Analysis of Tennis Matches Dataset

Data Mining Project

Jacopo Bandoni
Alex Colucci
Domenico Romano

Dipartimento di Informatica
Università di Pisa
05/01/2022

Contents

1 Introduction

In this report, we address several data mining tasks. We start with the analysis of a dataset containing tennis matches by doing data understanding and feature engineering with the aim of applying a clustering analysis to identify interesting patterns among the players' profiles. Then we perform a prediction analysis with the aim of discriminating strong players from weak ones. Finally, there is a time series analysis on a dataset of the temperatures of some world cities.

2 Data understanding

2.1 Dataset overview

The `tennis_matches` dataset contains **186128** total observations and in the following image 2.1 we can see a summary of the null values for each attribute.

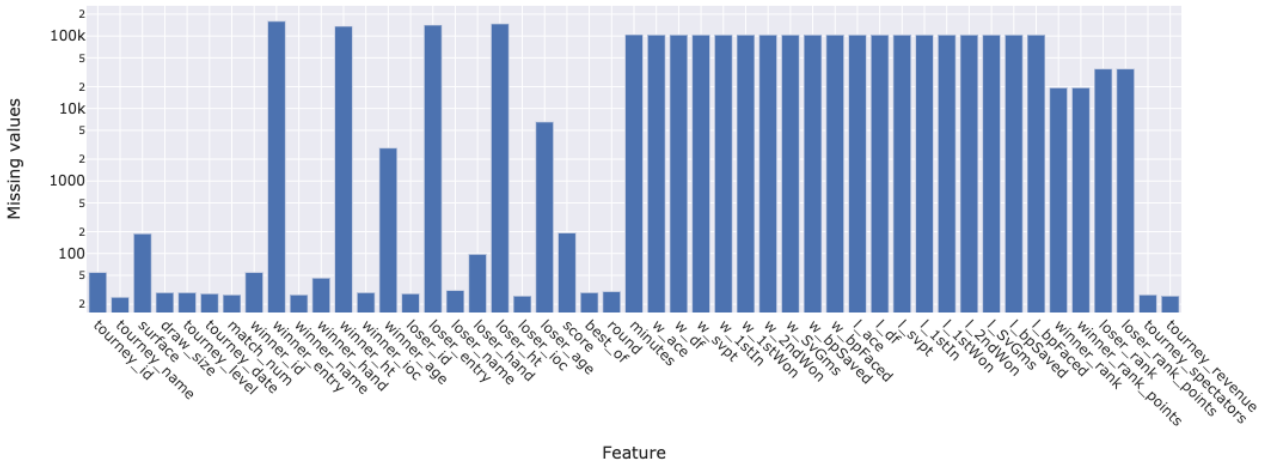


Figure 1: The number of null values for each attribute

2.1.1 Feature understanding

Below, the semantic analysis for each of the **48** attributes:

tourney_id The identifier of the tourney. There are 4854 unique tourneys.

tourney_name The name of the tourney. There are 2488 unique names. 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. 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. There are 19 unique values. 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. 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.

winner_id, loser_id The player_id used in this dataset for the winner/loser of the match.

winner_entry, loser_entry is an acronym that indicates how a player is qualified in a tournament. For example:

- `Q` (Qualifier): player who reaches the tournament's main draw by competing in a pre-tournament qualifying competition instead of automatically qualified by virtue of their world ranking, being a wild card, or other exemption.
- `WC` (Wild Card): player allowed to play in a tournament, even if their rank is not adequate or they do not register in time. Typically, a few places in the draw are reserved for wild cards, which may be for local players who do not gain direct acceptance or for players who are just outside the ranking required to gain direct acceptance. Wild cards may also be given to players whose ranking has dropped due to a long-term injury.
- `LL` (Lucky Loser): player or team that gains acceptance into the main draw of a tournament when a main draw player or team withdraws.

and many others.

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

winner_name, loser_name Winner's/loser's name.

winner_ht, loser_ht Winner's/loser's height in centimetres

winner_ioc, loser_ioc Winner's/loser's three-character country code.

winner_age, loser_age Winner's/loser's age, in years, depending on the date of the tournament.

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. 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.

minutes The duration of the match.

w_ace, l_ace Winner’s/loser’s number of aces.

w_df, l_df Winner’s/loser’s number of double faults.

w_svpt, l_svpt Winner’s/loser’s number of serve points.

w_1stIn, l_1stIn Winner’s/loser’s number of first serves made.

w_1stWon, l_1stWon Winner’s/loser’s number of first-serve points won.

w_2ndWon, l_2ndWon Winner’s/loser’s number of second-serve points won.

w_SvGms, l_SvGms Winner’s/loser’s number of serve games.

w_bpSaved, l_bpSaved Winner’s/loser’s number of breakpoints saved.

w_bpFaced, l_bpFaced Winner’s/loser’s number of breakpoints faced.

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.

winner_rank_points, loser_rank_points Number of ranking points.

tourney_spectators The number of total spectators of the tourney.

tourney_revenue The total tournament earnings.

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 leading, trailing and 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.

loser_ht Several outliers with loser_ht = 2.0 2, even in this case we decide to search on the internet the correct values and replace those with the outliers.

winner_ht Several outliers with winner_ht = 2.0 and also other outliers with height < 146 3. 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 4. Then we decide to replace those values with the medium age of the corresponding players.

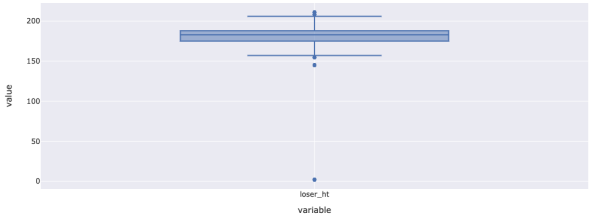


Figure 2: loser_ht boxplot

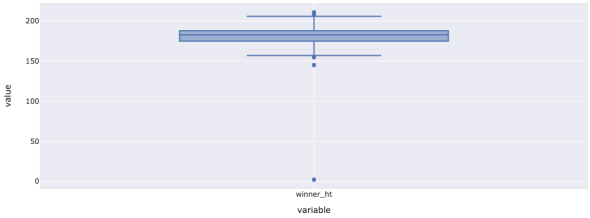


Figure 3: winner_ht boxplot

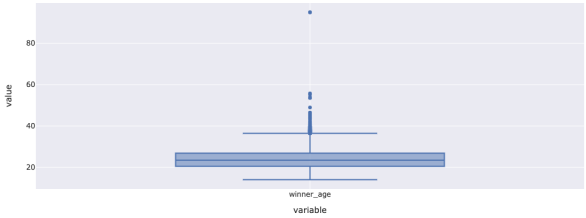


Figure 4: winner_age boxplot

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.

4 Players dataset

In this chapter we will create a new **player dataset** where we will define new features interesting for describing the player profile and his behaviour derivable from matches.

4.1 Feature engineering

gender The sex of the player. We obtained it performing a join with the datasets. For each player where the gender was missing, we replaced that gender by checking in all his game the most frequent gender's opponent.

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).

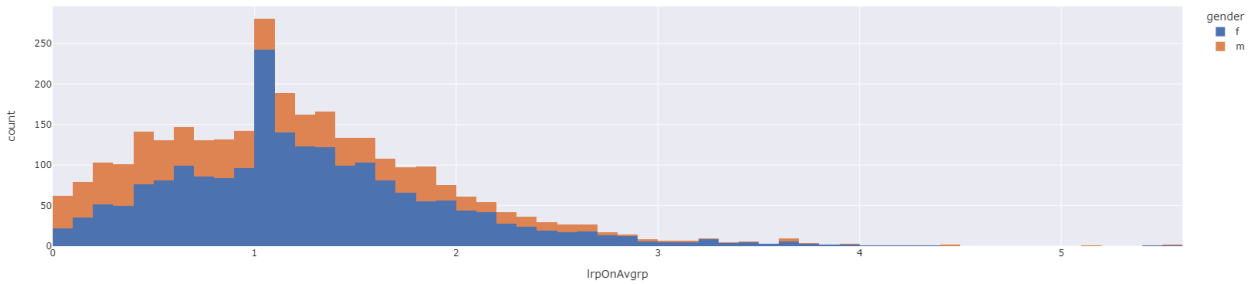


Figure 5: lrpOnAvgrp Histogram

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.

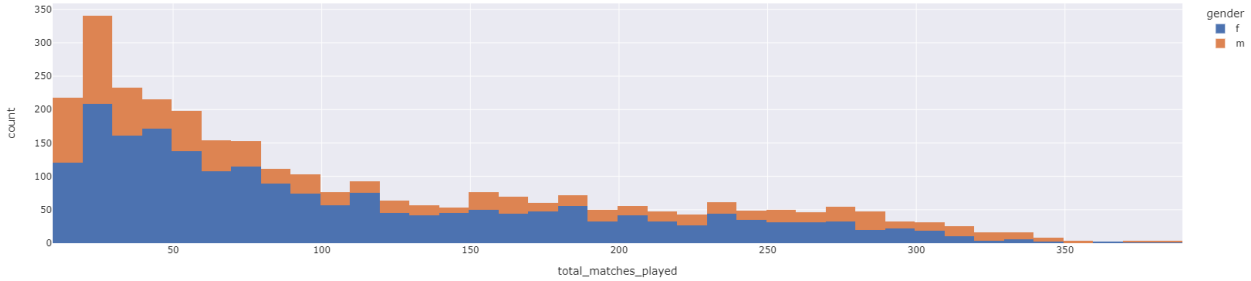


Figure 6: total_matches_played Histogram

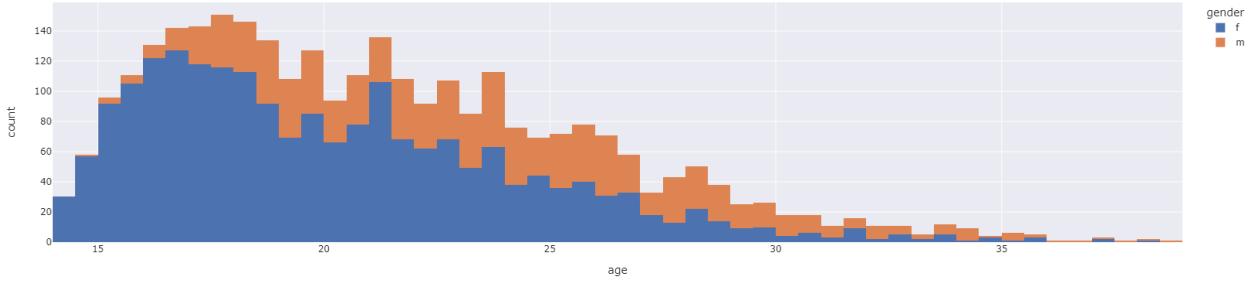


Figure 7: age Histogram

4.2 Players dataset cleaning

- We dropped all the records where *lrpOnMxrp* (and therefore the other features regarding the ranking points) had value null and we performed a mean imputation on the *age* (4 missing values).
- We removed all the players who have played less than 15 matches for the clustering or less than 4 in the case of classification.

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:

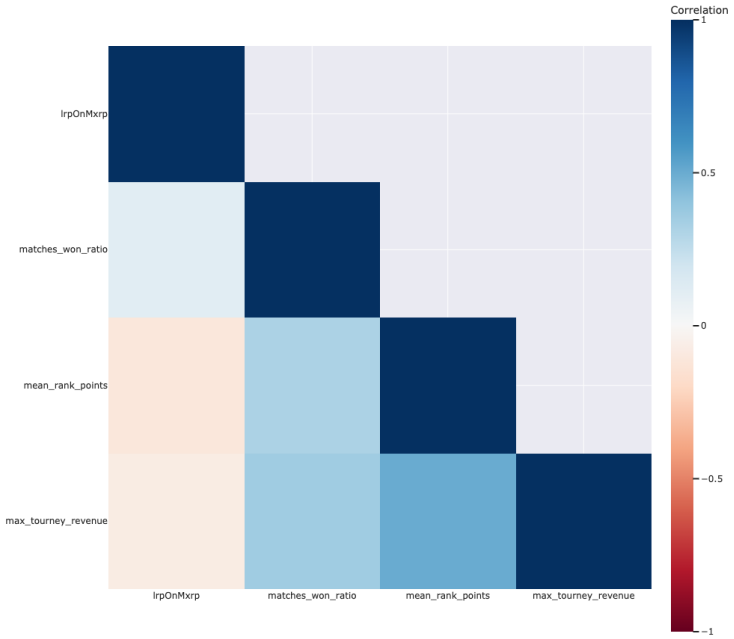


Figure 8: Correlation Plot

At this point the remaining features were: '*lrpOnMxrp*', '*matches_won_ratio*', '*mean_rank_points*', '*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, we performed a normalization of the data with MinMaxScaler in order to assign the same weight to each clustered feature. The 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. 9). On the other hand, even the Silhouette score had the maximum

value with $k = 4$ (Fig. 10), hence it was trivial to decide that the optimal k overall was $k = 4$. Moreover, the Silhouette score per cluster is represented in Fig. 11 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.

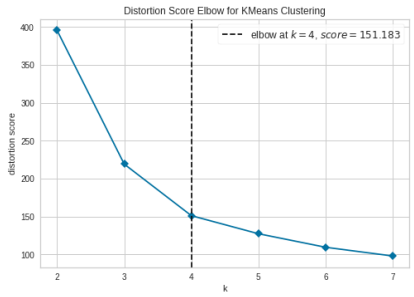


Figure 9: Elbow rule

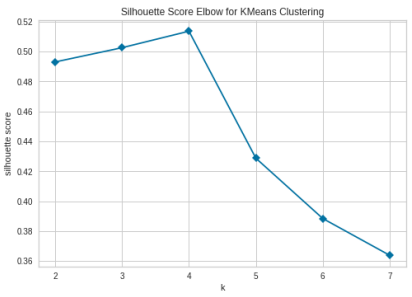


Figure 10: Silhouette score

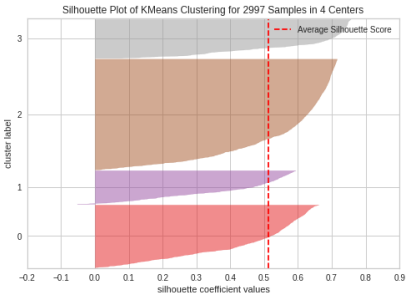
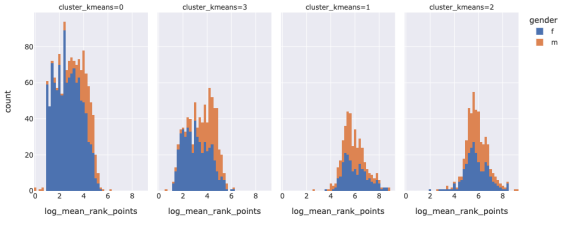


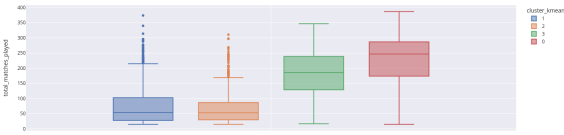
Figure 11: Silhouette score average

5.3.1 Results interpretation

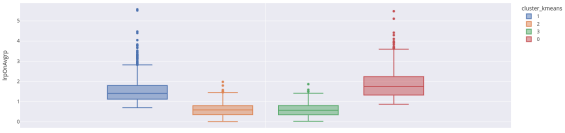
(a) histogram of age for male/female



(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 12: Distributions of features within the dataset

The clustering algorithm manages to find interesting groupings. Analysing Figure 12 (a), it can be seen that cluster 0 has a lower average age and cluster 1 a higher one. On the other hand, in Figure 12 (b) 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 12 (c), 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 12 (d), 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. 14) 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.

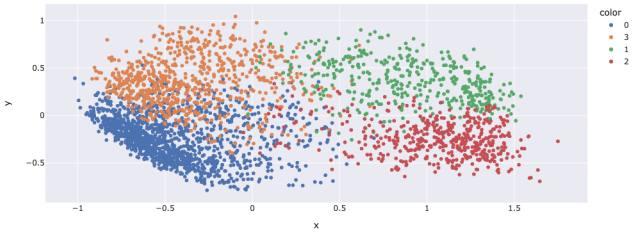


Figure 13: PCA visualization

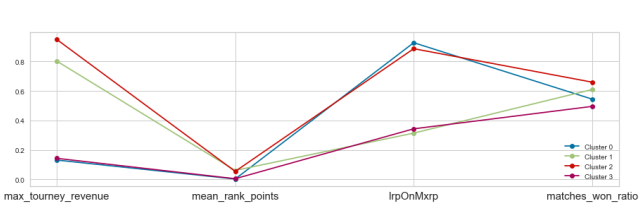


Figure 14: Plotting k-means centroids

In Figure 13 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 15 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.

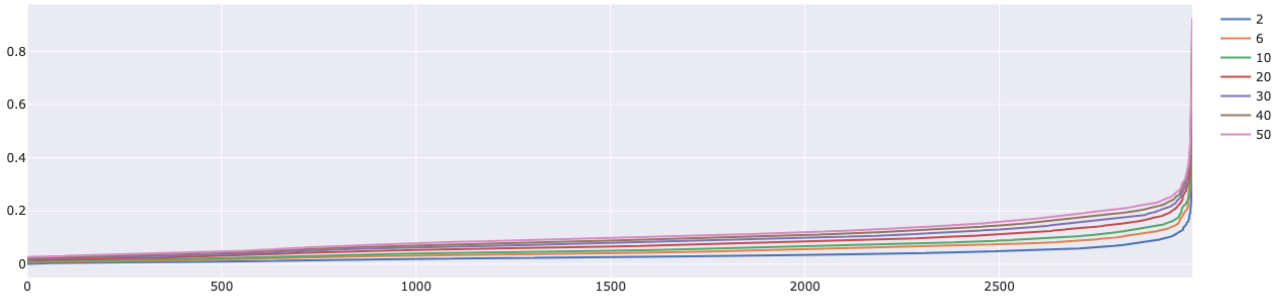


Figure 15: Noise points with the k-th nearest neighbor at farther distance

Then, in order to find the proper parameters, we run a **grid-search** (Fig. 16) 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.

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 17, where the PCA visualization is shown, while in Figure 18 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:

- **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.

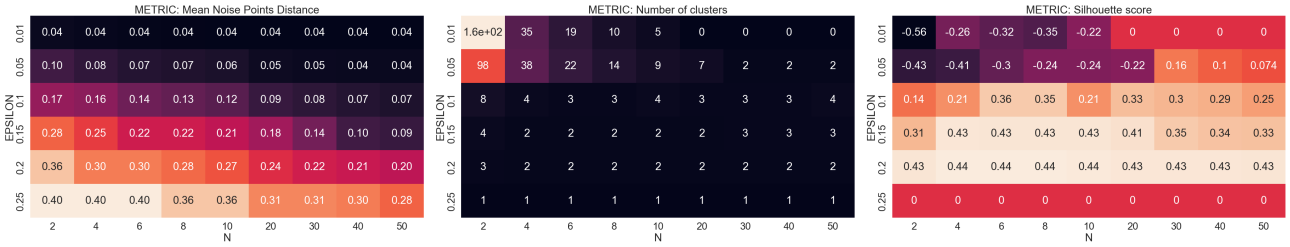


Figure 16: Chosen hyper-parameters $eps = 0.2$ and $n = 6$

Figure 17: PCA visualization

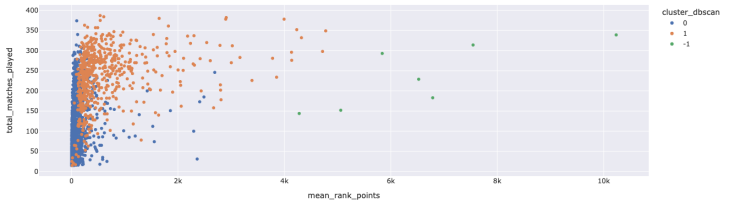


Figure 18: Results of DBSCAN

- **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 $0,71$.

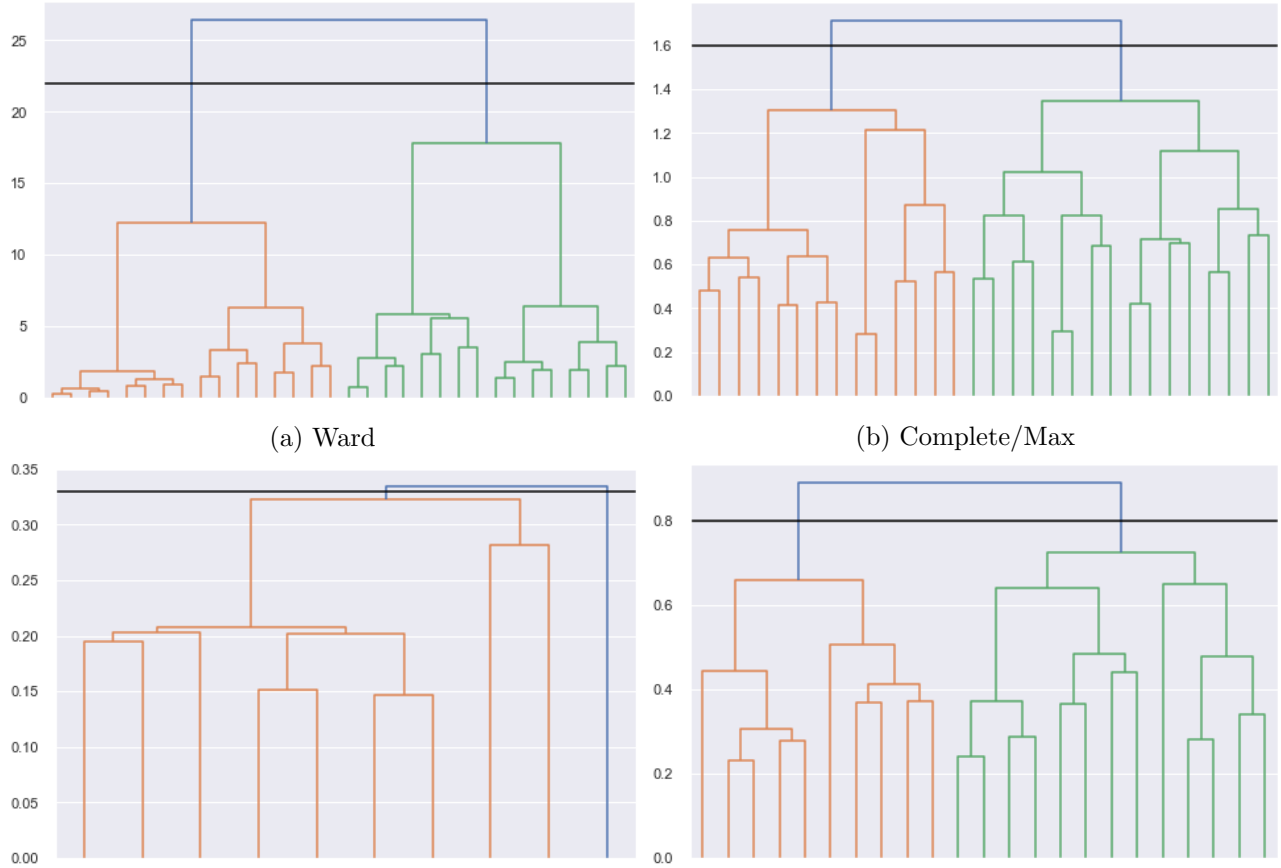
5.5 Hierarchical

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 19, 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 19: 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 20. 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 20: Principal component analysis for EMA

5.7.3 X-means

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.

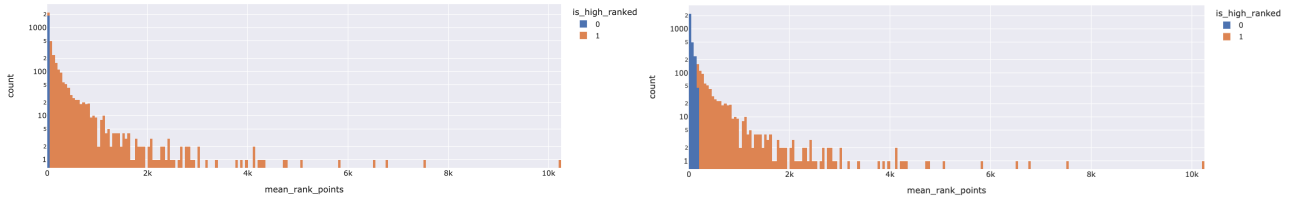
6 Classification

The objective of the classification task was to identify strong players and weak players. In order to do that, the starting dataset used is the Players Dataset already defined. The goal here was to have more data. Hence, to consider more players, we set a lower threshold on the number of matches played (increasing the entries from 2997 to 3798).

6.1 Pre-processing

Label extraction The dataset is not provided with a label for this kind of classification task, so a homemade one was created taking into account the `mean_rank_points` feature. First of all, it is necessary to discriminate between strong and weak players. To do this, there are several options, we have opted for:

- **median**, this ensures to obtain balanced classes, hence 1899 for each class
- **mean**, with the drawback of obtaining unbalanced classes: 3009 weak, 789 strong. This reminds us of the pattern identifies during clustering analysis, where 80% of the players are weak and 20% are strong. Of course, this better captures the inherent competitiveness of the game of tennis.



(a) Median

(b) Mean

Figure 22: Different options to split (log scale on y).

Feature selection The next issue to address are the features to use for the classification purpose. We have only considered the numerical features, thus losing the information about the gender and the hand of the player. Then we dropped those features with a big amount of missing values such as the in-match statistics with the aim of maximizing the number of players in the dataset. Lastly, we dropped the derived features from rank points, such as *max_rank_points*, *last_rank_points*, *variance_rank_points*. And of course in the end, *mean_rank_points* was also deleted.

Normalization For all the algorithms, we applied a MinMaxScaler but the decision tree to make the interpretation more meaningful.

Oversampling In the case of the label extraction with the mean, the classes are unbalanced, so we thought of doing an experiment on the normal dataset and one on the dataset where SMOTE was applied on the minority class, which is an oversampling technique that synthesizes new records.

6.2 Training and validation results

Once the dataset was ready, we performed the analysis using different classifiers on both the median and mean computed label. But we decided to discuss the results obtained for the mean label, because we think that it better suits the definition of player's strength given the fact that it follows a Pareto distribution. We tested different classification models and performed a grid search with a k-fold of 5 to find the best parameters. In the following part we will discuss the results on the dataset where SMOTE has been applied.

Results Analysis Regardless of the dataset used, we have noticed that the algorithms that perform well with some consistencies are the **Neural Network**, **Random Forest** and **SVM**. While those that get bad results with some consistencies are **Rule Based** and **Naive Bayes** (see Table 3). The **Random Forests** have great results, and this is also due to the fact that data are tabular. For **Naive Bayes**, this is probably due to the fact that the algorithm assumes the variables to be conditionally independent among each other. About the **Neural Networks**, we chose to use very simple architectures; the one with the best performance has a hidden layer with 20 neurons and `max_iter`=200. We also observed that for `max_iter` values grater than 200, an overfitting behaviour occurs (Fig. 23). We have also noticed that some of the optimal parameters that have

been identified by the grid search do not match the heuristics. For **KNN** `n_neighbors`, we have the best values of neighbor as 7 or 1 instead of $\sqrt{|training\ set|}$ (about 46) which is the value that can be proposed as heuristic. For **Random Forests**, we have optimal values with `max_features`='None', instead of $\sqrt{\#\ avaiable\ attributes}$ or $log_2(\#\ available\ attributes)+1$.

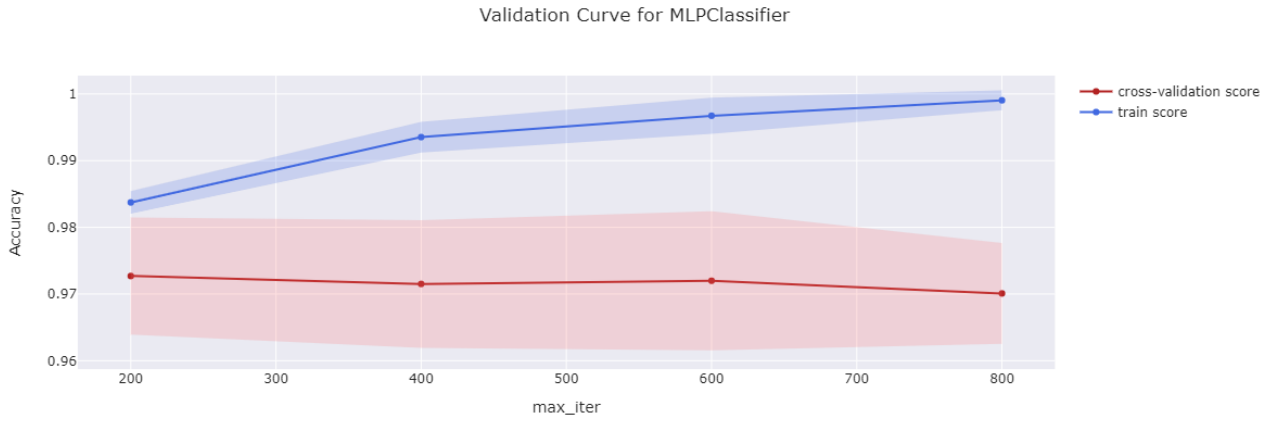
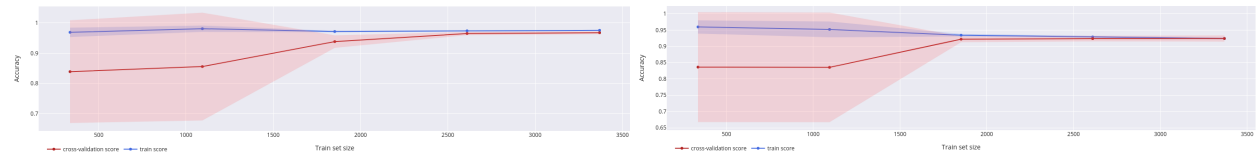


Figure 23: Neural Networks validation curve

Would adding more data for the train have been useful? For each individual algorithm, we have defined a learning curve, which allows us to analyse the accuracy on the train set and on the validation set as the size of the dataset varies. What we can see in general, looking at the learning curves in figure X, is that the accuracy on the validation set initially has a very high standard deviation, but with more than 1800 samples it starts to converge. So we can say that most of the models would not have benefited much with the addition of new data. This means that probably, even using the starting dataset with a larger threshold of matches played, we would have obtained good results.



(a) Random Forest

(b) Naive Bayes

Figure 25: Learning curve for different algorithms.

The best parameters obtained through the grid search can be found for each classifier in the attached notebook.

6.2.1 Decision Tree interpretability

The decision tree performed very well. The best model in terms of accuracy has a `max_depth` of 12, but we preferred to choose a model with a depth of 8 in order to have a better interpretability. With this move, the accuracy on validation dropped just a little (from 0.9648 to 0.9634). It seemed to be a good strategy keeping in mind Occam's razor and also because as can be seen in the validation curve in Figure 27, the standard deviation when the depth is 8 is narrower than its counterpart with 14, where there seems to be a slight overfitting. This ensures that is statistically more significant.

As we can see in Figure 26, the first and most significant split is made on *max_tourney_spectators*. An interesting thing to note is that the left node of the first level has a very low impurity with respect to the right one. This happens because the ones who have never played in a big tourney (with a large amount of spectators), are basically low-ranked, while a discrete amount of low-ranked players could play in important tourneys from time to time. The following splits are made on other features describing the importance of the tourneys in which players have played and their performance.

	Validation (SMOTE)				Validation (Unbalanced)			
Algorithm	**A**	**F1**	**P**	**R**	**A**	**F1**	**P**	**R**
Decision Tree	.96	.96	.96	.96	.95	.89	.91	.87
Rule Based	.92	.92	.93	.91	.95	.88	.86	.90
Random Forest	.96	.96	.95	.98	.96	.91	.90	.90
AdaBoost	.96	.96	.94	.97	.95	.89	.89	.89
KNN	.96	.96	.94	.99	.95	.90	.88	.91
Naive Bayes	.92	.92	.93	.91	.93	.85	.80	.91
SVM	.98	.98	.97	.98	.96	.91	.92	.90
Neural Network	.97	.97	.96	.98	.96	.91	.92	.89

Table 3: Validation metrics for all the models on the normal dataset and on the SMOTE dataset.

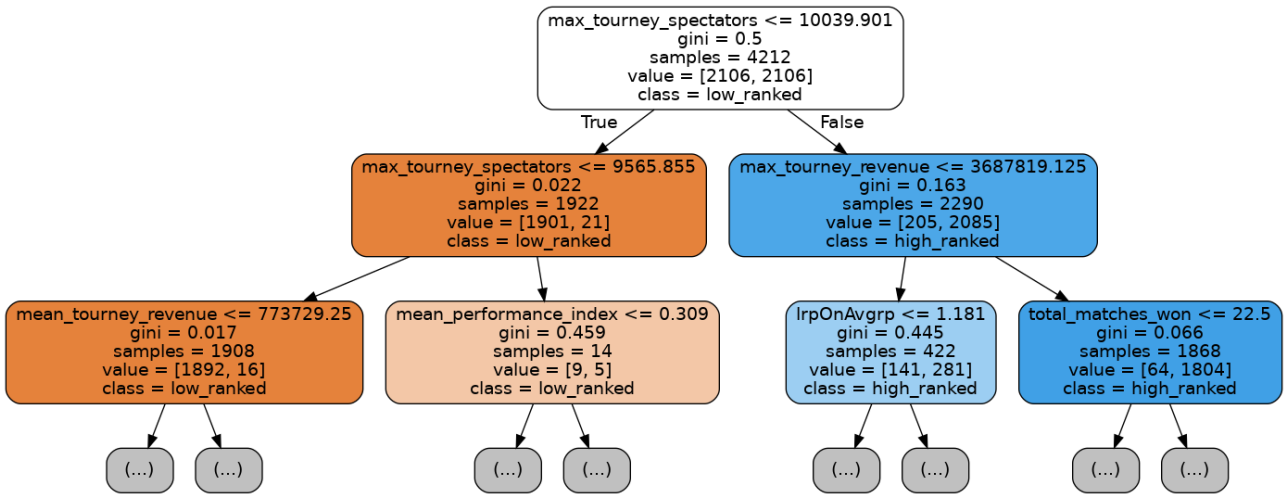


Figure 26: First 3 levels of the Decision Tree.

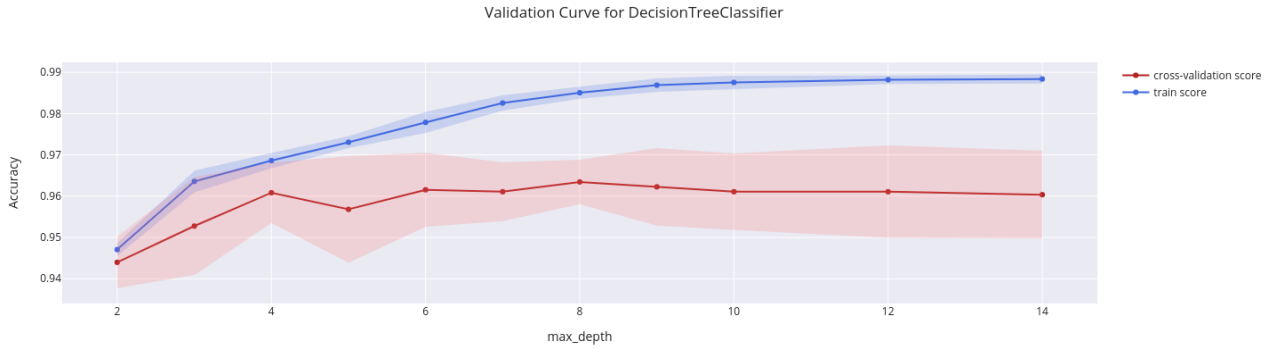


Figure 27: Learning curve decision tree

Let's open a digression for the rule based classifiers, more specifically we used **RIPPER** which in general should have similar performance to a decision tree implemented with the CART algorithm. Even this classifier mainly discriminated the players based on the above-mentioned features, such as *max_tourney_spectators* and *max_tourney_revenue*. Obviously, the importance of some minor features differ, but this is also true for the decision tree when initializing with different random seeds.

6.3 Comparison

After carrying out the analysis and identifying the best models through cross validation, we obtained the following results shown in Figure 29 (the precision and the recall reported are about the high_ranked class, which is also the one with the smallest number of instances). Due to the fact that the test dataset is unbalanced,

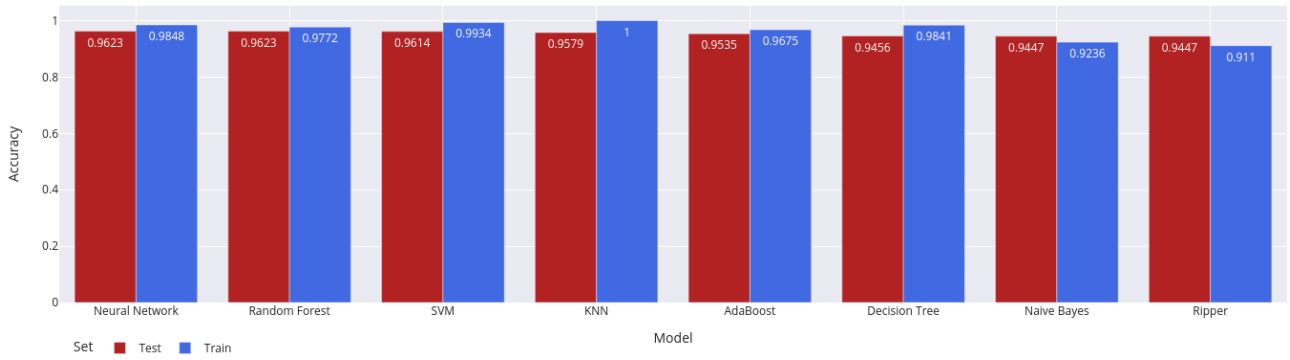


Figure 28: Accuracy on test set (red) and training set (blue).

a ROC curve could provide optimistic results. Accuracy may also be subject to over-optimistic bias, so we can analyse the best performance of the models using the F1 metric. So it is easy to see that the best models are, Random Forest (91.38%), Neural Network (91.06%), and SVM (90.79%). It is also interesting to note that the worst models that are Decision Tree and Rule Based are also the ones that have the best explainability.

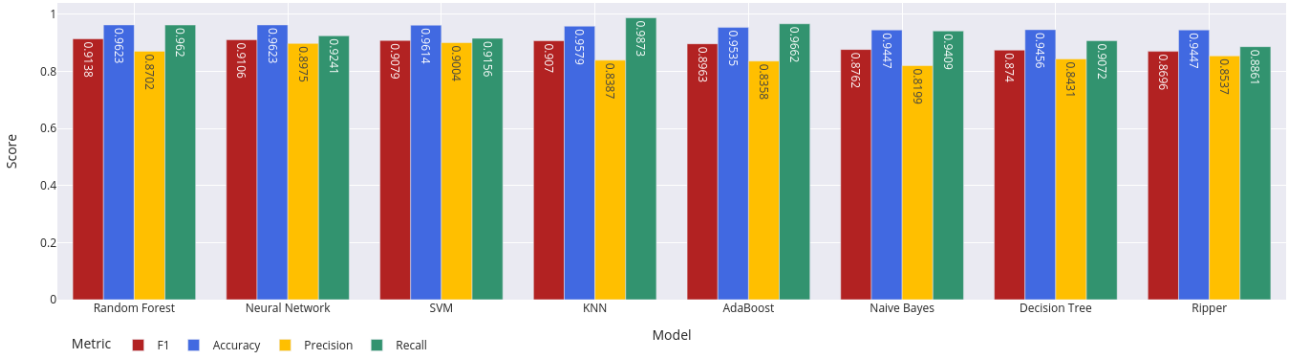
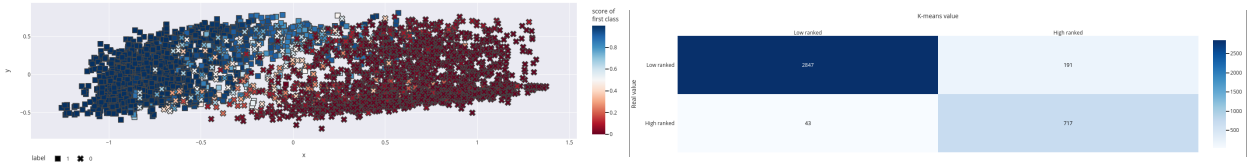


Figure 29: Test metrics

6.3.1 Digression on K-means

In addition to the above, we would like to point out that our clustering analysis (Sec. 5) had already identified a partitioning between strong and weak players. A k-means with a k=2 in fact identifies precisely the strong and weak players (Fig. 31 (b)), and the distribution seems to be Pareto. This is in line with what was said during the decision for the label extraction.

We can in fact appreciate and visualize through a PCA (Fig. 31 (a)) the classification carried out by one of the best algorithms, namely the Random Forest, which by looking at the same visualization vaguely reminds us the distinction performed by the clustering algorithm.



(a) Random Forest PCA visualization

(b) K-means with respect to the label

Figure 31: Comparison with clustering.

7 Time series analysis

7.1 Overview

In this section we will analyse the dataset *CityGlobalTemperature2000-2009.csv* a collection of temperature measurements from 100 cities. For each record we had

- **AverageTemperature**: the measurament of the average temperature.
- **AverageTemperatureUncertainty**: the measurament of the temperature standard deviation.
- **City**: the city related to the measurement.
- **Country**, **Longitude**, **Latitude**: of the related city.
- **Time**: with respect to when the measurement was made.

with data spanning across 10 years. We will exploit this data to find a meaningful group of cities with respect to temperature trends.

7.2 Data Analysis

We initially plotted *AverageTemperature* and *AverageTemperatureUncertainty* attributes regarding some random cities with respect to the time in which they were measured to find some patterns.
While for *AverageTemperature* we can observe the cyclic behaviour derived from the seasonality 32a for what it regards *AverageTemperatureUncertainty* we couldn't find a pattern via this view 32b.

(a) Average Temperature of Lahore city

(b) Average Temperature Uncertainty of Rome city

Figure 33: Average temperatures plot.

7.3 Data Transformation & Feature Engineering

In this phase we created a new dataset (*df_city*) where each record was related to a particular city, and we added features composed ad hoc for the clustering.
We decided to create:

- 12 feature ***month_avg*** where each one represent the average temperature in a given month (*AverageTemperature*) averaged over all the years with respect to a given city.
- 12 feature ***month_var*** where each one represent the temperature variance in a given month (*AverageTemperatureUncertainty*) averaged over all the years with respect to a given city.
- One last feature, ***AverageTemperatureUncertainty*** regarding temperature variance of a given city averaged over all the months and all the years. this feature was created if the previous one did not lead to good results and if the variance of the average temperature was not a parameter dependent on the month in which it is measured.

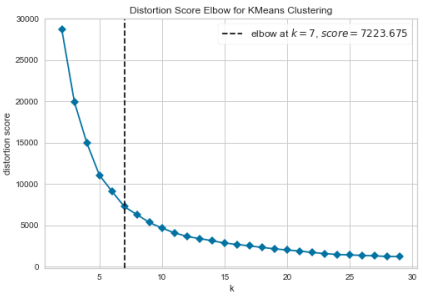
7.4 Clustering

In these phases, we apply the k-means clustering algorithm using different metrics to compute the distance and different combinations of features.
We tried the following combinations of features with euclidean distance as a metric:

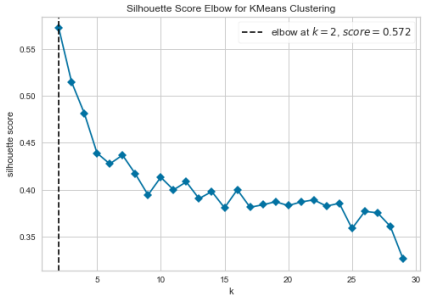
1. Just the 12 *month_avg* features.

2. Just the 12 *month_var* features.
3. The 12 *month_avg* features combined with the other 12 *month_var* features.
4. The 12 *month_avg* features combined with the other 12 *month_var* features.
5. The 12 *month_avg* features combined with the *AverageTemperatureUncertainty* feature.

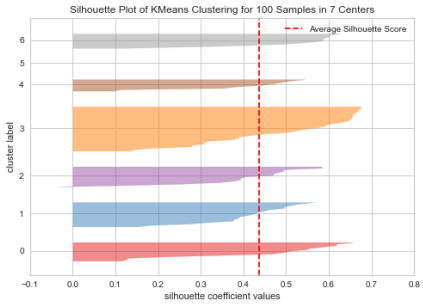
The best results were obtained with the first configuration, where we set k=7 as the number of cluster.



(a) Distortion Score

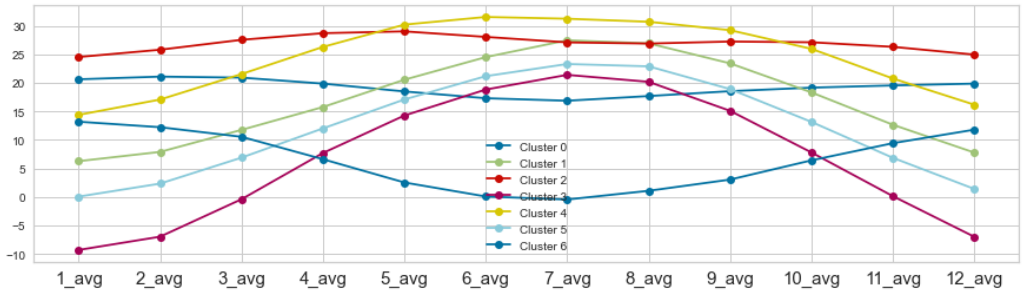


(b) Silhouette score

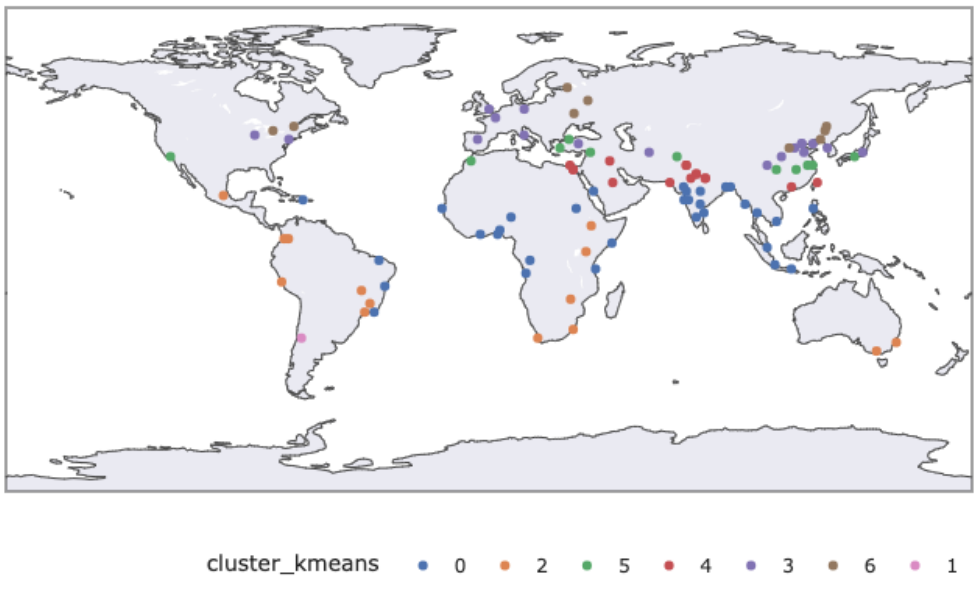


(c) Cluster Silhouette score

We obtained a silhouette score of 0.4369 but what it was more appreciable was the semantic meaning. We can observe how each cluster described a different seasonality 35a and how group of cities are in correspondence with a given a latitude which typically characterized cold or hot places 36a.



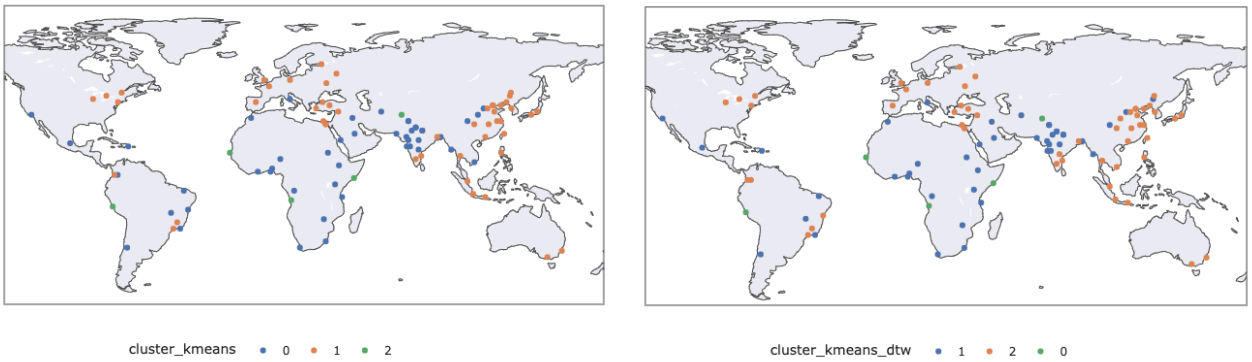
(a) Clustering centroids



(a) Plot on the world map

7.4.1 Further experiments

Trying to cluster on the variance features of the temperature results in worst results with respect to the average. So in case those results were caused by a misalignment in the time series, we tried to approach the clustering by exploiting a Dynamic Time Warping distance in combination with k-means instead of the euclidean distance. However, even with this attempt, the results tend to be almost equal to the Euclidean distance:



(a) World map plot with euclidean distance

(b) World map plot with Dynamic Time Warping distance