Practical Machine Learning Project

Vedant Walia

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 dumbbell 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 Source

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

Loading the Dataset

My goal in this project is to predict the manner in which they did the exercise via the *classe* variable within the data set.

```
library(randomForest)
library(rpart)
library(ggplot2)
library(caret)
library(gbm)
library(plyr)

download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "./pml-training.csv", method = "curl")

download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              destfile = "./pml-testing.csv", method = "curl")

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

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

Cleaning the Data

Now let us clean the data by removing the irrelevant parts like the columns with no data and also the ones which won't be helping me with the prediction.

```
features <- names(testing[,colSums(is.na(testing)) == 0])[8:59]
training <- training[,c(features,"classe")]
testing <- testing[,c(features,"problem_id")]
```

Partitioning the Data

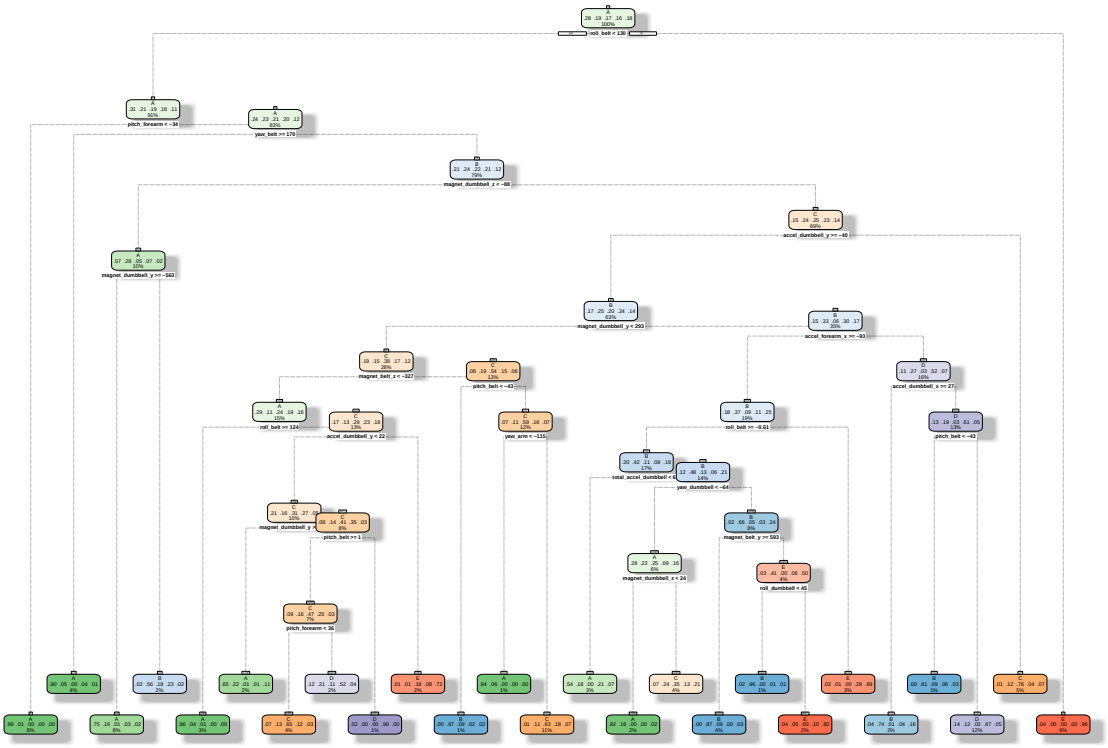
we will split our data into a training data set (60% of the total cases) and a testing data set (40% of the total cases). This will allow us to estimate the out of sample error of our predictor.

```
set.seed(2121212)

inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
training_part <- training[inTrain,]
testing_part <- training[-inTrain,]
```

Building the Decision Tree

```
set.seed(2121212)
library(rattle)
training_tree <- rpart(classe ~ ., data = training_part,
                       method="class",
                       control = rpart.control(method = "cv", number = 10))
fancyRpartPlot(training_tree)
```



Rattle 2020−Oct−15 18:44:16 vedan

Predicting with the Decision Tree

```
set.seed(2121212)
prediction <- predict(training_tree, testing_part, type = "class")
classe <- as.factor(testing_part$classe)
confusionMatrix(prediction, classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1936  265   19  115   67
##          B   10  840   72   60   60
##          C   55  257 1183  238  202
##          D  174  146   53  784   77
##          E   57   10   41   89 1036
##
## Overall Statistics
##
##                Accuracy : 0.7366
##                  95% CI : (0.7267, 0.7463)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.6663
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8674   0.5534   0.8648  0.60964   0.7184
## Specificity            0.9170   0.9681   0.8839  0.93140   0.9692
## Pos Pred Value         0.8060   0.8061   0.6114  0.63533   0.8402
## Neg Pred Value         0.9456   0.9004   0.9687  0.92408   0.9386
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2467   0.1071   0.1508  0.09992   0.1320
## Detection Prevalence   0.3061   0.1328   0.2466  0.15728   0.1572
## Balanced Accuracy      0.8922   0.7607   0.8743  0.77052   0.8438
```

From the above table we can see that the accuracy of the Decision Tree Model is around 75%.

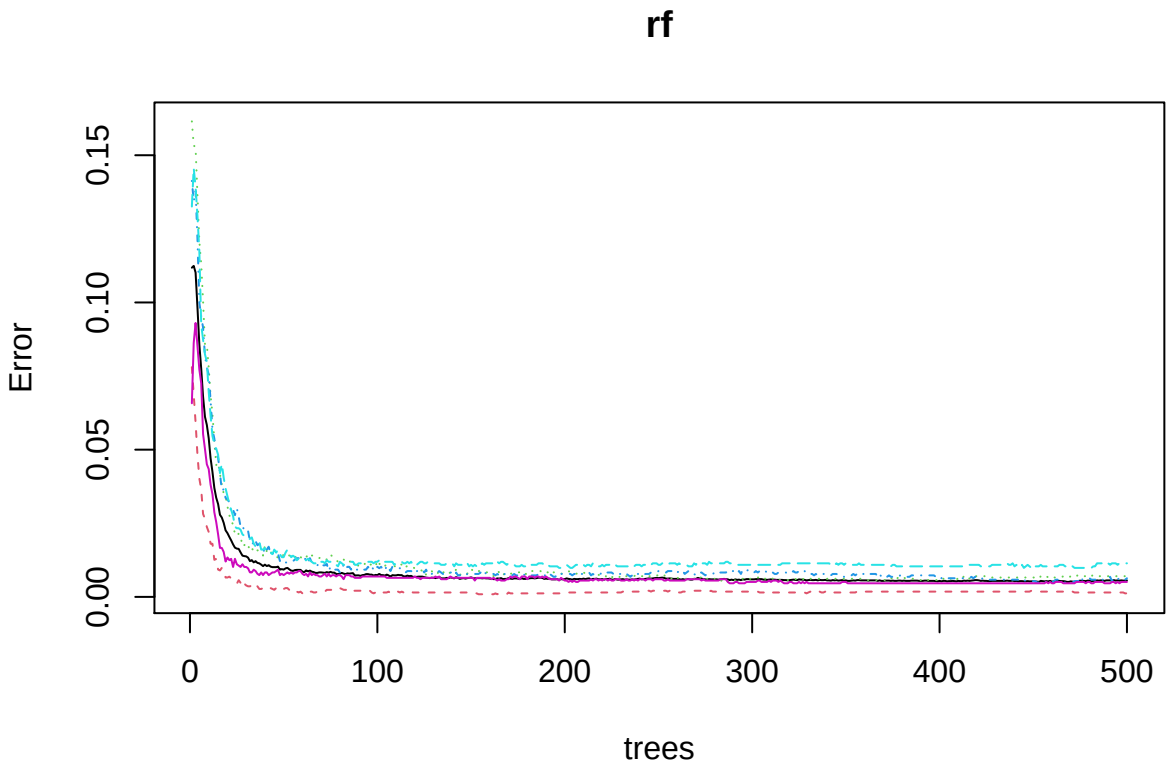
Building Random Forest Model

Now let's create a model with random forest to see if the accuracy improves.

```
set.seed(2121212)

rf <- randomForest(as.factor(classe) ~ ., data = training_part,
                   method = "rf",
                   importance = T,
                   trControl = trainControl(method = "cv",
                                            classProbs=TRUE,
                                            savePredictions=TRUE,
                                            allowParallel=TRUE,
                                            number = 10))

plot(rf)
```



Predicting with Random Forest

One of the nice features of the random forest algorithm is that when decision trees are created, the algorithm withholds a fraction of the samples for testing while the model is being created. This allows random forests to make a reasonably unbiased measure of out-of-sample error without doing traditional cross-validation.

Let's try and predict the outcomes now and see the accuracy of the same.

```
prediction <- predict(rf, testing_part, type = "class")
classe <- as.factor(testing_part$classe)
confusionMatrix(prediction, classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    6    0    0    0
##          B    0 1505    7    0    0
##          C    0    7 1358   13    2
##          D    0    0    3 1270    1
##          E    0    0    0    3 1439
##
## Overall Statistics
##
##                Accuracy : 0.9946
```

```
##                  95% CI : (0.9928, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9932
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9914   0.9927   0.9876   0.9979
## Specificity            0.9989   0.9989   0.9966   0.9994   0.9995
## Pos Pred Value         0.9973   0.9954   0.9841   0.9969   0.9979
## Neg Pred Value         1.0000   0.9979   0.9985   0.9976   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1918   0.1731   0.1619   0.1834
## Detection Prevalence   0.2852   0.1927   0.1759   0.1624   0.1838
## Balanced Accuracy      0.9995   0.9952   0.9946   0.9935   0.9987
```

From the above table we can see that the accuracy of the Random Forest Model is around 99%. Hence, we it would be better to use it for predicting the values for the testing data (*pml-testing.csv*).

```
predict_RF <- predict(rf, testing)
predict_RF
```

```
##  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
```