Practical Machine Learning - Peer Assessment Report

Ashish Verma

03 June 2017

Brief Requirement

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, our 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).

The goal of this analysis is to predict the manner in which they did the exercise. We should create a report describing how we built our model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did.

Basic Data Preparation and Environment Setup

```
## Set appropriate working where the downloaded training file has been saved
  setwd("C:\\Users\\Ashish\\Downloads\\")

## Load the necessary dependant packages
  library(caret)
  library(randomForest)
  library(ggplot2)
  library(Hmisc)
```

Data Loading

```
## Load the data
  trainingset <- read.csv("pml-training.csv",row.names=1,na.strings = "")
  testingset <- read.csv("pml-testing.csv",row.names=1,na.strings = "")

## View Data Dimensions
  dim(trainingset)
```

```
## [1] 19622   159
```

Basic Data Cleansing

In this step we will do 2 types of data cleansing:
a) Remove those columns which have “near zero variance”
b) Remove the columns which have missing values

```
# Remove near zero covariates
  datanzv <- nearZeroVar(trainingset,saveMetrics=TRUE)
  trainingset <- trainingset[,!datanzv$nzv]
  testingset <- testingset[,!datanzv$nzv]

# Remove columns with missing values
  training_filter_missing <- trainingset[,(colSums(is.na(trainingset)) == 0)]
  testing_filter_missing <- testingset[,(colSums(is.na(testingset)) == 0)]
```

Now we will distribute the training data in training set and validation set by using “classe” variable

```
## Split the data into 70% training and 30% validation records
  intrain <- createDataPartition(y=trainingset$classe, p=0.7, list=FALSE)
  final_training <- training_filter_missing[intrain,]
  final_validation <- training_filter_missing[-intrain,]

## Check dimensions of the cleansed records
  dim(final_training)
```

```
## [1] 13737    58
```

```
## Find correlations among various columns
  correlations <- abs(sapply(colnames(final_training[, -ncol(trainingset)]), funct
ion(x) cor(as.numeric(final_training[, x]),as.numeric(final_training$classe), metho
d = "spearman")))
  correlations
```

```
##            user_name raw_timestamp_part_1 raw_timestamp_part_2
##         0.0133077250         0.1950000227         0.0154180740
##       cvtd_timestamp           num_window            roll_belt
##         0.1360270048         0.0108307435         0.1189657524
##           pitch_belt             yaw_belt     total_accel_belt
##         0.0477582830         0.0683941737         0.0800431519
##         gyros_belt_x         gyros_belt_y         gyros_belt_z
##         0.0126501598         0.0055680203         0.0058297037
##         accel_belt_x         accel_belt_y         accel_belt_z
##         0.0429561216         0.0224825430         0.1300106471
##        magnet_belt_x        magnet_belt_y        magnet_belt_z
##         0.0009860016         0.1982514071         0.1425068941
##             roll_arm            pitch_arm              yaw_arm
##         0.0569486091         0.1860952444         0.0288545708
##      total_accel_arm          gyros_arm_x          gyros_arm_y
##         0.1576125379         0.0204486530         0.0254061455
##          gyros_arm_z          accel_arm_x          accel_arm_y
##         0.0066934404         0.2500624781         0.0736214851
##          accel_arm_z         magnet_arm_x         magnet_arm_y
##         0.1070635786         0.2779174776         0.2617600283
##         magnet_arm_z        roll_dumbbell       pitch_dumbbell
##         0.1520013183         0.0858189705         0.0961772549
##         yaw_dumbbell total_accel_dumbbell     gyros_dumbbell_x
##         0.0089913226         0.0142787680         0.0130120674
##     gyros_dumbbell_y     gyros_dumbbell_z     accel_dumbbell_x
##         0.0188649811         0.0176157513         0.1279728030
##     accel_dumbbell_y     accel_dumbbell_z    magnet_dumbbell_x
##         0.0149409345         0.0819537438         0.1523116477
##    magnet_dumbbell_y    magnet_dumbbell_z         roll_forearm
##         0.0517475527         0.2038604700         0.0491153067
##        pitch_forearm          yaw_forearm  total_accel_forearm
##         0.3207879579         0.0479063214         0.1196584659
##      gyros_forearm_x      gyros_forearm_y      gyros_forearm_z
##         0.0173491404         0.0015189318         0.0033133390
##      accel_forearm_x      accel_forearm_y      accel_forearm_z
##         0.2052609496         0.0201284083         0.0108829061
##     magnet_forearm_x     magnet_forearm_y     magnet_forearm_z
##         0.1938893248         0.1134356752         0.0525193493
##               classe
##         1.0000000000
```

Inference from above correlation:
There seems to be no predictors which are strongly correlated with the outcome variable, so linear regression model may not be a good option to proceed. Hence, Random forest model will be used further to model.

Detailed Data Modelling using Random Forest Method

```
## Set desired seed
  set.seed(1234)
## Fit rf model
  rf_fit_model <- randomForest(classe~.,data=final_training)
  rf_fit_model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = final_training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.09%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3906    0    0    0    0 0.0000000000
## B    1 2657    0    0    0 0.0003762227
## C    0    3 2393    0    0 0.0012520868
## D    0    0    5 2247    0 0.0022202487
## E    0    0    0    3 2522 0.0011881188
```

Model Evalution and Inferencing

```
## Find the impotance on the model
  importance(rf_fit_model)
```

```
##                      MeanDecreaseGini
## user_name                   128.04077
## raw_timestamp_part_1       1135.09670
## raw_timestamp_part_2         10.88581
## cvtd_timestamp             1641.75938
## num_window                  653.40684
## roll_belt                   603.06749
## pitch_belt                  348.23349
## yaw_belt                    389.96098
## total_accel_belt            147.53590
## gyros_belt_x                 43.42132
## gyros_belt_y                 58.09941
## gyros_belt_z                139.30409
## accel_belt_x                 71.73682
## accel_belt_y                 75.44127
## accel_belt_z                216.18054
## magnet_belt_x               122.22995
## magnet_belt_y               227.28563
## magnet_belt_z               213.91378
## roll_arm                    124.59484
## pitch_arm                    62.59288
## yaw_arm                      84.76913
## total_accel_arm              30.54858
## gyros_arm_x                  47.56700
## gyros_arm_y                  47.91339
## gyros_arm_z                  21.10711
## accel_arm_x                 106.71436
## accel_arm_y                  61.38015
## accel_arm_z                  42.80959
## magnet_arm_x                102.98663
## magnet_arm_y                 82.91898
## magnet_arm_z                 63.45366
## roll_dumbbell               229.49099
```

```
## pitch_dumbbell              88.77596
## yaw_dumbbell               139.87975
## total_accel_dumbbell       142.48631
## gyros_dumbbell_x            47.48819
## gyros_dumbbell_y           108.40330
## gyros_dumbbell_z            27.09812
## accel_dumbbell_x           151.62741
## accel_dumbbell_y           238.08454
## accel_dumbbell_z           155.36619
## magnet_dumbbell_x          258.81678
## magnet_dumbbell_y          378.79281
## magnet_dumbbell_z          368.32203
## roll_forearm               283.95978
## pitch_forearm              345.89839
## yaw_forearm                 64.82954
## total_accel_forearm         35.43667
## gyros_forearm_x             26.84669
## gyros_forearm_y             43.18445
## gyros_forearm_z             30.19199
## accel_forearm_x            151.13179
## accel_forearm_y             51.97917
## accel_forearm_z            106.43727
## magnet_forearm_x            80.67360
## magnet_forearm_y            88.37910
## magnet_forearm_z           111.34509
```

```
## Evaluate using confusion matrix
  confusionMatrix(predict(rf_fit_model,newdata=final_validation[,-ncol(final_valid
ation)]),final_validation$classe)
```

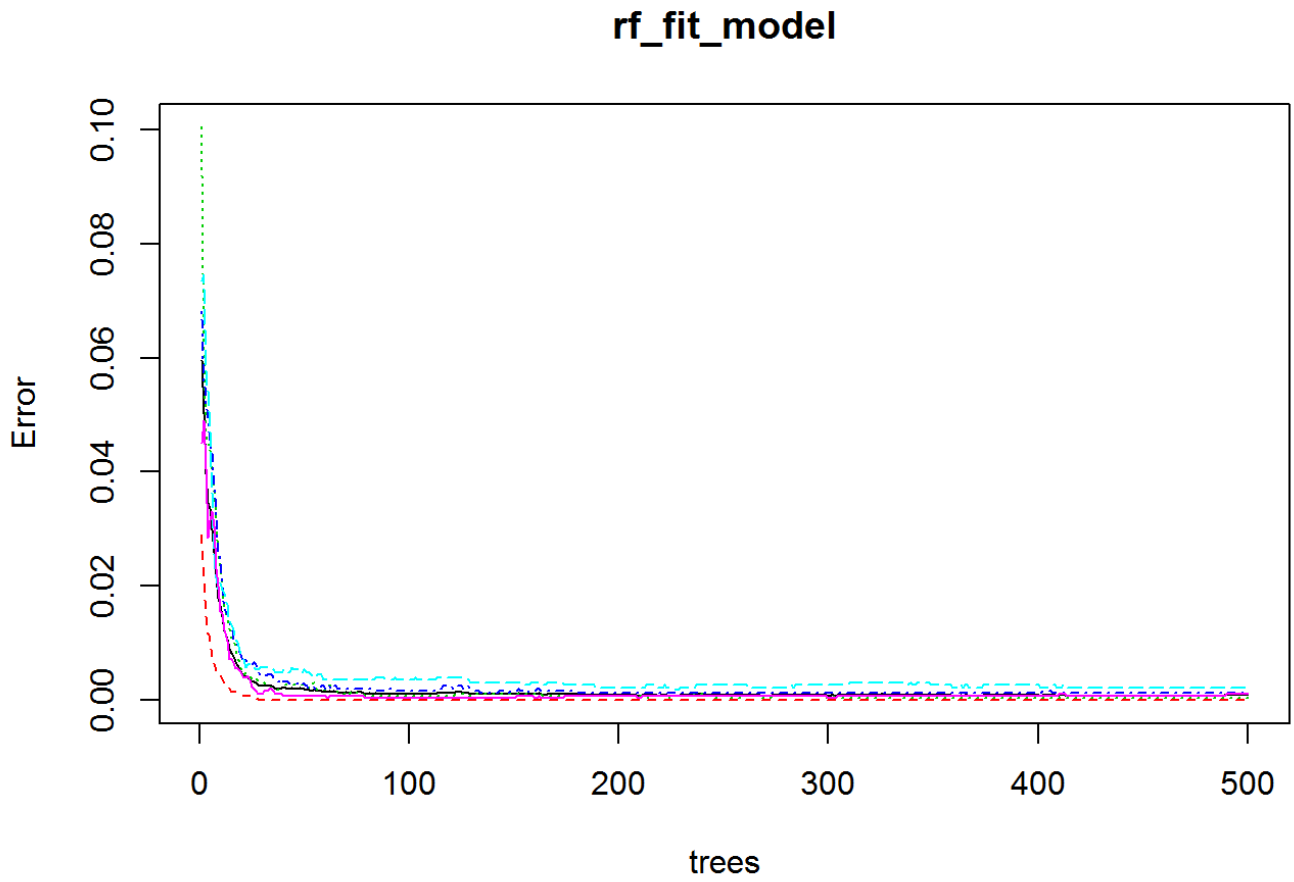
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    4
##          E    0    0    0    0 1078
##
## 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            1.0000   0.9991   0.9981   0.9979   0.9963
## Specificity            0.9998   0.9996   0.9996   0.9992   1.0000
## Pos Pred Value         0.9994   0.9982   0.9981   0.9959   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9996   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1635   0.1832
## Detection Prevalence   0.2846   0.1937   0.1743   0.1641   0.1832
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9986   0.9982
```

```
## Check the model accuracy
  accuracy<-c(as.numeric(predict(rf_fit_model,newdata=final_validation[,-ncol(fina
l_validation)])==final_validation$classe))
  accuracy<-sum(accuracy)*100/nrow(final_validation)
  accuracy
```

```
## [1] 99.84707
```

The model accuracy is ~ ***99.84%***

```
plot(rf_fit_model)
```



Final Prediction

This last step will use the above model to predict the outcome "classe" for the provided test sample of data.

```
predict_testing <- predict(rf_fit_model, newdata=testing_filter_missing[-1,])
```