USING DECISION TREE AND RANDOM FOREST TO CLASSIFY BANKING CUSTOMERIN R (WEEK 3 ASSIGNMENT)

Course: ALY 6020

Name: Yuanying Li

1.INTRODUCTION

Precision marketing is a marketing technique that suggests successful marketing is to retain, cross-sell and upsell existing customers. Precision marketing is directed at existing customers to encourage brand loyalty and spur buying behavior. Precision marketing relies less on creating persuasive ads, and more on creating deals, offers, and gimmicks that will appeal to existing customers.

As a bank, increase profit and reduce unnecessary expenditure is one of the issues for running a bank. So it is very important to use precision marketing. For example, to expand the business, a bank needs to know what kind of clients would active their credit card so that a bank could target marketing incentive to those most likely to activate and use for their business transactions. On the contrary, a bank also want to isolate the cards that would likely never be activated to reduce wasted marketing spend.

In this project, we will investigate the utility of machine learning for detecting active or inactive credit card client applying the Decision Tree to define marketing strategies from credit card usage behavior of customers.

2. DATA AND PREPROCESSING

2.1 Define the decision variables

We will utilize the usage behavior of about 9000 active credit card holders during the last 6 month from Kaggle.

Here is the explanation of features' name in dataset.

- RESULTS: define this customer is active or inactive card user.
- CUST*ID* : Identification of Credit Card holder (Categorical)
- BALANCE : Balance amount left in their account to make purchases
- BALANCEFREQUENCY : How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated)
- PURCHASES : Amount of purchases made from account
- ONEOFF*PURCHASES* : Maximum purchase amount done in one-go
- INSTALLMENTSPURCHASES : Amount of purchase done in installment
- CASH*ADVANCE* : Cash in advance given by the user

- PURCHASESFREQUENCY : How frequently the Purchases are being made, score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased)
- ONEOFFPURCHASESFREQUENCY : How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased)
- PURCHASESINSTALLMENTSFREQUENCY : How frequently purchases in installments are being done (1 = frequently done, 0 = not frequently done)
- CASHADVANCEFREQUENCY : How frequently the cash in advance being paid
- CASHADVANCETRX : Number of Transactions made with "Cash in Advanced"
- PURCHASES*TRX* : Numbe of purchase transactions made
- CREDITLIMIT : Limit of Credit Card for user
- PAYMENTS : Amount of Payment done by user
- MINIMUM_PAYMENTS : Minimum amount of payments made by user
- PRCFULLPAYMENT : Percent of full payment paid by user
- TENURE : Tenure of credit card service for usersmoothness (local variation in radius lengths)

Based on these names, it is unlikely to know how each relates to active or inactive. These patterns will be revealed as we continue in the machine learning process.

2.2 Exploratory Data Analysis

Let's explore the data and see whether we can shine some light on the relationships. In doing so, we will prepare the data for use with the Decision Tree. We'll begin by importing the CSV data file, as we have done, saving the data to the cc_df data frame:

Code snippets:

```
#Load the "CC GENERAL_updated_final_version" data and assign it to cc_df
cc_df <- read.csv("CC GENERAL_updated_final_version.csv",header = TRUE)
View(cc_df)


#Prodide information about the structure of credit card data set
#This data set contains 8920 observation and 20 variables
str(cc_df)

#Check Null values Number
sum(is.na(cc_df))
```

Results:

	RESULTS	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMEI
1	INACTIVE	C10001	40.900749	0.818182	95.40	0.00	95.40
2	INACTIVE	C10002	3202.467416	0.909091	0.00	0.00	0.00

```
'data.frame':   8950 obs. of  20 variables:
 $ RESULTS                         : Factor w/ 2 levels "ACTIVE","INACTIVE": 2 2 :
 1 1 1 2 2 ...
 $ CUST_ID                         : Factor w/ 8950 levels "C10001","C10002",..: :
 4 5 6 7 8 9 10 ...
 $ BALANCE                         : num  40.9 3202.5 2495.1 1666.7 817.7 ...
 $ BALANCE_FREQUENCY               : num  0.818 0.909 1 0.636 1 ...
 $ PURCHASES                       : num  95.4 0 773.2 1499 16 ...
 $ ONEOFF_PURCHASES                : num  0 0 773 1499 16 ...
 $ INSTALLMENTS_PURCHASES          : num  95.4 0 0 0 0 ...
 $ CASH_ADVANCE                    : num  0 6443 0 206 0 ...
 $ ONEOFF_PURCHASES_FREQUENCY      : num  0 0 1 0.0833 0.0833 ...
 $ PURCHASES_INSTALLMENTS_FREQUENCY: num  0.0833 0 0 0 0 ...
 $ PURCHASES_FREQUENCY             : num  0.1667 0 1 0.0833 0.0833 ...
 $ CASH_ADVANCE_TRX                : int  0 4 0 1 0 0 0 0 0 0 ...
 $ PURCHASES_TRX                   : int  2 0 12 1 1 8 64 12 5 3 ...
 $ CREDIT_LIMIT                    : num  1000 7000 7500 7500 1200 1800 13500 230(
 11000 ...
 $ PAYMENTS                        : num  202 4103 622 0 678 ...
 $ MINIMUM_PAYMENTS                : num  140 1072 627 NA 245 ...
 $ PRC_FULL_PAYMENT                : num  0 0.222 0 0 0 ...
 $ TENURE                          : int  12 12 12 12 12 12 12 12 12 12 ...
 $ CASH_ADVANCE_FREQUENCY          : num  0 0.25 0 0.0833 0 ...
 $ USED_FREQUENCY                  : num  0.1667 0.25 1 0.1667 0.0833 ...
> #Check Null values Number
> sum(is.na(cc_df))
[1] 314
```

Observations:

1. Sample size has 8950 rows with 20 variables, the dependent variable is called RESULTS. There is 314 missing values in this dataset.

2. The second variable is an integer variable named CUST_ID. As this is simply a unique identifier (ID) for each customer in the data, it does not provide useful information, and we will need to exclude it from the model.

2.3 Data preprocessing

In this study we firstly focus on 19 factors that possibly influence result, as we mentioned before, CUST_ID is not necessary to keep for predicting the final result, so that we could just drop it and remove all null values.

```
> summary(cc_df)
     RESULTS          BALANCE          BALANCE_FREQUENCY    PURCHASES
 ACTIVE   :5215   Min.   :    0.0   Min.   :0.0000    Min.   :    0.00
 INACTIVE:3421   1st Qu.:  148.1   1st Qu.:0.9091    1st Qu.:   43.37
                  Median :  916.9   Median :1.0000    Median :  375.40
                  Mean   : 1601.2   Mean   :0.8950    Mean   : 1025.43
                  3rd Qu.: 2105.2   3rd Qu.:1.0000    3rd Qu.: 1145.98
                  Max.   :19043.1   Max.   :1.0000    Max.   :49039.57
 ONEOFF_PURCHASES   INSTALLMENTS_PURCHASES  CASH_ADVANCE
 Min.   :    0.00   Min.   :    0.00        Min.   :    0.0
 1st Qu.:    0.00   1st Qu.:    0.00        1st Qu.:    0.0
 Median :   44.99   Median :   94.78        Median :    0.0
```

Code snippets:

```
#Remove NA
cc_df <- na.omit(cc_df)
sum(is.na(cc_df))

#Drop the unnecessary columns of the dataframe
cc_df<-select(cc_df, -c(CUST_ID, USED_FREQUENCY, TENURE))
```

2.4 Date visualization

2.4.1 RESULTS analysis

RESULTS is of particular interest as it is the outcome we hope to predict. We could calculate the proportion of diagnosis result to see distribution of benign or malignant mass.

Code snippets:

```
#Active Rate
table(cc_df$RESULTS)
# Active rates in proportions
prop.table(table(cc_df$RESULTS))
```

Results:

```
  ACTIVE INACTIVE
    5215     3421
> # Active rates in proportions
> prop.table(table(cc_df$RESULTS))

   ACTIVE  INACTIVE
0.6038675 0.3961325
```

Observation:

1. The table() output indicates that 5215 customer are active while 3412 are inactive. Now, when we look at the prop.table() output, we notice that the values have been labeled active and inactive with 60.38 percent and 39.61 percent of the masses, respectively. it shows in real life, the percentage of inactive customer is not small, which mean this kind of customer may cause a lot of cost for a bank.

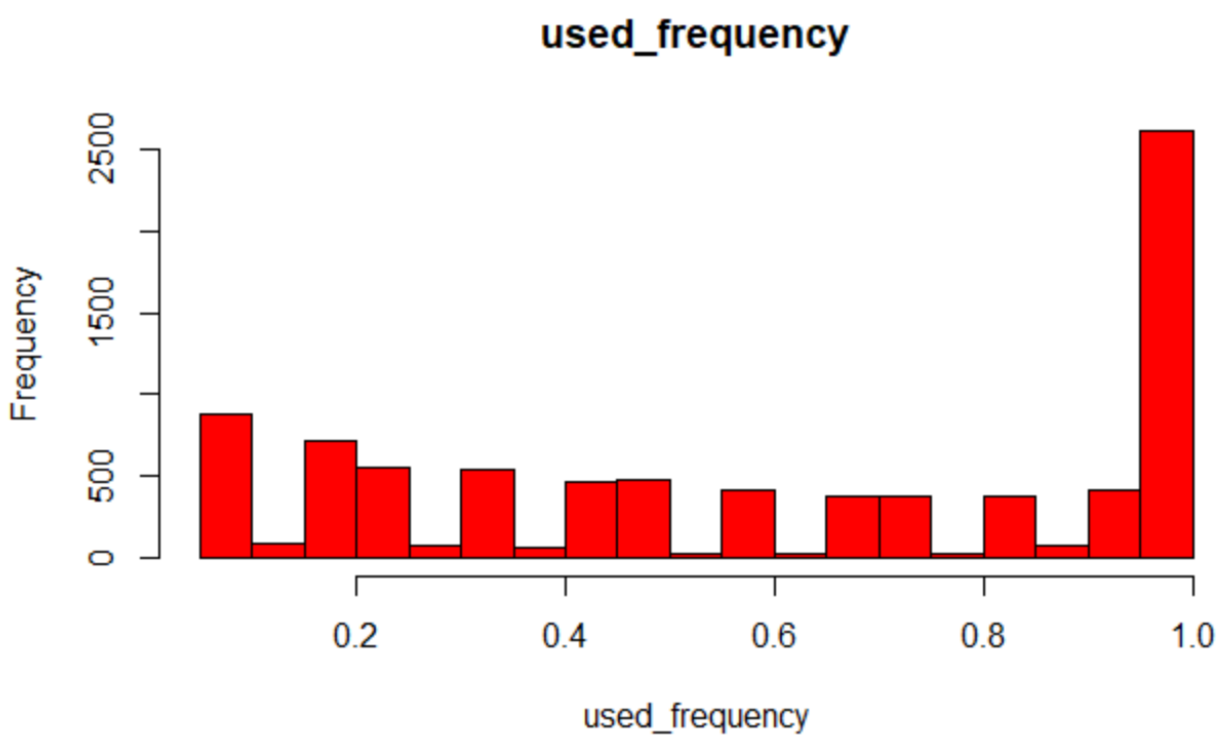
2.4.2 Summary Data and check distribution

Use the summary command to calculate each feature's mathematics detail and plot distribtion.

Code snippets:

```
#Data summary
summary(cc_df)
```

Results:



3.DECISION TREE

3.1 Split train dataset and test train

In this section, we can simulate this scenario by dividing our data into two portions: a training dataset that will be used to build the Decision Tree model and a test dataset that will be used to estimate the predictive accuracy of the model.

Using the data extraction methods, Managing and Understanding Data, we will split the data frame into credit_train and credit_test.

Code snippets:

```
#TRAIN, TEST & SPLIT
#Data splicing basically involves splitting
n <- nrow(cc_df)
n_train <- round(0.8*n)
n_train

#set seed
set.seed(2020)
train_indices <- sample(1:n, n_train)
credit_train <- cc_df[train_indices, ]
View(credit_train)
credit_test <- cc_df[-train_indices, ]
```

3.2.2 Train the model

Equipped with our training data and labels vector, we are now ready to apply.

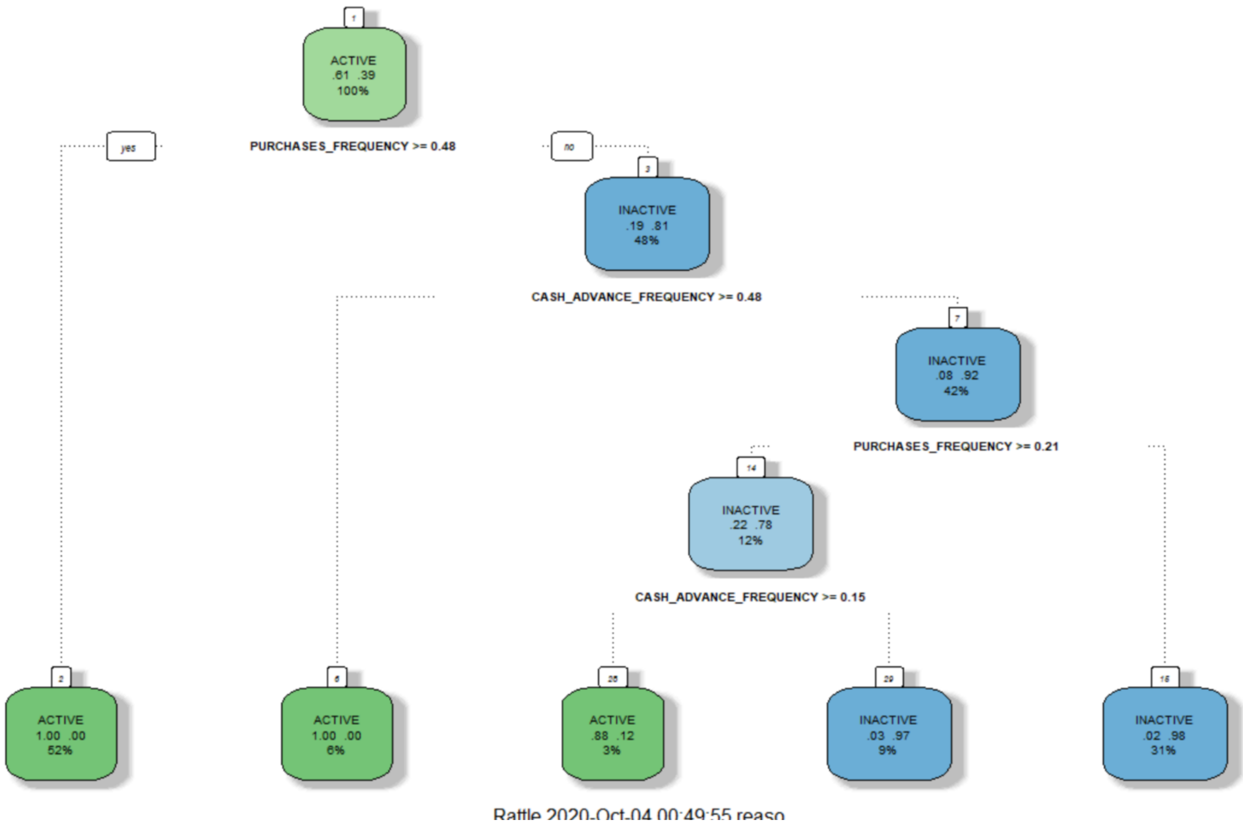
We now have nearly everything that we need to apply the decision tree to this data. We've split our data into training and test datasets.

Now we can use the Decision Tree function to classify the test data and get the tree:

Code snippets:

```
#Using Decision tree Algorithm.
require(rpart)
#Building decision tree
my_tree_two <- rpart(formula = RESULTS ~.,data = credit_train, method = "class")

#graphical interface for data mining
fancyRpartPlot(my_tree_two)
```

Result:



Observation:

1. Start at the root node (depth 0 over 2, the top of the graph):

At the top, it is the overall probability of active. It shows the proportion of results that customer belong to. 61 percent of customer are active.

This node asks whether purchases frequency is greater than 0.48. If yes, then go down to the root's left child node (depth 2). 52 percent are purchase frequency greater that 0.48 with a active 100 percent.

keep on going like that to understand what features impact the likelihood of survival. The results is only related to cash_advance_frequency and purchase_frequency,

Note that, one of the many qualities of Decision Trees is that they require very little data preparation. In particular, they don't require feature scaling or centering.By default, rpart() function uses the Gini impurity measure to split the note. The higher the Gini coefficient, the more different instances within the node.

3.3 Evaluation Model

After using a classification, evaluation is one of the important parts to do, so we predict the diagnosis from test model and check the accuracy comparing with confusion matrix.

Code snippets:

```
#Making Predictions with decision trees
#Make predictions on the test set
my_prediction <- predict(my_tree_two, credit_test, type = "class")

#Get the accurancy of Desicion Tree
confusionMatrix(data = my_prediction, reference = credit_test$RESULTS)
```

Result:

```
Confusion Matrix and Statistics

          Reference
Prediction ACTIVE INACTIVE
  ACTIVE      995        6
  INACTIVE     16      710

               Accuracy : 0.9873
                 95% CI : (0.9808, 0.992
    No Information Rate : 0.5854
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9738

 Mcnemar's Test P-Value : 0.05501

            Sensitivity : 0.9842
            Specificity : 0.9916
         Pos Pred Value : 0.9940
         Neg Pred Value : 0.9780
             Prevalence : 0.5854
         Detection Rate : 0.5761
   Detection Prevalence : 0.5796
      Balanced Accuracy : 0.9879

       'Positive' Class : ACTIVE
```

Observation:

1. The cell percentages in the table indicate the proportion of values that fall into four categories. The top-left cell indicates the true negative results. These 995 of 1727 values are cases where the mass was active and the decision tree algorithm correctly identified it as such. The bottom-right cell indicates the true positive results, where the classifier and the clinically determined label agree that the mass is inactive. A total of 710 of 1727 predictions were true positives.

3. The accuracy of this classification is 98%.

4. RANDOM FOREST

Even today's most sophisticated modeling techniques face this tension between underfitting and overfitting. But, many models have clever ideas that can lead to better performance. One example is the cleverly named Random Forest.

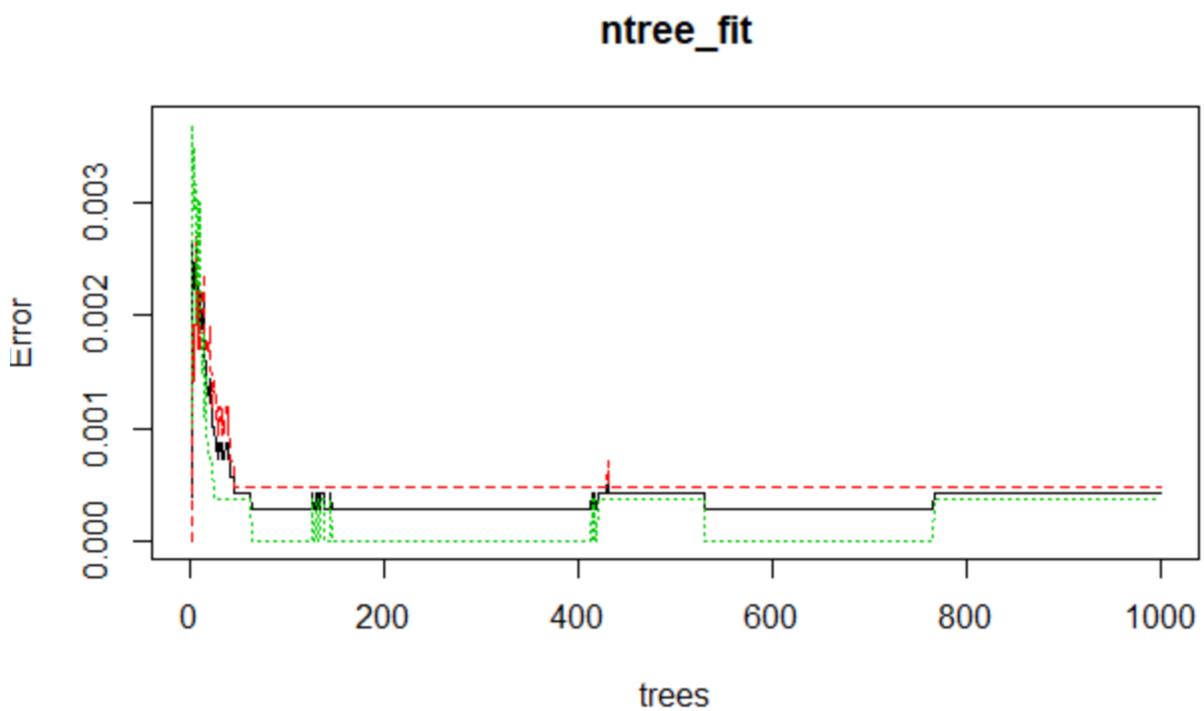
The random forest uses many trees, and it makes a prediction by average the predictions of each component tree. It generally has much better predictive accuracy than a single decision tree and it works well with default parameters. If you keep modeling, you can learn more models with even better performance, but many of those are sensitive to getting the right parameters. To remove the overfitting problem of the dataset we can use random forest method which gives far better results than the decision trees.

Random forest has two important parameters in the function of randomForest() are Ntree and Mtry, in which Ntree is the number of base classifiers contained, and the default is 500, Mtry is the number of variables contained in each decision tree, and the default is logN. When the amount of data is small, the optimal parameter value can use for loop to select.

Code snippets:

```
err<-as.numeric()
for(i in 1:(length(names(credit_train)))-1){
  mtry_test <- randomForest(RESULTS~., data=credit_train, mtry=i)
  err<- append( err, mean( mtry_test$err.rate ) )
}
print(err)
mtry<-which.min(err)
ntree_fit<-randomForest(RESULTS~., data=credit_train, mtry=mtry, ntree=1000)
plot(ntree_fit)
```

Results:



Observation:

1. After checking plot, we know when ntree = 300 or 700, the error is stable. So we could get the final parameter for model and check varImpPlot()

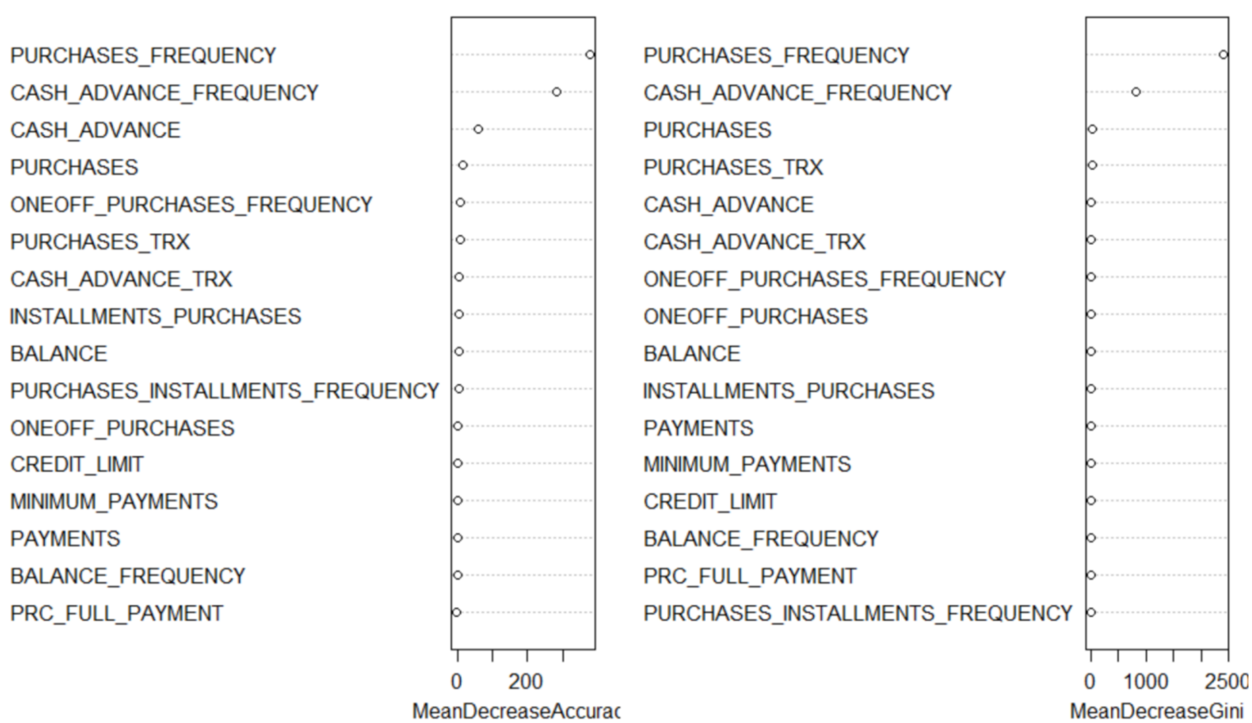
Code snippets:

```
#Final Random Forest model
rf<-randomForest(RESULTS~., data=credit_train, mtry=mtry, ntree=300, importance=T )
rf

#Get importance
importance(rf)
varImpPlot(rf)
```

Results:

```
Call:
 randomForest(formula = RESULTS ~ ., data = credit_train, mtry = mtry,      ntree = 300, importance = T)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 14

        OOB estimate of  error rate: 0.06%
Confusion matrix:
         ACTIVE INACTIVE  class.error
ACTIVE     4201        3 0.0007136061
INACTIVE      1     2704 0.0003696858
> #Get importance
> importance(rf)
                                      ACTIVE    INACTIVE MeanDecreaseAccuracy MeanDecreaseGini
BALANCE                            5.0603263   2.8013370             5.940312        1.6945487
BALANCE_FREQUENCY                  1.9822595   1.5404434             2.412048        0.5136000
PURCHASES                          2.7454889  16.9415568            13.621335       23.8793473
ONEOFF_PURCHASES                   2.4059316   5.6900779             5.804916        1.6056605
INSTALLMENTS_PURCHASES             0.3177926   4.9545067             4.979042        1.0976287
CASH_ADVANCE                      12.3801879  36.5364129            34.894424       22.7774899
ONEOFF_PURCHASES_FREQUENCY         4.4137020   9.6160811             9.740775        5.5637744
PURCHASES_INSTALLMENTS_FREQUENCY   1.1472463   3.8905109             3.974730        8.3229507
PURCHASES_FREQUENCY              205.3509621 338.7613059           335.115251     2118.5048774
CASH_ADVANCE_TRX                   6.0758639  12.0594323            11.210726       68.6312285
PURCHASES_TRX                      4.2782952  12.1807574            12.766608      292.9330945
CREDIT_LIMIT                       1.4710940   1.5986185             2.326357        0.5759983
PAYMENTS                           2.9570537   1.7127016             3.552943        1.4129774
MINIMUM_PAYMENTS                   3.3239203   0.4359870             3.410017        1.0554254
PRC_FULL_PAYMENT                   0.9247613   0.8650358             1.220343        0.2560963
CASH_ADVANCE_FREQUENCY           220.2677674 116.5396173           189.518234      741.4945995
> varImpPlot(rf)
```

rf



Observation:

1. According to above graph, we could see purcheses_frequency and cash_advence_frequency are the most important feature, which means impact the most on dependent variable.

4.1 Evaluation Model

Use test set to predict and calculated the F-value is calculated to evaluate model.

Code snippets:

```
pred1<-predict(rf,newdata=credit_test)
Freq1<-table(pred1,credit_test$RESULTS)
tp<-as.data.frame(Freq1)[4,3]
tn<-as.data.frame(Freq1)[1,3]
fn<-as.data.frame(Freq1)[2,3]
fp<-as.data.frame(Freq1)[3,3]
p<-tp/(tp+fp)
r<-tp/(tp+fn)
f<-2/(1/p+1/r)
f
```

Results:

```
> f
[1] 0.9993022
```

Observation:

1. Random forest model giving us 99% accuracy, 1% better than decision tree but notice there is no much change specificity.

5.CONCLUSION

In our project, We used Decision Tree and Random Forest above to predict whether the client is active, While 98 % accuracy in Decision Tree and 99% accuracy Random Forest seems impressive for a few lines of R code. Also, we could use this data to cluster. Segmentation of customers can be used to define marketing strategies so that a bank could precision marketing different customer and increase profit.

6.REFERENCE

1. prediction of heart diseases

https://www.kaggle.com/naik170106027/prediction-of-heart-diseases

2. Heart Disease UCI EDA, PCA, KMEANS, HC, RF with R

https://www.kaggle.com/ekrembayar/heart-disease-uci-eda-models-with-r