PracticalMachineLearning

Rishi Krishnan Murugesan

2/10/2023

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity. 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.

The goal of this project is 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.

Data

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

Training Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

Test Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Load required libraries

```
library(lattice)
library(ggplot2)
library(caret)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(rpart)
library(rpart.plot)
```

Data Load, clean and Feature Selection

```


train_url <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test_url <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

train_data <- read.csv(url(train_url), na.strings = c("NA", "", "#DIV/0!"))
test_data <- read.csv(url(test_url), na.strings = c("NA", "", "#DIV/0!"))

dim(train_data)
```

Load Data

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

```
dim (test_data)
```

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

```


set.seed(2023)
input_train <- createDataPartition(train_data$classe, p=0.8, list=FALSE)
training_set = train_data[input_train, ]
test_set = train_data[-input_train, ]
dim(training_set)
```

Create two parition with 80% and 20% from training data

```
## [1] 15699   160
```

```
dim(test_set)
```

```
## [1] 3923  160
```

```
nzv_col <- nearZeroVar(training_set)
training_set <- training_set[ , -nzv_col]
test_set <- test_set[ , -nzv_col]
dim(training_set)
```

Check for near-zero-variance variables and remove them

```
## [1] 15699   126
```

```
dim(test_set)
```

```
## [1] 3923  126
```

```
na_spc_var <- sapply(training_set, function(x) mean(is.na(x))) > 0.8
summary(na_spc_var)
```

Remove features/columns that are mostly(80%) NA

```
##    Mode   FALSE    TRUE
## logical      59      67
```

```
training_set <- training_set[, na_spc_var == FALSE]
test_set <- test_set[, na_spc_var == FALSE]
dim(training_set)
```

```
## [1] 15699    59
```

```
dim(test_set)
```

```
## [1] 3923   59
```

```
training_set <- training_set[, -(1:5)]
test_set <- test_set[, -(1:5)]
dim(training_set)
```

Remove the first 5 columns as they are not useful features

```
## [1] 15699    54
```

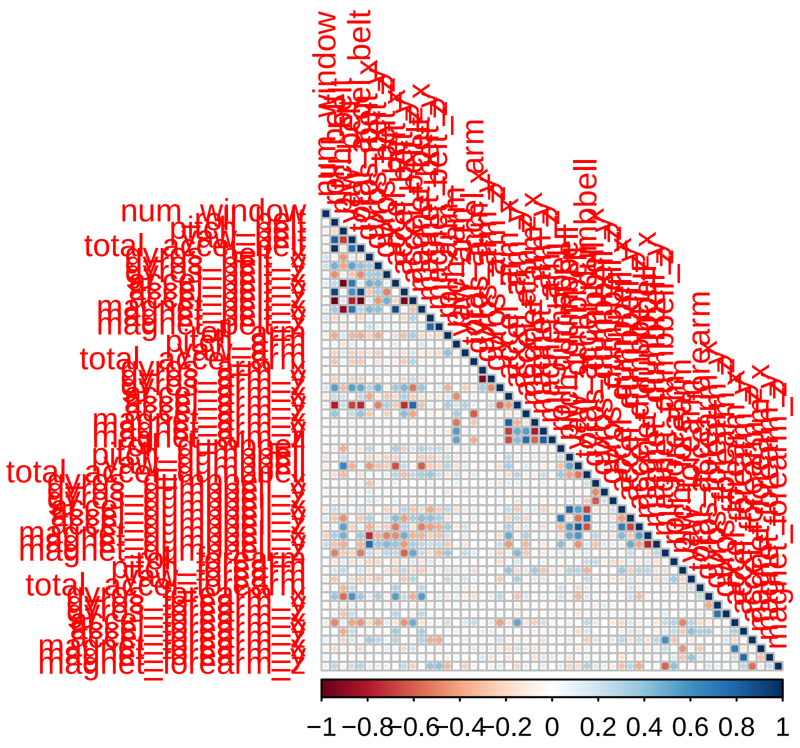
```
dim(test_set)
```

```
## [1] 3923   54
```

Perform Correlation Analysis

```
corr_matrix <- cor(training_set[, -54])
corrplot(corr_matrix, type="lower", t1.cex=0.6, t1.col = rgb(0,0,0))
```

```
## Warning in text.default(pos.xlabel[, 1], pos.xlabel[, 2], newcolnames, srt =
## tl.srt, : "t1.cex" is not a graphical parameter

## Warning in text.default(pos.xlabel[, 1], pos.xlabel[, 2], newcolnames, srt =
## tl.srt, : "t1.col" is not a graphical parameter

## Warning in text.default(pos.ylabel[, 1], pos.ylabel[, 2], newrownames, col =
## tl.col, : "t1.cex" is not a graphical parameter

## Warning in text.default(pos.ylabel[, 1], pos.ylabel[, 2], newrownames, col =
## tl.col, : "t1.col" is not a graphical parameter

## Warning in title(title, ...): "t1.cex" is not a graphical parameter

## Warning in title(title, ...): "t1.col" is not a graphical parameter
```



Positive correlated variables are displayed in blue color. While the negative correlation is displyed in Red. Color intensity indicates the correlation strength.

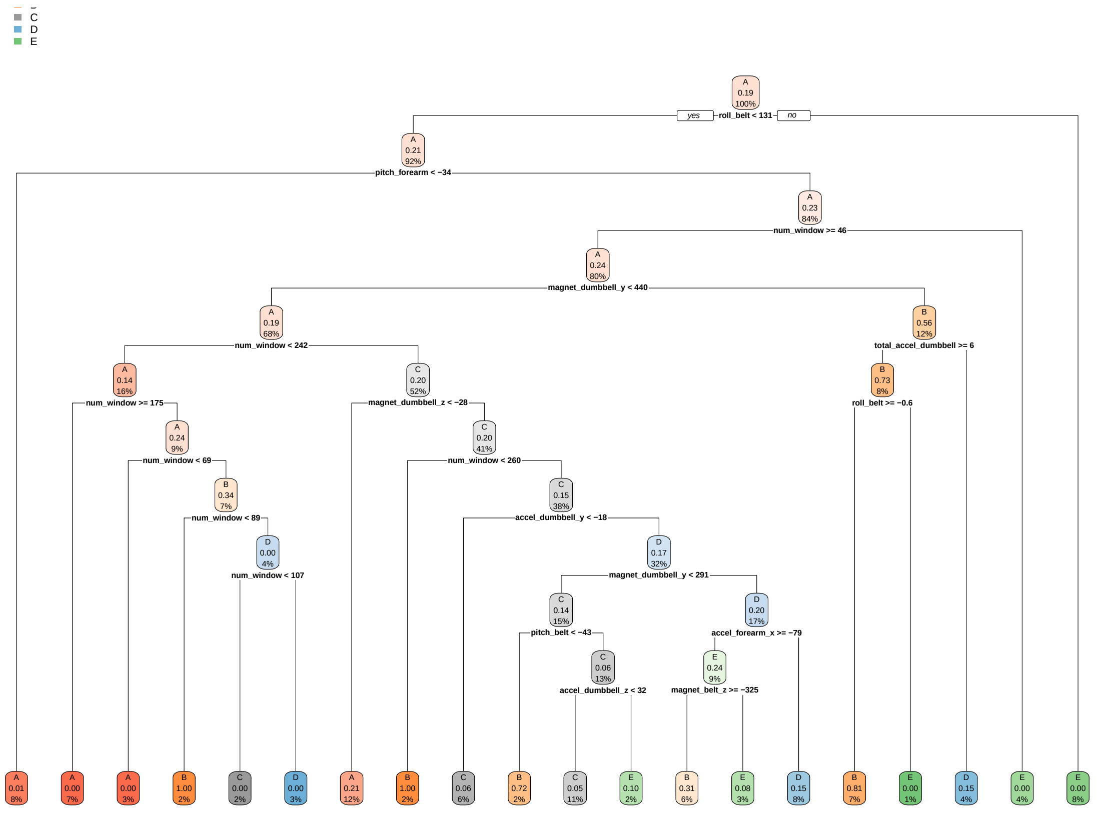
Prediction Models

1. Decision Trees

a. Build the model

```r
set.seed(2023)
model_decision_tree <- rpart(classe ~ ., data=training_set, method="class")
rpart.plot(model_decision_tree, extra=106)
```

```
## Warning: extra=106 but the response has 5 levels (only the 2nd level is
## displayed)
```



b. Make Prediction

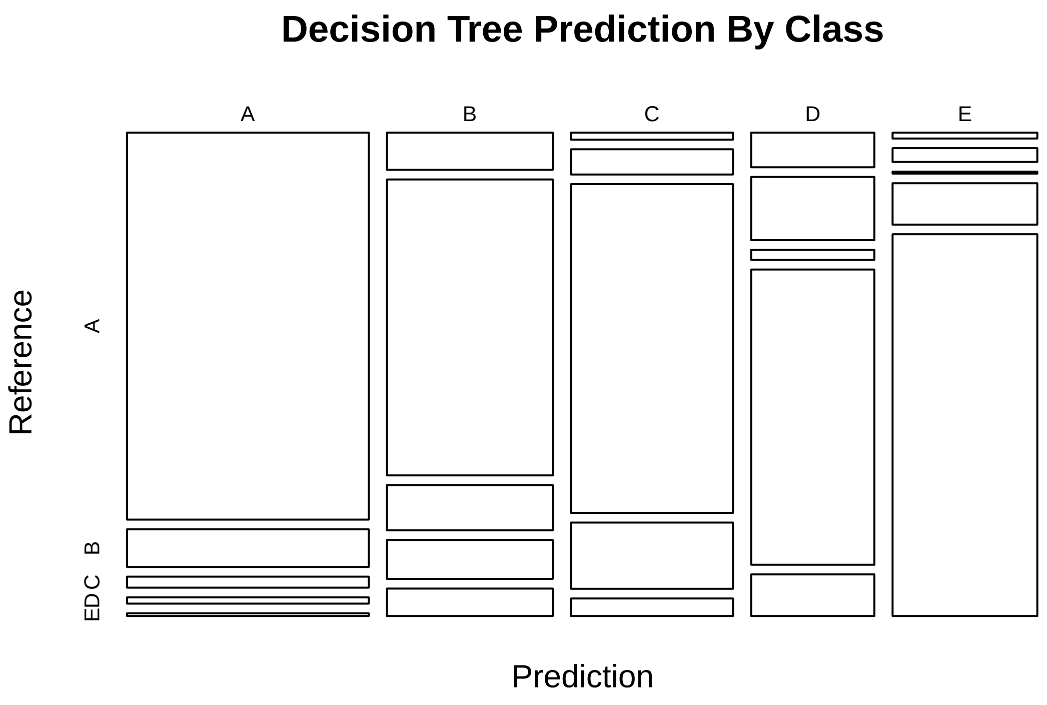
```r
predict_decision_tree <- predict(model_decision_tree, newdata = test_set, type="class")
conf_matrix_dt <- confusionMatrix(predict_decision_tree, as.factor(test_set$classe))
conf_matrix_dt
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 985   96   28   16    7
##          B  65 517  79  68  48
##          C  12  43 561 113  30
##          D  45  82  13 383  54
##          E   9  21   3  63 582
##
## Overall Statistics
##
##                Accuracy : 0.7719
##                  95% CI : (0.7584, 0.7849)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7112
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8826   0.6812   0.8202  0.59565   0.8072
## Specificity            0.9476   0.9178   0.9389  0.94085   0.9700
## Pos Pred Value         0.8701   0.6654   0.7391  0.66378   0.8584
## Neg Pred Value         0.9531   0.9231   0.9611  0.92230   0.9572
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2511   0.1318   0.1430  0.09763   0.1484
## Detection Prevalence   0.2886   0.1981   0.1935  0.14708   0.1728
## Balanced Accuracy      0.9151   0.7995   0.8795  0.76825   0.8886
```

The prediction accuracy of the decision tree model is 77.19

c. Plot the accuracy of the model by class

```
plot(conf_matrix_dt$table, col=conf_matrix_dt$byClass, main="Decision Tree Prediction By Class")
```



2. Random Forest Model

a. Build the model

```
model_random_forest <- randomForest(as.factor(classe)~ ., data=training_set, importance=TRUE, ntrees=20
```

b. Make Prediction

```
predict_random_forest <- predict(model_random_forest, newdata = test_set, type="class")
conf_matrix_rf <- confusionMatrix(predict_random_forest, as.factor(test_set$classe))
conf_matrix_rf
```

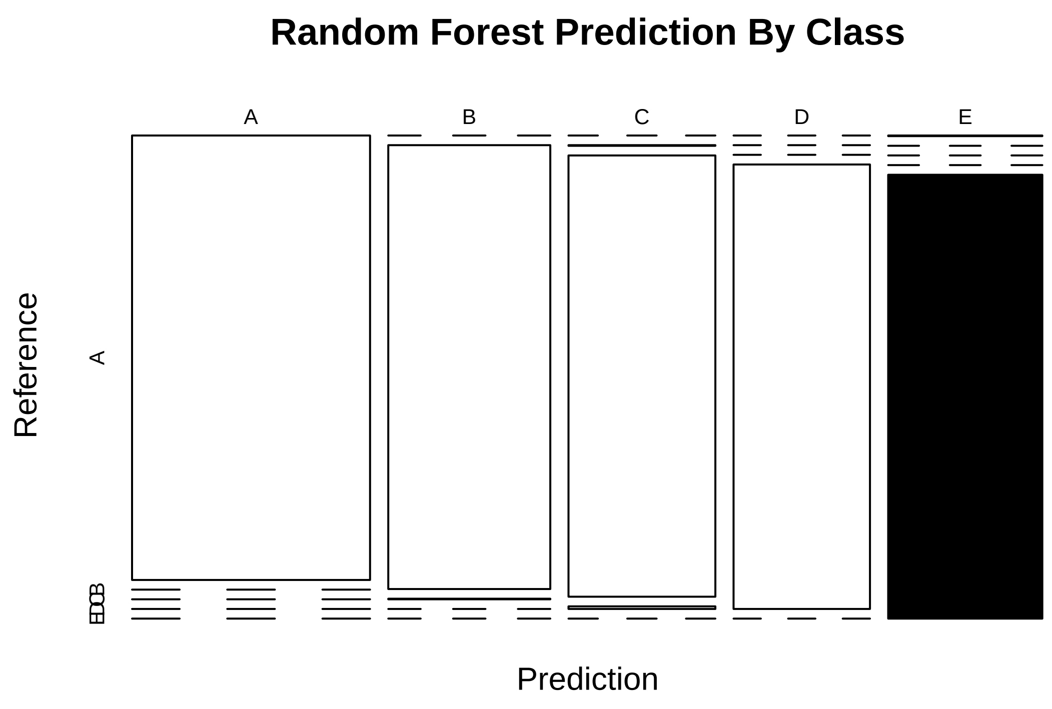
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    0    0    0    0
##          B    0  758    1    0    0
##          C    0    1  683    4    0
##          D    0    0    0  639    0
##          E    1    0    0    0  721
##
## Overall Statistics
##
```

```
##                  Accuracy : 0.9982
##                    95% CI : (0.9963, 0.9993)
##       No Information Rate : 0.2845
##       P-Value [Acc > NIR] : < 2.2e-16
##
##                     Kappa : 0.9977
##
##    Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9987   0.9985   0.9938   1.0000
## Specificity            1.0000   0.9997   0.9985   1.0000   0.9997
## Pos Pred Value         1.0000   0.9987   0.9927   1.0000   0.9986
## Neg Pred Value         0.9996   0.9997   0.9997   0.9988   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1932   0.1741   0.1629   0.1838
## Detection Prevalence   0.2842   0.1935   0.1754   0.1629   0.1840
## Balanced Accuracy      0.9996   0.9992   0.9985   0.9969   0.9998
```

The prediction accuracy of the decision tree model is 99.82

c. Plot the accuracy of the model by class

```
plot(conf_matrix_rf$table, col=conf_matrix_rf$byClass, main="Random Forest Prediction By Class")
```



Model Selection

Based on the prediction accuracy between the Decision Tree model at 77.19% and Random Forest model at 99.82%, Random Forest model is chosen to make predictions for the test_data set.

Predict Test Data using Random Forest Model

```r
predict_rft_test_data <- predict(model_random_forest, newdata = test_data, type="class")
predict_rft_test_data
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```