Machine Learning Example

Gary R Seamans

April 2018

```r
library(prettydoc)
library(ggplot2)
library(ggRandomForests)
library(caret)
library(knitr)
library(rpart)
library(e1071)
library(naivebayes)
#suppressMessages(library(rattle))
library(rattle)
library(randomForest)
set.seed(500)
```

Project Description

In this project the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which the participants did each exercise. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: Weight Lifting Data Description (see the section on the Weight Lifting Exercise Dataset).

Loading the data

The data sets were downloaded to a local directory from:

- Training Data
- Test Data

```r
wt_training <- read.csv(file = "./data/pml-training.csv")
wt_test <- read.csv(file = "./data/pml-testing.csv")
```

Exploratory Data Analysis and Cleaning

Intial analysis shows that there were 100 columns containing NAs. Once these were removed from the data sets 60 columns remained. Of these 60 columns, 7 contained data unrelated to accelerometer measurements and were removed leaving 53 columns in the training and test sets.

```r
# Data dimensions
dim(wt_training)
```

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

```r
dim(wt_test)
```

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

```r
# Check for columns containing NAs
naColumns <- colnames(wt_test)[colSums(is.na(wt_test)) > 0]
length(naColumns)
```

```
## [1] 100
```

```r
# Remove columns with only NAs
wtc_test <- wt_test[, !names(wt_test) %in% naColumns]
wtc_train <- wt_training[, !names(wt_training) %in% naColumns]
dim(wtc_test)
```

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

```r
dim(wtc_train)
```

```
## [1] 19622    60
```

```r
# Names of remaining columns
names(wtc_train)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

```r
# Remove columns unrelated to accelerometer readings
rm_names <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
              "cvtd_timestamp", "new_window", "num_window")
wtc_test <- wtc_test[, !names(wtc_test) %in% rm_names]
wtc_train <- wtc_train[, !names(wtc_train) %in% rm_names]
dim(wtc_test)
```

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

```r
dim(wtc_train)
```

```
## [1] 19622    53
```

```r
# Change all of the classes, except *classe* to numeric
wtc_train[1:52] <- lapply(wtc_train[1:52], as.numeric)
wtc_test[1:52] <- lapply(wtc_test[1:52], as.numeric)
# Get rid of the ID column
```

```
wtc_test <- wtc_test[1:52]
```

Partioning

The *wtc_trainig* set was partioned into *train* and *test* datsets. These data sets will be used for training and testing each of the models.

```
# Create the partitions
train_part <- createDataPartition(y = wtc_train$classe, p = 0.7, list = FALSE)
train <- wtc_train[train_part,]
test <- wtc_train[-train_part,]
# Make all of the columns of interest the same class
train[1:52] <- lapply(train[1:52], as.numeric)
test[1:52] <- lapply(test[1:52], as.numeric)
# Remove *classe* from the test set
#test <- test[,-53]

dim(train)
```

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

```
dim(test)
```

```
## [1] 5885   53
```

Modeling and Testing

Three different methods were used: - Random Forests - Naive Bayes - Boosting with trees (gbm)

Cross validation is builtin to Random Forests and Boosting with trees (gbm) and the defaults were used. Of the three different models used Naive Bayes performed the worst with defaults being used (but could be improved), Random Forests preformed the best closely followed by Boosting with trees.

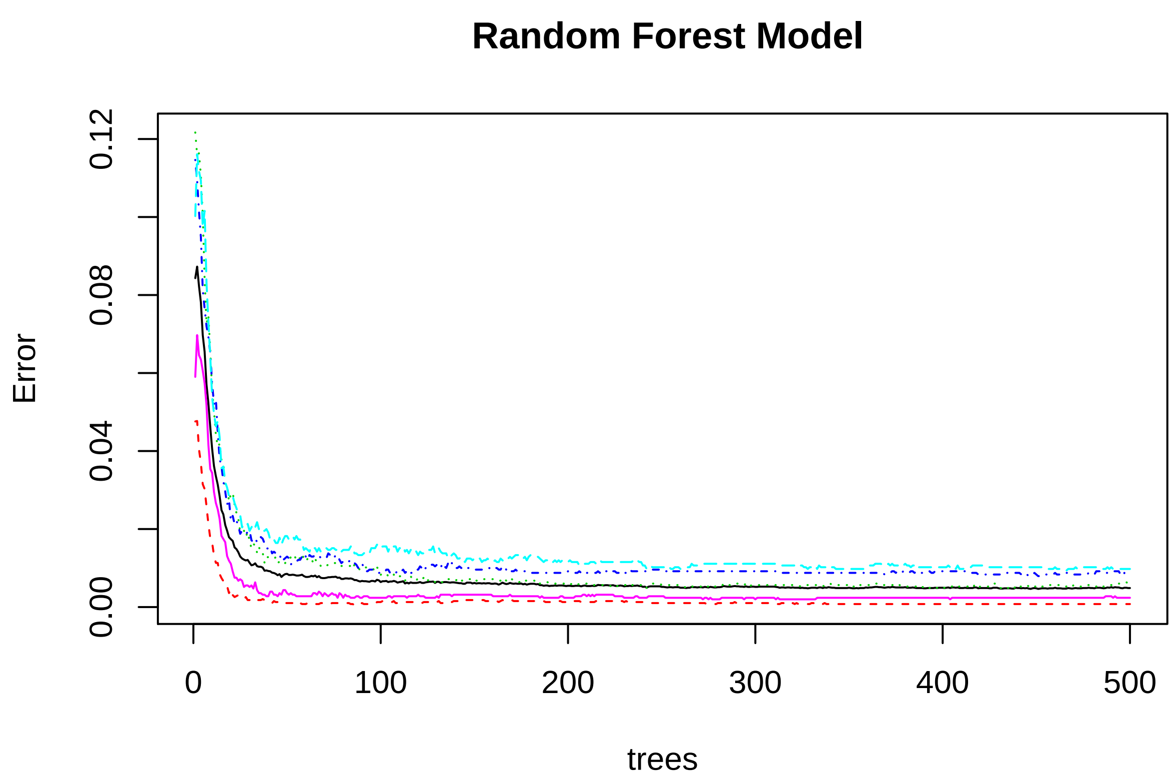
Random Forest, Naive Bayes, and Boosting with trees models.

```
# Create the Random Forest
set.seed(1313)
rf_fit <- randomForest(classe ~ ., data = train)
rf_pred <- predict(rf_fit, test, type = "class")
# Confusion matrix for the random forest
confusionMatrix(rf_pred, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673   11    0    0    0
##          B    1 1126    8    0    0
##          C    0    2 1015    9    0
##          D    0    0    3  954    0
##          E    0    0    0    1 1082
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9886   0.9893   0.9896   1.0000
## Specificity            0.9974   0.9981   0.9977   0.9994   0.9998
## Pos Pred Value         0.9935   0.9921   0.9893   0.9969   0.9991
## Neg Pred Value         0.9998   0.9973   0.9977   0.9980   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1913   0.1725   0.1621   0.1839
## Detection Prevalence   0.2862   0.1929   0.1743   0.1626   0.1840
## Balanced Accuracy      0.9984   0.9933   0.9935   0.9945   0.9999
```

```r
plot(rf_fit, main = " Random Forest Model")
```

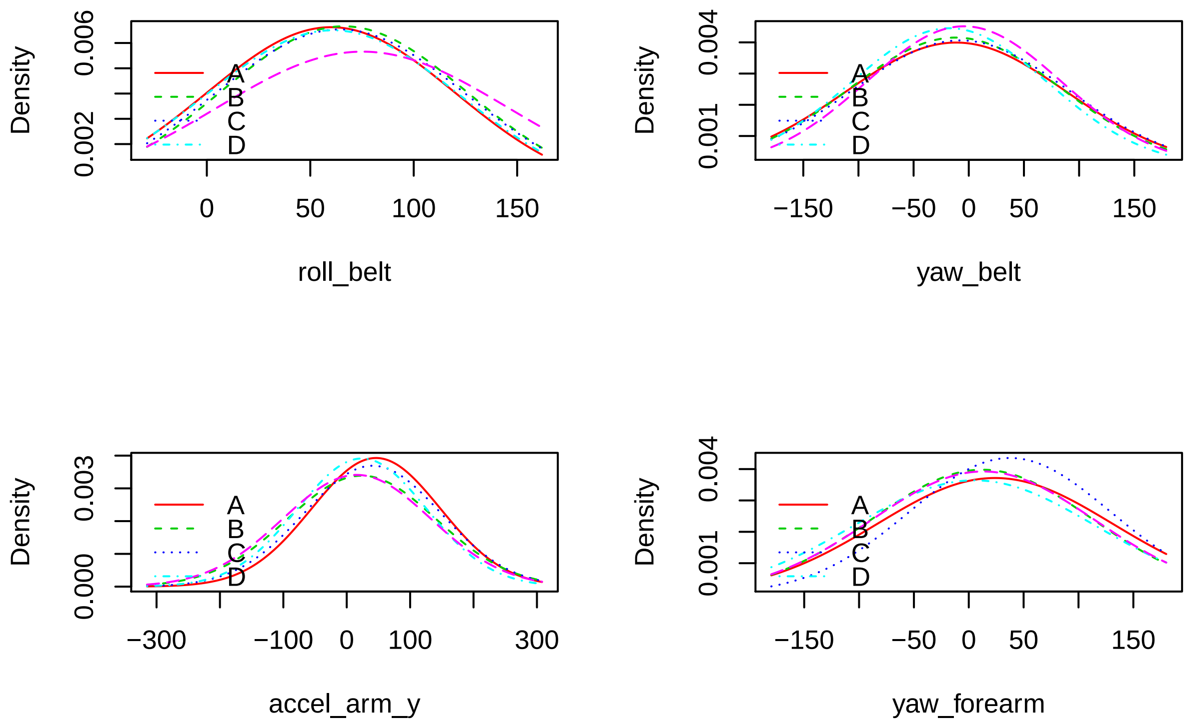



```r
# Test it on the validation set
```

```r
# Create a Naive Bayes classifier
nb_model <- naive_bayes(classe ~ ., data = train)
nb_predict <- predict(nb_model, test)
confusionMatrix(nb_predict, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061  124  238   94   54
##          B  101  715  107   24  239
##          C  288  160  518  257   94
##          D  188   85  128  461  143
##          E   36   55   35  128  552
##
## Overall Statistics
##
##                Accuracy : 0.5619
##                  95% CI : (0.5491, 0.5747)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4475
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6338   0.6277  0.50487  0.47822   0.5102
## Specificity            0.8789   0.9008  0.83556  0.88945   0.9471
## Pos Pred Value         0.6754   0.6029  0.39332  0.45871   0.6849
## Neg Pred Value         0.8579   0.9098  0.88879  0.89693   0.8956
## Prevalence             0.2845   0.1935  0.17434  0.16381   0.1839
## Detection Rate         0.1803   0.1215  0.08802  0.07833   0.0938
## Detection Prevalence   0.2669   0.2015  0.22379  0.17077   0.1370
## Balanced Accuracy      0.7563   0.7643  0.67022  0.68383   0.7286
```

```r
# Plot examples of the marginal probabilities, only four were selected for the examples.
par(mfrow=c(2,2))
plot(nb_model, which = c("roll_belt","yaw_belt","accel_arm_y","yaw_forearm"))
```

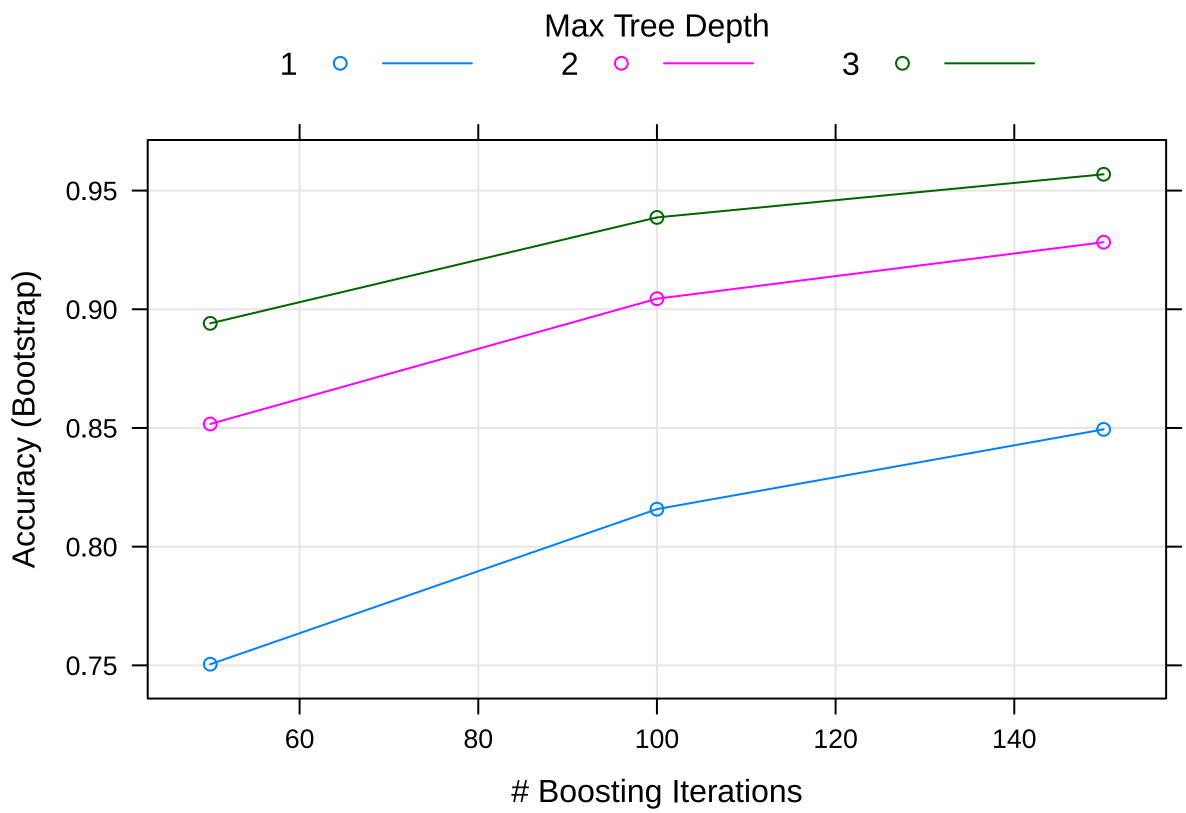



```r
par(mfrow=c(1,1))
# Boosting with trees *gbm*
boost_model <- train(classe ~ ., method = "gbm", data = train, verbose = FALSE)
boost_predict <- predict(boost_model, test)
confusionMatrix(boost_predict, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1643   33    0    3    2
##          B   17 1075   27    2   16
##          C    6   22  981   25   10
##          D    7    7   17  923   13
##          E    1    2    1   11 1041
##
## Overall Statistics
##
##                Accuracy : 0.9623
##                  95% CI : (0.9571, 0.967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9523
##  Mcnemar's Test P-Value : 7.521e-05
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9815   0.9438   0.9561   0.9575   0.9621
## Specificity            0.9910   0.9869   0.9870   0.9911   0.9969
## Pos Pred Value         0.9774   0.9455   0.9397   0.9545   0.9858
## Neg Pred Value         0.9926   0.9865   0.9907   0.9917   0.9915
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2792   0.1827   0.1667   0.1568   0.1769
## Detection Prevalence   0.2856   0.1932   0.1774   0.1643   0.1794
## Balanced Accuracy      0.9862   0.9654   0.9716   0.9743   0.9795
```

```
plot(boost_model)
```



Test Data predictions

The best result was using Random Forests so that is what I used for the test data set.

```
test_prediction <-  predict(rf_fit, wtc_test, type = "class")
test_prediction
```

```
##  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
```