UNIVERSITY COLLEGE DUBLIN

STAT40800 : Data Programming with Python

PROJECT REPORT

on

AIR POLLUTION FORECASTING USING LSTM IN KERAS

Submitted by,

Justin Joseph

Student Number : 18201354

Submitted on : Dec 09, 2018

INTRODUCTION

Air pollution is the contamination of air by harmful substances such as gases, particulates, and biological molecules. It is mainly caused by burning of fossil fuels, agricultural activities (such as excessive use of pesticides, insecticides and fertilizers), exhaust from factories and industries, mining activities and due to household activities. Air pollution can cause respiratory and heart problems in human beings. It is also the main reason for serious issues, such as global warming, acid rain, eutrophication and depletion of the ozone layer. Air pollution is a very serious issue in the modern world and it is necessary to keep the levels of pollution to a minimum for sustainable life

PM2.5 is one of the attributes that measures the quality of air and in this project we use PM2.5 as the measure for air pollution. PM2.5 particles have diameters less than or equal to 2.5 microns and are also called fine particles. They are very light particles and tend to stay longer in the air. Due to the small size, these particles can travel more deeper into the lungs and cause very harmful effects to life.

Forecasting the PM2.5 levels will give an insight about the upcoming pollution levels and using this, necessary measures can be taken in advance to tackle the problem of air pollution. In this project, we will forecast the pollution at the next hour using the weather conditions and pollution of the prior hours. The LSTM model of KERAS package in python is used for forecasting.

ABOUT THE DATASET

The dataset used is “Beijing PM2.5” dataset and it is obtained from https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data . It has hourly reports on weather and the level of pollution for 5 years (from Jan 1st, 2010 to Dec 31st, 2014) at the US Embassy in Beijing, China. The columns in the dataset are:

1. No - Row Number
2. year - Year of data in this row
3. month - Month of data in this row
4. day - Day of data in this row
5. hour - Hour of data in this row
6. pm2.5 - PM2.5 concentration
7. DEWP - Dew point
8. TEMP - Temperature
9. PRES - Pressure
10. cbwd - Combined wind direction

11. lws - Cumulated wind speed
12. ls - Cumulated hours of snow
13. lr - Cumulated hours of rain

NEW PACKAGES USED

1. **Keras**

   Keras is an open source high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK or Theano. The advantages of Keras are:

   - Allows for easy and fast prototyping (through user friendliness, modularity and extensibility).
   - Supports both convolutional networks and recurrent networks, as well as combinations of the two.
   - Runs seamlessly on CPU and GPU.

2. **Bokeh**

   Bokeh is an interactive visualization python library that targets modern web browsers for presentation. Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets. Anyone can use to easily and quickly create interactive plots, dashboards and data applications.

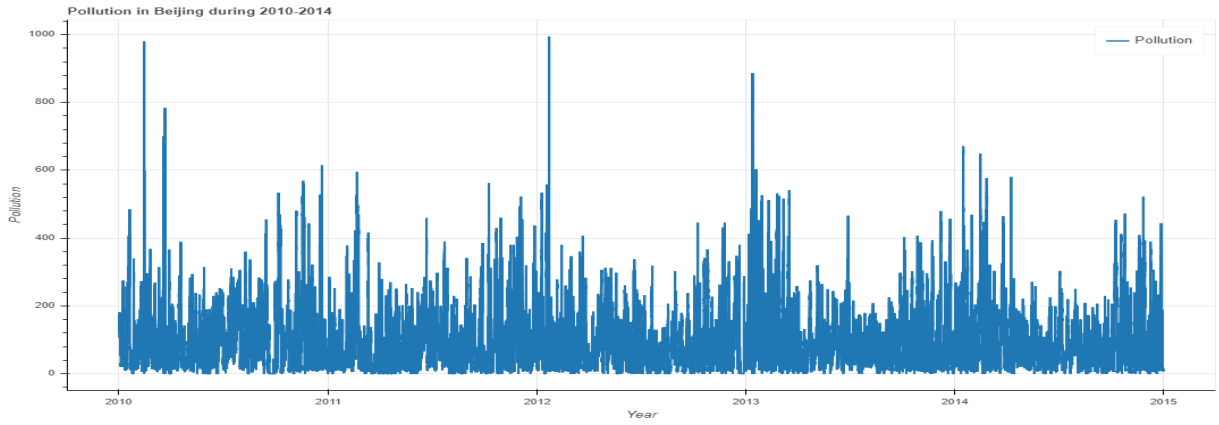
PROCEDURE

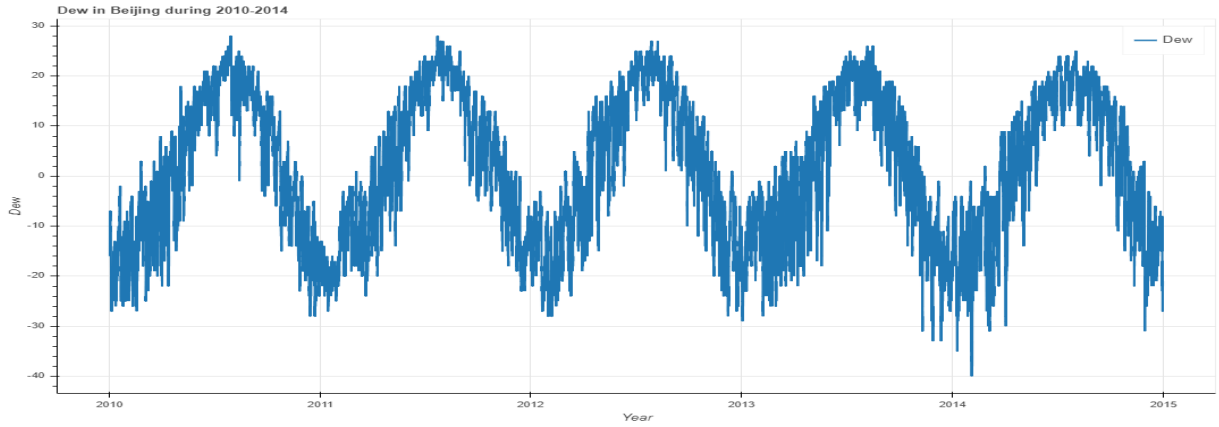
1. **Load the dataset, reformat the dataset and visualization**

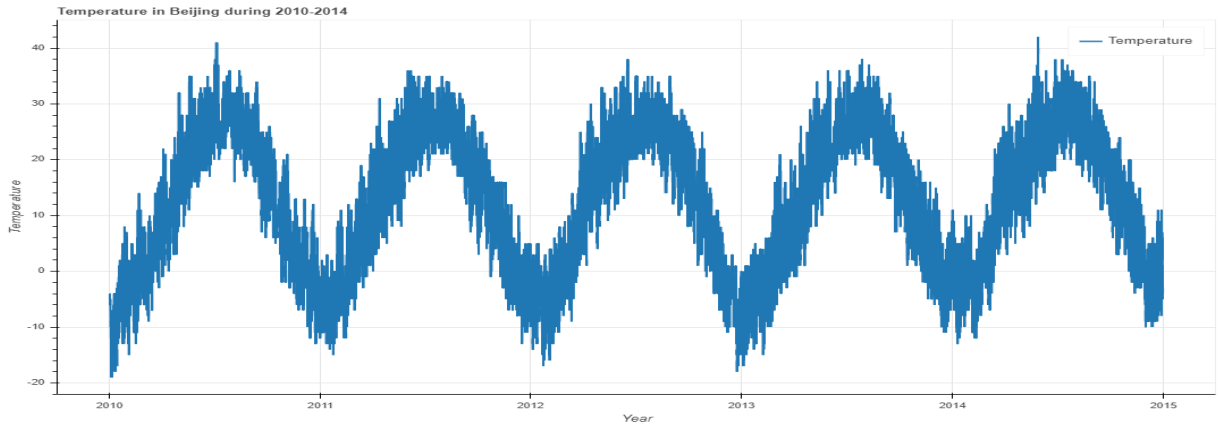
   The date and time information can be clubbed into a single date-time information which can be used as the index for the dataframe, and this is our first step. The data is loaded into python using **read_csv** from the **Pandas** library. The first column "**No**" is of no importance for the forecasting and hence is dropped from the dataset. Also the columns are renamed with better names. Also since the first 24 hours of data has **NA** values, it is removed. Rest of the **NA** values are replaced with **0**. The code below is used to perform this operations.

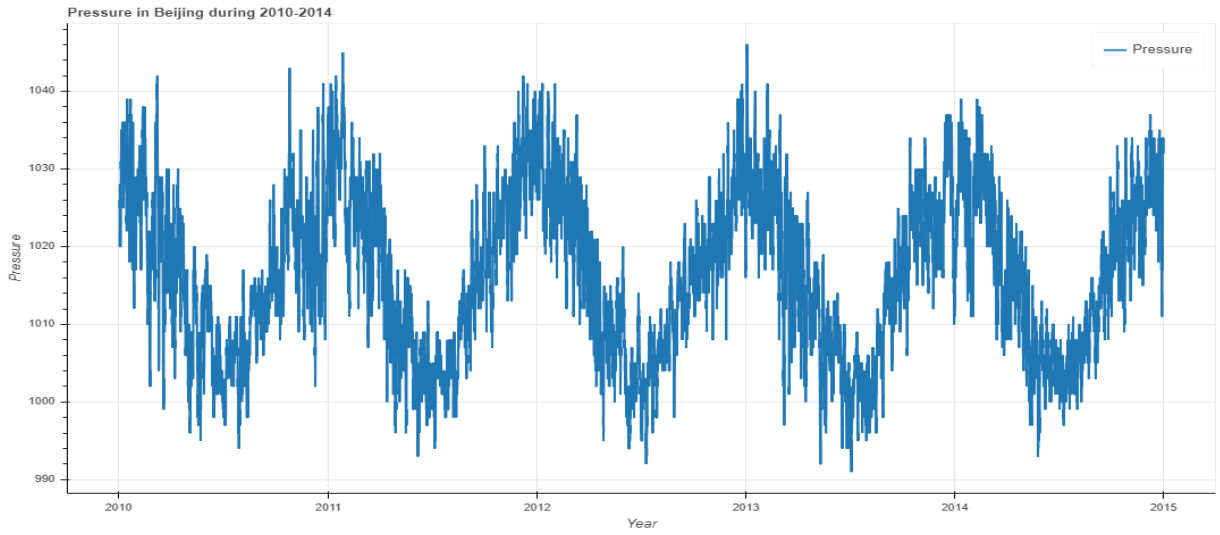
```
In [ ]: data=pd.read_csv('C:/Users/HP/Downloads/Study/Python/Assignment/Project 2/raw.csv',parse_dates=[['year','month','day',
                    'hour']],index_col=0,date_parser=date_reformat)
        data.drop('No',axis=1,inplace=True) #Drop the column "No" as it has no significance in prediction
        data.columns=['Pollution','Dew','Temperature','Pressure','Wind Direction','Wind Speed','Snow','Rain'] #Rename the columns
        data.index.name='Date' #Rename the index
        data=data[24:] #Drop the first 24 hours of data as it has NA values
        data['Pollution'].fillna(0,inplace=True) #Replace all the other NA values with 0
```

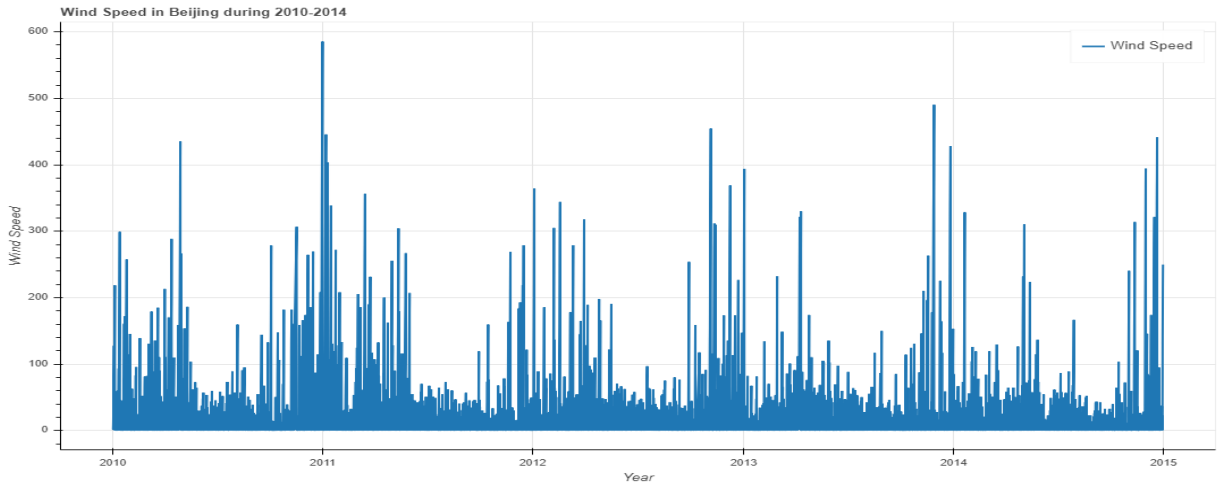
Now using the **Bokeh** plot, we can make the plot for all the 7 variables for 5 year time period.

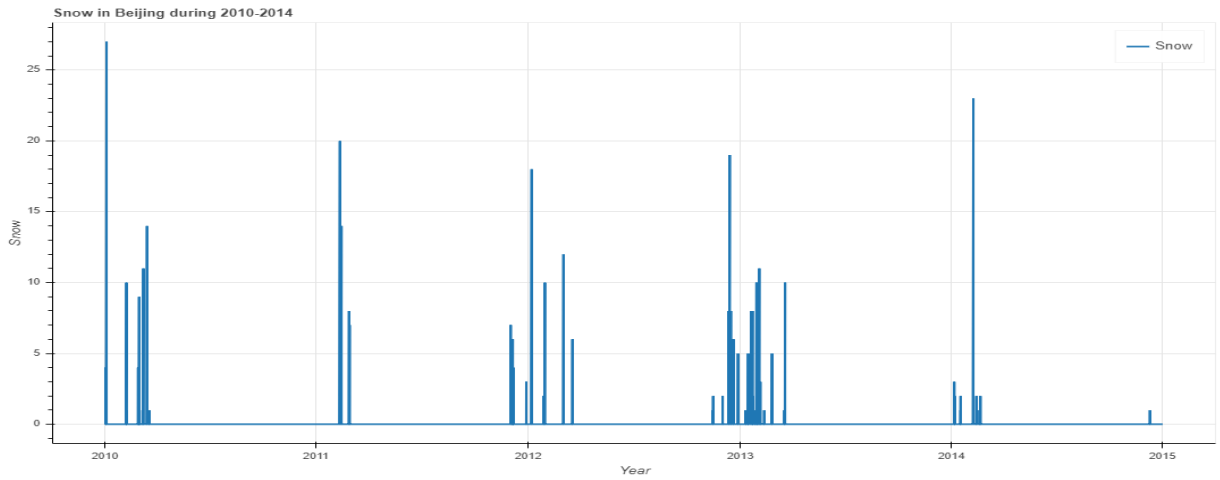


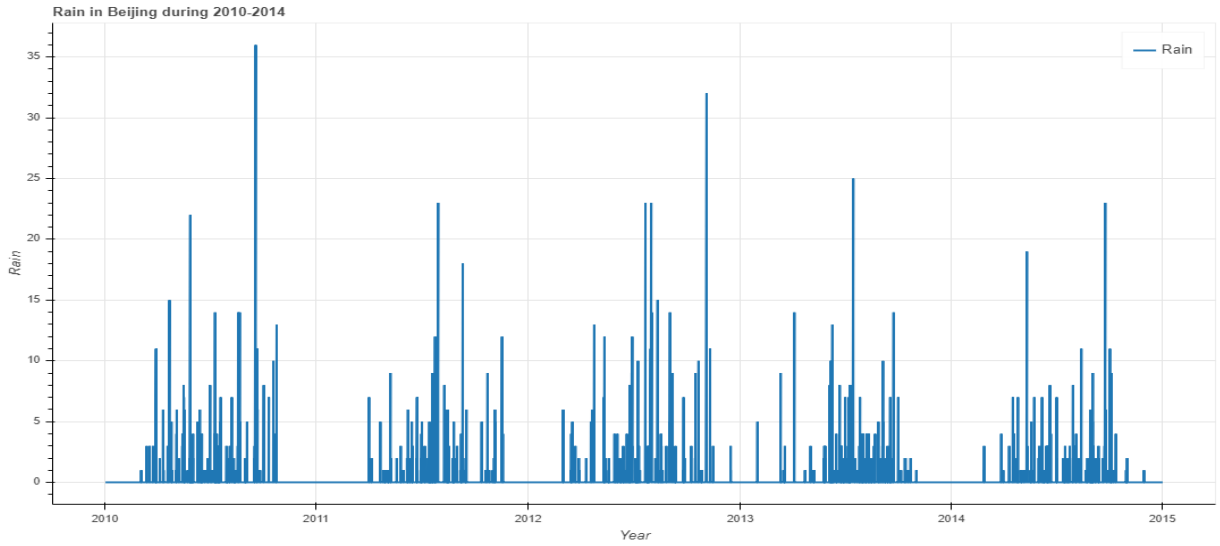





Pressure in Beijing during 2010-2014
Pressure
1040
1030
1020
1010
1000
990
Pressure
2010
2011
2012
2013
2014
2015
Year


Wind Speed in Beijing during 2010-2014
Wind Speed
600
500
400
300
200
100
0
Wind Speed
2010
2011
2012
2013
2014
2015
Year


Snow in Beijing during 2010-2014
Snow
25
20
15
10
5
0
Snow
2010
2011
2012
2013
2014
2015
Year



The below code is used to make the plots:

```
In [ ]: bp.output_file("Pollution.html")
p=bp.figure(title='Pollution in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Pollution',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,0],legend=data.columns[0],line_width=2)
bp.show(p)

bp.output_file("Dew.html")
p=bp.figure(title='Dew in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Dew',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,1],legend=data.columns[1],line_width=2)
bp.show(p)

bp.output_file("Temperature.html")
p=bp.figure(title='Temperature in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Temperature',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,2],legend=data.columns[2],line_width=2)
bp.show(p)

bp.output_file("Pressure.html")
p=bp.figure(title='Pressure in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Pressure',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,3],legend=data.columns[3],line_width=2)
bp.show(p)

bp.output_file("Wind Speed.html")
p=bp.figure(title='Wind Speed in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Wind Speed',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,5],legend=data.columns[5],line_width=2)
bp.show(p)
```

```
bp.output_file("Snow.html")
p=bp.figure(title='Snow in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Snow',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,6],legend=data.columns[6],line_width=2)
bp.show(p)

bp.output_file("Rain.html")
p=bp.figure(title='Rain in Beijing during 2010-2014',x_axis_label='Year',y_axis_label='Rain',
            x_axis_type='datetime',plot_width=1100)
p.line(data.index,val[:,7],legend=data.columns[7],line_width=2)
bp.show(p)
```

2. Data Preparation

The categorical variable "**Wind Speed**" is converted to numerical format using encoding.

```
encode=skp.LabelEncoder()
val[:,4]=encode.fit_transform(val[:,4])
```

All the variables are then normalized using the code below:

```
scale=skp.MinMaxScaler(feature_range=(0,1))
scaled_val=scale.fit_transform(val)
```

The time series problem is then converted to a supervised learning problem. The supervised learning problem is framed such that the pollution of the current hour is predicted given the pollution and weather conditions at the prior time. Below function is used for this.

```
def timeseries_to_supervised(d,input=1,output=1): #Function to convert timeseries to supervised structure
    vars=1 if type(d) is list else d.shape[1]
    df=pd.DataFrame(d)
    cols,names=list(),list()
    #
    for i in range(input,0,-1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(vars)]
    #
    for i in range(0,output):
        cols.append(df.shift(-i))
        if i==0:
            names += [('var%d(t)' % (j+1)) for j in range(vars)]
        else:
            names += [('var%d(t+%d)' % (j+1,i)) for j in range(vars)]
    aggregate=pd.concat(cols,axis=1)
    aggregate.columns=names
    return aggregate
```

3. Define the Model

The data is split into train and test dataset before defining the model. Training is done with 80% of the total data (4 years of data) and the rest of the data (1 year) is used for testing.

```
# Split dataset into train and test datasets
val_reformat=reformatted.values
train_hours=(365*24)*4 # Train with Four years of data
train=val_reformat[:train_hours, :]
test=val_reformat[train_hours:, :]

# Split into input and output
train_X,train_Y=train[:, :-1],train[:, -1]
test_X,test_Y=test[:, :-1],test[:, -1]
```

The Long Short-Term Memory (LSTM) recurrent neural network (RNN) can easily model problems with multiple input attributes. Hence this is a better choice to model time series forecasting problems.

A LSTM with 100 neurons in the first hidden layer and 1 neuron in the output layer is defined for prediction. Mean Absolute Error (MAE) loss function and Adam optimizer is used for optimization.

```
a = km.Sequential()
    a.add(kl.LSTM(100, input_shape=(data.shape[1], data.shape[2])))
    a.add(kl.Dense(1))
    a.compile(loss='mae', optimizer='adam')
    return a
```

The input variable into the LSTM model is reshaped into 3D format as expected by the LSTM.

```
# Reshape into a 3D array suitable for the LSTM model
train_X=train_X.reshape((train_X.shape[0],1,train_X.shape[1]))
test_X=test_X.reshape((test_X.shape[0],1,test_X.shape[1]))
print(train_X.shape,train_Y.shape,test_X.shape,test_Y.shape)
```

The model is fit for 50 epochs. Each epoch will have a batch size of 72. By giving the **validation_data** argument, a track of the training loss and test loss is kept.

```
# Fit network
history=model.fit(train_X,train_Y,epochs=50,batch_size=72,validation_data=(test_X,test_Y),verbose=2,shuffle=False)
```

4. **Make Prediction using the Model**

After fitting the model, the model is used to forecast for the complete test data. The values are then inverted back to the ordinary form.

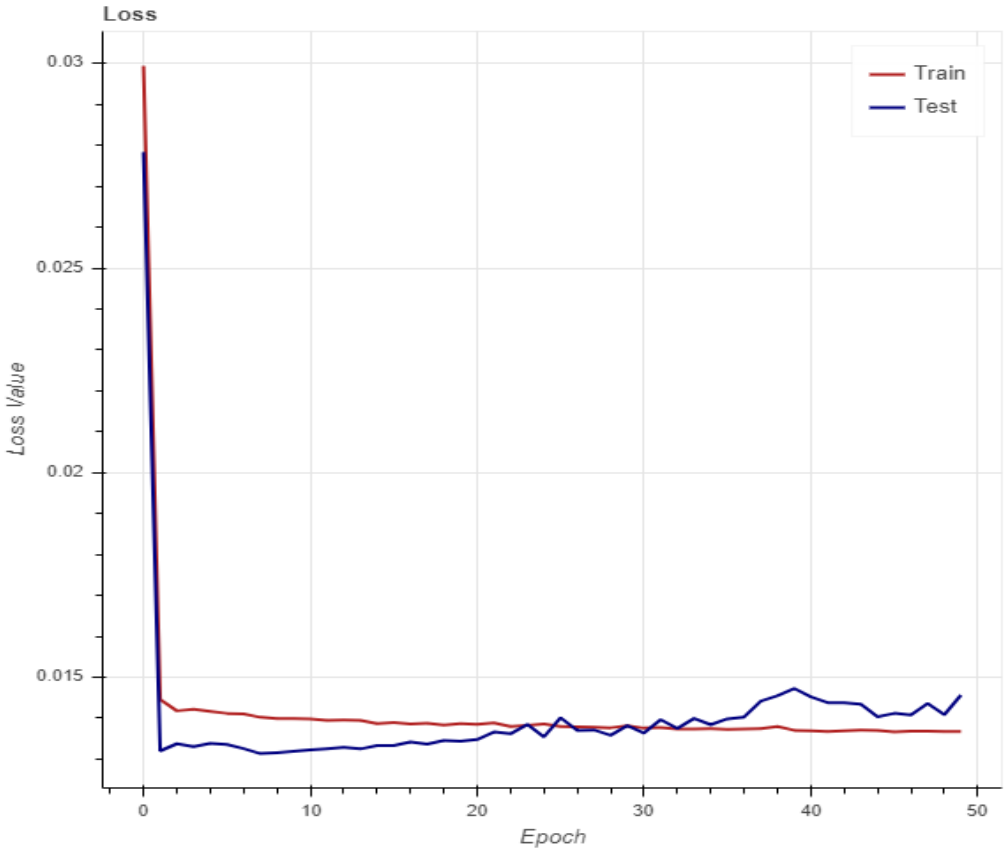
```
y_hat=model.predict(test_X)
test_X=test_X.reshape((test_X.shape[0],test_X.shape[2]))
test_Y=test_Y.reshape((len(test_Y),1))

# Invert scaling for actual
inv_Y=invert_scaling(test_X,test_Y)

# Invert scaling for forecast
inv_Y_hat=invert_scaling(test_X,y_hat)
```

Using the predicted value and the actual value, the root mean square error (RMSE) value is computed to find the error.

```
# Calculate RMSE
RMSE=mt.sqrt(skm.mean_squared_error(inv_Y,inv_Y_hat))
```

Using **Bokeh** plot, we can plot the train loss and the test loss that happens during the training.

```
bp.output_file("Loss.html")
p=bp.figure(title='Loss',x_axis_label='Epoch',y_axis_label='Loss Value')
p.line(list(range(len(history.history['loss']))),history.history['loss'],legend='Train',line_color='firebrick',line_width=2)
p.line(list(range(len(history.history['val_loss']))),history.history['val_loss'],legend='Test',line_color='navy',line_width=2)
bp.show(p)
```



CONCLUSION

The model achieves a RMSE of 26.17 which is pretty decent. From the above graph, it can be observed that the test loss is initially lesser than the train loss and then it fairly rises above the train loss.

REFERENCES

1. Dataset : https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data
2. Information on Keras : https://keras.io/
3. Information on Bokeh : https://bokeh.pydata.org/en/latest/