Activity recognition - predictive analytics

Activity recognition - predictive analytics

Dhyanesh babu 6 April 2018

Summary

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

Dataset

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)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Steps performed

1. Loading the data
2. Cleaning the data
3. Model selection
4. Predicting test output

loading the data

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## 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
```

```
library(plyr)


training_URL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.cs
v"
testing_URL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(training_URL))
testing <- read.csv(url(testing_URL))
```

Cleaning the data

```
set.seed(32233)
#Remove few columns

training <- subset(training, select=-c(1:7))
testing <- subset(testing, select=-c(1:7))


threshold_val <- 0.95 * dim(training)[1]

# Remove 95 % of the column values are NA

include_columns <- !apply(training, 2, function(y) sum(is.na(y)) > threshold_val ||
sum(y=="") > threshold_val)
training <- training[, include_columns]

dim(training)
```

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

```
#variance near zero columns

nearZvar <- nearZeroVar(training, saveMetrics = TRUE)
training <- training[ , nearZvar$nzv==FALSE]
dim(training)
```

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

spliting the data

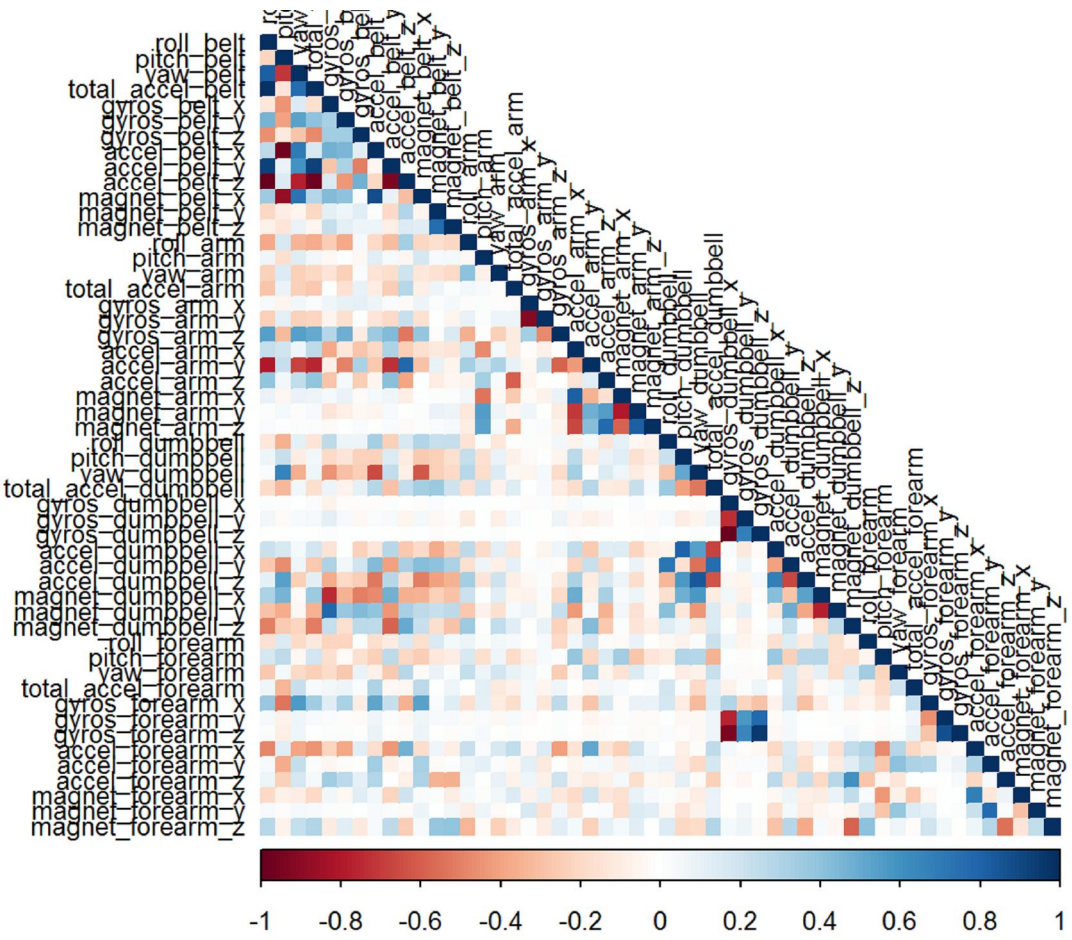
```
#split the dataframe

label <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[label, ]
test <- training[-label, ]


library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrMat <- cor(train[,-53])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,
0))
```



Model selection

Decision Tree

Traning

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

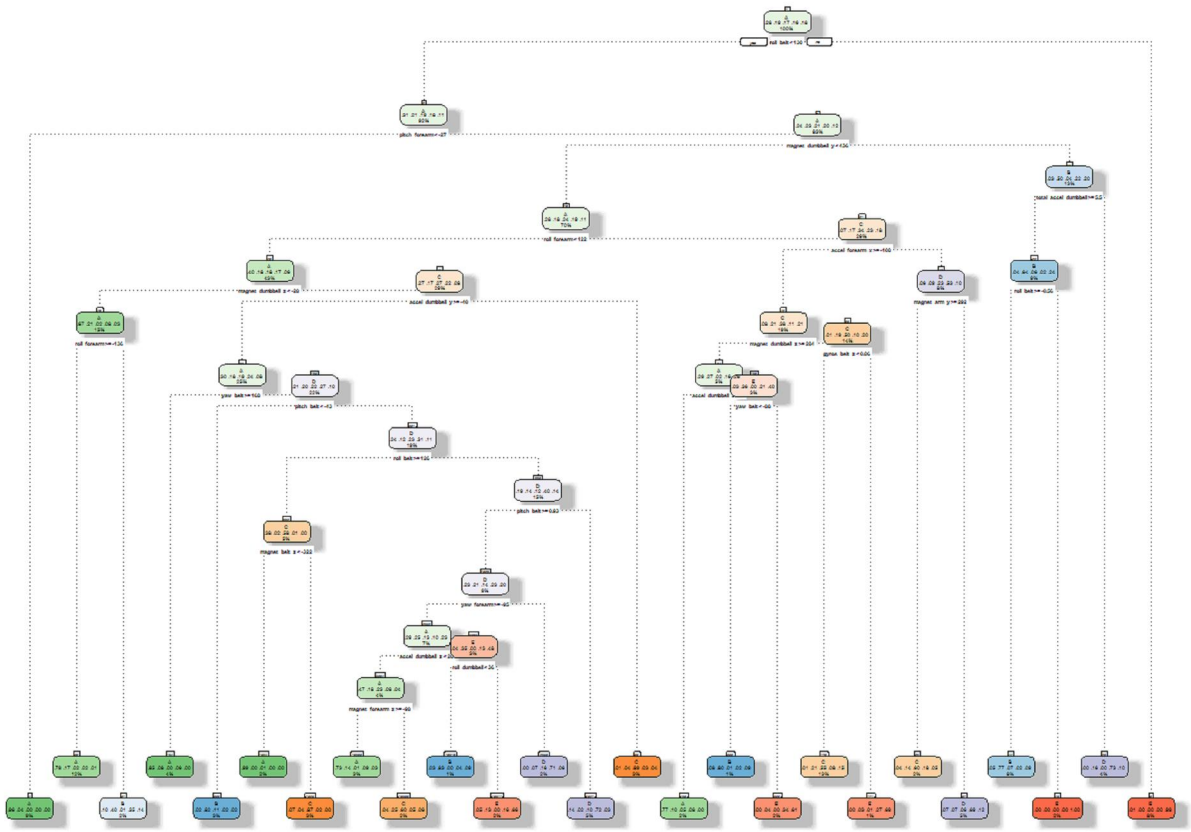
```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
set.seed(13908)
model_decision_tree <- rpart(classe ~ ., data = train, method = "class")
fancyRpartPlot(model_decision_tree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2018-Apr-06 12:59:43 287618

```
predict_decision_tree <- predict(model_decision_tree, test, type = "class")
conf_matric_decision_tree <- confusionMatrix(predict_decision_tree, test$classe)
conf_matric_decision_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1500  183   21   59   10
##          B   67  664   50   65   85
##          C   39  196  875  104  151
##          D   55   74   77  660   89
##          E   13   22    3   76  747
##
## Overall Statistics
##
##                Accuracy : 0.7555
##                  95% CI : (0.7443, 0.7664)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6904
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8961   0.5830   0.8528   0.6846   0.6904
## Specificity            0.9352   0.9437   0.8992   0.9401   0.9763
## Pos Pred Value         0.8460   0.7132   0.6410   0.6911   0.8676
## Neg Pred Value         0.9577   0.9041   0.9666   0.9383   0.9333
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2549   0.1128   0.1487   0.1121   0.1269
## Detection Prevalence   0.3013   0.1582   0.2319   0.1623   0.1463
## Balanced Accuracy      0.9156   0.7634   0.8760   0.8124   0.8333
```

Random forest

```
library(caret)
set.seed(13908)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
model_Random_Forest <- train(classe ~ ., data = train, method = "rf", trControl = c
ontrol)
model_Random_Forest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.73%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    3    2    0    2 0.001792115
## B   18 2634    5    1    0 0.009029345
## C    0   16 2370   10    0 0.010851419
## D    0    1   28 2221    2 0.013765542
## E    0    1    4    7 2513 0.004752475
```

predict

```
predict_random_forest <- predict(model_Random_Forest, test)
conf_matrix_Random_Forest <- confusionMatrix(predict_random_forest, test$classe)
conf_matrix_Random_Forest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   11    0    0    0
##          B    0 1126    5    1    1
##          C    0    2 1017    6    3
##          D    0    0    4  956    5
##          E    0    0    0    1 1073
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9886   0.9912   0.9917   0.9917
## Specificity            0.9974   0.9985   0.9977   0.9982   0.9998
## Pos Pred Value         0.9935   0.9938   0.9893   0.9907   0.9991
## Neg Pred Value         1.0000   0.9973   0.9981   0.9984   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1913   0.1728   0.1624   0.1823
## Detection Prevalence   0.2863   0.1925   0.1747   0.1640   0.1825
## Balanced Accuracy      0.9987   0.9936   0.9945   0.9949   0.9957
```

predicting test output

```
predict_random_forest<- predict(model_Random_Forest, testing)
predict_random_forest
```

```
##  [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
```