Pratical Machine Learning

Monisha Mohan

2024-06-30

Introduction

In this project, we aim to predict the manner in which exercises were performed using accelerometer data from wearable devices. The dataset contains measurements from the belt, forearm, arm, and dumbbell of 6 participants performing barbell lifts in 5 different ways.

Data Source

The training data for this project are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)]

The test data are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)]

The data for this project come from this source: [http://groupware.les.inf.puc-rio.br/har (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 Dataset

The Dataset has been downloaded from the internet and has been loaded into two seperate dataframes, train_data and test_data. The training data set has 19622 number of records and the test data set has 20 records. The number of variables is 160.

```
library(caret)
library(randomForest)
library(ggplot2)
library(gbm)
# Load the data
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
train_data <- read.csv(train_url)
test_data <- read.csv(test_url)
dim(train_data)
```

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

```
dim(test_data)
```

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

Data Cleaning

Removing Variables which are having nearly zero variance.

```
non_zero_var <- nearZeroVar(train_data)
train_data <- train_data[,-non_zero_var]
test_data <- test_data[,-non_zero_var]
dim(train_data)
```

```
## [1] 19622   100
```

```
dim(test_data)
```

```
## [1]  20 100
```

Removing Variables which are having NA values.

```
na_val_col <- sapply(train_data, function(x) mean(is.na(x))) > 0.90
train_data <- train_data[,na_val_col == FALSE]
test_data <- test_data[,na_val_col == FALSE]
dim(train_data)
```

```
## [1] 19622    59
```

```
dim(test_data)
```

```
## [1] 20 59
```

Removing non-numeric variables

```
train_data <- train_data[,8:59]
test_data <- test_data[,8:59]

dim(train_data)
```

```
## [1] 19622    52
```

```
dim(test_data)
```

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

Splitting Data into train and test

Split the training data into training data 70% of the total data and test data 30% data

```
part <- createDataPartition(train_data$classe, p=0.7, list=FALSE)
train_set <- train_data[part,]
test_set <- train_data[-part,]
dim(train_set)
```

```
## [1] 13737    52
```

```
dim(test_set)
```

```
## [1] 5885   52
```

Random Forest Tree with ntree=30

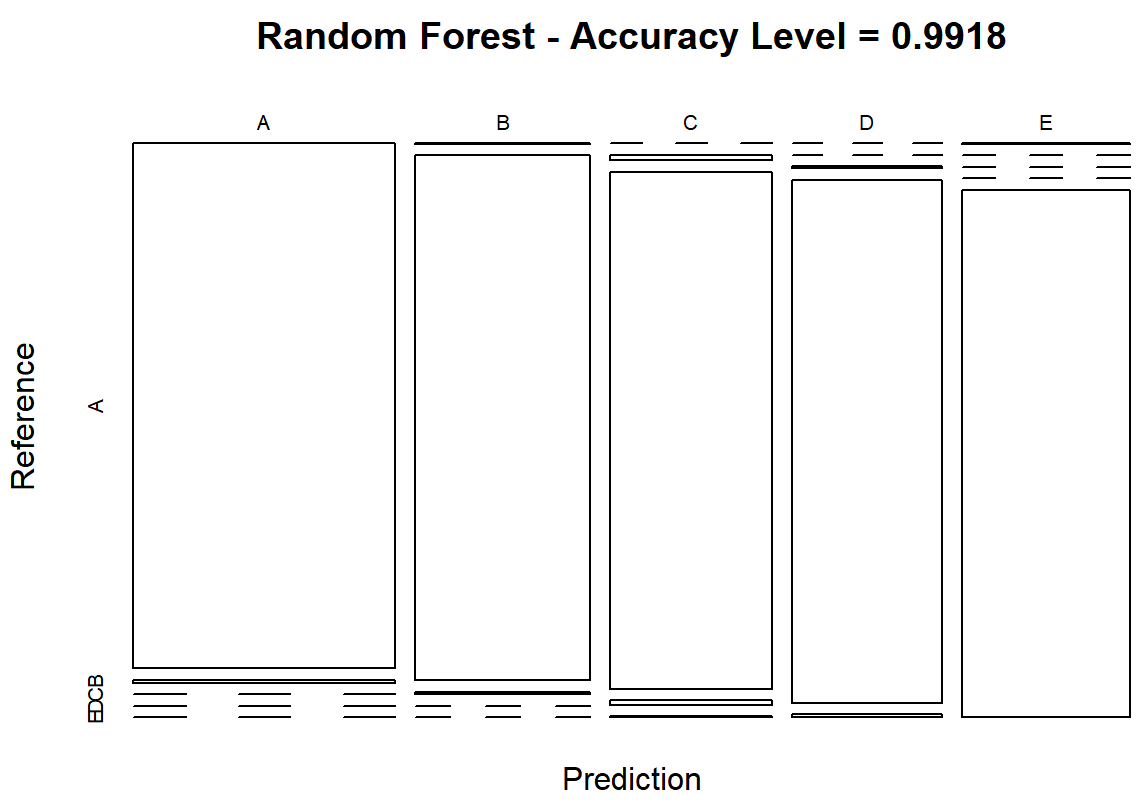
```
RF_30 <- train(classe ~ ., data = train_set, method = "rf", ntree = 30)
RF_30pred <- predict(RF_30, test_set)

ts <- factor(test_set$classe)
pred_30 <- factor(RF_30pred)

RF_30pred_conf <- confusionMatrix(pred_30, ts)
RF_30pred_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   10    0    0    0
##          B    1 1118    5    0    0
##          C    0   11 1017    9    2
##          D    0    0    4  955    5
##          E    1    0    0    0 1075
##
## Overall Statistics
##
##                Accuracy : 0.9918
##                  95% CI : (0.9892, 0.994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9897
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9816   0.9912   0.9907   0.9935
## Specificity            0.9976   0.9987   0.9955   0.9982   0.9998
## Pos Pred Value         0.9941   0.9947   0.9788   0.9907   0.9991
## Neg Pred Value         0.9995   0.9956   0.9981   0.9982   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1900   0.1728   0.1623   0.1827
## Detection Prevalence   0.2858   0.1910   0.1766   0.1638   0.1828
## Balanced Accuracy      0.9982   0.9901   0.9934   0.9944   0.9967
```

```
plot(RF_30pred_conf$table, col = RF_30pred_conf$byClass,
     main = paste("Random Forest - Accuracy Level =",
                  round(RF_30pred_conf$overall['Accuracy'], 4)))
```



Gradient Boosting Model

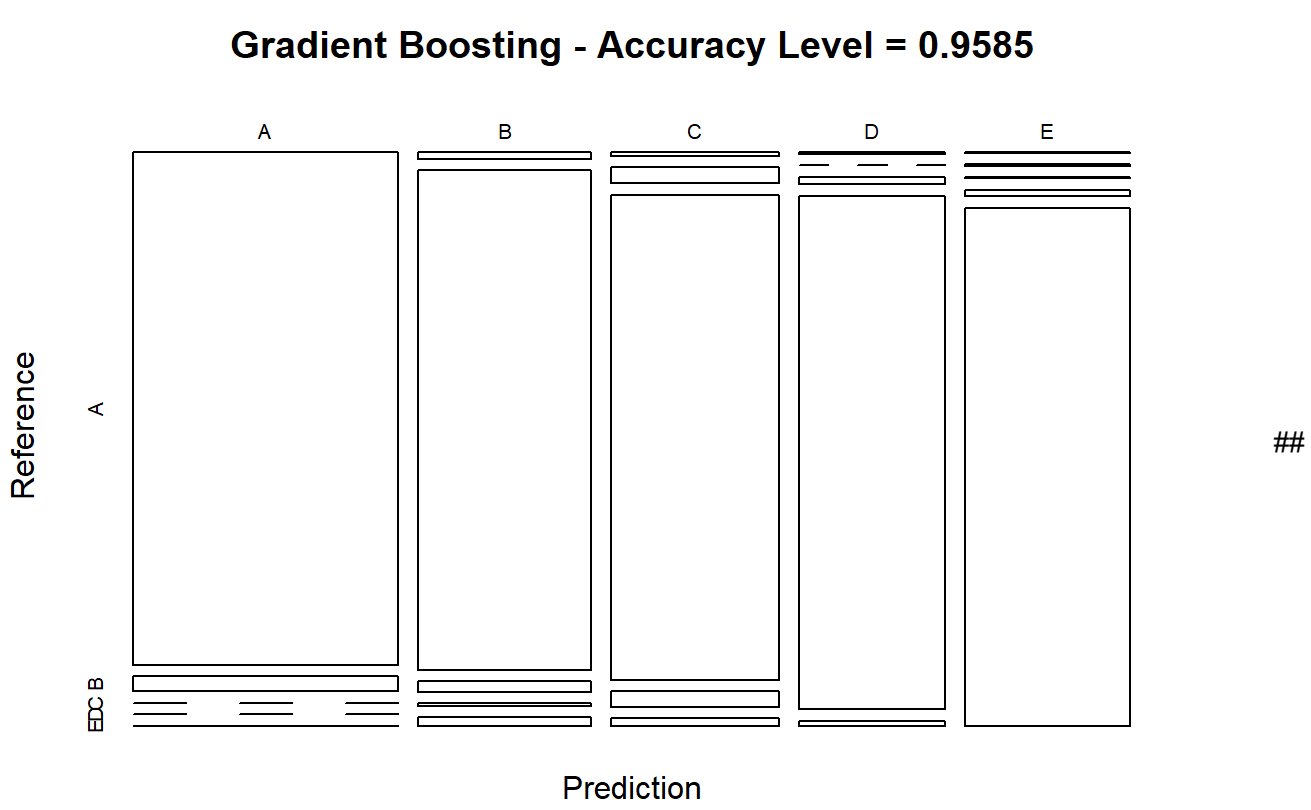
```
GBM_modfit <- train(classe ~ ., data = train_set, method = "gbm", verbose = FALSE)
GBM_modfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 51 had non-zero influence.
```

```
GBM_prediction <- predict(GBM_modfit, test_set)
gbm <- factor(GBM_prediction)
GBM_pred_conf <- confusionMatrix(gbm, ts)
GBM_pred_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   49    0    0    1
##          B   14 1054   23    5   19
##          C    7   33  988   32   16
##          D    3    0   13  914   10
##          E    1    3    2   13 1036
##
## Overall Statistics
##
##                Accuracy : 0.9585
##                  95% CI : (0.9531, 0.9635)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9475
##
##  Mcnemar's Test P-Value : 1.556e-10
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9851   0.9254   0.9630   0.9481   0.9575
## Specificity            0.9881   0.9871   0.9819   0.9947   0.9960
## Pos Pred Value         0.9706   0.9453   0.9182   0.9723   0.9820
## Neg Pred Value         0.9940   0.9822   0.9921   0.9899   0.9905
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2802   0.1791   0.1679   0.1553   0.1760
## Detection Prevalence   0.2887   0.1895   0.1828   0.1597   0.1793
## Balanced Accuracy      0.9866   0.9563   0.9724   0.9714   0.9768
```

```
plot(GBM_pred_conf$table, col = GBM_pred_conf$byClass,
     main = paste("Gradient Boosting - Accuracy Level =",
                  round(GBM_pred_conf$overall['Accuracy'], 4)))
```



Conclusion From the above data, the Random Forest model has definitely more accuracy than GBM. Hence we will be selecting Random Forest model for final prediction from test_data.

Final Prediction on test

```
Final_RF_prediction <- predict(RF_30, test_data )
Final_RF_prediction
```

```
##  [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
```