University of Pisa

Department of Informatics

Data Mining

Academic Year 2019/2020

Loan Status Project Report

Maris Basha [591203]
Tommaso Cavalieri [597707]
Haris Dukic [584829]
Pierpaolo Sepe [587639]

May 27, 2020

Contents

1 Data Understanding

The dataset used for the analysis presented in this report is likely to be provided by a bank since it contains information regarding the **status of loans** and also some historical data on the clients to whom they were given, such as their credit problems in the past.

1.1 Data semantics

The explored dataset contains **100,514 records**, even though 515 of them do not contain any value and 10,214 are duplicates, so the dataset we have actually analyzed contains 89785 records. These instances present **19 features** (not all them as we will observe in Section 1.3), of which further detail is given in Table 1.1.

Type	Specific	Name	Range
Numeric	Discrete	Number of Open Accounts	0 - 76
		Bankruptcies	0 - 7
		Tax Liens	0 - 15
		Number of Credit Problems	0 - 15
		Months since last delinquent	0 - 176
Numeric	Continuous	Current Loan Amount	10802.0 - 789250.0
		Credit Score	585.0 - 751.0
		Annual Income	76627.0 - 165557393.0
		Monthly Debt	0.0 - 435843.28
		Current Credit Balance	0.0 - 32878968.0
		Maximum Open Credit	0.0 - 1539737892.0
		Years of Credit History	3.6 - 70.5
Categorical	Nominal	Loan Status	{'fully paid', 'charged off'}
		Years in current job	1 - 10+ years
		Term	{'short term', 'long term'}
		Home Ownership	{'home mortgage', 'rent', 'own home', 'have mortgage'}
		Purpose	{...}
Categorical	Identification number	Loan ID	-
		Customer ID	-

Table 1.1 Features descprition

1.2 Distribution of the variables and statistics

The first step in the analysis was the exploration of the distributions of different variables, which were graphically analyzed using **histograms** (for continuous features) and **bar plots** (for discrete features). Particular attention was given to the ***Loan Status*** variable, as it is the one in which the bank, which has been considered as the client of this study, might be more interested. It can be seen pretty clearly in Figure 1.1 that its distribution is a little unbalanced, which is not a surprising result since we can expect that an efficient bank receives back the vast majority of the money it lends. In fact this result seems to be in line with the average loans' fully payment percentage on the market today. Even more **unbalanced** are the distributions of the variables *Bankruptcies, Number of Credit Problems* and *Tax Liens*, which all have a large majority of values 0.

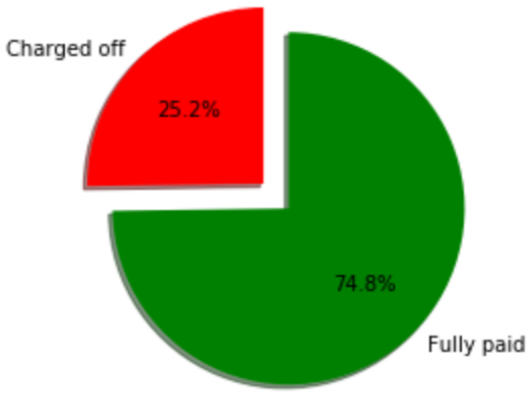


Figure 1.1 Loan status overall distribution

Some of the most interesting plots are presented in Figure 1.2, from which it is possible to appreciate that for example the *Credit Score* distribution has an exponential growth, *Current Loan Amount* seems to have a poissonian distribution, the majority of the customers have had their job for more than 10 years and 86% of them did not have any previous credit problem.

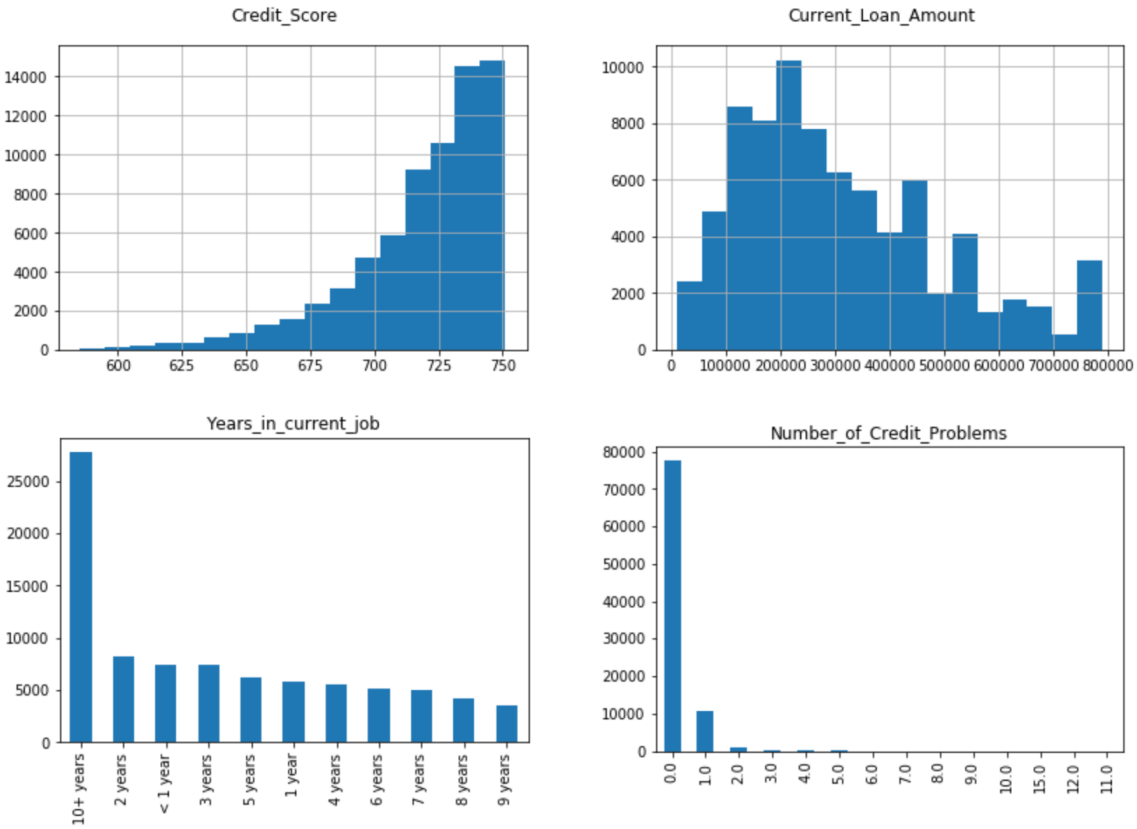


Figure 1.2 Variables distributions

The influence of other variables, such as *Term* (the duration of the loan) or *Home Ownership*, on the distribution of *Loan Status* was also tested. From the plots in Figure 1.3 it is possible to capture that long term loans are more likely to be subject to default, as well as loans given to people who do not have a house of their property.

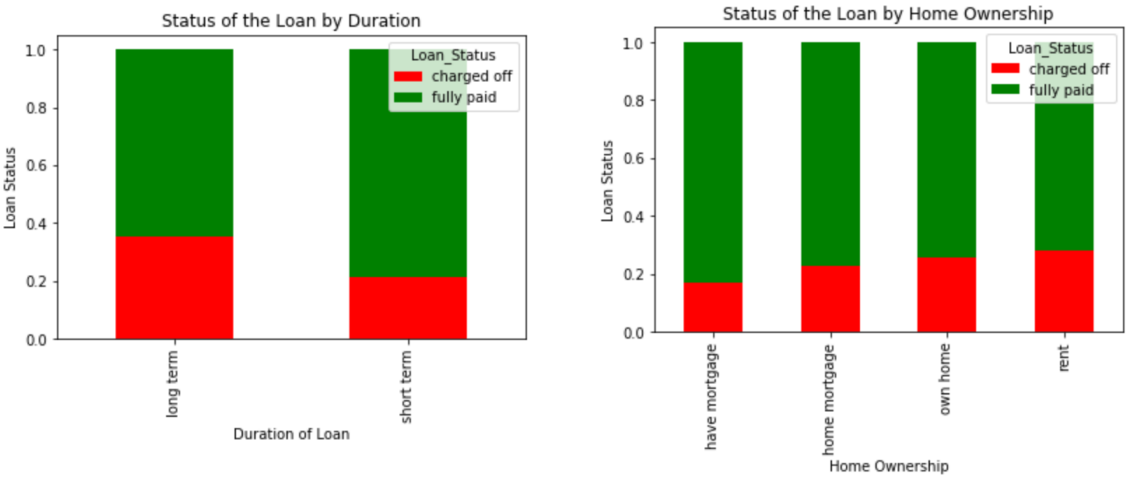


Figure 1.3 Loan status vs. other features

1.3 Data quality

In this section the quality of the data will be discussed, such as the presence of **outliers** and **missing values**. The first thing that was noticed is that *Current Loan Amount* contained a lot of records with value equal to 99999999, which clearly was not the actual value for the feature (way too high) but a way of recording missing values for this variable and it was therefore treated as such. The same applies for records with *Maximum Open Credit* equal to 0, since it has been supposed that a customer, in order to be in this dataset, should have been in debt at least once during his/her credit history. Moreover, 4551 records had a *Credit Score* between 5000 and 8000, which was easily detected as a compilation error since all the other records had a value between 600 and 800 for such attribute; for this reason, these values were divided by 10, in order to make them fit in the distribution. Please note that all these changes were already taken into consideration in the Table 1.1. The variable that presented the highest number of missing values was *Month Since Last Delinquent*, which had more than 50% of missing values and was therefore excluded from the rest of the analysis. Other features that presented a high number of missing values (more than 10,000) were *Credit Score*, *Annual Income* and *Current Loan Amount*, while *Maximum Open Credit*, *Bankruptcies* and *Tax Liens* had a low number of missing values (less than 700). This problem was dealt with by setting the missing values to match the distribution for the categorical variables and equal to the mean in the continuous variables, in order to not affect such characteristics of the features. Note that such procedure was applied after eliminating the outliers, which would have otherwise influenced too much such values. After some further investigation on the missing values for the variable *Bankruptcies*, it was found out that for those records the correspondent values of *Number of Credit Problems* were only 0 and 1 and since it is not possible for the former to be higher than the latter, instead of using the distribution on the whole dataset, values were given according to the frequencies of 0 and 1. The second main concern in data quality assessment was to look for outliers. This problem was dealt with by observing **boxplots** for the various attributes, which highlighted that the variables *Credit Score, Annual Income, Monthly Debt, Number of Open Account, Years of credit History, Current Credit Balance, Maximum Open Credit* and *Number of Credit Problems* had such problem. As already mentioned, after eliminating the outliers, the previously assigned values to the missing ones of the continuous features were substituted using the new, more reliable means. The dataset obtained after this procedure had **80723 records** and **18 features**.

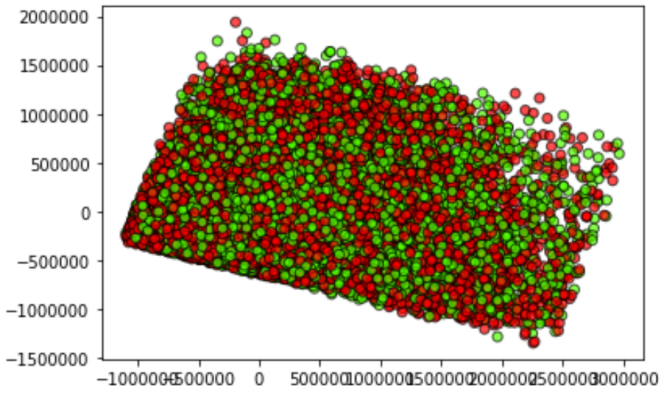


Figure 1.4 Principal Component Analysis

1.4 Variables transformations

After replacing missing values and excluding outliers, some further transformations were considered necessary to prepare the dataset for further anaylsis. The values of the variable *Purpose* were rearranged into fewer values, namely *home improvements, debt consolidation, buy house, business loan, buy a car, take a trip, medical bills, vacation, educational expenses* and *other*. Similarly the values of the variable *Years in the Current Job* were regrouped into: *<1 year, 1-5 years, 6-10 years* and *10+ years*.

1.5 Pairwise correlations and elimination of redundant variables

As it can be observed in Figure 1.5, *Bankruptcies* and *Number of Credit Problems* seemed to be **highly correlatad**, which is something that we might expect also from the semantics of these features, since banktuptcies are actually one specific type of credit problems a customer might encounter. *Bankruptcies* was hence deleted, as well as *Months since last Delinquent* (as mentioned before), *Loan ID* and *Customer ID*, which are useless since thay are identification numbers . Moreover, the variable *Tax Liens* was deleted too, since it was extremely unbalanced (98% of 0s). The cleaned dataset used for the analysis in Sections 2, 3 and 4 is therefore composed of **80723 rows** and **14 coulmns**.

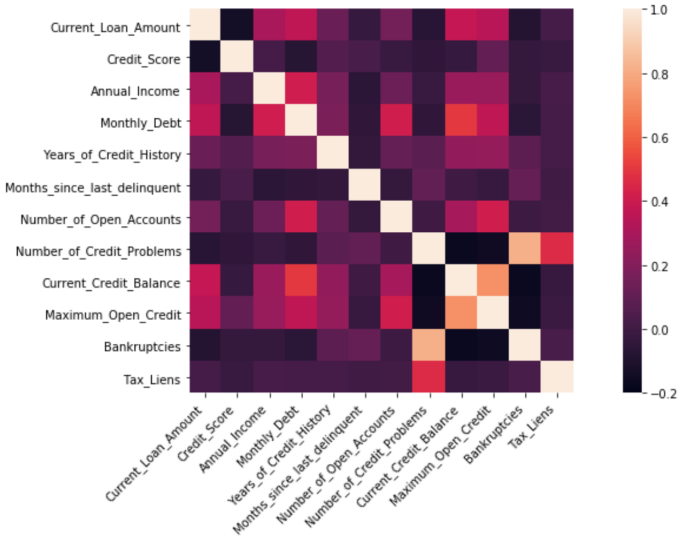


Figure 1.5 Correlation matrix

2 Clustering

The cleaned dataset described at the end of the Section 1 was used, as anticipated, for some further analysis. The first task, presented in this section, was to try to identify some clusters in the dataset by applying three different techniques: **center-based** clustering with K-means, **density-based** clustering with DBSCAN and **hierarchical** clustering with the agglomerative type.

2.1 Analysis by K-means

The first thing to do before starting to explore the possible clusters in the dataset is to select which attributes should be used for this goal. In fact, basic clustering algorithms such as the ones applied in this study have problems dealing with categorical attributes, therefore the variables *Loan Status, Term, Years in current job, Home Ownership* and *Purpose* were not considered for this task. Furthermore, specifically for the application of k-means, after comparing the results obtained with different configurations, *Number of credit problems* and *Credit score* were excluded too, because their highly unbalanced distributions influenced negatively the discovering of clusters. The metric chosen for the analysis was the **euclidean distance**. Since k-means needs as an input the desired number of clusters to be found, once selected the attributes and the distance function, the best value for **k** needed to be identified too. For this purpose, the so-called ***elbow method*** was applied: it consists in looking for a drastic change of slope in the plot of the **SSE (Sum of Squared Errors)**, but since it was hard to spot such *elbow* in our data, the **silhouette score** was added to this plot too, as it can be observed in Figure 2.1.

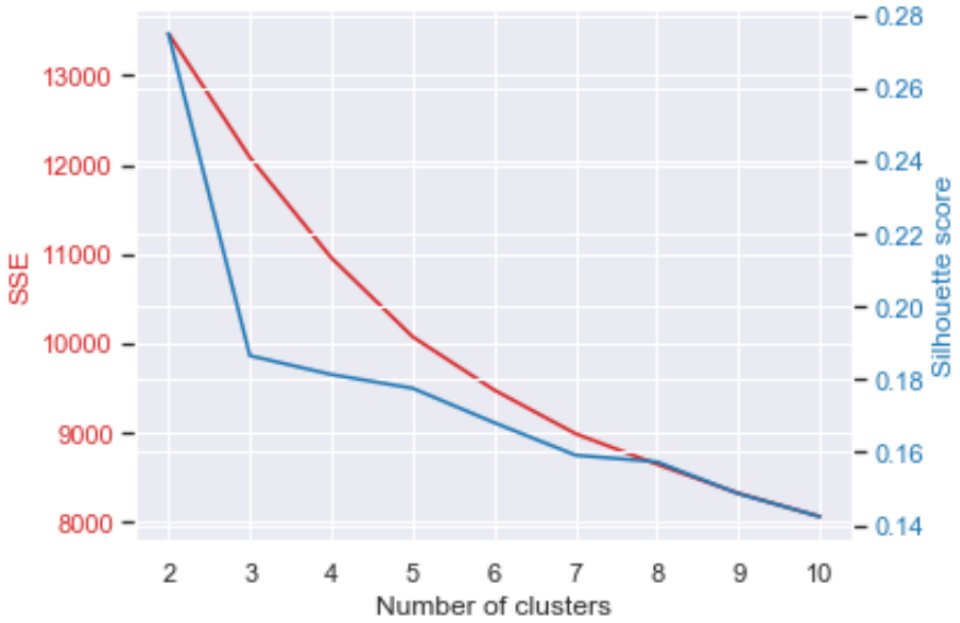


Figure 2.1 SSE vs silhouette score

As reflected in Table 2.1, there is not a unique choice that seems extremely better than the others because of the SSE-silhouette **trade-off**. The two best k amongst the explored ones seemed to be 5 and 8 and, after some further investigation, **5** was selected as the **ideal number of clusters** for this algorithm. Here follows, in Figure 2.2, 2.3 and 2.4 some visual results showing where the 5 centroids of the individuated clusters collocate in the various distributions and how the categorical variables are distributed in them.

k	SSE	Silhouette score
2	13466.56	0.2751
3	12081.75	0.1866
4	10958.77	0.1814
5	10074.46	0.1776
6	9479.87	0.1681
7	8990.45	0.1592
8	8648.33	0.1574
9	8324.24	0.1486
10	8063.48	0.1423

Table 2.1 SSE and silhouette scorse for k $\in (2, 10)$

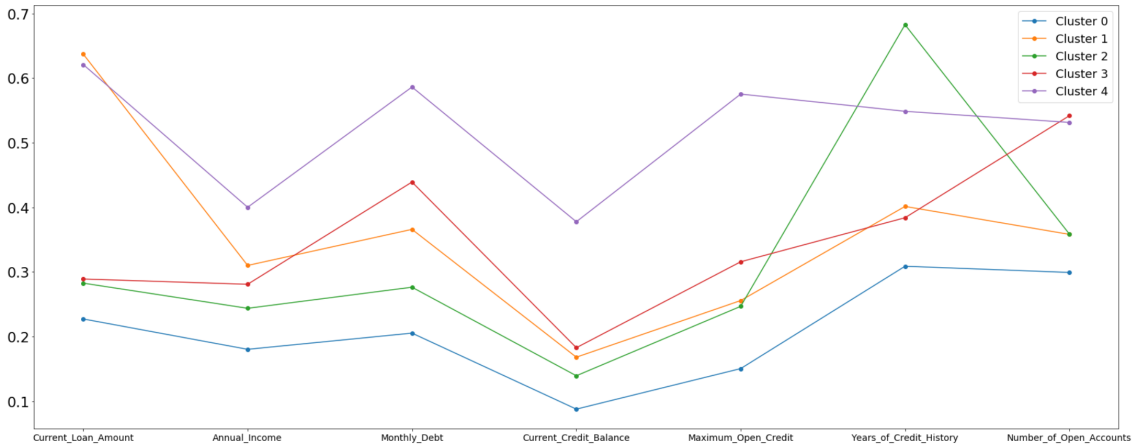


Figure 2.2 Parallel centroids of the k-means clustering

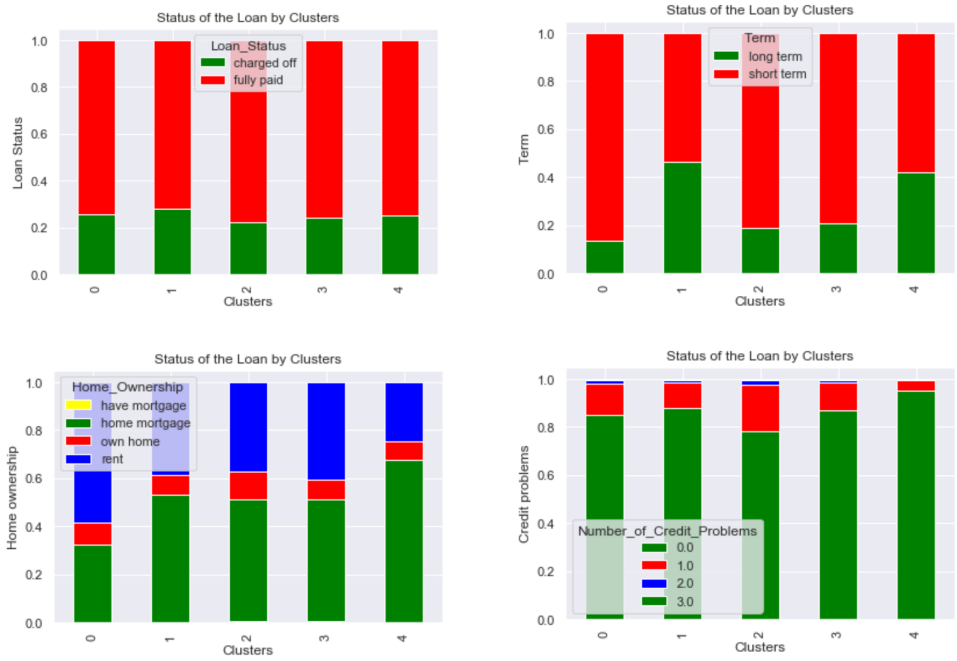


Figure 2.3 Variables distributions within clusters

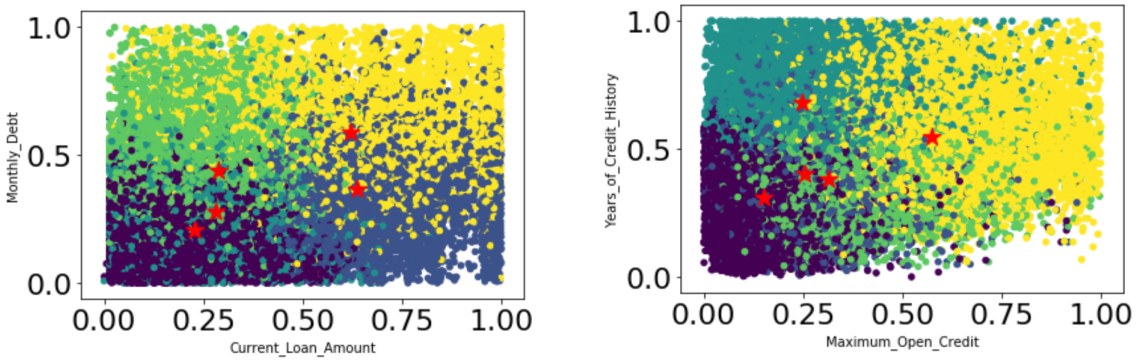


Figure 2.4 The clusters' centroids in the overall distribution

2.2 Analysis by density-based clustering

Switching to another class of clustering, the density-based, the selected algorithm was **DBSCAN**. The categorical features as well as the *Number of credit problems* were excluded, but the variable *Credit score* was considered for the analysis in this case. The metric used for this algorithm was the **euclidean distance**. Differently from the previous algorithm, DBSCAN needs two parameters to be tuned, **epsilon** and **min points**. While for the former there is a method to find the optimal value, for the latter there are a couple of heuristics such as using the double of the number of features considered or the number of features+2, which were the values explored in this study. Once the values for min points were selected, k-nearest neighbors was applied using 10 and 16 as the value for k. The distance from each point to the 10^{th} or 16^{th} closest neighbour was then sorted increasingly and plotted. Figure 2.5 shows such plot for the 16^{th} nearest neighbour (the one for the 10^{th} nearest neighbour was very similar); in such plot the optimal value for epsilon will be found at the point of maximum curvature. In Table 2.2 it is possible to observe some of the results obtained for different values of epsilon.

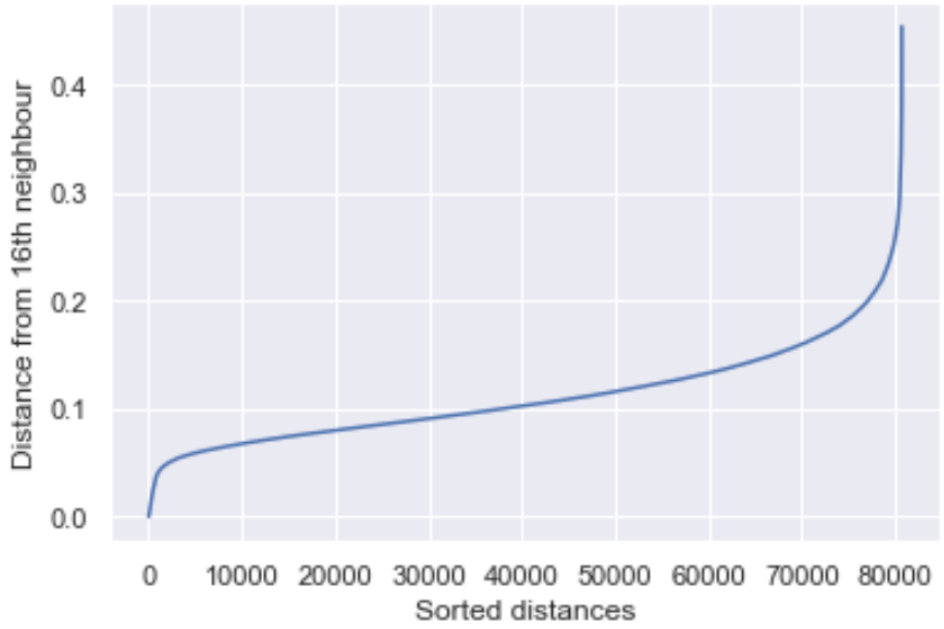


Figure 2.5 Distance from 16^{th} neighbour

Min Points	Epsilon	N°of Clusters	Cluster composition
10	0.15	14	-
10	0.16	10	-
16	0.17	2	{21992, 53794}
16	0.18	4	{17917, 62970, 6, 10}
16	0.19	4	{14431, 69367, 13, 2}
10	0.20	5	-
10	0.21	6	-
10	0.22	5	-
10	0.23	3	{3894, 76823, 6}
16	0.24	2	{3899, 76824}
16	0.25	2	{2983, 77740}

Table 2.2 DBSCAN parameter tuning

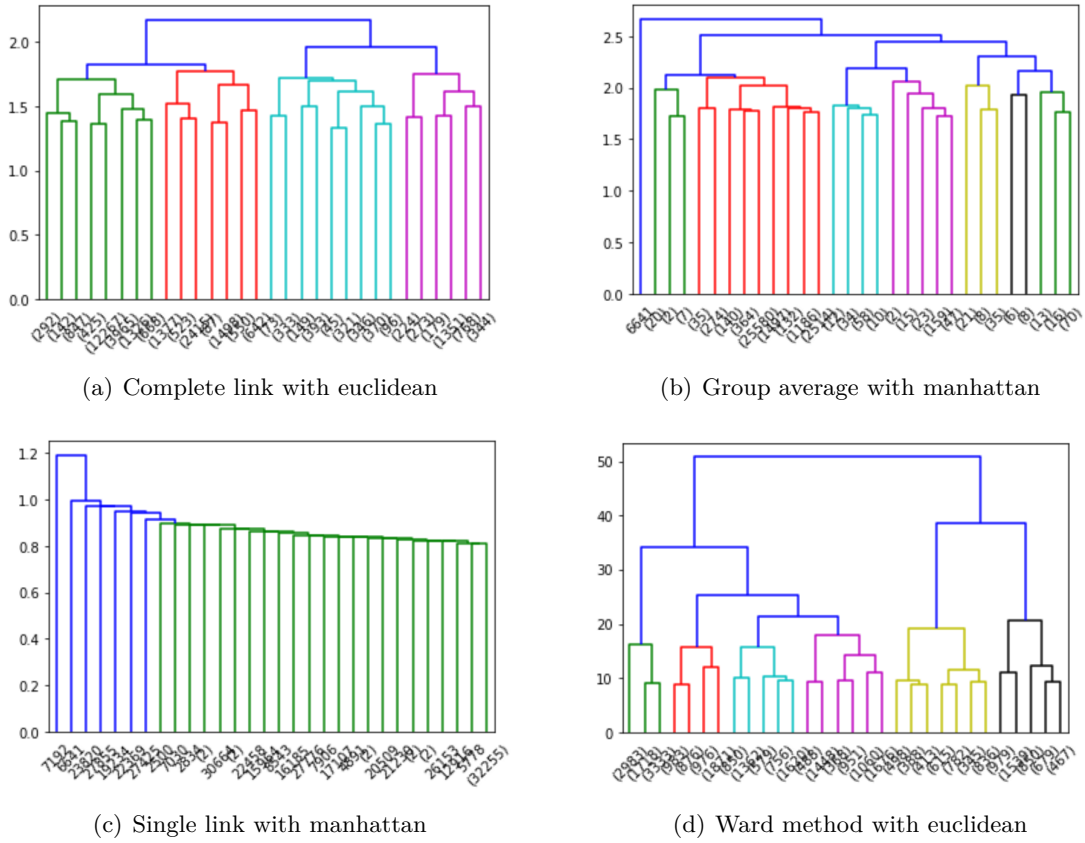
Since the results were not very satisfactiory, **OPTICS** algorithm was also implemented. Unfortunately, the results did not improve and therefore it would have been meaningless to include them in this report. This highlighted that the dataset composition is not keen on performing well with density-based clustering, probably because of a **low variation of the density** of the points. In fact during the parameter tuning it was evident that the ones performing better in terms of silhouette score had a really bad cluster composition (one cluster containing approximately 85% of the records). The selected couple of parameters was min points=**16** and epsilon=**0.17**, which had a quite good performance in terms of silhouette score (0.2366). Unfortunately some further investigation on the obtained clusters highlighted that this result was probably due to the presence of a big component in the data and not to an actually well-performing clustering approach.

2.3 Analysis by hierarchical clustering

The last class of clustering techniques implemented was the hierarchical one, specifically the **agglomerative** type. Unfortunately one of the drawbacks of this algorithm is its computational cost, which in many cases is $O(n^3)$; for this reason the dataset was undersampled beforehand, obtaining a working copy composed of 32,289 records, holding the same properties and distributions as the original one. Once completed this necessary process, the analysis was performed using different **linkage criteria** (i.e. single, complete, group average and Ward method) both with the **euclidean** and **Manhattan** distances. For every combination of method and metrics (Ward+manhattan excluded) a **dendogram** was plotted and, after some investigation, the optimal values for the cut were found. In Table 2.3 it is possible to appreciate the number of clusters that every combination generated, while Figure 2.6 shows the best dendograms obtained for each linking method. The selected features were the same as the one used in density-based clustering. By looking at the different dendograms, the best combinations individuated were **complete link** and **Ward method**, both with euclidean distance as metric. These two compositions were therefore explored more deeply: the former resulted in 4 clusters composed as follows {18319, 6915, 54221, 1268} while the latter generated 6 clusters, whose composition was the following {26015, 18766, 8102, 10015, 10816, 7009}. Even though the ward method seemed to give a better cluster composition, its silhouette score (0.064) was a way lower than the one resulting from the complete link method (0.1464).

Method	Metric	N°of Clusters
Complete	euclidean	4
Complete	manhattan	4
Average	euclidean	8
Average	manhattan	8
Single	euclidean	13
Single	manhattan	8
Ward	euclidean	6

Table 2.3 Hierarchical clustering method selection



(a) Complete link with euclidean

(b) Group average with manhattan

(c) Single link with manhattan

(d) Ward method with euclidean

Figure 2.6 Dendograms with different methods and metrics

2.4 Comparison and final evaluation of the best approach

As already mentioned in Section 2.2 density-based clustering seemed inappropriate for the analyzed dataset and therefore it was not considered, due to its uncapability to lead to actual clusters. The final decision was then between k-means and hierarchical clustering. Given the very low silhouette score obtained by applying the hierarchical clustering with the Ward method, that should not be considered as the best option. The final decision was then between hierarchical clustering with complete link and 4 clusters and k-means with 5 clusters: both silhouette score and clusters' composition point the latter as the best algorithm to cluster the dataset. Please note that even though this was the best solution amongst the ones explored, it is still an unsatisfactory result, which indicates that the dataset was **not suitable** for clustering or at least not for this basic techniques.

3 Association Rules

In this section we present the second task of the analysis: association rules/pattern mining, where the goal was to find **frequent itemsets** and interesting **association rules** in the dataset. Before starting to explore all the possible frequent itemsets, it was necessary to apply some preliminary transformations. In order to implement the ***apriori*** algorithm, all the features needed to be transformed into binary variables (1 - true, 0 - false), which meant creating **dummy variables** for every value contained in the records. Unfortunately continuous attributes cannot be transformed directly into dummies, so firstly they needed to be discretized. The choice was to **discretize** each feature into 10 intervals, accordingly to its distribution: each interval corresponded to a decile, which means that the new values from 1 to 10 indicated if the value was respectively in the 10^{th}, 20^{th}, 30^{th} etc. percentile of that distribution. After this first step, all the attributes were transformed into dummy variables, obtaining a **80723 rows** $\times$ **113 columns** dataset.

3.1 Frequent patterns extraction

On the new dataset the apriori algorithm was finally implemented with different values for the **minimum support** (from 0.2 to 0.8). In Figure 3.1 it is possible to notice that the number of itemsets decreases quite significantly in the first steps, from 0.2 to 0.4.

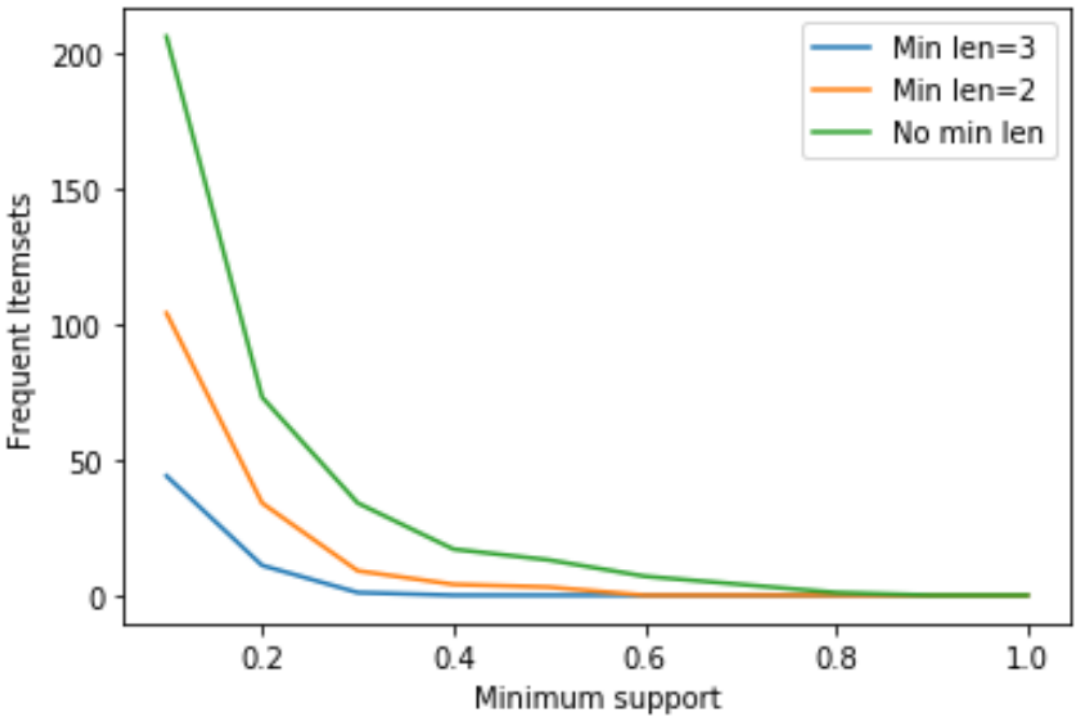


Figure 3.1 Frequent itemsets number per min-support

The Table 3.1 shows how some of the itemsets obtained are composed by a single variable, such as *Loan Status, Term, Purpose* and *Number of Credit Problems*; this result derives from the unbalance of such features already addressed in Section 1. Moreover, all the bigger itemsets we can see in the table are composed by these variables, which gave us a hint we might have some difficulties in extracting some useful information in this section.

Support	Itemset
0.748535	Loan Status = fully paid
0.747978	Term = short term
0.794321	Purpose = debt consolidation
0.860672	Number of Credit Problems = 0
0.584778	Term = short term & Loan Status = fully paid
0.593883	Purpose = debt consolidation & Loan Status = fully paid
0.644352	Number of Credit Problems = 0 & Loan Status = fully paid
0.585831	Purpose = debt consolidation & Term = short term
0.640264	Number of Credit Problems = 0 & Term = short term
0.684192	Number of Credit Problems = 0 & Purpose = debt consolidation
0.500911	Number of Credit Problems = 0, Term = short term & Loan Status = fully paid
0.512308	Number of Credit Problems = 0, Purpose = debt consolidation & Loan Status = fully paid
0.500948	Number of Credit Problems = 0, Purpose = debt consolidation & Term = short term

Table 3.1 Frequent itemsets with support bigger than 0.5

3.2 Association rules extraction

Because of the high decrease in the number of generated itemsets with a minimum support bigger than 0.2, we decided to use that value as a threshold and perform the extraction of the association rules based on **confidence** and **lift** values. Each rule is made by an antecedent and a consequent itemsets and confidence measures how often items in the consequent appear together with items that contain the antecedent. Obviously, since our dataset is not a trantactional one, in our study items do not refer to actual items, but to a record having a specific value for a specific attribute. Unfortunately, the high unbalanced distributions in some variables, i.e. *Number of Credit problem* which has an 86% of 0 values, make it very hard to extract significant rules basing the selection on confidence, since the high value of this measure is often not related to the goodness of the rule itself, but on the high frequency of the consequent itemset in such rule. For this reason, the lift method was preferred, which takes into account also the statistical dependence of the antecedent and the consequent and gives a better insight on how much information we actually gain by extracting the rule. The antecedent and consequent are statistically indipendent if lift takes value 1, they are negatively associated if it is lower and positively if it is higher. Therefore, rules with lift bigger than 1.1 were investigated and some of them are shown in Table 3.2. In the table it is possible to see how renting an apartment and having been working in the current position for less than 5 years are highly correlated (the former influences the latter and viceversa), which is quite logical given the high investment needed to purchase a house, probably not affordable for someone who has just started working. We can also see how not having any previous credit problem and a short term loan seem to bring to a fully paid loan, also implying that those clients are renting their house. The fact that the six rules in the table are in fact three couples where antecedent and consequent are swapped, it indicates that instead of actual rules, they are more likely simple **concurrencies** of such values.

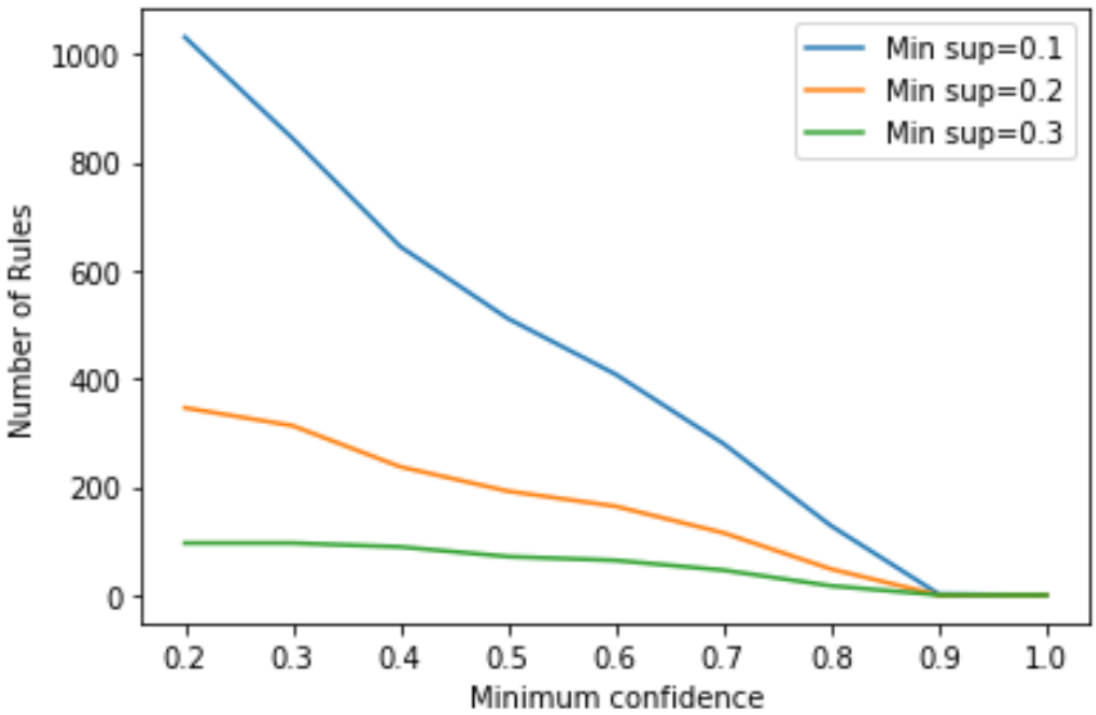


Figure 3.2 Frequent itemsets number per min-support

Antecedent	**Consequent**	**Confidence**	**Lift**
Home Ownership = rent	Years in current job = 1-5 years	0.461244	1.182827
Years in current job = 1-5 years	Home Ownership = rent	0.517218	1.182827
Home Ownership = rent & Loan Status = fully paid	Term = short term & Purpose = debt consolidation	0.669172	1.142262
Term = short term & Purpose = debt consolidation	Home Ownership = rent & Loan Status = fully paid	0.361514	1.142262
Home Ownership = rent & Loan Status = fully paid)	Number of Credit Problems = 0 & Term = short term	0.722287	1.128109
Number of Credit Problems = 0 & Term = short term	Home Ownership = rent & Loan Status = fully paid	0.357035	1.128109

Table 3.2 Rules with lift bigger than 1.12

3.3 Missing value replacement

Unfortunately, in the association rule extraction, **no consistent rule** emerged that could somehow be helpful in replacing the missing values addressed in Section 1.3. In fact, the columns presenting such problem did not appear as a consequent in the rules with high confidence or lift. Therefore, the way missing values were handled still holds as explained in Section 1.3.

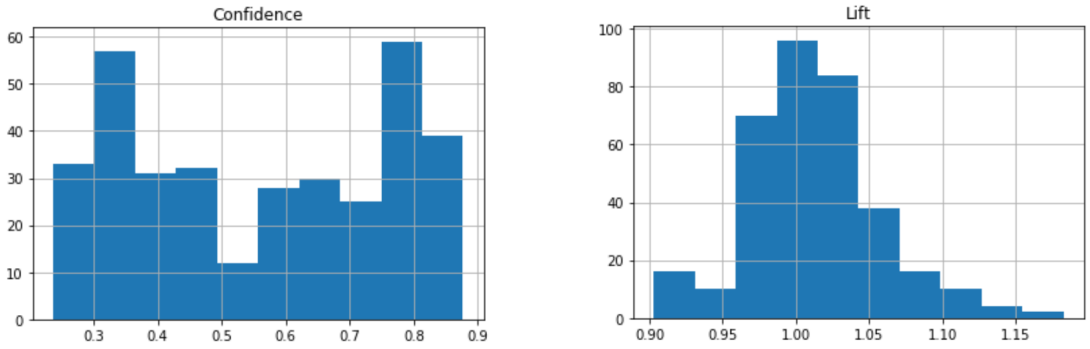


Figure 3.3 Histograms of rules' confidence and lift

3.4 Target variable prediction

The last thing we tried to implement was a **label prediction** for the target variable. By extracting the rules which had *Loan Status = fully paid*, we selected amongst them the ones with the highest confidence and a significant lift, which can be observed in Table 3.3. It seems pretty clear how the shortage of the term of the loan influences its full payment. In fact short term loans usually have low interest rates and cannot exceed certain amounts of money and this might be the reason behind this result. We used these rules to label the *Loan Status* as *fully paid*, while all the other instances which did not match these conditions were set as *charged off*. Due to the high unbalance in the target label it was impossible to extract a significant rule to label *Loan Status = charged off*. By applying this method the obtained number of **true positives** (fully paid predicted correctly) was 46773, the number of **false positives** (charged off predicted as fully paid) was 13065, the **false negatives** (fully paid predicted as charged off) were 13651 and **true negatives** (charged off predicted correctly) 7234. The **accuracy** of this method is 66.9%, which is very low considering that the dumbest classifier, which lables each record as *fully paid*, has a 74.85% of accuracy because of the unbalance in the data. For this reason, more efficient classification techniques will be explored in Section 4.

Antecedent	**Consequent**	**Confidence**	**Lift**
Years in current job = 1-5 years & Term = short term	Loan Status = fully paid	0.783253	1.046382
Home Ownership = home mortgage & Term = short term	Loan Status = fully paid	0.812729	1.085760
Purpose = debt consolidation & Term = short term	Loan Status = fully paid	0.782872	1.045872
Number of Credit Problems = 0 & Term = short term	Loan Status = fully paid	0.782350	1.045175

Table 3.3 Rules with fully paid status as a consequent

4 Classification

The last task of this study was classification, whose purpose is to build an accurate model that is able to predict whether a given loan is or will be fully paid or charged off. In order to do that, six different classifiers were explored: Decision Tree, Random Forest, Support Vector Machine, Neural Network, K-Nearest Neighbours, and Naïve Bayes. All these methods will be applied to the usual dataset and four variations of it obtained through some resampling methods.

4.1 Resampling methods

As already addressed in Section 1.2, the dataset has an unbalanced composition towards the *fully paid* label for the *Loan Status* attribute (75% of the records), hence **random resampling** methods (both oversampling and undersampling) were applied. Please note that the new examples that are generated by the random oversampling technique do not add any new information to the model. On the other hand, **Synthetic Minority Oversampling Technique (SMOTE)**, which synthesizes new examples from the existing ones, was also implemented. SMOTE works by selecting records that are close in the feature space, drawing a line between them and generating a new sample on a point along that line. The last approach explored was **Adaptive Synthetic Sampling (ADASYN)** which generates synthetic samples inversely proportional to the density of the instances in the minority class, thus generating more synthetic examples in regions of the feature space where the density of records from the minority class is low and fewer or none where such density is high. These resampling methods were used because the bias in the dataset could influence many machine learning algorithms, leading some of them to entirely ignore the minority class.

4.2 Decision Tree vs. Random Forest

4.2.1 Hyperparameters tuning and validation

Decision Tree and Random Forest were applied to the imbalanced, undersampled, oversampled, *smoted* and *adasyned* datasets, which were all divided as follows: 80% was used for training and 20% for testing. Continuous attributes were scaled between 0 and 1, while categorical ones were transformed via the one hot encoding technique. Before applying the **K-Fold Cross-Validation** with parameter tuning using the **Grid Search** technique, experimental trials were done in order to find a range of 'good' hyperparameters, with the goal of reducing the computational time of the process. Once completed, the following step was applying Grid Search with K-Fold Cross-Validation, using K=3. Here follows the searched **hyperparameters for Decision Tree**:

- **Imbalanced** dataset: *criterion* = ['Gini', 'entropy'], *max_depth* = (1, … , 10), *splitter* = ['best', 'random'], *min_impurity_decrease* = [1e-16, 1e-14], *max_features* = [None, 'auto', 'sqrt']
- **Oversampled** dataset: *criterion* = ['Gini', 'entropy'], *max_depth* = (30, … , 50), *splitter* = ['best', 'random'], *min_impurity_decrease* = [1e-16, 1e-14], *max_features* = [None, 'auto', 'sqrt']
- **Undersampled** dataset: *criterion* = ['Gini', 'entropy'], *max_depth* = (1, … ,20), *splitter* = ['best', 'random'], *min_impurity_decrease* = [1e-05, 1e-04], *max_features* = [None, 'auto', 'sqrt']

- **Smoted** and **Adasyned** dataset: *criterion* = ['Gini', 'entropy'], *max_depth* = (1, … , 20), *splitter* = ['best', 'random'], *min_impurity_decrease* = [1e-16, 1e-14], *max_features* = [None, 'auto', 'sqrt']

and the explored **hyperparameters Random Forest**:

- **Imbalanced dataset, Undersampled dataset**: *criterion* = ['Gini', 'entropy'], *max_depth* = (1, … , 20), *n_estimators*=[50, 70]
- **Oversampled dataset**: *criterion* = ['Gini', 'entropy'], *max_depth* = (30, … , 50), *n_estimators*=[50, 70]
- **Smoted dataset, Adasyned dataset**: *criterion* = ['Gini', 'entropy'], *max_depth* = (10, … ,30), *n_estimators*=[50, 70]

The results obtained by the the grid search are shown in the Table 4.1.

	Best hyperparameters		**Best score**	
Dataset	**Decision Tree**	**Random Forest**	**Decision tree**	**Random Forest**
Imbalanced	*criterion = entropy* *max_depth = 4* *splitter = random* *min_impurity_decrease = 1e-16* *max_features = auto*	*criterion = Gini* *max_depth = 12* *n_estimators = 70*	0.7491	0.7498
Oversampled	*criterion = entropy* *max_depth = 43* *splitter = best* *min_impurity_decrease = 1e-16* *max_features = auto*	*criterion = Gini* *max_depth = 38* *n_estimators = 70*	0.7752	0.8411
Undersampled	*criterion = Gini* *max_depth = 9* *splitter = random* *min_impurity_decrease = 1e-4* *max_features = None*	*criterion = entropy* *max_depth = 10* *n_estimators = 70*	0.5832	0.6014
SMOTED	*criterion = entropy* *max_depth = 9* *splitter = random* *min_impurity_decrease = 1e-16* *max_features = None*	*criterion = Gini* *max_depth = 29* *n_estimators = 70*	0.7996	0.8226
ADASYNED	*criterion = Gini* *max_depth = 10* *splitter = random* *min_impurity_decrease = 1e-16* *max_features = None*	*criterion = Gini* *max_depth = 29* *n_estimators = 70*	0.8033	0.8253

Table 4.1 Best configurations of Decision Tree and Random Forest classifiers

4.2.2 Model Evaluation

Once the best corresponding models for both classifiers were chosen, they were retrained and tested on the various test sets. The results obtained by such procedure are presented in Table 4.2, while the results of the Random Forest models retrained and tested on the before mentioned datasets are shown in Table 4.3.

Dataset	Target values	Precision		Recall		F1-score		AUC		Accuracy	
		training	test	training	test	training	test	training	test	training	test
Imbalanced	charged off	0.6000	0.600	0.0007	0.0007	0.0015	0.0015	0.5	0.5	0.7492	0.7463
	fully paid	0.7492	0.7463	0.9998	0.9998	0.8566	0.8547				
Oversampled	charged off	0.9972	0.7813	0.9993	0.9333	0.9983	0.8505	1.00	0.83	0.9983	0.8353
	fully paid	0.9993	0.9163	0.9972	0.7366	0.9983	0.8167				
Undersampled	charged off	0.5852	0.5666	0.6134	0.5990	0.5989	0.5823	0.59	0.57	0.5886	0.5734
	fully paid	0.5923	0.5809	0.5636	0.5482	0.5776	0.5641				
SMOTED	charged off	0.9746	0.9688	0.6244	0.6247	0.7611	0.7596	0.8	0.8	0.8042	0.8015
	fully paid	0.7241	0.7214	0.9837	0.9797	0.8342	0.8309				
ADASYNED	charged off	0.9707	0.9609	0.6434	0.6363	0.7739	0.7656	0.81	0.80	0.8089	0.8006
	fully paid	0.7267	0.7184	0.9800	0.9729	0.8346	0.8265				

Table 4.2 Results obtained with the best Decision Tree models

Dataset	Target values	Precision		Recall		F1-score		AUC		Accuracy	
		training	test	training	test	training	test	training	test	training	test
Unbalanced	charged off	0.9926	0.7414	0.0580	0.0105	0.1096	0.0207	0.53	0.5	0.7636	0.7480
	fully paid	0.7601	0.7480	0.9999	0.9988	0.8637	0.8554				
Oversampled	charged off	1.000	0.9009	1.000	0.9216	1.000	0.9112	1.00	0.91	1.000	0.9098
	fully paid	1.000	0.9191	1.000	0.8978	1.000	0.9083				
Undersampled	charged off	0.7276	0.5926	0.7057	0.5809	0.7240	0.6003	0.72	0.59	0.7203	0.5936
	fully paid	0.7134	0.5945	0.7349	0.6062	0.5776	0.5641				
SMOTED	charged off	1.000	0.9518	0.9949	0.6963	0.9975	0.8042	1.00	0.83	0.9975	0.8298
	fully paid	0.9950	0.7590	1.000	0.9644	0.9975	0.8495				
ADASYNED	charged off	1.000	0.9481	0.9917	0.7057	0.9958	0.8091	1.00	0.83	0.9958	0.8296
	fully paid	0.9915	0.7567	1.000	0.9595	0.9957	0.8461				

Table 4.3 Results obtained with the best Random Forest models

4.2.3 Interpretation of Decision Trees

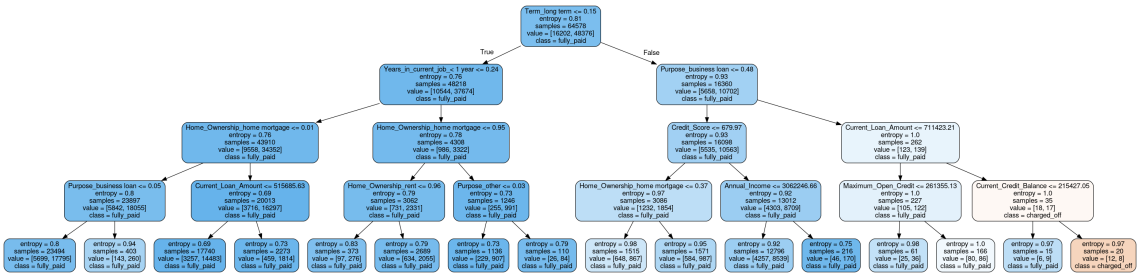


Figure 4.1 Decision Tree on the Imbalanced Dataset

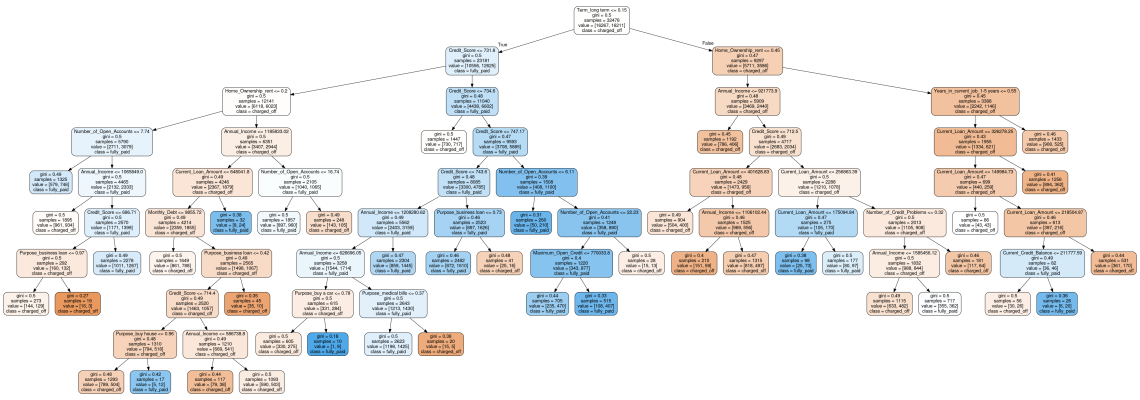


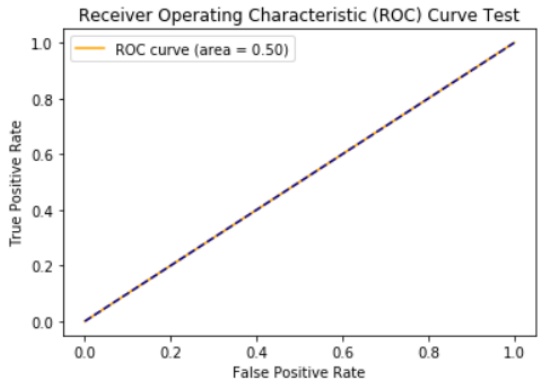
Figure 4.2 Decision Tree on the Undersampled Dataset

The first thing that should be mentioned is that all the Decision Trees generated for the various datasets identify the attribute *Term* as the most important for classifying the records. Such result is not surpsising and reflects what has been already observed in Section 3.4. In the Figures 4.1 and 4.2 are presented Decision Trees applied to the imbalanced and undersampled datasets, since the others were too large to be included in this report. From the Decision Tree applied to the imbalanced dataset it emerges that the variables with a high **feature importance** are *Years in current job*, *Purpose* and *Home Ownership*. On the other hand, when referring to results obtained on the unbalanced dataset, an important role is played also by *Credit Score, Number of open accounts, Current Loan amount* and *Annual Income*. An interesting observation is that in most of the records where the client rents the house where he/she lives, the loan status was *charged off*.

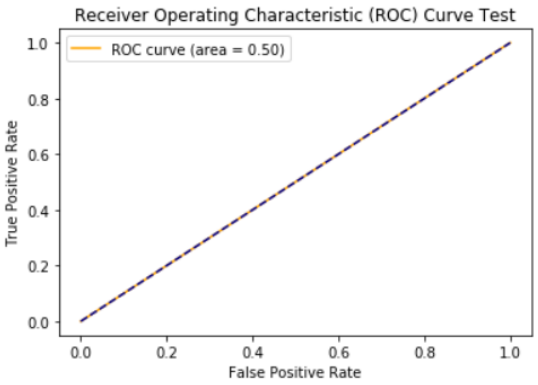
4.3 Discussion of the best prediction model

The final model of the Decision Tree classifier retrained and tested on the imbalanced dataset has **accuracy** 74.92% and 74.63% respectively, which are extremely close the percentages of *fully paid* records presented in the training and test set. In fact, such model learnt to predict almost everything as belonging to the majority class (99.96% of the records were predicted as *fully paid*). The resulting **ROC** curve is the bisector, as it can be seen in Figure 4.3, and the **AUC (area under the curve)** score is equal to 0.5 both during training and test phase, thus the model has no discrimination capacity to distinguish between the two classes. The results do not improve much with the Random Forest, performing with a 76.36% on the training set and an AUC score of 0.53. Moving to the oversampled datasets, it is observed from the Tables 4.2 and 4.3 that the final models of both classifiers performed very well during the training phase, with an accuracy of 99.83% and 100% respectively, but the Decision Tree classifier behaved poorly on previously unseen data. It reached an accuracy of 83.53%, with an AUC score of 0.83, while the Random Forest classifier reached a much better accuracy, equal to 90.98%, and an AUC score of 0.91. For what concerns the undersampled datasets, unfortunately they presented a too small number of records and so both classifiers behaved poorly when applied to them. Reffering to the smoted datasets, we can see that the Decision Tree reached an accuracy of a bit more than 80% for both training and test phase, while the Random Forest showed great performances during the training phase, reaching an accuracy of 99.75%, but it was not able to generalize well on unseen data, where it reached accuracy of 82.98%. Finally, on the adasyned datasets we can observe that the final models of both classifiers performed almost the same as the final models applied to the smoted dataset. Here follows some further observetions:

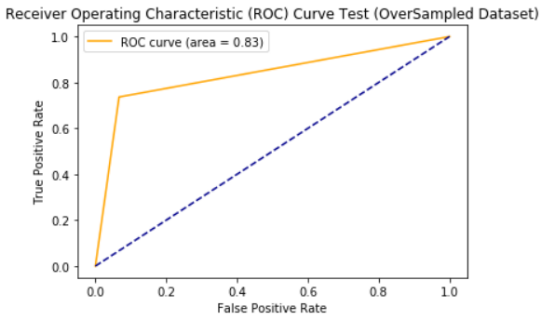
- even though, as Figure 4.4 shows, accuracy increases together with the depth of the tree, the final model of the Decision Tree classifier applied to the oversampled dataset led to overfitting, since its depth is pretty high (43);
- the best model of the Decision Tree classifier was the one applied to the smoted dataset, where the depth of the tree is 9 and entropy is used as criterion; the AUC score of that model is 0.8 both on the training and test set;
- the best overall prediction model is the Random Forest classifier trained and tested on the oversampled dataset, since it reached accuracy of 90.98% and AUC score 0.91; it uses gini as criterion and creates 70 trees with a maximum depth of 38.



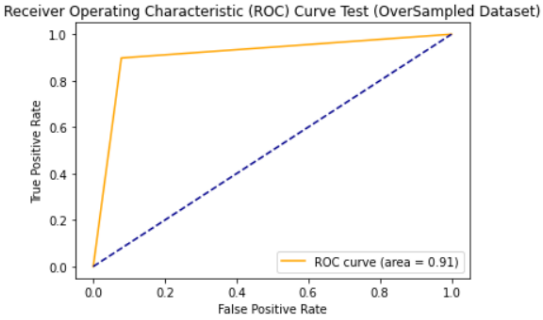
(a) Decision tree on the imbalanced dataset



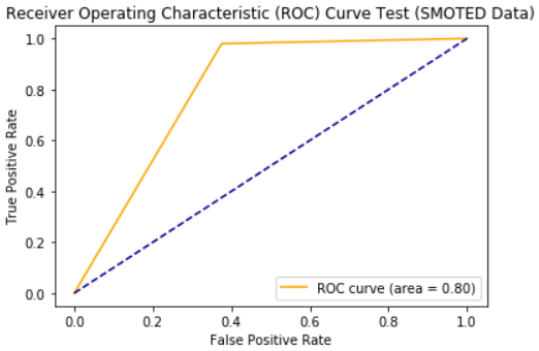
(b) Random Forest on the imbalanced dataset



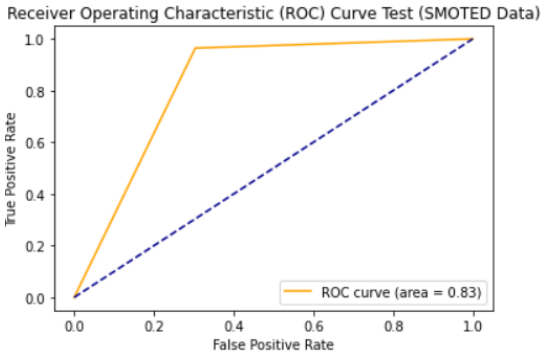
(c) Decision tree on the oversampled dataset



(d) Random Forest on the oversampled dataset



(e) Decision tree on the adasyned dataset



(f) Random Forest on the adasyned dataset

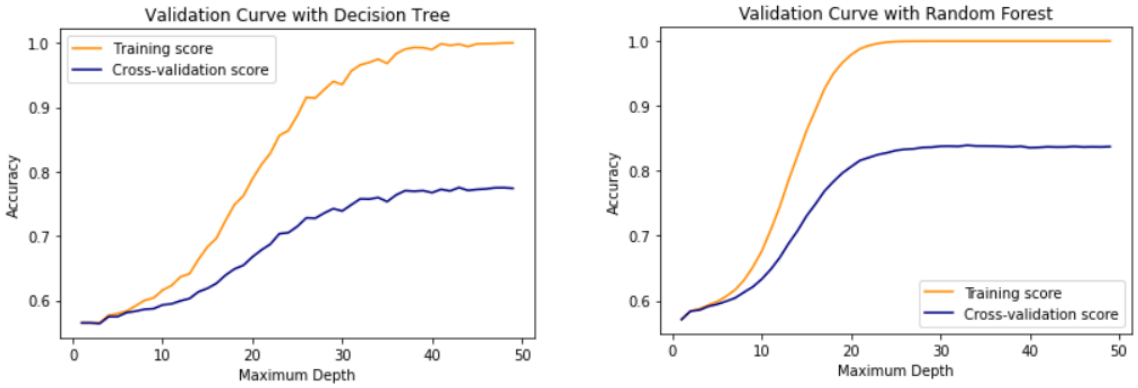
Figure 4.3 ROC curves of Decision Tree and Random Forest classifiers on test sets

4.4 Advanced classifiers

As anticipated in this section's introduction, further classification methods were explored. These classifiers were applied to the imbalanced, undersampled, oversampled, smoted and adasyned dataset as before. In order to find the best model that corresponds to each classifier K-Fold Cross-Validation with parameter tuning using Grid Search was used, with K chosen equal to 3. In the following subsections, only the final results will be presented.

4.4.1 Neural Network vs. Support Vector Machine vs. Naïve Bayes

The final models of the Neural Network and Support Vector Machine classifiers, retrained and tested on the imbalanced dataset provided an accuracy of 74.91% on the training set,



(a) Decision Tree classifier

(b) Random Forest classifier

Figure 4.4 Accuracy vs. Depth

while they reached 74.62% on the test set. **Naïve Bayes** classifier provided accuracy of 74.89% on the training set, while on the test set, the accuracy was 74.61%. The final **Neural Network** model consisted of:

- two hidden layers, where the first layer was made by 50 neurons, while the second layer consisted of 100 neurons;
- learning rate = 0.1;
- momentum, whose value is 0.7, where Nesterov Momentum is enabled;
- the Adam solver, used for weights optimization.

The final **Support Vector Machine** model consisted of the Radial Basis function kernel.

4.4.2 K-Nearest Neighbours

The **KNN** classifier achieved the best accuracy and the AUC score on the oversampled dataset. The accuracy on the training part of that dataset was 100%, with an AUC score of 1.00, while the accuracy on the testing part was 82.92%, with the AUC score 0.83. Unfortunately, this model, with the number of neighbours euql to one, seemed to bring to an overfitting problem.

4.5 Conclusion

The problems with the imbalanced dataset were not resolved by using the advanced classification techniques explored in Section 4.4. In fact the proportion of the class *fully paid* in the part of the imbalanced dataset used for training is 74.91% and in the one used for testing is 74.62% and the final models of the Neural Network, Support Vector Machine and Naïve Bayes classifier learnt to predict every record as belonging to this class (100% of labels predicted as *fully paid*). According to the AUC score, which is 0.5 in all three cases, both on the training and test sets, the models have no discrimination capacity to distinguish between the two classes just as before. Such models perform poorly on the other datasets, providing an accuracy around 50% both on the training and test sets, and an AUC score smaller than 0.6. For this reason Neural Network, Support Vector Machine and Naïve Bayes should not be chosen for this particular dataset and its variations, while k-nearest neighbours classifier led to overfitting as already mentioned. We can then conclude that the best overall classifier is still the Random Forest trained and tested on the oversampled dataset, with an accuracy of 90.98% and an AUC score equal to 0.91.

5 Final considerations

The provided dataset has a majority of numerical features regarding mainly the credit history of the client the records refer to, while the few categorical attributes provide more general information such as the number of years in their current job or whether they own their house or not. Despite this composition, the dataset performed poorly on clustering as explained in Section 2, probably because, as highlighted in Section 4.2.3, the most important features are the categorical ones. For this reason, more advanced clustering techniques that include also such variables should be tested in order to obtain better defined clusters in this population. Pattern mining allowed us to extract some rules, but they were highly influenced by the unbalanced features such ad *Loan Status* or *Term*. The part of the analysis that gave the most interesting results was surely classification, as reported in Section 4, even though only the Random Forest classifier seemed to bring an actual improvement in the results, while the other tested classifiers (Decision Tree, Neural Network, Support Vector Machine, K Nearest Neighbours and Naïve Bayes) did not perform much better than the simplest one, which assign the label *fully paid* to every record. In any case, a better analysis could have been made by consulting the bank beforehand, in order to understand better the meaning and the calculation behind some variables, such as *Credit Score* or *Current Credit Balance*. It is very likely that the bad performances of some tasks were due to the general imbalance in the dataset since, as addressed many times before, it can be expected for this type of dataset to present a much higher number of 'good creditors' than 'bad creditors', resulting in naturally unbalanced data.