Practical Machine Learning

KS

4/18/2021

Overview

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).

Data Source

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Data Loading

The data was downloaded from the Weight Lifting Exercise Dataset database. Downloaded data was uncompressed and read into R environment.

```
train_weight <- read.csv("C:/Users/kavis/Documents/Kavi files/Git-R Files/datasciencecoursera/Practical
dim(train_weight)
```

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

```
test_weight <- read.csv("C:/Users/kavis/Documents/Kavi files/Git-R Files/datasciencecoursera/PracticalMa
dim(test_weight)
```

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

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.4

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.3
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.4
```

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

```
## Warning: package 'rpart.plot' was built under R version 4.0.4
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```r
library(ggplot2)
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.5

## Loaded gbm 2.1.8
```

Data cleaning

```r
# Removing zero variance using nearZerVar function.
non_zer_var <- nearZeroVar(train_weight)
train_clean <- train_weight[, -non_zer_var]
test_clean <- test_weight[, -non_zer_var]

dim(train_clean)
```

```
## [1] 19622   124
```

```r
dim(test_clean)
```

```
## [1]  20 124
```

```r
# Removing all NA values in training and testing data set.
na_val <- sapply(train_clean,function(x) mean(is.na(x))) > 0.95

train_clean <- train_clean[,na_val == FALSE]
test_clean <- test_clean[,na_val == FALSE]

dim(train_clean)
```

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

```r
dim(test_clean)
```

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

```r
# Removing all non numeric values in dataset.
train_clean <- train_clean[,8:59]
test_clean <- test_clean[, 8:59]

dim(train_clean)
```

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

```r
dim(test_clean)
```

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

Data Partitioning

Cross validation will be performed by splitting the training data(60%) and testing(40%) data.

```r
inTrain <- createDataPartition(train_clean$classe, p= 0.6 ,list=FALSE)
training <- train_clean[inTrain,]
testing <- train_clean[-inTrain,]
dim(training)
```

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

```r
dim(testing)
```

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

Random Forest Model

```r
# Build Random Forest Model
set.seed(111)
controlRFM <- trainControl(method="cv", number=3,verboseIter = FALSE)
RFM <- train(classe~., data=training,method="rf",ntree=5,trControl=controlRFM)
RFM$finalModel
```

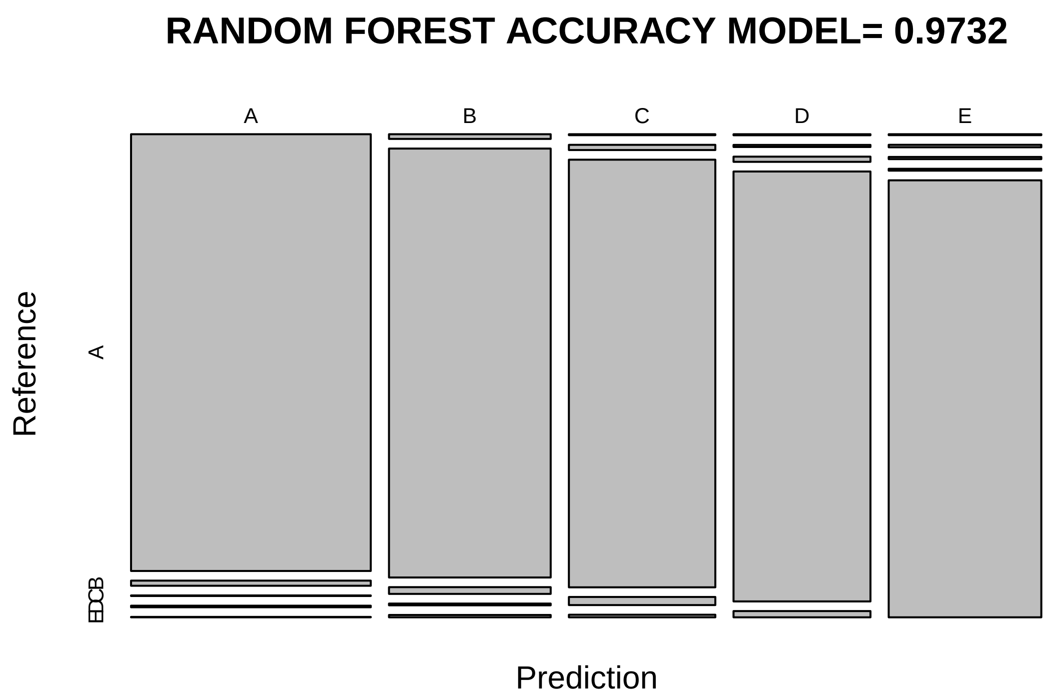
```
##
## Call:
##  randomForest(x = x, y = y, ntree = 5, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 5
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 7.79%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 2939   69   12   20   12  0.03702490
## B   91 1812   59   29   42  0.10870635
## C   22   76 1677   51   27  0.09498111
## D   28   40   68 1566   32  0.09688581
## E   12   61   35   40 1789  0.07640681
```

```r
# Predict the RF model using predict()
predict_RFM <- predict(RFM, testing)
conf_RFM <- confusionMatrix(predict_RFM,as.factor(testing$classe))
conf_RFM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2207   26    3    8    2
##          B   16 1462   23    6    8
##          C    3   16 1320   25    8
##          D    3    6   15 1241   18
##          E    3    8    7    6 1406
##
## Overall Statistics
##
##                Accuracy : 0.9732
##                  95% CI : (0.9694, 0.9767)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9661
##
##  Mcnemar's Test P-Value : 0.1443
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9888   0.9631   0.9649   0.9650   0.9750
```

```
## Specificity            0.9931   0.9916   0.9920   0.9936   0.9963
## Pos Pred Value         0.9826   0.9650   0.9621   0.9673   0.9832
## Neg Pred Value         0.9955   0.9912   0.9926   0.9931   0.9944
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2813   0.1863   0.1682   0.1582   0.1792
## Detection Prevalence   0.2863   0.1931   0.1749   0.1635   0.1823
## Balanced Accuracy      0.9909   0.9774   0.9784   0.9793   0.9856
```

```
plot(conf_RFM$table, col=conf_RFM$byclass,main = paste("RANDOM FOREST ACCURACY MODEL=", round(conf_RFM$
```



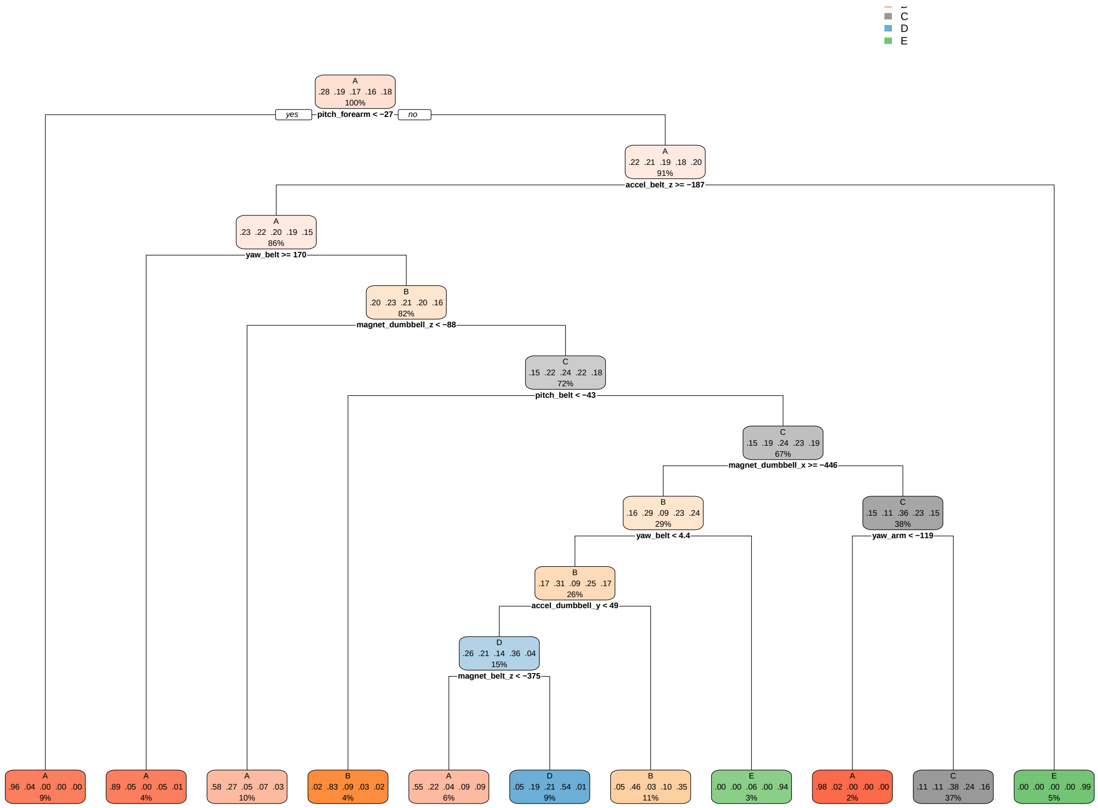
Decision Tree Model

```
# Build Decion Tree  Model
DT <- train(classe~., data=training,method="rpart")
# Predict the DT model using predict()
predict_DT <- predict(DT, testing)
conf_DT <- confusionMatrix(predict_DT,as.factor(testing$classe))
conf_DT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1845  390   62  146   78
##          B   51  701   55   84  292
##          C  284  301 1082  670  450
##          D   49  126  156  386    7
##          E    3    0   13    0  615
##
## Overall Statistics
##
##                Accuracy : 0.59
##                  95% CI : (0.579, 0.6009)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4794
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8266  0.46179   0.7909  0.30016  0.42649
## Specificity            0.8796  0.92383   0.7368  0.94848  0.99750
## Pos Pred Value         0.7319  0.59256   0.3882  0.53315  0.97464
## Neg Pred Value         0.9273  0.87738   0.9435  0.87363  0.88538
## Prevalence             0.2845  0.19347   0.1744  0.16391  0.18379
## Detection Rate         0.2352  0.08934   0.1379  0.04920  0.07838
## Detection Prevalence   0.3213  0.15078   0.3552  0.09228  0.08042
## Balanced Accuracy      0.8531  0.69281   0.7639  0.62432  0.71200
```

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



Applying the selected model to the test data.

The Accuracy of the 2 modeling methods above are : RF : 0.9787 DT : 0.4808 After checking overall statistical data, ConfusionMatrix show, that RandomForestModel performs better than Decision Tree Model. So RF predicts more accuracy ,will be applied to predict the quiz.

```
PredictTest <- predict(RFM, newdata= test_weight)
PredictTest
```

```
##  [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
```