Final Report

Contents

Project Objective

The project is about understanding the premium payment pattern of customers of an Insurance company. For the study, we have customer data available primarily covering:

a) Customer demographic information (e.g. Age, Income, Marital Status, residence area type etc.)
b) Insurance policy and premium payment related information (e.g. premium, renewal, sourcing channel etc.)
c) Customer risk profile (risk score)

The objective of this project is to predict the probability that a customer will default the premium payment or not.

Defining Business Problem

Insurance is a form of risk management tool which allows the insured party to hedge the risk of a uncertain loss. Herein the entity offering the protection against the risk is called '**Insurer**' and the customer taking the protection is called '**Insured**'. For purchasing the protection ('**Insurance**') against the uncertain loss from the **Insurer,** the **Insured** has to pay a premium termed as **'Insurance premium'** or simply **'premium'** which is usually periodic in nature.

Insured has the right to claim compensation under the **Insurance policy** till the time the **premiums** are duly paid and the policy is therefore renewed on the likely date of renewal. However, at the discretion of **Insured,** if the **premium** is not paid ever after the due date or in other words the **Insured** default the **premium** payment, the **Insurance policy** gets lapse and any further claim cannot be raised under the **Insurance policy**.

At present, Life Insurance, Health Insurance and General Insurance (Non-Life) are the commonly used risk management tools in Indian market. **Insurers** are governed by autonomous regulatory bodies (**Insurance Regulatory and Development Authority** in India) to protect the interest of the **Insured** and to prevent mis-selling and other unfair practices.

Insurance is an important risk management tool which is relevant to all sections of society to mitigate uncertain losses triggered due to various unforeseen events. Developed countries has higher insurance penetration and developing countries are catching up with increasing awareness and affordable cost of buying insurance.

From a commercial point of view, **premium** paid by the customer is the major revenue source for **Insurer**. Default in **premium** payments results in significant revenue losses and hence **Insurer** would like to know upfront which type of customers would default **premium** payments.

The objective of this project is to predict the probability that a customer will default the **premium** payment, so that the insurance agent can proactively reach out to the policy holder to follow up for the payment of **premium**. Simultaneously, it will also help understand customer demographics which are more likely to default and to price the premium amount in accordance to the same.

Approach

Introduction of Business Problem, Data understanding, Exploratory Data Analysis , Data pre- processing, Model building & comparison and finally insights and recommendation from the best model.

Colour code

For better clarity, we will follow below color coding through out the report:

R Command, R Output

Data Dictionary

The dataset has 79853 records with total 17 different variables. The target or the dependent variable in the given dataset is "renewal", which has values as 0 or 1. "0" indicates that customer has not renewed the premium and "1" indicates that customer has renewed the premium.

Below is the list of variables along with the description and categorization:

Variables	Description	Type
Id	Unique customer ID	Continuous
perc_premium_paid_by_cash_credit	% of the premium paid by cash payments	Continuous
age_in_days	Age of the customer in days	Continuous
Income	Income of the customer	Continuous
Count_3-6_months_late	Number of times premium was paid 3-6 months late	Continuous
Count_6-12_months_late	Number of times premium was paid 6-12 months late	Continuous
Count_more_than_12_months_late	Number of times premium was paid more than 12 months late	Continuous
Marital Status	0 indicates that customer is Unmarried and 1 indicates that customer is Married	Indicator
Veh_owned	Number of vehicles owned (1-3)	Indicator
No_of_dep	Number of dependents in the family on the customer(1-4)	Indicator
Accomodation:	0 indicates that customer has rented the accommodation and 1 indicates that customer has owned the accommodation	Indicator
Risk_score	Risk score of customer	Continuous
no_of_premiums_paid	Number of premiums paid till date	Continuous
sourcing_channel	Channel through which customer was sourced (A/B/C/D/E)	Indicator
residence_area_type	Residence type of the customer (Rural/Urban)	Indicator
premium	Premium amount	Continuous
renewal	0 indicates that customer has not renewed the premium and 1 indicates that customer has renewed the premium	Indicator

Data overview & Exploratory Data Analysis

Let's start with understanding the date first.

```
Classes 'tbl_df', 'tbl' and 'data.frame':                 79853 obs. of     17 variables:
 $ id                               : num   1 2 3 4 5 6 7 8 9 10 ...
 $ perc_premium_paid_by_cash_credit: num    0.317 0 0.015 0 0.888 0.512 0 0.994 0.019 0.018
...
 $ age_in_days                      : num   11330 30309 16069 23733 19360 ...
 $ Income                           : num   90050 156080 145020 187560 103050 ...
 $  Count_3-6_months_late           :  num  0 0  1 0 7 0 0 0 0 0     ...
 $  Count_6-12_months_late          :  num  0 0  0 0 3 0 0 0 0 0     ...
 $  Count_more_than_12_months_late  :  num  0 0  0 0 4 0 0 0 0 0     ...
 $  Marital Status                  :  num  0 1  0 1 0 0 0 0 1 1     ...
 $  Veh_Owned                       :  num  3 3  1 1 2 1 3 3 2 3     ...
 $  No_of_dep                       :  num  3 1  1 1 1 4 4 2 4 3     ...
 $  Accomodation                    :  num  1 1  1 0 0 0 1 0 1 1     ...
 $ risk_score                       : num   98.8 99.1 99.2 99.4 98.8 ...
 $ no_of_premiums_paid              : num   8 3 14 13 15 4 8 4 8 8 ...
 $ sourcing_channel                 : chr   "A" "A" "C" "A" ...
 $ residence_area_type              : chr   "Rural" "Urban" "Urban" "Urban" ...
 $ premium                          : num   5400 11700 18000 13800 7500 3300 20100 3300 540
0 9600 ...
 $ renewal                          : num   1 1 1 1 0 1 1 1 1 1 ...
```

Below is in an initial overview of data available with us:

- The dataset consists of 17 variables and 79853 customer observations.
- We are to build a model which need to predict the probability that a customer will default the premium payment. Hence in our analysis '*renewal*' would be the target or the response variable i.e. the Dependent variable and other variables would be independent or the predictor variables
- Data has a mix of Indicator and Continuous variables which mainly covers Customer's demographic information, premium payment related behavior and Risk profiling
- Data limitation/assumptions: Based on above and visual inspection of data, below are some of the limitations to the information that can be inferred:
    - Currency of 'Income' is not provided. We can assume it to be Indian Rupees for our study.
    - 'Veh_Owned' doesn't clarify the type of vehicles owned (2-wheeler or a 4-wheeler or both)
    - 'No_of_dep' doesn't clarify the age group of dependents (kids, adults, elderly)
    - 'risk_score' doesn't clarify clearly its relation to the creditworthiness of customer (is it directly proportional or inversely proportional?). Also there is no information provided on the calculation methodology of it.

Data Preparation (basic)

Refer **Appendix -1** for the R code. Below are the findings:

1. Data has no missing value
2. Label encoding
    a. Converting 'residence_area_type' values to 1 and 0 (Rural =1, Urban =0)
    b. 'Converting 'Source Channel values to 1,2,3,4,5 (A, B, C, D, E)
3. For better readability, we have added new column
    a. 'cashPercent' to to display Cash premium payment in % terms.
    b. 'age' to display customer's age in years for improved readability
    c. 'countLatePayment' as a substitute for 'Count_3-6_months_late', 'Count_6-12_months_late', 'Count_more_than_12_months_late'

Variable-wise Exploratory Data Analysis

#summary(dataset)

Income	Count_3-6_months_late	Count_6-12_months_late
Min. : 24030	Min. : 0.0000	Min. : 0.00000
1st Qu.: 108010	1st Qu.: 0.0000	1st Qu.: 0.00000
Median : 166560	Median : 0.0000	Median : 0.00000
Mean : 208847	Mean : 0.2484	Mean : 0.07809
3rd Qu.: 252090	3rd Qu.: 0.0000	3rd Qu.: 0.00000
Max. :90262600	Max. :13.0000	Max. :17.00000

Count_more_than_12_months_late	Marital Status	Veh_Owned	No_of_dep
Min. : 0.00000	Min. :0.0000	Min. :1.000	Min. :1.000
1st Qu.: 0.00000	1st Qu.:0.0000	1st Qu.:1.000	1st Qu.:2.000
Median : 0.00000	Median :0.0000	Median :2.000	Median :3.000
Mean : 0.05994	Mean :0.4987	Mean :1.998	Mean :2.503
3rd Qu.: 0.00000	3rd Qu.:1.0000	3rd Qu.:3.000	3rd Qu.:3.000
Max. :11.00000	Max. :1.0000	Max. :3.000	Max. :4.000

Accomodation	risk_score	no_of_premiums_paid	residence_area_type
Min. :0.0000	Min. :91.90	Min. : 2.00	Min. :0.0000
1st Qu.:0.0000	1st Qu.:98.83	1st Qu.: 7.00	1st Qu.:0.0000
Median :1.0000	Median :99.18	Median :10.00	Median :0.0000
Mean :0.5013	Mean :99.07	Mean :10.86	Mean :0.3966
3rd Qu.:1.0000	3rd Qu.:99.52	3rd Qu.:14.00	3rd Qu.:1.0000
Max. :1.0000	Max. :99.89	Max. :60.00	Max. :1.0000

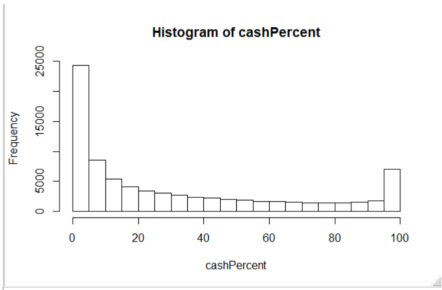
premium	renewal	cashPercent	age
Min. : 1200	Min. :0.0000	Min. : 0.00	Min. : 21.00
1st Qu.: 5400	1st Qu.:1.0000	1st Qu.: 3.40	1st Qu.: 41.00
Median : 7500	Median :1.0000	Median : 16.70	Median : 51.00
Mean :10925	Mean :0.9374	Mean : 31.43	Mean : 51.61
3rd Qu.:13800	3rd Qu.:1.0000	3rd Qu.: 53.80	3rd Qu.: 62.00
Max. :60000	Max. :1.0000	Max. :100.00	Max. :103.00

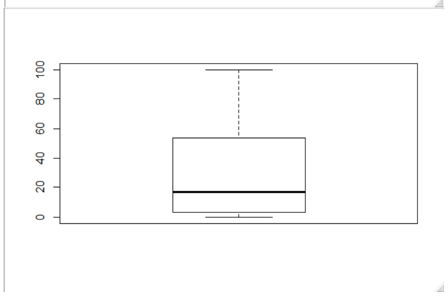
countLatePayment
Min. : 0.0000
1st Qu.: 0.0000
Median : 0.0000
Mean : 0.3864
3rd Qu.: 0.0000

Max. :19.0000

Data Analysis: Univariate

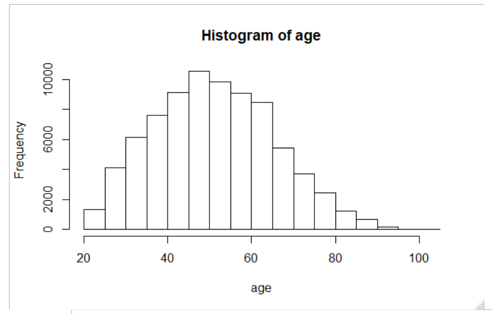
```
hist(cashPercent)
boxplot(cashPercent)
```

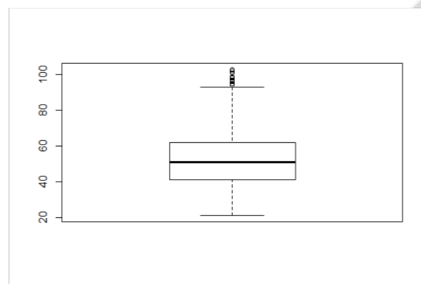




- **Values range from 0 to 100 with majority of data points falling in the lower range of 0% to 5%**
- **Mean = 31.43%**
- **Data has outliers**

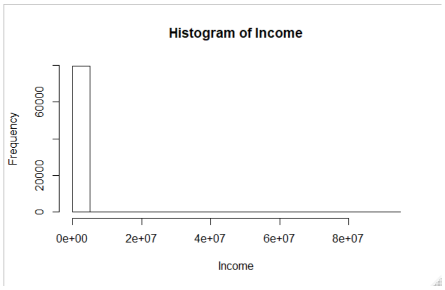
```
hist(age)
boxplot(age)
```

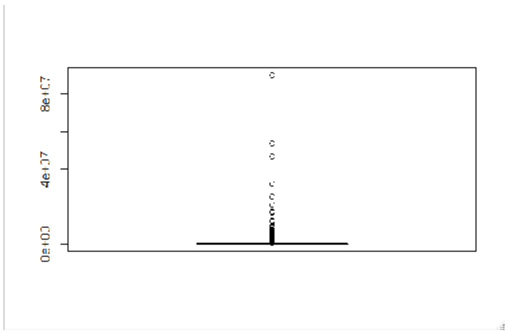




- **Values range from 21 to 103 years with data appearing to be somewhat normally distributed**
- **Mean = ~51 years = Median**
  Data has outliers

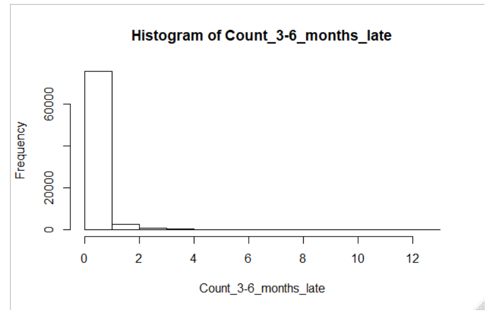
hist(Income)
boxplot(Income)

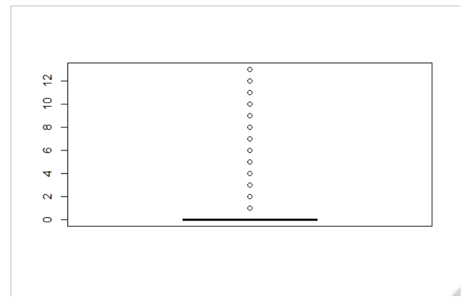




- **Data has a wide range of 24,030 to 90,262,600 (right skew)**
- **Mean = 208847**
- **Data has too many outliers**

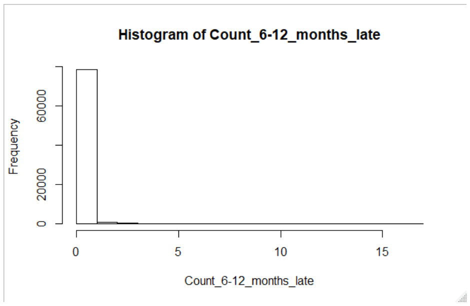
hist(`Count_3-6_months_late`)
boxplot(`Count_3-6_months_late`)

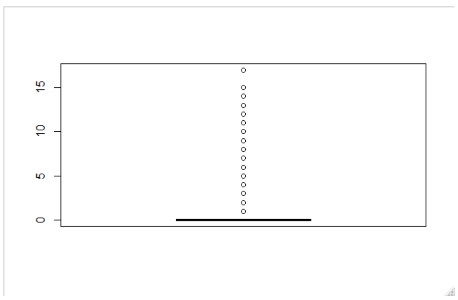




- **Data varies from 0 to 13 with majority delay counts are 0 to 1 (right skew)**
- **Mean = 0.2484**
- **Data has too many outliers**

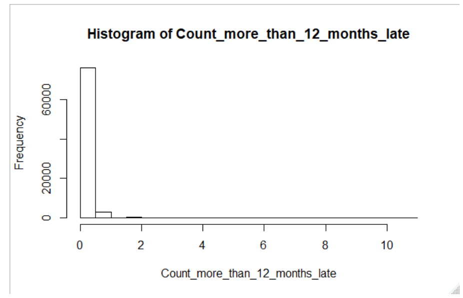
hist(‘Count_6-12_months_late’)
boxplot(‘Count_6-12_months_late’)

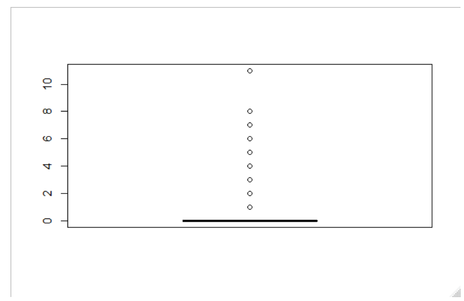




- **Data varies from 0 to 17 with majority delay counts are skewed towards 0 to 2 (right skew)**
- **Mean = 0.07809**
- **Data has too many outliers**

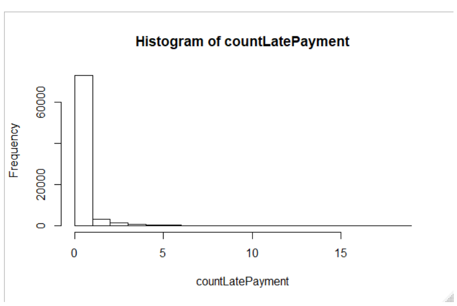
hist(Count_more_than_12_months_late)
boxplot(Count_more_than_12_months_late)

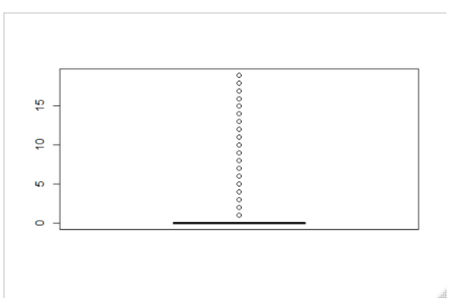




- **Data varies from 0 to 11 with majority delay counts are skewed towards 0 to 1 (right skew)**
- **Mean = 0.05994**
- **Data has too many outliers**

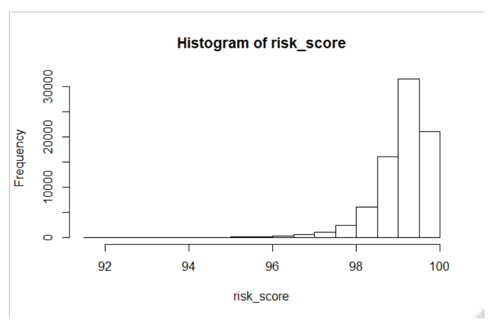
hist(countLatePayment)
boxplot(countLatePayment)

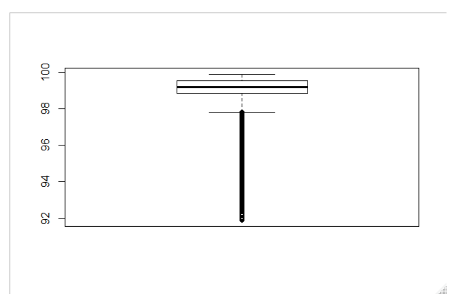




- **Data varies from 0 to 19 with majority delay counts are skewed towards 0 to 1 (right skew)**
- **Mean = 0.3864**
- **Data has too many outliers**

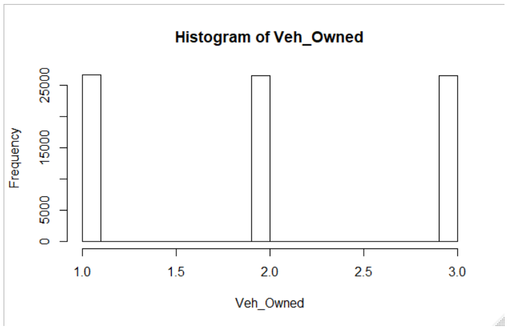
hist(risk_score)
boxplot(risk_score)





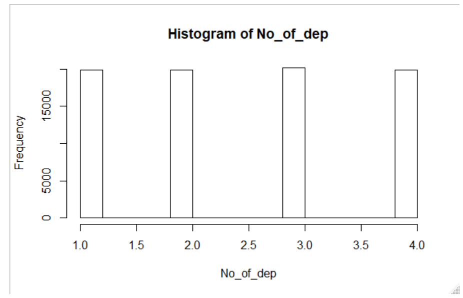
- **Data varies from 91.90 to 99.89 with majority data skewed towards 99.0 to 99.5 (left skew)**
- **Mean = 99.07**
- **Data has too many outliers**

```
hist(Veh_Owned)
boxplot(Veh_Owned)
```



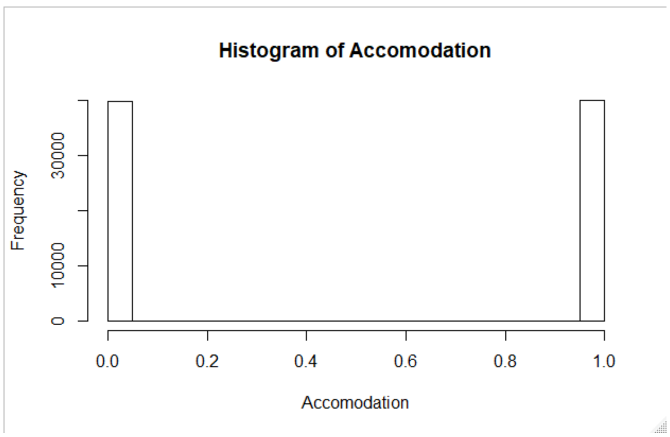
- **Data has 3 categories with almost equal no of cases for 1/2/3 vehicles owners**

```
hist(No_of_dep)
boxplot(No_of_dep)
```



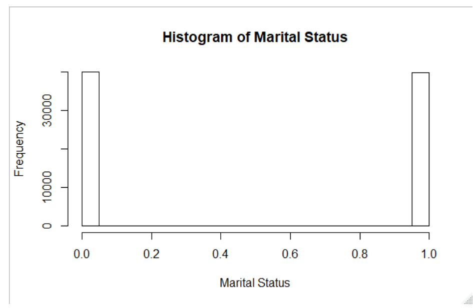
- **Data has 4 categories with almost equal no of 1,2,3,4 dependent cases**

```
hist(Accomodation)
boxplot(Accomodation)
```



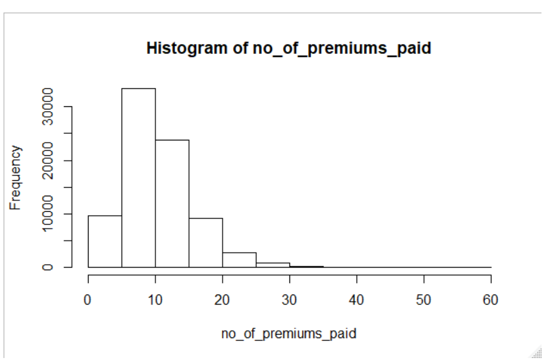
- **Data has 2 categories with almost equal no of Owned and Rented cases**

```
hist(`Marital Status`)
boxplot(`Marital Status`)
```

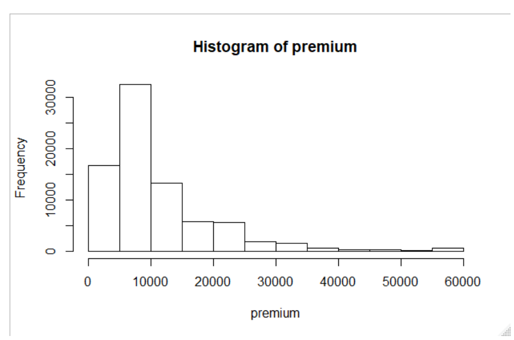


- **Data has 2 categories with Unmarried customers count is slightly more than Married customers.**

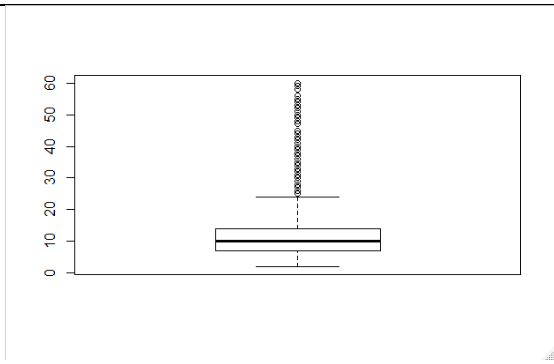
```
hist(no_of_premiums_paid)
boxplot(no_of_premiums_paid)
```




```
hist(premium)
boxplot(premium)
```

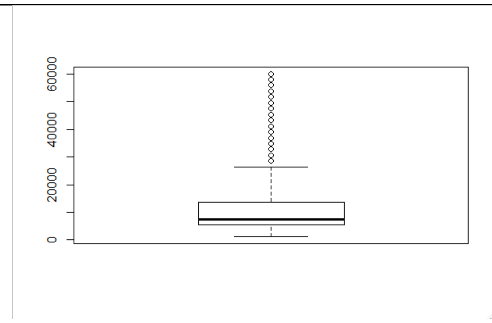


<table>
<tr>
<td>



- **Data varies from 2 to 60 with majority data falling between 5 to 15 (right skew)**
- **Mean = 10.86**
- **Data has too many outliers**

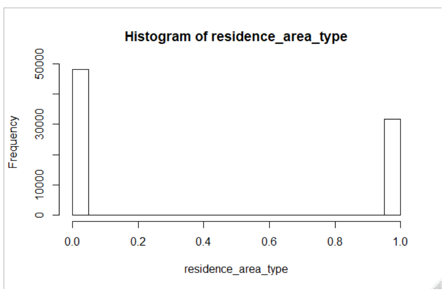
</td>
<td>



- **Data varies from 1200 to 60000 with majority data falling between 5000 to 10000 (right skew)**
- **Mean = 10925**
- **Data has too many outliers**

</td>
</tr>
<tr>
<td>

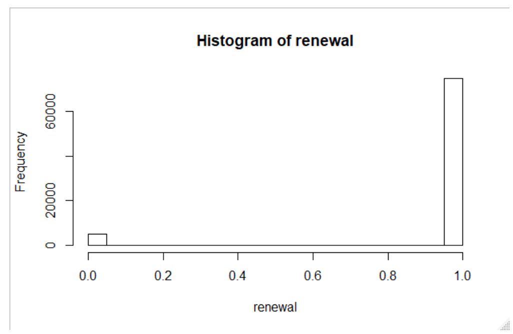
hist(residence_area_type)
boxplot(residence_area_type)



- **Data has 2 categories with more number of Urban (0) cases than Rural (1)**

</td>
<td>

hist(renewal)
boxplot(renewal)



- **Data has 2 categories with more number of renewed cases than non-renewed cases. It may lead to data imbalance problem which needs to be properly handled**

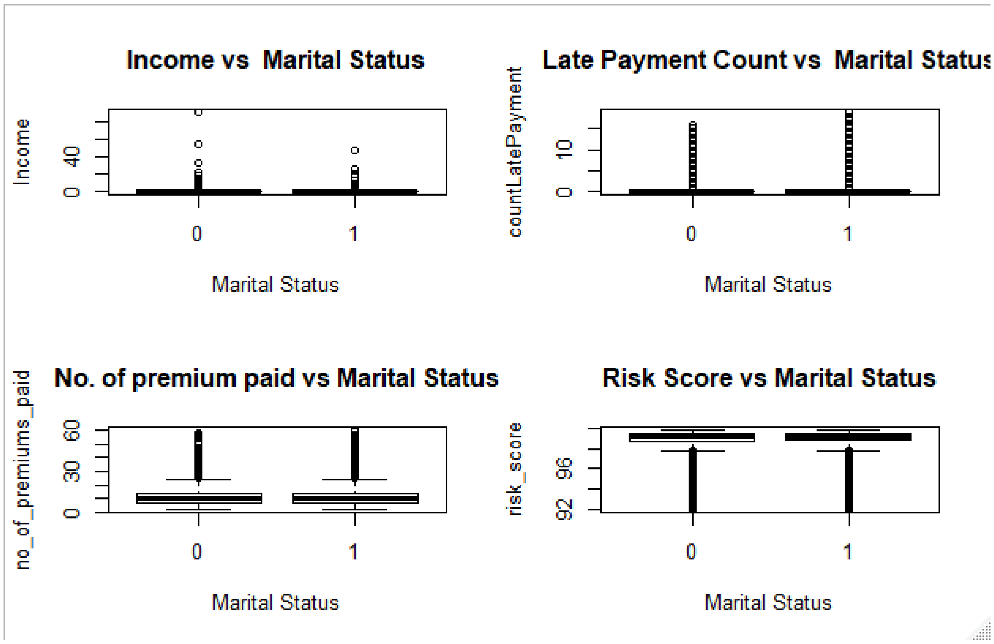
</td>
</tr>
</table>

Data Analysis: Bivariate and Multivariate

Refer **Appendix -1** for the R code.

Data Analysis: Bivariate

- **Marital status vs Income, Late Payment, No of premium paid & Risk score**

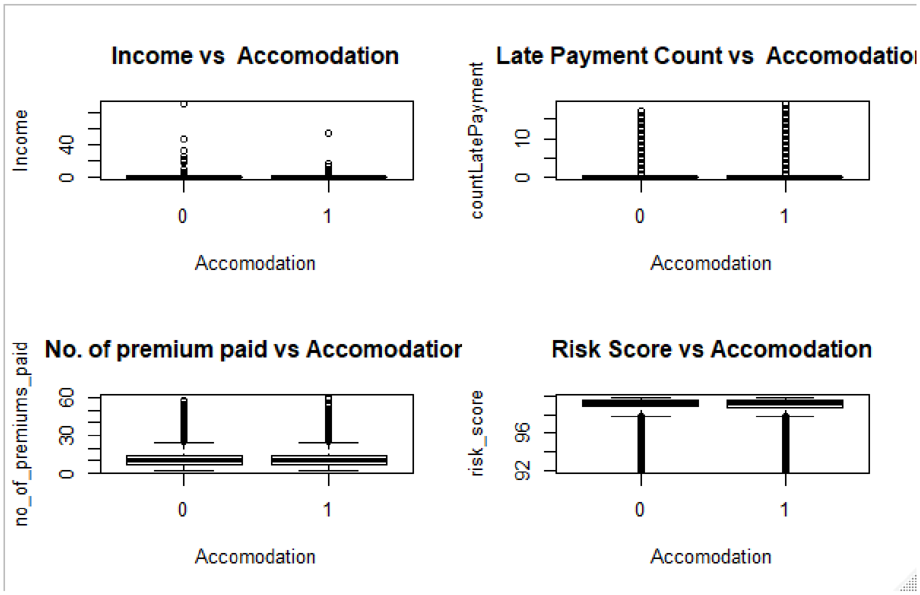


No significant difference across parameters between Married and Unmarried customers.

- **Residence Area Type vs Income, Late Payment, No of premium paid & Risk score**

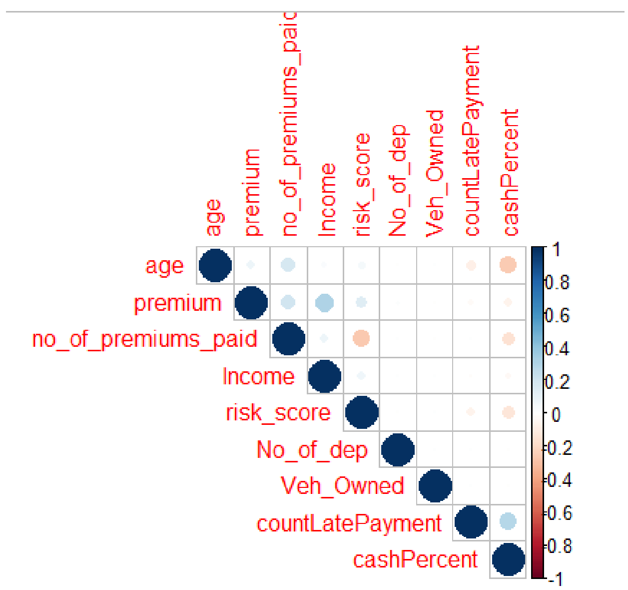
No significant difference across parameters between Urban and Rural resident customers.

- **Accommodation vs Income, Late Payment, No of premium paid & Risk score**



No significant difference across parameters between Rented and owned apartment customers.

Data Analysis: Multivariate



- Positive correlation between Premium and Income.
- Positive correlation between Cash Premium Percent and Count of late payment.
- Positive correlation between Premium and No of premiums paid.
- Positive correlation between Age and No of premiums paid.
- Negative correlation between Risk Score and No of premiums paid.
- Negative correlation between Age and Cash Premium Percent
- Negative correlation between Cash Premium Percent and No of premiums paid
- Negative correlation between Risk Score and Cash Premium Percent

There is no high correlation among variables, in general. However, from the above we can infer that: Customers making higher % of cash payment are likely to make more delayed payments and are likely to

have lower Risk Score. Higher age customers have paid more number of premiums but lesser premium amount in cash. Higher Income customers are likely to pay higher Premium.

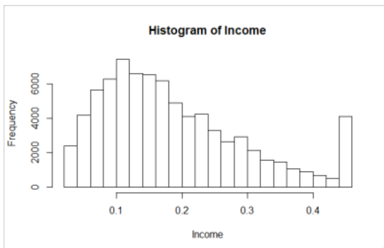
Outlier treatment

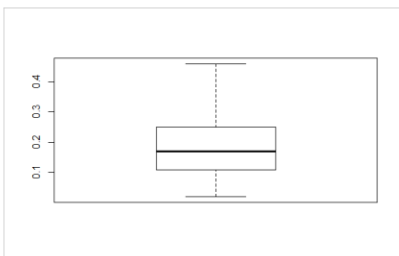
We have noticed that following parameters have outliers present:

Age – we will not treat this variable for outliers.

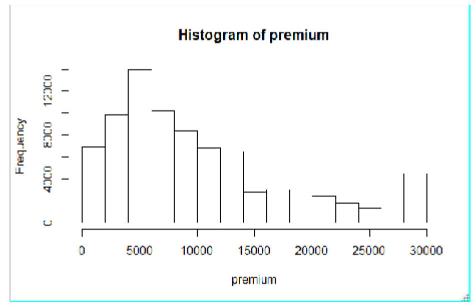
Rest of the affected variables are treated for outliers using capping methodology (Refer **Appendix -1** for the R code)

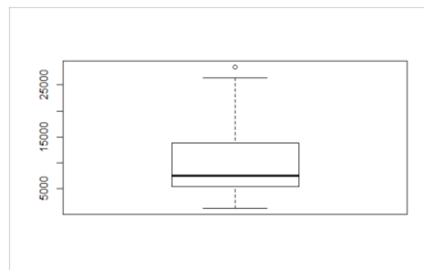
Income



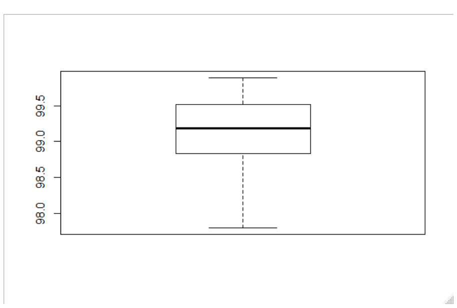


Premium

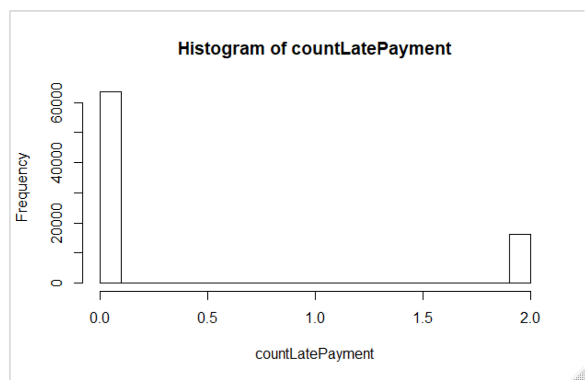


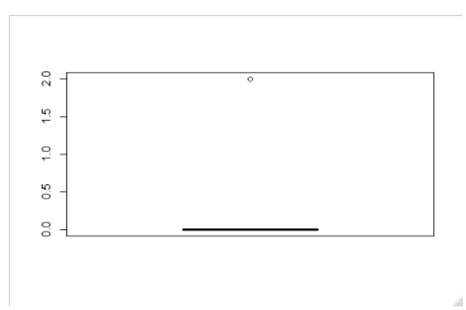


Risk Score

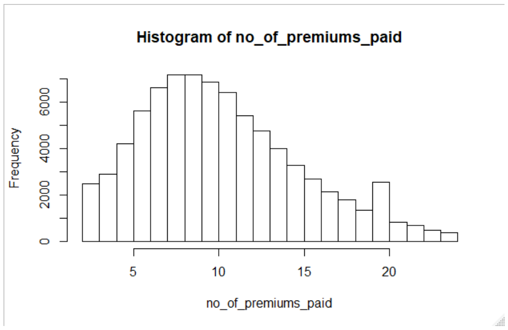


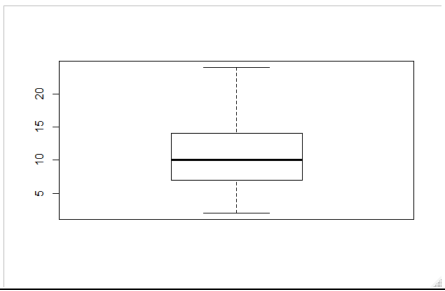
Count Late Payment





No of Premiums paid





Data normalization

Data variables are normalized to avoid any one variable overshadowing the model and data remains uniform.

Refer **Appendix -1** for the R code.

Synthetic Minority Over-sampling Technique (SMOTE)

It is a methodology to handle class imbalance problems. This is a statistical technique for increasing the number of cases in your dataset. The module works by generating new instances from existing minority cases.

In the dataset there is a clear class imbalance as renewal has only 6% of cases which has defaulted and remaining are No default cases.

Lets split the data such that we have 70% of the data is Train Data and 30% of the data is my Test Data and synthetically add entries to make it balanced. (**Appendix -1**)

Train data – SMOTE

Prior to smote operation no of entries for renewal

```
   0     1
 3475 52300
```

Post smote operation no of entries for renewal

```
    0     1
20850 34750
```

Test data - SMOTE

Prior to smote operation no of entries for renewal

```
   0     1
 1523 22555
```

Post smote operation no of entries for renewal

```
    0     1
16753 15230
```

Interpretation - With SMOTE operation new instances have been added to both test and train dataset therefore addressing the imbalance problem.

Refer **Appendix -1** for the R code.

Logistic Regression

Steps:

- Analyze the Base data provided to us vis-à-vis the modified data and test if the modification is adding value to the model.
    - Base data has individual columns for count for late payment (3_6 months , 6_12 months, >12 months) and modified data has single aggregated column for count of late payments.
- Run Logistic Regression function on train data and observe the significant variables
- Re-Run Logistic Regression function with significant variables
- Build the prediction model
- Use test data to analyze the model

Refer Appendix -2 for the related R code

Results:

- Based on Logistic regression with both Base and Modified data, "Veh_Owned" ,"Sourcing_channel" and "Residence_area_type" are insignificant variables. Intercept is significant in both models.
  However, as Modified data model is not adding any value to the Base data model, we will continue with the Base data model i.e. individual values for count of delay columns.
- Based on Logistic Regression, Income, \`Count_3-6_months_late\`, \`Count_6-12_months_late\`, Count_more_than_12_months_late, Marital Status, No_of_dep, Accommodation, risk_score, no_of_premiumspaid, premium, cashPercent and age are significant variables.
- Regression equation is log odds(y) = 0.54873 + 28.58339 * Income – 7.57962 * \`Count_3-6_months_late\` - 20.56685* \`Count_6-12_months_late\` - 10.71174* Count_more_than_12_months_late + 0.05883* Marital Status – 0.09284*No_of_dep -0.04515*Accommodation + 1.23170*risk_score – 1.96536 * no_of_premiums_paid + 0.50082*premium -1.89267* cashPercent + 1.49127*age
- Income, Marital Status, risk_score , premium , age have positive coefficients which means higher values of these variables will result in a likely renewal.
- \`Count_3-6_months_late\`, \`Count_6-12_months_late\`, Count_more_than_12_months_late, No_of_dep, Accommodation, no_of_premiums_paid, cashPercent have negative coefficients which means higher values of these variables will NOT result in a likely renewal.

K-Nearest Neighbour (KNN)

Steps:

- Run KNN function with various values of K and find the optimum value
- Finally based on optimum value, build model and assess model performance parameters

Refer Appendix-3 for the related R code

Results

- K=3 provides the optimal result

Naïve Bayes

Steps:

- Run NB function
- Run Predict function

Refer Appendix-4 for the related R code

Results/Inference

- Is NB applicable here? - As we have seen in Multivariate analysis earlier, the data correlation is not significantly high across variables and they are independent of each other, we can apply NB model

Random forest

Steps:

- Generate random forest using 'renewal' as dependent variable and others as independent variable.
- Predict values and assess model performance

Refer Appendix-5 for the related R code

Results:

- The error rate plot w.r.t number of trees reveals that anything more than, say 50, trees is really not that valuable. So we can assume odd value of 51 trees to confirm with the majority rule application.
- Based on Random Forest, Income, Count_3_6_months_late, Count_6_12_months_late, Count_more_than_12_months_late, No_of_dep, risk_score, no_of_premiums_paid, sourcing_channel, premium, cashPercent, age are significant variables.

Bagging (Ensemble method)

Bagging (aka Bootstrap Aggregating): is a way to decrease the variance of your prediction by generating additional data for training from your original dataset using combinations with repetitions to produce multisets of the same cardinality/size as your original data.

Refer Appendix-6 for the related R code

Model Performance Measurement

Parameter	Logistic regression (threshold=0.5)	Random Forest	KNN	NB	With Bagging
Classification Error	0.27	0.10	0.37	0.25	0.17
Accuracy	0.73	0.90	0.63	0.74	0.83
Loss	0.41	0.04	0.57	0.37	0.26
Opportunity Loss	0.11	0.15	0.14	0.13	0.07

Top 2 models per above comparison are Logistic Regression and Random Forest .

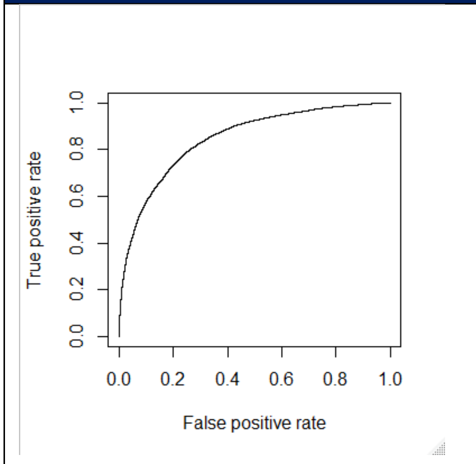
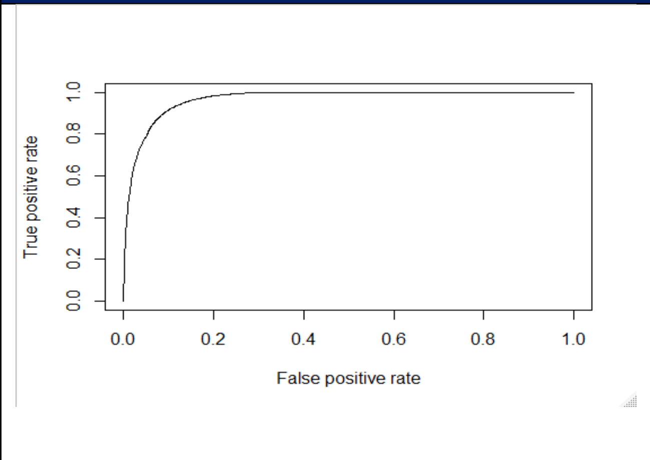
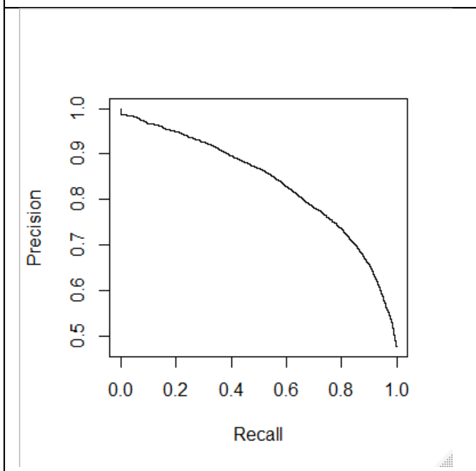
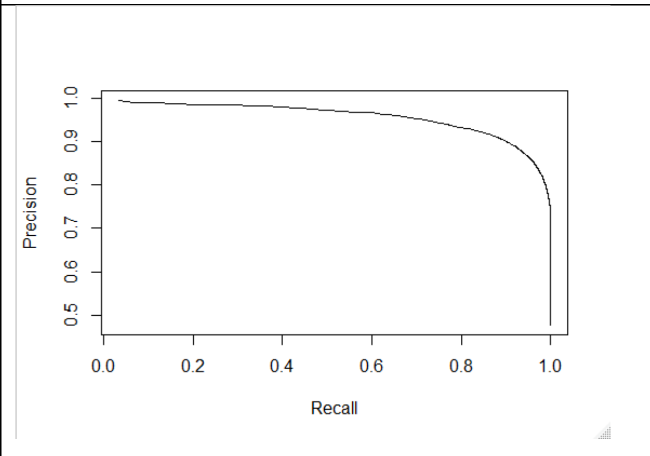
Let's further compare these 2 models:

Parameter	Logistic regression*	Random Forest
Classification Error Rate	0.24	0.10
Accuracy	0.76	0.90
Specificity TN/(TN+FP)	0.86	0.85
Sensitivity TP/(TP+FN)	0.65	0.96
AUC	0.85	0.97
KS	0.54	0.82
Gini	0.69	0.46

** Keeping a higher threshold of 0.75 to ensure high specificity which is capturing no of defaulters (renewal =0 means a defaulter in our data)*

ROC curve and Precison-recall curve

- A ROC curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR). The TPR is the proportion of observations that were correctly predicted to be positive out of all positive observations (TP/(TP + FN)). Similarly, the FPR is the proportion of observations that are incorrectly predicted to be positive out of all negative observations (FP/(TN + FP)).
- A ROC curve shows the trade-off between sensitivity (TPR) and specificity (1 – FPR). If the curve is closer to the top-left corner it indicates a better performance. As a baseline, a random classifier is expected to give points lying along the diagonal (FPR = TPR). The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.
- A precision-recall curve shows the relationship between precision (positive predictive value) and recall (sensitivity) for every possible cut-off.
- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)
- The closer a Precision-Recall Curve is to the upper right corner, the better the performance is.

Logistic regression	Random Forest
	
	

Interpretation of Model Measures:

- Random Forest has a lower CER
- Random Forest has a higher accuracy
- Logistic Regression has a lower specificity.

- Logistic Regression has a lower sensitivity
- Logistic Regression has a lower AUC
- Random Forest has a higher K-S
- As mentioned above from the curves it can be seen that the ROC curve for Random forest is closer to the left corner and the PRC curve is closer to the right corner.

Therefore, from above comparisons it can be seen that Random Forest has overall better performance indicators.

Summary

- **The dataset consists of 17 variables and 79853 customer observations with a combination of Indicator and continuous variables.**
- **Data mainly covers Customer's demographic information, premium payment related behavior and Risk profile information.**
- **'renewal' would be the target or the response variable i.e. the Dependent variable and other variables would be independent or the predictor variables.**
- **There is no missing value in the data.**
- **Data has outliers present and is skewed on most of the numeric variables.**
- **Most of the categorical variables have equal representation of categories.**
- **'renewal' has higher count of renewed cases than non-renewed cases. It may lead to data imbalance problem which needs to be properly handled.**
- **We have introduced a new column for total count of late premium payment by adding the 3- 6 months, 6-12 months, more than 12 months late count figures.**
- **For better readability, we have converted Age in years and cash premium payment percentage in %**
- **We are ignoring 'Sourcing Channel' as details about the various categories (A,B,C,D) aren't provided**
- **As per bivariate analysis, there is no significant variation in the premium, count of late payment, risk score and No of premium paid vis-à-vis Marital staus, Accommodation and Residence Type**
- **No significant correlation is present among variables. Few important inferences which we can draw from correlation plots are - Customers making higher % of cash payment are likely to make more delayed payments and are likely to have lower Risk Score. Higher age customers have paid more number of premiums but lesser premium amount in cash. Higher Income customers are likely to pay higher Premium.**

- We treated the variables for the outliers.

- ‘renewal’ has higher count of renewed cases than non-renewed cases. It may lead to data imbalance problem which needs to be properly handled. We addressed the same through methods like SMOTE and Bagging

- With the synthetically enhanced dataset, we performed model study through various techniques like Logistic Regression, Random Forest,KNN, Naïve Bayes and Bagging

- Based on Model performance measures, we observed that Random Forest has the best performance indicators .

- Both as per Logistic Regression and Random Forest Techniques a mix of demographic and financial parameters are significant.
    - Both models confirmed that “Vehicle owned” and “Residence area type” are not significant
    - Logistic Regression confirmed that “Sourcing Channel” is not significant and Random Forest confirmed that “Marital Status” is not significant variable for the study.
    - Income, Marital Status, risk_score , premium , age have positive coefficients which means higher values of these variables will result in a likely renewal.
    - \`Count_3-6_months_late\`, \`Count_6-12_months_late\`, Count_more_than_12_months_late, No_of _dep, Accommodation, no_of_premiums_paid, cashPercent have negative coefficients which means higher values of these variables will result in a NON-likely renewal.

- Correctly identifying customers with non-renewal likelihood is critical.
    - Business can use ‘Random Forest’ predictions to identify such customers and run their programs with them for increased retention/acquisition.

- As per models, a mix of demographic and financial parameters are significant here. Business can use it along with correlations information of variables in the strategy formulation
    - E.g. Higher ‘Income’, ‘Marital Status’, ‘Risk score’ , ‘Premium’ and ‘Age’ will result in a likely renewal but higher ‘Count of late payment’, ‘No of dependent’ , ‘Accommodation’, ‘No of premiums paid’, and ‘cash Percent’ will result in a likely non-renewal

- Following approach can be taken to make the model more reliable:
    - Get more data from business. A larger dataset might expose a different and perhaps more balanced perspective on the classes.

Appendix -1: (Data pre-processing, Exploratory Data Analysis, Outlier treatment, SMOTE)

Data pre-processing

```
#Set working directory and read the data file
setwd("#File Path") library("readxl")
dataset=read_excel("premium.xlsx" , sheet = 1)

# Analyze data str
(dataset)
```

- **Let's first check if data has any missing values**

```
#check for NA
anyNA(dataset)
[1] FALSE
```

As above checking is providing 0 NA values, we can conclude that data has no missing value.

- **Converting 'residence_area_type' values to 1 and 0 (Rural =1, Urban =0)**

```
#convert values to 1 and 0
dataset$ residence_area_type=ifelse(dataset$ residence_area_type=="Rural",1,0)

# check first few values
head(dataset$residence_area_type)
[1] 1 0 0 0 0 1
```

- **Introduce new column 'cashPercent' to display Cash premium payment in % terms for improved readability**

```
# Current -      check first few values of 'perc_premium_paid_by_cash_credit'
head(dataset$perc_premium_paid_by_cash_credit)

[1] 0.317 0.000 0.015 0.000 0.888 0.512


#For better readability,adding a new column 'cashPercent' to dataset
dataset$cashPercent = dataset$perc_premium_paid_by_cash_credit*100

# Revised - check first few values of new column 'cashPercent'
head(dataset$cashPercent)
[1] 31.7     0.0    1.5    0.0 88.8 51.2
```

We will be using this new column 'cashPercent' instead of 'perc_premium_paid_by_cash_credit'

- **Introduce new column 'age' to display customer's age in years for improved readability**

```
# current - check first few values of the column 'age_in_days'
head(dataset$age_in_days)

[1] 0.317 0.000 0.015 0.000 0.888 0.512

#For better readability, adding new column 'age' to our dataset
dataset$age = round(dataset$age_in_days/365, digits=0)

# check first few values of the new column 'age'
head(dataset$age)  [1]
31 83 44 65 53 46
```

We will be using this new column 'age' instead of 'age_in_days'

- **Remove columns which are duplicate or are not required**

#remove 'Id', 'perc_premium_paid_by_cash_credit', 'age_in_days', 'sourcing_channel'
dataset = dataset[,c(-1,-2,-3,-14)]
#check the filtered columns
str(dataset)

```
Classes 'tbl_df', 'tbl' and 'data.frame':                79853 obs. of     16 variables:
 $ Income                            : num   90050 156080 145020 187560 103050 ...
 $  Count_3-6_months_late            :  num   0 0  1 0 7   0 0 0 0 0    ...
 $  Count_6-12_months_late           :  num   0 0  0 0 3   0 0 0 0 0    ...
 $  Count_more_than_12_months_late:  num   0 0  0 0 4   0 0 0 0 0    ...
 $  Marital Status                   :  num   0 1  0 1 0   0 0 0 1 1    ...
 $  Veh_Owned                        :  num   3 3  1 1 2   1 3 3 2 3    ...
 $  No_of_dep                        :  num   3 1  1 1 1   4 4 2 4 3    ...
 $  Accomodation                     :  num   1 1  1 0 0   0 1 0 1 1    ...
 $ risk_score                        : num   98.8 99.1 99.2 99.4 98.8 ...
 $ no_of_premiums_paid               : num   8 3 14 13 15 4 8 4 8 8 ...
 $ residence_area_type               : num   1 0 0 0 0 1 1 0 0 1 ...
 $ premium                           : num   5400 11700 18000 13800 7500 3300 20100 3300
5400 9600 ...
 $ renewal                           : num   1 1 1 1 0 1 1 1 1 1 ...
 $ cashPercent                       : num   31.7 0 1.5 0 88.8 51.2 0 99.4 1.9 1.8 ...
 $ age                               :  num   31  83  44  65  53  46  45  39  76  82  ...
 $ countLatePayment                  : num   0 0 1 0 14 0 0 0 0 0 ...
```

- **Attach column names**

```
#Attach column names for ease of operation
attach(dataset)
```

Data Analysis: Bivariate

- **Marital status vs Income, Late Payment, No of premium paid & Risk score**

```
par(mfrow=c(2,2))
boxplot(Income~`Marital Status`, main="Income vs         Marital Status")
boxplot(countLatePayment~`Marital Status`, main="Late Payment Count vs            Marital Sta
tus")
boxplot(no_of_premiums_paid~`Marital Status`,main="No.of premium paid vs Marital St atus")
boxplot(risk_score~`Marital Status`, main="Risk Score vs Marital Status")
```

- **Residence Area Type vs Income, Late Payment, No of premium paid & Risk score**

```
par(mfrow=c(2,2))
boxplot(Income~residence_area_type, main="Income vs        Residence Area Type")
boxplot(countLatePayment~residence_area_type, main="Late Payment Count vs            Residenc
e Area Type")
boxplot(no_of_premiums_paid~residence_area_type, main="No. of premium paid vs Resid ence
Area Type")
boxplot(risk_score~residence_area_type, main="Risk Score vs Residence Area Type")
```

- **Accommodation vs Income, Late Payment, No of premium paid & Risk score**

```
par(mfrow=c(2,2))
boxplot(Income~Accomodation, main="Income vs      Accomodation")
boxplot(countLatePayment~Accomodation, main="Late Payment Count vs          Accomodation")
```

```
boxplot(no_of_premiums_paid~Accomodation, main="No. of premium paid vs Accomodation ")
boxplot(risk_score~Accomodation, main="Risk Score vs Accomodation")
```

Data Analysis: Multivariate

```
dataset_const_vars = dataset[,c(1,6,7,9,10,12,14,15,16)]
corrmatrix = cor(dataset_const_vars)
library(corrplot)
corrplot(corrmatrix,method='circle', type='upper', order='FPC')
```

Outlier treatment

```
x <- Income
qnt <- quantile(x, probs=c(.25, .75), na.rm = T) caps <-
quantile(x, probs=c(.05, .95), na.rm = T) H <- 1.5 *
IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]
Income=x
hist(Income)
boxplot(Income)
```

Data normalization

```
#normalize variables
normalize<-function(x){
  +return((x-min(x))/(max(x)-min(x)))}
dataset$Income = normalize(dataset$Income)
dataset$risk_score = normalize(dataset$risk_score)
dataset$premium = normalize(dataset$premium)
dataset$age = normalize(dataset$age)
dataset$cashPercent = normalize(dataset$cashPercent)
dataset$no_of_premiums_paid = normalize(dataset$no_of_premiums_paid)
```

Synthetic Minority Over-sampling Technique (SMOTE)

It is a methodology to handle class imbalance problems. This is a statistical technique for increasing the number of cases in your dataset. The module works by generating new instances from existing minority cases.

#we are splitting the data such that we have 70% of the data is Train Data and 30% of the data is my Test Data

```
library(DMwR)
set.seed(1234)
pd<-sample(2,nrow(dataset),replace=TRUE, prob=c(0.7,0.3))
train<-dataset[pd==1,]
test<-dataset[pd==2,]
```

```
#count of train data prior to SMOTEoperation
table(train$renewal) 0
    1
 3475 52300
```

Train data - SMOTE

```
# SMOTE operation on TRAIN data
train=as.data.frame(train)
train$renewal=as.factor(train$renewal)
smote.train <- SMOTE(renewal ~., data = train, perc.over = 500, k = 5, perc.under = 200)
```

```
table(smote.train$renewal) 0
    1
20850 34750
```

Test data - SMOTE

```
#count of test data prior to SMOTEoperation
table(test$renewal) 0
        1
 1523 22555

# SMOTE operation on TEST data
test=as.data.frame(test)
test$renewal=as.factor(test$renewal)
smote.test <- SMOTE(renewal ~., data = test, perc.over = 1000, k = 5, perc.under = 100)

table(smote.test$renewal) 0
        1
16753 15230
```

Interpretation - With SMOTE operation new instances have been added to both test and train dataset therefore addressing the imbalance problem.

```
#now put our SMOTE data into our best xgboost
smote_features_train<-as.matrix(smote.test[,c(1:9,11:13)])
smote_label_train<-as.matrix(smote.test$renewal)

smote.xgb.fit <- xgboost(
  data = smote_features_train,          label = smote_label_train,
  eta = 0.7,       max_depth = 5,    nrounds = 50,     nfold = 5,
  objective = "binary:logistic",              # for regression models
    verbose = 0,                          # silent,early_stopping_rounds = 10)

smote_features_test<-as.matrix(smote.test[,c(1:9,11:13)])
smote.test$smote.pred.class <- predict(smote.xgb.fit, smote_features_test)
table(smote.test$renewal,smote.test$smote.pred.class>=0.5)


FALSE   TRUE
  0 15937
                  81
```

Classification Error Rate (CER) = (109+816) / 31983 = 0.02

Appendix -2: Logistic Regression

```
#logistic regression with BASE data
smote.train.base=smote.train[,c(-17)]
german_logistic <- glm(renewal~., data=smote.train, family=binomial(link="logit"))
summary(german_logistic)

Call:
glm(formula = renewal ~ ., family = binomial(link = "logit"),
     data = smote.train.base)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
 -2.9063   -0.7805    0.4821    0.7117    4.9180

Coefficients:
                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                         0.569342   0.124784   4.563 5.05e-06
Incom                              29.289523   9.155302   3.199  0.00138
`Count_3-                          -7.567173   0.200938 -37.659  < 2e-16
6_months_late`                             -   ***
Count_more_than_12_months_late -               0.557323 -36.896  < 2e-16
Marital_Status                      0.058799   0.021295   2.761  0.00576
Veh_Owne                            0.001708   0.026081   0.065  0.94779
No_of_de                           -0.092289   0.028972  -3.185  0.00145
Accomodatio                        -0.045288   0.021280  -2.128  0.03332
risk_score                          1.225606   0.12111   10.12   < 2e-16
no_of_premiums_p                           -   0.139809 -          ***
sourcing_channel                   -0.012202   0.010246  -1.191  0.23369
residence_area_type                 0.017801   0.021788   0.817  0.41391
premium                             0.512490   0.10674    4.801 1.58e-06
cashPercen                                 -   0.031966 -          < 2e-16
t age                               1.891940   59.185              ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

        Null deviance: 73566  on 55599  degrees of freedom
    Residual deviance: 53982  on 55584  degrees of freedom
AIC: 54014

Number of Fisher Scoring iterations: 6


#logistic regression with MODIFIED data
# remove individual columns for count ofdelays
smote.train.filter=smote.train[,c(-2,-3,-4)]
german_logistic_filter <- glm(renewal~., data=smote.train.filter, family=binomial(link="l ogit"))
summary(german_logistic_filter)

Call:
glm(formula = renewal ~ ., family = binomial(link = "logit"),
     data = smote.train.filter)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
 -2.9725   -0.7984    0.4799    0.7116    4.3158

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       6.396e-01  1.254e-01   5.102 3.36e-07
Income            ***
Marital_Status    5.140e-02  2.122e-02   2.422 0.015441 *
Veh_Owne         -8.379e-04  2.600e-02  -0.032 0.974289
No_of_de         -9.881e-02  2.886e-02  -3.423 0.000618
```

```
Accomodation          -4.868e-02  2.121e-02  -2.295 0.021729 *
risk_score             1.107e+00  1.216e-01   9.105  < 2e-16
*** no_of_premiums_paid -1.608e+00            1.394e-01 -11.536
sourcing_channel      -1.062e-02  1.021e-02  -1.040 0.298429
residence_area_type    1.201e-02  2.171e-02   0.553 0.580071
premium                5.247e-01  1.074e-01   4.886 1.03e-06
cashPercen            -1.924e+00  3.178e-02 -60.548  < 2e-16
t age                  *** 1.479e+00          7.227e-02
countLatePayme                    20.461      < 2e-16 ***
---
Signif. codes:   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

       Null deviance: 73566  on 55599  degrees of freedom
   Residual deviance: 54335  on 55586  degrees of freedom
AIC: 54363

Number of Fisher Scoring iterations: 6
```

Significant variables are highlighted in Bold in above matrix. Based on comparison of above 2 models as there is no significant change in the models with modified data, we will go with base data model only.

```
#Re-run logistic regression with BASE data only for significant variables
german_logistic_base <- glm(renewal~ .-Veh_Owned -sourcing_channel -residence_area_type,
data=smote.train.base, family=binomial(link="logit"))
summary(german_logistic_base)

Call:
glm(formula = renewal ~ . - Veh_Owned - sourcing_channel - residence_area_type,
    family = binomial(link = "logit"), data = smote.train.base)

Deviance Residuals:
     Min        1Q    Median       3Q      Max
 -2.8768   -0.7809    0.4819   0.7123   4.9202

Coefficients:
                                       Estimate Std. Error z value Pr(>|z|)
(Intercept)                              0.5487    0.12130   4.524 6.08e-06
Incom                                   28.5833    9.07623   3.149  0.00164
`Count_3-                              -7.57962    0.20075 -37.756  < 2e-16
6_months_late`                                -    ***
Count_more_than_12_months_late -                  0.55726 -36.907  < 2e-16
Marital_Status                          0.05883    0.02129   2.763  0.00573
No_of_dep                                     -    **
Accomodatio                             -          0.02128  -2.122  0.03386
risk_score                              1.23170    0.12084            < 2e-16
no_of_premiums_p                              -    0.13963 -          ***
aid premium                             1.96536    0.1062    4.714 2.43e-06
cashPercent                             0.50082    0.03196 -59.224  < 2e-16
age                                           -    0.0714   20.878  < 2e-16
---
Signif. codes:   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

       Null deviance: 73566  on 55599  degrees of freedom
   Residual deviance: 53984  on 55587  degrees of freedom
AIC: 54010

Number of Fisher Scoring iterations: 6

vif(german_logistic_base)
Income        `Count_3-6_months_late`        `Count_6-12_months_late`  [redacted]
```

2.037777	1.055976	1.064043	[redacted]

Count_more_than_12_ onths_late m	Marital_Status	No_of_dep
1.054082	1.001350	1.001282
Accomodation	risk_score	no_of_premiums_paid
1.000589	1.198355	1.433690
premium	cashPercent	age
1.973951	1.162685	1.107753

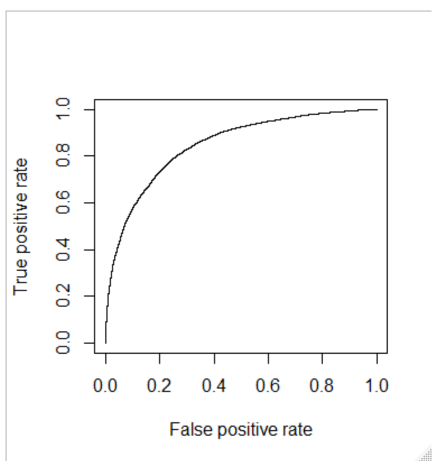
#Predict and assess model performance

```
smote.test.base=smote.test[,c(-17)]
smote.test.base$log.pred<-predict(german_logistic_base, smote.test.base[,c(-14)], type="r esponse")
tab.logit= table(smote.test.base$renewal,smote.test.base$log.pred>0.5) tab.logit

     FALSE  TRUE
  0   9873  6880
  1   1650 13580
```

sensitivity (TRUE POSITIVE RATE) is 13580/15230 = 0.89
Our specificty is 9873/16753 = 0.59
FALSE POSITIVE RATE = 1-0.59 = 0.41

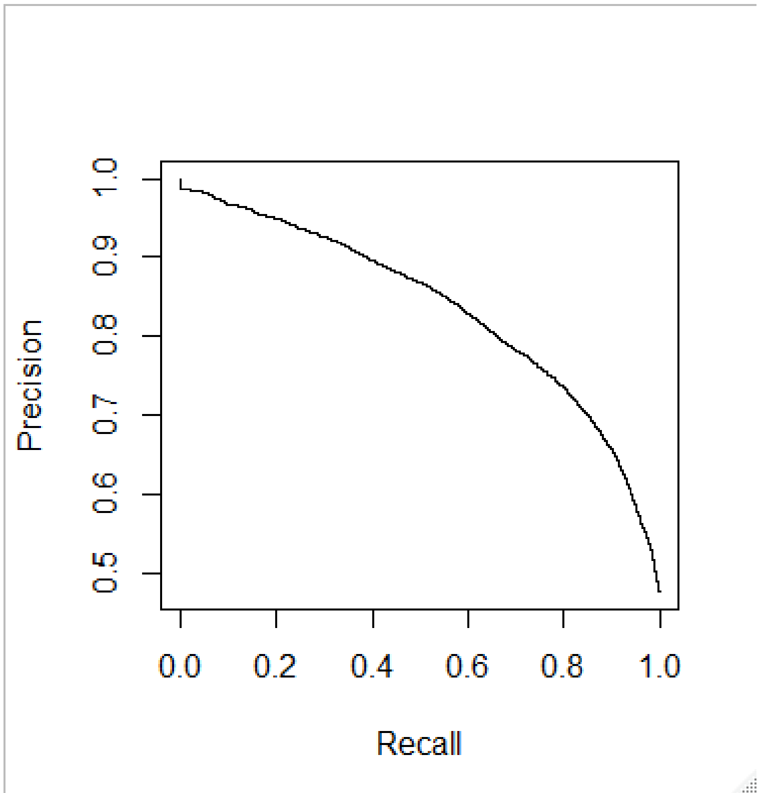
Classification Error Rate (CER) = (1650+6880) /31983 = 0.27

```
accuracy.logit<-sum(diag(tab.logit))/sum(tab.logit) accuracy.logit
[1] 0.7332958

loss.logit<-tab.logit[1,2]/(tab.logit[1,2]+tab.logit[1,1]) loss.logit
[1] 0.4106727

opp.loss.logit<-tab.logit[2,1]/(tab.logit[2,1]+tab.logit[2,2]) opp.loss.logit
[1] 0.1083388

tot.loss.logit<-0.95*loss.logit+0.05*opp.loss.logit tot.loss.logit
[1] 0.395556

#ROC Plot
pred= prediction(smote.test.base$log.pred,smote.test.base$renewal) perf =
performance(pred, "tpr", "fpr")
plot(perf)
```




```
# KS
train.ks <- max(attr(ks.train, "y.values")[[1]] - (attr(ks.train, "x.values")[[1]])) train.ks
```

```
[1] 0.540154

# AUC
train.auc = performance(pred, "auc")
train.area = as.numeric(slot(train.auc, "y.values")) train.area
[1] 0.8472208

# Gini
train.gini = (2 * train.area) - 1
train.gini
[1] 0.6944415

#Precision-recall curve
```




```
PRcurve(smote.test.base$log.pred,smote.test.base$renewal)
```

Appendix -3: K-Nearest Neighbour

```
# Let's try with different values of k and decide the best one

#knn3 library(class)
knn_fit<- knn(train = smote.train.base[,c(1:13,15:16)], test = smote.test.base[,c(1:13,15
:16)], cl= smote.train.base[,14],k = 3,prob=TRUE)
table(smote.test.base[,14],knn_fit)

    knn_fit
          0     1
  0     721  9535
  1    2145 13085
```

Classification Error Rate (CER) = (2145+9535) /31983 = 0.37

```
#knn5
knn_fit_5<- knn(train = smote.train.base[,c(1:13,15:16)], test = smote.test.base[,c(1:13, 15:16)], cl=
smote.train.base[,14],k = 5,prob=TRUE)  table(smote.test.base[,14],knn_fit_5)

          0     1
  0    758    917
  1    2398 12832
```

Classification Error Rate (CER) = (2398+9171) / 31983 = 0.36
As there is no significant accuracy improvement with increase in K, we will settle with K value of 3 only for this study.

```
tab.knn.3 = table(smote.test.base[,14],knn_fit)

accuracy.knn.3<-sum(diag(tab.knn.3))/sum(tab.knn.3)
accuracy.knn.3
[1] 0.634806

loss.knn.3<-tab.knn.3[1,2]/(tab.knn.3[1,2]+tab.knn.3[1,1])
loss.knn.3
[1] 0.56915

opp.loss.knn.3<-tab.knn.3[2,1]/(tab.knn.3[2,1]+tab.knn.3[2,2])
opp.loss.knn.3
[1] 0.14084

tot.loss.knn.3<-0.95*loss.knn.3+0.05*opp.loss.knn.3 tot.loss.knn.3
[1] 0.547736
```

Appendix – 4: Naïve Bayes

#performing Naïve Bayes model alogrithm

```
NB<-naiveBayes(x=smote.train.base[-14], y=smote.train.base$renewal)
```

#predicting the model values

```
y_pred.NB<-predict(NB,newdata=smote.test.base[-14])


tab.NB=table(smote.test.base[,14],y_pred.NB) tab.NB

  pred_nb
        0       1
        0 10580
                617
```

Classification Error Rate (CER) = (1974+6173) / 31983 = 0.25

```
accuracy.NB<-sum(diag(tab.NB))/sum(tab.NB)
accuracy.NB
[1] 0.7452709

loss.NB<-tab.NB[1,2]/(tab.NB[1,2]+tab.NB[1,1]) loss.NB
[1] 0.368471

opp.loss.NB<-tab.NB[2,1]/(tab.NB[2,1]+tab.NB[2,2])
opp.loss.NB
[1] 0.129613

tot.loss.NB<-0.95*loss.NB+0.05*opp.loss.NB tot.loss.NB
[1] 0.356528
```

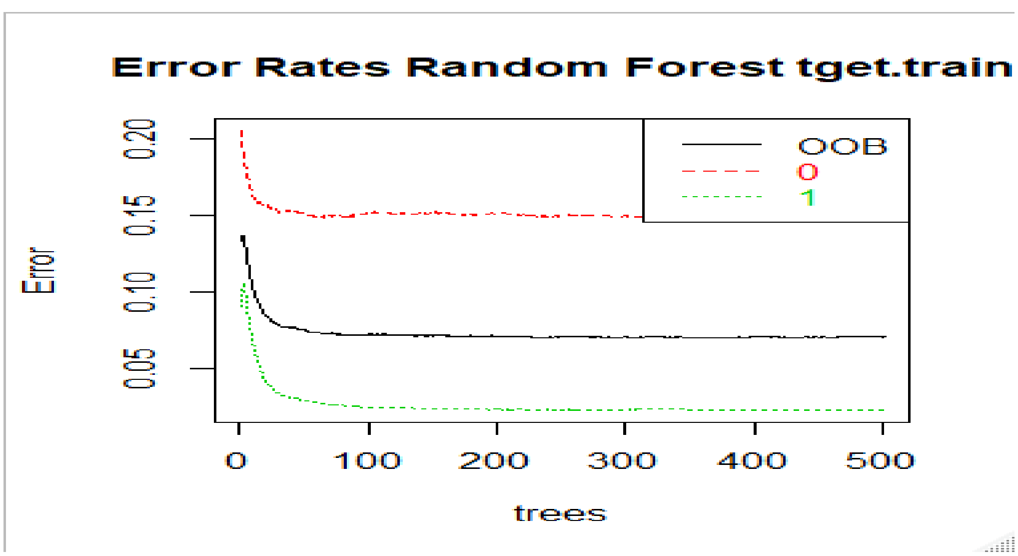
Appendix – 5: Random Forest Model

Lets build our first random forest

```
#change name of columns
names(smote.train)[2]=paste("Marital_Status")
names(smote.train.base)[2]=paste("Count_3_6_months_late")
names(smote.train.base)[3]=paste("Count_6_12_months_late")

#generate random forest

rndFor = randomForest(renewal ~ ., data = smote.train.base,
            ntree= 501, mtry = 3, nodesize = 10,
            importance=TRUE)

#print random forest
rndFor
```

```
Call:
 randomForest(formula = renewal ~ ., data = smote.train.base,                 ntree = 501, mtry = 3,
nodesize = 10, importance = TRUE)
                 Type of random forest: classification
                       Number of trees: 501
No. of variables tried at each split: 3

        OOB estimate of    error rate: 7.05%
Confusion matrix:
      0     1 class.error 0
17712    3138    0.1505036
1     782 33968    0.0225036
```

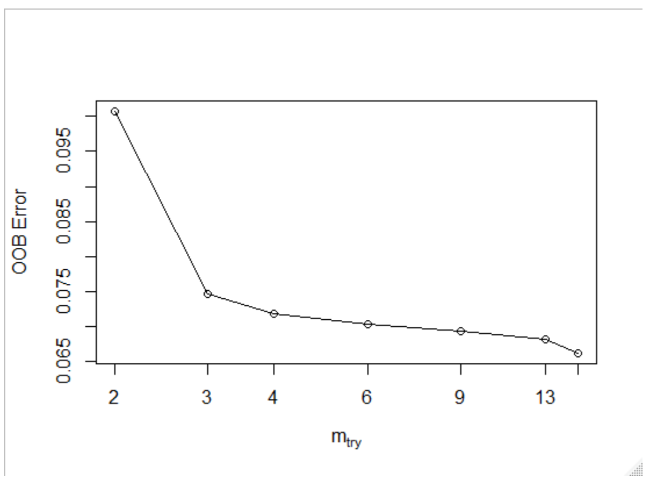
```
# plot error rates for train data

plot(rndFor, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest tget.train")
```



Above chart confirms that 50 trees is a reasonable good assumption as error rate decrease is minimal or absent post that value. So we will assume odd value of 51 trees to confirm with the majority rule application.

#Now we will "tune" the Random Forest by trying different m values. We will stick with 51 trees (odd number of trees are prefe   rable). The returned forest, "tRndFor" is the one corresponding to the best m

```
tRndFor = tuneRF(x = smote.train.base[,c(-14)],
+                     y=smote.train.base$renewal,
+                     mtryStart = 3,
+                     ntreeTry = 51,
+                     stepFactor = 1.5,
+                     improve = 0.0001,
+                     trace=TRUE,
+                     plot = TRUE,
+                     doBest = TRUE,
+                     nodesize = 10,
+                     importance=TRUE
+ )
mtry = 3    OOB error = 7.46%
Searching left ...
mtry = 2         OOB error = 10.08%
-0.3506869 1e-04
Searching right ...
mtry = 4         OOB error = 7.19%
0.03615329 1e-04
mtry = 6         OOB error = 7.04%
0.02050513 1e-04
mtry = 9         OOB error = 6.94%
0.01531785 1e-04
mtry = 13        OOB error = 6.82%
0.01711174 1e-04
mtry = 15        OOB error = 6.62%
0.02927987 1e-04
```



Mtry value of 15 has the least OOB error

```
importance(tRndFor)
```

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
Income	67.92898	105.65886	121.16545	844.24048
Count_3_6_months_late	195.92048	921.62423	749.59459	7764.73709
Count_6_12_months_late	170.11518	599.64333	616.57350	4979.45340
Count_more_than_12_months_late	108.52080	413.64473	441.24783	1320.08876
Marital_Status	53.36151	43.28856	60.64441	106.31004
Veh_Owned	69.27497	50.96418	73.03585	192.21885
No_of_dep	78.05300	52.09450	71.12222	359.65090
Accomodation	52.84058	37.54377	55.95264	97.83341
risk_score	96.38161	136.30285	161.52780	1026.61749
no_of_premiums_paid	105.07042	92.11541	101.91274	2136.61517
sourcing_channel	61.16836	61.28799	67.61055	414.77392
residence_area_type	47.26433	33.11641	50.84177	96.63796
premium	70.53807	74.12498	77.39822	1544.89556
cashPercent	217.29748	185.82308	269.44731	1902.49864
age	127.74839	132.73797	169.25447	1015.79921

Lets make predictions and measure the prediction error rate.

```
names(smote.test.base)[2]=paste("Count_3_6_months_late")
names(smote.test.base)[3]=paste("Count_6_12_months_late")
smote.test.base$predict.class = predict(tRndFor, smote.test.base[,c(-14)], type="class")
smote.test.base$prob1 = predict(tRndFor, smote.test.base[,c(-14)], type="prob")[,"1"]
tbl=table(smote.test.base$renewal, smote.test.base$predict.class)
tbl
```

```
        0     1
  0 14161  2592
  1   554 14676
```

Classification Error Rate (CER) = (554+2592) / 31983 = 0.10

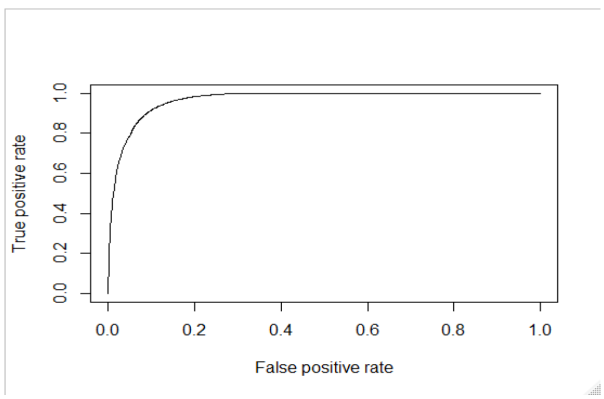
```
accuracy.rf<-sum(diag(tbl))/sum(tbl)
accuracy.rf
[1] 0.9016352
```

```
loss.rf<-tbl[2,1]/(tbl[2,1]+tbl[1,1]) loss.rf
[1] 0.03764866
```

```
opp.loss.rf<-tbl[1,2]/(tbl[1,2]+tbl[2,2]) opp.loss.rf
[1] 0.1501042
```

```
tot.loss.rf<-0.95*loss.rf+0.05*opp.loss.rf tot.loss.rf
[1] 0.04327144
```

```
predObj = prediction(smote.test.base$prob1, smote.test.base$renewal) perf =
performance(predObj, "tpr", "fpr")
plot(perf)
```



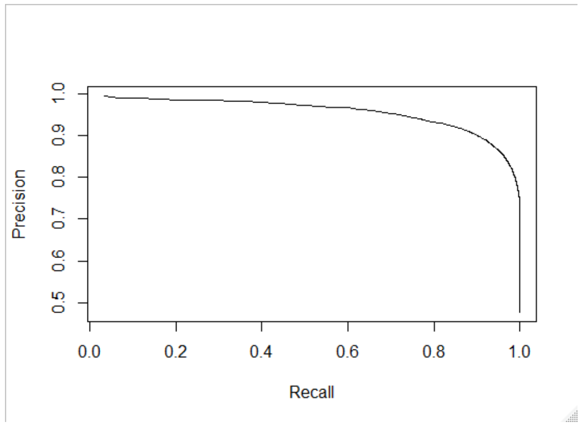

```
KS = max(perf@y.values[[1]]-perf@x.values[[1]]) KS
[1] 0.8186116
```

```
auc = performance(predObj,"auc");
auc = as.numeric(auc@y.values) auc
[1] 0.9679322
```

```
gini = ineq(smote.test.base$prob1, type="Gini") gini
[1] 0.4597775
```

```
#Precision-recall curve
```

```
PRcurve(smote.test.base$prob1,smote.test.base$renewal)
```



Appendix – 6: Bagging

```
library(ipred)
library(rpart)
German.bagging <- bagging(renewal ~., data= smote.train.base,control=rpart.control(maxdep th=5, minsplit=4))

smote.test.base$pred.class <- predict(German.bagging, smote.test.base[-14])


tab.baggin = table(smote.test.base$renewal, smote.test.base$pred.class)

            0      1
  0 12331
           442
```

Classification Error Rate (CER) = (1071+4422) /31983 = 0.17

```
accuracy.bagging<-sum(diag(tab.bagging))/sum(tab.bagging)
accuracy.bagging
[1] 0.8282525

loss.bagging<-tab.bagging[1,2]/(tab.bagging[1,2]+tab.bagging[1,1]) loss.bagging
[1] 0.2639527

opp.loss.bagging<-tab.bagging[2,1]/(tab.bagging[2,1]+tab.bagging[2,2]) opp.loss.bagging
[1] 0.07032173

tot.loss.bagging<-0.95*loss.bagging+0.05*opp.loss.bagging
tot.loss.bagging
[1] 0.2542712
```