Practical ML Course Project

Karthik Chawala

September 24, 2017

Project Summary

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.

Data:

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

Goal:

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Getting & Splitting the Data

Firstly load the required R packages, get and split the data for traning and testing the models.

```
# loading packages
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
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:rattle':
##
##     importance

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

```
library(knitr)

set.seed(123)

# get the data
trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training.data <- read.csv(url(trainURL), na.strings = c("NA","#DIV/0!",""))
testing.data <- read.csv(url(testURL), na.strings = c("NA","#DIV/0!",""))

## split the data
inTrain <- createDataPartition(training.data$classe, p = 0.7, list = FALSE)
my.Training <- training.data[inTrain, ]
my.Testing <- training.data[-inTrain, ]

dim(my.Training)
```

```
## [1] 13737   160
```

```
dim(my.Testing)
```

```
## [1] 5885  160
```

Preparing the Data

Secondly we will remove near zero variance variables and clean the training variables.

```
# removing near zero variance variables
nzv <- nearZeroVar(my.Training, saveMetrics = TRUE)
my.Training <- my.Training[,nzv$nzv == FALSE]

nzv <- nearZeroVar(my.Testing, saveMetrics = TRUE)
my.Testing <- my.Testing[, nzv$nzv == FALSE]

my.Training <- my.Training[c(-1)]

# clean the variables
training.temp <- my.Training
for(i in 1:length(my.Training)) {
  if(sum(is.na(my.Training[, i])) / nrow(my.Training) >= .7) {
    for(j in 1:length(training.temp)) {
```

```
      if(length(grep(names(my.Training[i]), names(training.temp)[j])) == 1){
        training.temp <- training.temp[, -j]
      }
    }
  }
}

my.Training <- training.temp
```

We will now prepare the training and testing data.

```
clean.data1 <- colnames(my.Training)
clean.data2 <- colnames(my.Training[ , -58])

my.Testing <- my.Testing[clean.data1]
testing.data <- testing.data[clean.data2]

dim(my.Testing)
```

```
## [1] 5885   58
```

```
dim(testing.data)
```

```
## [1] 20 57
```

```
# coerce the data
for (i in 1:length(testing.data)) {
  for(j in 1: length(my.Training)) {
    if(length(grep(names(my.Training[i]), names(testing.data)[j])) ==1) {
      class(testing.data[j]) <- class(my.Training[i])
    }
  }
}

testing.data <- rbind(my.Training[2, -58], testing.data)
testing.data <- testing.data[-1,]
```

Predicting with Random Forests

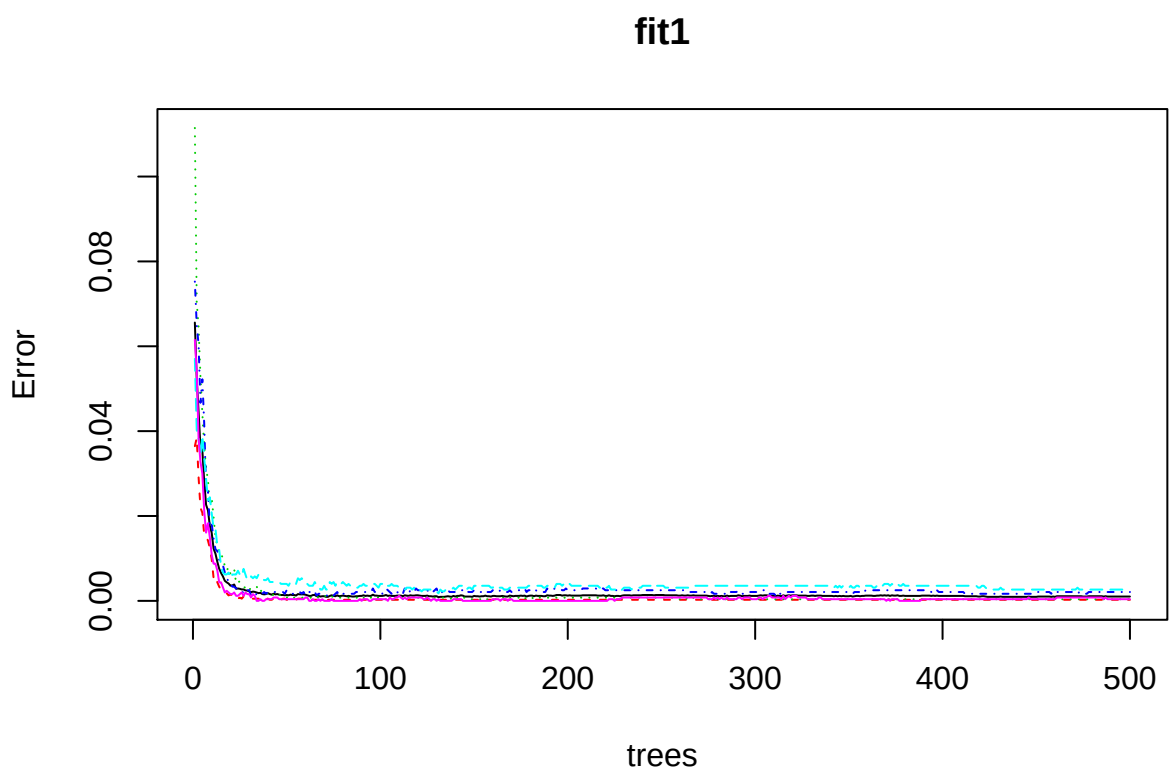
```
set.seed(123)
fit1 <- randomForest(classe ~ ., my.Training)
prediction1 <- predict(fit1, my.Testing, type = "class")

confusionMatrixRF <- confusionMatrix(prediction1, my.Testing$classe)
confusionMatrixRF
```

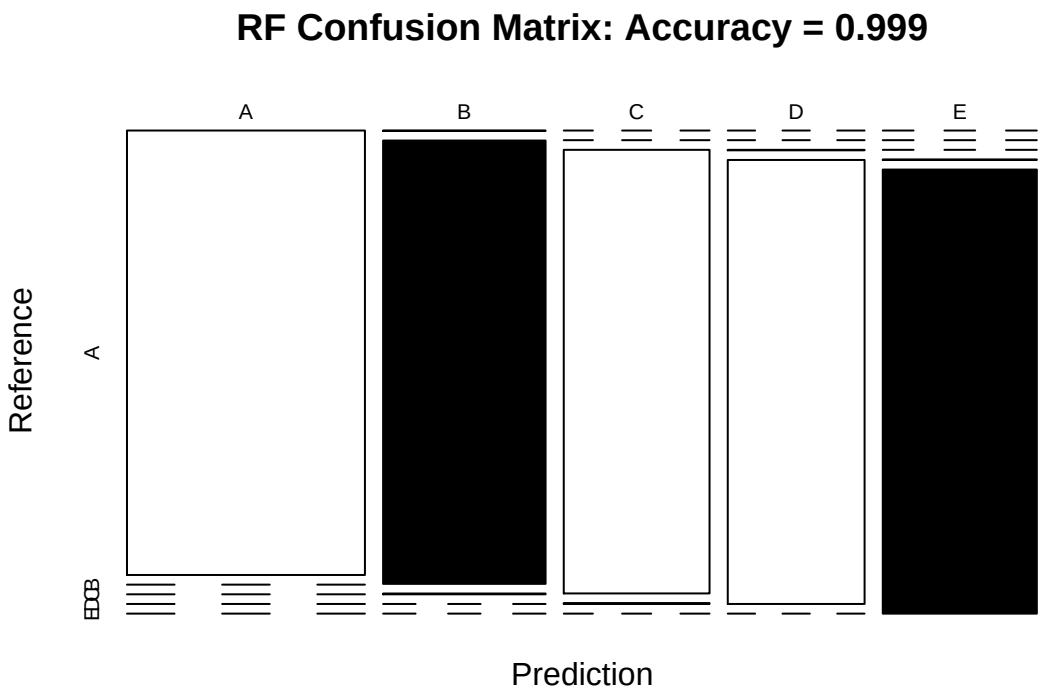
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    0
##          B    1 1139    2    0    0
##          C    0    0 1023    2    0
##          D    0    0    1  961    0
```

```
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 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            0.9994   1.0000   0.9971   0.9969   1.0000
## Specificity            1.0000   0.9994   0.9996   0.9998   0.9998
## Pos Pred Value         1.0000   0.9974   0.9980   0.9990   0.9991
## Neg Pred Value         0.9998   1.0000   0.9994   0.9994   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1935   0.1738   0.1633   0.1839
## Detection Prevalence   0.2843   0.1941   0.1742   0.1635   0.1840
## Balanced Accuracy      0.9997   0.9997   0.9983   0.9983   0.9999
```

```
plot(fit1)
```

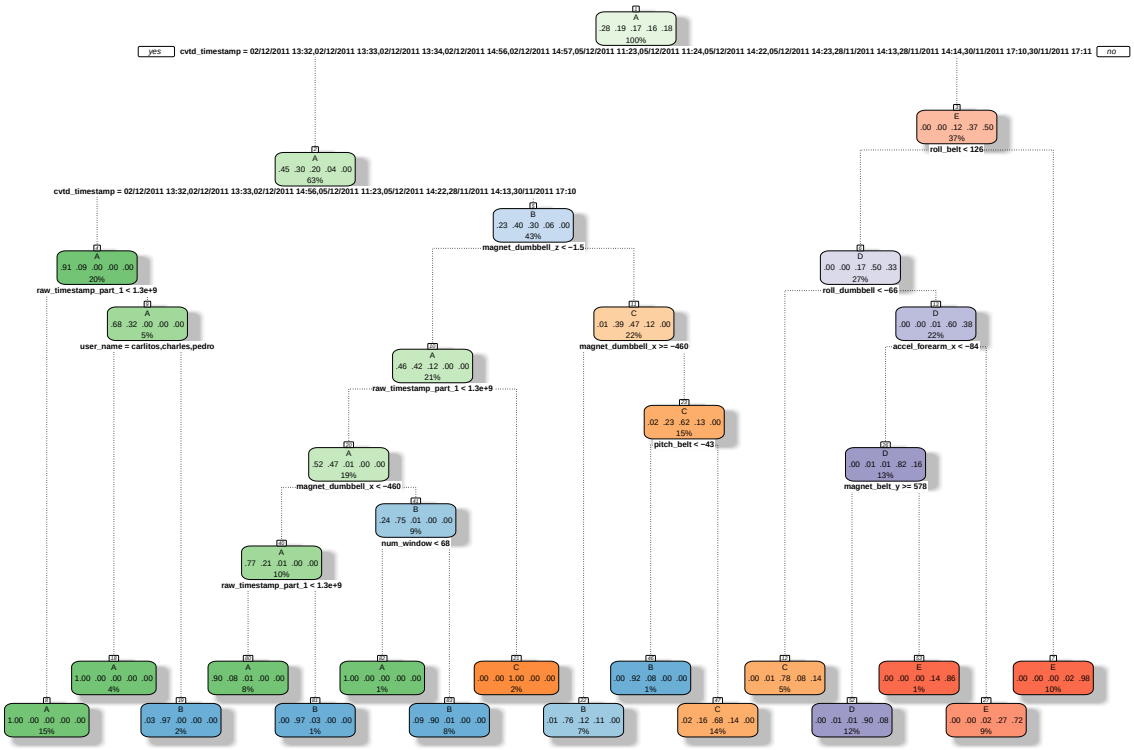


```
plot(confusionMatrixRF$table, col = confusionMatrixRF$byClass, main = paste("RF Confusion Matrix: Accura
```



Predicting with Decision Trees

```
set.seed(123)
fit2 <- rpart(classe ~ ., my.Training, method = "class")
fancyRpartPlot(fit2)
```



Rattle 2017−Sep−24 17:02:06 karth

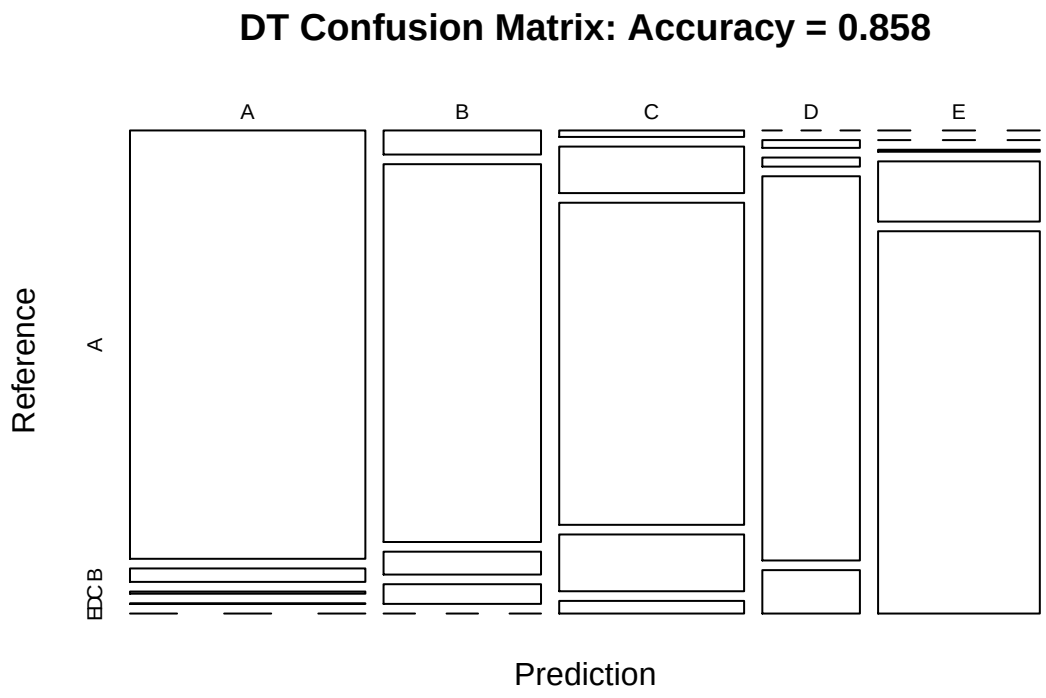
```
prediction2 <- predict(fit2, my.Testing, type = "class")

confusionMatrixDT <- confusionMatrix(prediction2, my.Testing$classe)
confusionMatrixDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1595   50    8    2    0
##          B   60  941   57   49    0
##          C   19  136  942  166   37
##          D    0   12   14  593   67
##          E    0    0    5  154  978
##
## Overall Statistics
##
##                Accuracy : 0.8579
##                  95% CI : (0.8488, 0.8668)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8203
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9528   0.8262   0.9181   0.6151   0.9039
## Specificity            0.9858   0.9650   0.9263   0.9811   0.9669
## Pos Pred Value         0.9637   0.8500   0.7246   0.8644   0.8602
## Neg Pred Value         0.9813   0.9586   0.9817   0.9286   0.9781
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2710   0.1599   0.1601   0.1008   0.1662
## Detection Prevalence   0.2812   0.1881   0.2209   0.1166   0.1932
## Balanced Accuracy      0.9693   0.8956   0.9222   0.7981   0.9354
```

```
plot(confusionMatrixDT$table, col = confusionMatrixDT$byClass, main = paste("DT Confusion Matrix: Accura
```



Summary

Since Random Forests prediction gave the most Accuracy of 99.9%. The expected out of sample error is 100-99.9 = 0.1%