# Practical Machine Learning Final Project

Darren Whitwood

2023-03-23

## Predictive Model For Exercise Data

## Executive Summary

In this report, we will develop a model for predicting which activity was performed based on the other columns in the data from Groupware @ LES found here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

We will clean the data, perform some exploratory analysis, and then train a GBM model which we will then validate against 20% of the data set aside at the outset. Based on this model predicting with over 95% accuracy, we will conclude that this is a powerful model that can be used to predict the testing cells (20 rows of data that are a separate part of the assignment and not reported on here).

## Getting and Cleaning the Data

First read the data from the CSV files.

```
require(caret)
require(dplyr)

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

When looking at the data at a high level, two things jump out. First, the first 7 columns are administrative data; it's possible that the user, timestamp, and other data here correlates with the activity but the fact that a certain test subject did activity A more often has no predictive power for the general population of wearable tech users. Second, a very large percentage of the data is NA.

Hence we will remove the first 7 columns, but also while we are at it, let's remove all of the "stddev" columns as well. The reader may verify that all isntances of a "stddev" column are next to a corresponding "var" column but not vice versa, so nothing is lost from removing these columns, as variance and standard deviation are each a function of the other.

```
names(pml.training[1:7])
columnsToRemove <- c(1:7)
pml.training <- pml.training[-columnsToRemove] %>%
    select(-contains("stddev"))
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

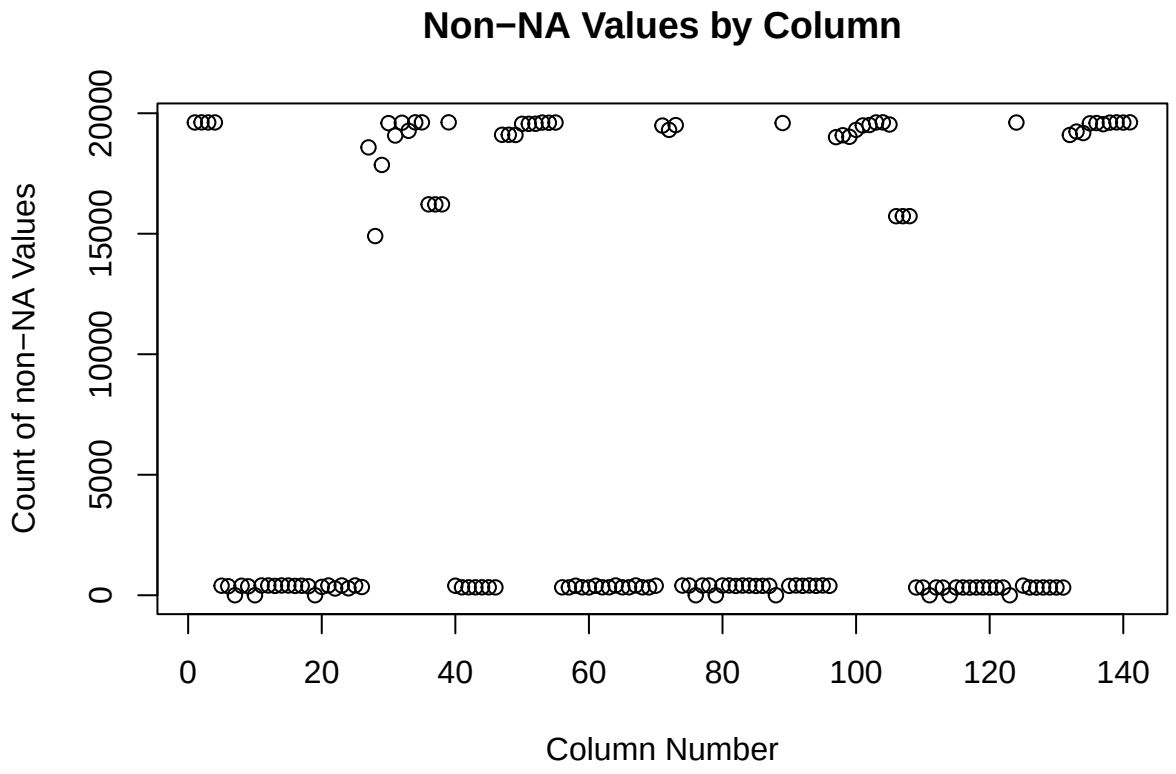
```
# By converting all columns besides the 'classe' at the right end, blanks and
# divide by 0 are coerced to NA.
pml.training[-c(length(pml.training))] <-
  pml.training[-c(length(pml.training))] %>% mutate_if(is.character, as.numeric)
pml.training[-c(length(pml.training))] <-
  pml.training[-c(length(pml.training))] %>% mutate_if(is.integer, as.numeric)

sum(is.na(pml.training)) / (nrow(pml.training)*ncol(pml.training))
```

```
## [1] 0.6124657
```

The simplest interpretation of the NA values is that they represent a motion being irrelevant to the activity that the person is performing. That is the same as having a 0 value for all columns, or at least that is a workable assumption that will play out in the prediction power of the model.

```
pml.training[is.na(pml.training)] <- 0
```

As a result we want to explore how the NA values are distributed. The following exploratory plot shows how there is a clear divide where some activities have values for nearly every row and others for very few rows. We will tentatively remove the columns that are NA for most rows, recognizing that on the one hand there might be a clear division where there is a correlation between the columns that have many (or few) NA values with certain activities but on the other hand that we might still have sufficient predicting power without them.



The way we will implement this is to create a new data frame that removes columns that are more than 2.5% NA values. There is significant wiggle room for the factor given the divide in the middle of the above plot.

```
pml.training.trim <- pml.training[, colSums(pml.training != 0) > 0.025 * nrow(pml.training)]
```

The last preparatory step is to divide the data into training and validation, since we have such a small test set (20 rows) that does not have correct answers to test against. Using the convention of 60/40 for training/validation, we partition as follows:

```
set.seed(2023-3-23)
inTrainig <- createDataPartition(pml.training.trim$classe, p = 3/5)[[1]]
pml.validation.trim <- pml.training.trim[-inTrainig,]
pml.training.trim <- pml.training.trim[inTrainig,]
```

#Model Creation

Let's take stock of our data objects now.

```
dim(pml.validation.trim)
```

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

```
dim(pml.training.trim)
```

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

With 53 columns, that means 52 predictors for a training set of 9421 rows. A single tree with so many predictors would help tell a story, at least at the top branches of the tree, and would be good for interpretation. However, this data set is still small enough to use a more sophisticated method, and the loss of explanatory power is minimal because a tree with so many variables will always be hard to interpret anyway. Rather than compromise by selecting fewer columns, we'll use gradient boosting so that all 52 predictors can be preserved and the observations can be systematically weighted.

```
gbm <- train(classe ~ ., method="gbm", data=pml.training.trim, verbose=FALSE)
```

To validate this model we can create a confusion matrix on the training data.

```
pred.train <- predict(gbm, pml.training.trim)
cm.train <- confusionMatrix(pred.train, as.factor(pml.training.trim$classe))
cm.train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3314   52    0    2    2
##          B   26 2190   46    7    8
##          C    7   36 1990   58   11
##          D    0    1   16 1858   17
##          E    1    0    2    5 2127
##
## Overall Statistics
##
##                Accuracy : 0.9748
##                  95% CI : (0.9718, 0.9775)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9681
##
##  Mcnemar's Test P-Value : 9.291e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9898   0.9609   0.9688   0.9627   0.9824
## Specificity            0.9934   0.9908   0.9885   0.9965   0.9992
## Pos Pred Value         0.9834   0.9618   0.9467   0.9820   0.9963
## Neg Pred Value         0.9960   0.9906   0.9934   0.9927   0.9961
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2814   0.1860   0.1690   0.1578   0.1806
## Detection Prevalence   0.2862   0.1934   0.1785   0.1607   0.1813
## Balanced Accuracy      0.9916   0.9759   0.9787   0.9796   0.9908
```

The 97.64% accuracy on the training data is excellent, indicating that we have a model that can reliably predict the activity type. Having settled on a model, we predict on the 40% of original rows that were set aside for validation, to measure the predictive power on new data.

```
pred <- predict(gbm, pml.validation.trim)
cm <- confusionMatrix(pred, as.factor(pml.validation.trim$classe))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2196   53    0    0    2
##          B   25 1421   37    2    8
##          C    8   42 1318   49   10
##          D    3    2   12 1227   17
##          E    0    0    1    8 1405
##
## Overall Statistics
##
##                Accuracy : 0.9644
##                  95% CI : (0.9601, 0.9684)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.955
##
##  Mcnemar's Test P-Value : 5.246e-10
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9361   0.9635   0.9541   0.9743
## Specificity            0.9902   0.9886   0.9832   0.9948   0.9986
## Pos Pred Value         0.9756   0.9518   0.9236   0.9730   0.9936
## Neg Pred Value         0.9936   0.9847   0.9922   0.9910   0.9942
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2799   0.1811   0.1680   0.1564   0.1791
## Detection Prevalence   0.2869   0.1903   0.1819   0.1607   0.1802
## Balanced Accuracy      0.9870   0.9624   0.9733   0.9745   0.9865
```

# Conclusion

The above confusion matrix indicates 96.44% accuracy for this model, which is sufficient to pass a reasonable 95% accuracy target. Thus a GBM on the selected columns using the methodology detailed in this report produces a viable model to predict the activity level for future scenarios. This accuracy can be interpreted as the out-of-sample error will be 1-.9644 = 3.56%.

Using this model we can predict the 20 test cells:

```
predFinal <- predict(gbm, pml.testing)
predFinal
```

```
##  [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
```