Exercise Prediction

alikasapoglu

July 27, 2019

Predicting the Exercise Type

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.

The goal of the 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.

1. Prepare Data

```
library(caret)
library(rpart)
library(randomForest)
```

Download Data

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

data_raw <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
data_test <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

data <- data_raw
```

Clear Data

Let's first remove the columns which we know that they are not effecting 'classe' variable

```
tempcolnames <- grep("name|timestamp|window|X", colnames(data), value=F)
data <- data[,-tempcolnames]
rm("tempcolnames")
```

Then let's remove the columns which have too much NA values. I've defined treshold as 75% for this.

```
treshold <- nrow(data)*0.75
okCols <- names(data[,colSums(is.na(data)) < treshold] )
data <- data[,okCols]
rm(okCols)
dim(data)
```

```
## [1] 19622    53
```

Lastly I'd like to remove the variables near zero variance, but I couldnt find any :)

```
NZV <- nearZeroVar(data, saveMetrics=TRUE)
sum(NZV$nzv)
```

```
## [1] 0
```

Split Data for Train and Validation

```
set.seed(142) # for reproducibility
totrain <- createDataPartition(y=data$classe,p=.70,list=F)
data_train <- data[totrain,]
data_mytest <- data[-totrain,]
rm(totrain)
```

2. Decision Tree

```
modelDT <- rpart(data_train$classe ~ ., data=data_train, method="class")

predDT <- predict(modelDT, data_mytest, type = "class")
confusionMatrix(predDT, data_mytest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1491  189   17   57   18
##          B   52  698   51   87   85
##          C   45  116  812  151  126
##          D   62   69   63  605   67
##          E   24   67   83   64  786
##
## Overall Statistics
##
##                Accuracy : 0.7463
##                  95% CI : (0.735, 0.7574)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6786
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8907   0.6128   0.7914   0.6276   0.7264
## Specificity            0.9333   0.9421   0.9099   0.9470   0.9504
## Pos Pred Value         0.8414   0.7174   0.6496   0.6986   0.7676
## Neg Pred Value         0.9555   0.9102   0.9538   0.9285   0.9391
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2534   0.1186   0.1380   0.1028   0.1336
## Detection Prevalence   0.3011   0.1653   0.2124   0.1472   0.1740
## Balanced Accuracy      0.9120   0.7774   0.8506   0.7873   0.8384
```

```r
confusionMatrix(predDT, data_mytest$classe)$overall[1]
```

```
##  Accuracy 
## 0.7463042
```

3. Random Forest

```r
modelRF <- randomForest(data_train$classe ~ .,   data=data_train, do.trace=F)
print(modelRF) # view results
```

```
## 
## Call:
##  randomForest(formula = data_train$classe ~ ., data = data_train,      do.trace = F) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
## 
##         OOB estimate of  error rate: 0.5%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B   14 2639    5    0    0 0.0071482318
## C    0   16 2378    2    0 0.0075125209
## D    0    0   23 2227    2 0.0111012433
## E    0    0    1    5 2519 0.0023762376
```

```r
predRF <- predict(modelRF, data_mytest, type = "class")
confusionMatrix(predRF, data_mytest$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1133    4    0    0
##          C    0    4 1018    6    1
##          D    0    0    4  958    4
##          E    0    0    0    0 1077
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9958          
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9946          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9922   0.9938   0.9954
## Specificity            0.9995   0.9992   0.9977   0.9984   1.0000
## Pos Pred Value         0.9988   0.9965   0.9893   0.9917   1.0000
## Neg Pred Value         1.0000   0.9987   0.9984   0.9988   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1730   0.1628   0.1830
## Detection Prevalence   0.2848   0.1932   0.1749   0.1641   0.1830
## Balanced Accuracy      0.9998   0.9969   0.9950   0.9961   0.9977
```

```r
confusionMatrix(predRF, data_mytest$classe)$overall[1]
```

```
##  Accuracy
## 0.9957519
```

Comparing the accuracy values of Decision Tree model and Random Forest model, we see that Random Forest works way better. Therefore, for predicting the given test data, we use the model we created by random forest method.

4. Predict the Given Test Data

```r
pred_test <- predict(modelRF, newdata = data_test, type="class")
print(pred_test)
```

```
##  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
```