Practical ML

Ayush Kumar

20/10/2020

Aim of this report

The goal of our project is to predict the manner in which they did the exercise. We use the "classe" variable in the training set. We may use any of the other variables to predict with. We have created a report describing how we built our model, how we used cross validation, what we think the expected out of sample error is, and why we made these choices.

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Source

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

Loading Libraries and Data

```r
library(caret)
```

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

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

```r
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.
```

```r
library(rpart)
library(RColorBrewer)
library(rpart.plot)
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

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

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
var_valid<- read.csv('pml-testing.csv', header=T)
var_train<- read.csv('pml-training.csv', header=T)
```

Cleaning and Exploring the data

After loading the data we will clean it in-order to remove non-usable rows and to also remove the NA values.

```
#Removing the NA values
training<- var_train[, colSums(is.na(var_train)) == 0]
validating<- var_valid[, colSums(is.na(var_valid)) == 0]
```

```
#Removing 7 rows which are irrelevant
sub_training1 <- training[, -c(1:7)]
sub_validating1 <- validating[, -c(1:7)]
```

Cleaning Data

Cross Validation and Preparing of Dataset We are splitting data into two parts. One part which is 70% will be used for training and the remaining 30% for testing.

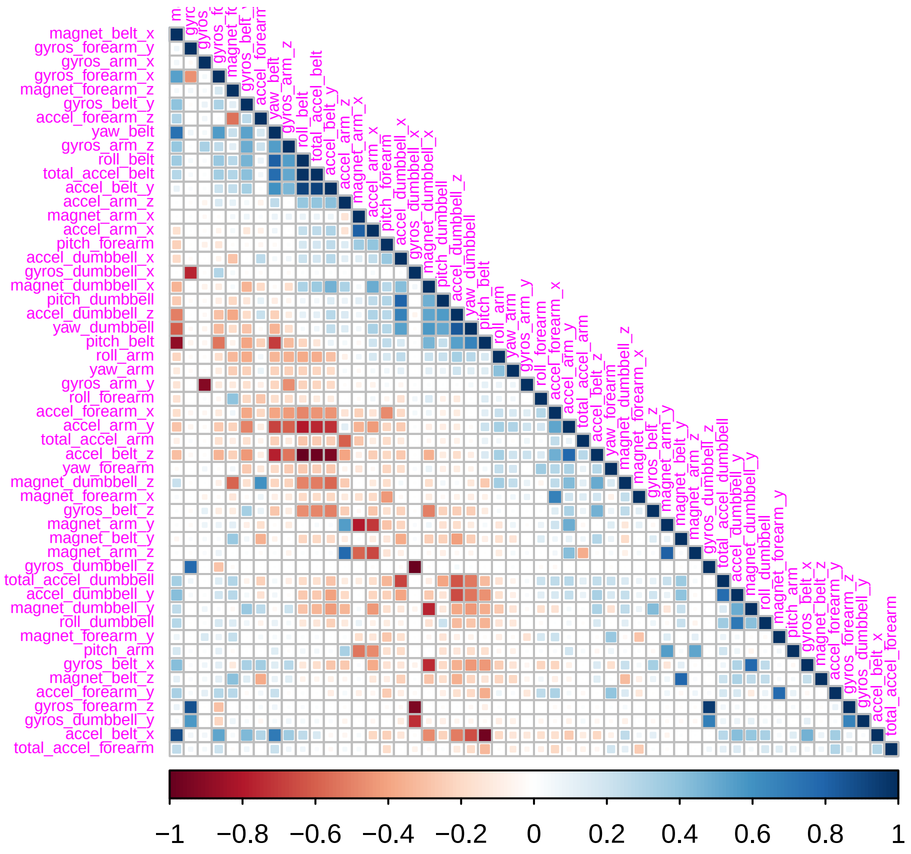
```
#Cross-Validation
set.seed(123)
Train <- createDataPartition(sub_training1$classe, p = 0.7, list = FALSE)
```

```
#Cross-Validation and Preparing
sub_training2 <- sub_training1[Train, ]
sub_testing2 <- sub_training1[-Train, ]
```

```
#Removing variables near to zero-variance
nearzero <- nearZeroVar(sub_training2)
sub_testing3  <- sub_testing2[, nearzero]
sub_training3 <- sub_training2[, -nearzero]
```

Another step of cleaning We are now going to plot the data we have available to see how much data we have cleaned.

```
#Plotting to see how much we have cleaned
cor_mat <- cor(sub_training3[, -53])
corrplot(cor_mat, order = "AOE", method = "square", type = "lower",
         tl.cex = 0.5, tl.col = rgb(1, 0, 1))
```



Bulding our Prediction Model

We are going to build random forest model and check it's accuracy on the cleaned data which we have made and then we use train function along with new cleaned data for three-crossed validation.

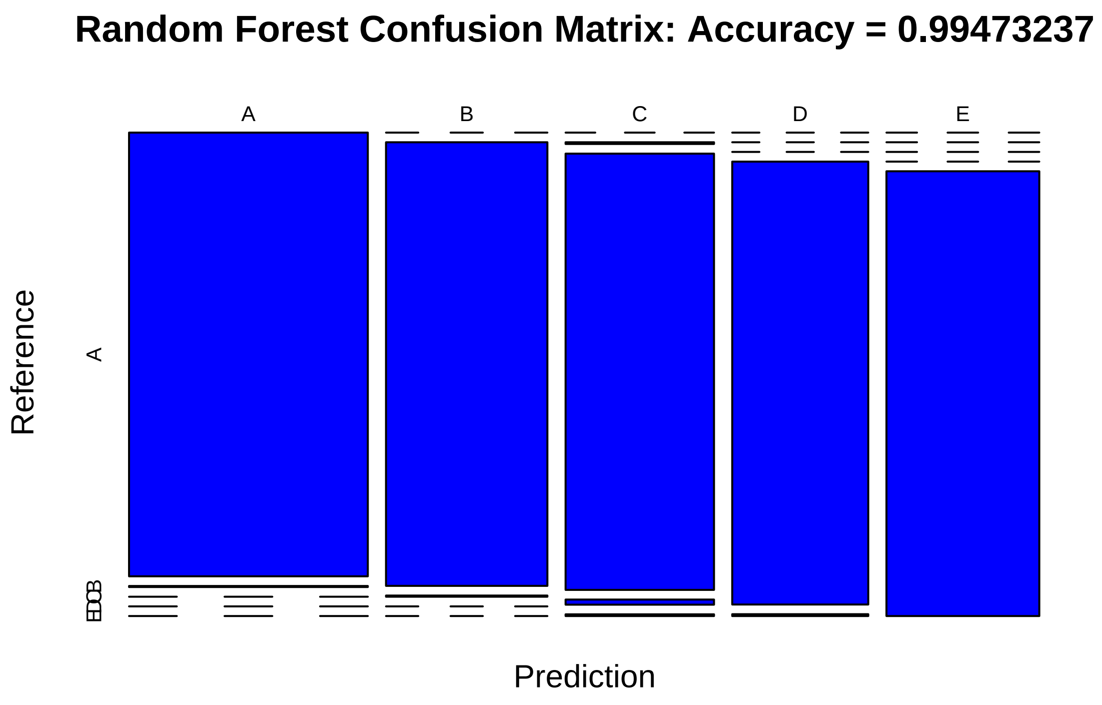
```
#Random Forest Model creation
random_forest_model <- randomForest(as.factor(classe) ~ ., data=sub_training2, method="class")
prediction_random_forest <- predict(random_forest_model, sub_testing2, type = "class")
```

```
#We are using this confusion_matrix to check its accuracy
cmrf<-confusionMatrix(prediction_random_forest, factor(sub_testing2$classe))
```

Random Forest Model We get a very high accuracy of 0.99473237. With this we will plot it on a graph

```
#Plotting the accuracy of Random Forest model on graph

plot(cmrf$table,col="blue", main = paste("Random Forest Confusion Matrix: Accuracy =", round(cmrf$overal
```



Hence due to it's high accuracy we could say that Random Forest model can be used.

Prediction on a rebuild model Now we are going to train our model again so we don't predict on a reduced set of data

```
#Again removing variable with zero variance but on raw data
Train_again <- nearZeroVar(var_train)
testing_again <- var_valid[, -Train_again]
training_again <- var_train[, -Train_again]
```

We need to again remove NA values as it will cause error in the process

```
#Removing NA values
not_applicable <- sapply(training_again, function(z) mean(is.na(z))) > 0.95
```

```
#Removing NA values
training_again_2 <- training_again[, not_applicable==F]
testing_again_2 <- testing_again[, not_applicable==F]
```

This time we will remove the first 5 variables which basically are varaibles which can't be used in our data like time-stamp etc.

```
#removing un-used data
training_again_3 <- training_again_2[, -(1:5)]
testing_again_3 <- testing_again_2[, -(1:5)]
```

We will use the train function in rf mode to train our data

```
#Training of data
variable_control <- trainControl(method="cv", number=3, verboseIter=F)
fitness_model <- train(classe ~ ., data=training_again_3, method="rf", trControl=variable_control)
final_model <- predict(fitness_model, newdata=testing_again_3)
```

Finally we predict the analysis on new test set

```
#testing and converting to character
final_model <- predict(fitness_model, newdata=testing_again_3)
final_model<- as.character(final_model)
```

Conclusion

We could easily see that due to it’s high accuracy Random Forest Model is the best fit for this data set and so we will go ahead with it.

```
#Using the Random Forest Algorithm we generate files for submission.
variable_write_PML <- function(z) {
  len <- length(z)
  for(k in 1:len) {
    file_name_var = paste0("problem_id_", k, ".txt")
    write.table(z[k], file=file_name_var, quote=F, row.names=F, col.names=F)
  }
}
```

```
#Final submission
variable_write_PML(final_model)
```

Sumbission