Practical Machine Learning Project

Lisa Mudgett

December 21, 2017

Introduction

The goal of the project is to use fitness data to predict the manner in which someone did an exercise.

This analysis used data from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). The data come from devices such as Jawbone Up, Nike FuelBand, and Fitbit, and for this project, we'll be looking at data collected from accelerometer on the belt, forearm, arm, and dumbell of six participants.

The data were split into testing and training data for this assignment, and can be found here:

Training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Test: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Analysis

Let's load the data and packages we'll need for analysis.

```
setwd('C:/Users/r624461/Desktop/Data Science/Practical Machine Learning')
training <- read.csv('pml-training.csv')
testing <- read.csv('pml-testing.csv')

library(caret)
```

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

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

```
## Warning: Installed Rcpp (0.12.10) different from Rcpp used to build dplyr (0.12.1
1).
## Please reinstall dplyr to avoid random crashes or undefined behavior.
```

```
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(e1071)
```

Doing some exploration on the data shows there are columns with near zero variance, columns that are almost always NA, and some informational columns that we won't need for predicting.

```
training <- training[, colSums(is.na(training)) ==0]
testing <- testing[, colSums(is.na(testing)) ==0]

nzv <- nearZeroVar(training)
training <- training[, -nzv]
testing <- testing[, -nzv]

training <- training[, -(1:5)]
testing <- testing[, -(1:5)]
```

For the out of sample error required in the assignment, I want to break the training set into another training and testing set. Since this is a medium size data set, I'll use the 60/40 split.

```
inTrain <- createDataPartition(y=training$classe, p=.6, list=FALSE)
newtrain <- training[inTrain,]
newtest <- training[-inTrain,]
```

Now let's build some models!

```
set.seed(4309)
```

Random Forest

```
fit_rf <- randomForest(classe~., data=newtrain, prox=TRUE)
predict_rf <- predict(fit_rf, newtest)
confusionMatrix(newtest$class, predict_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    1    0    0    0
##          B    5 1510    3    0    0
##          C    0    7 1361    0    0
##          D    0    0   13 1273    0
##          E    0    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9962
##                  95% CI : (0.9945, 0.9974)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9952
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9947   0.9884   0.9992   1.0000
## Specificity            0.9998   0.9987   0.9989   0.9980   0.9998
## Pos Pred Value         0.9996   0.9947   0.9949   0.9899   0.9993
## Neg Pred Value         0.9991   0.9987   0.9975   0.9998   1.0000
## Prevalence             0.2850   0.1935   0.1755   0.1624   0.1837
## Detection Rate         0.2843   0.1925   0.1735   0.1622   0.1837
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9988   0.9967   0.9936   0.9986   0.9999
```

Hey, pretty good results! Our accuracy is 99.48%, so the out of sample error is about half a percent.

Boosting

```
set.seed(11712)
fit_gbm <- train(classe~., method="gbm", data=newtrain, verbose=FALSE)
predict_gbm <- predict(fit_gbm, newtest)
confusionMatrix(newtest$class, predict_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    6    0    0    0
##          B   22 1478   17    1    0
##          C    0   11 1352    5    0
##          D    2    6   28 1250    0
##          E    3    6    2   12 1419
##
## Overall Statistics
##
##                Accuracy : 0.9846
##                  95% CI : (0.9816, 0.9872)
##     No Information Rate : 0.2872
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9805
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9880   0.9808   0.9664   0.9858   1.0000
## Specificity            0.9989   0.9937   0.9975   0.9945   0.9964
## Pos Pred Value         0.9973   0.9736   0.9883   0.9720   0.9840
## Neg Pred Value         0.9952   0.9954   0.9927   0.9973   1.0000
## Prevalence             0.2872   0.1921   0.1783   0.1616   0.1809
## Detection Rate         0.2837   0.1884   0.1723   0.1593   0.1809
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9935   0.9872   0.9820   0.9902   0.9982
```

Boosting is pretty good too - 98.41% accuracy on the test set, so the out of sample error is about 1.5%. The Random Forest was a better predictor.

Linear Discriminant Analysis

```
set.seed(22115)
fit_lda <- train(classe~., method="lda", data=newtrain)
predict_lda <- predict(fit_lda, newtest)
confusionMatrix(newtest$class, predict_lda)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1844   60  138  174   16
##          B  202  972  206   71   67
##          C  134  118  914  152   50
##          D   64   48  168  964   42
##          E   56  209  114  132  931
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7169          
##                  95% CI : (0.7068, 0.7269)
##     No Information Rate : 0.2931          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6419          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8017   0.6908   0.5935   0.6457   0.8418
## Specificity            0.9300   0.9152   0.9280   0.9493   0.9242
## Pos Pred Value         0.8262   0.6403   0.6681   0.7496   0.6456
## Neg Pred Value         0.9188   0.9313   0.9034   0.9194   0.9727
## Prevalence             0.2931   0.1793   0.1963   0.1903   0.1410
## Detection Rate         0.2350   0.1239   0.1165   0.1229   0.1187
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8659   0.8030   0.7608   0.7975   0.8830
```

Yikes, we're moving in the wrong direction! The accuracy of this prediction method is only 71.41%, so nearly 30% out of sample error.

Model Selection

Based on the three prediction models I tested above, the Random Forest method was the best, with a 99.49% accuracy on my cross-validation test sample.