LOAN APPROVAL USING PREDICTIVE ANALYTICS

To minimize loss from the bank's perspective, the bank needs a decision rule regarding who to give approval of the loan and who not to. An applicant's demographic and socio-economic profiles are considered by loan managers before a decision is taken regarding his/her loan application. The Bank.data contains data on 9 input variables and the classification target indicating whether an applicant is considered a **Good or a Bad credit risk** for 1000 loan applicants. A predictive model developed on this data is expected to provide a bank manager guidance for making a decision whether to approve a loan to a prospective applicant based on his/her profiles.

The variables in this dataset are:

• Age (numeric)

• Sex (categorical: male, female)

• Job (categorical: 0 - unskilled and non-resident, 1 - unskilled and resident, 2 - skilled, 3 - highly skilled)

• Housing (categorical: own, rent, or free)

• Saving accounts (categorical: little, moderate, quite rich, rich)

• Checking account (categorical: little, moderate, rich)

• Credit amount (numeric, in USD)

• Duration (numeric, in month)

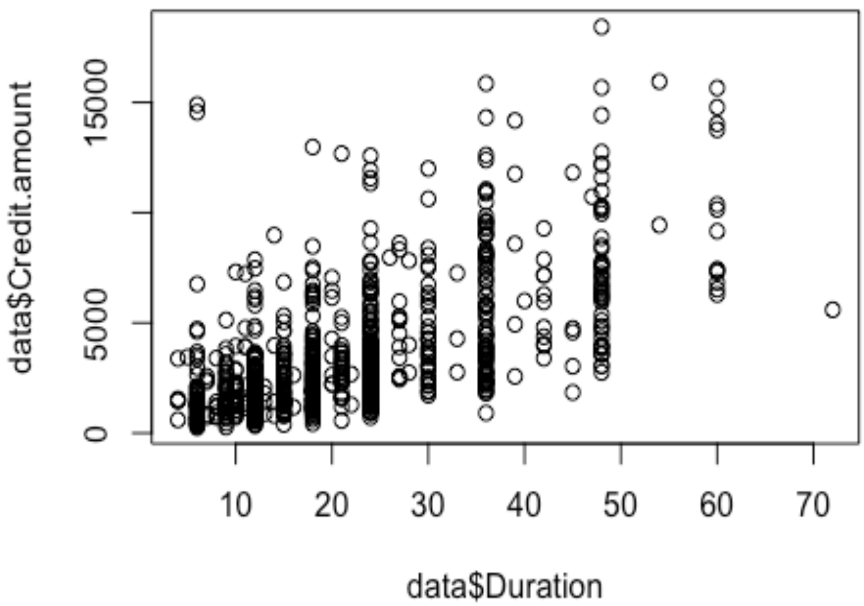
• Purpose (categorical: car, furniture/equipment, radio/TV, domestic appliances, repairs, education, business, vacation/others)

• Target (categorical: 1 - Good, 0 - Bad)

```
data <- read.csv("~/Downloads/Bank.data.csv")
head(data)

##   ID Age    Sex Job Housing Saving.accounts Checking.account Credit.amount
## 1  0  67   male   2     own            <NA>           little          1169
## 2  1  22 female   2     own          little         moderate          5951
## 3  2  49   male   1     own          little             <NA>          2096
## 4  3  45   male   2    free          little           little          7882
## 5  4  53   male   2    free          little           little          4870
## 6  5  35   male   1    free            <NA>             <NA>          9055
##   Duration             Purpose Target
## 1        6            radio/TV      1
## 2       48            radio/TV      0
## 3       12           education      1
## 4       42 furniture/equipment      1
## 5       24                 car      0
## 6       36           education      1
```

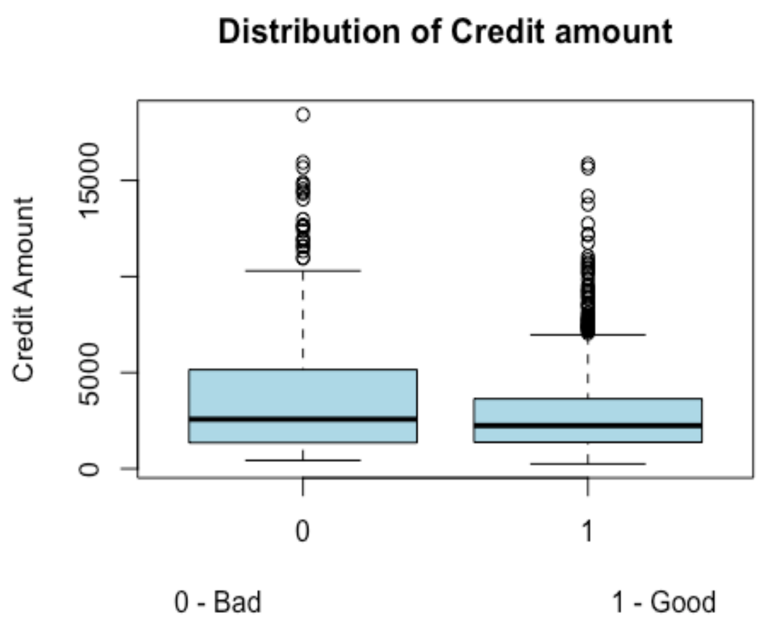
```r
#formatting data
data$Sex <- as.factor(data$Sex)
data$Job <- as.factor(data$Job)
data$Housing <- as.factor(data$Housing)
data$Saving.accounts <- as.factor(data$Saving.accounts)
data$Checking.account <- as.factor(data$Checking.account)
data$Purpose <- as.factor(data$Purpose)
data$Target <- as.factor(data$Target)

data$Age <- as.numeric(data$Age)
data$Credit.amount <- as.numeric(data$Credit.amount)
data$Duration <- as.numeric(data$Duration)

summary(data)

##        ID             Age            Sex       Job     Housing
##  Min.   :  0.0   Min.   :19.00   female:310   0: 22   free:108
##  1st Qu.:249.8   1st Qu.:27.00   male  :690   1:200   own :713
##  Median :499.5   Median :33.00                2:630   rent:179
##  Mean   :499.5   Mean   :35.55                3:148
##  3rd Qu.:749.2   3rd Qu.:42.00
##  Max.   :999.0   Max.   :75.00
##
##    Saving.accounts Checking.account Credit.amount      Duration
##  little    :603    little  :274     Min.   :  250   Min.   : 4.0
##  moderate  :103    moderate:269     1st Qu.: 1366   1st Qu.:12.0
##  quite rich: 63    rich    : 63     Median : 2320   Median :18.0
##  rich      : 48    NA's    :394     Mean   : 3271   Mean   :20.9
##  NA's      :183                     3rd Qu.: 3972   3rd Qu.:24.0
##                                     Max.   :18424   Max.   :72.0
##
##                Purpose    Target
##  car                :337   0:300
##  radio/TV           :280   1:700
##  furniture/equipment:181
##  business           : 97
##  education          : 59
##  repairs            : 22
##  (Other)            : 24

#5 number summary for the credit amount  (minimum, 1st quartile, median, 3rd quartile, maximum)
summary(data$Credit.amount)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     250    1366    2320    3271    3972   18424

#correlation of Duration and Credit
cor(data$Duration, data$Credit.amount)

## [1] 0.6249842
```

```r
plot(data$Duration, data$Credit.amount)
```



Variables are NOT highly correlated so we can keep both variables for the analysis (Usually, high correlation is considered above 0.75)

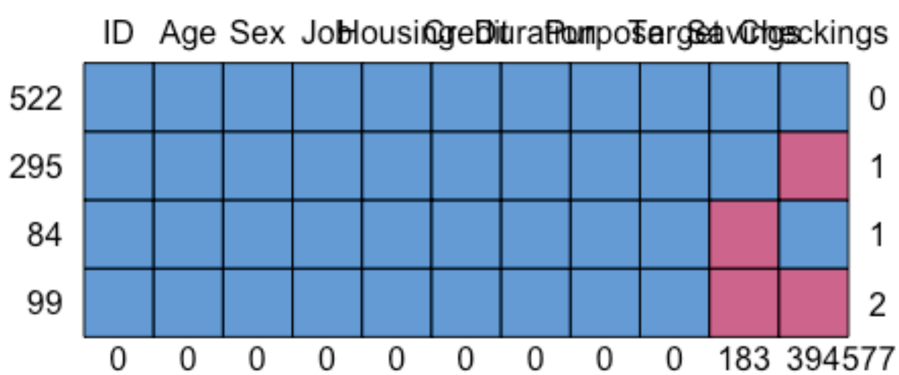
```r
# Distribution of Credit amount for applicants considered as “Good” and “Bad”
credit risk.

boxplot(data$Credit.amount  ~  data$Target, main ="Distribution of Credit amo
unt", xlab = "0 - Bad                                                 1 - Good",
ylab = "Credit Amount", title = "xccc" , col = "lightblue")
```



```
# table that contains the frequency of different housing types (free, own, re
nt) for "Good" and "Bad" instances.
table(data$Housing, data$Target, useNA = "ifany", dnn= c("Housing", "Credit")
)

##         Credit
## Housing   0   1
##    free  44  64
##    own  186 527
##    rent  70 109

#Renaming some variables
names(data)[names(data) == "Checking.account"] <- "Checkings"
names(data)[names(data) == "Saving.accounts"] <- "Savings"
names(data)[names(data) == "Credit.amount"] <- "Credit"
```

HANDLING MISSING VALUES

```
#Handling missing values
library(mice)

## Warning: package 'mice' was built under R version 4.0.2

##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind

md.pattern(data)
```




```
##     ID Age Sex Job Housing Credit Duration Purpose Target Savings Checking
s
## 522  1   1   1   1       1      1        1       1      1       1
1   0
## 295  1   1   1   1       1      1        1       1      1       1
0   1
```

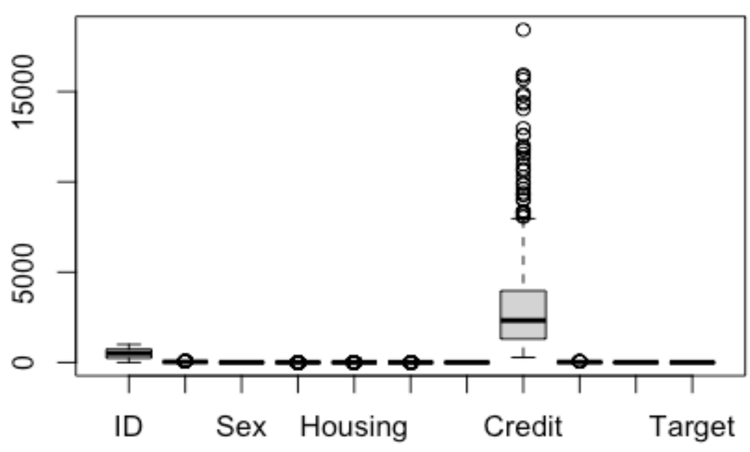
```
## 84   1   1   1   1       1      1       1      1      1      0
1   1
## 99   1   1   1   1       1      1       1      1      1      0
0   2
##      0   0   0   0       0      0       0      0      0    183       39
4 577
```

The output tells us that 522 samples are complete, 295 samples miss ONLY Checking.account, 84 samples miss only the saving.accounts and 99 samples miss both Saving.accounts and Checking.account.

```
#As far as categorical variables are concerned, replacing categorical variabl
es is usually not advisable. Some common practice include replacing missing c
ategorical variables with the mode of the observed ones, however, it is quest
ionable whether it is a good choice.

#Since all NA values are from categorical variables:

#REMOVE ALL ROWS  WITH NA VALUES
Data <- na.omit(data)
sum(is.na(Data))

## [1] 0
```

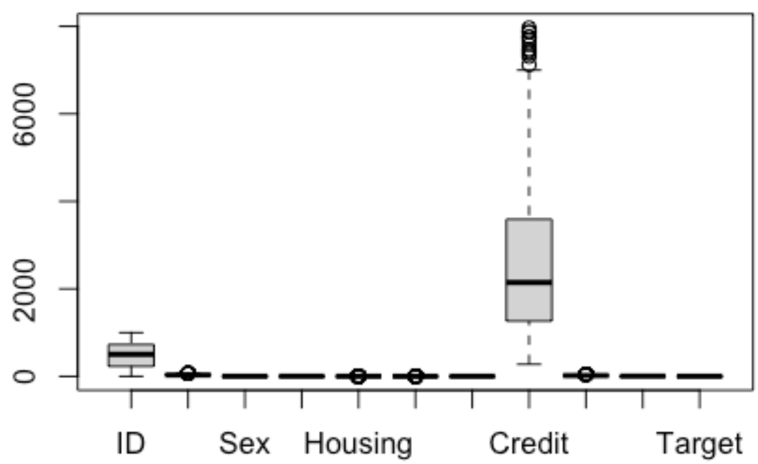
HANDLING OUTLIERS

```
library(dplyr)

boxplot(Data)
```




```
#credit variable
credit_outliers = boxplot.stats(Data$Credit)$out # We first save all the outl
iers in the vector
credit_outliers

##  [1] 12579 14421 12612 15945 11938 10623 10961  8978 11998 10722  9398   99
60
```

```
## [13] 14782 14318 12976 11760 11328  8318  9034  8086  9857 14027 11560  80
65
## [25]  9271  9283  9629 15857 11816 15672 18424 14896 10297  8358  8386  82
29

#duration variable
Duration_outliers = boxplot.stats(Data$Duration)$out # We first save all the
outliers in the vector
Duration_outliers

## [1] 60 54 60 60 72 60 60 60

#REMOVING OUTLIERS FROM DATA
Data<- Data[-which(Data$Duration %in% Duration_outliers),]
Data<- Data[-which(Data$Credit %in% credit_outliers),]
boxplot(Data)
```



Train/Test split

```
set.seed(123)
indx  <- sample(2,nrow(Data), replace=TRUE, prob = c(0.7, 0.3))
train <- Data[indx==1, ]
test  <- Data[indx==2, ]
```

Logistic Regression Model using Forward Selection Technique

Forward Selection considers one variable at a time:

-If the variable improves the model (reduces AIC), we include it

-Otherwise, we don't include it.

We look at r-squared, f-test or AIC.

We will check all possibilities from the null case to the full case

```r
#Extreme Cases
full <- glm(Target ~ . , data=Data, family = "binomial")
null <- glm(Target ~ 1 , data=Data, family = "binomial")
step(null, scope = list(lower=null, upper=full), direction="forward")
```

```
## Start:  AIC=652.98
## Target ~ 1
##
##              Df Deviance    AIC
## + Duration    1   622.77 626.77
## + Checkings   2   638.19 644.19
## + Savings     3   640.57 648.57
## + Housing     2   643.20 649.20
## + Sex         1   648.04 652.04
## + Age         1   648.46 652.46
## + ID          1   648.80 652.80
## <none>            650.98 652.98
## + Credit      1   650.83 654.83
## + Job         3   649.00 657.00
## + Purpose     7   641.08 657.08
##
## Step:  AIC=626.77
## Target ~ Duration
##
##              Df Deviance    AIC
## + Credit      1   609.26 615.26
## + Checkings   2   612.15 620.15
## + Sex         1   617.87 623.87
## + Savings     3   614.97 624.97
## + Housing     2   617.31 625.31
## + ID          1   619.91 625.91
## + Age         1   620.16 626.16
## <none>            622.77 626.77
## + Purpose     7   608.85 626.85
## + Job         3   620.28 630.28
##
## Step:  AIC=615.26
## Target ~ Duration + Credit
##
##              Df Deviance    AIC
## + Checkings   2   597.86 607.86
## + Housing     2   601.96 611.96
## + Savings     3   600.45 612.45
## + Sex         1   605.29 613.29
## + Purpose     7   594.51 614.51
## + Age         1   607.10 615.10
## <none>            609.26 615.26
## + ID          1   607.27 615.27
## + Job         3   608.57 620.57
##
```

```
## Step:  AIC=607.86
## Target ~ Duration + Credit + Checkings
##
##           Df Deviance    AIC
## + Sex      1   593.64 605.64
## + Housing  2   591.77 605.77
## + Savings  3   590.55 606.55
## <none>         597.86 607.86
## + Age      1   595.87 607.87
## + Purpose  7   583.96 607.96
## + ID       1   596.39 608.39
## + Job      3   597.12 613.12
##
## Step:  AIC=605.64
## Target ~ Duration + Credit + Checkings + Sex
##
##           Df Deviance    AIC
## + Savings  3   585.88 603.88
## + Housing  2   589.07 605.07
## <none>         593.64 605.64
## + Purpose  7   579.86 605.86
## + Age      1   592.33 606.33
## + ID       1   592.52 606.52
## + Job      3   593.15 611.15
##
## Step:  AIC=603.88
## Target ~ Duration + Credit + Checkings + Sex + Savings
##
##           Df Deviance    AIC
## + Housing  2   580.99 602.99
## + Purpose  7   571.70 603.70
## <none>         585.88 603.88
## + Age      1   584.86 604.86
## + ID       1   584.97 604.97
## + Job      3   585.07 609.07
##
## Step:  AIC=602.99
## Target ~ Duration + Credit + Checkings + Sex + Savings + Housing
##
##           Df Deviance    AIC
## <none>         580.99 602.99
## + Age      1   579.73 603.73
## + ID       1   580.19 604.19
## + Purpose  7   568.19 604.19
## + Job      3   579.85 607.85

##
## Call:  glm(formula = Target ~ Duration + Credit + Checkings + Sex +
##     Savings + Housing, family = "binomial", data = Data)
##
```

```
## Coefficients:
##       (Intercept)          Duration            Credit  Checkingsmoderate
##         0.2091507        -0.0728745         0.0002961          0.2653101
##     Checkingsrich           Sexmale   Savingsmoderate  Savingsquite rich
##         1.0498166         0.3809951         0.0592703          0.5239743
##       Savingsrich        Housingown       Housingrent
##         1.5387266         0.4530058        -0.0441255
##
## Degrees of Freedom: 480 Total (i.e. Null);  470 Residual
## Null Deviance:       651
## Residual Deviance: 581   AIC: 603
```

AIC = −2log(Likelihood) + 2K

AIC = Residual deviance + 2 × number of parameters.

AIC is a single number score that can be used to determine which of multiple models is most likely to be the best model for a given dataset.

A lower AIC score is better.

"The Akaike information criterion (AIC) is an estimator of out-of-sample prediction error and thereby relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Thus, AIC provides a means for model selection."

```
#Log Regression with variables selected
Log_Reg <- glm(Target ~ Duration + Credit + Checkings + Sex + Savings + Housi
ng, family = "binomial", data = Data)

predictions <- predict(Log_Reg, newdata = test, type="response")
#probability of being in class GOOD
Class <- ifelse(predictions >= 0.5 , 1, 0)

#Confusion Matrix
table(test$Target , Class, dnn=c("Predictions", "Actual"))

##            Actual
## Predictions  0  1
##           0 23 26
##           1 17 70

#accuracy function
accuracy<- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  accuracy <- (y["TN"] + y["TP"])/ sum(y)
  return(as.numeric(accuracy))
}
```

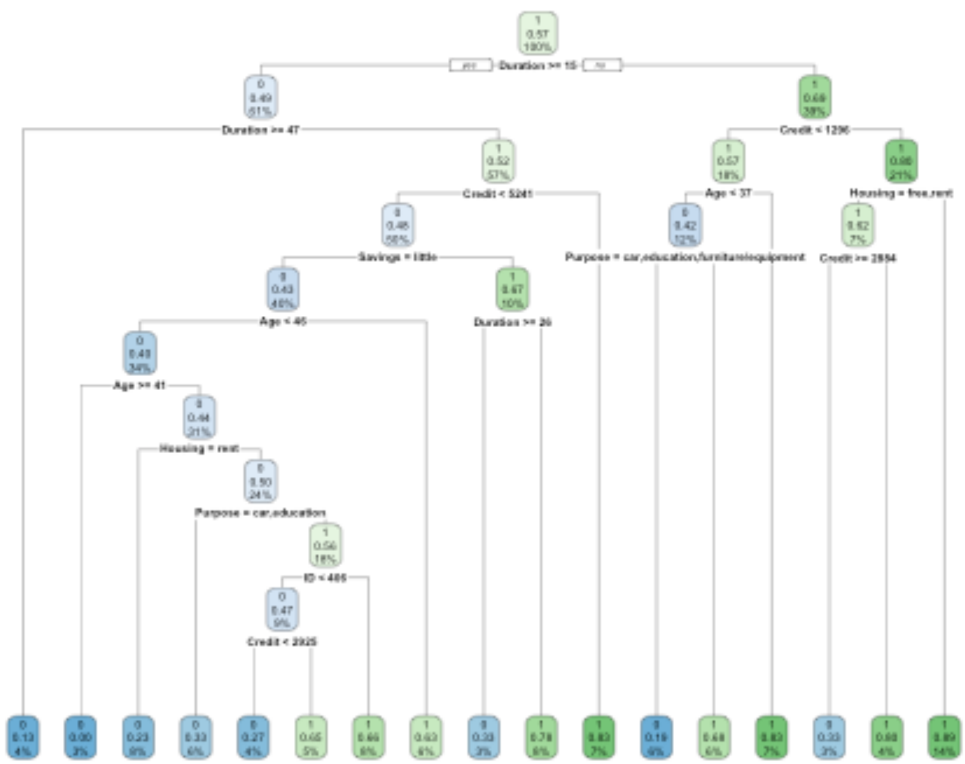
```r
accuracy(test$Target, Class)

## [1] 0.6838235

summary(Log_Reg)

##
## Call:
## glm(formula = Target ~ Duration + Credit + Checkings + Sex +
##     Savings + Housing, family = "binomial", data = Data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1902  -1.1094   0.6413   0.9470   1.6766
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         2.092e-01  4.278e-01   0.489 0.624885
## Duration           -7.287e-02  1.250e-02  -5.828 5.62e-09 ***
## Credit              2.961e-04  7.829e-05   3.782 0.000155 ***
## Checkingsmoderate   2.653e-01  2.187e-01   1.213 0.225134
## Checkingsrich       1.050e+00  3.693e-01   2.842 0.004478 **
## Sexmale             3.810e-01  2.182e-01   1.746 0.080752 .
## Savingsmoderate     5.927e-02  3.205e-01   0.185 0.853304
## Savingsquite rich   5.240e-01  5.060e-01   1.036 0.300405
## Savingsrich         1.539e+00  6.568e-01   2.343 0.019149 *
## Housingown          4.530e-01  3.322e-01   1.364 0.172637
## Housingrent        -4.413e-02  3.787e-01  -0.117 0.907254
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 650.98  on 480  degrees of freedom
## Residual deviance: 580.99  on 470  degrees of freedom
## AIC: 602.99
##
## Number of Fisher Scoring iterations: 4
```

We can see the significant variables with a *.

DECISION TREE

```r
library(rpart)

library(rpart.plot)
```

```r
tree_model <- rpart(Target ~ . , data=train)

rpart.plot(tree_model)
```



rpart(formula, data= train, parms= , control=)

control - controls how to split. control = rpart.control(minsplit=10)

minsplit = 10 -> at least 10 instances must be in each node so that it could be split further

minbucket = 10 -> min num of instances expected in terminal nodes

cp - complexity parameter -> is used to control the size of the decision tree and to select the optimal tree size.

want the tree with the min error & also min size of tree

- when cp is large – size of tree is small and error is larger
- when cp is small – size of tree is large and error is smaller

Example: rpart(y~., data, parms=list(split=c(“information”,“gini”)), cp = 0.01, minsplit=20, minbucket=7, maxdepth=30)

```r
print(tree_model)
```

```
## n= 345
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##    1) root 345 148 1 (0.4289855 0.5710145)
##      2) Duration>=14.5 211 104 0 (0.5071090 0.4928910)
##        4) Duration>=46.5 15   2 0 (0.8666667 0.1333333) *
##        5) Duration< 46.5 196  94 1 (0.4795918 0.5204082)
##         10) Credit< 5240.5 173  83 0 (0.5202312 0.4797688)
##           20) Savings=little 137  59 0 (0.5693431 0.4306569)
##             40) Age< 45.5 118  47 0 (0.6016949 0.3983051)
##               80) Age>=40.5 10   0 0 (1.0000000 0.0000000) *
##               81) Age< 40.5 108  47 0 (0.5648148 0.4351852)
##                162) Housing=rent 26   6 0 (0.7692308 0.2307692) *
##                163) Housing=free,own 82  41 0 (0.5000000 0.5000000)
##                  326) Purpose=car,education 21   7 0 (0.6666667 0.3333333)
*
##                  327) Purpose=business,furniture/equipment,radio/TV,repair
s,vacation/others 61  27 1 (0.4426230 0.5573770)
##                    654) ID< 406 32  15 0 (0.5312500 0.4687500)
##                     1308) Credit< 2925 15   4 0 (0.7333333 0.2666667) *
##                     1309) Credit>=2925 17   6 1 (0.3529412 0.6470588) *
##                    655) ID>=406 29  10 1 (0.3448276 0.6551724) *
##             41) Age>=45.5 19   7 1 (0.3684211 0.6315789) *
##           21) Savings=moderate,quite rich,rich 36  12 1 (0.3333333 0.66666
67)
##             42) Duration>=25.5 9   3 0 (0.6666667 0.3333333) *
##             43) Duration< 25.5 27   6 1 (0.2222222 0.7777778) *
##         11) Credit>=5240.5 23   4 1 (0.1739130 0.8260870) *
##      3) Duration< 14.5 134  41 1 (0.3059701 0.6940299)
##        6) Credit< 1296 63  27 1 (0.4285714 0.5714286)
##         12) Age< 36.5 40  17 0 (0.5750000 0.4250000)
##           24) Purpose=car,education,furniture/equipment 21   4 0 (0.809523
8 0.1904762) *
##           25) Purpose=business,domestic appliances,radio/TV,repairs 19   6
1 (0.3157895 0.6842105) *
##         13) Age>=36.5 23   4 1 (0.1739130 0.8260870) *
##        7) Credit>=1296 71  14 1 (0.1971831 0.8028169)
##         14) Housing=free,rent 24   9 1 (0.3750000 0.6250000)
##           28) Credit>=2884 9   3 0 (0.6666667 0.3333333) *
##           29) Credit< 2884 15   3 1 (0.2000000 0.8000000) *
##         15) Housing=own 47   5 1 (0.1063830 0.8936170) *
```

node), split, n, loss, yval, (yprob)

split- variable name & condition

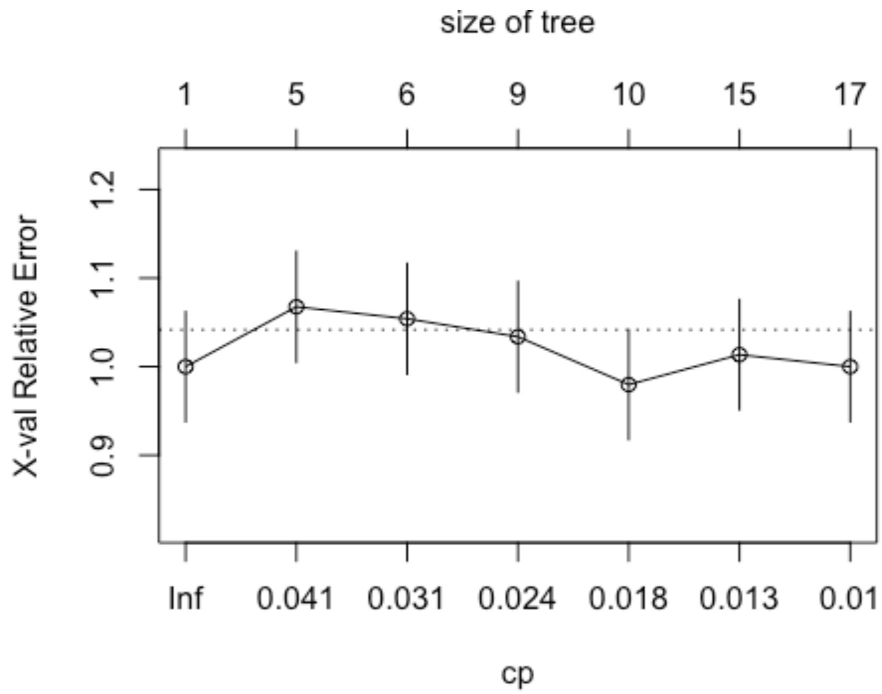
n - number of instances on the node

loss - instances predicted in the wrong class yval - predicted class

yprob - probability of being in each class [(x,y) first num x corresponds to the class predicted in the first node]

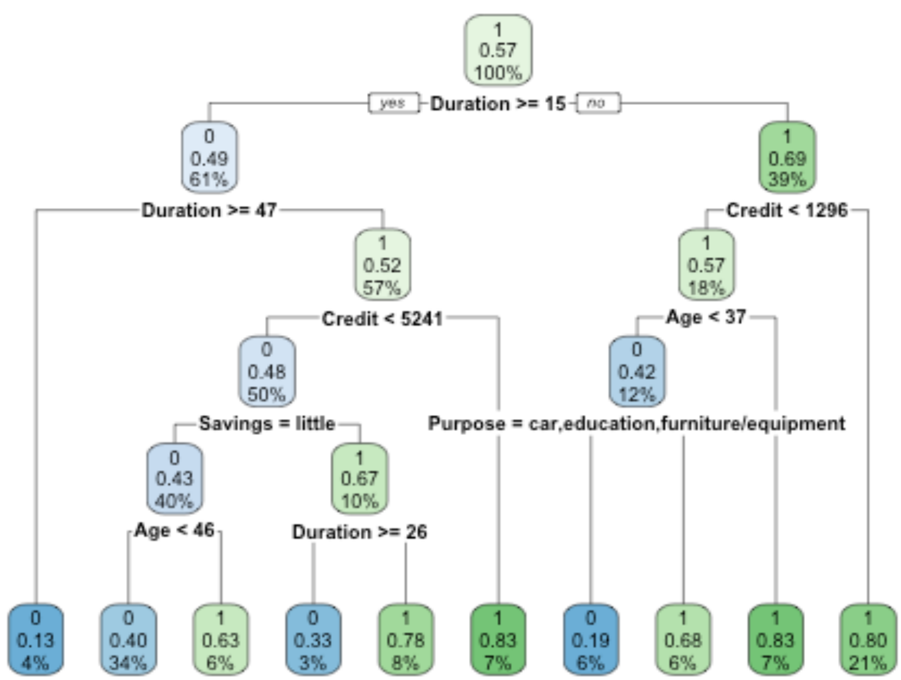
asteriks - terminal nodes

```
tree_pred_class <- predict(tree_model, test, type = "class")
table(tree_pred_class, test$Target)

##
## tree_pred_class  0  1
##               0 18 26
##               1 31 61

accuracy(tree_pred_class, test$Target)

## [1] 0.5808824

#To find best value of CP we can use printcp(), plotcp(), or summary() and ch
oose CP with minimun xerror
printcp(tree_model)

##
## Classification tree:
## rpart(formula = Target ~ ., data = train)
##
## Variables actually used in tree construction:
## [1] Age      Credit   Duration Housing  ID       Purpose  Savings
##
## Root node error: 148/345 = 0.42899
##
## n= 345
##
##         CP nsplit rel error  xerror     xstd
## 1 0.050676      0   1.00000 1.00000 0.062114
## 2 0.033784      4   0.79730 1.06757 0.062529
## 3 0.029279      5   0.76351 1.05405 0.062463
## 4 0.020270      8   0.67568 1.03378 0.062348
## 5 0.015766      9   0.65541 0.97973 0.061948
## 6 0.010135     14   0.56081 1.01351 0.062215
## 7 0.010000     16   0.54054 1.00000 0.062114

plotcp(tree_model)
```



Tree with best CP value and minimum prediction error

```
optimal <- which.min(tree_model$cptable[ ,"xerror"])
cp <- tree_model$cptable[optimal, "CP"]
tree_pruned <- prune(tree_model, cp = cp)
rpart.plot(tree_pruned)
```



```
tree_pruned_pred_class <- predict(tree_pruned, test, type = "class")
table(tree_pruned_pred_class, test$Target)
```

```
##
## tree_pruned_pred_class  0  1
##                      0 24 30
##                      1 25 57
```

```
accuracy(tree_pruned_pred_class, test$Target)
```

```
## [1] 0.5955882
```

We can see accuracy increased by using best CP value

SVM Model

First model with our selected values of cost and gamma

```
library(e1071)
```

```
svmModel<- svm(Target ~ . , data=train, type="C-classification", cost=100)
summary(svmModel)
```

```
##
## Call:
## svm(formula = Target ~ ., data = train, type = "C-classification",
##     cost = 100)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  239
##
##  ( 112 127 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
preds <- predict(svmModel, newdata=test)
preds
```

```
##   4   8  10  13  16  28  33  35  39  42  52  55  60  68  95 102 112 113 12
6 128
##   1   1   1   1   0   1   1   1   1   1   1   0   1   1   1   0   0   1
1   0
## 129 130 138 141 157 164 167 168 180 192 195 196 204 213 214 219 240 258 26
```

```
2 274
##   0   1   1   0   0   0   1   1   1   1   0   0   0   0   1   1   0   1
0   0
## 285 287 300 310 313 339 345 347 356 363 368 389 392 397 399 426 439 458 46
2 463
##   1   0   1   0   1   1   1   0   0   1   0   1   0   1   0   0   0   1
0   0
## 467 471 480 501 503 513 516 517 519 530 538 539 540 544 557 563 566 567 57
5 596
##   0   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0   0
1   0
## 597 598 601 605 606 611 612 628 635 640 645 647 651 656 660 665 667 691 70
4 710
##   0   1   1   1   0   1   0   1   0   0   1   0   1   1   0   1   1   1
1   1
## 721 728 731 738 748 753 757 763 767 775 780 789 791 794 812 814 816 824 83
6 851
##   1   0   1   0   0   0   1   1   0   1   1   0   1   0   1   0   0   0
0   0
## 875 886 891 897 924 926 927 930 951 952 953 958 976 980 986 999
##   1   1   1   0   1   0   1   0   1   1   0   1   1   1   1   0
## Levels: 0 1
```

```r
table(test$Target, preds)
```

```
##    preds
##      0  1
##   0 28 21
##   1 30 57
```

```r
accuracy(test$Target, preds)
```

```
## [1] 0.625
```

type = “C-classification” - binary classification

cost - cost of misclassification:

- if high-> not many misclassified points, margin can be small
- if low -> make more mistakes, margin is larger

Gamma defines how far the influence of a single training example reach.(influence of points either near or far away from the hyperplane.)

- Higher value of gamma → every point has close reach data → chance of overfitting, decision boundary looks wiggly.
- Low value of gamma → every point has far reach data → decision boundary looks smoother

to find best cost and gamma values, use cross validation

This function uses cv to find best value of gamma

Gamma in SVM is usually a value between 0 and 1

```
tunesvm <- tune(svm, Target ~. , data = train, kernel="radial", ranges = list
(gamma = seq(.01, 0.1, by = .01), cost = 100, tunecontrol = tune.control(nrep
eat = 10, sampling = "cross", cross = 10)))
tunesvm

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost tunecontrol
##   0.01  100       FALSE
##
## - best performance: 0.3911765
```

SVM Model using best gamma value obtained by doing CV

```
SVMmodel_tuned<- svm(Target~., data = train, type="C-classification" , cost =
100, gamma=0.01)
predicted_svm <- predict(SVMmodel_tuned, newdata = test)

accuracy(test$Target, predicted_svm)

## [1] 0.6691176

recall <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  recall <- (y["TP"] /  (y["TP"]+ y["FN"]))
  return(as.numeric(recall))
}

 recall(test$Target, predicted_svm)

## [1] 0.7701149

 precision <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  precision <- (y["TP"] / (y["TP"]+ y["FP"]))
  return(as.numeric(precision))
}

 precision(test$Target, predicted_svm)
```

```
## [1] 0.7282609

 accuracy<- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  accuracy <- (y["TN"] + y["TP"])/ sum(y)
  return(as.numeric(accuracy))
}
 accuracy(test$Target, predicted_svm)

## [1] 0.6691176
```

NAIVE BAYES MODEL

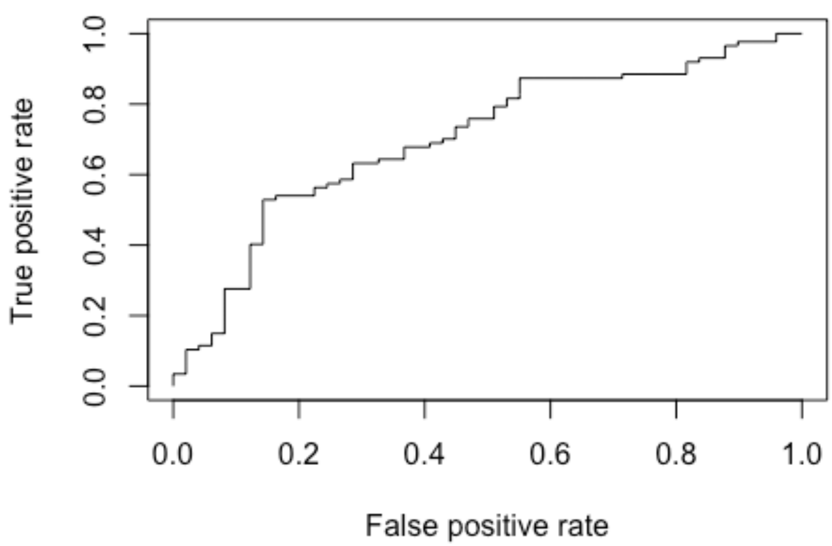
```
naive_model <- naiveBayes(Target ~ ., data= train)
naive_pred_class <- predict(naive_model, test, type="class", laplace =1)
naive_pred_class

##   [1] 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1
1 0 1
##  [38] 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1
0 1 1
##  [75] 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1
0 1 1
## [112] 0 1 1 1 0 0 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0
## Levels: 0 1

naive_pred_prob <- predict(naive_model, test, type="raw")

#confusion matrix
table(naive_pred_class, test$Target, dnn= c("Prediction", "Actual"))

##             Actual
## Prediction   0  1
##            0 19 17
##            1 30 70

accuracy(naive_pred_class, test$Target)

## [1] 0.6544118
```

Evaluation of Models using ROC Curves

```
library(ROCR)

# 2 main functions: prediction & performance
# prediction(True Labels, Predicted Probabilities for positive class)
```

 # ROC Curve for SVM model

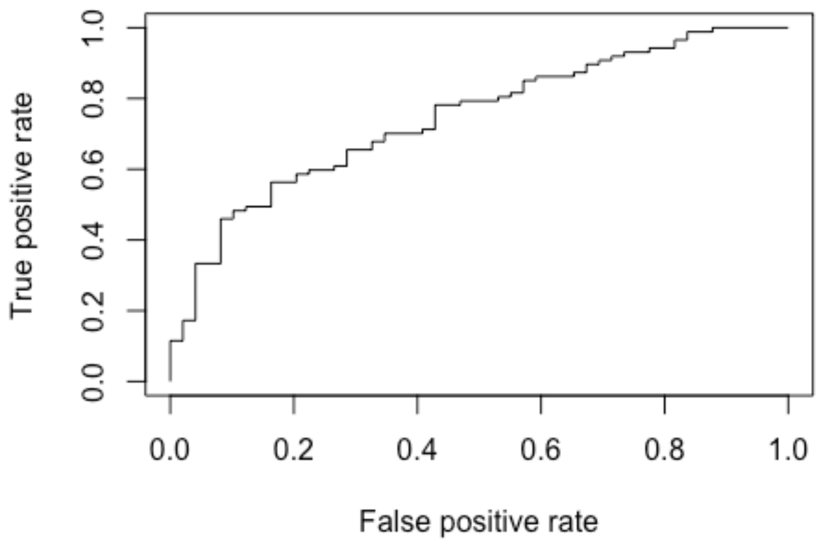
```
svmModel1<- svm(Target ~ . , data=train, probability = TRUE, type="C-classifi
cation", gamma=0.01, cost=100, decision.values = TRUE)
pred_svm <- predict(svmModel1, newdata=test, probability = TRUE, decision.val
ues = TRUE)
# returns predicted class, and probabilities of belonging on each class
pred_prob_svm <- attr(pred_svm, "probabilities")
# store results of the probabilities of being in each class
pred_prob_svm_good <- pred_prob_svm[,2]
# store prob of ONLY being in class 1 - Good
pred <- prediction(pred_prob_svm_good, test$Target)

#ROC CURVE CHART
perf_roc_svm <- performance(pred, "tpr", "fpr")
plot(perf_roc_svm)
```




```
#AUC
auc_svm <- performance(pred, "auc")
auc_svm <- unlist(slot(auc_svm, "y.values"))
auc_svm

## [1] 0.7037298
```

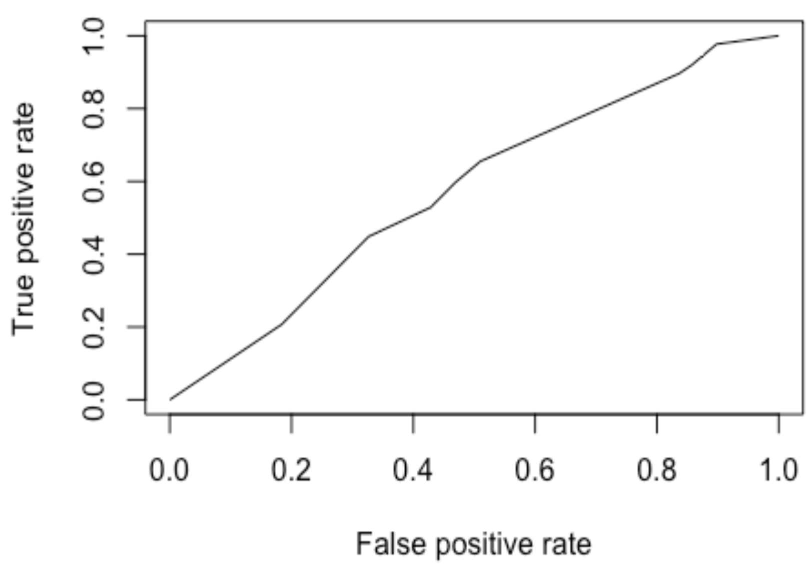
 # ROC Curve for Logistic Regression Model

```
pred_lr <- prediction(predictions, test$Target)
perf_roc_lr <- performance(pred_lr, "tpr", "fpr")
plot(perf_roc_lr)
```




```
#AUC
auc_LR <- performance(pred_lr, "auc")
auc_LR <- unlist(slot(auc_LR, "y.values"))
auc_LR

## [1] 0.7447807
```

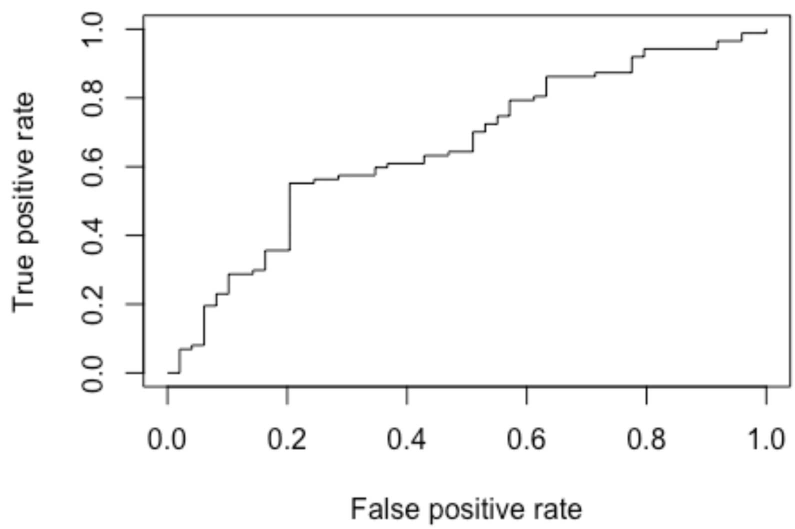
```
# ROC Curve for Decision Tree Model
tree_pruned_pred_probs <- predict(tree_pruned, test)
#probability of being in each class
tree_pruned_pred_probs_positive <- tree_pruned_pred_probs[,2]
#probability of being in class 1-Good

pred_decision_tree <-prediction(tree_pruned_pred_probs_positive, test$Target)
perf_roc_decision_tree <- performance(pred_decision_tree, "tpr", "fpr")
plot(perf_roc_decision_tree)
```



```r
#AUC
auc_DT <- performance(pred_decision_tree, "auc" )
auc_DT <- unlist(slot(auc_DT, "y.values"))
auc_DT

## [1] 0.575651
```

 # ROC Curve for Naive Bayes Model

```r
naive_pred_prob_yes <- naive_pred_prob[,2]
pred_nb <- prediction(naive_pred_prob_yes, test$Target)
perf_roc_nb <- performance(pred_nb, "tpr", "fpr")
plot(perf_roc_nb)
```




```r
#AUC
auc_NB <- performance(pred_nb, "auc")
auc_NB <- unlist(slot(auc_NB, "y.values"))
auc_NB

## [1] 0.6572836
```

Overall, our Logistic Regression Model is performing better than SVM, DT, and Naive Bayes.

The accuracy of the Logistic Regression Model: 0.6838235, AUC: 0.7447807

The accuracy of the SVM: 0.6691176 , AUC: 0.7037298

The accuracy of the DT: 0.5955882 , AUC:0.575651

The accuracy of the NB: 0.6544118 , AUC:0.6572836