ML_Project

Chenyu

5/26/2020

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).

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
```

Objective

Predict a variable from "classe"

Process data

Read data from the given website

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "~/Learning/training.csv"
testFile  <- "~/Learning/testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile, method="curl")
}
if (!file.exists(testFile)) {
```

```
  download.file(testUrl, destfile=testFile, method="curl")
}
```

Glimse at the data first

```
train <- read.csv("~/Learning/training.csv")
test <- read.csv("~/Learning/testing.csv")
dim(train)
```

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

```
dim(test)
```

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

```
sum(complete.cases(train))
```

```
## [1] 406
```

We could find we got 160 variables, but many of them is missing or meaningless, thus we could clean up the data before learning.

```
train_df <- train[, colSums(is.na(train)) == 0]
test_df <- test[, colSums(is.na(test)) == 0]

classe <- train_df$classe

trainRemove <- grepl("^X|timestamp|window", names(train_df))
train_df <- train_df[, !trainRemove]
trainCleaned <- train_df[, sapply(train_df, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(test_df))
test_df <- test_df[, !testRemove]
testCleaned <- test_df[, sapply(test_df, is.numeric)]

dim(trainCleaned)
```

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

```
dim(testCleaned)
```

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

We further divide the cleaned training data set into training vs. validation set.

```
set.seed(123)

train_ind <- sample(seq_len(nrow(train_df)),size = floor(0.75*nrow(train_df)))

train_set <- trainCleaned[train_ind, ]
validate_set <- trainCleaned[-train_ind, ]
```

Build Model

```
model <- randomForest(classe ~., data=train_set, type="class")
summary(model)
```

```
##                 Length Class  Mode
## call               4  -none- call
```

```
## type               1  -none- character
## predicted      14716  factor numeric
## err.rate        3000  -none- numeric
## confusion         30  -none- numeric
## votes          73580  matrix numeric
## oob.times      14716  -none- numeric
## classes            5  -none- character
## importance        52  -none- numeric
## importanceSD       0  -none- NULL
## localImportance    0  -none- NULL
## proximity          0  -none- NULL
## ntree              1  -none- numeric
## mtry               1  -none- numeric
## forest            14  -none- list
## y              14716  factor numeric
## test               0  -none- NULL
## inbag              0  -none- NULL
## terms              3  terms  call
```

Cross Validation

```
confusionMatrix(predict(model, validate_set), validate_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1367    0    0    0    0
##          B    1  947    6    0    0
##          C    0    1  859    9    0
##          D    0    0    1  784    2
##          E    0    0    0    3  926
##
## Overall Statistics
##
##                Accuracy : 0.9953
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.2788
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9941
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9989   0.9919   0.9849   0.9978
## Specificity            1.0000   0.9982   0.9975   0.9993   0.9992
## Pos Pred Value         1.0000   0.9927   0.9885   0.9962   0.9968
## Neg Pred Value         0.9997   0.9997   0.9983   0.9971   0.9995
## Prevalence             0.2788   0.1932   0.1765   0.1623   0.1892
## Detection Rate         0.2786   0.1930   0.1751   0.1598   0.1887
## Detection Prevalence   0.2786   0.1945   0.1771   0.1604   0.1894
```

```
## Balanced Accuracy      0.9996   0.9986   0.9947   0.9921   0.9985
```

Predict 20 different test cases

```
test_result <- predict(model, test_df)
test_result
```

```
##  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
```