RECOMMENDATION OF CLASSIFIERS FOR THE CREDIT SECTOR

DATA MINING IN BUSINESS

Ágatha del Olmo Tirado | 2nd BIA | 09/10/2023

VNIVERSITAT DE VALÈNCIA

BUSINESS INTELLIGENCE & ANALYTICS

INDEX

INTRODUCTION

This report deals with research focused on the task of classifying bank customers on the basis of "creditscore" data. The main objective of this study is to develop the best possible classifier to distinguish between customers who will have a "good status" in relation to their loan and those who will have a "bad status" or who will not repay it adequately.

To carry out this research, different methods and computer tools will be used, including the use of classifiers such as JRip, Logistics or Naïve-Bayes and other techniques available on platforms such as R and Weka. The "creditscore" database is made up of 15 variables and contains information from more than 3845 customers, providing a robust dataset to perform the analyses.

This report will detail the research process, from selecting classifiers to evaluating different metrics and considering the costs associated with incorrect classification decisions.

PREVIOUS TASKS IN WEKA EXPLORER

We'll start by using Weka, a graphical environment that allows us to create and analyze experiments on classification tasks, and specifically its Explorer interface. Throughout the process we will be using the "o" seed.

THE ZEROR CLASSIFIER AND ITS MEANING

In the research process, one of the first approaches has been the use of the ZeroR classifier. This classifier yields surprisingly high results in terms of overall accuracy. However, to fully understand the motive behind this seemingly fantastic performance, it is essential to consider the context in which the "creditscore" database and this classifier are located.

On the one hand, accuracy is a metric that represents the fraction of elements correctly classified as positive out of all those that the model has classified as positive, a: VP / (VP + FP). On the other hand, ZeroR is a very simple classifier that is based on cataloging all individuals as the majority class of a dataset. This means that, as you would expect, most customers pay their loans, because all the false positives are those of the "good" group in which we have put all the "bad" ones, and if the accuracy is very high it means that there is a big difference in number between the groups. This observation has great implications in the context of the classification task, we can expect the classification of the model to be unbalanced and have more customers with good status than with bad, the disproportion should be approximated.

RELEVANCE OF THE VARIABLE "CAPACITY TO SAVE" WITH ONER

The use of the OneR classifier has yielded excellent results in terms of overall accuracy, which can make it seem like a very interesting classifier. However, it is essential to understand how this works.

The OneR classifier searches among all the variables which, when used to classify, gives the lowest total error. The discriminant variable it uses is the "capacity to save", that is, the savings capacity of each customer. Although it may seem logical to use this variable given its relevance in the financial context, when examining the proposed rules for classification, questions arise about its appropriateness.

When OneR uses one variable to predict about another, the selected variable is expected to be the most discriminant of all. This means that you need to be able to divide the data into groups with as much intergroup heterogeneity as possible. However, in this case, the variable "capacity to save" does not seem to fully meet this expectation.

The main problem is that, instead of establishing clear and effective divisions, the OneR classifier creates classifications based on layers or bands of savings capacity values. As we see in the image, the interpretation is meaningless: for example, the range from -57 to -34 is classified as "good", while the range from -34 to -28.5 is classified as "bad".

```
capacahorro:
        < -57.0 -> malo
        < -34.0 -> bueno
        < -28.5 -> malo
        < -13.5 -> bueno
        < -10.74        -> malo
        >= -10.74       -> bueno
(3619/3845 instances correct)
```

If your way of classifying was based on "poor capacity vs good capacity", you would be a classifier, at the very least, with much more sense of interpretation, but by using bands, you allow the classifier in some instances to make mistakes by classifying a poor saving capacity as good or vice versa.

EFFECT OF A COST-SENSITIVE CLASSIFICATION ON ZEROR AND ONER

Given the condition of this database, the error of classifying a person unable to repay the loan as capable of doing so carries a large cost that we will represent with a 3 in the cost matrix, since the monetary cost that this error has is more direct. To study its effect, we can see how it is modified when using the ZeroR and OneR classifiers.

The ZeroR classifier, as we have already mentioned, puts all individuals into the group more often than, in this case, the "good status", that is, the majority are able to repay the loan. Therefore, we find ourselves with an error of those in group b or complete "bad status", since none of them is well classified as b. There are 281 "missclassified" instances, which is the totality of the b, which, although they do not represent a very large part of the database, when you give them a triplicate cost, you get a considerable total cost of 843. In conclusion, we find that this classifier does not interest us at all because it does not adapt to the nature of the data.

With the OneR classifier you get a lower total cost of 610. This cost arises from the sum of the "good" clients incorrectly classified as "bad", with a cost of 1 each and a total of 58 misclassified instances, giving 58, and the "bad" clients classified as "good", with a cost of 3 each and 184 instances, which gives 549.

The key difference lies in how ZeroR, since it classifies all customers as "good," inevitably faces a high cost when mistakes are made; while OneR uses a somewhat more "sophisticated" criterion by relying on "ability to save" as the main variable, and although it makes errors in classification, it reduces the total cost compared to the other classifier.

EVALUATION OF STATISTICAL CLASSIFIERS: LOGISTIC REGRESSION AND LINEAR DISCRIMINANT ANALYSIS (LDA)

In this stage of the analysis, we have used the functional classifiers, Logistic Regression and Linear Discriminant Analysis (LDA) and the Naïve Bayes decision tree.

In principle, LDA cannot be used in this database due to a fundamental constraint: it requires all input variables to be numeric. However, there are nominal attributes in the creditscore database that are not numerical, and this poses a challenge.

Given this limitation, it has been decided to binarize the nominal variables, so that they are compatible with the LDA. This is achieved by using the "NominalToBinary" filter, which creates binary variables for each level of the factor present in the nominal attribute. However, binarization can significantly increase the dimensionality of the data and can have negative effects on some classifiers, for example, it has been observed that, in particular, Naïve Bayes shows a higher sensitivity to binarization, while Logistic Regression is hardly affected. This is due to the fact that Naïve assumes a conditional independence, which is greatly affected, as we create variables that may seem independent while they are not, and on the other hand, Logistic Regression does not make such strong assumptions and is more capable of handling large amounts of data.

After binarization, a significant problem persists in the database: the high number of instances with missing values. To address this issue, the "ReplaceMissingValues" filter has been used, which replaces the missing values with the mean or mode of their respective attributes.

With these pre-processing, it has been possible to apply both Linear Discriminant Analysis (LDA) and Logistic Regression and Naïve Bayes to an already coherent and comparable dataset through the classifier of Weka's Explorer interface. The results are as follows:

	Logistics	LDA	Naïve Bayes
Total Cost	81	229	195
Hit rate	98.9857%	97.6333%	96.697%
Accuracy in the "good" class	0,994	0,981	0,99
Accuracy in the "bad" class	0,932	0,906	0,726
Weighted Average Accuracy	0,99	0,975	0,971

The best results have been indicated in green, and there is a clear winner: Logistics. Despite the fact that LDA and Naïve Bayes have demonstrated good performance in ranking customers based on their status, logistics has had the best of all performances by far.

ANALYSIS OF THE DIFFERENT CLASSIFIERS

ANALYSIS OF THE NAIVEBAYES, LOGISTICS, J48, JRIP AND HOEFFDING TREE CLASSIFIERS IN THE WEKA EXPERIMENTER

We now use Weka's Experimenter interface. This interface interests us more than Explorer because it shows us comparison of hypotheses to determine whether the best or worst performance of one classifier or another is really significant or not. We use the binarized database and the nominal database, with the missing values previously replaced. To carry out the most complete analysis possible, we are going to compare 5 different classifiers: Naivebayes, Logistics, J48, JRip and Hoeffding Tree. The characteristics of these classifiers to take into account to better understand them are the following:

- Naïvebayes: this classifier is based on obtaining conditional probabilities and its exit is a probability of belonging to group x. Although it is quick and easy to interpret, its ability is limited by "duplicating" the variables after binarizing them, since it will assume them to be independent when they are clearly dependent. On the other hand, the results are usually very accurate because even so the order of the probabilities does not usually alter much.
- Logistics: This classifier is based on a logistic function and is used in precisely binary classification problems, where the result can only be dichotomous in nature. This is sensitive to correlation and there is a risk of overfitting, but as in the Naïvebayes, the results remain the same despite the fact that the estimate is somewhat different from what it would have been.
- J48: This classifier is a version of the classic C4.5 but more limited, it builds a decision tree from the root node creating subsets and minimizing variability. In this case, pre-pruning and post-pruning methods are usually implemented to reduce error.
- JRip: This rule-based classifier looks for patterns and applies them to new data through the Repeated Incremental Pruning for Error Reduction (RIPPER) algorithm in both pruning and growth.
- Hoeffding Tree: This classifier "grows" a decision tree, where a node expands when there is sufficient statistical evidence that there is a future optimal division. It's simple to implement and interpret, but once you create a node you can't change it, you can't fix it.

To analyze them, we will look at various indicators such as Percent_correct, Kappa_statistic, TruePositiveRate, FalsePositiveRate, IR_Precision, and F_Mesure. First, let's look at what each one means:

- Percent Correct: percentage of correctly classified individuals out of the total.
- Kappa_statistic: Compare the acuacuity with what would be obtained at random
- TruePositiveRate: Also called sensitivity, the ratio of true positives to the total positives in the dataset.

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)},$$

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

- FalsePositiveRate: Also called specificity, the ratio of false positives to the total negatives in the dataset.
- IR_Precision: Calculate the number of individuals correctly classified in relation to the total number of positives,
- F_Mesure: A harmonic weighted average between accuracy and completeness that provides a good balance between these.
- Area_under_ROC: The area under the ROC curve in a graph showing the TPR on the y-axis and the FPR on the x-axis, measures the overall performance of the model.

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

Each classifier will be represented by a code in parentheses, being (1) J48, (2) JRip, (3) Hoeffding Tree, (4) NaïveBayes, and (5) Logistics. Hypothesis testing is carried out at all times with a 5% level of significance.

1. ANALYSIS OF PERCENT_CORRECT

Dataset (1) trees. J4 | (2) rules (3) trees (4) bayes (5) funct

---

'WekaExcel-weka.filters.u(100) 97.17 | 97.17 93.21 * 96.79 99.01 V

'WekaExcel-weka.filters.u(100) 97.13 | 97.23 92.93 * 97.30 99.01 V

(v/ /*) | (0/1/0) (0/0/1) (0/1/0) (1/0/0)

Logistics has obtained the highest accuracy, with 99.01%, this may suggest that, in this specific dataset, the relationships between the characteristics and the target variable are predominantly linear. On the other hand, the Hoeffding Tree has obtained an accuracy of 93.21% in the first base and 92.93% in the second, values significantly lower than the others, which may be due to its inability to learn from its mistakes and go backwards in the construction of the tree.

2. ANALYSIS OF KAPPA_STATISTIC

Dataset (1) trees. J | (2) rule (3) tree (4) baye (5) func

'WekaExcel-weka.filters.u(100) 0.79 | 0.79 0.36 * 0.78 0.93 V

'WekaExcel-weka.filters.u(100) 0.78 | 0.80 0.35 * 0.80 0.93 V

(v/ /*) | (0/1/0) (0/0/1) (0/1/0) (1/0/0)

Logística has obtained a Kappa Statistic of 0.93, indicating a very high agreement between its predictions and the actual values, and is significantly effective in this dataset. On the other hand, the Hoeffding Tree has obtained a Kappa Statistic of 0.36 and 0.35 in each database, indicating a significantly low agreement compared to the others.

3. ANALYSIS OF TRUE POSITIVE RATE

Dataset (1) trees. J | (2) rule (3) tree (4) baye (5) func

'WekaExcel-weka.filters.u(100) 0.99 | 0.98 0.98 0.97 * 1.00 V

'WekaExcel-weka.filters.u(100) 0.99 | 0.98 0.98 0.99 1.00 V

(v/ /*) | (0/1/0) (0/1/0) (0/0/1) (1/0/0)

Logistics has achieved the highest True Positive Rate, with a value of 1.00. This indicates that Logistic Regression is significantly effective in identifying positive instances in the dataset, achieving a detection considered perfect. On the other hand, Naïve Bayes has obtained a significantly lower True Positive Rate, with a value of 0.97.

4. ANALYSIS OF FALSE POSITIVE RATE

Dataset (1) trees. J | (2) rule (3) tree (4) baye (5) func

'WekaExcel-weka.filters.u(100) 0.21 | 0.19 0.66 V 0.12 * 0.07 *

'WekaExcel-weka.filters.u(100) 0.22 | 0.17 0.67 V 0.20 0.07 *

(v/ /*) | (0/1/0) (1/0/0) (0/0/1) (0/0/1)

In this case we measure the other way around, we look for the classifier with the lowest value. Naïve Bayes and Logistic Regression obtained a fairly low False Positive Rate, with a value of 0.12 and 0.07 respectively. This suggests they are significantly the most effective at avoiding false positives, making them an excellent choice for minimizing these types of errors. The Hoeffding Tree has, on the other hand, the highest False Positive Rate, with a value of 0.66. This suggests that this one is significantly less effective compared to the other classifiers evaluated.

5. ANALYSIS OF IR_PRECISION

Dataset (1) trees. J | (2) rule (3) tree (4) baye (5) func

'WekaExcel-weka.filters.u(100) 0.98 | 0.99 0.95 * 0.99 V 0.99 V

'WekaExcel-weka.filters.u(100) 0.98 | 0.99 0.95 * 0.98 0.99 V

(v/ /*) | (0/1/0) (0/0/1) (1/0/0) (1/0/0)

Naïve Bayes and Logistics have obtained an IR Accuracy value of 0.99, which is a positive indication to be significantly (Naïve Bayes is only significant for the first database, the binarized one) efficient in accuracy. But the Hoeffding Tree has the lowest IR Accuracy value, at 0.95, and this suggests that it achieves significantly less efficient accuracy compared to the other classifiers evaluated.

6. ANALYSIS OF F_MESURE

Dataset (1) trees. J | (2) rule (3) tree (4) baye (5) func

'WekaExcel-weka.filters.u(100) 0.98 | 0.98 0.96 * 0.98 0.99 V

'WekaExcel-weka.filters.u(100) 0.98 | 0.99 0.96 * 0.99 0.99 V

(v/ /*) | (0/1/0) (0/0/1) (0/1/0) (1/0/0)

Logistic regression has reached the highest value of F-Measure, with 0.99. This suggests that Logistic Regression is able to correctly classify positive and negative instances without significantly compromising either. On the other hand, the Hoeffding Tree has a significantly lower F-Measure value, at 0.96, so it shows a less efficient balance between accuracy and recall compared to the other classifiers evaluated.

7. ANALYSIS OF AREA_UNDER_ROC

Dataset (1) trees. J | (2) rule (3) tree (4) baye (5) func

'WekaExcel-weka.filters.u(100) 0.92 | 0.90 0.65 * 0.98 V 1.00 V

'WekaExcel-weka.filters.u(100) 0.92 | 0.91 0.65 * 0.98 V 1.00 V

(v/ /*) | (0/1/0) (0/0/1) (1/0/0) (1/0/0)

Logistic regression obtained the highest value, with 1.00. This suggests that Logistic Regression is perfectly effective at discriminating between positive and negative classes, making it an outstanding option. Naïve Bayes also had a high value of 0.98. Both have a significantly high capacity compared to the rest. However, the Hoeffding Tree has the lowest value at 0.65. This suggests that it is significantly less effective compared to the other classifiers evaluated.

In summary, logistic regression has consistently and significantly demonstrated the best performance throughout the entire evaluation. This suggests that, in this specific dataset and just looking at the results of the experiment in WEKA, Logistic Regression is the preferred classifier for the classification task. Something interesting to comment on is the fact that the value did not change in any metric in both databases (binarized and non-binarized) when using this classifier, and it is not expected that the binarization of the variables will have an impact on the performance of the logistic regression, because if it has not been manually converted as we did in the first database from nominals to binaries, It does this automatically through the "one-hot-encoding" technique, which results in the same thing.

ANALYZING THE LDA CLASSIFIER IN RSTUDIO

To continue with the comparison of the classifiers we have to use R. First we analyze the LDA. This discriminant function is based on machine learning, looking for the characteristics that most differentiate the groups, maximizing the distance between the class means and minimizing the interclass variance. We're going to need our database binarized, but for it to work properly we change the headers from the duplicates in Excel to owned, rented, otras_viviendas, etc.

After importing the database and performing a lda() with the "MASS" package, we obtained the confusion matrix, which helps us to see the result more clearly.

	0 (good)	1 (bad)
0	3543	21

1	67	214

The confusion matrix shows the number of instances that were classified correctly (diagonally) and those that were classified incorrectly. The x-axis represents the actual values, and the Y-axis represents the predicted values. In class 0 (good status), 3543 instances were classified correctly, and 21 were classified incorrectly. In class 1 (bad status), 214 instances were classified correctly, but 67 were classified incorrectly. It is interesting to note that we are concerned about the second mistake more than the other since it has a direct financial impact on the company, and this one, in this case, is greater, which if we add the fact that there is a much lower proportion of "bad", it is a very high error, and we see this more clearly in the percentage of those correctly classified:

Class	0 (good)	1 (bad)
% Correctly Classified	99.41%	76.16%

Despite having a total percentage of correct answers that remains at 97.71%, what we feared happens: despite having a solid overall performance, there is significantly more classification error in the second group than in the first.

We then performed a linear discriminant analysis with the lda() function, which shows us the prior probabilities of belonging to classes "0" and "1", of which it is much higher for class 0.

0 (good)	1 (bad)
0.92691808	0.07308192

This makes sense in the context of the problem, because as we should expect, there are far fewer people who don't pay back loans.

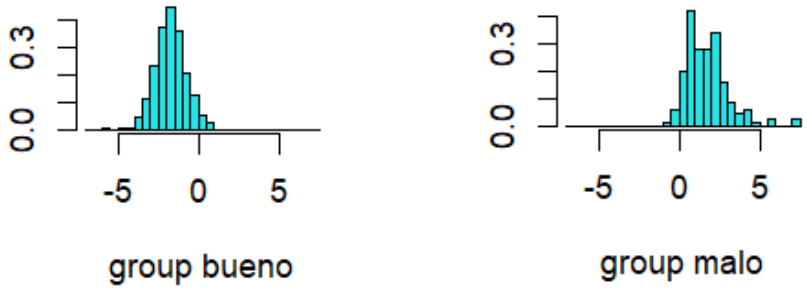
We also obtain the discrimination coefficients. These coefficients indicate how the predictor variables are combined to perform the classification.

Variables	**Discrimination coefficients**
Age	-1.091369E-02
property	-1.690713E-01
rent	-1.006806E-01
otras_viviendas	6.157888E-01
timePres	-1.457970E-02
age	-4.875063E-03
married	2.524644E-03
widower	3.735027E-01
bachelor	-1.148735E-01
separate	3.596654E-01
divorced	4.095982E-01
fixed	-5.839712E-01
temporary	-3.474206E-01
freelance	6.603696E-01
otros_trabajos	1.219505E+00

expense	3.673844E-03
revenue	1.383552E-02
assets	-2.086010E-05
debt	5.432841E-05
amount	2.398743E-03
price	-2.584014E-04
funded	-8.614060E-03
Capacity	-2.155696E-02

Variables with a positive coefficient (otras_viviendas, married, widowed, separated, divorced, freelancer, otros_trabajos, expenses, income, debts and amount) indicate that their highest value is associated with class 1, and those with a negative coefficient (seniority, property, rent, time, age, single, fixed, temporary, assets, price, financed, capacity) indicate that their higher value is associated with belonging to group 0.

Finally, we've set up a chart showing the ranking:



In summary, a high overall percentage of correct answers is observed, suggesting that the model is effective in ranking. However, it is also noted that class "0" has a higher percentage of correct classification compared to class "1".

As for the comparison between this and the logistic one, we can study the accuracy, the Logistic Regression had an accuracy of 99.01%, while the LDA had an accuracy of 97.71%. In terms of accuracy, Logistic Regression outperforms LDA. Since Logistic Regression has a higher correct percentage and provides additional metrics, it seems to be the preferred classifier in this specific context.

ANALYZING THE RPART CLASSIFIER IN RSTUDIO

Now we'll look at the RPart classifier to compare it to the others, again with RStudio. RPart focuses on dividing the dataset into more efficient subsets from a C4.5 tree.

In order for us to use the rpart() function, we convert all variables in RStudio to numeric.

The output of rpart() is merely predictive in this case, and is composed of the node number, the split criterion, the number of individuals in the group (n), the deviance,

and the predicted value (yval). And terminal nodes are expressed with a "*" at the end. Some examples are:

1) root 3784 257.5761000 0.073467230: This is the root node, which includes all observations (3,784) and has a deviation of 257.58 and a ratio of EstatusPrest observations equal to 7.35%.

2) capacity>=-28.5 3471 99.0025900 0.029386340: The first child node is based on a division of the variable 'capacity'. If 'capacahorro' is greater than or equal to -28.5, observations go to this node. There are 3,471 observations on this node (313 less than its parent node), with a deviation of 99 and an EstatusPrest observation ratio equal to 2.94%.

4) amount< 2262.5 3417 63.7635400 0.019022530: This node represents a second division based on the variable 'amount'. If 'amount' is less than 2262.5, observations fall into this node. There are 3,417 individuals (54 less than in the previous one) in this node, with a deviation of 63.76 and a proportion of EstatusPrest observations equal to 1.9%.

The tree continues to divide the observations into successive child nodes based on different predictor variables and cut-off values until there is a final node (*) and instead of creating a child node it creates a "sister node", that is, a node that arises from the "parent node" of which it has just been terminated, and follows the same pattern.

This decision tree is a classification model that uses predictor variables to predict the output variable EstatusPrest (class 0 or 1). The tree snaps to the data and divides the observations based on the values of the predictor variables to classify them into different EstatusPrest categories.

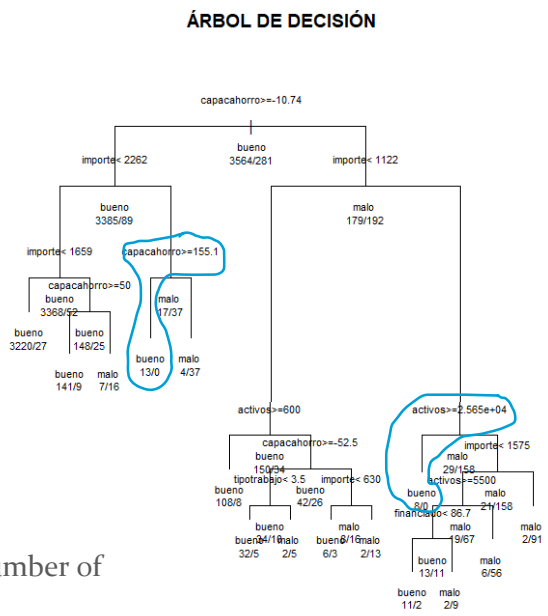
Now we convert the tree to a classification tree and not a regressive one, and for that we convert the EstatusPrest variable to a factor type. We get another tree, but now, in addition to the information it gave us before, it stops giving us the deviation, it gives us loss (number of misclassified individuals, and yprob becomes a kind of vector with the probabilities of belonging to each class. To make it easier to understand, the first three lines look like this:

1) root 3845 281 good (0.926918075 0.073081925)

2) capacity>=-10.74 3474 89 good (0.974381117 0.025618883)

4) Amount< 2262.5 3420 52 Good (0.984795322 0.015204678)

Node 1) includes all individuals and the original probability of belonging to each class, then branches out with savings capacity greater than or equal to 10.74, individuals decrease and the probability of belonging to the first class grows, and the same happens in the next node that emerges with an amount less than 226.2 and already achieves a 98.47% probability of belonging to the "good" group.



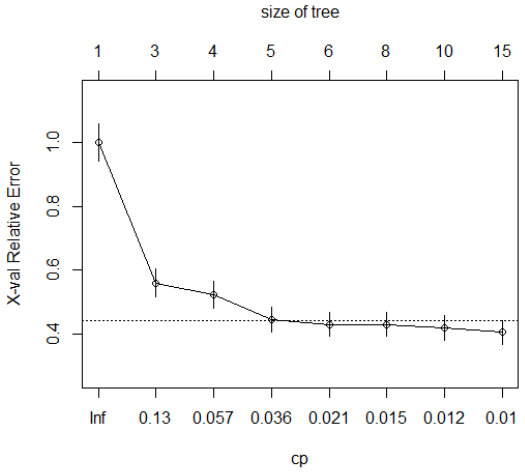
The tree graph clearly shows the partitions. The end nodes with the highest probability of belonging to the "good" class are:

10) capacity>=155.0825 13 0 good (1.00 0.00) *

14) Assets>=25650 8 0 Good (1.00 0.00) *

We can see them circled in blue indicating n/loss, i.e. number of individuals in the group/individuals misclassified.

Below we can obtain the error reduction table, which tells us the following:



-CP: complexity parameter that measures how much overfitting of the model is penalized. This value decreases from 0.23 to 0.01 (lower x-axis on the chart).

-nsplit: Number of splits in the tree. In this case it was 14 divisions (top x-axis on the chart).

-REL Error: Error in the training set after performing a split. This value starts from 1 and as the tree splits it drops to 0.28.

-xerror: This is the cross-validation error of the model. Represents the error in a validation dataset. This also starts from 1 and ends up decreasing exponentially to 0.41 (y-axis on the graph).

-xstd: The standard deviation of the crossover error. It also decreases, but more timidly, from 0.06 to 0.04 approximately.

Finally, we get the confusion matrix, the proportion of true positives, and the acuracity:

	Well	Bad boy
Well	3539	25
Bad boy	54	227

This classifier might be preferable to the previous one since the error of classifying a bad one as good has decreased somewhat, from 67 to 54. This is also reflected in the proportion of correctly classified as bad, which goes from 0.07308192 to 0.8078292.

Well	Bad boy
0.9929854	0.8078292

The overall acuacuity is summarized at 97.95%.

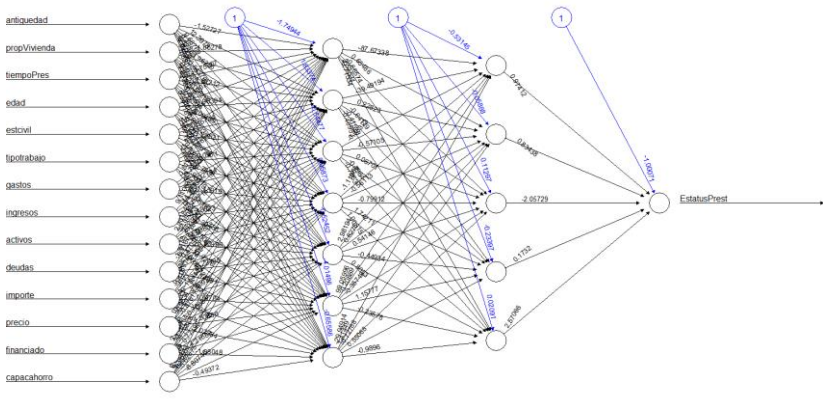
Now, comparing these results with the previous classifiers, the Logistic Regression had an accuracy of 99.01% and LDA of 97.71%. Comparing this to that of RPart, 97.95%, Logistics seems to have a slightly higher performance, and LDA very slightly lower. In addition, as we have seen, LDA avoids classification errors from "bad" to "good" with worse efficiency, which we want to minimize. That said, when it comes to Logistics, Rpart is in the shadows, but the accuracy between LDA and RPart is so similar that even though the latter is somewhat lower, we still prefer it as it minimizes the aforementioned error.

NEURAL NETWORK CLASSIFIER ANALYSIS IN RSTUDIO

For the preprocessing of the data, some variables had to be converted to numeric and the NA had been replaced by the medians of their variables to avoid future problems in the model.

Then, we normalize the variables and divide the data into the training set ('train'), which is made up of 70% of the individuals, and the test set, made up of the rest of the individuals.

Once the training and test data have been defined, we create a formula to simplify the work and not always have to type the name of the variables. We study its relationship with an improved propagation algorithm neural network model and, in this case, with two hidden layers of 7 and 5 neurons where iterations will stop when the error change is less than 5% between one iteration and another. Providing the summary() of the model is very difficult for him, he goes very slowly with the few observations that there are compared to those that could be. The model tree looks like this:



After the model is created, the prediction is made, the output is values that represent the output values of the model for observations in the test set. Each number indicates the probability that the observation belongs to the "good" class according to the neural network. For example, in the first row, we get this output:

8 9.823324E-01

Thus, the probability of being "good" is approximately 0.98 (98.23%). And so on.

Once the predictions have been made, we denormalize them together with the actual data by applying a scale to the data. In this way, we can compare the actual data and the predicted data in their original form and thus obtain the error from the difference. The most important features of the error summary are as follows:

Min	Mean	Max
-1.016936	-0.002478	0.996167

If the minimum is about -1, it means that there has been at least one prediction that has been underestimated. The average error is approx -0.002, which indicates that on average the predictions have been lower by 0.002 points, which tends to be underestimated. The maximum is almost 1 point as well, so the values have been practically overestimated by the same amount as they have been underestimated, with a very slight difference (0.002 points as we see in the average).

To analyze the ranking, we first create a function that classifies individuals as 0 and 1 based on whether their predictions are closer to these values. When applying the function, we find a list of 0s and 1s that indicate the group to which each individual belongs. If we add up how many 0s there are, or how many individuals go to the "good" group, we get 1069, and if we calculate those in the other group we get 85 individuals. To see which amount was misclassified and which wasn't, we can analyze the confusion matrix:

	Well	Bad boy
Well	1069	6
Bad boy	6	79

This classifier considerably reduces the two errors, but acuacuity would have to be studied since individuals are technically fewer by not using a binarized database.

Well	Bad boy
0.9943873	0.9294118

In the classification of the good it has a similar efficiency to the previous ones but in the classification of the bad ones it works much better, it goes from 0.8078292 to 0.9294118.

The overall acuacuity is summarized at 98.96%.

Comparing with the rest of the interesting classifiers, Logistics obtained an acurate of 99.01%, LDA of 97.71% and RPart of 97.95%, it is clear that despite the fact that Logistics continues to have the best efficiency as a classifier, the neural network is second in the ranking, since it not only provides a fairly solid acuacuity but also, It provides us with other metrics such as error and has a high efficiency in the accuracy of the "bad" group that interests us so much. But we have to keep in mind that it had a lower performance in terms of speed of model creation.

CONCLUSIONS AND RECOMMENDATIONS

In conclusion, the classifier we would choose for this specific study would be Logistic Regression. Here are the main reasons:

1. Interpretability: Logistic regression is a highly interpretable model. You can easily understand and communicate how each variable affects the likelihood of belonging to one group or another.

2. Effective for Binary Classification Problems: It's a great option when you're working on problems that support or require binary classification, as is this case.

3. It does not require assumptions about the distribution of the data: Unlike others such as Naïve Bayes, who assume a particular distribution for the data, logistic regression does not require such assumptions.

4. Efficient and fast: Logistic regression is highly and significantly efficient as we have seen throughout the problem and does not fail to work well with large data sets.

5. Availability of different metrics: Unlike other classifiers with which it is difficult to obtain many metrics and quickly, with logistics and using the WEKA Experimenter it is very simple.

6. Large reduction of second error: it is the best classifier when it comes to avoiding classification errors from "bad" to "good".

However, it doesn't have to be perfect, they all have some disadvantages, some of them are the following:

1. Assumption of linearity: Although it does not assume other forms, it does assume a linear relationship between the independent variables and the probability of the outcome. It is not suitable for highly nonlinear or complex problems.

2. Can be sensitive to outliers: Outliers in the data can negatively affect the performance of the model and would lead to even longer preprocessing.

3. It does not handle highly correlated variables well: The high correlation between variables can distort the results, but in reality the order of probabilities would not cease to be the same and would be little affected.

5. You need a sufficient amount of data: To avoid overfitting, a sufficient amount of data is needed to train the model effectively, which can be very expensive in real life, but not so relevant in our case since the company's data history for a few years would be enough.

In summary, logistic regression is a useful and effective model, especially in binary classification problems with data that follow a linear relationship. However, in situations where the relationship between variables and outcome is highly non-linear, other models may be more appropriate.

If any of these were to present themselves as a big problem in the future, the other classifier we would recommend is the neural network, as it can better deal with complex, non-linear data. In addition, they can automatically learn from the data, but for that very reason they also require a significant amount of data and training time, so they would be a solid choice in situations where the complexity of the problem justifies these resources.

BIBLIOGRAPHY

Raj, A. (2021, December 16). Perfect recipe for classification using logistic regression. *Medium*. https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592

Leonardosrocco, V. a. P. B. (2012, June 19). *Classification trees, J48 algorithm*. Data, Mining, BI & Social Stuff. https://leonardosrocco.wordpress.com/2012/06/18/arboles-de-clasificacion-el-

caso-del-algoritmo-j48-un-metodo-inexplorado-de-clasificacion-prediccion-y-generador-de-hipotesis-en-el-campo-de-la-sociologia/

Gonzalez, L. (2022). Logistic regression – theory. 🤖 *Learn AI*. https://aprendeia.com/algoritmo-regresion-logistica-machine-learning-teoria/

Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (n.d.). https://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html

One hot encoding | Interactive Chaos. (n.d.). https://interactivechaos.com/es/manual/tutorial-de-machine-learning/one-hot-encoding