Practical Machine Learning: Predicting Type of Excerise using Fitness Tracker

Binesh M Shakya

Friday, February 26, 2016

These scripts have been built, tested and produced with windows 7 and R studio Version 0.98.1062

Synopsis

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 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Sources

The **training data** is available here: (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The **testing data** is available here: (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)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 (http://groupware.les.inf.puc-rio.br/har)

Intended Results

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. This report will describes how we built our model, how you we use cross validation, and why we made the choices. We will also use our prediction model to predict 20 different test cases.

Reproducibility

In order to reproduce the same results, we wil need a certain set of packages used in this project and to install, run this command:

```
install.packages("caret")
install.packages("randomForest")
install.packages("rpart")
install.packages("rpart.plot")
install.packages("e1071")
install.packages("ggplot2")
```

Loading data in our working environment and setting overall seed for reproduceablity

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

# setting the overall seed for reproduceability
set.seed(1234)
```

Loading Training Data set and Traning Data Set from URL and set all missing with NA

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "./data/pml-training.csv"
testFile  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile = trainFile, method = "curl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile = testFile, method = "curl")
}
rm(trainUrl)
rm(testUrl)

trainingset <- read.csv(trainFile, na.strings=c("NA","#DIV/0!", ""))
testingset <- read.csv(testFile, na.strings=c("NA","#DIV/0!", ""))
```

Reading Data

```
# Check dimensions for number of variables and number of observations
dim(trainingset)
dim(testingset)

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

# Some variables are irrelevant to our current project: user_name, raw_timestamp_part_1, raw_timestamp_part_,2 cvtd_timestam
p, new_window, and  num_window (columns 1 to 7). We can delete these variables.
trainingset   <-trainingset[,-c(1:7)]
testingset <-testingset[,-c(1:7)]

# and have a look at our new datasets:
dim(trainingset)
dim(testingset)
head(trainingset)
head(testingset)
```

Cross-Validation

The training data set contains 160 variables and 19622 obs. The testing data set contains 160 variables and 20 obs.In order to perform cross-validation, the training data set is partionned into 2 sets: subTraining (75%) and subTest (25%). This will be performed using random subsampling without replacement.

```
subsamples <- createDataPartition(y = trainingset$classe, p = 0.75, list = FALSE)
subTraining <- trainingset[subsamples, ]
subTesting <- trainingset[-subsamples, ]
dim(subTraining)
dim(subTesting)
head(subTraining)
head(subTesting)
```

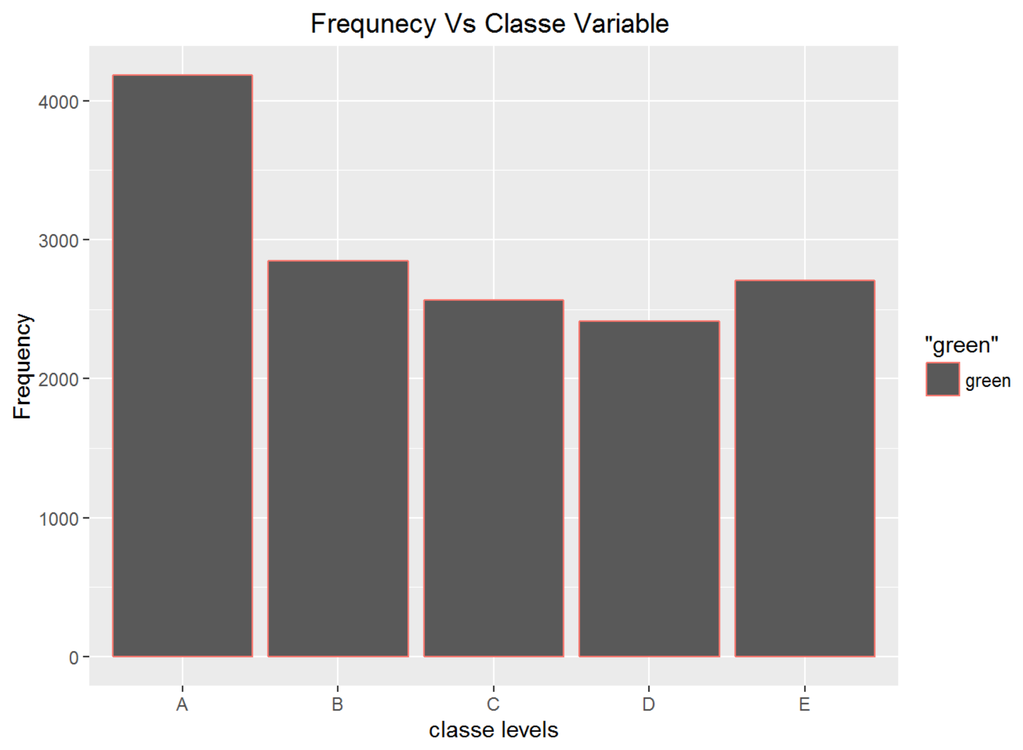
Classe Variable Data vs frequency

The variable “classe” contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the subTraining data set and compare one another.

Classe	Description
A	exactly according to the specification
B	throwing the elbows to the front
C	elifting the dumbbell only halfway
D	lowering the dumbbell only halfway
E	throwing the hips to the front

E throwing the hips to the front

```
qplot(subTraining$classe, geom = "bar", col = "green", main = "Frequnecy Vs Classe Variable ",
    xlab = "classe levels", ylab = "Frequency")
```



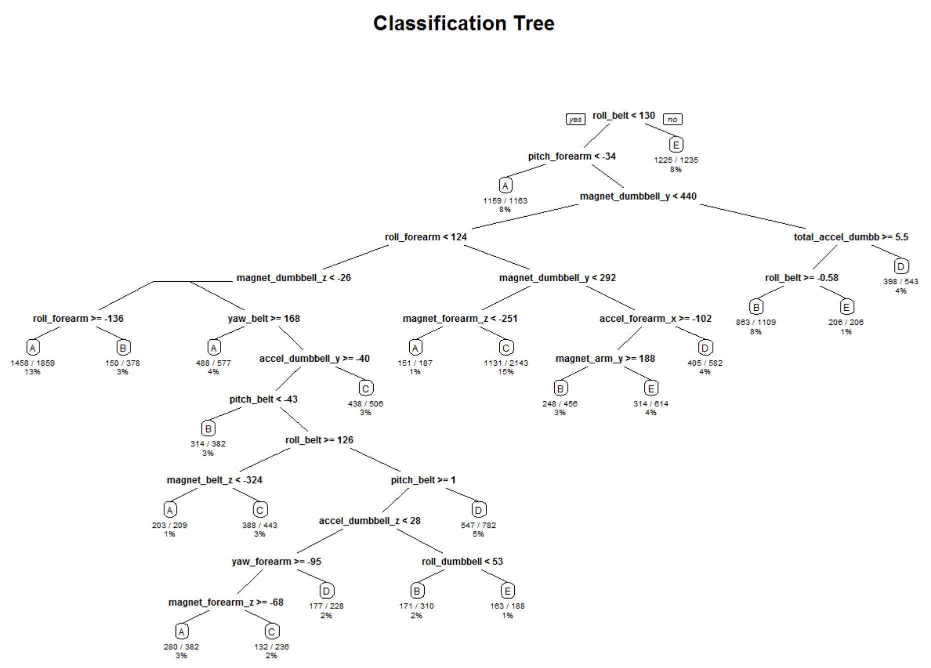
From the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent with more than 4000 occurrences while level D is the least frequent with about 2500 occurrences.

Using Decision Tree Prediction Model

```
model1 <- rpart(classe ~ ., data = subTraining, method = "class")

# Predicting:
prediction1 <- predict(model1, subTesting, type = "class")

# Plot of the Decision Tree
rpart.plot(model1, main = "Classification Tree", extra = 102, under = TRUE,
    faclen = 0)
```



Test results on our subTesting data set:

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

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1235  157   16   50   20
##          B   55  568   73   80  102
##          C   44  125  690  118  116
##          D   41   64   50  508   38
##          E   20   35   26   48  625
## 
## Overall Statistics
## 
##                Accuracy : 0.7394
##                  95% CI : (0.7269, 0.7516)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6697
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8853   0.5985   0.8070   0.6318   0.6937
## Specificity            0.9307   0.9216   0.9005   0.9529   0.9678
## Pos Pred Value         0.8356   0.6469   0.6313   0.7247   0.8289
## Neg Pred Value         0.9533   0.9054   0.9567   0.9296   0.9335
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2518   0.1158   0.1407   0.1036   0.1274
## Detection Prevalence   0.3014   0.1790   0.2229   0.1429   0.1538
## Balanced Accuracy      0.9080   0.7601   0.8537   0.7924   0.8307
```

Using Random Forest Prediction Model

```r
model2 <- randomForest(classe ~ ., data = subTraining, method = "class")

# Predicting:
prediction2 <- predict(model2, subTesting, type = "class")
```

Test results on our subTesting data set:

```r
# Test results on subTesting data set:
confusionMatrix(prediction2, subTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    3    0    0    0
##          B    1  944   10    0    0
##          C    0    2  843    6    0
##          D    0    0    2  798    0
##          E    0    0    0    0  901
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9951          
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9938          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9947   0.9860   0.9925   1.0000
## Specificity            0.9991   0.9972   0.9980   0.9995   1.0000
## Pos Pred Value         0.9979   0.9885   0.9906   0.9975   1.0000
## Neg Pred Value         0.9997   0.9987   0.9970   0.9985   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1925   0.1719   0.1627   0.1837
## Detection Prevalence   0.2849   0.1947   0.1735   0.1631   0.1837
## Balanced Accuracy      0.9992   0.9960   0.9920   0.9960   1.0000
```

Result

The result from above Prection Model using Random Forest vs Decisson Trees, we see Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.9951 (95% CI : (0.9927, 0.9969)) compared to 0.7394 (95% CI : (0.7269, 0.7516)) for Decision Tree model. The accuracy of the model is 0.995. The expected out-of-sample error is estimated at 0.005, or 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 missclassified.

Submission

```r
# predict outcome levels on the original Testing data set using Random
# Forest algorithm
Final_prediction <- predict(model2, testingset, type = "class")
Final_prediction
```

```
##  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
# Write files for submission
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE,
            col.names = FALSE)
    }
}

pml_write_files(Final_prediction)
```

References

- (http://statweb.stanford.edu/~tibs/ElemStatLearn/)http://statweb.stanford.edu/~tibs/ElemStatLearn/ (http://statweb.stanford.edu/~tibs/ElemStatLearn/)
- (http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm (http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)
- (http://www-bcf.usc.edu/~gareth/ISL/)http://www-bcf.usc.edu/~gareth/ISL/ (http://www-bcf.usc.edu/~gareth/ISL/)