Pratical Machine Learning Project

Lingna Chai

May 27, 2018

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. 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. This project is 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

The goal of the project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set.

Data Import and Cleanup

For the purpose of this project, import below R packages first.

Data set can be found from below URL link. Download the Training and Testing Data set from the links and identify NULL values.

```
#rm(list=ls())
url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training = read.csv(url(url1),na.strings = c("NA","#DIV/0!",""))
testing = read.csv(url(url2),na.strings = c("NA","#DIV/0!",""))
```

For modeling purpose, further split the Training data set into trainingset (70%) and testingset (30%) for validation.

```
inTrain <- createDataPartition(training$classe,p=0.7,list=FALSE)
trainingset <- training[inTrain,]
testingset <- training[-inTrain,]

dim(trainingset)
```

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

Total of 160 variables including target variable “classe” available in the Training set. This would be too many variables to use for modeling. Exclude variables that has null measures and only focus on variables that are fully populated. Further exclude identifier variables that is not required for modeling. Total variables decreased from 160 to 46 after cleanup.

```
NA_count <- sapply(1:dim(trainingset)[2],function(x)sum(is.na(trainingset[,x])))
remove_list <-c(1:7,which(NA_count>0))

trainingset <- trainingset[,-remove_list]
testingset <-testingset[,-remove_list]
dim(trainingset)
```

```
## [1] 13737    53
```

```
#str(trainingset)
```

Modeling and Validation

Use random forest method and train the model using Traningset. The calculation took a rather long time. To avoid multiple calculation, calculated model data is saved under local file “all.RData” (calculated from the commented train command below). Apply the trained random forest model onto testing set. Below is the output result from validation.

Accuracy on the testingset is 99%.

```
load("all.RData")
#mod1 <- train(classe~., data = trainingset, method = "rf", trControl= trainControl(method="cv"), number=4)
pred1 <- predict(mod1,testingset)
confusionMatrix(pred1,testingset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671   13    0    0    0
##          B    3 1125    7    2    0
##          C    0    1 1017    7    2
##          D    0    0    2  954    6
##          E    0    0    0    1 1074
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.99, 0.9946)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9877   0.9912   0.9896   0.9926
## Specificity            0.9969   0.9975   0.9979   0.9984   0.9998
## Pos Pred Value         0.9923   0.9894   0.9903   0.9917   0.9991
## Neg Pred Value         0.9993   0.9971   0.9981   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1912   0.1728   0.1621   0.1825
## Detection Prevalence   0.2862   0.1932   0.1745   0.1635   0.1827
## Balanced Accuracy      0.9976   0.9926   0.9946   0.9940   0.9962
```

Prediction

Apply the same random forest model on Testing data set. And the result 100% matches with the actual result (from course quiz).

```
pred_test= predict(mod1,testing)
pred_test
```

```
##  [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
```