Practical Machine Learning Project

Roberto Ortiz

Sunday, April 24, 2016

Overview

This document is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning, as part of the Specialization in Data Science. It was built up in RStudio, using its knitr functions, meant to be published in html format. This analysis meant to be the basis for the course quiz and a prediction assignment writeup. The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

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, your 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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX

Data Loading

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 http://groupware.les.inf.puc-rio.br/har. Full source:

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.

My special thanks to the above mentioned authors for being so generous in allowing their data to be used for this kind of assignment.

A short description of the datasets content from the authors' website:

"Six young health 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).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg)."

```
library("caret")
```

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

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

```
setwd("C:/Users/ROBERTO/Desktop/VARIOS/CURSOS LIBRES/COURSERA/Data Science/Course 08 - Practical Machine
rm(list = ls())
if (!file.exists("pml-training.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "pml-t
}
if (!file.exists("pml-testing.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "pml-te
}
submit <- read.csv("pml-testing.csv", sep = ",", na.strings = c("", "NA"))
data <- read.csv("pml-training.csv", sep = ",", na.strings = c("", "NA"))
dim(submit)
```

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

```
dim(data)
```

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

Cleanup the data

Here, I remove columns full of NAs and remove features that are not in the submit set. The features containing NAs are the variance, mean and stddev within each window for each feature. Since the submit dataset has no time-dependence, these values are useless and can be disregarded. I also remove the first 7 features since they are related to the time-series or are not numeric.

```
# Remove columns full of NAs.
features <- names(submit[,colSums(is.na(submit)) == 0])[8:59]
# Only use features used in submit cases.
data <- data[,c(features,"classe")]
submit <- submit[,c(features,"problem_id")]
dim(submit)
```

```
## [1] 20 53
```

```
dim(data)
```

```
## [1] 19622    53
```

Bootstrap

Next, I withhold 25% of the dataset for testing after the final model is constructed.

```
set.seed(916)
inTrain = createDataPartition(data$classe, p = 0.75, list = F)
training = data[inTrain,]
testing = data[-inTrain,]
dim(training)
```

```
## [1] 14718    53
```

```
dim(testing)
```

```
## [1] 4904   53
```

Feature Selection

Some features may be highly correlated. The PCA method mixes the final features into components that are difficult to interpret; instead, I drop features with high correlation (>90%).

```
outcome = which(names(training) == "classe")
highCorrCols = findCorrelation(abs(cor(training[,-outcome])),0.90)
highCorrFeatures = names(training)[highCorrCols]
training = training[,-highCorrCols]
outcome = which(names(training) == "classe")
```

The features with high correlation are accel_belt_z, roll_belt, accel_belt_y, accel_belt_x, gyros_arm_y, gyros_forearm_z, and gyros_dumbbell_x.

Feature Importance

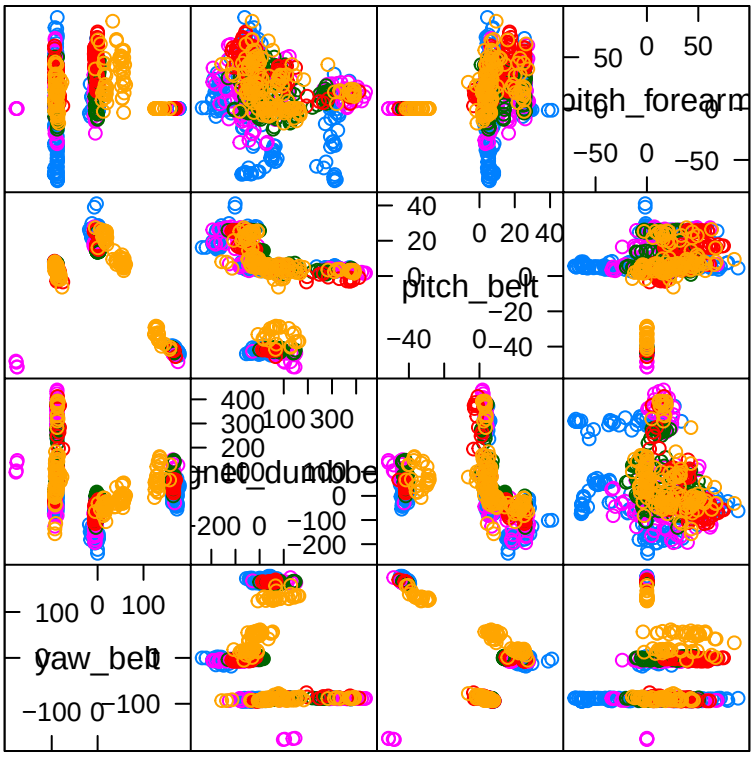
The random forest method reduces overfitting and is good for nonlinear features. First, to see if the data is nonlinear, I use the random forest to discover the most important features. The feature plot for the 4 most important features is shown.

```
library("randomForest")
```

```
## randomForest 4.6-10
```

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

```
fsRF = randomForest(training[,-outcome], training[,outcome], importance = T)
rfImp = data.frame(fsRF$importance)
impFeatures = order(-rfImp$MeanDecreaseGini)
inImp = createDataPartition(data$classe, p = 0.05, list = F)
featurePlot(training[inImp,impFeatures[1:4]],training$classe[inImp], plot = "pairs")
```



Scatter Plot Matrix

The most important features are pitch_belt, yaw_belt, total_accel_belt and gyros_belt_x.

Training

Train using the random forest and k-nearest neighbors for comparison.

```
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 3.2.5
```

```
ctrlKNN = trainControl(method = "adaptive_cv")
modelKNN = train(classe ~ ., training, method = "knn", trControl = ctrlKNN)
ctrlRF = trainControl(method = "oob")
modelRF = train(classe ~ ., training, method = "rf", ntree = 200, trControl = ctrlRF)
resultsKNN = data.frame(modelKNN$results)
resultsRF = data.frame(modelRF$results)
```

Testing Out-of-sample error

The random forest will give a larger accuracy compared to k-nearest neighbors. Here, I give the confusion matrix between the KNN and RF models to see how much they agree on the test set, then I compare each model using the test set outcomes.

```
fitKNN = predict(modelKNN, testing)
fitRF = predict(modelRF, testing)
confusionMatrix(fitKNN, fitRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1358   41   14    7   13
##          B    6  834   32    4   35
##          C   12   25  778   56   24
##          D   17   25   20  724   35
##          E    7   27    8   14  788
##
## Overall Statistics
##
##                Accuracy : 0.9139
##                  95% CI : (0.9057, 0.9217)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8911
##  Mcnemar's Test P-Value : 1.111e-13
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9700   0.8761   0.9131   0.8994   0.8804
## Specificity            0.9786   0.9805   0.9711   0.9763   0.9860
## Pos Pred Value         0.9477   0.9155   0.8693   0.8819   0.9336
## Neg Pred Value         0.9879   0.9704   0.9815   0.9802   0.9736
## Prevalence             0.2855   0.1941   0.1737   0.1642   0.1825
## Detection Rate         0.2769   0.1701   0.1586   0.1476   0.1607
## Detection Prevalence   0.2922   0.1858   0.1825   0.1674   0.1721
## Balanced Accuracy      0.9743   0.9283   0.9421   0.9379   0.9332
```

```
confusionMatrix(fitKNN, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1355   44   13    8   13
##          B    4  835   33    4   35
##          C   12   22  778   59   24
##          D   17   25   19  725   35
##          E    7   23   12    8  794
##
## Overall Statistics
##
##                Accuracy : 0.915
##                  95% CI : (0.9068, 0.9226)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8924
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9713   0.8799   0.9099   0.9017   0.8812
## Specificity            0.9778   0.9808   0.9711   0.9766   0.9875
## Pos Pred Value         0.9456   0.9166   0.8693   0.8831   0.9408
## Neg Pred Value         0.9885   0.9715   0.9808   0.9807   0.9736
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2763   0.1703   0.1586   0.1478   0.1619
## Detection Prevalence   0.2922   0.1858   0.1825   0.1674   0.1721
## Balanced Accuracy      0.9745   0.9303   0.9405   0.9392   0.9344
```

```
confusionMatrix(fitRF, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  944    8    0    0
##          C    0    0  844    8    0
##          D    0    0    3  796    6
##          E    0    0    0    0  895
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9913, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9871   0.9900   0.9933
## Specificity            0.9986   0.9980   0.9980   0.9978   1.0000
## Pos Pred Value         0.9964   0.9916   0.9906   0.9888   1.0000
## Neg Pred Value         1.0000   0.9987   0.9973   0.9980   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1721   0.1623   0.1825
## Detection Prevalence   0.2855   0.1941   0.1737   0.1642   0.1825
## Balanced Accuracy      0.9993   0.9964   0.9926   0.9939   0.9967
```

Applying the Selected Model to the Test Data

The accuracy of the 2 regression modeling methods above are: Random Forest = 0.993 and KNN =0.915

In that case, the Random Forest model will be applied to predict the 20 quiz results (testing dataset) as shown below.

```
predictTEST <- predict(modelRF, newdata=submit)
predictTEST
```

```
##  [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
```