Credit Scoring in R

The basic difference of traditional modeling and machine learning is that, in traditional modeling we intend to set up a modeling framework and try to establish relationships while in machine learning we allow the model to learn from the data by understanding the hidden patterns. Hence the first one requires analyst to have solid understanding of statistical techniques and business knowledge while the later one is more complex in nature and computational intensive, hence requires higher computation power of the systems and analyst needs to be tech savvy.

Kindly note that while traditional techniques perform well on small to large amount of data, machine learning will certainly learn better on high-dimensional and complex data such as Big Data set up.

Aim:

To develop a self-learning model to assess the credit-worthiness of future customers on the basis of historical data

Business Requirement:

To replace a rule based model in favor of a model which is intelligent enough to take into account the past payment behavior of customers to assess the credit worthiness of future applicants.

Dataset:

Dataset consists of 58 dependent variables and 302 rows.

Model:

Binary classification model with levels of **serious defaulter** - 1 and **not a serious defaulter** – 0, Null Hypothesis being that the customer will not default in the future with the alternative hypothesis being that the customer will default.

Salient Features of the model are:

1. Individual and ensemble modeling
2. Discerning the best model between the individual and ensemble ones
3. Selecting the optimum cutoff score to separate the potential defaulters from the non-defaulters
4. Cross Validation to ensure maximum accuracy of prediction

Code:

1. **Loading required packages**

Load the various packages for data visualization, data manipulation and machine learning

2. **Reading the data**

MySQL connection with an in built query to load the data set for analysis

3. **Splitting the data into train and test sets**

Currently using a 85:15 split due to a low volume of data

4. **Resampling of data**

This is done in order to balance out imbalanced data sets with one output class dominating the other. Code checks for whether one class outweighs the other by 10% or greater in number then it performs a Resampling in order to balance the data set

5. **Train Control**

Function which guides the model training process. 10 fold repeated cross validation is used as the argument passed under "method"

6. **Training the model**

Five machine learning algorithms are used for training, namely:

1. Random Forests
2. Gradient Boosting
3. Neural Networks
4. C5.0 decision trees
5. Logistic Regression

Feature and Model Selection:

Initially all variables are used to train each algorithm. The algorithms then on an individual level, score each variable based on their influence on prediction. We can then select the best combination of variables of each algorithm by verifying that the combination ensures maximum performance.

Code selects the variables for each algorithm in such a way so as to maximize accuracy on the test set (accuracy metrics will be explained later in this document).

Each variable in each model is given a score out of 100 based on their importance

After the variable selection process is complete we have a set of 5 models with their optimum number of variables and a certain maximum value of the performance metrics attained by each of them.

We then narrow down on the top n models which clear the performance criteria

For e.g.

Model Name	Optimum Number of Variables	Maximum Accuracy	Cutoff (Probability of default)
RF	14	80%	0.35
NN	16	75%	0.40
GBM	19	77%	0.50
C5.0	21	88%	0.31
GLM	18	61%	0.23

We can see in the above table the RF reached a maximum accuracy of 80% with an optimum of 14 variables, C5.0 reached a maximum accuracy of 88% with an optimum of 21 variables etc.

We choose a cutoff of 70% (to say) and thus we are left with RF and C5.0 as our final models.

We also check for the optimum cutoff value (which minimizes prediction error) simultaneously a

7. **Ensemble modeling**

Once we have narrowed down on our final models as described in step 6, we go forth with the option of combining them in order to check if this yields better results than the individual models.

Currently the ensemble model used is taking a simple average of the probability of default given by the individual final algorithms. In the future, more methods such as weighted average and algorithm stacking may be used to enhance the results.

Once the ensemble model has been formed, the code then computes the optimum cutoff value for the ensemble as well based on the same performance metrics used in step 6.

8. **Checking if the Ensemble is better than individual models**

Then we check if the performance of the ensemble exceeds all of the individual models. If yes, then we go ahead with the ensemble otherwise we go ahead with the best of all the individual models.

9. **Predicting on new data**

We then train the model on the entire data set and then use the model to predict on the new data point (new CRASS request).