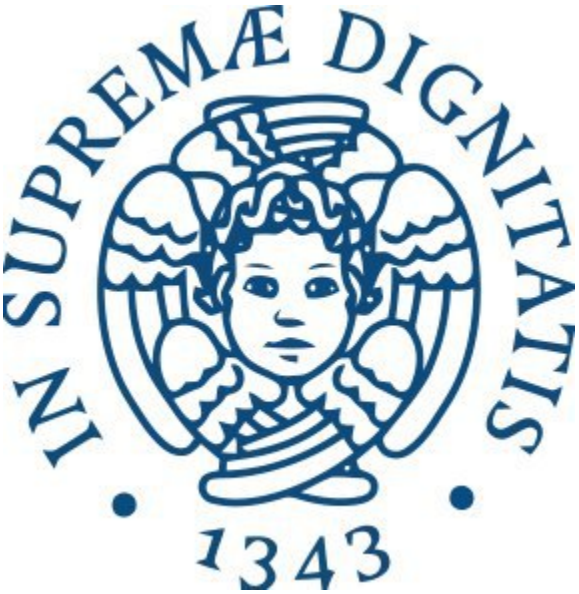
Data Mining Project Report

A.A. 2021 / 2022



Group 3

Mariagiovanna Rotundo

560765

Nunzio Lopardo

600005

Renato Eschini

203021

Summary

1. Introduction

This document contains the report about the data analysis work for the 2021/2022 Data Mining project. It involves analyzing a dataset of tennis matches played in a given period in order to then extract player profiles, find correlations and similarities, and make predictions.
As tools for data analysis we are using Python, in particular, we have used:

- Juptyer-Notebook
- Pandas
- NumPy
- Scikit-learn
- Scipy
- Seaborn (for plots)
- Pyclustering
- Yellowbrick (for finding the best K value for K-Means)

and various other Python libraries and utilities. The work was carried out in phases, below the description of each phase.

2. Data Understanding and Preparation

This chapter describes the Data Understanding and Data Preparation phases.

2.1 Data Understanding

The data understanding phase starts with the analysis and the study of attributes to catch their meaning and correlation, then we move on records analysis to find errors and missing values.

The datasets provided for this project hold information about official tennis tournament matches between 2016 and 2021 and about players separated by sex. They are subdivided into three CSV (*comma separated values*) files:

- **tennis_matches.csv**: *each row represents a match and contains 186129 rows.*
- **male_players.csv**: *each row represents a male player and contains 55209 rows;*
- **female_players.csv**: *each row represents a female player and contains 46173 rows;*

Tennis Matches Dataset

The first dataset, tennis_matches, contains all the information about the tennis matches played for official tournaments in the last five years. Each row is a match and has **49 columns**

including categorical, numerical, ordinal, and ratio-scaled attributes. Their initial part contains information about the tourney like the name, the playground surface, and the level. In the second part, there is information about the performance of both players (winner and loser) in that match and some data on their ranking in the ATP and WTA leaderboards. The table below shows these attributes, for all those that have the prefix "winner" there is a counterpart for the loser player.

Categorical	Ordinal	Numerical	Ratio-Scaled
tournay_id	match_num	draw_size	winner_ht
tournay_name	winner_rank	minutes	winner_age
surface		winner_ace	
tournay_level		winner_df	
winner_id		winner_svpt	
winner_ioc		winner_1stln	
winner_hand		winner_1stwon	
winner_entry		winner_2stwon	
best_of		w_svgms	
		winner_rank_points	
		w_bdsaved	
		w_bdfaced	

Male and Female Dataset

The first analysis shows that female and male files are similar in structure. In fact, apart from the gender of the player that is implicitly defined by the file itself, both files contain names of people (tennis players) formed by two columns: first and last name.

Data quality

This phase involves the evaluation of data quality, in particular, data are analyzed to find inconsistent, null, or duplicate values. We verified the correctness of the data by developing algorithms to compare and recalculate values of various attributes, using plots and consulting external sources.

For each file, first, we have analyzed the attributes checking if there are **missing values** looking at **nulls**. We have seen that in each column of all the datasets, except the surname column in the female dataset, there are null values. We notice that in the tennis dataset the columns about the match statistics (service point, ace, double faults, etc…) have more than half of the values

nulls. Then, we analyze **duplicates and typos/errors**. In all the datasets there are some duplicate rows (at most around 500).

Null values for each attribute

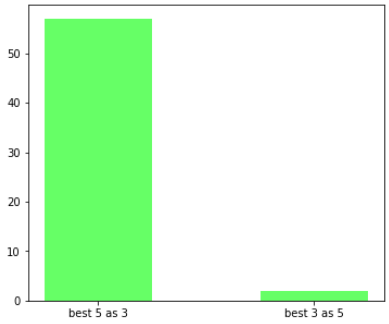
tourney_id	55
tourney_name	25
surface	188
draw_size	29
tourney_level	29
tourney_date	28
match_num	27
winner_id	55
winner_entry	160301
winner_name	27
winner_hand	46
winner_ht	136787
winner_ioc	29
winner_age	2853
loser_id	28
loser_entry	141974
loser_name	31
loser_hand	98
loser_ht	147780
loser_ioc	26
loser_age	6538
score	199
best_of	29
round	30
minutes	104468
w_ace	103818
w_df	103816
w_svpt	103818
w_1stIn	103818
w_1stWon	103816
w_2ndWon	103819
w_SvGms	103817
w_bpSaved	103813
w_bpFaced	103816
l_ace	103815
l_df	103809
l_svpt	103813
l_1stIn	103824
l_1stWon	103817
l_2ndWon	103816
l_SvGms	103810
l_bpSaved	103817
l_bpFaced	103822
winner_rank	19409
winner_rank_points	19427
loser_rank	35283
loser_rank_points	35300
tourney_spectators	27
tourney_revenue	26

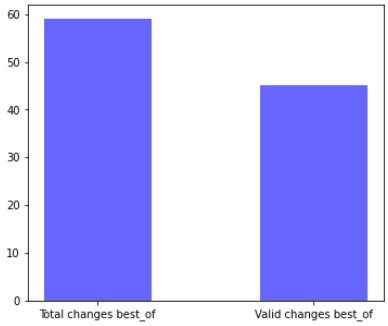
To verify the consistency of the attributes the performed checks are based on its semantic, type and possible relationships with other features.

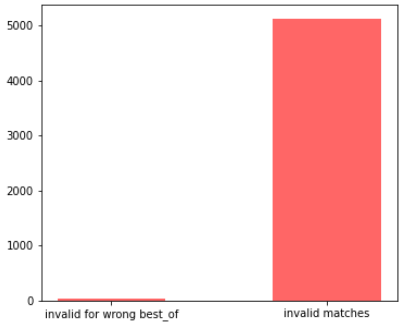
- About categorical values with a finite domain we have to check that all values in the dataset are valid (i.e. are in the domain of the attribute). For example, for the tourney id, we have checked that, for not null values, all the string starts with 4 digits representing a year, for the tourney level that all the values are valid levels, etc.

  For **attributes** that are **unique for each player** (each player can have only one value for that attribute considering all the matches), we have checked that for a player there are no more than one value except for an unknown value where present (for example we check that the hand a player uses for the matches is always the same except for the value that indicates that the hand is not known). The work on this issue has revealed that some players have more *names* bound with the same *id* and vice versa.

  We have analyzed the **relations between different attributes** looking for attributes with dependencies between them. For example, we have noticed that the *score* attribute and the *best_of* attribute are related because the *best_of* defines the maximum amount of playable sets in the match. So we have analyzed these inconsistencies to find a pattern that we could use to verify the correctness of these values. During the analysis, we have found, about the cited example, some rows where the score presents a number of sets bigger than the one indicated by the *best_of* field.







- For numerical attributes, we first verify the presence of not allowed values like negative numbers, outliers, or inconsistencies between attributes. The first check revealed that none of the columns presents negative values.
  We have found unrealistic values and errors that we identify as outliers. An example of an error is in the winner_height column (in which the height is expressed in centimeters) where a player with a height of 2 appears, and this has been classified as an error. In the case of outliers, in longer matches, the number of served points is much higher with respect to the average.

For the datasets of players, we have checked if all the names and surnames in male and female datasets are valid, so they don't have invalid characters like "?" or numbers. We noticed that rows with **quotation marks ("") before the split symbol** are loaded by pandas in the wrong way, reading 2 fields as only one string and leaving the other one as null. Furthermore, in the male dataset, there are some rows with "unknown" values that have to be considered as missing values. About duplicates, we have noticed that there are some common rows (74) between both male and female datasets, so names that can be both female or male.

Correlation Analysis

We performed correlation analysis on the different numerical values identified. To the resulting correlation matrix, we have applied a filter with a threshold fixed at 0.8 so as to highlight the values closest to 1. We can highlight that there are some aspects very correlated such as between **"w_svpt, w_1stIn, w_1stWon, w_SvGms"** or between **"w_bpSaved"** and "**w_bpFaced"**.

w_svpt, w_1stIn, w_1stWon, w_SvGms are all correlated and this can be interpreted as the fact that more service game (w_SvGms) a player does and, as consequence, more serve points (w_svpt) he does (because if the player serves in a game, he serves for all the points in that game, so these number increase together). If a player serves, the number of the first serves (W_1stIn) increases. Increasing the number of first serves, the probability of winning some of

these serves increases so, the more first serves the player does, the more won first serves (W_1stWon) we expect. The same reasoning is done also for the loser (**l_svpt, l_1stIn, l_1stWon, l_SvGms**).

Some statistics of winner and loser are correlated too, for example, **w_svpt** and **l_svpt** or **l_1stIn.** This can be interpreted as the fact that the winner and loser serve in an alternate way so, if the number of serves of the winner increase (for example because of more games), then also the number of the loser increase.

w_bpSaved and w_bpFaced are correlated and this can be interpreted as the fact that more are the breakpoints faced and more can be the breakpoint saved. The same reasoning is done also for the loser (**l_bpSaved** and **l_bpFaced**).

draw_size, **tourney_revenue**, **tourney_spectators** are all correlated. This can be interpreted as the fact that the tourneys with more players attract more people to see the matches and they have a bigger budget with respect to the tourneys with fewer players.

2.2 Data Preparation

In the data preparation, we use observations done in the understanding phase to correct data in all the 3 datasets (male, female, and tennis). In the **male** and **female** datasets, we correct names and surnames removing characters that can't be there.

In the **tennis** dataset, all the features require to be corrected. We correct the values of some analyzed correlated columns where possible, but this is not always possible. For example, in matches, the information about the number of double faults, first serves, the first serves won, number of aces, number of breakpoints (faced and saved), and the second serves won can be retrieved by no other feature and they depend on the match. So, to correct errors and null values we have used the default value -1. Other features that can't be retrieved analyzing other variables are some categorical information as for example the round and the score because their values depend on the played match.

Instead, some examples of features that can be obtained with the analysis of the dataset are *best of*, minutes, service game of winner and loser that can be obtained from the score because these features depend on it. For example, the *best of* depends on the number of sets, the minutes, and the service games by the number of games, and the number of sets and games are information contained in the score. This is because if the maximum of sets is 3, the match is *best of* 3, if the number of games increases, the minutes increase as consequence, and the

service games increase for both the winner and loser since they serve in an almost alternated way. To compute the minutes looking at the score, we compute an average time to play a game.

In other cases, some values can be retrieved looking at other rows, in particular when need to static (information that does not change for tourney and player) information about tourney id, name or surface or also for the player's name, id, hand, or height.

In some cases, to correct the errors, we use an external source. For example, we have used the external file country-codes_csv.csv to obtain the correct IOC value looking at the wrong one (because expressed in other standards) associated with winner and loser.

For all these attributes, when, for one or more matches, we cannot compute a correct value we assign a default value.

Dataset Players

After finishing the analysis and correction of the various datasets, we have moved on player profiling. After collecting this basic information we have started feature extraction aimed to define the player performance. Below is the list of the extracted attributes, collected in **semantic groups**:

- **Personal information** [*id, whole_name, sex, IOC, hand, height, birthdate*]: aggregate personal information of tennis players. The data are extracted from the corrected dataset except for the birthdate which is calculated by taking the age in days of the player in a match and calculating the date of birth based on the date of the match.
- **Rank** [*best_rank, best_rank_points*]: the maximum value (among all the player matches) of position and score in the rank.
- **Information on the matches played** [*best_of_3_match, best_of_5_match, best_of_3_wins, best_of_5_wins, m_16, w_16, ..., m_21, w_21, nmatch, wmatch, tot_minutes, n_tourney, w_tourney, lmatch*]: relating to the played matches, *best_of_*_match* and *best_of_*_win* are the matches played and won in the "at the best of ..." formula, *m_** matches played in the year, *w_** matches won in the year, number of matches played and won, number of minutes played, number of tournaments played, won and lost.
- **Related to the service** [*ace, sv1st, sv1st_win, sv2nd_win, df, bpS*]: number of aces played, first and second service wins, number of double fouls, and number of breakpoints saved.
- **Related to the surface** [*w_surface_Hard, w_surface_Clay, w_surface_Grass, w_surface_Carpet, l_surface_Hard, l_surface_Clay, l_surface_Grass, l_surface_Carpet*]:

this is the aggregate information of how the players have behaved on the various playing surfaces.

Correlation Analysis

Also after the correction to the dataset of tennis, we have performed correlation analysis on the different numerical values identified to analyze if some correlations are changed. To the resulting correlation matrix, we have applied a filter with a threshold fixed at 0.85.

We notice that some features correlated before changes continue to be correlated:

- w_svpt, w_1stIn, w_1stWon (also l_svpt, l_1stIn, l_1stWon)
- w_bpSaved, w_bpFaced (also l_bpSaved, l_bpFaced)
- draw_size, tourney_revenue, tourney_spectators

We can highlight that there are also some more aspects very correlated with respect to the dataset before the changes:

- **w_2ndWon** is correlated to **w_svpt** (and **w_1stIn, w_1stWon**) and this can be interpreted as the fact that the second serves are a part of service points, so more w_svpt increase and more can w_2ndWon increase too. The same reasoning is done also for l_2ndWon and l_svpt (and l_1stIn, l_1stWon).
- **w_svpt, w_1stIn, w_1stWon, w_2ndWon, l_svpt, l_1stIn, l_1stWon, l_2ndWon** are all correlated. For the correlation between the winner and loser attributes the same reasoning as the one done in the understanding correlation analysis is done.
- **w_bpFaced, w_svpt, w_1stIn, l_svpt, l_1stIn** are all correlated. The increasing number of service points by the players can be interpreted in two ways:
    a. the players make many services in a small number of games reaching many times a draw and this as consequences lead to having many breakpoints because in case of the draw we have a breakpoint every time a point by a player is done after the draw.
    b. The players play many games and to win a game a player must reach the situation where he needs only one point to win. So, a breakpoint, to win a game, is present for sure. So, for these reasons, more service points are done by the players, and more breakpoints increase. These columns are correlated also to w_1stIn and l_1stIn because to serve a point a player must do a first serve.

  This reasoning is done also for **l_bpFaced**.
- **w_SvGms, l_SvGms** are correlated. Possible reasoning is that they are correlated because the service is alternated between the winner and loser player by games (the

serves approximately the same number of games in a match. So, if w_SvGms increase also l_SvGms increase (cannot have high w_SvGms and low l_SvGms or vice versa because of tennis rules).

Correlation before and after the changes

After the changes to the dataset of tennis, we use a threshold slightly bigger than the one used in the understanding because we consider that before the changes there are more errors and this can influence the correlation.

We can notice that some correlations are introduced and others are removed. For example, the correlation between w_svpt and w_SvGms, which is quite high before the changes, disappears after the changes, and the w_2ndWon, which was not correlated to anything before changes, is highly correlated for example with w_svpt, w_1stIn, w_1stWon.

3. Clustering analysis

For the clustering analysis, we apply the various clustering algorithms to the df_players dataset that contains player profiles. Not all the information of this dataset is used. We use only more interesting columns and we consider only players with a number of matches played over a specific threshold to not consider players with few significant data. We consider only numerical data because categorical ones influence the results of the clustering. After the selection of the numerical data we want to consider, we analyze the correlation between these data removing the highly correlated ones. This allows us to reduce the dimensionality of the space to analyze without losing important information. We consider highly correlated two columns if they have a correlation bigger than 0.9.
The second phase is to apply the studied clustering techniques, in particular:

- K-means
- Density-Based Clustering
- Hierarchical clustering

As alternative clustering techniques using the pyclustering (https://pyclustering.github.io/) library, we have considered:

- X-means
- G-means

For these algorithms we try to find the best parameters to use to obtain the best clusters, for example, for the density-based clustering, we analyze the number of clusters and the mean

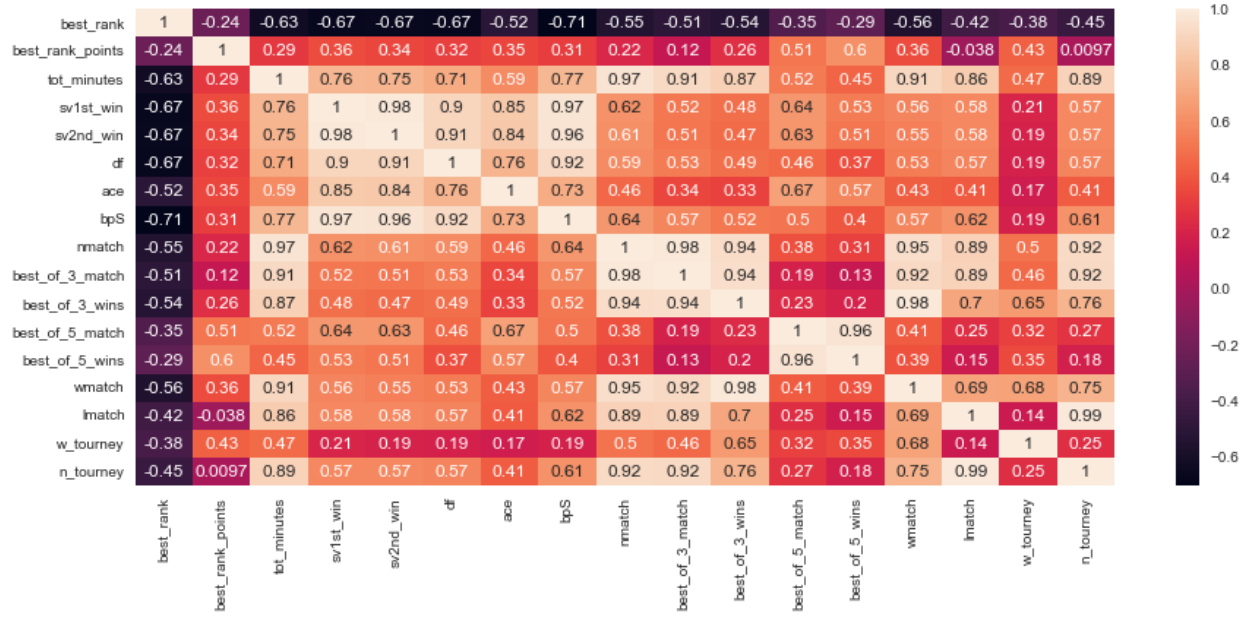
distance between noise points and closer points to find the best radius and the best minimum number of points to use to create clusters.

In the last part related to clustering, we have a brief conclusion on the various algorithms used and the validity of the results obtained.

3.1 Clustering Analysis by K-means

Elimination of highly correlated features

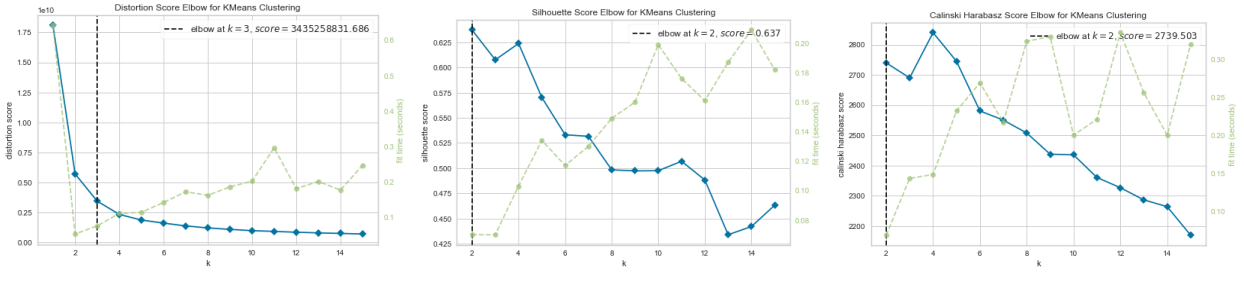
We begin by examining the correlations between the attributes of the dataset to be clustered in order to identify the highly correlated couples. Dropping redundant attributes benefits the analysis by reducing the dimensionality of the dataset and raising the influence that more useful features could have on the whole clustering process.

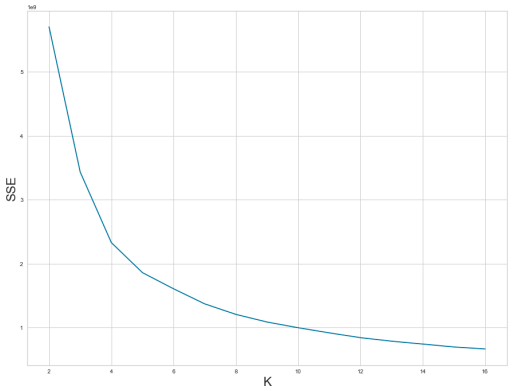


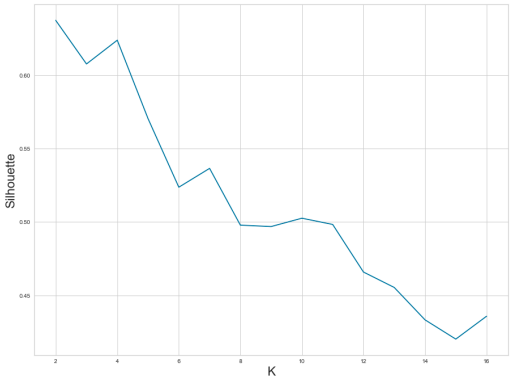
We remove *'best_of_3_match', 'best_of_5_match', 'sv2nd_win', 'tot_minutes'* that have a high correlation, which we think are redundant and do not contribute to characterizing the cluster.

Identification of the best value of k

We have tried to find the best K-value with the Elbow method using a ready-made Yellowbrick KElbowVisualizer library (metrics including *distortion*, *silhouette*, and *Calinski-Harabasz*) and by calculating the *silhouette score* and *SSE* with metrics by using Sklearn functions.







K	SSE	Silhouette
2	5705165593	0.637339
3	3435279242	0.607642
4	2331120432	0.623832
5	1861197140	0.570428
6	1608318552	0.523712
7	1372740132	0.536540
8	1208329512	0.497878
9	1088953560	0.496919
10	999632750	0.502610
11	919182254	0.498346
12	844332730	0.465942
13	789325162	0.455446
14	744308359	0.433394
15	699014957	0.420270
16	667663930	0.435851

We can see from the table on the right, that a high silhouette score (greater than 0.6) with a low SSE value we get with **K = 4**, so we decided to use it.

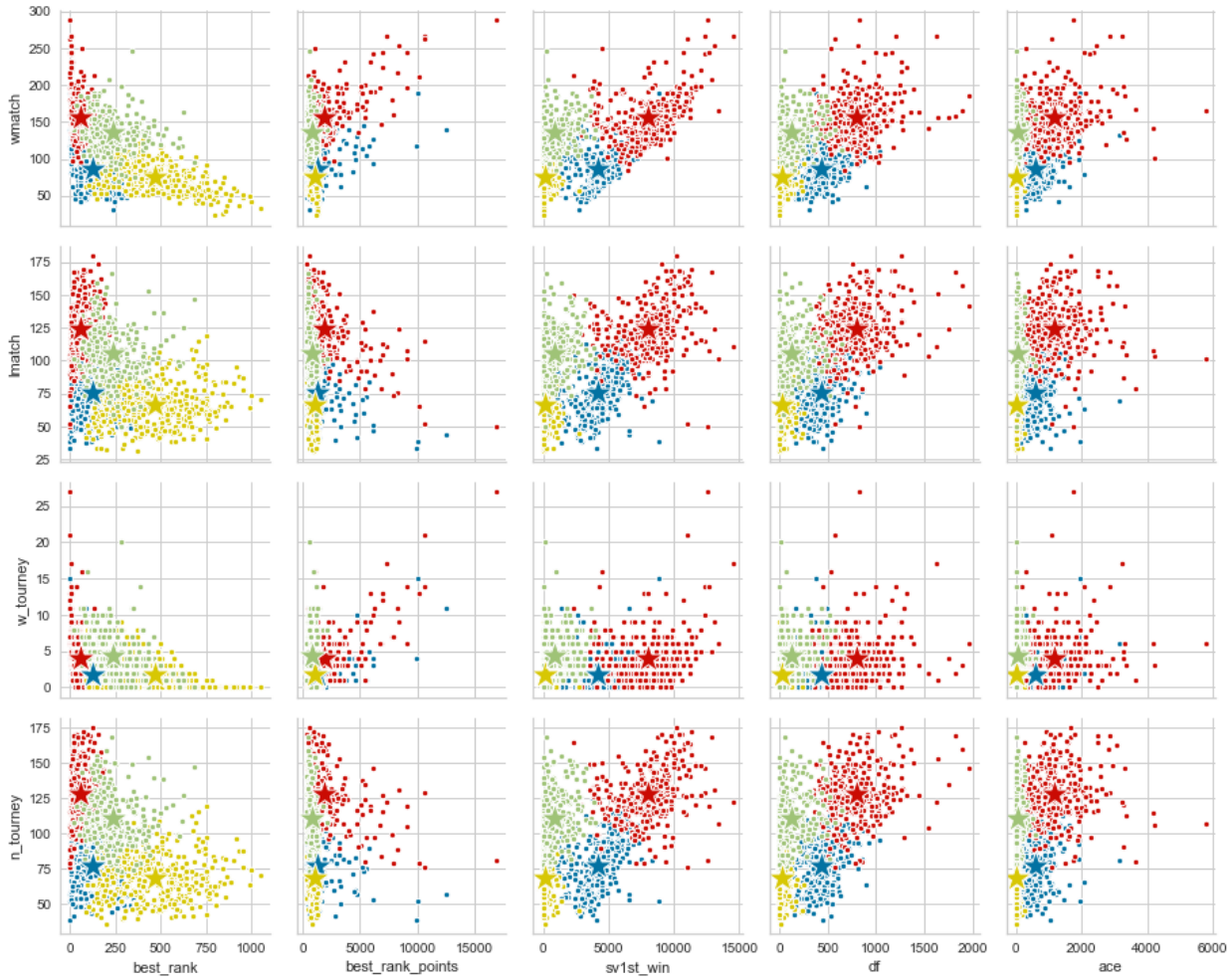
Normalization

We normalize the values using MinMax in order to have the same scale on all attributes.

Characterization of the obtained clusters

We set the number of clusters to 4, the labels of clusters are [0, 1, 2, 3] and the number of items for each cluster is [293, 318, 400, 254].

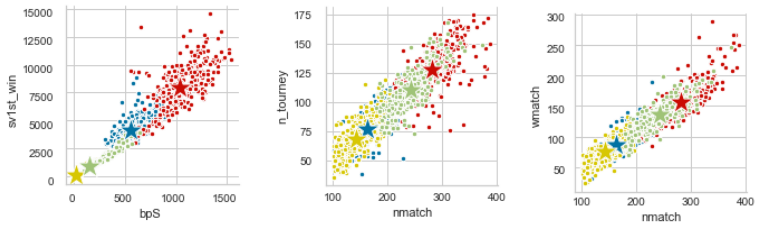
We plot clusters using PairGrid. The next image shows a piece of the whole plot, where is possible to notice the clustering results of the k-means using on the y-axis the attributes *n_tourney*, *w_tourney*, *nmatch*, *wmatch* and on the x-axis *best_rank*, *best_rank_points*, *sv1st_win*, *df,* and *ace*. In addition, the centroids of the clusters are highlighted with a star symbol.



Evaluation of the clustering results

By setting **K = 4**, the items in the clusters are distributed evenly. We can see that some combinations of attributes show a diagonal trend from bottom left to top right, it happens when there is a high correlation between attributes such as:

corr(*sv1st_win, bpS*)=0.9682; corr(*nmatch, n_tourney*)=0.9217; corr(nmatch, wmatch)=0.9456;



3.2 Analysis by density-based clustering

To avoid problems due to the different scale of values, for the density-based clustering the dataset is normalized. For the normalization 2 scaler are used: the StandardScaler and the MinMaxScaler and both the new scaled dataset are analyzed. For the analysis by density-based clustering, the DBscan algorithm is used. This algorithm requires 2 parameters:

- the radius (ε), that is the distance in which looking for the neighbors,
- the minimum number of points to create a cluster.

So, to find the best clusters we analyze what are the best parameters to use. To find the best combination of the 2 parameters we define to metrics:

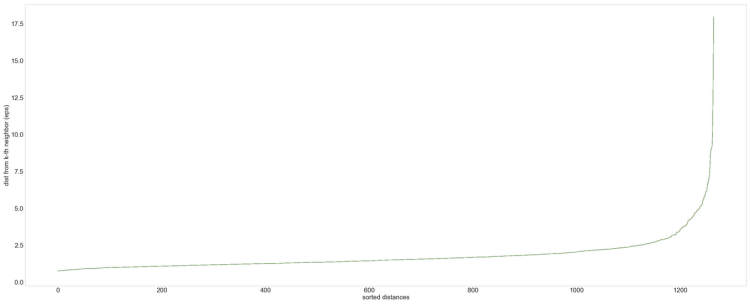
1. the mean distance between the noise points and the closer points (6 closer points, we use the 6-NN),
2. The number of clusters we obtain.

So, we define a list of values of ε and a number of minimum points to try. We use an elbow method on the distances to find an upper bound to the ε to try. After the choice of the best parameters, we plot the clusters that we obtain.

Study of the clustering parameters

The best parameters and the clusters that we obtain change by looking at the used scaled dataset.

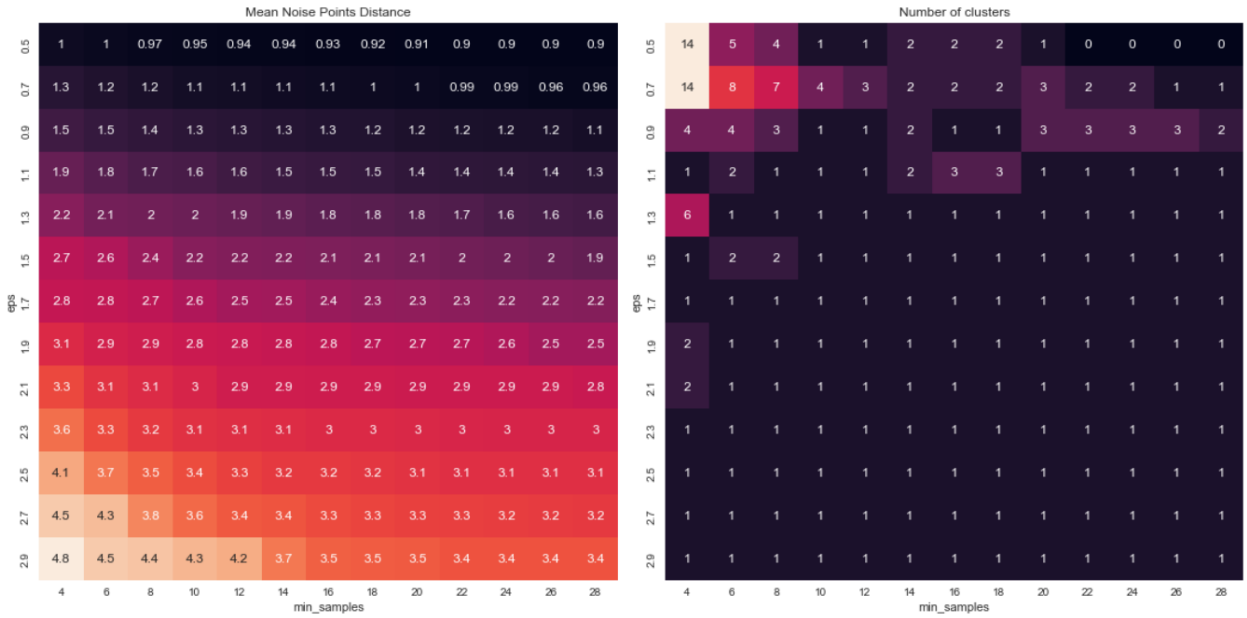
StandardScaler parameters



For the dataset scaled with the StandardScaler, the values used for ε are between 0.5 and 2.9 (we use 3 as the upper bound of ε using the elbow method on the distances on the picture on the right) and for the minimum number of points, the used values are between 4 and 28.

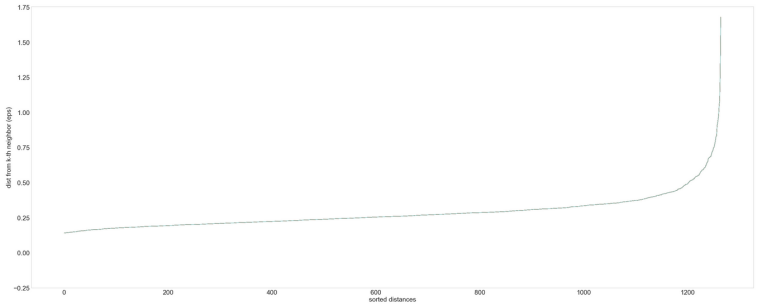
The value of metrics we define, using combinations of the chosen values, are summarized by the following pictures (on the left there is the mean distance and on the right, the number of clusters), where the values of ε are on the y-axis and the values of min number of point are on the x-axis. Looking at these metrics we choose ε=1.1 and the minimum number of points=16 to

obtain 3 clusters (plus one for the cluster for unclassified point), which can be a good number of clusters for our data, having on average more distant noise points. We obtain clusters with the following number of points: [427, 803, 19, 16].

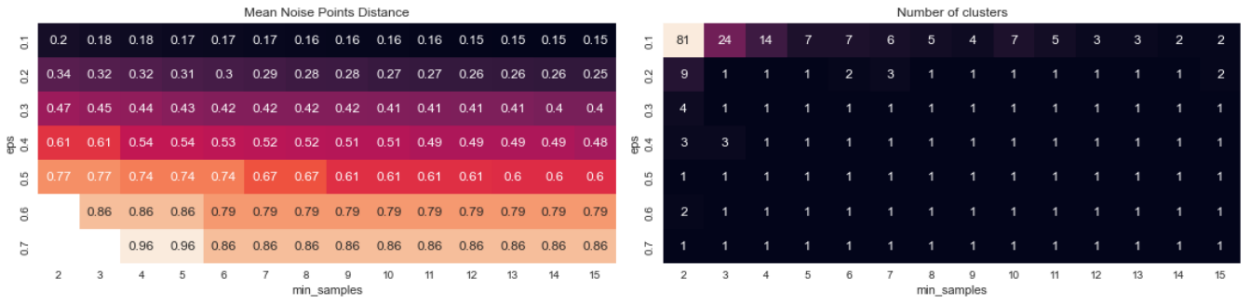


MinMaxScaler parameters

For the dataset scaled with the MinMaxScaler, the values used for ε are between 0.1 and 0.75 (we choose the upper bound of ε using the elbow method on the distances on the picture on the right) and for the minimum number of points, the used values are between 2 and 15.



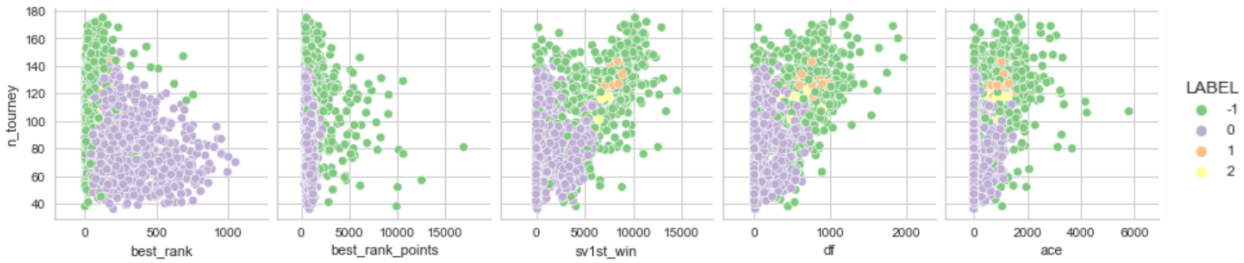
The value of metrics we define, using combinations of the chosen values, are summarized by the following pictures.



Looking at these metrics we choose ε=0.1 and the minimum number of points=9 to obtain 4 clusters (plus one for the cluster for unclassified point). We choose the minimum number of points equal to 9 and not 2 to avoid having only one big cluster and then only clusters of about 2 elements (as happens with ε=0.3 and the minimum number of points=2). We obtain clusters with the following number of points: [986, 21, 151, 95, 12].

Characterization and interpretation of the obtained clusters

With StandardScaler, the cluster of unclassified points has around 430 points while with the MinMaxScaler it has almost 1000 points. Considering the classified points, the **StandardScaler** has one very big cluster composed of 800 points and 2 other clusters very small (around 20 points). So, the clusters are very unbalanced. In the following picture, some plots of the obtained clusters are shown.



We can notice that only the cluster for unclassified data and the cluster with 800 points are quite well separated but, for the others, this is not true.

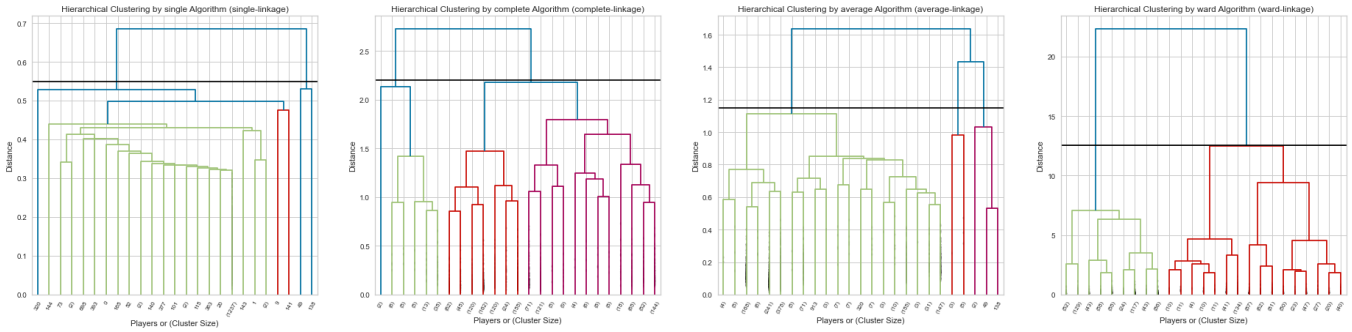
Considering the classified points, the **MinMaxScaler** has clusters with smaller unbalance. In the following picture, some plots of the obtained clusters are shown. Here, the obtained clusters seem to be slightly well separated with respect to the clusters obtained with the use of StandardScaler, so the MinMaxScaler seems to work slightly better than the StandardScaler. But, for both the used scaled dataset, the number of unclassified points and the not-so-well separated clusters make results are not as clearly describable.

Choosing different parameters does not help to obtain better clusters. The performances can

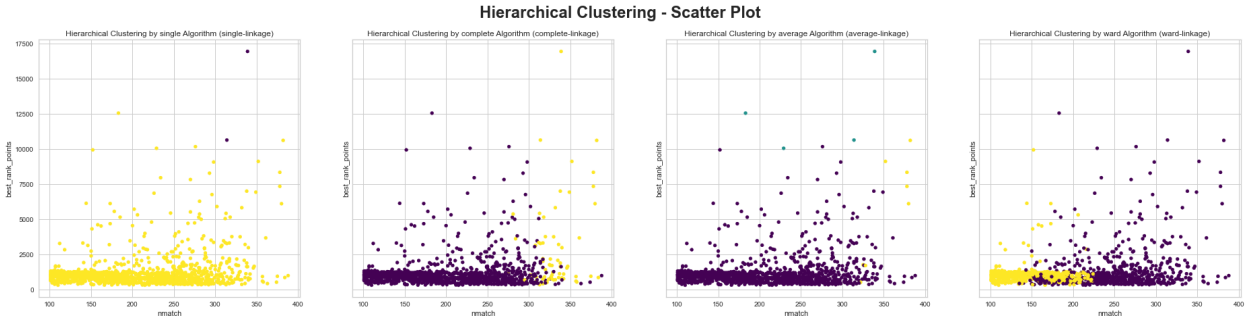
improve a lot using more features (for example considering also the removed feature with high correlation) to increase the dimension of the considered space obtaining cluster quite well separated.

3.3 Analysis by Hierarchical clustering

For hierarchical clustering has been used the four different linking techniques *single, complete, average*, and *ward,* and the results have been analyzed and visualized through a dendrogram plot. For all the four methods has been used the *euclidean distance* to measure the records distance. Also in this case we used a MinMax normalization technique for the dataset.
The image below shows the dendrogram for all linking methods. Looking at the hierarchy and the colors of the clusters is possible to identify the presence of one big cluster, mostly for the *single* and *average* linkage. This is due to the high similarity of records making the minimum linkage solution fall in the creation of one big cluster, this property also affects the calculation of the average between objects making it visible again for the average method. Concerning the complete linkage, this phenomenon is reduced by the search for the most different items leading to the split of the largest cluster and the creation of a third group.



Using the dendrogram cut technique the number of clusters is always between two and three. To operate these cuts has been found the longest vertical segment without a cluster interruption. In the following scatter plot it is possible to see the results of the clustering after the cut. In the third plot, a few blue points remark the presence of the third cluster in the average linkage.

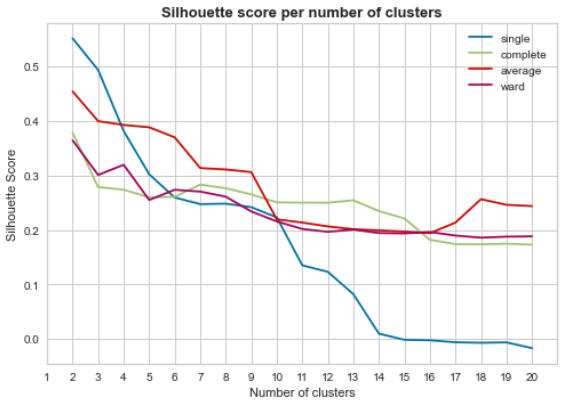


To evaluate clustering accuracy, we used three metrics: silhouette score, Davies-Bouldin index, and the Calinski and Harabasz score.

We also analyzed the silhouette score trend using an iterative approach on the cluster number. This method reveals that for all linkage techniques the best number of clusters is two.

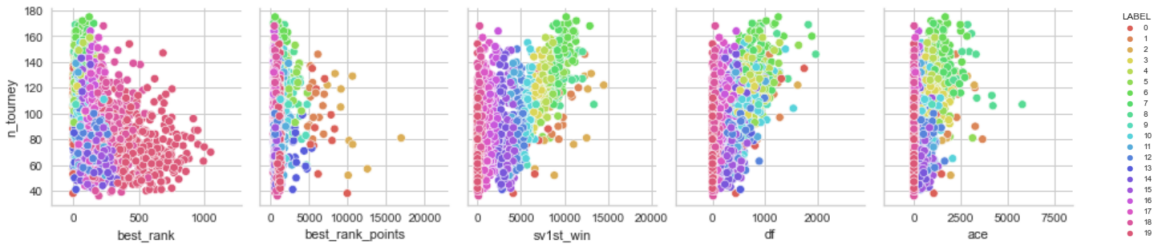
As you can see from the plots and the scores in the table, the hierarchical clustering, in general, does not obtain good results. In the case of the ward method it is possible to obtain a sufficient distinction between two classes of players and, in addition, looking the plot on the right you can see that the ward method could get good results also with 4 clusters.

	Silhouette	Davies_bouldin	Calinski Harabasz
single	0,55	0,48	15,30
complete	0,38	0,80	268,56
average	0,40	0,82	30,41
ward	0,36	1,01	1.010,13



3.4 X-MEANS

The X-means starts with only one cluster and splits clusters using the Bayesian Information Criterion. We fix the maximum number of clusters at 20, so if X-means reaches this number of clusters it stops to split. The number of clusters obtained is 20 and they have the following number of points: [78, 61, 53, 62, 158, 77, 126, 189, 32, 60, 26, 85, 28, 17, 84, 62, 30, 19, 11, 7]. The SSE of this clustering is 628542779.1048735. In the following picture, some plots of clustering are shown. We can see that looking at some dimensions, they are quite well separated.



3.5 G-MEANS

The G-means algorithm starts with a small number of centers and grows the number of centers. Each iteration of the G-Means algorithm splits into two centers whose data appear not to come from a Gaussian distribution. G means repeatedly making decisions based on a statistical test for the data assigned to each center.

We set the max number of clusters to 20. The result is in 20 cluster with this number of items for each: [139, 188, 76, 146, 64, 63, 54, 24, 80, 13, 57, 81, 15, 31, 9, 16, 85, 26, 73, 25]

The SSE returned is 676312960.5330845, we compare this value with X-Means because they are very similar.

3.6 Final clustering evaluation

Keep in mind all the methods that we have considered for clustering analysis, the one that on the selected dataset **separates** better the clusters and has clusters quite **balanced** (the clusters have similar sizes) is the **K-Means**. Meanwhile, the other two approaches have more unbalanced clusters, because there is a big cluster and the others are small, and a worse separation for the reasonings above. About the **X-Means** and **G-Means**, we can notice that they have similar results: both find 20 clusters (that is the maximum fixed in the method), and also the obtained clusters found by both algorithms are very similar (in size and shape of clusters). Furthermore, the clustering of X-Means and G-Means shows very similar SSE.

The main difference between X-Means and G-Means and K-Mean is that for the K-Means the better number of clusters is 4, while the other 2 methods consider as best number the 20 (in the range of numbers between 0 and 20). As we expect, since the K-Means has a smaller number of clusters, it has an SSE bigger than the one of the X-Means and G-Means.

4. Predictive Analysis

The predictive analysis aims to predict player ranking. For this task, we use the dataset of players discretizing categorical features. To assign labels, the **best_rank** feature with a

threshold of 50 is used, so if the *best_rank* of a player is smaller or equal to 50 the player is a high ranked player, otherwise he is a low ranked player. We consider for this task only the players that have the *best_rank* known (greater or equal to 1).

For this task, different classificators are evaluated. In particular, the following classificators are considered:

- Decision Tree (DT)
- Support Vector Machine (SVM)
- Rule-Based (RB)
- Gaussian Naive Bayes (GNB)
- AdaBoost (AB)
- Random Forest (RF)
- Neural Network (NN)
- k-Nearest Neighbors (KNN)

Since the dataset for train and test are imbalanced, for the classification task, 3 different approaches are evaluated:

1. The classification is done on the train and test set without considering the unbalanced labels;
2. The classification is done on the train and test given different weights to the 2 classes (high rank and low rank);
3. The classification is done using an oversampled train set using SMOTE.

We have also considered other changes to the dataset for specific classificators like Neural Networks and K-Nearest Neighbors. For the KNN, we removed the categorical attributes (hand and sex) because it is based on distance, and categorical attributes influence in a wrong way the results. For the NN we used a normalized dataset (*MinMaxScaler*) because after some tests we noticed that the original dataset leads the training model to a huge underfitting.

We consider the high class as the important class, so to measure the classificators performances we analyze precision and recall only of this class. In particular, to compare the different classificators we use the F1 measure.

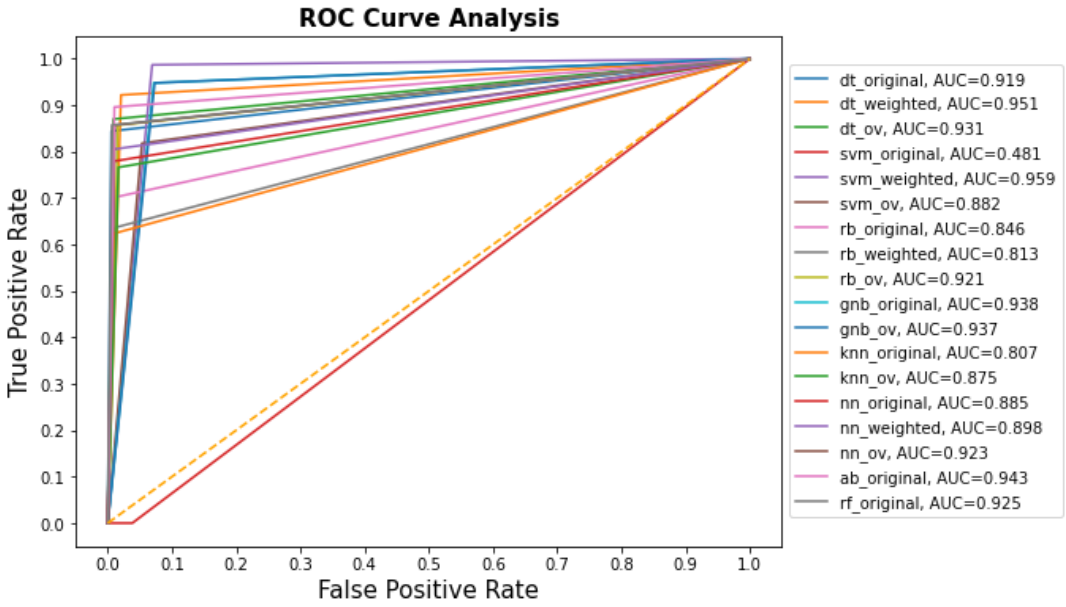
Analyzing the performance of the various classifiers, considering different splits for training and test set, we have noticed that different models, such as DT, RF, AB, and NN, often tend to have very good performance. In particular, we can notice that DT, AB, and RF typically have an F-score bigger than 0.85 for the test set and for training greater than 0.90. Instead, the NN shows an F-score around 0.80 both for training and test sets. Also, RB and GNB have quite

good scores but less than the previous classifiers, in fact, the scores are respectively around 0.70 and 0.60. The KNN model tends to have performances a little bit worse than NN but presents overfitting because on training set has an F1-score of around 1 while on the test set it is around 0.70. The SVM is the only classificator that presents a huge underfitting, indeed it classifies all the samples in both training and test as instances of the “low” class (its f-score is 0).

The DT, SVM, RB, and NN classifiers are evaluated using a **weighted dataset** and we have noticed that except for SVM the performances are similar to the ones of the previous analysis. The SVM, instead, presents a big improvement because the F1-score, both from training and test sets, from 0 increases to around 0.60.

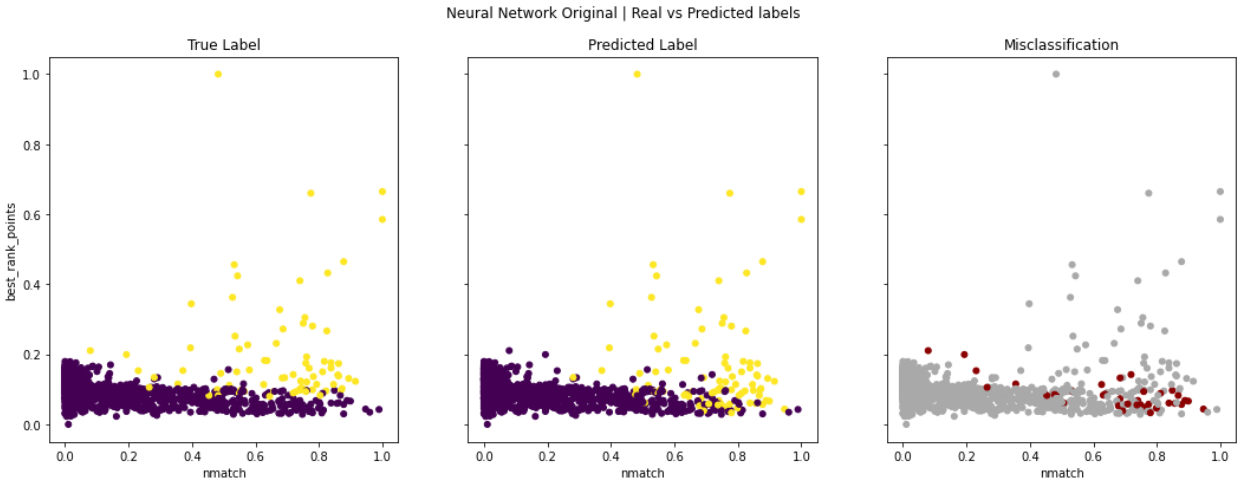
Instead, using the **oversampled dataset** we tested all classifiers except AB and RF. Here we have noticed that the performances slightly increase with respect to those obtained in the dataset which does not take into account the imbalance, with the exception of the NN, which has a higher degree of overfitting, and of the SVM. Its performances depend a lot on the oversampled training set, and, for this reason, can have performances slightly better than the ones in the first analysis or similar to the ones in case of use of weights.

In the following picture, the ROC curves of the different classificators are shown.



In the following picture, an example of classification on players (using the best rank point and the number of matches on the axis) obtained using the Neural Network is shown. In the picture on the left, the true labels are present, in the picture on the center the predicted labels are

shown, while in the picture on the right there are misclassification errors.



5. Time Series Analysis

For task 4 we have chosen **TASK 4.1**. To find groups of similar cities with respect to the temperature trends we have extracted the mean values of temperature collected in time. We consider the trend of 2 cities similar if the shape of the time series is similar. We consider 2 trends similar also in the case they are translated in time and the amplitude of the oscillations of the temperatures are different.

With these premises, we started retrieving the time series associated with the cities obtained 100 time series (1 for each city). Since that to study the similarity we use the concepts of the distance between time series, considering also the euclidean distance, we check if all the time series have the same length and we verify that this is true. After this check, we start to evaluate the similarity using the clustering techniques. The used clustering methods are K-means and hierarchical clustering

K-MEANS

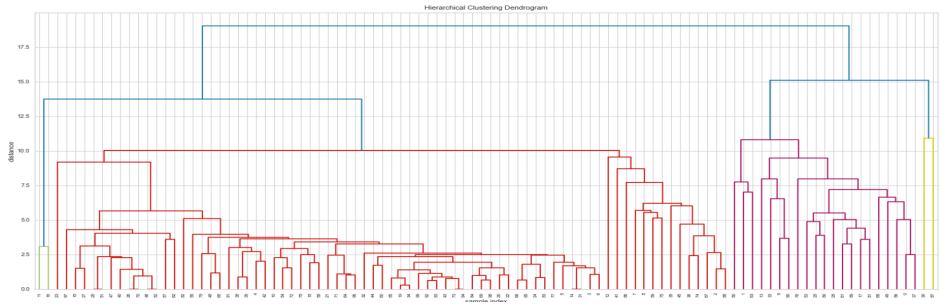
For Kmeans is used the library tslearn. For the Kmeans, we analyze the results using 2 different distances: the Euclidean distance and the Dynamic Time Warping (DTW). For the analysis with the Euclidean distance, we correct the offset translation and the amplitude scale distortions.

For both the distances, we study what is the correct number of clusters using the elbow method and we apply the obtained value for the clustering. Then we compute the clusters and, in the

end, we print the time series and the cities divided by clusters.

HIERARCHICAL CLUSTERING

Since the dataset is small (it is composed of 100 time series), we also use hierarchical clustering to analyze the time series. As distance for the clustering, we use the Euclidean distance and, for the distance inter-cluster, we consider the average of the points of the clusters. Then, looking at the dendrogram (shown in the picture below), we choose the number of clusters (that is 11) and we apply the algorithm. In the end, we print the time series and the cities divided by clusters.



RESULTS

Using the **k-means and** the **euclidean distance** we obtain 7 clusters. The first thing that we can notice is that there is a cluster quite bigger than all the others (it contains half of the time series). Looking at the time series of the clusters, in all of them, the time series seems to have a shape quite similar.

Instead, using the DTW we obtain 10 clusters. Here the clusters are better balanced because there are not clusters very big and others very small like in the previous case. Looking at the time series of the clusters, they seem to have similar shapes but translated or evolving at slightly different speeds also if some clusters appear to be more confused with respect to the ones obtained with euclidean distance.

Looking at the clusters of cities we notice that many cities put in the same cluster by the k-means with the euclidean distance are put in the same cluster also by the k-means with DTW.

Analyzing the dendrogram for the hierarchical clustering we select a number of clusters equal to 11. We obtain imbalance clusters: for example, there is a cluster of 60 time series and others with only one or 2 time series. Also in this case, even if some clusters are quite big, the time series in the clusters seem to have similar shapes.

All the clusters of time series and of cities are shown in the Time_series_analysis notebook.

So, all the 3 approaches group many cities together (some cities are put together by all the 3 methods) and all the approaches seem to work well. However, hierarchical clustering generates more clusters with only one city that instead we don’t have with the k-means.

6. Conclusions

The analyzed dataset "tennis_matches.csv" was very rich in information but it contained many errors.

In the understanding phase we tried to understand how the data were structured and their meaning. The errors have been identified and corrected using both statistical and graphical tools and external datasets identified on the network (e.g. the IOC dataset).

Having no knowledge about tennis, we had to study the meaning of the attributes, for example how the score and breakpoints work, in order to validate them and complete the missing data in the preparation phase. We also tried to imagine what are the attributes that can characterize a player, aimed at creating the dataset of player profiles. These attributes have been extracted and calculated from the matches dataset by aggregating the values on the players dataset.

The profiles have been grouped by similarity, based on the attributes of the dataset, using various clustering algorithms. We have seen that K-Means is the algorithm that executes fairly balanced clusters.

About the classifiers, we have noticed that most of them have good performances, such as DT, RF, AB, and NN. Using the weighted dataset the SVM had a significant improvement in performance with respect to underfitting on the original dataset. Using oversampled dataset most of the models slightly increase their accuracy.

For the last optional TASK, we decided to choose the one related to the time series because, for the various members of the group, it seemed the most interesting and we liked to deepen the subject. The results obtained seem very good to us, in particular in the plots visible on the relative Jupyter notebook (not shown in this document for reasons of space) it is possible to see how the grouped cities follow the same trend of increase and decrease of the average temperatures.