Predicting Exercise Manner

Ahmed Yahya Khaled

10/7/2020

Project : Practical Machine Learning

Coursera

Introduction

People do exercises and One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise.

The outcome variable is `classe` , a factor variable with 5 levels. For our data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)

Required Packages

These packages need to install if not done already

```
# install.packages("tidyverse") # ggplot2 is encapsulated in tidyverse
# install.packages("caret")
# install.packages("randomForest")
# install.packages("rpart")
# install.packages("rpart.plot")
```

Then we need to load the packages

```
library(tidyverse)
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
```

Import Dataset

links :
training set : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

test set : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

set working directory :

```
setwd("E:/My_learning/R/Practical Machine Learning _ Coursera/Project")
```

import from working directory :

```
training <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))

# View(training) ; View(testing)
```

set seed for reproducibility

```
set.seed(1213)
```

Data Processing

data cleaning

```
# removing columns with NA
training <- training[ , colSums(is.na(training)) == 0]
testing <- testing[ , colSums(is.na(testing)) == 0]

# # removing irrelevant columns
training <- training[ , -c(1:7)]
testing <- testing[ , -c(1:7)]

dim(training) ; dim(testing)
```

```
## [1] 19622    53
```

```
## [1] 20 53
```

Cross Validation

For cross validation purpose splitting the `training` data into `subTraining` (75%) & `subTesting` (25%) data

```
inTrain <- createDataPartition(y = training$classe, p = 0.75, list = F)
subTraining <- training[inTrain, ] ; subTesting <- training[-inTrain, ]
dim(subTraining) ; dim(subTesting)
```

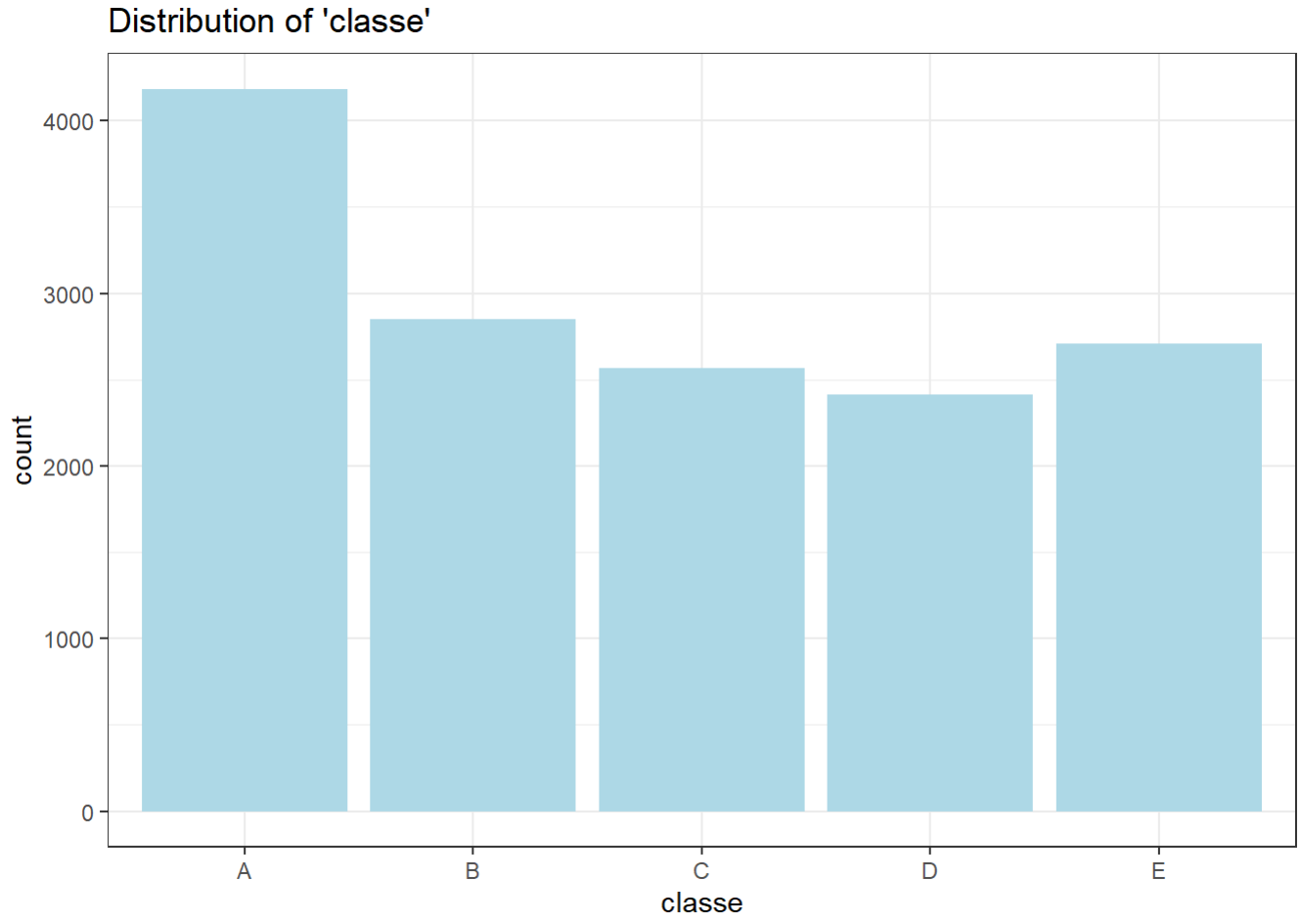
```
## [1] 14718    53
```

```
## [1] 4904   53
```

Data Exploration

As independent variable are many, checking only the dependent variable - `classe`

```
ggplot(subTraining)+
  geom_bar(aes(x = classe), fill = 'lightblue') +
  ggtitle("Distribution of 'classe'") +
  theme_bw()
```




```
table(subTraining$classe)
```

```
##
##    A    B    C    D    E
## 4185 2848 2567 2412 2706
```

Prediction Models

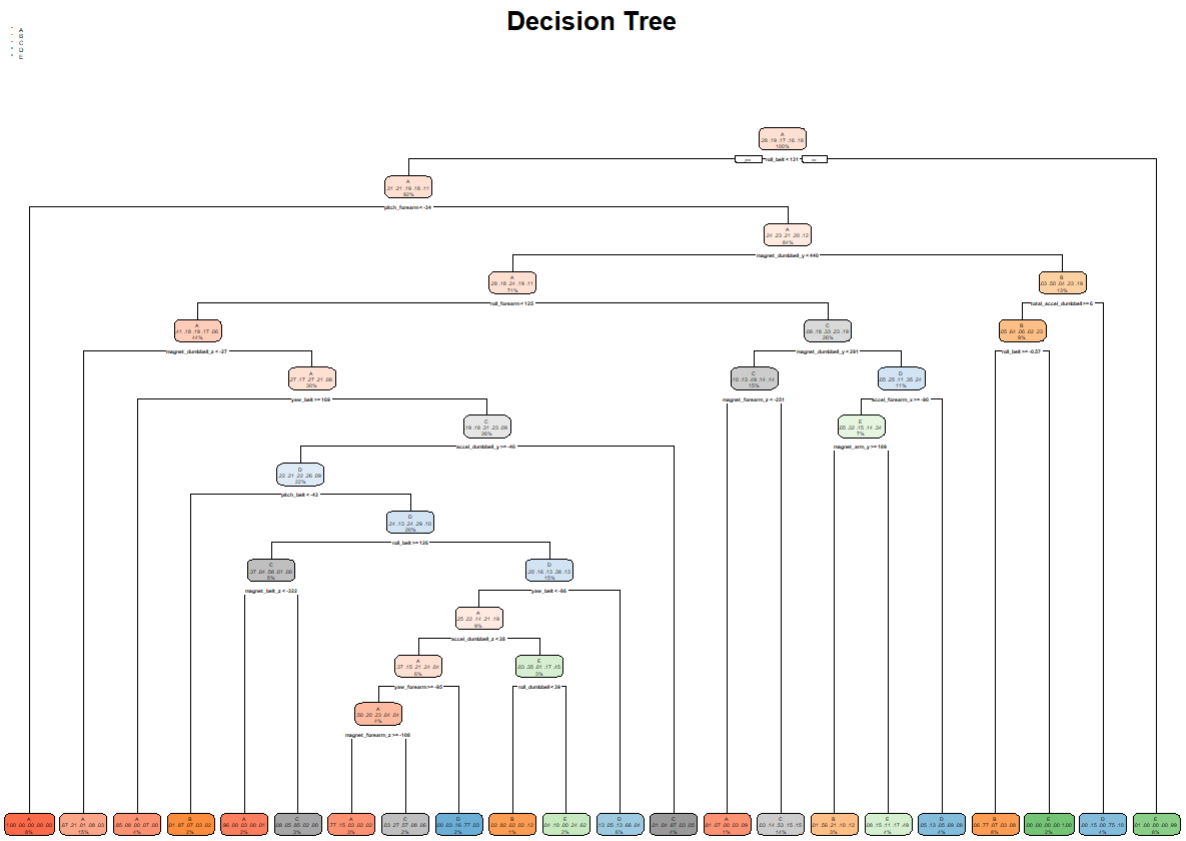
Decision Tree

fitting model

```
FitDt <- rpart(classe ~ . ,
               data = subTraining, method = "class")
```

ploting trees

```
rpart.plot(FitDt, main="Decision Tree")
```



predicting value

```
ypredDt <- predict(FitDt, newdata = subTesting, type = "class")
table(ypredDt, subTesting$classe)
```

```
##
## ypredDt    A    B    C    D    E
##       A 1262  217   15   78   41
##       B   26  496   78   22   66
##       C   37  125  693  136  106
##       D   48   71   48  505   58
##       E   22   40   21   63  630
```

confusion matrix

```
confusionMatrix(ypredDt, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1262  217   15   78   41
##          B   26  496   78   22   66
##          C   37  125  693  136  106
##          D   48   71   48  505   58
##          E   22   40   21   63  630
##
## Overall Statistics
##
##                Accuracy : 0.7312
##                  95% CI : (0.7186, 0.7436)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6584
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9047   0.5227   0.8105   0.6281   0.6992
## Specificity            0.9000   0.9515   0.9002   0.9451   0.9635
## Pos Pred Value         0.7824   0.7209   0.6317   0.6918   0.8119
## Neg Pred Value         0.9596   0.8926   0.9574   0.9284   0.9344
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2573   0.1011   0.1413   0.1030   0.1285
## Detection Prevalence   0.3289   0.1403   0.2237   0.1489   0.1582
## Balanced Accuracy      0.9023   0.7371   0.8554   0.7866   0.8314
```

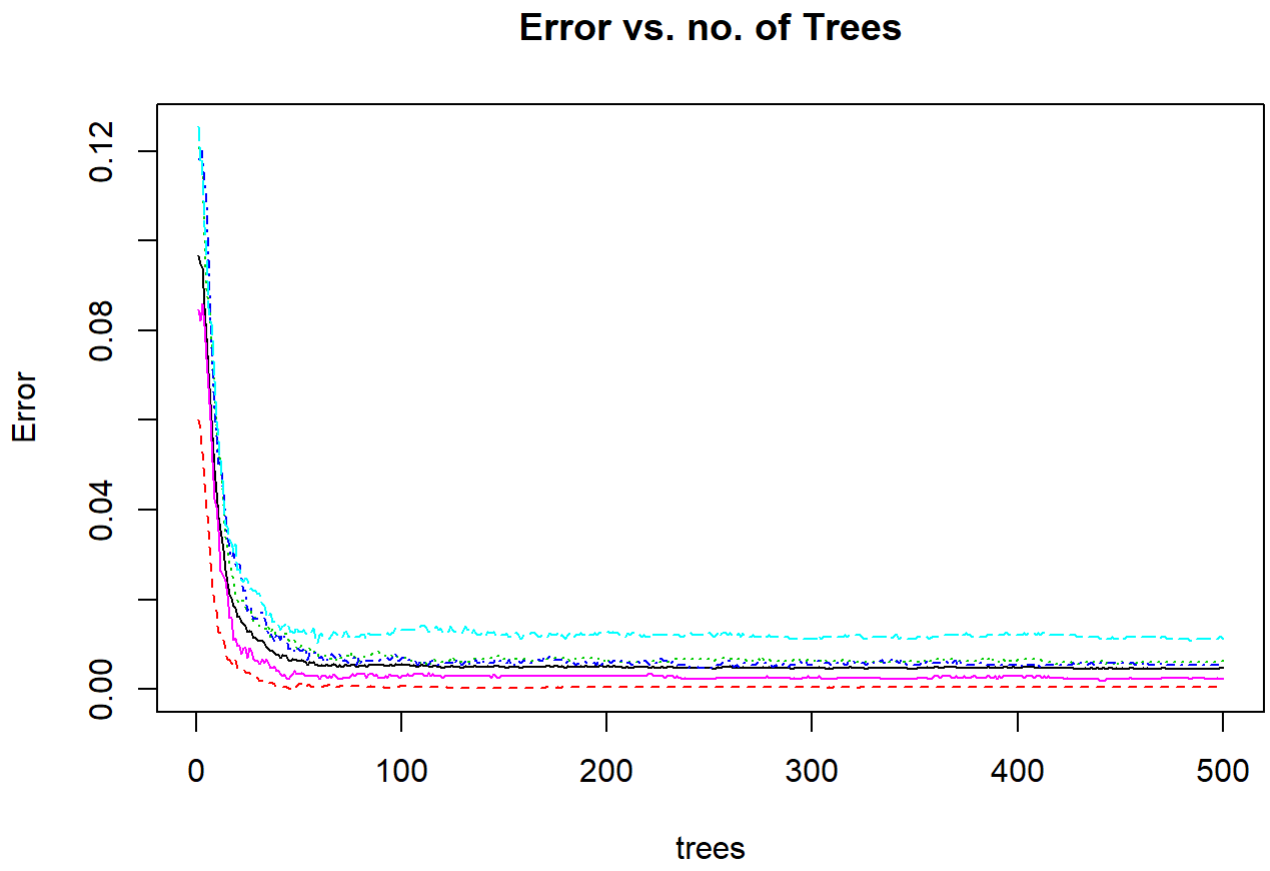
Random Forest

fitting model

```
FitRf <- randomForest(classe ~ . ,
                      data = subTraining, methods = "class" )
```

plotting error vs number of trees

```
plot(FitRf, main = "Error vs. no. of Trees")
```



predicting value

```
ypredRf <- predict(FitRf, newdata = subTesting, type = "class")
table(ypredRf, subTesting$classe)
```

```
##
## ypredRf    A    B    C    D    E
##       A 1395    3    0    0    0
##       B    0  943    3    0    0
##       C    0    3  851   11    0
##       D    0    0    1  792    1
##       E    0    0    0    1  900
```

confusion matrix

```
confusionMatrix(ypredRf, subTesting$classe)
```

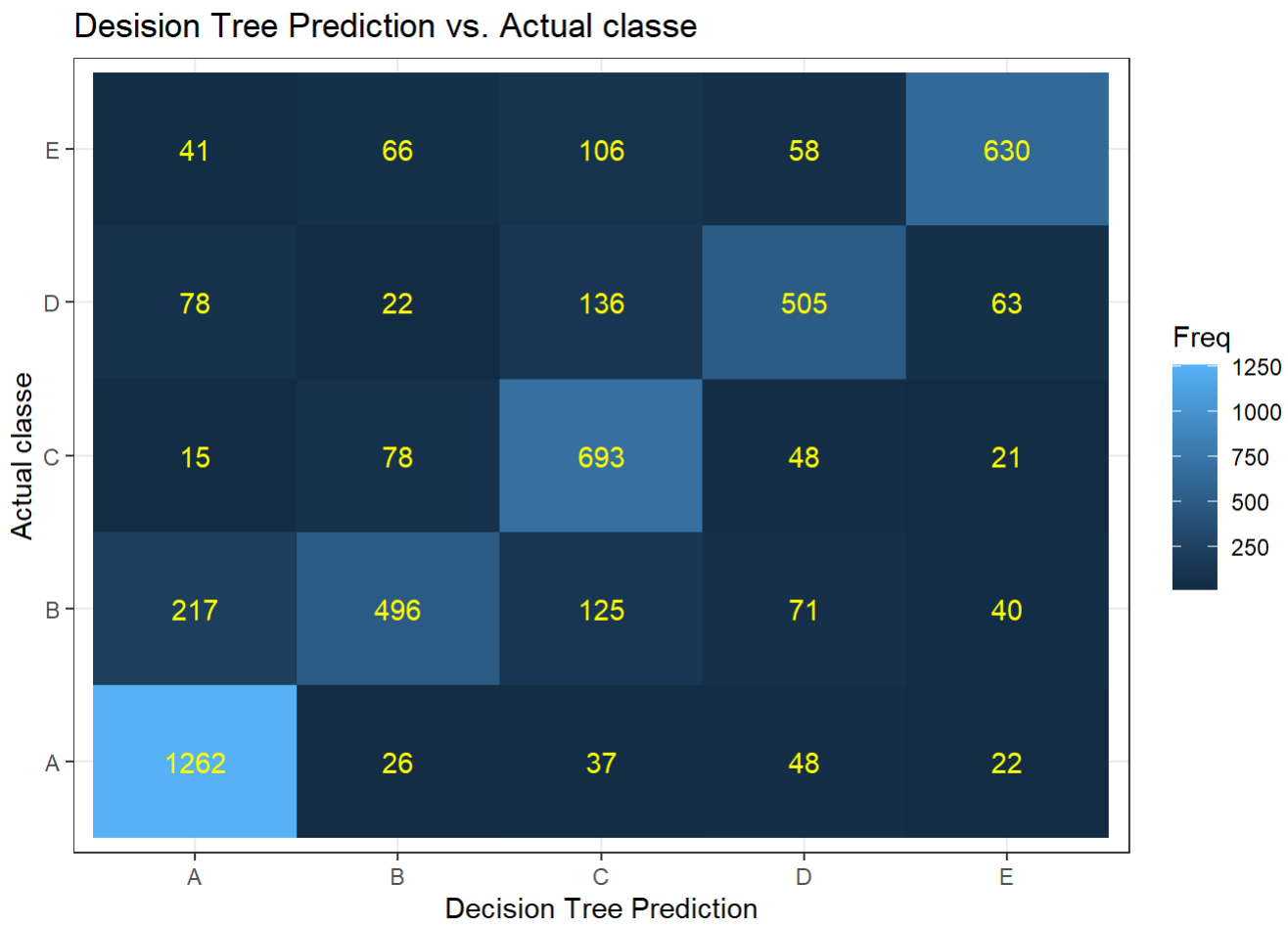
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  943    3    0    0
##          C    0    3  851   11    0
##          D    0    0    1  792    1
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9953
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9941
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9937   0.9953   0.9851   0.9989
## Specificity            0.9991   0.9992   0.9965   0.9995   0.9998
## Pos Pred Value         0.9979   0.9968   0.9838   0.9975   0.9989
## Neg Pred Value         1.0000   0.9985   0.9990   0.9971   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1923   0.1735   0.1615   0.1835
## Detection Prevalence   0.2851   0.1929   0.1764   0.1619   0.1837
## Balanced Accuracy      0.9996   0.9965   0.9959   0.9923   0.9993
```

Model Comparison

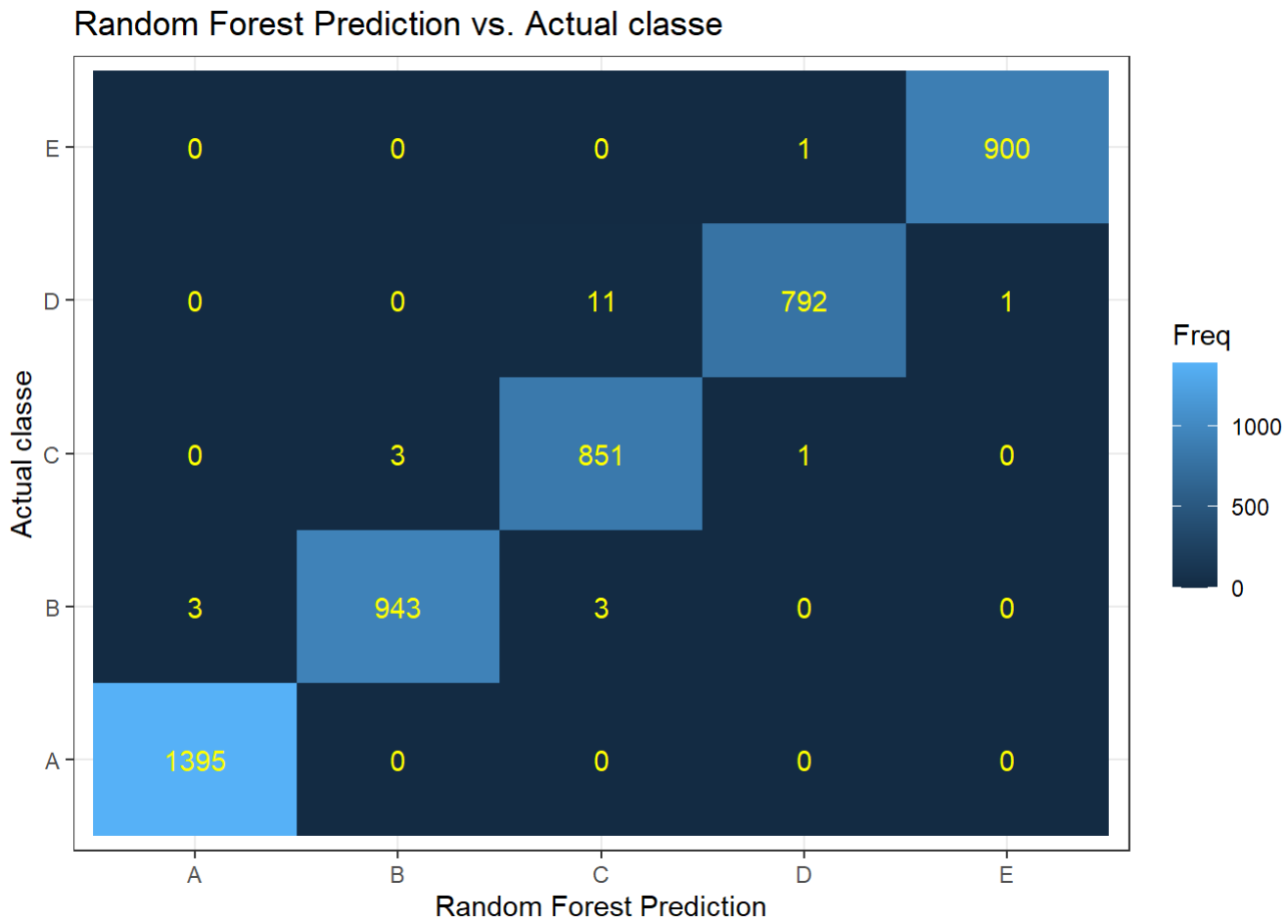
Let’s present the outcome of the two models in Tiles plot

```
cmDt <- as.data.frame(table(ypredDt, subTesting$classe))
cmRf <- as.data.frame(table(ypredRf, subTesting$classe))

ggplot(cmDt) +
  geom_tile(aes(x = ypredDt, y = Var2, fill = Freq)) +
  geom_text(aes(x = ypredDt, y = Var2, label = Freq), color = "yellow") +
  ggtitle("Desision Tree Prediction vs. Actual classe") +
  labs(x = "Decision Tree Prediction", y = "Actual classe") +
  theme_bw()
```




```
#/
ggplot(cmRf) +
  geom_tile(aes(x = ypredRf, y = Var2, fill = Freq)) +
  geom_text(aes(x = ypredRf, y = Var2, label = Freq), color = "yellow") +
  ggtitle("Random Forest Prediction vs. Actual classe") +
  labs(x = "Random Forest Prediction", y = "Actual classe") +
  theme_bw()
```



Apparently, Random Forest is provding better prediction here.

Feature	Decision Tree	Random Forest
Accurqacy	0.739	0.995
95% CI	(0.719, 0.743)	(0.993, 0.997)

Prediction on `test set`

Applying the Random Forest model on the testing data

```
testing$classe <- predict(FitRf, newdata = testing, type = "class")
table(testing$classe)
```

```
##
## A B C D E
## 7 8 1 1 3
```