# Credit Risk Modelling - German Bank

*Subramanian Kannappan*

*9th Nov 2019*

## 1) Executive summary

The project is to develop Credit Risk Modeling for the German Bank on the loans that were provided to their customers. This is very crucial for the Bank as this carries huge risk for them if the loans are not repaid by the borrowers.

Based on the list of details captured in the dataset we will develop the model to classify the credit provided to the borrowers as bad or good. The expectation of the German Bank from the credit risk model to be developed is to suggest them an optimal way to
(1) to minimise their loss and there by their risk, by identifying maximum number of bad loans
(2) maximise the business & there by the profit by identifying the good customers so that potential business is not missed.

This means that we can not go for choosing a model just becuase it has high Accuracy value as there is a possibility that it may have very low sensitivity but high specificity .

Hence when developing the models we will look for those models which strike good balance between
Sensitivity (i.e identifiying the bad loans) and
Specificity (i.e identifiying good loans)

## 2) Initial project setup

### 2.1) Installing the required packages

```r
# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(tree)) install.packages("tree", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
if(!require(gmodels)) install.packages("gmodels", repos = "http://cran.us.r-project.org")
if(!require(brnn)) install.packages("brnn", repos = "http://cran.us.r-project.org")
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
if(!require(MASS)) install.packages("MASS", repos = "http://cran.us.r-project.org")
if(!require(kernlab)) install.packages("kernlab", repos = "http://cran.us.r-project.org")
if(!require(rocc)) install.packages("rocc", repos = "http://cran.us.r-project.org")
if(!require(rminer)) install.packages("rminer", repos = "http://cran.us.r-project.org")
```

### 2.2) Loading the required libraries

```r
library(tidyverse)
library(caret)
library(readr)
library(data.table)
library(randomForest)
library(rpart)
library(rpart.plot)
library(gmodels)
library(brnn)
library(xgboost)
library(MASS)
library(dplyr)
library(corrplot)
library(kernlab)
library(rocc)
library(ggplot2)
library(rminer)
library(e1071)
library(reshape2)
library(gbm)
source("http://pcwww.liv.ac.uk/~william/R/crosstab.r")
```

### 2.3) Data Loading

Data is downloaded from the the site https://www.kaggle.com/kabure/german-credit-data-with-risk.

NOTE : As the site needs login credential the file is not directly read from the site from with in the code. The readers are expected to download the file and copy it to their local folder before proceeding further to execute this code. In my case I have stored it in folder called subbu under C drive. Please replace this with which ever folder you have placed the data set.For easy reference I have provided the file itself.

```r
# Loading the file
credit_data<- read_csv("C:\\Subbu\\german_credit_data.csv")
```

## 3) Exploratory Data Analysis (EDA)

In this section we will peform following actions
1) Explore the basic structure of the data set, 2) Analyse the data distribution - to check whether any specific feature has undue prevalence that will influence the outcome of the model 3) Check for correlation - so that similar features can be excluded when building the model 4) Data cleansing - Remove outliers and unwanted columns

### 3.1) Exploring the basic structure

```r
#check the basic structure & content
str(credit_data)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 1000 obs. of  11 variables:
##  $ X1               : num  0 1 2 3 4 5 6 7 8 9 ...
##  $ Age             : num  67 22 49 45 53 35 53 35 61 28 ...
##  $ Sex             : chr  "male" "female" "male" "male" ...
##  $ Job             : num  2 2 1 2 2 1 2 3 1 3 ...
##  $ Housing         : chr  "own" "own" "own" "free" ...
##  $ Saving accounts : chr  NA "little" "little" "little" ...
##  $ Checking account: chr  "little" "moderate" NA "little" ...
##  $ Credit amount   : num  1169 5951 2096 7882 4870 ...
##  $ Duration        : num  6 48 12 42 24 36 24 36 12 30 ...
##  $ Purpose         : chr  "radio/TV" "radio/TV" "education" "furniture/equipment" ...
##  $ Risk            : chr  "good" "bad" "good" "good" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_double(),
##   ..   Age = col_double(),
##   ..   Sex = col_character(),
##   ..   Job = col_double(),
##   ..   Housing = col_character(),
##   ..   `Saving accounts` = col_character(),
##   ..   `Checking account` = col_character(),
##   ..   `Credit amount` = col_double(),
##   ..   Duration = col_double(),
##   ..   Purpose = col_character(),
##   ..   Risk = col_character()
##   .. )
```

The file has 11 features with combination of numberic and character fields. Later we need to convert the character fields to Factors

```
#Let us check the sample content
credit_data[1:10, ]
```

```
## # A tibble: 10 x 11
##       X1   Age Sex     Job Housing `Saving account~ `Checking accou~
##    <dbl> <dbl> <chr> <dbl> <chr>   <chr>            <chr>
##  1     0    67 male      2 own     <NA>             little
##  2     1    22 fema~     2 own     little           moderate
##  3     2    49 male      1 own     little           <NA>
##  4     3    45 male      2 free    little           little
##  5     4    53 male      2 free    little           little
##  6     5    35 male      1 free    <NA>             <NA>
##  7     6    53 male      2 own     quite rich       <NA>
##  8     7    35 male      3 rent    little           moderate
##  9     8    61 male      1 own     rich             <NA>
## 10     9    28 male      3 own     little           moderate
## # ... with 4 more variables: `Credit amount` <dbl>, Duration <dbl>,
## #   Purpose <chr>, Risk <chr>
```

There are two main observations from the above sample content.
a) The first column seems to be just the row numbers and hence can be removed.
b) Atleast there are 2 columns which has NA values. We also need to check for NA values for remaining columns and get them removed.

```r
# Removing the unwanted column
credit_data<- credit_data %>% dplyr::select(-X1)

# Let us first identify the total list of columns which has NA values
names(which(sapply(credit_data, anyNA)))
```

```
## [1] "Saving accounts"  "Checking account"
```

This confirms that there are only 2 columns with NA values as was identified with sample content

```r
# Total number of fields which has NA Values
sum(is.na(credit_data))
```

```
## [1] 577
```

```r
# count of NAs in each column
sapply(credit_data, function(count) sum(is.na(count))) %>%
   knitr::kable(col.names = c("\'NA\' Value counts"))
```

| | 'NA' Value counts |
|---|---:|
| Age | 0 |
| Sex | 0 |
| Job | 0 |
| Housing | 0 |
| Saving accounts | 183 |
| Checking account | 394 |
| Credit amount | 0 |
| Duration | 0 |
| Purpose | 0 |
| Risk | 0 |

```r
# Total number of rows which has NA Values
sum(!complete.cases(credit_data))
```

```
## [1] 478
```

This shows that out of 577 NA values there are almost 99 records which has NA values for both savings and checking accounts.

```r
# Removing the records with NA values from the dataset
na_credit_data<- na.omit(credit_data)

# Let us check from the count whether all NA values has been removed
nrow(na_credit_data)
```

```
## [1] 522
```

It confirms that all the NA values has been removed correctly and we have got the remaining 522 values

Before proceeding further we shall rename the columns which has space in their names as some of the built in functions throws error

```r
# Renaming the columns with space between the names
setnames(na_credit_data, old=c("Credit amount"), new=c("creditamount"))
setnames(na_credit_data, old=c("Saving accounts"), new=c("savingsaccounts"))
setnames(na_credit_data, old=c("Checking account"), new=c("checkingaccount"))

# Number of unique values in each column after removing NA values.
apply(na_credit_data, 2, function(x) length(unique(x))) %>%
   knitr::kable(col.names = c("Unique Value counts"))
```

| | Unique Value counts |
|---|---:|
| Age | 52 |
| Sex | 2 |
| Job | 4 |
| Housing | 3 |
| savingsaccounts | 4 |
| checkingaccount | 3 |
| creditamount | 503 |
| Duration | 30 |
| Purpose | 8 |
| Risk | 2 |

The above info is required later on how we will do Exploratory Data Analysis on diffrent field. For those numeric values with more number of unique values we need to work out a way to do analysis on those columns

```r
# List of different values for the "discrete" value features

unique(na_credit_data$savingsaccounts) %>%
   knitr::kable(col.names = c(" Savings Accounts Categories"))
```

| Savings Accounts Categories |
|---|
| little |
| moderate |
| quite rich |
| rich |

```r
unique(na_credit_data$checkingaccount) %>%
   knitr::kable(col.names = c("Checkig Accounts Categories"))
```

| Checkig Accounts Categories |
|---|
| moderate |
| little |
| rich |

```r
unique(na_credit_data$Housing) %>%
   knitr::kable(col.names = c("Housings Categories"))
```

| Housings Categories |
|---|
| own |
| free |
| rent |

```r
unique(na_credit_data$Purpose) %>%
   knitr::kable(col.names = c("List  of Purposes"))
```

| List of Purposes |
|---|
| radio/TV |
| furniture/equipment |
| car |
| business |
| domestic appliances |
| repairs |
| vacation/others |
| education |

```r
# Unique values of the numberic field "Job"
sort(unique(credit_data$Job)) %>%
   knitr::kable(col.names = c("Job Types"))
```

| Job Types |
|---:|
| 0 |
| 1 |
| 2 |
| 3 |

While the meaning of numberic value features are explicit, the feature "Job" needs its details to be explained. The feature "Job" has 4 different values meaning of which stands as below
0 - Unskilled and Non-resident,
1 - Unskilled and Resident,
2 - Skilled,
3 - Highly skilled

## 3.2) Checking for Correlation

```r
# Check for any strong correlation between any of the numberic variables
# so that such features can be excluded
credit_data %>%
   dplyr::select(Age, Job, creditamount, Duration) %>% cor() %>% knitr::kable()
```

|  | Age | Job | creditamount | Duration |
|---|---|---|---|---|
| Age | 1.0000000 | 0.0156732 | 0.0327164 | -0.0361364 |
| Job | 0.0156732 | 1.0000000 | 0.2853853 | 0.2109097 |
| creditamount | 0.0327164 | 0.2853853 | 1.0000000 | 0.6249842 |
| Duration | -0.0361364 | 0.2109097 | 0.6249842 | 1.0000000 |

The above output shows that there is no high correlation between any 2 features that warrants exclusion from being considered for building the models

## 3.3) Data exploration and visualization

### 3.3.1) Distribution of Risk data

```
# Distribution of Risk feature in the original data set
ct_risk <- CrossTable(credit_data$Risk)
```

```
ct_risk$t %>% knitr::kable()
```

| bad | good |
|---|---|
| 300 | 700 |

The initial set of data shows high prevalence of good credit rating over bad credit rating. This will be significantly impacting any model that we derive .

```
# Checking for "Prevalence" issue post remvoing the NA values
ct_risk_na <- CrossTable(na_credit_data$Risk)
```

```
ct_risk_na$t %>% knitr::kable()
```

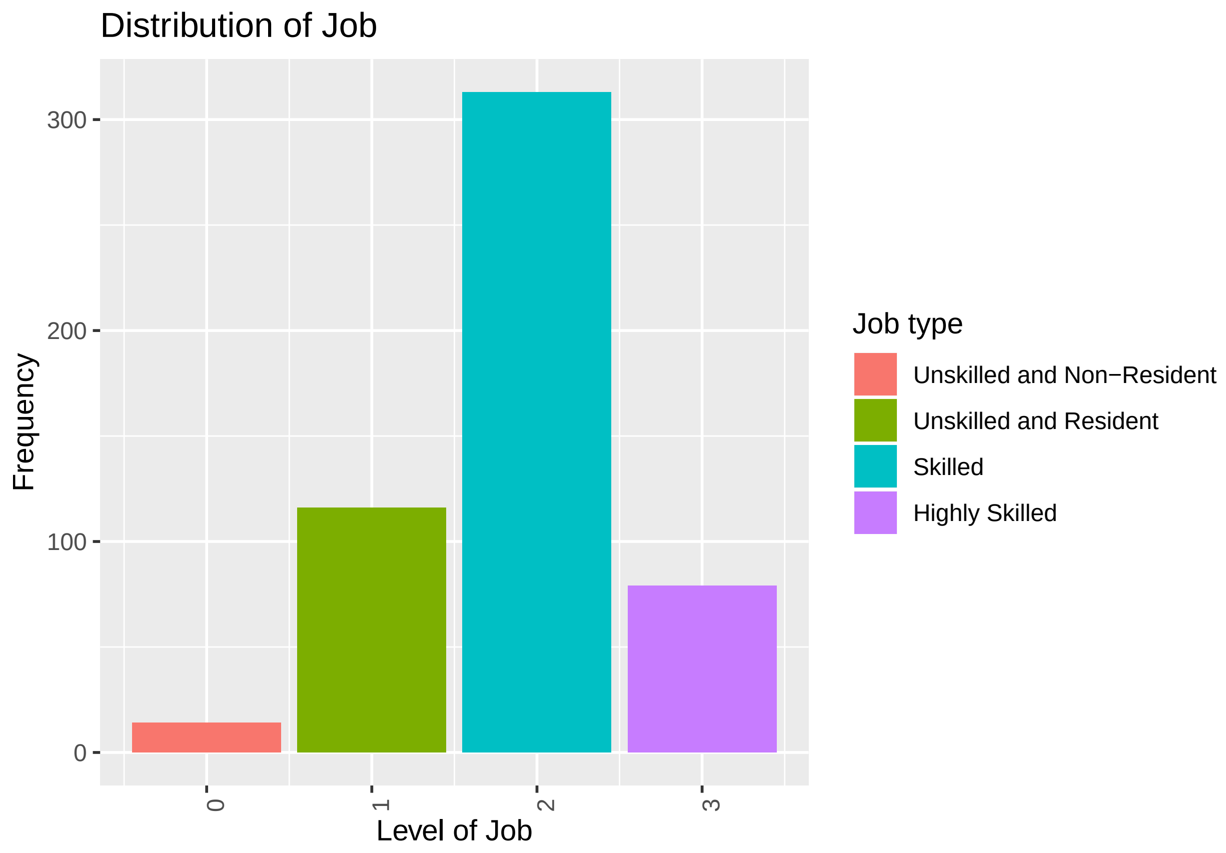
| bad | good |
|---|---|
| 231 | 291 |

Now post removing the NA records both the possible outcomes of Credit Risk category are almost evenly balanced in the data set

We shall do some more EDA to figure out whether any finetuning of the data can be done and see whether that will reduce the prevalence issue .

### 3.3.2) Distribution of Job data

```
# Now let us analyse the distribution of Jobs from the dataset
ggplot(na_credit_data, aes(Job) ) +
   geom_bar(aes(fill = as.factor(na_credit_data$Job))) +
   scale_fill_discrete(name="Job type",
                        labels=c( "Unskilled and Non-Resident","Unskilled and Resident",
                                  "Skilled", "Highly Skilled")) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(x= "Level of Job",y= "Frequency" , title = "Distribution of Job")
```

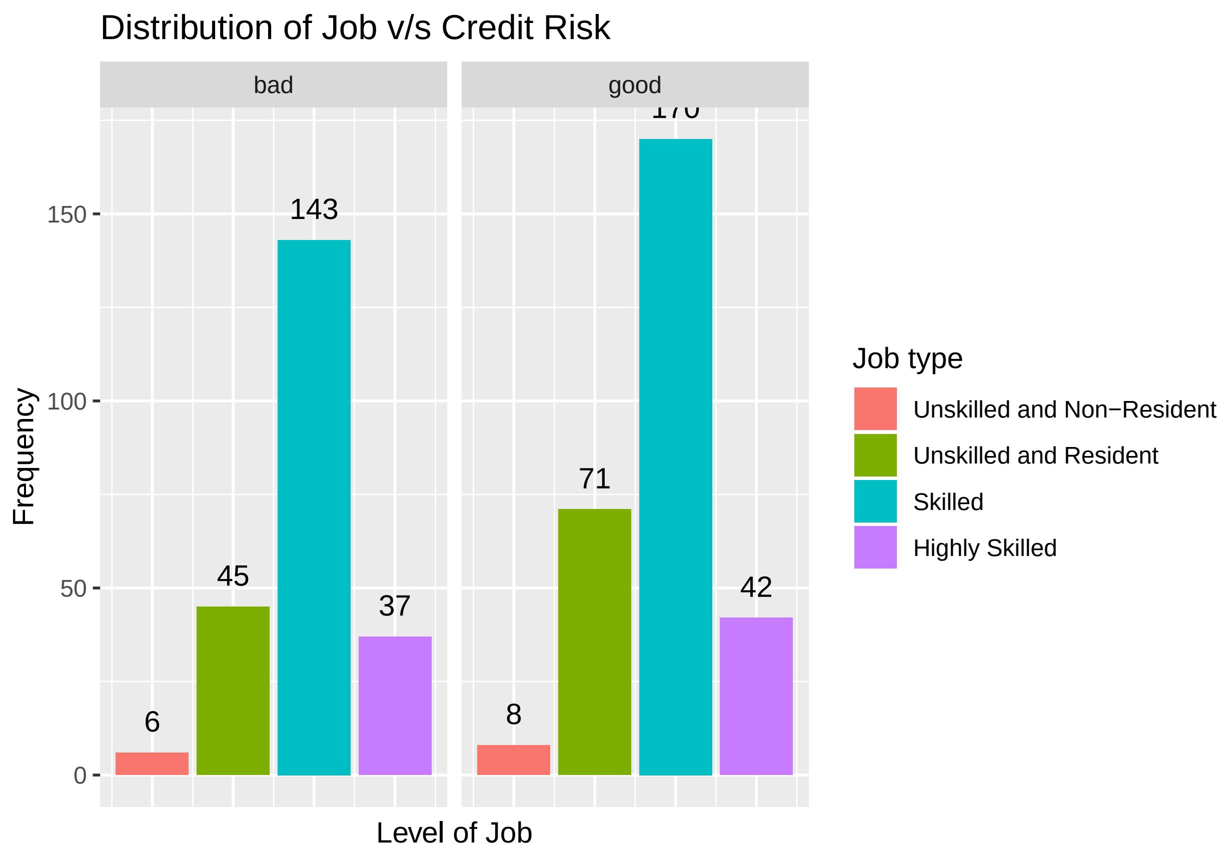


The graph clearly shows that prevalence of Skilled worker is far higher over other types where as Unskilled and Non Resident is very meager. This is understandable as Bank would prefer to provide more loans to skilled worker as the probability to them If any model is build with feature selection based on Job then this will have very high impact on the outcome.

```
# Below is the table in terms of percentage of distribution
CrossTable(na_credit_data$Job, format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##           |        0  |        1  |        2  |        3  |
##           |-----------|-----------|-----------|-----------|
##           |       14  |      116  |      313  |       79  |
##            |     2.68% |    22.22% |    59.96% |    15.13% |
##            |-----------|-----------|-----------|-----------|
##
##
```

```
# We shall plot the distribution of Job against Credit Risk
ggplot(na_credit_data, aes(Job) ) +
  geom_bar(aes(fill = as.factor(na_credit_data$Job))) +
  scale_fill_discrete(name="Job type",
                      labels=c( "Unskilled and Non-Resident","Unskilled and Resident",
                                "Skilled", "Highly Skilled")) +
  theme(axis.text.x=element_blank(),axis.ticks.x=element_blank()) +
  labs(x= "Level of Job",y= "Frequency" , title = "Distribution of Job v/s Credit Risk") +
  geom_text(stat='count', aes(label=..count..), vjust = -1) +
  facet_grid( . ~ Risk)
```



From the graph we can see that Within each of the given Job type there is no prevalence issue in terms of credit risk distribution. Surprisingly the number of bad loans is considerably higher even for the Skilled worker categories where the repayment by the unskilled & resident type borrowers are repaying the loan better .

```r
# Below table provides the percentage of distribution of Job against Credit Risk
CrossTable(na_credit_data$Job, na_credit_data$Risk,
           prop.c = FALSE , prop.t = FALSE , prop.chisq = FALSE ,
           format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##                    | na_credit_data$Risk
## na_credit_data$Job |       bad  |      good  | Row Total |
## -------------------|-----------|-----------|-----------|
##                  0 |         6  |         8  |        14  |
##                    |    42.86% |    57.14% |     2.68% |
## -------------------|-----------|-----------|-----------|
##                  1 |        45  |        71  |       116  |
##                    |    38.79% |    61.21% |    22.22% |
## -------------------|-----------|-----------|-----------|
##                  2 |       143  |       170  |       313  |
##                    |    45.69% |    54.31% |    59.96% |
## -------------------|-----------|-----------|-----------|
##                  3 |        37  |        42  |        79  |
##                    |    46.84% |    53.16% |    15.13% |
## -------------------|-----------|-----------|-----------|
##       Column Total |       231  |       291  |       522  |
## -------------------|-----------|-----------|-----------|
##
##
```

### 3.3.3) Distribution of Gender data

```r
# Checking the Distribution of Male / Female ratio
CrossTable(na_credit_data$Sex, format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##           |   female  |     male  |
##           |-----------|-----------|
##           |      168  |      354  |
##           |    32.18% |    67.82% |
##           |-----------|-----------|
##
##
```

The data shows that almost 2/3 of the population is male and clearly this will have impact on the model where feature selection explicitly involves the gender

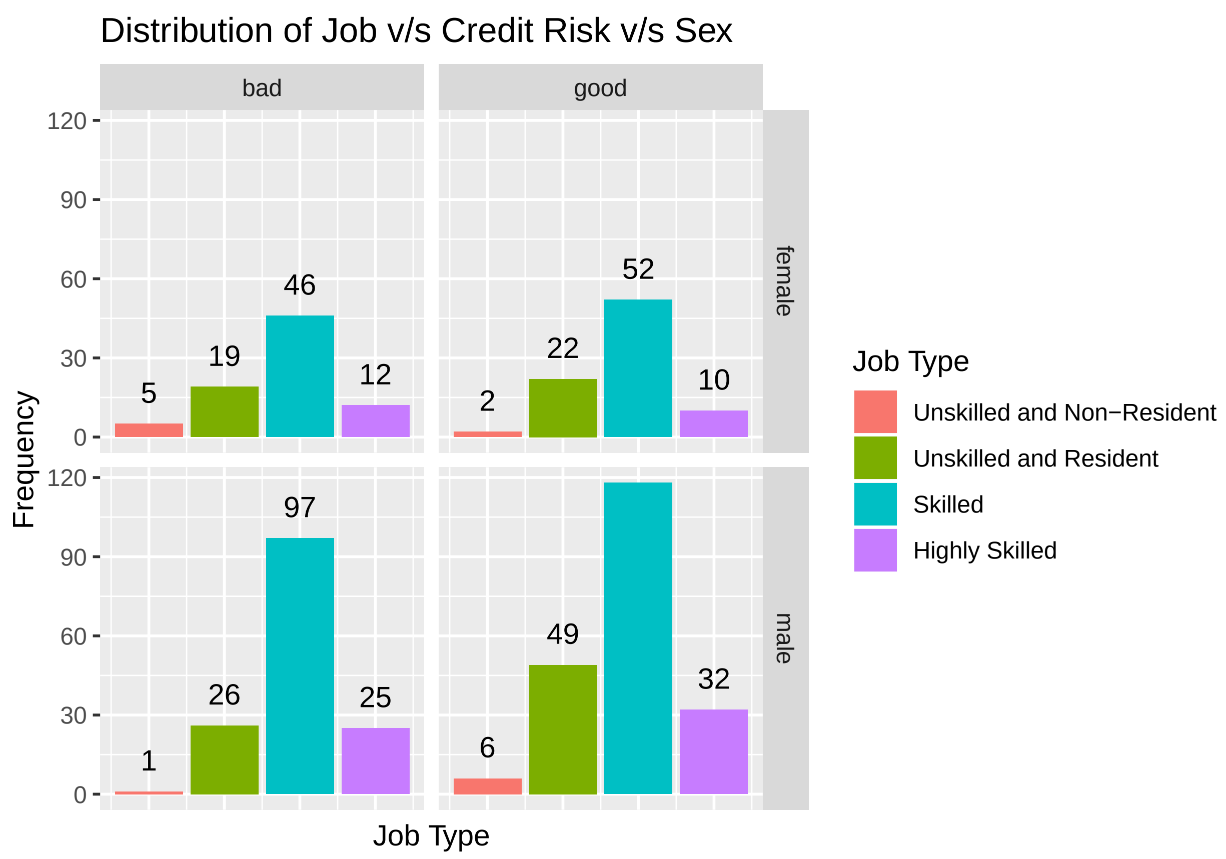
```r
# Counts on no.of occurences based on the 3 predictors Sex, Job & Risk category Split on the
crosstab(na_credit_data, row.vars = c("Sex", "Job"),
         col.vars = "Risk", type = "frequency")
```

```
##            Risk bad good Sum
## Sex    Job
## female 0          5    2   7
##        1         19   22  41
##        2         46   52  98
##        3         12   10  22
##        Sum       82   86 168
## male   0          1    6   7
##        1         26   49  75
##        2         97  118 215
##        3         25   32  57
##        Sum      149  205 354
## Sum    0          6    8  14
##        1         45   71 116
##        2        143  170 313
##        3         37   42  79
##        Sum      231  291 522
```

```r
# Percentage distribution of data based on the 3 features
crosstab(na_credit_data, row.vars = c("Sex", "Job"), col.vars = "Risk", type = "t")
```

```
##            Risk    bad   good    Sum
## Sex    Job
## female 0          0.96   0.38   1.34
##        1          3.64   4.21   7.85
##        2          8.81   9.96  18.77
##        3          2.30   1.92   4.21
##        Sum       15.71  16.48  32.18
## male   0          0.19   1.15   1.34
##        1          4.98   9.39  14.37
##        2         18.58  22.61  41.19
##        3          4.79   6.13  10.92
##        Sum       28.54  39.27  67.82
## Sum    0          1.15   1.53   2.68
##        1          8.62  13.60  22.22
##        2         27.39  32.57  59.96
##        3          7.09   8.05  15.13
##        Sum       44.25  55.75 100.00
```

```r
# Now we shall plot the distribution of Job v/s Credit Risk v/s Sex
ggplot(na_credit_data, aes(Job) ) +
   geom_bar(aes(fill = as.factor(na_credit_data$Job))) +
   scale_fill_discrete(name="Job Type",
                       labels=c( "Unskilled and Non-Resident","Unskilled and Resident",
                                 "Skilled", "Highly Skilled")) +
   theme(axis.text.x=element_blank(),axis.ticks.x=element_blank()) +
   labs(x= "Job Type",y= "Frequency" , title = "Distribution of Job v/s Credit Risk v/s Sex") +
   geom_text(stat='count', aes(label=..count..), vjust = -1) +
   facet_grid( Sex ~ Risk)
```

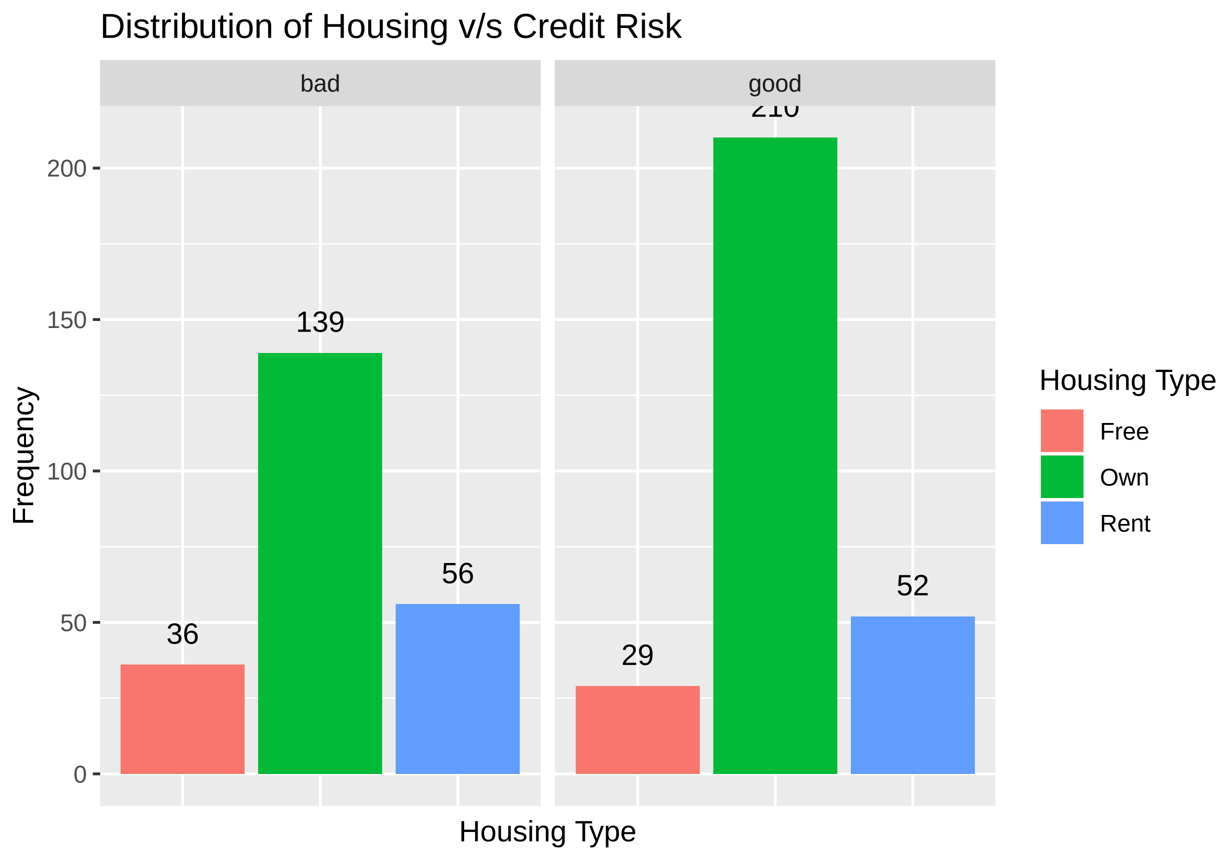


### 3.3.4) Distribution of Housing data

```
# Below is the table for data distribution of Housing v/s Credit risk
CrossTable(na_credit_data$Housing, na_credit_data$Risk,
           prop.c = FALSE , prop.t = FALSE , prop.chisq = FALSE ,
           format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##                        | na_credit_data$Risk
## na_credit_data$Housing |       bad  |      good  | Row Total |
## -----------------------|-----------|-----------|-----------|
##                   free |        36  |        29  |        65  |
##                        |    55.38% |    44.62% |    12.45% |
## -----------------------|-----------|-----------|-----------|
##                    own |       139  |       210  |       349  |
##                        |    39.83% |    60.17% |    66.86% |
## -----------------------|-----------|-----------|-----------|
##                   rent |        56 |        52 |       108 |
##                        |    51.85% |    48.15% |    20.69% |
## -----------------------|-----------|-----------|-----------|
##           Column Total |       231 |       291 |       522 |
## -----------------------|-----------|-----------|-----------|
##
##
```

```
# Graph plot for Housing distribution v/s Credit risk in the data set
ggplot(na_credit_data, aes(Housing) ) +
    geom_bar(aes(fill = as.factor(na_credit_data$Housing))) +
    scale_fill_discrete(name="Housing Type", labels=c( "Free","Own", "Rent")) +
    theme(axis.text.x=element_blank(),axis.ticks.x=element_blank()) +
    labs(x= "Housing Type",y= "Frequency" , title = "Distribution of Housing v/s Credit Risk") +
    geom_text(stat='count', aes(label=..count..), vjust = -1) +
    facet_grid( . ~ Risk)
```



Clearly we see there is very High Prevalence of data for "own" Housing category. Data distribution shows that the Bank is already taking precaution of giving less loan for those who are living in free house. The data also Vindicates that majority of those in Free housing are not paying the loan. Interstingly we observe that majority of those in rented house are also not paying back the loan though it is slightly lesser than the Free housing category case.

### 3.3.5) Distribution of creditamount data

Now we shall do the EDA analysis on numberic data fields - Creditamount, Duration & Age

We have already noticed above that the number of unique values of Creditamount, Loan duration and Age are very high. Hence to analyse them we shall group them into different bucket ranges

```
# Converting the credit amount column to range of 2500 dollars each
credit_range  <-
   function(n) {
        if(n %% 2500 > 0)
           paste("Credit(",((floor(n/2500))*2500)+1,"-",(floor(n/2500)+1)*2500, ")" ,sep="")
        else
           paste("Credit(",((floor(n/2500)-1)*2500)+1,"-",(floor(n/2500))*2500 ,")", sep="")
   }

na_credit_data<- na_credit_data %>%
                                 mutate(CreditRange = sapply(creditamount, credit_range))

# Changing the factor values sequence to get them in ascending order
# for our Cross table output
na_credit_data$CreditRange <-
  factor(na_credit_data$CreditRange,
         levels = names(sort(table(na_credit_data$CreditRange), decreasing = TRUE) ))

# Table for distribution of Credit amount
CrossTable(na_credit_data$CreditRange, na_credit_data$Risk,
           prop.c = FALSE , prop.t = FALSE , prop.chisq = FALSE ,
           format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##                            | na_credit_data$Risk
## na_credit_data$CreditRange |      bad  |     good  | Row Total |
## ---------------------------|-----------|-----------|-----------|
##             Credit(1-2500) |      114  |      165  |      279  |
##                            |   40.86%  |   59.14%  |   53.45%  |
## ---------------------------|-----------|-----------|-----------|
##          Credit(2501-5000) |       60  |       87  |      147  |
##                            |   40.82%  |   59.18%  |   28.16%  |
## ---------------------------|-----------|-----------|-----------|
##          Credit(5001-7500) |       25  |       28  |       53  |
##                            |   47.17%  |   52.83%  |   10.15%  |
## ---------------------------|-----------|-----------|-----------|
##         Credit(7501-10000) |       14  |        7  |       21  |
##                            |   66.67%  |   33.33%  |    4.02%  |
## ---------------------------|-----------|-----------|-----------|
##        Credit(10001-12500) |         7  |         3  |        10  |
##                            |    70.00%  |    30.00%  |     1.92%  |
## ---------------------------|------------|------------|------------|
##        Credit(12501-15000) |         8  |         0  |         8  |
##                            |   100.00%  |     0.00%  |     1.53%  |
## ---------------------------|------------|------------|------------|
##        Credit(15001-17500) |         2  |         1  |         3  |
##                            |    66.67%  |    33.33%  |     0.57%  |
## ---------------------------|------------|------------|------------|
##        Credit(17501-20000) |         1  |         0  |         1  |
##                            |   100.00%  |     0.00%  |     0.19%  |
## ---------------------------|------------|------------|------------|
##               Column Total |       231  |       291  |       522  |
## ---------------------------|------------|------------|------------|
##
##
```

Data also shows that the bank give major portion of its loan for small amount borrowers ( less than 5000) and less number of loans for higher amounts Also from the above table we can notice that we would need more data in the Credit range between 12501-15000 & 17501-20000 for them to be considered as currently all the credits are bad and evaluating them will be of less use Similarly for credit range 15501-17500 can be ignored as we just have around 0.5% in the entire data set

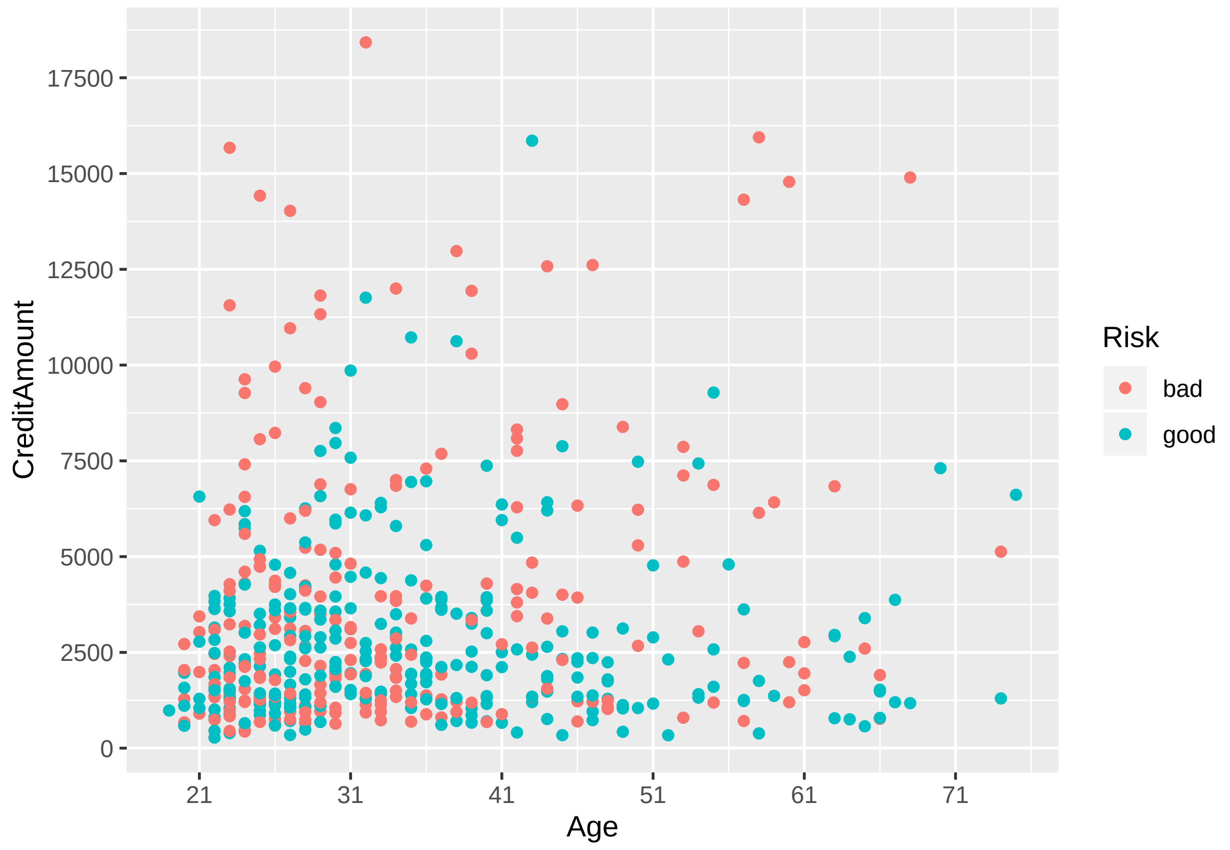
### 3.3.6) Distribution of Age data

```
# Converting the age column to range of 10 years each
age_range<- function(n) {
                if(n %% 10 > 0)
                  paste("Age(",((floor(n/10))*10)+1,"-",(floor(n/10)+1)*10, ")" ,sep="")
                else
                  paste("Age(",((floor(n/10)-1)*10)+1,"-",(floor(n/10))*10 ,")", sep="")
            }

na_credit_data<- na_credit_data %>%
                                     mutate(AgeRange = sapply(Age, age_range))

CrossTable(na_credit_data$AgeRange, na_credit_data$Risk,
           prop.c = FALSE , prop.t = FALSE , prop.chisq = FALSE , format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##                          | na_credit_data$Risk
## na_credit_data$AgeRange |       bad  |      good  | Row Total |
## ------------------------|------------|------------|------------|
##              Age(11-20) |         4  |         5  |         9  |
##                          |     44.44% |     55.56% |      1.72% |
## -------------------------|------------|------------|------------|
##               Age(21-30) |       111  |       123  |       234  |
##                          |     47.44% |     52.56% |     44.83% |
## -------------------------|------------|------------|------------|
##               Age(31-40) |        58  |        86  |       144  |
##                          |     40.28% |     59.72% |     27.59% |
## -------------------------|------------|------------|------------|
##               Age(41-50) |        33  |        43  |        76  |
##                          |     43.42% |     56.58% |     14.56% |
## -------------------------|------------|------------|------------|
##               Age(51-60) |        16  |        18  |        34  |
##                          |     47.06% |     52.94% |      6.51% |
## -------------------------|------------|------------|------------|
##               Age(61-70) |         8  |        14  |        22  |
##                          |     36.36% |     63.64% |      4.21% |
## -------------------------|------------|------------|------------|
##               Age(71-80) |         1  |         2  |         3  |
##                          |     33.33% |     66.67% |      0.57% |
## -------------------------|------------|------------|------------|
##             Column Total |       231  |       291  |       522  |
## -------------------------|------------|------------|------------|
##
##
```

From the above table we can see that bank provides very less number of credits to people above 70 year which is understandable. As this dataset is very less we can exclude it from our consideration when we build the models

```
# We shall plot a graph with the above age and credit amount

Age <-  na_credit_data$Age
CreditAmount <- na_credit_data$creditamount
Risk <- na_credit_data$Risk

qplot( Age, CreditAmount, colour = Risk ) +
   scale_x_continuous(breaks = c(seq(11,80,10))) +
   scale_y_continuous(breaks = c(seq(0,20000,2500)))
```

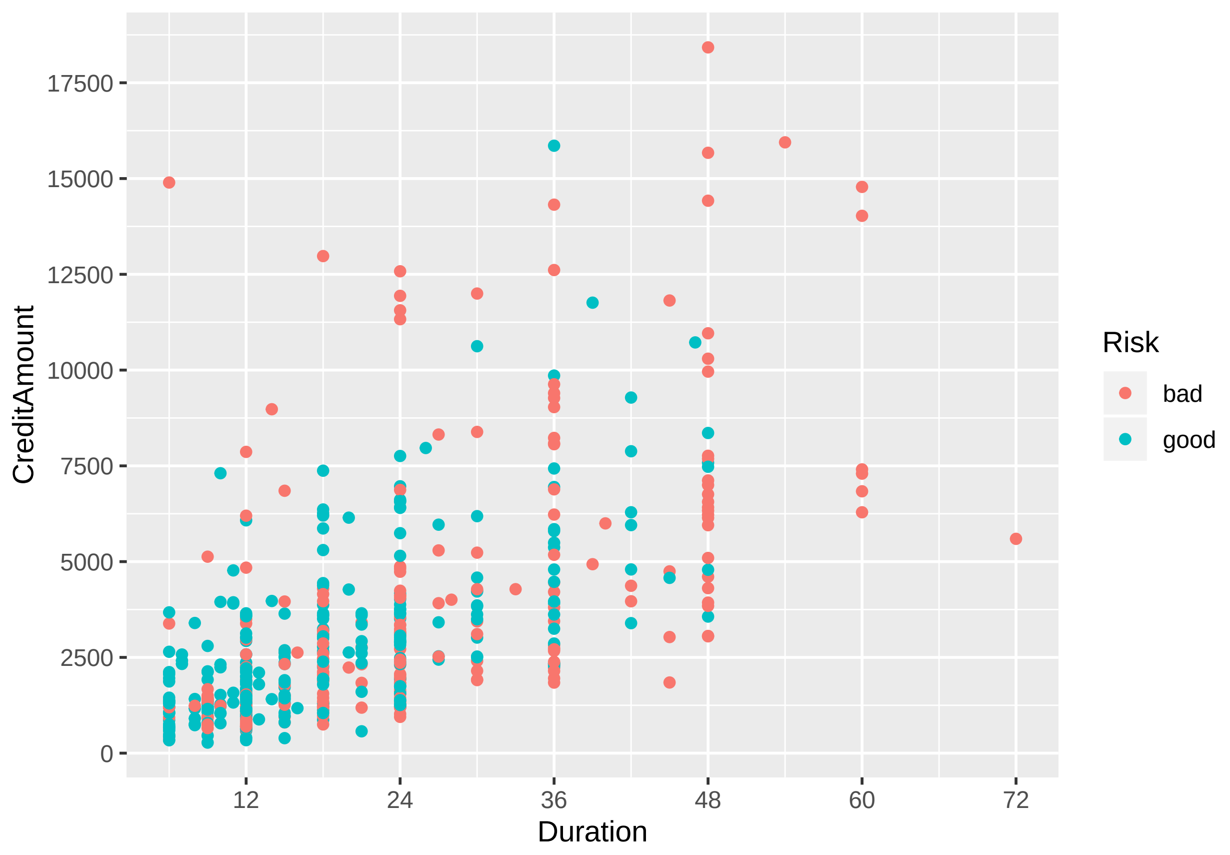


### 3.3.7) Distribution of Duration data

```
#Converting the Durations column to range of 12 months each
duration_range <-
   function(n) {
            if(n %% 12 > 0)
                 paste("Months(",((floor(n/12))*12)+1,"-",(floor(n/12)+1)*12, ")" ,sep="")
            else
                 paste("Months(",((floor(n/12)-1)*12)+1,"-",(floor(n/12))*12 ,")", sep="")
   }

na_credit_data <-
   na_credit_data %>%
      mutate(MonthsRange = sapply(Duration, duration_range))

CrossTable(na_credit_data$MonthsRange, na_credit_data$Risk,
           prop.c = FALSE , prop.t = FALSE , prop.chisq = FALSE , format = 'SPSS', digits = 2)
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  522
##
##                              | na_credit_data$Risk
## na_credit_data$MonthsRange |       bad  |      good  | Row Total |
## ---------------------------|-----------|-----------|-----------|
##              Months(1-12) |        60  |       130  |       190  |
##                            |    31.58% |    68.42% |    36.40% |
## ---------------------------|-----------|-----------|-----------|
##             Months(13-24) |        88  |       112  |       200  |
##                            |    44.00% |    56.00% |    38.31% |
## ---------------------------|-----------|-----------|-----------|
##             Months(25-36) |        43  |        35  |        78  |
##                            |    55.13% |    44.87% |    14.94% |
## ---------------------------|-----------|-----------|-----------|
##             Months(37-48) |        32  |        14  |        46  |
##                            |    69.57% |    30.43% |     8.81% |
## ---------------------------|-----------|-----------|-----------|
##             Months(49-60) |         7  |         0  |         7  |
##                            |   100.00% |     0.00% |     1.34% |
## ---------------------------|-----------|-----------|-----------|
##             Months(61-72) |         1  |         0  |         1  |
##                            |   100.00% |     0.00% |     0.19% |
## ---------------------------|-----------|-----------|-----------|
##              Column Total |       231  |       291  |       522  |
## ---------------------------|-----------|-----------|-----------|
##
##
```

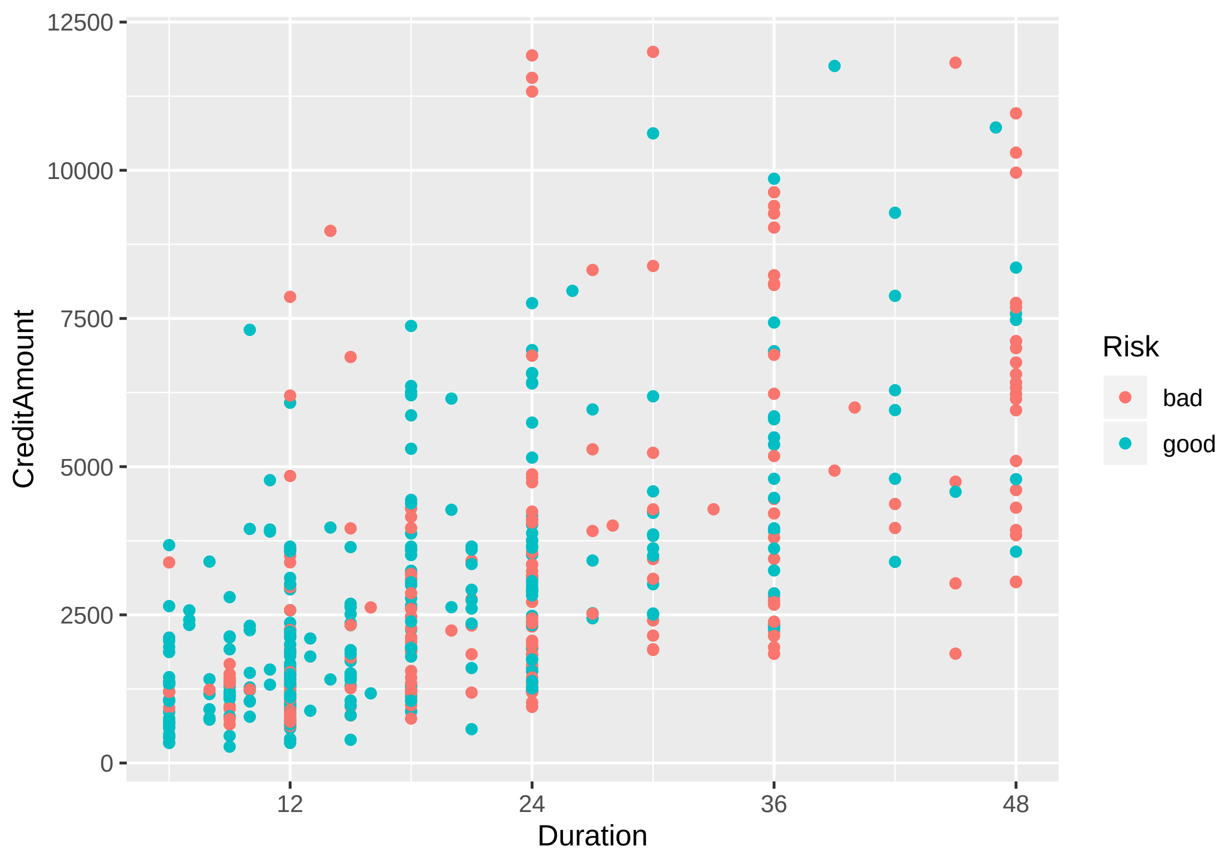
```
# We shall plot a graph with above Duration and credit amount

Duration <- na_credit_data$Duration
CreditAmount <- na_credit_data$creditamount
Risk <- na_credit_data$Risk

qplot(Duration, CreditAmount, colour = Risk ) +
  scale_x_continuous(breaks = c(seq(0,90,12))) +
  scale_y_continuous(breaks = c(seq(0,20000,2500)))
```



We shall remove those credits which has loan duration of more than 4 years ( 49-60 and 61-72 Months) as in the current data set all of them are bad credits and using them for modelling will not yield any useful result

## 3.4) Removing outliers & additional columns

```
# Remove those outlier values that will cause undue influence in the models
na_credit_data <- na_credit_data %>% filter(Age < 71)
na_credit_data <- na_credit_data %>% filter(creditamount < 12501)
na_credit_data <- na_credit_data %>% filter(Duration < 49)

# Plot - post removing the values

Duration <- na_credit_data$Duration
CreditAmount <- na_credit_data$creditamount
Risk <- na_credit_data$Risk

qplot(Duration, CreditAmount, colour = Risk ) +
  scale_x_continuous(breaks = c(seq(0,90,12))) +
  scale_y_continuous(breaks = c(seq(0,20000,2500)))
```



```r
#Remove the redunt columns of Age, Duration and Credit amount
na_credit_data <- na_credit_data %>%
                                           dplyr::select ( -contains("Range") )

# Before proceeding to build the models We shall change the character columns to factors
na_credit_data$Risk <- as.factor(na_credit_data$Risk)
na_credit_data$Sex <- as.factor(na_credit_data$Sex)
na_credit_data$Housing <- as.factor(na_credit_data$Housing)
na_credit_data$Purpose <- as.factor(na_credit_data$Purpose)
na_credit_data$savingsaccounts <-
   as.factor(na_credit_data$savingsaccounts)
na_credit_data$checkingaccount <-
   as.factor(na_credit_data$checkingaccount)
```

## 4) Building the Models

### 4.1) Algorithm selection

After extensively analyzing different algorithms that are best suited for classification purpose the below list of algorithms have been shortlisted for consideration for our Credit risk Model

1) Logistic Regression
2) Decision Tree
3) Random Forest
4) Gradient Boosting

    5) Knn
    6) Linear Discriminant Analysis (LDA)
    7) Quadriatic Discriminant Analysis (QDA)
    8) Support Vector Machines (SVM)

Of the above listed methods where ever possible we shall finetune the outcome with those features ( variables) that are identified to have made significant impact in the model building

Also for each of the above algorithm we shall build models using both
    1) Base method from each of the resepctive algorithm library like Random Forest, gbm (gradient Boosting Model) etc
    2) Cross validation using the train function from the Caret library

## 4.2) Create data partition

```
# Segregating the data set into training and testing purpose
set.seed(1250)
train_index<- createDataPartition(y = na_credit_data$Risk,
                                  times = 1, p = 0.9, list = FALSE)
train_data<- na_credit_data [train_index,]
test_data<- na_credit_data [-train_index,]
```

## 4.3) Model building based on base method

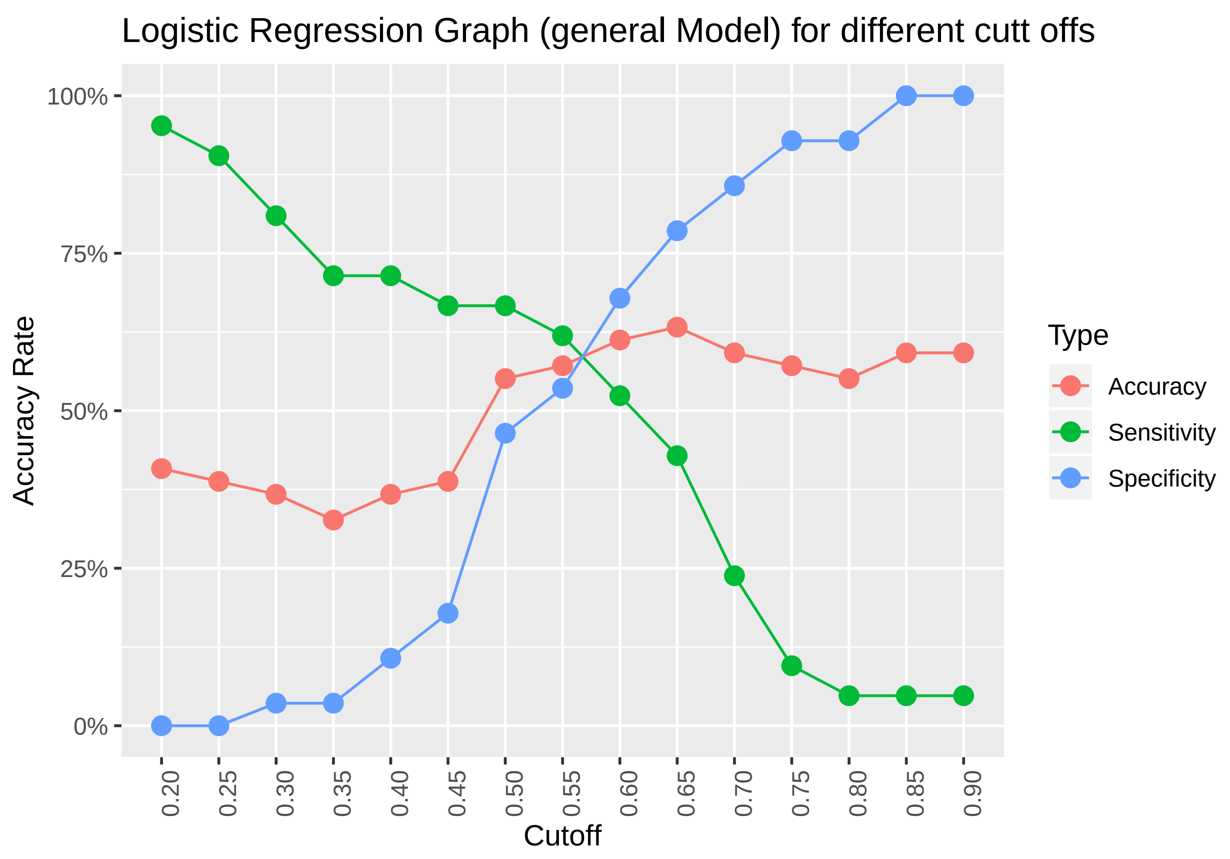
```
# Dataframe to store the results of different models for comparisions
df_gen_models <- data.frame(matrix(vector(),ncol=4))
colnames(df_gen_models) <- c("Model", "Accuracy", "Sensitivity", "Specificity" )
```

### 4.3.1) Logistic Regression Model

```
# Building the Logistic Regression Model
glm_gen_model <- glm(Risk~ ., data = train_data, family = 'binomial')
glm_gen_predict<- predict(glm_gen_model, test_data, type = 'response')

# Data frame to store the results of different cutoff value of Logistic regression
df_glm_gen_model <- data.frame(matrix(vector(),ncol=5))
colnames(df_glm_gen_model) <- c("Model","Cutoff", "Accuracy", "Sensitivity", "Specificity" )

# Function to evaluate and store the results of Logistic regression model
# for different cutoff values
glmfun <- function(a) {
            glm_gen_predict_val<- factor(ifelse(glm_gen_predict>= a, "bad", "good"))
            glm_gen_cf <- confusionMatrix( glm_gen_predict_val, test_data$Risk)
            df_glm_gen_model <<-
               df_glm_gen_model %>%
                   add_row(Model = "Logistic Regression",
                           Cutoff = a,
                           Accuracy = glm_gen_cf$overall['Accuracy'] ,
                           Sensitivity = glm_gen_cf$byClass['Sensitivity'],
                              Specificity =  glm_gen_cf$byClass['Specificity'])
           }

seqval<- seq(0.20,.90, 0.05)
glm_output <- sapply(seqval,glmfun)

df_glm_gen_model %>% dplyr::select(-Model) %>% knitr::kable()
```

| Cutoff | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| 0.20 | 0.4081633 | 0.9523810 | 0.0000000 |
| 0.25 | 0.3877551 | 0.9047619 | 0.0000000 |
| 0.30 | 0.3673469 | 0.8095238 | 0.0357143 |
| 0.35 | 0.3265306 | 0.7142857 | 0.0357143 |
| 0.40 | 0.3673469 | 0.7142857 | 0.1071429 |
| 0.45 | 0.3877551 | 0.6666667 | 0.1785714 |
| 0.50 | 0.5510204 | 0.6666667 | 0.4642857 |
| 0.55 | 0.5714286 | 0.6190476 | 0.5357143 |
| 0.60 | 0.6122449 | 0.5238095 | 0.6785714 |
| 0.65 | 0.6326531 | 0.4285714 | 0.7857143 |
| 0.70 | 0.5918367 | 0.2380952 | 0.8571429 |
| 0.75 | 0.5714286 | 0.0952381 | 0.9285714 |
| 0.80 | 0.5510204 | 0.0476190 | 0.9285714 |
| 0.85 | 0.5918367 | 0.0476190 | 1.0000000 |
| 0.90 | 0.5918367 | 0.0476190 | 1.0000000 |

```
# Transforming the values to different data frame format for graphical representation
df_glm_gen_model_tmp <- df_glm_gen_model
df_glm_gen_model_tmp <- gather(df_glm_gen_model_tmp, Type, Val, Accuracy:Specificity)
df_glm_gen_model_tmp <- df_glm_gen_model_tmp [order(df_glm_gen_model_tmp$Cutoff), ]


# Graphical plot for the Logistic regression model for different cutoff values
ggplot(df_glm_gen_model_tmp, aes(Cutoff, Val, color = Type)) +
   geom_line() +
   geom_point(size = 3) +
   scale_y_continuous(labels = scales::percent) +
   scale_x_continuous(breaks = seq(0.1, 0.9, by = 0.05)) +
   theme(axis.text.x = element_text(angle = 90, hjust = 1),
         panel.grid.minor.x = element_blank()) + labs(y = "Accuracy Rate",
        title = "Logistic Regression Graph (general Model) for different cutt offs")
```



Looking at the graph we can see that for the cutoff value of 0.50 both the sensitivity and specificity are significantly better and hence forms the best options .

Now we shall finetune the Logistic Regression model by finding out the list of variables that made significant impact in the outcome of the model

```
# List out the features that made significant impact on the model building
glm_gen_model_vi <- as.data.frame(varImp(glm_gen_model))
glm_gen_model_vi <- data.frame( feature  = rownames(glm_gen_model_vi),
                               impact = glm_gen_model_vi$Overall)
glm_gen_model_vi[order(glm_gen_model_vi$impact,decreasing = T),][1:5,] %>% knitr::kable()
```

| | feature | impact |
|---|---|---|
| 12 | Duration | 4.927872 |
| 10 | checkingaccountrich | 2.902467 |
| 8 | savingsaccountsrich | 2.426665 |
| 2 | Sexmale | 2.213194 |
| 15 | Purposeeducation | 1.928841 |

The table shows the top 5 features in the order of their impact. We shall use them to finetune our Logistic regression Model

```
# Creating the data frame to store the results of different cutoff value for the
# finetuned version of  Logistic regression
```

```r
df_glm_gen_model_ft <- data.frame(matrix(vector(),ncol=5))
colnames(df_glm_gen_model_ft) <- c("Model", "Cutoff", "Accuracy", "Sensitivity", "Specificity" )
seqval<- seq(0.20,.90, 0.05)

# Building the finetuned Logistic Regression Model
glm_gen_model_ft <- glm(Risk~ Duration + checkingaccount + savingsaccounts + Sex , data = train_data, fa
glm_gen_predict_ft <- predict(glm_gen_model_ft, test_data, type = 'response')


# Function to evaluate and store the results of finetuned version of the Logistic regression model
# for different cutoff values
glmfun_ft <-
   function(a) {
       glm_gen_predict_val_ft <- factor(ifelse(glm_gen_predict_ft >= a, "bad", "good"))
       glm_gen_cf_ft<- confusionMatrix(glm_gen_predict_val_ft, test_data$Risk)
       df_glm_gen_model_ft <<-
           df_glm_gen_model_ft %>%
               add_row(Model = "GLM_FineTuned",
                       Cutoff = a,
                       Accuracy = glm_gen_cf_ft$overall['Accuracy'] ,
                       Sensitivity = glm_gen_cf_ft$byClass['Sensitivity'],
                       Specificity =  glm_gen_cf_ft$byClass['Specificity'])
   }

glm_ft_output <- sapply(seqval, glmfun_ft)
df_glm_gen_model_ft %>% dplyr::select(-Model) %>%knitr::kable()
```

| Cutoff | Accuracy | Sensitivity | Specificity |
|---:|---:|---:|---:|
| 0.20 | 0.3877551 | 0.9047619 | 0.0000000 |
| 0.25 | 0.4081633 | 0.9047619 | 0.0357143 |
| 0.30 | 0.3673469 | 0.8095238 | 0.0357143 |
| 0.35 | 0.4285714 | 0.8095238 | 0.1428571 |
| 0.40 | 0.4081633 | 0.7142857 | 0.1785714 |
| 0.45 | 0.4693878 | 0.6666667 | 0.3214286 |
| 0.50 | 0.5306122 | 0.6190476 | 0.4642857 |
| 0.55 | 0.5714286 | 0.6190476 | 0.5357143 |
| 0.60 | 0.5918367 | 0.6190476 | 0.5714286 |
| 0.65 | 0.6122449 | 0.4761905 | 0.7142857 |
| 0.70 | 0.5510204 | 0.2380952 | 0.7857143 |
| 0.75 | 0.5714286 | 0.1428571 | 0.8928571 |
| 0.80 | 0.5714286 | 0.0476190 | 0.9642857 |
| 0.85 | 0.5918367 | 0.0476190 | 1.0000000 |
| 0.90 | 0.5918367 | 0.0476190 | 1.0000000 |

Looking at the above table we can see that for the cutoff value of 0.60 the combination of both the sensitivity & specificity are of significant level when compared to other pairs. . Comparing the best values of both the models Logistic regression we see that vanila model has better results in terms of sensitivity . So as far as Logistic regression model is concerned we will select the Vanila model for comparing with the results of other models

```
#Adding the Short listed values of the Logistic regression model to the comparision data frame
df_gen_models <-  df_glm_gen_model %>% filter(Cutoff== 0.50) %>%
                    dplyr::select(-Cutoff)  %>% rbind(df_gen_models, .)
```

### 4.3.2) Decision Tree Model

```
# Building the Decision Tree Model
dt_gen_model  <- rpart(Risk~., data=train_data)
dt_gen_predict <- predict(dt_gen_model, test_data)
dt_gen_predict <- factor(ifelse(dt_gen_predict[,"bad"] >= .50, "bad", "good"))
dt_gen_cf<- confusionMatrix( dt_gen_predict, test_data$Risk)
dt_gen_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   12   11
##       good   9   17
##
##                Accuracy : 0.5918
##                  95% CI : (0.4421, 0.73)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.4454
##
##                   Kappa : 0.1765
##
##  Mcnemar's Test P-Value : 0.8231
##
##             Sensitivity : 0.5714
##             Specificity : 0.6071
##          Pos Pred Value : 0.5217
##          Neg Pred Value : 0.6538
##              Prevalence : 0.4286
##          Detection Rate : 0.2449
##    Detection Prevalence : 0.4694
##       Balanced Accuracy : 0.5893
##
##        'Positive' Class : bad
##
```

```
# Listing the TOP 5 variables that made significant impact in  Decision tree model
dt_gen_model_vi <- as.data.frame(varImp(dt_gen_model))
dt_gen_model_vi <- data.frame(feature   = rownames(dt_gen_model_vi),
                              impact = dt_gen_model_vi$Overall)
dt_gen_model_vi[order(dt_gen_model_vi$impact,decreasing = T),][1:5,] %>%knitr::kable()
```

| | feature | impact |
|---|---|---|
| 3 | creditamount | 49.55898 |
| 4 | Duration | 33.24255 |

| | feature | impact |
|---|---|---|
| 2 | checkingaccount | 23.69406 |
| 7 | Purpose | 21.57166 |
| 8 | savingsaccounts | 21.14075 |

```
# Finetuning the Decision Tree model with the high impacting features
dt_gen_model_ft  <-
   rpart(Risk~ creditamount + Duration + checkingaccount + Purpose + savingsaccounts,
         data=train_data)
dt_gen_predict_ft <- predict(dt_gen_model_ft, test_data)
dt_gen_predict_ft <- factor(ifelse(dt_gen_predict_ft[,"bad"] >= .50, "bad", "good"))
dt_gen_cf_ft<- confusionMatrix( dt_gen_predict_ft, test_data$Risk)
dt_gen_cf_ft
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   12   12
##       good   9   16
##
##                Accuracy : 0.5714
##                  95% CI : (0.4221, 0.7118)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.5600
##
##                   Kappa : 0.1404
##
##  Mcnemar's Test P-Value : 0.6625
##
##             Sensitivity : 0.5714
##             Specificity : 0.5714
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.6400
##              Prevalence : 0.4286
##          Detection Rate : 0.2449
##    Detection Prevalence : 0.4898
##       Balanced Accuracy : 0.5714
##
##        'Positive' Class : bad
##
```

Comparing the outcomes of both the vanila nad the finetune version of Decisition tree model we see that while th sensitivity is same in both the models the specificity is higher for the vanila version. So we will short list the vanila version of Decision Tree model for comparing with the other models

```
df_gen_models <-
   df_gen_models  %>%
      add_row(Model = "CART",
              Accuracy = dt_gen_cf$overall['Accuracy'] ,
              Sensitivity = dt_gen_cf$byClass['Sensitivity'],
              Specificity =  dt_gen_cf$byClass['Specificity'] )
```

### 4.3.3) Random Forest Model

```
# Bulding the Random Forest Model
rf_gen_model  <- randomForest(Risk~., data=train_data)
rf_gen_predict <- predict(rf_gen_model, test_data)
rf_gen_cf<- confusionMatrix( rf_gen_predict, test_data$Risk)
rf_gen_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   10   11
##       good  11   17
##
##                Accuracy : 0.551
##                  95% CI : (0.4023, 0.6933)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.6693
##
##                   Kappa : 0.0833
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.4762
##             Specificity : 0.6071
##          Pos Pred Value : 0.4762
##          Neg Pred Value : 0.6071
##              Prevalence : 0.4286
##          Detection Rate : 0.2041
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.5417
##
##        'Positive' Class : bad
##
```

```
# Finding out the list of variables that made significant impact in the outcome of the Random Forest mo
rf_gen_model_vi <- as.data.frame(varImp(rf_gen_model))
rf_gen_model_vi <- data.frame(feature   = rownames(rf_gen_model_vi),
                              impact = rf_gen_model_vi$Overall)
rf_gen_model_vi[order(rf_gen_model_vi$impact,decreasing = T),][1:5,] %>% knitr::kable()
```

| | feature | impact |
|---|---|---|
| 7 | creditamount | 56.40122 |
| 1 | Age | 40.52444 |
| 8 | Duration | 38.48705 |
| 9 | Purpose | 25.26020 |
| 3 | Job | 13.06178 |

```r
# Finetuning the Random Forest model with the high impacting features
rf_gen_model_ft  <- rpart(Risk~ creditamount + Age + Duration +  Purpose + Job, data=train_data)
rf_gen_predict_ft <- predict(rf_gen_model_ft, test_data)
rf_gen_predict_ft <- factor(ifelse(rf_gen_predict_ft[,"bad"] >= .50, "bad", "good"))
rf_gen_cf_ft<- confusionMatrix( rf_gen_predict_ft, test_data$Risk)
rf_gen_cf_ft
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad    9    9
##       good  12   19
##
##                Accuracy : 0.5714
##                  95% CI : (0.4221, 0.7118)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.5600
##
##                   Kappa : 0.1091
##
##  Mcnemar's Test P-Value : 0.6625
##
##             Sensitivity : 0.4286
##             Specificity : 0.6786
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.6129
##              Prevalence : 0.4286
##          Detection Rate : 0.1837
##    Detection Prevalence : 0.3673
##       Balanced Accuracy : 0.5536
##
##        'Positive' Class : bad
##
```

Comparing the outcomes of both the vanila and the finetune version of Random Forest model we see that sensitivity is higher for the vanila version. So for Random Forest method also we will short list the vanila version for comparing with the other models

```r
df_gen_models <-
   df_gen_models  %>%
      add_row(Model = "Random Forest",
              Accuracy = rf_gen_cf$overall['Accuracy'] ,
              Sensitivity = rf_gen_cf$byClass['Sensitivity'],
              Specificity =  rf_gen_cf$byClass['Specificity'])
```

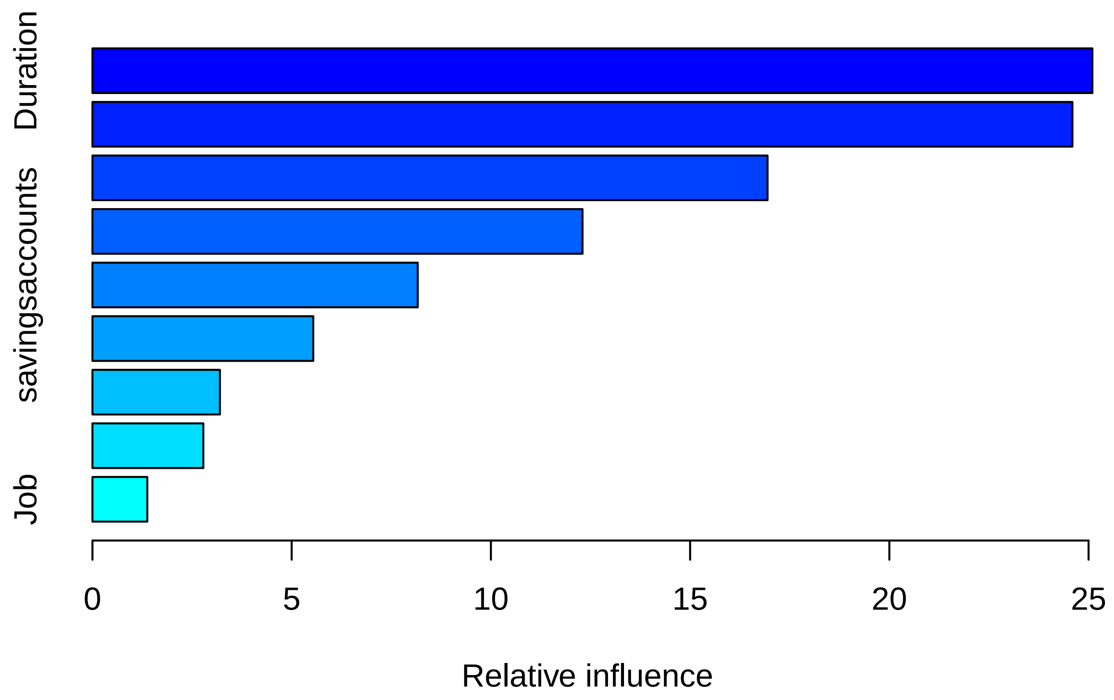
### 4.3.4) Gradient Boosting Model

```r
# Building the Greadient Bossting Model
gbm_gen_model <- gbm(Risk ~., data = train_data, distribution = "multinomial")
gbm_gen_predict <- predict(gbm_gen_model, test_data ,n.trees = 100,type = "response")
```

```r
gbm_gen_labels = colnames(gbm_gen_predict)[apply(gbm_gen_predict, 1, which.max)]
gbm_gen_result = data.frame(test_data$Risk, gbm_gen_labels)
gbm_gen_cf = confusionMatrix(test_data$Risk, as.factor(gbm_gen_labels))
gbm_gen_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   14    7
##       good  12   16
##
##                Accuracy : 0.6122
##                  95% CI : (0.4624, 0.748)
##     No Information Rate : 0.5306
##     P-Value [Acc > NIR] : 0.1582
##
##                   Kappa : 0.2312
##
##  Mcnemar's Test P-Value : 0.3588
##
##             Sensitivity : 0.5385
##             Specificity : 0.6957
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5714
##              Prevalence : 0.5306
##          Detection Rate : 0.2857
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.6171
##
##        'Positive' Class : bad
##
```

```r
# Finding out the high impacting features of the Gradient Boosting model
summary(gbm_gen_model)
```



```
##                               var   rel.inf
## Duration                 Duration 25.095304
## creditamount         creditamount 24.595256
## Age                           Age 16.943823
## Purpose                   Purpose 12.299956
## savingsaccounts savingsaccounts  8.164221
## checkingaccount checkingaccount  5.541961
## Housing                   Housing  3.199691
## Sex                           Sex  2.783144
## Job                           Job  1.376643
```

```
# Finetuning the Gradient boosting model by the TOP 5 high impacting features
gbm_gen_model_ft <- gbm(Risk ~ creditamount + Duration   + Age + Purpose + savingsaccounts,
                        data = train_data, distribution = "multinomial")
gbm_gen_predict_ft <- predict(gbm_gen_model_ft, test_data, n.trees = 100,type = "response")
gbm_gen_labels_ft = colnames(gbm_gen_predict)[apply(gbm_gen_predict_ft, 1, which.max)]
gbm_gen_result_ft = data.frame(test_data$Risk, gbm_gen_labels_ft)
gbm_gen_cf_ft = confusionMatrix(test_data$Risk, as.factor(gbm_gen_labels_ft))
gbm_gen_cf_ft
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   13    8
##       good  10   18
##
##                Accuracy : 0.6327
##                  95% CI : (0.4829, 0.7658)
##     No Information Rate : 0.5306
##     P-Value [Acc > NIR] : 0.09826
##
##                   Kappa : 0.2588
##
##  Mcnemar's Test P-Value : 0.81366
##
##             Sensitivity : 0.5652
##             Specificity : 0.6923
##          Pos Pred Value : 0.6190
##          Neg Pred Value : 0.6429
##              Prevalence : 0.4694
##          Detection Rate : 0.2653
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.6288
##
##        'Positive' Class : bad
##
```

In Gradient boosting model also we see that Sensitivity of the vanila version is higher than the finetuned version. Hence vanila version will be short listed for this model for comparing with the other models

```
df_gen_models <-
    df_gen_models  %>%
        add_row(Model = "Gradient Boosting",
                Accuracy = gbm_gen_cf$overall['Accuracy'] ,
                Sensitivity = gbm_gen_cf$byClass['Sensitivity'],
                Specificity =  gbm_gen_cf$byClass['Specificity'])
```

### 4.3.5) KNN Model

```
# Dataframe to store outcomes of KNN model for various values of K
df_knn_gen_models <- data.frame(matrix(vector(),ncol=5))
colnames(df_knn_gen_models) <- c("Model", "K", "Accuracy", "Sensitivity", "Specificity" )

# Function to evaluate and store the results of KNN model for different "K" values
knnfun <- function(a) {
              knn_gen_model  <- knn3(Risk~., data=train_data, k = a)
              knn_gen_predict <- predict(knn_gen_model, test_data)
              knn_predict_val <- factor(ifelse(knn_gen_predict[,"bad"] >= .50, "bad", "good"))
              knn_gen_cf<- confusionMatrix( knn_predict_val, test_data$Risk)
              df_knn_gen_models <<-
                  df_knn_gen_models  %>%
                      add_row(Model = "KNN",
                              K = a,
                              Accuracy = knn_gen_cf$overall['Accuracy'] ,
                              Sensitivity = knn_gen_cf$byClass['Sensitivity'],
                              Specificity =  knn_gen_cf$byClass['Specificity'])
```

```
          }

knn_output <- sapply(seq(2,9,1), knnfun)

df_knn_gen_models[order(df_knn_gen_models$Sensitivity,
                        df_knn_gen_models$Specificity,decreasing = T),] %>%
                        dplyr::select(-Model)  %>% knitr::kable()
```

|   | K | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| 1 | 2 | 0.5306122 | 0.9047619 | 0.2500000 |
| 3 | 4 | 0.6734694 | 0.8571429 | 0.5357143 |
| 5 | 6 | 0.5510204 | 0.8095238 | 0.3571429 |
| 4 | 5 | 0.7346939 | 0.7142857 | 0.7500000 |
| 7 | 8 | 0.5510204 | 0.6666667 | 0.4642857 |
| 2 | 3 | 0.6326531 | 0.6190476 | 0.6428571 |
| 6 | 7 | 0.6326531 | 0.6190476 | 0.6428571 |
| 8 | 9 | 0.6530612 | 0.5714286 | 0.7142857 |

For K=2 4 & 6 though the Sensitivity is high we see that Specifisity is very less which means that the bank will be losing considerable potential customer and there by business. On the other hand for K = 5 both Sensitivity and Specificity are significantly good. So from the KNN model we will choose the outcome of K = 5 for comparing with other models

```
df_gen_models <-  df_knn_gen_models %>% filter(K==5) %>%
                  dplyr::select(-K) %>% rbind(df_gen_models, .)
```

#### 4.3.6) Linear Discriminant Analysis (LDA) Model

```
# Building the LDA model
lda_gen_model  <- lda(Risk~., data=train_data)
lda_gen_predict <- predict(lda_gen_model, test_data)$class
lda_gen_cf<- confusionMatrix( lda_gen_predict, test_data$Risk)
lda_gen_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad    7   12
##       good  14   16
##
##                Accuracy : 0.4694
##                  95% CI : (0.3253, 0.6173)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.9431
##
##                   Kappa : -0.0964
##
##  Mcnemar's Test P-Value : 0.8445
```

```
##
##             Sensitivity : 0.3333
##             Specificity : 0.5714
##          Pos Pred Value : 0.3684
##          Neg Pred Value : 0.5333
##              Prevalence : 0.4286
##          Detection Rate : 0.1429
##    Detection Prevalence : 0.3878
##       Balanced Accuracy : 0.4524
##
##        'Positive' Class : bad
##
```

```
df_gen_models <-
   df_gen_models  %>%
      add_row(Model = "LDA",
              Accuracy = lda_gen_cf$overall['Accuracy'] ,
              Sensitivity = lda_gen_cf$byClass['Sensitivity'],
              Specificity =  lda_gen_cf$byClass['Specificity'] )
```

### 4.3.7) Quadriatic Discriminant Analysis (QDA) Model

```
# Building the QDA model
qda_gen_model  <- qda(Risk~., data=train_data)
qda_gen_predict <- predict(qda_gen_model, test_data)$class
qda_gen_cf<- confusionMatrix( qda_gen_predict, test_data$Risk)
qda_gen_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   11   15
##       good  10   13
##
##                Accuracy : 0.4898
##                  95% CI : (0.3442, 0.6366)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.9025
##
##                   Kappa : -0.0116
##
##  Mcnemar's Test P-Value : 0.4237
##
##             Sensitivity : 0.5238
##             Specificity : 0.4643
##          Pos Pred Value : 0.4231
##          Neg Pred Value : 0.5652
##              Prevalence : 0.4286
##          Detection Rate : 0.2245
##    Detection Prevalence : 0.5306
##       Balanced Accuracy : 0.4940
##
##        'Positive' Class : bad
##
```

```
df_gen_models <-
    df_gen_models  %>%
        add_row(Model = "QDA",
                Accuracy = qda_gen_cf$overall['Accuracy'] ,
                Sensitivity = qda_gen_cf$byClass['Sensitivity'],
                Specificity =  qda_gen_cf$byClass['Specificity'] )
```

### 4.3.8) Support Vector Machines (SVM) Model

```
# Building the SVM model
svm_gen_model  <- svm(Risk~., data=train_data)
svm_gen_predict <- predict(svm_gen_model, test_data)
svm_gen_cf<- confusionMatrix( svm_gen_predict, test_data$Risk)
svm_gen_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad    8    7
##       good  13   21
##
##                Accuracy : 0.5918
##                  95% CI : (0.4421, 0.73)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.4454
##
##                   Kappa : 0.1358
##
##  Mcnemar's Test P-Value : 0.2636
##
##             Sensitivity : 0.3810
##             Specificity : 0.7500
##          Pos Pred Value : 0.5333
##          Neg Pred Value : 0.6176
##              Prevalence : 0.4286
##          Detection Rate : 0.1633
##    Detection Prevalence : 0.3061
##       Balanced Accuracy : 0.5655
##
##        'Positive' Class : bad
##
```

```
df_gen_models <-
   df_gen_models  %>%
       add_row(Model = "SVM",
                Accuracy = svm_gen_cf$overall['Accuracy'] ,
                Sensitivity = svm_gen_cf$byClass['Sensitivity'],
                Specificity =  svm_gen_cf$byClass['Specificity'] )

# Summary of the Models we have build so far.
df_gen_models[order(df_gen_models$Sensitivity,
                    df_gen_models$Specificity,decreasing = T),] %>% knitr::kable()
```

| | Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| 5 | KNN | 0.7346939 | 0.7142857 | 0.7500000 |
| 1 | Logistic Regression | 0.5510204 | 0.6666667 | 0.4642857 |
| 2 | CART | 0.5918367 | 0.5714286 | 0.6071429 |
| 4 | Gradient Boosting | 0.6122449 | 0.5384615 | 0.6956522 |
| 7 | QDA | 0.4897959 | 0.5238095 | 0.4642857 |
| 3 | Random Forest | 0.5510204 | 0.4761905 | 0.6071429 |
| 8 | SVM | 0.5918367 | 0.3809524 | 0.7500000 |
| 6 | LDA | 0.4693878 | 0.3333333 | 0.5714286 |

Of the 8 models we have build KNN seems to be best option as of now

The models that we have build so far has been build with one fold model . We can now try cross validation method which employes multi fold approach to check whether it produces better output than the previously build general models

## 4.4) Model building based on cross validation ("train") method

Similar to general model building exercise in section 4.3 , here in cross validation method also we shall finetune those models for which important variables that made significant impact on the model building can be identified and finetune the models based based on them and then select the one which provides best result for the sensitivity/specificity pair

```
# Create data frame to store the output of different "train" models
df_train_models <- data.frame(matrix(vector(),ncol=4))
colnames(df_train_models) <- c("Model", "Accuracy", "Sensitivity", "Specificity")

# Control parameter for the train model
ctrl <- trainControl(
method="repeatedcv",
repeats=5,
summaryFunction=twoClassSummary,
classProbs=TRUE,
allowParallel = TRUE)
```

### 4.4.1) Logistic Regression

```
# Building the Logistic Regression model
glm_train_model <- train(Risk ~ ., method = "glm" ,  family = "binomial",
                         data = train_data, trControl = ctrl)
glm_train_predict <- predict(glm_train_model, test_data)
```

```r
glm_train_cf <- caret::confusionMatrix(glm_train_predict, test_data$Risk)

# List of Top 5 variables that made significant impact in Logistic Regression model
glm_train_model_vi <- varImp(glm_train_model)
glm_train_model_vi <- data.frame(feature   = rownames(glm_train_model_vi$importance),
                                 impact = glm_train_model_vi$importance$Overall)
glm_train_model_vi[order(glm_train_model_vi$impact,decreasing = T),][1:5,] %>% knitr::kable()
```

| | feature | impact |
|---|---|---|
| 12 | Duration | 100.00000 |
| 10 | checkingaccountrich | 58.60060 |
| 8 | savingsaccountsrich | 48.87518 |
| 2 | Sexmale | 44.51183 |
| 15 | Purposeeducation | 38.69963 |

```r
# Finetuning the Logistic Regression model by the high impacting features
glm_train_model_ft <-
   train(Risk ~ Duration + checkingaccount + savingsaccounts + Sex + Purpose,
         method = "glm" ,  family = "binomial", data = train_data, trControl = ctrl)
glm_train_predict_ft <- predict(glm_train_model_ft, test_data)
glm_train_cf_ft <- caret::confusionMatrix(glm_train_predict_ft, test_data$Risk)

glm_train_cf_ft
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad    8   11
##       good  13   17
##
##                Accuracy : 0.5102
##                  95% CI : (0.3634, 0.6558)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.8438
##
##                   Kappa : -0.012
##
##  Mcnemar's Test P-Value : 0.8383
##
##             Sensitivity : 0.3810
##             Specificity : 0.6071
##          Pos Pred Value : 0.4211
##          Neg Pred Value : 0.5667
##              Prevalence : 0.4286
##          Detection Rate : 0.1633
##    Detection Prevalence : 0.3878
##       Balanced Accuracy : 0.4940
##
##        'Positive' Class : bad
##
```

Between the vanila (i.e without any feature finetuning) & the finetuned model the sensitivity & Specificity of the finetune model is higher. So we will short list that for comparing with other models

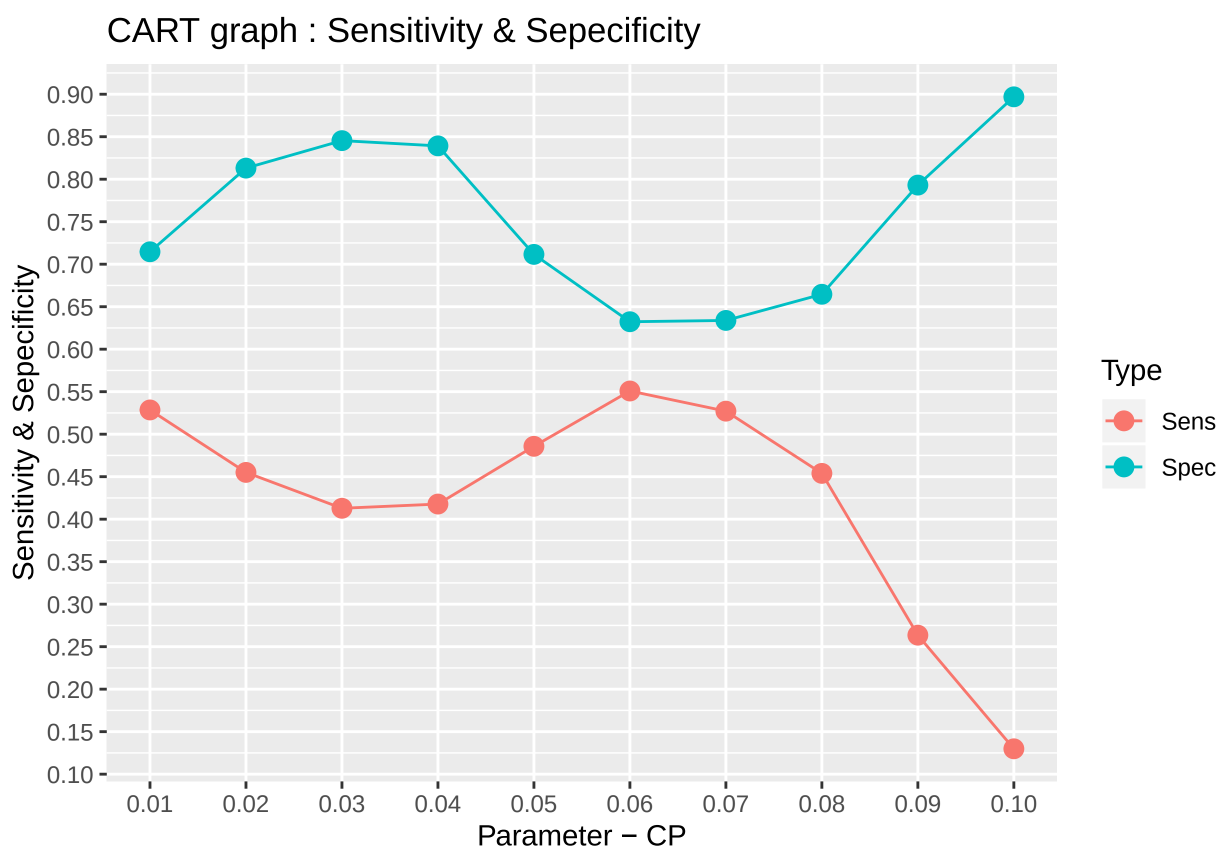
```
# Store the result to the train model dataframe
df_train_models <-
   df_train_models %>%
       add_row(Model = "Logistic Regression_Train",
               Accuracy = glm_train_cf_ft$overall['Accuracy'] ,
               Sensitivity = glm_train_cf_ft$byClass['Sensitivity'],
               Specificity =  glm_train_cf_ft$byClass['Specificity'])
```

### 4.4.2) Decision Tree

```
# Building the Decision Tree
dt_train_model <- train(Risk ~ ., method = "rpart" , data = train_data,
                        metric="ROC",tuneGrid = expand.grid(.cp = seq(0.01,0.1,.01)),
                        trControl=ctrl)

# Result of the Decision tree model for different CP values
dt_train_model$results %>% dplyr::select(cp, Sens, Spec) %>% knitr::kable()
```

| cp | Sens | Spec |
|---|---|---|
| 0.01 | 0.5285263 | 0.7146154 |
| 0.02 | 0.4550526 | 0.8130769 |
| 0.03 | 0.4128421 | 0.8453846 |
| 0.04 | 0.4177895 | 0.8392308 |
| 0.05 | 0.4857368 | 0.7115385 |
| 0.06 | 0.5508421 | 0.6323077 |
| 0.07 | 0.5271053 | 0.6338462 |
| 0.08 | 0.4538947 | 0.6646154 |
| 0.09 | 0.2635263 | 0.7930769 |
| 0.10 | 0.1298421 | 0.8969231 |

```
# Transforming the Decision tree train model output to required format for plotting the graph
dt_train_model_tmp <- as.data.frame(dt_train_model$results) %>% dplyr::select(cp,Sens, Spec)
dt_train_model_tmp <- gather(dt_train_model_tmp, Type, Val, Sens:Spec)
dt_train_model_tmp <- dt_train_model_tmp [order(dt_train_model_tmp$cp ), ]

# Plotting the graph for Decision Tree model
ggplot(dt_train_model_tmp, aes(cp, Val, color = Type)) +
   geom_line() +
   geom_point(size = 3) +
   scale_y_continuous(breaks = seq(0.10, 1.0, by = .05)) +
   scale_x_continuous(breaks = seq(.01, 1.0, by = .01)) +
   theme(panel.grid.minor.x = element_blank()) +
   labs(y = "Sensitivity & Sepecificity", x = "Parameter - CP",
        title = "CART graph : Sensitivity & Sepecificity ")
```



From the above graph plot we can see that for cp parameter of .01 we get optimum sensitivity/specificity pair values

```r
# Data Record for the max Sensitivity value
dt_train_model$results[dt_train_model$results$Sens == max(dt_train_model$results$Sens),] %>%
          dplyr::select (cp, Sens, Spec)  %>% knitr::kable()
```

|   | cp | Sens | Spec |
|---|---|---|---|
| 6 | 0.06 | 0.5508421 | 0.6323077 |

```r
# Listing the TOP 5 variables that made significant impact Decision tree model
dt_train_model_vi <- varImp(dt_train_model)
dt_train_model_vi <- data.frame(feature   = rownames(dt_train_model_vi$importance),
                                impact = dt_train_model_vi$importance$Overall)

dt_train_model_vi[order(dt_train_model_vi$impact,decreasing = T),][1:5,] %>% knitr::kable()
```

|   | feature | impact |
|---|---|---|
| 4 | creditamount | 100.00000 |
| 5 | Duration | 71.43269 |
| 3 | checkingaccountrich | 39.90781 |
| 1 | Age | 31.35727 |
| 12 | savingsaccountsmoderate | 28.32043 |

```r
# Finetuning the Decision Tree model by the high impacting features
dt_train_model_ft <-
  train(Risk ~ creditamount + checkingaccount + savingsaccounts + Age + Purpose ,
        method = "rpart" , data = train_data,  metric="ROC",
        tuneGrid = expand.grid(.cp = seq(0.01,0.1,.01)), trControl=ctrl)

# Below is the OUtput for the finetuned version for differnt cp values
dt_train_model_ft$results %>% dplyr::select(cp, Sens, Spec) %>% knitr::kable()
```

| cp | Sens | Spec |
|---|---|---|
| 0.01 | 0.4505263 | 0.6761538 |
| 0.02 | 0.1524737 | 0.9007692 |
| 0.03 | 0.0867895 | 0.9615385 |
| 0.04 | 0.0836316 | 0.9676923 |
| 0.05 | 0.0825789 | 0.9700000 |
| 0.06 | 0.0602632 | 0.9753846 |
| 0.07 | 0.0176316 | 0.9807692 |
| 0.08 | 0.0084211 | 0.9876923 |
| 0.09 | 0.0000000 | 1.0000000 |
| 0.10 | 0.0000000 | 1.0000000 |

```r
# Data record for the max Sensitivity value from the finetune model
maxVal <- max(dt_train_model_ft$results$Sens)
dt_train_model_ft$results[dt_train_model_ft$results$Sens == maxVal,] %>%
  dplyr::select(cp, Sens, Spec)  %>% knitr::kable()
```

| cp | Sens | Spec |
|---|---|---|
| 0.01 | 0.4505263 | 0.6761538 |

Between the vanila and the finetune version of Decisition tree model we see that the sensitivity is higher for the vanila model same in both the models the specificity is higher for the vanila version. So we will short list the vanila version

By default the train model will pickup the data record with maximum ROC value. Since our requirement is to have the one with optimum Sensitivity + Specificity value pair we shall rebuild the model with the Cp value of the record with optimum sensitivity value

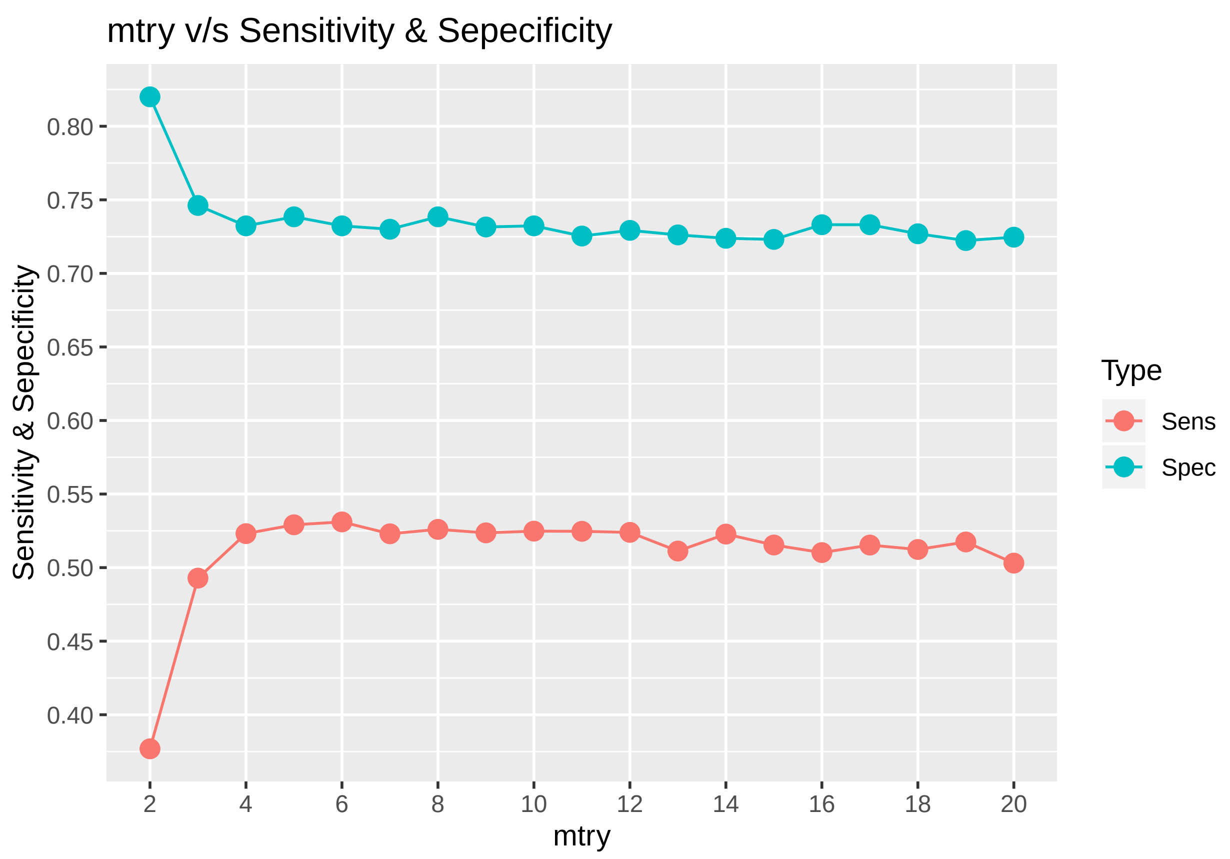
```r
# cp parameter value for the data record with maximum sensitivity
cpVal =
  dt_train_model$results[dt_train_model$results$Sens == max(dt_train_model$results$Sens),]$cp

rm(dt_train_model)
dt_train_model <- train(Risk ~ ., method = "rpart" , data = train_data,
                        metric="ROC",tuneGrid = expand.grid(.cp = cpVal ), trControl=ctrl)
dt_train_predict<- predict(dt_train_model, test_data)
dt_train_cf<- caret::confusionMatrix(dt_train_predict, test_data$Risk)
dt_train_model$results  %>% knitr::kable()
```

| cp | ROC | Sens | Spec | ROCSD | SensSD | SpecSD |
|---|---|---|---|---|---|---|
| 0.06 | 0.5945901 | 0.4951579 | 0.6638462 | 0.0671355 | 0.2451937 | 0.1872667 |

```
df_train_models <-
   df_train_models %>%
       add_row( Model = "DT_Train",
                Accuracy = dt_train_cf$overall['Accuracy'] ,
                Sensitivity = dt_train_cf$byClass['Sensitivity'],
                Specificity =  dt_train_cf$byClass['Specificity'])
```

#### 4.4.3) Random Forest

```
# Building the RF model for various values of mtry - the tuneGrid parameter
rf_train_model <- train(Risk ~ ., method = "rf" , data = train_data,
                        metric="ROC", tuneGrid = expand.grid(.mtry = seq(2,20,1)),
                        trControl=ctrl)

# Transforming the data frame to the structure to build the graph
rf_train_model_tmp <- as.data.frame(rf_train_model$results) %>% dplyr::select(mtry,Sens, Spec)
rf_train_model_tmp <- gather(rf_train_model_tmp, Type, Val, Sens:Spec)
rf_train_model_tmp <- rf_train_model_tmp [order(rf_train_model_tmp$mtry ), ]

# Building the graph
ggplot(rf_train_model_tmp, aes(mtry, Val, color = Type)) +
   geom_line() +
   geom_point(size = 3) +
   scale_y_continuous(breaks = seq(0.1, 1.0, by = .05)) +
   scale_x_continuous(breaks = seq(2, 20, by = 2)) +
   theme(panel.grid.minor.x = element_blank()) +
   labs(y = "Sensitivity & Sepecificity", title = "mtry v/s Sensitivity & Sepecificity ")
```



```
# Data set of the record with maximum Sensitivity from the vanila model
maxVal = max(rf_train_model$results$Sens)
rf_train_model$results[rf_train_model$results$Sens == maxVal ,]  %>% knitr::kable()
```

| | mtry | ROC | Sens | Spec | ROCSD | SensSD | SpecSD |
|---|---|---|---|---|---|---|---|
| 5 | 6 | 0.6815162 | 0.5310526 | 0.7323077 | 0.0759968 | 0.1255132 | 0.0892616 |

```
# The TOP 5 features that had significant impact in the model outcome
rf_train_model_vi <- varImp(rf_train_model)
rf_train_model_vi <- data.frame(feature   = rownames(rf_train_model_vi$importance),
                                impact = rf_train_model_vi$importance$Overall)
rf_train_model_vi[order(rf_train_model_vi$impact,decreasing = T),][1:5,] %>% knitr::kable()
```

| | feature | impact |
|---|---|---|
| 11 | creditamount | 100.00000 |
| 1 | Age | 66.73142 |
| 12 | Duration | 64.66187 |
| 3 | Job | 20.75695 |
| 2 | Sexmale | 11.15292 |

```
# Building the finetuned version of the Random forest method
rf_train_model_ft <- train(Risk ~ creditamount + Age + Duration + Job + Sex,
                           method = "rf" , data = train_data, metric="ROC",
                           tuneGrid = expand.grid(.mtry = seq(2,20,1)), trControl=ctrl)

# Data set of the record with maximum Sensitivity from the finetuned model
maxVal = max(rf_train_model_ft$results$Sens)
rf_train_model_ft$results[rf_train_model_ft$results$Sens == maxVal,] %>% knitr::kable()
```

| | mtry | ROC | Sens | Spec | ROCSD | SensSD | SpecSD |
|---|---|---|---|---|---|---|---|
| 18 | 19 | 0.64592 | 0.5104737 | 0.6915385 | 0.0770291 | 0.1142505 | 0.0835056 |

Comparing the vanila & finetuned version , Sensitivity + Specificity pair is good for finetuned version. Similar to the decision tree model case We shall re generate the Random Forest model also with the mtry value of finetuned method which has max Sensitivity value

```
maxVal = max(rf_train_model_ft$results$Sens)
mtryVal = rf_train_model_ft$results[rf_train_model_ft$results$Sens == maxVal,]$mtry
rm(rf_train_model_ft)
rf_train_model_ft <- train(Risk ~ creditamount + Age + Duration + Job + Sex,
                           method = "rf" , data = train_data, metric="ROC",
                           tuneGrid = expand.grid(.mtry = mtryVal), trControl=ctrl)
rf_train_predict_ft <- predict(rf_train_model_ft, test_data)
rf_train_cf_ft <- caret::confusionMatrix(rf_train_predict_ft, test_data$Risk)
df_train_models <-
   df_train_models %>%
       add_row(Model = "Random Forest_Train",
               Accuracy = rf_train_cf_ft$overall['Accuracy'] ,
               Sensitivity = rf_train_cf_ft$byClass['Sensitivity'],
               Specificity =  rf_train_cf_ft$byClass['Specificity'])
```

### 4.4.4) Gradient Boosting

NOTE : This model takes lot of time for execution. For my machine configuration of 6 GB with i3 processor it took around 25 min for each of the execution. If this model is selected in the final comparision then this performance factor has to be taken into consideration before deploying it for production purpose

```
# Building the graident boosting model
xgb_train_model<- train(Risk ~ ., method = "xgbDART" , data = train_data)

xgb_train_predict<- predict(xgb_train_model, test_data)
xgb_train_cf<- confusionMatrix(xgb_train_predict, test_data$Risk)

# Listing  the TOP 5 variables that made significant impact in the Gradient boosting model
xgb_train_model_vi <- varImp(xgb_train_model)
xgb_train_model_vi <- data.frame(feature   = rownames(xgb_train_model_vi$importance),
                                 impact = xgb_train_model_vi$importance$Overall)
xgb_train_model_vi[order(xgb_train_model_vi$impact,decreasing = T),][1:5,] %>% knitr::kable()
```

| feature | impact |
|---|---|
| creditamount | 100.00000 |
| Duration | 80.80198 |
| Age | 31.24505 |
| savingsaccountsrich | 18.08257 |
| checkingaccountrich | 17.05311 |

```r
# Bilding the graident boosting model based on the high impacting features
xgb_train_model_ft <-
   train(Risk ~ Duration + checkingaccount + savingsaccounts + Sex + Purpose,
         method = "xgbDART" , data = train_data)
xgb_train_predict_ft <- predict(xgb_train_model_ft, test_data)
xgb_train_cf_ft <- confusionMatrix(xgb_train_predict_ft, test_data$Risk)
```

Comparing the vanilla and feature selection model of Gradient boosting we select the vanila model as the Sensitivity/Specificity value pair is better than the feature selection version

```r
# Adding the Gradient boosting result the to train model data frame
df_train_models <<-
    df_train_models %>%
        add_row(Model = "Gradient Boosting_Train",
                Accuracy = xgb_train_cf$overall['Accuracy'] ,
                Sensitivity = xgb_train_cf$byClass['Sensitivity'],
                Specificity =  xgb_train_cf$byClass['Specificity'])
```

### 4.4.5) KNN Model

```r
# Building the KNN model
knn_train_model<- train(Risk ~ ., method = "knn" , data = train_data, metric="ROC",
                        tuneGrid = expand.grid(k = seq(1,9,1)),trControl = ctrl )

# knn_train_model$results %>% dplyr::select(k, Sens, Spec) %>% knitr::kable()
knn_train_model
```

```
## k-Nearest Neighbors
##
## 453 samples
##   9 predictor
##   2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 408, 408, 407, 408, 408, 408, ...
## Resampling results across tuning parameters:
##
##   k  ROC        Sens       Spec
##   1  0.5344281  0.4498947  0.6184615
##   2  0.5494221  0.4490000  0.5807692
##   3  0.5593016  0.4571579  0.6476923
##   4  0.5460536  0.4420000  0.6369231
##   5  0.5485081  0.4092632  0.6676923
##   6  0.5561255  0.4041053  0.6507692
##   7  0.5632834  0.3781053  0.6792308
##   8  0.5610486  0.3885263  0.6661538
##   9  0.5587105  0.3811579  0.7053846
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

Above details of the KNN model shows that sensitivity /specificity pair is significant when k is 3. Though technically K=2 has slightly higher senisitivity than k=3 , Specificity is significantly higher for k=3 over K=2. Hence K=3 has been selected. As explained above the Model by default is considering the value which has higher ROC and hence shows different value of K. As our requirement is to have better Sensitivity followed by specificity we will choose the value K=3 and rebuild the model

```
rm(knn_train_model)
knn_train_model <- train(Risk ~ ., method = "knn" , data = train_data, metric="ROC",
                         tuneGrid = expand.grid(k = 3),trControl = ctrl )
knn_train_predict<- predict(knn_train_model, test_data)
knn_train_cf <- caret::confusionMatrix(knn_train_predict, test_data$Risk)
```

Above shows the sensitivity and specificity details of the KNN method. This shows improvement over other 'train' models though it is far below than the one we tried with general commands

```
# Adding the KNN result to the train model data frame
df_train_models <-
    df_train_models %>%
        add_row(Model = "KNN_Train",
                Accuracy = knn_train_cf$overall['Accuracy'] ,
                Sensitivity = knn_train_cf$byClass['Sensitivity'],
                Specificity =  knn_train_cf$byClass['Specificity'])
```

### 4.4.6) Linear Discriminant Analysis (LDA)

```
# Building the LDA model
lda_train_model <- train(Risk ~ ., method = "lda" , data = train_data,   trControl=ctrl)
lda_train_predict<- predict(lda_train_model, test_data)
lda_train_cf<- caret::confusionMatrix(lda_train_predict, test_data$Risk)
lda_train_cf
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction bad good
##       bad    7   12
##       good  14   16
##
##                Accuracy : 0.4694
##                  95% CI : (0.3253, 0.6173)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.9431
```

```
##
##                   Kappa : -0.0964
##
##  Mcnemar's Test P-Value : 0.8445
##
##             Sensitivity : 0.3333
##             Specificity : 0.5714
##          Pos Pred Value : 0.3684
##          Neg Pred Value : 0.5333
##              Prevalence : 0.4286
##          Detection Rate : 0.1429
##    Detection Prevalence : 0.3878
##       Balanced Accuracy : 0.4524
##
##        'Positive' Class : bad
##
```

Above shows the sensitivity and specificity details of the LDA method. We see that this is far below that what we have observed from other models

```
# Adding the output to train model dataframe
df_train_models <-
    df_train_models %>%
        add_row(Model = "LDA_Train",
                Accuracy = lda_train_cf$overall['Accuracy'] ,
                Sensitivity = lda_train_cf$byClass['Sensitivity'],
                Specificity =  lda_train_cf$byClass['Specificity'])
```

### 4.4.7) Quadriatic Discriminant Analysis (QDA)

```
# Building the QDA model
qda_train_model<-
   train(Risk ~ ., method = "qda" , data = train_data,  metric="ROC", trControl=ctrl)
qda_train_predict<- predict(qda_train_model, test_data)
qda_train_cf<- caret::confusionMatrix(qda_train_predict, test_data$Risk)
qda_train_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   11   15
##       good  10   13
##
##                Accuracy : 0.4898
##                  95% CI : (0.3442, 0.6366)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.9025
##
##                   Kappa : -0.0116
##
```

```
##  Mcnemar's Test P-Value : 0.4237
##
##             Sensitivity : 0.5238
##             Specificity : 0.4643
##          Pos Pred Value : 0.4231
##          Neg Pred Value : 0.5652
##              Prevalence : 0.4286
##          Detection Rate : 0.2245
##    Detection Prevalence : 0.5306
##       Balanced Accuracy : 0.4940
##
##        'Positive' Class : bad
##
```

QDA Models' result is below the best one we have got so far

```
# Adding the QDA result to train model dataframe
df_train_models <-
    df_train_models %>%
        add_row(Model = "QDA_Train",
                Accuracy = qda_train_cf$overall['Accuracy'] ,
                Sensitivity = qda_train_cf$byClass['Sensitivity'],
                Specificity =  qda_train_cf$byClass['Specificity'])
```

### 4.4.8) Support Vector Machines (SVM)

```
# Preparig the 2 finetuning parameters for the SVM method
grid <- expand.grid(sigma = c(seq(.01, 0.10, 0.01)),
     C = c(seq(0.1, 1.0, .10)))

# Building the SVM train model
svm_train_model <- train(Risk ~ ., method = "svmRadial" ,
                         data = train_data, tuneGrid = grid, trControl = ctrl)
```

As it will be nearly 100 records we are not displaying the output of the svm train model. Instead we shall check the out come with a graph

```
# Extracting only those values from the model output that are required to plot the graph
svm_train_model_tmp <- as.data.frame(svm_train_model$results) %>%
                     dplyr::select(sigma, C, Sens, Spec)
```

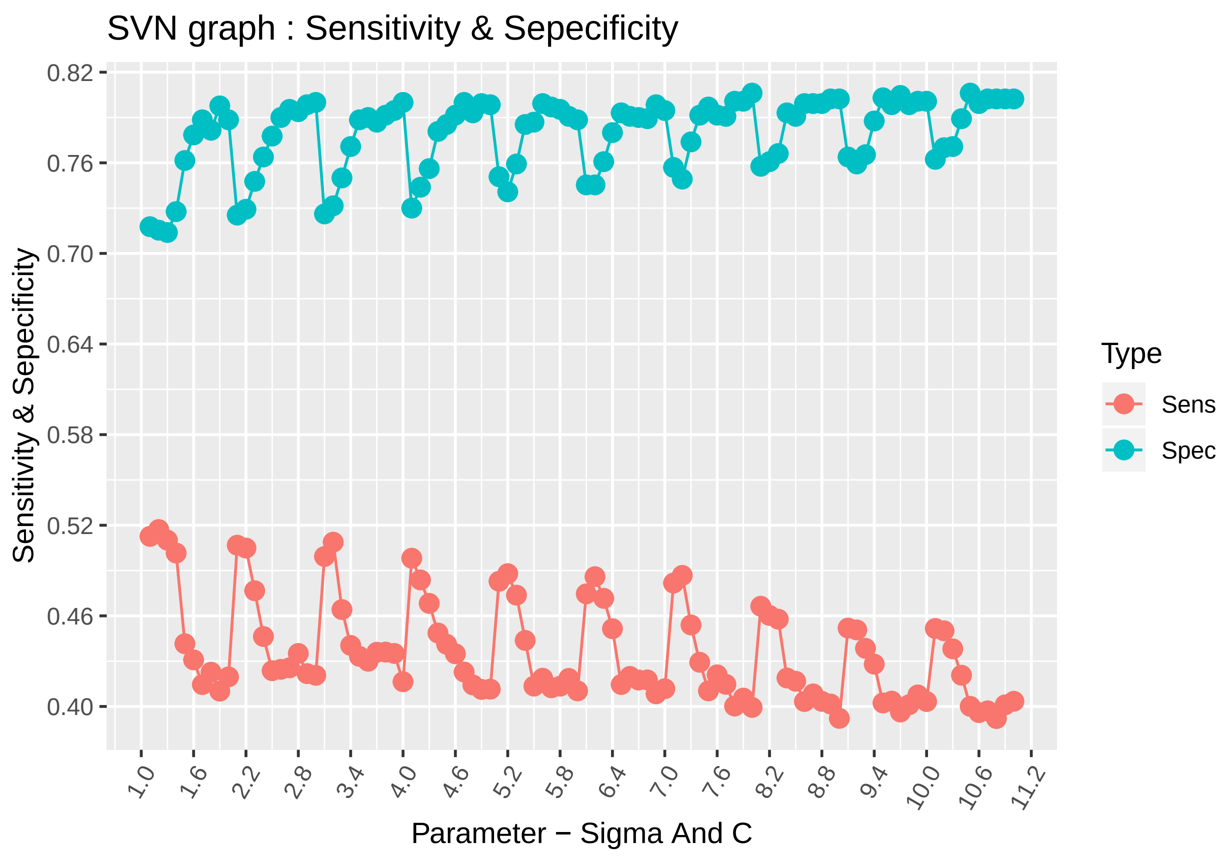
Now we shall plot the graph for SVM model. Since there are two different tuneGrid parameters of Sigma and C we shall apply the formula of ( Sigma*100 + C ) which shall then be used to plot the graph

```
svm_train_model_tmp <- svm_train_model_tmp %>%
   mutate(SigmaAndC = svm_train_model$results$sigma * 100 + svm_train_model$results$C) %>%
   dplyr::select(SigmaAndC, Sens, Spec)

#Transforming the data into another format in order to plot the graph
svm_train_model_tmp <- gather(svm_train_model_tmp, Type, Val, Sens:Spec)
```

```r
#Sorting it based on SigmaAndC value
svm_train_model_tmp <- svm_train_model_tmp[order(svm_train_model_tmp$SigmaAndC ), ]

# Displaying only top 5 out of 200 values
svm_train_model_tmp[1:5,] %>% knitr::kable()
```

| | SigmaAndC | Type | Val |
|---|---|---|---|
| 1 | 1.1 | Sens | 0.5126316 |
| 101 | 1.1 | Spec | 0.7176923 |
| 2 | 1.2 | Sens | 0.5171579 |
| 102 | 1.2 | Spec | 0.7153846 |
| 3 | 1.3 | Sens | 0.5101053 |

```r
# Plotting the graph
ggplot(svm_train_model_tmp, aes(SigmaAndC, Val, color = Type)) +
   geom_line() +
   geom_point(size = 3) +
   scale_y_continuous(breaks = seq(0.10, 1.0, by = .06)) +
   scale_x_continuous(breaks = seq(1.00, 11.5, by = .60)) +
   theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
   labs(y = "Sensitivity & Sepecificity", x = "Parameter - Sigma And C",
   title = "SVN graph : Sensitivity & Sepecificity ")
```



By default the model chooses the one with max ROC. As it is nearly 100 records we just displayed the best value which the model will select by default

```r
svm_train_model$results %>%
  filter(sigma == svm_train_model$bestTune$sigma, C == svm_train_model$bestTune$C )  %>%
    knitr::kable()
```

| sigma | C | ROC | Sens | Spec | ROCSD | SensSD | SpecSD |
|---|---|---|---|---|---|---|---|
| 0.03 | 0.9 | 0.6941822 | 0.4351053 | 0.7946154 | 0.0682126 | 0.1024705 | 0.0877781 |

But our requirement is to select the one which has better combination of Sensitivity & Specificity. So we shall display the top 5 results of maxium Sensitivity values

```r
svm_train_results <- data.frame(svm_train_model$results)
svm_train_results[ order(svm_train_results$Sens,decreasing = T), ][1:5,] %>%
  dplyr::select(sigma, C, Sens, Spec) %>% knitr::kable()
```

| | sigma | C | Sens | Spec |
|---|---|---|---|---|
| 2 | 0.01 | 0.2 | 0.5171579 | 0.7153846 |
| 1 | 0.01 | 0.1 | 0.5126316 | 0.7176923 |
| 3 | 0.01 | 0.3 | 0.5101053 | 0.7138462 |
| 22 | 0.03 | 0.2 | 0.5087895 | 0.7315385 |
| 11 | 0.02 | 0.1 | 0.5067895 | 0.7253846 |

We can see that the SVM model produce the best value when sigma = .01 and C = .3. Hence we will rebuild the model for that value

```r
# selecting the tuning parameter of the record which yielded maximum sensitivity
values <- data.frame(sigma,C)
maxVal = max(svm_train_model$results$Sens)
values <- svm_train_model$results[svm_train_model$results$Sens == maxVal,] %>%
  dplyr::select(sigma,C)


rm(svm_train_model)

# Finetuning the SVM model with parameters that yielded maximum Sensitivity value
svm_train_model <-
  train(Risk ~ ., method = "svmRadial" , data = train_data,
        tuneGrid = expand.grid(sigma= values$sigma, C = values$C), trControl = ctrl)
svm_train_predict<- predict(svm_train_model, test_data)
svm_train_cf<- caret::confusionMatrix(svm_train_predict, test_data$Risk)

# Confusion matrix Output of the SVM Train model
svm_train_cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad    9   13
##       good  12   15
##
##                Accuracy : 0.4898
##                  95% CI : (0.3442, 0.6366)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.9025
##
##                   Kappa : -0.0355
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.4286
##             Specificity : 0.5357
##          Pos Pred Value : 0.4091
##          Neg Pred Value : 0.5556
##              Prevalence : 0.4286
##          Detection Rate : 0.1837
##    Detection Prevalence : 0.4490
##       Balanced Accuracy : 0.4821
##
##        'Positive' Class : bad
##
```

Result of the SVM model is found to be well below the maximum one we have observed so far

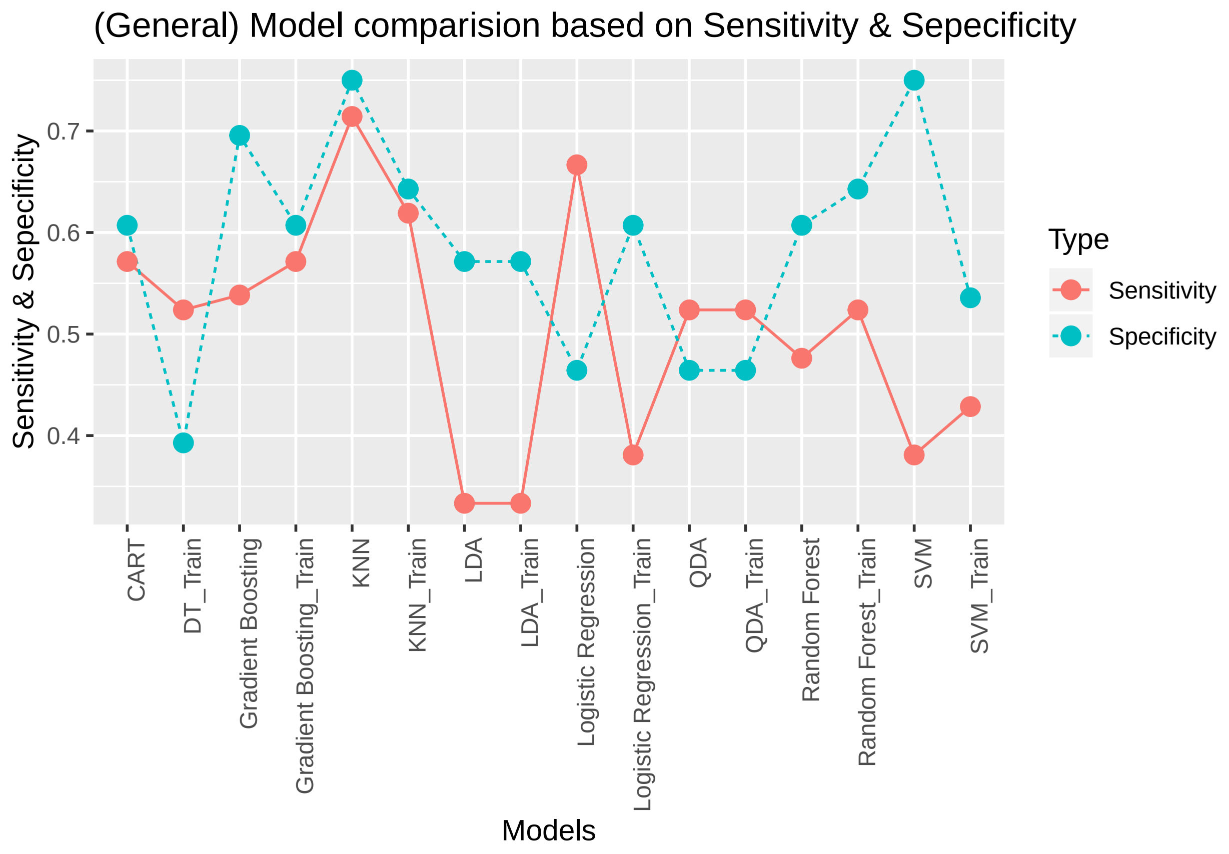
```r
# Adding the SVM result to the train model dataframe
df_train_models <-
    df_train_models %>%
        add_row(Model = "SVM_Train",
                Accuracy = svm_train_cf$overall['Accuracy'] ,
                Sensitivity = svm_train_cf$byClass['Sensitivity'],
                Specificity =  svm_train_cf$byClass['Specificity'])

# Comparing the train Models
df_train_models %>%knitr::kable()
```

| Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Logistic Regression_Train | 0.5102041 | 0.3809524 | 0.6071429 |
| DT_Train | 0.4489796 | 0.5238095 | 0.3928571 |
| Random Forest_Train | 0.5918367 | 0.5238095 | 0.6428571 |
| Gradient Boosting_Train | 0.5918367 | 0.5714286 | 0.6071429 |
| KNN_Train | 0.6326531 | 0.6190476 | 0.6428571 |
| LDA_Train | 0.4693878 | 0.3333333 | 0.5714286 |
| QDA_Train | 0.4897959 | 0.5238095 | 0.4642857 |
| SVM_Train | 0.4897959 | 0.4285714 | 0.5357143 |

## 5) Result

We shall now combine the results of models build above using both the base and cross validation train methods and then draw the plot based on Sensitivity and specificity. As mentioned in the beginning we shall then finalize the best model based on balanced combination of both sensitivity and specificity value

```r
# Combining the output from both the general & Train models
df_combined_models <- rbind(df_train_models, df_gen_models)
df_combined_models_tmp <- gather(df_combined_models, Type, Val, Sensitivity: Specificity)
df_combined_models_tmp <- df_combined_models_tmp[order(df_combined_models_tmp[,1]),]
df_combined_models_tmp$Model <-
   factor(df_combined_models_tmp$Model, levels = unique(df_combined_models_tmp$Model))

# Plotting the graph for the combined list of model
ggplot(df_combined_models_tmp, aes(Model, Val, color = Type, group = Type)) +
   geom_line(aes(linetype = Type)) +
   geom_point(size = 3) +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
   labs(y = "Sensitivity & Sepecificity", x = "Models",
        title = "(General) Model comparision based on Sensitivity & Sepecificity ")
```



```r
# Tabular display of the combined model result
df_combined_models[order(df_combined_models[,3], decreasing = T),]%>% knitr::kable()
```

|  | Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| 13 | KNN | 0.7346939 | 0.7142857 | 0.7500000 |
| 9 | Logistic Regression | 0.5510204 | 0.6666667 | 0.4642857 |
| 5 | KNN_Train | 0.6326531 | 0.6190476 | 0.6428571 |
| 4 | Gradient Boosting_Train | 0.5918367 | 0.5714286 | 0.6071429 |
| 10 | CART | 0.5918367 | 0.5714286 | 0.6071429 |
| 12 | Gradient Boosting | 0.6122449 | 0.5384615 | 0.6956522 |
| 2 | DT_Train | 0.4489796 | 0.5238095 | 0.3928571 |
| 3 | Random Forest_Train | 0.5918367 | 0.5238095 | 0.6428571 |
| 7 | QDA_Train | 0.4897959 | 0.5238095 | 0.4642857 |
| 15 | QDA | 0.4897959 | 0.5238095 | 0.4642857 |
| 11 | Random Forest | 0.5510204 | 0.4761905 | 0.6071429 |
| 8 | SVM_Train | 0.4897959 | 0.4285714 | 0.5357143 |
| 1 | Logistic Regression_Train | 0.5102041 | 0.3809524 | 0.6071429 |
| 16 | SVM | 0.5918367 | 0.3809524 | 0.7500000 |
| 6 | LDA_Train | 0.4693878 | 0.3333333 | 0.5714286 |
| 14 | LDA | 0.4693878 | 0.3333333 | 0.5714286 |

## 6) Conclusion

Based on the output graph we can conclude that the general KNN model is the one that will suit the most for the German bank as it provides both high sensitivity and specificity.

Due to system limitation in terms of processor and RAM I have selected this data set which has only 1000 records. Would have been more happier :-) should have high end systems where in I could have gone for data set with more records and also picked up few more algorithms to build the model for comparision.