

Prediction Assignment Writeup

Benjamin Mathew

5 December 2016

Overview

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 dumbbell 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).

Data Loading and Exploratory Analysis

Dataset Overview

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)."

Environment Preparation

We first upload the R libraries that are necessary for the complete analysis.

```
rm(list=ls())                # free up memory for the download of the data sets
library(knitr)
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
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.
```

```
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(corrplot)
set.seed(12345)
```

Data Loading and Cleaning

The next step is loading the dataset from the URL provided above. The training dataset is then partitioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing dataset is not changed and will only be used for the quiz results generation.

```
# set the URL for the download
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

# create a partition with the training dataset
```

```
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet <- training[-inTrain, ]
dim(TrainSet)
```

```
## [1] 13737 160
```

```
dim(TestSet)
```

```
## [1] 5885 160
```

Both created datasets have 160 variables. Those variables have plenty of NA, that can be removed with the cleaning procedures below. The Near Zero variance (NZV) variables are also removed and the ID variables as well.

```
# remove variables with Nearly Zero Variance
```

```
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet <- TestSet[, -NZV]
dim(TrainSet)
```

```
## [1] 13737 106
```

```
dim(TestSet)
```

```
## [1] 5885 106
```

```
# remove variables that are mostly NA
```

```
AllNA <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet <- TestSet[, AllNA==FALSE]
dim(TrainSet)
```

```
## [1] 13737 59
```

```
dim(TestSet)
```

```
## [1] 5885 59
```

```
# remove identification only variables (columns 1 to 5)
```

```
TrainSet <- TrainSet[, -(1:5)]
TestSet <- TestSet[, -(1:5)]
dim(TrainSet)
```

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

```
dim(TestSet)
```

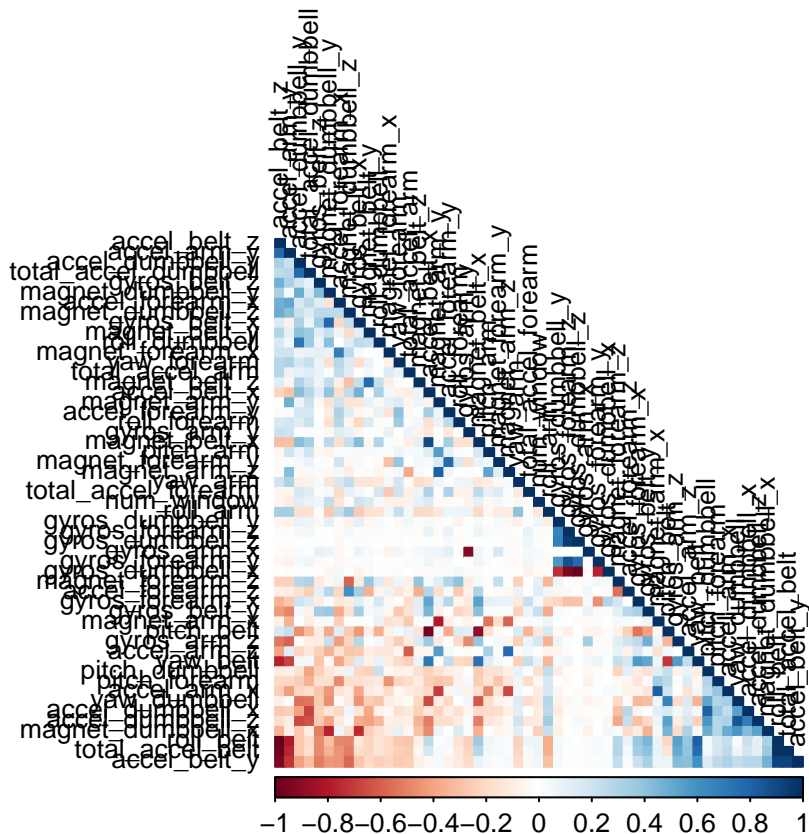
```
## [1] 5885 54
```

With the cleaning process above, the number of variables for the analysis has been reduced to 54 only.

Correlation Analysis

A correlation among variables is analysed before proceeding to the modeling procedures.

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
          tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



The highly correlated variables are shown in dark colors in the graph above. To make an even more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets. Nevertheless, as the correlations are quite few, this step will not be applied for this assignment.

Prediction Model Building

Three methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

a) Method: Random Forest

```
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                           trControl=controlRF)
modFitRandForest$finalModel

##
## Call:
## randomForest(x = x, y = y, mtry = param$mtry)
##           Type of random forest: classification
##           Number of trees: 500
```

```

## No. of variables tried at each split: 27
##
##          OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904     1     0     0     1 0.0005120328
## B     5 2652     1     0     0 0.0022573363
## C     0     5 2390     1     0 0.0025041736
## D     0     0     7 2245     0 0.0031083481
## E     0     1     0     5 2519 0.0023762376

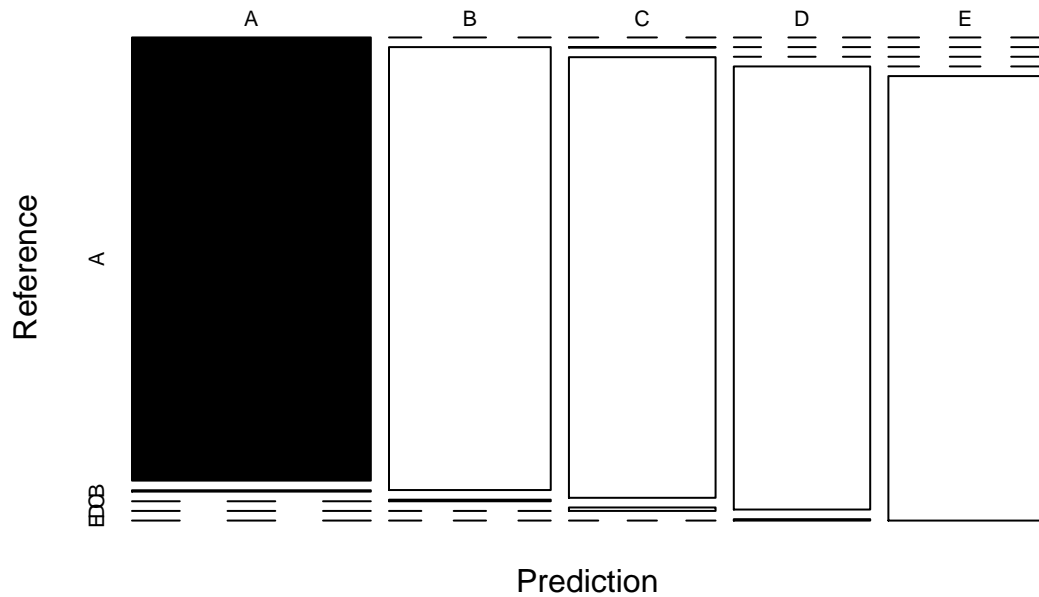
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 1674     5     0     0     0
##          B     0 1133     4     0     0
##          C     0     1 1022     8     0
##          D     0     0     0  956     3
##          E     0     0     0     0 1079
##
## Overall Statistics
##
##          Accuracy : 0.9964
##          95% CI : (0.9946, 0.9978)
##          No Information Rate : 0.2845
##          P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.9955
##          McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: A Class: B Class: C Class: D Class: E
## Sensitivity      1.0000   0.9947   0.9961   0.9917   0.9972
## Specificity      0.9988   0.9992   0.9981   0.9994   1.0000
## Pos Pred Value   0.9970   0.9965   0.9913   0.9969   1.0000
## Neg Pred Value    1.0000   0.9987   0.9992   0.9984   0.9994
## Prevalence       0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate    0.2845   0.1925   0.1737   0.1624   0.1833
## Detection Prevalence 0.2853   0.1932   0.1752   0.1630   0.1833
## Balanced Accuracy 0.9994   0.9969   0.9971   0.9955   0.9986

# plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMatRandForest$overall['Accuracy'], 4)))

```

Random Forest – Accuracy = 0.9964



b) Method: Decision Trees

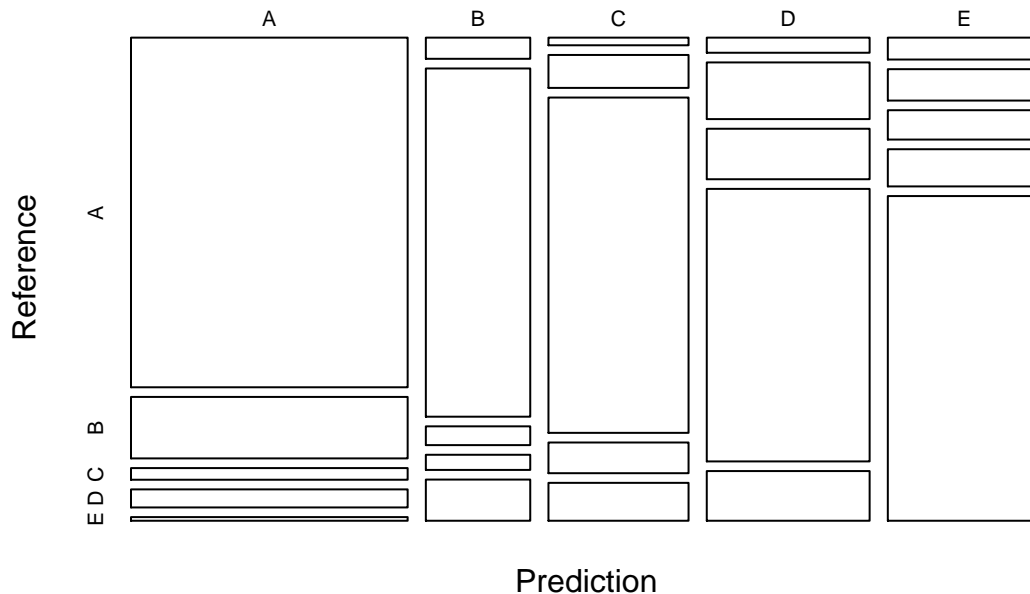
```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)
```

Warning: labs do not fit even at cex 0.15, there may be some overplotting


```
##                               Class: A Class: B Class: C Class: D Class: E
## Sensitivity                   0.9140  0.50483  0.7242  0.7282  0.7264
## Specificity                   0.9014  0.96650  0.9502  0.9100  0.9396
## Pos Pred Value                0.7866  0.78338  0.7543  0.6131  0.7305
## Neg Pred Value                0.9635  0.89051  0.9422  0.9447  0.9384
## Prevalence                    0.2845  0.19354  0.1743  0.1638  0.1839
## Detection Rate                0.2600  0.09771  0.1263  0.1193  0.1336
## Detection Prevalence          0.3305  0.12472  0.1674  0.1946  0.1828
## Balanced Accuracy              0.9077  0.73566  0.8372  0.8191  0.8330

# plot matrix results
plot(confMatDecTree$table, col = confMatDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(confMatDecTree$overall['Accuracy'], 4)))
```

Decision Tree – Accuracy = 0.7368



c) Method: Generalized Boosted Model

```
# model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM <- train(classe ~ ., data=TrainSet, method = "gbm",
                  trControl = controlGBM, verbose = FALSE)
```

```
## Loading required package: gbm
## Loading required package: survival
```



```
##
## Attaching package: 'survival'

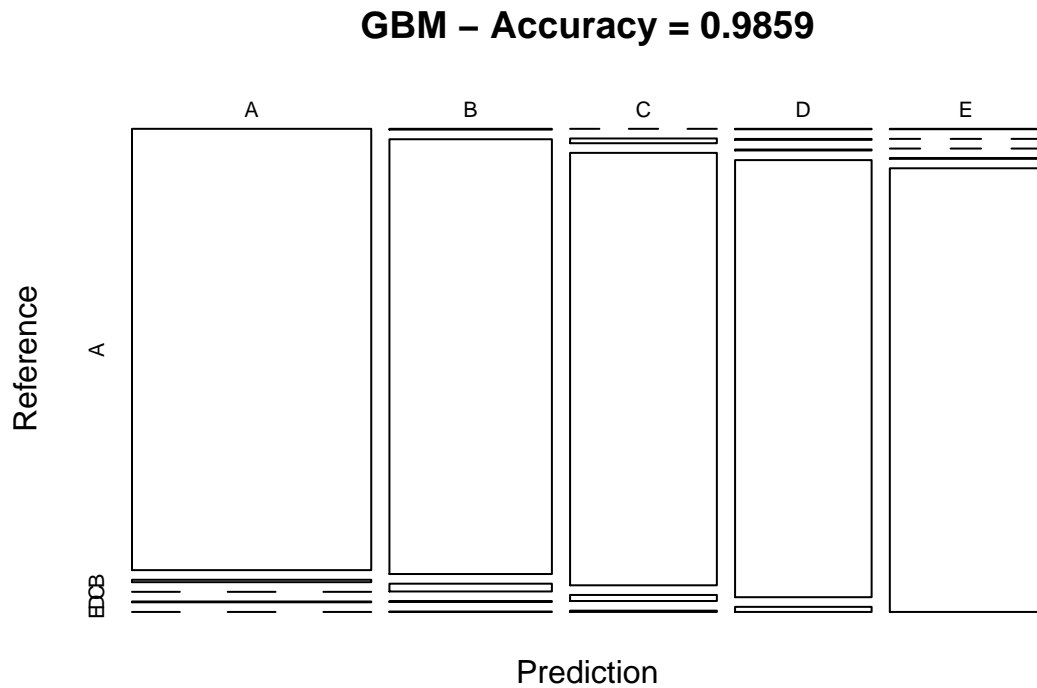
## The following object is masked from 'package:caret':
##
##      cluster

## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr
modFitGBM$finalModel

## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 43 had non-zero influence.
# prediction on Test dataset
predictGBM <- predict(modFitGBM, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, TestSet$classe)
confMatGBM

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    A    B    C    D    E
##      A 1670    9    0    3    0
##      B   2 1117   20    2    1
##      C    0   11 1004   14    3
##      D    1    2    2  944   11
##      E    1    0    0    1 1067
##
## Overall Statistics
##
##              Accuracy : 0.9859
##              95% CI : (0.9825, 0.9888)
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9822
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.9976  0.9807  0.9786  0.9793  0.9861
## Specificity          0.9972  0.9947  0.9942  0.9967  0.9996
## Pos Pred Value       0.9929  0.9781  0.9729  0.9833  0.9981
## Neg Pred Value       0.9990  0.9954  0.9955  0.9959  0.9969
## Prevalence           0.2845  0.1935  0.1743  0.1638  0.1839
## Detection Rate       0.2838  0.1898  0.1706  0.1604  0.1813
## Detection Prevalence 0.2858  0.1941  0.1754  0.1631  0.1816
## Balanced Accuracy    0.9974  0.9877  0.9864  0.9880  0.9929
```

```
# plot matrix results
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'], 4)))
```



Applying the Selected Model to the Test Data

The accuracy of the 3 regression modeling methods above are:

1.Random Forest : 0.9963 2.Decision Tree : 0.7368 3.GBM : 0.9839

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

```
predictTEST <- predict(modFitRandForest, newdata=testing)
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
```