temp

March 4, 2020

In [49]:
```
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
```

In [236]:
```
%load_ext autoreload
%autoreload 2
```

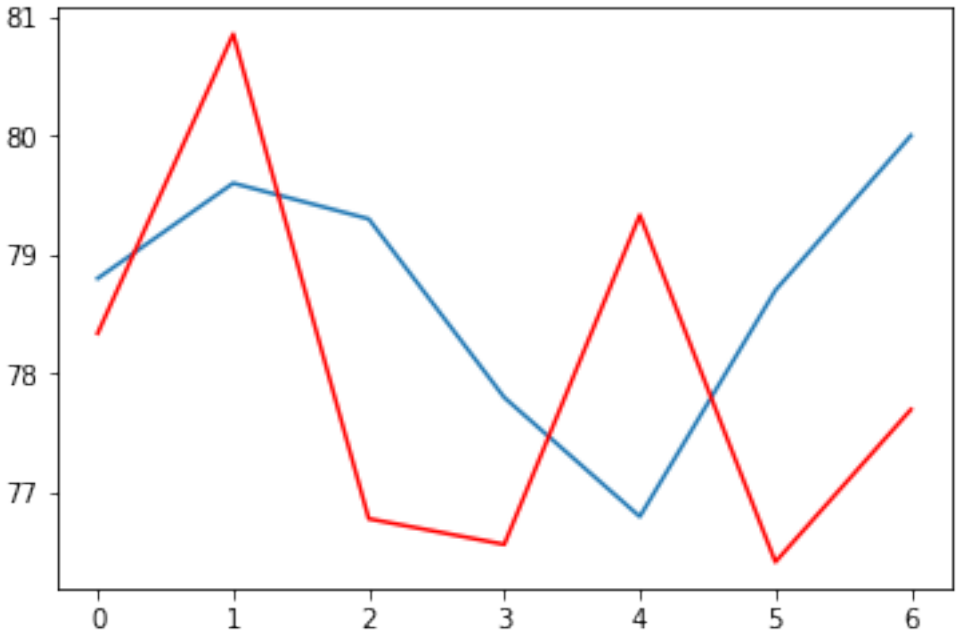
In [52]:
```
series = read_csv('temperature - temperature.csv', header=0, index_col=0)
values = DataFrame(series.values)
dataframe = concat([values.shift(1), values], axis=1)
dataframe.columns = ['t-1', 't+1']
X = dataframe.values
train, test = X[1:len(X)-7], X[len(X)-7:]
train_X, train_y = train[:,0], train[:,1]
test_X, test_y = test[:,0], test[:,1]
```

In [53]:
```
import matplotlib.pyplot as plt
```

In [54]:
```
from statsmodels.tsa.ar_model import AutoReg as AR
from sklearn.metrics import mean_squared_error
```

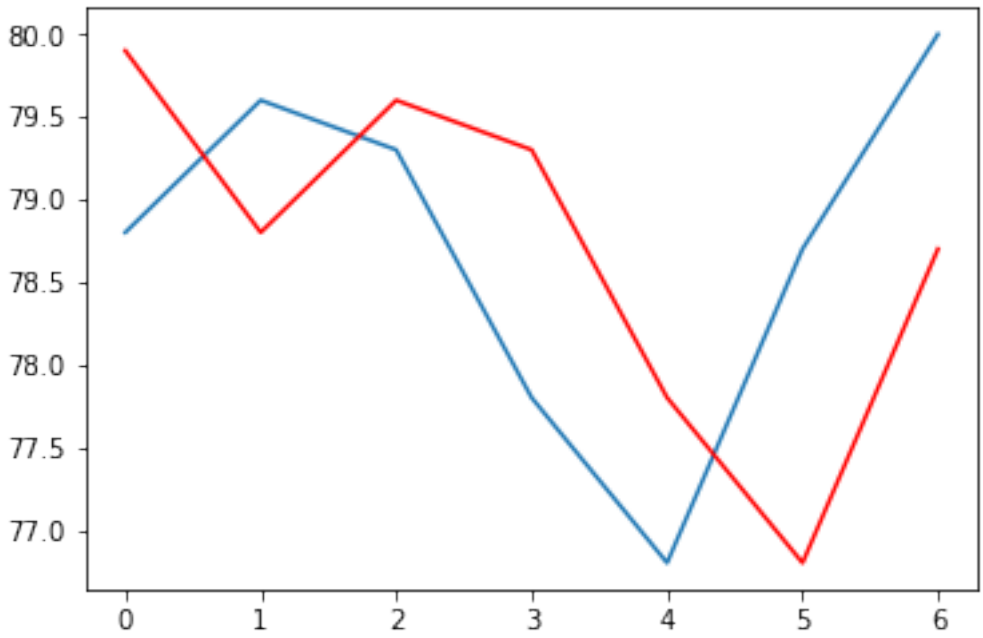
In [55]:
```
X = series.values
train, test = X[1:len(X)-7], X[len(X)-7:]
model = AR(train, lags=365*3)

model_fit = model.fit()
print('Lag: %s' % model_fit)
print('Coefficients: %s' % model_fit.params)

predictions = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=

error = mean_squared_error(test, predictions)
print('Test Error: %.3f' % error)

pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```

```
Lag: <statsmodels.tsa.ar_model.AutoRegResultsWrapper object at 0x7f2bb5dbe860>
Coefficients: [2.05155974e+01 3.16997098e-01 4.97988658e-02 ... 4.91296199e-03
 1.09503421e-02 5.30677068e-03]
Test Error: 3.793
```




```
In [56]: def model_persistence(x):
             return x


         predictions = list()
         for x in test_X:
             yhat = model_persistence(x)
             predictions.append(yhat)
         test_score = mean_squared_error(test_y, predictions)
         print('Test Error: %.3f' % test_score)
         pyplot.plot(test_y)
         pyplot.plot(predictions, color='red')
         pyplot.show()
```

```
Test Error: 1.499
```

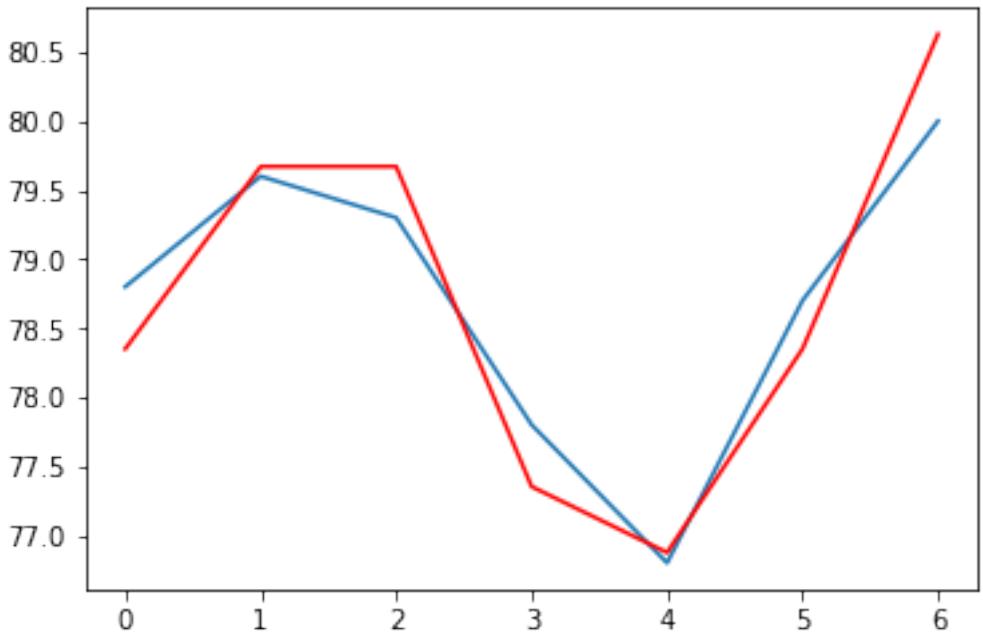



```
In [57]: import keras

In [58]: from keras import layers

In [59]: from keras import Sequential

In [60]: from keras import backend as K
         cfg = K.tf.ConfigProto()
         cfg.gpu_options.allow_growth = True
         K.set_session(K.tf.Session(config=cfg))

In [61]: model = Sequential()

         model.add(layers.Embedding(input_dim=1000, output_dim=64))

         model.add(layers.SimpleRNN(64))

         model.add(layers.Dense(1))

         model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, None, 64)          64000
_________________________________________________________________
```

```
simple_rnn_3 (SimpleRNN)     (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 65
=================================================================
Total params: 72,321
Trainable params: 72,321
Non-trainable params: 0
_________________________________________________________________
```

```
In [62]: #series.shape

In [45]: x = series['temperature'][:-100]

In [46]: y = series['temperature'][1:-99]

In [47]: model.compile(loss='mae', optimizer='adam')

In [63]: # model.fit(x, y, epochs=40, batch_size=64)

In [20]: # from keras import backend as K
         # K.tensorflow_backend._get_available_gpus()

In [64]: # model.predict(series['temperature'][-6:-2])

In [59]: series['temperature'][-5:-1]

Out[59]: Date
         01-23-2020    79.3
         01-24-2020    77.8
         01-25-2020    76.8
         01-26-2020    78.7
         Name: temperature, dtype: float64

In [60]: test_Res = model.predict(test)

In [30]: test

Out[30]: array([[78.8],
                [79.6],
                [79.3],
                [77.8],
                [76.8],
                [78.7],
                [80. ]])

In [61]: pyplot.plot(test)
         pyplot.plot(test_Res, color='red')
         pyplot.show()
```



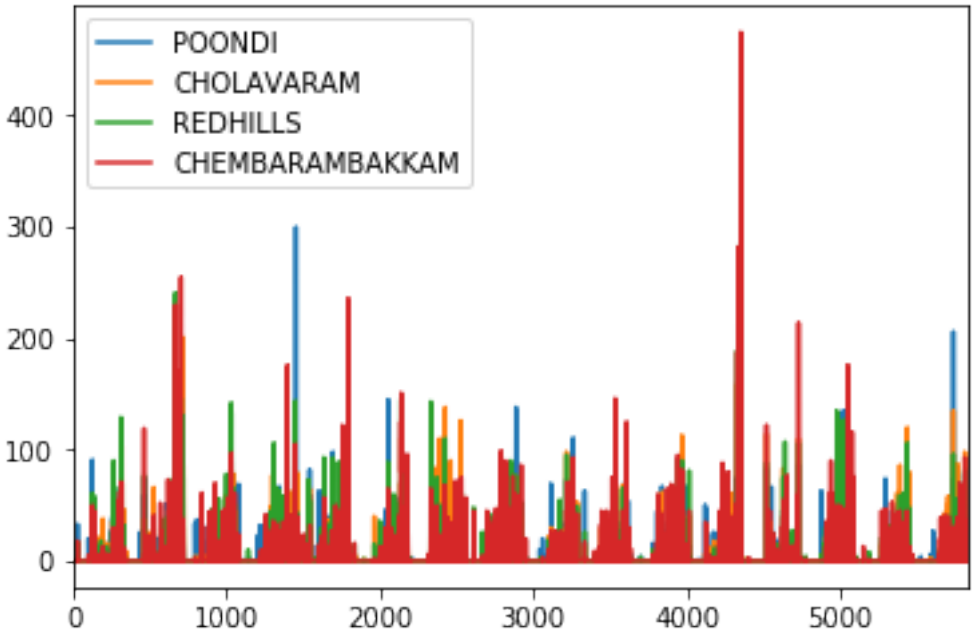

```
In [20]: ls
```

```
'0th review.pptx'*                  INCHENAI.txt*
 chennai_reservoir_levels.csv*      temperature.csv*
 chennai_reservoir_rainfall.csv*  'temperature - temperature.csv'*
 EDA.ipynb*                         temp.ipynb*
```

```
In [21]: rain = read_csv('chennai_reservoir_rainfall.csv')
```

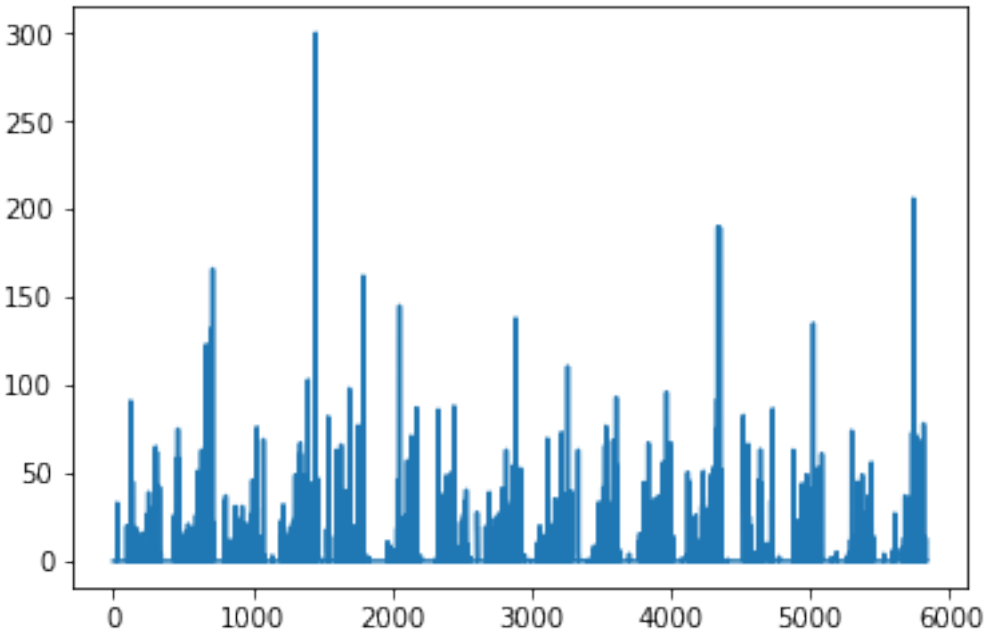
```
In [22]: rain.plot()
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb7101aa3c8>
```



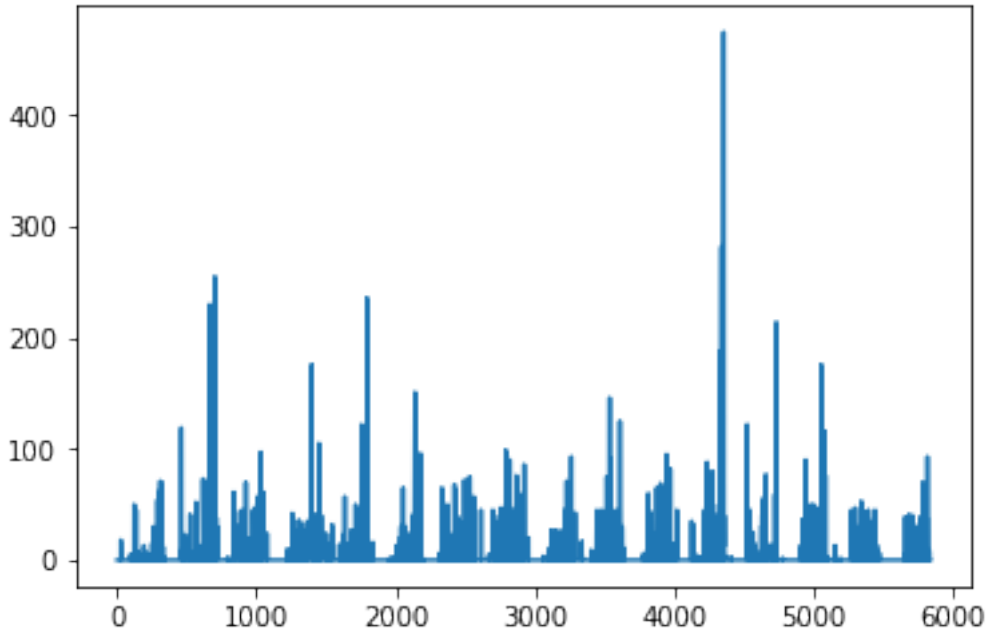

```
In [23]: plt.plot(rain['POONDI'])
```

```
Out[23]: [<matplotlib.lines.Line2D at 0x7fb7a319c0f0>]
```



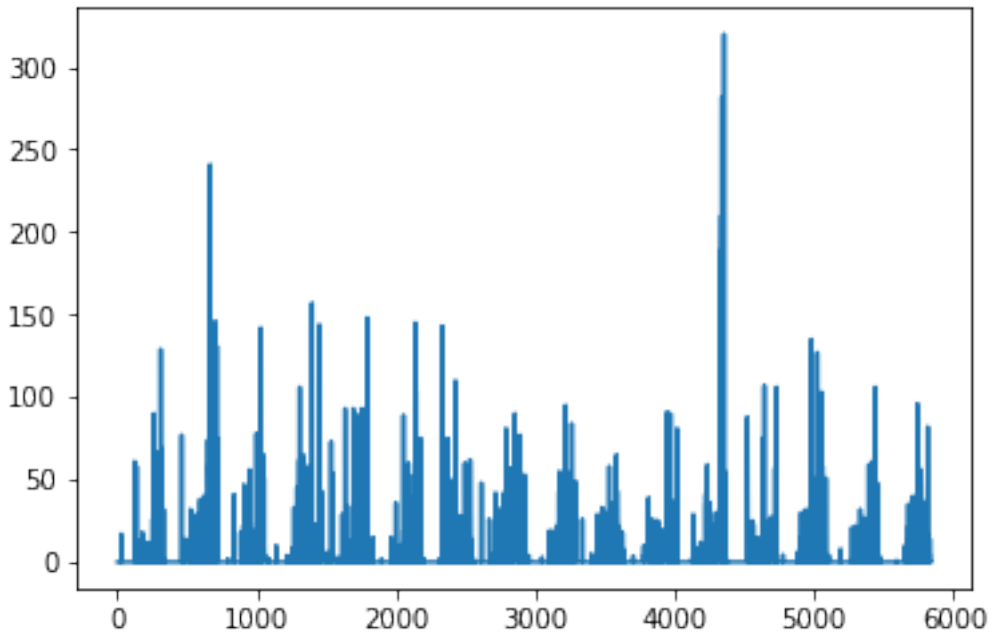
In [24]: plt.plot(rain['CHEMBARAMBAKKAM'])

Out[24]: [<matplotlib.lines.Line2D at 0x7fb7101ac3c8>]



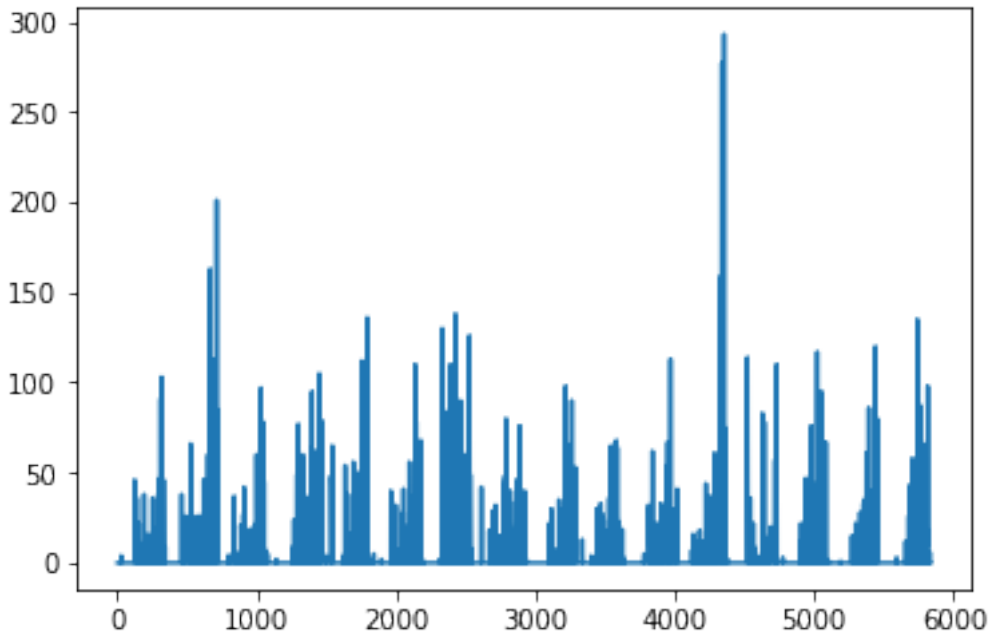
In [25]: plt.plot(rain['REDHILLS'])

Out[25]: [<matplotlib.lines.Line2D at 0x7fb71018ed30>]



In [26]:
```
plt.plot(rain['CHOLAVARAM'])
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x7fb7a30b7518>]
```



In [27]:
```
model = Sequential()

model.add(layers.Embedding(input_dim=1000, output_dim=64))

model.add(layers.LSTM(64))

model.add(layers.Dense(1))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 64)          64000
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
```

```
Total params: 97,089
Trainable params: 97,089
Non-trainable params: 0
_________________________________________________________________
```

```
In [35]: def Model():
             model = Sequential([layers.Embedding(input_dim=1000, output_dim=64),
                                 layers.LSTM(64),
                                 layers.Dense(1)])
             model.compile(loss='mae', optimizer='adam')
             return model
```

```
In [36]: md = Model()
```

```
In [65]: # md.fit(x, y, epochs=20, batch_size=64)
```

```
In [54]: x[0:5]
```

```
Out[54]: Date
         01-01-1995    72.4
         01-02-1995    73.5
         01-03-1995    72.6
         01-04-1995    75.2
         01-05-1995    74.8
         Name: temperature, dtype: float64
```

```
In [55]: md.predict(x[0:5])
```

```
Out[55]: array([[74.1002  ],
                [75.920105],
                [74.1002  ],
                [76.24014 ],
                [75.72263 ]], dtype=float32)
```

```
In [84]: dp10 = model.predict(dt)
```

```
In [86]: dt = x[0:1]
```

```
In [85]: dt
```

```
Out[85]: array([[76.87471 ],
                [76.87471 ],
                [76.87471 ],
                [76.87471 ],
                [76.87471 ],
                [76.87471 ],
                [78.35028 ],
                [78.35028 ],
                [78.35028 ],
                [79.6707  ],
                [77.352325]], dtype=float32)
```

```
In [87]: dt

Out[87]: Date
         01-01-1995    72.4
         Name: temperature, dtype: float64

In [88]: dp1 = []

In [89]: for i in range(6):
             dt = model.predict(dt)
             dp1.append(dt)

In [ ]:

In [90]: dp1

Out[90]: [array([[73.81857]], dtype=float32),
          array([[74.47859]], dtype=float32),
          array([[76.134125]], dtype=float32),
          array([[76.87471]], dtype=float32),
          array([[76.87471]], dtype=float32),
          array([[76.87471]], dtype=float32)]

In [78]: dp10

Out[78]: array([[86.18312 ],
                [86.18312 ],
                [88.2327  ],
                [87.686356],
                [88.2327  ]], dtype=float32)

In [14]: n_steps = 3

In [16]: def split_sequences(sequences, n_steps):
             X, y = [], []
             for i in range(len(sequences)):
                 end_ix = i + n_steps
                 if end_ix > len(sequences):
                     break
                 seq_x, seq_y = sequences[i:end_ix, : -1], sequences[end_ix-1, -1]
                 X.append(seq_x)
                 y.append(seq_y)
             return array(X), array(y)


In [18]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

In [19]: temp = pd.read_csv('processed-temperature.csv')
```

```
In [27]: temp = temp.drop(columns='Unnamed: 0')

In [29]: ls

'0th review.pptx'*                 processed-temperature.csv*
 chennai_reservoir_levels.csv*     temperature.csv*
 chennai_reservoir_rainfall.csv*  'temperature - temperature.csv'*
 EDA.ipynb*                        temp.ipynb*
 INCHENAI.txt*                     text_generation.ipynb*


In [30]: rain = pd.read_csv('chennai_reservoir_rainfall.csv')

In [32]: rain.describe()

Out[32]:              POONDI    CHOLAVARAM      REDHILLS  CHEMBARAMBAKKAM
         count  5836.000000  5836.000000  5836.000000      5836.000000
         mean      3.464402     3.758773     3.828992         4.012688
         std      13.065897    14.267783    15.011809        16.251079
         min       0.000000     0.000000     0.000000         0.000000
         25%       0.000000     0.000000     0.000000         0.000000
         50%       0.000000     0.000000     0.000000         0.000000
         75%       0.000000     0.000000     0.000000         0.000000
         max     300.000000   293.000000   320.000000       475.000000

In [34]: temp.head()

Out[34]:          Date  temperature
         0  1995-01-01         72.4
         1  1995-01-02         73.5
         2  1995-01-03         72.6
         3  1995-01-04         75.2
         4  1995-01-05         74.8

In [35]: rain.head()

Out[35]:          Date  POONDI  CHOLAVARAM  REDHILLS  CHEMBARAMBAKKAM
         0  01-01-2004     0.0         0.0       0.0              0.0
         1  02-01-2004     0.0         0.0       0.0              0.0
         2  03-01-2004     0.0         0.0       0.0              0.0
         3  04-01-2004     0.0         0.0       0.0              0.0
         4  05-01-2004     0.0         0.0       0.0              0.0

In [37]: temp[temp['Date'] == '2004-01-01']

Out[37]:             Date  temperature
         3287  2004-01-01         76.9

In [38]: idx = 3287
```

```
In [42]: temp.shape

Out[42]: (9158, 2)

In [57]: tmp = temp[3287:]

In [59]: tmp.iloc[0]

Out[59]: Date           2004-01-01
         temperature          76.9
         Name: 3287, dtype: object

In [72]: tmp.iloc[-1]

Out[72]: Date           2020-01-27
         temperature            80
         Name: 9157, dtype: object

In [68]: rain.iloc[-1]

Out[68]: Date                23-12-2019
         POONDI                       0
         CHOLAVARAM                   0
         REDHILLS                     0
         CHEMBARAMBAKKAM              0
         Name: 5835, dtype: object

In [82]: tmp[tmp['Date'] == '2019-12-24']

Out[82]:             Date  temperature
         9123  2019-12-24         77.5

In [85]: tmp = tmp.loc[:9123]

In [97]: tmp.shape

Out[97]: (5836, 2)

In [96]: tmp = tmp.drop(tmp.index[-1])

In [216]: # rain.head()

In [98]: rain["Date"] = pd.to_datetime(rain["Date"], format='%d-%m-%Y')

In [217]: rain.head()

Out[217]:         Date  POONDI  CHOLAVARAM  REDHILLS  CHEMBARAMBAKKAM
          0 2004-01-01     0.0         0.0       0.0              0.0
          1 2004-01-02     0.0         0.0       0.0              0.0
          2 2004-01-03     0.0         0.0       0.0              0.0
          3 2004-01-04     0.0         0.0       0.0              0.0
          4 2004-01-05     0.0         0.0       0.0              0.0
```

```
In [100]: water_level = pd.read_csv('chennai_reservoir_levels.csv')

In [218]: water_level.head()

Out[218]:          Date  POONDI  CHOLAVARAM  REDHILLS  CHEMBARAMBAKKAM
          0  01-01-2004     3.9         0.0     268.0              0.0
          1  02-01-2004     3.9         0.0     268.0              0.0
          2  03-01-2004     3.9         0.0     267.0              0.0
          3  04-01-2004     3.9         0.0     267.0              0.0
          4  05-01-2004     3.8         0.0     267.0              0.0

In [124]: # input_seq = temp['temperature'][1:] + rain['POONDI'][1:] + water_level['POONDI'][:

In [163]: input_seq = pd.DataFrame()

In [164]: input_seq['temperature'] = tmp['temperature'][1:].values
          input_seq['rain'] = rain['POONDI'][1:].values
          input_seq['water_level'] = water_level['POONDI'][:-1].values

In [165]: tmp['temperature'][1:].values.shape

Out[165]: (5835,)

In [166]: rain['POONDI'][1:]

Out[166]: 1       0.0
          2       0.0
          3       0.0
          4       0.0
          5       0.0
                 ...
          5831    0.0
          5832    0.0
          5833    0.0
          5834    0.0
          5835    0.0
          Name: POONDI, Length: 5835, dtype: float64

In [167]: input_seq['rain']

Out[167]: 0       0.0
          1       0.0
          2       0.0
          3       0.0
          4       0.0
                 ...
          5830    0.0
          5831    0.0
          5832    0.0
          5833    0.0
          5834    0.0
          Name: rain, Length: 5835, dtype: float64
```

```
In [168]: tmp.shape

Out[168]: (5836, 2)

In [169]: tmp.index = range(5836)

In [173]: output_seq = water_level['POONDI'][1:].values

In [171]: input_seq.values

Out[171]: array([[  75.1,    0. ,    3.9],
                 [  74.3,    0. ,    3.9],
                 [  75.6,    0. ,    3.9],
                 ...,
                 [  79.7,    0. , 1529. ],
                 [  78.5,    0. , 1522. ],
                 [  79. ,    0. , 1514. ]])

In [219]: # tmp

In [220]: input_seq.head()

Out[220]:    temperature  rain  water_level
          0         75.1   0.0          3.9
          1         74.3   0.0          3.9
          2         75.6   0.0          3.9
          3         76.3   0.0          3.9
          4         76.0   0.0          3.8

In [179]: output_seq.resize(output_seq.shape[0], 1)

In [180]: dataset = hstack((input_seq, output_seq))

In [178]: input_seq.shape

Out[178]: (5835, 3)

In [182]: X, y = split_sequences(dataset, n_steps)

In [184]: y.shape

Out[184]: (5833,)

In [186]: n_features = X.shape[2]
          # define model
          model = Sequential()
          model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
          model.add(Dense(1))
          model.compile(optimizer='adam', loss='mse')
```

```
In [221]: # fit model
          model.fit(X[:-50, :, :], y[:-50], epochs=100)
```

```
Epoch 1/100
5783/5783 [==============================] - 1s 122us/step - loss: 3545.9042
Epoch 2/100
5783/5783 [==============================] - 1s 120us/step - loss: 3476.8187
Epoch 3/100
5783/5783 [==============================] - 1s 128us/step - loss: 3478.0554
Epoch 4/100
5783/5783 [==============================] - 1s 135us/step - loss: 3502.2877
Epoch 5/100
5783/5783 [==============================] - 1s 124us/step - loss: 3453.2360
Epoch 6/100
5783/5783 [==============================] - 1s 125us/step - loss: 3551.1283
Epoch 7/100
5783/5783 [==============================] - 1s 126us/step - loss: 3498.3779
Epoch 8/100
5783/5783 [==============================] - 1s 125us/step - loss: 3478.6346
Epoch 9/100
5783/5783 [==============================] - 1s 121us/step - loss: 3494.8484
Epoch 10/100
5783/5783 [==============================] - 1s 126us/step - loss: 3529.5244
Epoch 11/100
5783/5783 [==============================] - 1s 125us/step - loss: 3513.8210
Epoch 12/100
5783/5783 [==============================] - 1s 135us/step - loss: 3523.9985
Epoch 13/100
5783/5783 [==============================] - 1s 144us/step - loss: 3535.2065
Epoch 14/100
5783/5783 [==============================] - 1s 127us/step - loss: 3514.7961
Epoch 15/100
5783/5783 [==============================] - 1s 121us/step - loss: 3583.1277
Epoch 16/100
5783/5783 [==============================] - 1s 125us/step - loss: 3557.9519
Epoch 17/100
5783/5783 [==============================] - 1s 129us/step - loss: 3524.7452
Epoch 18/100
5783/5783 [==============================] - 1s 138us/step - loss: 3487.1807
Epoch 19/100
5783/5783 [==============================] - 1s 126us/step - loss: 3465.8071
Epoch 20/100
5783/5783 [==============================] - 1s 125us/step - loss: 3523.2553
Epoch 21/100
5783/5783 [==============================] - 1s 123us/step - loss: 3500.3443
Epoch 22/100
5783/5783 [==============================] - 1s 121us/step - loss: 3519.5872
Epoch 23/100
```

```
5783/5783 [==============================] - 1s 153us/step - loss: 3474.8714
Epoch 24/100
5783/5783 [==============================] - 1s 133us/step - loss: 3721.7310
Epoch 25/100
5783/5783 [==============================] - 1s 127us/step - loss: 3474.1379
Epoch 26/100
5783/5783 [==============================] - 1s 128us/step - loss: 3539.8035
Epoch 27/100
5783/5783 [==============================] - 1s 134us/step - loss: 3594.3512
Epoch 28/100
5783/5783 [==============================] - 1s 128us/step - loss: 3478.1662
Epoch 29/100
5783/5783 [==============================] - 1s 127us/step - loss: 3457.3082
Epoch 30/100
5783/5783 [==============================] - 1s 145us/step - loss: 3487.0570
Epoch 31/100
5783/5783 [==============================] - 1s 126us/step - loss: 3484.9879
Epoch 32/100
5783/5783 [==============================] - 1s 125us/step - loss: 3452.5899
Epoch 33/100
5783/5783 [==============================] - 1s 123us/step - loss: 3535.5253
Epoch 34/100
5783/5783 [==============================] - 1s 121us/step - loss: 3497.7751
Epoch 35/100
5783/5783 [==============================] - 1s 124us/step - loss: 3507.4616
Epoch 36/100
5783/5783 [==============================] - 1s 121us/step - loss: 3484.5387
Epoch 37/100
5783/5783 [==============================] - 1s 120us/step - loss: 3518.2336
Epoch 38/100
5783/5783 [==============================] - 1s 123us/step - loss: 3504.6314
Epoch 39/100
5783/5783 [==============================] - 1s 122us/step - loss: 3525.4275
Epoch 40/100
5783/5783 [==============================] - 1s 120us/step - loss: 3510.1822
Epoch 41/100
5783/5783 [==============================] - 1s 122us/step - loss: 3431.3550
Epoch 42/100
5783/5783 [==============================] - 1s 120us/step - loss: 3518.0483
Epoch 43/100
5783/5783 [==============================] - 1s 121us/step - loss: 3442.5421
Epoch 44/100
5783/5783 [==============================] - 1s 130us/step - loss: 3487.6259
Epoch 45/100
5783/5783 [==============================] - 1s 125us/step - loss: 3468.7245
Epoch 46/100
5783/5783 [==============================] - 1s 138us/step - loss: 3456.1804
Epoch 47/100
```

```
5783/5783 [==============================] - 1s 131us/step - loss: 3468.8204
Epoch 48/100
5783/5783 [==============================] - 1s 122us/step - loss: 3491.0281
Epoch 49/100
5783/5783 [==============================] - 1s 123us/step - loss: 3501.5311
Epoch 50/100
5783/5783 [==============================] - 1s 122us/step - loss: 3475.2645
Epoch 51/100
5783/5783 [==============================] - 1s 123us/step - loss: 3441.1848
Epoch 52/100
5783/5783 [==============================] - 1s 122us/step - loss: 3452.1192
Epoch 53/100
5783/5783 [==============================] - 1s 122us/step - loss: 3470.3578
Epoch 54/100
5783/5783 [==============================] - 1s 123us/step - loss: 3489.8401
Epoch 55/100
5783/5783 [==============================] - 1s 122us/step - loss: 3549.1811
Epoch 56/100
5783/5783 [==============================] - 1s 127us/step - loss: 3576.1878
Epoch 57/100
5783/5783 [==============================] - 1s 124us/step - loss: 3520.0043
Epoch 58/100
5783/5783 [==============================] - 1s 127us/step - loss: 3468.0529
Epoch 59/100
5783/5783 [==============================] - 1s 122us/step - loss: 3513.7375
Epoch 60/100
5783/5783 [==============================] - 1s 129us/step - loss: 3515.1183
Epoch 61/100
5783/5783 [==============================] - 1s 122us/step - loss: 3494.1792
Epoch 62/100
5783/5783 [==============================] - 1s 122us/step - loss: 3506.5036
Epoch 63/100
5783/5783 [==============================] - 1s 127us/step - loss: 3462.8644
Epoch 64/100
5783/5783 [==============================] - 1s 125us/step - loss: 3467.3967
Epoch 65/100
5783/5783 [==============================] - 1s 123us/step - loss: 3488.9697
Epoch 66/100
5783/5783 [==============================] - 1s 123us/step - loss: 3589.1605
Epoch 67/100
5783/5783 [==============================] - 1s 123us/step - loss: 3539.9708
Epoch 68/100
5783/5783 [==============================] - 1s 122us/step - loss: 3465.9816
Epoch 69/100
5783/5783 [==============================] - 1s 122us/step - loss: 3457.6498
Epoch 70/100
5783/5783 [==============================] - 1s 122us/step - loss: 3500.3505
Epoch 71/100
```

```
5783/5783 [==============================] - 1s 122us/step - loss: 3723.3815
Epoch 72/100
5783/5783 [==============================] - 1s 124us/step - loss: 3655.5218
Epoch 73/100
5783/5783 [==============================] - 1s 124us/step - loss: 3607.8366
Epoch 74/100
5783/5783 [==============================] - 1s 124us/step - loss: 3548.5376
Epoch 75/100
5783/5783 [==============================] - 1s 125us/step - loss: 3526.2781
Epoch 76/100
5783/5783 [==============================] - 1s 126us/step - loss: 3499.6075
Epoch 77/100
5783/5783 [==============================] - 1s 123us/step - loss: 3499.4505
Epoch 78/100
5783/5783 [==============================] - 1s 124us/step - loss: 3497.2485
Epoch 79/100
5783/5783 [==============================] - 1s 123us/step - loss: 3507.4348
Epoch 80/100
5783/5783 [==============================] - 1s 121us/step - loss: 3472.6291
Epoch 81/100
5783/5783 [==============================] - 1s 123us/step - loss: 3460.8719
Epoch 82/100
5783/5783 [==============================] - 1s 125us/step - loss: 3498.3064
Epoch 83/100
5783/5783 [==============================] - 1s 124us/step - loss: 3444.4510
Epoch 84/100
5783/5783 [==============================] - 1s 122us/step - loss: 3519.9193
Epoch 85/100
5783/5783 [==============================] - 1s 129us/step - loss: 3608.9999
Epoch 86/100
5783/5783 [==============================] - 1s 124us/step - loss: 3494.2311
Epoch 87/100
5783/5783 [==============================] - 1s 129us/step - loss: 3456.7376
Epoch 88/100
5783/5783 [==============================] - 1s 128us/step - loss: 3477.2309
Epoch 89/100
5783/5783 [==============================] - 1s 131us/step - loss: 3444.6021
Epoch 90/100
5783/5783 [==============================] - 1s 127us/step - loss: 3526.2411
Epoch 91/100
5783/5783 [==============================] - 1s 129us/step - loss: 3484.2936
Epoch 92/100
5783/5783 [==============================] - 1s 125us/step - loss: 3490.6116
Epoch 93/100
5783/5783 [==============================] - 1s 130us/step - loss: 3524.6297
Epoch 94/100
5783/5783 [==============================] - 1s 124us/step - loss: 3522.5599
Epoch 95/100
```

```
5783/5783 [==============================] - 1s 128us/step - loss: 3468.0549
Epoch 96/100
5783/5783 [==============================] - 1s 126us/step - loss: 3464.8816
Epoch 97/100
5783/5783 [==============================] - 1s 127us/step - loss: 3477.2957
Epoch 98/100
5783/5783 [==============================] - 1s 125us/step - loss: 3492.1176
Epoch 99/100
5783/5783 [==============================] - 1s 136us/step - loss: 3496.4303
Epoch 100/100
5783/5783 [==============================] - 1s 133us/step - loss: 3493.6423
```

```
Out[221]: <keras.callbacks.History at 0x7f0e18b10d68>
```

```
In [195]: # input_seq.iloc[-10: -1]
```

```
In [202]: y.shape
```

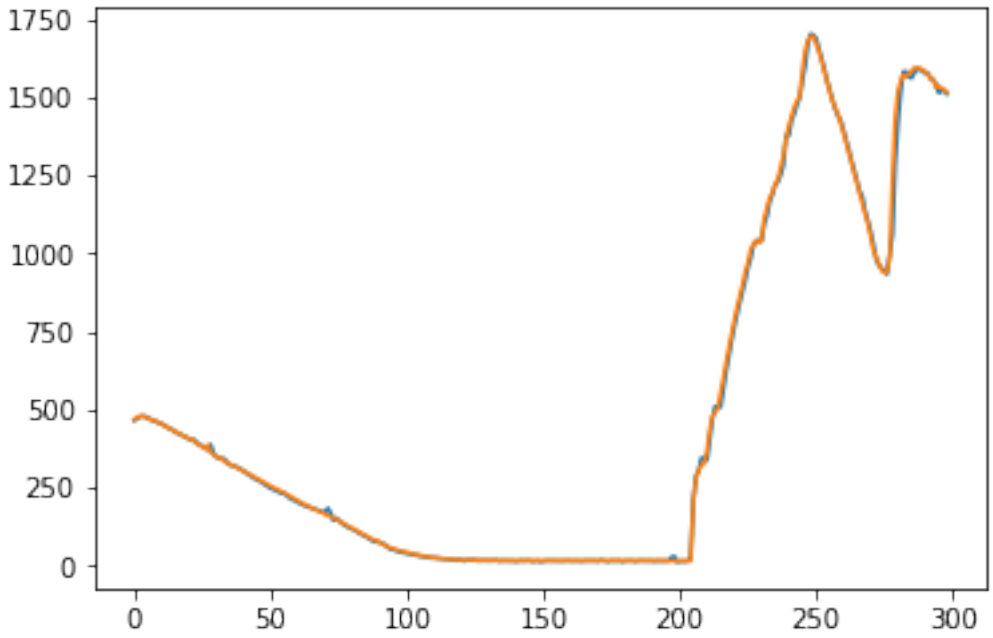
```
Out[202]: (5833,)
```

```
In [222]: inp = X[-300:-1, :, :]
```

```
In [223]: r = model.predict(inp)
          res.append(r)
```

```
In [224]: plt.plot(r)
          plt.plot(y[-300:-1])
```

```
Out[224]: [<matplotlib.lines.Line2D at 0x7f0e18acb198>]
```



```
In [244]: model_json = model.to_json()
          with open("model.json", "w") as json_file:
              json_file.write(model_json)


          model.save_weights("model.h5")

Saved model to disk


In [227]: model.summary()

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 50)                10800
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,851
Trainable params: 10,851
Non-trainable params: 0
_________________________________________________________________


In [7]: from IPython.display import SVG
        from keras.utils.vis_utils import model_to_dot
        from keras.utils import plot_model

In [1]: # SVG(model_to_dot(model, show_shapes=True).create(prog='dot', format='svg'))

In [1]: from keras.models import load_model

In [4]: from keras.models import model_from_json
        json_file = open('model.json', 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        model = model_from_json(loaded_model_json)

        model.load_weights("model.h5")

In [5]: # model = load_model("model.h5")

In [ ]:
```