Data-Science.R

Himanshu Singh

2021-03-19

install.packages("xgboost")

## Installing package into 'C:/Users/Himanshu Singh/Documents/R/win-library/4.0'  
## (as 'lib' is unspecified)

## package 'xgboost' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'xgboost'

## Warning in file.copy(savedcopy, lib, recursive = TRUE):  
## problem copying C:\Users\Himanshu Singh\Documents\R\win-  
## library\4.0\00LOCK\xgboost\libs\x64\xgboost.dll to C:\Users\Himanshu  
## Singh\Documents\R\win-library\4.0\xgboost\libs\x64\xgboost.dll: Permission  
## denied

## Warning: restored 'xgboost'

##   
## The downloaded binary packages are in  
## C:\Users\Himanshu Singh\AppData\Local\Temp\RtmpUtceFi\downloaded\_packages

install.packages("tidyverse")

## Installing package into 'C:/Users/Himanshu Singh/Documents/R/win-library/4.0'  
## (as 'lib' is unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\Himanshu Singh\AppData\Local\Temp\RtmpUtceFi\downloaded\_packages

install.packages("DiagrammeR")

## Installing package into 'C:/Users/Himanshu Singh/Documents/R/win-library/4.0'  
## (as 'lib' is unspecified)

## package 'DiagrammeR' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\Himanshu Singh\AppData\Local\Temp\RtmpUtceFi\downloaded\_packages

# libraries we're going to use  
library(xgboost) # for xgboost

## Warning: package 'xgboost' was built under R version 4.0.4

library(tidyverse) # general utility functions

## Warning: package 'tidyverse' was built under R version 4.0.4

## -- Attaching packages ------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2 v purrr 0.3.4  
## v tibble 3.0.1 v dplyr 0.8.5  
## v tidyr 1.0.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 4.0.2

## -- Conflicts ---------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::slice() masks xgboost::slice()

library(DiagrammeR)

## Warning: package 'DiagrammeR' was built under R version 4.0.4

# read in our data & put it in a data frame  
diseaseInfo <- read\_csv("C:/Users/Himanshu Singh/Desktop/Outbreak\_240817.csv")

## Parsed with column specification:  
## cols(  
## .default = col\_character(),  
## Id = col\_double(),  
## latitude = col\_double(),  
## longitude = col\_double(),  
## sumAtRisk = col\_double(),  
## sumCases = col\_double(),  
## sumDeaths = col\_double(),  
## sumDestroyed = col\_double(),  
## sumSlaughtered = col\_double(),  
## humansAge = col\_double(),  
## humansAffected = col\_double(),  
## humansDeaths = col\_double()  
## )

## See spec(...) for full column specifications.

# set a random seed & shuffle data frame  
set.seed(1234)  
diseaseInfo <- diseaseInfo[sample(1:nrow(diseaseInfo)), ]  
  
# print the first few rows of our dataframe  
head(diseaseInfo)

## # A tibble: 6 x 24  
## Id source latitude longitude region country admin1 localityName  
## <dbl> <chr> <dbl> <dbl> <chr> <chr> <chr> <chr>   
## 1 219318 Natio~ 28.4 46.0 Asia Saudi ~ Easte~ Hafr-Elbatin  
## 2 219097 OIE 45.6 11.4 Europe Italy Veneto CASTELGOMBE~  
## 3 219828 OIE 52.4 23.2 Europe Poland Podla~ Adamowo   
## 4 221042 OIE 36.6 10.7 Africa Tunisia Nabeul Beni khalled  
## 5 217753 OIE 46.1 4.43 Europe France Rhone~ POULE LES E~  
## 6 228469 OIE 35.3 129. Asia Republ~ Kyong~ Gijang-gun   
## # ... with 16 more variables: localityQuality <chr>, observationDate <chr>,  
## # reportingDate <chr>, status <chr>, disease <chr>, serotypes <chr>,  
## # speciesDescription <chr>, sumAtRisk <dbl>, sumCases <dbl>, sumDeaths <dbl>,  
## # sumDestroyed <dbl>, sumSlaughtered <dbl>, humansGenderDesc <chr>,  
## # humansAge <dbl>, humansAffected <dbl>, humansDeaths <dbl>

# get the subset of the dataframe that doesn't have labels about   
# humans affected by the disease  
diseaseInfo\_humansRemoved <- diseaseInfo %>%  
 select(-starts\_with("human"))  
  
# get a boolean vector of training labels  
diseaseLabels <- diseaseInfo %>%  
 select(humansAffected) %>% # get the column with the # of humans affected  
 is.na() %>% # is it NA?  
 magrittr::not() # switch TRUE and FALSE (using function from the magrittr package)  
  
# check out the first few lines  
head(diseaseLabels) # of our target variable

## humansAffected  
## [1,] TRUE  
## [2,] FALSE  
## [3,] FALSE  
## [4,] FALSE  
## [5,] FALSE  
## [6,] FALSE

head(diseaseInfo$humansAffected) # of the original column

## [1] 1 NA NA NA NA NA

# select just the numeric columns  
diseaseInfo\_numeric <- diseaseInfo\_humansRemoved %>%  
 select(-Id) %>% # the case id shouldn't contain useful information  
 select(-c(longitude, latitude)) %>% # location data is also in country data  
 select\_if(is.numeric) # select remaining numeric columns  
  
# make sure that our dataframe is all numeric  
str(diseaseInfo\_numeric)

## tibble [17,008 x 5] (S3: tbl\_df/tbl/data.frame)  
## $ sumAtRisk : num [1:17008] NA 53 NA 61 93 12 103 49 13 NA ...  
## $ sumCases : num [1:17008] NA 4 1 1 1 NA 1 9 10 1 ...  
## $ sumDeaths : num [1:17008] NA 0 1 0 0 6 NA 0 10 0 ...  
## $ sumDestroyed : num [1:17008] NA 0 0 0 0 6 NA 0 3 1 ...  
## $ sumSlaughtered: num [1:17008] NA 0 0 0 0 NA NA 0 0 0 ...

# check out the first few rows of the country column  
head(diseaseInfo$country)

## [1] "Saudi Arabia" "Italy" "Poland"   
## [4] "Tunisia" "France" "Republic of Korea"

# one-hot matrix for just the first few rows of the "country" column  
model.matrix(~country-1,head(diseaseInfo))

## countryFrance countryItaly countryPoland countryRepublic of Korea  
## 1 0 0 0 0  
## 2 0 1 0 0  
## 3 0 0 1 0  
## 4 0 0 0 0  
## 5 1 0 0 0  
## 6 0 0 0 1  
## countrySaudi Arabia countryTunisia  
## 1 1 0  
## 2 0 0  
## 3 0 0  
## 4 0 1  
## 5 0 0  
## 6 0 0  
## attr(,"assign")  
## [1] 1 1 1 1 1 1  
## attr(,"contrasts")  
## attr(,"contrasts")$country  
## [1] "contr.treatment"

# convert categorical factor into one-hot encoding  
region <- model.matrix(~country-1,diseaseInfo)  
  
# some of the species  
head(diseaseInfo$speciesDescription)

## [1] NA   
## [2] "domestic, cattle"   
## [3] "wild, wild boar"   
## [4] "domestic, cattle, domestic, goat, domestic, sheep"  
## [5] "domestic, cattle"   
## [6] "domestic, unspecified bird"

# add a boolean column to our numeric dataframe indicating whether a species is domestic  
diseaseInfo\_numeric$is\_domestic <- str\_detect(diseaseInfo$speciesDescription, "domestic")  
  
# get a list of all the species by getting the last  
speciesList <- diseaseInfo$speciesDescription %>%  
 str\_replace("[[:punct:]]", "") %>% # remove punctuation (some rows have parentheses)  
 str\_extract("[a-z]\*$") # extract the least word in each row  
  
# convert our list into a dataframe...  
speciesList <- tibble(species = speciesList)  
  
# and convert to a matrix using 1 hot encoding  
options(na.action='na.pass') # don't drop NA values!  
species <- model.matrix(~species-1,speciesList)  
  
# add our one-hot encoded variable and convert the dataframe into a matrix  
diseaseInfo\_numeric <- cbind(diseaseInfo\_numeric, region, species)  
diseaseInfo\_matrix <- data.matrix(diseaseInfo\_numeric)  
  
# get the numb 70/30 training test split  
numberOfTrainingSamples <- round(length(diseaseLabels) \* .7)  
  
# training data  
train\_data <- diseaseInfo\_matrix[1:numberOfTrainingSamples,]  
train\_labels <- diseaseLabels[1:numberOfTrainingSamples]  
  
# testing data  
test\_data <- diseaseInfo\_matrix[-(1:numberOfTrainingSamples),]  
test\_labels <- diseaseLabels[-(1:numberOfTrainingSamples)]  
  
# put our testing & training data into two seperates Dmatrixs objects  
dtrain <- xgb.DMatrix(data = train\_data, label= train\_labels)  
dtest <- xgb.DMatrix(data = test\_data, label= test\_labels)  
  
# train a model using our training data  
model <- xgboost(data = dtrain, # the data   
 nround = 2, # max number of boosting iterations  
 objective = "binary:logistic") # the objective function

## [01:04:35] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.  
## [1] train-logloss:0.448429   
## [2] train-logloss:0.313402

# generate predictions for our held-out testing data  
pred <- predict(model, dtest)  
  
# get & print the classification error  
err <- mean(as.numeric(pred > 0.5) != test\_labels)  
print(paste("test-error=", err))

## [1] "test-error= 0.0139161113288906"

# train an xgboost model  
model\_tuned <- xgboost(data = dtrain, # the data   
 max.depth = 3, # the maximum depth of each decision tree  
 nround = 2, # max number of boosting iterations  
 objective = "binary:logistic") # the objective function

## [01:04:35] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.  
## [1] train-logloss:0.448429   
## [2] train-logloss:0.313402

# generate predictions for our held-out testing data  
pred <- predict(model\_tuned, dtest)  
  
# get & print the classification error  
err <- mean(as.numeric(pred > 0.5) != test\_labels)  
print(paste("test-error=", err))

## [1] "test-error= 0.0139161113288906"

# get the number of negative & positive cases in our data  
negative\_cases <- sum(train\_labels == FALSE)  
postive\_cases <- sum(train\_labels == TRUE)  
  
# train a model using our training data  
model\_tuned <- xgboost(data = dtrain, # the data   
 max.depth = 3, # the maximum depth of each decision tree  
 nround = 10, # number of boosting rounds  
 early\_stopping\_rounds = 3, # if we dont see an improvement in this many rounds, stop  
 objective = "binary:logistic", # the objective function  
 scale\_pos\_weight = negative\_cases/postive\_cases) # control for imbalanced classes

## [01:04:35] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.  
## [1] train-logloss:0.446740   
## Will train until train\_logloss hasn't improved in 3 rounds.  
##   
## [2] train-logloss:0.311939   
## [3] train-logloss:0.228258   
## [4] train-logloss:0.173351   
## [5] train-logloss:0.136223   
## [6] train-logloss:0.110756   
## [7] train-logloss:0.093108   
## [8] train-logloss:0.080866   
## [9] train-logloss:0.072260   
## [10] train-logloss:0.066269

# generate predictions for our held-out testing data  
pred <- predict(model\_tuned, dtest)  
  
# get & print the classification error  
err <- mean(as.numeric(pred > 0.5) != test\_labels)  
print(paste("test-error=", err))

## [1] "test-error= 0.0139161113288906"

# train a model using our training data  
model\_tuned <- xgboost(data = dtrain, # the data   
 max.depth = 3, # the maximum depth of each decision tree  
 nround = 10, # number of boosting rounds  
 early\_stopping\_rounds = 3, # if we dont see an improvement in this many rounds, stop  
 objective = "binary:logistic", # the objective function  
 scale\_pos\_weight = negative\_cases/postive\_cases, # control for imbalanced classes  
 gamma = 1) # add a regularization term

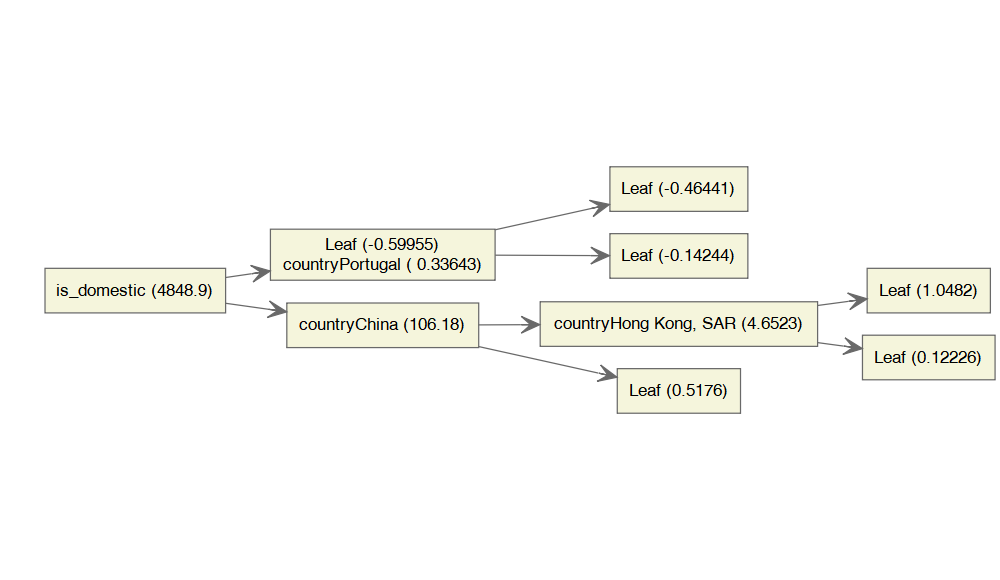
## [01:04:35] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.  
## [1] train-logloss:0.446740   
## Will train until train\_logloss hasn't improved in 3 rounds.  
##   
## [2] train-logloss:0.311927   
## [3] train-logloss:0.228251   
## [4] train-logloss:0.173355   
## [5] train-logloss:0.136235   
## [6] train-logloss:0.110763   
## [7] train-logloss:0.093123   
## [8] train-logloss:0.080873   
## [9] train-logloss:0.072273   
## [10] train-logloss:0.066278

# generate predictions for our held-out testing data  
pred <- predict(model\_tuned, dtest)  
  
# get & print the classification error  
err <- mean(as.numeric(pred > 0.5) != test\_labels)  
print(paste("test-error=", err))

## [1] "test-error= 0.0139161113288906"

# plot them features! what's contributing most to our model?  
xgb.plot.multi.trees(feature\_names = names(diseaseInfo\_matrix),   
 model = model)

## Column 2 ['No'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.



# convert log odds to probability  
odds\_to\_probs <- function(odds){  
 return(exp(odds)/ (1 + exp(odds)))  
}  
  
  
# probability of leaf above countryPortugul  
odds\_to\_probs(-0.599)

## [1] 0.3545725

# get information on how important each feature is  
importance\_matrix <- xgb.importance(names(diseaseInfo\_matrix), model = model)  
  
# and plot it!  
xgb.plot.importance(importance\_matrix)

