C8W4 Assignment

Nicholas Neo

12 July 2020

Problem Statement

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 uses 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. (See the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here. The test data are available here.

Goal

The goal of this project is to predict the manner in which the participants did the exercise. This is the **classe** variable in the training set. You may use any of the other variables to predict with. You should describe how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Load Data

```r
# Load packages
library(dplyr)
```

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(caret)
```

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

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

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

```r
library(gbm)
```

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

```
## Loaded gbm 2.1.5
```

```r
# Get data from the web
train.URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
train.filename <- "pml-training.csv"
test.URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
test.filename <- "pml-testing.csv"
if (!file.exists(train.filename)) {
  download.file(train.URL, train.filename, method = "curl")
}
if (!file.exists(test.filename)) {
  download.file(test.URL, test.filename, method = "curl")
}
# Read data
training <- read.csv(train.filename, na.strings = c("NA", ""))
testing <- read.csv(test.filename, na.strings = c("NA", ""))
```

Data cleaning and splitting

```r
# Drop all columns with NAs
training <- select_if(training, colSums(is.na(training)) == 0)
testing <- select_if(testing, colSums(is.na(testing)) == 0)
# Drop irrelevant columns which will not be good predictors.
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
# Split training data into training set and validation set
set.seed(7777)
train.part <- createDataPartition(training$classe, p = 0.7, list = FALSE)
training.sub <- training[train.part, ]
validation <- training[-train.part, ]
```

Model fitting and prediction with validation set

We will compare predictions with 2 models, gradient boosting (gbm) and random forests (rf). For each model fit, we will also conduct k-fold cross validation with 5 folds.

Gradient boosting

```r
# Fit model
gbm.fit <- train(classe ~ .,
                 data = training.sub,
                 method = "gbm",
                 trControl = trainControl(method = "cv", number = 5),
                 verbose = FALSE)
print(gbm.fit)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10991, 10989, 10989
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7548961  0.6891221
##   1                  100      0.8222337  0.7749864
##   1                  150      0.8566657  0.8186494
##   2                   50      0.8561555  0.8177910
##   2                  100      0.9066035  0.8818201
##   2                  150      0.9318639  0.9137946
##   3                   50      0.8936456  0.8653589
##   3                  100      0.9408897  0.9252051
##   3                  150      0.9606904  0.9502698
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Predict with the validation set.

```r
predict.gbm <- predict(gbm.fit, newdata = validation)
# Calculate out-of-sample accuracy
confusionMatrix(predict.gbm, validation$classe)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9595582
```

Accuracy is about 95%, the out-of-sample error is about 5%.

Random forests

```r
# Fit model
rf.fit <- train(classe ~ .,
                 data = training.sub,
                 method = "rf",
                 trControl = trainControl(method = "cv", number = 5),
                 verbose = FALSE)
print(rf.fit)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10990, 10991, 10988
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9912645  0.9889498
##   27    0.9907550  0.9883053
##   52    0.9809269  0.9758681
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Predict with the validation set.

```r
predict.rf <- predict(rf.fit, newdata = validation)
# Calculate out-of-sample accuracy
confusionMatrix(predict.rf, validation$classe)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9920136
```

Accuracy with random forests is about 99%, the out-of-sample error is about 1%.

Hence, Random Forest is chosen to predict the test set since it has a higher accuracy as compared to gradient boosting

Predict with test set using Random Forest

```r
predict.test <- predict(rf.fit, newdata = testing)
predict.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
```