yantry2

yam rozen

2023-06-17

```
library(caret)
```

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

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

```
library(lightgbm)
```

```
## Loading required package: R6
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
library(Matrix)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::slice()  masks lightgbm::slice()
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(stringr)
library(tictoc)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following object is masked from 'package:tictoc':
##
##     shift
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(word2vec)
library(foreach)
```

```
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

clean text function

```
text.clean = function(x)
{
  x  =  gsub("<.*?>", " ", x)
  x  =  iconv(x, "latin1", "ASCII", sub="")
  x  =  gsub("[^[:alnum:]]", " ", x)
  x  =  tolower(x)
  x  =  stripWhitespace(x)
  x  =  gsub("^\\s+|\\s+$", "", x)
  return(x)
}
```

train pre process

```
setwd("C:/Users/yamro/OneDrive/desktop/limudim/ds_app/HW6")
df_train1 <- read_csv("data/df_train1.csv")
```

```
## New names:
## Rows: 14000 Columns: 46
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (34): title, brand, style, heel_type, heel_height, width, shoe_width, ma... dbl
## (12): ...1, Unnamed: 0, id, price, n_sold, n_watchers, free_shipping, lo...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
x_train <- df_train1 %>% select(-c("brand", "price", "location"))
x_train <- x_train %>% mutate(id = ...1) %>% select(-c("...1", "Unnamed: 0"))
y_train <- df_train1 %>% select(price)
y_train <- log(y_train)
x_train$new_country <- text.clean(x_train$new_country)
x_train$new_brand <- text.clean(x_train$new_brand)
x_train$category <- text.clean(x_train$category)
x_train$style <- text.clean(x_train$style)
x_train$condition <- text.clean(x_train$condition)

#Preprocessing
# let's drop features with more than 5000 missing values
na_counts <- colSums(is.na(x_train))
column_to_remove <- which(na_counts > 5000)
x_train <- x_train[, -column_to_remove]

x_train <- x_train %>% mutate_if(is.numeric, ~replace_na(.,0))
x_train <- x_train %>% mutate_if(is.character, ~replace_na(.,"Unknown"))


# Count the occurrences of each element in the column
counts <- table(x_train$style)
# Get the elements that appear less than 5 times
less_than_5 <- names(counts[counts < 50])
# Replace elements with "less than 5 appearances"
x_train$style[x_train$style %in% less_than_5] <- "less than 50 appearances"

# Count the occurrences of each element in the column
counts_brand <- table(x_train$new_brand)
# Get the elements that appear less than 10 times
less_than_10_brand <- names(counts_brand[counts_brand < 15])
# Replace elements with "less than 10 appearances"
x_train$new_brand[x_train$new_brand %in% less_than_10_brand] <- "less than 15 appearances"
```

```
# factorize numeric vars
x_train$free_shipping <- as.factor(x_train$free_shipping)
x_train$longtime_member <- as.factor(x_train$longtime_member)
x_train$same_day_shipping <- as.factor(x_train$same_day_shipping)
x_train$fast_safe_shipping <- as.factor(x_train$fast_safe_shipping)
x_train$returns <- as.factor(x_train$returns)
x_train$feedback <- as.factor(x_train$feedback)

# factorize some categorial vars
x_train$style <- as.factor(x_train$style)
x_train$condition <- as.factor(x_train$condition)
x_train$category <- as.factor(x_train$category)
x_train$new_brand <- as.factor(x_train$new_brand)
x_train$new_country <- as.factor(x_train$new_country)

all_data_after_processing <- cbind(x_train, y_train)
all_data_after_processing$price <- NULL

log_price <- y_train$price
```

test pre process

```
df_test1 <- read_csv("data/df_test1.csv")
```

```
## New names:
## Rows: 1044 Columns: 45
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (34): title, brand, style, heel_type, heel_height, width, shoe_width, ma... dbl
## (11): ...1, Unnamed: 0, id, n_sold, n_watchers, free_shipping, longtime_...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
x_test <- df_test1 %>% select(-c("brand", "location"))
x_test <- x_test %>% mutate(id = ...1) %>% select(-c("...1", "Unnamed: 0"))

#Preprocessing
# let's drop features with more than 5000 missing values
x_test <- x_test[, -column_to_remove]
x_test$new_country <- text.clean(x_test$new_country)
x_test$new_brand <- text.clean(x_test$new_brand)
x_test$category <- text.clean(x_test$category)
x_test$style <- text.clean(x_test$style)
x_test$condition <- text.clean(x_test$condition)



x_test <- x_test %>% mutate_if(is.numeric, ~replace_na(.,0))
x_test <- x_test %>% mutate_if(is.character, ~replace_na(.,"Unknown"))

# we are using the list of styles 'less_than_5' from the training
```

```
x_test$style[x_test$style %in% less_than_5] <- "less than 50 appearances"

x_test$new_brand[x_test$new_brand %in% less_than_10_brand] <- "less than 15 appearances"
# factorize numeric vars
x_test$free_shipping <- as.factor(x_test$free_shipping)
x_test$longtime_member <- as.factor(x_test$longtime_member)
x_test$same_day_shipping <- as.factor(x_test$same_day_shipping)
x_test$fast_safe_shipping <- as.factor(x_test$fast_safe_shipping)
x_test$returns <- as.factor(x_test$returns)
x_test$feedback <- as.factor(x_test$feedback)

# factorize some categorial vars
x_test$style <- as.factor(x_test$style)
x_test$condition <- as.factor(x_test$condition)
x_test$category <- as.factor(x_test$category)
x_test$new_brand <- as.factor(x_test$new_brand)
x_test$new_country <- as.factor(x_test$new_country)
all_data_after_processing_test <- x_test
all_data_after_processing_test$price <- NULL
```

combine both

```
all <- bind_rows(all_data_after_processing, all_data_after_processing_test)

all <- all %>%select(-title)
```

prepere for modeling

modeling

```
tic("oof glm preds train")

model_glm = glmnet(x = sparse_train, y = log_price, alpha = 0, lambda = 10^(-1.5))
res_glm_test = predict(model_glm, sparse_test)
set.seed(13)
folds = createFolds(log_price, k=5)
oof_glm = foreach(i = 1:5, .combine = rbind) %do% {
model_glm = glmnet(x = sparse_train[-folds[[i]],], y = log_price[-folds[[i]]], alpha = 0, lambda = 10^(
res_glm = predict(model_glm, sparse_train[folds[[i]],])
data.frame(i = i, res_glm = res_glm)
}
names(oof_glm) <- c("i", "pr_glm")
toc()
```

```
## oof glm preds train: 0.29 sec elapsed
```

modellgb

```
tic("add glm preds to train-set and overall glm pred to test-set")
sparse_train = cbind(rbind(sparse_train[folds[[1]],],sparse_train[folds[[2]],],
                           sparse_train[folds[[3]],],sparse_train[folds[[4]],],sparse_train[folds[[5]],]
log_price = log_price[c(folds[[1]],folds[[2]],folds[[3]], folds[[4]], folds[[5]])]
```

```
sparse_test = cbind(sparse_test, glm_pred = res_glm_test[,1])

tic("lgb model create")
dtrain = lgb.Dataset(sparse_train, label = log_price)
dtest = lgb.Dataset(sparse_test,reference = dtrain)
toc()
```

```
## lgb model create: 0 sec elapsed
```

```
tic("model creating")
params <- list(
  objective = "regression"
  , metric = "rmse"
  , learning_rate = 0.1
)
model <- lgb.train(
  params = params
  , data = dtrain
  , nrounds = 10000

)
```

```
## [LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
## [LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.003793 se
## You can set ‘force_row_wise=true‘ to remove the overhead.
## And if memory is not enough, you can set ‘force_col_wise=true‘.
## [LightGBM] [Info] Total Bins 599
## [LightGBM] [Info] Number of data points in the train set: 14000, number of used features: 173
## [LightGBM] [Info] Start training from score 3.534938
```

```
toc()
```

```
## model creating: 46.44 sec elapsed
```

```
tic("Lgb Prediction")
res_lgb <- predict(model, sparse_test)

toc()
```

```
## Lgb Prediction: 0.28 sec elapsed
```