Prediction Assignment Writeup

MsGret

September 25, 2020

Introduction

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**.

Data Preprocessing

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Read the Data

Read the data:

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

trainSet <- read.csv(url(trainUrl), na.strings = c("NA", ""))
dim(trainSet)
```

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

```
testSet <- read.csv(url(testUrl), na.strings = c("NA", ""))
dim(testSet)
```

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

Clean the Data

Remove variables with NA value and some useless variables (`X`, `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`, `new_window`, `num_window`):

```
notNA <- colSums(is.na(trainSet)) == 0
trainSet <- trainSet[, notNA]
testSet <- testSet[, notNA]

removeVar <- grep("^X|user|timestamp|window", names(trainSet))
trainSet <- trainSet[, -removeVar]
testSet <- testSet[, -removeVar]

dim(trainSet)
```

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

```r
dim(testSet)
```

```
## [1] 20 53
```

Cleaned train data set contains 19622 observations and 53 variables, and the test data set contains 20 observations and 53 variables. **Appendix A** shows correlation matrix of these 53 variables.

Split the Data

Separate our `trainSet` into `traning` and `dev` (development) sets for Machine Learning

```r
library(caret)
```

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

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

```r
set.seed(2020)

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

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

```r
dev <- trainSet[-inTrain, ]
dim(dev)
```

```
## [1] 5885   53
```

Prediction Models

Compare prediction based on the two models:

- generalized boosted model;
- random forest.

Generalized Boosted Model

We will use **5-fold cross validation** when applying the algorithm.

```r
set.seed(2020)
ctrl <- trainControl(method = "cv", number = 5)
gbmModel <- train(classe ~ ., data = training, method = "gbm",
                  trControl = ctrl,
                  verbose = FALSE)

gbmPrediction <- predict(gbmModel, dev)
length(gbmPrediction)
```

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

```r
length(dev$classe)
```

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

```r
confusionMatrix(table(gbmPrediction, dev$classe))
```

```
## Confusion Matrix and Statistics
##
##
## gbmPrediction    A    B    C    D    E
##             A 1647   45    0    1    3
##             B   19 1058   36    4   14
##             C    4   36  982   42   11
##             D    4    0    6  916   12
##             E    0    0    2    1 1042
##
## Overall Statistics
##
##                Accuracy : 0.9592
##                  95% CI : (0.9538, 0.9641)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9484
##
##  Mcnemar's Test P-Value : 5.25e-13
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9289   0.9571   0.9502   0.9630
## Specificity            0.9884   0.9846   0.9809   0.9955   0.9994
## Pos Pred Value         0.9711   0.9355   0.9135   0.9765   0.9971
## Neg Pred Value         0.9936   0.9830   0.9909   0.9903   0.9917
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1798   0.1669   0.1556   0.1771
## Detection Prevalence   0.2882   0.1922   0.1827   0.1594   0.1776
## Balanced Accuracy      0.9861   0.9568   0.9690   0.9729   0.9812
```

Random Forest

"Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees… A random forest dissimilarity can be attractive because it handles mixed variable types very well, is invariant to monotonic transformations of the input variables, and is robust to outlying observations." https://www.wikiwand.com/en/Random_forest

```
set.seed(2020)
ctrl <- trainControl(method = "cv", number = 5)
rfModel <- train(classe ~ ., data = training, method = "rf", trControl = ctrl)
rfPrediction <- predict(rfModel, dev)
confusionMatrix(table(rfPrediction, dev$classe))
```

```
## Confusion Matrix and Statistics
##
##
## rfPrediction    A    B    C    D    E
##            A 1672   10    0    0    0
##            B    0 1127   11    0    6
##            C    1    2 1005    5    3
##            D    0    0   10  959    2
##            E    1    0    0    0 1071
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9913
##                  95% CI : (0.9886, 0.9935)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.989
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9895   0.9795   0.9948   0.9898
## Specificity            0.9976   0.9964   0.9977   0.9976   0.9998
## Pos Pred Value         0.9941   0.9851   0.9892   0.9876   0.9991
## Neg Pred Value         0.9995   0.9975   0.9957   0.9990   0.9977
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1915   0.1708   0.1630   0.1820
## Detection Prevalence   0.2858   0.1944   0.1726   0.1650   0.1822
## Balanced Accuracy      0.9982   0.9929   0.9886   0.9962   0.9948
```

Conclusion and Final Prediction

Random forest has better performance (Accuracy: 0.9913) than the generalized boosted model (Accuracy: 0.9592).

Let's test `rfModel` in the `testSet`:

```
rfPrediction <- predict(rfModel, testSet)
rfPrediction
```

```
##  [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
```