Overview

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. The aim of this project is to predict the manner in which participants perform a barbell lift. The data comes from http://groupware.les.inf.puc-rio.br/har wherein 6 participants were asked to perform the same set of exercises correctly and incorrectly with accelerometers placed on the belt, forearm, arm, and dumbell.

For the purpose of this project, the following steps would be followed:

1. Data Preprocessing
2. Exploratory Analysis
3. Prediction Model Selection
4. Predicting Test Set Output

Data Preprocessing

First, we load the training and testing set from the online sources and then split the training set further into training and test sets.

```r
library(caret)
setwd("~/Projects/R/Coursera-Practical-Machine-Learning-Assignment-1/")
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(trainURL))
testing <- read.csv(url(testURL))

label <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[label, ]
test <- training[-label, ]
```

From among 160 variables present in the dataset, some variables have nearly zero variance whereas some contain a lot of NA terms which need to be excluded from the dataset. Moreover, other 5 variables used for identification can also be removed.

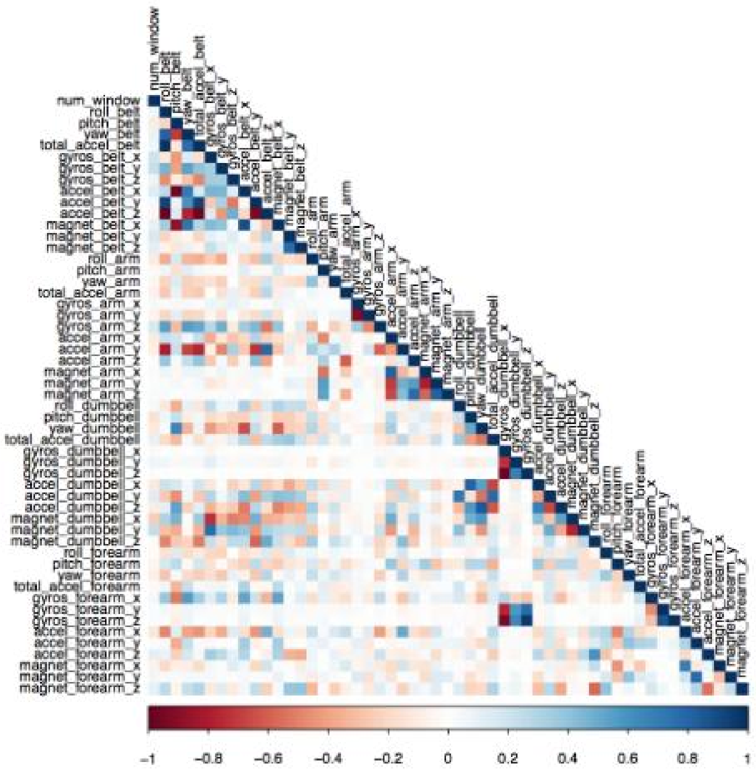
```r
NZV <- nearZeroVar(train)
train <- train[ ,-NZV]
test <- test[ ,-NZV]

label <- apply(train, 2, function(x) mean(is.na(x))) > 0.95
train <- train[, -which(label, label == FALSE)]
test <- test[, -which(label, label == FALSE)]

train <- train[ , -(1:5)]
test <- test[ , -(1:5)]
```

As a result of the preprocessing steps, we were able to reduce 160 variables to 54.

Exploratory Analysis

Now that we have cleaned the dataset off absolutely useless varibles, we shall look at the dependence of these variables on each other through a correlation plot.

```r
library(corrplot)
corrMat <- cor(train[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



In the plot above, darker gradient correspond to having high correlation. A Principal Component Analysis can be run to further reduce the correlated variables but we aren't doing that due to the number of correlations being quite few.

Prediction Model Selection

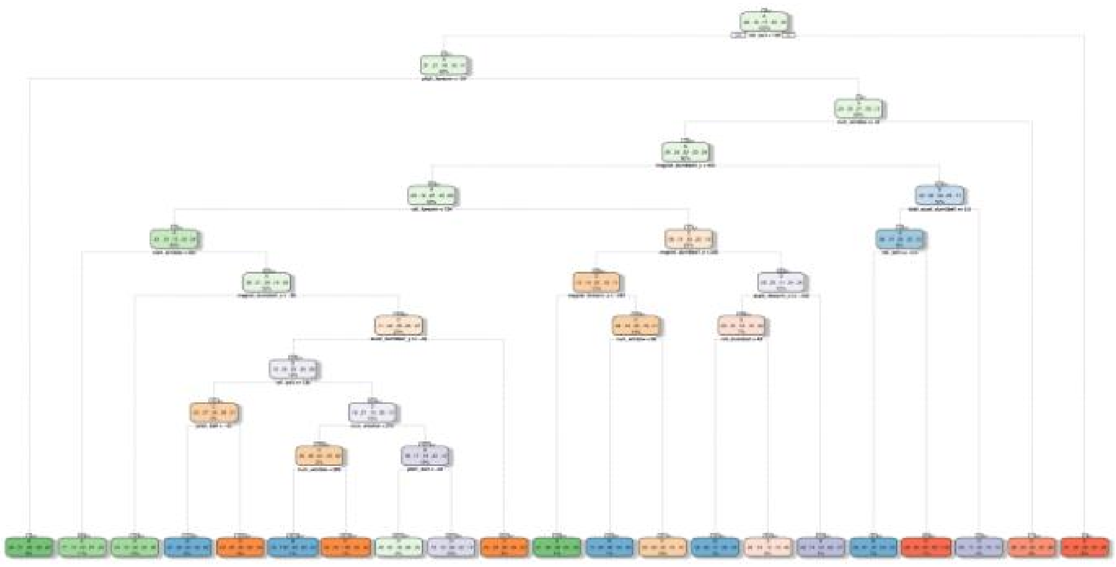
We will use 3 methods to model the training set and thereby choose the one having the best accuracy to predict the outcome variable in the testing set. The methods are Decision Tree, Random Forest and Generalized Boosted Model.

A confusion matrix plotted at the end of each model will help visualize the analysis better.

Decision Tree

```r
library(rpart)
library(rpart.plot)
library(rattle)
```

```r
set.seed(13908)
modelDT <- rpart(classe ~ ., data = train, method = "class")
fancyRpartPlot(modelDT)
```



Rattle 2017-Aug-16 01:03:52 Yash_Kumar_Singh

```r
predictDT <- predict(modelDT, test, type = "class")
confMatDT <- confusionMatrix(predictDT, test$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1505  233   44   80   29
##          B   39  609   36   21   25
##          C   21   76  818  143   88
##          D   86  145   51  612  131
##          E   23   76   77  108  809
##
## Overall Statistics
##
##                Accuracy : 0.7397
##                  95% CI : (0.7283, 0.7509)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6696
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8990   0.5347   0.7973   0.6349   0.7477
## Specificity            0.9083   0.9745   0.9325   0.9161   0.9409
## Pos Pred Value         0.7959   0.8342   0.7138   0.5971   0.7402
## Neg Pred Value         0.9577   0.8972   0.9561   0.9276   0.9430
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2557   0.1035   0.1390   0.1040   0.1375
## Detection Prevalence   0.3213   0.1240   0.1947   0.1742   0.1857
## Balanced Accuracy      0.9037   0.7546   0.8649   0.7755   0.8443
```

Random Forest

```
library(caret)
set.seed(13908)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
modelRF <- train(classe ~ ., data = train, method = "rf", trControl = control)
modelRF$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: 0.24%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    7 2645    5    1    0 0.0048908954
## C    0    4 2392    0    0 0.0016694491
## D    0    0    8 2243    1 0.0039964476
## E    0    0    0    5 2520 0.0019801980
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1130    1    0    0
##          C    0    3 1025    6    0
##          D    0    1    0  958    3
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9990   0.9938   0.9972
## Specificity            0.9988   0.9998   0.9981   0.9992   1.0000
## Pos Pred Value         0.9970   0.9991   0.9913   0.9958   1.0000
## Neg Pred Value         1.0000   0.9981   0.9998   0.9988   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1920   0.1742   0.1628   0.1833
## Detection Prevalence   0.2853   0.1922   0.1757   0.1635   0.1833
## Balanced Accuracy      0.9994   0.9959   0.9986   0.9965   0.9986
```

Generalized Boosted Model

```
library(caret)
set.seed(13908)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = train, trControl = control, method = "gbm", verbose = FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 44 had non-zero influence.
```

```
predictGBM <- predict(modelGBM, test)
confMatGBM <- confusionMatrix(predictGBM, test$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669   15    0    0    0
##          B    5 1110    9   10    9
##          C    0   13 1012    7    2
##          D    0    1    5  945   13
##          E    0    0    0    2 1058
##
## Overall Statistics
##
##                Accuracy : 0.9845
##                  95% CI : (0.981, 0.9875)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9804
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9745   0.9864   0.9803   0.9778
## Specificity            0.9964   0.9930   0.9955   0.9961   0.9996
## Pos Pred Value         0.9911   0.9711   0.9787   0.9803   0.9981
## Neg Pred Value         0.9988   0.9939   0.9971   0.9961   0.9950
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1886   0.1720   0.1606   0.1798
## Detection Prevalence   0.2862   0.1942   0.1757   0.1638   0.1801
## Balanced Accuracy      0.9967   0.9838   0.9909   0.9882   0.9887
```

As Random Forest offers the maximum accuracy of 99.75%, we will go with Random Forest Model to predict our test data class variable.

Predicting Test Set Output

```
predictRF <- predict(modelRF, testing)
predictRF
```

```
##  [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
```