Culture Data Competition

0. Setting

Library Call

In [1]:
```
# Library Call
import glob
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
import time
import datetime as dt
import random
# !pip install folium
# !pip install haversine
import folium
from haversine import haversine


# 한글 폰트 패치.
matplotlib.rcParams['font.family']='Malgun Gothic'
matplotlib.rcParams['axes.unicode_minus'] = False

# 그래프에 retina display 적용
%config InlineBackend.figure_format = 'retina'

# ggplot style skima
plt.style.use("ggplot")

# seaborn plot style definition
sns.set_style("whitegrid")
sns.set_context("talk")

# 경고문 처리
import warnings
warnings.filterwarnings('ignore')
```

Visualization Font Setting

In [2]:
```
import platform

def get_font_family():
    system_name = platform.system()

    if system_name == "Darwin" :
        font_family = "AppleGothic"
    elif system_name == "Windows":
        font_family = "Malgun Gothic"
    else:
```

```
        !apt-get install fonts-nanum -qq  > /dev/null
        !fc-cache -fv

        import matplotlib as mpl
        mpl.font_manager._rebuild()
        findfont = mpl.font_manager.fontManager.findfont
        mpl.font_manager.findfont = findfont
        mpl.backends.backend_agg.findfont = findfont

        font_family = "NanumBarunGothic"
    return font_family

%config InlineBackend.figure_format = 'retina'

plt.rc("font", family=get_font_family())
plt.rc("axes", unicode_minus=False)

get_font_family()
```

Out[2]:
```
'Malgun Gothic'
```

User Function Definition

In [3]:
```
# Merge Same DataFrame Format
def Multiple_Data_Load(flist):
    df = []
    for file in flist:
        ele = pd.read_csv(file)
        df.append(ele)
    return pd.concat(df)

# Return Col names by Metadata Seat
def return_col(meta):
    df = pd.read_excel('Metadata/'+meta,index_col='순서')
    col = df['컬럼한글명'].values
    return col

# return MissingValue Existence
def MissingValue_Existence(df):
    value = df.isna().sum().sum()
    return f"MissinValue's counts : {value}"
```

1. Data Load

In [4]:
```
# Data Load
card_shop_df = pd.read_csv('data/ak_lwicc_card_mrhst_info_202103.csv')
card_shop_df.columns=return_col('차상위계층 카드 가맹점 정보_컬럼정의서.xls')
print('card_shop_df.shape :',card_shop_df.shape)
card_shop_df.head(2)
```

```
card_shop_df.shape : (5027, 12)
```

Out[4]:

	일련번호	가맹점명	가맹점위도	가맹점경도	가맹점시도코드	가맹점시도명	가맹점시군구코드	가맹점시군구명	가맹점행정동코드	가맹점행정동명	가맹점구분코드	가맹점구분코드명
0	14	GS25 R 부산동구2점	35.136637	129.065045	21	부산광역시	21030	동구	2103071	범일1동	B	급식카드가맹점
1	15	GS25 R 부산동구3점	35.136637	129.065045	21	부산광역시	21030	동구	2103071	범일1동	B	급식카드가맹점

In [5]:
```
# Data Information
card_shop_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5027 entries, 0 to 5026
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   일련번호        5027 non-null   int64  
 1   가맹점명        5027 non-null   object 
 2   가맹점위도       5027 non-null   float64
 3   가맹점경도       5027 non-null   float64
 4   가맹점시도코드     5027 non-null   int64  
 5   가맹점시도명      5027 non-null   object 
 6   가맹점시군구코드    5027 non-null   int64  
 7   가맹점시군구명     5027 non-null   object 
 8   가맹점행정동코드    5027 non-null   int64  
 9   가맹점행정동명     5027 non-null   object 
 10  가맹점구분코드     5027 non-null   object 
 11  가맹점구분코드명    5027 non-null   object 
dtypes: float64(2), int64(4), object(6)
memory usage: 471.4+ KB
```

2. Data EDA & Preprocessing

In [6]:
```
# 급식카드가맹점 : B | 문화누리가맹점 : M
card_shop_df['가맹점구분코드'].value_counts()
```

Out[6]:
```
B    3772
M    1255
Name: 가맹점구분코드, dtype: int64
```

In [7]:
```
# Feature Selection
card_shop_df.drop(['일련번호'],axis=1,inplace=True)
card_shop_df = card_shop_df[card_shop_df['가맹점구분코드명'] == '문화누리가맹점'].re
print('card_shop_df.shape :',card_shop_df.shape)
card_shop_df.head(2)
```

```
card_shop_df.shape : (1255, 11)
```

Out[7]:

	가맹점명	가맹점위도	가맹점경도	가맹점시도코드	가맹점시도명	가맹점시군구코드	가맹점시군구명	가맹점행정동코드	가맹점행정동명	가맹점구분코드	가맹점구분코드명
0	삼천리자전거동래점	35.202679	129.083916	21	부산광역시	21060	동래구	2106051	수민동	M	문화누리가맹점
1	행복한스튜디오	35.152427	129.054766	21	부산광역시	21050	부산진구	2105052	부전2동	M	문화누리가맹점

In [8]:
```
# Subset for Mapping
map_subset = card_shop_df[['가맹점명','가맹점위도','가맹점경도']]
map_subset.columns = ['가맹점명','위도','경도']
print('map_subset :',map_subset.shape)
map_subset.head()
```

```
map_subset : (1255, 3)
```

Out[8]:

	가맹점명	위도	경도
0	삼천리자전거동래점	35.202679	129.083916
1	행복한스튜디오	35.152427	129.054766
2	네이쳐앤트리	35.061973	128.984193
3	신평태권도장	35.093446	128.973558
4	송무인 신금 태권도	35.252920	129.013612

In [9]:
```
# Adding Cost
random.seed(42)
cost = []
for i in range(map_subset.shape[0]):
    # 비용은 10,000원에서 60,000원 사이로 무작위 배열
    ele = random.randrange(10000,40000,1000)
    cost.append(ele)
map_subset['비용'] = cost
map_subset.head()
```

Out[9]:

	가맹점명	위도	경도	비용
0	삼천리자전거동래점	35.202679	129.083916	30000
1	행복한스튜디오	35.152427	129.054766	13000
2	네이쳐앤트리	35.061973	128.984193	10000
3	신평태권도장	35.093446	128.973558	33000
4	송무인 신금 태권도	35.252920	129.013612	18000

In [10]:
```
# Adding Category
np.random.seed(42)
category = ['생활체육', '문화공연관람', '여행']
map_subset['카테고리'] = np.random.choice(category,map_subset.shape[0],replace=True
map_subset.head()
```

Out[10]:

	가맹점명	위도	경도	비용	카테고리
0	삼천리자전거동래점	35.202679	129.083916	30000	여행
1	행복한스튜디오	35.152427	129.054766	13000	생활체육
2	네이쳐앤트리	35.061973	128.984193	10000	여행
3	신평태권도장	35.093446	128.973558	33000	여행
4	송무인 신금 태권도	35.252920	129.013612	18000	생활체육

생활체육, 문화공연관람, 여행

3. Recommendation Algorithm

In [47]:

```
# WGS84 좌표계(위도 경도) 기준으로 거리 계산
def distance(df, coord):
    dist = [haversine(coord, [df.iloc[i]['위도'], df.iloc[i]['경도']], unit='m') fo
    return dist

# 중복되는 카테고리 없이 예산 Deposit 안에서 Top n개의 문화시설 추천.
def decision_rule(df,deposit):
    # 거주지와의 거리 차이 순으로 오름차순 정렬
    df = df.sort_values('거리').reset_index(drop=True)

    # 탐색
    remain_cat= ['생활체육', '문화공연관람', '여행']
    accepted = []
    idx = 0
    neg = 0
    n = 10
    remain_num = 30
    while True:

        # 다음 문화시설 데이터 객체화
        next_shop = df.iloc[idx]
        next_cat = next_shop['카테고리']
        next_cost = next_shop['비용']

        # 채택(Accepted) : 아직 선택되지 않은 카테고리이며, 예산 안에 이용가능할 경우
        if (next_cat in remain_cat) & (next_cost <= deposit):
            remain_cat.remove(next_cat)
            accepted.append(idx)
            deposit -= next_cost
            neg = 0
        else:
            neg += 1

        # 정지 규칙 : 탐색에 연속해서 n번 이상 실패할 경우 종료
        if neg >= n:
            break

        # 다음 루프를 위한 파라미터 조정
        idx += 1
        if not bool(remain_cat):
            remain_cat= ['생활체육', '문화공연관람', '여행']

    print('남은 금액 :',deposit)
    accepted_df = df.iloc[accepted].reset_index(drop=True)
    not_accepted_df = df.drop(accepted).reset_index(drop=True)
```

```
    return accepted_df, not_accepted_df[:remain_num]

def Mapping_Node(df,df_not,coord):
    # 사용자 거주지 기준 지도 생성
    m = folium.Map(
        location = coord,
        zoom_start = 15,
        tiles='cartodbpositron')

    # 승인(Accepted) : 추천 문화시설 매핑
    for i in range(df.shape[0]):
        folium.Marker([df.iloc[i,1],df.iloc[i,2]],
                      tooltip = f'Accepted<br />Name: {df.iloc[i,0]}<br />Cost: {df.i
                      icon = folium.Icon('blue',icon='star')).add_to(m)

    # 주변(Not_Accepted) : 주변 문화시설 매핑
    for i in range(df_not.shape[0]):
        folium.Marker([df_not.iloc[i,1],df_not.iloc[i,2]],
                      tooltip = f'Not Accepted<br />Name: {df_not.iloc[i,0]}<br />Cos
                      icon = folium.Icon('green',icon='ok-sign')).add_to(m)

    # 사용자 위치 매핑 및 수렴 범위 시각화
    boundary = df.iloc[-1]['거리']
    folium.Marker(coord,
                  tooltip = '<b>User</b>',
                  icon = folium.Icon('black',icon='user')).add_to(m)

    folium.Circle(coord,
                  color = 'red',
                  fill_color = 'red',
                  radius=boundary).add_to(m)
    return m

# 추천 시스템
def Recommendation_Algorithm(df,coord,deposit):
    # 사용자의 거주지를 기준으로 인근 문화시설과의 거리 계산.
    df['거리'] = distance(df,coord)

    # 중복되는 카테고리 없이 예산 Deposit 안에서 이용할 수 있는 Top n개의 문화시설 추
    df_accepted, df_not = decision_rule(df, deposit)

    # 사용자 위치를 중심으로 추천 문화시설 및 주변시설 매핑
    m = Mapping_Node(df_accepted, df_not, coord)

    return df_accepted, m
```

4. Service Inplementation

In [26]:
```
# Example Coordination
busan_coord = [35.1796, 129.0756]
busan_station_coord = [35.1152, 129.0422]
busan_namgu_coord = [35.1366, 129.0844]
```

In [48]:
```
# Service Implementation
df, m = Recommendation_Algorithm(map_subset,busan_namgu_coord,100000)
df
```

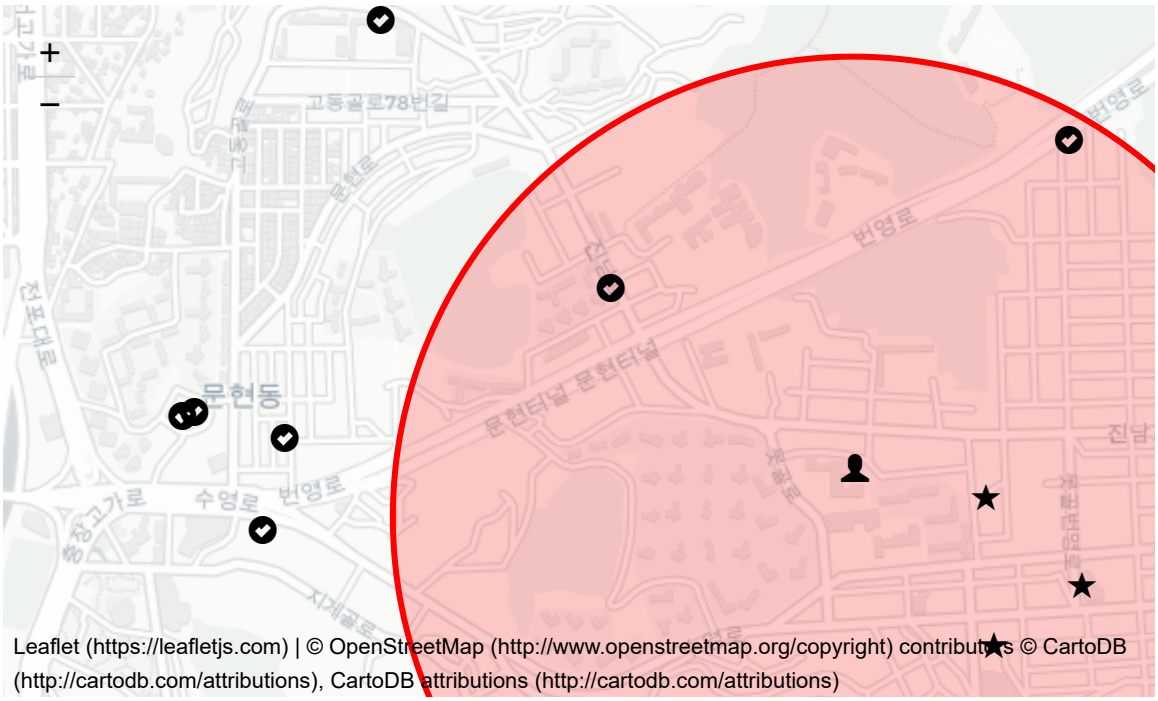
남은 금액 : 3000

Out[48]:

	가맹점명	위도	경도	비용	카테고리	거리
0	지성도서	35.136089	129.087233	25000	문화공연관람	263.806919
1	아이랜드	35.133482	129.087437	23000	여행	443.255061
2	세븐브릭스	35.134532	129.089295	37000	생활체육	501.002211
3	팬레코드	35.135508	129.094187	12000	여행	898.216027

In [49]:
```
# Mapping
m
```

Out[49]:



In [50]:
```
# # Map Figure Save
# m.save('figure/Recommendation_Map_0718.html')
```