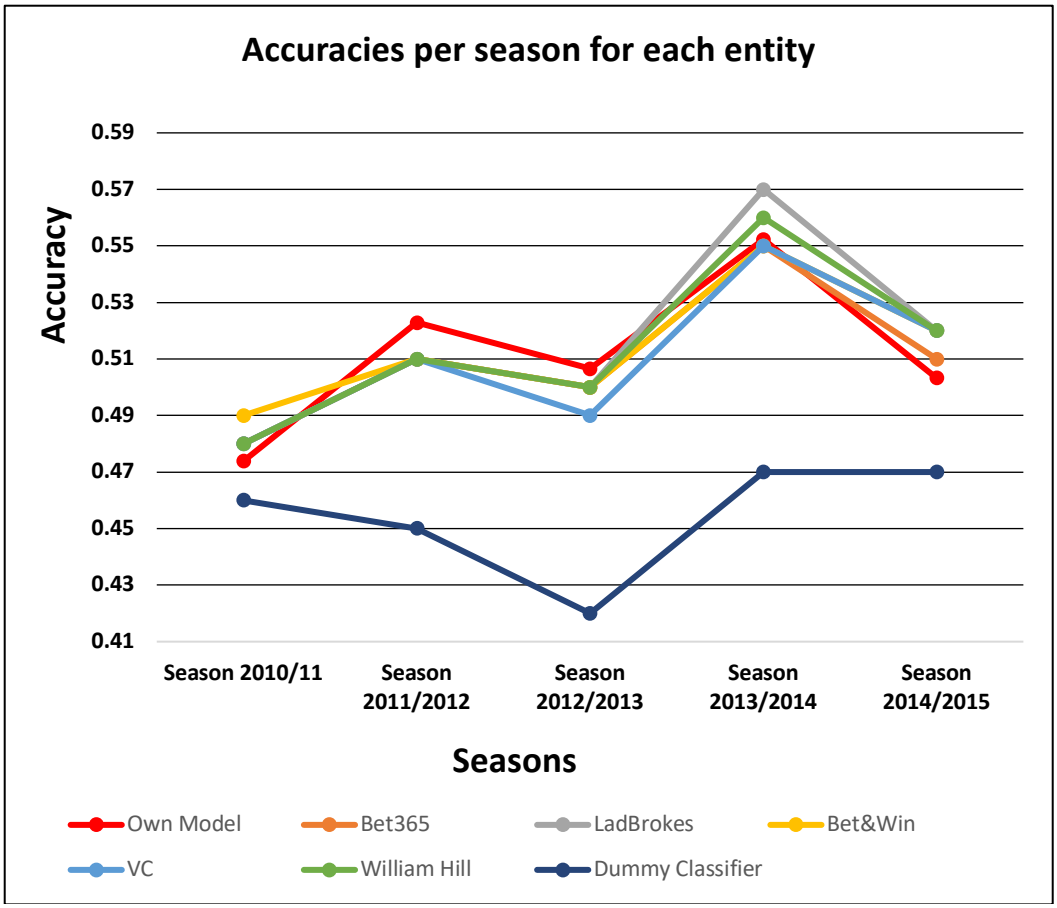

Fig.5.9: Accuracies per season in the German Bundesliga for each entity.

The accuracies plotted for the 'Own Model' (in red) are the results obtained with cross-validation (using forward chaining) for the grid-search of the pipeline with the highest generalization performance (Logistic Regression with Principal Component Analysis for the Virtual Data-set, accuracy obtained: 0.51). Many useful observations may be drawn from fig 5.9:

1. The model designed has been able to outperform the book-makers during the 2011/2012 and 2012/2013 seasons of the German Bundesliga.

2. For the 2013/2014 season, the project's model was able to obtain an equal performance compared to the sports-betting operators. It is worth mentioning that for all predictors, this was the most predictable season found in the data-set. This could have been due to the high

percentage of home-wins (as shown by the 'dummy classifier') and the relatively high number of away-wins compared to draws (as shown in fig 3.1) during this season.

3. The author's model obtained a slightly inferior performance in the 2010/2011 and 2014/2015 seasons compared to the sports betting operators. The 2010/2011 inferior performance was to be expected due to the minimal amount of data that the produced pipeline could be trained on (only one season according to the cross-validation scheme: the 2009/2010 season). It remains to be explored why the author's performance in 2014/2015 slightly suffered when comparing the merits of this pipeline with the other seasons.

5.5 Summary of the chapter

This chapter has presented and evaluated the results obtained by individually answering the questions posed as part of the problem formulation. Additionally, some of the gaps found in the literature review have been indirectly answered.

It was seen that results obtained by authors without the use of forward chaining for cross-validation in the non-test dataset most likely produced overly optimistic results. It has also been explained that it is not clear at this stage whether the authors' machine learning model selection was the optimal one (due to the limited number of models employed in their research).

The chapter that follows presents the conclusions that may be reached from this research and the recommendations about possible steps to be followed to find a superior model able to outperform the sport-betting giants. It is within this sixth chapter that the dissertation is concluded.

6 Conclusions and recommendations for further work

6.1 Conclusions

This dissertation constitutes an initial solid attempt to the prediction of football matches using multi-classification machine learning methods. It has been shown that predicting the outcome of football matches (an unbalanced classification problem) with accuracies higher than the industry benchmarks, is a tough challenge, though not an impossible one, as evidenced in the results section.

It was also learnt that some authors in the existing literature claim results that perhaps do not allow a direct comparison with the results obtained in this project.

The author believes that the final objectives agreed upon with the supervisor have been successfully explored and met. These are stated below:

1. Machine learning models have been built in order to predict the outcome of a sport: football.

2. The models developed have been assessed with respect to the accuracy scoring metric.

3. The research questions, posed as problem formulation have been thoroughly answered and analysed:

- The features with the highest predicting ability were found for both data-sets. It was found that for both separate data-sets, there was not a strong dominant feature.
- By analyzing the results obtained from the deployment of several machine-learning pipelines, it was determined that the data collected from EA Sports (regarding football players' ratings), in this study, possess a higher predicting power than real-world historical data predicting the outcome of football-matches. This is a quite remarkable finding considering that EA Sports' main objective is to bring the passion of football to video-game players and in the process of doing so, EA Sports has (perhaps) inadvertently

created a powerful set of features which can be used to predict the outcome of football matches.

- Logistic Regression coupled with Principal Component Analysis was the pipeline that led to the best generalization performance among the several machine learning pipelines deployed. An interesting finding was that it cannot be predetermined which machine learning algorithm will outperform the others, even if it is known that specific ones have yielded best results for other authors.
- The best model developed was not able to achieve similar accuracies to the models reported in the literature. This may be due to several reasons. One of them, as it was shown earlier is that not only do leagues vary in predictability, but also seasons within each league have different predictability. As the league of choice was the German Bundesliga and no other paper was found regarding the prediction of football-games in this league, it was concluded that an alternative evaluation path should be followed. This other path of evaluation compared accuracies obtained with the ones obtained by the industry giants. This alternative approach to the assessment of results is further supported by the fact that some of the results obtained by other authors in the literature may be considered as not comparable (as validation techniques not appropriate for this domain were employed).

While it was not possible to find an absolute consistent superior model that outperformed the book-makers throughout all seasons with regards to accuracy as the scoring metric, it is firmly believed that with the recommendations that follow, this goal may be achieved.

6.2 Recommendations for further work

6.2.1 Further work on acquiring more data

An overlooked strategy to improve the training of machine learning models is the collection of additional data. Machine learning nowadays is easily accessible through relevant libraries in various programming languages (R, Python, Java, etc.). What often differentiates the players in

the data-science field is the access to valuable data. The Director of Research of Google P. Norvig once said: '*We don't have better algorithms. We just have more data*' (Kdnuggets.com, 2019).

A good analogy is to think about how humans actually learn from experience; the more experience, the more consolidated the knowledge becomes. Data would be the experience in the machine learning models, and therefore, data boosts the power of the machine learning models. The fig.6.1 below supports the above argument.

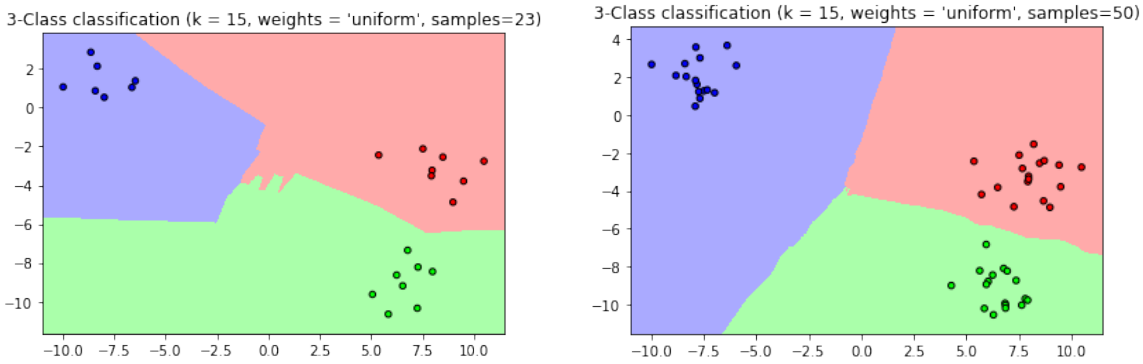

Fig 6.1: Different boundaries learnt by the same Nearest Neighbour Classifier with 23 data-points and 50 data-points.

With more data, clearer boundaries can be defined by the respective models, and hence the scoring metric to maximise or minimize can be improved. Therefore, it is recommended to obtain the football players which played in each match and their respective ratings and attributes for as long as EA Sports has had them (this would constitute a significant challenge on its own). In addition, it is recommended to obtain relevant statistics such as ball possession, goals conceded, goals scored, etc. for each football match played.

6.2.2 Further work on feature extraction and engineering

Once the strategy of acquiring more data has been fully exploited, another interesting place to focus on is the representation of the data via feature extraction and engineering. In this study, the form of the teams was obtained by observing cumulative statistics during the last three

matches. The 'form' features were also extended to include the form of the home team when playing at home and the form of the away team when playing away. Usually, a football team plays one week at home and the following week away. So, if the last three matches while playing home or away for any given football team have to be considered, it may be expected that the team has to play six matches before any relevant statistics are derived. When measuring the form using the last three matches and taking into consideration each team's status (home or away), there is a significant proportion of datapoints (around the first 60 football matches for each season) that do not contain any significant feature values. This happens because the teams have not yet played three football matches while being either home if they are playing home, or away if they are playing away (these values are 0 before any relevant statistics are derived). When conducting principal component analysis (retaining 95 % of the variance) in order to reduce the dimensionality of the real-world data, it is believed that the problem outlined above is depicted in the conglomeration of points seen towards the centre of the below fig.6.2 (orbiting near the origin in what appears to be a thick downwards sloped line).

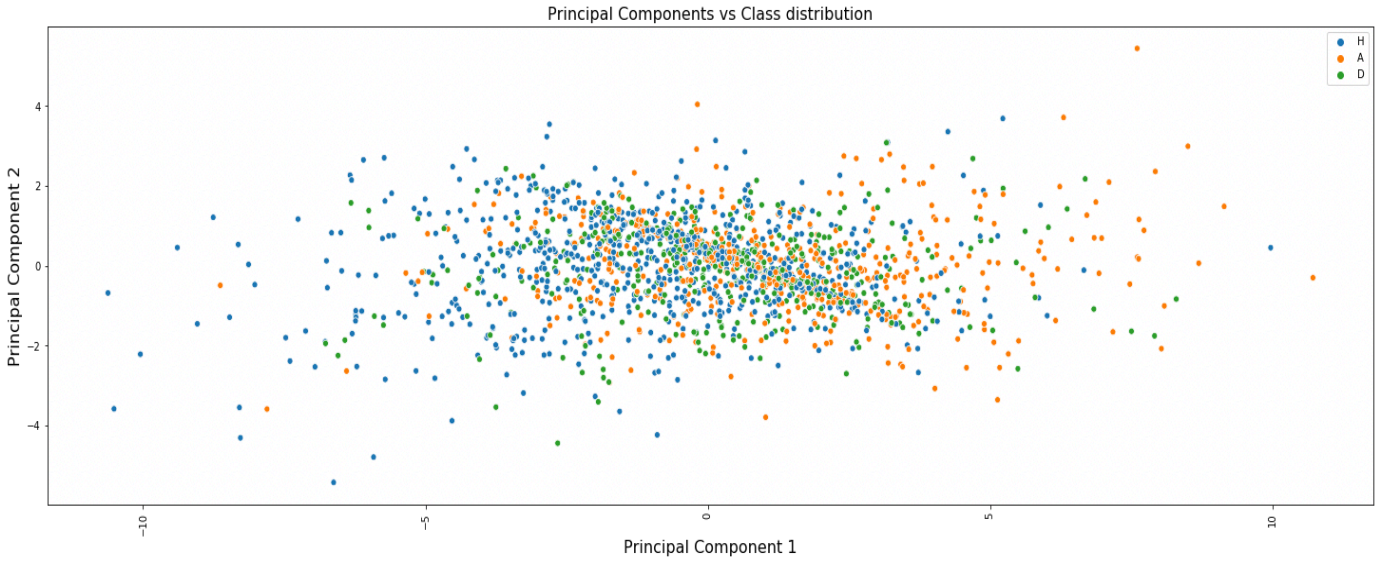

Fig.6.2: Principal Component 2 vs Principal Component 1 for the different datapoints in the Machine Learning Real Dataset.

As previously discussed (Ulmer and Fernandez, 2013, p.2), one possible solution so that the data points become more disperse would be to attribute the value of 'form' for those matches where

there is not enough information (as there is not enough historical data) with a function/routine that considers the number of weeks with information actually available.

Another way to improve the representation of the form/streakiness data of a team is to conduct a similar study as the one performed by Ulmer and Fernandez. According to the authors, the value of weeks, X, to be considered for the calculation of value of form can be defined by a value of X which minimizes the error rate of classification. It is important to note that separate studies would have to be conducted for the different machine learning models currently available in sci-kit learn. Please note that this may be a very laborious task.

It may be expected that by following the above recommendations, a two-dimensional representation of the data would result similar as to the one shown below. Figure 6.3 shows the two-dimensional representation of the virtual dataset whose features were solely extracted from the Sofifa website (Borjigin, 2019).

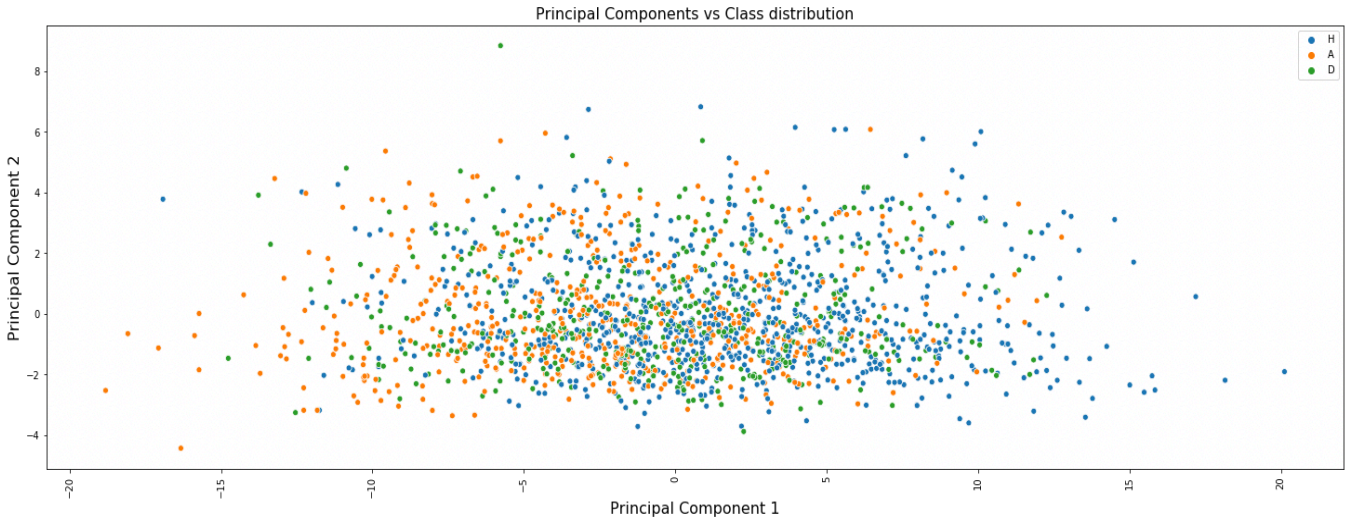

Fig.6.3: Principal Component 2 vs Principal Component 1 for the different datapoints in the Machine Learning Virtual Dataset.

6.2.3 Further work on ensemble methods

'Ensemble models in machine learning combine the decisions from multiple models to improve the overall performance' (Medium, Simple guide for ensemble learning methods, 2019).

With ensemble methods, it is possible to combine the predictive strengths from each machine learning model by reducing variance, noise and bias. The accuracy of the ensemble model will be normally higher than the accuracies obtained using single models.

From a simple perspective, the ensemble methods may be classified as simple and advanced.

If the problem of predicting football matches' results is considered, several machine learning algorithms would be used to forecast the result (home win, draw, away win). The simple ensemble methods would produce as a final result either the mode, the average or the weighted average of the results forecasted by the different algorithms. The weighted average method assigns different weights to the algorithms according to some importance criteria.

Advanced ensemble methods require the selection of a model category. The advanced ensemble method combines several models of the selected category. After the base model is chosen, there are two possible approaches to produce a final result: bagging and boosting.

In the bagging method, samples with replacement are randomly selected from the training set and an algorithm is built for each of those subsets. The final result is calculated by either averaging or voting the results obtained by each of the models built using the subsets.

The boosting method is iterative. The algorithm first gets trained using the complete dataset. In this first run the weight assigned to all the data points is the same. Results are obtained. The data points for which poor predictions were obtained get in subsequent runs higher weights. In this way, several models which form part of a series are created. Each of these models shows high prediction performance for a part of the dataset. A final prediction model is created combining the different models built during the iterative process.

Table 6.1 below shows the advantages and disadvantages of the ensemble methods.

Table 6.1: Advantages and disadvantages of the ensemble methods. (Medium, Simple guide for ensemble learning methods, 2019).

Advantages	Disadvantages
High accuracy predictions. The aggregated model is more stable and robust than each of the models combined. The final ensemble models are able to capture both linear and non-linear connections between features and labels.	The models become complex, and therefore difficult to interpret. High processing time. It is a complex problem to select the models that will be used to create the ensemble.

6.2.4 Further work on hyper-parameter optimization

In this dissertation, grid search (exhaustive search of the search space) and random search (random sampling of the search space) have been used as hyper-parameter optimization strategies.

One of the main advantages of these two strategies is that they can be run in parallel to explore the search space. Nonetheless, they may be considered as inefficient and computationally expensive as the combination of tried values are "blind" to the previously tried ones and hence some valuable information may be lost in the following tries (i.e. unimportant parameters whose values do not seem to affect cross-validation performance). This is especially true for grid search.

It may then become apparent that a more efficient way to sweep the search-space may be implemented taking into account previously searched values. There is a strategy able to do this, it is called Bayesian optimization. The main benefits of using Bayesian optimization are the considerable decrease in the computational time employed for the hyper-parameter optimization and the better generalization performance for the models. A library able to carry

out Bayesian optimization is Hyperopt. Other possible methods also used for hyper-parameter searching are genetic and racing algorithms, particle swarm optimization and coupled simulated annealing (Claesen and De Moor, 2015, p.3).

The introduced optimization methods may prove useful when deploying new pipelines with different engineered features as the speed gains (time saved) in the obtention of results can readily inform the data scientist which features or combination of features look to be the most promising ones.

6.2.5 Automation of the feature obtention

Machine learning algorithms may obtain higher accuracy predictions if highly informative features are accurate and available. Video recordings of football matches are usually available. Some football teams (i.e. Liverpool) are working on a code-base which employs video-tracking and assigns numerical data to all the events happening in the play field (Nytimes.com, 2019). At the time of writing, the Swedish company Signality offers this solution commercially (Signality.com, 2019).

The implementation of these techniques requires teams with high expertise in football, image processing, coding and machine learning. For the machine learning practitioner, this is a very interesting field full of challenges and a whole new array of possibilities for data acquisition.

6.3 Summary of the chapter and lessons learnt

Machine learning is a highly empirical driven field. The results claimed by some authors for very similar datasets (such as the English premier league or the Spanish La Liga) have not been achieved in this study despite using similar feature extraction techniques and feature selection methods. This may very well be due to an innate difference found across football-teams in different leagues as demonstrated earlier, but also to the use of different validation techniques not appropriate for this problem. Small datasets (such as the one used in this research) might

suffice for a proof of concept but for commercially inclined machine learning applications, much more data is required.

Accuracy may not be the most appropriate metric for the prediction of football matches' results, as the dataset is unbalanced. A possible alternative could be to repeat this same study with the scoring-metric of the f1-score (the harmonic mean of the recall and precision). However, it is brought to the attention of the reader that the f1-score metric has not been used in many of the papers used for benchmarking purposes.

It has also been discussed that the models developed for this project, show ample room for improvement with regards to their predictive power. Despite this, the obtained results are highly encouraging. For the 2011/2012 and 2012/2013 seasons in the German Bundesliga, the results obtained were superior to the accuracies achieved by the professional betting agencies.

For the readers driven by financial gain, it might be worth bringing up that achieving a higher accuracy than book-makers does not necessarily mean financial gains. It is a complex research problem to devise a betting strategy that has a positive financial return in the long-term. A starting point may be constructing a more accurate model than the book-makers, but even if a superior model is found, this does not mean that profits can be generated.

As a matter of fact, Kaunitz, Zhong and Kreiner (2017) were are able to find an inefficiency in the sports betting market. Having thoroughly validated their strategy, they put it to implementation. According to them, when the book-makers noticed their outlying behavior (generating consistent profits) some of their transactions started to get blocked or had to be manually approved. In some cases, they reported a transaction limit lower than the initial amount betted. It may be concluded from their paper that the house always wins.

6.4 Project's recapitulation

The use of data science techniques to predict the results of football games has become increasingly popular. However, forecasting the result of a match constitutes a difficult challenge. This is due to the significant random component of the number of goals scored by each team and because of the difficulty in predicting a draw.

Abundant attempts have been made to produce machine learning models for the prediction of football results, but it is difficult to benchmark these results with the ones obtained in this project. This is due to the shortcomings presented by some authors in their methodology and because of the varying predictability of the different football leagues and their seasons.

Nevertheless, it was shown that predicting the outcome of football matches (an unbalanced classification problem) with accuracies higher than the industry benchmarks, while a challenging task, is not an impossible one, as evidenced by the results obtained.

There remain many more unanswered questions arising from this research (they may be found in Appendix E) and it is hoped that one day, future researchers are able to find answers to these intriguing questions.

References

Anon, (2019). Neuron Structure and Function. University of Newcastle. Australia. [online] Available at: https://www.studocu.com/en/document/university-of-newcastle-australia/psychology-introduction-2/lecture-notes/neuron-structure-and-function/1052390/view [Accessed 9 June 2019].

Aoki R., Assuncao R. and Vaz de Melo P. (2017). Luck is hard to beat: The difficulty of sports prediction. Department of Computer Science. UFMG. Belo Horizonte, Brazil. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1367-1376.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research 13 (2012) 281-305.

Borjigin, K. (2019). *Players FIFA 19 Jul 18, 2019 SoFIFA*. [online] Sofifa.com. Available at: https://sofifa.com/ [Accessed 2 June 2019].

Buursma D. (2011). Predicting sports events from past results - Towards effective betting on football matches. University of Twente. Holland, pp.1-6.

Claesen, M., De Moor, B. (2015). Hyperparameter Search in Machine Learning. MIC 2015: The XI Metaheuristics International Conference. p.14-1 to p.14-5.

Constantinou, A., Fenton, N. and Neil, M. (2012). pi-football: A Bayesian network model for forecasting Association Football match outcomes. *Knowledge-Based Systems*, 36, pp.322-339.

Dertat, A. Medium. (2019). *Applied Deep Learning - Part 1: Artificial Neural Networks*. [online] Available at: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6 [Accessed 9 June 2019].

Dixon, M. and Coles, S. (1997). Modelling Association Football Scores and Inefficiencies in the Football Betting Market. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 46(2), pp.265-280.

Dubitzky, W., Lopes, P., Davis, J. and Berrar, D. (2018). The Open International Soccer Database for machine learning. *Machine Learning*, 108(1), pp.9-28.

En.wikipedia.org. *Linear regression*. (2019). [online] Available at: https://en.wikipedia.org/wiki/Linear_regression [Accessed 8 June 2019].

En.wikipedia.org. *Reinforcement learning*. (2019). [online] Available at: https://en.wikipedia.org/wiki/Reinforcement_learning [Accessed 6 June 2019].

Fifaindex.com. (2019). *Player Stats Database - FIFA 19 - FIFA Index*. [online] Available at: https://www.fifaindex.com/ [Accessed 4 June 2019].

Football-data.mx-api.enetscores.com. (2019). *Football Livescores | Football Betting | Free Bets | Football Results*. [online] Available at: http://football-data.mx-api.enetscores.com/ [Accessed 9 June 2019].

Football-data.co.uk. (2019). Football Betting | Football Results | Free Bets | Betting Odds. [online] Available at: http://www.football-data.co.uk/ [Accessed 10 June 2019].

Hessels, J. (2018). Improving the prediction of soccer match results by means of Machine Learning. Tilburg University, The Netherlands.

Highlights, M., Cup, F., Kits, F., Kits, 2., 1, F., Updates, F., Updates, M., Open, A., Open, F., Open, U., 2019, S., Cowboys, D., Eagles, P., Rams, L., United, M., Barcelona, F., Madrid, R., Saint-Germain, P., Munich, B., Milan, A., Roma, A., City, L., United, N., Money, S., Sports, O., SPORTS, M., 2018, F., Warriors, G., Lakers, L. and Cavaliers, C. (2019). *25 World's Most Popular Sports (Ranked by 13 factors)*. [online] TOTAL SPORTEK. Available at: https://www.totalsportek.com/most-popular-sports/ [Accessed 1 July 2019].

Hill, I. (1974). Association Football and Statistical Inference. *Applied Statistics*, 23(2), p.203.

Igea, A. (2019). Machine Learning Algorithms for Sports' Results Prediction – Individual Project Report. University of Strathclyde, Glasgow.

Joseph, A., Fenton, N. and Neil, M. (2006). Predicting football results using Bayesian nets and other machine learning techniques. *Knowledge-Based Systems*, 19(7), pp.544-553.

Kaggle.com. (2019). Complete FIFA 2017 Player dataset (Global). [online] Available at: https://www.kaggle.com/artimous/complete-fifa-2017-player-dataset-global [Accessed 11 June 2019].

Kaggle.com. (2019). European Football Games. [online] Available at: https://www.kaggle.com/waterchiller/european-football-games [Accessed 11 June 2019].

Kaggle.com. (2019). FIFA 18 Complete Player Dataset. [online] Available at: https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset [Accessed 11 June 2019].

Kaggle.com. (2019). FIFA 19 complete player dataset. [online] Available at: https://www.kaggle.com/karangadiya/fifa19 [Accessed 11 June 2019].

Kaggle.com. (2019). Football Events. [online] Available at: https://www.kaggle.com/secareanualin/football-events [Accessed 11 June 2019].

Kaggle.com. (2019). The Most Predictable League | Kaggle. [online] Available at: https://www.kaggle.com/yonilev/the-most-predictable-league [Accessed 14 Aug. 2019].

Kaggle.com. (2019). World Soccer - archive of soccer results and odds. [online] Available at: https://www.kaggle.com/sashchernuh/european-football [Accessed 11 June 2019].

Kaunitz, L., Zhong, S., Kreiner, J. (2017). Beating the bookies with their own numbers – and how the online sport betting market is rigged. Universities of Tokyo, Monash and Sao Paulo.

Kdnuggets.com. (2019). In Machine Learning, What is Better: More Data or better Algorithms. [online] Available at: https://www.kdnuggets.com/2015/06/machine-learning-more-data-better-algorithms.html [Accessed 18 Aug. 2019].

Klyuchka, Y., Cherednichenko, O., Vasylenko, A. and Yakovleva, O. (2017). Forecasting the results of football matches on the Internet based information. *Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies*, 0(55), pp.51-59.

Leung, C. and Joseph, K. (2014). Sports Data Mining: Predicting Results for the College Football Games. *Procedia Computer Science*, 35, pp.710-719.

Maher, M. (1982). Modelling association football scores. *Statistica Neerlandica*, 36(3), pp.109-118.

Mathien, H. Kaggle.com. (2019). *European Soccer Database*. [online] Available at: https://www.kaggle.com/hugomathien/soccer [Accessed 3 June 2019].

Medium. (2019). *Applications of Reinforcement Learning in Real World*. [online] Available at: https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12 [Accessed 8 June 2019].

Medium. (2019). Feature Selection Methods in Machine Learning. [online] Available at: https://medium.com/@sagar.rawale3/feature-selection-methods-in-machine-learning-eaeef12019cc [Accessed 10 June 2019].

Medium. (2019). Is Optimizing your Neural Network a Dark Art?. [online] Available at: https://medium.com/autonomous-agents/is-optimizing-your-ann-a-dark-art-79dda77d103 [Accessed 10 June 2019].

Medium. (2019). Simple guide for ensemble learning methods. [online] Available at: https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2 [Accessed 10 Aug. 2019].

Medium. (2019). The Curse of Dimensionality!. [online] Available at: https://medium.com/diogo-menezes-borges/give-me-the-antidote-for-the-curse-of-dimensionality-b14bce4bf4d2 [Accessed 10 Aug. 2019].

Moroney M. (1956). Facts from figures, 3rd edition. Penguin: London.

Muller, A. and Guido, S. (2018). Introduction to Machine Learning with Python. O'Reilly Media.

Nielsen, M. (2019). Neural Networks and Deep Learning. [online] Neuralnetworksanddeeplearning.com. Available at: http://neuralnetworksanddeeplearning.com/chap2.html [Accessed 9 June 2019].

Noun Project. (2019). Noun Project. [online] Available at: https://thenounproject.com/search/ [Accessed 10 June 2019].

Nytimes.com. (2019). *How Data (and Some Breathtaking Soccer) Brought Liverpool to the Cusp of Glory*. [online] Available at: https://www.nytimes.com/2019/05/22/magazine/soccer-data-liverpool.html [Accessed 2 July 2019].

O'Reilly, Safari. (2019). *Scala Machine Learning Projects*. [online] Available at: https://www.oreilly.com/library/view/scala-machine-learning/9781788479042/e5456dbe-d0f4-47e8-b175-a5a7291ea420.xhtml [Accessed 8 June 2019].

Razali, N., Mustapha, A., Yatim, F. and Ab Aziz, R. (2017). Predicting Football Matches Results using Bayesian Networks for English Premier League (EPL). *IOP Conference Series: Materials Science and Engineering*, 226, 012099.

Reep, C., Pollard, R. and Benjamin, B. (1971). Skill and Chance in Ball Games. *Journal of the Royal Statistical Society. Series A (General)*, 134(4), p.623.

Rue, H. and Salvesen, O. (2000). Prediction and Retrospective Analysis of Soccer Matches in a League. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(3), pp.399-418.

Sawe, B. (2019). *The Most Popular Sports in the World*. [online] WorldAtlas. Available at: https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html [Accessed 1 July 2019].

Shin, J. and Gasparyan, R. (2014). A novel way to Soccer Match Prediction. Stanford University, pp.1-5.

Signality.com. (2019). Performance, Signality. [online]
Available at: https://www.signality.com/performance/ [Accessed 6 Aug. 2019].

Statinfer.com. (2019). *204.6.8 SVM: Advantages Disadvantages and Applications – Statinfer*. [online] Available at: https://statinfer.com/204-6-8-svm-advantages-disadvantages-applications/ [Accessed 9 June 2019].

Tavakol, M., Zafartavanaelmi, H., and Brefeld, U. (2016). Feature extraction and aggregation for predicting the Euro 2016. Universities of Luneburg and Darmstadt, p.1-7.

Ulmer, B. and Fernandez, M. (2013). Predicting Soccer Match Results in the Premier League. School of Computer Science. Stanford University, pp.1-5.

Vaibhavi J. MarkTechPost. (2019). *How reinforcement learning can help in solving real-world problems? | MarkTechPost*. [online] Available at: https://www.marktechpost.com/2019/04/22/how-reinforcement-learning-can-help-in-solving-real-world-problems/ [Accessed 9 June 2019].

Yezus, A. (2014). Predicting outcome of soccer matches using machine learning. Mathematics and Mechanics Faculty. Saint-Petersburg State University, pp.1-12.

Yiannakis, A., Selby, M., Douvis, J. and Han, J. (2006). Forecasting in Sport. *International Review for the Sociology of Sport*, 41(1), pp.89-115.

Wunderlich, F. and Memmert, D. (2018). The Betting Odds Rating System: Using soccer forecasts to forecast soccer. *PLOS ONE*, 13(6), p.e0198668.

Appendix A

Table A.1: auxiliary functions

Function	Description	Inputs	Outputs
get_n_last_matches	Obtains the last 'n' matches for a given team	Dataframe matches' information. home_team_api_id or away_team_api_id. Date from which to look backwards to for the 'n' last matches. Number of matches to look for since the provided date.	For a given team a dataframe containing all relevant information from the previous 'n' matches since the requested date.
get_all_previous_matches	Obtains all the matches played by a given team before and including a given date.	Dataframe matches' information. home_team_api_id or away_team_api_id. Date from which to look backwards for the team's previous matches. home-team matches wanted in the previous matches played? away-team matches wanted in the previous matches played?	Dataframe with all the matches played by a given team before and including the date given.
get_statistic	Obtains relevant statistic for each match of a given team.	Row in the Pandas dataframe. home_team_api_id or away_team_api_id. Statistic to get from each match for a given team.	Relevant statistic for given team in a match(int): an integer with the relevant statistic
obtain_team_shots_on_target	Obtains the relevant team	Row in the Pandas dataframe.	An integer with the relevant

	shots on target for each match.	home_team_api_id or away_team_api_id.	team shots on target.
get_cum_total_statistics	Obtains the cumulative statistics for both teams leading to the current match. (*).	Row in the Pandas dataframe. Dataframe matches' information. Statistic to get from each match for both teams.	Two integers with the relevant statistic of each team: one for the home team, the other for the away team.
get_cum_positional_statistics	Obtains cumulative statistics for both football teams leading to the current match. Note: It only obtains the previous match statistics for home-team matches for the HOMETEAM and the away-team matches for the AWAYTEAM. (*).	Row in the dataframe. Dataframe with matches information. Statistic to get from each match for both teams.	Relevant statistics for both teams: two integers with the relevant statistics one for the home team, the other for the away team.
get_points_for_each_team	Derives the number of points to be awarded for each team that played in a given match: H(3,0) D(1,1) and A(0,3).	Row in the Pandas dataframe	Points to be awarded to home team (int) and points to be awarded to away team (int).
get_positions_agg_ratings	Obtains the aggregate team_players statistics for a given set of team-players of a match.	Row in the Pandas dataframe. home_team_ratings or away_team_ratings. List of players (the column headers). Corresponding Y coordinates of the players above (in	Integers with the goal keeper's rating, the defenders aggregate rating, the midfielders aggregate rating and the forwards

		their respective order).	aggregate rating.

(*) No data leakage occurs as information from current match is not included.

Appendix B

Table B.1

Algorithm: SVM default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	0	0	100	**A**	0.00	0.00	0.00	100
Actual D	0	0	71	**D**	0.00	0.00	0.00	71
Actual H	0	0	135	**H**	0.44	1.00	0.61	135
				accuracy			0.44	306
				macro avg	0.15	0.33	0.20	306
				weighted avg	0.19	0.44	0.27	306

Table B.2

Algorithm: SVM with PCA				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	31	0	69	**A**	0.52	0.31	0.39	100
Actual D	13	0	58	**D**	0.00	0.00	0.00	71
Actual H	16	0	119	**H**	0.48	0.88	0.62	135
				accuracy			0.49	306
				macro avg	0.33	0.40	0.34	306
				weighted avg	0.38	0.49	0.40	306

Table B.3

Algorithm: Random Forest default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	27	11	62	**A**	0.48	0.27	0.35	100
Actual D	13	5	53	**D**	0.16	0.07	0.10	71
Actual H	16	16	103	**H**	0.47	0.76	0.58	135
				accuracy			0.44	306
				macro avg	0.37	0.37	0.34	306
				weighted avg	0.40	0.44	0.39	306

Table B.4

Algorithm: Random Forest with PCA				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	36	0	64	**A**	0.52	0.36	0.43	100
Actual D	15	0	56	**D**	0.00	0.00	0.00	71
Actual H	18	0	117	**H**	0.49	0.87	0.63	135
				accuracy			0.50	306
				macro avg	0.34	0.41	0.35	306
				weighted avg	0.39	0.50	0.42	306

Table B.5

Algorithm: random forest with select K best				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	34	0	66	**A**	0.50	0.34	0.40	100
Actual D	13	0	58	**D**	0.00	0.00	0.00	71
Actual H	21	0	114	**H**	0.48	0.84	0.61	135
				accuracy			0.48	306
				macro avg	0.33	0.39	0.34	306
				weighted avg	0.37	0.48	0.40	306

Table B.6

Algorithm: Random forest no pre-processing				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	33	0	67	**A**	0.48	0.33	0.39	100
Actual D	15	0	56	**D**	0.00	0.00	0.00	71
Actual H	21	0	114	**H**	0.48	0.84	0.61	135
				accuracy			0.48	306
				macro avg	0.32	0.39	0.33	306
				weighted avg	0.37	0.48	0.40	306

Table B.7

Algorithm: Gaussian Naïve Bayes default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**Precision**	**recall**	**f1 - score**	**support**
Actual A	49	24	27	**A**	0.49	0.49	0.49	100
Actual D	21	21	29	**D**	0.30	0.30	0.30	71
Actual H	31	26	78	**H**	0.58	0.58	0.58	135
				accuracy			0.48	306
				macro avg	0.45	0.45	0.45	306
				weighted avg	0.48	0.48	0.48	306

Table B.8

Algorithm: Gaussian Naïve Bayes with dimensionality reduction				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**Precision**	**recall**	**f1 - score**	**support**
Actual A	38	0	62	**A**	0.51	0.38	0.43	100
Actual D	17	0	54	**D**	0.00	0.00	0.00	71
Actual H	20	0	115	**H**	0.50	0.85	0.63	135
				accuracy			0.50	306
				macro avg	0.33	0.41	0.35	306
				weighted avg	0.39	0.50	0.42	306

Table B.9

Algorithm: XG Boost default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	29	9	62	**A**	0.52	0.29	0.37	100
Actual D	12	5	54	**D**	0.21	0.07	0.11	71
Actual H	15	10	110	**H**	0.49	0.81	0.61	135
				accuracy			0.47	306
				macro avg	0.40	0.39	0.36	306
				weighted avg	0.43	0.47	0.41	306

Table B.10

Algorithm: XG Boost randomized search				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	36	0	64	**A**	0.50	0.36	0.42	100
Actual D	14	0	57	**D**	0.00	0.00	0.00	71
Actual H	22	0	113	**H**	0.48	0.84	0.61	135
				accuracy			0.49	306
				macro avg	0.33	0.40	0.34	306
				weighted avg	0.38	0.49	0.41	306

Table B.11

Algorithm: Logistic Regression default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**Precision**	**recall**	**f1 - score**	**support**
Actual A	31	8	61	**A**	0.53	0.31	0.39	100
Actual D	8	11	52	**D**	0.46	0.15	0.23	71
Actual H	19	5	111	**H**	0.50	0.82	0.62	135
				accuracy			0.50	306
				macro avg	0.50	0.43	0.41	306
				weighted avg	0.50	0.50	0.45	306

Table B.12

Algorithm: Logistic Regression with PCA				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	43	1	56	**A**	0.51	0.43	0.47	100
Actual D	18	0	53	**D**	0.00	0.00	0.00	71
Actual H	23	0	112	**H**	0.51	0.83	0.63	135
				accuracy			0.51	306
				macro avg	0.34	0.42	0.37	306
				weighted avg	0.39	0.51	0.43	306

Table B.13

Algorithm: KNN neighbours default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	30	0	70	**A**	0.53	0.30	0.38	100
Actual D	10	3	58	**D**	0.75	0.04	0.08	71
Actual H	17	1	117	**H**	0.48	0.87	0.62	135
				accuracy			0.49	306
				macro avg	0.58	0.40	0.36	306
				weighted avg	0.56	0.49	0.42	306

Table B.14

Algorithm: KNN with dimensionality reduction				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	33	0	67	**A**	0.54	0.33	0.41	100
Actual D	13	0	58	**D**	0.00	0.00	0.00	71
Actual H	15	0	120	**H**	0.49	0.89	0.63	135
				accuracy			0.50	306
				macro avg	0.34	0.41	0.35	306
				weighted avg	0.39	0.50	0.41	306

Table B.15								
Algorithm: KNN with feature selection				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	34	0	66	**A**	0.47	0.34	0.39	100
Actual D	16	0	55	**D**	0.00	0.00	0.00	71
Actual H	23	0	112	**H**	0.48	0.83	0.61	135
				accuracy			0.48	306
				macro avg	0.32	0.39	0.33	306
				weighted avg	0.36	0.48	0.40	306

Table B.16

Algorithm: Decision Tree default parameters				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	29	28	43	**A**	0.35	0.29	0.32	100
Actual D	17	21	33	**D**	0.27	0.30	0.28	71
Actual H	38	30	67	**H**	0.47	0.50	0.48	135
				accuracy			0.38	306
				macro avg	0.36	0.36	0.36	306
				weighted avg	0.38	0.38	0.38	306

Table B.17

Algorithm: Decision Tree whole grid				**Dataset:** Bundesliga 2015/2016 season - virtual				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	35	0	65	**A**	0.46	0.35	0.40	100
Actual D	15	0	56	**D**	0.00	0.00	0.00	71
Actual H	26	0	109	**H**	0.47	0.81	0.60	135
				accuracy			0.47	306
				macro avg	0.31	0.39	0.33	306
				weighted avg	0.36	0.47	0.39	306

Table B.18

Algorithm: SVM default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	4	0	96	**A**	0.44	0.04	0.07	100
Actual D	1	0	70	**D**	0.00	0.00	0.00	71
Actual H	4	2	129	**H**	0.44	0.96	0.60	135
				accuracy			0.43	306
				macro avg	0.29	0.33	0.22	306
				weighted avg	0.34	0.43	0.29	306

Table B.19

Algorithm: SVM with PCA				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	27	0	73	**A**	0.49	0.27	0.35	100
Actual D	12	0	59	**D**	0.00	0.00	0.00	71
Actual H	16	0	119	**H**	0.47	0.88	0.62	135
				accuracy			0.48	306
				macro avg	0.32	0.38	0.32	306
				weighted avg	0.37	0.48	0.39	306

Table B.20

Algorithm: Random Forest default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	44	9	47	**A**	0.47	0.44	0.45	100
Actual D	22	6	43	**D**	0.18	0.08	0.12	71
Actual H	28	18	89	**H**	0.50	0.66	0.57	135
				accuracy			0.45	306
				macro avg	0.38	0.39	0.38	306
				weighted avg	0.41	0.45	0.43	306

Table B.21

Algorithm: Random Forest with PCA				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	41	0	59	**A**	0.45	0.41	0.43	100
Actual D	22	0	49	**D**	0.00	0.00	0.00	71
Actual H	29	0	106	**H**	0.50	0.79	0.61	135
				accuracy			0.48	306
				macro avg	0.31	0.40	0.34	306
				weighted avg	0.36	0.48	0.41	306

Table B.22

Algorithm: random forest with select K best				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	29	0	71	**A**	0.51	0.29	0.37	100
Actual D	12	0	59	**D**	0.00	0.00	0.00	71
Actual H	16	0	119	**H**	0.48	0.88	0.62	135
				accuracy			0.48	306
				macro avg	0.33	0.39	0.33	306
				weighted avg	0.38	0.48	0.39	306

Table B.23

Algorithm: Random forest no pre-processing				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	28	0	72	**A**	0.52	0.28	0.36	100
Actual D	13	0	58	**D**	0.00	0.00	0.00	71
Actual H	13	0	122	**H**	0.48	0.90	0.63	135
				accuracy			0.49	306
				macro avg	0.33	0.39	0.33	306
				weighted avg	0.38	0.49	0.40	306

Table B.24

Algorithm: Gaussian Naïve Bayes default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**Precision**	**recall**	**f1 - score**	**support**
Actual A	47	35	18	**A**	0.45	0.47	0.46	100
Actual D	24	19	28	**D**	0.21	0.27	0.23	71
Actual H	34	37	64	**H**	0.58	0.47	0.52	135
				accuracy			0.42	306
				macro avg	0.41	0.40	0.41	306
				weighted avg	0.45	0.42	0.43	306

Table B.25

Algorithm: Gaussian Naïve Bayes with dimensionality reduction				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**Precision**	**recall**	**f1 - score**	**support**
Actual A	29	0	71	**A**	0.43	0.29	0.35	100
Actual D	19	0	52	**D**	0.00	0.00	0.00	71
Actual H	20	0	115	**H**	0.48	0.85	0.62	135
				accuracy			0.47	306
				macro avg	0.30	0.38	0.32	306
				weighted avg	0.35	0.47	0.38	306

Table B.26

Algorithm: XG Boost default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	40	2	58	**A**	0.49	0.40	0.44	100
Actual D	15	4	52	**D**	0.40	0.06	0.10	71
Actual H	26	4	105	**H**	0.49	0.78	0.60	135
				accuracy			0.49	306
				macro avg	0.46	0.41	0.38	306
				weighted avg	0.47	0.49	0.43	306

Table B.27

Algorithm: XG Boost randomized search				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	35	1	64	**A**	0.47	0.35	0.40	100
Actual D	16	0	55	**D**	0.00	0.00	0.00	71
Actual H	23	0	112	**H**	0.48	0.83	0.61	135
				accuracy			0.48	306
				macro avg	0.32	0.39	0.34	306
				weighted avg	0.37	0.48	0.40	306

Table B.28

Algorithm: Logistic Regression default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**Precision**	**recall**	**f1 - score**	**support**
Actual A	41	1	58	**A**	0.45	0.41	0.43	100
Actual D	22	3	46	**D**	0.60	0.04	0.08	71
Actual H	29	1	105	**H**	0.50	0.78	0.61	135
				accuracy			0.49	306
				macro avg	0.52	0.41	0.37	306
				weighted avg	0.51	0.49	0.43	306

Table B.29

Algorithm: Logistic Regression with PCA				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	38	0	62	**A**	0.44	0.38	0.41	100
Actual D	20	0	51	**D**	0.00	0.00	0.00	71
Actual H	29	0	106	**H**	0.48	0.79	0.60	135
				accuracy			0.47	306
				macro avg	0.31	0.39	0.34	306
				weighted avg	0.36	0.47	0.40	306

Table B.30

Algorithm: KNN neighbours default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	43	1	56	**A**	0.45	0.43	0.44	100
Actual D	23	0	48	**D**	0.00	0.00	0.00	71
Actual H	30	1	104	**H**	0.50	0.77	0.61	135
				accuracy			0.48	306
				macro avg	0.32	0.40	0.35	306
				weighted avg	0.37	0.48	0.41	306

Table B.31

Algorithm: KNN with dimensionality reduction				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	33	0	67	**A**	0.51	0.33	0.40	100
Actual D	13	0	58	**D**	0.00	0.00	0.00	71
Actual H	19	0	116	**H**	0.48	0.86	0.62	135
				accuracy			0.49	306
				macro avg	0.33	0.40	0.34	306
				weighted avg	0.38	0.49	0.40	306

Table B.32

Algorithm: KNN with feature selection				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	35	0	65	**A**	0.44	0.35	0.39	100
Actual D	19	0	52	**D**	0.00	0.00	0.00	71
Actual H	25	0	110	**H**	0.48	0.81	0.61	135
				accuracy			0.47	306
				macro avg	0.31	0.39	0.33	306
				weighted avg	0.36	0.47	0.40	306

Table B.33

Algorithm: Decision Tree default parameters				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	42	28	30	**A**	0.42	0.42	0.42	100
Actual D	15	23	33	**D**	0.30	0.32	0.31	71
Actual H	44	26	65	**H**	0.51	0.48	0.49	135
				accuracy			0.42	306
				macro avg	0.41	0.41	0.41	306
				weighted avg	0.43	0.42	0.43	306

Table B.34

Algorithm: Decision Tree whole grid				**Dataset:** Bundesliga 2015/2016 season - real				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	56	0	44	**A**	0.42	0.56	0.48	100
Actual D	30	0	41	**D**	0.00	0.00	0.00	71
Actual H	46	0	89	**H**	0.51	0.66	0.58	135
				accuracy			0.47	306
				macro avg	0.31	0.41	0.35	306
				weighted avg	0.36	0.47	0.41	306

Appendix C

Table C.1: The 49 engineered features for the 'virtual' Bundesliga	
home_attacking_away_attacking_Diff	2142 non-null float64
home_attacking_away_skill_Diff	2142 non-null float64
home_attacking_away_movement_Diff	2142 non-null float64
home_attacking_away_power_Diff	2142 non-null float64
home_attacking_away_mentality_Diff	2142 non-null float64
home_attacking_away_defending_Diff	2142 non-null float64
home_attacking_away_goalkeeping_Diff	2142 non-null float64
home_skill_away_attacking_Diff	2142 non-null float64
home_skill_away_skill_Diff	2142 non-null float64
home_skill_away_movement_Diff	2142 non-null float64
home_skill_away_power_Diff	2142 non-null float64
home_skill_away_mentality_Diff	2142 non-null float64
home_skill_away_defending_Diff	2142 non-null float64
home_skill_away_goalkeeping_Diff	2142 non-null float64
home_movement_away_attacking_Diff	2142 non-null float64
home_movement_away_skill_Diff	2142 non-null float64
home_movement_away_movement_Diff	2142 non-null float64
home_movement_away_power_Diff	2142 non-null float64
home_movement_away_mentality_Diff	2142 non-null float64
home_movement_away_defending_Diff	2142 non-null float64
home_movement_away_goalkeeping_Diff	2142 non-null float64
home_power_away_attacking_Diff	2142 non-null float64
home_power_away_skill_Diff	2142 non-null float64
home_power_away_movement_Diff	2142 non-null float64
home_power_away_power_Diff	2142 non-null float64
home_power_away_mentality_Diff	2142 non-null float64
home_power_away_defending_Diff	2142 non-null float64
home_power_away_goalkeeping_Diff	2142 non-null float64
home_mentality_away_attacking_Diff	2142 non-null float64
home_mentality_away_skill_Diff	2142 non-null float64
home_mentality_away_movement_Diff	2142 non-null float64
home_mentality_away_power_Diff	2142 non-null float64
home_mentality_away_mentality_Diff	2142 non-null float64
home_mentality_away_defending_Diff	2142 non-null float64
home_mentality_away_goalkeeping_Diff	2142 non-null float64
home_defending_away_attacking_Diff	2142 non-null float64
home_defending_away_skill_Diff	2142 non-null float64
home_defending_away_movement_Diff	2142 non-null float64
home_defending_away_power_Diff	2142 non-null float64
home_defending_away_mentality_Diff	2142 non-null float64
home_defending_away_defending_Diff	2142 non-null float64
home_defending_away_goalkeeping_Diff	2142 non-null float64
home_goalkeeping_away_attacking_Diff	2142 non-null float64
home_goalkeeping_away_skill_Diff	2142 non-null float64
home_goalkeeping_away_movement_Diff	2142 non-null float64
home_goalkeeping_away_power_Diff	2142 non-null float64
home_goalkeeping_away_mentality_Diff	2142 non-null float64

home_goalkeeping_away_defending_Diff	2142 non-null float64
home_goalkeeping_away_goalkeeping_Diff	2142 non-null float64

Table C.2: The 12 engineered features for the 'real' Bundesliga	
GCumTot_Diff	2142 non-null float64
STCumTot_Diff	2142 non-null float64
PointsCumTot_Diff	2142 non-null float64
GCum_Diff	2142 non-null float64
STCum_Diff	2142 non-null float64
PointsCum_Diff	2142 non-null float64
GFormTot_Diff	2142 non-null float64
STFormTot_Diff	2142 non-null float64
PointsFormTot_Diff	2142 non-null float64
GForm_Diff	2142 non-null float64
STForm_Diff	2142 non-null float64
PointsForm_Diff	2142 non-null float64

Appendix D

Table D.1

Bookmaker: B365 – BET365				**Dataset:** Bundesliga 2015/2016 season				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	48	0	52	**A**	0.53	0.48	0.51	100
Actual D	22	0	49	**D**	0.00	0.00	0.00	71
Actual H	20	0	115	**H**	0.53	0.85	0.66	135
				accuracy			0.53	306
				macro avg	0.36	0.44	0.39	306
				weighted avg	0.41	0.53	0.45	306

Table D.2

Bookmaker: LB – LadBrokes				**Dataset:** Bundesliga 2015/2016 season				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	46	0	54	**A**	0.52	0.46	0.49	100
Actual D	22	0	49	**D**	0.00	0.00	0.00	71
Actual H	20	0	115	**H**	0.53	0.85	0.65	135
				accuracy			0.53	306
				macro avg	0.35	0.44	0.38	306
				weighted avg	0.40	0.53	0.45	306

Table D.3

Bookmaker: BW – Bet&Win				**Dataset:** Bundesliga 2015/2016 season				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	46	0	54	**A**	0.53	0.46	0.49	100
Actual D	20	0	51	**D**	0.00	0.00	0.00	71
Actual H	21	0	114	**H**	0.52	0.84	0.64	135
				accuracy			0.52	306
				macro avg	0.35	0.43	0.38	306
				weighted avg	0.40	0.52	0.44	306

Table D.4

Bookmaker: VC				**Dataset:** Bundesliga 2015/2016 season				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	47	0	53	**A**	0.52	0.47	0.49	100
Actual D	22	0	49	**D**	0.00	0.00	0.00	71
Actual H	22	0	113	**H**	0.53	0.84	0.65	135
				accuracy			0.52	306
				macro avg	0.35	0.44	0.38	306
				weighted avg	0.40	0.52	0.45	306

Table D.5

Bookmaker: WH – William Hill				**Dataset:** Bundesliga 2015/2016 season				
Confusion matrix				**Classification report**				
	Predicted A	**Predicted D**	**Predicted H**		**precision**	**recall**	**f1 - score**	**support**
Actual A	46	0	54	**A**	0.54	0.46	0.50	100
Actual D	20	0	51	**D**	0.00	0.00	0.00	71
Actual H	19	0	116	**H**	0.52	0.86	0.65	135
				accuracy			0.53	306
				macro avg	0.36	0.44	0.38	306
				weighted avg	0.41	0.53	0.45	306

Appendix E

Natural extensions for future research arising from this project

Can the best features of both data-sets be used in order to create an even more robust predictive model?

What would happen if we add the English Premier League or any other league(s) to the training data? Would we see any significant improvements in the accuracy of the model?

Can Machine Learning be an approach that is not the optimal?

Should perhaps Monte-Carlo simulations be explored as a more indicative tool of predicting the outcome of football matches?

For readers trying to explore financial gains/opportunities

Hypothesis: To maximise financial returns, one hypothesized strategy would be to bet on those events where your model is very "precise" at predicting one class and assigns a much larger probability of the event to happen compared to the book-makers. When book-makers underestimate the probability of an event occurring, this is considered as having an edge (also known as opportunistic betting).

While analyzing the results, it was noticed that book-makers never predicted a 'Draw' as the most likely outcome of a football match. Could a betting strategy where a user only bets on draws that meet the conditions described above be used to generate consistent profits?

For readers interested in finding superior models to the book-makers

Take all the information from the recommendations for further work section and implement the suggested ideas in your research to increase the accuracy of a model (may be more than one as learnt in ensemble methods) and explore the different leagues and seasons.