Data Mining project report

Dalla Noce Niko, Ristori Alessandro, Giuseppe Lombardi

Master Degree in Computer science.

n.dallanoce@studenti.unipi.it, a.ristori5@studenti.unipi.it, g.lombardi11@studenti.unipi.it.

Data Mining, Academic Year: 2021/2022

Date: 10/11/2021

https://github.com/nikodallanoce/DataMiningProject

Contents

1 Data understanding and preparation

We focused our work on the matches dataset that contains tennis matches played in many tournaments across the world over the last five years, it includes both male and female players and has 186.129 records and 49 features. We decided not to do any cleaning or integration on the male and female datasets since they were only used to get the sex for each player.

1.1 Data understanding

The data understanding task focuses on analyzing the dataset and its integration by removing duplicated values, fixing missing values and solving the possible outliers.

1.1.1 Data semantics

In this phase we focused on understanding the meaning of each feature inside the dataset.

- **tourney_id**: it's a **string** that uniquely identifies each tournament. It is mainly composed by two part separated by a dash "-", where the first one indicates the year when the tournament was played and the second one refers to the tournament identifier, i.e. '2018-W-WITF-EGY-03A-2018'. The dataset contains some null value for this feature (only for the last tournament), while it has 4853 unique values.

- **tourney_name**: it's a string that represent the name of the tournament and it's not unique because the name of the tournament is usually the same over the years. It has some missing values only in the last part of the dataset.

- **surface**: it indicates the surface where the matches took place. It's a categorical attribute which its possible values are: hard, clay, grass and carpet. Only few matches are missing this feature.

- **draw_size**: it's a float that indicates the number of players in a tournament.

- **tourney_level**: it's a string indicates whether the tournament is for male or female players and also its level (mastr, ATP500, ATP1000 etc.). Has some missing values in the last part of the dataset just like for the tourney_id.

- **tourney_date**: it's the date when the match took place. In the dataset it's a float in the form YYYYMMDD. To use it as real date, a parsing to a date type is needed. Missing values are present only in the last part of the dataset.

- **match_number**: it's a float number that represents the match number in a tournament. Due to its meaning, it should be parsed to an integer.

- **winner_id**: in the dataset, it's a float representing the id of the match winner. For sure, it is unique for the player with the same gender, while the same id can appear for a male player and for a female player as well, since they play in different tours (ATP and WTA).

- **winner_entry**: it's a string that indicates how a player joined the tournament. There are a lot of missing values for this feature.

- **winner_hand**: it's a string representing the winner player's favourite hand. Its possible values are 'U' for unknown, 'R' for right and 'L' for left. Actually there are also null values, but they can be treated as unknown.

- **winner_ht and loser_ht**: float representing the winner player's height. There are a lot of missing values for this field that can not be inferred by other records where this it's filled.

- **winner_ioc and loser_ioc**: a 3-character string representing the player's nationality. Some of them have been written using different standards. For instance, a German player is present both as 'GER' and as 'DEU'. Also, some players appears with more than one nationality because they switched nationality during the last years.

- **winner_age and loser_age**: a float that indicates the age of the winner. The decimal part represents the percentage of the days days left to birthday. Some of these values are missing in the dataset, but few of them can be precisely inferred.

- **best_of**: a float indicating the number of set for a match. We parse it to an integer.

- **round**: the match round (e.g. F stands for final and SF for semifinal).

- **minutes**: indicates how much a match lasted. It's a float that we converted to an integer.

- **w_ace and l_ace**: number of aces (valid serve won, first or second, not touched by the opposing player). It's a float, but actually is an integer.

- **w_df and l_df**: number of double faults committed by the player (more specifically the number of invalid second serves). It's a float converted to int.

- **w_svpt and l_svpt**: this may be confusing, it isn't the number of points obtained after a player serve but the number of serves done by the player (the former is referred on specialised sites as serve points won).

- **w_1stIn and l_2stIn**: number of valid first serve by the player.

- **w_1stWon and l_1stWon**: number of points won after a valid first serve by the player.

- **w_2ndWon and l_2ndWon**: number of points won after a valid second serve by the player, be aware that a point lost after a second serve could be a double fault (this means that the second serve wasn't valid).

- **w_SvGms and l_SvGms**: number of games (not points) where the player was serving.

- **w_bpSaved and l_bpSaved**: number of breakpoints (the player is a point from losing a game where he's serving) saved (the player serving won the point).

- **w_bpFaced and l_bpFaced**: number of breakpoints that the player faced (see previous feature).

- **w_rank and l_rank**: the player's rank, in tennis points are awarded based on the performance at each tournament.

- **w_rank_points and l_rank_points**: the player's ranking points.

- **tourney_revenue**: the tourney revenue coming from ticket sales etc. It isn't the prize of the tournament (which would have been more interesting for a player analysis).

- **tourney_spectators**: number of spectators of the entire tournament.

1.1.2   Type casting

In this dataset, most of the features have an incorrect data type. We parse the tourney_date from float to pandas date type. Furthermore, every float attribute has been parsed to integer except for the winner/loser age and the tourney_revenue.

1.1.3   Dropping useless matches

There are many matches without any statistics, highlighted in Figure 1. Such matches are from minor tournaments and the players who played in them don't have many matches in the dataset, we decided to drop them.
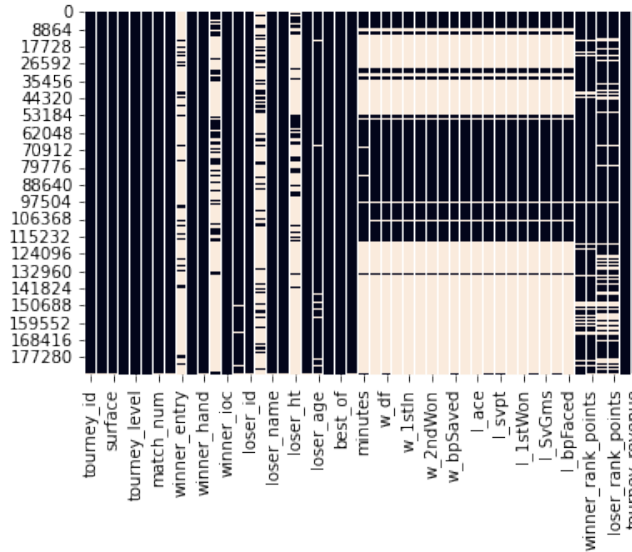

Figure 1: The missing values heatmap.

Furthermore, starting from line n. 186073 to the end, we found a block of records that is a copy of some matches of the Taipei tournament, but with some field left empty. Having noticed this, we dropped this useless part of the dataset.

5

1.1.4   Dropping useless features:

For our purpose, an analysis of the players, some features are not interesting or straight useless, so we decided to drop them. We don't need the **tourney name**, since we don't differentiate between the tournaments (masters atp, ATP1000 etc.), moreover we already have the **tourney id** and **date**.

The **draw size** is useless for the players and the **tourney level** can't be used to differentiate between men and women, since the latter can play the same kind of tournaments of the former. For what concerns **match num**, it's a progressive number for the matches of a tournament, sometimes is totally arbitrary, we don't need it.

Winner entry (and **loser entry**) is a string that shows if the player qualified by a wildcard, as a lucky loser etc., it's not a very useful attribute since no site lists them, moreover is the feature with the most missing values.

The last features we dropped are **tourney revenue** and **tourney spectators**, the former is the amount of revenue generated by the tournament and not the winner prize which would have been really interesting. The latter is the amount of spectators of the entire tournament, not match to match, so we can't retrieve information about how famous the players involved in a certain match are or if they can play better under more pressure.

1.1.5   Dropping duplicates

We found that the matches dataset has 302 duplicated records. Due to the domain of this dataset, these duplicates are useless, so we drop them.

1.1.6   Data integration

The data integration task consists in solving the various issues found in the dataset by filling the missing values, where it's possible, and fixing the various problems that may arise (e.g.: players with two hands preferred or with more than one ioc).

Player id and name   In our analysis, we found that there are three players ('Guy Stokman', 'Giuseppe Tresca', 'Kuan Yi Lee') that appear in the dataset with more than one id associated. Moreover, there's only one player with an actual homonym (Kuan Yi Lee), one is a male (id: 134120) and the other one is female (id: 221745), but we discovered that the actual name of the male player is Kuan-Yi, so we change his name to differentiate between the two players. For what concerns the other two players, we assign them their last id.

Deepening the analysis, we found that ids are shared only between a male and female player and this is allowed since different tours exist (ATP and WTA) for each sex, so we don't need to apply any change, moreover we won't work with ids so this won't affect our analysis.

Surface   We found that the "surface" feature has some null values. Moreover, the number of tournaments without a surface are 42 while the number of matches without a surface are 117. In order to solve this problem, for those tournaments that lack the surface we can retrieve it if at least one match of the same tournament has it. Unluckily there are no tournaments for which we can retrieve the surface (all of them are from the Davis or Federation cup, where the

surface changes from event to event), so we chose to sample such values from the distribution of the surfaces.

Winner/loser hand We have to manage the missing values for both "winner_hand" and "loser_hand". To do this, we checked if the value is missing only for a particular player or if he/she never has the hand defined and, in this case, we assign it to him/her. After this step, however, the missing values are still present, but since the domain of this feature is 'U', 'L', 'R', we can safely treat null values as 'U'. The players with unknown hand are 1840. Among them, we retrieved the actual preferred hand for one player, Amina Anshba who is left-handed.

Winner/loser height This feature contains missing values too, so we address this problem as before, looking for those players that have a known height somewhere in the dataset. A deeper analysis show us that 'David Goffin' has two heights registered: 180 cm and 163 cm. Since his real height is the first one, we fixed it in in all occurrences in which he appears.

Winner/loser ioc Since there are no missing values for this feature, we only need to check if there are players with more than one ioc. Since we dropped records and players that have played too few matches, we haven't found players with more than one ioc. In the original dataset instead, it happens. Anyway, we found that player called 'Xinmu Zhou', has 'UNK' nationality, but he is Chinese actually, so we assign him 'CHN'. Lastly, we discovered that sometimes, the acronyms of the nations are wrong, since some of them don't follow the international standard (e.g.: GRE for Greece instead of GRC, or TPE for Taiwan instead of TWN) or have both at the same time (e.g.: DEU and GER for Germany), so we fixed them, at least the ones we found.

Winner/loser age There are 58 players with unknown age. Among them there are only two players whose age can be inferred from the other information inside the dataset, we fix the missing ages during the data preparation task.

Winner/loser rank This information is missing for those players that played only few matches. We fixed the missing values with the last tournament rank for each player (or the next if not present) that already had a rank. For those without rank we assumed they didn't have any points so we assigned them the maximum rank in found in the dataset.

Winner/loser rank points Same work as for the rank feature, instead this time we assign the minimum value for those players without any rank points (they are the same we found during the rank integration).

1.1.7 Outlier detection

Given that all the feature distribution are Gaussian or half normal, to analyze outliers we must first compute the first quartile, the third quartile, the median and the interquartile for each feature. Then we compute the lower bound $L = Q1 - 1.5 * IQR$ and the upper bound $U = Q1 + 1.5 * IQR$ for each feature. In case some numerical data was less or greater than the

lower or upper bound respectively, we can identify an outlier. For half-normal distribution we don't consider the lower bound. In case of an outlier that can be fixed with a real value, we do it. In this way we found, for example, two players that have too low height and, in fact, these data were wrong. We fixed it with the players' real height. Whereas high values correspond to those players that are actually tall. We have applied this analysis for every numerical feature.

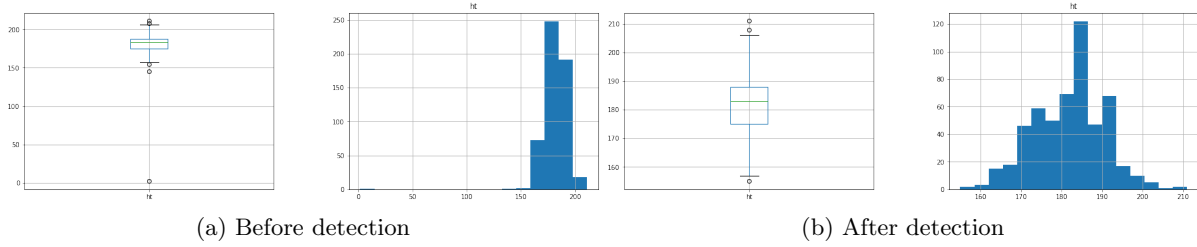(a) Before detection   (b) After detection

Figure 2: An example of a feature's distribution after dealing with outliers.

1.1.8 Correlations

We plot the correlation matrix in order to visualize whether there are correlation between the features. In Figure 3, the more the color is red the more the features are correlated.
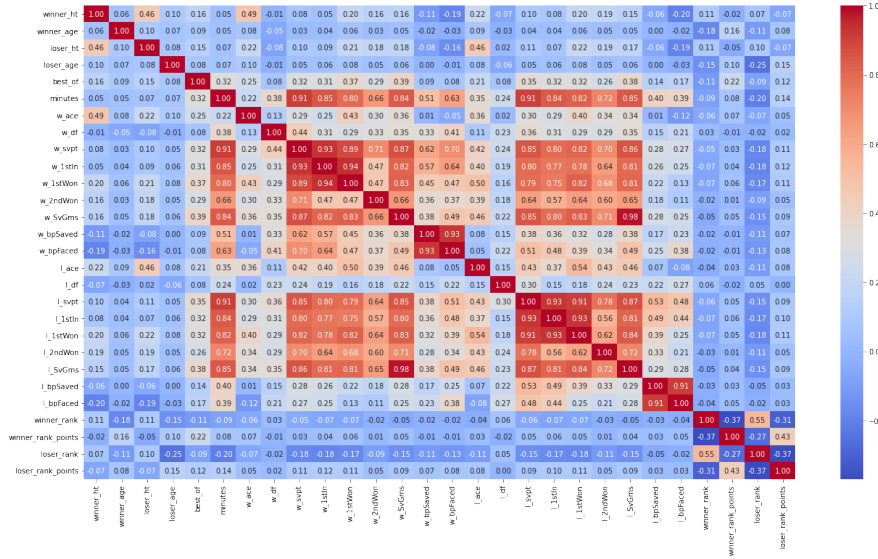

Figure 3: Correlation matrix of the tennis matches dataset.

After looking at the correlation matrix we dropped the minutes feature since we deemed it as useless after doing a huge help during the outlier detection by showing "inusual matches".

1.2   Data preparation

In this section we focus on describing our new dataset, cleaned and composed by some new features derived by the original ones.

1.2.1   Building the player's profile

The purpose of data preparation is building a profile that will feed the clustering analysis, as a starting point we decided to build such profile from the features we had from the matches dataset.

Sex   First of all, when we started building our new dataset, we assigned the sex to each player inside tennis_matches using the female_players and male_players datasets. Some players of the tennis_matches dataset were not found in the "sex" dataset due to spelling errors in their name, but we solved this problem by looking online.

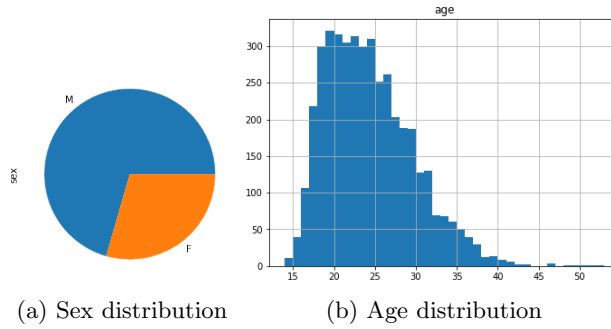(a) Sex distribution   (b) Age distribution

Figure 4: Sex and age distribution

Age   Since we are doing a "current" analysis, we assign to each one of the players the last age they appear in the original dataset. Some of them have an unknown one, we had to sample it.

Ioc   This was the easiest one since we didn't have to deal with missing values.
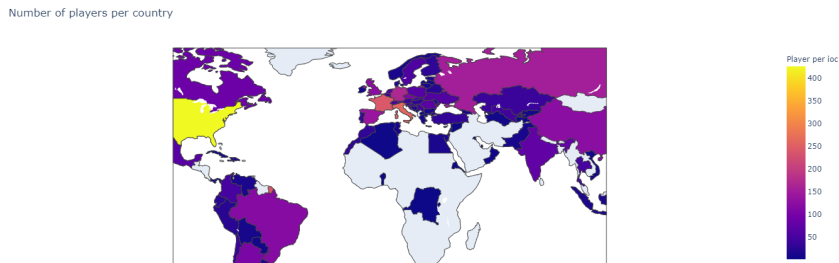

Figure 5: IOC distribution

Height We solved the issue of having many players without an height by sampling their height from the distributions based on countries and sex where this is possible, otherwise we assign them the mean height of their respective sex.

Hand Just like for the ioc we just needed to assign the hand to the players since the missing values were already dealt with.

Wins and losses We calculated the total matches won or lost by the player and also how many matches they won on a specific surface.

Tournaments won We calculated the tournaments won by the player by looking at how many times he appeared in a final as the winner.

Surfaces We inserted the amount of wins by a player for each surface he/she played on.

Statistics, rank and rank points For each player we calculated all the statistics coming from the cleaned matches dataset, futhermore we assigned at each player their rank and rank points, eliminating from the dataframe those played less than ten matches.

1.2.2 Building new features

Having inserted the feature coming from the cleaned matches dataset it was time to look at the correlation between the features. We saw there were many highly-correlated feature, so we decided to build new ones from those that we had in the dataframe, such as [*num_matches, p_wins, p_w_Hard, p_w_Clay, p_w_Grass, mean_ace and p_aces, mean_double_faults and p_double_faults, mean_1st_in and p_1st_in, mean_1st_won and p_1st_won, mean_2nd_won and p_2nd_won, mean_bp_saved and p_bp_saved, mean_bp_faced, mean_sv_games, mean_sv_points*]

Categorical features We built categorical attributes that split players by age, height and rank ranges.

We then dropped the features that were too high correlated with another one or we deemed useless for the players analysis (n_matches, mean_aces, mean_sv_games, mean_double_faults). After adding the new derived features, we checked that they were not correlated each other by calculating again the correlation matrix, as showed in Figure 6.
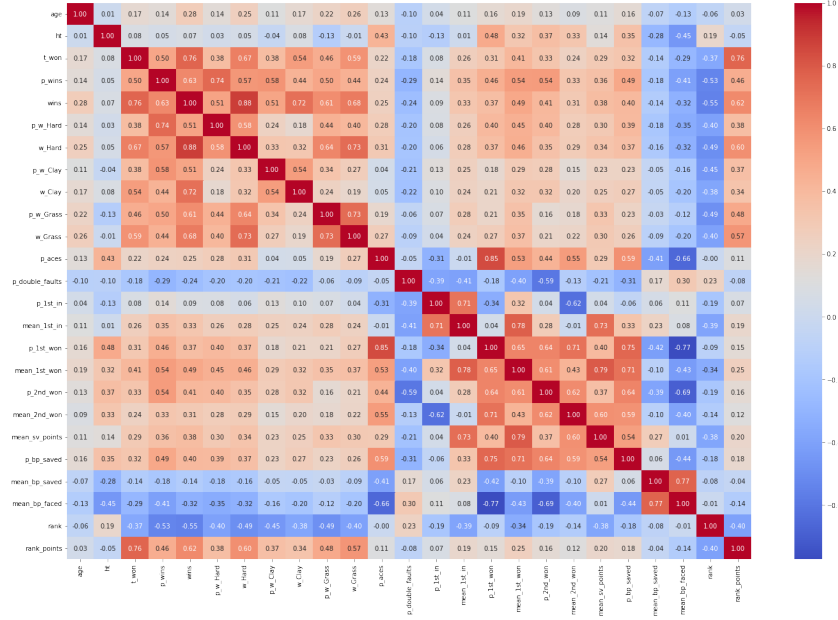
Figure 6: Correlation matrix of the new dataset with the features we added.

2 Clustering analysis

2.1 Preprocessing

In order to let the cluster algorithms work, we had to preprocess the data. Initially we tried to use the MinMax scaler, but we saw that K-Means was not working well. Due to that, we decided to normalize such features with their z-score by using the scikit-learn StandardScaler method.

2.2 K-Means

2.2.1 Feature selection

In order to apply K-Means, we selected only some features to avoid the course of dimensionality. **We selected the attributes that best differentiate strong players from weak ones**. We have also tried the PCA (principal components' analysis) approach to understand the most relevant features, but the results, obtained by the K-Means on these features, were not satisfying in terms of silhouette score. After several experiments, we obtained the best result choosing as parameters the number of tourney won (*t_won*), the percentage of wins (*p_wins*) and the rank (*rank*) of each player. We also tried different set of features (percentage of wins on each surface and statics) which we also explored with the k-means algorithm, but the explanation of the results it's not easy for someone who isn't fond of tennis, so we decided to work mainly with the three features listed before.

2.2.2 Choosing the best K

The choice of the parameter k in the k-means approach is crucial since it identifies the number of clusters resulting from the algorithm's execution, but before choosing the best k we let the algorithm work for twenty times and then we analyzed the indicators in Figure 8. We choose $k = 3$ by following the informal elbow rule for the SSE graph in Figure 7, moreover we wanted an high value for the silhoutte score and a low one for the Davies-Bouldin score as we can see for $k = 3$ on Figure 8.
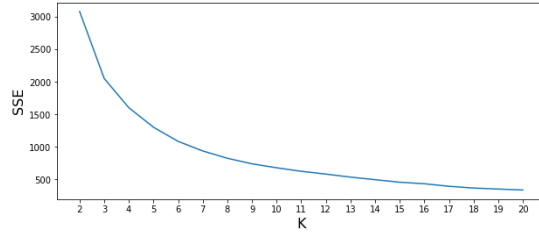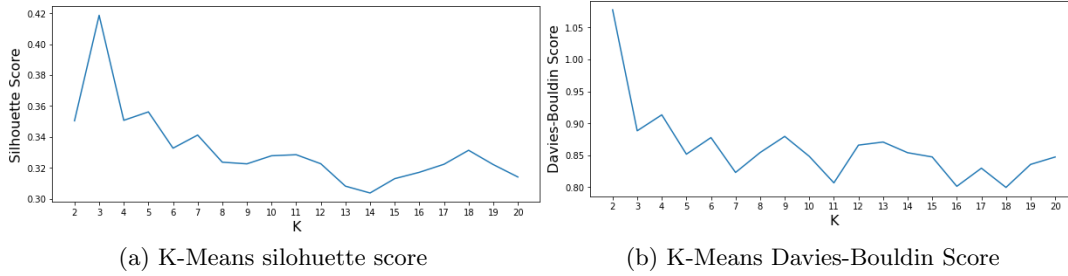

Figure 7: K-Means SSE over K clusters.

(a) K-Means silohuette score

(b) K-Means Davies-Bouldin Score

Figure 8: K-Means metrics over k clusters.

2.2.3 Cluster analysis

We applied the PCA on the previously chosen features to analyze the K-means results, then we showed the scatterplot of the first two principal components. As we can see in Figure 9, the players are divided from left to right by their skills: the weak ones, the average ones and the strong ones. The points on the top right correspond to the top tennis players, such as Djokovic, Nadal and Zverev; we have highlighted Novak Djokovic, the player with rank 1, in the plot to better show the distribution of the players by strength. We have noticed that some players with a high rank are in cluster 0, this is probably due to the fact that they have a high win rate or have won some minor tournaments.
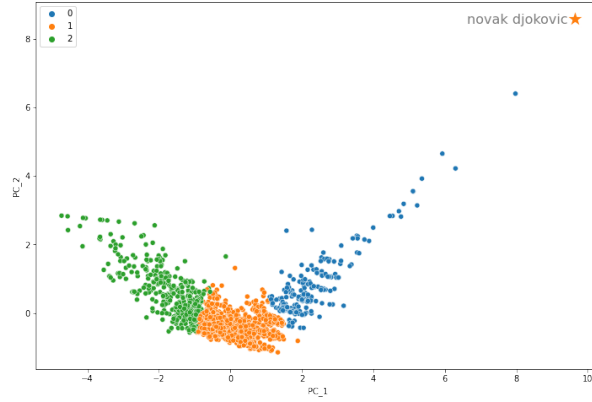
Figure 9: The 3 clusters obtained using K-Means.
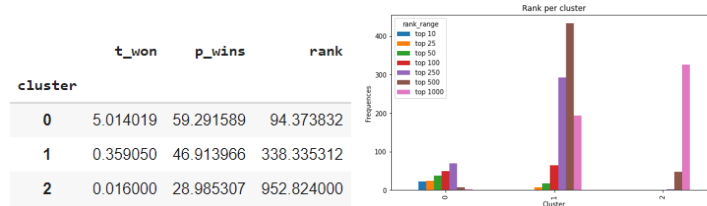
Figure 10: Statistics of the feature for each cluster.

2.3 DBScan

We decided to work with the dataframe used in the k-means analysis (the one with "t_won", "p_wins", "rank"), since we think that it better describes who are the best players and those who aren't.

2.3.1 Determining eps and min points

The distance between data points is computed using the Euclidean metric, and for selecting the best eps and min_samples values we first performed a grid search whose results can be found on the notebook. From the results we choose $min\_samples = 8$ and by looking at Figure 11 we choose $eps = 0.55$ since it's the value at which the graph's curvature is the highest. The silhouette score is equal to 0.6208
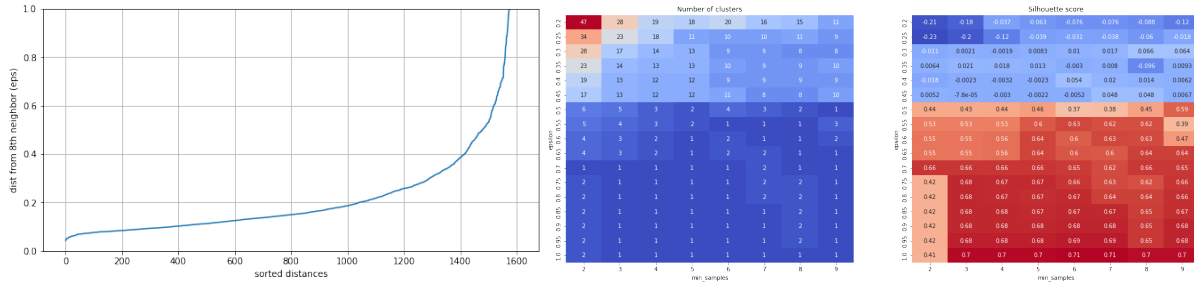
Figure 11: DBScan: determining eps and min_samples

2.3.2 Cluster analysis

Analyzing the clusters obtained by the DBScan method (Figure 12), we've seen that there is a single large cluster (the orange one) and the blue points referring to the outliers. In the outliers' cluster there are the players at the extremes: the players who have won very few matches and the strongest players in the world, respectively on the left and right side of the main clsuter. There are also outliers in the center of the plot, representing players who have won some minor tournaments.
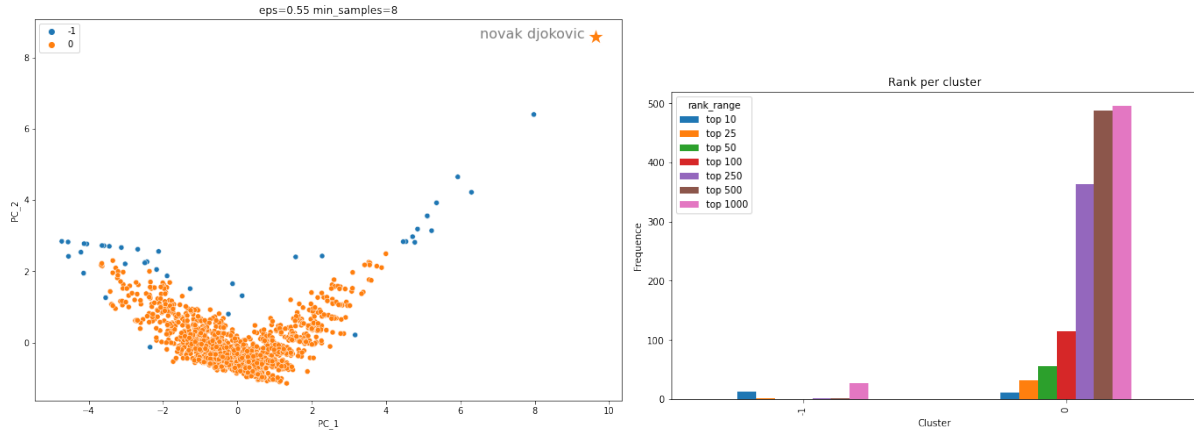
(a) Clusters generated by applying DBScan, plotted along the 2 P.C.

(b) Distribution of players' ranks per cluster

Figure 12: DBScan: cluster analysis.

2.4 Hierachical Clustering

The analysis of the clustering via hierarchical clustering has been conducted using the same set of attributes of the previous algorithms, in order to get results comparable in terms of indicators and properties among the approaches.

2.4.1 Distance methods

Like we did with the other clustering algorithms, we used the Euclidean metric to compute the distance between the pairs of points. We have tried several type of agglomerative clustering that differ by the merging criterion of the clusters like min, max, average and ward. In the hierarchical clustering approach, we have not assumed any particular number of clusters: any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level.

2.4.2 Comparing dendrograms

Cut threshold The "best cut" of the dendrograms, is a cut that passes from the longest vertical segment not interrupted by horizontal lines. Anyway we avoided to choose a cut that would have produced too unbalanced clusters.

Method	cluster id : its dimension	Silhouette
Complete	0: 759, 1: 12, 2: 827, 3: 2	0.3082
Single	0: 1598, 1: 1, 2: 1	0.8013
Average	0: 1580, 1: 2, 2: 18	0.6485
Ward	0: 330, 1: 296, 2: 974	0.3756

Even if the silhouette is pretty good using the single and average methods, we can not say the same thing looking at the dendrograms, since the clusters are unbalanced, and as a consequence, most of the points falls in one big cluster. The only exceptions in this behaviour is produced by applying Ward method, that produced an almost balanced clusters closer to the K-Means ones, as shown in Figure 14.
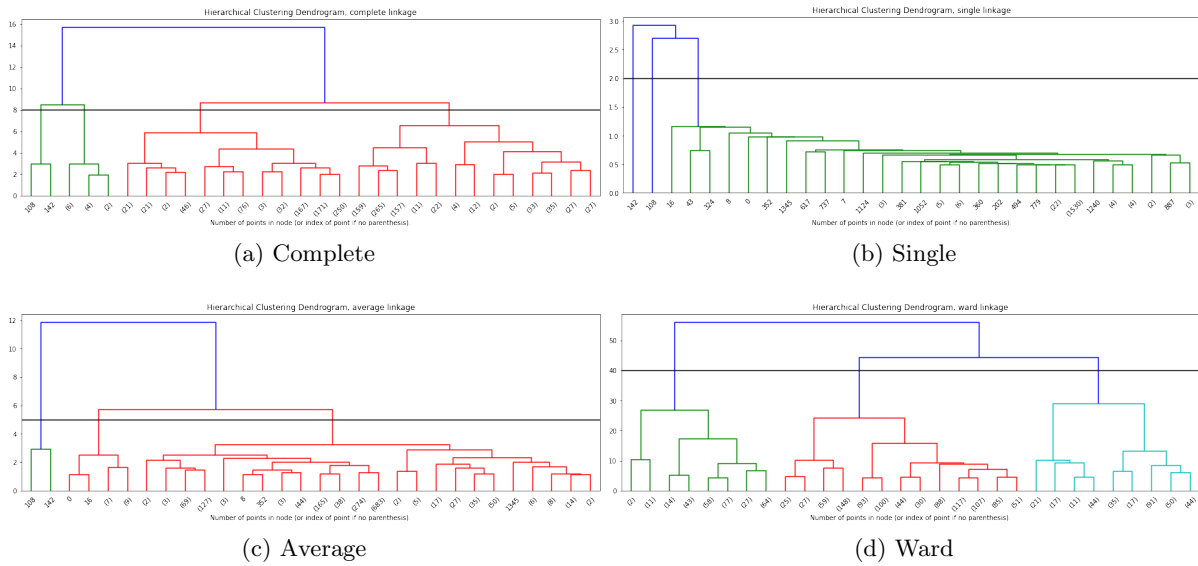

(a) Complete

(b) Single

(c) Average

(d) Ward

Figure 13: Hierarchical clustering methods comparison: dendrograms.

15

2.4.3 Comparing clusters

By looking at the pcas plot from Figure 14, we can define the hierarchical clusterings with complete and ward linkage as the best ones among the four methods. The single linkage was the worst one since it created two cluster with one player each (Djokivic and Nadal which are the best overall player), meanwhile the average one separated the extremely strong players (again Djokivic and Nadal) from the strong ones (green cluster) and the average or weak ones (blue cluster). The only approach that didn't have such drastic separation were the complete and ward linkages, which is why we considered them as the best among foru approaches.

(a) Complete

(b) Single

(c) Average

(d) Ward
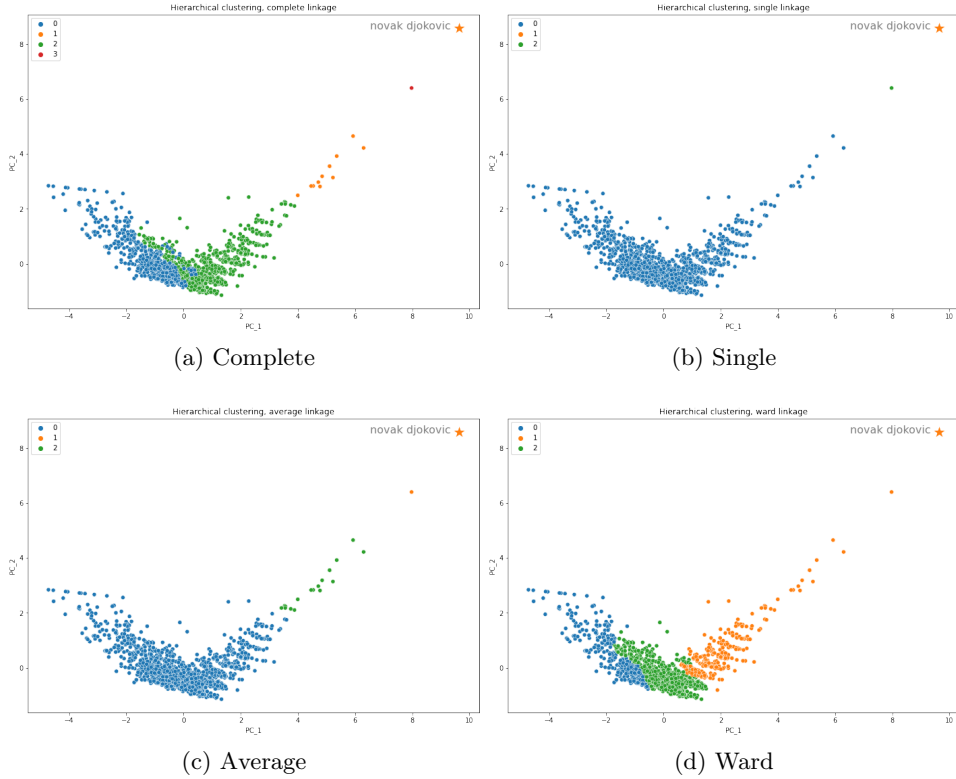
Figure 14: Hierarchical clustering: plotting along the PCAs.

2.5 Validation by external measures

We computed the *similarity matrix* S given a particular clustering. To obtain this matrix, the indices are sorted according to the labels of the clusters and the component $S(i,j)$ is equal to $e^{-d(i,j)}$, where $d(i,j)$ is the euclidean distance between the points i and j. If we have well-separated clusters, then the similarity matrix should be roughly block-diagonal. We also computed the entropy of each feature per cluster: a low entropy score of each attribute indicates a more predictable and less uncertain trend within the clusters.

(a) Similarity Kmeans

	t_won	p_wins	rank	total	cluster
0	1.22241	5.96804	6.4523	13.642740	0
1	1.24287	5.02835	5.16675	11.437966	1
2	0.926832	5.31888	5.58171	11.827420	2

(b) Entropy Kmeans

(c) Similarity
Complete

	t_won	p_wins	rank	total	cluster
0	1.29796	5.92966	6.27411	13.501734	0
1	1.09861	2.48491	2.48491	6.068426	1
2	1.04161	5.75331	6.17364	12.968556	2
3	0	0.693147	0.693147	1.386294	3

(d) Entropy Complete

(e) Similarity
Ward

	t_won	p_wins	rank	total	cluster
0	1.40096	5.47581	5.62733	12.504099	0
1	1.293	5.29267	5.54986	12.135523	1
2	1.03742	5.83546	6.29367	13.166547	2

(f) Entropy Ward

	t_won	p_wins	rank	total	cluster
0	1.17746	6.17805	6.57116	13.926668	0
1	0	0	0	0.000000	1
2	0	0	0	0.000000	2

(g) Entropy Single

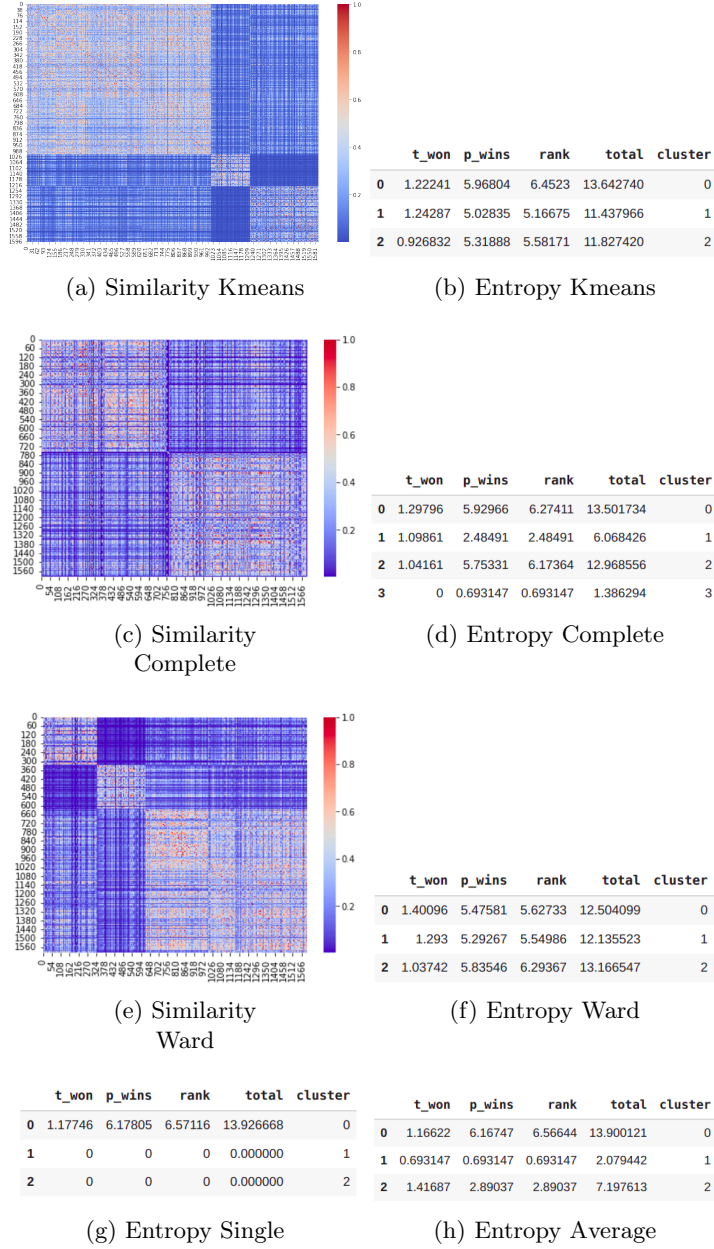|	t_won	p_wins	rank	total	cluster
0	1.16622	6.16747	6.56644	13.900121	0
1	0.693147	0.693147	0.693147	2.079442	1
2	1.41687	2.89037	2.89037	7.197613	2

(h) Entropy Average

Figure 15: Similarity matrices and Entropy of attributes per cluster for each method

2.6   Other Clustering approaches explored

For the final work on clustering we went to explore two more approaches from the pyclustering package as suggested by the professor, our choice fell on X-Means and SOM soft-clustering.

2.6.1 X-Means

From the pyclustering package documentation: X-means clustering method starts with the assumption of having a minimum number of clusters, and then dynamically increases them. X-means uses specified splitting criterion to control the process of splitting clusters.

This approach is different from the previous ones we tried, since it doesn't need to know beforehand the number of clusters, but it starts with a pre-determined amount and then splits them into smaller ones. The results change from one try to another, but we've seen that this approach usually worked better for our dataframe.

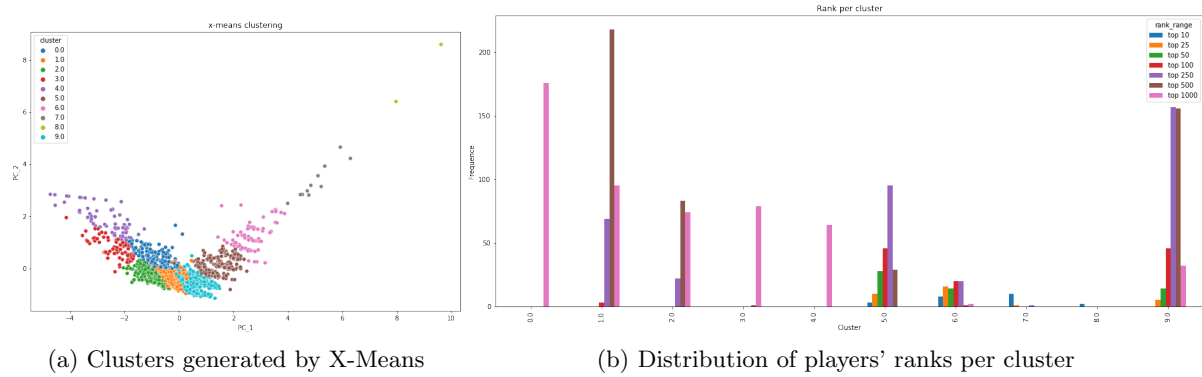(a) Clusters generated by X-Means   (b) Distribution of players' ranks per cluster

Figure 16: X-Means: pca and analysis.

By looking at Figure 16 we can see that X-Means separated better the players by their skills than our previous attempts, so we think that it did the best job among everything we tried.

2.6.2 SOM Soft-Clustering

From the pyclustering package documentation: this algorithm uses an amount of clusters that should be allocated as the size of a SOM map. The captured objects by the neurons are clusters. We expected the pyclustering library to provide us a way to plot the SOM lattice but this wasn't the case, so we went, as always, with the pca just like the other approaches. For the choice of k we did the same work as K-Means and we found that the best choice was for $k = 3$ but we couldn't calculate in any way the SSE since the implementation by the library doesn't come with a way to obtain the cluster centroids, so we only considered the Silhoutte and Davies-Bouldin scores.

(a) Clusters generated by SOMsc
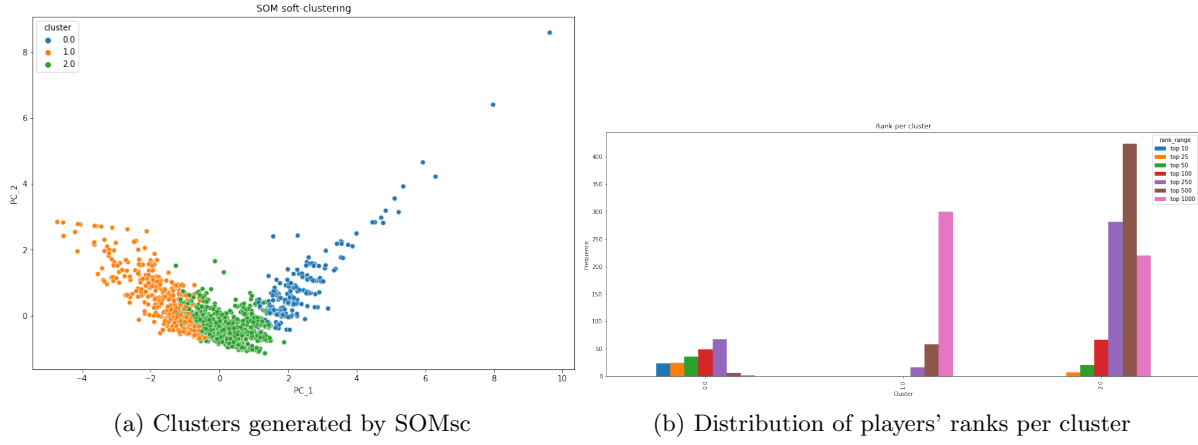
(b) Distribution of players' ranks per cluster

Figure 17: SOM soft-clusering: plot along PCAs and analysis.

The results obtained, which can be seen on Figure 17, are similar to the ones obtained by k-means, so we think we don't need to explain anything more about this approach.