Human Activity Recognition Prediction Study

RB Wiley

August 20, 2015

Executive Summary

Human Activity Recognition (HAR) has gained increased attention by an increasingly health-aware public. This is paralleled by a community of researchers interested in collecting and analyzing those data for a myriad of beneficial purposes. These include the development of context-aware systems having many potential applications for HAR. Such preventive applications as elderly monitoring and monitoring energy expenditure. Other commercially profitable applications have arisen such as supporting weight-loss programs and digital assistants for weight lifting exercises.

The Question

This report focuses on the question: can modeling of such data be used to develop prediction algorithms that are of the quality of the exercise activities performed.

The Input Data and a Citation for the source of the data

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Select and load the data for analysis.

Read training and test datasets into tables; Note: these data are already partitioned and are obtained as predivided training and testing sets. Note: not shown in this report: (View some training data (head()); view the data structures to confirm they match.) To assure integrity of the test process, the testing data is not even viewed at this step of the modeling and analysis preparation.

```
## Loading required package: lattice
## Loading required package: ggplot2
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.

training <- read.csv( "./data/pml-training.csv")
testing <- read.csv("./data/pml-testing.csv")
```

While viewing the structure of each data set (str()) and summary statistics (summary()), it is observed that there are lots of NA and missing (or null) values in the training dataset.

Some 'scrubbing' of the data is necessary before proceeding with EDA and the model development.
The training set will be partitioned to provide a validation subset of 25% of the initial training dataset.
But 1st: Perform EDA

Exploratory Data Analysis (EDA)

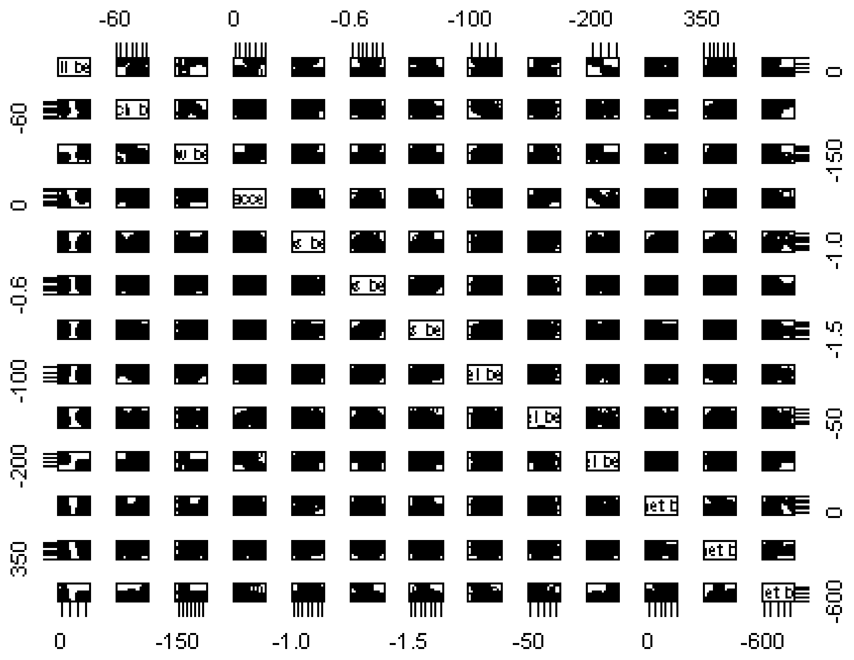
Six subject participants performed one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl. They varied the exercise in five ways (the five levels in the variable classe as follows:
1. Exactly according to the specification (Class A), 2. Throwing the elbows to the front (Class B), 3. Lifting the dumbbell only halfway (Class C), 4. Lowering the dumbbell only halfway (Class D) and 5. Throwing the hips to the front (Class E).

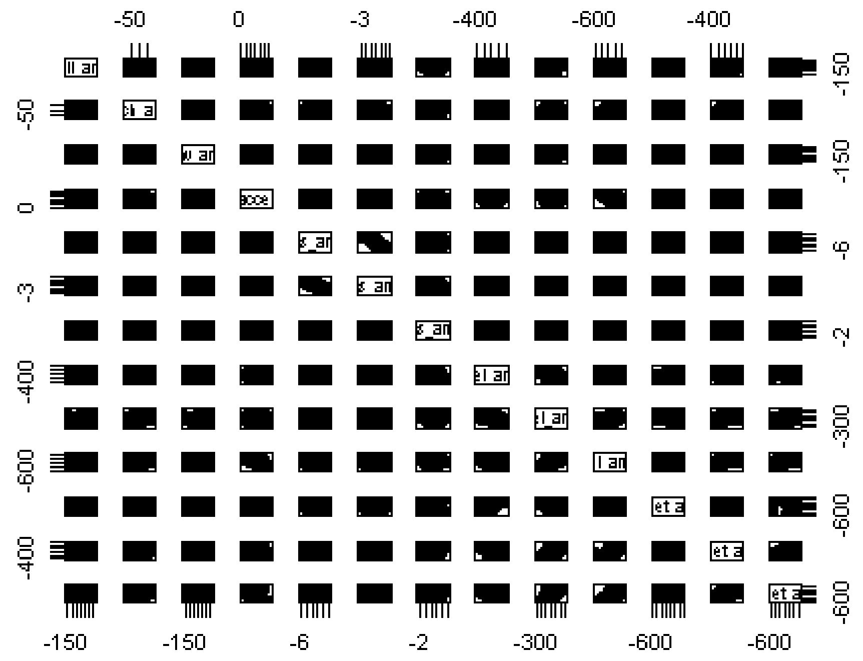
Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes in performing the exercise.
To Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3jWKhXZDo

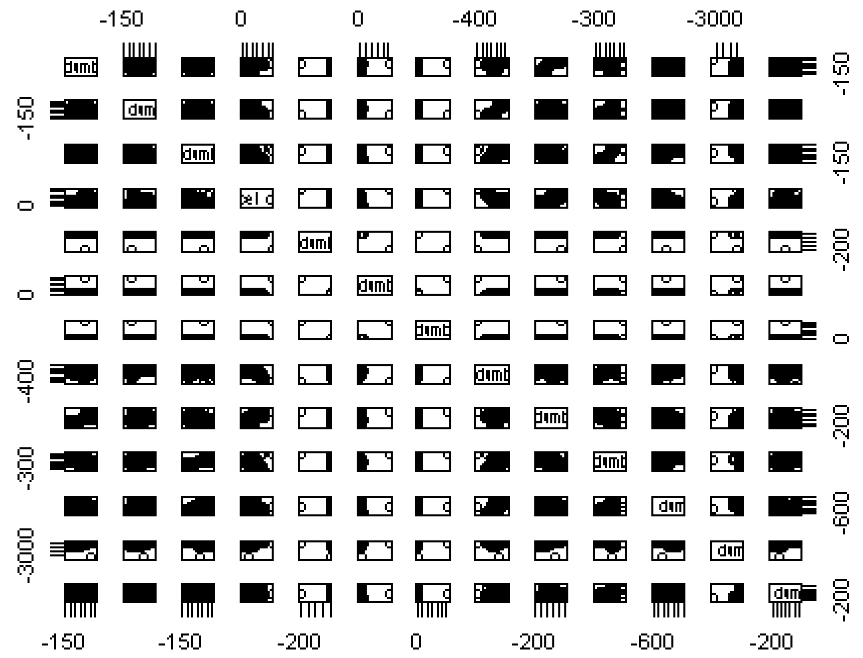
Clean the data for blanks and NAs

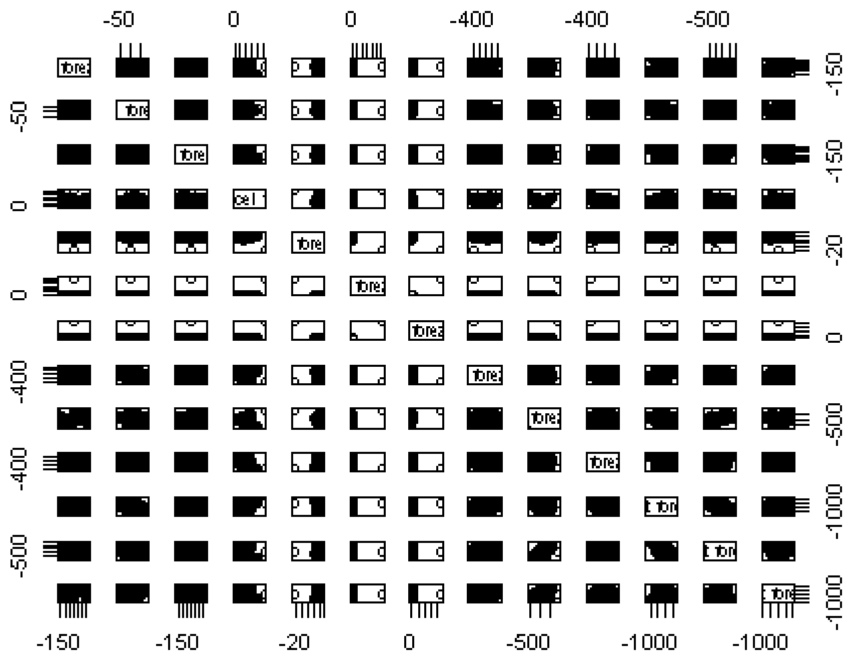
The data are processed to remove variables that are predomonantly blank or NA.

Correlation matrix plots give a rough idea about all combinations of variables' relationships. Examples are presented below:




-50
0
-3
-400
-600
-400
-150
-150
-6
-2
-300
-600
-600
-150
-150
-6
-2
-300
-600
-600
-50
0
-3
-400
-600
-400


-150
0
0
-400
-300
-3000
-150
-150
-200
0
-200
-600
-200
-150
-150
-200
0
-200
-600
-200
-150
0
0
-400
-300
-3000



Some variables appear to be primarily zero and thus will not improve via transformation. The gyro_forearm, y and z, are examples of this and are found in the 2nd plot.
Many more plots and descriptive statistics were run but not included in this report.

Features of Training data

All variables having NA or blank values are removed from the data sets. This leaves 53 variables for training, validating and testing the model.

```
inTrain <- createDataPartition(y=TrainingCln$classe, p=0.75, list=FALSE)
validation1 <- TrainingCln[-inTrain,]
training1 <- TrainingCln[inTrain,]
dim(validation1) ; dim(training1)

## [1] 4904   53

## [1] 14718    53
```

Experiments with several model types produced lesser quality predictions. The Random Forest method produced the most accurate prediction when validated. Only that result is presented below.

```
set.seed(10001)
TrainControl <- trainControl(method = "cv", number = 5, returnResamp = "all")
ModelfitRF <- train(classe ~ ., data=training1,  method="rf",
                    tuneLength = 1,
                    trainControl = TrainControl, ntree = 25)
ModelfitRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 25, mtry = param$mtry, trainControl =
..1)
##                Type of random forest: classification
##                      Number of trees: 25
## No. of variables tried at each split: 17
##
##         OOB estimate of  error rate: 1.11%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4175    6    2    2    0 0.002389486
## B   25 2809   11    2    1 0.013693820
## C    0   27 2522   17    1 0.017530191
## D    0    5   35 2368    4 0.018242123
## E    1    5    9   10 2681 0.009238729

ResultRF <- predict(ModelfitRF, newdata = validation1)
```

Model Validation Results

```
confusionMatrix(ResultRF, validation1$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    8    0    0    0
##          B    1  940    4    0    1
##          C    0    1  850    7    0
##          D    0    0    1  797    6
##          E    0    0    0    0  894
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9915, 0.996)
##     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            0.9993   0.9905   0.9942   0.9913   0.9922
## Specificity            0.9977   0.9985   0.9980   0.9983   1.0000
## Pos Pred Value         0.9943   0.9937   0.9907   0.9913   1.0000
## Neg Pred Value         0.9997   0.9977   0.9988   0.9983   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1917   0.1733   0.1625   0.1823
## Detection Prevalence   0.2859   0.1929   0.1750   0.1639   0.1823
## Balanced Accuracy      0.9985   0.9945   0.9961   0.9948   0.9961
```

The validated model produces a very accurate result per the confusion matrix summary. The 99.65% accuracy with a ±95% confidence interval of [0.9945, 0.998] and a p-value near zero. The out of sample error is 0.99%.

Prepare Test Data Set and Test the Model

The test sets' extra variables (having NA and blanks) must be removed as was done for the training set.

```
TestingNoNA <- testing
TestingNoNA[ testing == '' | testing == 'NA'] <- NA
TestNoNAList <- which(colSums(is.na(TestingNoNA))!=0)
TestingNoBlank <- TestingNoNA[, -TestNoNAList]
TestingCln <- TestingNoBlank[,-(1:7)]      #drop 1st 7 attribute\timestamp
columns
```

Running the Random Forest Model to predict the test data set

The model is used to predict the exercise variation (TestingCln$classe) for the test set. The result is then displayed.

```
Resulttest <- predict(ModelfitRF, newdata = TestingCln)
Resulttest

##  [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
```

The test data set result is shown.

Conclusion

Based on the validated model result, the Random Forest method prduced a very accurate prediction with low out-of-sample error.