Prediction Assignment Writeup

Haoyi He

9/24/2020

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

Data

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 this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Data Processing

Packages being used.

```r
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
```

First, download the traning and testing dataset from Link provided above.

```r
set.seed(9898)

trainurl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testurl  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
train_data <- read.csv(url(trainurl), strip.white = TRUE, na.strings = c("NA",""))
test_data  <- read.csv(url(testurl),  strip.white = TRUE, na.strings = c("NA",""))

dim(train_data)
```

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

Then, Splitting the training data into traing set(70%) and validataion set(30%).

```
in_train  <- createDataPartition(train_data$classe, p=.7, list=FALSE)
train_set <- train_data[ in_train, ]
test_set  <- train_data[-in_train, ]

dim(train_set)
```

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

```
dim(test_set)
```

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

Next, clean up the varaibles with no variance.

```
nearzerodata <- nearZeroVar(train_data)
train_set_final <- train_set[,-nearzerodata]
test_set_final <- test_set[,-nearzerodata]

dim(train_set_final)
```

```
## [1] 13737   117
```

```
dim(test_set_final)
```

```
## [1] 5885  117
```

By looking at the dataset, it appears that some of the column have NA values mostly, we can trim the dataset further more.

```
train_set_final <- train_set_final[, colSums(is.na(train_set_final)) == 0]
test_set_final <- test_set_final[, colSums(is.na(test_set_final)) == 0]

# removed non related columns
train_set_final <- train_set_final[,-(1:5)]
test_set_final <- test_set_final[,-(1:5)]
```

From this point, we can use the trimed data to build models.

Data Modeling

use K-fold Cross Validation for 3 iterations while traning the dataset.

```r
# Random forest model
set.seed(9898)
fit_rf  <- train( classe ~.,
                   data = train_set_final,
                   method = "rf",
                   trControl = trainControl(method="cv", number=3)
                   )

# Generalized boosted model
set.seed(9898)
fit_bm <- train( classe ~.,
                  data = train_set_final,
                  method = "gbm",
                  trControl = trainControl(method="cv", number=3),
                  verbose = FALSE)
```

Build predictions.

```r
prf = predict(fit_rf,test_set_final)
pbm = predict(fit_bm,test_set_final)
```

Validating model accuracy

```r
#confusion matrix for Random forest model
confusionMatrix(prf,as.factor(test_set_final$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    5    0    0    0
##          B    0 1131   11    0    0
##          C    0    2 1015    2    0
##          D    0    1    0  961    1
##          E    1    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9939, 0.9974)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9930   0.9893   0.9969   0.9991
## Specificity            0.9988   0.9977   0.9992   0.9996   0.9996
## Pos Pred Value         0.9970   0.9904   0.9961   0.9979   0.9982
## Neg Pred Value         0.9998   0.9983   0.9977   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1922   0.1725   0.1633   0.1837
## Detection Prevalence   0.2851   0.1941   0.1732   0.1636   0.1840
## Balanced Accuracy      0.9991   0.9953   0.9942   0.9982   0.9993
```

```
#confusion matrix for Generalized boosted model
confusionMatrix(pbm,as.factor(test_set_final$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    5    0    1    0
##          B    4 1123   20    5    2
##          C    0   11 1004   12    2
##          D    0    0    2  944    9
##          E    0    0    0    2 1069
##
## Overall Statistics
##
##                Accuracy : 0.9873
##                  95% CI : (0.9841, 0.99)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9839
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9860   0.9786   0.9793   0.9880
## Specificity            0.9986   0.9935   0.9949   0.9978   0.9996
## Pos Pred Value         0.9964   0.9731   0.9757   0.9885   0.9981
## Neg Pred Value         0.9990   0.9966   0.9955   0.9959   0.9973
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1908   0.1706   0.1604   0.1816
## Detection Prevalence   0.2848   0.1961   0.1749   0.1623   0.1820
## Balanced Accuracy      0.9981   0.9897   0.9867   0.9885   0.9938
```

As we see from above, **random forest model** has better accuracy(0.998) compare to **generalized boosted model**(0.9885).

Predict test cases

Finally, predict the test cases based on testing dataset by using random forest model.

```
test_data_final <- test_data[,-nearzerodata]
test_data_final <- test_data_final[, colSums(is.na(test_data_final)) == 0]
```

```r
test_data_final <- test_data_final[,-(1:5)]

# predict the test dateset
finalpredict = predict(fit_rf,test_data_final)

print(finalpredict)
```

```
##  [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
```