Practical Machine Learning: Course Project

Pin Choengtawee

June 19, 2016

Exceutive Summary

The goal of this report was building a predictive model to predict the manner in which participants did the exercise by using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. After performed data loading and data cleaning, first step was to devide data into two sets: which are trainging set and testing set. Next was to build a fit model by using the Random Forest method and apply the select model to the test set. As a result, The predictive model developed using Random Forest was able to achieve over 99.99% accuracy, or less than 0.03% out-of-sample error, and was able to predict the 20 test cases with 100% accuracy.

Data Loading and Exploratory Analysis

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)

```
df <- "pml_training.csv"
if (!file.exists(df)) {
    url <-
        "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
    df <- "pml_training.csv"
    download.file(url, destfile = df)
}
training <- read.csv(df, na.strings = c("NA","#DIV/0!",""))

df <- "pml_testing.csv"
if (!file.exists(df)) {
    url <-
        "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
    download.file(url, destfile = df)
}
testing <- read.csv(df, na.strings = c("NA","#DIV/0!",""))
```

Data Cleaning

Data cleaning includes removing columns with Near Zero Values, Removing columns with NA or is empty, and removing V1 which seems to be a serial number.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.4
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
# Remove columns with Near Zero Values
subTrain <-
    training[, names(training)[!(nzv(training, saveMetrics = T)[, 4])]]

# Remove columns with NA or is empty
subTrain <-
    subTrain[, names(subTrain)[sapply(subTrain, function (x)
        ! (any(is.na(x) | x == "")))]]

# Remove V1 which seems to be a serial number
subTrain <- subTrain[,-1]
subTrain <- subTrain[, c(1:3, 5:58)]
```

Prediction Model Building

Befor building the prediction model, devide the data set into two sets: a training set and a validation/test set

```
inTrain <- createDataPartition(subTrain$classe, p = 0.6, list = FALSE)
TrainSet <- subTrain[inTrain,]
ValidationSet <- subTrain[-inTrain,]
dim(TrainSet)
```

```
## [1] 11776    57
```

```
dim(ValidationSet)
```

```
## [1] 7846   57
```

a) Building a fit model by using a Random Forest method

```
#install.packages("doParallel")
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
# Check if model file exists
model <- "modelFit.RData"
if (!file.exists(model)) {

    # If not, set up the parallel clusters.
    require(parallel)
    require(doParallel)
    cl <- makeCluster(detectCores() - 1)
    registerDoParallel(cl)

    fit <- train(TrainSet$classe ~ ., method = "rf", data = TrainSet)
    save(fit, file = "modelFit.RData")

    stopCluster(cl)
} else {
    # Good model exists from previous run, load it and use it.
    load(file = "modelFit.RData", verbose = TRUE)
}
```

```
## Loading objects:
##   fit
```

b) Applying the selected model to the validation/test data

```
predTrain <- predict(fit, TrainSet)
```

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

```
confusionMatrix(predTrain, TrainSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    1    0    0
##          C    0    0 2053    3    0
##          D    0    0    0 1927    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9991, 0.9999)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9995   0.9984   1.0000
## Specificity            1.0000   0.9999   0.9997   1.0000   1.0000
## Pos Pred Value         1.0000   0.9996   0.9985   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9999   0.9997   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1743   0.1636   0.1838
## Detection Prevalence   0.2843   0.1936   0.1746   0.1636   0.1838
## Balanced Accuracy      1.0000   0.9999   0.9996   0.9992   1.0000
```

```
predValidation <- predict(fit, ValidationSet)
confusionMatrix(predValidation, ValidationSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1516    1    0    0
##          C    0    1 1367    3    0
##          D    0    0    0 1283    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9983, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9993   0.9977   1.0000
## Specificity            0.9998   0.9998   0.9994   1.0000   1.0000
## Pos Pred Value         0.9996   0.9993   0.9971   1.0000   1.0000
## Neg Pred Value         1.0000   0.9997   0.9998   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1742   0.1635   0.1838
## Detection Prevalence   0.2846   0.1933   0.1747   0.1635   0.1838
## Balanced Accuracy      0.9999   0.9993   0.9993   0.9988   1.0000
```

From the validation subset, the accuracy is still very high, at above 99%. Given the level of accuracy, there is no need to build another prediction model for better accuracy or to stack multiple prediction models. The following is the lists of important predictors in the model.

```
varImp(fit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 60)
##
##                      Overall
## raw_timestamp_part_1 100.000
## num_window            54.144
## roll_belt             45.436
## pitch_forearm         28.385
## magnet_dumbbell_z     21.020
## yaw_belt              19.157
## magnet_dumbbell_y     17.935
## pitch_belt            17.498
## roll_forearm          12.365
## roll_dumbbell          8.275
## accel_belt_z           7.662
## magnet_dumbbell_x      7.565
## accel_dumbbell_y       7.447
## accel_forearm_x        6.340
## magnet_belt_y          5.678
## total_accel_dumbbell   5.463
## magnet_belt_z          5.457
## accel_dumbbell_z       5.323
## yaw_dumbbell           3.603
## magnet_belt_x          3.216
```

```
fit$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: 31
##
##         OOB estimate of  error rate: 0.13%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3348    0    0    0    0 0.0000000000
## B    4 2274    1    0    0 0.0021939447
## C    0    2 2051    1    0 0.0014605648
## D    0    0    4 1924    2 0.0031088083
## E    0    0    0    1 2164 0.0004618938
```

Even though the reported OOB Estimated Error is at 13%, the validation accuracy is high at over 99% and Cross-Validation out-of-sample error rate is only 0.03%. Thus, the prediction model should be applied to the final testing set and predict the classe in the 20 test cases.

```
predTesting <- predict(fit, testing)
predTesting
```

```
##  [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
```

Create Files

```
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
}

pml_write_files(predTesting)
```