BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN ENGLISH

DIPLOMA THESIS

Simo - Tennis Player Success Prediction Using Deep Learning

Supervisor
Assist. PhD. Alina Delia Călin

Author
Tiutin Andrada-Georgia

2022

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ENGLEZĂ

LUCRARE DE LICENŢĂ

Simo - Predicții În Succesul Jucătorului De Tenis Folosind Deep Learning

Conducător științific
Asist. Dr. Alina Delia Călin

Absolvent
Tiutin Andrada-Georgia

2022

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 Women Tennis Association single tennis matches.

Using an open-source dataset containing historical data from all levels of professional tennis, a total of 41 features were extracted based on previous research and data analysis. The implemented neural network and logistic regression models greatly outperform the models existed before, over loss and accuracy: 93.28% accuracy and 0.02 loss (neural network) and 90.29% accuracy and 0.20 loss (logistic regression).

Finally the models are evaluated over the already finished matches from 2019-2021 and on actual matches that are taking place now.

The paper is structured in six chapters. In the first chapter the motivation behind this thesis and the related work with this subject are presented. Because a model needs data to be trained, in chapter 2 we will see the features extracted, which are taken into consideration. Starting with chapter 3, two models are described from a theoretical point of view. In chapter 4, the detailed implementation of the application Simo is presented, ending with the results obtained in chapter 5. Finally, in chapter 6 the final conclusions are drawn.

The proposed models are based on an original viewpoint. The novelty consists in using a large number of features, in contrast with other models where a significantly small number of features were used. This is the main reason for which a high accuracy was obtained. Likewise, they are the first models which predicts women's tennis matches, all the other attempts are based on men's matches. Besides these, I implemented four algorithms which use the artificial neural network model, in order to predict which young players have a big probability of becoming a tennis star, which players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Chapter Outline

The paper is structured in six chapters. In the first chapter the motivation behind this thesis and the related work with this subject is presented. Because a model needs data to be trained, in chapter 2 we will see the features extracted, which are taken into consideration. Starting with chapter 3, two models are described from a theoretical point of view. In chapter 4, the detailed implementation of the application Simo is presented, ending with the results obtained in chapter 5. Finally, in chapter 6 the final conclusions are drawn.

1.2 The Background Of Tennis

Tennis represents a sport which is played in two ways: single (against another player) or between teams of two players, using a racket [Bar]. A tennis match can take place on four different surfaces: hard, clay, grass, carpet. The purpose is to win two sets composed of 6 games each. A game is won when a player wins 4 points. In case both of them win 3 points, the game will continue until one of them will win two consecutive points. The official rules are posted by the International Tennis Federation and are found online [1]. Likewise, the tournaments are split in seven levels, by their importance. The most important category is Grand Slam, where there are only 4 tournaments in a year: US Open, Wimbledon, Roland Garros, Australian Open. For each match won, a player receives prize money and a number of points, depending on the category of the tournament and the reached round in the tournament. These points are used in order to create a rank with all players, deciding who are the best players.

[1] https://www.itftennis.com/en/about-us/governance/rules-and-regulations/

During the last decade, tennis became one of the most popular sport. Millions of people are watching each tournament, supporting their favourite player on the road of becoming the best player in the world. Its first evidences are dated back in the 11th century in France, where a raw form of tennis was practiced by monks in monasteries [Fre], as it can be observed in the Figure 1.1. As years passed by, the sport developed, outlining its rules, creating different playing surfaces and producing more and more performant rackets and sport equipment.

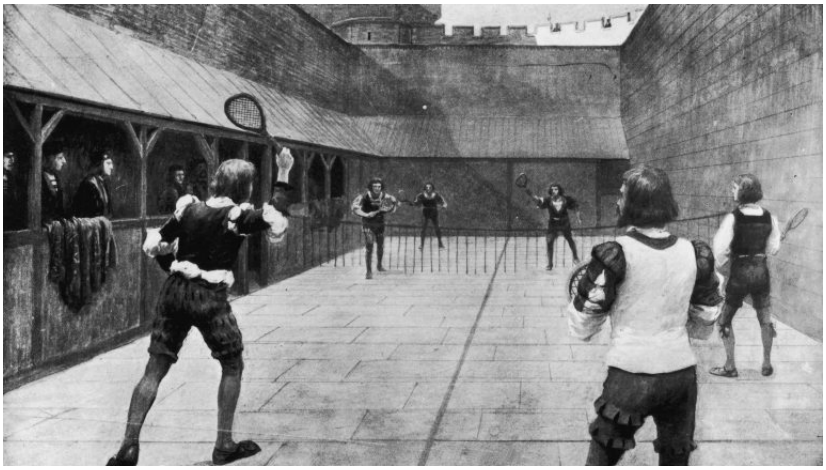


Figure 1.1: Raw form of tennis played in monasteries. Figure source [Foy]

1.3 Problem Statement

In the history of sports, many people have had the tendency to try to predict the outcome of a game. In the last decade, the domain of tennis predictions has seen an important development. More and more people have been attracted to this phenomenon, so a couple of millions of dollars are spent each year in this field. This thing is facilitated due to the large number of tennis events that take place during a year (over 50). It is considered to be the second most important sport, after football, in the domain of sports betting. What attracts so much people to tennis betting is the fact that only two people are involved in a single game (contrary to most sports), so there are chances for both of them to win, especially if the better ranked player is having an injury or a bad day [Hum14].

There are two types of predictions in tennis: pre-match and in-play. The first one, tries to anticipate the result of a match before the start of it and the second one will present the evolution of the winning probabilities during the match [Pet17]. In this paper, we will focus only on pre-match predictions.

In 1999, Boulier and Stekler created the first mathematical model which tries to predict the result of a match. The predictions were made based on the ranking

of both players. Starting from that point, many scientists have starting to create new machine learning models to obtain better results and a big accuracy. The most popular models are the hierarchical ones, described by Barnett and Clarke (2005) or Knottenbelt (2012).

These models are based on the scoring system and will use a Markov chain, where each probability is established using the percentage of service of each player. Also, there exists approaches which use artificial neural networks or regressions models. One of the first models of this types were created by Sipko and Knottenbelt (2015) and they obtain better results than the previous existing models [Pet17]. Thus, the objective of this thesis is to continue on this branch, using information already existed in this domain. Basically, there will be data about a tennis match and we will try to predict as accurate as possible for each player, the chances of winning the match [DS20]. It will be taken into consideration data about the tournament and characteristics of each player.

1.4 Existing Solutions

In the domain of tennis predictions there are three main categories of approaches: regression/neural network, pairwise comparison and hierarchical/point-based. We will go through each of them and see how they were implemented.

1.4.1 Neural Network And Regression Models

This type of models use as data, a set of features for each player. The features used can vary from one implementation to another. In the beginning, the teams formed by Boulier and Stekler (1999) [BS99] and the one formed by Clarke and Dyte (2000), used as main features the number of points and the place in the rankings for the players. Then, later on, features like age, height, the number of wins against the opponent, the hand used were taken into consideration and better results were obtained. In 2015, Sipko and Knottenbelt have proved the efficiency of adding features representing different percentages: aces, double faults, first serve, break points faced/saved, victories in the last games or on different surfaces [SK15].

After the set of features is created, a function takes it as input data and predicts the chances of each player to win a match. The function used can be either a regression or a neural network with parameters learned during training [Pet17].

1.4.2 Pairwise Comparison Models

This type of model, assigns to each player a positive parameter (a metric), representing the overall skill. In 1952, the pair of researchers, Bradley and Terry, discovered a famous kind of pairwise comparison model, which is comprehensively used in artificial intelligence and in different sports [BT52]. The model is called Bradley-Terry, after its discoverers. It predicts the result of a paired comparison between two persons X and Y, given the respective parameters x and y. Thus, the probability is computed as:

$$P(X\,beats\,Y) = \frac{x}{x+y} \tag{1.1}$$

In 2011, McHale and Morton, developed an application for tennis predictions, which uses such a model. For each player, it was assigned a single metric which is adjusted using the percentage of winning in a fixed period of time [MM11].

Similar with Bradley-Terry model, pairwise comparison can be done using rating systems. A useful rating system is the Elo, created by Arped Elo with the purpose to reflect the skills of chess players [Pet17]. However, this type of system updates the skills of a player after each match, being a dynamic one.

1.4.3 Point Based Models

A match of tennis is composed by sets, which in turn are composed by 6 games each. A game is won when a player wins 4 points. In case both of them win 3 points, the game will continue until one of them will win two consecutive points. The match ends when the first player wins two sets. Draws are not allowed in tennis.

Point based models are types of models which look for the player's chance to win a point, either on serve or on return. It is assumed that each point in a match is distributed identically and independent from the other points [Pet17].

In 2001, Klassen and Magnus, tried to prove that the points are identically and independent distributed. They reached to the conclusion, that the assumption is false. If a player wins a point, it will have a positive impact on winning the next one. Even though, they reached to the conclusion that the hypothesis is not true, they stated that the deviations are significantly small. For must purposes it is fine to use a constant probability for a match [KM01]. Using the theory, in 2003, they developed an approach to approximate player's probability to win a point, by modelling the match using a Markov chain as in Figure 1.2. The states will be represented by a score [KM03]. So, if there is the probability to win a point on serve, then the chances to win the next point is computed in closed-form.

Establishing this model of tennis predictions, the last thing that needs to be

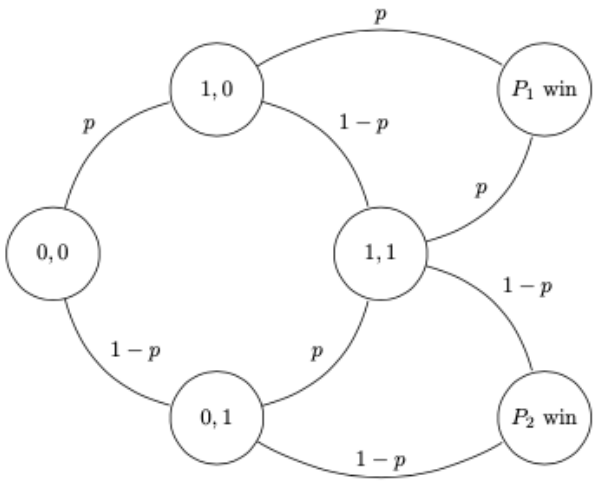


Figure 1.2: Markov chain. Figure source [Pet17]

solved is the method to determine player's probability to win a point by serving. The simplest way to do this is to average the frequency of serve winning points and return for each player.

1.4.4 Discussion On Approaches

Neural Networks models and point based models use as input features different percentages like serve and ace probabilities. Usually, these features are not known, but they can be derived by averaging historical statistics. Thus, these two types of models are in contradict with the approach suggested by McHale and Morton (2011), which uses the model Bradley-Terry, where the player's feature metric is learned by model. Unlike other models which use lower level point information, McHale and Morton apply their model to game data level. Nonetheless, their idea can be extend to point level infomation very easily. So, it will provide a way to predict many input features used by other models like: service percentage, ace percentage, double fault percentage and so on. The only problem with this approach is the fact that non-transitive relation between players is not allowed.

The majority of the models presented so far have the same plan on dealing with the most important factors: the effects of surface and the variation on player skills through time. It means that the input data is filtered based on its relevance. Thus, the most recent matches played on the same surface are the most relevant. Sipko and Knottenbelt (2015) showed that filtering by surface provide better performance than treating all the surfaces the same. However, the model can't learn features of the surfaces themselves, so it won't be able to generalise between surface types. For instance, when almost the entire data of a player is on one surface.

1.5 Novelty Of The Proposal

As we will see in the next chapters, the model proposed in this thesis, for predicting the outcome of a match will reach an accuracy of 93%, outweighing the models presented before.

The model obtained will be used further in order to try to predict which young players have a big probability of becoming a tennis star, which players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament. This second part of the thesis is very little explored, finding only one experiment regarding these topics. In the following, it will be presented this experiment.

1.5.1 Predicting The Success Of A Player Using Binary Classification Methods

The model is constructed starting with the performance of a player in his first 40 matches. Its purpose is to predict the success of a player in the world of tennis. This prediction becomes a binary classification problem. It was resolved using two deep learning methods: Neural Network and Logistic Regression.

After a detailed analysis of the ranking system, in order to determine the player's success, the top 30 ranking was used as a metric. Thus, the classification of a player's success was used to predict if the corresponding player will reach top 30 at some point.

The machine learning models were trained using extracted features from a data set containing matches from 2000 to 2017. Hyperparameter tuning and feature selection were performed on the models for optimization reasons.

The results obtained by testing the models on a test set are: 82% for Logistic Regression and 78% for Neural Network. [Vom17]

1.6 Publication

A paper based on this thesis, was submitted to the Student Scientific Communications Session, organised by the Babes-Bolyai University in collaboration with Technical University from Cluj-Napoca. Its status is still pending, until 20 June, when the accepted papers will be announced.

Chapter 2

Data Processing

One of the most important piece of a good learning algorithm is represented by the dataset. The learning process can not take place if the data used is poor. Usually, it is very difficult and expensive to create a high-quality dataset. But happily, there exists an amazing open source dataset for tennis matches.

2.1 Data Used

The data used was taken from an open source database, created by Sackmann Jeff and published online on GitHub [Jef]. The database contains information about every match that took place since 1920 up till present. The matches that are considered, are the ones from Grand Slam, Women Tennis Association Masters 1000, Women Tennis Association 500/250/128 Series, including also the Olympic Games and The Cup of Federations. Regulary, are added new matches in the dataset, and it covers almost 1000000 matches.

2.2 Cleaning The Data

From the dataset, are removed the matches that were stopped before completion. They aren't relevant to the model used for learning, because they are usually the result of disqualification or injuries. The ability of a disqualified or injured player is not reflected in a proper manner in the outcome of a match, thus it is not helpful for the learning process. After this step of cleaning, almost 4% percent from data is removed.

2.3 The Extraction Of Features

A machine learning model, needs a dataset of training examples, in order to learn the correspondence realized between the input and the output values. Thus, the dataset must contain a useful set of features, where each record is represented by a tennis match, together with the output value (the player who won the match together with the player who lost the match). This step represents feature extraction.

Before performing the step of feature extraction, the dataset is sorted by date, in order to use only matches from the past, because data from the future is not relevant for the learning process.

2.3.1 Types Of Features

The features extracted can be split into two categories: player features and environment features. The player features are not equal for both players and they can be represented by: birth date, rank, number of points and so on. We need two values to model such a feature. On the other hand, the environment features are the same for each player, so we will use only one value to model them. The tournament, the surface of the court, the level of the tournament are examples of such features.

2.3.2 Target Value - Neural Network Model

The output value is defined as:

$$OutputValue = \begin{cases} (1,0), & Player\ 1\ Won,\ Player\ 2\ Lost \\ (0,1), & Player\ 1\ Lost,\ Player\ 2\ Won \end{cases}$$

It can be observed in the Table 2.1 that the output value can be easily changed from (1, 0) to (0, 1) or the other way around, by simply changing the order of the player 1 and player 2 in the input value.

Table 2.1: A match where player1 is the winner, represented in two ways - Neural Network

Tournament features	Player1 Features	Player2 Features	(1,0)

Tournament features	Player2 Features	Player1 Features	(0,1)

2.3.3 Target Value - Logistic Regression Model

The output value is defined as:

$$OutputValue = \begin{cases} 1, & Player\ 1\ Won \\ 0, & Player\ 1\ Lost \end{cases}$$

It can be observed in the Table 2.2 that the output value can be easily changed from 1 to 0 or the other way around, by simply changing the order of the player 1 and player 2 in the input value.

Table 2.2: A match where player1 is the winner, represented in two ways - Logistic Regression

Tournament features	Player1 Features	Player2 Features	1

Tournament features	Player2 Features	Player1 Features	0

2.3.4 Environment Features

As environment features we will use a triplet formed by: the tournament, its surface and level. In the beginning, we will assign a random value to each possible tournament, surface and level, so these three values will be used as features.

In the dataset, there exists 60 different tournaments, and they are used as features because previous successes in a tournament can help the player to be more determined in the future editions.

The surface has an essential influence on the expected result of a match. It is proved that the players have some surfaces where they obtain better or worse results. There can be four types of surfaces: clay, hard, carpet, grass. This feature is not sufficient on its own. It is useful to add player features that reflect past performances on these surfaces, as we will see in the next sections.

There are 8 types of levels in tennis: Grand Slam, Women Tennis Association Masters 1000, Women Tennis Association 500/250/128 Series, Olympic Game, The Cup Of Federations (Billie Jean King Cup) and The Women Tennis Association Finals. The higher victory's level is, the more important it is for the player's confidence and for her place in the world rank.

2.3.5 The Rating System

A feature used to reflect the player's skills is represented by the rating systems, provided by Women Association Tennis. A player receives points depending on how far she got in a tournament, with more important tournaments being worthy of more points. The points earned by a player, are dropped after one year since the tournament took place. So, a player's Women Tennis Association-ranking points represent the performance in the last year.

Besides the number of points, the position in world ranking is used as a feature. If a player is situated on the first positions will have a lot of benefits like: at many tournaments will skip the first round and it will paired in the first match with one of the worst ranked player that participate at a given tournament, which will clearly give her a big advantage.

2.3.6 Hand

The hand used is also added as a feature. 1 represents that a player is right handed and 0 is for left handed.

There exists a hypothesis which says that the left handed players have a little advantage when playing against a right handed player. The explanation is that because the number of left handed players is relative small, it is hard for the opponents to anticipate their movements or are used with strategies that play directly the left handed players' strengths (for example, hitting the balls toward the right side) [Ste].

The hypothesis that the left handed players are better athletes, comes from the fact that their brain's right hemisphere is responsible for both the visual spatial awareness and the dominant hand [Ste]. In fact, some of the most valuable players were left handed: the 18 times champions of grand slam Martina Navratilova, the former number one Monica Seles, the two times Wimbledon winner Petra Kvitova, the former number one and three times winner of grand slam Angelique Kerber and so on.

2.3.7 Height

Height is an important characteristic in tennis because it can make a player's serve harder to return due to its bounce, height and power. Also, it can be an advantage for pace, angles, reach, on the groundstokes or at the net.

Lack of stature does not represent a required criteria to particiapte in a tournament. In fact, the US Open champion Sloane Stephens has 173 cm. The former number 1 and three times grand slam winner, Ashleigh Barty has 167 cm .The top Romanian player, Simona Halep, has 164 cm [Stu].

The serve represents the aspect in which small players can suffer. Because they don't hit the ball down, they can't produce the same power as taller players. But, they have advantage over the taller players on returning the serve [Stu].

Thus, in our model the height will be modeled as another feature.

2.3.8 Fatigue

Another important feature is represented by the fatigue. As in any sport, if fatigue is accumulated, it will have a harmful effect on a player's performance. It has been proved, that in tennis, the function of skeletal muscle decrease after long matches. At the same time, these negative effects are increased after consecutive days of matches.

Even tough the fatigue is not desirable, it should be avoided long periods of inactivity. In Figure 2.1, it can be observed that a period of maximum three weeks of break can increase the player's chances of winning the next matches. This break is welcomed especially after a series of long tournaments. But, more than three weeks of inactivity can lead to rustiness or bad results.

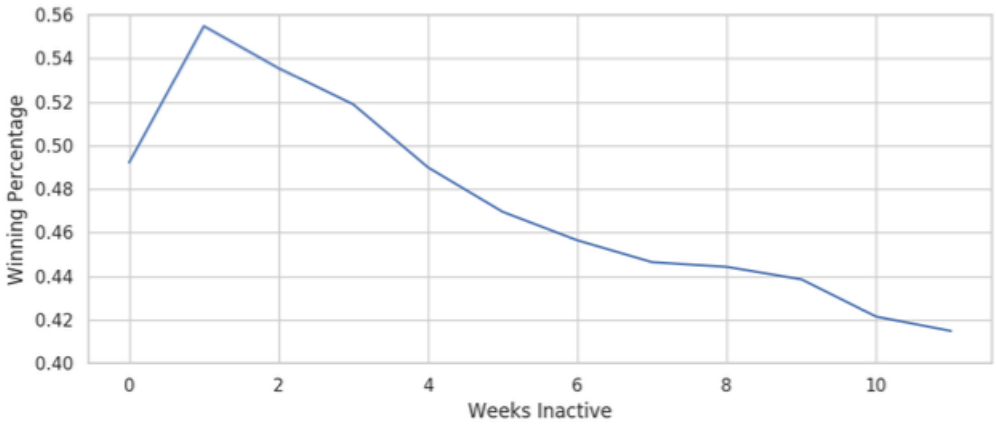


Figure 2.1: Winning percentage over inactivity. Figure source [DS20]

2.3.9 Age

In 2010, Juan Prieto-Rodriguez and Julio del Corral proved that the difference of age between two opponents has an important effect on the match's result. For instance, when a high-ranked player plays against a younger player, her chances of winning the match decrease. But, when the better ranked player is 15 years younger than her opponent, her chances of winning the match are with 20% higher than playing against an opponent with the same age. So, the age is a valid candidate to be chosen as a feature. [DCPR10]

2.3.10 Percentages

A set of 10 percentages features, are calculated for each player, representing their tennis skills. It is composed by: victory percentage (in all matches played), clay/grass/hard/carpet victory percentage (only one is chosen as feature, depending on the surface on which the match takes place), ace percentage, double fault percentage, winning on first serve percentage, winning on second serve percentage, first serve success percentage, overall win on serve percentage, breakpoints faced percentage, breakpoints saved percentage.

These percentages are very useful when trying to predict the result of a match, because they represent the strongest and the weakest points of a player. We can observe, that the most percentages are related to the serve. That is, because the serve is the key to a match. Every point, can not start without a player serving, and its evolution depends very much on how the ball was served. For example, when the player who is serving tends to lose the match, will be very careful on how she will serve, but if she is in advantage, she will risk on serve in order to obtain an ace.

2.3.11 Last Matches Won

Another important feature used is the percentage of victories in the last 5 matches. Of course, a high percentage means that the player is in good shape which is beneficial for her confidence. So, there is a big chance that she will continue in the same manner, obtaining victories.

Likewise, the victories or the losses can lead to a better or a worse position in the world ranking. This fact is important for the future tournaments, where the position is crucial when creating the draw. So, a good position can lead to easier matches in the first rounds, while a lower position means that the player will face the favourites players to win that specific tournament.

2.3.12 Last Matches Won On Surfaces

A similar feature with the previous one is the percentage of victories in the last 5 matches, on a given surface. The value for this feature will be computed differently, depending on the match's surface that is used for training or testing.

A player with great results on a surface, will continue to dominate it and to obtain victories. The same goes for a surface, where she does not have great results.

Usually, each player has a favourite surface where she obtains the most notable results in the career. For instance, Simona Halep won almost half of her titles on clay, one of them being the grand slam Roland Garros. The world number one player, Iga Swiatek won as well half of her tournaments, on the same surface, hard.

2.3.13 Win Percentage Over The Opponent

The last feature used is represented by the percentage of victories in the last 5 matches, against the opponent. The sum of these two percentages for each player must be equal with 100, in the case that there is at least one match between them.

When this percentage is equal with 0, that means that the players did not play against each other yet, so this feature will not be very representative. But, when the percentages are like: 80%-20%, 100%-0%, the player with the biggest percentage tends to dominate the other player. If we take as example Simona Halep, she is one of the best player in the world. And of course, she can beat any player. But, in the matches against Serena Williams, we can observe the domination imposed by Serena. Even tough, Halep is in top, she has a percentage of 20%-80% against Serena.

The same goes when the percentages are very close to each other. It means that both of them succeeded in beating the opponent at some point in time, so their confidence before the match will not be affected.

2.3.14 Summary

To sum it up, a table with all the features is created. So, in Table 2.3, we can observe all those 41 features: 3 environment features and 38 player features. These features together with the target value will be used as input and output values by the machine learning models presented in next chapter.

Table 2.3: The Features Extracted

Category	Type	Number of Features Extracted
Tournament	Environment	1
Surface	Environment	1
Level	Environment	1
Rank	Player	2
Points	Player	2
Hand	Player	2
Height	Player	2
Fatigue	Player	2
Age	Player	2
Percentages	Player	20
Last Matches Won Percentage	Player	2
Last Matches Won On Surfaces Percentage	Player	2
Win Percentage Over The Opponent	Player	2

Chapter 3

Predictions System

In this chapter, there are presented two machine learning models, used to predict the outcome of a tennis match. First, a logistic regression model will be presented, then a neural network one, using the features extracted in the last chapter. The models will be evaluated using the same metric.

3.1 Symmetric Binary Probabilistic Classification

For a model to be good, it is not enough only to predict if a player will win or lose a match. Thus, we need to create machine learning models, which are capable to predict also, the chances of winning/losing for each player.

It would be a nice idea to be able to train the models using the probability of each player to win a certain match, but unfortunately it is not realizable. We know if a player won or lost a match, not the probability.

Further, we will use the name of probabilistic classification in order to refer to the classifiers which are able to give the probability distribution for a set of classes. A classification which has only two possible classes is called binary classification. So, if we know for a player that the chances of winning the match is *p*, then the chances of losing is equal to *1-p*.

Our models will need to be symmetric. As we saw in the last chapter, the match can be represented in two ways, by simply changing the target value. A model is considered symmetric, if a player's probability of winning is *p*, then the other player's probability of winning is equal to *1-p*. So:

$$P(Player1_{(i)}) = 1 - P(Player2_{(i)}) \tag{3.1}$$

where *i* represents a certain match.

3.1.1 Metrics

The metrics represent functions used to measure the quality of a machine learning model. They are similar with the loss function, except that their results are not used in the training process.

The accuracy is one of the easiest metric used for classification. This metric computes as fraction the number of matches correct classified over the total number of matches used. The advantage of this metric is that it is easy to interpret, but it does not take into consideration the predictions' confidence.

3.2 Logistic Regression

Logistic Regression represents a model known for its fast training time and being less vulnerable to overfitting. It belongs to the linear category of classifiers. Using the logistic function it models a binary variable.

In Figure 3.1, we can observe that the domain of values for the logistic function is [0,1].

Suppose, that we have n features in our model: $f_1, f_2, ..., f_n$. Then, the model will contain n+1 parameters (weights): $\theta_0, \theta_1, ..., \theta_n$. The prediction is computed as:

$$\phi(\theta_0 + \theta_1 f1 + ... + \theta_n fn) \tag{3.2}$$

where $\phi(x)$ represents the logistic function. Because the values possible for the equation 3.2 is the interval [0,1], it can be easily considered as a probability.

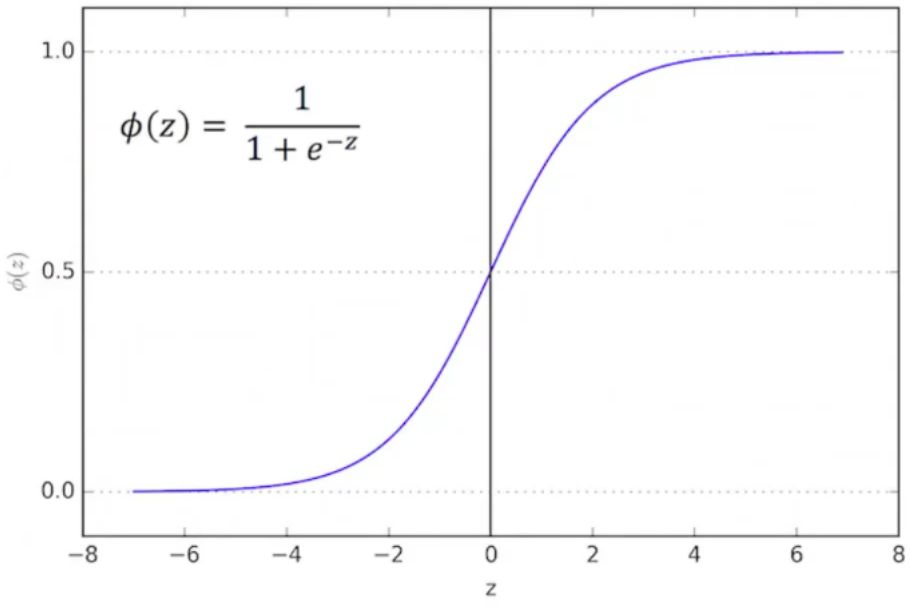


Figure 3.1: Logistic Function. Figure source [Sur]

By minimizing the loss on predictions, the weights are optimized. Usually it is done using the gradient descent algorithm.

3.2.1 Cost Function

The role of the cost function is to mirror the model's behaviour. Basically, it computes how close are the actual and output values.

In the case of linear regression, mean squared error is used as the cost function. But, in the case of logistic regression it doesn't work because it might give a non convex solution, containing many local optima, as in the Figure 3.2.

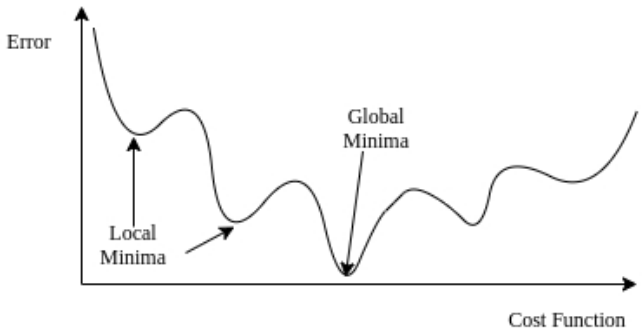


Figure 3.2: Cost Function. Figure source [Ayl]

Thus, we can't find an optimal solution in this case, using the gradient descent method. A logarithmic function is used as the cost function. This function allows us to run the gradient descent method, by being convex for all the input values and having only one minimum.

In the case of binary classification problem, the logarithmic function depends on the value y. Thus, the function is defined as:

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & if\ y = 1 \\ -log(1 - h_\theta(x) & if\ y = 0 \end{cases}$$

where $h_\theta(x)$=$\frac{1}{1+e^{(-\theta^T x)}}$.

The graphs of the function are 3.3

Because the output value can be only 0 or 1, the equation can be simplified as:

$$cost(h_\theta(x), y) = -y_{(i)} * log(h_\theta(x_{(i)})) - (1 - y_{(i)}) * log(h_\theta(x_{(i)})) \tag{3.3}$$

where *i* represents a certain match.

In the case of n matches, the formula becomes:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[-y_{(i)} * log(h_\theta(x_{(i)})) - (1 - y_{(i)}) * log(h_\theta(x_{(i)}))] \tag{3.4}$$

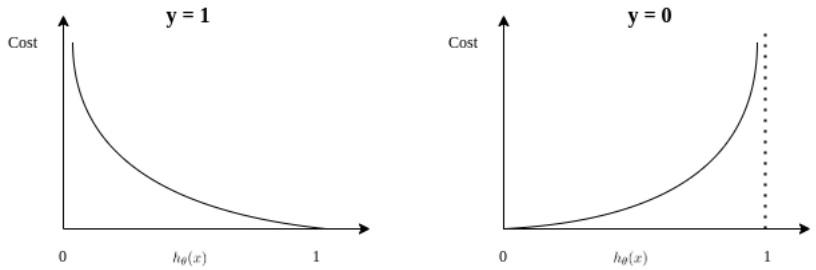


Figure 3.3: Cost Function. Figure source [Ayl]

3.2.2 Gradient Descent Method

The gradient descent method represents an optimal algorithm which has the ability to find the minimum value of a differentiable function. Different values are applied until the best ones are found, which minimize the error while minimizing the cost of the model's parameters.

The method is applied to the cost function, in order to find an optimal solution.

To minimize the cost function, the gradient descent method is run on each parameter θ_j:

$$\theta_j = \theta_j - \alpha * \frac{\partial}{\partial \theta_j} * J(\theta) \tag{3.5}$$

where

$$\frac{\partial}{\partial \theta_j} * J(\theta) = \frac{1}{n} * \sum_{i=1}^{n} (h_\theta(x_i) - y_i) * x_{j(i)} \tag{3.6}$$

3.2.3 The Simplification Of The Model

In case of logistic regression, the prediction is computed as a linear combination of features. Due to this fact, we can reduce the number of model's parameters. Thus, it will make the model less vulnerable to overfitting.

The environment features should be ignored, so their corresponding parameters are set to 0. If we take as an example an environment feature whose weight would be greater than 0, then it would increase either the player1's chance of winning or the player2's chance of winning, depending on the input's form conform with the Table 2.3.3. So this is a contradiction. The same goes if the weight has a value less than 0. Thus, in case of logistic regression, the environment features are not useful.

The intercept is a notion very used in the domain of logistic regression, and it refers to the parameter $p0$. Suppose that we have a match, where there is no existent information, so no features known. Thus, the probability will be equal with $\phi(p0)$.

The probability of the match, of course should be 0.5. By solving the equation:

$$\phi(p0) = 0.5 \tag{3.7}$$

we obtain that p0=0.

In our case, $p0$ is set to 0, to avoid training it to noise.

3.2.4 Balancing The dataset

The dataset must contain an equal number of wins and losses. So, the data will be split in two equal parts, half of it with the target value 1 and the other half with the target value 0. Obviously, the order of ther player1 and the player2 in the input data must be swapped accordingly.

3.2.5 Standardization

Scaling of features represents an important step in the process of working with datasets. The data contains different types of features of various dimensions and scales. So, it will affect the data in a negative way. It leads to a biased outcome of predictions in terms of accuracy rates and misclassification error.

In this context, the standardization appears. Represents a scaling technique, which makes the data scale-free, using the formula:

$$z = \frac{x - \mu}{\sigma} \tag{3.8}$$

The data set scales with a zero mean and a unit variance.

3.3 Artificial Neural Networks

Artificial Neural Network is a model which tries to simulate the work performed by the neurons in the human brain. It is more powerful than a logistic regression model since it can make a prediction which does not have to be a linear combination of features. But this advantage comes with a cost by being more vulnerable to overfitting and computationally expensive. It also has more hyperparameters to be tuned.

In the Figure 3.4, it is represented a fully connected feedforward network. It always contains one input layer and one output layer. Besides these two layers, it can also contain hidden layers, which connect the input layer to the output layer. Each layer is composed by neurons.

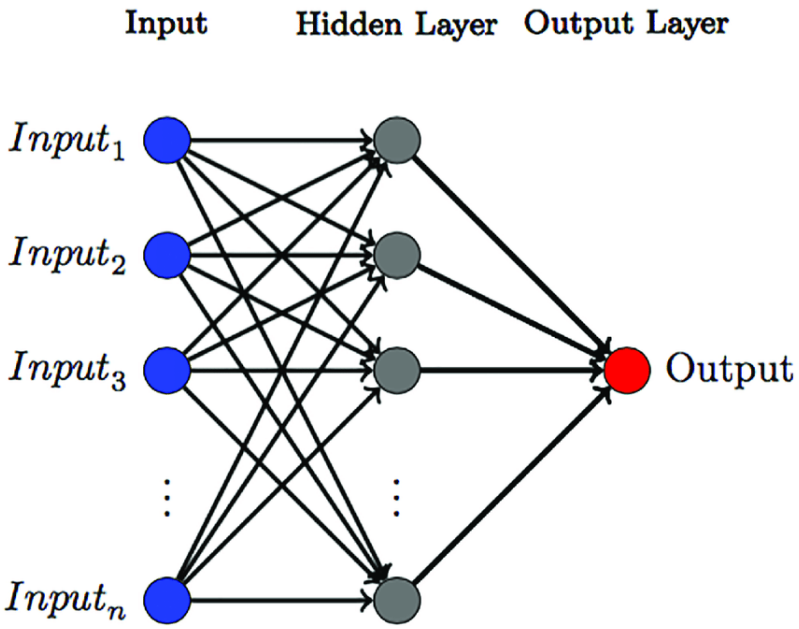


Figure 3.4: Feed Forward Neural Network. Figure Source [BMS18]

3.3.1 Fully Connected Deep Network

A neural network is considered fully connected, when all the nodes from a layer are connected to all the nodes from the next layer. In Figure 3.4, these links between nodes are drawn as arrows.

3.3.2 Feed Forward Neural Network

It is a feed forward neural network, if the arrows are directed toward the output layer. Every link between two neurons from adjacent layers has an associated weight. Throughout this thesis, the weight belonging to the connection between the neuron a and the neuron b, will be denoted by w_{a-b}.

The main purpose of a feed forward neural network is to approximate a function f. It defines a mapping f($\mathbf{x}$,θ)=$\mathbf{y}$ and tries to find the best value for θ, such that the best approximation for the function f is obtained.

The information flows through the function f, which is evaluated by $\mathbf{x}$, through the intermediate computations used to define f, reaching in the end the output $\mathbf{y}$. There aren't any connections in which the outputs are fed back into itself.

3.3.3 The Neurons

We can observe in the Figure 3.4, that each layer can have a different number of neurons. But, the number of neurons for the input and the output layer need to fulfill some rules. Thus, the number of neurons in the input layer is equal with the

number of features extracted as input data. In our case, there will be 41 neurons in the input layer. The number of neurons in the output layer is equal with the number of output values. For our case, will be equal with two. The number of hidden layers and the number of neurons in these layers represent the hyperparameters.

The nodes from the input layer have as value the corresponding feature. The value of a node n_j from the first hidden layer is computed using the formula:

$$n_j = \phi(bias + \sum_{i=1}^{41} x_i w_{i-j}) \tag{3.9}$$

where bias represents the trainable bias weight (which is unique for each neuron) and $\phi(x)$ represents the activation function.

The bias works as an intercept in the case of logistic regression. The purpose of the activation function is to introduce non linearity. The most used activation functions are: the rectifier, also called ReLu, the hyperbolic tangent and the sigmoid [KO11].

We can observe that a neural network without a hidden layer is equivalent with the logistic regression. Only when we add a hidden layer to the model, it will be able of nonlinear classification. The universal approximation theorem states that only one hidden layer is necessary to approximate any continuous function.

3.3.4 Activation Functions

The role of the activation function is to help the neural network model to learn complex patterns from the dataset. If it was to make a comparison with the human brain, the activation function is placed at the end of a neuron, deciding what should be fired to the next neuron. The same goes for the artificial neural network: the function takes the output signal from a neuron, converts it and pass it to the next neuron. The comparison can be seen in the Figure 3.5.

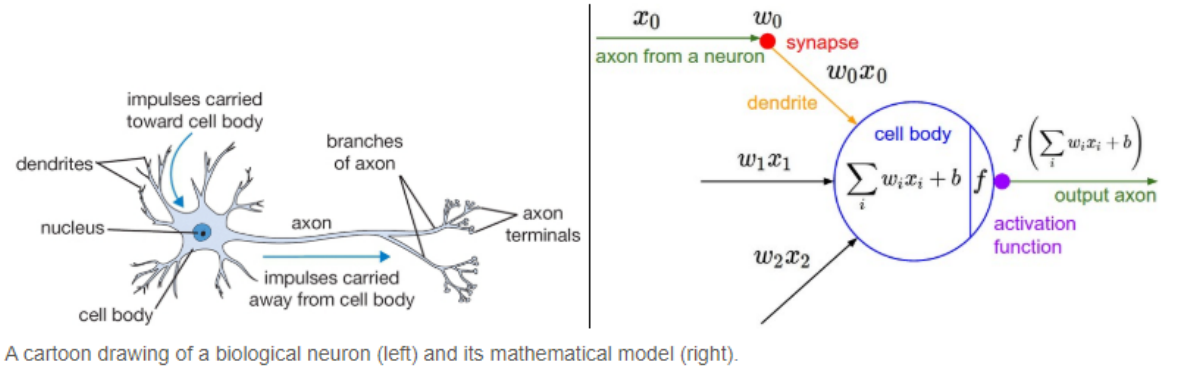
Another task done by the function is to keep the output of a neuron in a required limit. If the value is not restricted, can become very big, especially if there is a neural network that has millions of parameters. So, it will lead to issues regarding the computations.

In order to find the most suitable activation function, we need to take into consideration some characteristics.

The Cost Of The Computations

The activation functions should be inexpensive to calculate, because in a neural network they are applied after each layer, so they are computed a couple of thousand times.

Vanishing Gradient Problem



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Figure 3.5: Biological Neuron VS Model Neuron. Figure Source [Van]

The neural networks are trained using the gradient descent method. The method consists in changing the neuron's weights to reduce the loss after each epoch. The gradients tend to vanish due to the network's depth and the activation shifting the value to 0. Thus, this is the vanishing gradient problem.

Zero Centered

The output of the activation function must be symmetric in 0, in order for the gradients to not shift in a single direction.

Examples Of Activation Functions

1. Sigmoid Function

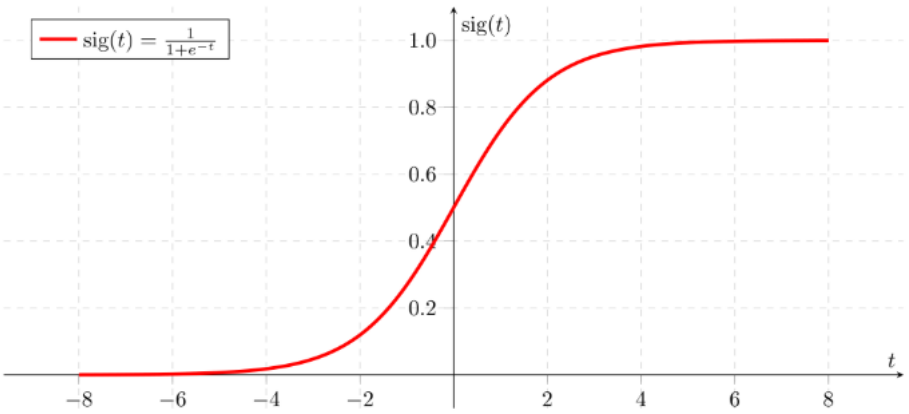


Figure 3.6: Sigmoid Function. Figure Source [Van]

This function (Figure 3.6) is rarely used. It is computationally expensive, not centered in zero and it produces the vanishing gradient problem. Usually, it is used in binary classification problems.

2. Softmax Function

It is a generalized sigmoid function. Because, it produces values in [0,1], it is used in the final layer of a model. This function can be found in multi class classification problems.

3. Tanh Function

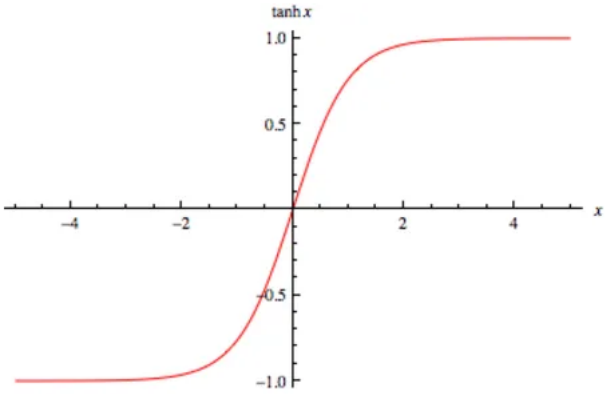


Figure 3.7: Tanh Function. Figure Source [Van]

It is better than the sigmoid function, by solving the problem of being centered in zero. Its representation can be seen in the Figure 3.7.

4. Rectified Liniar Unit Function (ReLu)

Figure 3.8: ReLu Function. Figure Source [Van]

It is a function (Figure 3.8) widely used. The advantages are that the vanishing problem is avoided and is inexpensive to calculate. The only disadvantage is of not being centered in zero. Because, for the negative inputs the outputs are zero, some nodes will die and not learn anything.

3.3.5 The Process

Let's see how the process will work. For each neuron, we will apply the activation function on the value of the weighted sum of the inputs, using the formula 3.9. These values are propagated through the hidden layers until they reach the output layer. Afterwards, in the output layer are computed the predictions. In our case there are two neurons in the output layer that uses the softmax activation function. In the process of training, the weights are optimized using the gradient descent method. The backpropagation method is used to compute the gradients [HN92].

3.3.6 Backpropagation

Backpropagation represents the essence of the neural network training. The method consists in tuning the weights based on the errors obtained after each epoch. By tuning the weights, the model becomes more reliable due to error reducing.

The algorithm calculate the gradient of the loss function, using the chain rule for a single weight. It computes one layer at a time.

Let's see how it actually works. Consider the Figure 3.9. The input X is modeled using the weights W (chosen in a random order). For each neuron, the output is computed, until the last layer is reached. Then, the errors are computed. Travel back to the input layer, through the hidden layers, adjusting the weights. This process is repeated until the desired output is obtained.

The advantages of backpropagation are: flexible method, generally works well, it does not have parameters that need to be tuned (except the number of input).

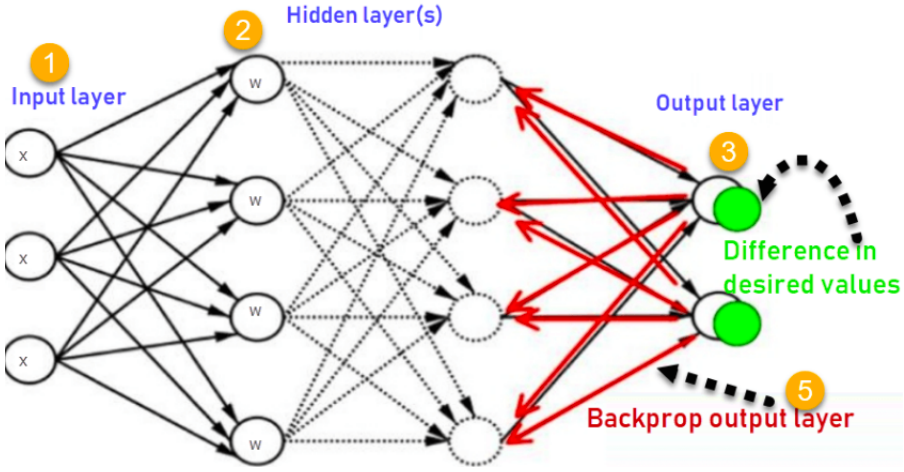


Figure 3.9: Backpropagation. Figure Source [Dan]

There are two types of backpropagation: static and recurrent.

Static Backpropagation

It is used in static classification. Simply maps a static input to a static output.

Recurrent Backpropagation

The propagation is fed forward until a fixed value is obtained. Then, the error is calculated and propagated backward.

3.3.7 Preprocessing

Before the process of training, the input features are standardized. Its advantage is that it leads to better results and faster training.

Each value from the input features is subjected to the process of standardization. Following it, the values will belong to the interval [-1,1].

The formula used in this process is:

$$f = \frac{2}{max - min} * f - \frac{max + min}{max - min} \tag{3.10}$$

where max and min are the maximum and the minimum numbers from the set of values for the feature *f*.

3.3.8 Loss Function

The function used as loss function is mean squared error. It's formula is:

$$error = \frac{1}{2} * (\hat{Y} - Y)^2 \tag{3.11}$$

This is the most common used loss function.

The loss function presents the error obtained from the prediction. Our task is to minimize the error. If the error is low, that means that $\hat{Y}$ is close to Y (the output value to the actual value).

The lower the error is means the higher accuracy for our network is. Once we chose the lost function, the process of backpropagation begins. The resulting data is fed backward through the neural network. The weights are modified as long as there exists differences between $\hat{Y}$ and Y.

The process is repeated until the error reach the smallest value possible. The optimal weights are found when the actual and output values are very close.

Chapter 4

Software Application

This practical chapter presents the way in which the theoretical details described in the previous chapters are implemented and integrated in the prediction system Simo.

4.1 Methodology

The main purpose of this research is to build a prediction system which tries to determine as accurate as possible, the outcome of a tennis match. Beside this, the system will predict which young players have a big probability of becoming a tennis star, which players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament.

The main goals of the application, that were set before implementing it, are: a high computation speed together with technologies that run on web applications and which support different libraries used in deep learning.

To obtain accurate and fast predictions, two models were implemented using Artificial Neural Network and Logistic Regression. The information used for the input data was extracted from a large database which contains data about all the matches from the last 100 years. In contrast with other approaches which use mostly the current ranking and the number of points for the players, it is more time consuming, but the information obtained is worthy. Before using the set of data as input, it will be subject to a process of standardization. Then, it will be passed as training set for the models implemented.

The application Simo is created in the context of artificial intelligence and deep learning, thus it was implemented in Python. Then, it is integrated in an embedded web platform and run as a web application. The application receives the name of two players and the tournament where the match takes place, and using a model will output the chances of both players for winning the match.

The main elements of the approach used that help to obtain relevant predictions and high computation speed are:

1. Extract the features from a large dataset
2. Use of neural networks and logistic regression as models
3. Hundreds of matches were used for the training and testinng process
4. Implementation fully on Python, great for embedded web platofrms

The approach consists of two main parts: implementation of the two models and applications of the neural network model. The first part contains four steps: data extraction, model implementation, training and testing. The second part will run some algorithms found by me that use the neural network model.

4.1.1 Extraction Of Features

In the process of features extraction multiple steps are involved. First, the profile of each player is created. Using the pandas framework, from csv files are read the entire list of players who have at least one match and the list of matches from 2021. The matches are then sorted in increasing order by the date of the match. Using these two lists, are checked which players are still active in 2021 and a basic profile is created for each one, containing: the name, the birth year, the country, the hand used and the height.

Then, each match from 1980 to 2021 is taken separately and data is extracted from it. Each player has a list of percentages, thus they are updated according to the information extracted from a certain match: the percentage of victories (for the winner), the percentage of victories in the last 5 matches (for the winner), the percentage of victories against the opponent (for the winner), the fatigue (depending on the number of sets played at the last two tournaments-both for winner and the loser), the percentage of victories on the surface on which the match took place (for the winner), ace percentage (both for winner and the loser), double fault percentage (both for winner and the loser), winning on 1st serve percentage (both for winner and the loser), winning on 2nd serve percentage (both for winner and the loser), first serve success percentage (both for winner and the loser), breakpoints faced and saved percentages (both for winner and the loser), the percentage of losses (for the loser), the percentage of losses in the last 5 matches (for the loser), the percentage of losses against the opponent (for the loser), the percentage of losses on the surface on which the match took place (for the loser).

Using the module Pickle provided by Python, the profiles are serialized and saved in a file. Also, using the data structure DataFrame provided by pandas, the profiles are saved as a csv file.

The next step consists in grouping data together to form a match. Thus, for each match from 2019 to 2021, the name of the tournament, the surface and the level, plus the players' profiles saved previously using pickle (now deserialized) to which are added the ranking and the number of points owned at that moment, are taken into consideration.

The list of matches is again saved using the module Pickle.

Because the input values used by the neurons, must be numbers, to the features which have a string as values, are assigned some natural numbers. For example, each tournament name, surface and level receives a natural number which is used through the entire application.

The last step which consists in the process of standardization, is different for the two types of models. In the case of neural network, each value will be transformed in order to belong to the interval [-1,1]. Because the percentages are values between 0 and 100, they only need to be multiplied with 0.01 to fulfill the requirement. For the other features, the Formula 3.10 is used.

In the case of logistic regression, we use the class provided by the library sklearn.preprocessing, StandardScaler. It standardize the features by removing the mean and scaling to the unit variance. Then, using its method transform(), the lists of training and testing input are transformed.

4.1.2 Logistic Regression

Before starting the learning process, the data is split in the training and testing set, using the ratio of 70-30 percentages. It is done using the method train_test_split, provided by the data science library scikit-learn. It returns four list, two corresponding to the training part, and the other two represents the testing set. For each of the two paired lists, one of them corresponds to the the input values, and one of them to the output values.

The model is created using the class LogisticRegression, which is found in the library sklearn.linear_model. Also, it provides two methods for the training and testing process: fit() and predict().

Validation Strategy

The model will be trained and tested on different years. Thus, the hyperparameters are tuned only once and are kept the same for each year. In order to find the most suitable values for the hyperparameters, the model is trained on matches from 2012

and tested using matches from 2013. The model is trained using the lbfgs solver and implemented using scikit-learn.

The result of this hyperparameters search can be observed in the Table 4.1 .

Table 4.1: The tuned hyperparamteres

Solver	Max Iterations
lbfgs	100

Solver

The solver is an algorithm used for the optimization problem. LBFGS represents an optimization algorithm which approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm using limited memory. It is very popular in the domain of machine learning, used to estimate parameters. How does it work? If we have a scalar function $f(x)$, it will be minimized over unconstrained values of the real vector x.

Max Iterations

Represents the maximum number of iterations used by the solver to converge.

4.1.3 Artificial Neural Network

As in the case of logistic regression, before starting the learning process, the data is split in the training and testing set, using the ratio of 70-30 percentages. It is done using the method train_test_split, provided by the data science library scikit-learn. It returns four list, two corresponding to the training part, and the other two represents the testing set. For each of the two paired lists, one of them corresponds to the the input values, and one of them to the output values.

The model is created using the class Sequential, which is found in the library keras.models. It contains the method add(), which puts a new layer to the model. The layers are created using the Dense class from keras.layers. In our case, we will have the input layer (with 41 neurons and the ReLu activation function), two hidden layers (with 100 neurons and 20 neurons plus the ReLu activation function) and one output layer (with 2 neurons and the softmax activation function). Also, the library keras provides two methods for the training and testing process: fit() and predict().

Validation Strategy

The complexity of our network model is based on the number of hyperparameters that need to be tuned. It is very hard to tune each hyperparameter individually, so we need to find optimal solutions regarding the architecture. In the most cases,

an architecture which has two hidden layers and which uses ReLu as an activation function, leads to good results.

As it was mentioned at the beginning of this chapter, the model will be trained and tested on different years. Thus, the hyperparameters are tuned only once and are kept the same for each year. In order to find the most suitable values for the hyperparameters, the model is trained on matches from 2012 and tested using matches from 2013. The neural network is trained using the Adam optimizer and implemented using Keras.

The result of this hyperparameters search can be observed in the Table 4.2 .

Table 4.2: The tuned hyperparamteres

Hidden Neurons	Dropout	Epochs	All Matches
100,20	0.5	50	Yes

Hidden Neurons

It refers to the number of neurons in the hidden layer. If a big number of neurons are used, it will let the model to work with complex functions, but it would be vulnerable to overfitting. Thus, I chose a smaller number of hidden neurons.

Dropout

It is an option to regularize a neural network. For each input in the training set, the output of a fraction of neurons, in the layers where the dropout is applied, it is set to 0. This will ensure that each neuron acts individually and it does not depend on the other neurons. Dropout is applied to the hidden layers.

Epochs

It represents the number of times the training set is traversed in the process of training. If a big number of epochs are used, the model is vulnerable to overfit, otherwise if a small number of epochs are used, the model will underfit.

All matches

This hyperparameter means if matches from all levels were taken into consideration. It has been observed, that the model works better, if all the matches are used. Probably comes from the fact that the neural network can learn from low level matches.

4.1.4 Applications Of The Neural Network Model

In what comes next, the model implemented in the last section will be used to predict which young players have a big probability of becoming a tennis star, which

players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament.

Rising Stars

In order to predict rising stars, I implemented an algorithm created by myself which uses the artificial neural network model implemented.

First let's start with the tournaments that take place during a year in the Women Tennis Association tour. There are 5 types of levels, depending on the tournament's prestige, plus the Olympic Games and The Billie Jean King Cup (The Cup of Federations), which are not taken into account due tot the fact that they don't bring any points to the players. Let's see the types of levels. The most important one is the Grand Slam. There are only 4 tournaments of this type and the maximum number of points that can be won is 2000. The next is the Premier Mandatory, which can bring maximum 1000 points to a player. Only 7 tournaments of this type exist. The next category is Premier: 14 tournaments during a year, with 500 points that can be win. The lower levels are: Women Tennis Association Series 250 (26 tournaments) and Women Tennis Association Series 125 (4 tournaments).

Considering the official calendar for a year 4.3 (where WTA comes from Women Tennis Association), I tried to create a possible virtual route for an average player. Of course, a player tries to go to the most prestigious tournaments, because they offer more points for a match won. So the route is composed by: 4 Grand Slams - 7 Premier Mandatory - 10 Premier - 8 Women Tennis Association Series 250 - 1 Women Tennis Association Series 125. The way I imagine it is: Abu Dhabi Open - Yarra Valley Classic (the start of the season) - Australian Open (a short break after) - Adelaide International - Qatar Open - Dubai Tennis Championship (very close locations Doha and Dubai) - Miami Open - Charleston Open - MUSC Health Open (two consecutive tournaments in the same location) - Stuttgart Open - Madrid Open - Italian Open (the known triple for the starting of the clay season) - Internationaux de Strasbourg (training before the grand slam) - Roland Garros - Birmingham Classic - Eastbourne International (start of the grass season) - Wimbledon - Hungarian Grand Prix - Palermo Open - Silicon Valley Classic (back in USA) - Canadian Open - Cincinnati Open (double premier mandatory, then a short break) - US Open - Luxembourg Open - Ostrava Open - Chicago Fall Tennis Classic (back in USA before the last premier mandatory of the year) - Indian Wells Open - Kremlin Cup - Transylvania Open - Linz Open.

After creating this virtual route, I took the players between the places 51 and 300, with age less than 25 years, who never reached top 50. For each player, I computed

Table 4.3: Calendar Tournaments

Week Of	Tournament	Category
January 4	Abu Dhabi Open	Premier
February 1	Yarra Valley Classic	Premier
February 8	Australian Open	Grand Slam
February 15	Phillip Island Trophy	WTA Series 250
February 22	Adelaide International	Premier
March 1	Qatar Open	Premier Mandatory
March 8	Dubai Tennis Championship	WTA Series 250
March 15	Monterrey Open	WTA Series 250
March 15	St. Petersburg Trophy	Premier
March 22	Miami Open	Premier Mandatory
April 5	Charleston Open	Premier
April 5	Copa Colsanitas	WTA Series 250
April 12	MUSC Health Open	WTA Series 125
April 19	Stuttgart Open	Premier
April 19	Istanbul Cup	WTA Series 250
April 26	Madrid Open	Premier Mandatory
May 10	Italian Open	Premier Mandatory
May 17	Serbia Ladies Open	WTA Series 125
May 17	Emilia-Romagna Open	WTA Series 250
May 24	Internationaux de Strasbourg	WTA Series 250
May 31	Roland Garros	Grand Slam
June 7	Nottingham Open	WTA Series 250
June 14	Birmingham Classic	WTA Series 250
June 21	Eastbourne International	Premier
June 28	Wimbledon	Grand Slam
July 5	Hamburg European Open	WTA Series 250
July 12	Hungarian Grand Prix	WTA Series 250
July 19	Palermo Open	WTA Series 250
August 2	Silicon Valley Classic	Premier
August 2	Winners Open	WTA Series 250
August 9	Canadian Open	Premier Mandatory
August 16	Cincinnati Open	Premier Mandatory
August 23	Tennis in the Land	WTA Series 250
August 23	Chicago Women's Open	WTA Series 125
August 30	US Open	Grand Slam
September 13	Luxembourg Open	WTA Series 250
September 13	Slovenia Open	WTA Series 125
September 20	Ostrava Open	Premier
September 27	Chicago Fall Tennis Classic	WTA Series 250
September 27	Astana Open	WTA Series 250
October 4	Indian Wells Open	Premier Mandatory
October 18	Kremlin Cup	Premier
October 25	Transylvania Open	WTA Series 250
November 8	Linz Open	WTA Series 250

a percentage for each type of level, in the following way:

$$PercentageLevel = \frac{\sum_{i=1}^{k} \sum_{j=1}^{50} percentage(tournament_i, player_j, player)}{n} \quad (4.1)$$

where $tournament_i$ represents a tournament from the list of tournaments of a given level, k is the number of tournaments for a level, $player_j$ is a player from top 50, $player$ represents the player for which we want to compute the percentage, n is the total number of matches taken into consideration and the function $percentage(tournament_i, player_j, player2)$ returns the player2's percentage of winning the match.

The percentage obtained represents the player's probability of winning a tournament from that category.

I created a table for each level (4.4, 4.5, 4.6, 4.7, 4.8), where depending on percentages, will be represented how far the player will go in the tournament and the number of points that are won.

So, for each percentage I will convert to the number of points, and multiply to the number of tournaments that the player is going to play for a category (using the route). Then, the total number of points for each level is added and we obtain the final result. If the result is greater than 1100 (the number of points for the player on the place 50), then that player has a big chance of becoming a tennis star (reaching top 50).

Table 4.4: Grand Slam Level

Percentage	Round	Points
30%	Round 1	10
30%-40%	Round 2	70
40%-50%	Round 3	130
50%-60%	Round 4	240
60%-70%	Quarter-Final	430
70%-80%	Semi-Final	780
80%-90%	Final	1300
90%-100%	Win	2000

TOP 10

The algorithm implemented to predict players who will reach top 10 is very similar with the one described before. Let's see the changes. The players who are taken into account are those from places between 11 and 30, who never reach top 10. Then, the route is a little bit change. I eliminated the Women Tennis Association Series

Table 4.5: Premier Mandatory Level

Percentage	Round	Points
30%	Round 1	10
30%-40%	Round 2	35
40%-50%	Round 3	65
50%-60%	Round 4	120
60%-70%	Quarter-Final	215
70%-80%	Semi-Final	390
80%-90%	Final	650
90%-100%	Win	1000

Table 4.6: Premier Level

Percentage	Round	Points
40%	Round 1	1
40%-50%	Round 2	30
50%-60%	Round 3	55
60%-70%	Quarter-Final	100
70%-80%	Semi-Final	185
80%-90%	Final	305
90%-100%	Win	470

Table 4.7: Women Tennis Association Series 250 Level

Percentage	Round	Points
50%	Round 1	16
50%-60%	Round 2	30
60%-70%	Quarter-Final	60
70%-80%	Semi-Final	110
80%-90%	Final	180
90%-100%	Win	280

Table 4.8: Women Tennis Association Series 150 Level

Percentage	Round	Points
60%	Round 1	40
60%-70%	Quarter-Final	57
70%-80%	Semi-Final	75
80%-90%	Final	95
90%-100%	Win	160

125, because very often, only players lower ranked participate in such tournaments, plus two Premieres, due tot the fact that because they are great players, they will pass many rounds in each tournaments, so the fatigue will be accumulated. Thus, the final route is: 8 Premier - 4 Grand Slam - 7 Premier Mandatory - 8 Women Tennis Association Series 250.

After computing the final points for each of these players, the ones who have at least 3300 points (the number of points owned by the player on the 10th position) are considered to have a big chance to reach top 10.

Grand Slam Winners

The algorithm implemented to predict the players who have chances to win a grand slam, is very similar with the previous ones. For this algorithm I took the first 128 players from the world ranking, because these are the players who participate in a grand slam and tested on those who never won a grand slam. Because only the grand slam are the tournaments of interest for us, only the percentages for this level is computed. The formula becomes:

$$PercentageTournament = \frac{\sum_{j=1}^{128} percentage(tournament, player_j, player)}{n} \quad (4.2)$$

where $tournament$ represents one of the four grand slams, $player_j$ is one of those 128 players taken into consideration, and different from the variable $player$, n is the number of matches and $percentage(tournament, player_j, player2)$ returns the player2's percentage of winning the match.

I took as valid results, those players who has a percentage which suggest that she will reach the final or the players who are very close to reach it ($> 75\%$), according to the Table 4.4.

Women Tennis Association Finals

The algorithm used to predict the players who will go to the final tournament, uses parts from the previous ones. I took the players from top 50 and compute for each of them a percentage using the formula:

$$PercentageLevel = \frac{\sum_{i=1}^{55} \sum_{j=1}^{50} percentage(tournament_i, player_j, player)}{n} \quad (4.3)$$

where $tournament_i$ represents a tournament from the entire list of tournaments, $player_j$ is a player from top 50, $player$ represents the player for which we want to compute the percentage, n is the total number of matches taken into consideration

and the function $percentage(tournament_i, player_j, player2)$ returns the player2's percentage of winning the match.

4.2 Analysis And Design

Considering the motivation presented in the previous section, the application Simo which is capable of predicting multiple things in tennis with the help of a neural network model, was developed.

The creation of the application has started with establishing the database. In the Figure 4.1 there is the representation of the entity-relationship model of the PostgresSQL database used. The table LoginTennisFan is used to create accounts for users and the table Rank keeps the number of correct predictions for each user. The tables Game, RisingStar, GrandSlamWinner, Top10 and Women Tennis Association Finals are used to retain the predictions made by the users. The last table, Archive, contains the neural network's game predictions tested by different users. In the programming language Python, the tool used to extract information is Psycopg2. It is used as an adapter for the database. Another Python tool used is Flask (for server integration).

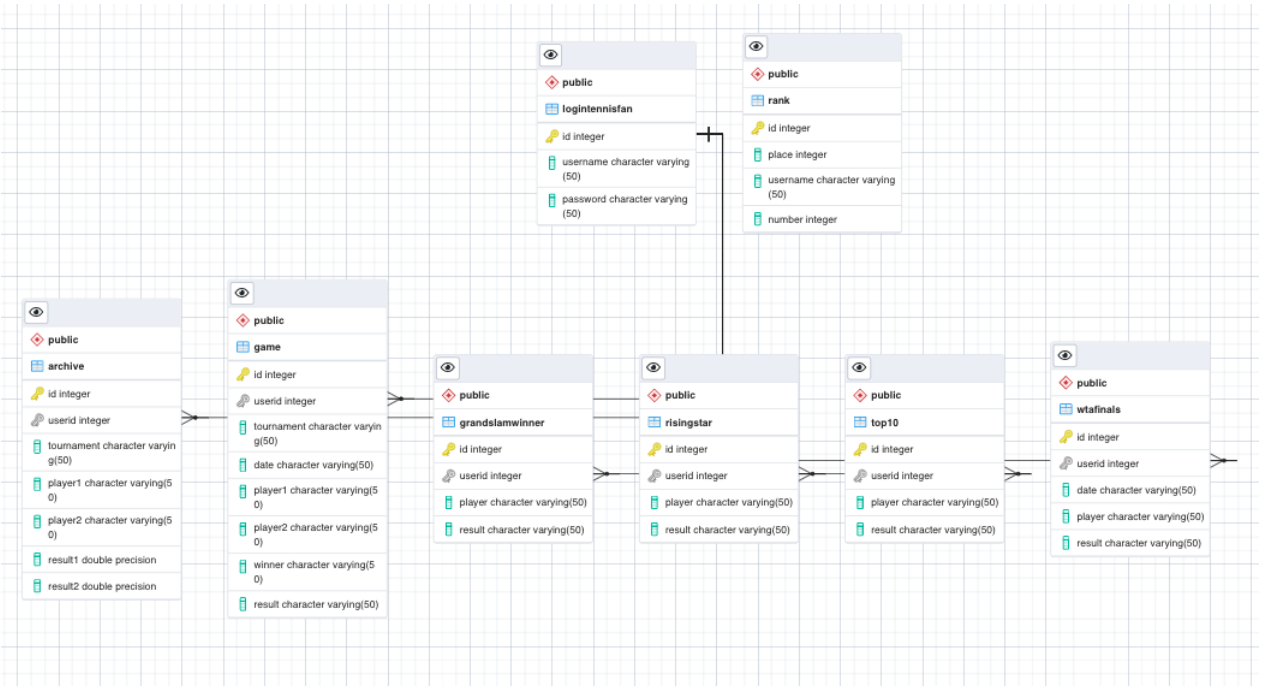


Figure 4.1: Entity-Relationship Diagram

Regarding the roles, in the Simo app there are two: admin and user. A new admin can be created only from the database, in order to avoid for other users to create an account of this type. The admin has only one task: to accept or to reject the predictions made by the users, according to the outcomes. The functionalities

performed by the user are: add new prediction, change the account's password, delete the account, view the predictions made by the computer reagrding matches, rising stars, top 10, grand slam winners, Women Tennis Association finals, view the archive of matches' predictions, see the list of players, the current ranking, the list of tournaments, list of statistics and the ranking between users. In the Figures 4.2 and 4.3, there are case diagrams with the admin's and user's functionalities.

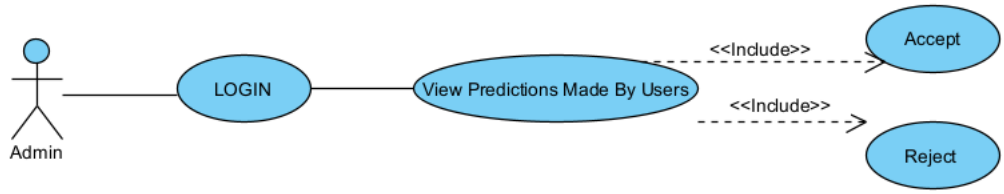


Figure 4.2: Case Diagram For Admin

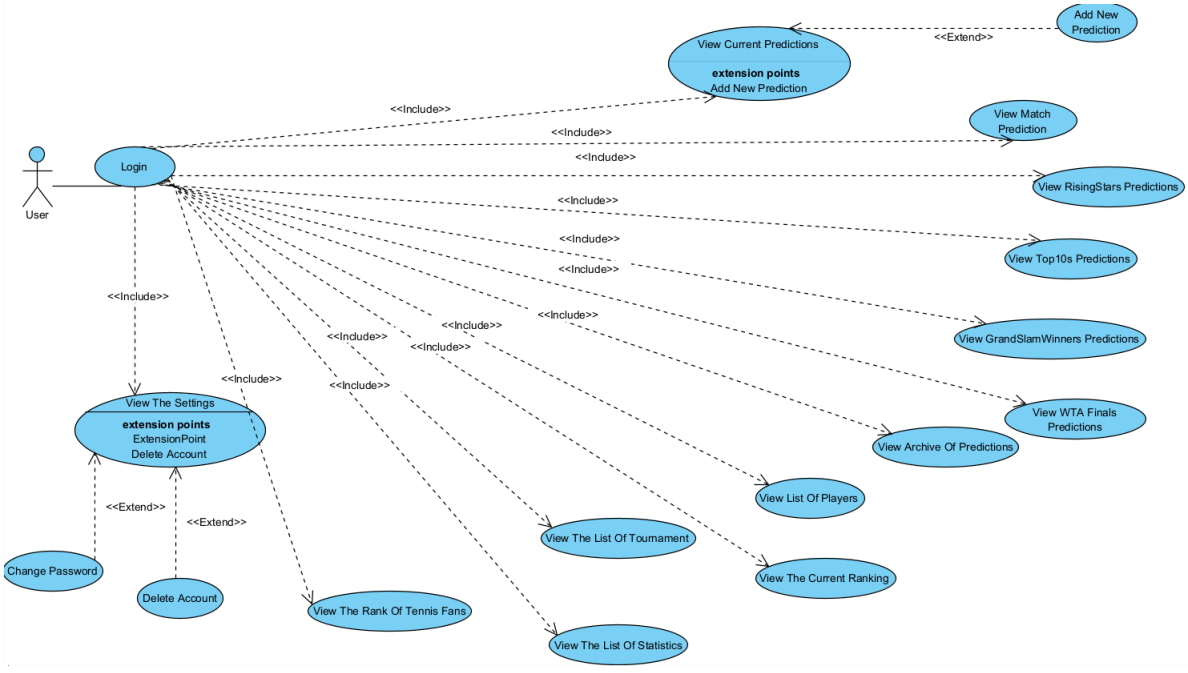


Figure 4.3: Case Diagram For User

In the Figure 4.4, the prediction algorithm is presented. This scenario, takes as input information about players and matches from the dataset. This information is then subject to the process of standardization. After, the steps of training and testing the models (neural network and logistic regression) are performed. In the end, the

models are capable of generating predictions.

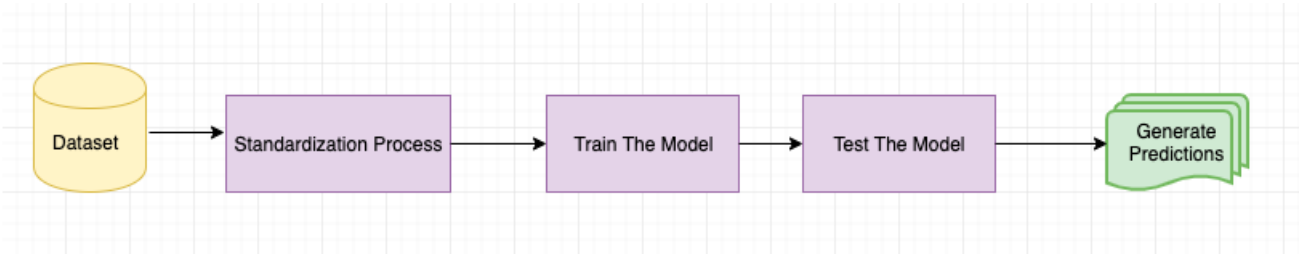


Figure 4.4: Architecture Diagram For Application

4.3 Implementation

In this section, the approach for creating the application, is presented in a more detailed way. The application Simo for tennis recommendations was developed entirely using Python for the backend side, while for the frontend, the Angular framework for web applications was used. The reason why I chose Python as the programming language of the application, is represented by the numerous libraries that supports artificial intelligence and machine learning algorithms. No other external library was used. For the data management, Flask is used as server integration (to connect the backend to the fronent) and Psycopg2 is an adapter used for the PostgresSQL database. For viewing the results, the libraries used are: matplotlib and sklearn. Regarding the machine learning operations, I used: pandas, keras, sklearn, numpy. In the end, a graphical user interface was developed in order to be used as a real application for tennis fans.

4.3.1 Installation and Libraries Needed

In order to install the application Simo, the next steps must be performed. Firstly, let's see the environment in which the application runs. The system configurations are:

1. macOS Big Sur Version 11.6 - Operating system
2. Processor 1,8 GHz Intel Core i5 - Processor

The software used is:

1. PostgreSQL database
2. Python 3.7
3. Angular CLI

The Python libraries used are:

1. Psycopg2 - the most popular PostgreSQL database adapter for the Python programming language
2. Flask - Python web framework
3. Pandas - a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language
4. Keras - an open-source software library that provides a Python interface for artificial neural networks
5. Sklearn - a free software machine learning library for the Python programming language
6. Numpy - a Python library used for working with arrays

4.3.2 Data Processing

For the process of features extraction, five classes were used. The most basic ones are the classes Match and Player, which simply define their attributes, used further by the other classes. The next class is represented by Player creation, where a list of profiles is made using the data from the matches between 1980 and 2021. Then, using these profiles, we create the matches that will be used for training and testing, in the class Match data creation. The last class is represented by Data_Treatment, which will put the data gathered in the previous class, subject to the process of standardization.

4.3.3 Logistic Regression

In the class Main, the logistic regression model is implemented as it was described in the Section 4.1.2. First, the data is transformed using the method of standardization. Then, the data is split in two sets: the training set and the testing set. The model is created using Logistic Regression. In the end, the model is trained and tested, as we can see in the Figure 4.5.

4.3.4 Artificial Neural Network

In the class Prediction, the neural network model is implemented as it was described in the Section 4.1.3. First, the standardize data is split in two sets: the training set and the testing set. Then, the model is created using Sequential, to which we add

```
X, X_test, Y, Y_test = train_test_split(train_x.values, train_y.values, test_size=0.3)
sc = StandardScaler()
X = sc.fit_transform(X)
X_test = sc.transform(X_test)
classifier = LogisticRegression(solver='lbfgs', max_iter=100)
classifier.fit(X, Y.ravel())
y_pred = classifier.predict(X_test)
```

Figure 4.5: Code Snippet:Main

one input layer, two hidden layers and one output layer. In the end, the model is trained and tested, as we can see in the Figure 4.6.

```
X, X_test, Y, Y_test = train_test_split(train_x.values, train_y.values, test_size=0.3)

model = Sequential()
model.add(Dense(100, input_dim=41, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(20, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
history = model.fit(X, Y, validation_data=(X_test, Y_test), epochs=50, verbose=1)
```

Figure 4.6: Code Snippet:Prediction

4.4 Simo Application

In this section, there are presented the functionalities contained in the application, together with their usage.

When the user enter the application, will see the Home Page 4.7 which contains the logo Simo and a button Tennis Fan. The user must click on the button in order to go to the login page.

The Login Page contains two empty input texts, one for the username and one for the password. The user must use its credentials in order to login in the application, or to create a new account if he does not have one. In order to create a new account, there is a SignUp link which redirects to a page used to create a new account.

The SignUp Page contains two empty input texts, one for the username and one for the password. The user must choose a username and a password according with his preferences, to have a valid account which can be used.

The Main Page 4.8 contains an upper menu and a left menu which correspond with different pages in the app. The top of the page contains the last results in tennis matches and the rest of the page is dedicated to the user's predictions. Also, the icon with an arrow is used to logout.

Figure 4.7: Home Page

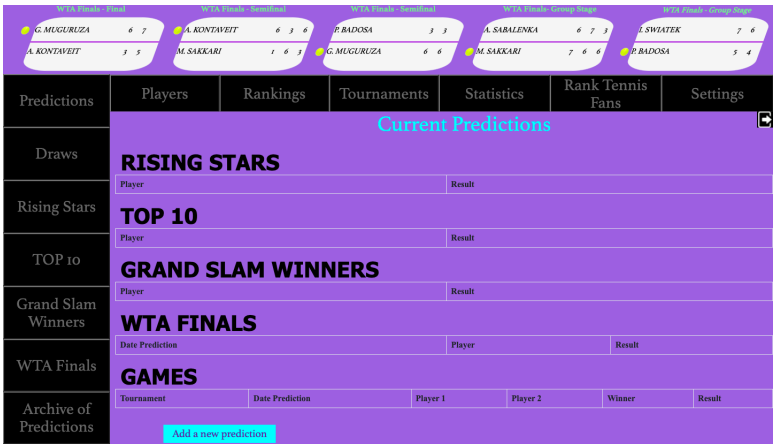


Figure 4.8: Main Page

By clicking the Draws button from the Main Page 4.8, the user is redirected to the Draw Page 4.9, where he can see the predictions made by the computer regarding different matches. All he has to do is to choose a tournament plus two players. A pie chart is generated with the winning percentages for each player.

The other buttons from the left menu (Rising Stars, TOP 10, Grand Slam Winners, WTA Finals) can be used to go to different pages, where a user can see predictions made by computer regarding: which young players have a big probability of becoming a tennis star, which players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament. An example of such a page can be seen in the Figure 4.10.

By clicking the Archive of Predictions button from the Main Page 4.8, the user is redirected to the Archive Page, where he can see the historic of matches tested using the computer.

The Settings button from the upper menu, is used to go to the Settings page, where the user can either change his password or to delete his account. The other buttons from the menu: Players, Rankings, Tournaments, Statistics present different

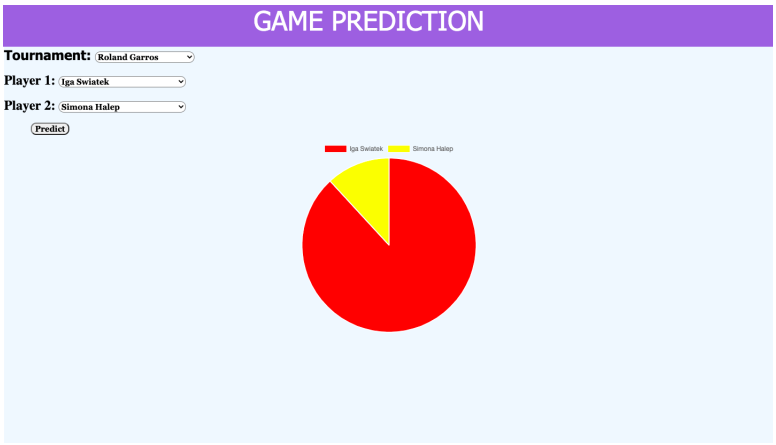


Figure 4.9: Draw Page

Figure 4.10: Rising Stars Page

stuff related to tennis.

The Add Prediction button from the Main Page 4.8, is used to go the Add Prediction Page 4.11. There, the user can make 5 types of predictions regarding: rising stars, top 10, grand slam winners, wta finals and matches. All he has to do is to fulfill the required data. After a prediction is added by a user, it will appear in the corresponding table with a status of pending.

By clicking the Rank Tennis Fans button from the Main Page 4.8, the user is redirected to the Rank Tennis Fans Page 4.12, where a ranking is created with the users regarding the number of correct predictions made by them.

There is a special type of user in the application: the admin, which has a special account. After he logs in, is redirected to the Admin Page 4.13. There is a list with all the predictions made by users and he can accept or reject one of them according with their outcome. Automatically, in the user page the corresponding status is changed from pending to accept/reject.

Figure 4.11: Add Prediction Page

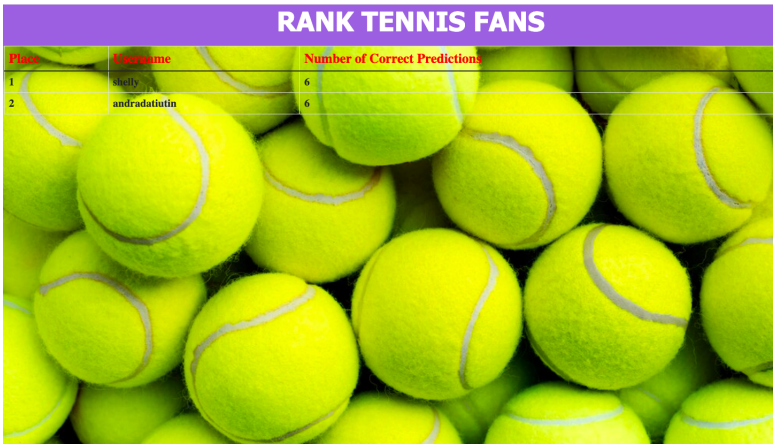


Figure 4.12: Rank Tennis Fans Page

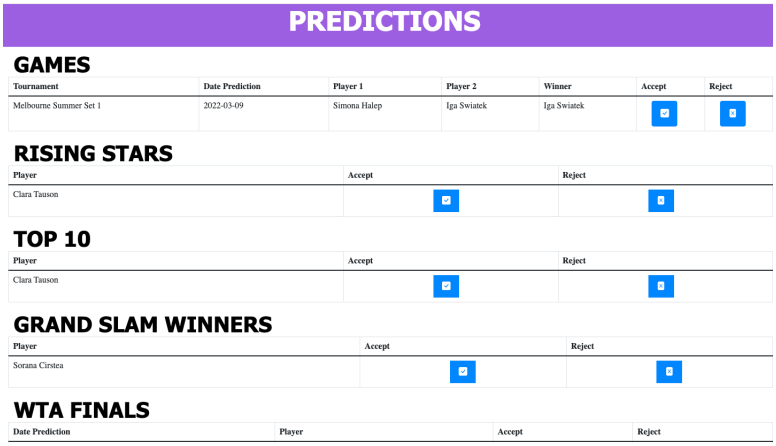


Figure 4.13: Admin Page

Chapter 5

Results Obtained

5.1 Logistic Regression

The model is trained using three different data sets with matches from 2019, 2020 and 2021. Its hyperparameters are the ones from the Table 4.1. The logistic regression is trained using the lbfgs solver and implemented using scikit-learn.

The results obtained are presented in the Table 5.1.

Table 5.1: Results obtained logistic regression

Model	Accuracy	Loss
Logistic Regression	90.29%	0.20

In the Figure 5.1 there is a representation of a confusion matrix. On the first row we have the number of correct matches with the target value 0 identified and the number of matches classified with 0 but being actually a 1. The same goes for the second row. The first number represents the number of matches classified as 1, but being actually a 0 and the number of matches correct classified as 1.

5.2 Artificial Neural Network

The artificial neural network is trained using three different data sets with matches from 2019, 2020 and 2021. Its hyperparameters are the ones from the Table 4.2. The neural network is trained using the Adam optimizer and implemented using Keras.

Let's see the comparison between these two models: neural network and logistic regression in the Table 5.2.

The neural network outweigh the logistic regression both in the loss and accuracy.

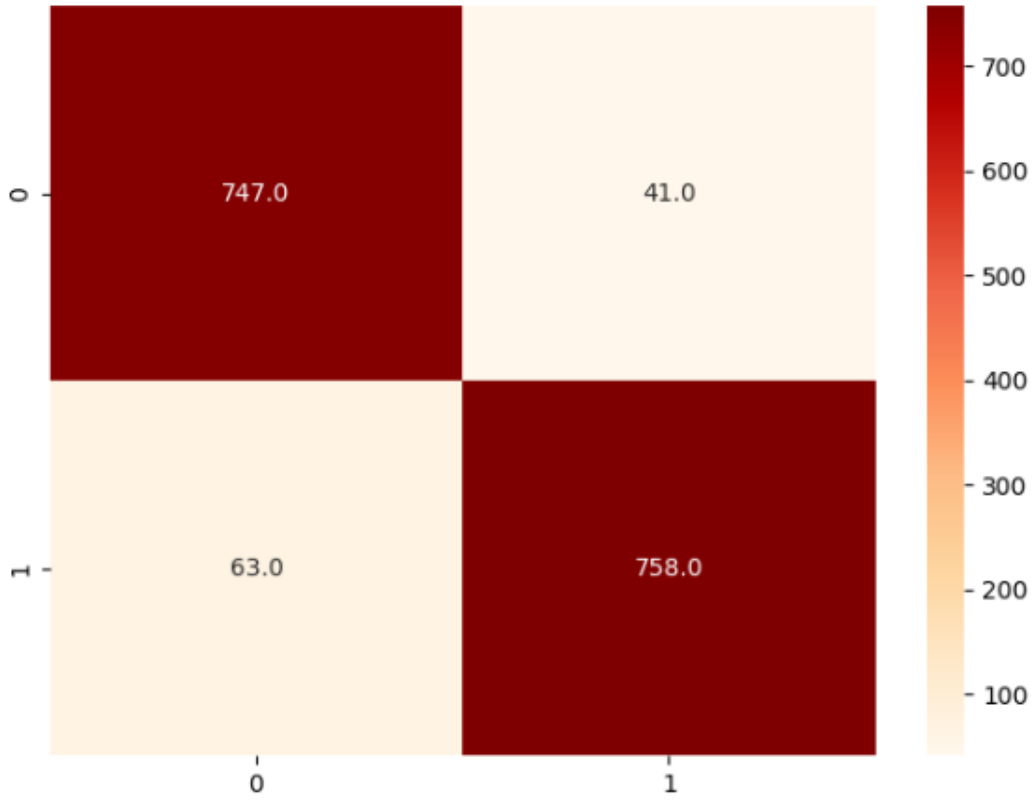


Figure 5.1: Confussion Matrix

Table 5.2: Results obtained neural network

Model	Accuracy	Loss
Neural Network	93.28%	0.02
Logistic Regression	90.29%	0.20

In the Figure 5.2 it can be observed that both the training and the testing process follow the same trend.

5.3 Applications Of The Neural Network Model

In what comes next, let's see the results of the artificial neural network model in predicting which young players have a big probability of becoming a tennis star, which players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament.

5.3.1 Rising Stars

In the Table 5.3 (where WTA comes from Women Tennis Association), we have the results obtained. The players: Anna Kalinskaya, Kaja Juvan, Clara Burel, Claire Liu,

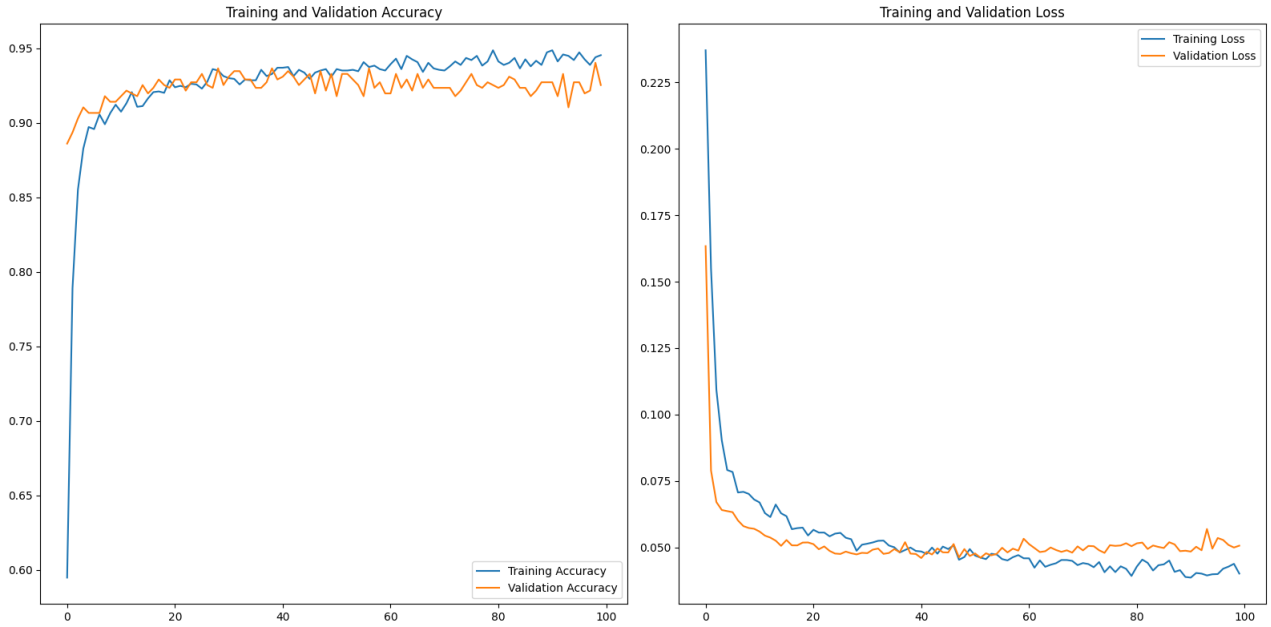


Figure 5.2: Training and Validation Accuracy. Training and Validation Loss

Magdalena Frech, Anastasia Potapova, Maddison Inglis and Caroline Dolehide are qualified for having a big chance to do something great in tennis.

In order to test that the algorithm works, I took the ranking from 2020 and I tested it on the young players, who reached top 50 in 2021. Out of 8 players who made it to top 50, the algorithm found 4: Paula Badosa (current ranking 2), Jessica Pegula (current ranking 13), Anhelina Kalinina (current ranking 38) and Ajla Tomljanovic (current ranking 40).

5.3.2 TOP 10

In the Table 5.4 (where WTA comes from Women Tennis Association), there are the results obtained. The players who have chances to reach top 10 are: Anastasia Pavlyuchenkova, Cori Gauff, Elena Rybakina. Also, from the previous algorithm, the players: Anhelina Kalinina, Claire Liu and Clara Burel have a number of points greater than 3300. So, they are also considered with chances to reach top 10.

In order to test that the algorithm works, I took the ranking from 2020 and I tested it on the players, who reached top 10 in 2021. Out of 3 players who made it to top 10, the algorithm found one: Iga Swiatek (the world number one).

5.3.3 Grand Slam Winners

Because there are 128 players, I put in the Table 5.5 only the players who obtain a satisfactory result for at least one grand slam.

Table 5.3: Rising Stars Predictions

Player	Premier	Grand Slam	WTA Series 250	Premier Mandatory	WTA Series 125	Result
Anna Kalinskaya	62.39%-100P	43.79%-130P	59.14%-30P	53.45%-120P	73.18%-75P	2765P
Kaja Juvan	65.15%-100P	46.78%-130P	62.01%-60P	52.72%-120P	75.36%-75P	2915P
Panna Udvardy	38.95%-1P	18.54%-10P	38.38%-16P	28.68%-10P	60.43%-40P	288P
Clara Burel	63.26%-100P	47.82%-30P	64.28%-60P	54.99%-120P	75.87%-75P	2915P
Claire Liu	75.04%-185P	61.77%-430P	74.97%-110P	66.26%-215P	85.51%-95P	6050P
Magdalena Frech	54.31%-55P	32.06%-70P	52.55%-30P	42.72%-65P	70.79%-75P	1550P
Diane Parry	28.94%-1P	8.62%-10P	24.50%-16P	16.36%-10P	46.98%-40P	288P
Jule Niemeier	40.47%-30P	24.14%-10P	43.68%-16P	33.35%-35P	64.21%-57P	770P
Anastasia Potapova	59.08%-55P	44.35%-130P	56.89%-30P	50.57%-120P	67.66%-57P	2207P
Maddison Inglis	79.86%-185P	61.03%-430P	78.44%-110P	72.51%-390P	91.06%-160P	7340P
Rebeka Masarova	35.70%-1P	11.87%-10P	30.66%-16P	23.07%-10P	54.25%-40P	288P
Alycia Parks	32.47%-1P	14.08%-10P	31.99%-16P	23.20%-10P	55.75%-40P	288P
Anna Blinkova	29.47%-1P	16.57%-10P	26.16%-16P	19.93%-10P	43.52%-40P	288P
Caroline Dolehide	60.60%-55P	32.65%-70P	54.20%-30P	43.74%-65P	72.95%-75P	2050P
Usue Maitane Arconada	11.05%-1P	4.08%-10P	8.54%-16P	6.53%-10P	19.78%-40P	288P
Lizette Cabrera	44.08%-30P	19.08%-10P	35.61%-16P	25.90%-10P	56.63%-40P	578P
Katarina Zavatska	48.81%-30P	22.94%-10P	46.26%-16P	36.73%-35P	67.54%-57P	690P
Francesca Di Lorenzo	20.75%-1P	9.98%-10P	18.83%-16P	14.94%-10P	37.11%-40P	288P
Aliona Bolsova	23.90%-1P	8.81%-10P	21.18%-16P	15.24%-10P	43.82%-40P	288P
Whitney Osuigwe	13.43%-1P	2.54%-10P	9.33%-16P	6.34%-10P	23.64%-40P	288P

Table 5.4: TOP 10 Predictions

Player	Premier	Grand Slam	WTA Series 250	Premier Mandatory	Result
Emma Raducanu	41.65%-30P	25.55%-10P	35.35%-16P	28.84%-120P	478P
Jessica Pegula	51.47%-55P	39.17%-70P	51.22%-30P	44.20%-65P	1415P
Anastasia Pavlyuchenkova	67.26%-100P	58.05%-240P	67.70%-60P	61.49%-215P	3745P
Cori Gauff	68.77%-100P	61.26%-430P	69.21%-60P	63.30%-215P	4465P
Elena Rybakina	64.75%-100P	55.80%-240P	64.42%-60P	57.86%-120P	3080P
Leylah Fernandez	45.69%-30P	40.28%-130P	45.34%-16P	42.17%-65P	1343P
Tamara Zidansek	43.35%-30P	32.03%-70P	45.98%-16P	37.82%-35P	893P
Elise Mertens	49.70%-30P	38.36%-70P	49.51%-16P	42.79%-65P	1103P
Veronika Kudermetova	57.82%-55P	47.92%-130P	58.05%-16P	52.23%-120P	2040P
Sorana Cirstea	50.79%-55P	37.22%-70P	51.25%-30P	42.89%-65P	1415P

Table 5.5: Grand Slam Winners Predictions

Player	Australian Open	Roland Garros	Wimbledon	US Open
Ons Jabeur	69.82%	75.69%	74.07%	69.54%
Belinda Bencic	79.00%	84.51%	77.41%	78.83%
Amanda Anisimova	74.69%	81.68%	76.92%	74.24%
Anhelina Kalinina	70.76%	81.25%	58.11%	71.31%
Ekaterina Alexandrova	72.72%	79.71%	76.86%	72.23%
Magda Linette	79.21%	85.82%	83.29%	79.84%
Petra Martic	85.02%	87.84%	84.69%	84.79%
Dayana Yastremska	68.20%	75.47%	53.15%	68.24%
Maddison Inglis	71.71%	76.41%	55.67%	71.81%
Laura Pigossi	72.91%	83.99%	60.93%	73.00%

In order to test that the algorithm works, I took the ranking from 2017 and I tested it on the players, who won a grand slam until 2022. Out of 8 players who won a grand slam, the algorithm found three: Iga Swiatek (who won Roland Garros in 2020), Simona Halep (who on Roland Garros in 2018 and Wimbledon in 2019) and Naomi Osaka (who on US Open in 2018 and 2020 and Australian Open in 2019 and 2021).

5.3.4 Women Tennis Association Finals

In the Table 5.6, there are the percentages obtained. The first 8 players with the biggest percentages are considered favourites to go the final tournament. They are: Ons Jabeur, Belinda Bencic, Iga Swiatek, Amanda Anisimova, Simona Halep, Naomi Osaka, Anhelina Kalinina, Paula Badosa.

In order to test that the algorithm works, I took the ranking from 2020 and I tested it on the players from top 50. Out of the 8 players who participate to Women Tennis Association Finals 2021, the algorithm found three: Ashleight Barty, Iga Swiatek and Aryna Sabalenka.

5.4 Special Results

After the model was finished, I tried to tested it on actual matches, to see how it performs in real life. Some notable results predicted correct, can be seen in the Table 5.7.

5.5 Comparison With Other Approaches

In the Table 5.8, there are presented other results obtained in the domain of tennis predictions. The number of features used by their models is significantly small comparing with the number of features used by the models presented in this thesis. Thus, I consider that given the large datasets used, which contains information about a couple of hundreds player, is very unlikely that the models will learn very good the players' profiles. Therefore, a lower accuracy is obtained, especially by the last three approaches.

Table 5.6: Women Tennis Association Finals Predictions

Player	Percentage
Iga Switatek	71.74%
Barbora Krejcikova	48.19%
Paula Badosa	76.90%
Maria Sakkari	35.93%
Anett Kontaveit	37.07%
Aryna Sabalenka	45.27%
Karolina Pliskova	45.96%
Danielle Collins	68.14%
Ons Jabeur	75.33%
Jelena Ostapenko	36.38%
Belinda Bencic	83.77%
Emma Raducanu	35.30%
Jessica Pegula	49.77%
Anastasia Pavlyuchenokva	66.22%
Victoria Azarenka	45.60%
Cori Gauff	67.86%
Angelique Kerber	40.19%
Elena Rybakina	65.21%
Simona Halep	70.29%
Madison Keys	54.77%
Leylah Fernandez	44.55%
Tamara Zidansek	43.73%
Elina Svitolina	43.54%
Petra Kvitova	40.48%
Sorana Cirstea	49.79%
Elise Mertens	58.68%
Veronika Kudermetova	56.66%
Camila Giorgi	51.60%
Marketa Vondrousova	58.40%
Amanda Anisimova	78.66%
Alize Cornet	65.38%
Naomi Osaka	76.71%
Anhelina Kalinina	78.84%
Ajla Tomljanovic	60.05%
Viktorija Golubic	54.96%
Shuai Zhang	59.45%
Clara Tauson	46.53%
Alison Riske	53.83%
Sloane Stephens	58.30%
Ekaterina Alexandrova	58.68%
Jasmine Paolini	58.30%
Sara Sorribes Tormo	48.30%
Shelby Rogers	46.86%
Yulia Putintseva	61.17%

Table 5.7: Results Obtained

Tournament	Round	Player1	Player2	Winner
Rome	Final	Iga Swiatek	Ons Jabeur	Iga Swiatek
Rome	Semi-Final	Iga Swiatek	Aryna Sabalenka	Iga Swiatek
Doha	Quarter-Final	Anett Kontaveit	Garbine Muguruza	Anett Kontaveit
Stuttgart	Semi-Final	Aryna Sabalenka	Paula Badosa	Aryna Sabalenka
Madrid	Final	Jessica Pegula	Ons Jabeur	Ons Jabeur
Istanbul	Final	Anastastia Potapova	Veronika Kudermetova	Anastastia Potapova
Stuttgart	Final	Iga Swiatek	Aryna Sabalenka	Iga Swiatek
Indian Wells	Final	Iga Swiatek	Maria Sakkari	Iga Swiatek
Indian Wells	Semi-Final	Iga Swiatek	Simona Halep	Iga Swiatek
Stuttgart	Round 1	Petra Kvitova	Karolina Pliskova	Karolina Pliskova
Australian Open	Round 1	Petra Kvitova	Karolina Pliskova	Karolina Pliskova
St. Petersburg	Round 2	Irina Begu	Petra Kvitova	Irina Begu
Rome	Round 3	Amanda Anisimova	Danielle Collins	Amanda Anisimova
Doha	Round 3	Paula Badosa	Cori Gauff	Cori Gauff

Table 5.8: Other results

Researchers	Number Of Features	Dataset	Algorithm	Accuracy
My model	41	Set of all the matches from 2019 to 2021	Logistic Regression	90.29 %
My model	41	Set of all the matches from 2019 to 2021	Artificial Neural Networks	93.28 %
Bryan L. Boulier, H. OSteklerr	4	The grand slam matches from 1999	Probit Model	86 %
Stephen R. Clarke, David Dyte	4	3003 matches from 1997	Logistic Regression	85 %
Michal Sipko	4	Set of all the matches from 1990 to 2015	Neural Network	61 %
Ian McHale, Alex Morton	2	Set of all the matches from 2000 to 2008	Bradley-Terry model	65 %
Franc J.G.M. Klaassen, Jan R. Magnus	8	Set of all the matches from Wimbledon (1992-1995 $\rightarrow$ 504 matches)	Point based model	60 %

Chapter 6

Conclusion

To sum it up, in this thesis it was developed the application Simo, used by tennis fans to see tennis predictions. It was fully implemented using Python, in order to be incorporated in an embedded web platform and run as a web application. Using an open-source dataset containing historical data from all levels of professional tennis, a total of 41 features are extracted based on previous research and data analysis. The implemented neural network and logistic regression models greatly outperforms the models existed before, over loss and accuracy: 93.28% accuracy and 0.02 loss (neural network) and 90.29% accuracy and 0.20 loss (logistic regression). Also, the users can make their own predictions. It is very intuitive, having a friendly user interface, which contains features like: see predictions made by the computer, view different pages related to tennis, add new predictions, see the ranking of users.

The proposed models are based on an original viewpoint. The novelty consists in using a large number of features, in contrast with other models where a significantly small number of features were used. This is the main reason for which a high accuracy was obtained. Likewise, they are the first models which predicts women's tennis matches, all the other attempts are based on men's matches. Besides these, I implemented four algorithms which use the artificial neural network model, in order to predict which young players have a big probability of becoming a tennis star, which players have the ability of winning a grand slam, which players can reach top 10 or which players can make it to the final tournament.

The domain of tennis predictions has seen an important development in the last decade. Thus, our application can be used in the domain of sports betting, so it comes in the help of those passionate about this stuff.

Regarding the improvement of the application, I would create another part for tennis players. That part will keep track of a player' statistics. Then, based on them and using artificial intelligence, the computer would make different suggestions like: trainings for self-improvement, trainings before a game (based on the opponent's characteristics), tactics before a match (similar with a coach).

Bibliography

[Ayl] Aylin Tokuç. Gradient Descent Equation in Logistic Regression. https://www.baeldung.com/cs/gradient-descent-logistic-regression. Online; Accessed On 10 May 2022.

[Bar] Barry Steven Lorge. Tennis. https://www.britannica.com/sports/tennis/Outstanding-players. Online; Accessed On 13 April 2022.

[BMS18] Björn RH Blomqvist, Richard P Mann, and David JT Sumpter. Using bayesian dynamical systems, model averaging and neural networks to determine interactions between socio-economic indicators. *Plos one*, 13(5):e0196355, 2018.

[BS99] Bryan L Boulier and Herman O Stekler. Are sports seedings good predictors?: an evaluation. *International Journal of Forecasting*, 15(1):83–91, 1999.

[BT52] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[Dan] Daniel Johnson. Back Propagation Neural Network: What is Backpropagation Algorithm in Machine Learning? https://www.guru99.com/backpropogation-neural-network.html. Online; Accessed On 9 May 2022.

[DCPR10] Julio Del Corral and Juan Prieto-Rodríguez. Are differences in ranks good predictors for grand slam tennis matches? *International Journal of Forecasting*, 26(3):551–563, 2010.

[DS20] Alexander De Seranno. Predicting tennis matches using machine learning. 2020.

[Foy] Foy Moory. The Fascinating History Of Tennis. https://www.tennisexpress.com/blog/the-fascinating-history-of-tennis/. Online; Accessed On 13 April 2022.

[Fre] Fred Simonsson. How Tennis Has Evolved Over The Years. https://tennispredict.com/how-tennis-has-evolved. Online; Accessed On 17 March 2022.

[HN92] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[Hum14] Tom Humphrey. Tennis trading on betfair. 2014.

[Jef] Jeff Sackmann. Tennis WTA. https://github.com/JeffSackmann/tennis_wta. Online; Accessed On 28 April 2022.

[KM01] Franc JGM Klaassen and Jan R Magnus. Are points in tennis independent and identically distributed? evidence from a dynamic binary panel data model. *Journal of the American Statistical Association*, 96(454):500–509, 2001.

[KM03] Franc JGM Klaassen and Jan R Magnus. Forecasting the winner of a tennis match. *European Journal of Operational Research*, 148(2):257–267, 2003.

[KO11] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[MM11] Ian McHale and Alex Morton. A bradley-terry type model for forecasting tennis match results. *International Journal of Forecasting*, 27(2):619–630, 2011.

[Pet17] Joss Peters. Predicting the outcomes of professional tennis matches. 2017.

[SK15] Michal Sipko and William Knottenbelt. Machine learning for the prediction of professional tennis matches. *MEng computing-final year project, Imperial College London*, 2015.

[Ste] Steph Yin. Do Lefties Have an Advantage in Sports? https://www.nytimes.com/2017/11/21/science/lefties-sports-advantage.html. Online; Accessed On 21 April 2022.

[Stu] Stuart Miller. How Short Tennis Players Compete in a Sport of Giants. https://www.nytimes.com/2018/09/02/sports/tennis/us-open-height.html. Online; Accessed On 21 April 2022.

[Sur] Surya Remanan. Logistic Regression: A Simplified Approach Using Python. https://towardsdatascience.com/logistic-regression-a-simplified-approach-using-python-c4bc81a8
Online; Accessed On 30 April 2022.

[Van] Vandit Jain. Everything you need to know about "Activation Functions" in Deep learning models. https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-
Online; Accessed On 9 May 2022.

[Vom17] Karan Vombatkere. Predicting professional tennis player success using binary classification methods. 2017.