Capstone Project

Table of Contents

Introduction

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 us to

1. Build a model that can predict the likelihood of a customer defaulting on premium payments (Who is likely to default)
2. Identify the factors that drive higher default rate (Are there any characteristics of the customers who are likely to default?)
3. 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?)

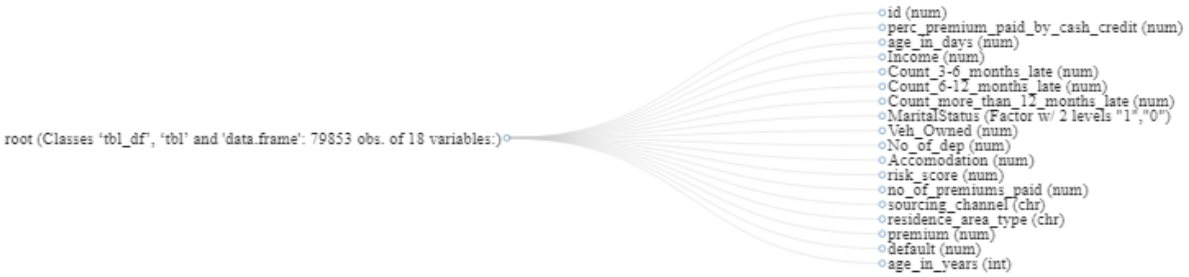
Data Report

Variable Identification

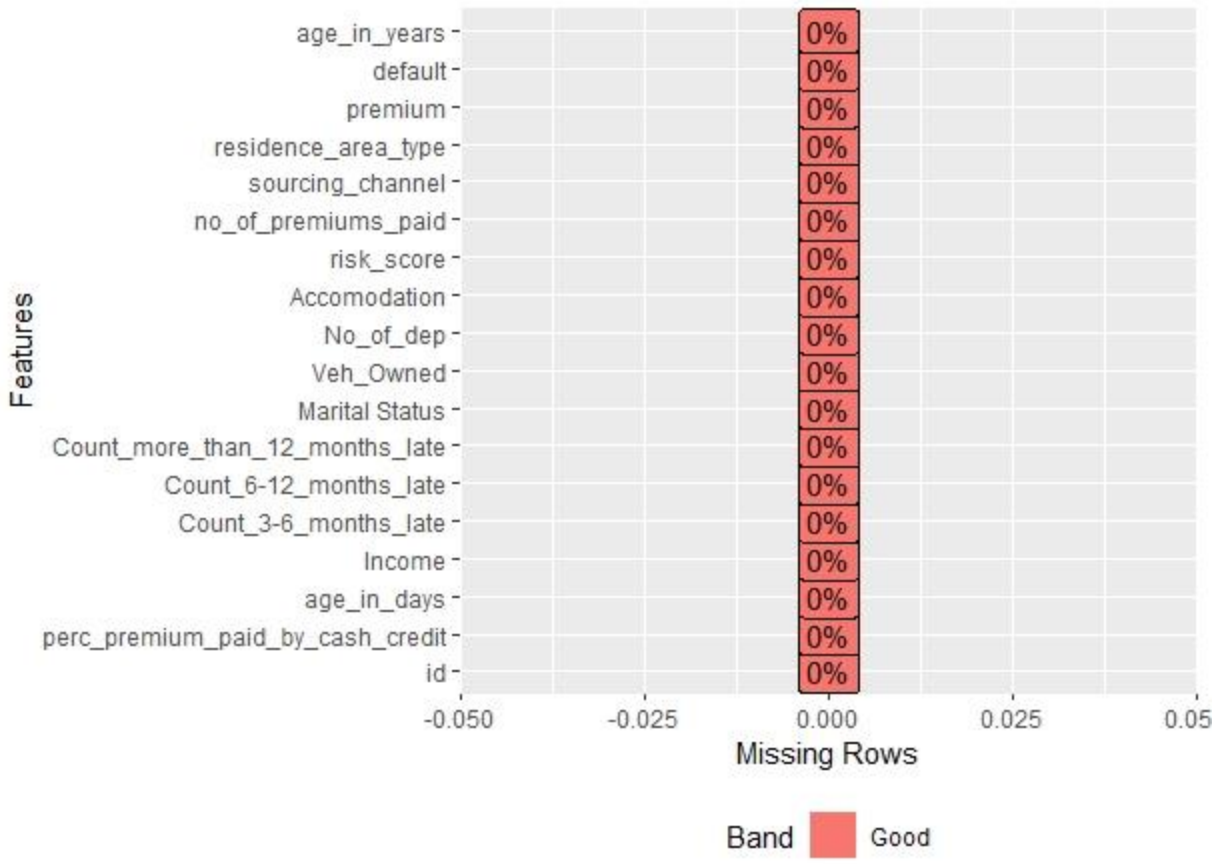
We cane use R functions to do as follows

- dim: we see that we have 79853 different observation in 17 variables.
- names: we see that all the names looking good and straightforward to work with accept some change we will come on soon.
- str: we identifying that:
    - perc_premium_paid_by_cash_credit: num
    - age_in_days : num
    - Income : num
    - Count_3-6_months_late : num
    - Count_6-12_months_late : num
    - Count_more_than_12_months_late : num
    - Marital Status : num
    - Veh_Owned : num
    - No_of_dep : num
    - Accomodation : num
    - risk_score : num
    - no_of_premiums_paid : num
    - sourcing_channel : chr
    - residence_area_type : chr
    - premium : num
    - default : num
- head & tail: shows that we are lucky we have quite bet a clear data.
- anyNA: we see that we don't have missing value at whole dataset.

Please refer Appendix A for Source Code.



Dataset structure



Missing Value

Initial Exploratory Data Analysis

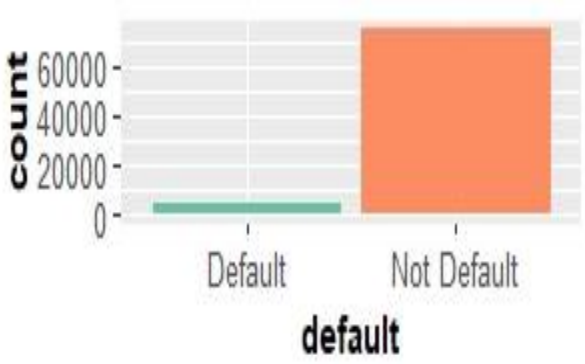
Univariate Analysis

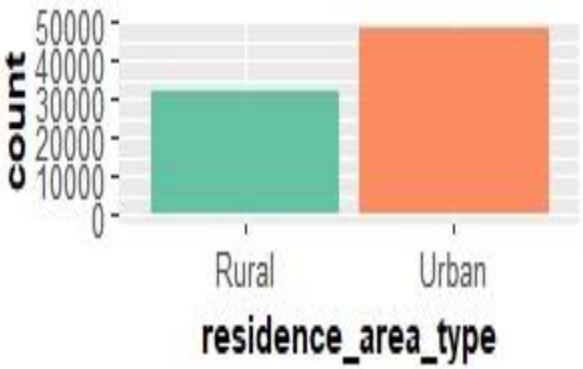
Five Numbers

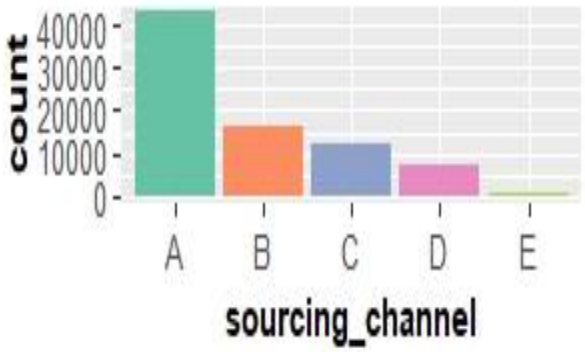
Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
age_in_years	21.0	42.0	52.0	56.4	63.0	104.0
Premium	1200	5400	7500	17580	13800	60000
no_of_premiums_paid	2.0	7.0	10.0	18.6	14.0	60.0
risk_score	91.90	98.83	99.18	97.86	99.52	99.89
No_of_dep	1	2	3	2.6	3	4
Veh_Owned	1	1	2	2	3	3
Income	24030	108010	166560	18162658	252090	90262600

In Dataset we fix the variable age_in_days to be age_in_years to easiest the dealing with years.

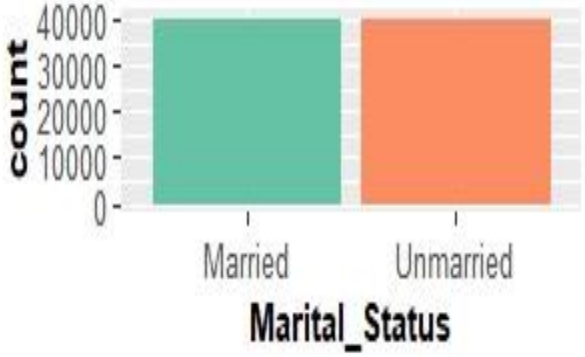
Charts

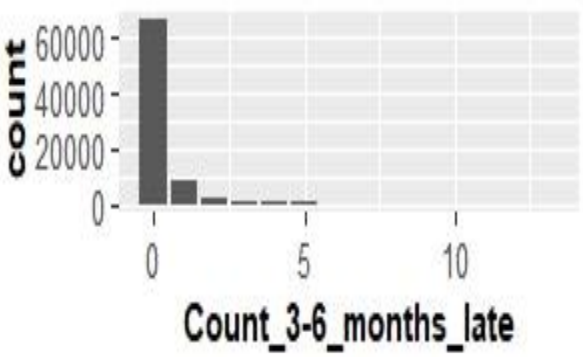

count
60000
40000
20000
0
Default
Not Default
default

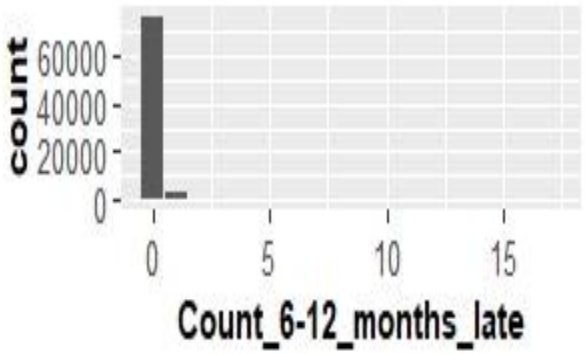

count
50000
40000
30000
20000
10000
0
Rural
Urban
residence_area_type

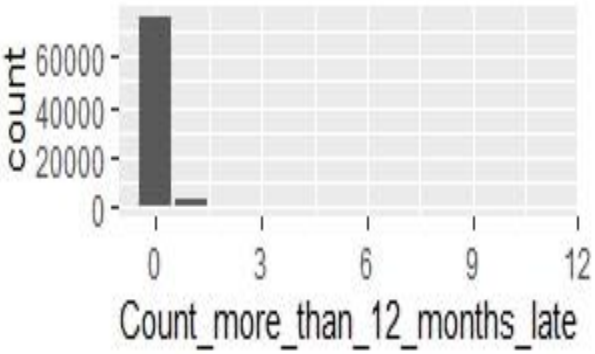

count
40000
30000
20000
10000
0
A
B
C
D
E
sourcing_channel


count
40000
30000
20000
10000
0
Owned
Rented
Accomodation


count
40000
30000
20000
10000
0
Married
Unmarried
Marital_Status

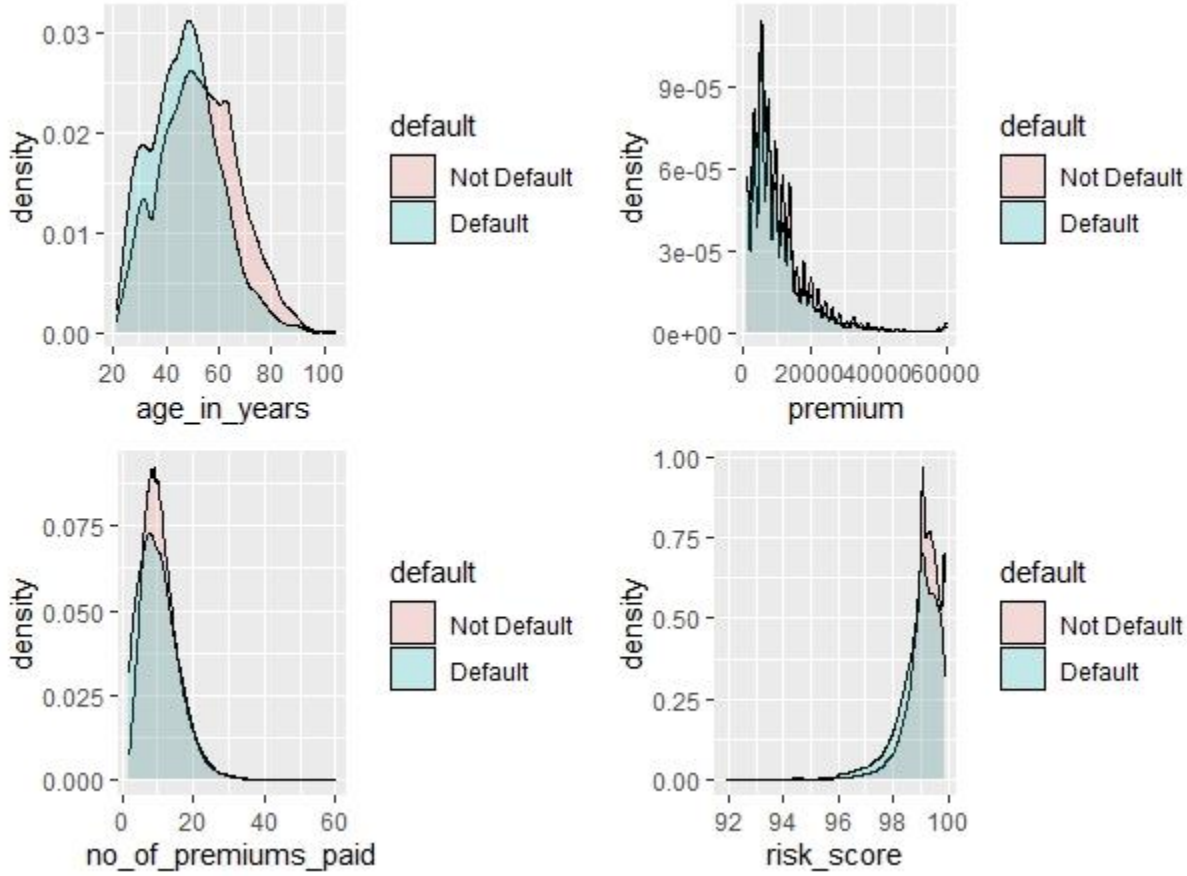

count
60000
40000
20000
0
0
5
10
Count_3-6_months_late


count
60000
40000
20000
0
0
5
10
15
Count_6-12_months_late

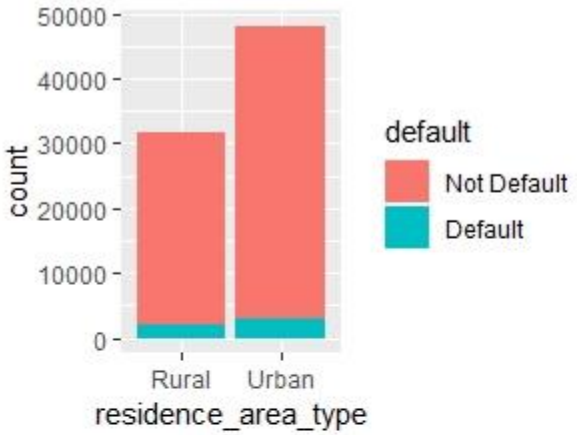

count
60000
40000
20000
0
0
3
6
9
12
Count_more_than_12_months_late

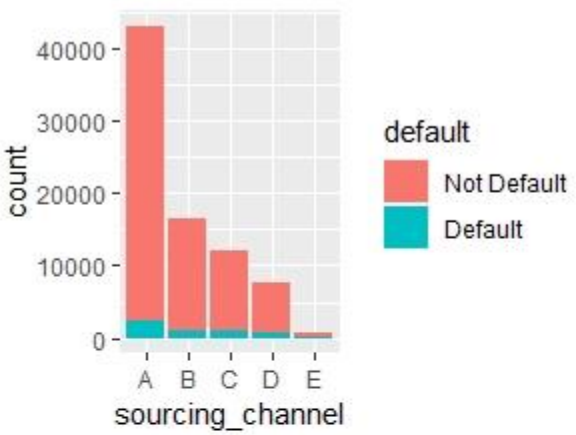
Bivariate Analysis

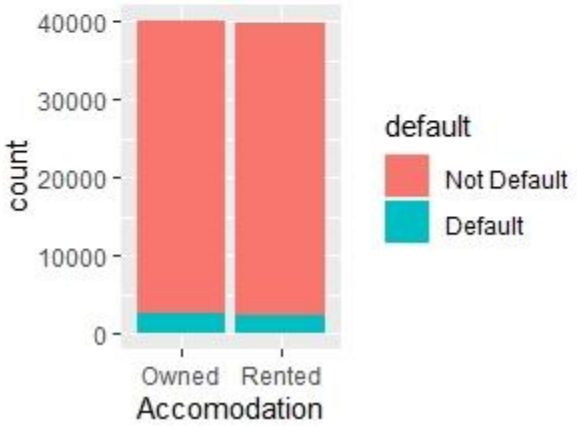
Default vs Numerical Variables

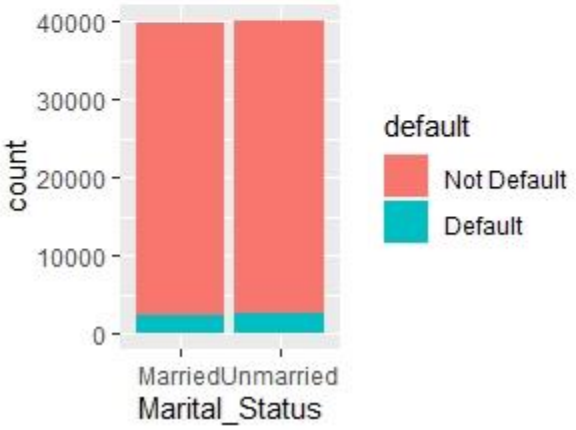


Default vs Categorical Variables

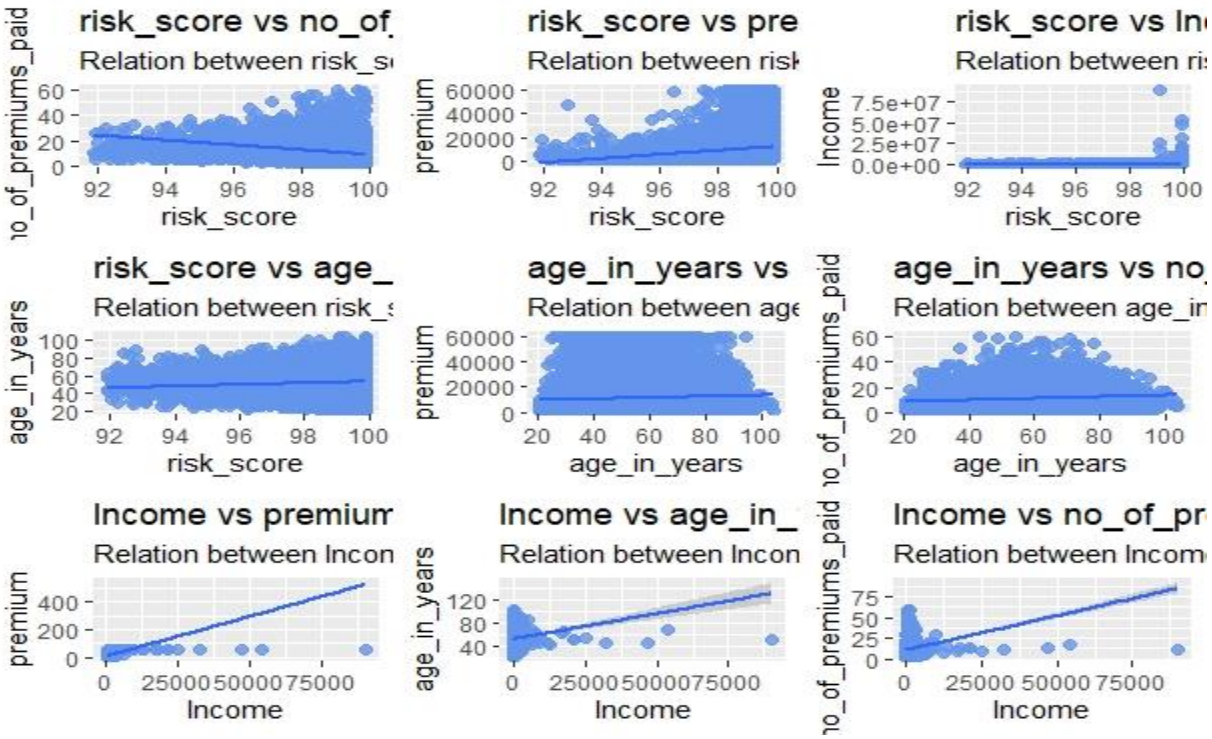


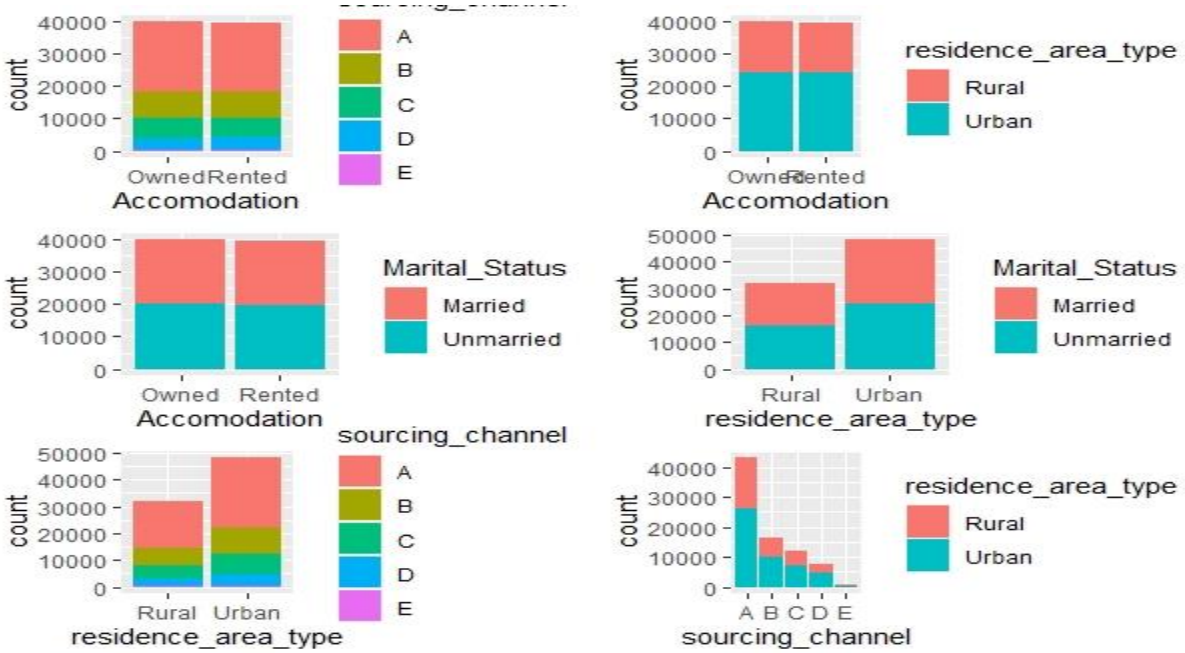






Correlation Analysis





Data pre-processing

Removal of Unwanted Variables

Variable to be removed are:

- Id: we don't need it.
- Age_in_days: while we are generateing new variable from this variable but in years format, so we don't need it any more.

Missing Value Treatment

There are no messing data in the dataset.

Outlier Treatment

The Method of outliers detection we are going to use is based on the **percentiles**. With the percentiles method, all observations that lie outside the interval formed by the 1 and 99. percentiles will be considered as potential outlier, after that simply we are going to remove the observation with outlier.

Please refer Appendix A for Source Code.

Variable Transformation

We realize that we need to maintain (change type, rename) some variable as follows:

- #Marital_Status
- #Count_3_6_months_late
- # Count_6_12_months_late
- #Veh_Owned
- #No_of_dep
- #Accomodation
- #sourcing_channel
- #residence_area_type
- #default

Addition of new variables

We add the following:

- Age_in_years

Please refer Appendix A for Source Code.

Exploratory Data Analysis

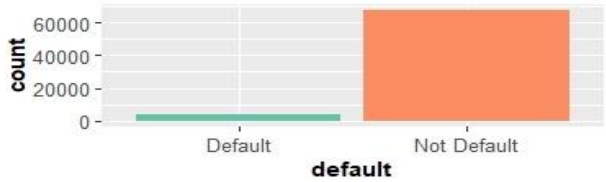
Relationship among variables, important variables

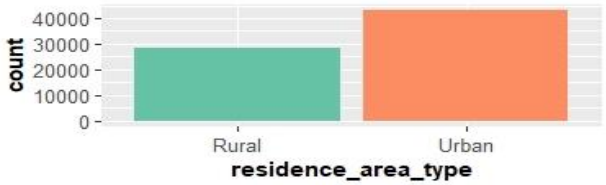
Five Numbers

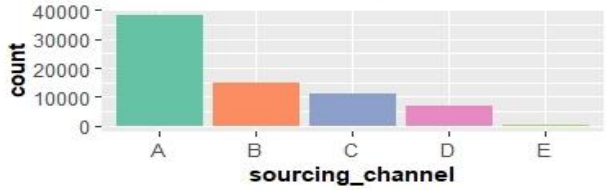
Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
age_in_years	25.0	42.0	52.0	53.6	62.0	87
Premium	1200	5400	7500	12960	13800	36900
no_of_premiums_paid	3.0	7.0	10.0	12.0	13.0	27.0
risk_score	91.96	98.83	99.18	97.87	99.51	99.89
No_of_dep	1	2	3	2.6	3	4
Veh_Owned	1	1	2	2	3	3

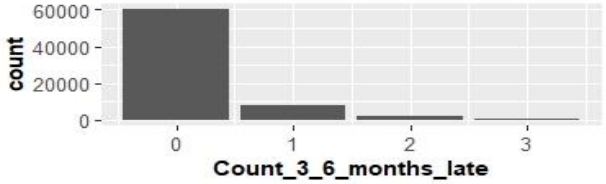
Insightful Visualizations

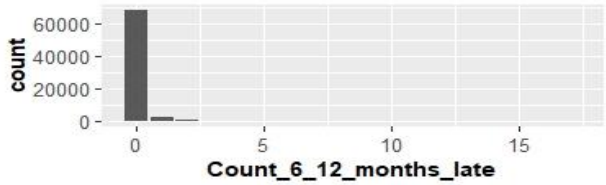
UNIVARIATE ANALYSIS

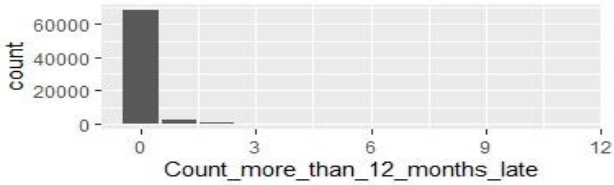




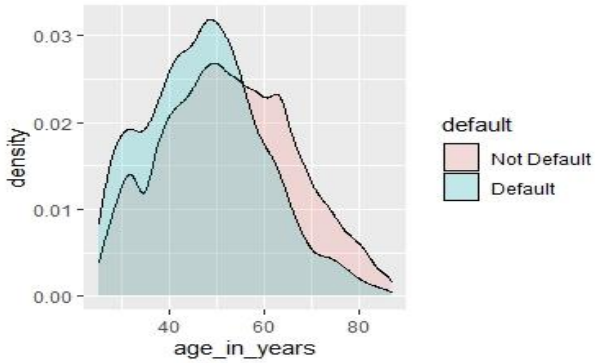


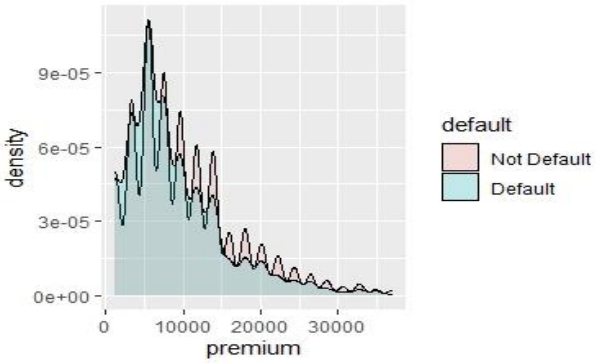


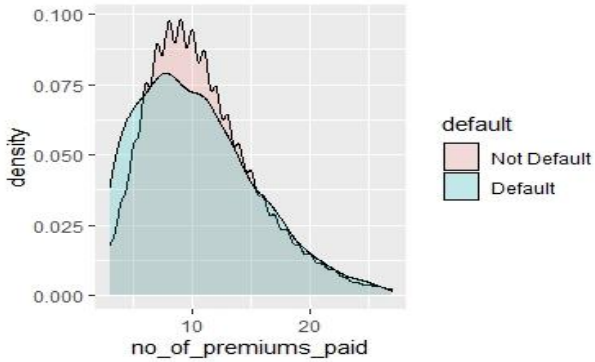


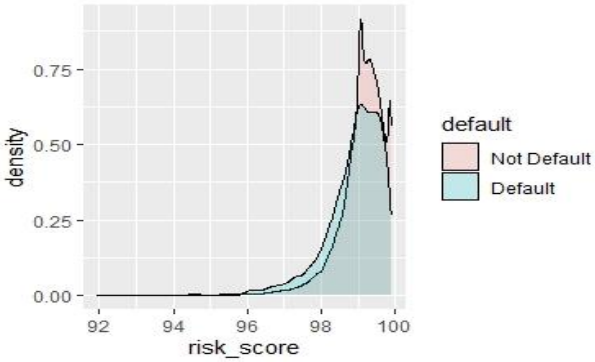


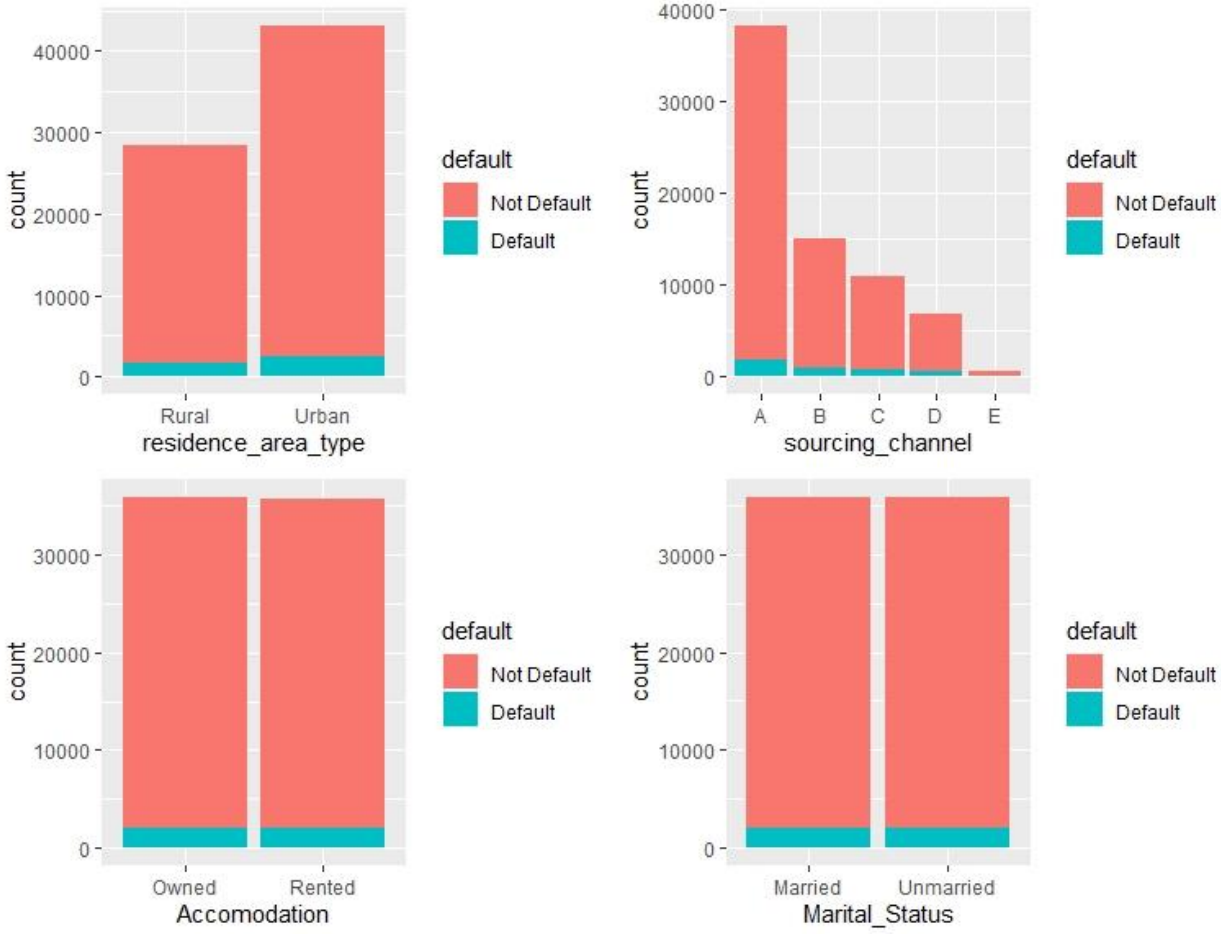
BIVARIATE ANALYSIS



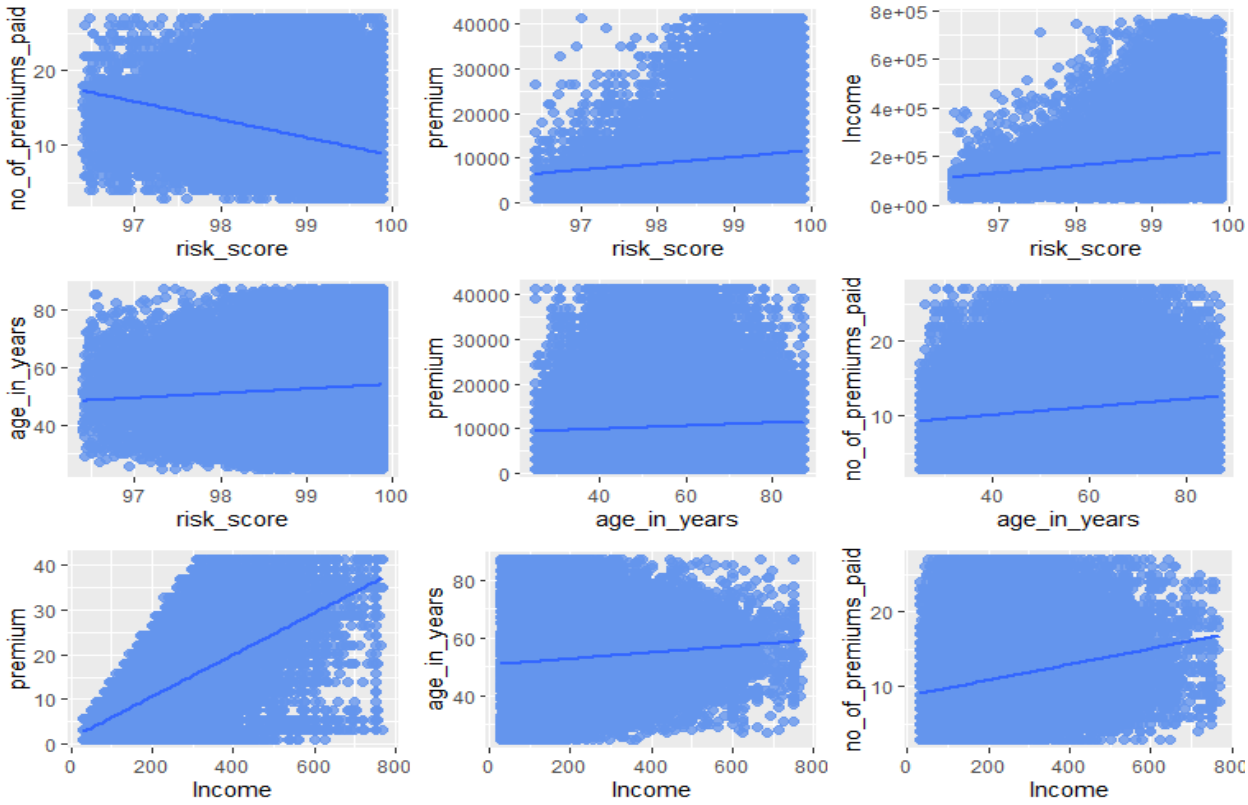


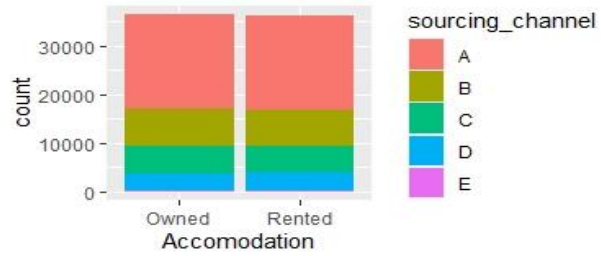


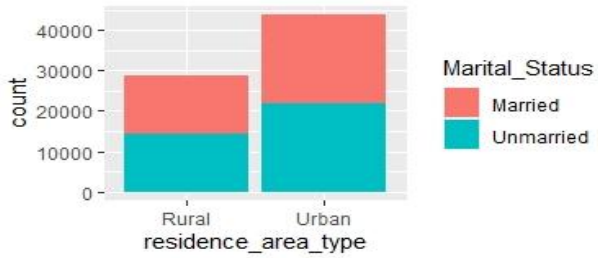


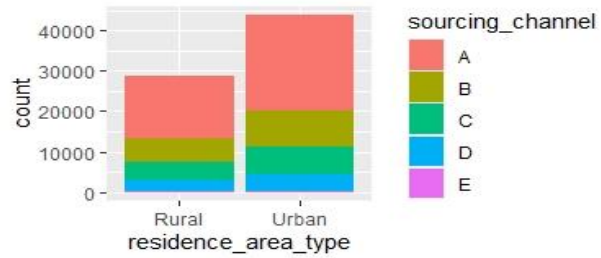

count
40000
30000
20000
10000
0
Rural
Urban
residence_area_type
default
Not Default
Default
A
B
C
D
E
sourcing_channel
Owned
Rented
Accomodation
Married
Unmarried
Marital_Status

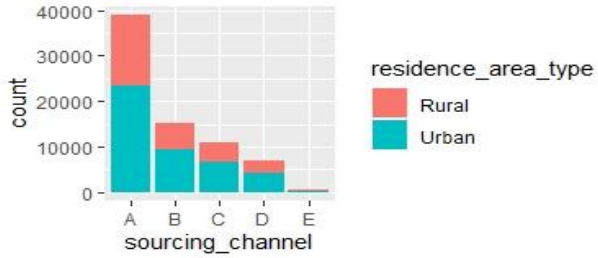
Correlation Analysis











Analytical approach

We are going to use Random forest and logistic regression techniques to build the models and then we will compare the results to choice the best model.

We will divide dataset into two parts as known as train data and test data as (80 %, 20%) respectively.

We will use confusion matrix, KS, Gini and AUC techniques to compare the models and chose the best one for our case.

Appendix A – Source Code

```
library(readxl)
library(plyr)
library(ggplot2)



Insurance_Premium_Default_Dataset <- read_excel("Data/Capstone
Project/Insurance Premium Default-Dataset.xlsx")
View(Insurance_Premium_Default_Dataset)




#Retrieve the dimension of an object.
dim(Insurance_Premium_Default_Dataset)

#Get the names of an object.
names(Insurance_Premium_Default_Dataset)

#Display the internal structure of an dataset.
str(Insurance_Premium_Default_Dataset)

#Returns the first 10 rows of the dataset.
head(Insurance_Premium_Default_Dataset, 10)

#Returns the last 10 rows of the dataset.
tail(Insurance_Premium_Default_Dataset, 10)

#Return a summary of the dataset variables.
summary(Insurance_Premium_Default_Dataset)

#check if ther is any NA value in dataset
anyNA(Insurance_Premium_Default_Dataset)


#preparing variables

#generate age_in_years
Insurance_Premium_Default_Dataset$age_in_years =
as.integer(format(round(Insurance_Premium_Default_Dataset$age_in_days/360,
0), nsmall = 0))



#convert from quantitative to qualitative

#Marital_Status
Insurance_Premium_Default_Dataset$`Marital Status` <-
factor(Insurance_Premium_Default_Dataset$`Marital Status`, order = F, levels
=c("1","0"))
Insurance_Premium_Default_Dataset$Marital_Status <-
factor(mapvalues(Insurance_Premium_Default_Dataset$`Marital Status`, from =
c("1", "0"), to = c("Married","Unmarried")))
```

```
#Accomodation
Insurance_Premium_Default_Dataset$Accomodation <-
factor(Insurance_Premium_Default_Dataset$Accomodation, order = F, levels
=c("1","0"))
Insurance_Premium_Default_Dataset$Accomodation <-
factor(mapvalues(Insurance_Premium_Default_Dataset$Accomodation, from =
c("1", "0"), to = c("Owned","Rented")))

#sourcing_channel
Insurance_Premium_Default_Dataset$sourcing_channel =
as.factor(Insurance_Premium_Default_Dataset$sourcing_channel)

#residence_area_type
Insurance_Premium_Default_Dataset$residence_area_type =
as.factor(Insurance_Premium_Default_Dataset$residence_area_type)

#default

Insurance_Premium_Default_Dataset$default =
as.factor(Insurance_Premium_Default_Dataset$default)
Insurance_Premium_Default_Dataset$default <-
factor(mapvalues(Insurance_Premium_Default_Dataset$default, from = c("1",
"0"), to = c("Not Default", "Default")))


#objects in the dataset can be accessed by simply giving their names
attach(Insurance_Premium_Default_Dataset)


summary(Insurance_Premium_Default_Dataset)



# Load DataExplorer for exploratory data analysis.
library(DataExplorer)

# This function helps to visualize data structure in network graph format.
plot_str(Insurance_Premium_Default_Dataset, type="d", fontSize = 25)
# plot missing data
plot_missing(Insurance_Premium_Default_Dataset)


# Check the fivenumber summary of variables
summary(fivenum(Insurance_Premium_Default_Dataset$age_in_years))
summary(fivenum(Insurance_Premium_Default_Dataset$premium))
summary(fivenum(Insurance_Premium_Default_Dataset$no_of_premiums_paid))
summary(fivenum(Insurance_Premium_Default_Dataset$risk_score))
summary(fivenum(Insurance_Premium_Default_Dataset$No_of_dep))
summary(fivenum(Insurance_Premium_Default_Dataset$Veh_Owned))
summary(fivenum(Insurance_Premium_Default_Dataset$Income))
```

```
### UNIVARIATE ANALYSIS

library(ggplot2)
library(grid)
library(gridExtra)
## visualize properties of all categorical variables


# Setting up the aesthetics
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"

# Plotting the bar charts
g1=ggplot(Insurance_Premium_Default_Dataset, aes(x=default, fill=default)) +
geom_bar()+ unipar + scale_fill_brewer(palette=col1)

# Plotting the bar charts
g2=ggplot(Insurance_Premium_Default_Dataset, aes(x=residence_area_type,
fill=residence_area_type)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)

# Plotting the bar charts
g3=ggplot(Insurance_Premium_Default_Dataset, aes(x=sourcing_channel,
fill=sourcing_channel)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)

# Plotting the bar charts
g4=ggplot(Insurance_Premium_Default_Dataset, aes(x=Accomodation,
fill=Accomodation)) + geom_bar()+ unipar + scale_fill_brewer(palette=col1)

# Plotting the bar charts
g5=ggplot(Insurance_Premium_Default_Dataset, aes(x=Marital_Status,
fill=`Marital_Status`)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)

# Plotting the bar charts
g6=ggplot(Insurance_Premium_Default_Dataset, aes(x=`Count_3-6_months_late`,
fill=`Count_3-6_months_late`)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)

# Plotting the bar charts
g7=ggplot(Insurance_Premium_Default_Dataset, aes(x=`Count_6-12_months_late`,
fill=`Count_6-12_months_late`)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)




# Plotting the bar charts
g8=ggplot(Insurance_Premium_Default_Dataset) + geom_bar(aes(x =
Count_more_than_12_months_late)) + scale_fill_brewer(palette=col2)
```

```
# Partitioning the barcharts
grid.arrange(g1,g2,g3,g4,g5,g6,g7,g8,ncol=2)


### BIVARIATE ANALYSIS


# Setting up the aesthetics
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"

# default vs numerical variables
p1=ggplot(Insurance_Premium_Default_Dataset,
          aes(x = age_in_years, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "age_in_years")


p2=ggplot(Insurance_Premium_Default_Dataset,
          aes(x = premium, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "premium")


p3=ggplot(Insurance_Premium_Default_Dataset,
          aes(x = no_of_premiums_paid, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "no_of_premiums_paid")


p4=ggplot(Insurance_Premium_Default_Dataset,
          aes(x = risk_score, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
```

```
                              labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "risk_score")

# Partitioning the boxplots
grid.arrange(p1,p2,p3,p4,ncol=2)




# Setting up the aesthetics
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)

# default vs categorical variables
# stacked bar chart
p8 = ggplot(Insurance_Premium_Default_Dataset,
         aes(x = residence_area_type,
             fill = factor(default,
                           levels = c("Not Default", "Default"),
                           labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "residence_area_type",
       title = "Custome default by residence_area_type") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked


p9 = ggplot(Insurance_Premium_Default_Dataset,
            aes(x = sourcing_channel,
                fill = factor(default,
                              levels = c("Not Default", "Default"),
                              labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "sourcing_channel",
       title = "Custome default by sourcing_channel") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked


p10 = ggplot(Insurance_Premium_Default_Dataset,
            aes(x = Accomodation,
                fill = factor(default,
                              levels = c("Not Default", "Default"),
                              labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "Accomodation",
       title = "Custome default by Accomodation") +
```

```
  geom_bar(position = "stack") #specifying the type of bar chart as stacked

p11 = ggplot(Insurance_Premium_Default_Dataset,
             aes(x = Marital_Status,
                 fill = factor(default,
                               levels = c("Not Default", "Default"),
                               labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "Marital_Status",
       title = "Custome default by Marital_Status") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked

# Partitioning the boxplots
grid.arrange(p8,p9,p10,p11,ncol=2)



# removing unwante
IDataset = Insurance_Premium_Default_Dataset[,c(2, 4 ,5, 6, 7, 9, 10, 11, 12,
13, 14, 15, 16, 18, 19, 17 ) ]



IDataset = IDataset %>%
  rename(
    Count_3_6_months_late = `Count_3-6_months_late`,
    Count_6_12_months_late = `Count_6-12_months_late`
  )


attach(IDataset)

#outlier treatment


#income
lower_bound <- quantile(IDataset$Income, 0.01)
upper_bound <- quantile(IDataset$Income, 0.99)

outlier_ind <- which(IDataset$Income < lower_bound | IDataset$Income >
upper_bound)

if( length(outlier_ind) > 0)
IDataset = IDataset[-outlier_ind, ]




#perc_premium_paid_by_cash_credit
lower_bound <- quantile(IDataset$perc_premium_paid_by_cash_credit, 0.01)
upper_bound <- quantile(IDataset$perc_premium_paid_by_cash_credit, 0.99)

outlier_ind <- which(IDataset$perc_premium_paid_by_cash_credit < lower_bound
| IDataset$perc_premium_paid_by_cash_credit > upper_bound)

if( length(outlier_ind) > 0)
```

```
  IDataset = IDataset[-outlier_ind, ]



#Count_3_6_months_late
lower_bound <- quantile(IDataset$Count_3_6_months_late, 0.01)
upper_bound <- quantile(IDataset$Count_3_6_months_late, 0.99)

outlier_ind <- which(IDataset$Count_3_6_months_late < lower_bound |
IDataset$Count_3_6_months_late > upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]



#Count_6_12_months_late
lower_bound <- quantile(IDataset$Count_6_12_months_late, 0.01)
upper_bound <- quantile(IDataset$Count_6_12_months_late, 0.99)

outlier_ind <- which(IDataset$Count_6_12_months_late < lower_bound |
IDataset$Count_6_12_months_late > upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]




#Count_more_than_12_months_late
lower_bound <- quantile(IDataset$Count_more_than_12_months_late, 0.01)
upper_bound <- quantile(IDataset$Count_more_than_12_months_late, 0.99)

outlier_ind <- which(IDataset$Count_more_than_12_months_late < lower_bound |
IDataset$Count_more_than_12_months_late > upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]



#Veh_Owned
lower_bound <- quantile(IDataset$Veh_Owned, 0.01)
upper_bound <- quantile(IDataset$Veh_Owned, 0.99)

outlier_ind <- which(IDataset$Veh_Owned < lower_bound | IDataset$Veh_Owned >
upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]
```

```r
#No_of_dep
lower_bound <- quantile(IDataset$No_of_dep, 0.01)
upper_bound <- quantile(IDataset$No_of_dep, 0.99)

outlier_ind <- which(IDataset$No_of_dep < lower_bound | IDataset$No_of_dep >
upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]




#risk_score
lower_bound <- quantile(IDataset$risk_score, 0.01)
upper_bound <- quantile(IDataset$risk_score, 0.99)

outlier_ind <- which(IDataset$risk_score < lower_bound | IDataset$risk_score
> upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]



#no_of_premiums_paid
lower_bound <- quantile(IDataset$no_of_premiums_paid, 0.01)
upper_bound <- quantile(IDataset$no_of_premiums_paid, 0.99)

outlier_ind <- which(IDataset$no_of_premiums_paid < lower_bound |
IDataset$no_of_premiums_paid > upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]




#premium
lower_bound <- quantile(IDataset$premium, 0.01)
upper_bound <- quantile(IDataset$premium, 0.99)

outlier_ind <- which(IDataset$premium < lower_bound | IDataset$premium >
upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]



#age_in_years
lower_bound <- quantile(IDataset$age_in_years, 0.01)
```

```r
upper_bound <- quantile(IDataset$age_in_years, 0.99)

outlier_ind <- which(IDataset$age_in_years < lower_bound |
IDataset$age_in_years > upper_bound)

if( length(outlier_ind) > 0)
  IDataset = IDataset[-outlier_ind, ]

#EDA again

# Check the fivenumber summary of variables
summary(fivenum(IDataset$age_in_years))
summary(fivenum(IDataset$premium))
summary(fivenum(IDataset$no_of_premiums_paid))
summary(fivenum(IDataset$risk_score))
summary(fivenum(IDataset$No_of_dep))
summary(fivenum(IDataset$Veh_Owned))
summary(fivenum(IDataset$Income))




### UNIVARIATE ANALYSIS

library(ggplot2)
library(grid)
library(gridExtra)
## visualize properties of all categorical variables


# Setting up the aesthetics
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"

# Plotting the bar charts
g1=ggplot(IDataset, aes(x=default, fill=default)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)

# Plotting the bar charts
g2=ggplot(IDataset, aes(x=residence_area_type, fill=residence_area_type)) +
geom_bar()+ unipar + scale_fill_brewer(palette=col1)

# Plotting the bar charts
g3=ggplot(IDataset, aes(x=sourcing_channel, fill=sourcing_channel)) +
geom_bar()+ unipar + scale_fill_brewer(palette=col1)

# Plotting the bar charts
g4=ggplot(IDataset, aes(x=Accomodation, fill=Accomodation)) + geom_bar()+
unipar + scale_fill_brewer(palette=col1)
```

```
# Plotting the bar charts
g5=ggplot(IDataset, aes(x=Marital_Status, fill=Marital_Status)) + geom_bar()+
unipar + scale_fill_brewer(palette=col1)

# Plotting the bar charts
g6=ggplot(IDataset, aes(x=Count_3_6_months_late, fill=Count_3_6_months_late))
+ geom_bar()+ unipar + scale_fill_brewer(palette=col1)

# Plotting the bar charts
g7=ggplot(IDataset, aes(x=Count_6_12_months_late,
fill=Count_6_12_months_late)) + geom_bar()+ unipar +
scale_fill_brewer(palette=col1)




# Plotting the bar charts
g8=ggplot(IDataset) + geom_bar(aes(x = Count_more_than_12_months_late)) +
scale_fill_brewer(palette=col2)


# Partitioning the barcharts
grid.arrange(g1,g2,g3,g4,g5,g6,g7,g8,ncol=2)



### BIVARIATE ANALYSIS


# Setting up the aesthetics
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"

# default vs numerical variables
p1=ggplot(IDataset,
          aes(x = age_in_years, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "age_in_years")


p2=ggplot(IDataset,
          aes(x = premium, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
```

```
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "premium")


p3=ggplot(IDataset,
          aes(x = no_of_premiums_paid, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "no_of_premiums_paid")


p4=ggplot(IDataset,
          aes(x = risk_score, #quantitative variable
              fill = factor(default,
                            levels = c("Not Default", "Default"),
                            labels = c("Not Default", "Default")))) +
  geom_density(alpha = 0.2) + #setting transparency of graph to keep overlaps
visible
  labs(fill = "default", # setting title of legend
       x = "risk_score")

# Partitioning the boxplots
grid.arrange(p1,p2,p3,p4,ncol=2)




# Setting up the aesthetics
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)

# default vs categorical variables
# stacked bar chart
p8 = ggplot(IDataset,
            aes(x = residence_area_type,
                fill = factor(default,
                              levels = c("Not Default", "Default"),
                              labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "residence_area_type") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked
```

```
p9 = ggplot(IDataset,
            aes(x = sourcing_channel,
                fill = factor(default,
                              levels = c("Not Default", "Default"),
                              labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "sourcing_channel") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked


p10 = ggplot(IDataset,
             aes(x = Accomodation,
                 fill = factor(default,
                               levels = c("Not Default", "Default"),
                               labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "Accomodation") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked

p11 = ggplot(IDataset,
             aes(x = Marital_Status,
                 fill = factor(default,
                               levels = c("Not Default", "Default"),
                               labels = c("Not Default", "Default")))) +
  labs(fill = "default", # setting title of legend
       x = "Marital_Status") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked

# Partitioning the boxplots
grid.arrange(p8,p9,p10,p11,ncol=2)



# correlation analysis
#scatter plot
c1 = ggplot(IDataset,aes(x = risk_score,y = no_of_premiums_paid)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "risk_score", #specifying the labels of axes and title of plot
       y = "no_of_premiums_paid") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables



#scatter plot
c2 = ggplot(IDataset,aes(x = risk_score,y = premium)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "risk_score", #specifying the labels of axes and title of plot
       y = "premium") +
```

```
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables


#scatter plot
c3 = ggplot(IDataset,aes(x = risk_score,y = Income)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "risk_score", #specifying the labels of axes and title of plot
       y = "Income") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables



#scatter plot
c4 = ggplot(IDataset,aes(x = risk_score,y = age_in_years)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "risk_score", #specifying the labels of axes and title of plot
       y = "age_in_years") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables



#scatter plot
c5 = ggplot(IDataset,aes(x = age_in_years,y = premium)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "age_in_years", #specifying the labels of axes and title of plot
       y = "premium") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables


#scatter plot
c6 = ggplot(IDataset,aes(x = age_in_years,y = no_of_premiums_paid)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "age_in_years", #specifying the labels of axes and title of plot
       y = "no_of_premiums_paid") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables
```

```
#scatter plot
c7 = ggplot(IDataset,aes(x = Income/1000,y = premium/1000)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "Income", #specifying the labels of axes and title of plot
       y = "premium") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables




#scatter plot
c8 = ggplot(IDataset,aes(x = Income/1000,y = age_in_years)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "Income", #specifying the labels of axes and title of plot
       y = "age_in_years") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables



#scatter plot
c9 = ggplot(IDataset,aes(x = Income/1000,y = no_of_premiums_paid)) +
  geom_point(color="cornflowerblue", #setting the colour, size and
transparency(alpha) of the points
             size = 2,
             alpha=.8) +
  labs(x = "Income", #specifying the labels of axes and title of plot
       y = "no_of_premiums_paid") +
  geom_smooth(method = "lm") # this adds a linear trend line which is useful
to summarize the relationship between the two variables



grid.arrange(c1,c2,c3,c4,c5,c6,c7,c8,c9,ncol=3)



# stacked bar chart
cc1 = ggplot(IDataset,
             aes(x = Accomodation,
                 fill = factor(sourcing_channel))) +
  labs(fill = "sourcing_channel", # setting title of legend
       x = "Accomodation") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked



cc2 = ggplot(IDataset,
```

```
            aes(x = Accomodation,
                fill = factor(residence_area_type))) +
  labs(fill = "residence_area_type", # setting title of legend
       x = "Accomodation") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked



cc3 = ggplot(IDataset,
            aes(x = Accomodation,
                fill = factor(Marital_Status))) +
  labs(fill = "Marital_Status", # setting title of legend
       x = "Accomodation") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked



cc4 = ggplot(IDataset,
            aes(x = residence_area_type,
                fill = factor(Marital_Status))) +
  labs(fill = "Marital_Status", # setting title of legend
       x = "residence_area_type") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked



cc5 = ggplot(IDataset,
            aes(x = residence_area_type,
                fill = factor(sourcing_channel))) +
  labs(fill = "sourcing_channel", # setting title of legend
       x = "residence_area_type") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked



cc6 = ggplot(IDataset,
            aes(x = sourcing_channel,
                fill = factor(residence_area_type))) +
  labs(fill = "residence_area_type", # setting title of legend
       x = "sourcing_channel") +
  geom_bar(position = "stack") #specifying the type of bar chart as stacked

grid.arrange(cc1,cc2,cc3,cc4,cc5,cc6,ncol=2)
```