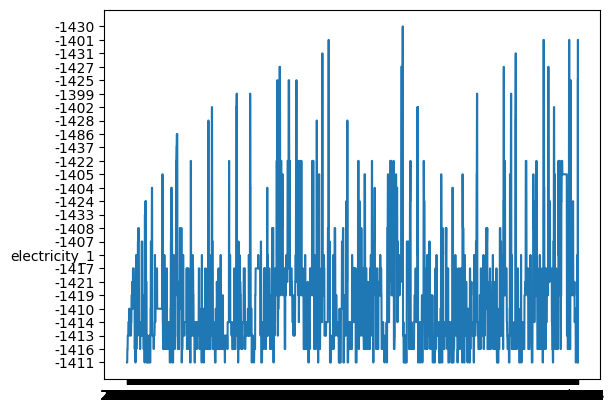
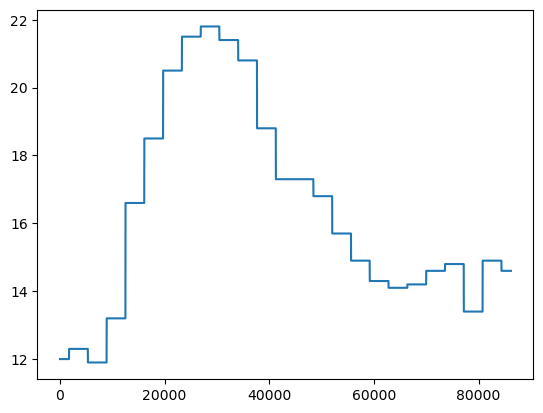
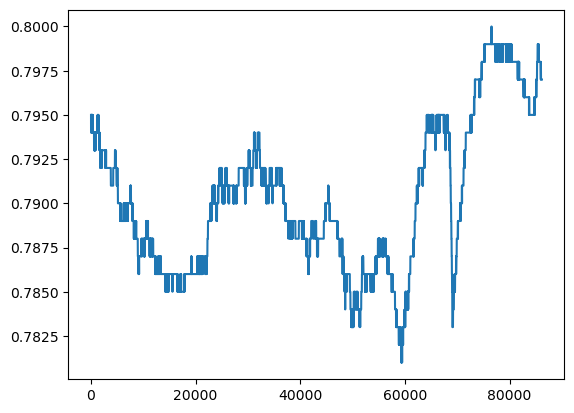
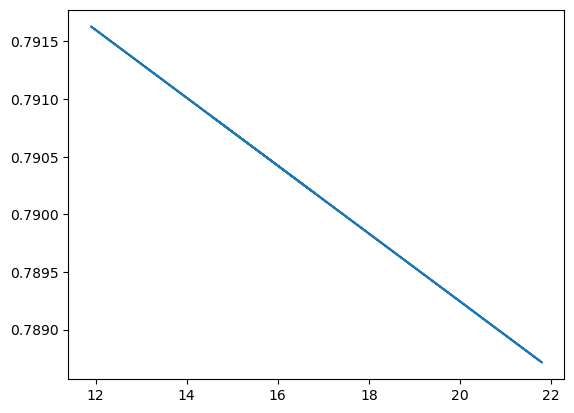
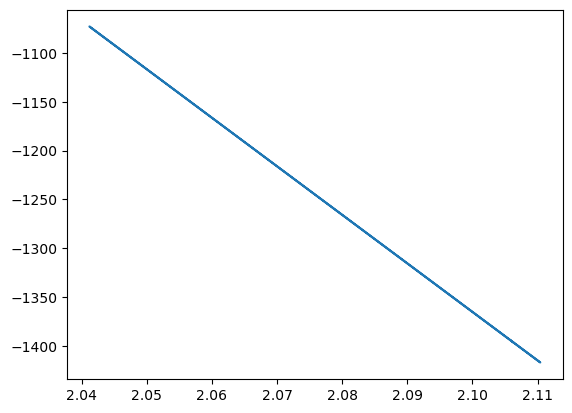
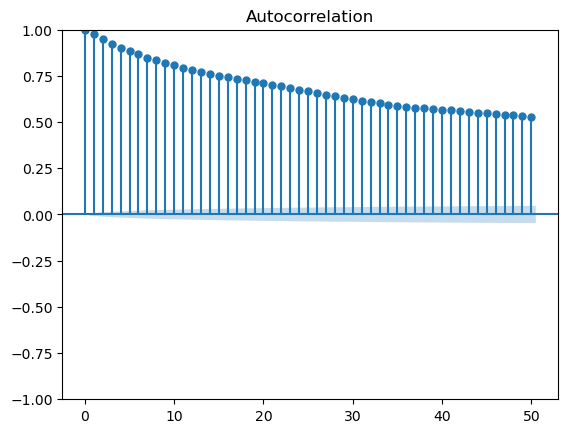
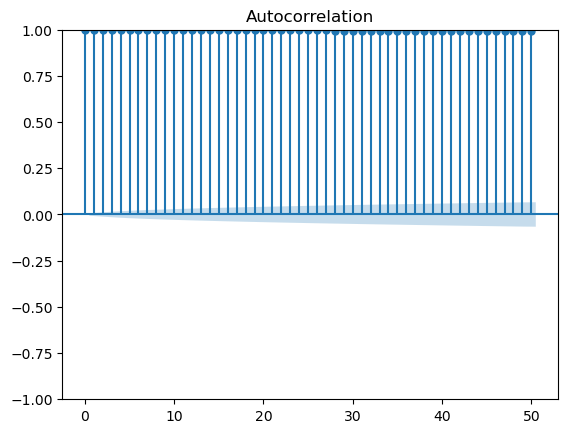
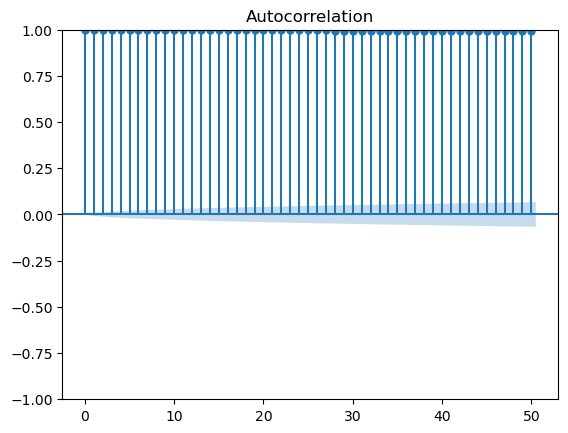
**CNCITY Dataset - Exploratory Data Analysis  
(February + April)**

1. **Parsing and tabulating the data**
   1. Done through Cassandra\_query and other Python packages

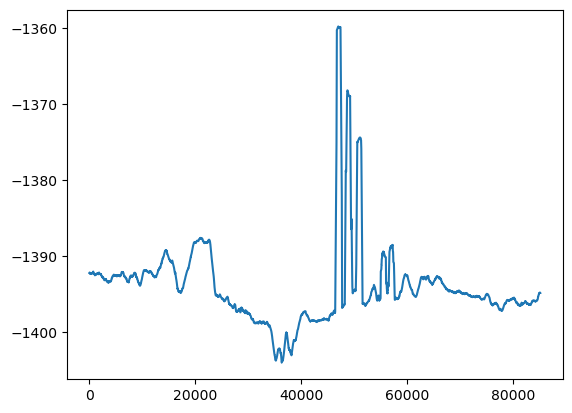
(20240603)

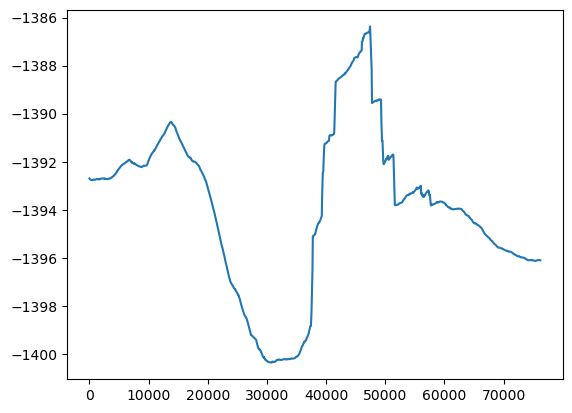
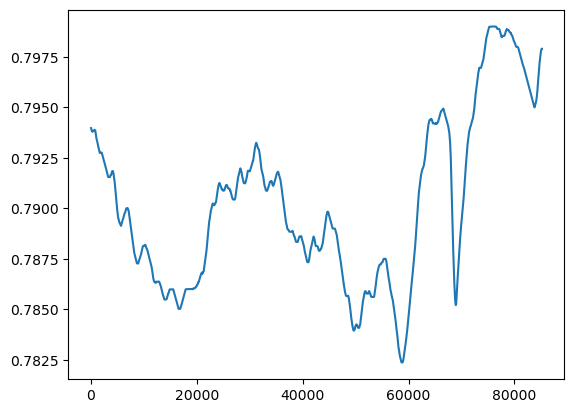
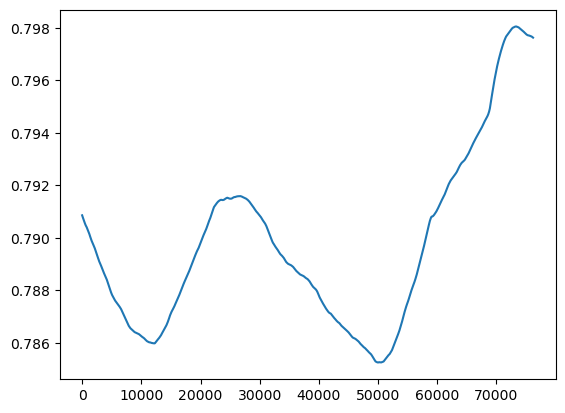
1. **Plotting each variable (stationarity, moving average, seasonality, etc)**
   1. Electricity\_1: seems relatively stationary, does not display seasonality 
   2. Outside\_temperature: ranges from 12 to 22 degrees Celsius
   3. Pressure\_1: 
2. **Linear Regression**
   1. Outside\_temperature and pressure\_1
      1. Split the data frames into train and test sets
      2. Train the model using x\_train and y\_train, then run it on x\_test to evaluate and further improve accuracy
      3. 
      4. Temperature and pressure are negatively correlated
      5. y = -0.0002944x + 0.795132
   2. Electricity\_1 and pressure\_2
      1. 
3. **Autocorrelation (Test for stationarity)**
   1. 
      1. Electricity\_1 - non-stationary
      2. Blue-shaded region: confidence interval
      3. **Suggests that the dataset might be non-stationary** 
         1. Slow decay of autocorrelation

* The autocorrelation plot tends to drop quickly in a stationary time series  
  + - 1. Persistent correlations
* Our dataset displays higher autocorrelation values at longer lags
  + 1. Need to make the series stationary before analyzing it (moving average method, rolling average, etc)
  1. **Pressure\_1 - non-stationary (log transformation did not work)**

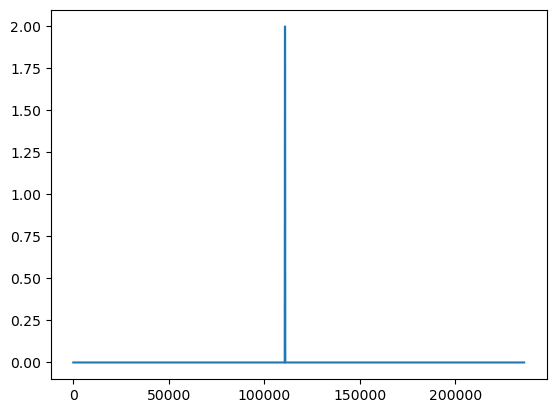
 (after applying log transformation)

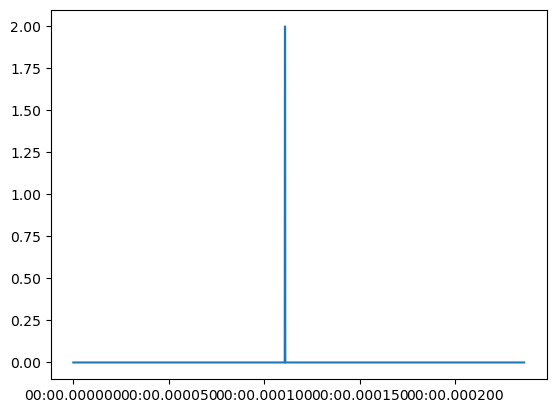
1. **Moving average (Rolling average)**
   1. Electricity\_1 (window = 1,000)



* 1. Window = 10,000  
     
     1. The steep rise and fall between indices 30,000 and 50,000 indicate periods of higher volatility
  2. Pressure\_1 (rolling average)
     1. Window = 1,000  
        
     2. Window = 10,000  
          
        

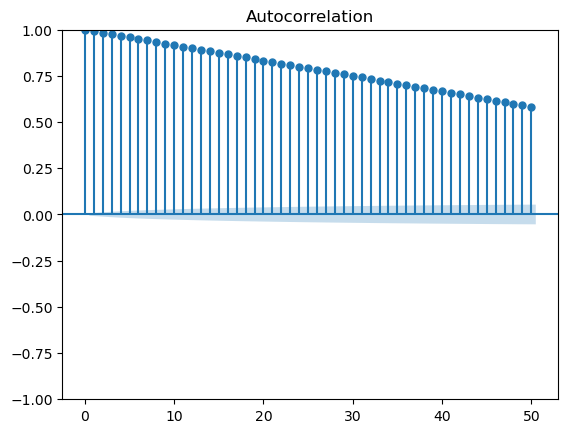
Plotted ‘is\_maintenance’ versus Time



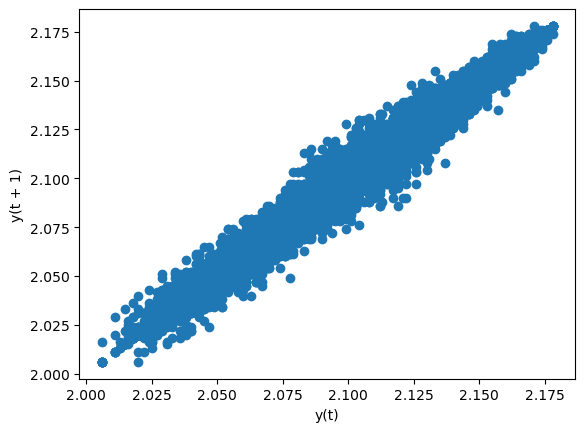


(maintenance\_df → the indices are different)

Autocorrelation plot (is\_maintenance)



Pressure\_2: (lag plot)



**6/13/2024:**

df['pressure\_1'].corr(df['pressure\_1'].shift(1000)): autoregression, lag = 1,000

Output: 0.9184823349674819

df['pressure\_1'].corr(df['pressure\_1'].shift(1000)): autoregression, lag = 1,000

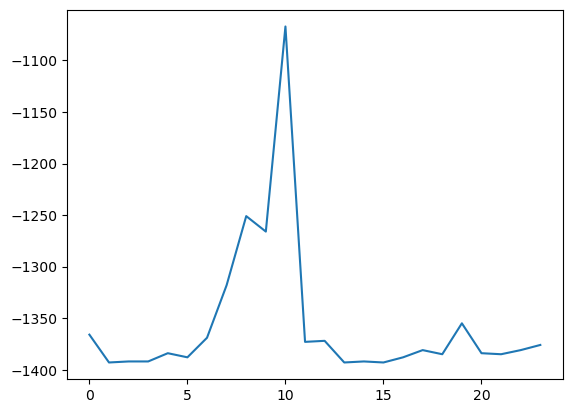
Devised a different approach to preprocessing the data:

* Wrote a function called ‘**process\_dataframe**’ that takes a dataframe and returns a new one.
* Our dataset has a column called ‘Time’ that starts at 0 and ends at 235959 (midnight)
* ‘Process\_dataframe’ takes the dataset and groups the data by hour based on the ‘time’ column
* Then it computes the mean, median, mode, or max of the other columns for each hour group
* After that it creates a new dataframe with all these values

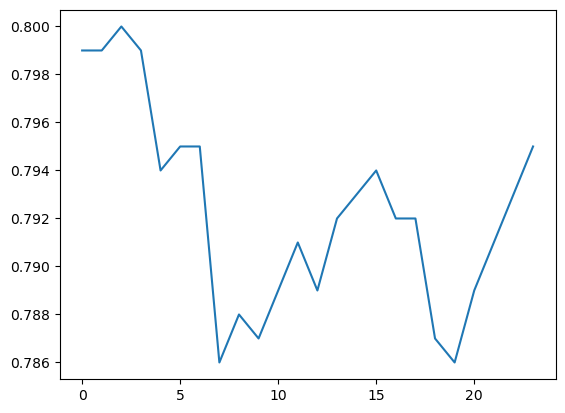
Result\_df: our new dataset (consolidated), has a new column called ‘Hour’

**Visualization: (graphs)**

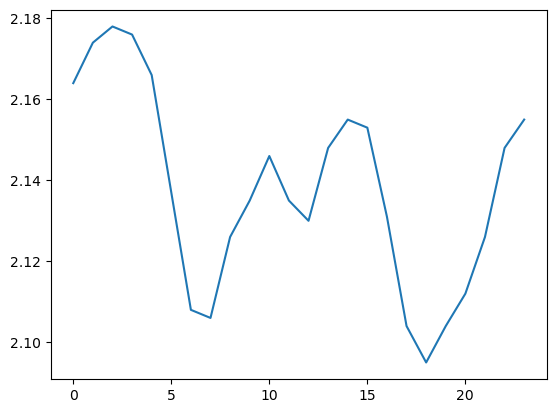
1. **Electricity\_1**



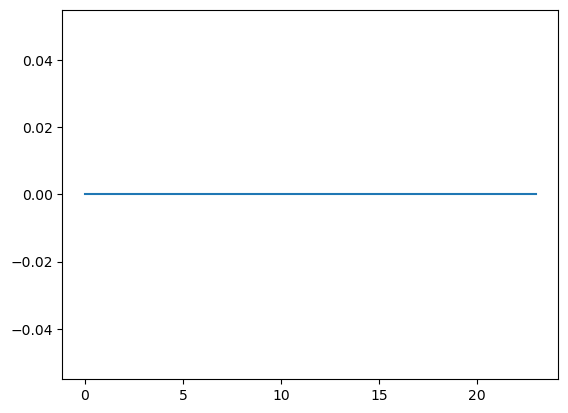
1. **Pressure\_1**

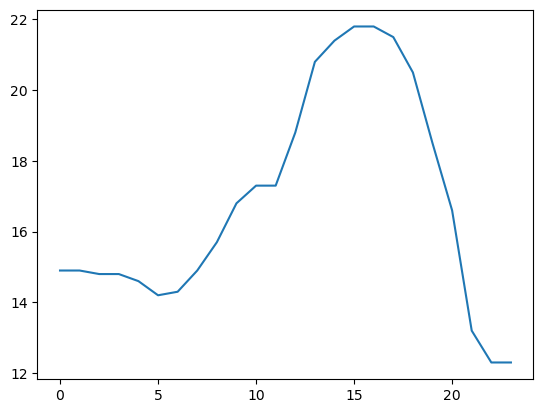
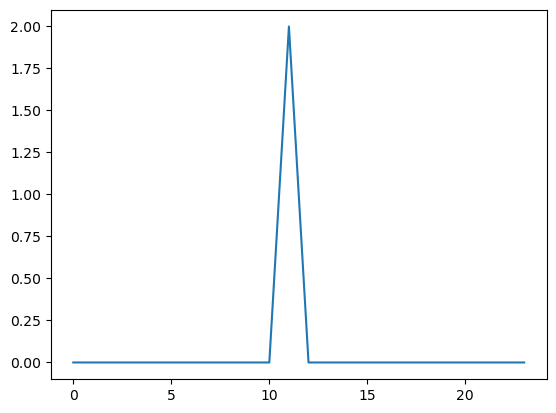
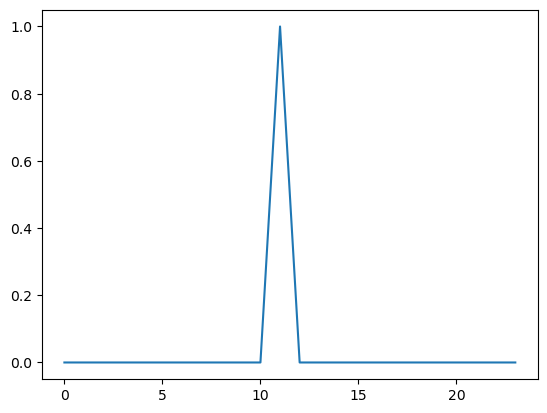
****

1. **Pressure\_2**

****

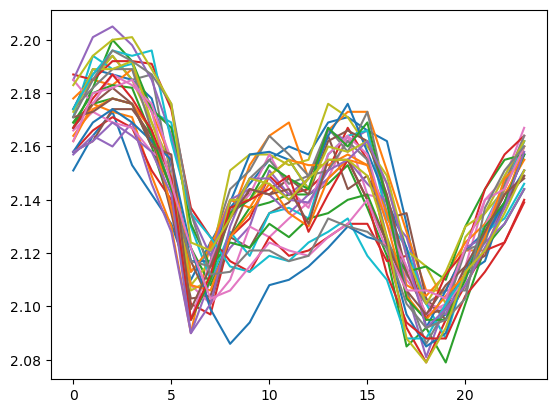
1. **Electricity\_2**

****

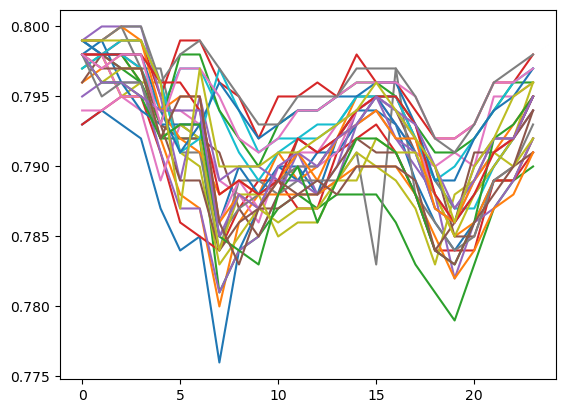
1. **Outside\_temperature  
   **
2. **Is\_maintenance  
   **
3. **Door\_open  
   **

**6/17/2024:**

**Pressure\_2:**

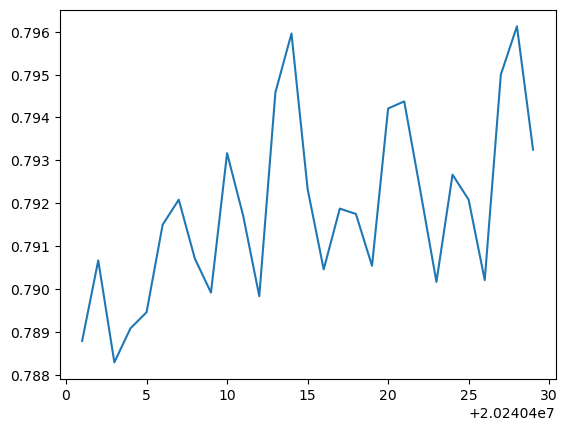
****

**Pressure\_1:**

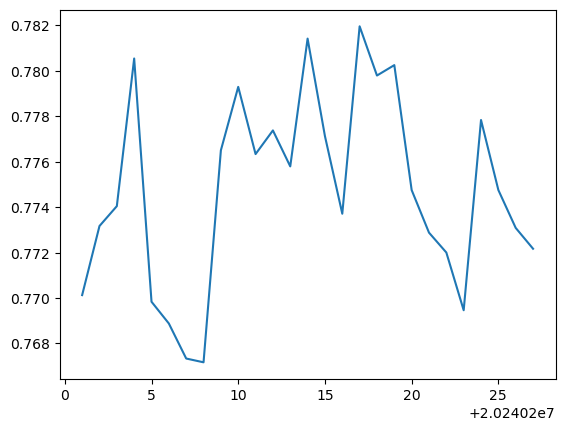


Pressure\_1 (mean): April

* Queried/processed a month’s worth of data

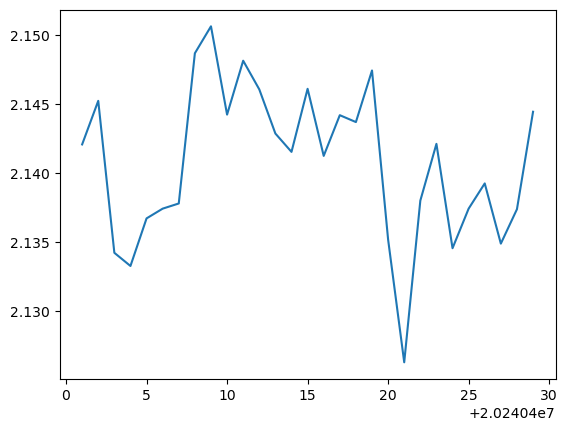


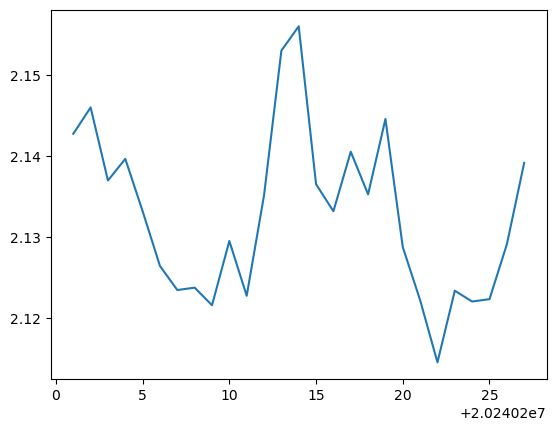
Pressure\_1 (mean): February

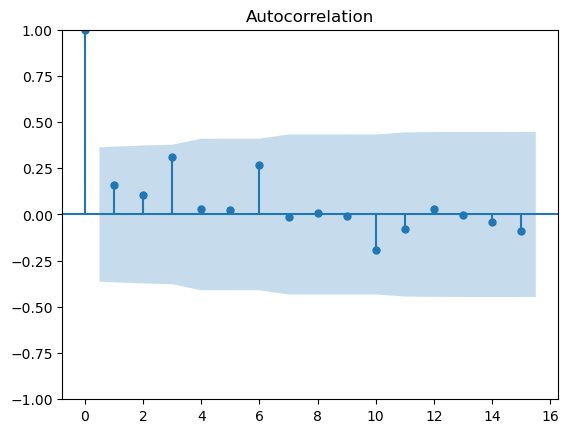


Did the same for ‘Pressure\_2’

**April:**



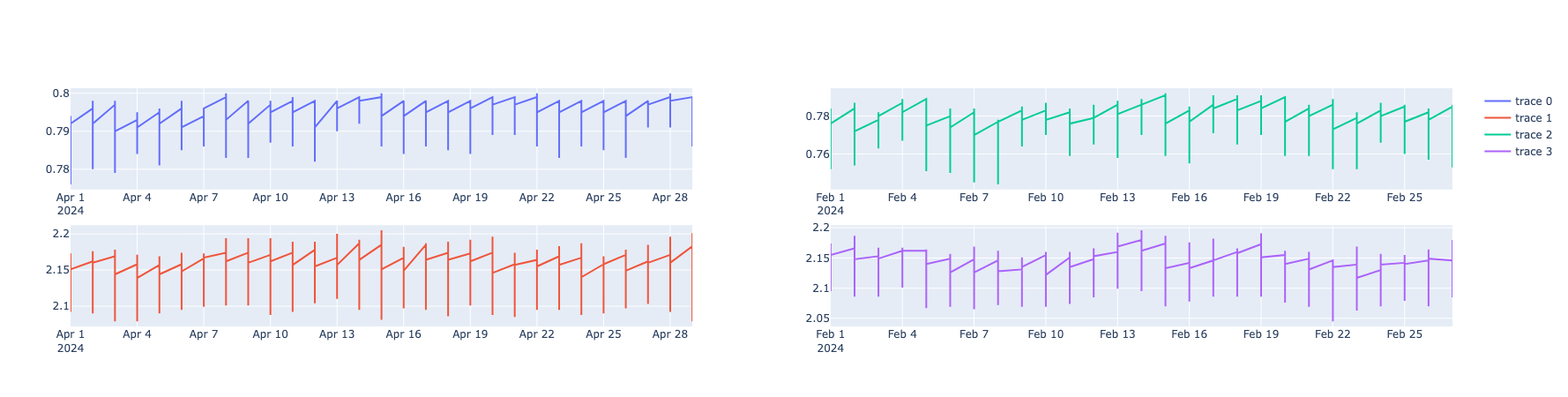
**February: **

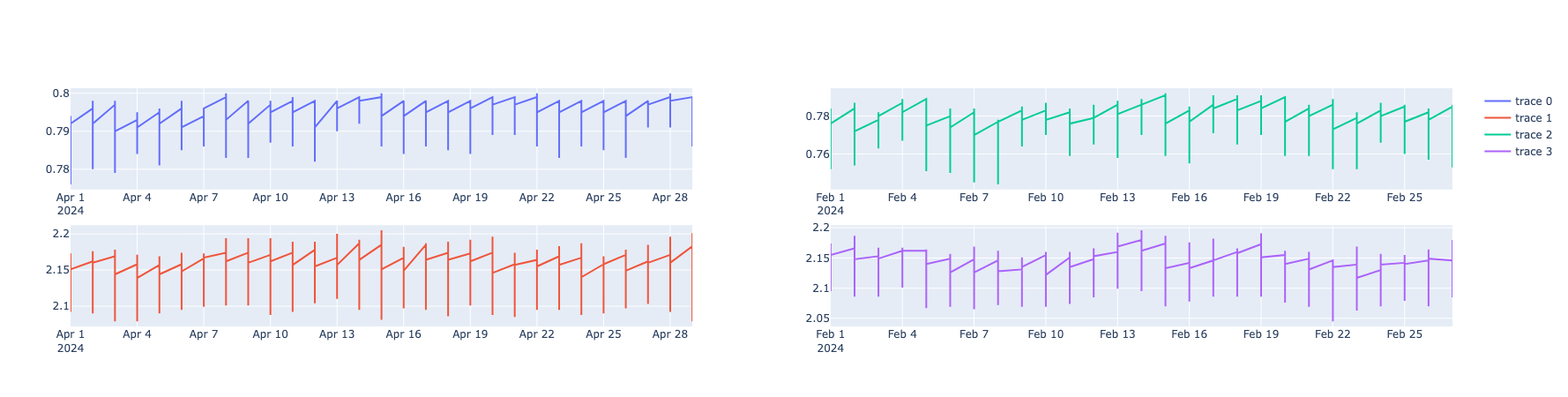
****

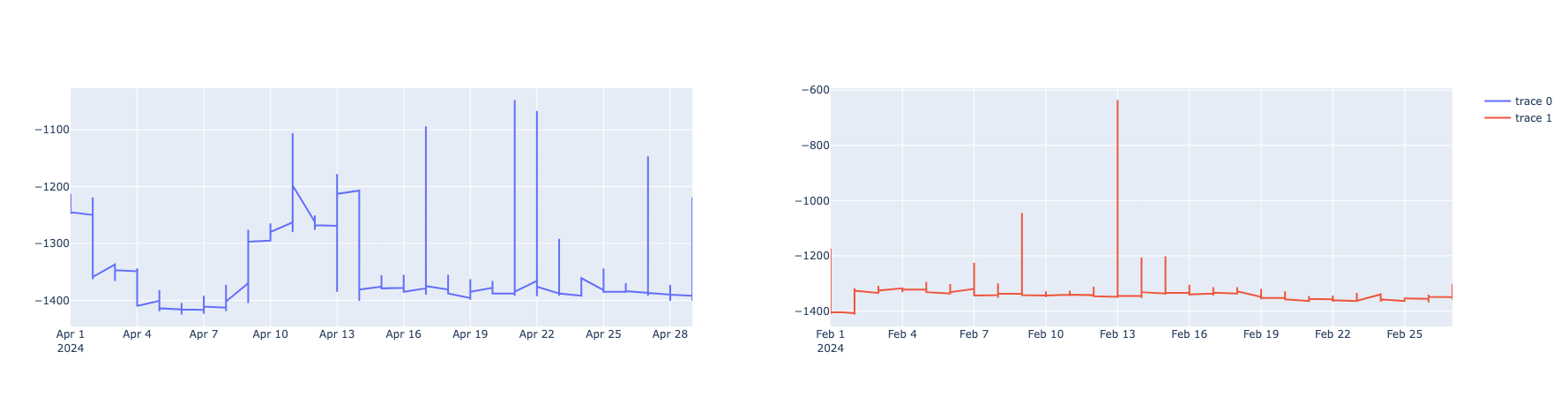
**Autocorrelation plot (variable = ‘pressure\_1’, max, month = April)**

**Lag 1 to 15:**

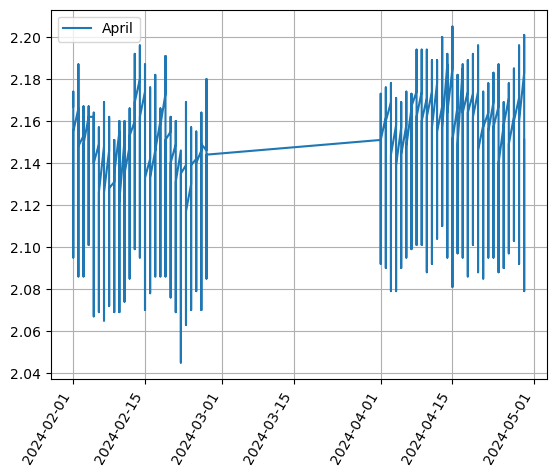
Lag 1: displays a significant spike above the confidence interval, suggests a significant positive autocorrelation

Electricity\_1 (April and February) 





February + April



The data from March seems to be embedded with irregularities and exceptions:

processed\_dataframe\_list\_4 = []

start\_date = 20240301

while start\_date < 20240330:

df\_2 = query(start\_date, 1)

new\_df\_2 = process\_dataframe(df\_2, 'max')

processed\_dataframe\_list\_4.append(new\_df\_2)

start\_date+=1

del df\_2

del new\_df\_2

final\_df\_mar = pd.concat(processed\_dataframe\_list\_4, axis =0)

final\_df\_mar.head()

~/anaconda3/lib/python3.11/site-packages/pandas/core/frame.py in ?(self, by, axis, ascending, inplace, kind, na\_position, ignore\_index, key)

**6754** elif len(by):

**6755** # len(by) == 1

...

**1779**

**1780** # Check for duplicates

**1781** if values.ndim > 1:

KeyError: 'time'

MongooseAI: (brevity is key) – for future interviews

Week 1,2: Titanic EDA (Kaggle): linear regression, logistic regression, decision trees, randomforestclassifier: used different classifiers to predict the surviving passengers

Week 3,4: CNCITY, Titanic

* CNCITY: (include a brief intro explaining the project, )
* Queried, processed (consolidated), and concatenated a month’s worth of data
* Measurement of misalignment, identifying outliers

Week 5: CNCITY

* Measurement of misalignment, ‘pressure\_1’
* Subplots
* Identifying false alarms and missed detections using the event data (db\_1 and IoT\_1 have different # of columns and rows)