Practical Machine Learning Course Project

MadraghRua

November 6, 2017

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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Initiation and Reproducibility of Packages

To reproduce these analyses you will need to have the same sets of packaegs to do the analysis with. In addition you will need to use the same random seed value to generate the same kinds of results.

Finally set the seed to 56789 to standardize outcomes from the analysis

```
set.seed(56789)
```

Getting Data from the Web

Initalize the current working directory as follows:

```
setwd("./Project")
```

Now download the training and test data from the web. Put this into a data folder under the Project directory.

```
trainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingFile <- "./data/pml-training.csv"
testingFile <- "./data/pml-testing.csv"

if(!file.exists("./data")) {
  dir.create("./data")
}
if(!file.exists(trainingFile)) {
  download.file(trainingURL, destfile = trainingFile, method = "auto")
}
if(!file.exists(testingFile)) {
  download.file(testingURL, destfile = testingFile, method = "auto")
}
```

Reading Data

Now read the two data files into frames.

```
trainingFrame = read.csv(trainingFile)
testingFrame = read.csv(testingFile)
dim(trainingFrame)
```

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

```
dim(testingFrame)
```

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

The training data set contains 19622 observations and 160 variable. The testing data set has 20 sets of observations, each with 160 variables. Of the variabes, the classe variable is the one we need to predict.

Cleaning Data

First we clean up the Near Zero Variance valiables

```
NZV <- nearZeroVar(trainingFrame, saveMetrics = TRUE)
head(NZV, 20)
```

```
##                         freqRatio percentUnique zeroVar   nzv
## X                        1.000000  100.00000000   FALSE FALSE
## user_name                1.100679    0.03057792   FALSE FALSE
## raw_timestamp_part_1     1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2     1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp           1.000668    0.10192641   FALSE FALSE
## new_window              47.330049    0.01019264   FALSE  TRUE
## num_window               1.000000    4.37264295   FALSE FALSE
## roll_belt                1.101904    6.77810621   FALSE FALSE
## pitch_belt               1.036082    9.37722964   FALSE FALSE
## yaw_belt                 1.058480    9.97349913   FALSE FALSE
## total_accel_belt         1.063160    0.14779329   FALSE FALSE
## kurtosis_roll_belt    1921.600000    2.02323922   FALSE  TRUE
## kurtosis_picth_belt    600.500000    1.61553358   FALSE  TRUE
## kurtosis_yaw_belt       47.330049    0.01019264   FALSE  TRUE
## skewness_roll_belt   2135.111111    2.01304658   FALSE  TRUE
## skewness_roll_belt.1  600.500000    1.72255631   FALSE  TRUE
## skewness_yaw_belt      47.330049    0.01019264   FALSE  TRUE
## max_roll_belt           1.000000    0.99378249   FALSE FALSE
## max_picth_belt          1.538462    0.11211905   FALSE FALSE
## max_yaw_belt          640.533333    0.34654979   FALSE  TRUE
```

Now we update the two frames for the corrections

```
trainingCorrected1 <- trainingFrame[, !NZV$nzv]
testingCorrected1 <- testingFrame[, !NZV$nzv]
dim(trainingCorrected1)
```

```
## [1] 19622   100
```

```
dim(testingCorrected1)
```

```
## [1]  20 100
```

Although the training data remains the same the testing data diminishes from 160 variables to 100.

Now we remove columns that contain textual descriptions that don't have accelerometer measurements.

```
text <- grepl("^X|timestamp|user_name", names(trainingCorrected1))
trainingCorrected2 <- trainingCorrected1[, !text]
testingCorrected2 <- testingCorrected1[, !text]
dim(trainingCorrected2)
```

```
## [1] 19622    95
```

```
dim(testingCorrected2)
```

```
## [1] 20 95
```

The training data set now has 95 variables. The testing data is down to 95 variables.

Next we remove all the NAs in the data.

```
nas <- (colSums(is.na(trainingCorrected2)) == 0)
trainingCorrected3 <- trainingCorrected2[, nas]
testingCorrected3 <- testingCorrected2[, nas]
dim(trainingCorrected3)
```

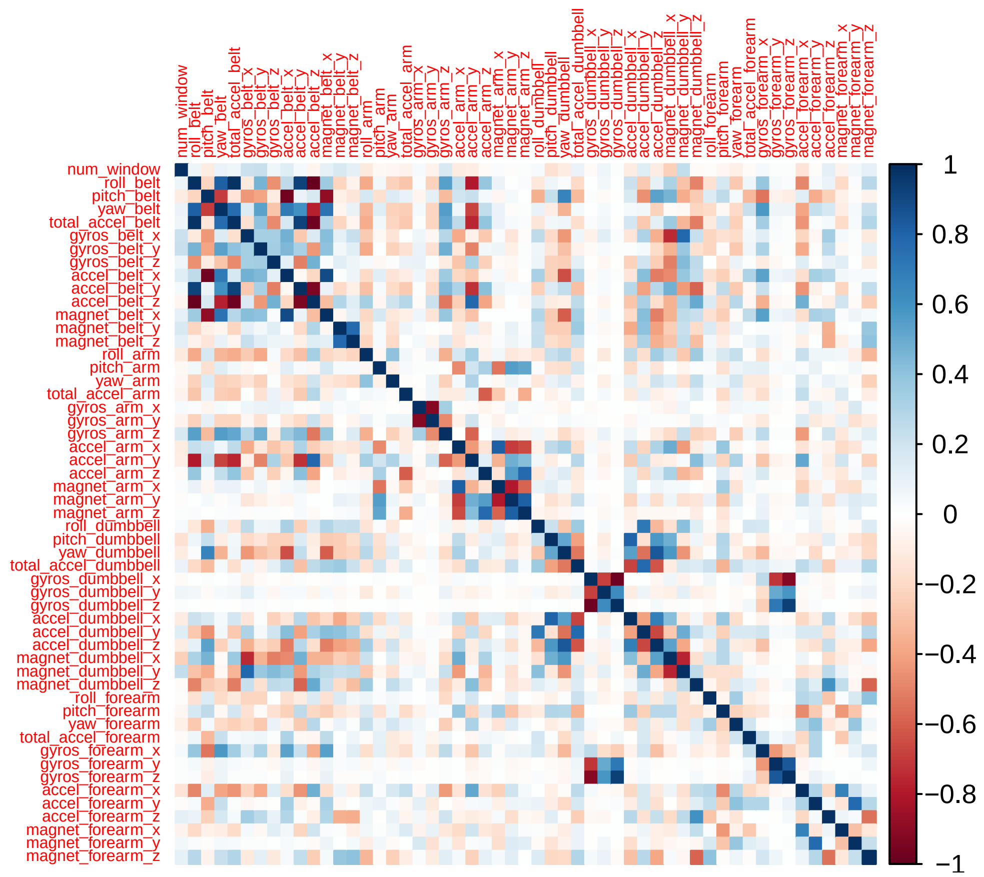
```
## [1] 19622    54
```

```
dim(testingCorrected3)
```

```
## [1] 20 54
```

At the end of all this cleaning and filtering the training data set has 19622 observations and 54 variables. The testing data set has 20 observations and 54 variables.

Generating a Correlation matrix of the Columns in the Training Data Set



##Partitioning the Training Set We make a 70%/30% split of the training set to train/validate.

```r
set.seed(56789)
trainPart <- createDataPartition(trainingCorrected3$classe, p = 0.70, list = FALSE)
validationSet <- trainingCorrected3[-trainPart, ]
trainingSet <- trainingCorrected3[trainPart, ]
dim(validationSet)
```

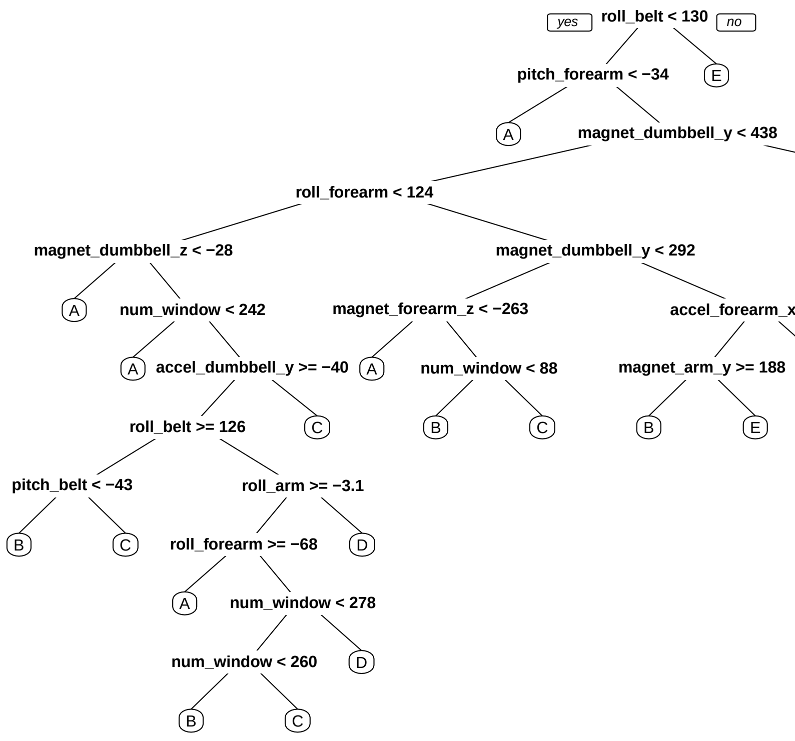
```
## [1] 5885   54
```

```r
dim(trainingSet)
```

```
## [1] 13737    54
```

Now the Training data has 13737 observations The Validation data has 5885 observations.

Data Modeling



A predictive model is created using a decision tree.

Now we look a how well the trained model fits the validation set data. A predictive model is created using a decision tree.

```
validationTree <- predict(tree, validationSet, type = "class")
confusionMatrix(validationSet$classe, validationTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1526   41   20   61   26
##          B  264  646   74  126   29
##          C   20   56  852   72   26
##          D   93   31  133  665   42
##          E   82   85   93  128  694
##
## Overall Statistics
##
##                Accuracy : 0.7448
##                  95% CI : (0.7334, 0.7559)
##     No Information Rate : 0.3373
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6754
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7688   0.7520   0.7270   0.6321   0.8494
## Specificity            0.9621   0.9019   0.9631   0.9381   0.9234
## Pos Pred Value         0.9116   0.5672   0.8304   0.6898   0.6414
## Neg Pred Value         0.8910   0.9551   0.9341   0.9214   0.9744
## Prevalence             0.3373   0.1460   0.1992   0.1788   0.1388
## Detection Rate         0.2593   0.1098   0.1448   0.1130   0.1179
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8654   0.8270   0.8450   0.7851   0.8864
```

```
accuracy <- postResample(validationTree, validationSet$classe)
outSampleError <- 1 - as.numeric(confusionMatrix(validationSet$classe, validationTree)$overall[1])
accuracy
```

```
##  Accuracy     Kappa
## 0.7447749 0.6754008
```

```
outSampleError
```

```
## [1] 0.2552251
```

So the estimated accuracy of the decision tree model is 74.5% and the Out of Sample is 25.5%

Random Forest

We now use Random Forest to fit a predictive model for activity.

```
modelRF <- train(classe ~ ., data = trainingSet, method = "rf", trControl = trainControl(method = "cv",
modelRF
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10990, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9946857  0.9932776
##   27    0.9978161  0.9972376
##   53    0.9957779  0.9946594
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Now we estimate random forest vs the validation set We now use Random Forest to fit a predictive model for activity.

```
validRF <- predict(modelRF, validationSet)
confusionMatrix(validationSet$classe, validRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    4 1135    0    0    0
##          C    0    1 1022    3    0
##          D    0    0    2  962    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9991   0.9980   0.9959   1.0000
## Specificity            1.0000   0.9992   0.9992   0.9996   0.9998
## Pos Pred Value         1.0000   0.9965   0.9961   0.9979   0.9991
## Neg Pred Value         0.9991   0.9998   0.9996   0.9992   1.0000
## Prevalence             0.2851   0.1930   0.1740   0.1641   0.1837
## Detection Rate         0.2845   0.1929   0.1737   0.1635   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9988   0.9991   0.9986   0.9977   0.9999
```

```
accuracy <- postResample(validRF, validationSet$classe)
outSampleError <- 1 - as.numeric(confusionMatrix(validationSet$classe, validRF)$overall[1])
```

The estimated Accuracy for Random Forest is 99.8%. The Out of Sample Error is 0.19%.

Random Forest vs the Original Test Set

Random Forest has performed best in these analyses. Noww we use the Random Forest to analyze the orignal test data set.

```
predict(modelRF, testingCorrected3[, -length(names(testingCorrected3))])
```

```
##  [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
```

So the test model results are: test 1 - B test 2 - A test 3 - B test 4 - A test 5 - A test 6 - E test 7 - D test 8 - B test 9 - A test 10 - A test 11 - B test 12 - C test 13 - B test 14 - A test 15 - E test 16 - E test 17 - A test 18 - B test 19 - B test 20 - B