# Decision Tree Tutorial Part 2 - German Credit Data

*John Kelly*

*11 February 2019*

## Lab 5a - Decision Trees

The German Credit Data contains data on 20 variables and the classification whether an applicant is considered a Good or a Bad credit risk for 1000 loan applicants. The objective of the model is whether to approve a loan to a prospective applicant based on his/her profiles.

Make sure all the categorical variables are converted into factors.

1.The function rpart will run a regression tree if the response variable is numeric, and a classification tree if it is a factor

2.rpart parameter - Method - "class" for a classification tree ; "anova" for a regression tree

3.minsplit : minimum number of observations in a node before splitting. Default value - 20

4.minbucket : minimum number of observations in terminal node (leaf). Default value - 7 (i.e. minsplit/3)

5.xval : Number of cross validations

6.Prediction (Scoring) : If type = "prob": This is for a classification tree. It generates probabilities - Prob(Y=0) and Prob(Y=1).

7.Prediction (Classification) : If type = "class": This is for a classification tree. It returns 0/1.

## Step 1: Read dataset into RStudio

```r
#read data file
mydata=read.csv("german_credit.csv")
```

```r
summary(mydata)
```

```
##  Creditability Account.Balance Duration.of.Credit..month.
##  Min.   :0.0   Min.   :1.000   Min.   : 4.0
##  1st Qu.:0.0   1st Qu.:1.000   1st Qu.:12.0
##  Median :1.0   Median :2.000   Median :18.0
##  Mean   :0.7   Mean   :2.577   Mean   :20.9
##  3rd Qu.:1.0   3rd Qu.:4.000   3rd Qu.:24.0
##  Max.   :1.0   Max.   :4.000   Max.   :72.0
##  Payment.Status.of.Previous.Credit    Purpose       Credit.Amount
##  Min.   :0.000                     Min.   : 0.000   Min.   :  250
##  1st Qu.:2.000                     1st Qu.: 1.000   1st Qu.: 1366
##  Median :2.000                     Median : 2.000   Median : 2320
##  Mean   :2.545                     Mean   : 2.828   Mean   : 3271
##  3rd Qu.:4.000                     3rd Qu.: 3.000   3rd Qu.: 3972
##  Max.   :4.000                     Max.   :10.000   Max.   :18424
##  Value.Savings.Stocks Length.of.current.employment Instalment.per.cent
##  Min.   :1.000        Min.   :1.000                Min.   :1.000
##  1st Qu.:1.000        1st Qu.:3.000                1st Qu.:2.000
##  Median :1.000        Median :3.000                Median :3.000
##  Mean   :2.105        Mean   :3.384                Mean   :2.973
##  3rd Qu.:3.000        3rd Qu.:5.000                3rd Qu.:4.000
```

```
##  Max.   :5.000        Max.   :5.000                Max.   :4.000
##  Sex...Marital.Status   Guarantors    Duration.in.Current.address
##  Min.   :1.000        Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000        1st Qu.:1.000   1st Qu.:2.000
##  Median :3.000        Median :1.000   Median :3.000
##  Mean   :2.682        Mean   :1.145   Mean   :2.845
##  3rd Qu.:3.000        3rd Qu.:1.000   3rd Qu.:4.000
##  Max.   :4.000        Max.   :3.000   Max.   :4.000
##  Most.valuable.available.asset  Age..years.    Concurrent.Credits
##  Min.   :1.000                 Min.   :19.00   Min.   :1.000
##  1st Qu.:1.000                 1st Qu.:27.00   1st Qu.:3.000
##  Median :2.000                 Median :33.00   Median :3.000
##  Mean   :2.358                 Mean   :35.54   Mean   :2.675
##  3rd Qu.:3.000                 3rd Qu.:42.00   3rd Qu.:3.000
##  Max.   :4.000                 Max.   :75.00   Max.   :3.000
##  Type.of.apartment No.of.Credits.at.this.Bank   Occupation
##  Min.   :1.000     Min.   :1.000              Min.   :1.000
##  1st Qu.:2.000     1st Qu.:1.000              1st Qu.:3.000
##  Median :2.000     Median :1.000              Median :3.000
##  Mean   :1.928     Mean   :1.407              Mean   :2.904
##  3rd Qu.:2.000     3rd Qu.:2.000              3rd Qu.:3.000
##  Max.   :3.000     Max.   :4.000              Max.   :4.000
##  No.of.dependents   Telephone     Foreign.Worker
##  Min.   :1.000    Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000    1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000    Median :1.000   Median :1.000
##  Mean   :1.155    Mean   :1.404   Mean   :1.037
##  3rd Qu.:1.000    3rd Qu.:2.000   3rd Qu.:1.000
##  Max.   :2.000    Max.   :2.000   Max.   :2.000
```

```r
# Check attributes of data
str(mydata)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ Creditability                    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Account.Balance                  : int  1 1 2 1 1 1 1 1 4 2 ...
##  $ Duration.of.Credit..month.       : int  18 9 12 12 12 10 8 6 18 24 ...
##  $ Payment.Status.of.Previous.Credit: int  4 4 2 4 4 4 4 4 4 2 ...
##  $ Purpose                          : int  2 0 9 0 0 0 0 0 3 3 ...
##  $ Credit.Amount                    : int  1049 2799 841 2122 2171 2241 3398 1361 1098 3758 ...
##  $ Value.Savings.Stocks             : int  1 1 2 1 1 1 1 1 1 3 ...
##  $ Length.of.current.employment     : int  2 3 4 3 3 2 4 2 1 1 ...
##  $ Instalment.per.cent              : int  4 2 2 3 4 1 1 2 4 1 ...
##  $ Sex...Marital.Status             : int  2 3 2 3 3 3 3 3 2 2 ...
##  $ Guarantors                       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Duration.in.Current.address      : int  4 2 4 2 4 3 4 4 4 4 ...
##  $ Most.valuable.available.asset    : int  2 1 1 1 2 1 1 1 3 4 ...
##  $ Age..years.                      : int  21 36 23 39 38 48 39 40 65 23 ...
##  $ Concurrent.Credits               : int  3 3 3 3 1 3 3 3 3 3 ...
##  $ Type.of.apartment                : int  1 1 1 1 2 1 2 2 2 1 ...
##  $ No.of.Credits.at.this.Bank       : int  1 2 1 2 2 2 2 1 2 1 ...
##  $ Occupation                       : int  3 3 2 2 2 2 2 2 1 1 ...
##  $ No.of.dependents                 : int  1 2 1 2 1 2 1 2 1 1 ...
##  $ Telephone                        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Foreign.Worker                   : int  1 1 1 2 2 2 2 2 1 1 ...
```

### Step 2:Check number of rows and columns

```
# Check number of rows and columns
dim(mydata)
```

```
## [1] 1000   21
```

### Step3:Make dependent variable a factor

```
# Make dependent variable as a factor (categorical)
mydata$Creditability = as.factor(mydata$Creditability)
```

### Step 4:Splt data in training and test datasets

```
# Split data into training (70%) and validation (30%)
dt = sort(sample(nrow(mydata), nrow(mydata)*.7))
train<-mydata[dt,]
val<-mydata[-dt,] # Check number of rows in training data set
nrow(train)
```

```
## [1] 700
```

### Step 5:View dataset

```
# To view dataset
edit(train)
```

### Step 6:Prepare and run Decision Tree

```
# Decision Tree Model
library(rpart)
mtree <- rpart(Creditability~., data = train, method="class", control = rpart.control(minsplit = 20, mi
```

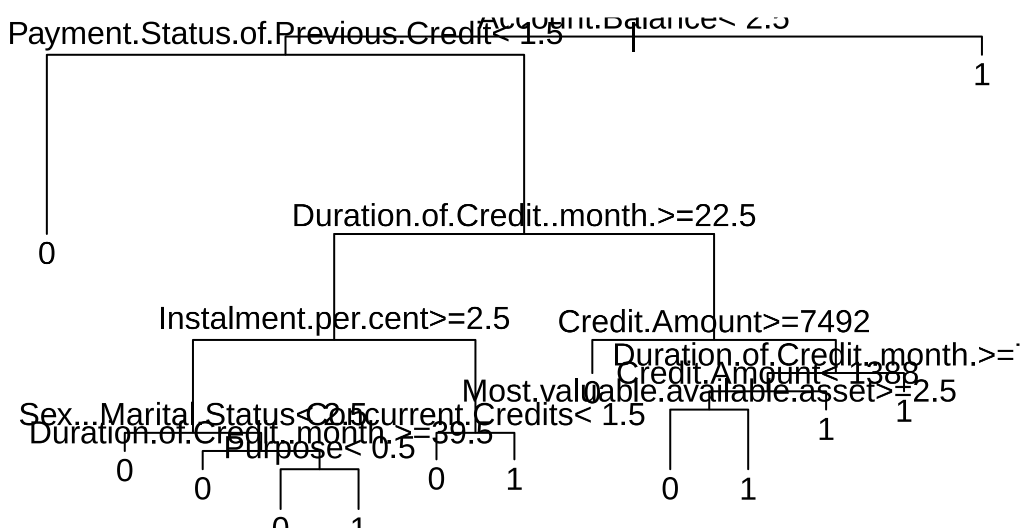
```
mtree
```

```
## n= 700
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 700 217 1 (0.31000000 0.69000000)
##     2) Account.Balance< 2.5 386 175 1 (0.45336788 0.54663212)
##       4) Payment.Status.of.Previous.Credit< 1.5 51  12 0 (0.76470588 0.23529412) *
##       5) Payment.Status.of.Previous.Credit>=1.5 335 136 1 (0.40597015 0.59402985)
##        10) Duration.of.Credit..month.>=22.5 142  63 0 (0.55633803 0.44366197)
##          20) Instalment.per.cent>=2.5 90  30 0 (0.66666667 0.33333333)
##            40) Sex...Marital.Status< 2.5 28   4 0 (0.85714286 0.14285714) *
##            41) Sex...Marital.Status>=2.5 62  26 0 (0.58064516 0.41935484)
##              82) Duration.of.Credit..month.>=39.5 17   3 0 (0.82352941 0.17647059) *
##              83) Duration.of.Credit..month.< 39.5 45  22 1 (0.48888889 0.51111111)
##               166) Purpose< 0.5 10   2 0 (0.80000000 0.20000000) *
##               167) Purpose>=0.5 35  14 1 (0.40000000 0.60000000) *
##          21) Instalment.per.cent< 2.5 52  19 1 (0.36538462 0.63461538)
##            42) Concurrent.Credits< 1.5 10   3 0 (0.70000000 0.30000000) *
```

```
##            43) Concurrent.Credits>=1.5 42  12 1 (0.28571429 0.71428571) *
##        11) Duration.of.Credit..month.< 22.5 193  57 1 (0.29533679 0.70466321)
##          22) Credit.Amount>=7491.5 7   1 0 (0.85714286 0.14285714) *
##          23) Credit.Amount< 7491.5 186  51 1 (0.27419355 0.72580645)
##            46) Duration.of.Credit..month.>=7.5 159  49 1 (0.30817610 0.69182390)
##              92) Credit.Amount< 1387.5 62  27 1 (0.43548387 0.56451613)
##               184) Most.valuable.available.asset>=2.5 13   2 0 (0.84615385 0.15384615) *
##               185) Most.valuable.available.asset< 2.5 49  16 1 (0.32653061 0.67346939) *
##              93) Credit.Amount>=1387.5 97  22 1 (0.22680412 0.77319588) *
##            47) Duration.of.Credit..month.< 7.5 27   2 1 (0.07407407 0.92592593) *
##     3) Account.Balance>=2.5 314  42 1 (0.13375796 0.86624204) *
```

### Step 7:Plot the Trees

```r
#Plot tree
plot(mtree)
text(mtree)
```

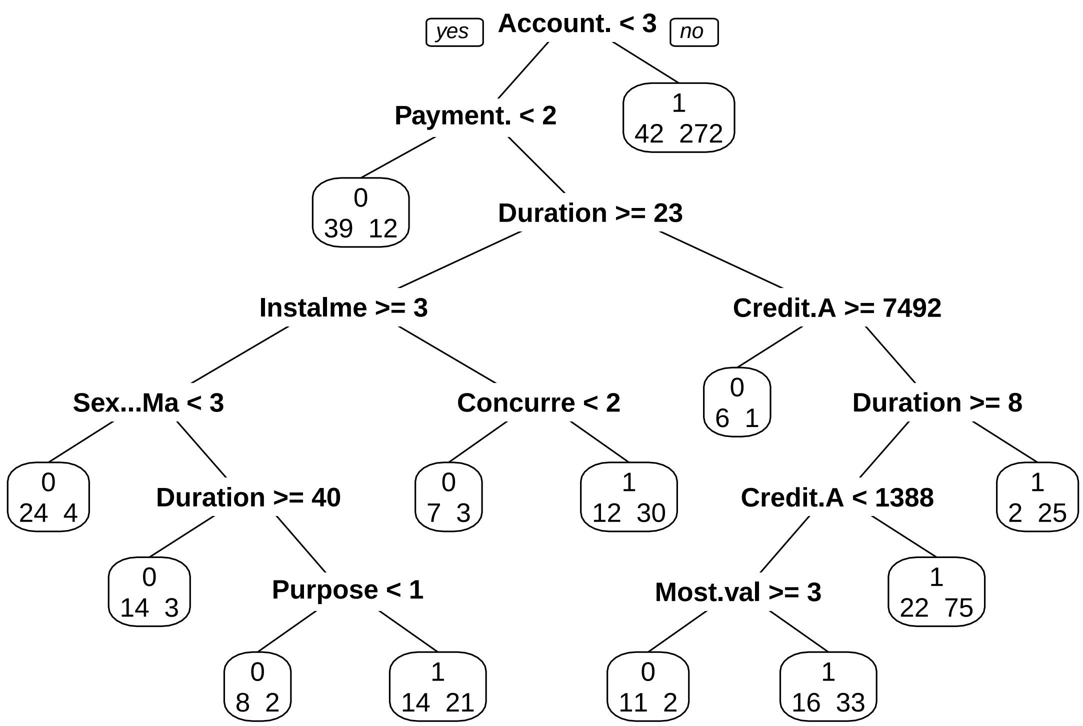



```r
#Beautify tree
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

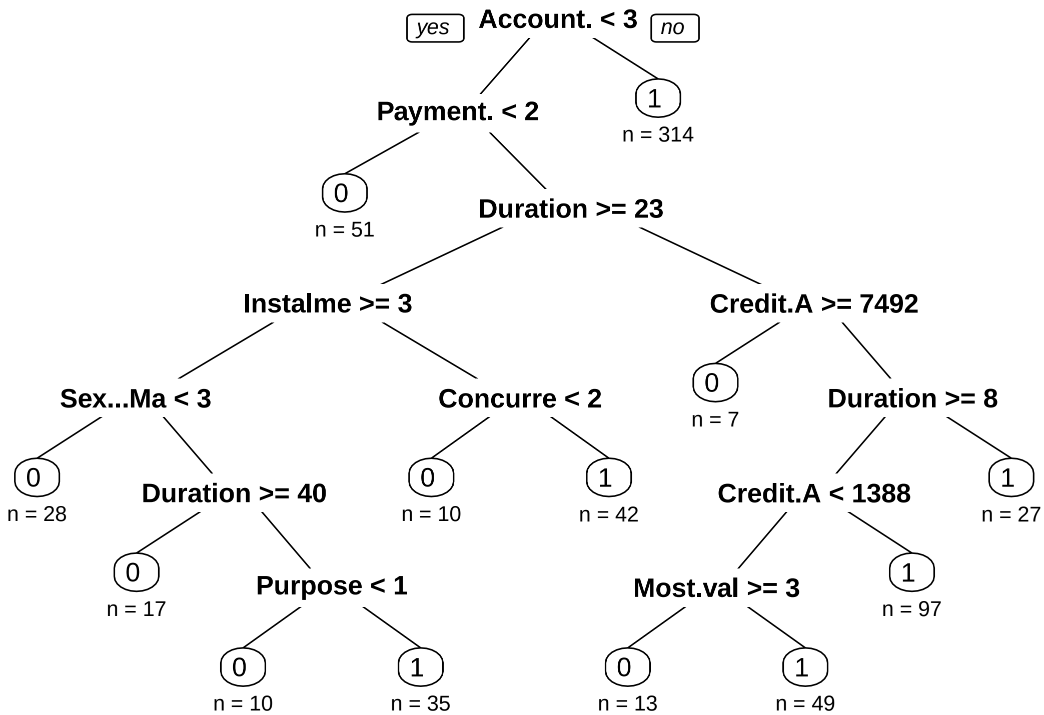
```r
library(rpart.plot)
library(RColorBrewer)
```

```r
#view1
prp(mtree, faclen = 0, cex = 0.8, extra = 1)
```



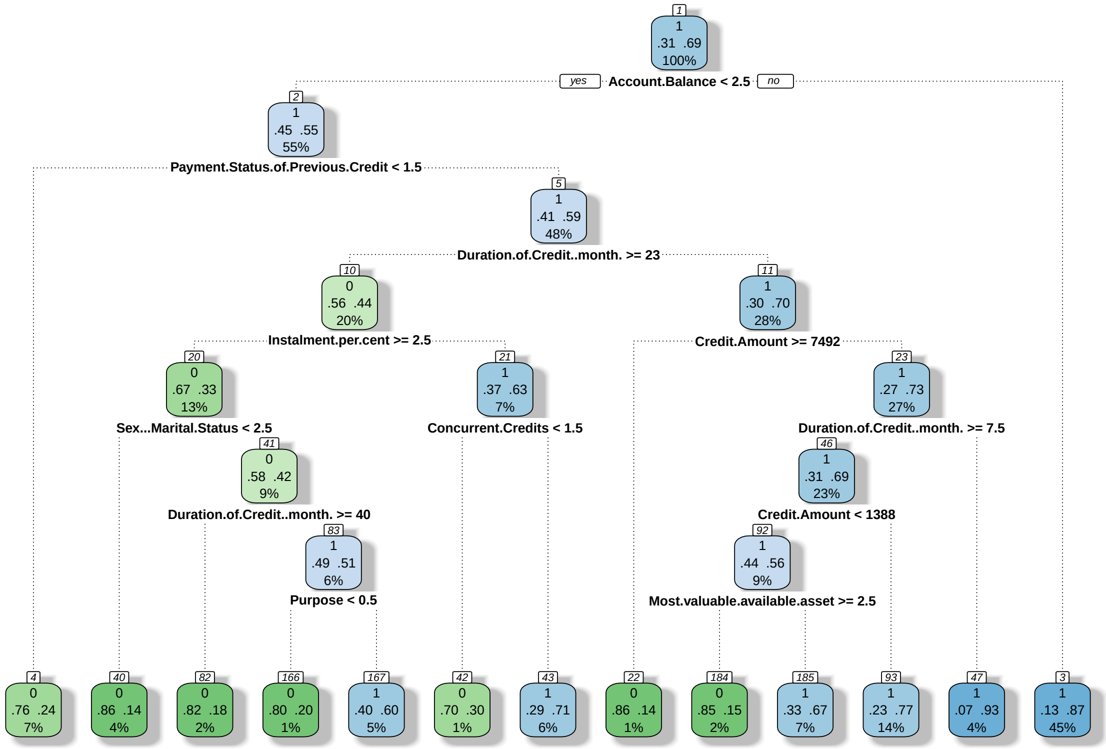

More plots

```r
#view2 - total count at each node
tot_count <- function(x, labs, digits, varlen)
{paste(labs, "\n\nn =", x$frame$n)}

prp(mtree, faclen = 0, cex = 0.8, node.fun=tot_count)
```




More Plots

```
#view3- fancy Plot
library(rattle)
#library(gKt)
#rattle()
fancyRpartPlot(mtree)
```



Rattle 2019−Feb−11 17:20:37 jpkel

### Step 8:Pruning

```
printcp(mtree)
```

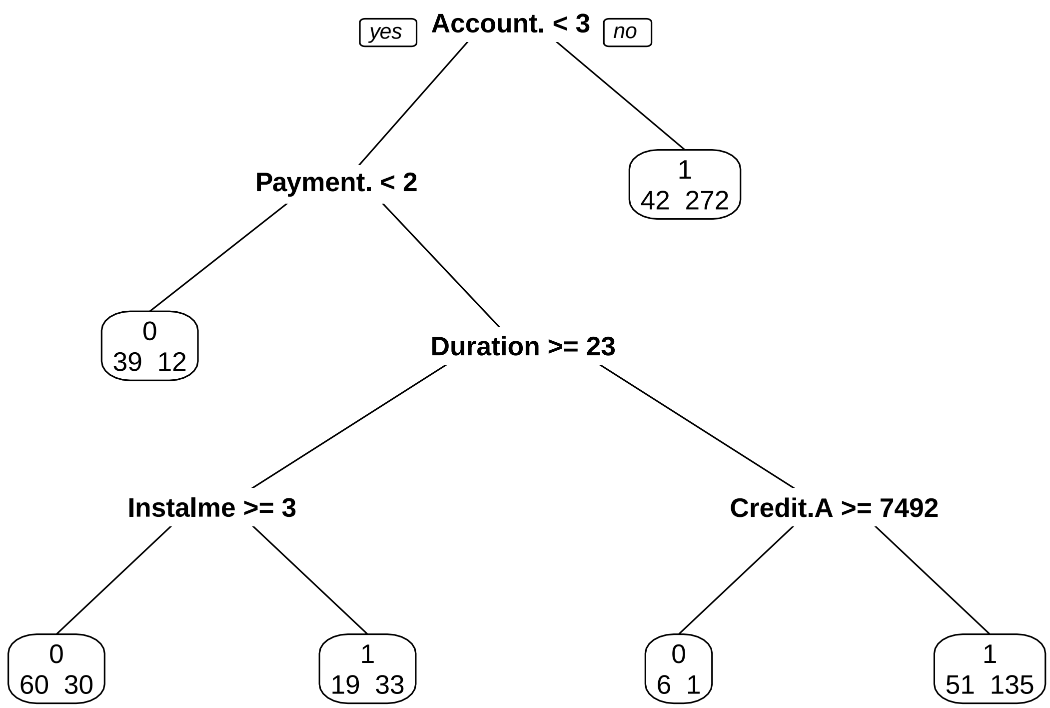
```
##
## Classification tree:
## rpart(formula = Creditability ~ ., data = train, method = "class",
##     control = rpart.control(minsplit = 20, minbucket = 7, maxdepth = 10,
##         usesurrogate = 2, xval = 10))
##
## Variables actually used in tree construction:
## [1] Account.Balance                   Concurrent.Credits
## [3] Credit.Amount                     Duration.of.Credit..month.
## [5] Instalment.per.cent               Most.valuable.available.asset
## [7] Payment.Status.of.Previous.Credit Purpose
## [9] Sex...Marital.Status
##
## Root node error: 217/700 = 0.31
##
## n= 700
##
##         CP nsplit rel error  xerror     xstd
## 1 0.062212      0   1.00000 1.00000 0.056389
## 2 0.023041      4   0.73733 0.92166 0.055080
## 3 0.018433      5   0.71429 0.91244 0.054913
## 4 0.013825      6   0.69585 0.94931 0.055563
```

```
## 5 0.010753      9   0.65438 0.97235 0.055948
## 6 0.010000     12   0.62212 0.94931 0.055563
```

```
bestcp <- mtree$cptable[which.min(mtree$cptable[,"xerror"]),"CP"]

# Prune the tree using the best cp.
pruned <- prune(mtree, cp = bestcp)
```

### Step 9:PLot the Pruned Tree

```
# Plot pruned tree
prp(pruned, faclen = 0, cex = 0.8, extra = 1)
```




### Step 10:Confusion Matrix

```
# confusion matrix (training data)
conf.matrix <- table(train$Creditability, predict(pruned,type="class"))
rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = ":")
colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = ":")
print(conf.matrix)
```

```
##
##             Pred:0 Pred:1
##   Actual:0    105    112
##   Actual:1     43    440
```

### Step 11:Evaluate the model

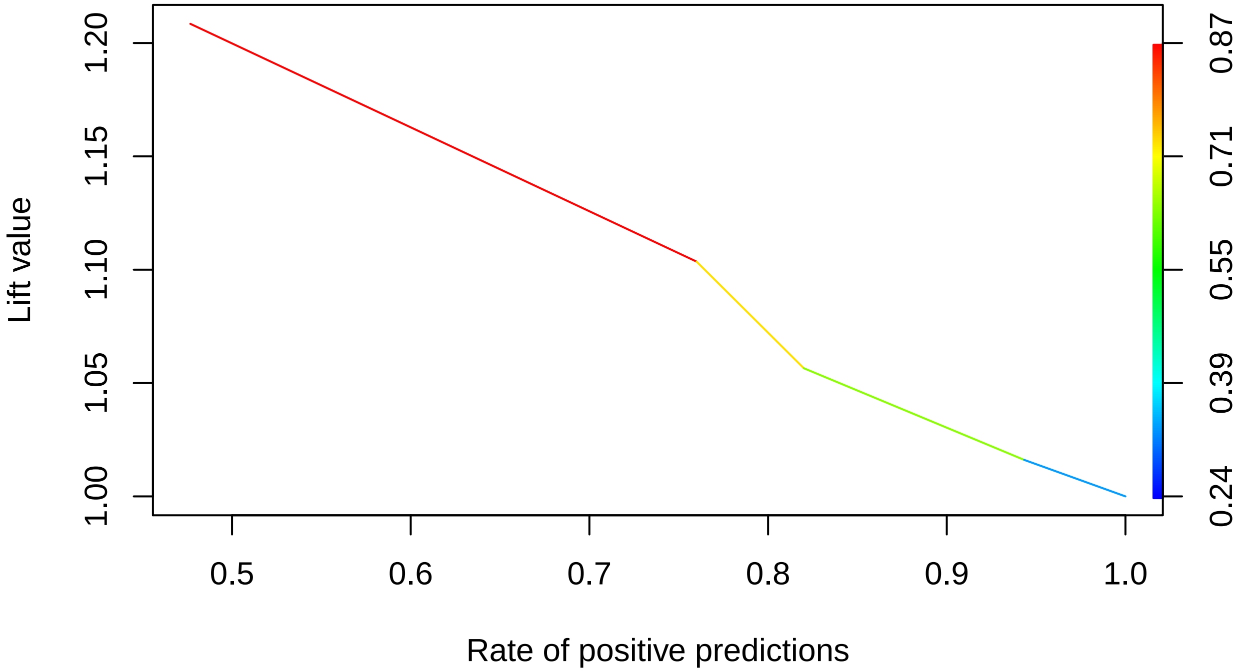
```
#Scoring
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

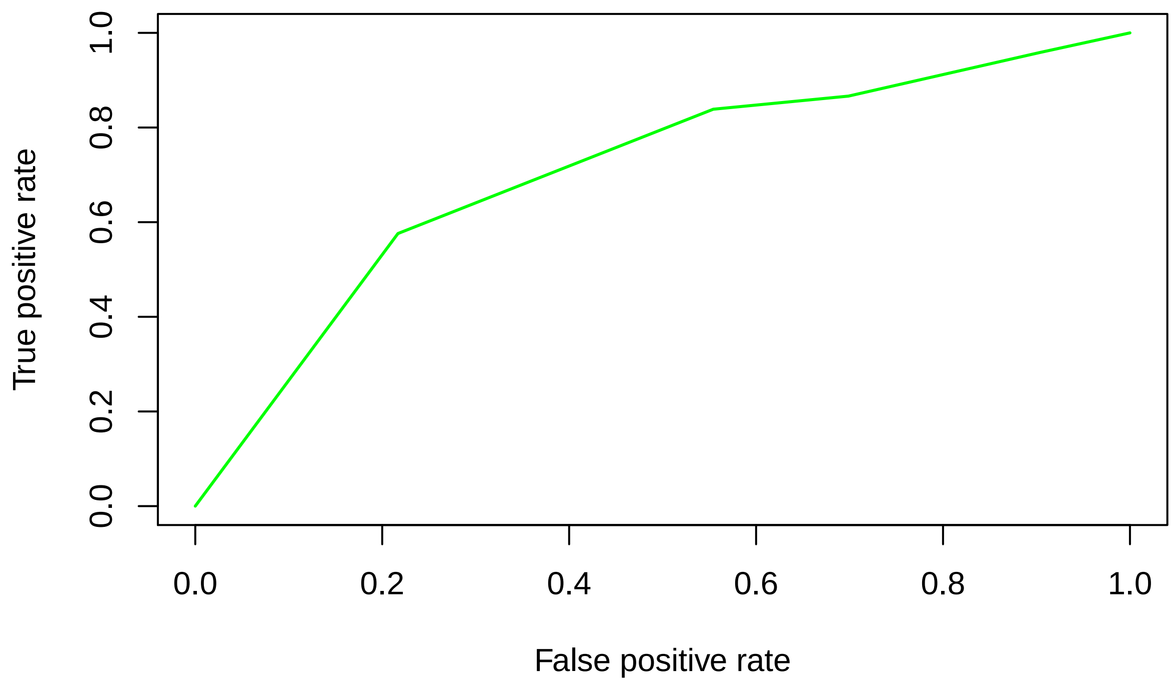
```
val1 = predict(pruned, val, type = "prob")
#Storing Model Performance Scores
pred_val <-prediction(val1[,2],val$Creditability)

# Calculating Area under Curve
perf_val <- performance(pred_val,"auc")
perf_val
```

```
## 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] 0.7056243
##
##
## Slot "alpha.values":
## list()
```

```
# Plotting Lift curve
plot(performance(pred_val, measure="lift", x.measure="rpp"), colorize=TRUE)
```




```r
# Calculating True Positive and False Positive Rate
perf_val <- performance(pred_val, "tpr", "fpr")

# Plot the ROC curve
plot(perf_val, col = "green", lwd = 1.5)
```




```
#Calculating KS statistics
ks1.tree <- max(attr(perf_val, "y.values")[[1]] - (attr(perf_val, "x.values")[[1]]))
ks1.tree
```

```
## [1] 0.3591694
```

### Step 12: Fancy Plot

```
# Advanced Plot
prp(pruned, main="Beautiful Tree",
    extra=106,
    nn=TRUE,
    fallen.leaves=TRUE,
    branch=.5,
    faclen=0,
    trace=1,
    shadow.col="gray",
    branch.lty=3,
    split.cex=1.2,
    split.prefix="is ",
    split.suffix="?",
    split.box.col="lightgray",
    split.border.col="darkgray",
    split.round=.5)
```

```
## cex 1   xlim c(0, 1)   ylim c(0, 1)
```

## Beautiful Tree