Machine Learning Course Project

Amanda Mae

8/18/2019

Executive Summary

The below code takes data from the Human Activity Recognition project and tries to predict the class of the activity based on a set of 160 variables. From running two models, classification trees and random forests, I found the random forests model to have the greatest accuracy, with an accuracy of 0.9736, and an out of sample error of 0.0264. The final step in this file applies the random forests model to the validation data provided.

Note: The dataset used in this project is a courtesy of "Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements"

Section One Loading, Cleaning, and Formatting Data

1. Preprare workspace by installing necessary packages and loading training and test data.

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.2
```

```
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## 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(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.5.2
```

```
## Loaded gbm 2.1.5
```

```
train_in <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), heade
valid_in <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"), header
dim(train_in)
```

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

```
dim(valid_in)
```

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

We now have a training set with 19,622 observations of 160 variables and a validation set with 20 observations of 160 variables.

2. Remove varibales with missing values as these will not be useful for prediction.

```
trainData<- train_in[, colSums(is.na(train_in)) == 0]
validData <- valid_in[, colSums(is.na(valid_in)) == 0]
dim(trainData)
```

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

```
dim(validData)
```

```
## [1] 20 60
```

This bring us to 93 and 60 variables respectively.

3. Remove the first seven variables. These have participant information which should not be used to predict movement time.

```
trainData <- trainData[, -c(1:7)]
validData <- validData[, -c(1:7)]
dim(trainData)
```

```
## [1] 19622    86
```

```
dim(validData)
```

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

4. Split the training data into 70% training and 30% test. This will test our models, and the 20 validData observations will only be used in the end to validate the final model.

```
set.seed(1234)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData <- trainData[inTrain, ]
testData <- trainData[-inTrain, ]
dim(trainData)
```

```
## [1] 13737    86
```

```
dim(testData)
```

```
## [1] 4124   86
```

5. Clean near zero variance.

```r
NZV <- nearZeroVar(trainData)
trainData <- trainData[, -NZV]
testData  <- testData[, -NZV]
dim(trainData)
```

```
## [1] 13737    53
```

```r
dim(testData)
```

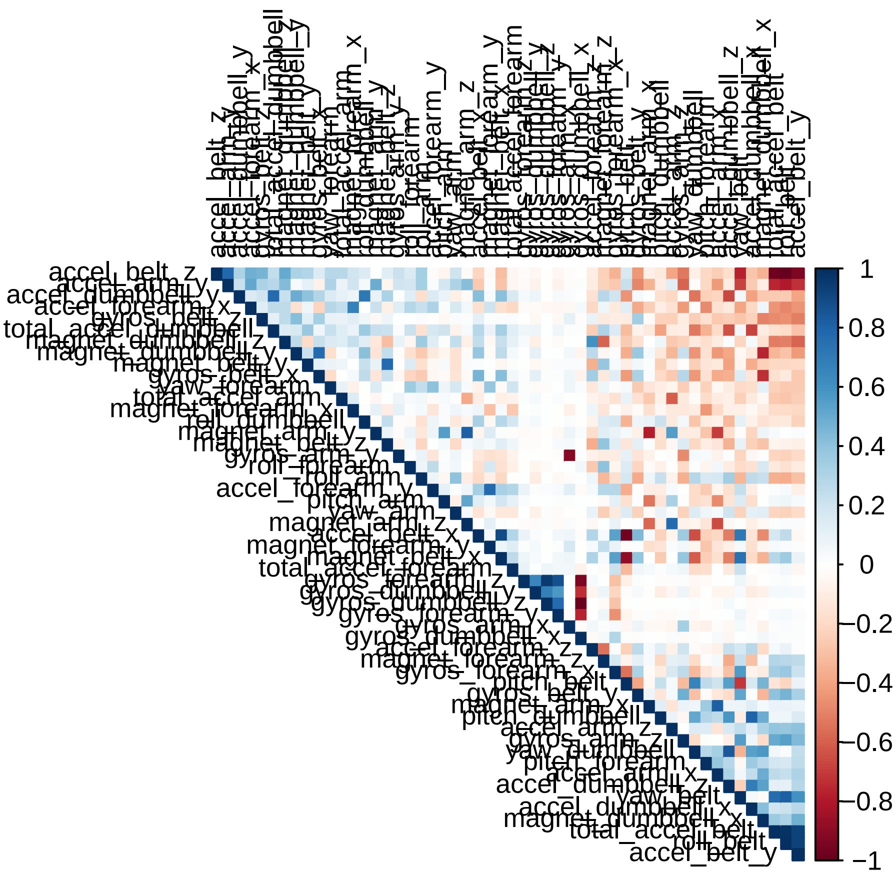
```
## [1] 4124   53
```

Now we have 53 variables to use to create our predictions.

Section Two **Correlation Plot**

1. Map a correlation plot using the r package corrplot.

```r
cor_mat <- cor(trainData[, -53])
corrplot(cor_mat, order = "FPC", method = "color", type = "upper",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



The darkest colors in the corrplot show the highest levels of correlation. These names, however, are hard to read, so I will create a variable, highlyCorrelated to print the names of the 75 most correlated variables, with a cutoff of 0.75.

```r
highlyCorrelated = findCorrelation(cor_mat, cutoff=0.75)
names(trainData)[highlyCorrelated]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "accel_dumbbell_z"  "accel_belt_x"
##  [7] "pitch_belt"        "magnet_dumbbell_x" "accel_dumbbell_y"
```

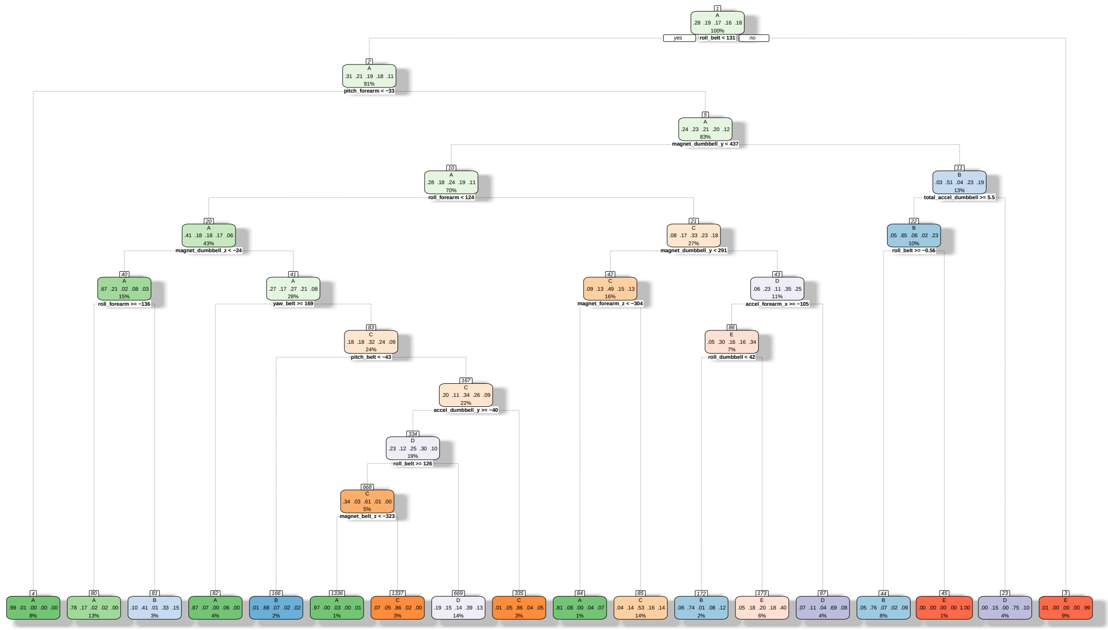
```
## [10] "magnet_dumbbell_y" "accel_arm_x"       "accel_dumbbell_x"
## [13] "accel_arm_z"       "magnet_arm_y"      "magnet_belt_z"
## [16] "accel_forearm_y"   "gyros_forearm_y"   "gyros_dumbbell_x"
## [19] "gyros_dumbbell_z"  "gyros_arm_x"
```

Section Three Model Building

I will use classification trees and random forests to try and predict the class variable.

1. Classification Trees: Build the model

```r
set.seed(12345)
decisionTreeMod1 <- rpart(classe ~ ., data=trainData, method="class")
fancyRpartPlot(decisionTreeMod1)
```



Rattle 2019−Aug−18 14:37:59 amandamae

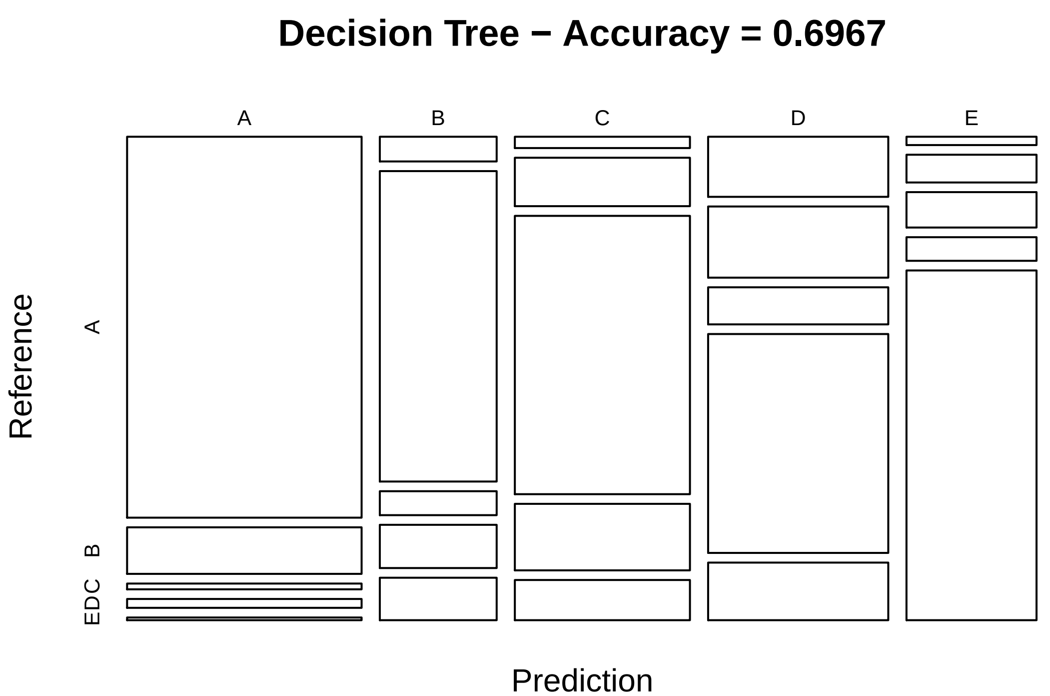
2. Classification Trees: Test model on test data and plot it.

```r
predictTreeMod1 <- predict(decisionTreeMod1, testData, type = "class")
cmtree <- confusionMatrix(predictTreeMod1, testData$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 990 121  15  23   7
##          B  32 402  31  56  55
##          C  22  94 540 129  78
##          D 120 142  74 437 115
##          E  12  40  51  34 504
##
## Overall Statistics
##
```

```
##                Accuracy : 0.6967
##                  95% CI : (0.6824, 0.7107)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6174
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8418  0.50313   0.7595   0.6436   0.6640
## Specificity            0.9437  0.94767   0.9054   0.8691   0.9593
## Pos Pred Value         0.8564  0.69792   0.6257   0.4921   0.7863
## Neg Pred Value         0.9373  0.88811   0.9476   0.9252   0.9268
## Prevalence             0.2852  0.19374   0.1724   0.1646   0.1840
## Detection Rate         0.2401  0.09748   0.1309   0.1060   0.1222
## Detection Prevalence   0.2803  0.13967   0.2093   0.2153   0.1554
## Balanced Accuracy      0.8928  0.72540   0.8324   0.7563   0.8117
```

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



Results of Classification Trees: The decision tree model has an accuracy of 0.6967. This is greater than 0.5 (the probability of flipping a coin), and therefore is a decent predictor.

3. Random Forests: Build the Model

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=trainData, method="rf", trControl=controlRF)
modRF1$finalModel
```

```
##
## Call:
```

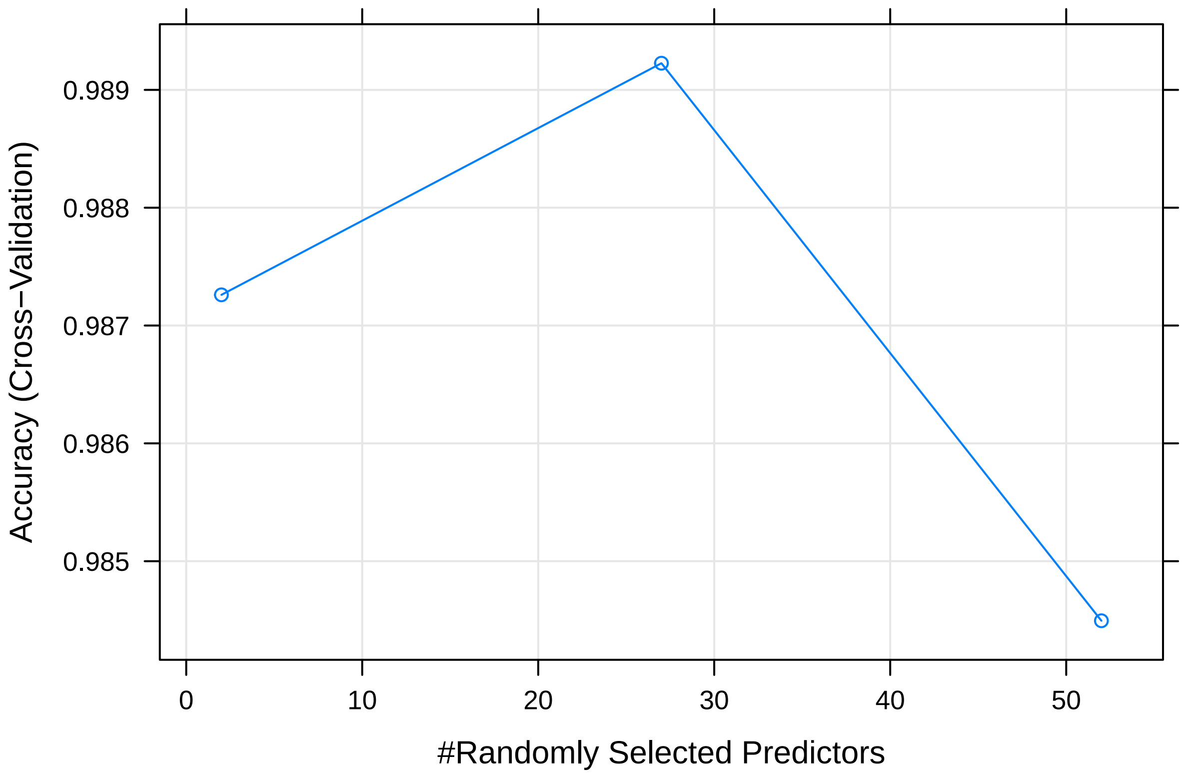
```
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.61%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   19 2633    6    0    0 0.009405568
## C    0   11 2380    5    0 0.006677796
## D    0    0   25 2225    2 0.011989343
## E    0    0    4    8 2513 0.004752475
```

4. Random Forests: Test model on test data and plot the model.

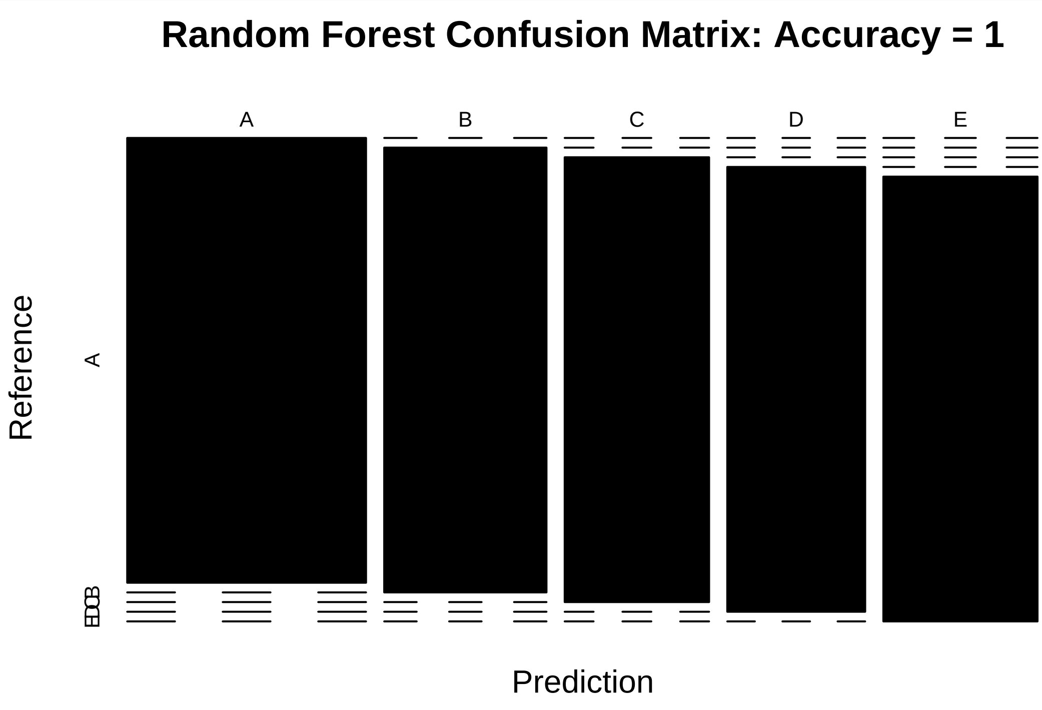
```
predictRF1 <- predict(modRF1, newdata=testData)
cmrf <- confusionMatrix(predictRF1, testData$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1176    0    0    0    0
##          B    0  799    0    0    0
##          C    0    0  711    0    0
##          D    0    0    0  679    0
##          E    0    0    0    0  759
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2852
##     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   1.0000   1.0000    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2852   0.1937   0.1724   0.1646    0.184
## Detection Rate         0.2852   0.1937   0.1724   0.1646    0.184
## Detection Prevalence   0.2852   0.1937   0.1724   0.1646    0.184
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

```
plot(modRF1)
```




```
plot(cmrf$table, col = cmrf$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round(cm
```



Results of Random Forest The random forests model has an accuracy of 0.9736 (almost 1!). This is greater than 0.5 (the probability of flipping a coin), and greater than the predictability of the classification trees model. This is my stronger model. The out of sample error for this model is 0.0264.

Section Four **Choose the best model and apply it to the validation set.**

```r
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
```

These results will now be used for the final quiz.