XGBoost

Chieh-Ya Chang

2024-12-09

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.3.3
```

```
library(data.table)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x purrr::lift()        masks caret::lift()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x dplyr::slice()       masks xgboost::slice()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dplyr)
```

```r
ObesityDataSet <- read.csv("/Users/zhangjieya/Desktop/ObesityDataSet.csv")
```

```r
set.seed(1234)
ObesityDataSet <- ObesityDataSet[sample(1:nrow(ObesityDataSet)), ]
head(ObesityDataSet)
```

```
##      Gender      Age   Height    Weight family_history_with_overweight FAVC
## 1004 Female 28.83056 1.700000  78.00000                              yes  yes
## 623  Female 18.00674 1.700000  50.00000                               no  yes
## 934    Male 22.81466 1.716289  75.68843                              yes  yes
## 400    Male 21.00000 1.650000  60.00000                               no   no
## 1626   Male 25.34140 1.786997 115.02536                              yes  yes
## 1103   Male 17.89478 1.731389  84.06488                              yes   no
##          FCVC      NCP       CAEC SMOKE     CH2O SCC      FAF      TUE
## 1004 3.000000 3.000000  Sometimes    no 1.699971  no 1.727114 1.000000
## 623  1.003566 3.238258 Frequently    no 1.014634  no 0.783676 1.000000
## 934  2.000000 3.000000  Sometimes    no 2.000000  no 0.092344 1.466496
## 400  3.000000 1.000000 Frequently    no 1.000000  no 0.000000 0.000000
## 1626 1.999530 3.000000  Sometimes    no 2.111908  no 1.453042 0.849503
## 1103 2.019674 2.843319  Sometimes    no 2.832004  no 1.000000 0.608607
##            CALC                MTRANS          NObeyesdad
## 1004 Frequently            Automobile Overweight_Level_II
## 623   Sometimes Public_Transportation Insufficient_Weight
## 934   Sometimes Public_Transportation  Overweight_Level_I
## 400   Sometimes             Motorbike       Normal_Weight
## 1626  Sometimes Public_Transportation     Obesity_Type_II
## 1103  Sometimes Public_Transportation Overweight_Level_II
```

```r
#age+physical activity(FAF)+water intake(CH2O)+number of main meals(NCP)
#+tech device usage(TUE)+obesity levels(NObeyesdad)
ObesityLabels <- ObesityDataSet %>%
    select(Age, FAF, CH2O, NCP, FCVC, TUE, NObeyesdad)

head(ObesityLabels)
```

```
##           Age      FAF     CH2O      NCP     FCVC      TUE          NObeyesdad
## 1004 28.83056 1.727114 1.699971 3.000000 3.000000 1.000000 Overweight_Level_II
## 623  18.00674 0.783676 1.014634 3.238258 1.003566 1.000000 Insufficient_Weight
## 934  22.81466 0.092344 2.000000 3.000000 2.000000 1.466496  Overweight_Level_I
## 400  21.00000 0.000000 1.000000 1.000000 3.000000 0.000000       Normal_Weight
## 1626 25.34140 1.453042 2.111908 3.000000 1.999530 0.849503     Obesity_Type_II
## 1103 17.89478 1.000000 2.832004 2.843319 2.019674 0.608607 Overweight_Level_II
```

```r
ObesityLabels$NObeyesdad <- recode(ObesityLabels$NObeyesdad,
                                   "Insufficient_Weight" = 0,
                                   "Normal_Weight" = 1,
                                   "Obesity_Type_I" = 2,
                                   "Obesity_Type_II" = 3,
                                   "Obesity_Type_III" = 4,
```

```
                                    "Overweight_Level_I" = 5,
                                    "Overweight_Level_II" = 6)
```

```
set.seed(137)
#split into training (80%) and testing set (20%)
parts = createDataPartition(ObesityLabels$NObeyesdad, p = 0.8, list = F)
train = ObesityLabels[parts, ]
test = ObesityLabels[-parts, ]
#define predictor and response variables in training set
train_x = data.matrix(train[, -7])
train_y = train[,7]
#define predictor and response variables in testing set
test_x = data.matrix(test[, -7])
test_y = test[, 7]
#define final training and testing sets
xgb_train = xgb.DMatrix(data = train_x, label = train_y)
xgb_test = xgb.DMatrix(data = test_x, label = test_y)
```

```
#define watchlist
watchlist = list(train=xgb_train, test=xgb_test)
#fit XGBoost model and display training and testing data at each round
#The "multi:softmax" is used for multi-class classification tasks
#where the target variable has more than two distinct classes.
model = xgb.train(data = xgb_train, max.depth = 3, watchlist=watchlist,
                  nrounds = 100, eval_metric = "mlogloss",
                  objective = "multi:softmax", num_class = 7)
```

```
## [1]  train-mlogloss:1.665347 test-mlogloss:1.690933
## [2]  train-mlogloss:1.489499 test-mlogloss:1.537188
## [3]  train-mlogloss:1.367907 test-mlogloss:1.426980
## [4]  train-mlogloss:1.280257 test-mlogloss:1.345961
## [5]  train-mlogloss:1.206299 test-mlogloss:1.284411
## [6]  train-mlogloss:1.143488 test-mlogloss:1.227475
## [7]  train-mlogloss:1.092124 test-mlogloss:1.185465
## [8]  train-mlogloss:1.041956 test-mlogloss:1.142353
## [9]  train-mlogloss:1.004490 test-mlogloss:1.114215
## [10] train-mlogloss:0.975361 test-mlogloss:1.093270
## [11] train-mlogloss:0.945898 test-mlogloss:1.072937
## [12] train-mlogloss:0.922920 test-mlogloss:1.057684
## [13] train-mlogloss:0.897084 test-mlogloss:1.035609
## [14] train-mlogloss:0.876981 test-mlogloss:1.022480
## [15] train-mlogloss:0.856782 test-mlogloss:1.009545
## [16] train-mlogloss:0.835447 test-mlogloss:0.996334
## [17] train-mlogloss:0.813924 test-mlogloss:0.983367
## [18] train-mlogloss:0.796302 test-mlogloss:0.971150
## [19] train-mlogloss:0.783512 test-mlogloss:0.961437
## [20] train-mlogloss:0.766889 test-mlogloss:0.954776
## [21] train-mlogloss:0.750205 test-mlogloss:0.945715
## [22] train-mlogloss:0.730948 test-mlogloss:0.934351
## [23] train-mlogloss:0.715464 test-mlogloss:0.926285
## [24] train-mlogloss:0.704429 test-mlogloss:0.923072
## [25] train-mlogloss:0.695805 test-mlogloss:0.918690
## [26] train-mlogloss:0.682738 test-mlogloss:0.911293
```

```
## [27] train-mlogloss:0.670820 test-mlogloss:0.908633
## [28] train-mlogloss:0.658373 test-mlogloss:0.902630
## [29] train-mlogloss:0.647893 test-mlogloss:0.898429
## [30] train-mlogloss:0.636873 test-mlogloss:0.893033
## [31] train-mlogloss:0.628153 test-mlogloss:0.888558
## [32] train-mlogloss:0.619818 test-mlogloss:0.885966
## [33] train-mlogloss:0.606635 test-mlogloss:0.880672
## [34] train-mlogloss:0.596316 test-mlogloss:0.875823
## [35] train-mlogloss:0.591176 test-mlogloss:0.876426
## [36] train-mlogloss:0.583384 test-mlogloss:0.873177
## [37] train-mlogloss:0.574933 test-mlogloss:0.871310
## [38] train-mlogloss:0.564367 test-mlogloss:0.867530
## [39] train-mlogloss:0.555986 test-mlogloss:0.864618
## [40] train-mlogloss:0.548803 test-mlogloss:0.861932
## [41] train-mlogloss:0.543822 test-mlogloss:0.862330
## [42] train-mlogloss:0.535468 test-mlogloss:0.857533
## [43] train-mlogloss:0.531248 test-mlogloss:0.857892
## [44] train-mlogloss:0.523533 test-mlogloss:0.855201
## [45] train-mlogloss:0.518466 test-mlogloss:0.854177
## [46] train-mlogloss:0.511442 test-mlogloss:0.850321
## [47] train-mlogloss:0.507320 test-mlogloss:0.848064
## [48] train-mlogloss:0.501739 test-mlogloss:0.846373
## [49] train-mlogloss:0.497558 test-mlogloss:0.843316
## [50] train-mlogloss:0.492046 test-mlogloss:0.843149
## [51] train-mlogloss:0.485225 test-mlogloss:0.840307
## [52] train-mlogloss:0.480497 test-mlogloss:0.837923
## [53] train-mlogloss:0.475730 test-mlogloss:0.837597
## [54] train-mlogloss:0.468968 test-mlogloss:0.835617
## [55] train-mlogloss:0.464067 test-mlogloss:0.832246
## [56] train-mlogloss:0.458253 test-mlogloss:0.830628
## [57] train-mlogloss:0.453040 test-mlogloss:0.828206
## [58] train-mlogloss:0.447670 test-mlogloss:0.827008
## [59] train-mlogloss:0.442150 test-mlogloss:0.825174
## [60] train-mlogloss:0.438862 test-mlogloss:0.824110
## [61] train-mlogloss:0.433749 test-mlogloss:0.821263
## [62] train-mlogloss:0.428994 test-mlogloss:0.818238
## [63] train-mlogloss:0.425699 test-mlogloss:0.815999
## [64] train-mlogloss:0.421798 test-mlogloss:0.814982
## [65] train-mlogloss:0.416581 test-mlogloss:0.813426
## [66] train-mlogloss:0.411576 test-mlogloss:0.811490
## [67] train-mlogloss:0.406722 test-mlogloss:0.809867
## [68] train-mlogloss:0.401199 test-mlogloss:0.806375
## [69] train-mlogloss:0.396466 test-mlogloss:0.801973
## [70] train-mlogloss:0.393781 test-mlogloss:0.802140
## [71] train-mlogloss:0.390852 test-mlogloss:0.802062
## [72] train-mlogloss:0.386825 test-mlogloss:0.799627
## [73] train-mlogloss:0.382869 test-mlogloss:0.797021
## [74] train-mlogloss:0.378509 test-mlogloss:0.796101
## [75] train-mlogloss:0.375974 test-mlogloss:0.794913
## [76] train-mlogloss:0.372190 test-mlogloss:0.792890
## [77] train-mlogloss:0.369074 test-mlogloss:0.792231
## [78] train-mlogloss:0.366214 test-mlogloss:0.792040
## [79] train-mlogloss:0.363219 test-mlogloss:0.792970
## [80] train-mlogloss:0.359828 test-mlogloss:0.792864
```

```
## [81] train-mlogloss:0.356691 test-mlogloss:0.793277
## [82] train-mlogloss:0.353371 test-mlogloss:0.792480
## [83] train-mlogloss:0.350763 test-mlogloss:0.791379
## [84] train-mlogloss:0.347446 test-mlogloss:0.791421
## [85] train-mlogloss:0.344465 test-mlogloss:0.792768
## [86] train-mlogloss:0.341495 test-mlogloss:0.792124
## [87] train-mlogloss:0.338092 test-mlogloss:0.790203
## [88] train-mlogloss:0.335271 test-mlogloss:0.787985
## [89] train-mlogloss:0.331528 test-mlogloss:0.786627
## [90] train-mlogloss:0.328746 test-mlogloss:0.787883
## [91] train-mlogloss:0.326709 test-mlogloss:0.786891
## [92] train-mlogloss:0.324539 test-mlogloss:0.785423
## [93] train-mlogloss:0.321200 test-mlogloss:0.785469
## [94] train-mlogloss:0.319219 test-mlogloss:0.783506
## [95] train-mlogloss:0.317211 test-mlogloss:0.784847
## [96] train-mlogloss:0.314609 test-mlogloss:0.783016
## [97] train-mlogloss:0.311238 test-mlogloss:0.783247
## [98] train-mlogloss:0.308483 test-mlogloss:0.782323
## [99] train-mlogloss:0.306627 test-mlogloss:0.781059
## [100]    train-mlogloss:0.304224 test-mlogloss:0.781695
```

```
#define final model
final = xgboost(data = xgb_train, max.depth = 3, nrounds = 99, objective = "multi:softmax",
                num_class = 7, verbose = 0)
```

```
preds <- predict(model, test_x)
confusionMatrixResult <- confusionMatrix(factor(preds), factor(test_y))
print(confusionMatrixResult)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2  3  4  5  6
##          0 36  2  5  1  0  6  2
##          1  6 46  5  2  2  9 10
##          2  0  2 47  1  0  7  4
##          3  1  1  5 53  0  2  5
##          4  0  0  0  0 59  1  0
##          5  7  3  1  0  0 30  5
##          6  3  4  7  2  0  6 32
##
## Overall Statistics
##
##                Accuracy : 0.7214
##                  95% CI : (0.6759, 0.7638)
##     No Information Rate : 0.1667
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.675
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.67925   0.7931   0.6714   0.8983   0.9672  0.49180
## Specificity           0.95640   0.9061   0.9600   0.9612   0.9972  0.95543
## Pos Pred Value        0.69231   0.5750   0.7705   0.7910   0.9833  0.65217
## Neg Pred Value        0.95380   0.9647   0.9359   0.9830   0.9944  0.91711
## Prevalence            0.12619   0.1381   0.1667   0.1405   0.1452  0.14524
## Detection Rate        0.08571   0.1095   0.1119   0.1262   0.1405  0.07143
## Detection Prevalence  0.12381   0.1905   0.1452   0.1595   0.1429  0.10952
## Balanced Accuracy     0.81782   0.8496   0.8157   0.9298   0.9822  0.72362
##                      Class: 6
## Sensitivity           0.55172
## Specificity           0.93923
## Pos Pred Value        0.59259
## Neg Pred Value        0.92896
## Prevalence            0.13810
## Detection Rate        0.07619
## Detection Prevalence  0.12857
## Balanced Accuracy     0.74548
```

```
sensitivity_values <- confusionMatrixResult$byClass[, "Sensitivity"]
class_names <- c("Insufficient Weight", "Normal Weight", "Obesity Type I",
                 "Obesity Type II", "Obesity Type III",
                 "Overweight Level I", "Overweight Level II")
sensitivity_table <- data.frame(Class = class_names,
                                Sensitivity = sensitivity_values)
print(sensitivity_table)
```

```
##                        Class Sensitivity
## Class: 0 Insufficient Weight   0.6792453
## Class: 1       Normal Weight   0.7931034
## Class: 2      Obesity Type I   0.6714286
## Class: 3     Obesity Type II   0.8983051
## Class: 4    Obesity Type III   0.9672131
## Class: 5  Overweight Level I   0.4918033
## Class: 6 Overweight Level II   0.5517241
```

```
predictions <- predict(final, xgb_test)
accuracy <- mean(predictions == test_y)
print(paste("Test Accuracy:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Test Accuracy: 71.9 %"
```

```
importance_matrix <- xgb.importance(model = final)
print(importance_matrix)
```

```
##    Feature      Gain     Cover Frequency
## 1:     Age 0.2391232 0.3398285 0.3012709
## 2:    FCVC 0.2195113 0.1285323 0.1181161
## 3:     TUE 0.1698511 0.1104574 0.1395465
## 4:    CH2O 0.1316473 0.1392817 0.1582357
## 5:     NCP 0.1205779 0.1429750 0.1230999
## 6:     FAF 0.1192891 0.1389250 0.1597309
```

```r
xgb.plot.importance(importance_matrix, main = "Feature Importance")
```