import requests

from bs4 import BeautifulSoup

from tqdm import tqdm

import pandas as pd

def c(raw):

tmp = []

for t in raw :

tmp.append(t.text)

return tmp

def c\_raw(num,variables,soup):

tmp = soup.find\_all(variables[num].lower())

return(tmp)

def c\_s\_f(variables,soup):

c\_s = []

for i in tqdm(range(len(variables))):

tmp = c\_raw(i,variables,soup)

tmp = c(tmp)

c\_s.append(tmp)

return c\_s

def open\_api\_load(key,item,variables):

start\_t = 1

end\_t = 1

sep = 1000

url = 'http://openapi.seoul.go.kr:8088/{}/xml/{}/{}/{}/'.format(key,item,start\_t,end\_t)

req = requests.get(url)

html = req.text

soup = BeautifulSoup(html, 'html.parser')

end = int(soup.find('list\_total\_count').text)

print(end)

end\_list = []

for i in range(1000, end, sep):

end\_list.append(i)

end\_list.append(end)

all\_df = []

for idx, i in tqdm(enumerate(end\_list)):

if idx != 0:

start = end\_list[idx-1] + 1

end = i

else:

start = 1

end = i

url = 'http://openapi.seoul.go.kr:8088/{}/xml/{}/{}/{}/'.format(key,item,start,end)

print('{}% 완료'.format(round((end/end\_list[-1])\*100)))

req = requests.get(url)

html = req.text

soup = BeautifulSoup(html, 'html.parser')

df = c\_s\_f(variables,soup)

df = pd.DataFrame(dict(zip(variables,df)))

all\_df.append(df)

one\_df = pd.concat(all\_df)

return one\_df

# 점포수 데이터.

store\_df = open\_api\_load(key = '735a436859686f323132397045786574',

variables = ['STDR\_YY\_CD',

'STDR\_QU\_CD',

'TRDAR\_SE\_CD',

'TRDAR\_SE\_CD\_NM',

'TRDAR\_CD',

'TRDAR\_CD\_NM',

'SVC\_INDUTY\_CD',

'SVC\_INDUTY\_CD\_NM',

'STOR\_CO',

'SIMILR\_INDUTY\_STOR\_CO',

'OPBIZ\_RT',

'OPBIZ\_STOR\_CO',

'CLSBIZ\_RT',

'CLSBIZ\_STOR\_CO',

'FRC\_STOR\_CO'],

item = 'VwsmTrdarStorQq')

# 매출 데이터

sales\_df = open\_api\_load(key = '735a436859686f323132397045786574',

variables = ['STDR\_YY\_CD',

'STDR\_QU\_CD',

'TRDAR\_SE\_CD',

'TRDAR\_SE\_CD\_NM',

'TRDAR\_CD',

'TRDAR\_CD\_NM',

'SVC\_INDUTY\_CD',

'SVC\_INDUTY\_CD\_NM',

'THSMON\_SELNG\_AMT'],

item = 'VwsmTrdarSelngQq')

# 상권변화지표.

index\_df = open\_api\_load(key = '735a436859686f323132397045786574',

variables = ['STDR\_YY\_CD',

'STDR\_QU\_CD',

'TRDAR\_SE\_CD',

'TRDAR\_SE\_CD\_NM',

'TRDAR\_CD',

'TRDAR\_CD\_NM',

'TRDAR\_CHNGE\_IX\_NM',

'OPR\_SALE\_MT\_AVRG',

'CLS\_SALE\_MT\_AVRG',

'SU\_OPR\_SALE\_MT\_AVRG',

'SU\_CLS\_SALE\_MT\_AVRG'],

item = 'VwsmTrdarIxQq')

# store\_df.to\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\store\_df.csv', index = False, encoding = 'ms949')

# sales\_df.to\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\sales\_df.csv', index = False, encoding = 'ms949')

# index\_df.to\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\index\_df.csv', index = False, encoding = 'ms949')

# 데이터 전처리

pop\_df = pd.read\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\pop\_df.csv', engine = 'python')

sales\_df = pd.read\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\sales\_df.csv', engine = 'python')

store\_df = pd.read\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\store\_df.csv', engine = 'python')

biz\_list = ['분식전문점','양식음식점','일식음식점','중식음식점','치킨전문점','커피·음료','패스트푸드점','한식음식점']

index = []

for i in sales\_df['SVC\_INDUTY\_CD\_NM']:

if i in biz\_list:

index.append(True)

else:

index.append(False)

sales\_df = sales\_df[index]

sns.kdeplot(sales\_df['THSMON\_SELNG\_AMT'])

# 기술 통계량.

list(sales\_df)

df = sales\_df.groupby(['STDR\_YY\_CD','STDR\_QU\_CD','SVC\_INDUTY\_CD\_NM','TRDAR\_SE\_CD\_NM'])['THSMON\_SELNG\_AMT'].mean()

df = df.reset\_index()

df.head()

import seaborn as sns

from matplotlib import font\_manager, rc

font\_name = font\_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get\_name()

rc('font', family=font\_name)

sns.set()

grid = sns.FacetGrid(df, row="SVC\_INDUTY\_CD\_NM", col="TRDAR\_SE\_CD\_NM", margin\_titles=True)

grid.map(plt.plot, "THSMON\_SELNG\_AMT" );

# 상권 영역별 비교.

sum(pop\_df[pop\_df['STDR\_YY\_CD'] == 2019]['TRDAR\_SE\_CD\_NM'].value\_counts() / 2)

## 골목상권

se\_nm = '전통시장'

df = sales\_df[sales\_df['TRDAR\_SE\_CD\_NM'] == se\_nm]

## 업종별 평균매출

mean = df.groupby('SVC\_INDUTY\_CD\_NM')['THSMON\_SELNG\_AMT'].mean()

median = df.groupby('SVC\_INDUTY\_CD\_NM')['THSMON\_SELNG\_AMT'].median()

## 업종별 표준편차

sd = df.groupby('SVC\_INDUTY\_CD\_NM')['THSMON\_SELNG\_AMT'].std()

## 업종별 추세

inclination\_dict = dict()

for biz in biz\_list:

df2 = df[df['SVC\_INDUTY\_CD\_NM'] == biz].groupby(['STDR\_YY\_CD','STDR\_QU\_CD'])['THSMON\_SELNG\_AMT'].mean()

df2.plot()

plt.title(se\_nm + ' ' + biz)

plt.show()

df2 = df2.to\_frame()

df2['y'] = range(len(df2))

inclination = df2.corr().iloc[0,1]

inclination\_dict[se\_nm + ' ' + biz] = inclination

# 분해 시계열. -> R로!!

import statsmodels.api as sm

df = sales\_df.groupby(['STDR\_YY\_CD','STDR\_QU\_CD','TRDAR\_SE\_CD\_NM','SVC\_INDUTY\_CD\_NM'])['THSMON\_SELNG\_AMT'].mean()

df = df.reset\_index()

df.to\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\sales\_df\_acc.csv', index = False, encoding = 'ms949')

# 분해 시계열 결과 해석.

import os

file\_list = os.listdir(r'C:\GitHub\HomeWork\Time\_series\final\_paper\ts\_r')

df\_list = []

for i in file\_list:

df = pd.read\_csv('C:/GitHub/HomeWork/Time\_series/final\_paper/ts\_r'+'/'+i)

df\_list.append(df)

sr = dict()

sr2 = dict()

for idx, biz in enumerate(biz\_list):

sr[biz] = df\_list[idx]['randommean']

sr2[biz] = df\_list[idx]['randomsd']

df = pd.DataFrame(sr)

df2 = pd.DataFrame(sr2)

df3 = pd.concat([df.iloc[0,:],df2.iloc[0,:]],axis = 1)

# ARIMA

df = pd.read\_csv(r'C:\GitHub\HomeWork\Time\_series\final\_paper\sales\_df\_acc.csv', engine = 'python')

list(df)

test = df.groupby('SVC\_INDUTY\_CD\_NM')['THSMON\_SELNG\_AMT'].mean()

se\_nm = '골목상권'

df = df[df['TRDAR\_SE\_CD\_NM'] == se\_nm]

list(df['SVC\_INDUTY\_CD\_NM'])

coffee = df[df['SVC\_INDUTY\_CD\_NM'] == '커피·음료']

chicken = df[df['SVC\_INDUTY\_CD\_NM'] == '치킨전문점']

list(coffee)

import matplotlib.pyplot as plt

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

series = chicken['THSMON\_SELNG\_AMT'].iloc[::-1]

series.plot()

plot\_acf(series)

plot\_pacf(series)

# 정상 과정.

import statsmodels as sm

import numpy as np

d\_1 = np.diff(series)

pd.Series(d\_1).plot()

sm.tsa.stattools.adfuller(series)

from statsmodels.tsa.arima\_model import ARIMA

model = ARIMA(series, order=(0,1,1))

model\_fit = model.fit(trend='c',full\_output=True, disp=1)

print(model\_fit.summary())

model\_fit.plot\_predict()

fore = model\_fit.forecast(steps=1)

print(fore)