Regression Model

Load Data

In [1]:
```
%autosave 20

#basic library
import pandas as pd
import numpy as np
import collections
from collections import defaultdict


#model training
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


# visulization
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn')

# Data statistics
from scipy import stats



# print all the outputs in a cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"


pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
Autosaving every 20 seconds
```

In [2]:
```
df = pd.read_excel('databaseForFunction.xlsx', index_col=0)
```

In [3]:
```
df.head()
```

Out[3]:

	State	County	Year	Month	PDensity	Population	SNAP_Applications	numberOfWorker
0	California	Alameda	2019	1	1898.5	1559308	5515	4
1	California	Alameda	2019	2	1898.5	1559308	4478	39
2	California	Alameda	2019	3	1898.5	1559308	5041	19
3	California	Alameda	2019	4	1898.5	1559308	5253	80
4	California	Alameda	2019	5	1898.5	1559308	8074	50

Add one more feature - dummy value for state

In [4]:
```
# split CA and texas using dummy

# create dummy variables
dummies = pd.get_dummies(df['State'], prefix='State')

# concatenate the dummy variables with the original dataframe
df = pd.concat([df, dummies], axis=1)
```

In [5]:
```
# drop State_Texas since using dummy

df = df.drop(columns='State_Texas')
```

Update DatabaseForFunction, May 8, 2023

1. Delete column ‘snap_per_capita’
2. Delete column ‘last_google_snap’ and ‘google_snap’ because of overlap meaning with google word ‘supplemental nutrition assistance program’
3. Add seasonal dummy variable

- Summer: whether last month is 6/7/8
- holiday: whether last month is 11/12

In [12]:
```
df.head()
```

Out[12]:

	State	County	Year	Month	PDensity	Population	SNAP_Applications	numberOfWorkers
0	California	Alameda	2019	1	1898.5	1559308	5515	4!
1	California	Alameda	2019	2	1898.5	1559308	4478	39
2	California	Alameda	2019	3	1898.5	1559308	5041	19
3	California	Alameda	2019	4	1898.5	1559308	5253	80
4	California	Alameda	2019	5	1898.5	1559308	8074	50

In [8]:
```
# 3. Add seasonal dummy variables

# 3.1 add 'summer' -> whether last month is 6/7/8
# 3.2 add 'holiday' -> whether last month is 11/12
summer_month = [6, 7, 8]
holiday_month = [11, 12]


df['summer'] = 0
df['holiday'] = 0

for i in range(len(df)):
    if df.iloc[i,16].month in summer_month:
        df.iloc[i, -2] = 1

    if df.iloc[i,16].month in holiday_month:
        df.iloc[i, -1] = 1
```

In [11]:
```
# 1. Delete column 'snap_per_capita'
# 2. Delete columms related with google research work snap application

df = df.drop(columns=['google_snap','last_google_snap'])
```

updated database May 8, 2023

In [13]:
```
df.to_excel('databaseForFunction_May8.xlsx')
```

Choose columns to train

- use last month date to predict this month's snap applications

In [14]:
```
col_keepinmodel = list(df.columns)
```

In [15]:
```
coltomove = ['State','County','Year','Month', 'date_time','last_date_ti
             'numberOfWorkers', 'numberOfDisaster','google_calfresh',\
             'google_food_bank','google_food_pantry','google_food_stamps
             'google_supplemental']

for c in coltomove:
    col_keepinmodel.remove(c)

col_keepinmodel
```

Out[15]:
```
['PDensity',
 'Population',
 'SNAP_Applications',
 'last_snap',
 'last_worker',
 'last_disaster',
 'last_google_calfresh',
 'last_google_food_bank',
 'last_google_food_pantry',
 'last_google_food_stamps',
 'last_google_supplemental',
 'State_California',
 'summer',
 'holiday']
```

In [16]:
```
df = df[col_keepinmodel]
```

In [17]:
```
df.head()
```

Out[17]:

	PDensity	Population	SNAP_Applications	last_snap	last_worker	last_disaster	last_google_c
0	1898.5	1559308	5515	0	0	0	
1	1898.5	1559308	4478	5515	45	0	
2	1898.5	1559308	5041	4478	397	0	
3	1898.5	1559308	5253	5041	191	0	
4	1898.5	1559308	8074	5253	808	0	

Build the Model

1. check the correlation

In [18]:
```
df.corr()['SNAP_Applications']
```

Out[18]:
```
PDensity                    0.631942
Population                  0.933607
SNAP_Applications           1.000000
last_snap                   0.947601
last_worker                 0.392523
last_disaster               0.036997
last_google_calfresh        0.192687
last_google_food_bank       0.004543
last_google_food_pantry     0.011594
last_google_food_stamps     0.045971
last_google_supplemental    0.043198
State_California            0.217959
summer                      0.004451
holiday                    -0.006436
Name: SNAP_Applications, dtype: float64
```

In [19]:
```
X = df.drop(columns=['SNAP_Applications'])
y = df.SNAP_Applications
```

2. Result of the linear regression model

In [20]:
```
import statsmodels.api as sm
import pandas as pd

# Add constant term to X
X = sm.add_constant(X)

# Fit linear regression model
model = sm.OLS(y, X).fit()

# Print summary of regression results
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
=========
Dep. Variable:      SNAP_Applications   R-squared:
0.925
Model:                            OLS   Adj. R-squared:
0.925
Method:                 Least Squares   F-statistic:
1.040e+04
Date:                Mon, 08 May 2023   Prob (F-statistic):
0.00
Time:                        17:24:40   Log-Likelihood:
-94156.
No. Observations:               10908   AIC:
1.883e+05
Df Residuals:                   10894   BIC:
1.884e+05
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
=======================
                              coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------------
const                     231.0256     53.814      4.293      0.000
125.540     336.511
PDensity                   -0.1757      0.039     -4.507      0.000
-0.252      -0.099
Population                  0.0027      5e-05     54.998      0.000
0.003       0.003
last_snap                   0.5937      0.007     82.494      0.000
0.580       0.608
last_worker                -0.1092      0.014     -7.749      0.000
-0.137      -0.082
last_disaster             316.9914     30.853     10.274      0.000
256.513     377.470
last_google_calfresh       -2.9388      1.920     -1.531      0.126
-6.702       0.824
last_google_food_bank       5.2657      1.654      3.184      0.001
2.024       8.507
last_google_food_pantry    -4.4055      1.090     -4.042      0.000
-6.542      -2.269
last_google_food_stamps     7.6416      4.091      1.868      0.062
-0.378      15.661
last_google_supplemental  -12.9203      4.276     -3.022      0.003
-21.302      -4.539
State_California         -190.9197     99.244     -1.924      0.054
-385.457       3.617
summer                     51.8136     32.650      1.587      0.113
-12.186     115.813
holiday                    73.0949     40.481      1.806      0.071
-6.255     152.445
==============================================================================
=========
Omnibus:                    23129.931   Durbin-Watson:
1.757
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        2576
55248.865
Skew:                          18.079   Prob(JB):
0.00
Kurtosis:                     755.058   Cond. No.
6.12e+06
=====================================================================
=========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors i
s correctly specified.
[2] The condition number is large, 6.12e+06. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

For question SVD did not converge

[https://blog.csdn.net/lijieling123/article/details/112910530 (https://blog.csdn.net/lijieling123/article/details/112910530)](https://blog.csdn.net/lijieling123/article/details/112910530)