Practical Machine Leaarning peer graded

Astha Agarwal

10/23/2020

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, the 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.

The goal of this project is to predict the manner in which they did the exercise. This is the `classe` variable in the training set, considering any of the other variables as predictors.

```
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.3

library(knitr)
library(data.table)

## Warning: package 'data.table' was built under R version 3.6.3

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.6.3

## Loading required package: rpart

library(rpart)
library(gbm)

## Warning: package 'gbm' was built under R version 3.6.3

## Loaded gbm 2.1.8
```

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

## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded
```

Exploring and cleaning the data.

```r
TU <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
taU  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
ts <- read.csv(url(TU))
dt <- read.csv(url(taU))
```

cleaning the data.

```r
tdt <- dt[, colSums(is.na(dt)) == 0]
TD <- ts[, colSums(is.na(ts)) == 0]
```

now predicting

```r
tdt <- tdt[, -c(1:7)]
TD <- TD[, -c(1:7)]
dim(tdt)

## [1] 19622    86
```

now we are deleting the variables that are non-zero referred to as ‘nz’ in this code

```r
set.seed(1234)
dtran <- createDataPartition(dt$classe, p = 0.7, list = FALSE)
tdt <- tdt[dtran, ]
TD <- tdt[-dtran, ]
dim(tdt)

## [1] 13737    86

dim(TD)

## [1] 4123   86
```

coercing

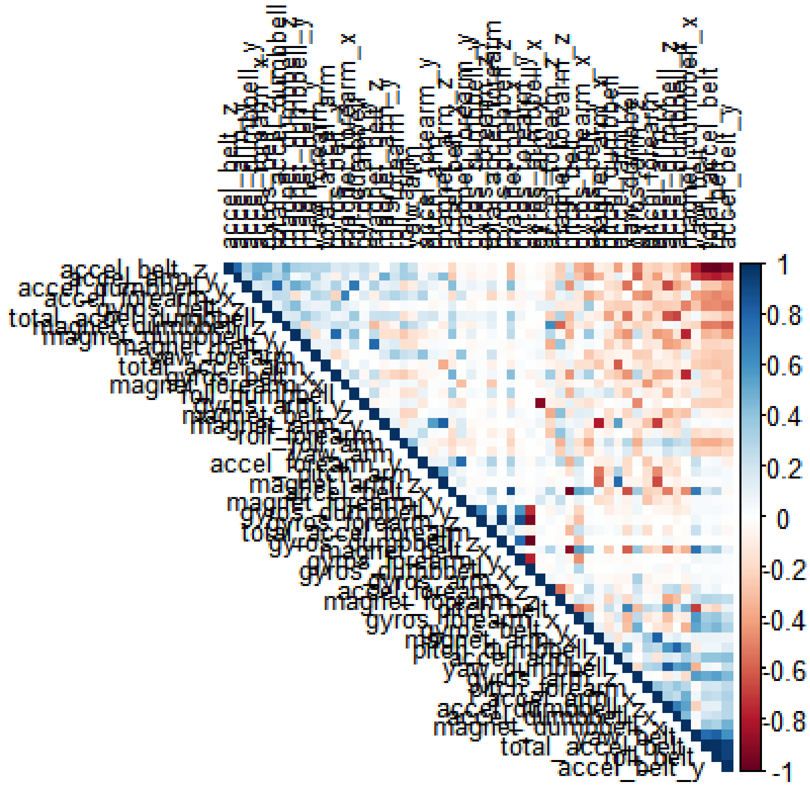
```r
nZ <- nearZeroVar(tdt)
tdt <- tdt[, -nZ]
TD <- TD[, -nZ]
dim(tdt)

## [1] 13737    53

dim(TD)

## [1] 4123   53
```

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

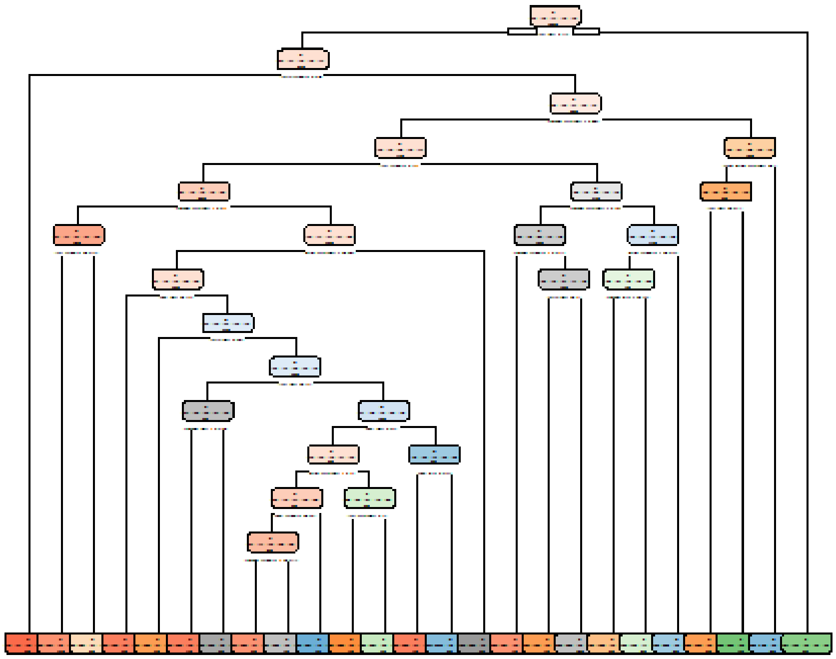


the corr. predic. are with the dark colour intersec.This is the observation in this case.

Next step is the building of our model for the dataset we are using.

```r
set.seed(20000)
tr <- rpart(classe ~ ., data=tdt, method = "class")
rpart.plot(tr)

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



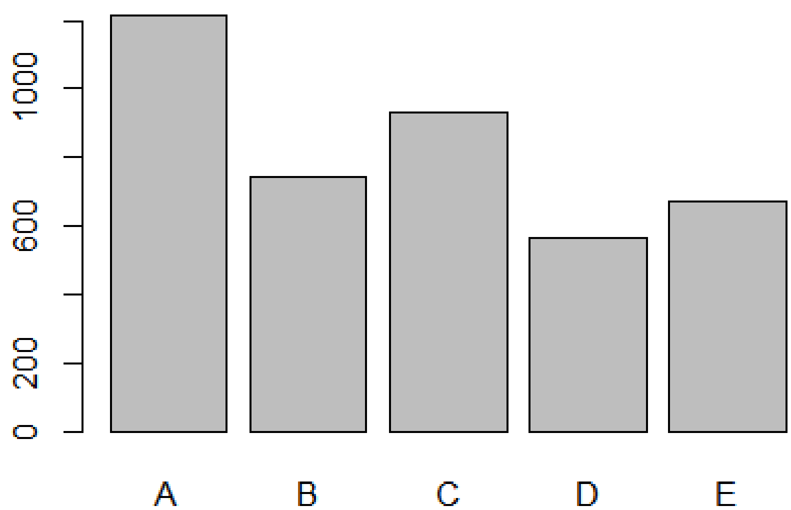
Prepare data partition, for later validation

```
mdp <- predict(tr, TD, type = "class")
ab <- confusionMatrix(mdp, TD$classe)
ab

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1067  105    9   24    9
##          B   40  502   59   63   77
##          C   28   90  611  116   86
##          D   11   49   41  423   41
##          E   19   41   18   46  548
##
## Overall Statistics
##
##                Accuracy : 0.7642
##                  95% CI : (0.751, 0.7771)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7015
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9159   0.6379   0.8279   0.6295   0.7201
## Specificity            0.9503   0.9284   0.9055   0.9589   0.9631
## Pos Pred Value         0.8789   0.6775   0.6563   0.7487   0.8155
## Neg Pred Value         0.9663   0.9157   0.9602   0.9300   0.9383
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2588   0.1218   0.1482   0.1026   0.1329
## Detection Prevalence   0.2944   0.1797   0.2258   0.1370   0.1630
## Balanced Accuracy      0.9331   0.7831   0.8667   0.7942   0.8416
```

```
plot(mdp)
```



Lets apply two models in this case: First is General boosted model. Second is gbm model.

```
set.seed(10000)
cand_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
val <- train(classe ~ .,data=tdt, method = "gbm", trControl = cand_gbm,
verbose = FALSE)
val$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

From the results, it appears that the random forest model has the best accuracy for testing dataset.

Conclusion

We can conclude by saying that RandomForest gives more accurate results than Decision Tree. Finally, I chosed the random forest model to the testing dataset for submission result.