Machine Learning

Jill Beck

April 3, 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

Objective

The goal of the project is to predict the manner in which an excercise was performed. To do this, Machine Learning techniques were used on a training set, and used to predict information about a testing set. We will be predicting on the variable "classe" in the data.

Data Processing

Bring in our data, take a look and clean it up. Remove columns with NA values and after reviewing data, make a decision to remove columns 1 through 7 as they are irrelevant to the prediction model.

```
library(caret)
```

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

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

```
library(randomForest)
```

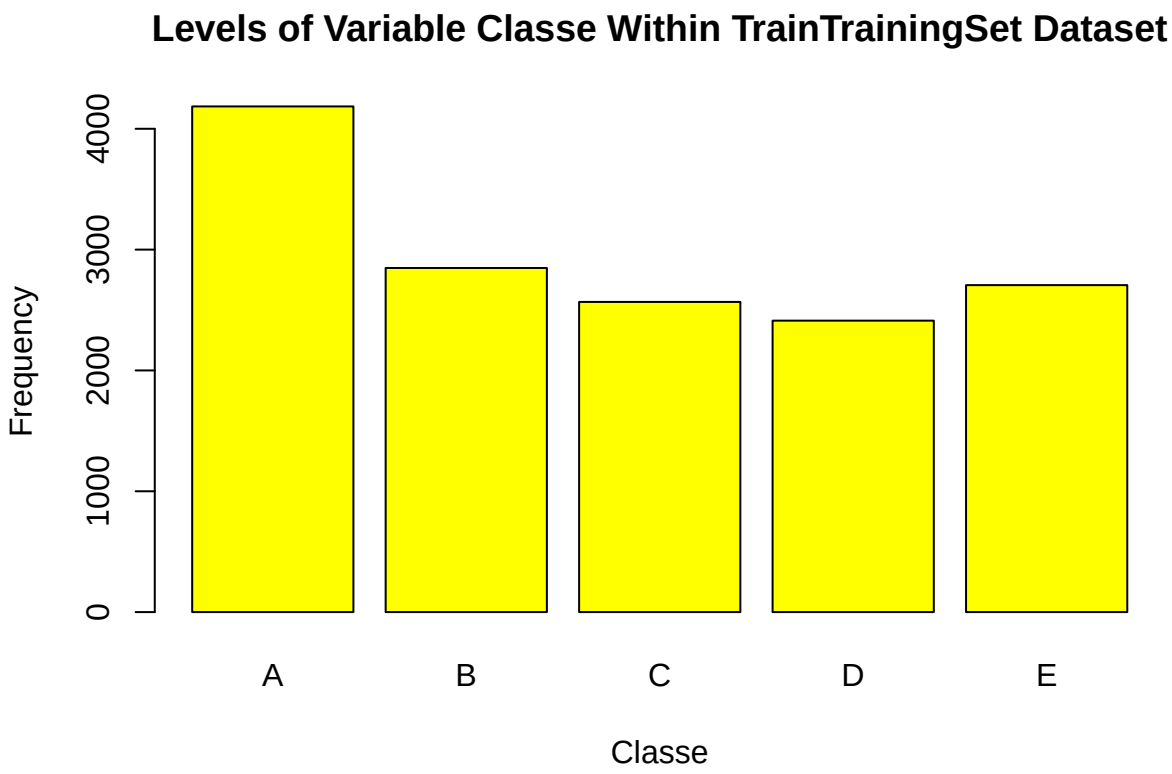
```
## randomForest 4.6-12
```

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

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

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

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

```
set.seed(1234)

# Data loading 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!", ""))

# Perform exploratory analysis.
# dim(trainingset); dim(testingset); summary(trainingset); summary(testingset); str(trainingset); str(t

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

# Delete variables that are irrelevant to current project: user_name, raw_timestamp_part_1, raw_timesta
trainingset <- trainingset[,-c(1:7)]
testingset <- testingset[,-c(1:7)]

# Partition data so that 75% of the training dataset is put into training and the remaining 25% to test
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="yellow", main="Levels of Variable Classe Within TrainTrainingSet Data
```

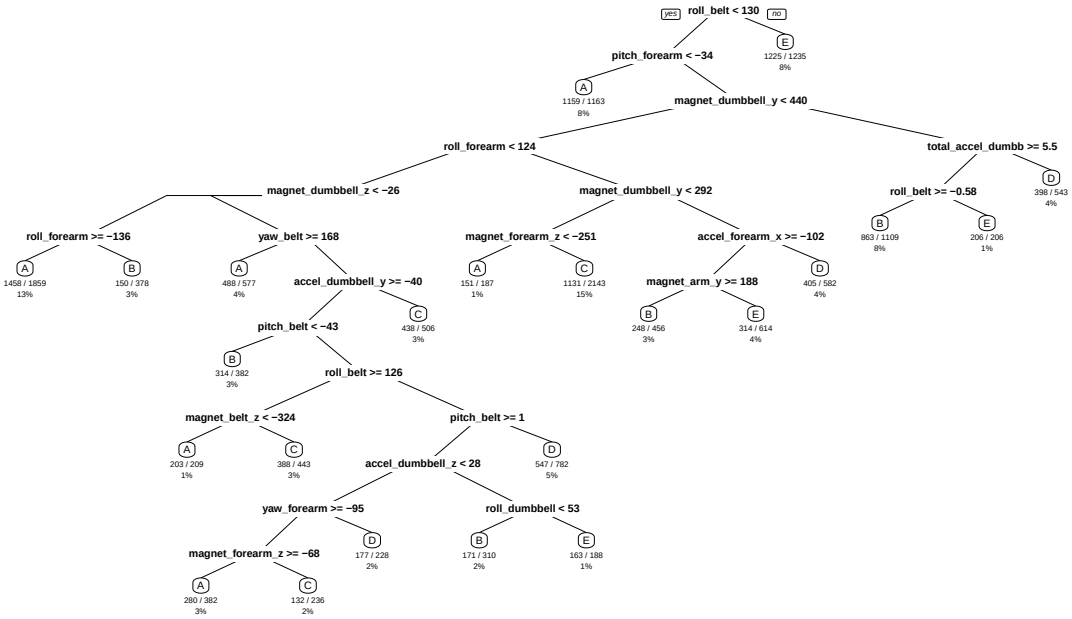


Based on the graph above, each level frequency is within the same order of magnitude. Level A is the most frequent while level D is the least frequent.

Prediction Model 1: Decision Tree

```
model1 <- rpart(classe ~ ., data=TrainTrainingSet, method="class")
prediction1 <- predict(model1, TestTrainingSet, type = "class")

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

Classification Tree




```
# Test results on our TestTrainingSet data set.
confusionMatrix(prediction1, TestTrainingSet$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
```

Prediction Model 2: Random Forest

```
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 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
```

Rationale on Prediction Model To Be Used

The 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 was then selected. The expected out-of-sample error is estimated at 0.005, or 0.5%.

Submission

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

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

```
##  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
```