An analysis of a Taiwanese Credit Card Dataset

Maddalena Amendola, Daniele Gadler

Riccardo Manetti, Gemma Martini

Abstract

In the context of banking data analysis, it is crucial to distinguish trustworthy customers that do pay their debts on time from the untrustworthy ones that delay their payments and eventually end up defaulting. Our aim was to provide a tool that would allow us to carry out such distinction on a dataset of Taiwanese credit card holders over a time period spanning from April 2005 to September 2005.

In our analysis, we were confronted with many problems, mainly: the time limitedness of the data, data unbalancing, the presence of missing values and semantic issues. After correcting the missing values and trying to solve the semantic issues, we applied three data mining techniques: clustering, classification and pattern mining.

In an initial data understanding part, we realized that the dataset had serious semantic coherence issues that hindered all the techniques used in our experiments; these techniques included traditional approaches as well as some innovative approaches (e.g. Deep Learning, 10-Fold dataset sub-division).

The claim of this report is that such dataset is not sufficiently accurate to distinguish defaulting customers from non-defaulting customers beyond the metrics and the results presented.

January 9, 2019

1 Data understanding

1.1 Data semantics

The dataset contains information on the credit cards of 10,000 different customers of a Taiwanese bank. For each user, the following information is known:

`Limit:` Numerical attribute that establishes the monthly credit amount granted by the bank to the customer;

`Sex:` Categorical attribute representing the customer's gender;

`Education:` Ordinal attribute that defines the customer's education.

`Status:` Categorical attribute that defines the customers's civil status;

`Age:` Numerical attribute representing the customer's age;

$\texttt{Ps}_m$`:` Payment Status for the April, May, June, July, August, September months. Numerical attribute that defines the state of payment of the customer's expenses. The possible values for each one of the six 'ps' columns with index m are:

$$ps_m = \begin{cases} -2 & \text{if no credit consumption} \\ -1 & \text{if consumption with same-month payment} \\ 0 & \text{if use of the revolving credit made available by the bank} \\ \geq 1 & \text{amount of months by which the payment was delayed} \end{cases} \tag{1}$$

$\texttt{Ba}_m$`:` Billing Amount for the April, May, June, July, August, September months. Numerical attribute that defines the total bill statement of a customer.

$$ba_m = \begin{cases} > 0 & \text{if the customer paid less than its } \text{ba}_{m-1} \\ < 0 & \text{if the customer paid more than its } \text{ba}_{m-1} \\ = 0 & \text{if no consumption} \end{cases} \tag{2}$$

$\texttt{Pa}_m$: Payment Amount. Numerical attribute defining how much was paid in the $m-1$ month. It only admits positive values.

`Credit_default` : categorical attribute and target variable to be predicted. "Yes / No" value indicating respectively whether the customer has had a default or has not had a default.

1.1.1 Semantic analysis formulation

We checked whether all the values of a specific attribute do belong to the domain of the considered attribute and identified semantic errors for the `age` and `ps` attributes. For the `age` attribute, we have found the value '-1' and we will deal with it as a missing value in Section 1.2, while for the `ps` attribute we have laid out four different correction rules:

RULE 1: If $\text{pa}_m > \text{ba}_{m-1}$ the customer has paid more than the due amount. In this situation, the ps_{m-1} ought to be -1. The error percentage, namely not respecting this rule, is 22.80%.

RULE 2: If ba ≤ 0 and it is equivalent to the difference between the previous ba and the corresponding sum paid, then we expect a ps = -2 (no consumption). The error percentage, namely not respecting this rule, is 32.92%.

RULE 3: If $\text{ps}_m \geq 0$ (indicating a postponement of the payment), then it is expected that the $\text{ba}_{m+1} \geq \text{ba}_m - \text{pa}_{m+1}$. The error percentage, namely not respecting this rule, is 2.25%.

Attribute Name	Issue type	Amount of values
sex	missing values, 'NaN'	100
education	missing values, 'NaN'	127
status	missing values, 'NaN'	1822
age	invalid values, '-1'	951

TABLE 1: Type and amount of missing and invalid values for the attributes containing missing and invalid values

1.2 Data quality

After careful analysis of the amount of missing values reported in Table 1, we decide to correct missing values in an attribute-by-attribute manner in the following order:

1. *Sex*: All the 100 'NaN' values of the `sex` attribute are replaced by the mode of the `sex` attribute, as it is a categorical attribute.
2. *Education*: All the 127 'NaN' values of the `education` attribute are replaced by the 'other' value, as we do not have any further information that could help us infer the education level of the considered customer.
3. *Status*: All the 1822 'NaN' values of the `status` attribute are replaced by the mode based both on the `sex` and `education` attributes. To do this, this we have grouped the rows by `sex` and `education`, then we have picked the group-wise mode of `status` attribute, as it is a categorical attribute.
4. *Age*: All the 951 values of the `age` attribute having a '-1' value are replaced by the median of the `age` attribute, taken group-wise over the `sex`, `education` and `status` attributes. As Figure 1 shows, this approach results in a more uniform dataset modification with respect to the usage of the group-wise median of the age taken over the `sex` and `education` attributes or the `sex` and `status` attributes, shown in Figure 1

Figure 2 shows the Quantile-Quantile (QQ) Plot of the `age` attribute. In particular, Figure 2 shows the QQ Plot of the `age` attribute **before** and **after** the correction operation is applied. We notice that the distribution of the `age` attribute follows a more normal distribution after the four correction operations have been applied to the `sex`, `education`, `status` and `age` attributes.

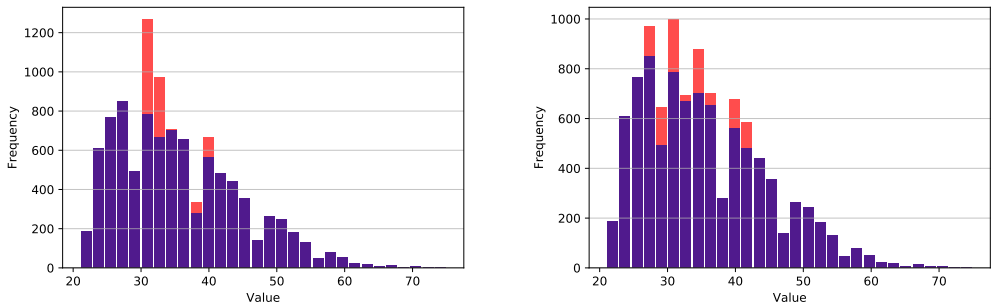

FIGURE 1: On the left-hand side, the age histogram where "-1" values are substituted by the age median of rows grouped by `sex`, `education` and `status` attributes. On the right-hand side, an age histogram, where "-1" values are substituted with the age median of rows grouped by sex, education and status attributes. Bars in blue represent the values' frequency before correction; bars in red represent values' frequency after correcting "-1" values.

1.2.1 Outliers' analysis

In the next few paragraphs, we discuss the adopted methodology for the identification of outlier data points. Only non-categorical attributes are considered as candidates for having outliers.

The first approach pursued for the outlier detection and removal phase was an **algebraic** one, but it did not produce satisfactory results; therefore, we followed a **visual inspection** approach.

For what concerns the `Age` and `Limit` attributes, we notice that it is perfectly reasonable that there exist fewer old customers w.r.t young customers, because it is expected that fewer old people use credit cards with respect to young people. Likewise, it is equally acceptable that there exist only few customers

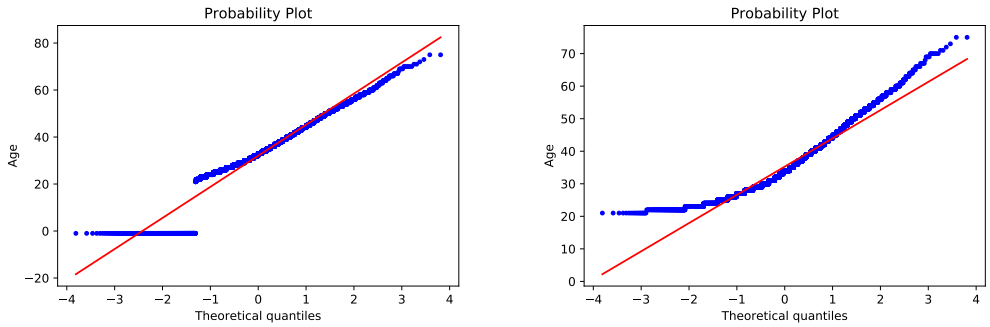

FIGURE 2: On the left-hand side, Initial QQ plot of the `age` attribute, also containing "-1" values. On the right-hand side, the QQ Plot of the `age` attribute after "-1" values' correction.

that have a very high limit at their disposal. For this reason we did not identify any outlier in the `Age` and `Limit` attributes.

The outliers were identified in the $\texttt{ba}_m$ and $\texttt{pa}_m$ attributes by means of a visual analysis of their boxplots. From Figure 3 and Figure 4, we argue that the dots below the lower whisker are outliers because they are extremely few and could potentially skew the dataset, hence the corresponding rows were dropped from the dataset. As a result of the ouliers' removal process we dropped 30 rows from the dataset, which now has 9970 rows.

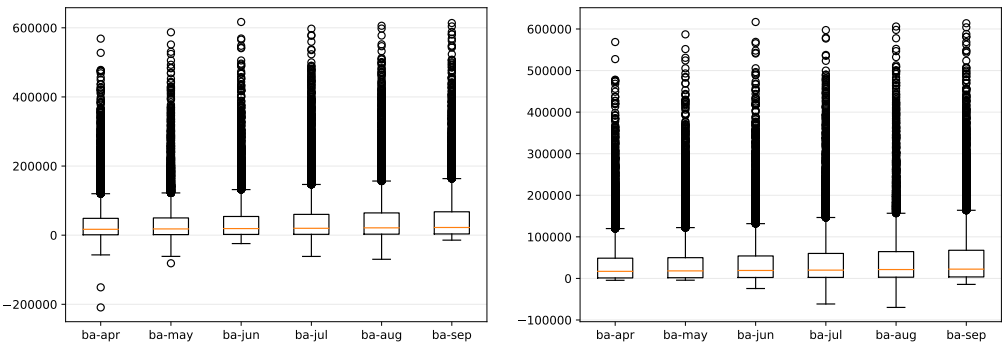

FIGURE 3: On the left-hand side, a boxplot of all `ba` columns over the data **before** outliers' removal. On the right-hand side, a box plot of all `ba` columns **after** outliers' removal. We identified 16 outliers in `ba-may` and 18 outliers in `ba-apr`.

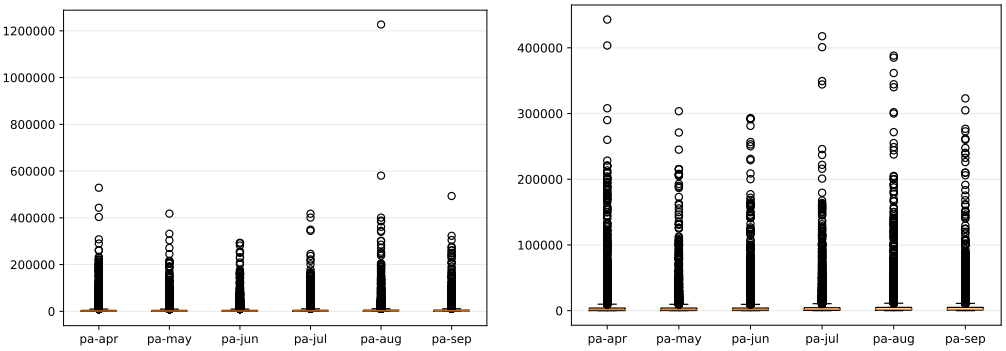

FIGURE 4: On the left-hand side, a boxplot of all `pa` columns over the data **before** outliers' removal. On the right-hand side, a box plot of all `pa` columns **after** outliers' removal. We identified 2 outliers in `pa-aug`, 1 outlier in `pa-may` and in `pa-apr`.

1.3 Distribution of the variables and statistics

Before delving into the details of the attributes' distribution, we analyze the mean values (μ) and the standard deviation (σ) of the numerical attributes of the dataset, see Table 2. Namely, we notice that

	count	mean	std	min	25%	50%	75%	max
limit	9970	167197.000000	128975.488596	10000.000000	50000.000000	140000.000000	240000.000000	780000.000000
age	9970	35.276400	8.911445	21.000000	28.000000	34.000000	41.000000	75.000000
ba-sep	9970	51490.704100	73740.383345	-14386.000000	3545.250000	22246.000000	67681.000000	613860.000000
ba-aug	9970	49239.438400	70777.470286	-69777.000000	2899.250000	21202.000000	64412.750000	605943.000000
ba-jul	9970	46957.46680	68948.62697	-61506.00000	2442.50000	19905.50000	60164.75000	597415.00000
ba-jun	9970	43306.114300	64519.910263	-24303.000000	2203.250000	19072.000000	54093.750000	616836.000000
ba-may	9970	40182.130200	60732.330157	-81334.000000	1650.000000	18071.000000	49906.500000	587067.000000
ba-apr	9970	38621.582700	59325.339137	-209051.000000	1150.000000	16977.000000	48680.750000	568638.000000
pa-sep	9970	5651.344900	15835.839092	0.000000	997.000000	2081.500000	5019.000000	493358.000000
pa-aug	9970	5973.676	22511.75	0.00000	780.0000	2000.000	5000.000	12270822
pa-jul	9970	5131.898600	15416.402957	0.000000	390.000000	1800.000000	4500.000000	417588.000000
pa-jun	9970	4719.769200	14483.407778	0.000000	261.500000	1500.000000	4000.000000	292962.000000
pa-may	9970	4734.702900	14912.375475	0.000000	200.000000	1500.000000	4000.000000	417990.000000
pa-apr	9970	5480.147400	19361.411204	0.000000	100.000000	1500.000000	4000.000000	528666.000000

TABLE 2: Basic statistics of the numerical attributes of the customer credit card dataset

the $\sigma(\texttt{ba}_m)$ is approximately three times as much as the $\sigma(\texttt{pa}_m)$, signifying that the $\texttt{ba}_m$ has a wider tail distribution.

We notice that defaulting customers are mostly concentrated in the $20-40$ age range, peaking in the 30 years of age, as we can see from Figure 5 however, this attribute does not appear to be important for distuinguishing defaulting from non-defaulting customers. Likewise for the categorical attributes in the dataset (`education`, `status`, `sex`) we notice that these attributes do not suggest a relevant substantial differing behaviour between defaulting and non-defaulting customers, as the sample flot of Figure 5 shows. The only exception is represented by the "others" value in the Education plot, which could suggest that "others" customers are more likely to default. However, the "others" group is underrepresented in the dataset (accounting for less than 1% of all rows) and can be neglected.

We also observe that customers with a low limit (in the range $[100000, 200000]$), tend to have more defaults compared to customers with a higher limit, as by Figure 6. Customers with a very high limit also tend to have many defaults, however the amount of customers with a very high limit is negligible.

From the left-hand side of Figure 6, we can see that the higher the ps, the higher is the probability of a default to occur. This behaviour is typical of all the $\texttt{ps}_m$ attributes in the considered dataset. Also, a $\texttt{ps}_m$ value close to 0, or < 0 is a strong indication that a customer is not going to default. On the other hand, a positive value of the $\texttt{ps}_m$ suggests that the customer is more likely to default. Moreover, observing the Figure 6 we noticed that the peak of the *ps* frequency distribution of 'yes' and 'no' is close to zero.

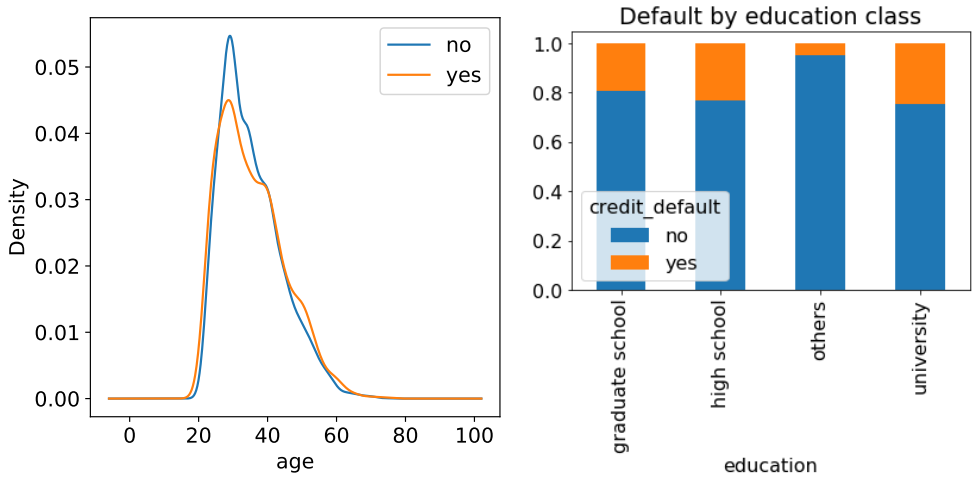

FIGURE 5: On the left, a crosstab of the `credit_default` over the `education` attribute. On the right, a plot of the distribution of the `age` attribute with respect to the `credit_default`.

2 Clustering

In this section, we describe the Clustering algorithms used to group customers into clusters to characterize customers based on their characteristics and especially based on their propensity to default.

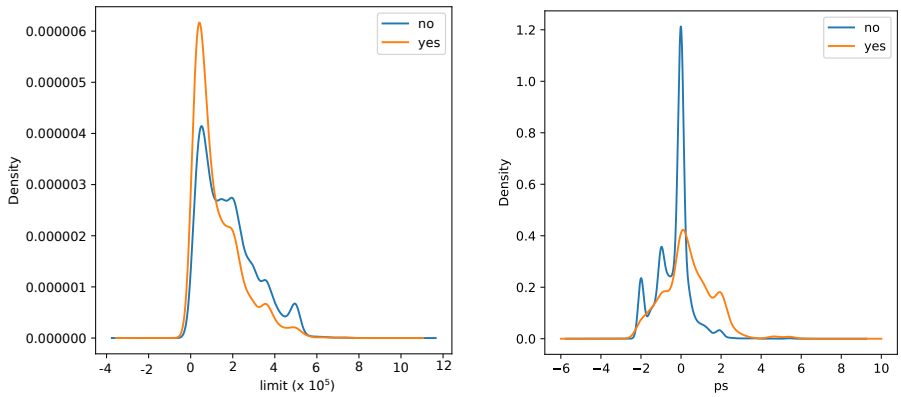

FIGURE 6: On the left, the distribution of the `limit` attribute with respect to the `credit_default`. On the right, the distribution of `ps` attribute with respect to the `credit_default`.

2.1 Pairwise correlation and data transformation

From table Figure 7, we observe that the $\texttt{pa}_m$ attributes are not strongly correlated with one another, whereas the $\texttt{ba}_m$ attributes are very strongly correlated and represent **redundant variables**. Consequently, we put together the $\texttt{ba}_m$ attributes into one single attribute, called `ba`, which is the mean of all $\texttt{ba}_m$ attributes.

Also, we put together the $\texttt{pa}_m$ into one single attribute called `pa`, which is again the mean of all $\texttt{pa}_m$ attributes. Despite the $\texttt{pa}_m$ not being redundant variables, we still operated this transformation so as to reduce the dataset dimensionality.

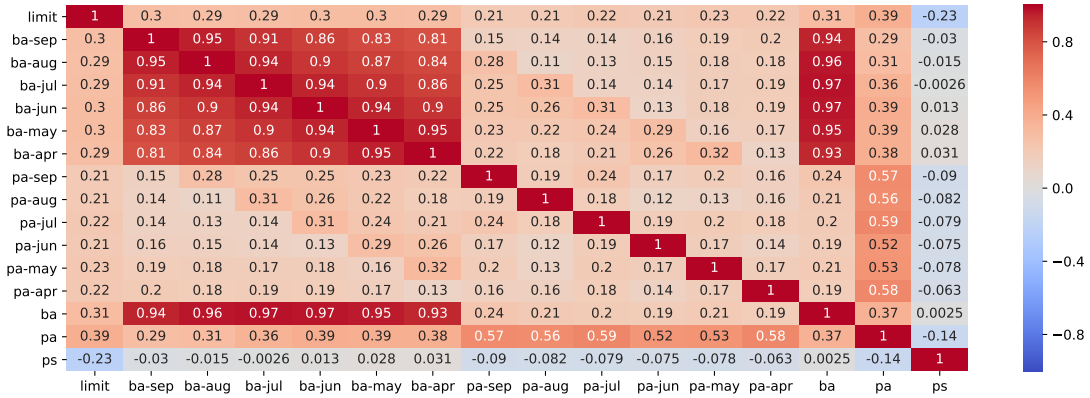

FIGURE 7: Correlation matrix computed on all numerical attributes in the dataset under study

We acknowledge the fact that the $\texttt{ps}_m$ attributes are categorical for all months considered in the dataset. Also, the $\texttt{ps}_m$ features both positive and negative integer values; consequently, we transform the '-2' and '-1' values into 0 to make the $\texttt{ps}_m$ assume positive values only in the range $[0, 8]$. We operate this transformation as the '-2' and '-1' values represent a situation where the customer has not had any credit consumption ('-2' value), or the payment of a month's consumption has been made in that same month ('-1' value). We compromise losing these pieces of further information with the possibility of using the `ps` for clustering.

This transformation of the $\texttt{ba}_m$, $\texttt{pa}_m$, $\texttt{pa}_m$ attributes produced good results in the **clustering** and **association rules mining** techniques.

As far as **classification algorithms** are concerned, we convert the ordinal attribute `education`, and the categorical attributes `credit_default`, `sex` and `status` into numerical attributes.

2.2 Distance functions

In the DB-Scan and in the hierarchical clustering algorithms, we made use of the Euclidean and Manhattan distance as the attributes considered for these two approaches are numerical, as described respectively in Section 2.6.1 and Section 2.5.1.

For the K-Means algorithm, instead, we only used the Euclidean distance because the attributes selected for K-Means are numerical and the K-Means algorithm of the scikit-learn library * did not allow to change the distance function.

2.3 Procedure for Clustering Algorithms

In the K-Means, hierarchical clustering and DB-Scan algorithms, we followed the following procedure to build clusterings:

1. Select the respective distance function(s) laid out in Section 2.2.
2. Select some sets of attributes onto which to run the clustering algorithms.
3. Identify adequate parameters (e.g: k in case of the k-means algorithm, minPoints and ε in case of DB-Scan, numClusters in hierarchical clustering).
4. Identify the best clustering out of the different clusterings obtained.
5. Characterize and describe the clusters contained in the best clustering.

2.4 K-Means

In this section we outline the procedure followed to generate clusterings with the K-means algorithm.

2.4.1 Attributes' selection

Considering that we are handling banking data, we study a customer based on the trust granted to him by the bank (represented by the `limit` attribute), the average of his monthly billing amounts (`ba` attribute), the average amount of his monthly payments(`pa` attribute) and his tendency to postpone a payment or to pay duly on time (`ps`).

Based on these observations and the data transformation carried out in Section 2.1, we hence consider the following groups of attributes, as they could well reflect the behaviour of a customer and at the same time could help reduce the dimensionality of the dataset:

Attribute Set 1: {`limit`, `ba`, `pa`, `ps`}

Attribute Set 2: {`limit`, `pa`, `ps`}

Attribute Set 3: {`ba`, `pa`, `ps`}

2.4.2 Identification of the best value of k

In order to pick the best parameter k for K-Means, we made use of the Knee method by computing the SSE for k $\in [2, 20]$. It is worth noting that we obtained extremely similar SSE decrease plots with Attribute Set 1,2,3. Figure Figure 8 shows the SSE decrease plot just for Attribute Set 3.

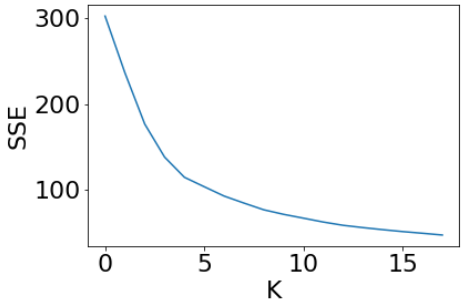

Figure 8: SSE plot for k $\in [2, 20]$ with Attribute Set 3.

From Figure 8, we noticed that a good value for k is 6 for Attribute Set 1. Looking at the SSE decrease plots of Attribute Set 2 and 3, we noticed that the best of k for Attribute Set 2 and 3 was also 6.

	Best k	SSE	Silhouette
Attribute Set 1	6	114	0.43
Attribute Set 2	6	58	0.47
Attribute Set 3	6	46	0.52

TABLE 3: Summary of the SSE, Silhouette and k values obtained for Attribute Set 1,2,3 with k-means

2.4.3 Identification of the best Clustering

From Table 3, we notice that the clustering with Attribute Set 3 has the best silhouette and the least SSE. Figure 9 shows a crosstab of the `credit_default` attribute value following the clustering with Attribute Set 3.

By comparing the clusterings obtained from the different Attribute Sets, we noticed that Attribute Set 3 is particularly interesting to characterize the behaviour of defaulting customers.

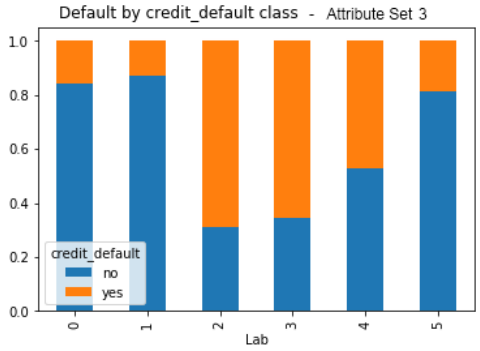

FIGURE 9: A crosstab of the `credit_default` attribute over the clustering obtained with Attribute Set 3

Cluster Index	Name	#Customers in Cluster	D	N	D/N	ba	pa	ps
0	Al	6253	983	5270	0,15	16965	3992	0,05
1	John	1618	211	1407	0,13	110307	10369	0,08
2	Jack	583	403	180	2,24	50251	2358	1,9
3	Jane	49	32	17	1,89	40557	631	4,6
4	Steve	1074	505	569	0,47	26088	2047	0,8
5	Mike	393	74	319	0,18	266851	16091	0,1

TABLE 4: Tabular representation of the centroids obtained for the clustering with $k = 6$ with Attribute Set 3. D = #Defaulting Customers. N = #Non-Defaulting Customers

2.4.4 Characterization and distribution of the best clustering

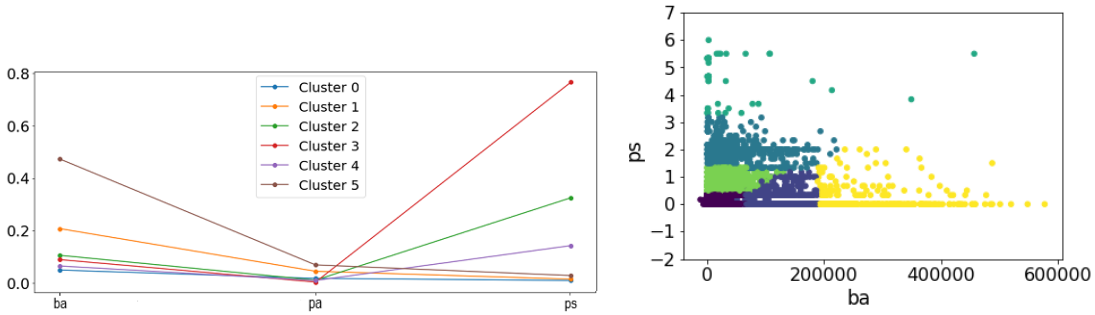

FIGURE 10: On the left-hand side, a visual representation of the centroids obtained by the parallel coordinates method for the clustering obtained with $k = 6$ with Attribute Set 3. On the right-hand side, a scatter plot of the `ba` attribute over the `ps` attribute for each cluster with $k = 6$ with Attribute Set 3.

From the left-hand side of Figure 10, we notice that the clusters' centroids are rather spread out one from another one at the `ba` and `ps` attributes, whereas they are rather close one to another at the `pa`

*https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

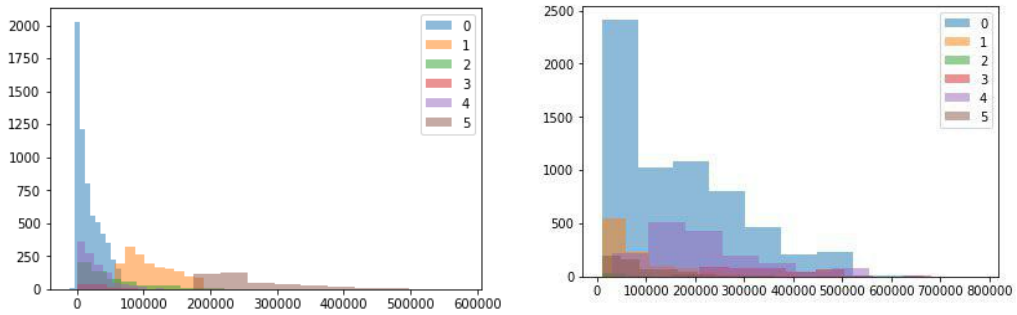

FIGURE 11: On the left, the distribution of the `ba`. On the right, the distribution of the `limit` for each cluster with $k = 6$ with Attribute Set 3

attribute. This suggests that the resulting clusters will be well separated wrt the `ba` and `ps` attributes, but not so well separated wrt the `pa`. This fact is further confirmed by the clusters' Scatter Plot shown in Figure 10, which shows customers being well-clustered wrt. the `ps` and `ba` attributes.

From Table 4, we obtained the following clusters (we recall that the limit is a measure of the the "trustworthiness" of a customer):

CLUSTER 0: **Al "The Conservative"**: Al is the most represented customer in the dataset with 6253 records (about 2/3 of the customers). He is characterized by a very low `ps`, and a very low D/N ratio, signifying that few customers belonging to this cluster will default and most customers will repay their debts on time. Customers in this clusters are also characterized by an average trust, a low `ba` and an average `pa` wrt. the overall distribution.

CLUSTER 1: **John, "The Regular Spender"**: John has similar characteristics to Al as far as D/N ratio, `ps` and `education` are concerned. However, the major difference between Al and John lies in the `ba` and `pa` amount, as John tends to spend 6 times as much as Al does, as his `ba` shows. Nevertheless, he still manages to pay his bills on time, as his `ps` of 0.08 shows. Also, his `pa` is about 2.6 times as much as the one of Al.

CLUSTER 2: **Jack, "The Typical Debtor"**: Jack is specified by a low trust from the bank, an average-low `ba` and an average-low `pa`, which is not sufficient to make him pay the billing amounts accumulated over the months. In fact, this cluster is also characterized by a high `ps` of about 1.9, meaning a significant delay in the payments. Also, most of the customers in this cluster tend to default, as the D/N Ratio of 2.24 shows. This customer type, with 583 occurrences, well-represents the typical defaulting customer, which generally does not default as far in time as Jane of Cluster 3.

CLUSTER 3: **Jane, "The Broke"**: Jane is characterized by a low trust from the bank, an average-low `ba`, but a very high D/N ratio of 1.89. The most important feature is her `ps` of 4.6 though. This `ps` is extremely high and denotes a clear tendency to delay payments up to the point in which she cannot repay them anymore and defaults. Her `pa` is also extremely low, and confirms the fact that she is not capable of paying the accumulated debts.

CLUSTER 4: **Steve, "The Parsimonious"**: Steve is characterized by the lowest `ba`, and a rather low `pa` wrt. the overall distribution and a low trust from the bank. Despite his attitude at spending little, about half of the customers in this cluster default; the `ps` of 0.8 shows that Steve tends to slightly delay his payments before paying them back.

CLUSTER 5: **Mike, "The Well-Off Guy"**: Mike is characterized by the highest trust among all clusters considered. Both his `ba` and `pa` are also highest wrt. the overall distribution of the `ba` and `pa`. His D/N ratio as well as his `ps` are very low, confirming the fact that he tends to pay duly on time, confirming that the bank's trust in his repayment capabilities is well deserved.

From the clusters presently described, we see that a high `ps` is a key factor for understanding when a customer is going to default. A low `limit` (trust from the bank), jointly with a very low `ba` and an average-low `ba` are also key indicators for understanding customers that are going to default.

Furthermore, we analysed the frequency distribution of categorical attributes (i.e. `education`, `status` and `sex`) over the obtained clusters, but no significant characterization of the resulting clusters could be found when compared with the attributes' respective global frequency distributions.

2.5 DB-scan

In this section, we explain the approach adopted to generate clusterings with the DBscan algorithm.

2.5.1 Attributes' selection

We identified the following sets of attributes onto which we run the DB-Scan algorithm based on the Correlation matrix of Figure 7:

Attribute Set 1: {`limit`, `ba`, `pa`, `ps`}.

Attribute Set 2: {`limit`, `ba-sep`, `ba-aug`, `ba-jul`, `ba-jun`, `ba-may`, `ba-apr`, `pa-sep`, `pa-aug`, `pa-jul`, `pa-jun`, `pa-may`, `pa-apr`};

Attribute Set 3: {`limit`, `ba-sep`, `ba-aug`, `ba-jul`, `ba-jun`, `ba-may`, `ba-apr`}.

2.5.2 Study of the clustering parameters

Based on the distance functions selected in Section 2.2, we selected 13 couples (ε, MinPts), $\forall$ MinPts $\in \{2^0, \ldots, 2^{12}\}$, where MinPts stands for MinPoints. The first eight couples are shown in Table 5. The k attribute was selected by applying the knee method and plotting the distance from the k-th nearest neighbour for $k \in 2^0, 2^{12}]$.

MinPts	ε		
	Euclidean	Cityblock	
1	0.23	0.63	0.03
2	0.25	0.65	0.04
4	0.28	0.7	0.05
8	0.3	0.8	0.06
16	0.32	0.85	0.08
32	0.35	0.9	0.09
64	0.38	1	0.1
128	0.42	1.1	0.13
256	0.46	1.2	0.15

Table 5: Min Points and corresponding ε, selected based on the Knee rule for every different distance type

2.5.3 Characterization and interpretation of the obtained clusters

After running the DB-Scan algorithm on all combinations of parameters, the resulting clusterings (only the ones having more than one cluster) have been sorted according to their silhouette value.

The top 10 clusterings obtained with the parameters described in Section 2.5.2 are shown in Table 6.

Based on the results of Table 6, we report a sample of the the scatter plots obtained from the clusterings with best Silhouette coefficient over a subset of attributes, namely: **Rank 1** clustering (i.e. clustering with the best silhouette coefficient) and **Rank 5** clustering (i.e. clustering with fifth best silhouette coefficient).

From these Figures, we observe that such clusters do not highlight any significant data shape nor any significant clusters can be found; instead, these clusterings just represent a cut of the clustering obtained via K-means, as shown in the clustering obtained with Attribute Set 3 of K-Means in Figure 10.

Therefore, based on the unsatisfactory clustering results obtained, we can conclude that the DB-scan Algorithm is not a good clustering technique for our dataset. No matter which ε and minPoints parameters and distance function is taken in combination with all the attribute sets considered. In all analyzed cases, we do not obtain relevant clusterings. This is imputable to the inherent data distribution of the dataset under study not being compatible with the functioning of the DB-Scan algorithm, which separates data into clusters when data is well-separated in clusters. In fact, our data is just characterized by one large cluster containing points very close to one another in it, and some outliers lying around this big cluster, which represent insignificant clusters. For this reason, no significant clustering can be identified.

Rank	Distance	Attribute Set	Eps	NumTrueClusters	NumNoisePts	Silhouette
1	cityblock	1	0.63	3	0	0.767
2	cityblock	1	0.65	3	3	0.747
3	cityblock	1	0.70	2	3	0.744
4	euclidean	1	0.23	4	0	0.672
5	cityblock	2	0.70	2	113	0.574
6	cityblock	3	0.65	4	22	0.543
7	euclidean	1	0.25	5	8	0.507
8	euclidean	1	0.28	5	5	0.507
9	euclidean	1	0.30	4	12	0.507
10	euclidean	1	0.32	4	13	0.505

TABLE 6: Summary of the results obtained with the DB-Scan algorithm with the Attribute Sets selected in Section 2.5.1 and the parameters described in Section 2.5.2

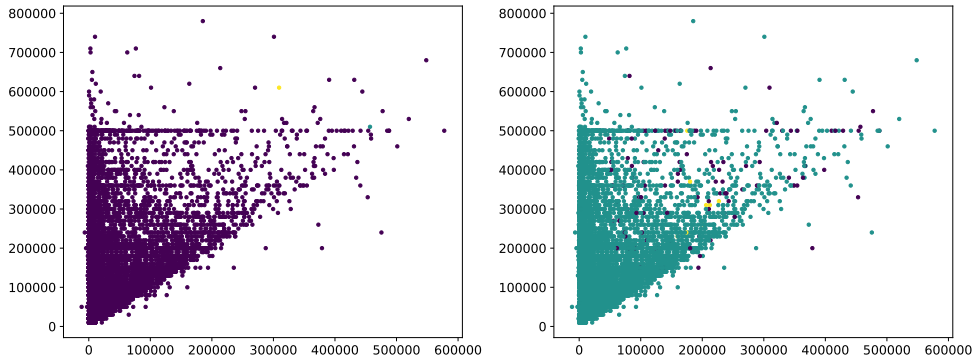

FIGURE 12: Scatter plots of `limit` (x axis) over `ba` (y axis). On the left-hand side, a scatter plot of the **rank 1** clustering according to the rank presented in Table 6; on the right-hand side, a scatter plot of the **rank 5** clustering according to Table 6

2.6 Hierarchical clustering

After selecting distance functions as by Section 2.2 and attributes as by Section 2.6.1, we establish the best number of clusters obtainable by cutting the dendogram at different heights, as described in Section 2.6.2. Afterwards, we evaluate the clusters obtained with the 'ward', 'complete', 'average' and 'single' linkage methods.

2.6.1 Attributes' selection

We selected four sets of attributes based on the correlation matrix shown in Figure 7 and on the attributes selected in Section 2.4.1:

ATTRIBUTE SET 1: {`limit`, `ba-apr`, `ba-may`, `ba-jun`, `ba-jul`, `ba-aug`, `ba-sep`, `pa-apr`, `pa-may`, `pa-jun`, `pa-jul`, `pa-aug`, `pa-sep`}

ATTRIBUTE SET 2: {`limit`, `ba`, `pa`, `ps`};

ATTRIBUTE SET 3: {`limit`, `ba`, `pa`};

ATTRIBUTE SET 4: {`ba`, `pa`, `ps`};

2.6.2 Identification of the best number of clusters

To establish the best number of clusters, we evaluate the *ward*, *complete* and *average* linkage methods. For each linkage method, we generated dendograms with numClusters $\in [2, 30]$, where numClusters is the amount of clusters considered. For each cluster model obtained with the Euclidean and Cityblock distance, we evaluate the silhouette coefficient, and report the results in Figure 13. From Figure 13, we observe that all linkage methods are characterized by a similar behaviour: as expected, the silhouette coefficient decreases as the number of clusters increases. Therefore, the best score is obtained for numClusters $= 2$. However, for the purpose of the present hierarchical clustering analysis, a clustering just with two clusters is not very significant. For this reason, we restrict numclusters to $\{3, 4, 5, 6\}$ from now onwards.

To have a visualization of the results, for each linkage method and for each cut of the dendogram we produce the scatter matrix on all attributes in the sets. We performed a visual inspection of the scatter plots, scatter matrix and the attributes' distribution for $3, 4, 5$ and 6 clusters; finally we decided that the best number of clusters were 6 for all four tests.

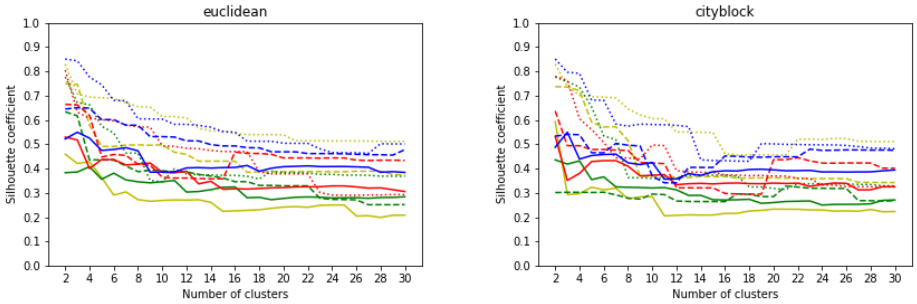

FIGURE 13: Silhouette score evaluated for numClusters $\in [2, 30]$. The left-hand-side plot shows the Silhouette coefficient obtained with the 'euclidean' metric; the right-hand-side plot shows the results obtained with the 'city block' distance. Each color is associated to one test: yellow = *test 1*, green = *test 2*, red = *test 3* and blue = *test 4*. Lines shapes' are mappings to linkage methods: 'ward' = contiguous line, 'complete' = dashed line and 'average' = dotted line.

2.6.3 Linkage methods and dendograms

At this point, we have established that numClusters = 6 and Euclidean distance form the best clustering model. In order to establish the best linkade method, we applied a *brute-force approach* to hierarchical clustering by making use of all four methods: 'single', 'complete', 'ward' and 'average'. Because the results obtained with the single method were very poor, we did not include this method in the results.

Table 7 reports the silhouette coefficient for the clustering and the number of customers per cluster, following the execution of each linkage method over the attribute sets described in Section 2.6.1 for numClusters = 6; afterwards, we have cut the resulting dendograms to form 6 clusters. From Section 2.6.1, we notice that the 'complete' and the 'average' methods produce extremely unbalanced clusters, with some clusters containing very few customers; on the other hand, the 'ward' method succeeds at creating more balanced clusters. For this reason, we selected the 'ward' method as best linkage method to produce clusterings.

		Attribute Set 1		**Attribute Set 2**		**Attribute Set 3**		**Attribute Set 4**	
	index	silhouette	elements	silhouette	elements	silhouette	elements	silhouette	elements
ward	1	0.2907	165	0.3807	548	0.4372	3559	0.4784	1415
	2		1082		2410		2390		6068
	3		915		3946		1662		32
	4		1510		1584		562		609
	5		2520		265		336		201
	6		3778		1217		1461		1645

TABLE 7: Results of the hierarchical clustering on the four sets of attributes, by cutting the dendogram to form 6 clusters with the 'average', 'complete' and 'ward' linkage method.

2.6.4 Identification and characterization of the best clustering

We identified attribute set 4 as the best set of attributes for hierarchical clustering as it captures customers with high D/N ratio in two clusters and is well fit for the purpose of describing defaulting customers, as shown in Figure 15.

By looking at Table 8 and at the clustering obtained Figure 16, we notice that clusters with index $2, 5$ and 6 have `limit` and `ba` directly proportional to their `pa` and are "safe" customers, with a very low D/N ratio. Instead, clusters with index 2 and 3 feature a `pa` no longer directly proportional to their `ba` and `limit`. Cluster 3 appears to be particularly interesting, as it captures the same situation captured by Cluster 3 with K-means, namely "broke" customers with an extremely low `pa` (almost 0) and an extremely high `ps` of 5.08 and a very high D/N ratio of 3.00. Cluster with index 4 also represents an interesting default situation, as it has a `ps` of 1.92 and an average-low `ba`.

Also, from Figure 16, we observe that Attribute Set 4 leads to a good clustering among the `ba` and `ps` attributes' combination, but a not-so-good clustering among the `ba` and `limit`.

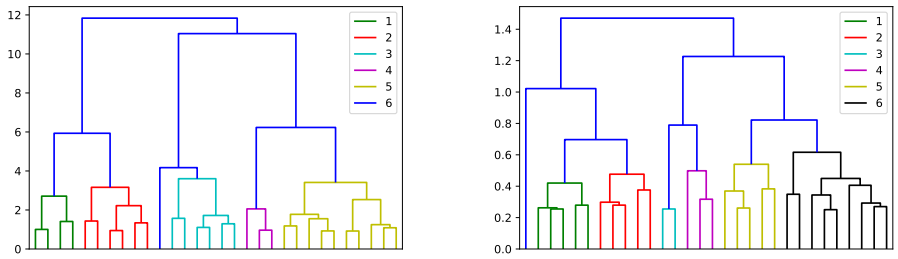

FIGURE 14: On the left-hand-side, the dendogram obtained with the 'ward' method. On the right-hand-side, the dendogram obtained with the 'complete' linkage method over the set Attribute Set$_4$ of attributes defined in Section 2.6.1. For both dendograms, the number of elements in each cluster is reported in Table 7.

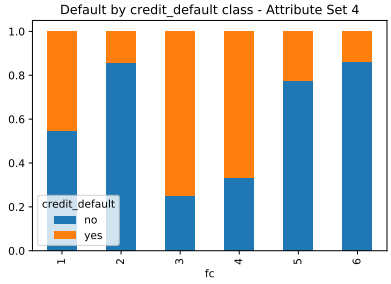

FIGURE 15: A crosstab of the distribution of the `credit_default` attribute with Attribute Set 4 and the 'ward' linkage method.

Cluster Index	# Elements in cluster	D	N	D/N	limit	ba	pa	ps
1	1415	640	775	0.83	82961	23977.59	1794.94	0.77
2	6068	864	5204	0.17	170308	17427.68	4377.19	0.04
3	32	24	8	3.00	57500	14161.07	5.61	5.08
4	609	408	201	2.03	110131	70720.94	3005.19	1.92
5	201	45	156	0.29	396965	316147.74	15605.88	0.09
6	1645	227	1418	0.16	222030	122562.94	10685.10	0.09

TABLE 8: Results of hierarchical clustering on the Attribute Set 4 with the 'ward' linkage method in tabular form

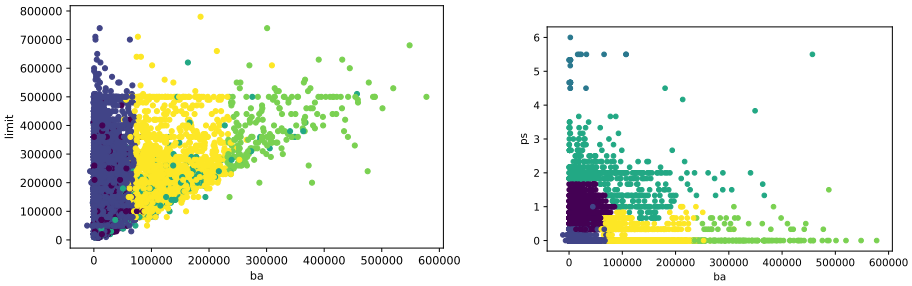

FIGURE 16: On the left the scatter plot on `ba` and `limit`. On the right, the scatter plot of the `ba` and the ps for the 6 clusters over Attribute Set 4.

2.7 Final choice of the best clustering approach

From the Observations drawn in Section 2.5.3, we evaluate the DB-Scan algortihm as the worst clustering approach among all the considered approaches, as no significant clusters could be identified despite all trials performed. The reason for its poor performance was identified to be the inherent data distribution in the dataset.

Hierarchical clustering and K-Means both produce similar and good clusterings on the same Attribute Set = {`ba`, `pa`, `ps`}. The results produced by Hierarchical clustering in Section 2.6.4 do seem to be more

Clustering Technique	Clustering Quality	Silhouette	Amount of clusters	Attribute Set	Distance
K-Means	Good	0.52	6	ba, pa, ps	Euclidean
Hierarchical Clustering	Best	0.4785	6	ba, pa, ps	Euclidean
DB-Scan	Poor	0.767	3	limit, ba, pa, ps	Cityblock

TABLE 9: Comparison of the best clusterings obtained with the K-means, Hierarchical and DB-Scan Clustering algorithms

interesting with respect to the ones obtained with K-means in Section 2.4.4 though. Despite having a lower Silhouette value as shown in Table 9, Hierarchical clustering manages to capture clusters both for the pairs of attributes: `limit`, `ba` and the `ps`, `ba`. Also, the clustering obtained with Hierarchical Clustering manages to capture the characteristics of defaulting customers very well; namely, it shows one small cluster of customers delaying their payments by an extreme amount of time (5.08) and that have close to 0 payment amounts (cluster 3), a small-sized cluster with customers delaying their payment by a discrete amount (1.92) with little spending (cluster 4) and a cluster with slightly defaulting customers and slightly delaying customers (cluster 1). All other clusters identified represent reliable customers that do pay their debts on time.

3 Classification

In this section, we describe the classification algorithms used to predict the target variable 'credit_default' and compare the performance of the algorithms employed.

3.1 Dataset Splitting and Procedure

We split the dataset by 80–20, namely 80% of the dataset is used for training the classification models and the remaining 20% of the dataset is used for testing. We did test our models both with the training and test data, to understand whether some classification models were overiffitng.

We adopt an approach aimed at chunking the training and test parts of the dataset into smaller chunks and train our model on a set of these chunks. The purpose of this further chunking was to increase the granularity of the training and test sets: the model having best performance on the Kaggle test dataset would be trained on the fold most closely resembling it (and hence best fitting it). This innovative approach was named n**-Fold Subdivision**, where n is the amount of chunks into which the training and test set are respectively subdivided. If n=10 we split the training set and the test set into 10 different folds for the training and test parts; each fold has 90% of the training set and 90% of the test set.

In the comparison of the different classification models considered, we will always make use of the F-Score computed on the test data, as the goal of our classification is to maximize the final F-Score computed on the Kaggle test dataset.

3.2 Decision tree

In order to learn the best possible decision tree, we firstly create a Simple Decision Tree (SDT) in Section 3.2.1 and Section 3.2.2, then proceed to optimize it by generating an Optimized Decision Tree (ODT) in Section 3.2.3.

3.2.1 Simple Decision Tree by 10-fold cross-validation

We create a decision tree based on the following parameters: max_depth = 2, min_samples_split = 2, min_samples_leaf = 1 and criterion=gini. We pick these parameters as we want create a tiny decision tree that will act as a baseline performance onto which we will improve via parameter optimization.

Then, we apply 10-**Fold Cross-validation Stratified** for each fold obtained applying the 10-**Fold Subdivision** technique. Table 13 reports the best model obtained.

3.2.2 Simple Decision Tree by downsampling

We notice that the dataset is strongly unbalanced towards non-defaulting customers (with 7762 entries) vs. defaulting customers (with 2208 entries). We tried to perform downsampling of the dataset to the minority class represented and created SDTs both on the data prior to downsampling and following downsampling. The results obtained with downsampling both on the training and the test dataset turned out to be very poor (F-Score of 0.68) and are surpassed by SDT generation by cross-validation in Section 3.2.1. For this reason, we do not consider downsampling with any other classification algorithm.

3.2.3 Optimized Decision Tree

In order to improve the performance of the Optimized Decision Tree (ODT), we apply the 10-Fold Subdivision technique.

For every fold, we perform hyper-parameter tuning for the SDT tree described at the beginning of Section 3.2.1 with the Grid Search and Randomized Search algorithms based on the parameters of Table 10. We made use of the 'Gini' splitting criterion, which turned out to be yielding the best performance when compared with the 'entropy' splitting criterion.

	Parameters' Configuration
Grid Search	'max_depth': [2:5] 'min_samples_split': [2, 5, 10, 20, 30, 40, 50, 100] 'min_samples_leaf': [1, 5, 10, 20, 30, 40, 50, 100]
Randomized Search	'max_depth': [2:100] 'min_samples_split': [2, 5, 10, 20, 30, 50, 100, 150, 200] 'min_samples_leaf': [1, 5, 10, 20, 30, 50, 100, 150, 200]

TABLE 10: Parameters' Configurations used to tune the Grid Search and Randomized Search algorithms

Table 11 shows the metrics obtained by applying Grid Search optimization via 10-fold sub-division, and the parameters identified from for each fold's best model. We compared the performance of the grid search and the randomized search optimization algorithms, and noticed that randomized search generated ODTs with a very high max_depth (max_depth >= 30), which are likely to overfit the data.

Index	**max depth**	**min samples leaf**	**min samples split**	**Data Type**	**Accuracy**	**Roc-Auc**	**Precision**	**Recall**	**F-Score**
1	2	100	2	Test	0.81	0.7	0.8	0.81	0.79
				Train	0.82	0.7	0.8	0.82	0.8
2	2	1	2	Test	0.81	0.69	0.8	0.81	0.79
				Train	0.82	0.7	0.81	0.82	0.79
3	2	1	2	Test	0.81	0.69	0.8	0.81	0.79
				Train	0.82	0.7	0.81	0.82	0.79
4	4	10	2	Test	0.81	0.73	0.89	0.81	0.79
				Train	0.83	0.77	0.82	0.83	0.81
5	2	1	2	Test	0.83	0.71	0.82	0.83	0.81
				Train	0.82	0.7	0.8	0.82	0.79
6	2	100	5	Test	0.81	0.68	0.79	0.81	0.78
				Train	0.82	0.71	0.81	0.82	0.8
7	2	100	5	Test	0.82	0.71	0.81	0.82	0.8
				Train	0.82	0.7	0.8	0.82	0.79
8	2	1	2	Test	0.82	0.69	0.81	0.82	0.79
				Train	0.82	0.7	0.81	0.82	0.79
9	4 / 2	100	5 / 2	Test	0.83	0.77	0.81	0.83	0.81
				Train	0.82	0.82	0.8	0.82	0.8
10	4	5	20	Test	0.81	0.74	0.79	0.81	0.79
				Train	0.83	0.76	0.81	0.83	0.81

TABLE 11: Parameters and metrics obtained for the best ODT for each fold index obtained with the Grid Search optimization algorithm and based on the 10-fold sub-division approach

3.2.4 Optimized Decision Tree Interpretation

From Table 11, we select the ODTs having best F-Score on the test set. In case of tie (F-Score 0.81), we compare further metrics on the test set in this order: roc-auc, accuracy, precision. The decision trees with best F-Score on the test dataset have been found to be the following ones:

1. **ODT (1) with index = 9**, max_depth = 4, min_samples_leaf = 100, min_samples_split = 5 with a test F-Score of 0.81.

2. **ODT (2) with index = 9**, max_depth = 2, min_samples_leaf = 100, min_samples_split = 2 with a test F-Score of 0.81.

ODT (1) and ODT (2) have the same performance metrics on the training and test dataset under study. In order to try and understand which one would perform better, we perform two submissions on Kaggle, and we obtain the same identical classification result of 0.81716 as F-Score.

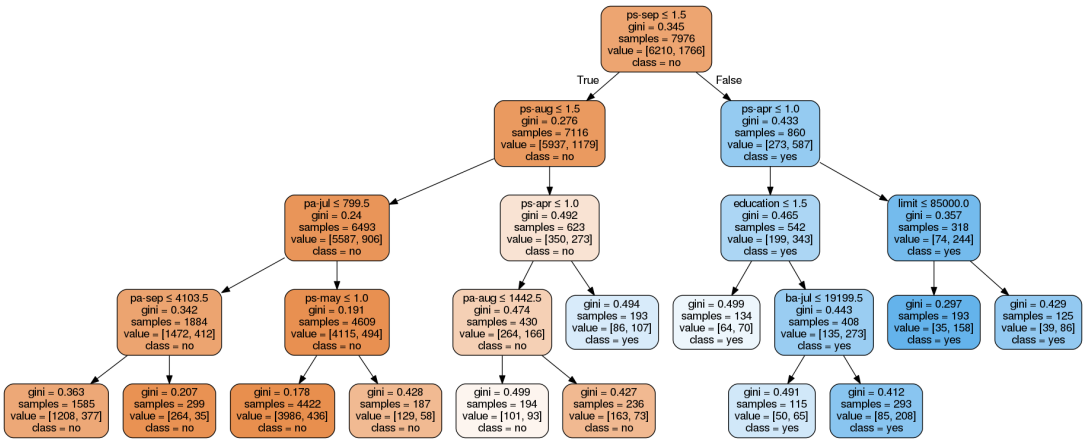

FIGURE 17: Complex ODT (1) with 4 levels.

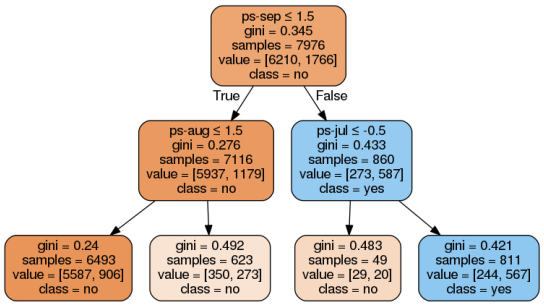

FIGURE 18: Simpler ODT (2) with 2 levels.

From Figure 17 and Figure 18 we notice that ODT (2) represents a "pruned" version of ODT (1); ODT (1) and ODT(2) have the exact same performance both on the training and test set as well as on the Kaggle test data. Therefore, we prefer ODT (2) over ODT (1) according to the Occam's Razor, telling us that a simpler model should always be preferable to a more complex one (especially if these two models are functionally equivalent)

Analysing ODT (1) and (2), we observe that the main split occurs at 'ps-sep' for both decision trees; then, on the left-hand side of 'ps-sep' we have a split on 'ps-aug' for both decision trees. ODT (2) manages to capture almost entirely the deep branching of ODT (1) prdoducing "no" as an answer for 7116 samples. The only non-captured values with a "yes" attribute are 193 samples, where 'ps-apr' branches based at a value of 1.0.

On the right-hand side of the branching at 'ps-sep' in ODT (1) we have a branching at 'ps-apr' again at the value 1.0. No matter which value 'ps-apr' assumes, the class assigned by the underlying branches is always going to be 'yes'. This fact suggests that we could have stopped earlier in our branching, and ODT (2), having a branching at 'ps-jul', hence manages to capture the right-hand side of 'ps-sep' in a more succinct way for the customers defaulting.

3.3 Random forest

Analogously to the approach followed for Decision Trees, we firstly create a Simple Random Forest Model in Section 3.3.1, then we proceed to optimize it in Section 3.3.2.

3.3.1 Simple Random Forest Model

We apply 10-fold cross-validation to create a Simple Random Forest Model. The basic Simple Random Forest model is initialized with the following parameters: n_estimators = 200, max_depth = None, min_samples_split = 2, min_samples_leaf = 1 and criterion = gini. We pick these parameters as we want create a simple random forest model that will act as a baseline comparison model onto which we improve via parameter optimization.

The simple random forest model with best performance on the test dataset out of the 10 models generated with the 10-Fold Subdivision technique is shown in Table 13. Its 1.0 metrics on the training dataset shows that this model overfits the training dataset.

3.3.2 Optimized Random Forest Model

We optimize the performance of the Simple Random Forest Model described at the beginning of Section 3.3.1 via hyperparameter tuning through the Grid Search and Randomized Search algorithms with the parameters' configuration of Table 10 and 10-fold dataset Sub-division of Section 3.2.3.

The best models (i.e: the ones with the best F-Score on the test dataset) appear to be the following ones.

1. **Model (1):**Optimized Random Forest with index = 4: It features max_depth = None, min_samples_split = 100, min_samples_leaf = 1 and has an F-Score of 0.82, obtained via Grid Search .
2. **Model (2):** Optimized Random Forest with index = 4: It features max_depth = 22, min_samples_split = 100, min_samples_leaf = 10 and has an F-Score of 0.82, obtained via Randomized Search.

In order to understand which model would perform best at classifying new unseen data, we perform two submissions on Kaggle. Model (1) appears to have an F-Score of 0.8165. Model (2) appears to have an F-Score of 0.81766. We can hence conclude that Model (2) is the best one obtained via Optimized Random Forest.

As already introduced in Section 1, the shape of the dataset does not allow a good classification of customers into defaulting and non-defaulting ones.

A numerical proof is given in Table 12, where it is easy to observe that the number of false negative (in other words, those customers who were identified as non-defaulting, although their actual behaviour was a defaulting one) and the number of false positive (the other way round) is pretty large.

		Actual values	
		Yes	**No**
Predicted values	**Yes**	887	347
	No	1321	7415

TABLE 12: Confusion matrix on the Optimized Random Forest Model (2), from the application of optimized random forest classification algorithm obtained with randomized search and index 4.

3.4 Deep Learning

We considered the deep learning model described at this page.[†]

We made use of one input layer with 23 nodes, encompassing all the features in the dataset (excluding the output variable), along with two hidden layers, respectively with 20 and 10 nodes and one output node. The activation functions used were "relu" for the input and the hidden layers, and "sigmoid" for the output layer. Resulting metrics are shown in Table 13

3.5 Classification Summary

Table 13 shows the metrics obtained for the best models of all the classification algorithms considered.

We notice that SDT, ODT, Simple Random Forest and Optimized Random Forest have a a comparable performance, with the Optimized version of the Random Forest and the Decision Tree scoring

[†] https://medium.com/@Saadism/credit-card-default-prediction-using-tensorflow-part-1-deep-neural-networks-ef22cfd4d278

Method (best F-Score)	Data Type	Accuracy	Roc-Auc	Precision	Recall	F-Score
Simple Decision Tree	Test	0.83	0.71	0.82	0.83	0.81
	Train	0.82	0.7	0.8	0.82	0.79
Optimized Decision Tree	Test	0.83	0.77	0.81	0.83	0.81
	Train	0.82	0.82	0.8	0.82	0.8
Simple Random Forest	Test	0.82	0.79	0.81	0.82	0.81
	Train	1.0	1.0	1.0	1.0	1.0
Optimized Random Forest	Test	0.83	0.81	0.82	0.83	0.82
	Train	0.83	0.86	0.82	0.83	0.81
Deep Learning–2 Layers	Test	0.7	0.537	0.68	0.71	0.7
	Train	0.711	0.531	0.68	0.71	0.69
Naive Bayes	Test	0.41	0.72	0.77	0.41	0.41
	Train	0.39	0.69	0.75	0.39	0.39
KNN with $k = 3$	Test	0.76	0.63	0.73	0.76	0.74
	Train	0.84	0.89	0.83	0.84	0.83

TABLE 13: Summary of the best classification models for the different classification techniques considered

better than their Simple counterpart (as expected). It is worth noticing that the 1.0 classification score obtained in all metrics of the Simple Random Forest model is a strong indicator of an overfitting model.

The model with best F-Score on the Test dataset appears to be the Optimized Random Forest one, with an F-Score of 0.82. On the other hand, Naive Bayes and Deep Learning appear to have a drastically lower F-Score, with Naive Bayes being the worst method at classifying both the training and test data. Deep Learning, in fact, does not produce relevant results because of the lack of a sufficiently large dataset and just features an F-Score of 0.7 on the test dataset. KNN, instead, appears to perform discretely, with an F-Score of 0.74. Though KNN's F-Score is worse than SDT's, ODT's and the Simple and Optimized version of Random Forest, it still features the highest classification score on the training dataset and highest Roc-Auc.

4 Pattern mining

In this section we address the problem of finding frequent itemsets and association rules in order to better understand the information hidden in the dataset.

4.1 Attribute selection and binning

As a first issue we dealt with the issue of finding the most informative attributes, thanks to the conclusions drawn from the Decision Trees of Section 3.2.4 and the clustering results in Section 2.7.

Since customers can not be distinguished between defaulting and non-defaulting ones wrt the `sex`, `education`, `status` and `age` attributes, we did not consider such attributes for pattern mining.

The resulting set of attributes is the following: `limit`, `ba` (average of the billing amount), `pa` (average of the payment amount) and `ps` (the payment status). For the `ps`, we considered: `ps-sep`, the last payment status of the customer; `ps_mode`, the mode of the payment statuses of a customer; `ps`, the average of a customer's payment statuses following data transformation as by Section 2.1.

In order to run the apriori algorithm we applied **equal-frequency binning** on limit, ba, pa and ps attribute.

4.2 Frequent itemsets extraction

We ran the `apriori` algorithm for frequent itemsets extraction, based on the attributes selected and the results have the shape described by means of Figure 19. We may observe that the frequency of those patterns that occur 1000 times (or, equivalently, have a support of 0.1) is almost three times the frequency of the patterns that have twice the support.

After some reasoning, we noticed that more than 50% of defaulting and non-defaulting customers fell in the first three bins of the `pa` and `ba` attributes. This fact made the defaulting and non-defaulting customers undistinguishable, therefore we discarded those itemsets containing elements from `pa` and `ba` attributes.

The resulting frequent itemsets, displayed in Table 14, Table 15, Table 16 and Table 17, are all **close** except for $\{[1.0, 1.5)$`ps`, `yes`, $[10000.0, 87000.0)$`limit`$\}$ which is only frequent. No maximal itemsets were found.

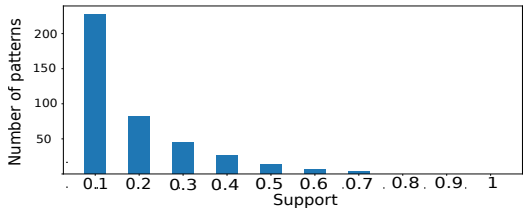

FIGURE 19: Frequency plot of itemsets with different values of support.

From the generated itemsets, we noticed an *expected behaviour*. Non-defaulting customers are characterized by a tendency to use revolving credit, not to consume or to pay in the same month as we can see from Table 14.

Furthermore, we noticed that defaulting customers are characterized by a higher tendency to delay payments, as Table 15 shows, although the support is very low (due to the low frequency of 'yes' value of the attibute `credit_default`).

Pattern	Support
no, $[0.0, 0.5)$ps	0.73
-2.0ps_mode, no	0.12
-1.0ps_mode, no	0.17

TABLE 14: Frequent itemsets for the *expected* behaviour of non-defaulting customers.

Pattern	Support
$[1.0, 1.5)$ps, yes, $[10000.0, 87000.0)$limit	0.02
$[2.0, 2.5)$ps, 2.0ps_mode, yes	0.02
$[3.0, 3.5)$ps, 2.0ps_mode, yes	0.0007

TABLE 15: Frequent itemsets for the *expected* behaviour of defaulting customers.

However, we also obtained some *unexpected* behaviour both for defaulting and non-defaulting customers, as shown in Table 16. There exist customers who do not default, despite having a behaviour typical of those who default.

The same anomaly was discovered for defaulting customers, as shown in Table 17.

Pattern	Support
2.0ps_mode, 2ps-sep, no	0.01
$[1.0, 1.5)$ps, no	0.03
$[2.0, 2.5)$ps, 2.0ps_mode, no	0.007
$[3.0, 3.5)$ps, 2.0ps_mode, no	0,0007

TABLE 16: Frequent itemsets for the *unexpected* behaviour of non-defaulting customers.

Pattern	Support
yes, $[0.0, 0.5)$ps	0.15
yes, 0.0ps_mode	0.1
-1.0ps_mode, yes	0,03
-2.0ps_mode, yes	0,02

TABLE 17: Frequent itemsets for the *unexpected* behaviour of defaulting customers.

By comparing Table 17 with Table 15, showing respectively the expected and unexpected behaviour of defaulting customers, we observe that there exist itemsets with higher support describing the unexpected behaviour of defaulting customers than those that describe their expected behaviour.

More reasonably the behaviour shown in Table 14 and Table 16 underlines a lower support for the unexpected itemsets, confirming the fact that unexpceted itemsets are indeed rare.

These two facts introduce the situation we are going to face in Section 4.3, where the predictive accuracy is hindered by the unexpected behaviour of a large quantity of defaulting customers.

4.3 Association Rules

In this section we discuss the most interesting association rules obtained through the apriori algorithm. The frequency of the obtained rules is shown in Figure 20. In this case, differently from the itemsets' case (Figure 19), the relation between the frequency of a certain rule is linear in the confidence value.

In order to add some more information about the rules that have been generated we refer the reader to Figure 20.

4.3.1 Association rules to predict the target variable

Given the frequent itemsets found in Section 4.2, it was decided to use the rules that best model the behaviour of the defaulting and non-defaulting customers.

For this reason, we used the rules obtained from the expected behaviour of defaulting customers, shown in Table 17, although these rules are characterized by a greater support wrt the rules obtained from the unexpected behaviour of defaulting customers and shown in Table 15.

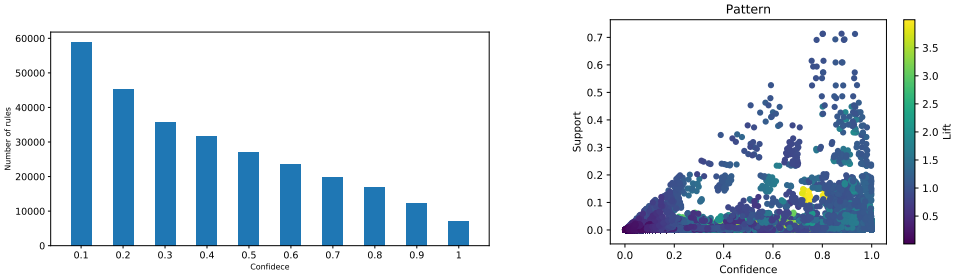

FIGURE 20: On the lefthand side, frequency plot of rules by varying the confidence value. On the righthand side a scatter plot of support, lift and confidence.

Although in frequent itemsets analysis it was highlighted that the majority of defaulting customers have the same behaviour of non-defaulting ones, the rules that outline the expected behaviour have the highest confidence and lift values.

The result of this process is described in Table 18, whereas the confusion matrix is shown in Table 19. From this table we compute the accuracy, which is 0.814.

RHS	LHS	Confidence	Lift
no	0ps_sep, 0.0ps_mode, [0.0, 0.5)ps	0.88	1.13
no	−1ps_sep	0.83	1.06
no	−1.0ps_mode	0.83	1.06
no	−2.0ps_mode	0.82	1.05
yes	7.0ps_mode, [10.0, 72206.875)	0.8	3.6
yes	3.0ps_mode	0.76	3.4
yes	3ps_sep	0.76	3.4
yes	4ps_sep	0.73	3.3
yes	2ps_sep	0.68	3.08
yes	5ps_sep	0.63	2.8
yes	2.0ps_mode	0.62	2.8
yes	[1.0, 1.5)ps	0.56	2.54

TABLE 18: Most significant association rules for predicting the target variable.

		Actual values	
		Yes	No
Pred. values	Yes	1058	701
	No	1150	7061

TABLE 19: Confusion matrix obtained from the application of the association rules for target variable prediction.

4.3.2 Association rules for missing values replacement

We recall from Section 1.2 that missing and invalid values are only present in the `age`, `sex`, `education` and `status` attributes. We generated association rules for these four attributes, without considering the `ba` and `pa` attributes, with `minimum support` = 0.05 (corresponding to 5 rows of the dataframe) and `minimum_confidence` = 0.05. The number of rules generated is 1187, but we removed 447 rules because they had a missing value either on the right-hand-side (RHS or consequent) or on the left-hand-side (LHS or antecedent). The RHS and its number of repetitions are reported in table 20 for all 740 remaining rules. For each row in the `age`, `sex`, `education` and `status` attributes where at least one missing or invalid value is present for each row, we selected the rule to correct the missing value whose itemset returns the maximal number of matches. In case two or more rules are returned, we pick the one with maximum confidence. An example of this approach is shown in Table 21 when trying to correct the `sex` attribute, where we considered the rules with RHS (Right-Hand-Side) equal to 'female' or 'male', and picked the rule with maximum confidence, hence producing 'female' as predicted value for sex.

The results of this procedure are reported in Table 22. As we can see, for the `education`, `status` and `age` attributes, the generated rules are able to predict all values except for three rows, which have an 'NaN' value in all attributes except for the `sex` attribute.

attribute	RHS	# rules	mean support	mean confidence	mean lift
age	[21.0, 34.5)	53	0.0550	0.4922	1.0416
	[34.5,48.0)	53	0.0380	0.2980	0.9174
	[48.0,61.5)	51	0.0122	0.1182	1.1696
	[61.5,75.0)	39	0.0011	0.0089	1.3347
sex	female	79	0.0446	0.6007	0.9963
	male	79	0.0290	0.4005	1.0351
education	graduate school	54	0.0380	0.3191	0.9180
	high school	55	0.0178	0.2037	1.2183
	other_education	35	0.0005	0.0068	1.8869
	university	55	0.0501	0.4662	0.9945
status	married	67	0.0366	0.3859	1.0268
	single	68	0.0417	0.4293	0.9884
	others_status	52	0.0009	0.0167	2.2250

TABLE 20: Mean metrics of the association rules for replacing missing values.

Sex	Education	Status	Age	Predicted Sex
NaN	graduate school	single	24	female

RHS	LHS	Support	Confidence	Lift
female	graduate school, single, [21.0, 34.5)	0.0533	0.6114	1.0140
male	graduate school, single, [21.0, 34.5)	0.0335	0.3839	0.9921

TABLE 21: Table above: Example of a row having a missing value in the `sex` attribute. Table below: eligible association rules used to predict the `sex` attribute. The highlighted rule is the one used to predict the `sex` attribute as it features the highest confidence.

Attribute	# Missing Values	# Predicted Values	Accuracy
Sex	100	100	0.9993
Education	127	124	0.9875
Status	1817	1814	0.9462
Age	945	942	0.9727

TABLE 22: Accuracy of the predicted values for the `sex`, `education`, `status` and `age` attributes with the association rules of Table 20.

5 Conclusions

In the analysis of this banking dataset, we used many data mining techniques with the intention of distinguishing trustworthy customers from untrustworthy ones. We presently summarize the results obtained with such techniques.

In the preliminary *Data Understanding* part (Section 1), we addressed issues related to data semantics and quality and eliminated redundant variables. In this section, we also pointed out that the main attributes that characterize defaulting customers from non-defaulting customers are the tendency of a customer to postpone his payments.

In the section about *Clustering* (Section 2), we showed how a customers' history of payments and tendency to postpone payments is crucial to cluster customers into groups of mostly-defaulting customers and mostly non-defaulting customers. None of the clustering methods we employed (i.e. K-Means, Hierarchical and DB-Scan clustering) highlighted exclusively defaulting or exclusively non-defaulting clusters.

For what concerns *Classification* (Section 3), we created several predictive models based on Decision Trees, Random Forest, Deep Learning, KNN, Naive Bayes. We obtained good classification results, although these were strongly limited by the semantic inconsistencies present in the dataset (especially hindering the classification of defaulting customers).

Last but not least we dealt with *Association Rules* (Section 4.3), where we showed the existence of interesting item sets that characterize the behaviour of defaulting and non-defaulting customers. Namely, non-defaulting customers were found to be characterized by an expected behaviour, having a very low payment status. Instead, defaulting customers were found to be characterized by an unexpected behaviour, as they often exhibit characteristics typical of non-defaulting customers.

By applying the association rules, we obtained excellent performance when predicting missing values. For the prediction of the target variable, we obtained a comparably lower accuracy wrt the one obtained in the classification phase.