hw4_JH

Daniel Brooks (daniel.brooks@spsmail.cuny.edu), Daniel Fanelli (daniel.fanelli@spsmail.cuny.edu), Christopher Fenton (christopher.fenton@spsmail.cuny.edu), James Hamski (james.hamski@spsmail.cuny.edu), Youqing Xiang (youqing.xiang@spsmail.cuny.edu)

July 8, 2016

I used Chris's data prep code

Libraries:

```
library(stringr)
#library(PerformanceAnalytics)
#library(aod)
library(ggplot2)
#library(Rcpp)
#library(Amelia)

#Note packages
library(dplyr)
library(rpart)
library(caret)
library(ROCR)
```

Loading the data:

```
ins <- read.csv('insurance_training_data.csv',na.strings=c("","NA"),stringsAsFactors = FALSE)
```

Data Exploration

```
##       INDEX TARGET_FLAG  TARGET_AMT    KIDSDRIV         AGE    HOMEKIDS
##           0           0           0           0           6           0
##         YOJ      INCOME     PARENT1    HOME_VAL     MSTATUS         SEX
##         454         445           0         464           0           0
##   EDUCATION         JOB    TRAVTIME     CAR_USE    BLUEBOOK         TIF
##           0         526           0           0           0           0
##    CAR_TYPE     RED_CAR    OLDCLAIM    CLM_FREQ     REVOKED     MVR_PTS
##           0           0           0           0           0           0
##     CAR_AGE  URBANICITY
##         510           0
```

There are 2116 incomplete observations (missing at least one variable value), these will need to either be imputed or dropped.

Data Transformation

Converting the money character fields to numeric.

Converting the categoricals to factors:

Models

Tree-based Models

Classification / regression trees are a intuitive way of analyzing the insurance dataset. You can imagine an insurance agent reading of a list of questions pertaining to variables from the dataset (what is the age of the driver? how many points are on the driver's license?) and using the responses to assess the risk, and therefore cost, of the policy. Unlike more 'black box' methods like principle components analysis, a non-technical end user can typically easily understand the output of a tree model.

A tree model performs three steps:

1) Partition each predictor
2) Take the mean of the response in each predictor and compute:

$$RSS(partition) = RSS(part_1) + RSS(part_2)$$

   Then, choose the partition that minimizes the residual sum of squares (RSS)
3) Subpartition the partitions in a recursive manner.

The split is made at each step based on the independent variable that results in the largest possible reduction in heterogeneity of the response variable. The end result is a rule tree which minimizes RSS. Note there is a bais-variance trade-off with tree based models - we could specify many steps and reduce our RSS error, but that would result in an overfitted model.

Tree-based model for TARGET_FLAG

First, we fit a tree-based model to the crash / no crash (TARGET_FLAG) binary variable. The rpart package treats a binary variable as a subset of categorical variables. Therefore, we set the method argument in the tree building function to method = "class". In order to asses our model, we split the dataset into a training set (80%) and test (20%) set.

```
#easier with dplyr, rewrite later
intrain <-createDataPartition(y=ins$TARGET_FLAG,p=0.8,list=FALSE)

ins.train <- ins[intrain,]
ins.test <- ins[-intrain,]

# See: http://www.statmethods.net/advstats/cart.html
# https://eight2late.wordpress.com/2016/02/16/a-gentle-introduction-to-decision-trees-using-r/

# grow tree
tree.model.flag <- rpart(TARGET_FLAG ~ KIDSDRIV + AGE + HOMEKIDS + YOJ + INCOME + PARENT1 + HOME_VAL + I
```

The CP table for the TARGET_FLAG tree based model shows the optimal prunings based on a complexity parameter.

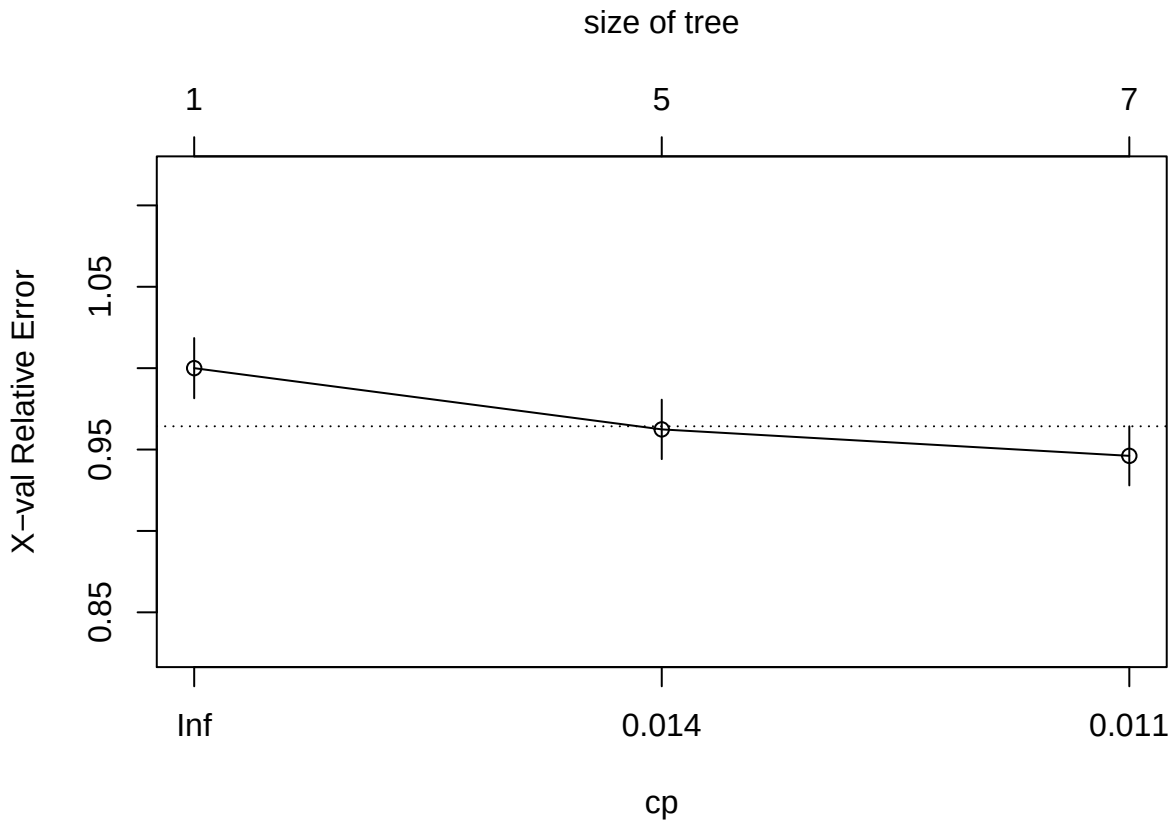
```
printcp(tree.model.flag) # display the results
```

```
##
## Classification tree:
## rpart(formula = TARGET_FLAG ~ KIDSDRIV + AGE + HOMEKIDS + YOJ +
```

```
##     INCOME + PARENT1 + HOME_VAL + MSTATUS + SEX + EDUCATION +
##     JOB + TRAVTIME + CAR_USE + BLUEBOOK + TIF + CAR_TYPE + RED_CAR +
##     OLDCLAIM + CLM_FREQ + REVOKED + MVR_PTS + CAR_AGE + URBANICITY,
##     data = ins, method = "class")
##
## Variables actually used in tree construction:
## [1] CAR_TYPE   HOME_VAL   JOB        OLDCLAIM   TRAVTIME   URBANICITY
##
## Root node error: 2153/8161 = 0.26382
##
## n= 8161
##
##         CP nsplit rel error  xerror     xstd
## 1 0.018695      0   1.00000 1.00000 0.018491
## 2 0.011147      4   0.92522 0.96238 0.018262
## 3 0.010000      6   0.90293 0.94612 0.018159
```

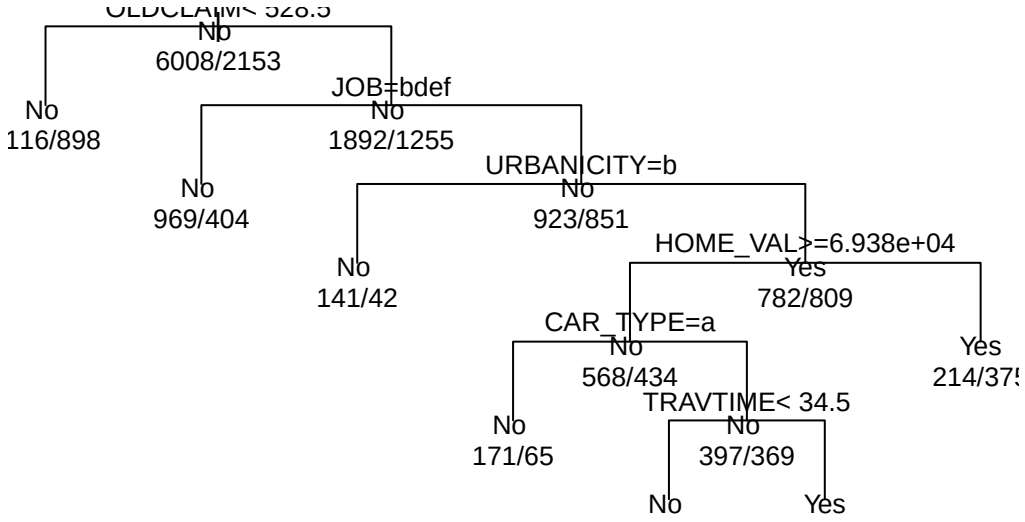
The relative error decreases very little moving from a 5 level tree to a 7 level tree, therefore we should favor the simpler 5-level model.

```
plotcp(tree.model.flag) # visualize cross-validation results
```




```
#summary(fit) # detailed summary of splits
```

```
# plot tree
plot(tree.model.flag, uniform = TRUE, main = "Classification Tree")
text(tree.model.flag, use.n = TRUE, all = TRUE, cex = .8)
```



Tree-based model for TARGET_AMT

The dependent variable TARGET_AMT was log-transformed before running the tree-based model. Using the defaul parameters, the independent varaibles did not contain enough information about TARGET_AMT to grow a tree beyond 2 levels. Therefore, the complexity parameter was decreased to 0.005. Despite this, the lowest error was achieved at level 2 so we pruned the tree there.

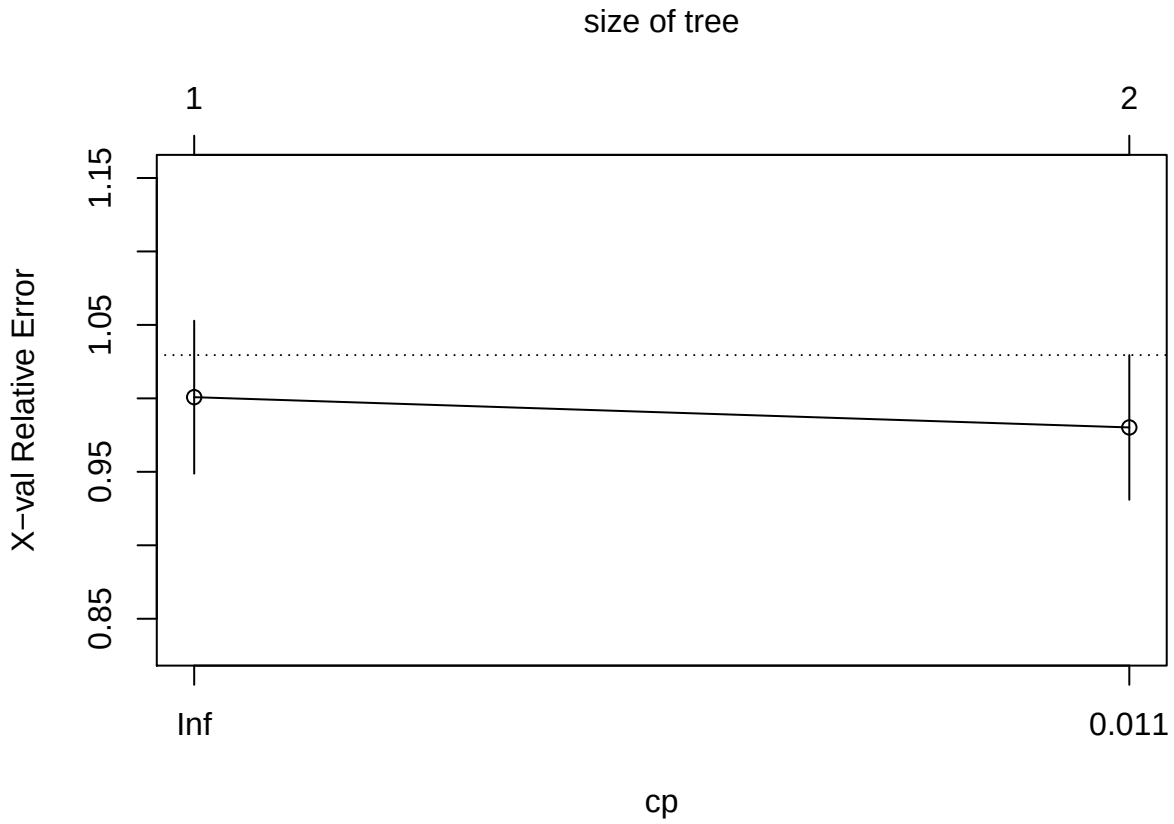
```
ins.train.AMT <- filter(ins.train, TARGET_FLAG == "Yes")

ins.train.AMT$TARGET_AMT <- log(ins.train.AMT$TARGET_AMT)

# grow tree
tree.model.flag2 <- rpart(TARGET_AMT ~ KIDSDRIV + AGE + HOMEKIDS + YOJ + INCOME + PARENT1 + HOME_VAL + I
```

```
printcp(tree.model.flag2) # display the results
```

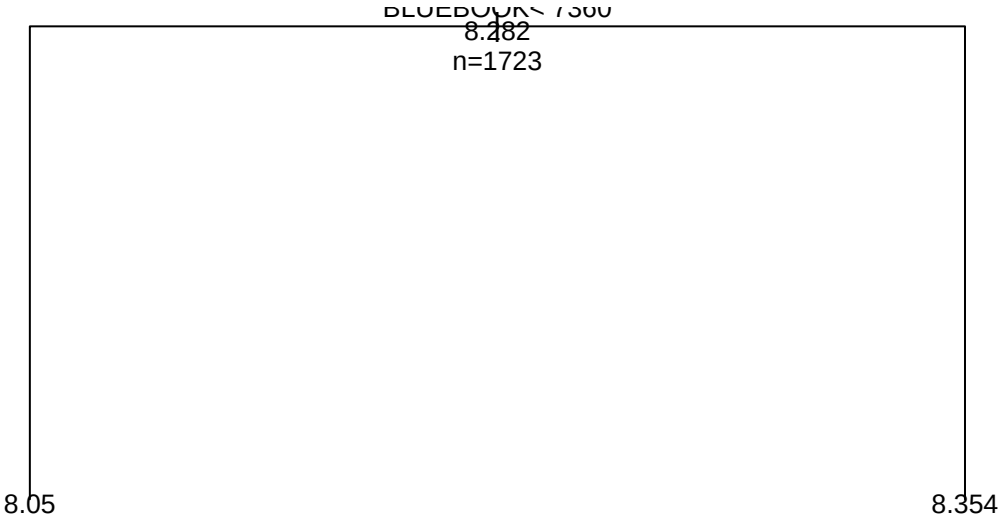
```
##
## Regression tree:
## rpart(formula = TARGET_AMT ~ KIDSDRIV + AGE + HOMEKIDS + YOJ +
##     INCOME + PARENT1 + HOME_VAL + MSTATUS + SEX + EDUCATION +
##     JOB + TRAVTIME + CAR_USE + BLUEBOOK + TIF + CAR_TYPE + RED_CAR +
##     OLDCLAIM + CLM_FREQ + REVOKED + MVR_PTS + CAR_AGE + URBANICITY,
##     data = ins.train.AMT, method = "anova", control = rpart.control(cp = 0.005))
##
## Variables actually used in tree construction:
## [1] BLUEBOOK
##
## Root node error: 1145.3/1723 = 0.66472
##
## n= 1723
##
##         CP nsplit rel error  xerror     xstd
```

```
## 1 0.025166      0   1.00000 1.00082 0.052039
## 2 0.005000      1   0.97483 0.98022 0.049236
```

```r
plotcp(tree.model.flag2) # visualize cross-validation results
```



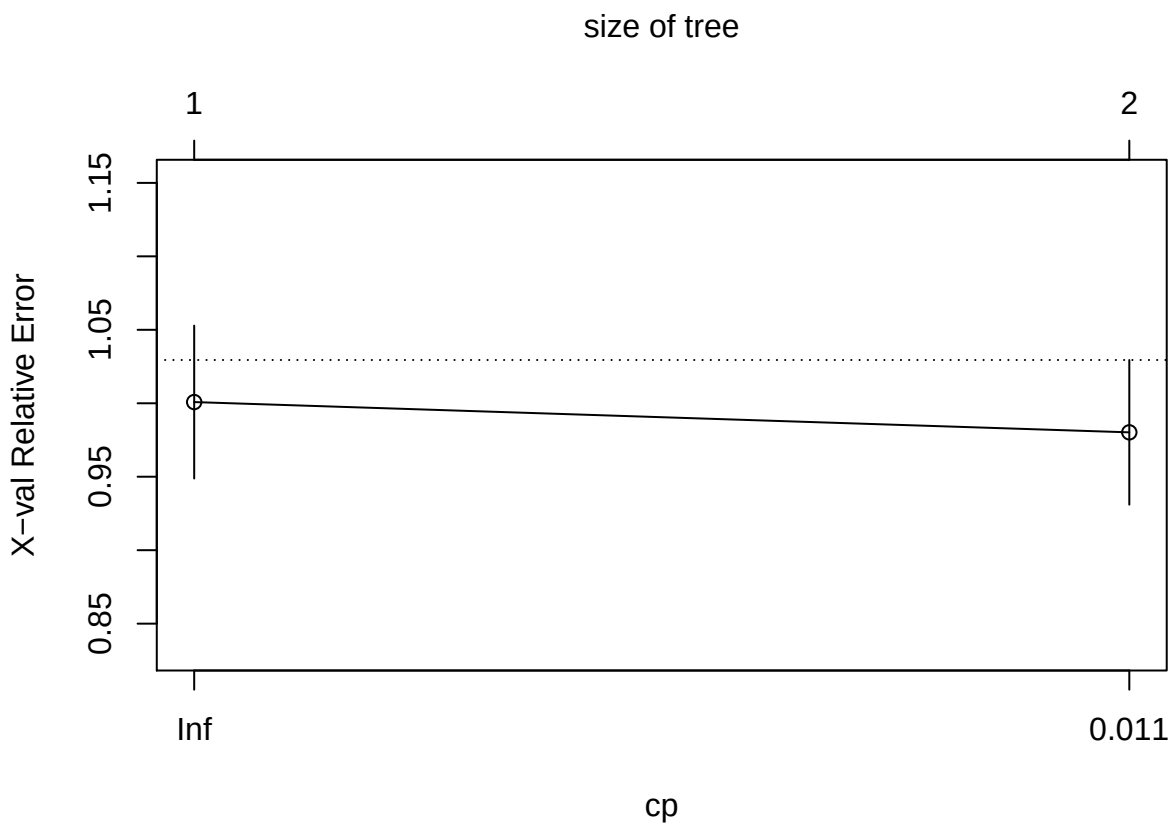

```r
#summary(fit2) # detailed summary of splits

# plot tree
plot(tree.model.flag2, uniform = TRUE, main = "Classification Tree")
text(tree.model.flag2, use.n = TRUE, all = TRUE, cex = .8)
```



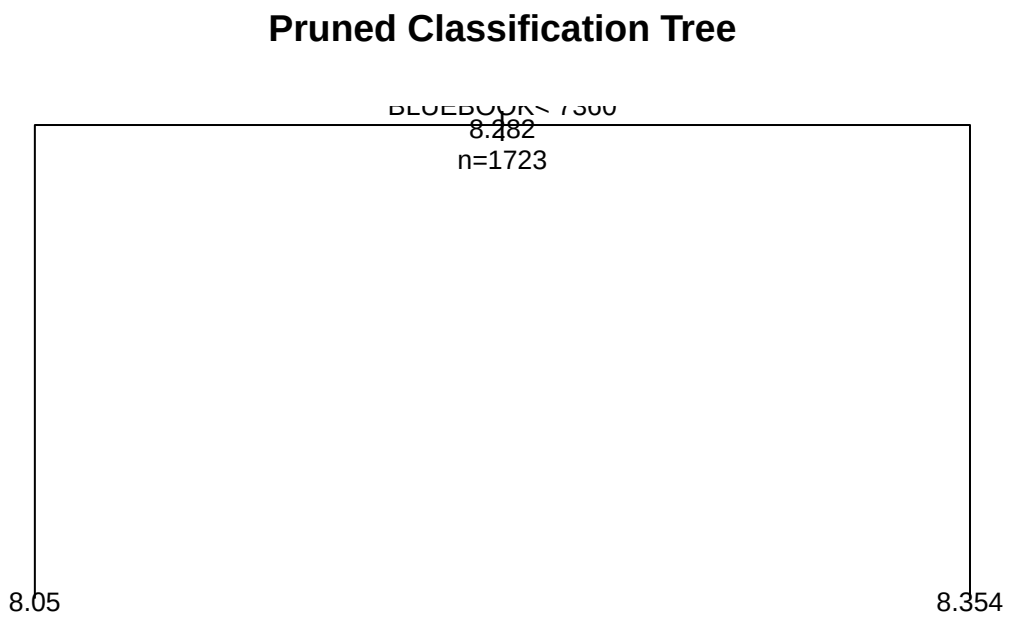

```
# prune the tree
tree.model.flag2.prune <- prune(tree.model.flag2, cp = tree.model.flag2$cptable[which.min(tree.model.fla

printcp(tree.model.flag2.prune)
```

```
##
## Regression tree:
## rpart(formula = TARGET_AMT ~ KIDSDRIV + AGE + HOMEKIDS + YOJ +
##     INCOME + PARENT1 + HOME_VAL + MSTATUS + SEX + EDUCATION +
##     JOB + TRAVTIME + CAR_USE + BLUEBOOK + TIF + CAR_TYPE + RED_CAR +
##     OLDCLAIM + CLM_FREQ + REVOKED + MVR_PTS + CAR_AGE + URBANICITY,
##     data = ins.train.AMT, method = "anova", control = rpart.control(cp = 0.005))
##
## Variables actually used in tree construction:
## [1] BLUEBOOK
##
## Root node error: 1145.3/1723 = 0.66472
##
## n= 1723
##
##         CP nsplit rel error  xerror     xstd
## 1 0.025166      0   1.00000 1.00082 0.052039
## 2 0.005000      1   0.97483 0.98022 0.049236
```

```
plotcp(tree.model.flag2.prune)
```




```
#summary(pfit2)
# plot the pruned tree
plot(tree.model.flag2.prune, uniform = TRUE, main = "Pruned Classification Tree")
text(tree.model.flag2.prune, use.n = TRUE, all = TRUE, cex = .8)
```



This trimmed model shows that the most important variable for predicting TARGET_AMNT is the Bluebook value of the car.

Model Analysis

Tree-based model for TARGET_FLAG

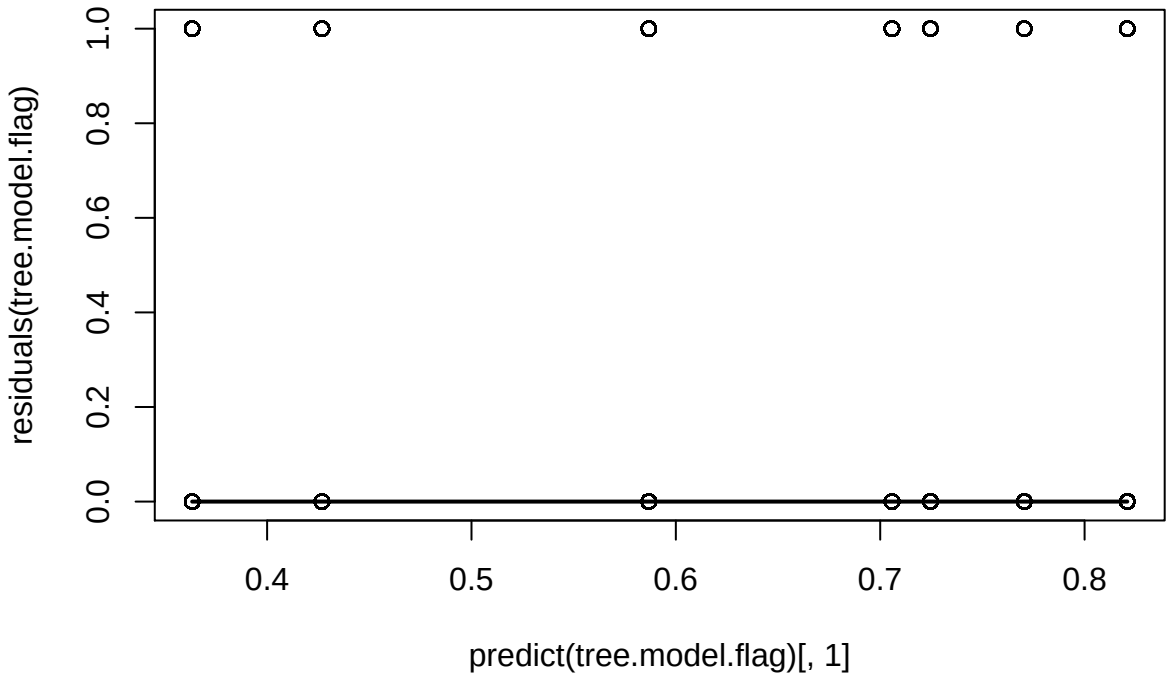
Confusion Matrix

```
rpart_predict <- predict(tree.model.flag, ins.test, type="class")
rpart_predict.probs <- predict(tree.model.flag, ins.test)
#confusion matrix
confusionMatrix(rpart_predict, ins.test$TARGET_FLAG)
```

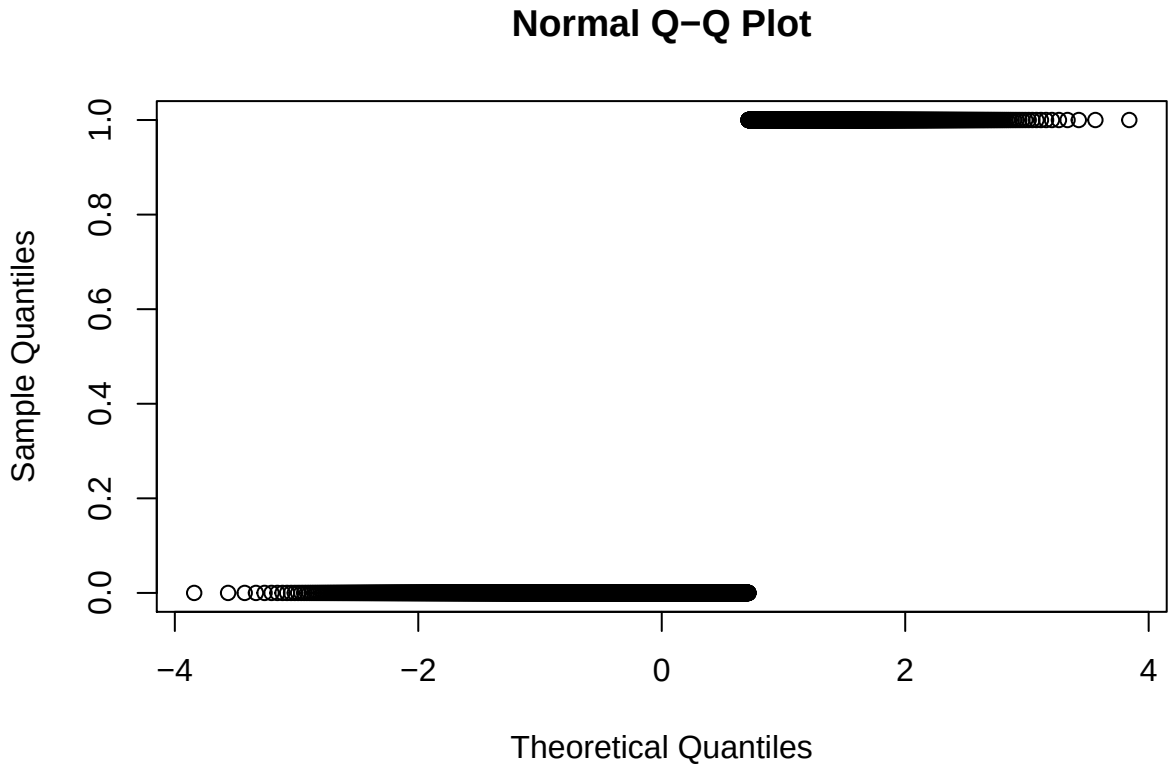
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1131  319
##        Yes   70  111
##
##                Accuracy : 0.7615
##                  95% CI : (0.74, 0.782)
##     No Information Rate : 0.7364
##     P-Value [Acc > NIR] : 0.01085
##
##                   Kappa : 0.2455
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9417
##             Specificity : 0.2581
##          Pos Pred Value : 0.7800
##          Neg Pred Value : 0.6133
##              Prevalence : 0.7364
##          Detection Rate : 0.6934
##    Detection Prevalence : 0.8890
##       Balanced Accuracy : 0.5999
##
##        'Positive' Class : No
##
```

```
plot(rpart_predict,residuals(tree.model.flag))
abline(h=0,lty=2,col="grey")
```

```
plot(predict(tree.model.flag)[,1], residuals(tree.model.flag))
lines(lowess(predict(tree.model.flag)[,1],residuals(tree.model.flag)),col="black",lwd=2)
```




```
qqnorm(residuals(tree.model.flag))
```



Tree-based model for TARGET_AMT

The QQ plot of the residuals vs response variable indicates a generally straight line.

```
qqplot(ins.train.AMT$TARGET_AMT, residuals(tree.model.flag2))
```