trainingForActivity

Kalyan Pagadala

October 13, 2018

Learning Activites

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 (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Fetch:

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

```
suppressWarnings(suppressMessages(library(caret)))
suppressWarnings(suppressMessages(library(randomForest)))
suppressWarnings(suppressMessages(library(e1071)))

if(!file.exists("activityTrain.csv")){
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "activit
yTrain.csv")

  download.file("https://d396qusza40orc.cloudfront.net/pre  dmachlearn/pml-testing.csv", "activi
tyTest.csv")
}

train <- read.csv('activityTrain.csv', na.strings = c("NA", "#DIV/0!", ""))
test <- read.csv('activityTest.csv', na.strings = c("NA", "#DIV/0!", ""))
```

Cleaning the Data

The data when explored in an Excel sheet shows many NA values and columns which are mostly empty. Some columns do not have relevant information for the prdiction purposes, such as: serial numbers, date of the record etc…

Removing such (mostly) empty as well as irrelevant columns increases the accuracy of prediction.

```
dim(train)
```

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

```
dim(test)
```

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

```
#str(train)

#Remove the serial number and date/time columns
train <- train[, -c(1,3:7)]
test  <- test [, -c(1,3:7)]

#Remove columns with mostly NA values
train <- train[, colSums(is.na(train)) == 0]
test  <- test [, colSums(is.na(test))  == 0]


dim(train)
```

```
## [1] 19622    54
```

```
dim(test)
```

```
## [1] 20 54
```

Hence, the number of features in the dataset has been successfully reduced from 160 to 54, which is computationally more feasible and practical.

The train set is partitioned into a train and test set to test our models.The partition is done 80-20

```
set.seed(60606)

indices <- createDataPartition(y=train$classe, p=0.8, list = FALSE)

train.data <- train[indices, ]
test.data  <- train[-indices,]

dim(train.data)
```

```
## [1] 15699    54
```

```
dim(test.data)
```

```
## [1] 3923   54
```

Prediction Model

The problem is classification based. So, a classification model like ‘trees’ would be suitable. Random forests is chosen to be the model as it can take into account the multiple variables and also does random sampling with replacement(bootstrapping).

Before training the model, a control function and pre-processing the data helps in faster training

```
conFun <- trainControl(method='cv', 5)
pre.proc <- c('center', 'scale')

pred.model <- train(classe~.,
                    data=train.data,
                    method = 'rf',
                    proxy=TRUE,
                    trControl=conFun,
                    preProcess=pre.proc,
                    ntree=128)

pred.model
```

```
## Random Forest
##
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (57), scaled (57)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12561, 12558, 12559, 12560, 12558
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9903181  0.9877510
##   29    0.9913368  0.9890404
##   57    0.9852217  0.9813033
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 29.
```

Applying the model on the test data gives the out of saple errors and a better accuracy metric for the trained model.

```
test.preds <- predict(pred.model, test.data)
confusionMatrix(test.preds, test.data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    4    0    0    0
##          B    1  754    3    0    0
##          C    0    1  679    6    0
##          D    0    0    2  637    3
##          E    0    0    0    0  718
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9921, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9934   0.9927   0.9907   0.9958
## Specificity            0.9986   0.9987   0.9978   0.9985   1.0000
## Pos Pred Value         0.9964   0.9947   0.9898   0.9922   1.0000
## Neg Pred Value         0.9996   0.9984   0.9985   0.9982   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1922   0.1731   0.1624   0.1830
## Detection Prevalence   0.2852   0.1932   0.1749   0.1637   0.1830
## Balanced Accuracy      0.9988   0.9961   0.9953   0.9946   0.9979
```

An accuracy of 99.62% is observes.The trained model is able to predict well for new data which wasn't shown during the training phase.

Test Data Set

The model is used to predict the classe for the test set provided.

```
predict(pred.model, 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
```