Data Mining Project Report

Lorenzo Santino - 605004 - l.santino1@studenti.unipi.it - MSc in Artificial Intelligence

Giuliano Galloppi - 646443 - g.galloppi@studenti.unipi.it - MSc in Big Data Technologies

Group 20

Data Mining course (309AA) - 9 CFU - A.Y. 2021/2022

Instructor: Professor Anna Monreale

Date: 15/11/2021

Abstract

In this report will describe the project we carried out for the Data Mining course. The project structure consist of 3 main task. In the first task we analyze a medium-large tennis matches dataset through the data understanding and preparation analysis to achieve as goal the creation of a player profile. In the second task we compare different clustering method on the players profile dataset of the previous task, comparing and discussing it. In the last task we perform a predictive analysis using different models of classification in order to predict and assign for each player, belong to the previous dataset task, a label that defines if a player is a high ranked or low ranked by exploiting the feature related to the rank of the players.

1 Task 1: Data Understanding and Preparation

1.1 Description of the dataset

We are going to analyse 3 dataset:

- *tennis_matches.csv*: contains information about the tennis matches from the 2016 into the 2021. The dataset contains **186128** record and **50** attributes.
- female.csv: contains the personal data (name and surname) of female players. The dataset contains **5208** records and **2** attributes.
- male.csv: contains the personal data (name and surname) of male players. The dataset contains **46172** records and **2** attributes.

1.2 Data Quality

In this phase we had to correct and improve the quality of data contained in the dataset. First of all, in each dataset we dropped the *duplicate records*, if they exists, that maybe could be inserted for some error of copy or entry. We discarded:

- **309** (**0.17%**) duplicate records from the Tennis Matches dataset (without taking into account the first column (record ID)).
- **524** (**0.95%**) duplicate records from the Male Players dataset.
- **511** (**1.11%**) duplicate records from the Female Players dataset.

Regardless the semantic of the attributes, we need a domain knowledge expert (thanks Anna F.) to understand the data in the real word. Now we can start to check, for each data set, the *syntactic accuracy* and the *semantic accuracy* of attributes.

1.2.1 Male and Female dataset

In the **Male and Female dataset**, because the attributes are the same, we treat them together. The dataset have only 2 string attributes, *name* and *surname*, that represent the name and surname of the (male or female) player.

We start to discard all the *null records* (name or surname column) because we don't really care about the players in this dataset; we just need the players witch are present in the tennis_match dataset. For the male dataset we discarded **125** (**0.23%**) records, and for the female dataset we discarded **1572** (**3.44%**) records. About the *syntax accuracy* we made the assumption that are accepted as syntax correct just the string having a length *grater than 1*, this make sense and allow us to eliminate record certainly present for a mistake. We have discarded **1009** (**1.83%**) records in the male dataset and **3726** (**8.07%**) records in the female dataset.

Summarizing, after this data cleaning analysis, we discarded **1134** (**2.05%**) records for the male dataset and **5298** (**11.47%**) records for the female dataset.

1.2.2 Tennis Match dataset

For the ***tennis_match*** dataset we have to analyze **50** attributes (a lot of them are grouped together for winner and loser players). Since the high number of attributes, a more accurate and necessary analysis have done first on the **core attributes** of the tennis match dataset. We define as ***core attribute*** (marked as *) the attribute such that if they missing we cannot determine a tennis match. This kind of attributes have allowed us to avoid useless dropping of records and to retrieve as much as possible values necessary for the following creation of player profile. Note: the analyse are done in a cascade way, therefore the results are affected by the results for the previous attributes.

The first unnamed attribute was dropped because it's useless: it counts only the number of records.

- *winner_name* (*) - *loser_name* (*) (String):
  - Syntax: not-numeric string.
  - Semantic: represent the name and surname of the winner or loser player.

We discovered **27** record that contains a null value for the winner name attribute and **20** records for the loser name attribute. Both winner and loser name are core attributes that describes a tennis match then, for this reason, we had use a ***data driven approach*** to find these missing values. To discover this information, we need to filter the match where the attributes tourney name and tourney date are not missing, otherwise we cannot exactly determinate the match. After this vertical filter, we have automatically discarded **19** record that have a null winner name attribute and **14** record that have a null loser name.

As next step, we create a 'view' of these attributes useful to retrieve the match (figure 1). Then it's possible to retrieve the information about matches from a trusted source of information: we use WTA Tennis site since we could isolate information that allow us to get the other one that are missing values.

After this process for the winner_name attribute we have updated **5** record and so we have discarded **22** record that was irrecoverable; instead for the loser_name attribute we started having **20** null records of which we have updated **5** and discarded **15**.

	tourney_name	tourney_date	winner_name	winner_ioc	loser_name	loser_ioc
186078	Taipei	20171113.0	Veronika Kudermetova	NaN	NaN	NaN
186083	Taipei	20171113.0	Priscilla Hon	AUS	NaN	RUS
186103	Taipei	20171113.0	Belinda Bencic	NaN	NaN	NED
186116	Taipei	20171113.0	Ingrid Neel	NaN	NaN	CAN
186127	Taipei	20171113.0	Priscilla Hon	AUS	NaN	USA
186099	Taipei	20171113.0	Belinda Bencic	SUI	NaN	SUI

Figure 1: Example of view of attributes needed for data driven

We have extracted two new attributes that describe the gender of the players from the winner_name and loser_name attribute. To determine the correct gender, we have searched each record in the winner and loser name inside the the male and female dataset, and assign the correct gender (Male, Female) doing also this assumption: since the tournament have a unique gender for the players, we can retrieve the opposite gender, using a function that we have defined, looking the winner or the loser (not null) sex.

- *winner_sex* - *loser_sex* (String):
  - Syntax: String that could have only the values "Male/Female".
  - Semantic: represent the gender of the winner/loser player.

We also made a statistics analysis on the gender distribution in the tennis matches dataset (fig. 2) and we discover that are **60657** (**32.66** %) of Male player and **125086** (**67.34** %) of Female player. Because the male and female tennis stats

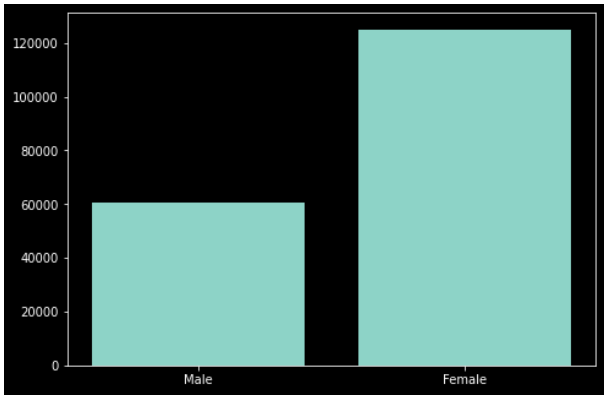


Figure 2: Male and Female distribution in the tennis matches dataset

could be different, and because the gender class populations are not balanced, we make different analysis on other attribute (for instance, the outliers detection).

- *tourney_date* (*) (Float at loading, next Datetime):
  - Syntax: Numerical attribute, composed by eight characters YYYYMMDD, four for year, two for month and the last two for day.
  - Semantic: represent the date of the tournament.

  We discovered **3** null record for the tourney_date attribute. After made a manually check on these few records, we have discovered how they are just simple duplicate record (in the data set exist record that have the same information and have also the null data information). Then we discard these **3** record. As last step, we have casted the data type into 'datetime' that was much more suitable.

- *tourney_name* (*) (String):
  - Syntax: not-numeric string.
  - Semantic: represent the name of the tourney.

  We discovered **2** null record for the tourney_name attribute. We observed that the two records have the same tourney_revenue attribute, that is unique for each tournament. Then, we made a cross-reference with this attribute and we have discovered the missing information that has been updated in the dataset.

- *tourney_id* (*) (String):
  - Syntax: string where the first 4 character need to be numeric.
  - Semantic: is a unique identifier for each tournament.

  We discovered **15** null record for the tourney_name attribute. We observed that the null record have the same tourney_date and tourney_name attribute. Then, we made a cross-reference with those attribute and we have discovered the missing information that has been updated in the dataset.

From the analysis made up to this point, we have *discovered knowledge* on the representation of matches: the triple <tourney_id, tourney_name, tourney_date> is a *unique reference* to a match.

- *winner_id - loser_id* (*) (Float at loading, next Integer):
  - Syntax: numerical attribute.
  - Semantic: unique identifier for winner and loser players.

We discovered **15** null record for the winner_id attribute and **7** null record for the loser_id attribute. Because they are unique for each player, we check through the winner_name and loser_name attribute, if the missing information is available in the data set. Whit this check, we have retrieved all the missing information. As last step, we cast the attribute from Float into Integer.

After the clean data analysis made on the core attributes, we drop all *nested duplicate* records: records that have duplicate core attributes. We discard **60** (**0.03** %) records.

- *surface* (String):
  - Syntax: categorical attribute. Can only assume the 'Hard', 'Clay', 'Grass', 'Carpet' values.
  - Semantic: kind of surface where the match is been played.

We discovered **170** null records corresponding to the surface attribute. To handle the missing information, we replaced them with the *mode* ("Hard" value) according to the distribution analysis viewed from the bar plot (fig. 3).

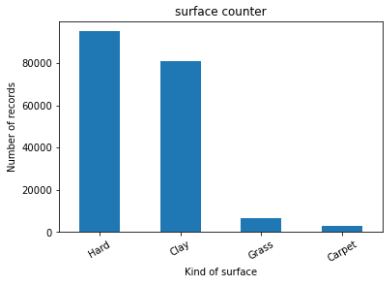


Figure 3: Bar Plot distribution of surface attribute

We also made a syntax check that don't have showed any break-syntax record.

- *draw_size* (Float):
  - Syntax: Numerical attribute.
  - Semantic: is the number of players in the draw

We discovered **9** null records for the draw_size attribute. We observed that the null values have the same tourney_id so we made a cross-reference with this attribute and we have discovered the missing information that has been updated in the dataset.

- *tourney_level* (String):
  - Syntax: ordinal attribute, it can contains only the range of values described in the semantic below.
  - Semantic: is the kind of tournament; it have a specific meaning for women or man. For the man: 'G' = grand slam 'M' = Masters 1000 'A' = other tour-level events, 'C' = Challengers, 'S' = Satellites/ITFs, 'F' = Tour finals and other season-ending events, 'D' = Davis Cup. For the woman: 'P' = Premier, 'PM' = Premier Mandatory, 'I' = International. Integer values describe the prize money (in thousands) of the various levels of ITFs (International Tennis Federation). Other codes, such as 'T1' for Tier I (and so on) are used for older WTA tournament designations. For both: 'D' = Federation/Fed/Billie Jean King Cup/Wightman Cup/Bonne Bell Cup. 'E' = exhibition (events not sanctioned by the tour, though the definitions can be ambiguous), 'J' = juniors, 'T' = team tennis, which does yet appear anywhere in the dataset but will at some point.

  We discovered **6** null record for the tourney_level attribute. We observer how the null record share the same unique triple (tourney_id, tourney_date, tourney_name). Using the previous discovered knowledge, we can update this missing values using the corresponding tourney_level available in the data set.

- *match_num* (Float):
  - Syntax: numerical attribute.
  - Semantic: a match-specific identifier.

  Having as purpose the creation of a *player profile*, the attribute match_num don't give us any relevant information, then we proceed to discard it.

- *winner_entry* - *loser_entry* (String):
  - Syntax: Categorical attribute, it could assume values : 'PR', 'Q', 'WC', 'LL', 'SE', 'ALT', 'SR', 'JE', 'A', 'ITF', 'P', 'I', 'IR', 'JR'. We have noted that exist a lower case value so we converted them to uppercase.
  - Semantic: means the access mode to the tournament for the winner/loser player.

  We discovered respectively **159940** and **141666** null records. For our analysis, to create a player profile, the attribute winner_entry and loser_entry are useless, because it don't give a significant characterization for the player, so we ***dropped*** these attributes.

- *winner_hand* - *loser_hand* (String):
  - Syntax: Categorical attribute. It assume only 'R' = right, 'L' = left, 'U' = unknown values.
  - Semantic: it describe the hand of the winner/loser player.

  We discover respectively **28** and **81** null records for these attributes. We used the winner_id and the loser_id to identify if the player have ever been played any match in the dataset to retrieve the information, if available, about the player hand. In this way excluding the information we have retrieved, only **16** and **41** remaining null records have been updated with 'U' (unknown) value.

- *winner_ht* - *loser_ht* (Float):
  - Syntax: Numerical attribute.
  - Semantic: it describes the height in centimetres of the winner/loser player.

  We discover respectively **136442** and **147411** null records for these attributes. We based the analysis differentiating on gender of the player, exluding outliers, null values were replaced by the mean differentiating on gender of the player. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *winner_ioc* - *loser_ioc* (String):
  - Syntax: Categorical attribute. It can be formed about only three characters. From our analysis we checked this boundary, without find any irregularity.
  - Semantic: it describes the country of the winner.

  We discover respectively **9** and **3** null records for these attributes. Using the *winner_id* and *loser_id* associate to them, they helped us to retrieve, if already exist, the player ioc in the dataset. In this way we have fixed all the null values that we mentioned before. We also made made a syntax check if the len of string is equal to 3.

- *winner_age* - *loser_age* (Float at loading, next Integer):
  - Syntax: Numerical attribute. To cast integer we discover from the analysis that exists null values which we managed in the section below.
  - Semantic: is the age of the winner depending on the date of the tournament.

  We discovered respectively **2835** and **6513** null records for these attributes. We used the winner/loser_id attributes with a cross-reference method to retrieve the missing age of the players, if existent, in the dataset. In this way we left, exluding outliers, only **2826** null record for the winner_age and **2052** for the loser_age that we fill with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *score* (String)
  - Syntax: Categorical attribute. Split by space are indicated the partial result of the match, between brackets there is it's contained tie-break result.
  - Semantic: score of the match, it contains the partial and final results.

  We discover **170** null records for the score attribute and, as goal to create the players profile, we decide to drop it because it dont't give any significant information.

- *best_of* (Float at loading, next Integer)
  - Syntax: Numerical attribute. It can assume only '3' or '5' value, for this reason we casted it to integer.

- Semantic: It describes the number of set played in the match.

We discovered **10** null record for the best_of attribute. All null values have the same tourney_id attribute then, because the best_of attribute is correlated to the tournament, then we can retrieve the missing information and replace them updating the dataset. As last step, we analyze the distribution of the best of category in the data set and we found that only **3173** (**1.71**%) records are `best of 5` and from the real word, we know that a matches played as best of 5 have different stats respect the matches played as best of 3. For this reason we decide to discard all records that are best of 5 and then drop the (now) useless *best_of* attribute.

- *round* (String):
  - Syntax: Categorical attribute, the only value that could assume that describe the round are : 'F', 'SF', 'QF', 'R16', 'R32', 'Q1', 'Q2', 'Q3', 'R64', 'R128' , 'RR', 'BR'.
  - Semantic: it describes the round of the match in tournament that it belongs.

  We discovered **9** null record for the round attribute. We used the data driven approach on it because it's not possible to extract correct information from the dataset. From external source we discovered information and with them we updated the missing values in the dataset.

- *minutes* (Float):
  - Syntax: Categorical attribute. We checked if there was negative values with negative result. There are 81660 records with valid values.
  - Semantic: it describes the length of the match in minutes.

  We discovered **104424** null record for the attribute. We based our on the gender of the player, and also checked if the matches is beed played in a best of 3 or a best of 5 round. Excluding outliers, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *w_ace* - *l_ace* (Float at loading, next Integer):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of ace done by the winner/loser player about the match.

  We discover respectively **103772** and **103772** null records for these attributes. Excluding outliers, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *w_df* - *l_df* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of double fail done by the winner/loser player about the match.

We discover respectively **103775** and **103775** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *w_svpt* - *l_svpt* (Float at loading, Integer next) :
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of point obtained with a service done by the winner/loser player.

  We discover respectively **103774** and **103774** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *w_1stIn* - *l_1stIn* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of first serves done by the winner/loser player.

  We discover respectively **103776** and **103775** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *w_1stWon* - *l_1stWon* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of first-serve points won done by the winner/loser player.

  We discover respectively **103772** and **103775** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *w_2ndWon* - *l_2ndWon* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of second-serves points won done by the winner/loser player.

  We discover respectively **103779** and **103777** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *w_SvGms* - *l_SvGms* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the winner's/loser's number of serve games.

We discover respectively **103772** and **103768** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *w_bpSaved* - *l_bpSaved* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of breakpoints saved by the winner/loser player.

  We discover respectively **103774** and **103773** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *w_bpFaced* - *l_bpFaced* (Float at loading, Integer next):
  - Syntax: Numerical attribute.
  - Semantic: It describes the number of breakpoints faced by the winner/loser player.

  We discover respectively **103774** and **103776** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *winner_rank* - *loser_rank* (Float) :
  - Syntax: Numerical attribute.
  - Semantic: It describes the winner/loser player's ATP or WTA rank, as the tourney date, or the most recent ranking date before the tourney date before the tourney_date

  We discover respectively **19375** and **35236** null records for these attributes. Excluding outliers and differentiating on gender of the player, the remaining null values **8127** and **6613** have been filled with the mean. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *winner_rank_points* - *loser_rank_points* (Float):
  - Syntax: Numerical attribute.
  - Semantic: It describes the winner/loser player's ATP or WTA ranking points.

  We discover respectively **19391** and **35251** null records for these attributes. For our analysis, to create a player profile, the attribute winner entry and loser entry are useless, because it don't give a significant characterization for the player, so we ***dropped*** these attributes.
- *tourney_spectators* (Float at loading, next Integer):
  - Syntax: Numerical attribute. It has to be casted to manage it properly.

- Semantic: It describes the number of spectators at the match.

We discovered **6** null records. From the analysis we observed that these records have the same tourney_id, then we retrieve and replace the missing values in the dataset using a cross-reference with other records. The detected outliers are replaced by the mean, differentiating on the gender of the player.

- *tourney_revenue* (Float):
  - Syntax: Numerical attribute.
  - Semantic: revenue of the tournament which the match belongs.

  We discovered **5** null records. From the analysis we observed that these records have the same *tourney_id*, then we retrieve and replace the missing values in the dataset using a cross-reference with other records. The detected outliers are replaced by the mean, differentiating on the gender of the player.
- *tie-break* (Integer): We have extracted this new attributes that describe if a match that envolves the players have reached a tie-break, retrieved by the score attribute.
  - Syntax: Numerical attribute, it could be only '1' or '0', as a boolean value.
  - Semantic: It indicates if a match went to a tie-break or not.

1.2.3 Outliers detection - Hampel X84

The **Hampel X84** is a outliers detection technique belonging to the category of the statistics-based fitting-distribution parametric technique. The Hampel X84 using the *median* and the *MAD* (Median Absolute Deviation), marks outlier those data points that are more than $\mathbf{1.4826} \cdot \theta$ MADs away from the median, where θ is the number of standard deviations away from the mean.

We apply the Hampel X84 technique every time we need to identify the outliers. In detail, after choose an appropriate input parameter θ, we define a left and right bound for the considered attribute. All data outside the bounds are marked as outlier.

1.3 "Player Profile" dataset

A new dataset was created to represent the profile of each player retrieved by previous dataset described.

1.3.1 The semantic of the "Player Profile" dataset

The following list describes the **54** attributes computed for the "Player Profile" dataset, included interesting *features* for describe the behaviour of player derived from the matches:

- **id**: id of the player used in the matches.

- **name**: name and surname of the tennis player.
- **sex**: the gender of the player.
- **ioc**: country code of the player (composed by three character).
- **age**: age of the player(estimate by a mean of his ages in the different tournament).
- **ht**: **mean** number of the height of a player during the period of matches.
- **hand**: serving hand of the player.
- **ace_sum**: total number of aces w.r.t. the matches that he played.
- **df_sum**: total number of double faults.
- **svpt_sum**: total number of serve points.
- **1stIn_sum**: total number of first serve made.
- **1stWon_sum**: total number of first-serve points won.
- **2stWon_sum**: total number of second-serve points won.
- **SvGms_sum**: total number of serve games.
- **bpSaved_sum**: total number of breakpoints saved.
- **bpFaced_sum**: total number of breakpoints faced.
- **tie-break_sum**: total number of tie-break played by the player.
- **ace_mean**: mean number of aces w.r.t. tha matches that the player patecipated.
- **df_mean**: mean number of double faults.
- **svpt_mean**: mean number of serve points.
- **1stIn_mean**: mean number of first serve made.
- **1stWon_mean**: mean number of first-serve points won.
- **2stWon_mean**: mean number of second-serve points won.
- **SvGms_mean**: mean number of serve games.
- **bpSaved_mean**: mean number of breakpoints saved.
- **bpFaced_mean**: mean number of breakpoints faced.
- **tie-break_mean**: mean number of tie-break played by the player.
- **tot_win**: total of matches winned by the player.

- **tot_win**: total of matches losed by the player.
- **tot_games**: total of matches played by the player.
- **winrate**: mean of winned matches.
- **ace_on_service(%)**: percentage of *total aces* related to the number of first serve made.
- **df_on_service(%)**: percentage of *total df* related to the number of first serve made.
- **point_on_first_service(%)**: percentage of *total 1stWon* related to the total number of 1stIn.
- **tie-break_rate(%)**: percentage of *total tie-break* w.r.t. the number of total games played by the player.
- **Hard_win**: total matches winned by the player on *hard* surface.
- **Clay_win**: total matches winned by the player on *clay* surface.
- **Grass_win**: total matches winned by the player on *grass* surface.
- **Carpet_win**: total matches winned by the player on *carpet* surface.
- **Hard_los**: total matches losed by the player on *hard* surface.
- **Clay_los**: total matches losed by the player on *clay* surface.
- **Grass_los**: total matches losed by the player on *grass* surface.
- **Carpet_los**: total matches losed by the player on *carpet* surface.
- **Hard_win(%)**: percentage of total matches winned by the player on *hard* surface respect to total matches played by the player.
- **Clay_win(%)**: percentage of total matches winned by the player on *clay* surface respect to total matches played by the player.
- **Grass_win(%)**: percentage of total matches winned by the player on *grass* surface respect to total matches played by the player.
- **Carpet_win(%)**: percentage of total matches winned by the player on *carpet* surface respect to total matches played by the player.
- **Hard_los(%)**: percentage of total matches losed by the player on *hard* surface respect to total matches played by the player.
- **Clay_los(%)**: percentage of total matches losed by the player on *clay* surface respect to total matches played by the player.
- **Grass_los(%)**: percentage of total matches losed by the player on *grass* surface respect to total matches played by the player.

- **Carpet_los(%)**: percentage of total matches losed by the player on *carpet* surface respect to total matches played by the player.
- **Best_Surface**: surface on which the player has won the most games.
- **Worste_Surface:** surface on which the player has loss the most games.

1.3.2 Correlations within the Players Profile

With the libraries used until here we have could create the correlation matrix (which excludes all the categorical attributes) to measure the correlation between attributes.

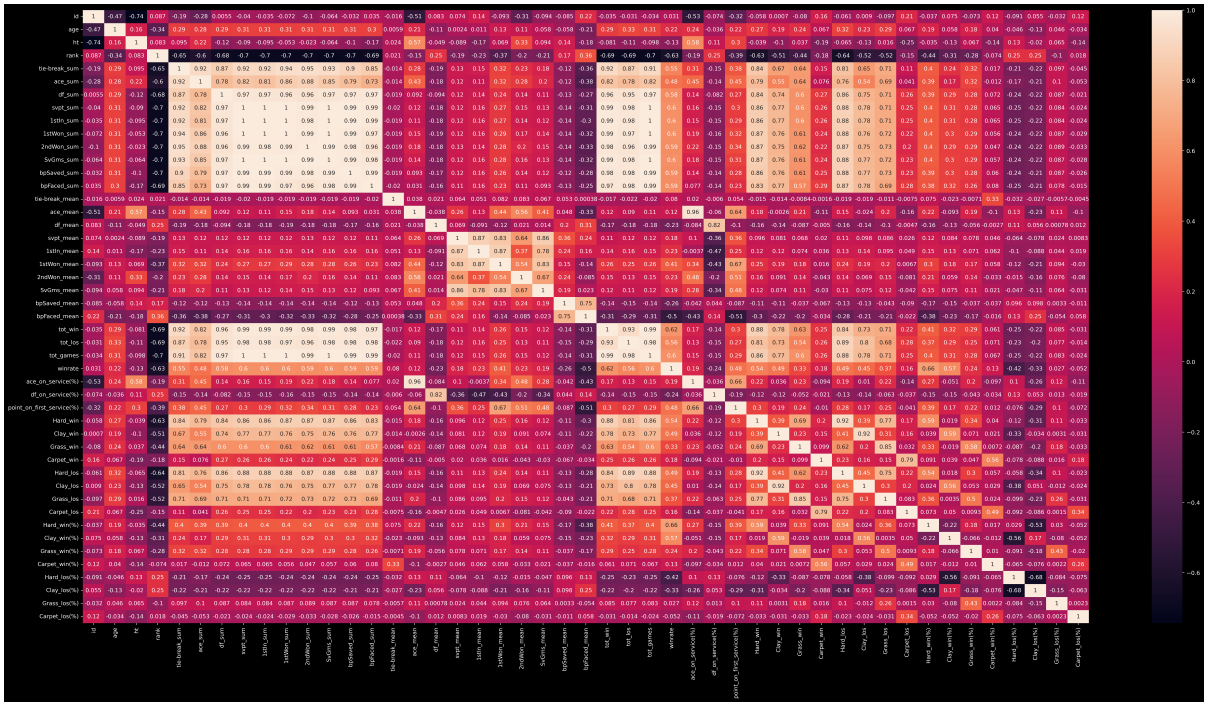


Figure 4: Correlation Matrix of *players_profile's* dataset attributes

From the correlation matrix we discovered some attributes that are potentially redundant. Them over the correlation threshold (equals to **0.85** to not lose too much information) are **24** over the total of **54**.

We decided to drop the following columns: "ace_sum", "df_sum", "svpt_sum", "1stIn_sum", "1stWon_sum", "2ndWon_sum", "SvGms_sum", "bpSaved_sum", "bpFaced_sum", "tie-break_sum", "tot_win", "tot_los", "Hard_win", "Clay_win", "Grass_win", "Carpet_win", "Hard_los", "Clay_los", "Grass_los", "Carpet_los", "ace_mean", "svpt_mean", "1stWon_mean", "tie-break_mean".

The new correlation matrix is the following:

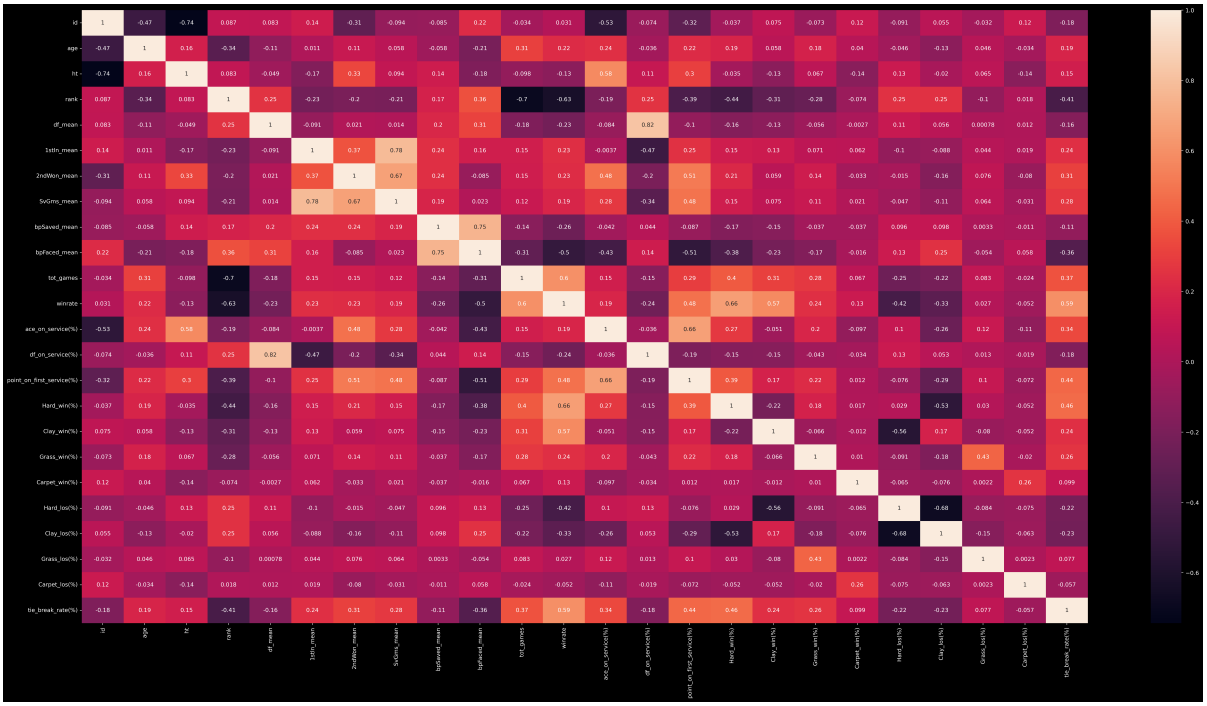


Figure 5: Correlation Matrix of *Final players_profile's* dataset attributes

After made all the previous analysis, we create a new dataset, **players profile** that contains **4370** records and **30** attribute.

2 Task 2: Clustering Analysis

2.1 Preprocessing: Feature Reduction

As first step, we proceed to separate the numerical features and the categorical features. Every clustering algorithm that will be applied, need *normalized data*, then we proceed to normalize the input numerical attribute using the **Z-Score normalization** because is stronger to handle outliers (w.r.t the MinMax normalization). To avoid the *curse of dimensionality* trap we need to reduce the numbers of features given to input to the clustering algorithms. For this purpose, we make a sub selection of the features, going to select the more representative players profile statistical attributes. They are the *rank*, *2ndWon_mean*, *ace_on_service(%)* and *point_on_first_service(%)*. This specific feature try to capture the concept of "good" tennis player, using the semantic of the real word.

In order to reduce more the dimensionality of the input space, we apply a **PCA** on this subset of features. To choose the optimal number of principal components, we apply a golden rule which specify that the first n principal components guarantee at least the

80% of the variance. We can easily observer in the following graph (fig. 6), how the first 2 principal components express 80% of the variance; then we pick the first **2** principal components.

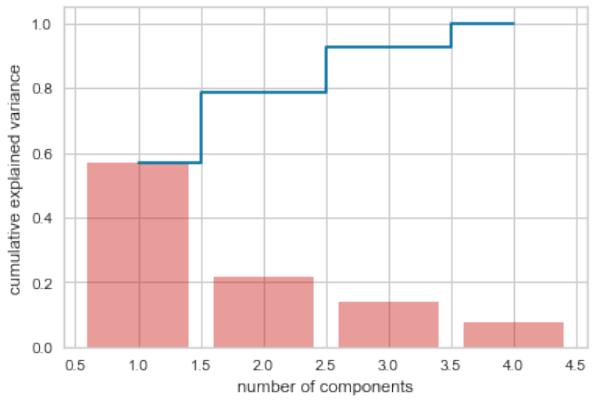


Figure 6: Explained variance ratio

2.2 K-means Clustering

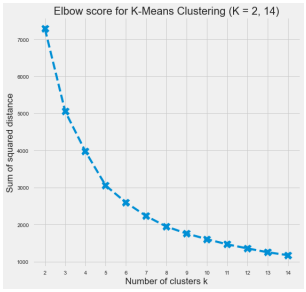
The K-means is a **partitional clustering** that has **center-based** clusters. This algorithm allows to divide a set of objects into **k** groups on the basis of their attributes.

2.2.1 Choosing *'K'* Value

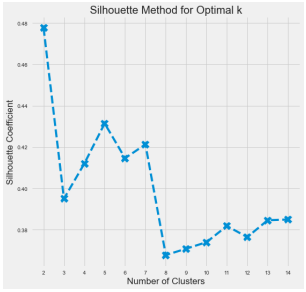
As we said above, the 'k' is the fundamental parameter that defines how many clusters will be the result of the algorithm. The main goal is to find the best value of k by running multiple executions. We have chosen to begin from an initial value of 2 and we continued with the execution of the k-means algorithm until reaching 14 which is the value chosen as the max boundary for k.

After we fit the model, we apply the **elbow method** heuristic looking the plot of the SSE measure as the clusters vary and we pick the optimal k observing the *elbow* point of the curve (fig 7 (a)). We candidate $\mathbf{k = 5}$ as optimal k parameter. We also compute the **silhouette score**: a indicator of both separation and cohesion among clusters. Observing the *silhouette* graph (fig 7 (b)) we candidate as optimal $\mathbf{k = 2}$ (maximum point of the plot) and $\mathbf{k = 5}$ (local maximum). The last method for choose k, is observing the *dendogram* created by the **Ward's method** fig 7 (c)), and looking how many main cluster are been created. We candidate as optimal $\mathbf{k = 2}$.

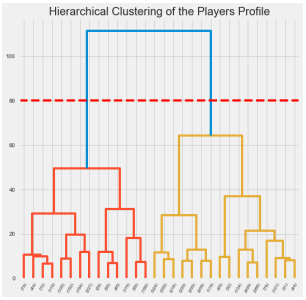
We use the candidate k input to perform a new k-means run (with more initialization instance) and we compare the the new models using the *SSE metrics*, the *Davies Bouldini* index, the *Silhouette score* and the *Calinski-Harabasz* index. After this analysis, we choose $\mathbf{k = 2}$ as optimal input 'k' parameter for the k-means clustering algorithm.



(a) Elbow Method 1



(b) Silhouette Method 2

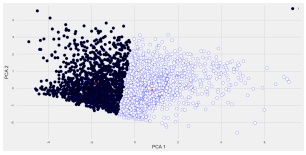


(c) Ward's Method 2

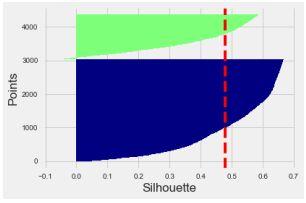
Figure 7: K-Means metrics

2.2.2 Analysis of the K-means clustering result

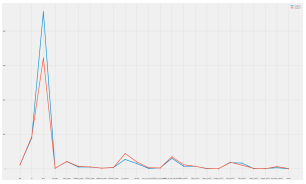
The k-mean algorithm have produced **2 clusters** with dimension **3034** and **1336**. We can also observer (fig 8 (a)) how the cluster have partitioned and well separated the input data. The red point are the centroids of the clusters. We can see the plot of the Silhouette score ((fig 8 (b))), where each different color (green and blue) represents a different cluster, and the dotted red line is the average silhouette score. In particular we have that Cluster 0 has a silhouette score of **0.51** and **0.38** for Cluster 1, with an overall average silhouette score of **0.47**. We can observer that a small fraction of points in Cluster 0 have a negative value but overall the Silhouette metric suggests a good clustering. As last observation, we show ((fig 8 (c))) the average values for each attribute divided by the clusters.



(a) Clusters



(b) Silhouette Score



(c) Average Values 2

Figure 8: Clustering results

Through this clustering partition, we can observer how the cluster have different mean attribute. In particular the players belonging in the Cluster 1, have a lesser rank and greater numbers of matches played. Also, the Cluster 1 show a greater win-rate and percentage of ace on the first service and ace on the first service, respect to the players in the Cluster 0. For this reason, we think the players in Cluster 1 are better tennis player respect to players in the Cluster 0, regardless the rank.

2.3 Density-Based Clustering

The DBSCAN clustering algorithm it's based on the data density in the dataspace, it is particularly effective for clusters of particular shape. It have used the same normalized dataset of K-Means. As input it has:

- *eps*: The maximum distance between two samples for them to be considered as in the same neighborhood.
- *min_samples*: the minimum number of points needed to create a cluster

We focused to understand which was the best combination for these parameters (*eps* and *number of min_sample/neighbor* using the knee method with the aim to define next the 'clusters'.

We first have derived plots from the distance computed about of the 'K-th neighbour, in this way we could find the best value of K and have a stime for the eps value. Then we

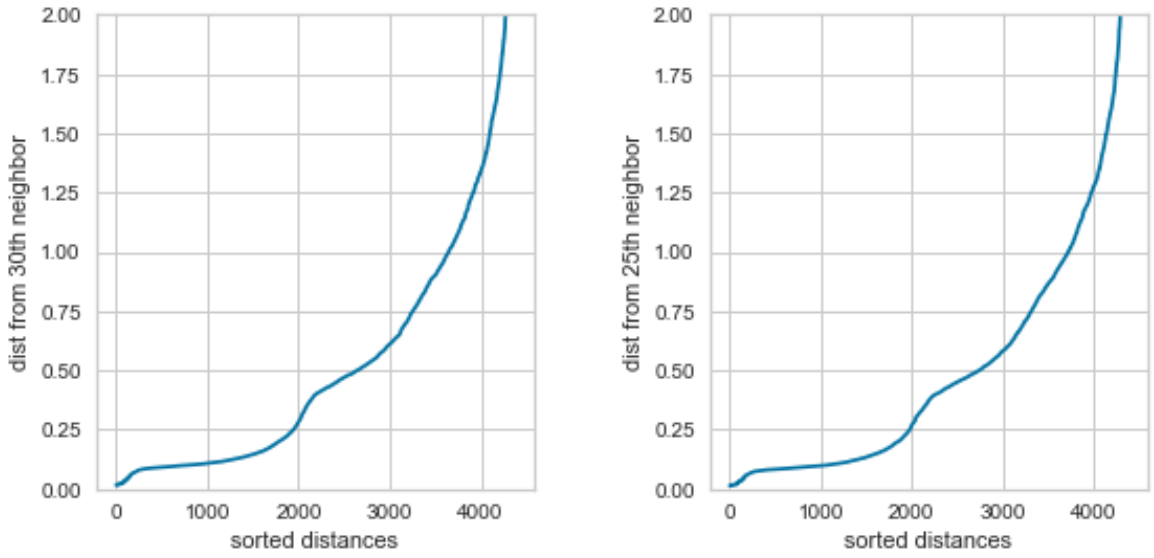


Figure 9: Curve of the distance regardless of the value of K.

create plots experimenting different values for k to confirm which are the best associations for different values of k and eps parameters fixing the first parameter and derived the second as a function of it, and then vice versa, in order to have a more stable and precise estimation.

To establish which was the optimal combination of the aforementioned parameters we execute many times the algorithm, getting different candidate combinations stored in a table. From their analysis, including influential factors such as noise,

After many runs of the algorithm, we established what was the optimal configuration of the two parameters also by computing and analyzing other interesting and influential factors, such as: the amount of noise detected, the number of clusters found and the population of each one.

Looking at the process and noticed that the result was quiet unbalanced, despite how even the slightest change in values can drastically change the configuration of the results, but not improving their performances.

As a final result we choose between the candidates, most similar to each other, the following combination of values: ***30*** for the number of neighbors and ***0.1*** for the eps parameter.

2.3.1 Result analysis and conclusions

We give an account of an example about barplot describing the clustering result, making clearly visible the number of cluster detected, having strong differences in the number of populations.

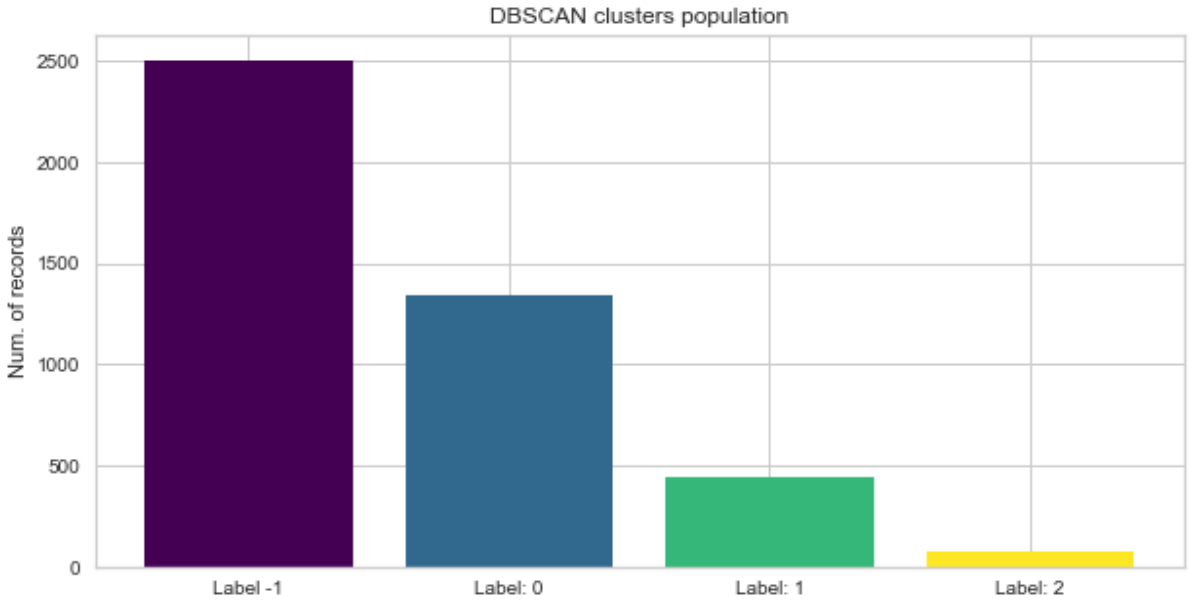


Figure 10: DBSCAN Clustering result on *eps = 0.1 - min_sample = 30*

So we had proceed to analyse them in order to interpret and understand how the clustering process has been performed (i.e. which criteria has followed to group different records).

Here is showed them distribution w.r.t. all the indicators of each single cluster.

The result of this clustering algorithm applied to this dataset is very unbalanced, and different choices of the parameters do not improve the situation, it make very difficult

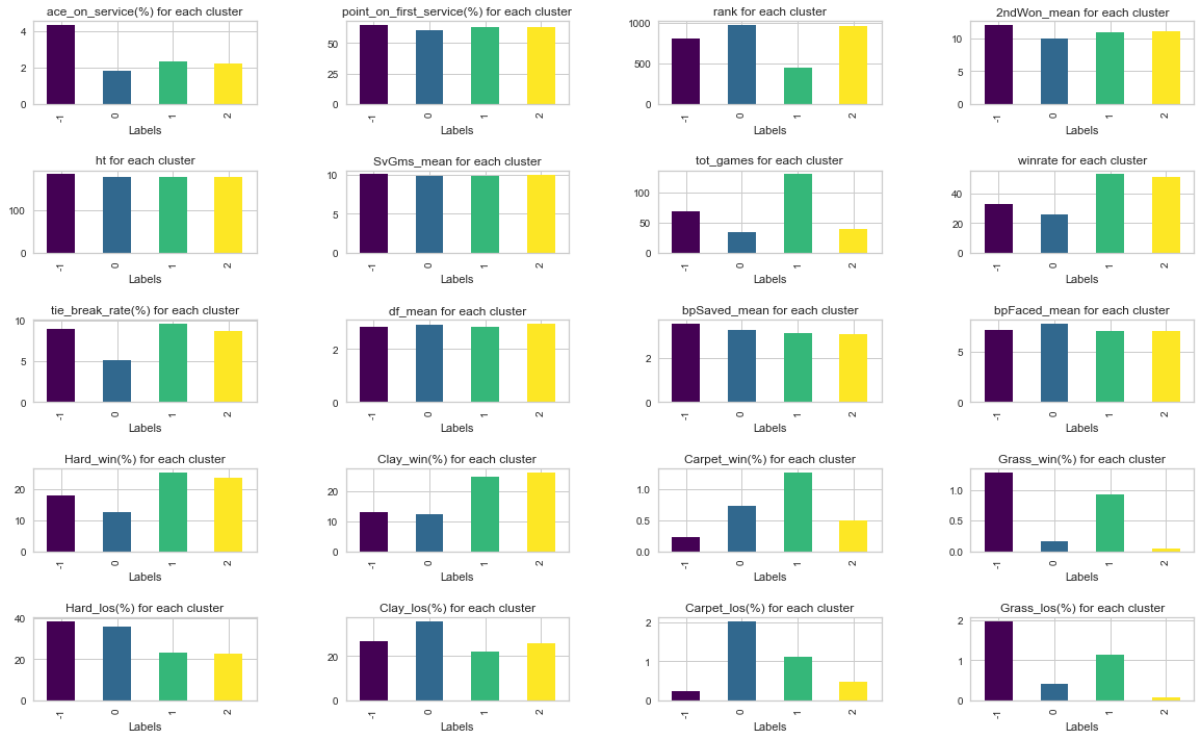


Figure 11: Bar Plot analysis of population belonging to the clusters

to esitmate good partition into clusters so that it allows us to define the labels well. Also different choices of the parameters do not improve the situation, some even make it worse.

2.4 Hierarchical Clustering

It is a type of clustering that produces a set of "nested clusters" or clusters grafted as *Hierarchical Trees* and it's useful when having an nested structure among the cluster is a necessity. The hierarchy of clusters can be visualized with a *dendrogram*: a tree structure where the horizontal lines represent the merging of two nodes, or clusters, and the vertical line provides information regarding the distance (or similarity) between said elements.

We used the *Agglomerative approach* that starts from clusters composed by single elements and step by step these ones will be merged with the nearest clusters. About this approach, on the Players Profile dataset that we had derived before, we have plotted and analyzed the four main *"linkage methods"* (MIN/MAX/AVG/WARD that are described in the related notebook).

The previous and the next figure are described using the parameters included in the Scipy dendrogram function, like the 'color_threshold'.

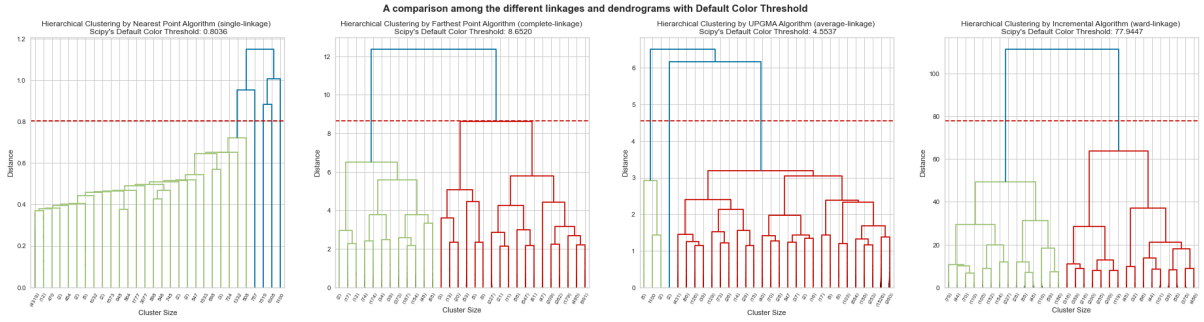


Figure 12: Dendrogram showing comparison among the different linkages

The optimal cut for each dendogram was retrieved by a function (provided from a web research), that iteratively cutting the tree at a height increased at each step. Each cut returns the number of clusters that are identified below the cut on the structure itself:

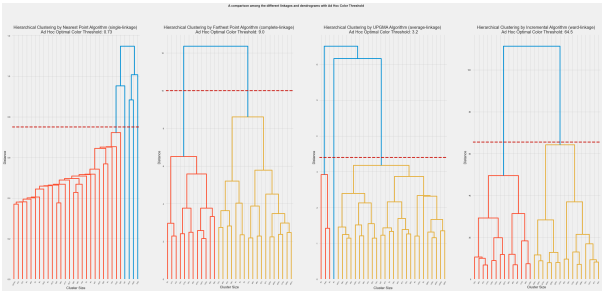


Figure 13: Resulting Dendrogram comparison among the different linkages and with specific Threshold

The clusterings for each of these linkages were computed by using their optimal cuts as indices of the ideal number of clusters. These dendrogram seemed give not so differences by the new cuts compared to the previous dendrogram.

As final step, each of the clustering were evaluated using both well-known internal metrics of Davies_Bouldini, Silhouette, Calinski_Harabasz and particulary the Cophenetic Correlation Coefficient. According to results of these metrics, the clusters from the *average linkage* with **copernich 0.754795** should be the best with the dataset.

2.5 Final evaluation:

After the discussion of three clustering approachs, the clustering K-Means fits better for our dataset according their results.

3 Task 3: Classification

The main objective for this task is to perform a predictive analysis in order to classify each player with a label that defines if s(he) is a high ranked player or a low ranked player.

To realize this task we have executed and compared different classifiers on the dataset, on which we have discretized their categorical feature in a pre-processing phase. The classifier are: ***SVM, Decision Tree, Random Forest, KNN, Naive Bayes, AdaBoost and Neural Network***.

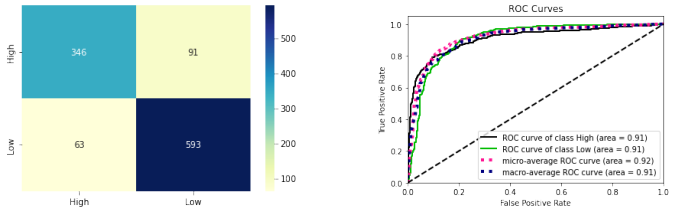
It was used the 'rank' feature with the 'quantile' function at **0.40** to compute a threshold that categorize the high and low rank class, defined as:

- ***High***: if the player's rank is less than the threshold, because from semantics we know that less is the rank, highest is the position of the player.
- ***Low***: Otherwise

After the division of data into train and test sets, we passed to define and fit all the models, and finally to performe a prediction both on train and test set.

3.1 SVM

SVM is a classifier that searches for an hyperplane that can linearly separate the data points, possibly in a transformed space, in the non-separable case. We use a Linear Support Vector Classifier and, after the grid search, we found that the best value for the regularization parameter is $\mathbf{C = 100}$.



(a) Confusion Matrix 1 (b) ROC Curve 2

Figure 14: K-Means metrics

In the Figure above is reported the confusion matrix and the ROC curve related to the results on the test set. we have have for the ***High*** precision = 0.84 and recall = 0.83, instead for ***Low*** precision = 0.89 and recall = 0.89.

3.2 Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. They correct decision trees' habit of overfitting to their training set.

Also here we combine the classifier with a grid search, found as best parameters:

- 'min_samples_split': 2,
- 'min_samples_leaf': 1,
- 'max_features': 5,
- 'max_depth': 3,
- 'criterion': 'gini',
- 'bootstrap': False

In the Figure below is reported the confusion matrix related to the results on the test set. we have have for the ***High*** precision = 0.86 and recall = 0.75 , instead for ***Low*** precision = 0.85 and recall = 0.92.

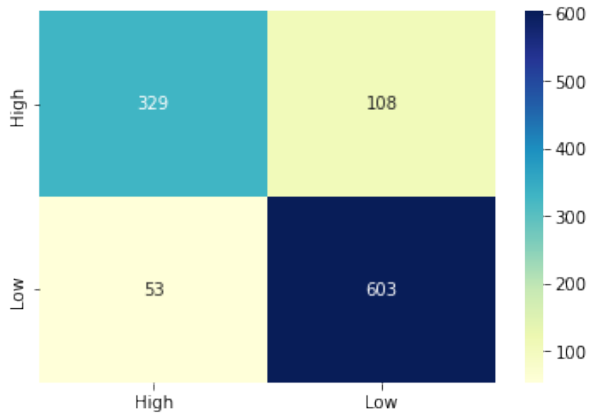


Figure 15: Confusion Matrix Decision Tree Classifier

3.3 Decision Tree

A **Decision Tree** model learns some decision rules from the training data, in order to grow a tree that is able to predict the class label for our test points. The best configuration found is:

- 'splitter': 'random',
- 'min_samples_split': 5,
- 'min_samples_leaf': 15,
- 'max_features': None,
- 'max_depth': 7,
- 'criterion': 'entropy'

In the Figure below is reported the confusion matrix related to the results on the test set. We have have for the ***High*** precision = 0.86 and recall = 0.75 , instead for ***Low*** precision = 0.85 and recall = 0.92.

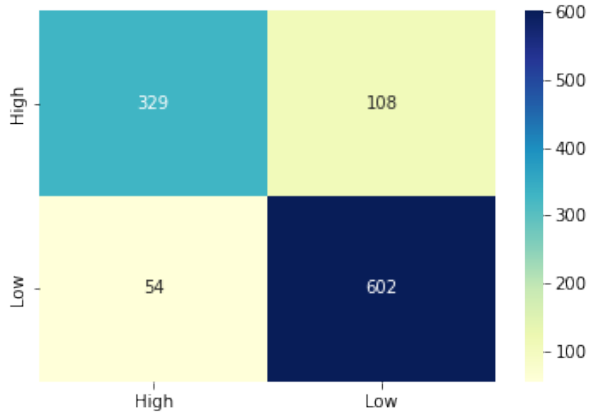


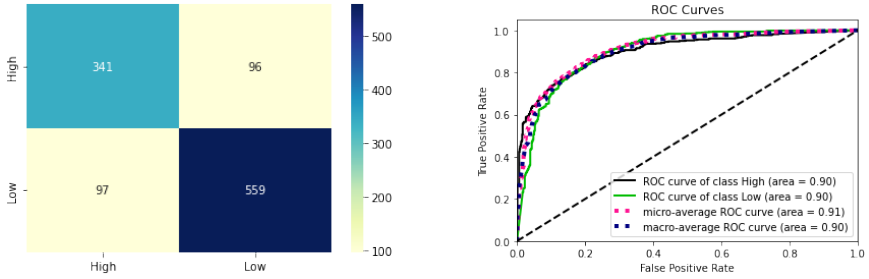
Figure 16: Confusion Matrix Decision Tree Classifier

3.4 KNN

The **K-Nearest Neighbors** model is an instance-based classifier, that, for each record, selects the class label based on the majority of its nearest neighbors. The grid search associated to the classifier get as the best configuration

- 'estimator__n_neighbors': 2
- 'estimator__weights': 'uniform'

In the Figure below is reported the confusion matrix related to the results on the test set. We have have for the ***High*** precision = 0.78 and recall = 0.78 , instead for ***Low*** precision = 0.85 and recall = 0.85. On the other hand, for the training set the model get 1 as the result for **Recall** and **Precision**.



(a) MATRIX (b) ROC Score

Figure 17: KNN Confusion Matrix and ROC Curve

3.5 Conclusions

To conclude, the model that performs the best is the SVM, followed by Random Forest and Decision Tree having a very high accuracy. The table below reports all the accuracy values related to the used models, underlining the little differences between the SVM and the Random Forest.

Model	SVM	RF	DT	KNN	NN	GNB
Accuracy	0.859	0.852	0.851	0.823	0.796	0.763