Practical Machine Learning

Vineet Jaiswal

July 30, 2017

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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Detail Analysis

Getting the data and do primary analysis

The very first thing is getting the data from specified location and do the primary analysis to be familiar with the records on which you are going to play

```
trainingData <- read.csv("D:\\ML\\Assignment\\Practical-Machine-Learning\\pml-training.csv",na.str
ings = c("NA","", "#DIV/0!"))

testingData <- read.csv("D:\\ML\\Assignment\\Practical-Machine-Learning\\pml-testing.csv",na.strin
gs = c("NA","", "#DIV/0!"))

dim(trainingData)
```

```
## [1] 19622   160
```

```
dim(testingData)
```

```
## [1]  20 160
```

```
# Testing to training difference
setdiff(colnames(testingData), colnames(trainingData))
```

```
## [1] "problem_id"
```

```
# Trainging to testing difference
setdiff(colnames(trainingData), colnames(testingData))
```

```
## [1] "classe"
```

With the analysis, we observed that training data has 19622 records and testing data has 20 records, both has 160 column but training has column “Classe” and testing has column “problem_id”

Cleaning the data for better result

Clean the data like if row has more than 50% NA and also remove unwanted columns which is not useful for this analysis and it may impact our predication

```
# Remove initial 7 columns which is not useful for our analysis
trainingData <- trainingData[, -c(1:7)]
testingData <- testingData[, -c(1:7)]

dim(trainingData)
```

```
## [1] 19622   153
```

```
dim(testingData)
```

```
## [1]  20 153
```

```
# Remove the column which has only NA values
trainingData<-trainingData[,colSums(is.na(trainingData)) == 0]
testingData <-testingData[,colSums(is.na(testingData)) == 0]

dim(trainingData)
```

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

```
dim(testingData)
```

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

Partition training data and visualisation

To perform validation test, we need to partition the training data(60:40) and visualize the data based on “Classe”

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

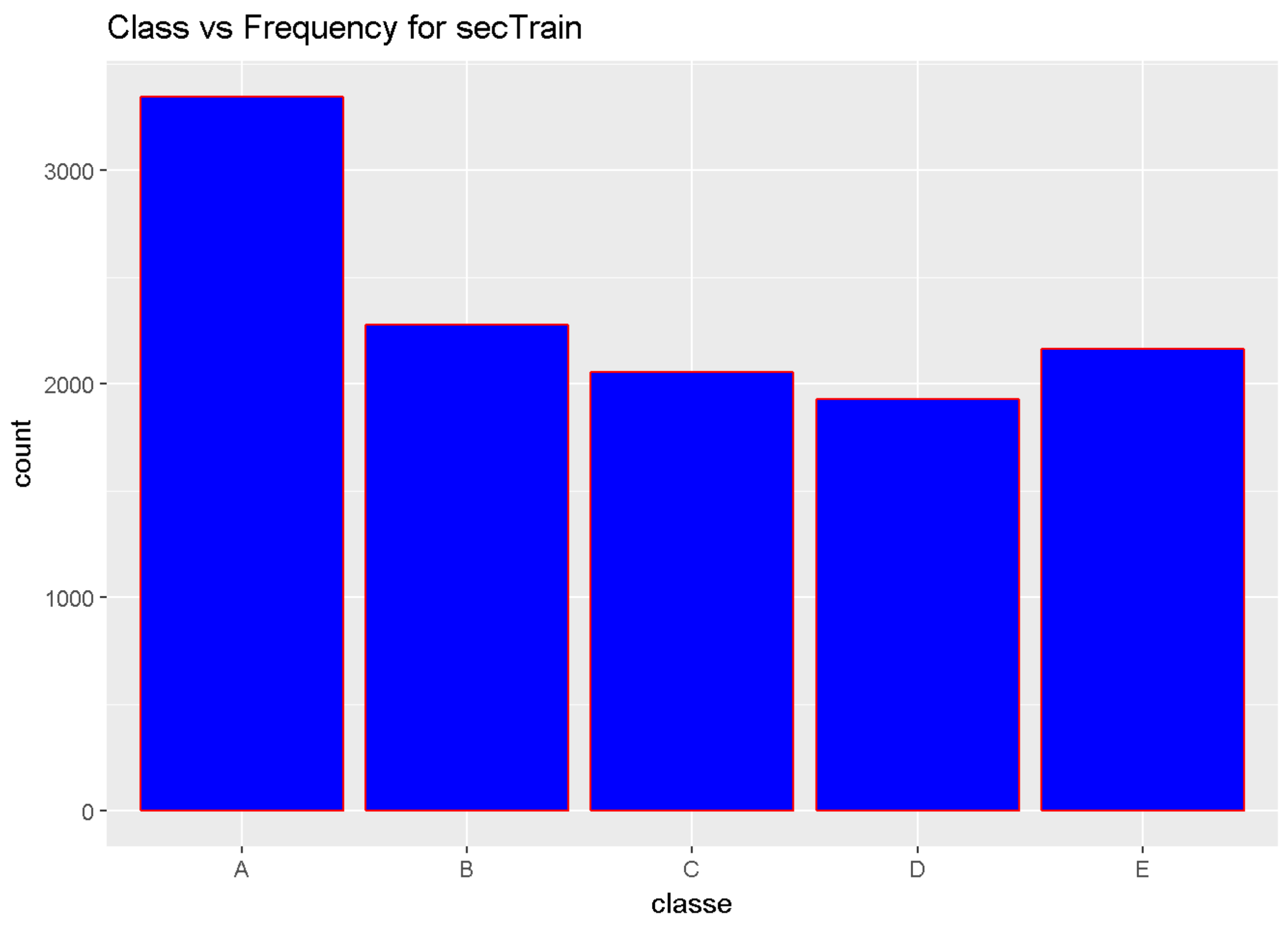
```
set.seed(2000)

partTraining <- createDataPartition(y = trainingData$classe, p=0.6, list = F)

secTrain <- trainingData[partTraining,]
secTest <- trainingData[-partTraining,]

dim(secTrain)
```

```
## [1] 11776    53
```

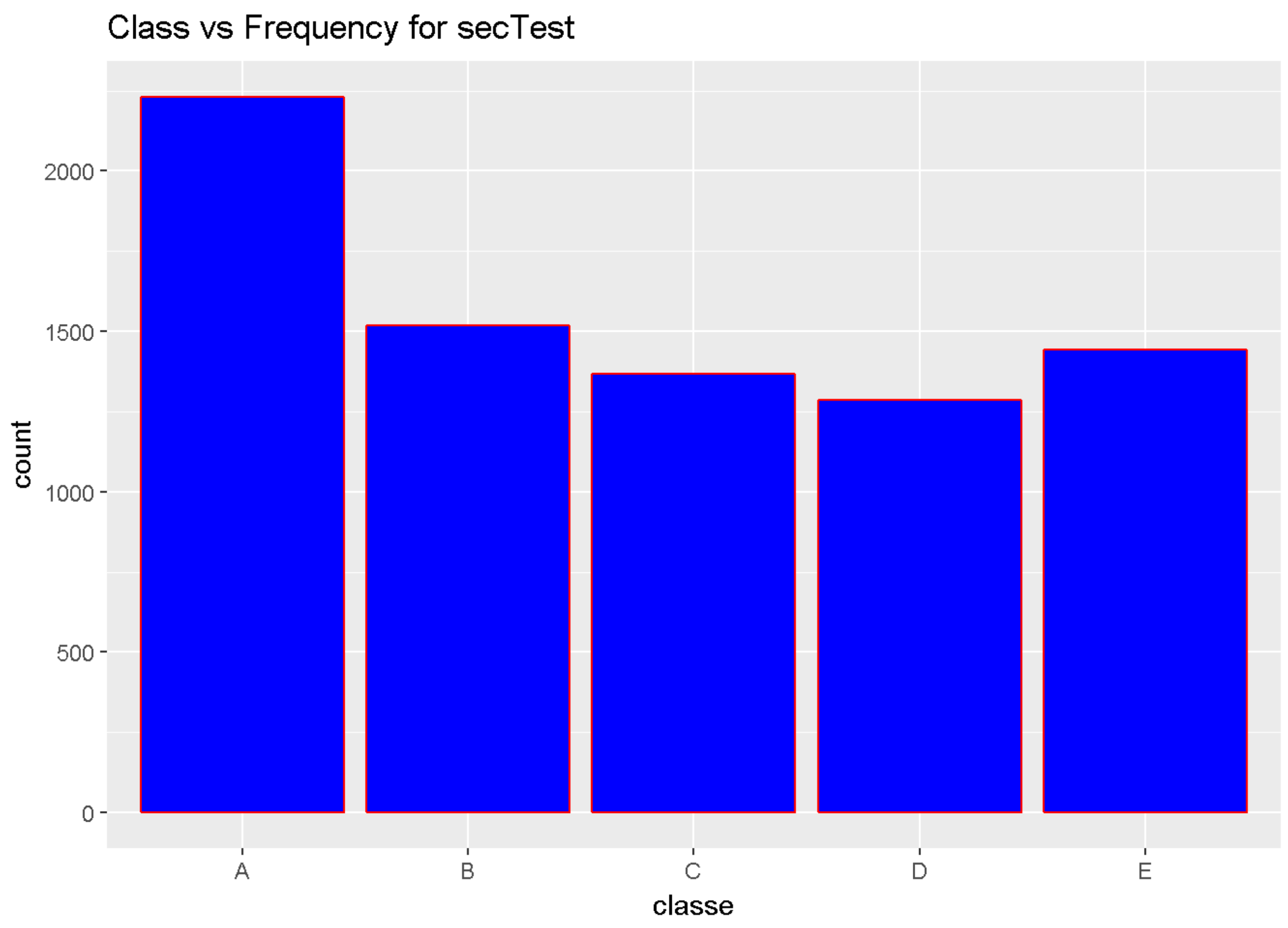
```
dim(secTest)
```

```
## [1] 7846   53
```

```
ggplot(secTrain, aes(classe)) + geom_bar(colour = "red", fill="blue") + xlab("classe") + ylab("cou
nt") + ggtitle("Class vs Frequency for secTrain")
```



```
ggplot(secTest, aes(classe)) + geom_bar(colour = "red", fill="blue") + xlab("classe") + ylab("coun
t") + ggtitle("Class vs Frequency for secTest")
```



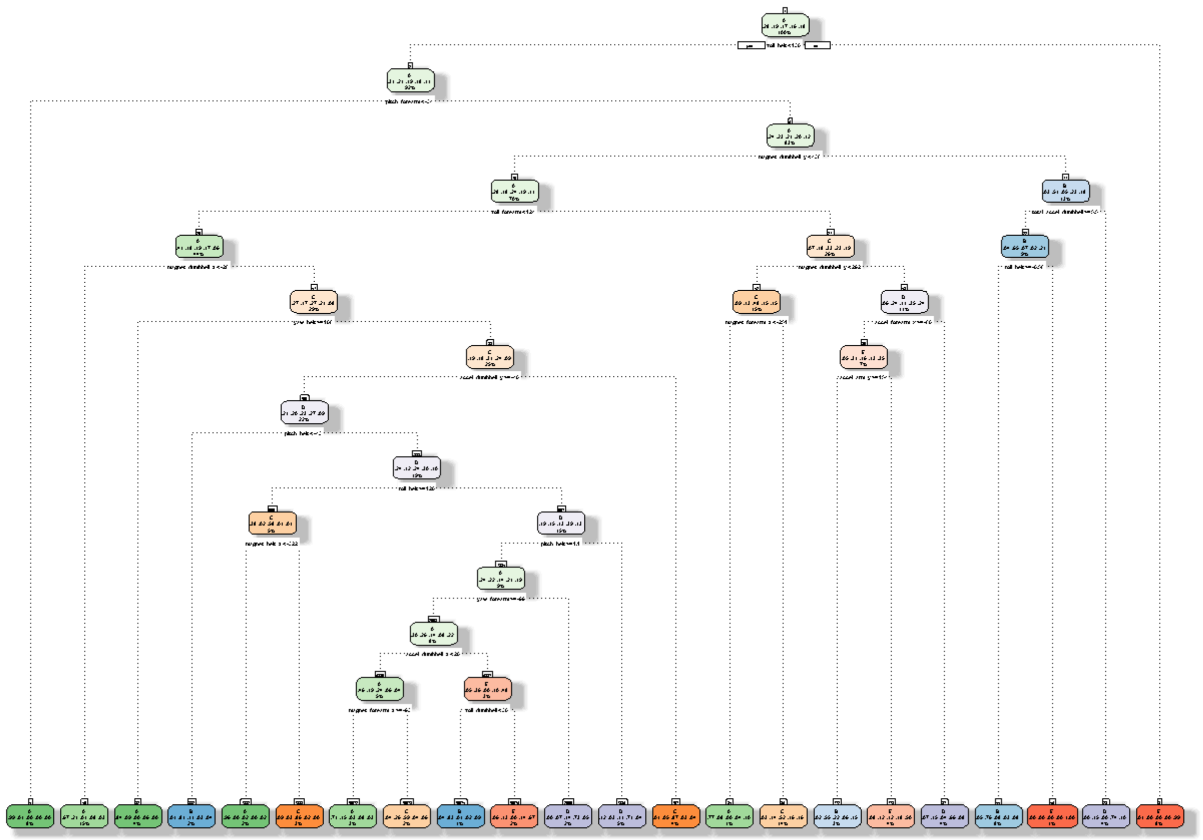
Implement first machine learning algorithm : decision tree

*Decision tree making and visualization

```
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
dt <- rpart(classe~., data = secTrain, method="class")

# View descision tree in more user intractive way
fancyRpartPlot(dt)
```



Rattle 2017-Jul-30 14:33:04 vjais1

- Prediction using classification decision tree

```
predictionDT <-  predict(dt, secTest, type = "class")

# Create confusion matrix to validate the records
confusionMatrix(predictionDT,secTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2004  315   24  153   44
##          B   55  822   95   36   87
##          C   67  192 1131  197  181
##          D   76  108   82  817   92
##          E   30   81   36   83 1038
##
## Overall Statistics
##
##                Accuracy : 0.7408
##                  95% CI : (0.7309, 0.7504)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6708
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8978   0.5415   0.8268   0.6353   0.7198
## Specificity            0.9045   0.9569   0.9017   0.9454   0.9641
## Pos Pred Value         0.7890   0.7507   0.6397   0.6953   0.8186
## Neg Pred Value         0.9570   0.8969   0.9610   0.9297   0.9386
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2554   0.1048   0.1441   0.1041   0.1323
## Detection Prevalence   0.3237   0.1396   0.2253   0.1498   0.1616
## Balanced Accuracy      0.9012   0.7492   0.8642   0.7904   0.8420
```

Implement second machine learning model : random forest

- Creating random forest

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

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

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

```
rf <- randomForest(classe~., data= secTrain, method = "class")
```

- Predicting on random forest classification

```
predictionRF <- predict(rf, secTest, type="class")

confusionMatrix(predictionRF, secTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    6    0    0    0
##          B    3 1508    5    0    0
##          C    0    4 1358   17    0
##          D    0    0    5 1269    2
##          E    0    0    0    0 1440
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9928, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9932
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9934   0.9927   0.9868   0.9986
## Specificity            0.9989   0.9987   0.9968   0.9989   1.0000
## Pos Pred Value         0.9973   0.9947   0.9848   0.9945   1.0000
## Neg Pred Value         0.9995   0.9984   0.9985   0.9974   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1922   0.1731   0.1617   0.1835
## Detection Prevalence   0.2849   0.1932   0.1758   0.1626   0.1835
## Balanced Accuracy      0.9988   0.9961   0.9947   0.9929   0.9993
```

Which machile learning algorithm to choose

As per both confusion matix result, Random forest gives 99% accuracy and decision tree gives 70%, so final analsis will be done on basis for random forest. With random forest, out of sample error is approximate 0.79%

Final Prediction

final prediction based on random forest model

```
finalPrediction <-  predict(rf, testingData, type = "class")
finalPrediction
```

```
##  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
```