Practical Machine Learning

Human Exercise Prediction

Niraj Nair

6 June, 2020

Introduction

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Cross-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. ### Getting and Cleaning Data Our data have already been broken into training and test sets. The first step is to load them. I've already downloaded them. Thr required packages are also loaded

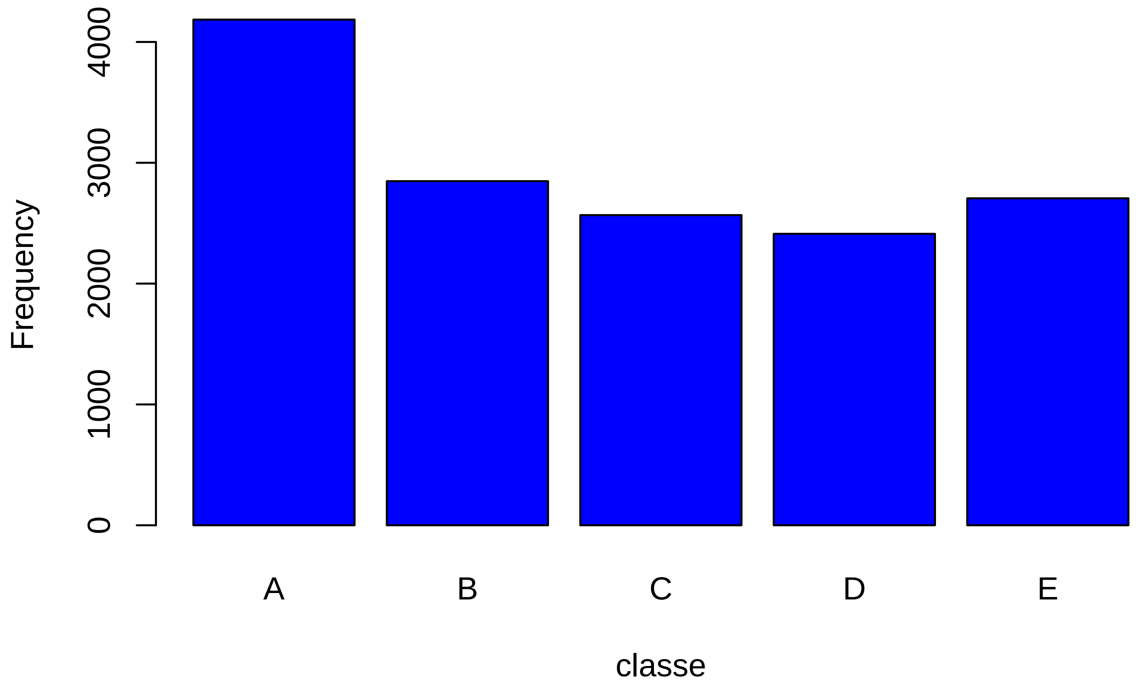
```r
library(lattice); library(ggplot2); library(caret); library(randomForest); library(rpart); library(rpar
```

We'll ignore the test data until we've selected a model. After reading in the training data, there's a little tidying to do:

- Replace empty cells and cells with errors with `NA`s
- Remove rows that only have data for the summary variables

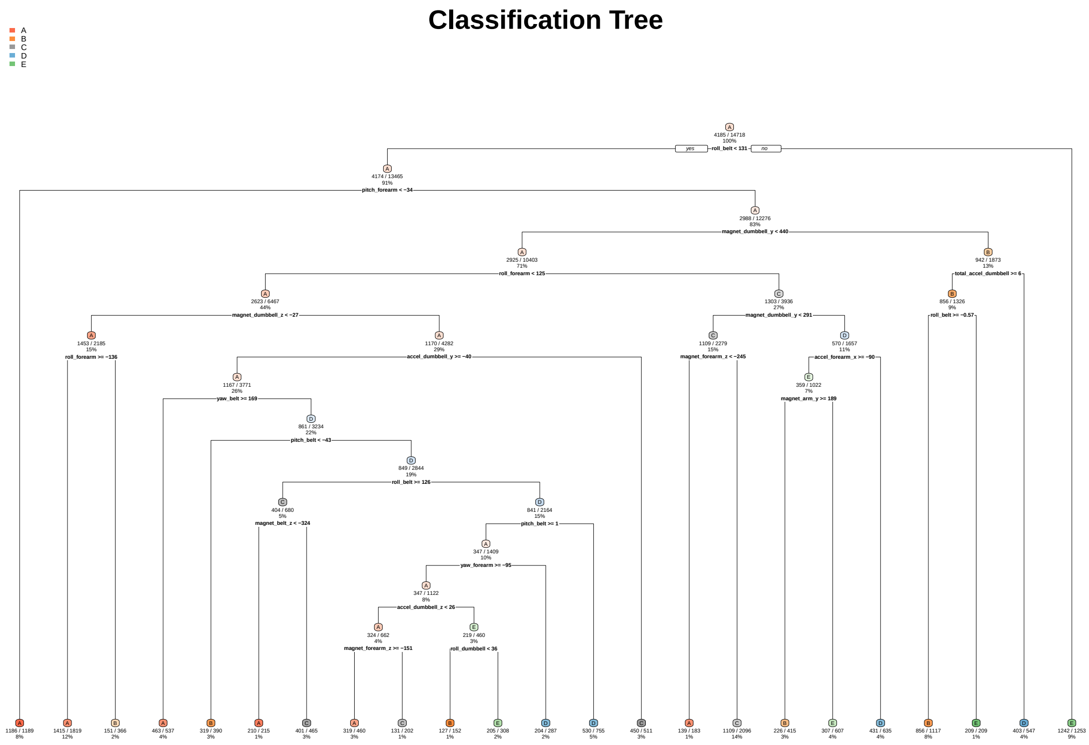
```r
set.seed(1234)

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

trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]
testingset <-testingset[,colSums(is.na(testingset)) == 0]

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

```
traintrainset <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)
TrainTrainingSet <- trainingset[traintrainset, ]
TestTrainingSet <- trainingset[-traintrainset, ]
TrainTrainingSet$classe <- factor(TrainTrainingSet$classe)

plot(TrainTrainingSet$classe, col="blue", xlab="classe", ylab="Frequency")
```



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)
```



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

```
## Confusion Matrix and Statistics
##
##
## prediction1    A    B    C    D    E
##           A 1251  149   15   61   17
##           B   38  572   75   60   75
##           C   39  117  696  117  122
##           D   49   58   51  508   58
##           E   18   53   18   58  629
##
## Overall Statistics
##
##                Accuracy : 0.7455
##                  95% CI : (0.7331, 0.7577)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6774
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8968   0.6027   0.8140   0.6318   0.6981
## Specificity            0.9310   0.9373   0.9024   0.9473   0.9633
## Pos Pred Value         0.8379   0.6976   0.6379   0.7017   0.8106
## Neg Pred Value         0.9578   0.9077   0.9583   0.9292   0.9341
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2551   0.1166   0.1419   0.1036   0.1283
## Detection Prevalence   0.3044   0.1672   0.2225   0.1476   0.1582
## Balanced Accuracy      0.9139   0.7700   0.8582   0.7896   0.8307
```

Prediction model 2: Random Forest

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

prediction2 <- predict(model2, TestTrainingSet, type = "class")

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

```
## Confusion Matrix and Statistics
##
##
## prediction2    A    B    C    D    E
##           A 1395    4    0    0    0
##           B    0  944    8    0    0
##           C    0    1  847    6    0
##           D    0    0    0  798    1
##           E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9906   0.9925   0.9989
## Specificity            0.9989   0.9980   0.9983   0.9998   1.0000
## Pos Pred Value         0.9971   0.9916   0.9918   0.9987   1.0000
## Neg Pred Value         1.0000   0.9987   0.9980   0.9985   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1727   0.1627   0.1835
## Detection Prevalence   0.2853   0.1941   0.1741   0.1629   0.1835
## Balanced Accuracy      0.9994   0.9964   0.9945   0.9961   0.9994
```

Decision

From the above confusion matrices we can understand that random forest is better for prediction than decision tree.

The expected out-of-sample error is estimated at 0.5%.

Conclusion

The outcome of using Prediction Model 2 (Random Forest) on test data is as follows:

```r
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
```