A Project Report

On

Detecting Air Pollution Levels using LSTMs and other Architectures

BY

Anshuman Pati : 2016B4A70470H

Under the supervision of

Dr. Manik Gupta

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

CS F366: LABORATORY ORIENTED PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(MARCH 2020)

ACKNOWLEDGMENTS

I would like to express my sincere thanks and deepest gratitude to Dr. Manik Gupta and the institute BITS Pilani Hyderabad Campus for giving me the wonderful opportunity to work on the lab oriented project on **Detecting Air Pollution Levels using LSTMs and other Architectures**.

I would like to thank Dr. Manik Gupta for her constant guidance during the project right from defining the problem statement to exploring different deep learning architectures to solve the problem and making a comparative analysis on these techniques. This has helped me understand what research is about and also helped me do a lot of research, explore and learn many new things in the process. I would also like to thank her for her constant support at each and every stage of the project.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled “**Detecting Air Pollution Levels using LSTMs and other Architectures”** submitted by Mr. ANSHUMAN PATI (ID No. 2016B4A70470H) in partial fulfillment of the requirements of the course CS F366, Lab Project Course, embodies the work done by him under my supervision and guidance.

Date: 15/03/2020 **(Dr. Manik Gupta)**

BITS- Pilani, Hyderabad Campus

ABSTRACT

The project is divided into two parts with one part focusing on air pollution level prediction for Hyderabad, India and the other part focusing on air pollution level prediction for Beijing, China. This provides a more accurate picture of how accurately our models predict the data for two completely different industrious cities which are governed by separate sets of legislations.

In the first part of our study we take a look at making hourly predictions for air pollution levels of Hyderabad with live data collected from several pollution sensors placed across different zones in Hyderabad. Based on the data collected we construct an LSTM model which makes multivariate predictions (PM2.5, dew, temperature, pressure, NO2, PM10, CO, Ozone, NOX, AQI) on an hourly basis so as to predict the level of air pollution in a locality and accurately determine the source of the emissions contributing to the pollution via sensor data. Moreover, the source of emissions can be validated using the satellite data of the emissions in that locality. We classify a given city into different zones based on the signature of the pollutant(s) it emits which is obtained from the satellite data as well as the predictions obtained about Air Quality Index (AQI) from deep learning architectures.

For the second part of the project we collect data from various stations across Beijing, namely Aotizhongxin, Changping, Dongsi, Guanyuan, Dingling, Guacheng, Huairou, Nongzhanguan, Shunyi, Tiantan, Wanliu, and Wanshouxigong. We perform various statistical and exploratory data analyses on the combined dataset. We then implement LSTM deep learning architecture on the proposed dataset and obtain the predictions and accuracy. After this we implement various novel Machine Learning algorithms and classifiers to classify the pollution level of the zones. These classifiers along with the LSTM model are then compared based on the accuracy of their prediction.

The entire repository can be found at https://github.com/AnshumanPati/Beijing-Air-Pollution-Prediction

This report can be found at https://drive.google.com/file/d/12Awb4-rtIpyixLGn9SS-Ew1E5q-lJY42/view?usp=sharing

CONTENTS

MOTIVATION

Over the past few decades, air pollution has risen as a global concern adhering to the rapid industrialization of the world, with more people being exposed to the risks associated with this modernisation. Recent advancements in the area of prediction using machine learning and deep learning algorithms have shown promise in terms of achieving better prediction accuracy for multiple domains. Air pollution has been a major concern in populous countries such as India and China since the last few years. Growing air pollution has led to an increase in health related ailments like stroke, heart disease, lung cancer etc. To monitor air pollution, particularly in major pollution prone cities, the government of India as well as several private companies have installed pollutant measuring sensors. The sensors regularly monitor the level of air pollutants like PM2.5, PM10, CO, NO2, SO2, O3. The data from these sensors can be used to effectively predict the air pollution level of a city, categorize pollution-prone cities and pinpoint the sources of emissions from a particular zone. This, in turn, can effectively reduce the problem of air pollution by identifying the sources of cities with bad Air Quality Index (AQI).

LITERATURE SURVEY

Over the past few decades, air pollution has risen as a global concern adhering to the rapid industrialization of the world, with more people being exposed to the risks associated with this modernisation. Recent advancements in the area of prediction using deep learning algorithms have shown promise in terms of achieving better prediction accuracy for multiple domains.

In [1], the authors have proposed models which are robust and have shown superiority over Artificial Neural Network model in predicting PM2.5 severity levels for multiple stations in New Delhi (www.who.int/phe/publications/air-pollution-global-assessment/en/, www.delhi.gov.in) for predicting upto 6 hours, 12 hours and 24 hours ahead of time. The authors have implemented Bi-directional LSTM network to model air pollutant severity for multiple locations in New Delhi, India. Their data incorporates the PM2.5 concentration along with several meteorological variables like temperature, rainfall, wind speed, etc. on an hourly basis. The authors have made a comparative study on how the implemented model was better than the standard ANN implemented framework for the same. However, the authors have predicted *severity* in their model rather than the actual values. This provides us with categorical attributes to identify the severity of air pollution in a locality but does not quantify this impact thereby making a comparative study (with several other models that can be implemented) implausible. The authors have carefully analyzed the advantages of using Bi-LSTMs owing to its structure to use both past and future influences for prediction. They have also explained how Air Quality Index (AQI) cannot be considered as a parameter for input since it is taken at 24 hours average time interval. Hence, the authors have considered PM2.5 and PM10 data for hourly data to estimate. Another drawback of this implementation is that the authors have provided rolling averages for missing values and abnormal values which could have been better filled up with regression data. The authors have stressed on how the multilayer perceptrons existing earlier could just predict the next value but could not influence the short and long term (seasonal or weekly) data. However, this method provides another drawback where the optimal delay for RNN architecture on temporal data is measured by trial and error which is not very efficient. Two separate networks, one in each direction could be obtained and trained on all inputs and averages.

In [2], the authors have used Keras, which is a high-level neural networks API written in Python and capable of running on top of Tensorflow, to build a neural network and run RNN with LSTM through Tensorflow. In fact, we will be using the same tools for our analysis and prediction. The training data used in the network is retrieved from the EPA (Environmental Protection Administration) of Taiwan from year 2012 to 2016 and is combined into 20-dimensions data; and the forecasting test data is the year 2017. The authors have conducted experiments to evaluate the forecasting value of PM2.5 concentration for the next four hours at 66 stations around Taiwan. The result shows that the proposed approach can effectively forecast the value of PM2.5. The authors have discarded any station which had lack of enough data for the model to act upon. The authors have filled in the missing values with rolling averages and used min-max normalization to properly scale the data. The missing values could have been better filled with regression outputs. The authors have compared the accuracy of their model to a model implementing Artificial Neural Network on the basis of RMSE values. One of the major drawbacks of this model is that LSTM can be used to forecast PM2.5 properly if the effect of climate or air pollution from abroad is not taken into account. More often than not this is not the case. Moreover, not enough data is present on the air pollutants emitted by transportation and industry. In the prediction of PM2.5 values in the more distant future through different atmospheric altitude airflow data, the fugitive or cumulative area of air pollutants can be learned more accurately. In the prediction of more distant PM2.5 values through different atmospheric height airflow data, the flow or cumulative characteristics of air pollutants can be learned more accurately. And it can provide accurate PM2.5 prediction data for a longer time.

In [3], the authors used LSTM model for multivariate time series forecasting in Keras and TensorFlow deep learning library in a Python SciPy environment with Machine Learning scikit-learn, Pandas, NumPy and Matplotlib libraries. The Raw pollution dataset of this work consists of many columns of CSV for many environmental factors concentrations merged to one CSV file with python merge script and minimized to pollution concentrations of each hour for the following: SO2, NO_2, NO, CO with the datetime. Later the dataset was minimized for two years (2015 and 2016) for best fitting the expected model in order to get the best forecasting for Madrid’s Air Quality. This paper framed the dataset to forecast the pollution concentrations at the next hour depending on the pollution concentrations of prior hours. The difficulty of predicting multivariate time series and the development of intelligent systems that can design predictive complex environmental dataset automatically requires a good Machine Learning model. To tackle such problems, the authors have proposed an approach based on Neural networks like Long Short-Term Memory (LSTM) in keras and TensorFlow environment. The proposed approach can process multivariate environmental time series data and is robust to the number of samples, numeric ranges of data etc.

Running the model many times with increasing the number of memory cells in a hidden layer validates the effectiveness of the approach in accomplishing multivariate time series forecasting. This model can be expanded to predict new environmental observations and also recommend varying the time step index of the sequence output and training epochs to see if there is a relationship between the index and how hard the problem is to learn. Records involving missing data were dropped during the preprocessing stage. This can be detrimental if the dataset is a sparse dataset.

In [4], the authors have proposed a Bayesian LSTM machine-learning model that utilizes proxy data including Aerosol Optical Depth (AOD) and meteorology which can explain 80% of the PM2.5 variability. The preliminary results show that air pollution control regulatory measures introduced in China and Beijing have reduced PM2.5 pollution in Beijing by 23% on average. The dataset comprises official hourly station-level PM2.5 concentration data from 1 January 2013 to 31 December 2017. In addition, the data also constitutes hourly PM2.5 concentration data recorded at the US Embassy, Beijing from 9 April 2008 to 30 June 2017. AOD data was collected from the NASA MODIS satellite database from 26 March 2008 to 21 May 2017. Eight features were selected based on data availability during the period of study. In addition, meteorology data, including temperature, pressure, humidity, visibility, precipitation, and wind speed, measured at the Beijing Capital International Airport, from 1 January 2008 to 31 December 2017, were collected. The pre-processed data is fed into a Bayesian deep learning model for training. In this study, the authors have focused on the Bayesian RNN, which is a particular type of Bayesian deep learning model capable of modelling time-series data. The proposed model was developed based on DeepMind Sonnet, a TensorFlow-based neural network library, and multiple linear regression, which is commonly used in econometric modelling, was selected as the baseline model. The experimental results were in coherence with the theory the authors have proposed. The R^2 value of the final fitted model was 80% when using the mean of the posterior over the network weight parameters for prediction; while the R^2 for the baseline model was 70% only. However, the proposed model has some drawbacks. Causal inference from observed data can be biased due to the lack of counterfactual outcomes. Moreover, there exists some selection bias with or without the regulatory interventions. The authors’ work only highlights the aggregate effect of APCR (Air Pollution Control Regulations) but not the individual effects. Additionally, limited data is another drawback.

METHODOLOGY

PART I: *Detecting and Predicting Air Pollution Levels of BITS Pilani, Hyderabad Campus*

Tools Used:

TensorFlow (Backend), Keras, Pandas, NumPy, Scikit-learn Libraries on Jupyter Notebook

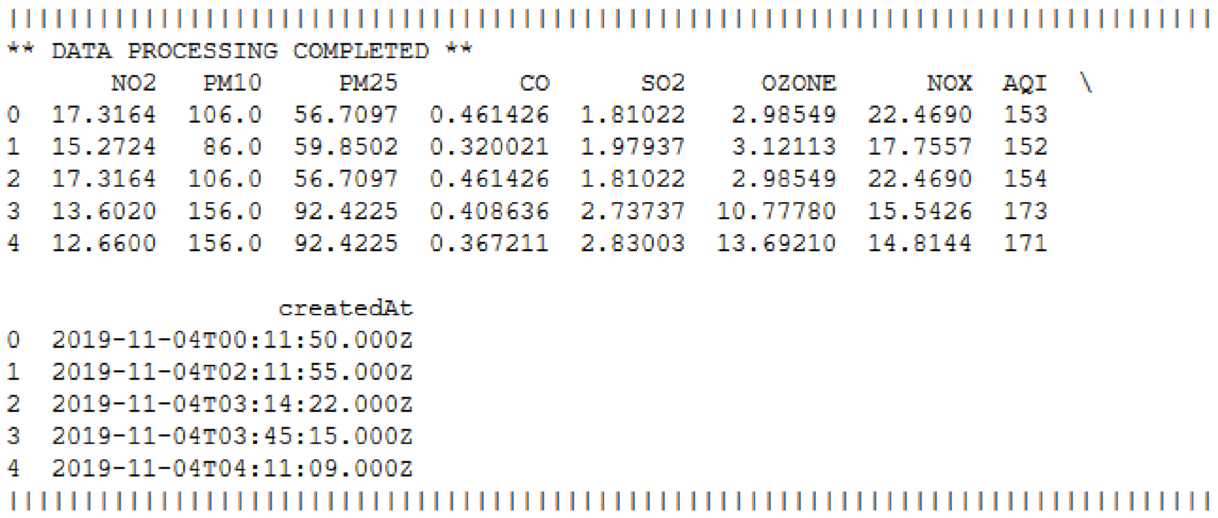
Steps followed:

1. Data preparation

   Data has been collected pertaining to BITS Pilani Hyderabad Campus zone at PIN Code : 500078 from real time sensors obtained from https://getambee.com/. The raw data collected has the following variables:
   - NO2 concentration
   - PM10 concentration
   - PM2.5 concentration
   - CO concentration
   - SO2 concentration
   - Ozone concentration
   - NOX concentration
   - Air Quality Index (AQI)
   - AQI category
   - Country
   - Place (or zone) name
   - Postal code
   - State
   - City
   - Division
   - Place Id
   - Latitude
   - Longitude
   - Date and exact time of recording

   Sampling Rate: 12 samples per day
   Data collected is for the months of November 2019, December 2019, and January 2020.

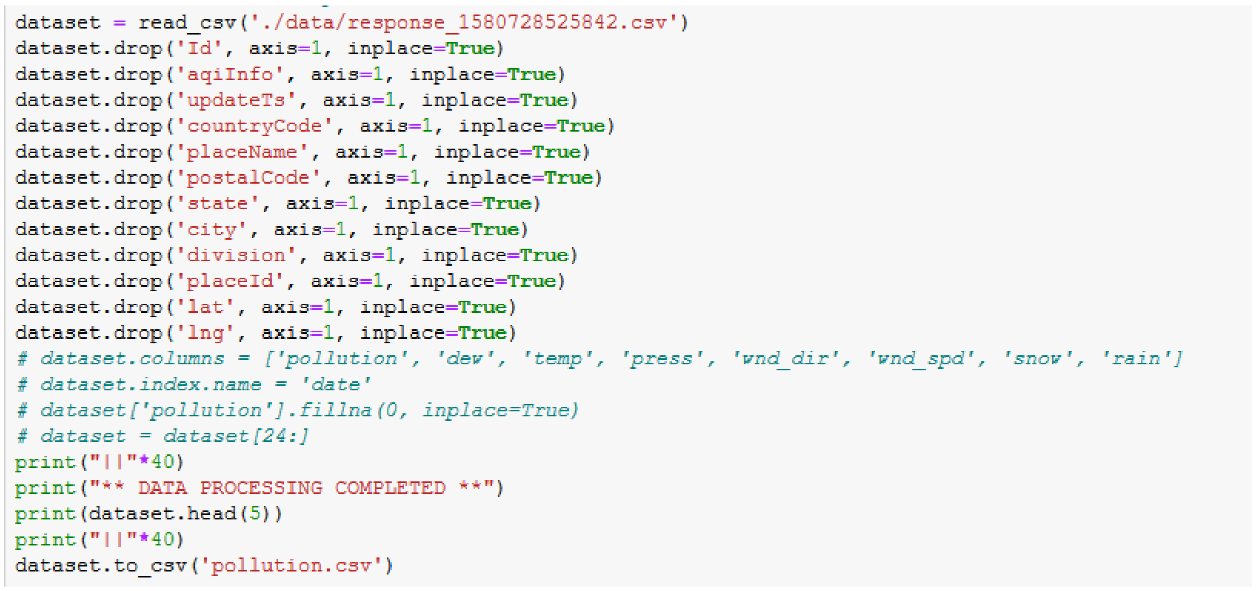
2. Data visualization



```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
** DATA PROCESSING COMPLETED **
       NO2   PM10     PM25        CO      SO2     OZONE      NOX  AQI  \
0  17.3164  106.0  56.7097  0.461426  1.81022   2.98549  22.4690  153
1  15.2724   86.0  59.8502  0.320021  1.97937   3.12113  17.7557  152
2  17.3164  106.0  56.7097  0.461426  1.81022   2.98549  22.4690  154
3  13.6020  156.0  92.4225  0.408636  2.73737  10.77780  15.5426  173
4  12.6600  156.0  92.4225  0.367211  2.83003  13.69210  14.8144  171

                  createdAt
0  2019-11-04T00:11:50.000Z
1  2019-11-04T02:11:55.000Z
2  2019-11-04T03:14:22.000Z
3  2019-11-04T03:45:15.000Z
4  2019-11-04T04:11:09.000Z
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

3. LSTM data preparation
4. Fit model along with regularization term
5. Evaluate model (From RMSE)

Data preparation:

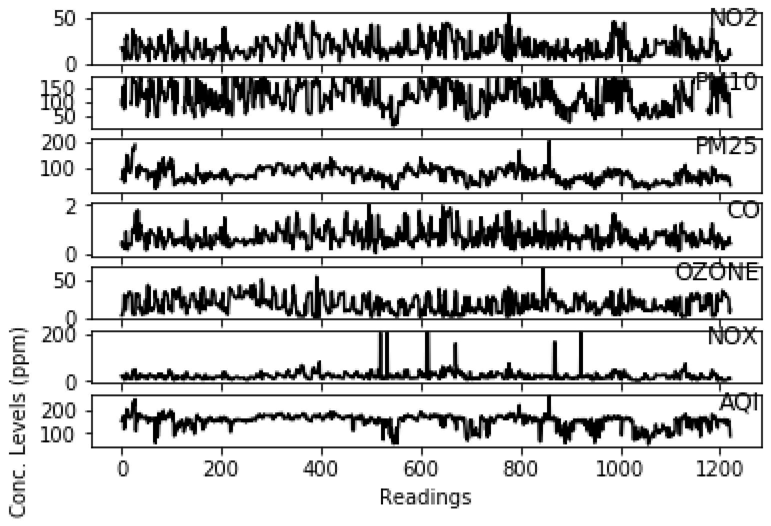


```python
dataset = read_csv('./data/response_1580728525842.csv')
dataset.drop('Id', axis=1, inplace=True)
dataset.drop('aqiInfo', axis=1, inplace=True)
dataset.drop('updateTs', axis=1, inplace=True)
dataset.drop('countryCode', axis=1, inplace=True)
dataset.drop('placeName', axis=1, inplace=True)
dataset.drop('postalCode', axis=1, inplace=True)
dataset.drop('state', axis=1, inplace=True)
dataset.drop('city', axis=1, inplace=True)
dataset.drop('division', axis=1, inplace=True)
dataset.drop('placeId', axis=1, inplace=True)
dataset.drop('lat', axis=1, inplace=True)
dataset.drop('lng', axis=1, inplace=True)
# dataset.columns = ['pollution', 'dew', 'temp', 'press', 'wnd_dir', 'wnd_spd', 'snow', 'rain']
# dataset.index.name = 'date'
# dataset['pollution'].fillna(0, inplace=True)
# dataset = dataset[24:]
print("||"*40)
print("** DATA PROCESSING COMPLETED **")
print(dataset.head(5))
print("||"*40)
dataset.to_csv('pollution.csv')
```

- Replace NA values
- Parse date-time into pandas dataframe index
- Specified clear names for each columns

Data visualization

- Used matplotlib to plot

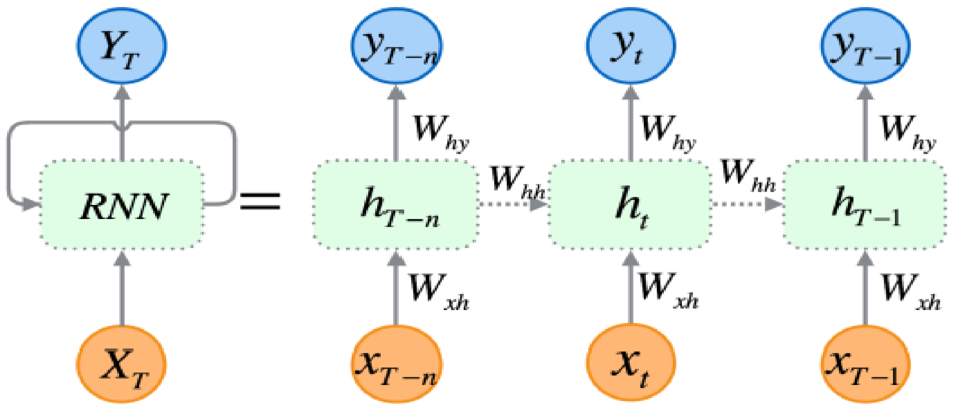


Corresponding to parameters obtained from BITS Pilani Hyderabad Campus

LSTM data preparation

LSTM:

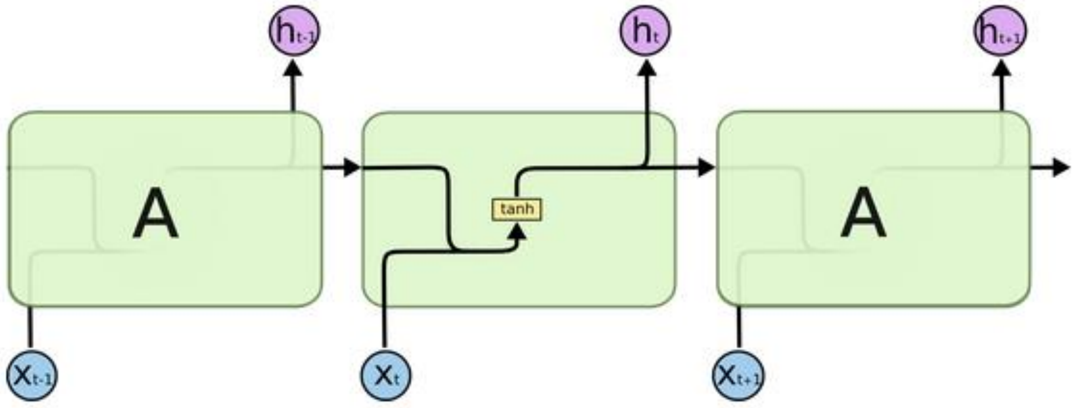
To understand LSTM (*Long Short Term Memory)* we first need to understand the basic RNN architecture.



1 Standard RNN architecture and an unfolded structure with T tim

In the above diagram, a chunk of neural network, A, looks at some input x_t and outputs a value h_t**.** This is especially useful in the case of temporal data handling since the outcome or prediction of a future event depends largely on the trend in the past. Moreover, as we can see in the diagram an unrolled RNN is just like a looped Neural Network. A loop allows information to be passed from one step of the network to the next.

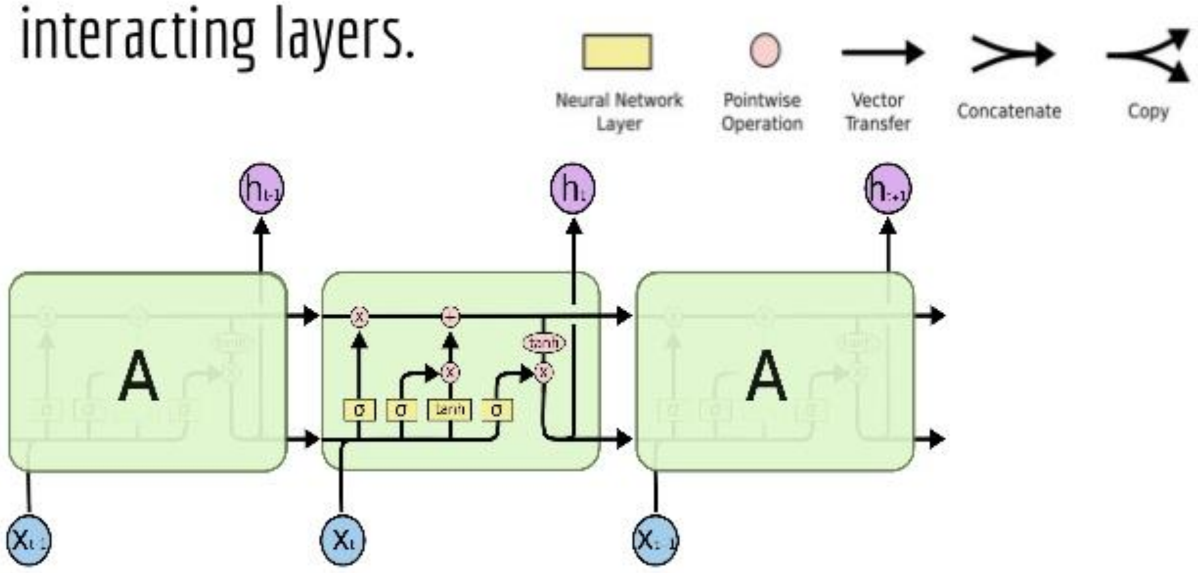
Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term predictions. All recurrent neural networks propagate as a chain of repeating modules. In standard RNNs, this repeating module is as simple as a single tanh layer.



The repeating module in a standard RNN contains a single layer.

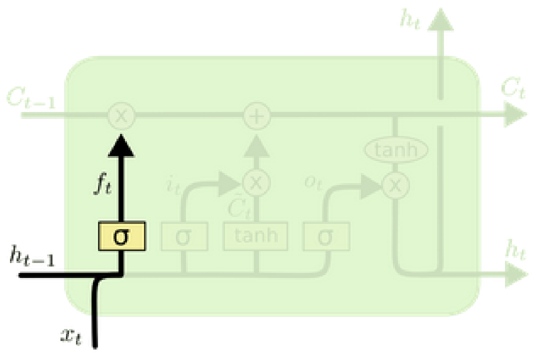
LSTMs too have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

The repeating module in an LSTM contains four interacting layers.

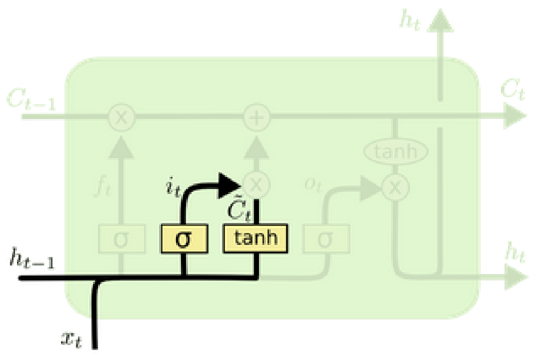


Each line in the above diagram carries an entire vector of input.

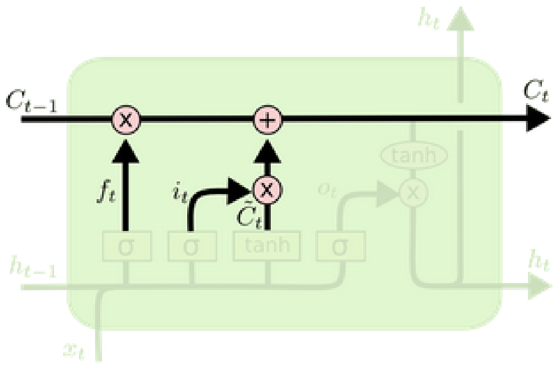
Functions Evaluated at each Layer of the LSTM



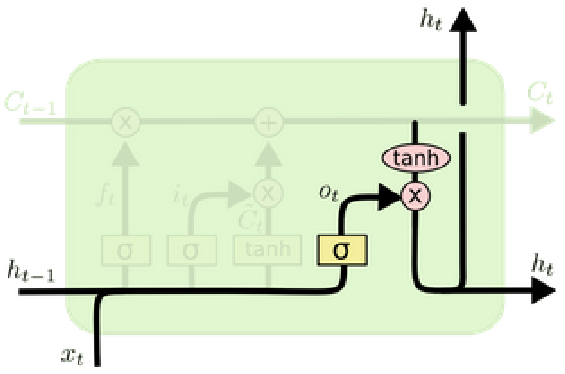
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



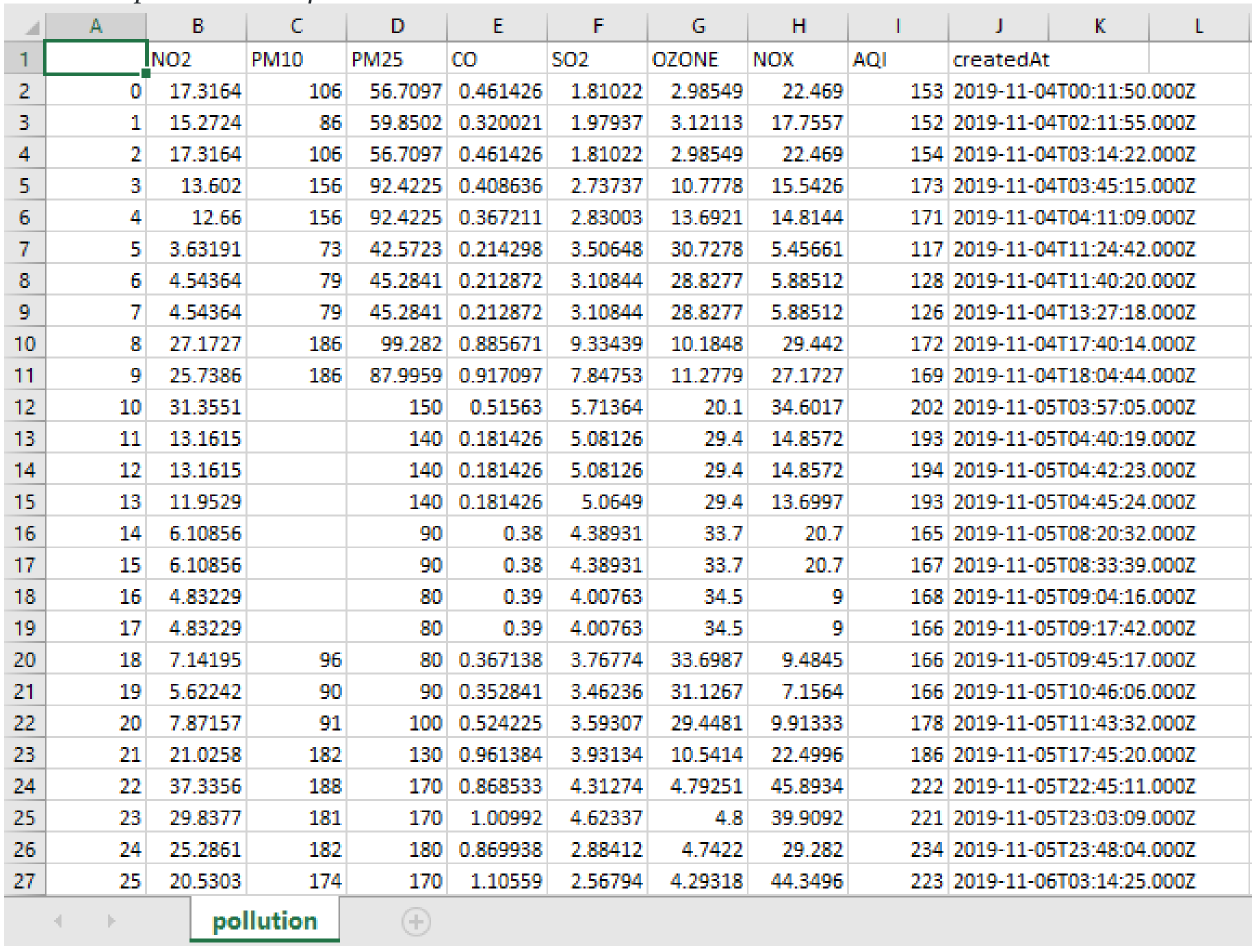
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

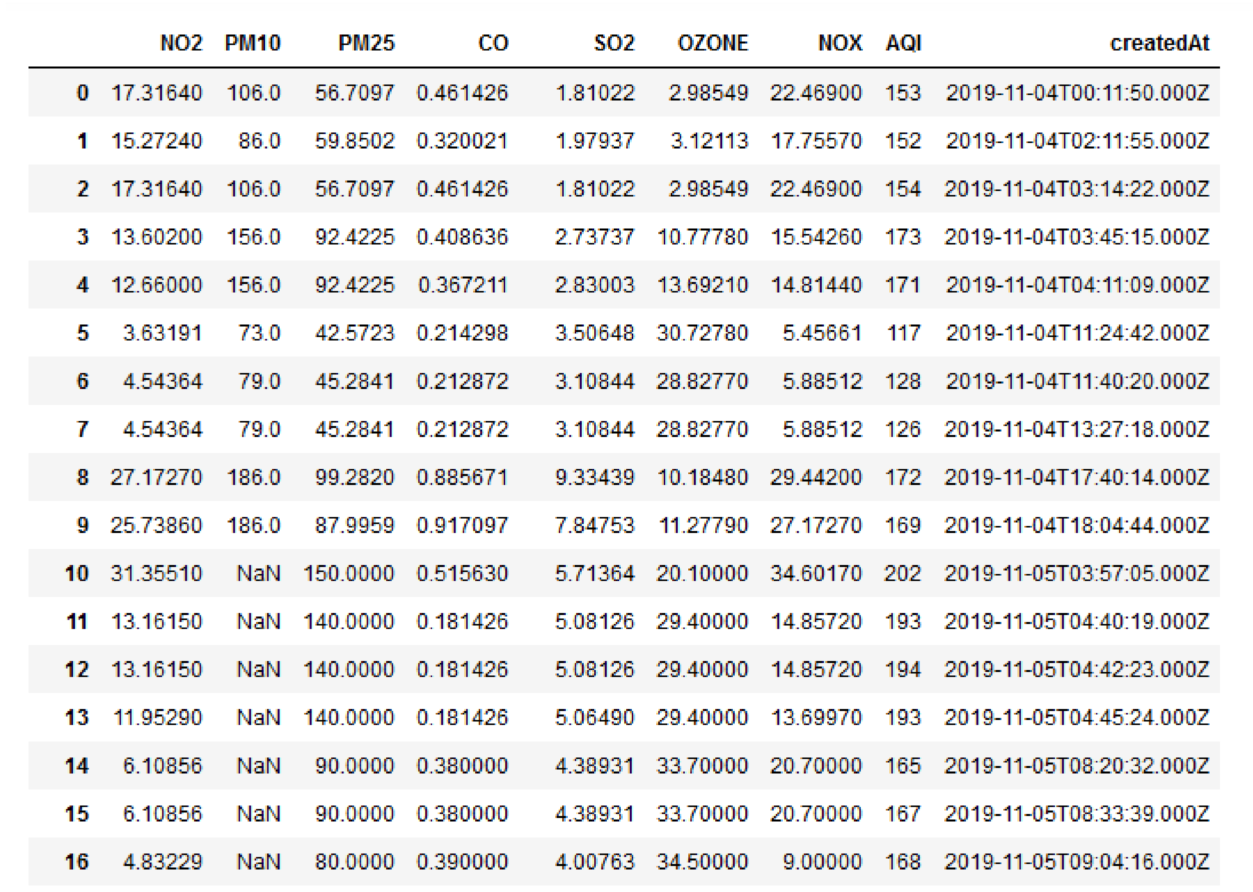
A snippet of the resultant prepared data is shown below:

Stored on pollution.csv file



	A	B	C	D	E	F	G	H	I	J	K	L
1		NO2	PM10	PM25	CO	SO2	OZONE	NOX	AQI	createdAt		
2	0	17.3164	106	56.7097	0.461426	1.81022	2.98549	22.469	153	2019-11-04T00:11:50.000Z		
3	1	15.2724	86	59.8502	0.320021	1.97937	3.12113	17.7557	152	2019-11-04T02:11:55.000Z		
4	2	17.3164	106	56.7097	0.461426	1.81022	2.98549	22.469	154	2019-11-04T03:14:22.000Z		
5	3	13.602	156	92.4225	0.408636	2.73737	10.7778	15.5426	173	2019-11-04T03:45:15.000Z		
6	4	12.66	156	92.4225	0.367211	2.83003	13.6921	14.8144	171	2019-11-04T04:11:09.000Z		
7	5	3.63191	73	42.5723	0.214298	3.50648	30.7278	5.45661	117	2019-11-04T11:24:42.000Z		
8	6	4.54364	79	45.2841	0.212872	3.10844	28.8277	5.88512	128	2019-11-04T11:40:20.000Z		
9	7	4.54364	79	45.2841	0.212872	3.10844	28.8277	5.88512	126	2019-11-04T13:27:18.000Z		
10	8	27.1727	186	99.282	0.885671	9.33439	10.1848	29.442	172	2019-11-04T17:40:14.000Z		
11	9	25.7386	186	87.9959	0.917097	7.84753	11.2779	27.1727	169	2019-11-04T18:04:44.000Z		
12	10	31.3551		150	0.51563	5.71364	20.1	34.6017	202	2019-11-05T03:57:05.000Z		
13	11	13.1615		140	0.181426	5.08126	29.4	14.8572	193	2019-11-05T04:40:19.000Z		
14	12	13.1615		140	0.181426	5.08126	29.4	14.8572	194	2019-11-05T04:42:23.000Z		
15	13	11.9529		140	0.181426	5.0649	29.4	13.6997	193	2019-11-05T04:45:24.000Z		
16	14	6.10856		90	0.38	4.38931	33.7	20.7	165	2019-11-05T08:20:32.000Z		
17	15	6.10856		90	0.38	4.38931	33.7	20.7	167	2019-11-05T08:33:39.000Z		
18	16	4.83229		80	0.39	4.00763	34.5	9	168	2019-11-05T09:04:16.000Z		
19	17	4.83229		80	0.39	4.00763	34.5	9	166	2019-11-05T09:17:42.000Z		
20	18	7.14195	96	80	0.367138	3.76774	33.6987	9.4845	166	2019-11-05T09:45:17.000Z		
21	19	5.62242	90	90	0.352841	3.46236	31.1267	7.1564	166	2019-11-05T10:46:06.000Z		
22	20	7.87157	91	100	0.524225	3.59307	29.4481	9.91333	178	2019-11-05T11:43:32.000Z		
23	21	21.0258	182	130	0.961384	3.93134	10.5414	22.4996	186	2019-11-05T17:45:20.000Z		
24	22	37.3356	188	170	0.868533	4.31274	4.79251	45.8934	222	2019-11-05T22:45:11.000Z		
25	23	29.8377	181	170	1.00992	4.62337	4.8	39.9092	221	2019-11-05T23:03:09.000Z		
26	24	25.2861	182	180	0.869938	2.88412	4.7422	29.282	234	2019-11-05T23:48:04.000Z		
27	25	20.5303	174	170	1.10559	2.56794	4.29318	44.3496	223	2019-11-06T03:14:25.000Z		

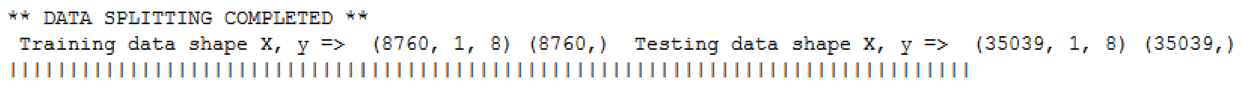
- Normalized data
- Transformed dataset into supervised learning problem
  We will frame the supervised learning problem as predicting the pollution at the current hour (t) given the pollution measurement and weather conditions at the prior time step.



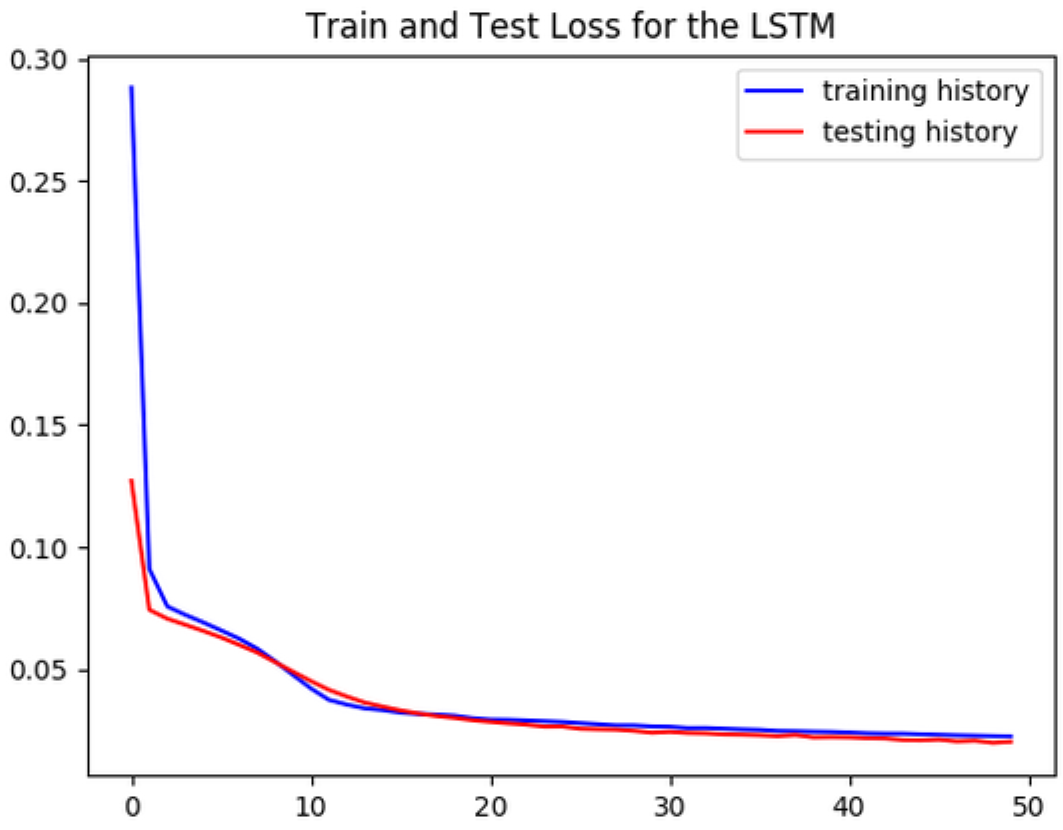
	NO2	PM10	PM25	CO	SO2	OZONE	NOX	AQI	createdAt
0	17.31640	106.0	56.7097	0.461426	1.81022	2.98549	22.46900	153	2019-11-04T00:11:50.000Z
1	15.27240	86.0	59.8502	0.320021	1.97937	3.12113	17.75570	152	2019-11-04T02:11:55.000Z
2	17.31640	106.0	56.7097	0.461426	1.81022	2.98549	22.46900	154	2019-11-04T03:14:22.000Z
3	13.60200	156.0	92.4225	0.408636	2.73737	10.77780	15.54260	173	2019-11-04T03:45:15.000Z
4	12.66000	156.0	92.4225	0.367211	2.83003	13.69210	14.81440	171	2019-11-04T04:11:09.000Z
5	3.63191	73.0	42.5723	0.214298	3.50648	30.72780	5.45661	117	2019-11-04T11:24:42.000Z
6	4.54364	79.0	45.2841	0.212872	3.10844	28.82770	5.88512	128	2019-11-04T11:40:20.000Z
7	4.54364	79.0	45.2841	0.212872	3.10844	28.82770	5.88512	126	2019-11-04T13:27:18.000Z
8	27.17270	186.0	99.2820	0.885671	9.33439	10.18480	29.44200	172	2019-11-04T17:40:14.000Z
9	25.73860	186.0	87.9959	0.917097	7.84753	11.27790	27.17270	169	2019-11-04T18:04:44.000Z
10	31.35510	NaN	150.0000	0.515630	5.71364	20.10000	34.60170	202	2019-11-05T03:57:05.000Z
11	13.16150	NaN	140.0000	0.181426	5.08126	29.40000	14.85720	193	2019-11-05T04:40:19.000Z
12	13.16150	NaN	140.0000	0.181426	5.08126	29.40000	14.85720	194	2019-11-05T04:42:23.000Z
13	11.95290	NaN	140.0000	0.181426	5.06490	29.40000	13.69970	193	2019-11-05T04:45:24.000Z
14	6.10856	NaN	90.0000	0.380000	4.38931	33.70000	20.70000	165	2019-11-05T08:20:32.000Z
15	6.10856	NaN	90.0000	0.380000	4.38931	33.70000	20.70000	167	2019-11-05T08:33:39.000Z
16	4.83229	NaN	80.0000	0.390000	4.00763	34.50000	9.00000	168	2019-11-05T09:04:16.000Z

Model Fitting

- Split data into train and test
- Split into i/p and o/p
- Reshape into 3D
- Define a 50 neuron followed by 1 neuron LSTM
- Add dropout at 30%
- Plot history of training and testing loss



```
** DATA SPLITTING COMPLETED **
 Training data shape X, y =>  (8760, 1, 8) (8760,)  Testing data shape X, y =>  (35039, 1, 8) (35039,)
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```



Training and Testing Loss **vs** *Sample Size*

Evaluate model

The plots show that the accuracy is reliable and hence the deep learning architecture LSTM is a reliable model to predict the time series data of a particular location. The loss also minimizes as the sample size increases thereby reducing bias in the data.

PART II: *Detecting and Predicting Air Pollution Levels of Beijing, China*

The objective here is to do exploratory data analysis and visualization of Air Quality Index based on given features of concentration of PM2.5 concentration, PM10 concentration, sulphur dioxide,nitrogen dioxide, respirable suspended particualte matter, suspended particulate matter, temperature, pressure, dew point, rain and classify the Air Quality as good, moderate, poor, unhealthy, healthy.

Tools Used:

TensorFlow (Backend), Keras, Pandas, NumPy, Scikit-learn Libraries on Jupyter Notebook

Steps followed:

1. Data preparation

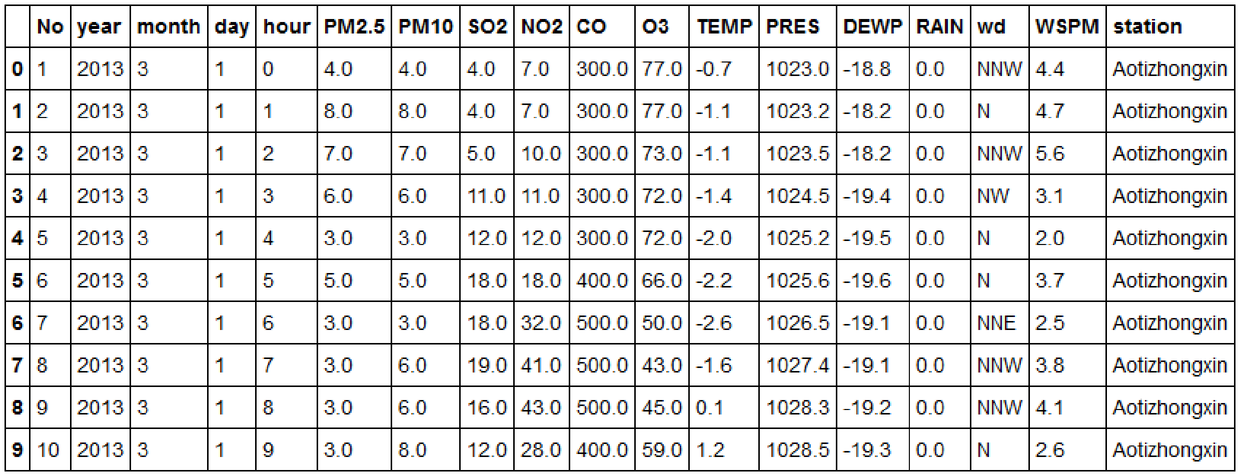
   Data has been collected for the years 2013-17 across several stations in Beijing, namely Aotizhongxin, Changping, Dongsi, Guanyuan, Dingling, Guacheng, Huairou, Nongzhanguan, Shunyi, Tiantan, Wanliu, and Wanshouxigong. The raw data collected has the following variables:
   - year: year in which the sample was collected
   - month: month in which the sample was collected
   - day: day in which the sample was collected
   - hour: hour in which the sample was collected
   - PM2.5: PM2.5 concentration (ug/m^3)
   - PM10: PM10 concentration (ug/m^3)
   - SO2: SO2 concentration (ug/m^3)
   - NO2: NO2 concentration (ug/m^3)
   - CO: CO concentration (ug/m^3)
   - O3: O3 concentration (ug/m^3)
   - TEMP: temperature (degree Celsius)
   - PRES: pressure (hPa)
   - DEWP: dew point temperature (degree Celsius)
   - RAIN: precipitation (mm)
   - wd: wind direction
   - WSPM: wind speed (m/s)
   - station: name of the air-quality monitoring site

     Sampling Rate: 1 sample per hour

2. Exploratory Data Analysis

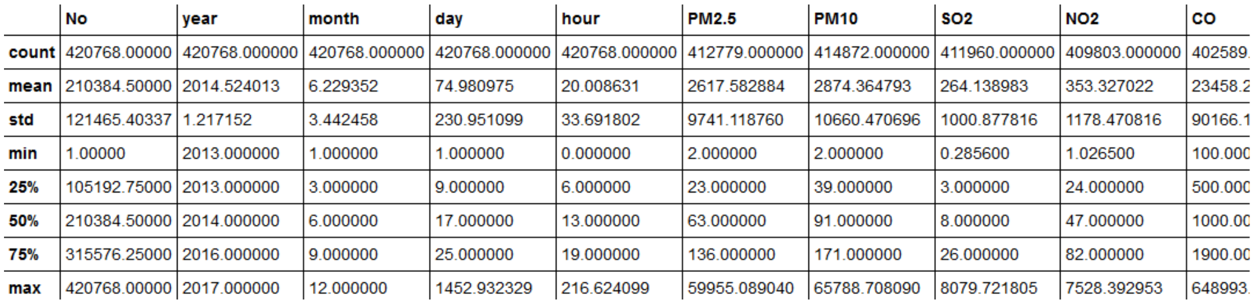
   Here, we visualize, analyze and infer the combined data of all the stations.

Dataset:



	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	wd	WSPM	station
0	1	2013	3	1	0	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0	-18.8	0.0	NNW	4.4	Aotizhongxin
1	2	2013	3	1	1	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2	-18.2	0.0	N	4.7	Aotizhongxin
2	3	2013	3	1	2	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5	-18.2	0.0	NNW	5.6	Aotizhongxin
3	4	2013	3	1	3	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5	-19.4	0.0	NW	3.1	Aotizhongxin
4	5	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	N	2.0	Aotizhongxin
5	6	2013	3	1	5	5.0	5.0	18.0	18.0	400.0	66.0	-2.2	1025.6	-19.6	0.0	N	3.7	Aotizhongxin
6	7	2013	3	1	6	3.0	3.0	18.0	32.0	500.0	50.0	-2.6	1026.5	-19.1	0.0	NNE	2.5	Aotizhongxin
7	8	2013	3	1	7	3.0	6.0	19.0	41.0	500.0	43.0	-1.6	1027.4	-19.1	0.0	NNW	3.8	Aotizhongxin
8	9	2013	3	1	8	3.0	6.0	16.0	43.0	500.0	45.0	0.1	1028.3	-19.2	0.0	NNW	4.1	Aotizhongxin
9	10	2013	3	1	9	3.0	8.0	12.0	28.0	400.0	59.0	1.2	1028.5	-19.3	0.0	N	2.6	Aotizhongxin

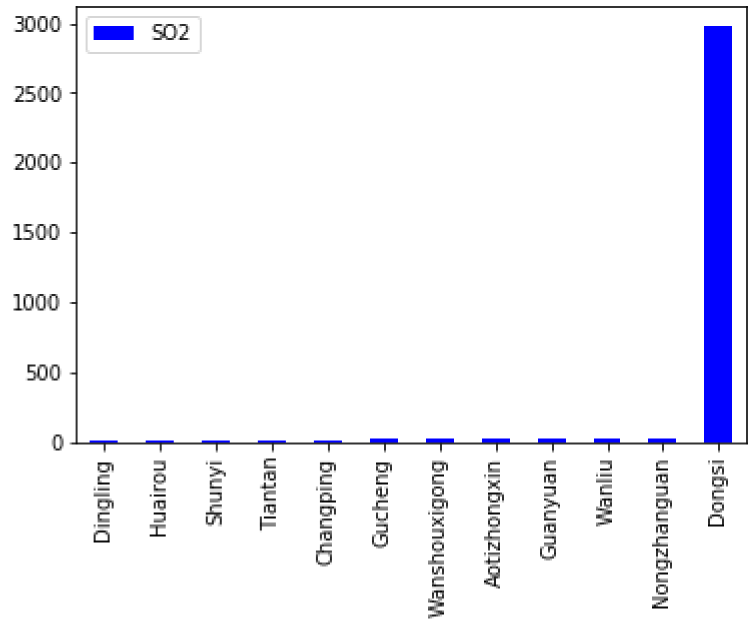
Basic statistical information on the dataset:

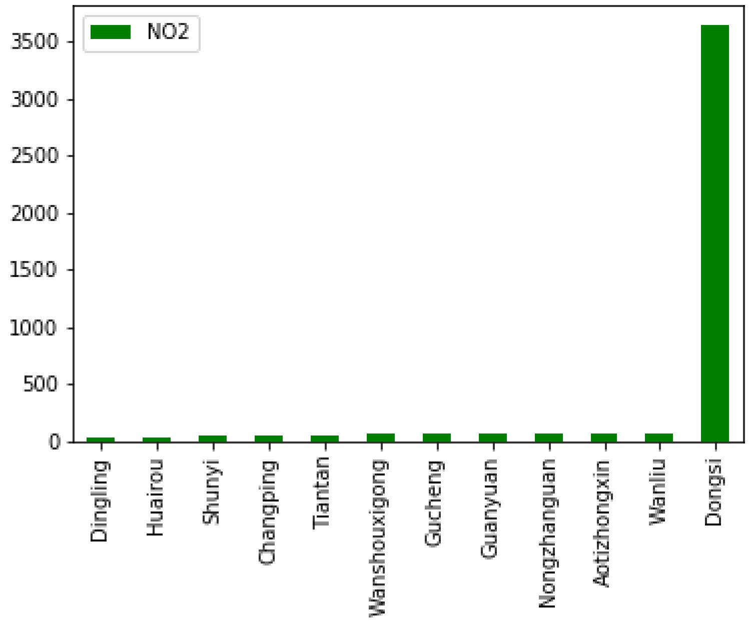


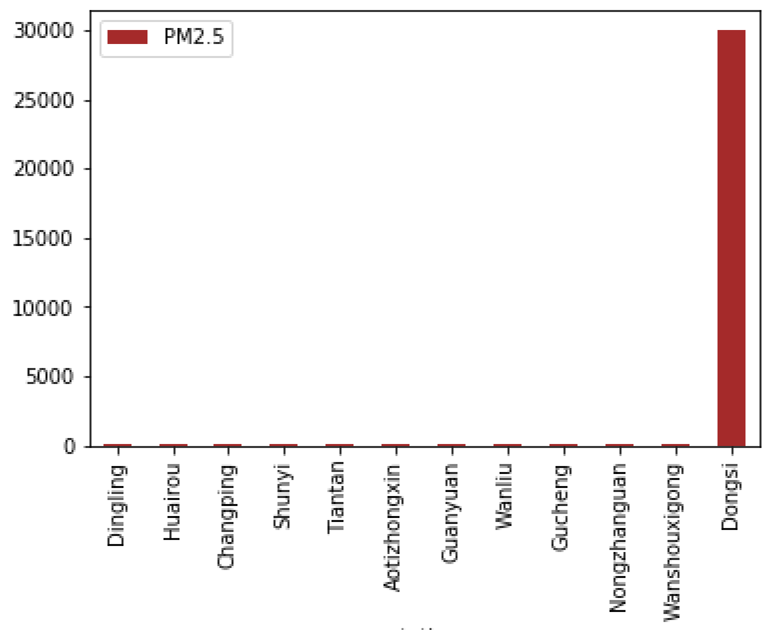
	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO
count	420768.00000	420768.000000	420768.000000	420768.000000	420768.000000	412779.000000	414872.000000	411960.000000	409803.000000	402589.
mean	210384.50000	2014.524013	6.229352	74.980975	20.008631	2617.582884	2874.364793	264.138983	353.327022	23458.2
std	121465.40337	1.217152	3.442458	230.951099	33.691802	9741.118760	10660.470696	1000.877816	1178.470816	90166.1
min	1.00000	2013.000000	1.000000	1.000000	0.000000	2.000000	2.000000	0.285600	1.026500	100.000
25%	105192.75000	2013.000000	3.000000	9.000000	6.000000	23.000000	39.000000	3.000000	24.000000	500.000
50%	210384.50000	2014.000000	6.000000	17.000000	13.000000	63.000000	91.000000	8.000000	47.000000	1000.00
75%	315576.25000	2016.000000	9.000000	25.000000	19.000000	136.000000	171.000000	26.000000	82.000000	1900.00
max	420768.00000	2017.000000	12.000000	1452.932329	216.624099	59955.089040	65788.708090	8079.721805	7528.392953	648993.

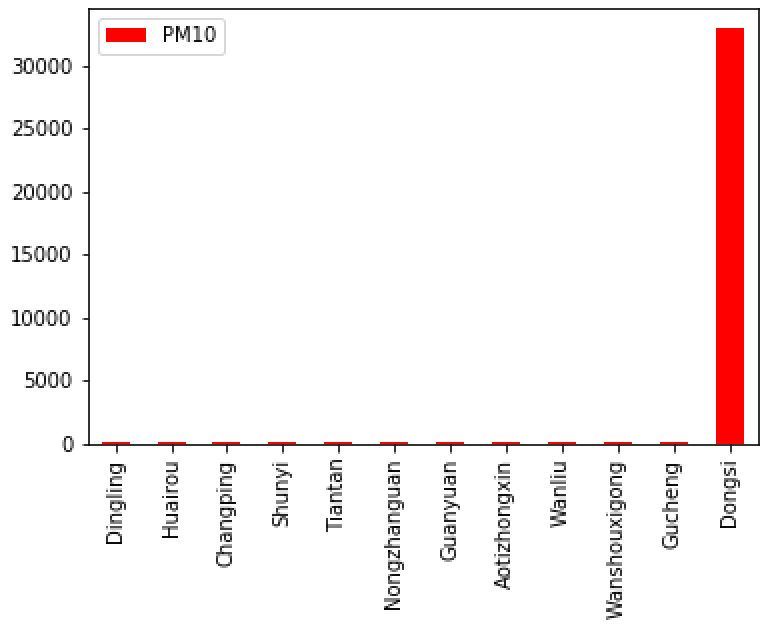
Data Dimensions:
420768 rows x 18 columns

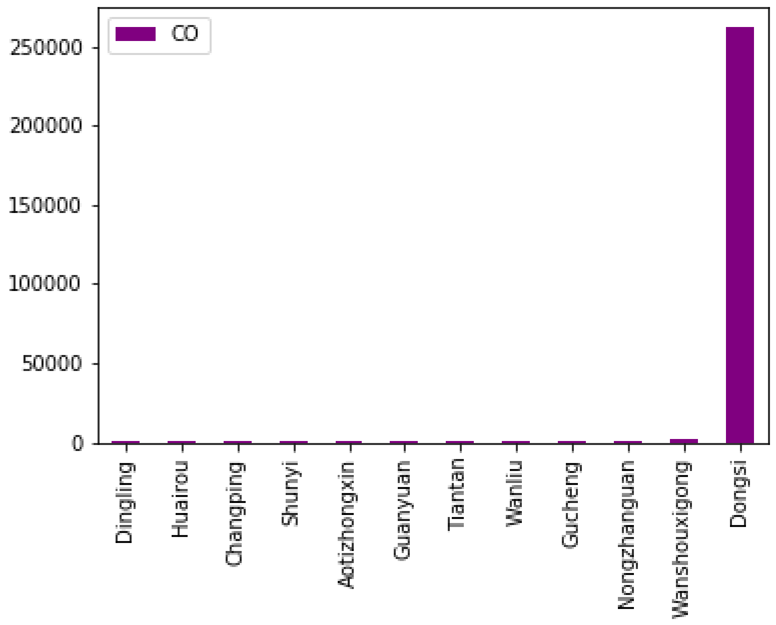
Visualization of Stations with Highest Pollutants:

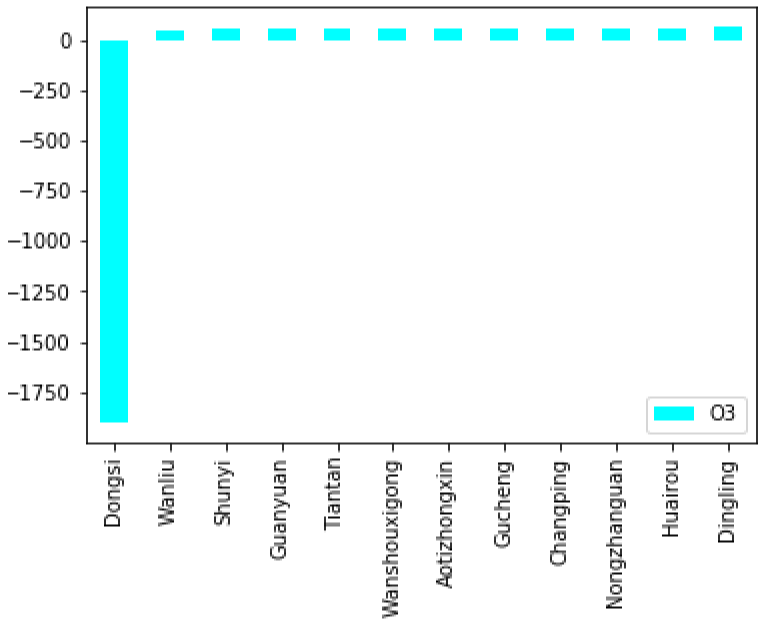





PM2.5
30000
25000
20000
15000
10000
5000
0
Dingling
Huairou
Changping
Shunyi
Tiantan
Aotizhongxin
Guanyuan
Wanliu
Gucheng
Nongzhanguan
Wanshouxigong
Dongsi

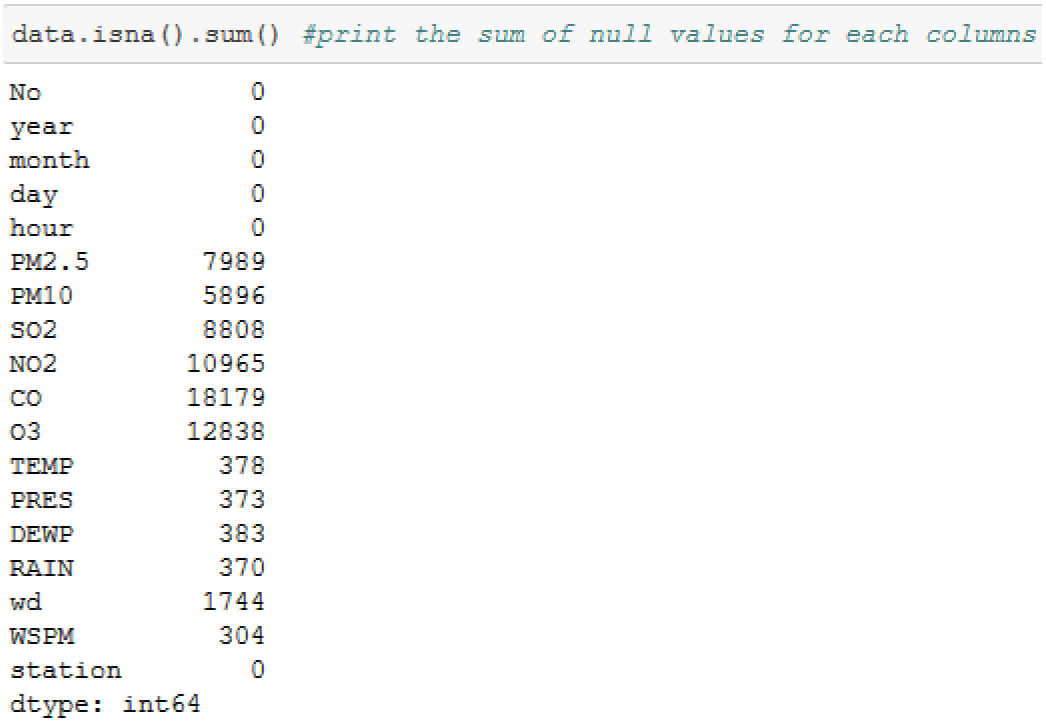

PM10
30000
25000
20000
15000
10000
5000
0
Dingling
Huairou
Changping
Shunyi
Tiantan
Nongzhanguan
Guanyuan
Aotizhongxin
Wanliu
Wanshouxigong
Gucheng
Dongsi





From these plots we can clearly infer that Dongsi is the most polluted region in Beijing.

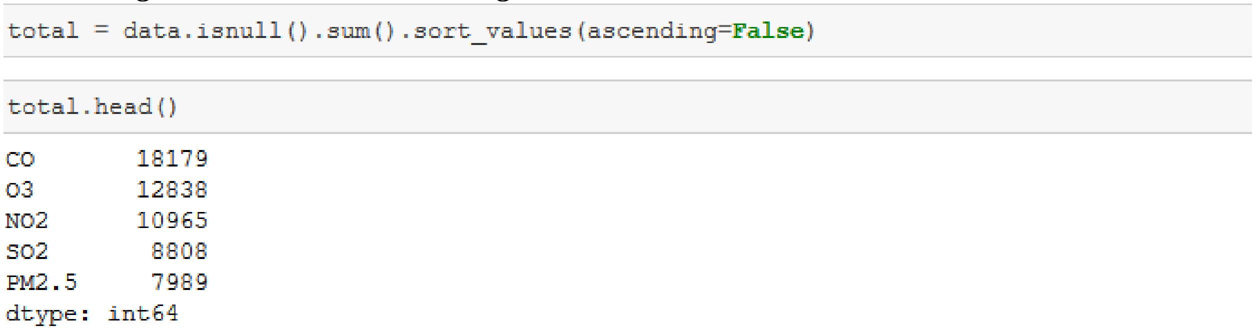
Null Values Count for each parameter:



```
data.isna().sum() #print the sum of null values for each columns
```

```
No              0
year            0
month           0
day             0
hour            0
PM2.5        7989
PM10         5896
SO2          8808
NO2         10965
CO          18179
O3          12838
TEMP          378
PRES          373
DEWP          383
RAIN          370
wd           1744
WSPM          304
station         0
dtype: int64
```

Calculating the total number of missing values:



```
total = data.isnull().sum().sort_values(ascending=False)
```

```
total.head()
```

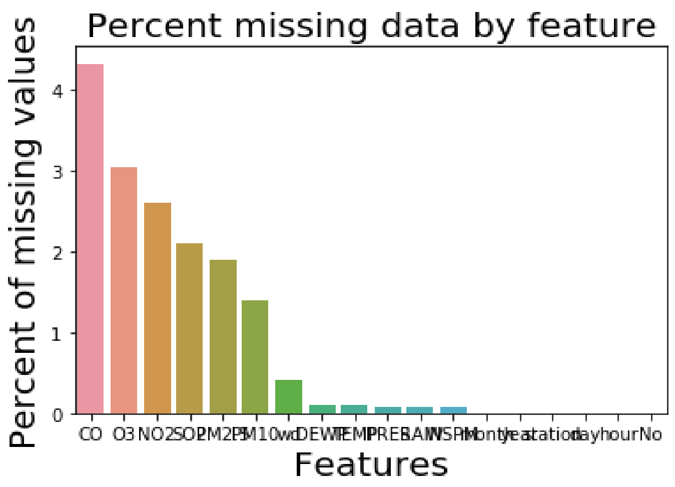
```
CO       18179
O3       12838
NO2      10965
SO2       8808
PM2.5     7989
dtype: int64
```

We then calculate the percent of null values for each columns (sum of null values / total non-null value) *100

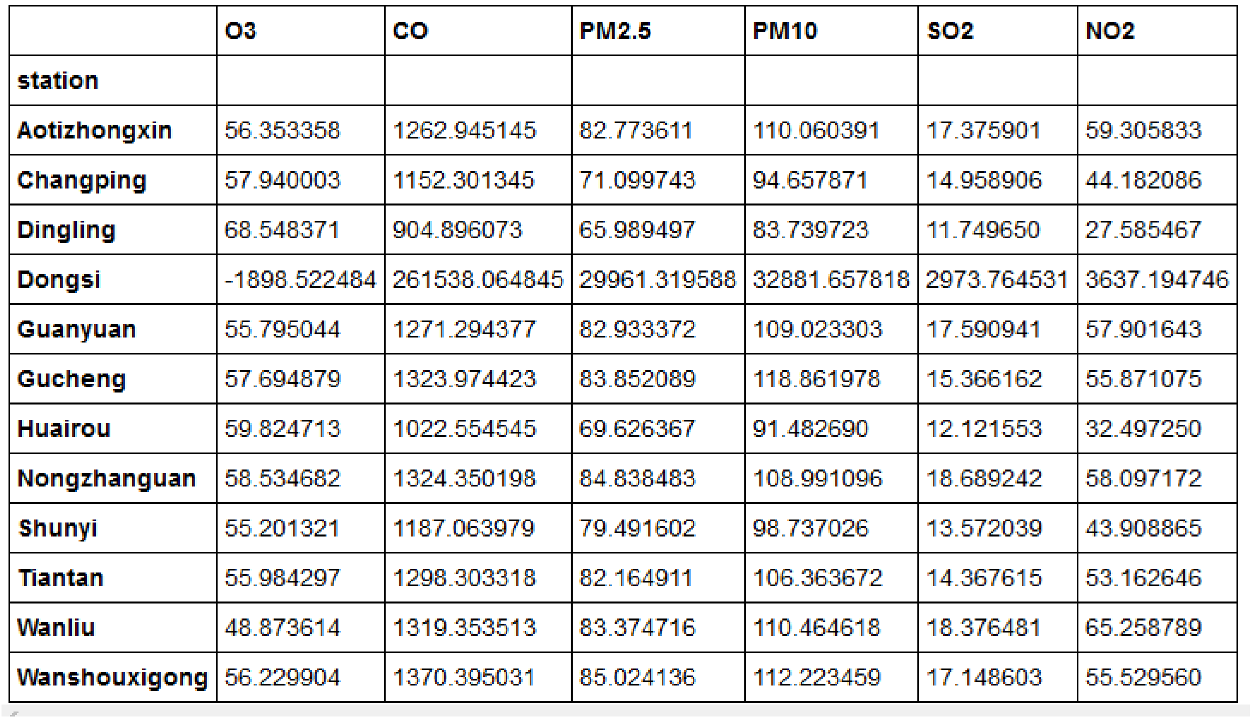
Percentage of Missing Values (Bar Plot):

```
sns.barplot(x=missing_data.index, y=missing_data['Percent'])
plt.xlabel('Features', fontsize=20)
plt.ylabel('Percent of missing values', fontsize=20)
plt.title('Percent missing data by feature', fontsize=20)
```

```
Text(0.5, 1.0, 'Percent missing data by feature')
```



Mean Distribution by Station:



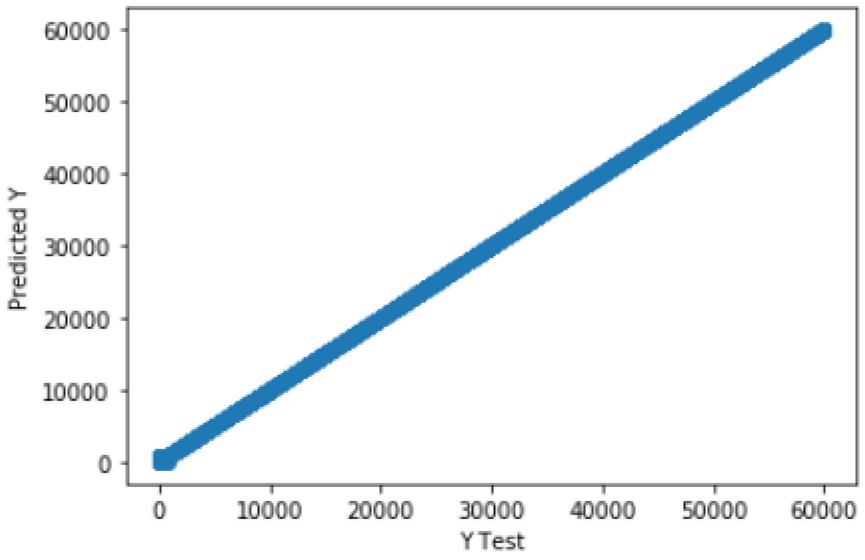
	O3	CO	PM2.5	PM10	SO2	NO2
station						
Aotizhongxin	56.353358	1262.945145	82.773611	110.060391	17.375901	59.305833
Changping	57.940003	1152.301345	71.099743	94.657871	14.958906	44.182086
Dingling	68.548371	904.896073	65.989497	83.739723	11.749650	27.585467
Dongsi	-1898.522484	261538.064845	29961.319588	32881.657818	2973.764531	3637.194746
Guanyuan	55.795044	1271.294377	82.933372	109.023303	17.590941	57.901643
Gucheng	57.694879	1323.974423	83.852089	118.861978	15.366162	55.871075
Huairou	59.824713	1022.554545	69.626367	91.482690	12.121553	32.497250
Nongzhanguan	58.534682	1324.350198	84.838483	108.991096	18.689242	58.097172
Shunyi	55.201321	1187.063979	79.491602	98.737026	13.572039	43.908865
Tiantan	55.984297	1298.303318	82.164911	106.363672	14.367615	53.162646
Wanliu	48.873614	1319.353513	83.374716	110.464618	18.376481	65.258789
Wanshouxigong	56.229904	1370.395031	85.024136	112.223459	17.148603	55.529560

For convenience of computation, the missing values have been filled by the mean which is grouped by station.

3. Linear Regression Modelling and Prediction

We use SO2, NO2, O3, PM10 variables (X) to predict PM2.5 concentration (Y). The model has been trained for 80% of the dataset selected randomly and the tested on the remaining 20% of the data.

The plot obtained was:

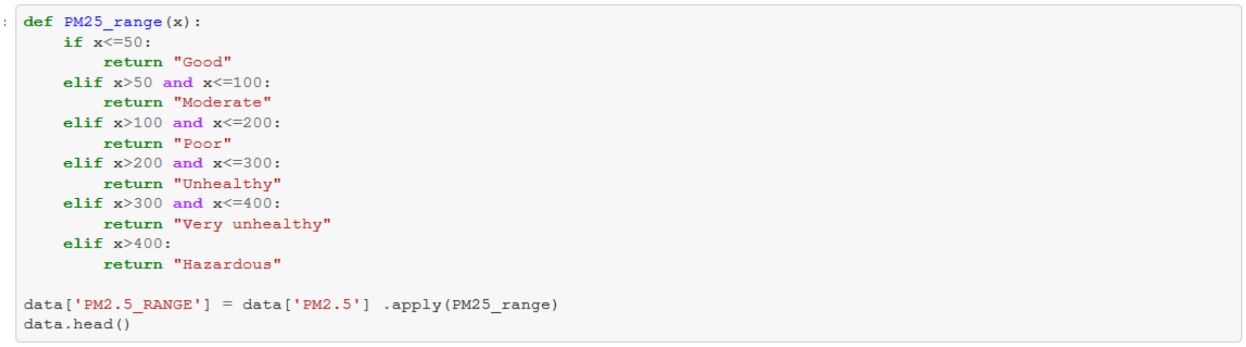


where X and Y have been defined above.

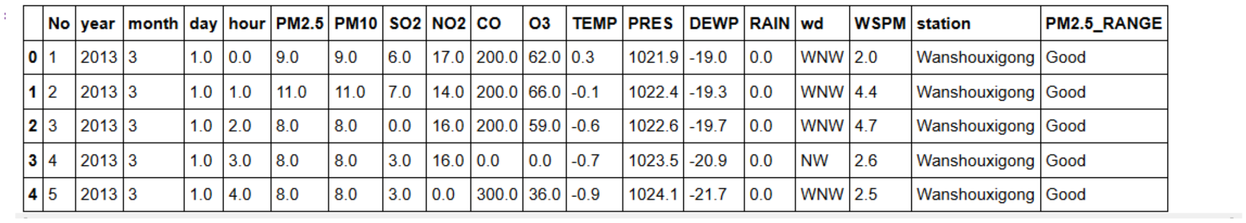
For evaluation, the Root Square Error and MSE come out to be 1 and 38.02 respectively.

4. Categorization and Classification of PM2.5 concentration

We assign labels to designation PM2.5 concentration levels indicating if it is good, poor or moderate. PM2.5 is an important component in determining air pollution levels.



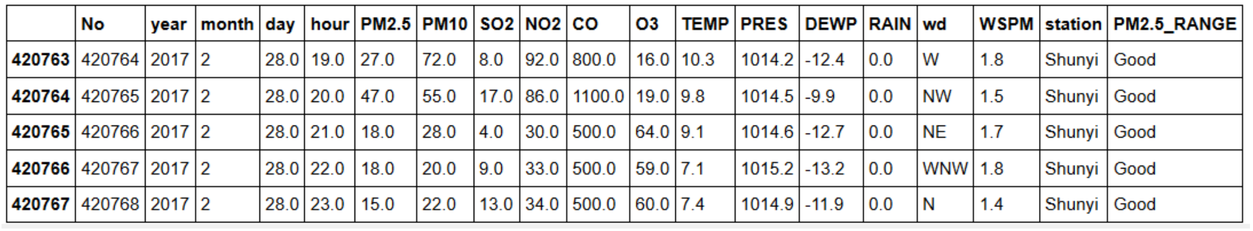
```
def PM25_range(x):
    if x<=50:
        return "Good"
    elif x>50 and x<=100:
        return "Moderate"
    elif x>100 and x<=200:
        return "Poor"
    elif x>200 and x<=300:
        return "Unhealthy"
    elif x>300 and x<=400:
        return "Very unhealthy"
    elif x>400:
        return "Hazardous"

data['PM2.5_RANGE'] = data['PM2.5'] .apply(PM25_range)
data.head()
```



	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	wd	WSPM	station	PM2.5_RANGE
0	1	2013	3	1.0	0.0	9.0	9.0	6.0	17.0	200.0	62.0	0.3	1021.9	-19.0	0.0	WNW	2.0	Wanshouxigong	Good
1	2	2013	3	1.0	1.0	11.0	11.0	7.0	14.0	200.0	66.0	-0.1	1022.4	-19.3	0.0	WNW	4.4	Wanshouxigong	Good
2	3	2013	3	1.0	2.0	8.0	8.0	0.0	16.0	200.0	59.0	-0.6	1022.6	-19.7	0.0	WNW	4.7	Wanshouxigong	Good
3	4	2013	3	1.0	3.0	8.0	8.0	3.0	16.0	0.0	0.0	-0.7	1023.5	-20.9	0.0	NW	2.6	Wanshouxigong	Good
4	5	2013	3	1.0	4.0	8.0	8.0	3.0	0.0	300.0	36.0	-0.9	1024.1	-21.7	0.0	WNW	2.5	Wanshouxigong	Good

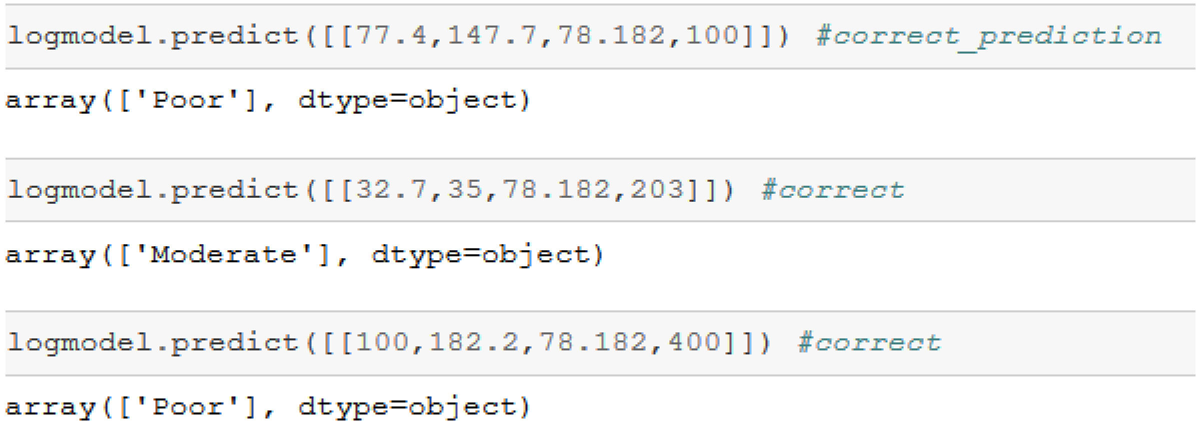
5. Logistic Regression for Classification

The sample data is assigned labels for training and comparision after preprocessing:



	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	wd	WSPM	station	PM2.5_RANGE
420763	420764	2017	2	28.0	19.0	27.0	72.0	8.0	92.0	800.0	16.0	10.3	1014.2	-12.4	0.0	W	1.8	Shunyi	Good
420764	420765	2017	2	28.0	20.0	47.0	55.0	17.0	86.0	1100.0	19.0	9.8	1014.5	-9.9	0.0	NW	1.5	Shunyi	Good
420765	420766	2017	2	28.0	21.0	18.0	28.0	4.0	30.0	500.0	64.0	9.1	1014.6	-12.7	0.0	NE	1.7	Shunyi	Good
420766	420767	2017	2	28.0	22.0	18.0	20.0	9.0	33.0	500.0	59.0	7.1	1015.2	-13.2	0.0	WNW	1.8	Shunyi	Good
420767	420768	2017	2	28.0	23.0	15.0	22.0	13.0	34.0	500.0	60.0	7.4	1014.9	-11.9	0.0	N	1.4	Shunyi	Good

We use the 80-20 split to train and test the model.
After the model is trained, it is tested for arbitrary data points for validation:



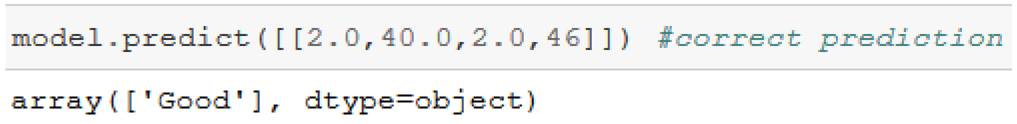
```
logmodel.predict([[77.4,147.7,78.182,100]]) #correct_prediction

array(['Poor'], dtype=object)

logmodel.predict([[32.7,35,78.182,203]]) #correct

array(['Moderate'], dtype=object)

logmodel.predict([[100,182.2,78.182,400]]) #correct

array(['Poor'], dtype=object)
```

The accuracy comes out to be 63.30%.

6. Random Forest Classification

The sample data is assigned labels for training and comparision after preprocessing. We then use the 80-20 split to train and test the model.

After the model is trained, it is tested at an arbitrary data points for validation:



```
model.predict([[2.0,40.0,2.0,46]]) #correct prediction

array(['Good'], dtype=object)
```

The accuracy comes out to be 76.30% which is much better than the logistic classifier.

7. Decision Tree Classification

The sample data is assigned labels for training and comparision after preprocessing. We then use the 80-20 split to train and test the model.

After the model is trained, it is tested at an arbitrary data points for validation:

```
model2.predict([[2.0,40.0,2.0,46]]) # correct prediction
```

```
array(['Good'], dtype=object)
```

The accuracy comes out to be 71.68% which is much better than the logistic classifier but still falls behind random forest classifier.

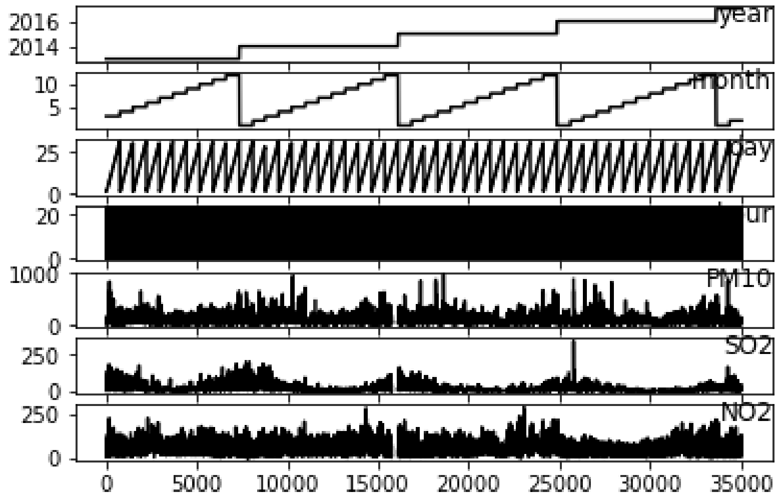
8. **LSTM Modelling and Prediction:**

The following were done for preprocessing:

- Replaced all NA values with mean
- Parsed date-time into pandas dataframe index
- Specified clear names for each columns

Data visualization

- Used matplotlib to plot

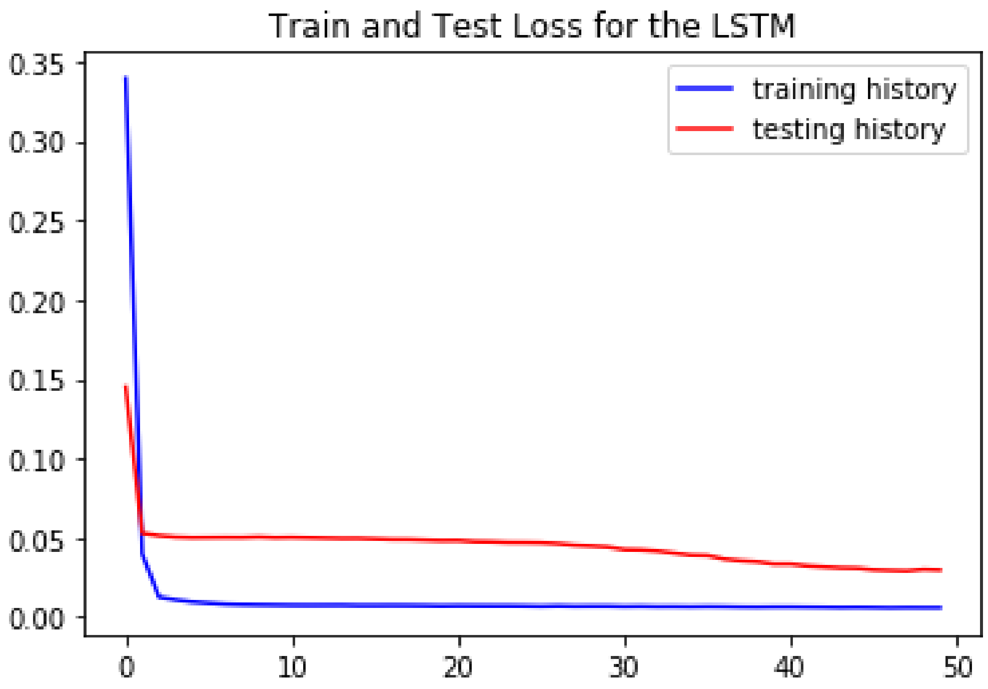


Corresponding to parameters obtained from various stations across Beijing

We do the following for Model Fitting and Visualization:

- Split data into train and test (80-20 split)
- Train the model for 50 epochs
- Split into i/p and o/p
- Reshape into 3D

- Define a 50 neuron followed by 1 neuron LSTM
- Add dropout at 30%
- Plot history of training and testing loss



Training and Testing Loss **vs** *Sample Size*

Evaluate model

The plots show that the accuracy is reliable and hence the deep learning architecture LSTM is a reliable model to predict the time series data of a particular location. The loss also minimizes as the sample size increases thereby reducing bias in the data.

Here, we can clearly visualize how LSTM is an extremely better alternative to predict time series models as compared to Regression, Decision Tree, and Random Forest Approaches. This sets LSTM architecture much better than conventional RNN and various machine learning algorithms.

FUTURE SCOPE

The model can make accurate hourly predictions on the data for PIN code 500078 from the data obtained by the sensors around BITS Pilani Hyderabad Campus. With this verification, this LSTM model can be expanded to analyze and predict the air pollution levels of the entire city of Hyderabad.

The various models worked on Beijing’s data further enhance the superiority of deep learning architectures. The comparative analysis provided herewith can be used for academic research on why LSTM architecture works better than most Machine Learning algorithms.

The accuracy can be significantly improved with special modifications to the LSTM architecture. For example, in the paper on *Air pollutant severity prediction using Bi-directional LSTM Network* by Ishan Verma, Rahul Ahuja, Hardik Meisheri, Lipika Dey; the authors provide an introduction to the use of Bi-directional LSTMs to improve the accuracy further owing to its feature to use both past and future influences to make accurate predictions. Further, a comparative analysis can be established between the GRU (Gated Recurrent Unit) which uses two gates in its architecture and the LSTM which uses three gates in its architecture.

Additionally, the LSTM model can be aided with the satellite aided clustering and classification of polluted zones to identify the sources of pollution. The satellite data analysis gives faster and clearer visualizations of a zone but the analysis is restricted by the coarser temporal resolution. Here, our deep learning model can provide the required finer temporal resolution. Both the models can be integrated to provide a tool to determine the air pollution level of a given zone and pinpoint the source(s) of pollutants quickly and accurately. The model can be incorporated with Reinforcement Learning approaches to further improve the accuracy of predictions.

CONCLUSION

The comparative analysis with vivid visualizations provide a clearer picture as to which model gives us the best accuracy. The ensembled model, along with the satellite data analysis can provide a tool to quickly as well as accurately determine the pollution threat level and the causes of pollution of various metropolitan cities. Such a tool can assist various municipal corporations governing these metropolitan cities to counter the threat posed by the growing air pollution levels across various cosmopolitan cities.

Moreover, the comparative analysis on the huge Beijing data clearly shows the need of deep learning architectures such as LSTM in time-series applications. LSTM closely models the future prediction based on the previous epochs in our dataset whereas most Machine Learning algorithms rely on a general trend in the data values to predict the future values. This will pave forward the way for deeper and more complex architectures in the future.

BIBLIOGRAPHY

[1] Ishan Verma, Rahul Ahuja, Hardik Meisheri, Lipika Dey, “Air pollutant severity prediction using Bi-directional LSTM Network”, *IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2018.*

[2] Yi-Ting Tsai, Yu-Ren,Zeng, Yue-Shan Chang, “Air pollution forecasting using RNN with LSTM”, *IEEE 16th Int. Conf. on Dependable, Autonomic & Secure Comp., 16th Int. Conf. on Pervasive Intelligence & Comp., 4th Int. Conf. on Big Data Intelligence & Comp., and 3rd Cyber Sci. & Tech. Cong., 2018.*

[3] Shaheen Alhirmizy, Banaz Qader, “Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution”, *IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2018.*

[4] Yang Han, Jacqueline C.K. Lam, Victor O.K. Li, “A Bayesian LSTM Model to Evaluate the Effects of Air Pollution Control Regulations in China”, *IEEE International Conference on Big Data (Big Data), 2018.*

[5] Wei Song, Jinkun Han, Junwei Xie, Yanning Gao, Liangliang Song, “System for Detecting and Forecasting PM2.5 Concentration Levels Using Long Short-Term Memory and LoRa”, *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2019.*

[6] Pratyush Singh, Lakshmi Narasimhan T, Chandra Shekar Lakshminarayanan, “DeepAir: Air Quality Prediction using Deep Neural Network”, *IEEE Region 10 Conference (TENCON 2019), 2019.*