PML

Seth Dobrin

March 14, 2016

Introduction

Devices that register almost every possible move or calorie usage on individuals are nearly ubiquitous. It is now very easy to collect and analyze large amounts of data about activity performed throughout the data and while exercising. However, very little effort has gone into evaluating weather activities are done correctly.

This project focuses on doing just such an analysis utilizing data collected by Groupware@LES. Six participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

A - exactly correctly
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

Activity was measured with sensors at the belt, upper arm and forearm.

This data set was used to in conjunction with a machine learning algorithm to predict if a test subject performed an activity correctly. Both Boosted Tree and Random Forrest algorithms were run and tested with the Random Forrest performing better.

Data Janitor Work

Load Data

```
trainFileName <- ("~/GitHub/PracticalML/Ppml-training.csv")
testFileName <- ("~/GitHub/PracticalML/pml-testing.csv")
trainRaw <- read.csv(trainFileName)
testRaw <- read.csv(testFileName)
dim(trainRaw)
## [1] 19622   160
dim(testRaw)
## [1]  20 160
```

The outcome being tested is the `classe` variable. The

Preprocess Data

Clean data by removing observations with messing and irrelevant values.

```
sum(complete.cases(trainRaw))
## [1] 406
```

Removing missing data:

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

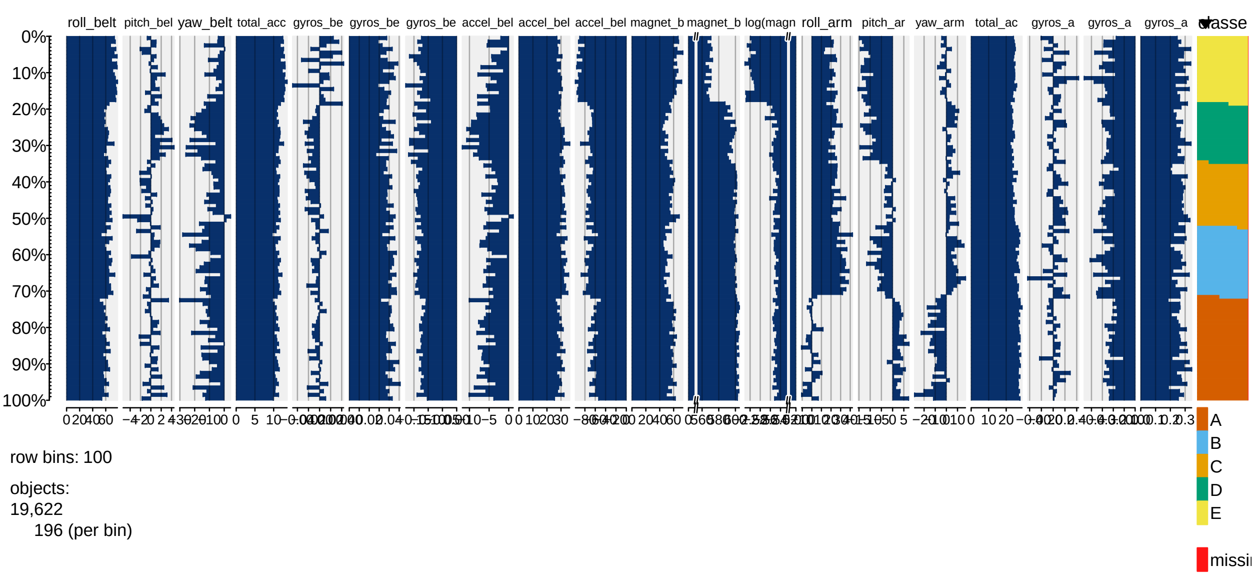
Remove irrelevant data:

```
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
dim(trainCleaned)
## [1] 19622    53
dim(testCleaned)
## [1] 20 53
```

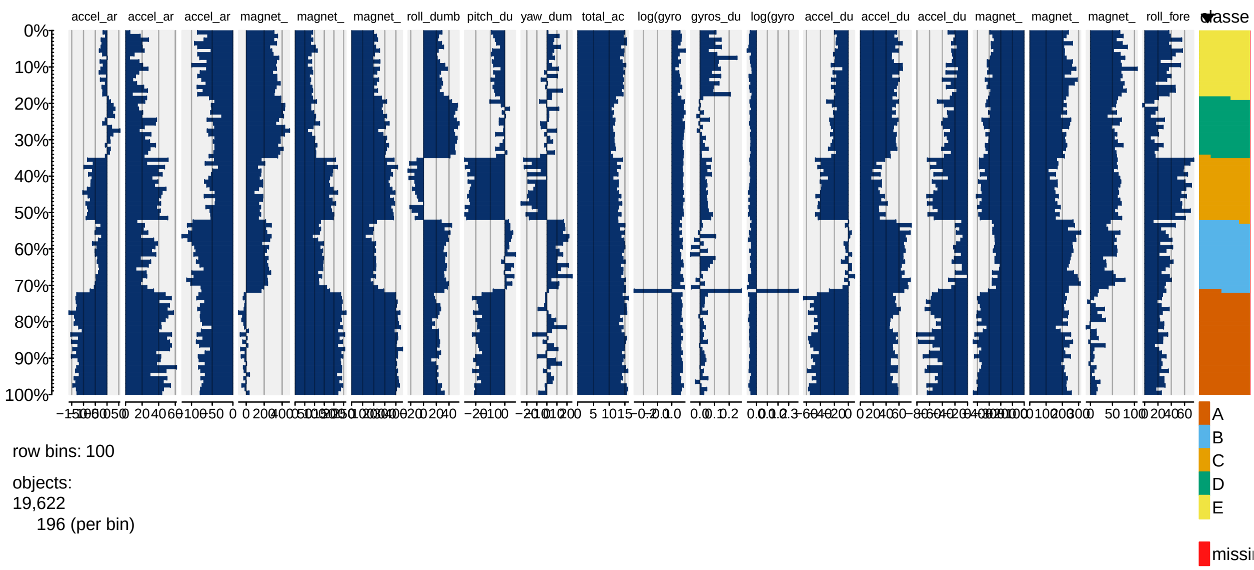
The data set was reduced to 53 variable from 160 variables my eliminating irrelevant variables and variables that contain missing data.

Data Exploration

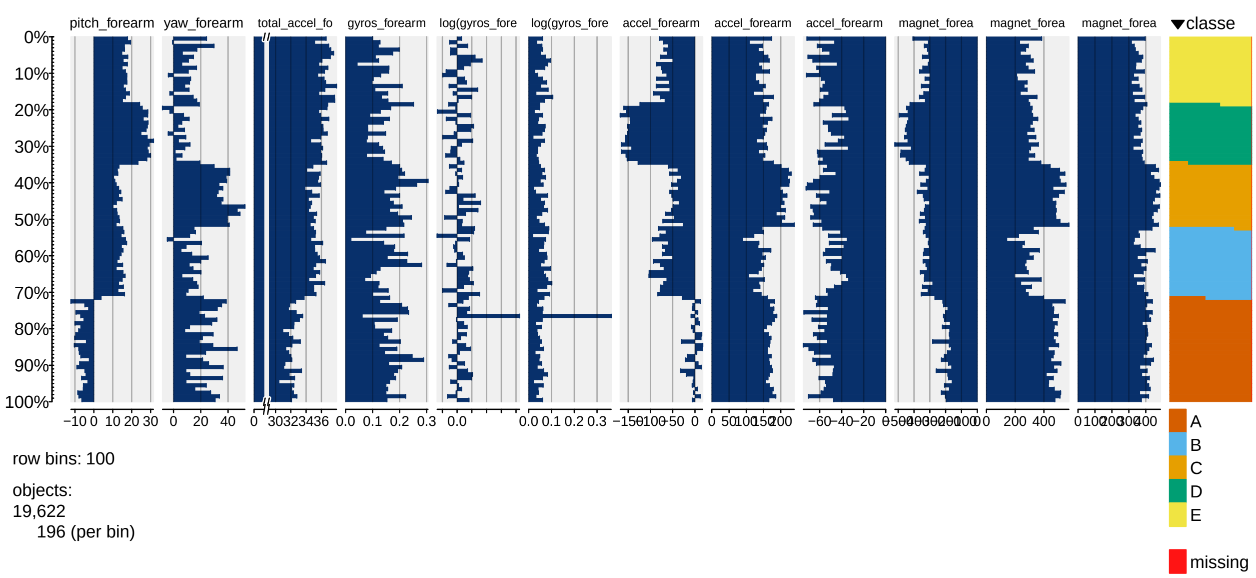
```
require(tabplot)
cols <- c(1:20, 53)
tableplot(trainCleaned[, cols], sortCol = "classe")
## Warning in grid.Call(L_convert, x, as.integer(whatfrom),
## as.integer(whatto), : font width unknown for character 0x9
```



```r
cols <- c(21:40, 53)
tableplot(trainCleaned[, cols], sortCol = "classe")
## Warning in grid.Call(L_convert, x, as.integer(whatfrom),
## as.integer(whatto), : font width unknown for character 0x9
```




```r
cols <- c(41:53)
tableplot(trainCleaned[, cols], sortCol = "classe")
## Warning in grid.Call(L_convert, x, as.integer(whatfrom),
## as.integer(whatto), : font width unknown for character 0x9
```



Partition the Data

Split the cleaned training set into a training set (70%) and a validation set (30%). The validation data set will be used for cross validation.

```r
require(caret)
set.seed(1969)
inTrain <- createDataPartition(trainCleaned$classe, p = 0.7, list = F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Model Build

`Boosted Tree` and `Random Forrest` will be run and compared to find the best model

Setup Parallel Processing

```r
require(parallel)
require(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
```

Boosted Tree

```r
require(bst)
## Warning: package 'bst' was built under R version 3.2.4
modelDt <- train(classe ~ ., data = trainData, method = "bstTree", trControl = fitControl)
summary(modelDt)
##              Length Class       Mode
## y             13737 -none-      numeric
## x            714324 -none-      numeric
## cost              1 -none-      numeric
## family            1 -none-      character
## learner           1 -none-      character
## yhat          13737 -none-      numeric
## offset            1 -none-      numeric
## ens             150 -none-      list
## control.tree      1 -none-      list
## risk            150 -none-      numeric
## ctrl             18 bst_control list
## maxdepth          1 -none-      numeric
## xselect          42 -none-      numeric
## coef            150 -none-      logical
## ensemble        150 -none-      list
## ml.fit           14 rpart       list
## meanx         13737 -none-      numeric
## int               0 -none-      NULL
## call              7 -none-      call
## xNames           52 -none-      character
## problemType       1 -none-      character
## tuneValue         3 data.frame  list
## obsLevels         5 -none-      character
```

An estimate of the performance of the model on the validation data set is obtained:

```
predictDt <- predict(modelDt, testData)
confusionMatrix(testData$classe, predictDt)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1242  432    0    0    0
##          B   67 1072    0    0    0
##          C    4 1022    0    0    0
##          D    4  960    0    0    0
##          E   14 1068    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.3932
##                  95% CI : (0.3807, 0.4058)
##     No Information Rate : 0.7738
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2279
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9331   0.2354       NA       NA       NA
## Specificity            0.9051   0.9497   0.8257   0.8362   0.8161
## Pos Pred Value         0.7419   0.9412       NA       NA       NA
## Neg Pred Value         0.9789   0.2663       NA       NA       NA
## Prevalence             0.2262   0.7738   0.0000   0.0000   0.0000
## Detection Rate         0.2110   0.1822   0.0000   0.0000   0.0000
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9191   0.5925       NA       NA       NA
```

```
accuracyDt <- postResample(predictDt, testData$classe)
summary(accuracyDt)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2279  0.2692  0.3105  0.3105  0.3519  0.3932
```

```
dt <- 1 - as.numeric(confusionMatrix(testData$classe, predictDt)$overall[1])
summary(dt)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6068  0.6068  0.6068  0.6068  0.6068  0.6068
```

Boosted Tree Test Set Prediction

The model is applied to the original testing data set downloaded from the data source. The problem_id column is removed first.

```r
resultDt <- predict(modelDt, testCleaned[, -length(names(testCleaned))])
summary(resultDt)
##  A  B  C  D  E
##  5 15  0  0  0
```

Random Forrest

```r
require(randomForest)
modelRf <- train(classe ~ ., data = trainData, method = "rf", trControl = fitControl)
stopCluster(cluster)
summary(modelRf)
##                 Length Class      Mode
## call                4  -none-     call
## type                1  -none-     character
## predicted       13737  factor     numeric
## err.rate         3000  -none-     numeric
## confusion          30  -none-     numeric
## votes           68685  matrix     numeric
## oob.times       13737  -none-     numeric
## classes             5  -none-     character
## importance         52  -none-     numeric
## importanceSD        0  -none-     NULL
## localImportance     0  -none-     NULL
## proximity           0  -none-     NULL
## ntree               1  -none-     numeric
## mtry                1  -none-     numeric
## forest             14  -none-     list
## y               13737  factor     numeric
## test                0  -none-     NULL
## inbag               0  -none-     NULL
## xNames             52  -none-     character
## problemType         1  -none-     character
## tuneValue           1  data.frame list
## obsLevels           5  -none-     character
```

An estimate of the performance of the model on the validation data set is obtained:

Prediction

```r
predictRf <- predict(modelRf, testData)
confusionMatrix(testData$classe, predictRf)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B   13 1125    1    0    0
##          C    0   17 1009    0    0
##          D    0    0   30  933    1
```

```
##          E    0    0    0    5 1077
##
## Overall Statistics
##
##                Accuracy : 0.9886
##                  95% CI : (0.9856, 0.9912)
##     No Information Rate : 0.2867
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9856
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9923   0.9851   0.9702   0.9947   0.9991
## Specificity            1.0000   0.9970   0.9965   0.9937   0.9990
## Pos Pred Value         1.0000   0.9877   0.9834   0.9678   0.9954
## Neg Pred Value         0.9969   0.9964   0.9936   0.9990   0.9998
## Prevalence             0.2867   0.1941   0.1767   0.1594   0.1832
## Detection Rate         0.2845   0.1912   0.1715   0.1585   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9961   0.9911   0.9833   0.9942   0.9990
```

Accuracy

```
accuracy <- postResample(predictRf, testData$classe)
summary(accuracy)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9856  0.9863  0.9871  0.9871  0.9879  0.9886
```

Out of Sample Error

```
oose <- 1 - as.numeric(confusionMatrix(testData$classe, predictRf)$overall[1])
summary(oose)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01138 0.01138 0.01138 0.01138 0.01138 0.01138
```

As a result, the estimated accuracy of the model is 98.886% and the estimated out-of-sample error is 1.13%.

Random Forrest Test Set Prediction

The model is applied to the original testing data set downloaded from the data source. The problem_id column is removed first.

```
resultRf <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
summary(resultRf)
## A B C D E
## 7 8 1 1 3
```

Final model and prediction

Comparing model accuracy of the two models generated, random forests and boosting, random forests model has overall better accuracy, therefore Random Forrest will be used for the prediction. The final random forests model contains 500 trees with 40 variables tried at each split.

Predict the test set and output.

```
prediction <- as.character(resultRf)
```