Analysis of Tennis Matches Dataset

Data Mining Project

Jacopo Bandoni
Alex Colucci
Domenico Romano

Dipartimento di Informatica
Università di Pisa
15/11/2021

Contents

1 Introduction

TODO

2 Data understanding

The `tennis_matches` dataset contains 186128 total observations. Below, the detailed analysis for each individual attribute.

tourney_id The identifier of the tourney. On 185764 observations, there are 4854 unique tourneys and 55 missing.

tourney_name The name of the tourney. On 185794 observations, there are 2488 unique names and 25 missing. The unique values are lower than those of tourney_id, hence more tournaments with the same name have been played over the years. Moreover, several naming conventions are used from which partial information about the host city name, prize, nationalities could be scraped.

surface The type of surface on which the players had the game. On 185793 observations, 26 are missing. The types are the following with their respective occurrences: Hard (95127), Clay (81013), Grass (6600), Carpet (3053).

draw_size The number of players in the draw, that is often rounded up to the next power of 2. The most common draw size for a tourney is 32 followed by $2, 4, 64$.

tourney_level The level of the tourney that is different for men and women, and hence it gives information about the sex of a tournament. On 185790 observations 19 are unique and 25 are missing. They are mostly character, but for ITF competitions it's an integer that states the prize. The only ambiguous level is the D that is used both for men and women, but it can be disambiguated thanks to the fact that for men it's used only in the tourney whose name is "Davis Cup F." Moreover, O is a category used to denote the Olympics, where both men and women can participate.

tourney_date The date when the tourney started. On 185791 observations of which 28 are missing. The matches were disputed between the year 2016 and 2021, with most of them in the range $2016 - 2019$. For the months, instead, November and December tends to be the one with fewer matches. More than 97% of them are on Monday.

match_num A match-specific identifier. Often starting from 1, sometimes counting down from 300, and sometimes arbitrary. On 186103 observations, 27 are missing.

winner_id, loser_id The player_id used in this repo for the winner/loser of the match. On 186103 observations, 55 for the winners and 28 for the losers are missing.

winner_entry, loser_entry The player_id used in this repo for the winner/loser of the match. On 186103 observations, 160008 for the winners and 141731 for the losers are missing.

winner_hand, loser_hand The hand used by the player. For ambidextrous players, this is their serving hand. On 186103 observations, 46 for the winners and 98 for the losers are missing. A possible value for those attributes is 'U' which stands for unknown, and it matches $49k$ and $62k$ for winners and losers respectively.

winner_name, loser_name Winner's/loser's name. On 186103 observations, 27 for the winners and 31 for the losers are missing.

winner_ht, loser_ht Winner's/loser's height in centimetres. On 186103 observations, 136516 for the winners and 147489 for the losers are missing.

winner_ioc, loser_ioc Winner's/loser's three-character country code. On 186103 observations, 29 for the winners and 26 for the losers are missing.

winner_age, loser_age Winner's/loser's age, in years, depending on the date of the tournament. On 186103 observations, 2853 for the winners and 6537 for the losers are missing.

round An acronym which identifies the stage of the match inside the tournament (e.g., 'f' stands for 'final', 'sf' for 'semifinal' and so on).

score The score of the match. On 186103 observations, 175 are missing. Every couple n1-n2 (e.g., 6-4) represents the score of a single set, where n1 are the games won by the winner and n2 those won by the loser of the match. When after the couple of numbers representing a set there is a number n3 between brackets (e.g., 7-6(4)), it means that the set ended at the tie-break and n3 represents the points scored during it by the loser of the set. When we find a couple between square brackets (e.g., [10-7]), it represents the result of the super tie-break, which is played in some tourneys, on the 6-6 of the last set (on the 12-12 in Wimbledon). In these cases, the score of the final set is omitted. We can also find some abbreviations, which indicate particular conditions:

- `RET, Ret., RE, RET+64`. Placed at the end of the score, to indicate the retirement of a player during the match.
- `W/O, Walkover`. It's the retirement of a player before the match starts.
- `DEF, Def.`. It's a default, i.e., the disqualification of a player.
- `BYE`. It's the automatic advancement of a player to the next round of a tournament without facing an opponent.

Furthermore, there are some errors: not recognized characters (16 times), HTML non-breaking spaces (15), "RET" without a score before (8), "2-May", "1-Feb" and a score with a wrong formatting.

best-of The maximum number of sets of the match. If "3", it means that the first player to achieve 2 sets, wins the match. If "5", a player must achieve 3 sets to win. On 186103 observations, 0 are missing.

minutes The duration of the match. On 186103 observations, 104461 are missing.

w_ace, l_ace Winner's/loser's number of aces. On 186103 observations, 103811 for the winners and 103808 for the losers are missing.

w_df, l_df Winner's/loser's number of double faults. On 186103 observations, 103809 for the winners and 103802 for the losers are missing.

w_svpt, l_svpt Winner's/loser's number of serve points. On 186103 observations, 103811 for the winners and 103806 for the losers are missing.

w_1stIn, l_1stIn Winner's/loser's number of first serves made. On 186103 observations, 103811 for the winners and 103817 for the losers are missing.

w_1stWon, l_1stWon Winner's/loser's number of first-serve points won. On 186103 observations, 103809 for the winners and 103810 for the losers are missing.

w_2ndWon, l_2ndWon Winner's/loser's number of second-serve points won. On 186103 observations, 103812 for the winners and 103809 for the losers are missing.

w_SvGms, l_SvGms Winner's/loser's number of serve games. On 186103 observations, 103810 for the winners and 103803 for the losers are missing.

w_bpSaved, l_bpSaved Winner's/loser's number of breakpoints saved. On 186103 observations, 103806 for the winners and 103810 for the losers are missing.

w_bpFaced, l_bpFaced Winner's/loser's number of breakpoints faced. On 186103 observations, 103809 for the winners and 103815 for the losers are missing.

winner_rank, loser_rank Winner's/loser's ATP or WTA rank, as of the tourney_date, or the most recent ranking date before the tourney_date. On 186103 observations, 19402 for the winners and 35259 for the losers are missing.

winner_rank_points, loser_rank_points Number of ranking points. On 186103 observations, 19420 for the winners and 35276 for the losers are missing.

tourney_spectators The number of total spectators of the tourney. On 186103 observations, 27 are missing.

tourney_revenue The total tournament earnings. On 186103 observations, 26 are missing.

3 Data cleaning and transformation

First we switched all the letters in lowercase, since in some cases there were equivalent values considered as different (e.g., "US Open" and "Us Open" in tourney_name), we also removed double spaces where present because they could lead as well into the same kind of problems. Then we applied the following changes:

score We set to NaN all the erroneous values. We also added the omitted final set score in the matches with super tie-break, in order to be able to compute the number of games won by the winner and those won by the loser. Lastly, we uniformed the different strings that represented the same concept (e.g., "w/o" and "walkover").

minutes This attribute was full of outliers. In particular, we noticed that all the values grater than 396 were not possible, also in relation to the score of the matches. So we substituted these values with NaN.

winner_ht Several outliers with winner_ht = 2.0 and also other outliers with height ¡ 146. In this case, we decide to search on the internet the correct values and replace those with the outliers.

winner_ht Several outliers with loser_ht = 2.0, even in this case we decide to search on the internet the correct values and replace those with the outliers.

winner_age There were two players with two occurrences having as winner_age 95 years. Then we decide to replace those values with the medium age of the corresponding players.

w_svpt, w_1stIn, w_1stWon, w_2ndWon and corresponding of the loser We noticed that for all these attributes there were 5 recurrent records in which the values were extremely high and not possible, also in relation to the score of the matches. We dropped these records.

gender For each player where the gender was missing, we replace that gender by checking in all his game the most frequent gender's opponent.

4 Players dataset

4.1 Feature engineering

matches_won_ratio The ratio between the number of the total games won and the total numbers of games played.

mean_performance_index, max_performance_index, min_performance_index The minimum, the maximum and the average value of the performance index, which is the ratio between the number of matches played by the player in a tourney and the number of matches he should have played in order to win the tourney.

mean_minutes, max_minutes, minutes_entropy The average, the maximum, and the Shannon entropy of the duration of the matches played by a player.

rel_ace, rel_df, rel_1stIn, rel_1stWon, rel_2ndWon The average of the ratios between the statistics (ace, df etc.) and the number of serve points of the player in the single matches.

1stWonOnTotWon The average of the ratios between the first serve point won and player's total serve points in the single matches.

2ndWonOnTotWon The average of the ratios between the second serve point won and the total points won by the player in the single matches.

rel_bpFaced The average of the ratios between the breakpoints faced and the player's total serve points in the single matches.

rel_bpSaved The average of the ratios between the breakpoints saved, and the breakpoint faced by the player in the single matches.

rel_ptsWon The average of the ratios between the points scored and the total by the player in the single matches.

rel_gmsWon The average of the ratios between the breakpoints saved, and the breakpoint faced by the player in the single matches.

lrpOnAvgrp The ratio between the player's last ranking points (last_rank_points) and his average ones (mean_rank_points).

lrpOnMxrp The ratio between the player's last ranking points (last_rank_points) and the maximum ones he ever achieved (max_rank_points).

Other trivial features name, gender, ht, age, hand, total_tourneys_played, total_matches_played, total_matches_won, last_rank_points, mean_rank_points, max_rank_points, variance_rank_points, mean_tourney_spectators, max_tourney_spectators, mean_tourney_revenue, max_tourney_revenue.

4.2 Features cleaning

At this stage we performed an initial cleaning of those features by filling some features (*age*, *mean_rank_points*, *max_rank_points*, *last_rank_points*, *variance_rank_points*) being null for some records by using the mean across all the records.

5 Clustering

In this part we explore an in-depth comparison of different clustering algorithms, such as K-Means, DBSCAN, Hierarchical. Then a small parenthesis is opened on other options such as the Expectation-maximization algorithm, X-Means and Fuzzy C-Means.

5.1 Features selection

At the beginnig of our research we tried to find a pattern that could relate the style of the player (*rel ace*, *rel df*, *rel 1stIn*, *rel 1stWon*, *rel 2ndWon*) with respect to his strength. After feature engineering, clustering and analysis we found no meaningful results. So, we decide to switch to the analysis that we will present in this report.
We have decided to select the features respecting the following criteria in order

- Selection of those features that may provide an interesting picture about the performance of the players.
- Dropped all the couple of features that have more than '70%' of correlation. Leaving just one feature per correlated pair.
- Removed features deemed unimportant that had a good percentage of null values.

As a results we obtained the following uncorrelated features:

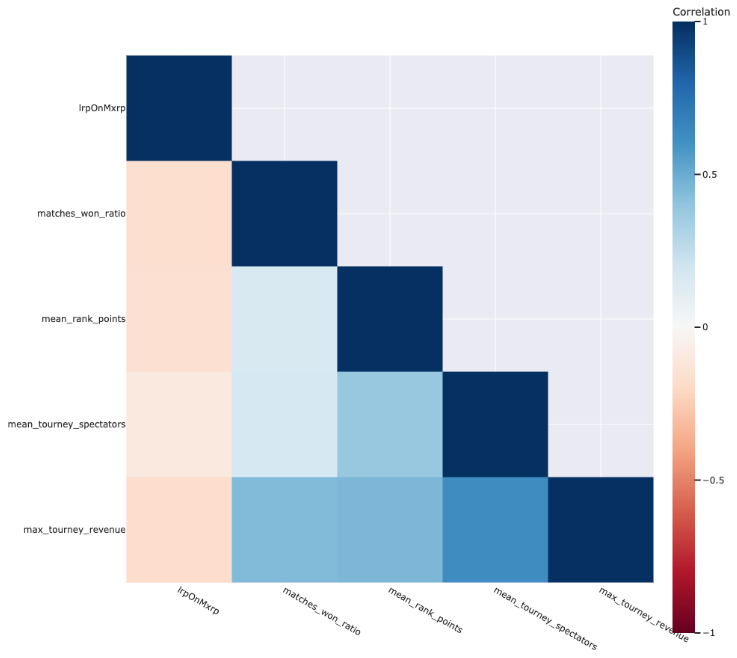


Figure 1: Correlation Plot

At this point the remaining features were: 'lrpOnMxrp', '*matches_won_ratio*', '*mean_rank_points*', '*mean_tourney_spectators*', '*max_tourney_revenue*'.
Those features will be used as a starting point in the clustering.

5.2 Pre-processing

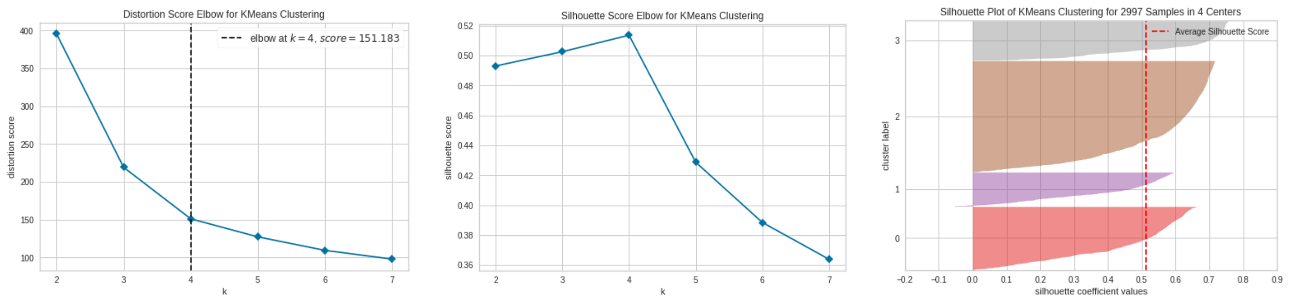
In order to prepare the data for the clustering, other steps were made:

- Additional cleaning
  - There were no null values present in the data: by dropping nan values from *lrpOnMxrp* feature.
  - Higher quality data: after iterating between feature selection and clustering, we found out that by filtering players just by taking the one who played at least 15 games, we were able to gain higher quality results.
- Normalization of the data with MinMaxScaler in order to assign the same weight to each clustered feature.

Given the steps performed, the final number of players used to cluster is equal to 2997.

5.3 K-means

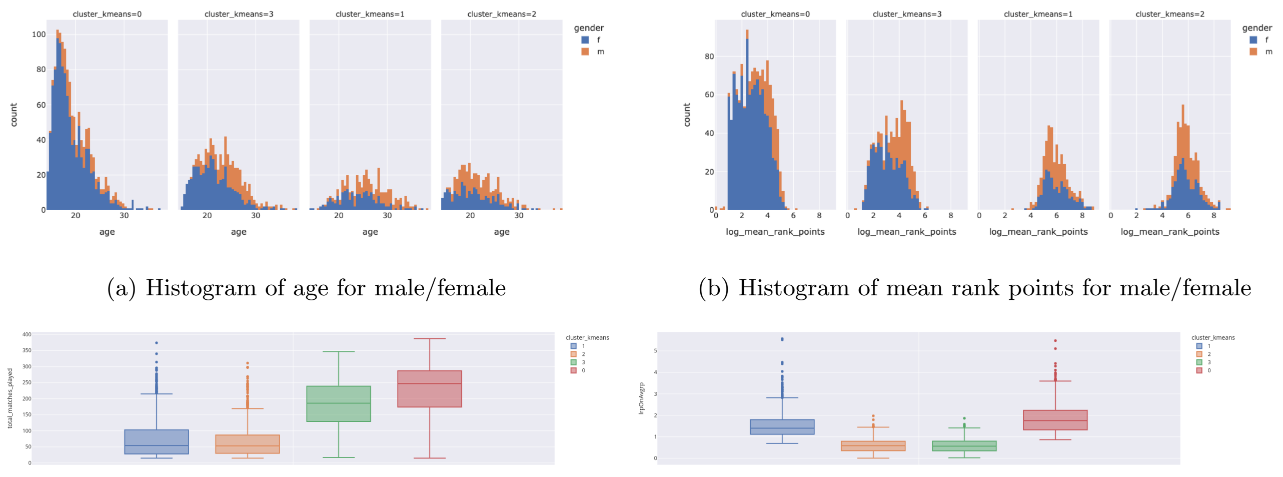
The first step was to find the **optimal K**. It was done by following the elbow rule, that suggested to use $k = 4$ where the SSE score was 151.183 (Fig. 2a). On the other hand, even the Silhouette score had the maximum value with $k = 4$ (Fig. 2b), hence it was trivial to decide that the optimal k overall was $k = 4$. Moreover, the Silhouette score per cluster is represented in Fig. 2c and each cluster is balanced enough. Cluster 1 is the only one where a tiny amount of points has a negative score. Nonetheless, the overall balance is extremely good, because there is no presence of clusters with a silhouette score below the average and there are no wide fluctuations in the size of the silhouette plots.



(a) Elbow rule (b) Silhouette score (c) Silhouette score average

Figure 2: K-means metrics

5.3.1 Results interpretation



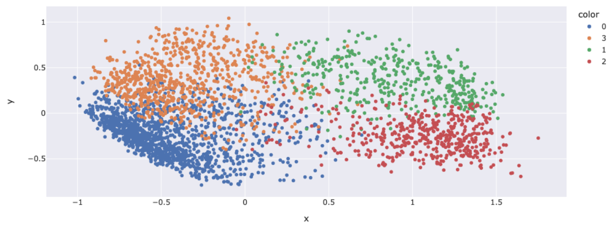
(a) Histogram of age for male/female (b) Histogram of mean rank points for male/female

(c) Box plot of total matches played (d) Box plot of last rank points on average rank points

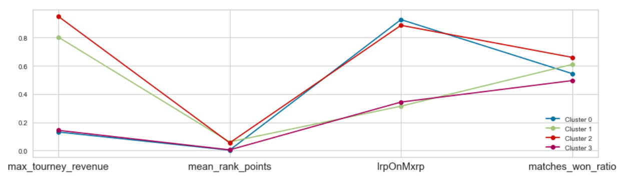
Figure 3: Distributions of features within the dataset

The clustering algorithm manages to find interesting groupings. Analysing Figure 3a, it can be seen that cluster 0 has a lower average age and cluster 1 a higher one. On the other hand, in Figure 3b you can see a clearer division compared to the case described above and this is also due to the fact that it was used for clustering. Clusters 0 and 3 have lower average rank points than the two clusters. In addition, one can appreciate how the clustering found an extremely similar grouping regardless of gender, and this is also true for the other features that are not represented in this report but can be found in the attached Jupyter notebook. In Figure 3c, instead, a clear division can be seen in terms of the experience accumulated by the players; in clusters 1 and 2 the number of games played is on average lower than in the other two clusters. Finally, in Figure Image 3d, we can appreciate the grouping by last rank points on average, which expresses the trend of the performance and cluster 0 and 1 have an average value that expresses an improvement, while the other two a worsening performance.
The interpretation we gave to the results came from looking at the graph of centroids (Fig. 4b) and looking at external features not used in the algorithm that seemed relevant, resulting in the following:

- **Cluster 0** represents the **young promises** (45.04%): those with low mean_rank_points with an increasing average trend of growth. They have the lowest age and a low experience.
- **Cluster 1** represent the **old glories** (13.64%): those with good rank points with a decreasing performance. They are the oldest and with a high experience.
- **Cluster 2** represents the **good players** (15.81%): those with good rank points with an increasing performance. They have an average age and with a high experience.
- **Cluster 3** represents the **bad players** (25.49%): with low rank points and a decreasing trend of growth. They have an average age and low experience.



(a) PCA visualization



(b) K-means centroids

Figure 4: K-means visualization

In Figure 4a we compute the Principal Component Analysis related to the whole feature of the dataset, and we plot with respect to the two greater components, here we can depict the good separation among the clusters that the k-means performed.

5.4 Density based

The set of feature used for DBSCAN is the same as K-means, as well as for the applied transformations.
To understand a good range of values for the `eps` parameter, we print the plot in Figure 5 to check the distances between the k-th nearest values for each possible point. Each distance is plotted in the x-axis, ordered with respect to the k-th nearest value.

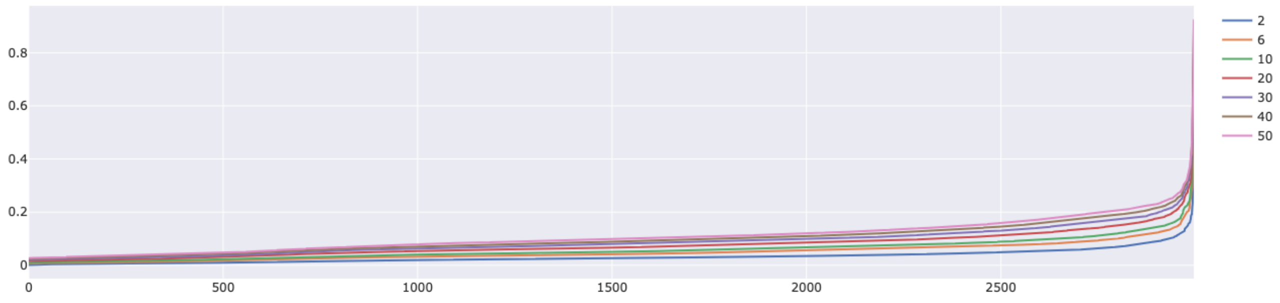


Figure 5: Noise points with the k-th nearest neighbor at farther distance

Then, in order to find the proper parameters, we run a **grid-search** (Fig. 6) using a range of values for $eps = [0.1, 0.3]$. We also calculated the silhouette score, although it is not the best metric for DBSCAN it did allow us to make a decision and choose between different parameters. The rationale behind the choice of parameters was the following. Looking at the graph, we see that for large values of `eps`, all points belong to one and only one cluster. For very small values, either the number of clusters is very large or all points are classified as noise. In addition, to reduce the options even further, probably it makes sense to consider number of clusters ranging from 2 to 4, hence $0.01 < eps < 0.25$. Finally, we took a high mean noise point distance to make sure that no dense clusters of noise formed, and at the same time for equal values we considered those values with a higher silhouette score. At this point, trying different combinations of parameters, one of the best choice was $eps = 0.2, n = 6$, to ensure a small number of outliers. With different parameters, either the number of outliers increased or clusters with a dozen elements were formed. In the next section, we comment on the results.

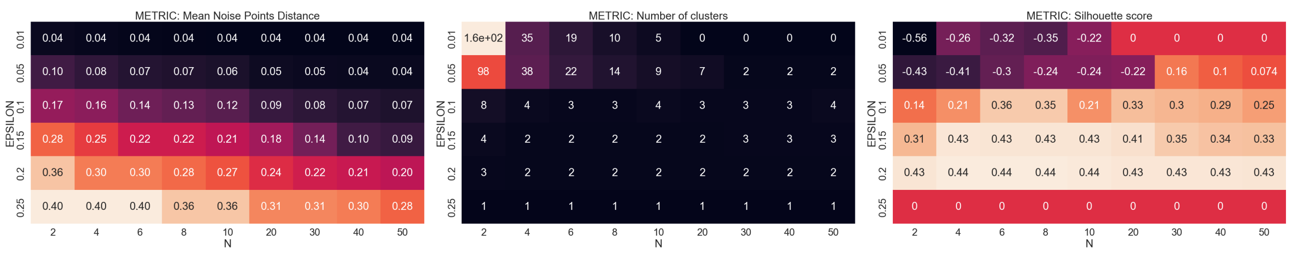
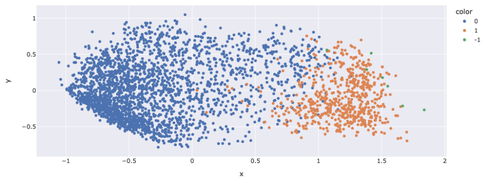


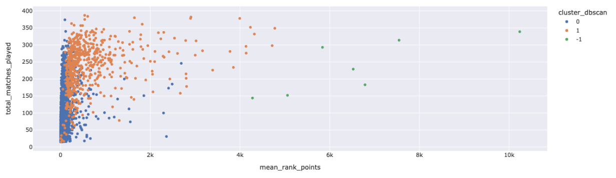
Figure 6: Chosen hyper-parameters $eps = 0.2$ and $n = 6$

5.4.1 Results interpretation

The DBSCAN identified two clusters that are extremely heterogeneous and produced the following results which can be seen in Figure 7a, where the PCA visualization is shown, while in Figure 7b where one can see a Pareto distribution, where 80% of the players are poor and 20% are good. In more detail, the results can be interpreted as follows:



(a) PCA visualization



(b) Results of DBSCAN

Figure 7: DBSCAN visualization

- **Cluster 0** represents the **average Joe** (80.08%). They have an average rank points of 72, and the average number of matches played is 81. Their trend of growth is slightly increasing, with a value of 1.14.
- **Cluster 1** represents the **good players** (19.68%). They have an average rank point of 658, and fairly experienced with an average number of matches played that is 237. Their trend of growth is increasing, with a value of 1.54.
- **Outliers** are the **Gods of tennis** (0.23%), such as Novak Djokovic, Rafael Nadal, Roger Federer, Simona Halep, Serena Williams. They have an outstanding average rank points of 6609 and an average age of 29 years (the average is 22 years old)! It's interesting to see that their average number of matches played is the same as Cluster 1, so the experience is the same. Moreover, their average trend of growth is decreasing with a value of 0, 71.

5.5 Hierarchical

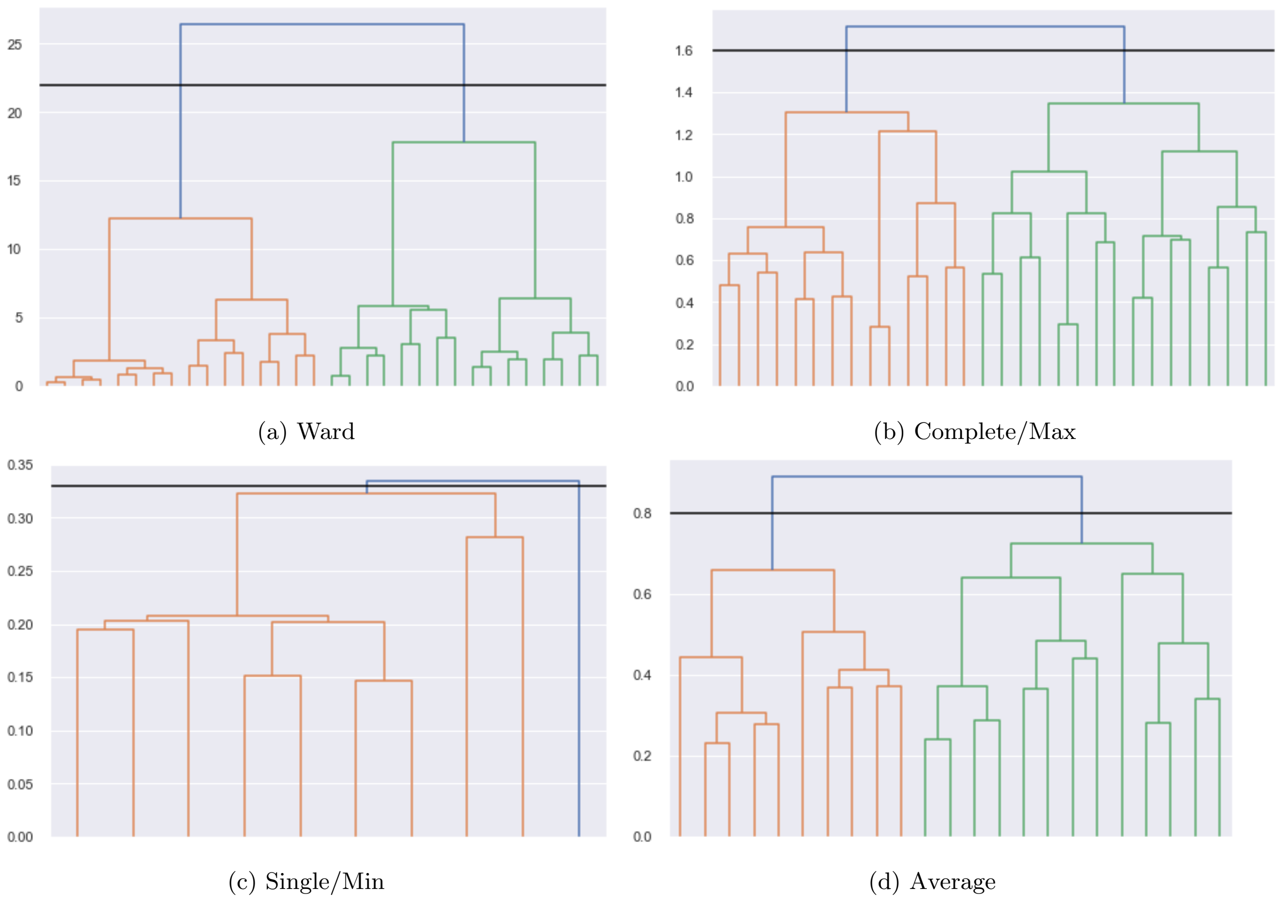
The set of feature used for DBSCAN is the same as K-means, as well as for the applied transformations.
The agglomerative hierarchical clustering was executed with Euclidean distance and with different linkage method for the inter-cluster similarity such as **Ward**, **Complete**, **Single** and **Average**. The corresponding representation is shown in the graphs in Figure 8, which only shows the last 9 merges. The number of clusters was chosen based on the Silhouette score and by trying to achieve a reasonable number of clusters: between 2 and 6. We also used the dendrogram as a proxy to study the similarity between various clusters with respect to a fixed *n_clusters* parameter.

5.5.1 Results interpretation

As we could expect, Max shows greater distances with respect to Min, while Average falls in between. This is obviously due to how the different distances are computed.
Regarding the qualitative results shown in Table 1, we can see how the *Average* method results the best both in terms of Silhouette score both in terms of homogeneity among the cluster's size. The *Single* method, on the other hand, performs the worst in terms of homogeneity and in terms of Silhouette score and by increasing the number of clusters, the results were a high number of singletons. In general, for the other methods, increasing the number of clusters led to a situation where the clustering tended to deteriorate in terms of silhouette metrics,

as can be seen in the table.
So taking the **Average method** with 2 clusters as a reference, the results can be interpreted as follows:

- **Cluster 0** represents the **good players** (30.36%): those with an average rank points of 577.62 and fairly experienced with an average number of matches played that is 201.26.
- **Cluster 1** represent the **bad players** (69.63%): those with an average rank points of 40.30 and with a low amount of experience that is an average number of matches played that is 73.31.



(a) Ward (b) Complete/Max

(c) Single/Min (d) Average

Figure 8: Dendrograms for hierarchical clustering

Linkage	K	Clusters'size	Silhouette
Average	2	2062, 935	0,482
Ward	2	2110, 887	0,476
Ward	4	1338, 772, 541, 346	0,467
Complete	2	2333, 664	0,463
Single	2	2996, 1	0,444
Average	4	1545, 924, 517, 11	0,421

Table 1: Comparison between different linkage method ordered by Silhouette score

5.6 Comparison

After experimenting with the different clustering algorithms, we can draw conclusions by referring to the results obtained, which are shown in Table 2.

- **K-means** identifies 4 fairly uniform clusters and manages to describe both weak and strong players. And for both, it identifies those that are going up and those that are going down in terms of performance. It also manages to achieve the highest Silhouette score compared to the other proposed methods.

- **DBSCAN** is the one that identifies and describes in a better way the players that are excellent at tennis, and in more general term is really good at identifying players outside the norm. At the same time, it's not great at clustering players into multiple groups that are acceptably balanced.
- **Hierarchical** the clustering results is fairly similar to the K-means both with 2 and 4 number of clusters, however in either cases it results in a lower Silhouette score.

In conclusion, the algorithm that best describes the types of players within the dataset is K-means.

Algorithm	**K**	**Clusters'size**	**Silhouette**
K-means	4	1350, 764, 474, 409	**0.513**
DBSCAN	2	2400, 590, (7)	0,476
Hierarchical (Average)	2	2087, 910	0,486
Hierarchical (Ward)	4	1338, 772, 541, 346	0,467

Table 2: Comparison between the different clustering algorithms

5.7 Other algorithms

In addition to the algorithms analysed above, we have also experimented with other algorithms available in the PyClustering library.

5.7.1 Fuzzy C-Means

The first was C-means which is a fuzzy algorithm, in other words it is soft clustering. The algorithm was initialized with k++ initializer to find the centroids, and then we played with the parameter indicating how fuzzy the results should be, first setting $m = 1.5$ and then $m = 2$. In the first case, the results were almost identical to k-means (Sec. 5.3), in the second case they deviated slightly, with small variations in cluster size, but the average remained particularly robust, in other words our interpretation of the clusters remained practically unchanged. Moreover, the Silhouette score decreased from 0.513 to 0.512.

5.7.2 Expectation Maximization Algorithm (EMA)

The algorithm was first used with the aim of finding a number of clusters of 4, but from a semantic point of view the results were not very different from those obtained in k-means, but the Silhouette score was much lower, so from a qualitative point of view we discarded this one. Then we tried a number of clusters equal to 2, thus obtaining the first with 1989 elements, the second with 1008 and a Silhouette of 0.458. A visualization of the result can be appreciated in the PCA reported in Figure 9. The interpretation that can be given is the same as that defined for the hierarchical (Sec. 5.5), but all in all with a lower Silhouette.

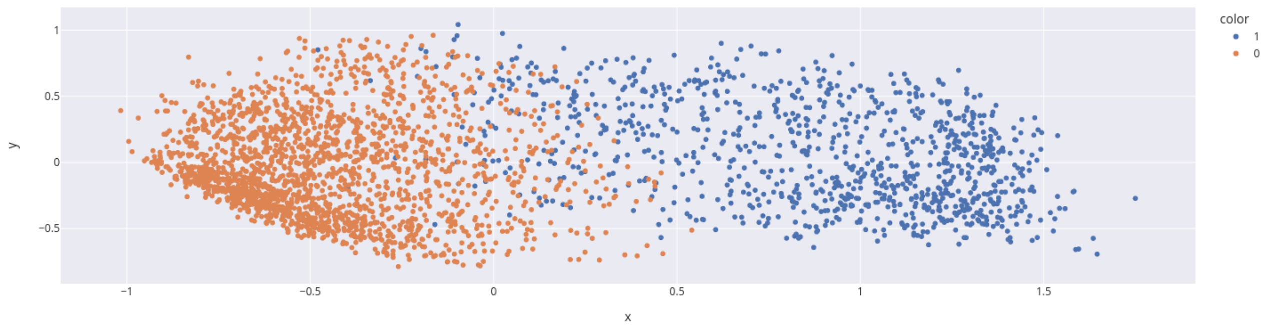


Figure 9: Principal component analysis for EMA

5.7.3 X-means

TODO check theory again
Another trail we made was with X-means. We initialised the centroids with k++ initializer and set a maximum number of clusters of 40 with the BIC algorithm performing the bisection. The results varied greatly, and often

the number of clusters was particularly high or even the maximum. So the results are not interpretable, most probably due to the fact that the distribution of the data is not globular.

6 Classification