Machine Learning Project

Arijit Nath

October 27, 2018

Overview

This document is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning.The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data.

Background

Human Activity Recognition (HAR) is a key research area that is gaining increasing attention, especially for the development of context-aware systems. There are many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

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

Download athe data

```
library(caret)
library(randomForest)

traindata <- "pml_training.csv"
if (!file.exists(traindata)) {
    url <-
        "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
   # target <- "pml_training.csv"
    download.file(url, destfile = traindata)
}


testdata <- "pml_testing.csv"
if (!file.exists(testdata)) {
    url <-
        "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
    download.file(url, destfile = testdata)
}
```

Load & CleanUp the data

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

```
## [1] 19622   160
```

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

```
## [1]  20 160
```

Data Partitioning

Since we are going to predict classes in the testing dataset, We'll split the training data into training and testing partitions and use the test file (pml-testing.csv) as a validation sample. We'll use cross validation within the training partition to improve the model fit and then do an out-of-sample test with the testing partition.

```
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
trainset <- training[inTrain, ]
testset  <- training[-inTrain, ]
dim(trainset)
```

```
## [1] 13737   160
```

```
dim(testset)
```

```
## [1] 5885  160
```

Check for near zero values

Both created datasets have 160 variables. In order to predict classes in the validation sample, We'll need to use features that are non-zero in the validation data set.

```
nzv <- nearZeroVar(trainset)
trainset <- trainset[, -nzv]
testset  <- testset[, -nzv]
dim(trainset)
```

```
## [1] 13737   132
```

```
dim(testset)
```

```
## [1] 5885  132
```

```
mostlyNA    <- sapply(trainset, function(x) mean(is.na(x))) > 0.95
trainset <- trainset[, mostlyNA==FALSE]
testset  <- testset[, mostlyNA==FALSE]
dim(trainset)
```

```
## [1] 13737    59
```

```
trainset <- trainset[, -(1:5)]
testset  <- testset[, -(1:5)]
dim(trainset)
```

```
## [1] 13737    54
```

```
dim(testset)
```

```
## [1] 5885   54
```

Prediction Model Building

We'll be using Random Forest method for our data mdodelling

```
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modfitRF <- train(classe ~ ., data=trainset, method="rf",
                          trControl=controlRF)
modfitRF$finalModel
```

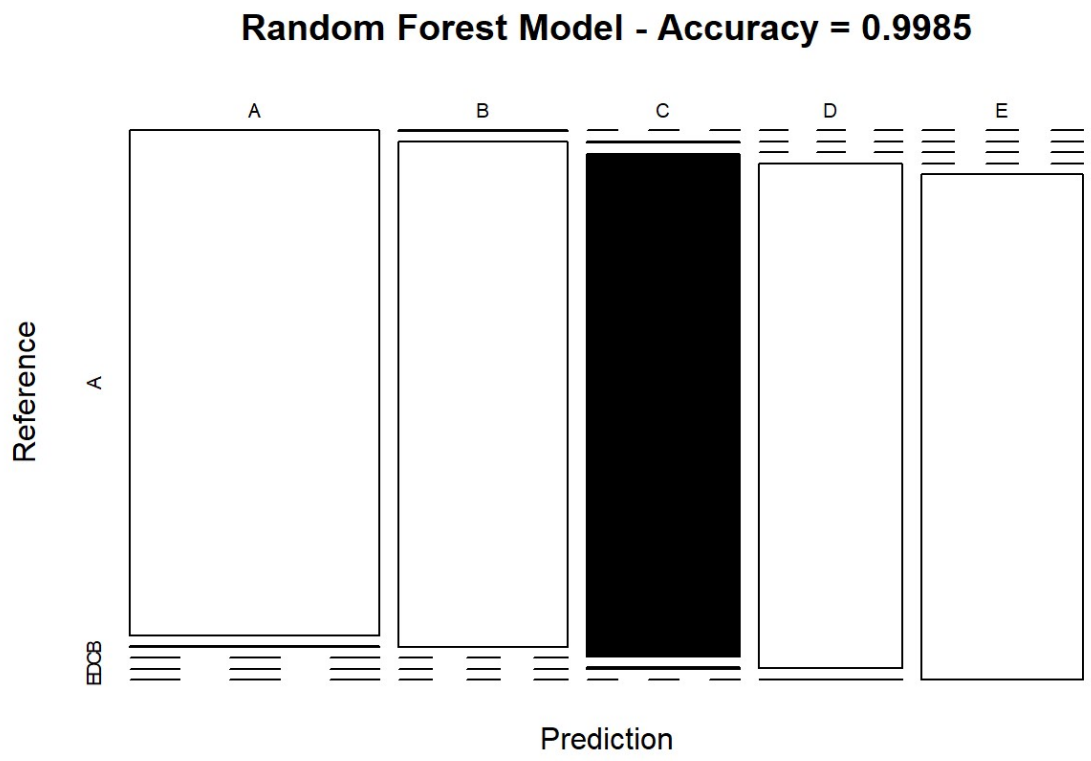
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.28%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    7 2648    2    1    0 0.0037622272
## C    0    5 2391    0    0 0.0020868114
## D    0    0   13 2238    1 0.0062166963
## E    0    0    0    8 2517 0.0031683168
```

Prediction on test data

```
predictRF <- predict(modfitRF, newdata=testset)
confMatRF <- confusionMatrix(predictRF, testset$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    1 1135    0    0    0
##          C    0    3 1026    3    0
##          D    0    0    0  961    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9971, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   1.0000   0.9969   0.9991
## Specificity            0.9998   0.9998   0.9988   0.9998   1.0000
## Pos Pred Value         0.9994   0.9991   0.9942   0.9990   1.0000
## Neg Pred Value         0.9998   0.9992   1.0000   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1743   0.1633   0.1837
## Detection Prevalence   0.2845   0.1930   0.1754   0.1635   0.1837
## Balanced Accuracy      0.9996   0.9981   0.9994   0.9983   0.9995
```

```
plot(confMatRF$table, col = confMatRF$byClass,
     main = paste("Random Forest Model - Accuracy =",
                  round(confMatRF$overall['Accuracy'], 4)))
```



Applying our Model to predict on the Test Data

Here, we have used Random Forest as our model, which has accuracy of .99, we applying the same on our test data to predict the results.

```
predictTEST <- predict(modfitRF, newdata=testing)
predictTEST
```

```
##  [1] 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
```