Practical Machine Learning

Joe Larson

March 19, 2017

Data Importing

Course Project Assignment

“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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).”

The scripts have been solely produced, tested and executed on Asus Zenbook, R X64 3.1.2 and RStudio Version 0.98.1091.

Project Goal

The goal of this project is to predict the manner in which they did the exercise. The prediction model will be used to predict 20 different test cases. The variable in the training set - “classe” will be the determiedn factor. Describing how the model is built, how to cross validates it, evaluate the expected out of sample error, and explain the rationales of any choice made.

The 5 possible -Classe- values or methods are:

A: Exactly according to the specification

B: Throwing the elbows to the front

C: Lifting the dumbbell only halfway

D: Lowering the dumbbell only halfway

E: Throwing the hips to the front

The training data set (TrainSet) contains 19622 observations and 160 variables. The testing data set (TestSet) contains 20 observations and 160 variables. The

aim is to predict the outcome of the “classe” variable in the training set, validate it, and test it on the TestSet.

Data cleaning

Clean off the variable that countains mostly NAs (80%). Remove ‘X’, ‘Name’ and ‘timestamps’ variables (they are the culumns 1-5)

```
# create a partition with the training dataset
inTrain  <- createDataPartition(training$classe, p=0.8, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]

# remove variables with Nearly Zero Variance
nzv <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -nzv]
TestSet  <- TestSet[, -nzv]

# remove variables that are mostly NA, if more than 20% data, information inclu
ded
removeNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.80
TrainSet <- TrainSet[, removeNA==FALSE]
TestSet  <- TestSet[, removeNA==FALSE]

# remove identification variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]

dim(TrainSet)
```

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

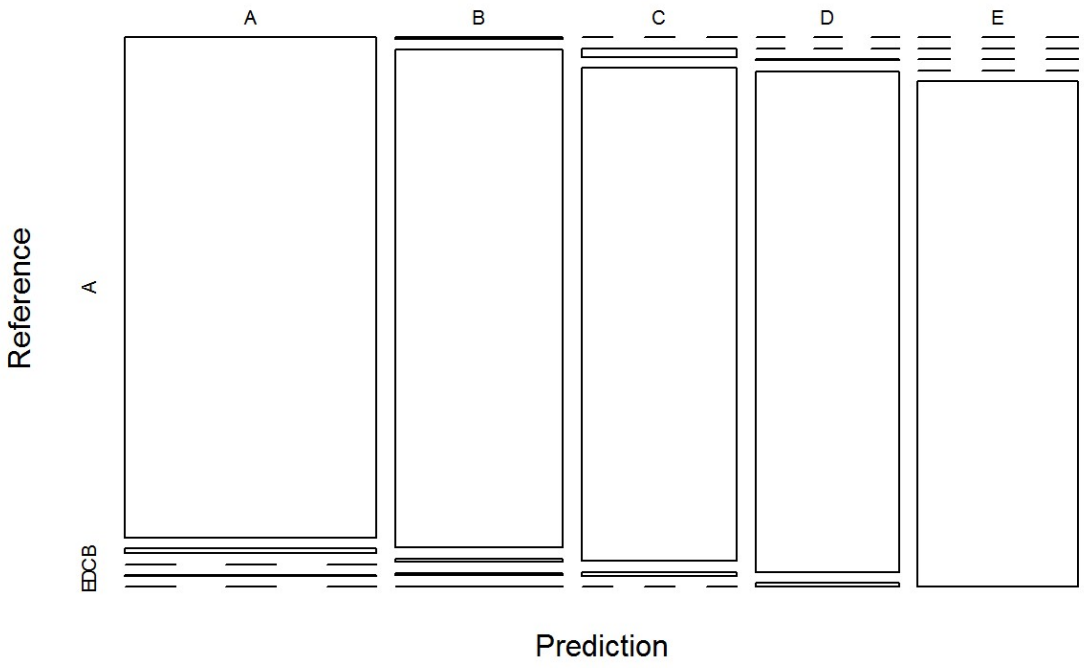
```
dim(TestSet)
```

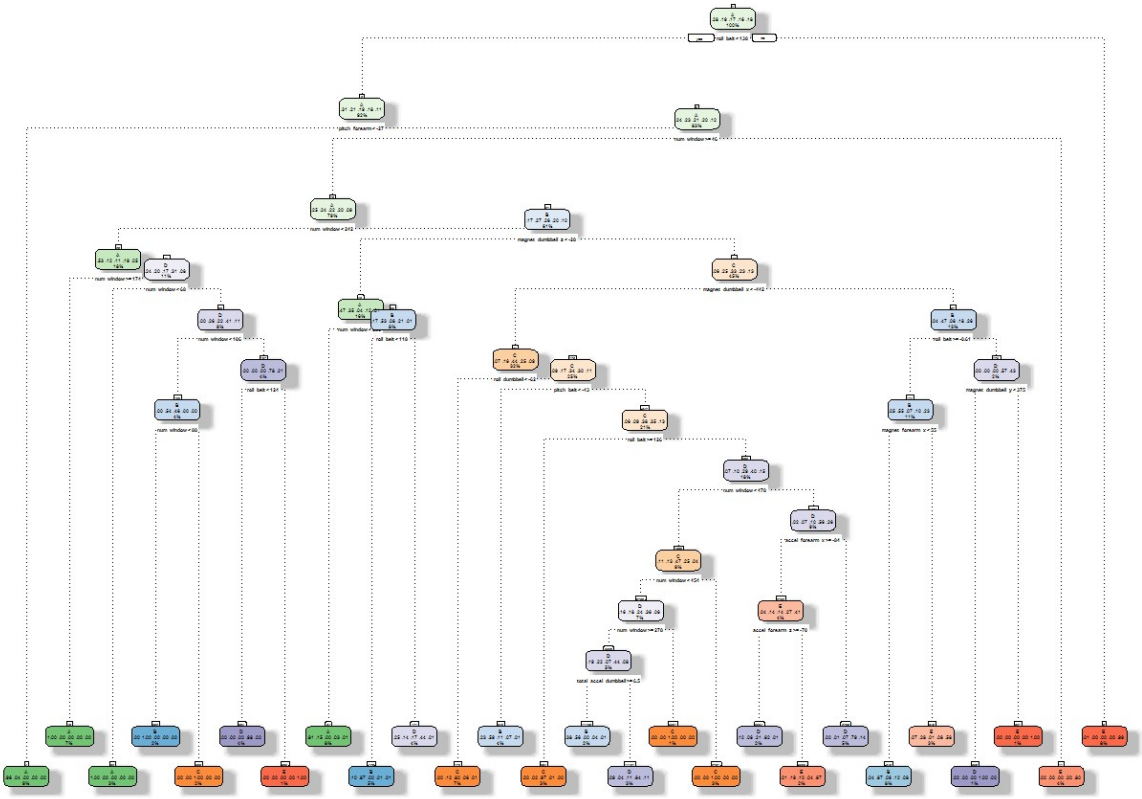
```
## [1] 3923   54
```

Including Plots of themethods tested

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

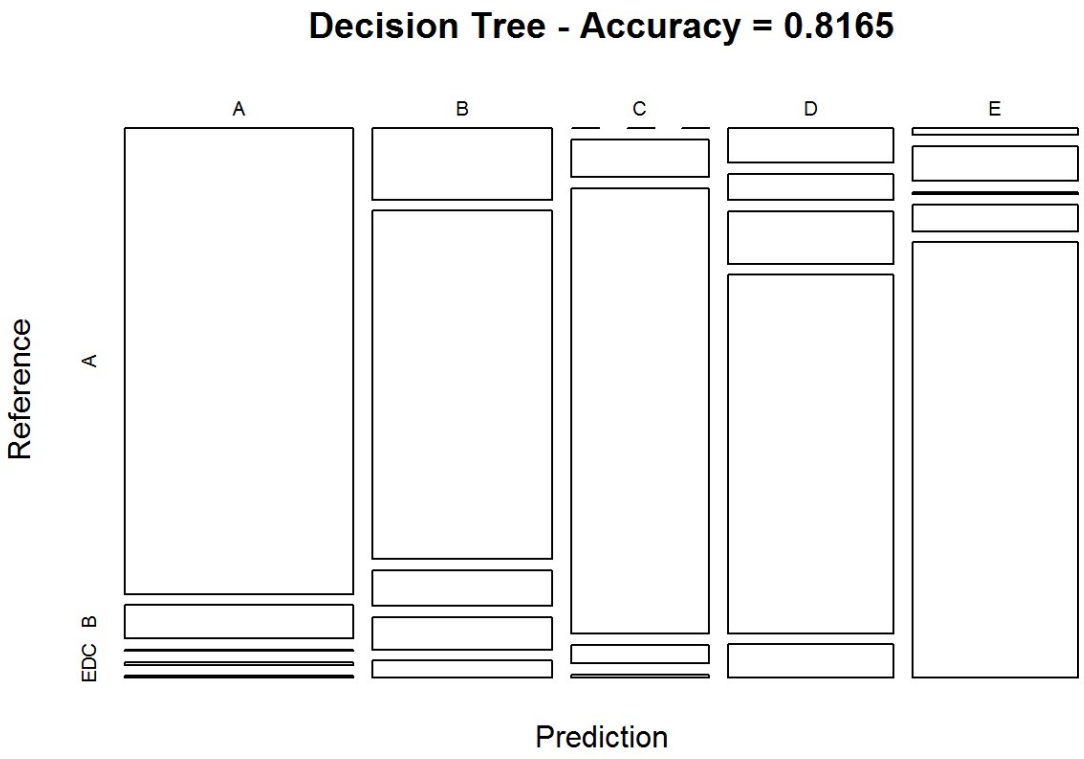
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114   11    0    1    0
##          B    2  736    5    3    1
##          C    0   12  678    5    0
##          D    0    0    1  634    5
##          E    0    0    0    0  715
##
## Overall Statistics
##
##                Accuracy : 0.9883
##                  95% CI : (0.9844, 0.9914)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9852
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9697   0.9912   0.9860   0.9917
## Specificity            0.9957   0.9965   0.9948   0.9982   1.0000
## Pos Pred Value         0.9893   0.9853   0.9755   0.9906   1.0000
## Neg Pred Value         0.9993   0.9928   0.9981   0.9973   0.9981
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1876   0.1728   0.1616   0.1823
## Detection Prevalence   0.2870   0.1904   0.1772   0.1631   0.1823
## Balanced Accuracy      0.9970   0.9831   0.9930   0.9921   0.9958
```


GBM - Accuracy = 0.9883
A
B
C
D
E
Reference
A
EDCB
Prediction



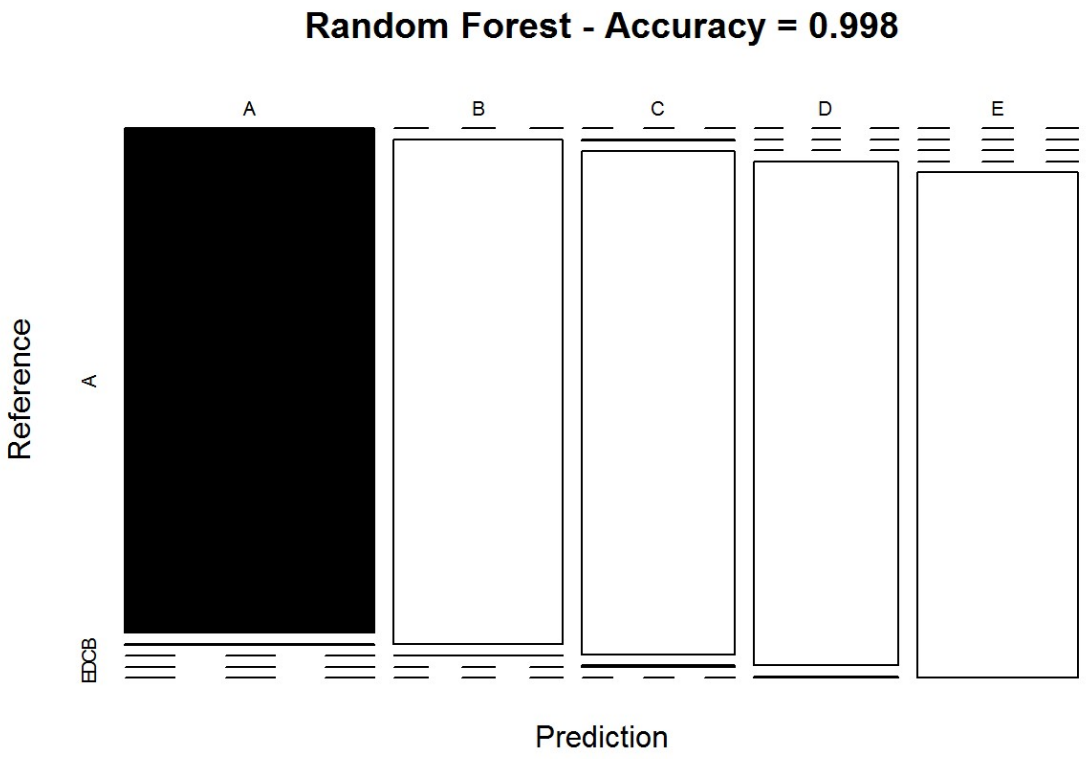
Rattle 2017-Mar-19 15:08:37 Joe

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 943  68   3   6   4
##          B 113 555  57  52  27
##          C   0  46 544  23   4
##          D  50  39  77 524  49
##          E  10  51   3  38 637
##
## Overall Statistics
##
##                Accuracy : 0.8165
##                  95% CI : (0.804, 0.8285)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7686
##  Mcnemar's Test P-Value : 1.755e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8450   0.7312   0.7953   0.8149   0.8835
## Specificity            0.9711   0.9213   0.9775   0.9345   0.9681
## Pos Pred Value         0.9209   0.6903   0.8817   0.7091   0.8620
## Neg Pred Value         0.9403   0.9346   0.9577   0.9626   0.9736
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2404   0.1415   0.1387   0.1336   0.1624
## Detection Prevalence   0.2610   0.2049   0.1573   0.1884   0.1884
## Balanced Accuracy      0.9081   0.8263   0.8864   0.8747   0.9258
```




```
##
## 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: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4462    1    0    0    1 0.0004480287
## B    5 3030    2    1    0 0.0026333114
## C    0    5 2733    0    0 0.0018261505
## D    0    0    7 2565    1 0.0031092110
## E    0    0    0    7 2879 0.0024255024
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    2    0    0    0
##          B    0  756    1    0    0
##          C    0    1  683    2    0
##          D    0    0    0  641    2
##          E    0    0    0    0  719
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.996, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9960   0.9985   0.9969   0.9972
## Specificity            0.9993   0.9997   0.9991   0.9994   1.0000
## Pos Pred Value         0.9982   0.9987   0.9956   0.9969   1.0000
## Neg Pred Value         1.0000   0.9991   0.9997   0.9994   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1927   0.1741   0.1634   0.1833
## Detection Prevalence   0.2850   0.1930   0.1749   0.1639   0.1833
## Balanced Accuracy      0.9996   0.9979   0.9988   0.9981   0.9986
```



##Performance evaluation of the Gradient Boosting Machine, Decision Tree and Random Forest algorithm.

The next step is to estimate the performance of the models on the validation data set here called (TestSet). A confusionMatrix function and both the estimated accuracy and the the estimated out-of-sample error of the model are calculated.

Prediction for the Test Data Set (TestSet).

Prediction using the model modelRandForest

Conclusions

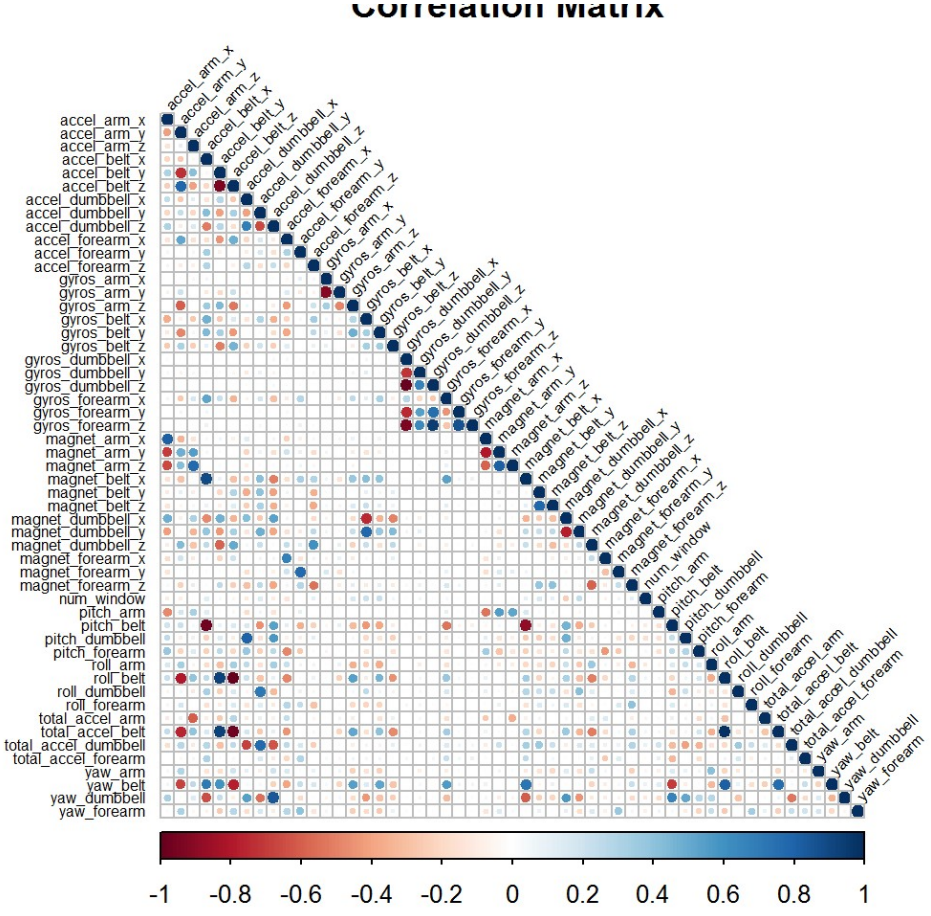
The random forest model provides the highest accuracy of the methods tested, 99.98%.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    2  756    1    0    0
##          C    0    1  683    0    0
##          D    0    0    2  641    0
##          E    0    0    0    2  719
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.996, 0.9991)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9987   0.9956   0.9969   1.0000
## Specificity            1.0000   0.9991   0.9997   0.9994   0.9994
## Pos Pred Value         1.0000   0.9960   0.9985   0.9969   0.9972
## Neg Pred Value         0.9993   0.9997   0.9991   0.9994   1.0000
## Prevalence             0.2850   0.1930   0.1749   0.1639   0.1833
## Detection Rate         0.2845   0.1927   0.1741   0.1634   0.1833
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9991   0.9989   0.9977   0.9981   0.9997
```

```
##  Accuracy     Kappa
## 0.9979607 0.9974205
```

```
## [1] 0.002039256
```

```
##  [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
```


Correlation Matrix
accel_arm_x
accel_arm_y
accel_arm_z
accel_belt_x
accel_belt_y
accel_belt_z
accel_dumbbell_x
accel_dumbbell_y
accel_dumbbell_z
accel_forearm_x
accel_forearm_y
accel_forearm_z
gyros_arm_x
gyros_arm_y
gyros_arm_z
gyros_belt_x
gyros_belt_y
gyros_belt_z
gyros_dumbbell_x
gyros_dumbbell_y
gyros_dumbbell_z
gyros_forearm_x
gyros_forearm_y
gyros_forearm_z
magnet_arm_x
magnet_arm_y
magnet_arm_z
magnet_belt_x
magnet_belt_y
magnet_belt_z
magnet_dumbbell_x
magnet_dumbbell_y
magnet_dumbbell_z
magnet_forearm_x
magnet_forearm_y
magnet_forearm_z
num_window
pitch_arm
pitch_belt
pitch_dumbbell
pitch_forearm
roll_arm
roll_belt
roll_dumbbell
roll_forearm
total_accel_arm
total_accel_belt
total_accel_dumbbell
total_accel_forearm
yaw_arm
yaw_belt
yaw_dumbbell
yaw_forearm
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1