Machine Learning for Tennis Match Prediction

Angad Beer Singh Dhillon[1]*, Pulkit Mathur[1]*, Uteerna Koul[1]*

Abstract
Tennis is one of the oldest sports in the world. The tennis games garner a huge audience and hence is an interesting topic for machine learning. This project adopts an innovative model which makes uses of the historical data up to the current year and apply various machine learning models to predict tennis match outcomes. Our project has three main objectives. First, we want to predict the winner of future tennis tournaments based on the previously available tennis match data. We used Logistic Regression, Naive Bayes, Random Forest, and Neural Networks to predict the future winner and Neural Networks gives us the best prediction. Next, we are also predicting the ranking of the players after each match and determining if they will increase, decrease or remain the same. Using our models we also are determining the form of each player after the tournament

Keywords
Logistic Regression — Naive Bayes — Random Forest — Neural Network — Activation functions — Hyper-parameters

[1] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
***Corresponding author**: adhillon@iu.edu, pulmath@iu.edu, ukoul@iu.edu

Contents

1. Introduction

1.1 Motivation

Sports is one of the most interesting and talked about topic all over the world. The Association of Tennis Professionals (ATP) features over 60 professional tennis tournaments in over 30 countries every year, drawing a huge number of spectators. Andy Murray's historic defeat of Novak Djokovic in the 2013 Wimbledon final was the most watched television broadcast of the year. Professional tennis is an interesting topic especially for machine learning as the players come from different countries and with different training history. Every player has a different style and specialty. Since tennis is played on three different surfaces, it needs three very different playing strategies for each type of surface. In 2017, 18-time Grand Slam champion Roger Federer lost to World No. 302 player Tommy Haas on a grass court. This kind of thing had never happened since 2002. Since there are a lot of variables that define a tennis match, this motivated us to apply machine learning to the tennis data and make predictions regarding the tennis matches. One player can be very good at clay surfaces just because the locality that they grew up in had more clay courts. Another deciding factor can be a player's hand of playing or a player's height etc. This different background of training and playing technique made us interested in this problem and motivated us to make predictions in this game of tennis.

1.2 Project Outline

The dataset for this project come from Jeff Sackmann [1] who is an author and sports data aggregator. After processing our data we tried different models on it and tried that to make different predictions for this game. We used a variety of models from Logistic Regression to Neural Networks to help us with our various research questions. Finally, with some tuning of these models, we were able to create a model which gave us the best predictions.

2. Background

The dataset[1] that we have contains features of players and the features of the match and type of tournament which can

be fairly informative features to predict the type of player that will win the match. The essential part of the problem is that we need to find the most important player features so that we can use these features to predict winners of any kind of match. We are dividing the data set into 67-33 split of training and test sets.

2.1 Related Work

The closely related work that has been done on the same data set predicts the betting rates on match[2] [3]. That work is different from our work because it uses more features such as score per set to predict the winner of the match. Another major difference is that we are trying to figure out the form of the player in the tournament based on the players' performance in the previous matches. We are also trying to predict the variation in the ranks of the players depending on their performance in the matches. Since we observed that player rankings change after every match we have taken that into consideration while predicting their ranks.

The questions we are trying to address here include trying to predict the winners of the tennis match. The features that ideally should be more important include the ranking of the player, hand of player i.e. if the player uses his left hand or right hand, a surface on which a player usually wins. We have enough categorical features in the dataset to train the model to solve this question. In this question, we have dropped certain columns like tournament ID, tournament name, player ID, player name as they do not have any different impact on the results and do not help us with the prediction of the winner. We have used different models for the prediction like Logistic Regression, Random Forest, Naive Bayes, and Neural Network. Out of all these models Neural Networks has proven to be the best while determining the winner.

To solve the second question we were first considering time series analysis but we did not have data that would be helpful to do this analysis. We did process the data in different ways to finally get data that can be used to predict the variation in the rank of players.

To predict the performance of the players in the tournament we are looking at the players' performance in the last 5 matches in the tournament and predicting the players form in the next matches in the tournament.

3. Dataset and Features

3.1 Dataset

The dataset that we are using for our project is retrieved from an open source data set available on Git Hub[1]. The dataset includes all matches from the from the Open Era which began in 1968 until September 2018. For the matches that were played after the year 2000, the match statistics include a number of aces hit, number of double faults, breakpoints faced and many more. We are splitting our data set into train and test using a 67-33 split.

In the dataset, the player file contains columns such as player_id, first_name, last_name, hand, birth_date, country_code. The columns for the ranking files includes ranking_rate, ranking, player_id, and ranking_points. The dataset also has ATP rankings mainly from 1983-2018 and rankings from 1973-1981 are only intermittent. For our questions, we had to drop data for some years as that data had a lot of missing values and using that was not helping our models.

3.2 Preprocessing

In the dataset, we had statistics from 1968 up until 2018 but the data from 1968 to 1990 had a lot of missing values. Since filling all those values on mere assumption was not the right thing to do so we decided to not use those data files. In those files, there were missing features as well. The dataset for ATP has been created in such a way that all winners were in a particular column and all losers were in a different column. We changed this to create a consistent dataset to be used in all of our questions. We merged the data from all the files starting from 1991 to 2018 and then we labeled the columns as "Player 1" and "Player 2" and we randomly assigned "Player 1" as either winner or loser and "Player 2" to be the other person. We also shuffled our data so we have a balanced dataset at the end of this preprocessing step. We did convert the categorical columns to numerical columns.

Data preprocessing is a technique which involves transforming the raw data that we have into an understandable format. Real-world data sometimes is incomplete, inconsistent and may contain some error. So we need to process this data before we use models in order to make some predictions. We did the following steps as part of the data preprocessing stage:

1. Data Cleaning: In this step, we checked the data and we had some missing values in it which we filled using the mean that we calculated for that column.
2. Data transformation: In this step, we normalized our columns so that we have a common scale for the entire column. We made sure that we did not lose any of the data in the columns while doing that.

3.3 Feature engineering and Selection

The dataset contained a large number of features that we could use to train our model which includes player information such as rank points, rank, age, height. It also included the match information like the best of and round. There were some columns in the dataset which did not affect our model like the tournament id, player id, player name etc. which we dropped from out initial dataframe that we used to train our model. After we trained our model, we used it to get the information of the player who can win the match.

The features that we have considered for our predictions can be divided into the following types:

1. Player information: ATP Rank points, ATP Ranks, height, age, seed, the hand of the player.
2. Match information: Match number, Match duration in minutes, best of, total rounds.

3. Performance information: Number of aces, double faults, breakpoints faced and saved; Percentages of winning on first and second serves, and first serve in rate.

For our first research question which dealt with predicting the winner of the match we had our dataset split into the following:

1. A vector of Input features (X), which described the player information for each match.
2. A target value (y), which indicates the result of the match. It can have only two outcomes i.e. 0 or 1. If player 1 won the match then the value would be 1 else it would be 0. $y \in \{0,1\}$

In addition to these features, we computed our own features like the player rank points for the current match and the previous match. We made sure that when the year changes the current and the previous rank points match. This was done to avoid any errors that might occur if the player hasn't been active each year.

4. Models and Methodology

4.1 Machine Learning in Tennis Match Predictions

Machine Learning is a field of Artificial Intelligence which gives the computers an ability to learn from the data using statistical techniques rather than programming it explicitly. In the context of tennis, the ATP tennis data can be used to form the training and testing dataset. For any particular prediction, we have modified and created the dataframes in a way that it takes into account all the features which can be useful to handle a particular research question that we are dealing with and also to which various machine learning algorithms can be applied to make a prediction.

4.2 Models

4.2.1 Logistic Regression

Logistic regression is a classification algorithm which also goes by the name Sigmoid Function. It is an S-shaped curve which takes any real-valued number and maps it to a value between 0 and 1. In Logistic Regression the hypothesis is of the form:

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

where g is the Logistic function. In this model we make an assumption about the binary classification labels i.e. they are drawn from a distribution such as:

$$P(y=1|x;\theta) = h_\theta(x)$$
$$P(y=0|x;\theta) = 1 - h_\theta(x)$$

Given a labeled set of training examples, we choose θ to maximize the log-likelihood:

$$l(\theta) = \sum_i y^{(i)} log\, h(x^{(i)}) \; + \; (1-y^{(i)})\, log(1 \; - \; h(x^{(i)}))$$

We can maximize the log-likelihood by stochastic gradient ascent under which the update rule is the following (where α is the learning rate) and we run until the update is smaller than a certain threshold:

$$\theta \leftarrow \theta \; + \; \alpha(y^{(i)} \; - \; h_\theta(x^{(i)})x^{(i)}$$

We are using the Scikit-Learn's implementation of Logistic Regression which minimizes the cost function to:

$$\min_{w,\,c} \sum_{i=1}^{n} log(e^{-y_i(X_i^T w + c)} \; + \; 1) \; + \; \frac{1}{2} w^T w$$

4.2.2 Naive Bayes

Naive Bayes is a conditional probability model. Given a vector $X = (x_1, x_2, ..., x_n)$ representing some n features, it assigns probabilities $p(C_k|x_1, x_2, ..., x_n)$ for each of K possible outcomes or classes C_k

Using Bayes theorem, the probability can be calculated by:

$$p(C_k|x) = \frac{p(C_k)\; p(x|C_k)}{p_{(x)}}$$

A Bayes classifier is the function that can assign a class label $y = C_k$ for some k as:

$$y = \underset{k\in\{1,..,k\}}{\text{argmax}}\; p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

Depending on the type of features a different kind of Naive Bayes classifier can be used such as in the case of discrete-valued features, Bernoulli and Multinomial Naive Bayes can be used.

4.2.3 Random Forest

Random Forest is a supervised learning algorithm. A model based on the random forest is faster to train. A tree in random Forest is grown by continually splitting the data based on the selected features. The split is chosen as the best split for the features.

A random forest is made up of many trees which are trained in a similar way. At a data point, the output of the Random forest is the average of the output of each tree for those data points.

4.2.4 Neural Networks

A neural network is a network or circuit of neurons. It is made of up of many layers. The input of each layer is either the output of the previous layer or the data. Each layer applies a linear transformation to the data and then an activation function which is non-linear. This is represented mathematically as

$$z^{[i]} = W^{[i]} a^{[i-1]} + b^{[i]}$$
$$a^{[i]} = g(z^{[i]})$$

Here, g is the activation function and $a^{[i]}$ is the output vector for each layer. The output for the network is the output of the last layer. The input to the first layer is the data, x.

The neural network needs to learn the biases $b^{[i]}$ and the weights $W^{[i]}$ through back propagation which uses gradient descent for updating the parameters until convergence. The batch size of gradient descent is the hyper-parameter of the algorithm.

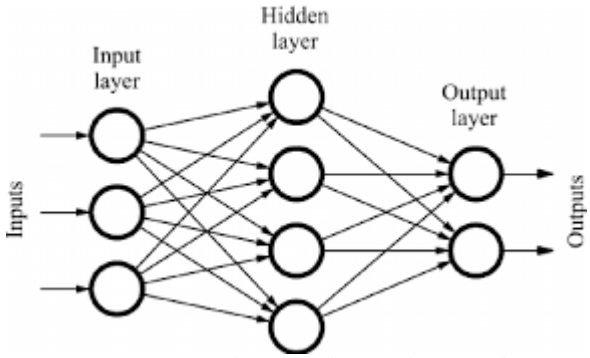

Figure 1. Feed Forward Neural Network

4.3 Methodology

We started off with some simple approaches to find a solution for our questions. One of the questions that we are solving involves predicting the winner of each match and based on that we are predicting the winner of the tournament. The one with the highest number of wins will be the winner of the tournament. We applied various machine learning models on our data to do prediction of the winner. We applied Logistic Regression, Naive Bayes, Random Forest, and Neural Networks. We initially got less accuracy on our test set but after tuning in the hyper-parameters multiple times and after a lot of trial and error we did arrive at the best possible values of those hyper-parameters which gave us a good accuracy. Out of all these models, neural networks do the best job while making a prediction. Once we got out prediction using the model, we used that to fetch the player details.

When training using Logistic Regression we changed the value of 'C' which is the inverse of regularization strength from 1 all the way up to 95 to check for various values that will give us a good model with a better accuracy. We changed the solver which is the algorithm that is used in the optimization problem along with 'C' to check which solver will help us get a better model. Finally, for a 'C' value of 95 and sag as a solver. Then we created a model using Gaussian Naive Bayes which gave us a test accuracy of 66%. We also created a model using Random Forest wherein we changed the depth of the tree and the number of estimators which is the number of trees in the forest. We tried and tested a number of values for these and finally for 140 estimators and the depth of 30 for the trees we got an accuracy of 87%. For this question, Neural Network gave us the best accuracy for Adam as the optimizer, an alpha value of 0.0001, iterations of 500. We have 3 hidden layers with 100 neurons each. We have set the tolerance value of 0.00000001 which stops the model at around 170 iterations and gave us the best accuracy of 89%.

For our second question, we are making a prediction of the ranks of the players after each match. We want to know if after the conclusion of the match the player rank increases, decreases or remains the same. For this question, we did a lot of brainstorming to get a perfect dataset which could help us. A couple of ideas that we explored included, we thought of creating a dataset in which we merge the winner and loser of a match in the same column and then do some preprocessing of the data but this resulted in a lot of data duplication which was not the best thing to proceed. Then we thought of doing a time series analysis for the same but then we did not have sufficient data for the same and it might give us a biased result. Finally, we decided on creating a new feature from the dataset which is the rank points before the match and the rank points of the players after the match. In the case of the change in the year, we made sure that the rank points are same at the start of the year for before the match and after the match. This was done because we had some players in our data who did not play for a couple of years which drastically reduced that rank points and we thought that while training the model that might result in wrong learning like the model may learn that because of that particular match, the player lost so many points so to prevent that from happening, we did match the rank points for before and after the match for the first match of each year.

Next, we performed linear regression on this data and we ended up getting a lesser R2 score for it but when we visualized the results that we obtained for the prediction that the model made, there was not much difference between the predicted rank points and the actual rank points of the players after each match. Since this required a lot of domain knowledge about how these rank points are allotted to the players based on the matches that they played we dropped this approach in order to avoid running into a situation that might result in incorrect predictions because of error on our part due to insufficient knowledge about it.

Finally, we decided to check if the rank points increases, decreased or remained the same after a match by making three classes for them. In case the rank points increased it would be in class +1, for a decrease in rank points it would be in class -1 and if they remain the same then it would be in class 0. We used many algorithms to create a model for this question. For Logistic Regression model, we obtained an accuracy of 72%. Applying neural networks on this data gave us an accuracy of 72%. With Random Forest we got an accuracy of 65% when the number of estimators was set to 100 and maximum depth was set to 20. For this approach, we got an accuracy of around 82% for the XG Boost model. We have created different models for the winners and the losers for this question.

For determining the rank of the players we got all the data for the players from 1991 to 2018 and we did a normalization as per the year. But the problem with this approach is that the number of players differs with each year which results in the number of classes and thus this did not give us a good

accuracy. The problem with this approach is that the in certain years the rank for a player can be 1 if they have a minimal 100 points whereas in the other years a rank point of even 3000 might not guarantee rank 1.

Finally, for predicting a rank we decided to use an approach wherein we predict the rank of the players year-wise. In this approach, we have a limitation which is because we can have different ranks for the same rank points and different ranks for the same rank points and hence the maximum accuracy that we have achieved using this approach is 45%. We are getting this accuracy by applying the Random Forest model to the data. But this accuracy is just for predicting the winner rank. Loser Rank prediction accuracy is just 14 %. The conclusion that we have derived is that losing a match does not affect the rank of the losing player directly. It is affected over time.

Now, we decided to encode the previous information of the player into the data. So now there are two extra columns which tell us the rank of the player before this current match was started and the rank of the player after the match ended. To do this, the rank of the player after the previous match of this player had ended is taken as the rank of this player before this match begins. We ran into another problem while doing this. Some players don't play for some years and when they come back after some years, their rank automatically goes down. Hence we have done this previous match encoding year by year so that we can take into account any absent years for a player. After doing some feature selection an accuracy of 80 % was achieved for the winning player with XG Boost Model but the losing player accuracy still stays at 14 % . Hence it is safe to say that the losing player accuracy's do not depend on just the current and the previous matches.

For our third question, we are trying to predict the player's performance in the future matches in the tournament based on their performance in the previous 5 matches. We have created a new feature i.e. the form feature for the two players in the match. Initially, we have given a random form number between 0-5 for each player. After each match, we update that forms feature based on the result of the match. After we have 5 matches for the players we use that to determine the form of the player based on the number of matches they have won. We have created a dataframe for each player and then we applied Random forest on that data and we got an accuracy of around 94% for one of the players.

5. Experiments and Results

For the first question, after we applied various models to the processed dataset we got the following accuracies:

Model	Training Accuracy	Test Accuracy
Logistic Regression	89.53	89.12
Naive Byes	66.32	66.54
Random Forest	97.43	87.36
Neural Network	99.70	89.92

In the case of our second question we have tried two approaches, one wherein we are including the previous rank information and in the other one we are excluding that information. The accuracies we obtained using the previous information are:

Model	Winner Set	Loser Set
XG Boost	80	14
Random Forest	47	12

The accuracies obtained after excluding the previous information are:

Model	Winner Set	Loser Set
Random Forest	45	12
Ada Boost	40	15

For our third question, we have placed a base limit of 800 matches to check the form of the players. We have used Logistic Regression with the C value of 95, Naive Bayes, Random Forest with a maximum depth of 10 and number of estimators as 150 and SVM with RBF kernel, out of these the best accuracy for the model is given by Logistic Regression. The result from the model is the form of those players.

6. Summary and Conclusions

With respect to the current data, we believe that we have created models which give us the best results. However, if we had a different dataset which incorporated a lot of other features that would have been relevant to the questions like importance of tournaments/matches for player, the number of matches a player had in the days before this match, the tightness in schedule and intensity of those matches, fitness of the player, number of matches the player played before in similar environment (location, weather), etc. then we can get a much better prediction for our questions.

Acknowledgments

We would like to thank our professor Dr. Gregory Rawlins for his unconditional support and guidance. We would also like to thank our Associate Instructors Zeeshan Ali Sayyed, Christopher Torres and Manoj Joshi for their guidance throughout the semester.

References

[1] Jeff Sackmann. Atp tennis rankings, results, and stats. https://github.com/JeffSackmann/tennis_atp, 2018.

[2] Michal Sipko. Machine learning for the prediction of professional tennis matches. 2015.

[3] Daniel Wright Andre Cornman, Grant Spellman. Machine learning for professional tennis match prediction and betting. 2017.