****

**INDUSTRIAL AND MANAGEMENT ENGINEERING DEPT,**

**IIT KANPUR**

IME672 - DATA MINING AND KNOWLEDGE DISCOVERY

INSTRUCTOR: Dr. Faiz Hamid

**PROJECT REPORT**

**PROBLEM: Automotive Driver Insurance Claim Prediction**

**SUBMITTED BY:**

1. **Nalla Ramu : 14407**
2. **Gaurav Kumar : 14240**
3. **Naveed : 14458**

**Date of Submission: 19 April 2018**

**Table of Contents**

**1 . Problem Description………………………………………………….…………………………………………..2**

**2. Data Preprocessing……………………………………………………………..…………………………………2**

1. **Data…………………………………………………………………………………..……………………....…..2**
2. **Variables……..……………………………………………………………………..……………………….....2**
3. **Uniqueness in the data…………………………………………...............……………………....….2**
4. **Changing the classes of the attributes……………………………………..…….............…….2**
5. **Removing ‘id’ from the data…………………………………….……………………………….……..2**
6. **Missing values…………………………………………………………………………………………...…...3**
7. **Merging Variable classes…………………………………………………………………….……....….4**
8. **Dimensional Reduction………………………………………………………………………..………... 5**
9. **Numerosity Reduction……………………………………………………………………………….…….5**

**3. Training the Model…………………………………………………………………….…………………………..6**

**4. Conclusion………………………………………………………………………………………………………………9**

1. **PROBLEM DESCRIPTION:**

**Porto Seguro** is the third largest insurance company in Brazil, wants to predict the the probability that a driver will initiate an auto insurance claim in the coming year. There are three data files that are given by the Kaggle.

1. Train Dataset: Contains the training data
2. Test Dataset: Contains the test data
3. Sample: is submission file showing the correct format

Training data contains an attribute labelled as “Target” which signifies that the claim was filed. If the target attribute is equal to 1, it indicates that the corresponding driver has claimed the insurance and if it is 0, then he is not.

**2. DATA PREPROCESSING:**

1. **Data :**

There are 59 different attributes and 595212 tuples present in the training dataset.

* Number of Rows = 595212
* Number of Columns = 59

**b)** **Variable names:**

Columns other than “id” and “target” have no meaning at all. But the last three letters of the name of the attribute name tells us the type of the attribute it is. If the attribute name ends with “cat”, it is a categorical type attribute and if it ends with “bin”, it is binary. The rest are numerical attributes. The attribute distribution is as follows: - Numeric attributes : 26

* Categorical attributes : 31
* Other : 02 (‘id’ and ‘target’)

**c) Uniqueness in the columns of the data**

Uniqueness of the data in columns was checked. And it is found no attribute has unique class.

**d)** **Changing the Classes of the attributes**

The entire data present in the training dataset are numbers. RStudio by default considering every column as a numeric type in the dataset. In order to restore the type of the data from the attribute name, a piece if R code was implemented to change the data types present in the training dataset.

**e) Removing “id” from the data**

This attribute “id” has no effect on the driver claiming the insurance. Therefore ‘id’ is removed.

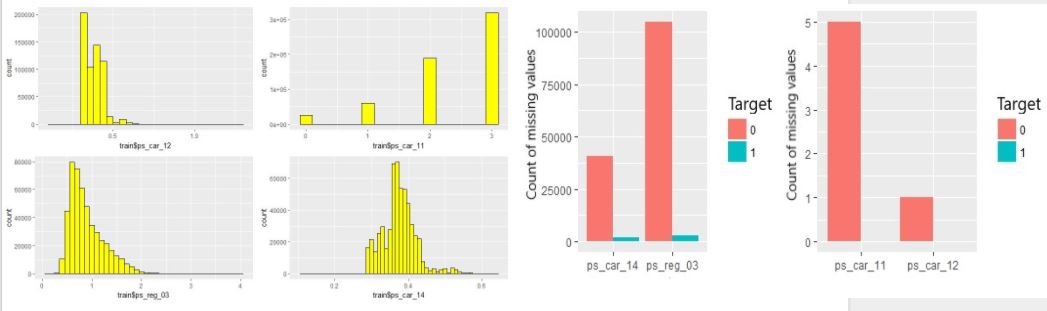
**f) Missing Values**

In the dataset the elements which have the value of ‘-1’ is to be considered as a missing value. First the values of ‘-1’ are changed to ‘NA’. Below shown the summary of the missing values present.

|  |  |
| --- | --- |
| **NUMERIC ATTRIBUTE MISSING VALUE COUNT**  ‘ps\_reg\_03’ has 107772 missing values  ‘ps\_car\_11’ has 5 missing values  ‘ps\_car\_12’ has 1 missing value  ‘ps\_car\_14’ has 42620 missing values | **CATEGORICAL ATTRIBUTE MISSING VALUE COUNT**  ‘ps\_ind\_02\_cat’ has 216 missing values  ‘ps\_ind\_04\_cat‘ has 83 missing values  ‘ps\_ind\_05\_cat‘ has 5809 missing values  ‘ps\_car\_01\_cat’ has 107 missing values  ‘ps\_car\_02\_cat’ has 5 missing values  ‘ps\_car\_03\_cat’ has 411231 missing values  ‘ps\_car\_05\_cat’ has 266551 missing values  ‘ps\_car\_07\_cat’ has 11489 missing values  ‘ps\_car\_09\_cat’ has 569 missing values |

The total number of tuples are 595212. It can be seen from the above missing value summary that 69% of the tuples has ‘ps\_car\_03\_cat’ attribute missing and 45% has ‘ps\_car\_05\_cat’ attribute missing.

**- Replacing Numeric attribute missing values:** Missing values can be replaced by mean, median, class mean, class median and so on. The distribution of the four numeric attributes by omitting the missing values are shown below in Fig.1.

Fig 1. Distribution of Attributes with numeric missing Fig 2. Numeric missing data with respect to target value

The data distribution in Fig.1 shows that the attributes are skewed. The analysis of the numeric missing data with the target value of the tuple shown in Fig.2. The histograms in Fig.2 shows that most of the missing values belong to the tuples which has target value as ‘0’. Also Fig.1 shows the data is skewed. Therefore, the missing values are replaced by the class median.

- **Replacing Categorical attribute missing values:** There are nine categorical attributes which has missing data. As a significant amount of the data is shared by these missing values, replacement by mode can alter the distribution of the data.

‘ps\_car\_03\_cat’ attribute has 69% of the tuples missing and ‘ps\_car\_05\_cat’ has 45% of the tuples missing. Replacement of the missing values in these attributes can disturb the variance of the data. Therefore, these two attributes are removed from the dataset.

**PMM:** The Primitive mean/mode matching (pmm) method is used for replacement of the categorical missing data of the other seven attributes. In this method a model is trained to impute the missing values using attributes with no missing data.

It fits several linear regression models and try to predict the values for the entire attribute of the missing data. Then the model compares the predicted values of missing values with that of the actual value and replaces the missing data with the mode of the compared actual data. After this step no missing data is present in the dataset.

**g) Merging Variable Classes**

There are some classes of the attributes in the training data which capture less than 1% of the entire data. Some of these variables and there classes are shown below in Fig.3.

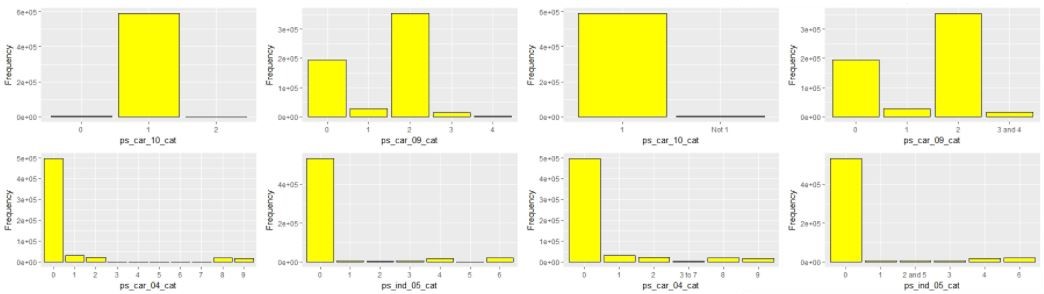


Fig 3. Variable distribution with their classes Fig 4. Variable distribution with their classes after merging

Table 1. Attributes with old classes and the merged new classes

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Old Classes** | **New Classes** |
| ps\_car\_10\_cat | 0,1,2 | 1 , Not 1 |
| ps\_car\_09\_cat | 0,1,2,3,4 | 0,1,2,3 and 4 |
| ps\_car\_04\_cat | 0,1,2,3,4,5,6,7,8,9 | 0,1,2,3 to 7, 8,9 |
| ps\_ind\_05\_cat | 0,1,2,3,4,5,6 | 0,1,2 and 5, 3,4,6 |

To reduce the computation time and load, some of these classes can be merged. It is ensured that the classes in attributes should capture at least 1% of the data. The details of the old and new classes of the four attributes are shown below in Table.1. The new classes distribution for the above variables is shown below in Fig.4.

**h) Dimensional Reduction**

There are now 56 attributes present in the data. If any two attributes are dependent, one attribute can be formed from the other attribute. Therefore one attribute can be reduced to reduce the computation load in training the model on the dataset. The Dimensional reduction is approached in different ways for numeric and categorical attributes

**- Numeric Dimensional Reduction (Correlation Analysis)**

Pearson’s product moment coefficient is calculated among the numeric attributes. The Correlation results is shown pictorially below in Fig.5:

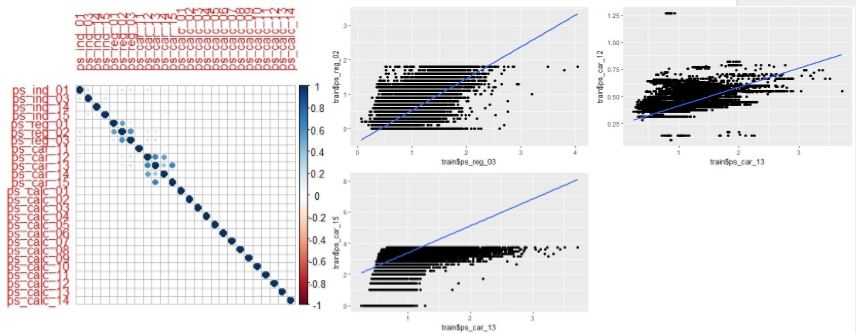


Fig 5. Pearson’s product coefficient between attributes Fig 6. Scatter plots between the correlated attributes

In Fig 5. There are some pairs of attributes which have noticeable Pearson’s product moment values. These pairs with their exact correlation values are shown in Table.2. Based on Table.2 correlation test results, three attributes can be removed if the threshold correlation value is kept as 0.6. The scatter plots between the correlated attributes are shown below in Fig.6.

Table 2. Pearson’s product coefficient of noticeable attribute pairs

|  |  |  |
| --- | --- | --- |
| **Attribute pairs** | | **Pearson’s coefficient value** |
| ps\_reg\_02 | ps\_reg\_03 | 0.6007 |
| ps\_car\_12 | ps\_car\_13 | 0.6633 |
| ps\_car\_13 | ps\_car\_15 | 0.6766 |

After Numeric dimension reduction the number of variables are 53.

**i) Numerosity Reduction (Outlier Detection)**

Numerosity Reduction is approached by two ways. 1. Cook’s distance computation

2. K-means Clustering

**- Cook’s Distance Computation:** Cook’s distance is useful for identifying outliers in the data objects. First it fits a regression model with all the data objects. Then it computes the influence of each data object by eliminating it and generating a new regression model. If the cook’s distance for a data object is large, the regression model coefficients changes by a large amount when the corresponding data object is removed and it is possibly an outlier.

The influence of the data object is computed by measuring the change in the regression model. Figure.7 shows the cook’s distance for all the data objects in the training dataset. Each point in Fig.7 corresponds to a data object and denoted by its ‘id’.

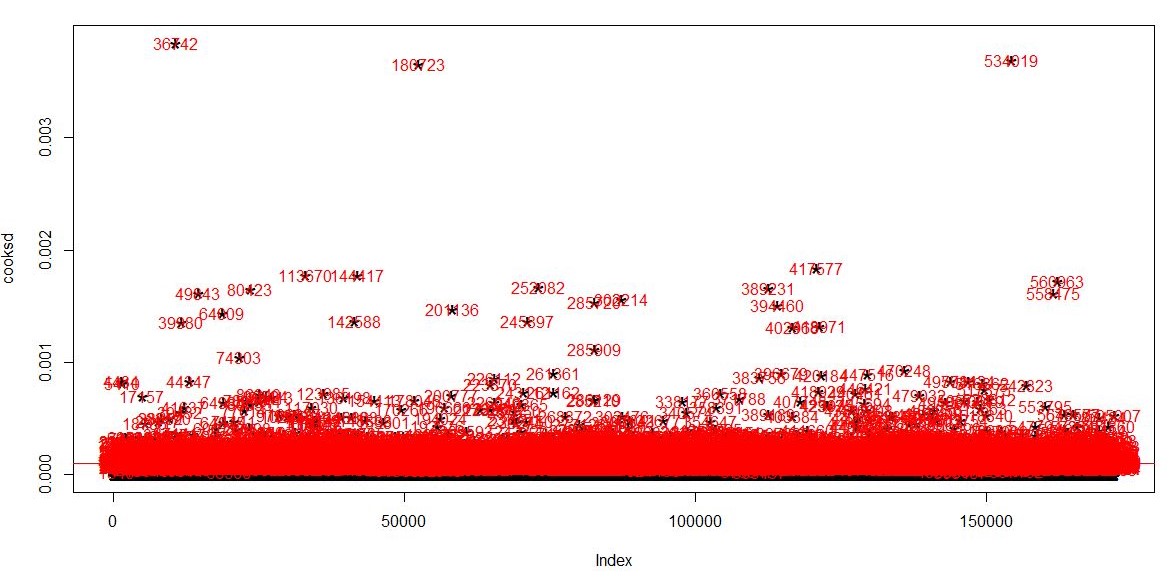


Figure 7. Cook’s Distance for the data objects

**- K-means clustering:** K-means clustering is a type of unsupervised learning used to cluster the data objects. A data object is an outlier if it is far away from the centers of the cluster. Here the number of clusters are decided by the user. Based on the K-means clustering some outliers are detected and those tuples are removed to ensure the better training of the model. After this stage the data is preprocessed and is ready for training a model

**3. TRAINING THE MODELS**

After preprocessing the data, it is evident that the data lacks some structure in it. The results of PCA which failed to reduce the dimensions of the data with a high confidence value. All the components had captured similar amount of variance. So, it is difficult to say at one go that which model would be perfect fit for our data and would result in the maximum accuracy.

So a plethora of techniques and methods are tried to learn a good model and we shall try to elaborate the results obtained for each model one by one. First we take a look at how we handled the data during training.

**-Sampling**

The training data set has 2 labels 0 and 1 which are present in unequal proportions. Label 1 is very less compared to label 0. So if we train the model with this training data, it is very likely that the learnt model would be biased and would fail to learn one case properly. We used the technique called over and under sampling. Class label 0 is under sampled to reduce its quantity and class label 1 is oversampled to increase its numbers.

**Different learning Methods Tried:**

1. **Naive Bayes Method:-**

In this model, every pair of features being classified is independent of each other. This assumption makes this method very naive and hence in complex scenarios it is often used as a baseline model. But again, we need to be careful as in certain scenarios with independent features it prove to be a good classifier. We tried Naive Bayes on 300000 data points.

1. **Logistic Regression:-**

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). A linear regression is not appropriate for predicting the value of a binary variable and hence the logistic regression arises. It uses a sigmoid function before prediction which makes it a non -linear model. Our training data has only 2 categories to predict and hence it can be a starting point to consider logistic regression for this case. We trained our model on the entire raining set

1. **Support Vector Machines:-**

SVM can be used for binary classification and its multi class variant is also available. SVM tries to learn a maximum margin hyperplane to separate the two classes of data. With the help of kernels, it fits linear classifiers in higher dimensions which leads to non- linear classifiers in current dimensions. Its complexity is very high when there are many data points. So, to get SVM train quickly, we decided to keep the number of training examples to 100000 only with each class balanced.

1. **Neural Networks:-**

Neural Networks is a non-linear classifier that takes a normalized attributes as input and consists of many hidden layers. Each hidden layer consists of nodes which are connected to nodes from next layer through weights. It takes a lot of time to train and tune Neural networks. So, we pruned the dataset so that training could be faster.

1. **Random Forest-**

It is a collection of decision trees where each tree is trained from a subset of attributes selected randomly from the superset. It makes sure that if some feature is really good, every tree will use it and then all trees will behave similarly so restrict available features. In our case number of trees was tuned to be 100.

1. **Boosting Algorithms-**

Two different boosting algorithms were tried to see how they perform on the train dataset.

1. Adaboost: It starts with a weak learner and feeds its error to the next model. Correctly classified points get down weighted and misclassified points are upweighted. Its final output is weighted sum taken from all the models learnt.
2. XgBoost: It is a gradient based boosting algorithm which tries to minimize the loss function using gradient descent method. It also introduces a regularization term in the loss function to reduce the problem of overfitting. It has been effective in a number of scenarios.

We tried both these boosting models on our training data.

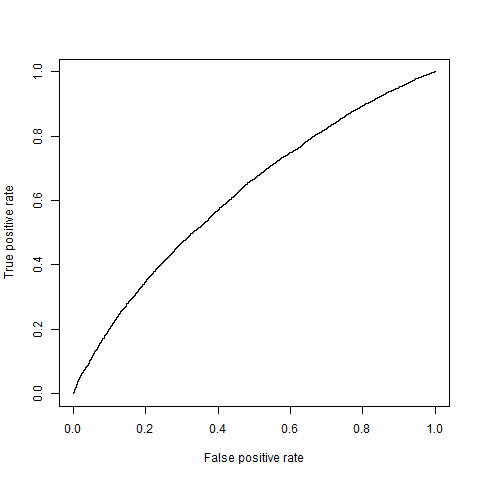
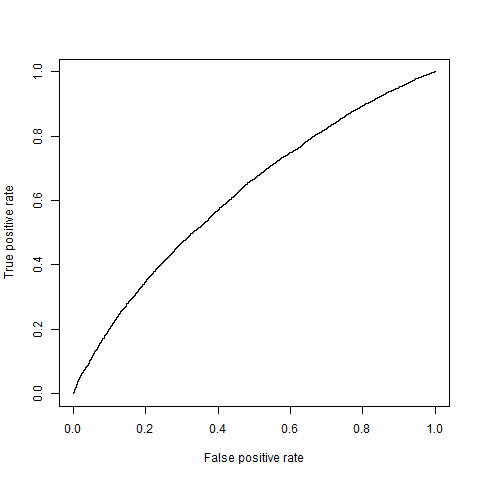
**Table showing Results obtained on all the models**

Table 3. Comparison of different models

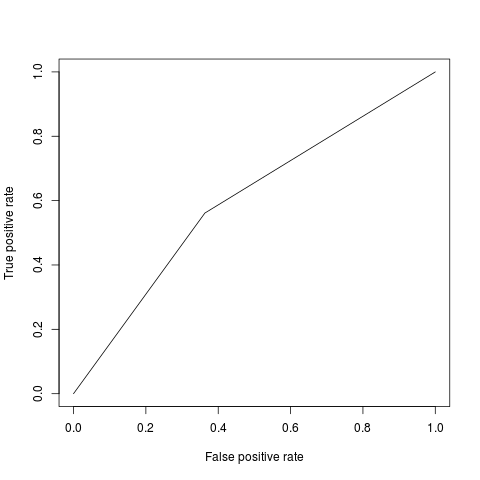
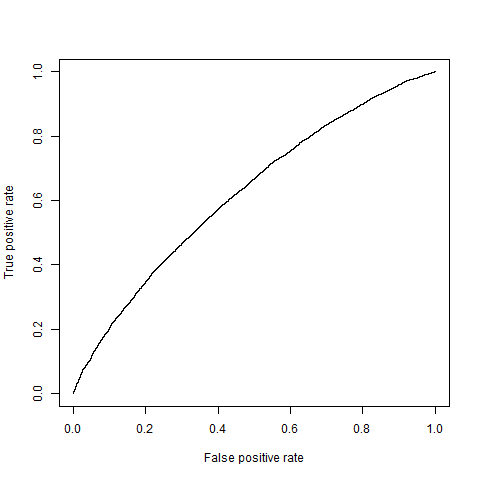
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Parameter | Naive bayes | Logistic Regression | SVM | Neural Network | Random Forest | AdaBoost | XgBoost |
| *Accuracy* | 57.53% | 58.63% | 59.88% | 58.71% | **85.35%** | 55.8% | 68.41% |
| *Sensitivity* | 0.6948 | 0.6270 | 0.6369 | 0.59 | **0.8834** | 0.583 | 0.6927 |
| *Specificity* | 0.4617 | 0.5410 | 0.5614 | 0.58 | **0.8241** | 0.548 | 0.6756 |

Best Model was **Random Forest with accuracy of 85.35%** on test data**.**

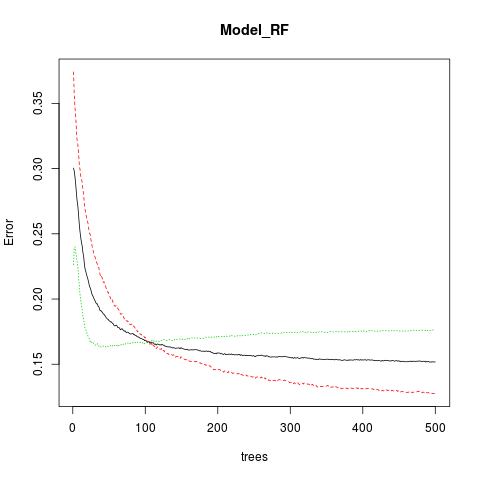
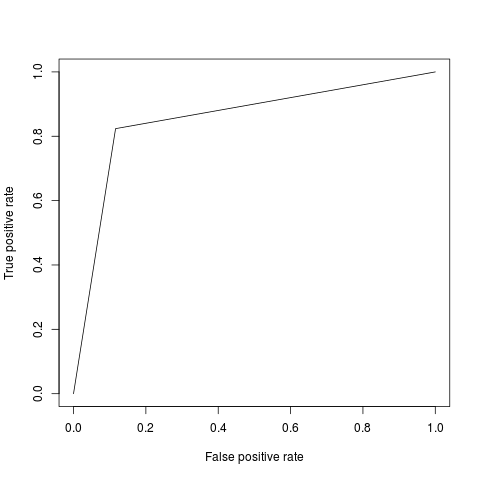
**Plots for different models:**



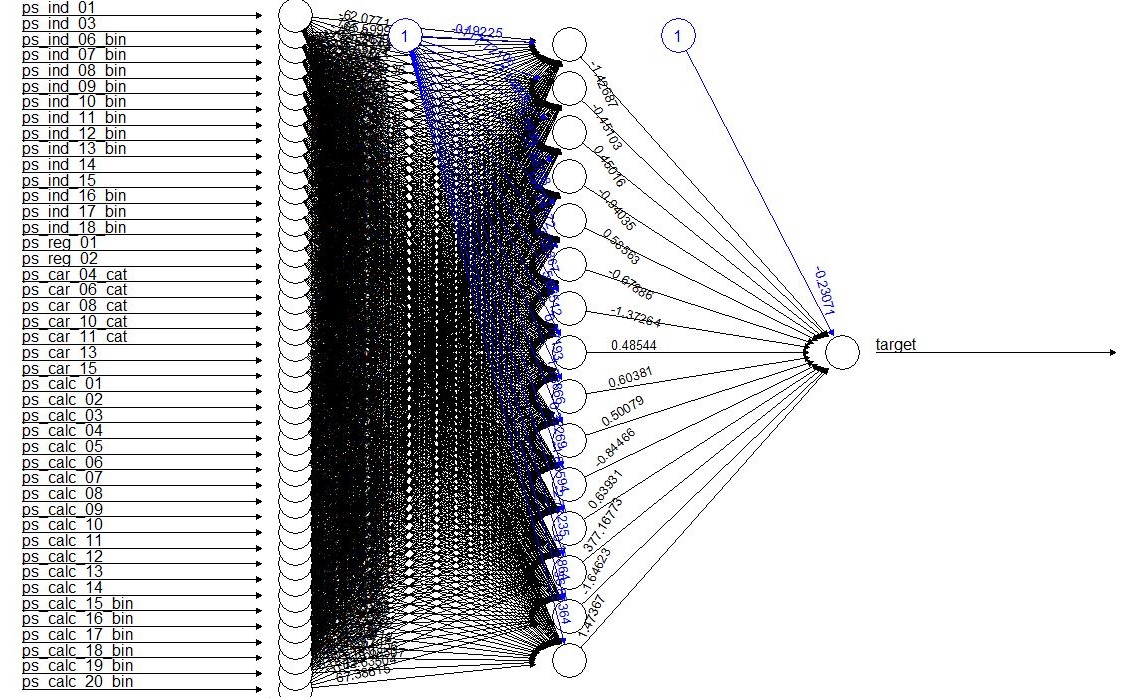
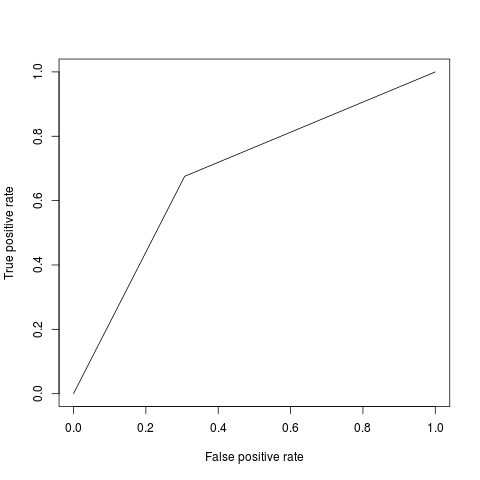
**Fig 8**: ROC Curve for Naive Bayes **Fig 9**: ROC Curve for Logistic regression

****

**Fig 10**: ROC curve for SVM **Fig 11**: ROC curve for Neural Network

****

**Fig 12:** Error vs Trees Plot for Random Forest **Fig 13**: ROC Curve for Random Forest

****

**Fig 14:** ROC curve for XgBOOST  **Fig 15**: Diagram of Neural Network

**Conclusions:**

All the code was written in R language and implemented in **RStudio Desktop 1.1.447.** After training multiple models **85.35% accuracy** was achieved which is not very far from the current state of the art method for this problem. After sufficient modifications and tuning, it is not a difficult task to even achieve 90% accuracy.