“SOLAR POWER PREDICTION USING DEEP LEARNING TECHNIQUES”

M.TECH PROGRAMMING (5YEARS)

BIG DATA ANALYTICS

SUBMISSION OF REVIEW - III

SLOT- **C1+TC1**

FACULTY IN CHARGE

PROF. PARIMALA M

Department of school of Information Technology

SUBMITTED BY

SADHU NARAYANA NAIDU- 16MIS0157

M SAI DEEPU - 16MIS0133

K MADHAN - 16MIS0063

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CONTENTS

Abstract:

Being in a world where predicting the future is at elbow's width, predicting the various aspects of the nature is also most important. On the other hand, today's world is in a state where nothing can run smoothly without the electricity. The various resources used in the generation of the electricity like the renewable and non – renewable energies are depleting at exponential rates which may lead to the non- renewable energies vanish completely. This kind of prediction is moving the mankind towards renewable energies where the rate of production is completely unpredictable. So in order to make the most unpredictable production rates to predictable production rates. in this paper the concept of the neural networks is used. To distinguish a better model for more accurate results in case of the forecasting various models like Recurrent neural networks and simple linear regression models along with windows feature selection were compared. From the experimental model that was developed in forecasting the solar power based on the existing values taken from the datasets made available on the governmental website https://nsrdb.nrel.gov/nsrdb-viewer varying from the year 1998 to the year 2012. Among the entire collected datasets, the years 1998 to 2010 used for training whereas the rest of the years are used for testing.

1. INTRODUCTION TO BIG DATA

i) Definition of big data:

Big data is where the data volume, acquisition velocity, or data representation limits the ability to perform effective analysis using traditional relational approaches or requires the use of significant scaling (mode nodes) for efficient processing

ii) Facts about the big data:

- Big Data Needs a Diverse Culture
- Hadoop Isn't the Holy Grail
- The Real Driver Behind Big Data Is the People Within the Organization
- There Is No Place Where Big Data Doesn't Exist
- Big Data Technicians Will Disappear, You Better Start Looking Around
- Big Data Does Require Big Security Measurements
- A Public Debate About Privacy Issues Is Inevitable
- Venture Capital Firms Are Investing Massively in Big Data Startups

- Governments Are Increasing Big Data Technology All Over the World
- Big Data IT Spending Will Grow But Remain Small Compared to Total IT Spending **iii) V's:**

Volume, Veracity, Velocity, Variety, Value Volume:

Amount of global digital data created, replicated and consumed. Big Data requires a large amount of storage space and organizations must constantly scale their hardware and software in order to accommodate increases.

Veracity:

Data resides in a variety of different formats including text, numbers, images, audio and video data.

Velocity:

New data is being generated quickly and organizations need to respond in real time. Ex.: Tele conversation, sensor data, stock exchange

Veracity:

(Accuracy) Data are noisy

Value:

Data itself is of no value unless it is processed

2. Data chosen for analysis:

i) Domain

Solar Power Forecasting

ii) Size of Data

300MB **iii) Format of**

Data

CSV Files

iv) Before Preprocessing The accurate

data size is around 286MB where it

contains the missing and mistaken data.

v) Method of Preprocessing

Data preprocessing: Data cleaning, data integration, data transformation, data reduction, data discretization.

vi) After Preprocessing

The raw data will be preprocessed before the data subjected to the predictive analysis.

We pre-process all the data instances as follows:

a) Remove all rows with any missing attributes.

b) Order the normal class attributes.

c) Remove the abnormal class attributes.

d) Selecting the finite rows excluding the INF and NaN rows.

3. Methods Chosen for analysis:

Recurrent neural networks, Linear Regression, Deep learning technique:

The time series based approach of the deep learning Recurrent neural network can be used for the effective predictive analysis. The so preprocessed will be subjected to the model training and then the model tries to predict the power values.

4.Description of the tool chosen for Analytics:

The Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include: - Live code - Interactive widgets - Plots - Narrative text - Equations - Images - Video

Model: Linear regression

In [63]:
```
from keras.models import Sequential
from keras.layers import Dense
import keras.backend as K
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

In [67]:
```
K.clear_session()

model= Sequential()
model.add(Dense(12,input_dim=1,activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

In [68]:
```
early_stop = EarlyStopping(monitor='loss',patience=1,verbose=1)
```

In [*]:
```
model.fit(X_train, y_train, epochs=200, batch_size=2, verbose=1, callbacks=[early_stop])
```

Model: Recurrent neural networks

In [132]:
```
from keras.models import Sequential
from keras.layers import Dense
import keras.backend as K
from keras.callbacks import EarlyStopping
from keras.layers import LSTM
```

In [133]:
```
X_train.shape
```

Out[133]:
```
(5688260, 1)
```

In [134]:
```
#3D tensor with shape (batch_size, timesteps, input_dim)
X_train[:, None].shape
```

Out[134]:
```
(5688260, 1, 1)
```

In [135]:
```
X_train_t = X_train[:, None]
X_test_t = X_test[:, None]
```

In [136]:
```

K.clear_session()
model = Sequential()

model.add(LSTM(6, input_shape=(1, 1)))

model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
```

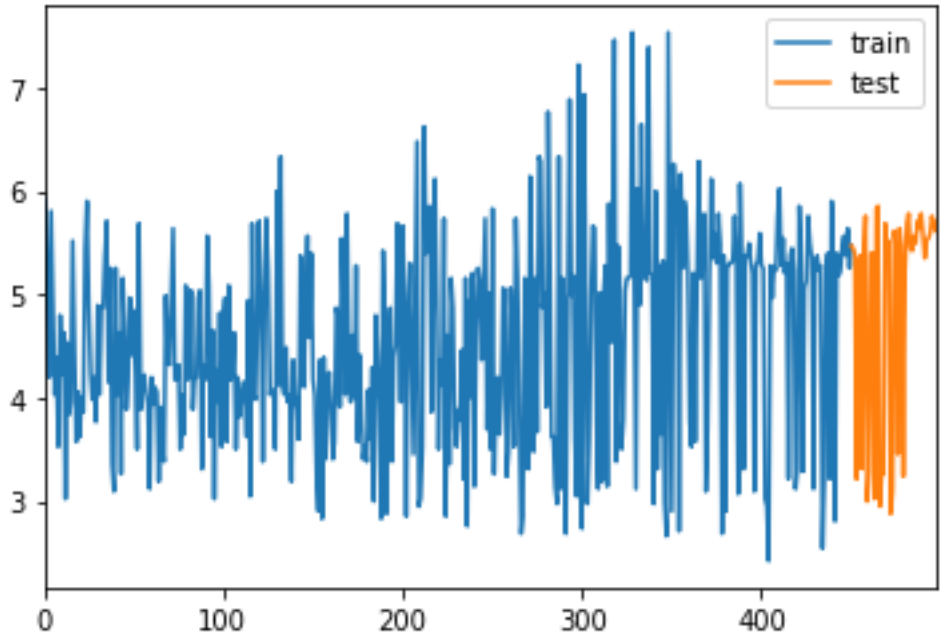
5. Results of Analytics:

```
In [72]: model.fit(X_train, y_train, epochs=200, batch_size=2, verbose=1, callbacks=[early_stop])
Epoch 1/200
450/450 [==============================] - 3s 7ms/step - loss: 0.2061A: 2
Epoch 2/200
450/450 [==============================] - 3s 6ms/step - loss: 0.0749
Epoch 3/200
450/450 [==============================] - 1s 3ms/step - loss: 0.0471
Epoch 4/200
450/450 [==============================] - 1s 3ms/step - loss: 0.0403
Epoch 5/200
450/450 [==============================] - 1s 3ms/step - loss: 0.0389
Epoch 6/200
450/450 [==============================] - 1s 3ms/step - loss: 0.0387
Epoch 7/200
450/450 [==============================] - 2s 3ms/step - loss: 0.0387
Epoch 8/200
450/450 [==============================] - 2s 4ms/step - loss: 0.0387A: 0s - los
Epoch 00008: early stopping
Out[72]: <keras.callbacks.callbacks.History at 0x21cea7c07b8>
```



```
In [78]: y_pred
Out[78]: array([[0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
                [0.4145258],
```

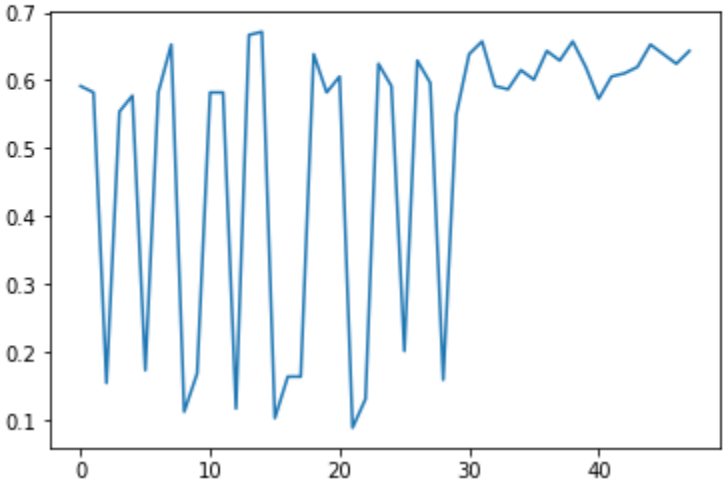
6. Visualization of Results:



Test data:

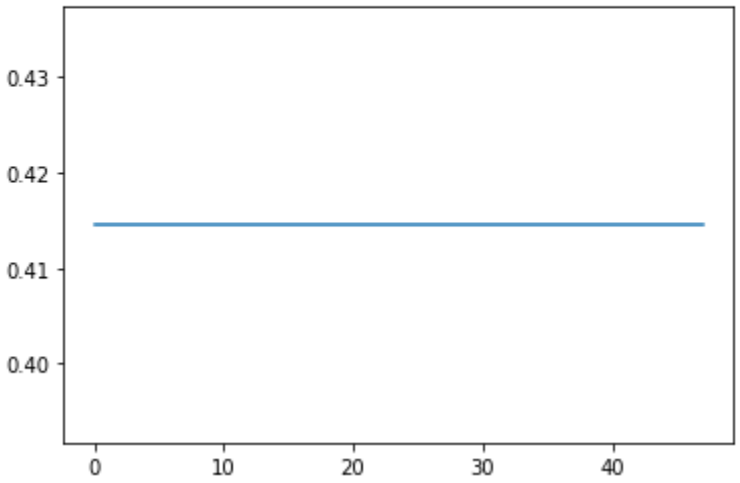
```
In [74]: plt.plot(y_test)

Out[74]: [<matplotlib.lines.Line2D at 0x21cead392e8>]
```



Predicted data:

```
In [75]: plt.plot(y_pred)
Out[75]: [<matplotlib.lines.Line2D at 0x21ceae0d5f8>]
```



7. Advantages and disadvantages of Big data analytics:

Advantages:

- Cost cutting
- Better decision making
- New products and services
- Fraud detection
- Control online reputation

Disadvantages:

- Incompatible tools
- New approach
- Chances of failure
- Correlation errors

- Security and privacy concerns

8. Conclusion:

Having a wide variety of models existing in the current world, having an exquisite model to pred ict the unknown attribute is always appreciable. Here, as a part our project we referenced various models which can be used in the prediction of the solar power. Being at the edge of depleting th
e non- renewable energy sources, it will always be a great opportunity to predict the solar power that can be generated in the coming forth years. So the study made by us as a part of this project i ndicated that the linear regression model with the rolling windows gave a very less error in comp arison to the most widely used deep learning technique, the recurrent neural networks with LST M and the rolling windows for the feature selection. The linear regression gave a very less error r ate of 0.0052579984202879015. hence it is advisable to use the linear regression as the model in order to predict the unknown attribute. Though the time series pattern can enable the prediction o f the numerical based models, linear regression provides the best in comparison to the LSTM of t he recurrent neural networks.

9. References:

https://nsrdb.nrel.gov/nsrdb-viewer https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/ https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networksand-ways-to-optimize-gradient-95ae5d39529f https://arxiv.org/abs/1412.6980v8 https://scikit-learn.org/stable/data_transforms.html https://console.cloud.google.com/compute/instances?project=blissful-trail-235407&instancessize=50

APPENDIX – I

FULLY CONNECTED LINEAR REGRESSION MODEL

```
# coding: utf-8


# In[1]:


import pandas as pd

import numpy as np

get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt



# In[2]:


df= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_1998_ghi.csv")



# In[3]:


df.head()
```

```
# In[4]:


df.drop(['FID'],axis=1,inplace= True)

df['year']=1998



# In[5]:


# df



# In[6]:


df1= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_1999_ghi.csv")



# In[7]:


df1.drop(['FID'],axis=1,inplace= True)

df1['year']=1999
```

```
# In[8]:


# df1



# df2= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2000_ghi.csv")
# df2.drop(['FID'],axis=1,inplace= True)
# df2['year']=2000
# df2

# df3= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2001_ghi.csv")
# df3.drop(['FID'],axis=1,inplace= True)
# df3['year']=2001
# df3

# df4= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2002_ghi.csv")
# df4.drop(['FID'],axis=1,inplace= True)
# df4['year']=2002
# df4

# df5= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2003_ghi.csv")
```

```
# df5.drop(['FID'],axis=1,inplace= True)

# df5['year']=2003

# df5


# In[9]:


df6= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_2005_ghi.csv")

df6.drop(['FID'],axis=1,inplace= True)

df6['year']=2005

# df6



# df7= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2006_ghi.csv")

# df7.drop(['FID'],axis=1,inplace= True)

# df7['year']=2006

# df7


# In[10]:


df8= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_2007_ghi.csv")

df8.drop(['FID'],axis=1,inplace= True)

df8['year']=2007
```

```
# df8



# In[11]:


df9= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_2008_ghi.csv")

df9.drop(['FID'],axis=1,inplace= True)

df9['year']=2008

df9



# In[12]:


df10= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_2010_ghi.csv")

df10.drop(['FID'],axis=1,inplace= True)

df10['year']=2010



# In[13]:


df10
```

```
# df11= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2011_ghi.csv")

# df11.drop(['FID'],axis=1,inplace= True)

# df11['year']=2011


# df11


# In[14]:


df12= pd.read_csv("C://Users//sadhu narayana naidu//Desktop//big
data//data//nsrdb_v3_0_1_2012_ghi.csv")

df12.drop(['FID'],axis=1,inplace= True)

df12['year']=2012



# In[15]:


df12


# In[16]:


df.drop(['gid'],axis=1,inplace= True)
```

```
df1.drop(['gid'],axis=1,inplace= True)
# df2.drop(['gid'],axis=1,inplace= True)
# df3.drop(['gid'],axis=1,inplace= True)
# df4.drop(['gid'],axis=1,inplace= True)
# df5.drop(['gid'],axis=1,inplace= True)
df6.drop(['gid'],axis=1,inplace= True)
# df7.drop(['gid'],axis=1,inplace= True)
df8.drop(['gid'],axis=1,inplace= True)
df9.drop(['gid'],axis=1,inplace= True)
df10.drop(['gid'],axis=1,inplace= True)
# df11.drop(['gid'],axis=1,inplace= True)
df12.drop(['gid'],axis=1,inplace= True)


# In[17]:


df=df.append(df1, ignore_index=True)


# In[18]:


# df=df.append(df2, ignore_index=True)
# df
```

```
# In[19]:


# df=df.append(df3, ignore_index=True)
# df



# In[20]:


# df=df.append(df4, ignore_index=True)
# df



# In[21]:


# df=df.append(df5, ignore_index=True)
# df



# In[22]:


df=df.append(df6, ignore_index=True)
```

```
df



# In[23]:


# df=df.append(df7, ignore_index=True)
df=df.append(df8, ignore_index=True)
df=df.append(df9, ignore_index=True)
df



# In[24]:


df=df.append(df10, ignore_index=True)
df=df.append(df12, ignore_index=True)



# In[25]:


df = df[np.isfinite(df['ghi'])]
df
```

```
# In[26]:


# split_date=5688265
split_date=1661183
train = df.loc[:split_date, ['ghi']]
test = df.loc[split_date:, ['ghi']]



# In[27]:


ax = train.plot()
test.plot(ax=ax)
plt.legend(['train', 'test'])



# In[28]:


x=df['year']
y=df['ghi']
plt.plot(x, y, '-ok', color='black');



# ax=train.plot(x='year',y='ghi',figsize=(16,8))
```

```
# test.plot(ax=ax,x='year',y='ghi',figsize=(16,8))
# plt.legend(['test','train'])

# In[29]:

from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()

train_sc= sc.fit_transform(train)
test_sc= sc.transform(test)

# In[30]:

train_sc[:4]

# In[31]:

X_train = train_sc[:-1]
y_train = train_sc[1:]

X_test = test_sc[:-1]
```

```
y_test = test_sc[1:]


# In[32]:


from keras.models import Sequential
from keras.layers import Dense
import keras.backend as K
from keras.callbacks import EarlyStopping


# In[33]:


K.clear_session()


model= Sequential()
model.add(Dense(12,input_dim=1,activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')


# In[ ]:
```

```
early_stop = EarlyStopping(monitor='loss',patience=1,verbose=1)



# In[ ]:


model.fit(X_train, y_train, epochs=200, batch_size=2, verbose=1, callbacks=[early_stop])



# In[ ]:


y_pred = model.predict(X_test)



# In[ ]:


plt.plot(y_test)
plt.plot(y_pred)



# In[ ]:


from sklearn.metrics import mean_squared_error
```

```
# In[ ]:

mean_squared_error(y_test, y_pred)



# In[ ]:

y_pred
# In[ ]:
y_test
# In[ ]:
```

APPENDIX - II

FULLY CONNECTED RECURRENT PREDICTOR

```
# coding: utf-8 # In[94]:

import pandas as pd import numpy as np
get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt # In[95]:
```

```
df= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_1998_ghi.csv") # In[96]:


df.head() # In[97]:


df.drop(['FID'],axis=1,inplace= True) df['year']=1998


# In[98]:
df # In[99]:


df1= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_1999_ghi.csv") # In[100]:


df1.drop(['FID'],axis=1,inplace= True)


df1['year']=1999 # In[101]:


df1
```

```
# In[102]:

df2= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2000_ghi.csv")

df2.drop(['FID'],axis=1,inplace= True) df2['year']=2000 df2 # In[103]:



df3= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2001_ghi.csv")

df3.drop(['FID'],axis=1,inplace= True) df3['year']=2001 df3 # In[104]:



df4= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2002_ghi.csv")
df4.drop(['FID'],axis=1,inplace= True) df4['year']=2002 df4

# In[105]:



df5= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2003_ghi.csv")

df5.drop(['FID'],axis=1,inplace= True) df5['year']=2003 df5 # In[106]:



df6= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2005_ghi.csv")

df6.drop(['FID'],axis=1,inplace= True) df6['year']=2005 df6 # In[107]:
```

```
df7= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2006_ghi.csv")

df7.drop(['FID'],axis=1,inplace= True) df7['year']=2006 df7

# In[108]:



df8= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2007_ghi.csv")

df8.drop(['FID'],axis=1,inplace= True) df8['year']=2007 df8 # In[109]:



df9= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2008_ghi.csv")

df9.drop(['FID'],axis=1,inplace= True) df9['year']=2008 df9 # In[110]:



df10= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2010_ghi.csv") df10.drop(['FID'],axis=1,inplace= True)

df10['year']=2010 # In[111]:



df10 # In[112]:
```

df11= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2011_ghi.csv")

df11.drop(['FID'],axis=1,inplace= True) df11['year']=2011 # In[113]: df11

In[114]:

df12= pd.read_csv("D:/Desktop/data/nsrdb_v3_0_1_2011_ghi.csv")

df12.drop(['FID'],axis=1,inplace= True) df12['year']=2012 # In[115]: df12 #

In[116]:

df.drop(['gid'],axis=1,inplace= True) df1.drop(['gid'],axis=1,inplace= True)
df2.drop(['gid'],axis=1,inplace= True) df3.drop(['gid'],axis=1,inplace= True)
df4.drop(['gid'],axis=1,inplace= True) df5.drop(['gid'],axis=1,inplace= True)
df6.drop(['gid'],axis=1,inplace= True) df7.drop(['gid'],axis=1,inplace= True)
df8.drop(['gid'],axis=1,inplace= True) df9.drop(['gid'],axis=1,inplace= True)
df10.drop(['gid'],axis=1,inplace= True) df11.drop(['gid'],axis=1,inplace= True)
df12.drop(['gid'],axis=1,inplace= True) # In[117]:

df=df.append(df1, ignore_index=True) df # In[118]:

df=df.append(df2, ignore_index=True) df # In[119]:

df=df.append(df3, ignore_index=True) df

In[120]:

df=df.append(df4, ignore_index=True) df # In[121]:

df=df.append(df5, ignore_index=True) df # In[122]:

df=df.append(df6, ignore_index=True) df

In[123]:

df=df.append(df7, ignore_index=True)

df=df.append(df8, ignore_index=True)

df=df.append(df9, ignore_index=True)

df=df.append(df10, ignore_index=True)

```
df=df.append(df11, ignore_index=True)

df=df.append(df12, ignore_index=True) df # In[124]:

df # In[125]:



df = df[np.isfinite(df['ghi'])] df



# In[126]:

split_date=5688265

train = df.loc[:split_date, ['ghi']] test =

df.loc[split_date:, ['ghi']] # In[127]:



ax = train.plot() test.plot(ax=ax)

plt.legend(['train', 'test']) # In[128]:



x=df['year'] y=df['ghi'] plt.plot(x, y, '-ok',

color='black');
```

```
# ax=train.plot(x='year',y='ghi',figsize=(16,8))
# test.plot(ax=ax,x='year',y='ghi',figsize=(16,8))
# plt.legend(['test','train'])
# In[129]:

from sklearn.preprocessing import MinMaxScaler sc=MinMaxScaler()

train_sc= sc.fit_transform(train) test_sc=

sc.transform(test) # In[130]: train_sc[:4] #

In[131]:

X_train = train_sc[:-1] y_train =
train_sc[1:]

X_test = test_sc[:-1] y_test = test_sc[1:]
# In[132]:
```

```
from keras.models import Sequential from keras.layers

import Dense import keras.backend as K from

keras.callbacks import EarlyStopping from keras.layers

import LSTM # In[133]: X_train.shape # In[134]:


#3D tensor with shape (batch_size, timesteps, input_dim)
X_train[:, None].shape


# In[135]:
X_train_t = X_train[:, None]
X_test_t = X_test[:, None] # In[136]:


K.clear_session()
model = Sequential()

model.add(LSTM(6, input_shape=(1, 1)))
```

```
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam') # In[ ]:


early_stop = EarlyStopping(monitor='loss',patience=1,verbose=1) # In[ ]:


model.fit(X_train_t, y_train, epochs=100, batch_size=1,
        verbose=1,
        callbacks=[early_stop]) # In[ ]:


y_pred = model.predict(X_test_t)
plt.plot(y_test)
plt.plot(y_pred)
```

THANKYOU