Practical Machine Learning - Course Project

Stefan

24 11 2020

Background

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).

Data

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://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.

Project Goal

The goal of this project is to predict the manner in whicht a someone exercised using some variables from training and testing data (classe variable).

Installing packages

```
library(caret)
library(knitr)
library(randomForest)
library(corrplot)
library(rpart)
library(rattle)
library(rpart.plot)
```

```r
library(e1071)
library(ggplot2)
library(cowplot)
library(randomForest)
```

Loading the training data

```r
Train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
Test_url  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

train_data <- read.csv(url(Train_url))
test_data <- read.csv(url(Test_url))
```

Splitting the training data for further analysis

```r
SubGroups=createDataPartition(train_data$classe, p=0.7, list=FALSE)

Training <- train_data[SubGroups, ]
Testing <- train_data[-SubGroups, ]
```

Removing Variables with near zero Variance

Since some variables have a near zero variance, they are excluded for further analysis

```r
NZV <- nearZeroVar(Training)

Trainset <- Training[ ,-NZV]
Testset <- Testing[ ,-NZV]
```

Removing variables mostly NA

Since some variables have many NAs, these variables are also excluded.

```r
label <- apply(Trainset, 2, function(x) mean(is.na(x))) > 0.95
Train <- Trainset[, -which(label, label == FALSE)]
Test <- Testset[, -which(label, label == FALSE)]
```

Removing variables

Since some variables like ID or timestamp are not relevant for the prediction, they should be removed

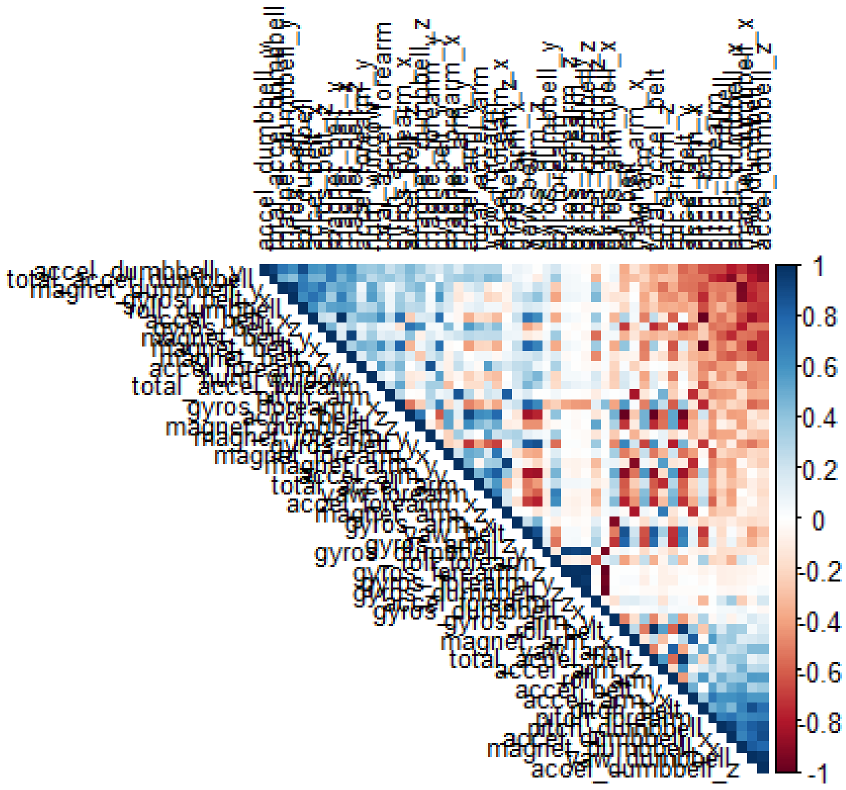
```r
Train <- Train[ , -(1:5)]
Test <- Test[ , -(1:5)]
```

Create correlation matrix and exclude high correlated variables

Before the analysis we check the individual variables for multicollinearity. Variables with high multicolliniarity are excluded in order not to falsify the results

```r
cor.matrix_train <- cor(Train[sapply(Train, is.numeric)])
```

```
cor_mat_train <- cor(cor.matrix_train[, -53])
corrplot(cor_mat_train, order = "FPC", method = "color", type = "upper",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



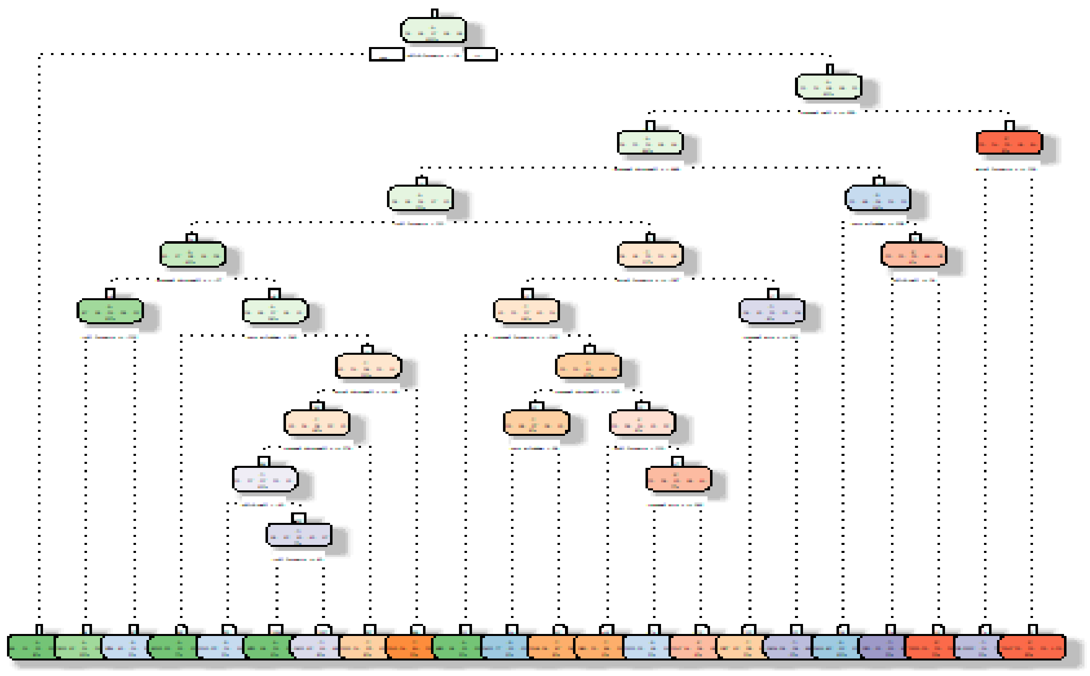
```
c <- findCorrelation(cor.matrix_train, cutoff = .90)
Trainset_prefinal <- Train[,-c]
```

Decission Tree

Beacouse we try to predict classes in form of groups and not numeric values, the first analysis is performed using a decision tree.With the help of the Confusion Matrix, the prediction accuracy should be better illustrated

```
set.seed(123)
DT_Model <- rpart(classe ~., data = Trainset_prefinal, method = "class")
fancyRpartPlot(DT_Model)

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


Rattle 2020-Nov-24 19:49:55 Stef

```
predictDT <- predict(DT_Model,Testset, type = "class")
ConMatDT <- confusionMatrix(predictDT, Testset$classe)
ConMatDT

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1479  162   10   37   66
##          B  107  648   80  151  145
##          C   36  207  840  140  145
##          D   49   85   76  595  117
##          E    3   37   20   41  609
##
## Overall Statistics
##
##                Accuracy : 0.7088
##                  95% CI : (0.697, 0.7203)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6312
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8835   0.5689   0.8187   0.6172   0.5628
## Specificity            0.9347   0.8982   0.8913   0.9336   0.9790
## Pos Pred Value         0.8432   0.5729   0.6140   0.6453   0.8577
## Neg Pred Value         0.9528   0.8967   0.9588   0.9256   0.9086
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2513   0.1101   0.1427   0.1011   0.1035
## Detection Prevalence   0.2980   0.1922   0.2325   0.1567   0.1206
## Balanced Accuracy      0.9091   0.7336   0.8550   0.7754   0.7709
```

Regarding the confusion Matrix, we get a prediction of 70,88 percent for the decision tree, which is not a good prediction. As

Random Forrest

Random Forrest is chosen as the second Model, because although it is usually more difficult to interpret, it provides better predictions. The Confusion Matrix should also clarify the accuracy here

```
set.seed(123)
RF <- randomForest(classe ~. , data= Trainset_prefinal, method="parRF")
predict_RF <- predict(RF, Testset, type = "class")

conMatRF <- confusionMatrix(predict_RF, Testset$classe)
conMatRF

## Confusion Matrix and Statistics
##
##             Reference
## Prediction     A    B    C    D    E
##            A 1674    2    0    0    0
##            B    0 1137    2    0    0
##            C    0    0 1023    8    0
##            D    0    0    1  955    1
##            E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9958, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9971   0.9907   0.9991
## Specificity            0.9995   0.9996   0.9984   0.9996   0.9998
## Pos Pred Value         0.9988   0.9982   0.9922   0.9979   0.9991
## Neg Pred Value         1.0000   0.9996   0.9994   0.9982   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1932   0.1738   0.1623   0.1837
## Detection Prevalence   0.2848   0.1935   0.1752   0.1626   0.1839
## Balanced Accuracy      0.9998   0.9989   0.9977   0.9951   0.9994
```

Looking at the confustion matrix, we can see that there is no misscalssification in the Random Forrest Modell with an Accurancy of 99,75 percent, which is exceptionally high.

Final Prediction

Since, as expected, the random forest model provides a better prediction than the decision tree, the final test set is predicted using the random forest model.

```
RF_final <- predict(RF,test_data)
RF_final

##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```