Machine Learning Project

H.F.

4/21/2021

Background Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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 is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

```
library(ggplot2);library(lattice); library(caret); library(randomForest); library(rpart); library(rpart.plot);library(e1071)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.5
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

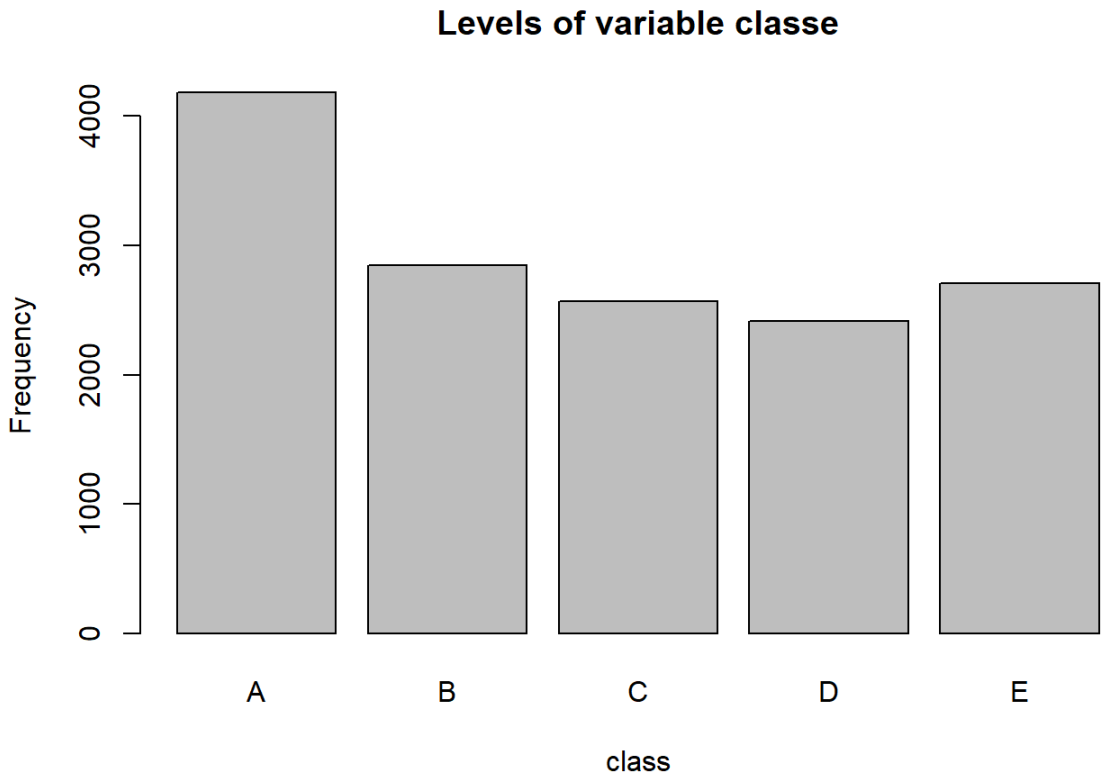
```
set.seed(555)
train_link<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_link<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
train <- read.csv(url(train_link), na.strings=c("NA","#DIV/0!", ""))
test <- read.csv(url(test_link), na.strings=c("NA","#DIV/0!", ""))

#CLEANING
train<-train[,colSums(is.na(train)) == 0]
test <-test[,colSums(is.na(test)) == 0]
train<-train[,-c(1:7)]
test <-test[,-c(1:7)]

train$classe=as.factor(train$classe) #otherwise cant plot and run the classification

#PARTITIONING
ss <- createDataPartition(y=train$classe, p=0.75, list=FALSE)
sub_train <- train[ss, ]
sub_test <- train[-ss, ]

plot(sub_train$classe,main="Levels of variable classe", xlab="class", ylab="Frequency")
```



Random Forest Model

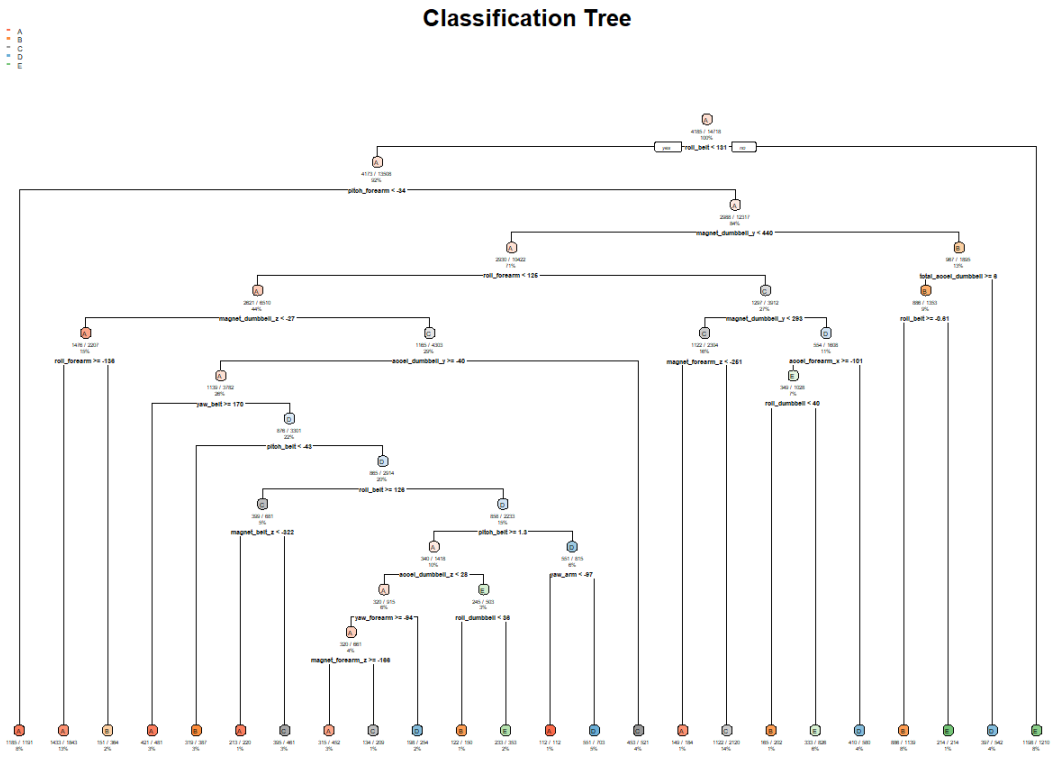
```
my_RF<- randomForest(classe ~. , data=sub_train, method="class")
pred_RF <- predict(my_RF, sub_test, type = "class")

# Test
confusionMatrix(pred_RF, sub_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    6    0    0    0
##          B    1  936   11    0    0
##          C    0    7  844    8    1
##          D    0    0    0  796    2
##          E    1    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9896, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9863   0.9871   0.9900   0.9967
## Specificity            0.9983   0.9970   0.9960   0.9995   0.9998
## Pos Pred Value         0.9957   0.9873   0.9814   0.9975   0.9989
## Neg Pred Value         0.9994   0.9967   0.9973   0.9981   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1909   0.1721   0.1623   0.1831
## Detection Prevalence   0.2853   0.1933   0.1754   0.1627   0.1833
## Balanced Accuracy      0.9984   0.9916   0.9916   0.9948   0.9982
```

Decision Tree Model

```
my_DT <- rpart(classe ~ ., data=sub_train, method="class")
rpart.plot(my_DT, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```




```
pred_DT <- predict(my_DT, sub_test, type = "class")
confusionMatrix(pred_DT, sub_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1283  149   18   48   12
##          B   38  532   50   74   75
##          C   32  135  680  119  118
##          D   17   68   55  495   41
##          E   25   65   52   68  655
##
## Overall Statistics
##
##                Accuracy : 0.7433
##                  95% CI : (0.7308, 0.7555)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6744
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9197   0.5606   0.7953   0.6157   0.7270
## Specificity            0.9353   0.9401   0.9002   0.9559   0.9475
## Pos Pred Value         0.8497   0.6918   0.6273   0.7322   0.7572
## Neg Pred Value         0.9670   0.8992   0.9542   0.9269   0.9391
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2616   0.1085   0.1387   0.1009   0.1336
## Detection Prevalence   0.3079   0.1568   0.2210   0.1378   0.1764
## Balanced Accuracy      0.9275   0.7503   0.8478   0.7858   0.8373
```

It seems Random Forrest produces better results so we use it for prediction on the test data:

```
predict(my_RF, test, type="class")
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```