## DATA EXPLORATION

### Reading the Data from excel file

```
require(XLConnect)

wb <- loadWorkbook("C:/R files/German Credit.xls") my_data <-
readWorksheet(wb,"Data",header=T)
```

### Checking the structure of the data

```
str(my_data)

## 'data.frame':    1000 obs. of  32 variables:
##  $ OBS.            : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ CHK_ACCT        : num  0 1 3 0 0 3 3 1 3 1 ...
##  $ DURATION        : num  6 48 12 42 24 36 24 36 12 30 ...
##  $ HISTORY         : num  4 2 4 2 3 2 2 2 2 4 ...
##  $ NEW_CAR         : num  0 0 0 0 1 0 0 0 0 1 ...
##  $ USED_CAR        : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ FURNITURE       : num  0 0 0 1 0 0 1 0 0 0 ...
##  $ RADIO.TV        : num  1 1 0 0 0 0 0 0 1 0 ...
##  $ EDUCATION       : num  0 0 1 0 0 1 0 0 0 0 ...
##  $ RETRAINING      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AMOUNT          : num  1169 5951 2096 7882 4870 ...
##  $ SAV_ACCT        : num  4 0 0 0 0 4 2 0 3 0 ...
##  $ EMPLOYMENT      : num  4 2 3 3 2 2 4 2 3 0 ...
##  $ INSTALL_RATE    : num  4 2 2 2 3 2 3 2 2 4 ...
##  $ MALE_DIV        : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ MALE_SINGLE     : num  1 0 1 1 1 1 1 1 0 0 ...
##  $ MALE_MAR_or_WID : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ CO.APPLICANT    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ GUARANTOR       : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ PRESENT_RESIDENT: num  4 2 3 4 4 4 4 2 4 2 ...
##  $ REAL_ESTATE     : num  1 1 1 0 0 0 0 0 1 0 ...
##  $ PROP_UNKN_NONE  : num  0 0 0 0 1 1 0 0 0 0 ...
##  $ AGE             : num  67 22 49 45 53 35 53 35 61 28 ...
##  $ OTHER_INSTALL   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RENT            : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ OWN_RES         : num  1 1 1 0 0 0 1 0 1 1 ...
##  $ NUM_CREDITS     : num  2 1 1 1 2 1 1 1 1 2 ...
##  $ JOB             : num  2 2 1 2 2 1 2 3 1 3 ...
##  $ NUM_DEPENDENTS  : num  1 1 2 2 2 2 1 1 1 1 ...
##  $ TELEPHONE       : num  1 0 0 0 0 1 0 1 0 0 ...
##  $ FOREIGN         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RESPONSE        : num  1 0 1 1 0 1 1 1 1 0 ...
```

**Function to convert the categorical variables to factors**

```
factor_convert <- function(x){
for(i in 1:(length(x) ))
  {
     x[,i] <- as.factor(x[,i])
        }
return(x)
}
```

**Inputing data to the function except the quantitative variables to convert the other variables to factors and removed the OBS variable as it is not necessary to build our model.**

```
data1 <- factor_convert(my_data[,-
c(1,3,11,14,23,27,29)]) data<-
cbind(data1,my_data[,c(3,11,14,23,27,29)]) attach(data)
```

**Proportion of Good to Bad Cases in the data.**

```
tbl<-table(data$RESPONSE)
prop.table(tbl)
##   0   1  ## 0.3 0.7 nrow(data[data$RESPONSE ==
"1",])/nrow(data[data$RESPONSE == "0",])

## [1] 2.333333
```

In the dataset, 30% observations are bad credit risk and 70% of the observations are good credit risk. Proportion of “Good” to “Bad” cases is 2.333.

**Check the summary of the data( Frequencies for categorical and the quartile values for numerical attributes)**

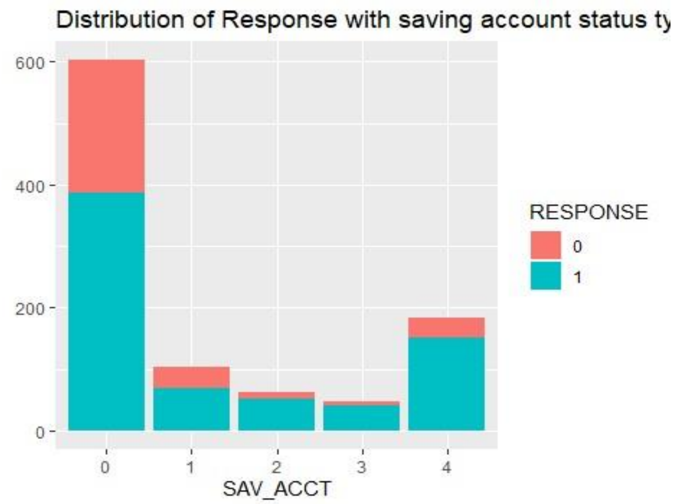
```r
summary(data)
```

```
##  CHK_ACCT HISTORY NEW_CAR USED_CAR FURNITURE RADIO.TV EDUCATION RETRAINING
##  0:274    0: 40   0:766   0:897    0:819     0:720    0:950     0:903
##  1:269    1: 49   1:234   1:103    1:181     1:280    1: 50     1: 97
##  2: 63    2:530
##  3:394    3: 88
##           4:293
##
##  SAV_ACCT EMPLOYMENT MALE_DIV MALE_SINGLE MALE_MAR_or_WID CO.APPLICANT
##  0:603    0: 62      0:950    0:452       0:908           0:959
##  1:103    1:172      1: 50    1:548       1: 92           1: 41
##  2: 63    2:339
##  3: 48    3:174
##  4:183    4:253
##
##  GUARANTOR PRESENT_RESIDENT REAL_ESTATE PROP_UNKN_NONE OTHER_INSTALL
##  0:948     1:130            0:718       0:846          0:814
##  1: 52     2:308            1:282       1:154          1:186
##            3:149
##            4:413
##
##                                                                    ##
RENT    OWN_RES JOB     TELEPHONE FOREIGN RESPONSE    DURATION
##  0:821   0:287   0: 22   0:596     0:963   0:300    Min.   : 4.0
##  1:179   1:713   1:200   1:404     1: 37   1:700    1st Qu.:12.0
##                  2:630                              Median :18.0
##                  3:148                              Mean   :20.9
##                                                     3rd Qu.:24.0
##                                                     Max.   :72.0


##      AMOUNT       INSTALL_RATE        AGE         NUM_CREDITS
##  Min.   :  250   Min.   :1.000   Min.   :19.00   Min.   :1.000
##  1st Qu.: 1366   1st Qu.:2.000   1st Qu.:27.00   1st Qu.:1.000
##  Median : 2320   Median :3.000   Median :33.00   Median :1.000
##  Mean   : 3271   Mean   :2.973   Mean   :35.55   Mean   :1.407
##  3rd Qu.: 3972   3rd Qu.:4.000   3rd Qu.:42.00   3rd Qu.:2.000   ##
Max.   :18424   Max.   :4.000   Max.   :75.00   Max.   :4.000   ##
NUM_DEPENDENTS
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.155
##  3rd Qu.:1.000
##  Max.   :2.000
```

## PLOTS

- **Bar plot to get the distribution of Response with saving account status type**

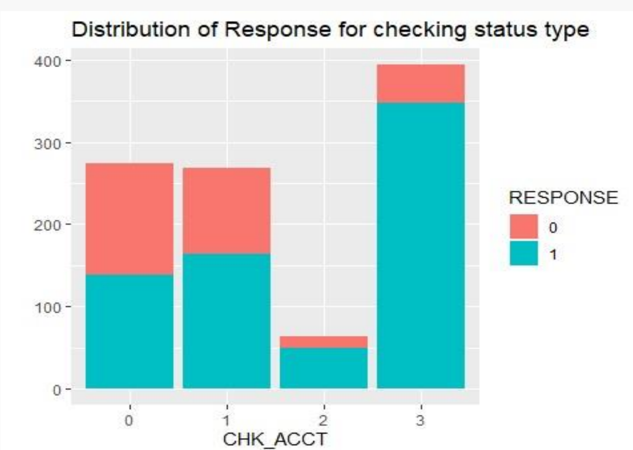
```
library(ggplot2)
qplot(SAV_ACCT, data=data, geom="bar",fill = RESPONSE ,main="Distribution of
Response with saving account status type")
```




This plot shows the distribution of the good and bad customers for the status type of savings account. Majority of the observations savings account balance is less than 100DM and out of those 63% are good applicants.

- **Bar plot to get the distribution on Response for checking account status type**

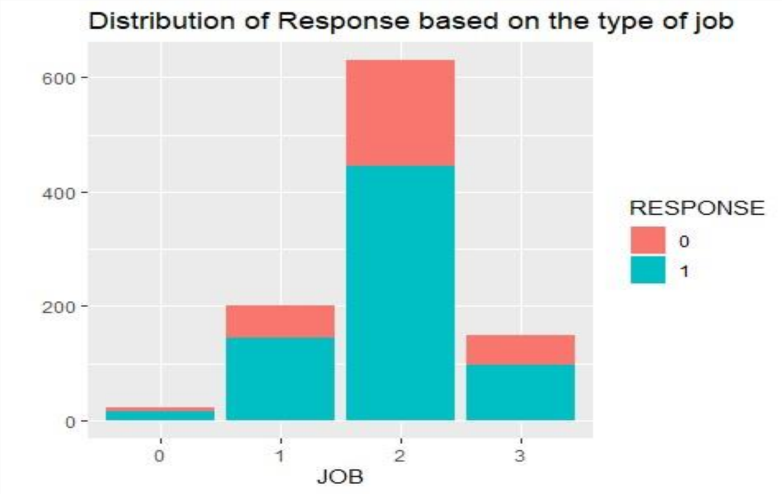
```
qplot(CHK_ACCT, data=data, geom="bar",

 fill = RESPONSE ,main="Distribution of Response for checking status type")
```




Checking account status 0 has higher proportion of bad credit records and status 3 has the l east.

- **Bar plot to get the distribution of Response based on job type**

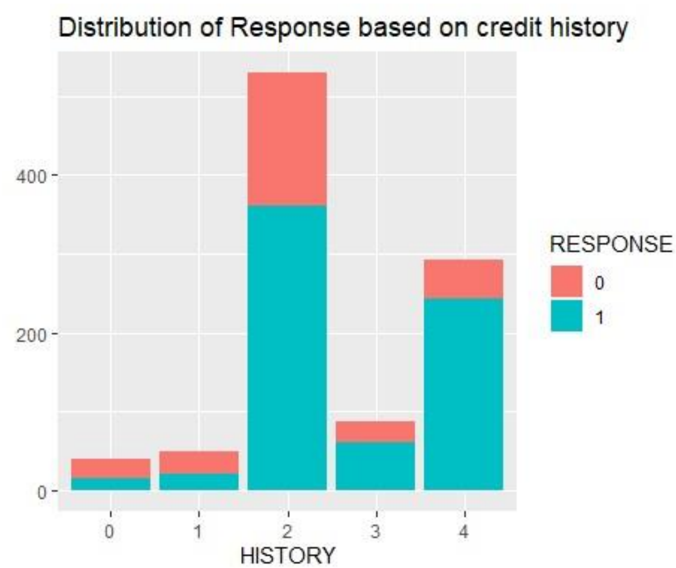
```
qplot(JOB, data=data, geom="bar",
            fill = RESPONSE ,main="Distribution of Response based on the type
of job")
```




Majority of the observations are Skilled employee/official and out of those 66% are good cu stomers.

- **Bar plot to get the distribution of Response based on credit history**

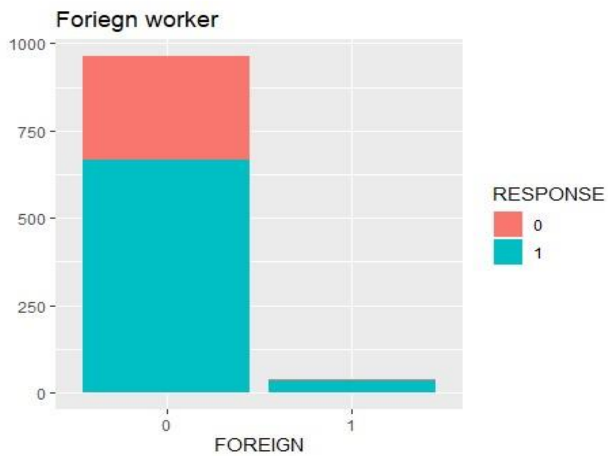
```
qplot(HISTORY, data=data, geom="bar",
            fill = RESPONSE ,main="Distribution of Response based on credit
h istory")
```




For the applicants who previously had no credits taken or all credits at this bank paid back duly, higher proportion are tagged as bad credit risk.

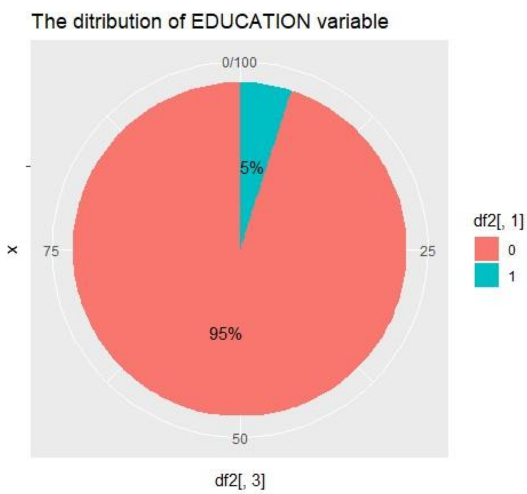
- **Bar plot to get the distribution of Response based on if the observation is foreign worker**

```
qplot(FOREIGN, data=data, geom="bar",
fill = RESPONSE ,main="Foriegn worker")
```




Out of all the applicants, Only 37 are foriegn workers and out of those only 4 are bad customers.

- **Pie chart for Distribution of Education Variable with Response**




Out of all the applicants, 95% are Educated ones

.

## Logistic model to check which are the significant variables

```{r}

log_model <- glm(RESPONSE~., data = data, family = "binomial") summary(log_model)

```

INSTALL_RATE, AMOUNT, FOREIGN, MALE_SINGLE, CHK_ACCT, HISTORY,SAV_ACCT, PRESENT_RESIDENT,DURATION, NEW_CAR are the significant variables which have a relationship with Target Variable RESPONSE.

## (b) MODEL BUILDING

We Split the data randomly into training (60%) and test (40%) partitions

```
set.seed(768)
index = sample(2, nrow(data), replace = T, prob = c(0.6,0.4))
TrainData = data.frame(data[index == 1, ]) TestData =
data.frame(data[index == 2,])
```

### DECISION TREE MODEL USING INFORMATION GAIN (RPART)

We are trying to build a Decision Tree classifier with criteria as information gain. First we try to get the best parameters needed for our model, i.e best Cost Complexity parameter and Maximum depth of the tree.

### PARAMETER TUNING

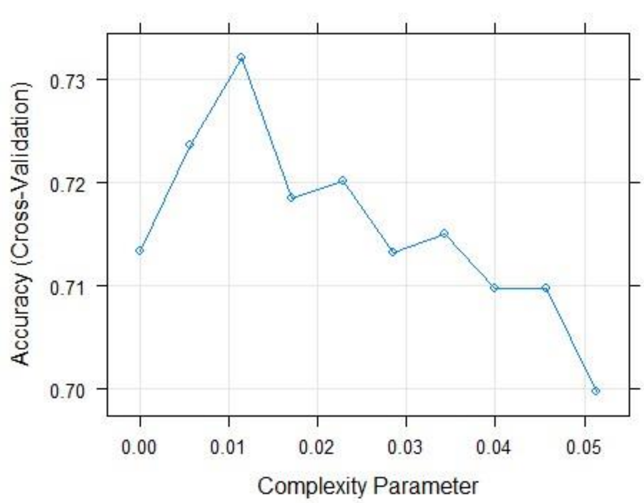
Using train function, we get the optimal value of CP which gives the highest accuracy.

```
library(caret)

x= TrainData[,-25] y
=TrainData$RESPONSE
ctrl = trainControl(method="cv",number =10)
set.seed(16)
dtree1 <- train(x,y, method ="rpart", parms = list(split =
"information"),met ric ="Accuracy", trControl = ctrl, tuneLength = 10)
print(dtree1)

## CART
##
```

```
## 586 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 528, 528, 528, 527, 526, 528, ...
## Resampling results across tuning parameters: ##
##   cp           Accuracy   Kappa
##   0.000000000  0.7133947  0.2597137
##   0.005714286  0.7236236  0.2695025
##   0.011428571  0.7321878  0.3025189
##   0.017142857  0.7185389  0.2670949
##   0.022857143  0.7201198  0.2635006
##   0.028571429  0.7132817  0.2289819
##   0.034285714  0.7150058  0.2345743
##   0.040000000  0.7098042  0.2171004
##   0.045714286  0.7098042  0.2171004
##   0.051428571  0.6997195  0.1267978 ##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01142857.
```

**Plot b/w the Complexity Parameter values with the respective Accuracy**

```
plot(dtree1)
```



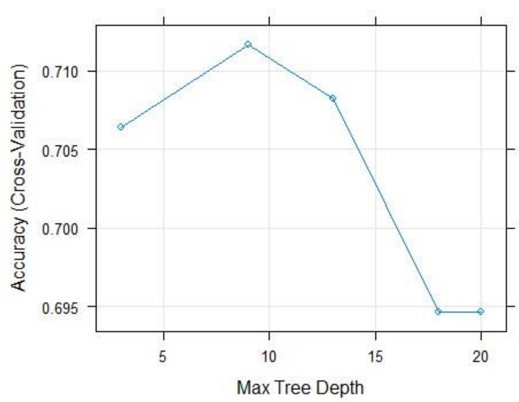

<u>The final optimal CP value is 0.01142857.</u>

**Next using train function we try to get the optimal maximum depth value for the tree.**

```
ctrl = trainControl(method="cv",number =10)
set.seed(10)
dtree1 <- train(x,y, method ="rpart2", parms = list(split =
"information"),me tric ="Accuracy", trControl = ctrl, tuneLength = 10)

## note: only 5 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 5 .

print(dtree1)

## CART
##
## 586 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 528, 527, 527, 527, 528, 528, ...
## Resampling results across tuning parameters:
##
##   maxdepth  Accuracy   Kappa
##    3        0.7064095  0.2203370
##    9        0.7116988  0.2536903
##   13        0.7082505  0.2423861
##   18        0.6946328  0.2187365 ##
20        0.6946328  0.2187365
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was maxdepth = 9.
```

**Plot between max depth value of the tree with accuracy.**

```
plot(dtree1)
```




The final optimal value for maxdepth = 9.

Now we use the optimal values of CP=0.01142857 and max depth=9 which is obtained from the train function and input those values to build the decision tree classifier now with information gain criteria.

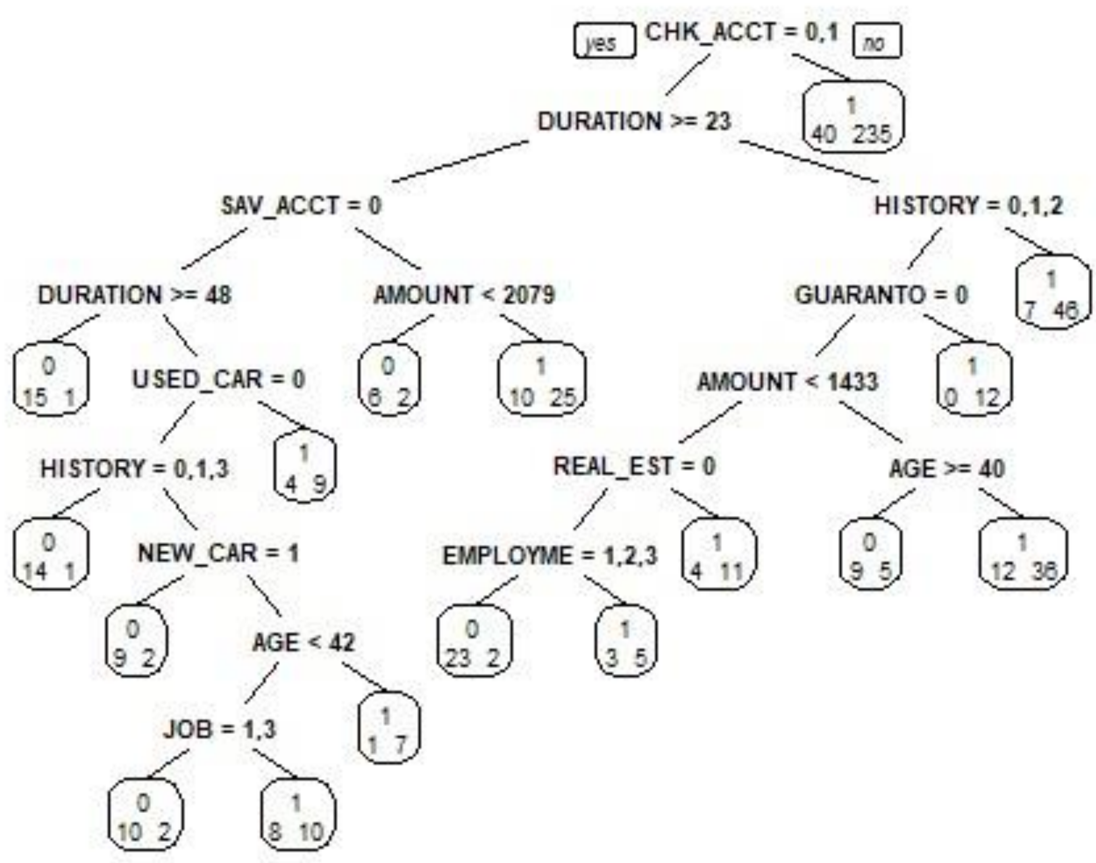
```
library(rpart) library(rpart.plot) rpart_data = rpart(RESPONSE~., data =
TrainData, method = "class", parms = li st(split = "information"),control =
rpart.control(minsplit = 20, cp = 0.01142 857,maxdepth =9))
```

**Decision tree**

```
print(rpart_data)
```

Root node is CHK_ACCT and the root node error is 0.29863

```
prp(rpart_data, extra = 1, yesno =1 )
```




**Best nodes for classifying the good applicants and the corresponding rules**

```
library(rpart.plot) a<-
rpart.rules(rpart_data) a
<- a[a$RESPONSE >=0.70,
]
a

##  RESPONSE
##   0.71 when CHK_ACCT is 0 or 1 & DURATION >=       23
& SAV_ACCT is 1 or 2 or 3 or 4 & AMOUNT >= 2079
##      0.73 when CHK_ACCT is 0 or 1 & DURATION <  23       & HISTORY is 0
or
1 or 2                                 & AMOUNT <  1433                  &
GUAR
ANTOR is 0                            & REAL_ESTATE is 1
##      0.75 when CHK_ACCT is 0 or 1 & DURATION <  23       & HISTORY is 0
or
1 or 2                                 & AMOUNT >= 1433                  &
GUAR
ANTOR is 0 & AGE <  40
##      0.85 when CHK_ACCT is 2 or 3
##      0.87 when CHK_ACCT is 0 or 1 & DURATION <  23       & HISTORY is
3 or 4
##      0.88 when CHK_ACCT is 0 or 1 & DURATION is 23 to 48 & HISTORY is
2 or 4 & SAV_ACCT is                  0                    & USED_CAR is 0
& AGE >= 42 & NEW_CAR is 0
##      1.00 when CHK_ACCT is 0 or 1 & DURATION <  23       & HISTORY is 0
or 1 or 2                                                                &
GUAR ANTOR is 1
```

**BEST NODES FOR CLASSIFYING GOOD CUSTOMERS ARE:**

CHK_ACCT = 2 or 3 (probability of classifying the good customers 0.85), AGE >= 42( probability
= 0.88), GUARANTOR = 1 ( probability = 1), History = 3 or 4 ( probability = 0.87), Age <40 (probability = 0.75), REAL_ESTATE = 1(probability = 0.73), AMOUNT >= 2079(Probability =
0.71)

These are the best nodes because they are having highest probability for classifying the good customers and least misclassification error rates.

**Corresponding Rules :**

- `CHK_ACCT is 0/1 & DURATION <23 & HISTORY is 0/1/2  & GUARANTOR is 1`
- `CHK_ACCT is 0/1 & DURATION is 23 to 48 & HISTORY is 2/4 & SAV_ACCT is 0 & USED_CAR is 0 & AGE >= 42 & NEW_CAR is 0.`
- `CHK_ACCT is 0 or 1 & DURATION <  23  & HISTORY is 3 or 4`
- `CHK_ACCT is 2 or 3`
- `CHK_ACCT is 0 or 1 & DURATION <  23 & HISTORY is 0/1/2 & AMOUNT >= 1433 & GUARANTOR is 0 & AGE <  40`
- `CHK_ACCT is 0 or 1 & DURATION <  23  & HISTORY is 0 or 1 or 2 & AMOUNT <  1433 & GUARANTOR is 0  & REAL_ESTATE is 1`

### Predict on train and test data

```
pred_Train = predict(rpart_data,newdata=TrainData, type="class") pred_Test =
predict(rpart_data, newdata=TestData, type="class")
```

### Error Metrics on train and test

```
confusionMatrix(TrainData$RESPONSE,pred_Train, positive ="1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  86  89
##          1  15 396
##
##                Accuracy : 0.8225
##                  95% CI : (0.7891, 0.8526)
##     No Information Rate : 0.8276
##     P-Value [Acc > NIR] : 0.6528
##
##                   Kappa : 0.5178
##
##  Mcnemar's Test P-Value : 8.172e-13
##
##             Sensitivity : 0.8165
##             Specificity : 0.8515
##          Pos Pred Value : 0.9635
##          Neg Pred Value : 0.4914
##              Prevalence : 0.8276
##          Detection Rate : 0.6758
##    Detection Prevalence : 0.7014
##       Balanced Accuracy : 0.8340
##
##        'Positive' Class : 1
##

c <- confusionMatrix(TestData$RESPONSE,pred_Test, positive = "1")
c

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  45  80
##          1  22 267
##
##                Accuracy : 0.7536
##                  95% CI : (0.7092, 0.7944)
##     No Information Rate : 0.8382
```

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3269
##
##  Mcnemar's Test P-Value : 1.663e-08
##
##             Sensitivity : 0.7695
##             Specificity : 0.6716
##          Pos Pred Value : 0.9239
##          Neg Pred Value : 0.3600
##              Prevalence : 0.8382
##          Detection Rate : 0.6449
##    Detection Prevalence : 0.6981
##       Balanced Accuracy : 0.7205
##                                           ##        'Positive'
Class : 1               ##
```

Here our major concern is to reduce the false positives and also have higher true positives. So, we check at sensitivity and specificity values. 67.16% Specificity and 76.95% sensitivity values on test data.

## Cost calculation

Each observation that is Falsely predicted as positive will give a loss of 500DM and predicting the good customers correctly will have a cost of 100DM. So, we calculate the overall cost that will be incurred.

```
Cost <- (c$table[4]*100)-(c$table[2]*500 )
Cost

## [1] 15700
```

The cost incurred will be 15700DM if we predict using this model.

**FINDING THE OPTIMAL CUTOFF POINT**

Now we try to improve our model by finding a better optimal value to minimize the costs incurred. We have previously predicted the values with threshold of 0.5 but now we try to find the best cutoff value. Also In this scenario false positives are five times as costly as false negatives. So, we give different weights to FP and FN and try to get the optimal cut point.

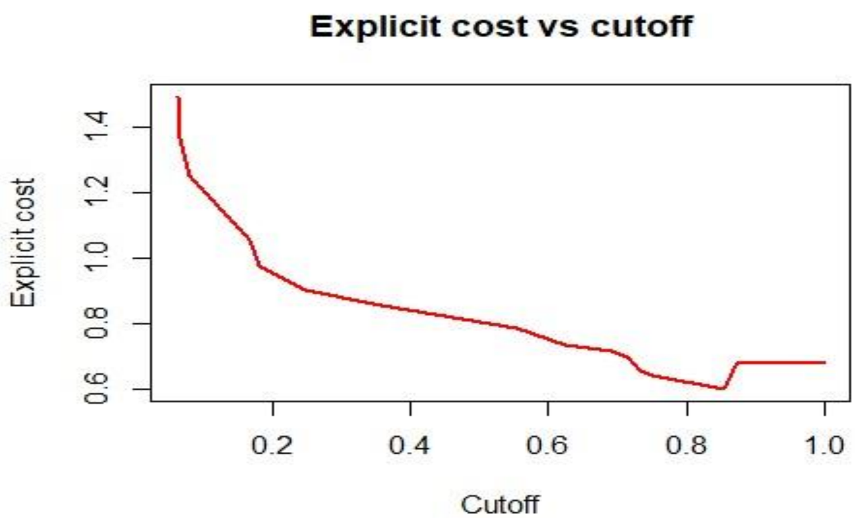
## We find the best optimal cutoff value and predict on train and test

```
library(ROCR)

prob_train <- predict(rpart_data, TrainData, type = "prob") prob_test <-
predict(rpart_data, TestData,type = "prob")
```

```
pred_train <- prediction(prob_train[,2], TrainData$RESPONSE)  pred_test <-
prediction(prob_test[,2],TestData$RESPONSE)
```

False positives are five times as costly as false negatives. So, we give different weights to FP and FN and try to get the optimal cut point.

```
cost.perf = performance(pred_train, "cost", cost.fp = 5, cost.fn = 1)

#Area under curve
auc = performance(pred_train, "auc") auc
plot(cost.perf,main="Explicit cost vs cutoff",col=2,lwd=2)
```




```
#Get the optimal cutoff value
a<-pred_train@cutoffs[[1]] [which.min(cost.perf@y.values[[1]])]
a
## 0.8545455
```

```
The optimal cutoff value is 0.8545455. Now we take this threshold value and p
redict the responses
```

**Predict on train and test and Evaluate the error metrics after changing the cutoff point.**

```r
# If the probability is greater than or equal to the threshold value, we pred
ict the class as 1 else 0.


pred_train_class <- ifelse(prob_train[,2] >= a , "1", "0") pred_train_class
<- as.factor(pred_train_class)

pred_test_class <- ifelse(prob_test[,2] >= a , "1", "0")
pred_test_class <- as.factor(pred_test_class)


#Evaluation Metrics confusionMatrix(pred_train_class,
TrainData$RESPONSE, positive = "1")

## Confusion Matrix and Statistics ##
##           Reference
## Prediction   0   1
##          0 127 111
##          1  48 300
##
##                Accuracy : 0.7287
##                  95% CI : (0.6907, 0.7643)
##     No Information Rate : 0.7014
##     P-Value [Acc > NIR] : 0.07997
##
##                   Kappa : 0.413
##
##  Mcnemar's Test P-Value : 8.792e-07
##
##             Sensitivity : 0.7299
##             Specificity : 0.7257
##          Pos Pred Value : 0.8621
##          Neg Pred Value : 0.5336
##              Prevalence : 0.7014
##          Detection Rate : 0.5119
##    Detection Prevalence : 0.5939
##       Balanced Accuracy : 0.7278
##                                             ##
'Positive' Class : 1
##

c<-confusionMatrix(pred_test_class, TestData$RESPONSE, positive = "1") c

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  90  85
##          1  35 204
##
##                Accuracy : 0.7101
##                  95% CI : (0.6638, 0.7534)
##     No Information Rate : 0.6981
```

```
##     P-Value [Acc > NIR] : 0.317
##
##                   Kappa : 0.3825
##
##  Mcnemar's Test P-Value : 7.711e-06
##
##             Sensitivity : 0.7059
##             Specificity : 0.7200
##          Pos Pred Value : 0.8536
##          Neg Pred Value : 0.5143
##              Prevalence : 0.6981
##          Detection Rate : 0.4928
##    Detection Prevalence : 0.5773
##       Balanced Accuracy : 0.7129
##                                            ##        'Positive'
Class : 1               ##
```

Sensitivity is 70.59% and Specificty is 72%. This is giving better metric values than the model with threshold 0.5 and also there is not much difference in error metrics on train and test data.

## Cost calculation

```
Cost <- (c$table[4]*100)-(c$table[2]*500 )
Cost

## [1] 2900
```

The cost incurred when we use this model to predict the credit risk will be 2900. This is lower when compared to the previous model. Putting an optimal threshold value helped in classifying better by reducing the false positives .

**DECISION TREE CLASSIFIER (CTREE)**

**PARAMETER TUNING**

Using train function to get the optimal value of maximum depth and minimum criterian which gives the highest accuracy.

```
library(party)

library(caret) x=
TrainData[,-25] y
=TrainData$RESPONSE
ctrl = trainControl(method="cv",number =10)
set.seed(8)
dtree2 <- train(x,y, method ="ctree2", metric ="Accuracy", trControl = ctrl)
print(dtree2)

## Conditional Inference Tree
##
## 586 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 527, 528, 528, 527, 528, 528, ...
## Resampling results across tuning parameters:
##
##   maxdepth  mincriterion  Accuracy   Kappa
##   1         0.01          0.7014144  0.0000000
##   1         0.50          0.7014144  0.0000000
##   1         0.99          0.7014144  0.0000000
##   2         0.01          0.7152046  0.2309388
##   2         0.50          0.7152046  0.2309388
##   2         0.99          0.7152046  0.2309388
##   3         0.01          0.7171888  0.2298402
##   3         0.50          0.7119297  0.2358038 ##
3         0.99          0.7135096  0.2298450 ##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were maxdepth = 3 and mincriterion
##  = 0.01.
```

The final values used for the optimal model were maxdepth = 3 and mincriterion = 0.01.

Now we use the optimal values of mincriterian and max depth obtained previously to build the decision tree classifier using ctree

```
ctree_data = ctree(RESPONSE ~ ., data = TrainData, control = ctree_control(mi
nsplit = 30, mincriterion = 0.01, maxdepth = 3))
```

# Predict on train and test data

```
pred_Train = predict(ctree_data,newdata=TrainData, type="response")
pred_Test = predict(ctree_data, newdata=TestData, type="response")
```

# Error Metrics on train and test

```
confusionMatrix(TrainData$RESPONSE,pred_Train, positive ="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  67 108
##          1  34 377
##
##                Accuracy : 0.7577
##                  95% CI : (0.7209, 0.7919)
##     No Information Rate : 0.8276
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3416
##
##  Mcnemar's Test P-Value : 9.01e-10
##
##             Sensitivity : 0.7773
##             Specificity : 0.6634
##          Pos Pred Value : 0.9173
##          Neg Pred Value : 0.3829
##              Prevalence : 0.8276
##          Detection Rate : 0.6433
##    Detection Prevalence : 0.7014
##       Balanced Accuracy : 0.7203
##
##        'Positive' Class : 1
##
```

```r
c <- confusionMatrix(TestData$RESPONSE,pred_Test, positive = "1")
c
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  40  85
##          1  25 264
##
##                Accuracy : 0.7343
##                  95% CI : (0.689, 0.7762)
##     No Information Rate : 0.843
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2703
##
##  Mcnemar's Test P-Value : 1.85e-08
##
##             Sensitivity : 0.7564
##             Specificity : 0.6154
##          Pos Pred Value : 0.9135
##          Neg Pred Value : 0.3200
##              Prevalence : 0.8430
```

```
##          Detection Rate : 0.6377
##    Detection Prevalence : 0.6981
##       Balanced Accuracy : 0.6859
##                                           ##        'Positive'
Class : 1              ##
```

61% Specificity and 75.64% sensitivity values on Test Data.

## Cost calculation for this model.

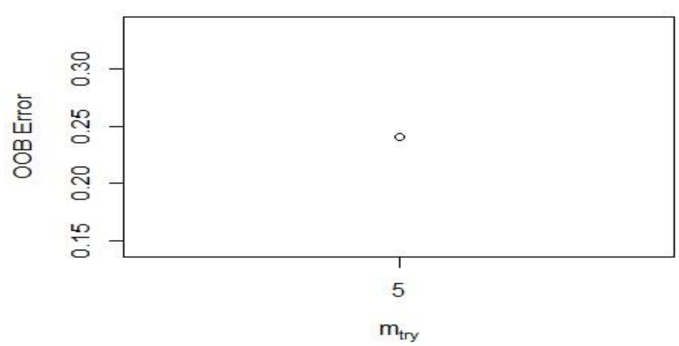
```
Cost <- (c$table[4]*100)-(c$table[2]*500 )
Cost

## [1] 13900
```

Cost incurred when we use this model is 13900.

## <u>RANDOM FOREST MODEL</u>

**PARAMETER TUNING – To get the best mtry value which has the least out of bag error.**

```
library(randomForest)

mtry <- tuneRF(TrainData[-25],TrainData$RESPONSE, ntreeTry=500, stepFactor=1,
improve=0.01, trace=TRUE, plot=TRUE)

## mtry = 5  OOB error = 24.06%
## Searching left ...
## Searching right ...
```




```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)

##       mtry  OOBError
## 5.OOB    5 0.2406143
print(best.m)

## [1] 5
```

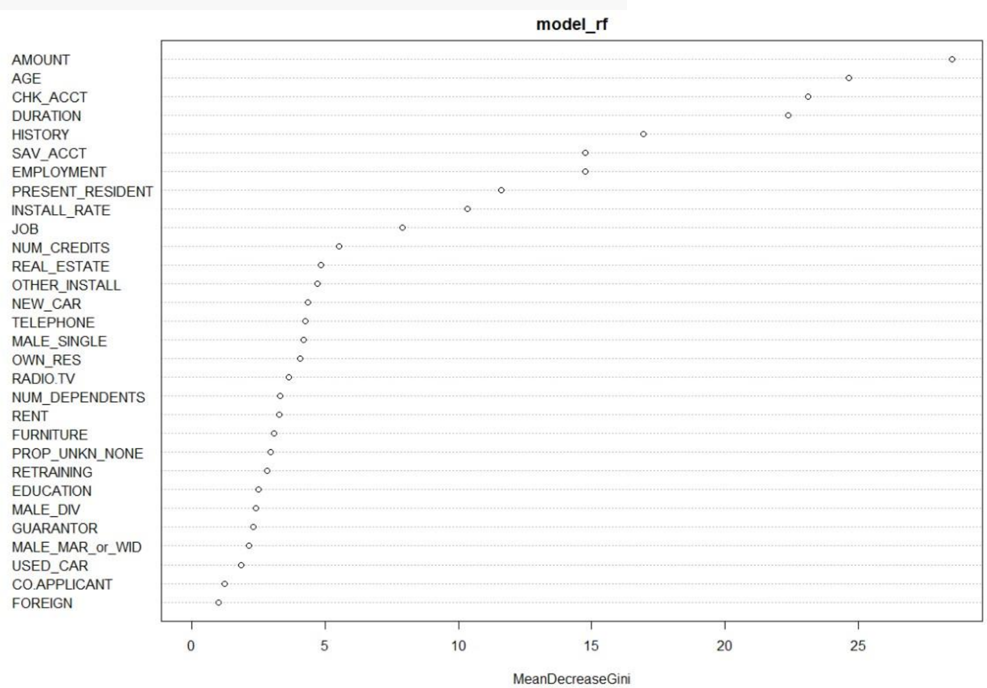
mtry value of 5 has the least out of bag error. So, we choose this and build the random forest model.

**MODEL**

```
set.seed(71)
model_rf <-randomForest(RESPONSE~.,data=TrainData, mtry=best.m, importance=TR
UE, proximity = TRUE,ntree=500) print(model_rf)


##
## Call:
##  randomForest(formula = RESPONSE ~ ., data = TrainData, mtry = best.m,
importance = TRUE, proximity = TRUE, ntree = 500)  ##                Type
of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 24.23% ##
Confusion matrix:
##    0   1 class.error
## 0 60 115  0.65714286
## 1 27 384  0.06569343
```

**Important variables plot**

```
importance(model_rf) varImpPlot(model_rf)
```




AMOUNT is the variable having the HIGHEST IMPORTANCE( Highest Mean Decrease in GINI)

## Predictions on train and test data

```
# Predict on the train data
preds_train_rf <- predict(model_rf, type ="class")
confusionMatrix(preds_train_rf, TrainData$RESPONSE, positive = "1")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0  60  27
##          1 115 384
##
##                Accuracy : 0.7577
##                  95% CI : (0.7209, 0.7919)
##     No Information Rate : 0.7014
##     P-Value [Acc > NIR] : 0.001415
##
##                   Kappa : 0.3239
##
##  Mcnemar's Test P-Value : 2.859e-13
##
##             Sensitivity : 0.9343
##             Specificity : 0.3429
##          Pos Pred Value : 0.7695
##          Neg Pred Value : 0.6897
##              Prevalence : 0.7014
##          Detection Rate : 0.6553
##    Detection Prevalence : 0.8515
##       Balanced Accuracy : 0.6386
##                                              ##
'Positive' Class : 1                ##

#predictions on test data
preds_rf <- predict(model_rf, TestData)
c<-confusionMatrix(preds_rf, TestData$RESPONSE, positive ="1")
c

## Confusion Matrix and Statistics ##
##             Reference
## Prediction   0   1
##          0  51  20
##          1  74 269
##
##                Accuracy : 0.7729
##                  95% CI : (0.7295, 0.8124)
##     No Information Rate : 0.6981
##     P-Value [Acc > NIR] : 0.0004108
##
##                   Kappa : 0.3861
##
##  Mcnemar's Test P-Value : 4.589e-08
##
##             Sensitivity : 0.9308
```

```
##             Specificity : 0.4080
##          Pos Pred Value : 0.7843
##          Neg Pred Value : 0.7183
##              Prevalence : 0.6981
##          Detection Rate : 0.6498
##    Detection Prevalence : 0.8285
##       Balanced Accuracy : 0.6694
##                                                        ##        'Positive'
Class : 1                ##
```

Specificity is 40.8% which is very low.

### Cost calculation

```
Cost <- (c$table[4]*100)-(c$table[2]*500 )
Cost

## [1] -10100
```

We can see that as Specificity for this model is very low, that is False positives are higher, there is a loss of 10100 that is incurred if we use this model to predict.

### CHANGING THE THRESHOLD

```
library(ROCR)
score1 <- model_rf$votes[,2]
pred <- prediction(score1, TrainData$RESPONSE)

cost.perf = performance(pred, "cost", cost.fp = 5, cost.fn = 1) a<-pred@cutoffs[[1]]
[which.min(cost.perf@y.values[[1]])]


# Predict on the train data
preds_train_rf <- predict(model_rf, TrainData,type ="prob") preds_test_rf <-
predict(model_rf, TestData,type ="prob")

pred_train_class <- ifelse(preds_train_rf[,2] >= a , "1", "0") pred_train_class <-
as.factor(pred_train_class)

pred_test_class <- ifelse(preds_test_rf[,2] >= a , "1", "0") pred_test_class <-
as.factor(pred_test_class)
```

```
#Error Metrics confusionMatrix(pred_train_class, TrainData$RESPONSE,
positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 175   9
##          1   0 402
##
##                Accuracy : 0.9846
##                  95% CI : (0.971, 0.993)
##     No Information Rate : 0.7014
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9639
##
##  Mcnemar's Test P-Value : 0.007661
##
##             Sensitivity : 0.9781
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9511
##              Prevalence : 0.7014
##          Detection Rate : 0.6860
##    Detection Prevalence : 0.6860
##       Balanced Accuracy : 0.9891
##                                            ##
'Positive' Class : 1
##

c<- confusionMatrix(pred_test_class, TestData$RESPONSE, positive = "1")
c

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 104 127
##          1  21 162
##
##                Accuracy : 0.6425
##                  95% CI : (0.5942, 0.6887)
##     No Information Rate : 0.6981
##     P-Value [Acc > NIR] : 0.9935
##
##                   Kappa : 0.3164
##
##  Mcnemar's Test P-Value : <2e-16
##
```

```
##            Sensitivity : 0.5606
##            Specificity : 0.8320
##         Pos Pred Value : 0.8852
##         Neg Pred Value : 0.4502
##             Prevalence : 0.6981
##         Detection Rate : 0.3913
##   Detection Prevalence : 0.4420
##      Balanced Accuracy : 0.6963
##
##       'Positive' Class : 1                ##
```

Though Specificity value is 83%, sensitivity values is low.

## Cost Calculation

```
Cost <- (c$table[4]*100)-(c$table[2]*500 )
Cost

## [1] 5700
```

The Cost incurred is 5700 which is low when compared to the model before when the threshold was 0.5

---

**DECISION TREE (RPART) WITH ONLY SIGNIFICANT VARIABLES**

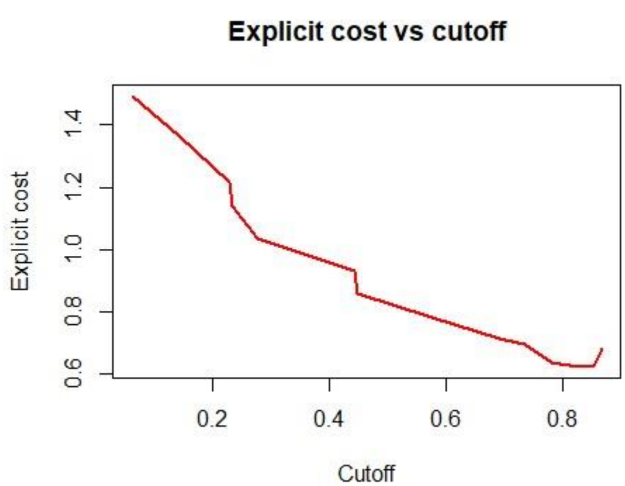
Took the variables with high meanDecreaseGini from the variable importance plot and trying to build a new decision tree model with only those variables.

```
rpart_data = rpart(RESPONSE~ AMOUNT + AGE + CHK_ACCT + DURATION +      HISTOR
Y + EMPLOYMENT + SAV_ACCT + PRESENT_RESIDENT + INSTALL_RATE +  JOB, data = Tr
ainData, method = "class", parms = list(split = "information"),control = rpar
t.control(minsplit = 30, cp = 0.01142857,maxdepth =9))
```

## We find the best optimal cutoff value and predict on train and test

```
library(ROCR)
prob_train <- predict(rpart_data, TrainData, type = "prob") prob_test <-
predict(rpart_data, TestData,type = "prob")
```

```
pred_train <- prediction(prob_train[,2], TrainData$RESPONSE) pred_test
<- prediction(prob_test[,2],TestData$RESPONSE)

cost.perf = performance(pred_train, "cost", cost.fp = 5, cost.fn = 1)

#Area under curve
auc = performance(pred_train, "auc")
auc

## An object of class "performance" ##
Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.7738547
##
##
## Slot "alpha.values":
## list() plot(cost.perf,main="Explicit cost vs
cutoff",col=2,lwd=2)
```




```
#Get the optimal cutoff value
a<-pred_train@cutoffs[[1]] [which.min(cost.perf@y.values[[1]])]
a

##       996
## 0.8545455
```

## Error on train and test

```
# If the probability is greater than or equal to the threshold value, we pred
ict the class as 1 else 0.
pred_train_class <- ifelse(prob_train[,2] >= a , "1", "0") pred_train_class
<- as.factor(pred_train_class)

pred_test_class <- ifelse(prob_test[,2] >= a , "1", "0")
pred_test_class <- as.factor(pred_test_class)


#Evaluation Metrics confusionMatrix(pred_train_class,
TrainData$RESPONSE, positive = "1")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##            0 128 130
##            1  47 281
##
##                  Accuracy : 0.698
##                    95% CI : (0.659, 0.7349)
##       No Information Rate : 0.7014
##       P-Value [Acc > NIR] : 0.5915
##
##                     Kappa : 0.3654
##
##    Mcnemar's Test P-Value : 7.116e-10
##
##               Sensitivity : 0.6837
##               Specificity : 0.7314
##            Pos Pred Value : 0.8567
##            Neg Pred Value : 0.4961
##                Prevalence : 0.7014
##            Detection Rate : 0.4795
##      Detection Prevalence : 0.5597
##         Balanced Accuracy : 0.7076
##                                              ##
'Positive' Class : 1              ##
```

```
c<-confusionMatrix(pred_test_class, TestData$RESPONSE, positive = "1")
c

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  92  96
##          1  33 193
##
##                Accuracy : 0.6884
##                  95% CI : (0.6414, 0.7327)
##     No Information Rate : 0.6981
##     P-Value [Acc > NIR] : 0.6869
##
##                   Kappa : 0.3533
##
##  Mcnemar's Test P-Value : 4.794e-08
##
##             Sensitivity : 0.6678
##             Specificity : 0.7360
##          Pos Pred Value : 0.8540
##          Neg Pred Value : 0.4894
##              Prevalence : 0.6981
##          Detection Rate : 0.4662
##    Detection Prevalence : 0.5459
##       Balanced Accuracy : 0.7019
##                                            ##
'Positive' Class : 1                 ##
```

Sensitivity value of 66.78 and specificty of 73.60

## Cost Calculation

```
Cost <- (c$table[4]*100)-(c$table[2]*500 )
Cost

## [1] 2800
```

Cost incurred is 2800.

**COMPARISON OF VARIOUS MODELS ERROR METRICS ON TEST DATA (CUTOFF = 0.5 )**

| | | Test Data | | |
|---|---|---|---|---|
| Model | Sensitivity | Specificity | Accuracy | Cost ((FP*-500) + (TP*100)) |
| | | | | |
| DecisionTree (RPART), Threshold =0.5 | 76.95 | 67.16 | 75.36 | 15700 DM |
| Decision tree (Ctree) Threshold =0.5 | 75.64 | 61.54 | 73.43 | 13900 DM |
| Random Forest, threshold =0.5 | 91 | 40.6 | 74.64 | -13200 DM |

First built Decision tree model with all the variables using rpart and ctree and then built random forest Model. In our problem, our major concern is to reduce the false positives and also to get higher true positives. So, we check at sensitivity and specificity values.

Out of all these models, when the threshold was 0.5, Decision tree using rpart was giving better error metric values. True Positives are higher and also false positives are low. Here we can observe that though random forest Model has high sensitivity, its specificity value is very low. We need a trade off between False Positives and True Positives. So, Decision tree using rpart is the best model we built with parameters ( CP = 0.01142857 and maximum depth of the tree = 9)

Previously we have predicted the values with threshold of 0.5 but now we try to find the best cutoff value. Also, In this scenario false positives are five times as costly as false negatives. So, we give different weights to FP and FN and try to get the optimal cut point.

Using this optimal threshold value we try to predict the classes. If the probability is greater than or equal to the threshold value, we predict the class as 1 else 0. Doing this can improve our model which helps to minimize the cost incurred.

**COMPARISON OF VARIOUS MODELS ERROR METRICS ON TEST DATA (NEW CUTOFF)**

| Model | Sensitivity | Specificity | Accuracy | Cost ((FP*-500) + (TP*100)) |
|---|---|---|---|---|
| | | | | |
| DecisionTree (RPART) with new optimal cutoff | 70.59 | 72 | 71.01 | 2900 DM |
| Decision Tree (RPART) with only significant variables, optimal threshold cutoff | 68.78 | 73.6 | 70.84 | 2800 DM |
| Random Forest with new optimal cutoff | 74.25 | 83.2 | 74.25 | 5700 DM |

Our objective is to reduce the False Positives as the cost incurred if we falsely predict it is high when compared to true positives. So, the metric that we are supposed to check is specificity. It should be higher. But also our True Positives should be high means Sensitivity also should be high.

From the models that we built, Random forest was giving us better specificity and also better sensitivity.