

**SCHOOL OF COMPUTER SCIENCE AND APPLICATIONS**

A Project Report

On

Air Pollution Forecasting using LSTM and GRU

Submitted in partial fulfillment of the requirements for the award of the Degree of

Master of Computer Science

Submitted by

Davada Juned Aslam

R20SCS06

under the guidance of

Internal Guide External Guide

Dr. Hemant K. S.

July 2022

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064

[www.reva.edu.in](http://www.reva.edu.in)



**SCHOOL OF COMPUTER SCIENCE AND APPLICATIONS**

# CERTIFICATE

The project work titled - **Air Pollution Forecasting using LSTM and GRU,** is beingcarried out under our guidance by **Davada Juned Aslam** , **R20SCS06** , a bonafide student of REVA University, and is submitting the project report in partial fulfillment, for the award of **Master of Computer Science** during the academic year **2021–22**. The project report has been approved, as it satisfies the academic requirements with respect to the Project Work prescribed for the aforementioned Degree.

**Signature with Date Signature with Date Signature with Date**

**Hemant K. S.**

**Internal Guide**

**Prof. Vijayalakshmi .K**

**Academic Vertical Head-PG**

**<Name of the External Guide>**

**External Guide**

**Signature with Date**

**­­­­­­­­­­­­­­­­­­­**

**Dr. S. Senthil**

**Professor and Director**

**External Examiner**

**Name of the Examiner with** **Affiliation** **Signature with Date**

1.

2.

**Company Certificate**

**DECLARATION**

I, Mr. Davada Juned Aslam , pursuing my **Master of Computer Science**, offered by School of Computer Science and Applications, REVA University, declare that this Project titled - “Air Pollution Forecasting using LSTM and GRU” , is the result of the Project Work done by me under the supervision of Dr. Hemanth K. S. (Associate Professor) and < External guide with designation, > at < name of the company where project work has been carried out>.

I am submitting this Project Work in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications by REVA University, Bengaluru, during the Academic Year 2021-22.

I further declare that this Project Report or any part of it has not been submitted for the award of any other Degree / Diploma of this University or any other University/ Institution.

*(Signature of the candidate)*

*Signed by me on:*

*Certified that this project work submitted by Davada Juned Aslam has been carried out under our guidance and the declaration made by the candidate is true to the best of my knowledge.*

*Signature of Internal Guide Signature of External Guide*

*Date :……….. Date :………..*

*Signature of Director of the School*

*Date :………..*

*Official Seal of the School*

**ACKNOWLEDGEMENT**

# I hereby acknowledge all those, under whose support and encouragement, I have been able to fulfil all my academic commitments successfully. In this regard, I take this opportunity to express my deep sense of gratitude and sincere thanks to School of Computer Science and Applications which has always been a tremendous source of guidance.

# I express my sincere gratitude to Dr. P. SHYAMA RAJU, Honorable Chancellor, REVA University, and Bengaluru for providing us the state-of-the-art facilities.

I am thankful to **Dr. M. DHANAMJAYA**, Vice Chancellor and **Dr. N.RAMESH**, Registrar, REVA University, Bengaluru for their support and encouragement.

I take this opportunity to express my heartfelt thanks to **Dr. S. SENTHIL**, Professor & Director, School of CSA, REVA University and my sincere thanks to **Prof. VIJAYALAKSHMI.K** Associate Professor, Academic Vertical Head for PG Programmes, School of CSA, REVA University, whose encouragement and best wishes provided impetus for the Project Work carried out.

# Also, my sincere gratitude goes to my internal guide,Dr. Hemant K. S. Associate Professor, for the valuable suggestion and constant encouragement towards the completion of this project. I am deeply indebted to my external guide, (Guide name, designation and organization) for permitting to commence this Project and utilize all the facilities provided by the University.

# 

# Last, but not the least, I thank my parents for their incredible support and encouragement throughout.

**ABSTRACT**

With the development of the industry in the last few years. We are facing an issue related to air quality. We are not getting the proper air quality because of the pollution out there in the environment due to harmful gases from the industry. It will affect human health and it can cause a serious issues related to the lungs. Air pollution can cause by different ways in today's world like the CO2 released from the car and some harmful chemicals which are released in the air by the industry. Air pollution can spread with the flow of air means wind direction and speed. To overcome this real-time problem there is much research going on nowadays that can forecast air pollution but it required a lot of computational power. Our approach in this research is that we are going to forecast air pollution using the very famous deep learning technique Recurrent Neural Networks (RNN) based framework with special structure memory cell known as Long Short term memory (LSTM) and Gated Recurrent Unit (GRU). We can easily forecast air pollution using this easily by just providing some last day’s data to the model. It will forecast the next 24 hours of data by just providing the last 15 days of air pollution of data**.**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **CHAPTERS** | **PAGE NO** |
| **1.** | **INTRODUCTION** | 9 |
| 1.1 INTRODUCTION TO PROJECT |
| - SOFTWARE AND HARDWARE SPECIFICATION |
| 1.2 COMPANY PROFILE |
| **2.** | **LITERATURE SURVEY** | 11 |
| **3.** | **SYSTEM ANALYSIS** | 12 |
| 3.1 EXISTING SYSTEM |
| 3.2 LIMITATIONS OF THE EXISTING SYSTEM |
| 3.3 PROPOSED SYSTEM |
| 3.4 ADVANTAGES OF THE PROPOSED SYSTEM |
| **4.** | **SYSTEM DEVELOPMENT STRATEGY** | 13 |
| 4.1 METHODOLOGY |
| 4.2 IMPLEMENTATION |
| **5.** | **SYSTEM DESIGN** | 21 |
| 5.1 HIGH LEVEL DESIGN (ARCHITECTURAL) |
| **6.** | **CODING** | 22 |
| 6.1 CODE |
| **7**. | **DATASET** | 42 |
| **8.** | **TESTING** | 43 |
| **9.** | **CONCLUSION AND SCOPE FOR FUTURE ENCHANCEMENT** | 45 |
| **10.** | **BIBILOGRAPHY** | 46 |
|  | **APPENDIX** |  |
|  | 1. **SNAPSHOTS** |  |
|  | 1. **PLAGIARISM REPORT** |  |

1. **INTRODUCTION**
   1. **INTRODUCTION TO PROJECT**

Air pollution has been one of the major concerns for the developing countries such as India since the last few years. Air pollution has caused the most number of deaths in the near past and count keep increasing every year. Number shows that more than 660 million Indians breathe polluted air every day. Breathing polluted air can cause many diseases like lung cancer , asthma , heart diseases many more. Air quality Index is measured adopted by the Indian government to quantify air pollution.

According to the WHO (World Health Organization), air pollution is the contamination of the indoor or outdoor environment by any chemical, physical or biological agent that modified the natural characteristics of the atmosphere. Air pollution can be divided into 2 parts indoor pollution from households and outdoor pollution from vehicles and industry. Air pollution can be felt by Household combustion, motor vehicles, industrial facilities, and forest fires are common resources of air pollution. WHO data shows that almost all the global population (90%) breathes air that exceeds WHO Health guidelines . Every 9 out of 10 people lives where air quality exceeds WHO guidelines. The World Health Organization (WHO) reported that air pollution causes 4.2 million premature deaths per year in cities and rural areas around the world. Air pollution in the cities and rural areas causes some dangerous diseases like stroke, heart disease, lung cancer, and acute and chronic respiratory diseases. Around the globe around 2.6 billion, people are exposed to dangerous levels of household air pollution. This is the data from WHO.

Air pollution forecasting techniques are being rapidly advanced and measuring pollution increase. Traditional approaches use some mathematical and statistical techniques. This conventional forecasting model takes a lot of computational power to forecast the data. With recent advancements in technology, we come up with Deep Learning which is very good for solving real-time problems in various domains like computer vision, Natural Language Processing, and many more. With the help of the Deep Learning we can obtain the best results, We can use the Deep Learning for the air pollutions forecasting of the data.

As you can see in the world Air pollution leads the third largest cause of death.

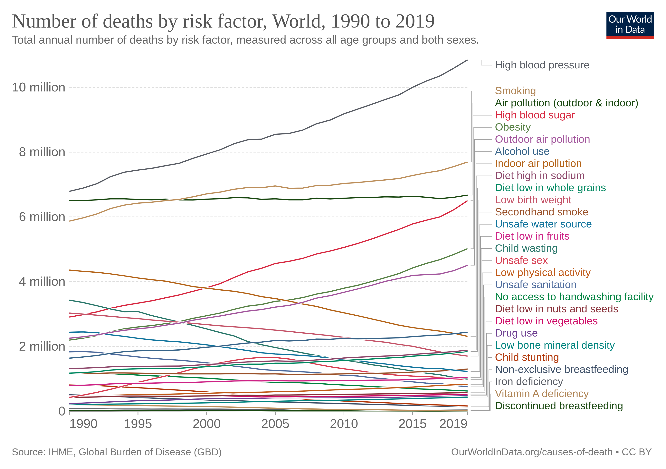


Figure 1 Represent the number of death cause by different diseases in the world

We can solve this problem with the help of emerging technologies like Deep Learning. Which will help us to forecast the time series data by providing the data from the previous time. In Deep Learning we have algorithms like LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit). Which will help us to forecast the data for the air pollution.

* 1. **SOFTWARE AND HARDWARE SPECIFICATION**
* Software Requirement
* Operating System : Windows 7 & above
* Tools : Numpy , Pandas , Tensorflow , Keras
* Programming Language : Python
* Hardware Requirement
* Processor : Intel® Core™ i3 or Ryzen 3 above
* RAM : 4 GB or above

1. **LITERATURE SURVEY**

Deep Learning approaches have become more popular in last few years[1]. The most popular Deep Learning techniques are Multi-Layer Perceptron (MLP), Deep Belief Network (DBN), Convolutional Neural Networks (CNN), Recurrent Neural Network (RNN), and Auto Encoder (AE)[4].We will use Recurrent Neural network for our work to obtain the result.

Many researchers are working on this problem of air pollution forecasting nowadays. Mostly they are focusing on the LSTM (Long short term memory) or GRU (Gated Recurrent Unit). But in this research, we are going to combine both of the famous models of Recurrent Networks[4]. In many fields, these 2 models are giving their best to provide the solution to the problem.

The LSTM model can work more efficient the on the hourly basis concentration of the air pollutions[1]. In this studies they have worked with the real time data of Vishakhapatnam for the 12 hours of air pollutions[1]. Based on this data we can say that we can use the data LSTM for the air pollution forecasting.

In this studies[5] they have used the three neural network model which are (1) Multilayer Perceptron Layer (MLP) (2) Radial Basis Function (RBF) (3) Square Multilayer Perceptron (SMLP), among all this network model the RBF has performed well while executions. Combination of neural network model with the traditional Machine Learning algorithm also shows that the combination of various model can can improve the accuracy.

We can collect the data from the wireless sensors and give it to the trained LSTM model for real time prediction of data for the air pollutions[2]. Collect the real time data for some days and create model based on that collected data.

1. **SYSTEM ANALYSIS**
   1. **EXISTING SYSTEM**

In the current scenario if we have to see for the air pollution then we can find the real time data means the data what the sensors has collected from the atmosphere. From that we can see the air quality index. There are many weather forecasting API but in that they are not providing the air quality forecasting of data. They will give us the information like temperature forecasting , rain forecasting , snow forecasting etc.. but they are not including the air pollution forecasting. The air pollution data are available on the government website and we can see that but that is real time what is going on currently in the atmosphere. The existing system is not capable of the air pollution forecasting of data.

* 1. **LIMITATION OF THE EXISTING SYSTEM**

If someone has to see the forecasting of data there is no any application which can provide the data for the air pollution. Which can lead to the problems like the breathing of polluted air.

* 1. **PROPOSED SYSTEM**

In the proposed system we are developing the system which can forecast the air pollution using the famous Deep Learning technique Recurrent Neural Network which are LSTM and GRU. Based on these 2 algorithms we are forecasting the data. For that we have to provide the data for the forecasting to that model. We are giving last 15 days on the hourly basis to this algorithm and based on that it will forecast the next 24 hours of data for the air pollution. We have used that algorithm both LSTM and GRU for doing this task. Even we have implemented the combine model which are LSTM + GRU for the forecasting of data. The model is quite accurate while forecasting of the data.

* 1. **ADVANTAGES OF PROPOSED SYSTEM**
* Provide the better accuracy.
* It will generate the next 24 hours of forecast for the air pollution.
* Provide the last 15 days of data and it will forecast the next 4 hours of data.

1. **DATASET**

The dataset what we have used for this project is taken from the Kaggle ([Link](https://www.kaggle.com/datasets/rupakroy/lstm-datasets-multivariate-univariate/code)). The dataset is for air pollutions based on the US embassy in Beijing, China. The data has collected every hour for the five years. The dataset is starting from 02-01-2010 00:00 to 31-12-2014 23:00 in total it is of 4 years in total.

The columns include in the dataset are :

1. The first column is of date & time which shows at what date and time the data has collected.
2. The second column is of PM2.5 concentration which is air pollution the data what we have to forecast.
3. The third column is of dew point which is the atmospheric temperature below which water droplets begin to condense and dew can form.
4. The fourth column is of temp which shows at that time what is the temperature at that time.
5. The fifth column is of press which is pressure the force exerted on a surface by the air above it as gravity pulls it to Earth.
6. The sixth column is of wind direction, the wind direction is of combined wind direction in which direction the air is flowing.
7. The seventh column is of wind speed, the wind speed is of cumulative wind speed for the entire hour.
8. The eight column is of snow, the cumulative hours of snow for the hour.
9. The ninth column is of rain, the cumulative hours of rain for the hour.

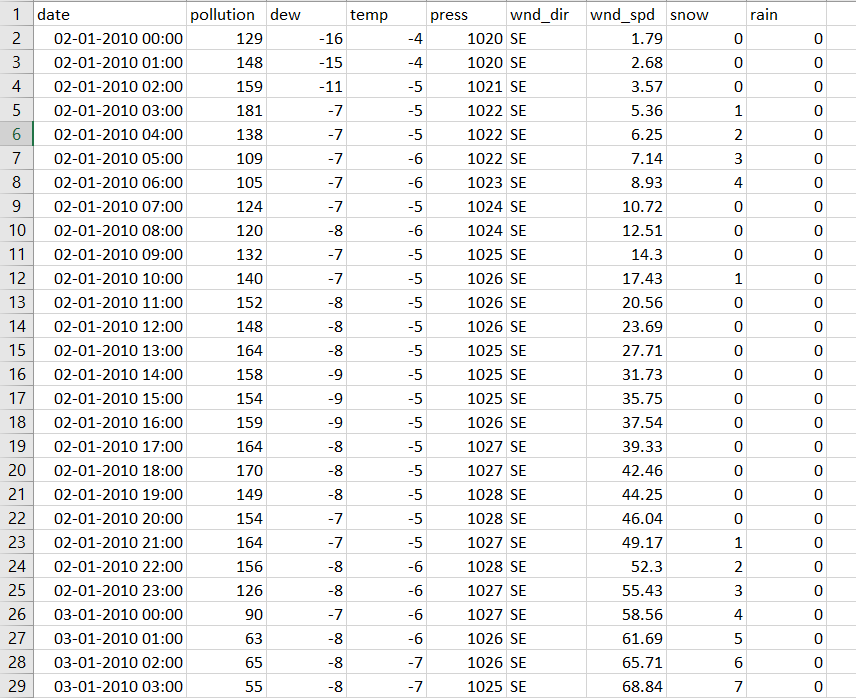


Figure 2 The sample data for the dataset which has collected

1. **SYSTEM DESIGN AND DEVELOPMENT**
   1. **METHODOLOGY**

we are going to use the most popular framework of Deep Learning which is LSTM (Long Short term memory) and GRU (Gated Recurrent Unit). As we all know LSTM which has the special ability for storing the previous execution data and store in the memory and can be used for predicting data. Recurrent Neural networks is not efficient when it comes to the long sequence of data. To overcome that we have introduce the LSTM(Long Short Term Memory) and GRU(Gated Recurrent Unit) which are useful when we have a long sequence of data.

LSTM and GRU were introduce to overcome the problem of long sequence of data. What it do is that it will they have internal mechanism of memory cell (The combination of gates is called memory cell). When the data are provided to the memory cell then it will define that whether we have to put the data or remove the data. The LSTM has a similar control flow as the recurrent neural network. It will send the data back again to it only while doing forward propagation. Every LSTM memory cell has the following Sigmoid and Tanh Activation function and three gates which are Input Gate, Forget Gate, and Output Gate.

If you see in the dataset the column with the name wnd\_dir which indicate the wind direction in which direction the wind is flowing. If we will see clearly the data are represent in the text values with the categorical data. And we cannot give that kind of data to the model because it will not understand the data at all. So for that we will convert that data’s in to the numerical values means we have to encode the data and to encode the data we have used label Encoding method.

This approach is simple and it involves converting each value in a column to a numbers. Taking a consideration of our dataset which has the categorical data. It will convert the value SE as to 1 and some other values to 2 and so on. If the same value appear again then it will take the same values and take the same output also like for SE it will take 1 only if it appear again in the dataset.

The dataset what we have collected has many numerical values. In that we have a large value like 100 or like this. If we give these values to our model then our model will be overfit. So, it will not generate the proper output bases on the given values. If we visualize the data the data will be varying from max to min values. To give it to the proper values and give the appropriate relation between the prediction values and the features. To take them in the relation we will use the Minmax scaler which is used to transform the values in to the between 0 and 1. When the values come to the values between 0 to 1 the model will give us the proper result.

Min Max scaling is the most used normalization technique in the Machine Learning. As we have discussed after scaling it will transform data to the range [0,1] meaning that the minimum and maximum values of feature/variable is going to be 0 and 1 respectively.

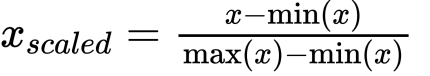


Figure 3 The Formula for the mix max scaler

Based on this equation the values will convert between [0,1] and give it to the model for the creating a model.

As we have used the 2 famous algorithm of Recurrent Neural Network which are LSTM and GRU. We will discuss both the algorithm and understand the working of that both the algorithms.

The working of LSTM(Long Short term memory) network. The LSTM required because in the RNN we have problem of gradient vanishing. LSTM have feedback connections which make them different to more traditional feedforward neural networks. This property of LSTM enables LSTM to process entire the sequence independently , but rather information about previous data in the sequence to help with the processing of new data point. LSTM model dependent on the three things:

* The current long-term memory of the network – know as the cell state
* The output at the previous point in time – known as the previous hidden state
* The input data at the current time step.

In total LSTM has three layers Input gate , forget gate and output gate.

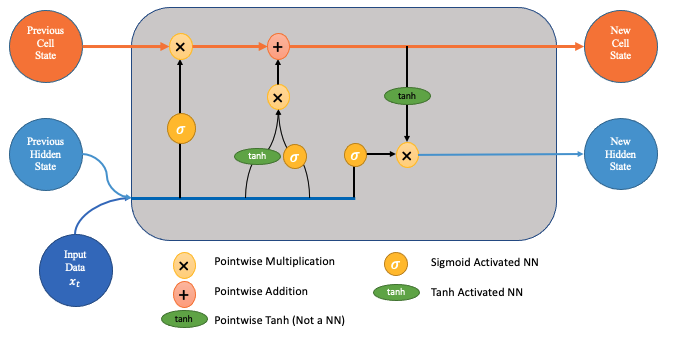


Figure 4 The LSTM architecture

As we can see in the figure that in LSTM we have 3 gate. LSTM using the sigmoid activation function for the in their gate architecture. If the outcome for the sigmoid is 0 then forget the data and if the outcome is 1 then put the data and go further step. In Forget gate, it will decide whether the data which we are providing for the processing are important or not based on the output of the sigmoid function.

In LSTM we have 3 gates namely forget gate, Input gate and output gate. Firstly, when data comes from the previous cell state it comes into the forget gate here, we are defining that weather we have to put the data or forgot the data. We will do that with the help of the sigmoid activation function (The output of sigmoid function will 0 or 1) after generating the output based on the input which we have received from the previous cell state. We will multiply that data to the current cell state So if the output will be 0 then we will forget the data or else if 1 then we will keep data and store into the current cell state. Here the task of forgot gate will over. Now the work for the input gate started and in that we will give the data from the previous cell state. Here firstly we will generate the output from the sigmoid activation function and along with that we will generate the output for the tanh (It will generate the output between -1 and +1) activation function. Then we will multiply it and add it to the current cell state. From the input gate we can understand that whether the data which we providing to the current cell state is important or not. Here the work of input gate will over. After that it comes the output gate in which we will first generate the output for the tanh activation function with the help of current cell state and then generate the output for the sigmoid of the previous cell state and then we will do product of that and send it to the next cell state. And then for the next cell state the process will continue like this.

When the data comes in to the cell state the first gate which the data will give it to is Forget gate. As the name implies that the gate will forget the data which comes in to their architecture. Here we will decide which bit of the cell state are useful give both the previous hidden state and new input data.

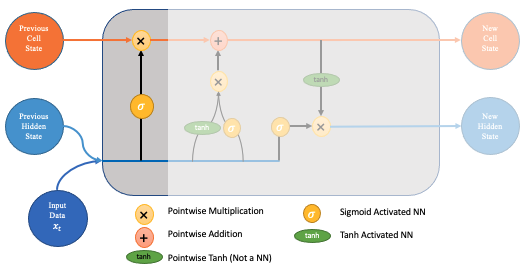


Figure 5 The Forget Gate for the LSTM

The previous hidden state and the new input data are fed into a neural network. This network generates a vector where each element is in the interval [0,1] (ensured by using the sigmoid activation). This network (within the forget gate) is trained so that it outputs close to 0 when a component of the input is deemed irrelevant and closer to 1 when relevant. It is useful to think of each element of this vector as a sort of filter/sieve which allows more information through as the value gets closer to 1.

The sigmoid activation function converts the data between 0 and 1. That is helpful to update or forget data because any number multiplied by ‘0’ will convert to 0, which can consider forgetting the data from the memory cell. And if the output is 1 then the value should be considered as the important data and kept that data in the memory cell. As we can see in the figure that the graph is from 0 to 1 and convert the values between 0 and 1.



Figure 6 Showing the range of sigmoid activation function (0-1)

These outputted values are then sent up and pointwise multiplied with the previous cell state. This pointwise multiplication means that components of the cell state which have been deemed irrelevant by the forget gate network will be multiplied by a number close to 0 and thus will have less influence on the following steps.

The second step us the input gate. The goal of this step is to determine what new information should be added to the networks long-term memory (cell state), given the previous hidden state and new input data.

At the first stage of input gate, the data which we will provide it to the current time stamp it will check whether the data what we have given it to the it will important to the model or not. If it is important then it will take that data and give it to the model for further training.

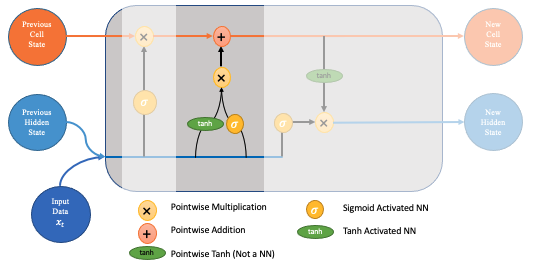


Figure 7 The Input gate of the LSTM

This can be done by the activation function called tanh activation function. What it will do is that it will combine the previous hidden state and new input data to generate a new memory update vector. This vector essentially contains information from the new input data given the context from the previous hidden state. This vector tells us how much to update each component of the long-term memory (cell state) of the network given the new data.

Note that we use a tanh here because its values lie in [-1,1] and so can be negative. The possibility of negative values here is necessary if we wish to reduce the impact of a component in the cell state.

As similar to the sigmoid Activation function even tanh is also playing a major role in the LSTM memory cell. Tanh will squish the values between -1 and 1 and this is useful in the Input gate and convert the values between -1 and 1 and multiply with the existing data of that Input gate.



Figure 8 Comparing the range of activation function and sigmoid activation function

At the second stage of input gate, it will check for the data whether it is important or not, at the first stage it will not check for that. In this second stage the sigmoid (0,1) activation functions come in to the picture from this we can understand the data what we have provided are valid data or not or whether we have to put that data for the model or leave it.

The output for this input gate will be product of the first stage output and second stage output. The resulting combine vector is then added to the cell state, resulting in the long-term memory of the networking in the long-term memory of the network being updated.

At the final we have output gate which will decide the new hidden state. To decide this, we will use three things; the newly updated cell state, the previous hidden state and the new input data. For this we will apply filter same as the forget gate. The inputs are the same and the activation function is also sigmoid ([0,1]).

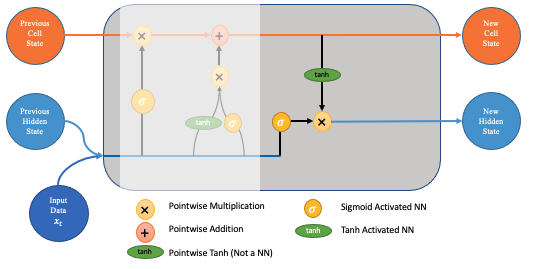


Figure 9 Output Gate for the LSTM architecture

The output gate will apply the filter to the newly updated cell state. This will take care that only necessary information will go in the output. Before applying the filter , we pass the cell state through the tanh activation function to force the values to change into the intervals of -1 and 1.

Same for the GRU (Gated recurrent unit). We can consider this as same as the LSTM because both are build on the same architecture. The GRU comes into the picture because in LSTM it is taking too much time and more computational power also to generate model and calculate all the values and create a pattern.

In GRU we only have 2 gates which are reset gate and update gate. The reset gate is similar to the forgot gate in the LSTM, it will justify whether the data from the previous cell state we have to keep it or reset that cell state. After that it comes the update gate. In update gate we will update the cell state with the help of sigmoid activation function and add the output of that with -1 and multiply it to the cell state. For the output of the Memory cell it will multiply with tanh activation function and then multiply with the update gate output and add it to the current cell state.

The architecture for the LSTM are look like this.

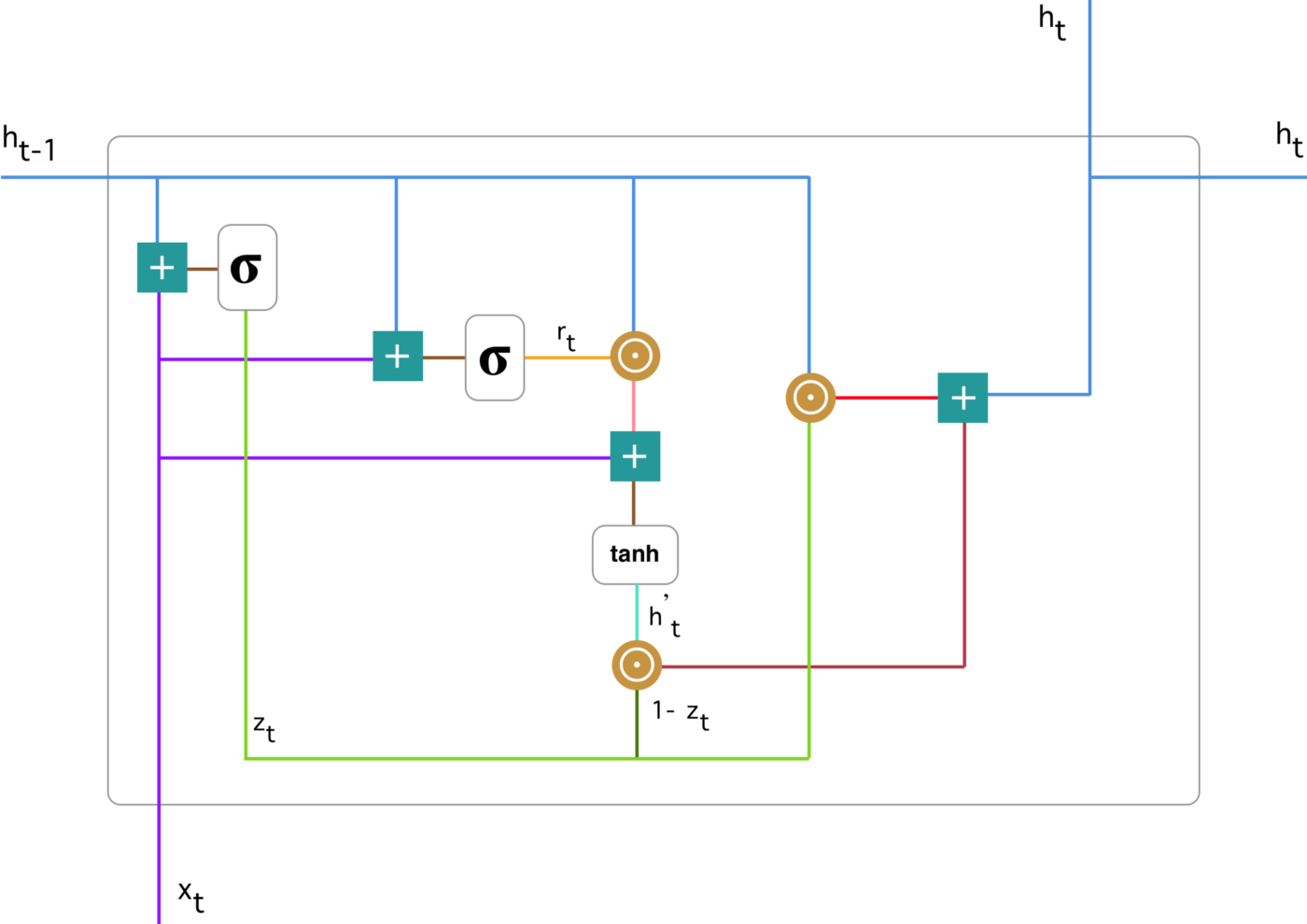
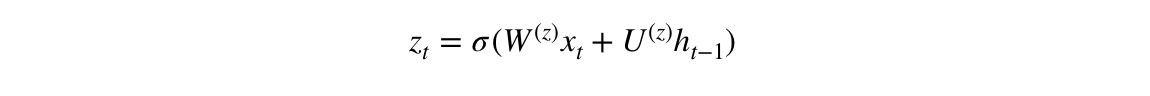


Figure 10 The GRU architecture

The GRU has manly only 2 gates which are reset gate and update gate. In this we are taking help of both sigmoid and tanh activation function which will help us to understand the data and making the decision for whether we have put the data or we have reset the data in the architecture.

The first step will be update gate which will helps the model to determine how much of the past information needs to be passed along to the future. That is really powerful because the model can decide to copy all the information from the past and eliminates the risk of vanishing gradient problem. The formula for calculating the update gate is :



When x\_t is plugged into the network unit, it is multiplied by its own weight W(z). The same goes for h\_(t-1) which holds the information for the previous t-1 units and is multiplied by its own weight U(z). Both results are added together and a sigmoid activation function is applied to squash the result between 0 and 1. Following the above schema, we have:

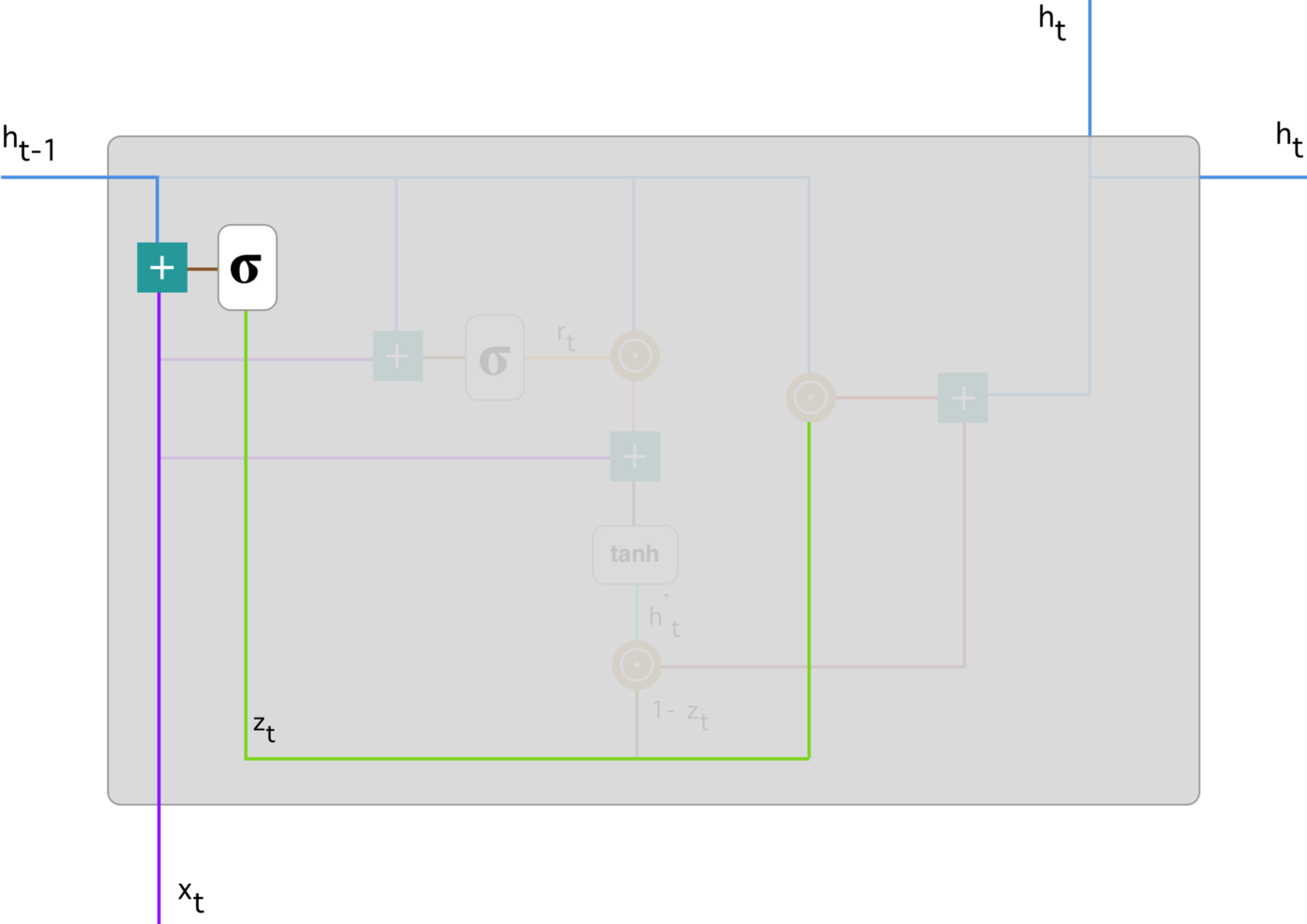
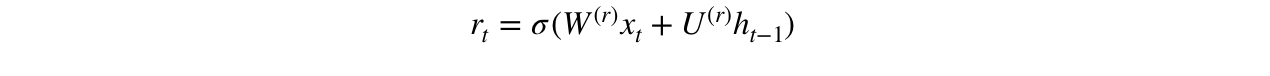


Figure 11 Update gate for the GRU

The second gate will be of reset gate, this gate is used from the model to decide how much of the past information to forget. It means it will determine which data is important in the form of the model creation and which information if we forget that it will not affect the model. The equation for the reset gate will be :



This formula is the same as the one for the update gate. The difference comes in the weights and the gate’s usage, which will see in a bit. The schema below shows where the reset gate is:

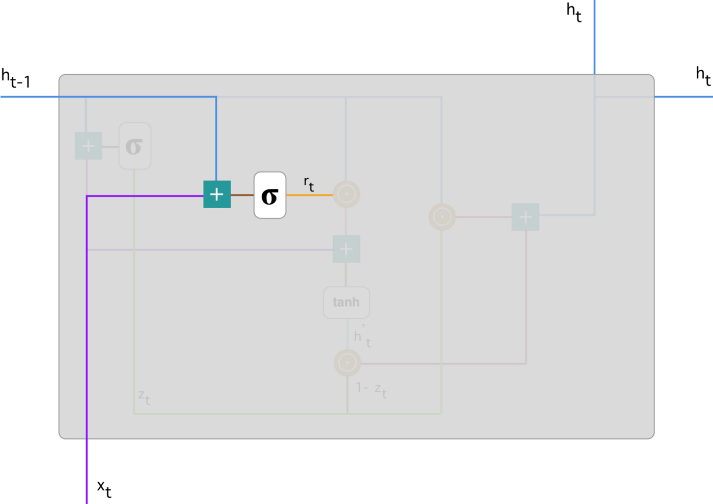
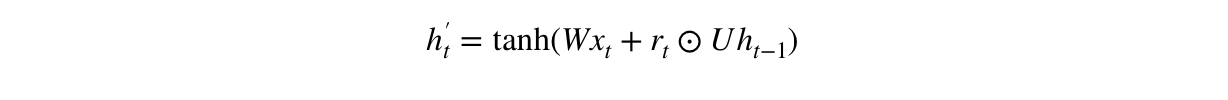
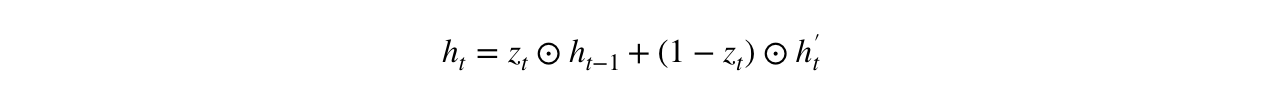


Figure 12 Forget gate for the GRU

We will understand now how exactly the gates are affecting to the final output of the model. In the first step we have done the reset gate. We introduce the new memory content which will use the reset gate to store the relevant information from the past. It can be calculated as



The second one will be update gate vector which holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content — h’\_t and what from the previous steps — h\_(t-1). That is done as follows:



We have used both the LSTM and GRU models for forecasting data. In our model, you have to give the last 15 days of data with the parameters of {date, pollution, dew, wind\_dir, wind\_spd, snow, rain, pollution} based on that we can forecast the next 24 hours of data which is pollution. We have to build a model to predict the next 24 hours of data. In this, the date is playing the most important role in forecasting data.

* 1. **IMPLEMENTATION**

For implementing the LSTM and GRU algorithm we have used TensorFlow library. For implementing this we have followed several steps which are as follow:

1. Data Pre-processing
2. Creating Model
3. Saving Model
4. Generate Output
5. Data Pre-processing:

For Data Pre-processing firstly we have to understand the data which we have. To understand the data we can use the EDA. Firstly if in our data set there is a requirement for the Encoding of data we have to do that we can do that with the help of a Label Encoder or One Hot Encoder. After that, we have to normalize the data so our model can learn from that without getting more confused.

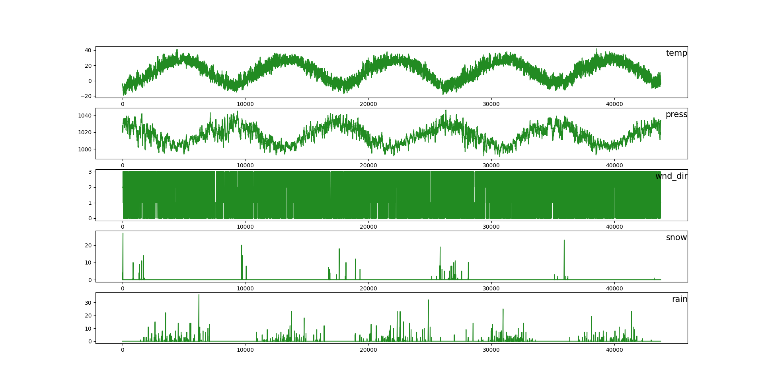


Figure 13 Represent of data variable from dataset

This graph in figure 5 represents the input variable data from the dataset and visualize it. It will easy to understand the data from which we can organize the data.

For Label Encoding process of the data the data should be processed and generate the one unique number for the each and every unique data in the column of that dataset. But in the case of One Hot Encoder the for every unique data in the column It will generate one number and rest of them assign to the same values.

The label encoder is required the dataset column name wnd\_dir as in that we have categorical values for the wind directions. We can achieve this by implementing using the sklearn library and in that library we have the method with the name of LabelEncoder. Before implementing this to our dataset in total we have 4 types of data in the wind direction column with name :

1. SE (South East wind)
2. CV (Calm and Variable wind)
3. NW (North west wind)
4. NE (North East wind)

So if we will give this values to our model it won’t accept the data itself. Because our model need the mathematical data to understand the relation between the feature values and prediction values. For that we need to encode the data with the help of LabelEncoder.

For that we will load the LabelEncoder from the sklearn firstly after than we will convert the text values into the numerical values.

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

dataset[dataset\_index] = encoder.fit\_transform(dataset[dataset\_index])

After implementing LabelEncoder to our dataset column our dataset will look something like this:

1. SE will convert to 2
2. CV will convert to 3
3. NW will convert to 1
4. NE will convert to 0

Now in our dataset we have only numerical values in the entire dataset we have converted that one text data into the numerical values. But when it comes to the dataset we can see that we have the values which are too big for our model. The data which we have has many variations in it. Because of that our model cannot learn the relation between the prediction values and features values. To overcome that we will normalize the data into the smaller values.

To do that we have used the Min Max scaler that is used to scale the data in to the variation of 0 and 1. Because of the scaling the data the all the data will come into the range of 0 and 1 so our model can understand the relation between the data in the dataset.

To implement this firstly we have to load the dataset in the induvial variable and transform it to the range of 0 to 1. But before that we have to do the reshaping of the data with the help of reshape method. We have to do this for the all the features available in the dataset.

As in our dataset we have in total 11 features and 1 prediction variable So we have to do the normalization of all the 11 features for our dataset.

x\_1 = dataset['dew'].values

x\_2 = dataset['temp'].values

x\_3 = dataset['press'].values

x\_4 = dataset['wnd\_spd'].values

x\_5 = dataset['wnd\_dir'].values

x\_6 = dataset['snow'].values

x\_7 = dataset['rain'].values

x\_8 = dataset['year'].values

x\_9 = dataset['month'].values

x\_10 = dataset['day'].values

x\_11 = dataset['hour'].values

y = dataset['pollution'].values #Prediction value

As we can see we have loaded all the dataset feature and prediction values to the specific variables.

If we visualize our dataset based on the actual values it has many large numbers which can lead to the problem for us. The data we can see as below :

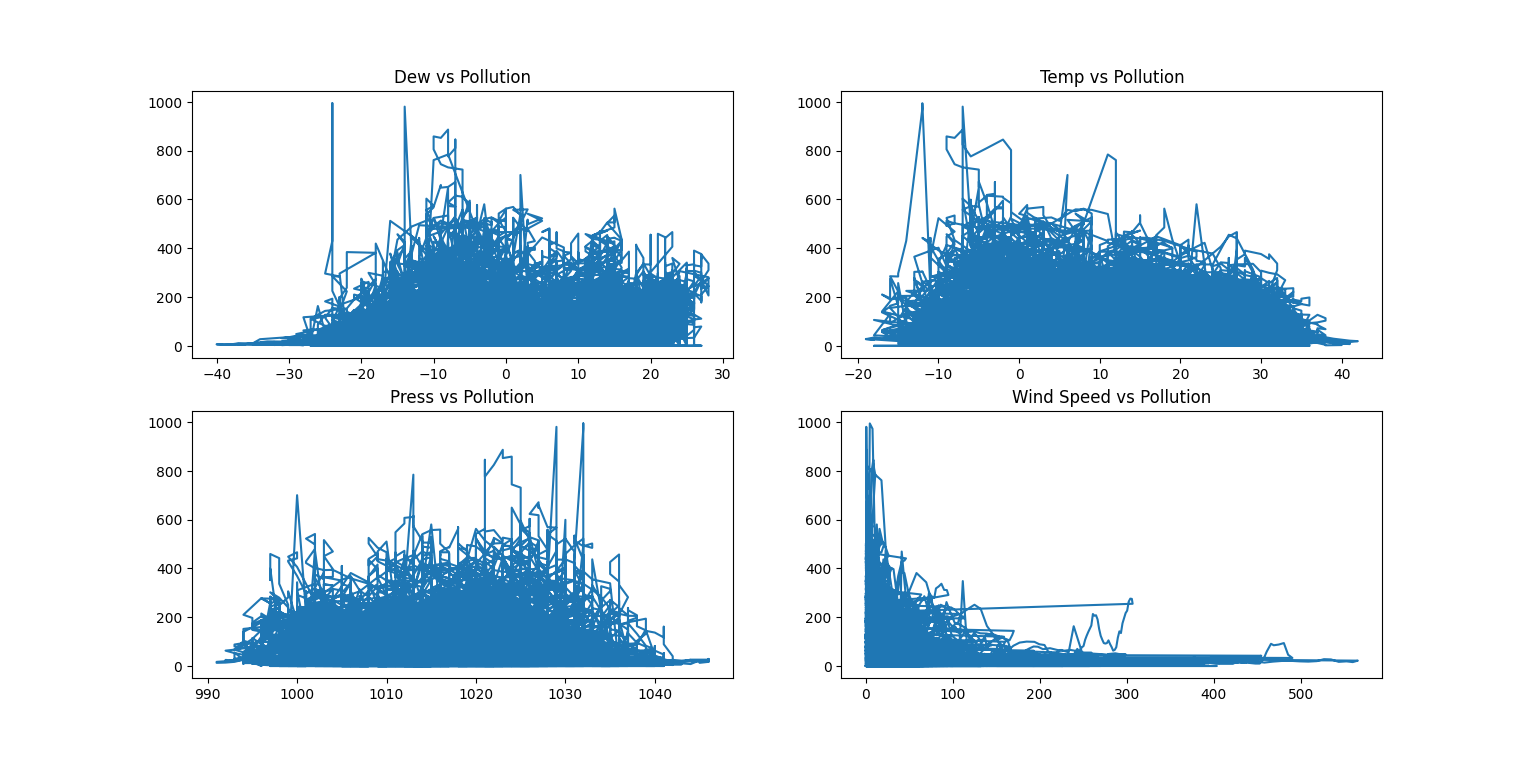


Figure 14 Dataset before normalizing data

As we can see that all the values are ranging from the 0 to 1000 for the pollution in the y-axis data and for the x-axis we have different features which are dew, temp, press and wind speed. We have only take the 4 parameters for this visualizing the data.

This below graph in figure 15 represent the data for the output variable for the last 15 days of data with respect to every hour which mean it shows the data of 360 hours. But in this we have plot without scaling the data. As we can see in the y-axis the value of the air pollution with respect to the number of hours which is 360.

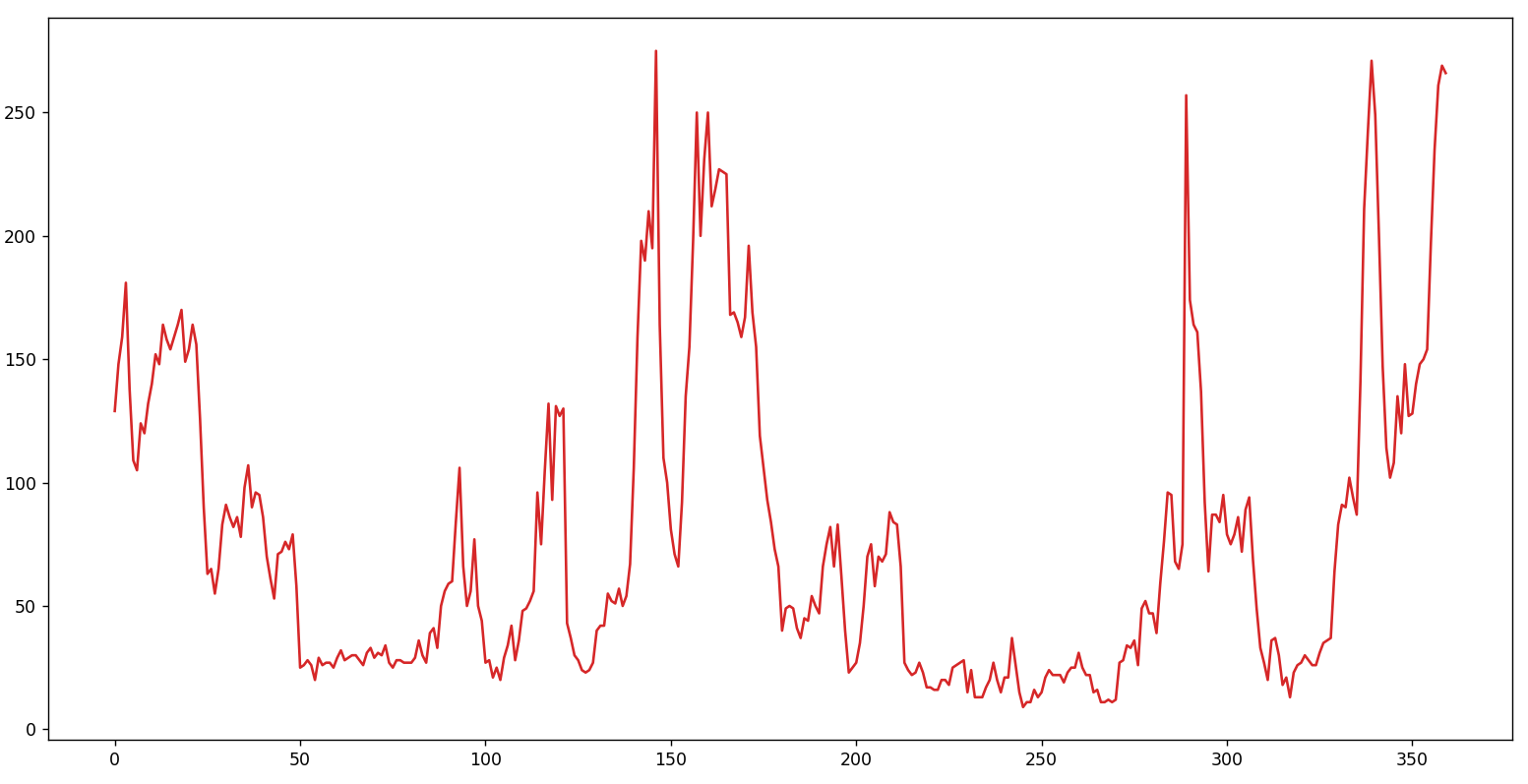


Figure 15 Data visualization for the Air Pollution on the hourly basis for the last 15 days which is 360 hours of data

For selecting the features which are appropriate for the model we have to understand the co-relation between the data. From that we can select the features which are useful to us. If we will select the features more which are highly co-related to the each-other then our model will come in the state of overfitting. Which means our data will work more accurate with the training data but when it comes to the testing data it will not predict accurate.

We have created the co-relation metrics for the dataset and but the dataset is not in the normalized data. From this we can understand the dataset in the manner of how well the dataset are interconnected with each other. It will compare all the dataset column with each other and generate the graph with the values between 0 and 1. 0 means the features are not well co-related and the values 1 means the features are highly co-related.

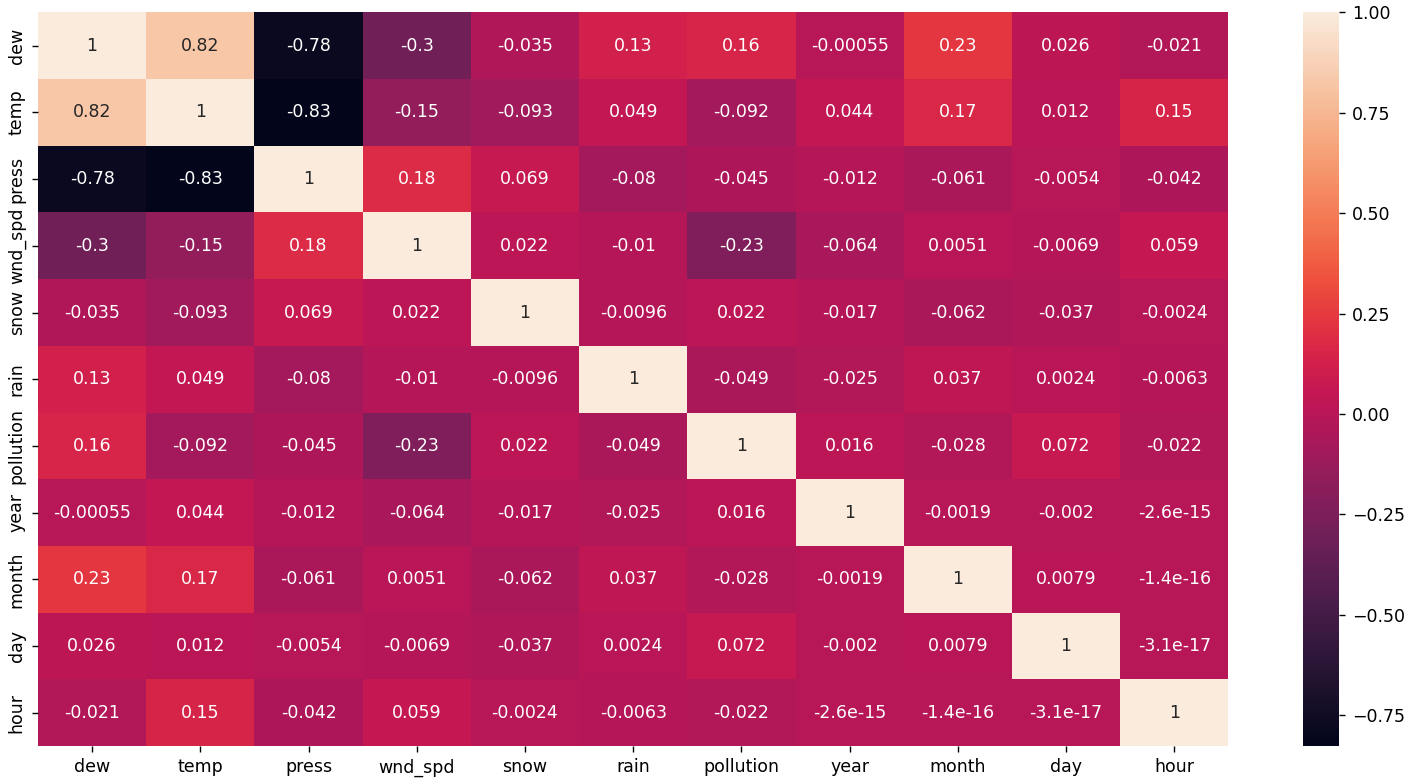


Figure 16 Heat map for the co-relation of data of the dataset

Now we will do the normalization of data with the help of Min max scaler of all the features and the output variable also. We can do that by applying this code.

scaler = MinMaxScaler(feature\_range=(0, 1))

x\_1\_scaled = scaler.fit\_transform(x\_1)

x\_2\_scaled = scaler.fit\_transform(x\_2)

x\_3\_scaled = scaler.fit\_transform(x\_3)

x\_4\_scaled = scaler.fit\_transform(x\_4)

x\_5\_scaled = scaler.fit\_transform(x\_5)

x\_6\_scaled = scaler.fit\_transform(x\_6)

x\_7\_scaled = scaler.fit\_transform(x\_7)

x\_8\_scaled = scaler.fit\_transform(x\_8)

x\_9\_scaled = scaler.fit\_transform(x\_9)

x\_10\_scaled = scaler.fit\_transform(x\_10)

x\_11\_scaled = scaler.fit\_transform(x\_11)

y\_scaled = scaler.fit\_transform(y)

As we can see that the feature range is varying from the 0 to 1. After that we will apply fit\_transform for all the variables. After this all the data are converted into the range of 0 and 1 and we will store this all the variable into the new dataset with the help of hstack. And we will create another dataframe for our dataset.

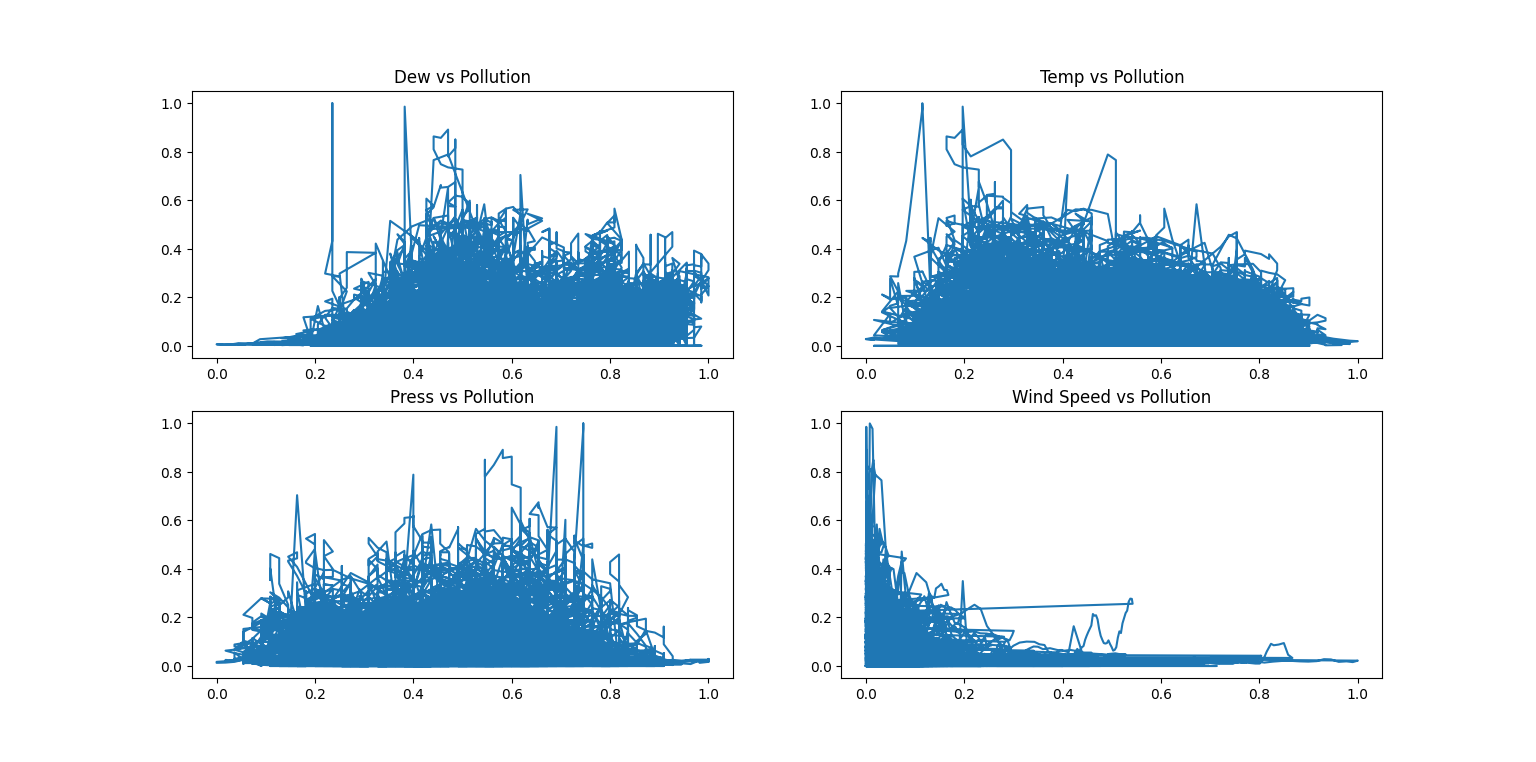
After that if we visualize the data which are normalized will look something like this.

Figure 17 The dataset visualization after normalizing the data

Here we can see the data are ranging from the 0 to 1 for both the axis x and y. Here we can see that for the all the four input parameters and output parameters. Now we can see that all the ranges are from 0 to 1. And for the same for the prediction data variable it will generate the graph for the prediction value with the normalized data from the dataset.

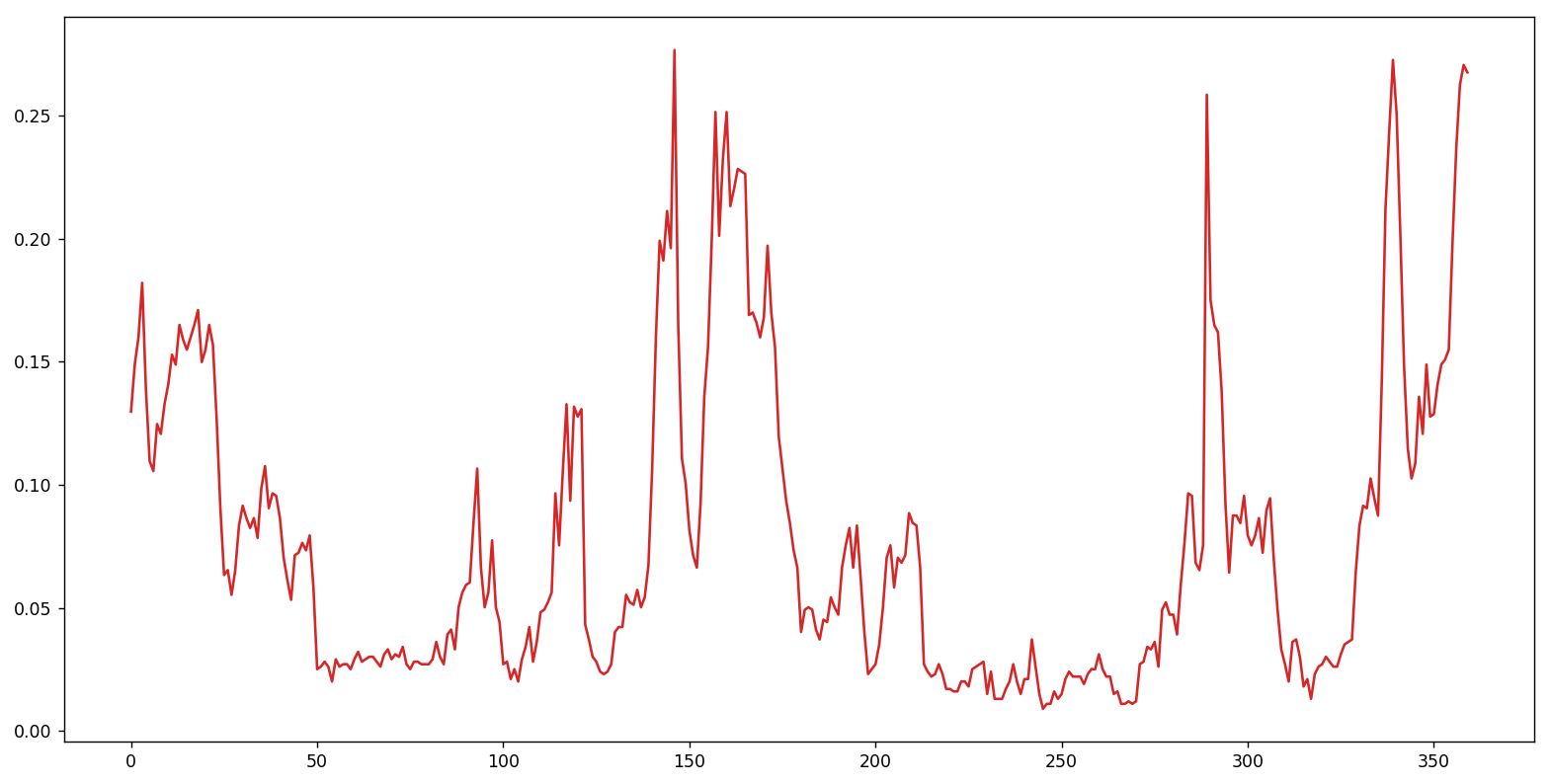


Figure 18 The Normalized data for the prediction variable pollution

As we can see that now the values are range from the 0.0 to 0.30 values. From the 15 days of data which is 360 hours of data on the hourly basis. Now if we will give this data to the model or model can learn more relation between the dataset and can generate the more efficient model.

1. Creating a model:

In total we have created 2 models which are LSTM model , GRU model.

For creating the GRU model , we have to create a GRU model. For this also we have used Input layer of 360 neurons which is 15 days on hourly basis. Two hidden layers with each of 50 neurons in it. And also added the Dropout layer which is also required to add.

The model architecture would look like this for the GRU model.

|  |  |  |
| --- | --- | --- |
| Layer (Type) | Output shape | Param # |
| gru (GRU) | (None , 360 , 50) | 9450 |
| dropout (Dropout) | (None , 360 , 50) | 0 |
| gru\_1 (GRU) | (None , 360 , 50) | 15300 |
| dropout\_1 (Dropout ) | (None , 360, 50) | 0 |
| gru\_2 (GRU) | (None , 50) | 15300 |
| Dense (Dense) | (None , 24) | 1224 |
| Activation (Activation) | (None , 24) | 0 |

In this architecture we have 2 layers of GRU and 2 layers of Dropout layer with 0.2. The input layer has the input shape of the (360,11). With the activation function of tanh and return activation function of sigmoid. We have added the dropout layer because we will remove some neurons which are not useful and the neurons which are not connecting in the next layer that kind of neurons are not important for us. So we will drop that neurons to reduce the tense of model. After that it start with the hidden layers of model in that the first hidden layer has the data with the 50 neurons in it. It will take the data as input from the previous layer. For the output layer we have used the dense layer with the activation layer of the linear activation function. Same as for the second hidden layer. For this architecture the total parameters we have is 41,274 and from that trainable parameter are 41,274. In this architecture also we have to provide the input for the last 15 days of data on the hourly basis and it will forecast the next 24 hours of data according to GRU model architecture.

For this to compile the model we have used the Mean square Error for visualizing the error rate for the model. And the optimizer of the adam optimizer with the learning rate for 0.001. And the metrics of the accuracy.

opt = keras.optimizers.Adam(learning\_rate=0.001)

# define model

model = Sequential()

# Creatin a model

model.add(GRU(50, activation='tanh' , recurrent\_activation='sigmoid' , return\_sequences=True , input\_shape=(n\_steps\_in, n\_features)))

model.add(Dropout(0.2))

model.add(GRU(50 , activation='tanh' , recurrent\_activation='sigmoid' , return\_sequences=True ))

model.add(Dropout(0.2))

model.add(GRU(50 , activation='tanh' , recurrent\_activation='sigmoid'))

model.add(Dense(n\_steps\_out))

model.add(Activation('linear'))

model.compile(loss='mae', optimizer=opt , metrics=['accuracy'])

For creating LSTM model, we have used the same parameters like GRU model creation. For creating a LSTM model, we have used the same layers for like Input layer, hidden layers and output layer. Firstly, the input layer is of 360 neurons and alongside of that we have 2 hidden layers each of 50 neurons and 2 layers of Dropout layer at the end we have Dense layer which is output layer.

The architecture for the LSTM model look like this

|  |  |  |
| --- | --- | --- |
| Layer (Type) | Output shape | Param # |
| lstm (LSTM) | (None , 360 , 50) | 9450 |
| dropout (Dropout) | (None , 360 , 50) | 0 |
| lstm\_1 (LSTM) | (None , 360 , 50) | 15300 |
| dropout\_1 (Dropout ) | (None , 360, 50) | 0 |
| lstm\_2 (LSTM) | (None , 50) | 15300 |
| Dense (Dense) | (None , 24) | 1224 |
| Activation (Activation) | (None , 24) | 0 |

For this to compile the model we have used the Mean square Error for visualizing the error rate for the model. And the optimizer of the adam optimizer with the learning rate for 0.001. And the metrics of the accuracy.

opt = keras.optimizers.Adam(learning\_rate=0.001)

# define model

model = Sequential()

# Creating a model

model.add(LSTM(50, activation='tanh' , recurrent\_activation='sigmoid' , return\_sequences=True , input\_shape=(n\_steps\_in, n\_features)))

model.add(Dropout(0.2))

model.add(LSTM(50 , activation='tanh' , recurrent\_activation='sigmoid' , return\_sequences=True ))

model.add(Dropout(0.2))

model.add(LSTM(50 , activation='tanh' , recurrent\_activation='sigmoid'))

model.add(Dense(n\_steps\_out))

model.add(Activation('linear'))

model.compile(loss='mse' , optimizer=opt , metrics=['accuracy'])

1. Saving model:

The creation of model is time consuming process. To avoid that process to do it every time we will save that model for the future use. In future if we have to use that model we can use that easily by just calling that model to our application. By the help of saving model we can use that model in any kind of application like web application , we can create a API from that and many more use case.

For creating a model we have to fir the model based on the given parameters like epoch values and step\_per\_epoch values and other values.

history = model.fit(train\_X , train\_y , epochs=300, steps\_per\_epoch=25 , verbose=1 ,validation\_data=(test\_X, test\_y) ,shuffle=False)

model.save('Air\_Pollution.h5')

We can save the model with the help of this code. And the name for the model will be the Air\_pollution.h5, and the model has saved for the future use.

1. Generate output:

The output which will generate from this model are next 24 hours of air pollution data. We just have to provide the input of the last 15 days of data with respect to the every hour which is in total 360 hours of data. We have to give last 360 hours of data and it will predict the next 24 hours of data.

To generate the output we have to first load the model which we can do by the help of tensorflow.

model = load\_model("Air\_Pollution.h5")

By this we can load the model which we have saved in the local machine. After we have to follow some steps for the data pre-processing which will pre-process the data which we have to give for the forecasting same as that we have to do scaling of data and even that we have to do normalization of data. After that we have to convert that data into the 1 Dimensional numpy array. Then we can give that model for the forecasting and based on that it will forecast the data.

y\_pred = model.predict(data)

Based on this it will predict the next 24 hours of data. But the data will be in the range of 0 to 1. Because we have normalized the data but we want the actual output of data. For that we have inverse transform the data with Min Max scaler as what they have predicted. We can achieve that by doing this :

y\_pred\_inv = scaler.inverse\_transform(y\_pred)

1. **SYSTEM DESIGN**
   1. **HIGH LEVEL DESIGN**

Give the data for the last 15 days on the hourly basis

Give it to the pre-trained

Model of GRU

Give it to the pre-trained

Model of LSTM

Generate the output for the next 24 hours

Generate the output for the next 24 hours

1. **RESULT**

We have trained the model on the basis of 3 types of epoch number 100, 200 and 300. For every model we have done this and based on this we have selected the best model which will more accurate for us. The all the 3 models has generated the output graph which will saw us how accurate is our model in every epoch levels.

Firstly we will go with the GRU model we have trained that model with the 3 epoch values which are 100, 200 and 300. With the 50 neurons in every layer. And based on that it has generated the output values and graphs. The first graph will represent that epoch values as we have train the model for the 300 epoch firstly we have said that how well the model has fit for the epoch values.

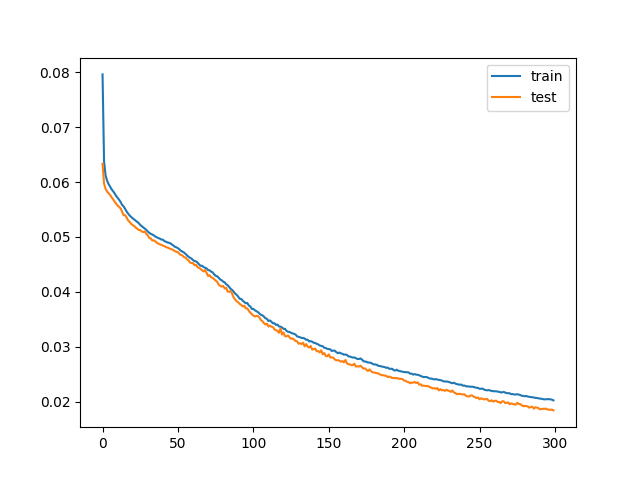


Figure 19 The graph for the 300 epoch values for the GRU with 50 neurons

And based on that trained model we have to predict the values by providing the values for the last 15 days of the data for the hourly basis. Which means in total we have 360 hours of data which we have to provide for the forecasting.

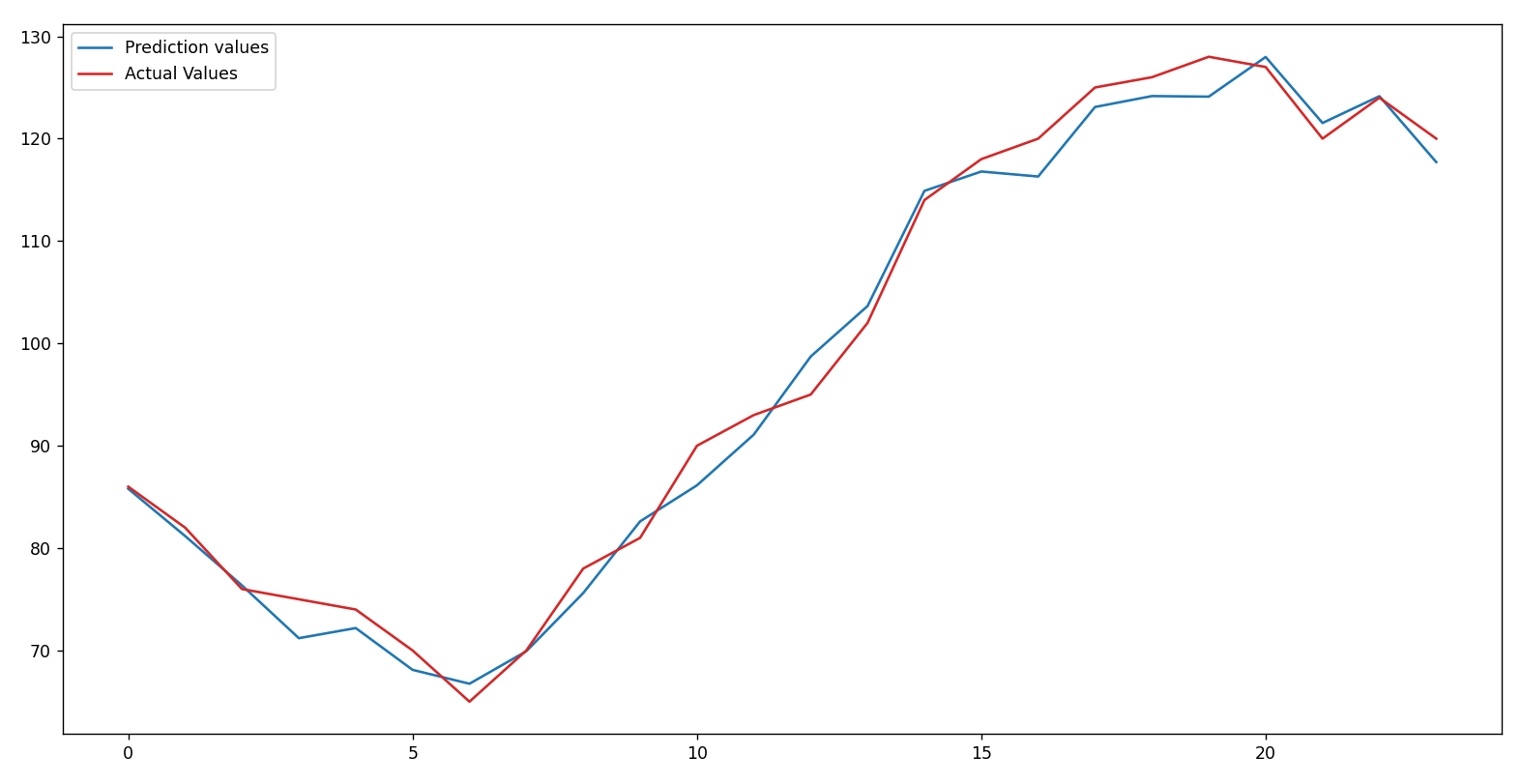


Figure 20 The forecasting of data for the next 24 hours data based on the data what we have provided for 50 neurons.

Same like this we have trained the model for the GRU and with the epoch values of 100 and 200 and generated the output for the same which is next 24 hours of data.

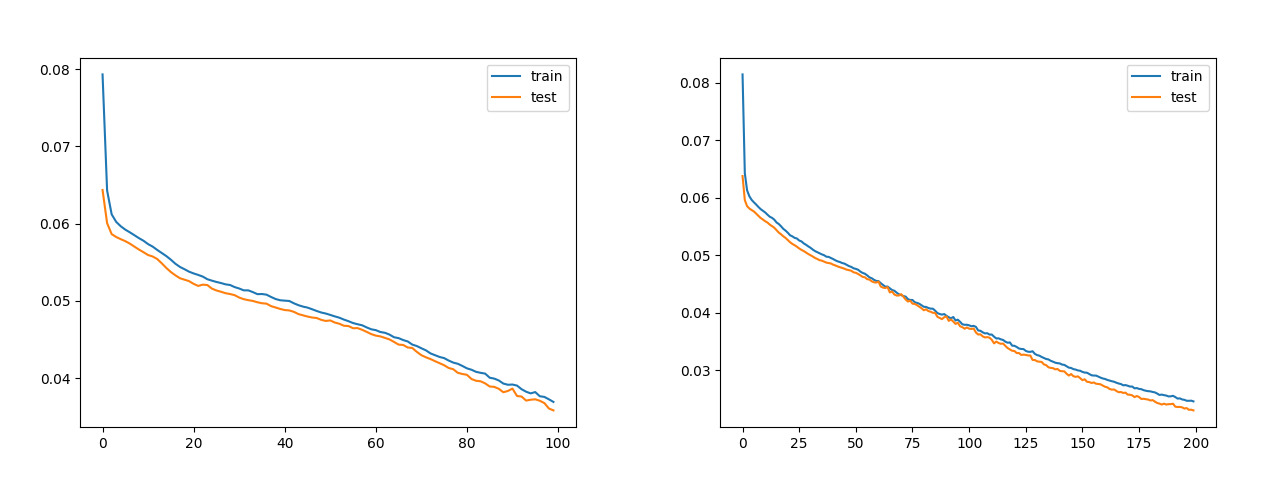


Figure 21 The graph of 100 and 200 epoch values for the 50 neurons for the GRU

The model what we have saved for both the 100 and 200 epoch values and based on that generated the output for the air pollution data. We will give the same data for the forecasting and it will generate the output. The output will generate in the graph and we have saved that model.

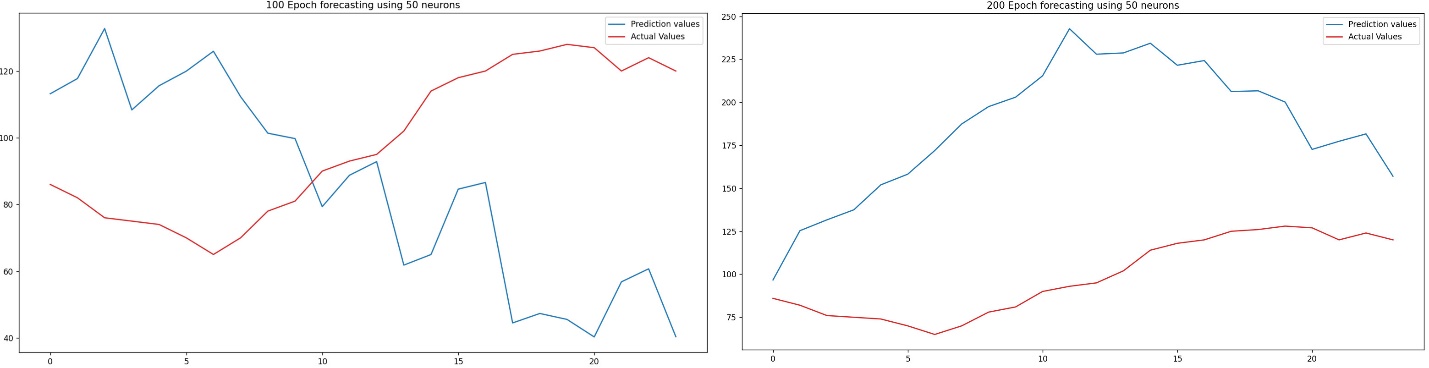


Figure 22 The graph for the both the model with 100 and 200 epoch values.

From the above given data we can see that model with 300 epoch values is performing well from all the 3 models of 100, 200 and 300 epoch values.

We have generated the same model for the 70 neurons in it each layer to see how accurate our model could be. For that we have created another model with the 70 neurons with all the 3 epoch values 100, 200 and 300. We have done this because we can compare our model with the it own means we will compare the model with 50 neurons and 70 neurons. From this we can see and understand which model is performing well in that.

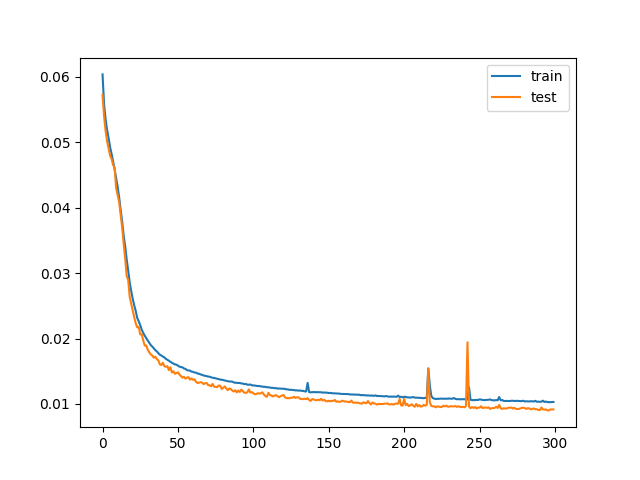


Figure 23 The train and test validation data for the model with 70 neurons and 300 epoch

The output what we have generated using the 70 neurons and 300 epoch values the graph. This is the same as we have to give the last 15 days of data for the every hour and it will generate the next 24 hours of data. It will generate the graph for the prediction value and Actual value. The graph for this model will look like this.

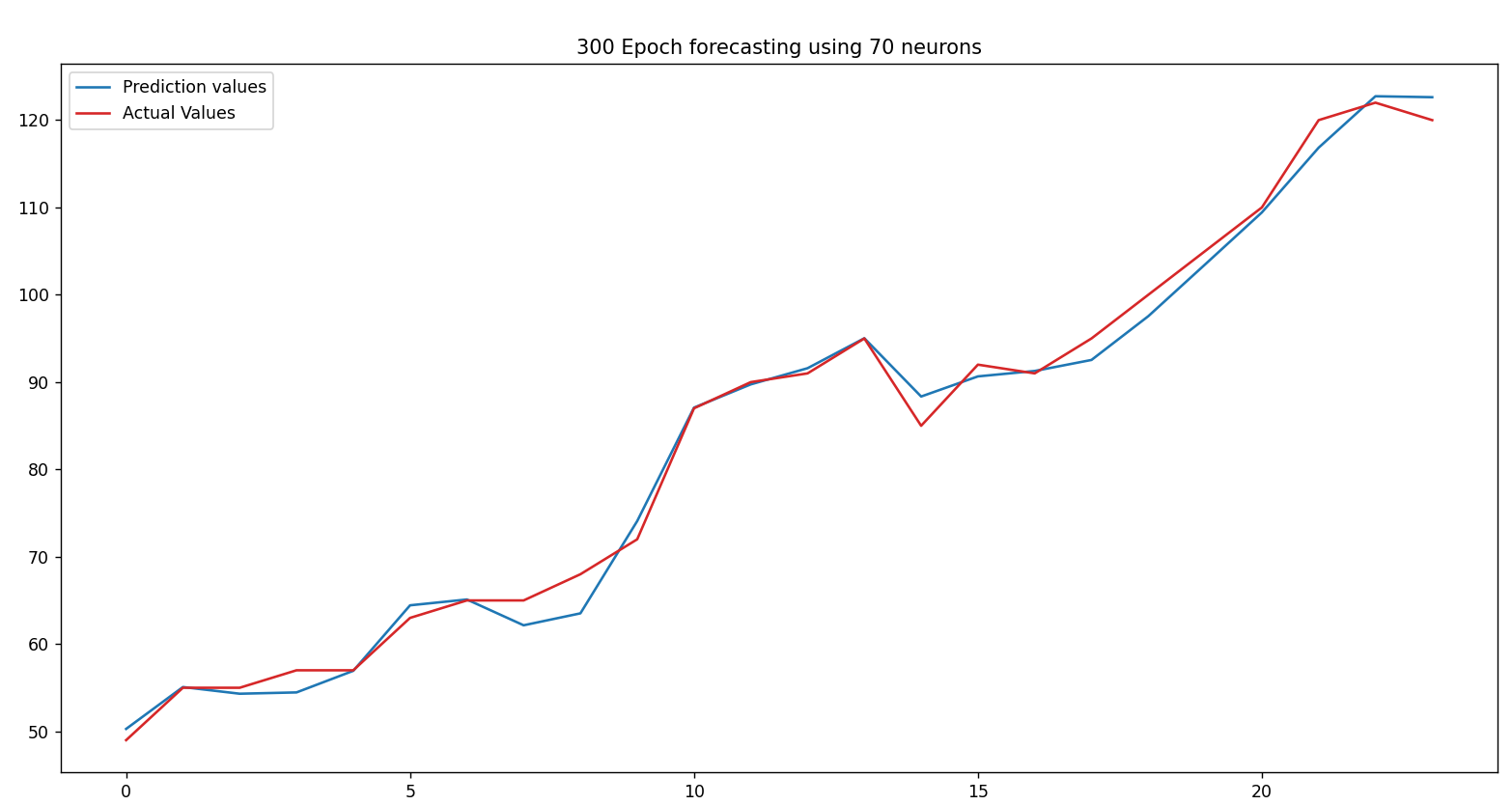


Figure 24 The output graph for the next 24 hours of data for the 70 neurons

Same for the 50 neurons in the 70 neurons also we have trained the 3 model in total with the 100, 200 and 300 epoch values from that 300 epoch values model output you can see in the figure 24 and now we will go with the model of 100 and 200 epoch values. The model architecture is same as earlier just we have done is that train the model with 100 and 200 epoch. The model training graph for both the epoch values you can see here

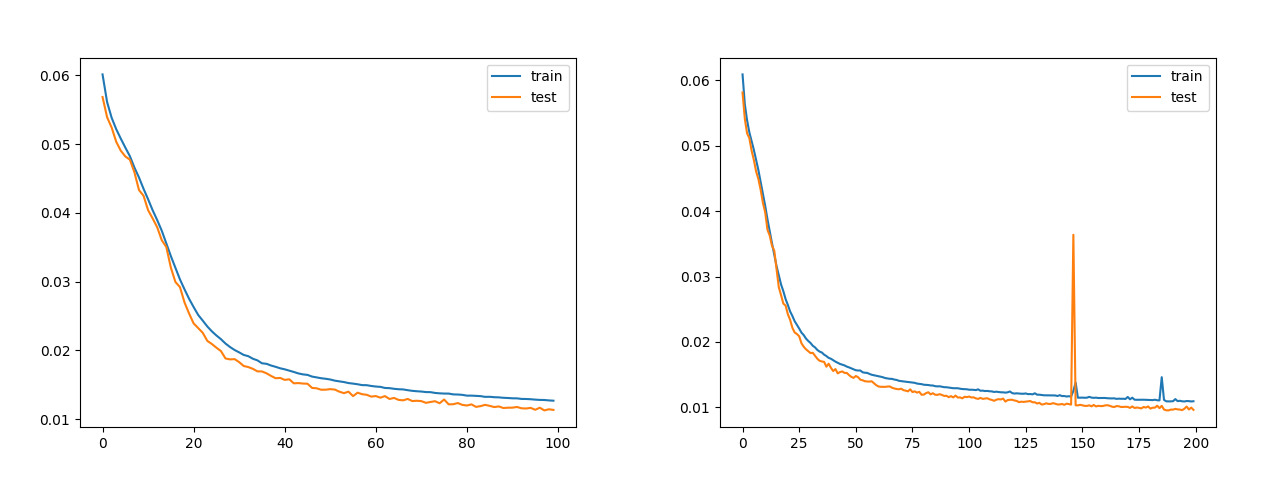


Figure 25 The training graph for the 100 and 200 epoch with 70 neurons

Based on the trained model the graph has generate and model have saved. But to see how accurate our model is we have done the prediction for the air pollution and created the graphs for that the graph is for both the prediction and accurate values. That graph for both the model predictions and actual values are.

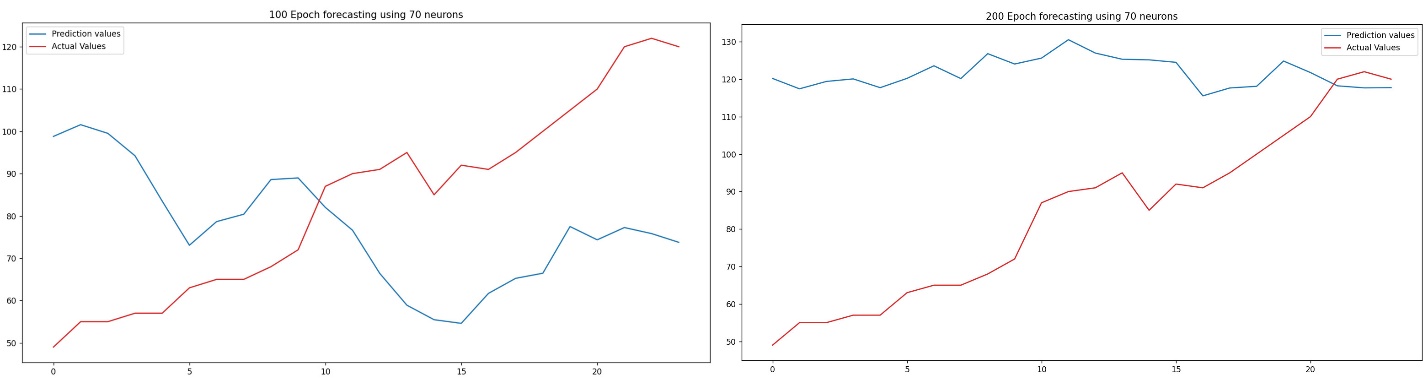


Figure 26 The output prediction graph for the actual and predicted values

1. **CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT**

With the help of this models, we can forecast the air pollution data for the next 24 hours by providing the last 15 days of data to it. As we have created 3 models for the forecasting of data that are LSTM, GRU. This will generate the output for the next 24 hours of data and the data are quite accurate also. From this everyone can see the forecasting of air pollution along side the weather forecasting which will help the users to understand the air pollution out there so he/she can take precaution so he/she can be safe from the diseases which are caused due to inhaling polluted air.

In Future we will feed more data to the model from which our model will become more accurate and learn more patterns from the data and our model will become more accurate. And also we are planning to launch this application as API from which anyone can access this API and use it to their application to show the data.

1. **BIBILOGRAPHY**

[1] K Srinivasa Rao, G. Lavanya Devi, N. Ramesh, "Air Quality Prediction in Visakhapatnam with LSTM based Recurrent Neural Networks", International Journal of Intelligent Systems and Applications(IJISA), Vol.11, No.2, pp.18-24, 2019. DOI: 10.5815/ijisa.2019.02.03.

[2] Belavadi, Sagar & Rajagopal, Sreenidhi & R, Ranjani & Mohan, Rajasekar. (2020). Air Quality Forecasting using LSTM RNN and Wireless Sensor Networks. Procedia Computer Science. 170. 241-248. 10.1016/j.procs.2020.03.036

[3] Q. Tao, F. Liu, Y. Li and D. Sidorov, "Air Pollution Forecasting Using a Deep Learning Model Based on 1D Convnets and Bidirectional GRU," in IEEE Access, vol. 7, pp. 76690-76698, 2019, doi: 10.1109/ACCESS.2019.2921578.

[4] X. Du, Y. Cai, S. Wang and L. Zhang, "Overview of deep learning," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016, pp. 159-164, doi: 10.1109/YAC.2016.7804882.

[5] J.B. Ordieres, E.P. Vergara, R.S. Capuz, R.E. Salazar,Neural network prediction model for fine particulate matter (PM2.5) on the US–Mexico border in El Paso (Texas) and Ciudad Juárez (Chihuahua),Environmental Modelling & Software, Volume 20, Issue 5, 2005, Pages 547-559, ISSN 1364-8152