STEP 0: LOAD THE LIBRARIES

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12
```

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

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

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

```
library(corrplot)
```

STEP 1: LOAD THE TRAINING AND TEST DATA

```
trainRaw = read.csv("./data/pml-training.csv")
testRaw = read.csv("./data/pml-testing.csv")
dim(trainRaw)
```

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

```
dim(testRaw)
```

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

STEP 2: UNDERSTAND THE PROBLEM: THE GOAL IS TO PREDICT THE MANNER IN WHICH THEY DID EXERCIES - “classe” variable

```
str(trainRaw$classe)
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

STEP 3: DATA CLEANING EXERCISE

```
trainRaw = trainRaw[, colSums(is.na(trainRaw)) == 0] # RETAIN COLUMNS WITHOUT NAs
testRaw = testRaw[, colSums(is.na(testRaw)) == 0] # RETAIN COLUMNS WITHOUT NAs
classe = trainRaw$classe
trainRaw = trainRaw[,-c(1,3,4,5,6,7)]
trainOnlyNum = trainRaw[, sapply(trainRaw, is.numeric)] #RETAIN NUMERIC COLUMNS
trainOnlyNum$classe = classe # THIS IS A FACTOR WE ARE TRYING TO PREDICT

testRaw = testRaw[,-c(1,3,4,5,6,7)]
testOnlyNum = testRaw[, sapply(testRaw, is.numeric)]
```

STEP 4: MODEL USING RANDOM FOREST - USE 70:30 FOR CROSS-VALIDATION

```
set.seed(1000) # For reproducibile purpose
inTrain = createDataPartition(trainOnlyNum$classe, p = 0.70, list = F)
trainData = trainOnlyNum[inTrain,]
testData = trainOnlyNum[-inTrain,]
controlRf <- trainControl(method = "cv", 5)
modelRf <-
    train(
        classe ~ .,
        data = trainData,
        method = "rf",
        trControl = controlRf,
        ntree = 250
    )
```

STEP 5: PREDICT USING THE TRAIN DATA

```
predictRf = predict(modelRf, testData)
```

STEP 6: FIND THE ACCURACY WITH OUT-SAMPLE DATA

```
confusionMatrix(testData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B    6 1132    1    0    0
##          C    0    7 1017    2    0
##          D    0    0   17  946    1
##          E    0    0    1    4 1077
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9929          
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.285           
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.991           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9912   0.9817   0.9937   0.9991
## Specificity            0.9993   0.9985   0.9981   0.9964   0.9990
## Pos Pred Value         0.9982   0.9939   0.9912   0.9813   0.9954
## Neg Pred Value         0.9986   0.9979   0.9961   0.9988   0.9998
## Prevalence             0.2850   0.1941   0.1760   0.1618   0.1832
## Detection Rate         0.2839   0.1924   0.1728   0.1607   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9949   0.9899   0.9950   0.9990
```

```
confusionMatrix(testData$classe, predictRf)$overall[1]
```

```
##  Accuracy 
## 0.9928632
```

```
accuracy = postResample(predictRf, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9928632 0.9909720
```

```
outOfSampleError =  1 - as.numeric(confusionMatrix(testData$classe, predictRf)$overall
[1])
outOfSampleError
```

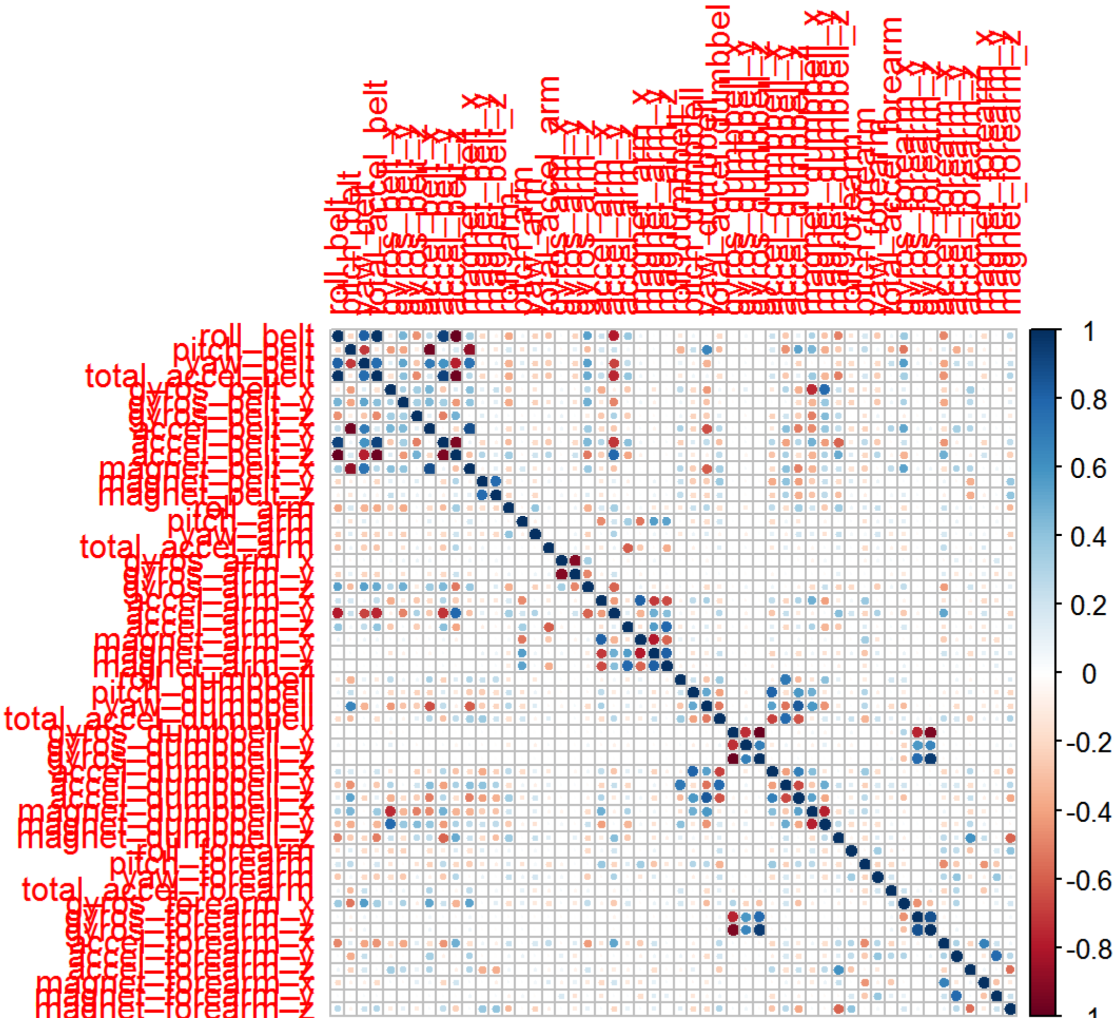
```
## [1] 0.007136788
```

```
result =    predict(modelRf, testOnlyNum[,-length(names(testOnlyNum))])
result
```

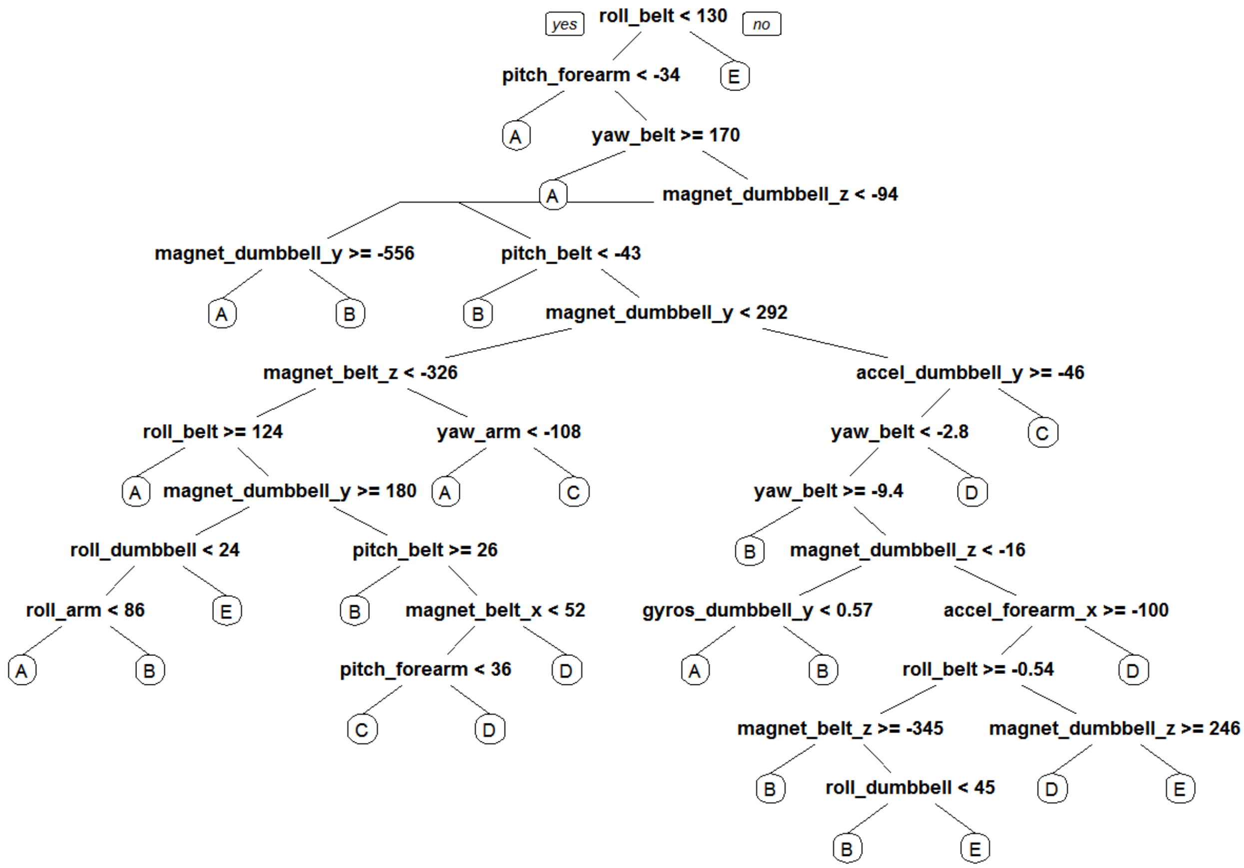
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

STEP 7: DESCRIPTIVE PLOT - CORRELATION BETWEEN PREDICTORS; TREE MODEL OUPUT

```
corrPlot = cor(trainData[,-length(names(trainData))])
corrplot(corrPlot, method = "circle")
```



```
treeModel = rpart(classe ~ ., data = trainData, method = "class")
prp(treeModel) # fast plot
```


yes
roll_belt < 130
no
pitch_forearm < -34
E
A
yaw_belt >= 170
A
magnet_dumbbell_z < -94
magnet_dumbbell_y >= -556
pitch_belt < -43
A
B
B
magnet_dumbbell_y < 292
magnet_belt_z < -326
accel_dumbbell_y >= -46
roll_belt >= 124
yaw_arm < -108
yaw_belt < -2.8
C
A
magnet_dumbbell_y >= 180
A
C
yaw_belt >= -9.4
D
roll_dumbbell < 24
pitch_belt >= 26
B
magnet_dumbbell_z < -16
roll_arm < 86
E
B
magnet_belt_x < 52
gyros_dumbbell_y < 0.57
accel_forearm_x >= -100
A
B
pitch_forearm < 36
D
A
B
roll_belt >= -0.54
D
C
D
magnet_belt_z >= -345
magnet_dumbbell_z >= 246
B
roll_dumbbell < 45
D
E
B
E