Ciências ULisboa Faculdade de Ciências da Universidade de Lisboa

University of Lisbon

Faculty of Sciences

Department of Informatics

PROJECT: *Assessing Classifiers*

André Filipe Bernardes Oliveira, 45648 – MI

Tânia Maldonado, 44745 - MBCC

Report of

Aprendizagem Automática

For the lecturer: João Marques da Silva

MBBC – MSc in Bioinformatics and Computational Biology

MI – MSc in Informatics

2016/2017

Index

Introduction

Our project targets the prediction of credit card default. The aim of our work was to predict whether a given client will or not default payment based on a client features and their usage of credit cards.

Machine Learning and Data mining techniques can be used to get those predictions. We were asked to implement and assess three classifiers for this problem. The classifiers we used, were support vector machines (SVMs), multilayer perceptron's (MLPs) and decision trees. We implemented these classifiers using R programming language. We decided to use R packages but we also present an attempt to implement MLP.

To train and test our classifiers we used Taiwan credit card default dataset (Default of credit card clients Data Set, 2016). It should be noted that before we used our classifiers we've processed the dataset. First, it was converted to a .csv file (so that it could be handled more easily by R). Then the first row of the file (variable labels) as deleted as it was unnecessary; we also rename one column so that we would get consistent labelling (PAY_0 was changed to PAY_1, because it appears to be a typo) (Folsom, 2016).

Problem statement

In the last decades, many countries, such as Taiwan, have been confronted with the financial crisis. This problem has been amplified by the bank's inadvertent hand over of credit cards to ineligible candidates and the over-issued cash. Furthermore, the neglected and excessive use of credit cards by the cardholders contributed even more for the accumulation of cash and credit cards debt (Yeh & Lien, 2009).

Risk prediction is an important tool for developed financial systems because it allows the system to determine how likely applicants (to credit cards) are to default with their repayments. Besides assessing the business performance or individual customers' credit risk, this tool is used to reduce the damage and the uncertainty associated with credit granting decision.

There are many risk prediction models based on many sophisticated statistical methods, such as discriminant analysis (DA) and logistic regression (LR) (Hand & Henley, 1997), however the recent progress of artificial intelligence and machine learning enabled the employment of expert systems (automated or semi-automated procedures capable of learning) such as artificial neural networks (ANN), like Multilayer perceptron

(MLP), and classification trees (CT) (Thomas, 2000).

From the perspective of risk control, estimating the probability of default will be more meaningful than classifying customers into risky or non-risky. Which means that the accuracy of the result is more important than the result itself (and that is a key aspect in data mining that contrasts with machine learning). Therefore, whether or not the estimated probability of default produced from data mining methods can represent the ‘‘real” probability of default is an important problem that we tried to cover in this project.

Classifiers used

Data mining techniques are used to discover patterns and rules in data (Yeh & Lien, 2009). This can be helpful to which customers grant credit cards, and which not.

According to Yeh & Lien, 2009 findings state that the two data mining methods with best performance are **artificial neural networks** (with low error rate that is similar to others but with greater accuracy), followed by **classification trees** (with one of the lowest error rate: 0.17 on validation).

Artificial neural networks use a neuron-like schema, computing non-linear equations to establish relationships between input and output variables. There are several implementations of multi-layer perceptron’s, the most widely known is back propagation. Back propagation neural network uses a feed-forward topology and supervised learning. The network consists of input layer and output layer and can have hidden layers and each of these layers is composed of several units called neurons. One of the limitations of these technique is that we cannot obtain simple probabilistic formula of classification (Yeh & Lien, 2009) (Russell & Norvig, 2009).

According to Thomas (2000), the classification performed by **Classification Trees (CT)** consists in the homogeneous distribution (in what concerns the default risk) of the applicants into groups of different default risks. This classifier can be applied when using qualitative or quantitative discrete response variables. CTs classificate the observations regarding all the explanatory variables and directed by the presence of the response variable. As previously stated the lower error rate associated with this classifier was the reason why we chose to use it. Yet one must consider that a small change might alter the structure of the tree (Yeh & Lien, 2009).

Support Vector Machine (or SVM) is a non-probabilistic binary linear classifier based on learning algorithms capable of analyzing data. Furthermore, SVM can perform a non-linear classification by resorting to the kernel trick. The study performed by Baesens et al. (2003) found that both the SVM and neural network classifiers yield a very good performance. For this reason, we will also implement this classifier.

Experimental results

In this section, we present the results for the 3 classifiers we selected. The data was divided in two sets: one for training and one for testing the model. Since the relative abundance of defaulters is much lower the rest, the evaluation of the model should consider more than the error rate.

Therefore, we decided to analyze false positives and false negatives because they can be more informative. This information was computed in form a confusion matrix and f-measure.

Experimenting with the R packages options for the classifiers, we try to adjust them by trial and error. Some attempts developing a scrip to try several configurations proved to be a not helpful. The best example was finding the best hidden layer configuration: as the search space is big (even considering the best options), and training with neuralnet took quite a while to run, it took too much time just to get a few results.

It should be noted that in all classifiers we excluded the column “ID” as it would be irrelevant for classification. Despite that, all the 23 features were used, no feature selection was performed.

For the **neural network** classifier, our code runs with a percentage of error of approximately 22%. In one of the best tries (using 10 nodes in one hidden layer, and other hidden layers with less nodes; for example the following configuration for hidden layers: 10, 6, 2), we obtained percent errors: 22.120000; precision: 0.5; recall: 0.000904159132; f-measure: 0.001805054152; which is not a good result. Moreover contrary to what was expected (Yeh & Lien, 2009) this model performed worse than the others (analyzing f-measure) while it should be the one with best performance. This might be due to this model be more affected by difference in relative abundance of defaulters and non-defaulters. We can assume that because it was the only case where we had to do a sampling that was not completely random (we had to force defaults to be present in the train sample or otherwise the neural network would consider everything as non-default).

The **neural network** classifier yields a plot like the following. It is important to note that this network does not contain the hidden layer configuration that we found to give better results; because we were unable to fit the plot on screen. Despite that, the best results were obtained with a first hidden layer of 10 nodes.

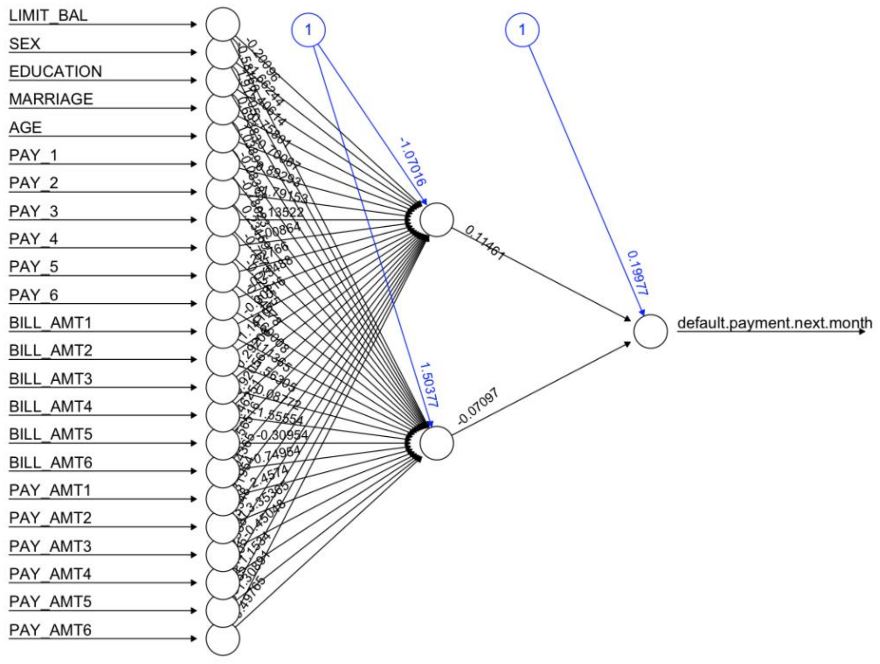

Fig. 1 - Neural Network Classifier Plot

The black lines show the connections between each layer and the weights on each connection, while the blue lines show the bias term added in each step. The bias can be thought as the intercept of a linear model. The net is essentially a black box so we cannot say that much about the fitting, the weights and the model.

The main main of multi layer perceptron algorithm is the following:

$$neuralnet(f, data = train, hidden = c(10,6,2),\ act.fct = \text{"}logistic\text{"}, linear.output = FALSE, threshold = 0.1)$$

where we highlight the fields ***hidden* = (**$\mathbf{10, 6, 2}$***)*** (the number of hidden neurons in each layer – using 3 hidden layers), ***act.fct = "logistic"*** (a function used for smoothing the result of the neurons and the weights), and ***linear.output = FALSE*** (if act.fct should not be applied to the output neurons set linear output).

For the **decision tree classifier**, our code is approached with a percentage of error of approximately 17%, which is consistent with what others have found (Yeh & Lien, 2009). Executing our classifier yields a percent errors: 17.270000; precision: 0.6697478992; recall: 0.3740028156; f-measure: 0.4799759109, which is a much better result than the one provided by neural networks. The resulting tree can be seen below:

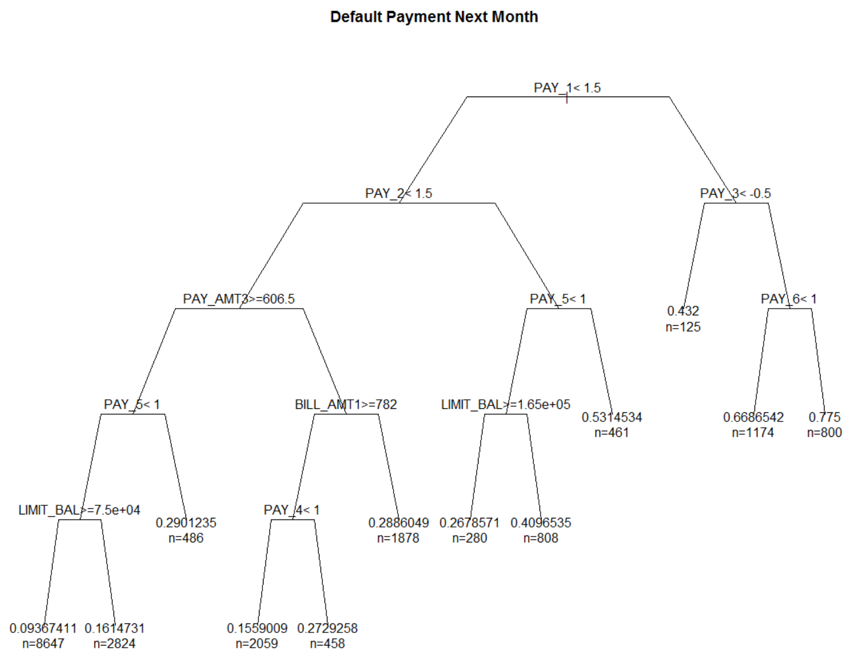

Fig. 2 - Decision Tree Classifier Plot

At each node of the tree, we check the value of one the input and depending of the (binary) answer, we continue to the left or to the right sub branch. When we reach a leaf, we find the prediction.

We used following snippet of code for training of the decision trees:

$$rpart(f, data = train, method = "anova", control = rpart.control(minsplit = 2, minbucket = 1, cp = 0.001))$$

where we highlight the fields ***minsplit* = *2*** (the minimum number of observations that must exist in a node in order for a split to be attempted), ***minbucket* = *1*** (the minimum number of observations in any terminal leaf node) and ***cp* = *0.001*** (complexity parameter; any split that does not decrease the overall lack of fit by a factor of cp is not attempted). Lastly, the ***method* = *"anova"*** is chosen by exclusion of parts, where the two available options were "class" and "anova", and anova being the one with better (fewer) error rates (18% vs 22% with the others).

For the **support vector machine classifier**, we get results of approximately 19% of error. The results from the svm function contained in R package we choose needed to be processed because it outputs the classification as 1 or 2; while it should output 0 or 1. In one example of classification we got percent errors: 19.080000, precision: 0.6277573529; recall: 0.3124428179; f-measure: 0.4172266341. These results showed that support vector machines performed worse than decision trees but better than artificial neural networks.

For this classifier no plot was provided. The library function *svm* integrated in our classifier was tuned with the following parameters:

$$svm(x, y, type = "C - classification", kernel = "polynomial", cost = 10)$$

where we highlight the fields ***type = "C-classification"*** ***kernel = "polynomial"***, and ***cost = 10.*** This means, we used SVM on classification mode (and not regression), it also used a Polynomial kernel that allows learning non-linear models (as we assume this one is; as no learning as possible with linear kernel).

To wrap up we can conclude that all classifiers yield a similar error rate but the accuracy of the models is quite different (as seen by f-measure). This is corroborate by the other's findings, including our main reference authors (Yeh & Lien, 2009).

Source code organization

To assure that the project as functioning, we decided to use R packages to implement the chosen machine learning techniques: support vector machines (e1071), multilayer perceptron's (neuralnet) and decision trees (rpart).

We tried to implement MLP using strategy provided by Russell & Norvig, 2009, but we failed to do so. It consisted of implementing backpropation algorithm running until a given number of epoches is reached. We also tried searching for similar solutions, but we couldn't finish, so we present what we have.

All source code is available at the "Code" folder, including our last atempt at implementing the MLP. All script use the dataset in .csv format contained in the "Dataset" folder, which is the file we modified as stated in the introduction.

Conclusions

In this project we created 3 classifiers that can be used to predict whether a client will default or not. These classifiers yielded results similar with error rate, but quite different accuracy. We do not confirm the Yeh & Lien (2009) findings in which we expected artificial neural networks to perform better that they did (and with the best results overall). Despite that we think that the results obtained with the other classifiers were reasonable (f-measure was not near 1). Given our results we can conclude that our best classifier was decision trees, followed by support vector machines and in the end multi-layer perceptron.

Regarding our own implementation of multi-layer perceptron, we failed to complete it but present a draft included in the source code.

Bibliography

Alice, M. (2015, 09 23). *Fitting a Neural Network in R; neuralnet package*. Retrieved 12 17, 2016, from DataScience+: http://datascienceplus.com/fitting-neural-network-in-r/

Baesens, B. V. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society, 54(6)*, 627–6.

Default of credit card clients Data Set. (2016, 1 26). Retrieved 12 11, 2016, from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Folsom, K. (2016, 03 12). *DATA 607: Project 2*. Retrieved 12 15, 2016, from RPubs: https://rpubs.com/kfolsom98/160773

Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society, Series A – Statistics in Society, 160(3)*, 523–541.

Neto, J. (2013, 05). *Classification & Regression Trees*. Retrieved 12 16, 2016, from http://www.di.fc.ul.pt/~jpn/r/tree/tree.html#classification-trees

Russell, S. J., & Norvig, P. (2009). *Artificial intelligence: a modern approach* (3rd ed.). Prentice Hall.

Thomas, L. C. (2000). A survey of credit and behavioral scoring: Forecasting financial risk of lending to consumers. *International Journal of Forecasting, 16*, 149–172.

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*.