Data Mining Report
Academic year 2018/2019

Roberta Currao
Andrea Lisi
Sophie Matter
Alessandro Pagiaro

January 2019

Introduction

In this project, we receive a dataset and perform several steps from the field of data mining on it in order to understand the data and gain information from it. This includes several steps:

- **Data Understating:** interpretation of the columns and their values (their semantic meaning), finding and handling unusual entries (outliers) and missing values, looking at the distributions of the variables and finding correlations between them,
- **Clustering:** trying to group similar values to find interesting structures, using K-means, Hierarchical Agglomerative Clustering and DBScan and pre-processing the data beforehand,
- **Association Rules and Pattern Mining:** trying to find frequent rules and patterns in the data to learn more about the dependencies of different values and attributes, and
- **Classification:** attempting to classify the data regarding one or more classes to be determined within the data understanding part, again to find relations within the data to be able to predict new data with a model built from this dataset.

In this order, we perform the different steps and report our results. We hope to extract meaningful knowledge from this data.

Contents

1 Data Understanding

1.1 Data Semantics

The dataset is a collection of information regarding people having a credit card with a bank. It has a total of 10000 rows, each row representing a customer, and 24 columns representing attributes regarding these customers: the attributes are divided into personal information such as `sex` and `marital status` and credit card information such as the `limit` and the history of bills, month by month, that each customer has to pay back. Such histories involve only the period between April and September 2005 (both included).
The problem we issue is "*could we predict whether or not a customer will lose his/her credit card?*": for this reason the **target variable** is a binary variable called `credit_default`, indicating whether or not a customer lost his/her credit card. Having a way to predict if a (potential) customer will lose his/her credit card would benefit for banks because releasing a credit card to a person means investing on this person; but also predict if people who are already customers will lose or not (with high probability) their cards would be good because the bank could take precautionary actions, especially for important clients such as big companies.
The name and the meaning of each attribute are summarized in table 1.

limit	The amount of available credit, the maximum amount the card user can spend
sex	The gender of the user
education	The highest education level of the user
status	The marital status of the user
age	The age of the user
ps-[sep, . . . , apr]	**Payment Status** **-2**: the credit card has not been used for any payment **-1**: the credit card has been used but the bill has been paid in full **0**: revolving credit was used: the credit card has been used, the bill has not been paid in full, so the bill revolves into the next month $\boldsymbol{x} \geq \mathbf{1}$: the credit card has been used, but the bill hasn't been paid until the x^{th} month
ba-[sep, . . . , apr]	**Billing amount** The (accumulated) bill, i.e. the money that was spent and was not payed back yet
pa-[sep, . . . , apr]	**Payed amount** The amount of money that has been paid back to the bank
credit_default	The target attribute, true if the credit card is in default, and false if it is not

Table 1: Table of attributes

All the values related to the money such as `limit`, `ba-*` and `pa-*` refer to NT dollar, dollars from Taiwan, and its current value in euro is 0.029 euro meanwhile in 2006 was 0.025 euro: the dataset refers to 2005 data.

1.2 Missing Values Analysis

The dataset presents a few columns with easy to see missing values: they are `Sex`, `Status`, `Education` and `Age`. The first three mentioned columns present missing entries meanwhile `Age` present a value of -1 that can be interpreted as a missing value. Table 2 shows how many missing values, in percentage, we have and, for the categorical columns, the occurrence of the meaningful values. We decided to treat the categorical missing values by substituting them with a meaningful value picked at random following the **probability of the occurrence** of such values with the purpose to maintain their distribution; instead, -1 entries of `Age` will be substituted with the **mean** of the whole column (obviously, excluding -1). All the other attributes (apart from the target attribute, the credit default one) are integer values and don't have an intuitive representation for missing reported value.

Payment Status: this attribute concerns all the columns of type `ps-*`. These columns are a mix of categorical and ordinal representation: a positive integer means number of months in delay and a -1 means

payment on time. There are two values, -2 and 0, that their meaning is not explained in the original dataset description: one, or both, of them could be an encoding for missing value but their occurrence is really high (especially for 0) and thus the columns would be meaningless. We contacted the author of the dataset and he answered that -2 and 0 represent in fact something meaningful (reported on table 1), and thus `ps-*` should not present any missing value.

Payment Amount: the columns `pa-*` present many 0 value but, from their semantics, we cannot be sure that they are a missing value: as far as we know 0 can represent both missing value and no payment to the bank. For this reason we DON'T treat 0 as a missing value.

	Miss Vals	Others	Substitution	Value_1 Occ	Value_2 Occ	Value_3 Occ
Sex	1%	-	Probabilistic	(Female) 39%	(Male) 61%	-
Status	18%	0.75%	Probabilistic	(Single) 43.6%	(Married) 37.3%	-
Education	1,2%	0.36%	Probabilistic	(Uni) 47.6%	(G S) 35.3%	(H S) 16.9%
Age	9.5%	-	Mean	-	-	-

Table 2: Missing values analysis

1.3 Outlier Detection

With the help of boxplots we analyzed the presence of outliers on our attributes. The attributes with "stronger" outliers, i.e. further from most of the values, are `ba-*` and `pa-*`; the other attributes have some values outside the 5-95 percentiles but not that far, so we decided to consider them as reasonable values.

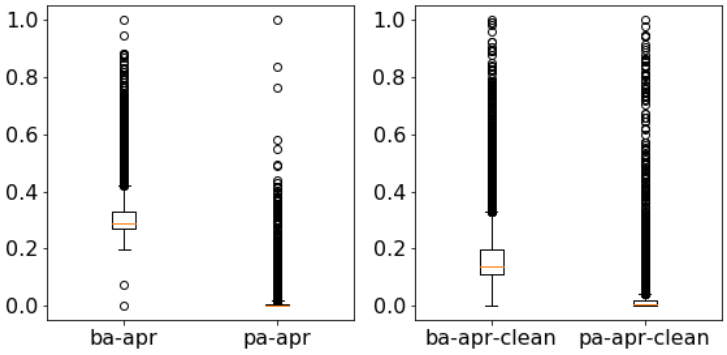

Figure 1: Outlier removal, month of April

For example, the attributes `ps-*` have all positive values above the 95 percentile just because the occurrences of 0, -1 and -2 are so many that we cannot consider the positive values as outliers since we would lose reasonable information, i.e. the delayed payments. In figure 1 we show the difference of value distribution of `ba-apr` and `pa-apr` before and after the cleaning. The choice of April is justified by the ease to see the outliers and because the other months have similar boxlplots. The values are normalized to put all the plots on the same measure scale. As we can see the values after the cleaning seem to be better distributed, even though outside the box and far from the mean. The cleaning removed in total 60 rows, i.e. 0.6% of the data and we think it's reasonable.

1.4 Distribution of the Variables and Statistics

We now show the distribution of the values and basic statistics of the resulting dataset after the cleaning steps listed so far. Figure 2 shows the distribution of `Education`, `Status` and `Sex`. In figure 3 we show the distribution of the variables `Age` and `Limit` represented as continuous values. From these we can notice that the majority of the users have smaller limits on their credit cards, highly packed below

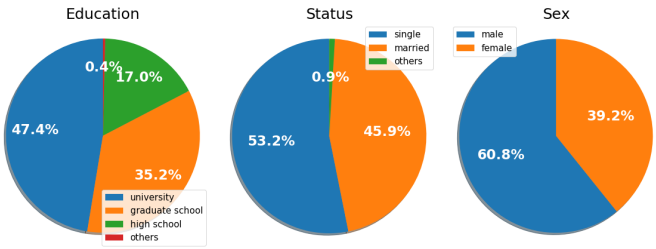

Figure 2: Distribution of personal data

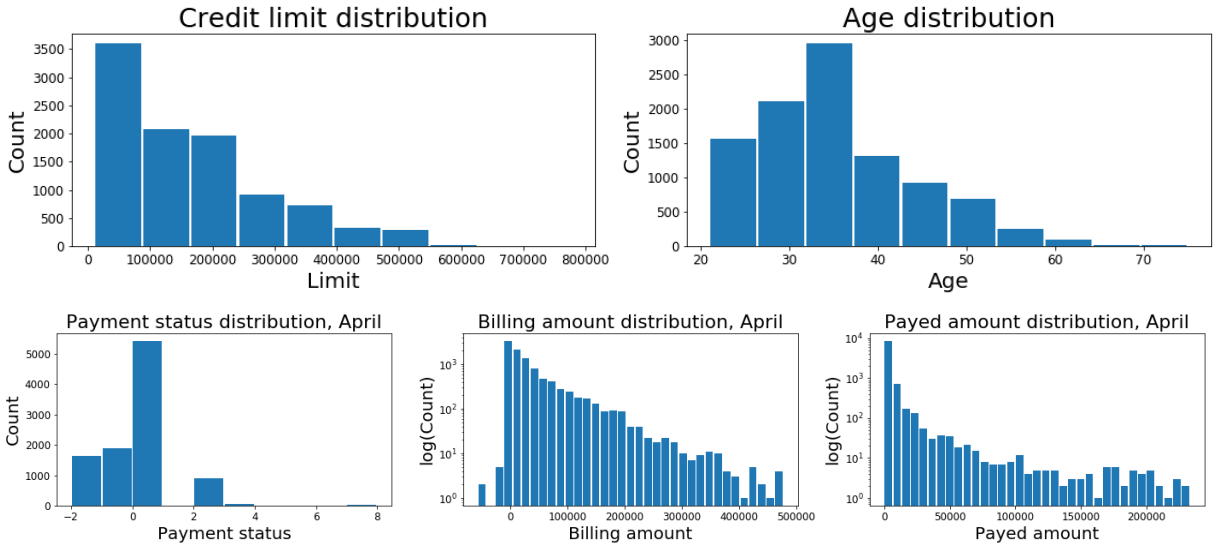

Figure 3: Distribution of discrete variables: Limit, Age, ps-apr, ba-apr and pa-apr

200K dollars, and that the majority of users are below the late 30s, probably the generation young enough to use the credit card for everyday payments such as online purchases.

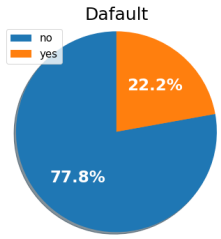

Figure 4: Default distribution

Later on, still in figure 3, there are variables related to the user's actions related to the credit card, `ps-*`, `ba-*` and `pa-*`. As for the outlier detection we show only a representative month for each value, for instance April, in order to keep the report less heavy. The leftmost histogram shows the distribution of the `ps-apr` column and this plot confirms what said in section 1.3, i.e. the great majority of values lie in range [-2, 0]. The next two histograms show the distribution of values of `ba-apr` and `pa-apr` on a logarithmic scale on the y-axis to have a bigger picture of the value distributions, otherwise we would have a single peak on the leftmost x-values and the others bars too short to be seen on the rest of the picture. These tell us that the majority of users have a lot of smaller debts with the bank (that can be related to the majority of smaller limits available) and the big majority of users give back to the bank amounts below 25K NT dollars each month: we can infer the majority of customers are middle-class people, who can afford a credit card but can't invest big amount on it. Finally, figure 4 shows the distribution of the target value, `credit_default`: it's self explanatory that 1/5 of the clients lose their credit card.

1.5 Pairwise Correlations and Eventual Elimination of Redundant Variables

To easy visualize the correlation between the variables we make use of a heatmap. Before doing that, we removed from the dataset all the non-numerical attributes plus the attributes `ps-*` since they encode how a customer deals with his/her credit card account month by month.
In figure 5 we show the correlation matrix. For sake of readability we removed the annotations, leaving the colors to represent the correlation between the attributes: dark green color represent high positive correlation, white color zero correlation. Negative correlation is not present. Let's list the correlations:

- a big dark green area: these are the attributes `ba-*` which they have a strong correlation between each other: the values lie in the interval [0.81, 0.95] and each month has higher correlation with its "neighbours", i.e. July has higher correlation with June and August. This behaviour can be justified because the attributes `ba-*` represent the accumulated bill that a customer has with his/her bank, so it's obvious that the bill of month *X* dependents more by the bill of month *X-1*;

- interesting enough, the attributes `pa-*` have a weak correlation between each other. They are located

on the bottom-right of the matrix and their values lie in the interval [0.14, 0.2];

- the attribute `limit`, the left and upper edges, has a medium correlation with all the attributes, except for attribute `age`, that is a weak correlation, with values lying in the interval [0.13, 0.28] ;
- the attribute `age` has weak correlation with all the other attributes.

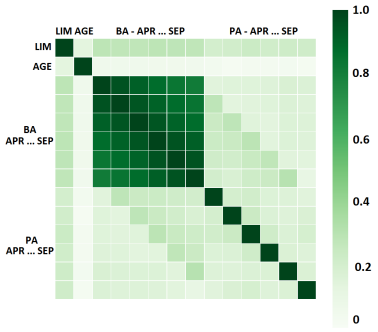

Figure 5: Heatmap of correlation matrix

From this we conclude that the only high correlation inside the dataset is between the attributes `ba-*`.
The two correlation matrices of the portions of the dataset with only customers with `credit_default = yes` and with `credit_default = no` have same structure of the correlation matrix already described, thus leading to any further information.
We don't feel to remove nor aggregate these attributes yet because having the history of the bills may be useful for certain prediction analyses based on temporal data. Anyway, we will take this into consideration in section 2 when we will consider the clustering analyses.

2 Clustering

2.1 Attribute Pre-processing

In order to apply the clustering algorithms we need to deal with the categorical attributes. The attributes `sex`, `education`, `status` and `credit_default` in our dataset do not have any ordinal meaning, thus we removed them from the dataframe used to apply the clustering algorithms. The attributes `ps-*` present both ordinal and categorical meanings, but since the "categorical" values (-2, 0) do not represent delayed payments they can be treated as the value -1 (recall, payed in time): since all these values can be seen as "no delayed payment" we transformed all in 0. As last step, since the clustering algorithms rely on distance measures such as the euclidean they are susceptible to the "curse of dimensionality", phenomena appearing when dealing with high-dimensional data. In order to reduce the dimension of the dataframe we aggregated the attributes `ps-*` `pa-*` and `ba-*`: the aggregation of `pa-*` is done by simply averaging them, while the aggregation of `ba-*` and `ps-*` follow a weighted average approach assigning weights in [1/6, 1] with fixed offset equal to 0.166 (=1/6) giving more weight to September and less weight to April. Table 3 shows a summary of the actions performed on the attributes.

Attribute	Actions
PS	-2, -1 turned into 0; weighted average, max weight on September
BA	Weighted average, max weight on September
PA	Simple average

Table 3: Attributes preprocessing

The resulting dataset has 5 columns: `limit`, `age`, `ps`, `ba` and `pa`, normalized with a min_max scaler.
For each of the clustering algorithms, we follow this approach:

1. Algorithm's parameter tuning;
2. Run Algorithm;
3. Define two **thresholds**, 0.1 and 0.3, and three **groups**: low (credit) default rate, medium default rate and high default rate;
4. For each cluster evaluate the ratio (credit default)/(cluster dimension):
   - if this ratio is < 0.1, put this cluster inside the **low** default rate group;
   - if this ratio is > 0.3, put this cluster inside the **high** default rate group;
   - otherwise, put this cluster inside the **medium** default rate;
5. For each **group** evaluate the distributions of the other attributes in order to find some sort of purity.

2.2 K-means

2.2.1 Parameter Tuning

We have run K-means changing the number of centroids between 5 and 50. Figure 6 shows how SSE (left) and Silhouette score (right) change when increasing the number of centroids K: both plots have K on the the x-axis and SSE and the Silhouette on the y-axis respectively. Both plots are used to choose a reasonable value of K to run K-means with. Looking at the Silhouette plot we can notice interesting values of K: K=11, K=17 and K=24 which are notable by being on the peaks. The red dots help the visualization of such values and the corresponding positions on the SSE plot.

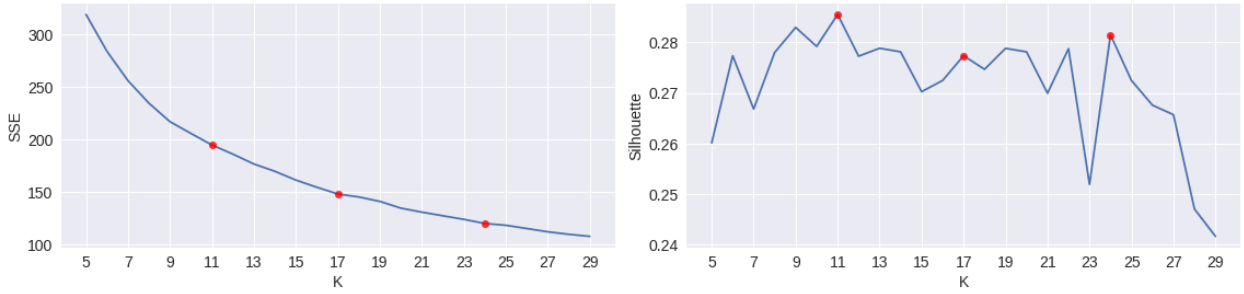

Figure 6: KMeans Parameter Tuning

2.2.2 Cluster Analysis: Credit Default Rate

With K=11 we get clusters all similar to each other with low purity w.r.t. the `credit_default` attribute, with percentages close to the percentage of the dataset (recall: 22,2%). With K=17, instead, we have some clusters with rates different from those of the whole dataset, even though they are consisted of a small amount of records, and it could be interesting to find some class of purity inside them w.r.t. other attributes (for instance all entries of the cluster are male or they are high graduated and so on...). With K=24 we have more extreme clusters but having in total the same number of records of K=17. After our analysis we decided to show those concerning K=17.

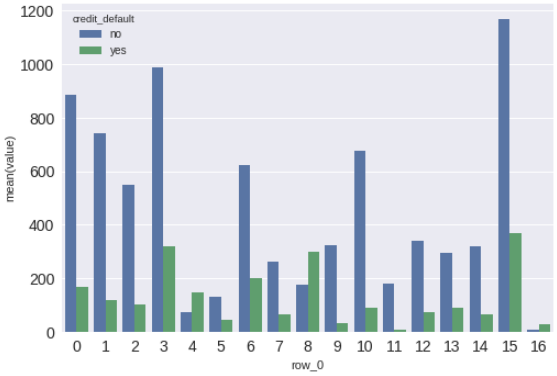

Figure 7: Default Rate for each cluster

Figure 7 shows, for each cluster (x-axis), how many records (y-axis) have `credit_default` = yes (green) and `credit_default` = no (blue). The rate of *yes* is between [0.037, 0.812] i.e. 3,7% and 81,2%.
The clusters are thus subdivided into 3 groups: as we can see in figure 7 clusters 9 and 11, having a small green bar, belong to the "low default rate" group with default rate equal to [0.084, 0.037] respectively, for a total of 542 records; clusters 4, 8 and 16, having the green bar higher than the blue one, belong to the "high default rate" group with default rate equal to [0.66, 0.63, 0.81] respectively, for a total of 726 records; the other clusters end up in the "medium default rate" group with default rate around 0.2.
Figure 8 shows a plot per group already mentioned, and for each cluster the average value of the attributes. What characterizes the "low rate" group is the high values of `limit` (around 400K NT dollars), and as consequence high value of `ba`; finally, `pa` values are also high and close to `ba`, meaning that people in these clusters have the possibility to pay the bank back with amounts close to their debts. In contrast, the "high rate" group has `limit` up to 60K NT dollars and `pa` being much lower than `ba`: we can state that people who have higher chance to lose their credit card are those belonging to a lower-class inside their society, who can not afford big credit cards and may have difficulty to sustain them.

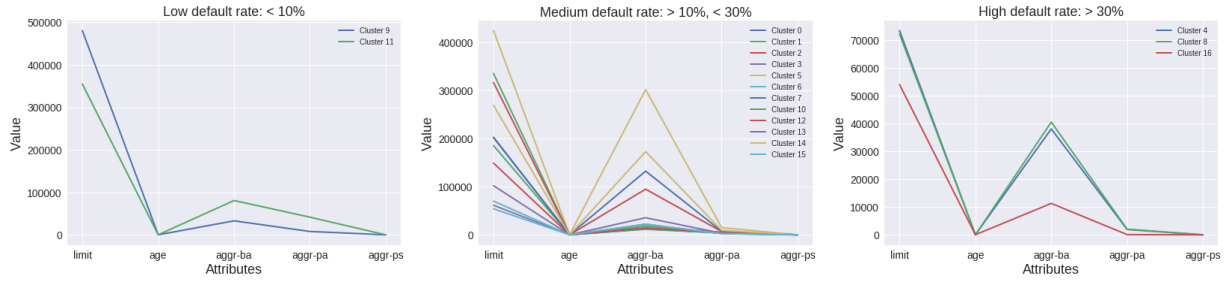

Figure 8: Cluster's average attribute's values

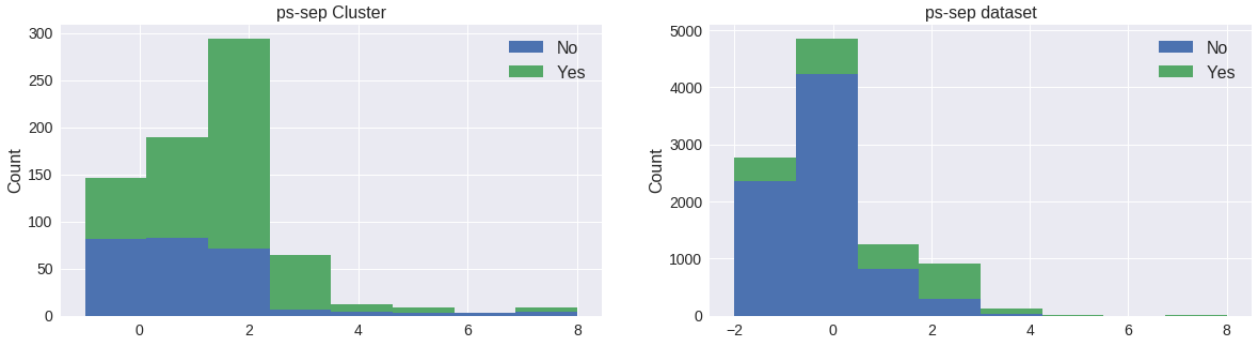

Figure 9: Distribution of ps-sep inside the high default rate group (left) and inside the dataset (right)

2.2.3 Cluster Analysis: Attributes Distribution

In this step we consider only the "purest" clusters w.r.t. the target variable to see any changes concerning the other attributes. In order to perform these analyses we use the whole set of attributes and not the "aggregated" version.
The **low** group does not present any differences with the dataset's distributions.
The **high** group has a different distribution concerning attributes `ps-*`: for sake of simplicity on figure 9 we show the distribution of `ps-sep`: on the left we have the distribution of our group, on the right of the whole dataset. For both cases the we have in green the amount of records with `credit_default` = *yes* and in blue = *no*. Our clusters are characterized by the peak on the values 1, 2 (recall, September payed off 1, 2 months in late) while we can see that the dataset has a majority of values 0 (recall, used revolving credit). The earlier months concerning our clusters have a peak on 2.

2.3 Hierarchical Agglomerative Clustering

2.3.1 Parameter Tuning

For the hierarchical agglomerative clustering algorithm we built the dendogram for every linkage method provided: single, complete, average and ward, using the euclidean and manhattan distance as distance metrics. For each clustering returned we computed the silhouette score and it turns out for all the clusterings to have a good score, with the one returned by the ward method to be the lowest.
Looking at the clusters we notice that all the methods but ward return a single big cluster and many tiny ones. Since the algorithm relies on `ps` and `pa` which we know to have values tightly packed around the 0, it is predictable that methods such as single and average linkage would suffer, but it seems that also complete linkage is not immune. As we will show in the next section, we have the exact same result with DBScan. Ward method does not seem to suffer from the high density of the data and returns balanced clusters. For these reasons, later on we analyze only HAC clustering returned by the **ward** linkage method with **euclidean** distance.

2.3.2 Cluster Analysis: Credit Default Rate

We have computed HAC in order to find a clustering. We have cut the dendrogram in order to find 17 clusters, because 17 is the number of clusters returned by K-means, and have seen if the result is comparable with the KMeans result. Figure 11 shows the resulting dendrogram, truncated in order to improve readability. The cut at distance 3.6 is represented by the black horizontal line and creates several clusters.
On figure 10, for each cluster, we show how many records have `credit_default` = *yes* (green) and = *no* (blue). The percentages of *yes* lie inside the interval [0.028, 0.781], i.e. 2.8% and 78.1%. The clusters 5, 6 and 10, with percentages of *yes* equal to [0.06, 0.02, 0.09] for a total of 925 records, belong to the low default rate group, clusters 4, 11 and 14, with percentages of *yes* equal to [0.64, 0.62, 0.78] for a total of 730 records, belong to the high default rate group and the rest to the medium one.

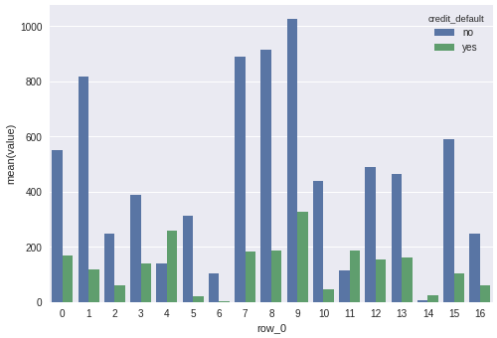

Figure 10: Default distribution for each cluster

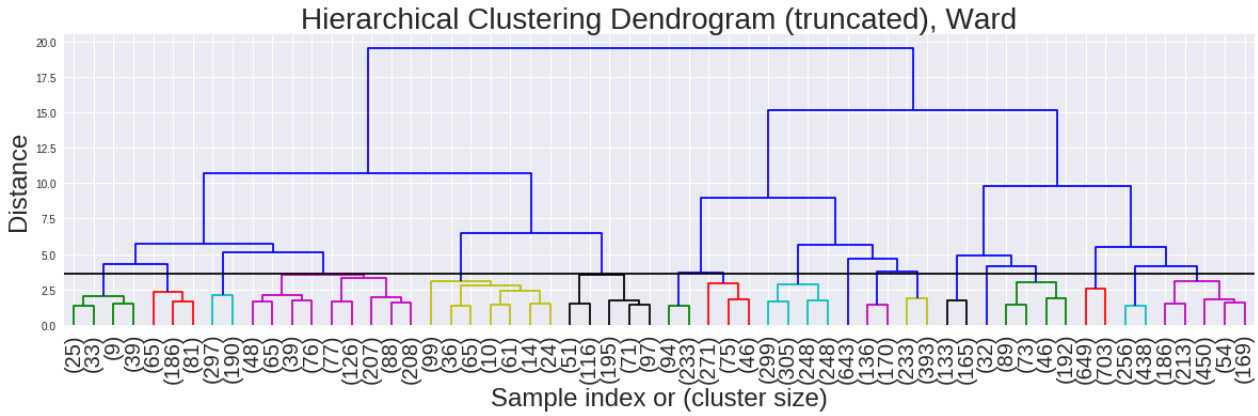

Figure 11: Dendogram returned by HAC with Euclidean metric and Ward linkage method

2.3.3 Cluster Analysis: Attributes Distribution

While for K-means the group of **low default rate** did not have any difference of distributions compared to the distribution of the dataset, in this case we do have one. Figure 13 shows the distribution of the `education` attribute for the low default rate group (left) and for the whole dataset (right). For both we have separated the records according to `credit_default` = *no* values on the left bar and = *yes* values on the right bar. While the dataset has a majority of records with *university* for both default outcomes (recall: university records are 47.4%), we have an opposite situation for the records inside our group, especially concerning those who have a default *yes*.

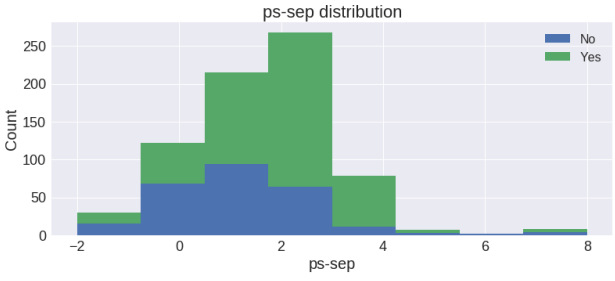

Figure 12: Credit default distribution for high default rate group on `ps-sep`

Concerning the **high default rate** group, like K-means, we get records with majority of values greater than

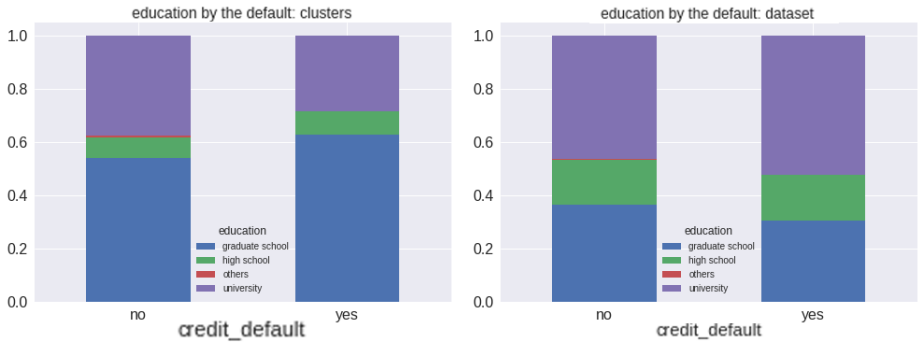

Figure 13: Distribution of `education` for low default rate group (left) and the whole dataset (right), divided by `credit_default`

0 on the `ps-*` attributes, meaning delayed payments. Figure 12 shows the distribution of `ps-sep` with green color identifying records who have default, and blue color identify records who do not.

2.4 DBScan

2.4.1 Parameter Tuning

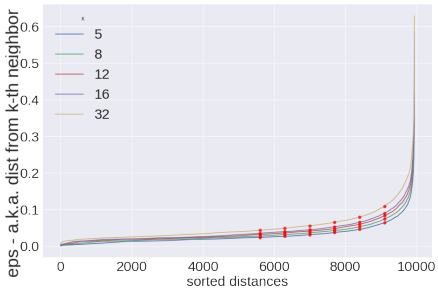

Figure 14: Sorted distances

Set *MinPts*=[5, 8, 12, 16, 32] we evaluate *eps* with the help of the plot of sorted distances shown in figure 14. The red dots show different combinations of (*MinPts*, *eps*) values we used to run DBScan more times.
After several runs of DBScan, and several parameter's tuning, we usually end up with clusterings with same structure:

- a very big cluster containing the big majority of the points and eventually 0 or a few more clusters. The noise is reasonable and the silhouette score is around [0.3, 0.5];
- many little clusters with a lot of noise and a very bad silhouette score.

When increasing *MinPts* further we end up on the second case, even with higher *eps*, and these configurations do not give us any good silhouette score either. On the left side of figure 15 we show one of the clustering with better silhouette score given by DBScan with *MinPts*=8 *eps*=0.065. The plot shows the scatter between `ba` (x-axis) and `pa` (y-axis) and a bit more information on top of it: the little black dots are the points classified as noise, the yellow points form the big cluster and the light blue color represent the small cluster.
We decided to run DBScan again on this clustering after removing the noise. Tuning *MinPts* and *eps* again and running DBScan one more time we get more or less the same structures listed before: big single cluster or many small ones with a lot of noise. The clustering we think is the "best" is shown on the right side of figure 15, with configuration *MinPts*=32 and *eps*=0.077, and it can be read in the same way as its neighbour, with the only difference being the number of clusters.
Since DBScan depends on `limit`, `ps`, `ba` and `pa`, these results may be justified by the high density of records that these variables present next to the 0, as shown in figure 3: for `pa`, these values occurrence is one order of magnitude grater than their successors. Since DBScan is density based, it might be susceptible to this high density of points and for this reason detects a single cluster with almost any parameter combination. As seen in section 2.3 we have the same result for HAC with single, average and complete linkage methods.

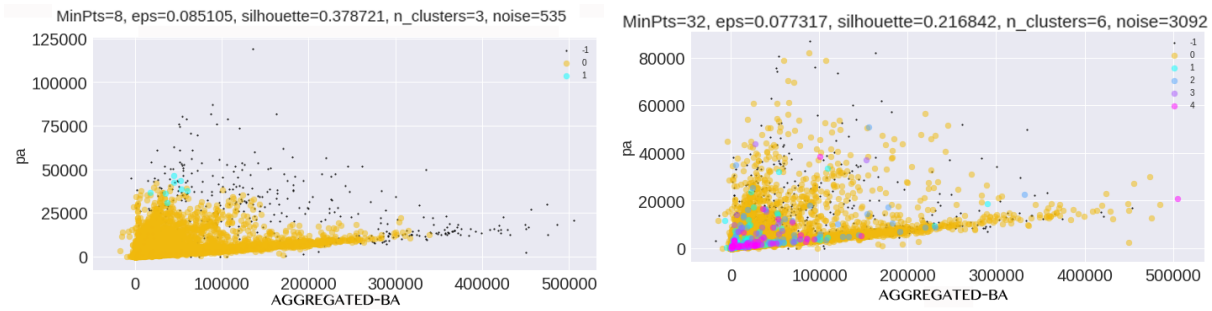

Figure 15: Result of DBScan: left, first run: right, second run

2.4.2 Cluster Analysis: Credit Default Rate

Figure 16 shows on the left, for each cluster but the noise, the credit default percentages (green =yes, blue =no) on the clustering resulted from the second run of DBScan in the same way as shown in figure 7. The values on the y-axis are the percentages and not the number of records in order to keep the figure packed given the differences of cluster's dimensions. Using the same metric as done for K-means and HAC we get an empty low rate group, i.e. there are no clusters with default rate less than 0.1, but we do get two "high default rate" clusters: clusters 2 and 4, with respectively [0.63, 0.74] default rate for a total of 146 records. Cluster 0, the big one, has a default rate a bit lower than the rate of the dataset (recall, 22,2%).

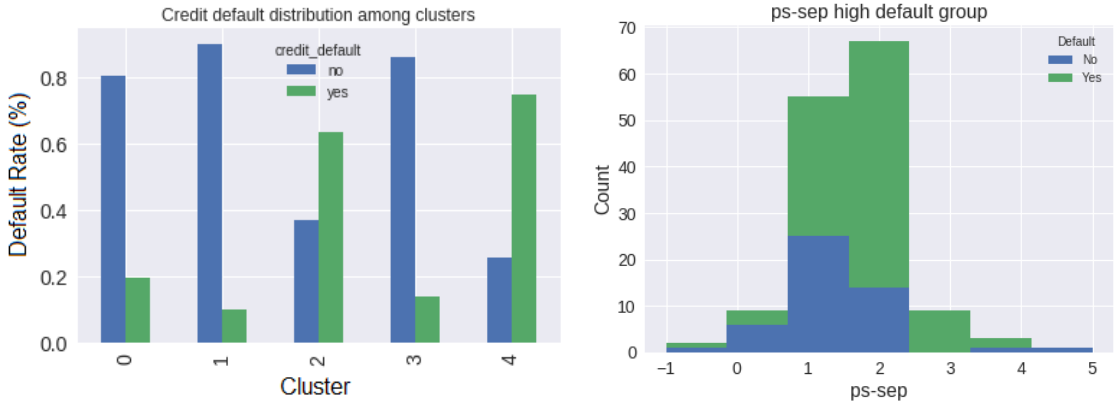

Figure 16: Credit default rate for each cluster (left) and `ps-sep` for high rate group (right)

2.4.3 Cluster Analysis: Attributes Distribution

As for K-means, the only interesting difference w.r.t. the dataset is `ps-*`. Figure 16 shows the distribution on the right of only `ps-sep`: also here we notice the peaks on positive values of `ps-sep` having the majority of the people on default (green). The previous months have also peak on value 2. Even though this group is very small, consisting of only 146 records, we have a confirmation about the impact of the attribute `ps-*` on the target variable.

Conclusions

The clustering analysis gave us useful information concerning the clusters with high default rate, even though the cluster sizes were a few hundreds records: they are characterized by the `ps-*` attributes having values > 0, i.e. delayed payments. And this is a result common to all clustering algorithms. Moreover, as we will see in section 4, `ps-sep` will be the most important variable concerning the decision tree models.

3 Association Rules and Pattern Mining

In order to perform this task we have done some changes to our dataset involving all the attributes with no categorical meaning. We grouped them into bins of fixed size and then we executed the algorithm. The itemsets and rules returned by the algorithm involved many items belonging to the group of variables `ba-*` since, as we have seen in figure 5, they are highly correlated between each other. In fact, we got rules such as (`ba-apr`=30000, `ba-may`=30000 → `ba-jun`=35000), which are legit rules because a debt amount on a given month will have similar amounts on the neighbour months. The itemsets and rules involving information like this were the big majority and it was hard to find different and interesting rules. For this reason, we decided to pre-process the dataset in the same way we did for the clustering analysis, which is summarized on table 3. In this way we will "remove" all the rules involving the same variable but on different periods.

Since the values of the categorical attributes are between 2 and 4, and because the aggregated attribute `ps` will now have a few values either, we need to discretize the attributes `age`, `limit`, `ba` and `pa` such that the number of distinct values is not too high, otherwise the importance of each discrete values would be lower. Table 4 shows the order of magnitude of the bucket sizes we used in order to discretize these attributes and the number of distinct values obtained. For the variable `limit`, `ba`, `pa` we tried different bucket sizes in order to see if we would spot any differences between the patterns. The attribute `ps` was binarized with values greater than 0 turned into 1. Finally, for each discrete value, we transformed it into a string inserting a placeholder in order to identify the attribute of a given number. The placeholders are shown in table 4.

Attribute	Bucket size	Distinct values count	Placeholder
`age`	5	11	_A
`limit`	O(30000)	25	_L
`ba`	O(3500)	22	_BA
`pa`	O(25000)	25	_PA
`ps`	[0, 1]	2	_PS

Table 4: Order of magnitude of the bucket sizes for our continuous variables and distinct values

3.1 Frequent Patterns

We executed the apriori algorithm changing the value of min support between [5%, 45%] with offset of 5. In table 5 we show the most interesting itemsets with a brief comment.

We add an analysis to the brief comment regarding the more interesting itemsets.
It is noticable that the item *0.0_PA* appears in many itemsets involving values `ba`, maybe because its occurrance is very high w.r.t. the other values (see figure 3), but to avoid too many distinct values we could not set the size of the bins much smaller. As a side effect we end up with an itemset of (*0.0_PA*, *no*) with 45% of support, which is counter-intuitive.
Rgarding itemsets concerning `ba`: those with positive values together with *yes* and those with negative values (i.e. credit and not debt) together with both *yes* and *no*. The itemsets with negative debt and *no* appear together with *0.0_PA* and *0_PS*: we could think that the owner of the card does not need to pay off the bank these months. But it is interesting that some people lost their card even though they had negative credit: this may result from external factors that the dataset does not include, or maybe because information is lost after the attributes aggregation and bucketing.
As a final note, the itemset (*yes*, *1_PS*) can be compared with the results we got from the clustering analyses concerning the high distribution of values grater than 0 of `ps` for clusters with high credit default rate.

3.2 Association Rules

We executed the apriori algorithm again in order to find interesting rules among the itemsets. Unfortunately, in order to have rules with enough confidence and lift we had to decrease a lot the minimum support value and for this reason the obtained rules have very low support. Table 6 shows a portion of the association

Itemset	Support	Comment
('yes', '8387.0_BA')	5%	Positive debt amount and payment=0 appear together with default="yes"
('yes', '8387.0_BA', '0.0_PA')	5%	

('yes', '-16613.0_BA')	5%	There are a few occurrences where negative debt (i.e. credit) leads to default="yes", even though ps=0
('yes', '-16613.0_BA', '0.0_PA')	5%	
('yes', '-16613.0_BA', '0_PS')	5%	

('-16613.0_BA', '0.0_PA', '0_PS', 'no')	15%	With a greater support we have that negative debt, together with ps=0, leads to default="no" even though pa=0
('-16613.0_BA', '0.0_PA', 'no')	15%	
('-16613.0_BA', '0_PS', 'no')	20%	

('yes', '1_PS')	10%	We have that half of yes (remark, 22%) have ps=1, i.e. delayed payments
('25.0_A', 'single')	15%	Almost all customers around 25 years old (that are around 1500-2000) are single

('graduate school', '0_PS')	25%	People with high level of education and ps=0 have less chance to lose their credit card
('graduate school', '0_PS', 'no')	25%	
('university', '0_PS', 'no')	25%	

('0.0_PA', 'no')	45%	As seen, the big majority of pa are close to the 0
('0_PS', 'no')	60%	It is ok because in original we have many records with values -2, -1 and 0

Table 5: The most interesting itemsets with their support

rules returned by our algorithm. We listed the rules which could give us information about the attributes we have missing values and the target variable and, for each rule, we show the support, confidence and lift values.

The first listed rules concern the target variable `credit_default`. The rules implying the value *yes* are all characterized by the presence of *1.0_PS* in the premise, together with other interesting values such as *10000.0_L*, the lowest amount of `limit` in the dataset, *0.0_PA* and *1500.0_PA*, the two lowest bucket values of the variable `pa`, and positive values of `ba`. All these rules have a small support, not high confidence but high lift value telling us that *yes* depends a lot on the values on the premise, especially *1.0_PS* (as we have seen for the clustering results).
Opposite features characterize the target value *no*, whose rules have high confidence but small lift, meaning that the values involved are almost independent from each other. Anyway, the values implying *no* are positive amount of pay off, `pa`, *0.0_PS* and small debt such as *3387.0_BA*. The low lift may result from the high frequency of both *no* and *0.0_PS*. The shown rules are the more general of all the returned ones, i.e. the rules with less items in the premise.
Concerning the columns `sex`, `education` and `status` the most important attribute seems to be `age`. Anyway, the confidence may be quite low, reaching also 60% and the lift value is mostly 1.4. The highest lift values are achieved by two rules that imply *high school*, both rules have *55.0_A* inside their premise, even though their confidence is very low. For all these cases we have considered the most general rules as well.

Rule	Support	Confidence	Lift
('10000.0_L', '1.0_PS') =>yes	1.3%	70%	3.1
('25000.0_L', '1.0_PS') =>yes	0.9%	65%	2.9
('1.0_PS', '0.0_PA') =>yes	3.5%	65%	2.9
('1.0_PS', '1500.0_PA') =>yes	1.9%	66%	3.0
('20.0_A', '1.0_PS') =>yes	0.8%	69%	3.1
('1.0_PS', '35.0_A') =>yes	1.8%	66%	3.0
('1.0_PS', '40.0_A') =>yes	0.8%	67%	3.0
('1.0_PS', '13387.0_BA') =>yes	1.2%	67%	3.0
('1.0_PS', '3387.0_BA') =>yes	0.8%	66%	3.0
('13500.0_PA') =>no	1.1%	94%	1.2
('13500.0_PA', '0.0_PS') =>no	1.1%	94%	1.2
('490000.0_L', '0.0_PS') =>no	2.2%	94%	1.2
('6000.0_PA', 'university', 'female', '0.0_PS') =>no	1.3%	94%	1.2
('3387.0_BA', '30.0_A', 'graduate school', '0.0_PS') =>no	1.3%	94%	1.2
('10000.0_L', '25.0_A') =>male	0.9%	66%	1.7
('20.0_A', 'married') =>female	1.2%	84%	1.4
('20.0_A', 'yes') =>university	1.5%	67%	1.4
('20.0_A', '1.0_PS') =>university	0.8%	69%	1.5
('55.0_A', 'female') =>high school	0.5%	51%	3.1
('55.0_A', 'no') =>high school	0.7%	46%	2.7
('490000.0_L') =>graduate school	1.4%	61%	1.7
('20.0_A') =>single	6.6%	82%	1.5
('40.0_A', -6613.0_BA) =>married	2.0%	75%	1.6

Table 6: The most interesting rules together with support, confidence and lift

3.3 Target Value Prediction

With these values we wrote a simple "classifier" based on the listed rules. Basically, the classifier checks the premises in order to find a combination of values that implies a *yes*, otherwise we return *no*. Tables 7 and 8 summarize the performance of our simple classifier. From the confusion matrix we notice that our rules have a low chance of predicting *yes* and thus we end up with a lot of *no* results giving us high accuracy (because we have 78% of *no* in the dataset) but a very low recall.

	True, Predicted	**False, Predicted**
True, Actual	685	1518
False, Actual	380	7357

Table 7: Confusion matrix of the rule based classifier

3.4 Missing Value Substitution

The last rows of table 6 show the rules concerning the attributes for which we have missing values. `age` has missing values too, but we did not get any rule implying its values, so we will not consider it during this

Accuracy	Precision	Recall	f1-score
0.809054	0.643192	0.310940	0.419217

Table 8: Performance of the rule based classifier

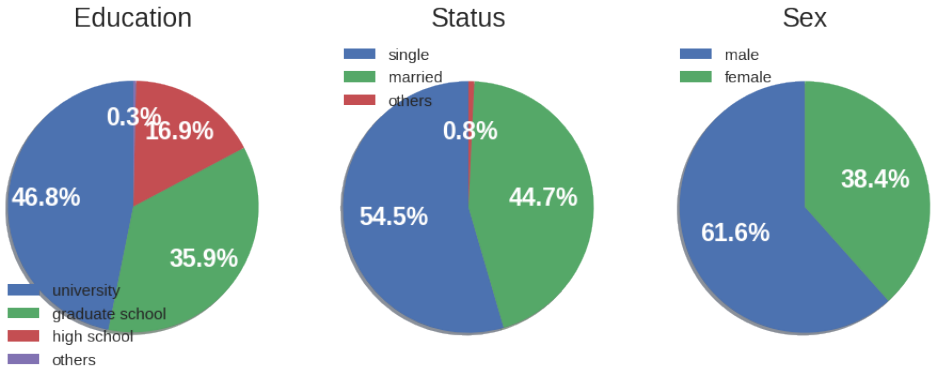

Figure 17: Distribution of the attributes `education`, `status` and `sex`

section. What we want to do is to see whether the chosen rules are good to replace missing values. Figure 17 shows the distribution of the values concerning the attributes `education`, `status` and `sex` after we used the rules to replace the missing values from the original dataset (before the cleaning phase). We can compare this pie charts to those of figure 2 and see that the distribution changes only about 1-2%.

4 Classification

To classify our data, we implemented various classification models, mainly focusing on the building of a decision tree, but also considering the performance of a neural network and a random forest on our dataset. The decision tree will be discussed in detail, whereas the neural network and random forest have been constructed out of own interest, which is why the report will only discuss them briefly.

The dataset was split into 70% training data, and 30% testing data, using KFold cross-validation to evenly test our models on all data.

Our decision process was composed by two steps: (1) Transform the dataset, (2) Tune the parameters of the model chosen.

4.1 Dataset Transformation

In order to find the best dataset transformation to train our model, we have built several datasets and trained our models with them.

- The first dataset is composed by using the whole dataset
- The second dataset is composed by transforming the categorical variable into indicator variables (a.k.a. dummy variables)
- The third dataset is composed using only the payment status columns
- The fourth dataset is composed using only the billing amount and the paid amount columns
- The fifth is the dataset with aggregated billing amount values and aggregated paid amount values (more details are in chapter 2)

A graphical visualization of these datasets is described below in table 9.

Table 9: Description of different datasets used to train our model

Dataset1	Dataset2	Dataset3	Dataset4	Dataset5
Limit	Limit			
Age	Age			
Sex	Sex_M Sex_F			
Education	Edu_University Edu_HighSchool Edu_Grad Edu_oth			
Status	Status_Single			
	Status_Mar			
	Status_oth			
PaymentStatus	PaymentStatus	PaymentStatus		
BillingAmount	BillingAmount		BillingAmount	Aggregated_BA
PaidAmount	PaidAmount		PaidAmount	Aggregated_PA
Class	Class	Class	Class	Class

4.2 Tuning Models

We tried to tune our models trying different values for every main parameter the model supports. Then, we have chosen the best configuration following three criteria: (1) maximizing the accuracy, (2) maximizing the F1-score, (3) minimizing the absolute value of the difference between the accuracy computed over the test and the training set: $|accuracy_{train} - accuracy_{test}|$.

4.3 Decision Tree

For the Decision Tree we computed an exhaustive search using all the combinations of the hyperparameters described in table 10 (table below). We found the best model using criteria (1) described in section 4.2, maximizing the accuracy score of the model (w.r.t. method used to choose best parameter configuration).

Criterion	Min Samples Split	Max Depth	Min Samples Leaf	Random State
gini	2	3	1	0
entropy	4	4	20	50
	6	5	40	100
	8	None	60	150
	...		...	
	20		200	

Table 10: The parameters we tried on our decision trees

To find the best decision tree model with highest accuracy score, we performed a grid search to find the best hyperparameters. However, independently of the model configuration, using the original dataset only ever gave us an accuracy score of 82% at best.
In an attempt to improve the model performance, the next step we tried was to alter our dataset in different ways to find out whether this improved anything. The way we altered the dataset is shown visually in table 9.
First, we encoded our categorical attributes into one-hot coding. This, however, did not improve the performance of the decision tree classifier.
Whilst still trying to improve our performance, it was very clear to see that the payment status played a very significant role in the whole classification task. And since dimensionality reduction is an often performed step before using a classifier on data, we thought that it could also work with our dataset. And indeed, only using the payment status, showed a slight, but still empirically better performance. Using

Dataset	Accuracy (train)	Accuracy (test)	Precision	Recall	F1-Score
1	0.82	0.82	0.66	0.36	0.46
2	0.82	0.81	0.66	0.36	0.46
3	0.82	0.82	0.67	0.37	0.47
4	0.79	0.78	0.58	0.13	0.2
5	0.82	0.81	0.62	0.33	0.43

Table 11: Performances of optimized decision trees on the different datasets

only the billing and paid amount, aggregated or not, was another idea, yet performed worse on our dataset, which is understandable, given the fact that the payment status is deemed so important for the classification.

4.3.1 Best Decision Tree

We chose the decision tree built with dataset 3, so the dataset containing only the payment statuses, as it yielded the model with the best cross-validated performance, which can be observed in table 11, and is also the least complex model out of the other decision tree models we built, since it requires simpler parameters for the construction of the classifier (read the maximum depth).
The tree uses the gini index, a minimum samples split of 2, a maximum depth of 4, a minimum samples leaf of 100 and a random state of 0.
The visualized tree is shown in figure 18.

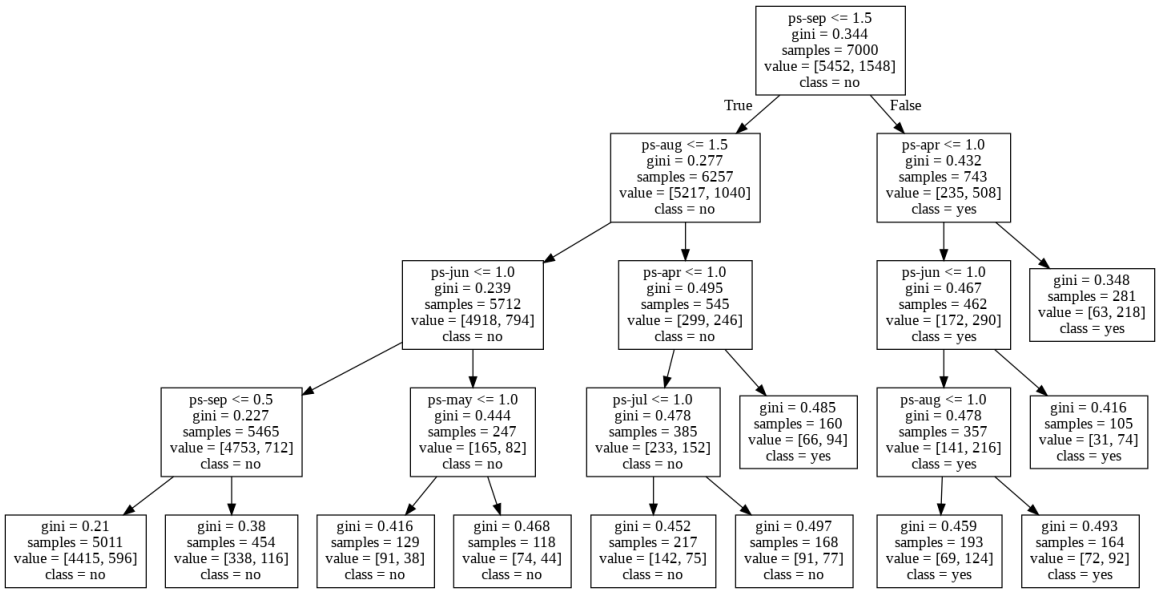

Figure 18: Optimized Decision Tree (payment status only)

Moreover, we have also computed the confusion matrix for the tree as seen in table 12.

	no (real)	yes (real)
no (predicted)	2236	100
yes (predicted)	430	234

Table 12: Confusion matrix for dataset of type 3 (payment status only)

4.3.2 Decision Tree Interpretation

The payment status of September is the most important attribute in our tree. This makes sense, considering that the payment status is aggregated, e.g. if the payment is in delay, the value of the payment status is a positive integer that grows each month the money is not paid back. And if a costumer is never in delay or does not even use the credit card, then a credit card default should not occur in the first place.
Therefore, also the other payment statuses are of more significant importance than the other attributes.
The other values, in fact, alone are not able to describe the payment statuses since you cannot understand the historical data that are given by the payment status variable.

Conclusion Our decision tree models never exceeded a cross-validated accuracy of over 82%. This could be explained by the dataset and our task to resemble a time series more than a simple classification task, and therefore can not actually yield good results using a simple classifier. Maybe, we can perform a better classification using some methods that are the time-aware, but we haven't tried any research on that topic.

4.4 Other Classification Algorithms

In order to find a classifier that performs better on our data, we also applied the random forest classifier, and a neural network.

4.4.1 Random Forest

We applied a random forest classifier. We computed an exhaustive search with several hyperparameters (not reported here, they are similar to the Decision Tree, see table 10) We have chosen the best model looking at $|accuracy_{test} - accuracy_{score}|$. With the best one we have found an accuracy of 81% as can be seen in table 13.

Accuracy	0.8125 (+/- 0.01)
F1-score	0.6473 (+/- 0.05)

Table 13: Accuracy and F1-score of the random forest

In general, the results from the random forest model performs worse than the decision tree. This behaviour can be explained given the structure of the data where maybe the best model is a sort of outlier from the average model.

4.4.2 Neural Network

For the neural network we computed an exhaustive search using the following values: We are not interested

Hidden Layers	Activation Functions	Alpha Parameter	Random State
(1)	logistic	1	0
(2)	tanh	0.5	50
(2,2)	relu	0.1	100
(3)		0.01	150
(5,1)		0.001	
(5,5,5)			
(10,4)			
(3, 3)			

in detail in the result of the search. The only result that we report is the best accuracy found with that model: $best_{accuracy} = 0.825$. To submit the result on Kaggle we used (hidden_layer_sizes=(2,2), alpha=0.5, activation="relu",max_iter=500) parameters.

5 Summary

Data understanding: During the first exploration of the dataset we studied its semantic and we cleaned it from missing values and outliers. During this phase we did not notice any patterns involving the attributes and the target variable.

Clustering: At this point we could find out a few common points involving the clusters: clusters with majority of `credit_default`=*no* have higher `limit` and `pa` and lower `ba` values (see figure 8), while clusters with majority of `credit_default`=*yes* have opposite characteristics and, most of all, have positive values for the attributes `ps-*`. Apart from one case, all the clusters do not present different distribution from the whole dataset concerning the attributes `education`, `sex` and `status`.

Frequent patterns and association rules: The itemsets and the rules obtained by running the apriori algorithm give us the same results of the clustering since we find together information such as `credit_default` = *yes* and `ps-*` > 0, but we also have itemsets involving negative debt (`ba-*` < 0) and `pa-*` ~ 0 with reasonable support and rules concerning `education`, `sex` and `status`, attributes we have not found almost any interesting purity inside clusters. The replacement of the missing values following the rules gives us similar distribution of the replacement we have done during the data understanding and the predictor built using the same rules is not capable to detect records with `credit_default`=*yes*.

Classification: We were able to produce a classifier that predicts `credit_default` with 82% accuracy. The decision tree performed best out of all tested models, but never exceeded this 82% accuracy, and other models only performed worse. This leads to the conclusion that classification might not be the ideal approach to make predictions on this dataset, but can be useful if an inaccuracy of 18% can be tolerated.