Practical Machine Learning - Course Project - Writeup

Stuart Ward

August 20, 2015

Report Sections:

1. Background
2. Source Data
3. Project Goal
4. High-level strategy
5. Initial data analysis and prep
6. Model building and accuracy/error metrics
   A. Quadratic Discriminant Analysis (QDA)
   B. Random Forest
7. Summary and submission score

1. 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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

2. Data The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har
Citation for this dataset and literature review:
Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H.
Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements.
Proceedings of 21st Brazilian Symposium on Artificial Intelligence.
Advances in Artificial Intelligence - SBIA 2012.
In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

3. Project Goal Predict (in the testing dataset) the manner in which they did the exercise. This is the "classe" variable in the training set.

4. High-level strategy For this project, I utilized a method recommended by a Kaggle competition winner. The essence of the strategy is to get an initial model as quick as possible and utilize that as a benchmark and a data point for further improvements.

Two techniques to achieve this strategy are to (1) find quick ways to simplify the data to only ‘clean’ rows/columns of data, and (2) utilize a model that trains very quickly.

I also keep in mind the words of wisdom from **Nate Silver**, when asked about his tools/process: “I use Stata for anything hardcore and Excel for the rest.”

5. Initial data analysis and prep

Loading in the data

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

In order to see how to best simplify the data for the initial model, we are fortunate that we have the test data available to review. Since this is the data we are going to make predictions on, we can easily see that there are many columns that filled either completely with NA or completely with blanks.

These columns can be removed from both test and training sets since they will have no predictive power on the test set.

```
training <- trainingFromFile[,-c(12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35
testing <- testingFromFile[,-c(12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
```

In addition, we can remove the initial 7 ‘bookkeeping’ columns, (as well as the problem_id column from the testing data), since these would not be representative of a typical data set.

```
training <- training[,-c(1,2,3,4,5,6,7)]
testing <- testing[,-c(1,2,3,4,5,6,7,60)]
```

6. Model building and accuracy/error metrics

A. This is a large, dense, robust data set; which is a sign that we **may not need to incur the performance and time costs that come with K-fold cross validation**. The first model will utilize the cross validation method of a simple hold-out validation set, consisting of a 70/30 random stratified split of the training data.

```
library(caret)
```

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

```
InTrain <- createDataPartition(y = training$classe, p=0.7, list=FALSE)
trainingForModel <- training[InTrain,]
trainingForValidation <- training[-InTrain,]
```

QDA is known to be a fast training (and surprisingly accurate) algorithm; this is the first model I’ll train

```
library(MASS)
```

Set start time to determine how long it takes to train the model

```
startTime <- Sys.time()
```

Train the model on the 70% of the training data

```
qdaTrain <- qda(classe ~ ., data =  trainingForModel)
```

Stop timer and report on training time

```
runTime <- Sys.time() - startTime
runTime
```

```
## Time difference of 0.506732 secs
```

Predict using the model on the 30% of the training data

```
qdaPred <- predict(qdaTrain, newdata = trainingForValidation[,-53])
```

Display the results of the predictions

```
confusionMatrix(qdaPred$class, trainingForValidation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1571   76    0    6    0
##          B   60  933   69    4   25
##          C   15  115  942  128   56
##          D   19    3    9  813   21
##          E    9   12    6   13  980
##
## Overall Statistics
##
##                Accuracy : 0.8902
##                  95% CI : (0.882, 0.8981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8612
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9385   0.8191   0.9181   0.8434   0.9057
## Specificity            0.9805   0.9667   0.9354   0.9894   0.9917
## Pos Pred Value         0.9504   0.8552   0.7500   0.9399   0.9608
## Neg Pred Value         0.9757   0.9570   0.9819   0.9699   0.9790
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2669   0.1585   0.1601   0.1381   0.1665
## Detection Prevalence   0.2809   0.1854   0.2134   0.1470   0.1733
## Balanced Accuracy      0.9595   0.8929   0.9268   0.9164   0.9487
```

Summary of QDA model results
- The time to train the model is less than one second
- The model accuracy is greater than 89% (out of sample error < 11%)

Given the speed of training, the QDA model accuracy is quite high.

B. Next, I utilize the Random Forest algorithm to see if it improves the model accuracy, yet still trains in a reasonable amount of time.

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Set start time to determine how long it takes to train the model

```
startTime <- Sys.time()
```

Train the model on the 70% of the training data

```
rfTrain <- randomForest(classe ~ ., data = trainingForModel)
```

Stop timer and report on training time

```
runTime <- Sys.time() - startTime
runTime
```

```
## Time difference of 30.1185 secs
```

Predict using the model on the 30% of the training data

```
rfPred <- predict(rfTrain, newdata = trainingForValidation)
```

Display the results of the predictions

```
confusionMatrix(rfPred, trainingForValidation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    3    0    0    0
##          B    0 1135    6    0    0
##          C    0    1 1015   11    0
##          D    0    0    5  953    3
##          E    1    0    0    0 1079
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9893   0.9886   0.9972
## Specificity            0.9993   0.9987   0.9975   0.9984   0.9998
## Pos Pred Value         0.9982   0.9947   0.9883   0.9917   0.9991
## Neg Pred Value         0.9998   0.9992   0.9977   0.9978   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1725   0.1619   0.1833
## Detection Prevalence   0.2848   0.1939   0.1745   0.1633   0.1835
## Balanced Accuracy      0.9993   0.9976   0.9934   0.9935   0.9985
```

Summary of Random Forest model results
- The time to train the model is approximately 30 seconds
- The model accuracy is greater than 99% (out of sample error $<$ 1%)
- The results confirm that *K-fold cross validation is not necessary* to produce a highly accurate model. For this particular data set, utilizing the cross validation method of a simple random stratified split of the data is appropriate.

The Random Forest model significantly improves accuracy and maintains reasonable training times.

7. Summary and submission score Utilizing a random forest model trained on all the training data, I predicted the results of the test data, created the output files. and submitted them with the results of **scoring 20 out of 20 correct** (see code below).

```
rfTrain <- randomForest(classe ~ ., data = training)
answers <- predict(rfTrain, newdata = testing)

pml_write_files = function(x){
        n = length(x)
        for(i in 1:n){
                filename = paste0("problem_id_",i,".txt")
                write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
        }
}
pml_write_files(answers)
```

The training data was such that very little preprocessing was necessary; a cross validation method of a simple random stratified split of the training data was appropriate; and no model tuning was necessary to achieve a predictive accuracy $> 99\%$.