Practical Machine Learning_Project Assignment

Mokolade Fadeyibi

10/26/2020

Overview

This report is the final course project for the Practical Machine Learning Module. The machine learning code in this report is further applied to 20 test cases in the test dataset for the purpose of prediction.

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 Loading and Processing

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

First we set the working directory:

```r
setwd("~/Documents/R_files/Course 8")
```

The R libraries required for the analysis were then loaded into the R environment:

```r
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(corrplot)
set.seed(121212)
```

Source data was downloaded and loaded:

```r
trainingdata <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testdata <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
```

Next, we partition the training set:

```r
inTrain  <- createDataPartition(trainingdata$classe, p=0.7, list=FALSE)
TrainSet <- trainingdata[inTrain, ]
```

```
TestSet  <- trainingdata[-inTrain, ]
dim(TrainSet)
```

```
## [1] 13737   160
```

```
dim(TestSet)
```

```
## [1] 5885  160
```

There are quite a number of missing values in the data. For example, examining the 12th column for NAs:

```
sum(is.na(TrainSet[,12]))
```

```
## [1] 13448
```

These will be removed and cleaned as below:

```
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]

NAvalues    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, NAvalues==FALSE]
TestSet  <- TestSet[, NAvalues==FALSE]

TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

```
## [1] 13737    54
```

After cleaning, we're left with 54 variables in the Train set.

```
dim(TestSet)
```

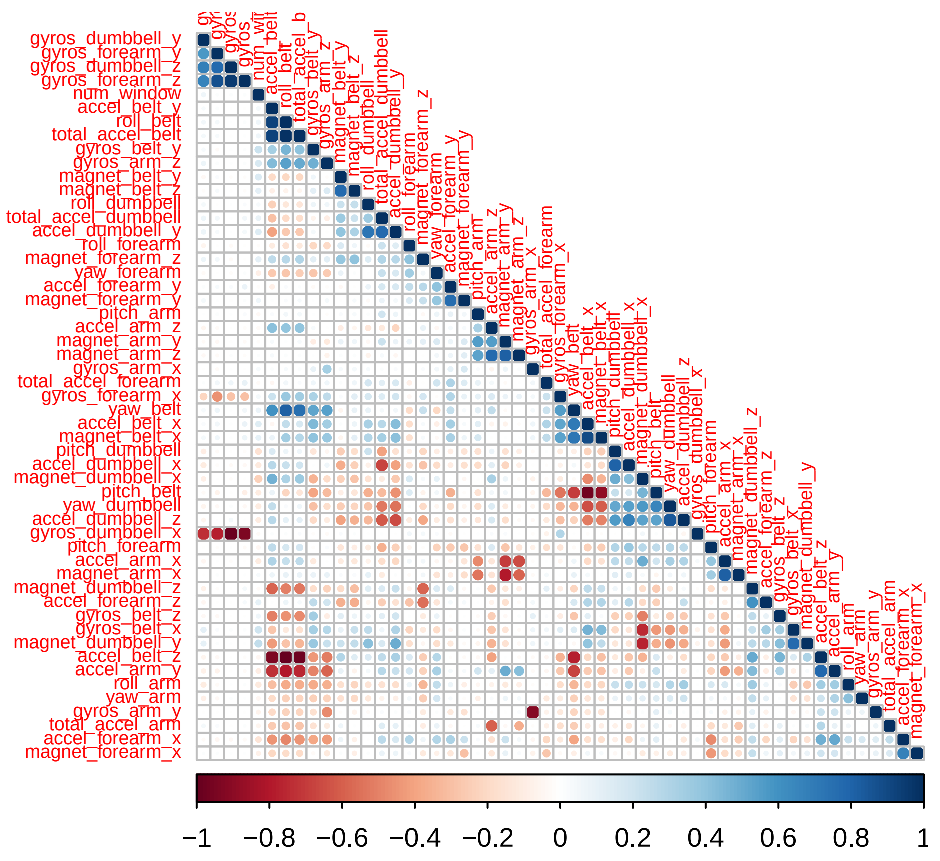
```
## [1] 5885   54
```

Similarly, the Test set has 54 variables after cleaning.

Corelation Analysis

We'll perform a corelation analysis on the data:

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "hclust" , type = "lower",tl.cex = 0.6)
```



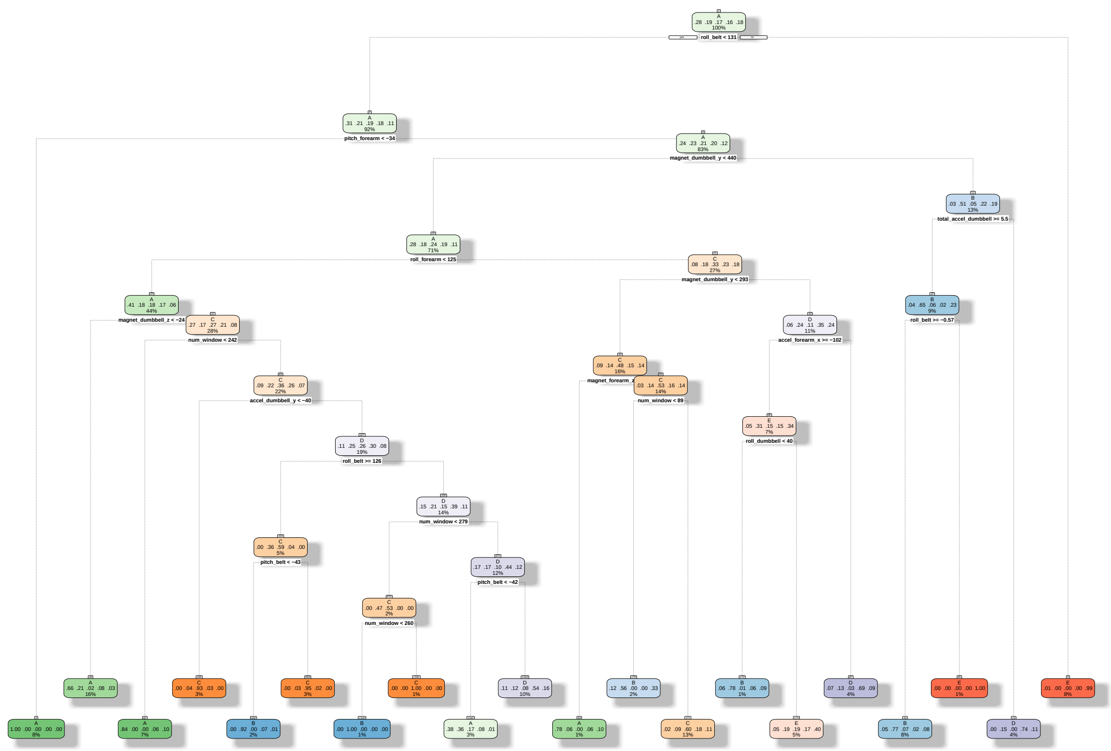
The image shows the level of correlation between the variables. Variables with high positive correlation are indicated by a dark blue colour while those with high negative correlation are denoted by a dark red colour.

Prediction Model

Two prediction models will be utilized. These are i) Decision Tree and ii) Random Forest.

a) Decision Tree

```
set.seed(121212)
DecTreeFit <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(DecTreeFit)
```



Rattle 2020−Oct−26 20:10:21 Mokolade

```
DecTreePrediction <- predict(DecTreeFit, newdata=TestSet, type="class")
DecTreeConfMat <- confusionMatrix(DecTreePrediction, TestSet$classe)
DecTreeConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1512  258   53  119   77
##          B   36  608   26   22   94
##          C   19   55  818  136  105
##          D   86  146   64  626  113
##          E   21   72   65   61  693
##
## Overall Statistics
##
##                Accuracy : 0.7234
##                  95% CI : (0.7117, 0.7348)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6479
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9032   0.5338   0.7973   0.6494   0.6405
## Specificity            0.8796   0.9625   0.9352   0.9169   0.9544
## Pos Pred Value         0.7489   0.7735   0.7220   0.6048   0.7599
## Neg Pred Value         0.9581   0.8959   0.9562   0.9303   0.9218
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2569   0.1033   0.1390   0.1064   0.1178
## Detection Prevalence   0.3431   0.1336   0.1925   0.1759   0.1550
## Balanced Accuracy      0.8914   0.7481   0.8662   0.7831   0.7974
```

This model has an accuracy of 72.3%.

b) Random Forest

```
set.seed(121212)
ctrlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
RandForestFit <- train(classe ~ ., data=TrainSet, method="rf", trControl=ctrlRF)
RandForestFit$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: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2650    2    0    0 0.0030097818
## C    0    3 2393    0    0 0.0012520868
## D    0    0    7 2244    1 0.0035523979
## E    0    1    0    6 2518 0.0027722772
```

```
RandForestPrediction <- predict(RandForestFit, newdata=TestSet)
RandForestConfMat <- confusionMatrix(RandForestPrediction, TestSet$classe)
RandForestConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1132    2    0    0
##          C    0    1 1024    2    0
##          D    0    0    0  962    6
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9954, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9981   0.9979   0.9945
## Specificity            0.9986   0.9996   0.9994   0.9988   1.0000
## Pos Pred Value         0.9964   0.9982   0.9971   0.9938   1.0000
## Neg Pred Value         1.0000   0.9985   0.9996   0.9996   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1740   0.1635   0.1828
## Detection Prevalence   0.2855   0.1927   0.1745   0.1645   0.1828
## Balanced Accuracy      0.9993   0.9967   0.9987   0.9984   0.9972
```

This model has an accuracy rate of 99.7%.

Therefore, the Random Forest model will be applied to our Test Data for prediction.

Project Prediction Quiz

```
predictTESTdata <- predict(RandForestFit, newdata=testdata)
predictTESTdata
```

```
##  [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
```