Coursera: Practical Machine Learning Prediction Assignment

C.Wang

Dec 02, 2016

Aim

Nowadays, heath or fitness tracking devices such as Jawbone Up, Nike FuelBand, and Fitbit allow people to collect a large amount of data about personal activity relatively inexpensively. However, most of the time people pay more attention to the "quantity" rather than the "quality" of a particular activity they do. The aim of this project is to predict how well people perform their activities by using the 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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Load packages that we need

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

Read in the data

```
training <- read.csv("~/R/Machine Learning/pml-training.csv", na.strings=c("NA",""))
testing <- read.csv("~/R/Machine Learning/pml-testing.csv", na.strings=c("NA",""))
dim(training)
```

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

```
dim(testing)
```

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

The training data set contains 19622 observations and 160 variables, and the testing data set contains 20 observations and 160 variables.

Data processing

First, we remove the missing values, and followed by removing variables that are irrelevant for prediction.

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
training <- training[,-c(1,2,3,4,5)]
testing <- testing[,-c(1,2,3,4,5)]
dim(training)
```

```
## [1] 19622    55
```

```
dim(testing)
```

```
## [1] 20 55
```

Spliting the data for cross-validation

Afterwards, we can split the training set into a pure training data part (70%) and a validation data part (30%).

```
set.seed(1500)
inTrain <- createDataPartition(training$classe, p=0.70, list=F)
trainData <- training[inTrain, ]
testData <- training[-inTrain, ]
```

Model Building and cross-validation

Next, we use the Random Forest algorithm for prediction.

```
model <- train(classe ~ ., data=trainData, method="rf", trControl = trainControl(method = "cv", number
model
```

```
## Random Forest
##
## 13737 samples
##    54 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10990, 10989, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9939582  0.9923571
##   28    0.9969428  0.9961326
##   54    0.9958505  0.9947509
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 28.
```

Then, we estimate the performance of the model with validation data set.

```
predict <- predict(model, testData)
confusionMatrix(testData$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    1
##          B    2 1136    0    1    0
##          C    0    3 1023    0    0
##          D    0    0    4  960    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.9964, 0.9989)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9974   0.9961   0.9979   0.9991
## Specificity            0.9998   0.9994   0.9994   0.9992   0.9998
## Pos Pred Value         0.9994   0.9974   0.9971   0.9959   0.9991
## Neg Pred Value         0.9995   0.9994   0.9992   0.9996   0.9998
## Prevalence             0.2846   0.1935   0.1745   0.1635   0.1839
## Detection Rate         0.2843   0.1930   0.1738   0.1631   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9993   0.9984   0.9977   0.9986   0.9994
```

Predicting for testing Data Set

Finally, we apply the model to the original testing data set.

```r
result <- predict(model, testing)
result
```

```
##  [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
```