FINAL REPORT

Rajeev Nitnawre

2/1/2021

Project Report - Insurance Premium Default Propensity Prediction

This report will contain: **1. Executive summary : problem statement, brief description of methods & final insights & recommendations 2. Approach : Logical steps to final model selection 3. Relevance and implementability of the conclusions and recommendations**

1. Executive Summary

1.1 Project Objective:

Premium paid by the customer is the major revenue source for insurance companies. Default in premium payments results in significant revenue losses and hence insurance companies would like to know upfront which type of customers would default premium payments.

The objective of this project

- Build a model that can predict the likelihood of a customer defaulting on premium payments (Who is likely to default)
- Identify the factors that drive higher default rate (Are there any characteristics of the customers who are likely to default?)
- Propose a strategy for reducing default rates by using the model and other insights from the analysis (What should be done to reduce the default rates?)

Problem Statement

“Explore the data to identify customers with the propensity to default on the premiums to be paid to the Insurance company. Identify the factors/ characteristics influencing these default behavior to predict the probability of **who** will default, so as to help the Insurance Agents to proactively reach out these customers with the right strategy/approach to make sure they pay their due premiums”

Need of the present study and business/social opportunity

- The current pandemic situation engulfing the world and the impact in which we all are struggling socially, psychologically & economically has taken a Physical and Mental Impact
- One of the major impact of the COVID 19 has been the economic fallout, closure of businesses , trade , commerce and the corresponding job loss.
- These impact especially the influx of income/salaries has affected the purchase of Life & Health insurance policy and tendency to default on the premium payment in salaried Generation X.
- The impact of being infected by the CoVid 19 virus and the medical and hospitalization expenses involved could be very challenging at such times especially for the elderly who have comparatively weaker immunity to fight this virus. + One need to be equipped with the right medical insurance at such times and the need to have a medical insurance and also you life covered with Life Insurance is of paramount importance.
- Defaulting on Premiums could cost one dearly when facing such medical emergencies. It becomes a social responsibility to make people aware that trying to default on premiums because of the

challenging economic constrains will be a short term action which could have a long term impact on the health/wealth condition of a individual and his/her dependents. This important message of how having your insurance active by paying premiums on time could be a life saver for all concerned, needs to be driven home.

1.2 Brief description of methods

1. The data provided was explored in detail to understand the nature of the various independent variables and with reference to the defendant variable (i.e.Default). The intention was to try and explore the relations and influence that these variables eventually have on the potential customers propensity to default on his insurance premiums.

2. After a sanitary check was conducted on the data-set for checking on the nature of the variables, missing values (if any), etc we conducted the

- **Univariant Analysis** to discover that 6.26% of the customers have defaulted in their payment of insurance premium.
- The categorical and numerical variables were explored to check on aspects like the mean, median, mode, outliers and other influencing aspect of each variable individually.
- **Bi Variant Analysis** was conducted to see how the dependent variable (i.e. Defaulter) was behaving amidst all the categorical and numerical variables.
- The distribution of the Defaulter & Non-Defaulters in the data and each variable were identified and noted especially for its concentration or the lack of it.
- The outliers in each variable were explored and noted for further treatment & exploration.
- **Correlation** between each vertical was checked using correlation chart and Density Plot to understand the degree of correlation and if they would be any high correlations which we need to consider for our analysis.
- These analysis gave us a indication of how unbalanced the data is and how much of an impact each variable is having on our dependent variable. This gave us an idea of how we further treat this data to explore it further and what techniques and algorithmic models we need to deploy to investigate the potential customers who may have the propensity to default looking at the trends, behavior, habits, patterns that we discover in our analysis.

3. **Pre-processing the data** and preparing it for further analysis

- The data set was then pre-processed by employing techniques like **Outlier Treatment**,**Missing Value Treatment**,**Variable Transformation**, **Removal of unwanted variable**, **Change Variable Name** & **Adding new variable**.
- The processed data was then explored once again to get a clearer and precise understanding of what we need to look at to investigate further in our quest to identify the potential premium defaulters. Once again we used **Univariant Analysis**,**Bi Variant Analysis**, **Correlation** and analyzed the Categorical variable for correlation with a Hypothesis testing using the **Chi Square test**.

4. Alternative Analytical Approach:

- Incase any variables were found to be highly correlated, they would be dropped from the data set as they will wrongly influence the model build ahead for analysis

- Partition the data into **train and test data set**.

    - Ensure the target variable in the data is a factor variable
    - Ensure the levels of the target variable are correct
    - Split the data set into a train and test set and ensure the distribution of the dependent variable is similar in the train and test data as in the original data set

Model Building Approach:

The approach here was to build various models and compare attributes like Accuracy, Sensitivity, Specificity, ROC curve, AUC, Gini, KS of the Training set with the Test set to determine which model will come closest to predict the potential defaulters.

SMOTE the trained data-set to handle the class unbalanced classification in the data set.

The models we built to analyze were:

1. Model 1 - **Simple Logistic Model** Logistic regression is a statistical model that uses Logistic function to model the conditional probability. The probability will always range between 0 and 1. In the case of binary classification the probability of defaulting premiums and not defaulting premiums will sum up to 1
2. Model 2 - **Naïve Bayes** Naïve Bayes is a classification method based on Bayes' theorem that derives the probability of the given feature vector being associated with a label. Naïve Bayes has a naive assumption of conditional independence for every feature, which means that the algorithm expects the features to be independent which not always is the case.
3. Model 3 - **KNN** KNN algorithms use data and classify new data points based on similarity measures. Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors. As you increase the number of nearest neighbors, the value of k, accuracy might increase.
4. Model 4 - **CART MODEL (Decision Tree)** Decision tree learning is a supervised machine learning technique for inducing a decision tree from training data. A decision tree is a predictive model which is a mapping from observations about an item to conclusions about its target value.
    - Built a CART model on the train data. Use the "rpart" and the "rattle" libraries to build decision tree. – create CART model 1 & validate for accuracy – Tuning the model: further tune the model for further accuracy – Model Validation: validate the new model – Model Evaluation: evaluate both the models on the test data & compare their accuracy.
    - Tune the model and prune the tree, if required.
    - Test the data on test set.
5. Model 5 - **Random Forest** RF classifier is an ensemble method that trains several decision trees in parallel with bootstrapping followed by aggregation, jointly refreed to as Bagging. – Incase there is no significant improvement in the CART model from the baseline model, we build the Random Forest. – Tune the Model – Model Validation: validate the new model – Model Evaluation: evaluate both the models on the test data & compare their accuracy.
6. Model 6 - **Gradient Boosting Machines** Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error.
7. Model 7 - **Xtreme Gradient Boosting** Extreme Gradient Boosting (XGBoost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. What makes it fast is its capacity to do parallel computation on a single machine.

- Compared all the model to decide which model for predicting defaulters most accurately
- Identify the **Important variables** used in the final selected model.
- Checked the Important variables which would help strategies how we can reduce the default rates by identifying the factors that drive them.
- Proper predictive model evaluation was important because we want our model to have the same predictive ability across many different data sets.
- It is important to note here that accuracy is not always the best metric to compare predictive models.

- We tried to figure out what would be the metrics of choice to evaluate a predictive model for identifying the potential defaulters for the Insurance company.

The aim was to create the best model to predict & identify the cohorts who are likely to default If we are able to predict the defaulters we will be able to achieve the goal. With this goal in mind, we can understand that the model that has the highest sensitivity (ability to predict the true positives) would be the best model. Model sensitivity can be improved by changing the probability threshold and ROC Curves are very helpful in that.

Hence we will use our best model and use the ROC to identify the likely defaulters. This will also narrow down on the strategy we need to deploy to address this cohort of potential defaulters by looking at their characteristics which could be the factors that drive high defaults.

1.3 Final Insight:

CART Model 1 emerges as the better models to work upon.

- On the Specificity, we see that the CART Model 1 performs better than all the other models and has a edge over the Random Forest Model.
- When we look at the Sensitivity of the CART models it perform best among all the models.

CART Model 1 will be most efficient in identifying the defaulters.

Business Insights & Recommendations:

- Having worked out ways to identify the cohort of customers who could have the prophecy to defaulter, we will need to dive a bit deeper into our findings to discover the patterns and behaviors of these customers. This will be important because we will eventually need to address these issues and provide easy to follow options and solutions to push them to pay thier premiums on time.
- Our findings show that some aspects are prominent amongst these cohorts of customers, namely
    - The customers who pay their premium late by 6 to 12 months
    - Number of Dependents - more the number of dependents make it difficult to manage premiums
    - Risk Score - customers with low risk score tend to default
    - Premiums paid is a good indicator
    - Number of premiums paid also is a good indicator We will need to identify the mix among these factors to understand what would be the options we could work out of these groups.

Recommendations:

- We need to keep in mind the current economic fallout due to the pandemic and the impact it has had on jobs/ business/income when we come our with solutions for the payments of the premiums. This could be in the lines of:
- Creating multiple options in paying premiums
- Multiple bouquets or sachet options for easy buy for specific plan or specfic period
- Options to pay premiums at various time plans like bi-annual, quaterly, monthly etc. depending on the kind of customer we target to address.
- We could approach the insurance company to get some more information to further distill and identify the cohorts with reference to some information which may not be provided in the shared data. This coulld include informations ike Gender, Type of Insurance (purchased/ defaulted), frequency of the premiums to be paid, etc.

2. Approach : Logical steps to final model selection

```
knitr::opts_chunk$set(error = FALSE,     # suppress errors
                      message = FALSE,   # suppress messages
                      warning = FALSE,   # suppress warnings
                      echo = FALSE,      # suppress code
                      cache = TRUE)      # enable caching
```

2.1 Environment Set up and Data Import

2.1.1 Install necessary Packages and Invoke Libraries

2.1.2 Import and Read the Dataset

2.1.3 Clear Description of the attributes (Name of columns) in the data set

The dataset contains the following information about 79854 policy holders:

1. id: Unique customer ID
2. perc_premium_paid_by_cash_credit: What % of the premium was paid by cash payments?
3. age_in_days: age of the customer in days
4. Income: Income of the customer
5. Marital Status: Married/Unmarried, Married (1), unmarried (0)
6. Veh_owned: Number of vehicles owned (1-3)
7. Count_3-6_months_late: Number of times premium was paid 3-6 months late
8. Count_6-12_months_late: Number of times premium was paid 6-12 months late
9. Count_more_than_12_months_late: Number of times premium was paid more than 12 months late
10. Risk_score: Risk score of customer (similar to credit score)
11. No_of_dep: Number of dependents in the family of the customer (1-4)
12. Accommodation: Owned (1), Rented (0)
13. no_of_premiums_paid: Number of premiums paid till date
14. sourcing_channel: Channel through which customer was sourced
15. residence_area_type: Residence type of the customer
16. premium : Total premium amount paid till now
17. default: Y variable - 0 indicates that customer has defaulted the premium and 1 indicates that customer has not defaulted the premium
18. agegroup: Age grouped in 8 buckets for a more descriptive analysis of Age vs various other variants- (later added to the data set)

Observation:

- The identity of the customers does not reveal the geographical or geographical characteristics.
- There is no Gender characteristic provided for analyzing the sex of the customers involved
- The age is displayed in “Number of days” which isn’t the normally used to calculate age. We would change this to “Number of Years” to easy analysis.
- There is no time period that is specified for this data, hence some assumptions will need to be made.
- Details of the Sourcing Channels mentioned could give more insight on the type of sourcing which could be considered while analysing the data and introspecting to look into for more insights.
- When we take an overview of the Age and the Number of Premiums Paid, we can introspect if there is any relation with the more number of premiums to age or not.

2.1.4 Variable Identification

Structure of Dataset

VIEW PLOT: The plot structure reveals the various variables of the data frame

Check the dimension of the dataset

```
## [1] 79853    17
```

DIMENSIONS: shows Columns = 17 and Rows = 79, 853

Sanity Checks

id <dbl>	perc_premium_paid_by_cash_credit <dbl>	age_in_days <dbl>	Inco… <dbl>	Count_3-6_months_late <dbl>
1	0.317	11330	90050	0
2	0.000	30309	156080	0
3	0.015	16069	145020	1
4	0.000	23733	187560	0
5	0.888	19360	103050	7
6	0.512	16795	113500	0

6 rows | 1-5 of 17 columns

id <dbl>	perc_premium_paid_by_cash_credit <dbl>	age_in_days <dbl>	Inco… <dbl>	Count_3-6_months_late <dbl>
79848	0.009	21545	133140	0
79849	0.249	25555	64420	0
79850	0.003	16797	660040	1
79851	0.012	24835	227760	0
79852	0.190	10959	153060	1
79853	0.000	19720	324030	0

6 rows | 1-5 of 17 columns

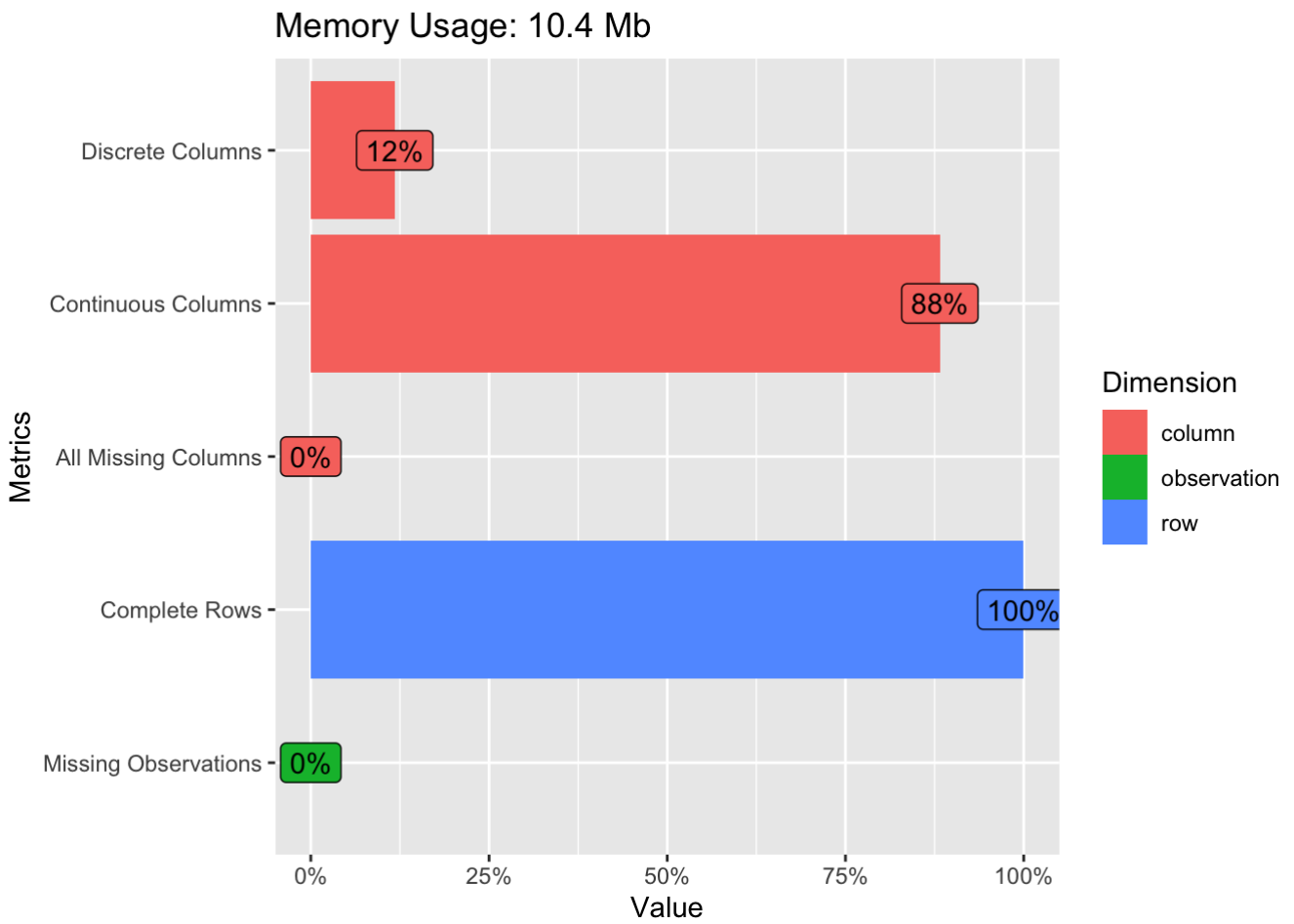
HEAD & TAIL: A uniform similarity is observed in the data frame's columns and rows using the "head' &"tail" function.

Check the structure of dataset

```
## tibble [79,853 × 17] (S3: tbl_df/tbl/data.frame)
##  $ id                                : num [1:79853] 1 2 3 4 5 6 7 8 9 10 ...
##  $ perc_premium_paid_by_cash_credit: num [1:79853] 0.317 0 0.015 0 0.888 0.512 0
0.994 0.019 0.018 ...
##  $ age_in_days                       : num [1:79853] 11330 30309 16069 23733 19360
...
##  $ Income                            : num [1:79853] 90050 156080 145020 187560 1030
50 ...
##  $ Count_3-6_months_late             : num [1:79853] 0 0 1 0 7 0 0 0 0 0 ...
##  $ Count_6-12_months_late            : num [1:79853] 0 0 0 0 3 0 0 0 0 0 ...
##  $ Count_more_than_12_months_late    : num [1:79853] 0 0 0 0 4 0 0 0 0 0 ...
##  $ Marital Status                    : num [1:79853] 0 1 0 1 0 0 0 0 1 1 ...
##  $ Veh_Owned                         : num [1:79853] 3 3 1 1 2 1 3 3 2 3 ...
##  $ No_of_dep                         : num [1:79853] 3 1 1 1 1 4 4 2 4 3 ...
##  $ Accomodation                      : num [1:79853] 1 1 1 0 0 0 1 0 1 1 ...
##  $ risk_score                        : num [1:79853] 98.8 99.1 99.2 99.4 98.8 ...
##  $ no_of_premiums_paid               : num [1:79853] 8 3 14 13 15 4 8 4 8 8 ...
##  $ sourcing_channel                  : chr [1:79853] "A" "A" "C" "A" ...
##  $ residence_area_type               : chr [1:79853] "Rural" "Urban" "Urban" "Urban"
...
##  $ premium                           : num [1:79853] 5400 11700 18000 13800 7500 330
0 20100 3300 5400 9600 ...
##  $ default                           : num [1:79853] 1 1 1 1 0 1 1 1 1 1 ...
```

STRUCTURE OF DATASET: There are some variables in the data set which are numerical in nature whose format needs to be changed for proper analysis.

Introducing Plot Dimensions



PLOT DIMENSIONS:

– The plot intro shows 12% of the columns are discrete in nature while 88% are continuous. This will change as the formats of some variables will be changed for analysis. – There are no missing columns or rows and no missing observations, which indicates the data is uniform and complete with no undesired discrepancies.

Get Summary of the dataset

```
##        id        perc_premium_paid_by_cash_credit  age_in_days
##  Min.   :    1   Min.   :0.0000                   Min.   : 7670
##  1st Qu.:19964   1st Qu.:0.0340                   1st Qu.:14974
##  Median :39927   Median :0.1670                   Median :18625
##  Mean   :39927   Mean   :0.3143                   Mean   :18847
##  3rd Qu.:59890   3rd Qu.:0.5380                   3rd Qu.:22636
##  Max.   :79853   Max.   :1.0000                   Max.   :37602
##      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
##  Min.   : 0.00000               Min.   :0.0000   Min.   :1.000
##  1st Qu.: 0.00000               1st Qu.:0.0000   1st Qu.:1.000
##  Median : 0.00000               Median :0.0000   Median :2.000
##  Mean   : 0.05994               Mean   :0.4987   Mean   :1.998
##  3rd Qu.: 0.00000               3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :11.00000               Max.   :1.0000   Max.   :3.000
##    No_of_dep      Accomodation      risk_score    no_of_premiums_paid
##  Min.   :1.000   Min.   :0.0000   Min.   :91.90   Min.   : 2.00
##  1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:98.83   1st Qu.: 7.00
##  Median :3.000   Median :1.0000   Median :99.18   Median :10.00
##  Mean   :2.503   Mean   :0.5013   Mean   :99.07   Mean   :10.86
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:99.52   3rd Qu.:14.00
##  Max.   :4.000   Max.   :1.0000   Max.   :99.89   Max.   :60.00
##  sourcing_channel   residence_area_type    premium         default
##  Length:79853       Length:79853        Min.   : 1200   Min.   :0.0000
##  Class :character   Class :character    1st Qu.: 5400   1st Qu.:1.0000
##  Mode  :character   Mode  :character    Median : 7500   Median :1.0000
##                                         Mean   :10925   Mean   :0.9374
##                                         3rd Qu.:13800   3rd Qu.:1.0000
##                                         Max.   :60000   Max.   :1.0000
```

```
##  [1] "id"                                "perc_premium_paid_by_cash_credit"
##  [3] "age_in_days"                       "Income"
##  [5] "Count_3-6_months_late"             "Count_6-12_months_late"
##  [7] "Count_more_than_12_months_late"    "Marital Status"
##  [9] "Veh_Owned"                         "No_of_dep"
## [11] "Accomodation"                      "risk_score"
## [13] "no_of_premiums_paid"               "sourcing_channel"
## [15] "residence_area_type"               "premium"
## [17] "default"
```

SUMMARY OF THE DATASET:

– The summary shows variables namely Marital Status, Vehicles Owned, Number of Dependents, Accommodations & Default will need to changed to factors for a correct representation of what the data is displaying. – Age is also displayed in 'Days" which need to be changed to 'Years" – After changing the characters of the mentioned variables, we will further explore the summary in detail.

2.1.5.a Variable Transformation- Convert "Age in Days" variable to "Age" - in years

VARIABLE TRANSFORMATION: The "Age in Days" column is used to create a new variable column of 'Age" which is represented in years.

2.1.5.b Variable Transformation- dividing "Income" by 1000

2.1.6.a Addition of New variables - Adding a new variable 'agegroup" to bucket the age in groups

ADDING A NEW VARIABLE - "agegroup": The "age" - in years is used to create a new feature variable i.e. Age Group. The age component is slotted into eight buckets from lowest to highest in ascending order. The eight age groups are as follows

- 1 = Ages between 12 & 29
- 2 = Ages between 30 & 39
- 3 = Ages between 40 & 49
- 4 = Ages between 50 & 59
- 5 = Ages between 60 & 69
- 6 = Ages between 70 & 79
- 7 = Ages between 80 & 89
- 8 = Ages between 90 & 103

2.1.6.b Addition of New variables - Adding a new variable 'riskscore_group" to bucket the risk scroes in groups

ADDING A NEW VARIABLE - "riskscore_bins": The "risk_score" variable is used to create a new feature variable i.e. riskscore_bins. The risk scores component are slotted into nine buckets for exploring the impact of the risk scores to various cohorts.

1.5 Removal of unwanted variables

```
## [1] 79853    18
```

DROPPING THE UNWANTED VARIABLES FROM THE DATA SET:

Change name of variables

*Changing names of the variable with space or '-' to avoid errors while building models.

2.1.7 Outlier treatment

2.1.7.a. Age

2.1.7.b. Income

2.1.7.c.Premium

2.1.7.d. no_of_premiums_paid

2.1.7.e. Premium paid in Cash

2.1.7.f. 'Count_3-6_months_late'

2.1.7.g. Count_6-12_months_late

2.1.7.h. Count_more_than_12_months_late

OUTLIER TREATMENT FOR VARIABLES HAVING SIGNIFICANT AND EXTREME OUTLIERS:

- The variables identified with significant/extreme outliers are:
  - age
  - Income
  - premium
  - number of premiums paid
  - Count of premium paid late by 3 to 6 months
  - Count of premium paid late by 6 to 12 months
  - Count of premium paid late by more than 12 months
- The above variables were seen to have considerable outliers during the Uni Variant Analysis. The correlation of these variables is checked for the impact of the Outlier Treatment on these variables.The outlier treatments will prevent the outliers to influence the models to misinterpret the data.

2.1.8 Changing variable as factors

CHANGING VARIABLES AS FACTORS: The variables, namely Marital Status, Vehicles Owned, Number of Dependents, Accommodations & Default are appropriately changed to factors to make them discrete observations.

2.2 Univariant Analysis

2.2.a Distribution of the dependent variable

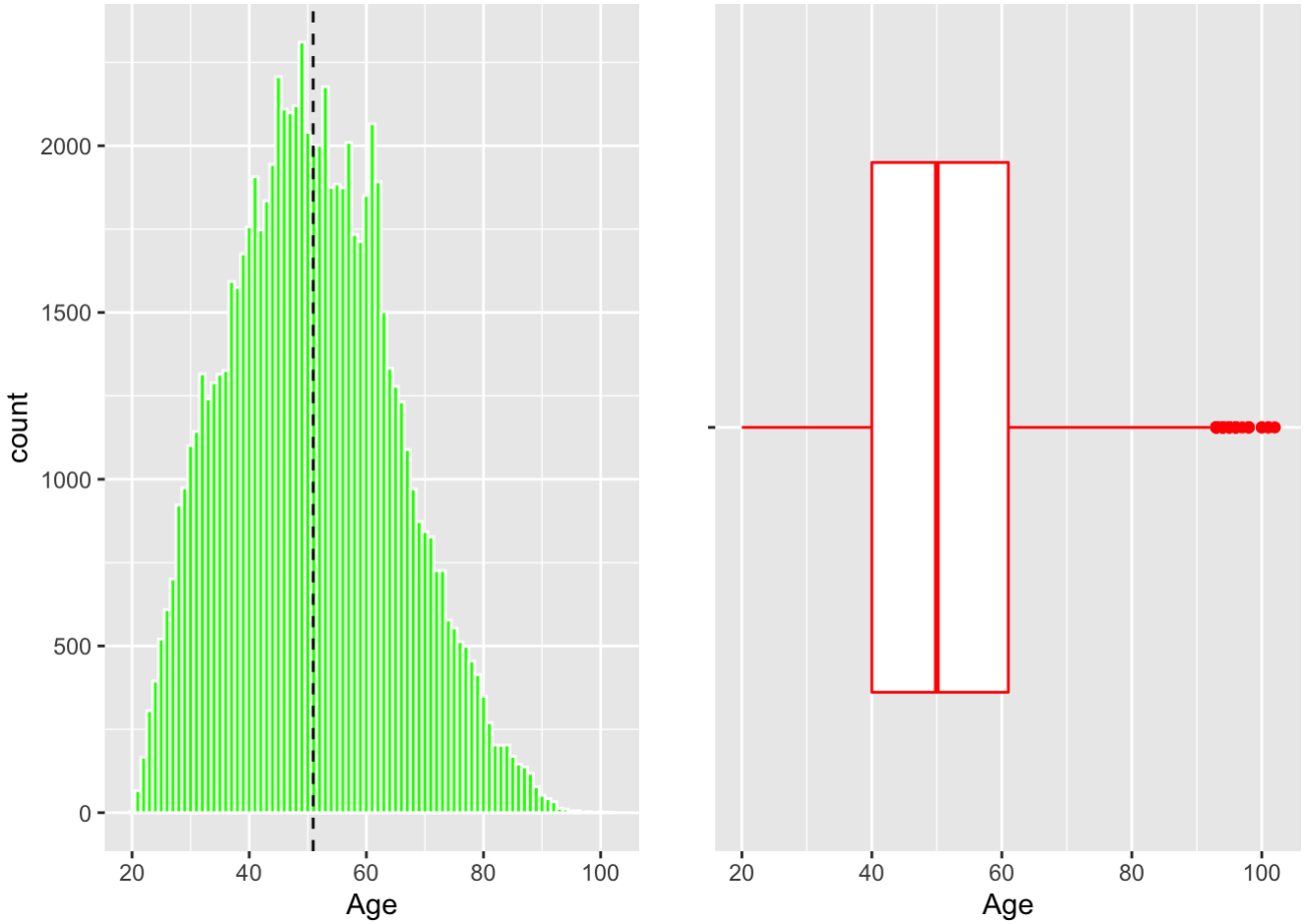
```
##
##          0          1
##  6.259001 93.740999
```

DEFAULT VARIABLE: The table split shows that 6.26% of the people have defaulted in their payment of insurance premium

2.2.b Function to draw histogram and boxplot of numerical variables using ggplot

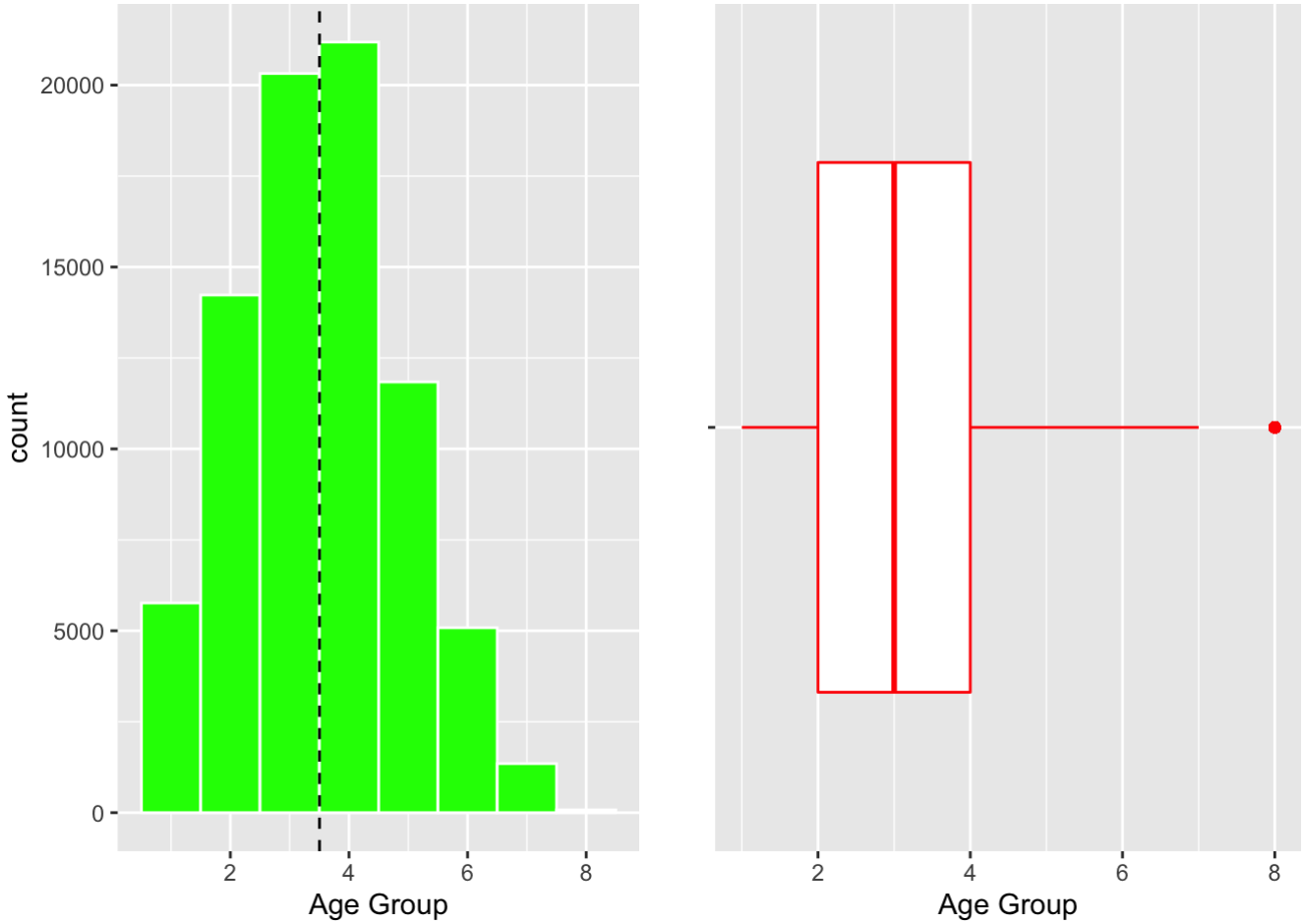
2.2.b Visualize properties of all categorical variables

1. Observations on Age:



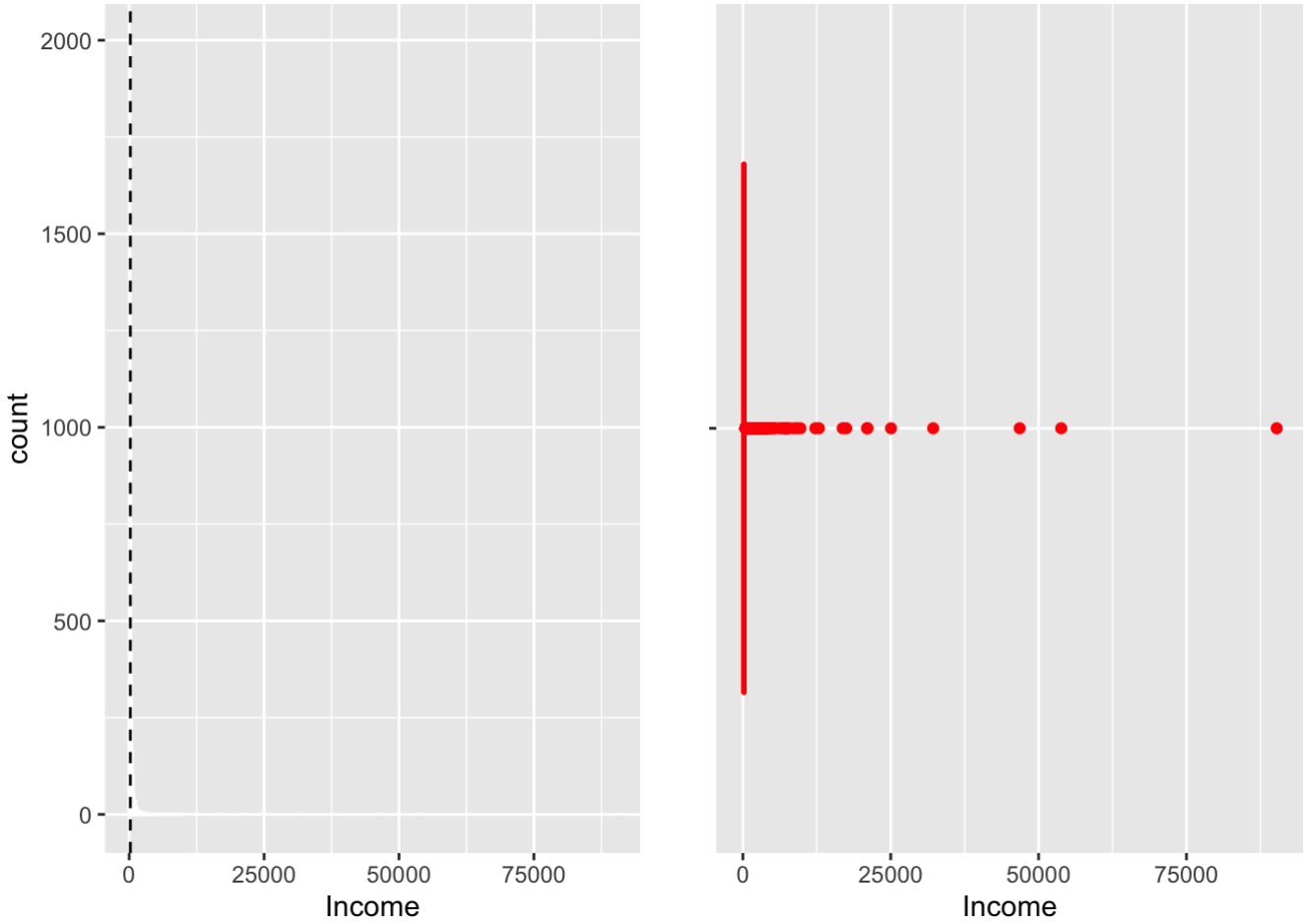
- There seems an almost normal distribution in the Age of the customers.
- The range is spread from 20 to 92 with approximately nine outliers between 93 and 102
- The range is concentrated around 40 years to 61 years
- Mean (50.91) & Median (50) both are'nt far apart suggesting there aren't many outliers influencing the mean.

2. Observations on Age Group:



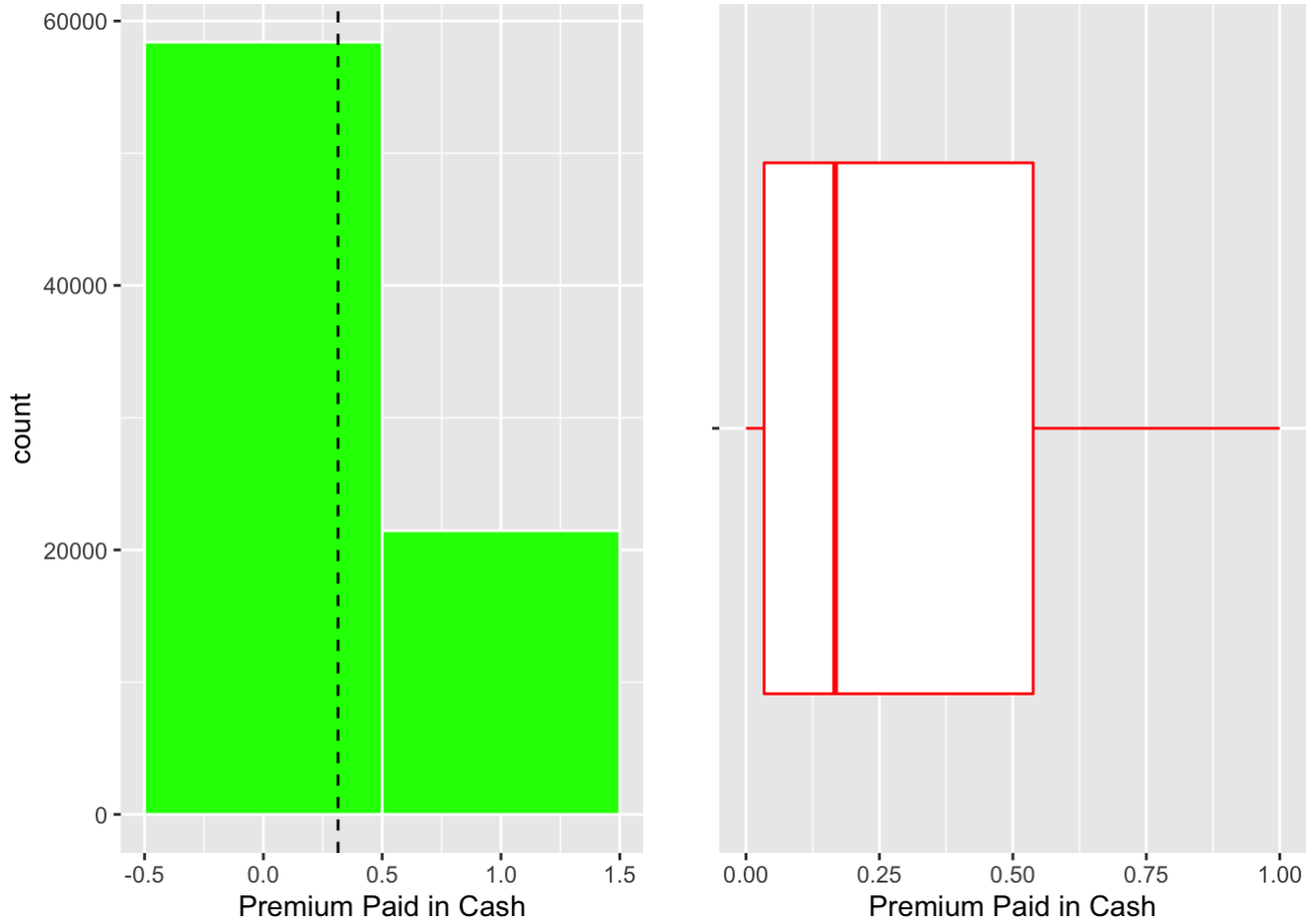
- Age Group 3 sees the maximum amount of individuals, followed by Group 4
- The median is at Group 3 which the mean is around 3.5 which shows
- The 3rd Quartile is at 4 while maximum stretched to 8 which shows maximum concentration between 2 & 4.

3. Observation on Income:



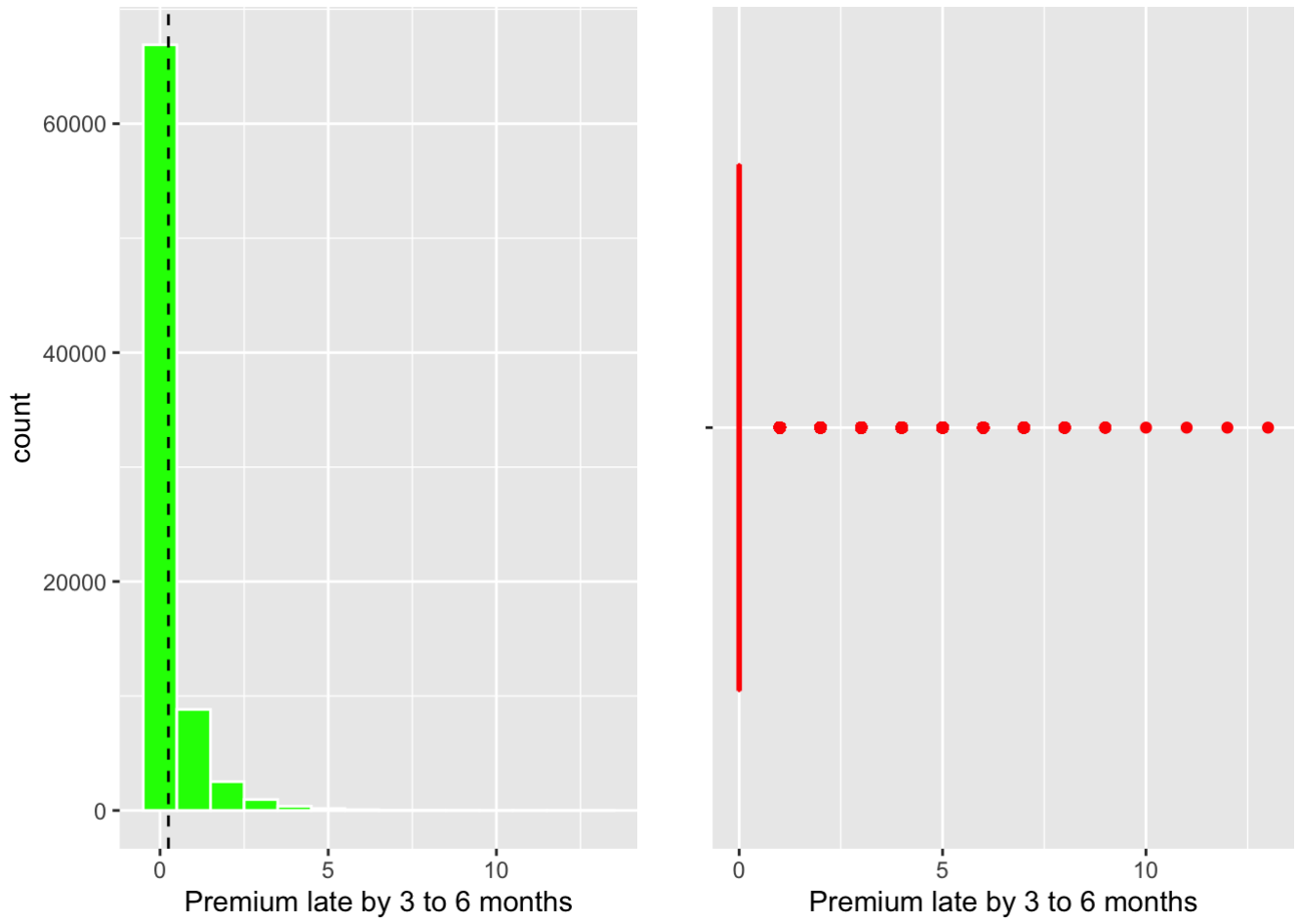
- The income range is very widely dispersed. The mean is 208,850 where the median is 166,560 which denotes some very extreme outliers are influencing the mean.
- There 3rd quartile is at 252,090 and the maximum stretches to 9,0262,600 which shows the income range is widely dispersed with some making very huge amount compared to the concentration which lie between 108,010 & 252,090

4. Observation on Premium paid in Cash:



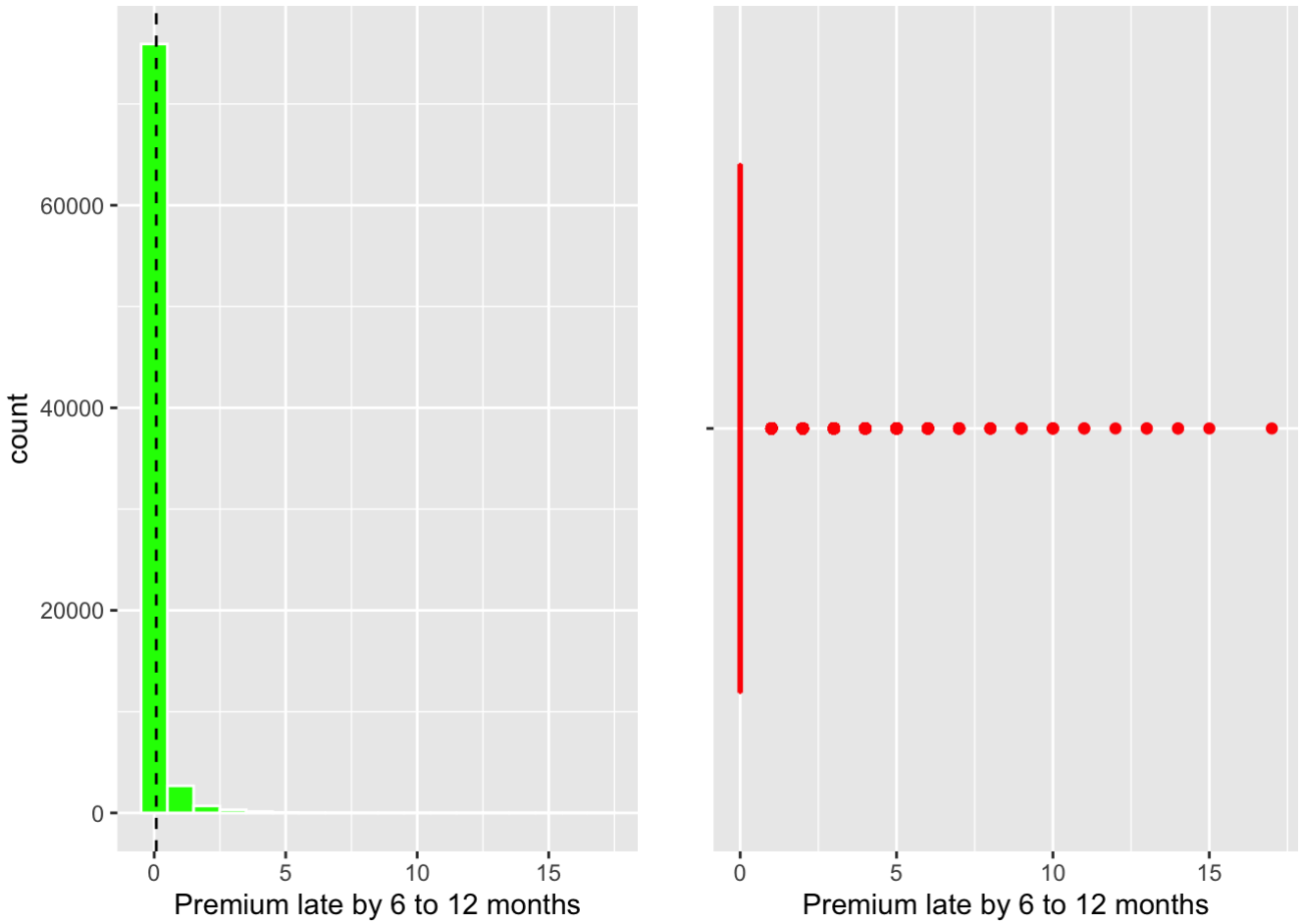
- The data seems right skewed (positive) with the bulk in the range of 3% to 53% around the median.
- The mean (31%) here is more than median (16.7%) suggesting outliers which are influencing the mean.

5 Observation on late payment of Premium by 3 to 6 months:



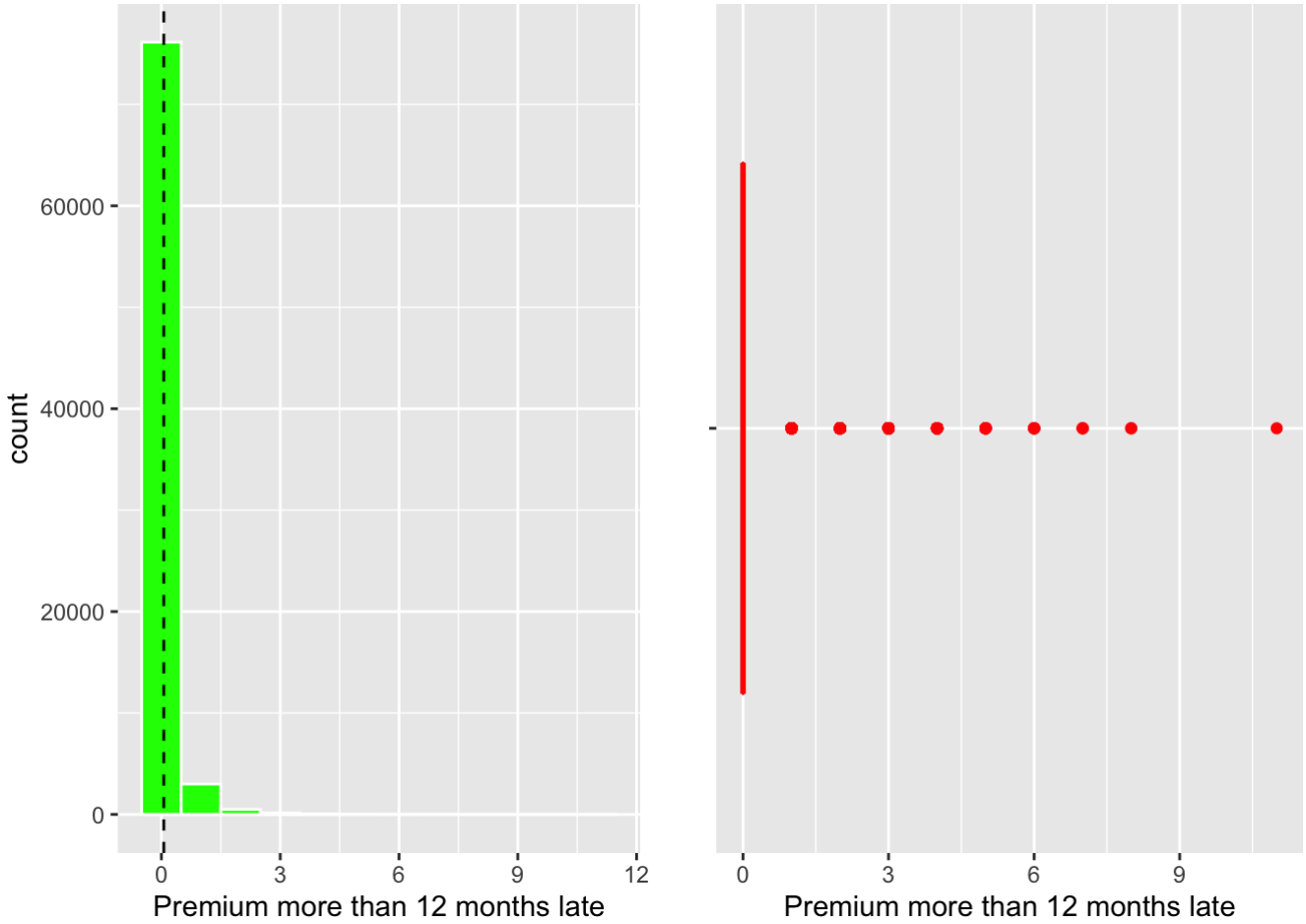
- Data suggest that there are very few who have paid their premiums late i.e. after 3 to 6 months.
- Infact the median and also the 3rd quartile is in 0 range.
- The mean is 0.24 which suggests outliers to a maximum of 13 number.
- Almost 84% of the customers have not paid their premiums late i.e. after 3 to 6 months.
- 11% have been late 1 time
- 35 have been late 2 times.
- 114 have been late more than 5 times and are major outliers here.

6.Observation on late payment of Premium by 6 to 12 months



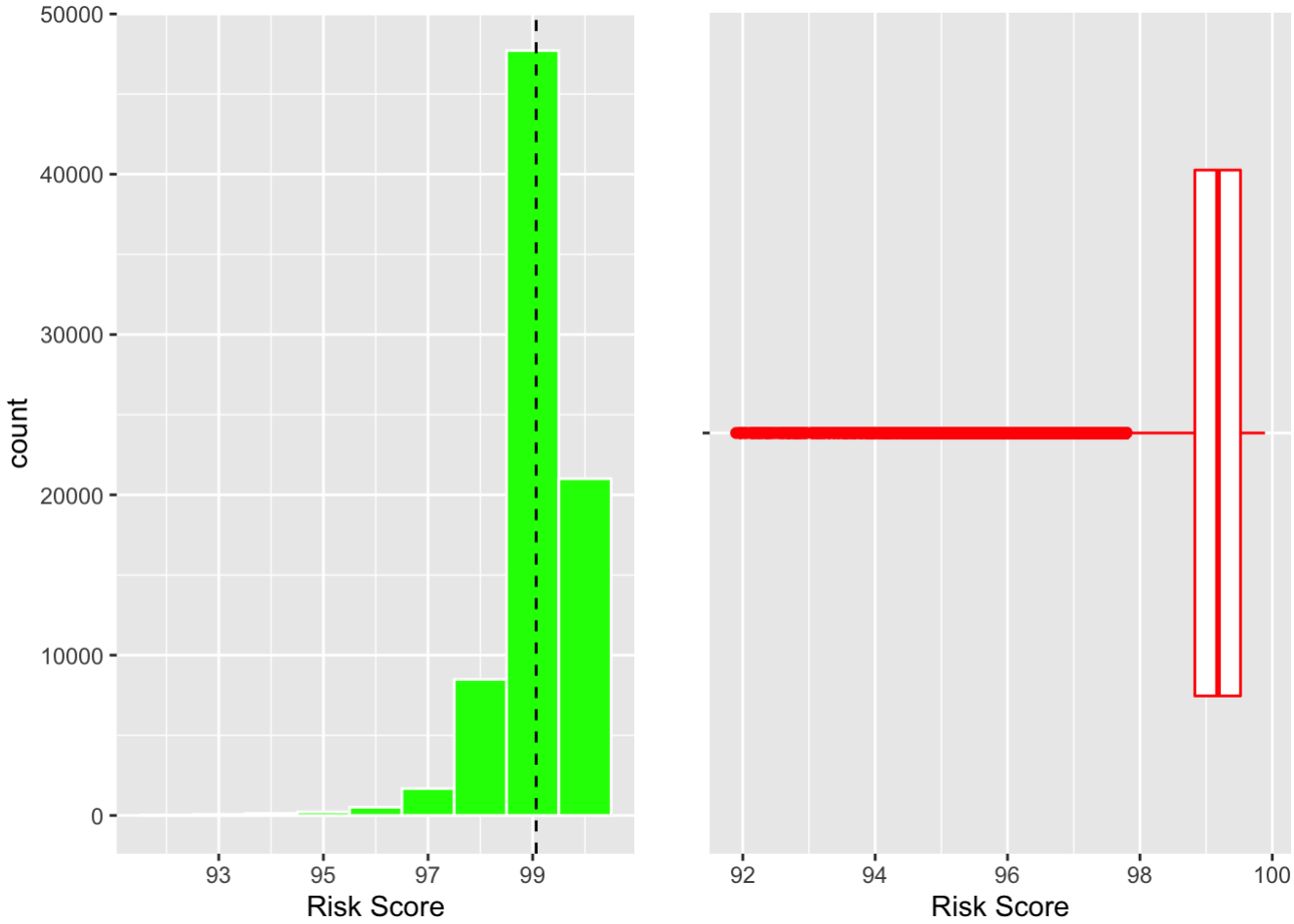
- Data suggest that there are very few who have paid their premiums late i.e. after 6 to 12 months.
- Infact the median and also the 3rd quartile is in 0 range.
- The mean is 0.07 which suggests outliers to a maximum of 17 number.
- More than 95% of the customers have not paid their premiums late i.e. after 6 to 12 months.
- 3.3% have been late 1 time
- 59 have been late more than 5 times and are major outliers here.

7. Observation on late payment of Premium by more than 12 months:



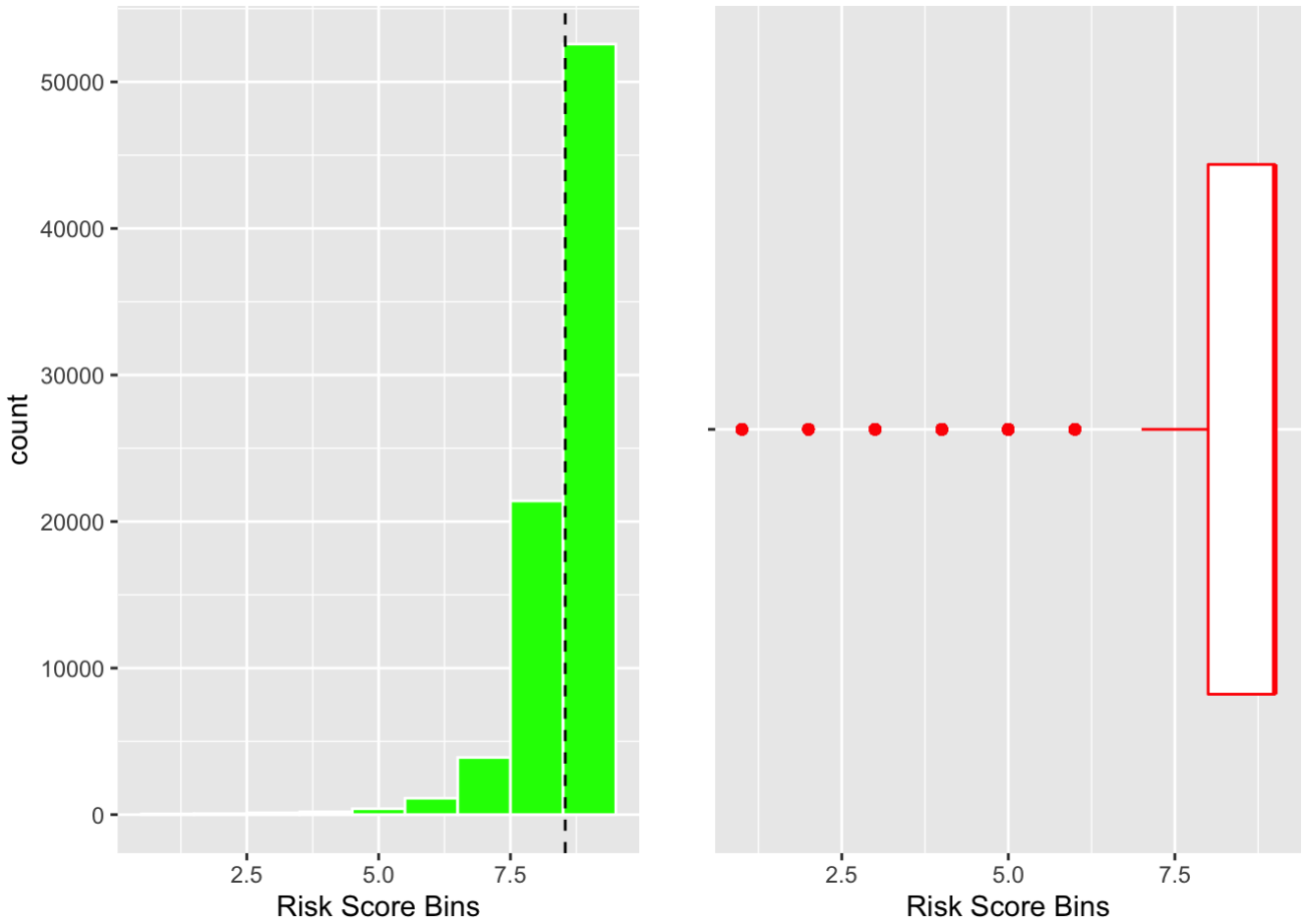
- Data suggest that there are very few who have paid their premiums later than 12 months
- Infact the median and also the 3rd quartile is in 0 range.
- The mean is 0.05 which suggests outliers to a maximum of 11 number.
- More than 95% of the customers have not paid their premiums late i.e. after 6 to 12 months.
- 3.7% have been late 1 time
- 12 have been late more than 5 times and are major outliers here.

8. Observation on Risk score of customers:



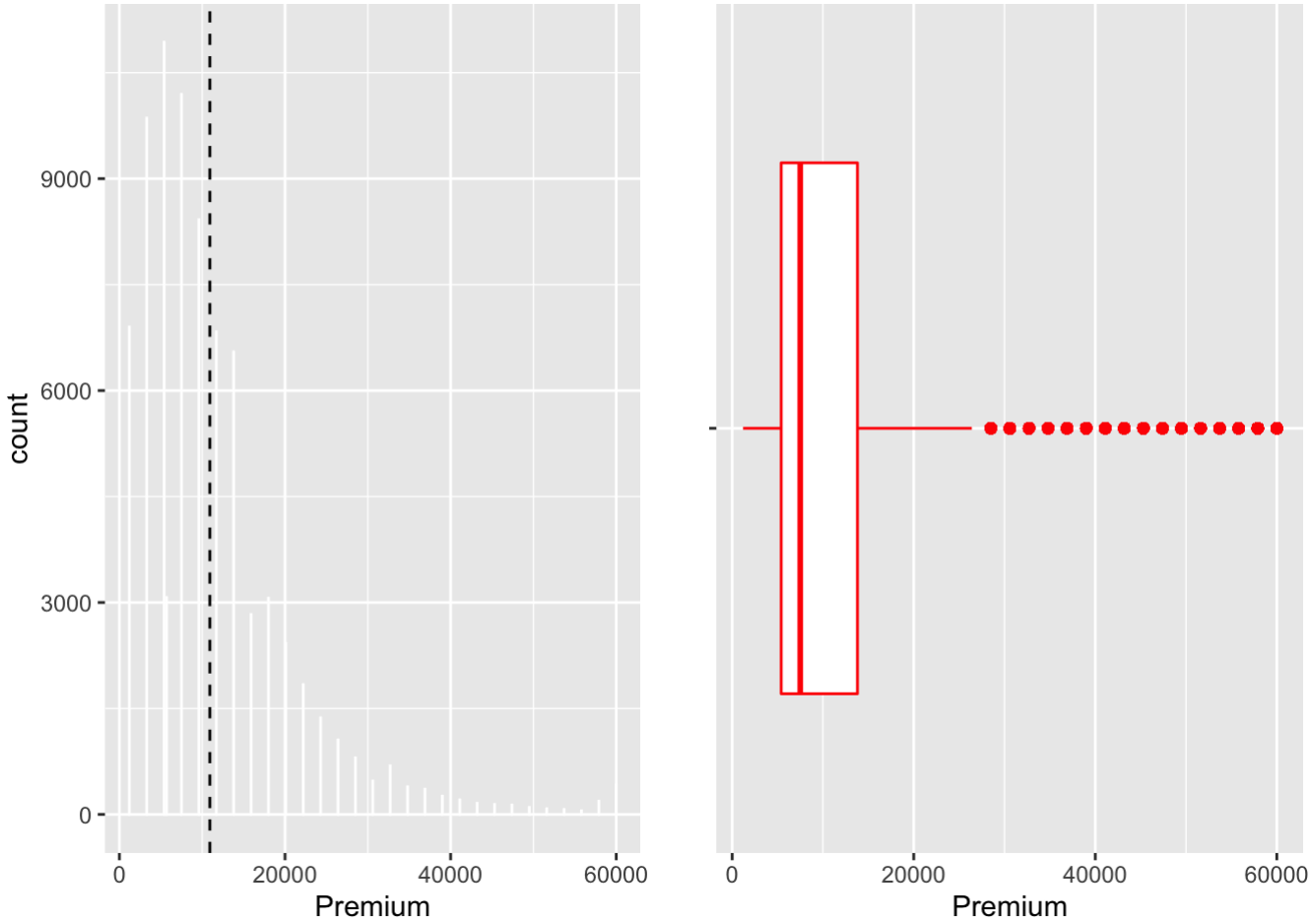
- Risk score which is similar to “credit score” is extremely left skewed (negative).
- The mean and Median are around 99 which denoted there aren’t any outliers influencing the data
- The data is majorly at the score of around 99.
- The left skewed data is skewed between 98 & 92.

9. Observation on Riskscore_bins of customers:



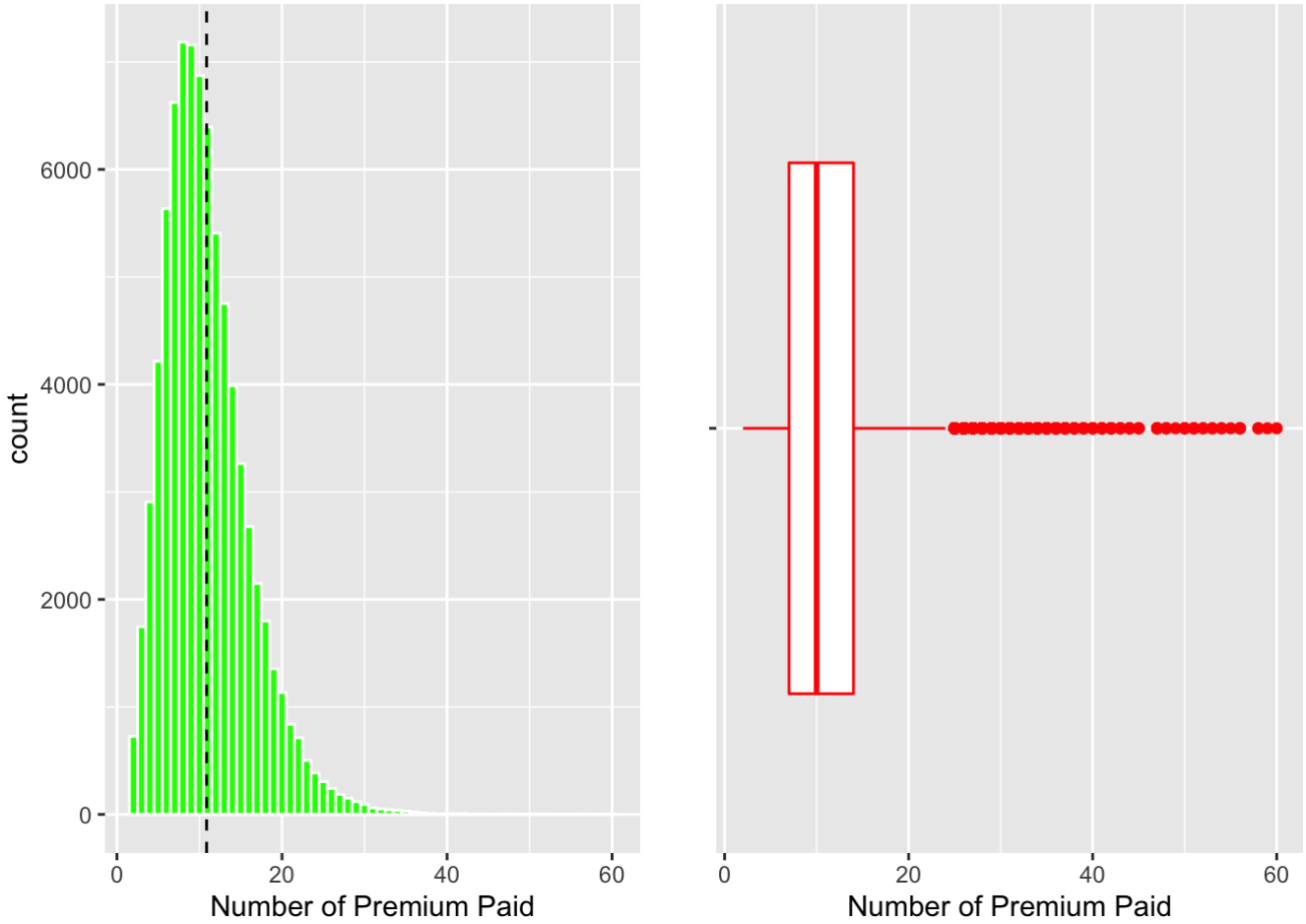
- The The median is bin 9 and mean is around 8.5 meaning majority fall in bin 9 category
- Outliers are to the left with 1st Quartile at 8 bins

10.Observation on Premiums paid by customers:



- Premiums paid by customers is quite positive skewed and spread out with a huge range between the minimum & maximum plus the range between 3rd quartile and 1st quartile.
- The Median is at 7500 but the mean is at 10,925 showing the presence of huge outliers which is skewing the data to the right.
- The range of outliers, ranges from 13800 (3rd Quartile) right upto $60,000 paid as premium.

11. Observation on the number of Premiums paid by the customers

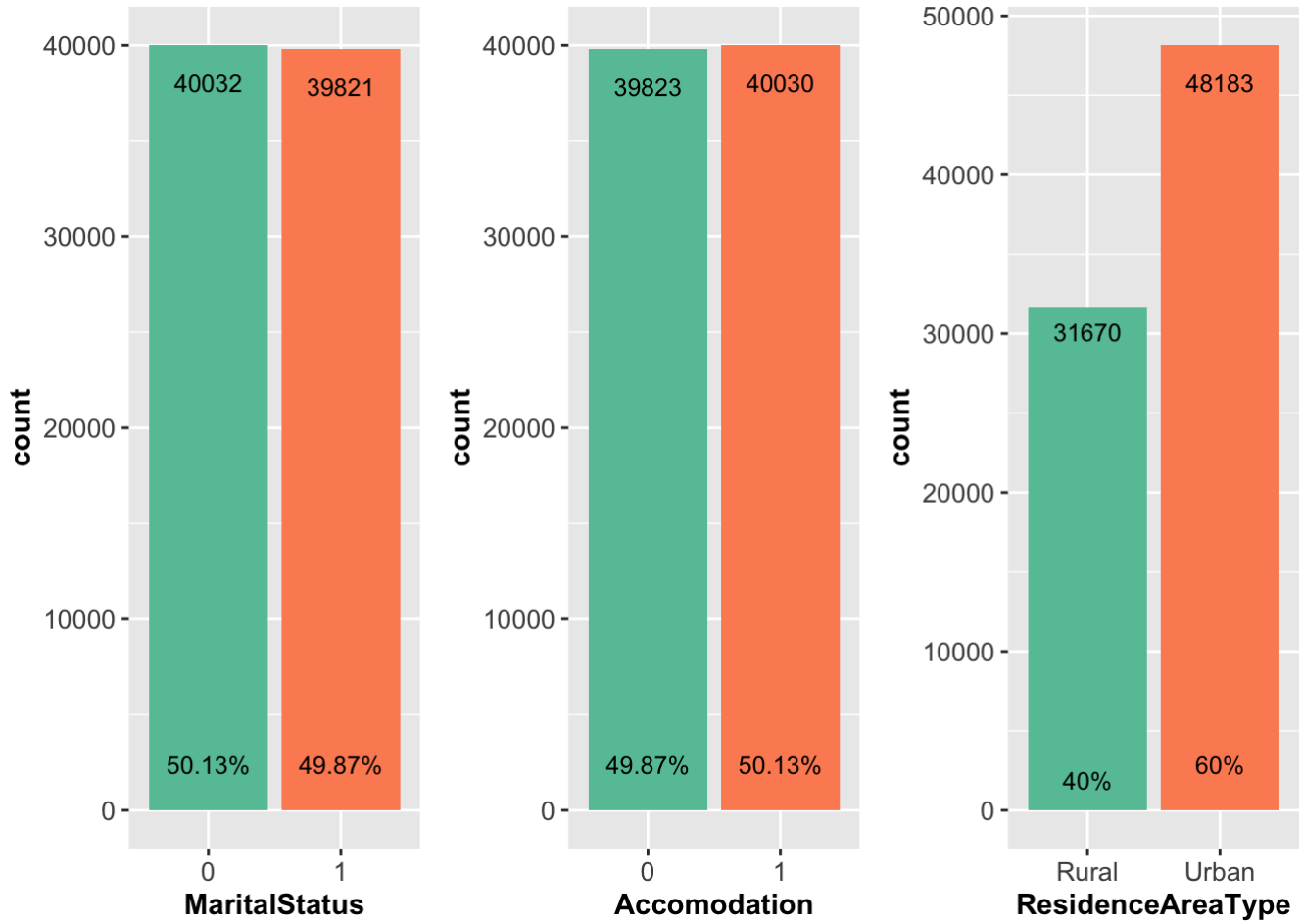


- The number of premiums paid by customers shows a trend similar to the premium paid by the customers.
- The data is positive skewed and spread out between 2 & 60 numbers of premiums paid.
- The mean & median are at 10.86 & 10 resp, showing the outliers aren't majorly influening the data.
- Though there are outliers which are far beyond the 3rd quartile (14) as far as 60.
- The bulk of the nos. of premiums are between 7 & 14.

2.2.c Setting up the aesthetics

2.2.d Plotting the Numerical variables

2.2.c Partitioning the barcharts



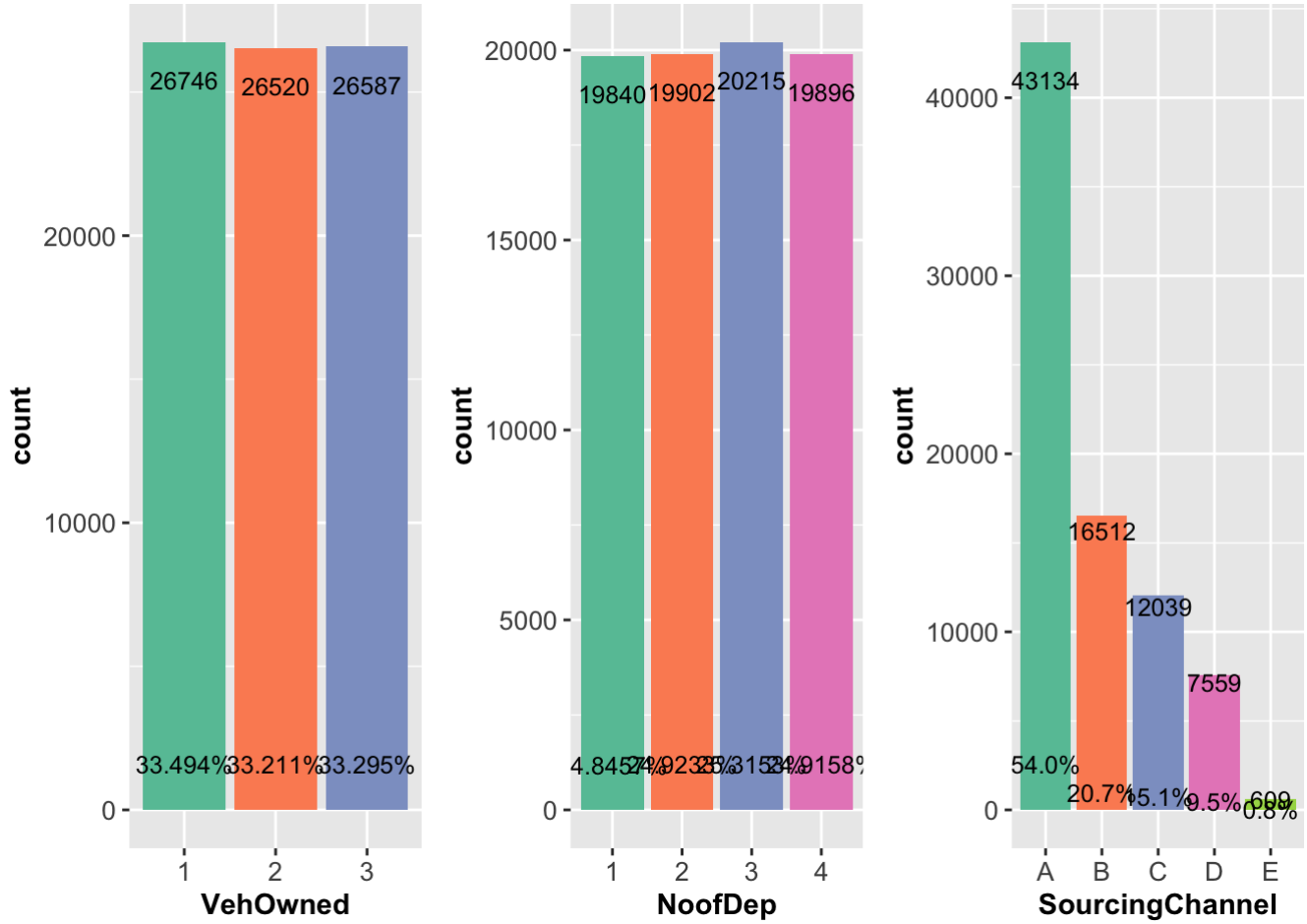
2.2.d Plotting the Numerical variables contd…

Marital Status: Not much difference in the 2 cohorts, with a slight edge for the unmarried.

Accommodation: Again not much difference in the ones owning their houses and renting them. A slight edge to the ones owning their houses.

Residence Area: There we see 60% of the customers coming from the urban area

2.2.e Partitioning the barcharts



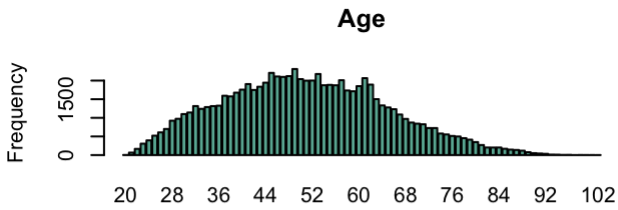
Number of vehicles owned: Again equally distributed at around 33% for 1, 2 & 3 vehicles owned by the customers

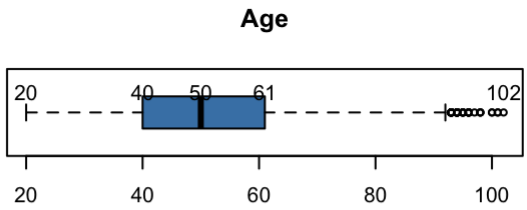
Number of Dependents: The four cohorts i.e. 1, 2, 3 & 4 have similar numbers in the data. All are between 24% & 25% of the share with a slight edge for 3 dependent at 25.31%.

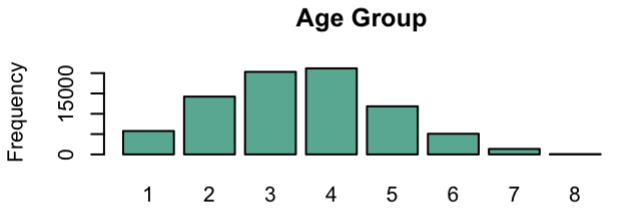
Sourcing Channels: of the five cohorts i.r. A,B,C,D & E the bulk of the customers at 54%vhave been sourced by Channel A. Substantial amount of customers come from Channel B (20.7%), Channel C (15%) & Channel D (9.5%)

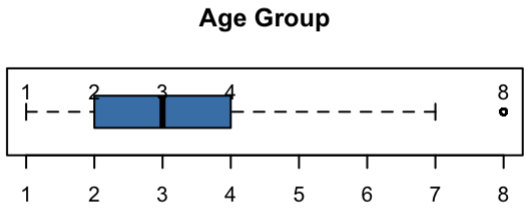
*We can see Sourcing Channels & Residence Area are the only two verticals from where we are able to see a diversion in the customer data.

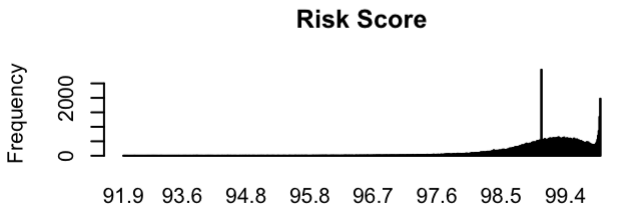
2.2.f Visualize properties of all continuous variables

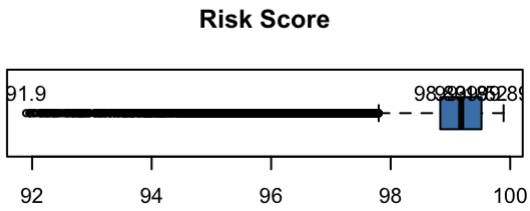


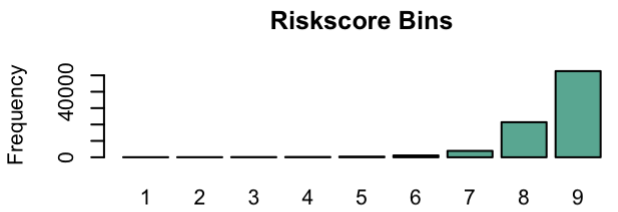


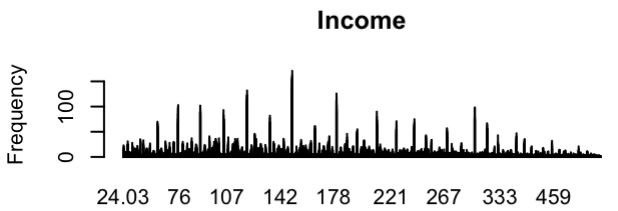


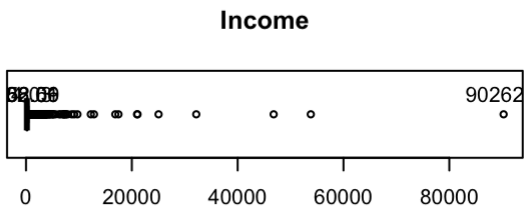












Age shows a almost normal distribution spread widely between 20 & 90, with the bulk between 20 & 90.

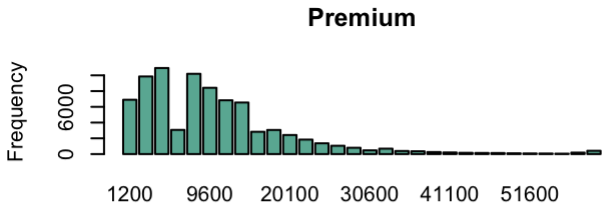
Age Groups follows the Age distribution, with highest concentration in Group 3 followed by Group 4. The median is at Group 4.

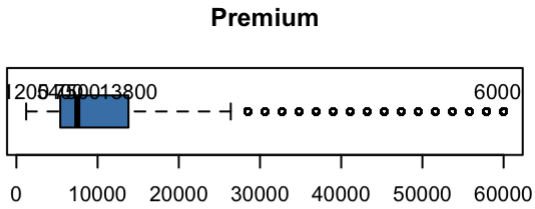
Risk Score sees a left skew with the concentration between 98.83 & 99.52. The tail is between 91.90 and 98.

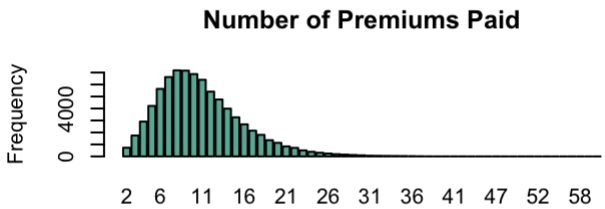
Risk-Score Bins sees a left skew with bin number 9 housing the bulk of the risk score followed by bin numeber 8. There are'nt any significant numbers of risk scores beyond these 2 bins.

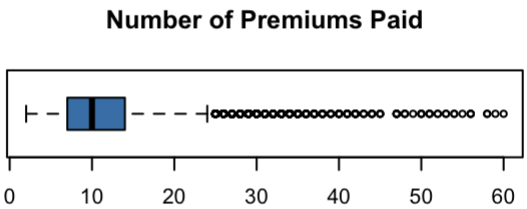
Income levels seems dispersed unevenly in the spread.

2.2.g Visualize properties of all continuous variables contd…





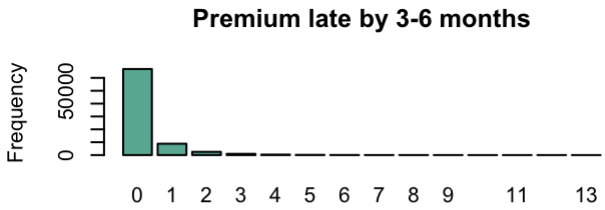


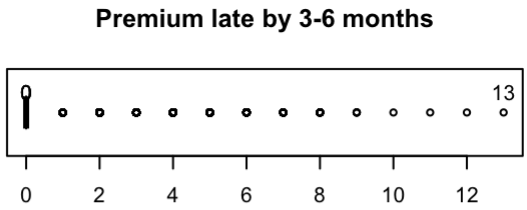


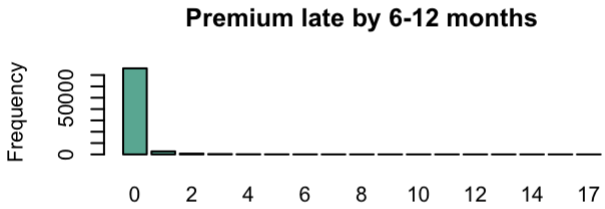
Premiums paid sees a right skew with a sharp dip in between the rise. The concentrations is between 5400 & 13800.

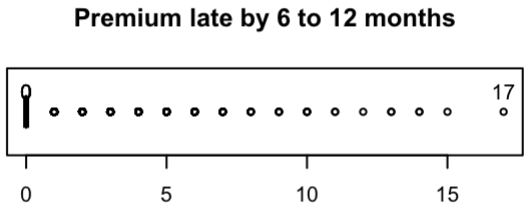
The number of Premiums Paid seems to have a normal distribution with a positive skew. The concentration is between 7 & 14. Many outliers far & wide up to 60.

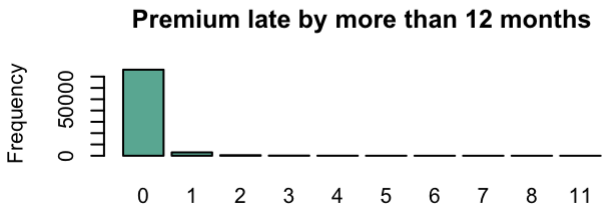
2.2.h Visualize properties of all continuous variables contd…

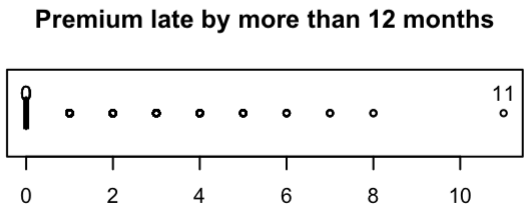












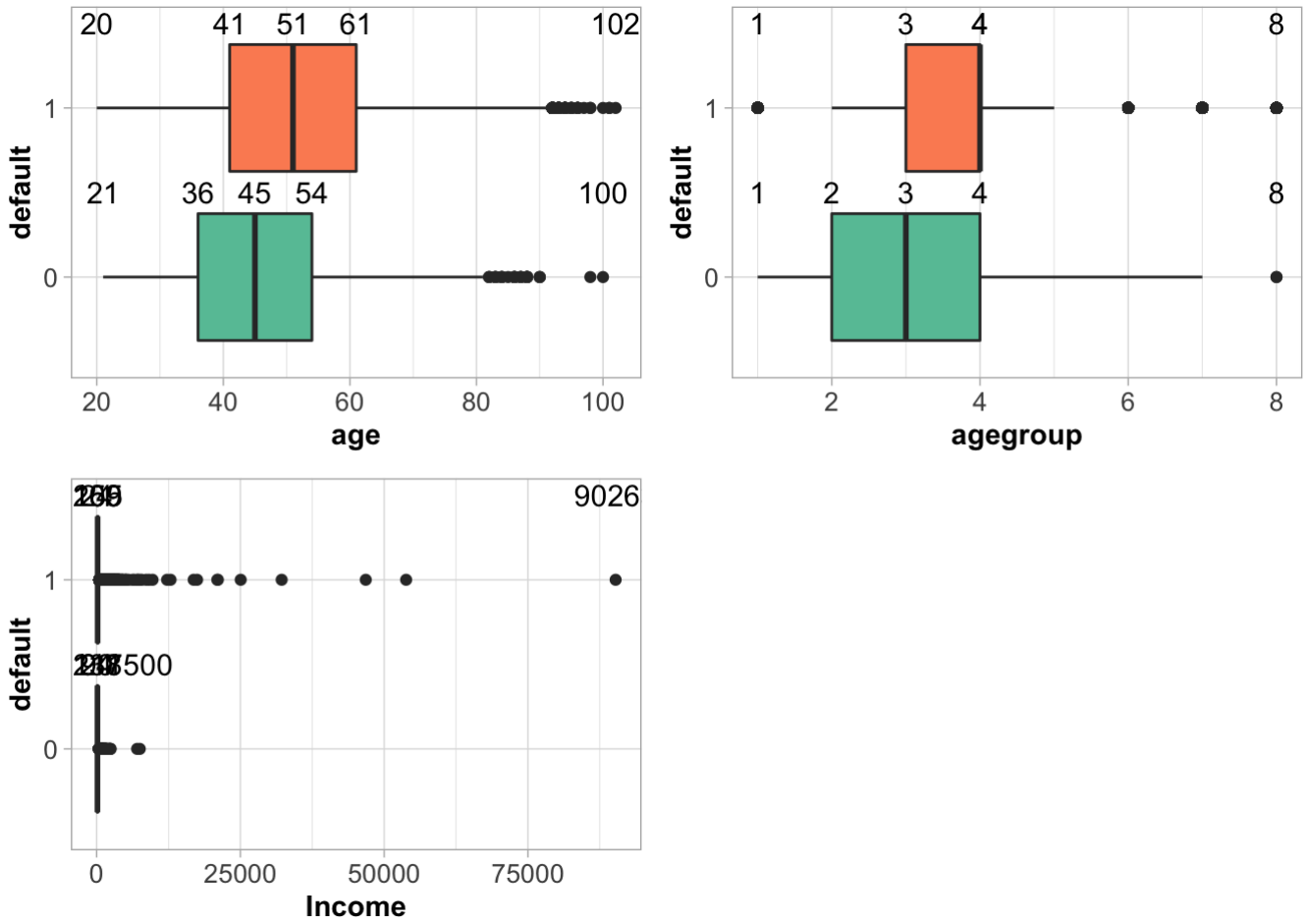
Premium late by 3 to 6 Months, 6 to 12 Months & more than 12 Months:

- Looking at the three cohorts above who have been late in paying their premiums on time, its apparent that maximum numbers in all three verticals are the ones who has not been late in paying their premiums on time.
- There seems comparatively more people who have delayed paying their premiums 1 or 2 time between 3-6 months compared to the ones who delayed their premiums between 6-12 months & beyond 12 months.

2.3 B-Variant Analysis

2.3.a Setting up the aesthetics

2.3.b Default v/s numerical variables

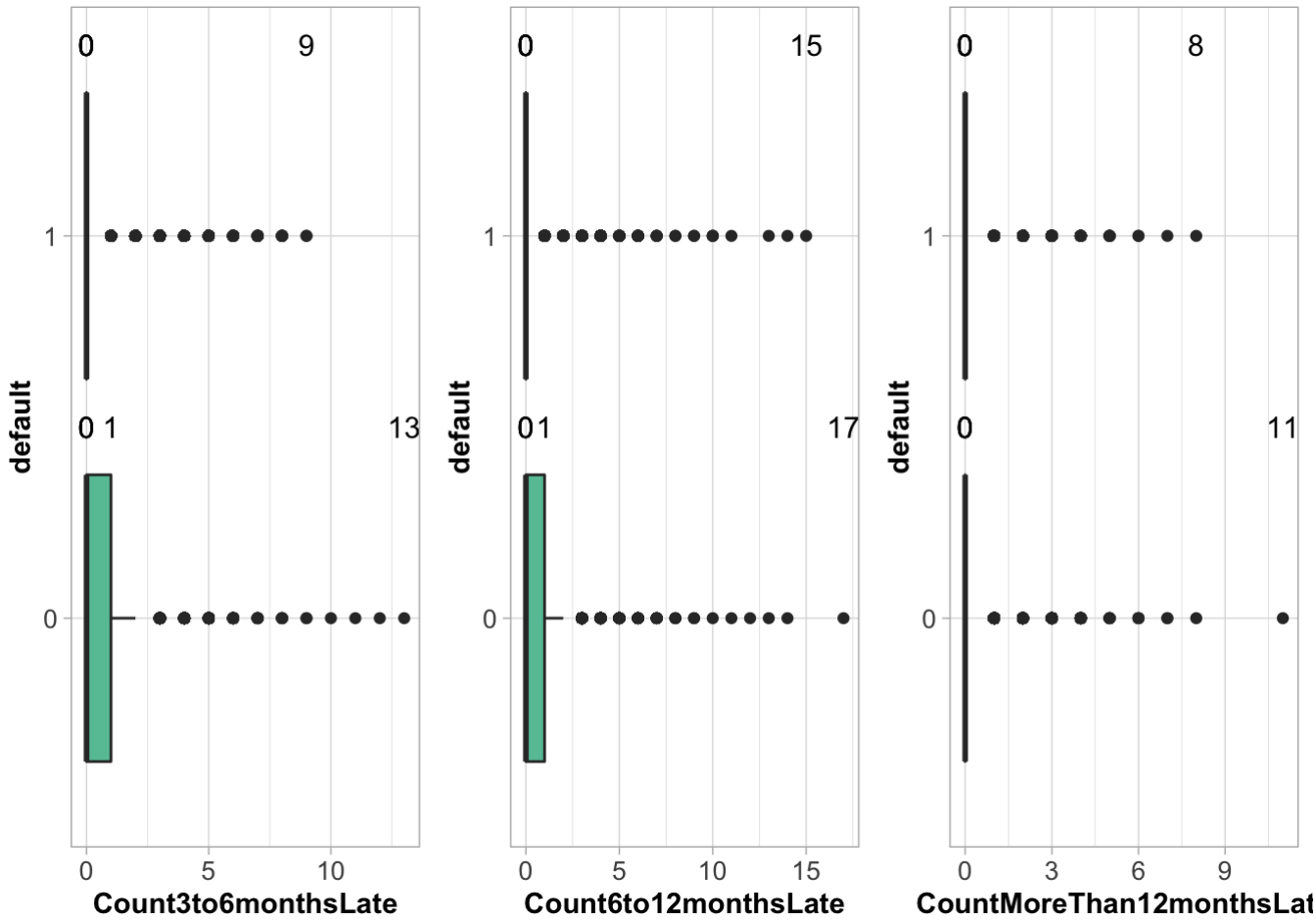


Age: The Defaulters are comparatively a younger cohort with the median at 46. The concentration range being between 37 & 54. The non-defaulters are at a median of 51 with a range between 41 & 62.

Age Group: The defaulters are falling in the concentration range between Group 2 & 4 with a median at 3. The non defaulters are ina a range of Group 3 & 5 with median of 4 which reflects that the concentration of defaulter comparatively fall iin a less age bracket

Income: The income levels of the defaulters show that they come from a low income category compared to the non-defaulters.

2.3.c Default v/s numerical variables contd…

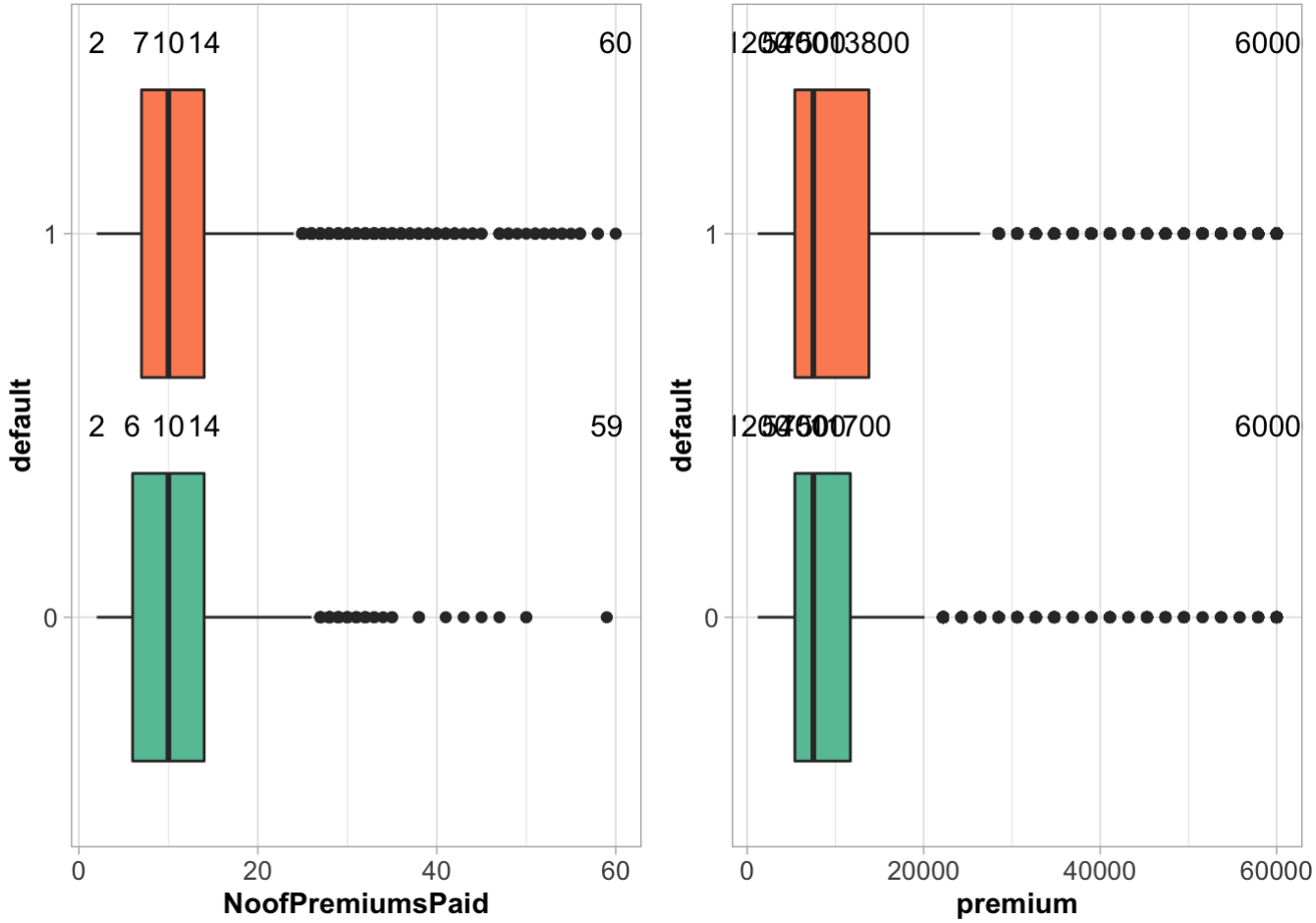


Delayed Premium of 3 to 6 Months: + More than 85% of the total non defaulters have never defaulted premium by 3 to 6 months. 10% defaulted once & 2.5% defaulted twice. + More than 53% of the total defaulters have never defaulted premium by 3 to 6 months. 23% defaulted once & more than 11% defaulted twice

Delayed Premium of 6 to 12 Months: + Almost 97% of the total non defaulters have never defaulted premium by 6 to 12 months. 2.5% defaulted once. + 70% of the total defaulters have never defaulted premium by 6 to 12 months. 16.5% defaulted once.

Delayed Premium of more than 12 Months: + 96.5% of the total non defaulters have never defaulted premium by more than 12 months, Approx 3% defaulted once. + More than 76% of the total defaulters have never defaulted premium by more than 12 months. 16.7% defaulted once.

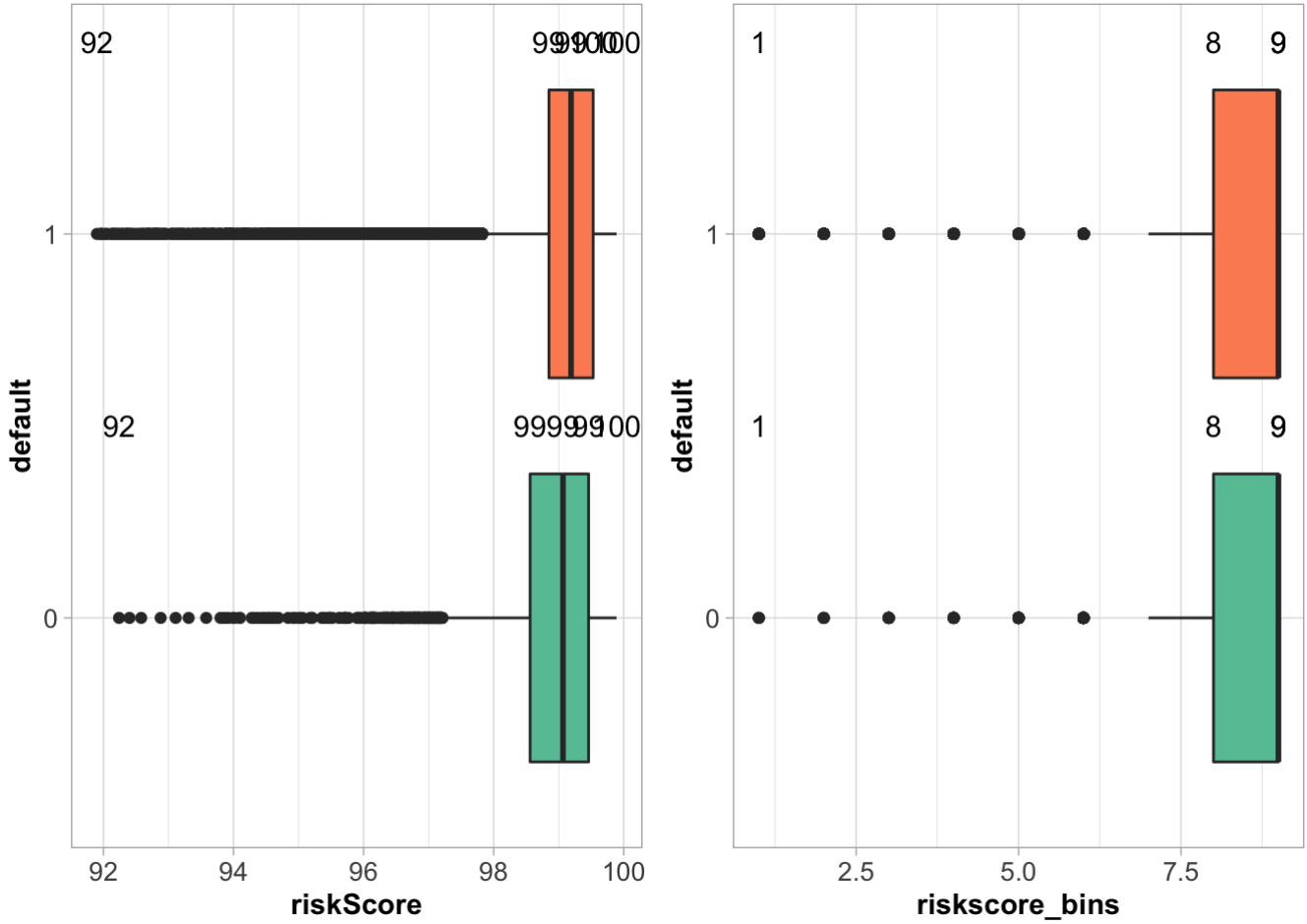
2.3.d Default v/s numerical variables contd…



Nos. of Premiums Paid: Both the cohorts i.e. of defaulters and non- defaulters show a similar range in the number of premiums paid. Both cohorts have a median of 10 and almost similar range.

Premium Paid: Both cohorts has the same median of 7500. The range for defaulters is comparatively smaller.

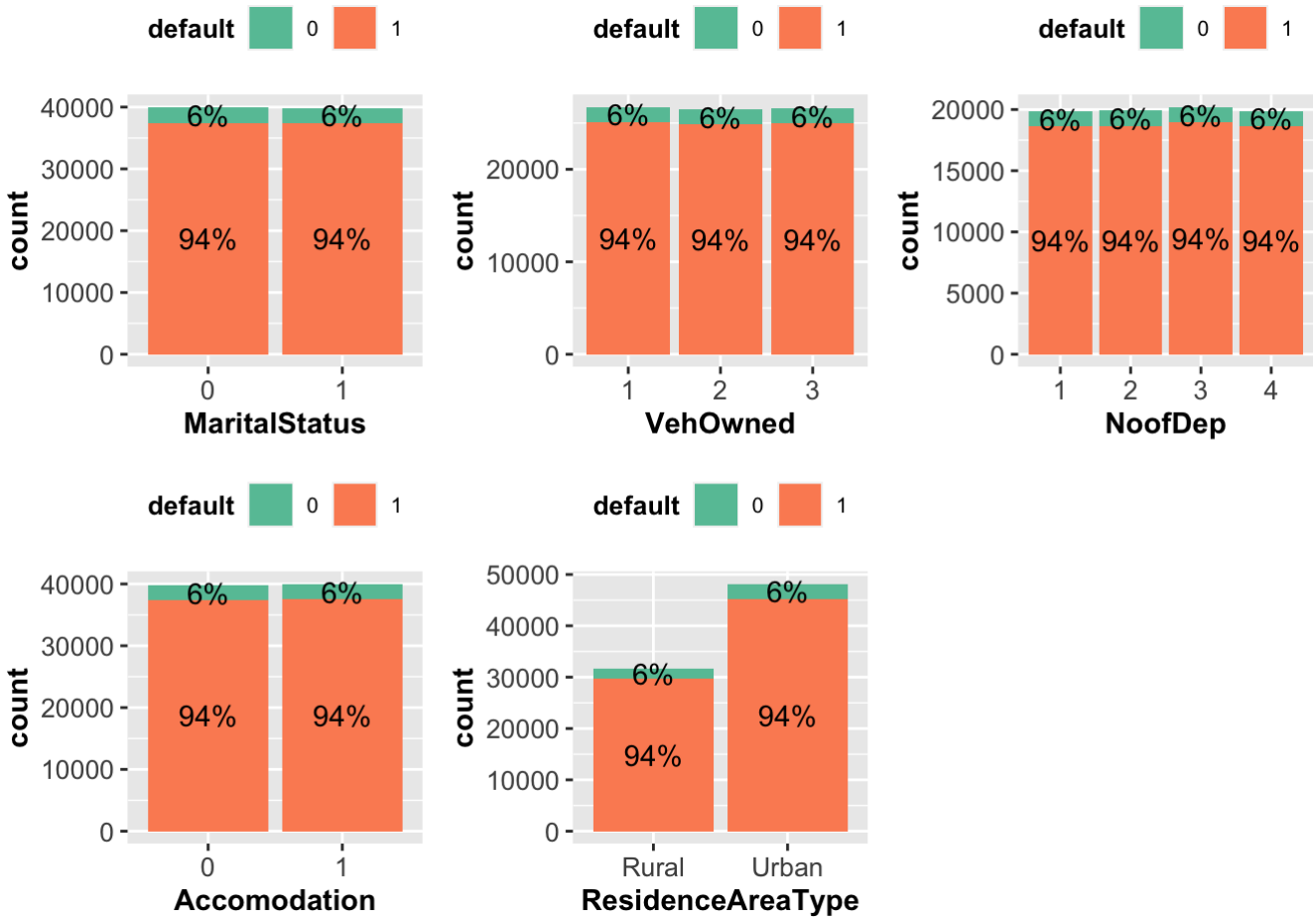
2.3.e Default v/s numerical variables contd…



Risk Score The defaulters & non-defaulters fall in the same range between 99 & 100.

2.3.f Setting up the aesthetics

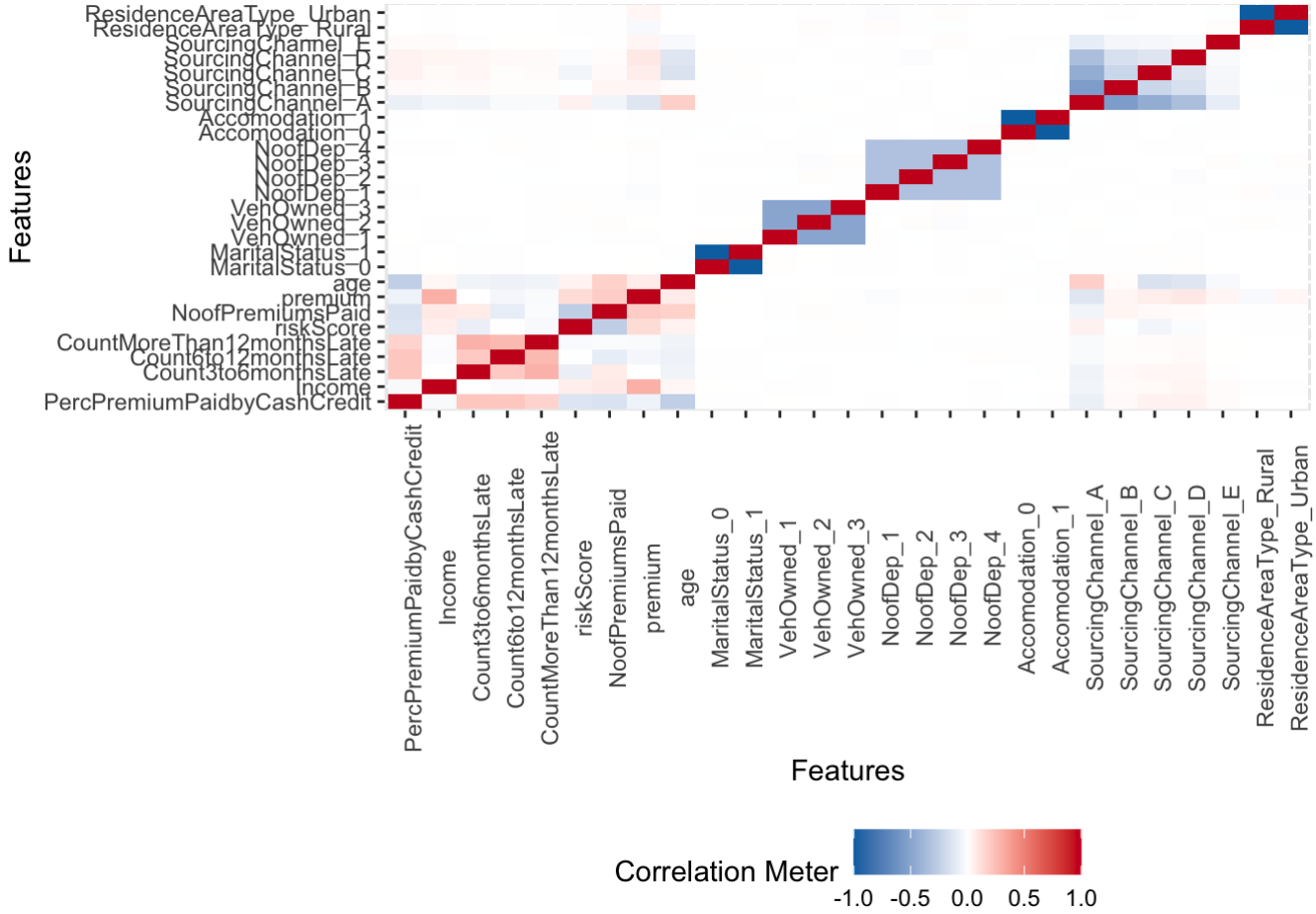
2.3.g “Default” vs categorical variables



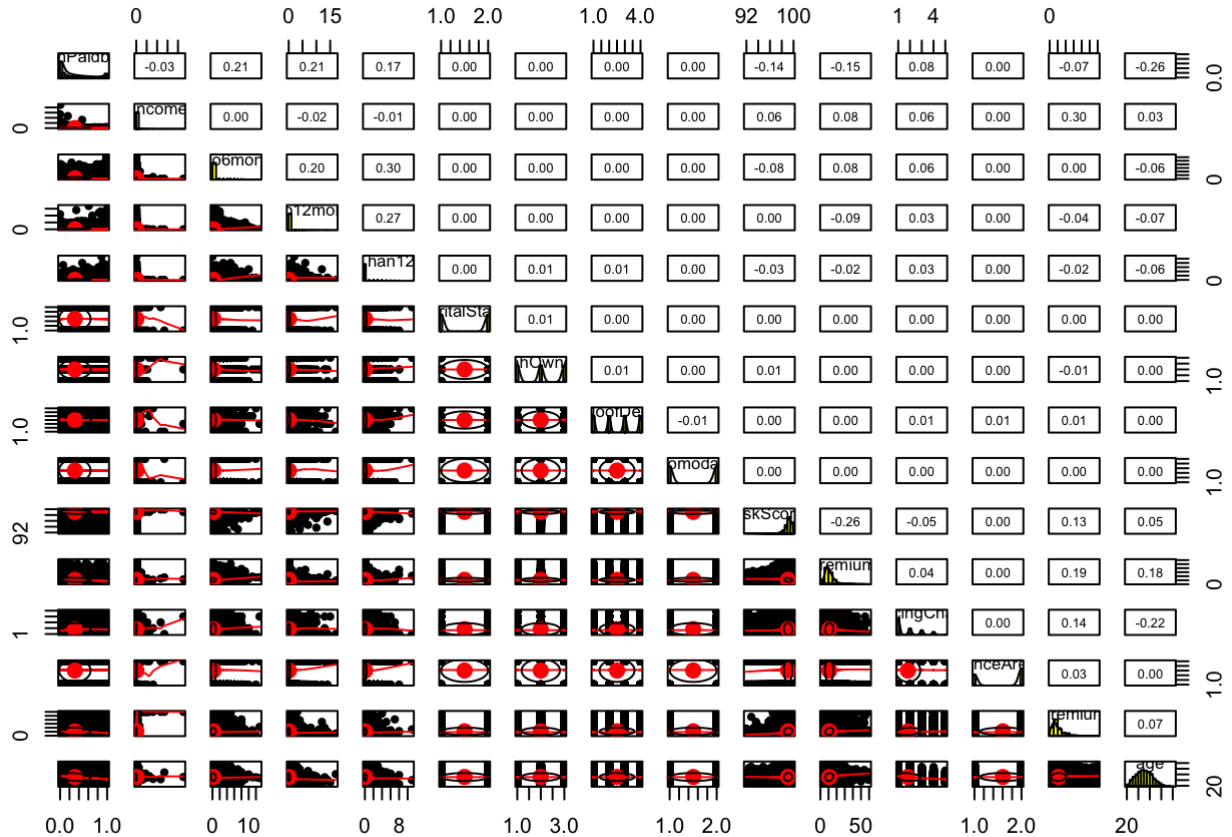
Observation:

*Same kind of data distribution witnessed between defaulters and non-defaulters in the “Marital Status”, “Number of Vehicles owned”, “Number of Dependents” & “Accommodations” & “Residence Area”

2.4.a Correlation Plot



2.4.b.Density Plot



Insightful visualizations

- Correlation - We can see a relative correlation between:
- Income & Premium = **0.30**
- Late Payment of Premium by 3 to 6 months & Late Payment of Premium by more than 12 months = **0.30**
- Late Payment of Premium by 6 to 12 months & Late Payment of Premium by more than 12 months = **0.27**
- Late Payment of Premium by 3 to 6 months & Late Payment of Premium by 6 to 12 months = **0.20**
- Premium paid in cash & Late Payment of Premium by 3 to 6 months = **0.21**
- Premium paid in cash & Late Payment of Premium by 6 to 12 months = **0.21**
- Premium paid in cash & Late Payment of Premium by more than 12 months = **0.17**
- Number of Premium Paid & Premium = **0.19**
- Number of Premium Paid & Age = **0.18**
- Premium & Sourcing Channel = **0.14**
- Premium & Risk Score = **0.13**

There will be further probe done with regards to the correlation and importance of the correlation of these variables to investigate the cohorts likely to default the insurance premium.

Analysing the Categorical variable for corelation

Perform a **Chi-Square test** which is a statistical method to determine if two categorical variables have a significant correlation between them.

—The **hypothesis testing** will essentially be: + Null Hypothesis - There is no correlation between the two variables
+ Alternate Hypothesis - Variable A is correlated with variable B with a set p-values, we will determine the statistical significance of the variables. p-values are << 0.05

2.4.c.Chi-Square Test

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  PremiumData$MaritalStatus and PremiumData$Accomodation
## X-squared = 0.61971, df = 1, p-value = 0.4312
```

```
##
##  Pearson's Chi-squared test
##
## data:  PremiumData$ResidenceAreaType and PremiumData$VehOwned
## X-squared = 4.7233, df = 2, p-value = 0.09426
```

```
##
##  Pearson's Chi-squared test
##
## data:  PremiumData$VehOwned and PremiumData$Accomodation
## X-squared = 0.97604, df = 2, p-value = 0.6138
```

```
##
##  Pearson's Chi-squared test
##
## data:  PremiumData$NoofDep and PremiumData$VehOwned
## X-squared = 14.83, df = 6, p-value = 0.02162
```

```
##
##  Pearson's Chi-squared test
##
## data:  PremiumData$SourcingChannel and PremiumData$ResidenceAreaType
## X-squared = 6.0392, df = 4, p-value = 0.1962
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  PremiumData$MaritalStatus and PremiumData$ResidenceAreaType
## X-squared = 0.0014792, df = 1, p-value = 0.9693
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  PremiumData$Accomodation and PremiumData$ResidenceAreaType
## X-squared = 0.043716, df = 1, p-value = 0.8344
```

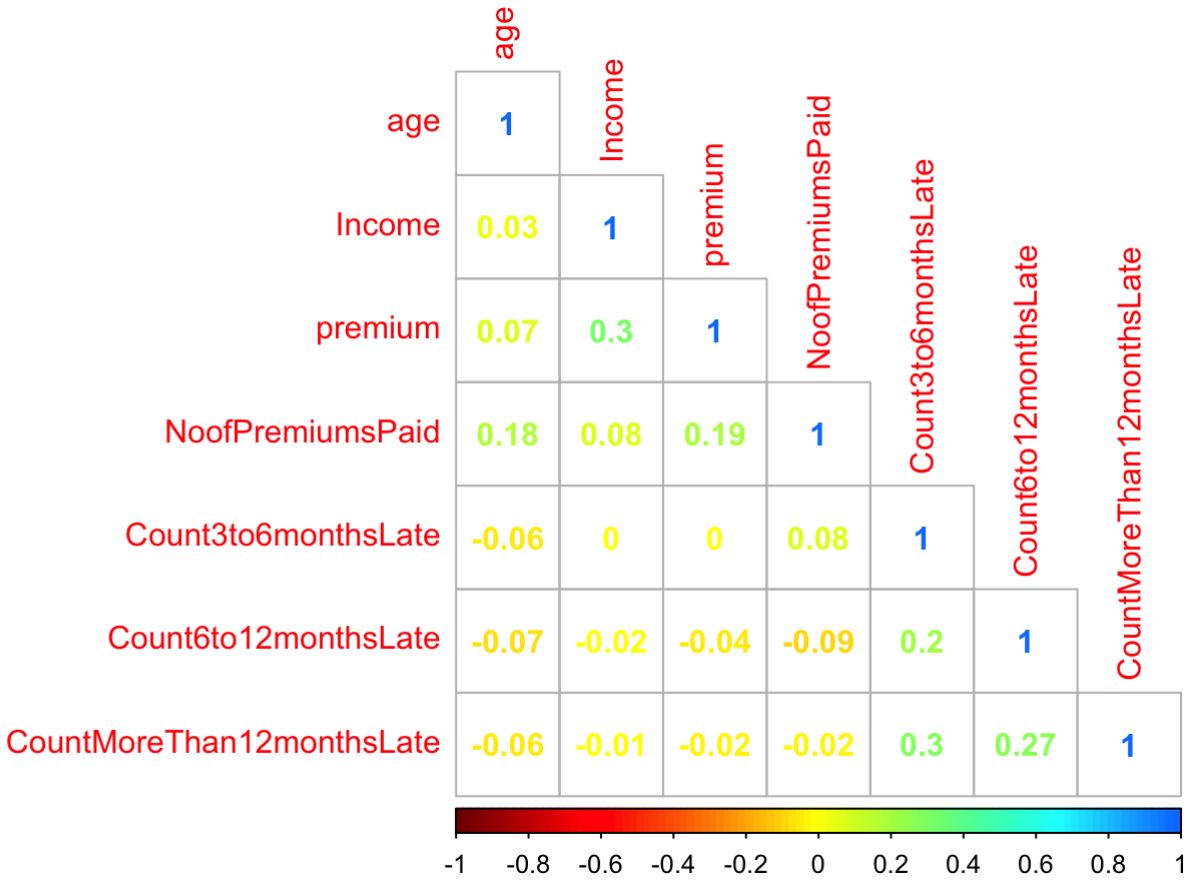
```
##
##  Pearson's Chi-squared test
##
## data:  PremiumData$ResidenceAreaType and PremiumData$VehOwned
## X-squared = 4.7233, df = 2, p-value = 0.09426
```

- The relationship between All the categorical variable are significant as p-value is more than significance level of 0.05. hence we reject the null hypothesis.

Analysing the numerical/continous variables

Create a subset of the numerical variables

Correlation plot between numerical variavles



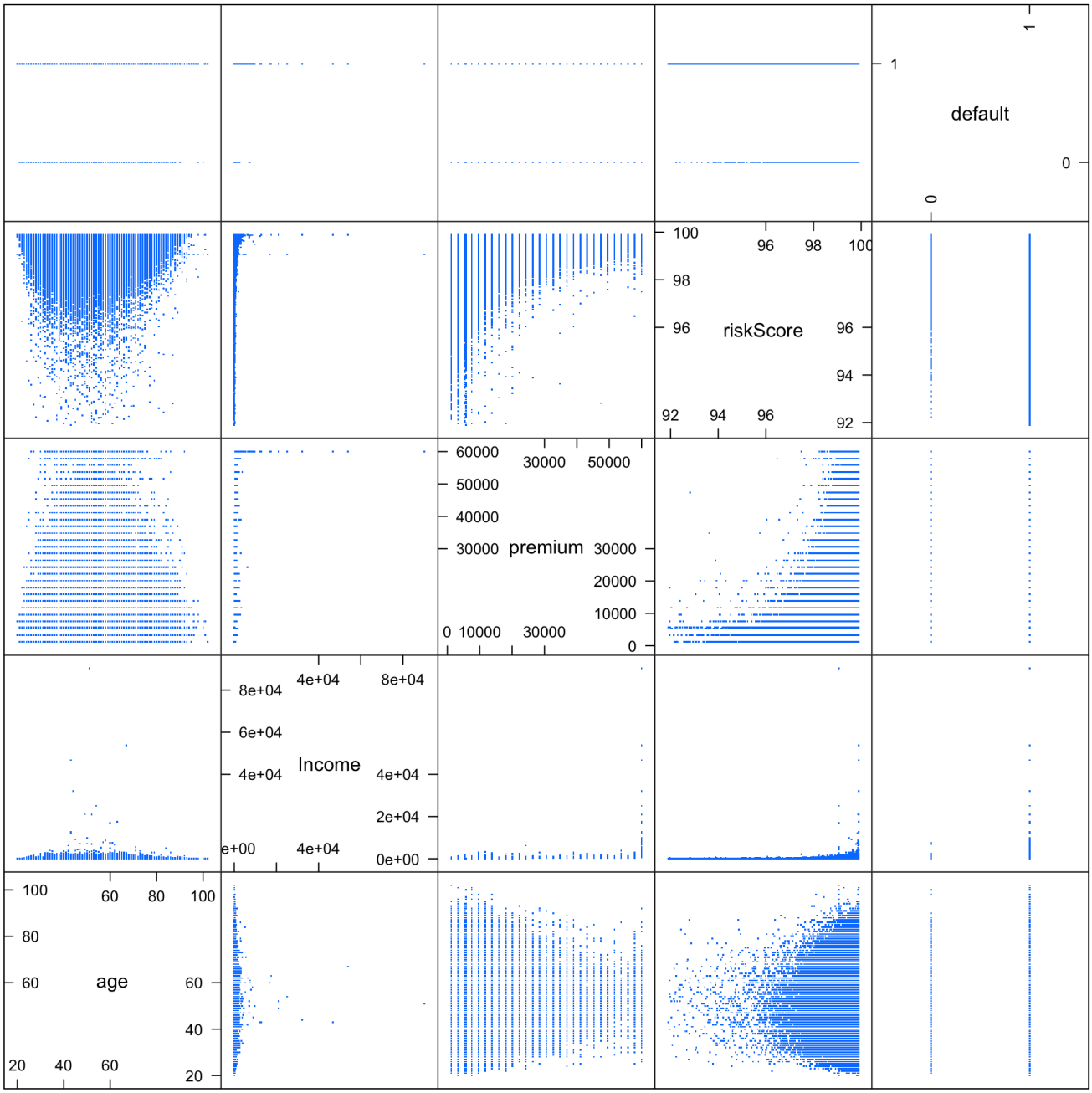
Observation There seems to be no high correlations between any of the numerical variables.

Adding the dependent variable back to filtered data

```
## [1] 79853    10
```

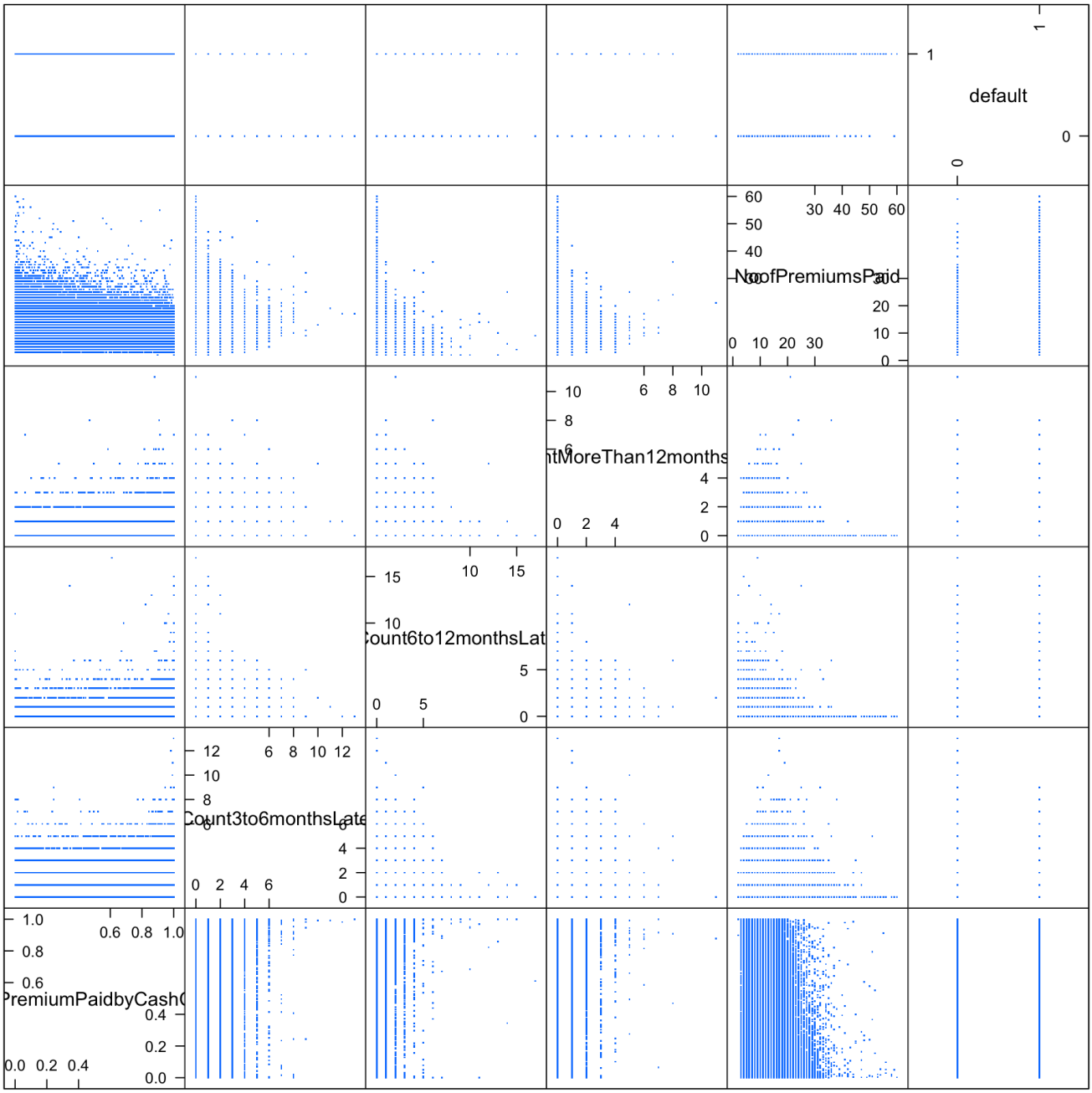
```
##  [1] "age"                          "Income"
##  [3] "riskScore"                    "premium"
##  [5] "PercPremiumPaidbyCashCredit" "Count3to6monthsLate"
##  [7] "Count6to12monthsLate"         "CountMoreThan12monthsLate"
##  [9] "NoofPremiumsPaid"             "default"
```

Plotting the different variables of filtered data



Scatter Plot Matrix

Observations * There isn't any linear correlation seen in with any of the variables. * Premiums comparatively seem to have more relation with the younger age segments * Income though dispersed is more concentrated around 40s and 70s. * Higher the income, the risk score keeps increasing



Scatter Plot Matrix

Observations

- Similar pattern observed between “Count_3-6_months_late”, “Count_6-12_months_late” & “Count_more_than_12_months_late”
- Similar behavior with regards to Number of Premiums paid with all the 3 late payment cohorts, namely “Count_3-6_months_late”, “Count_6-12_months_late” & “Count_more_than_12_months_late”
- Number of Premiums paid & Premium paid in cash seem more in “Count_3-6_months_late” cohort.
- Premium paid in cash is higher between 0-30 number of premiums paid.

1.8 Ensure that the target variable is a factor & Rename the levels & Relevel

```
## [1] "Defaulters"     "Non_Defaulters"
```

1.9 Retain “age” and remove the variable “Age Group” after EDA

3. Modelling: Create Multiple Models

3.1 Split the Data into Train & Test (80-20 split)

```
##
##     Defaulters Non_Defaulters
##       6.259881      93.740119
```

```
##
##     Defaulters Non_Defaulters
##       6.255479      93.744521
```

```
##
##     Defaulters Non_Defaulters
##       6.259001      93.740999
```

- The newly form Train & Test data have similar distribution of the dependent variable.

3. Model Processing

3.1 SMOTE - Smotering the Train Data Set before we crate models for analysis

```
##
##     Defaulters Non_Defaulters
##           3999          59884
```

```
##
##     Defaulters Non_Defaulters
##     0.06259881     0.93740119
```

```
##
##     Defaulters Non_Defaulters
##       25.50336       74.49664
```

```
##
##     Defaulters Non_Defaulters
##         151962         443889
```

- We SMOTE the trained dataset to handle the class unbalanced classification in the data set.

Define the training control

3.2 Model 1 - CART Model

3.2.a. Cart Model 1

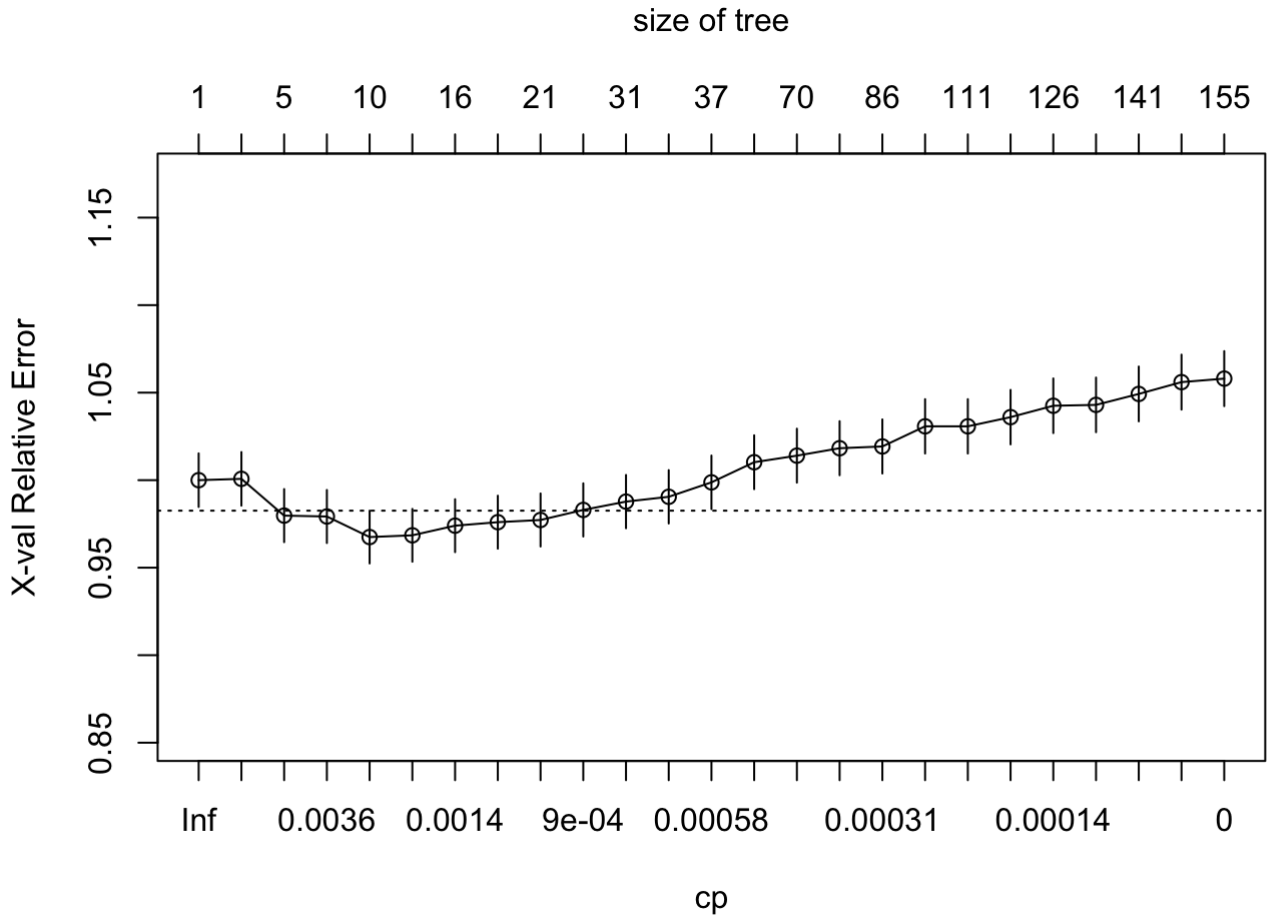
Built Cart Model on train data set,use the “rpart” and the “rattle” libraries to build decision trees.

- We will need to prune this decision tree

Model Tuning

The cost complexity table can be obtained using the printcp or plotcp functions

```
##
## Classification tree:
## rpart(formula = default ~ ., data = PremiumData_Train, method = "class",
##     control = r.ctrl)
##
## Variables actually used in tree construction:
##  [1] age                          Count3to6monthsLate
##  [3] Count6to12monthsLate         CountMoreThan12monthsLate
##  [5] Income                       MaritalStatus
##  [7] NoofDep                      NoofPremiumsPaid
##  [9] PercPremiumPaidbyCashCredit premium
## [11] ResidenceAreaType            riskScore
## [13] SourcingChannel              VehOwned
##
## Root node error: 3999/63883 = 0.062599
##
## n= 63883
##
##            CP nsplit rel error  xerror     xstd
## 1  9.0023e-03      0   1.00000 1.00000 0.015310
## 2  5.3763e-03      2   0.98200 1.00075 0.015316
## 3  4.3761e-03      4   0.97124 0.97974 0.015165
## 4  3.0008e-03      6   0.96249 0.97924 0.015161
## 5  1.7504e-03      9   0.95174 0.96749 0.015076
## 6  1.5004e-03     11   0.94824 0.96849 0.015083
## 7  1.2503e-03     15   0.94224 0.97399 0.015123
## 8  1.1253e-03     18   0.93848 0.97599 0.015138
## 9  1.0003e-03     20   0.93623 0.97724 0.015147
## 10 8.1270e-04     26   0.93023 0.98300 0.015188
## 11 7.5019e-04     30   0.92698 0.98775 0.015223
## 12 6.6683e-04     32   0.92548 0.99050 0.015242
## 13 5.0013e-04     36   0.92173 0.99875 0.015301
## 14 4.5011e-04     47   0.91573 1.01025 0.015383
## 15 3.3342e-04     69   0.90248 1.01400 0.015410
## 16 3.1258e-04     72   0.90148 1.01825 0.015440
## 17 3.0008e-04     85   0.89697 1.01925 0.015447
## 18 2.5006e-04     90   0.89547 1.03076 0.015528
## 19 2.0839e-04    110   0.88997 1.03076 0.015528
## 20 1.6671e-04    122   0.88747 1.03601 0.015565
## 21 1.2503e-04    125   0.88697 1.04251 0.015610
## 22 1.0717e-04    133   0.88597 1.04301 0.015614
## 23 7.1446e-05    140   0.88522 1.04926 0.015657
## 24 3.5723e-05    147   0.88472 1.05601 0.015704
## 25 0.0000e+00    154   0.88447 1.05801 0.015718
```



The unncessarily complex tree above can be pruned using a complexity threshold. Using a complexity threshold of 0.062 gives us a relatively simpler tree. Variables actually used in the tree construction:

- Age
- Income
- perc_premium_paid_by_cash_credit
- Count_6to12_months_late
- Risk_Score

Cart Model 2

```
##
## Classification tree:
## rpart(formula = default ~ ., data = PremiumData_Train, method = "class",
##     control = r.ctrl)
##
## Variables actually used in tree construction:
## character(0)
##
## Root node error: 3999/63883 = 0.062599
##
## n= 63883
##
##      CP nsplit rel error xerror    xstd
## 1 0.062      0         1      1 0.01531
```

```
## n= 63883
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 63883 3999 Non_Defaulters (0.06259881 0.93740119) *
```

Variables actually used in tree construction:

- Income
- perc_premium_paid_by_cash_credit

Let us check the variable importance

```
##        Count6to12monthsLate PercPremiumPaidbyCashCredit
##                  737.8686722                 263.5382731
##         Count3to6monthsLate   CountMoreThan12monthsLate
##                  177.6947893                 135.0233069
##                    riskScore                      Income
##                  102.6920693                 101.5507095
##            NoofPremiumsPaid                         age
##                   87.9651014                  63.3086932
##              riskscore_bins                     premium
##                   51.6845038                  50.7344685
##             SourcingChannel                     NoofDep
##                   24.4332061                   7.3832253
##                     VehOwned               MaritalStatus
##                    6.7144927                   4.8097249
##           ResidenceAreaType                Accomodation
##                    3.8078456                   0.2595493
```

- Top Important Variable according to the CART model are:

1. Count 6to 12 months Late
2. Perc Premium Paid by Cash Credit
3. Count 3 to 6 months Late
4. Count More Than 12 months Late
5. risk Score
6. Income

Model Validation

```
##                 train_predict.class_CART
##                  Defaulters Non_Defaulters
##   Defaulters              0           3999
##   Non_Defaulters          0          59884
```

```
## [1] 0.9374012
```

The training model had a 93.74% accuracy.

Model Evaluation

```
##                 test_predict.class_CART
##                  Defaulters Non_Defaulters
##   Defaulters              0            999
##   Non_Defaulters          0          14971
```

```
## [1] 0.9374452
```

- The test model had a 93.74% accuracy.
- The CART model does not overfit the data

Both the Train & Test Models give an accuracy of 93%.

Predict Default class and probability for Cart Model

Confusion Matrix

```
##                 cart_model1_predict_class
##                  Defaulters Non_Defaulters
##   Defaulters            173            826
##   Non_Defaulters        218          14753
```

```
##                 cart_model2_predict_class
##                  Defaulters Non_Defaulters
##   Defaulters              0            999
##   Non_Defaulters          0          14971
```

The Cart Model 1 identifies True Positive and False Negatives better than cart Model 2 which is not able to identifiers Defaulters.

Accuracy

```
## [1] 0.9346274
```

```
## [1] 0.9374452
```

Sensitivity of Model on Test Data

```
## [1] 0.9237946
```

```
## [1] 0.9374452
```

Specificity of models on Test Data

```
## [1] 0.01083281
```

```
## [1] 0
```

Precision

```
## [1] 0.9237946
```

```
## [1] 0.9374452
```

KS

AUC

Gini

Concordance - Discordance

METRICS - Cart Model 1 & 2

```
## [1] 0.93462743 0.92379461 0.01083281 0.92379461 0.52002928 0.80719234 0.03904797
## [8] 0.17065151
```

```
## [1]  9.374452e-01  9.374452e-01  0.000000e+00  9.374452e-01  0.000000e+00
## [6]  5.000000e-01 -2.278008e-16  0.000000e+00
```

Cart Model 1 is a better option among all the coefficients.

Model 2 : Logistic Regression Model

```
## Confusion Matrix and Statistics
##
##
## lrpred     1     2
##      1   122    96
##      2   877 14875
##
##                Accuracy : 0.9391
##                  95% CI : (0.9353, 0.9427)
##     No Information Rate : 0.9374
##     P-Value [Acc > NIR] : 0.2028
##
##                   Kappa : 0.1822
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.122122
##             Specificity : 0.993588
##          Pos Pred Value : 0.559633
##          Neg Pred Value : 0.944325
##              Prevalence : 0.062555
##          Detection Rate : 0.007639
##    Detection Prevalence : 0.013651
##       Balanced Accuracy : 0.557855
##
##        'Positive' Class : 1
##
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 22)
##
##                              Overall
## PercPremiumPaidbyCashCredit 100.0000
## Count3to6monthsLate          73.9971
## Count6to12monthsLate         73.9658
## CountMoreThan12monthsLate    48.1635
## age                          31.3554
## NoofPremiumsPaid             15.5037
## SourcingChannelD              7.6219
## riskscore_bins                7.5710
## NoofDep4                      7.2236
## NoofDep2                      6.5433
## NoofDep3                      6.0244
## SourcingChannelC              5.9771
## premium                       3.9595
## Income                        3.5620
## SourcingChannelE              2.3987
## Accomodation1                 2.1313
## MaritalStatus1                1.0409
## VehOwned3                     0.9924
## SourcingChannelB              0.9381
## ResidenceAreaTypeUrban        0.8338
```

Predict Default class and probability for Logistic Regression Model

Creating Confusion Matrix

```
##    lrmod_predict_class
##     Defaulters Non_Defaulters
##   1        122            877
##   2         96          14875
```

Accuracy

```
## [1] 0.9390733
```

Sensitivity of Model on Test Data

```
## [1] 0.9314339
```

Specificity of models on Test Data

```
## [1] 0.007639324
```

Precision

```
## [1] 0.9314339
```

KS

AUC

Gini

Concordance - Discordance

METRICS - Logistic Regression models

Model 3 Naive Bayes Model

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = x, y = y, laplace = param$laplace, usekernel = TRU
E,      adjust = param$adjust)
## - Laplace: 0
## - Classes: 2
## - Samples: 63883
## - Features: 22
## - Conditional distributions:
##     - KDE: 22
## - Prior probabilities:
##     - Defaulters: 0.0626
##     - Non_Defaulters: 0.9374
##
## ---------------------------------------------------------------------------------
```

Confusion matrix

```
## Confusion Matrix and Statistics
##
##
## nb_predictions_test     1     2
##                   1     0     3
##                   2   999 14968
##
##                Accuracy : 0.9373
##                  95% CI : (0.9334, 0.941)
##     No Information Rate : 0.9374
##     P-Value [Acc > NIR] : 0.5474
##
##                   Kappa : -4e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9997996
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.9374335
##              Prevalence : 0.0625548
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0001879
##       Balanced Accuracy : 0.4998998
##
##        'Positive' Class : 1
##
```

Predict Default class and probability for Naive Bayes Model

Creating Confusion Matrix

```
##    model_nb_predict_class
##     Defaulters Non_Defaulters
##   1          0            999
##   2          3          14968
```

Accuracy

```
## [1] 0.9372574
```

Sensitivity of Model on Test Data

```
## [1] 0.9372574
```

Specificity of models on Test Data

```
## [1] 0
```

Precision

```
## [1] 0.9372574
```

KS

AUC

Gini

Concordance - Discordance

METRICS - Naive Bayes

Model 4: KNN Model

```
## k-Nearest Neighbors
##
## 63883 samples
##    16 predictor
##     2 classes: 'Defaulters', 'Non_Defaulters'
##
## Pre-processing: centered (22), scaled (22)
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 51106, 51107, 51106, 51107, 51106
## Resampling results across tuning parameters:
##
##   k  ROC        Sens       Spec
##   5  0.6815582  0.1227763  0.9895798
##   7  0.7013952  0.1180278  0.9919177
##   9  0.7169573  0.1080269  0.9931868
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

Predict Default class and probability for KNN Model

Creating Confusion Matrix

```
##    knn_model_predict_class
##     Defaulters Non_Defaulters
##   1         98            901
##   2        103          14868
```

Accuracy

```
## [1] 0.9371321
```

Sensitivity of Model on Test Data

```
## [1] 0.9309956
```

Specificity of models on Test Data

```
## [1] 0.006136506
```

Precision

```
## [1] 73.97015
```

KS

AUC

Gini

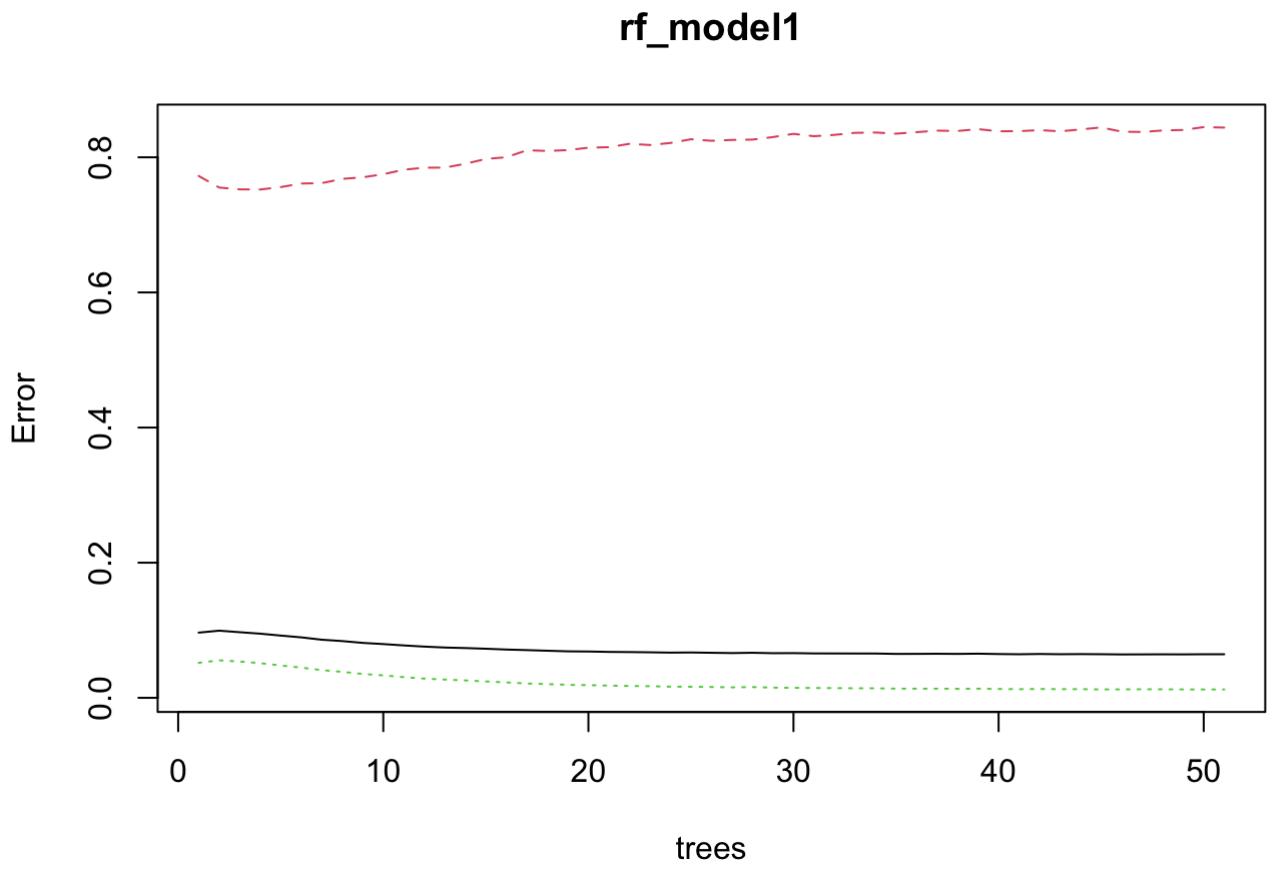
Concordance - Discordance

METRICS - KNN model

Model 5:Random Forest Model - BAGGING

Build the first RF model

```
##
## Call:
##  randomForest(formula = default ~ ., data = PremiumData_Train,      ntree = 51, mt
ry = 10, nodesize = 10, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 51
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 6.45%
## Confusion matrix:
##                Defaulters Non_Defaulters class.error
## Defaulters            624           3375  0.84396099
## Non_Defaulters        744          59140  0.01242402
```



*

The plot reveals that anything more than, say 25 trees, is really not that valuable.

```
##                                Defaulters Non_Defaulters MeanDecreaseAccuracy
## PercPremiumPaidbyCashCredit  0.0377318635   2.909855e-03         5.094211e-03
## Income                      -0.0068183643   1.135954e-02         1.022227e-02
## Count3to6monthsLate          0.0277452531   2.429606e-03         4.016983e-03
## Count6to12monthsLate         0.0590044233   5.482588e-03         8.835590e-03
## CountMoreThan12monthsLate    0.0211397552   2.885551e-03         4.030433e-03
## MaritalStatus                0.0003258127   5.489370e-05         7.226458e-05
## VehOwned                     0.0008394800   1.662384e-04         2.082725e-04
## NoofDep                      0.0003483739   1.110811e-04         1.258488e-04
## Accomodation                -0.0006320705   8.448429e-05         3.983830e-05
## riskScore                   -0.0017305112   8.295095e-03         7.667630e-03
## NoofPremiumsPaid            -0.0054863073   6.479943e-03         5.734032e-03
## SourcingChannel              0.0005821129   9.255772e-04         9.044261e-04
## ResidenceAreaType           -0.0003330769   1.087722e-05        -9.888526e-06
## premium                     -0.0056359467   7.080243e-03         6.284542e-03
## age                          0.0048228316   3.144410e-03         3.252619e-03
## riskscore_bins              -0.0016625405   2.649928e-03         2.379216e-03
##                             MeanDecreaseGini
## PercPremiumPaidbyCashCredit        717.02741
## Income                             758.20717
## Count3to6monthsLate                245.13160
## Count6to12monthsLate               664.07419
## CountMoreThan12monthsLate          199.72050
## MaritalStatus                       44.12219
## VehOwned                           114.91902
## NoofDep                            186.36907
## Accomodation                        46.61676
## riskScore                          575.85439
## NoofPremiumsPaid                   350.33260
## SourcingChannel                    209.58445
## ResidenceAreaType                   43.28510
## premium                            258.23699
## age                                488.79709
## riskscore_bins                      40.50699
```

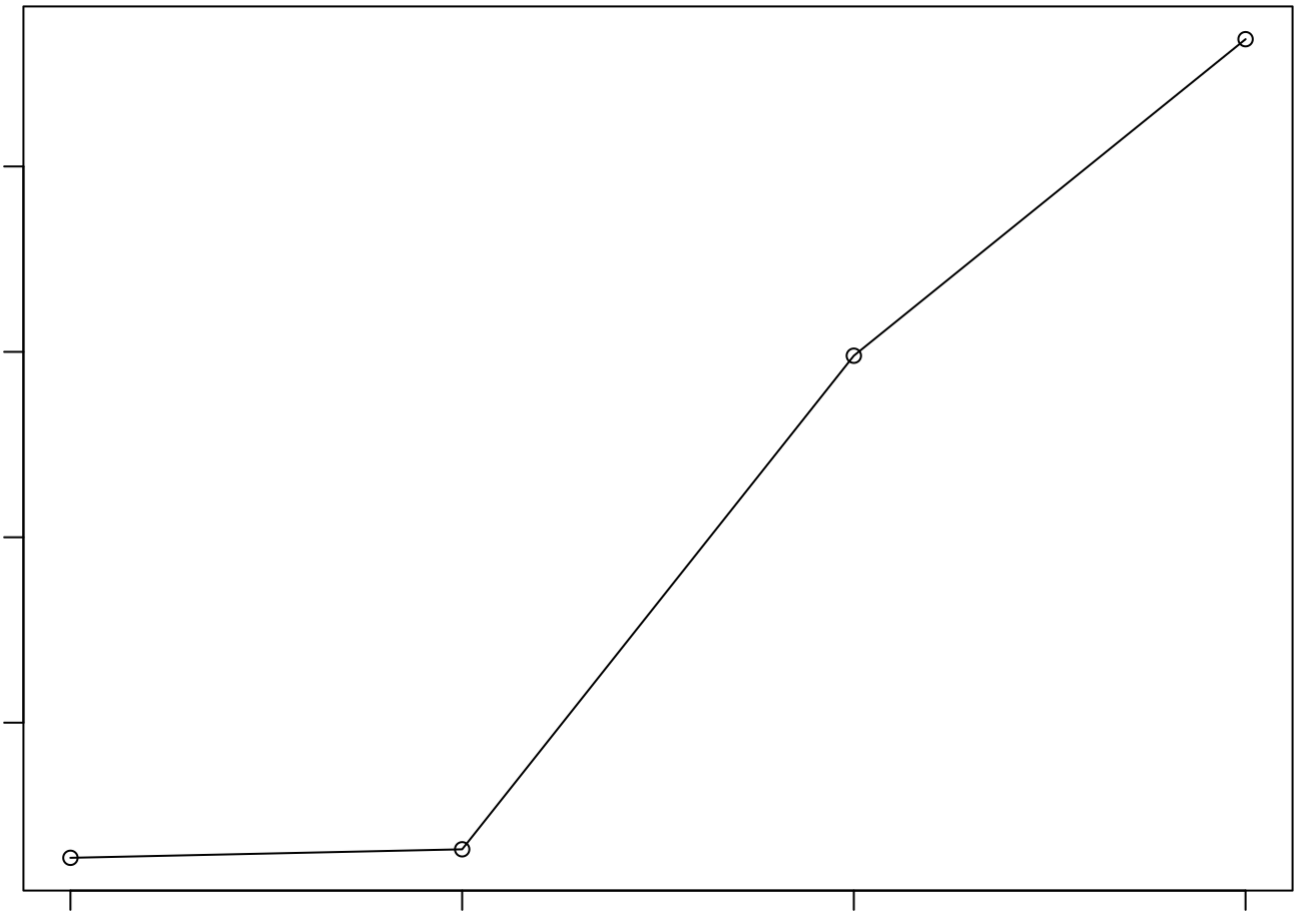
- Important variables as per RF1 are:

1. Count_6to12_months_late
2. No of Dep
3. risk_score
4. no_of_premiums_paid
5. premium
6. Veh_Owned
7. perc_premium_paid_by_cash_credit

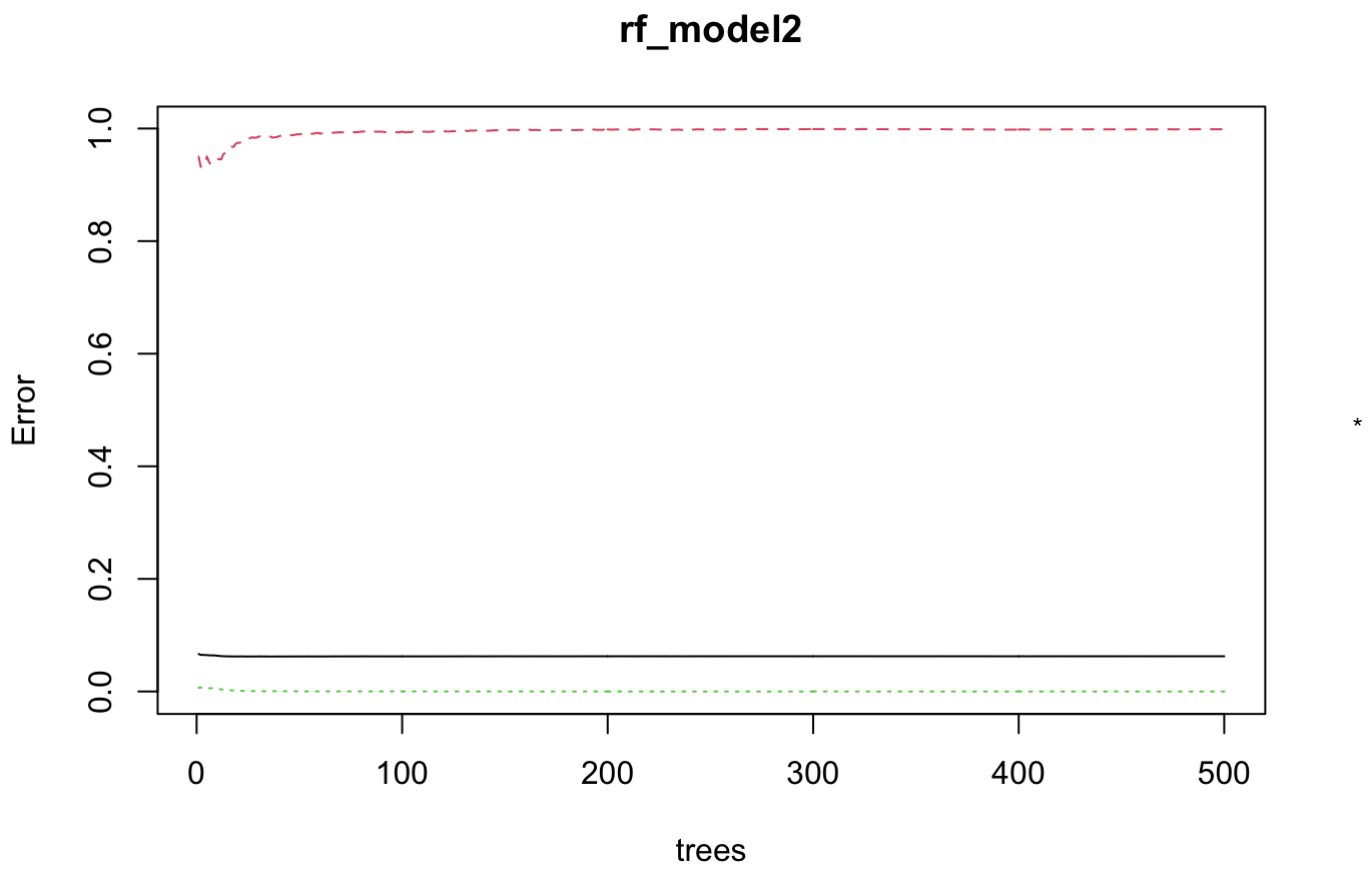
Random Forest Model 2

We will take ntree = 25 (odd number of trees are preferred)

```
## mtry = 4  OOB error = 6.5%
## Searching left ...
## mtry = 2     OOB error = 6.23%
## 0.040969 1e-04
## mtry = 1     OOB error = 6.23%
## 0.0007379374 1e-04
## Searching right ...
## mtry = 8     OOB error = 6.67%
## -0.07090666 1e-04
```



*The optimal number of mtry is 3.



*

tuneRF returns rf_model2. It is the random forest of 25 trees built with m = 1

Model Validation

```
##                 train_predict.class_RF
##                  Defaulters Non_Defaulters
##   Defaulters              30           3969
##   Non_Defaulters           0          59884
```

```
## [1] 0.9378708
```

- The Train set has 95.24% accuracy
- The Train set shows a good collection on True Positives & False Negatives

Model Evaluation

```
##    test_predict.class_RF
##     Defaulters Non_Defaulters
##   1          2            997
##   2          0          14971
```

```
## [1] 0.9375704
```

- The Test set has 93.86% accuracy
- Both model show a good accuracy and dont overfit the data, but the true positive identification is better on the Train set.

Predict Default class and probability for Random Forest

Creating Confusion Matrix

```
##    rf_model1_predict_class
##     Defaulters Non_Defaulters
##   1        159            840
##   2        154          14817
```

```
##    rf_model2_predict_class
##     Defaulters Non_Defaulters
##   1          2            997
##   2          0          14971
```

Random Forest Model 1 is able to identify the defaulters more correctly than RF Model 2

Accuracy

```
## [1] 0.9377583
```

```
## [1] 0.9375704
```

Sensitivity of Model on Test Data

```
## [1] 0.9278021
```

```
## [1] 0.9374452
```

Specificity of models on Test Data

```
## [1] 0.009956168
```

```
## [1] 0.0001252348
```

Precision

KS

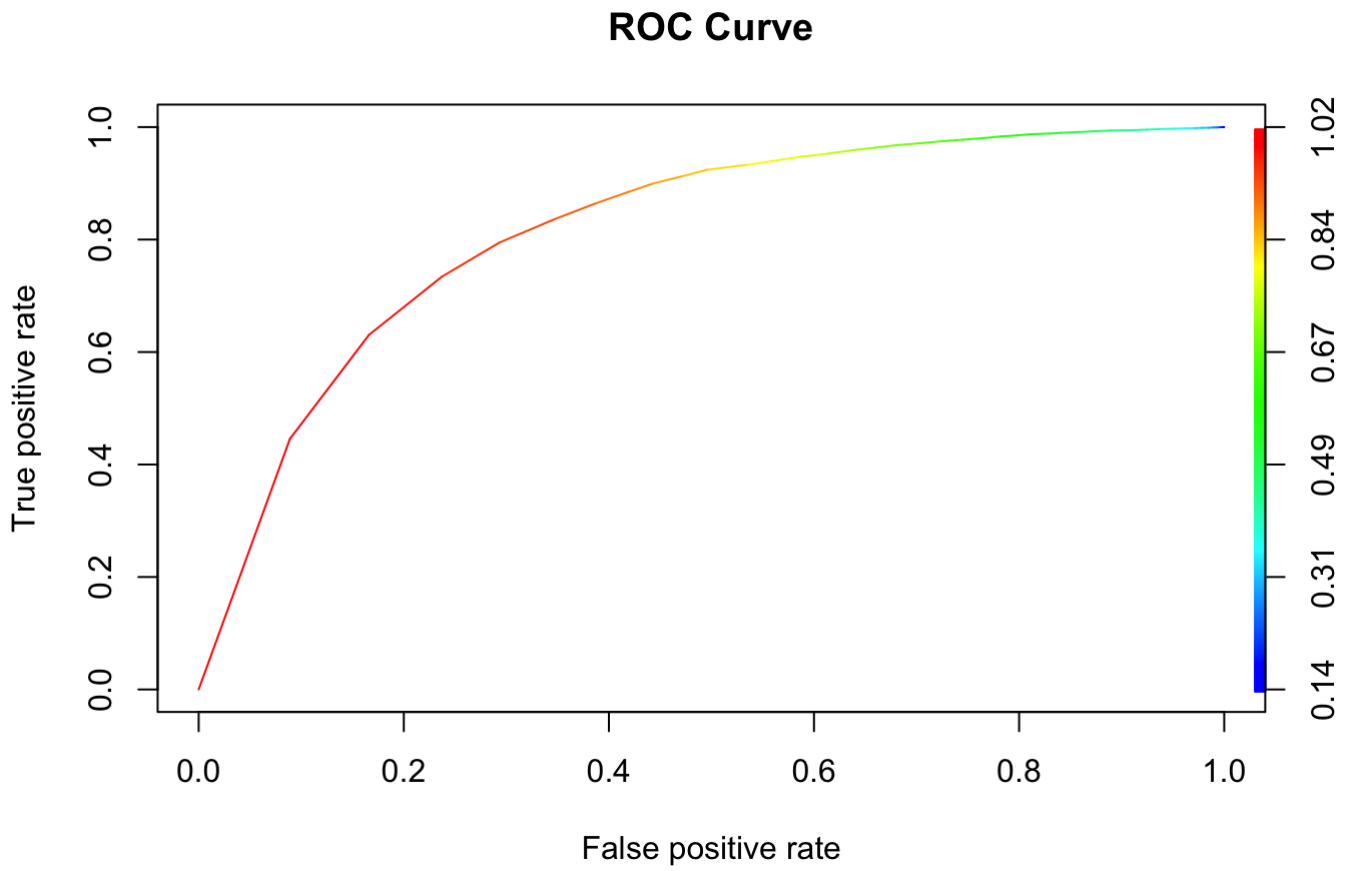
AUC

Gini

Concordance - Discordance

METRICS - Random Model 1 & 2

ROC Curves



We see from the curve that we can reach ~0.9 tpr at fpr ~ 0.4. Let us check the probability cutoff at that point.

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "False positive rate"
##   ..@ y.name      : chr "True positive rate"
##   ..@ alpha.name  : chr "Cutoff"
##   ..@ x.values    :List of 1
##   .. ..$ : num [1:46] 0 0.0891 0.1662 0.2372 0.2933 ...
##   ..@ y.values    :List of 1
##   .. ..$ : num [1:46] 0 0.446 0.631 0.734 0.795 ...
##   ..@ alpha.values:List of 1
##   .. ..$ : num [1:46] Inf 1 0.98 0.961 0.941 ...
```

	cut <dbl>	fpr <dbl>	tpr <dbl>
1	Inf	0.0000000	0.0000000
2	1.000	0.1171171	0.5067130
3	0.998	0.1981982	0.6810500
4	0.996	0.2532533	0.7599359
5	0.994	0.2942943	0.8043551
6	0.992	0.3233233	0.8344800

6 rows

Take fpr threshold = 0.4. Subset the dataframe cutoff to find the maximum tpr below this fpr threshold.

	cut <dbl>	fpr <dbl>	tpr <dbl>
9	0.986	0.3983984	0.8818382
8	0.988	0.3793794	0.8720192
7	0.990	0.3593594	0.8549195
6	0.992	0.3233233	0.8344800
5	0.994	0.2942943	0.8043551
4	0.996	0.2532533	0.7599359

6 rows

Take probability cut off= 0.934. and predict attrition using rf_model2.

New RF Model

```
##    class_prediction_with_new_cutoff
##         0     1
##   1   706   293
##   2  3074 11897
```

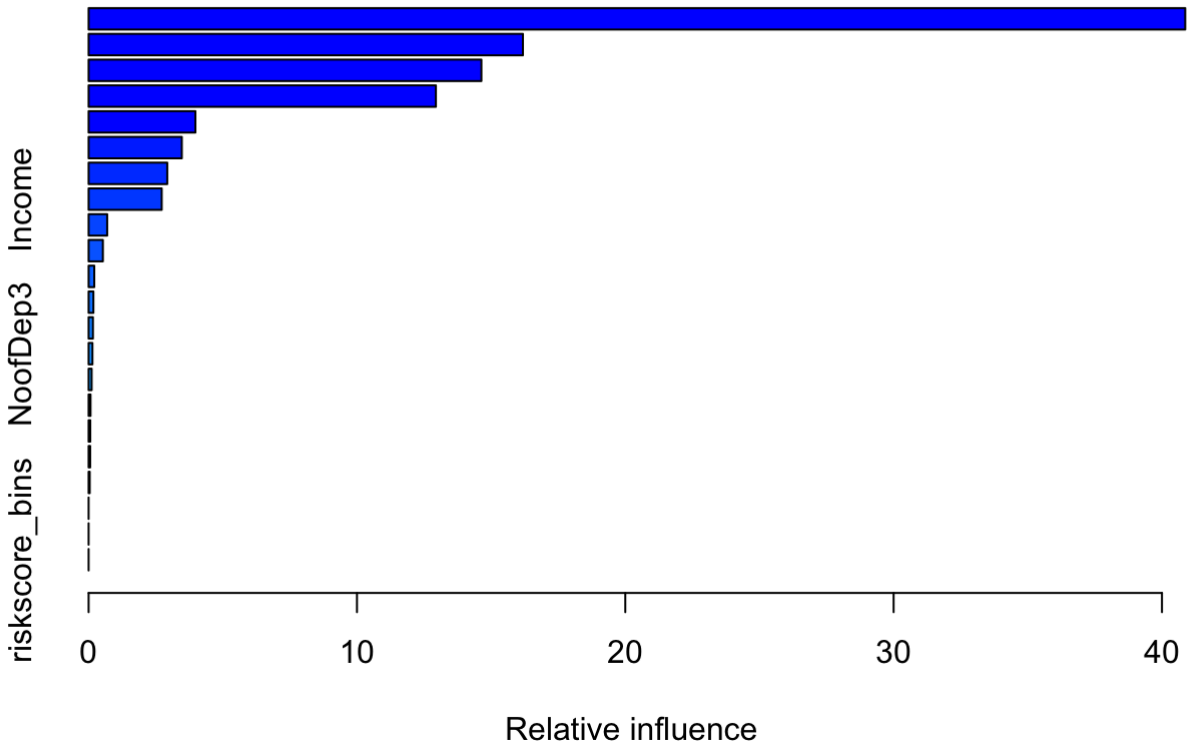
```
## [1] 0.7891672
```

```
## [1] 0.7449593
```

```
## [1] 0.04420789
```

- The new RF model shows lesser Accuracy, Sensitivity & Specificity then the earlier train & test models.

Model 6 : Gradient Boosting Machines



	var <chr>	rel.inf <dbl>
Count6to12monthsLate	Count6to12monthsLate	40.86764284
Count3to6monthsLate	Count3to6monthsLate	16.18443832
PercPremiumPaidbyCashCredit	PercPremiumPaidbyCashCredit	14.63737444
CountMoreThan12monthsLate	CountMoreThan12monthsLate	12.94106028
NoofPremiumsPaid	NoofPremiumsPaid	3.97942740
riskScore	riskScore	3.47353798
age	age	2.93066109
Income	Income	2.72438693
premium	premium	0.69199608
NoofDep4	NoofDep4	0.53162471

1-10 of 22 rows Previous **1** 2 3 Next

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction       Defaulters Non_Defaulters
##   Defaulters            615            383
##   Non_Defaulters       3384          59501
##
##                Accuracy : 0.941
##                  95% CI : (0.9392, 0.9428)
##     No Information Rate : 0.9374
##     P-Value [Acc > NIR] : 6.849e-05
##
##                   Kappa : 0.2268
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.153788
##             Specificity : 0.993604
##          Pos Pred Value : 0.616232
##          Neg Pred Value : 0.946187
##              Prevalence : 0.062599
##          Detection Rate : 0.009627
##    Detection Prevalence : 0.015622
##       Balanced Accuracy : 0.573696
##
##        'Positive' Class : Defaulters
##
```

Predict Default class and probability for Gradient Boost Model

Creating Confusion Matrix

```
##    gbm_model_predict_class
##     Defaulters Non_Defaulters
##   1        147            852
##   2        126          14845
```

Accuracy

```
## [1] 0.9387602
```

Sensitivity of Model on Test Data

```
## [1] 0.9295554
```

Specificity of models on Test Data

```
## [1] 0.009204759
```

Precision

```
## [1] 0.9295554
```

KS

AUC

Gini

Concordance - Discordance

METRICS - Gradient Boosting Model

```
## [1] 0.938760175 0.929555416 0.009204759 0.929555416          NA 0.828231879
## [7]          NA         NaN
```

Model 7 : Xtreme Gradient boosting Machines

Predict using the trained model & check performance on test set

Predict Default class and probability for XG Boost

Creating Confusion Matrix

```
##    xgb_model_predict_class
##     Defaulters Non_Defaulters
##   1        113            886
##   2         81          14890
```

Accuracy

```
## [1] 0.939449
```

Sensitivity of Model on Test Data

```
## [1] 0.9323732
```

Specificity of models on Test Data

```
## [1] 0.007075767
```

Precision

KS

AUC

Gini

Concordance - Discordance

METRICS - XG Boost model

MODELS COMPARISION

	cart_model1_metrics <dbl>	cart_model2_metrics <dbl>	rf_model1_metrics <dbl>	rf_model2
Accuracy	0.93462743	9.374452e-01	0.937758297	9.37
Sensitivity	0.92379461	9.374452e-01	0.927802129	9.37
Specificity	0.01083281	0.000000e+00	0.009956168	1.25

	cart_model1_metrics <dbl>	cart_model2_metrics <dbl>	rf_model1_metrics <dbl>	rf_model2
Precision	0.92379461	9.374452e-01	47.338658147	7.48
KS	0.52002928	0.000000e+00	0.501376401	5.11
Auc	0.80719234	5.000000e-01	0.818309158	8.17
Gini	0.03904797	-2.278008e-16	0.050461781	1.15
Concordance	0.17065151	0.000000e+00	NaN	

8 rows | 1-5 of 10 columns

Interpreting the best model:

Having tried out various models and techniques, we have to realise the variables which are important from the perspective of the basic objective of the project i.e. **identifying upfront, the customers who have the propensity to default of their premium payments**.

1. **Confusion Matrix**

**Models	True Positive
CART Model1	173
CART Model2	0
Logistic Regression	122
Naive Bayes	0
KNN	98
RF Model1	159
RF Model2	2
Gradient Boost	147
XG Boost	113

2. **Specificity** will highlight the non-defaulters, which will help us identify and go after the other cohort i.e. the Defaulters. We have set the positive class to 1 (non defaulters) hence we will look at improving and checking which model performs better on the Specificity. Among all the models we see that the **Cart Model 1** performs better than Random Forest 1 and other models.
3. **Sensitivity** The indicator for identifying on the defaulters would involve looking at the Sensitivity which identifies the positive class. **All the models are giving very high Sensitivity score (above 92)**. Hence we can safely say all models (apart from Gradient Boost Model) qualify on the Sensitivity threshold.
4. **Accuracy** will tell us the proportion of the Defaulters & Non-Defaulters which also forms an important indicator while selecting the right model.

All the models are giving very high Accuracy score (above 93). Hence we can safely say all models qualify on the Accuracy threshold.

5. **AUC ROC** which is performance measurement for classification at various thresholds, ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the

model is capable of distinguishing between the defaulters and non-defaulters. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes

**Models	AUC**
Cart Model 1	80
RF Model 1	81
KNN	73
Gradient Boost	82
XG Boost	82

All the have a good AUC score

6. **Kolmogorov-Smirnov Test** will help us decide if the sample comes from a population with specific distribution. The p-value returned by the k-s test has the same interpretation as other p-values. You reject the null hypothesis that the two samples were drawn from the same distribution if the p-value is less than your significance level.

All the models reject the Null Hypothesis that the two samples are from the same distribution, so are good on the KS Test.

7. **GINI coefficient** which is measure of the distribution of defaulters across the population is a good indicator of inequality in the population. A level 0 would indicate all are same (equality) and as the numbers increase, so does the degree of inequality in the population.

**Models	GINI**
Cart Model 1	0.3
RF Model 1	0.5
Logistic Regression	0.3
KNN	0.4
Gradient Boost	0.8
XG Boost	0.8

Cart Model & Logistic Regression models shows least degree of inequality compared to others. **This would be a crucial indicator to consider between Cart Model 1 & Random Forest 1, who seem to be filtering out as the most preferred Models to work upon.**

8. **Precision - Positive Predictive Value (PPV)** is the probability that customers identified with positive test as defaulters are truly defaulters, i.e. how often is a positive test represents true positive.

**Models	Precision**
Cart Model 1	37.73
RF Model 1	50.27
Logistic Regression	69.33
KNN	61.33
Gradient Boost	0.92

**Models	Precision**
XG Boost	0.93

Precision levels between Cart Model 1 & RF 1 would be of contention here as we need to decide between these models as they tick the boxes on most verticals.

CONCLUSION

CART Model 1 emerges as the better models to work upon.

- On the Specificity, we see that the CART Model 1 performs better than all the other models and has a edge over the Random Forest Model.
- When we look at the Sensitivity of the CART models it perform equally good among all the models.
- The overall performance of the CART Model 1 on Accuracy, Precision, AUC, KS, GINI also is better than most models and it it seems most versatile to filter the 'defaulters' for our analysis.

CART Model 1 will be most efficient in identifying the defaulters.

3. Relevance and implementability of the conclusions and recommendations

Business Insights:

- Having worked out ways to identify the cohort of customers who could have the prophecy to defaulter, we will need to dive a bit deeper into our findings to discover the patterns and behaviors of these customers. This will be important because we will eventually need to address these issues and provide easy to follow options and solutions to push them to pay their premiums on time.
- Our findings show that some aspects are prominent among these cohorts of customers, namely
    - The customers who pay their premium late by 3 to 6 months
    - The customers who pay their premium late by 6 to 12 months
    - The customers who pay their premium late by more than 12 months
    - Premium Paid in Cash Credit
    - Risk Score - customers with low risk score tend to default
    - Income
    - Premiums paid is a good indicator
    - Number of premiums paid also is a good indicator
- **Income** surely seems to be the factor which influences the payment of premiums. Higher Income customers also seem to perform better on the Risk Score which suggests that they also pay their premiums on time.
- The delay in paying premiums in all three verticals (i.e.**The customers who pay their premium late by 3 to 6 months, 6 to 12 months and by more than 12 months**) seem to follow a similar trend in the user behavior patterns. This indicates a behavior and pattern where a delinquent is eventually turning into a defaulter. It will be very effective to identify such delinquent customers and address their challenges for paying their premium on time by understanding their issues and coming with tailored solutions to help them pay their premiums on time.
- The delay in paying premiums affects the **risk score** and the risk score of defaulters indicates the pattern.
- **Premiums paid in Cash Credit** is also strong indicator and this seems to influence the payment of premiums on time.

- There is a mindset developed among the defaulters which seems go un-addressed which is resulting in having the insurance and the premiums to be paid for them at a very low priority which needs to be addressed by providing relevant solutions and offerings.

Recommendations:

- We need to keep in mind the current economic fallout due to the pandemic and the impact it has had on jobs/ business/income when we come our with solutions for the payments of the premiums. This could be in the lines of:
- Creating a proactive product which is rightly customized for those whose situation has/could deteriorate during the pandemic. Loss of Income/Jobs/Business could result in physical/mental deterioration for self and family which needs to be addressed. In other words **the solutions will have to be predominantly from the customers point of view**.
- The tendency to stay insured and pay their premiums on time is low among the defaulters and this needs to be addressed. The tendency to default on the premiums may not necessarily and singularly be because of they can't afford to pay the premium.
- In the current situation rather than providing a long term offering, what needs to be offered would be a solution which addresses the current issue in hand and something that will rightly fit in their scheme of things.
- A comprehensive but simple product which is easy to issue and pick needs to be developed.
- Multiple bouquets or sachet options for easy buy for specific plan or specific period
- Options to pay premiums at various time plans like bi-annual, quarterly, monthly etc. depending on the kind of customer we target to address.
- The issues of paying premiums on time and necessarily "in cash" needs to be addressed.
- The importance of having insurance for self and family has to be driven home and an inclusive and interactive product/solution should be created involving all stakeholders by keeping the customer at the center.
- We could approach the insurance company to get some more information to further distill and identify the cohorts with reference to some information which may not be provided in the shared data. This could include information like Gender, Type of Insurance (purchased/ defaulted), frequency of the premiums to be paid, etc.
- At times the reason for defaulting on premiums may not necessarily be the inability to pay the premium but could be the complex efforts and complicated procedures which a customer may shy away from.

Example of a Product Solution to reduce defaults in paying premiums:

- To address the ground situation and issues faced currently by customers, some tailor made small ticket solutions/ packages can be introduced which are simple products, easy to issue and pick. These could have solutions of addressing specific illness or issues which are tailored and less comprehensive and more affordable.
- There should be an interactive and proactive program created for customers, like for example the currently prevalent digital mark and measure vitality programs which are run by insurance companies where the customers are engaged in the program by having their interactive App on the customers smartphones where the customer is incentivised and rewarded for having a healthier lifestyle. Such vitality program engage and monitor the lifestyle of the customer and pushes them to maintain a healthy walk-sleep-eat-live pattern which is incentivised by offering reward points which could be converted into:
    - Decrease premium as per the healthier scores
    - offering cashless payments in lieu of the reward points generated

- using the reward points for shopping cashless from the various on-boarded merchants
- sharing the health score card of the customers with various other financial, health, business where the customers credit worth could be consolidated by the healthy reports which will increase the credibility and help him/her to get the needed service.
- such solutions will push them to maintain a healthy lifestyle which will help them stay mentally/physically fit and positive during challenging times.
- if the customer shows a good initiative and (their health and lifestyle suggests lower chances of mortality), the insurance company can provide options towards deferring the premium amount/ period according to the customers requirement during challenging times. Insurance companies would be happy to provide solutions to retain their customers as long as their health isn't an issue to worry.
- A healthy and stress free mind and body would be less dependent of drugs and medicine, will have a tendency to stay fit and positive and have a mindset which will want to prioritize health i.e. also stay insured and pay premiums on time.

Code Appendix

```
knitr::opts_chunk$set(error = FALSE,     # suppress errors
                      message = FALSE,   # suppress messages
                      warning = FALSE,   # suppress warnings
                      echo = FALSE,      # suppress code
                      cache = TRUE)      # enable caching
library(ggplot2) # For graphs and visualisations
library(gridExtra) # To plot multiple ggplot graphs in a grid
library(DataExplorer) # To plot correlation plot between numerical variables
library(caTools) # Split Data into Test and Train Set
library(rpart) # To build CART decision tree
library(rattle) # To visualise decision tree
library(randomForest) # To build a Random Forest
library(ROCR) # To visualise the performance classifiers
library(ineq) # To calculate Gini
library(InformationValue) # For Concordance-Discordance
library(readxl) # to read excel file
library(dplyr) #provides a set of tools for efficiently manipulating datasets
library(caTools) # Split Data into Test and Train Set
library(caret) # for confusion matrix function
library(corrplot) # for a graphical display of a correlation matrix, confidence inter
val or general matrix.
library(randomForest) # to build a random forest model
library(rpart.plot) # to plot decision tree model
library(xgboost) # to build a XG Boost model
library(DMwR) # for SMOTE
library(naivebayes) # for implementation of the Naive Bayes
library(e1071) # to train SVM & obtain predictions from the model
library(mlr) # for a generic, object-oriented, and extensible framework
library(gbm) #For power-users with many variables
library(car) # use for multicollinearity test (i.e. Variance Inflation Factor(VIF))
library(MASS) # for step AIC
library(grid) # for the primitive graphical functions
library(ROCR) # To plot ROC-AUC curve
library(InformationValue) # for Concordance-Discordance
library(class) # to build a KNN model
library(knitr) # Necessary to generate sourcecodes from a .Rmd File
library(parallel)
library(ipred)
library(psych)
library(olsrr) #use for multicollinearity test
r <- mclapply(1:10, function(i) {
  Sys.sleep(10)  ## Do nothing for 10 seconds
}, mc.cores = 10)      ## Split this job across 10 cores
require("knitr")
opts_knit$set(root.dir = "/Users/rajeevnitnawre/Downloads/DSBA/Capstone Project - Ins
urance/Project 1/")
PremiumData= read_excel("Insurance Premium Default-Dataset.xlsx")
plot_str(PremiumData)
dim(PremiumData)
# Look at the first and last few rows to ensure that the data is read in properly
head(PremiumData)
tail(PremiumData)
str(PremiumData)
plot_intro(PremiumData)
summary(PremiumData)
```

```
colnames(PremiumData)
PremiumData <- PremiumData %>%
  mutate(age =  age_in_days/ 365.2425)
PremiumData$age=as.integer(PremiumData$age)
PremiumData$Income<- PremiumData$Income/1000
age=PremiumData$age
PremiumData$agegroup=cut(age,8,labels = c('1','2','3','4','5','6','7','8'))
PremiumData$age= round(as.numeric(PremiumData$age),0)
risk_score=PremiumData$risk_score
PremiumData$riskscore_bins=cut(risk_score,9,labels = c('1','2','3','4','5','6','7',
'8','9'))
PremiumData = subset(PremiumData, select = -c(id,age_in_days))
dim(PremiumData)
names(PremiumData)[1]<-paste("PercPremiumPaidbyCashCredit")
names(PremiumData)[3]<-paste("Count3to6monthsLate")
names(PremiumData)[4]<-paste("Count6to12monthsLate")
names(PremiumData)[5]<-paste("CountMoreThan12monthsLate")
names(PremiumData)[11]<-paste("NoofPremiumsPaid")
names(PremiumData)[12]<-paste("SourcingChannel")
names(PremiumData)[13]<-paste("ResidenceAreaType")
names(PremiumData)[7]<-paste("VehOwned")
names(PremiumData)[8]<-paste("NoofDep")
names(PremiumData)[10]<-paste("riskScore")
names(PremiumData)[6]<-paste("MaritalStatus")
  outlier_treatment_fun_age = function(PremiumData,age){
  capping = as.vector(quantile(PremiumData[,age],0.99))
  flooring = as.vector(quantile(PremiumData[,age],0.01))
  PremiumData[,age][which(PremiumData[,age]<flooring)]= flooring
  PremiumData[,age][which(PremiumData[,age]>capping)]= capping
  #print('done',var_name)
  return(PremiumData)
}
new_vars = c('age','Income','premium','NoofPremiumsPaid','PercPremiumPaidbyCashCredi
t',
             'Count3to6monthsLate','Count6to12monthsLate','CountMoreThan12monthsLate'
)
outlier_treatment_fun_Income = function(PremiumData,Income){
  capping = as.vector(quantile(PremiumData[,Income],0.99))
  flooring = as.vector(quantile(PremiumData[,Income],0.01))
  PremiumData[,Income][which(PremiumData[,Income]<flooring)]= flooring
  PremiumData[,Income][which(PremiumData[,Income]>capping)]= capping
  #print('done',var_name)
  return(PremiumData)
}
outlier_treatment_fun_Premium = function(PremiumData,premium){
  capping = as.vector(quantile(PremiumData[,premium],0.99))
  flooring = as.vector(quantile(PremiumData[,premium],0.01))
  PremiumData[,premium][which(PremiumData[,premium]<flooring)]= flooring
  PremiumData[,premium][which(PremiumData[,premium]>capping)]= capping
  #print('done',var_name)
  return(PremiumData)
}
outlier_treatment_fun_No_Premium = function(PremiumData,NoofPremiumsPaid){
  capping = as.vector(quantile(PremiumData[,NoofPremiumsPaid],0.99))
  flooring = as.vector(quantile(PremiumData[,NoofPremiumsPaid],0.01))
  PremiumData[,NoofPremiumsPaid][which(PremiumData[,NoofPremiumsPaid]<flooring)]= flo
```

```
oring
  PremiumData[,NoofPremiumsPaid][which(PremiumData[,NoofPremiumsPaid]>capping)]= capp
ing
  #print('done',var_name)
  return(PremiumData)
}
outlier_treatment_fun_No_Premium = function(PremiumData,PercPremiumPaidbyCashCredit){
  capping = as.vector(quantile(PremiumData[,PercPremiumPaidbyCashCredit],0.99))
  flooring = as.vector(quantile(PremiumData[,PercPremiumPaidbyCashCredit],0.01))
  PremiumData[,PercPremiumPaidbyCashCredit][which(PremiumData[,PercPremiumPaidbyCashC
redit]<flooring)]= flooring
  PremiumData[,PercPremiumPaidbyCashCredit][which(PremiumData[,PercPremiumPaidbyCashC
redit]>capping)]= capping
  #print('done',var_name)
  return(PremiumData)
}
 outlier_treatment_fun_3to6 = function(PremiumData,Count3to6monthsLate){
  capping = as.vector(quantile(PremiumData[,Count3to6monthsLate],0.99))
  flooring = as.vector(quantile(PremiumData[,Count3to6monthsLate],0.01))
  PremiumData[,Count3to6monthsLate][which(PremiumData[,Count3to6monthsLate-months_lat
e]<flooring)]= flooring
  PremiumData[,Count3to6monthsLate][which(PremiumData[,Count3to6monthsLate-months_lat
e]>capping)]= capping
  #print('done',var_name)
  return(PremiumData)
}
 outlier_treatment_fun_6to12 = function(PremiumData,Count6to12monthsLate){
  capping = as.vector(quantile(PremiumData[,Count6to12monthsLate],0.99))
  flooring = as.vector(quantile(PremiumData[,Count6to12monthsLate],0.01))
  PremiumData[,Count6to12monthsLate][which(PremiumData[,Count6to12monthsLate]<floorin
g)]= flooring
  PremiumData[,Count6to12monthsLate][which(PremiumData[,Count6to12monthsLate]>cappin
g)]= capping
  #print('done',var_name)
  return(PremiumData)
}
 outlier_treatment_fun_ = function(PremiumData,CountMoreThan12monthsLate){
  capping = as.vector(quantile(PremiumData[,CountMoreThan12monthsLate],0.99))
  flooring = as.vector(quantile(PremiumData[,CountMoreThan12monthsLate],0.01))
  PremiumData[,CountMoreThan12monthsLate][which(PremiumData[,CountMoreThan12monthsLat
e]<flooring)]= flooring
  PremiumData[,CountMoreThan12monthsLate][which(PremiumData[,CountMoreThan12monthsLat
e]>capping)]= capping
  #print('done',var_name)
  return(PremiumData)
}

PremiumData$MaritalStatus=as.factor(PremiumData$MaritalStatus)
PremiumData$Accomodation=as.factor(PremiumData$Accomodation)
PremiumData$default=as.factor(PremiumData$default)
PremiumData$VehOwned=as.factor(PremiumData$VehOwned)
PremiumData$NoofDep=as.factor(PremiumData$NoofDep)
PremiumData$default=as.factor(PremiumData$default)
PremiumData$SourcingChannel=as.factor(PremiumData$SourcingChannel)
PremiumData$ResidenceAreaType=as.factor(PremiumData$ResidenceAreaType)
prop.table(table(PremiumData$default))*100
```

```
plot_histogram_n_boxplot = function(variable, variableNameString, binw){
  h = ggplot(data = PremiumData, aes(x= variable))+
    labs(x = variableNameString,y ='count')+
    geom_histogram(fill = 'green',col = 'white',binwidth = binw)+
    geom_vline(aes(xintercept=mean(variable)),
               color="black", linetype="dashed", size=0.5)
  b = ggplot(data = PremiumData, aes('',variable))+
    geom_boxplot(outlier.colour = 'red',col = 'red',outlier.shape = 19)+
    labs(x = '',y = variableNameString)+ coord_flip()
  grid.arrange(h,b,ncol = 2)
}

plot_histogram_n_boxplot(PremiumData$age,"Age",1)
PremiumData$agegroup=as.numeric(PremiumData$agegroup)
plot_histogram_n_boxplot(PremiumData$agegroup,"Age Group",1)
plot_histogram_n_boxplot(PremiumData$Income,"Income",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$PercPremiumPaidbyCashCredit,"Premium Paid in Cas
h",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$Count3to6monthsLate,"Premium late by 3 to 6 mont
hs",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$Count6to12monthsLate,"Premium late by 6 to 12 mo
nths",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$CountMoreThan12monthsLate,"Premium more than 12
 months late",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$riskScore,"Risk Score",1)
fig.align = 'left'
PremiumData$riskscore_bins=as.numeric(PremiumData$riskscore_bins)
plot_histogram_n_boxplot(PremiumData$riskscore_bins,"Risk Score Bins",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$premium,"Premium",1)
fig.align = 'left'
plot_histogram_n_boxplot(PremiumData$NoofPremiumsPaid,"Number of Premium Paid",1)
unipar = theme(legend.position = "none") +
  theme(axis.text = element_text(size = 10),
        axis.title = element_text(size = 11),
        title = element_text(size = 13, face = "bold"))

# Define color brewer
col1 = "Set2"

g1=ggplot(PremiumData, aes(x=MaritalStatus, fill=MaritalStatus)) + geom_bar()+ unipar
+ scale_fill_brewer(palette=col1) +
  geom_text(aes(label = scales::percent(..prop..), group = 1), stat= "count", size =
3.3, position = position_stack(0.06))+
  geom_text(aes(label = ..count.., group = 1), stat= "count", size = 3.3, position =
 position_stack(0.95))

g4=ggplot(PremiumData, aes(x=Accomodation, fill=Accomodation)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1) +
  geom_text(aes(label = scales::percent(..prop..), group = 1), stat= "count", size =
```

```
3.3, position = position_stack(0.06))+
  geom_text(aes(label = ..count.., group = 1), stat= "count", size = 3.3, position =
 position_stack(0.95))

g6=ggplot(PremiumData, aes(x=ResidenceAreaType, fill=ResidenceAreaType)) + geom_bar()
+ unipar + scale_fill_brewer(palette=col1) +
  geom_text(aes(label = scales::percent(..prop..), group = 1), stat= "count", size =
3.3, position = position_stack(0.06))+
  geom_text(aes(label = ..count.., group = 1), stat= "count", size = 3.3, position =
 position_stack(0.95))

fig.align = 'left'
grid.arrange(g1,g4,g6,ncol=3)
g2=ggplot(PremiumData, aes(x=VehOwned, fill=VehOwned)) + geom_bar()+ unipar + scale_f
ill_brewer(palette=col1) +
  geom_text(aes(label = scales::percent(..prop..), group = 1), stat= "count", size =
3.3, position = position_stack(0.06))+
  geom_text(aes(label = ..count.., group = 1), stat= "count", size = 3.3, position =
 position_stack(0.95))

g3=ggplot(PremiumData, aes(x= NoofDep, fill=NoofDep)) + geom_bar()+ unipar + scale_fi
ll_brewer(palette=col1) +
  geom_text(aes(label = scales::percent(..prop..), group = 1), stat= "count", size =
3.3, position = position_stack(0.06))+
  geom_text(aes(label = ..count.., group = 1), stat= "count", size = 3.3, position =
 position_stack(0.95))


g5=ggplot(PremiumData, aes(x=SourcingChannel, fill=SourcingChannel)) + geom_bar()+ un
ipar + scale_fill_brewer(palette=col1) +
  geom_text(aes(label = scales::percent(..prop..), group = 1), stat= "count", size =
3.3, position = position_stack(0.06))+
  geom_text(aes(label = ..count.., group = 1), stat= "count", size = 3.3, position =
 position_stack(0.95))

fig.align = 'left'
grid.arrange(g2,g3,g5,ncol=3)
fig.align = 'left'
par(mfrow = c(3,2));

text(x= barplot(table(PremiumData$age),col='#69b3a2', main = "Age",ylab = "Frequency"
),
     y = 0, table(PremiumData$age), cex=1,pos=1);
boxplot(PremiumData$age, col = "steelblue", horizontal = TRUE, main = "Age");
text(x = fivenum(PremiumData$age), labels = fivenum(PremiumData$age), y = 1.25)

text(x= barplot(table(PremiumData$agegroup),col='#69b3a2', main = "Age Group",ylab =
"Frequency"),
     y = 0, table(PremiumData$age), cex=1,pos=1);
boxplot(PremiumData$agegroup, col = "steelblue", horizontal = TRUE, main = "Age Grou
p");
text(x = fivenum(PremiumData$agegroup), labels = fivenum(PremiumData$agegroup), y =
1.25)

text(x= barplot(table(PremiumData$riskScore),col='#69b3a2', main = "Risk Score",ylab
 = "Frequency"),
```

```
y = 0, table(PremiumData$riskScore), cex=1,pos=1); boxplot(PremiumData$riskScore, col
= "steelblue", horizontal = TRUE, main = "Risk Score"); text(x = fivenum(PremiumData
$riskScore), labels = fivenum(PremiumData$riskScore), y = 1.25)

text(x= barplot(table(PremiumData$riskscore_bins),col='#69b3a2', main = "Riskscore Bi
ns",ylab = "Frequency"),
     y = 0, table(PremiumData$riskscore_bins), cex=1,pos=1);
boxplot(PremiumData$riskscore_bins, col = "steelblue", horizontal = TRUE, main = "Ris
kscore Bins");
text(x = fivenum(PremiumData$riskscore_bins), labels = fivenum(PremiumData$riskscore_
bins), y = 1.25)

text(x= barplot(table(PremiumData$Income),col='#69b3a2', main = "Income",ylab = "Freq
uency"),
     y = 0, table(PremiumData$Income), cex=1,pos=1);
boxplot(PremiumData$Income, col = "steelblue", horizontal = TRUE, main = "Income");
text(x = fivenum(PremiumData$Income), labels = fivenum(PremiumData$Income), y = 1.25)

fig.align = 'left'
par(mfrow = c(3,2));

text(x= barplot(table(PremiumData$premium),col='#69b3a2', main = "Premium",ylab = "Fr
equency"), y = 0, table(PremiumData$premium), cex=1,pos=1); boxplot(PremiumData$premi
um, col = "steelblue", horizontal = TRUE, main = "Premium"); text(x = fivenum(Premium
Data$premium), labels = fivenum(PremiumData$premium), y = 1.25)

text(x= barplot(table(PremiumData$NoofPremiumsPaid),col='#69b3a2', main = "Number of
 Premiums Paid",ylab = "Frequency"),
y = 0, table(PremiumData$NoofPremiumsPaid), cex=1,pos=1); boxplot(PremiumData$NoofPre
miumsPaid, col = "steelblue", horizontal = TRUE, main = "Number of Premiums Paid"); t
ext(x = fivenum(PremiumData$no_of_premiums_paid), labels = fivenum(PremiumData$no_of_
premiums_paid), y = 1.25)

fig.align = 'left'
par(mfrow = c(3,2));
text(x= barplot(table(PremiumData$Count3to6monthsLate),col='#69b3a2', main = "Premium
late by 3-6 months",ylab = "Frequency"), y = 0, table(PremiumData$Count3to6monthsLat
e), cex=1,pos=1);
boxplot(PremiumData$Count3to6monthsLate, col = "steelblue", horizontal = TRUE, main =
"Premium late by 3-6 months");
text(x = fivenum(PremiumData$Count3to6monthsLate), labels = fivenum(PremiumData$Count
3to6monthsLate), y = 1.25)

text(x= barplot(table(PremiumData$Count6to12monthsLate),col='#69b3a2', main = "Premiu
m late by 6-12 months",ylab = "Frequency"), y = 0, table(PremiumData$Count6to12months
Late), cex=1,pos=1);
boxplot(PremiumData$Count6to12monthsLate, col = "steelblue", horizontal = TRUE, main
 = "Premium late by 6 to 12 months");
text(x = fivenum(PremiumData$Count6to12monthsLate), labels = fivenum(PremiumData$Coun
t6to12monthsLate), y = 1.25)

text(x= barplot(table(PremiumData$CountMoreThan12monthsLate),col='#69b3a2', main = "P
remium late by more than 12 months",ylab = "Frequency"), y = 0, table(PremiumData$Cou
ntMoreThan12monthsLate), cex=1,pos=1);
boxplot(PremiumData$CountMoreThan12monthsLate, col = "steelblue", horizontal = TRUE,
 main = "Premium late by more than 12 months"); text(x = fivenum(PremiumData$CountMor
```

```
eThan12monthsLate), labels = fivenum(PremiumData$CountMoreThan12monthsLate), y = 1.25
)

bipar1 = theme(legend.position = "none") + theme_light() +
  theme(axis.text = element_text(size = 10),
        axis.title = element_text(size = 11),
        title = element_text(size = 13, face = "bold"))

# Define color brewer
col2 = "Set2"
fig.align = 'left'
p=ggplot(PremiumData, aes(x = default, y = age, fill = default)) + geom_boxplot(show.
legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary(fun = quantil
e, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_nudge(x=0.5),
 size=4, color = "black") + coord_flip()

p1=ggplot(PremiumData, aes(x = default, y = agegroup, fill = default)) + geom_boxplot
(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary(fun = q
uantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_nudge(x=
0.5), size=4, color = "black") + coord_flip()

p2=ggplot(PremiumData, aes(x = default, y = Income, fill = default)) + geom_boxplot(s
how.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary(fun = qua
ntile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_nudge(x=0.
5), size=4, color = "black") + coord_flip()

grid.arrange(p,p1,p2,ncol=2)
fig.align = 'left'
p3=ggplot(PremiumData, aes(x = default, y = Count3to6monthsLate, fill = default)) + g
eom_boxplot(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summ
ary(fun = quantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=positi
on_nudge(x=0.5), size=4, color = "black") + coord_flip()

p4=ggplot(PremiumData, aes(x = default, y = Count6to12monthsLate, fill = default)) +
 geom_boxplot(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_su
mmary(fun = quantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=posi
tion_nudge(x=0.5), size=4, color = "black") + coord_flip()

p5=ggplot(PremiumData, aes(x = default, y = CountMoreThan12monthsLate, fill = defaul
t)) + geom_boxplot(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ st
at_summary(fun = quantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position
=position_nudge(x=0.5), size=4, color = "black") + coord_flip()

grid.arrange(p3,p4,p5, ncol=3)
fig.align = 'left'
p6=ggplot(PremiumData, aes(x = default, y = NoofPremiumsPaid, fill = default)) + geom
_boxplot(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary
(fun = quantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_
nudge(x=0.5), size=4, color = "black") + coord_flip()

p7=ggplot(PremiumData, aes(x = default, y = premium, fill = default)) + geom_boxplot
(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary(fun = q
uantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_nudge(x=
0.5), size=4, color = "black") + coord_flip()

grid.arrange(p6,p7,ncol=2)
```

```
fig.align = 'left'
p8=ggplot(PremiumData, aes(x = default, y = riskScore, fill = default)) + geom_boxplo
t(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary(fun =
 quantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_nudge
(x=0.5), size=4, color = "black") + coord_flip()

p9=ggplot(PremiumData, aes(x = default, y = riskscore_bins, fill = default)) + geom_b
oxplot(show.legend = FALSE)+ bipar1 + scale_fill_brewer(palette=col2)+ stat_summary(f
un = quantile, geom = "text", aes(label=sprintf("%1.0f", ..y..)),position=position_nu
dge(x=0.5), size=4, color = "black") + coord_flip()

grid.arrange(p8,p9,ncol=2)
bipar2 = theme(legend.position = "top",
               legend.direction = "horizontal",
               legend.title = element_text(size = 10),
               legend.text = element_text(size = 8)) +
  theme(axis.text = element_text(size = 10),
        axis.title = element_text(size = 11),
        title = element_text(size = 13, face = "bold"))
library(dplyr)

d1 <- PremiumData %>% group_by(MaritalStatus) %>% count(default) %>% mutate(ratio=sca
les::percent(n/sum(n)))
p8=ggplot(PremiumData, aes(x=MaritalStatus, fill=default)) + geom_bar()+ bipar2 + sca
le_fill_brewer(palette=col2) +
  geom_text(data=d1, aes(y=n,label=ratio),position=position_stack(vjust=0.5))

d2 <- PremiumData %>% group_by(VehOwned) %>% count(default) %>% mutate(ratio=scales::
percent(n/sum(n)))
p9=ggplot(PremiumData, aes(x=VehOwned, fill=default)) + geom_bar()+ bipar2 + scale_fi
ll_brewer(palette=col2) +
  geom_text(data=d2, aes(y=n,label=ratio),position=position_stack(vjust=0.5))

d3 <- PremiumData %>% group_by(NoofDep) %>% count(default) %>% mutate(ratio=scales::p
ercent(n/sum(n)))
p10=ggplot(PremiumData, aes(x= NoofDep, fill=default)) + geom_bar()+ bipar2 + scale_f
ill_brewer(palette=col2) +
  geom_text(data=d3, aes(y=n,label=ratio),position=position_stack(vjust=0.5))

d4 <- PremiumData %>% group_by(Accomodation) %>% count(default) %>% mutate(ratio=scal
es::percent(n/sum(n)))
p11=ggplot(PremiumData, aes(x=Accomodation, fill=default)) + geom_bar()+ bipar2 + sca
le_fill_brewer(palette=col2) +
  geom_text(data=d4, aes(y=n,label=ratio),position=position_stack(vjust=0.5))

d5 <- PremiumData %>% group_by(ResidenceAreaType) %>% count(default) %>% mutate(ratio
=scales::percent(n/sum(n)))
p12=ggplot(PremiumData, aes(x=ResidenceAreaType, fill=default)) + geom_bar()+ bipar2
 + scale_fill_brewer(palette=col2) +
  geom_text(data=d5, aes(y=n,label=ratio),position=position_stack(vjust=0.5))



grid.arrange(p8,p9,p10,p11,p12,ncol=3)
fig.align = 'left'
plot_correlation(PremiumData[,c(-15,-17,-18)])
fig.align = 'left'
```

```
library(psych)
pairs.panels(PremiumData[,c(-15,-17,-18)],
             method = "pearson", # correlation method
             hist.col = "yellow",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
)
chisq.test(PremiumData$MaritalStatus,PremiumData$Accomodation)
chisq.test(PremiumData$ResidenceAreaType,PremiumData$VehOwned)
chisq.test(PremiumData$VehOwned,PremiumData$Accomodation)
chisq.test(PremiumData$NoofDep,PremiumData$VehOwned)
chisq.test(PremiumData$SourcingChannel,PremiumData$ResidenceAreaType)
chisq.test(PremiumData$MaritalStatus,PremiumData$ResidenceAreaType)
chisq.test(PremiumData$Accomodation,PremiumData$ResidenceAreaType)
chisq.test(PremiumData$ResidenceAreaType,PremiumData$VehOwned)
subset_PremiumData= PremiumData[, c("age","Income","riskScore","premium",
                                    "PercPremiumPaidbyCashCredit","Count3to6monthsLat
e",
                                    "Count6to12monthsLate","CountMoreThan12monthsLat
e",
                                    "NoofPremiumsPaid")]
new_vars = c('age','Income','premium','NoofPremiumsPaid','Count3to6monthsLate',
             'Count6to12monthsLate','CountMoreThan12monthsLate')

correlations = cor(PremiumData[,new_vars])

col1 <- colorRampPalette(c("#7F0000", "red", "#FF7F00", "yellow", "#7FFF7F",
                           "cyan", "#007FFF"))
corrplot(correlations,number.cex = 1,method = 'number',type = 'lower',col = col1(100
))
subset_PremiumData$default<-PremiumData$default
dim(subset_PremiumData)
colnames(subset_PremiumData)
newNamesMean = c("age","Income","premium", "riskScore")

bcM.data = (subset_PremiumData[,newNamesMean])

bcM.diag = subset_PremiumData[,10]
scales <- list(x=list(relation="free"),y=list(relation="free"), cex=10)
caret::featurePlot(x=bcM.data, y=bcM.diag, plot="pairs", scales=scales,pch=".")
newNamesMean = c("PercPremiumPaidbyCashCredit","Count3to6monthsLate","Count6to12month
sLate",
                 "CountMoreThan12monthsLate","NoofPremiumsPaid")

bcM.data = (subset_PremiumData[,newNamesMean])

bcM.diag = subset_PremiumData[,10]
scales <- list(x=list(relation="free"),y=list(relation="free"), cex=10)
caret::featurePlot(x=bcM.data, y=bcM.diag, plot="pairs", scales=scales,pch=".")
PremiumData$default<- as.factor(PremiumData$default)
levels(PremiumData$default) <- c("Defaulters", "Non_Defaulters")
PremiumData$default <- relevel(PremiumData$default, ref = "Defaulters") # Reference c
lass : Defaulter
levels(PremiumData$default)
PremiumData=PremiumData[,-17]
set.seed(123)
```

```
trainIndex <- createDataPartition(PremiumData$default, p = .80, list = FALSE)

PremiumData_Train <- PremiumData[ trainIndex,]
PremiumData_Test  <- PremiumData[-trainIndex,]

prop.table(table(PremiumData_Train$default))*100
prop.table(table(PremiumData_Test$default))*100
prop.table(table(PremiumData$default))*100
rm(PremiumData)
rm(subset_PremiumData)

table(PremiumData_Train$default)
prop.table(table(PremiumData_Train$default))

PremiumData_Train <- as.data.frame(PremiumData_Train)
PremiumData_Train$default <- as.factor(PremiumData_Train$default)


smote_train <- SMOTE(default ~ ., data  = PremiumData_Train[,c(-12,-13)],
                     perc.over = 3700,
                     perc.under = 300
                    )

prop.table(table(smote_train$default))*100
table(smote_train$default)
fitControl <- trainControl(
  method = 'repeatedcv',           # k-fold cross validation
  number = 5,                      # number of folds or k
  repeats = 1,                      # repeated k-fold cross-validation
  allowParallel = TRUE,
  classProbs = TRUE,
  summaryFunction=twoClassSummary# should class probabilities be returned
)
r.ctrl = rpart.control(minsplit = 50, minbucket = 10, cp = 0, xval = 10)
cart_model1 <- rpart(formula = default~., data = PremiumData_Train, method = "class",
control = r.ctrl)
printcp(cart_model1)
plotcp(cart_model1)
cart_model2 = prune(cart_model1, cp= 0.062 ,"CP")
printcp(cart_model2)
cart_model2
cart_model1$variable.importance
# Variable importance is generally computed based on the corresponding reduction of p
redictive accuracy
# when the predictor of interest is removed.
# Predicting on the train dataset
train_predict.class_CART <- predict(cart_model2, PremiumData_Train, type="class") # P
redicted Classes
train_predict.score_CART <- predict(cart_model2, PremiumData_Train) # Predicted Proba
bilities

# Create confusion matrix for train data predictions
tab.train_CART = table(PremiumData_Train$default, train_predict.class_CART)
tab.train_CART
```

```
# Accuracy on train data
accuracy.train_CART = sum(diag(tab.train_CART)) / sum(tab.train_CART)
accuracy.train_CART
# Predicting on the test dataset
test_predict.class_CART <- predict(cart_model2, PremiumData_Test, type="class") # Pre
dicted Classes
test_predict.score_CART <- predict(cart_model2, PremiumData_Test) # Predicted Probabi
lities

# Create confusion matrix for test data predictions
tab.test_CART = table(PremiumData_Test$default, test_predict.class_CART)
tab.test_CART

# Accuracy on test data
accuracy.test_CART = sum(diag(tab.test_CART)) / sum(tab.test_CART)
accuracy.test_CART

# Predict on test data using cart_model1
cart_model1_predict_class = predict(cart_model1, PremiumData_Test, type = 'class')
cart_model1_predict_score = predict(cart_model1, PremiumData_Test, type = 'prob')

# Predict on test data using cart_model2
cart_model2_predict_class = predict(cart_model2, PremiumData_Test, type = 'class')
cart_model2_predict_score = predict(cart_model2, PremiumData_Test, type = 'prob')
#Confusion Matrix of Cart_Model1
conf_mat_cart_model1 = table(PremiumData_Test$default, cart_model1_predict_class)
conf_mat_cart_model1

#Confusion Matrix of Cart_Model2
conf_mat_cart_model2 = table(PremiumData_Test$default, cart_model2_predict_class)
conf_mat_cart_model2
# Accuracy of cart Model 1
accuracy_cart_model1 = sum(diag(conf_mat_cart_model1)) / sum(conf_mat_cart_model1)
accuracy_cart_model1

# Accuracy of cart Model 2
accuracy_cart_model2 = sum(diag(conf_mat_cart_model2)) / sum(conf_mat_cart_model2)
accuracy_cart_model2
# Sensitivity of Cart Model1
sensitivity_cart_model1 = conf_mat_cart_model1[2,2] / sum(conf_mat_cart_model1)
sensitivity_cart_model1
# Sensitivity of Cart Model2
sensitivity_cart_model2 = conf_mat_cart_model2[2,2] / sum(conf_mat_cart_model2)
sensitivity_cart_model2
# Specificity of Cart Model 1
specificity_cart_model1 = conf_mat_cart_model1[1,1] / sum(conf_mat_cart_model1)
specificity_cart_model1
# Specificity of Cart Model 2
specificity_cart_model2 = conf_mat_cart_model2[1,1] / sum(conf_mat_cart_model2)
specificity_cart_model2
# Precision of Cart Model 1
precision_cart_model1 = conf_mat_cart_model1[2,2] / sum(conf_mat_cart_model1)
precision_cart_model1

# Precision of Cart Model 2
precision_cart_model2 = conf_mat_cart_model2[2,2] / sum(conf_mat_cart_model2)
```

```
precision_cart_model2
# Using library ROCR functions prediction and performance
library(ROCR)
# KS of Cart Model 1
pred_cart_model1 = prediction(cart_model1_predict_score[, 2], PremiumData_Test$defaul
t)
perf_cart_model1 = ROCR::performance(pred_cart_model1,"tpr","fpr")
ks_cart_model1 = max(attr(perf_cart_model1,'y.values')[[1]] - attr(perf_cart_model1,
'x.values')[[1]])

pred_cart_model2 = prediction(cart_model2_predict_score[, 2], PremiumData_Test$defaul
t)
perf_cart_model2 = ROCR::performance(pred_cart_model2,"tpr","fpr")
ks_cart_model2 = max(attr(perf_cart_model2,'y.values')[[1]] - attr(perf_cart_model2,
'x.values')[[1]])
# Using library ROCR
auc_cart_model1 = ROCR::performance(pred_cart_model1, measure = "auc")
auc_cart_model1 = auc_cart_model1@y.values[[1]]

auc_cart_model2 = ROCR::performance(pred_cart_model2, measure = "auc")
auc_cart_model2 = auc_cart_model2@y.values[[1]]
library(ineq)

#Gini of Cart Model 1
gini_cart_model1 = ineq(cart_model1_predict_score[, 2],"gini")

#Gini of Cart Model 2
gini_cart_model2 = ineq(cart_model2_predict_score[, 2],"gini")
#Concordance of Cart Model 1
concordance_cart_model1 = Concordance(actuals = ifelse(PremiumData_Test$default == 'D
efaulters', 1,0), predictedScores = ifelse(cart_model1_predict_class == 'Defaulters',
1,0))

#Concordance of Cart Model 2
concordance_cart_model2 = Concordance(actuals = ifelse(PremiumData_Test$default == 'D
efaulters', 1,0), predictedScores = ifelse(cart_model2_predict_class == 'Defaulters',
1,0))
cart_model1_metrics = c(accuracy_cart_model1, sensitivity_cart_model1, specificity_ca
rt_model1, precision_cart_model1, ks_cart_model1, auc_cart_model1, gini_cart_model1,
 concordance_cart_model1$Concordance)

cart_model2_metrics = c(accuracy_cart_model2, sensitivity_cart_model2, specificity_ca
rt_model2, precision_cart_model2, ks_cart_model2, auc_cart_model2, gini_cart_model2,
 concordance_cart_model2$Concordance)

cart_model1_metrics
cart_model2_metrics
PremiumData_Train$default<- as.factor(PremiumData_Train$default)
PremiumData_Test$default<- as.factor(PremiumData_Test$default)
lrmod <- caret::train(default ~ .,
                      method     = "glm",
                      metric     = "Sensitivity",
                      data       = PremiumData_Train)


lrpred<-predict(lrmod,newdata=PremiumData_Test)
```

```
lrpred=as.numeric(lrpred)
PremiumData_Test$default=as.numeric(PremiumData_Test$default)
confusionMatrix(table(lrpred,PremiumData_Test$default))

caret::varImp(lrmod)
#Predict on test data using Logistic Regression Model
lrmod_predict_class = predict(lrmod, PremiumData_Test, type = 'raw')
lrmod_predict_score = predict(lrmod, PremiumData_Test, type = 'prob')
#Confusion Matrix of Logistic Regression Model
conf_mat_lrmod = table(PremiumData_Test$default, lrmod_predict_class)
conf_mat_lrmod

# Accuracy of Logistic Regression Model
accuracy_lrmod = sum(diag(conf_mat_lrmod)) / sum(conf_mat_lrmod)
accuracy_lrmod
# Sensitivity of Logistic Regression Model
sensitivity_lrmod = conf_mat_lrmod[2,2]/ sum(conf_mat_lrmod)
sensitivity_lrmod
# Specificity of Logistic Regression Model
specificity_lrmod = conf_mat_lrmod[1,1] / sum(conf_mat_lrmod)
specificity_lrmod
# Precision of Logistic Regression Model
precision_lrmod = conf_mat_lrmod[2,2] / sum(conf_mat_lrmod)
precision_lrmod
# Using library ROCR functions prediction and performance
library(ROCR)
# KS of Logistic Regression
pred_lrmod = prediction(lrmod_predict_score[, 2], PremiumData_Test$default)
perf_lrmod = ROCR::performance(pred_lrmod,"tpr","fpr")
ks_lrmod = max(attr(perf_lrmod,'y.values')[[1]] - attr(perf_lrmod,'x.values')[[1]])
# Using library ROCR
auc_lrmod = ROCR::performance(pred_lrmod, measure = "auc")
auc_lrmod = NA
library(ineq)
#Gini of Logistic Regression Model
gini_lrmod = ineq(lrmod_predict_score[, 2],"gini")
#Concordance of Logistic Regression Model
concordance_lrmod = Concordance(actuals = ifelse(PremiumData_Test$default == 'Default
ers', 1,0),
                                predictedScores = ifelse(lrmod_predict_class == 'D
efaulters', 1,0))
lrmod_metrics = c(accuracy_lrmod, sensitivity_lrmod, specificity_lrmod, precision_lrm
od,
ks_lrmod, auc_lrmod, gini_lrmod, concordance_lrmod$Concordance)
PremiumData_Train$default<- as.factor(PremiumData_Train$default)
PremiumData_Test$default<- as.factor(PremiumData_Test$default)
model_nb <- caret::train(default ~ ., data = PremiumData_Train,
                         method = "naive_bayes")

summary(model_nb)
nb_predictions_test <- predict(model_nb, newdata = PremiumData_Test, type = "raw")
nb_predictions_test=as.numeric(nb_predictions_test)
PremiumData_Test$default=as.numeric(PremiumData_Test$default)

confusionMatrix(table(nb_predictions_test,PremiumData_Test$default))
```

```
#Predict on test data using Logistic Regression Model
model_nb_predict_class = predict(model_nb, PremiumData_Test, type = 'raw')
model_nb_predict_score = predict(model_nb, PremiumData_Test, type = 'prob')
#Confusion Matrix of Logistic Regression Model
conf_mat_model_nb = table(PremiumData_Test$default, model_nb_predict_class)
conf_mat_model_nb

# Accuracy of Naive Bayes Model
accuracy_model_nb = sum(diag(conf_mat_model_nb)) / sum(conf_mat_model_nb)
accuracy_model_nb
# Sensitivity of Naive Bayes Model
sensitivity_model_nb = conf_mat_model_nb[2,2]/ sum(conf_mat_model_nb)
sensitivity_model_nb
# Specificity of LNaive Bayes Model
specificity_model_nb = conf_mat_model_nb[1,1] / sum(conf_mat_model_nb)
specificity_model_nb
# Precision of Naive Bayes Model
precision_model_nb = conf_mat_model_nb[2,2] / sum(conf_mat_model_nb)
precision_model_nb
# Using library ROCR functions prediction and performance
library(ROCR)
# KS of Naive Bayes
pred_model_nb = prediction(model_nb_predict_score[, 2], PremiumData_Test$default)
perf_model_nb = ROCR::performance(pred_model_nb,"tpr","fpr")
ks_model_nb = max(attr(perf_model_nb,'y.values')[[1]] - attr(perf_model_nb,'x.values'
)[[1]])
# Using library ROCR
auc_model_nb = ROCR::performance(pred_model_nb, measure = "auc")
auc_model_nb = NA
library(ineq)
#Gini of Naive Bayes Model
gini_model_nb = ineq(model_nb_predict_score[, 2],"gini")
#Concordance of Naive Bayes Model
concordance_model_nb = Concordance(actuals = ifelse(PremiumData_Test$default == 'Defa
ulters', 1,0),
                                   predictedScores = ifelse(model_nb_predict_class ==
'Defaulters', 1,0))
model_nb_metrics = c(accuracy_model_nb, sensitivity_model_nb, specificity_model_nb, p
recision_model_nb,
ks_model_nb, auc_model_nb, gini_model_nb, concordance_model_nb$Concordance)
set.seed(123)
knn_model <- caret::train(default ~ ., data = PremiumData_Train,
                          preProcess = c("center", "scale"),
                          method = "knn",
                          tuneLength = 3,
                          trControl = fitControl)
knn_model
knn_predictions_test <- predict(knn_model, newdata = PremiumData_Test, type = "raw")
#confusionMatrix(knn_predictions_test, PremiumData_Test$default)

#Predict on test data using KNN Model
knn_model_predict_class = predict(knn_model, PremiumData_Test, type = 'raw')
knn_model_predict_score = predict(knn_model, PremiumData_Test, type = 'prob')
#Confusion Matrix of KNN Model
conf_mat_knn_model = table(PremiumData_Test$default, knn_model_predict_class)
conf_mat_knn_model
```

```
# Accuracy of KNN Model
accuracy_knn_model = sum(diag(conf_mat_knn_model)) / sum(conf_mat_knn_model)
accuracy_knn_model
# Sensitivity of KNN Model
sensitivity_knn_model = conf_mat_knn_model[2,2]/ sum(conf_mat_knn_model)
sensitivity_knn_model
# Specificity of KNN Model
specificity_knn_model = conf_mat_knn_model[1,1] / sum(conf_mat_knn_model)
specificity_knn_model
# Precision of KNN Model
precision_knn_model = conf_mat_knn_model[2,2] / sum(conf_mat_knn_model[,'Defaulters'
])
precision_knn_model
# Using library ROCR functions prediction and performance
library(ROCR)

# KS of KNN Model
pred_knn_model = prediction(knn_model_predict_score[, 2], PremiumData_Test$default)
perf_knn_model = ROCR::performance(pred_knn_model,"tpr","fpr")
ks_knn_model = max(attr(perf_knn_model,'y.values')[[1]] - attr(perf_knn_model,'x.valu
es')[[1]])
# Using library ROCR
auc_knn_model = ROCR::performance(pred_knn_model, measure = "auc")
auc_knn_model = auc_knn_model@y.values[[1]]
library(ineq)
#Gini of KNN Model
gini_knn_model = ineq(knn_model_predict_score[, 2],"gini")
#Concordance of KNN Model
concordance_knn_model = Concordance(actuals = ifelse(PremiumData_Test$default == 'Def
aulters', 1,0),
                                    predictedScores = ifelse(knn_model_predict_clas
s == 'Defaulters', 1,0))
knn_model_metrics = c(accuracy_knn_model, sensitivity_knn_model, specificity_knn_mode
l, precision_knn_model,
ks_knn_model, auc_knn_model, gini_knn_model, concordance_knn_model$Concordance)

set.seed(1000)
par(mar=c(1,1,1,1))

PremiumData_Train$default=as.factor(PremiumData_Train$default)
PremiumData_Test$default=as.factor(PremiumData_Test$default)
rf_model1 = randomForest(
  default ~ .,
  data = PremiumData_Train,
  ntree = 51,
  mtry = 10,
  nodesize = 10,
  importance = TRUE
)
print(rf_model1)
plot(rf_model1)
print(rf_model1$importance)
set.seed(1000) # To ensure reproducibility
par(mar=c(1,1,1,1))
rf_model2 = tuneRF(x = PremiumData_Train[, -15], # matrix or data frame of predictor/
independent variables
```

```
                    y = PremiumData_Train$default, # response vector (factor for class
ification, numeric for regression)
                    mtrystart = 5, # starting value of mtry
                    stepfactor=1.5, # at each iteration, mtry is inflated (or deflate
d) by this value
                    ntree=25, # number of trees built for each mtry value
                    improve=0.0001, # the (relative) improvement in OOB error must be
 by this much for the search to continue
                    nodesize=10, # Minimum size of terminal nodes
                    trace=TRUE, # prints the progress of the search
                    plot=TRUE,
                    doBest=TRUE, # return a forest using the optimal mtry found
                    importance=TRUE #
)
plot(rf_model2)
# Predicting on the train dataset
train_predict.class_RF <- predict(rf_model2, PremiumData_Train, type="class") # Predi
cted Classes
train_predict.score_RF <- predict(rf_model2, PremiumData_Train, type = 'prob') # Pred
icted Probabilities

# Create confusion matrix for train data predictions
tab.train_RF = table(PremiumData_Train$default, train_predict.class_RF)
tab.train_RF

# Accuracy on train data
accuracy.train_RF = sum(diag(tab.train_RF)) / sum(tab.train_RF)
accuracy.train_RF
# Predicting on the test dataset
test_predict.class_RF <- predict(rf_model2, PremiumData_Test, type="class") # Predict
ed Classes
test_predict.score_RF <- predict(rf_model2, PremiumData_Test, type = 'prob') # Predic
ted Probabilities

# Create confusion matrix for test data predictions
tab.test_RF = table(PremiumData_Test$default, test_predict.class_RF)
tab.test_RF

# Accuracy on test data
accuracy.test_RF = sum(diag(tab.test_RF)) / sum(tab.test_RF)
accuracy.test_RF

# Predict on test data using rf_model1
rf_model1_predict_class = predict(rf_model1, PremiumData_Test, type = 'class')
rf_model1_predict_score = predict(rf_model1, PremiumData_Test, type = 'prob')

# Predict on test data using rf_model2
rf_model2_predict_class = predict(rf_model2, PremiumData_Test, type = 'class')
rf_model2_predict_score = predict(rf_model2, PremiumData_Test, type = 'prob')
#Confusion Matrix of Random Forest Model 1
conf_mat_rf_model1 = table(PremiumData_Test$default, rf_model1_predict_class)
conf_mat_rf_model1

#Confusion Matrix of Random Forest Model 2
conf_mat_rf_model2 = table(PremiumData_Test$default, rf_model2_predict_class)
conf_mat_rf_model2
```

```
# Accuracy of RF Model 1
accuracy_rf_model1 = sum(diag(conf_mat_rf_model1)) / sum(conf_mat_rf_model1)
accuracy_rf_model1
# Accuracy of RF Model 1
accuracy_rf_model2 = sum(diag(conf_mat_rf_model2)) / sum(conf_mat_rf_model2)
accuracy_rf_model2
# Sensitivity of RF Model1
sensitivity_rf_model1 = conf_mat_rf_model1[2,2] / sum(conf_mat_rf_model1)
sensitivity_rf_model1
# Sensitivity of RF Model2
sensitivity_rf_model2 = conf_mat_rf_model2[2,2] / sum(conf_mat_rf_model2)
sensitivity_rf_model2
# Specificity of RF Model 1
specificity_rf_model1 = conf_mat_rf_model1[1,1] / sum(conf_mat_rf_model1)
specificity_rf_model1
# Specificity of RF Model
specificity_rf_model2 = conf_mat_rf_model2[1,1] / sum(conf_mat_rf_model2)
specificity_rf_model2
# Precision of RF Model 1
precision_rf_model1 = conf_mat_rf_model1[2,2] / sum(conf_mat_rf_model1[,'Defaulters'
])

# Precision of RF Model 2
precision_rf_model2 = conf_mat_rf_model2[2,2] / sum(conf_mat_rf_model2[,'Defaulters'
])
# Using library ROCR functions prediction and performance
library(ROCR)
# KS of RF Model 1
pred_rf_model1 = prediction(rf_model1_predict_score[, 2], PremiumData_Test$default)
perf_rf_model1 = ROCR::performance(pred_rf_model1,"tpr","fpr")
ks_rf_model1 = max(attr(perf_rf_model1,'y.values')[[1]] - attr(perf_rf_model1,'x.valu
es')[[1]])

pred_rf_model2 = prediction(rf_model2_predict_score[, 2], PremiumData_Test$default)
perf_rf_model2 = ROCR::performance(pred_rf_model2,"tpr","fpr")
ks_rf_model2 = max(attr(perf_rf_model2,'y.values')[[1]] - attr(perf_rf_model2,'x.valu
es')[[1]])
# Using library ROCR
auc_rf_model1 = ROCR::performance(pred_rf_model1, measure = "auc")
auc_rf_model1 = auc_rf_model1@y.values[[1]]

auc_rf_model2 = ROCR::performance(pred_rf_model2, measure = "auc")
auc_rf_model2 = auc_rf_model2@y.values[[1]]
library(ineq)
#Gini of RF Model 1
gini_rf_model1 = ineq(rf_model1_predict_score[, 2],"gini")

#Gini of RF Model 2
gini_rf_model2 = ineq(rf_model2_predict_score[, 2],"gini")
#Concordance of RF Model 1
concordance_rf_model1 = Concordance(actuals = ifelse(PremiumData_Test$default == 'Def
aulters', 1,0), predictedScores = ifelse(rf_model1_predict_class == 'Defaulters', 1,0
))

#Concordance of RF Model 2
concordance_rf_model2 = Concordance(actuals = ifelse(PremiumData_Test$default == 'Def
```

```
aulters', 1,0), predictedScores = ifelse(rf_model2_predict_class == 'Defaulters', 1,0
))
rf_model1_metrics = c(accuracy_rf_model1, sensitivity_rf_model1, specificity_rf_model
1, precision_rf_model1, ks_rf_model1, auc_rf_model1, gini_rf_model1, concordance_rf_m
odel1$Concordance)

rf_model2_metrics = c(accuracy_rf_model2, sensitivity_rf_model2, specificity_rf_model
2, precision_rf_model2, ks_rf_model2, auc_rf_model2, gini_rf_model2, concordance_rf_m
odel2$Concordance)
pred_rf_model1 = prediction(rf_model1_predict_score[, 2], PremiumData_Test$default)
perf_rf_model1 = ROCR::performance(pred_rf_model1,"tpr","fpr")
plot(perf_rf_model1, main = "ROC Curve" ,colorize = TRUE)
str(perf_rf_model1)

cutoffs <-
  data.frame(
  cut = perf_rf_model2@alpha.values[[1]],
  fpr = perf_rf_model2@x.values[[1]],
  tpr = perf_rf_model2@y.values[[1]]
  )

head(cutoffs)
View(cutoffs)
cutoffs <- cutoffs[order(cutoffs$tpr, decreasing=TRUE),]
head(subset(cutoffs, fpr < 0.4))
class_prediction_with_new_cutoff = ifelse(rf_model1_predict_score[, 2] >= 0.934, 1, 0
)
new_confusion_matrix = table(PremiumData_Test$default, class_prediction_with_new_cuto
ff)
new_confusion_matrix

new_accuracy = sum(diag(new_confusion_matrix)) / sum(new_confusion_matrix)
new_accuracy

new_sensitivity = new_confusion_matrix[2,2] / sum(new_confusion_matrix)
new_sensitivity

new_specificity = new_confusion_matrix[1,1] / sum(new_confusion_matrix)
new_specificity

rfnew_model_metrics = c(new_accuracy, new_sensitivity, new_specificity, NA , NA, NA,
NA, NA)
gbm_model <- caret::train(default ~ ., data = PremiumData_Train,
                          method = "gbm",
                          trControl = fitControl,
                          verbose = FALSE)

summary(gbm_model)
gbm_predictions_test <- predict(gbm_model, newdata = PremiumData_Train, type = "raw")
confusionMatrix(gbm_predictions_test, PremiumData_Train$default)
#Predict on test data using Logistic Regression Model
gbm_model_predict_class = predict(gbm_model, PremiumData_Test, type = 'raw')
gbm_model_predict_score = predict(gbm_model, PremiumData_Test, type = 'prob')
#Confusion Matrix of Gradient Boosting Model
conf_mat_gbm_model = table(PremiumData_Test$default, gbm_model_predict_class)
conf_mat_gbm_model
```

```
# Accuracy of Gradient Boosting Model
accuracy_gbm_model = sum(diag(conf_mat_gbm_model)) / sum(conf_mat_gbm_model)
accuracy_gbm_model
# Sensitivity of Gradient Boosting Model
sensitivity_gbm_model = conf_mat_gbm_model[2,2]/ sum(conf_mat_gbm_model)
sensitivity_gbm_model
# Specificity of Gradient Boosting Model
specificity_gbm_model = conf_mat_gbm_model[1,1] / sum(conf_mat_gbm_model)
specificity_gbm_model
# Precision of Gradient Boosting Model
precision_gbm_model = conf_mat_gbm_model[2,2] / sum(conf_mat_gbm_model)
precision_gbm_model
# Using library ROCR functions prediction and performance
library(ROCR)
# KS of Gradient Boosting
#pred_gbm_model = prediction(gbm_model_predict_score[, 1000], PremiumData_Test$defaul
t)
#perf_gbm_model = ROCR::performance(pred_gbm_model,"tpr","fpr")
#ks_gbm_model = max(attr(perf_gbm_model,'y.values')[[1]] - attr(perf_gbm_model,'x.val
ues')[[1]])
ks_gbm_model = NA
# Using library ROCR
auc_gbm_model = ROCR::performance(pred_model_nb, measure = "auc")
auc_gbm_model = auc_gbm_model@y.values[[1]]
library(ineq)
#Gini of Gradient Boosting Model
#gini_gbm_model = ineq(gbm_model_predict_score[, 2],"gini")
gini_gbm_model=NA
#Concordance of Gradient Boosting Model
concordance_gbm_model = Concordance(actuals = ifelse(PremiumData_Test$default == 'Def
aulters', 1,0),
                                    predictedScores = ifelse(gbm_model_predict_class =
= 'Defaulters', 1,0))
gbm_model_metrics = c(accuracy_gbm_model, sensitivity_gbm_model, specificity_gbm_mode
l, precision_gbm_model,
ks_gbm_model, auc_gbm_model, gini_gbm_model, concordance_gbm_model$Concordance)
gbm_model_metrics
cv.ctrl <- trainControl(method = "repeatedcv", repeats = 1,number = 3,
                        summaryFunction = twoClassSummary,
                        classProbs = TRUE,
                        allowParallel=T)

xgb.grid <- expand.grid(nrounds = 100,
                        eta = c(0.01),
                        max_depth = c(2,4),
                        gamma = 0,               #default=0
                        colsample_bytree = 1,    #default=1
                        min_child_weight = 1,    #default=1
                        subsample = 1            #default=1
)

xgb_model <- caret::train(default~.,
                          data=PremiumData_Train,
                          method="xgbTree",
                          trControl=cv.ctrl,
                          tuneGrid=xgb.grid,
```

```
                          verbose=T,
                          nthread = 2
)

xgb_predictions_test <- predict(xgb_model, newdata = PremiumData_Test, type = "raw")
#confusionMatrix(xgb_predictions_test, PremiumData_Test$default)
#Predict on test data using XG Boost Model
xgb_model_predict_class = predict(xgb_model, PremiumData_Test, type = 'raw')
xgb_model_predict_score = predict(xgb_model, PremiumData_Test, type = 'prob')
#Confusion Matrix of XG Boost Model
conf_mat_xgb_model = table(PremiumData_Test$default, xgb_model_predict_class)
conf_mat_xgb_model

# Accuracy of XG Boost Model
accuracy_xgb_model = sum(diag(conf_mat_xgb_model)) / sum(conf_mat_xgb_model)
accuracy_xgb_model
# Sensitivity of XG Boost Model
sensitivity_xgb_model = conf_mat_xgb_model[2,2]/ sum(conf_mat_xgb_model)
sensitivity_xgb_model
# Specificity of XG Boost Model
specificity_xgb_model = conf_mat_xgb_model[1,1] / sum(conf_mat_xgb_model)
specificity_xgb_model
# Precision of XG Boost Model
precision_xgb_model = conf_mat_xgb_model[2,2] / sum(conf_mat_xgb_model)
# Using library ROCR functions prediction and performance
library(ROCR)
# KS of XG Boost Model
pred_xgb_model = prediction(xgb_model_predict_score[, 2], PremiumData_Test$default)
perf_xgb_model = ROCR::performance(pred_xgb_model,"tpr","fpr")
ks_xgb_model = max(attr(perf_xgb_model,'y.values')[[1]] - attr(perf_xgb_model,'x.valu
es')[[1]])
# Using library ROCR
auc_xgb_model = ROCR::performance(pred_xgb_model, measure = "auc")
auc_xgb_model = auc_xgb_model@y.values[[1]]
library(ineq)
#Gini of XG Boost Model
gini_xgb_model = ineq(xgb_model_predict_score[, 2],"gini")
#Concordance of XG Boost Model
concordance_xgb_model = Concordance(actuals = ifelse(PremiumData_Test$default == 'Def
aulters', 1,0),
                                    predictedScores = ifelse(xgb_model_predict_class =
= 'Defaulters', 1,0))
xgb_model_metrics = c(accuracy_gbm_model, sensitivity_xgb_model, specificity_xgb_mode
l, precision_xgb_model,
ks_xgb_model, auc_xgb_model, gini_xgb_model, concordance_xgb_model$Concordance)
comparison_table = data.frame(cart_model1_metrics,cart_model2_metrics, rf_model1_metr
ics,rf_model2_metrics, lrmod_metrics,model_nb_metrics,knn_model_metrics,gbm_model_met
rics,xgb_model_metrics)

rownames(comparison_table) = c("Accuracy", "Sensitivity", "Specificity", "Precision",
"KS", "Auc", "Gini", "Concordance")

comparison_table
```