Practical Machine Learning - Assignment PDF

Viren

09/03/2022

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. 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 (see the section on the Weight Lifting Exercise Dataset).

Step1: Load all the Libraries

```
library(knitr)
library(caret)
```

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

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

```
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
```

```
library(rpart)
library(rpart.plot)
```

Step2: Download and clean the Data

```r
if(!file.exists("./data")){dir.create("./data")}
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(trainUrl,destfile="./data/pml-training.csv")

if(!file.exists("./data")){dir.create("./data")}
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testUrl,destfile="./data/pml-testing.csv")
```

```r
init_TrainrawData <- read.csv((trainUrl)) #, na.strings=c("NA","#DIV/0!",""))
init_TestrawData  <- read.csv((testUrl)) # ,, na.strings=c("NA","#DIV/0!",""))

dim(init_TrainrawData)
```

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

```r
dim(init_TestrawData)
```

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

```r
#Removing Variables which are having nearly zero variance.

non_zero_var <- nearZeroVar(init_TrainrawData)


org_training_data <- init_TrainrawData[,-non_zero_var]
org_testing_data <- init_TestrawData[,-non_zero_var]

dim(org_training_data)
```

```
## [1] 19622   100
```

```r
dim(org_testing_data)
```

```
## [1]  20 100
```

```r
##Removing Variables which are having NA values > 95%.
na_val_col <- sapply(org_training_data, function(x) mean(is.na(x))) > 0.95

org_training_data <- org_training_data[,na_val_col == FALSE]
org_testing_data <- org_testing_data[,na_val_col == FALSE]

dim(org_training_data)
```

```
## [1] 19622    59
```

```r
dim(org_testing_data)
```

```
## [1] 20 59
```

```
colnames(org_training_data)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "num_window"
##  [7] "roll_belt"            "pitch_belt"           "yaw_belt"
## [10] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
## [13] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [16] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [19] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [22] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [25] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [28] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [31] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [34] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [37] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [40] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [43] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [46] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [49] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [52] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [55] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [58] "magnet_forearm_z"     "classe"
```

```
colnames(org_testing_data)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "num_window"
##  [7] "roll_belt"            "pitch_belt"           "yaw_belt"
## [10] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
## [13] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [16] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [19] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [22] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [25] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [28] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [31] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [34] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [37] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [40] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [43] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [46] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [49] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [52] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [55] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [58] "magnet_forearm_z"     "problem_id"
```

```
#remove identification only variables
org_training_data <- org_training_data[,8:59]
org_testing_data <- org_testing_data[,8:59]

dim(org_training_data)
```

```
## [1] 19622    52
```

```
dim(org_testing_data)
```

```
## [1] 20 52
```

```
colnames(org_training_data)
```

```
##  [1] "pitch_belt"           "yaw_belt"             "total_accel_belt"
##  [4] "gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
##  [7] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"
## [10] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [13] "roll_arm"             "pitch_arm"            "yaw_arm"
## [16] "total_accel_arm"      "gyros_arm_x"          "gyros_arm_y"
## [19] "gyros_arm_z"          "accel_arm_x"          "accel_arm_y"
## [22] "accel_arm_z"          "magnet_arm_x"         "magnet_arm_y"
## [25] "magnet_arm_z"         "roll_dumbbell"        "pitch_dumbbell"
## [28] "yaw_dumbbell"         "total_accel_dumbbell" "gyros_dumbbell_x"
## [31] "gyros_dumbbell_y"     "gyros_dumbbell_z"     "accel_dumbbell_x"
## [34] "accel_dumbbell_y"     "accel_dumbbell_z"     "magnet_dumbbell_x"
## [37] "magnet_dumbbell_y"    "magnet_dumbbell_z"    "roll_forearm"
## [40] "pitch_forearm"        "yaw_forearm"          "total_accel_forearm"
## [43] "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
## [46] "accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
## [49] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"
## [52] "classe"
```

```
colnames(org_testing_data)
```

```
##  [1] "pitch_belt"           "yaw_belt"             "total_accel_belt"
##  [4] "gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
##  [7] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"
## [10] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [13] "roll_arm"             "pitch_arm"            "yaw_arm"
## [16] "total_accel_arm"      "gyros_arm_x"          "gyros_arm_y"
## [19] "gyros_arm_z"          "accel_arm_x"          "accel_arm_y"
## [22] "accel_arm_z"          "magnet_arm_x"         "magnet_arm_y"
## [25] "magnet_arm_z"         "roll_dumbbell"        "pitch_dumbbell"
## [28] "yaw_dumbbell"         "total_accel_dumbbell" "gyros_dumbbell_x"
## [31] "gyros_dumbbell_y"     "gyros_dumbbell_z"     "accel_dumbbell_x"
## [34] "accel_dumbbell_y"     "accel_dumbbell_z"     "magnet_dumbbell_x"
## [37] "magnet_dumbbell_y"    "magnet_dumbbell_z"    "roll_forearm"
## [40] "pitch_forearm"        "yaw_forearm"          "total_accel_forearm"
## [43] "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
## [46] "accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
## [49] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"
## [52] "problem_id"
```

```
# create a partition with the training dataset
inTrain  <- createDataPartition(org_training_data$classe, p=0.6,list=FALSE)
TrainSet <- org_training_data[inTrain, ]
```

```
TestSet  <- org_training_data[-inTrain, ]

dim(TrainSet)
```

```
## [1] 11776    52
```

```
dim(TestSet)
```

```
## [1] 7846   52
```

Step3: Decision Tree Model

```
DT_modfit <- train(classe ~ ., data = TrainSet, method="rpart")
dim(DT_modfit)
```

```
## NULL
```

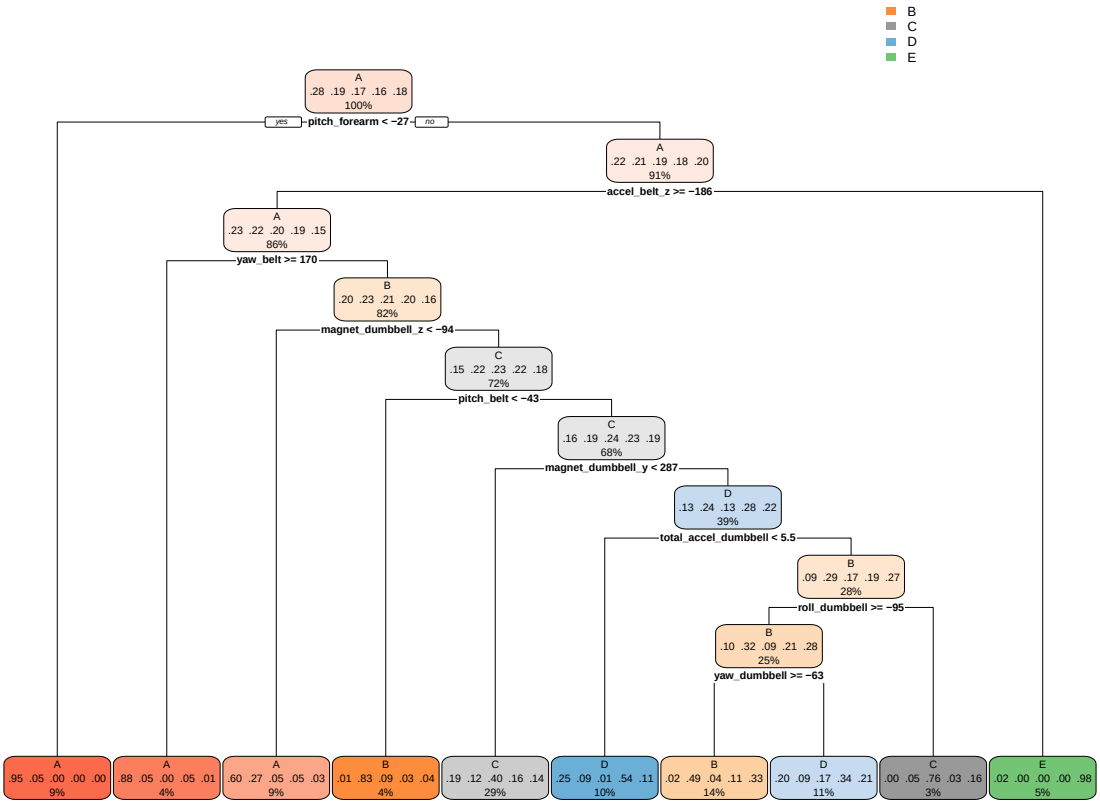
```
##Prediction in terms of Decision Tree Model

DT_prediction <- predict(DT_modfit, TestSet)
confusionMatrix(as.factor(TestSet$classe), DT_prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395   40  429  362    6
##          B  271  842  250  154    1
##          C   35   61 1104  168    0
##          D   69  126  396  695    0
##          E   22  377  360  256  427
##
## Overall Statistics
##
##                Accuracy : 0.5688
##                  95% CI : (0.5578, 0.5798)
##     No Information Rate : 0.3236
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4601
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7785   0.5823   0.4348  0.42508  0.98387
## Specificity            0.8617   0.8944   0.9503  0.90485  0.86306
## Pos Pred Value         0.6250   0.5547   0.8070  0.54044  0.29612
## Neg Pred Value         0.9293   0.9046   0.7785  0.85671  0.99891
## Prevalence             0.2284   0.1843   0.3236  0.20839  0.05531
```

```
## Detection Rate         0.1778   0.1073   0.1407  0.08858  0.05442
## Detection Prevalence   0.2845   0.1935   0.1744  0.16391  0.18379
## Balanced Accuracy      0.8201   0.7383   0.6925  0.66496  0.92347
```

```
rpart.plot(DT_modfit$finalModel, roundint=FALSE)
```



Step4: Random Forest

```
set.seed(111)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf", trControl=controlRF)

predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, as.factor(TestSet$classe))
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2225   11    0    0    0
##          B    6 1504   13    0    0
##          C    0    2 1346   17    3
##          D    0    0    9 1268    7
##          E    1    1    0    1 1432
```

```
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.9886, 0.9929)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9969   0.9908   0.9839   0.9860   0.9931
## Specificity            0.9980   0.9970   0.9966   0.9976   0.9995
## Pos Pred Value         0.9951   0.9875   0.9839   0.9875   0.9979
## Neg Pred Value         0.9988   0.9978   0.9966   0.9973   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2836   0.1917   0.1716   0.1616   0.1825
## Detection Prevalence   0.2850   0.1941   0.1744   0.1637   0.1829
## Balanced Accuracy      0.9975   0.9939   0.9903   0.9918   0.9963
```

Conclusion Conclusion Based on the results, the random forest algorithm has a better accuracy than the decision tree model. We are getting 99.08% in sample accuracy, while the decision tree gives us only 56.11% in sample accuracy. For the final prediction the Random Forest model is therefore used.

Final Prediction

```
predict(modFitRandForest, org_testing_data)
```

```
##  [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
```