Assignment: Prediction Assignment Writeup

Krishan Bhatt

23 April, 2016

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 (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

The test data are available here:

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

What needs to be 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.

For this 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)

wo models will be tested using decision tree and random forest. The model with the highest accuracy will be chosen as our final model.

validation

Cross-validation will be performed by subsampling our training data set randomly without replacement into 2 subsamples: TrainTrainingSet data (75% of the original Training data set) and TestTrainingSet data (25%). Our models will be fitted on the TrainTrainingSet data set, and tested on the TestTrainingSet data. Once the most accurate model is choosen, it will be tested on the original Testing data set.

Install packages and load the required libraries

```r
library(lattice);
library(ggplot2);
library(caret);
```

```
## Warning: package 'caret' was built under R version 3.2.4
```

```r
library(randomForest);
```

```
## Warning: package 'randomForest' was built under R version 3.2.4

## 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
```

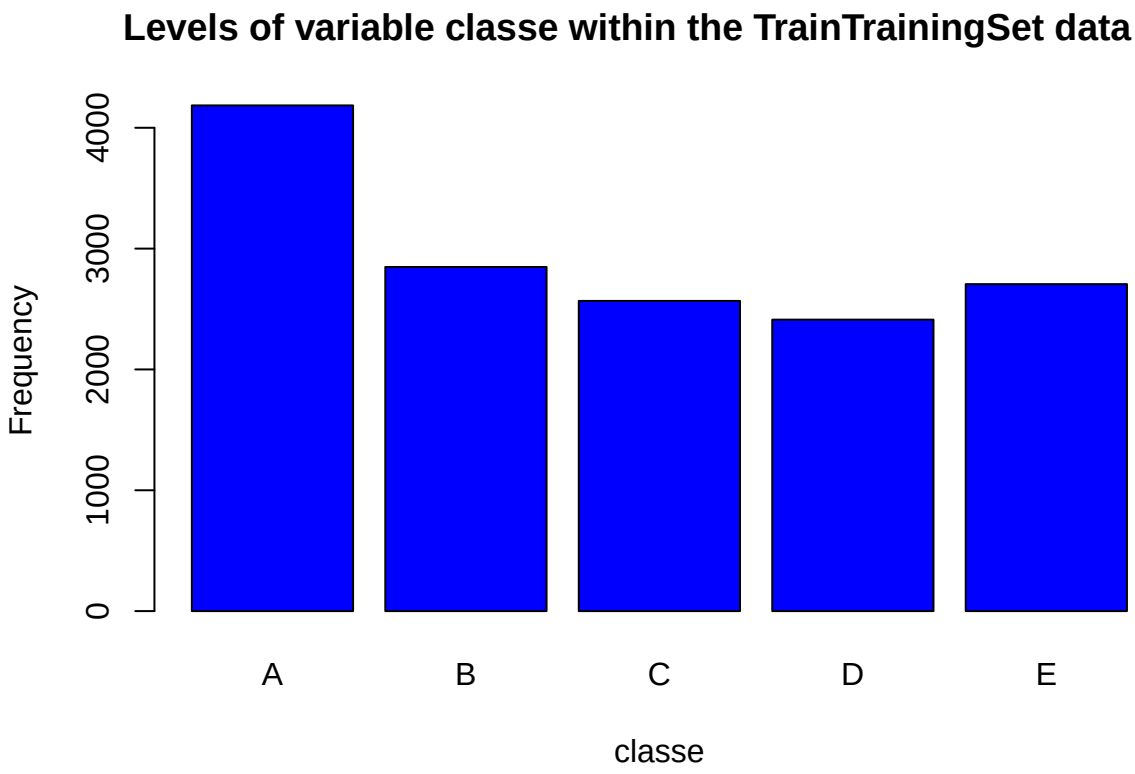
```r
library(rpart);
```

```
## Warning: package 'rpart' was built under R version 3.2.5
```

```r
library(rpart.plot);
```

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

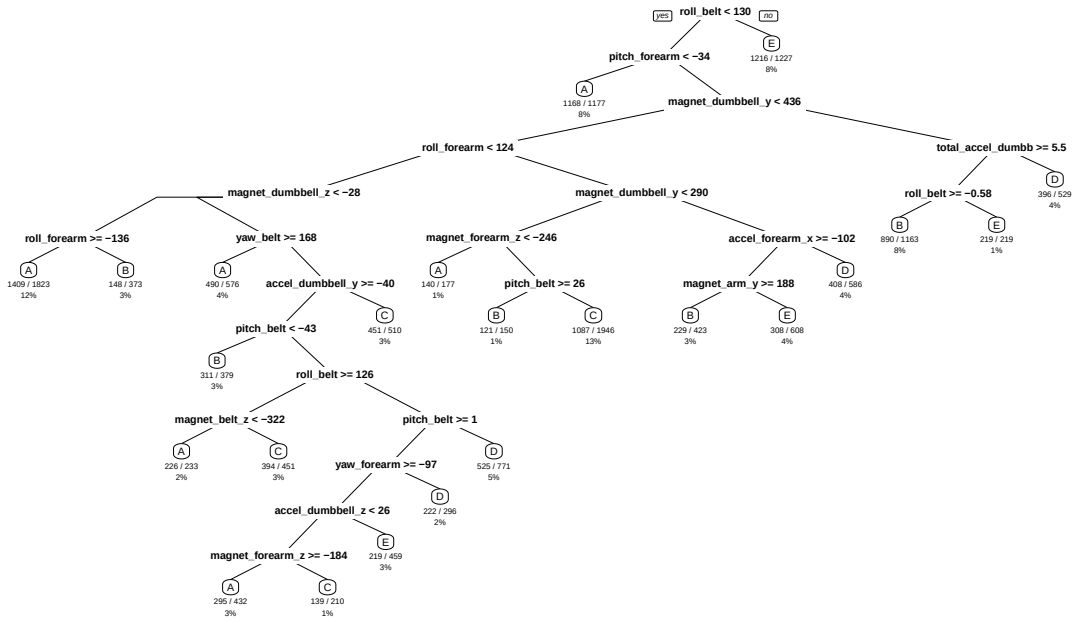
```r
set.seed(2)

# data load and clean up
trainingset <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!", ""))
testingset  <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!", ""))

# Exploratory analysis
# Delete columns with all missing values
trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]
testingset <-testingset[,colSums(is.na(testingset)) == 0]

# Taking only necessary variables, delete irrelevant varialbes

trainingset   <-trainingset[,-c(1:7)]
testingset    <-testingset[,-c(1:7)]

# partition the data so that 75% of the training dataset into training and the remaining 25% to testing
traintrainset <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)
TrainTrainingSet <- trainingset[traintrainset, ]
TestTrainingSet <- trainingset[-traintrainset, ]

# The variable "classe" contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow

plot(TrainTrainingSet$classe, col="blue", main="Levels of variable classe within the TrainTrainingSet da
```



Based on the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent while level D is the least frequent.

Prediction model 1: Decision Tree

```
model1 <- rpart(classe ~ ., data=TrainTrainingSet, method="class")
prediction1 <- predict(model1, TestTrainingSet, type = "class")
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Test results on our TestTrainingSet data set

```
confusionMatrix(prediction1, TestTrainingSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1252  140   16   50   19
##          B   38  540   63   66   80
##          C   34   89  702  125  102
##          D   41   82   50  505   42
##          E   30   98   24   58  658
##
## Overall Statistics
##
##                Accuracy : 0.7457
##                  95% CI : (0.7333, 0.7579)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6778
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8975   0.5690   0.8211   0.6281   0.7303
## Specificity            0.9359   0.9375   0.9136   0.9476   0.9475
## Pos Pred Value         0.8477   0.6861   0.6673   0.7014   0.7581
## Neg Pred Value         0.9583   0.9007   0.9603   0.9285   0.9398
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2553   0.1101   0.1431   0.1030   0.1342
## Detection Prevalence   0.3012   0.1605   0.2145   0.1468   0.1770
## Balanced Accuracy      0.9167   0.7533   0.8673   0.7878   0.8389
```

```
model2 <- randomForest(classe ~. , data=TrainTrainingSet, method="class")
prediction2 <- predict(model2, TestTrainingSet, type = "class")
```

Test results on TestTrainingSet data set

```
confusionMatrix(prediction2, TestTrainingSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    9    0    0    0
##          B    0  939    5    0    0
##          C    0    1  850    5    0
##          D    0    0    0  797    1
##          E    0    0    0    2  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   0.9895   0.9942   0.9913   0.9989
## Specificity            0.9974   0.9987   0.9985   0.9998   0.9995
## Pos Pred Value         0.9936   0.9947   0.9930   0.9987   0.9978
## Neg Pred Value         1.0000   0.9975   0.9988   0.9983   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1733   0.1625   0.1835
## Detection Prevalence   0.2863   0.1925   0.1746   0.1627   0.1839
## Balanced Accuracy      0.9987   0.9941   0.9963   0.9955   0.9992
```

Decision on which Prediction Model to Use:

Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to Decision Tree model with 0.739 (95% CI: (0.727, 0.752)). The

Random Forests model is choosen. The expected out-of-sample error is estimated at 0.005, or 0.5%.

Submission

Here is the final outcome based on the Prediction Model 2 (Random Forest) applied against the Testing dataset

```
predictfinal <- predict(model2, testingset, type="class")
predictfinal
```

```
##  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
```