ISYE 6501 Intro Analytics Modeling – HW7

Question 10.1 Using the same crime data set `uscrime.txt` as in Questions 8.2 and 9.1, find the best model you can using (a) a regression tree model, and (b) a random forest model. In R, you can use the `tree` package or the `rpart` package, and the `randomForest` package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don't just stop when you have a good model, but interpret it too).

Firstly, I randomly split the data into training (70%) and test set (30%). I created a regression tree using the `rpart` package, because the training set only have 31 data points. I got a very simple model, only one variable-Po1 is used. I use this model to predict the test set by returning the mean response at the node level. I got MSE_Test: 127137.9.

Next step, I tried to tune the model by specify minsplit = 5 and maxdepth = 12. This time, I got a more complicate model, using 4 variables. But the model predict almost every test data point to one group, the MSE_Test: 153470.5 is also get higher.*(Fig.1)*

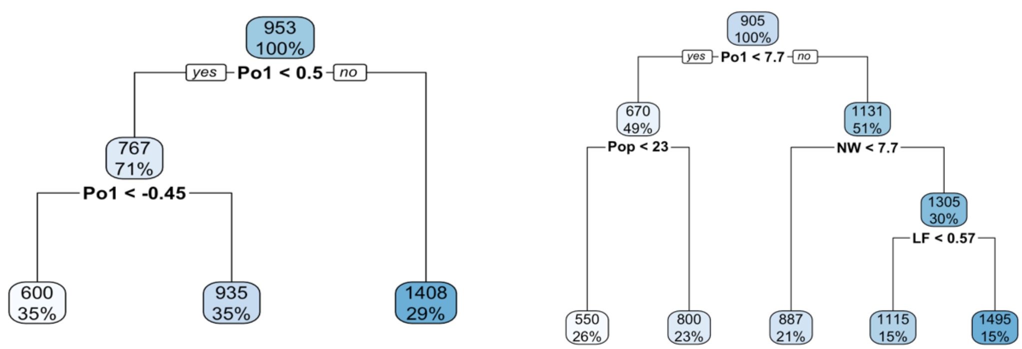

Fig 1

Using the same training set, I created a random forest model, when the number of trees equal to 75. I got the smallest MSE_Train: 76964.01 and the MSE_Test is 62481.18 which is way better than the regression tree model. As we can see from fig.2 overfitting seems not a problem at the Random Forest model. Unlike regression tree, random forest using all variables with different importance.

```
> rf_model$importance
       IncNodePurity
M         112898.397
So          8021.158
Ed        202263.738
Po1      1144847.103
Po2       919061.793
LF        227164.232
M.F       249621.826
Pop       175220.006
NW        435120.516
U1         93019.201
U2         87479.622
Wealth    585997.220
Ineq      130178.429
Prob      386096.686
Time       90196.566
```

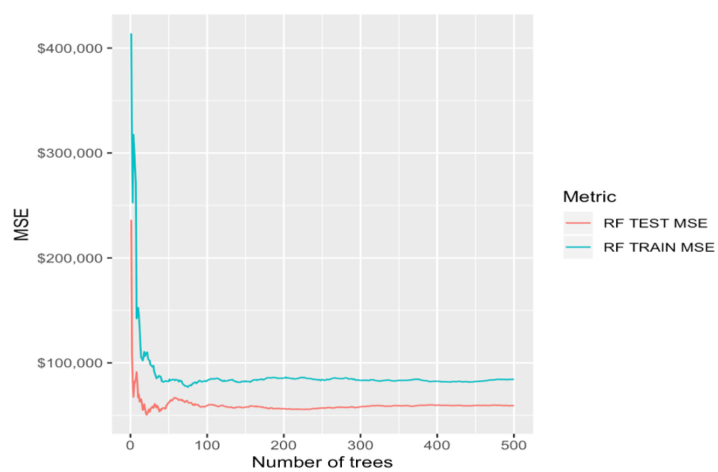

Fig 2

I tried to tune the Random forest model by testing for the best mtry variable (Number of variables randomly sampled as candidates at each split) by default its # of variable/3 for the regression model, which is 5 at previous model. Using the *tuneRF* function we can see when mtry=6, the error is smaller for the training set. But at the smallest MSE_Train 75642.65, I get the MSE_Test 65866.72 which is slightly bigger than the original model. *(Fig.3)*

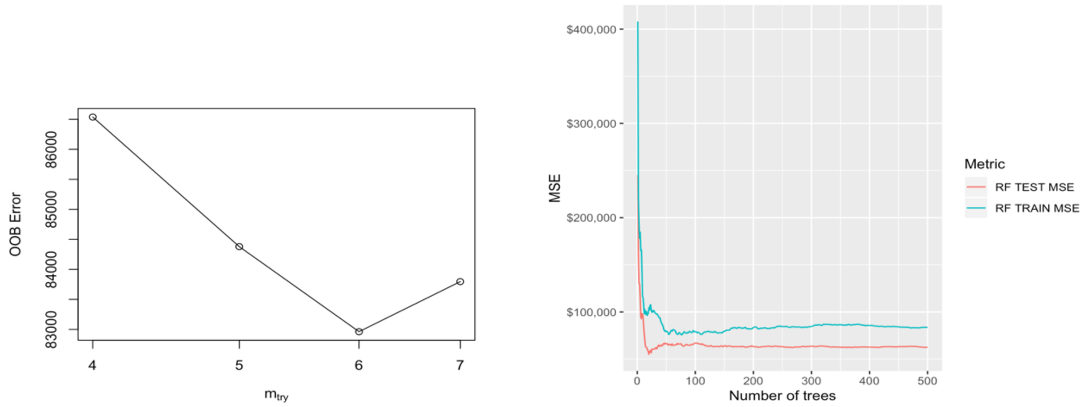

Fig.3

Overall random forest gives a very good performance. Decent MSE and no sign for overfitting even with a very small dataset. But it's hard to get any insight form the model, the only thing we can see about the variables is the overall importance, no able to see the variable interaction.

However, the regression tree is very interpretable and easy to understand. But it has poor predictive accuracy, especially with such a small dataset.

Question 10.2 Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.
Predict the likelihood of admission to Georgia tech graduate school.
Predictors:

- GRE Scores (0-340)
- Recommendation Letters Strength (0-5)
- University Rating (0-5)
- Undergraduate GPA (0-4)
- Research Experience (Binary 0 or 1)

Question 10.3
1. Using the GermanCredit data set `germancredit.txt` from http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german / (description at http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the `glm` function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use `family=binomial(link="logit")` in your `glm` function call.

The first step is to read in the dataset and do some simple data summary. From the code, I get to know this dataset contains 1,000 data points, 20 predictors, and one response with 2 categories (1-70% or 2-30%). I scaled three variables -"V2","V5","V13", which have nlevel higher than 30 and adjust the response to 0 and 1. I also randomly split the data into training (70%) and test set (30%). Then I fit the train set to get the mode:

```
Call:  glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train2)
Coefficients:
(Intercept)       V1A12       V1A13       V1A14       V3A31       V3A32       V3A33       V3A34
   1.083211   -0.352028   -1.498818   -1.667980    0.325185   -0.620800   -0.753585   -1.633994
      V4A41      V4A410       V4A42       V4A43       V4A44       V4A45       V4A46       V4A48
  -1.887654   -1.681220   -0.643308   -0.973516   -0.815840   -0.414114   -0.044210  -15.662076
      V4A49       V6A62       V6A63       V6A64       V6A65       V7A72       V7A73       V7A74
  -0.742853   -0.336720    0.033593   -2.282201   -0.873686   -0.264708   -0.632523   -1.468184
      V7A75          V8       V9A92       V9A93       V9A94     V10A102     V10A103         V11
  -0.529125    0.315001   -0.183325   -0.613230    0.002749    0.506374   -0.760330   -0.034933
    V12A122     V12A123     V12A124     V14A142     V14A143     V15A152     V15A153         V16
  -0.014121    0.054116    0.643360    0.403094   -0.530283   -0.362192   -1.049015    0.624281
    V17A172     V17A173     V17A174         V18     V19A192     V20A202          V2          V5
   0.708998    0.635778    0.711153   -0.147032   -0.334658   -0.820270    0.449962    0.378930
        V13
  -0.276036
Degrees of Freedom: 658 Total (i.e. Null); 610 Residual
Null Deviance    807.3
Residual Deviance: 580.2
AIC: 678.2
```

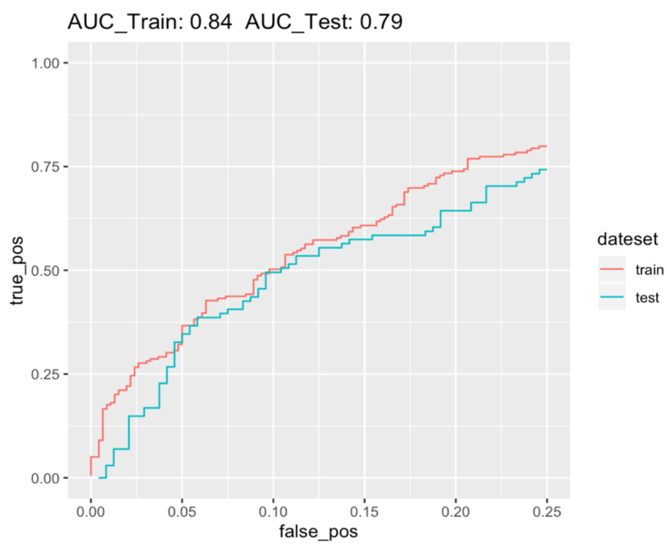

FIG.4

2. Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between “good” and “bad” answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.

In this step, I tried threshold from 0.1 to 0.9 and did a summary for TPR, TNR, ERROR, and Cost for the test set. From figure 5, we can see as we increase the threshold, TPR keeps decreasing while TNR keeps increasing. So, we need to find a good balance by taking the cost into consideration. Incorrectly identifying a bad customer as good which is the False Positive.

Here we assume the cost for one FN is 1, and FP is 5. Then we have a total cost for different thresholds - FN+FP*5. Since the FP cost is 5 times worse than FN, a higher threshold can make more people categorize as negative. But in the real case, we don’t want to categorize everyone to “bad” answer, so if we know how many we can “earn” but correctly identifying a customer, we can better select a good threshold. Here since we don't have that, I will just select 0.7 here, since the error is not very high, and we also get a low cost.

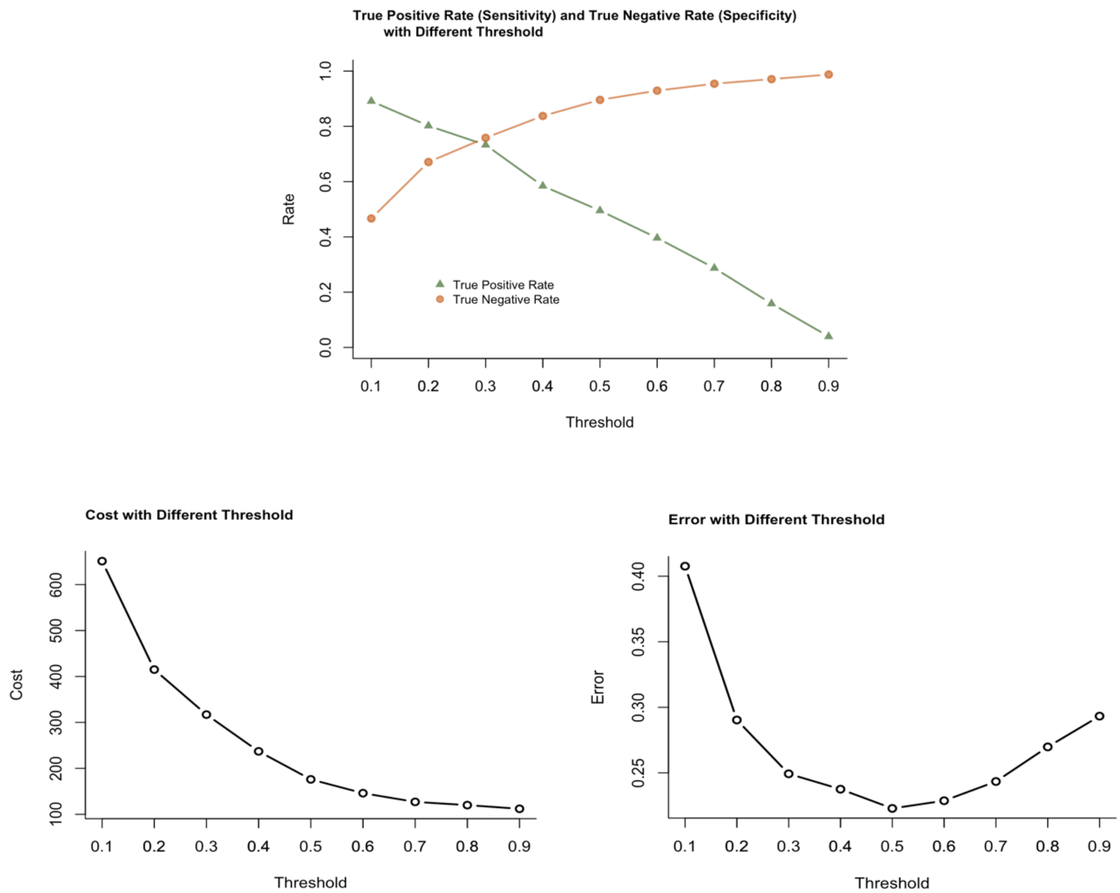

FIG.5

Code

```
# ISYE 6501 Intro Analytics Modeling - HW7
# IP uscrime.txt germancredit.txt

#Question 10.1 Using the same crime data set uscrime.txt as in Questions 8.2 
and 9.1,
#Find the best model you can using (a) a regression tree model, and (b) a ran
dom forest model.
library(randomForest)
library(tree)
library(rpart)
library(rpart.plot)
library(tibble)
library(ggplot2)
library(dplyr)
library(tidyverse)
```

```
df<-read.delim("uscrime.txt", header = TRUE, sep = "\t")
#scale
fn <- function(x) scale(x,  scale = TRUE)
df_scaled<-as.data.frame(lapply(df[,-16], fn))
df_scaled$Crime<-df$Crime

#splite train and test
set.seed(666)
g <- sample(1:2,size=nrow(df_scaled),replace=TRUE,prob=c(0.7,0.3))
train <- df_scaled[g==1,]
test <- df_scaled[g==2,]

# Fit model:regression tree
set.seed(123)
reg_tree <- rpart(
  formula = Crime ~ .,
  data    = train,
  method  = "anova"
)
rpart.plot(reg_tree)
plotcp(reg_tree)

predict(reg_tree, test, type = 'vector')
test_residials<-predict(reg_tree, test, type = 'vector')-test$Crime
MSE_test_reg<-mean(test_residials^2) #MSE Train

# Tune model:regression tree
rgt_tune <- rpart(
  formula = Crime ~ .,
  data    = df,
  method  = "anova",
  control = list(minsplit = 5, maxdepth = 12, xval = 2)
)
rpart.plot(rgt_tune)
plotcp(rgt_tune)

predict(rgt_tune, test, type = 'vector')
test_residials2<-predict(rgt_tune, test, type = 'vector')-test$Crime
MSE_test_rgt<-mean(test_residials2^2) #MSE Train

# Fit glm model: random forest
set.seed(123)
rf_model <- randomForest(
  formula = Crime ~ .,
  data    = train,
  xtest   = test[,-16],
  ytest   = test[,16]
)
```

```r
rf_model$importance
which.min(rf_model$mse)
rf_model$mse[which.min(rf_model$mse)]
rf_model$test$mse[which.min(rf_model$mse)]

MSE_train_rf <- rf_model$mse
MSE_test_rf <- rf_model$test$mse

#plot error
# compare error
tibble::tibble(
  `RF TRAIN MSE`  = MSE_train_rf,
  `RF TEST MSE`  = MSE_test_rf,
  ntrees = 1:rf_model$ntree
) %>%
  gather(Metric, MSE, -ntrees) %>%
  ggplot(aes(ntrees, MSE, color = Metric)) +
  geom_line() +
  scale_y_continuous(labels = scales::dollar) +
  xlab("Number of trees")


##tuning mtry
set.seed(123)
rf_tune<- tuneRF(
  x          = train[,-16],
  y          = train[,16],
  ntreeTry   = 500,
  mtryStart  = 5,
  stepFactor = 1.3,
  improve    = 0.01,
  trace      = FALSE      # to not show real-time progress
)

set.seed(123)
rf_model2 <- randomForest(
  formula = Crime ~ .,
  data    = train,
  xtest   = test[,-16],
  ytest   = test[,16],
  mtry=6
)

which.min(rf_model2$mse)
rf_model2$mse[which.min(rf_model2$mse)]
rf_model2$test$mse[which.min(rf_model2$mse)]

MSE_train_rf2 <- rf_model2$mse
MSE_test_rf2<- rf_model2$test$mse
```

```r
#plot error
# compare error
tibble::tibble(
  `RF TRAIN MSE` = MSE_train_rf2,
  `RF TEST MSE` = MSE_test_rf2,
  ntrees = 1:rf_model2$ntree
) %>%
  gather(Metric, MSE, -ntrees) %>%
  ggplot(aes(ntrees, MSE, color = Metric)) +
  geom_line() +
  scale_y_continuous(labels = scales::dollar) +
  xlab("Number of trees")

#Question 10.3 Using the GermanCredit data set, use logistic regression
# 10.3.1. Find a good predictive model for whether credit applicants are good
credit risks or not.
df2<-read.delim("germancredit.txt", header = FALSE, sep = " ")
#### Display Head Lines ####
head(df2,2)
#### Show summary, number of row of last column##
nrow(df2)
sum(df2$V21-1)/nrow(df2)
ncol(df2)
for (i in attributes(df2)$names ){
  print(paste0('Name:',i,'   Class:',class(df2[[i]]),'   nlevels:',length(uni
que(df2[[i]]))))
  print(summary(df2[[i]]))
}
sapply(df2,function(x) sum(is.na(x)))

numerc_var<-c("V2","V5","V13")
fn <- function(x) scale(x,  scale = TRUE)
f_scaled<-as.data.frame(lapply(df2[,numerc_var], fn))
f_scaled$V21=df2$V21-1
df2_scaled<-cbind(subset(df2, select = -c(V2,V5,V13,V21)),f_scaled)

#splite train and test
set.seed(666)
g2 <- sample(1:2,size=nrow(df2_scaled),replace=TRUE,prob=c(0.7,0.3))
train2 <- df2_scaled[g==1,]
test2 <- df2_scaled[g==2,]

library(pROC)
cal_ROC <- function(y_pred, y_real, dateset=NULL)
{ outcome <- as.numeric(factor(y_real))-1
  pos <- sum(outcome) # total known positives
  neg <- sum(1-outcome) # total known negatives
  pos_probs <- outcome*y_pred # probabilities for known positives
```

```r
  neg_probs <- (1-outcome)*y_pred # probabilities for known negatives
  true_pos <- sapply(y_pred,
                     function(x) sum(pos_probs>=x)/pos) # true pos. rate
  false_pos <- sapply(y_pred,
                     function(x) sum(neg_probs>=x)/neg)
  if (is.null(dateset))
    result <- data.frame(true_pos, false_pos)
  else
    result <- data.frame(true_pos, false_pos, dateset)
  result %>% arrange(false_pos, true_pos)
}

# Fit glm model: gaussian model
logit_model<-glm(V21~.,family = binomial(link = "logit"),train2)
logit_test_pred<-predict(logit_model,test2,type="response")
ROC.train <- cal_ROC(y_pred=logit_model$fitted.values,
                      y_real=train2$V21,
                      dateset="train")
ROC.test <- cal_ROC(y_pred=logit_test_pred,
                     y_real=test2$V21,
                     dateset="test")
ROCs <- rbind(ROC.train, ROC.test)
auc_test<-auc(test2$V21,logit_test_pred)
auc_train<-auc(train2$V21,logit_model$fitted.values)
ggplot(ROCs, aes(x=false_pos, y=true_pos, color=dateset)) +
  geom_line() + xlim(0, 0.25)+
  ggtitle(paste0("AUC_Train: ",round(auc_train, digits = 2),"  AUC_Test: ",ro
und(auc_test, digits = 2)))

# 10.3.2. Determine a good threshold probability based on your model.

summary_table <- data.frame(row.names=paste0("th",c(0.1,0.2,0.3,0.4,0.5,0.6,0
.7,0.8,0.9)),
                            threshold= c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)
)
for (th in c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)){
      col<-paste0("th",th)
      fitted<-(logit_test_pred>=th)
      table(test2$V21,fitted)
      summary_table[col,"tn"] = table(test2$V21,fitted)[1]
      summary_table[col,"fn"] = table(test2$V21,fitted)[2]
      summary_table[col,"fp"] = table(test2$V21,fitted)[3]
      summary_table[col,"tp"] = table(test2$V21,fitted)[4]
}
summary_table$P=summary_table$tp+summary_table$fp
summary_table$N=summary_table$tn+summary_table$fn
summary_table$TPR=summary_table$tp/(summary_table$tp+summary_table$fn)
summary_table$TNR=summary_table$tn/(summary_table$tn+summary_table$fp)
summary_table$Error=(summary_table$fp+summary_table$fn)/(summary_table$P+summ
ary_table$N)
```

```
summary_table$Cost=summary_table$fp*5+summary_table$fn

plot(summary_table$TPR~summary_table$threshold , type="b" , bty="l" , xlab="Threshold" ,
     ylab="Rate" , col=rgb(0.2,0.4,0.1,0.7) , lwd=2 , pch=17, ylim=c(0,1))+
  lines(summary_table$TNR~summary_table$threshold, col=rgb(0.8,0.4,0.1,0.7),lwd=2,pch=19,type="b")+
  axis(side=1, at=seq(0.1, 0.9, by=0.1), labels =c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9) )+
  title("True Positive Rate (Sensitivity) and True Negative Rate (Specificity)
        with Different Threshold",adj =0,cex.main=0.9)+
  legend("bottomleft",
         legend = c("True Positive Rate", "True Negative Rate"),
         col = c(rgb(0.2,0.4,0.1,0.7),
                 rgb(0.8,0.4,0.1,0.7)),
         pch = c(17,19),
         bty = "n",
         pt.cex = 1,
         cex = 0.8,
         text.col = "black",
         horiz = F ,
         inset = c(0.15, 0.15))

plot(summary_table$Error~summary_table$threshold, type="b" , bty="l" , xlab="Threshold" ,
     ylab="Error" , lwd=2  )+
  axis(side=1, at=seq(0.1, 0.9, by=0.1), labels =c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9) )+
  title("Error with Different Threshold",adj =0,cex.main=0.9)

plot(summary_table$Cost~summary_table$threshold, type="b" , bty="l" , xlab="Threshold" ,
     ylab="Cost" , lwd=2  )+
  axis(side=1, at=seq(0.1, 0.9, by=0.1), labels =c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9) )+
  title("Cost with Different Threshold",adj =0,cex.main=0.9)
```