Practical Machine Learning Project

J.E.Black

July 22, 2016

Executive Summary

We obtained a rather large collection of data detailing measurments taken as people performed weight lifting exercises, using five different methods, both correct and incorrect. The particpants wore various instruments to allow applicable data to be collected.

We used practical machine learning techniques to analyze the data, determine reasonable methods for performing such analysis, and developed a prediction model which, given certain specific metrics, could predict the method used with an accuracy exceeding 95%, using only the "top 5" most significant variables collected.

Further analysis to determine and select additional significant covariables is expected to improve the accuracy of our model. From our analysis, we would expect that inclusion of four additional covariables would push the accuracy rate to approximately 99% or better.

Background and analysis details follow.

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

Our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of six participants to predict the manner in which they did the exercise (this is the "classe" column in the training data). Each was asked to perform barbell lifts correctly and incorrectly in five different ways. That data was recorded and made available to us as described below.

Exploratory Data Analysis

We acknowledge and thank the group who present their research on the website at: http://groupware.les.inf.puc-rio.br/har, from which we obtained the the data used in our analysis. They have been very generous in allowing us to use their data.

- The training data used are are available from: "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

- The test data are available from: "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

We quote the authors' description of the data:

```
Six young health 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.
```

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4F6NogDKh

Preparation

In order to set up the environment such that we can create reproducible results, we need to load the necessary R libraries, and initialize a few parameters (not shown).

Obtain Data

First check to see if we've already downloaded and saved a copy of the data; if not, then download it from the public source in the internet (not shown).

Pruning and Cleaning

Preliminary examination of the data indicated that there were numerous occurances of empty fields, and strings such as "NA" and "#DIV/0!" that were indicative of problems with the data.

We'll read the data from the "csv" files, and adjust the problem fields; even in the test data, although the anomolies were observed only in the training data. We just do this to make the data more uniform, so that it's easier to reduce the number of predictors in later steps (not shown).

Before we continue with additional cleaning of the data, we'll perform a few simple "sanity checks;" such as making sure that both data frames have the same number of columns, and that the columns have the same names in each data frame, except for "classe" and "problem_id" which are special cases applicable to the exercise (not shown).

If we make it this far, then both the test data and the training data are reasonably uniform, and we can continue processing. For now, we'll set the Test Data aside, and work with the Training set.

We'll attempt to reduce the number of predictors by removing the columns that we don't expect to influence the prediction, and then we'll split the training data into a training/modeling and a validation data set.

Currently we have 160 variables; we should be able to prune that down a bit by taking out column that have near zero variance, and thus would not influence the prediction very much.

```
nZ <- nearZeroVar(trnFrame)
trnFrame2 <- trnFrame[, -nZ]
```

Now we have only 124 columns. Let's remove columns that are mostly "NA;" any column that is 90% or more "NA" is not likely to influence the prediction.

```
mostlyNA <- sapply(trnFrame2, function(x) mean (is.na(x))) > 0.9
trnFrame2 <- trnFrame2[, mostlyNA==FALSE]
```

By removing the "mostly NA" columns, we've reduced the count to 59.

Although it is possible that the time of day that the exercise was performed could have some influence on the outcome, we don't consider that it would have a significant effect on the prediction model. Similarly, we couldn't justify that a window would affect the outcome. Therefore, in order to further reduce the number of predictors, we'll remove columns 1 through 6, which appear to be a sample number, the subject's name, a couple of time stamps, and "num_window." (not shown)

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

Now we've reduced the number of variables to 53; however we still have 19622 rows in the data set. We'll reduce that count when we split the "trnFrame2" data into two separate sets; one for training, the other for validation.

Creating a Cross Validation set

We'll create a cross validation data set for comparison to the model we'll create using the training data subset.

A common ration for training / testing / validation data sets is about 70% / 15% / 15% respectively.
Our usual preference is a 60% / 40% split for training vs validation. However in this case, the testing data set contains only 20 observations, while the original training data set is almost three orders of magniture larger. Even a 15% cut is still about two orders of magniture larger than the test data. Given the relative sizes, and the fact that we're not comfortable with a split much greater than 70% / 30%, we will arbitrarily partition the testing data using a random 70% sample to use for model building; and using the remaining 30% for validation / accuracy measurement.

```
set.seed(randomSeed)
nSplit <- createDataPartition(y=trnFrame2$classe, p=0.70, list=FALSE)
modData <- trnFrame2[nSplit,]
valData <- trnFrame2[-nSplit,]
```

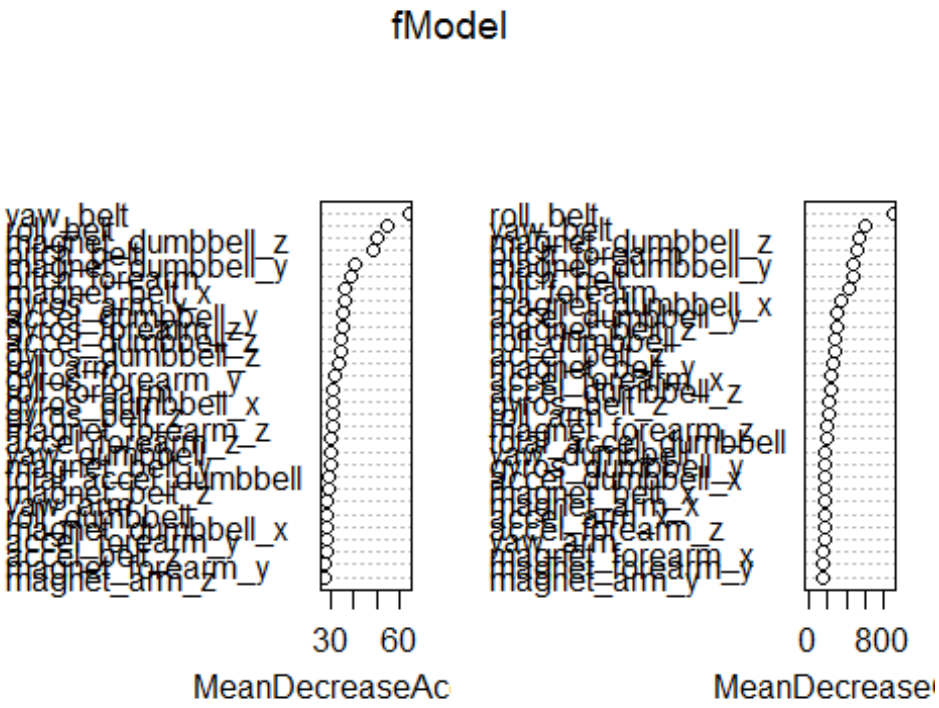
Now we have a dataset for modeling with 13737 observations and a validation data set with 5885. However, both of these data sets still have 53 columns. That's still a lot of

covariates. We'll need to run some additional analysis to determine their relative importance and thus their influence on the prediction algorithm.

Correlation Analysis

After playing with several models, we decided that a reasonable way to determine the relative importance of the numerous covariables, is to use the Random Forest algorithm, with the "importance" argument set to "TRUE" to assess the importance of the variables as predictors. After running the Random Forrest model numerous times with various limits on the number of trees built, we elected to run it without a row limit for this display.

```
set.seed(randomSeed)
fModel <- randomForest(classe~., data=modData, importance=TRUE)
varImpPlot(fModel)
```



We initially selected 10 variables as most significant; however, we observed an observable decrease in accuracy after the top 4, and another after the top 6. We therefore elected to work with the top six covariable going forward. In order to make the top 6, a variable had to place well in the top 10 in several different models (not shown). The variable selected place in the top 6 in multiple models. The top two variables tended to alternate between first and second place in the various models. A single variable appeared in third place in most models. This leads us to believe that we need to check first and second to see if they are highly correlated with each other.

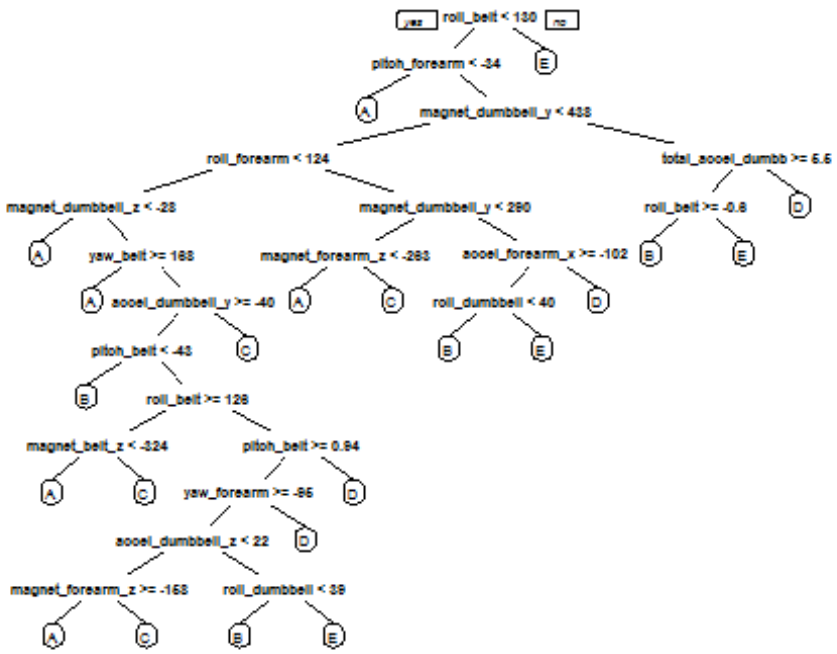
The top six selected variables are shown below, where we'll check them for correlation.

```
top6 <- c("yaw_belt", "roll_belt", "magnet_dumbbell_z", "pitch_forearm",
"pitch_belt", "magnet_dumbbell_y")
top6Corr <- cor(modData[, top6])
top6Corr
```

```
##                      yaw_belt  roll_belt magnet_dumbbell_z pitch_forearm
## yaw_belt           1.00000000  0.8156966        -0.2155322   -0.03074632
## roll_belt          0.81569659  1.0000000        -0.5002850    0.17231561
## magnet_dumbbell_z -0.21553225 -0.5002850         1.0000000   -0.17287670
## pitch_forearm     -0.03074632  0.1723156        -0.1728767    1.00000000
## pitch_belt        -0.69420905 -0.2088877        -0.2623780    0.25227283
## magnet_dumbbell_y -0.03616394 -0.2888057         0.2566349   -0.14670215
##                    pitch_belt magnet_dumbbell_y
## yaw_belt           -0.6942091       -0.03616394
## roll_belt          -0.2088877       -0.28880574
## magnet_dumbbell_z  -0.2623780        0.25663489
## pitch_forearm       0.2522728       -0.14670215
## pitch_belt          1.0000000       -0.36253795
## magnet_dumbbell_y  -0.3625379        1.00000000
```

From the above, we see that our top two (i.e. "roll_belt" and "yaw_belt") show about an 82% correlation with each other; our suspicions are confirmed; we'd be hard pressed to call them "independent covariates."

Since they seem to be so closely correlated, we should eliminate one of them from the analysis. The problem is choosing which one is the more influential covariate. One way to discover this is to draw a "Classification Tree" using the modeling data.



Note that the Classification Tree above selected "roll_belt" as the first discriminant among the top 6, indicating that it is more important than other covariants. We previously ran a Classification tree using 53 covariants with similar results. This finding indicates that, of the top two covarients, "roll_belt" is more important, and should be kept, while "yaw_belt" is dropped from the list. Thus we have our "top5."

```
top5 <- top6[2:6]
top5Corr <- cor(modData[, top5])
top5Corr
```

```
##                    roll_belt magnet_dumbbell_z pitch_forearm pitch_belt
## roll_belt           1.0000000        -0.5002850     0.1723156 -0.2088877
## magnet_dumbbell_z  -0.5002850         1.0000000    -0.1728767 -0.2623780
## pitch_forearm       0.1723156        -0.1728767     1.0000000  0.2522728
## pitch_belt         -0.2088877        -0.2623780     0.2522728  1.0000000
## magnet_dumbbell_y  -0.2888057         0.2566349    -0.1467022 -0.3625379
##                   magnet_dumbbell_y
## roll_belt                -0.2888057
## magnet_dumbbell_z         0.2566349
## pitch_forearm            -0.1467022
## pitch_belt               -0.3625379
## magnet_dumbbell_y         1.0000000
```

From the above, we see that none of the remaining "top 5" covarients exibits more than 50% correlaton with any of the others. Thus, we can reasonably consider our top 5 to be a set of relatively independent variables.

Our top 5 independent variables: roll_belt, magnet_dumbbell_z, pitch_forearm, pitch_belt, magnet_dumbbell_y.

So let's make a new training set for modeling that contains only our selected "top-5."

```
modFrame <-
data.frame(modData$classe,modData$roll_belt,modData$magnet_dumbbell_z,modData
$pitch_forearm,modData$pitch_belt,modData$magnet_dumbbell_y)
names(modFrame)<-c("classe",top5)
dim(modFrame)
```

```
## [1] 13737     6
```

That's somewhat better; now we have a data frame that contains 6 columns, and only 13737 rows / observations. Let's use that for training.

Prediction Modeling

During our analysis, we explored using Decision Trees, Generalized Boosted Models, and Random Forest models. We found some unacceptable error rates among some of them (not shown).

Of these methods, we found the Random Forest algorithm to yield the more satisfactory results.

After evaluating a trade-off between performance and error rate, we elected to use the Random Forest algorithm for training, hoping that we have not mistaken the “best possible solution” for the “most likely solution.”

We are using the Random Forest algorithm, with our top 5 variables (listed above) for training. Our top 5 variables are relatively independent, with a maximum correlation of less than 50% between any of them.

Training

Since the training data are so large, we will attempt to keep k-fold cross-validation as simple as practicable.

```
set.seed(randomSeed)
nFolds <- 4
modFit <- train(classe ~ ., method="rf", data=modFrame,
                trControl=trainControl(method="cv", number=nFolds))
modFit

## Random Forest
##
## 13737 samples
##     5 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 10302, 10303, 10302, 10304
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9540658  0.9418973
##   3     0.9533378  0.9409773
##   5     0.9468587  0.9327796
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Accuracy

We'll use the "confusionMatrix" function from "caret" to get a reasonable idea of the accuracy of our model.

We'll apply it against that "Validation" set that we've been keeping.

```
vPredictions <- predict(modFit, newdata=valData)
confMat <- confusionMatrix(vPredictions, valData$classe)
confMat

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1623   35    9    6    4
##          B   14 1041   17    8   19
##          C   26   31  986   23    6
##          D    9   15   14  923    2
##          E    2   17    0    4 1051
##
## Overall Statistics
##
##                Accuracy : 0.9556
##                  95% CI : (0.9501, 0.9608)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9439
##  Mcnemar's Test P-Value : 0.0002072
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9695   0.9140   0.9610   0.9575   0.9713
## Specificity            0.9872   0.9878   0.9823   0.9919   0.9952
## Pos Pred Value         0.9678   0.9472   0.9198   0.9585   0.9786
## Neg Pred Value         0.9879   0.9795   0.9917   0.9917   0.9936
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2758   0.1769   0.1675   0.1568   0.1786
## Detection Prevalence   0.2850   0.1867   0.1822   0.1636   0.1825
## Balanced Accuracy      0.9784   0.9509   0.9717   0.9747   0.9833
```

This indicates that our accuracy is almost 96%, using only our top 5 variables as predictors.

Error Rate

The error rate is 1 - accuracy; which would imply that the error rate using only our top 5 variables is about 4%. We could probably have done better by including additional "high value" variable from our list in the final computations.

Coursera Challenge

Now we'll attempt to answer the 20 observations challenge represented by the "Testing" data set.

We've been holding the test data in "tstFrame."

We'll compute our "predictions" for "classe" and insert them in a new column in the "tstFrame."

```
cPredictions <- predict(modFit, newdata=tstFrame)
tstFrame$classe <- cPredictions
```

```
# Create a data frame with just the problem-ID's and predictions
cFrame <- data.frame(problem_id=tstFrame$problem_id, classe=tstFrame$classe)
cFrame
```

```
##    problem_id classe
## 1           1      B
## 2           2      A
## 3           3      B
## 4           4      A
## 5           5      A
## 6           6      E
## 7           7      D
## 8           8      B
## 9           9      A
## 10         10      A
## 11         11      B
## 12         12      C
## 13         13      E
## 14         14      A
## 15         15      E
## 16         16      E
## 17         17      A
## 18         18      B
## 19         19      B
## 20         20      B
```

Finally, we need to create twenty ".txt" files that will be uploaded to Coursera.

The twenty files to be named "problem_1.txt" ... "problem_20.txt," and are to contain only the predictions.

```
predictions <- cFrame$classe
createTxt = function (x) {
  n = length(x)
  for (i in 1:n) {
    fName = paste0("problem_",i,".txt")
    write.table(x[i], file=fName, row.names=FALSE, col.names=FALSE,
quote=FALSE)
  }
}
createTxt(predictions)
```

Conclusion

The accuracy of the model, as tested against the validation data set was within acceptable limits, indicating a prediction accuracy of 96% and error of 4%.

We expect that inclusion of the four next most significant covariables will result in increasing accuracy to approximately 99% or better.