MACHINE LEARNING PROJECT

Rohit Kharat

16/09/2020

Introduction

There is 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. The participants under consideration were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The main purpose of the project is to predict the manner in which they did the exercise.

The data for this project comes from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and can be found here.

Data Loading and Preprocessing

First, let's load the required R packages.

```
suppressPackageStartupMessages({
        library(caret)
        library(randomForest)
        library(corrplot)
        library(randomForest)
        library(rpart)
        library(rpart.plot)
        library(rattle)
})
```

Now, let's download the training and testing data sets and read them into R.

```
URLtrain<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(URLtrain, "traindata.csv")
traindata <- read.csv("traindata.csv")
dim(traindata)

## [1] 19622   160

URLtest<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(URLtest, "testdata.csv")
testdata <- read.csv("testdata.csv")
dim(testdata)
```

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

```
table(traindata$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

As stated in the assignment instructions, the target variable is the **classe** variable.

Then, let's do some preprocessing on the data sets. First, we will eliminate the variables that have variances close to zero in both data sets. After that, we will eliminate the columns containing the missing values.

```
# Eliminating the variables with near zero variances
NZV <- nearZeroVar(traindata)
traindata <- traindata[ , -NZV]
testdata <- testdata[ , -NZV]
# Eliminating the variables with missing values
train_wo_na <- traindata[ , colSums(is.na(traindata))==0]
test_wo_na <- testdata[ , colSums(is.na(testdata))==0]
# Eliminating the unnecessary columns
unnecColumnsTrain <- c("X", "user_name", "raw_timestamp_part_1",
         "raw_timestamp_part_2", "cvtd_timestamp", "num_window")
unnecColumnsTest <- c("X", "user_name", "raw_timestamp_part_1",
         "raw_timestamp_part_2", "cvtd_timestamp", "num_window", "problem_id"
)
traindata_clean<-train_wo_na[,!(names(train_wo_na) %in% unnecColumnsTrain)]
testdata_clean<-test_wo_na[,!(names(test_wo_na) %in% unnecColumnsTest)]
dim(traindata_clean)
```

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

```
dim(testdata_clean)
```

```
## [1] 20 52
```

```
# the test data set will be used for applying the best model to 20 test cases
.
```

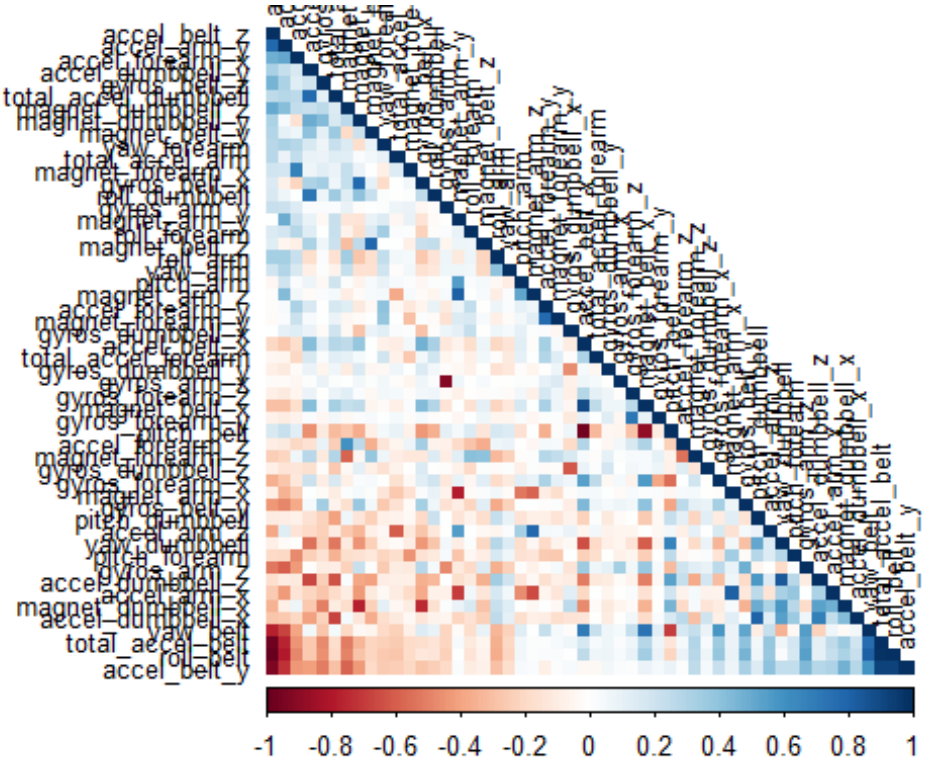
Now, we need to split the training data into training data set and validation data set.

```
set.seed(09132020)
split <- createDataPartition(y = traindata$classe, p = 0.7, list = FALSE)
train_clean <- traindata_clean[split,] # the dataset for building the model
validation_clean<-traindata_clean[-split,] #the dataset for validating the mo
del
```

Model Building

Before we start building the models, let's first look at the correlation between the variables.

```
correlationMatrix <- cor(train_clean[, -53])
corrplot(correlationMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



In the graph above the correlated variables are presented in dark colors. Since the number of correlations is quite small the linear regression model will not be appropriate for this case.

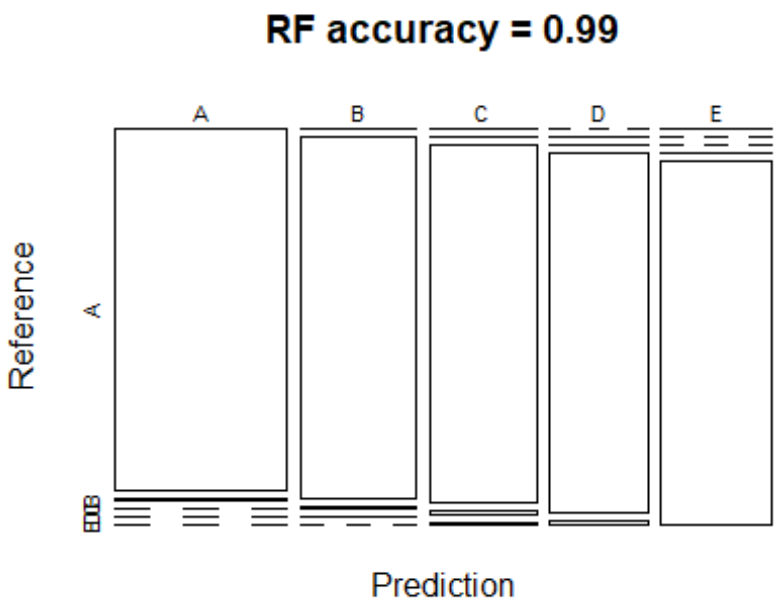
1. Now, let's first try the **Random Forest method**.

```
set.seed(448)
RFcontrol <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
RFmodel<-train(classe ~ .,method = "rf",data = train_clean,trControl=RFcontro
l)
RFmodel$finalModel

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    3    2    0    1 0.001536098
## B   24 2626    8    0    0 0.012039127
## C    0   11 2376    9    0 0.008347245
```

```
## D    0    1   26 2224    1 0.012433393
## E    1    1    4    4 2515 0.003960396
```

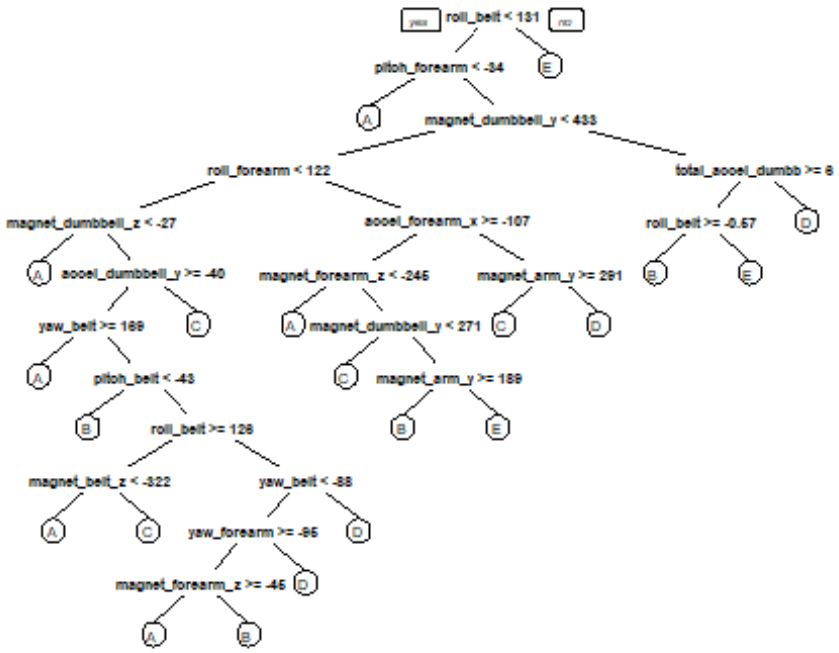
Let's test the RF model using the **validation_clean** data set.

```
RFpredict <- predict(RFmodel, newdata = validation_clean)
RFconfusionMatrix<-confusionMatrix(RFpredict,as.factor(validation_clean$class
e))
RFconfusionMatrix

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    6    0    0    0
##          B    1 1129    3    1    0
##          C    1    3 1019   14    2
##          D    0    1    4  947    8
##          E    1    0    0    2 1072
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9894, 0.9941)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9899
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9912   0.9932   0.9824   0.9908
## Specificity            0.9986   0.9989   0.9959   0.9974   0.9994
## Pos Pred Value         0.9964   0.9956   0.9808   0.9865   0.9972
## Neg Pred Value         0.9993   0.9979   0.9986   0.9965   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1918   0.1732   0.1609   0.1822
## Detection Prevalence   0.2850   0.1927   0.1766   0.1631   0.1827
## Balanced Accuracy      0.9984   0.9951   0.9945   0.9899   0.9951

## Plotting the Confusion Matrix
plot(RFconfusionMatrix$table, col = RFconfusionMatrix$byClass,
     main=paste("RF accuracy =",round(RFconfusionMatrix$overall['Accuracy'],2
)))
```



2. Then, we will use the **Decision Tree method**.

```
set.seed(526)
DTmodel <- rpart(classe ~ ., data=train_clean, method="class")
prp(DTmodel)
```



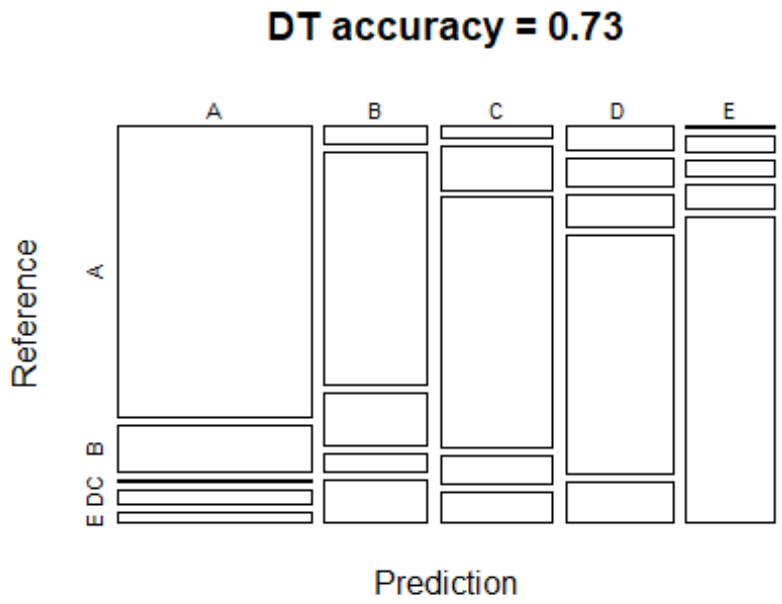
Let's test the Decision Tree Model on our **validation_clean** data set.

```
DTpredict <- predict(DTmodel, newdata = validation_clean, type = "class")
DTconfusionMatrix<-confusionMatrix(DTpredict,as.factor(validation_clean$class
e))
DTconfusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1503  244   11   77   46
##          B   54  644  141   55  114
##          C   38  131  744   90   87
##          D   70   84   91  686  116
##          E    9   36   39   56  719
##
## Overall Statistics
##
##                Accuracy : 0.73
##                  95% CI : (0.7185, 0.7413)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6572
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8978   0.5654   0.7251   0.7116   0.6645
## Specificity            0.9102   0.9233   0.9288   0.9266   0.9709
## Pos Pred Value         0.7990   0.6389   0.6826   0.6552   0.8370
## Neg Pred Value         0.9573   0.8985   0.9412   0.9425   0.9278
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2554   0.1094   0.1264   0.1166   0.1222
## Detection Prevalence   0.3196   0.1713   0.1852   0.1779   0.1460
## Balanced Accuracy      0.9040   0.7444   0.8270   0.8191   0.8177
```

```r
## Plotting the Confusion Matrix
plot(DTconfusionMatrix$table, col = DTconfusionMatrix$byClass,
     main=paste("DT accuracy =",round(DTconfusionMatrix$overall['Accuracy'],2
)))
```



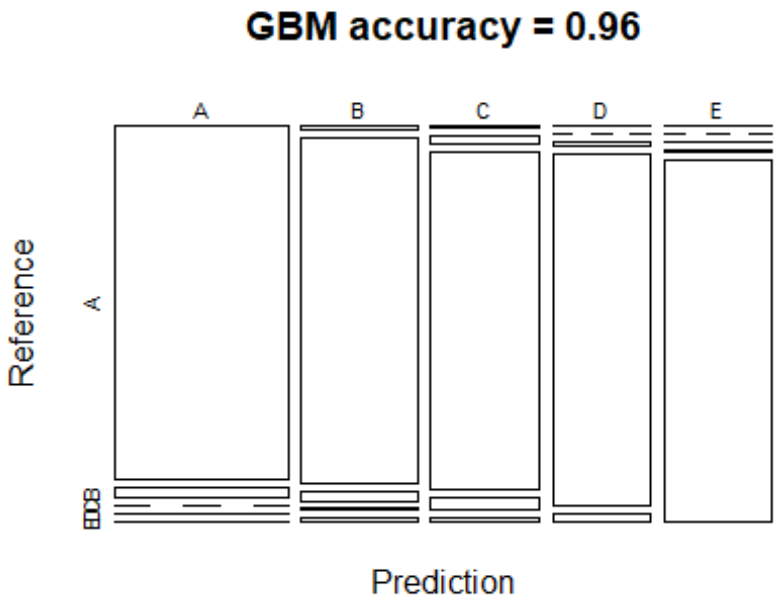
3. Finally, let's consider the **Generalized Boosted Model**

```r
set.seed(550)
GBMcontrol <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
GBMmodel <- train(classe ~ ., data=train_clean, method = "gbm",
                  trControl = GBMcontrol, verbose = FALSE)
GBMmodel$finalModel
```

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

Let's now predict using the **validation_clean** data set.

```
GBMpredict <- predict(GBMmodel, newdata = validation_clean)
GBMconfusionMatrix<-confusionMatrix(GBMpredict,as.factor(validation_clean$cla
sse))
GBMconfusionMatrix

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   41    0    1    1
##          B   14 1076   26    5   10
##          C    9   22  984   35   10
##          D    1    0   14  916   19
##          E    1    0    2    7 1042
##
## Overall Statistics
##
##                Accuracy : 0.963
##                  95% CI : (0.9578, 0.9676)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9531
##
##  Mcnemar's Test P-Value : 1.093e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9851   0.9447   0.9591   0.9502   0.9630
## Specificity            0.9898   0.9884   0.9844   0.9931   0.9979
## Pos Pred Value         0.9746   0.9514   0.9283   0.9642   0.9905
## Neg Pred Value         0.9940   0.9867   0.9913   0.9903   0.9917
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2802   0.1828   0.1672   0.1556   0.1771
## Detection Prevalence   0.2875   0.1922   0.1801   0.1614   0.1788
## Balanced Accuracy      0.9874   0.9665   0.9717   0.9716   0.9805

## Plotting the Confusion Matrix
plot(GBMconfusionMatrix$table, col = GBMconfusionMatrix$byClass,
     main=paste("GBM accuracy =",round(GBMconfusionMatrix$overall['Accuracy']
,2)))
```



Applying the best model to the test data set

For the test the model we will use **testdata_clean** data set.

Since the accuracy rate of the **Random Forest model** is the highest, this model will be applied to the 20 test cases.

```
TestPredict <- predict(RFmodel, newdata = testdata_clean)
TestPredict

##  [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
```