MLProject

Vaibhav Chugh

8/13/2021

Introduction

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. 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.

Goal

In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal of your project is to predict the manner in which they 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 create a report describing 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.

Fetching Data

```
library(caret)
library(randomForest)
library(e1071)


if (!file.exists("train_data.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile="train_d
ata.csv")
}
if (!file.exists("test_data.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile="test_dat
a.csv")
}

train_init <-  read.csv("train_data.csv")
test_init <-  read.csv("test_data.csv")
dim(train_init)
```

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

PreProcessing

```
#Removing NA Values
train_init <- train_init[,colSums(is.na(train_init)) == 0]
test_init <- test_init[,colSums(is.na(test_init)) == 0]

#Partitioning
trainOb <- createDataPartition(train_init$classe, p = 0.8, list = FALSE)
Training <- train_init[trainOb, ]
Test <- train_init[-trainOb, ]

#Removing features with near zero variance
nzvcol <- nearZeroVar(Training)
Training <- Training[, -nzvcol]

#Removing non-numeric columns
categories <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
    "cvtd_timestamp", "new_window", "num_window")
Training <- Training[, !names(Training) %in% categories]

Training$classe = factor(Training$classe)
Test$classe =  factor(Test$classe)
```

Training

Using random forest algorithm since the feature selection is automatic from all the columns in the dataframe.

```
rf <- randomForest(classe ~ ., data = Training, importance = TRUE, ntrees = 10)
```

Performance of our model

In-sample error (Training set)

```
train_perf <- predict(rf, Training)
print(confusionMatrix(train_perf, Training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Out-sample error (Test set)

```
test_perf <- predict(rf, Test)
print(confusionMatrix(test_perf, Test$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1113    4    0    0    0
##          B    2  752    2    0    0
##          C    0    3  681    3    0
##          D    0    0    1  640    1
##          E    1    0    0    0  720
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9957          
##                  95% CI : (0.9931, 0.9975)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9945          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9973   0.9908   0.9956   0.9953   0.9986
## Specificity            0.9986   0.9987   0.9981   0.9994   0.9997
## Pos Pred Value         0.9964   0.9947   0.9913   0.9969   0.9986
## Neg Pred Value         0.9989   0.9978   0.9991   0.9991   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2837   0.1917   0.1736   0.1631   0.1835
## Detection Prevalence   0.2847   0.1927   0.1751   0.1637   0.1838
## Balanced Accuracy      0.9979   0.9948   0.9969   0.9974   0.9992
```

Accuracy on the test set is 99.7%

Predicting the test cases

```
testC_perf <- predict(rf, test_init)
testC_perf
```

```
##  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
```