TENNIS MATCH PREDICTION USING MACHINE LEARNING

Claudia Sgarra
Department of Computer Science
University of Bari
c.sgarra2@studenti.uniba.it

Andrea Montemurro
Department of Computer Science
University of Bari
a.montemurro23@studenti.uniba.it

June 11, 2022

ABSTRACT

Tennis is one of the most followed sports in the world. For several reasons it could be interesting to predict the outcome of a match, so machine learning can be applied in such a topic to predict the result of one match by looking historical data. Hence, in this project we adopt a set of models which makes uses of the historical data up to the current year and apply various machine learning models to predict tennis match outcomes. The main objectives of our project is to predict the winner of future tennis matches based on the previously available tennis match data. We compared Logistic Regression, Decision Tree, Random Forest, Boosting (AdaBoost and Logit Boost) and a simple Neural Network to predict the match winner and the Neural Network gives us the best prediction accuracy.

1 Introduction

Tennis is an international sport, enjoyed by fans in countries all over the world. Tennis player came from all over the world and each players come equipped with different playing styles and specialties. There are a huge number of variables that define every tennis match, making the sport both exciting and unpredictable. On the other hand, predict the outcome of a match can be very useful for several reasons, for example for computing the odds of the bets, or for better planning the season of the athletes, who would certainly be more interested in participating in tournaments that would have greater chances of winning. In this project we want to compare different algorithms already known in the literature, to predict the outcomes of the matches. The dataset for this project come from Jeff Sackmann [1] who is an author and sports data aggregator. After processing the data we tried different models for predicting the outcome of the matches. We used a variety of models from Logistic Regression to Neural Networks to help us with our goal. 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, features of the matches and type of tournament which can be fairly informative features to predict the type of player that will win the match.

2.1 Related work

The first attempts in tackling the problem of tennis match prediction started in 1999 with the first regression model of Boulier and Stekler [2] to forecast tennis matches based on a player's ATP-ranking. Ever since then, interest has continued to rise, leading many researchers to develop machine learning models of varying complexities and types. Nowadays, most approaches in the literature with the goal of predicting tennis matches fall into one of three categories: point-based, pairwise comparison and regression. Pairwise comparison is a way of comparing players in pairs trough the use of some underlying metrics defining each player. A popular type of pairwise comparison model is the Bradley-Terry model [3]: it allows to predict the outcome of a paired comparison of individuals i and j, given a respective metric. Point-based models relate to a type of prediction models that in essence only try to estimate the probability of winning a point on serve and return. Assuming that every point is independent and identically distributed

(IID) from all other points in the match, the winning probability of each player can be derived in closed-form [4]. Klaassen and Magnus developed a way to estimate winning probabilities from any point in the match by modeling the tennis match as a discrete-time Markov chain, where each state represents a corresponding score [5]. Contrary to point-based models, regression models try to model the players winning probabilities directly. They achieve this by extracting a set of features describing the characteristics of each player for every match in time. The relationship between these features and winning probability is then learned using a machine learning model capable of probabilistic classification. Regression based models perform better than the other approaches when the goal is to predict winning probabilities before the match has started as they do not rely on the false assumption of IID points. On the other hand, point-based have more flexibility when it comes to in-play predictions and can even be used to estimate the chance of a particular score outcome. Since the goal of this work is to predict outcomes before the match has started, regression based methods are the most sensible choice.

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 provide access to information about players, the outcomes of matches and statistics related to player performance in particular matches, for all matches from the Open Era which began in 1968 until the current month. Table 1 summarises the most relevant data available.

3.2 Preprocessing

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 for the numerical features, and the mode for the categorical features.
2. Data transformation: In this step, we normalized our columns so that we have a common scale for the entire column.

Data cleaning In the dataset, we had statistics from 1968 up until 2022 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, we decided to not use those data files. In those files, there were missing features as well.

Data transformation 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: we merged the data from all the files starting from 1991 to 2021 (20 years of matches) 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. So in our final dataset we will have column "y" with label 0 when the winner of the game is Player 1, 1 otherwise.

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.

At the end, our final dataset contain a total of 99360 entries.

3.3 Features

The dataset contained a large number of features (45) that we could use to train our model. The features that we have considered for our predictions can be divided into the following types:

- Player information: ATP Rank points, ATP Ranks, height, age, seed, the hand of the player.
- Match information: Match number, Match duration in minutes, best of, total rounds.
- Performance information: Number of aces, double faults, breakpoints faced and saved; Percentages of winning on first and second serves, and first serve in rate.

Player details
Name
Data of birth
Country of birth
Prize money
ATP rating point over time
Match details
Tournament name
Tournament type
Surface
Location
Date
Result
Prize money
Odds
Per-match stats for both players
First serve percentage
Aces
Double faults
Unforced errors
Percentage of points won on first serve
Percentage of points won on second serve
Percentage of receiving points won
Winners
Break point
Net approaches
Total points won
Fastest serve
Average first serve speed
Average second serve speed
Odds

Table 1: Most relevant features

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. For our 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.

4 Models and Methodology

4.1 Machine Learning in Tennis Match Predictions

4.2 Models

Different machine learning algorithms exist to solve different types of problems. We can approach the tennis prediction problem in two ways:

- As a regression problem, in which the output is real-valued. The output may represent exaxt result of the match (e.g 3-1).
- As a binary classification problem, in which we can attempt to classify matches into either a 'winning' or a 'losing' category.

So we considered the tennis prediction problem as a binary classification problem.

Logistic Regression Logistic regression is a classification algorithm which apply a sigmoid or logistic function to a linear funcion of data.

$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x} + w_0\right)$$

where the sigmoid is defined as

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

It is an S-shaped curve which takes any real-valued number and maps it to a value between 0 and 1. In this model we make an assumption about the binary classification labels i.e. they are drawn from a distribution such as:

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

Given a labeled set of training examples, we learn a model that maximize the likelihood:

$$\max_{\mathbf{w}} L(\mathbf{w}) = \max_{\mathbf{w}} \prod_{i=1}^{N} p\left(t^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}\right)$$

We can convert the maximization problem into minimization so that we can write the loss function. We are using the Scikit-Learn's implementation of Logistic Regression which minimizes the cost function to:

$$\ell_{\log}(\mathbf{w}) = -\log L(\mathbf{w}) = -\sum_{i=1}^{N} \log p\left(t^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}\right)$$

$$= -\sum_{i=1}^{N} t^{(i)} \log\left(1 - p\left(C = 0 \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right) - \sum_{i=1}^{N} \left(1 - t^{(i)}\right) \log p\left(C = 0 \mid \mathbf{x}^{(i)}; \mathbf{w}\right)$$

Decision Tree The decision tree Algorithm belongs to the family of supervised machine learning algorithms. The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree. We can write the model in the following form:

$$f(x) = E[y \mid x] = \sum_{m=1}^{M} w_m I\left(x \in R_m\right) = \sum_{m=1}^{M} w_m \phi\left(x; v_m\right)$$

where R is the m'th region, w is the mean response in this region, and v encodes the choice f variable to split on.

Random Forest A random forest model is a generalization of a Random Tree Model in which several Decision Trees are trained. A tree is grown by continually "splitting" the data (at each node of the tree) according to a randomly chosen subset of the features. The split is chosen as the best split for those features (meaning it does the best at separating positive from negative examples). Given a data point, the output of a Random Forest is the average of the outputs of each tree for that data point.

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x)$$

AdaBoost AdaBoost implements boosting, wherein a set of weak classifiers is connected in series such that each weak classifier tries to improve the classification of samples that were misclassified by the previous weak classifier. The goal of boosting is to solve the following optimization problem:

$$(\beta_m, \theta_m) = argmin_{(\beta,\theta)} \sum_{i=1}^{N} L\left(y_i, f_{m-1}\left(\mathrm{x}_i\right) + \beta\phi\left(\mathrm{x}_i; \theta\right)\right)$$

It consider a binary classification problem with exponential loss.

LogitBoost LogitBoost is an alternative ensamble methods that implements boosting computing logistic loss. The advantage of this with respect to the AdaBoost is that Logit Boost, in theory, performs better with outliers. The goal is to minimze the expected log-loss, given by

$$L_m(\phi) = \sum_{i=1}^{N} \log\left(1 + \exp\left(-2y_i\left(f_{m-1}(\mathrm{x}) + \phi\left(\mathrm{x}_i\right)\right)\right)\right)$$

Neural Network A neural network is a network of neurons organized into 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. The weighted input are summed together

$$z = \sum_{i=0}^{n} w_i x_i$$

and z is then passed through a function f to produce the output

$$y = f(z) = f(\mathbf{x}\mathbf{w} + b)$$

The neural network needs to learn the biases b and the weights W 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.

4.3 Methodology

We train different models on the same dataset and we compare them by looking at the error metrics. Since we cast the problem into a classification problem, we use as metrics Precision, Recall and F1-measure.

An important aspect of this part of the experiment is the hyper-parameter tuning of our models. We used Randomized Search in order to pick the best hyper-parameters for each model that we train, hence to select the best one configuration for each of them. Moreover, we used as validation method the 10-fold cross validation.

Randomized Search for hyper-parameter tuning Machine learning models have hyper-parameters that have to be set in order to customize the model performance to the dataset.

Often the general effects of hyper-parameters on a model are known, but how to best set a hyper-parameter and combinations of interacting hyper-parameters for a given dataset is challenging. There are often general heuristics or rules of thumb for configuring hyper-parameters.

A better approach is to objectively search different values for model hyper-parameters and choose a subset that results in a model that achieves the best performance on a given dataset. This is called hyper-parameter tuning.

In such a procedure, a Search Space space is defined, which is the volume to be searched where each dimension represents a hyper-parameter and each point represents one model configuration.

Random Search defines a search space as a bounded domain of hyper-parameter values and randomly sample points in that domain, in contrast to the Grid Search that defines a search space as a grid of hyper-parameter values and evaluate every position in the grid.

We used Random Search because usually is more rapid in finding best configurations.

Best Hyper-parameters The search give us the best hyper-parameters for each model.

In the case of the **Logistic Regression**, we obtain that the model with C value equal to 1291.55, an l2 penalty and with the "SAG" solver.

For the **Decision Tree** we obtain that the best model has a Max Depth equal to 65 and uses the square root of the total features that we have, while for the **Random Forest** we obtain also that the max number of estimator is equal to 190.

We used also another ensemble method, **AdaBoost** that uses at most 180 estimators, but this one performs less better with respect to the **Random Forest**.

Finally we also tested a simple **Neural Network** for our problem, and we choose its parameters with a trial and error approach. Of course, it has an input layer with the same number of neurons as the features that we have, and then there are three fully connected layers stacked. We use dropout and HE initialization in the three hidden layers and ReLU as activation function. Since we are in a binary classification problem, we use a single neuron in the output layer with the sigmoid activation function, and also we train using the Adam optimizer and with the Binary Cross-entropy as loss.

Model	Precision	Recall	f1-score	Accuracy
Logistic Regression	0.91	0.91	0.91	0.91
Decision tree	0.72	0.72	0.72	0.72
Random forest	0.90	0.90	0.90	0.90
AdaBoost	0.87	0.87	0.87	0.87
Logit Boost	0.87	0.87	0.87	0.87
Neural Network	0.93	0.93	0.92	0.93

Table 2: Table with performance metrics.

5 Results and conclusion

As showed in the Table 2, the model that perform better is our simple Neural Network, which have an overall accuracy of 93%. In contrast, the worst method is the Decision Tree with an accuracy of 72%. The best ensemble method is Random Forest with an accuracy of 90%, and both the boosting methods have the same accuracy, equal to 87%, but are very slow in the training. We have to notice that also a simple method like a Logistic Regression reach very good performance (90%), but the Neural Network scale also better with our big dataset, since the training is performed in a shorter time.

References

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

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

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

[4] L. Vaughan-williams Chunping Liu Hannah Gerrard, "How well do ELO-based ratings predict professional tennis matches?" Nottingham Trent University, Tech. Rep., 2019.

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