Practical Machine Learning Project

Desiree Leentjie Wilson

13/05/2021

Introduction

Given the data sets from accelerometers on the belt, forearm, arm, and dumbell of 6 research study participants, we need to create a model and use predictions to show how practical machine learning can be used. We have two datasets, which are the Trainig data and Testing Data.We are required to predict for the tested labels.

Backgraound

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).

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://web.archive.org/web/20161224072740/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.

Loading the required libraries for the project

```
library(caret)
```

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

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.0.5

## Loading required package: ggplot2

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

```r
library(ggplot2)
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

## 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.
```

```r
library(corrplot)
```

```
## corrplot 0.88 loaded
```

Reading the data

```r
urlTraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

urlTesting <-  "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```r
trainingData <- read.csv(url(urlTraining))

testingData <- read.csv(url(urlTesting))
```

Creating a partition for training data

```r
inTrain  <- createDataPartition(trainingData$classe, p=0.7, list=FALSE)
trainSet <- trainingData[inTrain, ]
testSet  <- trainingData[-inTrain, ]
```

Showing dimensions and head of train Set and test set to get the feel of the data we are working with.

```r
dim(trainSet)
```

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

```
dim(testSet)
```

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

We can see that our data has a lot of NA;s so we have to work with clean data.

Cleaning the Data

```
remNA <- sapply(trainSet, function(x) mean(is.na(x))) > 0.95
trainSet <- trainSet[, remNA==F]
testSet <- testSet[, remNA==F]
```

Remove variables with nearly zero variance

```
nzv <- nearZeroVar(trainSet)
trainSet <- trainSet[, -nzv]
testSet <- testSet[, -nzv]
```

The first five variables do not make sense so we remove them

```
trainSet <- trainSet[, -(1:5)]
testSet<- testSet[, -(1:5)]

dim(trainSet)
```

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

```
head(trainSet,1)
```

```
##   num_window roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x
## 1         11      1.41       8.07    -94.4                3            0
##   gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y accel_belt_z
## 1            0        -0.02          -21            4           22
##   magnet_belt_x magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm
## 1            -3           599          -313     -128      22.5    -161
##   total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y
## 1              34           0           0       -0.02        -288         109
##   accel_arm_z magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell
## 1        -123         -368          337          516      13.05217
##   pitch_dumbbell yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x
## 1        -70.494    -84.87394                   37                0
##   gyros_dumbbell_y gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y
## 1            -0.02                0             -234               47
##   accel_dumbbell_z magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z
## 1             -271              -559               293               -65
##   roll_forearm pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1         28.4         -63.9        -153                  36            0.03
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1               0           -0.02             192             203
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z classe
## 1            -215              -17              654              476      A
```

We can see that our data set is clean from just showing the first line.

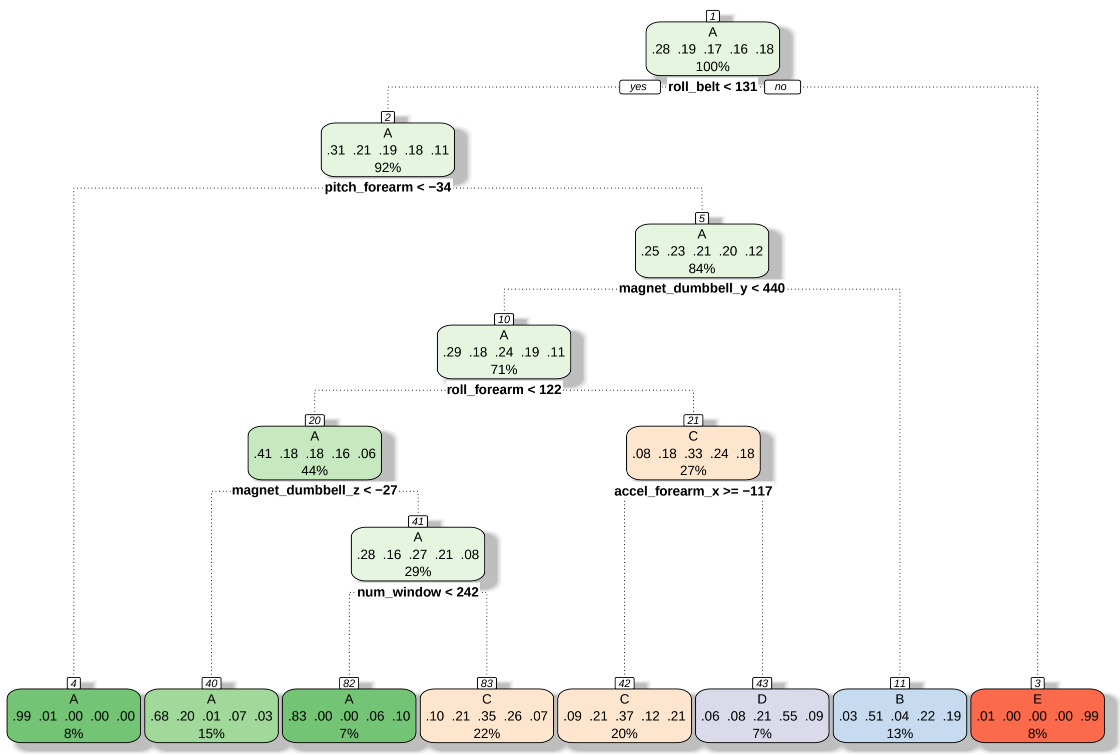
Testing Models

we are going to test a few models.

Decision Tree Model

Fit Model

```
control <- trainControl(method="cv", number=3, verboseIter=FALSE)

modelTree <- train(classe~., data=trainSet, method="rpart", trControl = control, tuneLength = 5)
fancyRpartPlot(modelTree$finalModel)
```



Rattle 2021−May−14 01:59:52 zandile

Prediction

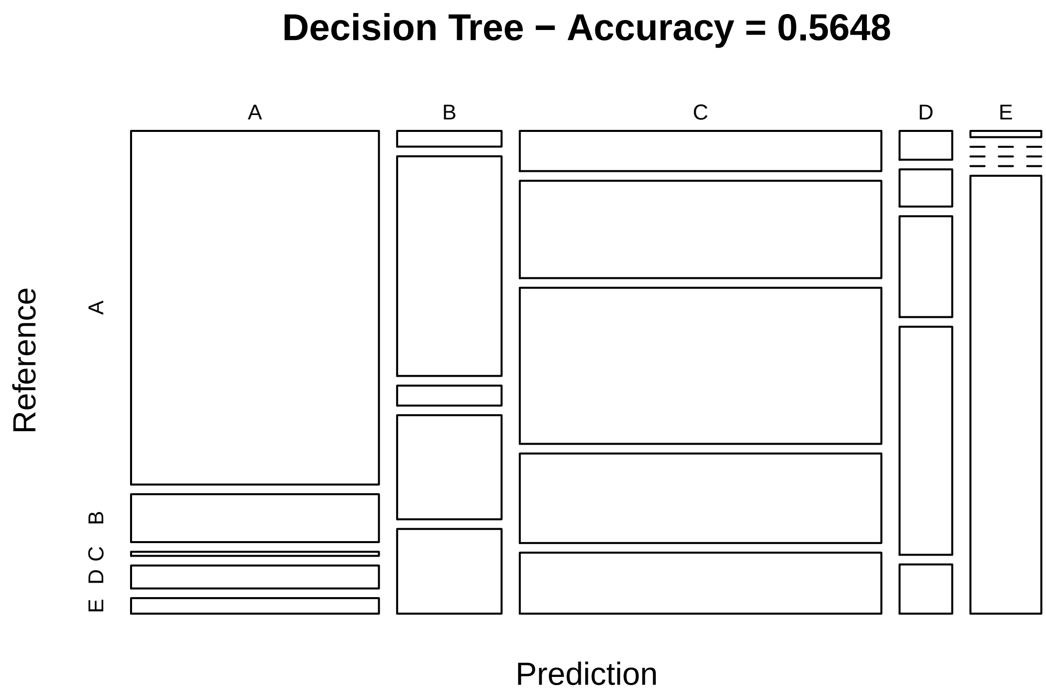
```
predTree <-  predict(modelTree, testSet)
confMatTree <- confusionMatrix(predTree, factor(testSet$classe))
confMatTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1387  188   16   90   61
##          B   26  363   33  172  140
##          C  230  557  893  512  349
```

```
##          D   24   31   84  190   41
##          E    7    0    0    0  491
##
## Overall Statistics
##
##                Accuracy : 0.5648
##                  95% CI : (0.552, 0.5775)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4495
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8286  0.31870   0.8704  0.19710  0.45379
## Specificity            0.9157  0.92183   0.6608  0.96342  0.99854
## Pos Pred Value         0.7962  0.49455   0.3514  0.51351  0.98594
## Neg Pred Value         0.9307  0.84935   0.9602  0.85966  0.89029
## Prevalence             0.2845  0.19354   0.1743  0.16381  0.18386
## Detection Rate         0.2357  0.06168   0.1517  0.03229  0.08343
## Detection Prevalence   0.2960  0.12472   0.4318  0.06287  0.08462
## Balanced Accuracy      0.8721  0.62026   0.7656  0.58026  0.72617
```

The prediction of the decision tree is of levels is shown above and has n=20 and 5 levels.

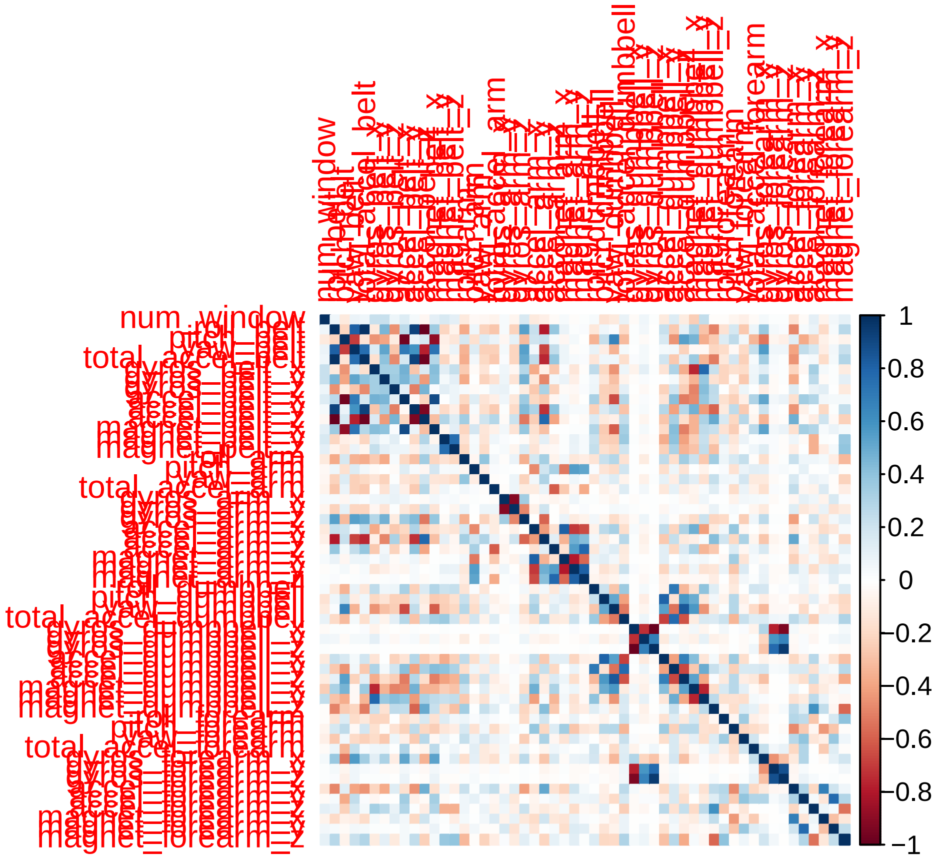
```
plot(confMatTree$table, col = confMatTree$byClass,
     main = paste("Decision Tree - Accuracy =",

                  round(confMatTree$overall['Accuracy'], 4)))
```



The chosen prediction model is the decision tree, the accuracy is 56% just above average however there are other better models that can be chosen, the random Forest Tree seems to be a better model than the Decision tree as it has the most accuracy.

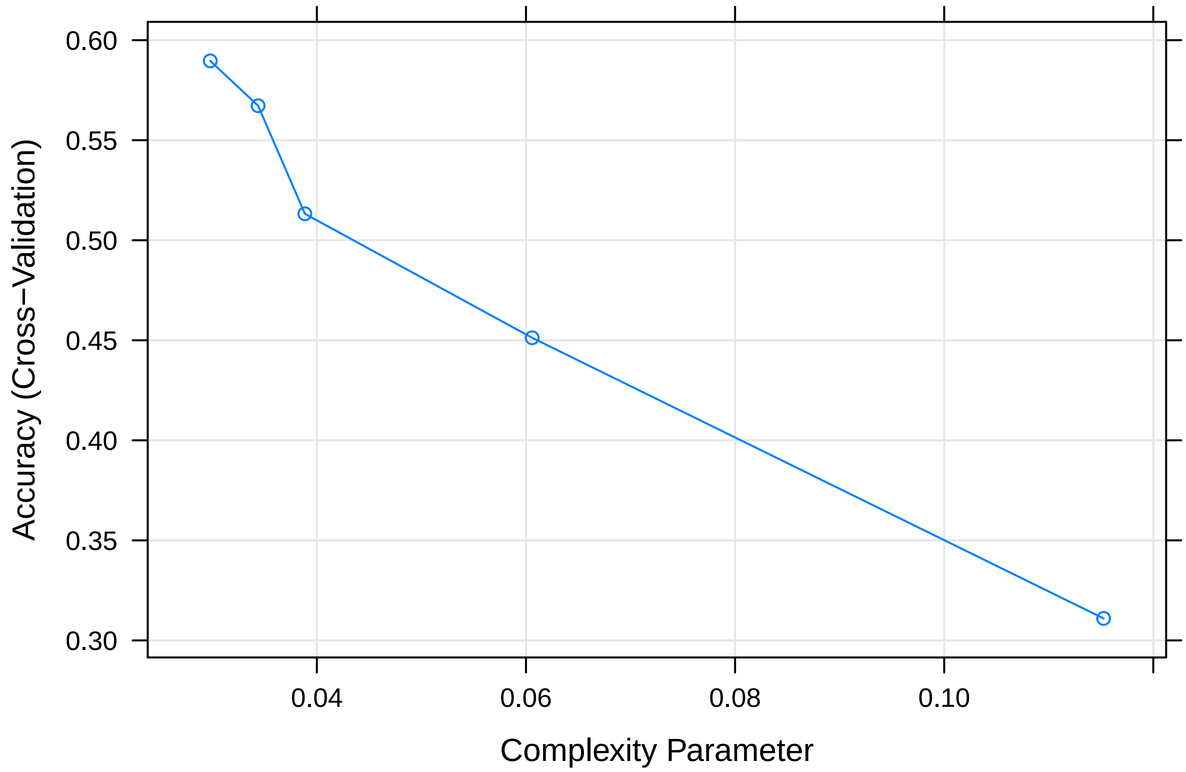
Appendix

```
corPlot <- cor(trainSet[, -length(names(trainSet))])
corrplot(corPlot, method="color")
```



Decision model Plot

```
plot(modelTree)
```



0.60
0.55
0.50
0.45
0.40
0.35
0.30
Accuracy (Cross−Validation)
0.04
0.06
0.08
0.10
Complexity Parameter