Prediction Assignment

AbdulAziz Ascandari

2024-09-21

Background

In recent years, the proliferation of wearable devices such as Jawbone Up, Nike FuelBand, and Fitbit has made it increasingly easy and affordable to collect extensive data on personal activities. These devices have become integral tools for the quantified self movement—a community of individuals who systematically track various aspects of their lives. The primary motivations for this practice include improving health, identifying behavioral patterns, and exploring technology. While there is a wealth of data available on the quantity of activities performed, there is a notable gap in quantifying the quality or effectiveness of these activities.

Aim

In this project, the objective is to utilize data collected from accelerometers placed on the belt, forearm, arm, and dumbbell of six participants. These participants were instructed to perform barbell lifts, both correctly and incorrectly, in five distinct ways. The analysis and modeling for this project will be conducted using the h2o library, a powerful tool for machine learning.The data for the analysis is taken from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Analysis

Preprocessing

Load Libraries

Here, we shall employ the h2o library for our prediction.

```
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit https://docs.h2o.ai
```

```
##
## ------------------------------------------------------------------------

##
## Attaching package: 'h2o'

## The following objects are masked from 'package:stats':
##
##     cor, sd, var

## The following objects are masked from 'package:base':
##
##     &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)
```

Load and clean data

```
set.seed(1234)
training <- read.csv("pml-training.csv")

# Remove near-zero variance features
nzv <- nearZeroVar(training)
training_clean <- training[, -nzv]

# Remove columns with more than 95% missing values
na_cols <- sapply(training_clean, function(x) mean(is.na(x))) > 0.95
train_clean <- training_clean[, !na_cols]

# Remove irrelevant columns (e.g., identifiers)
train_clean <- train_clean[, -c(1:7)]  # Adjust the range as needed
```

Data Splitting

```
# Initialize H2O cluster
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         23 minutes 42 seconds
##     H2O cluster timezone:       Africa/Casablanca
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.44.0.3
##     H2O cluster version age:    9 months
##     H2O cluster name:           H2O_started_from_R_biolab_pzv484
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   7.64 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.4.1 (2024-06-14)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is (9 months) old. There may be a newer version
available.
## Please download and install the latest version from: https://h2o-
release.s3.amazonaws.com/h2o/latest_stable.html
```

```
# Convert data to H2O frame
train_h2o <- as.h2o(train_clean)
```

```
##   |
|                                                                      |   0%
|
|======================================================================| 100%
```

```
# Split into train, validation, and test sets (70%, 15%, 15%)
splits <- h2o.splitFrame(data = train_h2o, ratios = c(0.7, 0.15), seed =
1234)
train <- h2o.assign(splits[[1]], "train")
valid <- h2o.assign(splits[[2]], "valid")
test  <- h2o.assign(splits[[3]], "test")
```

Model Training

Gradient Boosting

```
# Convert the response column 'classe' to a factor (categorical variable)
train$classe <- as.factor(train$classe)
valid$classe <- as.factor(valid$classe)

y <- "classe"
x <- setdiff(names(train), y)

# Train GBM model
gbm_model <- h2o.gbm(
  x = x, y = y,
  training_frame = train,
  validation_frame = valid,
  ntrees = 500, max_depth = 6, learn_rate = 0.1,
  seed = 1234
)
```

```
##   |
|                                                                      |   0%
|
|===                                                                   |   4%
|
|=======                                                               |  11%
|
|=============                                                         |  19%
|
|==================                                                    |  27%
|
|========================                                              |  36%
|
|==============================                                        |  45%
|
|====================================                                  |  54%
|
|==========================================                            |  63%
|
|================================================                      |  73%
|
|==========================================================            |  83%
|
|====================================================================  |  90%
|
|======================================================================| 100%
```

Random Forest

```
rf_model <- h2o.randomForest(
  x = x, y = y,
  training_frame = train,
```

```
  validation_frame = valid,
  ntrees = 500, max_depth = 6,
  seed = 1234
)
```

```
##   |
|                                                                      |   0%
|
|===                                                                   |   5%
|
|======                                                                |   9%
|
|==================                                                    |  25%
|
|========================================                              |  58%
|
|=================================================                     |  74%
|
|======================================================================| 100%
```

Naive Bayes

```
nb_model <- h2o.naiveBayes(
  x = x, y = y,
  training_frame = train,
  validation_frame = valid,
  nfolds = 10,
  seed = 1234
)
```

```
##   |
|                                                                      |   0%
|
|====================================                                  |  55%
|
|======================================================================| 100%
```

XGBoost

```
xgb_model <- h2o.xgboost(
  x = x, y = y,
  training_frame = train,
  validation_frame = valid,
  seed = 1234
)
```

```
##   |
|                                                                      |   0%
|
|=============                                                         |  18%
|
|========================================                              |  60%
```

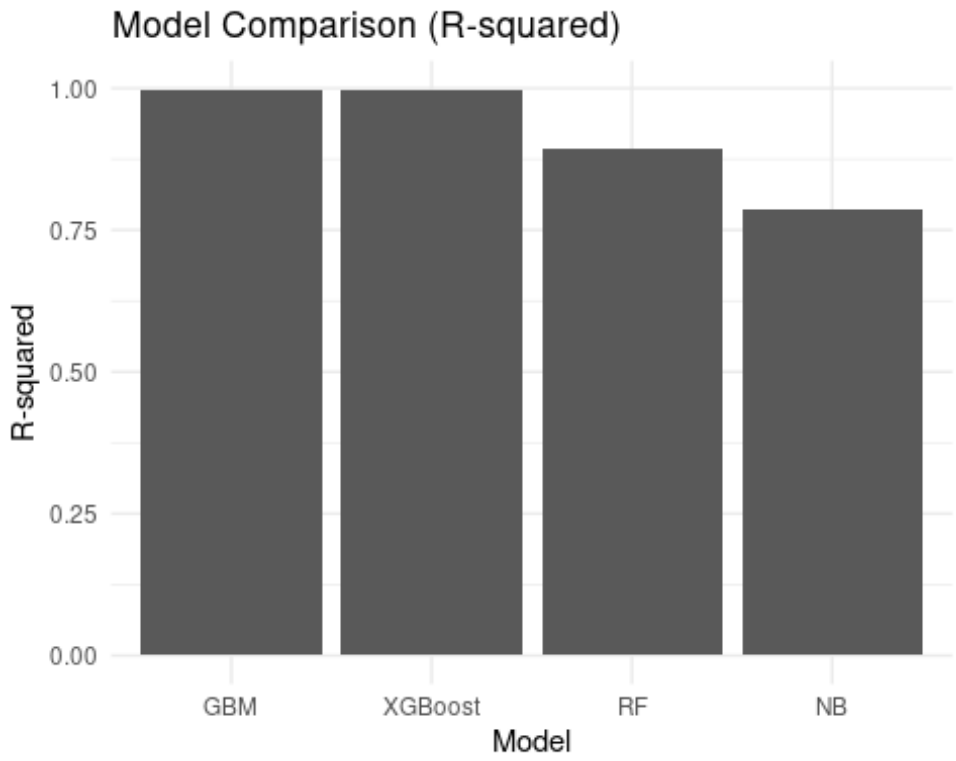
```
|
|======================================================================| 100%
```

Model Evaluation

Compare R-squared for Each Model

```
rsquare <- data.frame(
  Model = c("GBM", "RF", "NB", "XGBoost"),
  R2 = c(
    h2o.r2(gbm_model, valid = TRUE),
    h2o.r2(rf_model, valid = TRUE),
    h2o.r2(nb_model, valid = TRUE),
    h2o.r2(xgb_model, valid = TRUE)
  )
)

# Plot the R-squared values
ggplot(rsquare, aes(x = reorder(Model, -R2), y = R2)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Model Comparison (R-squared)", x = "Model", y = "R-squared")
```



From the R^2 plot the GBM model is selected. The confusion matrix is then printed.

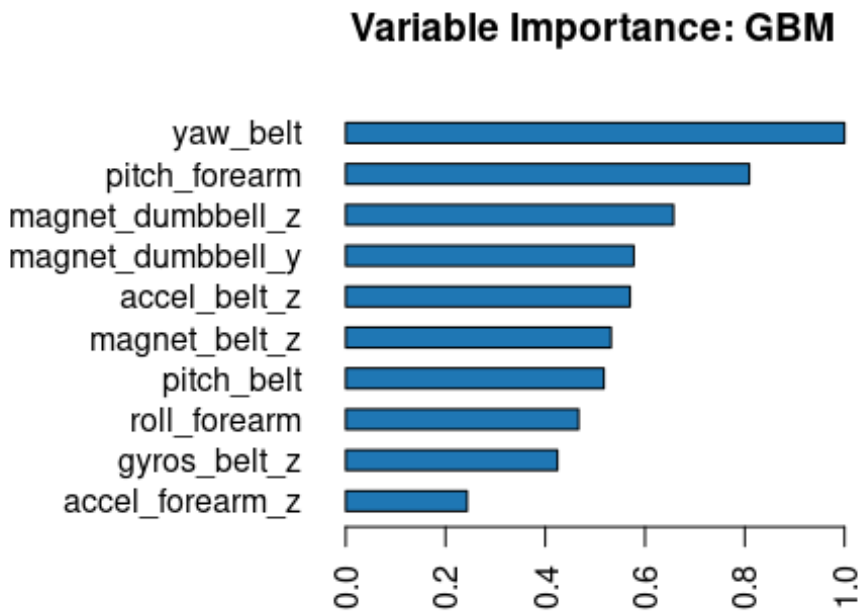
Confusion matrix of the selected Model (GBM)

```
h2o.confusionMatrix(gbm_model, valid = TRUE)
```

```
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##          A   B   C   D   E  Error         Rate
## A      816   1   0   0   0 0.0012 =    1 / 817
## B        1 574   0   0   0 0.0017 =    1 / 575
## C        0   2 527   1   0 0.0057 =    3 / 530
## D        0   0   6 469   0 0.0126 =    6 / 475
## E        0   0   0   2 541 0.0037 =    2 / 543
## Totals 817 577 533 472 541 0.0044 = 13 / 2,940
```

Variable Importance Plot (GBM)

```
h2o.varimp_plot(gbm_model)
```



Model Testing

Load Test Data

```
test_data <- read.csv("pml-testing.csv")
test_h2o <- as.h2o(test_data)
```

```
##   |
|                                                                      |   0%
|
|======================================================================| 100%
```

Predict Using the GBM Model

```
predictions <- h2o.predict(gbm_model, test_h2o)
```

```
##   |
|                                                                      |   0%
|
|======================================================================| 100%
```

```
predicted_classes <- as.data.frame(predictions$predict)
print(predicted_classes)
```

```
##    predict
## 1        B
## 2        A
## 3        B
## 4        A
## 5        A
## 6        E
## 7        D
## 8        B
## 9        A
## 10       A
## 11       B
## 12       C
## 13       B
## 14       A
## 15       E
## 16       E
## 17       A
## 18       B
## 19       B
## 20       B
```