ALY 6040 Module 3 Technique Practice

Student's name: Mohammad Hossein Movahedi

Assignment title: Module 3 Technique Practice

Course number and title: ALY6040 71368 Data Mining Applications SEC 09 Fall 2022 CPS [TOR-A-HY]

Term: 202315_A Fall 2022 CPS Quarter First Half

Instructor's name: Hootan Kamran, Ph.D.

Oct 2, 2022,

Introduction

In this assignment, I will use the random forest technique to solve our main business question in the group project which is predicting that weather a stock will be profitable or not/

There are five main steps in this project.

1. Loading the data
2. Dividing data into testing and training groups
3. Creating the random forest
4. Finding the best ‘mtry’
5. Examining the forest and validation

Code walkthrough

Loading the data

The first part of the Code is Just Installing and loading the Libraries.

```
print('Individual assingnment 3')
#Installing and loading nessesarry libraries
install.packages('pacman')
install.packages('tidyverse')
install.packages('tidymodels')
install.packages('leaps')
install.packages('glmnet')
install.packages('BMA')
install.packages('janitor')
install.packages("randomForest")
install.packages('effects')

library(tidyverse)
library(tidymodels)
library(leaps)
library(pacman)
library(glmnet)
library(BMA)
library(janitor)
library(caTools)
library(randomForest)
```

In the next part, I load the dataset and delete the AnalystRating variable because it's basically the same thing as class variable but with more information also I change class from number to factor.

```
#loading the cleaned dataset

dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating))
dim(dt)
```

Now, we can move to the next part

Dividing data into testing and training groups

```
# spiting data into test and train

# Splitting data in train and test data
set.seed(123)  # Setting seed
split <- sample.split(dt, SplitRatio = 0.7)
split

train <- subset(dt, split == "TRUE")
test <- subset(dt, split == "FALSE")
```

With a split ratio of 0.7, we divide the dataset into training and testing

Creating the random forest

Now we create a random forest with 2000 trees

```
# Fitting Random Forest to the train dataset
set.seed(123)  # Setting seed
classifier_RF = randomForest(class~.,
                             data = train,
                             ntree = 2000)

classifier_RF
```

The result is shown below

```
> classifier_RF

Call:
 randomForest(formula = class ~ ., data = train, ntree = 2000)
               Type of random forest: classification
                     Number of trees: 2000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 14.92%
Confusion matrix:
  0   1 class.error
0 3  81 0.964285714
1 3 476 0.006263048
```

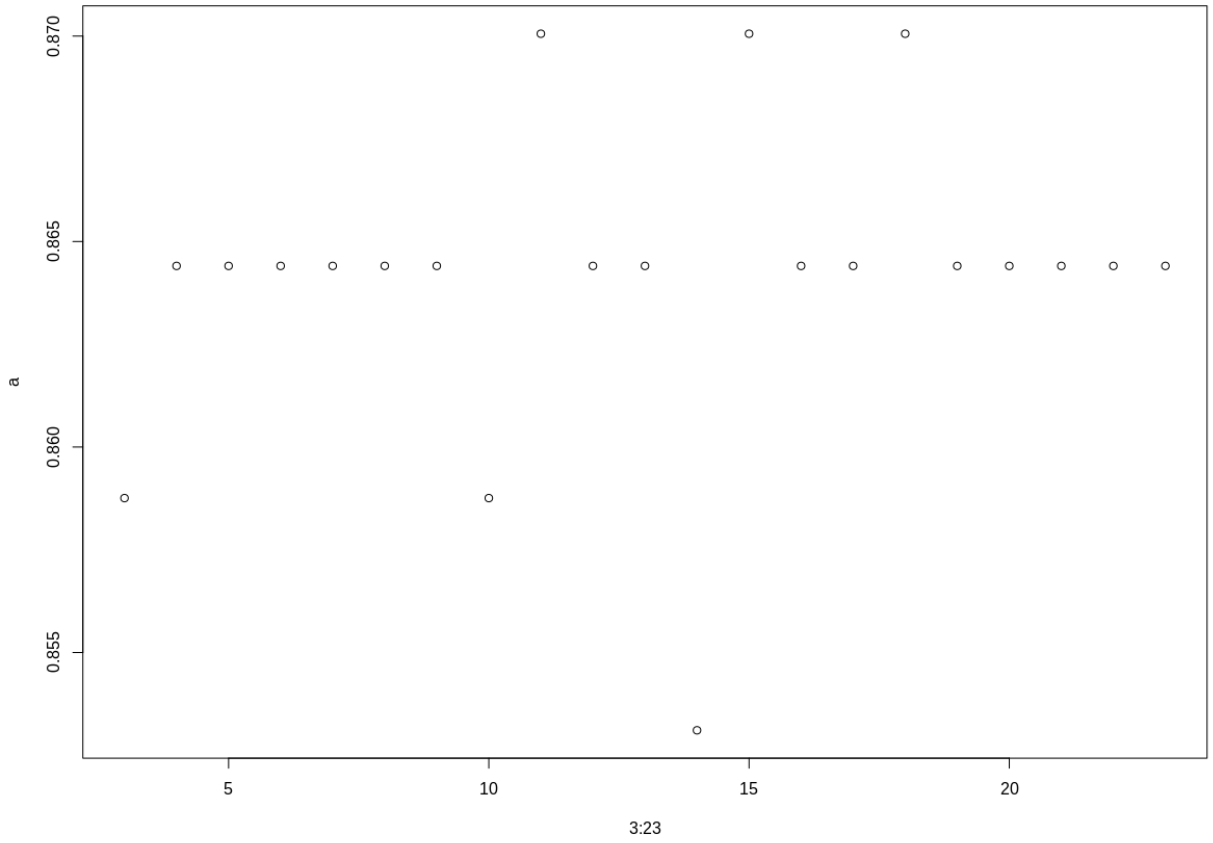
As can be seen, the estimated error for this forest is around 15% which very good

Finding the best ‘mtry’

In this step we use a loop to find the best mtry

```
a=c()
i=5
set.seed(123)  # Setting seed
for (i in 3:23) {
  model3 <- randomForest(class ~ ., data = train, ntree = 2000, mtry = i,
importance = TRUE)
  predValid <- predict(model3, newdata = test[-23])
  a[i-2] = mean(predValid == test$class)
}
a
plot(3:23,a)
```

The result is shown below



as It can be seen the best result is generated by 11 variables. so I adjusted the code

```
# adjusted forest
set.seed(123)  # Setting seed
classifier_RF = randomForest(class~.,
                             data = train,
                             mtry = 11,
                             ntree = 2000)

classifier_RF
```

Due to the random nature of the random forest, the results didn't change that much

Validation

At last, we test the prediction on the testing dataset

```
# Predicting the Test set results
y_pred = predict(classifier_RF, newdata = test[-23])

# Confusion Matrix
mtx = table(test[,23], y_pred)
```

we used the tree to predict the class based on attributes. And then we calculate the accuracy of the model

```
#Calculating accuracy
t <- table(mushrooms_test$class,pred)
confusionMatrix(t)
```

The resulting confusion matrix is shown below

```
> confusionMatrix(mtx)
Confusion Matrix and Statistics

   y_pred
      0   1
  0   1  25
  1   0 151

               Accuracy : 0.8588
                 95% CI : (0.7986, 0.9065)
    No Information Rate : 0.9944
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0639

 Mcnemar's Test P-Value : 1.587e-06

            Sensitivity : 1.00000
            Specificity : 0.85795
         Pos Pred Value : 0.03846
         Neg Pred Value : 1.00000
             Prevalence : 0.00565
         Detection Rate : 0.00565
   Detection Prevalence : 0.14689
      Balanced Accuracy : 0.92898
```

```
        'Positive' Class : 0
```

As can be seen, the detection rate of our model is terrible, indicating that the random forest failed in this prediction

Conclusion

The final result is not desirable results; however it shows that random forest can sometimes create inaccurate results

References

R-bloggers. (2018). *How to implement Random Forests in R | R-bloggers*. [online] Available at: https://www.r-bloggers.com/2018/01/how-to-implement-random-forests-in-r/ [Accessed 13 Oct. 2022].

Maklin, C. (2019). *Random Forest In R - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/random-forest-in-r-f66adf80ec9 [Accessed 13 Oct. 2022].

Appendix

```
print('Individual assingnment 3')
#Installing and loading necessary libraries
install.packages('pacman')
install.packages('tidyverse')
install.packages('tidymodels')
install.packages('leaps')
install.packages('glmnet')
install.packages('BMA')
install.packages('janitor')
install.packages("randomForest")
install.packages('effects')

library(tidyverse)
library(tidymodels)
library(leaps)
library(pacman)
library(glmnet)
library(BMA)
library(janitor)
library(caTools)
library(randomForest)

#loading the cleaned dataset

dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating))
dim(dt)

# running random forest for the class variable
# spiting data into test and train

# Splitting data in train and test data
set.seed(321)  # Setting seed
split <- sample.split(dt, SplitRatio = 0.8)
split

train <- subset(dt, split == "TRUE")
test <- subset(dt, split == "FALSE")
```

```
# Fitting Random Forest to the train dataset
set.seed(123)  # Setting seed
classifier_RF = randomForest(class~.,
                             data = train,
                             mtry = 11,
                             ntree = 2000)

classifier_RF

# finding mtry
a=c()
i=5
set.seed(123)  # Setting seed
for (i in 3:23) {
  model3 <- randomForest(class ~ ., data = train, ntree = 100, mtry = i,
importance = TRUE)
  predValid <- predict(model3, newdata = test[-23])
  a[i-2] = mean(predValid == test$class)
}
a
plot(3:23,a)

# adjusted forest
set.seed(123)  # Setting seed
classifier_RF = randomForest(class~.,
                             data = train,
                             mtry = 11,
                             ntree = 2000)

classifier_RF


# Predicting the Test set results
y_pred = predict(classifier_RF, newdata = test[-23])

# Confusion Matrix
mtx = table(test[,23], y_pred)
#Calculating accuracy

confusionMatrix(mtx)

# Plotting model
plot(classifier_RF)
```

```
varImpPlot(classifier_RF)
# Importance plot
k <- importance(classifier_RF)
k

a=c()
i=5
set.seed(123)  # Setting seed
for (i in 3:23) {
  model3 <- randomForest(class ~ ., data = train, ntree = 2000, mtry = i,
importance = TRUE)
  predValid <- predict(model3, newdata = test[-23])
  a[i-2] = mean(predValid == test$class)
}
a
plot(3:23,a)
```