ARIMA

IMPORT LIBRARY

statsmodels: It provides classes and functions for estimating and testing statistical models. This includes linear and non-linear regression models, time series analysis, and various statistical tests.

pmdarima: This library is specifically designed for time series analysis and provides an implementation of the AutoRegressive Integrated Moving Average (ARIMA) model. It's built on top of statsmodels.

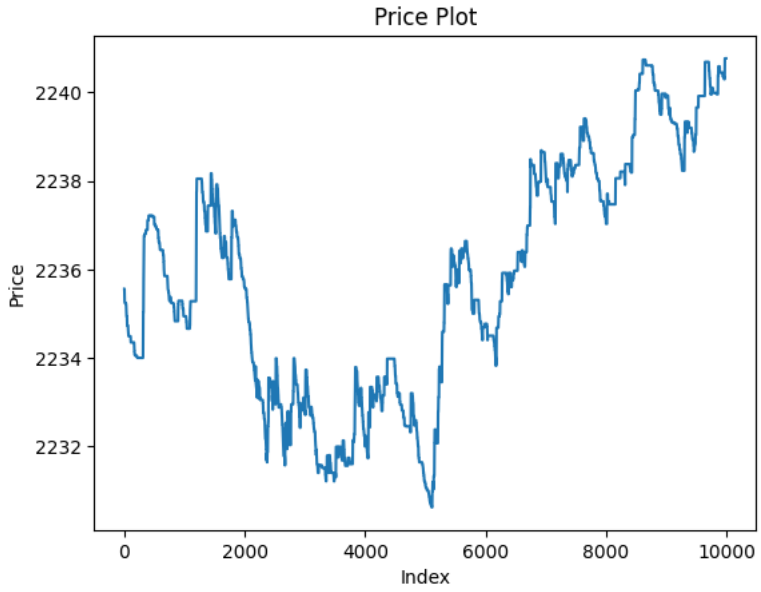
```
!pip install pmdarima
!pip install statsmodels
!pip install pyramid
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 8.9 MB/s eta 0:00:00
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.7)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.4)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (23.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.2
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.4)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.4->statsmodels>=0.13.2->pmdarima) (1.16.
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (0.14.1)
Requirement already satisfied: numpy<2,>=1.18 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.23.5)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.11.4)
Requirement already satisfied: pandas!=2.1.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.5.3)
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (0.5.4)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (23.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.0->statsmodels
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.0->statsmodels) (2023.3.
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.4->statsmodels) (1.16.0)
Collecting pyramid
  Downloading pyramid-2.0.2-py3-none-any.whl (247 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.3/247.3 kB 2.2 MB/s eta 0:00:00
Collecting hupper>=1.5 (from pyramid)
  Downloading hupper-1.12-py3-none-any.whl (22 kB)
Collecting plaster (from pyramid)
  Downloading plaster-1.1.2-py2.py3-none-any.whl (11 kB)
Collecting plaster-pastedeploy (from pyramid)
  Downloading plaster_pastedeploy-1.0.1-py2.py3-none-any.whl (7.8 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from pyramid) (67.7.2)
Collecting translationstring>=0.4 (from pyramid)
  Downloading translationstring-1.4-py2.py3-none-any.whl (15 kB)
Collecting venusian>=1.0 (from pyramid)
  Downloading venusian-3.1.0-py3-none-any.whl (13 kB)
Collecting webob>=1.8.3 (from pyramid)
  Downloading WebOb-1.8.7-py2.py3-none-any.whl (114 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.0/115.0 kB 15.1 MB/s eta 0:00:00
Collecting zope.deprecation>=3.5.0 (from pyramid)
  Downloading zope.deprecation-5.0-py3-none-any.whl (10 kB)
Collecting zope.interface>=3.8.0 (from pyramid)
  Downloading zope.interface-6.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (247
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.1/247.1 kB 15.9 MB/s eta 0:00:00
Collecting PasteDeploy>=2.0 (from plaster-pastedeploy->pyramid)
  Downloading PasteDeploy-3.1.0-py3-none-any.whl (16 kB)
Installing collected packages: translationstring, zope.interface, zope.deprecation, webob, venusian, plaster, PasteDeploy, hupper, plas
Successfully installed PasteDeploy-3.1.0 hupper-1.12 plaster-1.1.2 plaster-pastedeploy-1.0.1 pyramid-2.0.2 translationstring-1.4 venusi
```

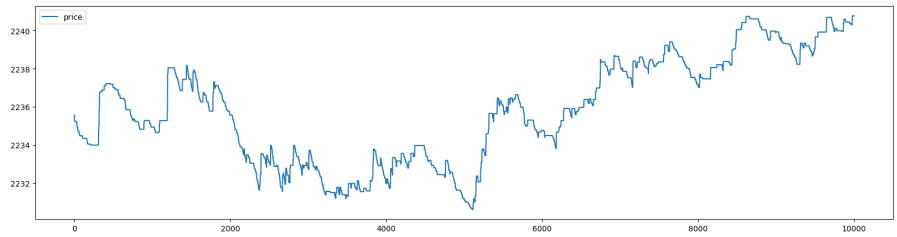
```
from dateutil.parser import parse
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
from pmdarima import auto_arima
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm


import pyramid
import pmdarima
from pmdarima.arima import auto_arima

import warnings
# Filter out Arrow-related RuntimeWarnings
warnings.filterwarnings("ignore", category=RuntimeWarning, module="pyarrow")
```

IMPORT DATA and PREPARE

```
# path = "Path ที่นำทางไปสู่ไฟล์ Dataset ที่ต้องการ"
cols = ['id','price','qty','amount','period','bid','offer']
#path = "/content/ETHUSDT-trades-2024-01-04.csv"
#path = "/content/ETHUSDT-trades-2024-01-05.csv"
path = "/content/ETHUSDT-trades-2024-01-06.csv"
df = pd.read_csv(path, names=cols)

df['period'] = pd.to_datetime(df['period'], unit='ms', errors='coerce')
df = df[df['bid'] == True]
df = df[ ['id', 'price'] ]
df = df.tail(10000).reset_index(drop=True)

df
#Plot the 'price' column
plt.plot(df['price'])
plt.xlabel('Index')
plt.ylabel('Price')
plt.title('Price Plot')
plt.show()
```



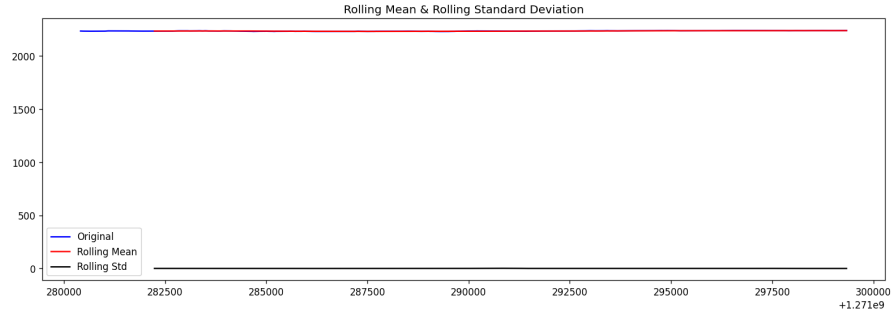

```
df['price'].plot(figsize = (20, 5), legend = True);
```




```
df = df.set_index('id')
```

Stationary Test

```
rolling_mean = df.price.rolling(window = 1000).mean()
rolling_std = df.price.rolling(window = 1000).std()
fig, ax = plt.subplots(1, 1, figsize=(16,5), dpi= 120)
plt.plot(df, color = 'blue', label = 'Original')
plt.plot(rolling_mean, color = 'red', label = 'Rolling Mean')
plt.plot(rolling_std, color = 'black', label = 'Rolling Std')
plt.legend(loc = 'best')
plt.title('Rolling Mean & Rolling Standard Deviation')
plt.show()

df.price
```




```
id
1271280414    2235.56
1271280415    2235.56
1271280416    2235.38
1271280417    2235.31
1271280418    2235.29
               ...
1271299332    2240.77
1271299333    2240.77
1271299334    2240.77
1271299335    2240.77
1271299337    2240.77
Name: price, Length: 10000, dtype: float64
```

```
result = adfuller(df['price'])
print('ADF Statistic: {}'.format(result[0]))
print('p-value: {}'.format(result[1]))
print('Critical Values:')
for key, value in result[4].items():
    print('\t{}: {}'.format(key, value))
```

```
 ADF Statistic: -1.0815754123544858
 p-value: 0.7223632153581998
 Critical Values:
        1%: -3.4310047528604803
        5%: -2.861829361784287
        10%: -2.5669240221402583
```

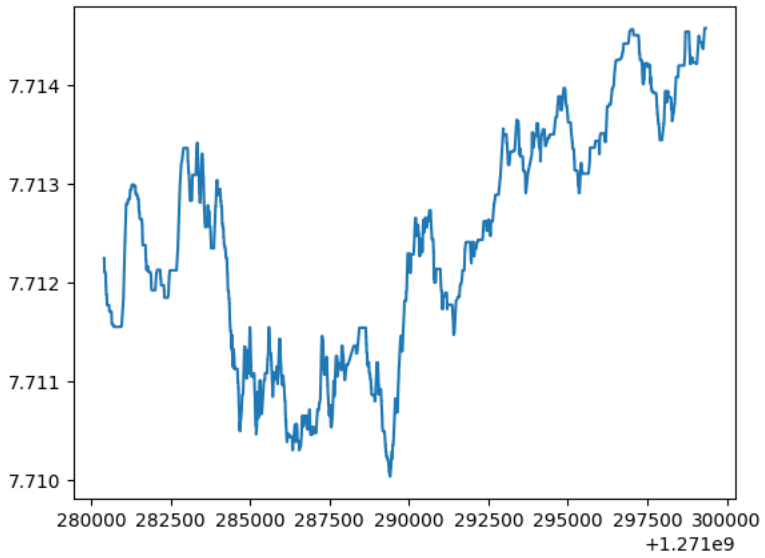
```
def get_stationarity(timeseries):

    # rolling statistics
    rolling_mean = timeseries.rolling(window=1000).mean()
    rolling_std = timeseries.rolling(window=1000).std()
    #rolling_mean = timeseries.rolling(window=12).mean()
    #rolling_std = timeseries.rolling(window=12).std()

    # rolling statistics plot
    original = plt.plot(timeseries, color='blue', label='Original')
    mean = plt.plot(rolling_mean, color='red', label='Rolling Mean')
    std = plt.plot(rolling_std, color='black', label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    # Dickey-Fuller test:
    result = adfuller(timeseries['price'])
    print('ADF Statistic: {}'.format(result[0]))
    print('p-value: {}'.format(result[1]))
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t{}: {}'.format(key, value))
```

ลบด้วยค่าเฉลี่ยเคลื่อนที่

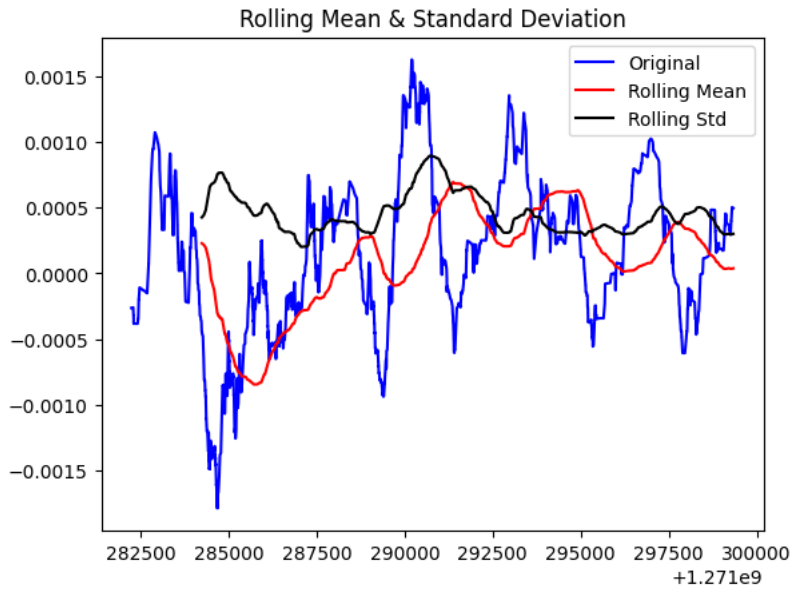
```
df_log = np.log(df)
plt.plot(df_log)

df_log
```

id	price
1271280414	7.712247
1271280415	7.712247
1271280416	7.712167
1271280417	7.712135
1271280418	7.712126
...	...
1271299332	7.714575
1271299333	7.714575
1271299334	7.714575
1271299335	7.714575
1271299337	7.714575

10000 rows × 1 columns




```
rolling_mean = df_log.rolling(window=1000).mean()
df_log_minus_mean = df_log - rolling_mean
df_log_minus_mean.dropna(inplace=True)
get_stationarity(df_log_minus_mean)

df_log_minus_mean
```




```
ADF Statistic: -3.0318585159883646
p-value: 0.03202443753955612
Critical Values:
        1%: -3.4310775239640425
        5%: -2.8618615183417497
        10%: -2.5669411391862456
```

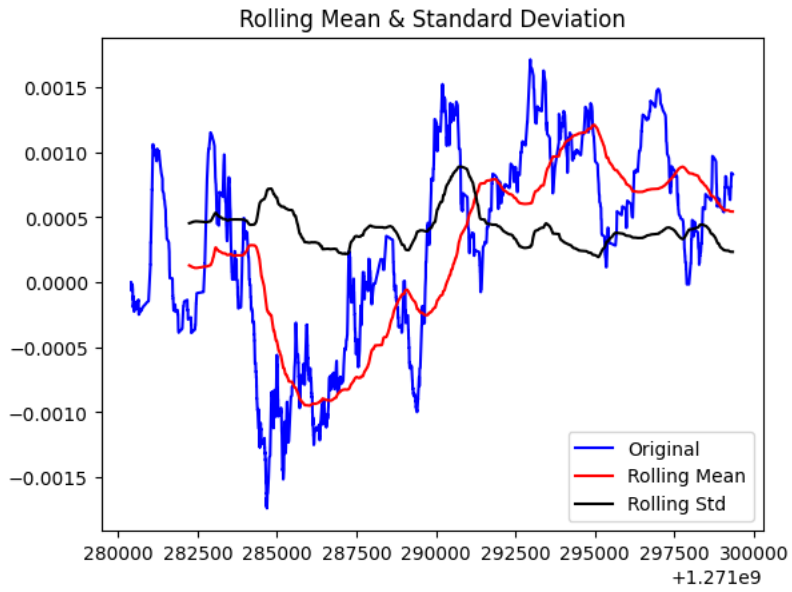
id	price
1271282239	-0.000263
1271282241	-0.000263
1271282242	-0.000263
1271282251	-0.000263
1271282252	-0.000262
...	...
1271299332	0.000497
1271299333	0.000497
1271299334	0.000496
1271299335	0.000496
1271299337	0.000495

9001 rows × 1 columns

อย่างที่เราเห็นหลังจากลบค่าเฉลี่ยค่าเฉลี่ยเคลื่อนที่และส่วนเบี่ยงเบนมาตรฐานจะเป็นแนวนอนโดยประมาณ ค่า p ต่ำกว่าเกณฑ์ 0.05 และสถิติ ADF ใกล้เคียงกับค่าวิกฤต ดังนั้นอนุกรมเวลาจึงเป็น stationaryการใช้การสลายตัวแบบเลขชี้กำลังเป็นอีกวิธีหนึ่งในการแปลงอนุกรมเวลาให้เป็นแบบคงที่

take_log

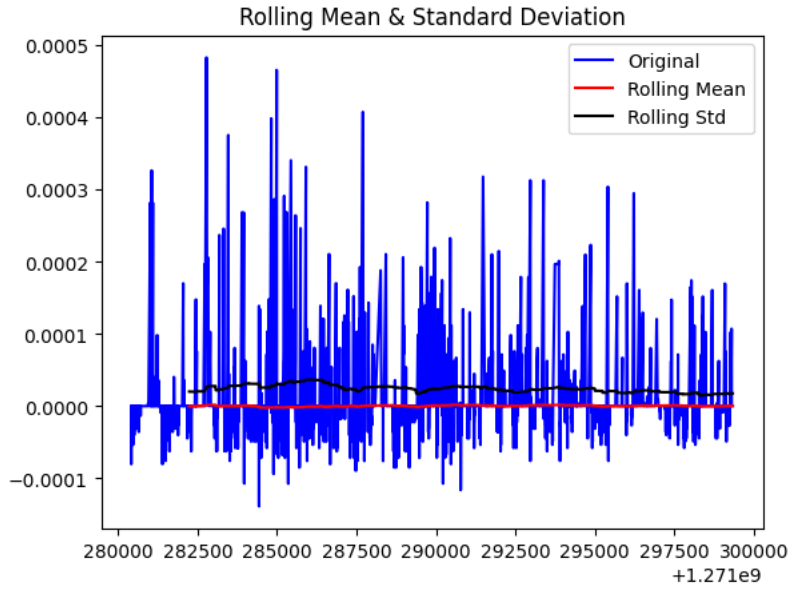
```
rolling_mean_exp_decay = df_log.ewm(halflife=1000, min_periods=0, adjust=True).mean()
df_log_exp_decay = df_log - rolling_mean_exp_decay
df_log_exp_decay.dropna(inplace=True)
get_stationarity(df_log_exp_decay)
```




```
ADF Statistic: -2.4957969734640804
p-value: 0.11650031864051996
Critical Values:
        1%: -3.4310047528604803
        5%: -2.861829361784287
        10%: -2.5669240221402583
```

ลบด้วยค่าก่อนหน้า

```
df_log_shift = df_log - df_log.shift()
df_log_shift.dropna(inplace=True)
get_stationarity(df_log_shift)

df.price
```




```
ADF Statistic: -27.3755995110716
p-value: 0.0
Critical Values:
        1%: -3.4310047528604803
        5%: -2.861829361784287
        10%: -2.5669240221402583
id
1271280414    2235.56
1271280415    2235.56
1271280416    2235.38
1271280417    2235.31
1271280418    2235.29
               ...
1271299332    2240.77
1271299333    2240.77
1271299334    2240.77
1271299335    2240.77
1271299337    2240.77
Name: price, Length: 10000, dtype: float64
```

Analysis for the ARIMA model

STATEMODEL.API

```
# 1,1,2 ARIMA Model
df.price
model = sm.tsa.ARIMA(df.price, order=(1,1,2))
result = model.fit()
print(result.summary())
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
                               SARIMAX Results
==============================================================================
Dep. Variable:                  price   No. Observations:                10000
Model:                 ARIMA(1, 1, 2)   Log Likelihood               14995.643
Date:                Sun, 07 Jan 2024   AIC                         -29983.286
Time:                        19:29:47   BIC                         -29954.445
Sample:                             0   HQIC                        -29973.523
                                - 10000
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8513      0.018     48.006      0.000       0.817       0.886
ma.L1         -0.7242      0.018    -40.300      0.000      -0.759      -0.689
```

```
ma.L2         -0.0347      0.007     -4.754      0.000      -0.049      -0.020
sigma2         0.0029   5.92e-06    492.716      0.000       0.003       0.003
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):           4668071.58
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               0.50   Skew:                             8.15
Prob(H) (two-sided):                  0.00   Kurtosis:                       107.59
===================================================================================

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

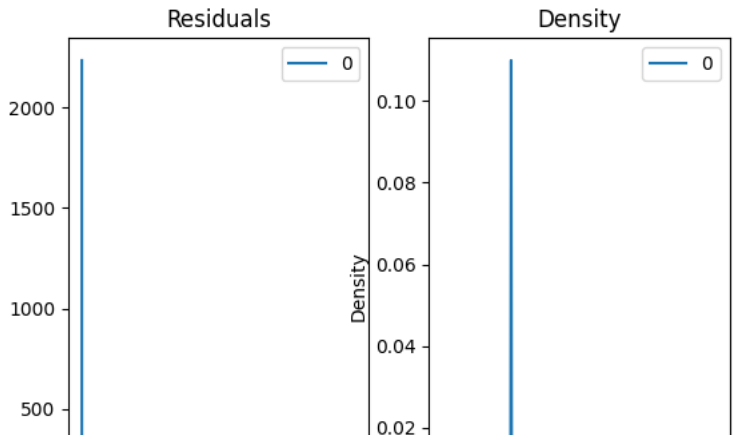
```
df.price
```

```
id
1271280414    2235.56
1271280415    2235.56
1271280416    2235.38
1271280417    2235.31
1271280418    2235.29
               ...
1271299332    2240.77
1271299333    2240.77
1271299334    2240.77
1271299335    2240.77
1271299337    2240.77
Name: price, Length: 10000, dtype: float64
```

```
# 1,1,1 ARIMA Model
model = sm.tsa.ARIMA(df.price, order=(1,1,1))
result = model.fit()
print(result.summary())
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
                               SARIMAX Results
==================================================================================
Dep. Variable:                   price   No. Observations:                10000
Model:                 ARIMA(1, 1, 1)   Log Likelihood               14991.593
Date:                Sun, 07 Jan 2024   AIC                          -29977.185
Time:                        19:32:30   BIC                          -29955.554
Sample:                             0   HQIC                         -29969.863
                                - 10000
Covariance Type:                  opg
==================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
ar.L1          0.8085      0.014     59.296      0.000       0.782       0.835
ma.L1         -0.6987      0.016    -44.109      0.000      -0.730      -0.668
sigma2         0.0029   5.94e-06    491.426      0.000       0.003       0.003
===================================================================================
Ljung-Box (L1) (Q):                   2.80   Jarque-Bera (JB):           4641423.98
Prob(Q):                              0.09   Prob(JB):                         0.00
Heteroskedasticity (H):               0.50   Skew:                             8.14
Prob(H) (two-sided):                  0.00   Kurtosis:                       107.29
===================================================================================

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

```
# Plot residual errors
residuals = pd.DataFrame(result.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()
```



pmdarima

ใช้ pmdarima ซึ่งข้อดีของวิธีนี้คือ เราไม่ต้องมาเดา order ให่ใส่ order ต่ำสุดสูงสุดไปแทน model จะเลือกให้เราเองอัตโนมัติ

```
# Fit auto_arima function to ETH price dataset
stepwise_fit = auto_arima(df['price'], start_p = 1, start_q = 1,
                          max_p = 2, max_q = 2, m = 4,
                          start_P = 0, seasonal = True,
                          d = None, D = 1, trace = True,
                          error_action ='ignore', # we don't want to know if an order does not work
                          suppress_warnings = True, # we don't want convergence warnings
                          stepwise = True)         # set to stepwise
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,1,1)[4] intercept   : AIC=inf, Time=41.26 sec
 ARIMA(0,0,0)(0,1,0)[4] intercept   : AIC=-12754.067, Time=0.99 sec
 ARIMA(1,0,0)(1,1,0)[4] intercept   : AIC=-26481.167, Time=11.59 sec
 ARIMA(0,0,1)(0,1,1)[4] intercept   : AIC=inf, Time=25.48 sec
 ARIMA(0,0,0)(0,1,0)[4]             : AIC=-12753.297, Time=0.50 sec
 ARIMA(1,0,0)(0,1,0)[4] intercept   : AIC=-24134.408, Time=0.92 sec
 ARIMA(1,0,0)(2,1,0)[4] intercept   : AIC=-27376.757, Time=25.63 sec
 ARIMA(1,0,0)(2,1,1)[4] intercept   : AIC=inf, Time=44.31 sec
 ARIMA(1,0,0)(1,1,1)[4] intercept   : AIC=inf, Time=41.88 sec
 ARIMA(0,0,0)(2,1,0)[4] intercept   : AIC=-13272.149, Time=5.99 sec
 ARIMA(2,0,0)(2,1,0)[4] intercept   : AIC=-27605.590, Time=32.46 sec
 ARIMA(2,0,0)(1,1,0)[4] intercept   : AIC=-26685.271, Time=19.83 sec
 ARIMA(2,0,0)(2,1,1)[4] intercept   : AIC=inf, Time=50.39 sec
 ARIMA(2,0,0)(1,1,1)[4] intercept   : AIC=inf, Time=37.30 sec
 ARIMA(2,0,1)(2,1,0)[4] intercept   : AIC=-27741.043, Time=45.24 sec
```