Practical Machine Learning: Final Project

VR SREEGANESH

15/09/2020

Overview

With the coming of highly personal devices such as JawBone Up, Nike FuelBand and fitbit, it is now possible to collect a large amount of data relating to personal activity. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm and dumbell of six participants. They were asked to perform barbell lifts correctly and incorrectly in five different ways.

Loading Data

The following is the code snippet required for loading the training data and testing data.

```r
training_data_from_source <- read.csv(
        url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),
        header = TRUE)
dim(training_data_from_source)
```

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

```r
testing_data_from_source <- read.csv(
        url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),
        header = TRUE)
dim(testing_data_from_source)
```

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

Data PreProcessing

In this step, we remove the first seven columns because they are redundant labels and doesn't add any information to our process.

```r
# We remove the first seven columns cause they mainly contain useless data
training_data_after_removing_seven_columns <- training_data_from_source[,-c(1:7)]
testing_data_after_removing_seven_columns <- testing_data_from_source[,-c(1:7)]
```

Now, we remove those columns with NA's or empty spaces occupying more than 90 percent of its values.

```r
indices_of_columns_to_remove <- which(
        (colSums(is.na(training_data_after_removing_seven_columns))/dim(training_data_after_removing_se
training_data_after_removing_big_NA <- subset(
        training_data_after_removing_seven_columns,
        select = -c(indices_of_columns_to_remove))
```

```
indices_of_columns_to_remove <- which(
        colSums(training_data_after_removing_big_NA == "")/ dim(training_data_after_removing_big_NA)[1]
training_data_after_removing_spaces <- subset(
        training_data_after_removing_big_NA,
        select = -c(indices_of_columns_to_remove))
```

Now, we need to find those columns that have a high correlation with each other. We then remove those variables so as to reduce the bias.

```
#choosing the factors that are actually important
correlation_matrix <- cor(training_data_after_removing_spaces[,-53])

# building a correlation plot
library(corrplot)
```

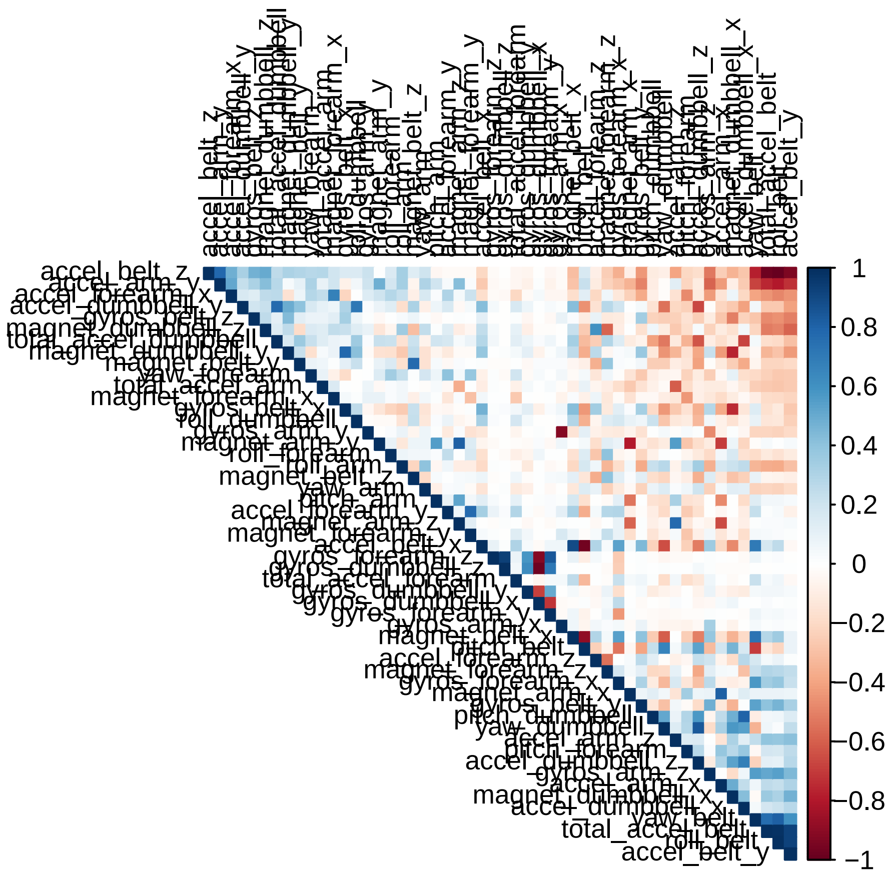
```
## corrplot 0.84 loaded
```

```
corrplot(correlation_matrix,
         order = "FPC",
         method = "color",
         type = "upper",
         tl.cex = 0.8,
         tl.col = rgb(0,0,0))

# finding the variables that are highly correlated
library(caret) # for findCorrelation
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```



```
highlyCorrelated <- findCorrelation(
        correlation_matrix,
        cutoff = 0.75)

# The training data after removing the indices that are too correlated
trainingData <- subset(
        training_data_after_removing_spaces,
        select = -c(highlyCorrelated))
```

Now, we split the dataset into two. We make a training set and a test set. The training set is to train different models. The testing set is to choose the model that performs the best.

```
#splitting the data to train and test data
inTrain <- createDataPartition(
        y = trainingData$classe,
        p = 0.7,
        list = FALSE)
trainData <- trainingData[inTrain,]
testData <- trainingData[-inTrain,]
```

Building a model

For this part of the project, we mainly build three models. They are the RPART model,the RandomForest model and gbm model, which is basically boosting with Trees.

We build all three models with the training data as 'trainData'. We then build a confusion matrix for each of those models and we find the accuracy of the model from those matrices.

```r
#building an rpart model.
modelFitrpart <- train(
        classe ~ .,
        data = trainData,
        method = "rpart")
confusionrpart <- confusionMatrix(
        as.factor(testData$classe),
        predict(
                modelFitrpart,
                newdata = testData))
confusionrpart <- as.matrix(confusionrpart)
accuracyrpart <- sum(diag(confusionrpart))/sum(confusionrpart)
print("The following is the confusion matrix for the rpart model")
```

```
## [1] "The following is the confusion matrix for the rpart model"
```

```r
confusionrpart
```

```
##     A   B   C   D   E
## A 995 310 272  95   2
## B 162 698 229  40  10
## C  25 211 745  44   1
## D  36 321 248 274  85
## E  12 468 192  42 368
```

```r
print(c("The accuracy of this model is",accuracyrpart))
```

```
## [1] "The accuracy of this model is" "0.523364485981308"
```

```r
# building a RF model.
modelFitRF <- train(
        classe ~ .,
        data = trainData,
        method = "rf",
        trControl = trainControl(method = "cv"),
        number = 3)
confusionRF <- confusionMatrix(
        as.factor(testData$classe),
        predict(
                modelFitRF,
                newdata = testData))
confusionRF <- as.matrix(confusionRF) # making it a matrix
accuracyRF <- sum(diag(confusionRF))/sum(confusionRF)
print("The following is the confusion matrix for the Random Forests model")
```

```
## [1] "The following is the confusion matrix for the Random Forests model"
```

```r
confusionRF
```

```
##      A    B    C   D    E
## A 1674    0    0   0    0
## B    5 1129    4   0    1
## C    0    5 1019   2    0
## D    0    0   13 946    5
## E    0    0    2   2 1078
```

```r
print(c("The accuracy of this model is",accuracyRF))
```

```
## [1] "The accuracy of this model is" "0.993372982158029"
```

```r
# building a gbm model.
modelFitgbm <- train(
        classe ~ .,
        data = trainData,
        method = "gbm",
        verbose = FALSE)
confusionGBM <- confusionMatrix(
        as.factor(testData$classe),
        predict(
                modelFitgbm,
                newdata = testData))
confusionGBM <- as.matrix(confusionGBM)
accuracyGBM <- sum(diag(confusionGBM))/sum(confusionGBM)
print("The following is the confusion matrix for the gbm model")
```

```
## [1] "The following is the confusion matrix for the gbm model"
```

```r
confusionGBM
```

```
##      A    B   C   D    E
## A 1641   14   4  15    0
## B   43 1036  43   5   12
## C    0   47 963  14    2
## D    3   11  45 898    7
## E    5   18  14  16 1029
```

```r
print(c("The accuracy of this model is",accuracyGBM))
```

```
## [1] "The accuracy of this model is" "0.945964316057774"
```

We see that the Random Forest model is the one that gives the highest accuracy. Thus, we shall use this model as our model. We shall run this model on the test set.

Running the Model on the Test Set

The following is the result produced by our model on the dataset that was provided as test set.

```r
finalResult <- predict(modelFitRF, newdata = testing_data_from_source)
print(finalResult)
```

```
##  [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
```