Prediction Assignment

Eunice Ang

September 30, 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.

a) Preparing the environment

Firstly, we load all the libraries that are required in our analysis.

```r
rm(list=ls())                # free up memory for the download of the data sets
library(knitr)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

```
## Warning: package 'lattice' was built under R version 3.6.3
```

```
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.3
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.6.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3

## Loading required package: tibble

## Warning: package 'tibble' was built under R version 3.6.3

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

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

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded
```

```r
set.seed(55)
```

b) Data Loading and Cleaning

Next, we load the dataset from the URLs provided above. The training dataset is split into a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations.

```r
# Create variables for the URLs for the download of data
Url_Train <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Url_Test <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Download the datasets
training <- read.csv(url(Url_Train))
testing  <- read.csv(url(Url_Test))

# Split the training dataset into test set and training set
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

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

```r
dim(TestSet)
```

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

Both created datasets have 160 variables. Now, we clean the data by removing NA values. The Near Zero variance (NZV) variables are also removed and the ID variables as well.

```r
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)
```

```
## [1] 13737   103
```

```r
dim(TestSet)
```

```
## [1] 5885  103
```

```r
# remove variables that are mostly NA
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]
dim(TrainSet)
```

```
## [1] 13737    59
```

```r
dim(TestSet)
```

```
## [1] 5885   59
```

```r
# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

```
## [1] 13737    54
```

```r
dim(TestSet)
```

```
## [1] 5885   54
```

After cleaning, the number of variables for the analysis has been reduced to 54 only.

c) Prediction Model Building

```r
set.seed(55)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
modFitRandForest$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: 27
##
##         OOB estimate of  error rate: 0.27%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2648    4    0    0 0.0037622272
## C    0    7 2387    2    0 0.0037562604
## D    0    0   10 2242    0 0.0044404973
## E    0    0    0    6 2519 0.0023762376
```

```r
#Prediction on TestSet
cvPred <- predict(modFitRandForest, TestSet)
confusionMatrix(cvPred, TestSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1134    0    0    2
##          C    0    1 1026    7    0
##          D    0    1    0  956    0
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9958, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9956   1.0000   0.9917   0.9982
## Specificity            0.9993   0.9996   0.9984   0.9998   0.9998
## Pos Pred Value         0.9982   0.9982   0.9923   0.9990   0.9991
## Neg Pred Value         1.0000   0.9989   1.0000   0.9984   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1927   0.1743   0.1624   0.1835
## Detection Prevalence   0.2850   0.1930   0.1757   0.1626   0.1837
## Balanced Accuracy      0.9996   0.9976   0.9992   0.9957   0.9990
```

d) Results

Random Forest model has an accuracy of 0.9975 and will be applied to predict the 20 quiz results (testing dataset) as shown below.

```r
testingPred <- predict(modFitRandForest, testing)
testingPred
```

```
##  [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
```