In [ ]:

```
#!/usr/bin/env python
# coding: utf-8
```

COVID - 19 and X Dataset

Assigned dataset - 20 https://github.com/michaelofsbu/CSE-544-Datasets

For the X dataset, we have chosen the Chicago Crime Data

https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

In[6]:

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime as dt
import re
import scipy.stats as stats
from scipy.stats import gamma
```

Mandatory tasks

Q 1

Load Data and get statistics

In[7]:

In [ ]:

```
covid_data = pd.read_csv('20.csv')
print(covid_data.describe())
```

In[10]:

In [ ]:

```
covid_PA_confirmed = covid_data['PA confirmed']
covid_RI_confirmed = covid_data['RI confirmed']
covid_PA_deaths = covid_data['PA deaths']
covid_RI_deaths = covid_data['RI deaths']
```

Check null values in data set

In[11]:

In [ ]:

```
print(covid_PA_confirmed.isnull().sum())
print(covid_RI_confirmed.isnull().sum())
print(covid_PA_deaths.isnull().sum())
print(covid_RI_deaths.isnull().sum())
```

No null values observed

getting individual data from cumulative data and assign zero for negative confirmed cases and deaths

In[12]:

In [ ]:

```
covid_PA_confirmed = covid_PA_confirmed.diff()
covid_PA_confirmed.fillna(0, inplace=True)
covid_PA_confirmed.loc[covid_PA_confirmed<0]=0
```

In [ ]:

```
covid_RI_confirmed = covid_RI_confirmed.diff()
covid_RI_confirmed.fillna(0, inplace=True)
covid_RI_confirmed.loc[covid_RI_confirmed<0]=0
```

In [ ]:

```
covid_PA_deaths = covid_PA_deaths.diff().reset_index(drop=True)
covid_PA_deaths.fillna(0, inplace=True)
covid_PA_deaths.loc[covid_PA_deaths<0]=0
```

In [ ]:

```
covid_RI_deaths = covid_RI_deaths.diff().reset_index(drop=True)
covid_RI_deaths.fillna(0, inplace=True)
covid_RI_deaths.loc[covid_RI_deaths<0]=0
```

In[13]:

In [ ]:

```
covid_PA_confirmed = pd.concat([covid_data['Date'],covid_PA_confirmed],axis=1)
covid_RI_confirmed = pd.concat([covid_data['Date'],covid_RI_confirmed],axis=1)
covid_PA_deaths = pd.concat([covid_data['Date'],covid_PA_deaths],axis=1)
covid_RI_deaths = pd.concat([covid_data['Date'],covid_RI_deaths],axis=1)
covid_PA_confirmed.columns=['Date', 'Count']
covid_PA_deaths.columns=['Date', 'Count']
covid_RI_confirmed.columns=['Date', 'Count']
covid_RI_deaths.columns=['Date', 'Count']
```

Outlier detection

In[18]:

Tukey's rule to check for outliers for daily Confirmed and death cases

alpha is taken as 1.5

In [ ]:

```
def outlier_detection(df):
    n = df.size
    df = df.sort_values(ascending=True)
    q1 = df[int(np.ceil(0.25*n))]
    q3 = df[int(np.ceil(0.75*n))]
    iqr = q3 - q1

    alpha = 1.5
    upper_limit = q3 + 1.5*iqr
    lower_limit = q1 - 1.5*iqr
    print('upper and lower limits for outliers: ',upper_limit,lower_limit)
```

```
    return df[((df < lower_limit) | (df > upper_limit))]
```

In[19]:

In [ ]:

```
print(outlier_detection(covid_PA_confirmed['Count']))
print(outlier_detection(covid_RI_confirmed['Count']))
print(outlier_detection(covid_PA_deaths['Count']))
print(outlier_detection(covid_RI_deaths['Count']))
```

No outliers found

- - - - - Q 2 - - - - -

Q2 - iii, iv

In[71]:

In [ ]:

```
def plot_ewma(X, test, predictions):
    plt.plot(X,test,label="original")
    plt.plot(X,predictions,label="Predictions")
    plt.xlabel('Dates - 4th week of August')
    plt.ylabel('Y')
    plt.legend(loc='upper left')
    plt.xticks(rotation=45)
    plt.show()

class EWMA:
    def __init__(self, alpha):
        self.alpha = alpha
    def predict(self,data,test):
        y_t_hat = data['Count'][0]
        #calculating y_t_hat
        for i in range(data.shape[0]):
            y_t = data['Count'][i]
            y_t_hat = self.alpha*y_t + (1-self.alpha)*y_t_hat

        mse_errors = []
        mape_errors = []
        predictions = np.zeros(len(test))
        #predicting values
        for i in np.arange(len(test)):
            y_t = test['Count'][i]
            print("Date: " + str(test['Date'][i]) + " - Test Prediction: " + "{:5.3f}".f
ormat(y_t_hat) + ", Actual: " + "{:5.3f}".format(y_t))

            #Ignoring zero data to avoid infinite error
            if y_t!=0:
                mape_errors.append(abs(y_t_hat-y_t)*100/y_t)
                mse_errors.append((y_t_hat-y_t)**2)
            predictions[i] = y_t_hat = self.alpha*y_t + (1-self.alpha)* y_t_hat
        plot_ewma(np.array(test['Date']), test['Count'], predictions)
        print("-------------------------Errors--------------------------------------")
        print("MAPE:" + "{:5.2f}".format(np.mean(mape_errors)))
        print("MSE:" + "{:5.2f}".format(np.mean(mse_errors)))
```

In[72]:

Exponentially Weighted Moving Average (Confirmed COVID Cases in August 2020)
redict Confirmed cases for 4th week of August with first three weeks data of August

In [ ]:

```
train_PA_confirmed = covid_PA_confirmed[(covid_PA_confirmed['Date']>='2020-08-01') & (cov
id_PA_confirmed['Date']<='2020-08-21')].reset_index(drop=True)
test_PA_confirmed = covid_PA_confirmed[(covid_PA_confirmed['Date']>='2020-08-22') & (covi
d_PA_confirmed['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
train_RI_confirmed = covid_RI_confirmed[(covid_RI_confirmed['Date']>='2020-08-01') & (cov
id_RI_confirmed['Date']<='2020-08-21')].reset_index(drop=True)
test_RI_confirmed = covid_RI_confirmed[(covid_RI_confirmed['Date']>='2020-08-22') & (covi
d_RI_confirmed['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
print("\n---------------------------- EWMA(0.5) -----------------------------\n")
print("\n-------- Predicting Pennsylvania confirmed cases --------\n")
ewma = EWMA(0.5)
ewma.predict(train_PA_confirmed,test_PA_confirmed)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island confirmed cases --------\n")
ewma = EWMA(0.5)
ewma.predict(train_RI_confirmed,test_RI_confirmed)
```

In [ ]:

```
print("\n---------------------------- EWMA(0.8) -----------------------------\n")
print("\n-------- Predicting Pennsylvania confirmed cases --------\n")
ewma = EWMA(0.8)
ewma.predict(train_PA_confirmed,test_PA_confirmed)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island confirmed cases --------\n")
ewma = EWMA(0.8)
ewma.predict(train_RI_confirmed,test_RI_confirmed)
```

---------------------------- EWMA(0.5) -----------------------------

-------- Predicting Pennsylvania confirmed cases --------

Date: 2020-08-22 - Test Prediction: 697.134, Actual: 796.000

Date: 2020-08-23 - Test Prediction: 746.567, Actual: 619.000

Date: 2020-08-24 - Test Prediction: 682.783, Actual: 426.000

Date: 2020-08-25 - Test Prediction: 554.392, Actual: 561.000

Date: 2020-08-26 - Test Prediction: 557.696, Actual: 501.000

Date: 2020-08-27 - Test Prediction: 529.348, Actual: 620.000

Date: 2020-08-28 - Test Prediction: 574.674, Actual: 835.000

------------------------Errors-------------------------------------

MAPE:21.66
MSE:24461.59

-------- Predicting Rhode Island confirmed cases --------

Date: 2020-08-22 - Test Prediction: 77.676, Actual: 0.000

Date: 2020-08-23 - Test Prediction: 38.838, Actual: 0.000

Date: 2020-08-24 - Test Prediction: 19.419, Actual: 0.000

Date: 2020-08-25 - Test Prediction: 9.709, Actual: 0.000

Date: 2020-08-26 - Test Prediction: 4.855, Actual: 700.000

Date: 2020-08-27 - Test Prediction: 352.427, Actual: 0.000

Date: 2020-08-28 - Test Prediction: 176.214, Actual: 0.000

------------------------Errors-------------------------------------

MAPE:99.31
MSE:483226.95

-------------------------- EWMA(0.8) ----------------------------

-------- Predicting Pennsylvania confirmed cases --------

Date: 2020-08-22 - Test Prediction: 704.619, Actual: 796.000

Date: 2020-08-23 - Test Prediction: 777.724, Actual: 619.000

Date: 2020-08-24 - Test Prediction: 650.745, Actual: 426.000

Date: 2020-08-25 - Test Prediction: 470.949, Actual: 561.000

Date: 2020-08-26 - Test Prediction: 542.990, Actual: 501.000

Date: 2020-08-27 - Test Prediction: 509.398, Actual: 620.000

Date: 2020-08-28 - Test Prediction: 597.880, Actual: 835.000

------------------------Errors-------------------------------------

MAPE:22.94
MSE:23197.88

-------- Predicting Rhode Island confirmed cases --------

Date: 2020-08-22 - Test Prediction: 19.712, Actual: 0.000

Date: 2020-08-23 - Test Prediction: 3.942, Actual: 0.000

Date: 2020-08-24 - Test Prediction: 0.788, Actual: 0.000

Date: 2020-08-25 - Test Prediction: 0.158, Actual: 0.000

Date: 2020-08-26 - Test Prediction: 0.032, Actual: 700.000

Date: 2020-08-27 - Test Prediction: 560.006, Actual: 0.000

Date: 2020-08-28 - Test Prediction: 112.001, Actual: 0.000

------------------------Errors-------------------------------------

MAPE:100.00
MSE:489955.85

In[73]:

Exponentially Weighted Moving Average (Confirmed COVID Cases in August 2020)
redict deaths for 4th week of August with first three weeks data of August

In [ ]:

```
train_PA_deaths = covid_PA_deaths[(covid_PA_deaths['Date']>='2020-08-01') & (covid_PA_de
aths['Date']<='2020-08-21')].reset_index(drop=True)
test_PA_deaths = covid_PA_deaths[(covid_PA_deaths['Date']>='2020-08-22') & (covid_PA_deat
hs['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
train_RI_deaths = covid_RI_deaths[(covid_RI_deaths['Date']>='2020-08-01') & (covid_RI_de
aths['Date']<='2020-08-21')].reset_index(drop=True)
test_RI_deaths = covid_RI_deaths[(covid_RI_deaths['Date']>='2020-08-22') & (covid_RI_deat
hs['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
print("\n--------------------------- EWMA(0.5) -----------------------------\n")
print("\n-------- Predicting Pennsylvania deaths --------\n")
ewma = EWMA(0.5)
ewma.predict(train_PA_deaths,test_PA_deaths)
ewma = EWMA(0.5)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island deaths --------\n")
ewma.predict(train_RI_deaths,test_RI_deaths)
```

In [ ]:

```
print("\n--------------------------- EWMA(0.8) -----------------------------\n")
print("\n-------- Predicting Pennsylvania deaths --------\n")
ewma = EWMA(0.8)
ewma.predict(train_PA_deaths,test_PA_deaths)
```

In [ ]:

```
ewma = EWMA(0.8)
print("\n-------- Predicting Rhode Island deaths --------\n")
ewma.predict(train_RI_deaths,test_RI_deaths)
```

--------------------------- EWMA(0.5) -----------------------------

-------- Predicting Pennsylvania deaths --------

Date: 2020-08-22 - Test Prediction: 19.131, Actual: 18.000

Date: 2020-08-23 - Test Prediction: 18.565, Actual: 2.000

Date: 2020-08-24 - Test Prediction: 10.283, Actual: 1.000

Date: 2020-08-25 - Test Prediction: 5.641, Actual: 26.000

Date: 2020-08-26 - Test Prediction: 15.821, Actual: 19.000

Date: 2020-08-27 - Test Prediction: 17.410, Actual: 11.000

Date: 2020-08-28 - Test Prediction: 14.205, Actual: 20.000

------------------------Errors-------------------------------------

MAPE:277.87
MSE:123.02

-------- Predicting Rhode Island deaths --------

Date: 2020-08-22 - Test Prediction: 0.888, Actual: 0.000

Date: 2020-08-23 - Test Prediction: 0.444, Actual: 0.000

Date: 2020-08-24 - Test Prediction: 0.222, Actual: 0.000

Date: 2020-08-25 - Test Prediction: 0.111, Actual: 0.000

Date: 2020-08-26 - Test Prediction: 0.055, Actual: 12.000

Date: 2020-08-27 - Test Prediction: 6.028, Actual: 0.000

Date: 2020-08-28 - Test Prediction: 3.014, Actual: 0.000

------------------------Errors-------------------------------------

MAPE:99.54
MSE:142.67

-------------------------- EWMA(0.8) ----------------------------

-------- Predicting Pennsylvania deaths --------

Date: 2020-08-22 - Test Prediction: 19.369, Actual: 18.000

Date: 2020-08-23 - Test Prediction: 18.274, Actual: 2.000

Date: 2020-08-24 - Test Prediction: 5.255, Actual: 1.000

Date: 2020-08-25 - Test Prediction: 1.851, Actual: 26.000

Date: 2020-08-26 - Test Prediction: 21.170, Actual: 19.000

Date: 2020-08-27 - Test Prediction: 19.434, Actual: 11.000

Date: 2020-08-28 - Test Prediction: 12.687, Actual: 20.000

------------------------Errors-------------------------------------

MAPE:209.19
MSE:142.47

-------- Predicting Rhode Island deaths --------

Date: 2020-08-22 - Test Prediction: 0.224, Actual: 0.000

Date: 2020-08-23 - Test Prediction: 0.045, Actual: 0.000

Date: 2020-08-24 - Test Prediction: 0.009, Actual: 0.000

Date: 2020-08-25 - Test Prediction: 0.002, Actual: 0.000

Date: 2020-08-26 - Test Prediction: 0.000, Actual: 12.000

Date: 2020-08-27 - Test Prediction: 9.600, Actual: 0.000

Date: 2020-08-28 - Test Prediction: 1.920, Actual: 0.000

-----------------------Errors-----------------------------------

MAPE:100.00
MSE:143.99

Q2 - i ii

In[88]:

In [ ]:

```
def plot_ar(X, test, predictions):
    plt.plot(X,test,label="original")
    plt.plot(X,predictions,label="Predictions")
    plt.xlabel('Dates - 4th week of August')
    plt.ylabel('Y')
    plt.legend(loc='upper left')
    plt.xticks(rotation=45)
    plt.show()

class AR:
    def __init__(self, p):
        self.p = p

    def predict(self, train, test):
        test_dates = np.array(test['Date'])
        test_counts = np.array(test['Count'])
        data_counts = np.hstack([train['Count'], test_counts])
        p = self.p
        t = data_counts.shape[0] - test_counts.shape[0]
        error = []
        mse = []
        predictions = np.zeros(test.shape[0])

        for i in range(t,t+test_counts.shape[0]):
            testx = [1]
            testx = np.hstack([[1], data_counts[i-p:i]])
            #calculating beta
            X = []
            Y = []
            for j in range(i):
                if(j+p < i):
                    X.append([1])
                    X[j] = X[j]+list(data_counts[j:j+p])
                    Y.append(data_counts[j+p])
                else:
                    break
            beta=np.matmul(np.linalg.inv(np.matmul(np.transpose(X),X)),np.matmul(np.tran
spose(X),Y))

            y_t_hat = predictions[i-t] = np.dot(testx,beta)
            y_t = data_counts[i]

            #Ignoring zero data to avoid infinite error
            if y_t!=0:
                error.append(abs(y_t_hat-y_t)*100/y_t)
                mse.append((y_t_hat - y_t)**2)
            print("Date: " + str(test_dates[i-t]) + " - Test prediction: " + "{:5.3f}".f
ormat(predictions[i-t]) + " | Actual: " + str(test_counts[i-t]))
```

```
        plot_ar(test_dates, test_counts, predictions)
        print("------------------------Errors---------------------------------------")
        print("MAPE: " + "{:5.3f}".format(np.mean(error)))
        print("MSE : " + "{:5.3f}".format(np.mean(mse)))
        return np.mean(error)
```

In[89]:

Auto Regression: (Confirmed COVID Cases in August 2020)
redict deaths for 4th week of August with first three weeks data of August

In [ ]:

```
train_PA_confirmed = covid_PA_confirmed[(covid_PA_confirmed['Date']>='2020-08-01') & (cov
id_PA_confirmed['Date']<='2020-08-21')].reset_index(drop=True)
test_PA_confirmed = covid_PA_confirmed[(covid_PA_confirmed['Date']>='2020-08-22') & (covi
d_PA_confirmed['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
train_RI_confirmed = covid_RI_confirmed[(covid_RI_confirmed['Date']>='2020-08-01') & (cov
id_RI_confirmed['Date']<='2020-08-21')].reset_index(drop=True)
test_RI_confirmed = covid_RI_confirmed[(covid_RI_confirmed['Date']>='2020-08-22') & (covi
d_RI_confirmed['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
print("\n--------------------------- AR(3) -----------------------------\n")
```

In [ ]:

```
print("\n-------- Predicting Pennsylvania confirmed cases --------\n")
ar3 = AR(3)
ar3.predict(train_PA_confirmed,test_PA_confirmed)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island confirmed cases --------\n")
ar3 = AR(3)
ar3.predict(train_RI_confirmed,test_RI_confirmed)
```

In [ ]:

```
print("\n--------------------------- AR(5) -----------------------------\n")
```

In [ ]:

```
print("\n-------- Predicting Pennsylvania confirmed cases --------\n")
ar3 = AR(5)
ar3.predict(train_PA_confirmed,test_PA_confirmed)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island confirmed cases --------\n")
ar3 = AR(5)
ar3.predict(train_RI_confirmed,test_RI_confirmed)
```

--------------------------- AR(3) -----------------------------

-------- Predicting Pennsylvania confirmed cases --------

Date: 2020-08-22 - Test prediction: 795.688 | Actual: 796.0

Date: 2020-08-23 - Test prediction: 737.243 | Actual: 619.0

Date: 2020-08-24 - Test prediction: 760.136 | Actual: 426.0

Date: 2020-08-25 - Test prediction: 637.590 | Actual: 561.0

Date: 2020-08-26 - Test prediction: 655.719 | Actual: 501.0

Date: 2020-08-27 - Test prediction: 670.987 | Actual: 620.0

Date: 2020-08-28 - Test prediction: 653.981 | Actual: 835.0

------------------------Errors-------------------------------------

MAPE: 24.573
MSE : 27257.157

-------- Predicting Rhode Island confirmed cases --------

Date: 2020-08-22 - Test prediction: 12.529 | Actual: 0.0

Date: 2020-08-23 - Test prediction: 192.698 | Actual: 0.0

Date: 2020-08-24 - Test prediction: 175.213 | Actual: 0.0

Date: 2020-08-25 - Test prediction: 160.637 | Actual: 0.0

Date: 2020-08-26 - Test prediction: 148.300 | Actual: 700.0

Date: 2020-08-27 - Test prediction: -15.950 | Actual: 0.0

Date: 2020-08-28 - Test prediction: -15.944 | Actual: 0.0

------------------------Errors-------------------------------------

MAPE: 78.814
MSE : 304372.448

--------------------------- AR(5) ----------------------------

-------- Predicting Pennsylvania confirmed cases --------

Date: 2020-08-22 - Test prediction: 806.785 | Actual: 796.0

Date: 2020-08-23 - Test prediction: 774.646 | Actual: 619.0

Date: 2020-08-24 - Test prediction: 739.230 | Actual: 426.0

Date: 2020-08-25 - Test prediction: 586.330 | Actual: 561.0

Date: 2020-08-26 - Test prediction: 585.629 | Actual: 501.0

Date: 2020-08-27 - Test prediction: 711.559 | Actual: 620.0

Date: 2020-08-28 - Test prediction: 680.020 | Actual: 835.0

------------------------Errors-------------------------------------

MAPE: 22.109
MSE : 23237.213

-------- Predicting Rhode Island confirmed cases --------

Date: 2020-08-22 - Test prediction: 21.037 | Actual: 0.0

Date: 2020-08-23 - Test prediction: 21.022 | Actual: 0.0

Date: 2020-08-24 - Test prediction: 21.007 | Actual: 0.0

Date: 2020-08-25 - Test prediction: 322.942 | Actual: 0.0

Date: 2020-08-26 - Test prediction: 258.800 | Actual: 700.0

Date: 2020-08-27 - Test prediction: -28.212 | Actual: 0.0

Date: 2020-08-28 - Test prediction: -28.187 | Actual: 0.0

-----------------------Errors------------------------------------

MAPE: 63.029
MSE : 194657.654

In[90]:

Auto Regression: (Confirmed COVID Cases in March 2020)
time series for the month of March (03/01/2020 to 03/31/2020)
we will predict Confirmed Covid Cases
for all Counties
print("Auto Regression: (Confirmed COVID Cases in March 2020)")

In [ ]:

```
train_PA_deaths = covid_PA_deaths[(covid_PA_deaths['Date']>='2020-08-01') & (covid_PA_de
aths['Date']<='2020-08-21')].reset_index(drop=True)
test_PA_deaths = covid_PA_deaths[(covid_PA_deaths['Date']>='2020-08-22') & (covid_PA_deat
hs['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
train_RI_deaths = covid_RI_deaths[(covid_RI_deaths['Date']>='2020-08-01') & (covid_RI_de
aths['Date']<='2020-08-21')].reset_index(drop=True)
test_RI_deaths = covid_RI_deaths[(covid_RI_deaths['Date']>='2020-08-22') & (covid_RI_deat
hs['Date']<='2020-08-28')].reset_index(drop=True)
```

In [ ]:

```
print("\n---------------------------- AR(3) ------------------------------\n")
```

In [ ]:

```
print("\n-------- Predicting Pennsylvania deaths --------\n")
ar3 = AR(3)
ar3.predict(train_PA_deaths,test_PA_deaths)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island deaths --------\n")
ar3 = AR(3)
ar3.predict(train_RI_deaths,test_RI_deaths)
```

In [ ]:

```
print("\n---------------------------- AR(5) ------------------------------\n")
```

In [ ]:

```
print("\n-------- Predicting Pennsylvania deaths --------\n")
ar3 = AR(5)
ar3.predict(train_PA_deaths,test_PA_deaths)
```

In [ ]:

```
print("\n-------- Predicting Rhode Island deaths --------\n")
ar3 = AR(5)
ar3.predict(train_RI_deaths,test_RI_deaths)
```

-------------------------- AR(3) ----------------------------

-------- Predicting Pennsylvania deaths --------

Date: 2020-08-22 - Test prediction: 17.888 | Actual: 18.0

Date: 2020-08-23 - Test prediction: 19.717 | Actual: 2.0

Date: 2020-08-24 - Test prediction: 16.444 | Actual: 1.0

Date: 2020-08-25 - Test prediction: 17.587 | Actual: 26.0

Date: 2020-08-26 - Test prediction: 26.527 | Actual: 19.0

Date: 2020-08-27 - Test prediction: 20.524 | Actual: 11.0

Date: 2020-08-28 - Test prediction: 14.622 | Actual: 20.0

-----------------------Errors-------------------------------------

MAPE: 373.762
MSE : 114.213

-------- Predicting Rhode Island deaths --------

Date: 2020-08-22 - Test prediction: 0.970 | Actual: 0.0

Date: 2020-08-23 - Test prediction: 2.213 | Actual: 0.0

Date: 2020-08-24 - Test prediction: 2.016 | Actual: 0.0

Date: 2020-08-25 - Test prediction: 1.851 | Actual: 0.0

Date: 2020-08-26 - Test prediction: 1.711 | Actual: 12.0

Date: 2020-08-27 - Test prediction: 3.749 | Actual: 0.0

Date: 2020-08-28 - Test prediction: -0.030 | Actual: 0.0

-----------------------Errors-------------------------------------

MAPE: 85.742
MSE : 105.865

-------------------------- AR(5) ----------------------------

-------- Predicting Pennsylvania deaths --------

Date: 2020-08-22 - Test prediction: 21.082 | Actual: 18.0

Date: 2020-08-23 - Test prediction: 14.471 | Actual: 2.0

Date: 2020-08-24 - Test prediction: 18.595 | Actual: 1.0

Date: 2020-08-25 - Test prediction: 24.901 | Actual: 26.0

Date: 2020-08-26 - Test prediction: 29.952 | Actual: 19.0

Date: 2020-08-27 - Test prediction: 24.052 | Actual: 11.0

Date: 2020-08-28 - Test prediction: 24.223 | Actual: 20.0

------------------------Errors-------------------------------------

MAPE: 371.691
MSE : 111.996

-------- Predicting Rhode Island deaths --------

Date: 2020-08-22 - Test prediction: 2.226 | Actual: 0.0

Date: 2020-08-23 - Test prediction: 2.228 | Actual: 0.0

Date: 2020-08-24 - Test prediction: 0.909 | Actual: 0.0

Date: 2020-08-25 - Test prediction: 2.938 | Actual: 0.0

Date: 2020-08-26 - Test prediction: 2.548 | Actual: 12.0

Date: 2020-08-27 - Test prediction: 4.419 | Actual: 0.0

Date: 2020-08-28 - Test prediction: -0.202 | Actual: 0.0

------------------------Errors-------------------------------------

MAPE: 78.769
MSE : 89.346

In [ ]:

```
covid_PA_confirmed_feb21 = covid_PA_confirmed[377:404].to_numpy()
covid_RI_confirmed_feb21 = covid_RI_confirmed[377:404].to_numpy()
covid_PA_deaths_feb21 = covid_PA_deaths[377:404].to_numpy()
covid_RI_deaths_feb21 = covid_RI_deaths[377:404].to_numpy()
```

In [ ]:

```
covid_PA_confirmed_mar21 = covid_PA_confirmed[404:435].to_numpy()
covid_RI_confirmed_mar21 = covid_RI_confirmed[404:435].to_numpy()
covid_PA_deaths_mar21 = covid_PA_deaths[404:435].to_numpy()
covid_RI_deaths_mar21 = covid_RI_deaths[404:435].to_numpy()
# print(covid_RI_deaths_mar21)
```

In[18]:

In [ ]:

```
def corrected_variance(arr):
```

```
    square_sum = 0
    mean = np.mean(arr)
    n = len(arr)
    for i in range(n):
        square_sum = square_sum + (arr[i] -mean)*(arr[i]-mean)
    return square_sum/(n-1)
```

In[19]:

In [ ]:

```
def walds_test_1sample(dist1,dist2, threshold, descr):

    dist1_mean = np.mean(dist1)
    dist2_mean = np.mean(dist2)
    numerator = dist2_mean - dist1_mean
    denominator = np.sqrt(dist2_mean/dist2.size)
    result = numerator/denominator
    w = np.abs(result)
    if(w>threshold):
      print("walds 1 sample testing for mean of "+str(descr)+" cases is w="+str(w) +" wh
ich is greater than z_alpha/2 = "+str(threshold)+" so reject the NULL hypothesis");
    else:
      print("walds 1 sample testing for mean of "+str(descr)+" cases is  w="+str(w)+ " w
hich is less than z_alpha/2 = "+str(threshold)+" so accept the NULL hypothesis")
```

In[20]:

In [ ]:

```
def walds_test_2sample(dist1, dist2, threshold, descr):
    dist1_mean = np.mean(dist1)
    dist2_mean = np.mean(dist2)

    numer = dist2_mean-dist1_mean
    denom = np.sqrt(dist2_mean/dist2.size + dist1_mean/dist1.size)
    w = abs(numer/denom)
    if(w>threshold):
      print("walds 2 sample testing for mean of "+str(descr)+" cases is w="+str(w) +" wh
ich is greater than z_alpha/2 = "+str(threshold)+" so reject the NULL hypothesis");
    else:
      print("walds 2 sample testing for mean of "+str(descr)+" cases is  w="+str(w)+ " w
hich is less than z_alpha/2 = "+str(threshold)+" so accept the NULL hypothesis")
```

In[21]:

z test

In [ ]:

```
def z_test(dist1, dist2, full, zthreshold, descr):
    dist1_mean = np.mean(dist1)
    dist2_mean = np.mean(dist2)

    numer = dist2_mean - dist1_mean
    denom = np.sqrt(corrected_variance(full)/full.size)

    z = abs(numer/denom)

    if(z>zthreshold):
      print("z test 1 sample testing for mean of "+str(descr)+" cases is w="+str(z) +" w
hich is greater than z_alpha/2 = "+str(zthreshold)+" so reject the NULL hypothesis");
    else:
      print("z test 1 sample testing for mean of "+str(descr)+" cases is  w="+str(z)+ "
which is less than z_alpha/2 = "+str(zthreshold)+" so accept the NULL hypothesis")
```

In[22]:

t test

In [ ]:

```
def t_test_1sample(dist1, dist2, tthreshold, descr):
    dist1_mean = np.mean(dist1)
    dist2_mean = np.mean(dist2)

    numer = dist2_mean - dist1_mean
    denom = np.sqrt(np.var(dist2)/dist2.size)

    t = abs(numer/denom)

    if(t>tthreshold):
      print("t test 1 sample testing for mean of "+str(descr)+" cases is w="+str(t) +" w
hich is greater than t threshold = "+str(tthreshold)+" so reject the NULL hypothesis");
    else:
      print("t test 1 sample testing for mean of "+str(descr)+" cases is  w="+str(t)+ "
which is less than t thresfold = "+str(tthreshold)+" so accept the NULL hypothesis")
```

In[23]:

t test

In [ ]:

```
def t_test_2sample_unpaired(dist1, dist2,  tthreshold, descr):
    dist1_mean = np.mean(dist1)
    dist2_mean = np.mean(dist2)

    numer = dist2_mean - dist1_mean
    denom = np.sqrt(np.var(dist2)/dist2.size + np.var(dist1)/dist1.size)

    t = abs(numer/denom)

    if(t>tthreshold):
      print("t test 2 sample testing for mean of "+str(descr)+" cases is w="+str(t) +" w
hich is greater than t threshold = "+str(tthreshold)+" so reject the NULL hypothesis");
    else:
      print("t test 2 sample testing for mean of "+str(descr)+" cases is  w="+str(t)+ "
which is less than t thresfold = "+str(tthreshold)+" so accept the NULL hypothesis")
```

In[24]:

walds 1 cases

In [ ]:

```
walds_test_1sample(covid_PA_confirmed_feb21, covid_PA_confirmed_mar21, 1.962, "PA confirm
ed")
walds_test_1sample(covid_RI_confirmed_feb21, covid_RI_confirmed_mar21, 1.962, "RI confirm
ed")
```

In [ ]:

```
walds_test_1sample(covid_PA_deaths_feb21, covid_PA_deaths_mar21, 1.962, "PA death")
walds_test_1sample(covid_RI_deaths_feb21, covid_RI_deaths_mar21, 1.962, "RI death")
```

In[25]:

z test cases

In [ ]:

```
z_test(covid_PA_confirmed_feb21, covid_PA_confirmed_mar21, covid_PA_confirmed.to_numpy(),
1.962, "PA confirmed")
```

```
z_test(covid_RI_confirmed_feb21, covid_RI_confirmed_mar21, covid_RI_confirmed.to_numpy(),
1.962, "RI confirmed")
```

In [ ]:

```
z_test(covid_PA_deaths_feb21, covid_PA_deaths_mar21, covid_PA_deaths.to_numpy(), 1.962, "
PA death")
z_test(covid_RI_deaths_feb21, covid_RI_deaths_mar21, covid_RI_deaths.to_numpy(), 1.962, "
RI death")
```

In[26]:

t test 1 sample cases

In [ ]:

```
t_test_1sample(covid_PA_confirmed_feb21, covid_PA_confirmed_mar21, 1.695, "PA confirmed")
t_test_1sample(covid_RI_confirmed_feb21, covid_RI_confirmed_mar21, 1.695, "RI confirmed")
```

In [ ]:

```
t_test_1sample(covid_PA_deaths_feb21, covid_PA_deaths_mar21, 1.695, "PA death")
t_test_1sample(covid_RI_deaths_feb21, covid_RI_deaths_mar21, 1.695, "RI death")
```

In[27]:

walds 2 sample cases

In [ ]:

```
walds_test_2sample(covid_PA_confirmed_feb21, covid_PA_confirmed_mar21, 1.962, "PA confirm
ed")
walds_test_2sample(covid_RI_confirmed_feb21, covid_RI_confirmed_mar21, 1.962, "RI confirm
ed")
```

In [ ]:

```
walds_test_2sample(covid_PA_deaths_feb21, covid_PA_deaths_mar21, 1.962, "PA death")
walds_test_2sample(covid_RI_deaths_feb21, covid_RI_deaths_mar21, 1.962, "RI death")
```

In[28]:

t test 1 sample cases

In [ ]:

```
t_test_2sample_unpaired(covid_PA_confirmed_feb21, covid_PA_confirmed_mar21, 1.672, "PA co
nfirmed")
t_test_2sample_unpaired(covid_RI_confirmed_feb21, covid_RI_confirmed_mar21, 1.672, "RI co
nfirmed")
```

In [ ]:

```
t_test_2sample_unpaired(covid_PA_deaths_feb21, covid_PA_deaths_mar21, 1.672, "PA death")
t_test_2sample_unpaired(covid_RI_deaths_feb21, covid_RI_deaths_mar21, 1.672, "RI death")
```

Applicability Of Tests:

Wald's Test:

1. **We require an asymptomatical normal estimator for wald's test and as n = 28, 30 is fairly low, this is not the ideal case for assumption of CLT for sample mean. Therefore, this test is not applicable.**

2. **The above reasoning works well for 2 sample test as well, since we need both estimators to be asymptomatically Normal. Therfore this test is also not applicable.**

Z-test:

3. **As we need to use true standard deviation in z-test , the dataset size is just 438, which is not particularly large. Neither the datasets are normally distributed. Therefore this testis not applicable.**

T-test:

4. **For one sample , similarly to above tests, the data inferred here is neither normally distributed nor large enough. Therefore the test is not applicable.**
5. **For unpaired 2 sample test, distributions need to be independent and normally distributed. But that doesnt seem the same in above case. Therefore the test is not applicable.**

In [ ]:

```
data = covid_data
data_oct_to_dec = data[data["Date"] >= '2020-10-01'][data["Date"] <= '2020-12-31']
```

In [ ]:

```
pa_confirmed = np.array(data_oct_to_dec["PA confirmed"]).astype('int')
ri_confirmed = np.array(data_oct_to_dec["RI confirmed"]).astype('int')
```

In [ ]:

```
pa_deaths = np.array(data_oct_to_dec["PA deaths"]).astype('int')
ri_deaths = np.array(data_oct_to_dec["RI deaths"]).astype('int')
```

In[461]:

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:

```
from scipy.stats import poisson, binom, geom
```

K-S Test 1-sample and 2-Sample

For 1-sample we are using MME to find the parameter estimates for PA data and testing if that is same for RI data

In[462]:

In [ ]:

```
def get_eCDF(data, s, e):

    n = len(data)
    cdf = [0.0]

    for i in range(0, n):
        cdf = cdf + [cdf[len(cdf)-1] + (1/n)]

    return cdf

def plot_eCDF(data, cdf, label, s, e):

    # Since the ecdf has two values for some x, we have to create new x and y lists from
cdf
    n = len(data)
    x, y = [s], [0]
```

```
    for i in range(0, n):
        x = x + [data[i], data[i]]
        y = y + [cdf[i], cdf[i+1]]

    x = x + [e]
    y = y + [1.0]


    plt.plot(x, y, label=label)
    plt.xlabel('x')
    plt.ylabel('CDF')
    plt.legend(loc='best')

def cdf_at_x(data, cdf, x):

    # finding the first element larger than the x and then we take cdf value at that poin
t
    x_i = -1
    for i, d in enumerate(data):
        if d >= x:
            x_i = i
            break

    return cdf[x_i]
```

In [ ]:

```
def k_s_test_2_sample(pa, ri, c = 0.05):

    # Sort the data to get the CDFs
    pa, ri = np.sort(pa), np.sort(ri)

    s = min(pa[0], ri[0]) - 100
    e = max(pa[len(pa) - 1], ri[len(ri) - 1]) + 100


    F_pa = get_eCDF(pa, s, e)
    F_ri = get_eCDF(ri, s, e)

    fig= plt.figure(figsize=(12,9))
    plt.grid(True)
    plt.title("K-S Test 2-Sample PA vs RI")
    plot_eCDF(pa, F_pa, "PA", s, e)
    plot_eCDF(ri, F_ri, "RI", s, e)

    # pa CDF at ri change points
    F_pa = [cdf_at_x(pa, F_pa, cp) for cp in ri]
    F_ri_minus, F_ri_plus = F_ri[0:-1], F_ri[1:]
    ks_stat = ks_index = 0

    # y value for the vertical line at ks_index
    ymin = 0

    for i in range(0, len(F_pa)):
        if abs(F_pa[i] - F_ri_minus[i]) > ks_stat:
            ks_stat = abs(F_pa[i] - F_ri_minus[i])
            ks_index = i
            ymin = min(F_ri_minus[ks_index], F_pa[ks_index])
        if abs(F_pa[i] - F_ri_plus[i]) > ks_stat:
            ks_stat = abs(F_pa[i] - F_ri_plus[i])
            ks_index = i
            ymin = min(F_ri_plus[ks_index], F_pa[ks_index])

    print("ks_stat is {} at {}, where as c is {}".format(ks_stat, ri[ks_index], c))
    if ks_stat > c:
        print("d > c, So, we reject Null Hypothesis")
    else:
        print("d <= c, So, we accept Null Hypothesis")
    plt.plot([ri[ks_index], ri[ks_index]],[ymin, ymin + ks_stat])
```

```
    plt.annotate("K-S statistic = " + str(ks_stat), xy = [ri[ks_index] + int((e-s)/100),
ymin + ks_stat/4], rotation = 90)
    plt.show()

    return
```

In [ ]:

```
def k_s_test_1_sample(pa, ri, dist='poisson', c = 0.05):

    pa_mean = np.mean(pa)
    pa_var = np.var(pa)

    # Sort the data to get the CDFs
    pa, ri = np.sort(pa), np.sort(ri)

    s = min(pa[0], ri[0]) - 100
    e = max(pa[len(pa) - 1], ri[len(ri) - 1]) + 100

    F_pa = []
    F_ri = get_eCDF(sorted_ri_confirmed, s, e)

    # pa CDF at ri change points
    if dist == 'poisson':
        lam_mme = pa_mean
        F_pa = [poisson.cdf(cp, lam_mme) for cp in ri]
    elif dist == 'binomial':
        p_mme = 1 - pa_var/pa_mean
        n_mme = pa_mean/p_mme
        F_pa = [binom.cdf(cp, n_mme, p_mme) for cp in ri]
    elif dist == 'geometric':
        p_mme = 1/pa_mean
        F_pa = [geom.cdf(cp, p_mme) for cp in ri]

    F_ri_minus, F_ri_plus = F_ri[0:-1], F_ri[1:]
    ks_stat = ks_index = 0

    for i in range(0, len(X)):
        if abs(F_pa[i] - F_ri_minus[i]) > ks_stat:
            ks_stat = abs(F_pa[i] - F_ri_minus[i])
            ks_index = i
        if abs(F_pa[i] - F_ri_plus[i]) > ks_stat:
            ks_stat = abs(F_pa[i] - F_ri_plus[i])
            ks_index = i

    print("ks_stat is {} at {}, where as c is {}".format(ks_stat, ri[ks_index], c))
    if ks_stat > c:
        print("d > c, So, we reject Null Hypothesis")
    else:
        print("d <= c, So, we accept Null Hypothesis")

    return
```

K-S Test 1 Sample - PA Confimed assumed to be poisson distribution vs RI Confimed, threshold, c = 0.05

Null Hypothesis: H_o = RI Confimed is poisson distribution with parameters obtained from MME on PA Confimed.

Alternate Hypothesis: H_1 = RI Confimed is poisson distribution with parameters obtained from MME on PA Confimed.

In[463]:

In [ ]:

```
k_s_test_1_sample(pa_confirmed, ri_confirmed)
```

K-S Test 1 Sample - PA Deaths assumed to be poisson distribution vs RI Deaths, threshold, c = 0.05

Null Hypothesis: H_o **= RI Deaths is poisson distribution with parameters obtained from MME on PA Deaths.**

Alternate Hypothesis: H_1 **= RI Deaths is poisson distribution with parameters obtained from MME on PA Deaths.**

In[464]:

In [ ]:

```
k_s_test_1_sample(pa_deaths, ri_deaths)
```

K-S Test 1 Sample - PA Confimred aussmed to be binomial distribution vs RI Confimred, threshold, c = 0.05

Null Hypothesis: H_o **= RI Confimred is binomial distribution with parameters obtained from MME on PA Confimred.**

Alternate Hypothesis: H_1 **= RI Confimred is binomial distribution with parameters obtained from MME on PA Confimred.**

In[465]:

In [ ]:

```
k_s_test_1_sample(pa_confirmed, ri_confirmed, dist='binomial')
```

K-S Test 1 Sample - PA Deaths assumed to be binomial distribution vs RI Deaths, threshold, c = 0.05

Null Hypothesis: H_o **= RI Deaths is binomial distribution with parameters obtained from MME on PA Deaths.**

Alternate Hypothesis: H_1 **= RI Deaths is binomial distribution with parameters obtained from MME on PA Deaths.**

In[466]:

In [ ]:

```
k_s_test_1_sample(pa_deaths, ri_deaths, dist='binomial')
```

K-S Test 1 Sample - PA Confirmed assumed to be geometric distribution vs RI Confimred, threshold, c = 0.05

Null Hypothesis: H_o **= RI Confimred is geometric distribution with parameters obtained from MME on PA Confimred.**

Alternate Hypothesis: H_1 **= RI Confimred is geometric distribution with parameters obtained from MME on PA Confimred.**

In[467]:

In [ ]:

```
k_s_test_1_sample(pa_confirmed, ri_confirmed, dist='geometric')
```

K-S Test 1 Sample - PA Deaths assumed to be geometric distribution vs RI Deaths, threshold, c = 0.05

Null Hypothesis: H_o **= RI Deaths is geometric distribution with parameters obtained from MME on PA Deaths.**

Alternate Hypothesis: H_1 **= RI Deaths is geometric distribution with parameters obtained from MME on PA Deaths.**

In[468]:

In [ ]:

```
k_s_test_1_sample(pa_deaths, ri_deaths, dist='geometric')
```

K-S Test 2 Sample - threshold, c = 0.05

Null Hypothesis: H_o = PA Confimred and RI Confirmed have similar distributions.

Alternate Hypothesis: H_1 = PA Confimred and RI Confimred have different distributions.

In[469]:

In [ ]:

```
k_s_test_2_sample(pa_confirmed, ri_confirmed)
```

K-S Test 2 Sample - threshold, c = 0.05

Null Hypothesis: H_o = PA Deaths and RI Deaths have similar distributions.

Alternate Hypothesis: H_1 = PA Deaths and RI Deaths have different distributions.

In[470]:

In [ ]:

```
k_s_test_2_sample(pa_deaths, ri_deaths)
```

Permutation Test - 1000 permutations, threshold, c = 0.05

In[473]:

In [ ]:

```
def perm_test(pa, ri, no_of_perm):

    count =0

    perm_diff = abs(np.mean(pa) - np.mean(ri))
    covid = np.array(pa.tolist() + ri.tolist())

    pa_n = pa.shape[0]
    N = covid.shape[0]

    for i in range(no_of_perm):
        perm = np.random.permutation(covid)
        d1 = perm[:pa_n+1]
        d2 = perm[pa_n+1:]
        if (abs(d1.mean() - d2.mean()) > perm_diff):
            count += 1

    p_val = count/no_of_perm

    return p_val
```

Null Hypothesis: H_o = PA Confirmed and RI Confirmed have similar distributions.

Alternate Hypothesis: H_1 = PA Confirmed and RI Confirmed have different distributions.

In[474]:

In [ ]:

```
c = 0.05
p_val_confirmed = perm_test(pa_confirmed, ri_confirmed, 1000)
```

```
print("p-value for confirmed cases = {} <= c = {}. So, we reject Null Hypothesis.".format
(p_val_confirmed, c))
```

Null Hypothesis: H_o = PA Deaths and RI Deaths have similar distributions.

Alternate Hypothesis: H_1 = PA Deaths and RI Deaths have different distributions.

In[475]:

In [ ]:

```
p_val_deaths = perm_test(pa_deaths, ri_deaths, 1000)
print("p-value for deaths = {} <= c = {}. So, we reject Null Hypothesis.".format(p_val_de
aths, c))
```

######################## 2D

In [ ]:

```
covid_data_actual = covid_data
covid_combined = pd.DataFrame()
covid_combined["date"] = covid_data_actual["Date"]
covid_combined['date'] = pd.to_datetime(covid_combined['date'])
covid_combined["cases"] = covid_data_actual["PA confirmed"] + covid_data_actual["RI conf
irmed"]
covid_combined["deaths"] = covid_data_actual["PA deaths"] + covid_data_actual["RI deaths"
]
covid_combined = covid_combined.set_index(['date'])
```

In[139]:

In [ ]:

```
covid_jun_four = covid_combined.loc['2020-6-1':'2020-6-28']
al = covid_combined.loc['2020-6-29':'2020-7-05']
# print(al['deaths'].mean())
# covid_jun_four.reset_index(drop=False, inplace=True)
lambda_mme = covid_jun_four['deaths'].mean()
# print(lambda_mme)
beta = lambda_mme
week_5 = covid_combined.loc['2020-6-29':'2020-7-05']
# week_5.reset_index(drop=False, inplace=True)
deaths_5 = list(week_5['deaths'])
week_6 = covid_combined.loc['2020-7-06':'2020-7-12']
# week_6.reset_index(drop=False, inplace=True)
deaths_6 = list(week_6['deaths'])
week_7 = covid_combined.loc['2020-7-13':'2020-7-19']
# week_7.reset_index(drop=False, inplace=True)
deaths_7 = list(week_7['deaths'])
week_8 = covid_combined.loc['2020-7-20':'2020-7-26']
# week_8.reset_index(drop=False, inplace=True)
deaths_8 = list(week_8['deaths'])
```

print(np.mean(deaths_5), np.mean(deaths_6), np.mean(deaths_7), np.mean(deaths_8))

In [ ]:

```
deaths = [deaths_5,deaths_6,deaths_7,deaths_8]
deaths
# deaths_alt = deaths_5+deaths_6+deaths_7+deaths_8
# print(deaths_alt)
```

In [ ]:

```
plt.figure(figsize=(16,8))
death_sum = 0
```

```
i=0
for d_i in deaths:
    death_sum += sum(d_i)
    alpha = death_sum + 1
    b = (i+1)*7 + (1/beta)
    x = np.linspace(gamma.ppf(0.01, alpha, scale=1/b), gamma.ppf(0.99, alpha, scale=1/b)
, 100)
    plt.title("Posterior Gamma distributions")
    label= "Week-" + str(i+5) + " MAP(mean): " + str(alpha/b)
    plt.plot(x, gamma.pdf(x, alpha, scale=1/b), label=label)
    plt.xlabel("Deaths")
    plt.ylabel("PDF of Gamma distribution")
    plt.legend()
    i+=1
plt.show()
```

In [ ]:

```

```

In [ ]:

```

```

Observations:

- **From the above graphs, we can say that as the weeks progress, MAP is reducing indicating a decrease in number of deaths**
- **We can also infer that as the time progresses, the number of deaths might saturate if the trend follows a similar pattern (rate)**