Practical Machine Learning Course Project

Aakil Kadali

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://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

The data for this project come from this source: 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.

```r
# Load Libraries

library(ggplot2)
library(caret)
library(randomForest)
```

```r
# Download files

train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train_name <- "pml-training.csv"
test_name <- "pml-testing.csv"

if (!file.exists(train_name)){
  download.file(train_url,train_name)
}

if (!file.exists(test_name)){
  download.file(test_url,test_name)
}
```

Exploratory Analsysis

```
# Load Data

train <- read.csv(train_name,header=TRUE, na.strings=c("NA","#DIV/0",""))
test <- read.csv(test_name,header=TRUE, na.strings=c("NA","#DIV/0",""))
```

We will need to remove columns with missing values in order to conduct our machine learning algorithms. We can also remove columns that are irelevant to the analysis such as user name, time stamps and window.

```
# Remove NA values
train <- train[,colSums(is.na(train)) == 0]
test <- test[,colSums(is.na(test)) == 0]

# Remove Unnecessary Columns
train <- train[,-c(1:7)]
test <- test[,-c(1:7)]
```

We can take a look at the dimensions of the transformed train data now.

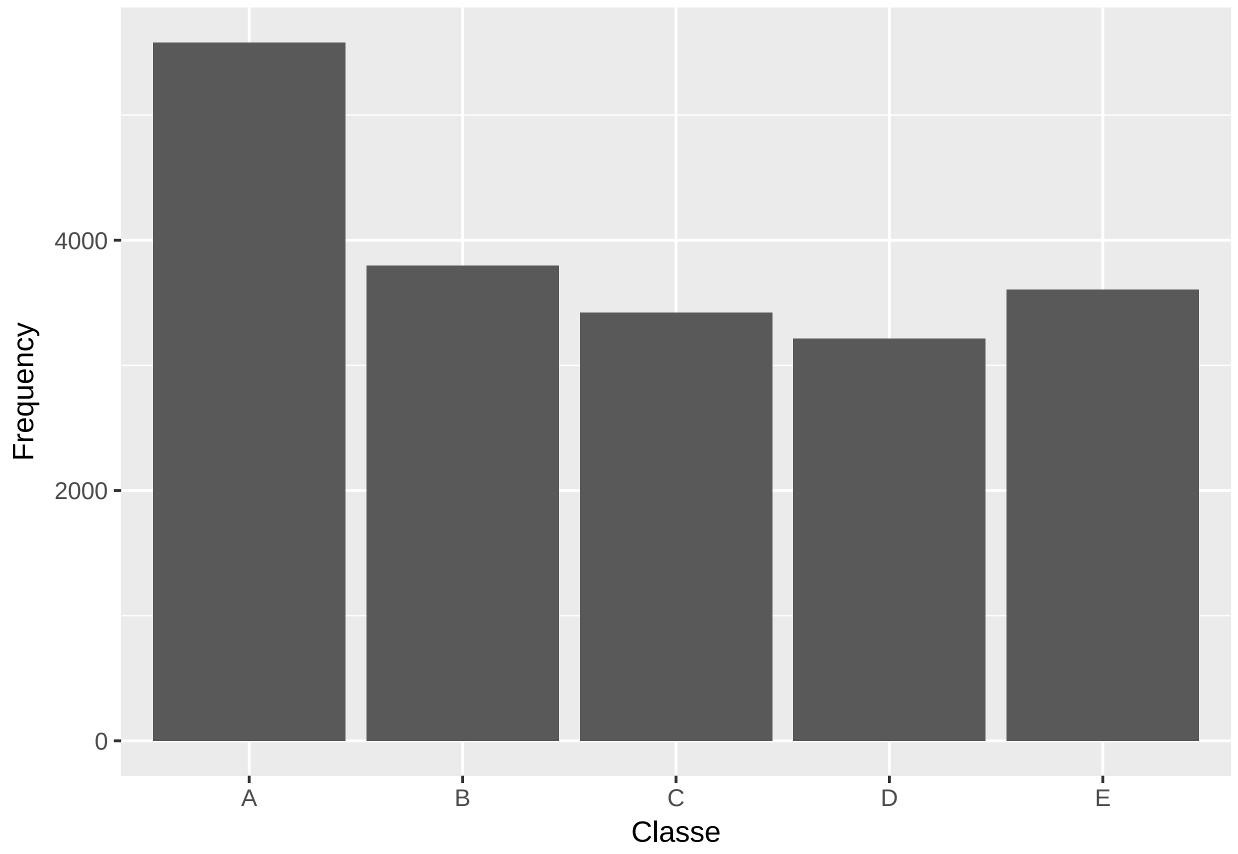
```
dim(train)
```

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

We can also examine the different classes of barbell exercises and erroneous exercises distributed in the training data set. The five classifications of dumbell exercises are:

- Class A: Exactly according to the specification.
- Class B: Throwing the elbows to the front
- Class C: Lifting the dumbell only halfway
- Class D: Lowering the dumbell only halfway
- Class E: Throwing the hips to the front

```
  qplot(x = classe, data = train, xlab = "Classe", ylab = "Frequency")
```



Partitioning

Here, we partition the training data into a training and validation set so that we can test the model.

```
# Set seed
set.seed(8665)
training_set_flag <- createDataPartition(train$classe,p=.7,list = FALSE)

Training_Set <- train[training_set_flag,]
Validation_Set <- train[-training_set_flag,]
```

We can take a look at the dimensions of the training set now.

```
dim(Training_Set)
```

```
## [1] 13737    53
```

Random Forest Modeling

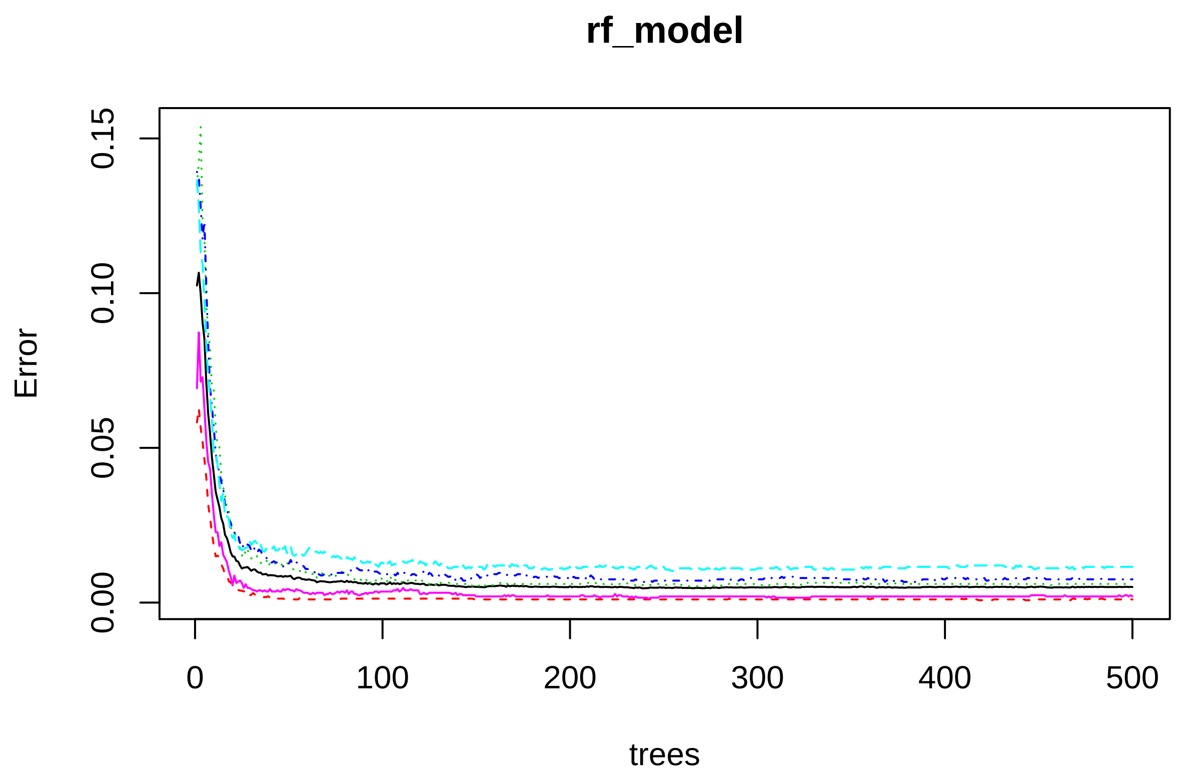
We can use the random forest machine learning algorithm as it is easily implementatble and accurate for larger datasets.

```
set.seed(8665)

rf_model <- randomForest(classe~.,data=Training_Set,importance = TRUE, ntrees = 12)
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Training_Set, importance = TRUE,      ntrees = 12)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.5%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   13 2642    3    0    0 0.006019564
## C    0   15 2378    3    0 0.007512521
## D    1    0   24 2226    1 0.011545293
## E    0    0    1    4 2520 0.001980198
```

```
plot(rf_model)
```



As it can be seen from the plot and the random forest data, the error is minimal with class D having the highest error rate at 1.1%.

We can now cross-validate using the testing data.

```
validation_test <- predict(rf_model,Validation_Set,type = "class")
confusionMatrix(Validation_Set$classe,validation_test)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    7 1129    3    0    0
##          C    0    9 1015    2    0
##          D    0    0    8  954    2
##          E    0    0    0    5 1077
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9939          
##                  95% CI : (0.9915, 0.9957)
##     No Information Rate : 0.2856          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9923          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9921   0.9893   0.9927   0.9981
## Specificity            1.0000   0.9979   0.9977   0.9980   0.9990
## Pos Pred Value         1.0000   0.9912   0.9893   0.9896   0.9954
## Neg Pred Value         0.9983   0.9981   0.9977   0.9986   0.9996
## Prevalence             0.2856   0.1934   0.1743   0.1633   0.1833
## Detection Rate         0.2845   0.1918   0.1725   0.1621   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9950   0.9935   0.9953   0.9986
```

Here, we see that the accuracy of the test is 99.39%

Prediction

Now we can use the prediction set using the test data set.

```r
predictTest <- predict(rf_model,newdata=test)
predictTest
```

```
##  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
```