Pratical Machine Learning

Fernanda H. P. Gomes

August 26, 2018

Overview

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 (see the section on the Weight Lifting Exercise Dataset).

Loading/cleaning the Raw Data

The data for this project come from this source

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)

```
setwd("D:/Internal Audit/R&D/Fernanda/Practice/R/Pratical Machine Learning")
train<- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
test<- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))

dim(train)
```

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

```
dim (test)
```

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

Data Partitioning

```
# Load the packages
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.3.3
```

```
## 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 2.1.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.3
```

```
library(rpart.plot)

# Set the seed
set.seed(1)

# remove variables with nearly 0
train <-train[,colSums(is.na(train)) == 0]
Test <-test[,colSums(is.na(test)) == 0]

# Delete unused columns
train <- train [,-c(1:7)]
test<- test [,-c(1:7)]

# remaining data
dim(train)
```

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

```
dim(test)
```

```
## [1]  20 153
```

```
# Partioning the data set
inTrain <- createDataPartition (train$classe, p=0.70, list=F)
train_data <- train[inTrain,]
test_data <- train[-inTrain,]
```

Data Prediction and Modeling

Testing Radom Forest model

```
# Defyning the best model fit
model_1 <- randomForest(classe ~., data=train_data, method="class")
model_1
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train_data, method = "class")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.51%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    5    1    0    1 0.001792115
## B   11 2643    4    0    0 0.005643341
## C    0   12 2381    3    0 0.006260434
## D    0    0   21 2230    1 0.009769094
## E    0    0    4    7 2514 0.004356436
```

```
# Predicting
prediction_1 <- predict (model_1, test_data, Type="class")

# Testing
confusionMatrix(prediction_1, test_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    0
##          B    2 1137    6    0    0
##          C    0    1 1019    5    0
##          D    0    0    1  958    3
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9982   0.9932   0.9938   0.9972
## Specificity            0.9998   0.9983   0.9988   0.9992   0.9998
## Pos Pred Value         0.9994   0.9930   0.9941   0.9958   0.9991
## Neg Pred Value         0.9995   0.9996   0.9986   0.9988   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1932   0.1732   0.1628   0.1833
## Detection Prevalence   0.2843   0.1946   0.1742   0.1635   0.1835
## Balanced Accuracy      0.9993   0.9983   0.9960   0.9965   0.9985
```

Testing Generalized Boosted Model

```
# Fitting model
fitControl <- trainControl(method="repeatedcv", number=5, repeats=1)
model_2 <- train(classe ~., data=train_data, method="gbm", trControl=fitControl, verbose=FALSE)
model_2
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10990, 10989, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7469615  0.6791158
##   1                  100      0.8191746  0.7711101
##   1                  150      0.8540441  0.8152629
##   2                   50      0.8539710  0.8149659
##   2                  100      0.9050739  0.8798668
##   2                  150      0.9296063  0.9109132
##   3                   50      0.8952464  0.8673672
##   3                  100      0.9386331  0.9223579
##   3                  150      0.9582151  0.9471325
##
## 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.
```

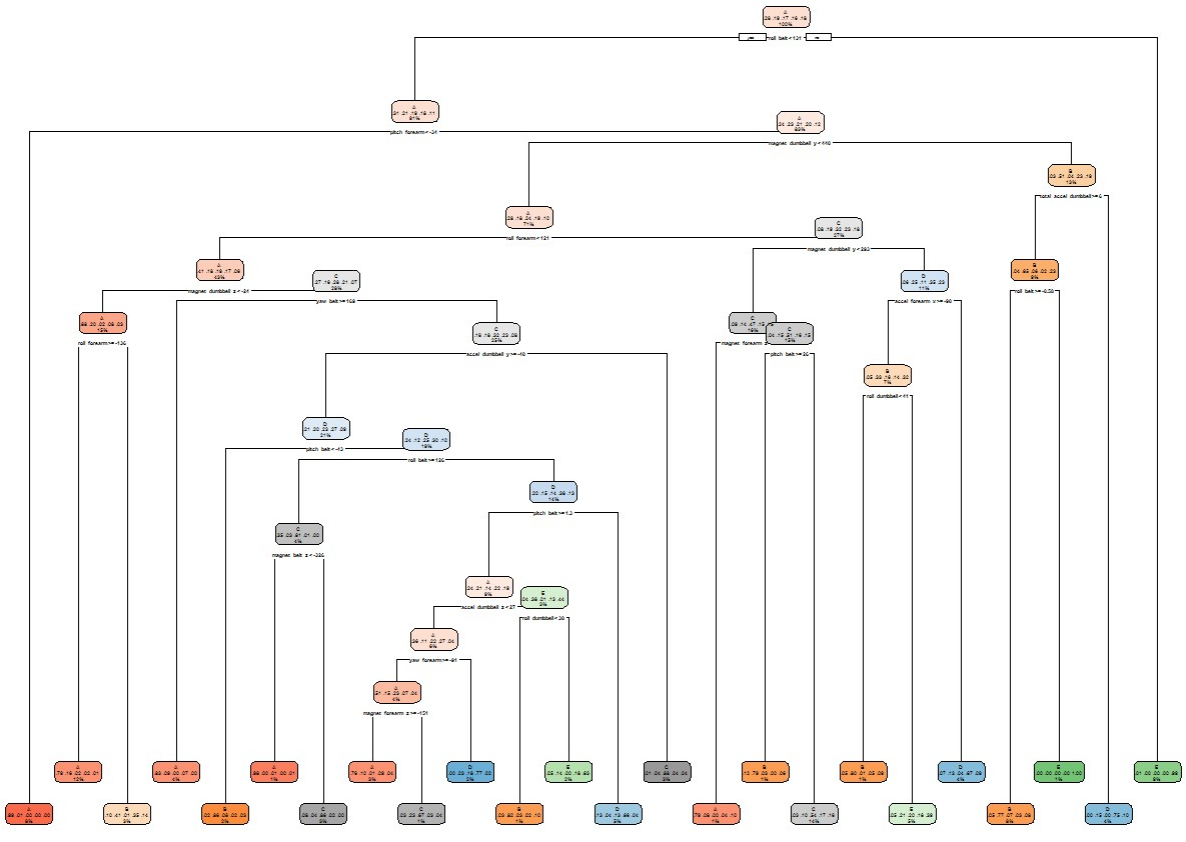
```
# Predicting
prediction_2 <- predict(model_2, test_data)

# Testing
confusionMatrix(prediction_2, test_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1637   35    0    1    1
##          B   18 1077   31    5    8
##          C    8   27  983   22   11
##          D    9    0   10  927   10
##          E    2    0    2    9 1052
##
## Overall Statistics
##
##                Accuracy : 0.9645
##                  95% CI : (0.9594, 0.9691)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9551
##  Mcnemar's Test P-Value : 2.974e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9779   0.9456   0.9581   0.9616   0.9723
## Specificity            0.9912   0.9869   0.9860   0.9941   0.9973
## Pos Pred Value         0.9779   0.9456   0.9353   0.9697   0.9878
## Neg Pred Value         0.9912   0.9869   0.9911   0.9925   0.9938
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2782   0.1830   0.1670   0.1575   0.1788
## Detection Prevalence   0.2845   0.1935   0.1786   0.1624   0.1810
## Balanced Accuracy      0.9846   0.9663   0.9720   0.9779   0.9848
```

Testing Decision tree

```
# Fitting model
model_3 <- rpart(classe ~ ., data=train_data, method="class")
rpart.plot(model_3)
```

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



```
# Predicting
prediction_3 <- predict(model_3, test_data, type="class")

# Testing
confusionMatrix(prediction_3, test_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1473  160   24   44   15
##          B   55  699   56   73  101
##          C   57  121  818  161  132
##          D   56   82   60  609   53
##          E   33   77   68   77  781
##
## Overall Statistics
##
##                Accuracy : 0.7443
##                  95% CI : (0.7329, 0.7554)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6764
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8799   0.6137   0.7973   0.6317   0.7218
## Specificity            0.9423   0.9399   0.9031   0.9490   0.9469
## Pos Pred Value         0.8584   0.7104   0.6346   0.7081   0.7539
## Neg Pred Value         0.9518   0.9102   0.9547   0.9294   0.9379
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2503   0.1188   0.1390   0.1035   0.1327
## Detection Prevalence   0.2916   0.1672   0.2190   0.1461   0.1760
## Balanced Accuracy      0.9111   0.7768   0.8502   0.7904   0.8344
```

Applying the selected Model to the Test Data

Based on the accuracy of the 3 models above: random forest- 0.9971 , GBM - 0.9643, and decision tree - 0.7443 ; the selected model to be used for the prediction is the Random FOrest model (model_1).

Below are the results for the predictions on the test data:

```
Prediction <- predict (model_1, test)
Prediction
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```