BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

REAL-TIME PREDICTION OF FOOTBALL MATCHES RESULTS

PREDIKCE VÝSLEDKŮ FOTBALOVÝCH ZÁPASŮ V REÁLNÉM ČASE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ALIAKSANDR DRANKOU**
AUTOR PRÁCE

SUPERVISOR **Ing. JIŘÍ HYNEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2020

Brno University of Technology
Faculty of Information Technology

Department of Information Systems (DIFS) Academic year 2019/2020

Bachelor's Thesis Specification

Student: **Drankou Aliaksandr**
Programme: Information Technology
Title: **Real-Time Prediction of Football Matches Results**
Category: Artificial Intelligence

Assignment:

1. Get acquainted with the principles of supervised learning and the problem of real-time prediction. Study existing software, tools or libraries which deal with the problem.
2. Analyze features of football matches which play role in the prediction of their results. Focus on the features which are important during a match. Consider betting odds.
3. Design a software for real-time prediction of football matches. Gather sufficient training set.
4. Implement the designed software.
5. Evaluate the results of predictions for different football leagues. Compare the results with betting odds. Design improvements of the software.

Recommended literature:

- Raschka, S.: *Python machine learning*. Packt Publishing Ltd., 2015, ISBN: 978-1-78355-513-0.
- Coursera: *Machine Learning.* [online]. 2019 [cit. 2019-10-13]. Available at: https://www.coursera.org/learn/machine-learning
- SciKit-Learn Developers: *SciKit-Learn User Guide* [online]. 2019 [cit. 2019-10-13]. Available at: http://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf

Requirements for the first semester:

- Items 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Hynek Jiří, Ing., Ph.D.**
Head of Department: Kolář Dušan, doc. Dr. Ing.
Beginning of work: November 1, 2019
Submission deadline: July 31, 2020
Approval date: October 16, 2019

Abstract

This thesis studies the problem of real-time football matches results prediction. It consists of several steps, including the acquisition of suitable dataset and training of the prediction model. The prediction model is represented by two types of neural networks: feedforward and LSTM recurrent neural network. Different combinations of input features are tested to achieve the best performing model. Both models achieved a classification accuracy of about 67.5%, where feedforward network accuracy starts from 54% at the beginning of the match and achieve 93.54% by the end of the match.

In addition to widely-used metrics such as categorical accuracy and log-loss, each model is evaluated in the simulated betting environment. Experiments within betting evaluation have shown that LSTM can't compete with feedforward network, as in each betting run LSTM network ended up with a balance, dropped by more than 90%. However, the feedforward network achieved an ROI (return on investment) of 0.39% in a betting simulation run with one of the configurations. As a result, a neural network approach, especially the feedforward network, has proved to be quite successful in terms of predicting real-time football matches results. Moreover it allowed to build a profitable betting strategy upon it.

Abstrakt

Tato práce se zabývá problematikou predikce výsledků fotbalových zápasů v reálném čase. Skládá se z několika kroků, včetně získání vhodného souboru dat a trénování predikčního modelu. Predikční model je reprezentován dvěma typy neuronových sítí: dopředné a rekurentní, která je představená LSTM. Různé kombinace vstupních parametrů jsou testovány pro dosažení nejlepšího výkonu modelů, včetně dostupných sázkových kurzů. Oba modely dosáhly klasifikační přesnosti přibližně 67,5%, kde dopředná neuronová síť začíná od přesnosti 54% na začátku zápasu a dosahuje přesnosti 93,54% na konci zápasu.

Kromě široce používaných metrik, jako je kategorická přesnost, každý model je vyhodnocován v simulovaném sázkovém prostředí. Experimenty v rámci hodnocení sázek ukázaly, že LSTM nemůže konkurovat dopředným neuronovým sítím, jelikož v každém sázkovém běhu skončila LSTM s bilancí nižší než o 90%. Dopředná neuronová síť však dosáhla návratnosti investic ve výši 0,39% při provádění simulace sázení s jednou z testovacích konfigurací. Výsledkem je, že neuronové sítě, zejména dopředné, se ukázaly jako docela úspěšné řešení, pokud jde o předpovídání výsledků fotbalových zápasů v reálném čase. Navíc, dopředná neuronová síť může posloužit jako základ pro úspěšnou strategii sázení.

Keywords

Machine learning, artificial neural network, deep learning, sports predictions, football, betting, LSTM

Klíčová slova

Strojové učení, umělá neuronová síť, hluboké učení, sportovní předpovědi, fotbal, sázení, LSTM

Reference

DRANKOU, Aliaksandr. *Real-Time Prediction of Football Matches Results.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Hynek, Ph.D.

Rozšířený abstrakt

Predikce výsledků sportovních událostí je pro mnoho výzkumných pracovníků poměrně náročným a zajímavým úkolem. Tato práce se zabývá metodami pro predikci výsledků fotbalových zápasů v reálném čase a na rozdíl od mnoha dalších studií na téma predikce výsledků sportovních událostí využívá data, která jsou dostupná v reálném čase. K tomu patří jednotlivé statistiky jednotlivých týmů jako počet útoků, střel na bránu apod., dostupné pro danou minutu zápasu. Navíc jsou využity kurzy sázkových kanceláří poskytované pro danou událost.

V dané práci jsou popsané postupy získání, předzpracování a rekonstrukce dostupných dat, které byly nutné pro maximalizaci výsledku trénovaných modelů. Zpracování dat zahrnovalo odstranění chybných či neúplných dat a irelevatních událostí. Důležitou část ve zpracování datové sady hraje rekonstrukce minutových statistik, jelikož data obsahují pouze minutu změny a hodnotu. Statistiky a sázkové kurzy byly rekonstruované na základě dostupných minut tak, že chybějící data pro danu minutu byly doplněny validními daty z předchozí minuty.

Celková architektura systému je reprezentovaná aplikací typu klient/server, kde klient se může dotazovat na aktuálně běžící zápasy a vyhodnocení predikce specifikovaného zápasů.

Samotný server se dotazuje na dvě mikroslužby, kde jedna reprezentuje službu pro přístup k datům a je napojená na API poskytovatele dat, druhá je reprezentovaná natrénovaným modelem, který provádí vyhodnocení jednotlivých zápasů a vrátí rozložení pravděpodobnosti výsledků daného zápasu.

Pro vytvoření modelu na vyhodnocení predikce byly použité dva typy neuronových sítí: dopředné a rekurentní, kde rekurentní síť je reprezentovaná LSTM (*long short-term memory*). Každý model byl vyhodnocen pro různé kombinace vstupních parametrů, kde se ukázalo, že nejvyšší přesnost predikce je zajištěna při využití jak statistik, tak i sázkových kurzů na danu minutu. K tomu byly navíc přidané jednotlivé kurzy ze začátku zápasu.

Experimentování s architekturou jednotlivých modelů a vstupními daty dovolilo dosáhnout nejvyšší přesností 67.81% u dopředné neuronové sítě a 67.46% u LSTM. Následující vyhodnocení modelů v simulaci sázkového prostředí prozradilo, že LSTM model má mnohem nižší výkonnost oproti modelu dopředné neuronové sítě. V nejlepším běhu evaluace modelu bylo dosaženo pozitivní procento návratnosti investic ve výši 0.39%. Při vyhodnocení modelů v sázkovém prostředí se porovnávaly dvě strategie managementu velikostí sázky: fixní sazka 3% z dostupného banku a procento vypočítané pomocí kritéria Kelly, které bylo ještě limitováno na maximální sázku 10% z banku. Ukázalo se, že kritérium Kelly způsobuje sázení větších sum a zvyšuje riziko ztráty velkého objemu při sérii neúspěšných sázek.

Celkově tato práce ukazuje možnosti, které poskytují neuronové sítě v problému predikce sportovních událostí, konkrétně fotbalu v reálném čase, a reprezentuje zajímavý směr průzkumu, který může být využit pro vytvoření úspěšných sázkových strategií.

Real-Time Prediction of Football Matches Results

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Jiří Hynek. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Aliaksandr Drankou
July 31, 2020

Acknowledgements

I would like to thank my supervisor Ing. Jiří Hynek for support and guidance throughout the thesis.

Contents

Chapter 1

Introduction

Football is the most popular sport in the world. There are about 4 billion people globally who consider themselves a football fan.[44] Furthermore, betting is a huge part of the football industry. Many of them are interested in forecasting football matches outcomes to gain profit or make watching the game even more exciting. As a result, football betting takes about 70% of total sports betting market with an estimated worth of 500 billion dollars a year, and it expects to keep growing.[31]

The betting industry has been developed dramatically over the last decades. Growth of the internet and fast digitalization accelerated this process significantly. Websites and mobile apps have replaced traditional betting offices, so gambling has become more accessible than ever. Hundreds and thousands of different events are covered by bookmakers every day, starting from common sports such as football or basketball, and to special events like politics or TV shows. As sports events remain the most popular way to bet on, bookmakers usually provide live streaming and detailed real-time statistics to provide better betting experience and attract more customers, as it very competitive industry. So, as bookmakers cover more matches each year, more data and statistics are being generated. Such a massive amount of data becomes a subject of interest for many researchers and professional gamblers who wants to build prediction models and betting strategies.

Predicting the outcomes of football matches is quite challenging. Many factors could affect the result of the match: teams' starting squads, tactics, physical and psychological conditions of individual players, pitch and weather conditions, and others. Moreover, lots of them are changing rapidly during the match. So, it is challenging for an average gambler to analyze changing statistics and odds fast enough to make successful bets. Besides, many matches take place simultaneously, so a person has to analyze multiple matches at the same time. Moreover, a bettor should remain calm and not let emotions or personal preferences affect betting.

This thesis aims to explore available historical data and apply machine learning algorithms and techniques to build the prediction model, which will be able to make predictions of football matches outcomes in real-time, based on available in-game statistics and bookmakers data. Also, different betting and money management strategies will be compared in terms of maximization of profit from bets.

Chapter 2

Online betting

2.1 Betting

Betting is an activity of predicting events' outcomes and wagering some value on it to get profit. History of betting comes from ancient times as the earliest concrete evidence of betting was dated 2300 bc in ancient China. [7] Nowadays, betting is a complex and developed industry. Bookmakers provide an enormous amount of different events that can be bet, starting from sports such as football or basketball, horse races, and virtual sports, or even politics. In general, any event that may have several possible outcomes, whether sporting or not, could be offered by a bookmaker to make a bet.

Bookmaker, or the bookie, is a person or usually, an organization that sets odds for events' outcomes, accepts bets and pays out winnings. [20] Every possible outcome of the event is defined by corresponding odds, which reflect the probability of a given outcome. Therefore, odds setting is one of the most critical parts of the bookmaker's business.

The odds setting process done by experienced and well-qualified people called odds compilers. They should have a solid understanding of mathematics and statistics and a good understanding of the related field. Using this knowledge and specific software, they need to analyze available data related to the event and carefully set as accurate odds as possible for each of the event's possible outcomes. Furthermore, odds compilers have to consider the latest news and unexpected changes to quickly react and change initial odds. For example, if we consider a football event, the starting squads of teams usually become available about an hour before the match start. However, if it turns out that one of the key players is missing in the starting squad, odds can change significantly.

Typically, the probabilities of all possible outcomes of the event sum up to 100%, but it is not valid for the betting market. To guarantee profits, bookmakers have to include a margin in every odd they provide. For example, consider an event with two equally possible outcomes, say a coin toss. In the fair market, the probabilities of both outcomes are 50%. Odds calculated by dividing the sum of all probabilities (100%) by the probability of the given output, so expected odds are 2.0 for each outcome. However, a bookmaker would offer lower odds, such as 1.9 on each outcome. It means that a profit margin of about 5% applied to odds. Having accepted one unit of value on each outcome, the bookmaker will get a profit of 0.1 units regardless of the result. Profit margin varies depending on the type of outcome. Simpler markets, such as match result, tend to have a lower margin as there is a lower chance of unexpected outcome and bookmaker does not have to put extra effort to keep the book balanced. Hence, margins for such markets are usually below 5%. The most popular events also tend to have a lower margin as there is more information available, and

bookmakers could set more accurate and competitive odds. Thus, the less information is available about the event, the higher the margin, which is applied to odds as bookmaker always tries to stay safe from unexpected loses. [8]

Profit margin helps the bookmaker not worry about the result of the event, but instead about the amount of money placed on each outcome, a balanced book. It means that roughly the same amount of money paid out as winning regardless of the outcome. Sometimes it is quite obvious which outcome would be preferred by most bettors. Therefore bookmaker needs to adjust odds a bit to keep it balanced. More money placed on favorite leads to lower odds, as the bookmaker is trying to minimize payout and reduce the number of such bets, as lower odds are not attractive for bettors. Accordingly, as the odds of the opposite outcome rise, it becomes a better deal for a gambler to give a try and gain more profit. As a result, the bookmaker is guaranteed to gain roughly the same amount in any case and therefore stay in long term profit. [9] [12]

Due to the high profitability, bookmaking is a very competitive industry represented by hundreds of different companies. Each of them is trying to have as many customers as possible to maximize gains. Nevertheless, as most bookmakers provide their services online, potential customers can compare odds across all available bookies and choose the higher ones. Therefore, besides setting accurate odds with included profit margin, they should also be competitive across the market, attracting more customers to make a bet with a given bookmaker.

2.2 Live Betting

With the development of the internet and technologies, most bookmakers provide their services online. Moreover, they provide real-time or so-called live betting so that bets could be placed not only before the event start but also during it. The same rules are applied to live betting, where odds should be set carefully and precisely keeping the balanced book and including profit margin.

However, it is a quite challenging task from a technical perspective, as bookmaker has to update odds in near real-time speeds to reflect what is happening. Moreover, odds compilers by themselves cannot catch such a fast pace and operate in a matter of seconds. Usually, it takes days or even weeks to set pre-match odds accurately enough, but here they do not even close to having this amount of time. Therefore, different complex computer systems used to help them manage to change odds. It follows some of the predefined patterns, which automatically update the odds based on data during the event. Compilers' work boils down to supervising these systems, correct patterns, and respond to unexpected or unusual situations that may occur during the event. As bookmaking is a private business, the main goal is to make a profit. Therefore, the bookmaker dictates all the rules. If he notices that there is a risk of incurring serious losses, odd could be dropped, or the entire betting market could be suspended for a giving event. Otherwise, bets are suspended every time an important or dangerous event occurred. It could be a clear scoring opportunity, goal, penalty, red card, or other important events. Moreover, each time bet placed on the live event, there is some delay, which can be up to 10 seconds, to process and accept a bet. As a result, if the odds change or suspended during this time, the bet will not be accepted. Live odds usually have a larger margin as odds compilers do not so accurately adjust it as pre-match odds, so higher margins compensate possible inaccuracies. [10, 8]

Many bookmakers use other services to get market odds and data feed. Betradar[1]is the market-leading supplier of betting-related data services, including pre-match and live odds. It also offers different betting stimulation tools such as betting widgets[2], live match trackers[3], etc., which increase the level of entertainment and provide customers data they are looking for, ensuring them stay longer and make more bets. [2] Opta[4] provides a wide range of options from historical data and stats to real-time data streams, covering a huge number of competitions from around the world. [5]

Despite available data and information, average punters following their emotions, not logic. The main way to stay in profit in the long term is to find value in bookmakers odds, so the real probability of the event is higher then one reflected in the bookmaker's odds. So basically, punters are constantly competing with skilled and experienced teams of odds compilers. However, they are only humans. Despite available resources, they are still liable to make mistakes. Especially in real-time, where time requirements are tight, and even if a computer program drives it, odds patterns are still manually updated and supervised by a human. Therefore, the main goal of a punter is to find an incorrect line and to bet on these outcomes, which are more likely to happen than the given odds reflect. Furthermore, even if given lines are correct, as it set by odds compilers and algorithms, most of the betting public is not as smart. Therefore if people place bets incorrectly, they could move the odds line and create an edge for smart punters. However, even if incorrect odds are found, it is not guaranteed to win such bet, but the probability of winnings increases. As a result, a good betting strategy and careful money-management could lead to sustainable profit in the long term. [3, 8, 6]

2.3 Related work

Betting is based on predictions of various outcomes, mostly sports events. Furthermore, the sports prediction problem has been quite a popular field of research for a long time. Many studies have tried to build a reliable prediction model with enough accuracy. Most of them utilize pre-match data, such as team performance and statistics over the past matches. Nearly any study has included real-time or in-play data or tried to predict the outcome based on what is happening during the event.

One of the researches who utilized in-play data was Lin (2017) [34]. He was the first who tried to apply real-time data in basketball matches outcome prediction. With selected game-level and player-level features, a logistic regression model was able to predict the correct outcome from 53% accuracy in early stages to 85% on the late stages of the game. Weissbock et al. (2014)[48] designed a neural network model based on in-game statistics of NHL matches resulting with an accuracy of 59.8%. He also found a theoretical upper bound of approximately 62% for single-game prediction in the NHL. Tax and Joustra [47] achieved an accuracy of 54.702% predicting Dutch league football matches with Naive Bayes and Multilayer Perceptron classifiers in combination with PCA (Principal Components Analysis). In 2012, Odachowski [38] attempted to predict upcoming sporting events' outcomes based on changes in bookmaker odds. Using the Bagging algorithm, his model achieved the effectiveness of 70%. Buursma et al. (2011) [19] implemented a system for predicting the results of football matches that beat the bookmakers' odds. He concluded

[1]https://www.betradar.com
[2]https://www.betradar.com/wp-content/uploads/sites/4/2019/11/betting-widgets-betradar-4.png
[3]https://www.betradar.com/wp-content/uploads/sites/4/2019/11/LMT-Plus-Soccer.png
[4]https://www.optasports.com

that with the classifier of accuracy about 55% and right betting strategy, which involves betting on a match only when the probability given by the classifier is higher than the probability given by the bookmakers' odds can lead to a profit at the bookmaker in the long run. Peterson and Nyquist et al. (2017)[40] studied different LSTM architectures of recurrent neural networks for predicting outcomes of football matches. They achieved prediction accuracy starting from 33.35% for many-to-one and 43.96% for many-to-many strategy, with increasing accuracy as longer data sequence becomes available during the time, and reached the maximum of 98.63% for many-to-one, and 88.68% for many-to-many strategy prediction accuracy for full-time data sequence. Goddijn, Moshokovich, and Challa et al. (2018)[24] compared various models, including logistic regression, a 3-layer neural network, and LSTM recurrent neural network to predict outcomes of premier league matches based on historical data. The highest classification accuracy of 51% has been achieved with the 3-layer neural network model.

According to previous studies, it is evident that the prediction of sports events outcomes is quite a difficult task, and the average models' performance is in the range of 55-65%. In most cases, neural networks outperform other machine learning models in terms of prediction accuracy and represent a state-of-the-art approach for sports outcomes prediction.

As a result, several neural network architectures, such as feedforward and LSTM, would be utilized in this thesis to build a successful model for real-time football match outcome prediction.

Chapter 3

Deep learning

3.1 Machine learning

Machine learning is one of the most exciting fields in computer science nowadays. It attempts to create computer programs that can improve their performance on a specific task by learning from experience and data, without being explicitly programmed to perform the task. Even though this field exists for quite a long time, significant popularity has come in recent years with the development of computing power and the emergence of large amounts of data.

Mitchell et al. (1997) [37] provides the following definition of machine learning:
„A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.“ In the context of this definition, the task is usually described as the processing of a given example of data to get the desired output. The most common machine learning tasks include classification, prediction, anomaly detection, etc.

From the perspective of the experience, machine learning algorithms could be categorized as supervised and unsupervised. Unsupervised algorithms used to learn useful properties and find hidden patterns in unlabeled data. For example, dividing large data set into clusters of similar examples, based on some features and similarities in data. On the contrary, supervised learning algorithms learn from labels or targets provided by the teacher. Each data point in the data set is labeled with the target value. Therefore, a supervised learning algorithm's task is to find some correlation in the data point's features and learn to predict the correct label from it. One of the most popular tasks for supervised learning is classification. It could be either binomial, e.g., classifying whether a given email is spam or not, or multi-class, predicting one of the several possible outcomes.

Evaluation of the machine learning model's performance is also a critical part of the overall model's success. Different metrics could be applied to evaluate the quality of a machine learning algorithm. The most common approach is to measure the accuracy of the model or its error rate, reflecting the number of correct and incorrect outputs, respectively. Other machine learning tasks require a different performance metric, which provides a continuous-valued score for each example, e.g., an average log-probability. Moreover, a good performing machine learning model must have the ability to generalize and perform well enough on previously unseen data. It defines if it could be successfully deployed in a real-world environment. Therefore, available data is usually divided into training and testing sets. Training data is used for model training only, while testing set serves for model

evaluation and consists of data examples, which are different from those in the training set. [25]

Machine learning has been proved to be a promising solution in the domain of sports prediction. Many researchers have been trying to utilize different approaches from machine learning to design successful prediction models. Some of them, such as Tax and Joustra [47] and Weissbock [48], achieved relatively high accuracy results in predictions with the use of neural networks. Others usually had a neural networks approach as one of the best-performing, among other machine learning approaches.

3.2 Artificial neural networks

The growth in the popularity of machine learning in recent decades has been primarily associated with increasing computing power and related development of neural networks. However, the fundamental idea of mimicking the human brain appeared back in the 1940s. McCulloch and Pitts et al. (1943) [35] proposed their version of a computational neuron. It refers to biological neuron, a nerve cell consisting of several dendrites, cell body(or soma), and axon. So the single neuron receives signals from other neurons via dendrites. These signals are being processed in soma, where an electrochemical potential is created, so neuron impulses, sending out signals via axon to other neurons. This allows neurons to communicate with each other, perform different computations, and process information.

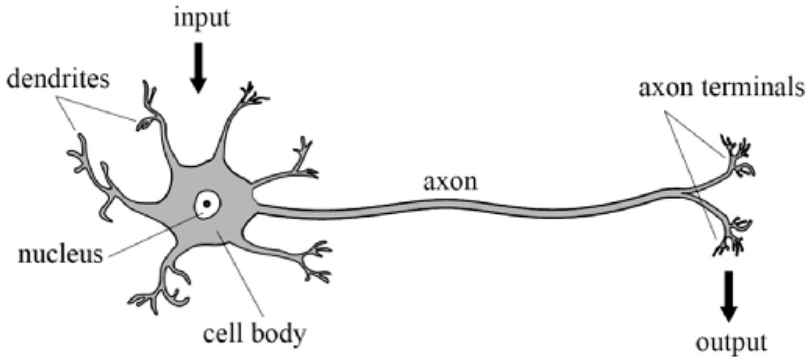

Figure 3.1: Structure of biological neuron.

Later on, in 1958, Rosenblatt [42] designed a perceptron, a simplified model of a biological neuron. So perceptron is a function that maps several inputs to binary value output, mimicking nerve cell's all-or-none behavior. The output is produced by the linear combination of inputs and weights. Therefore, if the resulted sum is larger than some threshold, perceptron outputs 1; otherwise, the output is set to 0. Mathematical representation of the perceptron is the following:

$$f(x) = \begin{cases} 1 & \text{if } w^t * x > 0 \\ 0 & \text{if } w^t * x \leq 0 \end{cases} \tag{3.1}$$

In general, perceptron output is being passed through a non-linear activation function, like a sigmoid or rectified linear unit(ReLU). Therefore, the perceptron is usually represented as a function $f(x)$, such as:

$$f(x) = \sigma(W \cdot x + b) \tag{3.2}$$

where W denotes the matrix of weights, the input feature vector x, bias b and σ is the the non-linear activation function.

The learning algorithm of the perceptron is simple and based on the update of weights and biases in response to an error between the output and the target value for given input and weights configuration. Therefore, the perceptron is a supervised learning algorithm that can learn the right weights and biases for a given input to produce the expected output. However, this model has one notable limitation. It can only solve linear classification problems, where classes could be easily separated with a straight line. So perceptron is only a binary classifier and not able to solve non-linear or multi-class classification problems.

In 1968, Minsky and Papert et al. [36] proved that it is impossible to train a single layer perceptron to learn an XOR function. Moreover, their work has shown that XOR function could be trained with a multilayer perceptron (or MLP). Furthermore, multilayer perceptron itself represents an artificial neural network consisting of at least three layers of nodes. There are one input and one output layer, with one or more hidden layers. It is a fully connected network, so each node in one layer is connected to every node in the following layer.

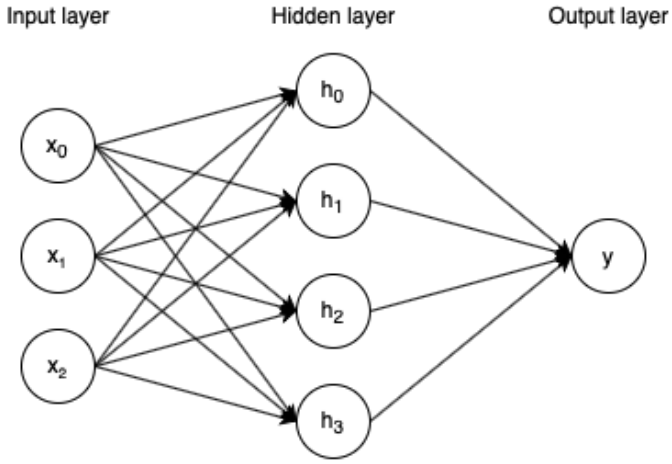

Figure 3.2: Multilayer perceptron with one hidden layer.

Each layer, except the input, contains nodes with non-linear activation function, such as sigmoid or ReLU. With an efficient learning algorithm called back-propagation, which was described by Rumelhart et al. in 1986 [43], a multilayer perceptron is capable of approximating any continuous function. The learning process itself involves repeated adjustments of the connections' weights and biases in the network in order to minimize an error between the actual and desired output of the model. As a result, unlike a single layer perceptron, a multilayer perceptron can distinguish non-linear separable data. The back-propagation learning algorithm and its different modifications remain the dominant approach for the training of neural networks. The use of many hidden layers increases a multilayer perceptron's capabilities and makes it a premise for deep learning.

3.3 Deep learning

After the success of the back-propagation learning algorithm and multilayer perceptron's promising capabilities, neural networks have gained a boost in popularity, which initiated

further development and research of multilayer neural networks. Deep learning itself refers to deep neural networks (DNN), which are multilayer neural networks with more than one hidden later. Moreover, deep learning's fundamental concept is to provide an abstraction above the representation of the complex data. So, DNN allows describing data as a composition of many more straightforward representations of the input. For example, an image recognition process could be divided into several tasks, such as detecting corners, contours, edges, and others, and then leading to the detection of the entire object.

Despite the apparent advantages of larger and more complex neural networks, several technical problems slowed down the development of deep learning. The training of models with lots of hidden layers was very complicated due to insufficient computational power. Innovations in hardware over the last decades and utilization of GPUs (Graphics Processing Unit), which provide parallel computing capabilities, made the training of complex networks possible. With breakthroughs in computational power, the number of data has increased dramatically, allowing to achieve state-of-the-art results in many applications, such as voice recognition, computer vision, and natural language processing. For example, in 2014, a research group at Facebook created a facial recognition system, called DeepFace, that was able to identify human faces in images. A nine-layer neural network was trained on about four million images and achieved 97.35% accuracy, which is as high as the human-level performance [46]. Such success led to even more adoption and popularity of deep learning and inspired many researchers to train deeper and more complex neural networks.

Nowadays, deep neural networks are represented by many architectures, consisting of tens and hundreds of hidden layers with many millions of weights connections, and each of them is particularly successful in specific types of applications. Due to consistently growing computational power, massive data sets, and improvements in optimization algorithms and model design, deep neural networks are expected to become larger and more powerful.

3.3.1 Feedforward neural networks

Feedforward networks are being a fundamental part of deep learning, as a multilayer perceptron represents it. These models are feedforward because of the flow of computation, which is always unidirectional: from inputs, through corresponding hidden layers, and to the output. Therefore, the computational flow could be described as a directed acyclic graph, which does not have any feedback connections. [25]

With genuinely designed network architecture, optimizations, and large enough data, feedforward networks achieve high accuracy in many tasks, including classification and regression problems. Some neural network architectures have been developed for specific tasks. For example, the convolutional neural network(CNN) represents a specific type of feedforward neural network that performs exceptionally well in computer vision field tasks. When a feedforward network is extended with feedback connections, it forms a recurrent neural network (RNN), which is especially successful in processing sequential data, such as text or speech.

3.3.2 LSTM

Long short-term memory network, or LSTM, is an architecture of a recurrent neural network, which was introduced by Hochreiter Schmidhuber et al. (1997) [28]. This architecture design solved long-term dependency problems, which is the inability of recurrent neural networks to learn dependencies in data when sequences become too long, e.g., predicting next word in a giving sentence, based on the context of the sentence, which occurred in

the text way before the actual one. This problem was studied by Bengio et al. (1994) [17], who described several reasons why it might be difficult to learn such dependencies. The fundamental problem is that propagated gradients tend to either vanish or explode while performing on the long sequences. LSTM, on the contrary, provides gradients that neither vanish nor explode and allow them to carry valuable information during long and short sequence processing. Such types of networks belong to the group of RNNs with so-called gated units, which outperform vanilla RNN architectures in the majority of tasks.

LSTM neural network includes several layers consisting of LSTM self-loop cells. A single LSTM cell could be unrolled into the chain-like representation, where the output of the given state is dependent on both the input and the output of the previous state. While there are many structures of single LSTM cell, the most common version consists of several gates and cell states, such as represented in Figure 3.3.

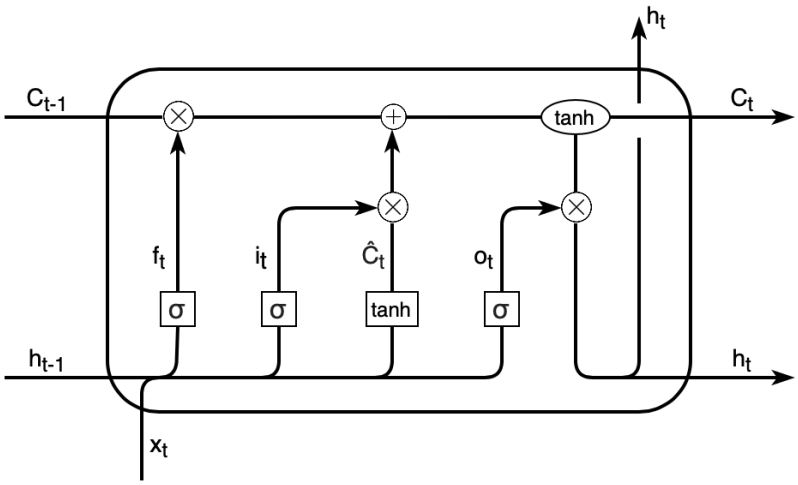

Figure 3.3: LSTM cell structure. (inspired by [11])

Corresponding gate functions work as a sort of control mechanism, which regulates the flow of information inside the cell. All of the gates are perceptrons with corresponding weights and biases that would be learned during the training.

The first gated unit is presented with the „forget“ gate, which decides how much information from the previous cell state would be forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.3}$$

It is computed as a product of weights and result of concatenation between actual input x_t and previous cell output h_t, which is put through a sigmoid activation function σ. It outputs a real value from the interval (0;1). Therefore, if the output is close to 0, the previous cell state would be mostly reset, hence forgotten, and otherwise, if the output of the forget gate is close to 1, the previous cell state would remain almost unchanged.

The „input gate“ i_t performs as a value filter of the new cell state value $\tilde{C}_t$. As well as the „forget“ gate, it is a perceptron with sigmoid activation function

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.4}$$

where the new cell state value is calculated by perceptron with tanh activation function

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.5}$$

The overall cell state is then updated by applying both „forget“ gate output and new cell value on the previous cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.6}$$

When the cell state is updated, the actual output of the LSTM cell could be computed by filtering the cell state C_t, which is already put through tanh function, by the „output gate“ o_t output, in the same way, the „input“ and „forget“ gates do.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.7}$$

$$h_t = o_t * \tanh(C_t) \tag{3.8}$$

Finally, when the LSTM cell output is produced, it is getting passed to the next LSTM cell, and the actual cell state and computational flow continue. [25, 39]

Such architecture of the LSTM cell allows to remember long-term dependencies and improve the performance of recurrent neural networks. It has been proven to be very successful in many applications, such as speech recognition (Graves and Jaitly, 2014[27]), machine translation (Sutskever et al., 2014[45]) and others. In terms of this thesis, LSTM network seems to be a promising solution for processing a time-series data of football matches.

3.4 Software tools

As machine learning popularity increased dramatically over time, so did the number of software tools. Many open-source frameworks have been created to encapsulate implementing details of different algorithms and provide a high-level application programming interface (API) to create a machine learning and deep learning models faster and easier.

Between the most popular software libraries belongs Theano (Bergstra et al., 2010 [18]; Bastien et al., 2012 [16]), PyLearn2 (Goodfellow et al., 2013c [26]), Torch (Collobert et al., 2011b [23]), Caffe (Jia, 2013) [29], Keras (Chollet, 2015) [21], and TensorFlow (Abadi et al., 2015) [15].

Theano

Theano is a math expression compiler and library written in Python, which provides optimized processing of mathematical expressions, especially manipulation with matrices. It could be compiled into optimized CPU and GPU instructions, which boost the processing speed comparing to pure Python. Therefore, it can power data-intensive computations and is being widely used by many researchers. [33]

Torch/PyTorch

Torch is an easy and efficient scientific computing framework developed by Facebook. It is implemented in fast scripting language LuaJIt, with underlying C/CUDA implementation, and has extensive GPU support. It provides high flexibility and speed in building scientific algorithms, coming with large number packages for machine learning, computer vision, signal processing, etc. It allows parallelizing computational graphs over CPUs and GPUs,

increasing algorithms efficiency and speed. Torch is a continually evolving framework widely used by companies such as Facebook, Google, Twitter, and etc.[22]

PyTorch is an open-source deep learning framework based on Torch library, which emphasizes strong GPU acceleration and full capabilities for deep learning. It also was developed by the Facebook AI[1] research team and provides easy-to-use API, which enables fast prototyping and experimentation possibilities, along with high-level efficiency and a rich set of tools for debugging and optimizations. Furthermore, it allows scalable distributed training and is a production-ready solution supported by major cloud platforms. Today, PyTorch is one of the most popular deep learning frameworks that power many companies' projects. For example, Tesla uses PyTorch for vehicle self-driving software. [41]

Caffe

Caffe is a deep learning framework developed by Berkeley AI Research (BAIR) and by community contributors. It is especially fast and efficient in implementations of the convolutional neural networks and computer vision tasks. Both CPU and GPU capabilities could be utilized, and the model could be deployed both on cloud platforms and mobile devices. It makes Caffe a perfect tool for academic research projects, startup prototypes, and many large-scale industrial applications. [29]

Tensorflow

Tensorflow is an end-to-end open-source platform for machine learning developed by the research team at Google. It offers multiple abstraction levels, so both high-level and low-level APIs could be utilized to build and train state-of-the-art models without sacrificing speed or performance. Tensorflow library has extensive support and rich infrastructure for cloud training and deployment, supporting CPUs, GPUs, and even TPUs environments.

It allows us to set a full ML production pipeline with Tensorflow Extended (TFX), enables support for mobile devices and web with TensorflowLite and Tensorflow.js, respectively. Therefore, it suitable for almost any platform and supports multiple client languages, such as C++, Java, and Go. Moreover, as the community of supporters is enormous, there are many guides and tutorials available for different kinds of ML tasks.

It could be effectively used for both research and scalable production deployments. And today, many large companies and industrial leaders, such as Airbnb, Qualcomm, Intel, Twitter, etc., use the Tensorflow library to build or improve their products and services.[4]

Keras

Keras is a deep learning library written in Python and running on top of the Tensorflow machine learning framework. It provides a high-level API of Tensorflow, enabling fast experimentation and an efficient way of developing machine learning solutions. Keras takes advantage of Tensorflow back-end capabilities, which includes high scalability, cross-platform support, and training on large GPU clusters, as well as TPUs, providing extreme computing power for training and inference.

In 2019, Keras was ranked as the #1 framework for deep learning by Kaggle community developers. As of early 2020, Keras has the strong support of community and researchers with almost 400,000 individual users. As a result, Tensorflow and Keras are being the most

[1] https://ai.facebook.com

popular solutions for deep learning nowadays. Ease of use and ecosystem with high-quality documentation and many extensive examples in every machine learning task makes it an excellent tool for both beginners and experienced machine learning researchers and experts. It has been adopted by scientific organizations, such as CERN and NASA. Also, many large companies built their features using Keras library, including Netflix, Uber, Yelp, Instacart, Zocdoc, Square, and many others.[14]

Such a rich set of different frameworks provides the necessary tools for any machine learning task and research. In this thesis, Keras library will be used for fast prototyping and experimentation with different deep learning architectures, and Tensorflow has been chosen for model serving.

Chapter 4

System design

This chapter describes the proposed system design, presents baseline neural network architectures, and describes available data and input features for the corresponding models.

4.1 Architecture

Predicting football matches outcomes in real-time brings certain requirements on the system's performance. It includes both parts of data acquisition and data processing. The system architecture is designed to be simple and robust, as well as effective. It is represented by the client-server application, where the server contains two microservices. Each microservice is an independent service performing a specific function. One is dedicated to data acquisition and requests the necessary information about football events from the data provider's API service. The second one includes a serving machine learning model, in this case, a trained neural network. Its task is to predict one of the three possible outcomes of the football match: home win, draw or away win. Therefore, the model outputs a probability distribution of these outcomes.

Such architecture is easy to maintain, as each service is independent and highly maintainable. It could be containerized and deployed separately, considering different hardware resources and scalability requirements. For example, a machine learning model could be deployed in the environment which includes GPU or other processing units for model performance acceleration. The complete client-server architecture is represented in Figure 4.1.

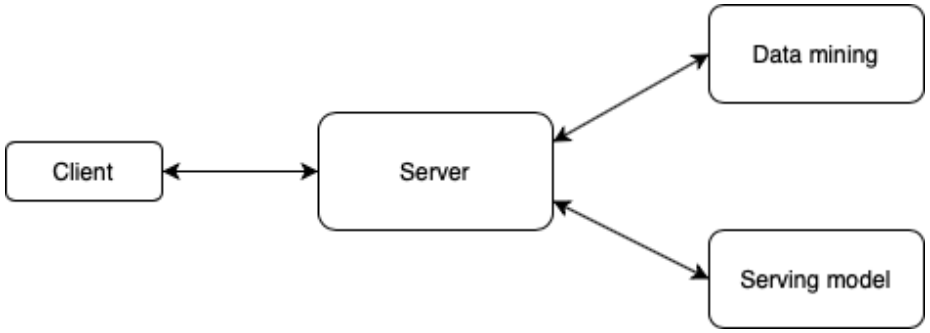

Figure 4.1: Client-server architecture, with two microservices on the server side.

4.2 Input features

Probably, the most important part of the system is a serving model, as it takes care of predictions. The quality of the model is highly dependent on the quality of the available data set and input features. data set for this study was obtained from Betsapi[1] paid service, which provides historical, as well as live, data for many sports via RESTful API.

Available football data includes general information about the event such as teams and league info, event status, etc. Odds data is represented by several odds markets, including full-time result odds, asian handicaps, etc., provided by many bookmakers. Betsapi service allows choosing between many popular bookmakers such as Bet365, WilliamHill, Pinnacle, and others. In this study, bookmaker's odds provided by Bet365[2] are used, as it default bookmaker and is one of the industry leaders with both pre-match and in-play odds. The most important data, in terms of this project, is in-game statistics, which are provided by Betsapi service with a Stats Trend endpoint. It consists of time-value stats for both teams during the match and includes *attacks*, *dangerous attacks*, *shots on target*, *shots off target*, *corners*, *goals*, *substitutions*, *yellow cards* and *red cards*. This set of features, extended with pre-match odds and available odds in the given minute of the event, will be used as an input features vector for predicting models.

4.3 Baseline models

Besides a high-quality data set, the neural network's accuracy depends on the model architecture. It defines an overall structure of the network, its depth, and the width of single layers. In this thesis, two kinds of neural network types are being compared: feedforward neural network and recurrent neural network, represented by LSTM.

Feedforward network

The baseline model of the feedforward neural network consists of 3 fully-connected layers with 16, 8, and 3 hidden units respectively. It accepts a one-dimensional array of features for a given minute of the event as input. Each layer, except the output, has a ReLU activation function, and the output layer has a softmax activation function, which normalizes the output of the network into probability distribution between classes. Also, the baseline model utilizes Adam optimizer [32], which implements an optimization algorithm based on stochastic gradient descent, and cross-entropy loss function.

LSTM

LSTM baseline model, in turn, consists of 3 layers, where the first layer is represented by the LSTM layer with 64 hidden units, followed by a single dropout layer with 0.5 dropout rate and an output layer with 3 hidden units with softmax activation function. An important difference between these models is its input representation. LSTM layer input has to be three-dimensional, where the dimensions define the size of the samples batch, the number of time steps, and the number of features at a given time step respectively. Therefore, available data should be reshaped accordingly to fit these requirements. As in

[1] https://www.betsapi.com
[2] https://www.bet365.com

the case with the feedforward network, Adam optimization algorithm is used, as well as cross-entropy loss function.

Baseline models' architectures have been designed simple as such networks are less likely to overfit and also have a lower training time. Therefore, such a model could provide fast feedback about chosen features in terms of accuracy. After choosing the most successful set of features for given baseline models, experiments with more complex architectures could be done along with hyperparameters tuning to create the model with maximum performance.

4.4 Evaluation metrics

Evaluation is an essential part of building an optimal and successful machine learning model. There are many kinds of evaluation metrics available, which allow measuring the quality of the trained model. These include metrics as classification accuracy, confusion matrix, log loss, and others. However, the use of a single evaluation metric may lead to a poorly performing model, as a model that performs well on one metric does not have to perform well on other metrics. Therefore, multiple evaluation metrics should be used as it ensures more optimal and a better performing model.

In the process of training of proposed neural networks, classification accuracy and log loss metrics are used as the primary evaluation metrics. Classification accuracy refers to the ratio of the correctly classified examples among the total number of examples, and logarithmic loss represents how far is a single prediction from an actual label class.

The entire available data set is divided into training and testing set in the 80/20 ratio. The training set data is shuffled and divided into training and validation sets in the ratio 80/20, and as long as a sample of the data set represents data for a single minute of the event, both training and validation sets have data points refer to the same event. So during the training process, weights are learned from the training set, and a validation step is performed using a validation set. After comparing several input features options, the best performing model is chosen by the highest accuracy and the lowest loss on the validation set. Model's architecture and hyperparameters are later tuned to achieve the best performing version of the model for given input features. The final evaluation of this model is done on the testing set, which consists of previously unseen data and provides information about how well the trained model generalizes.

In addition to traditional machine learning evaluation metrics, one custom metric will be applied to measure the model's prediction capabilities in terms of betting. The betting evaluation includes simulation of a betting environment, where events are chosen randomly from the test set. Bets are made if they meet conditions of the betting simulator and have value. A value bet represents a bet where the probability of the outcome is higher than that reflected by the bookmaker's odds. Such a strategy allows to maximize profit from bets in the long run. [13]

Besides, different stake methods are compared, including percentage stake and the Kelly criterion. The percentage stake refers to staking a fixed percent of the available balance on each bet. Kelly criterion staking method was described in 1956 by Kelly et al. [30], and represents an algorithm for bet sizing, which is proved to lead to higher returns in the long run. The Kelly stake is calculated as follows

$$f = \frac{p * b - 1}{b} \tag{4.1}$$

where p is the probability of the win, and b is the decimal odds provided by the bookmaker. If the Kelly stake equals to zero or negative, then the criterion recommends betting nothing. Otherwise, it represents the percentage of the actual balance to bet.

The overall betting efficiency of the model is represented in return on investment (ROI), representing the expected return for each placed bet.

Chapter 5

Implementation

This chapter describes implementation details of the components designed in Figure 4.1, and training process the serving model.

5.1 Server

Application's server side is implemented in the Go programming language with use of high-performance gRPC framework, which is based on RPC protocol. It provides an easy and efficient way to define a service with Protocol Buffers, supports many platforms and generates an efficient client code for a variety of languages. [1]

The server provides two endpoints: listing all in-play football events and providing prediction for the event, specified by its id. Internally server communicates with two microservices, which ensures general functionality. Data service represents an API wrapper for Betsapi service and provides necessary endpoints for data acquisition. The serving model is ensured by TensorFlow Serving[1] and comes with a Docker container, which contains trained models.

Each microservice is running separately and communicates with server via gRPC API on corresponding port. As server starts, it connects to both microservices as a client and then becomes ready to accept incoming requests on its own specified port.

The client has to generate client code for one of the supported languages from the server's .proto file, which defines all messages and RPC calls. After that, the client is ready to connect and communicate with the server via gRPC methods calls.

5.2 Serving model

Process of creating a serving model includes several steps: data preprocessing, training, hyperparameters tuning, and model evaluation.

Data preprocessing

Quality of data set plays an essential role in the resulting success of the model, as knowledge is learned from provided data. As soon as raw data, retrieved from the Betsapi service, does not meet project requirements, it should be preprocessed before feeding into a neural network. Betsapi football events API returns data from many leagues around the world,

[1] https://www.tensorflow.org/tfx/guide/serving

and, unfortunately, it contains several leagues irrelevant for a presented study, such as beach soccer and e-sports tournaments. A list of inappropriate leagues was created, and such events were filtered out. Besides, some events returned from the Betsapi service had an inconsistent or wrong data, such as missing or abnormal values, and were removed from the data set as well. Input features for the neural networks are represented by data available at the given minute of the event, including event-related statistics and bookmakers odds. However, Betsapi service provides a minute of the event when the change occurred and value itself. Therefore, a full-time, minute-by-minute statistics data has been reconstructed from the available data, so values for missing minutes were filled with values of the previous minute with complete data. If data is missing for the first minute of the event, all features are set to zero. As described in 2.2, bookmakers odds could be suspended or unavailable due to many reasons at the given minute. Therefore unavailable odds in the dataset are replaced with -1. As a result, such time-series reconstruction extends the training data set for the feedforward network, where the state of the match at the given minute represents each data point. Furthermore, it provides the necessary data set for the LSTM network, which has a time sequence representation of the data.

Another critical part of data preprocessing is data normalization. Different data normalization techniques are used to overcome model training problems, improve performance, and reduce training time. Normalized data prevent exploding gradient problems in neural networks and help to converge gradient descents more quickly. In this study, the feature scaling approach used to fit input variables into the lower values range. This step is done with the help of sklearn[2] library and MinMaxScaler estimator, which scales and translates each feature from the training set into the range between zero and one. Scaler parameters for individual features are stored and applied later on both validation and testing set and in a production environment.

The overall clean data set is represented by more than 86306 events distributed between 1067 leagues across 150 countries. Top 10 league distribution is represented in Figure5.1, which shows that friendly matches belong to the most represented group across all leagues.

Corresponding outcome classes are distributed unevenly within dataset with 45.34% of home win matches, 31.57% away win and 23.08% draw. Classes distribution is visualized in Figure 5.2.

After all, the available data set has been split into training and testing set, with 69044 and 17262 matches accordingly. Moreover, the training set has been split again into training and validation with 55235 and 13809 matches. The training set is used to update the model's weights during the training step, while the validation set is used to evaluate the model's performance at the given training step. The testing set consists of matches that are not available during the training process and used to evaluate how well the actual model generalizes previously unseen data. The betting evaluation part is performed on the randomly chosen set of matches from the testing set.

Model training

Both LSTM and feedforward neural networks were implemented in Python programming language with the use of Tensorflow framework and Keras. Several combinations of input features were compared, which represented as follows: 1-stats, 2-stats and minute odds, 3-stats and start odds, 4-stats, minute odds, and start odds.

[2]https://scikit-learn.org/

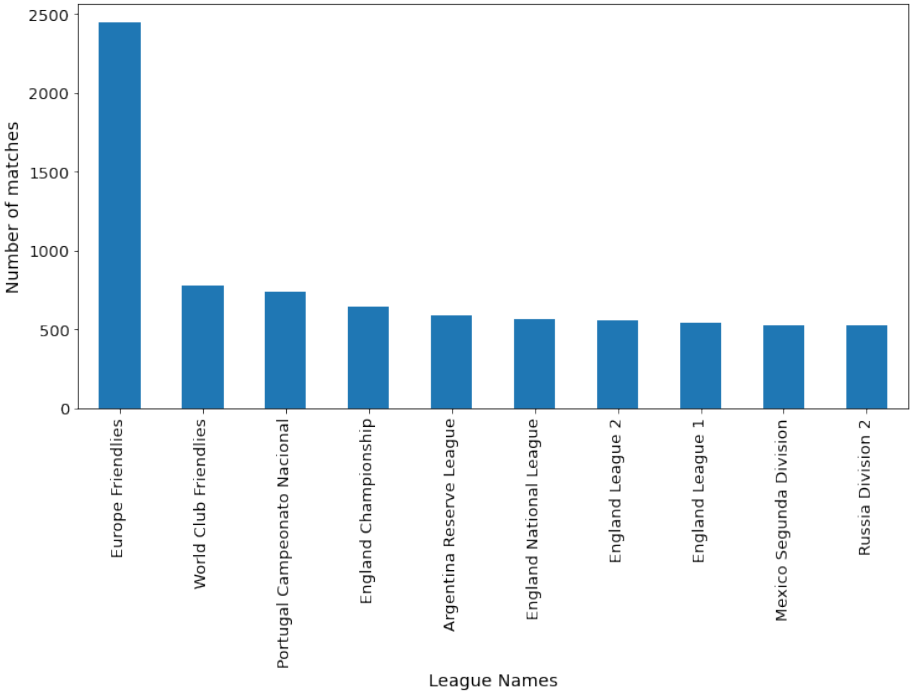

Figure 5.1: Distribution of top 10 leagues in the data set by the number of events.

In the case of a feedforward neural network, the base model training process consists of 20 epochs with a batch size of 64. During the training process, the model with the best validation loss value stored. In Table 5.1, results of training are compared for each input features combination.

Input	Training		Validation		Test	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
1	0.7590	0.6525	0.7420	0.6647	0.7498,	0.6592
2	0.7262	0.6746	0.7108	0.6843	0.7221	0.0.6780
3	0.7306	0.6717	0.7137	0.6831	0.7411	0.6645
4	0.7251	0.6749	0.7104	0.6851	0.7217,	0.6781

Table 5.1: Comparison of the result of training feedforward neural network base models for each combination of input features.

Based on the results of base models training with different input combinations, it is evident that more data ensure higher accuracy and lower loss, including both validation and testing set. Therefore, for further hyperparameters tuning a model with input features representing 4th combination was chosen.

Hyperparameters tuning is done with Hyperas[3] open-source library, which provides an easy and powerful wrapper for experimentation with hyperparameters, such as the number of model's layers and hidden units for each layer, activation functions, optimization algorithm, and batch size. For feedforward neural network optimization, hyperas configuration was set to compare neural networks with two hidden layers, where each layer has one of

[3]https://github.com/maxpumperla/hyperas

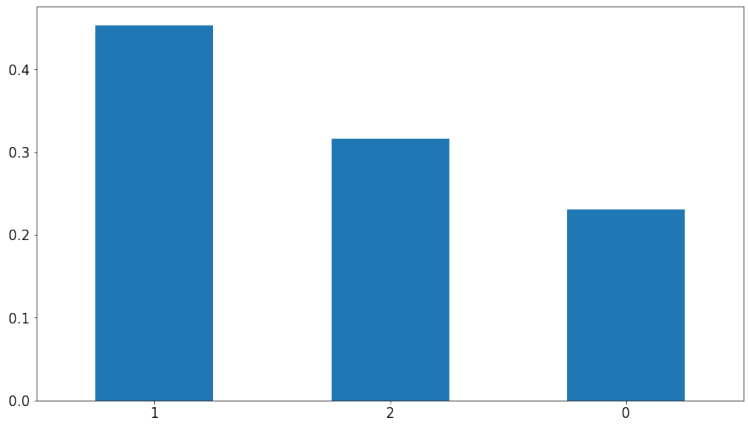

Figure 5.2: Classes distribution over the data set. (1-*home win*, 2-*away win*, 0-*draw*)

the predefined values for the number of hidden units and activation function. The output layer always has an activation function and has three hidden units. The overall model is compiled with one of the predefined optimizers and learning rates, and the training process is done in batches of one of the predefined sizes. Configuration parameters for feedforward neural network are represented in the Table 5.2.

Number of hidden units	8, 16, 32, 64
Activation function	ReLU, sigmoid
Optimizer	Adam, RMSprop, SGD
Learning rate	10^{-3}, 10^{-2}, 10^{-1}
Batch size	32, 64, 128

Table 5.2: Hyperparameters configuration for feedforward neural network optimization.

After 30 different combinations of hyperparameters, a model with best validation accuracy with validation accuracy of 0.6854 and validation loss of 0.7100 was chosen as the best performing one and its architecture is defined as follows 5.3

1st layer	16 hidden units (ReLU activation)
2nd layer	32 hidden units (ReLU activation)
Optimizer	Adam
Learning rate	10^{-2}
Batch size	64

Table 5.3: Architecture and hyperparameters of best-performing model of feedforward neural network.

The training process of LSTM base model was done in batches with 128 data samples within 5 epochs. Low number of training epochs caused by large dataset and high requirements on the resources of the LSTM neural network. Best models of the training process for each of the input features combinations are represented in the Table 5.4

Result of base models training for the LSTM network are very similar to the result of feedforward network architecture. Best performance is achieved with combination of all available data for given minute of the event. Therefore, this model was chosen for

Input	Training		Validation		Test	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
1	0.7680	0.6485	0.7544	0.6603	0.7619	0.6537
2	0.7357	0.6701	0.7250	0.6793	0.7307	0.6716
3	0.7371	0.6687	0.7392	0.6713	0.7392	0.6689
4	0.7323	0.6705	0.7226	0.6796	0.7294	0.6725

Table 5.4: Comparison of the result of training LSTM base models for each combination of input features.

hyperparameters tuning. However, due to the high requirements on the resources and time-consuming training of the LSTM network, architecture optimization and hyperparameters tuning was done manually on the small number of configuration on neural network with two LSTM layers. Following model configuration were compared 5.5

Number of hidden units	32, 64
Optimizer	Adam, RMSprop
Batch size	64, 128

Table 5.5: Hyperparameters configuration for LSTM neural network optimization.

LSTM network trained with Adam optimizer with batches of 128 samples and architecture represented with the first LSTM layer with 64 hidden units and the second LSTM layer with 32 hidden achieved the highest performance with validation accuracy of 0.6848 and loss 0.7118.

Best performing models of both neural network types were later used for the final evaluation in terms of classification accuracy and betting performance.

5.3 Evaluation

Betting evaluation simulates a real-world betting environment, where the event's data changes every minute. The betting simulator has several initialization parameters such as minimum and maximum odds for the bets, staking method, and a value bet threshold. A value bet represents a bet where the probability of the outcome is higher than the probability reflected by the bookmaker's odds. Therefore a value bet threshold represents the difference between two probabilities.

Events for betting evaluation are randomly chosen from the testing set and made up 1000 events. Each event data is sorted by minute in ascending order. Therefore, the algorithm consequently iterating over the available data minute-by-minute and model produces a probability distribution of the event's outcome. These probabilities are converted into odds and compared to the actual odds. A bet is placed if it meets the predefined conditions of the simulator, such as minimum and maximum odds, and the difference between actual and produced odds is higher than a specified value bet threshold. Due to the uneven distribution of classes in the data set and default goal of each team is the win, draw bets are excluded from the betting evaluation.

By default, betting simulator odds range is limited to the range between 1.5 and 2.5 to minimize bets on outcomes with very low probability. Value bet threshold values evaluated starting from 0.1 and up to 0.3 with the step of 0.05.

Moreover, betting simulator includes two types of betting stake: percentage stake, which is set to 3%, and the Kelly stake, which is calculated for each event based on available odds and probability of the corresponding outcome, however, if stake amount suggested by Kelly criterion is limited to 10% to avoid betting of large amounts.

As minutes of the event is ordered naturally and simulates real-world betting environment, the first bet opportunity is used to place a bet, and no more bets could be made for the giving event later. If the outcome is predicted correctly, the stake multiplied by given odds is returned as winning and added to the balance. Otherwise, the bet is lost, and the stake amount is subtracted from the available balance. The starting balance of betting simulator is 100 units. If the actual balance drops lower than 10% of the starting balance, evaluation stops.

Chapter 6

Experiments and results

This chapter presents results and experiments of implemented prediction models.

6.0.1 Classification

Accuracy is representing one of the available metrics of the prediction models and reflects the rate of correct predictions among the total number of prediction for the provided dataset. Result of best-performing models architectures for each type of network is represented in the Table 6.1.

Model	Loss	Accuracy
Feedforward	0.7204	0.6781
LSTM	0.7248	0.0.6746

Table 6.1: Model evaluation for best performing feedforward and LSTM networks.

On the available testing set feedforward network outperformed LSTM model by less than a half percentage, however, classification accuracy for both models is equally successful and could be considered as a relatively good result, taking into consideration fact that average performance of the sports prediction models varies in the range between 55% and 65%.

Moreover, classification accuracy changes during the match, as at the beginning of the event, there is still much time left, however as the match coming to its end, classification accuracy increasing. Corresponding minute-by-minute accuracy changes are represented in Figure 6.1, and accuracies for 15-minutes windows during the match are shown in 6.2

	0m	15m	30m	45m	60m	75m	90m
Accuracy	54.07%	57.20%	60.23%	64.80%	71.06%	80.1%	93.54%

Table 6.2: Accuracy for each 15m window of the event.

6.0.2 Betting evaluation

Another metric utilized in this study is betting simulation. It allows to test trained model in a real-world problem environment and verify how well each model performs, as classification accuracy may not represent an actual value of the model due to uneven representation of the classes in a training set. All bets are placed either on home win or away win, and results of betting evaluation for different configurations are represented in 6.3 and 6.4

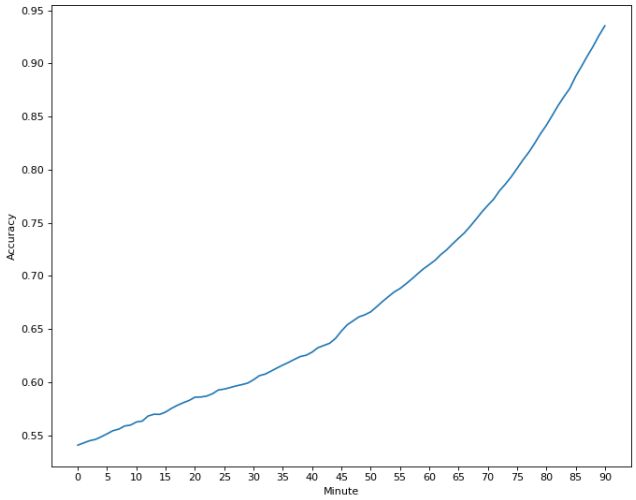

Figure 6.1: Feedforward neural network classification accuracy minute-by-minute.)

Value bet	Stake	N of bets	Accuracy	Lowest balance	Highest balance	ROI
0.1	3%	553	37.97%	77.43(-22.57%)	233.92(+133.92%)	-1%
0.1	Kelly	553	37.97%	11.16(-88.84%)	210.39(+110.39%)	-4%
0.15	3%	528	38.07%	100.00(-0.00%)	311.14(+211.14%)	0.39%
0.15	Kelly	528	38.07%	11.38(-88.62%)	218.49(+118.49%)	-3%
0.2	3%	496	36.29%	84.41(-15.59%)	209.18(+109.18%)	-0.79%
0.2	Kelly	326	36.50%	9.99 (-90.01%)	215.17(+115.17%)	-4.02%
0.25	3%	467	35.33%	65.25(-34.75%)	144.66 (+44.66%)	-2.40%
0.25	Kelly	300	35.67%	9.51(-90.49%)	250.22(+150.22%)	-5.14%
0.3	3%	453	35.10%	58.99(-41.01%)	145.48(+45.48%)	-3.05%
0.3	Kelly	290	35.52%	9.67(-90.33%)	264.26 (+164.26%)	-4.60%

Table 6.3: Feedforward neural network betting evaluation for bookmakers odds in range between 1.5 and 3.5.

Value bet	Stake	N of bets	Accuracy	Lowest balance	Highest balance	ROI
0.1	3%	411	39.42%	9.97(-90.03%)	109.39(+9.39%)	-14.92%
0.1	Kelly	100	40.00%	9.63(-90.37%)	110.97(+10.97%)	-16%
0.15	3%	424	39.62%	9.89(-90.11%)	109.39(+9.39%)	-14.48%
0.15	Kelly	99	40.40%	9.20(-90.80%)	110.97(+10.97%	-17%
0.2	3%	420	39.52%	9.76(-90.24%)	109.39(+9.39%)	-15.17%
0.2	Kelly	99	40.40%	9.38(-90.62%)	110.97(+10.97%)	-16.52%
0.25	3%	98	38.78%	9.91(-90.09%)	113.97(+13.97%)	-16.50%
0.25	Kelly	142	41.55%	9.28(-90.72%)	127.36(+27.36%)	-13.37%

Table 6.4: LSTM neural network betting evaluation for bookmakers odds in range between 1.5 and 3.5.

Results of betting evaluation for both models reveal that the LSTM model has negative ROI and each time betting evaluation has stopped due to significant balance drop. Feedforward network has shown better results with, but the ROI of most runs are negative. However, for a run with a value bet threshold of 0.15, the model performed with 0.39% ROI and remain profitable. Also, Kelly criterion stake limited to the maximum of 10% turned out to be a quite ineffective staking method, as it manipulates with relatively high amounts and bringing more risk into the betting strategy. It results in losing most of the available balance in case of a series of unsuccessful bets.

Chapter 7

Conclusion

In this thesis, two types of neural networks were implemented to predict the outcomes of football matches in real-time. Feedforward neural network achieved a classification accuracy of 67.81% on the testing set with input features represented by an actual minute, stats, minute odds and odds at the start of the match. However, accuracy varies depending on the minute of the event, and starts from 54.07% at the beginning of the match and achieves up to 93.54% accuracy at the last minute of the event. Moreover, the feedforward network proved to be successful in the betting environment with 0.39% ROI (return on investment) in of the configurations.

LSTM neural network achieved similar results in terms of classification accuracy, but it failed in the betting evaluation step, as soon as each betting evaluation run ended up with the balance dropped by more than 90%.

As a result, a feedforward neural network provides the best performance for real-time prediction of football matches and allows to build a profitable betting strategy.

Regarding the success of the implemented models, future improvements could be made to achieve even better results. Pre-match data could be utilized and either produce a separate prediction model or extend an existing one. Also, there is plenty of space for experimentation with different betting strategies and betting markets in general.

Bibliography

[1] *About gRPC*. Available at: https://grpc.io/about/.

[2] *Betting Stimulation*. Available at: https://www.betradar.com/betting-stimulation/.

[3] *Can You Beat the Sports Betting Live Odds Algorithm?* Available at: https://schoolofbets.com/can-you-beat-the-live-sports-betting-odds-algorithm/.

[4] *Case studies*. Available at: https://www.tensorflow.org/about/case-studies.

[5] *Data Feeds*. Available at: https://www.optasports.com/services/data-feeds/.

[6] *Detailed Guide to Live Sports Betting*. Available at: https://www.gamblingsites.org/sports-betting/beginners-guide/live-betting.

[7] *The History of Gambling*. Available at: https://www.gambling.net/history/.

[8] *How Do Bookmakers Calculate Odds, Set Prices And Make Money?* Available at: https://www.onlinebetting.org.uk/betting-guides/how-do-bookmakers-set-odds-and-make-money.html.

[9] *How do bookmakers make money*. Available at: https://help.smarkets.com/hc/en-gb/articles/214180825-How-do-bookmakers-make-money.

[10] *Live betting: The odds are dancing*. Available at: https://www.bookmakers.bet/16669/live-betting-the-odds-are-dancing/.

[11] *Understanding LSTM Networks*. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[12] *Understanding What a Bookmaker Does and How They Make Money*. Available at: https://www.gamblingsites.com/sports-betting/introduction/what-a-bookmaker-does/.

[13] *What is a value bet?* Available at: https://www.pinnacle.com/en/betting-articles/educational/what-is-a-value-bet.

[14] *Why Keras*. Available at: https://keras.io/why_keras/.

[15] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Available at: https://www.tensorflow.org/.

[16] Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. et al. Theano: new features and speed improvements. *CoRR*. november 2012.

[17] Bengio, Y., Simard, P. and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*. february 1994, vol. 5, p. 157–66.

[18] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R. et al. Theano: a CPU and GPU math expression compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy)*. january 2010, Vol. 4.

[19] Buursma, D. Predicting sports events from past results Towards effective betting on football matches. In:. 2011.

[20] Chen, J. *Bookie*. Available at: https://www.investopedia.com/terms/b/bookie.asp.

[21] Chollet, F. et al. *Keras* [https://keras.io]. 2015.

[22] Collobert, R., Kavukcuoglu, K. and Farabet, C. *Torch: a scientific computing framework for LuaJIT* [http://torch.ch]. 2011.

[23] Collobert, R., Kavukcuoglu, K. and Farabet, C. Torch7: A Matlab-like Environment for Machine Learning. january 2011.

[24] Goddijn, S., Moshkovich, E. and Challa, R. A Sure Bet: Predicting Outcomes of Football Matches. 2018.

[25] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[26] Goodfellow, I., Warde Farley, D., Lamblin, P., Dumoulin, V., Mirza, M. et al. Pylearn2: A machine learning research library. *CoRR*. august 2013.

[27] Graves, A. and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. *31st International Conference on Machine Learning, ICML 2014*. january 2014, vol. 5, p. 1764–1772.

[28] Hochreiter, S. and Schmidhuber, J. Long Short-term Memory. *Neural computation*. december 1997, vol. 9, p. 1735–80.

[29] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J. et al. Caffe: Convolutional Architecture for Fast Feature Embedding. *ArXiv preprint arXiv:1408.5093*. 2014.

[30] Jr, J. A New Interpretation of Information Rate. *Information Theory, IRE Transactions on*. october 1956, vol. 35, p. 185 – 189.

[31] Keogh, F. and Rose, G. *Football betting - the global gambling industry worth billions*. Available at: https://www.bbc.com/sport/football/24354124.

[32] Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*. december 2014.

[33] lab, I. *Theano* [http://deeplearning.net/software/theano/index.html]. 2008.

[34] Lin, R. Mason: Real-time NBA Matches Outcome Prediction. In:. 2017.

[35] McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. 1943, vol. 5, no. 4, p. 115–133. Available at: https://doi.org/10.1007/BF02478259.

[36] Minsky, Papert, M. and Seymour. Perceptrons : An Introduction to Computational Geometry. *258 p. : ill.* january 1969.

[37] Mitchel, T. M. *Machine Learning*. March 1997.

[38] Odachowski, K. and Grekow, J. Using Bookmaker Odds to Predict the Final Result of Football Matches. In:. September 2012, p. 196–205.

[39] Olah, C. *Understanding LSTM Networks*. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[40] Peterson, D. and Nyquist, R. Football Match Prediction using Deep Learning. 2017.

[41] PyTorch. *PyTorch at Tesla - Andrej Karpathy, Tesla* [https://www.youtube.com/watch?v=oBklltKXtDE]. November 2019.

[42] Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*. 1958, p. 65–386.

[43] Rumelhart, D., Hinton, G. and Williams, R. Learning Representations by Back Propagating Errors. *Nature*. october 1986, vol. 323, p. 533–536.

[44] Sawe, B. E. *The Most Popular Sports In The World*. Available at: https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html.

[45] Sutskever, I., Vinyals, O. and Le, Q. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*. september 2014, vol. 4.

[46] Taigman, Y., Yang, M., Ranzato, M. and Wolf, L. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In:. September 2014.

[47] Tax, N. and Joustra, Y. Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach. september 2015.

[48] Weissbock, J. Forecasting Success in the National Hockey League using In-Game Statistics and Textual Data. 2015.