University of Liège

Big Data Project

Review 1 - Pre-analysis, literature review

Master 1 in Data Science & Engineering

Authors:
Tom Crasset
Maxime Lamborelle
Antoine Louis

Professors :
G. Louppe
P. Geurts
B. Cornelusse

Academic year 2018-2019

1 Previous Results

Previously, the statistics about players were fixed and taken in October 2018 (date at which we crawled the web). Thus, R. Federer for example has had the same statistics for a tournament played in 2006 as for a tournament played in 2017. This model did not take into account the evolution of the players and is this has as consequence that the model is biased.

Moreover, the training set contains all ATP matches (about 50 000) matches. The different models we tried have an accuracy on the testing set of about 65%. But when we try to predict the winner of the tournament (predicting the results of all possible matches between the first 32 seeds), the model shows great difficulty to predict the winner (probability of 50% chance to win for both player).

2 New Data Set

To solve these problems we had only one solution : change our dataset. We decided to take the dataset from JeffSackmanns Github repository [1]. All data about matches between 1993 and 2018 are merged into a single `csv` file of almost 80 000 games.

To solve the second problem and better distinguish the best players, we only keep data from matches which oppose two players that are ranked between the first and the 32nd place at the ATP ranking. This leaves us with a dataset of 7800 matches which is large enough to train a model.

2.1 Compute player statistics

From this dataset, we now have to compute the players statistics for all matches. These statistics are no longer constant for a player and will change for each of its matches.

To compute a particular statistic of a player, we take a weighted sum of this statistic over all the previous matches of this player. If x_1, x_2, ..., x_n are all the statistic of a player from his first game and the game n, then for that statistic we compute the feature:

$$\mathbf{x} = \frac{\sum_{i=1}^{n} W_i x_i}{\sum_{i=1}^{n} W_i} \tag{1}$$

where W_i is the weight of the i^{th} match of the player. This weight is computed as a decreasing function of the time. Indeed, a game which was played 10 years ago has to have a very low weight whereas a game which has been played in the current year has to have a bigger weight. This weight represents the players current performance.

Formally these weights are computed using the function :

$$W_i = 0.8^{\Delta t}$$

where $\Delta t = t_{now} - t_{past}$ is the elapsed time between the considered matches (in years). This function is shown in FIGURE 1.

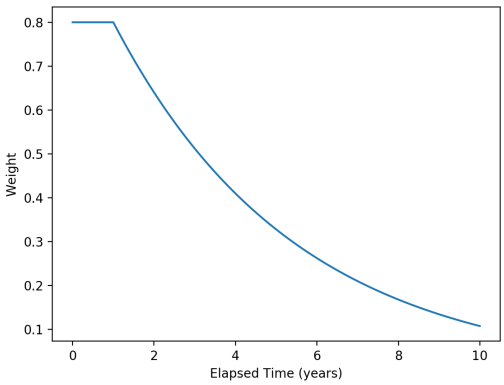

Figure 1: Time discounting function

2.2 Symmetric Model

We want our model to be symmetric. If player A wins against player B then player B loses against player A. To do so, for each game of our dataset we create two entries in our learning set. One where player A plays against player B and one where player B plays against player A. Thanks to this technique, our model is symmetric but also we have doubled the number of entry in our learning set which contains now 15 600 entry.

2.3 Summary

At the end of the process we have a dataset which contains 48 features. To make our model work better we need to normalized our data thanks to the function `scale` from the sklearn library.

The Table 1 shows the summary of the features.

Information about the game	**Information about players**
Year	Name
Day	Height
Number of sets	Age
Surface	Rank
Draw Size	ATP Points
Tourney level (ATP, Master, Grand Slam,..)	Number of ace
Round (semi-final, final, ...)	First Serve Won
	Second serve won
	Break point Saved
	Break point faced
	Service games won
	Service points won
	Double Faults

Table 1: Features

3 Model

As the seeds of this years Roland Garros tournament aren't available, we're going to test our model on the 2018 French Open. To do this, we're going to use the last match that each player has competed in to judge his performance in the tournament.

We will now test several model on this dataset using a test train split of 0.2 and the results are shown on Table 2. We made sure that the testing dataset to compute the accuracy was not taken into account when computing the feature selection. Otherwise, we would have had a contamination and the true accuracy of the model would be incorrect. Another thing to point out is that the test set should be matches that were played after the ones the model was trained on. Indeed, one can't use data from the future to predict what happened in the past.

Models	**Test accuracy**
Log. Regression	0.41
MLP	0.58
RandomForest w/o feat. selection	0.61
RandomForest	0.67

Table 2: Testing accuracy of different models using a subset of features except iii) : i) Logistic regression ii) MLP with 1 hidden layer of 100 neurons iii) RandomForest iv) RandomForest with the most important features

In Figure 2, the importance of every feature is shown. They were computed using a large RandomForest and averaged over 10 folds.

After the best 6 features, a certain stagnation in importance can be noticed and then another sharp drop after 15 features. However, one thing to notice is that the best feature, *bestof_diff*, is the only one among the 15 best features that depends on the match. Moreover, that feature can only take 2 values, a match is either a best of 3 or a best of 5. Our guess is that this feature should be disregarded. After recomputation of the best features by omitting *bestof_diff*, the best features are depicted in Figures 3 and 4.

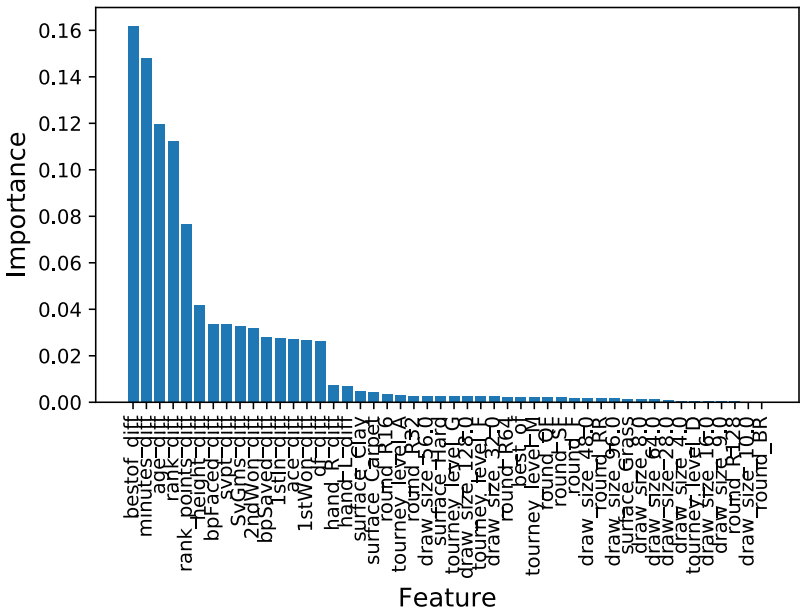

Figure 2: Feature importances calculated as an average over 10 folds.

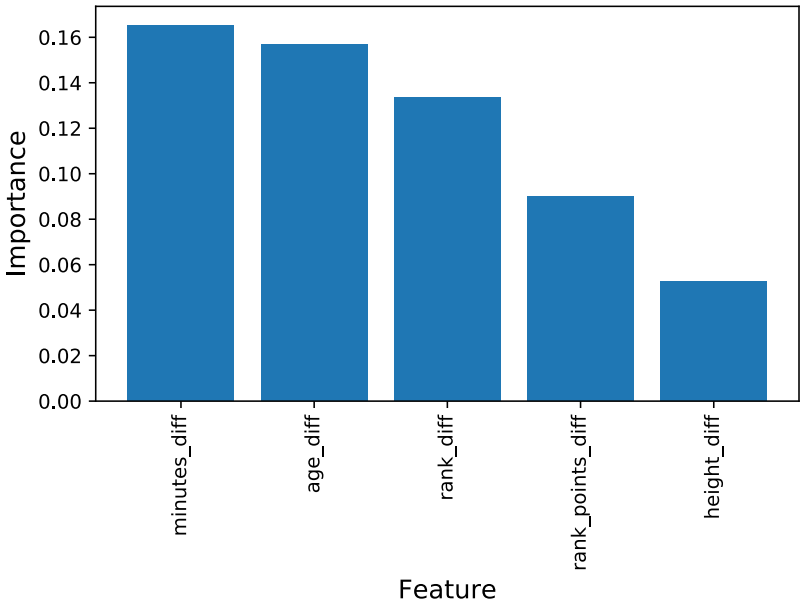

Figure 3: Best 5 features calculated as an average over 10 folds.

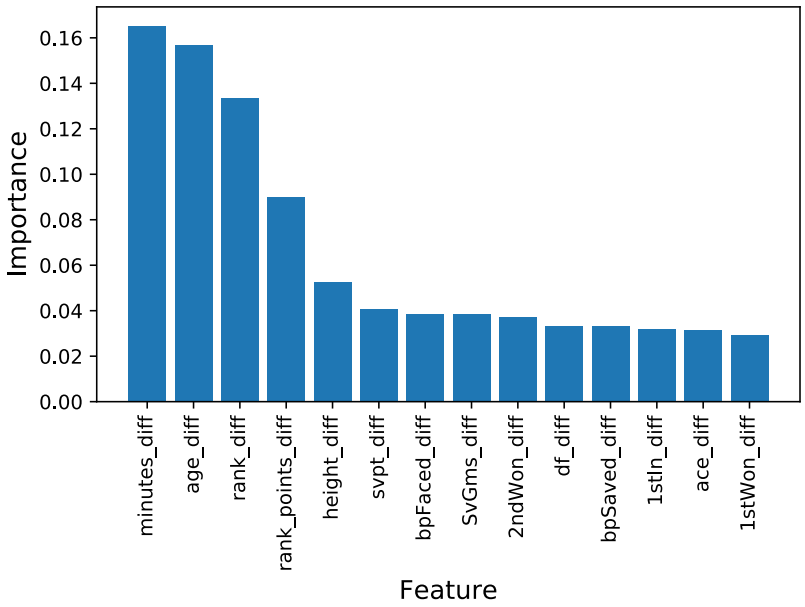

Figure 4: Best 14 features calculated as an average over 10 folds.

In Table 2, one can see that reducing the number of features to 5 features significantly improves our accuracy on the RandomForest model with all the features. This might be due to the fact that the rest of the features made the model overfit to the data. If we further reduce the amount of features, the accuracy decreases , which is expected and the model underfits the data.

Just as before, we used combinations to make every seed play against each other , and then predict the probability of each of the players to win this match. Then, using a Monte Carlo simulation over all the possible draws, we are able to predict a winner of the tournament. Using the best accuracy model, 2 players stand out in our simulation, as shown in table 3. All other players had a probability of winning smaller than 0.04.

However, in the actual tournament, Shapovalov lost in the 2nd round and Berdych lost in the first round. Our model predicted that Nadal, the actual winner of the tournament, would win with a probability of only 0.00981. Unfortunately, we can recompute the tournament with the actual draw of Roland Garros 2018 as we chose to only use seeds, by making the assumption that it would be one of them that would win the match.

Name	Probability
DENIS SHAPOVALOV	0.342
TOMAS BERDYCH	0.226

Table 3: Probability of the player winning the 2018 French Open

4 Further improvements

One possible improvement that could be made is to use surface weighting. As tennis is played on different surfaces, one can expect that each player has a favorite surface and moreover, each surface impacts the plays differently e.g. the bounce of the ball. Thus, for predicting an upcoming match on a particular surface, a player's past matches on the same surface will be more informative than those on other surfaces.

Also, some surfaces are more similar than others and such the skill of a player on one surface might be more or less correlated to his skill on another surface. It is well known that there is a much higher correlation between performance on grass and hard courts than between grass and clay courts.

As we did with time discounting, we could assign a higher weight to the previous matches that have the same or similar surface as the one we are trying to compute.

References

[1] Sackmann J. (2015-2018). tennis_atp, *GitHub repository*. Retrieved from https://github.com/JeffSackmann/tennis$_a tphttps : //github.com/JeffSackmann/tennis_atp$.