# Practical Machine Learning Project

George Dai

5/31/2020

## Introduction

In this project assignment, we are asked to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner of the participants how they do the exercises.

## Data Downloading

First, let's load the libraries.

```
library(rpart)
library(rpart.plot)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Then, let's download the data.

```
trainURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
testURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
download.file(trainURL, 'pml-training.csv', method = 'curl')
download.file(testURL, 'pml-testing.csv', method = 'curl')
trainRaw <- read.csv('pml-training.csv')
testRaw <- read.csv('pml-testing.csv')
trainDim <- dim(trainRaw)
testDim <- dim(testRaw)
```

We have 19622 observations in the training set and 20 with a total of 160 variables.

## Data Processing

In this section, we are going to clean the dataset by removing observations with missing values and variables that are not used.

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
col.rm <- c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_wi
training.rm <- which(colnames(trainRaw) %in% col.rm)
trainRaw <- trainRaw[, -training.rm]
testRaw <- testRaw[, -training.rm]
```

## Data Slicing

Consider the testSet as a validation set, we would like to split the trainSet into a training and testing pair. We will split 75% into training and 25% into testing. This split can also serve to compute the out-of-sample error.

```
set.seed(333)
in_train <- createDataPartition(trainRaw$classe, p = 0.75, list = FALSE)
trainData <- trainRaw[in_train, ]
testData <- trainRaw[-in_train, ]
trainDim2 <- dim(trainData)
testDim2 <- dim(testData)
```

With this split, we have 14718 observation for the training and 4904 observations for testing. The original 20 observations are now reserved as the validation set.

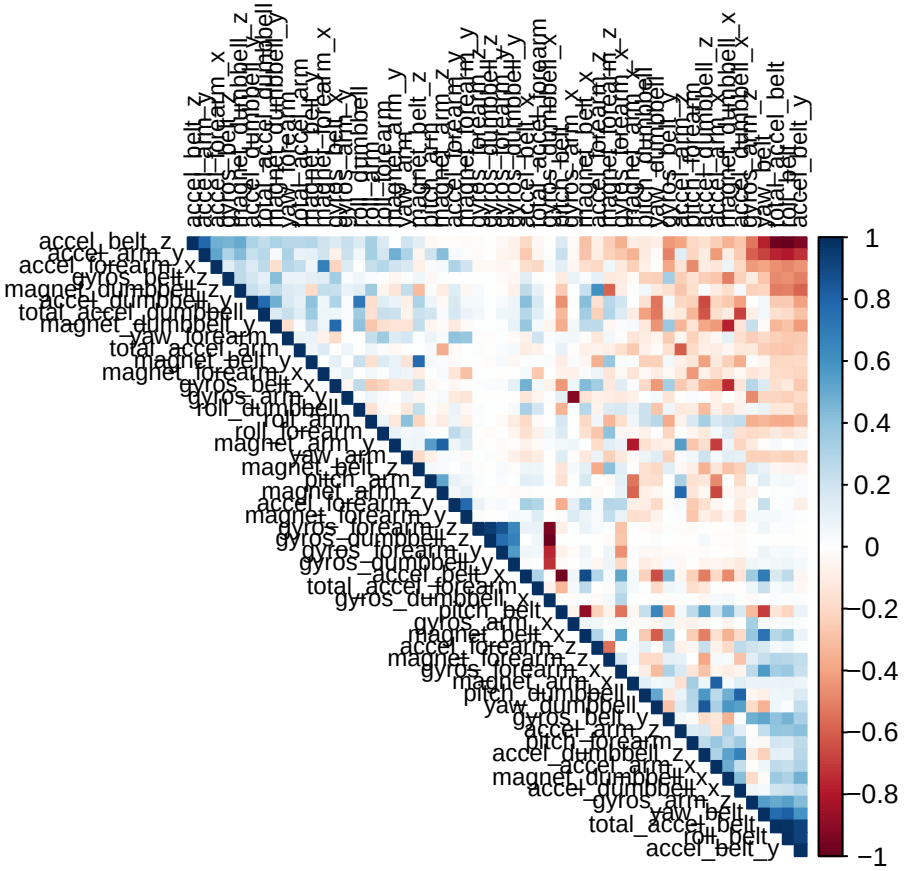
Next, we remove all variables with near-zero variance.

```
nzv <- nearZeroVar(trainData)
trainData <- trainData[, -nzv]
testData <- testData[, -nzv]
trainDim2 <- dim(trainData)
testDim2 <- dim(testData)
#testRaw <- testRaw[, -nzv]
```

With this, we are left with 53 variables.

Now, we can plot the correlation matrix among these variables.

```
cor_mat <- cor(trainData[,-53])
corrplot(cor_mat, order = 'FPC', method = 'color', type = 'upper', tl.cex = 0.75, tl.col = rgb(0,0,0))
```




We can find correlations that are considered as highly correlated if the correlation coefficient is greater than 0.8.

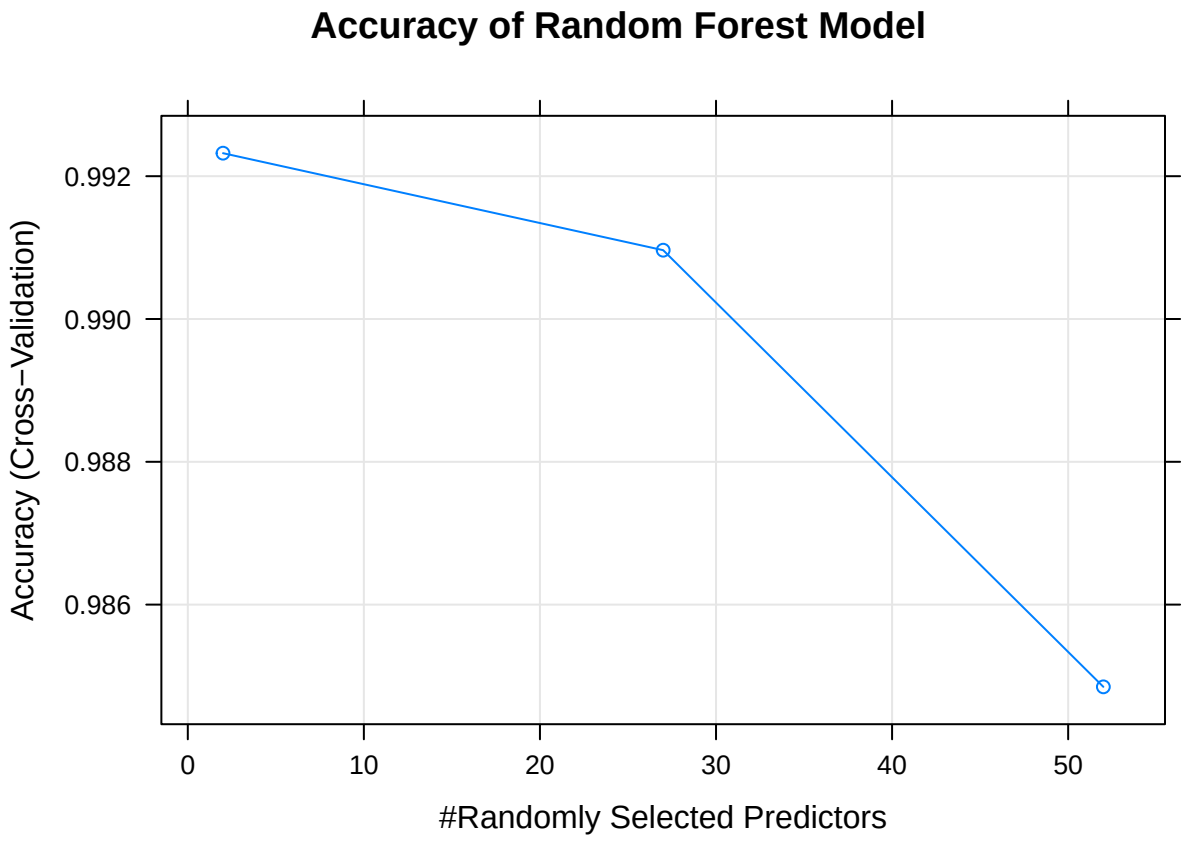
```
highCorr <- findCorrelation(cor_mat, cutoff = 0.8)
names(trainData)[highCorr]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"     "accel_dumbbell_z"
##  [5] "accel_belt_x"     "pitch_belt"       "accel_arm_x"      "accel_dumbbell_x"
##  [9] "magnet_arm_y"     "gyros_forearm_y"  "gyros_dumbbell_x" "gyros_dumbbell_z"
## [13] "gyros_arm_x"
```

## Training the Dataset

Using the **Random Forest** algorithm, we can train the trainData set as follows.

```
controlRF <- trainControl(method = 'cv', 5)
modelRF <- train(classe ~ ., method = 'rf', data = trainData, trControl = controlRF, verbose = FALSE)
plot(modelRF, main='Accuracy of Random Forest Model')
```




Next, we estimate the performance of the modelon the validation set.

```
predictRF <- predict(modelRF, testData)
confusionMatrix(testData$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    3    0    0    0
##          B    5  943    1    0    0
##          C    0    9  844    2    0
##          D    0    0   17  787    0
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9923
##                  95% CI : (0.9894, 0.9945)
##     No Information Rate : 0.2849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9902
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9874   0.9791   0.9962   1.0000
## Specificity            0.9991   0.9985   0.9973   0.9959   0.9998
## Pos Pred Value         0.9978   0.9937   0.9871   0.9789   0.9989
## Neg Pred Value         0.9986   0.9970   0.9956   0.9993   1.0000
## Prevalence             0.2849   0.1947   0.1758   0.1611   0.1835
## Detection Rate         0.2838   0.1923   0.1721   0.1605   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9978   0.9930   0.9882   0.9960   0.9999
```

```
accuracy <- postResample(predictRF, testData$classe)
oose <- 1 - as.numeric(confusionMatrix(testData$classe, predictRF)$overall[1])
accPer <- format(100*accuracy[1], digits = 3)
oosePer <- format(100*oose, digits = 2)
```

The accuracy for the test dataset is 99.2%. The out-of-sample error (OOSE) is 0.77%.

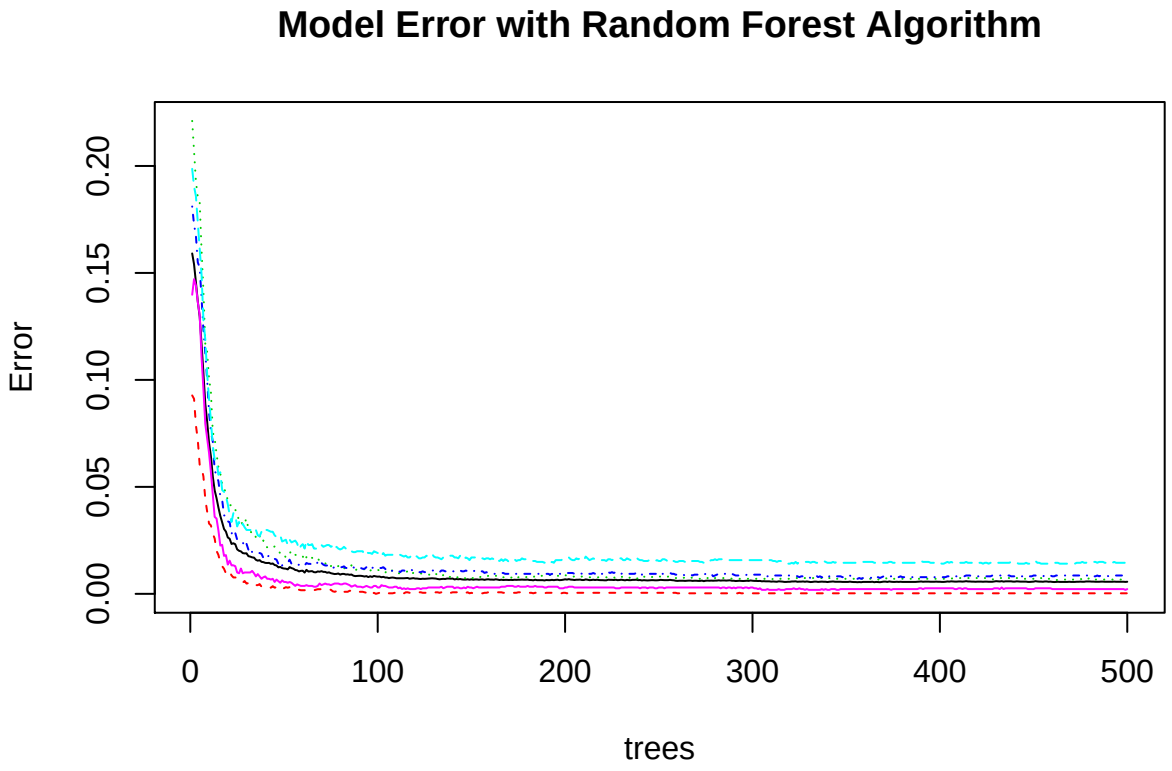
## Prediction

Finally, we use the 20 dataset as the validation set. The prediction can be run as follows.

```
modelRF$finalModel$classes
```

```
## [1] "A" "B" "C" "D" "E"
```

```
plot(modelRF$finalModel, main = 'Model Error with Random Forest Algorithm')
```




```
result <- predict(modelRF, newdata = testRaw)
#confusionMatrix(testRaw$classe, result)
print(result)
```

```
##  [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
```