# Machine Learning Peer Graded Assignment

Pranav

19/10/2020

##Overview

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 behaviour, or because they are tech geeks.
The goal of this project is to predict how they did the exercise. There is the "classe" variable in the training set. Create a report describing t model, how cross-validation is used, what are the expected out of sample errors, and the choices. Use a prediction model to predict 20 different test cases.

##Understanding the Data

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

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

##Loading required packages and boot Variables

```r
#Data set variable declaration
train.data   <- 'E:/pml-training.csv'
test.data <- 'E:/pml-testing.csv'
train.data.url    <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.data.url  <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

#Directories
if (!file.exists("data")){
  dir.create("data")
}
if (!file.exists("data/submission")){
  dir.create("data/submission")
}
#To check if R packages are present
IscaretInstalled <- require("caret")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.3

## Loading required package: ggplot2
```

```r
if(!IscaretInstalled){
    install.packages("caret")
    library("caret")
    }
IsrandomForestInstalled <- require("randomForest")
```

```
## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.6.3

## 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
```

```r
if(!IsrandomForestInstalled){
    install.packages("randomForest")
    library("randomForest")
    }
IsRpartInstalled <- require("rpart")
```

```
## Loading required package: rpart
```

```r
if(!IsRpartInstalled){
    install.packages("rpart")
    library("rpart")
    }
IsRpartPlotInstalled <- require("rpart.plot")
```

```
## Loading required package: rpart.plot

## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```r
if(!IsRpartPlotInstalled){
    install.packages("rpart.plot")
    library("rpart.plot")
    }
# Setting seed for reproducibility
set.seed(1010)
```

## Data Processing

Downloading and processing of data is complete. Some transformation and cleaning are to be performed, so that `NA` values are to be dropped. Irrelevant columns (columns 1 to 7) will be removed from the subset.

The `pml-training.csv` data is used to devise training and testing sets. The `pml-test.csv` data is used to predict and answer the 20 questions based on the trained model.

```r
# Downloading required data
download.file(train.data.url, train.data)
download.file(test.data.url,test.data )

# Cleaning the data
train_data   <-read.csv(train.data, na.strings=c("NA","#DIV/0!", ""))
test_data <-read.csv(test.data , na.strings=c("NA", "#DIV/0!", ""))
train_data<-train_data[,colSums(is.na(train_data)) == 0]
test_data <-test_data[,colSums(is.na(test_data)) == 0]

# Subsetting the data
train_data   <-train_data[,-c(1:7)]
test_data <-test_data[,-c(1:7)]
```

## Cross-validation

Cross-validation is performed by dividing the training data set into two ratios - training(75%) and testing(25%).

```r
subvarSams <- createDataPartition(y=train_data$classe, p=0.75, list=FALSE)
subvarTrain <- train_data[subvarSams, ]
subvarTest <- train_data[-subvarSams, ]
```

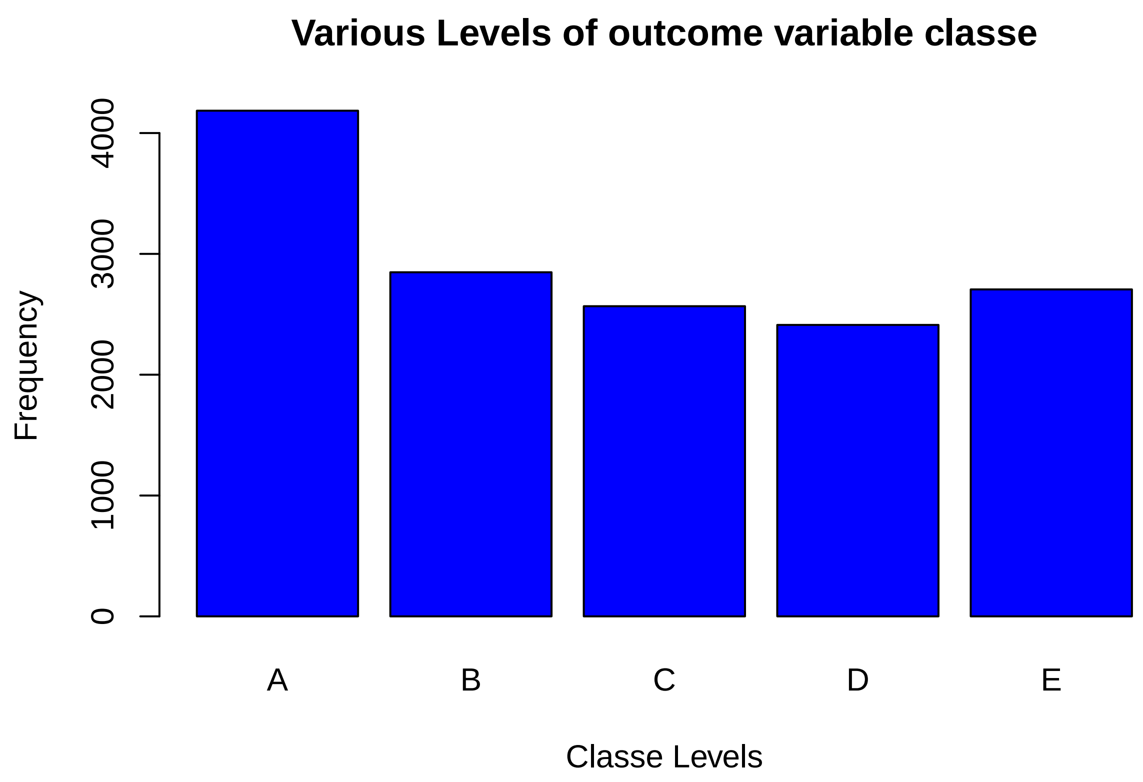
## Expected out-of-sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the sub-Testing data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

## Exploratory analysis

The variable `classe` contains five levels. The plot of the outcome variable shows the frequency of each level in the sub-Training data.

```r
plot(subvarTrain$classe, col="blue", main="Various Levels of outcome variable classe", xlab="Classe Leve
```



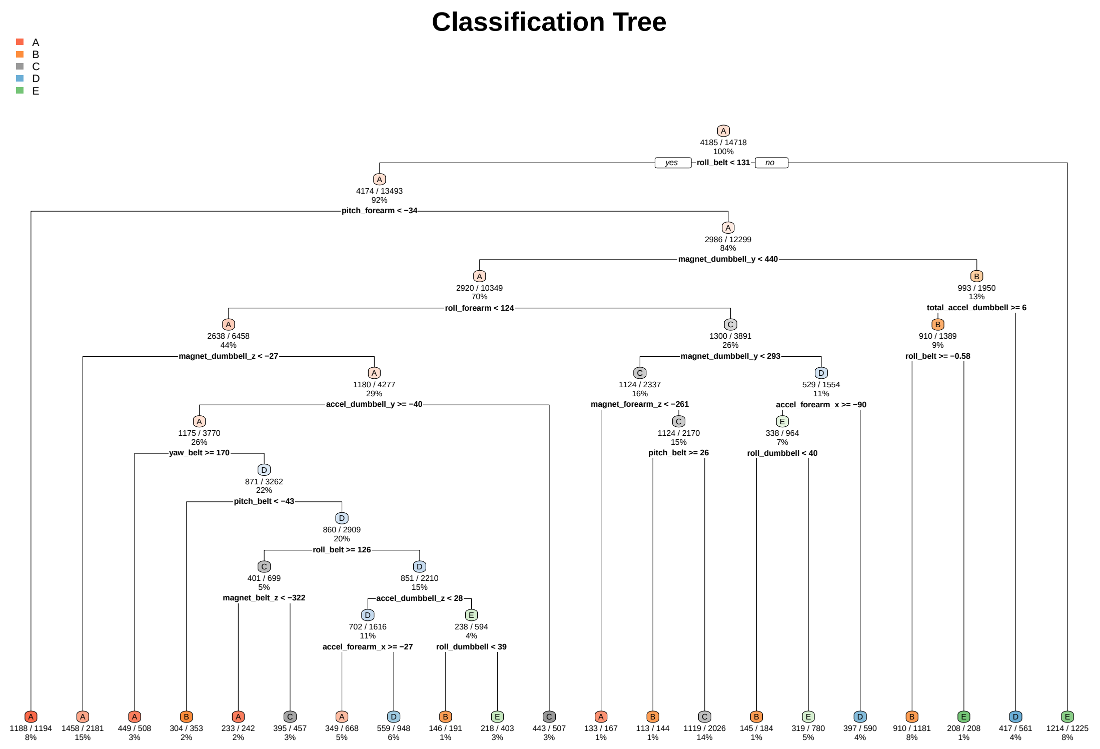
The plot tells us that Level A is the most frequent and D is the least frequent.

## Prediction models

In this section, a decision tree and random forest will be applied to the data.

### Decision tree

```r
# Fit model
modelFyDT <- rpart(classe ~ ., data=subvarTrain, method="class")
# Perform prediction
predictDT <- predict(modelFyDT, subvarTest, type = "class")
# Plot result
rpart.plot(modelFyDT, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Following confusion, the matrix shows the errors of the prediction algorithm.

```r
confusionMatrix(predictDT, subvarTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1246  206   38  107   40
##          B   35  511   40   33   50
##          C   32   76  647  115  103
##          D   56   81   77  463   43
##          E   26   75   53   86  665
##
## Overall Statistics
##
##                Accuracy : 0.7202
##                  95% CI : (0.7074, 0.7328)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6441
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8932   0.5385   0.7567  0.57587   0.7381
## Specificity            0.8886   0.9601   0.9195  0.93732   0.9400
## Pos Pred Value         0.7611   0.7638   0.6650  0.64306   0.7348
## Neg Pred Value         0.9544   0.8966   0.9471  0.91850   0.9410
## Prevalence             0.2845   0.1935   0.1743  0.16395   0.1837
## Detection Rate         0.2541   0.1042   0.1319  0.09441   0.1356
## Detection Prevalence   0.3338   0.1364   0.1984  0.14682   0.1845
## Balanced Accuracy      0.8909   0.7493   0.8381  0.75659   0.8391
```

**Random forest**

```
# Fit model
modelFyRF <- randomForest(classe ~ ., data=subvarTrain, method="class")
# Perform prediction
predictRF <- predict(modelFyRF, subvarTest, type = "class")
```

Following confusion, the matrix shows the errors of the prediction algorithm.

```
confusionMatrix(predictRF, subvarTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  946    7    0    0
##          C    0    2  848    5    0
##          D    0    0    0  799    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9918   0.9938   0.9978
## Specificity            0.9997   0.9982   0.9983   0.9995   1.0000
## Pos Pred Value         0.9993   0.9927   0.9918   0.9975   1.0000
## Neg Pred Value         1.0000   0.9992   0.9983   0.9988   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1729   0.1629   0.1833
## Detection Prevalence   0.2847   0.1943   0.1743   0.1633   0.1833
## Balanced Accuracy      0.9999   0.9975   0.9950   0.9966   0.9989
```

## Conclusion

### Result

The Confusion Matrices show that the Random Forest algorithm performs better than a Decision Tree. The accuracy for the Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to 0.739 (95% CI: (0.727, 0.752)) for Decision Tree model. The Random Forest model is chosen.

### Expected out-of-sample error

The expected out-of-sample error is estimated at 0.5%. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples, will be unclassified.

## Submission

In this section, the files for the project submission are generated using the random forest algorithm on the testing data.

```r
# Perform prediction
predictSub <- predict(modelFyRF, test_data, type="class")
predictSub
```

```
##  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
```

```r
# Writing into files for submission
ml_write.files = function(x){
  y = length(x)
  for(j in 1:y){
    file_name = paste0("./data/submission/problem_id_",j,".txt")
    write.table(x[j],file=file_name,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
ml_write.files(predictSub)
```