

Access test to data historian

this is to use python dataframe

```
In [1]: import dask
import pandas as pd
import numpy as np
import sys
if sys.version_info[0] < 3:
    from StringIO import StringIO
else:
    from io import StringIO

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn; seaborn.set()
```

```
In [2]: import requests

url = "http://localhost:8080/api/grafana/export/csv"

payload = """
{
  "range": {
    "from": "2019-11-27T00:00:00.000Z",
    "to": "2019-11-29T23:59:00.000Z"
  },
  "targets": [{
    "target": "ack@2e46118f-c68d-40f2-8a2a-28586f2584ae"
  }],
  "format": "csv",
  "maxDataPoints": 550
}"""
headers = {
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data = payload)

TESTDATA = StringIO(response.text)

df = pd.read_csv(TESTDATA, sep=",")

df['timestamp']=df['date']
df['date'] = pd.to_datetime(df['date'], unit='ms')
#df['day'] =df['date'].dt.day
#df['hour'] =df['date'].dt.hour
#df['metric_type']=df['metric'].str.extract(r'(\S+)\..*')
#df['metric_id']=df['metric'].str.extract(r'.*?(\S+)')
df.index = df['date']

values=df['value']

df
```

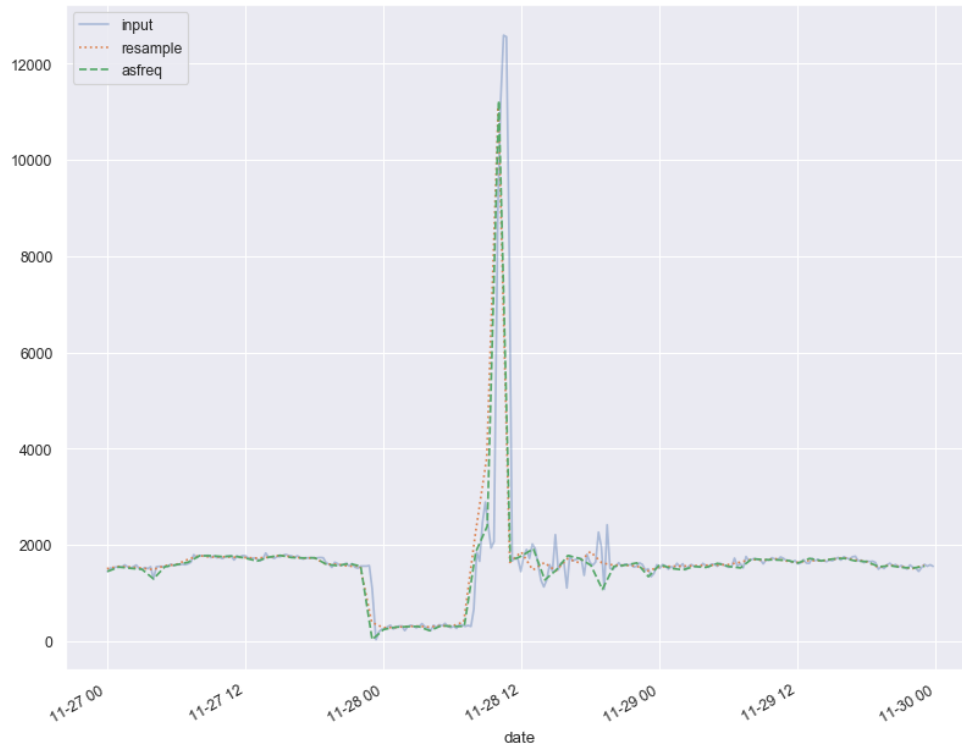
Out[2]:

		metric	value	date	timestamp
date					
2019-11-27 00:02:37	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1448.733333	2019-11-27 00:02:37	1574812957000	
2019-11-27 00:17:37	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1518.400000	2019-11-27 00:17:37	1574813857000	
2019-11-27 00:32:37	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1516.800000	2019-11-27 00:32:37	1574814757000	
2019-11-27 00:47:37	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1542.333333	2019-11-27 00:47:37	1574815657000	
2019-11-27 01:02:37	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1540.866667	2019-11-27 01:02:37	1574816557000	
...
2019-11-29 22:46:20	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1529.266667	2019-11-29 22:46:20	1575067580000	
2019-11-29 23:01:20	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1596.000000	2019-11-29 23:01:20	1575068480000	
2019-11-29 23:16:19	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1557.533333	2019-11-29 23:16:19	1575069379000	
2019-11-29 23:31:19	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1587.800000	2019-11-29 23:31:19	1575070279000	
2019-11-29 23:46:19	ack@2e46118f-c68d-40f2-8a2a-28586f2584ae	1556.000000	2019-11-29 23:46:19	1575071179000	

287 rows × 4 columns

Let's plot values from the time interval

```
In [3]: plt.figure(num=None, figsize=(12, 10), dpi=80, facecolor='w', edgecolor='k')
values.plot(alpha=0.4, style='-')
values.resample('1H').mean().plot(style=':')
values.asfreq('1H', method='bfill').plot(style='--');
plt.legend(['input', 'resample', 'asfreq'],loc='upper left');
```



```
In [4]: values.resample('60T').mean()['2019-11-28']
```

```
Out[4]: date
2019-11-28 00:00:00    281.883333
2019-11-28 01:00:00    284.316667
2019-11-28 02:00:00    305.183333
2019-11-28 03:00:00    311.133333
2019-11-28 04:00:00    294.183333
2019-11-28 05:00:00    318.450000
2019-11-28 06:00:00    314.150000
2019-11-28 07:00:00    401.733333
2019-11-28 08:00:00    2215.900000
2019-11-28 09:00:00    3821.233333
2019-11-28 10:00:00    11103.500000
2019-11-28 11:00:00    1638.366667
2019-11-28 12:00:00    1853.700000
2019-11-28 13:00:00    1479.383333
2019-11-28 14:00:00    1632.366667
2019-11-28 15:00:00    1459.316667
2019-11-28 16:00:00    1714.900000
2019-11-28 17:00:00    1632.900000
2019-11-28 18:00:00    1861.350000
2019-11-28 19:00:00    1626.433333
2019-11-28 20:00:00    1579.888889
2019-11-28 21:00:00    1578.950000
2019-11-28 22:00:00    1562.700000
2019-11-28 23:00:00    1460.400000
Freq: 60T, Name: value, dtype: float64
```

```
In [5]: values.asfreq('1H', method='bfill')['2019-11-28']
```

```
Out[5]: date
2019-11-28 00:02:37      244.266667
2019-11-28 01:02:37      289.666667
2019-11-28 02:02:37      301.733333
2019-11-28 03:02:37      306.866667
2019-11-28 04:02:37      218.266667
2019-11-28 05:02:37      320.600000
2019-11-28 06:02:37      308.533333
2019-11-28 07:02:37      307.200000
2019-11-28 08:02:37      1853.866667
2019-11-28 09:02:37      2393.866667
2019-11-28 10:02:37     11235.133333
2019-11-28 11:02:37      1667.666667
2019-11-28 12:02:37      1772.266667
2019-11-28 13:02:37      1921.733333
2019-11-28 14:02:37      1271.666667
2019-11-28 15:02:37      1480.866667
2019-11-28 16:02:37      1779.200000
2019-11-28 17:02:37      1721.200000
2019-11-28 18:02:37      1577.133333
2019-11-28 19:02:37      1076.066667
2019-11-28 20:02:37      1555.333333
2019-11-28 21:02:37      1581.866667
2019-11-28 22:02:37      1626.200000
2019-11-28 23:02:37      1340.666667
Freq: H, Name: value, dtype: float64
```

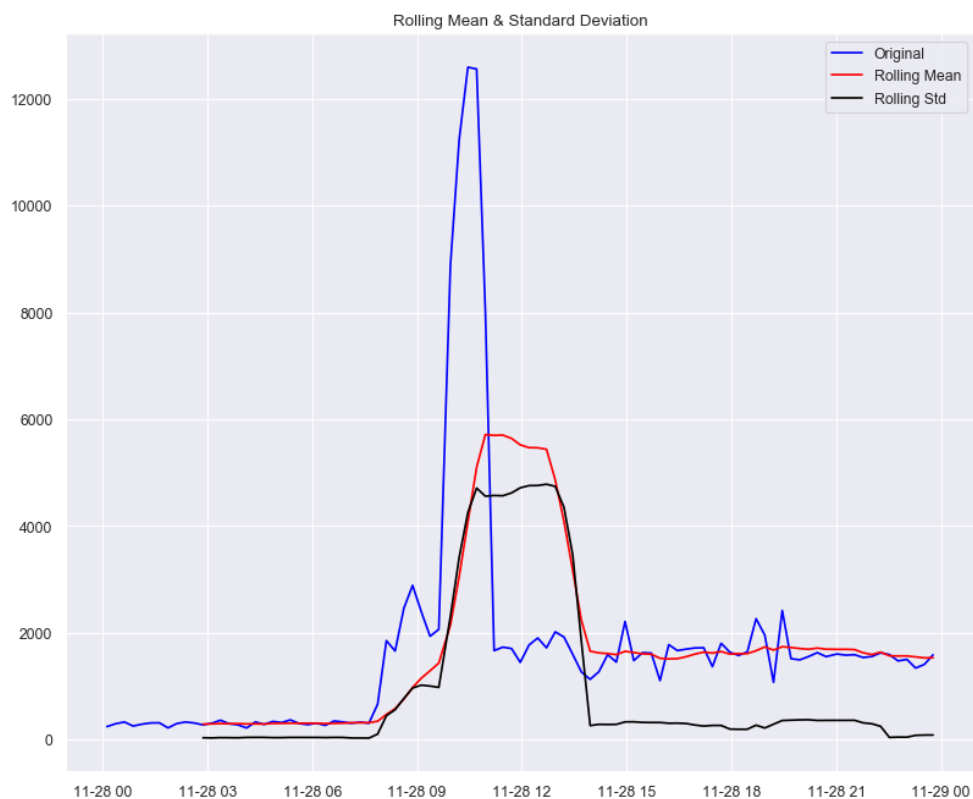
```
In [6]: from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    #Determining rolling statistics
    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()

    #Plot rolling statistics:
    plt.figure(num=None, figsize=(12, 10), dpi=80, facecolor='w', edgecolor='k')
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print( 'Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print( dfoutput )

test_stationarity(values['2019-11-28'])
```



```
Results of Dickey-Fuller Test:
Test Statistic      -3.965292
p-value             0.001604
#Lags Used          1.000000
Number of Observations Used  93.000000
Critical Value (1%)  -3.502705
Critical Value (5%)  -2.893158
Critical Value (10%) -2.583637
dtype: float64
```

```
In [7]: values.rolling(2).mean()
```

```
Out[7]: date
2019-11-27 00:02:37      NaN
2019-11-27 00:17:37    1483.566667
2019-11-27 00:32:37    1517.600000
2019-11-27 00:47:37    1529.566667
2019-11-27 01:02:37    1541.600000
...
2019-11-29 22:46:20    1489.800000
2019-11-29 23:01:20    1562.633333
2019-11-29 23:16:19    1576.766667
2019-11-29 23:31:19    1572.666667
2019-11-29 23:46:19    1571.900000
Name: value, Length: 287, dtype: float64
```

```
In [ ]:
```

```
In [ ]:
```