Exercise Quality Prediction

Kishibe Rohan

17/07/2020

made with knitr

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, we will use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants to predict the manner of exercise.

Note: The dataset used in this project is a courtesy of "Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements"

Data Loading and Preprocessing

```
#Load required libraries
library(caret)
```

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

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

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

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
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:rattle':
##
##     importance

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

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Getting and Cleaning Data

```
trainDT <- read.csv('pml-training.csv',header=TRUE)
validDT <- read.csv('pml-testing.csv',header=TRUE)

#Remove the variables containing missing values
trainData <- trainDT[,colSums(is.na(trainDT))==0]
validData <- validDT[,colSums(is.na(validDT))==0]

#Remove the first 7 variables as they do not impact the output
trainData <- trainData[,-c(1:7)]
validData <- validData[,-c(1:7)]
```

```
dim(trainData)
```

```
## [1] 19622    86
```

```
dim(validData)
```

```
## [1] 20 53
```

Split into train and test set

```
set.seed(1234)               #for reproducibilty
inTrain <- createDataPartition(trainData$classe,p=0.7,list=FALSE)
trainData <- trainData[inTrain,]
testData  <- trainData[-inTrain,]

#Remove variables that have near zero variance
nearZeroVariance <- nearZeroVar(trainData)
trainData <- trainData[,-nearZeroVariance]
testData  <- testData[,-nearZeroVariance]
```

```
dim(trainData)
```

```
## [1] 13737    53
```

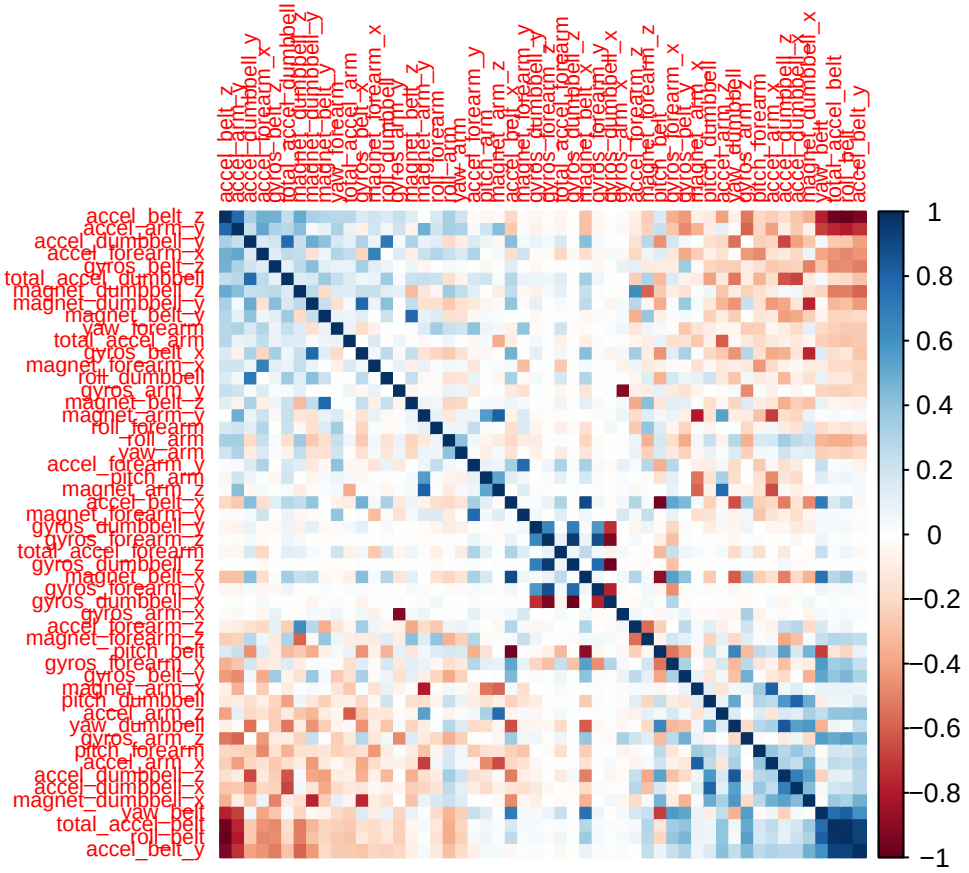
```
dim(testData)
```

```
## [1] 4123   53
```

Correlation between variables

Correlation Matrix Visualization

```
corrPlot <- cor(trainData[,-53])
corrplot(corrPlot,order="FPC",method="color",tl.cex = 0.7)
```



The correlted variables are those that have a dark color intersection.

Obtain the correlated variables

```
#Obtain the variables with correlation with a cut off equal to 0.75

correlated <- findCorrelation(corrPlot,cutoff = 0.75)
names(trainData)[correlated]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "accel_dumbbell_z"  "accel_belt_x"
##  [7] "pitch_belt"        "magnet_dumbbell_x" "accel_dumbbell_y"
## [10] "magnet_dumbbell_y" "accel_dumbbell_x"  "accel_arm_x"
## [13] "accel_arm_z"       "magnet_arm_y"      "magnet_belt_z"
## [16] "accel_forearm_y"   "gyros_forearm_y"   "gyros_dumbbell_x"
## [19] "gyros_dumbbell_z"  "gyros_arm_x"
```

Model Building

First,we will use the Random Forest algorithm and then compare it with a Generalized Boosted Model to decide the better model for the data.

Random Forest

```
controlRF <- trainControl(method="cv",number=5,verboseIter = FALSE)
modelRF <- train(classe ~ . ,data=trainData,method="rf",trControl=controlRF,ntree=5 )
modelRF$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: 27
##
##         OOB estimate of  error rate: 5.66%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3441   46   18   20    7  0.02576444
## B   70 2161   70   33   26  0.08432203
## C   10   73 2021   55   17  0.07123162
## D   24   44   68 1879   23  0.07801766
## E    8   28   27   31 2143  0.04202056
```

Validate the model on the test data to find out its accuracy.

```
predictRF <- predict(modelRF,newdata = testData)
confusionMatrix(predictRF,as.factor(testData$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1163    1    0    0    0
##          B    2  786    0    0    0
##          C    0    0  738    0    0
##          D    0    0    0  672    0
##          E    0    0    0    0  761
##
## Overall Statistics
##
##                Accuracy : 0.9993
##                  95% CI : (0.9979, 0.9998)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9983   0.9987    1.000    1.000   1.0000
## Specificity            0.9997   0.9994    1.000    1.000   1.0000
## Pos Pred Value         0.9991   0.9975    1.000    1.000   1.0000
## Neg Pred Value         0.9993   0.9997    1.000    1.000   1.0000
## Prevalence             0.2826   0.1909    0.179    0.163   0.1846
## Detection Rate         0.2821   0.1906    0.179    0.163   0.1846
## Detection Prevalence   0.2823   0.1911    0.179    0.163   0.1846
## Balanced Accuracy      0.9990   0.9991    1.000    1.000   1.0000
```

The accuracy rate using random forest is **0.9993** and the out-of-sample error is *0.0007*

Generalized Boosted Regression

```
set.seed(1234)
controlGBM <- trainControl(method="repeatedcv",number = 3,repeats = 1)
modelGBM <- train(classe ~ . ,data=trainData,method="gbm",trControl=controlGBM)
modelGBM$finalModel
```

```
print(modelGBM)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 1 times)
## Summary of sample sizes: 9159, 9158, 9157
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7516931  0.6851589
##   1                  100      0.8224515  0.7753222
##   1                  150      0.8525159  0.8133518
##   2                   50      0.8579028  0.8199982
##   2                  100      0.9039099  0.8783724
##   2                  150      0.9295339  0.9108244
##   3                   50      0.8970671  0.8697069
##   3                  100      0.9380512  0.9216162
##   3                  150      0.9585065  0.9475037
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Validate the model on the test data to find out its accuracy.

```
predictGBM <- predict(modelGBM,newdata = testData)
confusionMatrix(predictGBM, as.factor(testData$classe))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1161   19    0    0    1
##          B    3  750   22    0    4
##          C    1   17  709   17    4
##          D    0    0    6  653    9
##          E    0    1    1    2  743
##
## Overall Statistics
##
##                Accuracy : 0.974
##                  95% CI : (0.9687, 0.9787)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9672
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9966   0.9530   0.9607   0.9717   0.9763
## Specificity            0.9932   0.9913   0.9885   0.9957   0.9988
## Pos Pred Value         0.9831   0.9628   0.9479   0.9775   0.9946
## Neg Pred Value         0.9986   0.9889   0.9914   0.9945   0.9947
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2816   0.1819   0.1720   0.1584   0.1802
## Detection Prevalence   0.2864   0.1889   0.1814   0.1620   0.1812
## Balanced Accuracy      0.9949   0.9721   0.9746   0.9837   0.9876
```

Accuracy rate for GBM is *0.974* and hence the out-of-sample error is: *0.026*

Predict for Test Data

```
result <- predict(modelRF,newdata = validData)
result
```

```
##  [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
```