Practical Machine Learning Assignment

Author: Biaka Imeah

Date: 17/04/2021

Overview

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 Processing

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(caret)
```

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

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

```
##
## Attaching package: 'ggplot2'
```

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

```
set.seed(12345)

## set the URL for the download
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"


#download the datasets
training <- read.csv(url(url_train))
testing  <- read.csv(url(url_test))

# create a partition with the training dataset
inTrain  <- createDataPartition(trainingSclasse, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
#Head(TrainSet)
#Head(TestSet)
dim(TrainSet)
```

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

```
dim(TestSet)
```

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

Data Cleaning and Pre-processing

The training and testing datasets have 160 variables with variables having a lot of NAs. The NA would be removed with the pre-processing approach below. The Near Zero variance (NZV) variables and ID variables are removed as well.

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

```
## [1] 13737   104
```

```
dim(TestSet)
```

```
## [1] 5885  104
```

```
## Removing  variables that are mostly NAs
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]

### Remove ID variables
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

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

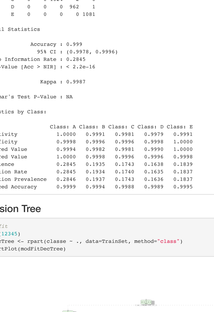
```
dim(TestSet)
```

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

Correlation Analysis

Determine correlation among variables before proceeding to developing the prediction model.

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "circle", type = "upper",
         tl.cex = 0.5)
```



The highly correlated variables are shown in dark colors in the graph above. The PCA approach wouldn't be utilized because the correlation are few. Hence, I will proceed to model development.

Prediction Modelling

Two methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset). The methods are: Random Forests and Decision Tree, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

1. Random Forest Model

```
# model fit
set.seed(12345)
controlRF <- trainControl(method='cv', number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method='rf',
                          trControl=controlRF)
modFitRandForest$finalModel
```

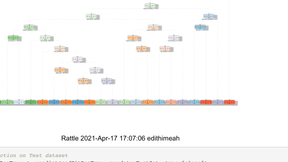
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    2    0    0    0 0.0005120328
## B    4 2647    4    1    0 0.0041384500
## C    0    5 2391    0    0 0.0020868114
## D    0    0    9 2243    0 0.0039964476
## E    0    0    0    5 2520 0.0019801980
```

```
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(factor(predictRandForest), factor(TestSet$classe))
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9978, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9979   0.9991
## Specificity            0.9998   0.9996   0.9996   0.9998   1.0000
## Pos Pred Value         0.9994   0.9982   0.9981   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9996   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1635   0.1837
## Detection Prevalence   0.2846   0.1937   0.1743   0.1636   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9989   0.9995
```

2. Decision Tree

```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method='class')
fancyRpartPlot(modFitDecTree)
```



Rattle 2021-Apr-17 17:07:06 edithimeah

```
# prediction on Test dataset
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type='class')
confMatDecTree <- confusionMatrix(factor(predictDecTree), factor(TestSet$classe))
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 1502  251   59   66   74
##          B   58  660   37   64  114
##          C    4   66  815  129   72
##          D   90  148   54  648  126
##          E   20   64   61   57  696
##
## Overall Statistics
##
##                Accuracy : 0.7342
##                  95% CI : (0.7228, 0.7455)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6625
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8973   0.5795   0.7942   0.6722   0.6433
## Specificity            0.9050   0.9425   0.9442   0.9151   0.9579
## Pos Pred Value         0.7897   0.7074   0.7505   0.6079   0.7751
## Neg Pred Value         0.9568   0.9033   0.9560   0.9344   0.9226
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2552   0.1121   0.1385   0.1101   0.1183
## Detection Prevalence   0.3232   0.1585   0.1845   0.1811   0.1526
## Balanced Accuracy      0.9011   0.7610   0.8693   0.7936   0.8006
```

Applying choosen model to test data

The accuracy of the random forest model and decision tree model are:

Random forest: 0.999

Decision Tree: 0.7342

The accuracy of the random forest model suggest that the model performs better than the decision tree model. Therefore, the decision tree model is apply to the test data.

```
predictTEST <- predict(modFitRandForest, newdata=testing)
predictTEST
```

```
##  [1] B A B A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```