Technische Universität Berlin

Fakultät V
Industrielle Automatisierungstechnik

Machine Learning to Predict Football Results

Final Report

Students Jannis Grönberg 356198
Date 08. March 2020

Contents

List of Figures

List of Tables

1 Introduction

The authors of this work are master degree students of the TU Berlin in the field of Computational Engineering Sciences. This interdisciplinary study primarily includes mechanical engineering and computer science. Therefore, the proposed project topic caught attention of the authors as it is an opportunity to apply theoretical knowledge to a defined practical problem.

In particular, the given task is to develop a multi layer perceptron to maximize the revenue when betting on teams of the first German Bundesliga.

1.1 Motivation

Data Science and big data analytics are one of the largest research areas of this decade. A common goal is to generate or maximize profits by analyzing data and effectively applying the findings. According to idc.com[1] worldwide revenues for big data and business analytics (BDA) solutions are forecast to reach 189.1 billion dollar this year, an increase of 12.0% over 2018.

Another big market with a global value of 104.31[2] billion dollars in 2017 is the sports betting market. While big companies increase their profits annually the overall market is still highly dependent on individual people placing bets and with that losing money.

This project should bring publicly available data, data science and sports betting together.

[1] https://www.idc.com/getdoc.jsp?containerId=prUS44998419

[2] https://www.globenewswire.com/news-release/2019/08/29/1908388/0/en/Global-Sports-Betting-Market-Size-Share-Will-Reach-USD-155-49-Billion-By-2024-Zion-Market-Research.html

1.2 Related Work

Recent publications cover the topic of predicting sport outcomes using artificial intelligence. A few of them are introduced in the following paragraphs.

Emmanuel Esumeh proposes a lazy-learning technique, a nearest neighbours algorithm, for predicting football game season winners of the Premier League [Esu15]. He uses the Euclidean distance within the six dimensional space of the number of wins, draws, losses, goals for, goals against, goal difference and points. Finally, after 19 matches played, all teams are compared by distance to the mean vector of season winners of previous seasons.

Corentin Herbinet however, considers the number of goals scored as too random based to use it as a measure for machine learning approaches [Her18]. He defines an "expected goals" metric, which is then taken as a source for different machine learning algorithms including an artificial neural network. Compared to different other algorithms as logistic regression or the k-nearest-neighbours algorithm his implementation of a neural network showed an accuracy of roughly 50.5 % only being beaten by a support vector machine with linear classification models having an accuracy of nearly 51 %. The odds of the bookmakers showed an accuracy of 52.1 % according to Herbinet and were therefore better than any proposed machine learning model implementation.

In summary, none of the introduced methods could beat the prediction of the bookmakers. However, the MLP shows the best accuracy among different machine learning techniques. Useful features can be found in the accumulated number of different outcomes, goals and points during a season. Additionally, features can be engineered leading to better results.

1.3 Research Questions

The given task is to develop a model based on a MLP to predict the outcome of football games of the German Bundesliga. Using this model and the prior knowledge of the introduced related work following research questions arise and should be answered.

Q1 Does the prediction of the model perform better than always betting on the home team?

Q2 Does the prediction of the model perform better than always betting on the likeliest outcome based on the quotes of the bookmakers?

Q3 Does engineering features lead to more accurate predictions?

Q4 Can strategies that maximize the profit of a bettor be found?

2 Theory

In this section theoretical concepts of the project are explained. First principle component analysis (PCA - section 2.1) is explained. PCA is implemented so as to reduce the feature space and save computation time. Afterwards an introduction to multi layer perceptrons (MLP) is given in section 2.2. MLP is used as the classification model for this project. This introduction is followed by explanations on how to scale (section 2.3) and encode (section 2.4) data. Section 2.5 defines the two evaluation metrics precision and recall and their combination F_1-score. At the end, section 2.6 explains the principal of k-fold cross validation.

2.1 Principal Component Analysis

The purpose of principal component analysis is to structure, simplify and illustrate large data sets by approximating a large number of statistical variables by a smaller number of linear combinations that are as meaningful as possible (the "principal components"). This makes data easier to explore and visualize.

Principal components analysis mainly reduces dimensions and finds relationships between hidden features. The goal of principal component analysis is to transform these data points into a q-dimensional subspace in a way that as little information as possible is lost and existing redundancy is summarized in the form of correlation in the data points.

Mathematically, a transformation of the main axis is carried out: Correlation of multidimensional features is minimized by transformation into a vector space with a new basis. The principal axis transformation can be specified by an orthogonal matrix, which is formed from the eigenvectors of the covariance matrix. The highest Eigenvalue corresponding to the highest eigenvector is considered as PC1, the new latent variable and new dimension in the direction of the highest variance. The rest of the principle components are extracted from the decomposed correlation matrix

accordingly. Features with low variance are considered to be less important and are dropped. [Bre].

Dimensionality reduction can be achieved in many ways. The most common techniques are:

1- Feature Elimination

2- Feature Extraction

Feature Elimination drops all variables except the ones' which might yield the best results. This method is simple and the variables are still interpretable. However, feature elimination leads to information loss, since there is no information gain from the dropped features [Bre].

Feature Extraction creates new independent variables, by combining the old independent variables. This technique creates the new features (latent variables) in a specific way and orders them by the highest prediction of dependent variables [Bre].

Principal component analysis focuses on feature reduction. It identifies a smaller number of uncorrelated features and combines them into principal components. Principal component analysis tries to identify the most important directions in the data and projects the data into smaller space by dropping the least important directions.

In detail principle component analysis works as follows: A data set organized with n rows and p columns is given. The data set is separated into Y and X. Y will be the validation set and X the training set. The variables in X are independent. The data set is then centered. This is to ensure that each column has a mean of zero. In the next step, X is standardized. Each observation in a column is divided by the standard deviation. The standardized matrix is called the covariance matrix. After that the covariance of Z is calculated by transposing Z and multiplying it with Z ($Covariance = Z^T Z$).

Eigenvectors and there corresponding eigenvalues of $Z^T Z$ are calculated. $Z^T Z$ is decomposed into PDP^{-1}, where P is the eigenvectors matrix and D the diagonal of the matrix containing the eigenvalues. The eigenvalues on the diagonal of D are associated with the corresponding column in P. Then the eigenvalues will be sorted from highest to lowest. The eigenvectors are sorted accordingly. The sorted eigenvectors matrix is called P^*.

At this point the new features will be calculated $Z^* = ZP^*$. This new matrix Z^*, is centred/standardized version of X but each observation of Z^* is a combination of the original variables, where the weights are determined by the eigenvector.

Figure 2.1 shows the original data X and on figure 2.2 the transformed data Z^* is illustrated.

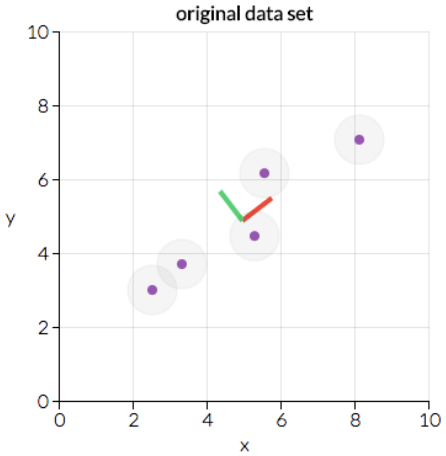

Figure 2.1: Data before PCA (X) [Bre]

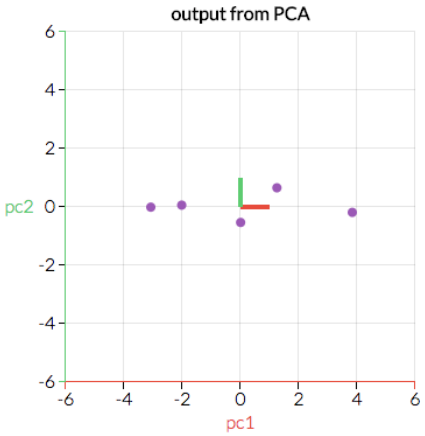

Figure 2.2: Data after PCA (Z^*) [Bre]

2.2 Multi Layer Perceptron (MLP)

Regularly artificial neural networks are referred to as neural networks or multi layer perceptrons (MLPs). A perceptron is a single neuron model which was a precursor to bigger neural networks.

Multi layer perceptrons mimic uncomplicated models of biological brains and are utilized in order to solve difficult computational tasks, such as the predictive modeling tasks in machine learning. The intention is not limited to creating realistic models of the brain, but to modelling difficult problems through the development of robust algorithms and data structures.

Neural networks function well because of their capability to learn the representation in training data and how to best link it to the output variable which is aimed to be predicted. Basically neural networks learn complex and non-linear mapping functions.They are mathematically able of learning any function and have been proven to be a comprehensive approximation algorithm. They are adaptable and can commonly be utilized for most of the classification tasks.

The predictive capability of Neural networks comes from the neural networks hierarchical or multi-layered structure. Features of different resolutions and scales are can chosen by the data structure and combined into higher-order features. For example from lines, to collections of lines to shapes. Each perceptron simulates a linear function that when combined with other turn into a none linear function.

In regression prediction problems MLPs are convenient for predicting a real-valued quantity, given a set of inputs as well as for classification prediction problems where inputs are assigned to a class or a label.

Classifying win, draws and losses for football games is a supervised learning task. There are various approaches to solve these tasks. One of these approaches is the Multi layer perceptron (MLP). MLP achieved outstanding results in similar tasks. Purucker used the supervised and unsupervised neural networks to analyze and predict the National Football League (NFL) win rate, and found that the Perceptron Multi Layer Neural Network (MLP) with supervised learning worked better than the self-organizing map network of Kohonen with unsupervised learning [Pur96]. Condon et al. used MPL to predict the score of a country which participated in the 1996 Summer Olympic Games[CGW99]. The results exceeded that of a regression model [Zoh].

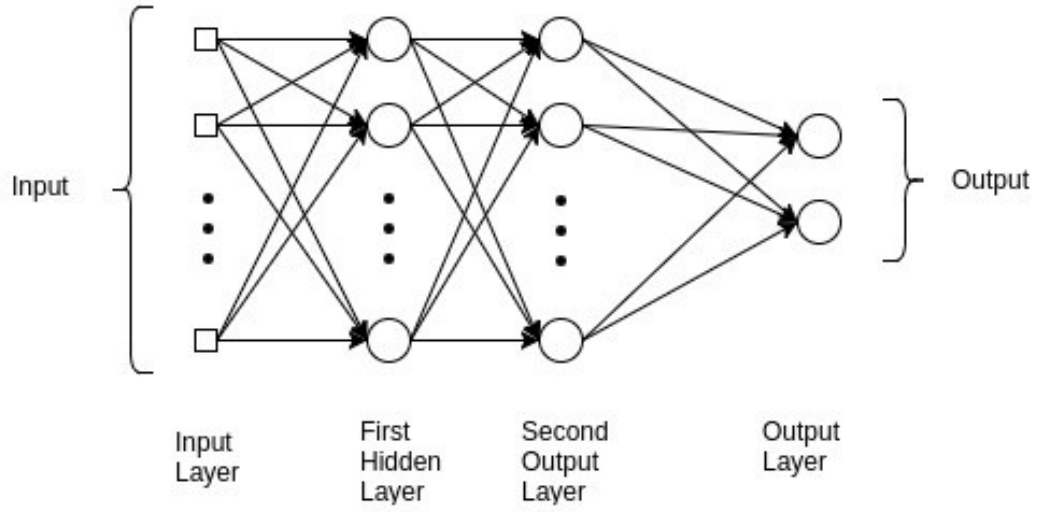

Figure 2.3: Multi layer perceptron [Kai]

Figure 2.3 pictures a multi layer perceptron. A Multi layer perceptron consists of multiple linear layers. Simple MLPs contain three layers, the first layer is the input layer and the last layer is the output layer. The middle layer is called the hidden layer.

Number as well as size of hidden layers can be changed in order to fit independent classification tasks optimally [Kai].

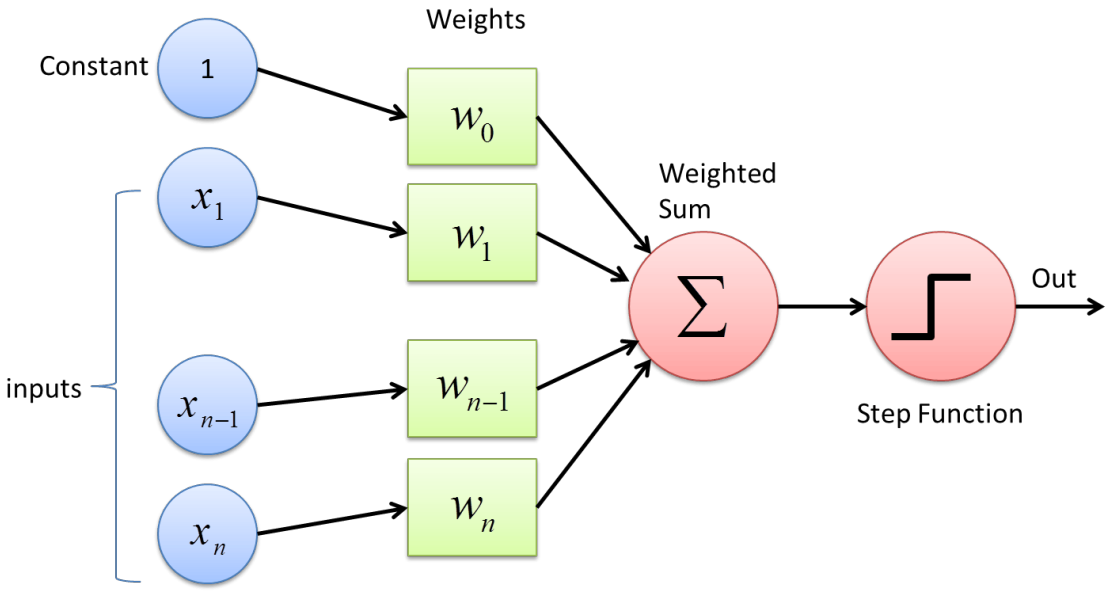

Figure 2.4: Perceptron [shab]

A layer is a combination of perceptrons. One perceptron (see figure 2.4) consists of 4 parts [shab]:

1. One or more input values
2. Weights and bias
3. Net sum
4. Activation function

Weights represent the strength of a particular node. The bias value allows to shift the activation function curve up or down. Inputs are multiplied with their weights and the sum of these is called weighted sum, which is applied to the correct activation function. The weights are updated when a classification error or misclassification is detected by using back propagation.

$$weight = weight + learning\_rate \cdot (expected - predicted) \cdot x \tag{2.1}$$

In a supervised classification task, each input is associated with a ground truth called label. The output of an MLP gives a probability for each label. The prediction of the model is the label with the highest predicted probability.

2.2.1 Activation Functions

Activation functions are a used to propagate the output of each perceptron through the model forward and onto the subsequent layer. The inputs are mapped onto the range of required values like (0,1) or (-1,+1) (depending on the function).

The Activation function falls into two categories [shaa]:

1. Linear activation functions
2. Softmax activation function
3. Non-linear activation functions

Figure 2.5 shows a linear activation function. The output of the function is not limited to a specific range.

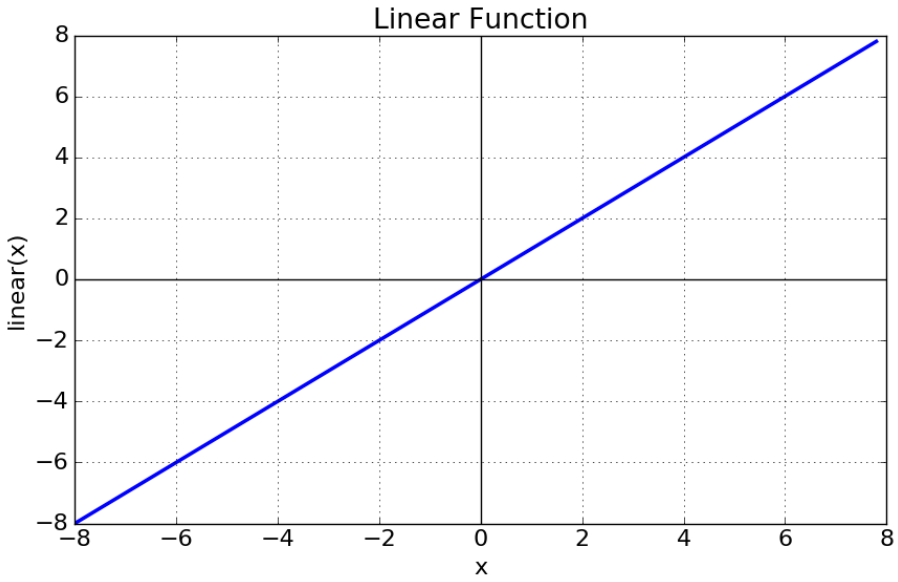

Figure 2.5: Linear activation function [shaa]

Instead of linear functions, non-linear activation functions are used more commonly. Mainly they are distinguished from linear functions by the basis of their range. The most two important non-linear activation function are:

1 Sigmoid or Logistic Activation Function.

2 ReLU (Rectified Linear Unit) Activation Function

Sigmoid or Logistic Activation Function are normally used in machine learning for the logistic regression and basic neural network implementations. They are the introductory activation units. For advanced Neural Network sigmoid functions are not favored due to diverse snags. In spite of the fact that sigmoid function and it's derivative are simple and therefore easily explainable there is a major disadvantage where information loss caused by the the short range derivative.

$$sigmoid(z) = \frac{1}{1+e^{-z}} = \frac{1}{2} \cdot \left(1 + tanh\frac{z}{2}\right) \tag{2.2}$$

The Softmax Activation Function is a generalized logistic activation function that has an analogous structure to the sigmoid function. This activation function is similar to the sigmoid activation function where well performance is witnessed when used as classifier. The most substantial distinction is that it is preferred in the output layer of deep learning models, especially when it is necessary to classify more than two classes. An input value is normalized by Softmax normalizes into a value vector, that follows a probability distribution, whose total sum is 1. This is beneficial for the output layer but if further layers in a Neural Network with this activation function exist, more information is compressed and lost at each layer. This boosts and leads to overall major data information loss.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.3}$$

The ReLU Activation Function is the most commonly used as activation function, especially in convolutional neural networks. It is linear for all positive values and zero for values smaller or equal to zero. Variants of ReLu might include : Noisy ReLU, Softplus (Smooth ReLU),Parametric ReLU , Leaky ReLU, and Exponential ReLU (ELU). Some of the disadvantages that come with ReLU are that ReLU is not centered at zero and cannot be differentiated at zero, but differentiable anywhere else. One more problem is the Dying ReLU problem where some ReLU Neurons essentially die and stay inactive regardless of the supplied input , impacting the gradient where no gradient flows. If large number of dead neurons are in a Neural Network it's functionality is negatively impacted. Leaky ReLU can be used as a workaround solution causing a leak and extending the range of ReLU.

$$R(z) = max(0, z) \tag{2.4}$$

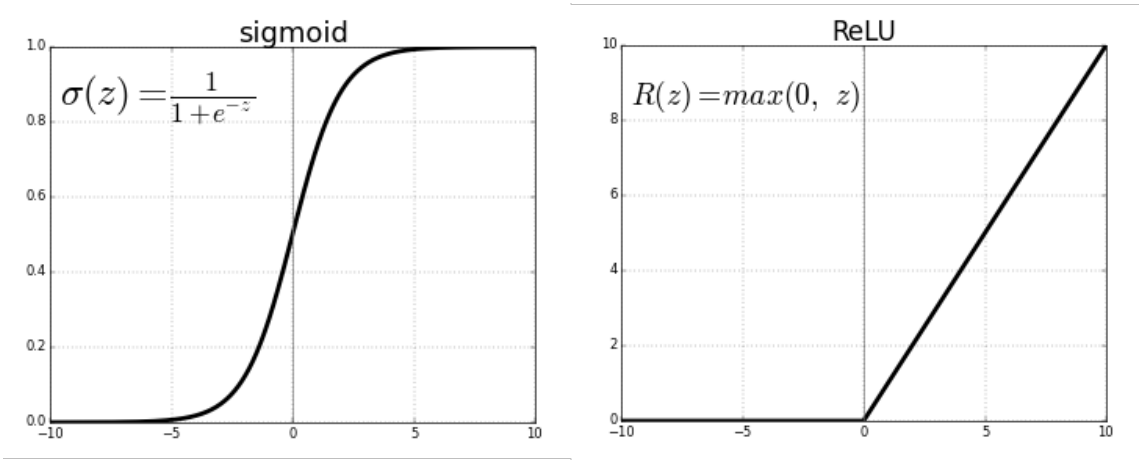

Figure 2.6: ReLU vs Logistic Activation Function [shaa]

2.2.2 Loss Function

Machines learn with help of loss function. It evaluates how well a specific algorithm models the given data. If predictions are totally off, loss functions result in large numbers.

Loss functions can be classified into two major categories depending on the learning task they are performed on:

1- Regressive loss functions

1- Classification loss functions

Regressive loss functions are used in regressive tasks, where the target variable is continuous. The most common regressive loss function is **Mean Square Error**.

Classification loss functions are used in case of classification tasks, where the target variable y is a binary variable, true and false. The most used classification loss function is the **Cross Entropy Loss**.

Cross Entropy loss increases if the predicted probability differs from the actual label. It multiplies the logarithm of the predicted probability for the ground truth class.

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 + y_i) log(1 - \hat{y}_i)) \tag{2.5}$$

If the output consists of one class, binary cross entropy is used, categorical cross entropy Otherwise

2.3 Scaling the Data

In article[1]: "About Feature Scaling and Normalization", Sebastian Raschka gives a detailed discussion about the necessity of scalers for machine learning models. Basically models require all input features to be in the same order of magnitude. If feature A is specified in thousands and feature B only holds values in milli-range, feature A would have a bigger impact. This is caused by the matrix multiplications inside of a neural network. As "typical neural network algorithms require data that are on a 0-1 scale"[1], only the concept of a min-max scaler is explained. For a detailed explanation of other scalers we refer to article[1].

"Using min-max scaling (often also called "normalization"), ... , data is scaled to a fixed range - usually 0 to 1. Min-Max scaling is typically done via the following equation"[1]:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.6}$$

2.4 One Hot Encoding

In many classification tasks the input features include categorical data (e.g. "Hertha", "Bayern" as examples for the feature team name). This is a problem as most machine learning algorithms cannot operate on labeled data directly, but require all input variables and output variables to be numeric.

Therefore words are converted into numeric data. If all variables of a feature have a defined order, assigning values that represent the defined order is sufficient.

[1] http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Original value	Encoded value
s	1
m	2
l	3
xl	4

Table 2.1: Encoded variables for clothes sizes

In table 2.1 four clothes sizes are encoded. After encoding the variables the encoded values still represent that order. A higher number is equivalent to a bigger clothing size. In cases where categorical values have no particular order different encoding approaches are used. As described by C. Seger [Seg18] one hot encoding performs better than other encoding techniques and should be considered as state of the art.

Original feature	One hot encoded features		
Team name	**Team name 1**	**Team name 2**	**Team name 3**
Hertha	1	0	0
Bayern	0	1	0
Dortmund	0	0	1

Table 2.2: One hot encoded variables for team names

Later data sets consists of 28 different team names. If simple numbers would be assigned to each team, a machine learning model would treat teams with close values almost equally even though there is no particular reason for this assumption. In table 2.2 the different team names don't represent a particular order. For each occurring name a new feature is created. A one is assigned to the feature in case of a match where the team plays. If the team is not present in the particular game a zero is assigned. This implies that for each occurring value in the encoded feature column a new feature is created. This results in a bigger feature space

2.5 Precision and Recall

Using the accuracy of a model as the sole method of evaluating test data has limited validity. If there is a large difference between the two numbers of classification, high

accuracy can be achieved by classifying all entries as a majority class. If one class A occurs 90 times and another class B occurs 10 times, an accuracy of 90% is achieved if the model classifies all examples as class A. However, the model is not able to successfully classify class B.

In order to measure the performance of a classification model more successfully, three metrics. In these metrics *True Positive (TP)* mean the number of correctly classified entries. Additionally there are two types of errors: *False Positive (FP)* are errors where a classifier identifies an entry as class A even though it should be the entry is of another class. *False Negative (FN)* are errors where the classifier does not classify entries of class A as such.

The evaluation metrics **Precision** and **Recall** are introduced to handle both of these errors independently. Precision and Recall are inversely proportional to each other. Understanding the difference between the two functions is important for building an efficient classification model.

Precision is the number of correctly classified results of a class divided by the number of all classified results of that class (see figure 2.7).

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

Precision indicates how accurate the determined results of the identification of a class are.

Recall is the number of correctly classified entries of a class, divided by the total number of all possible correct classifications of this class (see figure 2.7).

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

Recall indicates how efficiently a model can identify a class.

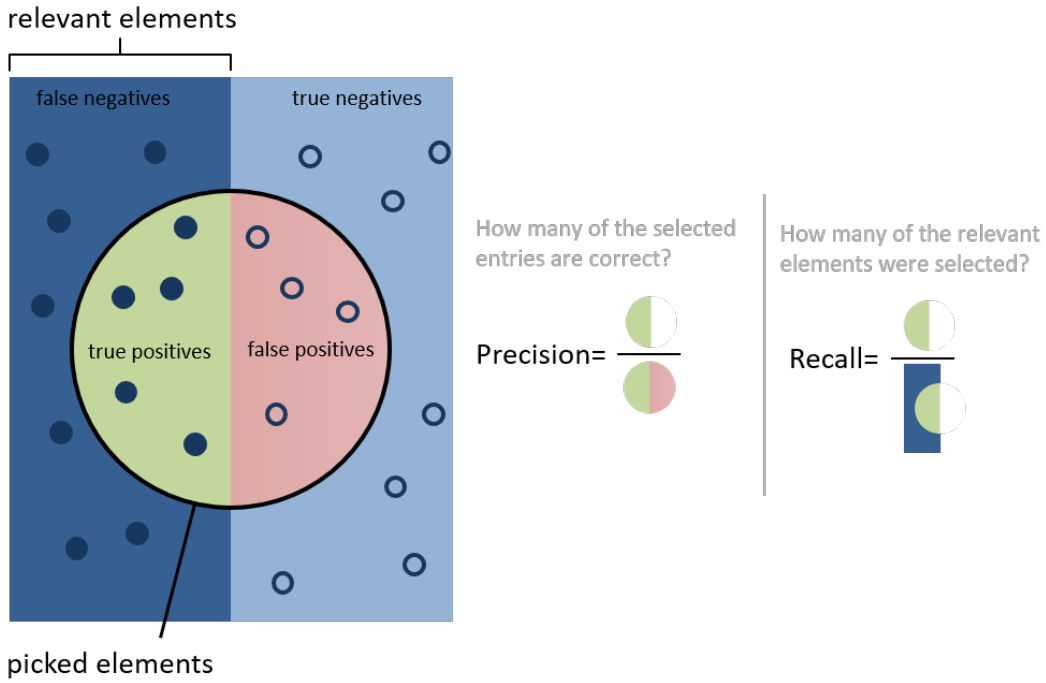

Figure 2.7: Precision und Recall

Looking at one of the two values alone is not enough to evaluate a classifier as a whole. Only by combining both values, a reliable statement about the performance of a can classifier be made. Therefore, both entries are usually combined by a function to form a new metric. This metric is called **F1-score**:

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Using F1-score, Precision and Recall are combined with the help of the harmonic mean.

2.6 K-fold Cross Validation

Usually a data set is divided once into training and test data. A training data set is often significantly larger than the test data set in order to extract as many features as possible when creating a model. However, it must be ensured that the test data set is not so small that no reliable statement about the results can be made. Another problem is the selection of appropriate data from the original data set for both, the training set and the test set.

If it is assumed that not all data provide the same amount of input information, a simple separation may result in either particularly "good" or particularly "bad" data for the problem. If the data is very similar, a particularly good result can be expected. If the data is not similar at all, a bad result is to be expected accordingly. Therefor the data in the training and test set have a significant influence on the result of a classifier. In order to avoid randomly selecting particularly suitable data, **k-fold cross validation** is used to split data sets.

Kohavi describes "that k-fold cross-validation requires less memory than other cross validation" [Koh95]. Therefore, the **k-fold cross validation**, illustrated in figure 2.8, will be dealt with in particular at this point.

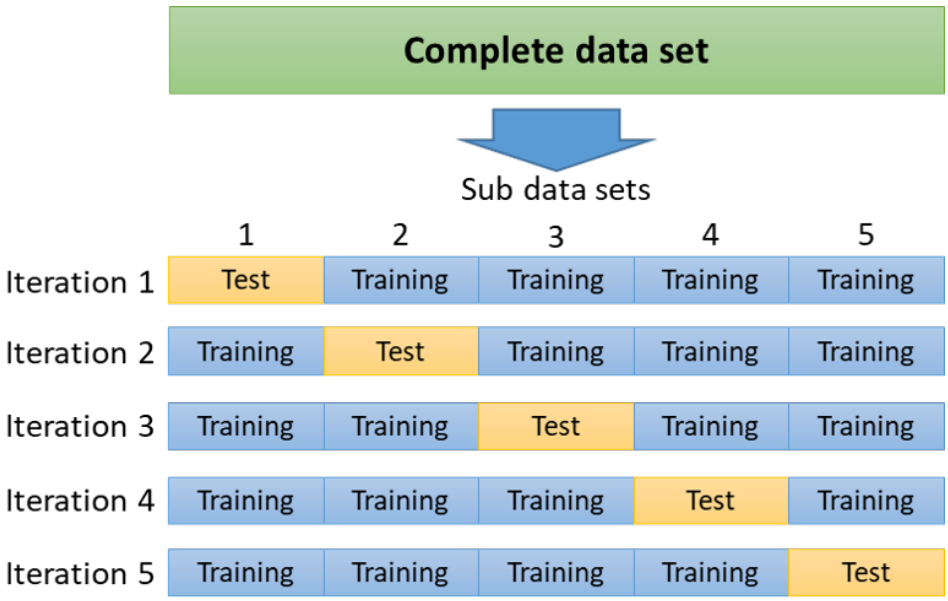

Figure 2.8: k-fold cross validation

The principle of k-fold cross validation is to divide the elements of a data set into k(number) equally large subsets. Subsequently, k Iterations are performed, in which the i-th subset T_i is used as test set. The remaining $k-1$ subsets are used as training data. The results of each iteration are stored.

The subsequently obtained metrics (Precision, Recall, and F1-score, see section 2.5) represent the average of all individual results of the test runs [Koh95]. A stratification of the individual subsets T_i results in the individual subsets having approximately the same distribution of classes as the entire data set. This has the effect of reducing the statistical variance of the estimate.

3 Implementation

Different steps need to be carried out in order to predict the football game outcome. Adequate data need to be found, cleaned and prepared. Additional features can be engineered based on the existing ones. The model type is chosen, trained and evaluated. During parameter tuning the model is optimized.

3.1 Data Sources

Three publicly available data sets, containing games of the German Bundesliga, were evaluated.

The most comprehensive data is from Kaggel [1] and contains over 25,000 matches, 10,000 players, FIFA video game player statistics and additional meta data as the position of the origin of each goal shot. Whereas the large number of features is generally a benefit, the drawback is that the last data point being captured is in the season between 2015 and 2016. As no other data set can fill the gap from 2016 and onward, a prediction of future matches would have to rely on over four year old data. For example team structures are changing over time, which the data set can not replicate.

Another data set can be found on football-data.co.uk [2], covering all the matches from 2002 until recently. The 61 features are separated season by season into multiple files. These need to be merged in order to get a complete set. Due to the consistency of the data this data set is chosen for further analysis.

[1] https://www.kaggle.com/hugomathien/soccer
[2] http://www.football-data.co.uk/

3.2 Feature Engineering

The chosen data set contains directly measurable dimensions as well as dimensions that are derived from the former ones. Those derived values are specified by a functional relationship and are called engineered features.

The measurable features are:

1. Team identity (i.e. names)
2. Goals scored by each team
3. Bookmaker odds
4. Game day of the season

Feature engineering uses these existing features to create new valuable information for the machine learning model, e.g. using this technique, Corentin Herbinet shows an improvement of his proposed prediction model by applying a certain metric to the data set [Her18].

The most important information in the context of betting is determining the winner of the match, which is done by the comparison of the total amounts of goals scored during the game. Furthermore, the goal difference can be calculated and used as an engineered feature.

Another engineered feature is the Elo rating. Introduced by Arpad Elo, it was originally used to estimate the skill level of chess players but is used in a wide range of sports nowadays. A score is assigned to each player which is increased in case of the player winning the match. When losing the game, the rating is decreased. In case of draws the value remains unaltered. The higher the score difference of two teams, the more likely is the higher ranked one to win a match.

Within a season the teams can be ranked. For each match, the winner is given three points, the losing team none. If the match ends with a draw, both teams will be given one point. The ranking can be found when sorting the teams by points in a descending order.

During each match, both teams played the same amount of matches during this season. The number of previous matches plus one is called the game day. This information might be of interest due to the seasonal character of the league. As the ultimate goal

is to be ranked as first place after the last game day, strategies and tactics might exist, which are dependant on the game day.

3.3 Data Set Characteristics

Consisting of 2,206 entries, the data set provides match information from 2012 until December 3rd 2019. Comparing the relative frequency of the possible outcomes of a match, as shown in figure 3.1, one can observe the home team win to be likelier than the away team win with probabilities of 45.6% respectively 29.8%. The unlikeliest outcome is a draw showing a probability of 24.5%. The research question Q1 asks for a comparison of the model to be developed with only betting on the home team. It should therefore be shown, if the accuracy of the model predicts better than 45.6%.

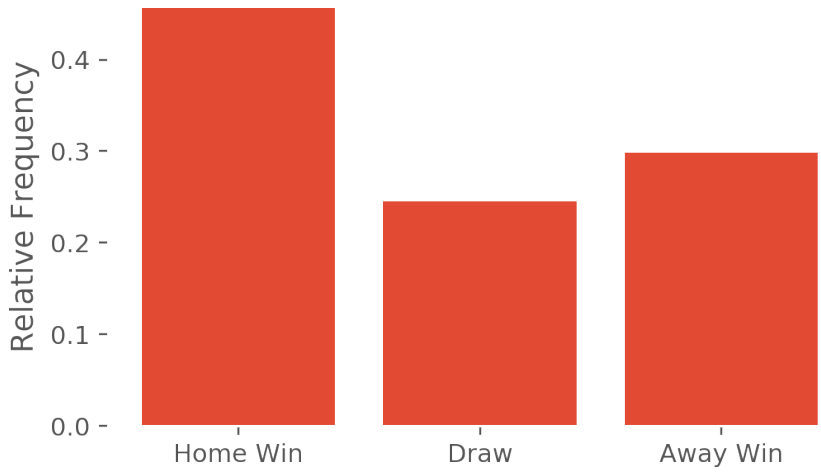

Figure 3.1: Probability of Possible Outcomes

The data set includes betting odds from bet365[3]. They represent the probability of the outcome of a match. A higher odd stands hereby for a lower likelihood of the outcome. When betting money on one of the three possible outcomes, the bet value is lost in case of the match ending differently. If the outcome was correctly predicted, the return value is the bet value multiplied with the betting odd. As shown by Smith et. al. betting odds incorporate useful information to the betting customer [SPW09]. Thus it is considered as performance benchmark for the model

[3]https://www.bet365.com/

to be developed in research question Q2. Always betting on the team with the lowest quote yields an accuracy of 45.7% if the team is playing home. In case of the team playing away the relative frequency of a correct prediction is 42.1%. The whole data set did not contain any match with draw being the likeliest event based on the odds.

Figure 3.2 illustrates the distribution of the betting odds. The interquartile range, shown as the colored body of the boxes, represent the range of the 50% of the data around the median. It can be seen that the away team odds show the most volatile behaviour. However, the highest median, and therefore the unlikeliest event, is given by the draw odds.

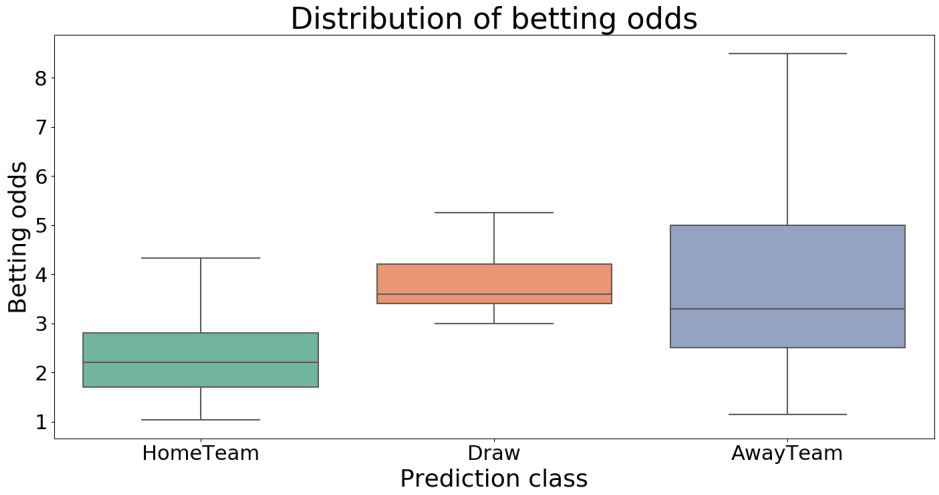

Figure 3.2: Distribution of Betting Odds

3.4 Implementation

Machine learning problems are typically solved within a trial and error process. Usually a first implementation is not sufficient and has to be improved in order to increase its performance. The following three subsections each describe individual interim results. After each of these models we evaluated and changed them in order to increase performance. We chose the following hyperparameters (see section 2.2) for all of the configuration options:

1. **Loss** The classification task: predict the outcome of a football match is a multi-class classification problem where each example belongs to a single class.

Therefore we chose categorical crossentropy[NBJ02] as a loss function (see section 2.2.2).

2. **Optimizer** The Adam-optimizer[KB14] is implemented as the optimizer.

3. **Weight initialization** The weights of the model are initialized by using random uniform ($minval = -0.01$, $maxval = 0.01$) initialization.

4. **Activation function** All of the models use two different activation functions (see section 2.2.1). The hidden layer uses rectified linear units (ReLu) as activation function. The output layer uses the softmax function in order to map the data onto probabilities between 0 and 1.

5. **Additional parameters** An early stopping method, which monitors the loss during training, is implemented to reduce training time and prevent over fitting. The training process is stopped once loss does not decrease anymore. During training the model trains $\frac{len(training\_data)}{1.3}$ steps per epoch.

3.4.1 Model 1 - Complete Data set

Layer type	Output shape	Trainable parameters
D1 (Dense)	8	136
D2 (Dense)	3	27
Total trainable parameters: 163		

Table 3.1: Model 1 - complete engineered features

Model 1 (3.1) takes all of the engineered features (16 features) , except the team names as input. First the model uses a min-max scaler (see section 2.3) to transform all features into values between 0 and 1. Afterwards the model projects the features onto 8 neurons in the hidden layer, resulting in 136 trainable parameters ($16 \cdot 8 = 128$ weights and 8 biases). The hidden layer is connected to the output layer which consists of three parameters. One representing a home team win, one representing away team win and the last representing a draw.

3.4.2 Model 2 - Principal Components

Layer type	Output shape	Trainable parameters
D1 (Dense)	8	40
D2 (Dense)	3	27
Total trainable parameters: 67		

Table 3.2: Model 2 - Principal components

Model 2 (3.2) requires the first four principal components which we derived from our data set. Model 2 projects these 4 principal components onto 8 neurons in the hidden layer, resulting in 40 trainable parameters ($4 \cdot 8 = 32$ weights and 8 biases). The hidden layer is connected to the output layer the same way as Model 1 (3.1).

3.4.3 Model 3 - Team Names and Betting Odds Only

Layer type	Output shape	Trainable parameters
D1 (Dense)	16	864
D2 (Dense)	3	51
Total trainable parameters: 915		

Table 3.3: Model 3 - Only team names and betting odds

Model 3 (see 3.2) uses only team names and betting odds as input features. The betting odds are transformed into variables between 0 and 1 using a min-max scaler (see section 2.3). The team names are one-hot encoded (see section 2.4). Together 53 input features (25 Home teams and 25 Away teams one hot encoded) and 3 betting odds are derived. The input features are mapped onto 16 neurons inside the hidden layer. This results in 864 trainable parameters ($53 \cdot 16 = 848$ weights and 16 biases). The hidden layer is fully connected to the three output classes the same way as the other two models (3.1).

4 Results

This section aims to present the results for the different models as well as the results for different betting strategies. First, the evaluation metrics precision and recall (see section 2.5) are presented in section 4.1. In section 4.2 different betting strategies are analyzed and compared to the usage of the three models from section 3.4. The achieved knowledge was used to create a tool. This tool uses a pre-trained model as back end to predict new outcomes of unseen matches. The tool needs to be provided with team names and betting odds, and returns the outcome as well as the calculated probability for the prediction. A detailed tool manual and how to use all of its features are provided in section 4.3.

4.1 Precision and Recall

In order to provide comparable insight, results for a random picking algorithm are shown in table 4.1. The algorithm picks randomly between the 3 classes. Each class has its distribution over the complete data set as probability for being picked.

	Precision	**Recall**	**F1-Score**	**Support**
Class 0 (Home)	0.46	0.47	0.47	1008
Class 1 (Draw)	0.24	0.25	0.24	652
Class 2 (Away)	0.29	0.28	0.28	546
Macro average	0.33	0.33	0.33	2206
Weighted average	0.36	0.36	0.36	2206
Accuracy	35.86%			

Table 4.1: Results for random choosing

Picking randomly results in weighted average F1-Scores of *0.36*. A recall of *0.25* for class 1 and *0.28* for class 2 means that choosing randomly misses around 75% of the these two classes. Not a single precision value is above *0.5*. This means, that if the algorithm picks a class. The decision is more likely to be wrong than right. Additionally an accuracy of *35.86%* shows, what is to be expected if random classes are chosen. That implies, the algorithm picks the correct class in one third of the time. This means, that 2 out of three predictions are wrong, which is worse than an algorithm that would have only predicted class 1 (accuracy: *45.64%*).

The results for Model 1 - Model 3, shown in table 4.2, are genuinely better. In general Model 1 and Model 2 have almost the same results. This shows, that reducing the feature space from 16 to 4 principal components does not result in a performance drop. In our case the feature reduction had a significant impact on the computation time. While the training for model 1 took **300.08 seconds**, the computation time had almost halved to **172.70 seconds** for training model 2.

Both models (see 3.4.1 and 3.4.2) perform well on the home class. *Four out five*(recall 0.8) possible betting opportunity's on the home team class. In *50%* (precision 0.5) of these cases the algorithm is correct. Both models predict the away team class with a precision of *0.49*. They differ in recall, but only by a margin of 0.06 (*0.42, 0.48*). The performance for the draw class is extremely bad. Even though both models predict the draw class correctly one out of four times, a recall of *0.04* shows that both models are not able to find and predict the draw class.

	Precision			**Recall**			**F1-Score**		
Model	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
Class 0 (Home)	0.52	0.54	0.54	0.81	0.80	0.57	0.64	0.64	0.55
Class 1 (Draw)	0.26	0.27	0.30	0.04	0.02	0.28	0.06	0.04	0.29
Class 2 (Away)	0.49	0.49	0.43	0.42	0.48	0.42	0.45	0.49	0.42
Macro average	0.42	0.43	0.42	0.42	0.44	0.42	0.38	0.39	0.42
Weighted average	0.44	0.46	0.45	0.50	0.52	0.45	0.44	0.45	0.45
Accuracy	Model 1: 50.54%			Model 2: 51.59%			Model 3: 45.33%		

Table 4.2: Results for Model 1-3 (see 3.4)

The biggest difference between model 3 (section 3.4.3) are the recall scores for the home team and the draw class. Recall for the home team class drops to *0.57* while

recall for the draw class increases to *0.28*. Model 3 is better in finding the draw class than the first two models. All other scores are nearly the same between the three models.

Model 3 performed reasonably well for the draw class. This is why we investigated the betting behavior for model 3 further. We trained the model again. This time the model was provided with weights for the different classes. In particular, the draw class received a weight of 3:1. This means that features from class 2 have a three times higher impact on the learning behavior of the model, in comparison to the home and away class.

The results for the altered version of model 3 can be seen in table 4.3. As expected the recall for the class draw almost doubled. Still precision stayed the same. The overall performance only slightly dropped to an F1-score of *0.42* in exchange of the performance boost of the draw class. The model is a bit over tuned for the draw class but in average the draw class has higher betting odds than the other two classes (see section 3.3.

	Precision	**Recall**	**F1-Score**	**Support**
Class 0 (Home)	0.54	0.45	0.49	1007
Class 1 (Draw)	0.29	0.50	0.37	541
Class 2 (Away)	0.46	0.30	0.36	658
Macro average	0.43	0.42	0.41	2206
Weighted average	0.45	0.42	0.42	2206
Accuracy	41.93%			

Table 4.3: Results for M3 weighted

4.2 Profit Results

This section presents charts that show the profit when using the different betting approaches. All profit charts start with 100€ and 3€ is set on each wager.

The first chart (figure 4.1). Shows the profit evaluation if a betting betting customer bets on the different features of the data set. Betting on betting odds, means that the betting customer chooses the smallest betting odd, because the smaller the betting odd the more likely the class is valued by the betting provider. Using this strategy

results in almost the same profit out of the two different betting sites. Although the loss is slightly higher if the betting customer chooses to bet on the website: B365 instead of BWin.

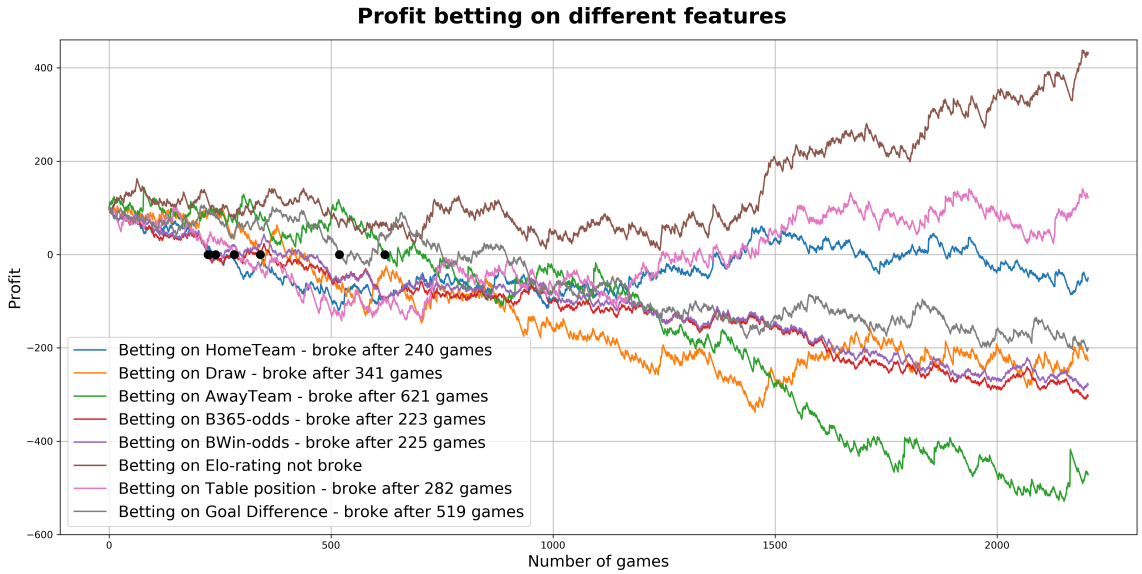

Figure 4.1: Profit when betting on on different features

Using the other betting strategies the betting customer bets on the class with the highest value for the respective feature. All of the strategies, except one, result in financial loss of the betting customer. Even though betting on table position finishes at around 100€, the betting customer already went broke after 282 games. The only betting strategy that resulted in a profit was: **betting on Elo ratings**. For the first 1250 games the profit ranges between +50€ and -70€. After 1250 there is almost a steady profit increase. The Elo rating strategy leaves the betting customer with a profit of more than 300 euros.

All of the following graphs rotate around the same betting strategies but use input from different models. The betting strategies are as listed below:

1. **Betting strategy B1:** Home betting (blue) if the predicted class is the home team class place a bet on the home team. Otherwise don't bet at all.

2. **Betting strategy B2:** Draw betting (green) if the predicted class is the draw team class place a bet on a draw. Otherwise don't bet at all.

3. **Betting strategy B3:** Home betting (orange) if the predicted class is the away team class place a bet on the away team. Otherwise don't bet at all.

4. **Betting strategy B4:** Home and away betting (red) if the predicted class is either the home team or the away team class, place a bet on the predicted class. Otherwise don't bet at all.

Figure 4.2 shows the profit if betting is done randomly. The algorithm that predicts the outcome randomly, simply chooses one of the three classes. The probability of a class C_x being chosen, is as follows:

$$p(C_X) = \frac{|C_x|}{\sum_{i=0}^{n} |C_i|} \tag{4.1}$$

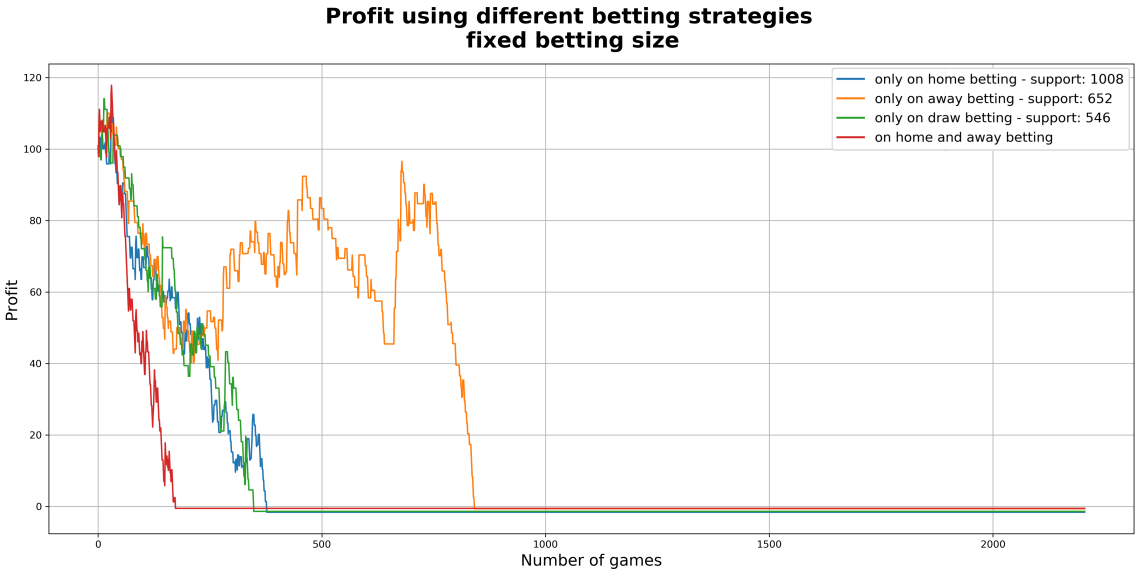

Figure 4.2: Profit when choosing the outcome randomly

The betting customer goes bankrupt using all of the strategies[1]. If the betting customer bets randomly on home team wins and draws, the complete stack of money is lost after around 400 games. Betting on away team wins delays the process of total loss, but after 750 games this strategy also results in a bankruptcy.

[1]here and in the following, bankrupt means, that the betting customer lost the complete amount of capital

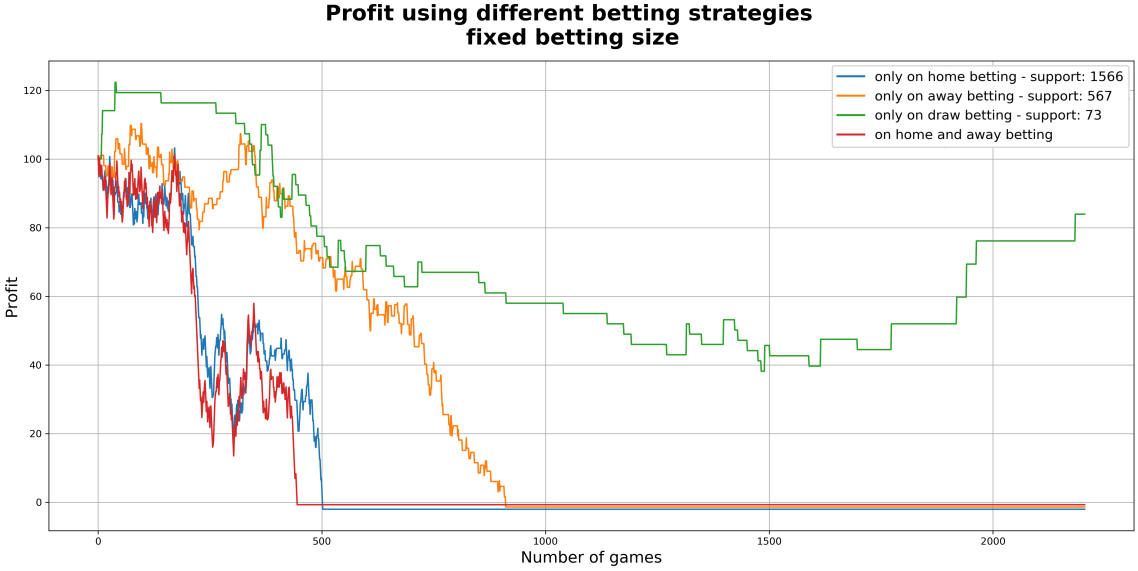

Figure 4.3: Profit when using the predictions of model 1

Model 1 (see figure 4.3) and model 2 (see figure 4.4) both result in bankruptcy for betting strategy B1 and B4. In the case of model 1 being coupled with betting strategy B3 the betting customer goes also bankrupt while coupling B3 with model 2 shows that the betting customer is able to retain 60% of the starting capital (100 euros). Using betting strategy B2 both models are able to stay solvent. While model 1 loses around 20% of the capital, model 2 is able to retain capital. As described in section 4.1 both models have a recall close to 0.00. This ratio is also shown in the graphs. Model 1 places only only 73 times bets for betting strategy B2. The amount of bets placed by model 2 is with 48 even lower.

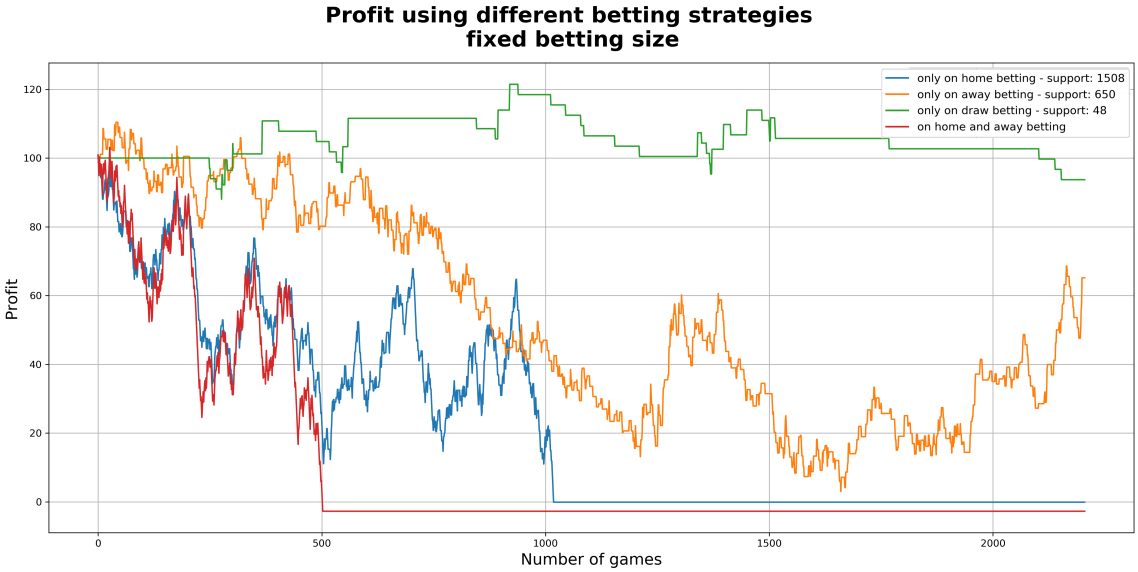

Figure 4.4: Profit when using the predictions of model 2

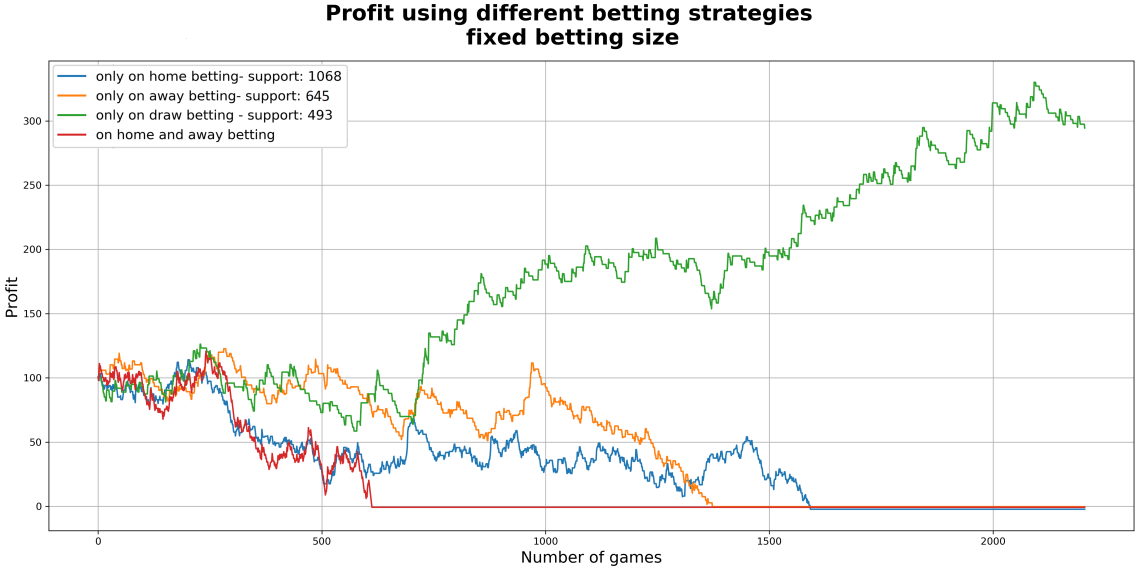

Figure 4.5: Profit when using the predictions of model 3

Figure 4.5 and figure 4.6 picture the profits gathered by model 3. For figure 4.6 the model was trained using weights in order to predict draws more often. Both times model 3 goes bankrupt for betting strategy B1, B3 and B4. Using betting strategy B2, model 3 generates profit. If weights are passed the profit increases as well as the amounts of bets placed. Without weights model 2 places 493 bets, with weights 931 bets are placed. In both cases model 3 is not losing 100 euros or more in a row. The biggest profit drop (-60 euros) occurs after 620 games if weights are provided. This means, that no matter the starting match day a betting customer using model 3 and betting strategy B2 would have always made profit.

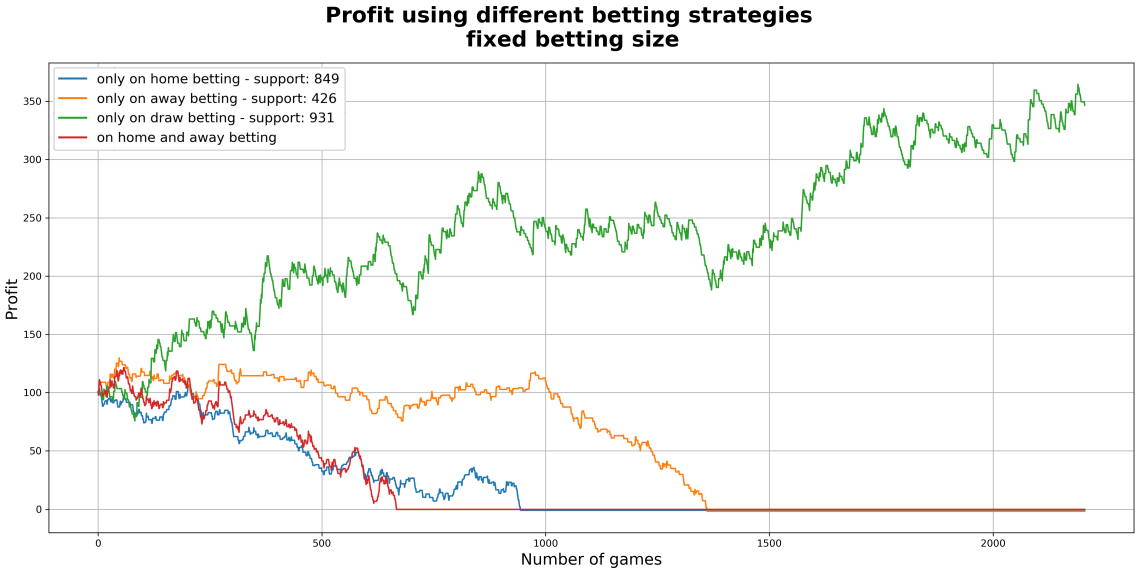

Figure 4.6: Profit when using the predictions of model 3 and weighting the draw class 3:1

The two best and simultaneously the only betting strategies that have generated profit are:

1. **Betting strategy B2**
2. **Betting on the Elo feature**

In order to maximise the profit a new betting strategy is implemented. This betting strategy resembles a combination of the two best performing strategies. If model 3 predicts a draw, the betting customer places a bet on the draw class. Otherwise the betting customer places a bet on the team with the highest Elo rating. The results for this strategy can be seen in figure 4.7. If both strategies are combined, the starting capital increases to 469 euros.

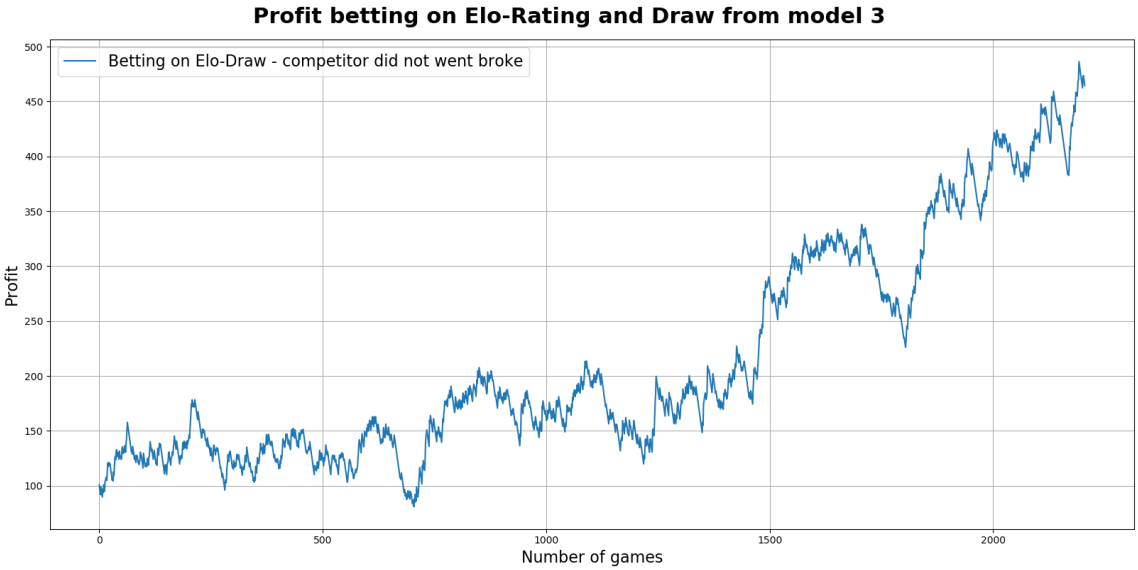

Figure 4.7: Profit when betting on a draw given the case that model 2 predicts a draw. Otherwise bet on the higher Elo rating

Additionally, results were generated for the same predictions using variable betting sizes. Each bet is placed with a betting size that is equal to 3% of the current money stack (e.g. bet 6 euros when the current money stack is 200 euros). The results are similar to the profit charts shown in this chapter and are therefor not discussed further. Although the deviation increases, the basic insight that can be extracted from the plots are the same. All of the variable betting size profit charts can be found in the appendix.

4.3 Betting Predictor Tool

In order to ensure usability of the developed model, a graphical user interface (GUI) was designed to allow user interaction. The trained and serialized MLP program can be used on any machine capable of running Python 3 (64bit) and having following libraries installed:

1. sklearn
2. numpy
3. tensorflow
4. keras
5. h5py
6. joblib

The GUI is shown in figure 4.8. In the first section (1) the user provides team names of the opponents and the odds of the bookmaker for each of the possible outcomes. Clicking on the "Predict" button yields a prediction carried out by the trained MLP. The predicted likelihood of the outcomes are shown after a short calculation period below the odds.

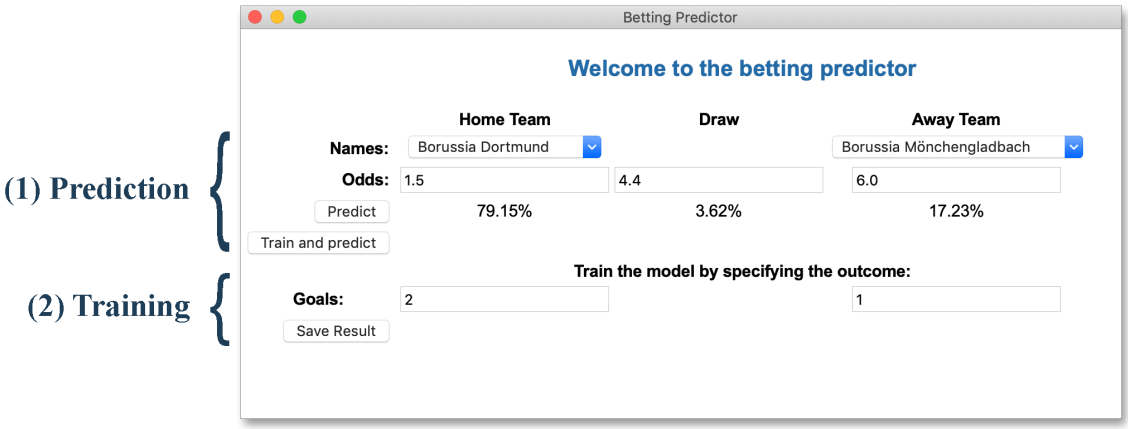

Figure 4.8: Betting Predictor Tool

As the MLP is trained on historic data, new data must be inserted by the user for keeping it up to date. Therefore a CSV file can be stored in the directory of the program, called "additional_data.csv". It must contain the team name columns "HomeTeam" and "AwayTeam", the odds for the outcomes of a home team win "B365H", a draw "B365D" and an away team win "B365A". Furthermore, the MLP needs to know the outcome of the game which is specified in the column

"FTR". The values are 0 for a home team win, 1 for a draw and 2 for an away team win. If additional data is given, the user should click on "Train and predict" in section (1) to retrain the MLP with the combined old and new data sets.

The second section (2) yields a further training step. By providing the goals after a match, the odds of the outcomes and names of the teams are saved into a CSV file. During the next prediction step in the upper section (1) the MLP is retrained using the added data and the prediction is calculated accordingly.

5 Evaluation

This chapter evaluates the results from chapter 4. Section 5.1 aims to evaluate the pros and cons from the different models and describes what the gathered results actually mean and how to interpret them. Section 5.1.1 describes a central problem when dealing with betting classifiers: there is a big difference between high evaluation metrics and making profit. In the last section 5.2 answers to the four research questions from section 1.3 are given.

5.1 Evaluation

	Precision	**Recall**	**F1-Score**
Class 0 (Home)	0.46	1	0.63
Class 1 (Draw)	0	0	0
Class 2 (Away)	0	0	0
Macro average	0.15	0.33	0.21
Weighted average	0.21	0.47	0.29
Accuracy	45.65%		

Table 5.1: Results for only choosing Home Class

Based only on the evaluation metrics precision and recall, all of the models performed better than an algorithm that predicts randomly (see table 4.1) and better than an algorithm that always chooses the home class (see table 5.1). All of the models achieve a weighted average F1-score of *0.45*. In comparison to 0.29 from table 5.1 and 0.36 from table 4.1 and weighted average F1-score of *0.45* substantially better.

For the home team class Model 1 and model 2 both were able to place a bet 4 out of 5 times out of all possible betting opportunities. Both models were right around 50%

of the time. The away team class was predicted with the same precision as the home team class, however, recall fell to 0.42 for model 1 and 0.48 for model 2. Almost every opportunity to bet on a draw was missed by both models. Compared to this predicting the draw class is a big strength of model 3. If we provide the model with weights it is able to place a bet on 50% of all betting opportunities for the class draw additionally the model is correct 30% of the times it predicts a draw (see table 4.3). The average odd for a draw is between 3.8 and 4.2 (see figure 3.2) with only a few single draw odds smaller than 3.0. These odds translate into probabilities ranging from 23.81% ($\frac{1}{3.8}$) till 26.32% ($\frac{1}{4.2}$). The precision from model 3 is higher than the average probability of the draw betting odds indicates that model 3 is able to make money from betting on this strategy.Looking at the profit charts (figure 4.5, figure 4.6) this assumption is proven.

Only betting 3 euros per bet increases the starting money by roughly 200 euros. If the betting customer increases the betting size to 10 euros per bet, the profit would also increase to increase to roughly 660 euros. This would mean that on average the betting customer would win 1.35 euros ($\frac{660}{493}\frac{euros}{bets}$) per bet that is placed. That is a high return of investment. To put that in perspective: the best performing share certificate from the last five years was **Netflix, Inc.** +406.28% [1]. This means that if a person would have invested 100 euros into the betting predictions of model 3 the person would have beat the best performing share certificate of the world by a factor of **1.63**. Please note that this only works for the prediction of the class draw.

5.1.1 Difference Between Good Scores and Increasing Profit

Most of the related work is evaluating the performance of their models only by accuracy. If a betting customer would rely only on this metric substantial statement can be made as to whether he would make money with it at all. Table 4.2 gives the impression that the different models perform considerably well. But if the profit charts 4.3 and 4.4 are evaluated it becomes clear that a high metric score is not a sufficient criteria for profit. The only two betting strategies that resulted in profit were betting on Elo rating and betting on draws with model 3. Model 3 had an F1-Score of *0.29* for the draw class. But because the betting odds on this class are considerably high this strategy generates profit. If both strategies are combined the profit maximized.

[1] https://finance.yahoo.com/quote/NFLX?p=NFLX

5.2 Answers to the Research Questions

This section evaluates the research questions from section 1.3:

Q1 Based on the evaluation metrics, predictions done by any of the described models perform clearly better than always betting on the home team. For a detailed comparison refer to table 5.1 (home team betting) and table 4.1 (model based betting). Based on the profit charts the models perform, with one exception, equally bad for generating profit as the home team betting strategy. Model 3 increases our profit when only betting on draws. Therefore, the prediction of the models perform better than always betting on the home team.

Q2 Evaluating the profit chart 4.1 clarifies that betting on the likeliest outcome based on the quotes of the bookmaker results in drastic losses. As model 3 is able to generate profit, the predictions of model 3 (only this model) perform better than always betting on the likeliest outcome based on the quotes of the bookmakers

Q3 Based on our evaluation, engineered features lead to more accurate predictions. Models that use the engineered features (model 1 and model 2) are more precise and score better when predicting the outcome of football matches, when compared to a model that only uses betting odds and team names (model 3). However, this does not mean, that all of the engineered features help with generating profit. The combination betting strategy B4, using model 3 and the featured Elo rating return the maximum profits. All other engineered features did not provide an increase in profits.

Q4 We found a strategy for maximizing the profit of a betting customer. Train a model on betting odds and team names. Provide the model with weights. In particular weight the draw class three times as high as the other two classes. Use the model to predict outcomes of games. If the model predicts a draw, place a bet on the draw class. Otherwise look at the engineered Elo and place a bet on the team with the highest Elo rating. The results for this strategy can be seen in figure 4.7.

6 Conclusion

This chapter summarizes the gathered information of this work. In section 6.1 the project management is evaluated. Section 6.2 is about additional work that was done over the course of the project but turned out to be not as useful as the other work.

6.1 Project Management

A summary of the work packages can be found in table 6.1. Additionally there is a complete Gantt chart, figure (7.1) in the appendix. Both show, that all work packages and milestone of the first half of the project were met on time. In contrast, both milestones of half two of the project were breached, namely the submission of the final presentation and the submission of the final report. The main reason for this delay was the preoccupation of the project participants with obligations of other modules and work besides university.

Work package	**Due date**	**Submission date**
Research Related Work	10.11.2019	10.11.2019
Research Data Sets	15.11.2019	15.11.2019
Data Cleaning and Preparation	29.11.2019	29.11.2019
Model Implementation and Evaluation	13.12.2019	13.12.2019
Midterm Presentation	20.12.2019	17.01.2020
Optional Tasks and Further Tuning	27.01.2020	24.02.2020
Final Report	10.02.2020	09.03.2020
Final Presentation	24.02.2020	10.03.2020

Table 6.1: Work packages and the submission dates

6.2 Additional Work

It is worth mentioning that additional efforts were made. These efforts turned out to be not as useful as the other work in this work and were therefor not investigated or specifically described in detail.

1. Factor analysis: A dimensionality reduction algorithm was implemented and the outputs were compared with PCA results. The team decided to use PCA rather than FA since PCA introduced new latent variables, the principal components. The relationship between the principal components were investigated to find possible interconnections between the new dimensions.

2. Model 1 threshold tuning: Model 1 was tuned by setting thresholds for probabilities. If the thresh-hold is exceeded the algorithm predicts the output class. A lot more of the classes (high recall) were found when a low threshold was introduced, and a high precision when a big threshold was introduced. Since both directions did not result in profit the effort was dropped.

3. Implementation of decision tree: This was done in parallel to model 1 using the same data set of the mentioned model to crosscheck accuracy. Additionally a decision tree is easily explainable, which a MLP is not. Due to lack of performance we decided not to us a decision tree.

4. MLP parameter optimization: A python program was developed to find the best accuracy upon changing the MLP parameters. This was not used since the computational power needed was high and time consuming.

5. Weather data: Using knowledge of the stadium location of the home team and the time and date of a match, weather data was downloaded and added to the data set. The features included the amount precipitation, the cloud coverage and the temperature. The correlation between the outcome of the match and the weather was too weak and the information were dropped accordingly. However, predicting other values as the total amount of goals scored during a game, these information might be useful. This is subject to further research.

6.3 Comparison with Related Work

C. Herbinet achieved an accuracy of 50.5% and an F1-Score of 0.382 [Her18]. Model 1 achieved an accuracy of 50.54% and model 2 an accuracy of 51.59%. Both models achieved an F1-Score of 0.45. Therefore, each of these two models performed better than the model proposed by Herbinet. He additionally evaluated M. J. Dixon and St. G. Coles approach of using a Poisson regression model [DC97]. Their approach would achieve an accuracy of 0.375 an F1 score of 0.324. Our approach beats those scores as well.

As outlined in section 5.1.1, only because a model achieves good results, its not able to generate profit. None of the related work papers ([Her18], [DC97], [JFN06], [DLMG19]) include an evaluation on whether the models would generate money. Section 5.1 is shows, that betting on all games is not able to generate money. Therefor, we assume that the approaches used in these papers will not be able to increase profits, without a thought out betting strategy.

The data sets provided by other papers were not complete. Therefor a new data set (see section 3.3) is engineered, that is not only complete but additionally includes a lot of new engineered features (see section **??**). Because the models use a different data set it is not possible to give a more detailed evaluation on how the models compare to other models.

6.4 Future Work

Our approach gives a solid foundation for building up future work. First of all it would be interesting to see how the models perform on new data. This new data can include unseen games from the Bundesliga as well as new data from other soccer leagues.

There are many betting strategies that can be tested. These include but are not limited to:

1. Betting on the amount of goals scored at each game.
2. Including the confidence (that is the probability) of the model into betting
3. Betting on multiple games at once

4. Combining model and betting strategy B3 with the so called Fibonacci betting strategy

It is to be investigated if any of these work. Moreover there are features that could be engineered and could potentially increase the betting profit. These are for example, number of injured players, average possession of the ball in the last games and the referee.

Furthermore all of the models can be tuned using automatic hyper parameter tuning, in order to determine the number of neurons, hidden layers and activation functions. A new loss function could be engineered that gets the value of each bet in terms of profit. A model that trains on such a loss function can learn to increase profits instead of minimizing the categorical cross entropy.

6.5 Summary

This project intended to bring together publicly available data, data science and sports betting to maximize the revenue of a betting customer. Machine learning was used to predict the outcome of a football match, which was then used for revenue optimization. The four related research questions were answered and gave the following results:

The predictions by the trained Multi Layer Perceptron (MLP) were evaluated if a betting customer would have taken the prediction as betting recommendation. It could be shown that the different models outperformed the two strategies of betting only on home teams or the team with the lowest bookmaker odds. As shown, one reason for these good results is the use of a certain selection engineered features. Especially the introduced Elo rating was identified as a valuable feature.

Finally, a strategy was proposed, which - other than the other strategies - does not ultimately end in a total loss of money but gains revenue.

Thus, the project could indeed bring together publicly available data, data science and sports betting to maximize the revenue of a betting customer.

7 Appendix

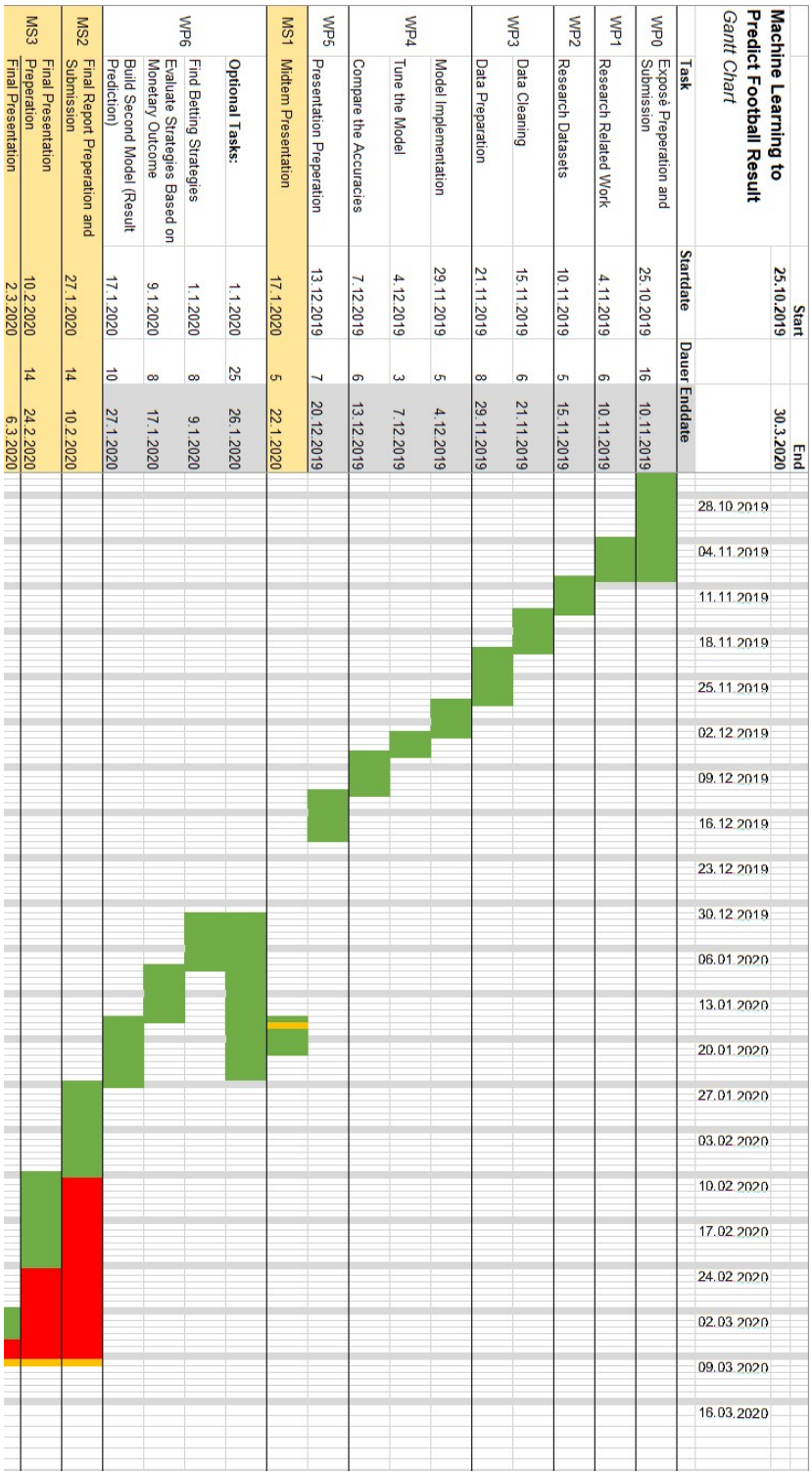

Figure 7.1: The Gantt Chart shows the planned and project milestones and date when the milestone was reached

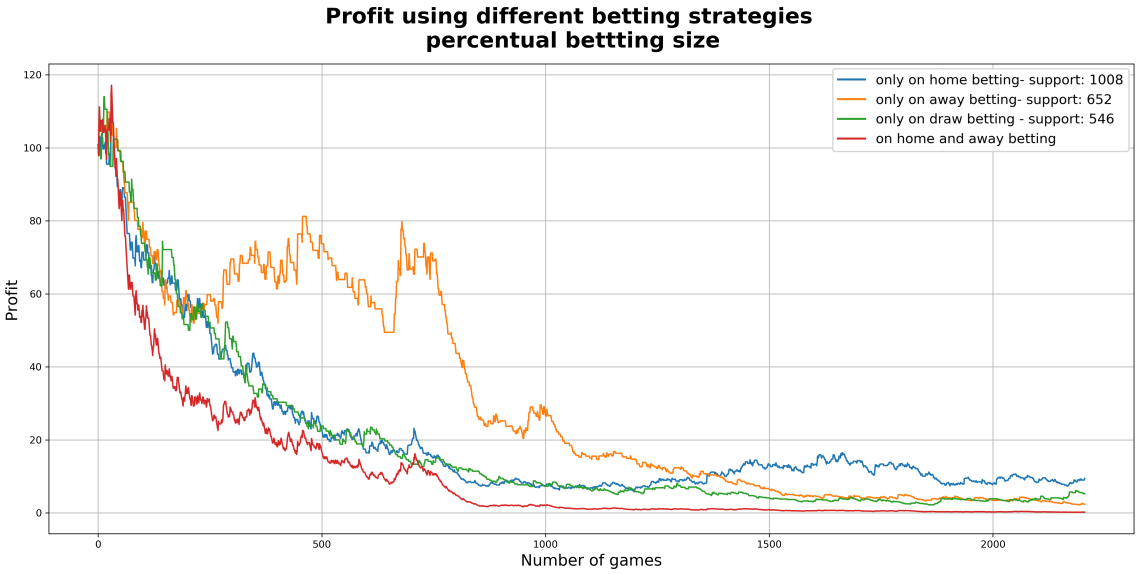

Figure 7.2: Profit when choosing the outcome randomly using variable betting size

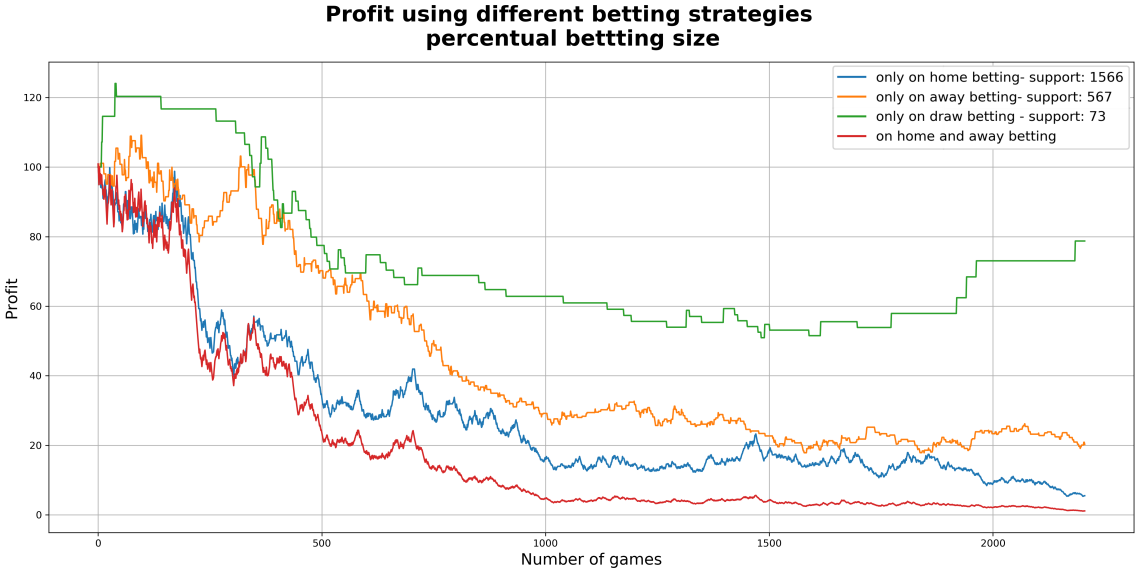

Figure 7.3: Profit when using the predictions of model 1 using variable betting size

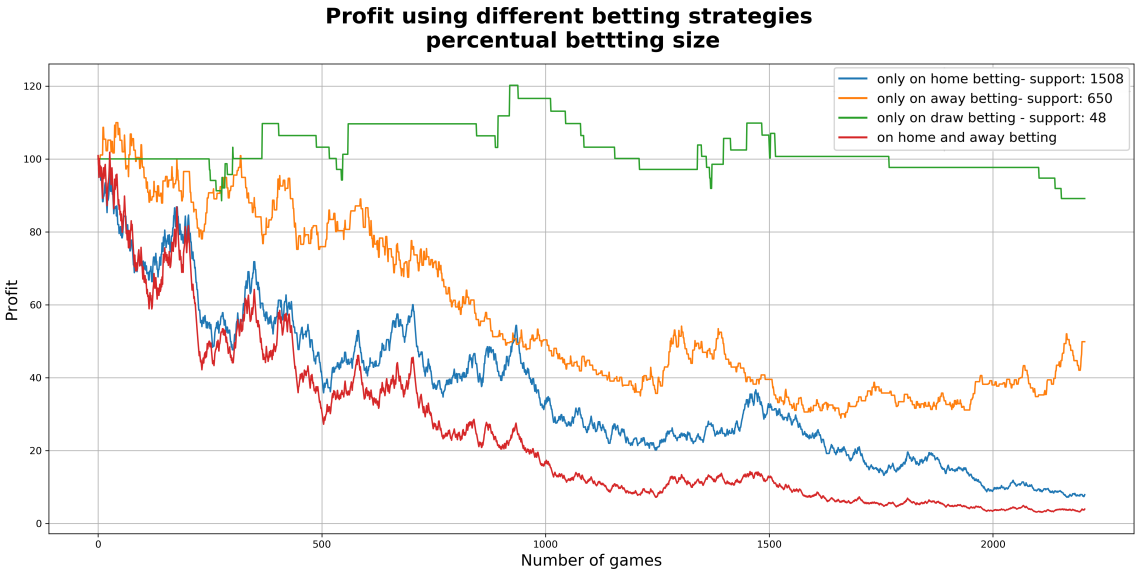

Figure 7.4: Profit when using the predictions of model 2 using variable betting size

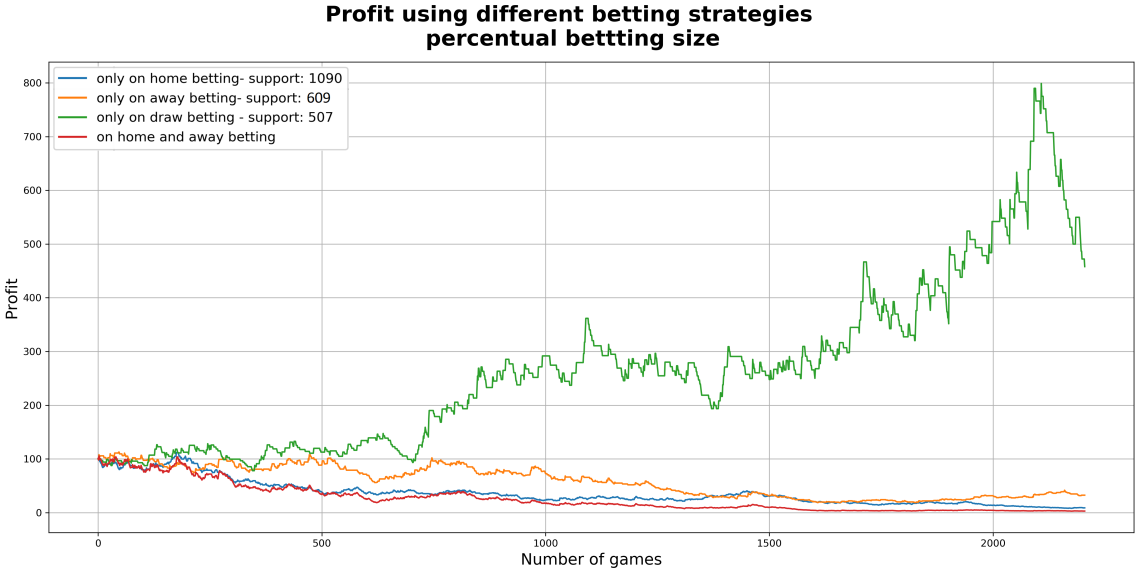

Figure 7.5: Profit when using the predictions of model 3 using variable betting size

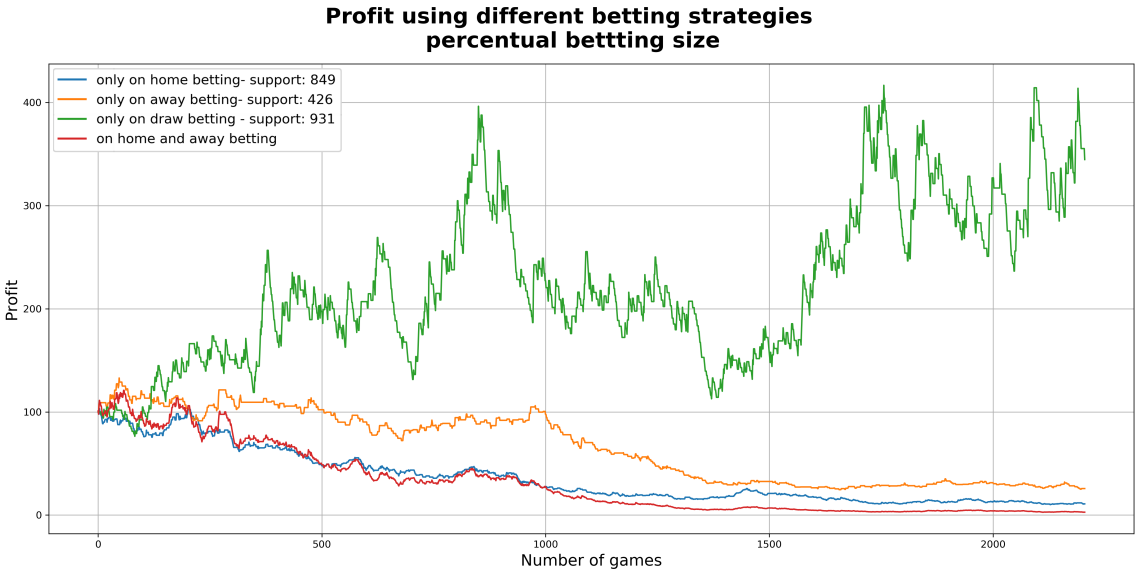

Figure 7.6: Profit when using the predictions of model 3 using variable betting size and providing weights

Bibliography

[Bre] Matt Brems. A one-stop shop for principal component analysis.

[CGW99] Edward M. Condon, Bruce L. Golden, and Edward A. Wasil. Predicting the success of nations at the summer olympics using neural networks. *Computers Operations Research*, 26(13):1243 – 1265, 1999.

[DC97] Mark J. Dixon and Stuart G. Coles. Modelling association football scores and inefficiencies in the football betting market. 1997.

[DLMG19] José Domingues, Bernardo Lopes, Petya Mihaylova, and Petia Georgieva. Incremental learning for football match outcomes prediction. In Aythami Morales, Julian Fierrez, José Salvador Sánchez, and Bernardete Ribeiro, editors, *Pattern Recognition and Image Analysis*, pages 217–228, Cham, 2019. Springer International Publishing.

[Esu15] Emmanuel Esumeh. Using machine learning to predict winners of football league for bookies. *International Journal of Artificial Intelligence*, 5:22, 06 2015.

[Her18] Corentin Herbinet. Predicting football results using machine learning techniques. *IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE*, 1:73, 06 2018.

[JFN06] A. Joseph, N.E. Fenton, and M. Neil. Predicting football results using bayesian nets and other machine learning techniques. *Knowledge-Based Systems*, 19(7):544 – 553, 2006. Creative Systems.

[Kai] Nitin Kumar Kain. Understanding of multilayer perceptron (mlp).

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[Koh95] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[NBJ02] G.E. Nasr, E. Badr, and C. Joun. Cross entropy error function in neural networks: Forecasting gasoline demand. pages 381–384, 01 2002.

[Pur96] M. C. Purucker. Neural network quarterbacking. *IEEE Potentials*, 15(3):9–15, Aug 1996.

[Seg18] Cedric Segeryy. Dissertation title, 2018.

[shaa] Sagar sharma. Activation functions in neural networks.

[shab] Sagar sharma. The fundamentals of neural networks.

[SPW09] Michael A. Smith, David Paton, and Leighton Vaughan Williams. Do bookmakers possess superior skills to bettors in predicting outcomes? *Journal of Economic Behavior Organization*, 71(2):539 – 549, 2009.

[Zoh] Mohamed A. Zohdy. Multilayer perceptron for prediction of 2006 world cup football game.