Practical Machine Learning Course Project

Ram Bhatta

Septemper 4, 2016

Executive summary

A large amount of data about personal activity can be collected using devices such as Jawbone Up, Nike FuelBand, and Fitbit. From these data, people can quantify how much of a particular activity they do. However, they rarely quantify how well they do it. In this course project, our goal is to use the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and analyse them. This project is focused on building model, using cross validation, estimation of expected sample error and reasons for making theose choices.

Data source

The training data for this project are available at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and the test data are available at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv. The data for this project come from http://groupware.les.inf.puc-rio.br/har and further information can be found at http://groupware.les.inf.puc-rio.br/har.

Loading data and appropriate libraries

```
train <- read.csv("~/Desktop/coursera/machine_learning/proj/pml-training.csv")
test <- read.csv("~/Desktop/coursera/machine_learning/proj/pml-testing.csv")
library(AppliedPredictiveModeling)
library(caret)
```

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

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

```
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(lattice)
```

Data processing

Let us clean the data by removing NAs and unnecessary columns

```
myindex <- c(4, 7:11, 37:49, 60:68, 84:86)
ml_train <- train[,myindex]
ml_train$class <- train$classe
ml_test <- test[,myindex]
```

Data partitioning

Now, let us split the data into two sets: 80% training and 20% test sets. This will help us for performing cross validation on our model.

```
set.seed(12345)
ml_train80 <- createDataPartition(ml_train$class, p = 0.80, list = FALSE)
train80 <- ml_train[ml_train80,]
test20 <- ml_train[-ml_train80,]
```

Model building

Let us start with the dicision tree

```
DT <- train(
  class ~.,
  method = "rpart",
  data = train80)
```

```
## Loading required package: rpart
```

```
prediction <- predict(
  DT,
  test20)
print(confusionMatrix(prediction, test20$class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 693 139 153  59  27
##          B 221 483 183 249 214
##          C  94  76 256  17  38
##          D 104  61  92 291   5
##          E   4   0   0  27 437
##
## Overall Statistics
##
```

```
##                Accuracy : 0.5506
##                  95% CI : (0.5349, 0.5662)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4307
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6210   0.6364  0.37427  0.45257   0.6061
## Specificity            0.8653   0.7260  0.93053  0.92012   0.9903
## Pos Pred Value         0.6471   0.3578  0.53222  0.52622   0.9338
## Neg Pred Value         0.8517   0.8927  0.87565  0.89555   0.9178
## Prevalence             0.2845   0.1935  0.17436  0.16391   0.1838
## Detection Rate         0.1767   0.1231  0.06526  0.07418   0.1114
## Detection Prevalence   0.2730   0.3441  0.12261  0.14096   0.1193
## Balanced Accuracy      0.7432   0.6812  0.65240  0.68634   0.7982
```

Let us now perform Random forest to see if the present accuracy of dicision tree (~55%) can be enhanced.

```
RF <- randomForest(
  class ~.,
  data = train80)
prediction <- predict(
  RF,
  test20,
  type = "class")
print(confusionMatrix(prediction, test20$class))
```

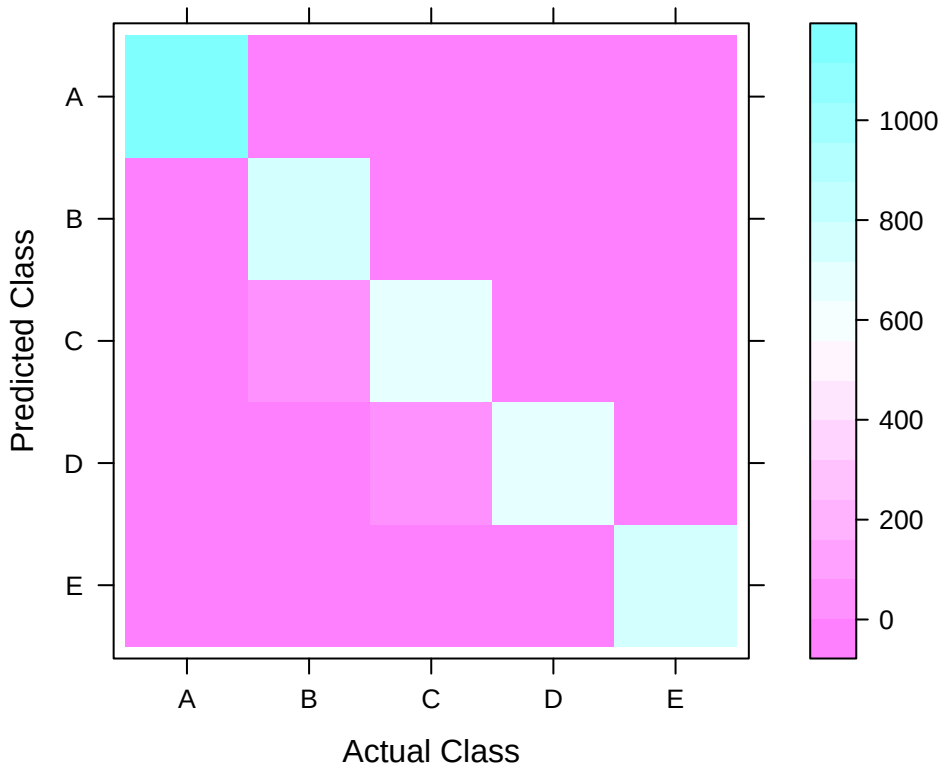
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    6    0    0
##          C    0    0  678    2    0
##          D    0    0    0  641    0
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.996, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9912   0.9969   1.0000
## Specificity            1.0000   0.9981   0.9994   1.0000   1.0000
## Pos Pred Value         1.0000   0.9922   0.9971   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9981   0.9994   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1728   0.1634   0.1838
## Detection Prevalence   0.2845   0.1950   0.1733   0.1634   0.1838
## Balanced Accuracy      1.0000   0.9991   0.9953   0.9984   1.0000
```

Model assessment

Comparing the above mentioned models, Random forest appears to be the best model because it gives above 99% accuracy.

```
predictionf <- read.table("myprediction", header = TRUE)
row.names(predictionf) <- predictionf$nn
predictionf <- predictionf[,2:6]
predictionf_matrix <- data.matrix(predictionf)
levelplot(predictionf_matrix[1:ncol(predictionf_matrix),ncol(predictionf_matrix):1], xlab="Actual Class
```



Model decision

Based on the above data analysis and cross-validation of models, we see that the Random Forest algorithm appears to be superior compared to other models. It yields above 99% accuracy and the out-of-sample error is low (~ 0.2%).

Applying model on 20 testing sets

```
print(predict(RF, newdata=test))
```

```
##  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
```