PRACTICAL MACHINE LEARNING COURSE PROJECT

Zulkarnain Abu Hassan

January 14, 2016

1.SYNOPSIS

The goal of this project is to predict the manner in which they did the exercise based on the Training Dataset & Testing Dataset given

The report should describe:

- "how you built your model"
- "how you used cross validation"
- "what you think the expected out of sample error is"
- "why you made the choices you did"

Ultimately, the prediction model is to be run on the test data to predict the outcome of 20 different test cases.

In the aforementioned study, six participants participated in a dumbell lifting exercise five different ways. The five ways, as described in the study, were "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). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes."

By processing data gathered from accelerometers on the belt, forearm, arm, and dumbell of the participants in a machine learning algorithm, the question is can the appropriate activity quality (class A-E) be predicted?

2.Loading the appropriate packages

```
library(caret)

## Warning: package 'caret' was built under R version 3.2.3

## Loading required package: lattice
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.3
```

```r
library(ggplot2)
library(gridExtra)

## Warning: package 'gridExtra' was built under R version 3.2.3

library(grid)
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.3

## 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(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.2.3

library(rattle)

## Warning: package 'rattle' was built under R version 3.2.3

## Rattle: A free graphical interface for data mining with R.
## Version 4.0.5 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

## Warning: package 'RColorBrewer' was built under R version 3.2.2
```

3.Getting and loading the data

```r
set.seed(12345)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

4.Partioning the training set into two

Partioning Training data set into two data sets, 60% for myTraining, 40% for myTesting:

```
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)

## [1] 11776   160

## [1] 7846  160
```

5. Cleaning the data

a. Remove NearZeroVariance variables

```
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]

nzv<- nearZeroVar(myTesting,saveMetrics=TRUE)
myTesting <- myTesting[,nzv$nzv==FALSE]
```

b. Remove the first column of the myTraining data set

```
myTraining <- myTraining[c(-1)]
```

c. Clean variables with more than 60% NA

```
trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
    if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==
1)  {
                trainingV3 <- trainingV3[ , -j]
            }
        }
    }
}

# Set back to the original variable name
myTraining <- trainingV3
rm(trainingV3)
```

d. Transform the myTesting and testing data sets

```
clean1 <- colnames(myTraining)
# remove the classe column
clean2 <- colnames(myTraining[, -58])
# allow only variables in myTesting that are also in myTraining
myTesting <- myTesting[clean1]
# allow only variables in testing that are also in myTraining
testing <- testing[clean2]

dim(myTesting)
```

```
## [1] 7846   58

dim(testing)

## [1] 20 57
```

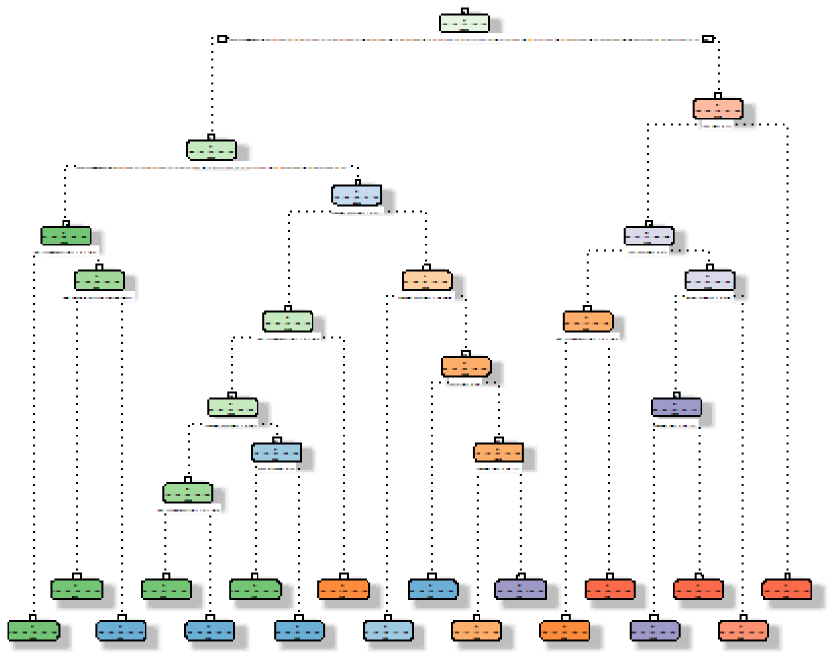
e. Coerce the data into the same type

```
for (i in 1:length(testing) ) {
    for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}

# To get the same class between testing and myTraining
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

6. Prediction Analysis using

a. Decision Trees

```
set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```

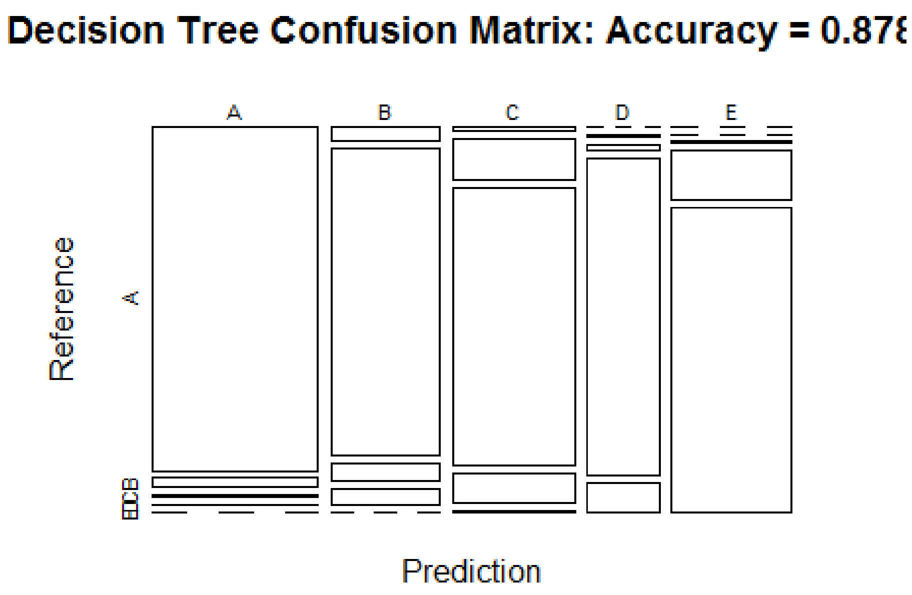


Rattle 2016-Jan-14 13:26:08 S11397

```r
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

```r
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree
Confusion Matrix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```

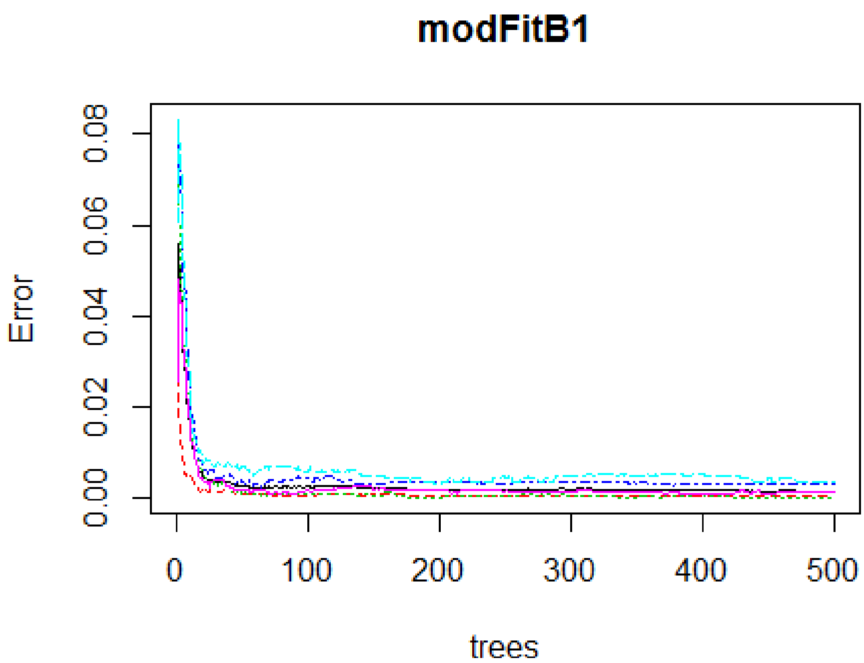


b. Random Forests

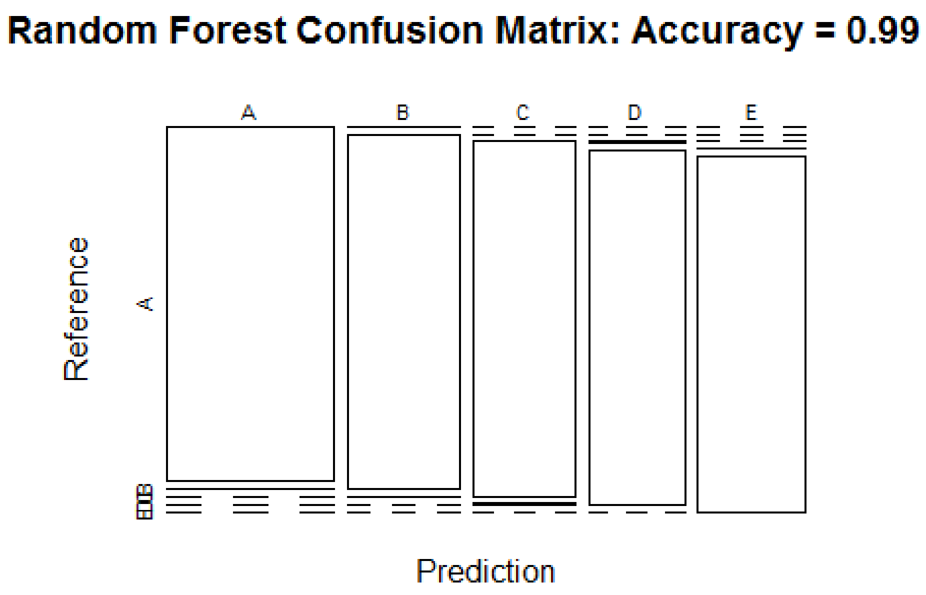
```
set.seed(12345)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    1    0    0
##          C    0    0 1366    3    0
##          D    0    0    1 1282    0
##          E    0    0    0    1 1442
## 
## Overall Statistics
## 
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9985   0.9969   1.0000
## Specificity            0.9996   0.9997   0.9995   0.9998   0.9998
## Pos Pred Value         0.9991   0.9987   0.9978   0.9992   0.9993
## Neg Pred Value         0.9998   0.9997   0.9997   0.9994   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1741   0.1634   0.1838
## Detection Prevalence   0.2846   0.1935   0.1745   0.1635   0.1839
## Balanced Accuracy      0.9996   0.9992   0.9990   0.9984   0.9999
```

```r
plot(modFitB1)
```




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



7. Predicting Results on the Test Data

Random Forests gave an Accuracy in the myTesting dataset of 99.89%, which was more accurate that what I got from the Decision Trees. The expected out-of-sample error is 100-99.89 = 0.11%.