Data acquisition from Kaggle

Important Note: You must sign up for the competition here and download your kaggle.json from your Kaggle account page. See Steps 1-2 here for more information.

In [ ]:
```
from google.colab import files

# UPLOAD YOUR KAGGLE.JSON
# Only run this cell if you need to upload kaggle.json
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[ ]:
```
{'kaggle.json': b'{"username":"catapultic","key":"bc709cc2cfed23022adc91952ba357
c7"}'}
```

In [ ]:
```
# Kaggle credentials setup
!pip install kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages
(1.5.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packag
es (from kaggle) (1.15.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-p
ackages (from kaggle) (5.0.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (f
rom kaggle) (4.62.3)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages
(from kaggle) (2021.5.30)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
s (from kaggle) (2.23.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-
packages (from kaggle) (2.8.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/d
ist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
kages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
t-packages (from requests->kaggle) (3.0.4)
```

In [ ]:
```
# Download Coupon Purchase Prediction data set
!kaggle competitions download -c coupon-purchase-prediction -p data
```

```
Warning: Looks like you're using an outdated API Version, please consider updati
ng (server 1.5.12 / client 1.5.4)
Downloading prefecture_locations.csv to data
  0% 0.00/2.00k [00:00<?, ?B/s]
100% 2.00k/2.00k [00:00<00:00, 3.63MB/s]
Downloading coupon_detail_train.csv.zip to data
 68% 5.00M/7.32M [00:00<00:00, 29.3MB/s]
```

```
100% 7.32M/7.32M [00:00<00:00, 35.9MB/s]
Downloading sample_submission.csv.zip to data
  0% 0.00/400k [00:00<?, ?B/s]
100% 400k/400k [00:00<00:00, 123MB/s]
Downloading coupon_area_test.csv.zip to data
  0% 0.00/14.0k [00:00<?, ?B/s]
100% 14.0k/14.0k [00:00<00:00, 12.4MB/s]
Downloading coupon_list_train.csv.zip to data
  0% 0.00/656k [00:00<?, ?B/s]
100% 656k/656k [00:00<00:00, 92.9MB/s]
Downloading documentation.zip to data
  0% 0.00/21.6k [00:00<?, ?B/s]
100% 21.6k/21.6k [00:00<00:00, 21.9MB/s]
Downloading user_list.csv.zip to data
  0% 0.00/627k [00:00<?, ?B/s]
100% 627k/627k [00:00<00:00, 88.8MB/s]
Downloading coupon_visit_train.csv.zip to data
 77% 65.0M/84.5M [00:00<00:00, 54.5MB/s]
100% 84.5M/84.5M [00:00<00:00, 133MB/s]
Downloading coupon_area_train.csv.zip to data
  0% 0.00/832k [00:00<?, ?B/s]
100% 832k/832k [00:00<00:00, 119MB/s]
Downloading coupon_list_test.csv.zip to data
  0% 0.00/11.6k [00:00<?, ?B/s]
100% 11.6k/11.6k [00:00<00:00, 11.5MB/s]
```

In [ ]:

```
# unzip and reorganize the zipped tables
# Master list of users
!unzip data/user_list.csv.zip -d data/

# Master list of coupons (train & test)
!unzip data/coupon_list_train.csv.zip -d data/
!unzip data/coupon_list_test.csv.zip -d data/

# Table containing physical areas where coupons are available (train & test)
!unzip data/coupon_area_train.csv.zip -d data/
!unzip data/coupon_area_test.csv.zip -d data/

# Purchase log of users buying coupons during the training period (train only)
!unzip data/coupon_detail_train.csv.zip -d data/

# Browsing log of users visiting coupons during the training period (train only)
!unzip data/coupon_visit_train.csv.zip -d data/
```

```
Archive:  data/user_list.csv.zip
  inflating: data/user_list.csv
Archive:  data/coupon_list_train.csv.zip
  inflating: data/coupon_list_train.csv
Archive:  data/coupon_list_test.csv.zip
  inflating: data/coupon_list_test.csv
Archive:  data/coupon_area_train.csv.zip
  inflating: data/coupon_area_train.csv
Archive:  data/coupon_area_test.csv.zip
  inflating: data/coupon_area_test.csv
Archive:  data/coupon_detail_train.csv.zip
  inflating: data/coupon_detail_train.csv
Archive:  data/coupon_visit_train.csv.zip
  inflating: data/coupon_visit_train.csv
```

In [ ]:

```
# Delete unused zip files
!rm -f data/*.zip
```

Translation of Japanese columns to English

Note: This does a full translation of the Japanese characters to English. It does not transliterate the Japanese place names to their English counterparts. We end up with the actual meaning of the Japanese names sometimes, like "Place which is by the water." That is okay - it is not important for training, they just help us explore the data and understand what we are looking at.

In [ ]:

```
# dependencies
%%capture
!pip install git+https://github.com/neuml/txtai#egg=txtai[pipeline]
!pip install pykakasi

# imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from txtai.pipeline import Translation
import pykakasi

translate = Translation()
kks = pykakasi.kakasi()
```

Translation helper functions

In [ ]:

```
# Lookup table of translations to save time
translations = {}

# Translates jp->en using txtai package (unless NaN)
def safe_translate(jp, transliterate=False):
  if pd.isna(jp) == False:
    if transliterate == True: # use pykakasi
      return ''.join([item['hepburn'].capitalize() for item in kks.convert(jp)])
    else:
      return translate(jp, 'en') # using txtai
  else:
    return jp

# Checks the translation dict first before translating
def lookup_or_translate(jp):
  if (jp not in translations):
    translations[jp] = safe_translate(jp) # pass transliterate=True to use kakas
  return translations[jp]

# Translates an entire column/list of data
def translate_list(data):
  translated = []
  for word in tqdm(data):
    t = lookup_or_translate(word)
    translated.append(t)
  return translated
```

In [ ]:

```
# Main loading function - takes a csv path, columns to translate,
# and returns a Pandas dataframe. Translates columns in-place.
def load_translate(csv_path, translate_columns=[]):
  df = pd.read_csv(csv_path)
  for c in translate_columns:
    df[c] = translate_list(df[c])
  return df
```

In [ ]:

```
# Create lists of columns that need to be translated for each table
# Coupon Visit Training set does not require any translation

# User list table
user_cols = ['PREF_NAME']

# Coupon list train and test
c_list_cols = ['CAPSULE_TEXT', 'GENRE_NAME', 'large_area_name',
               'ken_name', 'small_area_name']

# Coupon detail
c_detail_cols = ['SMALL_AREA_NAME']

# Coupon area train and test
c_area_cols = ['SMALL_AREA_NAME', 'PREF_NAME']

# Prefecture locations
c_pref_cols = ['PREF_NAME', 'PREFECTUAL_OFFICE']
```

In [ ]:

```
# Perform the translations and load the data into DataFrames
df_users = load_translate('data/user_list.csv', user_cols)
df_area_train = load_translate('data/coupon_area_train.csv', c_area_cols)
df_area_test = load_translate('data/coupon_area_test.csv', c_area_cols)
df_c_list_train = load_translate('data/coupon_list_train.csv', c_list_cols)
df_c_list_test  = load_translate('data/coupon_list_test.csv', c_list_cols)
df_c_detail_train = load_translate('data/coupon_detail_train.csv', c_detail_cols
df_visit_train = load_translate('data/coupon_visit_train.csv')
df_locations = load_translate('data/prefecture_locations.csv', c_pref_cols)
```

```
100%|██████████| 22873/22873 [00:13<00:00, 1753.43it/s]
100%|██████████| 138185/138185 [00:13<00:00, 10442.66it/s]
100%|██████████| 138185/138185 [00:00<00:00, 1506073.77it/s]
100%|██████████| 2165/2165 [00:00<00:00, 1499697.47it/s]
100%|██████████| 2165/2165 [00:00<00:00, 974216.09it/s]
100%|██████████| 19413/19413 [00:01<00:00, 13570.95it/s]
100%|██████████| 19413/19413 [00:00<00:00, 104246.77it/s]
100%|██████████| 19413/19413 [00:02<00:00, 8221.92it/s]
100%|██████████| 19413/19413 [00:00<00:00, 1045063.39it/s]
100%|██████████| 19413/19413 [00:00<00:00, 1357587.47it/s]
100%|██████████| 310/310 [00:00<00:00, 684694.18it/s]
100%|██████████| 310/310 [00:00<00:00, 664741.43it/s]
100%|██████████| 310/310 [00:00<00:00, 677205.33it/s]
100%|██████████| 310/310 [00:00<00:00, 582020.70it/s]
100%|██████████| 310/310 [00:00<00:00, 410039.18it/s]
100%|██████████| 168996/168996 [00:00<00:00, 1475148.38it/s]
100%|██████████| 47/47 [00:00<00:00, 205132.45it/s]
100%|██████████| 47/47 [00:09<00:00,  4.99it/s]
```

In [ ]:

```
# Save CSV files to translated output.
!mkdir data_translated
dir = 'data_translated'

df_users.to_csv(f'{dir}/user_list.csv')
```

```
df_area_test.to_csv(f'{dir}/coupon_area_test.csv')
df_area_train.to_csv(f'{dir}/coupon_area_train.csv')
df_c_detail_train.to_csv(f'{dir}/coupon_detail_train.csv')
df_c_list_test.to_csv(f'{dir}/coupon_list_test.csv')
df_c_list_train.to_csv(f'{dir}/coupon_list_train.csv')
df_visit_train.to_csv(f'{dir}/coupon_visit_train.csv')
df_locations.to_csv(f'{dir}/prefecture_locations.csv')

!zip -r translated_data.zip data_translated/
```

```
adding: data_translated/ (stored 0%)
adding: data_translated/coupon_visit_train.csv (deflated 77%)
adding: data_translated/prefecture_locations.csv (deflated 51%)
adding: data_translated/coupon_area_test.csv (deflated 89%)
adding: data_translated/coupon_detail_train.csv (deflated 66%)
adding: data_translated/coupon_list_test.csv (deflated 78%)
adding: data_translated/coupon_list_train.csv (deflated 79%)
adding: data_translated/coupon_area_train.csv (deflated 89%)
adding: data_translated/user_list.csv (deflated 55%)
```

```
In [ ]:
```