THE_ICONIC_Interview_Assignment_PART_2_DS

Nikita sharma

22/05/2019

Import libs/dependencies

```
library(jsonlite)
library(caret)
```

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

```
## Loading required package: 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(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(Matrix)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(DataExplorer)
```

Game on!

Let's have a look at the data.

```
data <- fromJSON('data.json')
nrow(data)
```

```
## [1] 46279
```

```
length(unique(data$customer_id))
```

```
## [1] 46030
```

```
head(data)
```

	customer_id <chr>	days_since_first_order <int>	days_since_last_order <int>
1	64f7d7dd7a59bba7168cc9c960a5c60e	2091	15672
2	fa7c64efd5c037ff2abcce571f9c1712	2082	528
3	18923c9361f27583d2320951435e4888	2072	144
4	aa21f31def4edbdcead818afcdfc4d32	2054	49200
5	668c6aac52ff54d4828ad379cdb38e7d	2053	49272
6	5f1e0812c41be939d971e34236d4de5b	2044	49056

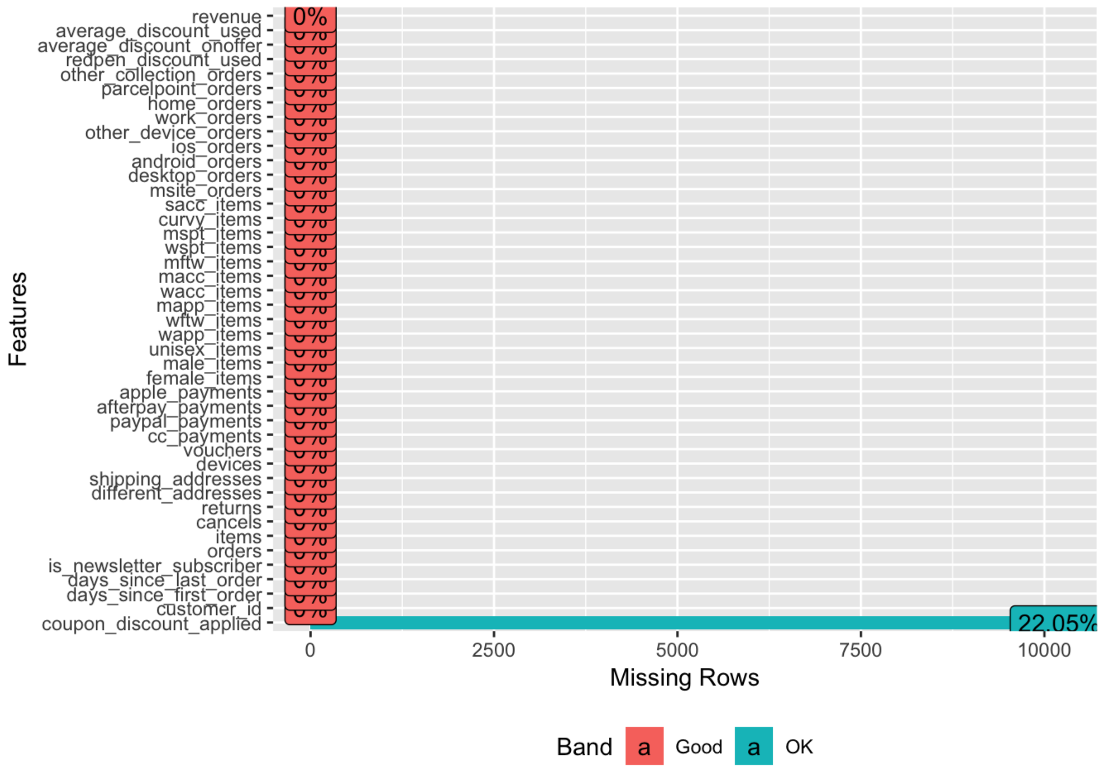
6 rows | 1-4 of 44 columns

We see that we have few duplicate customers in the dataset. We will ignore these for this activity.

Data Cleaning

Let's see if missing values are present in our dataset. We will attempt to find and fix the corrupt column in the dataset. Here we will also clean our data, convert categorical variables to numeric using label encoding, normalizing columns etc.

```
plot_missing(data)
```



```
# days_since_last_order : is a factor of 24
data$days_since_last_order <- data$days_since_last_order / min(data$days_since_last_order)

# redpen_discount_used
data$redpen_discount_used <- scale(data$redpen_discount_used)
# revenue
data$revenue <- scale(data$revenue)
#coupon_discount_applied
data$coupon_discount_applied <- scale(data$coupon_discount_applied)
# average_discount_used
data$average_discount_used <- scale(data$average_discount_used)


# Characters to Factors for label encoding
data$is_newsletter_subscriber <- factor(data$is_newsletter_subscriber)

# Impute Zero to all missing value of numeric columns
data[sapply(data, is.numeric)] <- lapply(data[sapply(data, is.numeric)],
                                          function(x) ifelse(is.na(x),0, x))
```

We can see that days_since_last_order is a factor of 24, so will divide the variable by 24. We can also see that coupon_discount_applied has 22.05% missing values, we will fill missing values with 0.

Heuristic Approach

As we don't have target variable here, we will generate our target variable based on following assumptions :

1. If a customer brought a male product more than 3 times than female product, we will assign the customer as Male i.e; 0.
2. If a customer brought a female product more than 3 times than male product, we will assign the customer as Female i.e; 1.

The model can be made stricter by making the threshold = 4 instead of 3.

```
sub_data <- data

# Hueristics:
# Labeling customer as Female
# if they bought 3 times more Female product
# than the Male products. (Make 4 time for more strict model)
# Vice versa for Male as well.
sub_data$gender <- -1
sub_data$gender <- ifelse(round(sub_data$female_items/(1 + sub_data$male_items)) >= 3,
                          1, sub_data$gender)
sub_data$gender <- ifelse(round(sub_data$male_items/(1+sub_data$female_items)) >= 3,
                          0, sub_data$gender)

nrow(sub_data)
```

```
## [1] 46279
```

```
nrow(sub_data[sub_data$gender == -1, ])
```

```
## [1] 26352
```

```
nrow(sub_data[sub_data$gender == 1, ])
```

```
## [1] 15609
```

```
nrow(sub_data[sub_data$gender == 0, ])
```

```
## [1] 4318
```

```
train_data_raw <-  sub_data %>%  filter(gender == 1 | gender == 0)
head(train_data_raw)
```

	customer_id <chr>	days_since_first_order <int>	days_since_last_order <dbl>
1	64f7d7dd7a59bba7168cc9c960a5c60e	2091	653
2	fa7c64efd5c037ff2abcce571f9c1712	2082	22
3	18923c9361f27583d2320951435e4888	2072	6
4	10bc5715ecec307441b3b4f378e58592	2030	364
5	040fb9742f9e14cf69c7a748bdf20137	2033	56
6	1936d9aa913e7a12e24af56c096419ca	2074	1009

6 rows | 1-4 of 45 columns

Test/Train split

We split our dataset into train-test in 80-20 ratio

```
smp_size <- floor(0.80 * nrow(train_data_raw))
set.seed(123)
train_ind <- sample(seq_len(nrow(train_data_raw)), size = smp_size)
train <- train_data_raw[train_ind, ]
test <- train_data_raw[-train_ind, ]

nrow(train)
```

```
## [1] 15941
```

```
nrow(test)
```

```
## [1] 3986
```

XG Boost

We are using Xgboost Classification model as our target variable is a class type.

```r
target <- train$gender
dtrain <- sparse.model.matrix( gender ~ . -customer_id, data = train)[,-1]
dtest <- sparse.model.matrix( gender ~ . -customer_id, data = test)[,-1]


best_param <- list()
best_seednumber <- 1234
best_rmse <- Inf
best_rmse_index <- 0


set.seed(123)
for (iter in 1:20) {
  param <- list(objective = "binary:logistic",
                eval_metric = "error",
                max_depth = sample(6:10, 1),
                eta = runif(1, .01, .3),
                gamma = runif(1, 0.0, 0.2),
                subsample = runif(1, .6, .9),
                colsample_bytree = runif(1, .5, .8),
                min_child_weight = sample(1:40, 1),
                max_delta_step = sample(1:10, 1)
  )

  cv.nround <-  20  # <<<<<
  cv.nfold <-  5 # 5-fold cross-validation
  seed.number  <-  sample.int(10000, 1) # set seed for the cv

  set.seed(seed.number)
  mdcv <- xgb.cv(data = dtrain, params = param, label=target,
                 nfold = cv.nfold, nrounds = cv.nround,
                 verbose = F, early_stopping_rounds = 8, maximize = FALSE)

  min_error_index  <-  which.min(mdcv$evaluation_log[, test_error_mean])
  min_error <-  min(mdcv$evaluation_log[, test_error_mean])


  if (min_error < best_rmse) {
    print('best error:')
    print(min_error)

    best_rmse <- min_error
    best_rmse_index <- min_error_index
    best_seednumber <- seed.number
    best_param <- param
  }
}
```

```
## [1] "best error:"
## [1] 6.28e-05
```

```r
# The best index (min_rmse_index) is the best "nround" in the model
nround = best_rmse_index
set.seed(best_seednumber)
xg_mod <- xgboost(data = dtrain,  label=target, params = best_param, nround = nround, verbose = F)

# Calculate RMSE
# predicting on test dataset
pred <- predict(xg_mod, newdata = dtest)

out_test <- select(test, customer_id, gender)
out_test$predicted_gender_raw <- pred
out_test$predicted_gender <- round(pred)

# ERROR
nrow(out_test[out_test$predicted_gender != out_test$gender, ]) / nrow(out_test)
```

```
## [1] 0
```

Feature Importance

```
importance <- xgb.importance(feature_names = colnames(dtrain), model = xg_mod)
print(head(importance, 20))
```

```
##                  Feature         Gain        Cover  Frequency
##  1:         female_items 6.483596e-01 2.763061e-01 0.20224719
##  2:           male_items 2.686706e-01 5.553471e-01 0.34831461
##  3:           wftw_items 3.977709e-02 2.100155e-02 0.03370787
##  4:           wapp_items 2.340289e-02 3.416040e-02 0.08988764
##  5:           wacc_items 9.141440e-03 2.878702e-02 0.03370787
##  6:                items 5.840918e-03 1.925574e-02 0.06741573
##  7:               orders 1.607097e-03 9.287848e-03 0.04494382
##  8:           wspt_items 1.012960e-03 7.727342e-03 0.02247191
##  9:           mapp_items 7.565410e-04 1.029485e-03 0.03370787
## 10:              returns 5.295590e-04 1.373076e-02 0.03370787
## 11:              revenue 3.902712e-04 4.874578e-03 0.04494382
## 12:           mftw_items 3.105365e-04 1.427485e-02 0.01123596
## 13:         unisex_items 1.102006e-04 3.355048e-04 0.01123596
## 14: redpen_discount_used 8.041287e-05 9.491674e-05 0.01123596
## 15:           mspt_items 9.834995e-06 1.378684e-02 0.01123596
```

Full dataset prediction

```
dtest_full <- sparse.model.matrix( ~ . -customer_id, data = data)[,-1]
full_pred <- select(data, customer_id)
full_pred$gender <- round(predict(xg_mod, newdata = dtest_full))
full_pred$gender <- ifelse(full_pred$gender == 1, "F", "M")
head(full_pred)
```

	customer_id <chr>	**gender** <chr>
1	64f7d7dd7a59bba7168cc9c960a5c60e	F
2	fa7c64efd5c037ff2abcce571f9c1712	F
3	18923c9361f27583d2320951435e4888	F
4	aa21f31def4edbdcead818afcdfc4d32	M
5	668c6aac52ff54d4828ad379cdb38e7d	M
6	5f1e0812c41be939d971e34236d4de5b	M

6 rows