Practical-Machine-Learning-July-2014-LeifUlstrup.Rmd

Leif Ulstrup

July 24, 2014

Executive Summary

The purpose of this project is to demonstrate the analysis and model building process of machine learning tools and techniques. The subject of this analysis and modeling exercise is a set of experimental data gathered from Brazilian university research team that have instrumented six test subjects with various motion sensors while they perform an exercise with a barbell using five techniques that range “perfect form” (A) to “very poor form” (E). The question these researchers are exploring is whether machine learning techniques can be used on the data they collected to predict whether a person is using proper form in their exercise or not. The ultimate vision of the team is to provide real-time feedback to those performing exercises to improve the quality of their exercise experience.

The results of the modeling below confirm that this is possible with high accuracy.

Context

The details of the experiment can be found here along with their research paper: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see section “Weight Lifting Exercises Dataset”“). See this paper: http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201 (http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201).

The A,B,C,D,E barbell lifting form type model is described in this excerpt from their paper: “…Participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in Five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and 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…”

Model Development

How the Model was Built

The project source data for this project consists of two data sets. One set (“trainingFile”) contains 19622 observations with 160 variables (columns in the matrix). The test (“testFile”) set contains 20 observations and also 160 variables.

The initial steps included studying the data in the training and test files through visual inspection of the files using head() and names() to get my bearings for further analysis. I then used the nearZeroVar() function on the two sets of data. What I discovered is that the trainFile had 60 of 160 columns reporting near zero values (NZV) and the test file had 101 of 160 with NZV. I also saw that the last column of the files noted the type of workout classifier ("classe") and that the first 7 columns of the file included information about the test subject, time of the test etc. that did not seem useful for the nature of this modeling (though may be useful in terms of analyzing the quality of the training inputs and the test subjects) and I excluded those from the data sets as part of the clean up process.

Since the testFile only used 101 columns of test inputs, I chose to immediately use that information to narrow the trainFile to the same set of columns and cleaned up both data sets so that only the same 53 columns where in use (53 = 160-100-7). I now had two smaller data files to work with ("trainingClean" and "testClean").

Important processing note: I also later discovered I had to apply the as.numeric() function to the numeric input fields and the factor() function to the "$classe" column due to errors that I got in the later stages using the machine learning processing tools. Finding the root cause of this processing problem and fix was very time consuming.

How Cross-validation was Used

The next step I followed in the machine learning model development process ("study design" and "cross-validation") was use of the createDataPartition() function from the caret package to randomly sample the trainingClean file and create a subset (70% of the total) for training purposes and another set for testing of the training (30%). I now had two data sets that I could use for machine learning model development - one for training the model and one reference set for testing derived from the training set.

Expected Sample Error Calculations

The next step was to follow the guidance from the JHU course on model development and pre-process the training data using the caret preProcess() function, being careful to exclude the "classe" column of factors with the measures A through E of barbell training form since the preProcess function expects all numeric values. I chose the method = "pca" for principal component analysis in the pre-processing stage in the hopes that this would distill the data to a more compact form given the 52 input variables and thus accelerate the learning process of the train() model development stage. I chose the "random forest" technique for model development based on a combination of the lecture overview of its utility, forum discussions, and independent text research.

Below are the results from the model that I chose to use for this project based on using the confusionMatrix() function with the reference data (testing set) derived from the training file compared to the predictions from the model. This includes the magnitude of the in-sample error rates. The accuracy measures were seemingly high at 97.5% and the reference A-E vs predicted A-E were tightly clustered around the diagonal for most of the measures. The model did have some trouble differentiating some of the predicted results; however, those readings are mostly concentrated in a small number of incorrect predictions. There is likely similarity between certain forms of the exercise and either better modeling techniques are better sensor placement may be needed to improve the differentiation and the prediction accuracy.

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 1667 2 3 1 1 B 27 1088 23 0 1 C 0 14 1003 9 0 D 3 1 37 921 2 E 0 2 10 10 1060

Overall Statistics

```
            Accuracy : 0.9752
              95% CI : (0.9709, 0.979)
 No Information Rate : 0.2884
 P-Value [Acc > NIR] : < 2.2e-16

               Kappa : 0.9686
```

Mcnemar's Test P-Value : 1.242e-09

Statistics by Class:

```
                     Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.9823 0.9828 0.9322 0.9787 0.9962 Specificity 0.9983 0.9893 0.9952 0.9913 0.9954 Pos Pred Value 0.9958 0.9552 0.9776 0.9554 0.9797 Neg Pred Value 0.9929 0.9960 0.9850 0.9959 0.9992 Prevalence 0.2884 0.1881 0.1828 0.1599 0.1808 Detection Rate 0.2833 0.1849 0.1704 0.1565 0.1801 Detection Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839 Balanced Accuracy 0.9903 0.9861 0.9637 0.9850 0.9958

Reasoning Behind Choices Made

For this first project using these techniques, I chose to stick with the recommended pre-processing and model building techniques that were recommended in the class and literature I reviewed. Also, I spent a considerable amount of time in the early stage of the project getting a better understanding of the data sets, cleaning them up, and also solving unexpected warnings and errors that thwarted my model building efforts. The processing time of ~12 minutes using 4 cores of a 3.4GHz OSX machine with 32GB of RAM for each model build was also a factor. I found on the forum discussion boards the recommended use of the doMC package and library to parallelize the analysis. This helped to speed the model development and testing; however, I ran out of time to explore additional model development to improve predictions.

I used my model on the 20 record test file set that was included in the course materials and is a required submission. The model correctly predicted 18 of 20 values on the first pass. I then used the confusion matrix results to prioritize the likely off diagonal results to predict the next most likely value. I was able to solve one of the two incorrect predictions on the first attempt and solve the second incorrect answer on the second attempt. I think there are numerous opportunities to both speed the processing of this data by reducing the number of input columns and also improve the accuracy of the predictions by diving deeper into which inputs cause the model to have trouble differentiating the correct prediction. All of this can be the subject of subsequent research.

The work by this Brazilian research team is very promising for their potential application of automated fitness coaching applications.

Link to Predictions

Here is the link to the public github site for the materials I used in developing this project, Rmd code, and the test set of 20 results expressed in *.txt files: https://github.com/leifulstrup/Practical-Machine-Learning-Project (https://github.com/leifulstrup/Practical-Machine-Learning-Project)

R Code

```
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
```

```
##
## The downloaded binary packages are in
##  /var/folders/g8/y4h6qnks65d1jtgt62yj7hr80000gn/T//RtmpRKPs5g/downloaded_packages
```

```
## Loading required package: doMC
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Warning: NAs introduced by coercion
## Random Forest
##
## 13737 samples
##    24 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     1         1      0.004        0.005
##   10    1         0.9    0.004        0.004
##   20    0.9       0.9    0.004        0.004
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1666    3    2    2    1
##          B   23 1104   10    0    2
##          C    2   14 1000    9    1
##          D    4    1   53  905    1
##          E    0    6   10    5 1061
##
## Overall Statistics
##
##                Accuracy : 0.975
##                  95% CI : (0.97, 0.979)
##     No Information Rate : 0.288
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.968
##  Mcnemar's Test P-Value : 1.53e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.983    0.979    0.930    0.983    0.995
## Specificity             0.998    0.993    0.995    0.988    0.996
## Pos Pred Value          0.995    0.969    0.975    0.939    0.981
## Neg Pred Value          0.993    0.995    0.985    0.997    0.999
## Prevalence              0.288    0.192    0.183    0.156    0.181
## Detection Rate          0.283    0.188    0.170    0.154    0.180
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.990    0.986    0.962    0.985    0.995
```