part_d

May 17, 2022

OUTPUT IS AT THE END OF THIS FILE. - 2 dataframes for LA and MD each with each method of prediction of AR and EWMA, and their associated metrics also printed along with them.

NOTE: Due to outlier removal in cleaning phase.

- for LA - 2 dates were removed in may 1-21 hence predictions start from 23 MAY, still considering 3 weeks of available data before 23 MAY
- Similarly, for MD - 3 dates were removed in may 1-21, and one date in 22-31, hence predictions start from 24 MAY, still considering 3 weeks of available data before 24 MAY and only 6 predictions are available(since 1 more date is removed from 22-31)

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[2]:
```
%cd /content/drive/Shareddrives/CSE544_Project/covid_dataset
!ls
```

```
/content/drive/Shareddrives/CSE544_Project/covid_dataset
backup
colab_pdf.py
COVID-19_Vaccinations_in_the_United_States_Jurisdiction.csv
covid_la_cleaned.csv
covid_la_cleaned_removed_outliers.csv
covid_md_cleaned.csv
covid_md_cleaned_removed_outliers.csv
__pycache__
United_States_COVID-19_Cases_and_Deaths_by_State_over_Time.csv
vacc_la_clean.csv
vacc_la_clean_removed_outliers.csv
vacc_md_clean.csv
vacc_md_clean_removed_outliers.csv
```

[3]:
```
#importing necessary libraries
import pandas as pd
import warnings
from pprint import pprint as pp
```

```
import numpy as np
warnings.filterwarnings("ignore")
```

[4]:
```
# Taking LA and MD vaccines cleaned csv iles
df_la = pd.read_csv("vacc_la_clean_removed_outliers.csv")
df_md = pd.read_csv("vacc_md_clean_removed_outliers.csv")
print(df_la.columns)
print(df_md.columns)
```

```
Index(['Date', 'Administered', 'Administered_daily'], dtype='object')
Index(['Date', 'Administered', 'Administered_daily'], dtype='object')
```

[5]:
```
# Type converting to numeric data for vaccines
df_la["Administered_daily"] = pd.to_numeric(df_la["Administered_daily"])
df_md["Administered_daily"] = pd.to_numeric(df_md["Administered_daily"])
print(df_la["Administered_daily"].dtype)
print(df_md["Administered_daily"].dtype)
```

```
float64
float64
```

[6]:
```
# Creating output dataframes to be shown in end
df_la_output = pd.DataFrame()
df_md_output = pd.DataFrame()
df_metrics = pd.DataFrame()
metrics_dict = dict()
metrics_dict['la'] = dict()
metrics_dict['md'] = dict()
```

[7]:
```
#helper for mse and mape - returns mse and mape of actual vs predictions
#inputs are lists

def calculate_metrics(actual, predicted):
  sse = 0
  mape_calc = 0


  for a,p in zip(actual, predicted):
    residual = a - p
    sse += residual ** 2

    mape_calc += abs(residual / a) * 100


  mse = sse / len(predicted)
  mape = mape_calc / len(predicted)
```

```
  return mse, mape
```

[8]:
```
#Q2- d (i) and (ii)
#Functions - Autoregression code block to return predictions based on AR

# Creating initial matrix of MLR
def start_matrix(values, p):
    X = []
    Y = []
    # MLR - X and Y matrices
    for i in range(len(values) - p):
        val = values[i:(i + p)]
        val = [1] + val
        X.append(val)
        Y.append(values[i + p])

    X = np.array(X)
    Y = np.array(Y)
    return X, Y

# Finding beta constants using OLS
def get_beta(X, Y):
    # finding beta using OLS
    Xt = np.transpose(X)
    XtdotX = np.dot(Xt, X)
    Xtdoty = np.dot(Xt, Y)
    beta = np.linalg.solve(XtdotX, Xtdoty)
    return beta

# return predictions using start matrix and calculating betas
def auto_regression(data, prev):
    X, Y = start_matrix(data[:-7], prev)
    beta = get_beta(X, Y)
    predictions = []
    for i in range(len(data) - 7, len(data)):
        value = data[i - prev:i]
        value = np.array([1] + value)
        x = np.dot(beta, value)
        predictions.append(x)
        value = np.reshape(value, (1, prev + 1))
        X = np.append(X, list(value), axis=0)
        Y = np.append(Y, data[i])
        beta = get_beta(X, Y)
    return predictions
```

[9]:
```
# Part 2D (iii) and (iv) for LA state
# Getting predictions for LA using AR(3) and AR(5)
```

```
# NOTE : 2 missing dates in month of may for LA because of outlier removal,␣
 ↪hence, predictions is from 23-05


# Processing LA DF for month of may
ar_df_la = df_la[(df_la.Date >= '2021-05-01') & (df_la.Date <= '2021-05-31')]

dates_list = list(ar_df_la['Date'])[0:28]
vaccines_administered_list = list(ar_df_la['Administered_daily'])[0:28]

actual = vaccines_administered_list[-7:]

dates = dates_list[-7:]

#Getting predictions and storing in output dataframe for LA
df_la_output['Date'] = dates_list[-7:]
df_la_output['Vaccines_administered'] = actual
df_la_output['AR(3)_predicted'] = auto_regression(vaccines_administered_list, 3)
df_la_output['AR(5)_predicted'] = auto_regression(vaccines_administered_list, 5)


mse_3, mape_3 = calculate_metrics(actual,␣
 ↪auto_regression(vaccines_administered_list, 3))
mse_5, mape_5 = calculate_metrics(actual,␣
 ↪auto_regression(vaccines_administered_list, 5))

# Storing metrics in metrics dict
metrics_dict['la']['AR(3)'] = ["mse:"+str(mse_3), "mape:"+str(mape_3)]
metrics_dict['la']['AR(5)'] = ["mse:"+str(mse_5), "mape:"+str(mape_5)]
```

[10]:
```
# Part 2D (iii) and (iv) for MD state
# Getting predictions for LA using AR(3) and AR(5)
# NOTE : 3 missing dates in month of may for MD because of outlier removal -␣
 ↪hence prediction is starting from 24/05


# Processing MD DF for month of may
ar_df_md = df_md[(df_md.Date >= '2021-05-01') & (df_md.Date <= '2021-05-31')]
# Removing outlier
ar_df_md = ar_df_md[ar_df_md['Date'] != '2021-05-30']

# Creating lists from processed data
dates_list = list(ar_df_md['Date'])[0:28]
vaccines_administered_list = list(ar_df_md['Administered_daily'])[0:28]


actual = vaccines_administered_list[-7:]
```

```
dates = dates_list[-7:]

#Getting predictions and storing in output dataframe for MA
df_md_output['Date'] = dates_list[-7:]
df_md_output['Vaccines_administered'] = actual
df_md_output['AR(3)_predicted'] = auto_regression(vaccines_administered_list, 3)
df_md_output['AR(5)_predicted'] = auto_regression(vaccines_administered_list, 5)


mse_3, mape_3 = calculate_metrics(actual,␣
 ↪auto_regression(vaccines_administered_list, 3))
mse_5, mape_5 = calculate_metrics(actual,␣
 ↪auto_regression(vaccines_administered_list, 5))


# Storing metrics in metrics dict
metrics_dict['md']['AR(3)'] = ["mse:"+str(mse_3), "mape:"+str(mape_3)]
metrics_dict['md']['AR(5)'] = ["mse:"+str(mse_5), "mape:"+str(mape_5)]
```

[11]:
```
#Q2- d (iii) and (iv)

#Creating the EWMA class

class EWMA:
    def __init__(self, alpha):
        self.alpha = alpha
        self.y_t_hat = 0
        self.y_t = 0


    # training EWMA for y_t_hat
    def train(self, train_data):
      #From slide 9 of lecture slide 25 -  in practice, starting condition y_1␣
 ↪= y_1_hat
      y_t_hat = y_t = train_data[0]
      alpha = self.alpha

      len_data = len(train_data)
      for day_idx in range(1, len_data):
        y_t_hat = alpha * y_t + (1 - alpha) * y_t_hat
        y_t = train_data[day_idx]

      self.y_t = y_t
      self.y_t_hat = y_t_hat


    # predicting using EWMA
```

```
    def predict(self, test_data):
      y_t_hat = self.y_t_hat
      alpha = self.alpha
      y_t = self.y_t

      len_data = len(test_data)

      predictions = [None] * len_data

      for idx in range(len(test_data)):

        y_t_hat = alpha * y_t + (1 - alpha) * y_t_hat
        predictions[idx]  = round(y_t_hat)
        y_t = test_data[idx]


      return predictions
```

[12]:
```
# 2-d(iii) & # 2-d(iv) - for LA


# Processing LA DF for month of may
data_df_la = df_la[(df_la.Date >= '2021-05-01') & (df_la.Date <= '2021-05-31')]

# Creating lists from processed data
vaccines_administered_list = list(data_df_la['Administered_daily'])

train_data = vaccines_administered_list[0:21]
test_data = vaccines_administered_list[21:28]


# print('2D - part(iii) - EWMA ALPHA = 0.5') for LA
EWMA_obj = EWMA(0.5)
EWMA_obj.train(train_data)
df_la_output['EWMA(ALPHA = 0.5)'] = EWMA_obj.predict(test_data)

mse, mape = calculate_metrics(test_data, EWMA_obj.predict(test_data))
metrics_dict['la']['EWMA(0.5)'] = ["mse:"+str(mse), "mape:"+str(mape)]


# print('2D - part(iv) - EWMA ALPHA = 0.5') for LA
EWMA_obj = EWMA(0.8)
EWMA_obj.train(train_data)
df_la_output['EWMA(ALPHA = 0.8)'] = EWMA_obj.predict(test_data)

# Storing metrics in metrics dict
mse, mape = calculate_metrics(test_data, EWMA_obj.predict(test_data))
```

```
metrics_dict['la']['EWMA(0.8)'] = ["mse:"+str(mse), "mape:"+str(mape)]
```

[13]:
```
# 2-d(iii) & # 2-d(iv) - for MD


# Processing MD DF for month of may
data_df_md = df_md[(df_md.Date >= '2021-05-01') & (df_md.Date <= '2021-05-31')]
# Removing outlier
ar_df_md = ar_df_md[ar_df_md['Date'] != '2021-05-30']

# Creating lists from processed data
vaccines_administered_list = list(data_df_md['Administered_daily'])

train_data = vaccines_administered_list[0:21]
test_data = vaccines_administered_list[21:28]


# print('2D - part(iii) - EWMA ALPHA = 0.5') for MD
EWMA_obj = EWMA(0.5)
EWMA_obj.train(train_data)
df_md_output['EWMA(ALPHA = 0.5)'] = EWMA_obj.predict(test_data)

mse, mape = calculate_metrics(test_data, EWMA_obj.predict(test_data))
metrics_dict['md']['EWMA(0.5)'] = ["mse:"+str(mse), "mape:"+str(mape)]


# print('2D - part(iv) - EWMA ALPHA = 0.5') for mD
EWMA_obj = EWMA(0.8)
EWMA_obj.train(train_data)
df_md_output['EWMA(ALPHA = 0.8)'] = EWMA_obj.predict(test_data)

# Storing metrics in metrics dict
mse, mape = calculate_metrics(test_data, EWMA_obj.predict(test_data))
metrics_dict['md']['EWMA(0.8)'] = ["mse:"+str(mse), "mape:"+str(mape)]
```

[14]:
```
# FINAL OUTPUT

print("FOR LA state :-")
display(df_la_output)
print('\n')
print("MSE and MAPE metrics for LA:")
pp(metrics_dict['la'])

print('\n\n\n')

print("FOR MD state :-")
display(df_md_output)
```

```
print('\n')
print("MSE and MAPE metrics for MD:")
pp(metrics_dict['md'])
```

```
FOR LA state :-

         Date  Vaccines_administered  AR(3)_predicted  AR(5)_predicted  \
0  2021-05-23                11004.0      8081.868369      8524.503335
1  2021-05-24                 6556.0     10651.451216      9396.695053
2  2021-05-25                 3522.0     12811.004447     11129.065550
3  2021-05-26                 9777.0     13413.275833     13112.532528
4  2021-05-27                11978.0     11437.466581     14525.737258
5  2021-05-28                 9913.0      9899.035518     14210.445631
6  2021-05-29                11528.0     10385.345579     13434.960056

   EWMA(ALPHA = 0.5)  EWMA(ALPHA = 0.8)
0              15685              17043
1              13345              12212
2               9950               7687
3               6736               4355
4               8257               8693
5              10117              11321
6              10015              10195



MSE and MAPE metrics for LA:
{'AR(3)': ['mse:18059672.855516963', 'mape:57.789138300127895'],
 'AR(5)': ['mse:15972310.908749688', 'mape:56.73276213781896'],
 'EWMA(0.5)': ['mse:19249396.14285714', 'mape:57.99348249757969'],
 'EWMA(0.8)': ['mse:18536534.85714286', 'mape:52.579645500335474']}



FOR MD state :-

         Date  Vaccines_administered  AR(3)_predicted  AR(5)_predicted  \
0  2021-05-23                36172.0     39876.897247     43661.122982
1  2021-05-24                20705.0     36197.925513     36483.659233
2  2021-05-25                22104.0     50010.540879     38514.675342
3  2021-05-27                40867.0     40059.369136     52053.996249
4  2021-05-28                28517.0     31125.235695     19024.839532
5  2021-05-29                52157.0     37439.610189     30565.076154
6  2021-05-31                28988.0     34772.133106     37858.944058

   EWMA(ALPHA = 0.5)  EWMA(ALPHA = 0.8)
0              36234              37195
```

```
1               28470               24003
2               25287               22484
3               33077               37190
4               30797               30252
5               41477               47776
6               20968                9923
```

```
MSE and MAPE metrics for MD:
{'AR(3)': ['mse:184292135.44386172', 'mape:38.65914614335676'],
 'AR(5)': ['mse:190645447.81307626', 'mape:43.40205765609294'],
 'EWMA(0.5)': ['mse:392595726.5714286', 'mape:1306.1820832399292'],
 'EWMA(0.8)': ['mse:538684146.4285715', 'mape:1508.2119276584729']}
```

[15]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic &> /dev/null
!jupyter nbconvert --to pdf /content/drive/Shareddrives/CSE544_Project/part_d/
 ↪part_d.ipynb  &> /dev/null
```