1 Overview

European Football, or Soccer, has been flooded with money in the past ten years. Nowadays, a few illustrious clubs are increasingly dominant both in their home leagues and in international tournaments (such as UEFA Champions League). In Germany, 11 of the past 15 championships have been won by Bayern Munich, Barcelona and Real Madrid have won 14 out of the past 15 seasons, in Italy Juventus have won 8 of 15 past seasons. The only surprise winner of one of the four big European leagues in the past 15 years is Leicester City, who have been able to secure the title once while the other 14 years there have been only three different teams winning the league. Given the dominance of these few disproportionately rich clubs many people argue that European Football has become more predictable. Using a deep neural network we attempt to predict matches (win, draw, loss) in the four largest European leagues over the past 14 seasons. This is enabled by the increasing amount of data that is collected in each match. Given data that is available before each match, we find that we can predict 54 percent of all results correctly. Using statistics that are collected in the course of each match we are able to find the right result 62 times out of 100.

2 Data

2.1 Source

We use data from Football-Data (2019). The datasets are available per season for many European football leagues and contain detailed statistics on each match. We decided to use data for the past 14 seasons for the German (Bundesliga), English (Premier League), Italian (Serie A) and Spanish (La Liga) league. While some data is available for more than the past 14 seasons, we found that beyond that it becomes somewhat inconsistent (missing values, missing statistics in some leagues). Each dataset includes data on 67 variables[1]. The majority of these variables are betting odds from different providers which we will not use for our predictions.

[1]For a description of the variables refer to: `http://football-data.co.uk/notes.txt`.

2.2 Data Analysis and Processing

We start by analyzing and cleaning the data. We then create additional variables to use as features in the neural net based on the data we have. The code to do so is in jupyter notebook `NN_PrepareData.ipynb`.

The establish a baseline estimate for the probability of a result, we look at the probabilities that the home team wins or loses or that the game ends in a draw. We find that 46.5 percent of games end with the home team winning, 25 percent end in a draw and with a 28.5 percent probability the home team loses. The frequencies of the three different results are roughly equal across the four leagues.

Our cleaned **short dataset** that is ready to be split into a training and testing set contains data on 19′938 matches played in the four leagues over 14 seasons[2]. It contains 28 features which are all available *before* the current match. They are explained in detail in Table 1. The code for preprocessing the data and creating the features is from Darsipudi and Tewari (2019) and adapted to fit our requirements. In theory a model trained on this dataset should allow us to predict the result *before* a match has started.

Additionally, we train a model on our cleaned **long dataset**. It consists of 19′922 matches with data for 40 features. The additional 12 features compared to the short dataset are all statistics which are gathered during the current match we are trying to predict and are available *after* it ended. They are specified in Table 2. These statistics are available only after the match was played, thus one could argue that there is little thrill in predicting games for which the full time result is already known. We try to predict the final result anyways because we want to find how much the performance improves once these detailed game statistics are available for prediction.

[2]The Bundesliga season 05/06 contained many missing values, thus we only use the past 13 seasons for this league.

2.3 List of Features

Table 1: Features of the short dataset

Name	Full Name	Description
FTR	Full Time Result	Either H=home win, D=draw or A=away win. What we try to predict.
HTGS	Home Team Goals Scored	HomeTeam's Nr. of Goals scored in a season up to the current match.
ATGS	Away Team Goals Scored	AwayTeam's Nr. of Goals scored in a season up to the current match.
HTGC	Home Team Goals Conceded	HomeTeam's Nr. of Goals conceded in a season up to the current match.
ATGC	Away Team Goals Conceded	AwayTeam's Nr. of Goals conceded in a season up to the current match.
HTP	Home Team Points	HomeTeam's Points in a season up to the current match.
ATP	Away Team Points	AwayTeam's Points in a season up to the current match.
HM1-5	Home Match 1-5	Home Team Result in past 1-5 matches, one column for each past match.
AM1-5	Away Match 1-5	Away Team Result in past 1-5 matches, one column for each past match.
HTWinStreak3/5	-	Home Team Win Streak 3 games/5 games, one column for 3, 5.
ATWinStreak3/5	-	Away Team Win Streak 3 games/5 games, one column for 3, 5.
HTLossStreak3/5	-	Home Team Loss Streak 3 games/5 games, one column for 3, 5.
ATLossStreak3/5	-	Home Team Loss Streak 3 games/5 games, one column for 3, 5.
HTGD	Home Team Goal Difference	Home Team goals scored (HTGS) - Home Team goals conceded (HTGC).
ATGD	Away Team Goal Difference	Away Team goals scored (HTGS) - Away Team goals conceded (HTGC).
DiffPts	Difference in Points	Home Team points - Away Team points up to current match.
DiffFormPts	Difference in Form Points	Home Team points in past 5 matches - Away Team points in past 5 matches.

Table 2: Additional features of the long dataset

Name	Description
HS	Home Team Shots
AS	Away Team Shots
HST	Home Team Shots on Target
AST	Home Team Shots on Target
HF	Home Team Fouls Committed
AF	Away Team Fouls Committed
HC	Home Team Corners
AC	Away Team Corners
HY	Home Team Yellow Cards
AY	Away Team Yellow Cards
HR	Home Team Red Cards
AR	Away Team Red Cards

3 Model

3.1 Sequential Deep Neural Network

To implement our version of a sequential DNN we use parts of the code described in Wong (2017) and parts from the jupyter notebook `multi-layer-pca-confusion.ipynb` from Lecture 3 of this Deep Learning and Neural Networks course.

Our choice of network parameters such as depth and width of the network, activation function, dropout rate, optimiser (Adam with different learning rates and decays) is primarily result-driven. We iterate over different parameters, train five models based on the same parameters

and take the average prediction accuracy on the test dataset as well as average training and validation loss as a benchmark for the different network configurations. The exact results for all models based on different parameters can be found in `NN_footballmodel.ipynb`.

The best performing model based on accuracy on the test set and training-/validation-loss was a 3-Layer (1 hidden layer, all dense layers) sequential Neural Network with Activation Function: *swish*[3] and softmax, Optimiser: Adam (learning rate: 0.0001, decay: 0.0), Dropout rate: 0.2 (short)/0.3 (long) and Loss: "categorical cross-entropy". Regarding the size of our neural network we found that increasing the width of beyond 100 nodes at any layer had no real benefits we could detect. Furthermore, adding more than 5 layers to our network does not increase accuracy but increases overfitting because the model gets too complex.
With the network described above, we are able to predict 62.5 percent of all results in the test dataset correctly using the long dataset for training and 53.5 percent using the short dataset for training.

Looking at the confusion matrix for the result predictions of our model trained on the short data set (Table 3), we see that we can predict 83 percent of all home team wins correctly. We only get 39 percent of all away team wins right and we are unable to predict any of the draws. A draw is obviously the hardest result to get right because many of the games ending in a draw don't have balanced statistics that would suggest it is a draw.

Table 3: Confusion Matrix for DNN accuracy on short data set.

AwayWin	0.39	0	0.6
Draw	0.23	0	0.77
HomeWin	0.17	0	0.83
	AwayWin	**Draw**	**HomeWin**

With the DNN trained on the long data set with statistics on the *current* match that we are trying to predict, we again get 83 percent of all home wins right as we can see in Table 4. Away wins are now classified correctly with 66 percent probability. Again, our model has trouble predicting draws accurately, we only classify 14 out of 100 draws correctly. Still, this is a big improvement in predicting draws correctly over the model trained on the short data set which

[3] A activation function by Google we discovered on `https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820`.

never got any draws right. This improvement makes sense because with detailed match data some important statistics such as home and away team shots on target may be balanced in a game that ultimately ended in a draw, which helps us predict these results.

Table 4: Confusion Matrix for DNN accuracy on long data set.

AwayWin	0.66	0.8	0.27
Draw	0.3	0.14	0.57
HomeWin	0.13	0.04	0.83
	AwayWin	**Draw**	**HomeWin**

3.2 Simple RNN

To implement our version of a simple RNN we use parts from the jupyter notebook `rnn_simple.ipynb` from Lecture 5 of this Deep Learning and Neural Networks course.

Using a simple RNN layer in combination with two dense hidden layers and a softmax output layer we find that we cannot quite achieve the prediction accuracy on the long data set we got from a simple sequential DNN. However, we can get close at 57.5% accuracy on the test set. When we used a LSTM layer instead of a simple RNN the accuracy was worse.

3.2.1 Dealing with Overfitting

To deal with slight overfitting we experienced with our data on very simple models, we first introduce some amount of dropout. Using a dropout rate of 0.2 (short ds) / 0.3 (long ds) we are already able to prevent overfitting. We feel like we do not need to add bias terms or weight regularisers to the layers to deal with overfitting because the performance is already pretty good. When we added bias and different weight regularisers to our layers the accuracy of the model suffered significantly while both training- and validation-loss grew larger and the difference between the two increases in some cases.

3.3 Possible Improvements

One obvious improvement to our model would be to use more data. Since detailed match statistics are not widely available for seasons earlier than the 2005/06, one could include games

from other leagues for the same 14 seasons we already have. We could either include the second divisions for each of the countries we have data for, or add leagues from new countries. It may be interesting to see if the our model could predict results from these leagues with a similar accuracy, or if the lower-level football played on these leagues would require a retraining.

Keeping the number of matches in our dataset constant, we could try to add more feature columns. An interesting feature would be the away- and home-team starting players' FIFA ranking. For the computer game FIFA each player is given a detailed skill-score each year. Obviously the skill-level of each team or skill-difference between the two teams should be closely linked with the end result of a match. Unfortunately we were unable to find FIFA player scores for the earlier seasons in our dataset. Also the starting line-up for each match isn't available in a structured form anywhere we could find.
Many football result prediction networks we found online also use betting odds as a feature. While these are certainly a very good guideline for the chances of each team, we feel like using them is a bit like cheating. These odds most likely include both wisdom of the crowds, i.e. what results do people bet on, and a sophisticated prediction by the broker's own model.

Lastly, there surely are many tweaks we could do to our neural network structure that would slightly improve the prediction capability. Because there are no clear rules for what parameters and structures work best in our prediction problem, the best approach is to try many different meta-parameters.

References

Darsipudi, K. and Tewari, G. (2019). Code for Conference Paper: predicting the football match winner using lstm model of recurrent neural networks. `https://github.com/krishnakartik1/LSTM-footballMatchWinner`. Github Project, retrieved on June 22, 2019.

Football-Data (2019). Football-Data historical data. `http://football-data.co.uk/data.php`. Website with data, retrieved on June 23, 2019.

Wong, D. (2017). Soccer game prediction using deep learning. http://daweiwong.com/2017/02/23/Soccer/. Blog Post, retrieved on June 22, 2019.