# Prediction Assignment Writeup

## Background

In this project,I will 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.

### Data loading and processing

```
setwd("C:/Users/rr111836/Desktop/Studies/Coursera/Assignment 3")

library(readr)
training<-read.csv("pml-training (1).csv",na.strings = c("NA", "#DIV/0!",
""))
testing <- read.csv("pml-testing (1).csv",na.strings = c("NA", "#DIV/0!",
""))
```

### Loading Required package

```
library(knitr)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(rpart.plot)
library(rattle)

## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

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:rattle':
##
##     importance

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

```
library(corrplot)

## corrplot 0.84 loaded

set.seed(301)
```

## Removing columns that contains NA values and irrelevant variables and Partioning the training set into training and crossvalidation datasets

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

## [1] 13737   160
```

## remove variables with Nearly Zero Variance

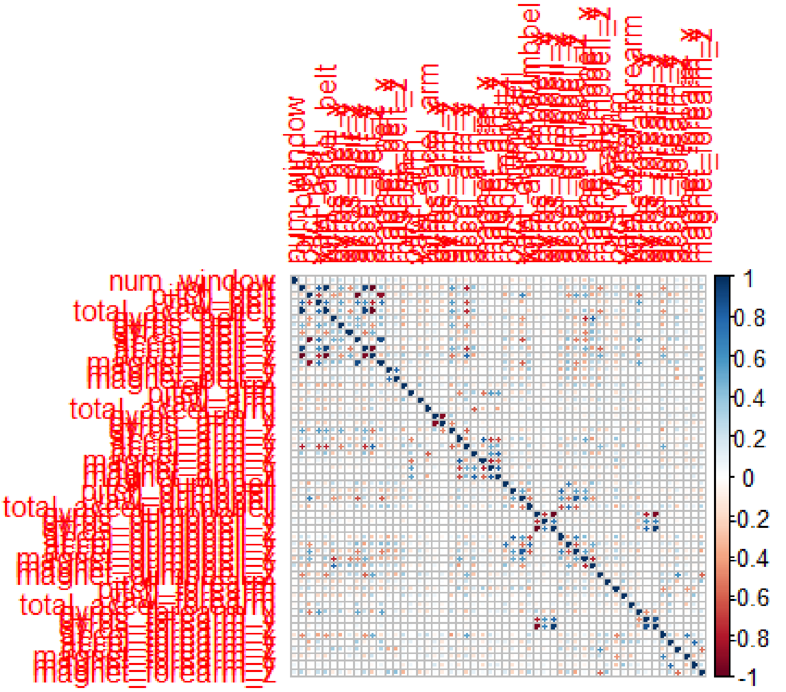
```
n0var <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -n0var]
TestSet  <- TestSet[, -n0var]
dim(TrainSet)

## [1] 13737   130

dim(TestSet)

## [1] 5885  130
```

###Remove Variables that are mostly NAs

```
AllNA <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet <- TestSet[, AllNA==FALSE]
dim(TrainSet)

## [1] 13737    59

dim(TestSet)

## [1] 5885   59

# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)

## [1] 13737    54
```

###check correlation among variables

```
M <- cor(TrainSet[, -54])
corrplot(M, method="circle")
```



## Random Forest method

```
# plot matrix results
# model fit
set.seed(3408)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
modFitRandForest$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: 0.27%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    7 2647    3    1    0 0.0041384500
## C    0    8 2388    0    0 0.0033388982
## D    0    0    9 2242    1 0.0044404973
## E    0    1    0    6 2518 0.0027722772
```

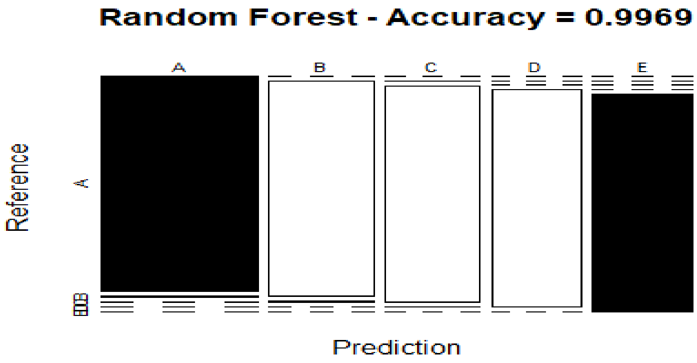
## Prediction

```
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   10    0    0    0
##          B    0 1128    6    0    0
##          C    0    1 1020    1    0
##          D    0    0    0  963    0
##          E    0    0    0    0 1082
## 
## Overall Statistics
## 
##                Accuracy : 0.9969
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9961
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9903   0.9942   0.9990   1.0000
## Specificity            0.9976   0.9987   0.9996   1.0000   1.0000
## Pos Pred Value         0.9941   0.9947   0.9980   1.0000   1.0000
## Neg Pred Value         1.0000   0.9977   0.9988   0.9998   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1917   0.1733   0.1636   0.1839
## Detection Prevalence   0.2862   0.1927   0.1737   0.1636   0.1839
## Balanced Accuracy      0.9988   0.9945   0.9969   0.9995   1.0000
```

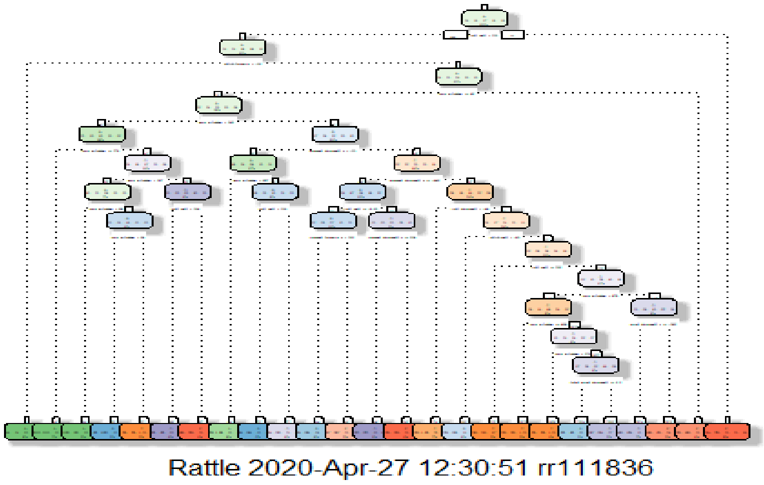
## Plot Matrix Results

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



## Decision Tree

```
# model fit
set.seed(3408)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

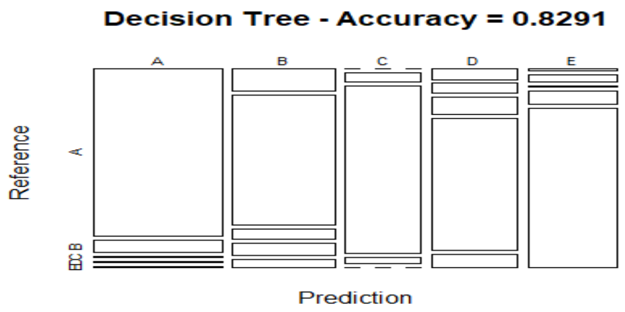


## Prediction on Test dataset

```
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, TestSet$classe)
confMatDecTree

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1441  107    2   15    5
##          B  156  880   73   80   56
##          C    0   48  848   29    0
##          D   64   58   98  761   72
##          E   13   46    5   79  949
## 
## Overall Statistics
## 
##                Accuracy : 0.8291          
##                  95% CI : (0.8192, 0.8386)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.7843          
## 
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8608   0.7726   0.8265   0.7894   0.8771
## Specificity            0.9694   0.9231   0.9842   0.9407   0.9702
## Pos Pred Value         0.9178   0.7068   0.9168   0.7227   0.8690
## Neg Pred Value         0.9460   0.9442   0.9641   0.9580   0.9723
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2449   0.1495   0.1441   0.1293   0.1613
## Detection Prevalence   0.2668   0.2116   0.1572   0.1789   0.1856
## Balanced Accuracy      0.9151   0.8479   0.9053   0.8650   0.9237
```

## Plot matrix results

```
plot(confMatDecTree$table, col = confMatDecTree$byClass, 
     main = paste("Decision Tree - Accuracy =",
                  round(confMatDecTree$overall['Accuracy'], 4)))
```



## Applying the selected Model to the Test Data

```
predictTEST <- predict(modFitRandForest, 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
```