سلام

توضیحاتی در خصوص کد میفرستم که با مراحل پیاده شده اشنا باشید، که چناچه کتابخانه ای را نصب نداشتید نتایج راببینید:

ابتدا دیتا از سایت در فایل اکسل ذخیره کردم در و ادرسی که دیتا وجود دارد را فراخوانی کردم

پس از مرحبه پیش پردازش دیتا صورت گرفته که نتایج در عکس های پایین است:

- فراخوانی دیتا و تعیین تعداد دیتا خطا دار

In [1]:
```
import pandas as pd
import numpy as np

history=pd.read_csv("C:\\Users\CR7\Desktop\opec_data.csv")
```

In [2]:
```
print(history.shape)
history.head()
```

```
(4779, 2)
```

Out[2]:

	Date	Value
0	2003-01-02	30.05
1	2003-01-03	30.83
2	2003-01-06	30.71
3	2003-01-07	29.72
4	2003-01-08	28.86

In [3]:
```
missing_values_count = history.isnull().sum()
missing_values_count
```

Out[3]:
```
Date     0
Value    0
dtype: int64
```

- تعریف ستون تاریخ و قرار دادن بجای ایندکس ها

In [4]:
```
history['Value'].dtype
```

Out[4]:
```
dtype('float64')
```

In [5]:
```
history['Date'].dtype
```

Out[5]:
```
dtype('O')
```

In [6]:
```
history['Date'] = pd.to_datetime(history['Date'], infer_datetime_format=True)

```

In [7]:
```
history['Date'].dtype
```

Out[7]:
```
dtype('<M8[ns]')
```

In [8]:
```
history = history.sort_values('Date')
history.isnull().sum()
history = history.groupby('Date')['Value'].sum().reset_index()

history_org = history.groupby('Date')['Value'].sum().reset_index()
history.head(4)

```

Out[8]:

	Date	Value
0	2003-01-02	30.05
1	2003-01-03	30.83
2	2003-01-06	30.71

- دیتا را ماهانه کردیم

In [9]:
```
history = history_org.set_index('Date')
y = history['Value']

```

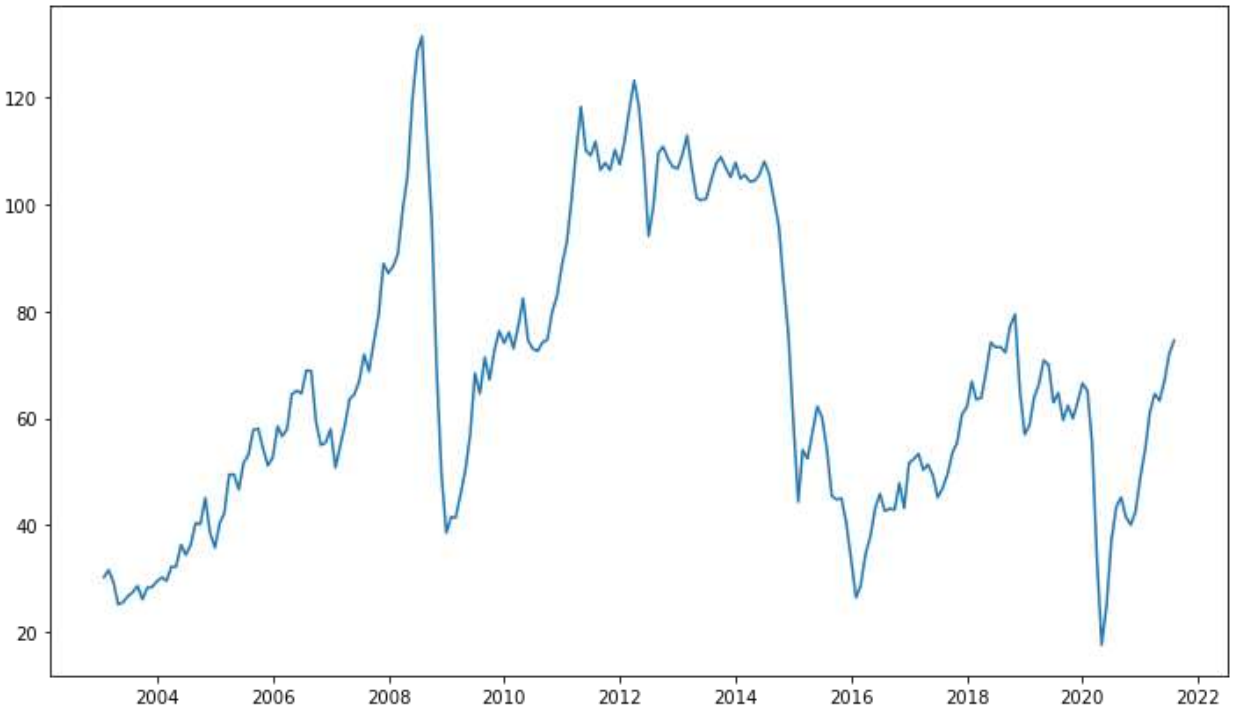
In [10]:
```
y = history['Value'].resample('M').mean()
print(len(y.values))
y.values
```

```
223
```

Out[10]:
```
array([ 30.33636364,  31.6435    ,  29.44      ,  25.23904762,
        25.62954545,  26.77857143,  27.50130435,  28.69428571,
        26.15090909,  28.34913043,  28.4825    ,  29.56952381,
        30.2647619 ,  29.6285    ,  32.22652174,  32.26380952,
        36.36571429,  34.49045455,  36.30681818,  40.41909091,
        40.21272727,  45.08380952,  38.59454545,  35.84913043,
        40.52380952,  42.1975    ,  49.49454545,  49.51047619,
        46.65454545,  51.69136364,  53.18238095,  57.8226087 ,
        58.04772727,  54.34619048,  51.13181818,  52.6352381 ,
        58.48380952,  56.618     ,  57.87130435,  64.44052632,
        65.11304348,  64.59772727,  68.88809524,  68.8073913 ,
        59.34428571,  54.97      ,  55.42227273,  57.9475    ,
        50.79272727,  54.5615    ,  58.59045455,  63.548     ,
        64.48304348,  66.89      ,  71.89272727,  68.70826087,
        74.1765    ,  79.31695652,  88.84272727,  87.054     ,
        88.35136364,  90.64380952,  99.0275    , 105.16318182,
       119.38818182, 128.33333333, 131.22173913, 112.41047619,
```

- وتستی جهت بررسی مانایی دیتاست انجام شد که نتیجه این شد دیتا ست مانایی ندارد و در گام های بعد مانا میشود:

این دیتای اصلی مااست:

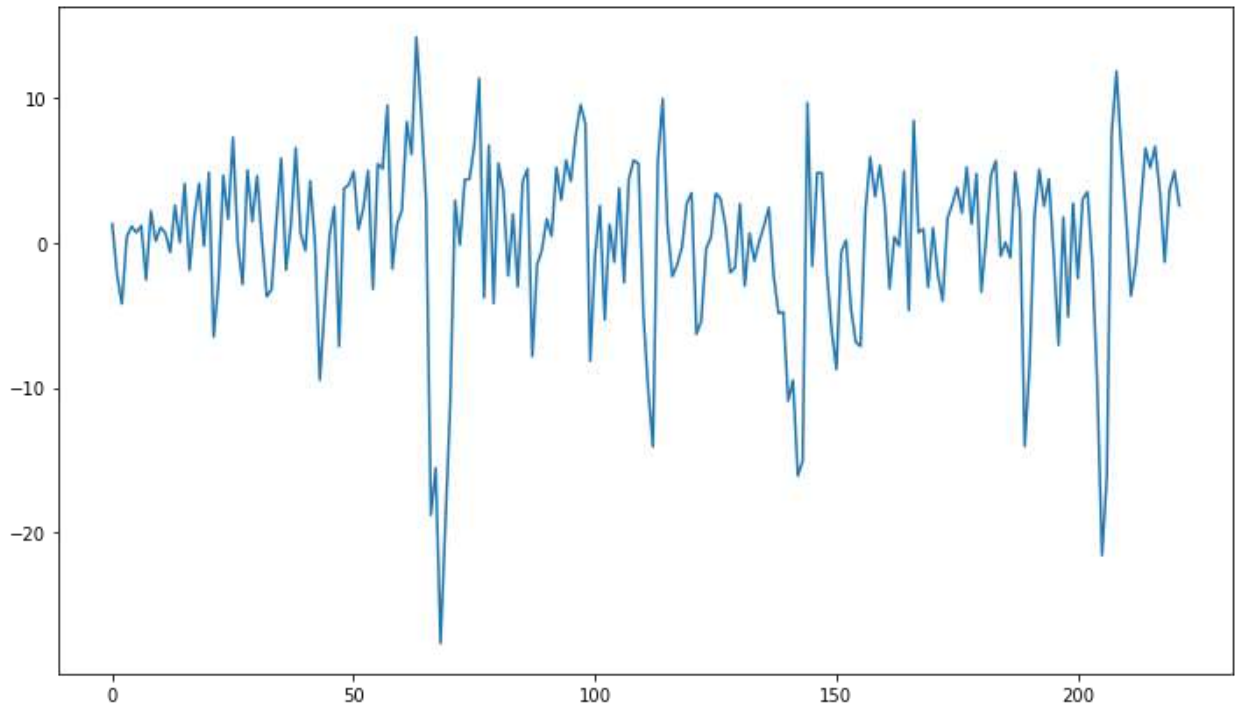


این کد برای مانایی:

In [34]:

```
from matplotlib import pyplot as plt

series=y[:]
X = series.values

diff = []
for i in range(1, len(series)):
    value = series[i] - series[i - 1]
    diff.append(value)
plt.figure(figsize=(12,7))

plt.plot(series)

plt.show()
print("")

print("  Now data is satatioanary:   ")
plt.figure(figsize=(12,7))

plt.plot(diff)
plt.show()
```

این دیتا بصورت مانا شده:

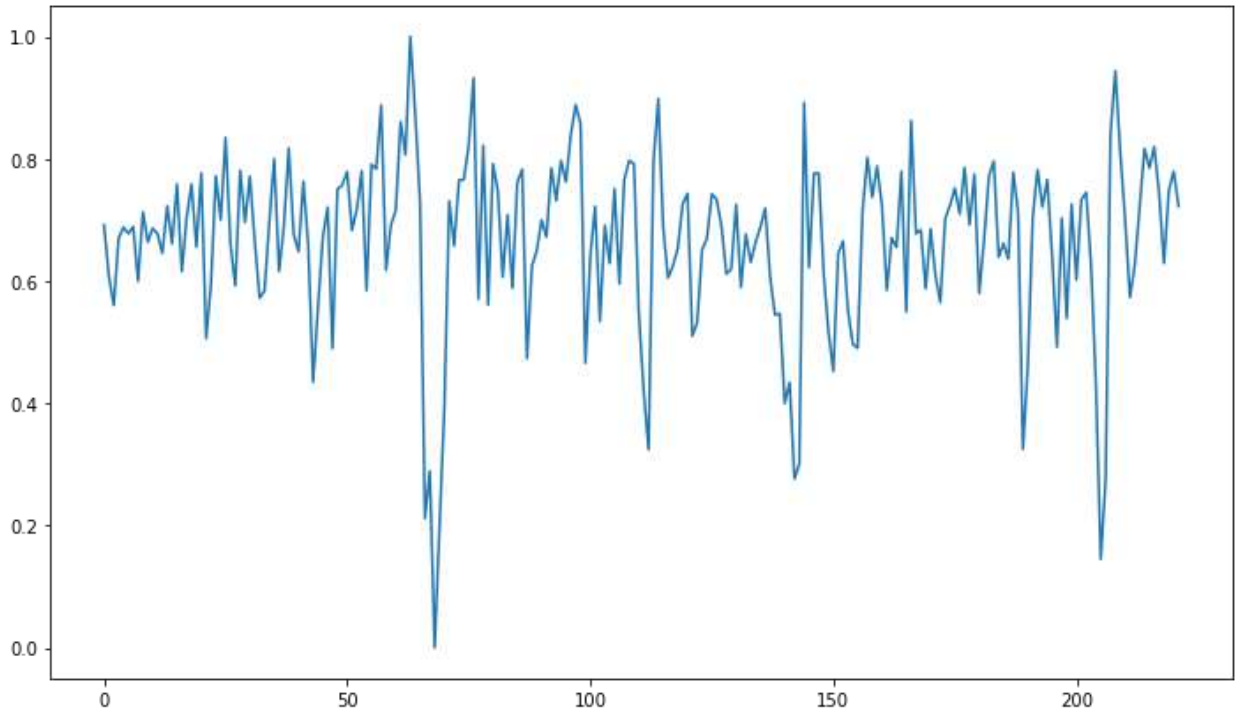


نرمال شده با متد عنوان شده در مقاله:

In [45]:

```
from sklearn.preprocessing import MinMaxScaler
diff = np.array(diff)
raw_seq = diff
raw_seq = raw_seq.reshape((len(raw_seq), 1))

# train the normalization
scaler = MinMaxScaler(feature_range=(0, 1))
scaler = scaler.fit(raw_seq)
print('Min: %f, Max: %f' % (scaler.data_min_, scaler.data_max_))
# normalize the dataset and print the first 5 rows
normalized = scaler.transform(raw_seq)


result = adfuller(normalized)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
plt.figure(figsize=(12,7))
print("  Now,data is satatioanary:   ")
plt.plot(normalized)
plt.show()
print("")

df = pd.DataFrame(normalized,columns = ['Column_A'])
a = np.array(df.values.tolist())
s=[]
for p in a:
    b=np.asscalar(p)
    s.append(b)
print(s)
```

حالا ما دیتا نرمال شده بین 0و1 داریم :



مدل سازی شبکه عصبی:*

نیاز است دیتا نرمال شده دسته بندی شود تا وارد شبکه عصبی گردد، که خروجی را میبینید

LSTM

In [18]:

```
# univariate data preparation
from numpy import array
# split a univariate sequence into samples



def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
            # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)


# define input sequence
#normalized=y[:]
raw_seq= data_norm.values

# choose a number of time steps
n_steps = 3
# split into samples
x, Y = split_sequence(raw_seq, n_steps)
# summarize the data
for i in range(len(x)): print(x[i], Y[i])
print(type(Y))
```

```
[0.3536605  0.35564285 0.32304778] 0.2947426008201296
[0.35564285 0.32304778 0.2947426 ] 0.30798144542698114
[0.32304778 0.2947426  0.30798145] 0.35948291060023347
```

در این گام کتابخانه ها فراخوانی شده و مدل ایجاد شده و پیش بینی برای ماه بعدی تعیین شده:

In [30]:
```
# univariate stacked lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense

# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
x = x.reshape((x.shape[0], x.shape[1], n_features))

```

In [36]:
```
# define model
model = Sequential()
model.add(LSTM(20, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
model.add(LSTM(4, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

```

In [37]:
```
# fit model
model.fit(x, Y, epochs=150, verbose=0,batch_size=15)

```

Out[37]:
```
<tensorflow.python.keras.callbacks.History at 0x1eb7faeb848>
```

In [ ]:
```
# univariate stacked lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense

# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
x = x.reshape((x.shape[0], x.shape[1], n_features))

```

In [ ]:
```
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

```

In [ ]:
```
# fit model
model.fit(x, Y, epochs=100, verbose=0)

```

In [ ]:
```
# demonstrate prediction
x_input = array([0.43367823, 0.47756743, 0.5004065212522483])
x_input = x_input.reshape((1, n_steps, n_features))
yhat_norm = model.predict(x_input, verbose=0)
print(yhat_norm)
```

**در اینجا ما سه دیتای اخر جدول را داده ایم و ماه بعد (2021-08-31) برای ماپیش بینی کرده است با مقدار 89.94952:

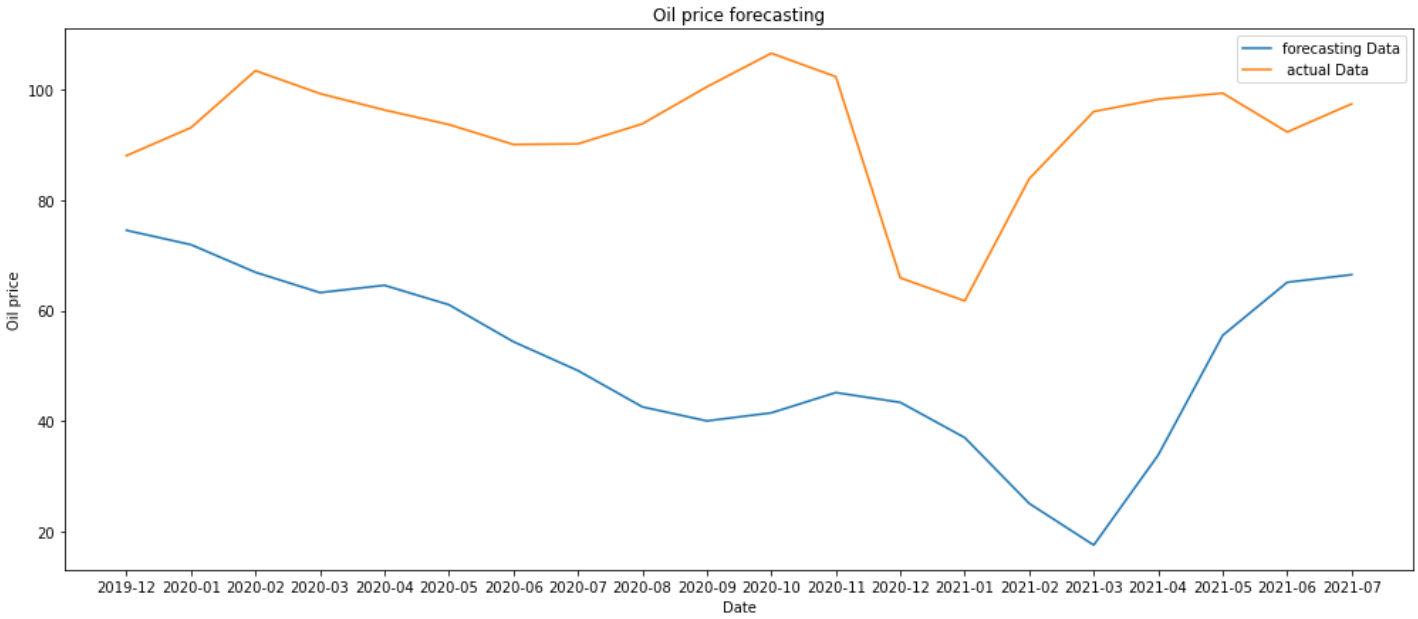
```
In [40]:
# demonstrate prediction
x_input = array([0.43367823, 0.47756743, 0.5004065212522483])
x_input = x_input.reshape((1, n_steps, n_features))
yhat_norm = model.predict(x_input, verbose=0)
print("مقدار پیش بینی شده که به صورت نرمال شده است:    {}".format(yhat_norm))

[[0.6365643]]    مقدار پیش بینی شده که به صورت نرمال شده است:

In [42]:
raw_seq = series.values
max_number=max(raw_seq)
min_number=min(raw_seq)
x_orginal=yhat_norm*(max_number-min_number)+min_number
print("مقدار پیش بینی شده که  به فرم دیتا اولیه در امده:    {}".format(x_orginal))

[[89.949524]]    مقدار پیش بینی شده که  به فرم دیتا اولیه در امده:

In [ ]:

In [ ]:
```

در فایل دوم به بررسی دقت مدل پرداخته شده است: (به علت تداخل محاسبات جدا قرار دادم، محتوا یکی است اما برای دقت مقداری روش متفاوت است که توضیحات در اینجا نمیگنجد)

میزان خطا را اینجا برای 20 دیتای اخر محاسبه شده است:

RMSE =39.747596142211712

و نتیجه بدست امده برای 20 دیتا اخر به این صورت است که مقدار واقعی و مقدار پیش بینی شده ان در شکل نشان داده شده است:


Oil price forecasting
forecasting Data
actual Data
Oil price
100
80
60
40
20
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05
2020-06
2020-07
2020-08
2020-09
2020-10
2020-11
2020-12
2021-01
2021-02
2021-03
2021-04
2021-05
2021-06
2021-07
Date