Data Mining 21/22 Project Report

Marco Costa
MSc in ICT Solutions Architect
m.costa22@studenti.unipi.it
545144

Luca Franceschi
MSc in Artificial Intelligence
l.franceschi5@studenti.unipi.it
544918

Alessandro Sardelli
MSc in Artificial Intelligence
a.sardelli3@studenti.unipi.it
561158

Contents

1 Task 1: Data Understanding and Data Preparation

1.1 Data Understanding

The dataset `tennis_matches` appears as a modest collection of tennis matches played between 2016 and 2021 all around the world. Each record contains information about a certain match, its players, and the tournament in which it was held; because of this, we expect a large amount of knowledge about players and tourneys to be redundant and possibly inconsistent. Moreover, about 29% of values are not reported: at best, these values would take part of the redundant data and would not introduce any additional knowledge, while in the worst case the dataset would be vastly incomplete. The dataset is also augmented with two external lists `male_players` and `female_players` that associate various tennis players to their gender, although correspondence between these players and the players in the dataset is not guaranteed.

1.1.1 Dataset features

We begun by analysing each of the individual attributes of the dataset, in order to understand their semantic role and the relations that we would expect to observe between them. We first noticed that four of them did not appear to be mentioned in the provided description; after a quick glance over the records, we found the followings:

`score` : the match result in the international tennis scoring system format, including the exact tiebreak outcome and whether a match ended up by retirement

`round` : the round of the match in the tournament, where `F, SF, QF` indicates the obvious *final, semi-final* and *quarterfinal*, while `Q1, Q2, Q3` and `R32, R64, R124` both indicate the same *previous* tournament phases in a different format

`tourney_spectators` : a unique value for each *tourney_id* indicating the average number of spectators for the entire tournament

`tourney_revenue` : a unique value indicating the total incoming of the tournament

At that point, the full understanding of all the attributes allowed us to dive through the actual records as to discover properties and quality issues of the data.

1.1.2 Data statistics

As a first step, we wanted to extract some statistics about players, matches and tourneys from the dataset records, as to compare them to real-world trends and expectations; this allowed us to acknowledge data quality issues such as noise/outliers occurrence and presence of any bias towards a certain type of records.

Prior to any kind of analysis, the dataset has been cleaned up from over 15000 duplicate records – about 0.16% of the dataset; it is worth noting however that duplicate information might still be present, as we cannot detect them before an appropriate data cleaning and preparation phase.

Here below are the most relevant results we observed:

- The ratio between the number of tourneys that are disputed on different court surfaces reflects what we would expect based on their level, meaning that the dataset does not seem to be biased towards tourneys held on a specific type of court. This is depicted in the bar chart of Figure 1, where we can note for example that the *Grand Slam* tournaments `G` are properly distributed between $\frac{1}{4}$ matches played on clay, $\frac{1}{4}$ played on grass and $\frac{2}{4}$ played on hard courts.

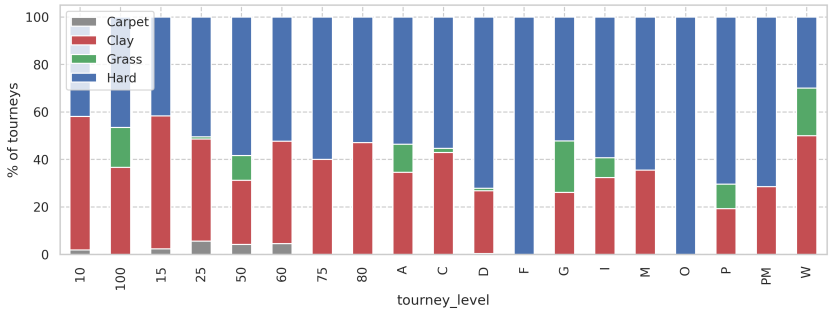


Figure 1: Comparison between the number of tourneys that have been played on a specific tennis court surface, grouped by their respective level

- The boxplot shown in Figure 2 would have allowed us to check whether the dataset typically contains the tournaments in their entirety, i.e. complete in all of their matches; in order to do this, the box plot shows the distribution of the number of matches found in the dataset for the same tourney, differentiating between draw sizes; what we needed to make sure of, is that each tourney is typically composed by a number of matches equals to the number of participants `draw_size` minus 1. We can see however that the obtained visualization looks quite different from expectations; after a closer look over the dataset records, we found that:
  - there exists tournaments in which contenders may face each other multiple times, e.g. in the *Davis Cup*;
  - tournaments with a large number of players include preliminary qualification matches;
  - without an appropriate data preparation phase, we still have single `tourney_id`s identifying both male and female matches of *Grand Slam* tournaments, with relative `draw_size` referring to the participants of just a single gender;
  - these last two points may also occur together.

  Because of these reasons, we concluded that the dataset fails to properly represent the dynamics and the format of different tournaments, thus preventing us from addressing our initial concern.

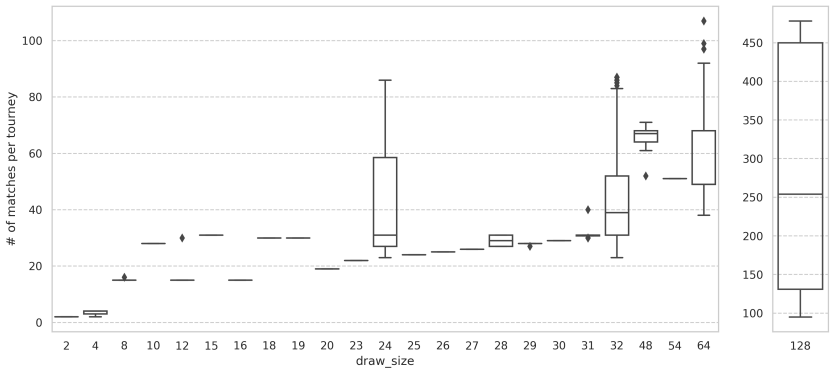


Figure 2: Typical number of recorded matches for the same ⟨`tourney_id`, `tourney_level`⟩, grouped by different `draw_size`s

- The number of matches recorded in the dataset per year appears to be more or less stationary between 2016 and 2019 according to the plot shown in Figure 3; we noted however that the dataset presents a drastic decrease in the year 2020: that intuitively correspond to the enactment of the first major social distancing measures to address the *COVID-19* pandemic; it's also worth noting that the pandemic affected the year 2021 too, although an inferior number of matches is also to be expected due to the dataset being formed of matches prior to September 2021. Lastly, the dataset also seemed to be balanced in terms of number of matches recorded per month, which appear to be in line with the typical annual schedule of tournaments;

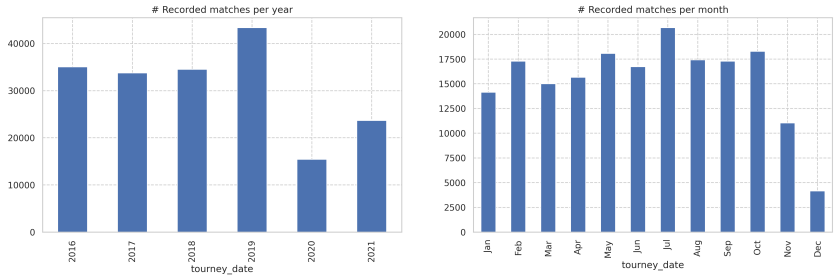


Figure 3: Number of recorded matches per year (left) and per month (right)

- As shown in the plots of Figure 4, we found:
  - the existence of a pretty strong linear relationship between the number of spectators of a tourney and its revenue, with a sample Pearson's correlation of about 0.94
  - a relation between the level of the tourney and both the number of its spectators and revenue

  we also got some hints about possible outliers, that may or may not be notable exceptions of tourneys with an unexpected outcome

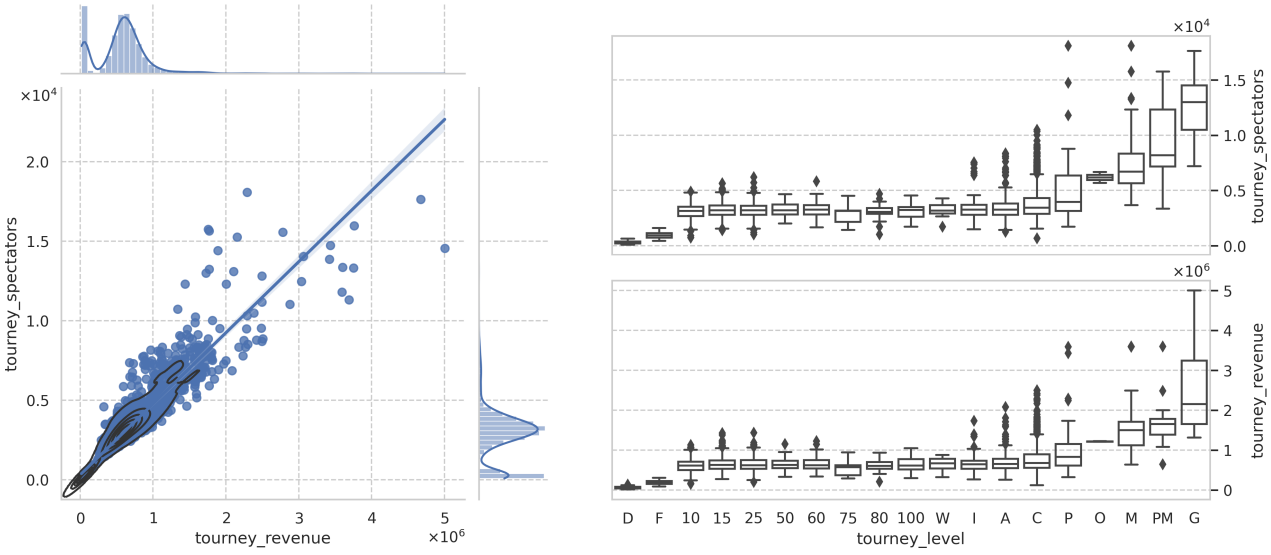


Figure 4: On the left, the scatter plot and linear regression highlighting the relation between the number of spectator of a tourney and its revenue, together with the uni-variate distribution of the two single variables; on the right, the boxplots depicting the distribution of these two features with respect to the tourney level

- The plots of Figure 5 are relevant to the four main features that the dataset provides about each player identity: height, age, handedness and nationality. We immediately spotted the

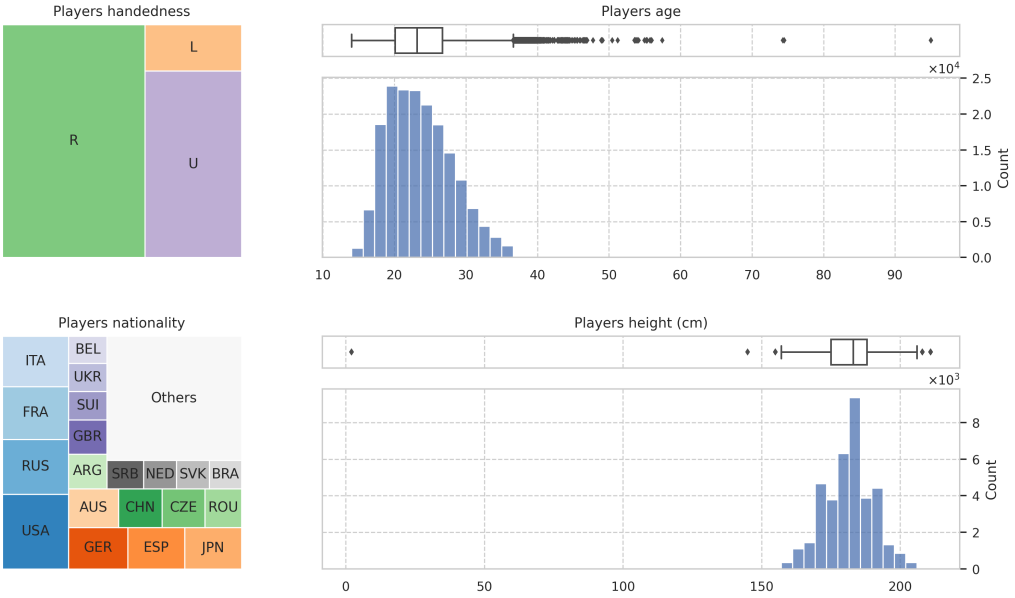


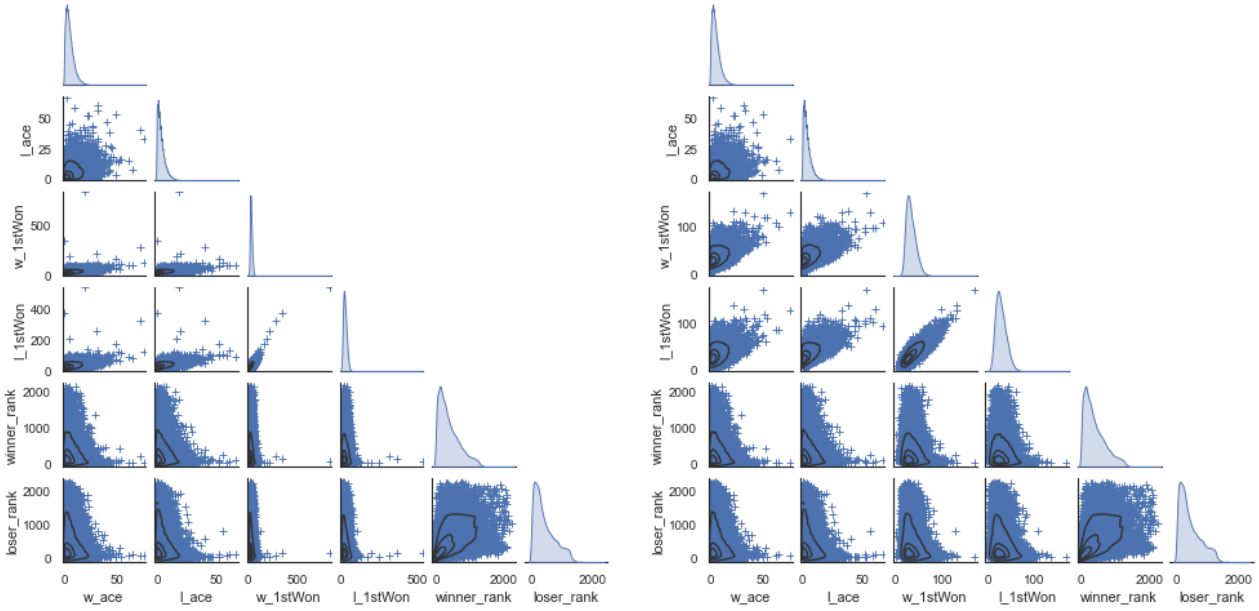
Figure 5: From left to right, top to bottom: distribution of the players' handedness; distribution of players' age, with focus on data inside of the whisker; distribution of the players' nationality, highlighting the top 20s; distribution of players' height, with focus on data inside of the whisker. The number of bins for the histogram plots are computed using the *Sturges' rule*

occurrence of some outliers due to erroneous data, as well as exceptional cases especially for the ages; however, we noted that these information present several issues:

  - there exist player names with multiple ids associated and vice-versa;
  - 32% of players have an unknown handedness, denoted by the `U` value;
  - each player id is intuitively associated to multiple age values, due to the fact that each of these records refers to a different moment in time; however, some of them have a too high standard deviation, hinting to some erroneous values or conflicting ids

- Putting side by side in a multiple scatter matrix some match statistics (Figure 6), apart from the outliers detection we can observe an intuitive behavior: as `winner_rank` and `loser_rank` decrease[1] the performance statistics increase in an inversely proportional way, indicating how better players tend to perform on average in comparison to weakest players, or even how the game needs to change in order to win against high ranked players

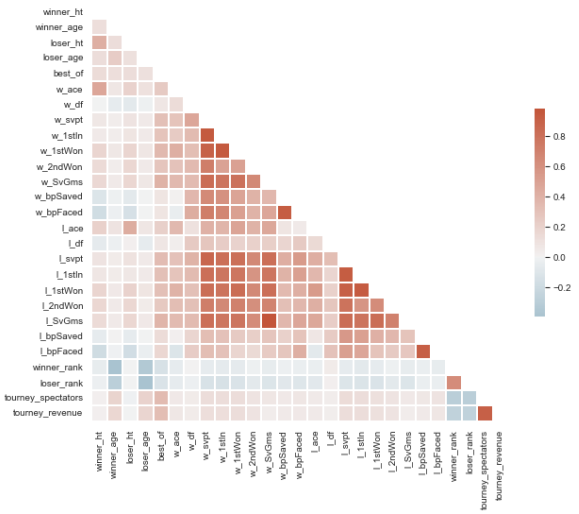
[1] Recalling that *lower is better*



(a) Plain dataset **(b)** After outliers removal

Figure 6: Multiple scatter matrix comparing some match performances and rank of the players. On the left on the plain dataset and on the right after the removal of the outliers

As far as it concerns the **data correlations** (plotted in Figure 7), it is worth noting how the center square of the heatmap, which represents the pairs of *winner* and *loser* statistics of the match, shows a good correlation. Again, this can be attributed to the reasoning done above, since most of the time the rank of the opponents is correlated, their performance must be correlated too (and it is from the heatmap!); which is the behavior we would expect from this kind of dataset.



Features		Correlation
`l_SvGms`	`w_SvGms`	0.981
`w_1stIn`	`w_1stWon`	0.943
`w_1stIn`	`w_svpt`	0.939
`l_1stWon`	`l_1stIn`	0.936
`w_bpSaved`	`w_bpFaced`	0.932
`l_svpt`	`l_1stIn`	0.932
`winner_rank`	`tourney_revenue`	−0.270
`tourney_revenue`	`loser_rank`	−0.270
`winner_rank`	`tourney_spectators`	−0.300
`winner_rank`	`loser_age`	−0.342
`winner_age`	`winner_rank`	−0.390
`loser_rank`	`loser_age`	−0.394

Figure 7: Correlations over the dataset. On the left the heatmap over the numerical variables of the dataframe, on the right the first and last six pairs of features sorted by correlation

1.1.3 Dependencies verification

Studying the dataset we observed that we don't have a field which can identify a precise tournament, in fact the field `tourney_id` represents both the *male* and *female* matches. Thus, for example, a dependency of the form

$$\texttt{tourney\_id} \implies \texttt{tourney\_level}$$

cannot be checked[2] without a feature which describes if the match is played between males or females. The dependencies found are the following[3]:

- `tourney_id` $\implies$ `tourney_name`, `surface`, `tourney_date`, `tourney_spectators`, `tourney_revenue`
- `tourney_id` $\wedge$ `tourney_sex` $\implies$ `draw_size`, `tourney_level`
- `winner_name` (`loser_name`) $\implies$ `winner_hand` (`loser_hand`), `winner_ioc` (`loser_ioc`), `winner_ht` (`loser_ht`)

Thus, to perform a sound dependencies verification we need to add a field `tourney_sex` into `tennis_matches`, which, in turn, implies to know the gender of the players. For this reason, **we had to postpone the dependencies check after the generation of the player list**, which will be described in Section 1.2.1.

1.2 Data Preparation

1.2.1 Players profile

The first phase of the data preparation concerned the generation of the players profile. In order to do this we had to reconstruct the sex of the players. However, taking the concatenation of the `male_players` and `female_players` dataset and then dropping the players not appearing in `tennis_matches` was not feasible since `male_players` $\cap$ `female_players` was non-null and too much large to be handled manually, since *the most* players in the two dataframes are not present in `tennis_matches`. Thus, the following approach was adopted:

1. Generate the list of the distinct names of tennis players appearing in `tennis_matches` with their extra information `hand, ioc, ht` (with possibly a lot of duplicates caused by inconsistent extra information)
2. Drop the `id` of the players[4] and keep the name as unique key, after checking that there are no homonyms exploiting the *additional information* of the players: `hand, ioc, ht`
3. Join the table with the concatenation of `male_players` and `female_players`, with the sex classification added as a feature
4. After a web search, we manually dropped the players with wrong sex appearing in both `male_players` and `female_players` by highlighting the duplicate rows having both genders[5]
5. Manage the same players with different extra information in this way: if corresponding fields are equals do nothing, if one is null and the other isn't keep the consistent one, if both are different manage them manually via a web search. **Note:** many outliers and *null* values have been discovered and fixed in this step.
6. Manage all the missing information via web-search.

In this way we obtained a consistent, no-duplicate player list with extra attributes associated.

Once obtained the list with all the players and their relative sex, adding the so called `tourney_sex` to each match was pretty straightforward and all the dependencies were verified too.

[2] In the same way we can replace `tourney_level` with `surface`, `draw_size`, and so on..
[3] We omitted all the trivial dependencies formed by the matches stats, like `w_svpt` $\leq$ `w_1stIn`
[4] The id values in `winner_id` and `loser_id` are completely messed up
[5] As an example: `Austin Smith, M; Austin Smith, F` $\bowtie$ `Austin Smith, U, USA` $\rightarrow$ `Austin Smith, U, USA, M; Austin Smith, U, USA, F`

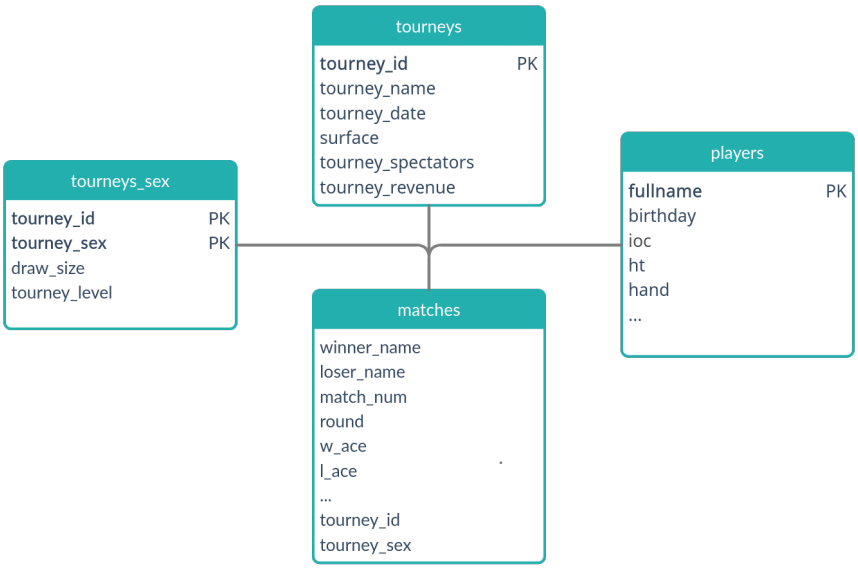


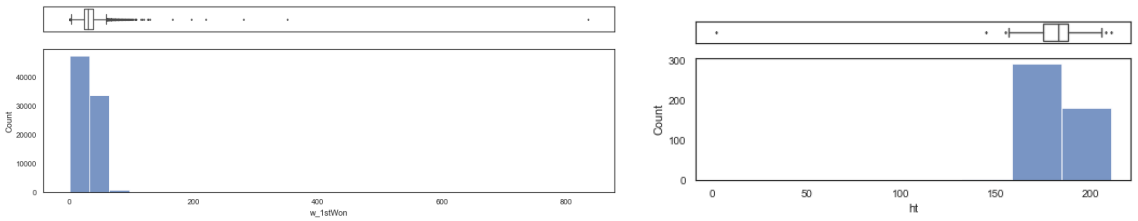
Figure 8: The final normalized schema of the dataframe

1.2.2 Removing the redundancies: towards a Normal Form

Since we verified all the dependencies found in Section 1.1.3 we can apply a "database" approach called **normalization**, where for each one of the dependencies we create a new table, the *primary keys* for those tables will be the *left* of the implication while the remaining attributes will be the *right* of the implication. Thus, these latter attributes can be removed from their original dataset (in our case `tennis_matches`), while the *keys* will be maintained to allow the possibility of reconstructing the original dataset through a **join** of the two tables. This approach allows to **remove all the redundant data** from the dataset.

The original dataset `tennis_matches` has been divided into four, smaller and simpler, dataset: `tourneys`, `tourneys_sex`, `players` and `matches`. The resulting, final schema of the dataset can be seen in Figure 8.

1.2.3 Outliers



(a) Number of first games won by the match winner

(b) Height of the players

Figure 9: Subset of the outliers in the dataset

While managing the categorical data was pretty straightforward, the numerical features with possible outliers discovered are the *height* (which we recall is not anymore divided into `winner_ht` and `loser_ht` but it is a single feature of the table `players`) and the age of a player, and the a player's *match statistics*. In Figure 6a and Figure 9 we can have a visual idea of the data distributions.

While for the *height* plotted above we can **surely** assert the presence of mistaken data, it's not so easy

to assert the same thing over the match statistics[6], so we adopted the following approach: through basic calculations we have placed a threshold over multiple statistics (as an example in a *3 set* match it is not reasonably possible that a player serves more than 144 times, since a maximum number of 39 games caused by the presence of the *tie-break* on the final set) and placed as *null*, since we don't want to lose other information like the match opponents, all the statistics which with *very high* probability were not reasonably possible[7]). Instead, all the statistics where a threshold could not be posed (like the statistics over *5 set* matches, which don't have the *tie-break* on the last set) were not touched, even if possibly mistaken.

What we have found is a list of *six* matches with **totally unreasonable** values. A portion of the dataframe after dropping these items can be seen in Figure 6b.

1.2.4 Filling `NaNs`

The following step consists in managing the *null* values of the dataset. Obviously we are not interested in filling the dataset of the matches but rather we are interested in filling the players' and tournaments' dataset missing information, since we are going to use them in the next tasks. Moreover, we are *highly* interested in **preserving the current distributions** when filling the null values.

The features which, at this step of the data preparation, have *null* values which need to be filled are:

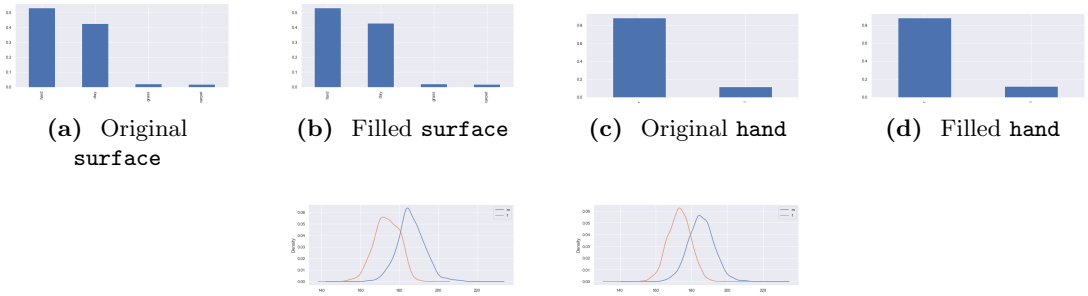
`tourneys:` `surface`

`players:` `ht`, `hand`, `ioc`

Both the `surface` and `hand` features have been filled by computing their original distribution (Figure 10a and 10c) and filling the missing values using the function `np.random.choice` over the original distribution (Figure 10b and 10d).

Filling the *height* of the players using the same method above would've been not optimal, first of all because it is not a *categorical* value but *numerical*, after that we would expect that the distributions for the height of males and females are different. For this reason we extracted both the distributions based on the gender (Figure 10e) and filled the missing values, according to their sex, using two different Normal distributions (Figure 10f).

Finally, the `ioc` has been filled manually via web-search since only *four* values were missing.



(a) Original `surface` **(b)** Filled `surface` **(c)** Original `hand` **(d)** Filled `hand`

(e) Original `ht` **(f)** Filled `ht`

Figure 10: Distributions before and after filling the missing values

[6] As an example, from the scatter of Figure 6a it is possible to see the historical 113 aces in a match from *John Isner* against *Nicolas Mahut* in *Wimbledon 2010*

[7] The full list of conditions can be seen in the Part 3 of the Task 1 notebook

1.2.5 New features

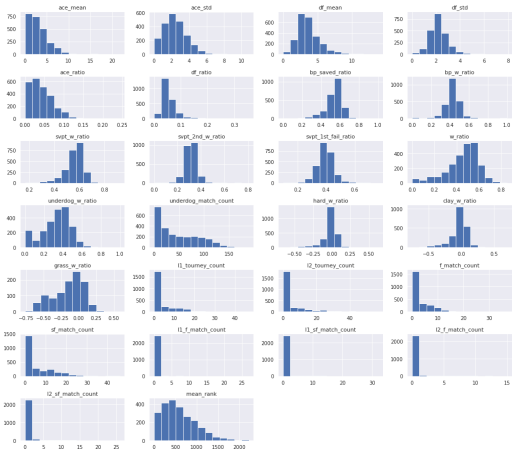
The last step of the data preparation consisted of **extracting the new features preparatory to the clustering task**. The features that we identified as *interesting* for the clustering are the following:

`stats_match_count`:	number of matches with available stats of each tennis player
`ace_mean`, `ace_std`, `df_mean`, `df_std`:	mean and standard deviation of number of aces and number of double faults
`bp_saved_ratio`:	ratio of saved breakpoint out of faced ones
`bp_w_ratio`:	ratio of won breakpoints out of fought ones
`svpt_w_ratio`:	ratio of won points out of the ones in which the player served
`svpt_1st_w_ratio`, `svpt_2nd_w_ratio`:	ratio of first-serve/second-serve won point out of the ones in which the player served
`svpt_1st_fail_ratio`:	ratio of points in which the player failed the first serve out of all the points in which the player served
`w_ratio`:	ratio of won matches out of played matches
`underdog_w_ratio` **and** `underdog_match_count`:	ratio of won matches as underdog (when the player has a lower rank than its opponent)
`grass_w_ratio`, `clay_w_ratio`, `hard_w_ratio`, `carpet_w_ratio`:	ratio of won matches on each type of tennis court
`l1_match`, `l2_match`:	number of matches played in Level 1 and Level 2 tournaments
`f_match_count`, `sf_match_count`:	number of finals and semifinals match played
`l1_f_match_count`, `l1_sf_match_count`, `l2_f_match_count`, `l2_sf_match_count`:	number of finals and semifinals match played in tournaments of Level 1 and Level 2[8]
`rank_semX`:	mean rank of the player in the semester $X \in [0, 12]$
`mean_rank`:	mean rank of the player along all the semesters
`rank_points_semX`:	mean rank points of the player in the semester $X \in [0, 12]$

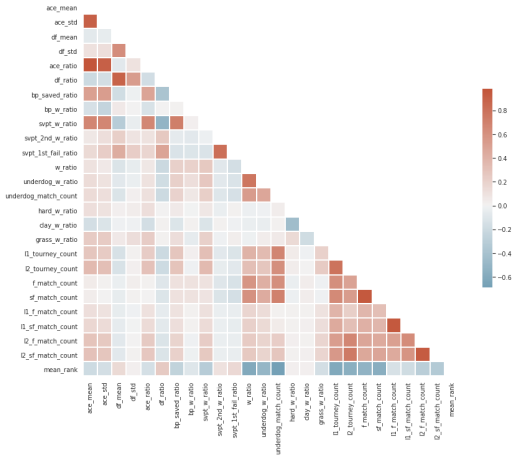
Preliminary to the extraction of the features we have omitted from the statistics **all the matches ended up by retirement or won by default**. This is a classic behavior when dealing with sports statistics in order to do not "fake out" the real values.

Once all the features have been extracted we proceeded on a simple analysis over them. The result of the analysis can be seen in Figure 11.

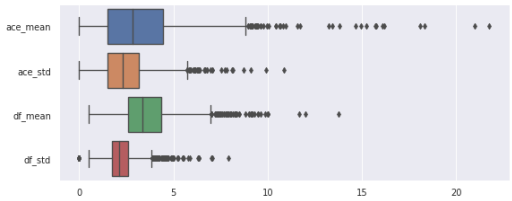
[8] We define as **Level 1** the *Grand Slams* and *Masters 1000*s tournaments and as **Level 2** the *ATP Tour* 500 and 250, or equivalents for the *WTA* circuit.



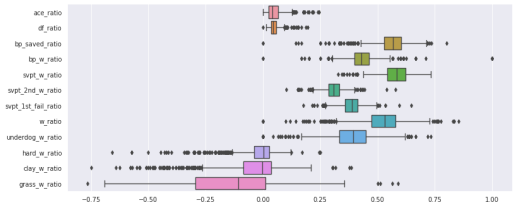
(a) Distribution of the features



(b) Heatmap of the correlations



(c) Boxplot over the ace and double faults features



(d) Boxplot over the ratio features

Figure 11: The analysis over the new features extracted

From the results we can do some observations about the newly-extracted features:

- The dataset contains mostly "second class" players, as we can see from all the features, but particularly from the Level 1 and Level 2 *match count*
- The *ratio* features behave in a *normal* distribution way, which is what we would expect
- A lot of players has no information regarding certain features, probably because of an insufficient number of matches
- There aren't features with an high value of correlation (apart from the trivial ones, e.g. the number of finals and semifinals reached in Level 1 tournaments)

2 Task 2: Clustering

Once we obtained a profile for each player, we moved on to Task 2, namely the clustering analysis phase. Having observed that most of the features we were able to extract aimed at describing *how* players play, we decided to challenge ourselves in being able to derive clusters of players based on their *playing style*.

A critical issue of this goal that we needed to face first, is the fact that the statistics of each player should have reflected as much as possible their actual playing style, ignoring any particularly lucky or unlucky matches; to make sure of this, we decided to treat only those players with a good amount of games within the dataset, meaning that their extracted statistics composing their profiles should be more reliable and accurate than those of others. The distribution of the `stats_match_count` feature showed that 50% of the players have less than 20 matches from which their statistics were previously extracted, while the other half consists of players having a larger amount, up to 367; since 20 matches

seemed like a sufficient amount to allow us to observe a certain playing style, we placed it as a lower bound to select the profiles that would then be subjected to the clustering analysis. In addition, this cutoff allowed us to remove players who had the lowest rankings, and that intuitively could not have developed a solid playing style yet.

The following sections will present the reasoning and the results we have obtained in order to pursue our goal, covering each of the three requested clustering techniques.

2.1 Hierarchical

The first clustering tecnhique that we decided to test was the *agglomerative hierarchical clustering*; the reason behind this choice is twofold:

1. it would have allowed us to assess how obvious the correct number of cluster was, and thus how many possible playing styles we could have afford to sharply identify;
2. it would have let us study which features allowed the cleanest division between different players stats and thus between different playing styles.

Within the player profiles we have identified the following nine features whose combinations of values could have hinted us to different playing styles:

a) `ace_ratio` b) `bp_saved_ratio` c) `bp_w_ratio` d) `hard_w_ratio`

e) `clay_w_ratio` f) `svpt_1st_fail_ratio` g) `svpt_2nd_w_ratio` h) `svpt_w_ratio`

i) `df_ratio`

as to understand whether and how much impact the player's actual skill has in defining its playing style, we have also decided to include these two additional features[9]:

j) `w_ratio` k) `mean_ratio`

Each of the so considered features was then standardized and the resulting data subjected to four different agglomerative clusterings, each using a different inter-cluster similarity method between *Ward*, *Complete*, *Average* and *Single*; the resulting dendrograms are shown in Figure 12.

[9]Keep in mind that, according to the correlations previously obtained and shown in Figure 11b, these last two features have little to none correlation with the other ones

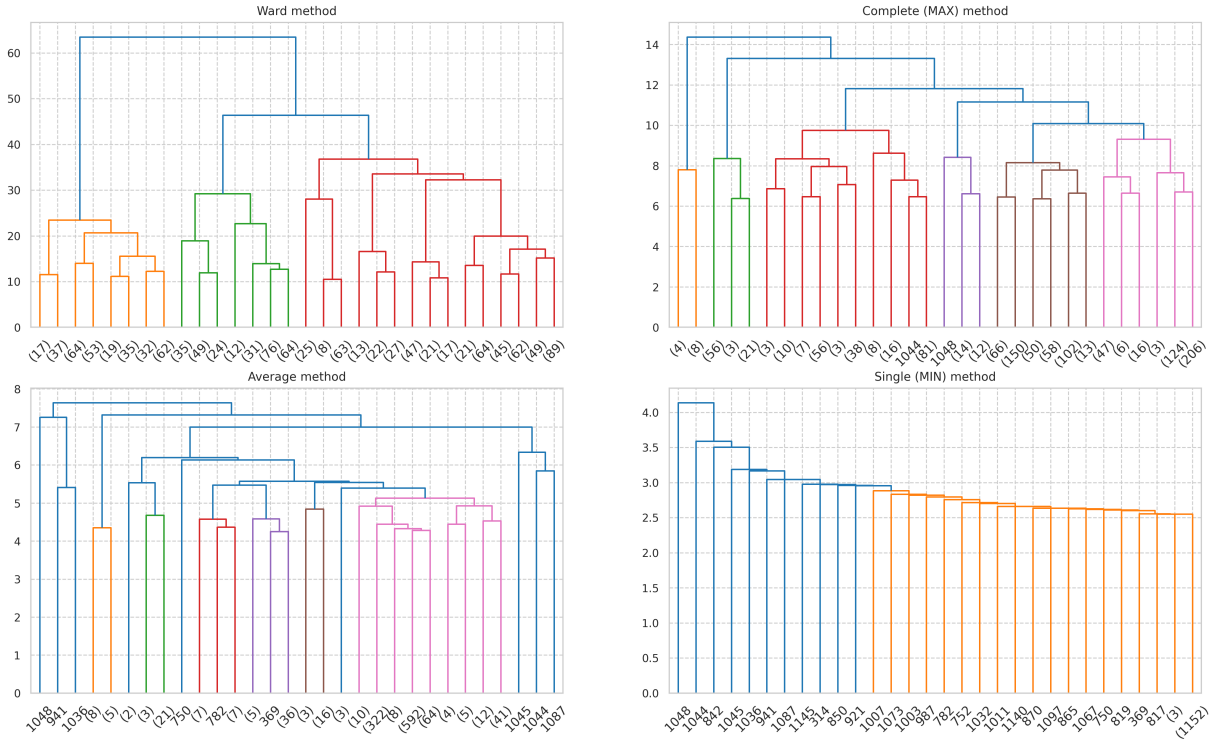


Figure 12: The four dendrograms obtained through hierarchical clustering, one for each inter-cluster similarity method considered

Since the agglomerative clustering algorithm proceeds by iteratively joining the most similar pair of existing clusters, we wanted to observe which subset of the eleven features had the greatest impact in making the latest formed clusters particularly distinguishable from one another, by observing the correlation between the cluster label assigned to each player and the features provided to the algorithm. The most relevant observations we derived are as follows:

Ward: The last formation using the Ward method is also the one that joins the least similar pair of clusters, hinting to 2 as being the best number of final clusters. Since 95% of male players are assigned to the first one, these two clusters show a strong separation between genders. The features that appeared to be more correlated to the cluster label are, in order, h), c), b) and a), although moving down the dendrogram to 3 total clusters, we note that the features j) and k) also have a strong impact on the division, meaning the players are split according to their skills

Complete: The small cluster of 12 players that the Complete method joins as the last step, is formed by only male players with a high feature a). The other large cluster, on the other hand, is constructed by joining players mainly differentiated by the features f), g) and again a). Lastly, going back to the 4 clusters, we find a clear division between low and high ranked players

Average: The dendrogram obtained using the Average method hints to 5 as the best number of clusters; however, 4 of them have a cardinality between 1 and 13, for a total of just 19 players. These clusters do not set themselves apart for a single specific feature: instead, they appear to be generally composed of outliers, each of them w.r.t. to a different large subset of the features

Single: Going towards the last iterations of the algorithm using the Single method, a progressively larger cluster is formed by merging with singleton clusters made of single players, whose feature values appear to be in contrast with those most commonly shared within the bigger and denser cluster

Although hierarchical clustering was unable to accomplish our goal of forming clusters based on playing styles, it did point out to issues and hints that benefited our subsequent clustering attempts, primarily:

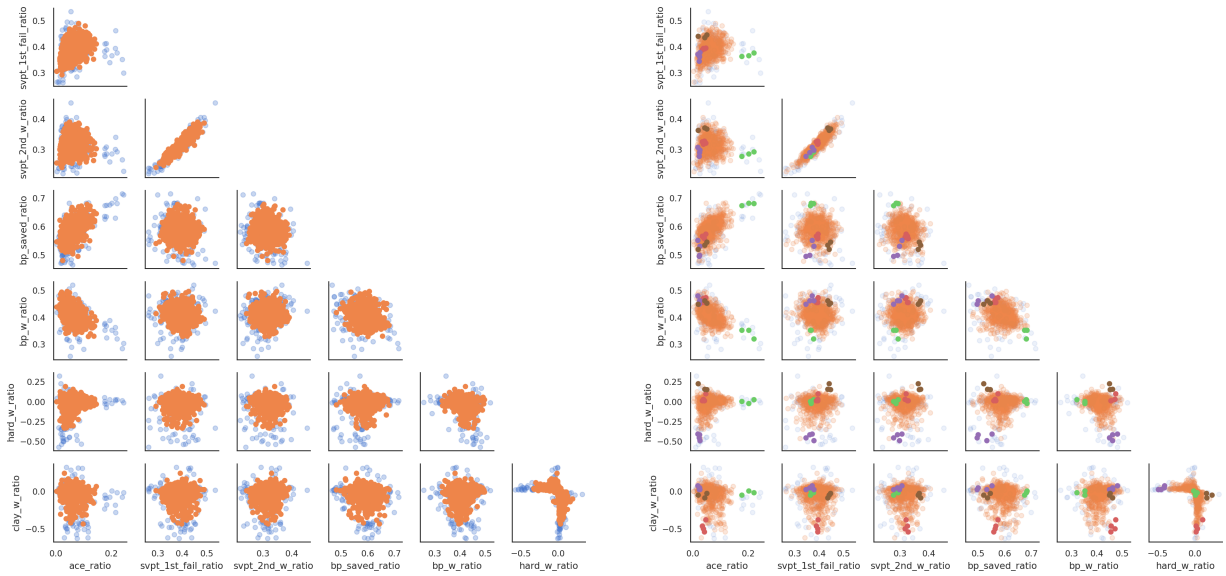
- player gender has a huge impact on expected feature values, meaning that for distance-based clustering we should first manually differentiate between males and females in order to uncover consistent playing styles shared by both sexes;
- ranking, and thus the actual abilities of players, have just as much of an impact, albeit a lighter one;
- because of the previous observation and because of the correlation discovered between cluster labels and features, the most relevant features are seven: from a) to g);
- players with a more aggressive, attacking style, characterized by high value on characteristics a) and b), were likely going to be part of the playing styles we would later identify.

2.2 DBSCAN

Our first attempt of clustering with a stricter set of features and a greater knowledge of the results we could expect to obtain, was the *density-based DBSCAN clustering*.

As a first trial, according to the suggestions found in [San+98], we set the DBSCAN parameter `minPts` to be twice the dimensionality of the data, thus 14; we then executed the algorithm varying its ϵ parameter. Generally, as shown in Figure 13a, every trial resulted in just a single cluster that gathered players having the most common feature values, positioned in the center of each gaussian-like distribution – intuitively, larger ϵ values resulted in the growth of this cluster. All the players that did not fall into this cluster were considered as noise points. These first results are consistent with the primary purpose of DBSCAN, which is to identify clusters based on density, but did not appear to be useful yet.

In order to find more than just a single cluster, we decided to try out each value for `minPts` in the range between 3 and 14; for each of them, a correct value for the ϵ parameter has been found by running the *K-nearest neighbors* algorithm with `k=minPts`, sorting the euclidean distances between every player and its k-th neighbor w.r.t. the considered subset of features and finally selecting the distance located at the knee point of the obtained curve as the best ϵ value. By doing so, we were able to find up to 5 different clusters, depicted in Figure 20b.



(a) First test clustering

(b) Final clustering. The green cluster represent players with an aggressive playing style

Figure 13: Some example of clustering obtained with the DBSCAN algorithm. The blue points were considered as noise

What we found was that some of the "satellite" clusters found around the larger central cluster actually represented some sort of playing style, such as aggressive with a high aces ratio; unfortunately, these were only found because of their fortunate circumstances, i.e. their distance from the main cluster and their sufficient density, and did not suit our goals.

2.3 K-means and X-means

Our last clustering attempt involved the use of the distance based clustering algorithms *K-means* and *X-means*[10].

The two algorithms have been executed on both male and female players separately, by also varying a minimum threshold set on the ranking of the players. In order to find the best k value, we made use of three different indicators:

- the number of clusters found by the X-means algorithm;
- the number of clusters suggested by the *elbow method* applied on the *SSE curves*;
- the number of clusters inferred by comparing various *silhouette plots*.

By the end of the analysis, we found out that both female and male players having a rank lower than 600 could be grouped into five different clusters as seen in the graphs in Figure 15; each cluster grouped players having similar playing styles, and each *centroid* shown in Figure 14 is representative of that style:

[10]The library used for computing the *X-means* is pyclustering

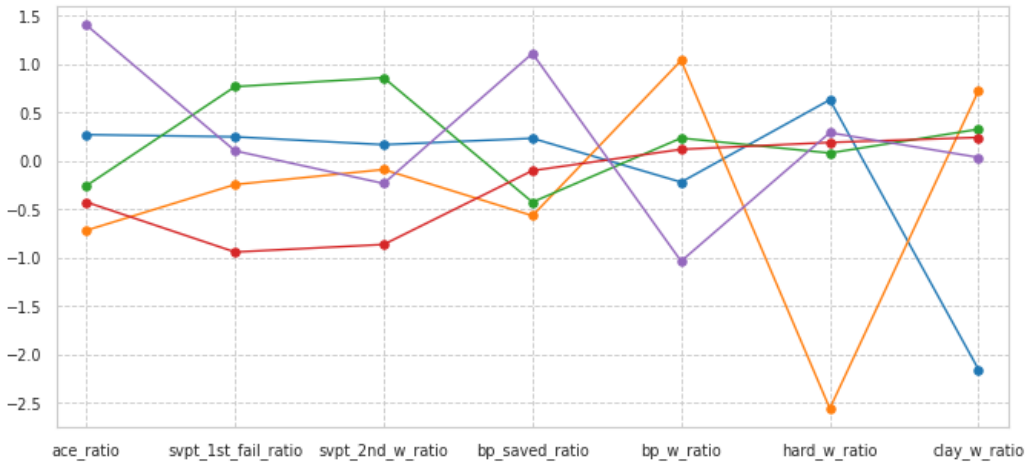


Figure 14: The five centroids representing each identified playing styles with K-means clustering.

Powerful serve players: those kind of players that set their self apart for their powerful and winning serves; they are able to score points mainly in their first serve thanks to numerous aces, causing a weaker second serve with an high ratio of failure; having the serve by their side, they are better at defending an opponent breakpoint rather than winning one by their own. They benefit from fast surfaces like hard ones while suffering from slower surfaces like clay.
Players in the cluster: John Isner, Kevin Anderson

Offensive baseliners: players having an offensive style with an high degree of aces; they favor fast surfaces while having hard times on slower ones: for this reason, it is hard to find high ranked players inside of this cluster.
Players in the cluster: Daniil Medvedev, Adrian Mannarino

Defensive baseliners: exactly the opposite of the Offensive baseliners, i.e. players with a defensive style that favour slow surfaces.
Players in the cluster: Marco Cecchinato

Total court players: they represent the majority of the top ranked players, since they are able to perform equally well on both kind of surfaces and to generally keep a low degree of error on serve.
Novak Djokovic, Rafael Nadal

Unbalanced players: for these players, it is hard to be able to identify a distinctive style of playing; this cluster is mainly formed by players which, in the course of the years, have been discontinuous in their results.
Players in the cluster: Andy Murray, Fabio Fognini

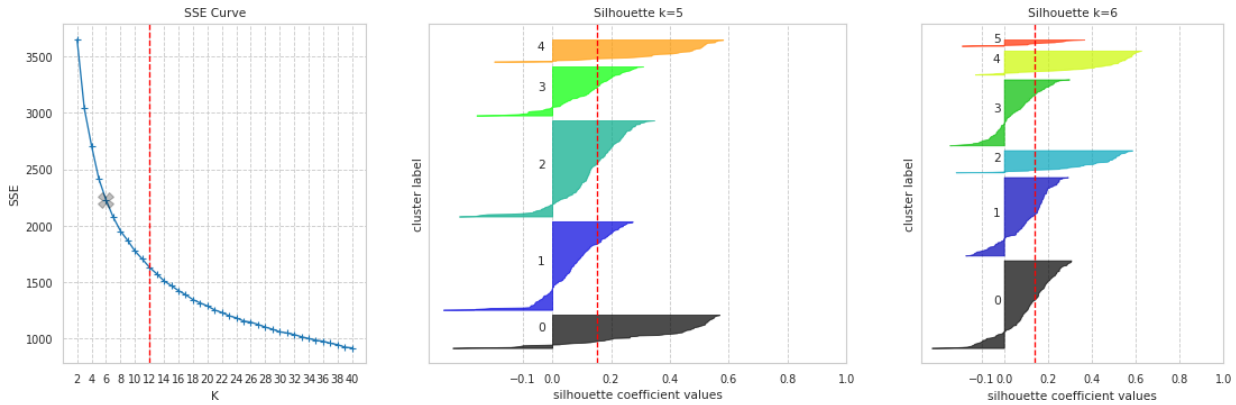


Figure 15: The SSE curve together with two silhouette plots of the K-means clustering applied on male players having a rank < 600: the one for the suggested $k = 6$ from X-means and the for the $k = 5$ we have chosen

3 Task 3: Predictive analysis

The classification task consists of predict if a player is labeled as *top-ranked* or *low-ranked*. In order to do this we exploited the features extracted during the Data Preparation phase (Section 1.2.5). So **no further features or player profiles were needed**.

In particular, the label was generated as follows:

A player is said to be top-rank (or label equals to 1) *if:*

- Has reached at least one time a **Level 1** or **Level 2** semi-final
- Has entered, at least in one semester, the *top 50* players in the *ATP* or *WTA* rank

Like we did in the Clustering analysis phase, we decided to remove from the dataset the players which did not reach *at least* 20 matches with available statistics, so as not to distort the classification task.

The remaining dataset is divided in this way:

- 1607 *low-rank* players
- 347 *high-rank* players

Which are obviously a really *small* number for a classification task, but we try to do our best!

Furthermore, the dataset is clearly imbalanced, which is a detail we'll need to consider later.

3.1 Model selection and evaluation

We have trained and tested **four different classification algorithms**: Decision Tree, Random Forest, Support Vector Machine and Neural Network.

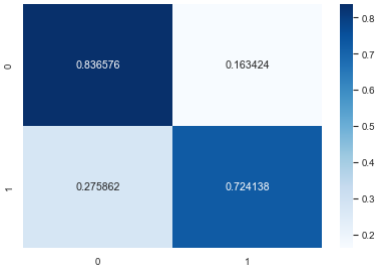
To perform tuning of the hyper-parameters we carry out for each model a randomized grid search, that allows to test the parameters with a larger domain of values; in order to do that, it integrates a *k-fold cross validation method*.

As a metric for model evaluation we use the AUC_ROC: this metric is equal to the probability that a classifier will rank a random positive sample higher than a random negative sample, and has been chosen due to the class imbalance in the dataset.

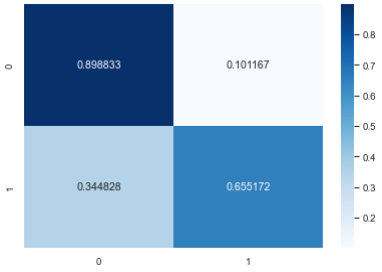
To test the final models, we implement an hold out approach, dividing the dataset in two parts, a training_set and a test_set, respectively in the amount of 75% and 25%.

3.2 Decision Tree and Random Forest

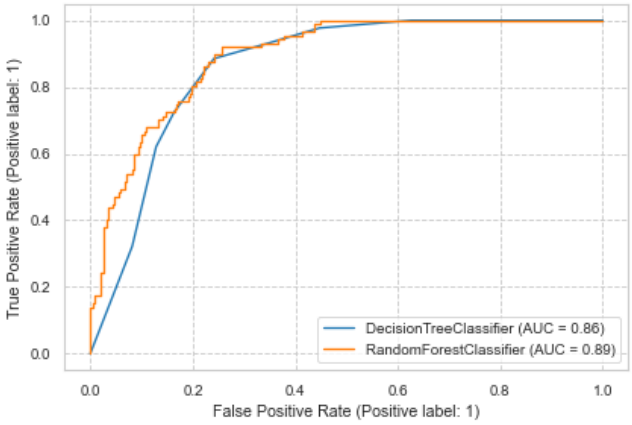
The first methods tested are the **Decision Tree** and the **Random Forest**. The parameters have been chosen using classical *reference* values and the results show a preference over the *entropy* criterion and tree depths of about 25 on both the Decision Tree and the trees inside the Random Forest. The results can be seen in Figure 16 and describe a good performance for both models with an high *AUC* value (0.86 for the Decision Tree and 0.89 for the Random Forest). However, in order to achieve a better accuracy the Random Forest model has decreased the rate of true positives and increased the rate of true negatives; trying other parameters didn't improve the performances neither.



(a) Confusion matrix of the Decision Tree



(b) Confusion matrix of the Random Forest



(c) *ROC* curve of the Decision Tree and Random Forest

Figure 16: Results of the best Decision Tree and Random Forest

3.3 Support Vector Machine

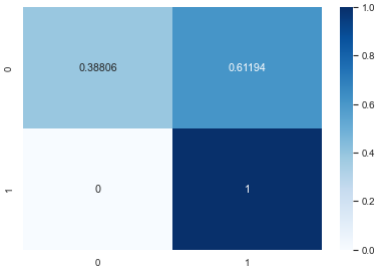
While performing the Randomized Search we've found that *this specific* classification model reached decent performances only through the *weight* parameter set to 'balanced' to correct the imbalancing of the class, where they are computed as:

$$\frac{\text{n\_samples}}{\text{n\_classes} * \texttt{np.bincount(y)}}$$

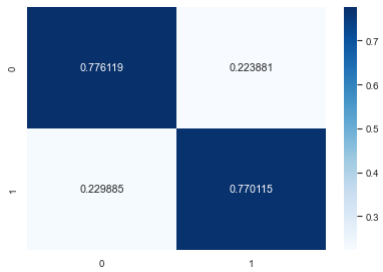
so it consists in automatically adjust weights inversely proportional to class frequencies.

The *SVM*s were tried over **three different kernels**: *Linear Kernel, Polynomial kernel* and *Sigmoid kernel*. In Figure 17 we can see the results. While all the three kernels achieved the same *ROC* score,

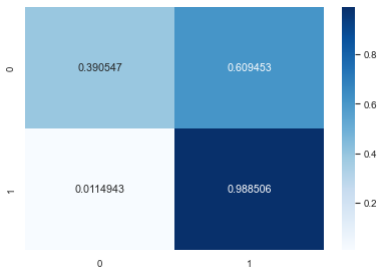
the *linear* and *sigmoid* kernel performed in a very similar way, with a *nearly* perfect classification of true positives but a low performance over the opposite label. Instead, with the *polynomial* kernel we managed to obtain a balanced classification over both labels, which made it the kernel of reference for this classification model.



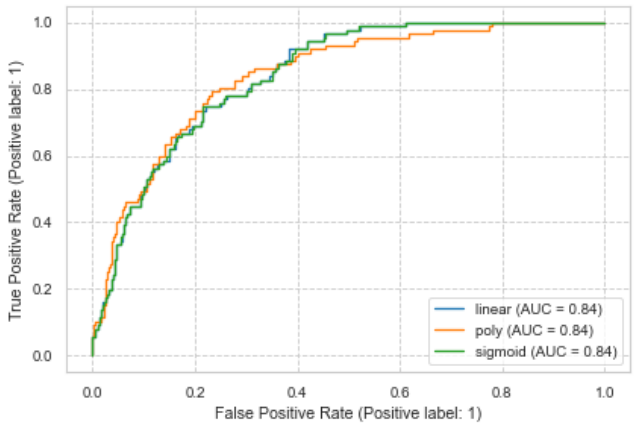
(a) Confusion matrix of the linear kernel



(b) Confusion matrix of the polynomial kernel



(c) Confusion matrix of the sigmoid kernel



(d) *ROC* curve of the different SVM kernels

Figure 17: Results of the best SVM kernels

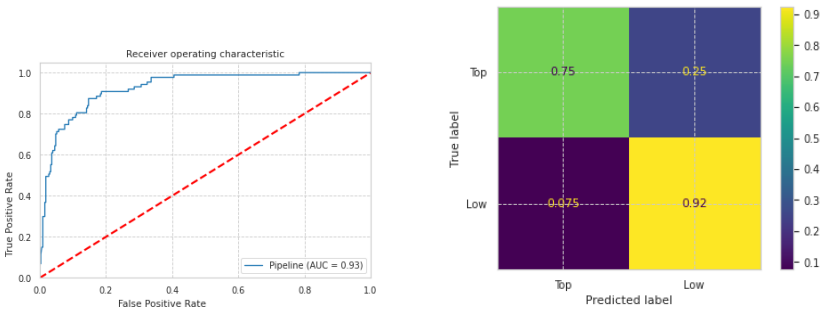
3.4 Neural Networks

Finally, we then tried to perform a model selection phase for a neural network classifier model. In order to face the unbalanceness of the dataset, this time we wanted to try out the oversampling `SMOTE` techinque, specifically its variant `SMOTENC` which also allows to treat categorical inputs, in our case the sex of the players.

The randomized grid search tried to vary the following hyperparameters:

- the amount of oversampling peformed by the SMOTENC technique on the underrepresented class of Top players
- the number of units in the first and second hidden layer of the neural network, ranging between $[32, 256]$
- the activation function of the hidden units, choosing between `relu`, `tanh` and `sigmoid`
- the L2 regularization term to avoid the phenomena of *overfitting*
- the learning rate for the *Adam* optimizer
- the size of the mini batch, ranging between $[16, 32, 64]$

The best neural network found was able to surpass every other classifier both in terms of the ROC's *AUC* and in terms of general accuracy, as it is shown in Figure 18



(a) *ROC curve* **(b)** *Confusion matrix*

Figure 18: Results of the best neural network classifier found

4 Conclusion

At first, we had a large dataset containing numerous tennis matches from high and low level tournaments. Through the phases of **Data Understanding** and **Data Preparation** we were able to reorder, analyze and extract useful information in terms on new features from the dataset.

Through the **clustering analysis** we've been able to distinguish the different players of the dataset according to their *play style* derived by the extracted features. Furthermore, the found styles represented the players in the dataset, at least for the ones we knew, in a very precise way.

Finally, in the **predictive analysis** we exploited different classification algorithms to build a classification model which identifies whether a player can be defined as *high-rank* or not. Thus, we compared those models achieving the best result through **neural networks**.

5 EXTRA: Time series analysis

5.1 Dataset

The dataset presents the values of monthly average temperature and its uncertainty of 100 cities of the world, collected in the period that goes from February of 2000 until January of 2010, so we have a total of 120 couple of values for each city.

For this particular task has been used only average temperature values

In the dataset, as we expected, we can note the phenomenon of seasonality, with trends that repeat themselves periodically, in this case every year; we will exploits this fact in our clustering analysis

To carry out clustering on cities' temperature trends has been used two different methods:

- The first method consists of generating a new time series starting from the original dataset, computing the mean of monthly values of Average_Temperature.

  We can see the final time series as an approximation or better a compression of 10 years in only one.

- The second method use the subsequential clustering approach, that consists in dividing the original time series in n parts and running on each of them the clustering algorithm, analysing and merging the final results

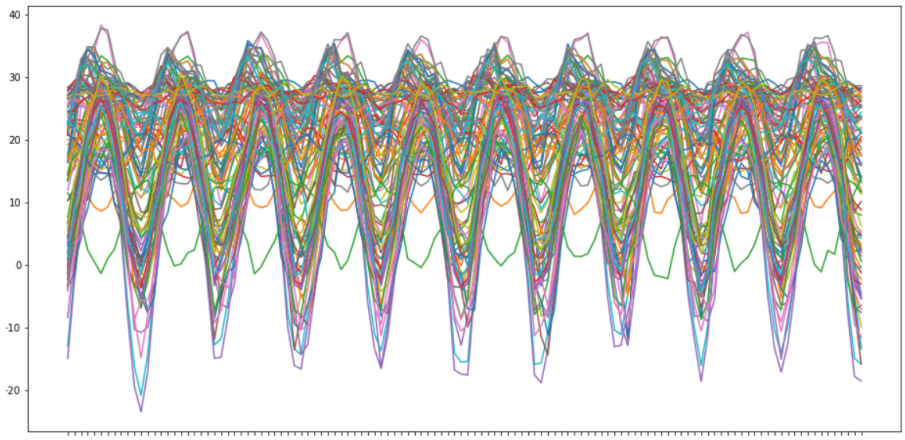
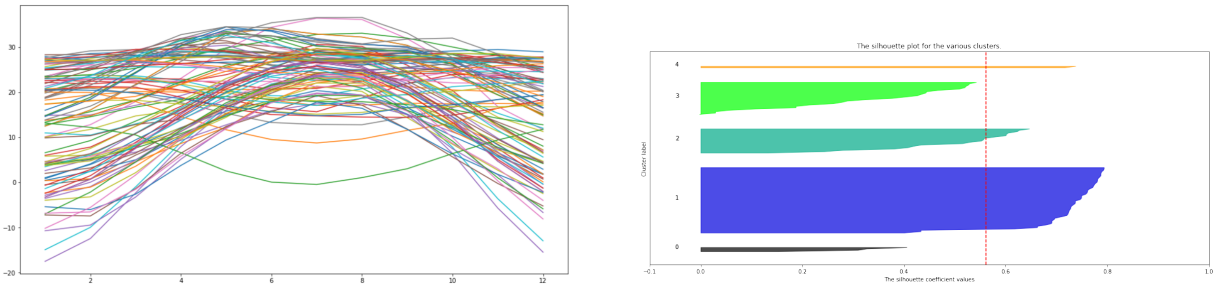


Figure 19: Original time series plot (x_axis=Time, y_axis=Celsius degrees)

Clustering algorithm used in both methods are k-means algorithm optimized for time series, provided by library *tslearn*, with which we choose to use the Euclidean metric. Before applying the algoritm, data has been scaled using method *TimeSeriesScalerMeanVariance*.

5.2 Method 1

As we anticipated, in this method we compute the mean of temperature trends of all 10 years obtaining time series as shown in figure.



(a) Average TS of 10 years

(b) silhouette score of k means clustering

Figure 20

We perform now a clustering using TimeSeries_Kmeans algorithm, and exploiting the elbow method we fix the hyperparameter k to 5, obtaining so 5 clusters each one containing respectively 4, 50, 19, 25 and 2 cities, with silhouette score as in figure 20(b).

5.3 Method 2

The second method take advantage of subsequential clustering and consist in splitting the dataset in 10 parts, one for each year and cluster on each of them. To perform each clustering, we use the same procedure as in method 1, so we use SSE to individuate the best k, that has been revealed to be 5 for each year. Finally, we analyze the 10 set of 5 clusters, and we compute final clusters grouping cities with more common clusters.

5.4 Analysis of results

Clusters resulting from both methods turn out to be equals, so for an overview, we decided to show ones of first method, that are more intuitive.

The 5 clusters are formed as follows (note that plot timeline starts from February):

- CLUSTER 1

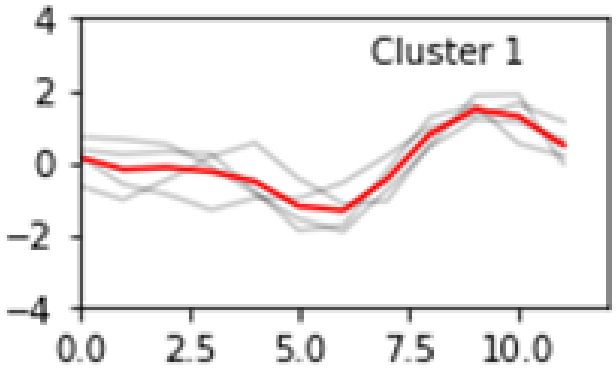


Figure 21: Cluster 1

We notice a trend where summers are cold, with temperatures below 0 and hot winter, we find in this cluster only 4 cities, Brasília Brazil, Surabaya Indonesia, Harare Zimbabwe, Fortaleza Brazil

- CLUSTER 2

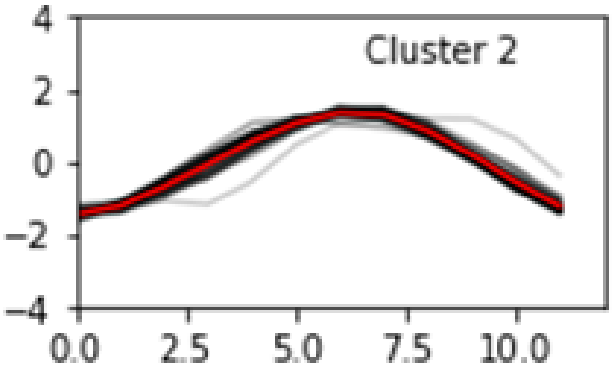


Figure 22: Cluster 2

Cities with cold winter and hot summer, infact we find cities like Rome, Paris, New York

- CLUSTER 3

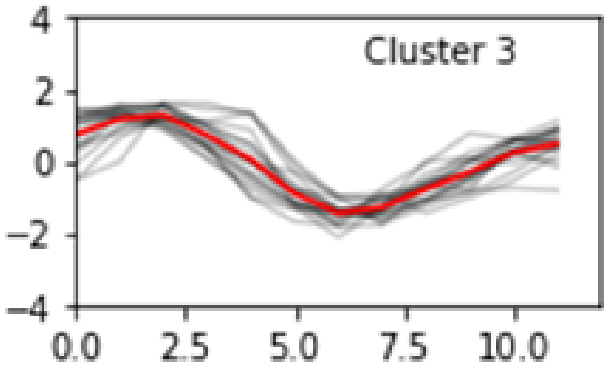


Figure 23: Cluster 3

Cities with very hot winter and colder summer, where we can see Lima Peru, Rio De Janeiro Brazil, Cape Town South Africa

- CLUSTER 4

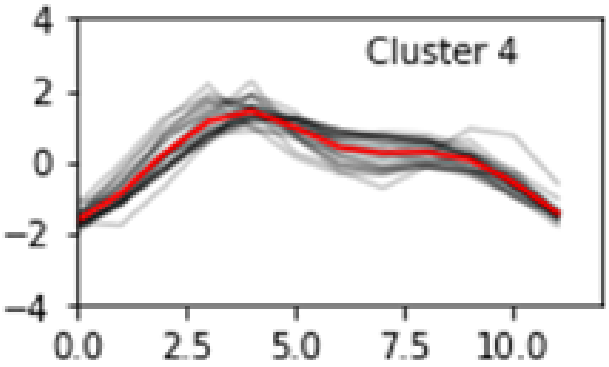


Figure 24: Cluster 4

Cluster four involves cities with high temperature between march and may, and decreasing in the rest of the year: we see that it encloses mainly south asia cities.

- CLUSTER 5

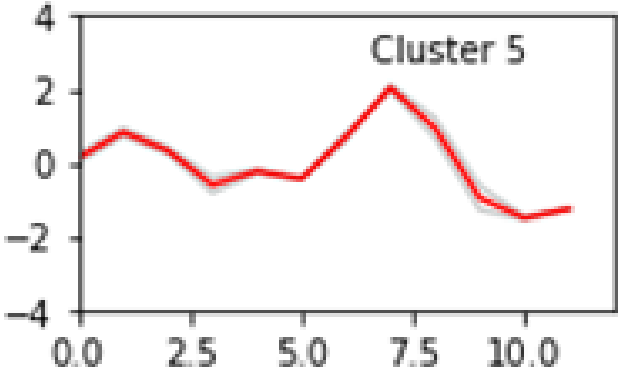


Figure 25: Cluster 5

Involves only two cities located in colombia that presents particular trends, with high temperature in September and October and cold winter

Clusters have been also plotted on a map, and as we can see it appears that clusters depends also from the geographical location of the cities.

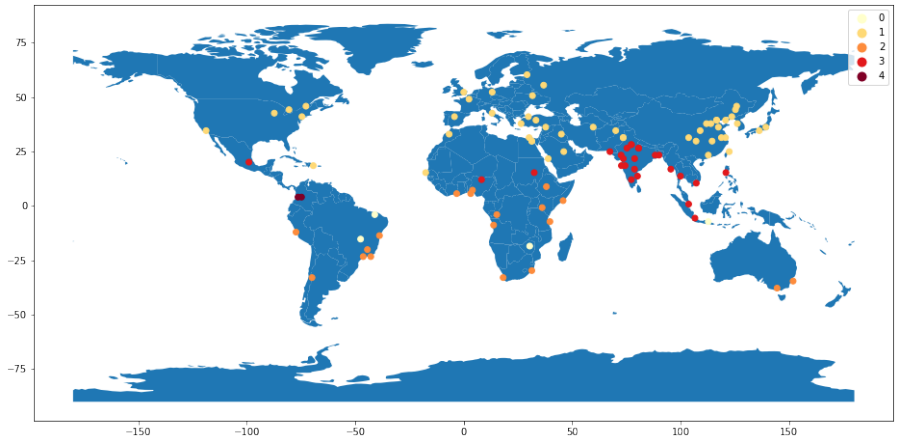


Figure 26: Plot of clusters on map