FACULTY OF ENGINEERING AND ARCHITECTURE

Predicting Tennis Matches Using Machine Learning

Alexander De Seranno
Student number: 01403449

Supervisors: Prof. dr. ir. Luc Martens, Prof. dr. ir. Toon De Pessemier
Counsellors: Prof. dr. ir. Toon De Pessemier, Kris Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2019-2020

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Predicting Tennis Matches Using Machine Learning

Alexander De Seranno
Student number: 01403449

Supervisors: Prof. dr. ir. Luc Martens, Prof. dr. ir. Toon De Pessemier
Counsellors: Prof. dr. ir. Toon De Pessemier, Kris Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2019-2020

GHENT UNIVERSITY

Preface

First of all, I would like to thank Prof. dr. ir. Luc Martens and Prof. Dr. ir. Toon De Pessemier for giving me the opportunity to work on this research project which is of great interest to me. I also want to express my gratitude for their valuable feedback and guidance throughout the process.

Furthermore, I would like to thank all of my family and friends who have supported me throughout my studies. Finally, my parents deserve a particular note of thanks, providing me with every opportunity to obtain a higher education.

Alexander De Seranno

June 2020

Permission of Use

May 30, 2020

Predicting Tennis Matches Using Machine Learning

by
Alexander De Seranno

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2019-2020

Supervisors: Prof. dr. Luc Martens, Prof. dr. ir. Toon de Pessemier
Counsellors: Prof. dr. ir. Toon De Pessemier, Kris Vanhecke

Abstract

Machine learning has shown promising results in the domain of sport prediction. We propose a machine learning approach to accurately forecast the winner of tour-level ATP singles tennis matches.

Using an open-source dataset containing historical data from all levels of professional tennis, a total of 84 features are extracted based on previous research and data analysis. Our logistic regression model greatly outperforms a baseline based on the official ATP-ranking over logistic loss, accuracy and calibration. Seeking further improvement a single hidden layer neural network further improved logistic loss and accuracy, with only a slight decrease in calibration. Finally the neural network is evaluated over the online tennis betting market using pinnacle closing odds. By iteratively improving the betting strategy, a significant profit of 119% is achieved over three seasons.

Keywords: machine learning, tennis, online sports betting

Predicting Tennis Matches Using Machine Learning

Alexander De Seranno

Supervisor(s): Prof. dr. Luc Martens, Prof. dr. ir. Toon de Pessemier

Abstract—**Machine learning has shown promising results in the domain of sport prediction. We propose a machine learning approach to accurately forecast the winner of tour-level ATP singles tennis matches.**

Using an open-source dataset containing historical data from all levels of professional tennis, a total of 84 features are extracted based on previous research and data analysis. Our logistic regression model greatly outperforms a baseline based on the official ATP-ranking over logistic loss, accuracy and calibration. Seeking further improvement a single hidden layer neural network further improved logistic loss and accuracy, with only a slight decrease in calibration. Finally the neural network is evaluated over the online tennis betting market using pinnacle closing odds. By iteratively improving the betting strategy, a significant profit of 119% is achieved over three seasons.

Keywords—**machine learning, tennis, online sports betting**

I. Introduction

THROUGHOUT the history of sports, people have always been fascinated by the ability to accurately forecast the outcomes of sport events. This human fascination of predicting the future in the context of sports is most apparent in the recent proliferation of online sports betting to a multi-billion dollar industry [1]. Tennis is without a doubt one of the most popular sports in the world. It is played by millions of people and the most prestigious tournaments draw a substantial amount of international attention. This popularity along with the chance to turn a profit in the betting market makes tennis an appealing research subject.

The goal of this thesis is to build on the current state-of-the-art in male professional tennis prediction. Concretely, given information about a tennis match, we want to accurately predict the winning probability of each player in an attempt to turn a profit in the online tennis betting market. Like most research , we restrict our focus to predicting professional ATP singles matches played at the tour-level. Meaning matches played in the main-draw of Grand Slams, ATP Masters 1000 tournaments, and ATP 500/250 Series tournaments.

A. Related Work

The first attempts in tackling the problem of tennis match prediction started in 1999 with the first regression model of Boulier and Stekler [2] to forecast tennis matches based on a player's ATP-ranking. Ever since then, interest has continued to rise, leading many researchers to develop machine learning models of varying complexities and types. Nowadays, most approaches in the literature with the goal of predicting tennis matches fall into one of three categories: point-based, pairwise comparison and regression.

Pairwise comparison is a way of comparing players in pairs trough the use of some underlying metric(s) defining each player. A popular type of pairwise comparison model is the Bradley-Terry model [3] which has been broadly applied in areas such as sports, and machine learning. It allows to predict the outcome of a paired comparison of individuals i and j, given a respective metric π_i and π_j.

$$P(\text{individual } i \text{ beats individual } j) = \frac{\pi_i}{\pi_i + \pi_j} \tag{1}$$

The first application of such model applied directly to tennis was developed by McHale and Morton [4] in 2011.

Point-based models relate to a type of prediction models that in essence only try to estimate the probability of winning a point on serve and return. Assuming that every point is independent and identically distributed (IID) from all other points in the match, the winning probability of each player can be derived in closed-form. Although it was proven that the IID assumption does not hold, researchers argued that deviations are small enough that using a constant probability throughout the match is fine for most purposes [5]. With this in mind, Klaassen and Magnus developed a way to estimate winning probabilities from any point in the match by modeling the tennis match as a discrete-time Markov chain, where each state represents a corresponding score [6]. Having established this model of a tennis match, the only thing that remains to be estimated are the serve point winning probabilities of each player in order to predict a full tennis match.

Contrary to point-based models, regression models try to model the players winning probabilities directly. They achieve this by extracting a set of features describing the characteristics of each player for every match in time. The relationship between these features and winning probability is then learned using a machine learning model capable of probabilistic classification. Regression models differ in the set of features they consider and the used machine learning model. Regression based models perform better than the other approaches when the goal is to predict winning probabilities before the match has started as they do not rely on the false assumption of IID points. On the other hand, point-based have more flexibility when it comes to in-play predictions and can even be used to estimate the chance of a particular score outcome. Since the goal of this work is to predict outcomes before the match has started, regression based methods are the most sensible choice.

II. Dataset

The bulk of the raw tennis match data was collected from an open-source dataset published by Jeff Sackmann on GitHub [7]. The dataset contains all tour-level main-draw matches starting from 1968. The dataset also contains Challenger matches starting starting from 1978 and Futures matches starting from

1991. Even though the goal of this thesis is to predict tour-level matches, match results at a lower level will also be used for training and feature extraction. When a player first competes in a tour-level match, his strength can still be assessed using data from the Futures and Challengers tournaments. In the absence of this data, the model would have to treat every new player as equal and would have to be overly conservative in its predictions. Even the best players spend a few years of their early career in the Futures and Challengers circuit before they are ready for the bigger tournaments.

Matches that were stopped before completion are removed from the dataset. It is not useful that the model learns form these matches, as they are usually the result of injuries or disqualification. In this scenario, the injured or disqualified player's strength is not properly reflected in the result of the match and is thus not useful as learning data for the model. Walkovers are removed from the dataset for the same reason.

A. Feature Extraction

Before features are extracted, the raw Sackmann dataset is sorted by date. This will ensure that each feature extractor will only use data from the past, as using matches from the future would result in data leakage. A brief overview of the extracted features is given below.

The absolute amount of wins and losses of each player are added as features. The matches are divided in three categories according to tournament prestige. Furthermore, the matches are divided by time, to discern matches played in the last semester, year, and of all time. This results in 18 features per player. We choose to not use winning percentage to more accurately present players with a low amount of matches.

Another way to reflect a player's strength is by the use of a rating system. While the official ATP-ranking has some shortcomings when used as a means to reflect current form, rating systems such as Elo have been successfully applied to tennis prediction [8]. For this reason, we used Glicko-2 [9] ratings, which is an improvement over Elo, as features.

Across all levels of tennis, the playing surface can have a sizable influence on the expected outcome of a match. To model this, a one-hot encoding of the surfaces is added along with the absolute amount of wins and losses of each player on every surface.

The other extracted features are related to:

- Home advantage
- Age
- Whether the match is best-of-thee or best-of-five
- Head-to-head record
- Previous success in this tournament
- Recent form / inactivity
- Time

An overview of all 84 extracted features is given in Table I.

TABLE I
EXTRACTED FEATURES

Category	Type	Number of Extracted Features
Wins & Losses	Player	36
Rating Systems	Player	6
Surface	Environment	4
Surface	Player	16
Home Advantage	Player	2
Age	Player	2
Best-Of-X	Environment	1
Head-To-Head	Player	2
Tournament	Player	4
Recent Matches	Player	8
Time	Environment	3
		84

III. Machine Learning Models

Initially, the focus is entirely on making a good predictor of tennis matches without considering the betting market. A better predictor will result in a better betting model provided that the same betting strategy is used.

First, a logistic regression model is fit solely on ATP-ranking as a baseline. Afterwards, We will attempt to beat this baseline with a logistic regression model and a neural network using the extracted features. All models are trained to minimize the logistic loss and are evaluated using the same testing strategy and metrics.

A. metrics

Evaluation metrics are a way to measure and compare the quality of machine learning models. it is often desirable to use multiple evaluation metrics, because some models may score well using one particular evaluation metric, but may perform poorly using another metric. The logistic loss itself, albeit harder to interpret, is also very useful to compare between the models, as minimizing a well chosen loss function should naturally improve the metrics.

The simplest metric for classification is accuracy. This measures the fraction of matches that are correctly categorized as either a win or a loss. Accuracy has the advantage that is the easiest to interpret, but unlike logistic loss, it does not take into account the confidence of the predictions.

Another metric that will be used is calibration. A calibration curve enables us to compare a model's predicted probability to the empirical probability of an event. Naturally, it is desirable for a model's predictions to be reflective of the true underlying probability in a probabilistic classification context.

B. Testing Strategy

To assess the generalization capacity of the trained models, a nested validation strategy is used. Model hyperparameters are only tuned in the inner validation loops, so an unbiased error estimation can be made in the outer validation loop. This nested validation strategy is illustrated in Figure 1

When dealing with time series data, precise care must be taken to avoid data leakage, which would lead to an overly optimistic error estimation. In order to achieve this, the rolling origin validation technique is used, wherein the training set for each testing set simply contains all matches that chronologically occurred before all matches in the test set [10]. Concretely, the outer validation loop splits the dataset in four train-test sets re-

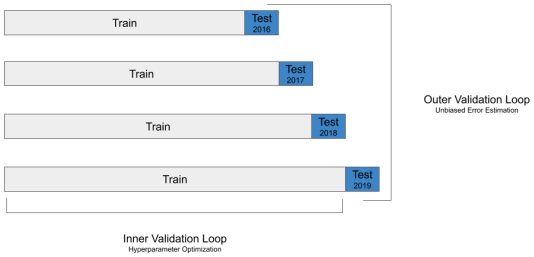

Fig. 1. Testing strategy using nested validation

sulting in four test years: 2016, 2017, 2018 and 2019. Matches played in qualifying, Challengers and Futures tournaments are filtered from the test sets, since we are only interested in predicting tour-level matches. To fairly compare the various models, these train-test sets are kept constant, but the validation strategy used to tune the hyperparameters for each training set may differ between models.

C. Hyperparameter Tuning

The inner loop validation strategy used for the logistic regression model is equal to the outer loop validation strategy. Only two hyperparameters need to be optimized for logistic regression. The L2 regularization parameter λ and whether to train on all available matches in the train set or only tour-level matches which more closely resemble the matches in the test set. These parameters are tuned using grid search. The result can be seen in Table II. It stands out that the model scored better by training only on tour-level matches as opposed to training on all data. This does not mean that the data from these lower-level matches is not used at all, since they are in some way also included in the extracted features of the the tour-level matches.

TABLE II
LOGISTIC REGRESSION TUNED HYPERPARAMETERS

Test Year	$\frac{1}{\lambda}$	All Matches?
2016	1000	No
2017	1000	No
2018	0.494	No
2019	0.869	No

The complexity of neural networks results in quite a bit more hyperparameters to tune compared to logistic regression. It is infeasible to tune every single component of a neural network architecture, so some assumptions are made about the optimal architecture for our problem. Specifically, it is assumed that an architecture with one hidden layer using the rectifier as activation function can lead to satisfactory results. Furthermore, for computational reasons the neural network hyperparameters are only tuned once and kept constant for all test years. To do this, for each candidate of hyperparameters, a network is trained on all matches before 2014 and validated on all tennis matches in 2014 and 2015. The long training time of the neural network makes Bayesian optimization the logical choice to guide the hyperparameter search. The resulting tuned hyperparameters are outlined in Table III. Unlike logistic regression, the neural network performed significantly better when trained on matches from all levels of tournaments. This is probably because the nonlinear decision function defined by the neural network can properly discern and learn from the huge amount of low-level matches without negatively affecting prediction for tour-level matches.

TABLE III
NEURAL NETWORK TUNED HYPERPARAMETERS

Hidden Neurons	Dropout	Batch Size	Epochs	Learning Rate	All Matches?
256	0.45	5696	59	0.00045	Yes

D. Evaluation

The logistic regression models are implemented using scikit-learn [11] and trained using the lbfgs solver. The neural network is implemented using Keras [12] and trained with the Adam optimizer. The logistic loss and accuracy of the trained models averaged over the four test years is depicted in Table IV.

TABLE IV
METRICS AVERAGED OVER ALL MATCHES IN THE TEST SETS

Model	Logistic Loss	Accuracy
Rank Baseline	0.646	64.5%
Logistic Regression	0.597	67.4%
Neural Network	0.592	68.2%

The full logistic regression model substantially outperforms the baseline in both metrics. We can conclude that the extracted features are much more informative than ATP-ranking on its own. In turn, the neural network outperforms logistic regression in both accuracy and logistic loss. While the improvement might seem marginal, it can have a substantial impact in the context of betting.

Finally the calibration plot of the three models is compared in Figure 2. As expected, the full logistic regression model is much better calibrated than the baseline, but also even slightly outperforms the neural network. All three models seem to have a bias of underestimating the winning chances of the underdog.

Overall, the neural network appears to be the best model, as it outperforms logistic regression in two of the three metrics.

IV. RESULTS AND EXPERIMENTS

We now attempt to make the trained neural network profitable in the online betting market by systematically improving the betting strategy. The logistic regression model will not be evaluated to avoid the data snooping bias. Consider for example the case where a high quantity of subpar models are evaluated. Due to some randomness in the betting market, it is likely that we could at least make one of these models profitable. However, it would be unreasonable to assume that this model would be profitable in the real world.

To test the neural network in the online betting market, Pinnacle closing odds of tour-level matches are obtained from Tennis-Data [13]. Using these odds, the online tennis betting market can be simulated. Simulations are run starting from 2016 until the end of 2019, for a total of 10099 matches. Starting from a

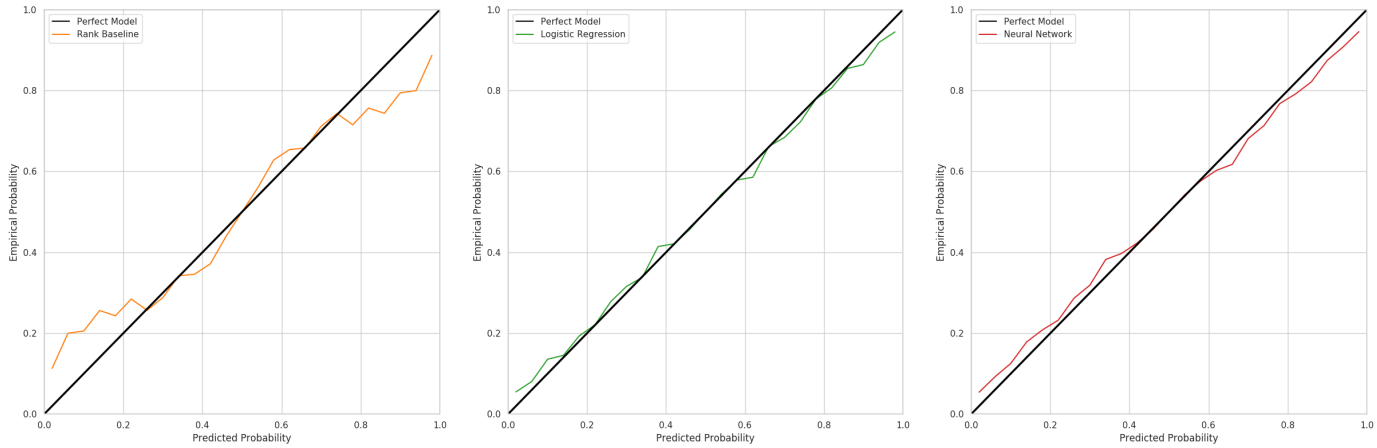

Fig. 2. Calibration curves

bankroll of 1, the ending bankroll for each strategy is shown in Table V.

As a baseline, two static betting strategies are simulated. These strategies are considered static because they follow a static betting rule: always unit (1% of bankroll) betting on the favorite or underdog respectively. As expected, these strategies lose almost all of their starting bankroll over a period of four years.

A first improvement over the static betting strategies is to use the neural network's predictions to decide which player to bet on. If the model's winning probability prediction is higher than the implied probability of the odd, according to the model this bet is profitable, in which case a unit bet is made. While it is a definite improvement over the static betting strategies, it still loses most of the bankroll over the four years.

The main flaw of the unit betting strategy is that the same amount of money is wagered every time, regardless of confidence. The Kelly criterion offers a way so solve this problem. The Kelly criterion is a formula for bet sizing that leads to optimal expected return as the number of bets tends to infinity [14]. Even though the Kelly criterion should in theory outperform the previous strategies, it has the worst results of all of them. This is because Kelly's promise of optimal bankroll growth relies on the assumption that the predictions are equal to the ground truth, which is of course too optimistic since the predictions are only an estimate.

A way to protect the model against its own optimism and reduce the volatility is to only bet a fixed fraction of the amount proposed by the Kelly criterion. This is known as fractional Kelly [15]. Using this approach with a fraction of $\frac{1}{30}$, a profit of 14.2% can be made over four years.

Researchers have attempted to tackle the problem of uncertainty in the predictions trough a more systematic approach. Baker and McHale [16] propose a modified Kelly criterion based on the error variance of the predictions instead of the ad-hoc method used in fractional Kelly, which has no theoretical basis. Instead of only betting a fraction of the Kelly criterion, the bets sizes are shrunk proportionally to the uncertainty of the predictions σ^2. The authors acknowledge it is very hard to directly quantify an optimal σ^2 theoretically, so instead we run simulations on matches from 2016 and select the σ^2 that leads to optimal bankroll growth. Under the assumption that the uncertainty of the model stays roughly the same, the selected σ^2 should also lead to good results in matches from 2017-2019. This process of selecting σ^2 is shown in Figure 3. Indeed we find that the optimal σ^2 for matches in 2016 is close to the optimum for the rest of the matches. The resulting simulation leads to a profit of 119%.

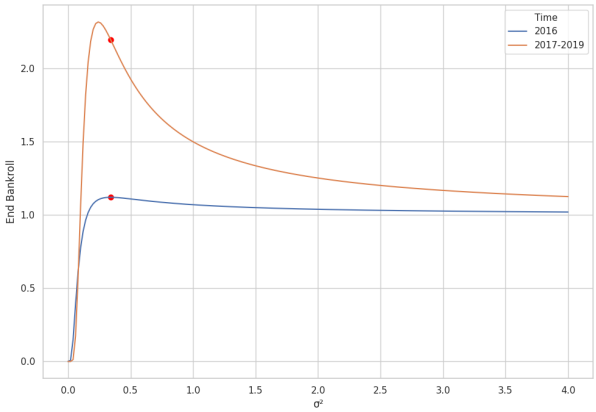

Fig. 3. Ending bankroll as a function over σ^2

TABLE V
ENDING BANKROLLS FOR ALL BETTING STRATEGIES

Strategy	End Bankroll
Underdog Unit Betting	0.002
Favorite Unit Betting	0.045
Neural network Unit Betting	0.324
Kelly Criterion	0.000
Fractional Kelly	1.14
Kelly Shrinkage	2.19

V. Conclusion

Using open-source data from all levels of professional tennis, an extensive list of features were extracted based on previous research and data analysis. Afterwards a logistic regression model was trained on these features and tested over four seasons significantly outperforming the baseline based on the official ATP-ranking over all metrics: accuracy, logistic loss, and calibration. Seeking further improvement a single hidden layer neural network was trained and compared to the logistic regression model. We found that the neural network further exceeded the performance in logistic loss and accuracy, with only a slight decrease in calibration. For our purpose of predicting tour-level matches, logistic regression attained better results by only training on tour-level matches. On the other hand, the neural network performed better when trained on matches from both tour-level and lower level tournaments.

Finally, the neural network's predictions are made profitable in the online betting market by systematically improving betting strategies. Ultimately, the best results were obtained by applying the bet sizing model proposed by Baker and McHale [16] which shrinks the bet size proportionally to the prediction uncertainty, resulting in a profit of 119% over three tennis seasons (2017-2019).

References

[1] T. Humphrey, *Tennis Trading On Betfair*. Trading Publications, 2014.

[2] B. L. Boulier and H. O. Stekler, "Are sports seedings good predictors?: An evaluation," *International Journal of Forecasting*, vol. 15, no. 1, pp. 83–91, feb 1999.

[3] R. A. Bradley and M. E. Terry, "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons," *Biometrika*, vol. 39, no. 3/4, p. 324, dec 1952.

[4] I. McHale and A. Morton, "A Bradley-Terry type model for forecasting tennis match results," *International Journal of Forecasting*, vol. 27, no. 2, pp. 619–630, apr 2011.

[5] F. J. Klaassen and J. R. Magnus, "Are points in tennis independent and identically distributed? Evidence from a dynamic binary panel data model," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 500–509, jun 2001.

[6] ——, "Forecasting the winner of a tennis match," *European Journal of Operational Research*, vol. 148, no. 2, pp. 257–267, jul 2003.

[7] J. Sackmann, "GitHub - JeffSackmann/tennis_atp: ATP Tennis Rankings, Results, and Stats," 2015. [Online]. Available: https://github.com/JeffSackmann/tennis_atp

[8] R. Praet, "Predicting Sport Results by using Recommendation Techniques," Ph.D. dissertation, Ghent University, 2016.

[9] M. E. Glickman, "Example of the Glicko-2 system," Boston University, Tech. Rep., 2013.

[10] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review," *International Journal of Forecasting*, vol. 16, no. 4, pp. 437–450, oct 2000.

[11] Scikit-learn, "sklearn.linear_model.LogisticRegression - scikit-learn 0.23.0 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[12] Chollet François, "Keras: The Python Deep Learning library," *Keras.Io*, 2015. [Online]. Available: https://keras.io/

[13] Tennis-Data, "Tennis Betting — Tennis Results — Tennis Odds," 2007. [Online]. Available: http://www.tennis-data.co.uk/alldata.php

[14] J. L. Kelly, "A new interpretation of information rate," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 185–189, 1956.

[15] L. C. Maclean, E. O. Thorp, and W. T. Ziemba, "Long-term capital growth: The good and bad properties of the kelly and fractional kelly capital growth criteria," *Quantitative Finance*, vol. 10, no. 7, pp. 681–687, 2010.

[16] R. D. Baker and I. G. McHale, "Optimal betting under parameter uncertainty: Improving the kelly criterion," *Decision Analysis*, vol. 10, no. 3, pp. 189–199, sep 2013.

Contents

Chapter 1

Introduction

Throughout the history of sports, people have always been fascinated by the ability to accurately forecast the outcomes of sport events. This human fascination of predicting the future in the context of sports is most apparent in the recent proliferation of online sports betting to a multi-billion dollar industry [1].

Tennis is without a doubt one of the most popular sports in the world. It is played by millions of people and the most prestigious tournaments draw a substantial amount of international attention. This popularity along with the chance to turn a profit in the betting market makes tennis an appealing research subject.

The first attempts in tackling the problem of tennis match prediction started in 1999 with the first mathematical model of Boulier and Stekler [2] to forecast tennis matches based on a player's ATP-ranking. Ever since then, interest has continued to rise, leading many researchers to develop machine learning models of varying complexities and types. The goal of this thesis is to build on the current state-of-the-art in male professional tennis prediction. Concretely, given information about a tennis match, we want to accurately predict the winning probability of each player in an attempt to turn a profit in the online tennis betting market.

During the professional tennis season, players compete in a wide array of tournaments all over the world organized by the International Tennis Federation (ITF) and the Association of Tennis Professionals (ATP). Ranking points are awarded proportional to the prestige of each tournament and how far a player manages to advance. These ranking points are used to

calculate the official ATP world ranking and serve as qualification and seeding of tournaments in the future. The tournament types ranked from most to least prestigious are: Grand Slams, Masters 1000, 500/250 Series, Challenger Tour and finally Futures tournaments. Like most research, our machine learning models will be evaluated on matches in events up until the 500/250 Series level. From now on these will be called tour-level tournaments. However, contrary to most current research, matches played in the lower-level tournaments will also be analyzed and incorporated in the feature set. This is because these matches can still contain useful information about up and coming players, as even the most successful players today, once competed at these lower level tournaments. Finally, tennis can be played individually against a single opponent (singles) or as a team of two players (doubles). Like most existing research we will restrict our focus to predicting singles matches.

1.1 Chapter Outline

In Chapter 2, we will begin by giving an overview of previous research with the goal of tennis prediction. We will also quickly go over the workings of an online bookmaker. Chapter 3 will discuss how a raw dataset of historical tennis matches was obtained and used to extract features. Afterwards, we try to beat the official ATP-ranking over multiple metrics with a logistic regression and neural network model trained on these extracted features. This is done in Chapter 4.

In Chapter 5, we attempt to turn a profit in the online betting market using the best model from the previous chapter by iteratively improving the betting strategy. Finally in Chapter 6, the results are summarized and some ideas for future work are suggested.

Chapter 2

Background

2.1 Related Work

Most approaches in the literature with the goal of predicting tennis matches fall into one of three categories: point-based, pairwise comparison and regression.

2.1.1 Pairwise Comparison Models

Pairwise comparison is a way of comparing players in pairs trough the use of some underlying metric(s) defining each player. A popular type of pairwise comparison model is the Bradley-Terry model [3] which has been broadly applied in areas such as sports, and machine learning. It allows to predict the outcome of a paired comparison of individuals i and j, given a respective metric π_i and π_j.

$$P(\text{individual } i \text{ beats individual } j) = \frac{\pi_i}{\pi_i + \pi_j} \tag{2.1}$$

The first application of such model applied directly to tennis was developed by McHale and Morton [4] in 2011. In their research, each player's winning capabilities are modeled by a single metric which is optimized using maximum likelihood over past matches, with an exponential decay to give more importance to recent matches.

Pairwise comparisons can also be be carried out by rating systems. One such rating system is the Elo rating system, originally developed by Arped Elo to more accurately reflect the skill level of chess players. These pairwise models can be used on their own [5], but they can also be incorporated in regression based models [6] or point-based models [7].

2.1.2 Point-Based Models

Tennis has a hierarchical scoring system, with a match consisting of sets, witch in turn consists of games, which finally consist of points. Matches are played until a player wins the necessary amount of sets needed to win the match (2 or 3), so there is no possibility for a draw.

Point-based models relate to a type of prediction models that in essence only try to estimate the probability of winning a point on serve and return. Assuming that every point is independent and identically distributed (IID) from all other points in the match, the winning probability of each player can be derived in closed-form.

This IID assumption of points was tested by Klaassen and Magnus [8] in 2001. They discovered that points are neither independent nor identically distributed. For example, winning the previous point has a positive effect on winning the current point. Although they found the IID assumption does not hold, they argued that deviations are small enough that using a constant probability throughout the match is fine for most purposes.

With this in mind, in 2003 they developed a way to estimate winning probabilities from any point in the match by modeling the tennis match as a discrete-time Markov chain, where each state represents a corresponding score [9]. Given each player's probability of winning a point on serve, the probability to win the next point, game, set and match can be calculated in closed-form. Figure 2.1 depicts such a Markov chain for predicting the probability to win a best-of-three set match, given the probability p to win a set.

Having established this model of a tennis match, the only thing that remains to be estimated are the serve point winning probabilities of each player. The most basic way to estimate these is to average the frequency of point wins on serve and return for each player. Nowadays, improvements in point-based models are mainly the result of improving these estimates trough the use of pairwise and regression models to model these point winning probabilities. For example, Donninger [7] used the Elo system for this purpose.

2.1.3 Regression Models

Contrary to point-based models, regression models try to model the players winning probabilities directly. They achieve this by extracting a set of features describing the characteristics

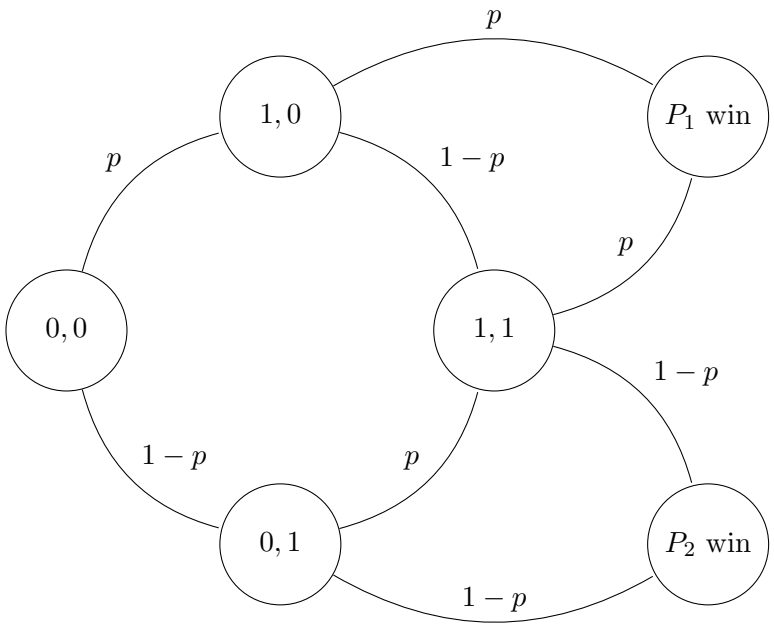

Figure 2.1: Markov chain for a best-of-three set match

of each player for every match in time. The relationship between these features and winning probability is then learned using a machine learning model capable of probabilistic classification. The models differ in the set of features they consider and the used machine learning model.

In 1999, Boulier and Stekler [2] used a probit model using solely the players ATP-ranking as features. It was the first model of its kind and has since been the inspiration for more complex features and machine learning models.

Perhaps the most extensive work in this category of models comes from Sipko (2015) [10]. 22 features from raw historical data, including abstract features, such as player fatigue and injury were extracted and used to train a logistic regression model and artificial neural network. It was also established to be profitable in the online betting market in the years 2013-2014.

Praet [6] implemented logistic regression purely for the purpose of classification using 25 extracted features with great emphasis on Elo features in particular.

Regression based models perform better than the other approaches when the goal is to predict winning probabilities before the match has started as they do not rely on the false assumption

of IID points [11]. On the other hand, point-based have more flexibility when it comes to in-play predictions and can even be used to estimate the chance of a particular score outcome. Since the goal of this thesis is to predict outcomes before the match has started, regression based methods are the most sensible choice.

2.2 Tennis Betting

Tennis has some interesting properties for betting. Given the unique scoring system, it is rather uncommon that the lesser player on that given day wins the match. This would argue in the favor of tennis being an easily predictable sport. Being an individual sport however, it is not hard to imagine that the performance of a single player is more volatile than the performance of a team in a team sport, which in turn makes tennis matches hard to predict. The popularity of the sport and this unpredictability make tennis an appealing betting market.

Bets can be placed trough a bookmaker, which functions as a market maker for sports wagers. The popularity of sports betting has seen a significant increase in the last few years as a result of online betting. It is not uncommon for one tennis match to have more than 20 different betting options before the match even starts. There is an abundance of tennis tournaments played around the world each year and the nature of the scoring system also serves well for live betting. During the game bettors can bet on a multitude of different options: who will win the next game/set, how many sets will be played, etc. However, in this thesis, the sole focus will be on predicting the winner of the match before play has started.

2.2.1 Betting Odds

The bookmaker assigns appropriate odds to each possible wager. These odds represent how probable different outcomes of a bet are. If an outcome is deemed unlikely to occur, the payout will be high. On the other hand, expected outcomes will have a lower payout. Odds can be represented in decimal, fractional or moneyline format. The most common representation in Europe is by far the decimal format, with the exception of British bookmakers, who usually use fractional odds. Henceforth, odds will always be represented in decimal format.

Let us consider the odds of the Wimbledon final of 2019 between Novak Djokovic and Roger

Federer. At the start of the match, the odds for a Djocovic win were rated at 1.55. If you would bet €2 on Djokovic and he wins, you would receive €3.1 in return (the stake is included in the return). If he loses, you would lose the €2 to to bookmaker. The same reasoning can be applied to Roger Federer, who had odds of 2.66. Note that the possible payout is determined at the time a bet is placed, as bookmakers may change their odds over time.

2.2.2 Implied Probability

The betting odds can be represented as probabilities by taking the inverse of the odds. This probability then represents the expected probability of an outcome occurring according to the bookmaker. In the example above, The implied probabilities for a Djokovic and Federer win are 64.5% and 37.6% respectively.

Converting betting odds into implied probabilities is useful for bettors as it helps to recognize value in a particular market. It indicates what percentage of the time you would need to be correct in your wager to have a positive expected return. So if you know for a fact an outcome is more likely than its implied probability, it is always profitable to bet on that outcome in the long run.

2.2.3 Bookmaker Margin

You may have noticed that the sum of the implied probabilities in the example above does not equal 100%. The excess over 100% equals the margin of the bookmaker (2.1% in this particular case). This margin is where the bookmakers make their profit.

There are two stages how the bookmakers decide the odds for each player. First, they make an initial guess of what they think are the true odds of the possible outcomes and add their margin. These odds are usually published 1-2 days before the tennis match and are called the opening odds. Afterwards, as customers start betting, they adjust both odds to attract bets in the right proportion to make a profit regardless of the match outcome. When successful, this set of odds and bets which guarantees a profit is called a Dutch book. In the other case, the bookmaker may have to pay out more than what was staked or may profit more than expected. This dynamic process of setting the odds is depicted in Figure 2.2.

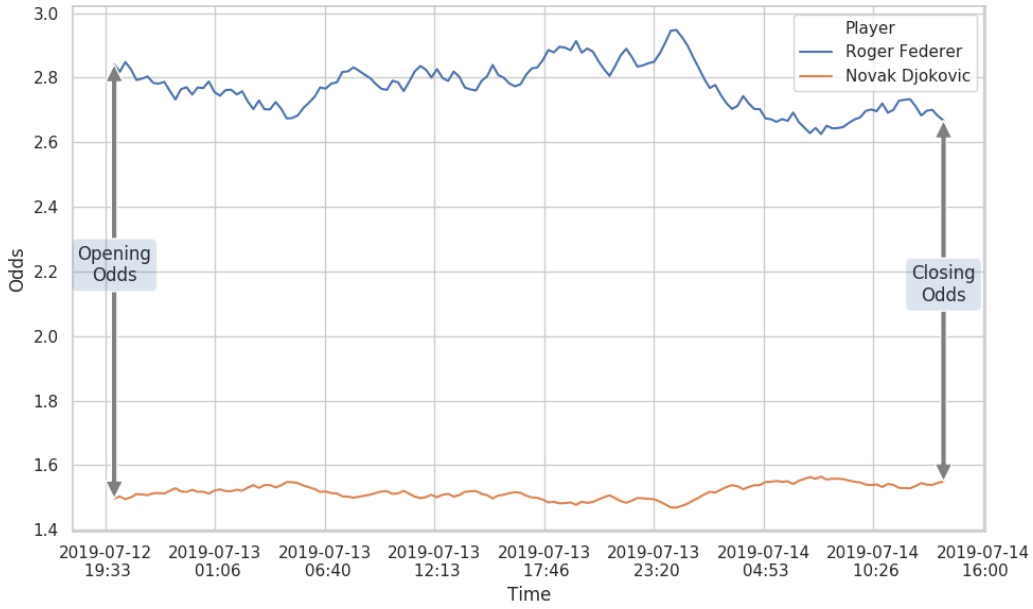

Figure 2.2: Fictitious example of how odds can drift away from the opening odds

2.2.4 Betting Market Efficiency

The odds just before a match starts are called the closing odds. They are a representation of all the money bet on that market up until the start of the match. They contain all the information known to the bettors and their opinions. Closing odds are a closer approximation to the true odds of the tennis match. There is more information available about the match since the opening odds were published that could influence the expected outcome: weather forecasts may have changed to favor a certain player, rumors about an injury may pop up, etc. This information does not necessarily need to be public. Bettors with inside information can also influence the odds as long as they wager enough money.

Bookmakers with a higher volume of wagered money can afford to operate with a smaller margin because their closing odds are more efficient. In this thesis, the closing odds from Pinnacle will be used to validate our models. Pinnacle is one of the sharpest (low margin) bookmakers available in the market.

The efficiency of Pinnacle's closing odds are shown in the form of a calibration curve in Figure 2.3. A calibration curve enables us to compare a model's predicted probability to the

empirical probability of an event. In this case the predicted probability is equal to the implied probability of the closing odds. The empirical probability is equal to 1 if the event occurred, otherwise 0. The predicted and empirical probabilities are binned and depicted by a blue line.

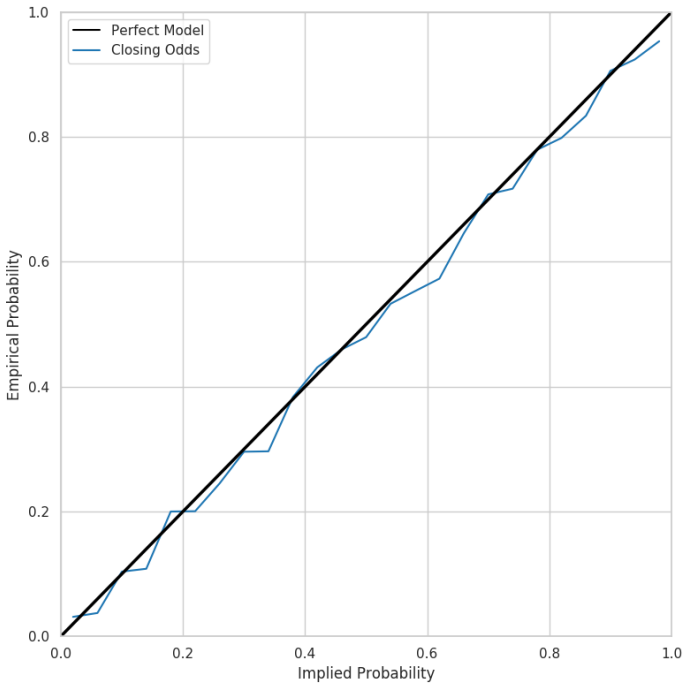

Figure 2.3: Pinnacle closing odds calibration plot.

From the calibration curve it is clear that, on average, the implied probability is higher than the empirical probability. This is a direct result of the margin applied by Pinnacle. As explained in Section 2.2.2, a bet is only profitable if the true probability of an event is higher that the implied probability of the odds. As a result, betting on every match with the same stake will result in a negative expected return regardless of the strategy. This notion is in line with research from Lyócsa and Výrost [12]. The Performance of 40 diverse betting rules on the tennis market were found to be not profitable after data-snooping bias was taken into account.

2.2.5 Beating the Market

Clearly, a profitable betting model should not bet on every match. At the very least a good model should bet more if it is confident and less when unsure. Consequently, if the goal is to make a profitable betting model, it should not only predict the winner of the match, but also the confidence of the prediction. As a whole the tennis betting market is very efficient, but that does not rule out that some matches have biases that can be exploited by a smart betting model.

Chapter 3

Dataset

A vital part of any machine learning algorithm is the dataset. Although the learning algorithms are usually the center of attention, no useful learning can take place when using a poor dataset. High-quality labeled datasets are usually difficult and expensive to produce. Fortunately there exist some excellent open-source datasets for tennis.

3.1 Tennis Match Data

The bulk of the raw tennis match data was collected from an open-source dataset published by Jeff Sackmann on GitHub [13]. The dataset contains all tour-level main-draw matches from Grand Slams, ATP Masters 1000 tournaments, and ATP 500/250 Series tournaments starting from 1968. The dataset also contains tour-level qualifying and Challenger matches starting from 1978. Finally, Futures tournaments are recorded starting from 1991. The dataset is updated regularly to include all recent matches.

The dataset currently covers a total of 788135 matches. The distribution of these matches is represented in Figure 3.1. Over half of the recorded matches are played in Futures tournaments while the rest is quite evenly split between Challenger and tour-level matches. This is to be expected, because only a fraction of the young Futures players will make it as a professional tennis player in the end.

Even though the goal of this thesis is to predict tour-level matches, match results at a lower level will also be used for training and feature extraction. When a player first competes

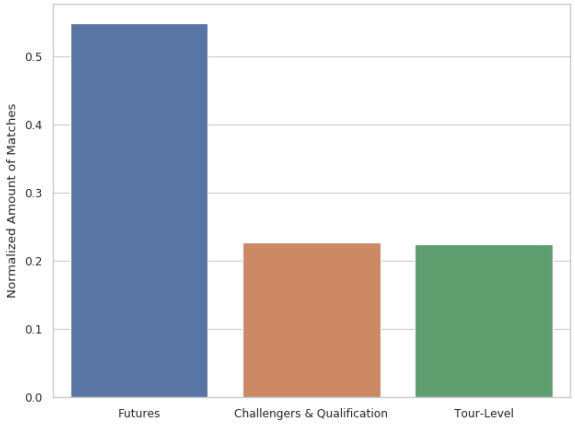

Figure 3.1: Distribution of match data divided by tournament level

in a tour-level match, his strength can still be assessed using data from the Futures and Challengers tournaments. In the absence of this data, the model would have to treat every new player as equal and would have to be overly conservative in its predictions. Even the best players spend a few years of their early career in the Futures and Challengers circuit before they are ready for the bigger tournaments.

3.2 Betting Odds Data

A probabilistic classifier can be built using the data above. However, to test the classifier's performance on the betting market, we need to have betting odds data to simulate a betting environment. Tennis-Data provides free match data of tour-level matches starting from the year 2000 [14]. From 2010 onward, it also contains the Pinnacle closing odds for all those matches. These closing odds for each match are simply linked to the corresponding match in the Sackmann dataset.

3.3 Data Cleaning

Matches that were stopped before completion are removed from the dataset. It is not useful that the model learns form these matches, as they are usually the result of injuries or disqualification. In this scenario, the injured or disqualified player's strength is not properly reflected in the result of the match and is thus not useful as learning data for the model. Walkovers

are removed from the dataset for the same reason. This data cleaning removes 27780 matches from the dataset, accounting for 3.5%.

3.4 Feature Extraction

Machine learning algorithms need a set of training examples to learn the relationship between input and desired output. Therefore, the initial set of raw data needs to be mapped to an informative set of features, where each data record represents one tennis match in time, alongside with the desired target value (win or loss). This process is called feature extraction.

Before the features are extracted, the raw Sackmann dataset is sorted by date. This will ensure that each feature extractor will only use data from the past, as using matches from the future would result in data leakage.

3.4.1 Types of Features

We will discern the extracted features as either environment features or player features. Environment features are equal for both players in a match. The date of a match, the playing surface, the weather and the tournament are examples of such features. As a result, one such feature can be modeled using only one value per match.

On the other hand, features such as age and ranking are player features because they are not equal for both players. These features need to be modeled by two separate features, one for $Player_1$ and one for $Player_2$.

3.4.2 Target

The desired target value is defined as follows:

$$Target = \begin{cases} 1, & \text{when } Player_1 \text{ wins} \\ 0, & \text{when } Player_1 \text{ loses} \end{cases} \tag{3.1}$$

Notice how the target value can always be made 0 or 1 by swapping the values of the player features. A basic example is shown in Table 3.1. The environment features obviously remain unchanged.

Table 3.1: Shows how one match can be represented in 2 different but equivalent ways

Year	$Player_1$ Rating	$Player_2$ Rating	Surface	Target
2011	200	115	Clay	1

⇕

Year	$Player_1$ Rating	$Player_2$ Rating	Surface	Target
2011	115	200	Clay	0

3.4.3 Wins & Losses

The most basic player feature is to use the amount of wins and losses of a player as a feature. The most obvious way to model this is to use the winning percentage as a single feature to model both wins and losses together. While this works well for players that have already played a lot of matches, it is a rather poor feature for players with a low amount of matches played. Consider for example a player who has played and won only one single match. Even though this player has a winning percentage of 100%, it is safe to assume this player is not as good as someone with an 80% win rate over many more matches. The opposite conclusion can be drawn for a player that lost the first match of his career. For this reason, this feature will be simply modeled by the absolute number of wins and losses. In doing so, the model can decide for itself if and how players with different amounts of matches will be handled.

Of course not every win and loss is equally informative about a player's strength. The level of the tournament and opponent naturally play a role in the importance of a given match. To make this distinction, the matches are divided in three categories:

1. Tour-level matches: the main-draw of Grand Slams, ATP Masters 1000 tournaments, and ATP 500/250 Series tournaments.

2. Qualifying matches for the tournaments above, as well as Challenger matches.

3. Futures matches.

Finally, the matches are divided another time per player by time. The skill of a player can rapidly increase or decrease with age. It would be unfair to weigh the matches of a player's

junior days equally to his recently played matches. To tackle this problem, a distinction is made for matches played in the last semester, year and whole career of that player.

This extra information does not come for free. That which first started as a simple winning percentage, is now modeled by 36 feature values. One value per combination in the following Cartesian product:

$$\{Player_1, Player_2\} \times \{wins, losses\} \times \{level_1, level_2, level_3\} \times \{semester, year, career\} \quad (3.2)$$

3.4.4 Rating Systems

Another way to reflect a player's strength is by the use of a rating system. The exact calculation of the official ranking utilized by the ATP is rather complex. Put simply, ranking points are awarded to a player according to how far they get in each tournament, with more prestigious tournaments being worth more points. These earned ranking points are dropped 52 weeks after after the tournament took place. A player's ATP-ranking points thus represent their performance in the last year. While this ATP-ranking serves as a valid tool for seeding draws and as a good way to encourage players to participate in more tournaments, it has some obvious flaws when used as a means to reflect current form:

- Ranking points are awarded irrespective of the opponents. Ideally, the level of the opponent should also be considered in assessing the value in a win or loss.
- Points can only be won, not lost. This way, players with average results, who played the maximum amount of allowed tournaments can achieve a better ranking than players with better results, who were unable to do so for whatever reason.
- Solely takes into account tournaments in the last year.

Therefore, ATP-ranking will not be used as a feature. Instead we will first consider Elo as an improvement over the ATP-ranking.

Elo rating was originally invented as a skill rating in chess, but can be easily used in any 2-player zero-sum game, including tennis, without any modifications. Elo's primary assumption is that the performance of each player is a normally distributed random variable. The expected

outcome of a match can then be calculated by taking into account the difference in rating between the two players:

$$E_1 = \frac{1}{1 + 10^{\frac{R_2 - R_1}{400}}} \tag{3.3}$$

Where:

$$E_1 = \text{expected winning percentage of } Player_1$$
$$R_i = \text{the Elo rating of } Player_i$$

After every match the winning player takes a number of Elo points from the losing one proportional to the elo difference before the match. An expected outcome will lead to only a few points being transferred. However, if an upset occurs, more points will be given from the high-rated to the low-rated player.

Elo alleviates the 3 main shortcomings of the ATP-ranking described above and has been shown to be a better predictor than ATP-ranking in predicting tennis matches [5]. Elo does however have some flaws of its own. Because rating is represented by only one number, there is no way to differentiate between players that might have different standard deviations to their performances.

The Glicko [15] and subsequent Glicko-2 [16] rating systems invented by Mark Glickman, were designed to resolve this issue. Besides the rating of a player, Glicko considers another variable for rating deviation (RD), which is simply a standard deviation that measures the uncertainty of a rating. This RD is decreased when a player competes in a match and is increased after some period of inactivity. As is the case in Elo, the rating is simply changed after every match, albeit not longer in a symmetrical fashion, but depending upon the respective RD's.

Finally, Glicko-2 build on Glicko by adding a rating volatility variable (σ) to indicate the degree of expected fluctuation in a player's rating. This allows the system to model players who have erratic performances differently from players that perform more consistently. The extra modeling power of Glicko compared to Elo comes at the cost of some hyperparameters which are optimized using Bayesian optimization.

Similarly to Elo, Glicko allows to estimate the expected outcome of a match:

$$g(\mathrm{RD}) = \frac{1}{\sqrt{1 + 3q^2\left(\mathrm{RD}^2\right)/\pi^2}}$$
$$E_1 = \frac{1}{1 + 10^{-g(\sqrt{\mathrm{RD}_1^2 + \mathrm{RD}_2^2})(R_1 - R_2)/400}} \tag{3.4}$$

This expected winning percentage could be used as a feature, but instead we will use the rating, *RD* and volatility of each player as features, so the model can learn and possibly improve this relation for tennis matches itself. To reiterate, the added rating features are:

$$\{Player_1, Player_2\} \times \{R, RD, \sigma\} \tag{3.5}$$

We compare the accuracy of the ATP-ranking and Glicko ratings systems with regards to predicting the winner in all tour-level matches from 2016-2019 in Table 3.2. Glicko outperforms the ATP system by almost 2%.

Table 3.2: Accuracy comparison between ATP-rank and Glicko

Rating System	Accuracy
ATP	64.5%
Glicko	66.4%

3.4.5 Surface

Across all levels of tennis, the playing surface can have a sizable influence on the expected outcome of a match. Even the best players have surfaces were they perform much better or worse. For illustration, The winning percentage by surface of 4 top level players is shown in Figure 3.2, wherein it is indeed clear that there exists a discrepancy in performance between the surfaces for each player.

To model this disparity in player performance by surface, 4 environment features are added for every match as a one-hot encoding over clay, hardcourt, grass and carpet:

$$\{?clay, ?hardcourt, ?carpet, ?grass\} \tag{3.6}$$

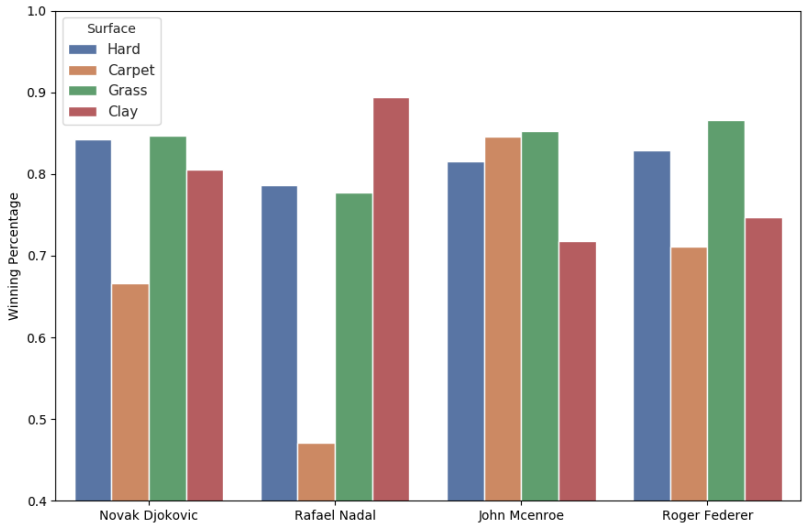

Figure 3.2: Winning percentage by surface

These environment features are not enough on their own. It is also necessary to add player features to reflect the past performance of a player on each surface. Simply using winning percentage per surface has the same disadvantages as described in Section 3.4.3. Therefore, it will again be represented by the absolute number of wins and losses, like so:

$$\{Player_1, Player_2\} \times \{wins, losses\} \times \{clay, hardcourt, carpet, grass\} \tag{3.7}$$

The need for a further subdivision in time is not necessary here, because the surface preference stays rather constant over a player's career.

3.4.6 Home advantage

Home advantage is a ubiquitous phenomenon in sport. It has been confirmed to exist in most team sport where the attendance of large crowds is possible [17]. Koning [18] studied the prevalence of home advantage in professional tennis matches. It was concluded that unlike on the women's tour, significant home advantage exists in the men's tour. This phenomenon will be indicated to the model by a player feature:

$$\{Player_1, Player_2\} \times \{?home\} \tag{3.8}$$

Where $?home$ is a boolean variable, which equals true for a player if the match is held in his home country, otherwise false.

3.4.7 Age

del Corral and Prieto-Rodríguez [19] established that the age difference between players has a pronounced effect on match outcomes. In particular, the probability of a higher-ranked player's victory decreases as this player competes against a younger player. It is noted that when the higher-ranked player is 15 years younger than the lower-ranked player, his winning probability is over 20% higher than when he plays against an opponent his own age. With this in mind, the age of each player is also added as a feature for the model:

$$\{Player_1, Player_2\} \times \{age\} \tag{3.9}$$

3.4.8 Best-Of-X

On the men's tour, matches are played to either best-of-three or best-of-five sets. Up until 2007, most Masters Series finals were played to best-of-five. Nowadays, all tournament matches are played according to the best-of-3 format, with the exception of the Grand Slams, were best-of-five sets is still played in every round.

As shown in Table 3.3, matches that are played to best-of-five sets are considerably more predictable than best-of-three set matches. Intuitively this makes sense as longer matches lower the influence of chance in the outcome of the match.

Table 3.3: Predictability of tour-level matches (2016-2019)

Best-Of	ATP-Ranking Accuracy
3	62.65%
5	72.19%

The favorite of the match is not influenced by the length of the match, but certainly the model should be more confident in its prediction for best-of-five matches. For this reason, this information is added as an environment feature.

$$\{bo\} \tag{3.10}$$

3.4.9 Head-To-Head

A rather straightforward feature to add is the head-to-head of the competing players. It speaks for itself that a particular playing style can have an inherent advantage over some other playing style. Consequently, it makes sense that past performances against an opponent can predict future performances. This feature is encoded as:

$$\{Player_1, Player_2\} \times \{wins\} \tag{3.11}$$

Where wins of course only count the wins versus the current opponent. Losses need not be encoded this time as they are always equal to the wins of the opponent.

3.4.10 Tournament

The same reasoning can also be applied to tournaments [6]. Previous success in a tournament may make a player more determined in future editions of the same tournament. This feature is added as:

$$\{Player_1, Player_2\} \times \{wins, losses\} \tag{3.12}$$

Where only the wins and losses in previous editions of the tournament are taken into account.

3.4.11 Recent Matches

As in any sport, accumulated fatigue can have a detrimental effect on performance. In tennis, skeletal muscle function has been reported to decrease after prolonged match-play [20]. Furthermore, it is noted that these negative effects may be amplified with consecutive days of match-play. We include this fatigue with the following feature:

$$\{Player_1, Player_2\} \times \{\#\text{games in last 2 weeks}\} \tag{3.13}$$

Although fatigue is undesirable, long periods of complete inactivity should also be avoided. Figure 3.3 shows that up to three weeks of inactivity appear to be beneficial for a player's winning chances in his next match. This inactivity can be a much welcome break on the densely packed tennis tour. Inactivity longer than this probably results in some rustiness, leading to a worse performance down the line. Therefore, the amount of weeks since each

player last competed is added:

$$\{Player_1, Player_2\} \times \{\#\text{weeks inactive}\} \tag{3.14}$$

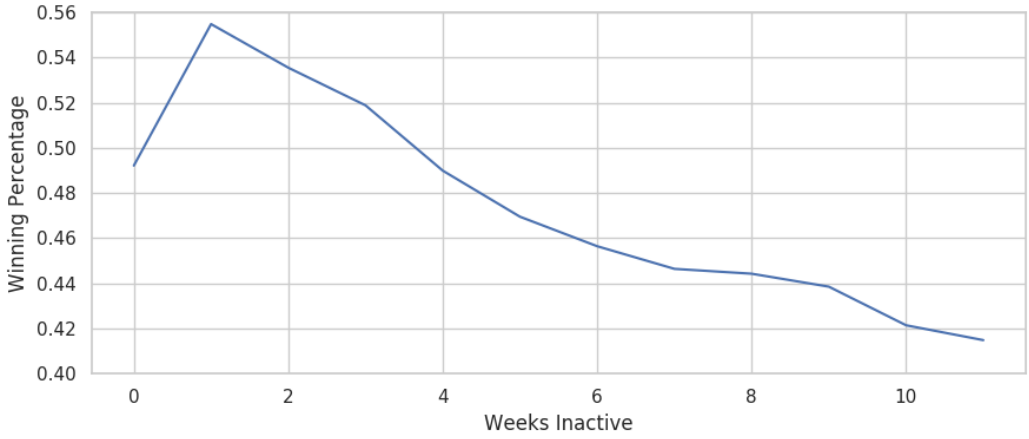

Figure 3.3: Winning percentage over inactivity

Finally some notion of a winning and losing streak is added. Intuitively, players that are in a winning mood are more likely to continue on their winning streak, while the opposite might be true for players in a slump. This Information is added as:

$$\{Player_1, Player_2\} \times \{\text{consecutive wins}, \text{consecutive losses}\} \tag{3.15}$$

3.4.12 Time

The final features are related to time. Tennis is an ever-changing sport so it might be the case that the significance of features slightly changes with time. Therefore, a notion of time is added to the features so the model can discern between different time periods. First of, the year the match was played in is simply added as in integer:

$$\{year\} \tag{3.16}$$

Then, the day of the year is added as a feature. This is encoded as a cyclical feature using the sine and cosine functions:

$$\{day\_cos, day\_sin\} \tag{3.17}$$

With:

$$day\_cos(day) = \cos(day \cdot \frac{2\pi}{365})$$
$$day\_sin(day) = \sin(day \cdot \frac{2\pi}{365})$$

To demonstrate, these functions are plotted in Figure 3.4. The use of a cyclical encoding instead of a simple integer will allow the model to better recognize that day 0 and 365 are almost identical, while as in integer they would be furthest apart.

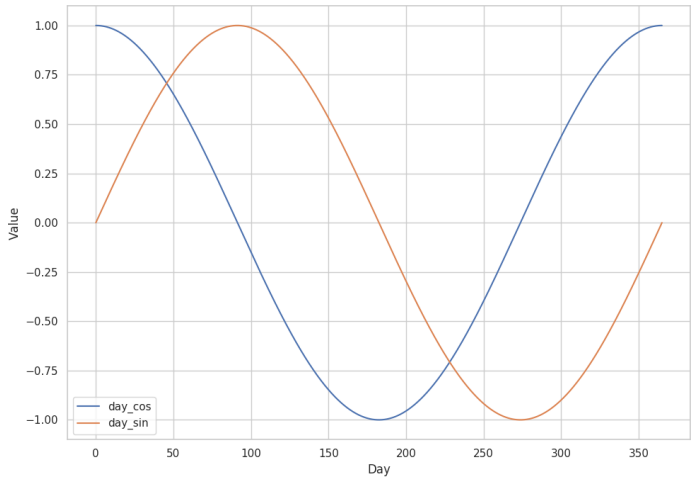

Figure 3.4: Day of year features

3.4.13 Summary

To summarize, all features are shown in Table 3.9 In total, the number of extracted features per tennis match is 84, consisting of 76 player features and 8 environment features. These extracted features and target (3.4.2) will be used as the input and output respectively of the machine learning models in the next chapter.

Table 3.4: All extracted features

Category	Type	Number of Extracted Features	Representation
Wins & Losses	Player	36	3.2
Rating Systems	Player	6	3.5
Surface	Environment	4	3.6
Surface	Player	16	3.7
Home Advantage	Player	2	3.8
Age	Player	2	3.9
Best-Of-X	Environment	1	3.10
Head-To-Head	Player	2	3.11
Tournament	Player	4	3.12
Recent Matches	Player	8	3.13, 3.14, 3.15
Time	Environment	3	3.16, 3.17
		84	

Chapter 4

Machine Learning Models

In this chapter different machine learning models will be implemented for the purpose of predicting tennis matches. For now, the focus will be entirely on making good predictors without considering the betting market. A better predictor will result in a better betting model provided that the same betting strategy is used.

First, a logistic regression model will be fitted solely on ATP-ranking as a baseline. Afterwards, We will attempt to beat this baseline with a logistic regression model and a neural network using the extracted features from last chapter. The different models will be evaluated using the same testing strategy and metrics.

4.1 Symmetric Binary Probabilistic Classification

in Section 2.2.5 it was discussed that only predicting if a player will win or lose a match is not enough for a smart betting model. As a result we will need to use machine learning models which are capable of predicting a probability distribution for winning and losing. Ideally we want to train our model on the true probability that a player wins a match. Sadly we only know whether a player won or lost, not the true probability. Classifiers that give a probability distribution over a set of classes are called probabilistic classifiers.

A classification problem with only two possible classes is called binary classification. A useful property of binary problems is that it suffices to only predict the chance of winning p. The chance of losing can then simply be derived as $1 - p$.

To meet this requirement the model will also need to be symmetric. As previously shown in Table 3.1, a tennis match can be represented in two different but equivalent ways by swapping the player features. The original and swapped feature representation will from now on be depicted by x^1 and x^2 respectively (and simply by x if it does not matter). The superscript indicates for which player in the original representation the winning probability is predicted. We call a model symmetric if for any given predicted winning probability p for $Player_1$, the predicted winning probability for $Player_2$ equals $1 - p$. So:

$$h_\theta(x^1_{(i)}) = 1 - h_\theta(x^2_{(i)}) \tag{4.1}$$

for any match i where $h_\theta(x)$ is the predicted winning probability of a given model with parameters θ.

4.1.1 Loss Function

Machine learning algorithms try to minimize a loss function during the training process. For our problem of probabilistic classification, logistic loss is the most logical choice, as it leads to well-calibrated probabilistic outputs [21]. Logistic loss is expressed by the following function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_{(i)}\ln\left(h_\theta\left(x_{(i)}\right)\right) + \left(1 - y_{(i)}\right)\ln\left(1 - h_\theta\left(x_{(i)}\right)\right)\right] \tag{4.2}$$

Where:

$$m = \text{total amount of matches}$$

$$y_{(i)} = \text{target of the match } i \text{ as described in Section 3.4.2}$$

The logistic loss is technically an unbounded positive real number. However, a reasonable upper bound, known as the non-informative prior, is obtained by predicting 0.5 for both classes of a binary classification problem, provided that the dataset is balanced. The upper bound of the loss then becomes $ln(0.5) \approx 0.693$. Note that the dataset can always be made balanced since for every winner there is a loser, but the way in which it will be made balanced will differ by model.

4.1.2 Metrics

Evaluation metrics are a way to measure and compare the quality of machine learning models. it is often desirable to use multiple evaluation metrics, because some models may score well using one particular evaluation metric, but may perform poorly using another metric. The logistic loss itself, albeit harder to interpret, is also very useful to compare between the models, as minimizing a well chosen loss function should naturally improve the metrics.

The simplest metric for classification is accuracy. This measures the fraction of matches that are correctly categorized as either a win or a loss. Accuracy has the advantage that is the easiest to interpret, but unlike logistic loss, it does not take into account the confidence of the predictions.

Another metric that will be used is calibration. In Section 2.2.4 the concept of probability calibration was discussed in the context of betting odds. Similarly, it is desirable for a model's predictions to be reflective of the true underlying probability in a probabilistic classification context. Minimizing logistic loss results in well calibrated results, so no postprossessing of the predicted probabilities will be necessary.

4.2 Testing Strategy

To assess the generalization capacity of the trained models, a nested validation strategy is used. Model hyperparameters will only be tuned in the inner validation loops, while an unbiased error estimation can be made in the outer validation loop. This nested validation strategy is illustrated in Figure 4.1

When dealing with time series data, precise care must be taken to avoid data leakage, which would lead to an overly optimistic error estimation. In order to achieve this, the rolling origin validation technique is used, wherein the training set for each testing set simply contains all matches that chronologically occurred before all matches in the test set [22].

Concretely, the outer validation loop splits the dataset in four train-test sets resulting in four test years: 2016, 2017, 2018 and 2019. Matches played in qualifying, Challengers and Futures tournaments are filtered from the test sets, since we are only interested in predicting

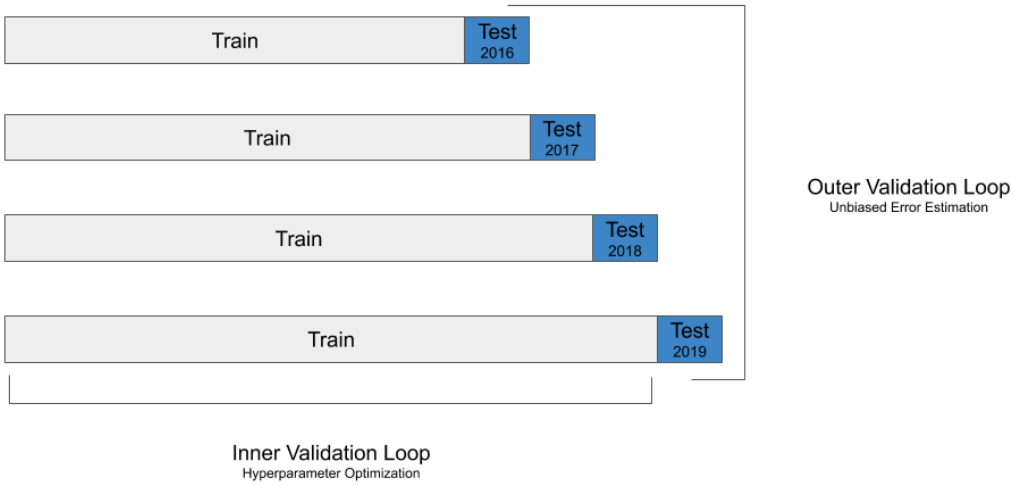

Figure 4.1: Testing strategy using nested validation

tour-level matches. Analyzing the performance on the different test years will give a better understanding of how the models perform. To fairly compare the various models, these train-test sets are kept constant, but the validation strategy used to tune the hyperparameters for each training set may differ between models.

4.3 Hyperparameter Tuning

Besides the model parameters, which are trained during the training process, most models usually have some hyperparameters as well. These hyperparameters are set before the learning process begins and needs to be optimized separately. The best tuning technique for a particular problem is mostly dependent on the amount of hyperparameters that need to be tuned and the training time of the model. The three most used ways to tune hyperparameters are explained below.

4.3.1 Grid Search

The conventional way of hyperparameter optimization is grid search. In grid search all combinations of hyperparameters are exhaustively tried. The combination that lead to the lowest validation loss after training the model will be chosen as hyperparameter values.

Since most hyperparameters are real unbounded variables, it is impossible to exhaustively try all values of a hyperparameter, let alone all combinations. Instead, a subset of the hyperparameter space needs to be specified so it becomes computationally feasible.

Still, a complete search over all hyperparameters quickly becomes impossible as the amount of hyperparameters grow. This is why grid search is mostly used if there is a low amount of hyperparameters that need to be tuned and the model has a fast training time.

4.3.2 Random Search

Contrary to grid search, random search does not exhaustively enumerate all hyperparameter combinations, but instead samples the combinations randomly. It can greatly outperform grid search when only a small subset of hyperparameters affect the result of the model.

The main advantage of random search is that it can be terminated at any time without leaving big parts of the hyperparameter space unsearched. Hence, if it is not computationally feasible to a full grid search, random search is preferred.

Random search is also sometimes performed as an initialization step for a full grid search. The found hyperparameters of the random search can be used to set the subset of the hyperparameter space for the grid search.

A large amount of hyperparameters is no problem for random search. However, because of the random sampling, it is only usable when the model has a short training time, so a lot of parameter combinations can be tried. Otherwise Bayesian optimization becomes the better choice.

4.3.3 Bayesian Optimization

Bayesian optimization is an optimization method that can be used on any black-box function and is used when it is very expensive to search the parameter space due to a long training time. Bayesian optimization fits an acquisition function (usually a Gaussian process) over the evaluated hyperparameters and respective loss achieved after training the model. The optimum of the acquisition function is then chosen as a new candidate for evaluation, after which a new acquisition function is fit and the process repeats itself.

In contrast to random search, a lot of effort is put into finding the next combination of hyperparameters to try. Consequently, this method really shines when the time needed to find the optimum of the acquisition function is low compared to training the model.

4.4 Logistic Regression

Logistic regression is a linear classifier that uses the logistic function to model a binary dependent variable. Logistic regression is desirable for its fast training time and being less prone to overfitting compared to more complex models, meaning that it usually generalizes well to to unseen data.

Given n features $x = \{x_1, x_2, \ldots, x_n\}$, a logistic regression model consists of $n+1$ real-valued model parameters or weights $\beta = \{\beta_0, \beta_1, \ldots, \beta_n\}$. A probabilistic prediction is then obtained by computing:

$$h_\theta(x) = \phi(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n) \tag{4.3}$$

with $\phi(z)$ being the logistic function:

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{4.4}$$

The logistic function from 4.4 squashes the unbounded result of the linear combination of features to a value between 0 and 1 so it can be interpreted as a probability, as shown in Figure 4.2.

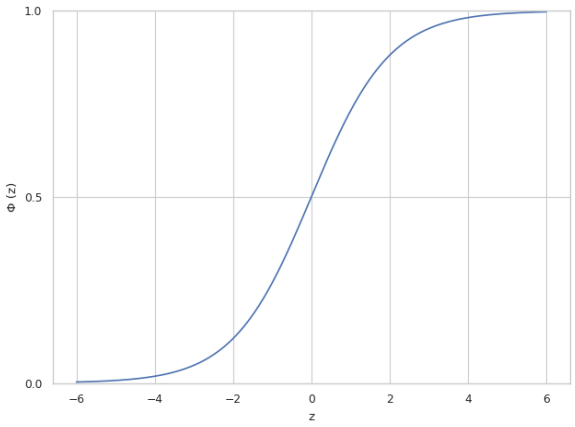

Figure 4.2: The logistic function

The β weights are optimized by minimizing the logistic loss on the predictions. For large datasets this is generally done using gradient descent on the weights over the logistic loss. A minima of the loss function is found by iteratively updating the model weights, taking steps proportional to the negative of the gradient and the learning rate η.

$$\beta_i = \beta_i - \eta \frac{\partial J(\theta)}{\partial \beta_i} \tag{4.5}$$

4.4.1 Model Simplifications

Because logistic regression is a linear model, the prediction only depends on a linear combination of the input features. This property allows us to drastically reduce the number of parameters of the logistic regression model. This reduction of parameters can be seen as a form of regularization, meaning it will make the model less susceptible to overfitting to the training data.

Intercept

The parameter β_0 is known as the intercept. If no information about a match was known, the input feature vector would become $x = \{\}$. Subsequently the predicted winning probability would become $\phi(\beta_0)$. Considering that the dataset is balanced and because there is no further information about the match this probability should equal 0.5:

$$\phi(\beta_0) = 0.5 \Leftrightarrow \beta_0 = 0 \tag{4.6}$$

Therefore, to avoid training the intercept weight to noise, it can just be set to 0 in our case.

Environment Features

Environment features should also be ignored and the corresponding β weight can be set to 0. Let us consider an environment feature x_e, which naturally has the same value in its x^1 and x^2 feature vector representation. If the corresponding weight β_e is greater than 0, it would indicate that this environment feature increases the chance of a win for $Player_1$. However, it would in turn also increase the chance for a $Player_2$ win, which is a contradiction. The same argument can be made for a negative β_e. Environment features are thus not useful when using logistic regression, but they will later be used by the neural network.

Player Features

After removing the environment features, we are left with just 76 player features. Another simplification follows from the observation that the weights for the same player feature of both players should be equal in absolute value, but with opposite sign. Consider the player features x_{p_1} and x_{p_2} with respective weights β_{p_1} and β_{p_2}. To fulfill the requirement for a symmetric model (4.1), the following must be true:

$$\phi(\beta_{p_1}x_{p_1} + \beta_{p_2}x_{p_2}) = 1 - \phi(\beta_{p_1}x_{p_2} + \beta_{p_2}x_{p_1}) \tag{4.7}$$

Using a property of the logistic function:

$$\begin{aligned}\phi(x) = 1 - \phi(-x) \Rightarrow \beta_{p_1}x_{p_1} + \beta_{p_2}x_{p_2} &= -(\beta_{p_1}x_{p_2} + \beta_{p_2}x_{p_1}) \\ \Leftrightarrow \beta_{p_1}(x_{p_1} + x_{p_2}) &= -\beta_{p_2}(x_{p_1} + x_{p_2}) \\ \Leftrightarrow \beta_{p_1} &= -\beta_{p_2}\end{aligned} \tag{4.8}$$

With this in mind, $\beta_{p_1}x_{p_1} + \beta_{p_2}x_{p_2}$ can simply be replaced by $\beta_{p_1}(x_{p_1} - x_{p_2})$. This halves the amount of features and consequently the model parameters to 38.

4.4.2 Balanced Dataset

In machine learning it is important that the desired targets of the training set are identically distributed as in the real world. If this is not the case, the model will learn this bias in the data and consequently make poor predictions.

Given a random match, with randomly assigned $Player_1$ and $Player_2$, there is an equal chance of 0.5 for a win or a loss. Therefore, the training dataset should be made balanced as well, containing an equal amount of losses and wins. This is achieved by randomly sampling half of the dataset and setting the target to 0, while the other half is set to 1. Of course the corresponding feature vector is also swapped between x^1 and x^2 if needed.

4.4.3 Inner Validation Strategy

Logistic regression itself only has one hyperparameter to be optimized, which controls the amount of regularization. This regularization parameter penalizes large values of β by adding

an L2 (alternatively L1) regularization term to the logistic loss function, like so:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_{(i)}\ln\left(h_\theta\left(x_{(i)}\right)\right) + \left(1-y_{(i)}\right)\ln\left(1-h_\theta\left(x_{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}|\beta_j|^2 \tag{4.9}$$

This regularization simplifies the model by constraining the weights to a smaller range, resulting in a model that is less likely to overfit. The degree of penalization is controlled by the λ parameter and needs to be optimized. The possible values for λ are usually explored in a logarithmic manner.

Another hyperparameter is whether train the model on all available matches in the train set or only tour-level matches which more closely resemble the matches in the test set. This is a boolean decision variable that can be optimized just like any other model hyperparameter.

The inner loop validation strategy used for the logistic regression model is equal to the outer loop validation strategy. Concretely, each of the four training sets is divided in four train and validation sets using the rolling origin validation technique. The validation sets are filtered so they only contain tour-level matches just like the test sets. The validation loss of a set of hyperparameters is then equal to the average loss of the four validation sets in a particular test year. As a result of the model simplifications and fast training time, the two parameters above can be optimized using grid search.

The resulting parameters are shown in Table 4.1. The optimal λ (often reported as $\frac{1}{\lambda}$) for every test year turns out to be rather small, so not much regularization is necessary for our problem. It also stands out that the model scored better by training only on tour-level matches as opposed to training on all data. This does not mean that the data from these lower-level matches is not used at all, since they are in some way also included in the extracted features of the the tour-level matches.

4.4.4 Preprocessing

Standardization of the input features is not an explicit requirement for logistic regression. However, when using regularization it is strongly advised to do so anyway. Standardization is the process of transforming the values of each feature in the training set to have mean 0 and variance 1 over all samples:

$$x_i = \frac{x_i - \bar{x_i}}{\sigma} \tag{4.10}$$

Table 4.1: Logistic regression hyperparameters for each test year

Test Year	$\frac{1}{\lambda}$	All Matches?
2016	1000	No
2017	1000	No
2018	0.494	No
2019	0.869	No

Where:

$$\bar{x_i} = \text{the average value of feature } i \text{ over all training samples}$$

$$\sigma = \text{the standard deviation of feature } i \text{ over all training samples}$$

The feature values from the validation or test set are not used to determine the transformation to avoid data leakage. The feature values in the validation or test set are then also transformed using the same transformation, using $\bar{x}$ and σ from the training set.

Standardization transforms all features to distributions with equal mean and standard deviation so that all features are now equally penalized by the regularization parameter.

4.4.5 Baseline

Finally we will set a baseline for the full logistic model by first training a small logistic model using only the ATP-ranking points of $Player_1$ and $Player_2$ as features. As discussed in Section 4.4.1, in the case of logistic regression this can be represented by a single feature: $rating_{p_1} - rating_{p_2}$. In turn , the regularization hyperparameter does not need to be optimized, because only one model parameter β_{rating} needs to be trained.

4.4.6 Evaluation

The full logistic regression model is trained on the four training sets using the tuned hyperparameters from Table 4.1 and compared to the baseline in Figure 4.3 and Table 4.2. The models were implemented using the scikit-learn logistic regression implementation [23] and trained using the lbfgs solver.

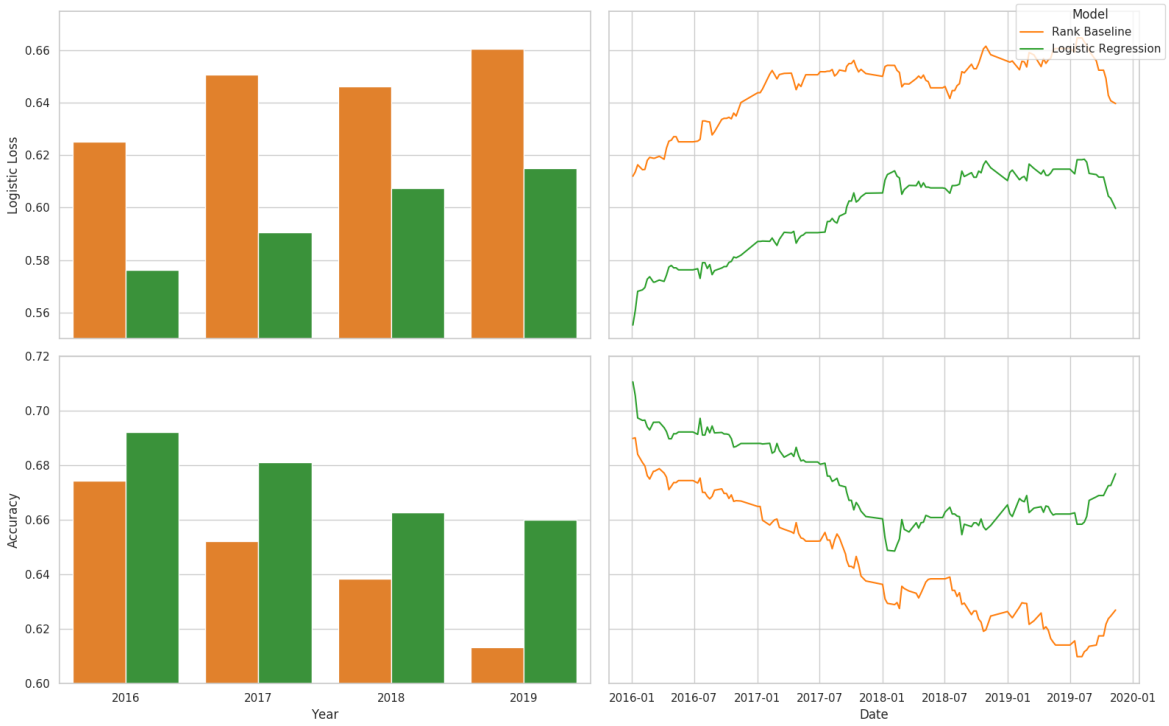

Figure 4.3: Comparison of the rank baseline with the logistic regression model

Table 4.2: Metrics averaged over all matches in the test sets

Model	Logistic Loss	Accuracy
Rank Baseline	0.646	64.5%
Logistic Regression	0.597	67.4%

The two upper plots compare the logistic loss between the two models, while the lower two compare accuracy. The plots on the left contain the same information as the plots on the right, but in a different format. The left plots are laid out by test year, while the plots on the right evaluate the metrics using a central moving average with a window size of one year. The table evaluates the metrics over all matches in the four test sets.

The full logistic regression model substantially outperforms the baseline in both metrics. We can conclude that the extracted features are much more informative than ATP-ranking on its own.

There is a noticeable downwards trend in the predictability over the years for both models. Since the baseline only uses ATP-rankings as a feature, we can conclude that the reason for this is that upsets are getting more and more common in recent years.

Finally the calibration plot of the two models is compared in Figure 4.4. As expected, the full logistic regression model is much better calibrated than the baseline. Both models have a bias of underestimating the winning chances of the underdog, but the full logistic model to a much lesser degree.

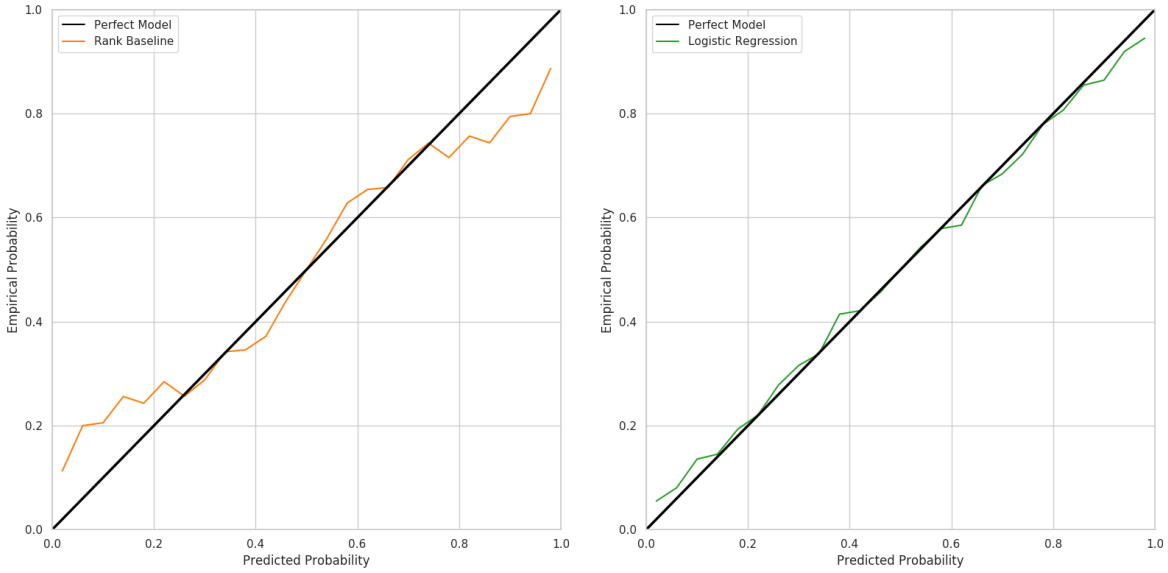

Figure 4.4: Calibration curves for the rank baseline and full logistic regression

4.5 Artificial Neural Network

Artificial neural networks are computing systems loosely based on biological neurons in the brain. Neural networks are inherently more powerful classifiers than logistic regression since they can compute predictions that can not be written as a linear combination of features. This power to model non linear relationships comes at the cost of being more prone to overfiting, more hyperparameters to tune and being computationally expensive.

A visualization of a fully connected feedforward neural network is shown in Figure 4.5. A neural network always consists of at least an input layer and an output layer. In addition,

the network can include a variable number of layers connecting the input and output layer. These layers are called hidden layers.

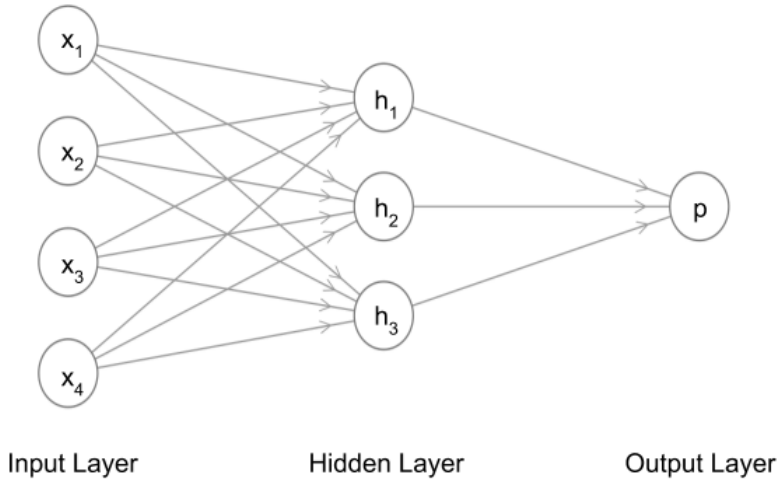

Figure 4.5: Three layer fully connected feedforward neural network

A network is fully connected when all neurons in layer n are connected to all neurons in layer $n+1$. In the figure these connections are depicted by arrows. Each connection between neuron o and neuron p has an accompanying value or weight associated with it w_{o-p}. A network is considered feedforward if the connections between the neurons do not form a cycle.

Each layer consists of a set number of neurons. The number neurons in the input and output layer depend on the problem at hand. For our problem, 84 input neurons are needed, one for each feature extracted in the last chapter. The number of output neurons equals one, as only the probability of a win needs to be predicted. On the other hand, the number of hidden layers and the amount of hidden neurons are hyperparameters.

The values of the neurons in the input layer are equal to the corresponding feature. The values of a neuron h_j in the subsequent layer is computed as follows:

$$h_j = \phi(b + \sum_{i=1}^{84} x_i w_{i-j}) \tag{4.11}$$

With:

$$b = \text{trainable bias weight unique to every neuron}$$
$$\phi(x) = \text{an activation function}$$

The bias weight fulfills the same purpose as the intercept in logistic regression, so is not strictly necessary for our problem. The activation function is used to introduce non-linearity. Popular choices for activation functions are the logistic function (sigmoid), hyperbolic tangent and the rectifier (ReLu) [24], [25].

By computing the value (4.11) for every neuron past the input layer, the neuron values are propagated trough the network layers until they finally reach the output layer. The values of the neurons in the output layer then form the predictions of the neural network. In the case of binary probabilistic classification, one output neuron with the logistic function as activation is used. During training, the network weights are optimized with a gradient descent method using backpropagation to compute the gradients [26].

Notice that a neural network without a hidden layer is equivalent to logistic regression. It is only by adding hidden layers to the network that it becomes capable of nonlinear classification. In fact it has been shown that just one hidden layer with a finite number of neurons is needed to approximate any continuous function. This notion is known as the universal approximation theorem for neural networks [27].

4.5.1 Symmetric Neural Network

The simplifications made to the logistic regression model in Section 4.4.1 are not viable for a neural network. The advantage is that the environment features can be used by the neural network and that nonlinear relationships can be modeled. A drawback is that the model is not inherently symmetric and will need to be trained on both representations of each match, x^1 and x^2.

To make the network symmetric, the predictions for a $Player_1$ and $Player_2$ win in match i

are computed as:

$$h_\theta^{'}(x_{(i)}^1) = \frac{h_\theta(x_{(i)}^1) + (1 - h_\theta(x_{(i)}^2))}{2}$$
$$h_\theta^{'}(x_{(i)}^2) = \frac{h_\theta(x_{(i)}^2) + (1 - h_\theta(x_{(i)}^1))}{2}$$

Intuitively this means that the predicted winning percentage of a player equals the average of that player's predicted winning chances in both feature representations. We verify that the requirement for a symmetric model (4.1) is now fulfilled:

$$\begin{aligned}
h_\theta^{'}(x_{(i)}^1) &\stackrel{?}{=} 1 - h_\theta^{'}(x_{(i)}^2) \\
\Leftrightarrow \frac{h_\theta(x_{(i)}^1) + (1 - h_\theta(x_{(i)}^2))}{2} &\stackrel{?}{=} 1 - \frac{h_\theta(x_{(i)}^2) + (1 - h_\theta(x_{(i)}^1))}{2} \\
\Leftrightarrow h_\theta(x_{(i)}^1) + (1 - h_\theta(x_{(i)}^2)) &\stackrel{?}{=} 2 - [h_\theta(x_{(i)}^2) + (1 - h_\theta(x_{(i)}^1))] \\
\Leftrightarrow 1 + h_\theta(x_{(i)}^1) - h_\theta(x_{(i)}^2) &\stackrel{?}{=} 1 + h_\theta(x_{(i)}^1) - h_\theta(x_{(i)}^2) \\
\Leftrightarrow 1 &= 1
\end{aligned}$$

Another requirement was that the dataset is balanced. By training on both feature representations this is indeed the case.

4.5.2 Inner Validation Strategy

The complexity of neural networks results in quite a bit more hyperparameters to tune compared to logistic regression. It is infeasible to tune every single component of a neural network architecture, so some assumptions are made about the optimal architecture for our problem. Specifically, it is assumed that an architecture with one hidden layer using the rectifier as activation function can lead to satisfactory results.

Furthermore, for computational reasons the neural network hyperparameters are only tuned once and kept constant for all test years. To do this, for each candidate of hyperparameters, a network is trained on all matches before 2014 and validated on all tennis matches in 2014 and 2015. The long training time of the neural network makes Bayesian optimization the logical choice to guide the hyperparameter search.

The resulting tuned hyperparameters are outlined in Table 4.3. The meaning of these hyperparameters is briefly explained below.

Table 4.3: Neural network tuned hyperparameters

Hidden Neurons	Dropout	Batch Size	Epochs	Learning Rate	All Matches?
256	0.45	5696	59	0.00045	Yes

Hidden Neurons

The amount of neurons in the hidden layer. More neurons allow the network to model more complex functions, but also increase the risk of overfitting.

Dropout

Dropout is one of the several options to regularize a neural network. For each training sample, the output of a fraction of neurons in the layer where dropout is applied is set to 0. This forces each neuron to be significant and not depend on other neurons. Dropout was only added to the hidden layer neurons.

Batch Size

Batch size defines how many training samples are used to compute the gradients and update the network weights. A bigger batch size makes the gradients less noisy, but some noisiness can be useful to escape from local minima.

Epochs

The amount of times the training set is fully traversed in the training process. When the amount of epochs is too high the network can overfit to to the training data. If it is too low it might underfit the data.

Learning Rate

Defines the magnitude of the weight updates after after each batch. A learning rate that is set too high will jump over the minima. On the other hand, a low learning rate can get stuck in a local minima.

All Matches?

Whether to use matches from all tournament levels in the training set. Unlike logistic regression, the neural network performed significantly better when trained on matches from all levels of tournaments. This is probably because the nonlinear decision function defined by the neural network can properly discern and learn from the huge amount of low-level matches (Figure 3.1) without negatively affecting prediction for tour-level matches.

4.5.3 Preprocessing

Just like in logistic regression the input features are standardized before training. Standardization is not a requirement for training neural networks, but it often leads to faster training times and better results.

4.5.4 Evaluation

The neural network is trained on the four training sets using the tuned hyperparameters from Table 4.3 and compared to the logistic regression model in Figure 4.6 and Table 4.4. The neural network was implemented using Keras [28] and trained with the Adam optimizer [29].

Table 4.4: Metrics averaged over all matches in the test sets

Model	Logistic Loss	Accuracy
Logistic Regression	0.597	67.4%
Neural Network	0.592	68.2%

The neural network outperforms logistic regression in both accuracy and logistic loss. While the improvement might seem marginal, it can have a substantial impact in the context of betting.

The calibration plot of the two models is compared in Figure 4.7. The two models have an almost identical calibration, with the logistic regression model having a very slight edge. Just like the baseline and logistic model, the neural network has a slight bias of underestimating the winning chances of the underdog. Overall, the neural network appears to be the best model, as it outperforms logistic regression in two of the three metrics.

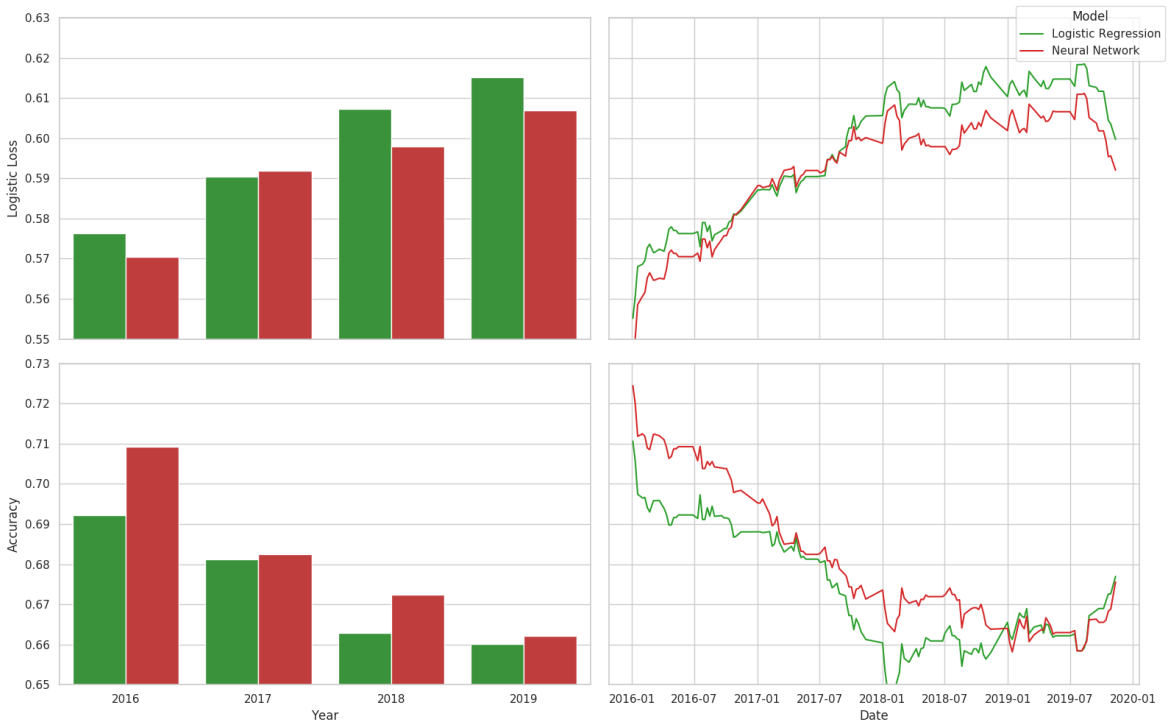

Figure 4.6: Comparison of the logistic regression model with the neural network

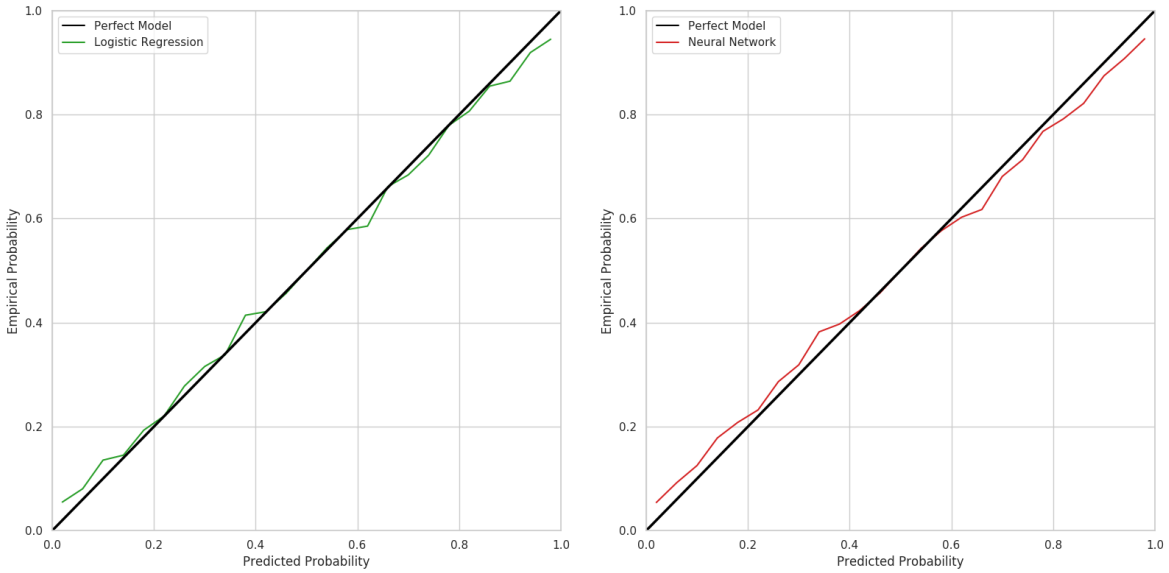

Figure 4.7: Calibration curves for the logistic regression model and neural network

Chapter 5

Results and Experiments

In this chapter we will attempt to make the trained neural network from last chapter profitable in the online betting market. Several betting strategies will be applied to the predictions of the network over all matches in the test years.

The logistic regression model will not be evaluated to avoid the data snooping bias. Consider for example the case where a high quantity of subpar models are evaluated. Due to some randomness in the betting market, it is likely that we could at least make one of these models profitable. However, it would be unreasonable to assume that this model would be profitable in the real world.

Apart from testing our tennis match prediction model on the online betting market, we could also evaluate it by comparing the performance to other approaches explained in Section 2.1. However, the observed variance in model performance over different test years makes comparing models impossible if they are not trained and tested on the exact same data. With this in mind, evaluating the model by betting market simulations seemed like the fairest approach.

5.1 Market Simulations

Using the Pinnacle closing odds obtained in Section 3.2, the online tennis betting market can be simulated. Each simulation will be run starting from 2016 until the end of 2019, for a total of 10099 matches.

5.1.1 Static Strategy Unit Betting

As a baseline, the performance of two static betting strategies is plotted in Figure 5.1. These strategies are considered static because they follow a static betting rule: always unit betting on the favorite or underdog respectively.

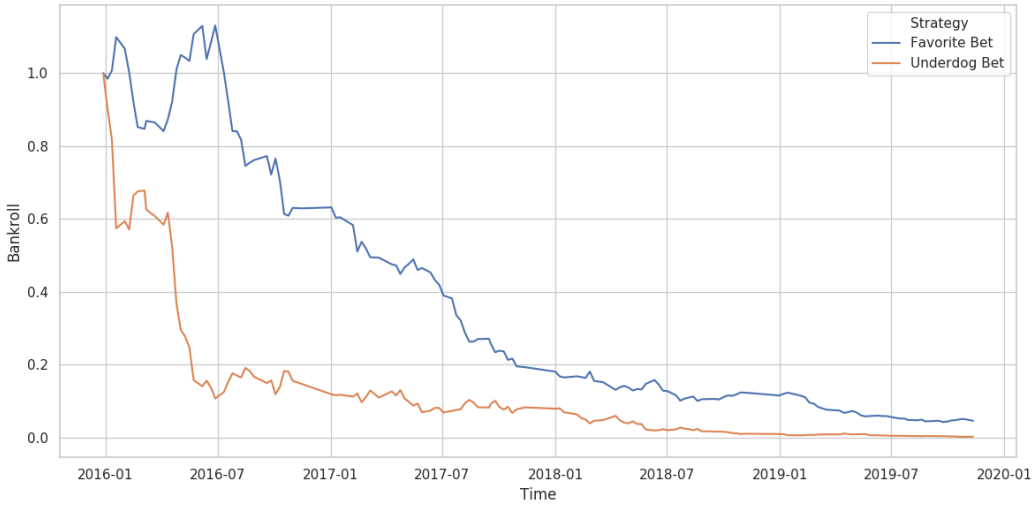

Figure 5.1: Static betting strategies

Unit betting means that the same amount of money is wagered on each bet, namely one unit. it is generally accepted that a unit is equal to approximately 0.5-3% of the current bankroll. In our simulations, one unit is set to 1%.

As expected, these strategies lose almost all of their starting bankroll over a period of four years. Consistently betting on the favorite has slightly better results than betting on the underdog. This is because most of the bookmaker's margin is usually applied to the underdog's winning odds. This phenomenon is known as the favorite-longshot bias [30]. Bookmakers exploit the fact that on average, bettors tend to overvalue underdogs and undervalue favorites, but it also serves as protection against better informed insiders.

5.1.2 Unit Betting With Model

A first improvement over the static betting strategies is to use the neural network's predictions to decide which player to bet on. If the model's winning probability prediction is higher than

the implied probability of the odd, according to the model this bet is profitable. So a bet is made on $Player_i$ if:

$$h_\theta(x^i) > \frac{1}{b^i} \tag{5.1}$$

With:

$$b^i = \text{Odds for a } Player_i \text{ win in decimal format}$$

The simulation for this strategy is plotted in Figure 5.2. While it is a definite improvement over the static betting strategies, it still loses most of the bankroll over the four years.

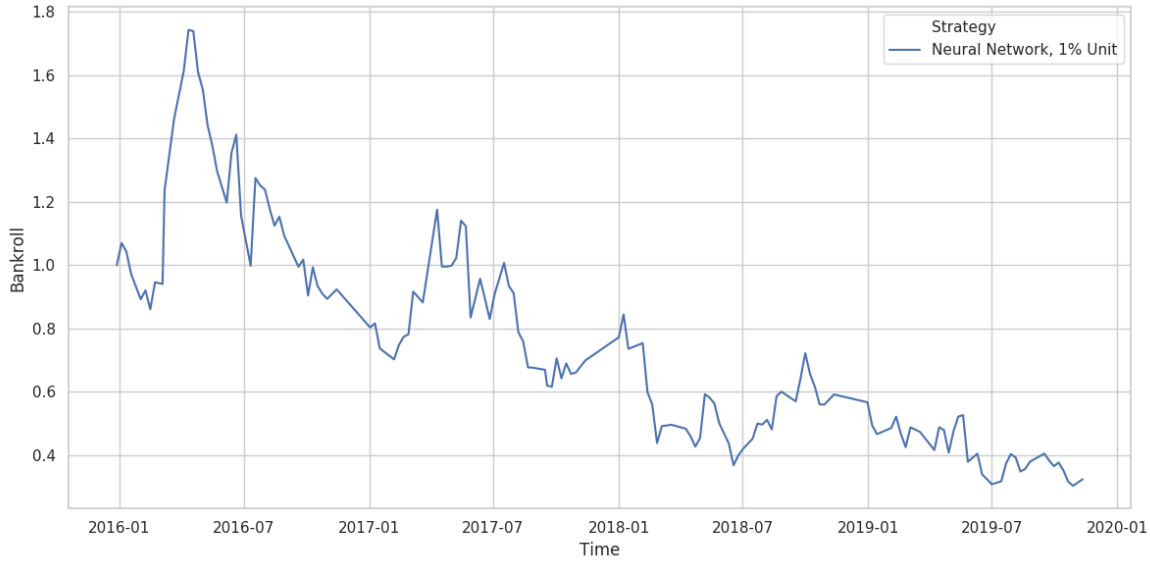

Figure 5.2: Neural network unit betting

Notice when using this betting rule it is possible the model won't bet on either player. This is desirable because when the closing odds are very efficient, which is quite likely as discussed in Chapter 2, betting on that match will always result in an expected loss due to the applied margin. Still, the neural network thinks a profitable bet can be made in 85.6% of matches, which is probably a bit optimistic. The main flaw of this betting strategy is not the model betting on so many matches, but more so betting the same amount each time, regardless of confidence. The Kelly criterion offers a way so solve this problem.

5.1.3 Kelly Criterion

The Kelly criterion is a formula for bet sizing that leads to optimal expected return as the number of bets tends to infinity [31]. Instead of wagering one unit on the bet, a fraction f^* of the bankroll is wagered:

$$f^* = \frac{h_\theta(x^i) \cdot b^i - 1}{b^i - 1} \tag{5.2}$$

This strategy places a bet only if f^* is strictly positive. This is the case when:

$$h_\theta(x^i) \cdot b^i - 1 > 0 \Leftrightarrow h_\theta(x^i) > \frac{1}{b^i}$$

So the same amount of bets are placed as in the last section.

The performance for this betting strategy is depicted in Figure 5.3. Even though the Kelly criterion should in theory maximize the bankroll, it has the worst results of all previously tried betting strategies. This is because Kelly's promise of optimal bankroll growth relies on the assumption that $h_\theta(x^i)$ equals the true probability of $Player_i$ winning the match, which is of course too optimistic since the neural network predictions are only an estimate.

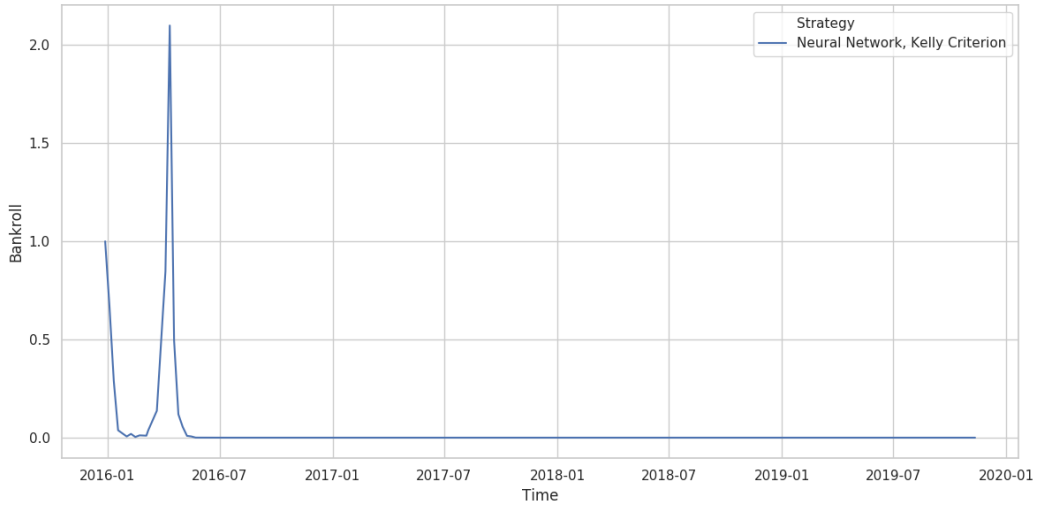

Figure 5.3: Kelly criterion Simulation

5.1.4 Fractional Kelly

A way to protect the model against its own optimism and reduce the volatility is to only bet a fixed fraction f of the amount proposed by the Kelly criterion f^*. This is known as

fractional Kelly [32].

$$f^* = f \cdot \frac{h_\theta(x^i) \cdot b^i - 1}{b^i - 1} \tag{5.3}$$

This approach is plotted for four different fraction values in Figure 5.4. All four simulations lead to a profit over the starting bankroll. Choosing an appropriate fraction value is mainly dependent on the amount of volatility the bettor is willing to tolerate, as higher values for f lead to the highest highs as well as the lowest lows.

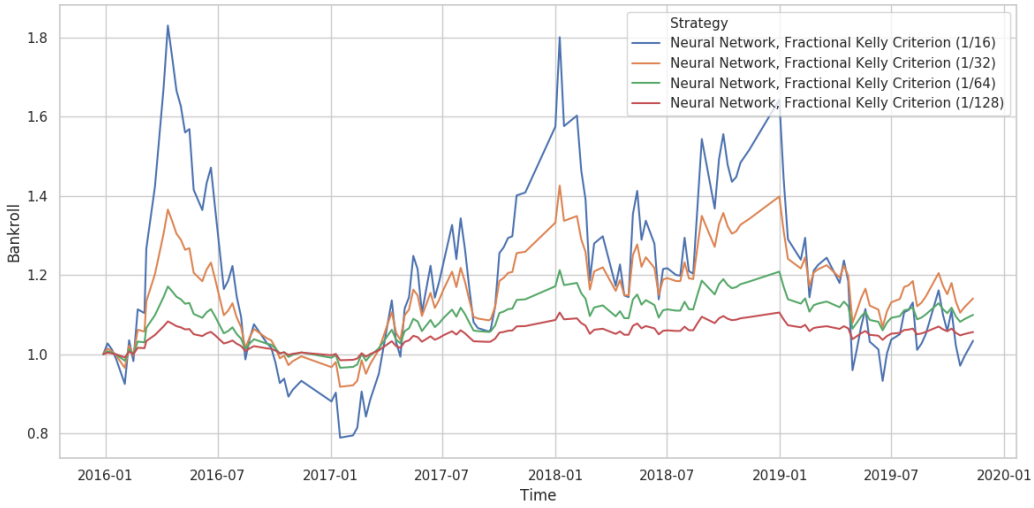

Figure 5.4: Fractional Kelly simulations

Perhaps a better way to visualize how the fraction influences the resulting bankroll is portrayed in figure 5.5. In this graph the resulting bankroll after four years is computed as a function over the chosen Kelly fraction. The simulation reaches break-even with a fraction of $\frac{1}{14.9}$ and optimal bankroll growth is achieved with $f = \frac{1}{30}$ resulting in a profit of 14.2% (depicted by a red dot). Analyzing the graph we can conclude that it is worthwhile to be very conservative in setting the Kelly fraction. When setting the fraction lower than optimal, profit will slowly diminish, but the bankroll will always remain greater than 1. On the other hand, setting the fraction higher than the break even value quickly leads to the risk of losing all money.

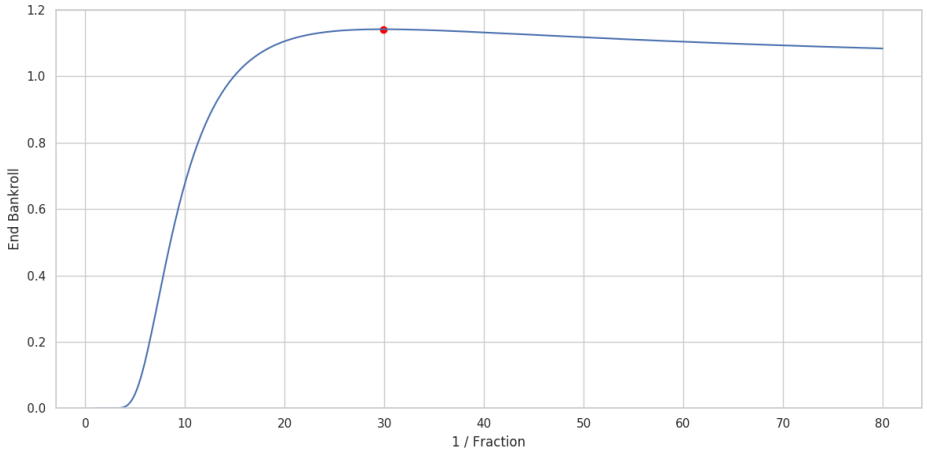

Figure 5.5: Ending bankroll as a function over $\frac{1}{f}$

5.1.5 Uncertainty Shrinkage

Researchers have attempted to tackle the problem of uncertainty in $h_\theta(x)$ trough a more systematic approach. Baker and McHale [33] propose a modified Kelly criterion based on the error variance of the predictions instead of the ad-hoc method used in fractional Kelly, which has no theoretical basis. they suggest to bet a fraction f^* of the bankroll:

$$f^* = \frac{(h_\theta(x^i) \cdot b^i - 1)^3}{(b^i - 1)[(h_\theta(x^i) \cdot b^i - 1)^2 + b^{i^2}\sigma^2]} \tag{5.4}$$

Where again a bet is only made if $f^* > 0$, which again is the case when $h_\theta(x^i) > \frac{1}{b^i}$.

The σ^2 parameter can be interpreted as the error variance of the predictions by the neural network. Consequently, The bets sizes are shrunk proportionally to this uncertainty σ^2. The authors acknowledge it is very hard to directly quantify an optimal σ^2 theoretically, so instead we will run simulations on matches from 2016 and select the σ^2 that leads to optimal bankroll growth. Under the assumption that the uncertainty of the model stays roughly the same, the selected σ^2 should also lead to good results in matches from 2017-2019. This process of selecting σ^2 is shown in Figure 5.6. Indeed we find that the optimal σ^2 for matches in 2016 is close to the optimum for the rest of the matches.

The resulting simulation is depicted in Figure 5.7 leading to a profit of 119.6%. We see that the bankroll seems to follow a rather similar trajectory to the fractional Kelly models, but it

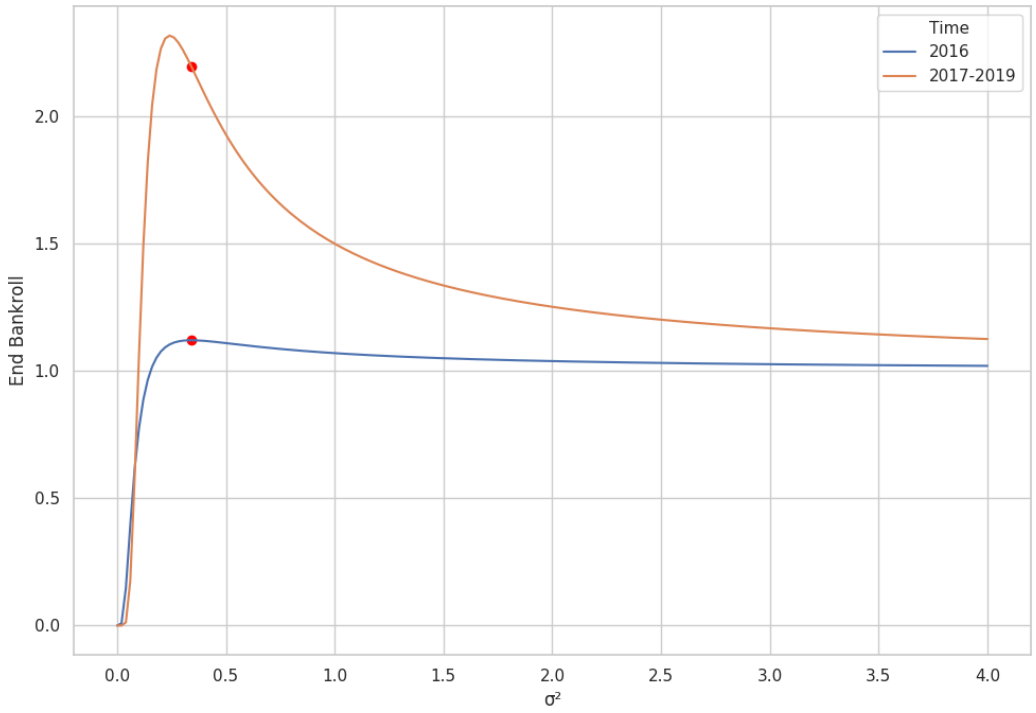

Figure 5.6: Ending bankroll as a function over σ^2

appears to better maximize the gains in the 2017 season while minimizing the loss thereafter.

The bet sizing of this model is also shown in Figure 5.8. Due to the shrinkage, the betting model becomes rather conservative in the bet sizing compared to naive unit betting on every match. Instead of betting 1% of the bankroll every match, the betting model now bets less than 1% in 88% of matches and less than 0.5% in roughly 80% of matches. The shrinkage thus limits the model to only bet large fractions of the bankroll when it is most certain in its predictions, which was ultimately the goal, as discussed in Section 2.2.5.

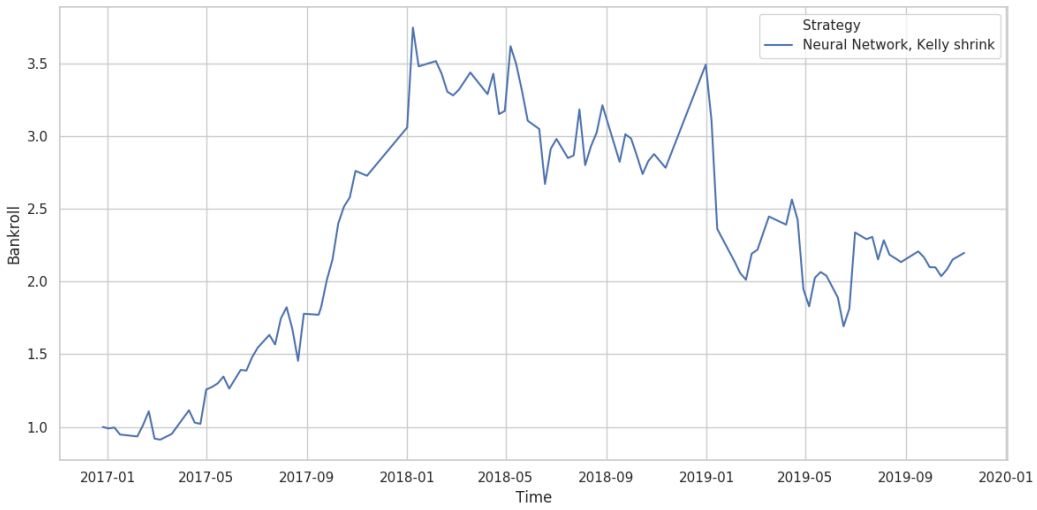

Figure 5.7: Kelly shrinkage simulation

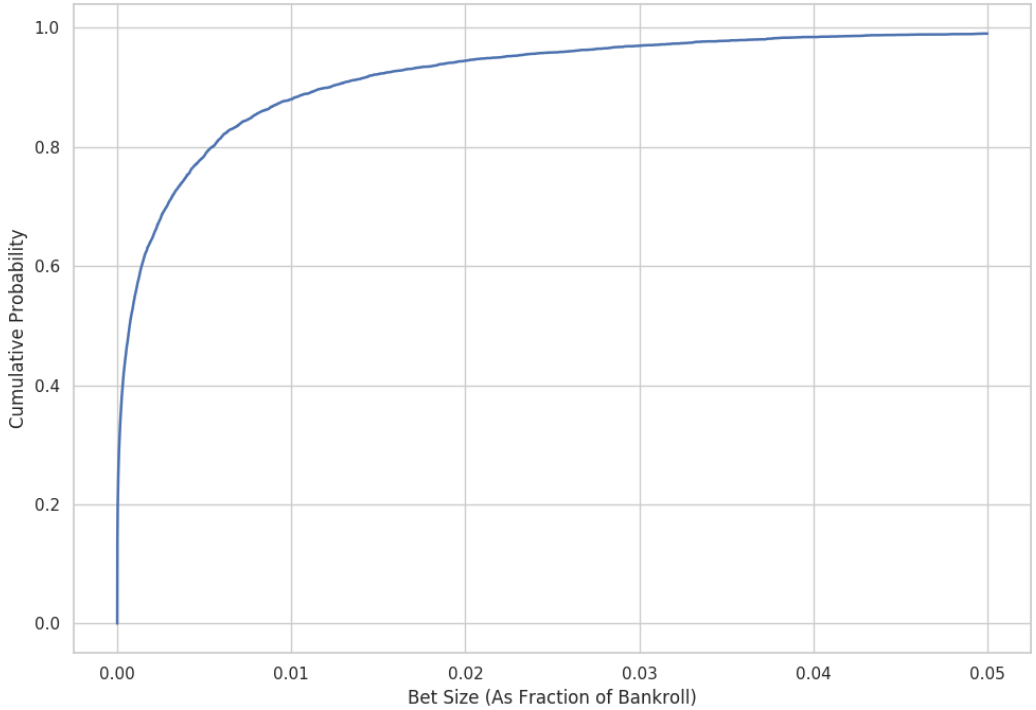

Figure 5.8: Bet sizing

Chapter 6

Conclusion

The goal of this thesis was to build on the current state-of-the-art in pre-match tennis prediction and attempt to turn a profit in the online betting market.

Using open-source data from all levels of professional tennis, an extensive list of features were extracted based on previous research and data analysis. Afterwards a logistic regression model was trained on these features and tested over four seasons significantly outperforming a baseline based on the official ATP-ranking over all metrics: accuracy, logistic loss, and calibration. Seeking further improvement a single hidden layer neural network was trained and compared to the logistic regression model. We found that the neural network further exceeded the performance in logistic loss and accuracy, with only a slight decrease in calibration. For our purpose of predicting tour-level matches, logistic regression attained better results by only training on tour-level matches. On the other hand, the neural network performed better when trained on matches from both tour-level and lower level tournaments.

Finally, the neural network's predictions are made profitable in the online betting market by systematically improving betting strategies. As expected, unit betting and full Kelly bet sizing were incapable of turning a profit due to the efficiency of the closing odds and margin. As hypothesized in Chapter 2, a successful betting model should bet conservatively and only when it has high confidence over the closing odds. With this in mind, fractional Kelly was tried leading to some marginal success with a profit up to 14.2%. Ultimately, the best results were obtained by applying the bet sizing model proposed by Baker and McHale [33] which shrinks the bet size proportionally to the prediction uncertainty, resulting in a profit of 119%

over three tennis seasons (2017-2019).

6.1 Future Work

Some suggestions for future work are given below.

6.1.1 Deeper Neural Network

During tuning of the neural network architecture, the assumption was made that one hidden layer would result in a sufficiently strong classifier. While satisfactory results were obtained using only one hidden layer, adding additional layers could perhaps result in an even better model. This option was not explored in this thesis due to the high computation cost of training deep neural networks.

6.1.2 Match Statistics

Sipko [10] obtained good results by mainly using aggregated match statistics of previous matches as features (e.g. aces, winners, unforced errors, ...). We neglected these kind of features because they are not available for most matches in the raw Sackmann dataset. Adding these features to the current feature set might improve performance, but it would require a richer dataset.

6.1.3 bias

The calibration curves show that all trained models inexplicably exhibit a slight bias of underestimating the winning chances of the underdog. It might be useful to investigate if this bias also exists in other research and if so, how to effectively solve it.

Bibliography

[1] T. Humphrey, *Tennis Trading On Betfair*. Trading Publications, 2014.

[2] B. L. Boulier and H. O. Stekler, "Are sports seedings good predictors?: An evaluation," *International Journal of Forecasting*, vol. 15, no. 1, pp. 83–91, feb 1999.

[3] R. A. Bradley and M. E. Terry, "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons," *Biometrika*, vol. 39, no. 3/4, p. 324, dec 1952.

[4] I. McHale and A. Morton, "A Bradley-Terry type model for forecasting tennis match results," *International Journal of Forecasting*, vol. 27, no. 2, pp. 619–630, apr 2011.

[5] L. Vaughan-williams Chunping Liu Hannah Gerrard, "HOW WELL DO ELO-BASED RATINGS PREDICT PROFESSIONAL TENNIS MATCHES?" Nottingham Trent University, Tech. Rep., 2019.

[6] R. Praet, "Predicting Sport Results by using Recommendation Techniques," Ph.D. dissertation, Ghent University, 2016.

[7] C. Donninger, "SPES: A Service-Points-Elo-System for beating the tennis betting market," 2015. [Online]. Available: http://www.godotfinance.com/pdf/ZoccerTennis.pdf

[8] F. J. Klaassen and J. R. Magnus, "Are points in tennis independent and identically distributed? Evidence from a dynamic binary panel data model," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 500–509, jun 2001.

[9] ——, "Forecasting the winner of a tennis match," *European Journal of Operational Research*, vol. 148, no. 2, pp. 257–267, jul 2003.

[10] M. Sipko, "Machine Learning for the Prediction of Professional Tennis Matches," *MEng Thesis*, pp. 1–64, 2015.

[11] S. A. Kovalchik, "Searching for the GOAT of tennis win prediction," *Journal of Quantitative Analysis in Sports*, vol. 12, no. 3, pp. 127–138, 2016.

[12] Š. Lyócsa and T. Výrost, "To bet or not to bet: a reality check for tennis betting market efficiency," *Applied Economics*, vol. 50, no. 20, pp. 2251–2272, apr 2018.

[13] J. Sackmann, "GitHub - JeffSackmann/tennis_atp: ATP Tennis Rankings, Results, and Stats," 2015. [Online]. Available: https://github.com/JeffSackmann/tennis{_}atp

[14] Tennis-Data, "Tennis Betting — Tennis Results — Tennis Odds," 2007. [Online]. Available: http://www.tennis-data.co.uk/alldata.php

[15] M. E. Glickman, "The glicko system," *Boston University*, no. June, pp. 1–5, 1995.

[16] ——, "Example of the Glicko-2 system," Boston University, Tech. Rep., 2013.

[17] A. M. Nevill and R. L. Holder, "Home Advantage in Sport An Overview of Studies on the Advantage of Playing at Home," Research Institute for Sport and Exercise Sciences, Liverpool John Moores University, Tech. Rep. 4, 1999.

[18] R. H. Koning, "Home advantage in professional tennis," *Journal of Sports Sciences*, vol. 29, no. 1, pp. 19–27, jan 2011.

[19] J. del Corral and J. Prieto-Rodríguez, "Are differences in ranks good predictors for Grand Slam tennis matches?" *International Journal of Forecasting*, vol. 26, no. 3, pp. 551–563, jul 2010.

[20] M. Reid and R. Duffield, "The development of fatigue during match-play tennis," pp. i7–i11, apr 2014.

[21] A. Painsky and G. Wornell, "On the Universality of the Logistic Loss Function," in *IEEE International Symposium on Information Theory - Proceedings*, vol. 2018-June. Institute of Electrical and Electronics Engineers Inc., aug 2018, pp. 936–940.

[22] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review," *International Journal of Forecasting*, vol. 16, no. 4, pp. 437–450, oct 2000.

[23] Scikit-learn, "sklearn.linear_model.LogisticRegression - scikit-learn 0.23.0 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear{_}model.LogisticRegression.html

[24] M. Sukma, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2015.

[25] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," may 2015. [Online]. Available: https://arxiv.org/abs/1505.00853

[26] R. HECHT-NIELSEN, "Theory of the Backpropagation Neural Network," in *Neural Networks for Perception*. Elsevier, jan 1992, pp. 65–93.

[27] B. C. Csáji, "Approximation with Artificial Neural Networks Huub ten Eikelder," Ph.D. dissertation, Eötvös Loránd University Hungary, 2001.

[28] Chollet François, "Keras: The Python Deep Learning library," *Keras.Io*, 2015. [Online]. Available: https://keras.io/

[29] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[30] J. Lahvička, "What causes the favourite-longshot bias? Further evidence from tennis," *Applied Economics Letters*, vol. 21, no. 2, pp. 90–92, jan 2014.

[31] J. L. Kelly, "A new interpretation of information rate," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 185–189, 1956.

[32] L. C. Maclean, E. O. Thorp, and W. T. Ziemba, "Long-term capital growth: The good and bad properties of the kelly and fractional kelly capital growth criteria," *Quantitative Finance*, vol. 10, no. 7, pp. 681–687, 2010.

[33] R. D. Baker and I. G. McHale, "Optimal betting under parameter uncertainty: Improving the kelly criterion," *Decision Analysis*, vol. 10, no. 3, pp. 189–199, sep 2013.