Machine Learning Lesson 8 Final Project

John Basbagill

April 9, 2017

Project Summary

This project predicts the manner in which subjects performed a certain exercise. Devices such as Fitbit and Jawbone Up collect a significant amount of data relating to physical activity. The data can be analyzed to find patterns in exercise behavior, and this is useful since it can highlight incorrect methods of using exercise equipment or performing exercises. The analysis can target exercises performed incorrectly, and users can then work on correcting these exercises.

The data for this project came from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. The participants performed barbell lifts correctly and incorrectly in 5 different ways. The 5 movements were: A: performing the movement correctly, B: throwing the elbows to the front (incorrect), C: lifting the dumbbell halfway (incorrect), D: lowering the dumbbell only halfway (incorrect), and E: throwing the hips to the front (incorrect). The dataset is here http://groupware.les.inf.puc-rio.br/har, the training data is here https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv, and the test data is here https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv.

Modeling Strategy and Results Summary

The modeling strategy is to create two models: a random forest model then a generalized boosted regression model (gbm). Model results will be compared with test results to see how well they predict the 5 lifts. The primary R modeling packages used include caret, randomForest, and gbm. The results showed that the Random forest model predicted with 99% accuracy, and the gbm model predicted with 96% accuracy. The models were compared against the 20 observations in the test data, and both models predicted the results with 100% accuracy.

```
library(readr)
library(caret)
```

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

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

```
library(rattle)
```

```
## R session is headless; GTK+ not initialized.
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

Reading the data

First we read in the data, removing columns with either empty characters or that contain NA or #DIV/0 entries.

```
train <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
test <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
dim(train)
```

```
## [1] 19622   160
```

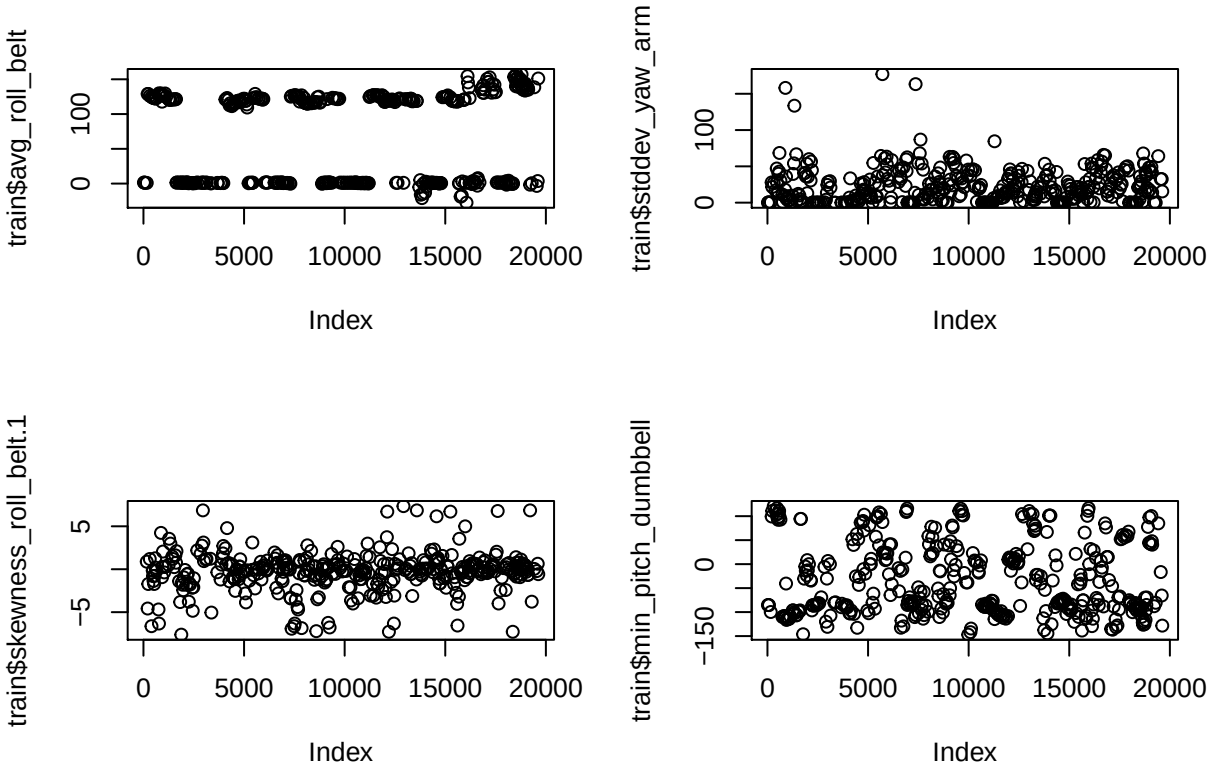
```
dim(test)
```

```
## [1]  20 160
```

Cleaning the data

Since the data have so many variables, let's first find ways to scope it down to a relevant number. By calling str(train) and then plotting a few columns with NAs in them, we see that many variables have a large number of NAs.

```
par(mfrow=c(2,2))
plot(train$avg_roll_belt)
plot(train$stddev_yaw_arm)
plot(train$skewness_roll_belt.1)
plot(train$min_pitch_dumbbell)
```



We will assume that relevant data is limited to those variables without a significant number of NA values and therefore limit the analysis to variables with a value for every observation. Additionally the first 7 columns include only general information about the participant names, times, etc., so these are not predictor variables and are removed.

```
train <- train[ ,-c(1:7)]
train <- train[, apply(train, 2, function(pred) !any(is.na(pred)))]
dim_train <- dim(train)
```

We have reduced the variables by over 50% from 160 to 53. Next we divide the training data into training (75%) and testing (25%) subsets.

Subset training and test datasets

```
in_train <- createDataPartition(train$classe, p = .75, list = FALSE)
train_subset <- train[in_train, ]
test_subset <- train[-in_train, ]
```

Random forest model

We create a random forest model using the trainControl() and train() functions, then look at the results by calling the final model as well as using the predict() function.

```
set.seed(5678)
fit <- trainControl(method = "cv", number = 5, classProbs = TRUE, verbose = TRUE)
fit_forest <- train(classe~., data = train_subset, method = "rf", trControl = fit, verbose =
        FALSE)
```

```
## + Fold1: mtry= 2
```

```
## - Fold1: mtry= 2
## + Fold1: mtry=27
## - Fold1: mtry=27
## + Fold1: mtry=52
## - Fold1: mtry=52
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=27
## - Fold2: mtry=27
## + Fold2: mtry=52
## - Fold2: mtry=52
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=27
## - Fold3: mtry=27
## + Fold3: mtry=52
## - Fold3: mtry=52
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=27
## - Fold4: mtry=27
## + Fold4: mtry=52
## - Fold4: mtry=52
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=27
## - Fold5: mtry=27
## + Fold5: mtry=52
## - Fold5: mtry=52
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 27 on full training set
```

```
print(fit_forest)
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11775, 11775, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9908957  0.9884828
##   27    0.9909637  0.9885690
##   52    0.9835576  0.9792008
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(fit_forest$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.66%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4178    5    1    0    1 0.001672640
## B   20 2823    4    1    0 0.008778090
## C    0    9 2546   12    0 0.008180756
## D    0    1   26 2382    3 0.012437811
## E    0    2    4    8 2692 0.005173688
```

```
pred_forest <- predict(fit_forest, newdata = test)
print(pred_forest)
```

```
##  [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
```

Generalized boosted regression model

We create a gbm using the train() function, then look at the results by calling the final model and confusion matrix as well as using the predict() function.

```
set.seed(5678)
fit_gbm <- train(classe~., data = train_subset, method = "gbm", trControl = fit, verbose = FALSE)
```

```
## Loading required package: plyr
```

```
## + Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
```

```
## - Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on full training s
```

```
print(fit_gbm)
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11775, 11775, 11773
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7556052  0.6901964
##   1                  100      0.8232084  0.7761730
##   1                  150      0.8533078  0.8143690
##   2                   50      0.8577931  0.8197657
##   2                  100      0.9094983  0.8854370
##   2                  150      0.9317840  0.9136727
##   3                   50      0.8985583  0.8715773
##   3                  100      0.9422475  0.9269220
##   3                  150      0.9614079  0.9511770
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
print(fit_gbm$finalModel)
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 45 had non-zero influence.
```

```
print(confusionMatrix(fit_gbm))
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 28.0  0.6  0.0  0.0  0.0
##          B  0.3 18.2  0.5  0.0  0.2
##          C  0.1  0.5 16.7  0.6  0.2
##          D  0.1  0.0  0.2 15.6  0.2
##          E  0.0  0.0  0.0  0.1 17.7
##
##  Accuracy (average) : 0.9614
```

```
pred_gbm <- predict(fit_gbm, newdata = test)
## errors_gbm <- confusionMatrix(pred_gbm, test$classe)
print(pred_gbm)
```

```
##  [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
```

Out of sample error

The out of sample error is equal to 1 minus the accuracy of the two models in the cross validation data. The accuracy values for the random forest and gbm moels were 99% and 96%, respectively, so the out of sample errors were 1% and 4%, respectively.

Summary

The out of bag error rate for the random forest model was 0.63% with a sample size average of about 11,774 out of 14,718 original samples and 52 predictors. 5-fold cross validation was used with the number of trees equal to 500, the number of variables tried at each split equal to 2, and the number of variables tried at each split equal to 26. The final model resulted in 99% accuracy and had a kappa avalue of 98%. The classification errors in the confusion matrix were very small, ranging from .0007 to .017. The model was 100% accurate when tested against the 20 observations in the testing model.

The gbm model held the number of trees equal to 150 and had an interaction depth equal to 3, shrinkage equal to 0.1, and n.minobsinnode equal to 10. The sample size was about 11,774 with 14,718 samples and 52 predictors. 5-fold cross validation was used with accuracy values increasing from 75% (interaction depth = 1 and number of trees = 50) to 96% (depth=1, trees=150) and kappa values increasing from 69% to 95%. 46 of the 52 predictors were found to have non-zero influence, and the average accuracy was 96%. Finally, the model was 100% accurate when compared against the 20 observations in the testing model.

Therefore the random forest model performed slightly better than the gbm in terms of overall accuracy, although both models were able to correctly predict all outcomes for the 20 observations in the test data.