Practical Machine Learning Course Project: Predicting proper exercise from wearable accelerometer sensor data

Jeffrey Thatcher

Saturday, February 27, 2016

1.0 Background

Machine learning model based on the exercise sensor data collected by Velloso et al, 2013.[1] Simply using the raw accelerometer data a prediction model was built to correctly classify accelerometer data from 20 Test cases. The outcome variable levels are A, B, C, D, and E. A comes from exercise that was performed correctly while B - E were from the same exercises that were performed incorrectly. The goal of the algorithm is to correctly classify how each person in the test set was performing their exercise.

[1] Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th Augmented Human (AH) International Conference in cooperation with ACM SIGCHI (Augmented Human'13) . Stuttgart, Germany: ACM SIGCHI, 2013.

2.0 Methods

2.1 Raw Sensor Data

The raw data was loaded and, by using the time stamp variables, a plot of the accelerometer data is generated. The data represents the sensor data from a person doing a biceps curl correctly (Fig.1 - green) or incorrectly (Fig.1 - red).

2.2 Plot of sensor data from one user

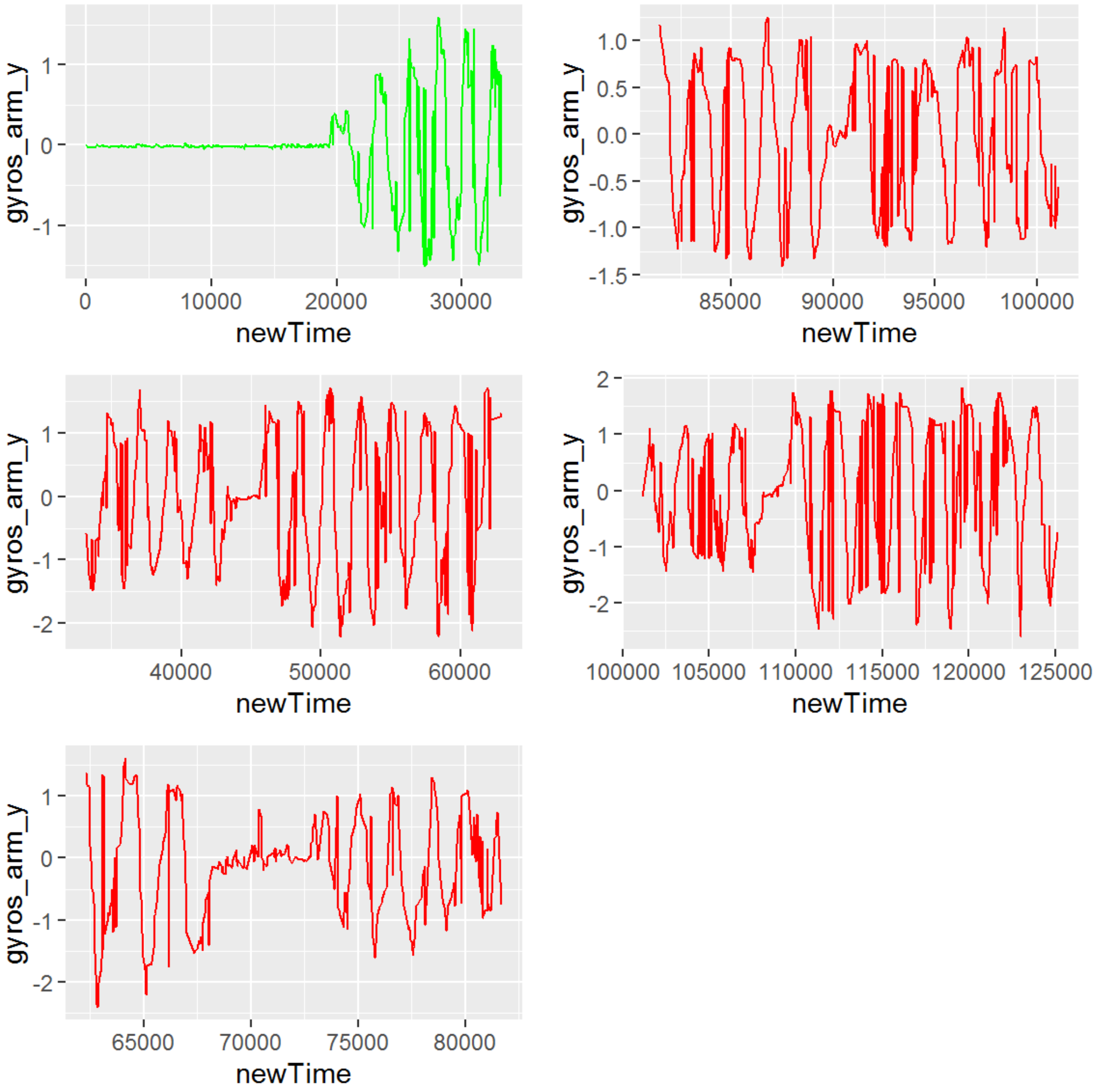


Figure 1. This figure represents the data collected from the `gyros_arm_z` sensor taken from the exerciser Carlitos while doing a biceps curl in five different ways. (green) correct curl body movement; (red) incorrect body movement.

2.3 Cleaning Data

To clean the data first, the column with no predictive values were removed, such as the time stamp and row number columns. Following that, columns that were empty and those that contained missing data (NA) were removed.

```
###Eliminate columns with missing and unrelated
cleanData <- function(dataframe){

        #Eliminate columns with unrelated data
        colToRemove <- c(grep("X", colnames(dataframe)),
                         grep("user_name", colnames(dataframe)),
                         grep("raw_timestamp_part_1", colnames(dataframe)),
                         grep("raw_timestamp_part_2", colnames(dataframe)),
                         grep("cvtd_timestamp", colnames(dataframe)),
                         grep("new_window", colnames(dataframe)),
                         grep("num_window", colnames(dataframe)),
                         grep("time", colnames(dataframe)),
                         grep("newTime", colnames(dataframe)),
                         grep("exerciseID", colnames(dataframe)),
                         grep("windowID", colnames(dataframe)))
        dataframe <- dataframe[,-colToRemove]

        #Eliminate columns with NA values
        dataframe <- dataframe[, colSums(is.na(dataframe)) == 0]

        return(dataframe)
}

trainData <- cleanData(trainData)
quizData <- cleanData(quizData)
```

2.4 Data Partitions

Once the sensor data was cleaned it was then prepared for predictive modeling. To prepare the data, the training dataset was separated into “Training” and “Test” datasets using the `caret` package’s `createDataPartition` function.

```
#Separate training and test data
# We will remove one person from each level of the variable `classe` as the test set
set.seed(1000)
trainIndex = createDataPartition(trainData$classe, p = 0.60,list=FALSE)
training = trainData[trainIndex,]
testing = trainData[-trainIndex,]
```

3.0 Machine Learning

Three predictive models were trained including: linear discriminant analysis (LDA); decision Tree; and bagged decision tree. Accuracy was determined by cross-validation for each of the three models and then we applied the model to the “testing” data set that was partitioned in section *2.4 Data Partitions*. The final model was selected based on the “testing”" data set accuracy. Testing data results are presented as a confusion matrix using the `confusionMatrix` function from the `caret` package.

3.1 Linear Discriminant Analysis

```
### Train lda Classifier
#fit model
Lda <- train(classe ~ ., method="lda", data=training)

#predict test data
pred <- predict(Lda, testing)
confusionMatrix(testing$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1837   49  159  176   11
##          B  242  942  222   49   63
##          C  141  111  889  185   42
##          D   75   62  173  930   46
##          E   54  262  124  158  844
##
## Overall Statistics
##
##                Accuracy : 0.6936
##                  95% CI : (0.6833, 0.7038)
##     No Information Rate : 0.2994
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6121
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7820   0.6606   0.5673   0.6208   0.8390
## Specificity            0.9281   0.9103   0.9237   0.9439   0.9126
## Pos Pred Value         0.8230   0.6206   0.6499   0.7232   0.5853
## Neg Pred Value         0.9088   0.9235   0.8953   0.9134   0.9747
## Prevalence             0.2994   0.1817   0.1997   0.1909   0.1282
## Detection Rate         0.2341   0.1201   0.1133   0.1185   0.1076
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8551   0.7854   0.7455   0.7824   0.8758
```

3.2 Trees

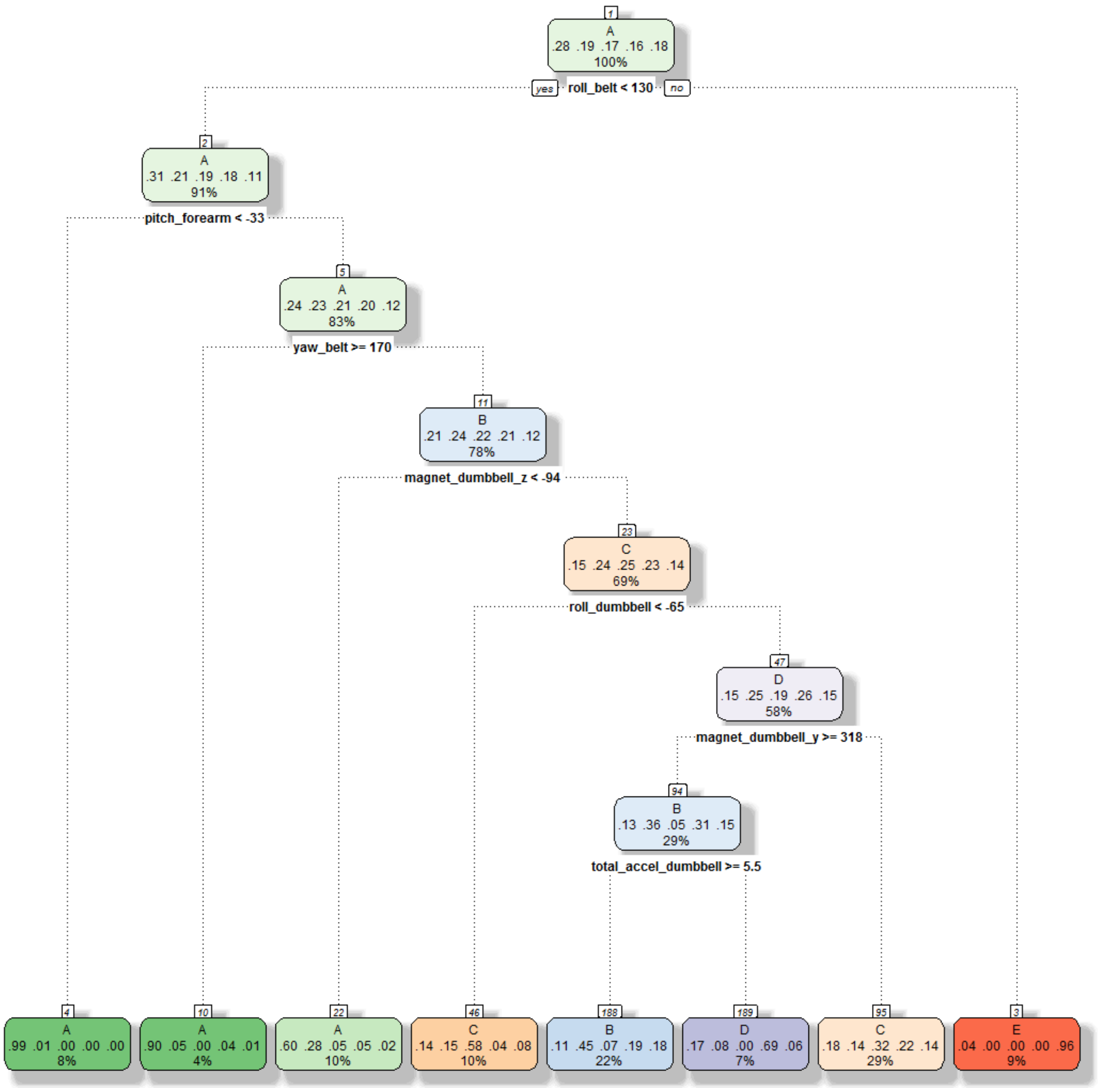
```
library(rattle)

#fit model
tree <- train(classe ~ ., method="rpart", data=training)

#predict test data
pred <- predict(tree, testing)
confusionMatrix(testing$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1330  196  563  111   32
##          B  239  767  463   49    0
##          C   32  113 1223    0    0
##          D   73  343  529  341    0
##          E   17  331  394   28  672
##
## Overall Statistics
##
##                Accuracy : 0.5523
##                  95% CI : (0.5412, 0.5633)
##     No Information Rate : 0.4043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4386
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7865  0.43829   0.3856  0.64461  0.95455
## Specificity            0.8535  0.87680   0.9690  0.87085  0.89219
## Pos Pred Value         0.5959  0.50527   0.8940  0.26516  0.46602
## Neg Pred Value         0.9357  0.84466   0.6991  0.97134  0.99500
## Prevalence             0.2155  0.22304   0.4043  0.06742  0.08973
## Detection Rate         0.1695  0.09776   0.1559  0.04346  0.08565
## Detection Prevalence   0.2845  0.19347   0.1744  0.16391  0.18379
## Balanced Accuracy      0.8200  0.65755   0.6773  0.75773  0.92337
```

```
fancyRpartPlot(tree$finalModel)
```



Rattle 2016-Feb-27 23:40:39 jeffthatcher

3.3 Bagging

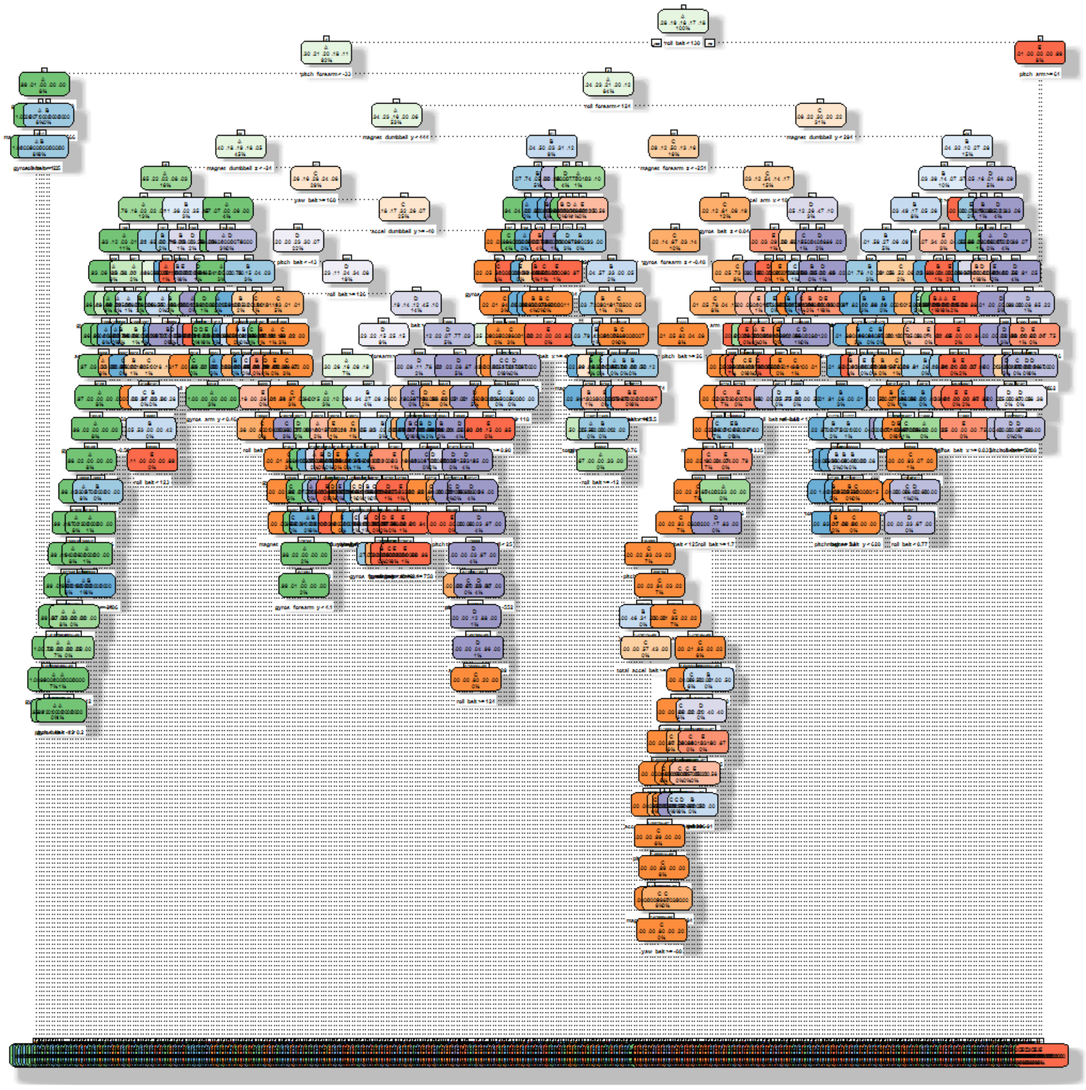
```
#fit model
bagging <- train(classe ~ ., method="treebag", data=training, nbagg = 25)

#predict test data
pred <- predict(bagging, testing)
confusionMatrix(testing$classe, pred)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    6    0    0    0
##          B   26 1477   14    1    0
##          C    1   15 1346    6    0
##          D    1    2   32 1250    1
##          E    0    0    3    8 1431
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9852          
##                  95% CI : (0.9823, 0.9878)
##     No Information Rate : 0.2873          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9813          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9876   0.9847   0.9649   0.9881   0.9993
## Specificity            0.9989   0.9935   0.9966   0.9945   0.9983
## Pos Pred Value         0.9973   0.9730   0.9839   0.9720   0.9924
## Neg Pred Value         0.9950   0.9964   0.9924   0.9977   0.9998
## Prevalence             0.2873   0.1912   0.1778   0.1612   0.1825
## Detection Rate         0.2837   0.1882   0.1716   0.1593   0.1824
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9933   0.9891   0.9807   0.9913   0.9988
```

```
fancyRpartPlot(summary(bagging)$mtrees[[25]]$btree)
```

```
## 
## Bagging classification trees with 25 bootstrap replications
```



Rattle 2016-Feb-28 00:03:45 jeffthatcher

Figure 3. One of the 25 trees built in the bagged trees model that was aggregated to form the final model.

4.0 Results

Final results in predicting the hold-out “testing” data set show that LDA was 69% accurate, the decision tree was 55% accurate, and the bagged trees model was 99% accurate.

The bootstrap aggregated decision tree model built using the treebag method in the `caret` function performed best of the three models tested. The tuning parameter `nbagg` was set to 25, which means the model is the result of aggregating 25 decision trees generated from bootstrapped training data. The model may perform better with more aggregates, the consequence of which may be even more reduced variance in the model. However, it is likely that the tradeoff between processing time to generate the model and increase in accuracy is not valuable considering the model is already very accurate anf it takes about 15 min to generate the model on a standard laptop computer (Core i7, 8GB RAM).