Machine Learning Exercise Movements

Andrew Washbrun

February 27, 2017

Executive Summary

The Internet of Things allows the classification of previously undocumented activities, such as exercise routines, in gross detail allowing machine learning prediction models to detect which type of exercise a user is attempting. Utilizing data from the Weight Lifting Exercise Dataset, classification and random forest prediciton models can predict what type of dumbell exercise a user is attempting based on accelerometer data gather on the user's arm, forearm, waist and dumbell.

Data Wrangling

Accelerometer data is download from dataset's website: http://groupware.les.inf.puc-rio.br/har . Training and validation sets are imported and cleaned up to remove uninterprettable values like NA, #DIV/0!, and empty values. Then the first several columns are removed because they contain non-movement related data like user and timestamp. Then the training set is split 70/30 into a training and testing set to prevent model overfitting for the validation set.

```
library(caret); library(rpart);library(rattle)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## 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.
```

```
training <- read.csv('pml-training.csv', na.strings=c("NA","#DIV/0!",""))
validation <- read.csv('pml-testing.csv', na.strings=c("NA","#DIV/0!",""))


#wrangling data into variables
validation <- validation[, colSums(is.na(training)) == 0]
training <- training[, colSums(is.na(training)) == 0]
training <- training[,-c(1:7)]
validation <- validation[,-c(1:7)]


inTrain <- createDataPartition(y=training$classe,
                               p=0.7, list=FALSE)
trainingData <- training[inTrain,]
testingData <- training[-inTrain,]

names(trainingData)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
```

```
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

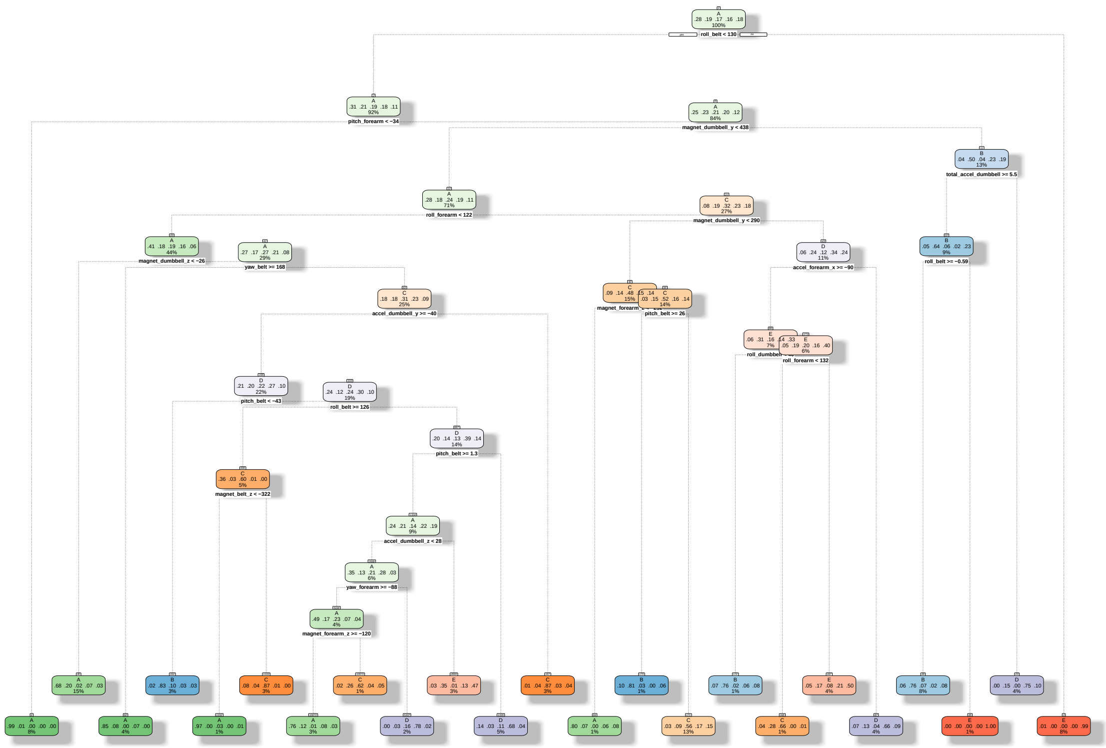
The remaining predictors are accelerometer data from the user and the "classe" variable which is the exercise type.

Recursive Partitioning

Classification trees partition data into logical trees where successive predictors lead to a classification. The classifiers are built such that the root of each branch contains a sufficiently "pure" class, meaning data with identical predictor values will be evaluated to that root class.

```r
#tree prediction
modFit <- rpart(classe ~ ., method='class', data=trainingData)
fancyRpartPlot(modFit)
```

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



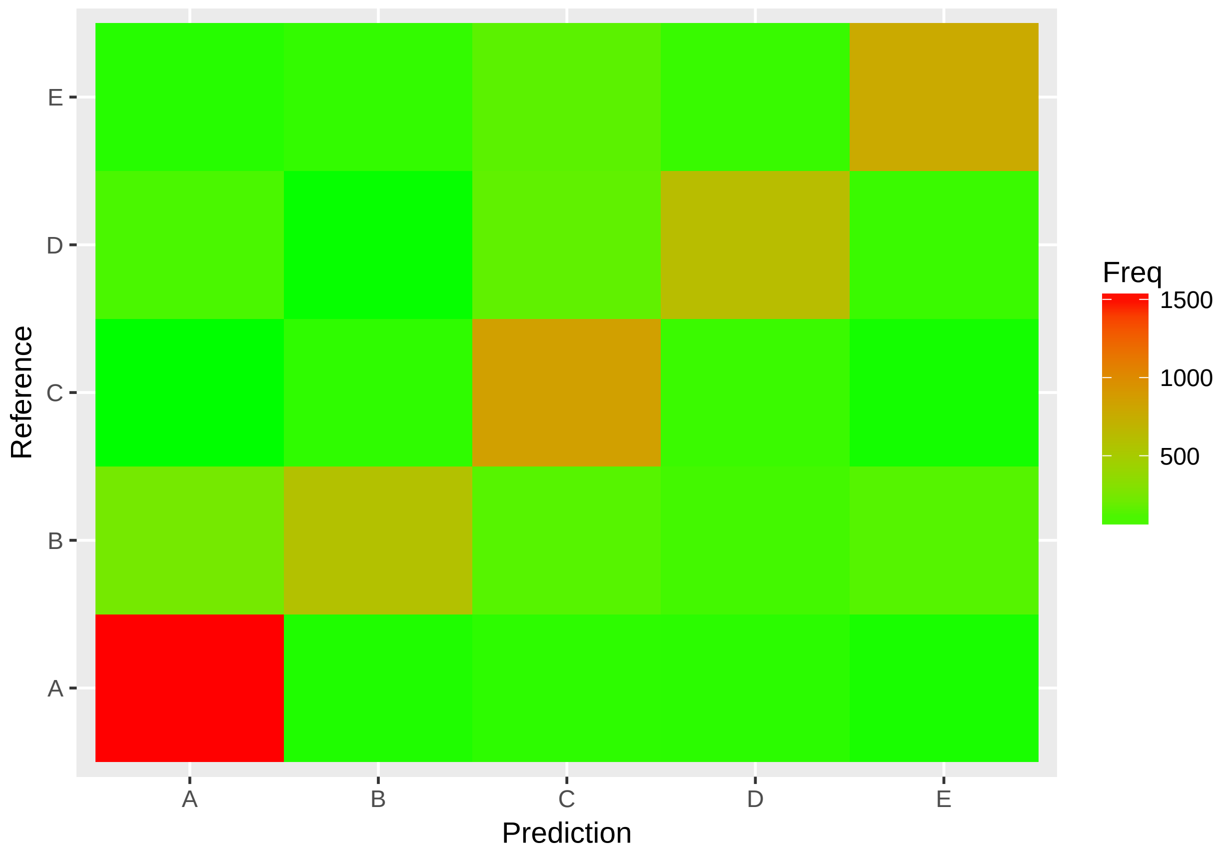
Rattle 2017−Feb−27 15:39:16 washbuan

```r
predictions <- predict(modFit,testingData,type='class')
confusionStats <- confusionMatrix(predictions,testingData$classe)
confusionStats
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1512  225   20   98   42
##          B   35  582   52   22   57
##          C   49  126  858  155  139
##          D   47   83   68  621   64
##          E   31  123   28   68  780
##
## Overall Statistics
##
##                Accuracy : 0.7397
##                  95% CI : (0.7283, 0.7509)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6695
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9032   0.5110   0.8363   0.6442   0.7209
## Specificity            0.9086   0.9650   0.9035   0.9468   0.9479
## Pos Pred Value         0.7970   0.7781   0.6466   0.7033   0.7573
## Neg Pred Value         0.9594   0.8916   0.9631   0.9314   0.9378
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2569   0.0989   0.1458   0.1055   0.1325
## Detection Prevalence   0.3223   0.1271   0.2255   0.1500   0.1750
## Balanced Accuracy      0.9059   0.7380   0.8699   0.7955   0.8344
```

```
tableStats <- as.data.frame(confusionStats$table)
ggplot(aes(Prediction,Reference),data=tableStats) +
  geom_tile(aes(fill=Freq)) + scale_fill_gradient(low="green", high="red")
```



The first plot illustrates the classification tree from the top (input) to the bottom (output), from the predictor values to the exercise outcome. The second output is a set of statistics for the model. The main takeaway is the accuracy: 71.5%. Not especially accurate so we'll employ another model. The last plot is an illustration of the Confusion Matrix statistics.

Random Forest

Random forests extend the idea of classification trees with random bootstrapping. By resampling and averaging models, a more robust decision tree is created.

```
#Random Forest
library(randomForest)
```

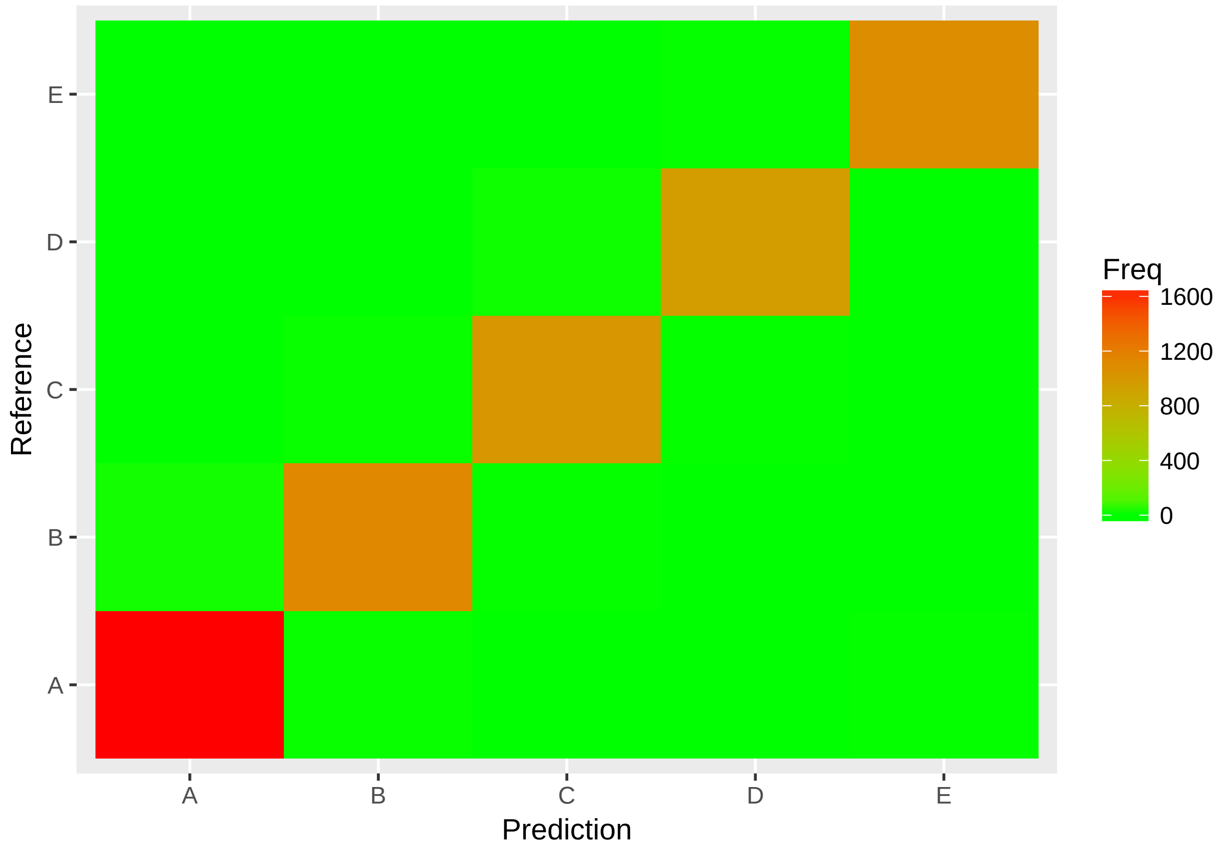
```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

```
forestModel <- randomForest(classe~.,data=trainingData)
forestPredictions <- predict(forestModel,testingData,type='class')
forestConfStats <- confusionMatrix(forestPredictions,testingData$classe)
forestConfStats
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    8    0    0    0
##          B    3 1129    3    0    0
##          C    0    2 1022    5    0
##          D    0    0    1  959    2
##          E    1    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9958
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9912   0.9961   0.9948   0.9982
## Specificity            0.9981   0.9987   0.9986   0.9994   0.9998
## Pos Pred Value         0.9952   0.9947   0.9932   0.9969   0.9991
## Neg Pred Value         0.9990   0.9979   0.9992   0.9990   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1918   0.1737   0.1630   0.1835
## Detection Prevalence   0.2851   0.1929   0.1749   0.1635   0.1837
## Balanced Accuracy      0.9979   0.9950   0.9973   0.9971   0.9990
```

```
forestTableStats <- as.data.frame(forestConfStats$table)
ggplot(aes(Prediction,Reference),data=forestTableStats) +
  geom_tile(aes(fill=Freq)) + scale_fill_gradient(low="green", high="red")
```



Comparing the accuracy between random forest and the standard classication tree model: 99.3% vs. 71.5% the random forest model is a more accurate model for prediction.