Practical Machine Learning Prediction Assignment - An Examination of How Well Exercises Were Performed

Author - G. Reynolds

Synopsis:

Six young healthy participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five 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) and throwing the hips to the front (Class E). The Class designations A-E are in the classe column of the dataset (column 160). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. More information can be gound at http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). To determine how well we could model this dataset, Random Forest and GBM were used, with 5-fold cross validation. The Random Forest created better accuracy on the validation data and this is the model that was chosen for the prediction on the final test dataset.

1. Data Processing and Data Loading:

First, packages necessary to do the coming calculations were loaded into the R environment.

```
library(caret); library(rpart); library(rpart.plot); library(randomForest); library(corrplo
t)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.1.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

To start the analysis, a working directory was created and the data files were downloaded into this directory. As a best practice, the directory was created if it did not already exist.

```
setwd("~/Documents/Coursera work files/Machine Learning")
if(!file.exists("data")){dir.create("data")}
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destf
ile= "data/pml-training.csv", method = "curl") #Use the curl method https downloads under t
he Macintosh environment

if(!file.exists("data")){dir.create("data")}
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfi
le= "data/pml-testing.csv", method = "curl")
dateDownloaded <- date()
```

The data were downloaded on Sun Apr 3 08:52:13 2016. The pre-processing of the data was started by reading in the training file and looking at a summary.

```
trainRaw <- read.csv("data/pml-training.csv"); dim(trainRaw) #19622 rows with 160 columns o
f data
```

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

```
testRaw <- read.csv("data/pml-testing.csv"); dim(testRaw) #20 rows with 160 columns of data
```

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

```
#summary(trainRaw)
```

The variable/column names are not in a suitable format (from a data analysis perspective). First we will clean the testing and training datasets by removing the '_' from the column names.

```
names(trainRaw)<-gsub("_", "", names(trainRaw))
names(testRaw)<-gsub("_", "", names(testRaw))
```

Next we remove the columns that we do not believe will contribute to the way in which the exercises were performed. These include the 'X' column and the columns related to timestamps, participant names and windows.This reduces the data to 153 columns (predictor variables).

```
pmlTrain<-trainRaw[-c(1:7)] #Have 153 columns
pmlTest<-testRaw[-c(1:7)] #Have 153 columns
```

2. Clean the Datasets

There are quite a few columns with NAs in the dataset. We could use complete.cases to accept only those rows of data that are complete across the dataset (see: http://stackoverflow.com/questions/28932098/extracting-complete-paired-values-non-na-from-a-matrix-in-r?lq=1 (http://stackoverflow.com/questions/28932098/extracting-complete-paired-values-non-na-from-a-matrix-in-r?lq=1)). However, when we do this, there are only 406 complete rows - the data will be reduced too much. So we will consider removing non-numeric columns in the training data. We need to put aside the classe column before proceeding. Once the non-numeric values are removed we will append the classe column back to the dataset.

```
naremovedpmlTrain <- pmlTrain[complete.cases(pmlTrain), ]
classe <-pmlTrain$classe
pmlTrainCleaned<-pmlTrain[,sapply(pmlTrain, is.numeric)] #Have 119 columns
pmlTrainCleaned <-pmlTrainCleaned[,colSums(is.na(pmlTrainCleaned)) == 0] #52 columns
pmlTrainCleaned$classe <- classe #Put the classe column back into the training set
```

Repeat the datacleanup for the test dataset. The 'problemid' column in the test set is numeric, so it will not be removed when we perform the is.numeric function.

```
pmlTestCleaned<-pmlTest[,sapply(pmlTest, is.numeric)]
pmlTestCleaned <-pmlTestCleaned[,colSums(is.na(pmlTestCleaned)) == 0] #53 columns
```

We now have a training set with 19622, 53 observations/columns and a test set with 20, 53 observations/columns. In order to do cross validation, we will further split the training dataset into training and validation datasets, using a 70/30 ratio. The training set has 13737 observations, while the validation set has 5885 observations. We will put aside this new validation dataset and focus on the new training set.

```
set.seed(125)
inTrain <-createDataPartition(pmlTrainCleaned$classe, p=0.7, list= FALSE)
trainData <- pmlTrainCleaned[inTrain,]
validationData <-pmlTrainCleaned[-inTrain,]
dim(trainData) #13737 observations, 53 columns
```

```
## [1] 13737    53
```

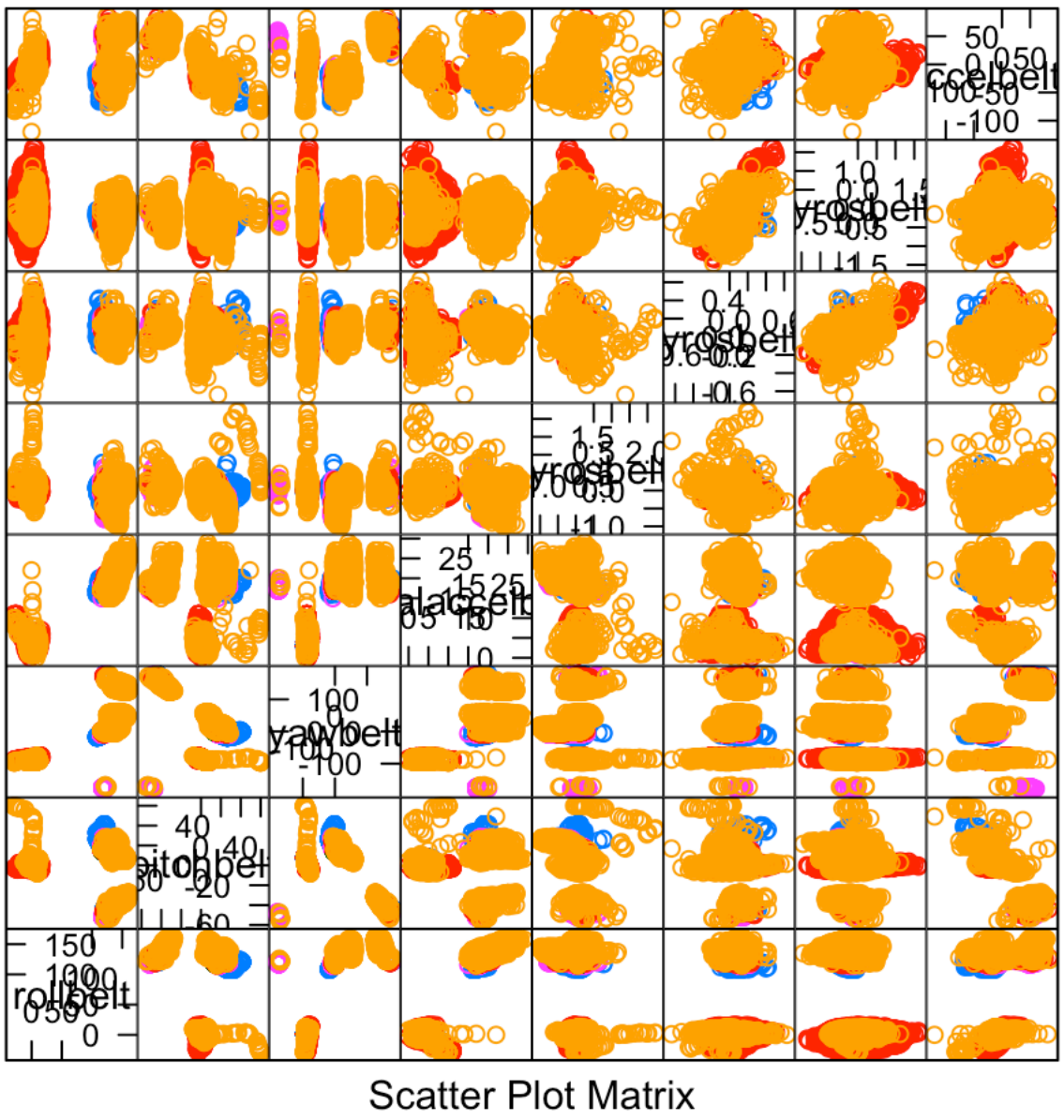
```
dim(validationData) #5885 observations, 53 columns
```

```
## [1] 5885   53
```

3. Exploratory Data Analysis

Create some feature plots of the variables versus classe. There are some correlations in these plots so we can look at how the variables might be correlated. At the 95% level, there are 12 variables that are highly correlated wtih each other.

```
featurePlot(x=trainData[,c(1:8)], y=trainData$classe, plot="pairs")
```



Scatter Plot Matrix

```
M <-abs(cor(trainData[,-53]))
diag(M) <-0
corrOutput<-which(M>0.95, arr.ind=T) #12 variables are highly correlated w/ each other
```

4. Data Modeling

Whichever models we choose to try, we want to perform cross validation to get a sense of the accuracy of the models. There are some videos (e.g. https://vimeo.com/75432414 (https://vimeo.com/75432414)) that show how to use for loops to set up n-fold datasets for cross validation. However, within R, one can use the 'trainControl' function to define the n-fold cross validation that can be used with models of choice (http://topepo.github.io/caret/training.html (http://topepo.github.io/caret/training.html)). For this project we will use 5-fold cross validation for all our modeling. Because there are quite a few variables that are correlated in the dataset, the first model to try will be Generalized Boosting Regression Modeling (GBM) with Principal Components Analysis. We first fit the model to the training set then predict on the validation dataset.

```
fitControl<-trainControl(method="cv", number=5, verbose=TRUE)
modelFitGBM <- train(classe~., method= "gbm", data=trainData, preProcess=c("pca"), trContro
l = fitControl, verbose=FALSE) #150 trees were used
```

```
## + 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 fu
ll training set
```

```
print(modelFitGBM)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction, scaled, centered
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10989, 10990, 10990, 10990, 10989
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD
##   1                   50      0.5552891  0.4287194  0.006917953
##   1                  100      0.6154179  0.5097327  0.008900414
##   1                  150      0.6463573  0.5501241  0.010609099
##   2                   50      0.6520347  0.5569839  0.006994384
##   2                  100      0.7227927  0.6483952  0.009145432
##   2                  150      0.7626848  0.6992174  0.006591472
##   3                   50      0.7139111  0.6370128  0.009228454
##   3                  100      0.7841599  0.7265082  0.009042706
##   3                  150      0.8169188  0.7682077  0.010682694
##   Kappa SD
##   0.008602615
##   0.011398352
##   0.013450263
##   0.009416390
##   0.011757119
##   0.008420718
##   0.011982422
##   0.011592348
##   0.013617102
##
## 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.
```

```
predictGBM <-predict(modelFitGBM, newdata=validationData)
confusionMatrix(predictGBM, validationData$classe) #Accuracy = 82.7%
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1508  107   50   35   16
##          B   34  849   88   20   71
##          C   50  114  836  109   67
##          D   77   32   34  779   32
##          E    5   37   18   21  896
##
## Overall Statistics
##
##                Accuracy : 0.8272
##                  95% CI : (0.8173, 0.8368)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7813
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9008   0.7454   0.8148   0.8081   0.8281
## Specificity            0.9506   0.9551   0.9300   0.9644   0.9831
## Pos Pred Value         0.8788   0.7994   0.7109   0.8166   0.9171
## Neg Pred Value         0.9602   0.9399   0.9597   0.9625   0.9621
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2562   0.1443   0.1421   0.1324   0.1523
## Detection Prevalence   0.2916   0.1805   0.1998   0.1621   0.1660
## Balanced Accuracy      0.9257   0.8503   0.8724   0.8863   0.9056
```

Second model to try: Random forest. We first fit the model to the training set then predict on the validation dataset.

```
trainControlRF <-trainControl(method="cv", number = 5)
modelFitRF<-train(classe ~., method="rf", data=trainData, trControl=trainControlRF, verbose
=FALSE)
print(modelFitRF) #52 predictors; (no preprocessing)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10990, 10989, 10989, 10989, 10991
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9904639  0.9879366  0.001529926  0.001935779
##   27    0.9914830  0.9892263  0.001078060  0.001363972
##   52    0.9879889  0.9848062  0.001817879  0.002300244
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
predictRF<-predict(modelFitRF, newdata=validationData)
confusionMatrix(predictRF, validationData$classe) #Accuracy 99.2%
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    7    0    0    0
##          B    0 1129    9    0    1
##          C    0    3 1009   10    5
##          D    0    0    8  953    1
##          E    0    0    0    1 1075
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9898, 0.9944)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9912   0.9834   0.9886   0.9935
## Specificity            0.9983   0.9979   0.9963   0.9982   0.9998
## Pos Pred Value         0.9958   0.9912   0.9825   0.9906   0.9991
## Neg Pred Value         1.0000   0.9979   0.9965   0.9978   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1918   0.1715   0.1619   0.1827
## Detection Prevalence   0.2856   0.1935   0.1745   0.1635   0.1828
## Balanced Accuracy      0.9992   0.9946   0.9899   0.9934   0.9967
```

```
modelFitRF$finalModel #out-of-sample error of 0.64%
```

```
##
## 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: 0.64%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    4    0    0    1 0.001280082
## B   18 2632    7    1    0 0.009781791
## C    0    9 2377   10    0 0.007929883
## D    0    1   25 2224    2 0.012433393
## E    0    0    3    7 2515 0.003960396
```

The random forest gives a higher degree of accuract (99.2%) versus the GBM model (82.7%). The random forest model also had an out of smaple error rate of 0.64%. We will use the random forest to predict on the 20 observations in the final test dataset.

5. Prediction Results

```
predict20<-predict(modelFitRF, newdata=pmlTest)
predict20 #Results: B A B A A E D B A A B C B A E E A B B B
```

```
##  [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
```

We therefore expect results of ‘B A B A A E D B A A B C B A E E A B B B’ for the 20 tests.