Renew and Return Rate Analysis

Iman Mousavi

2023-10-07

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
contracts = pd.read_excel('Data contracts.xlsx')
```

Exploring Data

First, let's take a look a the head of data:

```
print(contracts.head())
```

```
   shop_id  package_order_id package_name contract_date  start_date  \
0    35120             57868        Car C    2021-08-22  2021-09-06
1    73135             55723        Car B    2021-06-08  2021-06-08
2    28746             49014        Car C    2021-01-16  2021-01-16
3    76180             63743        Car C    2022-08-09         NaN
4    63157             46291        Car B    2020-11-23  2020-11-23

     end_date real_end_date  Listing_limit industry category          region  \
0  2022-10-06           NaT             80  re_auto
1  2022-07-15           NaT             50  re_auto
2  2021-05-17           NaT            500  re_auto
3  2023-02-08           NaT             10  re_auto
4  2021-03-23           NaT             10  re_auto
```

```
    city
0
1
2
3
4
```

It would be wise to change farsi columns into English, but becasue of time limit, I ignore this step.

Data Types

```
print(contracts.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14419 entries, 0 to 14418
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   shop_id           14419 non-null  int64
 1   package_order_id  14419 non-null  int64
 2   package_name      14419 non-null  object
 3   contract_date     14417 non-null  object
 4   start_date        14418 non-null  object
 5   end_date          14419 non-null  object
 6   real_end_date     8 non-null      datetime64[ns]
 7   Listing_limit     14419 non-null  int64
 8   industry          14419 non-null  object
 9   category          14419 non-null  object
 10  region            14416 non-null  object
 11  city              14419 non-null  object
dtypes: datetime64[ns](1), int64(3), object(8)
memory usage: 1.3+ MB
None
```

`contract_date`, `start_date`, and `end_date` should be transformed into datatime objects.

NOTE: There are missing values in columns contract_date, start_date, and region.

```
contracts['contract_date'] = pd.to_datetime(contracts['contract_date'])

contracts['start_date'] = pd.to_datetime(contracts['start_date'])

contracts['end_date'] = pd.to_datetime(contracts['end_date'])
```

Checking date columns again:

```
print(contracts.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14419 entries, 0 to 14418
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   shop_id           14419 non-null  int64
 1   package_order_id  14419 non-null  int64
 2   package_name      14419 non-null  object
 3   contract_date     14417 non-null  datetime64[ns]
 4   start_date        14418 non-null  datetime64[ns]
 5   end_date          14419 non-null  datetime64[ns]
 6   real_end_date     8 non-null      datetime64[ns]
 7   Listing_limit     14419 non-null  int64
 8   industry          14419 non-null  object
 9   category          14419 non-null  object
 10  region            14416 non-null  object
 11  city              14419 non-null  object
dtypes: datetime64[ns](4), int64(3), object(5)
memory usage: 1.3+ MB
None
```

We have also categorical column like industry, category, region, and city. In case of necessity, those columns will be defined as categories later.

Unique values of each column

```
print(contracts.nunique())
```

```
shop_id              8123
package_order_id    14417
package_name            9
contract_date        1186
start_date           1130
end_date             1549
real_end_date           7
Listing_limit          48
industry                2
category               11
region                 29
city                  282
dtype: int64
```

There are 8123 unqiue values of `shop_id`. Also, we have rows with the same `package_order_id`, which doesn't make a sense.

```
print(contracts[contracts.duplicated(subset='package_order_id', keep=False)].\
    sort_values('package_order_id'))
```

```
       shop_id  package_order_id package_name contract_date start_date  \
9828     14546              5660    General C    2019-07-20 2019-07-20
11579    14546              5660    General C    2019-07-20 2019-07-20
280      74876             60314        Car B    2022-01-08 2022-01-08
11651    74876             60314        Car B    2022-02-07 2022-01-08

        end_date real_end_date  Listing_limit industry           category  \
9828  2019-10-21           NaT              5  re_auto
11579 2019-10-21           NaT              5  general
280   2022-07-09           NaT             10  re_auto
11651 2022-07-09           NaT             10  general

         region   city
9828
11579
280
11651
```

According to `industry` and `category` columns, these two orders are different from each other; however, their `shop_id`, `package_name`, `start_date` and `end_date` and even their `city` and `region` are identical!

There must be a mistake at data entry pipeline. Due to high uncertainty, all four rows are discarded from the following analysis.

```
contracts.drop_duplicates(subset='package_order_id', keep=False, inplace=True)
```

Missing Values

```
missing_conditions = contracts['contract_date'].isna() | contracts['start_date'].isna() |
print(contracts[missing_conditions])
```

```
       shop_id  package_order_id package_name contract_date start_date  \
3        76180             63743        Car C    2022-08-09        NaT
10600    71562             53139    General A    2021-03-13 2021-03-13
11147    68825             48980    General B    2021-01-13 2021-01-13
12884    68392             48821    General C    2021-01-06 2021-01-06
12955    76557             64920    General C           NaT 2022-10-26
13769    76554             64903    General A           NaT 2022-10-26

        end_date real_end_date  Listing_limit industry  \
3     2023-02-08           NaT             10  re_auto
10600 2021-09-26           NaT             15  general
11147 2021-04-14           NaT              5  general
12884 2021-05-07           NaT             30  general
12955 2023-01-24           NaT              5  general
13769 2023-10-25           NaT              5  general

                      category           region    city
3
10600                             NaN
11147                                NaN
12884                                   NaN
12955
13769
```

NaT for date columns and NaN for other types are both standard ways of missingness indications.

Month column

```
contracts['month'] = contracts['end_date'].dt.month
contracts['month'] = np.where(contracts['real_end_date'].notnull(), contracts['real_end_da
```

Months found in this data set:

```
print(contracts['month'].unique())
```

```
[10.  7.  5.  2.  3.  4.  8.  1. 11. 12.  6.  9.]
```

Let's do the calculation for the last three months of year (i.e. October, November, and December).

Renew and Return Rate Calculation

First, a new column represents any future `start_date` for a new contract:

```
contracts_sorted = contracts.sort_values(by = ['shop_id', 'start_date'])
contracts_sorted['start_date_next'] = contracts_sorted.groupby('shop_id')['start_date'].sh
```

Then, days between `end_date` of previous contract and `start_date` of new one is determined:

```
contracts_sorted['days_to_new'] = contracts_sorted['start_date_next'] - contracts_sorted['

# If real_end_date exists:
contracts_sorted['days_to_new'] = np.where(contracts_sorted['real_end_date'].notnull(), co

contracts_sorted['days_to_new'] = contracts_sorted['days_to_new'].dt.days
```

A new column represents whether the conditions of renewal or return have been met:

```
# Renew
contracts_sorted['renew'] = contracts_sorted['days_to_new'] <= 30
# Return
contracts_sorted['return'] = contracts_sorted['days_to_new'] > 30
```

Let's filter only those rows that its `end_date` is within the last three months of year:

```
condition_months = contracts_sorted['month'].isin([10, 11, 12])
contracts_endseason = contracts_sorted[condition_months]
```

Last but not least, our interested rate are calculated:

```
renew_return_rates = contracts_endseason.groupby(['region', 'category', 'month'])['renew',
renew_return_rates = renew_return_rates.reset_index()
```

```
/var/folders/0s/5nvdy8kx70q7tvgqp_3xvyx00000gn/T/ipykernel_5150/4062478544.py:1: FutureWarni

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use
```

Renaming columns:

```
renew_return_rates.rename(columns = {'renew':'renew_rate_perc', 'return':'return_rate_perc
```

The dataframe is like the following:

```
print(renew_return_rates.head())
```

```
           region           category  month  renew_rate_perc  return_rate_perc
0                              10.0             40.0              20.0
1                              11.0            100.0               0.0
2                              12.0             20.0              10.0
3                          11.0            100.0               0.0
4                          12.0             50.0               0.0
```