Introduction

Building a prediction is a common experiment in today's world due to machine learning technique´s rising, the great amount of data (quantity and access) and the number of people that gets involved in the field every day. Nevertheless, it is not a redundant task, it requires certain steps and many trials to achieve the expected results.

In this sense, and also for educational purposes, this work aims to address the machine learning process to deploy several techniques or algorithms to assess a binary classification problem. Specifically, we are going to work with a "credit card default" dataset and the main goal is to train and predict some algorithms to classify either if a person would fall or not in credit default considering demographic and banking characteristics. To accomplish this, we will use unsupervised and supervised approaches, also meta classifiers are included. These methods will be tested following three types of analysis: (1) with all the features available in the data set, (2) with a filter feature subset selection and (3) with a wrapper feature subset selection.

This document is divided as follows: In the first part we made a brief description of the data, then we present the methodology that was used; afterward, we addressed how the experiment or machine learning process was conducted and their respective results, finally it outlines some conclusions.

Description of the data

For this work, as the dataset was used the "Taiwan's credit card default clients" file. This one was taken from the UCI Machine Learning Repository and it was composed of 24 features and 30.000 instances. Basically, it contains some demographic and banking information from credit cardholders of a bank in Taiwan for 2005. The features are described as follow:

1	Balance	Amount of the given credit (NT$)
2	Gender	1 = male, 2 = female
3	Education	1 = graduate school; 2 = university; 3 = high school; 4 = others
4	Marital status	1 = married; 2 = single; 3 = others
5	Age	In years
6	Repayment status in September	History of past payment. Monthly payment records. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
7	Repayment status in August	
8	Repayment status in July	
9	Repayment status in June	
10	Repayment status in May	
11	Repayment status in April	
12	Amount of bill statement in September	NT dollar
13	Amount of bill statement in August	
14	Amount of bill statement in July	
15	Amount of bill statement in June	
16	Amount of bill statement in May	
17	Amount of bill statement in April	
18	Amount of previous payment September	
19	Amount of previous payment August	

20	Amount of previous payment July	
21	Amount of previous payment June	
22	Amount of previous payment May	
23	Amount of previous payment April	
24	Default payment	Yes = 1, No = 0

The data didn't have any missing value, but it was necessary to do some data preparation that will be addressed further in this paper.

Methodology

To achieve our main goal and for educational purposes, we took different approaches to analyze the data, so several experiments were conducted. First, we tried to assess the problem using various supervised classification algorithms including all the features that we had in the data set; the employed techniques were: *K-Nearest Neighbor, Classification Trees, Support Vectors Machines, Artificial Neural Network, Logistic Regression, Bayesian Classifier, Rule induction and Discriminant Analysis.*

After collecting the results of the first phase, it was used a univariate filter feature subset selection; for this task, the filter was established manually and set by percentage, keeping the 60% of the most important features according to the information gain method. The same eight algorithms were applied using the feature selection described above.

Then, we examined the feature selection with a wrapper method and deployed the same algorithms. In each of the three previous steps, a benchmark analysis was made between the estimated models. It is mean, that for the "all variables" study a benchmark was done to select the model with the best accuracy, for the "filter method" part another benchmark was conducted and so on.

Finally, metaclassifiers and some unsupervised classification algorithms were also employed; for the metaclassifiers part, a generic bagging was deployed using "mlr" package, the applied algorithm it the bagging was Artificial Neural Network because it showed the best accuracy in the "all variables" benchmark. Ada Boosting and Random Forests algorithm were also used.

Experiment

All the work related to machine learning algorithms and data preparation was done using R software, basically, "mlr" package, and others.

As mentioned before, the dataset did not have any missing values, but the data wrangling and engineering was necessary because of the categorical features, they did not appear as it was stated in the repository. Therefore, the first thing that we did was a dummy encoding for sex, education and marriage features, where the following levels were dropped in each one: men, other education, other marital status.

Furthermore, in the data exploration, we realized that the repayment status was not between the proper range (-1 to 9), it was slid one unit (going for -2 to 8); thus, we added one unit to the scale. Regarding this feature, we chose not to encode as a dummy due to it was ordinal, so it was used like that.

The proper explanatory data analysis was conducted, checking correlations, skewness, kurtosis, normality distribution and so on; we found issues in the dataset with these measures. For instance, most of the numerical type features presented positive skewness. Moreover, we checked for potential outliers, scaling the features in a range from 0 to a 1, and dropping the ones that showed values higher than 3 (Euclidean distance threshold). The dataset was reduced to 29,265 instances, so we dropped around 735 instances that were potential outliers.

For the proper deployment of the algorithms, the dataset was cut in two parts. The first subset was the training one, correspond to the 80 % of the observations in the original dataset, and it was used to train all the algorithms, also to predict and assess the generalization of the model. On other hand, the test set was only used to evaluate one model (classification tree), to establish how well the model would behave with unseen data.

Once the dataset was splitted, we proceeded to scale the features because some of them were in different units of measurement. To fulfill this task, we used the min-max scaling function. Afterward, some statistics summery was applied to check the results, we realized that for the machine algorithms deployment we would have some trouble with predictions because the classes of our label feature. To solve this unbalance data issue, we performed a data balance process to avoid that in our prediction the model will be tempted to misclassify the minority class; in general, machine learning algorithms tends to follow the majority, so they will perform well with the most common class and not that good with the uncommon one.

After all the process described before, we applied eight machine learning algorithms on the training set following the "mlr" approach (create a task, create a learner, train, predict, resampling). We implemented the following algorithms: K-NN Neighbor, Classification Tree, Logistic Regression, Support Vector Machine, Naive Bayes, Artificial Neural Network, Linear Discriminant Analysis, and Rule Induction. Basically, we trained each one of them, then obtained a prediction for the training set, observed their performance and finally we evaluated their generalization with a resampling approach. Finally, to examine if the models would improve their performance measures, we used some features subset selection approaches and we also applied three metaclassifiers to assess their performance against the standalone algorithms.

It is important to remark that for resampling in feature selection and for the assessed of the algorithms it was used repeated cross-validation, setting the parameters in 3 folds and 2 repetitions; except for Support Vector Machine where was applied "hold-out" because it became computationally too costly the “repcv” method.

Results

In the following section the results of the experiment are presented. First, we aboard all the supervised methods, then the outputs of the features subset selection and the results of the deployment of the metaclassifier. Finally, we will take a glimpse of the unsupervised approaches.

Supervised algorithms

1. All the features

K- Nearest Neighbor: This algorithm will classify a new observation taking into account its features and the features of the observations that looks like it, in other words, each new instance class will be predicted based on the similarity of its properties with the ones of other instances. For example, the algorithm will identify *k* credit holders that have similar attributes to the person that we are trying to define if he is going to fall in default or not and then uses the most common class of those *k* credit holders to predict the risk (class) of the new person.

Table 5. K- Nearest Neighbor algorithm output

Confusion matrix output

```
          predicted
true           0    1 -err.-
 0        14816  616    616
 1          331 9957    331
 -err.-     331  616    947
```

Prediction performance output

```
      acc       mmce
0.9631804 0.0368196
```

Resampling output

```
 acc.mean   mmce.mean
0.8031105   0.1968895
```

The performance of the k- nearest neighbor algorithm showed us some signs of alarm, the accuracy of the forecast is extremely high. According to this, the model made an accurate prediction in 96 % observations, this measure tracks the number of correct predictions of the model against the true class of the instances (the observed class). Hence, we can assume that the model is overfitting the training set; the evidence of this issue can be seen in the resampling aggregate, where the accuracy drops to 80 %, meaning that with new data, our model will classify correctly the 80 % of the new instances. In resume, the model estimates almost perfectly in the training set but not that well in the "real world".

Classification trees: As its name suggests, this algorithm follows a tree structure to predict a class, where each tree has its root node (a relevant feature to predict the class), leaf nodes (following features) and branches (output of the test). Works by partitioning the feature space into a number of smaller scenarios with similar response values using a set of splitting rules. The segmentation process is typically carried out using only one explanatory variable at a time.

Table 6. Classification trees algorithm output

Confusion matrix output	Prediction performance output

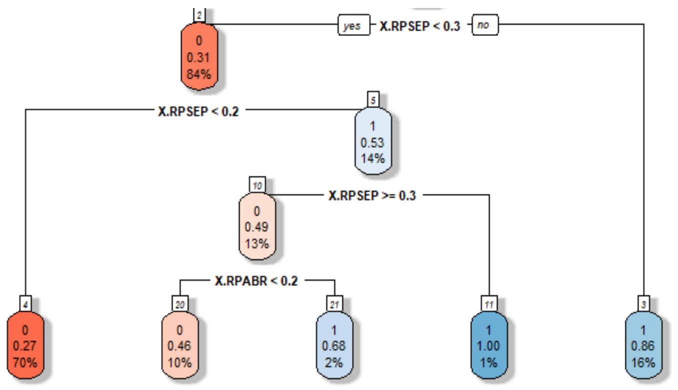
`predicted`
`true          0     1 -err.-`
`0       14667   765    765`
`1        6070  4218   6070`
`-err.-   6070   765   6835`
	`acc       mmce`
	`0.7342535 0.2657465`

	Resampling output

	`acc.mean  mmce.mean`
	`0.7310847 0.2689153`

Contrary to the output of the KNN algorithm, the classification tree exposes a more stable process of training and generalization. In this case, the performance measures for both summaries are almost the same, meaning that, for instance, the training model predicted correctly 73 % of the cases and that this performance will be the same way with new data. The resampling aggregates allow us to assess the generalization of the model, in other words, how well it is going to predict the new data or unseen data.

Figure 1. Classification Tree graph



For our data, the algorithm applied for building the tree used some of the available features, but the most relevant ones were X.RPSEP and X.RPABR. Thus, according to this algorithm, the possibility of falling in credit default will be associated with the repayment status in September and the repayment status in April.

Logistic regression: according to I-Chen & Chen-hiu "*logistic regression is used for binary classification; problems that are confined to two classes, basically, a logistic regression model specifies that an appropriate function of the fitted probability of the event is a linear function of the observed values of the available explanatory variables*".

Table 7. K- Logistic regression algorithm output

Confusion matrix output	Prediction performance output

`predicted`
`true          0     1 -err.-`
`0       13444  1988   1988`
`1        5243  5045   5243`
`-err.-   5243  1988   7231`
	`acc       mmce`
	`0.7188569 0.2811431`

	Resampling output

	`acc.mean   mmce.mean`
	`0.7177486  0.2822514`

For the logistic regression output we are going to focus our attention on the mean misclassification error (MMCE), this performance measure evaluates the number of the wrong classifications from the total input of the training set (in this case). The objective in a machine learning project is to minimize it; in our logistic model the mmce was 28 %, that means that this algorithm misclassified 28 % of the instances according to the observed class.

Going a little bit deeper on the output of the logistic regression we realized that some features were not statistically significant to the model (according to their p-value) some of them were: marital status, BILLJUN, BILLJUL, RPJUN and others. Meanwhile, features like RPSEP, BILLAGO, BILLJUL and others, increase the possibility of falling in credit default.

Support Vector Machines: The main purpose of this algorithm is to find a hyperplane in some feature space to build a classifier that allows us to separate the two classes or predict to which of the group belongs a new instance based on his attributes.

Table 8. SVM algorithm output

Confusion matrix output

```
        predicted
true         0    1 -err.-
  0      14050 1382   1382
  1       5111 5177   5111
  -err.-  5111 1382   6493
```

Prediction performance output

```
      acc      mmce
0.7475505 0.2524495
```

Resampling output

```
 acc.mean  mmce.mean
0.7379278  0.2620722
```

So far, SVM algorithm is the one with the best accuracy in the prediction performance, as well as in the resampling aggregate, thus we can assume that the deployment of this model will help us to classify correctly a greater amount of new data into the risk of falling in credit default. We must recall that the resampling approach for this algorithm is not the same that was used for the others, partly by the computational cost of this method (too high).

Naïve Bayes Classifier: this approach is based on the assumption of conditional independence, which means that it assumes that given the values of other attributes the effect of a particular feature on a class is independent of the first ones.

Table 9. Naive Bayes algorithm output

Confusion matrix output

```
        predicted
true        0     1 -err.-
  0      4290 11142  11142
  1      1167  9121   1167
  -err.- 1167 11142  12309
```

Prediction performance output

```
      acc      mmce
0.5214230 0.4785770
```

Resampling output

```
 acc.mean  mmce.mean
0.5230949   0.4769051
```

As we can see, the performance of this model decrease if we compare it with the previous ones, it shows an accuracy of 52 %, which means that with this algorithm we did an accurate prediction only in half of the times. Given our dataset, with the Naive Bayes

Classifier, it would be possible to predict if a credit holder will fall in default only 50 % of the time. This result in the accuracy of the prediction is explained because this is a parametric approach, it assumes normality and independence, two premises that are difficult to fulfill, especially the second one.

Artificial Neural Network - ANN: are inspired by the way the human brain works, particularly, on the biological neuron structure. The approach holds all its power in a neuron, which is the central unit and the one in charge of the computations; thus, it receives features as inputs and then operates a function of the weighted sum of these plus the bias to generate an output. Essentially, the prediction of the class it's done by a learning process that is based on examples.

Table 10. Artificial Neural Network algorithm output

Confusion matrix output	Prediction performance output

`        predicted`
`true        0    1 -err.-`
`  0      13708 1724   1724`
`  1       4997 5291   4997`
`  -err.-  4997 1724   6721`
	`      acc      mmce`
	`0.7386858 0.2613142`

	Resampling output

	` acc.mean  mmce.mean`
	`0.7344477   0.2655523`

With the Neural Network method, we were capable of predicting that a credit holder would fall in default with an acceptable chance of misclassification. According to the performance measures, the model misclassificate an observation in 26 % of the opportunities.

Linear discriminant analysis: The model is composed of a discriminant function based on linear combinations of the explanatory variables that provide the best possible discrimination between groups; which in turn, are distributed as multivariate normal with a common variance- covariance matrix.

Table 11. Linear discriminant analysis algorithm output

Confusion matrix output	Prediction performance output

`        predicted`
`true        0    1 -err.-`
`  0      13678 1754   1754`
`  1       5430 4858   5430`
`  -err.-  5430 1754   7184`
	`      acc      mmce`
	`0.7206843 0.2793157`

	Resampling output

	` acc.mean  mmce.mean`
	`0.7189153   0.2810847`

For the discriminant analysis, we examined the linear and the quadratic approach, finding better results with the first one for our data. According to this, the model behaves more or less equal than the previous ones, the accuracy is above 70 %.

Rule induction: is a technique that creates "if-else-then"- statements from a set of input variables and an output variable to predict the class. In "mlr" package we had available JRip algorithm, according to Sharma. S., et al, "*JRip classifier algorithm implements a proportional rule learner algorithm, Repeated Incremental Pruning to Produce Error*

Reduction (RIPPER), which was proposed by William W. JRip uses sequential covering algorithms for creating ordered rule lists”.

Table 12. Rule induction algorithm output

Confusion matrix output	Prediction performance output

`        predicted`
`true        0    1 -err.-`
`  0      14197 1235   1235`
`  1       4678 5610   4678`
`  -err.-  4678 1235   5913`
	`   acc       mmce`
	`0.7701011 0.2298989`

	Resampling output

	` acc.mean  mmce.mean`
	`0.7564151  0.2435849`

By far, this model exposed the best accuracy to predict the risk of credit default. As in the classification tree algorithm, the JRip method developed the rule by the X.RPSEP feature, so this one is really important to estimate the risk of default. Also, the resampling output shows the quality of the model, taking into account the generalization of this for new data.

Benchmark: to judge, at the same time, the performance of all the algorithms used in this first part, it was made a benchmark experiment. The main idea with this was to compare and rank the methods to establish which one helps us to classify better in the training set and to determine which would generalize better in new data.

As we can see in table 13, taking out the KNN algorithm due to the possible overfit problems, the approaches that showed better performance to generalize on new data were: JRip, SVM and Classification Trees.

Table 13. Benchmark experiment output, analysis with all the features

```
  task.id        learner.id acc.test.mean mmce.test.mean
1    task       classif.kknn     0.8008749      0.1991251
2    task      classif.rpart     0.7315708      0.2684292
3    task     classif.logreg     0.7182154      0.2817846
4    task       classif.ksvm     0.7375779      0.2624221
5    task classif.naiveBayes     0.5229976      0.4770024
6    task       classif.nnet     0.7119575      0.2880425
7    task      classif.linDA     0.7189930      0.2810070
8    task       classif.JRip     0.7541213      0.2458787
```

2. Filter Feature Subset Selection

For the second part of the analysis, it was used a filter feature subset selection approach, where the goal was to deploy all the algorithms but only with the features that had more relevance to the model according to the information gain approach. The relevance of each variable is assessed individually. To estimate the models, we set the filter threshold manually, holding the 60 % of the most important features.

As can be seen in table 14, with feature subset selection the accuracy of all the models remains almost equal as in the "all variables" analysis, expect the KNN algorithm, which shows a decrease in the accuracy and an increase in the mmce. In this step, KNN also showed overfitting problems and SVM reduced it computationally cost but it was not significant. For the other algorithms, we noticed similar behaviors in relation to the first analysis. In this section, the three best approaches in terms of performance to generalize on new data were: JRip, SVM and Artificial Neural Network (losing KNN).

Table 14. Benchmark experiment output, analysis with FFSS

```
task.id     learner.id acc.test.mean mmce.test.mean
 1    task   cl.kknn.f     0.7777994      0.2222006
 2    task   cl.rpart.f    0.7310268      0.2689732
 3    task   cl.logreg.f   0.7208593      0.2791407
 4    task   cl.nBayes.f   0.5157463      0.4842537
 5    task   cl.ksvm.f     0.7386665      0.2613335
 6    task   cl.JRip.f     0.7537717      0.2462283
 7    task   cl.nnet.f     0.7363144      0.2636856
 8    task   cl.linDA.f    0.7206649      0.2793351
```

3. Wrapper Feature Subset Selection

The wrapper feature selection approach is done by subsets of attributes, each one is chosen by a random process. Afterward, a model is fit and the performance is checked. Then, this is done for a lot of features combinations in a resampling setting and the best sequence is used. For this part, we also deployed the eight algorithms that we have been working with.

Analyzing the results of the wrapper method, we realize that in this scenario the majority of the algorithms display equivalent measurements among them, and in contrast with the two previous approaches, these values also move in the same range. Thus, we could assume that for our data there is no difference or improvement between deploying the algorithms with all the features or with some of them.

Table 15. Benchmark experiment output, analysis with Wrapper

```
task.id   learner.id acc.test.mean mmce.test.mean
1  task   cl.kknn.wra    0.7686232      0.2313768
2  task   cl.rpart.wra   0.7277798      0.2722202
3  task   cl.logreg.wra  0.7135889      0.2864111
4  task   cl.n.Bayes.wra 0.6172824      0.3827176
5  task   cl.ksvm.wra    0.7175164      0.2824836
6  task   cl.JRip.wra    0.7476672      0.2523328
7  task   cl.nnet.wra    0.7311818      0.2688182
8  task   cl.linDA.wra   0.7201593      0.2798407
```

4. Predict on the test set

To this point, we have done the essential part of a machine learning project, we began with the explanatory data analysis and data wrangling, passing through the feature engineering process and going deeper training eight different models, testing their results on the training sample, also was evaluated their generalization range and then some feature selection was done attempting to improve our models. So, we only have left to test a model with unseen data or new data, that's mean that we need to deploy our model in the test set. To complete this task, we will use the classification tree algorithm.

The test set corresponds to the 20% of the original data that we did no use to train our models, recall that before the training part we split our data set in two parts, thus, this test set had not been used until now. To use it, was necessary a balance process as in the training set.

Table 16. Classification Tree algorithm output test set vs training set

Prediction performance output	
Test set	**Training set**

`acc mmce`
`0.7434848 0.2565152`
	`acc mmce`
	`0.7342535 0.2657465`

The performance of the classification tree on the test set was even better than the one we saw for the training set; although, the improvement was not considerably high. The accuracy measure reached a value of 74.3%, stating that for the test set it predicted correctly the 74.3% of the instances' true class.

5. Metaclassifiers

Metaclassifiers were also employed in the training set with the idea of evaluating how they would improve the performance of the standalone algorithms. Basically, these type of methods use a set models, that can be built with the same algorithm but different subsets of observations or with several algorithms, to assess many times the performance of each model and them report an aggregate measure, for instance, they could train different models of k-nn algorithm, let's say ten, with various subsets of observations and then to report the performance measure it will take the mean of all the models.

Bagging: This term is derived on the method of bootstrap aggregation, which creates subsets of training observations by random selection with replacement. Thus, we have m training subsets composed by n quantity of observations from the original training set.

Using the “mlr” package we deployed a generic bagging with the Artificial Neural Network algorithm.

Table 17. Generic Bagging algorithm output

Confusion matrix output	Prediction performance output

`        predicted`
`true        0    1 -err.-`
`  0      13917 1515   1515`
`  1       5203 5085   5203`
`  -err.-  5203 1515   6718`
	`acc mmce`
	`0.7388025 0.2611975`

	Resampling output

	`acc.mean mmce.mean`
	`0.711741 0.288259`

Basically, in this bagging we deployed the Artificial Neural Network algorithm m times, each one of them with different subsets of the whole training set. Comparing this output with the NN implemented above, the performance measure did not improve.

Random Forest: are a modification of bagged decision trees that build a large collection of not correlated classification trees to improve predictive performance. According to Boehmke & Greenwell "*Bagging trees introduces a random component into the tree building process by building many trees on bootstrapped copies of the training data. Bagging then aggregates the predictions across all the trees; this aggregation reduces the variance of the overall procedure and results in improved predictive performance*"

Table 17. Random Forest algorithm output

Confusion matrix output	**Prediction performance output**
predicted true 0 1 -err.- 0 15396 36 36 1 65 10223 65 -err.- 65 36 101	acc mmce 0.996073095 0.003926905
	Resampling output
	acc.mean mmce.mean 0.8322706 0.1677294

The random forest for the training data set showed that the classification trees algorithms could predict correctly 99,6 % of the times if the credit holder belongs to the default or to the not default category, by comparing the predicted class against the observed class. As in the k-nn standalone algorithm, we can assume that the forest is overfitting the training set; the evidence of this issue can be seen in the resampling aggregate, where the accuracy drops to 80 %, stating that for new data the model does not behave so perfectly.

Boosting: is based on the idea that each observation has a weight. The goal is to in each iteration minimize the sum of the weights of the misclassified observations. The errors of each iteration are used to update the weight of the observations in the training set, in such way that the misclassified ones will have a heavier weight than the accurate classified ones. In this sense, the next model gives more relevance to the wrong predictions and it will try to minimize then.

Table 17. AdaBoos algorithm output

Confusion matrix output	**Prediction performance output**
predicted true 0 1 -err.- 0 14078 1354 1354 1 5130 5158 5130 -err.- 5130 1354 6484	acc mmce 0.7479005 0.2520995
	Resampling output
	acc.mean mmce.mean 0.7463063 0.2536937

Taking into account the meta classifiers, Adaboost algorithm is the one with the better performance, it is also stable in the sense that the measures of prediction performance and generalization are almost the same.

Unsupervised algorithms

To conclude the machine learning process, we performed some unsupervised algorithms following the clustering approach. In this sense, our main goal was to distinguish groups of observations that looked similar regarding their attributes or features. In the case of our dataset, we wanted to find groups of credit card holders that had similar characteristics between them but significant differences among other groups. To do this, we used the hierarchical method, the k-means algorithm and we also deployed the Gaussian mixture model. To implement the unsupervised algorithms, we used a subset of the original dataset because it was computationally too costly to use all the observations.

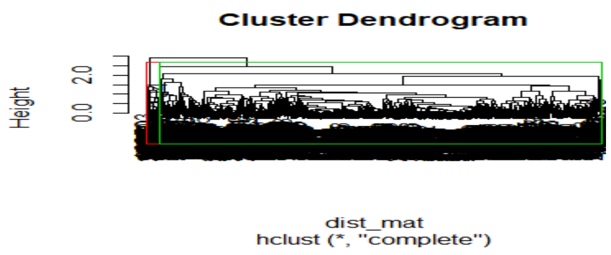
K-means: based on the proximity or distance between observations (neighborhood approach), this means that it tries to group the closest points to build a group. The idea is to find centroids in the cloud of data points and from this, assigned observations to the nearest centroid to build a cluster. One disadvantage of this method it's that you need to know the number of clusters that you want to establish.

To deploy the k-means algorithm on our "default credit card" dataset, we set two groups, the size of them were 1515 and 929.

Hierarchical approach: are based on a tree structure, where you can start building the clusters by observations and putting together many of them to build the groups or by one big cluster and in each step disaggregate it to determine which one are the observations belonging to each group.

For our data, we found two groups as shows the cluster dendrogram. It was also possible to establish that with the hierarchical approach we could identify correctly in the 60% of the time to which group belongs an observation.

Figure 2. Dendrogram



Gaussian Mixture Model: takes into account the distribution of the observations, it assumes a normal distribution on the data so it takes for granted that each Gaussian distribution represent a cluster. Therefore, Gaussian Mixture Model tends to group the data points belonging to a single distribution together.

For our data, the algorithm established nine possible clusters but we already know that we have only two (default or not default). So, maybe due to the assumptions that the algorithm takes, it had troubles to fit our data.

Table 18. GMM output

```
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 9 com
ponents:
 log-likelihood    n   df      BIC      ICL
       141775.5 2444 1745 269937.7 269919.8
Clustering table:
  1   2   3   4   5   6   7   8   9
203 169 243 277 556 310 255 335  96
```

Conclusions

As each method was deployed we realize that there were no significant differences between the models in terms of the performance measures, except for KNN and Naive Bayes algorithms; the first one was overfitting the data and the second one presented some troubles with the assumptions, the other six models behave equally. In this sense, it would be worth it to work without the standard parameters of each approach to evaluate if it would be any differentiation among them (was not within the scope of this work).

This similarity in the performance measures was even observed when metaclassifiers and feature subset selection was applied, calling our attention the fact that the random forest method showed overfit.

After all the training process was decided to test one solution with unseen data, we choose the classification tree algorithm to do this; the output exhibited that the performance measure in this step was even better than the one we saw for the training set and that the algorithm pointed as the most relevant features the repayment status in September and the repayment status in April, thus, the possibility of falling in credit default will be associated with them.

References

- Boehmke., Bradley & Greenwell., Brandon. (2019). Hands on Machine Learning with R. CRC Press. A Chapman & Hall Book.
- Hernández Orallo, J.; Ramírez Quintana, M.J., Ferri Ramírez, C. (2004). Introducción a la Minería de Datos. Pearson, Madrid.
- MLR Package Documentation. Tomado de: https://mlr.mlr-org.com/
- Parra, Francisco. (2017). Estadística y Machine Learning con R. Tomado de: https://rstudio-pubs static.s3.amazonaws.com/293405_4029f1f23f834b7195189d5504a436b2.html
- Shai Shalev-Shwartz and Shai Ben-David. (2014). Understanding Machine Learning: to theory to algorithms. Cambridge University Press, New York.
- SR Islam, W Eberle, SK Ghafoor. (2018). “Credit default mining using combined machine learning and heuristic approach” Proceedings of the 2018 International Conference on Data Science, 16-22.
- Sharma, Sunakshi & Mehra, Vipul. (2018). Default Payment Analysis of Credit Card Clients. 10.13140/RG.2.2.31307.28967.
- Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.