Practical Machine Learning - Course Project

Saurabh Bhalerao

August 4, 2017

Prediction Assignment Writeup

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit 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, we will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participant They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The five ways are exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Only Class A corresponds to correct performance. The goal of this project is to predict the manner in which they did the exercise, i.e., Class A to E. 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 Processing

Import the Data

First,Uploading R libraries.

```
library(caret)
library(rpart)
library(rpart.plot)
library(lattice)
library(ggplot2)
library(rattle)
library(randomForest)
library(corrplot)
```

Data Loading & Cleaning

```r
#set URL for Download
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.
csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v"

#download the data set
training <- read.csv(url(trainUrl), na.strings = c("NA", ""))
testing <- read.csv(url(testUrl), na.strings = c("NA", ""))

#Partition training data into two data sets: Train (60%) & Test (40%)
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)

## [1] 11776   160

## [1] 7846  160
```

Cleaning Data

```r
# Cleaning NearZeroVariance Variables
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]
myTesting <- myTesting[,nzv$nzv==FALSE]
dim(myTraining);dim(myTesting)

## [1] 11776   121

## [1] 7846  121

# Remove the columns with more NA values
AllNA    <- sapply(myTraining, function(x) mean(is.na(x))) > 0.95
myTraining <- myTraining[, AllNA==FALSE]
myTesting  <- myTesting[, AllNA==FALSE]
dim(myTraining);dim(myTesting)

## [1] 11776    59

## [1] 7846   59

# Remove first 5 columns
myTraining <- myTraining[, -(1:5)]
myTesting  <- myTesting[, -(1:5)]
dim(myTraining);dim(myTesting)

## [1] 11776    54

## [1] 7846   54
```

Prediction Model

Three methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below.

1) Random Forest

```
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=myTraining, 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: 0.3%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    7 2268    3    1    0 0.0048266784
## C    0    5 2048    1    0 0.0029211295
## D    0    0   10 1919    1 0.0056994819
## E    0    1    0    5 2159 0.0027713626
```

Prediction using above model & check of accuracy

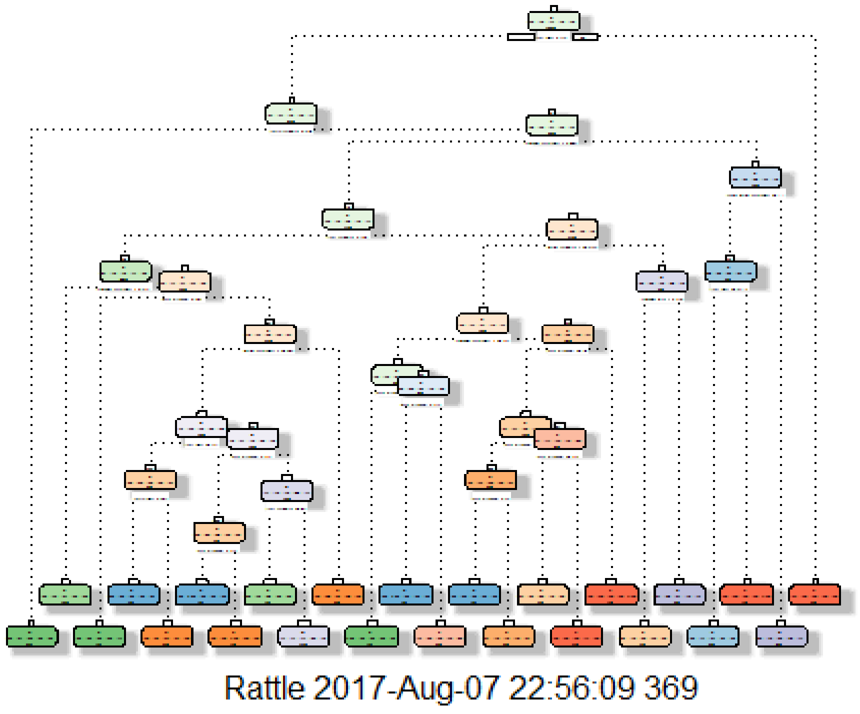
```
predictRandForest <- predict(modFitRandForest,myTesting)
confMatRandForest <- confusionMatrix(predictRandForest, myTesting$classe)
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    4    0    0    0
##          B    0 1513    5    0    0
##          C    0    1 1363    6    0
##          D    0    0    0 1280    1
##          E    1    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.9977
##                  95% CI : (0.9964, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9971
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9967   0.9963   0.9953   0.9993
## Specificity            0.9993   0.9992   0.9989   0.9998   0.9998
## Pos Pred Value         0.9982   0.9967   0.9949   0.9992   0.9993
## Neg Pred Value         0.9998   0.9992   0.9992   0.9991   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1928   0.1737   0.1631   0.1837
## Detection Prevalence   0.2849   0.1935   0.1746   0.1633   0.1838
## Balanced Accuracy      0.9994   0.9980   0.9976   0.9976   0.9996
```

2) Decision Trees

```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitDecTree)

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



Prediction on test data and checking accuracy of model

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

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1966  309   21  130   91
##          B   43  776   43   29   46
##          C   17  196 1170  134   29
##          D  166  220  127  915  210
##          E   40   17    7   78 1066
##
## Overall Statistics
##
##                Accuracy : 0.7511
##                  95% CI : (0.7414, 0.7606)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6844
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8808   0.5112   0.8553   0.7115   0.7393
## Specificity            0.9019   0.9746   0.9420   0.8898   0.9778
## Pos Pred Value         0.7811   0.8282   0.7568   0.5586   0.8825
## Neg Pred Value         0.9501   0.8926   0.9686   0.9402   0.9434
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2506   0.0989   0.1491   0.1166   0.1359
## Detection Prevalence   0.3208   0.1194   0.1970   0.2088   0.1540
## Balanced Accuracy      0.8913   0.7429   0.8986   0.8006   0.8585
```

3) Generalized Boosted Method

```
# model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=myTraining, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)

## Loading required package: gbm

## Loading required package: survival

##
## Attaching package: 'survival'
## The following object is masked from 'package:caret':
## 
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

## Loading required package: plyr

modFitGBM$finalModel

## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

Prediction on test data and checking accuracy of model

```
predictGBM <- predict(modFitGBM, newdata=myTesting)
confMatGBM <- confusionMatrix(predictGBM, myTesting$classe)
confMatGBM

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    7    0    0    0
##          B    5 1496   12    6    2
##          C    0   15 1352   15    0
##          D    1    0    3 1261    7
##          E    0    0    1    4 1433
## 
## Overall Statistics
## 
##                Accuracy : 0.9901
##                  95% CI : (0.9876, 0.9921)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9874
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9973   0.9855   0.9883   0.9806   0.9938
## Specificity            0.9988   0.9960   0.9954   0.9983   0.9992
## Pos Pred Value         0.9969   0.9836   0.9783   0.9914   0.9965
## Neg Pred Value         0.9989   0.9965   0.9975   0.9962   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2837   0.1907   0.1723   0.1607   0.1826
## Detection Prevalence   0.2846   0.1939   0.1761   0.1621   0.1833
## Balanced Accuracy      0.9980   0.9908   0.9918   0.9894   0.9965
```

Conclusion : Accuracy of Random Forest =0.9996 Accuracy of Decision Trees = 0.8083 Accuracy of Generalized Boosted Method = 0.9883 Random forest has given the best accuracy.

Predicting on Testing set

```
(predict(modFitRandForest, testing))

##  [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
```