PRACTICAL MACHINE LEARNING ASSIGNMENT

~~~~~~

Yogesh Rampariya

4 September, 2019

Introduction

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, your 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).

Data Source

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading the Dataset

Download the data files from the Internet and load them into two data frames. We ended up with a training dataset and a 20 observations testing dataset that will be submitted to Coursera

```r
library(caret)
```

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

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

```r
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(ipred)
setwd("C:/Users/Prabhu/Documents/practical Machine Learning")
training <- read.csv("training.csv")
mytestdata <- read.csv("test.csv")
```

Cleaning Data

Identify all column having No variation which doesnot affect outcome and doesnot participate actively in deciding final outcome

```
trainnzv <- nearZeroVar(training)
trainnzv
```

```
##  [1]   6  12  13  14  15  16  17  20  23  26  51  52  53  54  55  56  57
## [18]  58  59  69  70  71  72  73  74  75  78  79  81  82  87  88  89  90
## [35]  91  92  95  98 101 125 126 127 128 129 130 131 133 134 136 137 139
## [52] 142 143 144 145 146 147 148 149 150
```

```
## Remove all unwanted column

newdata <- training[, - trainnzv]
ncol(newdata)
```

```
## [1] 100
```

Case 1 (Remove all NA from data)

```
any(is.na(newdata))
```

```
## [1] TRUE
```

```r
newdata1 <- complete.cases(newdata)
table(as.factor(newdata1))
```

```
##
## FALSE  TRUE
## 19216   406
```

```r
## Considering only active rows
newdata2 <- na.omit(newdata)

## Removing 1st and 2nd column as it doesnot affect outcome classe variable
newdata2 <- newdata2[, -c(1,2)]

nrow(newdata2)
```

```
## [1] 406
```

```r
ncol(newdata2)
```

```
## [1] 98
```

Case 1 Data Partition and Model Building

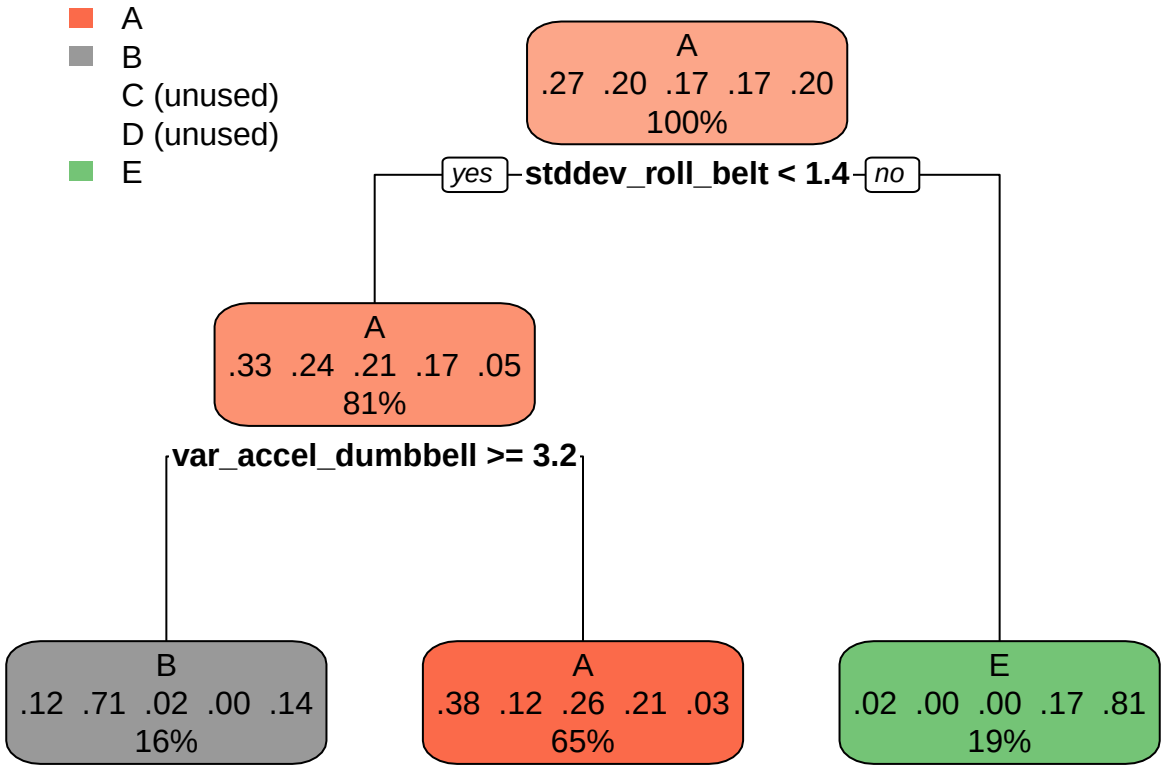
```r
datapart <- createDataPartition(newdata2$classe, p = 0.75, list = FALSE)
traindata <- newdata2[datapart,]
testdata <- newdata2[- datapart,]
## Decision Tree Model
model1 <- train(classe ~ ., data = traindata, method="rpart")
model1
```

```
## CART
##
## 307 samples
##  97 predictor
##   5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 307, 307, 307, 307, 307, 307, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.09333333  0.4870811  0.34513041
##   0.12888889  0.4203864  0.24228498
##   0.20444444  0.3077768  0.06881024
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.09333333.
```

```r
prediction1 <- predict(model1, testdata)
confusionMatrix(prediction1, testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 25  6 15 15  4
##          B  2 12  2  1  1
##          C  0  0  0  0  0
##          D  0  0  0  0  0
##          E  0  1  0  1 14
##
## Overall Statistics
##
##                Accuracy : 0.5152
##                  95% CI : (0.4125, 0.6168)
##     No Information Rate : 0.2727
##     P-Value [Acc > NIR] : 2.813e-07
##
##                   Kappa : 0.3578
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9259   0.6316   0.0000   0.0000   0.7368
## Specificity            0.4444   0.9250   1.0000   1.0000   0.9750
## Pos Pred Value         0.3846   0.6667      NaN      NaN   0.8750
## Neg Pred Value         0.9412   0.9136   0.8283   0.8283   0.9398
## Prevalence             0.2727   0.1919   0.1717   0.1717   0.1919
## Detection Rate         0.2525   0.1212   0.0000   0.0000   0.1414
## Detection Prevalence   0.6566   0.1818   0.0000   0.0000   0.1616
## Balanced Accuracy      0.6852   0.7783   0.5000   0.5000   0.8559
```

```r
rpart.plot(model1$finalModel, roundint=FALSE)
```




```
## Very low accuracy

## Bagging Model

bagdata <- bagging(classe ~ ., data = traindata)
prediction2 <- predict(bagdata, testdata)
confusionMatrix(prediction2, testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 27  1  0  0  0
##          B  0 18  2  0  0
##          C  0  0 14  1  0
##          D  0  0  1 13  3
##          E  0  0  0  3 16
##
## Overall Statistics
##
##                Accuracy : 0.8889
##                  95% CI : (0.8099, 0.9432)
##     No Information Rate : 0.2727
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8597
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9474   0.8235   0.7647   0.8421
## Specificity            0.9861   0.9750   0.9878   0.9512   0.9625
## Pos Pred Value         0.9643   0.9000   0.9333   0.7647   0.8421
## Neg Pred Value         1.0000   0.9873   0.9643   0.9512   0.9625
## Prevalence             0.2727   0.1919   0.1717   0.1717   0.1919
## Detection Rate         0.2727   0.1818   0.1414   0.1313   0.1616
## Detection Prevalence   0.2828   0.2020   0.1515   0.1717   0.1919
## Balanced Accuracy      0.9931   0.9612   0.9057   0.8580   0.9023
```

```r
## Quite Improvement but still not upto mark

## Random Forrest Model
model <- train(classe ~ ., data = traindata[, -1], method = "rf", ntree = 100)
prediction <- predict(model, testdata[, -1])
confusionMatrix(prediction, testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 24  3  1  3  1
##          B  1 14  2  1  2
##          C  0  0 14  1  0
##          D  1  0  0 11  2
##          E  1  2  0  1 14
##
## Overall Statistics
##
##                Accuracy : 0.7778
##                  95% CI : (0.6831, 0.8552)
##     No Information Rate : 0.2727
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7179
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8889   0.7368   0.8235   0.6471   0.7368
## Specificity            0.8889   0.9250   0.9878   0.9634   0.9500
## Pos Pred Value         0.7500   0.7000   0.9333   0.7857   0.7778
## Neg Pred Value         0.9552   0.9367   0.9643   0.9294   0.9383
## Prevalence             0.2727   0.1919   0.1717   0.1717   0.1919
## Detection Rate         0.2424   0.1414   0.1414   0.1111   0.1414
## Detection Prevalence   0.3232   0.2020   0.1515   0.1414   0.1818
## Balanced Accuracy      0.8889   0.8309   0.9057   0.8052   0.8434
```

```
## Unexpected result shows there is flow in analysis and need to consider more columns
```

Case 2 Eliminating Unnecessary column

```
removena <- sapply(newdata, function(x) mean(is.na(x))) > 0.95
newnadata <- newdata[,removena == FALSE]

ncol(newnadata)
```

```
## [1] 59
```

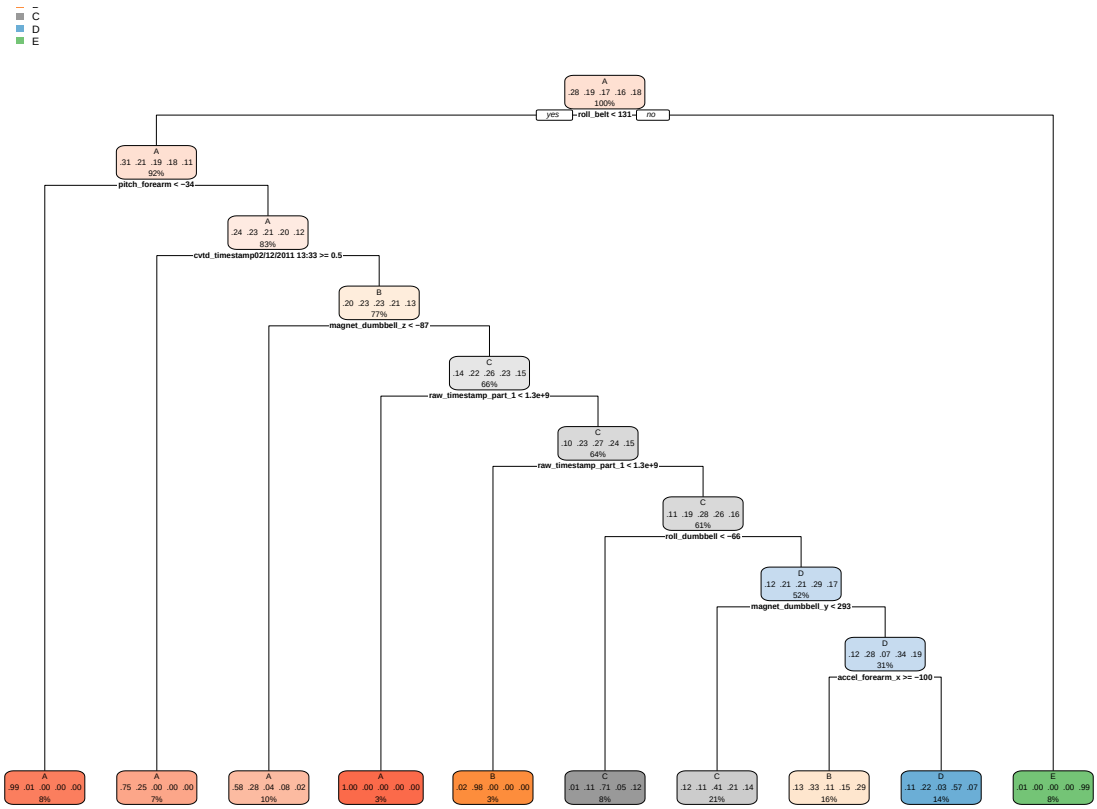
```
datana <- createDataPartition(newnadata$classe, p = 0.75, list = FALSE)
newtraindata <- newnadata[datana, ]
newtestdata <- newnadata[- datana,]

## Decision tree Model
## Eliminating first column as Name doesnot play any role in decisiding Outcome
model3 <- train(classe ~ ., data = newtraindata[,-1], method = "rpart")
prediction3 <- predict(model3, newtestdata[, -1])
confusionMatrix(prediction3, newtestdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1042  259   36   34   15
##          B  105  395   98  122  246
##          C  160  164  702  261  189
##          D   83  131   19  387   59
##          E    5    0    0    0  392
##
## Overall Statistics
##
##                Accuracy : 0.595
##                  95% CI : (0.5811, 0.6088)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4883
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7470  0.41623   0.8211  0.48134  0.43507
## Specificity            0.9020  0.85563   0.8088  0.92878  0.99875
## Pos Pred Value         0.7518  0.40890   0.4756  0.56996  0.98741
## Neg Pred Value         0.8997  0.85932   0.9554  0.90130  0.88706
## Prevalence             0.2845  0.19352   0.1743  0.16395  0.18373
```

```
## Detection Rate         0.2125  0.08055   0.1431  0.07892  0.07993
## Detection Prevalence   0.2826  0.19698   0.3010  0.13846  0.08095
## Balanced Accuracy      0.8245  0.63593   0.8149  0.70506  0.71691
```

```r
## Not Quite Accurate
rpart.plot(model3$finalModel, roundint=FALSE)
```




```r
## Bagging Model

bagdata1 <- bagging(classe ~ ., data = newtraindata[, -1])
prediction4 <- predict(bagdata1, newtestdata[,-1])
confusionMatrix(prediction4, newtestdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  849    4    0
##          D    0    0    4  797    3
##          E    0    0    2    3  898
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9967
##                  95% CI : (0.9947, 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   1.0000   0.9930   0.9913   0.9967
## Specificity            1.0000   1.0000   0.9990   0.9983   0.9988
## Pos Pred Value         1.0000   1.0000   0.9953   0.9913   0.9945
## Neg Pred Value         1.0000   1.0000   0.9985   0.9983   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1731   0.1625   0.1831
## Detection Prevalence   0.2845   0.1935   0.1739   0.1639   0.1841
## Balanced Accuracy      1.0000   1.0000   0.9960   0.9948   0.9977
```

```
## Random Forrest Model
model5 <- train(classe ~ ., data = newtraindata[, -1], method = "rf", ntree = 100)
prediction5 <- predict(model5, newtestdata[, -1])
confusionMatrix(prediction5, newtestdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    0    0
##          D    0    0    0  804    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9992, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
prediction5
```

```
##    [1] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##   [35] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##   [69] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [103] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [137] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [171] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [205] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [239] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [273] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [307] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [341] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [375] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [409] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [443] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [477] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [511] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [545] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [579] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [613] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [647] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [681] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [715] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [749] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [783] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [817] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [851] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [885] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [919] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [953] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
##  [987] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1021] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1055] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1089] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1123] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1157] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1191] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1225] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1259] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1293] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1327] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1361] A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
## [1395] A B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1429] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1463] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1497] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1531] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1565] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
```

```
## [1599] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1633] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1667] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1701] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1735] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1769] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1803] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1837] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1871] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1905] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1939] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [1973] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2007] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2041] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2075] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2109] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2143] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2177] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2211] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2245] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2279] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
## [2313] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B C C
## [2347] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2381] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2415] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2449] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2483] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2517] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2551] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2585] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2619] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2653] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2687] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2721] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2755] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2789] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2823] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2857] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2891] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2925] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2959] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [2993] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [3027] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [3061] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [3095] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [3129] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [3163] C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
## [3197] C C C D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3231] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3265] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3299] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3333] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3367] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3401] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
```

```
## [3435] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3469] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3503] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3537] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3571] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3605] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3639] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3673] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3707] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3741] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3775] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3809] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3843] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3877] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3911] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3945] D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
## [3979] D D D D D D D D D D D D D D D D D D D D D D D D E E E E E E E E
## [4013] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4047] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4081] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4115] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4149] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4183] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4217] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4251] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4285] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4319] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4353] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4387] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4421] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4455] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4489] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4523] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4557] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4591] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4625] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4659] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4693] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4727] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4761] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4795] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4829] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4863] E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E E
## [4897] E E E E E E E E
## Levels: A B C D E
```

```
# Determination of Course Project Prediction Quiz Portion

# Steps include refining test data and applying best model ontain from train data
# Random Forest Model


ncol(mytestdata)
```

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

```
colnames(mytestdata)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
```

```
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "problem_id"
```

```
testnzv <- nearZeroVar(mytestdata)
newtest.data <- mytestdata[, - testnzv]
remove.na <- sapply(newtest.data, function(x) mean(is.na(x))) > 0.95
newtest.na <- newtest.data[,remove.na == FALSE]
str(newtest.na)
```

```
## 'data.frame':    20 obs. of  59 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name            : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2 3 ...
##  $ raw_timestamp_part_1: int  1323095002 1322673067 1322673075 1322832789 1322489635 1322673149 1322
##  $ raw_timestamp_part_2: int  868349 778725 342967 560311 814776 510661 766645 54671 916313 384285 .
##  $ cvtd_timestamp       : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10 10 1 6 11 11 10 3 2 ...
##  $ num_window           : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt            : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt           : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt             : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...
##  $ total_accel_belt     : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ gyros_belt_x         : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y         : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z         : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x         : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y         : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z         : int  -179 39 49 -156 27 38 35 42 32 -158 ...
```

```
##  $ magnet_belt_x       : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y       : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z       : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm            : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm           : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm             : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm     : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ gyros_arm_x         : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y         : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z         : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x         : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y         : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z         : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x        : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y        : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z        : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ roll_dumbbell       : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell      : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell        : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ total_accel_dumbbell: int  9 31 29 18 4 29 29 29 3 2 ...
##  $ gyros_dumbbell_x    : num  0.64 0.34 0.39 0.1 0.29 -0.59 0.34 0.37 0.03 0.42 ...
##  $ gyros_dumbbell_y    : num  0.06 0.05 0.14 -0.02 -0.47 0.8 0.16 0.14 -0.21 0.51 ...
##  $ gyros_dumbbell_z    : num  -0.61 -0.71 -0.34 0.05 -0.46 1.1 -0.23 -0.39 -0.21 -0.03 ...
##  $ accel_dumbbell_x    : int  21 -153 -141 -51 -18 -138 -145 -140 0 -7 ...
##  $ accel_dumbbell_y    : int  -15 155 155 72 -30 166 150 159 25 -20 ...
##  $ accel_dumbbell_z    : int  81 -205 -196 -148 -5 -186 -190 -191 9 7 ...
##  $ magnet_dumbbell_x   : int  523 -502 -506 -576 -424 -543 -484 -515 -519 -531 ...
##  $ magnet_dumbbell_y   : int  -528 388 349 238 252 262 354 350 348 321 ...
##  $ magnet_dumbbell_z   : int  -56 -36 41 53 312 96 97 53 -32 -164 ...
##  $ roll_forearm        : num  141 109 131 0 -176 150 155 -161 15.5 13.2 ...
##  $ pitch_forearm       : num  49.3 -17.6 -32.6 0 -2.16 1.46 34.5 43.6 -63.5 19.4 ...
##  $ yaw_forearm         : num  156 106 93 0 -47.9 89.7 152 -89.5 -139 -105 ...
##  $ total_accel_forearm : int  33 39 34 43 24 43 32 47 36 24 ...
##  $ gyros_forearm_x     : num  0.74 1.12 0.18 1.38 -0.75 -0.88 -0.53 0.63 0.03 0.02 ...
##  $ gyros_forearm_y     : num  -3.34 -2.78 -0.79 0.69 3.1 4.26 1.8 -0.74 0.02 0.13 ...
##  $ gyros_forearm_z     : num  -0.59 -0.18 0.28 1.8 0.8 1.35 0.75 0.49 -0.02 -0.07 ...
##  $ accel_forearm_x     : int  -110 212 154 -92 131 230 -192 -151 195 -212 ...
##  $ accel_forearm_y     : int  267 297 271 406 -93 322 170 -331 204 98 ...
##  $ accel_forearm_z     : int  -149 -118 -129 -39 172 -144 -175 -282 -217 -7 ...
##  $ magnet_forearm_x    : int  -714 -237 -51 -233 375 -300 -678 -109 0 -403 ...
##  $ magnet_forearm_y    : int  419 791 698 783 -787 800 284 -619 652 723 ...
##  $ magnet_forearm_z    : int  617 873 783 521 91 884 585 -32 469 512 ...
##  $ problem_id          : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
newtest.na$problem_id
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
prediction6 <- predict(model5, newtest.na)
prediction6
```

```
##  [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
```

Conclusion

- Based on result obtain Random forrest Model provides best possible predicting model for Case 2.
- As we can we from the result, the random forest algorithem far outperforms the decision tree in terms of accuracy. We are getting 99.99% in sample accuracy, while the decision tree gives us only nearly 50% in sample accuracy