**Practical Machine Learning Course Project**

Y. Zheng

3/9/2020

PART 1: Overview and Background

This is the write up of the course project for Practical Machine Learning. RStudio version 3.6.3 was used for this project.

In this project, data collected from Jawbone Up, Nike FuelBand, and Fitbit was used to track 6 participants’ activities. These 6 participates performed barbell lifts either correctly or incorrectly in 5 different ways. The task is to predict the manner they did the exercise. The variable “classe” in the training data set will be the dependent variable. Other variables in the training data set could be used as independent variables for this project. The tasks of this project include:

1. Describe the predictive model, cross validation, and the expected out of sample error.
2. Use the model to predict 20 different test cases.

PART 2: Code and Results

2-1: library used in this project

> library(caret)

> library(AppliedPredictiveModeling)

> library(rattle)

> library(rpart)

> library(rpart.plot)

> library(RColorBrewer)

> library(randomForest)

> library(ggplot2)

> library(corrplot)

> library(gbm)

2-2: Explore and Clean the Data Sets

There are 19622 records and 160 variables in the training data. The validation data set contains 20 records and 160 variables.

> training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header=TRUE)

>

> dim(training)

[1] 19622 160

>

> valid\_in <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header=TRUE)

>

> dim(valid\_in)

[1] 20 160

>

> str(training)

'data.frame': 19622 obs. of 160 variables:

$ X : int 1 2 3 4 5 6 7 8 9 10 ...

$ user\_name : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...

$ raw\_timestamp\_part\_1 : int 1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...

$ raw\_timestamp\_part\_2 : int 788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...

$ cvtd\_timestamp : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...

$ new\_window : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

$ num\_window : int 11 11 11 12 12 12 12 12 12 12 ...

$ roll\_belt : num 1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...

$ pitch\_belt : num 8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...

$ yaw\_belt : num -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...

$ total\_accel\_belt : int 3 3 3 3 3 3 3 3 3 3 ...

$ kurtosis\_roll\_belt : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...

$ kurtosis\_picth\_belt : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...

$ kurtosis\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_roll\_belt : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_roll\_belt.1 : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

$ max\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ max\_picth\_belt : int NA NA NA NA NA NA NA NA NA NA ...

$ max\_yaw\_belt : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

$ min\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ min\_pitch\_belt : int NA NA NA NA NA NA NA NA NA NA ...

$ min\_yaw\_belt : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

$ amplitude\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ amplitude\_pitch\_belt : int NA NA NA NA NA NA NA NA NA NA ...

$ amplitude\_yaw\_belt : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...

$ var\_total\_accel\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ avg\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ stddev\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ var\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ avg\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ stddev\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ var\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ avg\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ stddev\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ var\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...

$ gyros\_belt\_x : num 0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...

$ gyros\_belt\_y : num 0 0 0 0 0.02 0 0 0 0 0 ...

$ gyros\_belt\_z : num -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...

$ accel\_belt\_x : int -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...

$ accel\_belt\_y : int 4 4 5 3 2 4 3 4 2 4 ...

$ accel\_belt\_z : int 22 22 23 21 24 21 21 21 24 22 ...

$ magnet\_belt\_x : int -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...

$ magnet\_belt\_y : int 599 608 600 604 600 603 599 603 602 609 ...

$ magnet\_belt\_z : int -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...

$ roll\_arm : num -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...

$ pitch\_arm : num 22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...

$ yaw\_arm : num -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...

$ total\_accel\_arm : int 34 34 34 34 34 34 34 34 34 34 ...

$ var\_accel\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ avg\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ stddev\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ var\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ avg\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ stddev\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ var\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ avg\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ stddev\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ var\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ gyros\_arm\_x : num 0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...

$ gyros\_arm\_y : num 0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...

$ gyros\_arm\_z : num -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...

$ accel\_arm\_x : int -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...

$ accel\_arm\_y : int 109 110 110 111 111 111 111 111 109 110 ...

$ accel\_arm\_z : int -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...

$ magnet\_arm\_x : int -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...

$ magnet\_arm\_y : int 337 337 344 344 337 342 336 338 341 334 ...

$ magnet\_arm\_z : int 516 513 513 512 506 513 509 510 518 516 ...

$ kurtosis\_roll\_arm : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...

$ kurtosis\_picth\_arm : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...

$ kurtosis\_yaw\_arm : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_roll\_arm : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_pitch\_arm : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_yaw\_arm : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...

$ max\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ max\_picth\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ max\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...

$ min\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ min\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ min\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...

$ amplitude\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ amplitude\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...

$ amplitude\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...

$ roll\_dumbbell : num 13.1 13.1 12.9 13.4 13.4 ...

$ pitch\_dumbbell : num -70.5 -70.6 -70.3 -70.4 -70.4 ...

$ yaw\_dumbbell : num -84.9 -84.7 -85.1 -84.9 -84.9 ...

$ kurtosis\_roll\_dumbbell : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...

$ kurtosis\_picth\_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...

$ kurtosis\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_roll\_dumbbell : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_pitch\_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...

$ skewness\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

$ max\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...

$ max\_picth\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...

$ max\_yaw\_dumbbell : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

$ min\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...

$ min\_pitch\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...

$ min\_yaw\_dumbbell : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

$ amplitude\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...

[list output truncated]

> trainData<-training[,colSums(is.na(training))==0]

> validData<-valid\_in[,colSums(is.na(testing))==0]

>

> dim(trainData)

[1] 19622 93

> dim(validData)

[1] 20 60

> trainData<-trainData[,-c(1:7)]

> validData<-validData[,-c(1:7)]

> dim(trainData)

[1] 19622 86

> dim(validData)

[1] 20 53

> set.seed(1234)

>

2-3: Generate Training and Testing Data Sets

The original training data set was split into 2 data sets: 70% training data and 30% test data. The split data sets were used to build prediction models. The correlation graph is shown in Figure 1.

> inTrain<-createDataPartition(trainData$classe,p=0.7,list=FALSE)

> trainData<-trainData[inTrain,]

> testData<-trainData[-inTrain,]

> dim(trainData)

[1] 13737 86

> dim(testData)

[1] 4123 86

>

> NZV<-nearZeroVar(trainData)

> trainData<-trainData[,-NZV]

> testData<-testData[,-NZV]

> dim(trainData)

[1] 13737 53

> dim(testData)

[1] 4123 53

> cor\_mat<-cor(trainData[,-53])

> corrplot(cor\_mat,order="FPC", method="color", type="lower",t1.cex=0.8,t1.col=rgb(0,0,0))

The following scripts were applied to find out correlations.

> highlyCorrelated=findCorrelation(cor\_mat,cutoff=0.75)

> names(trainData)[highlyCorrelated]

[1] "accel\_belt\_z" "roll\_belt"

[3] "accel\_belt\_y" "total\_accel\_belt"

[5] "accel\_dumbbell\_z" "accel\_belt\_x"

[7] "pitch\_belt" "magnet\_dumbbell\_x"

[9] "accel\_dumbbell\_y" "magnet\_dumbbell\_y"

[11] "accel\_dumbbell\_x" "accel\_arm\_x"

[13] "accel\_arm\_z" "magnet\_arm\_y"

[15] "magnet\_belt\_z" "accel\_forearm\_y"

[17] "gyros\_forearm\_y" "gyros\_dumbbell\_x"

[19] "gyros\_dumbbell\_z" "gyros\_arm\_x"

2-4: Build Prediction Models

2-4-1: Using Classification Tree to Predict

Figure 2 demonstrated the classification tree. And Confusion Matrix suggested (Figure 3) the performance regarding to each variable. According to Figure 3, the accuracy ratte of the model is about 0.76 and the out of sample error is about 0.23.

> set.seed(12345)

There were 12 warnings (use warnings() to see them)

> decisionTreeMod1<-rpart(classe~.,data=trainData,method="class")

> fancyRpartPlot(decisionTreeMod1)

Warning message:

labs do not fit even at cex 0.15, there may be some overplotting

> predictTreeMod1<-predict(decisionTreeMod1,testData,type="class")

> cmtree<-confusionMatrix(predictTreeMod1,testData$classe)

> cmtree

Confusion Matrix and Statistics

Reference

Prediction A B C D E

A 1067 105 9 24 9

B 40 502 59 63 77

C 28 90 611 116 86

D 11 49 41 423 41

E 19 41 18 46 548

Overall Statistics

Accuracy : 0.7642

95% CI : (0.751, 0.7771)

No Information Rate : 0.2826

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7015

Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

Class: A Class: B Class: C

Sensitivity 0.9159 0.6379 0.8279

Specificity 0.9503 0.9284 0.9055

Pos Pred Value 0.8789 0.6775 0.6563

Neg Pred Value 0.9663 0.9157 0.9602

Prevalence 0.2826 0.1909 0.1790

Detection Rate 0.2588 0.1218 0.1482

Detection Prevalence 0.2944 0.1797 0.2258

Balanced Accuracy 0.9331 0.7831 0.8667

Class: D Class: E

Sensitivity 0.6295 0.7201

Specificity 0.9589 0.9631

Pos Pred Value 0.7487 0.8155

Neg Pred Value 0.9300 0.9383

Prevalence 0.1630 0.1846

Detection Rate 0.1026 0.1329

Detection Prevalence 0.1370 0.1630

Balanced Accuracy 0.7942 0.8416

> plot(cmtree$table, col = cmtree$byClass,

+ main = paste("Decision Tree - Accuracy =", round(cmtree$overall['Accuracy'], 4)))

2-4-2: Using Random Forest to Predict

As shown in Figure 4, the accuracy of the random forest is as high as 1, and the out of sample error is 0. In order to test whether there is overfitting, Figure 5 was used. According to Figure 4, there is no overfitting for random forest prediction model.

> controlRF<-trainControl(method="cv",number=3,verboseIter=FALSE)

> modRF1<-train(classe~.,data=trainData,method="rf",trControl=controlRF)

> modRF1$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: 0.7%

Confusion matrix:

A B C D E class.error

A 3902 3 0 0 1 0.001024066

B 19 2634 5 0 0 0.009029345

C 0 17 2369 10 0 0.011268781

D 0 1 26 2224 1 0.012433393

E 0 2 5 6 2512 0.005148515

> predictRF1<-predict(modRF1,newdata=testData)

> cmrf<-confusionMatrix(predictRF1,testData$classe)

> cmrf

Confusion Matrix and Statistics

Reference

Prediction A B C D E

A 1165 0 0 0 0

B 0 787 0 0 0

C 0 0 738 0 0

D 0 0 0 672 0

E 0 0 0 0 761

Overall Statistics

Accuracy : 1

95% CI : (0.9991, 1)

No Information Rate : 0.2826

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 1.000 1.000 1.0000

Specificity 1.0000 1.0000 1.000 1.000 1.0000

Pos Pred Value 1.0000 1.0000 1.000 1.000 1.0000

Neg Pred Value 1.0000 1.0000 1.000 1.000 1.0000

Prevalence 0.2826 0.1909 0.179 0.163 0.1846

Detection Rate 0.2826 0.1909 0.179 0.163 0.1846

Detection Prevalence 0.2826 0.1909 0.179 0.163 0.1846

Balanced Accuracy 1.0000 1.0000 1.000 1.000 1.0000

>

> plot(modRF1)

>

> plot(cmrf$table, col = cmrf$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round(cmrf$overall['Accuracy'], 4)))

|  |
| --- |
| 2-4-3: Using Generalized Boosted Regression Models to Predict  According to the results shown below, the generalized boosted regression models has accuracy of about 0.97, and do the out of sample error is about 0.03.  > set.seed(12345)  > controlGBM<-trainControl(method="repeatedcv", number=5, repeats=1)  > modGBM<-train(classe~.,data=trainData,method="gbm",trControl=controlGBM,verbose=FALSE)  > modGBM$finalModel  A gradient boosted model with multinomial loss function.  150 iterations were performed.  There were 52 predictors of which 52 had non-zero influence.  >  > print(modGBM)  Stochastic Gradient Boosting  13737 samples  52 predictor  5 classes: 'A', 'B', 'C', 'D', 'E'  No pre-processing  Resampling: Cross-Validated (5 fold, repeated 1 times)  Summary of sample sizes: 10990, 10990, 10989, 10991, 10988  Resampling results across tuning parameters:  interaction.depth n.trees Accuracy Kappa  1 50 0.7521285 0.6858434  1 100 0.8227397 0.7756753  1 150 0.8522224 0.8130469  2 50 0.8564452 0.8181267  2 100 0.9059465 0.8809760  2 150 0.9301168 0.9115592  3 50 0.8969931 0.8695557  3 100 0.9392159 0.9230740  3 150 0.9587251 0.9477728  Tuning parameter 'shrinkage' was held constant at a value of 0.1  Tuning parameter 'n.minobsinnode' was held constant at a value of 10  Accuracy was used to select the optimal model using the largest value.  The final values used for the model were n.trees = 150, interaction.depth =  3, shrinkage = 0.1 and n.minobsinnode = 10.  >  > predictGBM<-predict(modGBM,newdata=testData)  > cmGBM<-confusionMatrix(predictGBM,testData$classe)  > cmGBM  Confusion Matrix and Statistics  Reference  Prediction A B C D E  A 1155 20 0 0 1  B 9 754 17 5 6  C 1 12 713 16 3  D 0 1 6 647 8  E 0 0 2 4 743  Overall Statistics    Accuracy : 0.9731  95% CI : (0.9677, 0.9778)  No Information Rate : 0.2826  P-Value [Acc > NIR] : < 2.2e-16    Kappa : 0.966    Mcnemar's Test P-Value : NA  Statistics by Class:  Class: A Class: B Class: C Class: D Class: E  Sensitivity 0.9914 0.9581 0.9661 0.9628 0.9763  Specificity 0.9929 0.9889 0.9905 0.9957 0.9982  Pos Pred Value 0.9821 0.9532 0.9570 0.9773 0.9920  Neg Pred Value 0.9966 0.9901 0.9926 0.9928 0.9947  Prevalence 0.2826 0.1909 0.1790 0.1630 0.1846  Detection Rate 0.2801 0.1829 0.1729 0.1569 0.1802  Detection Prevalence 0.2852 0.1919 0.1807 0.1606 0.1817  Balanced Accuracy 0.9922 0.9735 0.9783 0.9792 0.9873  2-4-4: Conclusion  In conclusion, random forest is the best predictive model for this project. The answers for the quiz questions are shown below.  > Results<-predict(modRF1,newdata=validData)  > Results  [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 |
|  |
| |  | | --- | | > | |

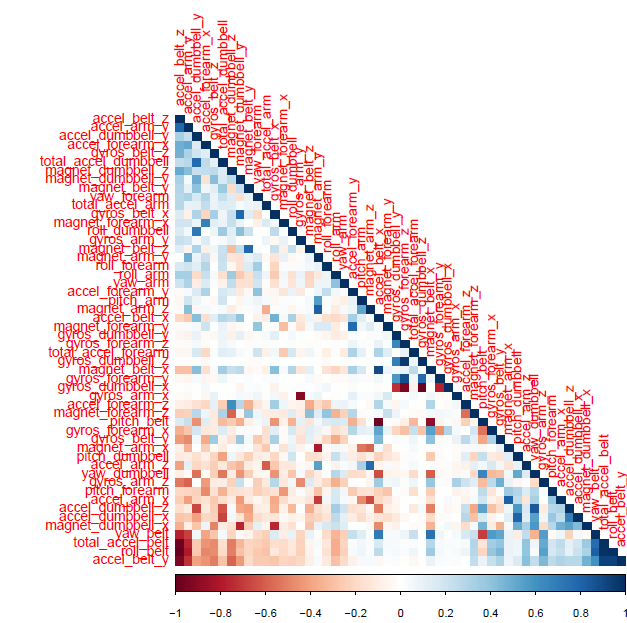


Figure 1: Correlation Graph (the correlated variables are shown in darker color in the graph)

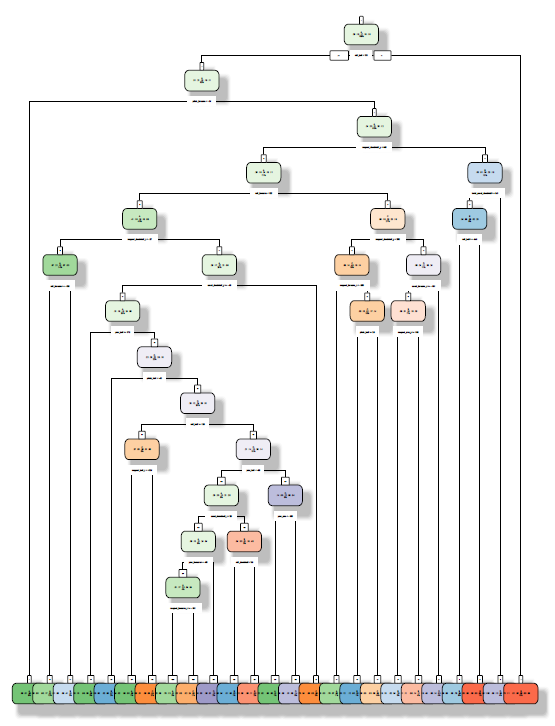


Figure 2: Classification Tree

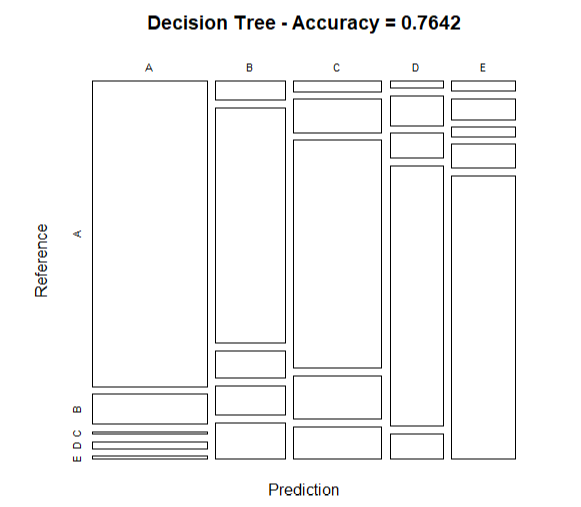


Figure 3: Decision Tree using Test Data

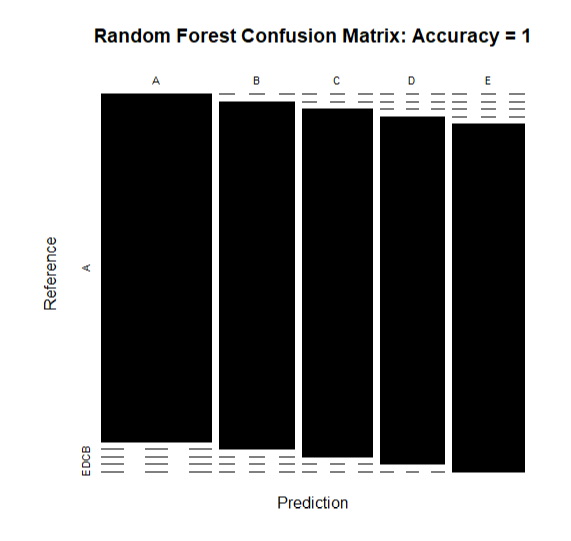


Figure 4: Random Forest Confusion Matrix

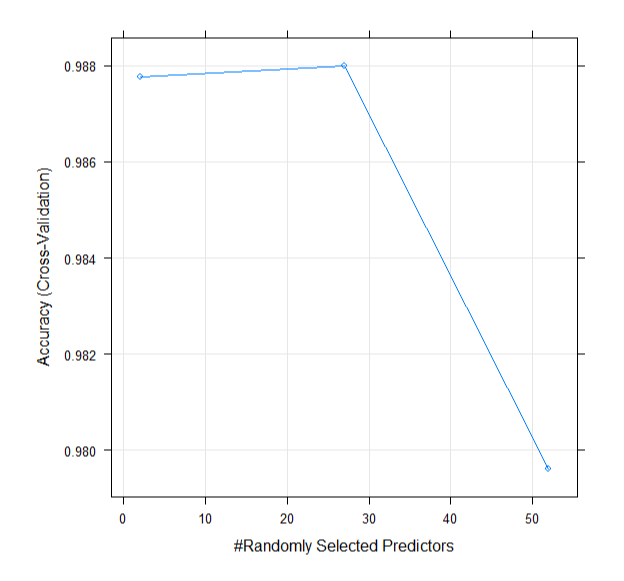


Figure 5: Detecting Overfitting for Random Forest Model

PART 3: References:

The related information for this project is shown in the following website: <http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

Note:

The enlarged graphs of Figure 1 and 2 will be provided when requested.