Assignment One

Muiz Murad

24/10/2019

Introduction

This report contains the cleaning, training and testing of the Portuguese Telemarketing data. From this, relevant variables are used to predict the likelihood that a loan customer will defult. A decision tree is constructed using the three packages; tree, rpart and party. From these three models the best tree modelin conjuction with confusion matrices is used to determine the best model.

```
#Make Sure All Packages Are Installed And Ready To Use
install.packages("dplyr")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contril
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'dplyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
install.packages("caret")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contril
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
install.packages("caTools")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contril
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'caTools' successfully unpacked and MD5 sums checked
```

```
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(caTools)
installed.packages("rpart")
```

```
##      Package LibPath Version Priority Depends Imports LinkingTo Suggests
##      Enhances License License_is_FOSS License_restricts_use OS_type Archs
##      MD5sum NeedsCompilation Built
```

```r
library(rpart)
install.packages("rpart.plot")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'rpart.plot' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(rpart.plot)
install.packages("e1071")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'e1071' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(e1071)
install.packages("randomForest")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'randomForest' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
install.packages("tree")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'tree' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(tree)
install.packages("party")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'party' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```r
install.packages("partykit")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'partykit' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(partykit)
```

```
## Loading required package: libcoin

##
## Attaching package: 'partykit'

## The following objects are masked from 'package:party':
##
```

```
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal, varimp
```

```r
install.packages("ggplot2")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## Warning: package 'ggplot2' is in use and will not be installed
```

```r
library(ggplot2)
install.packages("xgboost")
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'xgboost' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\13599863\AppData\Local\Temp\RtmpukfKJm\downloaded_packages
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

Using Oracle as our database we will extract the necessary data using the ROracle pakage with User ID's and Passwords already defined.

```r
#Connect To SQL DataBase To Extract DataSet
library(ROracle)
```

```
## Loading required package: DBI
```

```r
UserID <- Sys.info()[["user"]]
Password <- Sys.info()[["user"]]

library(ROracle)
drv <- dbDriver("Oracle")

## Refer to Oracle Database Net Services Administator's Guide for
## details on connect string specification.
host <- "oracle.vittl.it.bond.edu.au"
port <- 1521
sid <- "inft320"
connect.string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=(PROTOCOL=tcp)(HOST=", host, ")(PORT=", port, "))",
  "(CONNECT_DATA=(SID=", sid, ")))", sep = "")

## Use username/password authentication.
con <- dbConnect(drv, username = "A13599863", password = "A13599863",
                 dbname = connect.string)
```

```r
## Run a SQL statement by creating first a resultSet object.
rs <- dbSendQuery(con, "select * from brucedba.BankMarketing")
```

After extracting the data from the database we can quickly view to check to see we have the correct data and variables.

```r
#We Now Extract DataPoints From The DataSet
#Extract All Rows
loan <- fetch(rs)
head(loan)
```

```
##   AGE         JOB MARITAL   EDUCATION DEFAULTCREDIT HOUSING LOAN   CONTACT
## 1  46  management married    basic.9y            no      no   no telephone
## 2  44    services married high.school            no     yes  yes telephone
## 3  51      admin.  single    basic.6y            no      no   no telephone
## 4  28    services married high.school            no     yes   no telephone
## 5  37 blue-collar married    basic.9y            no      no   no telephone
## 6  34  technician married high.school            no      no   no telephone
##   MONTH DAY_OF_WEEK DURATION CAMPAIGN PDAYS PREVIOUS    POUTCOME
## 1   may         mon      128        2   999        0 nonexistent
## 2   may         mon      107        1   999        0 nonexistent
## 3   may         mon      303        2   999        0 nonexistent
## 4   may         mon       81        1   999        0 nonexistent
## 5   may         mon      270        1   999        0 nonexistent
## 6   may         mon      228        1   999        0 nonexistent
##   EMP_VAR_RATE CONS_PRICE_IDX CONS_CONF_IDX EURIBOR3M NR_EMPLOYED  Y
## 1          1.1         93.994         -36.4     4.857        5191 no
## 2          1.1         93.994         -36.4     4.857        5191 no
## 3          1.1         93.994         -36.4     4.857        5191 no
## 4          1.1         93.994         -36.4     4.857        5191 no
## 5          1.1         93.994         -36.4     4.857        5191 no
## 6          1.1         93.994         -36.4     4.857        5191 no
```

```r
tail(loan)
```

```
##       AGE        JOB  MARITAL         EDUCATION DEFAULTCREDIT HOUSING LOAN
## 41183  26    student   single       high.school            no      no   no
## 41184  54 management  married       high.school            no      no   no
## 41185  36     admin.  married university.degree            no      no   no
## 41186  40     admin.  married       high.school            no     yes   no
## 41187  39     admin.   single university.degree            no     yes  yes
## 41188  33     admin. divorced       high.school            no      no   no
##         CONTACT MONTH DAY_OF_WEEK DURATION CAMPAIGN PDAYS PREVIOUS
## 41183  cellular   aug         wed       95        3   999        3
## 41184  cellular   aug         wed      171        3   999        1
## 41185  cellular   aug         wed      212        7     6        1
## 41186 telephone   aug         thu      173        6   999        1
## 41187 telephone   aug         fri       45        1   999        0
## 41188 telephone   aug         fri      107        1   999        0
##          POUTCOME EMP_VAR_RATE CONS_PRICE_IDX CONS_CONF_IDX EURIBOR3M
## 41183     failure         -1.7         94.027         -38.3     0.894
## 41184     failure         -1.7         94.027         -38.3     0.894
## 41185     success         -1.7         94.027         -38.3     0.894
## 41186     failure         -1.7         94.027         -38.3     0.891
## 41187 nonexistent         -1.7         94.027         -38.3      0.89
```

```
## 41188 nonexistent         -1.7         94.027         -38.3      0.89
##       NR_EMPLOYED  Y
## 41183      4991.6 no
## 41184      4991.6 no
## 41185      4991.6 no
## 41186      4991.6 no
## 41187      4991.6 no
## 41188      4991.6 no
```

As Duration is a varaible we determine after meeting the client, this will not be used in the plotting of the model and we will be required to drop this variable from the dataset.

```
#Clean DataSet And Remove Unecessary Variable(s)
clean_loan <- select(loan,-c(DURATION))
```

As these variables are defined as characters, we have to convert these factors to numeric to enable R to interept them correctly.

```
#Convert Factors to Numeric
factornames <- c("AGE","PDAYS","PREVIOUS","EMP_VAR_RATE","CONS_PRICE_IDX","CONS_CONF_IDX","EURIBOR3M","I

clean_loan[,factornames] <- lapply(factornames, function (x) as.numeric(as.character(clean_loan[,x])))
```

As these variables are defined as characters, we have to convert these factors to categorical to enable R to interept them correctly.

```
#Convert Factors to Categorical
factornames1 <- c("JOB","MARITAL","EDUCATION","DEFAULTCREDIT","HOUSING","LOAN","CONTACT","MONTH","DAY_OI

clean_loan[,factornames1] <- lapply(factornames1, function (x) as.factor(as.character(clean_loan[,x])))
```

This is just to confirm that the Characters have been converted correctly.

```
#Check To See That Characters Have Been Converted
summary(clean_loan)
```

```
##       AGE                 JOB           MARITAL
##  Min.   :17.00   admin.     :10422   divorced: 4612
##  1st Qu.:32.00   blue-collar: 9254   married :24928
##  Median :38.00   technician : 6743   single  :11568
##  Mean   :40.02   services   : 3969   unknown :   80
##  3rd Qu.:47.00   management : 2924
##  Max.   :98.00   retired    : 1720
##                  (Other)    : 6156
##                EDUCATION     DEFAULTCREDIT      HOUSING
##  university.degree  :12168   no     :32588   no     :18622
##  high.school        : 9515   unknown: 8597   unknown:  990
##  basic.9y           : 6045   yes    :    3   yes    :21576
##  professional.course: 5243
##  basic.4y           : 4176
##  basic.6y           : 2292
##  (Other)            : 1749
##       LOAN            CONTACT          MONTH       DAY_OF_WEEK
##  no     :33950   cellular :26144   may    :13769   fri:7827
##  unknown:  990   telephone:15044   jul    : 7174   mon:8514
##  yes    : 6248                     aug    : 6178   thu:8623
##                                    jun    : 5318   tue:8090
##                                    nov    : 4101   wed:8134
```

```
##                                    apr    : 2632
##                                    (Other): 2016
##     CAMPAIGN          PDAYS          PREVIOUS           POUTCOME
##  Min.   : 1.000   Min.   :  0.0   Min.   :0.000   failure    : 4252
##  1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.000   nonexistent:35563
##  Median : 2.000   Median :999.0   Median :0.000   success    : 1373
##  Mean   : 2.568   Mean   :962.5   Mean   :0.173
##  3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.000
##  Max.   :56.000   Max.   :999.0   Max.   :7.000
##
##   EMP_VAR_RATE      CONS_PRICE_IDX  CONS_CONF_IDX     EURIBOR3M
##  Min.   :-3.40000   Min.   :92.20   Min.   :-50.8   Min.   :0.634
##  1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7   1st Qu.:1.344
##  Median : 1.10000   Median :93.75   Median :-41.8   Median :4.857
##  Mean   : 0.08189   Mean   :93.58   Mean   :-40.5   Mean   :3.621
##  3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4   3rd Qu.:4.961
##  Max.   : 1.40000   Max.   :94.77   Max.   :-26.9   Max.   :5.045
##
##   NR_EMPLOYED     Y
##  Min.   :4964   no :36548
##  1st Qu.:5099   yes: 4640
##  Median :5191
##  Mean   :5167
##  3rd Qu.:5228
##  Max.   :5228
##
```

```r
str(clean_loan)
```

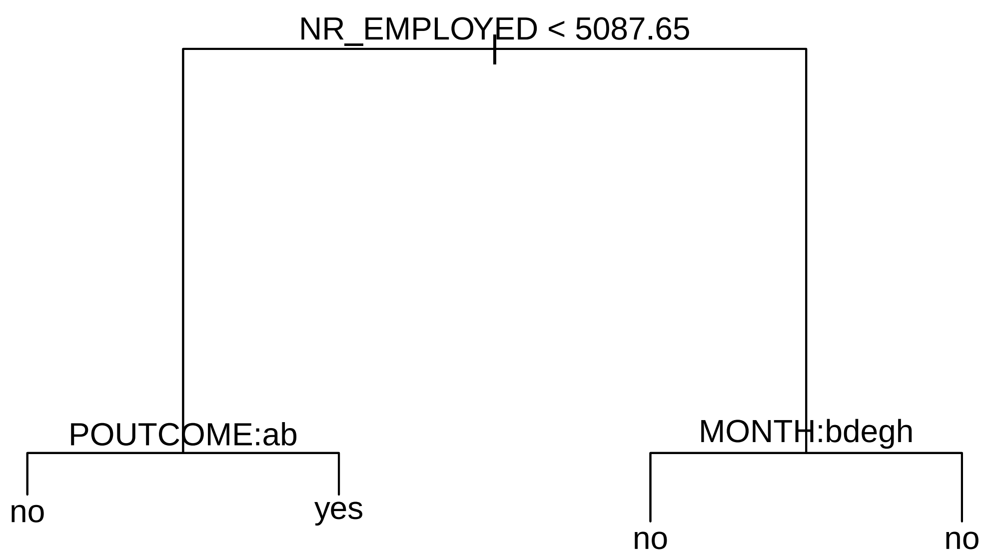
```
## 'data.frame':    41188 obs. of  20 variables:
##  $ AGE           : num  46 44 51 28 37 34 46 42 36 54 ...
##  $ JOB           : Factor w/ 12 levels "admin.","blue-collar",..: 5 8 1 8 2 10 2 2 1 2 ...
##  $ MARITAL       : Factor w/ 4 levels "divorced","married",..: 2 2 3 2 2 2 2 2 2 2 ...
##  $ EDUCATION     : Factor w/ 8 levels "basic.4y","basic.6y",..: 3 4 2 4 3 4 2 3 7 1 ...
##  $ DEFAULTCREDIT : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 1 1 2 1 1 2 ...
##  $ HOUSING       : Factor w/ 3 levels "no","unknown",..: 1 3 1 3 1 1 3 3 3 1 ...
##  $ LOAN          : Factor w/ 3 levels "no","unknown",..: 1 3 1 1 1 1 1 3 1 1 ...
##  $ CONTACT       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
##  $ MONTH         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ DAY_OF_WEEK   : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ CAMPAIGN      : num  2 1 2 1 1 1 1 2 3 1 ...
##  $ PDAYS         : num  999 999 999 999 999 999 999 999 999 999 ...
##  $ PREVIOUS      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ POUTCOME      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ EMP_VAR_RATE  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ CONS_PRICE_IDX: num  94 94 94 94 94 ...
##  $ CONS_CONF_IDX : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ EURIBOR3M     : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ NR_EMPLOYED   : num  5191 5191 5191 5191 5191 ...
##  $ Y             : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1 ...
```

Split the data into training and test sets to ensure that we are able to have data to test the model with as the training data will also produce high accuracies as it has been used to create the model already. The test data acts as forgein data on the model to produce accuracies.

```r
#Split Data Into Train And Test Data
sample_data = sample.split(clean_loan, SplitRatio = 0.70)
datatrainset <- subset(clean_loan, sample_data == TRUE)
datatestset <- subset(clean_loan, sample_data == FALSE)
```

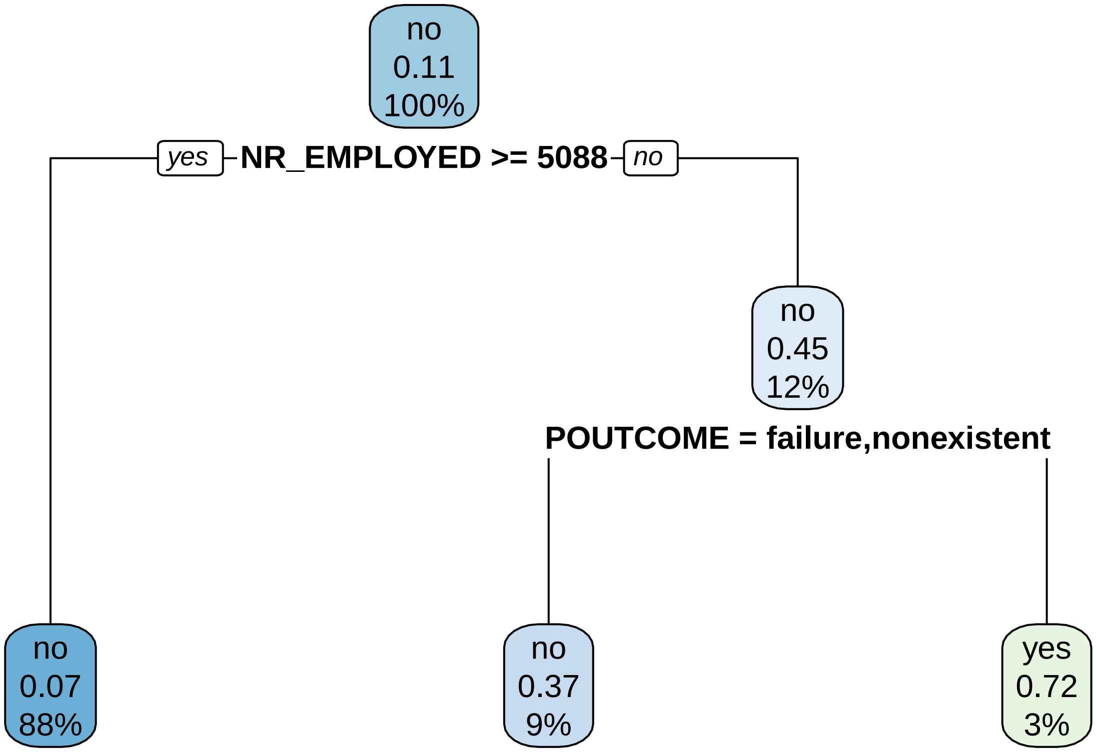
As seen below we have produced a basic Decision Tree to see the basic structure of the data when plotted.

```r
#Plot Basic Tree
basictree<-tree(Y~.,data = datatrainset,method = "class")
plot(basictree)
text(basictree)
```



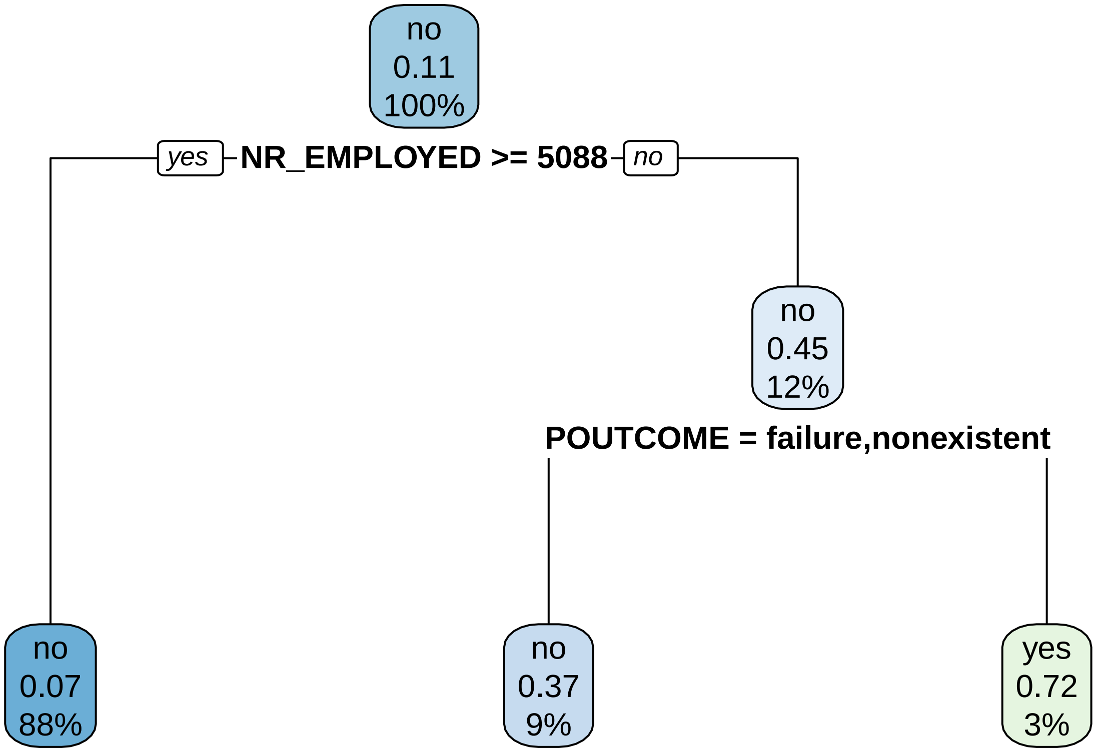
Plot an RPart tree to create a different type of tree

```r
#Plot RPart Tree
binary.model <- rpart(Y~., data = datatrainset,cp=0.006)
rpart.plot(binary.model)
```



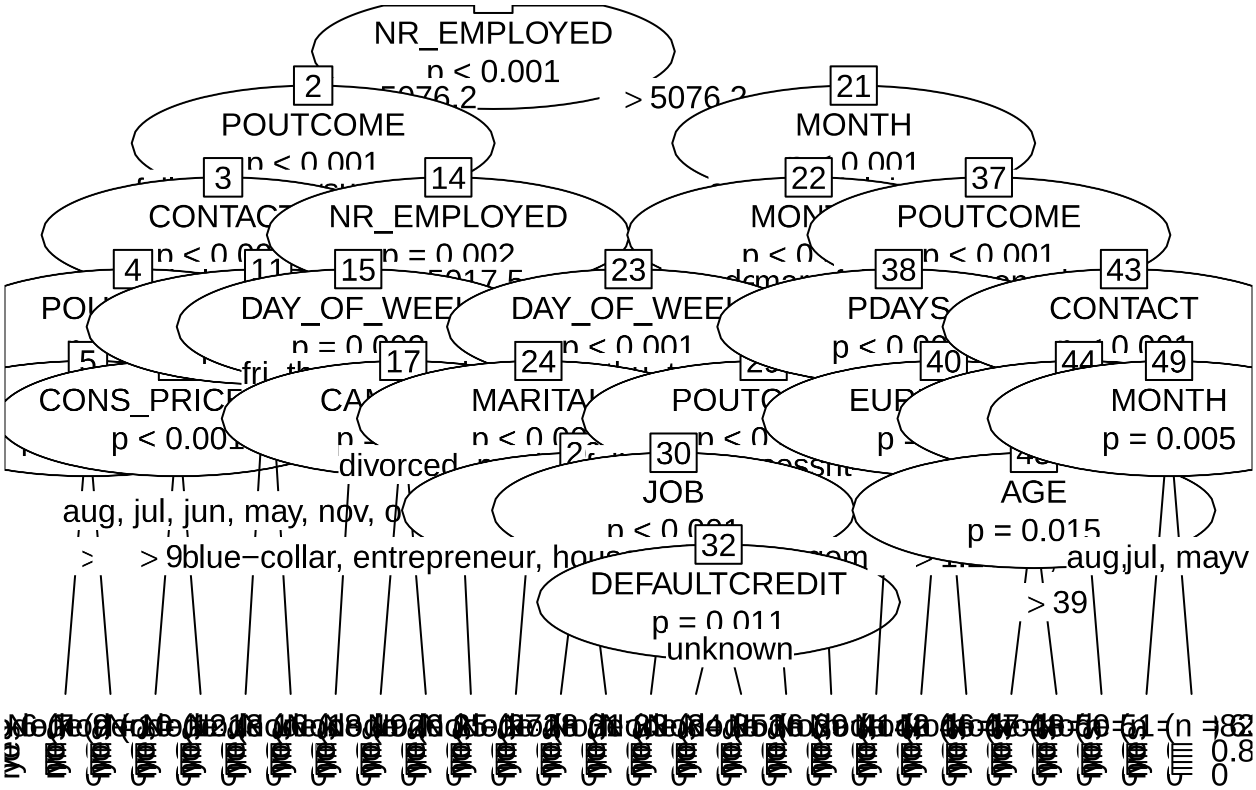
As it can be seen the tree is too large, we will use an autotune function to select the best CP value to plot the tree again.

```
prunedrparttree <- prune(binary.model,cp=binary.model$cptable[which.min(binary.model$cptable[,"xerror"]
rpart.plot(prunedrparttree)
```



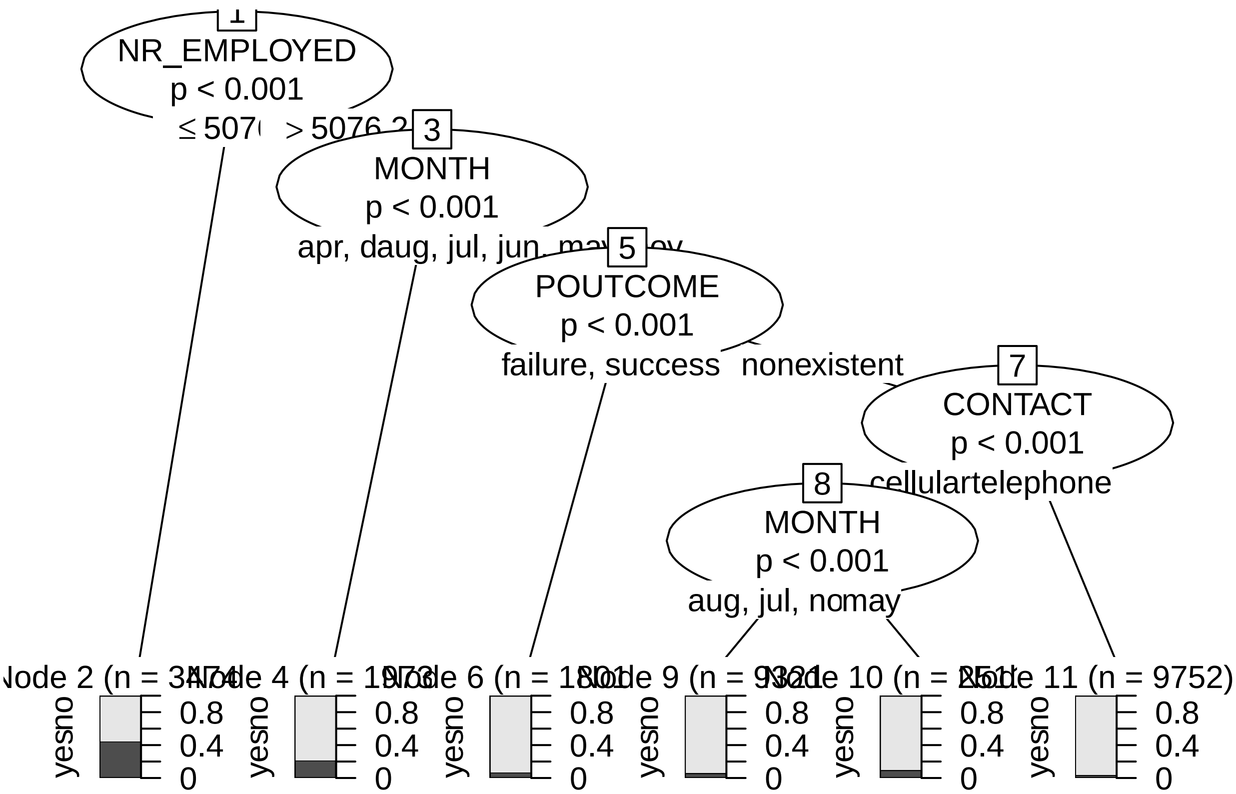
Plot Party Tree to create another tree to use as a comparison.

```
#Plot Party Tree
myFormula <- (Y~.)
decisiontree2 <- ctree(myFormula, data=datatrainset)
plot(decisiontree2)
```



Use the control function to plot refined tree to reduce clutter.

```r
#Create Control Function To Tune Hyperparamaters
ctreecontrol <- ctree_control(minsplit=5000,mincriterion = 0.999)

#Plot Pruned Party Tree
prunedctree <- ctree(myFormula, data=datatrainset,control = ctreecontrol)
plot(prunedctree)
```



Generate confusion matrices for all trees to view accuracy of Decision Trees.

```
#Generate Confusion Matrices For Models Using Caret Package

##Basic Tree Confusion Matrix
tree.predict1 <- predict(basictree, datatestset, type = "class")
confusionMatrix(datatestset$Y, tree.predict1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10870    95
##        yes  1154   237
##
##                Accuracy : 0.8989
##                  95% CI : (0.8935, 0.9042)
##     No Information Rate : 0.9731
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2422
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9040
##             Specificity : 0.7139
##          Pos Pred Value : 0.9913
##          Neg Pred Value : 0.1704
```

```
##               Prevalence : 0.9731
##           Detection Rate : 0.8797
##     Detection Prevalence : 0.8874
##        Balanced Accuracy : 0.8089
##
##         'Positive' Class : no
##
```

```
##RPart Tree Confusion Matrix
tree.predict2 <- predict(binary.model, datatestset, type = "class")
confusionMatrix(datatestset$Y, tree.predict2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10870    95
##        yes  1154   237
##
##                Accuracy : 0.8989
##                  95% CI : (0.8935, 0.9042)
##     No Information Rate : 0.9731
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2422
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9040
##             Specificity : 0.7139
##          Pos Pred Value : 0.9913
##          Neg Pred Value : 0.1704
##              Prevalence : 0.9731
##          Detection Rate : 0.8797
##    Detection Prevalence : 0.8874
##       Balanced Accuracy : 0.8089
##
##        'Positive' Class : no
##
```

```
##Party Tree Confusion Matrix
tree.predict3 <- predict(prunedctree, datatestset)
confusionMatrix(datatestset$Y, tree.predict3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10965     0
##        yes  1391     0
##
##                Accuracy : 0.8874
##                  95% CI : (0.8817, 0.8929)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8874
##             Specificity :     NA
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 1.0000
##          Detection Rate : 0.8874
##    Detection Prevalence : 0.8874
##       Balanced Accuracy :     NA
##
##        'Positive' Class : no
##
```

Usig Random Forest Package print the default random forest model and determine best mtry value for the model.

```r
#Use RandomForest Package To Generate Deafult Random Forest Model

set.seed(1234)
trControl <- trainControl(method = "cv",number = 2,search = "grid")

rf_default <- train(Y~.,data = datatrainset, method = "rf", metric= "Accuracy", trControl = trControl)

print(rf_default)
```

```
## Random Forest
##
## 28832 samples
##    19 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 14416, 14416
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8981687  0.2269790
##   27    0.8933130  0.3382255
##   52    0.8926540  0.3397156
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Expand the search to determine better mtry value.

```r
#Expand Search To Values Between 1 to 10 To Determin Best Mtry Value
set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(1: 10))
rf_mtry <- train(Y~.,data = datatrainset,method = "rf", metric = "Accuracy",tuneGrid = tuneGrid,trContro
print(rf_mtry)
```

```
## Random Forest
##
## 28832 samples
##    19 predictor
```

```
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 14416, 14416
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.8873474  0.0005457571
##    2    0.8977872  0.2256785564
##    3    0.8992786  0.2708186741
##    4    0.9005966  0.3065350032
##    5    0.9004925  0.3207267442
##    6    0.8999376  0.3233433343
##    7    0.8993133  0.3240491113
##    8    0.8989664  0.3273042733
##    9    0.8990011  0.3300295401
##   10    0.8991398  0.3357646492
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```r
#After Expanding Search Method --> Store best Mtry Value For Later Use
rf_mtry$bestTune$mtry
```

```
## [1] 4
```

```r
max(rf_mtry$results$Accuracy)
```

```
## [1] 0.9005966
```

```r
best.mtry <- rf_mtry$bestTune$mtry
best.mtry
```

```
## [1] 4
```

Determine The Best Number Of Max Nodes For The Model

```r
#Determine Best NumerOf Max Nodes
store_maxnode <- list(rf_mtry)
tuneGrid <- expand.grid(.mtry = best.mtry)
for (maxnodes in c(5: 15)) {
  set.seed(1234)
  rf_maxnode <- train(Y~.,
                      data = datatrainset,
                      method = "rf",
                      metric = "Accuracy",
                      tuneGrid = tuneGrid,
                      trControl = trControl,
                      importance = TRUE,
                      nodesize = 14,
                      maxnodes = maxnodes,
                      ntree = 300)
  current_iteration <- toString(maxnodes)
  store_maxnode[[current_iteration]] <- rf_maxnode
}
results_mtry <- resamples(store_maxnode)
```

```r
summary(results_mtry)
```

```
##
## Call:
## summary.resamples(object = results_mtry)
##
## Models: Model01, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
## Number of resamples: 2
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model01 0.8999029 0.9002497 0.9005966 0.9005966 0.9009434 0.9012902    0
## 5       0.8879023 0.8890642 0.8902261 0.8902261 0.8913880 0.8925499    0
## 6       0.8897059 0.8909198 0.8921337 0.8921337 0.8933477 0.8945616    0
## 7       0.8876942 0.8896712 0.8916482 0.8916482 0.8936251 0.8956021    0
## 8       0.8915094 0.8922378 0.8929661 0.8929661 0.8936945 0.8944229    0
## 9       0.8962264 0.8963651 0.8965039 0.8965039 0.8966426 0.8967814    0
## 10      0.8969201 0.8969374 0.8969548 0.8969548 0.8969721 0.8969895    0
## 11      0.8978219 0.8979086 0.8979953 0.8979953 0.8980820 0.8981687    0
## 12      0.8973363 0.8976311 0.8979259 0.8979259 0.8982207 0.8985155    0
## 13      0.8971282 0.8973883 0.8976484 0.8976484 0.8979086 0.8981687    0
## 14      0.8970588 0.8975444 0.8980300 0.8980300 0.8985155 0.8990011    0
## 15      0.8971282 0.8975444 0.8979606 0.8979606 0.8983768 0.8987930    0
##
## Kappa
##                Min.    1st Qu.     Median       Mean    3rd Qu.      Max.
## Model01 0.305879528 0.30620727 0.30653500 0.30653500 0.30686274 0.3071905
## 5       0.010589834 0.03414238 0.05769493 0.05769493 0.08124747 0.1048000
## 6       0.043744043 0.07069878 0.09765351 0.09765351 0.12460824 0.1515630
## 7       0.007340562 0.04755717 0.08777377 0.08777377 0.12799037 0.1682070
## 8       0.072347182 0.09131966 0.11029213 0.11029213 0.12926461 0.1482371
## 9       0.172029642 0.17764699 0.18326434 0.18326434 0.18888168 0.1944990
## 10      0.182673336 0.18740684 0.19214035 0.19214035 0.19687385 0.2016074
## 11      0.204578640 0.20713408 0.20968953 0.20968953 0.21224497 0.2148004
## 12      0.202174891 0.20403373 0.20589258 0.20589258 0.20775142 0.2096103
## 13      0.199555023 0.20473843 0.20992184 0.20992184 0.21510526 0.2202887
## 14      0.201352479 0.20752071 0.21368894 0.21368894 0.21985716 0.2260254
## 15      0.204873647 0.21126821 0.21766277 0.21766277 0.22405733 0.2304519
##         NA's
## Model01    0
## 5          0
## 6          0
## 7          0
## 8          0
## 9          0
## 10         0
## 11         0
## 12         0
## 13         0
## 14         0
## 15         0
```

Expand Search To Determine If A Higher Number Of Nodes Is Possible By Increasing The Node Size Between 20 and 30 From 5 and 15.

```r
#Retry To See If A Higher Number Of Nodes Is Possible
store_maxnode <- list(rf_mtry)
tuneGrid <- expand.grid(.mtry = best.mtry)
for (maxnodes in c(20: 30)) {
  set.seed(1234)
  rf_maxnode <- train(Y~.,
                      data = datatrainset,
                      method = "rf",
                      metric = "Accuracy",
                      tuneGrid = tuneGrid,
                      trControl = trControl,
                      importance = TRUE,
                      nodesize = 15,
                      maxnodes = maxnodes,
                      ntree = 300)
  key <- toString(maxnodes)
  store_maxnode[[key]] <- rf_maxnode
}
results_node <- resamples(store_maxnode)
summary(results_node)
```

```
##
## Call:
## summary.resamples(object = results_node)
##
## Models: Model01, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
## Number of resamples: 2
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model01 0.8999029 0.9002497 0.9005966 0.9005966 0.9009434 0.9012902    0
## 20      0.8976138 0.8979606 0.8983074 0.8983074 0.8986543 0.8990011    0
## 21      0.8980993 0.8982901 0.8984809 0.8984809 0.8986716 0.8988624    0
## 22      0.8974057 0.8978045 0.8982034 0.8982034 0.8986022 0.8990011    0
## 23      0.8980993 0.8982901 0.8984809 0.8984809 0.8986716 0.8988624    0
## 24      0.8980300 0.8982901 0.8985502 0.8985502 0.8988103 0.8990705    0
## 25      0.8981687 0.8983421 0.8985155 0.8985155 0.8986890 0.8988624    0
## 26      0.8983074 0.8984115 0.8985155 0.8985155 0.8986196 0.8987236    0
## 27      0.8983074 0.8984982 0.8986890 0.8986890 0.8988797 0.8990705    0
## 28      0.8978219 0.8981167 0.8984115 0.8984115 0.8987063 0.8990011    0
## 29      0.8981687 0.8983074 0.8984462 0.8984462 0.8985849 0.8987236    0
## 30      0.8979606 0.8982554 0.8985502 0.8985502 0.8988450 0.8991398    0
##
## Kappa
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model01 0.3058795 0.3062073 0.3065350 0.3065350 0.3068627 0.3071905    0
## 20      0.2135260 0.2196321 0.2257382 0.2257382 0.2318444 0.2379505    0
## 21      0.2207933 0.2251298 0.2294664 0.2294664 0.2338029 0.2381394    0
## 22      0.2122525 0.2180571 0.2238616 0.2238616 0.2296661 0.2354706    0
## 23      0.2246161 0.2261284 0.2276406 0.2276406 0.2291528 0.2306650    0
## 24      0.2199424 0.2244981 0.2290537 0.2290537 0.2336094 0.2381650    0
## 25      0.2273552 0.2295875 0.2318198 0.2318198 0.2340521 0.2362844    0
## 26      0.2296097 0.2296232 0.2296368 0.2296368 0.2296504 0.2296640    0
## 27      0.2246146 0.2286183 0.2326219 0.2326219 0.2366255 0.2406292    0
```

```
## 28      0.2186716 0.2237998 0.2289280 0.2289280 0.2340562 0.2391844    0
## 29      0.2229201 0.2259993 0.2290784 0.2290784 0.2321576 0.2352368    0
## 30      0.2190908 0.2249886 0.2308864 0.2308864 0.2367843 0.2426821    0
```

Repeat Process To Determine Best Number Of Ntree(s) Which Is An Essential HyperParameter To Optimise Tree

```r
#Now That We have The Best Value Of Mtry and MaxNode We Can Tune The Number Of Trees Using Same Method .
store_maxtrees <- list(rf_mtry)
for (ntree in c(250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000)) {
  set.seed(5678)
  rf_maxtrees <- train(Y~.,
                       data = datatrainset,
                       method = "rf",
                       metric = "Accuracy",
                       tuneGrid = tuneGrid,
                       trControl = trControl,
                       importance = TRUE,
                       nodesize = 14,
                       maxnodes = 24,
                       ntree = ntree)
  key <- toString(ntree)
  store_maxtrees[[key]] <- rf_maxtrees
}
results_tree <- resamples(store_maxtrees)
summary(results_tree)
```

```
##
## Call:
## summary.resamples(object = results_tree)
##
## Models: Model01, 250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000
## Number of resamples: 2
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model01 0.8999029 0.9002497 0.9005966 0.9005966 0.9009434 0.9012902    0
## 250     0.8980300 0.8984288 0.8988277 0.8988277 0.8992266 0.8996254    0
## 300     0.8978912 0.8982728 0.8986543 0.8986543 0.8990358 0.8994173    0
## 350     0.8978912 0.8982728 0.8986543 0.8986543 0.8990358 0.8994173    0
## 400     0.8978912 0.8984115 0.8989317 0.8989317 0.8994520 0.8999723    0
## 450     0.8979606 0.8983941 0.8988277 0.8988277 0.8992612 0.8996948    0
## 500     0.8982381 0.8986543 0.8990705 0.8990705 0.8994867 0.8999029    0
## 550     0.8982381 0.8986196 0.8990011 0.8990011 0.8993826 0.8997642    0
## 600     0.8980300 0.8984635 0.8988971 0.8988971 0.8993306 0.8997642    0
## 800     0.8980300 0.8984288 0.8988277 0.8988277 0.8992266 0.8996254    0
## 1000    0.8981687 0.8984809 0.8987930 0.8987930 0.8991052 0.8994173    0
## 2000    0.8982381 0.8986196 0.8990011 0.8990011 0.8993826 0.8997642    0
##
## Kappa
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model01 0.3058795 0.3062073 0.3065350 0.3065350 0.3068627 0.3071905    0
## 250     0.2274909 0.2346921 0.2418934 0.2418934 0.2490946 0.2562959    0
## 300     0.2239122 0.2318449 0.2397776 0.2397776 0.2477103 0.2556429    0
## 350     0.2200896 0.2283841 0.2366786 0.2366786 0.2449731 0.2532676    0
```

```
## 400     0.2258095 0.2334073 0.2410051 0.2410051 0.2486029 0.2562008    0
## 450     0.2253886 0.2327247 0.2400608 0.2400608 0.2473969 0.2547330    0
## 500     0.2274949 0.2347652 0.2420355 0.2420355 0.2493057 0.2565760    0
## 550     0.2281247 0.2346824 0.2412401 0.2412401 0.2477977 0.2543554    0
## 600     0.2274909 0.2342070 0.2409231 0.2409231 0.2476393 0.2543554    0
## 800     0.2274909 0.2330494 0.2386079 0.2386079 0.2441664 0.2497249    0
## 1000    0.2272836 0.2331819 0.2390803 0.2390803 0.2449787 0.2508771    0
## 2000    0.2262323 0.2328150 0.2393977 0.2393977 0.2459803 0.2525630    0
```

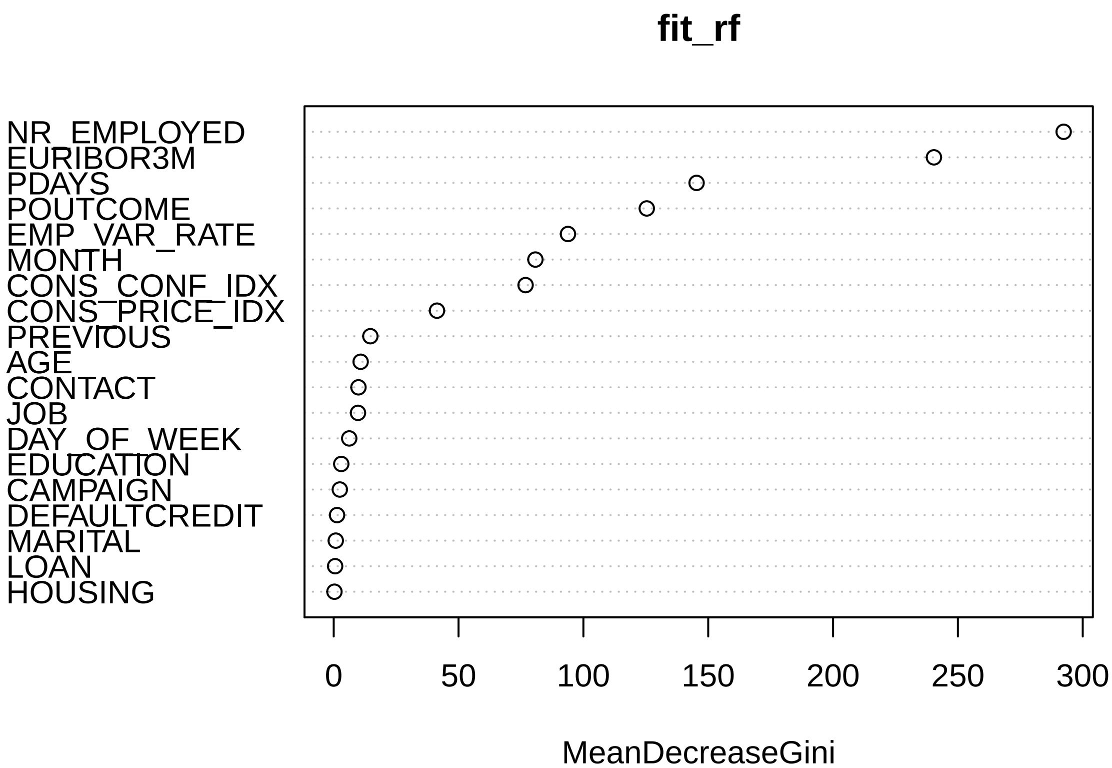
Fit Optimal Values To New Model To Check Accuracy

```r
#As We Now Have The Final Model --> We Can Train The Random Forest With The Best Parameters As Determin
fit_rf <- train(Y~.,
                datatrainset,
                method = "rf",
                metric = "Accuracy",
                tuneGrid = tuneGrid,
                trControl = trControl,
                importance = TRUE,
                nodesize = 30,
                ntree = 1000,
                maxnodes = 15)

#Determine Confusion Matrix For Optimal Model
prediction <-predict(fit_rf, datatestset)
confusionMatrix(prediction, datatestset$Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10890  1173
##        yes    75   218
##
##                Accuracy : 0.899
##                  95% CI : (0.8935, 0.9043)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 1.932e-05
##
##                   Kappa : 0.2287
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9932
##             Specificity : 0.1567
##          Pos Pred Value : 0.9028
##          Neg Pred Value : 0.7440
##              Prevalence : 0.8874
##          Detection Rate : 0.8814
##    Detection Prevalence : 0.9763
##       Balanced Accuracy : 0.5749
##
##        'Positive' Class : no
##
```

We Can Now Plot Variable Importance To Determine Variables With Highest Information Gain And Overall Importance.

```r
#As We Now Have Best Parameters For Model, We Can Plot Variable Importance Using Random Forest Package .

#DataTrainSet
fit_rf <- randomForest(Y ~ ., data = datatrainset,ntree = 1000, nodesize = 30,maxnodes=15)
##Plot As Graph
varImpPlot(fit_rf)
```

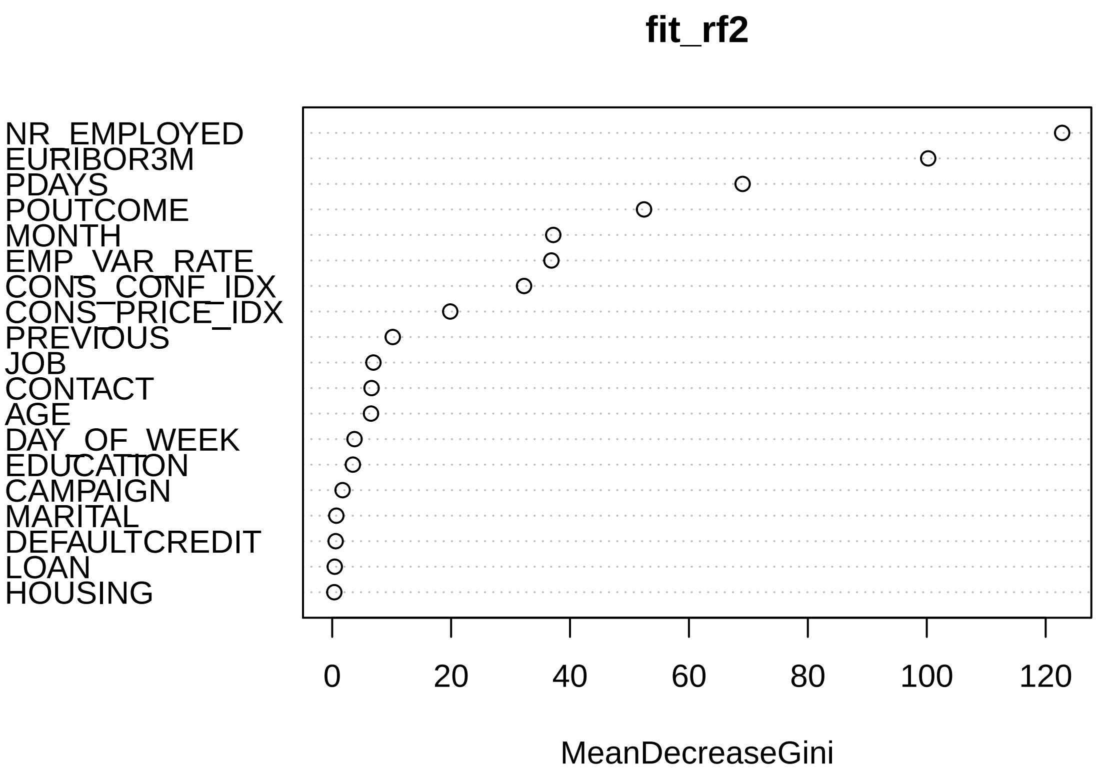



```r
##Plot As Importance Table
varImp(fit_rf)
```

```
##                     Overall
## AGE              10.7902903
## JOB               9.7249602
## MARITAL           0.8242236
## EDUCATION         3.0286610
## DEFAULTCREDIT     1.3374461
## HOUSING           0.2862270
## LOAN              0.5703225
## CONTACT           9.9142080
## MONTH            80.7786845
## DAY_OF_WEEK       6.2168234
## CAMPAIGN          2.4511491
## PDAYS           145.3424987
## PREVIOUS         14.6805120
## POUTCOME        125.3963916
## EMP_VAR_RATE     93.8236104
```

```
## CONS_PRICE_IDX  41.3655266
## CONS_CONF_IDX   76.8468261
## EURIBOR3M      240.4032279
## NR_EMPLOYED    292.3597416
```

```r
#DataTestSet
fit_rf2 <- randomForest(Y ~ ., data = datatestset,ntree = 1000, nodesize = 30,maxnodes=15)
##Plot As Graph
varImpPlot(fit_rf2)
```




```r
##Plot As Importance Table
varImp(fit_rf2)
```

```
##                   Overall
## AGE             6.5257154
## JOB             6.9309534
## MARITAL         0.6924874
## EDUCATION       3.4766752
## DEFAULTCREDIT   0.5803034
## HOUSING         0.3499085
## LOAN            0.4285604
## CONTACT         6.6205884
## MONTH          37.1831823
## DAY_OF_WEEK     3.7552360
## CAMPAIGN        1.7447953
## PDAYS          69.0278214
## PREVIOUS       10.1809478
```

```
## POUTCOME        52.4509934
## EMP_VAR_RATE    36.8703053
## CONS_PRICE_IDX  19.8505609
## CONS_CONF_IDX   32.2678097
## EURIBOR3M      100.2388788
## NR_EMPLOYED    122.7735155
```

```r
#Plot Variable Importance Table For TrainData And TestData

##DataTrain
xgb_fit <- train(Y ~ .,
                 data = datatrainset,
                 method = "xgbLinear")
caret_imp <- varImp(xgb_fit)
caret_imp
```

```
## xgbLinear variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                             Overall
## NR_EMPLOYED                 100.000
## EURIBOR3M                    42.218
## AGE                          25.276
## CONS_CONF_IDX                15.240
## CAMPAIGN                     13.086
## POUTCOMEsuccess              10.923
## PDAYS                         8.318
## CONS_PRICE_IDX                5.305
## PREVIOUS                      5.047
## CONTACTtelephone              4.432
## MONTHoct                      3.662
## DAY_OF_WEEKmon                3.303
## EMP_VAR_RATE                  2.954
## DAY_OF_WEEKtue                2.655
## HOUSINGyes                    2.638
## DEFAULTCREDITunknown          2.469
## LOANyes                       2.292
## EDUCATIONuniversity.degree    2.290
## JOBtechnician                 2.217
## MARITALsingle                 1.985
```

```r
xgb_imp <- xgb.importance(feature_names = xgb_fit$finalModel$feature_names,
                          model = xgb_fit$finalModel)

##DataTest
xgb_fit2 <- train(Y ~ .,
                 data = datatestset,
                 method = "xgbLinear")
caret_imp2 <- varImp(xgb_fit2)
caret_imp2
```

```
## xgbLinear variable importance
##
##   only 20 most important variables shown (out of 52)
```

```
##
##                              Overall
## NR_EMPLOYED                  100.000
## EURIBOR3M                     56.312
## AGE                           47.050
## CAMPAIGN                      22.173
## PDAYS                         20.560
## CONS_CONF_IDX                 16.153
## CONS_PRICE_IDX                 8.158
## PREVIOUS                       7.908
## HOUSINGyes                     6.537
## EMP_VAR_RATE                   5.950
## CONTACTtelephone               5.151
## DAY_OF_WEEKwed                 4.928
## DAY_OF_WEEKtue                 4.500
## LOANyes                        4.259
## EDUCATIONuniversity.degree     4.129
## DAY_OF_WEEKthu                 4.097
## DAY_OF_WEEKmon                 4.089
## MONTHoct                       3.975
## JOBblue-collar                 3.222
## MONTHnov                       3.213
```

```r
xgb_imp2 <- xgb.importance(feature_names = xgb_fit2$finalModel$feature_names,
                          model = xgb_fit2$finalModel)
```

```r
#Print Information Gain Table
print(xgb_imp)
```

```
##                         Feature         Gain       Cover   Frequency
##  1:                 NR_EMPLOYED 0.3548166674 0.063202033 0.014534884
##  2:                   EURIBOR3M 0.1497966145 0.231966182 0.168604651
##  3:                         AGE 0.0896831128 0.134668216 0.197189922
##  4:               CONS_CONF_IDX 0.0540741100 0.055791162 0.033914729
##  5:                    CAMPAIGN 0.0464324447 0.093508859 0.100290698
##  6:             POUTCOMEsuccess 0.0387561948 0.014166456 0.005813953
##  7:                       PDAYS 0.0295145093 0.045380482 0.032945736
##  8:              CONS_PRICE_IDX 0.0188233429 0.026956855 0.034883721
##  9:                    PREVIOUS 0.0179081483 0.049232130 0.032461240
## 10:            CONTACTtelephone 0.0157238352 0.042193088 0.016472868
## 11:                    MONTHoct 0.0129943754 0.033916735 0.005813953
## 12:              DAY_OF_WEEKmon 0.0117206377 0.013860925 0.018895349
## 13:                EMP_VAR_RATE 0.0104811184 0.006106552 0.015988372
## 14:              DAY_OF_WEEKtue 0.0094206474 0.006562337 0.015988372
## 15:                  HOUSINGyes 0.0093610316 0.011331625 0.029069767
## 16:         DEFAULTCREDITunknown 0.0087588010 0.015035044 0.012112403
## 17:                     LOANyes 0.0081329409 0.005450522 0.015503876
## 18:   EDUCATIONuniversity.degree 0.0081255629 0.007869675 0.018895349
## 19:               JOBtechnician 0.0078671667 0.004274255 0.015019380
## 20:               MARITALsingle 0.0070429075 0.011532693 0.015988372
## 21:              MARITALmarried 0.0057465068 0.002244542 0.015019380
## 22: EDUCATIONprofessional.course 0.0056951349 0.004282098 0.013081395
## 23:          EDUCATIONhigh.school 0.0055160032 0.002655630 0.011143411
## 24:              JOBblue-collar 0.0053509775 0.003476284 0.010174419
## 25:              DAY_OF_WEEKthu 0.0049636290 0.003999786 0.012596899
```

```
## 26:             EDUCATIONbasic.9y 0.0049156181 0.002465861 0.010658915
## 27:             EDUCATIONbasic.6y 0.0046754749 0.002640346 0.008236434
## 28:                DAY_OF_WEEKwed 0.0046176313 0.010333257 0.009205426
## 29:                      MONTHjul 0.0044140709 0.001706689 0.007751938
## 30:                   JOBservices 0.0041226089 0.001115271 0.007267442
## 31:                      MONTHaug 0.0039163064 0.001968198 0.007267442
## 32:                      MONTHmay 0.0037767763 0.003260160 0.006782946
## 33:              EDUCATIONunknown 0.0037634648 0.003898575 0.008720930
## 34:              JOBself-employed 0.0035622817 0.009773633 0.008236434
## 35:                    JOBretired 0.0034427549 0.001374495 0.007267442
## 36:                HOUSINGunknown 0.0031056843 0.005199534 0.005813953
## 37:                 JOBmanagement 0.0030734181 0.002414194 0.007267442
## 38:               JOBentrepreneur 0.0029863594 0.011183373 0.007267442
## 39:                    JOBstudent 0.0027719778 0.006960332 0.007267442
## 40:                      MONTHjun 0.0025255649 0.004261216 0.006298450
## 41:                      MONTHnov 0.0014193078 0.003736467 0.004844961
## 42:                 JOBunemployed 0.0013499499 0.002331692 0.003391473
## 43:                  JOBhousemaid 0.0011423719 0.014653559 0.003875969
## 44:                      MONTHsep 0.0010801884 0.003007258 0.002906977
## 45:                    JOBunknown 0.0010012933 0.007675755 0.002906977
## 46:                MARITALunknown 0.0007704536 0.002838649 0.001453488
## 47:                      MONTHmar 0.0005488973 0.002402631 0.001453488
## 48:                      MONTHdec 0.0003111243 0.005134689 0.001453488
##                         Feature         Gain       Cover   Frequency
```

```r
print(xgb_imp2)
```

```
##                         Feature         Gain        Cover   Frequency
##  1:                 NR_EMPLOYED 0.2679181636 5.479068e-02 0.012679162
##  2:                   EURIBOR3M 0.1508701047 2.026025e-01 0.173098126
##  3:                         AGE 0.1260559923 1.406908e-01 0.215545755
##  4:                    CAMPAIGN 0.0594048461 8.683226e-02 0.096471885
##  5:                       PDAYS 0.0550837405 5.127284e-02 0.027563396
##  6:               CONS_CONF_IDX 0.0432776618 6.099188e-02 0.032524807
##  7:              CONS_PRICE_IDX 0.0218560892 5.706279e-02 0.031973539
##  8:                    PREVIOUS 0.0211878211 2.126500e-02 0.033076075
##  9:                  HOUSINGyes 0.0175150040 5.265300e-03 0.026460860
## 10:                EMP_VAR_RATE 0.0159416152 1.949256e-02 0.012127894
## 11:            CONTACTtelephone 0.0137994961 2.703714e-02 0.015435502
## 12:              DAY_OF_WEEKwed 0.0132038101 6.360311e-03 0.015986770
## 13:              DAY_OF_WEEKtue 0.0120553080 6.463673e-03 0.020948181
## 14:                     LOANyes 0.0114105892 9.054379e-03 0.015986770
## 15:   EDUCATIONuniversity.degree 0.0110616355 3.287273e-03 0.020396913
## 16:              DAY_OF_WEEKthu 0.0109772322 9.361397e-03 0.018743109
## 17:              DAY_OF_WEEKmon 0.0109543224 1.452679e-02 0.019845645
## 18:                    MONTHoct 0.0106489036 3.567948e-02 0.007166483
## 19:              JOBblue-collar 0.0086310625 9.440074e-03 0.013781698
## 20:                    MONTHnov 0.0086088678 5.145934e-03 0.008269019
## 21:              MARITALmarried 0.0084618659 3.543327e-03 0.015986770
## 22:        DEFAULTCREDITunknown 0.0080276388 1.107697e-02 0.011025358
## 23:         EDUCATIONhigh.school 0.0078322256 7.509253e-03 0.013781698
## 24:               MARITALsingle 0.0075723092 9.523087e-03 0.012679162
## 25: EDUCATIONprofessional.course 0.0067743843 8.700940e-03 0.011576626
## 26:           EDUCATIONbasic.9y 0.0060743252 8.706124e-03 0.009371555
## 27:            EDUCATIONunknown 0.0059228779 7.909836e-03 0.009371555
```

```
## 28:            EDUCATIONbasic.6y 0.0053873655 4.734191e-03 0.007166483
## 29:               JOBtechnician 0.0048153589 1.450717e-02 0.010474090
## 30:             JOBself-employed 0.0046570232 5.880277e-03 0.006063947
## 31:                  JOBretired 0.0040755522 5.383543e-03 0.007166483
## 32:                 JOBservices 0.0040111921 1.150750e-02 0.007166483
## 33:                    MONTHjul 0.0039318582 1.637637e-03 0.006063947
## 34:                    MONTHmar 0.0034579453 3.836776e-03 0.002756340
## 35:               JOBmanagement 0.0032334607 1.920040e-03 0.004410143
## 36:                JOBhousemaid 0.0031027246 4.076225e-03 0.004961411
## 37:            POUTCOMEsuccess 0.0030208771 7.384665e-03 0.004410143
## 38:                    MONTHjun 0.0029015997 9.528598e-04 0.006063947
## 39:               JOBunemployed 0.0027160948 5.615346e-03 0.004961411
## 40:                  JOBstudent 0.0023826436 8.077753e-04 0.002756340
## 41:                    MONTHdec 0.0020888595 8.933568e-04 0.002205072
## 42:            JOBentrepreneur 0.0020462043 1.036525e-02 0.004410143
## 43:                  JOBunknown 0.0020404909 1.373975e-02 0.004410143
## 44:              HOUSINGunknown 0.0017458782 6.416110e-03 0.004410143
## 45:                    MONTHaug 0.0016827978 1.541703e-03 0.002756340
## 46:                    MONTHmay 0.0005686417 2.739301e-03 0.001653804
## 47:                    MONTHsep 0.0005269964 9.229898e-05 0.001102536
## 48:              MARITALunknown 0.0004785425 1.237562e-02 0.002756340
##                      Feature         Gain        Cover   Frequency
```

From Above We Can Select The Most Important Variables To Be Used For The Company

```r
#Calculate Benchmark Value

frequencytable <- table(clean_loan$Y)
probabilitytable1 <- frequencytable/sum(frequencytable)
BV <- probabilitytable1 [1]
print(BV)
```

```
##        no
## 0.8873458
```

From The Above We Can See That The Optimal Model Has A Higher Accuracy Than The Benchmark Value However It Has The Same Value As The RPart Tree Resulting In This Being The Best Model To Use

Model Decision

It is seen that the three confusion matricies returned 100% accuracy in the model, this is questionable however the large amount of data increases the accuracy of the model. However, this also increases variablility in the model. With a 95% confidence inteval there is 5% chance of error. It is evident that all models perform and predict very well due to the large dataset and due to the explainability of the defult outcome based on so many varibales. Therefore we would be indifferent as to which model is used. However, we will present the party model to the management team as the party model has high levels of versitility which is valued in potential changes made to the model in the future and expansion of the company. However, on further thought it is clear this model has been overfit and therefore we can prune the model in order to refine it

Conclusion

RPart Model accuracy:0.9013

With 90.13% accuracy using the trained model on test data this is a model that has a high level of accuracy.

Previous overfitting was found to be result of conflicting variables which were missed in the initial trim. of data.

However, as this refining prune process of the data occured after data was taken out it was easier to locate the variables with low information gain.

From our Importance Plot we can see that by looking at the most overall importance of the variables in the model we can specify in on the likelihood of a loan defaulting.

Varaiables: . NR_EMPLOYED . AGE . EURIBOR3M . CAMPAIGN . CONS_CONF_IDX . PDays . EMP_VAR_RATE