Coursera Project - Machine Learning

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

Preperations

```
memory.limit(9999999999)
```

```
## [1] 1e+10
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.6.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3
```

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

```
## Warning: package 'tibble' was built under R version 3.6.1
```

```
## 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.
## Geben Sie 'rattle()' ein, um Ihre Daten mischen.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## 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
```

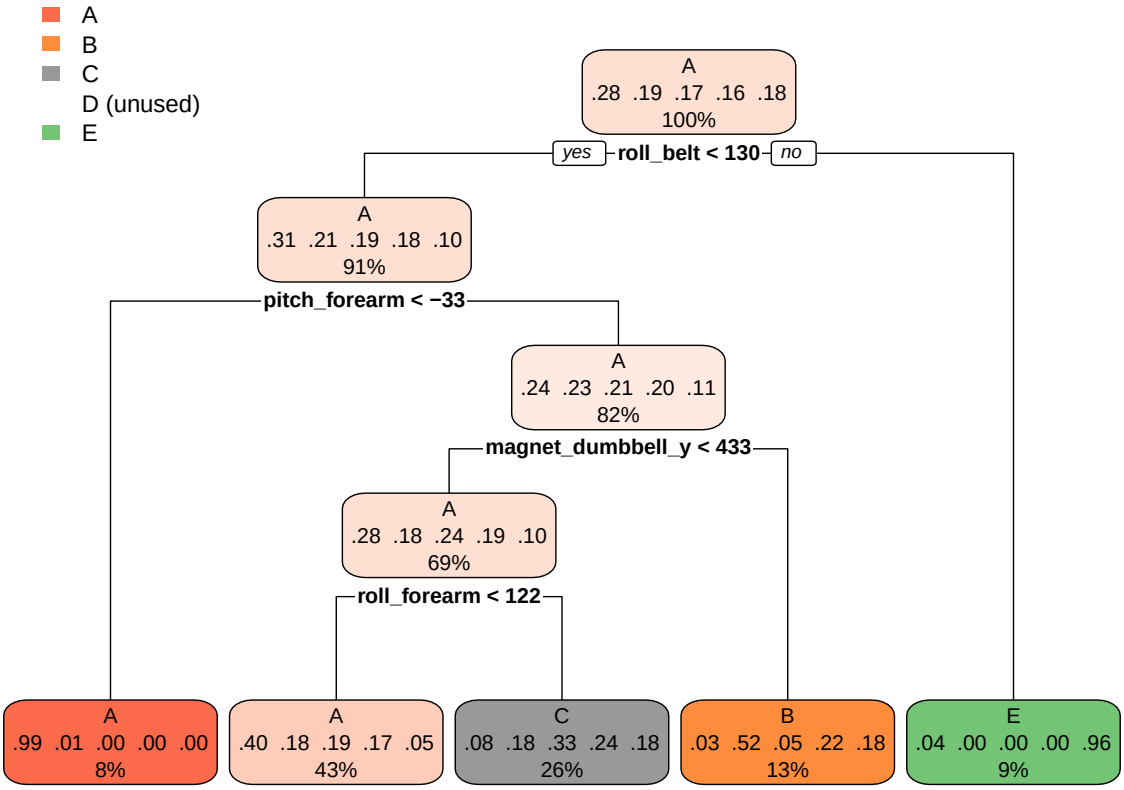
```
##Data Loading

setwd = "D:\\Workspace\\Data_Scientit_Coursa\\R_stuff\\machine_learning"

# set the URL for the download
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

#Cleaning the input data

#Removing every variable without any value

training<- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]

#Removing col1-7, as these cols do not have any influence to the models
training_c <- training[,-c(1:7)]
testing_c <- testing[,-c(1:7)]
```

```
##Cleaning further (near zero values)
NZV <- nearZeroVar(training_c)
training_c <- training_c[, -NZV]

#Splitting the train set into a train set and a test set.

set.seed(50)
training_s <- createDataPartition(training_c$classe, p = 0.5, list = FALSE)
train_train <- training_c[training_s, ]
train_test <- training_c[-training_s, ]

#Rename the testing set into validation set
validation <- testing_c
```

Model building and testing

1) Decisions Tree

```
set.seed(50)
mod_decisionTree <- train(classe ~., method='rpart', data=train_train)
rpart.plot(mod_decisionTree$finalModel)
```



```r
# prediction on Test dataset
predict_decisionTree <- predict(mod_decisionTree, train_test)
confusionMatrix(train_test$classe, predict_decisionTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2480   67  206    0   37
##          B  766  661  471    0    0
##          C  777   57  877    0    0
##          D  690  314  604    0    0
##          E  263  248  480    0  812
##
## Overall Statistics
##
##                Accuracy : 0.4924
##                  95% CI : (0.4824, 0.5023)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 0.9985
##
##                   Kappa : 0.3376
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4984  0.49072   0.3324       NA  0.95642
## Specificity            0.9359  0.85383   0.8837   0.8361  0.88941
## Pos Pred Value         0.8889  0.34826   0.5126       NA  0.45036
## Neg Pred Value         0.6444  0.91330   0.7826       NA  0.99538
## Prevalence             0.5072  0.13731   0.2689   0.0000  0.08654
## Detection Rate         0.2528  0.06738   0.0894   0.0000  0.08277
## Detection Prevalence   0.2844  0.19348   0.1744   0.1639  0.18379
## Balanced Accuracy      0.7171  0.67228   0.6081       NA  0.92291
```

With a model accuracy of 0.7171 and a out of sample error of about 0.28, the model is far from perfect.

2) Random Forrest

```r
set.seed(50)
control_rf <- trainControl(method="cv", number=3, verboseIter=FALSE)
mod_rf <- train(classe ~ ., data=train_train, method="rf",trControl=control_rf)
mod_rf$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: 1.11%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 2785    5    0    0    0 0.001792115
## B   20 1866   11    2    0 0.017377567
## C    0   20 1681   10    0 0.017533606
## D    0    3   25 1578    2 0.018656716
## E    0    0    1   10 1793 0.006097561
```

```r
# prediction on Test dataset
predict_rf <- predict(mod_rf, newdata=train_test)
confMat_rf <- confusionMatrix(predict_rf, train_test$classe)
confMat_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2782   22    0    0    0
##          B    1 1858   12    1    4
##          C    5   18 1691   14    7
##          D    1    0    8 1593    4
##          E    1    0    0    0 1788
##
## Overall Statistics
##
##                Accuracy : 0.99
##                  95% CI : (0.9878, 0.9919)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9874
##
##  Mcnemar's Test P-Value : 2.165e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9789   0.9883   0.9907   0.9917
## Specificity            0.9969   0.9977   0.9946   0.9984   0.9999
## Pos Pred Value         0.9922   0.9904   0.9746   0.9919   0.9994
## Neg Pred Value         0.9989   0.9950   0.9975   0.9982   0.9981
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2836   0.1894   0.1724   0.1624   0.1823
## Detection Prevalence   0.2858   0.1912   0.1769   0.1637   0.1824
## Balanced Accuracy      0.9970   0.9883   0.9914   0.9945   0.9958
```

The accuracy rate is near to 1 and therfore very high.

Using the Random Forrest model for the validation set

```
predict(mod_rf, validation)
```

```
##  [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
```

Conclusion

The Random Forrest algorithem is the best choise and will be used for the Course Project Prediction Quiz.