Manner of Exercise

Swathi Vijayakumar

January 28, 2016

Purpose and background

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 inorder to improve health and find patterns in behavior. For this project we will 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. The goal of your project is to predict the manner in which they did the exercise.

Loading training data, testing Data and packages

```r
train_url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
HAR_Train = read.csv(url(train_url))
HAR_Test = read.csv(url(test_url))
dim(HAR_Train)
```

```
## [1] 19622   160
```

```r
dim(HAR_Test)
```

```
## [1]  20 160
```

```r
library(caret); library(rattle); library(randomForest)
```

```
## Warning: package 'caret' was built under R version 3.4.1
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.1
```

```
## Warning: package 'rattle' was built under R version 3.4.1
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## Warning: package 'randomForest' was built under R version 3.4.1
```

```
## 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
```

Data clean-up and transformation

Step 1: Remove columns with execisive number of missing values.

When you look at a summary of the data, it's clear that for some of the measurement columns, almost all of the data is missing. Count the number of missing values each column and remove all the columns with almost 98% of the values missing.

```
na_countTrain = sapply(HAR_Train, function(y) sum(length(which(is.na(y)))))
na_countTrain = data.frame(na_countTrain)

HAR_Train = HAR_Train [,colSums(is.na(HAR_Train)) < 19216]
dim(HAR_Train)
```

```
## [1] 19622    93
```

```
names = colnames(HAR_Train)
HAR_Test = HAR_Test[,names(HAR_Test) %in% names]
dim(HAR_Test)
```

```
## [1] 20 92
```

Step 2: Remove the first 5 column and near zero covariates

```
nzv = nearZeroVar(HAR_Train)
nzv
```

```
##  [1]  6 12 13 14 15 16 17 18 19 20 43 44 45 46 47 48 52 53 54 55 56 57 58
## [24] 59 60 74 75 76 77 78 79 80 81 82
```

```
HAR_Train = HAR_Train[,-nzv]
HAR_Test = HAR_Test[,-nzv]

HAR_Train = HAR_Train[,-(1:5)]
HAR_Test = HAR_Test[,-(1:5)]

dim(HAR_Train)
```

```
## [1] 19622    54
```

```
dim(HAR_Test)
```

```
## [1] 20 53
```

Model Building

Split the training data into training and validation sets. Since the the typical split for a training and a "testing" data set are 80%, 20% respectively that's how we'll split the data.

```
set.seed(230)
inTrain = createDataPartition(y=HAR_Train$classe, p=0.8, list=FALSE)
training = HAR_Train[inTrain,]
validation = HAR_Train[-inTrain,]
dim(training)
```

```
## [1] 15699    54
```

```r
dim(validation)
```

```
## [1] 3923   54
```

Predicting with Trees

We'll strart with building a decision tree on the training data set followed by predicting classe in the validation data set using this model.

```r
fitTree = train(classe~. ,data = training, method = "rpart")
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 3.4.1
```

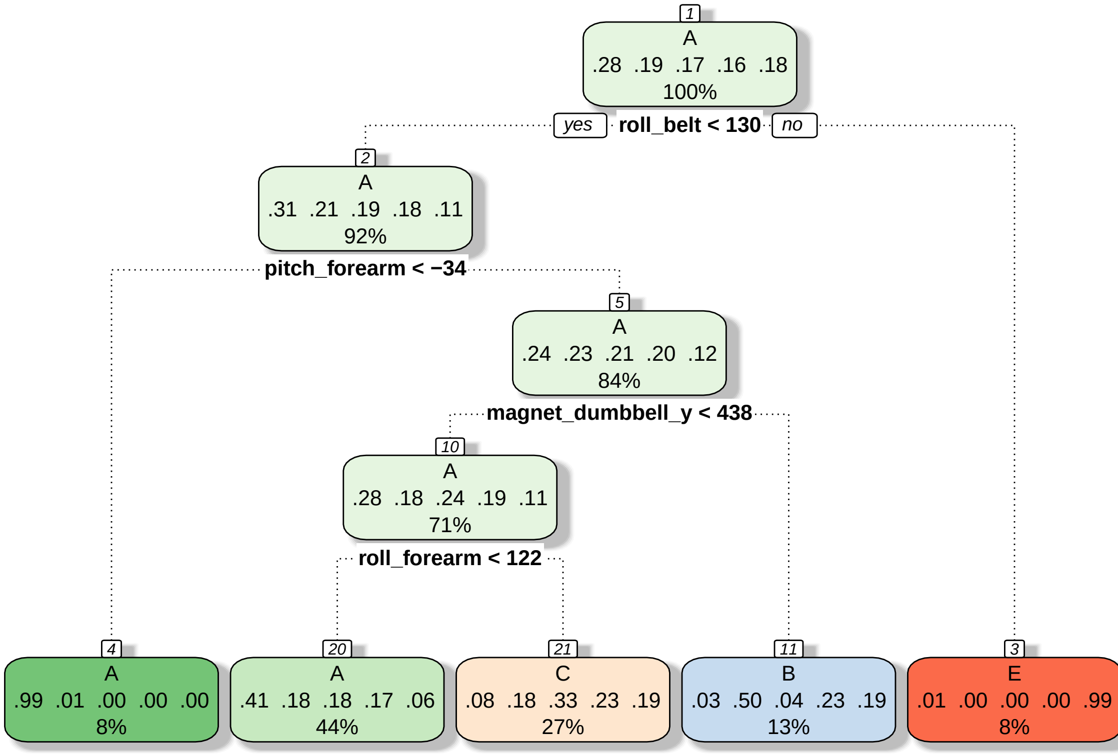
```r
print(fitTree)
```

```
## CART
##
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 15699, 15699, 15699, 15699, 15699, 15699, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03894081  0.5460738  0.41756040
##   0.06013945  0.3898231  0.16539594
##   0.11241656  0.3254069  0.06292539
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03894081.
```

```r
print(fitTree$finalModel, digits = 3)
```

```
## n= 15699
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 15699 11200 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130 14412  9960 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34 1269     9 A (0.99 0.0071 0 0 0) *
##      5) pitch_forearm>=-34 13143  9950 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 438 11098  7970 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 122 6860  4060 A (0.41 0.18 0.18 0.17 0.062) *
##         21) roll_forearm>=122 4238  2850 C (0.076 0.18 0.33 0.23 0.19) *
##       11) magnet_dumbbell_y>=438 2045  1010 B (0.033 0.5 0.043 0.23 0.19) *
##    3) roll_belt>=130 1287    12 E (0.0093 0 0 0 0.99) *
```

```r
fancyRpartPlot(fitTree$finalModel)
```



Rattle 2017−Jul−03 13:51:32 S.Vijayakumar

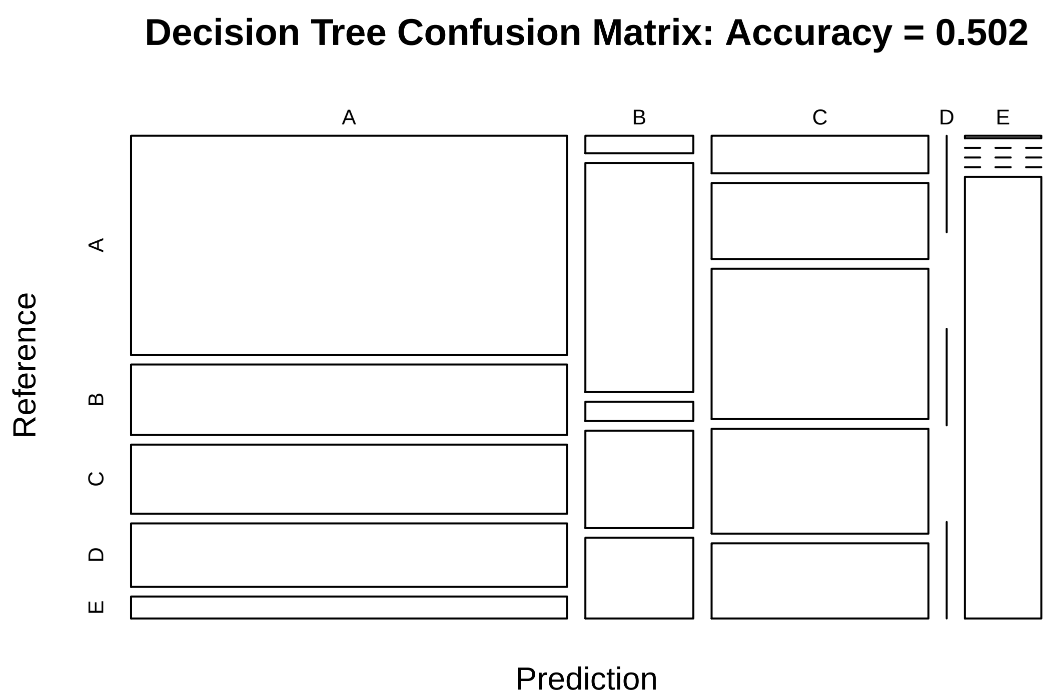
```
predTree = predict(fitTree, validation)


matrixTree = confusionMatrix(predTree, validation$classe)
matrixTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1008  324  318  292  101
##          B   20  261   22  111   92
##          C   86  174  344  240  172
##          D    0    0    0    0    0
##          E    2    0    0    0  356
##
## Overall Statistics
##
##                Accuracy : 0.5019
##                  95% CI : (0.4861, 0.5177)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3489
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9032  0.34387  0.50292   0.0000  0.49376
## Specificity            0.6313  0.92257  0.79253   1.0000  0.99938
## Pos Pred Value         0.4934  0.51581  0.33858      NaN  0.99441
## Neg Pred Value         0.9426  0.85426  0.88304   0.8361  0.89762
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2569  0.06653  0.08769   0.0000  0.09075
## Detection Prevalence   0.5208  0.12898  0.25899   0.0000  0.09126
## Balanced Accuracy      0.7673  0.63322  0.64773   0.5000  0.74657
```

```
plot(matrixTree$table, col = matrixTree$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy
```



The decision tree model used bootstrapped resampling with 25 reps. No model pre-processing was performed. Unfortunetly using a decision tree gives us a very poor accuracy (only 50%) when predicting in the validation set. Not the best model.

Predicting with Random Forest

We will next build a random forest model on the testing data set and predict classe in the validation data set using this model.

```
fitrf = randomForest(classe~. ,data = training)

print(fitrf)
```

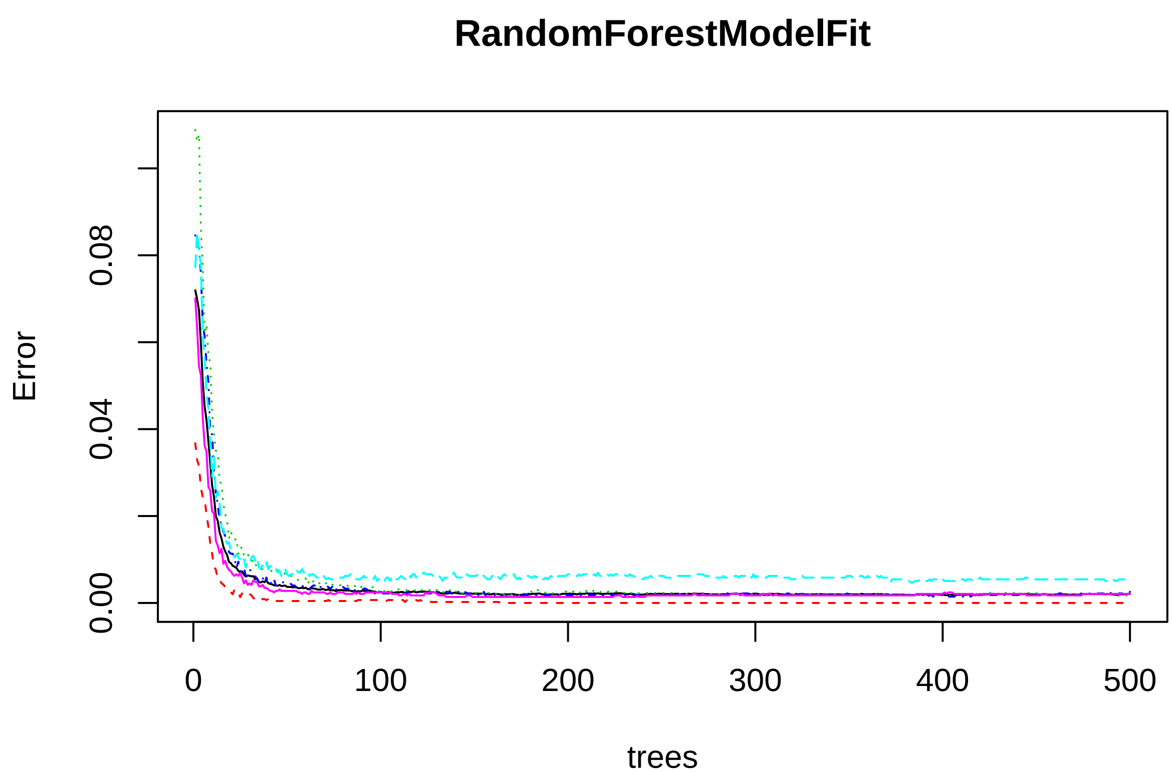
```
##
```

```
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.21%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4464    0    0    0    0 0.000000000
## B    5 3032    1    0    0 0.001974984
## C    0    6 2731    1    0 0.002556611
## D    0    0   13 2559    1 0.005441119
## E    0    0    0    6 2880 0.002079002
```

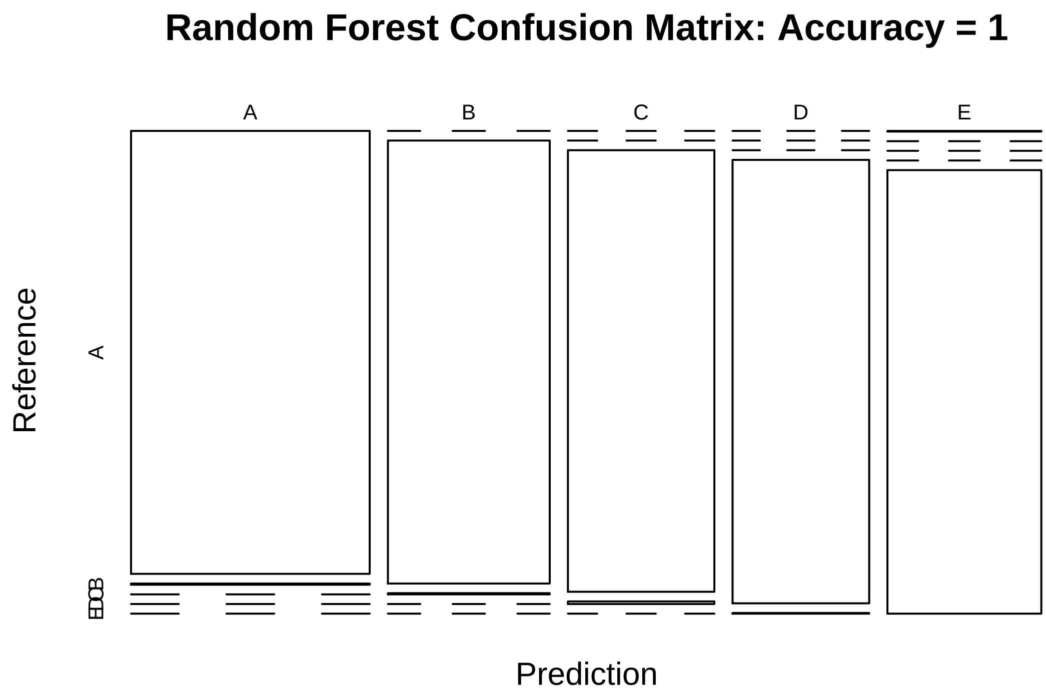
```r
predrf = predict(fitrf, validation)

matrixRF = confusionMatrix(predrf, validation$classe)

plot(fitrf, main = paste("RandomForestModelFit"))
```




```r
plot(matrixRF$table, col = matrixRF$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =",
```



For this dataset Random forest uses calssification with 500 trees and 7 variables at each split. Predicting on the validation set gives us a very good accuracy (99.7%). The out of sample error is 1-0.997 = 0.003. This is a good out of sample error. We will continue using the random forest model to predict on the testing set.

Use Random forest to make a prediction on the test set.

The final step is to run the final prediction on the 20 test set data points using the final Random Forest model.

```
predrfTest = predict(fitrf, HAR_Test)
predrfTest
```

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