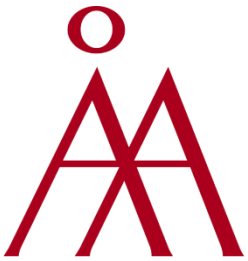


Football Match Prediction Using Machine Learning

Author: Fredrik Sjöberg - 42011

Supervisor: Sepinoud Azimi Rashti

Master's thesis in Computer Engineering

Åbo Akademi University, Finland

Faculty of Science and Engineering (FNT)

Information Technology

2023

Abstract

Football is the most popular sport in the world. It has become so popular since it is easy to play and affordable. Football can be played both inside and outside almost at any place in the world and little equipment is needed to play it. The rules are simple and easy to understand. Since football is so popular, matches and tournaments exhibit a high level of availability for both men and women at any age worldwide. Therefore, it is not surprising that professional football constitutes a colossal market. The most interesting aspect of football is of course the results of the matches and how they can be affected to go in favour of one's team. This is where predicting the results of the football matches is useful.

Machine learning is an excellent method for making predictions. In this thesis multiple machine learning algorithms and data sets are compared and analysed to find the best approach. The higher prediction accuracy a model achieves, the more accurate its predictions are. When predicting football match results, there are three possible outcomes: win for the home team, draw, or win for the away team. To tackle such classification problems using machine learning, multiclass classification can be used.

One of the most important aspects of building a successful machine learning model is to find a suitable data set and data features. Feature selection is difficult and can be made in many ways, but it is decisive for the results of the predictions. Data sets containing match statistics, team related features, and player related features, were used in this thesis. They were tested with different machine learning algorithms, to find the best possible combination.

Keywords: machine learning, multiclass classification, predictions, football

Table of Contents

Preface

I would like to thank my supervisor Sepinoud for her guidance and support throughout the project. I would also like to thank my employer for letting me prioritise my studies over work.

1. Introduction

Predicting the results of football matches is challenging. It is impossible to be done perfectly even with the help of machine learning. If it would be possible, the excitement of the sport would be lost and sports betting organisations would go bankrupt. One of the primary factors that football has such a large popularity around the world is that it is an unpredictable sport. In every match, it is possible for a weaker team to win over a stronger team and no matter how superior the stronger team is, it cannot be certain that it will win. Seemingly unpredictable events like the weather and a lucky last-minute goal can be the decisive event that will help an underdog defeat the giant. However, we know that it is possible to predict football matches with an accuracy over 90% [1]. In this thesis we investigate which approaches and strategies are the best to be able to achieve the most accurate football match predictions using machine learning.

In the recent years, the amount of data available online about football and other sports have increased massively. This has made it possible for researchers and hobbyists to develop and improve football prediction methods themselves. The goal of some people in this area is to find the best prediction methods for making as high profit as possible in sports betting. However, in such cases the aim is also to find the weak spots of the predictions of the sport betting organisations, rather than only finding the best possible match predictions. The aim of this thesis is not to find the best methods for making profit on sports betting sites, but purely to find out how predictable the game of football really is and how the best possible predictions can be made.

In this thesis multiple machine learning algorithms are compared, with the aim to find the best possible model for football match predictions. Bayesian networks and logistic regression are the most common algorithms to be used for football predictions according to previous research papers on this topic. Except these algorithms, the random forest algorithm was also chosen to be used in this implementation. The wanted output of the created models are the overall result of football matches and not the exact score. The three possible outcomes are win for

the home team, draw, or win for the away team. Multiclass classification is the most suitable type of machine learning algorithms to solve such problems.

The second chapter of the thesis gives an overview of machine learning, machine learning algorithms, and machine learning related problems and solutions. The focus is on multiclass classification. The third chapter consists of an overview of previous research in the topic. It also describes how football predictions and sport predictions in general can be made. The fourth chapter describes the own implementation done on football predictions. It consists of an overview and the data collection and data preparation phases. In the fifth chapter the models of the implementation are described, and the results of the models are visualised and evaluated. The thesis ends with discussion and conclusion.

2. Machine Learning

Machine learning is a subset of artificial intelligence and computer science. It is a technique that learns from data using algorithms, and tries to develop to improve its accuracy. This chapter gives an overview of machine learning, machine learning algorithms, and machine learning related problems and solutions.

2.1 Overview

Machine learning is an artificial intelligence technique that improves the performance of computer systems of a task by letting them train and learn from experience. Machine learning algorithms learn to make predictions from observations and data instead of being programmed exactly what to do. However, machine learning is still quite narrow, which means the problem must be defined well so that the machine can focus on that desired activity [2]. A machine learning model is the result of the training with a machine learning algorithm on a specific data set. The machine learning algorithms must usually be modified to be used for each case depending on the input data, which differ from case to case, and therefore the machine learning models become unique. The goal of machine learning algorithms and models is to maximize the accuracy of their predictions of the desired output values [3].

The difference between problems in machine learning and manual programming is that the algorithm is unknown in machine learning. An algorithm is like a recipe, it is followed to perform tasks on the input data to generate the output data. In problems that can be performed using manual programming, both the input- and output data are known. In other problems the input data is also known, however, the output data is unknown. Only the expected format of the output data is known, such as numbers, yes or no, or categorical classifications. In other words, it is not known how to transform the input data into output data. Machine learning can be used to solve these types of problems. [4]

The first obvious step in machine learning is defining the problem. The data and algorithm one needs, depend on what kind of problem is defined. The first practical

step in machine learning is data collection. Both the quantity and quality of the data impact the results of machine learning models. The data set to use in machine learning can be a mix of one or many data sources collected by oneself or already existing data sources. The data sources can be, for example, databases, files, web services, and other APIs. The data itself can contain numerical data, text data, temporal data, and categorical data. Since the training of machine learning models is dependent on the input data, some of the most important parts of machine learning are the choice of what input data set to use and how to divide it into training- and test data sets. Machine learning models rely on data to be able to know what to do, the performance of previous models, and possible improvement points [2]. Machine learning can also handle large volumes of data that is difficult to analyse using regular data management tools, such data is called big data [2]. By leveraging large data sets, learning algorithms can achieve higher accuracy, which have made the use of big data popular in machine learning recently [4].

Since machine learning relies on data, the data set selection and feature selection are very important parts of it. When the data set has been chosen, the next step is data preparation, which consists of data cleaning, feature selection, and feature extraction. Data cleaning is needed so that the values are represented correctly, and noise and faulty data are removed, for example, missing data and duplicate values. Feature selection is then done to make the data set more efficient in the machine learning algorithm or model. Feature selection means that only the most important features of the data set are selected for further use. This will reduce the training time of the model, the model will be more comprehensible, and the result will have a higher level of generalisation. The feature selection can be made based of how much information the features give and what accuracy the features generate. However, sometimes it can be difficult to determine whether a feature is unnecessary, it depends on the other features in the data set and the dependencies between them. Feature extraction reduces the number of features by combining them, without any loss of information. A popular technique for feature extraction is Principal Component Analysis (PCA), which reduces the number of features by combining them linearly. This also makes the features uncorrelated, but all the important information is still left. [3]

Machine learning works by manipulating data using mathematical functions. Both the input and the generated output are represented mathematically. The input data must therefore be converted to mathematical representations and the resulting output data must be converted back to useful data again [2]. This is the case for text data, which is often categorical data. This conversion process is called feature encoding. The most known feature encoding technique is one-hot encoding. One dummy variable is created for each categorical value in the data set. If an observation belongs to a category, the dummy variable is set to 1, otherwise 0. Another simpler feature encoding technique is label encoding. Every categorical value is converted into a unique random number. However, this can cause problems in machine learning algorithms which interpret the values as ordinal instead of categorical.

The next step after the data preparation is to choose a machine learning algorithm for the model. It is challenging to find the most suitable algorithm for a problem. The performance of an algorithm depends on the type of data set used. The data set exhibits varying degrees of compatibility with different machine learning algorithms. However, the choice is not black and white, multiple algorithms can perform equally well, and therefore it is usually recommended to try multiple algorithms. Different types and examples of machine learning algorithms are described in Chapter 2.2.

When the machine learning algorithm has been chosen, it is time to train the model with the data set. The input data set must be split into two subsets: a training data set and a test data set. The training data set should have more data than the test data set, around 80% training data and 20% test data is a popular split ratio. The machine learning algorithm is trained using the training data set. After the training episode, the algorithm is supposed to make predictions on the new unseen data set, the test data set, based on what it has learned from the training data. The machine learning algorithms tries to find regularities in the training data to be able to make predictions on the test data. The regularities are then generalised and built into the model. The performance of the model on unseen data is called generalisation [5]. The model must never be trained using test data, nor tested using training data [3]. Both scenarios would generate a high test accuracy, however, the model has “cheated”

and will perform badly on new unseen data. The model has “cheated”, since it has already seen the data which it is tested on, and therefore it has made a perfect fit for it.

When the model has been tested, the output values and an overall result of the model will be received. To be able to know how good the model is, it should be compared with other models or previous research. The performance of machine learning models can be evaluated to find its strengths and weaknesses. Usually, the result of a machine learning model is termed as accuracy, however, it can only be calculated for classification models. In regression an evaluation metric called R-squared is used instead. R-squared measures the distance between the data points and the regression line, which essentially is the error of the predictions. The output value of the metric is a percentage between 0 and 100%. When the R-squared value is 100%, all the data point is predicted exactly right at the regression line. When the R-squared value is 0%, the model has not been able to explain any of the variation in the data around its mean [3]. Usually a higher R-squared value means a better performance of the regression model. However, what percentage is considered as a high R-squared value depends on the problem.

For classification models, the most common evaluation metrics are accuracy, precision, recall and f1-score. A popular visualisation of the performance of classification models is the so-called confusion matrix. It is a table that contains the four different classification results: true positives, false positives, true negatives, and false negatives. True positives are observations that are predicted and actually positive, false positives are observations that are predicted positive but are actually negative, true negatives are observations that are predicted and actually negative, and false negatives are observations that are predicted negative but are actually positive. In binary classification the size of the confusion matrix is 2x2 (two rows and two columns), but confusion matrices work for multiclass classification too. Then the size of the confusion matrix is n times n, where n is the number of classes. [3, 6]

The accuracy of a classification model is all the correctly classified observations divided by the total number of observations. The precision is how well the model has made positive predictions. It is measured as the true positives divided by all

positives. The recall is how well the model have classified positives correctly. It is measured as the true positives, divided by the true positives plus the false negatives. The f1-score is a measure similar to accuracy. It is the harmonic mean of precision and recall, and is measured as two times the precision times the recall, divided by, the precision plus the recall. The f1-score is higher than the accuracy when the number of observations in the classes are uneven. The output value of all these metrics are percentages usually represented as decimals between 0 and 1. The model performs better when the metrics has a value closer to 1. However, one must be suspicious of a model that produce an accuracy of close to 100%, then there might have been too little test data or the test data might have been leaked in the training phase. [3]

One way to evaluate machine learning models is by using the so-called cross-validation technique. In cross-validation the data set is randomly split into k subsets of equal size [3]. One of the subsets is selected as test data, and all the other subsets are used as training data. The test subset selection is repeated k times, so that every subset has been used as test data. The result is then the average of all the tests. This technique is called k-fold cross-validation and is the most common cross-validation technique [3]. Cross-validation will both improve the performance of the model and the generalisation of the results [6]. Another evaluation metric is the ROC (receiver operating characteristic) curve [3]. It plots the connection between the true-positive rate and the false-positive rate. The area under the curve is called the AUROC (area under ROC curve). The greater AUROC, the better overall performance of the model [6].

There are several types of machine learning approaches. The three main categories are supervised learning, unsupervised learning and reinforcement learning. Supervised learning is when the machine learns from labelled training data [7]. Labelled data is data that are tagged with some labels containing information such as characteristics and classifications. The machine is given examples of input data and their corresponding output data. By learning from these examples, the machine tries to find patterns and using them it creates mathematical models that maps inputs to outputs [8]. The model can then be used on new unseen data to predict the output data. The two most common supervised learning model groups are classification

models and regression models. Classification models maps the input data into a limited number of classes that have been predetermined [8]. Regression models maps input data into a numerical domain with any value within the range.

In unsupervised learning, only the input data is known while the output is not predetermined [4]. Since the supervisor does not tell the machine how the correct output data should look like, the unsupervised learning algorithms must find out the relations in the data themselves. They try to find similarities in the data and patterns that occurs more often than others. By finding these patterns they can tell what generally is going to happen with certain input data [4]. Unlike in supervised learning, the training data used in unsupervised learning are not labelled. There are not any predetermined classes either. The goal of unsupervised learning is therefore to find classification labels for the data [8]. The models categorize the input data into groups called clusters, by searching for commonalities in the data. Unsupervised classification is called cluster analysis or clustering [9].

Semi-supervised learning is a machine learning approach that can be put in between supervised learning and unsupervised learning. It is a mixture of them since it uses both labelled and unlabelled training data. In semi-supervised learning the amount of labelled data is usually much smaller than the amount of unlabelled data [8]. The goal of semi-supervised learning is to learn a model to predict classes more correctly with future test data than a model that has been learned using only labelled data. Semi-supervised learning reminds of the way humans learn, for children the environment is full of unlabelled data in the beginning but supervisors teach them new labelled data gradually. [10]

Reinforcement learning is a machine learning approach which learns by receiving a picture of the environment in which it operates. The machine has a specific goal in the dynamic environment and any action that the machine does in it, impacts the environment. The environment gives the machine feedback, which the machine can use to learn from and improve itself. [8]

2.2 Algorithms

The core of the machine learning model is the machine learning algorithm itself. There are multiple options of machine learning algorithms for every type of machine learning approach. However, the choice of machine learning algorithm is not black and white. Although a problem seems to fit for a particular machine learning approach, there might be other types of machine learning approaches and algorithms that works equally well. The tree main types of machine learning algorithms are regression, classification and clustering [3]. Machine learning algorithms might consist of a set of rules, a tree structure or a type of network [7].

Regression is a method that shows the relation between variables by creating a function that fits the data best. The output value is the dependent variable, which is the effect of the input variables called independent variables. When the independent variables are manipulated, they will cause a change in the dependent variable that is measured. Regression algorithms are based on supervised learning. They are specialised to predict numerical values. An example of a regression problem is to predict the price of a house. It may learn features of the house and prices of other similar houses in the same area.

The most known regression algorithm is linear regression. Since linear regression is a supervised algorithm, it uses labelled training data to train a model to predict labels of new unlabelled test data. The goal of the algorithm is to find a line that best fits the dataset by finding relationships and dependencies in the data. This means that linear regression predicts a continuous target variable, which can be used to predict the outcome of any unseen test data. The equation of simple linear regression can be written as a linear algebraic equation (y = a*x + b). However, in machine learning multiple independent variables are usually used to predict one single dependent variable. Then the equation can be expressed as in Equation 1, where y is the dependent variable, x are the independent variables, a determine the slope and e is the y-intercept. [3, 8]

$$y = a0 + a1 * x2 + \cdots + an * xn + e$$

Equation 1. *Linear regression*

Classification algorithms are also based on supervised learning. Classification algorithms predict categorical values instead of numerical values as regression algorithms do. The categorical values to which the input data should be classified into, are called classes. The classes are finite and predetermined for each problem. A classification with only two classes is called binary classification. A classification with three or more classes is called multiclass classification. An example of a classification problem is to classify images into which animal that are pictured on them. The classes could, for example, be dog, cat and horse. For a human such a problem might be simple, but for large quantities of images it would be a boring and time-consuming task. This is of course why machine learning and computers in general are useful.

There are multiple popular classification algorithms. Some of them are decision tree, random forest, support vector machine, k-nearest neighbour, logistic regression and Naïve Bayes [3]. Decision tree is a classification algorithm that shows the results as a flowchart, which often looks like a treelike structure. The decision tree classifies the input data by flowing through the tree [10]. The tree consists of nodes with edges (branches) between them. The first node has no incoming edges and is called the root. All the other nodes in the tree have exactly one incoming edge. The last nodes in the tree have no outgoing edges and are called leaves. Since each leaf has been assigned to one class, the leaves determine the output values of the algorithm [8]. Most often the decision tree is a binary tree, which means that each node has only two outgoing edges. Decision trees use conditional logic to find the path of an input value to the output value [3]. In each node there is a condition which determines to which node the data point should continue to. In a binary tree the condition can be seen as an if-else statement, true or false, or yes or no. Decision trees can also be used for regression. Then the leaves are numerical values instead of classes.

Random forest is a classification algorithm which consists of decision trees. However, the random forest algorithm uses multiple trees instead of just one as the decision tree algorithm. A random forest trains the trees with random subsets of the original data set, and the result is determined with a majority vote between the trees. This makes random forests more generalised than individual decision trees [3].

Random forests can also be used for regression. Then the numeric prediction is made by taking the average of the output of the trees [3]. An example of a random forest structure and its decision trees is visualised in Figure 1.

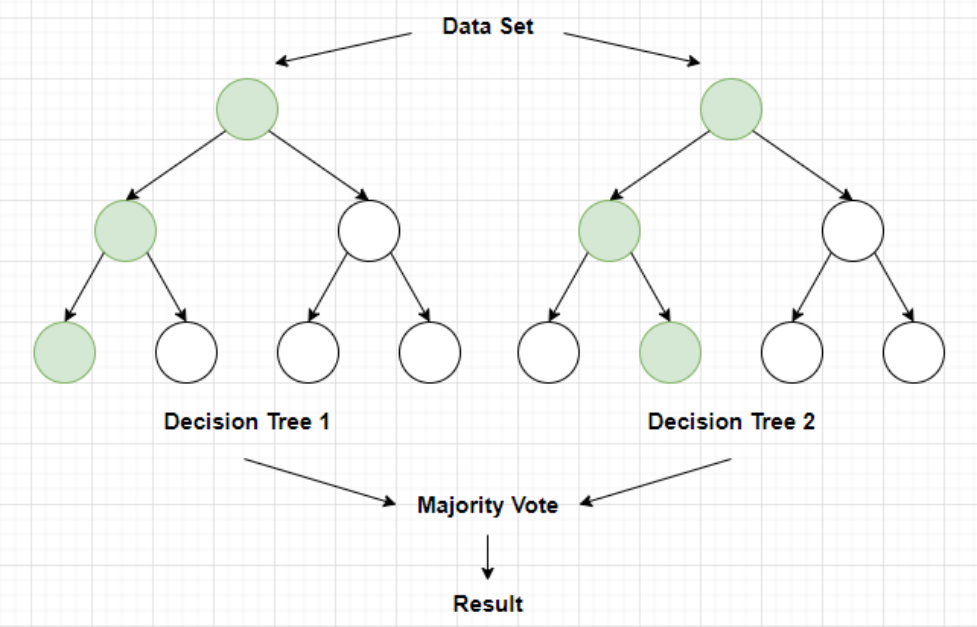


Figure 1. *A random forest structure where the green nodes create the decision path.*

Support vector machine is a classification algorithm that can be used for both classification and regression problems. It classifies the input data into two classes, which makes it a binary linear classifier. The data points in a space are separated into the two classes so that the gap between them is as wide as possible. New test data is then mapped to a class based on which side of the gap they are on. Support vector machine can also perform nonlinear classification. This can be done using the so-called 'kernel trick', which projects the data on a higher dimension feature space. [10]

K-nearest neighbour is a classification algorithm that can be used for both classification and regression problems. It classifies data points that are close to each other to the same class. It looks at the k nearest neighbours of a new point, and the new point is then added to the class which most of its neighbours are belonging to. K can be any number, but an odd value of k makes the occurrence of a tie less probable. In case of a tie, the class can be chosen based on the class of the closest neighbour, randomly, or after decreasing or increasing the value of k. K-nearest

neighbour is a simple algorithm, but it can be effective for problems like searching for similar items. [3]

Logistic regression is a supervised algorithm related to linear regression. The most significant difference between them is that linear regression is a regression algorithm while logistic regression is a classification algorithm, despite its name. Logistic regression predicts categorical dependent variables while linear regression predicts continuous dependent variables. Logistic regression is usually used as a binary classification algorithm and therefore the output can be one of two classes, like true or false, and yes or no. The output value is determined using probabilities of a data point being in one of the two classes. The probabilities are calculated using a sigmoid function, which is an activation function. It multiplies each feature with a weight and then add them up. The output values to which the sigmoid function maps the input values, must be between 0 and 1. The data points that are lower than a threshold value, are classified to one of the classes, and the data points that are higher than the threshold value, are classified to the other class. The equation of logistic regression can be expressed as in Equation 2, where y is equal to the outcome probability, and z is a model parameter which includes the x-value, the midpoint of the curve and the scale of the curve. The resulting probability will always be in the range between 0 to 1, which is required in a sigmoid function. [3, 8]

$$y = \frac{1}{1 + e^{-z}}$$

Equation 2. *Logistic regression*

Logistic regression can be seen as the cousin of the linear regression. In a sense, logistic regression is linear regression on which a sigmoid function has been applied [3]. In Figure 2. the difference between linear regression and logistic regression is visualised. Logistic regression produces a S-shaped curve which is called sigmoid curve or logistic curve. Since the output values must be between 0 and 1, input values that go to negative or positive infinity will have an output value of 0 and 1 respectively. Linear regression on the other hand, simply produces a continuous line. However, a common problem for both logistic regression and linear regression

is multicollinearity [3]. Multicollinearity is when an independent variable is strongly correlated with another independent variable. It is a problem since it makes it difficult for the model to find significant effects of changing the independent variables, although they might be of great importance.

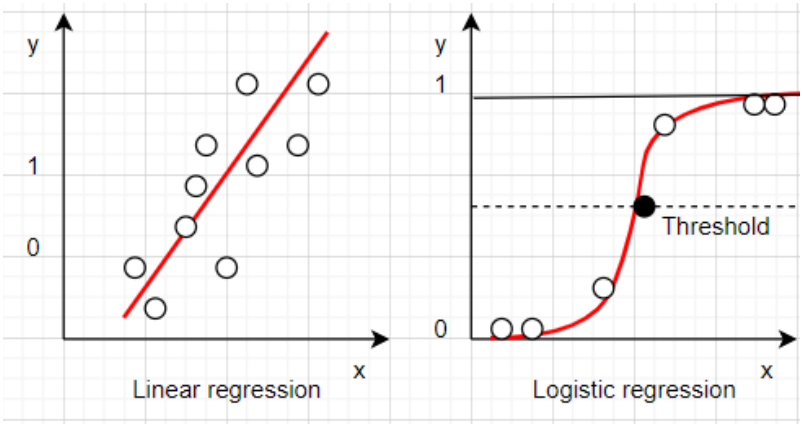


Figure 2. *Linear regression on the left, logistic regression on the right.*

Naïve Bayes is a classification algorithm and a special case of a so-called Bayesian network. A Bayesian network is a graphical model based on probabilities. The nodes in the graph are random variables and the edges are conditional probabilities of the variables. The dependencies between the random variables are calculated using Bayes theorem. It describes the conditional probability of an event, which means the probability of an event happening given that another event has happened before. The Bayes Theorem is based on previous knowledge of the conditions that are related to the event. The equation of Bayes Theorem is expressed in Equation 3, where A and B are events. It is expressed as the conditional probability of A happening when B is true, is equal to, the conditional probability of B happening when A is true, times the probability of A, divided by the probability of B. [3, 11]

$$P(A|B) = \frac{P(B|A) * P(A)}{\mathrm{P(B)}}$$

Equation 3. *Bayes Theorem*

The graph of a Bayesian network is a directed acyclic graph (DAG), because of the conditional probabilities and dependencies between the nodes in the graph. In such a graph, there are no cycles or self-connections allowed, which means that each

edge goes from one node to another without creating a loop. Learning a Bayesian network can be divided into two main steps: structure learning and parameter learning. Structure learning is finding out the directed acyclic graph that best represents the relationships in the data. This can be done by using algorithms like the DAG search algorithm and the K2-algorithm. In the beginning they give all DAG structures the same probability, and then they search for the structure that gives the maximum probability for the data, which is called the Bayesian score. Parameter learning means learning the distribution of conditional probabilities. The maximum likelihood estimator is the method used for parameter learning. It estimates the parameters of the probability distribution by maximising a likelihood function so that the probability of the data is as high as possible. [11]

Naïve Bayes is a more restricted form of Bayesian network. Naïve Bayes assumes that all the attributes are independent of each other [12]. The class nodes should have no parents, and are therefore the root nodes. Each node can only have one parent and the nodes corresponding to the attribute variables should have no connections or edges between them [12]. An example of the Naïve Bayes structure is visualised in Figure 3. The c-node is the class node and the a-nodes are the nodes corresponding to attribute variables. Naïve Bayes can be used for both binary- and multiclass classification and it works well with both small and large datasets [3]. It usually performs better using categorical input data than using numerical input data. Naïve Bayes is good at predicting data based on historical results [3].

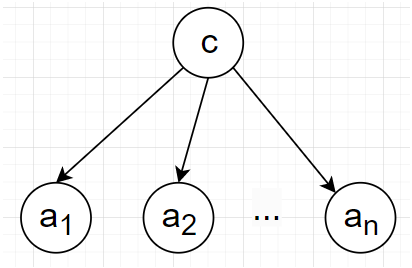


Figure 3. *Naïve Bayes Structure*

Naïve Bayes is a popular classification algorithm because it is simple to construct since it doesn´t need structure learning. The classification process is also efficient since the attributes are assumed to be independent. However, it also creates a problem when selecting features for the model. Features that are correlated to each

other, may not perform well in a Naïve Bayes model. This can be solved by selecting feature subsets using, for example, the forward selecting method or the best-first search algorithm [12]. The most significant difference between Naïve Bayes and logistic regression is the classification type. Naïve Bayes is a generative classifier, which means that it learns the probability distribution of the data and understands how the data looks like in general. Logistic regression on the other hand, is a discriminative classifier, which means that it learns decision boundaries and divides the data using those boundaries [8].

Clustering is a type of unsupervised machine learning algorithms. Clustering is the unsupervised equivalent to the supervised classification algorithms. The goal of them both is to divide the input data into separate classes or clusters, however in clustering there are no predetermined classes, like there are in classification [9]. Instead of dividing the input data set according to how the training data was divided, clustering divides the data using statistical methods where data with common features are grouped into the same clusters. Clustering can also be seen as dividing the original data set into subsets with similar data [9, 13]. There are different types of clustering methods, for example partitional clustering, density-based clustering and hierarchical clustering.

Since the human intelligence is based in our brains, they are an interesting topic in artificial intelligence and machine learning. Artificial neural networks are learning algorithms that imitate the functions of biological neural networks in the brain [4]. The brain consists of billions of neurons which are connected to each other. The neurons are information messengers. They use different frequencies that are summed when going from neuron to neuron [4]. They can be seen as weights, which can be represented in a computer as zeros and ones. A neural network consists of multiple layers. The first is the input layer, in the middle are the hidden layers and last comes the output layer [3]. The neurons receive their input from all neurons on the previous layer, and their own output are forwarded to every neuron on the following layer [4].

As machine learning is a subset of artificial intelligence, deep learning is a subset of machine learning. Deep learning is based on artificial neural networks, and ‘deep’ means that there are multiple layers used (at least two hidden layers) [3].

Deep neural networks woks like artificial neural networks. Each layer combines the values from the previous layer, and therefore they develop all the time by learning more and more complicated functions [4]. The abstraction increases for every layer, which means that unnecessary data that do not help to solve the problem are hidden or not considered. Deep learning uses hyper parameters, chosen by oneself, to set up the model. The most important hyper parameters are the number of hidden layers, the number of neurons in the hidden layers and weight initialisations [3]. Deep learning has become interesting recently since it is not focused to do one specific thing, but can be used for many different purposes. It can also make use of large amounts of data, like big data, if the computer systems are powerful enough. Deep learning also finds the best features from the data set by themselves, which makes the work simpler [4]. In other types of machine learning, one must do the feature extraction and feature selection by hand or using algorithms, but deep learning does the feature extraction automatically. Neural networks are mostly used for classification problems, but can also be used for regression problems.

Three popular neural network types in deep learning are artificial neural network (ANN), recurrent neural network (RNN), and convolutional neural network (CNN). In artificial neural networks, a biological neuron is represented as a mathematical model called perceptron. It takes features as input and tries to classify them into classes by separating them with a line or plane. ANN uses sigmoid functions for the separation, and therefore the ANN algorithm is like logistic regression. However, logistic regression only uses one perceptron, while ANN uses one for each neuron in their networks. In addition, the output of ANN is only the class association, but in logistic regression the probability of a variable belonging to a class is also provided. RNN is a neural network architecture that can handle sequential data. It is especially suitable for language translation, speech recognition and text generation. In an RNN, the neurons can have connections between themselves in the same layer in addition to connections between the layers [4]. CNN is a specific type of an artificial neural network, that is especially useful in image recognition. In ANN, each input is related to only one pixel in the image, but in CNN the spatial relationship between the pixels is also considered. [14]

2.2.1 Multiclass Classification

The most basic classification algorithms are binary classification algorithms, which only have two classes. Multiclass classification algorithms, which have more than two classes, are much more complicated. Some multiclass classification problems may have more than thousand different classes, for example in image recognition and text translation [15]. Multilabel classification is similar to multiclass classification, however, in addition to having many classes multilabel classification can also assign one or more classes to each observation [15]. The difference between the three types of classifications can be seen in, for example, image classification. If the images only picture 'a cat' or 'a dog', it is binary classification. If the images picture 'a cat', 'a dog' or 'a human', it is multiclass classification. However, if the images pictures 'cats', 'dogs' and 'humans' and they can be mixed in a single image, for example 'a human' and 'a dog' in the same image, it is multilabel classification.

Many binary classification algorithms can naturally be used for multiclass classification, while other can be transformed to be used for it. Algorithms that can be used naturally for both binary- and multiclass classification are decision trees, k-nearest neighbour, Naïve Bayes and neural networks. Decision trees classifies the observations using a tree-like structure. Each leaf in the tree represents one class. In binary classification the decision tree only has two outputs, and therefore two leaves. However, more leaves can be added, thereby representing multiple classes, which enables multiclass classification. Since the random forest classifier is built out of decision trees, it will also work for multiclass classification. [16]

K-nearest neighbour can naturally be used for multiclass classification. New data points will be added to the category which most of its neighbours belong to, also if there are more than two classes. Naïve Bayes can also be used for multiclass classification. The observations will be categorised to the class with the highest probability of all classes, whether there are two or more classes. In neural networks the number of output neurons can be increased to enable it for multiclass classification, so that each output neuron represents one class. Each neuron tells the probability of the observation belonging to that class. [16]

Other binary classification algorithms can be transformed to be used for multiclass classification using either the One-vs-rest strategy or the One-vs-one strategy. The One-vs-rest strategy (One-vs-all, OVA) uses one binary classification model for each class, where the class is put against all other classes. The One-vs-one strategy (All-vs-all, AVA) uses one binary classification model for each pair of classes. In both strategies the observations are classified to the class with the highest score. Logistic regression is a binary classification algorithm that can be transformed using one of these strategies to be used for multiclass classification. [15, 16]

2.3 Problems and Solutions

To be able to receive improved and valid results, machine learning algorithms are usually computed multiple times. A so-called epoch occurs each time the whole training data set is used for training the machine learning model. When the problem and input data is complex, more repetitions are needed [2]. However, the result should not radically change even though the input data changes. If it does, the model has not been good enough and so-called overfitting or underfitting has occurred.

Overfitting and underfitting are the most common causes of poor results in a machine learning algorithm or model. Overfitting is when the ability of a model to solve the problem suddenly is not increasing anymore [17]. It occurs in the training process when the model starts to also learn noise. It means that a model cannot generalise seen data to unseen data well enough [18]. Overfitting causes the model to perform well on the training data, but badly on new testing data. Instead of learning how the data behave, the model has learned all the data by rote. That means it has also memorised noise and abnormalities in the training data [17]. When the model encounters new type of information in the test data set, the model will struggle to understand it according to the procedure learned using the train data set. So, even if the model is a perfect fit to the training data, it is not guaranteed that the prediction accuracy of the testing data will be high [7].

The causes of overfitting are usually a too small train data set, too much noise in the train data set, or too many inputs in the algorithm. There are multiple strategies for reducing and avoiding overfitting. A method that is good at avoiding overfitting

is the so-called early stopping method. Its purpose is to stop the training before the accuracy of the model stops increasing or starts decreasing [17]. The training accuracy increases always overtime, but the validation accuracy increases in the beginning but slows down and eventually starts decreasing in the end. At the same time the training error decreases overtime while the validation error first decreases but after a time it starts to increase [17]. The difficult part is to find where the point is that the accuracy and error rate shifts at. The point can be found by following the validation accuracy while the training is going on. After each epoch the accuracy is evaluated and when the accuracy stops improving or starts worsening the training should be stopped [18]. Often validation is made between the training and testing in the early stopping method. Validation helps to find the best values for the hyper-parameters (for example size of neural network). This increases the generalisation level and reduces the bias while increasing the variance [18]. When the model has been finalised using the validation data set, finally the test data set is used to test the final model.

Since noise learning is one cause of overfitting, one solution to reduce the overfitting is to reduce the amount of noise. That can be done by removing unused or unimportant data and features. Unimportant features can also be given less weight than the other features so that they are not taken into consideration as much as the other ones [18]. Often overfitting can be reduced by expanding the training data set. Especially if the model is complicated, more parameters are needed to achieve good results [18]. Small data sets are overall more vulnerable to overfitting than large data sets [17].

Underfitting is the opposite of overfitting. Instead of having a too good model that learns all the test data as in overfitting, underfitting occurs when the model is too simple to represent the variability of the data [17]. Unlike overfitting, underfitting means that a model performs badly on both training- and test data. The solutions for reducing or avoiding underfitting is practically like the solutions for overfitting. To achieve a higher accuracy of the model the important features must be increased and therefore a larger data set must be used. Another solution is to complicate the model or increase the training time or number of epochs and iterations.

In Figure 4. the different results of a supervised model are visualised. When underfitting has occurred, the model has not been able to represent the data in an optimal way. Such a model will have high error rate in both the training- and test data. When the model is balanced, the accuracy is optimised and the model can be used on new unseen data still with high accuracy. The model ignores outlier points and noise. However, when overfitting occurs, the model has been fitted to the training data too well. It considers every point in the training data, since it has not been generalised enough. The model will not be able to produce high accuracy on unseen data.

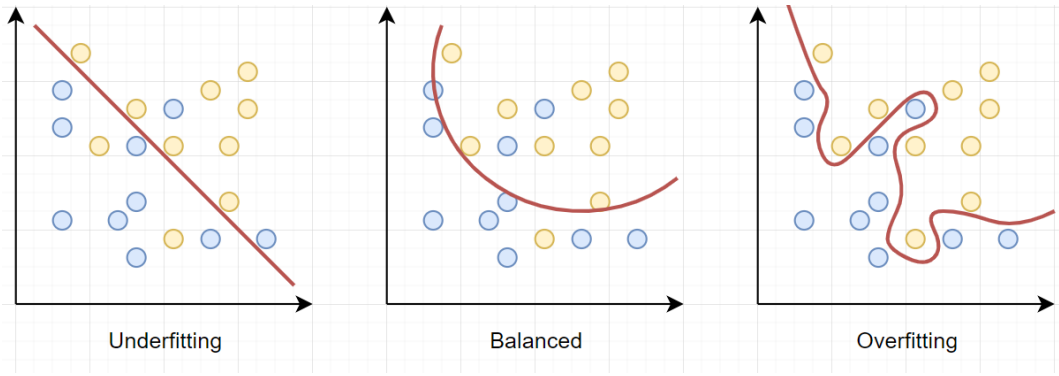


Figure 4. *From the left: Underfitting, balanced and overfitting in supervised learning.*

Overfitting and underfitting occurs especially in supervised learning. However, there are equivalents to them in unsupervised learning as well, which occurs in clustering. In unsupervised learning the equivalent to overfitting and underfitting is the problem of finding the optimal number of clusters to use [9]. The equivalent to overfitting is when the number of clusters are too many in proportion to the data points. This will lead to the clusters only including a few data points each, and that they are not clustered according to their similarities. The equivalent to underfitting is thus the opposite, which is when the number of clusters are too few so that the data points are clustered together although they do not have many common traits between them. The different results of a clustering model are visualised in Figure 5.

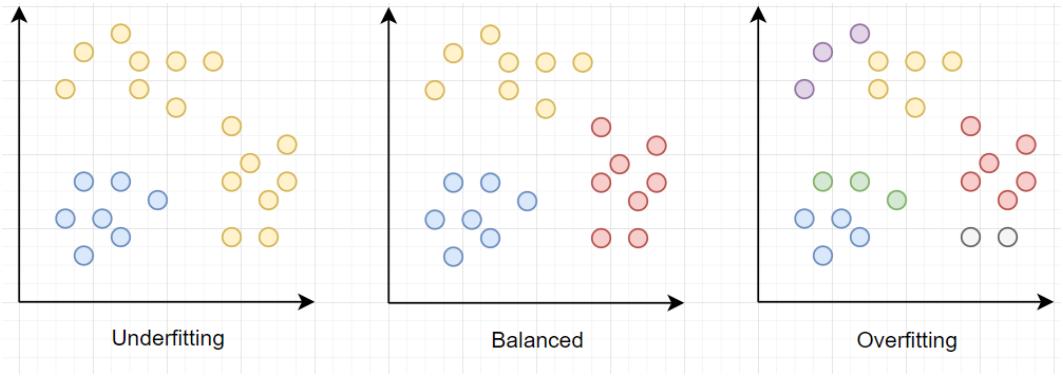


Figure 5. *From the left: Underfitting, balanced and overfitting in unsupervised learning.*

Another cause of overfitting and underfitting are too many inputs in the algorithm. When there are too many inputs, the model will have a high accuracy on average but with a low consistency. For some data sets the model may work well, but for other data sets it may work badly. To solve this, the so-called bias-variance trade-off must be improved. Bias-variance trade-off means the balance between the bias and the variance in a model [18]. Since we want the machine learning model to be generalised and perform as good as possible on new unseen data, the so-called generalisation error should be minimised [5]. This can be done by keeping the bias and variance in balance. Bias in machine learning is how much a model learns unnecessary things, like noise, and makes too many assumptions. Variance in machine learning is how well the model can handle changes in the training data set [5]. Bias can be seen as the training error rate and variance as the testing error rate. It would be optimal to include all the patterns found in the input data, but still have a high level of generalisation of the data, but they interfere with each other so they cannot be achieved simultaneously. A model with a low bias and a high variance will result in overfitting [5]. On the other hand, a too high bias means that the training error rate has been high and this will result in underfitting. A model with a low bias is usually more complex, and a model with high bias is usually less complex [5]. Therefore, the bias-variance trade-off can also be seen as a trade-off in complexity [18]. Although, these points can help in building a successful model, everything depends on the data set and the model must be build different from case to case. The key in building a machine learning model while avoiding overfitting and underfitting, is to compromise between accuracy and consistency [18].

3. Previous Research

Predicting results in sports have always been popular, especially betting on the winner or exact result. However, in the recent years the ability to predict things have increased drastically thanks to development in techniques and data gathering. Using machine learning we can now find relations between different factors that influence the result of sports, that we were not aware of previously. The increase of available data about sports, also helps to improve the ability to make predictions. Nowadays the results of a game or competition can be found in seconds and statistics of them can be analysed in detail. This chapter describes how sport predictions can be made in general and how especially football predictions can be made.

3.1 Sport Predictions

Predicting results in sports are in the interest of various groups of people. Managers and agents want to predict the results of competitions to be able to change the outcome to their advantage. The teams or athletes may want to know their status compared to other competitors ahead of a competition. Fans want to predict the results to know how well their athlete or team will do. Predicting sport results are also of high importance in so-called fantasy leagues, which have grown in popularity recently. They are based on the results of real-world sports, and the participants receive points based on how well they have predicted the outcome. The predictions are usually made by choosing a team of players from different teams, that the participant think is going to do well in the near future. Fantasy leagues are most often based on team sports, for example football, American football and ice hockey.

Since the business of sports in general is enormous, predictions of results in sport are of course also associated with money. Sports betting is a large market globally, especially thanks to the growth of online betting. It is expected to grow even more when more and more people around the world can watch sports [19]. The most popular sports in betting differ from country to country, but overall, the most

common ones are team sports like football, basketball, cricket and American football, racing sports like track cycling and horse racing, and other sports like tennis, Esports and boxing [19].

Betting is when one predicts a result and then playing money on it. There are two main types of sports betting, either one must predict the exact score of the game, or one must predict the overall result of the game, which essentially is who will win. When predicting the overall result, there are three possible outcomes; win for the home team, draw, or win for the away team. Home and away are used to explain which team are playing on their home ground. If the game is played on a neutral ground, the home team are usually drawn randomly. The three different outcomes are usually represented as '1' for a home win, 'X' for a draw and '2' for an away win. If the right answer is predicted, one will retrieve more money than the stake depending on the odds, however if the prediction is wrong the stake is lost.

Betting odds are becoming more and more popular due to online betting, and the number of sports and competitions available on betting sites are increasing all the time. Betting odds are said to be the most accurate probability forecast for sports [20]. It is especially useful to use odds in sports where the probability of the outcome varies from game to game. In sports or games where the probabilities of the outcome are even, there are no direct use of odds. Betting odds can be seen as the potential win relative to the stake.

The probability is how likely it is that an event will happen. The range is between zero and one, where zero means that the event is impossible to happen and one means that the event will happen with all certainty. Odds are related to the probability and can be calculated using a simple formula. The odds are the ratio between the probability of an event to happen and the probability of an event not to happen. For example, if the odds of an event to happen are 1:2, it will happen in one out of three times and the probability of it is 0.33 or 33%.

Betting odds however are usually represented a little bit different from normal odds. A traditional way of displaying betting odds is using fractional odds. Fractional odds are the win to be paid out to the bettor in case of a correct bet, relative to the stake. In case of a win, one will retrieve the stake plus the fractional odds times the

stake. This means that only the potential win can be seen from the fractional odds and not how much one will retrieve in the total with the stake included. For example, if the fractional odds of a game are 1/2 and the bet is correct, one will retrieve the stake plus half of the stake as a winning price. The probability of it to happen is 67%. If the fractional odds of a game are 1/1 and the bet is correct, one will retrieve the stake two times. The probability of it to happen is 50%. If the fractional odds of a game are 2/1 and the bet is correct, one will retrieve the stake three times. The probability of it to happen is 33%.

Betting odds can also be displayed as decimal odds. Decimal odds are more comprehensible than fractional odds, since they directly tell how many times the stake that one will retrieve if the bet is correct. Using the same examples as for fractional odds; if the fractional odds are 1/2 the decimal odds are 1.5, which means the money one will retrieve if the bet is correct will be 1.5 times the stake. If the fractional odds are 1/1 the decimal odds are 2, which means the money one will retrieve if the bet is correct will be 2 times the stake. If the fractional odds are 2/1 the decimal odds are 3, which means the money one will retrieve if the bet is correct will be 3 times the stake. So fractional odds can be converted to decimal odds by converting the fractions to decimals and add one.

A bookmaker is a person or organisation that organise sport betting events. For the bookmakers to make a profit from the betting, they must manipulate the odds to not match the true probabilities of the outcome. If the odds would be the same as the probabilities of the outcome, all the money put as stake would be given back as wins to the bettors, and there would be no profit left for the bookmakers. The bookmakers manipulate the odds by increasing the probabilities of all the possible outcomes so that the total probability goes over 100%. For example, the true probabilities of the outcome of football games have been proven to be less than the odds with a few percent [21]. This will cause the total wins paid out to the bettors to be lower than the total stakes. The leftover is the profit of the bookmakers. This also means that there are more people losing their money in sports betting than there are people winning money.

Bookmakers takes help from data- scientists and analysts to make predictions of sport results [21], but they don´t predict the results only by themselves. They use

the predictions of the public to balance the odds so that they will make more profit. Still, it is quite surprising how well the odds match the result of games, which means that the public can predict sport results quite accurately without the help of sport experts, bookmarkers or machine learning models.

However, except that many people are addicted to and lose their money on sports betting, match fixing and money laundering are also problems caused by sports betting. When there are much money on the line, some people want to take risks and illegal actions to be able to win. Match fixing can be done by bribing the referee or competitors of the game, and means that they try to manipulate the game to have the outcome of the match-fixers bets. According to a report made by Sportradar [19], the occurrence of match fixing is increasing in sports worldwide, due to the fast growing of the market. The most cases of suspicious games have occurred in football, e-sports and basketball. In football most cases occur in the lower-level competitions, where there are not as many organisations that governs or monitors the betting. The players are more vulnerable targets for match-fixers in lower league tiers since their athlete education and income is not at the same level as in higher-level leagues.

According to the annual match fixing report by Sportradar [19], the most popular sport in betting is by far football, with a market of over 700 billion Euro and over 50% of the sports betting made in 2021. Other popular sports in betting in 2021 were tennis and basketball. In these three sports, the greatest number of suspicious matches also occurred. The most popular competitions in sports betting in 2021 were UEFA Champions league (football), English Premier League (football), and Indian Premier League (cricket). According to Sportradar, match fixing has become more widespread after the COVID-19 pandemic, spreading to new sports and lower-level leagues.

There are significant differences between sports and competitions in how well their results can be predicted. One of the primary factors is how much the strengths of the competitors differ from each other. If the difference in strength is high, the result will be more predictable. This can be the case in so-called closed leagues, where the teams do not change depending on their performance after a season. This means that bad teams still must play against teams that are considerably stronger than

them, instead of being relegated to a lower-level league with teams with more matching strength. Leagues with one or a few teams that are relatively superior to the other teams, will also be more predictable. Another factor that can improve the predictability is if seeding is used in knockout competitions. Seeding is when stronger competitors are drawn to compete against weaker ones, so that the strong competitors have a fair chance to reach the final (if the games would be drawn randomly, two strong competitors might have to face each other already in the first rounds of the knockout stage). Sports and competitions that can end in a draw, is usually more difficult to predict than sports with only 'win' or 'loss' as possible outcomes [22].

One of the most effortless sports to predict are sports with high difference in the strength of the competitors, like Formula 1. The team with the best cars and the best driver of that team will win most of the races. Some of the most difficult sports to predict are football and ice hockey. Since they are relatively low scoring sports, it is difficult to predict the outcome although one of the teams may have a strength advantage. Low scoring sports are more difficult to predict because the probability that a weaker team can beat a stronger team is higher since one single goal is more crucial and can be scored with luck. Sports like basketball and tennis are more predictable since they are high scoring sports, in which luck and unpredictable events have less importance on the outcome.

Bunker and Thabtah [23] have proposed a framework for sport result prediction. It includes six main steps: domain understanding, data understanding, data preparation, modelling, evaluation and deployment. It is like the procedure of most machine learning problems. Domain understanding means to know the problem and the characteristics of the sport in question. The problem could be to predict the overall result, the exact numerical results of the matches or to predict which matches it should pay off to bet on. The characteristics of the sport are of course important to know to be able to find the factors that impacts the outcome of matches. The data used in the model must be understood as well. The data can be match statistics, team related, or player related. The amount of publicly available data online about sports have been increasing, which makes it more straightforward for researchers and hobbyists to collect the data needed for their sport predictions. If there are no

suitable data sets publicly available, it is of course possible to extract online data to make an own database and data sets. [23]

In team-sports the most important feature types for predicting the outcome, are previous match statistics, team related features and individual player related features, of both the competing teams [24]. In individual sports the most important feature types for predicting the outcome are player related features and previous match statistics [25]. However, the prediction model will not perform well if the features are not processed before use. Most researchers in sports predictions do feature selection and feature extraction before using the machine learning algorithm [26]. As for all machine learning problems, one of the most important aspects to achieve a good result in sport predictions is the feature selection. The original data set can be divided into feature subsets [23]. The different subsets can include features like, for example, match related features, team related features, standings features and betting odds features. The subsets can be tested individually combined with a machine learning algorithm to select the best features. Feature selection algorithms can also be used to find the most important features.

Data preparation is important to achieve valid results out of the machine learning model. Feature extraction must be made on all the used features. Beyond that, match related features must be handled differently to other external features [23]. Since the match statistics are known only during or after the match has been played, they must be treated differently. Usually, an average of each match related feature is calculated to be used in the model. The best number of matches to be used for the averaging process has been found to be the past 20 matches [27]. The target variable must then be defined. For classification problems in sport predictions the target is the class variable, which usually consists of three values (home win, draw, or away win), but can also consists of only two values (home win or away win) [23].

The choice of machine learning algorithms according to the selected features is also of great importance. The most common machine learning algorithms in sports predictions are classification algorithms. The wanted output is usually the overall result of the game as win, draw or away win. The other option is to predict the total score of a game. Regression algorithms can be used for such numerical prediction problems. The most used classification algorithms in sport predictions are different

types of artificial neural networks [23]. Other classification algorithms suitable for sport predictions are Bayesian networks and logistic regression. Artificial neural networks are suitable for sport predictions since they can be flexible on how to interpret the class variables. They can also deal with dependencies between the input variables, while Bayesian networks on the other hand see all input variables as independent to each other.

In the modelling phase the chosen algorithms can be used for testing on the different feature subsets, to find the best possible combination. Sport prediction data sets can be split into training- and test data sets in multiple ways. Bunker and Thabtah [23] argue that the best option is to do the training test split round-by-round, especially for sports where each team plays exactly one match per round. The order of the matches still must be considered; matches should only be predicted based on previous matches. They argue that the best way is to use a few rounds as test data equally spread within all the rounds. For each test round, the training data should consist of all the previous rounds. The average of the accuracies of all the tests could then be used as the overall model accuracy. Another way to do the training test split round-by-round is by using a block of the first rounds as training data, and a block consisting of a few of the last rounds as test data. This is slightly simpler since only one accuracy is produced and an average of the tests is not needed. If the amount of data in the original data set is higher and includes multiple seasons, the training test split can be done by splitting season-by-season instead of round-by-round. Although, in most sports the strength and player of the teams changes each season and therefore it could be misleading to do the predictions based on previous seasons. Another option is to predict the results match-by-match. In that case the training data is all the previous matches and the match in question is the test data. However, if the goal is to predict the outcome of a whole season, this option demands the training data to be updated for every match. [23]

Machine learning based sport predictions most often use prediction accuracy as the main evaluation metric. A classification matrix can be used to visualise the results in classification problems. Since class values of sport prediction data sets usually are quite balanced, classification accuracy can be used to measure the model performance [23]. In many sports there are slightly more home wins than away wins

due to the home advantages. There are more wins than draws, however, since there are two types of wins the class values are still quite balanced. When predicting the overall result of a game (home win, draw or away win), the prediction accuracy of the model must be over 33% to be higher than chance. A model with a prediction accuracy below 50% can still be considered as unsuccessful. A model with a prediction accuracy of between 70% and 90% is good, although it is subjective and depends on the type of problem in question. If the prediction accuracy is close to 100%, it is usually a sign of overfitting rather than a perfect model. It is difficult to find an average prediction accuracy of all sports predictions since they cannot all be compared to each other. However, most research papers have claimed a prediction accuracy of between 60% and 80%, with around 70% being the average [26].

Since most sport prediction models use data of previous game results, cross-validation is not appropriate to use for evaluating sport predictions [23, 28]. If temporal data is used in cross-validation, the classification algorithm may see training data that has happened later in time than the test data, which would not be possible in a real-world scenario. If sport predictions are supposed to be made on game to game, the chronological order of the data must be preserved, and therefore cross-validation is not a suitable option for evaluation. However, it is not a problem when match results are not considered or when only previous match results are used for a whole season at a time.

3.2 Football Predictions

Football is one of the most difficult sports to predict. That a team has a high strength advantage over the opponent does not automatically means that it will win. Since football is a low scoring sport, the probability of a weaker team winning or drawing against a stronger one, is higher than in many other sports. In football, the results may depend on features of individual players, the tactics and motivation of the teams, the recent form of players and teams, and external factors like weather and home ground [24]. Which ground the match is played on, is an important factor due to the home advantage phenomenon. The team playing on their home ground will have more experience of the weather conditions and the pitch, which type and

condition may vary between grounds. The away team must travel to the ground of the opponent and the travel distance might impact their energy. In addition, there are usually more home fans than away fans in the stadiums, which will help the home team since it receives more positive support while the away team might receive disturbances and heckling.

The percentage of home wins, away wins, and draws, are different from league to league and changes between the seasons. Overall football matches most often end up in a home win, followed by away win, and least often in a draw. Around 45% of matches end up in a home win, 30% in an away win, and 25% in a draw [29, 30]. During the 2020/2021 season many leagues played without fans due to the COVID-19 pandemic. During that season the home win percentage decreased to around 40%, and the away win percentage increased to 35%. This proves that home advantage, especially when fans are attending, has a great impact in football. [29]

Although football is difficult to predict in comparison to other sports, studies have shown that it is possible to achieve a high prediction accuracy. According to the review on sport predictions made by Horvat [26], most football match predictions achieve a prediction accuracy between 55% and 70%, but there are a few studies that claim they have achieved prediction accuracies up to 90%. In the last years several studies have been made on making football match predictions, some more determined to achieve the highest possible prediction accuracy and other more focused on finding a successful model for future research serving as a benchmark.

Owramipur et al [1] used Bayesian networks in their football match predictions in 2013. They achieved an outstanding prediction accuracy of 92%, however, they only focused on one team for one specific season (FC Barcelona, Spanish La Liga 2008-2009). They used multiple types of features for their predictions, both psychological and non-psychological. The values of main features that affected the result mostly in favour of one team, were when the average age of the players was in the middle, the result of the five last games were good, there were not any injured main players, the psychological state of the players was good, the performance of all the players was high, and the weather was good.

Shin and Gasparyan [31] used virtual data collected from a video game to predict football matches in 2014. They collected data of player attributes from 'FIFA 2015', which is a football video game made by EA Sports. The matches they predicted were one season of Spanish La Liga. They used multiple models for their prediction, with algorithms like, for example, the support vector machine, logistic regression and Naïve Bayes. The results using virtual data were compared with results using real data, which were collected based on average performances in previous matches. Some models using virtual data achieved a higher prediction accuracy than models using real data. The best model for both data categories were the one using linear support vector machine. Using virtual data, the model achieved an accuracy of around 79%, while using real data the model achieved an accuracy of around 73%. Their research proved that virtual video game data is not always totally artificial and imaginary, but instead it could be used for predicting real world scenarios with good results.

Tax and Joustra [28] made football match predictions for the Dutch League Eredivisie in 2015. They used different feature sets including match statistics and betting odds from a period of 13 seasons. They used multiple classification algorithms, which they compared with each other using the different feature sets. When all match features were used, Naïve Bayes and artificial neural network resulted in the highest classification accuracy of 54.7%. When a hybrid model of both the match- and betting odds features were used, they achieved a classification accuracy of 56.1%.

Prasetio and Harlili [32] used logistic regression to predict football match results in 2016. However, they only used two possible outcomes: home win or away win. They used multiple seasons of the English Premier League for training, to predict the 2015/2016 season of the same league. The main features they used were "home offence", "home defence", "away offence", and "away defence", out of which the defence-related features were the most important. The prediction accuracy of their model was 69.5%.

Razali et al [33] used Bayesian networks in their research on football match predictions in 2017. They predicted the results for three seasons of the English

Premier League (EPL), using match statistics as the main factors. They achieved an average accuracy of 75.09%.

Chen [34] made football match predictions based on player features in 2019. The predictions were made on multiple seasons of Spanish La Liga. The origin of the player features was from the football video game FIFA. Three algorithms were compared: convolutional neural network, random forest and support vector machine. The best result was achieved using the convolutional neural network with an accuracy of over 57%. Using the other two algorithms an accuracy of around 54% was achieved.

Rahman [35] predicted football match results using deep learning in 2020. The data set used contained rankings and team performances based on previous football match results of international teams. The matches predicted were the matches in the FIFA World Cup 2018 (international football world championships). The predictions were made using artificial neural networks and deep neural networks. The model predicted the overall result of FIFA World Cup 2018 group stage matches correctly with an accuracy of 63.3%.

Taspinar et al [30] made football match predictions for Italy Serie A League in 2021. They predicted the match results for multiple seasons using features based on match statistics. When using a logistic regression model on all the features, they achieved an average classification accuracy of 88.85%. However, they concluded that the accuracy could be improved by selecting effective features. When they only selected the most important features, the accuracy increased to 89.63%.

As for sport predictions in general, the most used machine learning algorithms for football predictions are Bayesian networks, logistic regression and artificial neural networks. It is recommended to test several of them to find the best method in combination with a particular data set [22]. All three algorithms suit well for classification problems like predicting the overall result in football matches. However, the prediction accuracy could be improved by treating the match result as only two classes: home win and away win [30]. Although this would only be realistic when predicting knockout competitions where there are no draws, since in football leagues draws are usually allowed.

Since the outcome of football matches are not evenly balanced between home wins, away wins, and draws, the machine learning models will also tend to predict the outcomes unevenly. It is most difficult to predict draws correctly since there are least number of matches with such an outcome. Moreover, the draw is 'in between' home win and away win, which means that the statistics of a match ending in a draw are similar to other situations (home win and away win on each side) [30]. On the other hand, home wins and away wins are the extremes and therefore simpler to classify. The difficulty of predicting draws correctly can be seen in many research papers [28, 30, 34]. An important direction to improve the overall prediction accuracy of one's model, is to improve the predictions of matches ending in a draw [30].

As we have seen, football match prediction using machine learning has been studied in multiple research papers during the recent years. However, many of them have not extensively compared different methods with each other, but instead focused on a specific method or algorithm. Many research papers also focused on a specific type of football related data for the predictions, instead of comparing the performance of, for example, team attributes and match statistics. To contribute to the research on how to find the best frameworks and methods for football predictions using machine learning, I decided to make my own implementation where I compare multiple methods and data types.

4. Implementation

In the same pace as artificial intelligence and machine learning are developing, so does also football predictions. The amount of available statistical data of matches and players in the game are increasing all the time. Every move and touch on the ball that the players do can be tracked on the field. However, football is a difficult sport to predict since luck and surprises are inherent factors. As we have seen in the previous chapter, machine learning models can still achieve high prediction accuracy on football matches. This chapter describes the own implementation that I have done on football predictions, to contribute to the research on how to find optimal frameworks and methods for football predictions using machine learning.

4.1 Overview

This implementation includes the whole process of machine learning, starting with data collection, followed by data preparation, model development, and model evaluation. The steps in the implementation process also largely follow the proposed framework for sport result predictions made by Bunker and Thabtah [23], in which the main steps are domain understanding, data understanding, data preparation, modelling, evaluation, and deployment. The goal of the implementation is to test and compare different approaches for football predictions using machine learning and to achieve a prediction accuracy that is higher than the average prediction accuracies of previous research in football predictions. According to Horvat [26], the average prediction accuracy for football predictions ranges from 55% to 70%. A prediction accuracy of above 60% was set up as an initial goal for this implementation.

This implementation predicts the overall result of football matches in a particular season of a particular football league of high professional standard. The overall results signify whether a football match ends in a win for the home team, a draw, or a win for the away team. The exact numerical result of the matches, that is the number of goals scored by each team, is not predicted. Since the overall result can

have three different outcomes (or classes), the machine learning problem of this implementation is a multiclass classification problem.

The data used in this implementation were collected from free publicly available sources online. The data used in the implementation consist of match statistics, team related attributes and player related attributes. The different types of data are first tested separately and then merged to test if this will improve the model performances. Multiple machine learning algorithms are used in the implementation, not only to try to achieve the highest possible prediction accuracy, but also to compare how the algorithms perform in football prediction problems overall. Since this implementation is a multiclass classification problem, the machine learning algorithms chosen are algorithms that support multiclass classification. All the different data types are tested on all the algorithms, to find the best possible combination of data set and algorithm. Although this requires the creation of many different models, the work is still reasonable since one can use the same data sets in all the algorithms with just small adjustments. Moreover, the basic structure of the machine learning algorithms can be the same regardless of the input data set used.

The predictions are done on the whole season at once. In models where the match statistics are used, the data must be kept as temporal data. In such cases, the test and training data are split so that the test data follow the training data in chronological order. Then the predictions are done match by match, although the goal of the predictions is still to achieve high prediction accuracy for the complete season. The implementation is not developed for the purpose of predicting single matches.

The programming languages used in the implementation are Python and SQL. Python is considered as one of the best programming languages for machine learning. SQL is used to query data from an SQLite database. Data preparation is made in the code editor Visual Studio Code. The machine learning model development is made with Python in Jupyter Notebook. Jupyter Notebook is an optimal tool for machine learning, especially since the code can be divided into cells so that loading data multiple times is not needed when experimenting. It is also a suitable tool for making visualisations.

4.2 Data Collection

Data collection is the first practical step in machine learning. However, before the data sets for the machine learning models can be created, one must know what data to use. Knowing what impacts the outcome of the sport in real-life is needed to achieve a successful sport prediction model. To know the main characteristics of the sport that will impact the machine learning problem, is called domain understanding [23]. In football the results depend on the performance of individual players, the performance of the teams overall, and external factors like weather. The data used in this implementation include player attributes, team attributes, and match statistics. Some combination of these three categories of football data have usually been used in previous football match prediction research. When testing all three of them, as well as combinations of them, one will understand how they impact the results of football matches and which combinations are the best for predicting the results. Since the predictions were supposed to be made on a whole season at a time, external factors like weather data were not considered in this implementation. The weather on a particular match day obviously cannot be known beforehand.

The quantity and quality of football data available online vary much from source to source. Both the quantity and quality of the data sets used in machine learning models influence the results. Match statistics of high standard leagues can effortlessly be found online, not only live scores and results of the current season, but also statistics from years back. The statistics can be found in data source types like CSV files and databases. The most effortless way to collect data for machine learning developers, is of course to directly find these kinds of ready data sets that suits their problem. However, if there are no suitable data sets for one's problem, online data can be extracted. The match statistics can be extracted from websites and API:s with live scores and results from many previous seasons. Most data sources about football match statistics include basic statistics like goals, possession, corners, cards, and fouls. Data sources with more in-depth match statistics like player performances during the game can be more difficult to find, especially ones that are free and publicly available. Some well-known websites with football statistics are Sofascore.com, Flashscore.com and Transfermarkt.com.

The data sets used in this implementation are a mix of already existing data sources and data collected from websites. The base data set used in the implementation is a data set named ‘European Soccer Database’ published by Hugo Mathien on Kaggle [36]. Kaggle is a community of data scientists and machine learning developers, where data sets are published for others to find and use in, for example, machine learning development. The ‘European Soccer Database’ consists of football data from eleven of the greatest football leagues in Europe. The database is published on Kaggle as an SQLite database. The football data in the database consists of match data, team attributes, and player attributes from season 2008 to 2016. Overall, it sums up to around 25 000 matches, 11 000 players, and 300 teams. The match statistics include statistics like goals, shots, fouls, cards, and possession. All the players who played in a match are listed in the match data as keys, which can be used to join the players with their player attributes. Besides that, the match data also includes betting odds from several betting sites and metadata like date, teams, season, and stage of the match. [36]

The team- and player attributes in the database originate from the football video game EA Sports FIFA. It is a video games series developed by the video game company Electronic Arts, labelled as EA Sports. They release a new game for every season, which have made it the best-selling sports video game franchise in the world. Every new game is named so that it reflects on which year the season will end, for example the season 2015-2016 is covered in the game called ‘FIFA16’. EA Sports FIFA is based on authentic real world football leagues and players, thanks to its partnership with the international football association FIFA. The player attributes in the game are based on real life performances and are updated many times during the season [37]. The player attributes and ratings are on a scale from 1-99, where 99 is the best rating. The team attributes are based on the players in the team and overall team performance. Although the team- and player attributes in EA Sports FIFA are regarded as one of the best ratings of footballers in the world, they are still subjective and cannot be totally accurate. For example, players in high standard leagues often have higher ratings automatically. Even though the team and player attributes would be totally accurate according to the real world, player performances will still fluctuate from match to match, and therefore they cannot alone give perfect football match predictions. Virtual football data from video

games have been proven to be able to achieve accurate football match predictions [31] and were therefore chosen to be tested and compared with other types of data in this implementation.

As described in the overview of the implementation, one season of a particular league had to be chosen. For the implementation the German Bundesliga was chosen, which is the highest football league division in Germany and one of the best leagues in the world. Any of the other leagues in the 'European Soccer Database' would have worked equally well, but Bundesliga was chosen since it has not been used as much as other top European leagues in previous football prediction research papers, and because it is an interesting league being one of the greatest leagues with close connections to the Nordic countries. The season 2015-2016 was chosen for the implementation since it is the most recent season available in the 'European Soccer Database'. The German Bundesliga has 18 teams and during one season there are totally 34 stages and 306 matches (one home match and one away match against every other team in the league).

The team- and player attributes in the 'European Soccer Database' were sourced by Mathien from the website Sofifa.com, which is a database for statistics in the EA Sports FIFA game [36]. The team attributes in the database include team ratings of their build-up play, chance creation, and defence types. However, these attributes are quite detailed and for some reason the overall team ratings were missing in the database. That is why some additional team attributes were collected, which then were added to the data set. These team attributes were the overall rating of the teams, and their defence, midfield and attack ratings. These team attributes were sourced from Fifaindex.com [38], which is a database for statistics in EA Sports FIFA like Sofifa.com. Only the team attributes for the selected league and season (German Bundesliga, 2015-2016) were collected and added to the data set.

The match statistics of the German Bundesliga season 2015-2016 in the 'European Soccer Database' lacked clean values for some attributes. For example, the goals were embedded in a piece of code and it would have required much cleaning to extract the relevant values. However, the most significant limitation of the data was that there was no attribute for directly determining the overall result of the match, that is 'home win', 'draw' or 'away win'. This could of course be calculated by

looking at the scored goals for the teams in each match, but that would have required some extra coding. Instead, a second data source including match statistics from the chosen league and season was used. The data source is a CSV file sourced from DataHub.io, which is a data hub for open data (free to use for everyone) [39]. This data file includes similar match statistics and metadata as the 'European Soccer Database', but with cleaner and more complete data for all attributes and with the important addition of the missing 'overall result' attribute. This attribute is named 'FTR', shortened from 'full time result'. The three possible values are 'H' for 'home win', 'D' for 'draw' and 'A' for 'away win'. These will be the output classes of the football prediction classification.

A diagram of the data used in the implementation, is visualised in Figure 6. The sources of the team attributes are the 'European Soccer Database' and the website Fifaindex.com. The 'European Soccer Database' is the only source of the player attributes, while the only source of the match statistics used in the implementation is the data set from DataHub.io. All three different data categories are joined with match data to form the final data sets. The source of the match data is the same data set from DataHub.io as the one used for match statistics. The match data consists of the date, home team, away team and overall result of the match. The matches are sorted in chronological order.

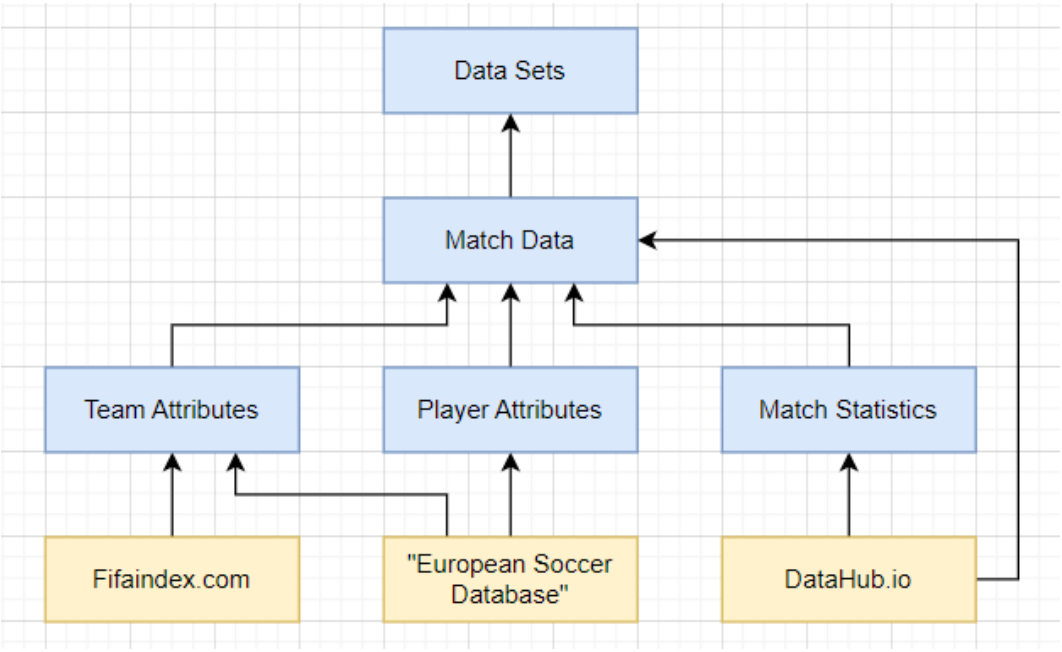


Figure 6. *Diagram of the data sources used for the data sets in the implementation.*

4.3 Data Preparation

In machine learning, the next step after data collection is data preparation. It consists of data cleaning, feature selection and feature extraction [3]. In this step one need to know what input data to use and what output data one wants. To know what data to use, the data sets one work with need to be known. In this implementation the goal was to use three different main data sets: one data set with team attributes, one with player attributes, and one with match statistics. The data preparation was made in Visual Studio Code using SQL and Python. To be able to explore and query the SQLite database European Soccer Database in Visual Studio Code, the SQLite extension was installed.

The base of every data set contains the same attributes, the match data. The date, the teams, and the overall result of every match, were taken from the data set sourced from DataHub.io. The overall result is the attribute called FTR, and it serves as the output data in the football prediction models. This attribute is the most important in the data sets, since the accuracy of the classification predictions is determined on how correctly this attribute can be predicted. The teams in each match consist of the name of the home team and the away team. However, the team names and the FTR attribute are represented as text data. The text data must be converted into mathematical representations using feature encoding. The three possible classes of the FTR attribute were dummy encoded to the numbers 0, 1, and 2. The team names were also encoded using dummy encoding, where every class is represented as a number instead of the text. There are 18 teams in the German Bundesliga, and therefore 18 possible classes for the home- and away team attribute.

The team attributes table in the main database, the European Soccer Database, includes team ratings on build-up play, chance creation, and defence types. The table includes a total of 25 attributes, of which three are different kinds of ids and one is the date when the team attributes were updated. However, of the remaining 21 attributes, only nine include numerical values. The rest of the attributes are called “classes of the attributes” and consists only of simplifications of the numerical values. The numerical values have been grouped according to some specific ranges

into a few classes in text form. For example, the values of the build-up play speed of the teams have been grouped into either slow, balanced, or fast. It is unnecessary to use both the numerical and classificational approach. Since the numerical values are more precise, only those nine attributes were included in the final team attributes data set.

To make an efficient data set for machine learning, only data that is relevant to one's problem must be used. Since the European Soccer Database includes around 300 teams, it was clear that only the 18 relevant teams from the German Bundesliga season 2015-2016, had to be queried from the team attributes table. The team attributes table does not have a country-id, which meant the relevant teams had to be queried using the team-ids instead. The team-ids for the 18 teams could be found by querying one stage of the 2015-2016 German Bundesliga season from the match table, since every team will only play one match in each stage and therefore there will only be 18 team-ids visible without duplicates or missing values.

All the team attributes for the relevant teams were queried together with their team names. The team names were needed to be able to join the team attributes with the match data sourced from DataHub.io. The match data does not use the same team-ids as the European Soccer Database does, and therefore the data must be joined using the team names instead. However, the team names are also different in the two data sources. Football teams usually have a long official name, but it is common to use a shortened version. This can obviously lead to different versions of the team names in different places, especially when the team names are not in English. In this case the European Soccer Database uses a longer version of the German football teams, than the match data from DataHub.io uses. For example, the European Soccer Database uses the team names FC Schalke 04 and Borussia Mönchengladbach, while the match data uses Schalke 04 and M'gladbach. To be able to join the two data sources using the team names, the team names used in the match data was mapped to the team-ids in the team attributes query. The final query result was exported to a CSV file, which can effortlessly be done in Visual Studio Code after an SQLite database have been queried using the SQLite extension.

In the FIFA video game, the main ratings of teams are the overall rating and the overall ratings in different fields: defence, midfield, and attack. The overall ratings

are calculated based on more detailed team attributes, which also are found in the European Soccer database. However, for some reason the overall team ratings were missing in the database, and therefore the overall ratings were added to the team attribute data set manually from a website. In situations where one need to retrieve data from websites, a normal approach is data scraping. In data scraping data can be imported from websites using computer programs, to be saved in files on one's own computer. In this case, the additional data needed included just four attributes for 18 teams. Therefore, this data was added to the team attributes data set manually. It would have been more work to create a data scraping program to retrieve the specific data automatically, than simply copying the 72 values manually.

The next task was to join the team attributes with the match data. This was done using Python in Visual Studio Code. The match data from DataHub.io was first joined with only the team attributes from the European Soccer Database. However, when figuring out that the overall ratings were missing from the data set, the match data were joined with the team attributes including the overall ratings instead. The match data was first extracted from the match statistics from DataHub.io. The original data source is a CSV file, and was read to a Pandas data frame. Pandas is a common Python library used for data analysis. The CSV file including the team attributes and overall ratings, was also read to a Pandas data frame. The two data frames were joined on the team names. The final data set was to include all matches of the selected season, with the team attributes of the teams facing each other in every match. Therefore, the match data was the base data frame, and joined with a left join with the team attributes two times, ones for the home team and ones for the away team. An 'h' and an underscore were added in front of the attribute names of the home team, and an 'a' and an underscore were added in front of the attribute names of the away team, to be able to distinguish between the attributes of each team.

The player attributes table in the main database, the European Soccer Database, includes player ratings on attributes like shooting, passing, physicality, pace, movement, defence, and goalkeeping. It is by far the largest table in the database, since it includes attributes of over 10 000 players for each season and version. The player ratings change from season to season, but can also be updated during a

season. The player attributes table includes a total of 42 attributes, of which three are different kinds of ids and one is the date when the player attributes were updated. Of the remaining 38 attributes, three include text data: a player's preferred foot, attacking rate and defence rate. These attributes were also removed from the data set. The 35 remaining player attributes all include numerical ratings between 1 and 99, which makes feature scaling unnecessary in the modelling phase.

To retrieve the player attributes for every team in the 2015-2016 season of the German Bundesliga, the match table must be used like for the team attributes. However, it does not make sense to add all the players of a team to every match, since there are only eleven players in the start of a match and a few players may come in later in the match as substitute players. The players sitting at the bench will not impact the result of the match, and therefore their attributes are unnecessary to use. Luckily, the match table includes the starting eleven of every team for every match. The substitute players are not included in the table and even though they would be, the use of substitute players would be very complex, since then one would have to add weights to the data according to how long they have been on the pitch to make the predictions realistic. In the match table the players are represented as player-ids, which can be used to join it with the player attributes table. The 'match' table also includes coordinates of the players on the field, which makes it possible to determine what formation a team uses and where each player is positioned. However, the players in the match table are ordered with the goalkeeper first, then the defenders, the midfielders and lastly the attackers, so the position of players can roughly be determined without using the coordinates.

The joining of the player attributes and the match data was more difficult than joining the team attributes with the match data. For every match, all the player attributes of the 22 players involved in the match are to be used. This means that the player attribute table had to be joined with the match data 22 times. To be able to do the joins more effortlessly, the player attributes were first joined with the match data in the European Soccer Database, and after that joined with the FTR column of the match data from DataHub.io. The joins could therefore be done using SQL only within the European Soccer Database. The select of the query consisted of the date, the home team-id and the away team-id from the match table, and all

the player attributes of the 22 players from the player attributes table. The match table was joined with the player attributes table using left-joins on the player-ids. The match data had to be filtered on the 2015-2016 season of the German Bundesliga and the player attributes had to be filtered on one specific version. The version of the player attributes was chosen as the first official release of the FIFA16 player ratings. However, some players move clubs within a season and some player ratings may have been incomplete during the first release, so the query resulted in 246 matches instead of the original 306 matches. In addition, 27 matches included player attributes with null values, which had to be removed, and therefore reduced the final match total to only 219 matches. The final query result was exported to a CSV file.

Since the teams in each match were represented as ids, the team names in the match data from DataHub.io had to be converted to the team-ids used in the European Soccer Database. This was done using Python in Visual Studio Code. A conversion table was created, that included all the 18 teams with their team names and their respective team-ids in the European Soccer Database. The table was left joined with the match data two times, ones for the home team and ones for the away team. When the team names were integrated in the data set, it could be joined with the 'FTR' column of the match data from DataHub.io. The join was done using a left join on the team names of each match.

The match statistics data used in this implementation was sourced from DataHub.io. The data source is a CSV file with match statistics of the German Bundesliga season 2015-2016. It includes information about the matches, such as date, participating teams, statistics, and betting odds. The statistics of the two teams are separated in the names of the attributes, so that the statistics of the home team are attributes including an 'H', and statistics of the away team are attributes including an 'A'. The match statistic attributes are the number of goals, shots, shots on target, fouls, corners, yellow cards, and red cards, that have occurred during the match. Furthermore, the overall result of the match (the FTR attribute that is also used in the match data combined with the team attributes and player attributes) and the goals scored are divided into both the fulltime- and halftime results. The betting odds have been gathered from over ten different betting sites, and consist of the

odds for home win, draw or away win. However, these statistics are not used in the implementation, since the most accurate betting odds are finalised exactly before the start of a match and therefore cannot be gathered for all matches of a whole season before the season has even started.

Although the match statistics data set from DataHub.io includes clean data and most of the attributes needed to make predictions, there was a major obstacle. The match statistics cannot be used directly to predict the results of the football matches, since the statistics obviously have been gathered after the matches have ended. It does not make sense to make predictions on matches that have already occurred, instead previous matches must be used for the predictions. The statistics of every match had to be converted into averages of previous matches. This makes the predictions more logical, but in a real-world problem the result of upcoming matches cannot be predicted using match statistics, except for the next upcoming match of a team. However, in this implementation the whole season was predicted at a time, but the chronological order of the matches was kept so that the results of matches later in time are not "leaked" in the learning process of the machine learning models.

The data preparation of the match statistics was done with Python in Visual Studio Code using the Pandas library. The first step was to read the data from the original data source file into a Pandas data frame. Only relevant attributes were read into the data frame, the betting odds were not considered. The type of every numerical attribute was changed to float instead of integer, so that they would be simpler to handle when calculating averages. The columns representing goals scored for each team in a match were copied, so that there would be one column representing goals scored and one column representing goals conceded for each team in a match.

In many football league tables online, the results of the last few matches for each team are visible. This gives a hint of the current form of a team. A team that has won all their recent matches is in form, but a team that has not won any of their recent matches is in poor form. The FTR attribute was used in the implementation to utilize the form of the teams. The FTR column was copied so that the form of each team can be seen in every match. The values were converted to numerical values to be able to calculate the averages. For each team, a win was set to two, a

draw to one, and a loss to zero. Thus, teams with a higher average exhibit superior form.

Since the collected data only includes statistics from one season, the first matches of the season will not have any data to make the averages on. Therefore, the statistics of the last match of the 2014-2015 season of every team were added to the data frame manually. The next step was to calculate the averages of the match statistics. For each match, the statistics of the specific match had to be changed into the averages of the past matches. For football match predictions, it has been found that the best number of matches to be used for the averaging process, is the past 20 matches [27]. This was also chosen as the maximum number of past matches considered in the averaging process of this implementation. In the 19 first stages of the season, only the number of available past matches were considered.

The averages were calculated using a temporary table to keep track of the statistics of past matches. A for-loop was used to iterate over the matches. In the beginning of every iteration, the match statistics were saved to temporary variables together with the stage number. The stage number was the same as the iteration order, which was kept track of using a variable. The statistics were to be saved in the temporary table, however, first the averages had to be calculated. Otherwise, the statistics of the current match would also be considered in the averages and thus leaking statistics that only should be seen after the match has ended.

Next the average of every attribute was calculated using the past match statistics in the temporary table. The temporary table was filtered on the team names and a sum of every attribute was calculated. The sums were then divided by the number of past matches used, to receive the final averages. The averages were rounded to three decimal places and then added to match statistics replacing the match statistics for the current match. Next the match statistics of the current match, which were saved to temporary variables, were added to the temporary table. The statistics of the two teams in every match were separated in the temporary table so that it would be simpler to find past statistics for a specific team. Lastly the stage number was increased by one, and if it had reached 20 the match statistics of the teams in the current match that had the lowest stage number in the temporary table, were removed from the table so that the maximum number of past matches would not

exceed 20. When all the matches had been iterated, the final data frame was saved to a CSV file.

The final data sets are shown in Figure 7. The player attributes data set consists of 774 columns, the team attributes data set consists of 30 columns, and the match statistics data set consists of 22 columns.

Player Attributes Data Set

Name	Description
Date	date
FTR	result
HomeTeam, AwayTeam	team name
overall_rating	22 players
potential	22 players
crossing	22 players
finishing	22 players
heading_accuracy	22 players
short_passing	22 players
volleys	22 players
dribbling	22 players
curve	22 players
free_kick_accuracy	22 players
long_passing	22 players
ball_control	22 players
acceleration	22 players
sprint_speed	22 players
agility	22 players
reactions	22 players
balance	22 players
shot_power	22 players
jumping	22 players
stamina	22 players
strength	22 players
long_shots	22 players
aggression	22 players
interceptions	22 players
positioning	22 players
vision	22 players
penalties	22 players
marking	22 players
standing_tackle	22 players
sliding_tackle	22 players
gk_diving	22 players
gk_handling	22 players
gk_kicking	22 players
gk_positioning	22 players
gk_reflexes	22 players

Team Attributes Data Set

Name	Description
Date	date
FTR	result
HomeTeam, AwayTeam	team name
buildUpPlaySpeed	home & away
buildUpPlayDribbling	home & away
buildUpPlayPassing	home & away
chanceCreationPassing	home & away
chanceCreationCrossing	home & away
chanceCreationShooting	home & away
defencePressure	home & away
defenceAggression	home & away
defenceTeamWidth	home & away
attack	home & away
midfield	home & away
defence	home & away
overall	home & away

Match Statistics Data Set

Name	Description
Date	date
FTR	result
HomeTeam, AwayTeam	team name
FTHG, FTAG	goals scored
FTHCG, FTACG	goals conceded
HS, AS	shots
HST, AST	shots on target
HF, AF	fouls
HC, AC	corners
HY, AY	yellow cards
HR, AR	red cards
HForm, Aform	form

Figure 7. *The attributes of the three final data sets.*

5. Prediction Models

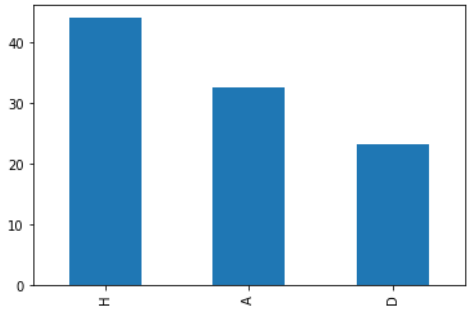
When the data collection and data preparation are done, data sets that can be used in the machine learning models are possessed. This chapter describes the prediction models created in the implementation and the evaluation of their results.

5.1 Model Selections

The machine learning algorithms used in this implementation were chosen based on their type, popularity, and use in previous research in football predictions. Since multiclass classification is needed, the chosen algorithms must be able to support it. The Naïve Bayes algorithm was chosen because it is one of the most known machine learning algorithms and it has been used in previous research in football predictions. Logistic regression and random forest were chosen since their approach is very different, and therefore more possible types of algorithms for making football predictions could be tested.

Every machine learning algorithm was implemented using Scikit-learn, which is a Python library specialised in machine learning operations. The target in the models is the FTR attribute, which consists of match results as categorical values. Therefore, the CategoricalNB method was used in the Naïve Bayes implementation. The random forest algorithm was implemented using the RandomForestClassifier method, with 100 decision trees as the 'n_estimators' parameter. The logistic regression algorithm was implemented using the LogisticRegression method. So that the algorithm could handle multiclass classification, the 'multi_class' parameter was set to 'ovr' (One-vs-rest). To balance the number of predictions per class, the classes were given weights in some logistic regression models. The weights were based on the average distribution of match results in football, which is around 45% for home wins, 30% for away wins, and 25% for draws [29]. The frequency distribution of the results in the 2015-2016 German Bundesliga season, is visualised in Figure 8. They are similar to the average distribution. However, the actual distribution was not used, since it should not be known before the matches have been played. The weights were switched around so that the weight for the

draws was the greatest. It was calculated with an equation where the proportions between the values was considered. The final weights were 42% for draws, 35% for away wins, and 23% for home wins.



H 44.117647
A 32.679739
D 23.202614

Figure 8. *The frequency distribution of the match results (the FTR attribute).*

5.2 Team Attributes

The team attributes data sets were used with each of the three selected machine learning algorithms. By combining the algorithms with different subsets of the original data sets, multiple models were created. A subset in this context is a part of the original data set, for example, to test the performance using just a few of the original features. New subsets were tested with all the three algorithms to receive an overview of their performance, and not only the result of one single model.

The data set was split randomly into training data and test data, so that 70% was training data and 30% was test data. Before the overall ratings of the teams were added to the team attributes data set, the data set was tested in the models. The original team attributes data set included nine team attributes and the team name of both the home team and the away team. These 20 attributes were used in the first models. The prediction accuracy of the Naïve Bayes model was 51.1%. The logistic regression model achieved a prediction accuracy of 53.3% and the random forest model a prediction accuracy of only 43.5%. The achieved accuracies were decent compared to previous research, bearing in mind that these were the first tests. To validate the models more accurately, the k-fold cross-validation technique was used. Using 10-fold cross-validation, the results were similar. Logistic regression achieved an average accuracy of 52%, Naïve Bayes an average of 51%, and random forest an average of 45%.

The most interesting aspect of the first tests, was to receive an estimate of the feature importance. The feature importance can be used for feature selection in the model testing phase. The feature importance using the random forest algorithm with the original team attributes data set, is visualised in Figure 9.

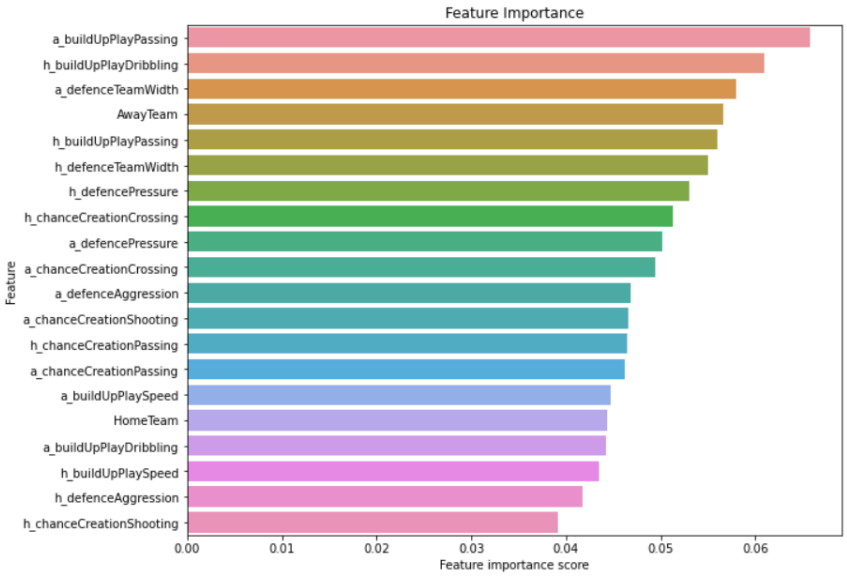


Figure 9. *The feature importance of the original team attributes data set using the random forest algorithm.*

According to the feature importance in Figure 9, the dribbling and passing in the build-up play, the defence pressure, and the chance creation crossing, were the most important attributes in this model. Another interesting finding is that the away team seems to be more important than the home team in predicting the outcome of a match. The next tests were done using a subset of the original team attributes, including the four most important attributes from the first test and the team name of each team. The models using these ten attributes performed better than the first models. All the three new models improved by a few percentages, the best one still being the logistic regression model, now with an accuracy of 59.4%.

When the team names were removed from the subset, the models performed more balanced. The prediction accuracy of the logistic regression model decreased to 57.6%, but the accuracy of the random forest model increased to 54.3% and the Naïve Bayes model achieved an accuracy of 52.2%. The confusion matrices of the three models using the subset with eight attributes, are visualised in Figure 10. We

can see that the best precision occurred for the home win class, but the highest recall occurred for the away win class. Overall, the f1-score was highest for home wins and worst for draws. The most common faulty predictions by the models, were that a match would end up in an away win instead of a draw or home win.

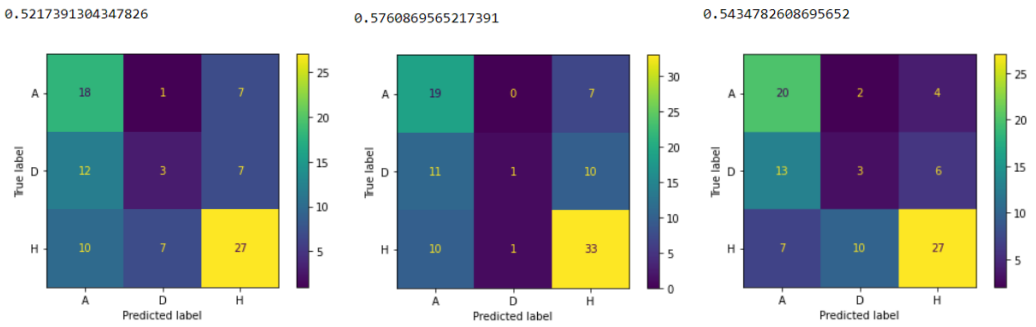


Figure 10. *Prediction accuracy and confusion matrix of each model. From the left: Naïve Bayes, logistic regression and random forest.*

The second version of the team attributes data set also included the overall ratings of the teams. When the four new attributes were added to the data set, the number of attributes per team was 14, which brings the total number of attributes in the data set to 28. The feature importance of this data set, according to the random forest model, is visualised in Figure 11.

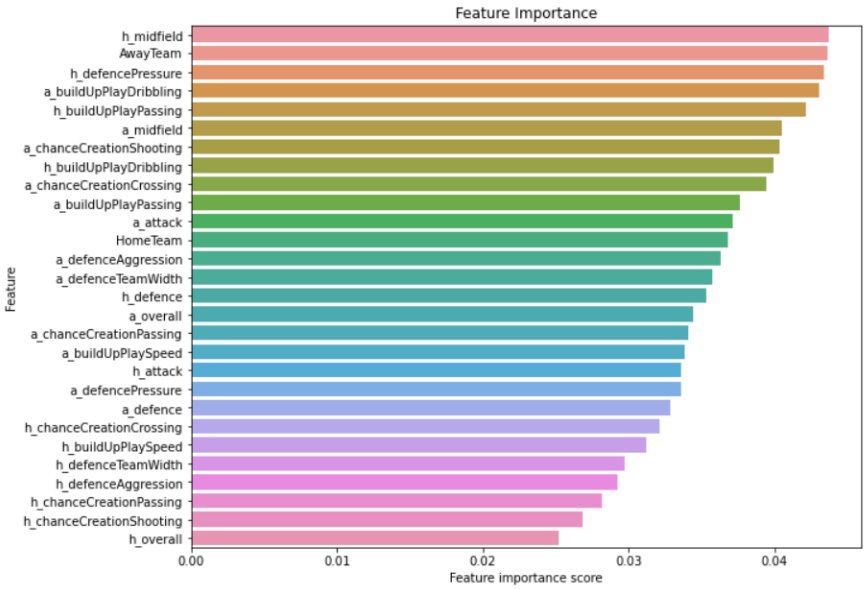
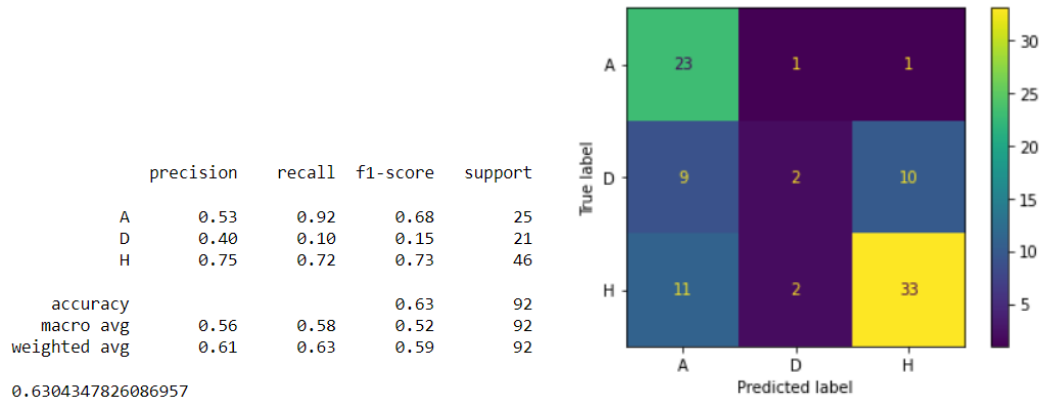


Figure 11. *The feature importance of the team attributes data set including overall ratings using the random forest algorithm.*

Using the whole data set, the prediction accuracy of the models was 52.2% for Naïve Bayes, 54.3% for logistic regression and 51.1% for random forest. According to the 10-fold cross-validation scores, logistic regression achieved an average accuracy of 53%, Naïve Bayes an average of 48%, and random forest an average of 46%. The same four team attributes as in the original data set, can be found among the top attributes for feature importance of this data set. We can also see that the midfield rating of the teams is the most important overall field rating.

Since the same four team attributes were again among the top attributes for feature importance using the additional overall ratings, they were again chosen to be used in subsets for more tests. The overall rating of the teams was removed and the midfield rating was added to the subset. According to the feature importance the defence rating and the attack rating have quite similar importance, however, in the most tests the models performed better using the defence rating instead of the attack rating. The logistic regression model achieved a prediction accuracy of 63.0% using the four original attributes, the midfield rating and the defence rating. The result of the model is visualised in Figure 12. However, we can see that barely any draws were predicted. When running the same algorithm and data with weights, the accuracy decreases a few percentages, but the recall of the draws was improved. For the Naïve Bayes and random forest models, the prediction accuracy did not change much regardless of combination of overall ratings together with the four original attributes, but according to the 10-fold cross-validation scores all three models improved by a few percentages using the six mentioned attributes.



	precision	recall	f1-score	support
A	0.53	0.92	0.68	25
D	0.40	0.10	0.15	21
H	0.75	0.72	0.73	46
accuracy			0.63	92
macro avg	0.56	0.58	0.52	92
weighted avg	0.61	0.63	0.59	92

0.6304347826086957

Figure 12. *The best single prediction result using team attributes data was achieved using logistic regression with a data set of six attributes.*

Although the data set that achieved the best single result among the team attributes data sets, included some overall ratings, the addition of the overall ratings did not improve the predictions significantly. In most of the tests, the logistic regression algorithm performed the best according to the prediction accuracy. However, it predicted only a few draws, and the f1-score of the draw class was very low. The Naïve Bayes algorithm performed slightly better than the random forest algorithm using team attributes data.

5.3 Player Attributes

The player attributes data sets were used with each of the three selected machine learning algorithms. By combining the algorithms with different subsets of the original data sets, multiple models were created. The data set was split randomly into training data and test data, so that 70% was training data and 30% was test data. The original player attributes data set included 35 attributes per player in every match and the team name of both the home team and the away team, which brings the total number of attributes to 772. These attributes were used in the first models. The prediction accuracy of the Naïve Bayes model was only 42.4%, however, it was done using the method GaussianNB instead of CategoricalNB. The CategoricalNB method did not function properly due to different set of values in the features. The logistic regression model achieved a prediction accuracy of 50.0% and the random forest model a prediction accuracy of 56.1%. The logistic regression model was even able to predict draws without weights, which the original data sets using team attributes and match statistics were not able to do. Using 10-fold cross-validation, Naïve Bayes achieved an average accuracy of 50%, logistic regression an average of 42%, and random forest an average of 53%.

Since the number of attributes in the first models was so high, the feature importance of an attribute was calculated by taking the mean of all players. This resulted in 35 attribute groups plus the team name attributes. The estimates of their feature importance are visualised in Figure 13. The away team attribute is clearly standing out of the rest attributes. All the goalkeeper related attributes had low feature importance, which was to be expected because the goalkeeper ratings of all players in a match were included in the data set. However, the attributes of the

players are ordered in the data set according to their position on the field during matches. By removing every goalkeeper related attribute, except those for the first and the twelfth player, only the attributes of the goalkeepers were left. Every normal player attribute of the goalkeepers was also removed. These changes in the data set improved the results of the Naïve Bayes and logistic regression models with a few percentages, but the random forest model performed slightly worse than in the first test.

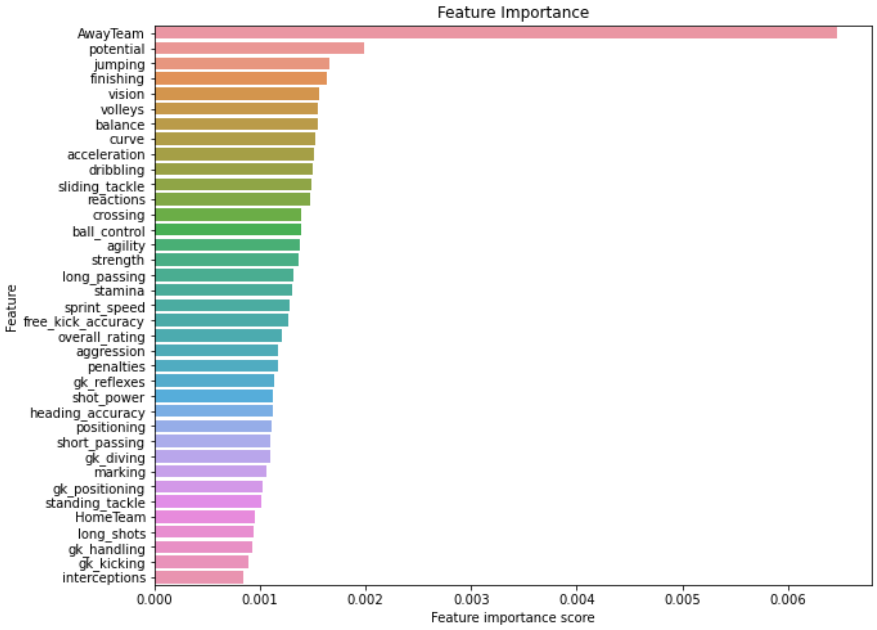


Figure 13. *The feature importance of the original team attributes data set using the random forest algorithm.*

For the next models, some attributes with a low feature importance were removed to test if it would improve the model predictions. When only a few attributes were removed, the predictions worsened. However, when half of the attributes were removed, the Naïve Bayes model achieved a prediction accuracy of 53.0% and the random forest model a prediction accuracy of 56.1%. This time the logistic regression model performed worst, achieving a prediction accuracy of only 39.4%. The confusion matrices of the three models are visualised in Figure 14. Using 10-fold cross-validation, the results were similar. Random forest achieved an average accuracy of 54%, Naïve Bayes an average of 51%, and logistic regression an average of 43%.

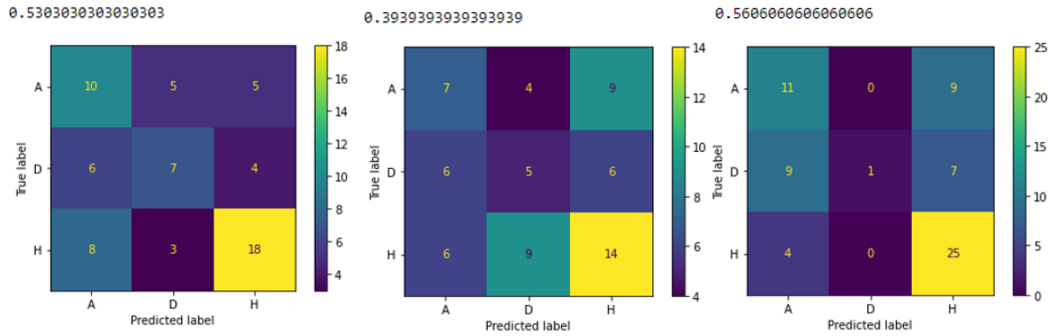


Figure 14. *Prediction accuracy and confusion matrix of each model. From the left: Naïve Bayes, logistic regression and random forest.*

The feature importance of the attributes used in this data set was balanced with small differences, which makes it difficult to make further feature selections. The feature importance is visualised in Figure 15. The 19 attributes used per player were on average more important than the 18 removed attributes.

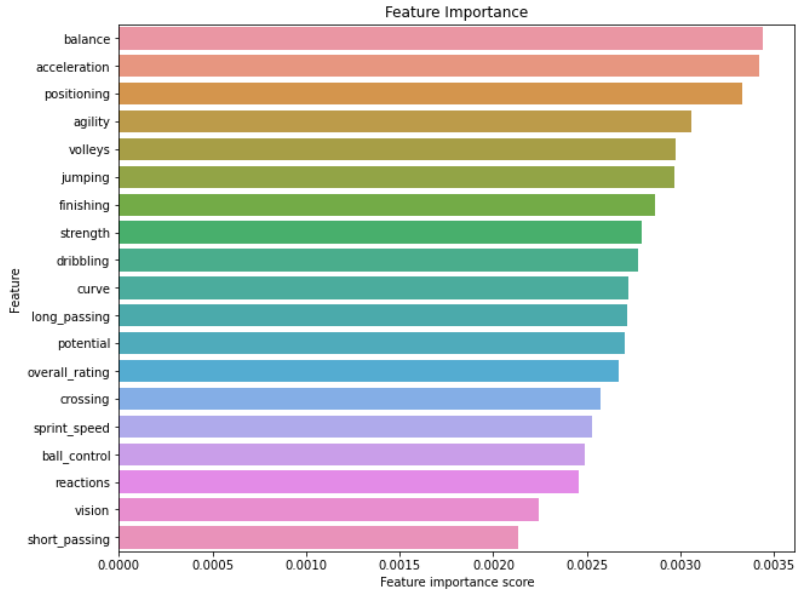


Figure 15. *The feature importance of the data set including half of the original player attributes, using the random forest algorithm.*

Since the number of attributes in the original data set was so high, the number of possible combinations of them are also very high. However, all further tests with player attributes data sets resulted in worse prediction accuracies than in the mentioned tests. In some models only one attribute of each player was tested. The overall rating and the potential of a player are examples of attributes that describe

the overall quality of a player more accurately. However, only a few of these attributes achieved a prediction accuracy of over 50% on their own. Since the playing position of the player attributes are known, we can look at the feature importance of the overall rating attribute for each player to compare their importance. It seems like the order from most to least important are attackers, midfielders, goalkeepers, and defenders. A higher player position also has a higher number, for example, the home attackers of the home team are number 8, 9, and 10 and the attackers of the away team are number 19, 20, and 21. Their feature importance is visualised in Figure 16.

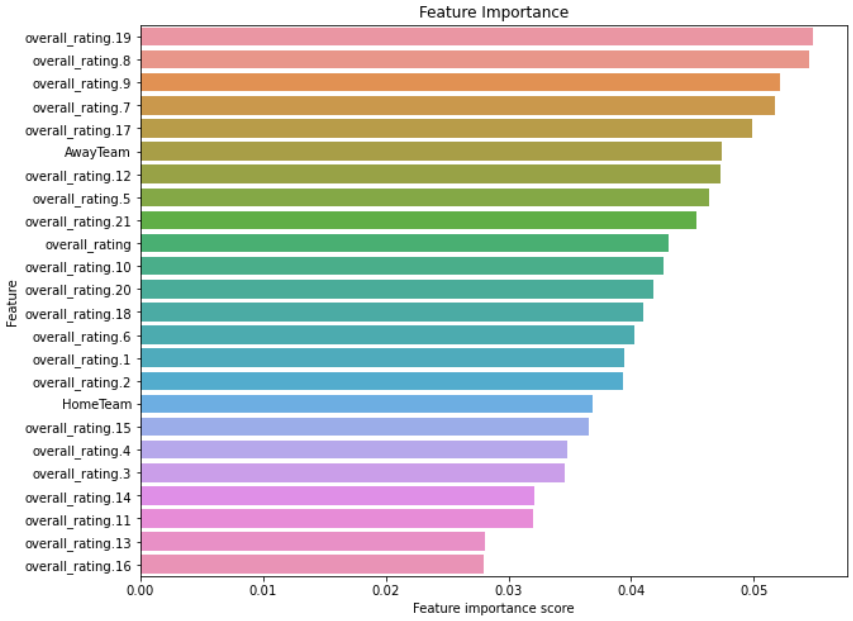


Figure 16. *The feature importance of the players' overall rating.*

5.4 Match Statistics

The match statistics data sets were used with each of the three selected machine learning algorithms. By combining the algorithms with different subsets of the original data sets, multiple models were created. The data set was split into training data and test data chronologically, so that the first 70% of all the matches was training data and the last 30% was test data. The original match statistics data set included nine team attributes and the team name of both the home team and the away team. These 20 attributes were used in the first models. The prediction accuracy of the Naïve Bayes model was 52.2%. The logistic regression model

achieved a prediction accuracy of 51.1% and the random forest model a prediction accuracy of 55.4%.

The feature importance of the data set could be analysed after the first tests. The feature selection in future models is based on the feature importance. The feature importance using the random forest algorithm with the original match statistics data set, is visualised in Figure 17. We can see that the red card attribute clearly has the least importance. Red cards usually have a significant impact on match results, but red cards from previous matches have a low impact on the results, except when a key player receives suspension from the next matches due to a red card. The form attribute also seems to be of low importance, although it usually indicates which team will win. Surprisingly, the number of fouls made by the home team has especially high feature importance.

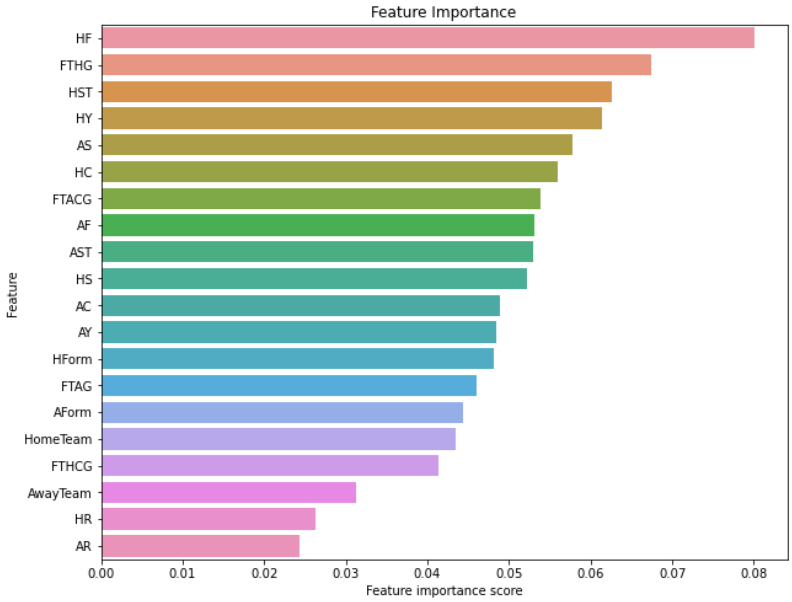


Figure 17. *The feature importance of the original match statistics data set using the random forest algorithm.*

The models performed more balanced when the red card columns were removed, but they did not achieve a higher accuracy than using the whole data set. When other attributes were removed, the models always performed worse. This suggests that the whole original data set was the best data set to use, and that if more match statistics would have been added the models, they may have performed even better. The random forest algorithm achieved the highest prediction accuracy in almost all

the tests. The worst algorithm for the match statistics was logistic regression. Again, it was mostly due to its inability to predict draws without the use of weights. When weights were added to the models, the logistic regression achieved a lower prediction accuracy but a higher f1-score for draws. The results of the three models using the whole match statistics data set, are visualised as confusion matrices in Figure 18. Since the values of the attributes are in different scales, feature scaling was tested on the logistic regression models. However, it did not improve the results.

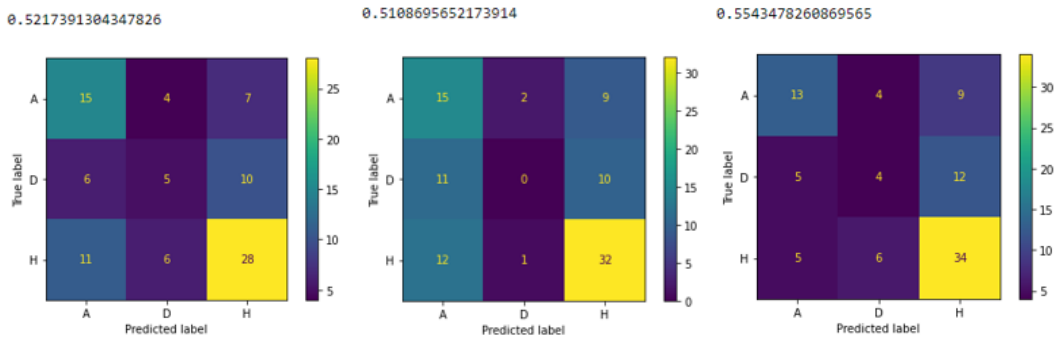


Figure 18. *Prediction accuracy and confusion matrix of each model. From the left: Naïve Bayes, logistic regression and random forest.*

The training accuracies of the Naïve Bayes and logistic regression models were slightly higher than their test accuracies, however, the training accuracies of the random forest models exceeded 90%. This is a sign of overfitting, which in this case may be due to a too small data set. To validate the models more accurately, the k-fold cross-validation technique could have been used. However, since the match statistics data consists of previous match results, cross-validation is not appropriate to use [23, 28]. If temporal data is used in cross-validation, the models would see training data that has happened later in time than the test data, which would not be possible in a real-world scenario. The model that achieved the highest prediction accuracy using match statistics, was the random forest model that used the whole original data set, achieving a prediction accuracy of 55.4%.

Link to the GitHub repository which includes the notebook used for making the predictions: https://github.com/fredriksjoberg17/MasterThesis

6. Discussion

Many different combinations of data sets and algorithms were tested in the implementation. Most of the models achieved a prediction accuracy of around 50%. A few of the best models reached over 60%, but the worst models barely reached 40%. The best model used six team attributes with the logistic regression algorithm, and it achieved a prediction accuracy of 63.0%. It is in the range between 55% and 70%, which is the average prediction accuracy achieved by football predictions [26]. The initial goal of this implementation was to achieve a prediction accuracy of above 60%, which was reached.

The logistic regression algorithm performed the best using the team attributes data sets. The Naïve Bayes algorithm performed slightly better than the random forest algorithm with the same data sets. However, using the player attributes data sets, the results were completely switched. Now random forest performed the best, followed by Naïve Bayes, while logistic regression performed very badly. The order was the same using match statistics data sets, but the differences between the algorithms were smaller. One of the reasons why the performance of a particular algorithm changes when the data set is changed, is the number of features in the data set. The random forest algorithm generally performs well when the number of features is high, but may struggle when the number of features is low. This can also be seen in this implementation, when the random forest algorithm performed best on the player attributes data set, which by far had the most features out of the three original data sets.

The Naïve Bayes algorithm performed most balanced out of the tested algorithms. It did not perform best with any of the original data sets, but some Naïve Bayes models performed the best by a little margin using smaller subsets. It is common for the Naïve Bayes algorithm to perform well generally, but it can still be less accurate compared to other classification algorithms [40]. Naïve Bayes assumes that all the attributes are independent of each other, which was not the case for many attributes used in this implementation [12]. The random forest algorithm performed best using both player attributes and match statistics, but worst using team attributes. The logistic regression algorithm was responsible for the best single

model performance, and was the best algorithm for team attributes overall. However, it was the worst algorithm for both player attributes and match statistics. The average prediction accuracy of all models was around 52% for every tested algorithm. Therefore, it is difficult to determine which algorithm is the best for predicting football matches according to this implementation. Logistic regression using team attributes was the best combination of algorithm and data set, random forest was the best algorithm using match statistics and player attributes, but Naïve Bayes was the most balanced algorithm.

The performance of the three types of data sets is more distinguishable. The team attributes data set performed best on average and produced the best single model. On average the team attributes models achieved a prediction accuracy of around 54%, and its best model achieved an accuracy of 63.0%, The match statistics models achieved an average accuracy of around 53% and its best model achieved an accuracy of 55.4%. The player attributes models achieved an average accuracy of only around 51% and its best model achieved an accuracy of 56.1%. The team attributes data set clearly was the best for football predictions according to this implementation. Although the number of features in the data set was low compared to the player attributes data set, it performed well also after feature selections and removals. Some tests with mixed data types were also done in the implementation. The most important team attributes and player attributes were joined and tested on the algorithms. The highest accuracy was achieved by a logistic regression model with a prediction accuracy of around 52%, but the average was around 50%.

One reason why the match statistics data set did not perform as well as the team attributes data set, may be that the values of some match statistics did not differ from each other enough, and therefore did not make any significant impact in the generalisation of the data. Since the values were averages, many attributes with low values, like scored goals, did not differ from each other as much as for other attributes, like fouls. It can also be one reason why fouls had the greatest feature importance according to Figure 17. However, feature scaling was tested in some logistic regression models, but it did not improve the results. The primary reason why the player attributes data set performed worst, was the size of the data set. The team attributes data set and the match statistics data set both included 306 rows, one

for every match played in a German Bundesliga season. However, the player attributes data set only included 219 rows due to incomplete data or null values. Since the number of features in the original data set included over 700 features, it means that the number of features was larger than the number of observations. This can lead to overfitting, especially for logistic regression. This is one reason why logistic regression performed so badly with player attributes in this implementation.

Although some models achieved a high prediction accuracy of over 60%, all the models shared a common issue. They were unable to accurately predict draws. It can be seen in the confusion matrices in Chapter 5, for example on the result of the best model in Figure 12. We can see that the recall is very low, and the precision is not over 50%. The confusion matrix of a well performing model should show a clear diagonal line of true predictions, which cannot be seen in many models in this implementation. It can be seen in Figure 19, which is a confusion matrix of the training results of a Naïve Bayes model. Weights were used to increase the number of predicted draws for models with a low f1-score of the draws. This improved the f1-score of the draws, but the overall prediction accuracy usually decreased. The difficulty of predicting draws correctly have been noticed in many research papers [28, 30, 34]. Draws are difficult to predict because least number of matches end up in a draw, and because the class in the middle is the most difficult to predict [30].

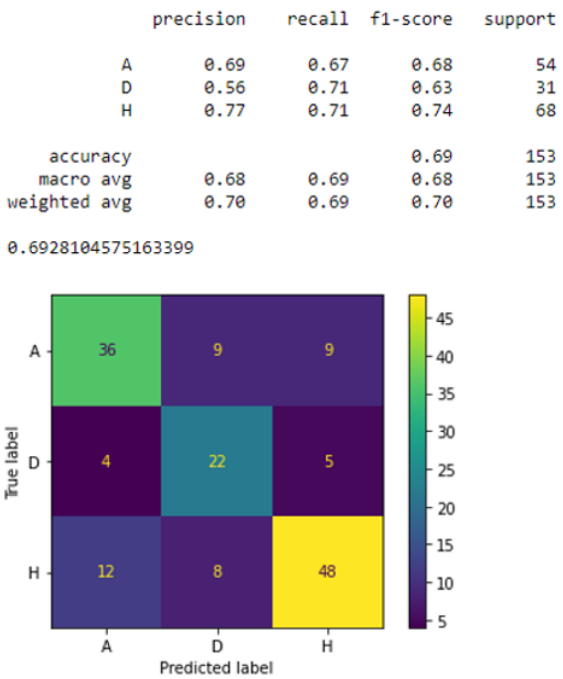


Figure 19. *Training results of a Naïve Bayes model.*

7. Conclusion

Football match prediction using machine learning can be performed using various techniques and methods. According to previous research [26], most football match predictions achieve an accuracy of between 55% and 70%, and so did the predictions in this implementation. The highest prediction accuracy achieved in this implementation was 63.0%, using six team related attributes with the logistic regression algorithm. To achieve a higher accuracy, the predictions could have been made using multiple seasons of data so that the models would have had more data to train on. The primary factor why a greater accuracy was not achieved in this implementation, was that the models were not able to predict enough draws correctly in relation to home wins and away wins.

Many previous studies about football match predictions have used Bayesian networks, logistic regression, and artificial neural networks. In this implementation, the Naïve Bayes algorithm did not perform best with any type of data, but it was the most balanced algorithm. Logistic regression performed best using team attributes, and random forest performed best using player attributes and match statistics. In future research, artificial neural network could be compared with these algorithms and tested on the three different data sets used in this implementation. The framework for sport result predictions made by Bunker and Thabtah [23] worked well for football predictions in this implementation.

This thesis supports the argument that virtual video game data could be used for predicting real football matches with great accuracy. Although it is impossible to predict football match results perfectly, it is possible to achieve accuracies that are much greater than for randomly made predictions. By comparing multiple different combinations of machine learning algorithms and data sets, surprisingly accurate football match predictions can be achieved.

Svensk sammanfattning

Förutsägelse av fotbollsmatcher med hjälp av maskininlärning

Fotboll är världens populäraste sport. Den har blivit så populär tack vare sin enkelhet och förmånlighet. Fotboll spelas världen över av personer tillhörande alla kön och i alla åldrar. Därför är det ingen överraskning att det rör sig mycket pengar inom professionell fotboll. Det intressantaste inom fotboll är förstås själva resultaten av matcherna och vad som påverkar dem. För att kunna påverka resultaten måste man kunna förutsäga dem med hjälp av data. Maskininlärning är en utmärkt metod för att göra förutsägelser. I den här avhandlingen analyseras och jämförs olika metoder och strategier för att hitta det bästa tillvägagångssättet. Ju bättre noggrannhet som en metod uppnår i sina förutsägelser, desto bättre anses den vara. Fotbollsmatcher kan sluta på tre olika vis: vinst för hemmalaget, oavgjort eller vinst för bortalaget. För att hantera sådana förutsägelser kan man använda flerklass klassificering.

Förutsägelse av fotbollsmatcher är inte en enkel uppgift. Det är omöjligt att förutsäga dem exakt, även med hjälp av maskininlärning. Om det skulle vara möjligt, skulle spänningen i sporten förloras och vadslagningsbyråerna skulle gå i konkurs. En orsak att fotbollen har fått en så stor popularitet är att den är så oförutsägbar. I varje match är det möjligt för ett svagare lag att vinna över ett starkare, inte bara med hjälp av taktiska drag, utan också med hjälp av tur. Faktorer så som vädret och hemmafavör kan leda till oväntade resultat.

Under de senaste åren har mängden tillgängliga data om fotboll och andra sporter ökat enormt. Det här har möjliggjort för forskare och privatpersoner att själva utveckla och förbättra metoder för att förutsäga fotbollsmatcher. För vissa personer är målet med förutsägelserna att uppnå en så hög vinst som möjligt i vadslagning. Däremot är då också syftet att hitta svaga punkter i vadslagningsbyråernas förutsägelser, och inte bara att förutsäga de rätta resultaten på matcherna. I denna avhandling är målet endast att analysera hur bra man kan förutsäga fotbollsmatcher med hjälp av maskininlärning.

Översikt av maskininlärning

Maskininlärning är ett delområde inom artificiell intelligens och datavetenskap. Det är en teknik som lär sig av data med hjälp av algoritmer och försöker uppnå bättre och bättre resultat. En maskininlärningsmodell är resultatet av tränandet med en specifik maskininlärningsalgoritm och specifika indata. Maskininlärning går ut på att en maskin tränas att göra förutsägelser från observationer och tidigare erfarenhet. Det enda som vi behöver berätta för maskinen är vad vi vill att ska förutsägas och med hjälp av vilka data. Skillnaden mellan manuell programmering och maskininlärning är att omvandlingen av indata till utdata är okänd i maskininlärning. I manuell programmering är både indata och utdata känd, och vi vet hur algoritmen ska skapas för att omvandlingen ska fungera korrekt. Omvandlingsprocessen i maskininlärning är däremot okänd. Utseendet på utdata kan vara känt, men inte vilken indata som resulterar i vilken utdata.

Det första steget i maskininlärning är att definiera problemet. Valet av indata och maskininlärningsalgoritm beror på hurudant problem som är definierat. Det första praktiska steget i maskininlärning är datainsamling. Både kvaliteten och kvantiteten på data påverkar maskininlärningsmodellernas resultat. Följande steg är databehandling, vilken består av datarensning och val av attribut. Data som innehåller felaktiga värden eller saknar värden måste tas bort eller korrigeras. Data i textformat måste dessutom konverteras till numeriska värden, eftersom maskininlärning använder sig av matematiska funktioner. Attributen kan väljas enligt hur relevanta de är eller hur bra resultat de ger i testmodeller.

Följande steg efter databehandlingen är valet av maskininlärningsalgoritm. Maskininlärning kan delas in i tre huvudtyper: övervakat lärande, oövervakat lärande och semi-övervakat lärande. Övervakat lärande använder indata som är klassificerade, medan oövervakat lärande använder indata som inte är klassificerade från början, utan det är helt och hållet maskinens uppgift att hitta någon slags struktur. De vanligaste typerna av maskininlärningsalgoritmer är regression och klassificering, som båda hör till övervakat lärande. Regression är en metod som visar relationen mellan variabler genom att skapa en funktion som passar data. Regression är speciellt bra på att förutsäga numeriska värden. Klassificering förutsäger kategoriska värden i stället för numeriska värden. Alla möjliga värden

på utdata, som indata ska klassificeras i, kallas för klasser. Om det finns fler än två klasser, kallas det för flerklass klassificering.

Efter att en maskininlärningsalgoritm har valts, kan modellen börja tränas med indata. Indata måste delas upp i träningsdata och testdata så att delen träningsdata är större. Testdata används för att testa hur bra maskininlärningsmodellen har lärt sig av träningsdata. Det sista steget i maskininlärning är utvärdering av resultaten. Den vanligaste måttet i utvärderingen är noggrannhet, alltså hur många procent av observationerna som förutsades rätt.

Projekt

Det finns stora skillnader mellan olika sporter och tävlingar, med tanke på hur bra man kan förutsäga dem. En av de viktigaste faktorerna är hur stor nivåskillnaden är mellan de tävlande. En annan viktig faktor är hur många poäng eller mål som de tävlande vanligtvis får i en tävling. Fotboll är en sport där lagen vanligtvis gör relativt få mål, vilket försvårar förutsägelsen av resultatet. Ändå har vissa studier inom förutsägelse av fotbollsmatcher lyckats uppnå noggrannheter närmare 90 procent [1]. De flesta studier har dock uppnått en noggrannhet mellan 55 och 70 procent [26]. De vanligaste maskininlärningsalgoritmerna som används för att förutsäga fotbollsmatcher är naiv bayesiansk klassificerare, logistisk regression och artificiella neuronnät.

Målet med detta projekt var att jämföra olika maskininlärningsalgoritmer och metoder, och analysera hur bra man kan förutsäga fotbollsmatcher med hjälp av dem. Projektet innefattar hela maskininlärningsprocessen, från datainsamling till utvärdering av resultaten. I projektet förutsades resultaten på fotbollsmatcher i den tyska högsta divisionen Bundesliga, säsong 2015–2016. Endast matchernas slutresultat förutsades, alltså inte exakt hur många mål som lagen gjorde. Förutsägelserna utfördes med hjälp av data som är relaterad till lagen som helhet, de individuella spelarna och lagens tidigare matchresultat.

Flera olika datakällor användes i projektet. Data om lagen och spelarna hämtades från en SQL-databas vid namn European Soccer Database. Den publicerades online på Kaggle av Hugo Mathien. Databasen innehåller attribut, vars ursprungliga källa är fotbollspelserien EA Sports FIFA. Lagens och spelarnas skicklighet inom olika

kategorier har betygssatts mellan 1–99, där 99 är det bästa möjliga betyget. Databasen innehåller också matchresultat, men vissa attribut var ofullständiga eller saknade värden. I stället hämtades matchstatistik på den valda ligan och säsongen från en CSV-fil på Datahub.io.

Nästa steg efter datainsamling är databehandling. Databehandlingen utfördes i programmet Visual Studio Code med programmeringsspråken Python och SQL. För att kunna förutsäga matcherna med hjälp av de tre olika typerna av data, var det nödvändigt att förena dem med information om varje match. Informationen som behövs är vilka lag som spelade mot varandra och matchresultatet. Efter att lag- och spelarattributen hade förenats med matcherna, togs onödiga attribut bort från tabellerna. Tabellerna konverterades sedan till CSV-filer, för att enkelt kunna hanteras i miljön för själva maskininlärningen.

Maskininlärningsmodellerna skapades med Python i Jupyter Notebook. Tre maskininlärningsalgoritmer testades i projektet: naiv bayesiansk klassificerare, logistisk regression och slumpmässig skog. Naiv bayesiansk klassificerare är baserat på sannolikheter och är bra på att beakta tidigare händelser. Logistisk regression är specialiserad på binär klassificering, men genom att använda flera olika kombinationer av klasser fungerar den också för flerklass klassificering. En slumpmässig skog består av flera beslutsträd, vilka är trädformade strukturer som använder villkorlig logik.

Indata delades upp i träningsdata och testdata, så att 70 procent användes som träningsdata och 30 procent som testdata. I den ursprungliga indata med lagattribut, fanns det 20 attribut. Maskininlärningsmodellerna uppnådde noggrannheter runt 50 procent då alla attribut användes. De attribut som hade minst betydelse på resultatet togs inte i beaktande i följande test. Det bästa resultatet med lagattribut, uppnåddes av en modell som använde sex attribut med algoritmen logistisk regression. Modellen lyckades uppnå en noggrannhet på 63 procent. I den ursprungliga indata med spelarattribut fanns det 35 attribut för alla de 22 spelarna i en match. Eftersom det totala antalet attribut blev så stort, testades kombinationer av endast några attribut per spelare. Den bästa modellen som använde spelarattribut, använde attribut över spelarnas totala skicklighet med algoritmen slumpmässig skog. Modellen lyckades uppnå en noggrannhet på 54 procent. För att kunna använda tidigare matchstatistik i förutsägelsen av kommande matcher måste man räkna ut

medeltal av attributen. Ett medeltal av statistiken, från de 20 senaste matcherna för varje lag i varje match, räknades ut. Vid användning av tidigare matchstatistik är det viktigt att beakta den kronologiska ordningen på matcherna, så att matcherna i träningsdata har inträffat före matcherna i testdata. Den bästa modellen som använde matchstatistik, använde algoritmen slumpmässig skog. Den lyckades uppnå en noggrannhet på 55 procent.

Slutsats

Förutsägelse av fotbollsmatcher kan utföras på många olika sätt med hjälp av maskininlärning. Enligt tidigare forskning uppnår de flesta förutsägelser av fotbollsmatcher en noggrannhet mellan 55 och 70 procent [26]. Det stämmer överens med det bästa resultatet som uppnåddes i mitt projekt. Den högsta noggrannheten uppnådd var 63 procent, i en modell som använde lagattribut med algoritmen logistisk regression. För att få ökad noggrannhet i projektet, kunde man ha använt data från flera säsonger, så att modellerna skulle få mera data att träna på. Den största orsaken till att inte en större noggrannhet uppnåddes i projektet, var att modellerna inte gjorde tillräckligt många korrekta förutsägelser av oavgjorda resultat, i relation till hemmavinster och bortavinster.

Många studier inom förutsägelse av fotbollsmatcher har använt naiv bayesiansk klassificerare, fast i detta projekt presterade den sämre än logistisk regression och slumpmässig skog i de flesta test som gjordes. En nackdel med naiv bayesiansk klassificerare är att den antar att alla attribut är oberoende från varandra, vilket inte är fallet i en stor del av den data som användes i detta projekt. Den bästa algoritmen för lagattribut var logistisk regression, medan den bästa algoritmen för spelarattribut och matchstatistik var slumpmässig skog. Naiv bayesiansk klassificerare presterade dock mest balanserat utav de tre algoritmerna.

Fastän det är omöjligt att förutsäga resultaten i fotbollsmatcher perfekt, går det att uppnå noggrannheter som är mycket större än slumpmässiga resultat. Genom att jämföra flera olika kombinationer av maskininlärningsalgoritmer och indata kan man uppnå förvånansvärt bra resultat.

Bibliography

[1] 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,* vol. 5, no. 5, 2013.

[2] W. Rahman, in *AI and machine learning*, New Delhi, India, SAGE Publications Pvt Ltd, 2020, pp. 19-20, 38-41.

[3] O. Campesato, in *Artificial intelligence, machine learning, and deep learning*, Dulles, Virginia ; Boston, Massachusetts ; New Delhi, Mercury Learning and Information, 2020, pp. 26-31, 39, 67-73, 87-89, 105.

[4] E. Alpaydin, in *Machine Learning*, Massachusetts, MIT Press, 2021, pp. 16-18, 105-107, 127-129, 143.

[5] J. Rogel-Salazar, in *Data Science and Analytics with Python*, New York, Chapman and Hall/CRC, 2017, pp. 102-104.

[6] Handelman et al, “Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods,” *American Journal of Roentgenology,* vol. 212, no. 1, pp. 38-43, 2019.

[7] T. Dietterich, “Overfitting and Undercomputing in Machine Learning,” *ACM Computing Surveys,* vol. 27, no. 3, 1995.

[8] V. Nasteski, “An overview of the supervised machine learning methods,” Bitola, Macedonia, 2017.

[9] N. Grira, M. Crucianu and N. Boujemaa, “Unsupervised and Semi-supervised Clustering: a Brief Survey,” INRIA Rocquencourt, Le Chesnay Cedex, France, 2015.

[10] M. Mohammed, M. B. Khan and E. B. M. Bashier, Machine Learning: Algorithms and Applications, Auerbach Publications, 2017.

[11] Liu et al, Computational and Statistical Methods for Analysing Big Data with Applications, Australia: Academic Press, 2016.

[12] J. Cheng and R. Greiner, “Comparing Bayesian Network Classifiers,” University of Alberta, Edmonton, Alberta, Canada, 1999.

[13] T. Soni Madhulatha, “AN OVERVIEW ON CLUSTERING METHODS,” *IOSR Journal of Engineering,* vol. 2, no. 4, pp. 719-725, 2012.

[14] Choi et al, “Introduction to Machine Learning, Neural Networks, and Deep Learning,” *Translational Vision Science & Technology,* vol. 9, no. 2, 2020.

[15] J. Brownlee, “4 Types of Classification Tasks in Machine Learning,” https://machinelearningmastery.com/types-of-classification-in-machine-learning/, Accessed 5 March 2023.

[16] M. Aly, “Survey on Multiclass Classification Methods,” 2005.

[17] H. K. Jabbar and R. Z. Khan, “METHODS TO AVOID OVER-FITTING AND UNDER-FITTING IN SUPERVISED MACHINE LEARNING (COMPARATIVE STUDY),” Aligarh Muslim University, Aligarh, India, 2015.

[18] X. Ying, “An Overview of Overfitting and its Solutions,” *Journal of Physics: Conference Series,* vol. 1168, no. 2, 2019.

[19] Sportradar Integrity Services, “BETTING CORRUPTION AND MATCH-FIXING IN 2021,” 2022.

[20] E. Štrumbelj, “On determining probability forecasts from betting odds,” *International Journal of Forecasting,* vol. 30, no. 4, pp. 934-943, 2014.

[21] L. Kaunitz, S. Zhong and J. Kreiner, “Beating the bookies with their own numbers - and how the online sports betting market is rigged,” Cornell University, 2017.

[22] G. Fialho, A. Manhães and J. P. Teixeira, “Predicting Sports Results with Artificial Intelligence - A Proposal Framework for Soccer Games,” *Procedia Computer Science,* vol. 164, pp. 131-136, 2019.

[23] R. P. Bunker and F. Thabtah, “A machine learning framework for sport result prediction,” *Applied Computing and Informatics,* vol. 15, no. 1, pp. 27-33, 2019.

[24] A. Almazov, “The main factor in predicting football bets,” https://alvin-almazov.com/theory/the-main-factor-in-predicting-football-bets/, Accessed 10 April 2023.

[25] S. Wilkens, “Sports prediction and betting models in the machine learning age: The case of tennis,” *Journal of Sports Analytics,* vol. 7, no. 2, pp. 99-117, 2021.

[26] T. Horvat, “The use of machine learning in sport outcome prediction: A review,” *WIREs Data Mining and Knowledge Discovery,* vol. 10, no. 5, 2020.

[27] D. Buursma, “Predicting sports events from past results “Towards effective betting on football matches”,” in *14th Twente Student Conference on IT*, Twente, Holland, 2011.

[28] N. Tax and Y. Joustra, “Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach,” TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2015.

[29] Betting Offers, “Home Advantage in Football: How Often Does the Home Team Win?,” https://www.bettingoffers.org.uk/how-big-is-home-advantage-in-football/, Accessed 7 February 2023.

[30] Y. S. Taspinar, I. Cinar and M. Koklu, “Improvement of Football Match Score Prediction by Selecting Effective Features for Italy Serie A League,” *MANAS Journal of Engineering,* vol. 9, no. 1, pp. 1-9, 2021.

[31] J. Shin and R. Gasparyan, “A novel way to Soccer Match Prediction,” Stanford University, 2014.

[32] D. Prasetio and D. Harlili, “Predicting football match results with logistic regression,” in *International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, Penang, Malaysia, 2016.

[33] Razali et al, “Predicting Football Matches Results using Bayesian Networks for English Premier League (EPL),” in *IOP Conf. Ser.: Mater. Sci. Eng.*, Malaysia, 2017.

[34] H. Chen, “Neural Network Algorithm in Predicting Football Match Outcome Based on Player Ability Index,” *Advances in Physical Education,* vol. 9, no. 4, 2019.

[35] M. A. Rahman, “A deep learning framework for football match prediction,” *SN Applied Sciences,* vol. 2, no. 165, 2020.

[36] H. Mathien, “European Soccer Database,” https://www.kaggle.com/datasets/hugomathien/soccer, United Kingdom, 2016.

[37] R. Murphy, “FIFA player ratings explained: How are the card number & stats decided?,” https://www.goal.com/en/news/fifa-player-ratings-explained-how-are-the-card-number--stats-decided/1hszd2fgr7wgf1n2b2yjdpgynu, Accessed 13 March 2023.

[38] FIFA Index, “EA Sports FIFA16,” https://www.fifaindex.com/teams/fifa16_73/?league=19&order=desc, Accessed January 2023.

[39] DataHub.io, “season-1516,” https://datahub.io/sports-data/german-bundesliga, Accessed January 2023.

[40] H. Bhavsar and A. Ganatra, “A Comparative Study of Training Algorithms for Supervised Machine Learning,” *International Journal of Soft Computing and Engineering (IJSCE),* vol. 2, no. 4, 2012.