Human Activity Recognition

Christopher Weiss

December 18, 2016

Executive Summary

Following from the paper *Qualitative Activity Recognition of Weight Lifting Exercises* by Velloso, et. al, this paper attempts to find a suitable model for predicting whether someone is doing a given exercise correctly based upon measurements taken during the exercises. When the exercise is done incorrectly, the type of errror is classified as to the type of defect. Details can be found here: http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf

Generating The Model

Downloading The Data

The data comes from the paper referenced above.

```
orig_training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
orig_testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

Cleaning The Data

The initial dataset has four diffferent categories of variables:

1. **Clarification Variables** – These are not measured input values, but serve to help navigate the data (such as new_window, and the timestamp data)
2. **Raw Input Data** – The actual data being recorded, in real time, during the experiments. This is the most interesting data from our perspective.
3. **Calculated Data** – Values that are calculated from the raw data. Most rows of this data are either empty, N/A, Div/0, or something like this. These will be ignored.
4. **Classification Variable** – Specifically, classe, which is our variable of interest. This variable returns the observed quality of the exercise performed, which is what we are trying to predict.

First, I shall remove the variables that are not numeric (calculated data and the output data), and the clarification variables (which occur in the first seven columns).

```
cont_meas <- vector()
for(i in 1:ncol(orig_training)){
  cont_meas[i] <- TRUE
  if(i <= 7){
    cont_meas[i] <- FALSE
  }
  if(class(orig_training[,i]) == "factor"){
```

```r
    cont_meas[i] <- FALSE
  }
  if(sum(is.na(orig_training[,i]))>0){
    cont_meas[i] <- FALSE
  }
}

cont_meas_d <- subset(orig_training, select=cont_meas)
sub_training <- data.frame(cont_meas_d, classe=orig_training$classe)
dim(sub_training)
```

```
## [1] 19622    53
```

The resulting dataframe has 52 input variables and the output variable classe.

Partition The Data

In order to be able to validate the accuracy of the model, I will partition the given test set into a training and a test set. I will use 60% of the data as the training set, and 40% as the internal testing set.

```r
set.seed(53593)
inTrain = createDataPartition(sub_training$classe, p = 3/5)[[1]]
training = sub_training[inTrain,]
testing = sub_training[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 11776    53
```

```
## [1] 7846   53
```

Select and Build Model

Because this is a classification problem, and because we are not working with a small dataset, I will use a randomForest model.

```r
set.seed(53593)
model <- randomForest(classe ~ ., data = training)
model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    2    0    0    0 0.0005973716
## B   12 2261    6    0    0 0.0078982010
## C    0   12 2035    7    0 0.0092502434
## D    0    0   23 1906    1 0.0124352332
## E    0    0    2    6 2157 0.0036951501
```

We can see that this model performs pretty darn well, with an error rate of .6%.

It is important to note that this is the "Out of Bag" error rate, which is an unbiased estimator of the error, according to the documentation:

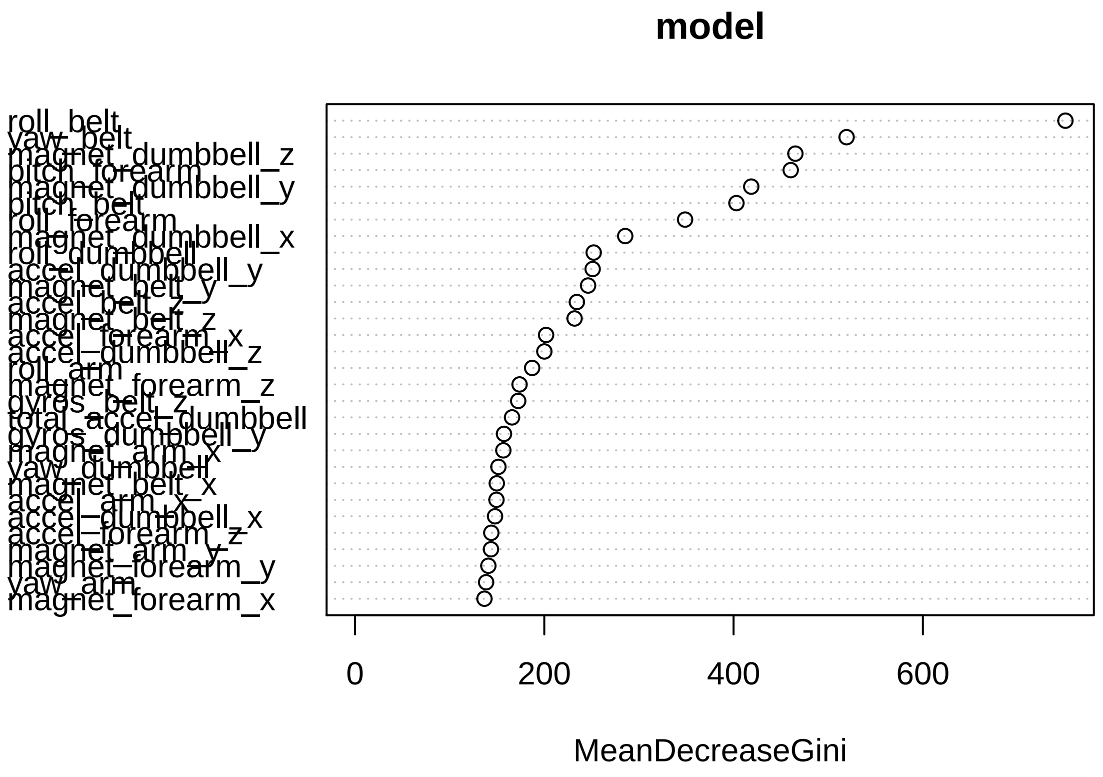
http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr

Therefore, I will not separately perform cross validation to measure the expected error rate, but take the .6% value.

Key variables

One of the most important questions on model construction is which of the variables are really key inputs. Here I will graph the inputs that are most important in the model.

```
varImpPlot(model)
```



This is a bit difficult to read, so I will list the top 12 here:

```
i <- varImp(model)
vardf <- data.frame(var=rownames(i),i)[order(-i),]
```

```r
ivar <- as.vector(vardf$var[1:12])
ivar
```

```
##  [1] "roll_belt"          "yaw_belt"           "magnet_dumbbell_z"
##  [4] "pitch_forearm"      "magnet_dumbbell_y" "pitch_belt"
##  [7] "roll_forearm"       "magnet_dumbbell_x" "roll_dumbbell"
## [10] "accel_dumbbell_y"  "magnet_belt_y"      "accel_belt_z"
```

Calculate Error On Internal Testing Data

Here I will use the model to predict the values of the internal testing data, and then calculate the error rate here, and see how it compares to the OOB estimate of the error (.6%).

```r
pred <- predict(model, newdata = testing)
conmat <- confusionMatrix(pred, testing$classe)
conmat$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   12    0    0    0
##          B    2 1505   15    0    0
##          C    0    1 1350   13    1
##          D    0    0    3 1272    5
##          E    1    0    0    1 1436
```

```r
conmat$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9931175      0.9912930      0.9910293      0.9948256      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

This compares favorably, with the lower end of the confidence interval estimate being ~1% error rate.

In order to see if we are overfitting the data, I am going to run the same series of estimates, but this time only using the top half of the variables, by order of decreasing Gini importance, and re-run another model, and test the accuracy of the prediction on the internal test set.

```r
ivar2 <- as.vector(vardf$var[1:(nrow(vardf)/2)])
training2 <- subset(training, select=ivar2)
training2 <- data.frame(training2, classe=training$classe)
set.seed(53593)
model2 <- randomForest(classe ~ ., data = training2)
model2
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
```

```
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    2    1    1    0 0.001194743
## B   15 2252   12    0    0 0.011847301
## C    0   14 2035    5    0 0.009250243
## D    0    0   19 1909    2 0.010880829
## E    0    0    2    9 2154 0.005080831
```

```
pred2 <- predict(model2, newdata = testing)
conmat2 <- confusionMatrix(pred2, testing$classe)
conmat2$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   14    0    0    0
##          B    2 1499   14    1    0
##          C    0    5 1351   13    3
##          D    0    0    3 1272    7
##          E    1    0    0    0 1432
```

```
conmat2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9919704      0.9898418      0.9897382      0.9938245      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

The second model has an out of bag estimate of the error rate of .85%, which is worse than the initial model, but it actually has a better fit on the internal training data. Because each of the two models has a confidence interval of the expected error that contains the other error rate, I am considering them to be equally accurate. And based upon the idea that a simpler model is better than a more complex one, for a given level of accuracy, I will use the model with only half of the input variables.

Predict Using External Data

Now I will use the model that has been generated to make 20 predictions, using the data provided in the original testing data set.

Data Cleaning on Testing Data

In order to input the testing data, the same variables that are used in the

```
ex_testing <- subset(orig_testing, select=ivar2)
dim(ex_testing)
```

```
## [1] 20 26
```

Predictions Using Testing Data

```r
pred_ex_testing <- predict(model2, newdata = ex_testing)
pred_ex_testing
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```