FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A data mining approach to predict probabilities of football matches

Tiago Filipe Mendes Neves

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor: João Mendes Moreira

July 14, 2019

A data mining approach to predict probabilities of football matches

Tiago Filipe Mendes Neves

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

July 14, 2019

Abstract

One of the reasons for football being the most popular sport in the planet is its unpredictability. Every day, fans around the world argue over which team is going to win the next game or the next competition. Many of these fans also put their money where their mouths are, by betting large sums on their predictions.

Due to the large amount of factors that can influence the outcome of a football match, it is incredibly difficult to correctly predict its probabilities. With the increasing growth of the amount of money invested in sports betting markets it is important to verify how far the machine learning techniques can bring value to this area. To solve this problem we propose building data-driven solutions designed through a data mining process. The data mining process allows us to build models that can give us predictions according to the data that we feed them.

In this thesis we study how does machine learning algorithms, in the context of data mining, perform when predicting football matches. The problem was approached from several ways: from the classification point of view, analyzing the problem in terms of accuracy, and from the point of view of classification probabilities, analyzing if the calculated probabilities are able to beat the bookmakers.

Following the CRISP-DM methodology, we started by formulating the problem and understanding the tools that were available. After we proceeded with the data acquisition and cleaning phase, gathering data from several sources and storing it in a database. The following step involved getting more familiar with the data, focusing on the exploratory data analysis. From the extracted knowledge we explored how features could be generated and analyzed how to select only the relevant. We then tested several base-learners such as decision trees, k-nearest neighbors and neural networks, followed by several ensemble techniques in order to improve the results. At last, we validated the results obtained in a full season and attempted some techniques in order to verify how the results could be improved.

Keywords: Sports predictions, data mining, machine learning, ensemble learning

Resumo

Uma das razões pela qual o futebol é considerado o desporto mais popular do planeta é a sua imprevisibilidade. Todos os dias, em todo o mundo, os adeptos debatem sobre que equipa vai ganhar o próximo jogo ou a próxima competição. Muitos destes adeptos acabam por apostar largas quantias nas suas previsões.

Devido a um grande número de factores que podem influenciar o resultado de um jogo de futebol, é incrivelmente difícil prever as probabilidades correctamente. Com o crescimento cada vez mais acentuado de dinheiro investido na área de apostas desportivas, torna-se importante perceber até onde os algoritmos de aprendizagem computacional podem trazer valor à área. Para resolver este problema, propomo-nos a construir soluções baseadas em dados, através do processo de mineração de dados. O processo de mineração de dados permite-nos construir modelos que nos conseguem prever as probabilidades com base nos dados que lhe fornecemos.

Nesta tese estudamos qual é a performance dos algoritmos de aprendizagem computacional, no contexto de mineração de dados, para prever jogos de futebol. O problema foi abordado de duas formas diferentes: analisando do ponto de vista de um problema de classificação, isto é, analisar o problema em termos de exatidão da previsão do favorito, e do ponto de vista de probabilidades de classificação, analisando se as probabilidades dos modelos são capazes de ter lucro contra as casas de apostas.

Seguimos a metodologia de CRISP-DM, onde começamos por formular o problema e perceber as ferramentas disponíveis. Após isso, procedemos à fase de angariação e de limpeza de dados, juntando várias fontes e criando uma base de dados. O passo seguinte envolveu a familiarização com os dados, focando-se no processo de exploração e análise dos dados. Do conhecimento extraído foi explorado como é que se poderia gerar features e posteriormente selecionar apenas as relevantes. Testamos depois vários base-learners, nomeadamente árvores de decisão, k-vizinhos mais próximos e redes neuronais. Posteriormente avaliamos vários métodos de ensemble na tentativa de melhorar os modelos. Por fim, validou-se os resultados na temporada mais recente presente nos dados e tentaram-se técnicas com vista à melhoria dos resultados.

Agradecimentos

Gostaria de agradecer em primeiro lugar ao meu orientador de tese, prof. Dr João Mendes Moreira, por me ter dado esta oportunidade de explorar este tópico pelo qual sou apaixonado, e por sempre me orientar na direção certa. Um breve agradecimento a todos os professores que me educaram ao longo dos anos.

Um agradecimento a todos os meus amigos, em especial para o que eu conheci no primeiro dia do meu 1º ano, para o que conseguiu entrar na tropa apesar de ser um grande trapalhão, para o que consegue arranjar tempo para estudar medicina e ser empreendedor de caracóis, para o que foi merecidamente banido no League of Legends e para o que tem um estilo de vida contrário ao meu mas com o qual formei uma excelente equipa nos últimos 5 anos.

Quero agradecer ao meu irmão pelo conhecimento e conselhos que me fornece e à minha avó por ter sido a minha mãe quando a minha mãe não lhe era possível ser.

Por fim, quero agradecer à minha mãe, que fez do seu objectivo de vida a educação dos seus filhos. Objectivo que agora é um sucesso.

Tiago Neves

Contents

List of Figures

List of Tables

Abbreviations

BN	Bayesian Network
CRISP-DM	CRoss Industry Standard Process for Data Mining
k-NN	k-Nearest Neighbor
NN	Neural Network
PCA	Principal Component Analysis
ROI	Return on investment
sklearn	scikit-learn
SPI	Soccer Power Index

Chapter 1

Introduction

1.1 Context

Gambling was always an interesting concept to human beings. If we ask a person if they would trade €1 for €0.95 they would immediately reject the proposal. Being guaranteed to lose money is something that is not usually accepted without being rewarded. In betting, the reward comes from the existing probability of winning money. Even though in the long term more money is lost than won, the human brain is blinded by the prospect of a big win.

For many reasons, including sports being one of the main forms of entertainment in the world, online gambling is growing, as can be seen in the figure 1.1. And with growing markets, opportunities to explore it arise.

Figure 1.1: Predicted growth of the gambling market in the European Union. Source EGBA (2018)

There is a niche part of gambling that is of interest to us: football betting. Unlike other forms of gambling, in football betting, the probabilities are not easily calculated or predefined. When we make a bet in roulette or in blackjack we can know the probabilities a priori, being easily calculable with probabilistic theory. This is not possible in football.

In a complex game such as football, it is also very complex to predict the games. Each game requires specialists opinion and complex models in order to set the initial probabilities. And then there is also an ability for the market to self regulate in a way that the bookmakers make the same profit regardless of the outcome.

With so many factors to take into account when calculating these probabilities, there is the possibility that errors or misjudgments occur. These can be taken advantage of by bettors: if the bookmaker is paying a price that is higher than what the real probability, by the law of large numbers, the bettors will make money in the long term.

1.2 Goals

The goal of the thesis is to evaluate how machine learning models perform in football match predictions, both in accuracy and probability-based approaches, with a special focus on evaluating if models can have the ability to make a profit. The question that we are trying to answer is: how do the predicted probabilities done by machine learning algorithms compare to the bookmakers' predicted probabilities.

There is a need to verify where the state of the art machine learning algorithms stand: are they unable to make correct predictions, getting constantly outsmarted by the market, or are they able to see patterns in data which the human eye can not see and extract profit out of that information.

1.3 Methodology

The methodology that was used in the development of this thesis was based on the CRISP-DM (CRoss Industry Standard Process for Data Mining), with some added details from agile methodologies, as for example, a backlog and prioritization of tasks with the most value associated. The phases of the CRISP-DM methodology follow next.

1.3.1 Phase one: business understanding

As considered by many, including the author Shearer (2000), this is the most important phase of the data mining project. Here we need to understand the objectives from the business perspective, define the data mining problem to solve and then develop a plan in order to achieve the objectives. Business understanding involves key steps such as determining business objectives, assessing the situation, determining the data mining goals and producing the project plan.

1.3.2 Phase two: data understanding

This phase (Shearer, 2000) starts with an initial data collection, followed by getting familiar with the data, identify data quality problems, discover initial insights into the data and/or detect interesting subsets.

It is crucial that the analyst understands the data that he has at its disposal: it guarantees that the following processes have a solid foundation since all the following processes are dependent on what we are able to extract from data. Making sure that the data is fully understood also makes sure that the best obtainable solution is achievable: if some part of the data is misinterpreted or ignored it might lead to models with sub-par performance.

1.3.3 Phase three: data preparation

It covers the construction of the final data set, that will later be fed into the models. The five steps in data preparation (Shearer, 2000) are the selection, cleaning, construction, integration, and formatting of data.

Since models learn from the data that they are fed with it is very important to make sure that the data that goes into training the models is within certain standards, such as being clean from defective samples and misread values.

Cleaning data is then essential and that means making sure that the various data sources are gathered together without creating more problems in the data, that missing values are correctly handled and the data is in the correct format.

The great majority of problems also require that the initial data is transformed into features that are relevant to the problem that we are trying to predict. In our scenario, we need to transform the data from the matches that the teams play into variables that quantify the strength of the team in individual areas of play in order to be able to compare them.

1.3.4 Phase four: modeling

This phase (Shearer, 2000) is where modeling techniques are selected and applied, calibrating their parameters to optimal values.

This is the step where we build the models with the available data and then assess the model performance. This step might require that we go back to the previous phase in order to reshape the data to improve our models since they are heavily reliant on the data that we feed them.

1.3.5 Phase five: evaluation

Before deploying the models it is important to validate and review the way the model was built in order to assure that a faulty model that does not meet the business criteria is not deployed.

In this phase (Shearer, 2000), the key steps are the evaluation of results, the review process and the determination of the next steps.

1.3.6 Phase six: deploy

The creation of the model does not end the data mining process. In this step is where we enter the production phase. It is needed to develop a way to present the knowledge that the models found, which is dependent on the final target and business goal.

1.4 Thesis structure

The remainder of this thesis is organized as follows:

- Chapter 2 introduces mathematical concepts related to betting.
- Chapter 3 is a brief overview of key data mining concepts, with a focus on the data preparation phase.
- Chapter 4 reviews some machine learning algorithms, focusing on the classification algorithms that have the ability to generate probabilities.
- Chapter 5 goes in-depth into ensemble learning, explaining the problem that is solved with ensembles and introducing algorithms for decision trees and neural networks.
- Chapter 6 briefly reviews previous work in the area of sports and football match predictions.
- Chapter 7 describes the methodology, presents the results and conclusions of the experiences that were conducted in order to predict football matches.
- Chapter 8 summarizes the work done and reflects on the obtained results. At last, future research ideas are pointed out.

Chapter 2

Betting mathematics

The knowledge of some key components of the math behind the betting process is crucial in order to understand the problem.

In this chapter, we describe key formulas and concepts of betting mathematics. First, we define what is an odd and the law of large numbers. Then we go more in-depth with the betting options that exist. Following that, there is an explanation of how the odds are set. At last, the concept of value betting, which allows us to profit from betting, is described.

2.1 Interchanging probabilities and odds

$$decimal\ odds = \frac{1}{probability\ of\ the\ outcome} \tag{2.1}$$

The importance of the equation 2.1 for betting is similar to Ohm's Law for circuit analysis or Newton's Laws of Motion for physics.

It allows us to interchange the fair price and probability of the outcome. We can, given an expected probability, calculate the expected price and based on it verify whether it is worth to bet or not.

2.2 Law of large numbers

The understanding of this concept is crucial in order to comprehend how to make a profit when betting. This law implies that the average value of a large number of events converges to the expected value.

This is crucial since the majority of bets end in binary situations: either you win or lose, which means that you do not have smooth, continuous changes in the balance. Changes are abrupt and discrete, which means that because of their random nature we need to test them in the long-term and not in short-term results.

2.3 Bookmakers and betting exchanges

There are two ways of betting:

- In bookmakers, that is, companies that allow us to bet against their odds. The bookmaker defines the odds and regulates them as they wish.
- In betting exchanges, we can buy and sell bets as if they were stocks, by defining the price and amounts that we want to sell/buy from the market. The exchange works as an intermediate that matches the bets from the costumers with each other, which means that the clients bet against each other.

At first sight, they might leave the impression that they are essentially the same, but when it comes to taxing their clients they have very different approaches.

On the bookmakers they introduce their taxes in the odds offered to the clients. That means that if we sum the probabilities of every outcome of a certain event it will result in a probability over 100%. Their business relies on you constantly placing bets and by complying with their system you will lose money in the long term. The value that you are expected to lose per unit you bet varies from bookmaker to bookmaker and from market to market, but it ranges from 3% to more than 18%. This type of taxation is equivalent to paying VAT when buying a product.

In contrast, betting exchanges work differently. They work by taxing our profits. That makes it possible to have fair odds, that is, where the sum of probabilities for every outcome is 100%. In exchange, you pay a tax when you withdraw your funds. The analogy that can be made is the way that the state taxes companies profits.

2.4 The odds

The odds that are offered have their cycle of life. They have a starting point and the live their life following a set of rules that allows bookmakers to maximize their profit and they die when the event starts.

2.4.1 How are initial odds calculated

In betting exchanges the initial odds are simply set by the bettors themselves, having their own criteria to define what they think is the correct price. Then, the gathering of their opinions define the odds, essentially wisdom of the crowd.

For bookmakers, as they do not have this system that has this ability to set the odds, they need to have a team of specialists on predictions that set an initial price. After that, they release the calculated probabilities to the market, but they take a larger than usual margin for themselves, allowing them to not lose much when mistakes happen. Then they gradually decrease their margins, increasing the odds until bettors start to place bets on the outcomes. With models becoming more reliable, the starting margins are getting lower.

2.4.2 How are odds balanced

The bookmaker's goal is to make a profit. The main goal in each match is to maximize their profit. For this to happen, the odds move in a way so that the profit of the bookmakers will always be the same, regardless of the outcome. This allows the bookmakers to reduce variability in their results. That means that when there is a lot of money being placed in one of the outcomes, the odds of the back (outcome to happen) will be lowered and the odds of the lay (outcome not to happen) will be increased.

As the price of the back of the outcome is lowered and the price to lay the outcome is higher, the bettors will be more tempted to put money in the lay bet than before since now its value is higher. This will balance the distribution of profits across all outcomes and the bookmaker will have the same profit independently of the outcome. This is the way that bookmakers have to guarantee that they maximize their profits, regardless of the outcome.

2.5 Value betting

The way to make a profit in the betting market is simple. If we consistently bet in odds that represent a lower probability than the real probability of the outcome, we will have an expected value, defined in equations 2.2 and 2.3, over 1. This means that it is expected to earn more than what was invested. By the law of large numbers, in the long-term, the return of a large number of bets will be approximately the same as the expected value.

$$expected\ value = probability(outcome) * odd(outcome) \tag{2.2}$$

$$expected\ value = \frac{probability(outcome)}{bookmakers\ probability(outcome)} \tag{2.3}$$

We call a bet on which the real probability of the outcome is higher than what the bookmakers pays us a value bet. The classic example of a value bet is betting heads on a 50/50% coin toss with odds of 2.10, where we get a 0.10 advantage on the fair odds of 2.00.

Generally, it is possible to find value in two situations:

- If we predict the odds to drop to a value that makes us have a profit by later betting against it.
- If the probability of the event happening is higher than the probability that the bookmaker pays us for.

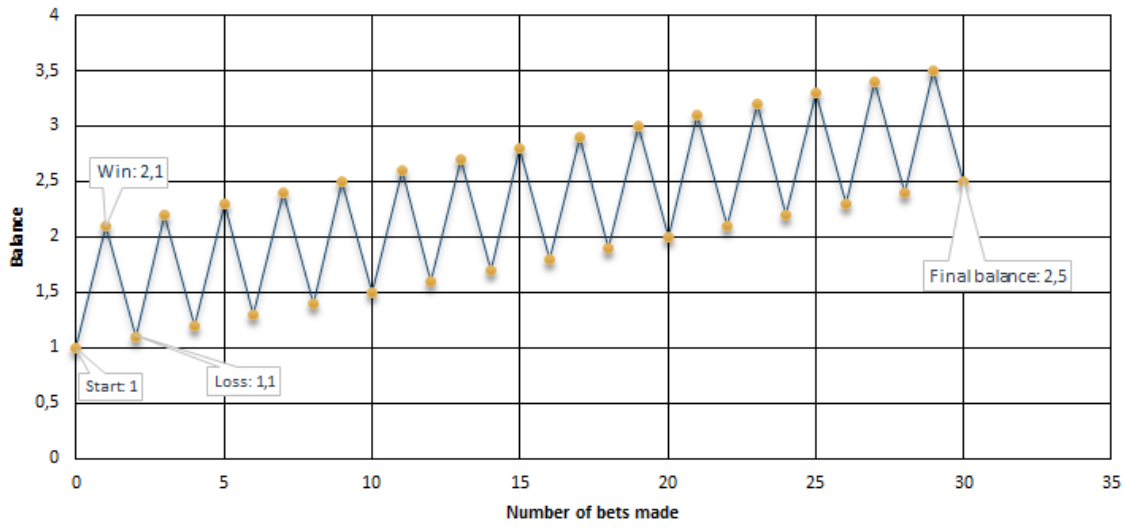


Figure 2.1: An example of how betting with a positive expected value yields profit in the long term. In this example, a bet on a coin toss in the outcome of heads (50%) at the odds of 2.1 (approx. 47,5%). Since the real probability is higher than the probability that the bookmaker is paying, this is a value bet.

Chapter 3

Data mining concepts and techniques

Data mining (Britannica, 2019) is the process of discovering useful patterns and relationships in large volumes of data. It is a field that combines tools from statistics, artificial intelligence and database management to analyze large data sets.

In the following chapter key steps of the data mining process are addressed. First, we start by talking about the importance of data pre-processing and the problems that we are trying to solve in this step. Then we talk approach two topics about features: the feature extraction and feature selection.

3.1 Data pre-processing

Data pre-processing techniques (Kuhn and Johnson, 2013) generally refers to the addition, deletion, or transformation of training set data. This is a step that can make or break a model's predictive ability.

Pre-processing the data is a crucial step. Data can be manipulated in order to improve our models by adding features with good predictive power or removing features that are negatively impacting the models.

The data pre-processing is dependent on the type of algorithm being used. For example, while tree-based algorithms have mechanisms that make them less sensitive to the data shape (for example, skewed data), others like the k-nearest neighbor have high sensitivity. Some algorithms also have an integrated feature generation, for example, multi-layer neural networks that have the ability to combine input features in a perceptron.

3.1.1 Curse of dimensionality

Due to the limited number of samples in the data sets, the number of features that can be used without degrading performance is limited. This happens because adding more features will cause an increase in the dimensional feature space, making it harder for the algorithms to separate the data. Hughes (1968) concluded that the accuracy of a classifier depends on the number of training

instances. Due to this study, the curse of dimensionality is also referred to as Hughes Phenomenon. The dimensionality increase can be visualized in figure 3.1.

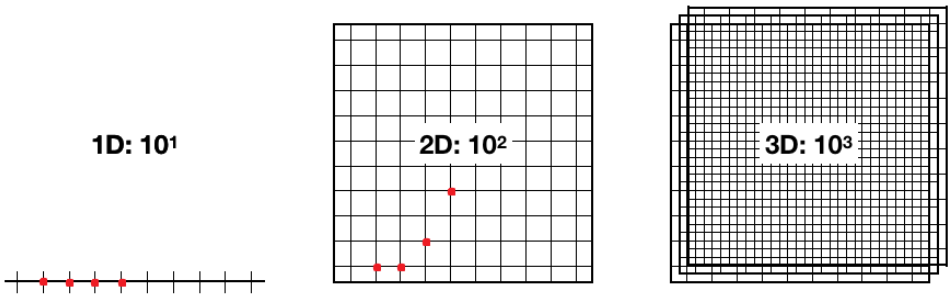


Figure 3.1: Visualization of the increased dimensionality. Note that the distance between two points is always equal or larger when we increase the dimensions.
Source Shetty (2019)

The Hughes Phenomenon, visualized in figure 3.2, shows that, with the same sized data set, as the number of features increases, the classifier performance increases as well until an optimal number of features. After that, the performance will decrease due to the factors before stated. The optimal number of features is defined by the size of the data set and the features themselves.

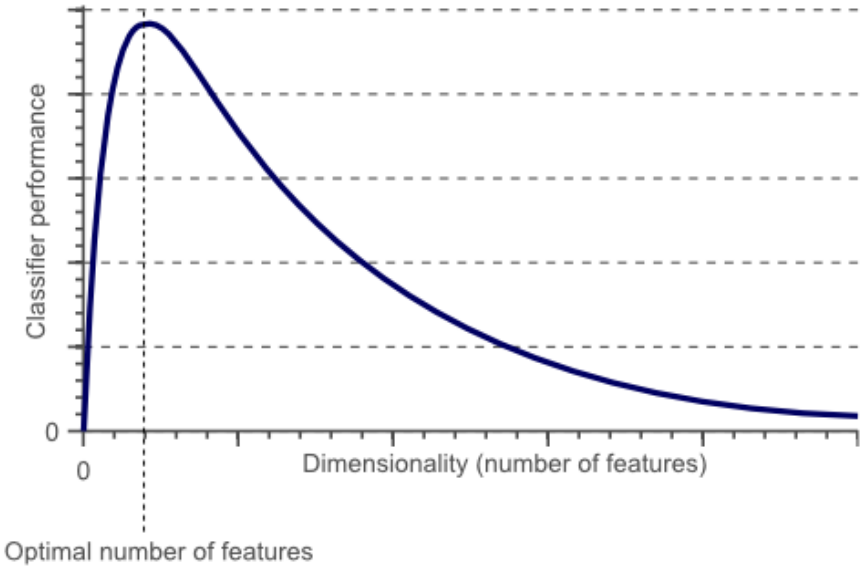


Figure 3.2: Visualization of the Hughes Phenomenon.
Source Shetty (2019)

In order to obtain the best performance possible from the models, we need to have the right features in the right amount. The following chapters will summarize some techniques that can be used for feature generation and feature selection.

3.2 Feature generation

Feature generation techniques (Kuhn and Johnson, 2013) aim at generating a smaller set of features that seek to capture a majority of the information in the original variables. The goal is to in fewer variables represent the original data with reasonable fidelity.

One of the most commonly used data reduction technique is the Principal Component Analysis (PCA). The PCA (Kuhn and Johnson, 2013) seeks to find linear combinations of the features, known as Principal Components, which capture the most possible variance. The goal is to have a set of Principal Components that capture the most variability possible while also being uncorrelated.

Many other techniques for feature extraction rely on feature templates, from which several features can be generated using only measures of central tendency such as mean, median and mode. Variation measures can also be used, for example, standard deviation.

3.3 Feature selection

The main reason behind feature selection is that fewer features means decreased computational time and model complexity.

If any features are highly correlated, this implies that they possess the same information. Removing one should not compromise the performance of the model. Also, some models are sensitive to the data distribution, therefore removing/refactoring these features can improve the model performance and/or stability.

There are three types of approaches to feature selection (Kuhn and Johnson, 2013): filter methods, wrappers and built-in methods.

3.3.1 Filter methods

The main goal when selecting the features to be used in a model is to verify if they hold any information that can help the prediction. For example, if a feature in all the instances has a fixed value (0 variability) the feature holds no information. For tree-based algorithms, the attribute is useless since it will never be selected to be used in a split, however, distance-based algorithms can use the value in the distance calculation, introducing a bias in the prediction. Therefore, if a feature has no variability it should be removed, and features with low variability should be carefully selected.

Alongside with verifying if the feature has any variability, checking for the correlation with the target variable, example in figure 3.3, can also be used as a method to verify if the feature has relevant information. This method should be used with caution since having no correlation with the target variable does not mean that the feature holds no information. It means that on average the feature holds no information, but whenever combined with other features it might reveal interesting patterns.

Lastly, collinearity (Kuhn and Johnson, 2013), the technical term for when a pair of features have a substantial correlation with each other, also has a negative impact on the models. As stated before, if any features are highly correlated, this implies that they possess the same information. Removing one should not compromise the performance of the model.

These selection techniques focus on leaving only the features with information that will help the model to make predictions. This way, not only is the model performance improved but also the computational time and complexity are reduced. In the latter case, a reduction in complexity might also lead to an increase in model interpretability.

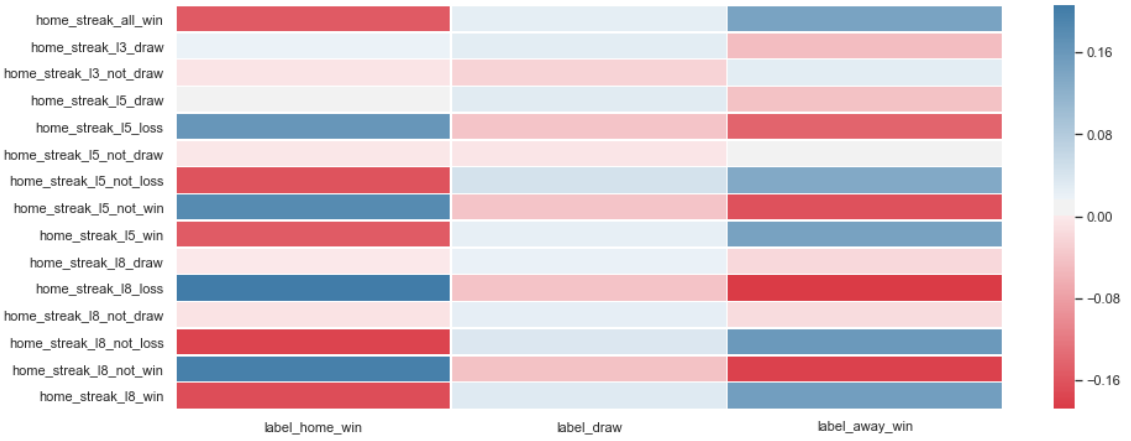


Figure 3.3: A visualization of the different ranges of correlation with the target label.

3.3.2 Wrappers

The wrapper methods (Kuhn and Johnson, 2013) evaluate multiple models with different feature space states and then finds the optimal combination that maximizes the model performance.

Some examples of wrapper methods are forward and backward selection. Search algorithms are also used in order to find the optimal feature set such as simulated annealing and genetic algorithms.

3.3.3 Built-in methods: random forest

Another selection method is based on the ability of some algorithms to automatically select features, in this case, the random forest. There are other algorithms with this ability, such as the Lasso, however, the focus will be on tree-based selection.

In order to select in the nodes the feature on which the node split will occur, tree-based algorithms need to calculate the feature importance. The feature importance calculation (Kuhn and Johnson, 2013), visible in figure 3.4, is based on how much the impurity is reduced across all trees in the forest when using a feature. This feature importance can be used as a selection technique for the models.

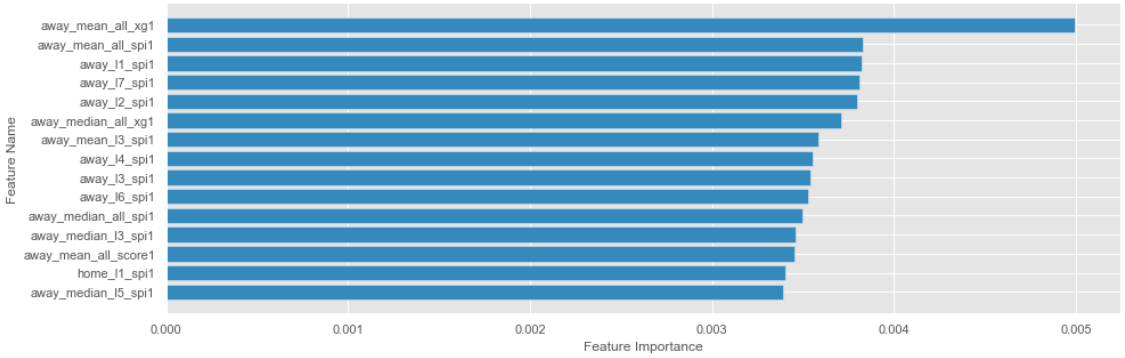


Figure 3.4: A visualization of an example of the feature importance in the random forest.

3.4 Machine Learning

The best simple definition of machine learning came in 1959 by Arthur Samuel. He described machine learning as "a field of study that gives computers the ability to learn without being explicitly programmed". Even though the exact quote exists in neither the Samuel (2000) paper nor the revised version Samuel (1967), this is still the basis for many of the definitions of machine learning.

Machine learning is an auxiliary tool in the data mining context that allows for the creation of models using data. It gives us the possibility of creating models based on experience and observation, being able to create an inference mechanism that will then allow for generalization through induction. Its goal is to create a function y = f(x) that when given an instance x, makes a prediction y. The way that the function f(x) is inferred depends on the algorithm used.

In this chapter, we start by addressing the tasks that can be achieved using machine learning in the context of data mining. We then talk about the machine learning algorithms, focusing on the base learners that have the ability to predict probabilities. For last we do a brief overview of the evaluation metrics that can be used in the context of sports predictions.

3.4.1 Data mining tasks

The two high-level primary goals of data mining are prediction and description. Prediction involves predicting unknown values, and description focus on finding human interpretable patterns in data. These goals can be achieved using several methods (Fayyad et al., 1996):

- Classification
  - Focus on learning a function that classifies a data instance into one of several predefined classes.
- Regression
  - Focus on learning a function that maps a data instance to a real-valued prediction variable.
- Clustering
  - Is a descriptive task where one seeks to identify a finite set of categories or clusters to describe the data.
- Summarization
  - Consists of finding a model that describes significant dependencies between variables.
- Change and deviation detection (also known as outlier/anomaly detection)
  - Focus on discovering the most significant changes in the data from previously measured or normative values.

3.4.2 Classification

Classification models (Kuhn and Johnson, 2013) usually generate two types of predictions. Like regression models, classification models produce a continuous valued prediction, confidence level, which is usually in the form of a probability (i.e., the predicted values of class membership for any individual sample are between 0 and 1 and sum to 1). In addition to a continuous prediction, classification models generate a predicted class, which comes in the form of a discrete category.

If the models are well calibrated the resulting confidence levels from the models can be used as the predicted probabilities.

3.5 Algorithms

In this section we describe the algorithms used. These were the chosen algorithms because they showed good performance in previous academic work, unlike others like naive Bayes.

3.5.1 Decision trees

A decision tree (Kuhn and Johnson, 2013) falls within the category of tree-based models and consists of nested if-then statements, separating the data according to defined criteria. They can be highly interpretable, can handle many types of variables and missing data. However, they are unstable and may not produce optimal predictive performance.

Each node of the tree has a test that is made over a selected attribute and then divides into more nodes in a recursive process until the stopping criterion is met. In order to define the attribute used in order to split the node measures, such as the Gini index and cross entropy, can be used. Depending on the decision tree implementation many hyperparameters can be used. The most important is perhaps the maximum depth of the tree that when lowered allows for an increase in generalization power and thus reducing overfitting. In trade, the model complexity is reduced. A decision tree example can be seen in figure 3.5.

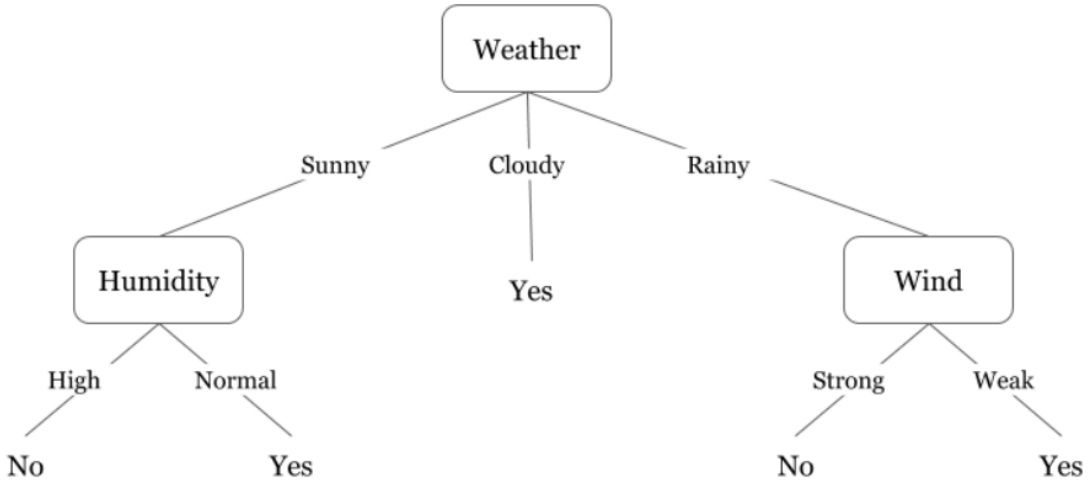


Figure 3.5: A example decision tree classifying if a person should or not play badminton. Source Gupta (2019)

3.5.2 k-nearest neighbor

The k-Nearest neighbor is a distance-based algorithm: the predicted value for a new instance is calculated based on the k closest instances available in the data. That is, the algorithm searches for instances similar to the one that is trying to predict and uses the past results in order to make a new prediction.

The parameters that we need to set is how many neighbors we use (k) and how is the distance between two instances calculated. The distance can be calculated using Euclidean or Minkowski distance metrics. One factor (Kuhn and Johnson, 2013) to take into account when using the k-NN is that the distance value between samples will be biased towards features with larger scales. This means that centering and normalizing or scaling the predictors will influence the performance of the algorithm.

The optimal k parameter is influenced by how many instances of data we have to train our model. A low ratio k/number of instances will lead to high difficulty in generalization since the influence of each of the instances is bigger and it leads to highly localized fitting. A high ratio will lead to instances that are not relevant to the prediction to be used and badly influence the results. The effect of the k value can be seen in figure 3.6.

The addition of more data increases the efficiency of the algorithm due to the reduction of the k/number of instances ratio, however, this ratio needs to be tracked in problems where the amount of instances varies over time in order to readjust the model according to its needs.

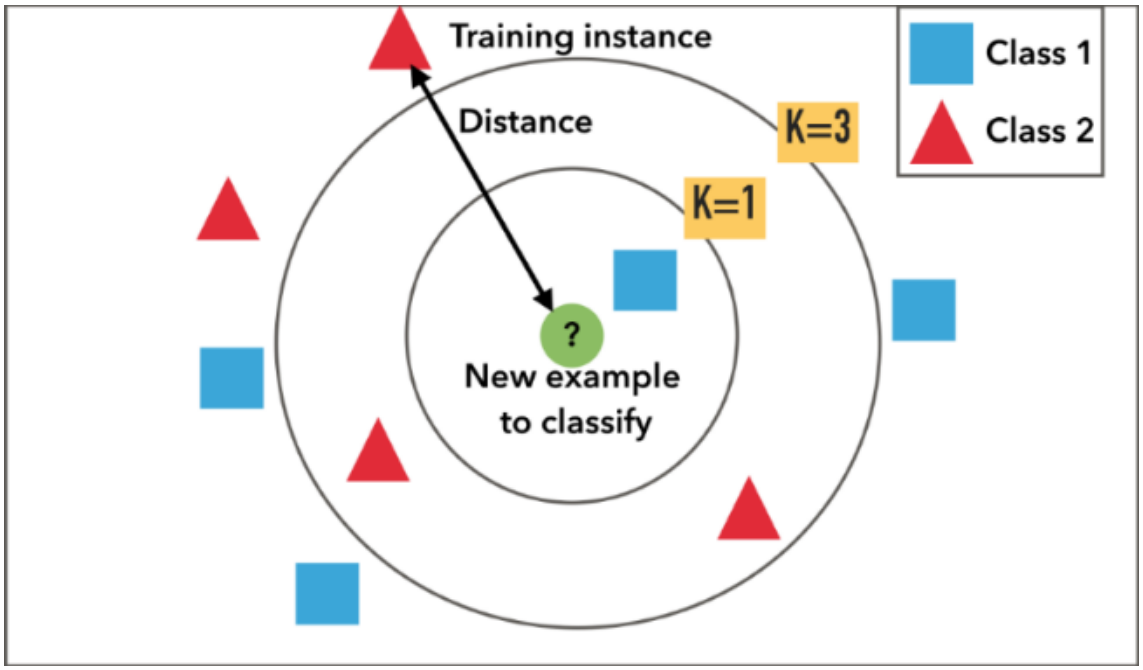


Figure 3.6: A example of a k-nearest neighbor classification. It is visible the dependence of the prediction on the k value, since the prediction from k=1 is different from k=3.
Source Bronshtein (2017)

3.5.3 Neural networks

Neural networks are aggregations of units called perceptron. A perceptron (Kuhn and Johnson, 2013) is a unit that calculates the linear combination of its inputs and then passes the result through an activation function. This can be seen in equation 3.1, where y represents the predicted value and d the total number of inputs. The architecture of a perceptron can be seen in figure 3.7.

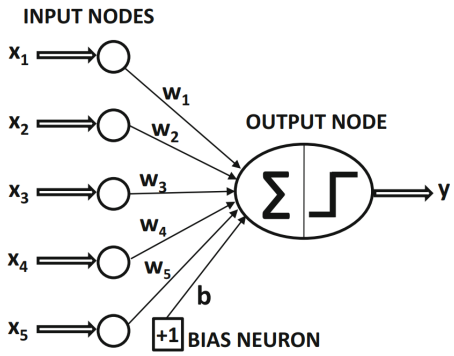


Figure 3.7: Basic architecture of the perceptron.
Source Aggarwal (2018)

$$y = activation\ function(\sum_{j=1}^{d} w_j x_j) \tag{3.1}$$

The perceptron is able to produce classification models with great performance when data is linearly separable. However, many real problems do not have the luxury of being linearly separable. In order to be able to deal with non-linear problems, the perceptron needs to be able to approximate more complex functions. On its own, the perceptron does not have this ability.

To have this ability a network of perceptrons needs to be built. By aggregating perceptrons in a sequential network, the model is now able to approximate nonlinear functions. The shape of the network will limit the complexity of functions that can be approximated. In figure 3.8 we can see an example of a neural network.

The process throughout which the neural networks are usually trained is called back propagation (Aggarwal, 2018). It contains two main phases referred to as forward and backward phases. In the forward phase, the output values are calculated and compared to the actual value and then the gradient of the loss function is calculated according to the results. After that, in the backward phase, the gradients are used to update the weights. This phase is called the backward phase because the gradients are learned from the output node to the input nodes.

While the ability of neural networks to be able to learn deep patterns in data may look like a positive point towards neural networks, sometimes this power can result in bad generalization results. To solve this problem several approaches can be used.

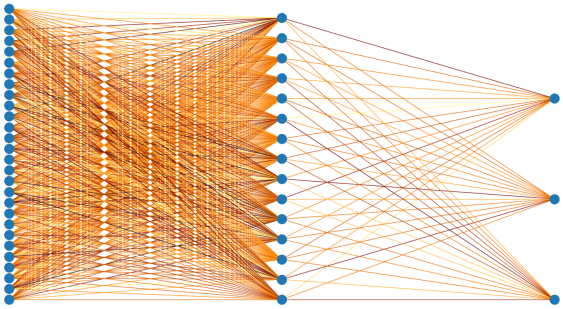


Figure 3.8: A visualization of a neural network with 28 inputs, 1 hidden layer of 15 nodes and 3 outputs.

The first approach is to introduce some form of regularization in the neural network. The reason for the lack of generalization ability comes from a large number of parameters, therefore, limiting the impact of these parameters will result in models with better generalization. One way to induce regularization is to use early stopping. When using early stopping the model will stop its training when the loss function fails to make sizable improvements over a certain period of time. This allows the patterns found in training to not become too deep, which is a source of generalization problems.

Another approach would be to use the dropout method (Aggarwal, 2018). In the dropout method, the network in each iteration of training selects a predefined set of nodes that will be trained during the iteration. This will incorporate regularization into the learning procedure. By dropping input and hidden units from the network, the dropout incorporates noise into the learning process, not allowing the model to learn deeper patterns.

Alongside with regularization techniques, a solution to the generalization problem is ensembling, and will be approached in the next chapter.

3.6 Evaluation metrics

Every problem needs its own way of evaluating the obtained results. In the case of sports betting, the metrics that we are mostly interested in are related to measuring the efficiency of the bets that we make.

3.6.1 Accuracy

The accuracy (equation 3.2) is a standard metric in classification problems.

$$accuracy = \frac{correct\ predictions}{number\ of\ predictions} \tag{3.2}$$

3.6.2 Profit (Rentability)

This evaluation metric is straight forward: we allow the model to make a one unit bet, and then, according to the outcome, add the profit/loss of the bet to our metric. Rentability is defined in equation 3.3.

$$rentability = \sum_{correct\ predictions} (odd - 1) \quad - \quad number\ of\ incorrect\ predictions \tag{3.3}$$

It gives us a broad perspective of how the models are performing. While the following metrics will give a result that is harder to interpret due to being influenced by more factors, this metric will give us the results that are more instinctively interpreted.

3.6.3 Return on investment (ROI)

The ROI metric, defined in equation 3.4, is the ratio of profit to the total amount of money invested on the bets.

$$ROI = \frac{profit}{investment} \tag{3.4}$$

This metric differs from the profit because it incorporates the amount of money invested. If we do unit bets it will allow us to differentiate between models with the same profit but make more or fewer bets. It is also easier to interpret from the business point of view: we know that if we bet 100 units in the model predictions we are expecting to win 100 * ROI units from those bets.

3.6.4 Difference between real profit and expected profit

Whenever we make a value bet we can calculate the amount that we are expecting to win in the long term by doing that bet. For example, if we bet in a 2.05 odd that should be a 2.00 odd we can expect to win 2.5 cents from that bet.

By comparing the profit that we are expected to obtain with the profit that actually has been made we can check if we are making a profit because the probabilities are correctly predicted or because we are simply getting lucky.

3.6.5 Ranked probability score

The ranked probability score (Weigel et al., 2006) is one where we can compare the calculated probabilities with the actual probabilities of the event happening.

The ranked probability score is defined in equation 3.5, where M is the number of forecasts, γ the predicted probability and the Y the observed probability.

$$RPS = \sum_{i=1}^{M} (\gamma i - Yi)^2 \tag{3.5}$$

Chapter 4

Ensemble learning

There are several ways with which we can improve the performance of machine learning models. We are taking a look at the most common approach, ensemble learning.

In this chapter, we first take a look at and formulate the problem that we are trying to solve with ensemble learning. Then a quick example based on data from the our data set illustrates and confirms the formulation of the problem. After that, ensemble learning is defined and ensemble algorithms are presented for general use, for decision trees and for neural networks.

4.1 The generalization problem

With so much ability from the algorithms to learn the complex functions in many domains there comes the problem of overfitting the data whenever we are not careful enough with the design of the learning process. Overfitting (Aggarwal, 2018) means that the algorithms provide excellent predictions on the training data, but performs badly when tested in previously unseen instances. In an extreme form of overfitting, it can be described as memorization of patterns in the data. Simply put, overfitted models do not generalize well to instances with which the models were not trained.

This is especially evident in neural networks because of them being able to adapt to the training data.

The ability of a learner to provide useful predictions for instances it has never seen before is referred to as generalization (Aggarwal, 2018).

4.2 Bias and Variance

The generalization problem can be quantified by measures of the bias and variance errors. In figure 4.1 we can visualize the effects on predictions.

4.2.1 Bias

Bias is related to the error caused by simplifying assumptions in the model that causes constant errors across different choices of training data. That is, if a model has high bias error it means that

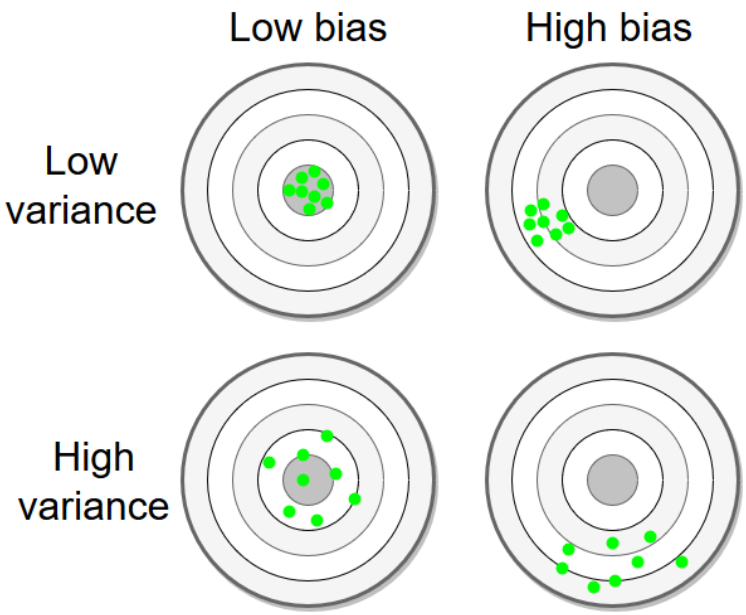


Figure 4.1: Representation of the bias-variance problem as an analogy to the precision-accuracy. Source Frank (2017)

it only recognized simple patterns in data and is a low complexity model, which causes the model to only make vague predictions. As stated by Aggarwal (2018), this error cannot be removed even with an infinite source of data.

The bias error calculation for an instance is represented in equation 4.1, where M is the number of predictions made, γ is the predicted value and y is the real value.

$$bias^2 = \frac{1}{M}\sum_{i}^{M} y_i - \gamma_i \tag{4.1}$$

4.2.2 Variance

Variance is the variability of model prediction for a given instance. A high variance means that a model learns deep patterns in the training data, creating high complexity models, that do not generalize well into the testing data. Slight changes in the training data induce big changes in the predictions. This error can be reduced by using bigger amounts of data.

An estimate of variance can be made by calculating the standard deviation of the predictions for each instance, as can be seen in equation 4.2.

$$estimated\ variance = \frac{1}{M}\sum_{i}^{M} stdev(\gamma_i) \tag{4.2}$$

4.2.3 Bias-Variance trade-off

Simple models result in a high bias and low variance, complex models in low bias and high variance. The bias-variance trade-off allows us to find an intermediate point that allows us to minimize the sum of the bias and variance error.

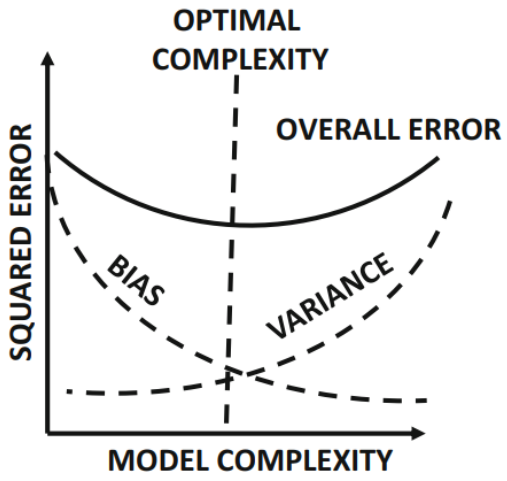


Figure 4.2: Representation of how bias, variance and overall error moves according to the model complexity.
Source Aggarwal (2018)

As seen in figure 4.2, there is an optimal model complexity that allows us to reduce the overall error, finding a compromise between the bias and variance error.

4.3 Scenario: Decision trees parameter optimization

One of the ways to observe how bias and variance moves is to observe the predictions of a simple model (decision tree), and compare how the bias and variance errors behave when we shift parameters.

This experiment consists of testing a simple decision tree regressor with default parameters, except the maximum depth. As stated in section 3.5.1, the maximum depth is one of the parameters that we can use to tune the model complexity. The higher the maximum depth, the higher the model complexity.

The test was executed by training 50 different models each with a different data set generated by randomly selecting 50% of the instances in data. The used features are the goals scored and conceded in the last 7 games by the teams in the game we are trying to predict.

In figure 4.3, we can confirm what was stated before: more complexity (models with higher maximum depth) leads to higher variance and lower bias.

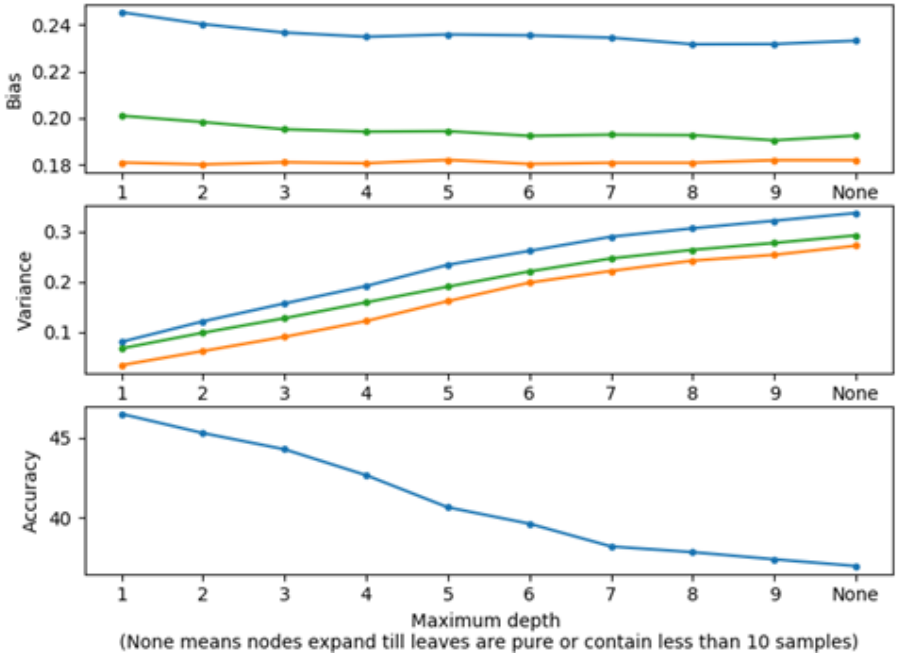


Figure 4.3: Movement of the Bias-Variance trade-off according to the model complexity.

In a problem of this complexity, slight improvements on the bias error lead to heavy penalization in terms of variance error, with the variance error being dominant over the bias error. This can be verified in the accuracy of the models. Even though the bias error is being reduced the accuracy metric is unable to improve.

The result of this dominance of the variance error leads to the less complex models being able to predict with better accuracy. Isolating the maximum depth parameter allows us to observe that the optimal complexity of the decision tree model is with very low maximum depth.

4.4 Regularization

The reason for overfitting is that the model captures the noise in the train data. One of the ways to reduce overfitting is regularization. Regularization (Aggarwal, 2018) is a form of penalizing the models for chasing complex models. This penalization is usually added in the loss function. It can be used in several machine learning algorithms.

Regularization parameters enable the model to be tuned to the optimal complexity of the models by manipulating its complexity.

4.5 Ensemble

Another way of reducing the error of a classifier (Aggarwal, 2018) is to find a way to reduce either the bias or the variance without affecting the other component.

The most used technique in order to reduce bias and variance is ensemble learning. Ensemble methods aggregate multiple models to obtain better performance than what could be obtained from any of the constituent models alone.

Ensemble learning has, depending on the algorithm used, several advantages: from reducing the bias and variance to the ability to be able to improve weak learners. However, it comes with the cost of several impacts on computational performance, due to the need for multiple models to be trained.

4.5.1 Bagging and Subsampling

The most used ensemble technique is bagging (Aggarwal, 2018), also known as bootstrap aggregating, mainly due to its simplicity. The basic idea is to generate new training data sets from the single instance of the base data by sampling, which can be performed with or without replacement. An example can be seen in figure 4.4.

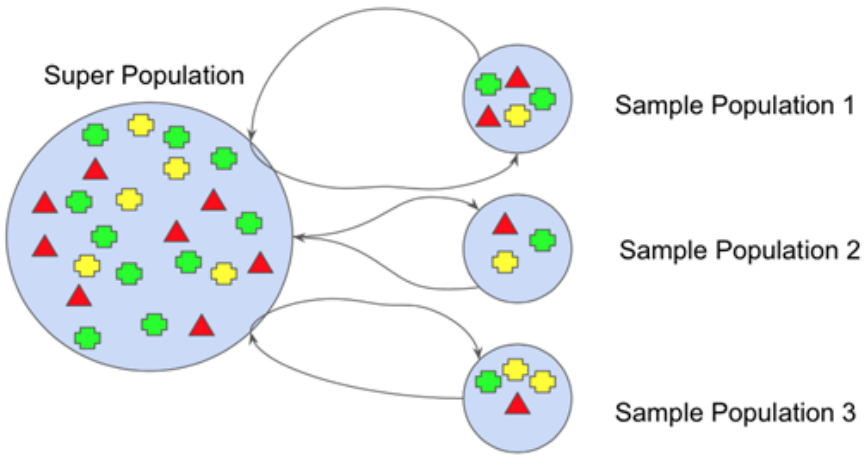


Figure 4.4: Production of subsamples in bagging.
Source: Shubham (2018)

Simply put, subsets of data are generated from the data set that contains a percentage of the instances. This creates variability which is the most important factor in the success of ensembles.

Then, a predefined amount of models is trained on those data sets and predictions are made. The final prediction of the ensemble is a function of all the probabilities, usually average, possibly weighted, median values or voting.

Bagging and subsampling techniques are primarily directed towards reducing variance, however, bias can also be slightly reduced.

4.5.2 Boosting

A boosting algorithm is an algorithm in which the primary goal is to reduce bias.

The most well known boosting algorithm is the AdaBoost (Freund and Schapire, 1997). The name comes from "adaptative boosting" since it boosts a weak learner based on adjusting to the errors in its predictions.

The AdaBoost consists in iteratively improving a weak learner by changing the weights of data instances according to its predictions so that instances that are wrongly predicted have a bigger weight than correctly predicted instances, forcing the models to learn them.

The final prediction of a boosting model is a weighted average of the models from each iteration. The weight of each model usually decreases over the iterations.

This technique might also allow for a decrease in variance, when comparing with a single estimate. It also is able to substantially reduce bias error. The price to pay for that is that this kind of ensembles is that not only it does not help reducing overfitting but it also faces the problem of increasing overfitting, and that is a factor to take into account when choosing the ensemble to use.

4.6 Ensembling decision trees

4.6.1 Bagging and AdaBoost

Since decision trees are usually unstable learners and have a high variability, they are able to make sizable improvements when ensembled. This is the case for both bagging and AdaBoost techniques. They are able to improve the results of the models in both bias and variance.

Even though both approaches work well, there are two algorithms that work with decision trees that are able to generate similar or better results.

4.6.2 Random forest

As defined by Ho (1995), "*the essence of the method* (random forest) *is to build multiple trees in randomly selected subspaces of the feature space*". According to the author and proven by several experiences, when the trees are generated from different feature subspaces they tend to complement each other's predictions, in a way that improves the classification accuracy, even when compared with bagging. The process of randomly selecting features for the model is called random subspace sampling.

The main difference from bagging is that not only are the instances subsampled, but also the features. This leads to increased variability from the trees that improves the ensemble model performance.

4.6.3 Gradient boosting

While in the AdaBoost algorithm the adaptation comes from giving more weight to the instances that are wrongly predicted, in the gradient boosting (Breiman, 1997) the instances to which more weight is given is identified by gradients. The gradient boosting technique (Kuhn and Johnson, 2013) works on a defined loss function, and seeks to find an additive models that minimizes this loss function.

The gradient boosting technique does not work exclusive on decision trees like the random forest.

4.6.3.1 XGBoost

An improved version of the gradient boosting technique was later developed, the XGBoost. The most important advantages that the XGBoost holds over the gradient boosting (Chen and Guestrin, 2016) are the implementation of regularization and parallel processing.

4.7 Ensembling neural networks

In neural networks, ensemble methods are usually focused on variance reduction. This is because (Aggarwal, 2018) neural networks are valued for their ability to build arbitrarily complex models, in which the bias is relatively low. This means that neural networks are able to generate high complexity models which leads to higher variance, manifested as overfitting. Therefore, when using neural networks we seek variance reduction in order to have better generalization prowess.

4.7.1 Bagging and boosting

While the bagging approach yields the expected results, high variance reduction and slight bias reduction, the boosting techniques fail to do so. Neural networks are already very strong learners and boosting techniques work better with weak learners in order to improve the models.

4.7.2 Negative correlation learning

Since ensemble models benefit from diversity between the base learners, it is useful to increase this diversity. In negative correlation learning (Liu and Yao, 1999), the approach is to train individual networks in an ensemble and combining them in the same process. All the neural networks in the ensemble are trained simultaneously and interactively through a correlation penalty term in their error function.

The difference from regular neural network training is in the loss function. Subtracted to the regular loss function, that is a function of the predicted value and the real value, is a percentage of the loss function calculated between the value predicted by the model and the ensemble predicted value. This can be seen in equation 4.3, where γ is a neural network prediction, ε the ensemble prediction and y is the real value. This incentives models to go in a different direction from the average value of the ensemble, creating diversity in the model's opinions and improving the model classification performance.

$$new\ loss\ function\ =\ loss\ function(\gamma, y)\ -\ \lambda\ *\ loss\ function(\gamma, \varepsilon) \tag{4.3}$$

Chapter 5

Data mining in the sports context

In this chapter, we take a look at the previous work done in the area of sports predictions. The search method was using Science Direct search engine to find the most relevant papers in the area, with a focus on the most recent work.

Academia has embraced the problem of predicting football matches very heavily. Some of the most notable work was done by Constantinou et al. (2012) with the usage of Bayesian networks in order to create an agent to predict probabilities in the English Premier League 2010/11 season, managing to obtain profit even though the test sample was very small. This model was then improved (Constantinou et al., 2013), reaching the conclusion that a less complex model can perform better, which suggests that selecting the right data is more important than just having a lot of data.

Bayesian networks seem to be the go-to prediction method in academia. Joseph et al. (2006) evaluated the predictions made by an expert-made Bayesian network for the London team Tottenham Hotspurs in the seasons of 1995/96 and 1996/97 and compared them with some of the state of the art machine learning prediction algorithms, like MC4 decision trees, naive Bayes, data-driven Bayesian and a K-nearest neighbour learner. The conclusion was that expert-based Bayesian networks were way more reliable at predicting the outcome of games, although this was probably due to the type of data available not fitting the algorithms used, with some of the attributes being the boolean variables to check if the most important Tottenham players were playing in the game. There was also a problem in the data dimensionality that leads to the results being biased towards the Bayesian network.

Owramipur et al. (2013) confirmed once again that Bayesian networks perform very well under the right conditions, as they were able to predict Barcelona games in the Spanish La Liga with 92% accuracy, which is not the best metric to evaluate the models for a team that is favourite to win every game, but shows its efficiency.

Rotshtein et al. (2005) used fuzzy based models with genetic and neural tuning to predict the results of games for the football championship of Finland, which achieved interesting results as it was done in data set with a low number of attributes. It predicted the spread (the difference between two teams score in a game) based only on previous results.

There is some relevant work done in predictions using neural networks. Mccabe and Trevathan (2008) compares the behaviour of artificial neural networks predicting football and rugby matches, reaching the conclusion that the predictions are better in sports with a lesser percentage of draws. The better accuracy in sports with a lesser percentage of draws comes from the fact that the draw is never the most expected result. This means that, in general, classification models do not predict draws since the probability of a draw is always lower than any of the probabilities of the teams to win the game.

Neural networks are also useful in predicting the sports betting exchanges movements. Dzalbs and Kalganova (2018) used artificial neural networks and cartesian genetic programming in order to test automated betting strategies in a betting exchange using the Betfair API.

Another relevant work was done by Hvattum and Arntzen (2010) by adapting the ELO rating system in order to make football predictions. The ELO rating system was developed by Elo (1978) to quantify the strength of chess players. There are similar ELO inspired rating systems but no relevant studies were found on them. Rating systems might be useful because they might allow us to generate important features for models.

A full data mining methodology was done by Baboota and Kaur (2019). The unanimously considered a crucial aspect in the data mining process is the feature engineering which is entirely dependent on the data that is available. The machine learning algorithms used were naive Bayes, support vector machines, random forest and gradient boosting. The ensemble algorithms, random forest and gradient boosting, are the algorithms that better perform, suggesting that ensemble algorithms are the go-to techniques in order to have the best predictions.

It is suggested the usage of a performance metric proposed by Epstein (1969), the ranked probability score, that measure how well forecasts are expressed as probabilities according to the observed outcome. Another metric that is relevant is the "losses per unit bet" (Buhagiar et al., 2018), where the performance of an algorithm is set by the results of the bets that the algorithm makes.

A summary of the literature review from the point of view of prediction methods can be seen in table 5.1.

On the topic of sports betting, one of the most studied phenoms studied in academia was the favourite-longshot bias – the odds that the bookmakers show are better at predicting the probabilities for the favourite to win than predicting the lower probabilities. Buhagiar et al. (2018) studies and confirms the existence of the favorite-longshot bias. The reason given is that the longshots are riskier for bookmakers if incorrectly priced, leading to a bigger tax on these odds. While the favourite-longshot bias might not be enough in order to make a profit it might lead to better results when exploited.

Table 5.1: A brief summary of the literature review.

Reference	Method	Data set	Seasons	Best algorithm	Accuracy
Constantinou et al. (2012)	BN	English PL	2010/2011	-	-
Constantinou et al. (2013)	BN	English PL	1993 to 2010	-	-
Joseph et al. (2006)	BN vs Machine learning	Tottenham games	1995 to 1997	K-NN	50,58%
Owramipur et al. (2013)	BN	Barcelona games	2008/2009	-	92%
Rotshtein et al. (2005)	Fuzzy model	Finnish PL	1994 to 2001	-	-
Mccabe and Trevathan (2008)	NN	English PL	2002 to 2006	-	54,60%
Dzalbs and Kalganova (2018)	NN	BetFair odds	1 Jan to 17 May 2016	-	-
Hvattum and Arntzen (2010)	ELO-based	4 English divisions	1993 to 2008	-	-
Baboota and Kaur (2019)	Data mining	English PL	2014 to 2016	Gradient Boosting	58,5% (not validated)

Chapter 6

The approach to the problem and results

As stated before, the used methodology was CRISP-DM. This chapter summarizes the methodology, results and conclusions of the developed experiences. The structure of this chapter follows the CRISP-DM methodology.

6.1 Business understanding

6.1.1 Determine the business objectives

Following the CRISP-DM methodology, the first step is the business understanding phase. In this phase, after research on the topic was made, the following business objectives were set:

- Calculate probabilities of football games.
- Verify calculated probabilities on different betting strategies.
- Extract value from the predictions.

Alongside these objectives, the business success criteria were also defined:

- Obtain a positive ROI.

6.1.2 Determine the data mining goals

From the business objectives derived the data mining goals and success criteria.

- Data mining goal
  - Obtain models that calculate probabilities of football games winners (1x2), maximizing its accuracy in order to calibrate its probabilities.
- Data mining success criteria
  - Models have a positive ROI from the point of view of the bettor.

6.1.3 Assess the situation

On the development of the project the following tools were used:

- Python programing language
  - pandas
  - sklearn
  - keras (tensorflow interface)
  - matplotlib and networkx
  - beautifulsoup

It was also assessed which data was freely available for use. The free data sets that were available and relevant in this context were:

- football-data.co.uk (Football-Data.co.uk) – general statistics and odds from matches
- fivethirtyeight.com (FiveThirtyEight) – complex football statistics such as expected goals

6.1.4 Evaluation metrics

In order to be able to evaluate the models, key metrics were defined. Some of these metrics are general and found in several other machine learning problems, others are business specific metrics to evaluate the betting performance of the models.

- General
  - Accuracy (defined in 3.6.1) – Percentage of correct predictions.
  - Bias (defined in 4.2.1) – Difference between the mean of the model predictions and the correct value.
  - Estimated variance (defined in 4.2.2) – Variability of the model prediction for a given instance.
- Specific
  - Profit (defined in 3.6.2) – Rentability of the models.
  - Return on Investment (ROI) (defined in 3.6.3) – Profit/Investment.

All of these metrics depend on how we interpret the results from the models. In this case, two different strategies are used:

- Strategy 1
  - Bet on the algorithm's favorite.
  - This strategy is more focused on the classification part of the problem.
- Strategy 2
  - Bet when the odd of the bookmaker is bigger than the odd calculated by the algorithm.
  - This strategy is more focused on the regression part of the problem.

The specific metrics also depend on how we manage our budget. Two different approaches are evaluated:

- Unit betting
  - Each bet is 1 unit.
- Bank Percentage betting
  - Each bet is 5% of our budget, with the budget starting at 20 units.

6.2 Data understanding

6.2.1 Collect the initial data

In order to obtain more knowledge on the games that we already had available, some features about the games were obtained via web scraping, adding more depth to the database.

- Transfermarkt - Players and teams market value
- Oddsportal - Odds from other markets, such as result and goals at half time
- Online Betting Academy - Deeper game statistics, such as offsides and ball possession

The important factor when adding more data to the database is to make sure that we are potentially adding more information. Redundant information in some scenarios is useful for error detection, however, in this context it will only make the database more complex.

6.2.1.1 Evaluation of the collected data

Web scraping allows us to have access to the nearly infinite source of data on the internet. However, web scraping usually results in several data quality problems, since they are dependent on a third party to not have errors and missing data in their systems. Unfortunately, some of these errors happened in the acquired data.

The main problem with the transfermarkt.com database was the missing values when the matches were rescheduled. This, and other problems, resulted in the database not having data for approximately 6% of the games previously available in the database.

The same problem occurred with the Online Betting Academy website, however, to a larger extent, with nearly 15% of instances not having data available.

With these problems resulting in substantial damage to the database, potentially losing 15% of the instances due to missing values, this data was excluded from an initial approach to the data mining process.

6.2.2 The used data

Having in this stage excluded the data obtained via web scraping, the remaining available data sets were the following:

- football-data.co.uk
- fivethirtyeight.com

6.2.2.1 football-data.co.uk data

In the football-data.co.uk data set the available features were divided into 3 categories:

- Labels, such as team names, dates and league name.

- Game statistics, such as goals, goals at half time, shots, shots on target, corners, fouls, yellow cards and red cards.
- Game odds, such as the Bet365 odds for the result or number of goals at the end of regular time.

A sample of the data set can be visualized in appendix C.

6.2.2.2 fivethirtyeight.com data

The fivethirtyeight.com *soccer-spi* data set also had 3 categories of features.

- Labels, such as team names, dates and league name.
- Game statistics, such as goals, expected goals, non-shot expected goals and adjusted score. More about expected goals in appendix A.
- Game prediction tools, such as the SPI ranking (an ELO based approach), projected score and predicted probabilities from 538's models.

A sample of the data set can be visualized in appendix D.

6.2.3 Explore the data

At this point, it was important to answer some questions that could lead us to take the correct approach when modeling. For example, whether it was important to separate leagues from each other due to football cultural differences or if it was better to keep the data together to improve our predictions. To answer this question, we explored how the distribution of the labels, game-winner, varies according to the leagues. The results can be visualized in figure 6.1.

The most important fact to retrieve from this figure is that between different championships there are different game characteristics that affect the outcome of matches. For example, comparing the difference between the Portuguese Liga and German 2. Bundesliga in the Home Wins department we can verify a difference of almost 10%. There are a lot of factors that can influence the percentage of home wins in a league, for example, stadiums with artificial turf instead of natural grass or even how damaged a natural grass field is can lead to increased difficulties for the teams that are not used to play in this kind of pitches. Alongside some other factors such as fan support, congestion of fixtures and difference in the budget between the smaller and bigger teams have a different impact on each of the leagues.

This means that, in an ideal world, we should train each league with its own data in order to obtain better results. However, in some of the leagues, the data set would be too small in order to do that. For example, the Portuguese Liga only has 500 instances, which can lead to a shortage of data for some of the machine learning algorithms used.

To verify if each league has enough data to be trained and tested with its own data, there is a need to find the point at which the models start to not improve when handed more data. To do

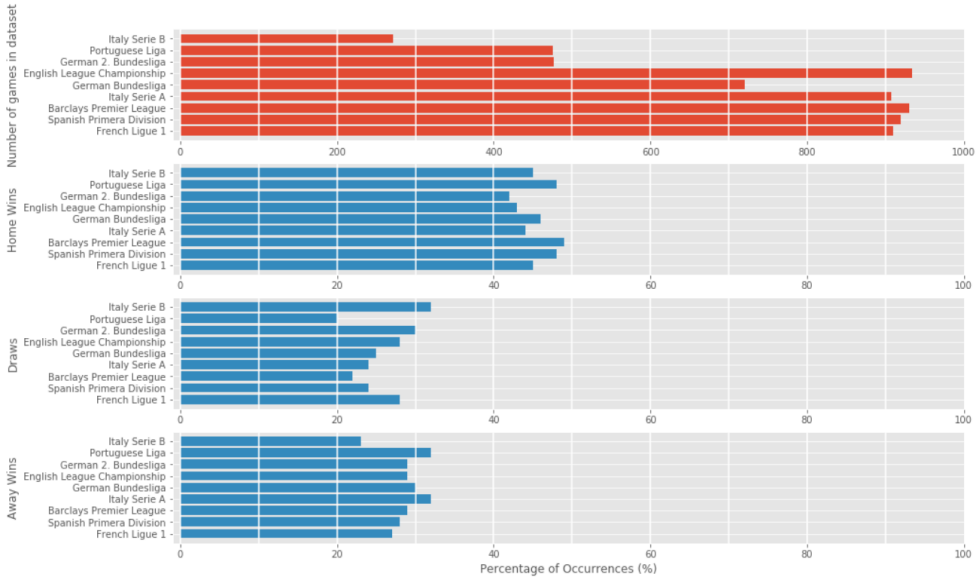


Figure 6.1: Analysis of the distribution of the final results of football matches according to the league.

that, a Python script tested different sizes of data sets on a Random Forest Classifier with 1000 estimators, using as features the raw statistics of the previous 7 games of both teams in a total of 448 features, with the results shown in figure 6.2.

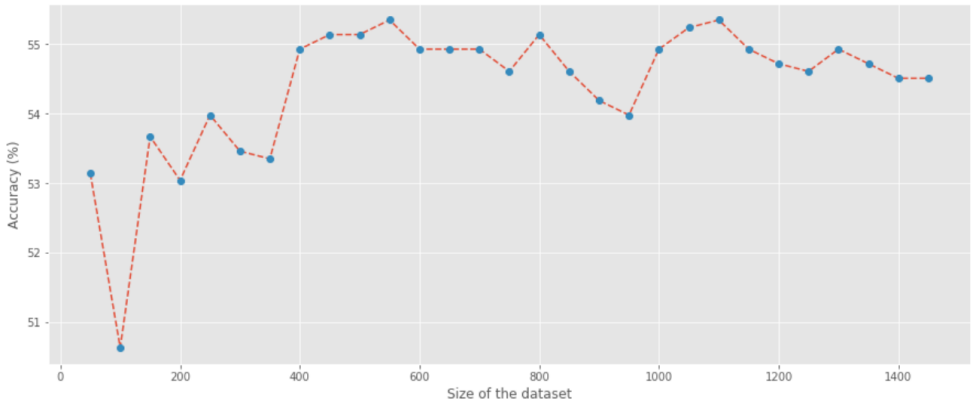


Figure 6.2: Accuracy of the random forest classifier model with 1000 estimators according to the number of instances used for training.

An accurate estimate would be around 500 instances until the model reaches peak accuracy. This is enough for the more extensive leagues such as the English Premier League to train and test in a 60% train and 40% test split. However, this would restrict us of experimenting with several leagues that don't even have enough instances to reach the peak accuracy, leaving us with no data for testing. This number of instances needed can potentially be lowered if the number of features in our models were reduced.

The purposed solutions are: cross-league validation, that is, hold each league out for training and then test the model in the league. This solution leads to the problem that the models are not trained with data from the league that they are being tested in. A second solution would be to do a regular cross or split validation with data from all leagues being used, knowing the risk that exists from the differences between the leagues. For now, the focus is on this second solution due to the ease of implementation and then the ability to easily evaluate each league separated from the others in the future, to further validate the results. The split-validation is the preferred method since it prevents the use of future data to predict a match.

Further exploratory data analysis can be seen in appendix B.

6.3 Data preparation

6.3.1 Train/test/validation split

In order to be able to validate the results there was the need to split the data into different categories: train, test and validation sets. For the validation set, it was chosen to use the 18/19 season data. This allowed validating the models in a whole season.

The division between the remaining data into train/test sets was done in 70% train, 30% test. Table 6.1 summarizes the data splits.

Table 6.1: Subsets generated from the data set.
∗If we include European secondary leagues, the number rises to 2629. However, these leagues are not included in the training data, and so, they will be excluded in a first instance.

Data set	Number of instances	Description
Train	2224	
Test	954	
Validation	1656 (2629)∗	isolated 18/19 season

6.3.2 Generation of features

Based on the experience acquired in the exploration of data it is now possible to start generating features to feed into the models. In order to generate features several techniques were used, all of them based in feature templates.

The problem that we face when generating features is that the goal is not to predict the game based on the data from the game. The goal is to predict the game based on the data from previous games.

The first technique is to include in the feature set the raw features from the data. Some of the modeling techniques are able to generate features by itself, for example in a perceptron, so it is useful to verify if they have any value when modeling and they also establish a good baseline in order to observe if the improvement is being made. For that, the statistics of the last 7 games from both teams are included in the feature set, each statistic as an individual feature.

Another used technique was to use measures of central tendency such as mean and median and variation measures like standard deviation in order to summarize the data from the previous k games in one feature, which largely reduces the number of features needed in relation to the raw features.

There are two more features types that can be generated from game data: the form of the team (the number of victories, draws and losses in the last k games) and the streaks (how many games in a row has a team won/drawn/lost in the last k games).

Added to this approach was the possibility of varying the k that indicated the number of past games used to generate the features. The values of k were carefully chosen since it heavily increases the number of features generated. Those values were k=3, 5, 7 and all. When k = all, it means that all the games from the team in the current season were used to calculate the feature. A summary of the features generated can be seen in table 6.2.

Table 6.2: A collection of the features generated. Each combination from the 4 columns corresponds to a feature. Functions in parenthesis were excluded.
∗Unprocessed features always refers to the last 7 games

Team	**Function**	**Stat**	**Number of games used**
Home	unprocessed*	goals scored	all
Away	mean	goals conceded	last 7
	median	half time goals scored	last 5
	count	half time goals conceded	last 3
	streak	shots made	
	(stdev)	shots conceded	
	(mode)	shots on target made	
		shots on target conceded	
		corners for	
		corners against	
		yellows received	
		yellows for opponent	
		reds received	
		reds for opponent	
		expected goals for	
		expected goals against	
		non-shot expected goals for	
		non-shot expected goals against	
		team SPI	
		opponent team SPI	
		game played at home (1) or away (0)	

The total number of features generated is 980.

6.3.3 Selection of features

Having such an extensive number of features might damage the models. This is due to the correlation between variables and also the curse of dimensionality, where the more features the model is trained on the more data instances are needed. In order to reduce the number of features that are going to be fed into the models, some techniques were attempted.

Before trying these techniques, it was important to establish a baseline model. The baseline model allows us to compare between the models that the selected features are generating and the baseline, and with that verify if we are actually improving our models. To serve as a baseline model it was chosen the random forest classifier due to the lack of necessary complex hyperparameters and because, being an ensemble technique, it allows for a low variance in our results. This random forest classifier will be trained using as features the raw statistics of the previous 7 games of both teams, totaling 448 features. The results can be seen in figure 6.3.

Also important is to define which metrics will be used to compare the models. The chosen metrics are accuracy and bias. Since this is a multiclass problem, the bias will be measured for each class and the exhibited value is the sum of the bias of the 3 classes.

Table 6.3: Baseline model performance.

Model	Accuracy	Bias
Baseline	54.51	0.6801

Another defined baseline is the existing bookmakers' predictions. By checking the bookmakers' favorite, we can use it as if it was a classification model. The results are shown in figure 6.3.

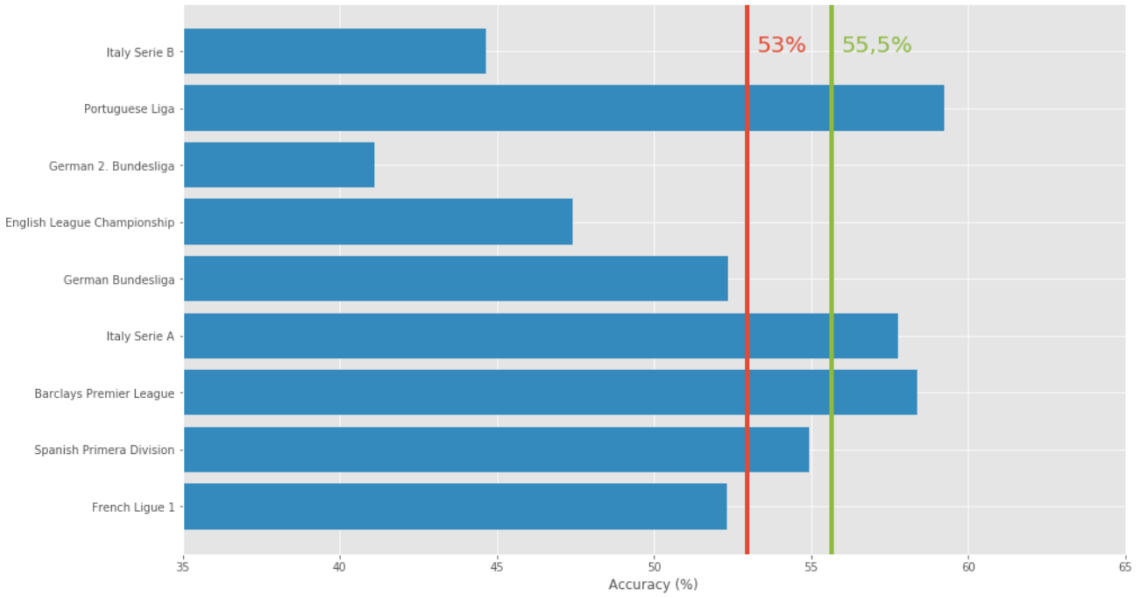


Figure 6.3: Percentage of correct predictions by the bookmakers across the different leagues.

The predictions of the bookmaker's averages 53% for all the leagues present in the data set, and average 55.5% when the European secondary leagues are removed (Italy Serie B, German 2. Bundesliga and English League Championship). Those will be two important baselines to evaluate our models at the end of the cycle.

6.3.3.1 Predefined feature sets

The first tests made were the already predefined sets of generated features. The results can be seen in the table 6.4.

Table 6.4: Results of the different feature sets on a random forest classifier with 1000 estimators.

Data set	No features	Results	
		Accuracy	Bias
Baseline (Raw data)	320	54.51	0.5755
Only Mean	256	54.4	0.5785
Only Median	256	52.94	0.5792
Only Form	32	51.68	0.6021
Only Streak	48	52.73	0.5945

These results do not show improvement over the baseline. However, they needed to be complemented in order to get a bigger idea of the impact of each feature set in different algorithms. One more test was performed, using the K-NN algorithm, due to the different type of inference system (distance-based instead of tree-based). The numbers of neighbors used was 5. The results can be seen in the table 6.5.

Table 6.5: Results of the different feature sets on a k-nearest neighbor classifier with k = 5.

Data set	No features	Results	
		Accuracy	Bias
Baseline (Raw data)	320	46.33	0.6873
Only Mean	256	44.03	0.6932
Only Median	256	45.81	0.6910
Only Form	32	42.56	0.7176
Only Streak	48	40.46	0.7322

In the end, none of these tests lead to any of the features to be excluded. It looks like the results are biased towards the number of features, making conclusions unreliable on this experiment. Even when equal numbers of features were used (mean vs median) the results were different in different algorithms.

6.3.3.2 Correlation-based selection

Another technique used was based on the correlations between the features generated and the data labels.

There are two problems related to correlations: the problem of correlations between the labels and the features, in which features should have a high correlation with labels, and the collinearity between features, since highly correlated features are useless since they do not add any information to the model and might worsen the results.

To filter the features, two constraints were added to the selected data sets: the first being that only features with an absolute correlation higher than 0.2 with the *one-hot encoded* label were kept. The second constraint was that if two features have collinearity bigger than 0.9, the feature with the lower correlation with the label was removed from the selected features. Both these values were tuned by iterative tries. This resulted in 136 remaining features in the first filter and 60 after the second. The results were tested in the same parameters as the baseline model, and can be seen in table 6.6.

Table 6.6: Results of the different correlation filtered feature sets on random forest classifier with 1000 estimators.

Data set	**No features**	**Results**	
		Accuracy	**Bias**
All features	980	54.82	0.679
Baseline (Raw data)	320	54.51	0.5755
Remove low target correlation only	136	54.3	0.2173
Correlation selected (0.9, 0.2)	60	53.7	0.2146

There is a clear trend occurring in the results. With more features, the results tend to get better, and the algorithm used (random forest) seem to be able to handle them.

6.3.3.3 Tree-based selection

The last explored option was based on the random forest classifier feature importance. Since the tree-based models need to calculate feature importance in order to create the decision trees, we can use this calculation to verify what features the model considers important.

The procedure was to train a random forest classifier with 5000 estimators in the whole data and then select the top k features in terms of feature importance, with which another random forest classifier will be trained and then check for improvements in the results.

Once again, the results shown in figure 6.4 confirm that the more features are used the better performance the algorithm has. Since none of the selection techniques showed big improvements over using all the features in the feature set, the full feature set will be used in the modeling phase.

The chosen baseline algorithm, random forest, has the ability to adapt in order to select and lower the impact of the bad features by himself. This hid some of the bad features, making the choice of this algorithm for a baseline unwise.

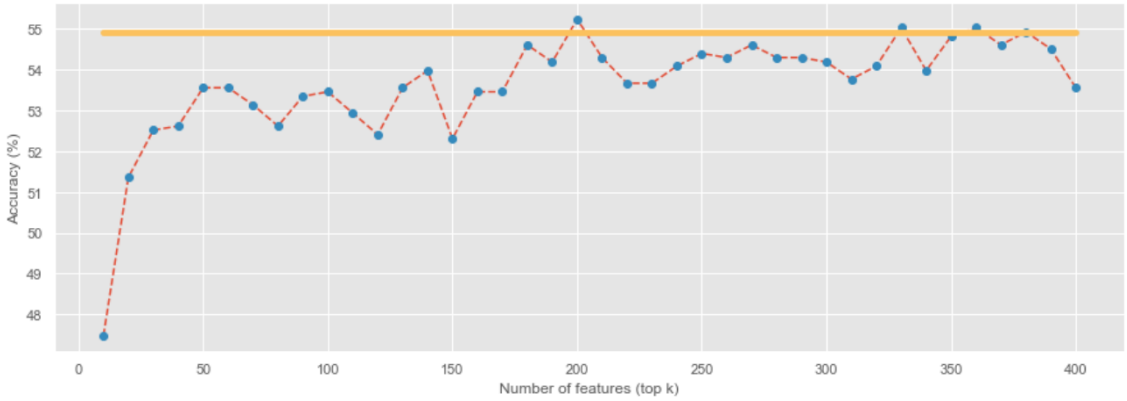


Figure 6.4: Performance of the top k features in feature importance tested on a random forest classifier with 1000 estimators. The line in yellow represents the accuracy obtained using all features.

6.4 Modeling

6.4.1 Tree-based models

6.4.1.1 Decision tree

To tune the decision tree classifier, the first step was to define a baseline. For that, the default parameters of the sklearn classifier were used. The relevant hyperparameters to this experiment were the max_depth=None, min_samples_split=2 and min_samples_leaf=1. By running with these hyperparameters, the tree would split their branches as long as there were at least 2 instances for a split to occur and it resulted in leaves with at least 1 sample. The results are shown in the table 6.7.

Table 6.7: Average results of 100 runs of the decision tree classifier with sklearn's default parameters. Both strategies yield equal results due to the predicted probabilities being in the form [1, 0, 0].

Predicting 954 games	**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**
Prediction accuracy	45.76%		45.76%	
Total profit	20.84	-9.09	20.84	-9.09
Expected loss	-5.0%	-5.0%	-5.0%	-5.0%
ROI	2.18%	-1.78%	2.18%	-1.78%
	Bias	1.0971	**Variance**	0.4795

Even though the results were acceptable for a weak learner, they were definitely improvable. Notably, the variance was quite high, and therefore the hyperparameters described before, max_depth, min_samples_split and min_samples_leaf could be tuned in order to reduce variance.

There are two approaches to tune these hyperparameters: the first is to define a state space and test all the possible combinations, or using optimization methods to select the better combination, to then choose the best performing hyperparameters. The second approach would be to manually adjust the model to obtain a more optimal model. Since this is a simple model that is fast to train, we will use the first approach.

The most influential parameter in the decision tree is the max_depth, because the lower the depth of the tree, the less influential the min_samples_split and min_samples_leaf hyperparameters will be. Therefore, the max_depth parameter optimization should be prioritized.

The metrics that are expected to improve are the increase in accuracy, variance reduction and potentially, bias reduction. The test will consist of running the algorithm 100 times with the max_depth going through the interval [1:30]. The results are in figure 6.5.

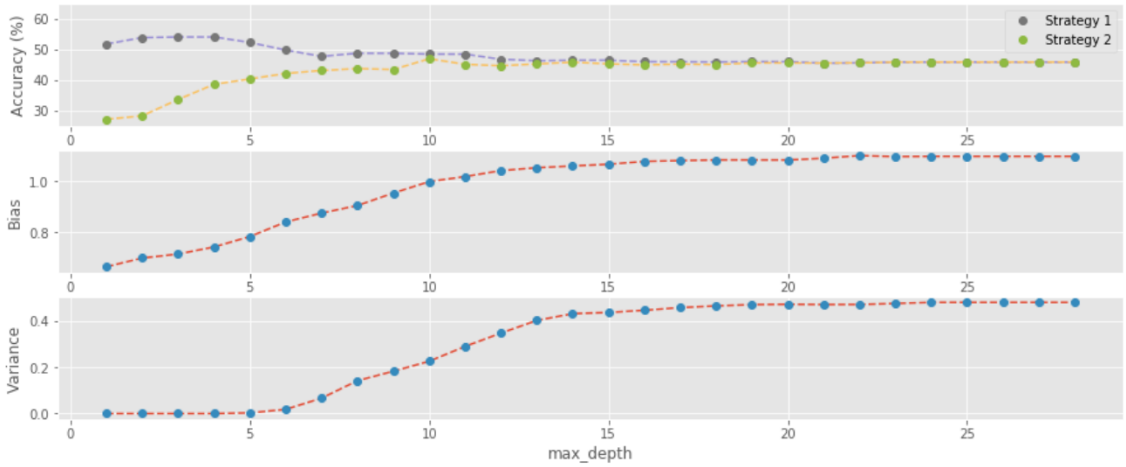


Figure 6.5: Optimal hyperparameter (max_depth) search in the decision tree classifier.

By checking the results, it is verifiable, in terms of bias and variance, the lower the max_depth, the better the results. However, accuracy metrics shows two different scenarios. By using strategy 1, betting on the favorite, the peak accuracy is in the range max_depth = [2, 4]. If we consider the strategy 2, the peak only comes when max_depth = 10. This leads to the conclusion that both strategies peak at different stages.

Since the ideal max_depth is much lower than the number of instances that the model is using, the impact of other parameters such as min_samples_split and min_samples_leaf is mitigated, therefore a further exploration of the state space is unnecessary. There are more hyperparameters that could be explored, however, the bulk of the improvement that could be made in this method was already achieved. Table 6.8 give us a more detailed view of the results.

These results outperform the expected. Even though the variance was reduced it is expected for the model to overfit for new data.

Table 6.8: Average results of 100 runs of the decision tree classifier predictions on the test set.

	max_depth = 3				*max_depth = 10*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Prediction accuracy	53.98%		33.54%		48.38%		46.85%	
Total profit	0.41	-14.4	-68.21	-19.96	48.36	29.97	48.63	25.69
Expected loss	-5.0%	-5.0%	-5.25%	-5.25%	-5.0%	-5.0%	-5.01%	-5.01%
ROI	0.04%	-3.77%	-7.5%	-10.44%	5.07%	0.21%	5.1%	0.08%
	Bias	0.7129	**Variance**	0	**Bias**	0.9978	**Variance**	0.2254

6.4.1.2 Decision tree ensembles

The next step to improve the predictions using decision trees is to ensemble them. Two of the go-to algorithms in this scenario are bagging and AdaBoost, with the focus on reducing variance and bias respectively. Both algorithms were tested using 50 estimators in the ensemble and the other hyperparameters were the default. The results are shown in table 6.9.

Table 6.9: Average results of 50 runs of decision tree ensembles (bagging and AdaBoost) in the test set, using 50 estimators.

	max_depth = 3				*max_depth = 10*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Bagging								
Prediction accuracy	54.21%		27.43%		53.64%		33.64%	
Total profit	-21.1	-17.66	-63.42	-19.93	-31.74	-17.8	-39.8	-19.06
Expected loss	-5.0%	-5.0%	-5.56%	-5.56%	-5.0%	-5.0%	-5.27%	-5.27%
ROI	-2.21%	-3.71%	-7.39%	-8.52%	-3.33%	-6.27%	-4.39%	-7.56%
	Bias	0.696	**Variance**	0.0411	**Bias**	0.7127	**Variance**	0.1498
AdaBoost								
Prediction accuracy	45.02%		24.48%		51.43%		49.92%	
Total profit	-100.09	-19.97	-37.3	-19.91	-25.98	-16.79	-19.96	-16.39
Expected loss	-5.0%	-5.0%	-5.06%	-5.06%	-5.0%	-5.0%	-5.0%	-5.0%
ROI	-10.49%	-15.46%	-3.96%	-9.18%	-2.72%	-5.37%	-2.09%	-5.34%
	Bias	0.6663	**Variance**	0.0078	**Bias**	0.8402	**Variance**	0.6099

As expected, the bagging approach was able to improve both of the decision tree predictions from the single models, in terms of classification (accuracy in strategy 1) and by reducing the variance on the more complex models by 1/3. The measured bias was also slightly improved. The improvement was bigger in the more complex models (max_depth = 10). However, strategy 2 numbers are considerably worse.

For the boosting technique, the expected bias reduction occurred in both types of models, however, bagging was able to better reduce bias when the models were more complex. This confirms that the AdaBoost algorithm works better with less complex models (weak learners), such as when max_depth=3, where it is able to outperform bagging in terms of bias. However, this bias reduction wasn't able to be capitalized into an accuracy improvement, which leads to disastrous results in terms of ROI.

These results were underwhelming, since neither of the methods was able to maintain the specific business metrics values from the decision trees, leading to heavy losses when compared to single models.

Further testing was done for bagging with 1000 estimators, for future comparisons with other ensemble methods. Results shown in table 6.10.

Table 6.10: Average results of 10 runs of decision tree bagging ensemble in the test set, using 1000 estimators.

	max_depth = 3				*max_depth = 10*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Prediction accuracy	54.39%		26.87%		54.44%		32.1%	
Total profit	-16.8	-17.25	-75.62	-19.97	-26.39	-18.15	-33.42	-19.7
Expected loss	-5.0%	-5.0%	-5.57%	-5.57%	-5.0%	-5.0%	-5.52%	-5.52%
ROI	-1.76%	-3.23%	-8.83%	-7.43%	-2.77%	-5.07%	-3.86%	-6.34%
	Bias	0.6956	**Variance**	0.0089	**Bias**	0.7122	**Variance**	0.0327

6.4.1.3 Random forest

Another ensemble method tested, this time a decision tree exclusive, was the random forest. This method had already been used in the feature selection evaluation, and it is important to summarize the results of this method in all the metrics.

In order to maintain the results comparable with the previous ensemble methods a low number of estimators (50) was chosen. The recommended number of estimators is much higher (1000+), and so, a test with 1000 estimators is also reported. The tests were made with no max_depth limitation, max_depth = 3 and max_depth = 10. The results are in table 6.11 for the test with 50 estimators and in the table 6.12 for the test with 1000 estimators.

Table 6.11: Average results of 50 runs of random forest classifier in the test set, using 50 estimators.

	max_depth = 3				*max_depth = 10*				*max_depth = None*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Prediction accuracy	54.81%		25.17%		54.36%		30.81%		53.17%		30.11%	
Total profit	-10.71	-16.01	-96.45	-19.99	-20.89	-16.55	-63.2	-19.82	-34.38	-18.14	-64.37	-19.87
Expected loss	-5.0%	-5.0%	-5.34%	-5.34%	-5.0%	-5.0%	-5.34%	-5.34%	-5.0%	-5.0%	-5.24%	-5.24%
ROI	-1.12%	-2.11%	-10.79%	-13.74%	-2.19%	-4.19%	-7.07%	-8.56%	-3.6%	-5.88%	-7.07%	-8.65%
	Bias	0.679	**Variance**	0.0398	**Bias**	0.7023	**Variance**	0.1417	**Bias**	0.698	**Variance**	0.1829

Looking into classification results, the random forest has the best performance yet, with the max_depth=3 beating the previous record. Even though the results of specific business metrics improved over other ensemble methods, they are still negative. When looking at the bias and variance metrics they seem to be less dependent on the hyperparameters, in opposition to the previous decision tree ensembles tested. However, both bagging and AdaBoost at their best results beat the random forest in terms of variance and bias respectively.

Table 6.12: Average results of 10 runs of random forest classifier in the test set, using 1000 estimators.

	max_depth = 3				*max_depth = 10*				*max_depth = None*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Prediction accuracy	54.74%		25.3%		55.22%		28.01%		54.72%		26.08%	
Total profit	-11.26	-16.32	-88.36	-19.99	-12.65	-16.26	-59.99	-19.95	-23.63	-17.74	-103.69	-19.99
Expected loss	-5.0%	-5.0%	-5.33%	-5.33%	-5.0%	-5.0%	-5.65%	-5.65%	-5.0%	-5.0%	-5.57%	-5.57%
ROI	-1.18%	-2.13%	-9.87%	-13.76%	-1.33%	-3.01%	-7.1%	-6.67%	-2.48%	-4.71%	-12.11%	-7.12%
	Bias	0.6788	**Variance**	0.0087	**Bias**	0.7031	**Variance**	0.0306	**Bias**	0.6982	**Variance**	0.0396

When comparing the results done with more estimators there is an improvement over the fewer estimators variant. It was expected a reduction in variance with a low impact on the bias. The results confirm the theory. However, both decision tree bagging and random forest yield similar results, with the biggest difference between the algorithms being the time needed for execution, with bagging being much more time demanding than the random forest.

6.4.1.4 Gradient boosting and XGboost

While the random forest algorithm seemed to improve the variance of the models, with gradient boosting the goal is to reduce the algorithm bias. XGBoost is an improved version of the gradient boosting algorithm, and therefore is expected to perform better. It does not support having no max_depth parameter, so it was set to the max_depth=50. The algorithm executed with 50 estimators and default parameters, since the main goal is to compare with the AdaBoost algorithm. The results are in table 6.13.

Table 6.13: Average results of 50 runs of both gradient boosting techniques in the test set, using 50 estimators.

	max_depth = 3				*max_depth = 10*				*max_depth = 50*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Gradient Boosting												
Prediction accuracy	53.07%		34.28%		51.68%		46.59%		44.35%		39.68%	
Total profit	-41.3	-18.96	-23.75	-19.26	-47.43	-19.15	-52.71	-19.4	-56.51	-19.79	-55.84	-19.85
Expected loss	-5.0%	-5.0%	-5.34%	-5.34%	-5.0%	-5.0%	-5.05%	-5.05%	-5.0%	-5.0%	-5.03%	-5.03%
ROI	-4.33%	-2.92%	-2.65%	-3.78%	-4.97%	-8.48%	-5.58%	-8.46%	-5.92%	-13.2%	-5.88%	-10.63%
	Bias	0.7165	**Variance**	0.0328	**Bias**	0.8123	**Variance**	0.2045	**Bias**	1.0188	**Variance**	0.0753
XGBoost												
Prediction accuracy	54.19%		32.91%		51.47%		45.91%		54.51%		48.22%	
Total profit	-29.53	-18.5	-40.44	-19.8	-62.47	-19.74	-15.21	-18.66	34.69	8.57	37.12	-2.3
Expected loss	-5.0%	-5.0%	-5.37%	-5.37%	-5.0%	-5.0%	-5.08%	-5.08%	-5.0%	-5.0%	-5.06%	-5.06%
ROI	-3.1%	-3.26%	-4.55%	-2.83%	-6.55%	-9.13%	-1.62%	-5.71%	3.64%	0.59%	3.94%	-0.18%
	Bias	0.7146	**Variance**	0.0	**Bias**	0.7977	**Variance**	0.0	**Bias**	0.817	**Variance**	0.0

As expected, the XGBoost outperformed the gradient boosting algorithm in nearly every metric and strategy.

When comparing with the AdaBoost technique, both boosting techniques yield superior results. The only relevant metric that they did not outperform the AdaBoost was the bias when max_depth = 3. This leads to the conclusion that, in this problem, gradient boosting techniques are superior to the AdaBoost.

For the first time since the decision tree classifier, positive business specific metrics were obtained when using the XGBoost with max_depth = 50.

The ensemble techniques were able to present more reliable results due to a heavy reduction in variance, allowing them to present much more consistent predictions. Even though their performance in the business-specific metrics was inferior to the decision trees, the metrics are more precise and approximate to the real expected values.

6.4.2 Distance-based models

6.4.2.1 k-nearest neighbor

The k-nearest neighbors algorithm only requires one parameter, the number of neighbors (k). The optimal k value can be easily found by searching the state space, which can be visualized in figure 6.6.

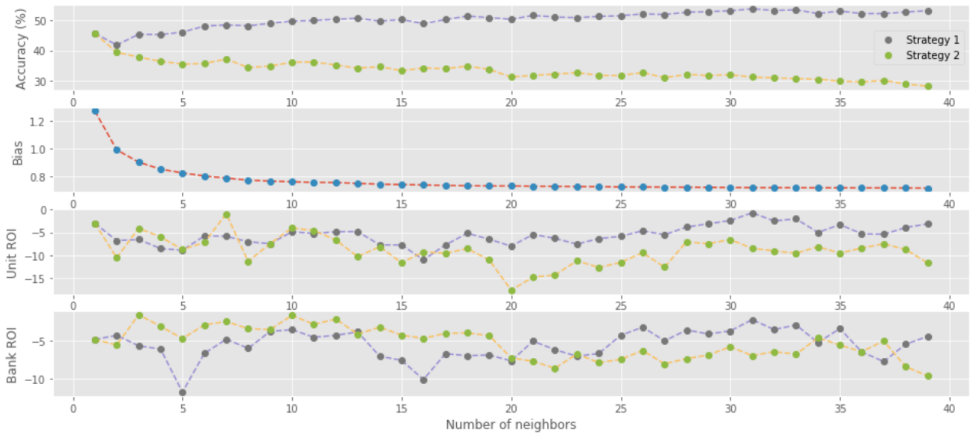


Figure 6.6: Optimal hyperparameter (k) search in the k-nearest neighbor.

In the results, we can see that the bigger the number of neighbors, the better the results are. However, after a certain number of neighbors, the results seem to hit a plateau. This can be easily observed in the bias plot, where improvement is scarce above the 10 neighbors. Since both the accuracy metric and the ROI continue to improve after the k = 10 until k = 30, the latter will be chosen to be the hyperparameter value. Only one run of the algorithm was made since the k-NN does not have intrinsic variability, and it is shown in table 6.14.

Table 6.14: Results of a single run of the k-nearest neighbors in the test set.

Predicting 954 games	**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**
Prediction accuracy	54.4%		34.8%	
Total profit	2.99	-13.29	-66.09	-19.94
Expected loss	-5.0%	-5.0%	-5.17%	-5.17%
ROI	0.31%	-2.14%	-7.17%	-5.58%
	Bias	0.7337	**Variance**	0

6.4.2.2 Ensemble k-NN

As the k-NN does not have intrinsic randomness, the variability that is needed in order for ensemble models to improve the algorithms has to come due to changes in the data set, for example, using bagging and AdaBoost. However, sklearn's implementation of k-NN does not support the AdaBoost. Due to that, only the bagging test was done.

Table 6.15: Average results of 50 runs of the bagging with k-nearest neighbors classifiers in the test set, using 50 estimators.

Predicting 954 games	**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**
Prediction accuracy	51.65%		34.79%	
Total profit	-35.14	-18.23	-83.67	-19.96
Expected loss	-5.0%	-5.0%	-5.09%	-5.09%
ROI	-3.68%	-5.81%	-8.93%	-10.61%
	Bias	0.7311	**Variance**	0

By checking table 6.15, we conclude that there is no improvement in using bagging. One of the reasons is that, as the bagging algorithm reduces the number of instances used, the k=30 value might not be the best. Further testing showed that even when adjusting k the results didn't improve. Another tested solution was drawing features with no replacement, however, to no success. Removing instances from the k-NN algorithm itself also damages its performance. Therefore, the conclusion is that k-NN is an algorithm that does not improve when ensembled via bagging in our circumstances.

6.4.3 Neural networks

In order to be able to do the same magnitude of tests in neural networks that were made in decision trees, a deeper feature selection had to be made in order to reduce the training time of the networks, especially from the ensemble methods. The new feature set is composed by the number of expected goals scored and conceded from the last 7 games for both teams, totaling 28 features. This way the network can be scaled down and trained faster. To enable more options while configuring the neural networks, Keras was the chosen library.

The first step is to find the ideal network architecture. By running a test with the default configurations that evaluated a set of architectures it is possible to approximate the ideal architecture. This test is exhibited in figure 6.7.

In the chosen test instances the best classification results came when only one layer was used, however, it looks like adding a layer seem to improve the business specific metrics. Two architectures were chosen in order to continue testing: (15,) and (10,5). All the layers use softmax as the activation function.

There are two hyperparameters that are also going to be evaluated: early stopping and dropout. While the dropout technique target is variance reduction, the early stopping increases the variability. For the models with early stopping, 500 epochs were used, with patience=10. The

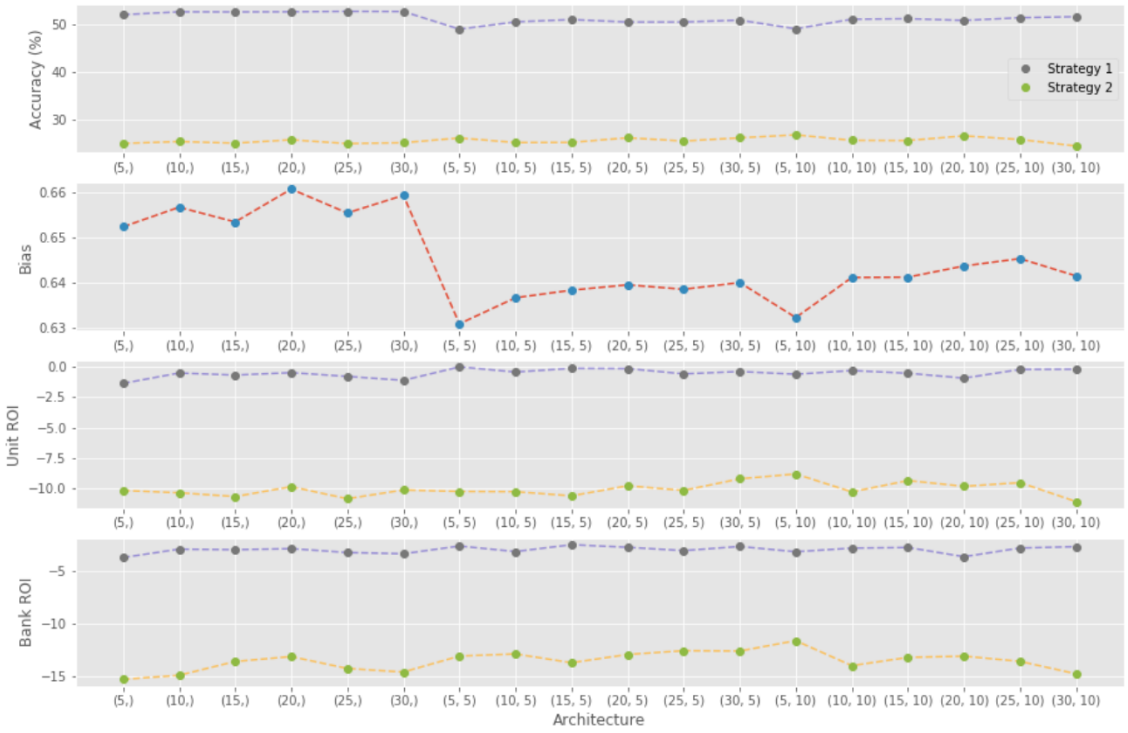


Figure 6.7: Optimal hyperparameter (network architecture) search for neural networks.

other models use 100 epochs. In models with dropout, the dropout_rate is 0.3. All the models used batch_size = 200 and learning rate of 0.025. The chosen loss function was categorical_crossentropy. These fixed hyperparameters were tuned by iterations. The tests are shown in table 6.16.

Table 6.16: Average results of 50 runs of the neural networks in the test set.

	(15,)				*(10,5)*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
without early stop								
Prediction accuracy	52.44%		31.25%		53.14%		28.83%	
Total profit	9.7	-10.97	-49.32	-19.79	8.67	-11.39	-65.41	-19.93
Expected loss	-5.0%	-5.0%	-5.19%	-5.19%	-5.0%	-5.0%	-5.36%	-5.36%
ROI	1.02%	-2.59%	-5.36%	-9.42%	0.91%	-2.12%	-7.35%	-13.92%
	Bias	0.6695	**Variance**	0.1016	**Bias**	0.6702	**Variance**	0.0719
with early stop								
Prediction accuracy	51.2%		31.66%		52.55%		29.44%	
Total profit	7.6	-11.53	-44.41	-19.69	1.67	-13.18	-66.41	-19.91
Expected loss	-5.0%	-5.0%	-5.16%	-5.16%	-5.0%	-5.0%	-5.33%	-5.33%
ROI	0.8%	-3.03%	-4.81%	-9.45%	0.18%	-2.97%	-7.4%	-13.98%
	Bias	0.6699	**Variance**	0.2018	**Bias**	0.6705	**Variance**	0.1497

After the first tests on the simpler networks, it is possible to state the differences between decision trees and neural networks in terms of performance. The neural networks are able to fit the data better, as verifiable by a lower bias. However, these results can be influenced by different feature sets.

As for the differences between using or not early stopping are in the variance created in the early stop version. This variance might allow us to improve the results of the ensemble models.

6.4.3.1 Neural network ensemble

More tests were done with neural networks, now focusing on ensembles and their performance. The tested algorithms were bagging, average dropout and AdaBoost.

The average dropout consists on making predictions by using the average probabilities of the several dropout neural networks. In bagging, a small caveat was added. We added random subspace sampling, the feature sampling method from the random forest. The feature_ratio used was 0.5, the same as the instance_ratio, both iterativelly tuned. The AdaBoost corresponds to an attempt of adaptation of the algorithm for the neural networks.

Table 6.17: Average results of 50 runs of the neural networks ensemble techniques in the test set (without early stopping).

	(15,)				*(10,5)*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Bagging								
Prediction accuracy	53.4%		30.46%		53.49%		28.33%	
Total profit	8.64	-11.48	-50.88	-19.89	2.59	-13.63	-71.92	-19.96
Expected loss	-5.0%	-5.0%	-5.22%	-5.22%	-5.0%	-5.0%	-5.4%	-5.4%
ROI	0.91%	-1.96%	-5.58%	-8.43%	0.27%	-2.33%	-8.14%	-15.52%
	Bias	0.6697	**Variance**	0.0399	**Bias**	0.6700	**Variance**	0.0323
Average Dropout								
Prediction accuracy	53.49%		28.61%		53.49%		25.61%	
Total profit	1.48	-14.43	-83.17	-19.98	11.76	-11.04	-91.9	-19.99
Expected loss	-5.0%	-5.0%	-5.2%	-5.2%	-5.0%	-5.0%	-5.21%	-5.21%
ROI	0.16%	-2.8%	-9.07%	-11.86%	1.23%	-1.41%	-10.04%	-13.15%
	Bias	0.6704	**Variance**	0.0099	**Bias**	0.6730	**Variance**	0.0096
AdaBoost								
Prediction accuracy	50.9%		34.65%		51.63%		33.17%	
Total profit	-6.89	-15.93	0.23	-16.88	-6.71	-13.71	8.49	-15.48
Expected loss	-5.0%	-5.0%	-5.14%	-5.14%	-5.0%	-5.0%	-5.18%	-5.18%
ROI	-0.72%	-2.85%	0.0%	-6.22%	-0.7%	-3.57%	0.89%	-4.84%
	Bias	0.6715	**Variance**	0.1768	**Bias**	0.6718	**Variance**	0.1675

In the results without the use of early stopping, table 6.17, it is possible to observe improvement in the classification performance in the bagging and average dropout, due to an increase in the accuracy. With the main target being the variance reduction both bagging and average dropout techniques were able to heavily reduce it.

The AdaBoost model failed to make any sizable improvement.

Table 6.18: Average results of 50 runs of the neural networks ensemble techniques in the test set using early stopping.

	(15,)				*(10,5)*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Bagging								
Prediction accuracy	52.4%		29.96%		53.43%		25.35%	
Total profit	2.13	-13.31	-67.7	-19.95	8.69	-10.99	-108.05	-19.99
Expected loss	-5.0%	-5.0%	-5.17%	-5.17%	-5.0%	-5.0%	-5.19%	-5.19%
ROI	0.22%	-2.18%	-7.33%	-8.27%	0.91%	-1.89%	-11.76%	-17.21%
	Bias	0.6701	**Variance**	0.0747	**Bias**	0.6738	**Variance**	0.0586
Average Dropout								
Prediction accuracy	53.81%		26.53%		53.49%		24.48%	
Total profit	6.4	-12.81	-102.08	-19.99	25.96	-3.03	-106.97	-20.0
Expected loss	-5.0%	-5.0%	-5.3%	-5.3%	-5.0%	-5.0%	-5.11%	-5.11%
ROI	0.67%	-1.64%	-11.34%	-13.95%	2.72%	-0.38%	-11.46%	-10.48%
	Bias	0.6712	**Variance**	0.0096	**Bias**	0.6760	**Variance**	0.0174
AdaBoost								
Prediction accuracy	50.77%		35.53%		51.45%		34.16%	
Total profit	1.03	-13.28	-6.79	-17.13	-3.32	-12.67	-0.18	-14.37
Expected loss	-5.0%	-5.0%	-5.12%	-5.12%	-5.0%	-5.0%	-5.13%	-5.13%
ROI	0.11%	-3.06%	-0.74%	-7.59%	-0.35%	-3.39%	-0.01%	-5.96%
	Bias	0.6709	**Variance**	0.2248	**Bias**	0.6722	**Variance**	0.2407

The improvements in ensembles with early stopping, table 6.18, were similar to the no early stopping variation, but the performance was generally worse than the models with no early stopping. Even though the variance results are still improved over the single models, they fail to be better than the approach with no early stop. This leads to the conclusion that the induced variation from the early stopping parameter failed to make a positive impact.

6.4.3.2 Negative correlation learning

The negative correlation learning algorithm was also evaluated. However, the mean squared error was not used as recommended (Liu and Yao, 1999). It was important to keep the categorical cross-entropy loss function due to performance and to be able to better compare with the other neural network ensembles.

The new loss function is now:

$$loss\ function = categorical\ crossentropy(y,\gamma) - \lambda * categorical\ crossentropy(\varepsilon,\gamma) \quad (6.1)$$

Where y is the label, γ the model prediction and ε the ensemble prediction. The used lambda was 0.2, tuned by experimenting different values. The results are shown in table 6.19.

Table 6.19: Average results of 50 runs of the negative correlation learning algorithm, with lambda = 0.2, in the test set.

	(15,)				*(10,5)*			
Predicting 954 games	**Strategy 1**		**Strategy 2**		**Strategy 1**		**Strategy 2**	
	Unit	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**	**Unit**	**Bank**
Prediction accuracy	53.03%		40.41%		53.91%		40.4%	
Total profit	14.53	-10.34	5.24	-17.33	29.42	0.49	4.08	-17.01
Expected loss	-5.0%	-5.0%	-5.12%	-5.12%	-5.0%	-5.0%	-5.26%	-5.26%
ROI	1.52%	-1.72%	0.56%	-2.84%	3.08%	-0.0%	0.45%	-3.6%
	Bias	0.6652	**Variance**	0.0331	**Bias**	0.6653	**Variance**	0.0295

The best architecture was the (10,5). The results beat the previous ensembles of neural networks, both in the classification, the accuracy is higher than any other instance of neural networks, and regression components, with the lower bias when comparing with all the other models tested and a positive ROI in strategy 2.

6.5 A survey of the state-of-the-art algorithms

To compile the results obtained in the modeling experimentation, the best performing algorithms were chosen:

- Random forest with max_depth=10
- XGBoost with max_depth=50
- Bagging neural network (architecture (15,)) without early stopping
- Average dropout (architecture (10,5)) without early stopping
- Negative correlation learning (architecture(10,5), λ=0.2)

From this point onwards the focus was on the strategy 1 with the unit betting approach. The results in the test sets are represented in figure 6.8.

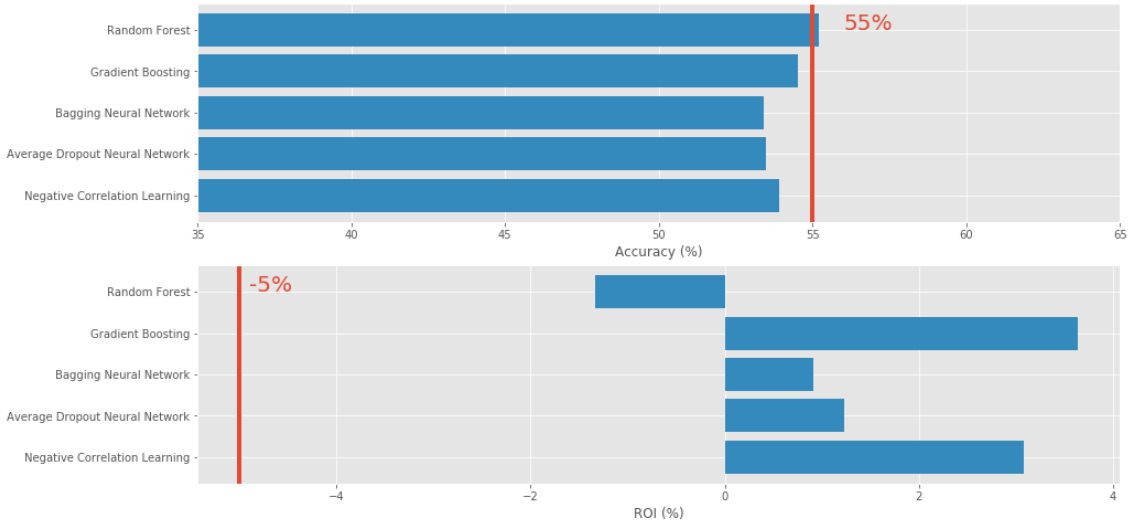


Figure 6.8: Average results of the best performing models in the test set. The red marker represents the expected values according to the bookmaker's data.

The next step is to validate the results in the validation set, in order to be sure that the results are generalizing well. The results are visible in figure 6.9.

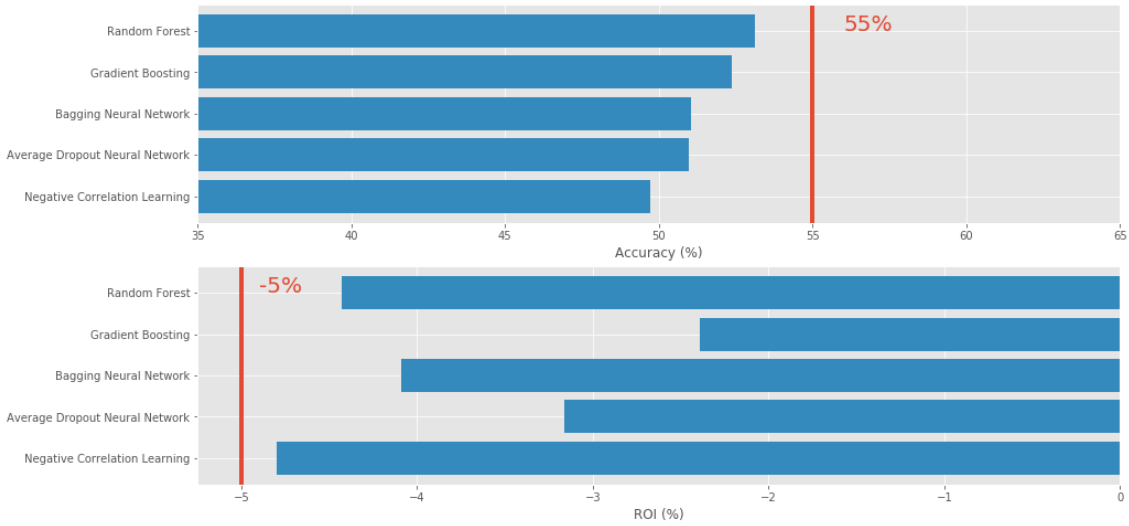


Figure 6.9: Average results of the best performing models in the validation set. The red marker represents the expected values according to the bookmaker's data.

Even though the models performed above the expected value in the business-specific metrics, the performance in validation is worse than in the test. This reduction in performance was expected, due to the model tuning being based on the test set which might have led to some overfitting. Another possible cause is that the approach to the problem was wrong. Some of the factors can be that data should be split by leagues or due to the fact that the 70/30 train/test split was including in the training set data from the season where the tests were made, something that did not happen *in validation* since no data from the 18/19 season was included in the training set. Both situations are evaluated next.

6.6 Result optimization

6.6.1 Intra-league testing

The first attempt to optimize the results previously obtained is to split the data by their respective leagues. As stated before, the problem with this approach is that there are very few instances to train and test the model. The results are in figure 6.10.

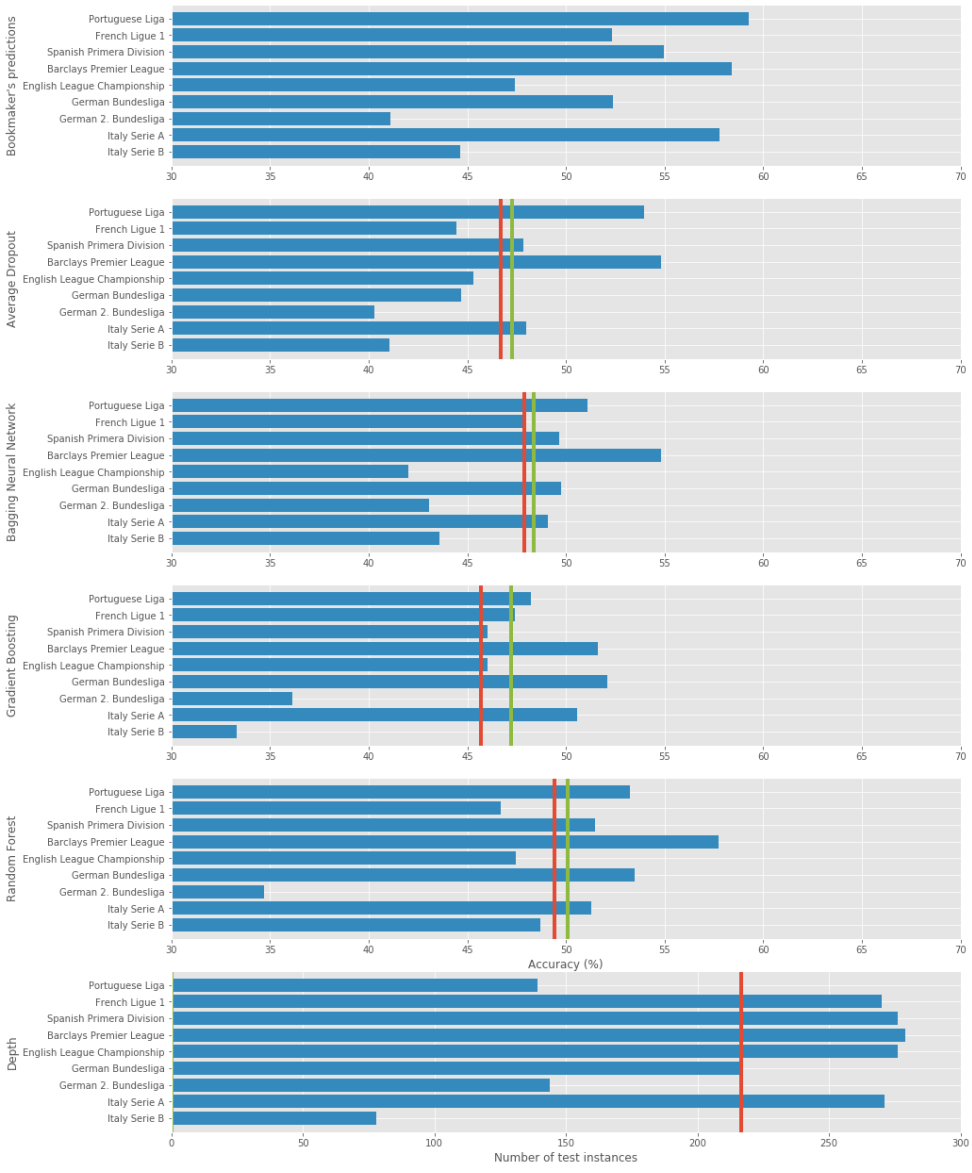


Figure 6.10: Average results of the tests in separated leagues. The red marker is the average value of the results from the leagues, the green marker is the weighted average of the results from the leagues with the weight being the number of test instances.

As it is possible to observe, the fear that the existing data was scarce for this method is proven by the fact that the leagues with more data are performing closer to the expected (the accuracy

in these leagues is closer to the bookmaker's accuracy than in the other leagues). The averaged results in none of the used algorithms are higher than the results previously obtained on the whole data set. While splitting the data into different data sets from different leagues are expected to improve the results, the amount of training data needed does not allow for that conclusion to be drawn from the results. For now, and in the data that is accessible, the results are worse when the leagues are split, leading us to the conclusion that the increased number of instances improves the performance of the algorithms, even when considering the different characteristics from the leagues.

6.6.2 Intra-season training

The second explorable scenario is that including a sample of the season's data in the training set might improve the algorithm's performance. This approach evaluates if the more recent data instances represent a better sample of the data than the rest of the data set, and therefore improve the results. In the current state, 0% of the data instances used in training are from the season from where the tests are drawn. The test represented in the figure 6.11 has approximately 40% of the new season is present in the training set.

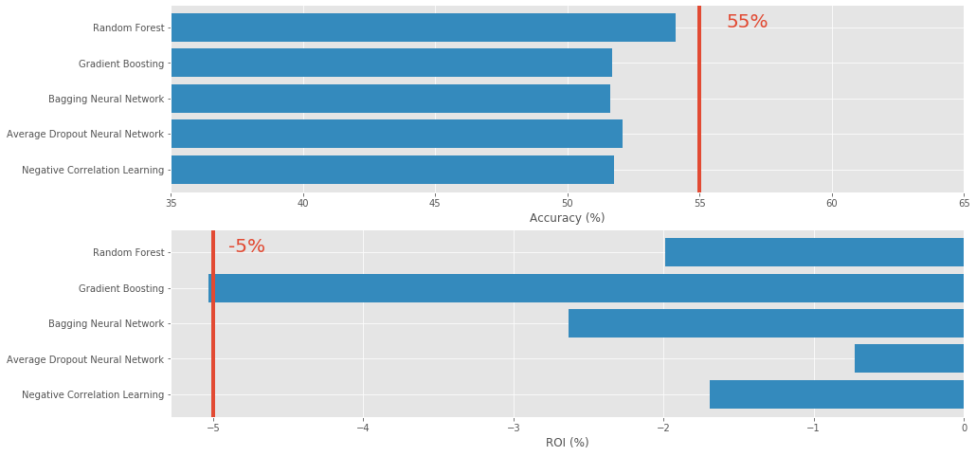


Figure 6.11: Average results of the validation tests in a smaller validation set. The red marker represents the expected values according to the bookmaker's data.

The results here are mixed. In some cases, random forest, average dropout and negative correlation learning, the classification performance improves over the last attempted validation. On the business specific metrics, the improvement is sizable.

Although some results point towards this being a solution, it is also necessary to take into account that the models might only improve because we are removing from the test set the early part of the season that might be harder to predict. This is explainable since the teams usually take time until they perform at their full potential, which leads to more irregular results at the beginning of the season.

6.6.3 Features and probability distributions

Due to the reduction of features available to the neural network in relation to the other algorithms, it was expected that the results of these tests show worse perform than the decision trees.

This actually happened when we consider the accuracy metric. Performance of the neural network ensembles is generally worse than the decision tree ensembles.

In figure 6.12, we can verify that the features used in the neural networks handicaped them in relation to other methods.

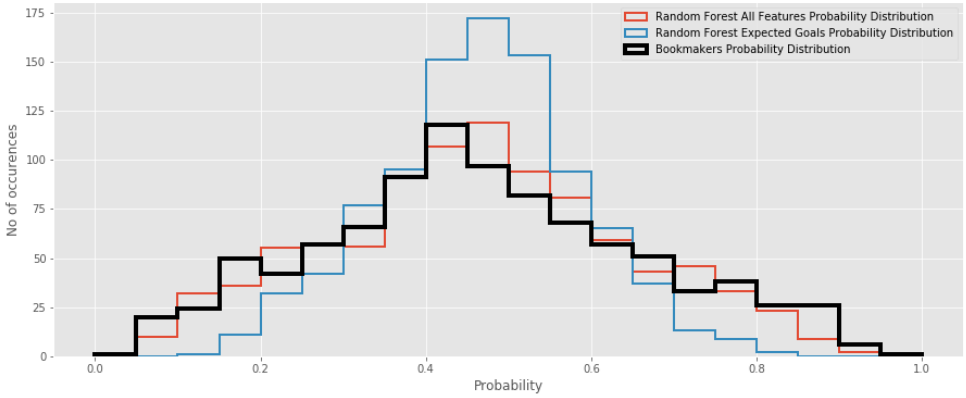


Figure 6.12: Probability distribution of the models home team predictions. In red we can see the distribution of probabilities when a Random Forest algorithm is trained with all features (980 in total), in blue when using only expected goals (28 in total). The black line represents the probabilities of the bookmakers.

By check the histograms in figure 6.12, we can verify that, when the random forest has all the features available, the resulting probabilities follow the distribution of the bookmakers probabilities. The same does not happen when we use only the expected goals.

When we use expected goals the distribution of the probabilities change, not being able to follow the bookmakers. As a result, the predictions are narrowed towards the mean. Probabilities over 0.8 are rare, even thought they have a significant portion of the bookmakers predictions. Same for probabilities under 0.15.

The generation and selection of features has a massive impact on the predictions, and the difference between the feature sets on the decision tree based models and the neural network based models does not allow for a fair comparison between the models.

Chapter 7

Conclusions

From the point of view of a machine learning problem, the classification results were good. They replicate results from other work done in the area of sports prediction, and while an improvement over previous work did not occur, the evaluation of certain metrics, such as the ROI, summarize these results on a new context.

In the regression problems, the results were not so good. Strategy 2 that was designed to profit from betting failed to yield any profit. However, most algorithms are able to be better than the expected ROI. That leads us to the conclusion that the probabilities generated from the machine learning algorithms are, at most, slightly better than the bookmakers' probabilities.

The model's tunning approach was not optimal since it focused on its performance on the test set, and it leads to overfitting in some models. This is a problem that occurs from the use of split validation since the test is reliant on only one result.

The overall results in terms of business perspective were reasonably good. The models were able to outperform the expected value, which, in an environment where the expected value is closer to 0%, such as betting exchanges, these models will yield profit. There are various components that can influence this profit positively that were not attempted. For example, exploring the variations of the odds from the bookmakers, since the used odds corresponded to the odds at the last moment before the game started. While no academic research was found on the topic, the general consensus between the betting community is that the worst time to bet in a game is right before it starts.

Unless there is access to sports betting exchanges, this models has its best value as a decision support system. It would be interesting to analyze how these models behave as decision support systems, where the bettor could introduce modifiers such as injuries, suspensions or even importance factors to the matches, dependent on European competitions played or if the team still has goals to reach in the league. The bettor could simply agree or discard any suggestion from the algorithm. Some of this information could even be stored to help future models to predict the games based not only in plain statistical data but also based on the opinion of the bettor himself.

7.1 Future work

Data preparation phase was let down by the feature selection, where the results were underwhelming. A different set of techniques could be tested in order to improve results, including in feature generation, namely Principal Component Analysis. Integration of the new sources of data, either by cleaning the assessed sources or finding additional cleaner sources would also lead to improvements.

Split-validation lead to some difficulties in stabilizing the business specific metrics due to the lack of an average that would reduce the variance of the results. The use of cross-validation, even when considering the possible use of future data influencing the results, should allow for more reliable results. A better solution would be that when more data becomes available, a sliding window could be used in order to generate different train/test sets.

There are some approaches that due to timing constraints were not able to be tested such as:

- Optimizing the models to maximize ROI instead of minimizing a loss function to maximize the accuracy.
- Multi-view approach to combine the predictions from several models.

Finally, in the conclusions, the suggestion was to use these models as a decision support system. The final CRISP-DM methodology step is deployment. This would require the total automation of the data acquiring process and some better interface implemented to access the predictions. Since the scope of the project was focused on modeling this point was deprioritized.

Appendix A

The expected goals metric

The expected goals metric (Opta) is measure of a shot quality. It is calculated from the likelihood of a shot ending in a goal, taking into account factors such as distance to the goal, angle of the shot, body part used to make the shot and whether it was a first touch shot or not.

From the previous shots data, it is possible to create a gradient that allows for estimation of the new shot data. An example of this gradient can be seen in figure A.1.

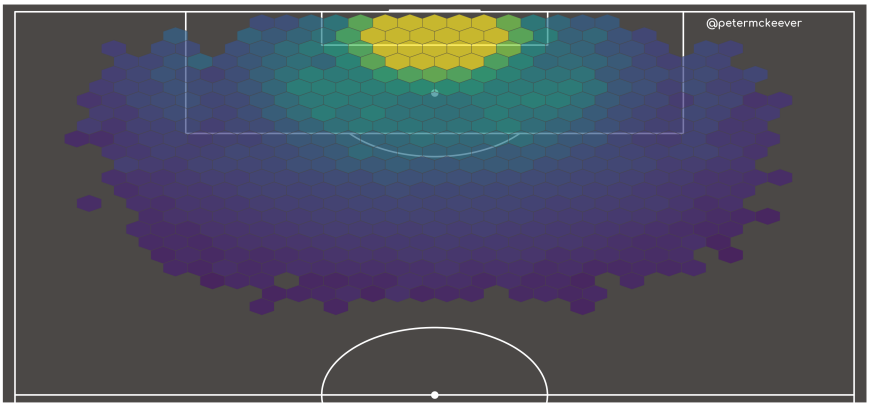


Figure A.1: Visualization of the expected goals from the different areas of the pitch, with the yellow hexagons representing the higher probabilities.
Source McKeever (2019)

This metric allows us to abstract from the binary goal metric. By incorporating the probabilities we can obtain a better representation of how the game was played between both teams, reducing the randomness associated with football.

$$expected\ goals = \sum_{for\ all\ shots} P(shot\ leading\ to\ a\ goal) \tag{A.1}$$

One example of the use of this metric is scouting. When looking for players, the scouts can use the expected goals metric to find players that are able to perform better than expected. One example can be seen in figure A.2.

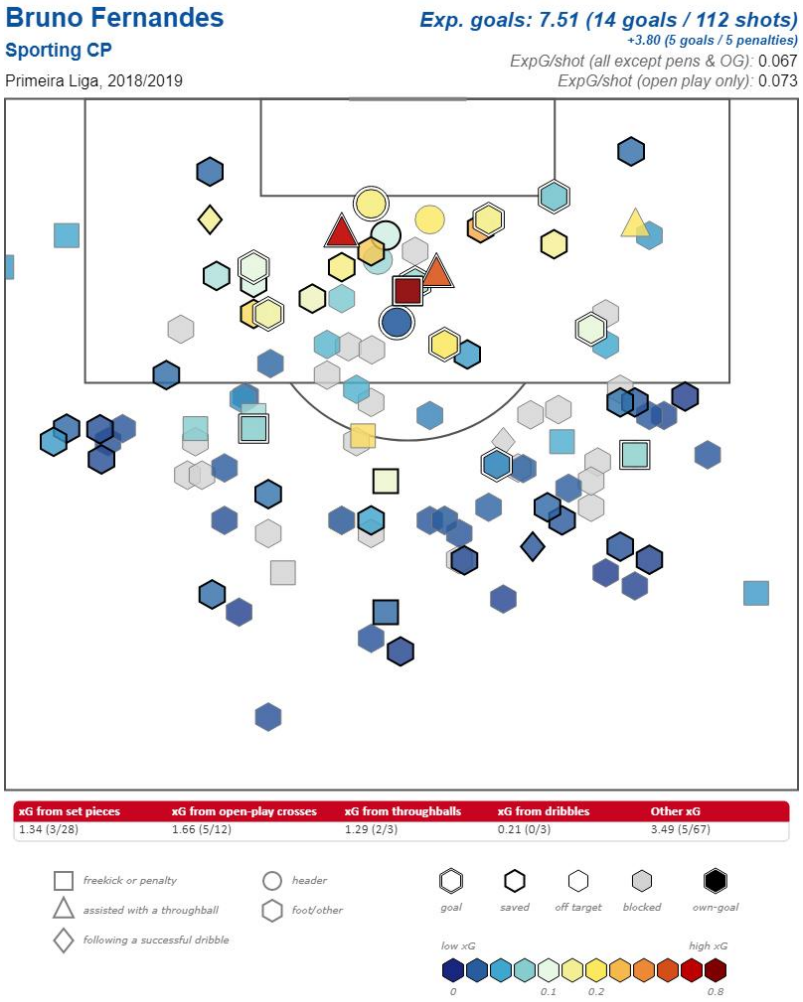


Figure A.2: Visualization of the expected goals from Bruno Fernandes shots in the 18-19 season. Source StatsBomb (2019)

Bruno Fernandes had 7.51 expected goals in the Portuguese Liga and was able to score 14 goals. This indicates that the player has performed way over the average when accounting for his finishing, which is a good indicator when looking for good players.

The same analysis can be made of the teams. If a good team is performing good in expected goals but having poor results in the matches it might indicate that they are only getting unlucky, being unable to capitalize their opportunities. Overall, expected goals are able to tell the story better than just goals.

Appendix B

Exploratory data analysis

B.1 The correlation matrix

The first step in order to have a first evaluation of the data is to check the correlation matrix. This tool allows for a broad visualization of the whole set of variables in data, identifying features that have high correlation with the target variable or even identifying features that hold the same information with the high inter-correlation. This matrix can be seen in figure B.1.

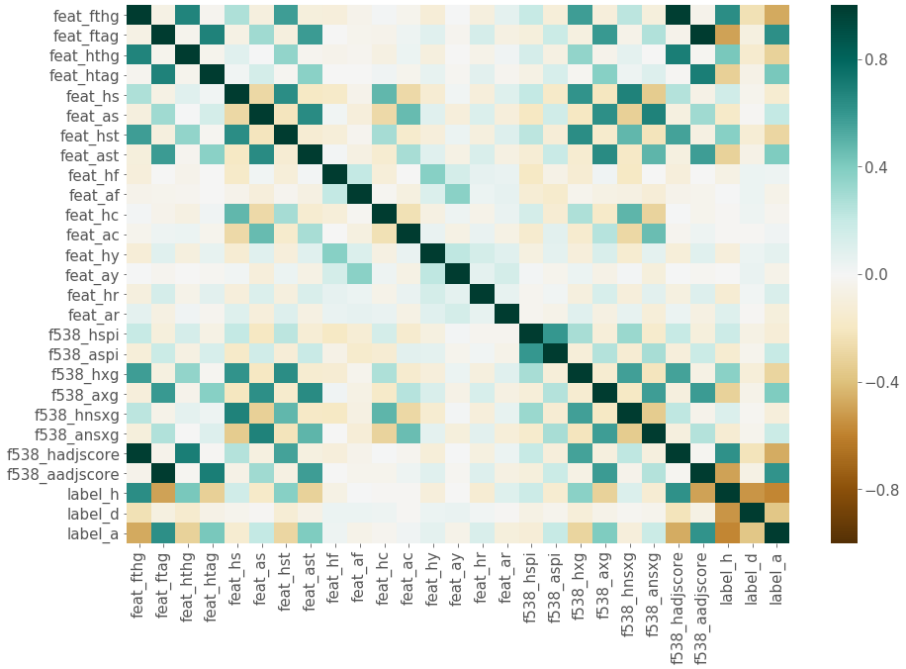


Figure B.1: Correlation matrix from all variables.

One of the problems when visualizing a correlation matrix is that the amount of information is very large and it makes the matrix hard to read. Since the focus is to look for high correlations between features and labels or other features, the matrix will be cleansed, removing irrelevant

information (low correlations). While having no correlation does not mean anything by itself, it is a good indicator that these variables have a low impact on the results. The cleansed matrix can be seen in figure B.2.

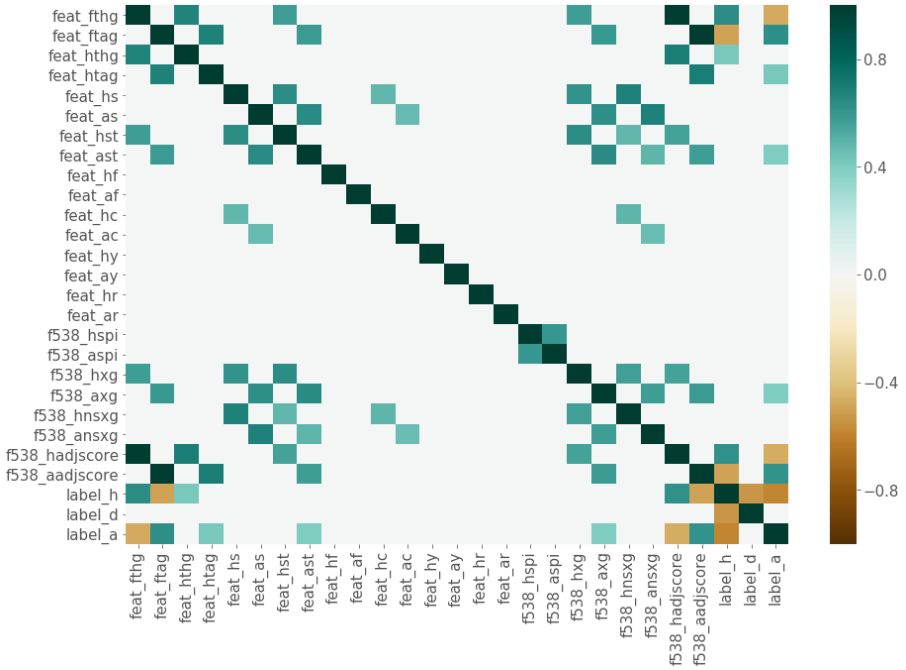


Figure B.2: Correlation matrix from all variables, filtered to only show correlations above 0.4.

Having cut correlation with absolute value under 0.4, we are left with the matrix with the most important correlations. As expected, shots on target (feat_hst and feat_ast) have a high correlation with goals, both full-time and half-time (feat_fthg and feat_ftag). It is interesting to note that this correlation does not happen with the total shots (feat_hs and feat_as). A reason for this is that shots on target are a good indicator that the attacking movement was well executed, the player wasn't being pressured or correctly defended when finishing, which might indicate that a team is able to create good scoring opportunities. As the total shots include the shots off target, that can be considered poorer opportunities, might result from shots from positions that held no treat to the defending team or simply because of the lack of quality of the attacking team, they tend to be worse predictors. Another factor for this bias towards shots on target might be that teams that are already winning are more picky when choosing their finishing movements, leading to better accuracy in the shots. To make predictions in our problem is more relevant to use the shots on target than the total shots metric.

Alongside with total shots, another strong correlation with shots on target is corners (feat_hc and feat_ac), which are a result of being able to put pressure on the defense. Again, total shots do not hold this correlation, showing that high shooting teams that lead to very little shots on target are not able to pressure the defending team.

A less relevant correlation is between fouls (feat_hf and feat_af) and yellow cards (feat_hy and

feat_ay), where more fouls lead to more yellow cards. More interestingly, yellow cards for either of the team seems to lead to yellow cards for the other team, due to an attempt of the referees to keep the game balanced when it comes to bookings. This is also slightly biased because stricter referees will have a overall bigger amount of yellows for both teams.

When verifying correlations with the target variables (label_h, label_d and label _a), the shots on target is one of the top correlations, alongside with the goals scored, both at full-time or half-time, shots on target and expected goals (f538_hxg and f538_axg).

The expected goals has interesting results. The correlation is high with both shots and shots on target metrics, with slightly weaker correlations with the goals. The correlation with shots is expected since the expected goals is calculated based on the addition of the quality of the shots made by a team. Since it is based on an addition, it means that the more shots a team has, the bigger the expected goals metric is.

B.2 Goals

The most important stat in football is the result, expressed in goals. It is then defined as a good starting point to start the analysis.

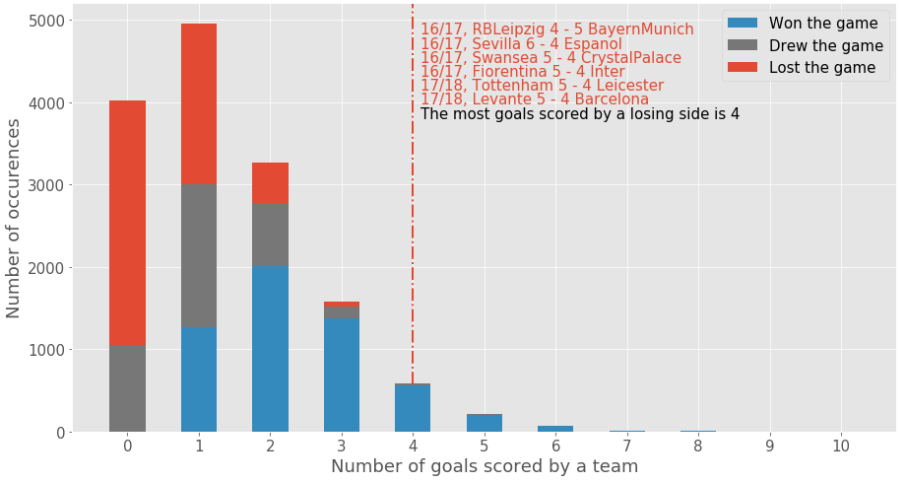


Figure B.3: Histogram of the number of goals scored by a team in a match. The histogram has 3 divisions, that depend on the final result of the match when a team has scored that amount of goals.

A quick visualization of figure B.3 allows us to verify that the most common amount of goals scored by a team is 1. While the second most common is 0, which guarantees that a team does not win the game, the guarantee that a team will win the game, based on the data, only comes when the team has scored 5 goals.

In figure B.4 we can verify that when accounting for both teams goals, the average total score is approximately 2.75 goals per game. We can also conclude that two of the most popular markets in football betting, over/under 2.5 goals and both teams to score, are approximately 50/50% bets.

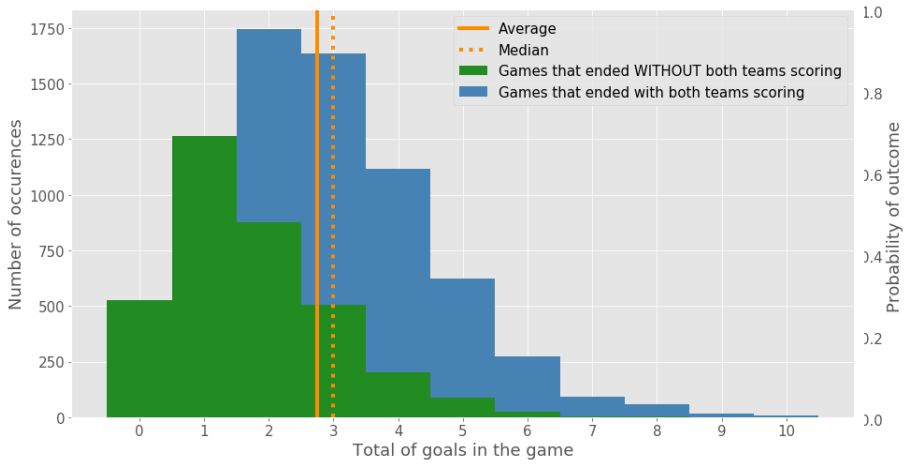


Figure B.4: Histogram of the number of goals scored by a team in a match. The histogram divides himself into two: whether both teams were able to score in the match or not.

Due to the different leagues present in the data set, it is interesting to observe if these results hold up when separating the data.

The most interesting results in figure B.5 is the discrepancy from average and medians, in particular in the Portuguese Liga. Even though there is a high average (5th place nearly tied with the Spanish Primera Division), the median is only 2.

What justifies this result is that the top teams consistently hold high-scoring victories over the smaller teams, eg. Benfica 10 - 0 Nacional and Belenenses 1 - 8 Sporting. This heavily increases the average number of goals, but the median is not as strongly affected as the average. These results can be considered outliers, but in the Portuguese Liga they are common. This occurs over the european football leagues and this difference between median and average helps us understand how competitive leagues are.

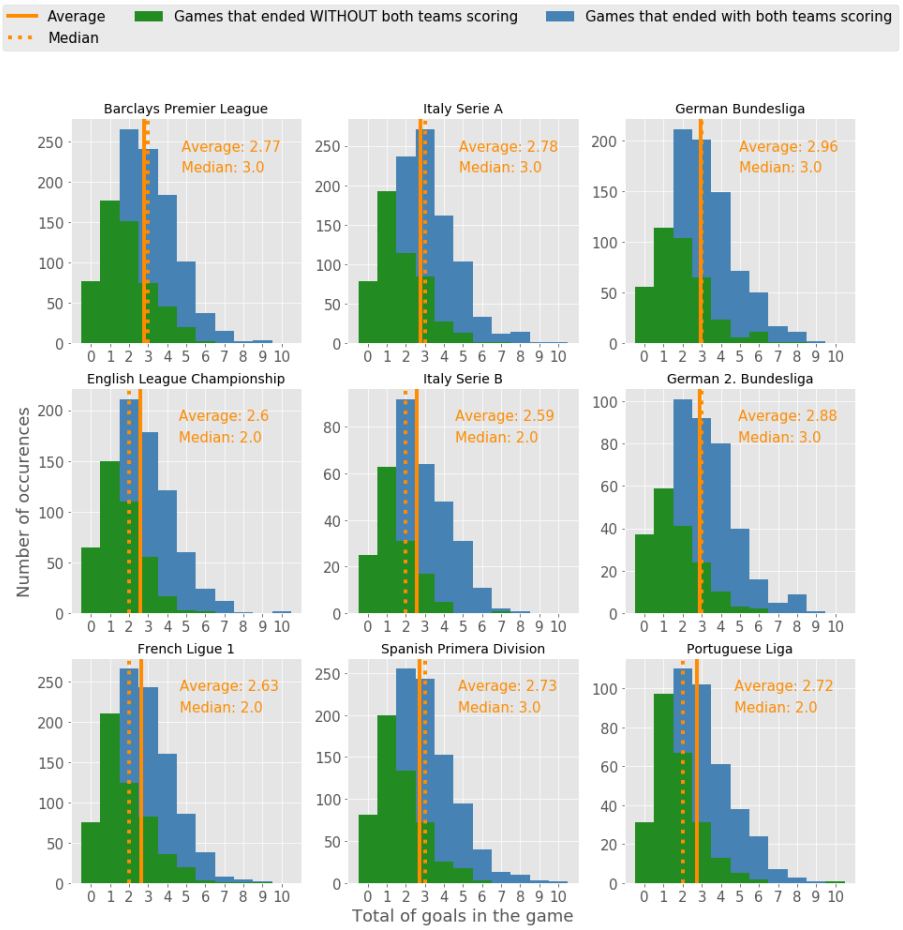


Figure B.5: Histogram of the number of goals scored by a team in a match, separated by league.

B.3 Shot-based variables

As observed in the correlation matrix, it looked like shots on target were substantially better at predicting the winner than total shots.

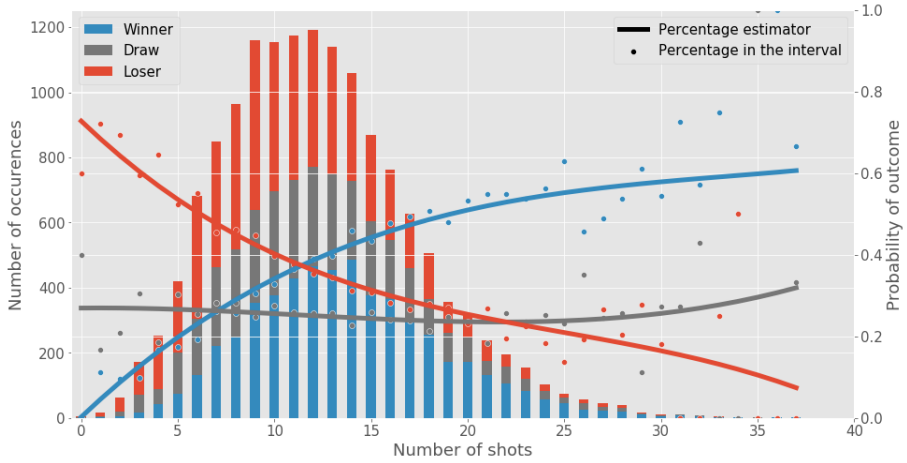


Figure B.6: Histogram of the number of shots by a team in a match.

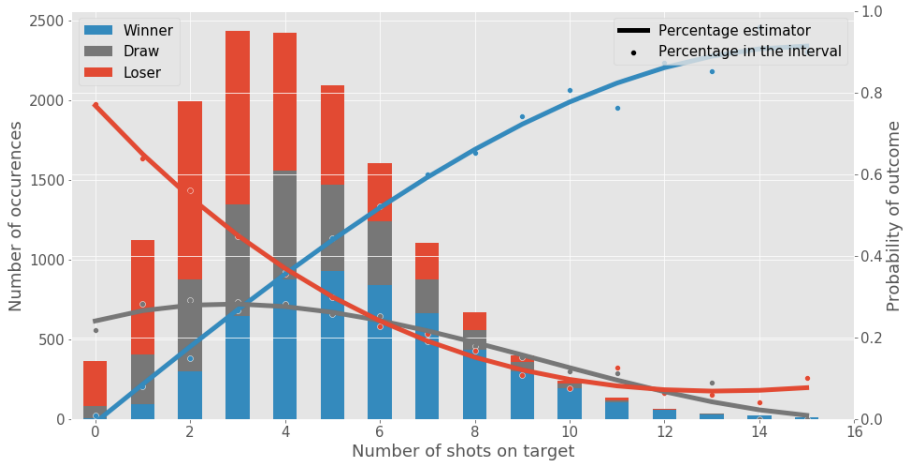


Figure B.7: Histogram of the number of shots on target by a team in a match.

In figures B.6 and B.7 is evident that more shots and shots on target leads to more wins. We can also confirm that the shots on target give bigger win probabilities than the total shots when the number increases. This makes shots on target a better predictor for when the classifiers are trying to split the data.

As the goal is not to predict one team performance but the result in football matches, we need variables that quantify the strength of a team to another. This strength can be calculated using differential features.

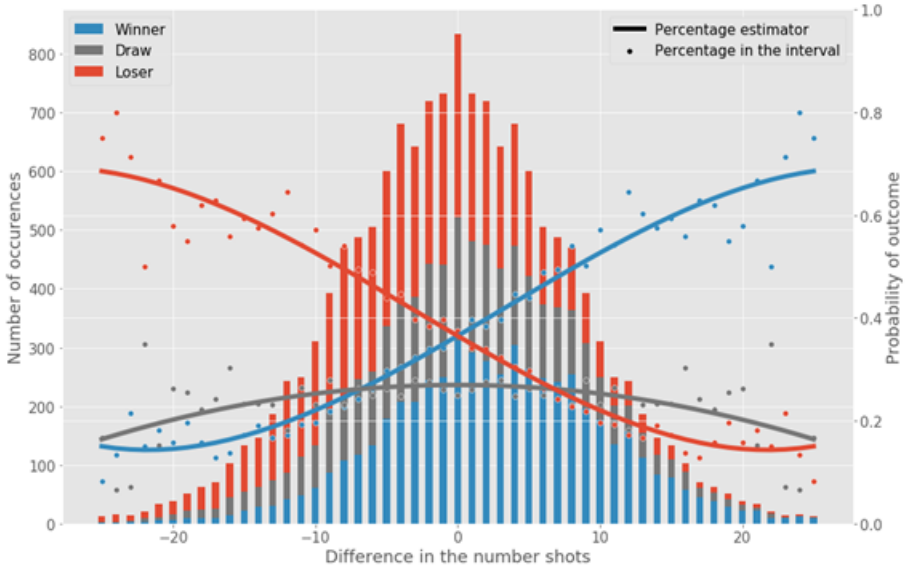


Figure B.8: Histogram of the difference of shots (home team - away team) in a match.

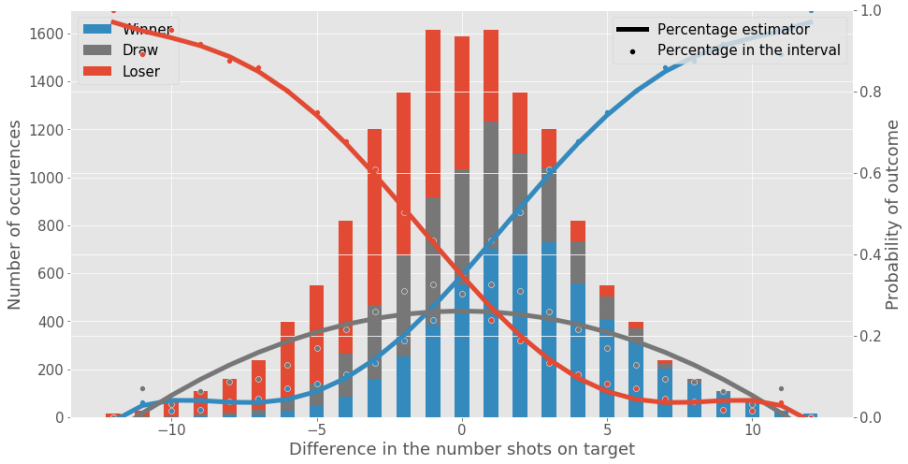


Figure B.9: Histogram of the difference of shots on target (home team - away team) in a match.

As it is verifiable in figures B.8 and B.9, the difference between both teams is also a good indicator for the winner of the match.

After verifying the total shots and shots on target variables, there are two key variables that are left to summarize, corners and expected goals.

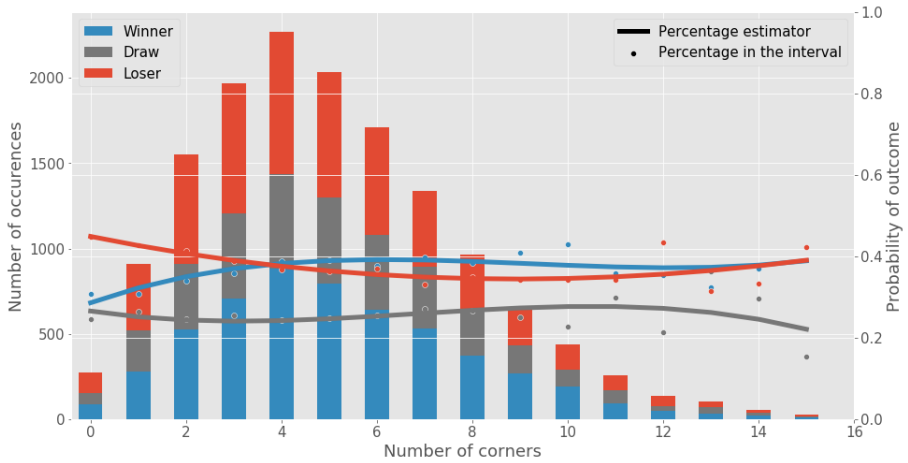


Figure B.10: Histogram of the number of corners by a team in a match.

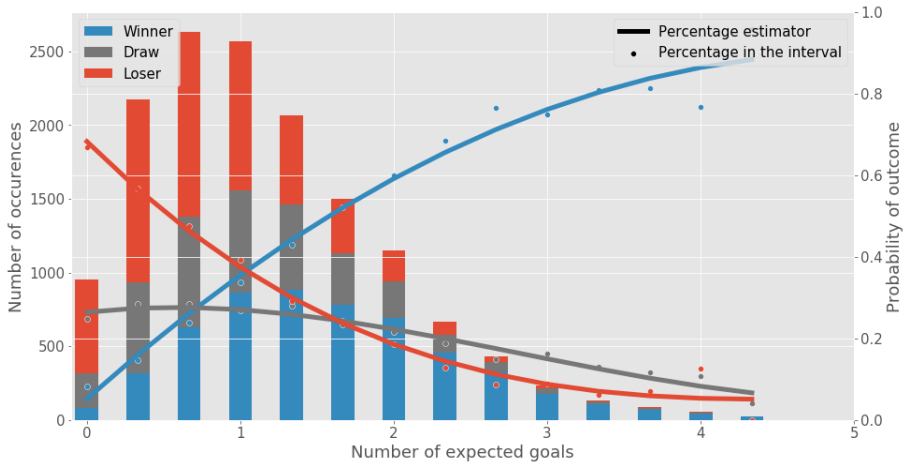


Figure B.11: Histogram of the number of expected goals by a team in a match.

As seen in figure B.10, the corners variable does not hold any value when used as the solo predictor for a football match. On the other hand, the expected goals, figure B.11, seem to be a good predictor for the winner of a football match.

Appendix C

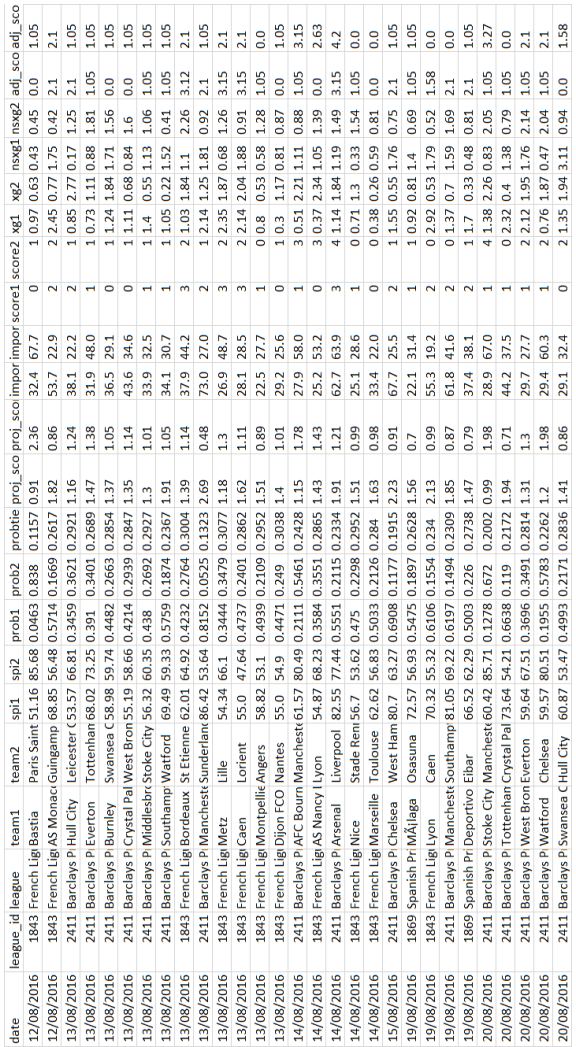
football-data.co.uk data

Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR	B365H	B365D	B365A	BbAv>2.5	BbAv<2.5
P1	06/08/2017	Aves	Sp Lisbon	0	2	A	0	1	A	12	12	2	9	15	16	6	3	1	0	0	0	9	4.2	1.4	1.95	1.83
P1	06/08/2017	Setubal	Moreirens	1	1	D	1	0	H	6	12	2	4	18	22	4	8	1	3	1	0	2.5	3	3	2.55	1.48
P1	07/08/2017	Feirense	Tondela	1	1	D	0	1	A	12	13	5	6	19	25	5	3	4	1	0	0	2.2	3.2	3.4	2.35	1.56
P1	07/08/2017	Portimone	Boavista	2	1	H	0	1	A	12	5	4	2	14	10	9	4	1	1	0	0	2.4	2.9	3.25	2.57	1.47
P1	07/08/2017	Rio Ave	Belenense	1	0	H	1	0	H	11	10	5	3	17	23	4	8	2	2	0	0	1.91	3.4	4	2.31	1.57
P1	08/08/2017	Maritimo	Pacos Ferr	1	0	H	0	0	D	7	4	3	1	13	14	0	7	2	2	0	0	2	3.2	4	2.3	1.57
P1	09/08/2017	Benfica	Sp Braga	3	1	H	2	1	H	15	6	9	1	13	18	6	2	2	3	0	0	1.36	4.75	8.5	1.73	2.05
P1	09/08/2017	Porto	Estoril	4	0	H	1	0	H	17	10	8	4	18	12	10	5	1	1	0	0	1.18	6.5	15	1.58	2.3
P1	10/08/2017	Guimaraes	Chaves	3	2	H	2	0	H	16	15	8	3	21	13	9	7	1	4	0	0	1.65	3.6	5.5	2.08	1.71
P1	11/08/2017	Sp Lisbon	Setubal	1	0	H	0	0	D	12	3	5	1	16	24	7	1	1	2	0	0	1.2	6.5	13	1.61	2.27
P1	12/08/2017	Belenense	Maritimo	1	0	H	0	0	D	10	7	4	5	20	18	9	4	3	1	0	0	2.4	3	3.2	2.41	1.53
P1	12/08/2017	Boavista	Rio Ave	1	2	A	0	1	A	5	7	2	2	17	17	2	3	3	3	0	0	3	3	2.5	2.57	1.47
P1	12/08/2017	Moreirens	Feirense	0	0	D	0	0	D	11	6	0	4	18	23	8	4	1	2	0	0	2.4	3	3.2	2.44	1.52
P1	13/08/2017	Pacos Ferr	Aves	2	2	D	0	2	A	16	7	5	3	17	20	2	3	1	3	0	0	1.91	3.3	4.2	2.33	1.56
P1	13/08/2017	Sp Braga	Portimone	2	1	H	1	0	D	8	8	4	1	14	21	5	7	3	3	0	0	1.53	4	6	1.88	1.88
P1	13/08/2017	Tondela	Porto	0	1	A	0	1	A	5	15	2	6	22	10	2	6	3	2	0	0	11	5	1.29	1.72	2.07
P1	14/08/2017	Chaves	Benfica	0	1	A	0	0	D	9	21	4	9	12	15	5	13	1	2	0	0	7	4.75	1.4	1.75	2.02
P1	14/08/2017	Estoril	Guimaraes	3	0	H	1	0	H	18	11	5	1	16	13	4	7	2	3	1	2	3.2	3.2	2.3	2.21	1.62
P1	18/08/2017	Rio Ave	Portimone	2	0	H	0	0	D	11	11	7	4	15	11	4	6	0	0	0	0	1.7	3.6	5	2.12	1.7
P1	19/08/2017	Benfica	Belenense	5	0	H	3	0	H	20	7	14	4	22	11	6	2	0	1	0	0	1.2	6.25	15	1.47	2.54
P1	19/08/2017	Guimaraes	Sp Lisbon	0	5	A	0	3	A	8	13	3	10	18	20	5	3	3	2	0	0	4.75	3.6	1.75	1.94	1.84
P1	19/08/2017	Tondela	Estoril	2	3	A	0	2	A	14	8	7	4	23	18	6	1	3	2	0	0	2.25	3.25	3.2	2.28	1.59
P1	20/08/2017	Aves	Sp Braga	0	2	A	0	0	D	9	18	4	9	14	10	4	8	1	0	1	0	3.1	3.25	2.3	2.23	1.61
P1	20/08/2017	Maritimo	Boavista	1	0	H	0	0	D	10	9	2	1	20	10	2	6	0	1	0	0	2.15	3.1	3.6	2.47	1.51
P1	20/08/2017	Porto	Moreirens	3	0	H	2	0	H	27	6	11	2	12	9	13	3	0	2	0	0	1.14	7.5	17	1.51	2.45
P1	20/08/2017	Setubal	Chaves	1	1	D	0	0	D	19	5	9	2	16	16	8	1	1	4	0	1	2.6	3	2.9	2.41	1.53
P1	21/08/2017	Feirense	Pacos Ferr	2	1	H	1	1	D	12	12	7	3	20	18	5	5	2	2	0	0	2.5	3.1	2.9	2.36	1.55
P1	25/08/2017	Belenense	Setubal	1	1	D	1	0	H	12	12	5	4	24	16	6	6	3	2	1	0	2.4	3	3.2	2.56	1.48
P1	26/08/2017	Chaves	Feirense	0	2	A	0	1	A	17	7	5	5	12	14	13	3	1	4	0	0	1.91	3.3	4.2	2.38	1.55

Figure C.1: A sample of the football-data.co.uk data. Available at http://www.football-data.co.uk/

Appendix D

fivethirtyeight.com data



date	league_id	league	team1	team2	spi1	spi2	prob1	prob2	probtie	proj_sco	proj_sco	impor	impor	score1	score2	xg1	xg2	nsxg1	nsxg2	adj_sco	adj_sco
12/08/2016	1843	French Ligi	Bastia	Paris Saint	51.16	85.68	0.0463	0.838	0.1157	0.91	2.36	32.4	67.7	0	1	0.97	0.63	0.43	0.45	0.0	1.05
12/08/2016	1843	French Ligi	AS Monaco	Guingamp	68.85	56.48	0.5714	0.1669	0.2617	1.82	0.86	53.7	22.9	2	2	2.45	0.77	1.75	0.42	2.1	2.1
13/08/2016	2411	Barclays P	Hull City	Leicester C	53.57	66.81	0.3459	0.3621	0.2921	1.16	1.24	38.1	22.2	2	1	0.85	2.77	0.17	1.25	2.1	1.05
13/08/2016	2411	Barclays P	Everton	Tottenham	68.02	73.25	0.391	0.3401	0.2689	1.47	1.38	31.9	48.0	1	1	0.73	1.11	0.88	1.81	1.05	1.05
13/08/2016	2411	Barclays P	Burnley	Swansea C	58.98	59.74	0.4482	0.2663	0.2854	1.37	1.05	36.5	29.1	0	1	1.24	1.84	1.71	1.56	0.0	1.05
13/08/2016	2411	Barclays P	Crystal Pal	West Bron	55.19	58.66	0.4214	0.2939	0.2847	1.35	1.14	43.6	34.6	0	1	1.11	0.68	0.84	1.6	0.0	1.05
13/08/2016	2411	Barclays P	Middlesbro	Stoke City	56.32	60.35	0.438	0.2692	0.2927	1.3	1.01	33.9	32.5	1	1	1.4	0.55	1.13	1.06	1.05	1.05
13/08/2016	2411	Barclays P	Southamp	Watford	69.49	59.33	0.5759	0.1874	0.2367	1.91	1.05	34.1	30.7	1	1	1.05	0.22	1.52	0.41	1.05	1.05
13/08/2016	1843	French Ligi	Bordeaux	St Etienne	62.01	64.92	0.4232	0.2764	0.3004	1.39	1.14	37.9	44.2	3	2	1.03	1.84	1.1	2.26	3.12	2.1
13/08/2016	2411	Barclays P	Manchest	Sunderlan	86.42	53.64	0.8152	0.0525	0.1323	2.69	0.48	73.0	27.0	2	1	2.14	1.25	1.81	0.92	2.1	1.05
13/08/2016	1843	French Ligi	Metz	Lille	54.34	66.1	0.3444	0.3479	0.3077	1.18	1.3	26.9	48.7	3	2	2.35	1.87	0.68	1.26	3.15	2.1
13/08/2016	1843	French Ligi	Caen	Lorient	55.0	47.64	0.4737	0.2401	0.2862	1.62	1.11	28.1	28.5	3	2	2.14	2.04	1.88	0.91	3.15	2.1
13/08/2016	1843	French Ligi	Montpellie	Angers	58.82	53.1	0.4939	0.2109	0.2952	1.51	0.89	22.5	27.7	1	0	0.8	0.53	0.58	1.28	1.05	0.0
13/08/2016	1843	French Ligi	Dijon FCO	Nantes	55.0	54.9	0.4471	0.249	0.3038	1.4	1.01	29.2	25.6	0	1	0.3	1.17	0.81	0.87	0.0	1.05
14/08/2016	2411	Barclays P	AFC Bourn	Manchest	61.57	80.49	0.2111	0.5461	0.2428	1.15	1.78	27.9	58.0	1	3	0.51	2.21	1.11	0.88	1.05	3.15
14/08/2016	1843	French Ligi	AS Nancy L	Lyon	54.87	68.23	0.3584	0.3551	0.2865	1.43	1.43	25.2	53.2	0	3	0.37	2.34	1.05	1.39	0.0	2.63
14/08/2016	2411	Barclays P	Arsenal	Liverpool	82.55	77.44	0.5551	0.2115	0.2334	1.91	1.21	62.7	63.9	3	4	1.14	1.84	1.19	1.49	3.15	4.2
14/08/2016	1843	French Ligi	Nice	Stade Ren	56.7	53.62	0.475	0.2298	0.2952	1.51	0.99	25.1	28.6	1	0	0.71	1.3	0.33	1.54	1.05	0.0
14/08/2016	1843	French Ligi	Marseille	Toulouse	62.62	56.83	0.5033	0.2126	0.284	1.63	0.98	33.4	22.0	0	0	0.38	0.26	0.59	0.81	0.0	0.0
15/08/2016	2411	Barclays P	Chelsea	West Ham	80.7	63.27	0.6908	0.1177	0.1915	2.23	0.91	67.7	25.5	2	1	1.55	0.55	1.76	0.75	2.1	1.05
19/08/2016	1869	Spanish Pr	MÁlaga	Osasuna	72.57	56.93	0.5475	0.1897	0.2628	1.56	0.7	22.1	31.4	1	1	0.92	0.81	1.4	0.69	1.05	1.05
19/08/2016	1843	French Ligi	Lyon	Caen	70.32	55.32	0.6106	0.1554	0.234	2.13	0.99	55.3	19.2	2	0	2.92	0.53	1.79	0.52	1.58	0.0
19/08/2016	2411	Barclays P	Manchest	Southamp	81.05	69.22	0.6197	0.1494	0.2309	1.85	0.87	61.8	41.6	2	0	1.37	0.7	1.59	1.69	2.1	0.0
19/08/2016	1869	Spanish Pr	Deportivo	Eibar	66.52	62.29	0.5003	0.226	0.2738	1.47	0.79	37.4	38.1	2	1	1.7	0.33	0.48	0.81	2.1	1.05
20/08/2016	2411	Barclays P	Stoke City	Manchest	60.42	85.71	0.1278	0.672	0.2002	0.99	1.98	28.9	67.0	1	4	1.38	2.26	0.83	2.05	1.05	3.27
20/08/2016	2411	Barclays P	Tottenhan	Crystal Pal	73.64	54.21	0.6638	0.119	0.2172	1.94	0.71	44.2	37.5	1	0	2.32	0.4	1.38	0.79	1.05	0.0
20/08/2016	2411	Barclays P	West Bron	Everton	59.64	67.51	0.3696	0.3491	0.2814	1.31	1.3	29.7	27.7	1	2	2.12	1.95	1.76	2.14	1.05	2.1
20/08/2016	2411	Barclays P	Watford	Chelsea	59.57	80.51	0.1955	0.5783	0.2262	1.2	1.98	29.4	60.3	1	2	0.76	1.87	0.47	2.04	1.05	2.1
20/08/2016	2411	Barclays P	Swansea C	Hull City	60.87	53.47	0.4993	0.2171	0.2836	1.41	0.86	29.1	32.4	0	2	1.35	1.94	3.11	0.94	0.0	1.58

Figure D.1: A sample of the fivethirtyeight.com data.
Available at https://fivethirtyeight.com/

Appendix E

Comparing state-of-the-art neural network ensemble methods in football predictions

Comparing state-of-the-art neural network ensemble methods in football predictions*

Tiago Mendes-Neves[1][0000−0002−4802−7558] and
João Mendes-Moreira[1,2][0000−0002−9081−2728]

[1] Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
[2] LIAAD-INESC TEC, Porto, Portugal

Abstract. For many reasons, including sports being one of the main forms of entertainment in the world, online gambling is growing. And in growing markets, opportunities to explore it arise. Machine learning can have an important role here. In this paper, neural network ensemble approaches, such as bagging, random subspace sampling, negative correlation learning and simply averaging predictions of several networks are compared. For each one of these methods several combinations of input parameters are evaluated. As the scope of the project is focused on modeling, other steps are simplified. We only used the expected goals metric as predictors, since it is able to have good predictive power, while keeping the computational demands lows. These models are compared in the football betting context, where we have access to metrics such as rentability to analyze the results in multiple perspectives. The results show that the optimal solution is goal-dependent, with the ensemble methods being able to increase the accuracy up to +3% over the single model.

Keywords: Sports betting · Neural networks · Ensemble learning.

1 Introduction

1.1 Sports betting

Gambling was always an interesting concept to human beings. If we ask a person if they want to trade 1€ for 0.95€ they will immediately reject the proposal. Being guaranteed to lose money is something that is not usually accepted without being rewarded. In betting, the reward comes from the existing probability of winning money. Even though in the long term more money is lost than won, the human brain is blinded by the prospect of a big win.

There is a niche part of gambling that is of interest to us: football betting. Unlike other forms of gambling, in football betting, the probabilities are not predefined or easily calculated.

* This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project: UID/EEA/50014/2019.

Bookmakers have the luxury of having access to the wisdom of the crowd that when combined with the ability for the market to self regulate, leaves them making a consistent profit regardless of the outcome.

Machine learning algorithms have the ability to calculate probabilities, and with every move on the football pitch being recorded, the amount of available data is enabling these models to become more accurate than ever.

The academia has embraced the problem of predicting football matches very heavily. The majority of the work done involves Bayesian networks [2] [7]. Some other algorithms used were fuzzy based model [11] and neural networks [9] [3].

1.2 Ensembling neural networks

The machine learning algorithm that is of interest to is the neural networks. Neural networks are connection-based models that have a very strong ability to assemble complex models. While neural networks having the ability to generate complex models may look like a plus, it can lead to overfitting, and with that, a bad generalization power, leading to sub-par performance when testing the models in previously unseen instances.

Ensemble models can help to solve this problem. Like the gathering of opinions in the betting markets improves the estimate over a single opinion, ensemble models gather the predictions of several models and by combining them allows the ensemble prediction to be better than any of the individual predictions alone.

Neural network ensembles have been used to solve several problems. Due to its ability to work with image data, the majority comes from the health sector [6]. However, other sectors, such as the financial [12], have research in the area.

What is missing from the academic perspective is a review of the available ensemble methods for neural networks. With neural networks having an extensive problem of overfitting and ensemble models having the ability to solve this problem, the need for a comparison of the options in different areas of research rises.

2 Describing the data

The experiments described in paper uses data from two sources: fivethirtyeight.com soccer-spi data set [4] and football-data.co.uk [5]. From the first, we retrieve the expected goals metric for every match from the season 2016/2017 to 2018/2019. On the second, we acquire the odds from the matches that we retrieved from the soccer-spi data set. Having done that, we were left with the data set described in table 1.

The data set has games from 6 leagues: English Premier League, French Ligue 1, Spanish La Liga, Italian Serie A, German Bundesliga and Portuguese Primeira Liga. On this last, the 2016/2017 season data is not available.

The training set will be composed of the 2016/2017 and 2017/2018 season data, in a total of 3178 games. For the test set, the full 2018/2019 season is used, amounting to 1656 games.

Table 1. Variables present in the assembled data set.

Variable	Data type	Description
game_id	Symbolic	Internal unique id for integrating the databases
season	Symbolic	Season in which the game occurred
season_day	Numeric	Day of the season (season start set to July 1st)
home_team	Symbolic	Home team identifier
away_team	Symbolic	Away team identifier
h_expected_goals	Numeric	Performance of the home team measured in expected goals
a_expected_goals	Numeric	Performance of the away team measured in expected goals
h_odd	Numeric	Average odd from the bookmakers for the home team to win
d_odd	Numeric	Average odd from the bookmakers for the draw
a_odd	Numeric	Average odd from the bookmakers for the away team to win

2.1 Expected goals

The expected goals metric [10] is a measure of a shot quality. It is calculated from the likelihood of a shot ending in a goal, taking into account factors such as distance to the goal, angle of the shot, body part used to make the shot and whether it was a first touch shot or not. The mathematical formulation can be seen in equation 1.

$$expected\ goals\ for\ a\ team = \sum_{for\ all\ team\ shots} P(shot\ leading\ to\ a\ goal) \quad (1)$$

3 Experimental setup

3.1 Performance metrics

The goal in the experiments is to obtain results that allow us to compare the performance of different ensemble algorithms. We are going to define a set of metrics that will be used, each one focusing on a part of the problem.

The first metric that will be used is the accuracy (equation 2). This is a metric that is of standard use in classification problems.

$$accuracy = \frac{correct\ predictions}{number\ of\ predictions} \quad (2)$$

The probabilities generated by the classification algorithms can be evaluated from two points of view: the betting and the regression point of view.

On the betting point of view, the defined metric was the rentability (equation 3), that tells us how much money the model made in relation to the stake. For this calculation, the algorithms will be always betting in the predicted favourite and the stake of each bet will be one unit.

$$rentability = \sum_{correct\ predictions} (odd - 1) \quad - \quad number\ of\ incorrect\ predictions \tag{3}$$

From the regression point of view, two metrics were used: bias and variance. Both bias and estimated variance are defined in equations 4 and 5, where M is the number of predictions made, γ is the predicted value and y is the real value. The bias is the error caused by the model's simplified assumptions that cause a constant error across different choices of training data. Variance is the variability of the model's predictions for a given instance. A high variance means that for the same instance, the same model trained in slightly different data will yield different results.

$$bias^2 = \frac{1}{M} \sum_{i}^{M} y_i - \gamma_i \tag{4}$$

$$estimated\ variance = \frac{1}{M} \sum_{i}^{M} stdev(\gamma_i) \tag{5}$$

The last metric measured is the average training time of the models. No optimization was made in any of the algorithms.

3.2 The feature set

The feature set will be generated on top of the expected goals metric. For that, we will assemble the expected goals scored and conceded in each of the last 7 games for both teams facing up. This will leave us with 14 features for each team and a total of 28 features for our model.

3.3 Base learners

There is a wide spectrum of choices when tuning a neural network, and these choices have a high impact the final results of the neural network, therefore it is necessary to establish what will be the base learners used in the ensemble models in order to keep them comparable.

Two architectures were tested. The first uses a single hidden layer with 15 nodes (15,), the second uses two hidden layers of 10 and 5 (10,5). In both cases, the layers use the *softmax* activation function, in order to have predictions in the form of probabilities.

A parameter that will be tested is the early stop. Since it naturally increases the variability of the models it might lead to better performance than the no early stop variant when ensembled.

Other parameters are constant thought the models. Learning rate is 0.025 and batch size is 200. The loss function used is categorical cross-entropy.

The table 2 summarizes the base learners used in the experiences.

Table 2. Presentation of the base learners.

Model	Architecture	Epochs	Early Stop	Patience
1	(15,)	100	No	
2	(15,)	500	Yes	10
3	(10,5)	100	No	
4	(10,5)	500	Yes	10

3.4 Proposed algorithms

Bagging Bagging [1] is perhaps the most common ensemble approach used. The idea is to generate new training data sets from a single instance of base data by sampling, which can be performed with or without replacement.

In our implementation, similarly to what is done in random forest with the random subspace sampling, there is a parameter that allows us to modify the feature subset in which the models are trained. This parameter will be called *feature_ratio*, and it indicates a percentage of the features that will be used by each model. The number of samples from the base data that will be used is indicated by the parameter *sample_ratio*, which is a percentage of the samples that will be used. Samples are drawn without replacement.

Simple average dropout networks (SADN) The SADN is an ensemble in which the goal of each model is to have low variance. The dropout parameter acts as regularization for the networks, not allowing them to become too complex and overfit. The predicted probabilities from the ensembles' models are then averaged and re-normalized in order to produce the ensemble predictions. On the experiments, the hidden layers will have a 0.3 dropout rate.

Negative correlation learning (NCL) In negative correlation learning [8], the approach is to train individual networks in an ensemble and combining them in the same process. All the neural networks in the ensemble are trained simultaneously and interactively through a correlation penalty term in their error function.

The difference from regular neural network training is in the loss function. Subtracted to the regular loss function (a function of the predicted value and the real value) is a percentage (λ parameter) of the loss function calculated between the value predicted by the model and the ensemble predicted value. This can be seen in equation 6, where γ is a neural network prediction, ε the ensemble prediction and y is the real value. This incentives models to go in a different direction from the average value of the ensemble when training, creating diversity in the model's opinions and improving the model classification performance.

$$new\ loss\ function\ =\ loss\ function(\gamma, y)\ -\ \lambda\ *\ loss\ function(\gamma, \varepsilon) \quad (6)$$

4 Experiments

4.1 Ensemble hyperparameter tuning

Before the comparison could be made, the ensemble model parameters need to be tuned. The first parameter defined is that each ensemble will have 50 base learners. While SADN does not need any additional parameter, both bagging and NCL have parameters that need to be tuned. To tune the parameters for the bagging method we do a space state search in order to find the optimal hyperparameters. This can be seen in figure 1.

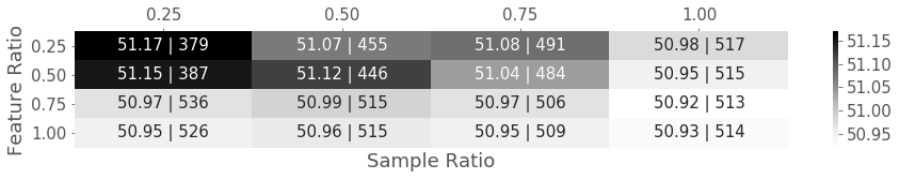


Fig. 1. Hyperparameters state space search for bagging in model 1, with results in the format $Accuracy|Variance/10^-4$. The results are averages of 10 runs.

From figure 1 we can conclude that both feature_ratio and sample_ratio improve the performance when lowered. This can be verified from both accuracy and variance perspective. The best performing hyperparameters (both parameters equal to 0.25) will be used.

For the NCL we need to set the parameter λ. In the tuning phase, low λ values seemed to perform better, as seen in figure 2. The chosen λ is 0.1.

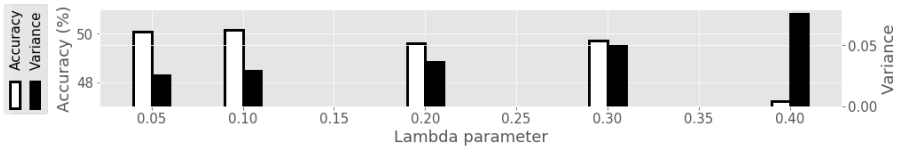


Fig. 2. Hyperparameters state space search for NCL. The tests were done on the model 1. The results are averages of 10 runs.

4.2 Ensemble methods comparison

Table 3 shows the results of the experiments. These results are the average of 50 runs of each algorithm.

Accuracy wise, the best performing model was the SADN (15,) with early stop. This is also the only instance where early stopping lead to better accuracy

Table 3. Results from the tests with optimal parameters. Note that the expected rentability (calculated by betting in every outcome of every game) is -77.22.

Evaluation Metric	Single model		Bagging		SADN		NCL
Early Stop	Yes	No	Yes	No	Yes	No	No
Architecture	(15,)						
Accuracy	48.95	50.18	50.83	51.22	51.70	51.24	50.01
Rentability	-96.23	-77.92	-73.46	-65.85	-43.24	-60.83	-71.39
Bias	0.6284	0.6287	0.6288	0.6287	0.6297	0.6293	0.6293
Variance	0.2199	0.1049	0.0766	0.0397	0.0104	0.0112	0.0229
Average execution time	4.18	1.65	89.65	24.72	82.19	140.47	102.9
Architecture	(10,5)						
Accuracy	50.21	50.61	51.08	51.35	50.91	51.00	51.31
Rentability	-62.75	-70.44	-52.16	-61.29	-49.66	-52.40	-53.27
Bias	0.6291	0.6286	0.6334	0.6294	0.6326	0.6311	0.6302
Variance	0.1710	0.0881	0.0528	0.0304	0.0133	0.0095	0.0136
Average execution time	2.56	1.71	43.00	25.51	107.39	170.22	126.46

results, since neither bagging or single models were able to improve when using early stop. In the (10,5) architecture the SADN with early stopping did not replicate this success.

The accuracy performance of the SADN (15,) enabled the rentability to also be the best.

Since these models were focused on variance reduction, it was expected that the bias did not change considerably. While improvements in bias are scarce, the cost of using ensembles was low, with none of the biases increasing by over 1% over the single model.

On the other side, the variance was immensely reduced, with some models achieving approximately 95% reduction. The most notable performance here was also obtained by the SADN algorithm, with the NCL being a close contender.

The more complex architecture (10,5) found it harder to improve results. While the performance jumped almost 3% in the best scenario for the (15,) architecture, the (10,5) failed to improve even 1%. However, with exception of the SADN, all the algorithms managed to perform better on the (10,5) architecture.

Since the (15,) is a less complex model, ensembling with a methods that do not induce more variability in predictions (SADN) yielded better results. On the other side, a more complex model, (10,5), needed algorithms that introduced variability in the models (bagging/NCL) to improve the predictions. This leads to the conclusion that the most important factor when ensembling is to find the right balance of variability in the ensemble's models. Either too much or too little variance will worsen the results.

In terms of quickness, bagging is the better option. Without early stopping, bagging is able to only take approximately 15x more time to train than the simple network, while training 50x more models, due to the reduced number of instances and features used. It is interesting to note that while the execution

time of the SADN is quite high, it is the only algorithm where training with early stopping lead to faster training times.

5 Conclusions

Ensembling neural networks improves the results of single models in terms of accuracy up to 3%. This accuracy improvement can have a massive impact from the business perspective, as it can be seen in the rentability, with the best performing ensemble cutting the losses in half over the single model variant.

In general, ensemble proves itself to be a reliable way to reduce variance in the neural network context.

5.1 Future work

While the results look promising, especially for SADN, the tests are only done in one data set. The algorithms need to be test across multiple data sets to verify if these conclusions are consistent or if they only hold true in the football prediction scenario.

Even in the same data set, tests with different parameters can be done. The chosen network architectures are only a small subset of the immense possibilities when tuning a neural network.

References

1. Aggarwal, Charu C.: Neural Networks and Deep Learning. eBook. Springer, Switzerland (2018)
2. Constantinou, Anthony C., Fenton, Norman E., Neil M.: Profiting from an inefficient association football gambling market: Prediction, risk and uncertainty using Bayesian networks. Knowledge-Based Systems **50**, 60–86 (2013)
3. Dzalbs, I., Kalganova, T.: Forecasting Price Movements in Betting Exchanges Using Cartesian Genetic Programming and ANN. Big Data Research **14**, 112–120 (2018)
4. FiveThirtyEight, fivethirtyeight.com. Last accessed 21st June 2019
5. Football-Data.co.uk, football-data.co.uk. Last accessed 21st June 2019
6. Harangi, B.: Skin lesion classification with ensembles of deep convolutional neural networks. Journal of Biomedical Informatics **86**, 25–32 (2018)
7. Joseph, A., Fenton, N.E., Neil, M.: Predicting football results using Bayesian nets and other machine learning techniques. Knowledge-Based Systems **19**(7), 544–553 (2006)
8. Liu, Y., Yao X.: Ensemble learning via negative correlation. Neural Networks **12**(10), 1399–1404 (1999)
9. Mccabe, A., Trevathan, J: Artificial Intelligence in Sports Prediction. In: Fifth International Conference on Information Technology: New Generations, pp. 1194–1197. IEEE Computer Society, Las Vegas, Nevada, USA (2008)
10. Opta, optasports.com. Last accessed 22nd June 2019
11. Rotshtein, A. P., Posner, M., Rakityanskaya, A. B.: Football predictions based on a fuzzy model with genetic and neural tuning. Cybernetics and Systems Analysis **41**(4), 619–630 (2005)
12. Szafranek, K.: Bagged neural networks for forecasting Polish (low) inflation. International Journal of Forecasting **35**(3), 1042–1059 (2018)

References

Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

R. Baboota and H. Kaur. Predictive analysis and modelling football results using machine learning approach for english premier league. *International Journal of Forecasting*, 35(2):741 – 755, 2019.

L. Breiman. Arcing the edge. In *Technical Report 486*, 1997.

Encyclopedia Britannica. https://www.britannica.com/technology/data-mining, 2019. Accessed on 05-06-2019.

A. Bronshtein. A quick introduction to k-nearest neighbors algorithm. https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7, 2017. Accessed on 08-06-2019.

R. Buhagiar, D. Cortis, and P. W. S. Newall. Why do some soccer bettors lose more money than others? *Journal of Behavioral and Experimental Finance*, 18:85 – 93, 2018.

T. Chen and G. Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016.

A. C. Constantinou, N. E. Fenton, and M. Neil. Profiting from an inefficient association football gambling market: Prediction, risk and uncertainty using bayesian networks. *Knowledge-Based Systems*, 50:60 – 86, 2013.

Anthony C. Constantinou, Norman E. Fenton, and Martin Neil. pi-football: A bayesian network model for forecasting association football match outcomes. *Knowledge-Based Systems*, 36:322 – 339, 2012.

I. Dzalbs and T. Kalganova. Forecasting price movements in betting exchanges using cartesian genetic programming and ann. *Big Data Research*, 14:112 – 120, 2018.

European Gaming & Betting Association EGBA. Eu market. https://www.egba.eu/eu-market/, 2018. Accessed on 05-06-2019.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., 1978.

E. S. Epstein. A scoring system for probability forecasts of ranked categories. *Journal of Applied Meteorology*, 8:985–987, 11 1969.

U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 03 1996.

FiveThirtyEight. https://fivethirtyeight.com/. Accessed on 22-06-2019.

Football-Data.co.uk. http://www.football-data.co.uk/. Accessed on 22-06-2019.

Frank. The bias-variance tradeoff. http://www.machinelearningtutorial.net/2017/01/26/the-bias-variance-tradeoff/, 2017. Accessed on 08-06-2019.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

S. Gupta. Decision tree tutorial. https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/, 2019. Accessed on 08-06-2019.

T. K. Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

L. M. Hvattum and H. Arntzen. Using elo ratings for match result prediction in association football. *International Journal of Forecasting*, 26(3):460 – 470, 2010.

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

Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, 2013.

Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399 – 1404, 1999.

A. Mccabe and J. Trevathan. Artificial intelligence in sports prediction. In *Fifth International Conference on Information Technology: New Generations*, pages 1194–1197, 2008.

P. McKeever. Building an expected goals model in python. http://petermckeever.com/2019/01/building-an-expected-goals-model-in-python/, 2019. Accessed on 16-06-2019.

Opta. https://www.optasports.com/services/analytics/advanced-metrics/. Accessed on 22-06-2019.

F. Owramipur, P. Eskandarian, and F. S. Mozneb. Football result prediction with bayesian network in spanish league-barcelona team. *International Journal of Computer Theory and Engineering*, pages 812–815, 2013.

A. Rotshtein, M. Posner, and A. B. Rakityanskaya. Football predictions based on a fuzzy model with genetic and neural tuning. *Cybernetics and Systems Analysis*, 41:619–630, 2005.

A. L. Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of Research and Development*, 11(6):601–617, 1967.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2):206–226, 2000.

C. Shearer. The crisp-dm model: the new blueprint for data mining. *J Data Warehouse*, 5:13–22, 01 2000.

B. Shetty. Curse of dimensionality. https://towardsdatascience.com/curse-of-dimensionality-2092410f3d27, 2019. Accessed on 05-06-2019.

J. Shubham. Ensemble learning - bagging and boosting. https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e, 2018. Accessed on 05-06-2019.

StatsBomb. https://twitter.com/StatsBomb/status/1136956960631398400/photo/2, 2019. Accessed on 16-06-2019.

A. P. Weigel, M. A. Liniger, and C. Appenzeller. The discrete brier and ranked probability skill scores. *Monthly Weather Review*, 2006.