Practical Machine Learning Course Project

C. Kelly

November 15, 2015

## Synopsis

Machine learning deals with pattern recognition in data and the use of alogorithims to make predictions about future events. A well-known example of practical machine learning is NetFlix proposing movie selctions for a customer based on past movie selections. By monitoring viewer selections over time, algorithims were written that suggested future selections for the customer.

This project focuses on data about personal activity collected using devices such as Jawbone Up, Nike FuelBand, and Fitbit. Many individuals like to collect data about their daily movements in order to improve their health, to find patterns in their behavior, or because they like to see "their numbers". These devices, and the data collected, are great quantitatively - it's easy to see how much of an activity is done in a day. But in general these devices and the associated measurements say nothing about the qualitity of the activity or movement. An effort was undertaken (<http://groupware.les.inf.puc-rio.br/har>) to qualify the measurements, that is, to let the user know if the activity or movement was being done correctly. Data from this project was generously provided to Coursera for educational purposes in the Practical Machine Learning course.

The data consists of measurements from accelerometers placed on the belt, forearm, and arm of six participants, as well as the dumbbell, used in a lifting exercise. Each participant performed a dumbbell lift in the correct manner (as instructed by the trainer) and in five incorrect ways, incorporating common mistakes, such as placing the elbows to the front of the body or lifting the dumbbell only halfway. Each lift, with its corresponding measurements, was assigned to a class indicating the qualitity of the lift: Classe A for lifts done correctly and Classe B, C, D, or E if done incorrectly.

The goal of the Coursera assignment is to use the many accelerometer measurements and classifications captured durnig the lifts to develop a model that correctly predicts the class to which a dumbbel lift belongs in the testing set. Such information would allow a user to determine if the lift was done correctly or not.

## Overall Approach

Two data sets were provided by Coursera: a training data set (<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>) and a test data set (<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>). The training data was partioned into a training set (60%) and an out-of-sample test set (40%). Prediction models were developed and the one with the highest accuracy applied to the out-of-sample test to ensure that it works. The selected model was then used to predict the qualitity of 20 lifts in the test data set; the predicted class results (A - E) of the test data set were submitted to Coursera for evaluation/grading.

### The Training Data: Initial Analysis, Processing and Model Building

The training data set was loaded:

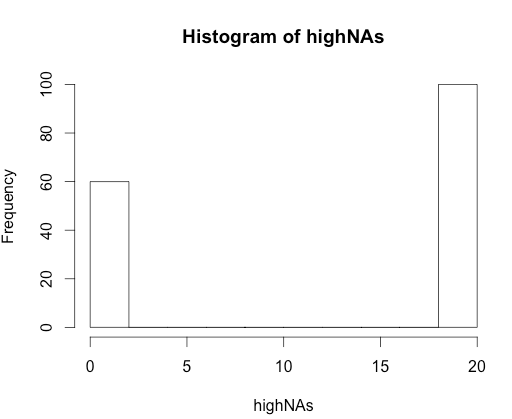
setwd("~/Documents/Coursera/1. DataScience/8. Practical Machine Learning/Course Project")

TrainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
  
if(!file.exists("pml-training.csv")) {  
 download.file(TrainURL, destfile = "pml-training.csv", method = 'curl')  
}  
TrainingData <- read.csv("pml-training.csv", na.strings = c("NA", ""))

**Initial Exploration**

An initial exploration of the training data was conducted:

dim(TrainingData) # Training data contains 160 variables and 19622 observations  
head(TrainingData) # Results hidden  
names(TrainingData) # Results hidden  
unique(TrainingData$classe) # Shows that all exercise classes (A - E) are represented   
highNAs <- colSums(is.na(TrainingData))

hist(highNAs) # There are columns with zero NAs and columns with ~19000 NAs and nothing in between  
  
**

table(highNAs)

Columns 1-7 contained non-accelerometer data such as user name, timestamp, etc. These columns were targeted for removal as they have no role in modeling the data. Additionally of the 160 variables (i.e., accelerometer measurements on the belt, forearm, arm and dumbbell capturing details of the exercise movement), 60 were complete (i.e., no NAs) while the remaining 100 variables hae nearly 20,000 missing values. These variables were also targeted for removal as they do not contribute to developing a prediction algorithim.

#### Data Pre-Processing: Tidying Up

First columns 1-7 were removed; columns with missing values were identified and removed. Finally the remaianing variables were checked for near zero variances; if variables exhibit essentially no variance then they would be removed as well.

tidyTrainingData <- TrainingData[, -(1:7)]   
  
NACols <- c()  
x <- length(colnames(tidyTrainingData))  
 for (i in 1:x) {  
 colSum <- colSums(is.na(tidyTrainingData[i]))  
   
 if (colSum > 1900) {  
 NACols <- c(NACols,i)  
 }   
 }  
tidyTrainingData <- tidyTrainingData[, -(NACols)]  
  
library(caret)

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

nsv <- nearZeroVar(tidyTrainingData, saveMetrics = TRUE)  
# The returned values of False indicate that all of the remaining variables should be considered in the model.

#### Data Partitioning

The tidy training data was partioned into a training set and an out-of-sample test set:

set.seed(334455)

TrainIndex <- createDataPartition(y=tidyTrainingData$classe, p = .60, list=FALSE)  
tidyTrainingSet <- tidyTrainingData[TrainIndex,]  
tidyOutSampleTestSet <- tidyTrainingData[-TrainIndex,]

#### Model Developement

Eight models were envisioned: Models 1 - 4 were based on the Classification and Regression Trees (CART, method = rpart) and Models 5 - 8 used the Random Forest method (method = rf). Within each set, the first model (Models 1 and 5) has no additional features (i.e., no pre-processing or cross-validation). The second model (Models 2 and 6) incorporated pre-processing (centering and scaling). The third model (Models 3 and 7) incorporated cross-valiation and the final model in each method set (Models 4 and 8) used both pre-processing and cross-validation.

The models based on the rpart method were investigated first followed by the Random Forest models.

##### Models using the CART (rpart) method:

library(rpart)  
# Model 1 (no additional features)  
set.seed(334455)  
Model1Fit <- train(classe ~ ., data = tidyTrainingSet, method = "rpart")  
print(Model1Fit, digits = 5)  
##CART

##11776 samples

##52 predictor

##5 classes: 'A', 'B', 'C', 'D', 'E'

##No pre-processing

##Resampling: Bootstrapped (25 reps)

##Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...

##Resampling results across tuning parameters:

## cp Accuracy Kappa Accuracy SD Kappa SD

## 0.03968913 0.4932138 0.34354110 0.06303997 0.10392753

## 0.04921096 0.4004392 0.18685580 0.06014452 0.10107793

## 0.11497390 0.3167178 0.05394947 0.04164765 0.06217517

##Accuracy was used to select the optimal model using the largest value.

##The final value used for the model was cp = 0.03968913.

# Model 2 (pre-processing only)  
set.seed(334455)  
Model2Fit <- train(classe ~ ., preProcess=c('center', 'scale'), data = tidyTrainingSet, method = "rpart")  
print(Model2Fit, digits = 5)  
##CART

##11776 samples

##52 predictor

##5 classes: 'A', 'B', 'C', 'D', 'E'

##Pre-processing: centered (52), scaled (52)

##Resampling: Bootstrapped (25 reps)

##Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...

##Resampling results across tuning parameters:

##cp Accuracy Kappa Accuracy SD Kappa SD

##0.03251068 0.5108935 0.36163872 0.03634654 0.05628157

##0.05987977 0.3788920 0.15038278 0.04327420 0.07037543

##0.11485524 0.3236184 0.06411121 0.04062571 0.06295213

##Accuracy was used to select the optimal model using the largest value.

##The final value used for the model was cp = 0.03251068.

# Model 3 (cross-validation only)  
set.seed(334455)  
Model3Fit <- train(classe ~ ., trControl = trainControl(method = 'cv', number = 4, allowParallel = TRUE), data = tidyTrainingSet, method = "rpart")  
print(Model3Fit, digits = 5)

##CART

##11776 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##No pre-processing

##Resampling: Cross-Validated (4 fold)

##Summary of sample sizes: 8833, 8833, 8831, 8831

##Resampling results across tuning parameters:

## cp Accuracy Kappa Accuracy SD Kappa SD

## 0.03251068 0.5014403 0.34821896 0.01141907 0.01490447

## 0.05987977 0.4284979 0.22999634 0.07163415 0.12067478

## 0.11485524 0.3446909 0.09214214 0.04033978 0.06146364

##Accuracy was used to select the optimal model using the largest value.

##The final value used for the model was cp = 0.03251068.

# Model 4 (both pre-processing and cross-validation)  
set.seed(334455)  
Model4Fit <- train(classe ~ ., preProcess=c('center', 'scale'), trControl = trainControl(method = 'cv', number = 4, allowParallel = TRUE), data = tidyTrainingSet, method = "rpart")  
print(Model4Fit, digits = 5)

##CART

##11776 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##Pre-processing: centered (52), scaled (52)

##Resampling: Cross-Validated (4 fold)

##Summary of sample sizes: 8833, 8833, 8831, 8831

##Resampling results across tuning parameters:

## cp Accuracy Kappa Accuracy SD Kappa SD

## 0.03251068 0.5014403 0.34821896 0.01141907 0.01490447

## 0.05987977 0.4284979 0.22999634 0.07163415 0.12067478

## 0.11485524 0.3446909 0.09214214 0.04033978 0.06146364

##Accuracy was used to select the optimal model using the largest value.

##The final value used for the model was cp = 0.03251068.

The accuracy of the 4 models based on the classification tree (rpart method) was very disappointing: accuracy values ranged from 0.484 to 0.494, which were essentially no better than a coin flip! Additionally, a plot of the classification tree for Model 1 (Figure 1) resulted in zero predictions for Classe D and it was known from the initial look at the data that all 5 classes of exercises (A - E) were in the training set. Thus, the rpart method was deemed to be totally inadequate as a prediction model for this data.

library(rattle)

## R session is headless; GTK+ not initialized.  
## Rattle: A free graphical interface for data mining with R.  
## Version 4.0.0 Copyright (c) 2006-2015 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

fancyRpartPlot(Model1Fit$finalModel)

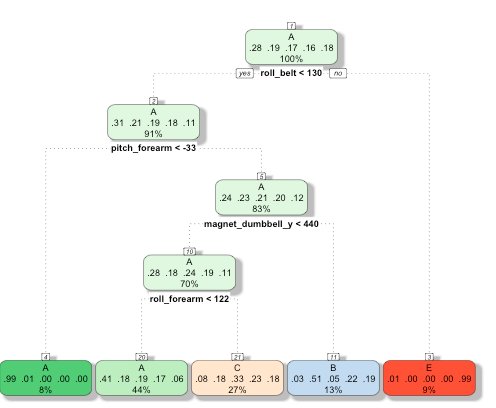


Figure 1: Classification Tree for Model 1

This plot was generated for Models 2, 3, and 4 as well; each showed similar outcomes. Only the tree from Model 1 is shown to do report length constraints.

With the inadequacy of rpart models established, an exploration of the models using the random forest method was thus warrented.

##### Models using the random forest (rf) method:

It was envisioned that four models using the random forest method would be built (Models 5 - 8) as described above. However based on entries in the Coursera discussion board it was apparent that the random forest method was quite time intensive. Indeed, Model 5 (random forest with no pre-processing or cross validation) ran for 4.5 hours on my machine (a MacBook Air, 1.3 GHz Intel i5 Processor) without ever completing. Model 6 (with pre-processing) encountered the same issue; Models 7 and 8 (cross-validation and cross-validation plus pre-processing) were run; each took roughly 30 minutes. The outout from Models 7 and 8 is shown in order to substantiate the final choice of model for this project.

library(randomForest)

## randomForest 4.6-12  
## Type rfNews() to see new features/changes/bug fixes.

# Model 5 (no additional features)  
#set.seed(334455)  
#Model5Fit <- train(classe ~ ., data = tidyTrainingSet, method = "rf", prox = TRUE)  
#print(Model5Fit, digits = 5)  
  
# Model 6 (pre-processing only)  
#set.seed(334455)  
#Model6Fit <- train(classe ~ ., data = tidyTrainingSet, preProcess=c('center', 'scale'), method = "rf", prox = TRUE, allowParellel = TRUE)  
#print(Model6Fit, digits = 5)  
  
# Model 7 (cross-validation only)  
set.seed(334455)  
Model7Fit <- train(classe ~ ., data = tidyTrainingSet, method = "rf", prox = TRUE, trControl = trainControl(method = 'cv', number = 3, allowParallel = TRUE))  
print(Model7Fit, digits = 5)

##Random Forest

##11776 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##No pre-processing

##Resampling: Cross-Validated (3 fold)

##Summary of sample sizes: 7850, 7851, 7851

##Resampling results across tuning parameters:

## mtry Accuracy Kappa Accuracy SD Kappa SD

## 2 0.98497 0.98098 0.0015509 0.0019626

## 27 0.98616 0.98249 0.0047140 0.0059633

## 52 0.97809 0.97228 0.0044111 0.0055776

##Accuracy was used to select the optimal model using the largest value.

##The final value used for the model was mtry = 27.

# Model 8 (both pre-processing and cross-validation)  
set.seed(334455)  
Model8Fit <- train(classe ~ ., data = tidyTrainingSet, method = "rf", prox = TRUE, preProcess=c('center', 'scale'), trControl = trainControl(method = 'cv', number = 3, allowParallel = TRUE))  
print(Model8Fit, digits = 5)

##Random Forest

##11776 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##Pre-processing: centered (52), scaled (52)

##Resampling: Cross-Validated (3 fold)

##Summary of sample sizes: 7850, 7851, 7851

##Resampling results across tuning parameters:

## mtry Accuracy Kappa Accuracy SD Kappa SD

## 2 0.98590 0.98216 0.0022993 0.0029097

## 27 0.98658 0.98303 0.0039663 0.0050170

## 52 0.97750 0.97153 0.0040452 0.0051148

##Accuracy was used to select the optimal model using the largest value.

##The final value used for the model was mtry = 27.

##### Model selection and out-of-sample testing:

The Model 7 (rf with 4-fold cross-validation) was chosen for moving forward in this project. Of the two random forest models run (Models 7 and 8), Model 7 had a slightly higher accuracy than Model 8 (0.98565 versus 0.98539) and took essentially the same amount of time to run.

Model 7 was used to test accuracy by running it on the out-of-sample test set:

predicted <- predict(Model7Fit, newdata = tidyOutSampleTestSet)

print(confusionMatrix(predicted, tidyOutSampleTestSet$classe), digits = 3)

##Confusion Matrix and Statistics

## Reference

##Prediction A B C D E

## A 2231 20 0 0 0

## B 1 1492 5 0 1

## C 0 6 1358 18 1

## D 0 0 5 1265 4

## E 0 0 0 3 1436

##Overall Statistics

## Accuracy : 0.992

## 95% CI : (0.99, 0.994)

## No Information Rate : 0.284

## P-Value [Acc > NIR] : <2e-16

## Kappa : 0.99

## Mcnemar's Test P-Value : NA

Statistics by Class:

## Class: A Class: B Class: C Class: D Class: E

##Sensitivity 1.000 0.983 0.993 0.984 0.996

##Specificity 0.996 0.999 0.996 0.999 1.000

##Pos Pred Value 0.991 0.995 0.982 0.993 0.998

##Neg Pred Value 1.000 0.996 0.998 0.997 0.999

##Prevalence 0.284 0.193 0.174 0.164 0.184

##Detection Rate 0.284 0.190 0.173 0.161 0.183

##Detection Prevalence 0.287 0.191 0.176 0.162 0.183

##Balanced Accuracy 0.998 0.991 0.994 0.991 0.998

As seen in the confusion Matrix, Model 7 was 99.2% accurate in predicting the class for each exercise in the out-of-sample data set. This corresponds to a 0.8% error rate (1 - accuracy) and provides confidence in using Model 7 for predicting the class outcomes in the test data set.

### Prediction Assignment

Model 7 was applied to the test set provided by Coursera for prediction purposes. The test data was loaded and tidied in the same manner as the training set; Model 7 was then used to predict the class outcome of the 20 exercises in the test data.

TestURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
  
if(!file.exists("pml-testing.csv")) {  
 download.file(TestURL, destfile = "pml-testing.csv", method = 'curl')  
}  
  
TestingData <- read.csv("pml-testing.csv", na.strings = c("NA", ""))  
  
#Check that the test file has the same parameters as the training data file:  
colnames\_TrainingData <- colnames(TrainingData)  
colnames\_TestingData <- colnames(TestingData)  
all.equal(colnames\_TrainingData[1:length(colnames\_TrainingData)-1], colnames\_TestingData[1:length(colnames\_TrainingData)-1])   
#Since TRUE is returned then all of the columns match up between the two data sets.  
  
#Explore and tidy the test set in the same manner as the training set:  
tidyTestingData <- TestingData[, -(1:7)]  
  
highNAs <- colSums(is.na(TestingData))  
table(highNAs)  
# Since 100 variables had only 20 missing values, these were not removed from the test data set.

Model 7 was run on the tidy test set with the following predicted outcomes:

predictions <- predict(Model7Fit, tidyTestingData)  
predictions <- as.character(predictions)  
predictions

##[1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"

As a last step, the outcomes were written to the files to be submitted for grading per the project instructions:

setwd("~/Documents/Coursera/1. DataScience/8. Practical Machine Learning/Course Project/Prediction Answer Files")  
  
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(predictions)

### Conclusion

Of the 8 models designed for this exercise, the 4 using a classification and regression tree approach (rpart) produced models that predicted the exercise class outcome no better than a coion flip. And none of those models predicted Classe D exercises even though that class existed in the training data set. Better models were produced using the random forest method: due to time constraints only two of the models were investigated: one with 4-fold cross validation and one with both pre-process and cross-validation. Both models had high accuracy (~99%) although Model 7 was slightly higher. Model 7 was used to predict the exercise class outcome of the 20 data points in the test set; with the error rate reported above of 0.8%, there should be only one miss at most in the predictions.