analysis_merged_type1

April 23, 2023

1 Analysis Template

1.1 Preprocess

[ ]:
```
# resolve dependency
# !pip install pmdarima
```

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
from pmdarima.arima import ADFTest , auto_arima
%matplotlib inline
```

[ ]:
```
data_path = "../data/nypd_assault.csv"
crime = "type1"
target = "count"
date = "date"
city = "merged"
fig_size = (20,5)
```

[ ]:
```
df_by_day_nyc = pd.read_csv(data_path)
df_by_day_nyc[date] = pd.to_datetime(df_by_day_nyc[date])
df_by_day_nyc.set_index(date, inplace=True)
```

[ ]:
```
data_path = "../data/battery_occurrence_per_day.csv"
target = "Count"
date = "Date"
```

[ ]:
```
df_by_day_chi = pd.read_csv(data_path)
df_by_day_chi[date] = pd.to_datetime(df_by_day_chi[date])
df_by_day_chi.set_index(date, inplace=True)
```

[ ]:
```
df_by_day=df_by_day_nyc.join(df_by_day_chi,how='inner')
```

[ ]:
```
df_by_day[target]=df_by_day[target]+df_by_day['count']
df_by_day.drop('count',axis=1,inplace=True)
```

[ ]:
```
df_by_day
```

[ ]:
```
            Count
2006-01-01    476
2006-01-02    269
2006-01-03    264
2006-01-04    268
2006-01-05    266
…             …
2021-12-27    192
2021-12-28    171
2021-12-29    156
2021-12-30    184
2021-12-31    215

[5844 rows x 1 columns]
```

1.2 Profiling

1.2.1 By day

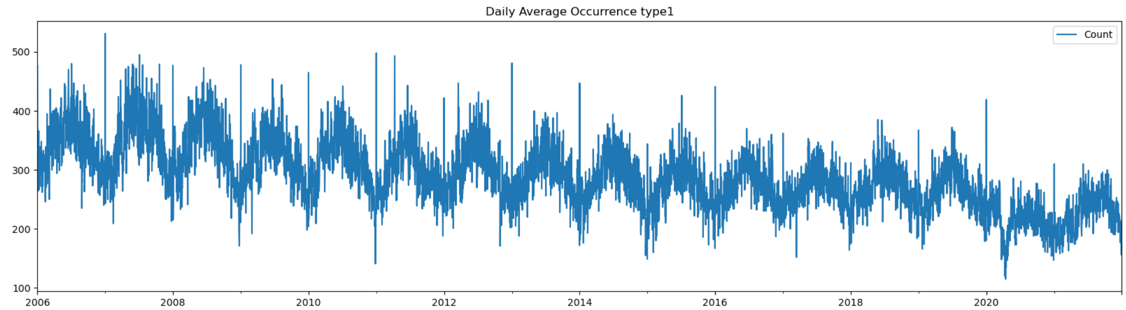
[ ]:
```
df_by_day.head()
```

[ ]:
```
            Count
2006-01-01    476
2006-01-02    269
2006-01-03    264
2006-01-04    268
2006-01-05    266
```

[ ]:
```
df_by_day.describe()
```

[ ]:
```
             Count
count  5844.000000
mean    289.381588
std      56.037430
min     115.000000
25%     251.000000
50%     286.000000
75%     324.000000
max     531.000000
```

[ ]:
```
df_by_day.plot(figsize=fig_size, title="Daily Average Occurrence " + crime)
plt.show()
```



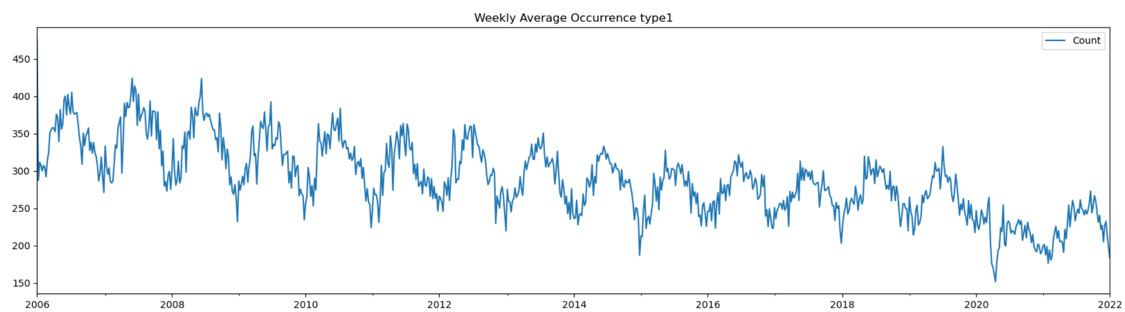
[ ]:
```
df_by_day[target].sort_values(ascending=False).head()
```

[ ]:
```
2007-01-01    531
2011-01-01    498
2007-07-05    495
2011-04-10    493
2013-01-01    481
Name: Count, dtype: int64
```

1.2.2 By week

[ ]:
```
df_by_week = pd.DataFrame(df_by_day[target].resample('W').mean())
```

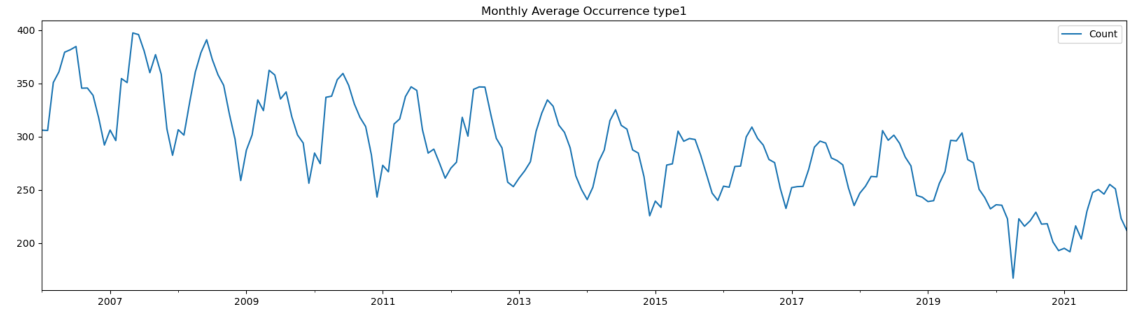
[ ]:
```
df_by_week.plot(
    figsize=fig_size,
    title="Weekly Average Occurrence " + crime)
plt.show()
```



1.2.3 By month

[ ]:
```
df_by_month = pd.DataFrame(df_by_day[target].resample('M').mean())
```

```
[ ]: df_by_month.plot(
         figsize=fig_size,
         title="Monthly Average Occurrence " + crime)
     plt.show()
```



1.3 Analysis

```
[ ]: #Ho: It is non stationary
     #H1: It is stationary

     def adfuller_test(count):
         result=adfuller(count)
         labels = ['ADF Test Statistic','p-value','#Lags Used','Number of␣
      ↪Observations Used']
         for value,label in zip(result,labels):
             print(label+' : '+str(value) )
         if result[1] <= 0.05:
             print("strong evidence against the null hypothesis(Ho), reject the null␣
      ↪hypothesis. Data has no unit root and is stationary")
         else:
             print("weak evidence against null hypothesis, time series has a unit␣
      ↪root, indicating it is non-stationary ")
```

1.3.1 Checking stationary

```
[ ]: adfuller_test(df_by_month[target])
```

```
ADF Test Statistic : -1.1711376179558388
p-value : 0.6858867530399138
#Lags Used : 14
Number of Observations Used : 177
weak evidence against null hypothesis, time series has a unit root, indicating
it is non-stationary
```

1.3.2 Checking seasonality

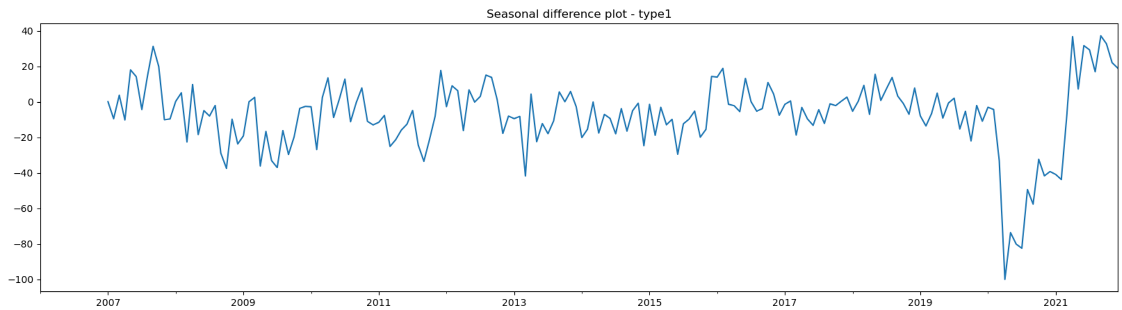
[ ]:
```
df_by_month['seasonal_first_difference'] = df_by_month[target] -␣
↪df_by_month[target].shift(12)
```

[ ]:
```
adfuller_test(df_by_month['seasonal_first_difference'].dropna())
```

```
ADF Test Statistic : -4.057446544331046
p-value : 0.001137844432949395
#Lags Used : 12
Number of Observations Used : 167
strong evidence against the null hypothesis(Ho), reject the null hypothesis.
Data has no unit root and is stationary
```

[ ]:
```
df_by_month['seasonal_first_difference'].plot(figsize=fig_size, title='Seasonal␣
↪difference plot - ' + crime)
```

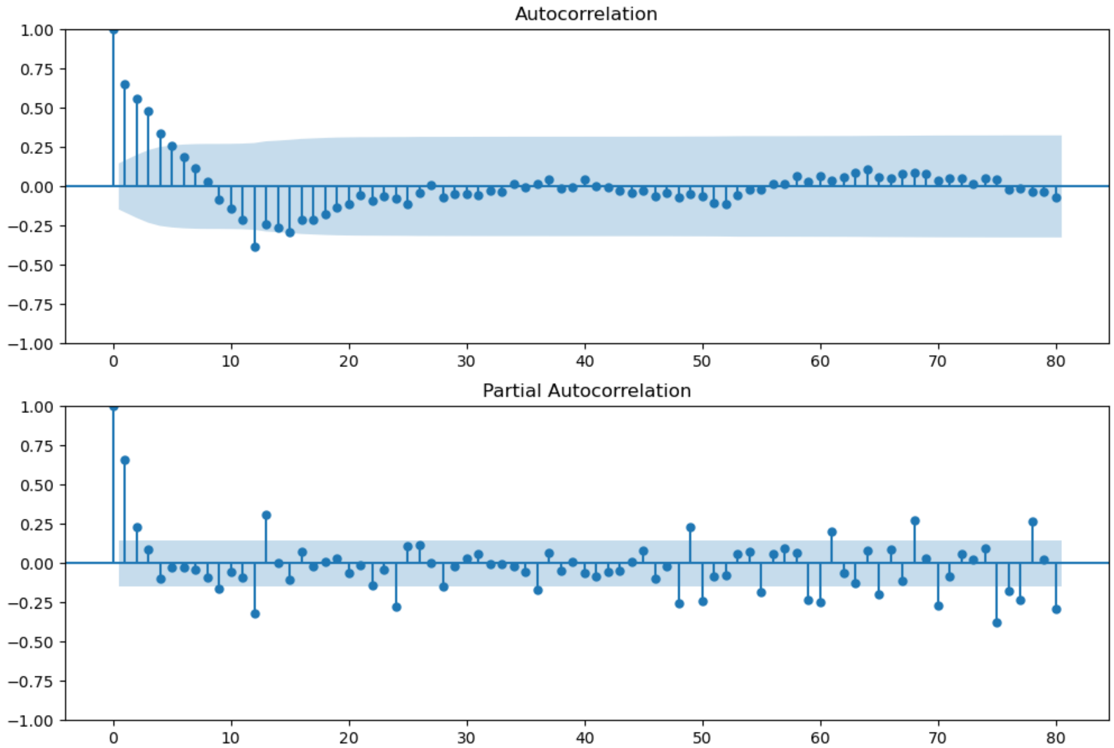
[ ]: <Axes: title={'center': 'Seasonal difference plot - type1'}>



1.3.3 Auto Regressive Model

[ ]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df_by_month['seasonal_first_difference'].iloc[13:
↪],lags=80,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df_by_month['seasonal_first_difference'].
↪iloc[13:],lags=80,ax=ax2)
```

```
/Users/xuyanchong/opt/anaconda3/lib/python3.9/site-
packages/statsmodels/graphics/tsaplots.py:348: FutureWarning: The default method
'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the
default will change tounadjusted Yule-Walker ('ywm'). You can use this method
now by setting method='ywm'.
  warnings.warn(
```

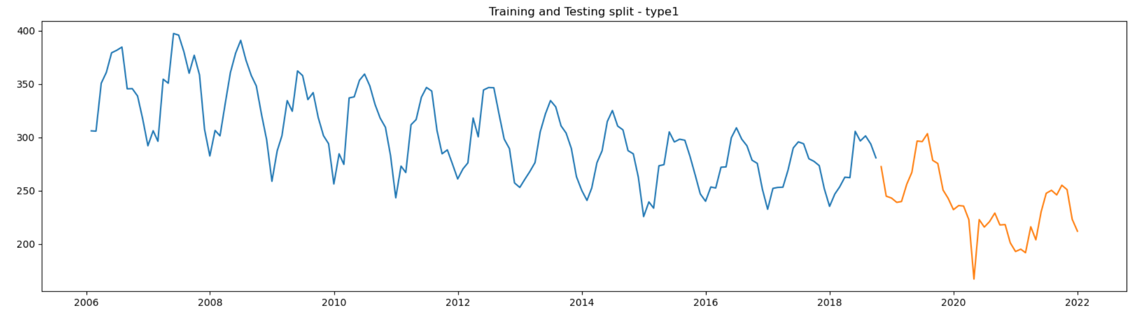


1.3.4 Implementing Seasonal Arima Model

[ ]:
```
adf_test=ADFTest(alpha=0.05)
adf_test.should_diff(df_by_month[target])
```

[ ]: (0.01, False)

[ ]:
```
start=int(df_by_month.shape[0]*0.8)
train=df_by_month[:start]
test=df_by_month[start:]
plt.figure(figsize=fig_size)
plt.plot(train[target])
plt.plot(test[target])
plt.title('Training and Testing split - '+ crime)
plt.show()
```



[ ]:
```
model=auto_arima(train[target],start_p=0,d=1,start_q=0,
          max_p=10,max_d=10,max_q=10, start_P=0,
          D=1, start_Q=0, max_P=10,max_D=10,
          max_Q=10, m=12, seasonal=True,
          error_action='warn',trace=True,
          supress_warnings=True,stepwise=True,
          random_state=20,n_fits=50)
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1163.271, Time=0.14 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=1109.836, Time=0.52 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=1075.398, Time=2.77 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=1110.967, Time=0.22 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=1076.900, Time=4.57 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=1076.813, Time=14.82 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=1089.929, Time=0.97 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=inf, Time=50.71 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=1115.759, Time=1.24 sec
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=1074.612, Time=2.35 sec
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=1112.861, Time=0.22 sec
 ARIMA(1,1,1)(1,1,1)[12]             : AIC=1076.241, Time=3.11 sec
 ARIMA(1,1,1)(0,1,2)[12]             : AIC=1076.192, Time=20.96 sec
 ARIMA(1,1,1)(1,1,0)[12]             : AIC=1090.601, Time=1.34 sec
 ARIMA(1,1,1)(1,1,2)[12]             : AIC=inf, Time=60.67 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=1090.474, Time=2.28 sec
 ARIMA(2,1,1)(0,1,1)[12]             : AIC=1074.697, Time=2.52 sec
 ARIMA(1,1,2)(0,1,1)[12]             : AIC=1074.666, Time=2.81 sec
 ARIMA(0,1,2)(0,1,1)[12]             : AIC=1075.568, Time=1.74 sec
 ARIMA(2,1,0)(0,1,1)[12]             : AIC=1085.717, Time=2.21 sec
 ARIMA(2,1,2)(0,1,1)[12]             : AIC=1076.542, Time=3.59 sec
 ARIMA(1,1,1)(0,1,1)[12] intercept   : AIC=1076.401, Time=1.87 sec

Best model:  ARIMA(1,1,1)(0,1,1)[12]
Total fit time: 181.636 seconds
```

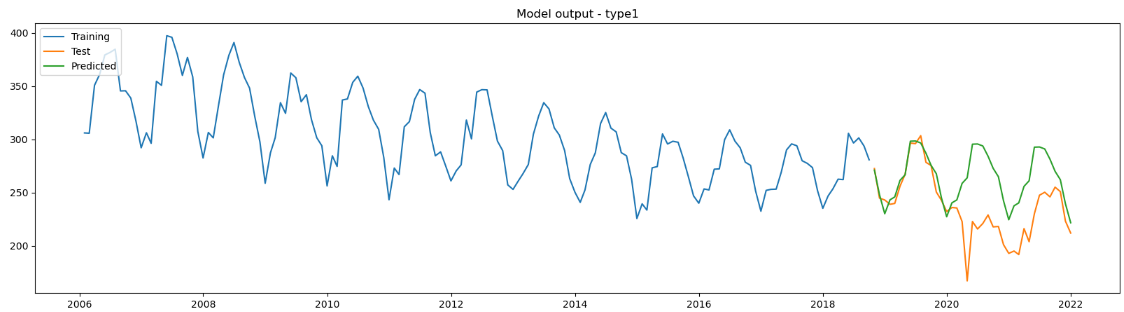
[ ]:
```
model.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                     SARIMAX Results
================================================================================
==========
Dep. Variable:                                  y   No. Observations:
153
Model:             SARIMAX(1, 1, 1)x(0, 1, 1, 12)   Log Likelihood
-533.306
Date:                            Sun, 23 Apr 2023   AIC
1074.612
Time:                                    01:34:33   BIC
1086.379
Sample:                                01-31-2006   HQIC
1079.394
                                     - 09-30-2018
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2391      0.118      2.029      0.042       0.008       0.470
ma.L1         -0.8508      0.061    -14.033      0.000      -0.970      -0.732
ma.S.L12      -0.6594      0.096     -6.879      0.000      -0.847      -0.472
sigma2       112.6244     14.294      7.879      0.000      84.609     140.640
================================================================================
===
Ljung-Box (L1) (Q):                   0.09   Jarque-Bera (JB):
2.51
Prob(Q):                              0.76   Prob(JB):
0.28
Heteroskedasticity (H):               0.56   Skew:
-0.33
Prob(H) (two-sided):                  0.05   Kurtosis:
2.98
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

[ ]:
```
prediction = pd.DataFrame(model.predict(n_periods = train.shape[0]),index=test.
  ↪index)
prediction.columns = ['predicted_crime']
```

```
plt.figure(figsize=fig_size)
plt.plot(train[target],label="Training")
plt.plot(test[target],label="Test")
plt.plot(prediction,label="Predicted")
plt.legend(loc = 'upper left')
plt.savefig('../output/%s_%s_pred.jpg' % (city,crime))
plt.title('Model output - '+crime)
plt.show()
```



[ ]:
```
np.sqrt(np.square(np.subtract(test[target].values,prediction['predicted_crime'].
↪values)).mean())
```

[ ]: 38.55733902461584