Practical Machine Learning- Final Project

Dimuthu Attanayake

10/24/2021

Introduction

This is the final project report for Practical Machine Learning course, a part of the Data Science Specialization offered by John Hopkins university on Coursera.

The objective of this project is to predict how well individuals performed personal activity, using data from Jawbone Up, Nike FuelBand, and Fitbit devices. The data for the project is from here and consists accelerometer data on the belt, forearm, arm, and dumbell, including both correct and incorrect performance of barbell lifts. The "classe" variable in the data set provides how well the activities are performed.

In this project, Decision Tree, Random Forest and Gradient Boosted tree models are trained and validated using the data. The best performing model was then used to predict the 20 cases with the test data provided.

The training data.

The test data

Preparation of data

Before training and testing the model, the data has to be prepared by exploring, cleaning and pre-processing.

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```r
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```r
library(lattice)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

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

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.0.5
## Loading required package: tibble
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

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

```
## Warning: package 'bitops' was built under R version 4.0.5
```

```
## 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.
```

Setting the working directory and loading the data

```r
setwd("C:/Users/dimut/desktop/Data Science/Practical Machine Learning")
testing <- read.csv("pml-testing.csv")
training <- read.csv("pml-training.csv")
```

Exploring the data

```r
dim(training)
```

```
## [1] 19622   160
```

```r
dim(testing)
```

```
## [1]  20 160
```

```r
typeof(training)
```

```
## [1] "list"
```

The training set has 160 variables and 19622 observations, while the test set has 160 variables and 20 observations.

Cleaning the data

During this step, all the unnecessary variables, including the missing values, metadata(which will not affect the outcome of classe), and non-zero variables will be removed.

```r
#Removing the missing values

trainData<- training[, colSums(is.na(training)) == 0]
testData <- testing[, colSums(is.na(testing)) == 0]
dim(trainData)
```

```
## [1] 19622    93
```

```r
dim(testData)
```

```
## [1] 20 60
```

```r
#Removing the metadata

trainData <- trainData[,-c(1:7)]
testData <- testData[,-c(1:7)]
dim(trainData)
```

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

```r
dim(testData)
```

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

```r
#Removing zero covariates

nvz <- nearZeroVar(trainData)
trainData <- trainData[,-nvz]
dim(trainData)
```

```
## [1] 19622    53
```

```r
typeof(trainData)
```

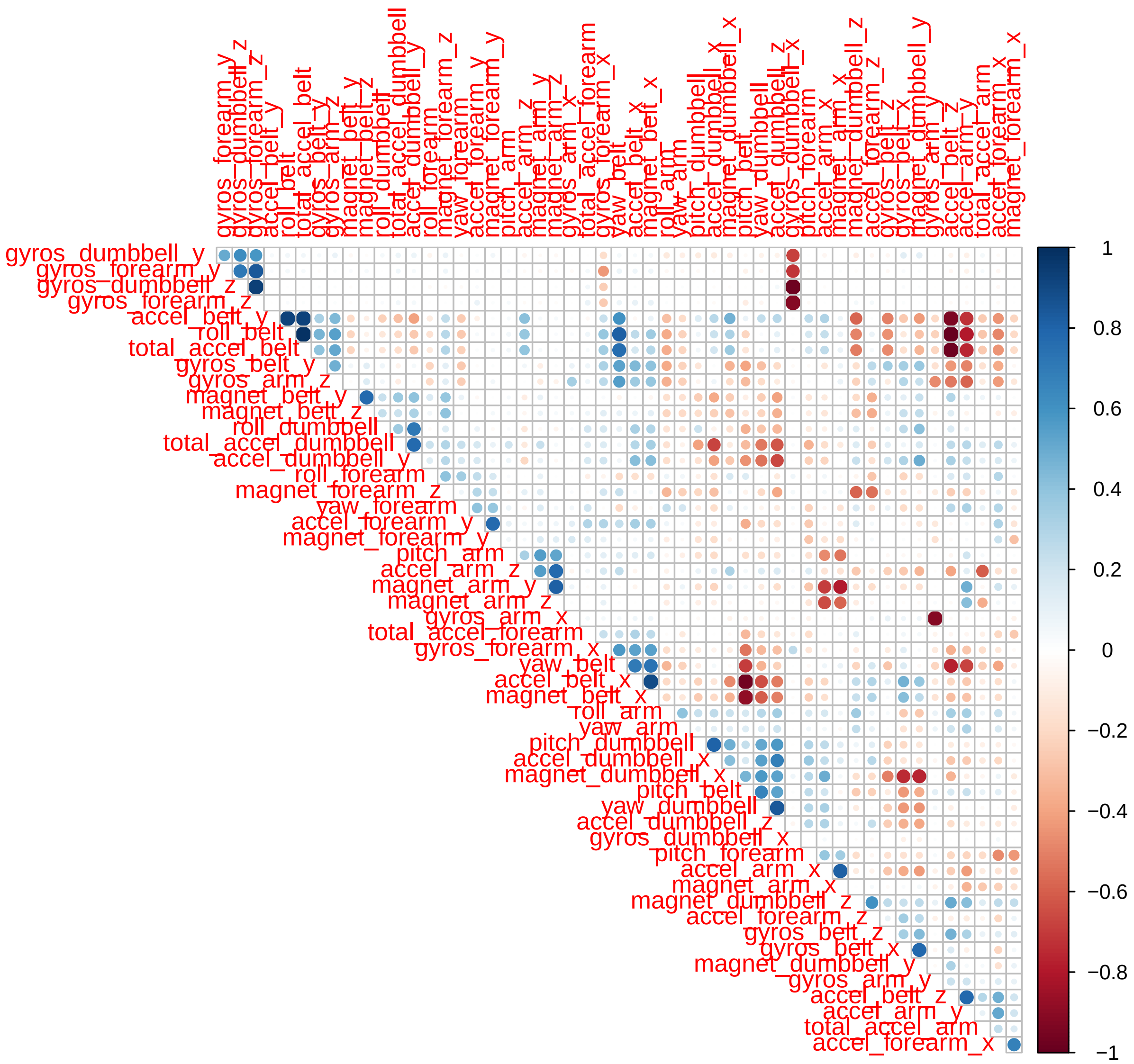
```
## [1] "list"
```

Now the variables of the training set has decreased to 53.

Creating a correlation plot of the variables

To understand how the variables correlate to each other, a correlation plot is used.

```r
D <- trainData
D <- as.data.frame(unclass(D))#converting list into factor
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="circle",diag = FALSE, order = "hclust",type = "upper")
```



Training the model

Before training the models,“trainData” is split into training and validation data sets. The initial training and testing of the models will be conducted using these two data sets.At this stage, five fold cross validation is utilised to better understand how the model will predict with real world data. Lastly, the “testData” will be kept aside for final testing of the best performing model.

```
#Splitting train data into train and validation sets

set.seed(100)
intrain <- createDataPartition(trainData$classe, p = 0.8, list = FALSE)
trainData <- trainData[intrain, ]
validationData <- trainData[-intrain, ]
dim(trainData)
```

```
## [1] 15699    53
```

```
dim(validationData)
```

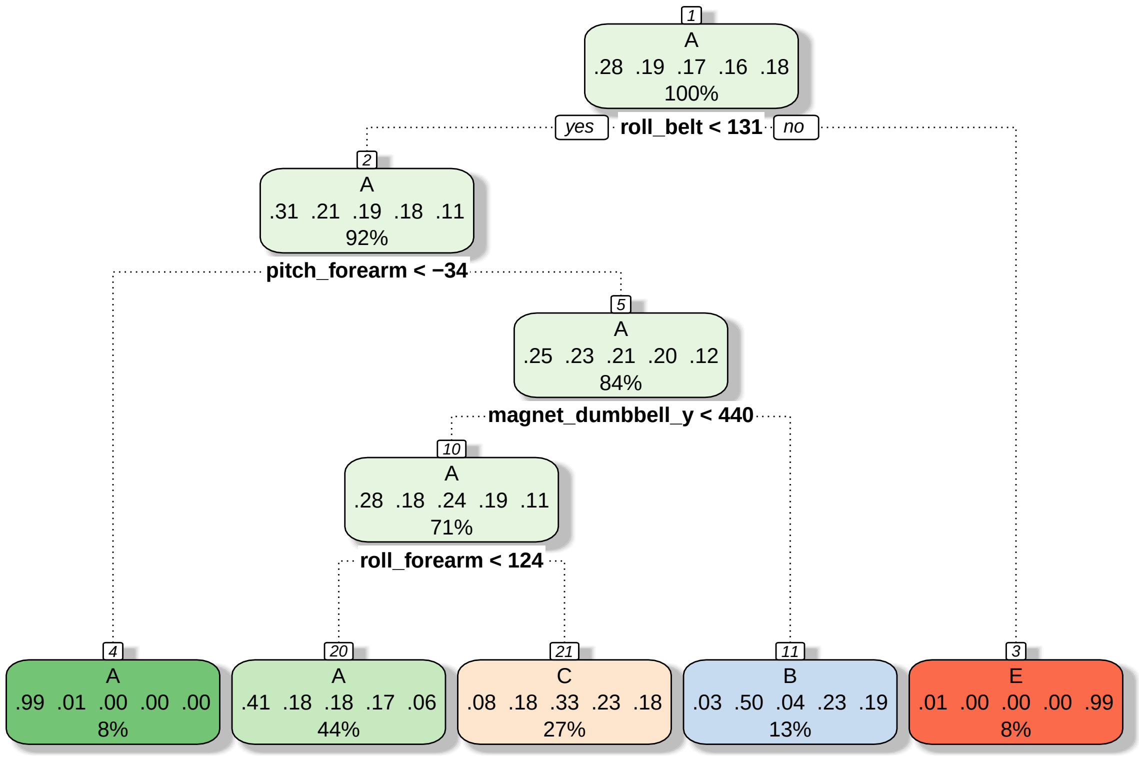
```
## [1] 3145   53
```

```
#Cross validation

cv <- trainControl(method="cv", number=5, verboseIter=F)
```

Decision Tree, Random Forest and Gradient Boosted Tree models are trained and validated to find out which performs the best with the given data set.

1. Decision Tree

```
#Decision tree

set.seed(125)
modDT <- train(classe ~ ., method = "rpart", data = trainData, trControl = cv)
fancyRpartPlot(modDT$finalModel) #tree diagram
```



Rattle 2021−Oct−24 21:09:58 dimut

```
#Prediction

DT <- predict(modDT, validationData)
DT_results <- confusionMatrix(DT, factor(validationData$classe))
DT_results
```

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

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 825 257 261 236  88
##          B  16 204  13  98  84
##          C  57 140 267 185 150
##          D   0   0   0   0   0
##          E   2   0   0   0 262
##
## Overall Statistics
##
##                Accuracy : 0.4954
##                  95% CI : (0.4778, 0.513)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3394
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9167  0.33943   0.4935    0.000  0.44863
## Specificity            0.6249  0.91706   0.7957    1.000  0.99922
## Pos Pred Value         0.4949  0.49157   0.3342      NaN  0.99242
## Neg Pred Value         0.9493  0.85458   0.8832    0.835  0.88823
## Prevalence             0.2862  0.19110   0.1720    0.165  0.18569
## Detection Rate         0.2623  0.06486   0.0849    0.000  0.08331
## Detection Prevalence   0.5300  0.13196   0.2541    0.000  0.08394
## Balanced Accuracy      0.7708  0.62825   0.6446    0.500  0.72392
```

Since the accuracy of the Desion Tree model is very low (less than 0.5) this is not a suitable model for prediction with this particular dataset.

2. Random Forest

```r
#Random forest

set.seed(125)
modRF <- train(classe~., data=trainData, method="rf", trControl = cv)
RF <- predict(modRF, validationData)
RF_Results <- confusionMatrix(RF, factor(validationData$classe))
RF_Results
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 900   0   0   0   0
##          B   0 601   0   0   0
##          C   0   0 541   0   0
##          D   0   0   0 519   0
##          E   0   0   0   0 584
##
```

```
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000    1.000    1.000   1.0000
## Specificity            1.0000   1.0000    1.000    1.000   1.0000
## Pos Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Neg Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Prevalence             0.2862   0.1911    0.172    0.165   0.1857
## Detection Rate         0.2862   0.1911    0.172    0.165   0.1857
## Detection Prevalence   0.2862   0.1911    0.172    0.165   0.1857
## Balanced Accuracy      1.0000   1.0000    1.000    1.000   1.0000
```

As the Random Forest is 100% accurate, this is a very good model for predicting.

3. Gradient Boosted Trees

```r
#Gradient boosted trees

set.seed(125)
modBT <- train(classe~., data=trainData, method="gbm", trControl = cv, verbose = F)
BT <- predict(modBT, validationData)
BT_results <- confusionMatrix(BT, factor(validationData$classe))
BT_results
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 887  17   0   0   1
##          B  10 577  14   1   1
##          C   3   7 523  17   3
##          D   0   0   3 500  10
##          E   0   0   1   1 569
##
## Overall Statistics
##
##                Accuracy : 0.9717
##                  95% CI : (0.9653, 0.9772)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9642
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9856   0.9601   0.9667   0.9634   0.9743
## Specificity            0.9920   0.9898   0.9885   0.9950   0.9992
## Pos Pred Value         0.9801   0.9569   0.9458   0.9747   0.9965
## Neg Pred Value         0.9942   0.9906   0.9931   0.9928   0.9942
## Prevalence             0.2862   0.1911   0.1720   0.1650   0.1857
## Detection Rate         0.2820   0.1835   0.1663   0.1590   0.1809
## Detection Prevalence   0.2878   0.1917   0.1758   0.1631   0.1816
## Balanced Accuracy      0.9888   0.9749   0.9776   0.9792   0.9868
```

Since Gradient Boosted Tree model indicate over 95% accuracy, it is also a very good model for predicting with this particular dataset.

From the above analysis, Random Forest models performs the best with 100% accuracy. Additionally, since the out of sample error is equal to 1- accuracy, it will be zero for the Random Forest model, while that of Desicion Tree model will be the highest.

RF with test data

Finally, the test data will be analysed with the Random Forest model(the best performing model), to predict the 20 test cases.

```r
RF1 <- predict(modRF, testData)
RF1
```

```
##  [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
```