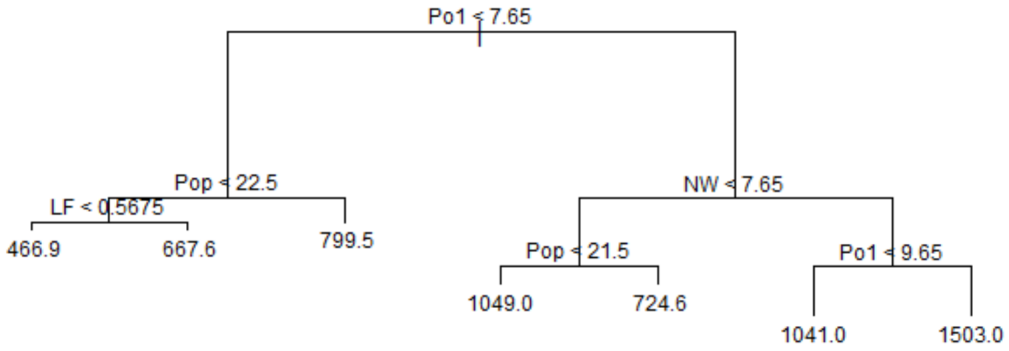
ISYE 6501 HW7

March 2021

1 Question 10.1

Since we are using the uscrime data again, I'll not review the data one more time. Using the Regression Tree method, I apply the tree model to all the variables and get a model used 4 variables ("Po1" "Pop" "LF" "NW") with 7 leaves and Residual mean deviance (mean of squared errors) of 47390. I tried to prune the tree and see if it improves the model, it turns out with 3 variables "Po1" "Pop" "NW" and 5 leaves with Residual mean deviance (sum of squared errors) of 54210. Prune the tree did not improve the model, so I will keep the original model. Prediction of the Crime rate for the new data point is 724.6, which is very different from linear regression result. From my point of view, I think the regression tree model is not a good approach for this case.



Using Random Forrest method, the model uses 5 variables in each tree and run a total of 100 trees. The Mean of squared residuals is 84798.7 which is much higher than the Regression tree model. The predicted value is 1170.733, which is much closer to the prediction made by linear regression in HW6. Even though the Regression tree model has a better fit, I would still say that the Random Forrest is a better approach. Not only that the predicted result is better, but also because the Random Forrest method eliminates bias and reduces over-fitting issue.

2 Question 10.2

I had a project, that I'm taking the features of music and determine how likely is it going to be a popular music. The popular music is defined using a data set that contains 10 year data of Billboard top 50 music for the year. Logistic regression is appropriate for this project since it could give me the possibility in percentage. Predictors are the music features, like loudness, tempo, beat, rap, genre, etc.

2.1 My R code for Question 10.1 is:

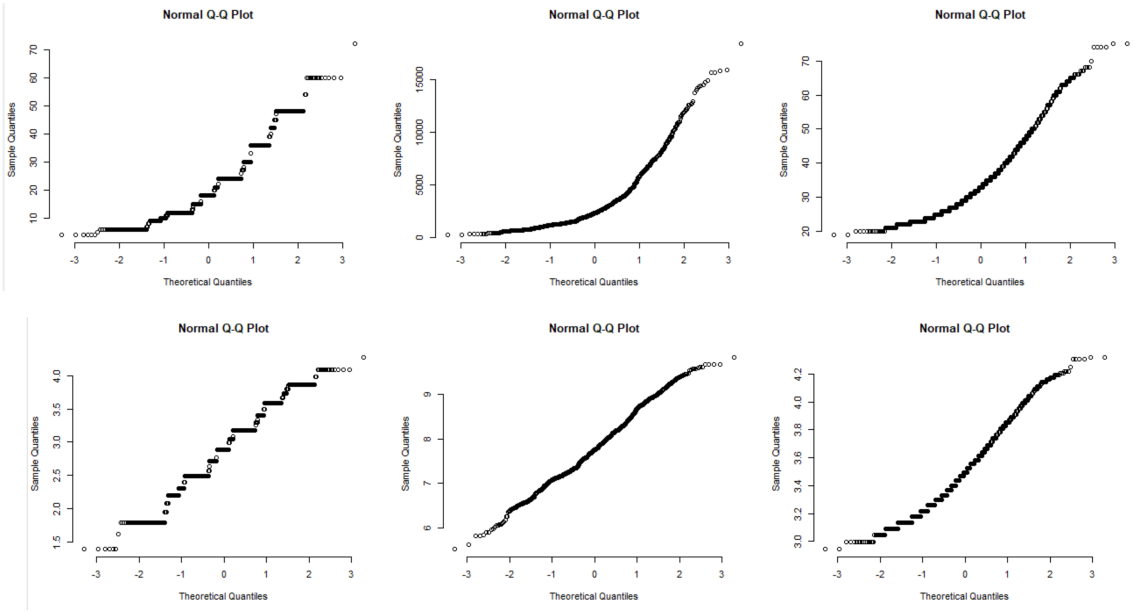
```
library(tree)
library(MASS)
library("randomForest")
set.seed(416)

df <- read.table("uscrime.txt", sep = '\t', stringsAsFactors = FALSE, header = TRUE)
head(df)

newdf <- data.frame (M = 14.0, So = 0, Ed = 10.0, Po1 = 12.0,
                     Po2 = 15.5, LF = 0.640, M.F = 94.0,
                     Pop = 150, NW = 1.1, U1 = 0.120,
                     U2 = 3.6, Wealth = 3200,
                     Ineq = 20.1, Prob = 0.04, Time = 39.0)

# Regression Tree model
reg.tree <- tree(Crime ~., data=df)
plot(reg.tree)
text(reg.tree, cex=.75)
summary(reg.tree)

# prune tree, see if model imporved
prune.tree=prune.tree(reg.tree ,best=5)
plot(prune.tree)
text(prune.tree , pretty =0)
summary(prune.tree)

# prediction
yhat.tree = predict (reg.tree ,newdata=newdf)
yhat.tree


# Random Forrest model
rdm.frt <- randomForest(Crime ~., data=df, importance=TRUE, do.trace=100, ntree=100)
print(rdm.frt)

# prediction
yhat.rf = predict (rdm.frt ,newdata=newdf)
yhat.rf
```

3 Question 10.3.1 logistic Regression

Loading the German Credit Card data and review the summary of the data. From the data description I know that V1 V3 V4 V6 V7 V9 V10 V12 V14 V15 V17 V19 V20 are all qualitative variables. Reviewing other numerical variables, it seems that the variance is kind of large for V2 V5 V13. So I draw qq-plots for the 3 variables then applied log transform on them, all of them got approved.

```
     V1                 V2           V3                 V4
Length:1000        Min.   : 4.0   Length:1000        Length:1000
Class :character   1st Qu.:12.0   Class :character   Class :character
Mode  :character   Median :18.0   Mode  :character   Mode  :character
                   Mean   :20.9
                   3rd Qu.:24.0
                   Max.   :72.0
      V5             V6                 V7                 V8
Min.   :  250   Length:1000        Length:1000        Min.   :1.000
1st Qu.: 1366   Class :character   Class :character   1st Qu.:2.000
Median : 2320   Mode  :character   Mode  :character   Median :3.000
Mean   : 3271                                         Mean   :2.973
3rd Qu.: 3972                                         3rd Qu.:4.000
Max.   :18424                                         Max.   :4.000
     V9                V10                V11            V12
Length:1000        Length:1000        Min.   :1.000   Length:1000
Class :character   Class :character   1st Qu.:2.000   Class :character
Mode  :character   Mode  :character   Median :3.000   Mode  :character
                                      Mean   :2.845
                                      3rd Qu.:4.000
                                      Max.   :4.000
     V13            V14                V15                V16
Min.   :19.00   Length:1000        Length:1000        Min.   :1.000
1st Qu.:27.00   Class :character   Class :character   1st Qu.:1.000
Median :33.00   Mode  :character   Mode  :character   Median :1.000
Mean   :35.55                                         Mean   :1.407
3rd Qu.:42.00                                         3rd Qu.:2.000
Max.   :75.00                                         Max.   :4.000
    V17                V18           V19                V20
Length:1000        Min.   :1.000   Length:1000        Length:1000
Class :character   1st Qu.:1.000   Class :character   Class :character
Mode  :character   Median :1.000   Mode  :character   Mode  :character
                   Mean   :1.155
                   3rd Qu.:1.000
                   Max.   :2.000
     V21
Min.   :1.0
1st Qu.:1.0
Median :1.0
Mean   :1.3
3rd Qu.:2.0
Max.   :2.0
```

V2 V5 V13 Before and After transform respectively



To prepare for the logistic regression, I have also changed the response variable from 1 2 to 0 1, where 0 represents bad and 1 represents good. Since there are 1000 data in the data set, I will split the set into train and test sets, where 70% in training set and 30% in testing set. Then fit the model and use stepAIC to select the best model which uses V1 V2 V3 V4 V6 V7 V8 V10 V13 V20 to predict response. The coefficients are as shown below:

```
Call:
glm(formula = V21 ~ V1 + V2 + V3 + V4 + V6 + V7 + V8 + V10 +
    V13 + V20, family = binomial, data = training_dataset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0978  -0.6983  -0.3553   0.6618   2.5613

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.024241   1.695527   1.194 0.232528
V1A12       -0.436644   0.256056  -1.705 0.088144 .
V1A13       -1.397902   0.470611  -2.970 0.002974 **
V1A14       -1.898708   0.283387  -6.700 2.08e-11 ***
V2           0.863112   0.199762   4.321 1.56e-05 ***
V3A31        0.542431   0.621116   0.873 0.382490
V3A32       -0.571921   0.479411  -1.193 0.232883
V3A33       -0.916558   0.570186  -1.607 0.107951
V3A34       -1.343217   0.512358  -2.622 0.008751 **
V4A41       -1.325358   0.415127  -3.193 0.001410 **
V4A410      -1.271815   0.839401  -1.515 0.129735
V4A42       -0.800039   0.308237  -2.596 0.009444 **
V4A43       -1.139149   0.306909  -3.712 0.000206 ***
V4A44       -1.875527   1.233908  -1.520 0.128514
V4A45       -0.268003   0.611249  -0.438 0.661059
V4A46        0.705965   0.452612   1.560 0.118818
```

```
V4A48       -15.464248 493.656573  -0.031 0.975010
V4A49        -0.998269   0.403783  -2.472 0.013425 *
V6A62        -0.252232   0.338588  -0.745 0.456301
V6A63        -0.211510   0.498929  -0.424 0.671618
V6A64        -1.602331   0.727103  -2.204 0.027544 *
V6A65        -1.075308   0.325677  -3.302 0.000961 ***
V7A72         0.281136   0.482789   0.582 0.560354
V7A73        -0.234377   0.453049  -0.517 0.604924
V7A74        -0.759562   0.499474  -1.521 0.128329
V7A75         0.001513   0.459764   0.003 0.997374
V8            0.162207   0.094617   1.714 0.086463 .
V10A102       0.475178   0.495914   0.958 0.337969
V10A103      -0.960834   0.481318  -1.996 0.045906 *
V13          -0.959106   0.393057  -2.440 0.014682 *
V20A202      -1.022004   0.708166  -1.443 0.148973
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 835.74  on 699  degrees of freedom
Residual deviance: 611.73  on 669  degrees of freedom
AIC: 673.73

Number of Fisher Scoring iterations: 14
```

With a default threshold of 0.5 (response greater than 0.5 considered as 1, response smaller than 0.5 considered as 0), the model has an accuracy of 0.7971. The Confusion Matrix is:

```
          Reference
Prediction   0   1
         0 455  96
         1  46 103
```

4 Question 10.3.2 logistic Regression

Estimate bad as good is 5 times worse than estimate good as bad, which means we need to increase our threshold in order to reduce cost. The cost will be none for True Positive (TP) and Ture Negative (TN). Set cost for False Negative (FN, falsely estimate good as bad) to 1, and set cost for False Positive (FP, falsely estimate bad as good) as 5. Then the cost for incorrect prediction is: Cost = FN + FP x 5.
My goal is to minimize the Cost and try to maximize TP in the same time. I did an analyze on raising threshold from 0.5 to 0.9 and the results are shown in table below. We can see that when increase threshold to 0.8 the Cost is smallest and we also obtain a maximum on TP. It looks like 0.8 will be good for threshold.

Threshold	0.5	0.6	0.7	0.8	0.9
Cost	326	261	220	198	198
True Positive	103	73	54	26	1
True Negative	455	474	486	496	501
False Positive	46	27	15	5	0
False Positive	96	126	145	173	198

Now I will see how this model performs in testing set. Run the model in testing data set, I got the Cost of 107 with 3 FP, 92 FN, 196 TN and, 9 TPs. The accuracy rate is 0.6833; it's lower than training data as expected.
I would say this model with a threshold of 0.8 works good on minimizing cost and maximize True Positive in the same time for this credit card case.

4.1 My R code is:

```
library(caret)
library(MASS)
library(dplyr)
library('e1071')
library(ROCR)

set.seed(416)

# read data
df <- read.table("germancredit.txt", sep = "", stringsAsFactors = FALSE, header = FALSE)
head(df)
summary(df)

# draw qq plot
par(mfrow=c(1,3))
qqnorm(df$V2, pch = 1, frame = FALSE)
qqnorm(df$V5, pch = 1, frame = FALSE)
qqnorm(df$V13, pch = 1, frame = FALSE)

# draw transformed data
par(mfrow=c(1,3))
qqnorm(log(df$V2), pch = 1, frame = FALSE)
qqnorm(log(df$V5), pch = 1, frame = FALSE)
qqnorm(log(df$V13), pch = 1, frame = FALSE)

# log transform of V2 V5 V13
head(df)
df$V2=log(df$V2)
df$V5=log(df$V5)
df$V13=log(df$V13)
df$V21=df$V21-1
head(df)

# split data into train and test
datasplit <- createDataPartition(df[,21], 1, p=0.7, list=FALSE)
training_dataset <- df[datasplit, ]
testing_dataset <- df[-datasplit, ]

# fit log regression
logfit <- glm(V21~.,data = training_dataset, family=binomial)
summary(logfit)

# Stepwise regression model
AIC_fit <- stepAIC(logfit, direction = "both", trace = FALSE)
summary(AIC_fit)

#check accuracy
yhat <- predict(AIC_fit ,newdata=training_dataset, type = "response")
training_dataset$yhat = yhat
head(training_dataset)

# Set threshold to 0.5
```

```
training_dataset <- training_dataset  %>% mutate(predict = 1*(yhat > 0.5) + 0)
cm_train5 <- confusionMatrix(data = factor(training_dataset$predict),
                             reference = factor(training_dataset$V21))
# False Positive cm$table[2,1]  &  False Negative cm$table[1,2]
cost_train5 = cm_train5$table[1,2]+cm_train5$table[2,1]*5
cost_train5
cm_train5

# Set threshold to 0.6
training_dataset <- training_dataset  %>% mutate(predict = 1*(yhat > 0.6) + 0)
cm_train6 <- confusionMatrix(data = factor(training_dataset$predict),
                             reference = factor(training_dataset$V21))
# False Positive cm$table[2,1]  &  False Negative cm$table[1,2]
cost_train6 = cm_train6$table[1,2]+cm_train6$table[2,1]*5
cost_train6
cm_train6

# Set threshold to 0.7
training_dataset <- training_dataset  %>% mutate(predict = 1*(yhat > 0.7) + 0)
cm_train7 <- confusionMatrix(data = factor(training_dataset$predict),
                             reference = factor(training_dataset$V21))
# False Positive cm$table[2,1]  &  False Negative cm$table[1,2]
cost_train7 = cm_train7$table[1,2]+cm_train7$table[2,1]*5
cost_train7
cm_train7

# Set threshold to 0.8
training_dataset <- training_dataset  %>% mutate(predict = 1*(yhat > 0.8) + 0)
cm_train8 <- confusionMatrix(data = factor(training_dataset$predict),
                             reference = factor(training_dataset$V21))
# False Positive cm$table[2,1]  &  False Negative cm$table[1,2]
cost_train8 = cm_train8$table[1,2]+cm_train8$table[2,1]*5
cost_train8
cm_train8

# Set threshold to 0.9
training_dataset <- training_dataset  %>% mutate(predict = 1*(yhat > 0.9) + 0)
cm_train9 <- confusionMatrix(data = factor(training_dataset$predict),
                             reference = factor(training_dataset$V21))
# False Positive cm$table[2,1]  &  False Negative cm$table[1,2]
cost_train9 = cm_train9$table[1,2]+cm_train9$table[2,1]*5
cost_train9
cm_train9



# predicting & accuaracy in testing set
yhat_test <- predict(AIC_fit ,newdata=testing_dataset, type = "response")
testing_dataset <- testing_dataset  %>% mutate(predict = 1*(yhat_test  > 0.8) + 0)
cm_test <- confusionMatrix(data = factor(testing_dataset$predict),
                           reference = factor(testing_dataset$V21))
cost_test = cm_test$table[1,2]+cm_test$table[2,1]*5
cost_test
cm_test
```