Practical Machine Learning - Final Project

Frank Ambrosio

May 7, 2017

Assignment

One thing that people regularly do is quantify how much of a particular activity they do, but they neglect to quantify how well they do it. In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. Our machine learning algorithm will be applied to the 20 test cases available in the test data.

Background

In this data set the variable representing the outcome, and the variable that we want to predict, is “classe”, a factor variable with 5 levels(A,B,C,D,E). To gather this data, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 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) -throwing the hips to the front (Class E) Accelerometers located at various points about the participant and their dumbell recorded data which we will use to predict which “classe” the data comes from.

Approach

Using Practical Machine Learning techniques we will use the accelerometer data provided at the following links to train models which will predict the manner in which certain exercises were performed. #####Data: training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) testing: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Loading Packages

To reproduce the results of this code certain packages must be installed and loaded.

```
library(caret)
```

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

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

```
library(rpart)
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(rattle)
```

```
## 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.
```

Loading the Data

The data from this project comes from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

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

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

Here we load the data sets into R, and tidy up the blank, NA and undefined values:

```
training <- read.csv(url('https://d396qusza40orc.cloudfront.net/predmachlearn/p
ml-training.csv'), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url('https://d396qusza40orc.cloudfront.net/predmachlearn/pm
l-testing.csv'), na.strings=c("NA","#DIV/0!",""))
```

Preprocessing the data

The training data set must be split into a training and (preliminary) testing set at a 60/40 ratio respectively (seed set for reproducibility).

```
set.seed(111)
```

Predictor variables with little or no values in their columns must be removed.

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

Here we partition the training data into training (mod_training) and validation (mod_testing) data sets.

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
mod_training <- training[inTrain, ]
mod_testing <- training[-inTrain, ]
finalmod_testset <- testing
```

The first seven columns of the data set are not accelerometer measurement data. These columns must be removed because only data measurements should be included in the machine learning algorithm. We perform this transformation on our final model test data set as well.

```
mod_training <- mod_training[,-c(1:7)]
mod_testing <- mod_testing[,-c(1:7)]
finalmod_testset <- finalmod_testset[,-c(1:7)]
```

Prediction Models

In these models we will be using the accelerometer data to predict the quality of the exercise which is represented in the variable “classe”.

Prediction Model 1

Decision Tree Model:

```
mod1 <- rpart(classe ~ ., data = mod_training, method = 'class')
```

Prediction:

```
pred1 <- predict(mod1, mod_testing, type = 'class')
```

Results:

```
confusionMatrix(pred1, mod_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2055  236   44   85   64
##          B   79  997  153  117  170
##          C   41  125 1085  192  149
##          D   26  120   86  812   88
##          E   31   40    0   80  971
##
## Overall Statistics
##
##                Accuracy : 0.7545
##                  95% CI : (0.7448, 0.764)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6882
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9207   0.6568   0.7931   0.6314   0.6734
## Specificity            0.9236   0.9180   0.9217   0.9512   0.9764
## Pos Pred Value         0.8273   0.6577   0.6815   0.7173   0.8654
## Neg Pred Value         0.9670   0.9177   0.9547   0.9294   0.9300
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2619   0.1271   0.1383   0.1035   0.1238
## Detection Prevalence   0.3166   0.1932   0.2029   0.1443   0.1430
## Balanced Accuracy      0.9221   0.7874   0.8574   0.7913   0.8249
```

The accuracy of this Decision Tree algorithm has been determined to be 0.7545 and the Estimated Out of Sample Error to be 0.2455.

Prediction Model 2

Random Forest Model:

```
mod2 <- randomForest(classe ~ ., data = mod_training, method = 'class')
```

Prediction:

```
pred2 <- predict(mod2, mod_testing, type = 'class')
```

Results:

```
confusionMatrix(pred2, mod_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230   20    0    0    0
##          B    2 1493    6    0    0
##          C    0    5 1362   18    2
##          D    0    0    0 1267    1
##          E    0    0    0    1 1439
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9909, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9911
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9835   0.9956   0.9852   0.9979
## Specificity            0.9964   0.9987   0.9961   0.9998   0.9998
## Pos Pred Value         0.9911   0.9947   0.9820   0.9992   0.9993
## Neg Pred Value         0.9996   0.9961   0.9991   0.9971   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1903   0.1736   0.1615   0.1834
## Detection Prevalence   0.2868   0.1913   0.1768   0.1616   0.1835
## Balanced Accuracy      0.9978   0.9911   0.9959   0.9925   0.9989
```

The accuracy of this Random Forest algorithm has been determined to be 0.9929 and the Estimated Out of Sample Error to be 0.0071.

Model Comparison

Because the accuracy of the Random Forest algorithm is greater than the accuracy of the Decision Tree algorithm we will proceed with the Random Forest Algorithm.

Validation

Here we apply our algorithm with cross validation. We use the mod_testing data set to cross validate our model. ####Prediction Model 3 Random Forest with Cross Validation Model:

```
mod3 <- train(classe ~ ., data = mod_training, method = "rf", trControl = train
Control(method = "cv", 5), ntree = 250)
```

Prediction:

```
pred3 <- predict(mod3, mod_testing)
```

Results:

```
confusionMatrix(mod_testing$classe, pred3)
```

```
## Confusion Matrix and Statistics
## 
##             Reference
## Prediction    A    B    C    D    E
##          A 2225    3    3    0    1
##          B   24 1486    8    0    0
##          C    0   10 1351    7    0
##          D    0    0   24 1260    2
##          E    0    0    2    1 1439
## 
## Overall Statistics
## 
##                Accuracy : 0.9892
##                  95% CI : (0.9866, 0.9913)
##     No Information Rate : 0.2866
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9863
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9893   0.9913   0.9733   0.9937   0.9979
## Specificity            0.9987   0.9950   0.9974   0.9960   0.9995
## Pos Pred Value         0.9969   0.9789   0.9876   0.9798   0.9979
## Neg Pred Value         0.9957   0.9979   0.9943   0.9988   0.9995
## Prevalence             0.2866   0.1911   0.1769   0.1616   0.1838
## Detection Rate         0.2836   0.1894   0.1722   0.1606   0.1834
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9940   0.9931   0.9854   0.9949   0.9987
```

The accuracy of this Random Forest algorithm has been determined to be 0.989 and the Estimated Out of Sample Error to be 0.011

Ultimately we must test our model on a set of data that has not been used at all throughout the building of this model. Fortunately we partitioned a set of testing data and saved it for our final model. Here we will attempt to use the model to make predictions with a test set it has never seen before.

Prediction:

```
pred4 <- predict(mod3, finalmod_testset)
pred4
```

```
##  [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
```