

A PREDICTIVE DATA FEATURE EXPLORATION BASED AIR QUALITY PREDICTION APPROACH

A MAJOR PROJECT REPORT

Submitted by

SOURABH KUMAR MISHRA [RA1611003020133]

YASH JOYSHER [RA1611003020170]

SOWMYA. S [RA1611003020195]

Under the guidance of

Dr. N. KANNAN

(Head of Department , Department of Computer Science and Engineering)

In partial fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM CAMPUS, CHENNAI -600089

MAY 2020

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled “**A PREDICTIVE DATA FEATURE EXPLORATION BASED AIR QUALITY PREDICTION APPROACH**” is the bonafide work of **SOURABH KUMAR MISHRA [RA1611003020133]**, **YASH JOYSHER [RA1611003020170]**, **SOWMYA. S [RA1611003020195]**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an occasion on this or any other candidate.

SIGNATURE

Dr. N. KANNAN., Ph.D.,
Professor and Head,
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram Campus, Chennai.

SIGNATURE

Dr. N. KANNAN., Ph.D.,
Professor and Head
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram Campus, Chennai.

Submitted for the project viva-voce held on.....at SRM Institute of Science and Technology , Ramapuram Campus, Chennai -600089.

INTERNAL EXAMINER

EXTERNAL EXAMINER

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI - 89**

DECLARATION

We hereby declare that the entire work contained in this project report titled **“A PREDICTIVE DATA FEATURE EXPLORATION BASED AIR QUALITY PREDICTION APPROACH”** has been carried out by SOURABH KUMAR MISHRA [RA1611003020133], YASH JOYSHER [RA1611003020170], SOWMYA. S [RA1611003020195] at SRM Institute of Science and Technology, Ramapuram Campus, Chennai- 600089, under the guidance of **Dr. N. Kannan, Head of Department**, Department of Computer Science and Engineering.

Place: Chennai

SOURABH KUMAR MISHRA

YASH JOYSHER

SOWMYA. S

Date:



Own Work Declaration
Department of Computer Science and Engineering

SRM Institute of Science & Technology

Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B. Tech

Student Name : Sourabh Kumar Mishra, Yash Joysher, Sowmya. S

Registration Number : RA1611003020133, RA1611003020170, RA1611003020195

Title of Work : A Predictive Data Feature Exploration Based Air Quality Prediction Approach

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalised in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ABSTRACT

Pollution largely influences both the society and its economy, and the consequences of this are largely unfocussed and does not get its due importance. Apart from causing serious health issues, the high pollution levels have its toll on the world economy wherein unimaginable financial losses are incurred. Technological advancement in the field of pollution control and monitoring has been a big enabler for governments across the world to take corrective actions over a long-sustained period of time. The awareness and concern for pollution levels has gone up significantly among the public and authorities. This dissertation aims at creating an air quality prediction system. The prime objective of the model proposed here is to efficiently predict pollution levels with minimal error. This is done based on historical pollution information. The drastic increase in pollution levels globally has led to severe consequences. This air quality prediction system uses, a set of factors like ambient pollutant levels of the area under consideration to predict the air quality. Data mining techniques like exploratory data analysis were used to determine which features or pollution causing factors were essential for the intended prediction. Statistical components are calculated from the data available which helps in unifying the multimodal data. This eases the end user's data inquiry process from the data net. This part comprises of logic creation and aligning the related data fields with each other. Once the information model is formed, analytics and mining techniques are used on the same. The air quality predictions are generated.

ACKNOWLEDGEMENT

We place on record our deep sense of gratitude to our lionized Chairman **Dr.R.SHIVAKUMAR** for providing us with the requisite infrastructure throughout the course.

We take the opportunity to extend our hearty and sincere thanks to our Dean, **Dr.G.SELVAKUMAR, M.Tech, PhD.**, for maneuvering us into accomplishing the project.

We take the privilege to extend our hearty and sincere guidance to the Professor and Head of the Department, **Dr.N.KANNAN, M.Tech, PhD.**, for his suggestions, support and encouragement towards the completion of the project with perfection.

We convey our hearty and sincere thanks to our Project Coordinator **Dr.V.SELLAM, M.Tech, PhD.**, for her fortification. We express our hearty and sincere thanks to our guide **Dr.N.KANNAN, Head of the Department**, Computer Science and Engineering Department for her/his sustained encouragement, consecutive criticism and constant guidance throughout this project work.

Our thanks to the teaching and non-teaching staff of the Computer Science and Engineering Department of SRM Institute of Science and Technology, Ramapuram Campus, who provided necessary resources for our project.

PROJECT STUDENT NAMES

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	Abstract	iv
	Acknowledgement	v
	List of Tables	ix
	List of Figures	x
	List of Abbreviations	xii
1	INTRODUCTION	1
1.1	Overview	1
1.2	Objective	3
1.3	Problem Statement	3
1.4	Purpose	4
1.5	Organization of the report	5
2	LITERATURE SURVEY	6
2.1	Introduction	6
2.2	Existing System	6
2.3	Issues in Existing System	7
2.4	Summary of Literature Survey	8
3	SPECIFICATIONS	16
3.1	Introduction	16
3.1.1	Purpose	16

3.1.2	Project Scope	18
3.2	Overall Description	18
3.2.1	Project features	18
3.2.2	Operating Environment	19
3.2.3	Design Construction	23
3.2.4	Assumptions and Dependencies	26
3.3	External Interface Requirements	32
3.3.1	Software Interface	32
3.3.2	Hardware Interface	33
3.4	Summary	34
4	CONCEPTS AND DESIGN	35
4.1	Introduction	35
4.1.1	Concept	35
4.1.2	Existing System Defenceless	36
4.2	System Architecture	38
4.2.1	Proposed architecture	38
4.2.2	Description	39
4.3	Summary	40
5	METHODOLOGY	41
5.1	Introduction	41

5.2	Exploratory Data Analysis	42
5.2.1	Algorithm	43
5.3	Calculating Quantiles	43
5.3.1	Algorithm	45
5.4	LSTM Prediction Module	45
5.4.1	Algorithm	47
5.5	Summary	47
6	SYSTEM IMPLEMENTATION	48
6.1	Introduction	48
6.2	Sample Code	48
6.3	Output	59
6.4	Implementation Screenshots	64
6.5	Summary	70
7	CONCLUSION AND FUTURE WORK	72
7.1	Conclusion	72
7.2	Future Work	72
	REFERENCES	73

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
2.1	Summary of Literature Survey	8

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	Air Quality Prediction	3
1.2	Impacts of Poor Air Quality	4
3.1	Pollution Statistics	17
3.2	Anaconda	20
3.3	Jupyter Notebook	21
3.4	Working of Jupyter Notebook	22
3.5	Optimization Algorithms	24
3.6	Supervised Learning	25
3.7	Python Libraries	27
3.8	Pandas Features	28
3.9	Keras Functioning	28
3.10	Keras	29
3.11	Seaborn	31
3.12	Python and Visualisation	32
3.13	Features of Python	33
4.1	Backpropagation Through Time	36
4.2	RNN vs LSTM	37
4.3	System Architecture	38
5.1	Steps in EDA	43
5.2	Quartiles	44
5.3	LSTM Network and its Components	46
6.1	Histogram for temperature dataframe	60
6.2	Histogram for humidity dataframe	60
6.3	Histogram for particulate matter dataframe	61

6.4	Fitting curve for temperature predictions	62
6.5	Fitting curve for PM10 predictions	62
6.6	Fitting curve for Humidity predictions	63
6.7	Reading the dataset	64
6.8	Condense dataset to hourly format	64
6.9	Computing quantiles	65
6.10	Training for temperature attribute	65
6.11	Train and Test RMSE values for temperature	66
6.12	Generating test predictions for Temperature	66
6.13	Final plot for temperature	67
6.14	Training for PM10 attribute	67
6.15	Train and Test RMSE values for PM10	68
6.16	Generating test predictions for PM10	68
6.17	Final plot for PM10	68
6.18	Training for Humidity Attribute	69
6.19	Train and Test RMSE for Humidity	69
6.20	Generating Test Predictions for Humidity	70
6.21	Final Plot for Humidity	70

LIST OF ABBREVIATIONS

S.No.	ABBREVIATION	EXPANSION
1	PM	Particulate Matter
2	LSTM	Long Short Term Memory
3	MLP	Multilayer Perceptrons
4	RNN	Recurrent Neural Network
5	ANN	Artificial Neural Network
6	BPTT	Backpropagation Through Time
7	UI	User Interface
8	ML	Machine Learning
9	EDA	Exploratory Data Analysis

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The awareness and concern for pollution levels has gone up significantly among the general public and authorities. It has led to a new focus from all communication channels like television, print media etc to provide all relevant data points regarding pollutant levels. It has also made it mandatory for research professionals to widen their search database to span longer periods of time and cover wider geographical area to make their inferences stronger and more relevant for extremely harmful pollutants. The concentration of pollutants in the air determines the air quality. Higher the concentration of these toxins lower is the air quality and vice versa. There are many external factors which in turn impact and determine the levels of these toxins.

Pollution largely influences both the society and its economy, and the consequences of this are largely unfocussed and does not get its due importance. Apart from causing serious health issues, the high pollution levels has its toll on the world economy wherein unimaginable financial losses are incurred. Technological advancement in the field of pollution control and monitoring has been a big enabler for governments across the world to take corrective actions over a long sustained periods of time. Pollutants like PM_{2.5}, PM₁₀ and ozone have impacted many countries, including advanced countries like US and large developing nations like India and China. These high levels of pollutants in the air has led to significant global climatic changes.

The technological and scientific advancements intended to improve the quality of life for humans comes with a lot of consequences. Coal fired power plants, vehicular pollution caused by the combustion engines in automobiles and few natural causes like wildfires and dust storms contributes to the high levels of aerosols in the atmosphere. Presence of aerosol in the atmosphere has led to significant and unpredictable

variations in the climatic conditions around the world. These aerosols are known for the changes they inflict on the scattering of radiations by the earth, which causes global warming.

The proposed system aims to create an air quality prediction model which works over three phases. The first is using exploratory data analysis to find the characteristics from the data set that are essential for predicting the air quality. This is followed by calculation of quantiles which basically helps in dividing the available data into equal sized adjacent subgroups. The next phase is where the prediction is done. This is done using Long Short-Term Memory network. It helps in predicting the next set of values in a sequence by learning from a series of observations from the past. It is also observed that the model gets trained with minimal error since LSTM is a type of recurrent neural network, which means that the error was reduced successively while training the prediction system.

Air pollution monitoring and controlling falls within the ambit of a predictive data net. It involves churning of large datasets and making logical inferences. Machine learning as a branch of study is ideally suited to handle large database and providing the necessary leads which further helps in making appropriate inferences. In a predictive model the inferences are made through repeated observations from the dataset. Data looked at from different perspectives can provide varied conclusions. Therefore, we can say that machine learning as a platform thrives on the datasets that are available.

The combined need of fitting into a predictive model and the suitability of machine learning in aiding these predictions makes it the best suited approach for prediction and control of air pollution. The machine learning algorithms, like humans, learn through an iterative process of getting trained over a particular set of historical data. This past air quality data coupled with temporal distribution enables the proposed prediction model to arrive at desirable results.

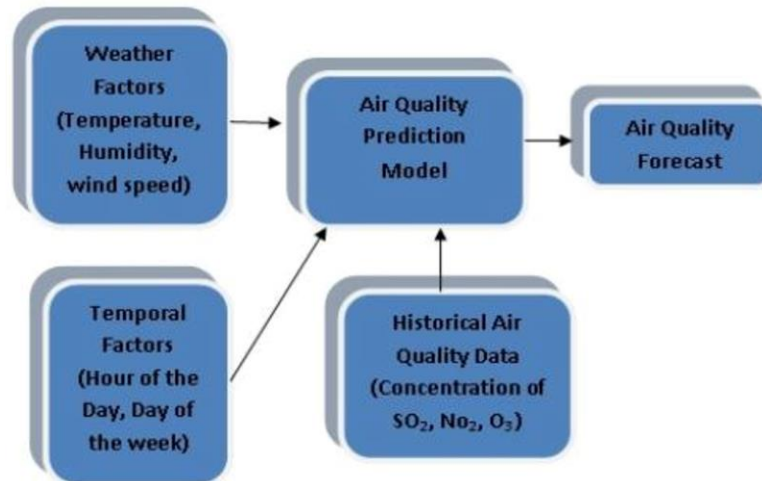


Fig 1.1 Air quality prediction

1.2 OBJECTIVE

The prime objective of the model proposed here is to efficiently and effectively predict pollution levels with minimal error. This is done based on historical pollution information. The drastic increase in pollution levels globally has led to severe consequences. Weather forecast is one of the most commonly available piece of information which is of great importance in determining air quality. This information coupled with temporal factors like date and hour and also the air pollutant information, gives us a prediction model which can learn from this historic data point that we provided and obtain the future values.

1.3 PROBLEM STATEMENT

In recent years, the global air pollution levels have risen drastically. This has been mainly due to ozone and particulate matter levels. Air pollutants have always been significant contributors to climate change. The excessive release of unwanted pollutants into the air is harmful to all living things on the planet.

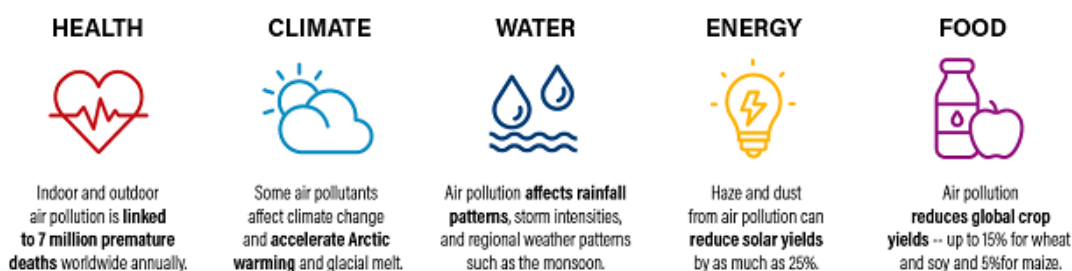


Fig 1.2 Impacts of poor air quality

1.4 PURPOSE

Rapid urbanisation has led to a significant depletion in air quality which has grave impingement on various facets of life such as health, nature, etc. As a result of this governments all over the world have started acting and proposing plans for improving the air quality. The basis for these decisions and action plans is the air quality prediction reports generated for the regions under consideration. Therefore, it is extremely essential to have a fool proof prediction methodology in order to ensure that effective measures are taken.

This model is trained on an air quality dataset from China but the same can be done for other countries also. China for example has released policies which aim at reducing PM10 levels by 18% in 2020 and the National Clean Air Programme (NCAP) proposed by the Ministry of Environment, Forest and Climate Change in India aims at a 20% to 30% reduction in PM10 levels by 2024. Air Quality prediction models like the one proposed here will facilitate the respective governments in keeping a proper track of pollution levels and achieving their desired pollutant levels with greater ease.

1.5 ORGANISATION OF REPORT

This project report deals with proposing a model for predicting air quality to provide useful insights and inferences which may go into the development of preventive measures for the future. This report is organized into seven chapters as follows:

Chapter -1

This chapter deals with introduction, system overview and states the objective for this project.

Chapter -2

The literature survey and the existing system along with its disadvantages are discussed in detail in Chapter 2.

Chapter -3

This chapter analyses the scope of the project and provides a high-level overview of the project coupled with all system requirement specifications.

Chapter -4

This chapter deals with the discussion of the architecture of the proposed system and evaluates the proposed system against the existing system.

Chapter -5

This chapter explains the methodology of the project along with a detailed description of the various modules and the algorithms associated with it.

Chapter -6

This explains in detail the system implementation, with code and screenshots.

Chapter -7

This chapter gives out the conclusion along with future enhancement.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This chapter provides an overview about the prior research that has been done in prediction algorithms and air quality prediction as a field. It also describes the issues in the existing systems which in turn define the purpose behind the newly proposed system. A literature survey is one of the most essential components of a dissertation as it sets the context of the entire work. This gives a clear idea about the short falls and perks of existing methods which can be used in the development of a better air quality prediction system which is more reliable.

An appreciation of previous work done in this domain provides a direction for us to proceed with the proposed system. The literature survey helps in justifying the model in this thesis and amplifies its importance to a great extent. It is a critical discussion that gives a summary of work which is of either general or specific relevance to the area of interest. This literature review provides insights into the

2.2 EXISTING SYSTEM

In one of the existing systems, numerical dataset was used like in all other systems. This undergoes a certain amount of pre-processing which is essential to obtain cleaned data for further processing. For air quality prediction the dataset is always spatio-temporal, i.e. it is spread across a certain timeline and over varied geographical zones. In this system the dataset they arrived at after the pre-processing was used to make a calendar map schema. They analysed attributes like PM2.5, Sulphur dioxide, carbon monoxide and nitrogen dioxide to do their predictions.

Another existing prediction system uses the hierarchical bayesian space time algorithm for predictions. This algorithm is used for working with itemsets and spotting those which are similar. This process uses hashing. Hashing helps in looking for the similar itemsets. It helps us bridge the gap between existing slow searching algorithms. This algorithm being faster reduces the time taken to execute the model and find the existing patterns and also the memory occupied by the system.

Hashing was used in this proposed system to speed up the memory access process as far as possible. The data underwent pre-processing followed by application of the above-mentioned hashing algorithm. This was then followed by concatenating the schema, with its patterns and the hash address. This method proved to be ineffective due to the high frequency of collisions that occurred in hashing. They generated two item sets for every partition.

The disadvantages of these existing approaches include their high computational complexity, less prediction efficiency, and the lack of dynamic behavior. These disadvantages led to the proposal of a new method for air quality prediction which is explained in the following sections. The identification of this spatio-temporal pattern is what goes on to help in the prediction. Models like Bayesian models which were used in some of the existing systems posed extremely high computational complexity. These models also have certain complex algebraic demands which discourage its usage in such applications.

2.3 ISSUES IN EXISTING SYSTEM

Prediction problems can be solved with three different categories of networks namely Multilayer Perceptrons (MLP), Recursive Neural Network (RNN), or Convolutional Neural Network. Most of the existing systems have used convolutional or multilayer perceptron networks. These networks are more suitable for classification and regression problems rather than time series prediction problems. It is generally observed that for spatio-temporal data, RNN's are more effective.

Models like Bayesian models which were used in some of the existing systems posed extremely high computational complexity. These models also have certain complex algebraic demands which discourage its usage in such applications. Cloud based approaches for air quality predictions bring with them the technical issues and connectivity problems. In methods like mapreduce there are disadvantages such as high latency and it is difficult to use for generating statistical inferences.

Many models have the issue of high computational cost which reduces the efficiency of the prediction net. Another common issue in the existing systems is the less accuracy of the results they generate and the fact that they don't correspond well with dynamically changing data which in the case of air quality prediction is quite a common thing that might be faced.

2.4 SUMMARY OF LITERATURE SURVEY

Table 2.1 Summary of Literature Survey

S.No	Title of the Paper	Authors	Publication	Concept	Advantages	Drawbacks
1.	Air Quality Index Prediction with Meteorological Data Using Feature Based Weighted Xgboost	Nandigala VenkatAnurag, Yagnavalky Burraboina, S.Sharanya	International Journal of Recent Technology and Engineering (IJRTE), Volume-8/ May 2019	This paper is based on the concept of using XGBoost model on meteorological data for predicting the air quality in the place of choice.	XGBoost spans shallow decision trees. These are interpretable trees which contribute to the ease of prediction.	XGBoost model has the disadvantage of causing overfitting in the absence of properly tuned hyperparameters. XGB training is a

						relatively complex and time consuming process.
2.	Prediction of Air Pollution Through Machine Learning Approaches on the Cloud	Ziyue Guan, Richard O. Sinnott	IEEE Conference on Big Data Computing Application and Technologies, 2018.	Air pollution data was collected from varied web based resources and linear regression and ANN was used to predict the levels of pollution	ANN has the advantage of a fairly good performance . It is also known for its ability to describe all kinds of relationship in the dataset, whether linear or non-linear.	ANN and linear models have drawbacks in predicting high PM2.5 values with accuracy and it works well only with certain specific data sets.
3.	Big Data Platform for Air Quality Analysis and Prediction	Yue Shan Chang, Kuan-Ming Lin, Yi-Ting Tsai	IEEE communication surveys & tutorials, vol. 16, no. 4, fourth quarter, 2018.	The system proposed here used Extract Transform and Load over and above the cloud computing	The advantages of this system include the fact that ETL proposes	ETL tools do not operate effectively in near real-time scenarios. These tools are also IT centric and

				platform. In this system cloud computing was used to provide browser support for the prediction model. They created API's for the same and used REST services.	reliable rules to be utilised for the data extraction and processing stages.	their accessibility is limited for analysts.
4.	Air Quality Monitoring In Urban Areas Using In-Situ And Satellite Data Within Era-Planet Project	Andrii Shelestov, Andrii Kolotii, Mykola Lavreniuk, Kyrylo Medyanovskiy	IEEE International Conference on Service Operations And Logistics, And Informatics, 2018	This paper proposed the use of satellite products which are available readily for on ground observations in the city of kyviv. They planned to develop air quality monitoring service for	Predictions based on satellite analysis are of great accuracy and show some extent of reliability also.	The major disadvantage was that this method was not cost effective and cannot be applied to regions which do not come under the coverage area of the satellites. Also it was found to be suitable

				urban areas based on remote sensing data and network of air quality sensors that can be used for real time AQ monitoring.		only in previously tested environments.
5.	A deep learning model for air quality prediction in smart cities	İbrahim KÖK, Mehmet Ulvi ŞİMŞEK, Suat ÖZDEMİR	IEEE International Conference on Big Data (BIGDATA)/ 2017	They proposed a novel model based on Long Short Term Memory (LSTM) networks to predict future values of air quality in a smart city. The LSTM network was trained using the particle filtering algorithm.	LSTM has far less errors for general as well as high PM2.5 values. LSTM also resolves vanishing and exploding gradient problems which were noticed in other networks.	The training process of This system is very time consuming and hence there is a direct demand for relatively large scale resources and when compared to non-sequential models, this model requires gathering increased amounts of historic data.

6.	Research Of Time Series Air Quality Data Based On Exploratory Data Analysis And Representati on	Changhui YU	IEEE Sensors Journal, Vol. 16, No. 1, January 1, 2016	The system proposed in this paper suggested the use of Exploratory Data Analysis for the initial phases followed by statistical analysis techniques to arrive at a visualisation.	EDA is extremely essential while developing a machine learning model. This ensures that the results obtained from the model are accurate.	While EDA ensured that the data was refined in order to provide better results, the statistical inference methods used here did not provide the best results.
7.	Air Quality Simulations using Big Data Programmin g Models	Haripriya Ayyalasom ayajula, Edgar Gabriel	October, 2016.	They compared the MapReduce and Spark models and their respective efficiencies in predicting the ambient air quality conditions. They conducted their experiment by	Since it was a comparative analysis of two methods, the advantages don't hold as much weight as compared to the relative efficiencies. In this analysis they showed	In hindsight we can also state the fact that though in a comparative analysis, one of these methods seemed better than the other, they still aren't as accurate as the LSTM method we propose in

				computing the average pollutant levels for an 8 hour time period using both the models that were being compared.	that spark was at least 30% better than MapReduce	our system here.
8.	A Comprehensive Evaluation of Air Pollution Prediction Improvement by a Machine Learning Method	Xia Xi, Zhao Wei .et. Al	IEEE International Conference on Service Operations And Logistics, And Informatics (SOLI)/ 2015	They utilized the forecast on pollution, weather, chemical component from WRF-Chem model as input features to design a comprehensive evaluation framework to compare various algorithms and to improve the prediction performance.	The output of WRF-Chem model contain lots of information which can be used as input to build statistical model or machine learning algorithms.	The comprehensive analysis of five different machine learning algorithms showed that these methods individually do not suffice for an efficient pollution prediction method.

9.	Comparative Study of Association Rule Mining and MiSTIC in Extracting Spatio-Temporal Disease Occurrences Patterns	Vipul Raheja, K. S. Rajan	IEEE International Conference on Big Data, June 2012.	The use of MiSTIC algorithm to address the problems encountered in the extraction of spatio temporal patterns which was observed in the traditional systems which used traditional algorithms for the same was proposed.	This paper makes a case for the use of MiSTIC algorithm to address these issues, compare the use of traditional association rule mining in context of Salmonellosis disease management, and share new insights.	The mining rules that were produced by the model proposed here were not entirely of interest due to their inability to mine the pervasiveness of the diseases in an effective manner. These drawbacks result as these methods ignore the inherent spatiotemporal dependency in such data.
10.	Prediction of Ambient Air Quality Based on Neural Network Technique	Mahanijah Md Kamal, Rozita Jailani and Ruhizan Liza	IEEE Student Conference on Research and Development, 2012	In this paper they utilized Artificial Neural Network (ANN) model with Back	BPNN is a fast and easy model, which offers flexibility since it	Its high dependence on input data and sensitivity to noisy data are some of the key disadvantages

		Ahmad Shauri		Propagation Neural Network (BPNN) for predicting the ambient air quality.	does not require prior knowledge and has no additional parameters to tune other than the input.	of the methods used herein.
--	--	--------------	--	---	---	-----------------------------

CHAPTER 3

SPECIFICATIONS

3.1 INTRODUCTION

In this chapter we discuss the basic overview of the system proposed in this thesis and we further explore the software and hardware requirements and the various constraints to keep in mind while developing the net. The complete details of the environment in which this system functions and all the assumptions that were made while developing it are also mentioned in the sections that follow. The purpose of a system requirement specification is essential to describe the functionalities to be fulfilled by the system and to provide a clear image of what to implement and using what software to use, to the developer.

3.1.1 Purpose

The Air Quality Prediction system aims at creating an efficient air quality speculation model which couples knowledge discovery in databases and Back Propagation Through Time algorithm. The technique is used to evaluate and analyse the factors that impact the study. It also aids the process of excluding or ignoring those factors which are either irrelevant or do not contribute much to the end result. By doing so the model enhances the accuracy levels of the prediction. It also reduces the number the computations that are required to be carried out arrive the desired efficiency. The quality of prediction arrived at by using this technique is high on accuracy, and can provide solutions which are practically implementable. Such predictions are so practical that it makes it easier for authorities like the environmental department to rely on them and arrive at actionable points.

Air pollution monitoring and controlling falls within the ambit of a predictive data net. It involves churning of large datasets and making logical inferences. Machine learning as a branch of study is ideally suited to handle large database and providing the necessary leads which further helps in making appropriate inferences. In a predictive model the inferences are made through repeated observations from the dataset. Data looked at from different perspectives can provide varied conclusions. Therefore, we can say that machine learning as a platform thrives on the datasets that are available.

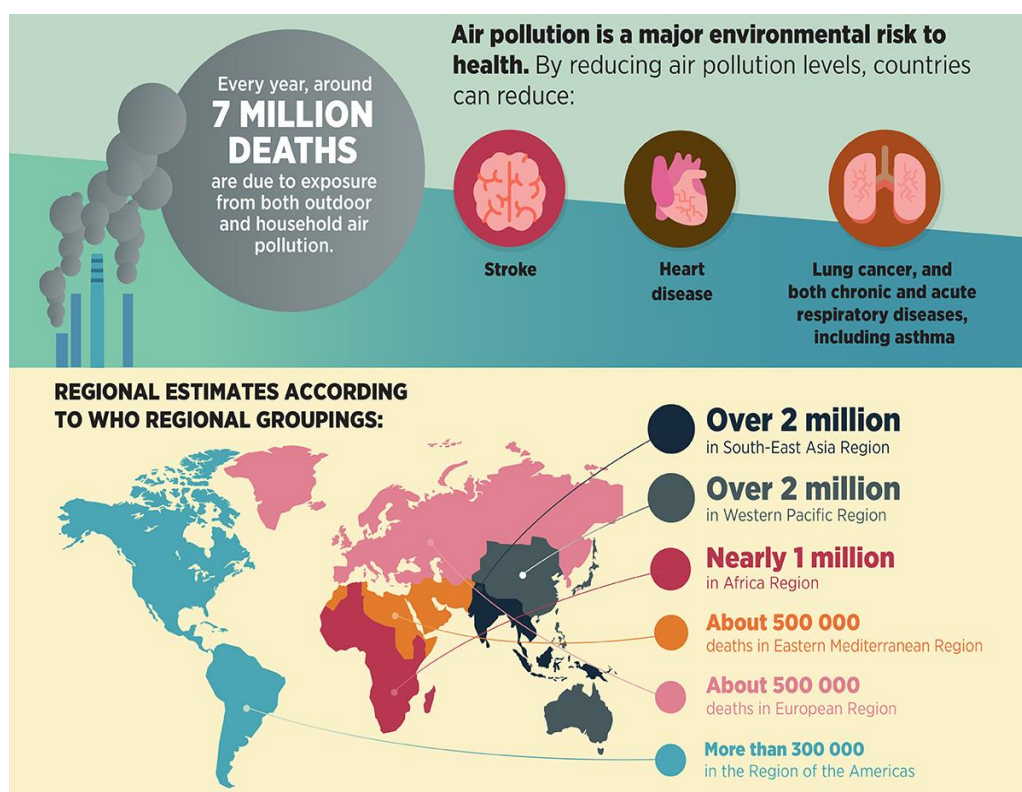


Fig 3.1 Pollution statistics

The network used in this system is Long Short-Term Memory network. This is a type of Recurrent Neural Network and its capable of learning order dependence. This is one of the extremely efficient networks used in time series prediction problems. Unlike traditional feed forward networks, RNN has a set of internal states. These states are responsible for storing information about past inputs for a specific amount of time which is decided depending on the input and the weights associated. LSTM has trainable parameters, noise

resistance and as mentioned above they can also store information for an arbitrary time period.

3.1.2 Project Scope

The horizon of this dissertation extends to the creation of an efficient air quality prediction model using the dataset obtained from atmosphere monitoring stations in Shijiazhuang which has an elaborate set of data about the various pollutants detected in the air in that region. This system aims to generate predictions of pollutant levels based on historically available data which is used in training the net. This model holds the capability to deal with dynamic data with ease and without the need for repeated training.

Data mining techniques like exploratory data analysis were used to determine which features or pollution causing factors were essential for the intended prediction. Statistical components are calculated from the data available which helps in unifying the . This eases the end user's data inquiry process from the data net. This part comprises of logic creation and aligning the related data fields with each other. Once the information model is formed, analytics and mining techniques are used on the same. The prediction model proposed herein is bound to prove to be an authentic way for predictions by environmental departments.

3.2 OVERALL DESCRIPTION

3.2.1 Project Features

The Air Quality Prediction system aims at creating an efficient air quality speculation net. The technique used here to evaluate and analyse the factors that impact the study involves various data mining algorithms. It also aids the process of excluding or ignoring those factors which are either

irrelevant or do not contribute much to the result. By doing so the model enhances the accuracy levels of the prediction. It also reduces the number the computations that are required to be carried out arrive the desired efficiency.

This project proposes a prediction model which can be used to generate the prediction of pollutant levels at a point in time as requested by the user. This model can provide information about the pollutant levels of O₃, CO, NO, NO₂, SO₂, VOC, PM_{2.5}, PM₄, PM₁₀, TSP, Temperature, humidity etc. This is done over three broadly classifies steps namely exploratory data analysis, calculation of quantiles and training and testing of the prediction net.

While the first two phases deal primarily with data cleaning and data mining, the last phase deals with creation of the actual LSTM net. The LSTM network used is expanded as long short term memory and this gets trained with back propagation algorithm. This is a type of recurrent neural networks which consist of loops which unwind and perform the training over a sequence of loss reducing steps. The connection between the nodes leads to the formation of a directed graph along a certain temporal series.

3.2.2 Operating Environment

The operating environment for this project is anaconda and Jupyter Notebook which comes along with it. Anaconda is a platform for Artificial Intelligence and machine learning developers and enthusiasts to develop end to end projects on their own systems. It gives them the leeway to experiment with more and more possibilities in this ever-expanding field of study. It is beneficial even to organisations as it helps them scale up from single servers to multiple servers. Package management is extremely simplified in Anaconda which in turn assists in easier and more convenient deployment. The divers tool sets that come along with Anaconda help in effective data gathering for the ML systems. It provides a UI support that ensures facile deployment of

projects. It's a distribution mainly for languages like Python and R. The commands for package installations in anaconda is extremely simple. The package management system that we get with Anaconda is called Conda and for the UI we have Anaconda Navigator. Python as a language has multiple versions which have varied developments in them. To support multiple such python versions Anaconda comes in extremely handy. Also, the libraries that anaconda provides are highly optimised and assist in extremely efficient development processes.

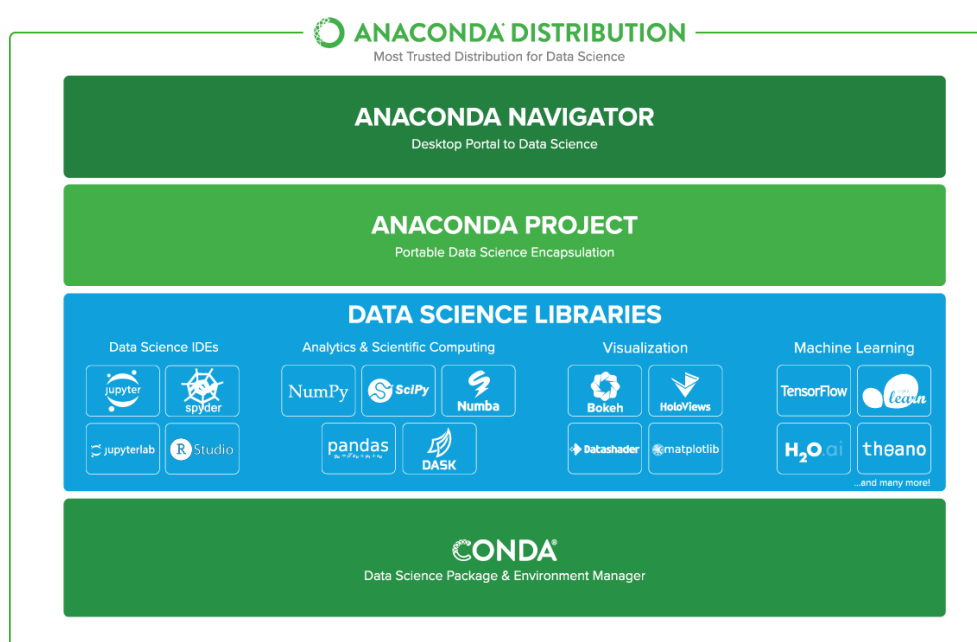


Fig 3.2 Anaconda

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Jupyter Notebook is extremely useful in pre-processing of data. The cleansing of the dataset can be done with utmost efficiency using jupyter notebook. It provides interactive environments where we can create notebooks which are nothing but our code documents. These notebooks are saved in our system under the jupyter notebook directory with a .ipynb extension. The notebooks are also called kernels and they need to be refreshed periodically in order to avoid timeout issues. Jupyter notebook is web based which serves as another key selling point for the application. After

cleaning, the data can also be subjected to a wide range of transformations using the various dependant libraries that are provided with it. Jupyter notebook helps us to code in python in a distributed format, i.e. we can write the code line by line and execute it at every step before proceeding to the next line of code. This is done by compiling the specific line of code using a remote compiler available at the jupyter company's facilities. This application can also be used along with other things like raspberry pi to provide unforeseen results. This application can also be used for obtaining valuable visualisations of data which lead to more effective inferences. The Pandas library that can be installed using conda install and imported in the code is what helps us mainly in generating these visualisations. There are few other libraries like matplotlib and seaborn also which assist in the same.



Fig 3.3 Jupyter notebook

A Jupyter Notebook consists of various UI components which are extremely useful while coding. The first visible page when we open the jupyter notebook is called the dashboard. From this we can choose which notebook, existing or new, do we want to edit currently. When the notebook is opened we can see multiple cells. These are the basic distributed code components which can be run individually to obtain outputs. These outputs will be displayed right below the cell that generated it. The primary architecture of jupyter notebook uses kernel and messaging. This facility is what helps us to run different codes in a wide range of languages. Some of these are Python, Scala, R, Go etc. But in this project we focus only on the Python utility of Jupyter Notebook.

There are three different types of cells which we can create in jupyter notebook. The first type is a code cell which by the name, suggests that it has live code which can be executed. The next type is Markdown cells. In these we can add text and also equations. For adding equations, LaTeX is used. These types of cells generally help in providing explanations about the code cells and their content. The last type of cells are called the raw cells. These are used to hold unformatted text snippets. Cells in jupyter notebook can undergo several operations.

We can insert cells, merge them, cut, copy and paste cells and do a lot more. Each cell is associated with a line number which is assigned to it depending on the sequence of execution. It is also possible to have HTML code snippets embedded in the markdown cells. Jupyter notebook also comes with a couple of in-built commands. These add certain interactive features to the notebooks. One of the most useful commands provided by jupyter is the %pdb command. This lets us debug the code and see what's happening at certain points in the code.

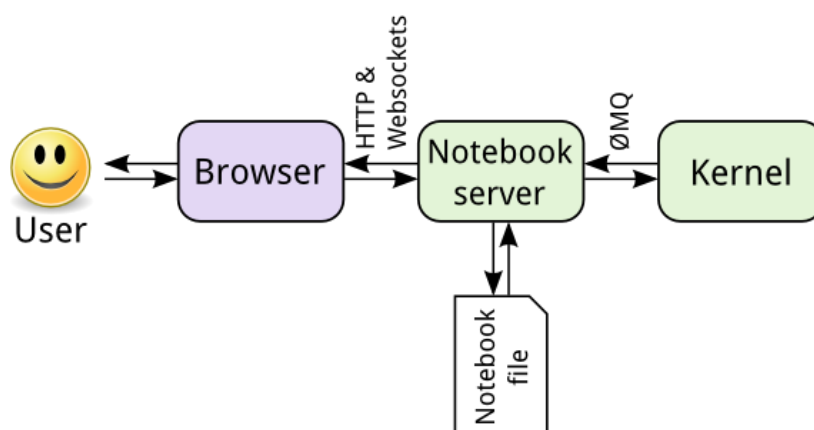


Fig 3.4 Working of Jupyter Notebook

There are various database options that can be used when we deal with real time data which needs to be handled as and when it arrives. In this

dissertation, however, we have used a csv file as the dataset with all the pollutant data. This dataset consists of 9230 records of Bandung region in China. We have this data in csv format i.e. a sequence of comma separated values. It is one of the simplest formats for storing tabular data. These records that we have consist of multiple values like the pollutant levels for SO₂, NO₂, O₃ etc. Jupyter consists of a module by the name csv which is the one that is used to read data from this csv file. This module needs to be imported at the beginning of the code in order to be used.

3.2.3 Design Construction

Machine Learning is as a field of study is an amalgamation of statistical analysis techniques, computational mathematics and computer science algorithms. Using the power of statistics and the methods associated with it, we can derive inferences which are backed by appropriate data. Machine learning consists of models which are created using the mathematical aspects and the computer algorithms implement these models.

Post the creation and implementation of these models comes an important step. This step is of optimizing the model and its performance. Optimization involves tweaking the hyperparameters of the model in order to attain the most optimum output. ML comprises of a variety of optimisation techniques. The optimisation technique used in this project is called Adam. This technique achieves optimization by adaptively changing the learning rate of the LSTM. Whenever the training loss becomes constant and arrives at a plateau, this optimization technique reduces the learning rate by a little to avoid going around the same loss through the remaining epochs.

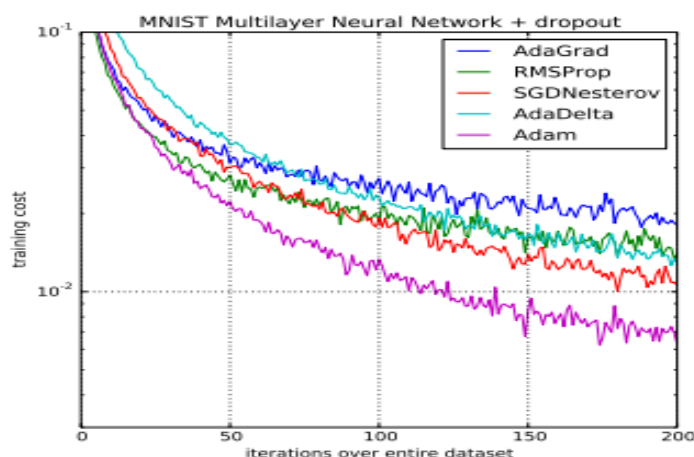


Fig 3.5 Optimization algorithms

In recent years we've observed an exponential rise in the amount of data available and this has in turn increased the requirement for effective systems that can deal with this data and draw accurate inferences and predictions. Deep learning algorithms are used for this purpose. They are more accurate than the traditional algorithms and generate precise predictions.

These algorithms observe the patterns in the dataset over time and get trained over this historical data. Using this the algorithms execute the classifications. Once the model has undergone training it uses this learnt behaviour to predict the outcome for every newly fed input also. Based on the final accuracy, one can optimize their models using various standardized approaches. Machine Learning Algorithms can be classified into 3 types as follows:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

In this project we have trained the LSTM model using Back propagation algorithm which is a type of supervised learning algorithm. In most supervised learning models, we generally aim to create a predictor function or a hypothesis function which is extremely accurate. In this process of supervised learning the system is supervised by an external supervisor who trains the model to learn

the patterns in the dataset by training it on the train dataset. This is done in loops till the time the most effective and accurate result is not obtained and till the time the prediction does not reach the highest possible accuracy. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients.

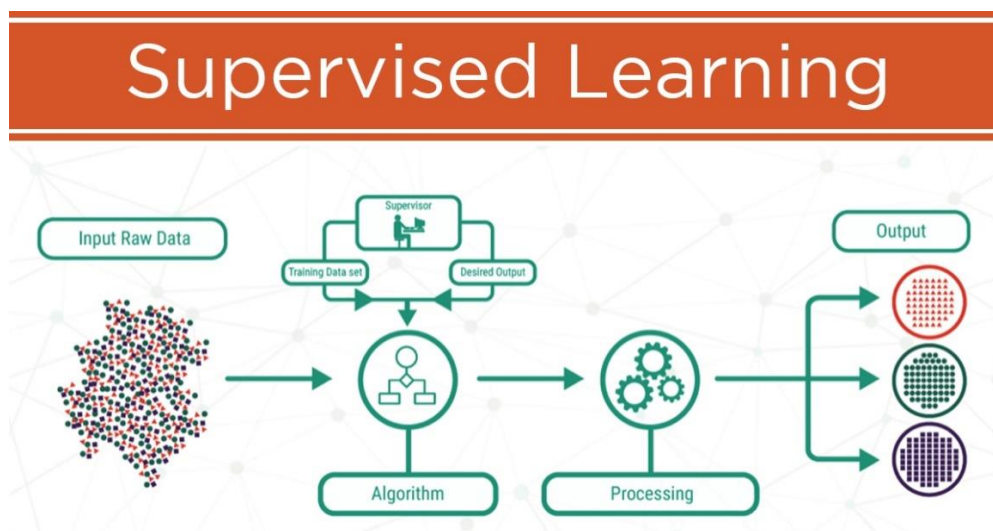


Fig 3.6 Supervised learning

Under supervised algorithms there are two major subcategories are:

1. Regression algorithms in which the predicted attribute lies on a continuous spectrum. These algorithms provide responses for the quantity related questions or numeric questions.
2. Classification algorithms are those where we get a binary prediction in the form of yes or no.

Steps Involved in design construction are:

1. The first step in designing a prediction system is data gathering. We gather the data from the air quality monitoring stations of Shijiazhuang and create a csv file which will serve as our dataset.
2. This data is then labelled and split into training and test dataset. This dataset contains spatio-temporal data.
3. The data is sorted and pre-processed. The pollutant levels are sorted into hourly data, which helps us to easily arrive at inferences.

4. The pre-processed data is then sent for Exploratory Data Analysis.
5. EDA is done in three phases in which the first is feature selection, in this phase the characteristics from the dataset which are essential for our analysis are determined.
6. Once these essential features are identified we move to the next phase of correlation analysis. Correlation analysis is like a preliminary technique used to identify the relationships, linear or non-linear, among the features selected in the previous step.
7. We then perform cluster analysis which gives us a clear idea about the patterns in air pollutant levels which in turn will help us in predicting the air quality at a future point in time.
8. Now the data analysis and processing are done and we proceed to create the training set and determine the input features which will be fed into this system.
9. Establish the algorithm which is suitable to obtain the accurate predictions. In this case that algorithm was identified as Back propagation through time.
10. Determine the model which is ideally suited for the experiment at hand which in this case is long short-term memory network.
11. Proceed to train the model with this algorithm and determine the predictions and the loss in prediction. Set the hyperparameters to a particular initial value.
12. Execute this algorithm on the train set and evaluate the loss and the accuracy of the predictions.
13. If the accuracy is found to be insufficient, tune the model further by altering hyperparameters like learning rate, number of hidden layers, etc.
14. If the model is accurate proceed by plotting the fitting curve for the feature under consideration.

3.2.4 Assumptions and Dependencies

The proposed system is created in Python and has a set of dependant libraries whose installation is crucial for the code to function. Some of those

libraries are pandas, keras, datetime, scipy, numpy, seaborn, sklearn. These are described in detail in the following paragraphs.



Fig 3.7 Python libraries

Pandas is one of the most important libraries which facilitates effective data analysis. It's the most popular among all the libraries provided by python. Its predominantly used in wrangling data. It is an open source library by Wes McKinney.

This library helps in easy handling and operation on huge dictionaries and lists which we generate from the available dataset. But one of the key aspects of Pandas is the fact that it takes data only in the form of a specific data structure called dataframe. These dataframes have a shape which is determined by the number of rows and columns present in it. Generally, CSV files or SQL databases are the inputs to pandas library. It is useful due to its support for a wide spectrum of file types also. Pandas is essential for dealing with time series problems like the air quality prediction problem we have at hand here. Performance optimization of the model can be achieved upto a great extent by using the pandas modules and classes. Missing data interpolation that we do in this proposed system is also achieved through the use of Pandas library and its Interpolate() method.



Fig 3.8 Pandas features

Keras is an open-source neural-network library written in Python. Designed to enable fast experimentations with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

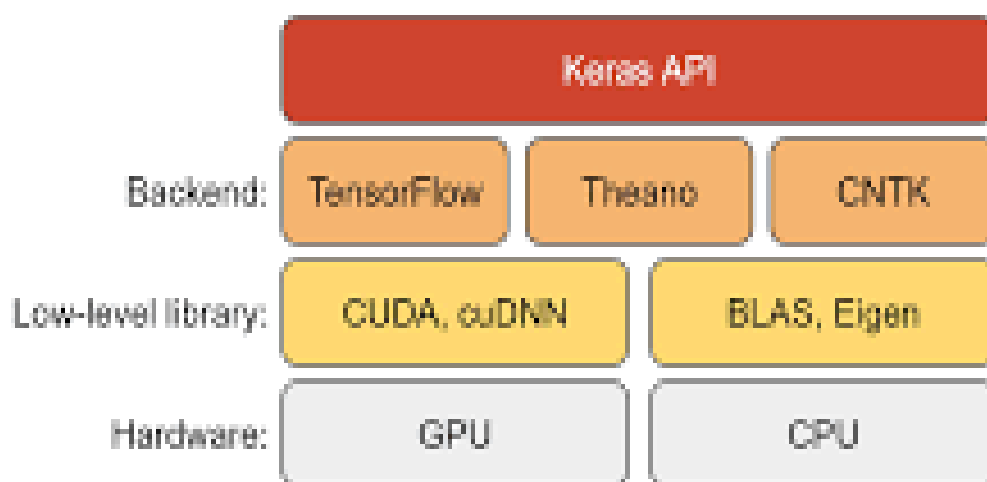


Fig 3.9 Keras Functioning

Keras is a model-level library, providing high-level building blocks for developing deep learning models. It does not handle low-level operations such as tensor products, convolutions and so on itself. Instead, it relies on a specialized, well optimized tensor manipulation library to do so, serving as the "backend engine" of Keras. Rather than picking one single tensor library and making the implementation of Keras tied to that library, Keras handles the problem in a modular way, and several different backend engines can be plugged seamlessly into Keras. Currently, Keras has three backend implementations available: the TensorFlow backend, the Theano backend, and the CNTK backend. In the proposed system, amongst these three, TensorFlow is used as the backend.

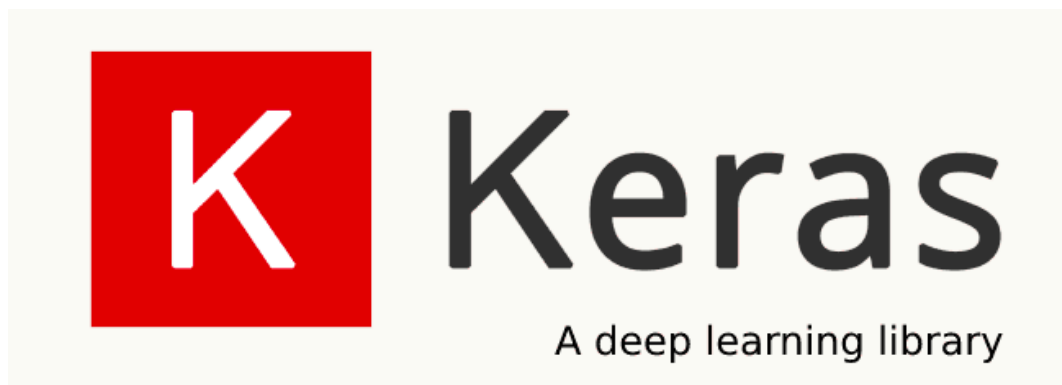


Fig 3.10 Keras

Unlike other languages, Python does not provide a datatype for storing date and time. This is available in the form of a module externally. To use this module in our system or code it is mandatory to import this module before using dates. This module is in built in the Python language unlike modules like pandas and NumPy which need to be installed separately and then later imported in the code. This module has multiple classes to perform various operations on temporal aspects of datasets. These varied functionalities are particularly useful in systems like ours where the distribution of data points is not only spatial but also temporal and the temporal aspects are extremely important.

The need for a free library in python for performing effective computations led to the creation of the SciPy library. This library has a wide range of functions for performing algebraic computations and for optimization activities. We can also perform image and signal processing with the help of certain specialised methods.

This library is a part of the stack of another library in Python called NumPy. It builds upon the array object of this parent library. The stack for NumPy also has many other libraries which come under its umbrella like Matplotlib, pandas etc. It is primarily responsible for providing scientific computational modules.

Python also provides a few other equally valuable and useful libraries like NumPy. This library is known for the assistance it provides for the use of humungous multi-dimensional arrays. It also provides methods for operating on these arrays and supports matrices. NumPy in Python serves as a replacement which fills the void of incapability created because we can't use MATLAB. Jim Hugunin is the one who initially formed this library. This library is known mainly for the numerical computation capabilities that it provides for operating on numerical dataset. Though NumPy provides similar functionality like that of MATLAB, MATLAB is a complete software in itself and hence has functionalities which exceed what NumPy can provide. This library helps users to quickly operate on really huge arrays with minimal effort.

Statistics and statistical representations play an important role in prediction systems like this. For portraying these stats we use a library called Seaborn which is provided by Python. This library helps us in doing data visualisations. Matplotlib is the parent library and seaborn is based on this. This library gives us the UI for creating impressive statistical graphics. This library also works in close coordination with Pandas library. There is a set of API's provided by this library. This API analyses the different interrelated variables. This library holds great value for developers because visualisation

of the data at hand is extremely crucial for deriving valid, data backed inferences.



Fig 3.11 Seaborn

Scikit-learn is a freely available library for machine learning enthusiasts. It plays a key role in most machine learning models. This provides modules which can be used for a majority of scientific operations. This library unlike the others that we saw previously, does not primarily provide support for data manipulation and cleaning but instead it helps in modelling the data. Scikit-learn provides some extremely valuable algorithms like the k- nearest neighbour or knn algorithm, random forest classifier algorithm etc. This sklearn library supports SciPy library also.

3.3 EXTERNAL INTERFACE REQUIREMENTS

3.3.1 Software Interface

Python and its libraries are the primary software requirements for this project and the python code is written in Anaconda using Jupyter notebook IDE. This language is open source and can be used freely by developers and ML enthusiasts over the world. The only requirement is to do some basic installations and then we're good to go. This language shows great potential due to its compatibility across various platforms, i.e. this can be installed in a wide range of operating systems like windows, mac etc. and it will work just as good on each of these operating systems. This language proves its effectiveness as a visualization tool time and again by providing libraries like seaborn, matplotlib etc.

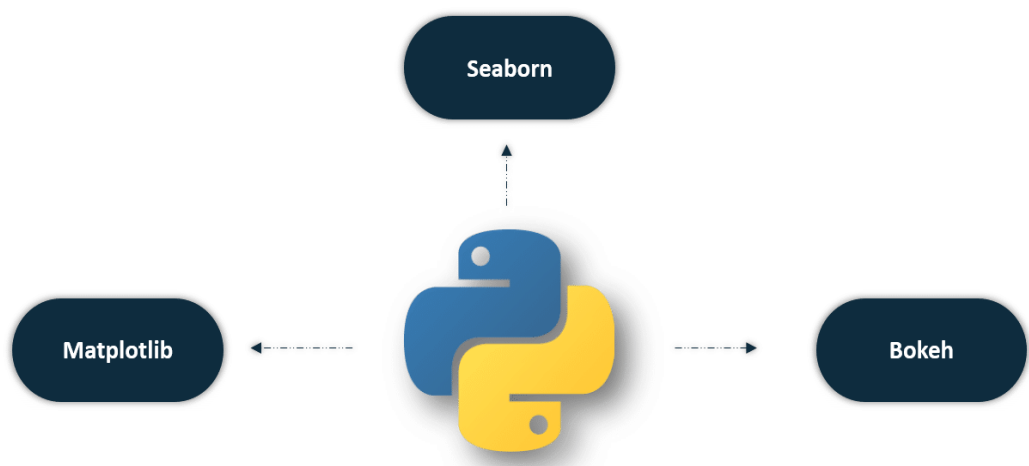


Fig 3.12 Python and visualization

Python as a language is extremely easy for developers to learn and utilize. Python is currently one of the most popularly used languages for machine learning. The fact that this is free and open source contributes significantly to its widespread usage. It's a high level language and it uses common words for programming and has minimal standard syntax which the

developer must adhere to. Python also allows a great deal of interoperability and can be used on multiple different operating systems.

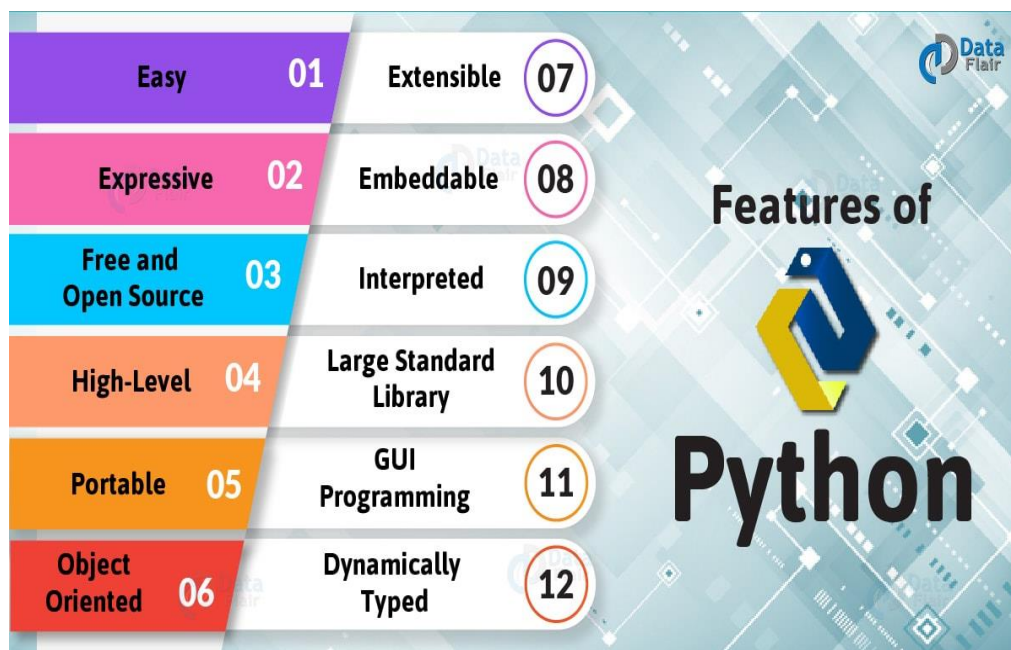


Fig 3.13 Features of Python

This is a dynamically typed language i.e. the python language interpreter automatically assigns a datatype to the variable that was created by the programmer. This typing is done at runtime and hence its called dynamic typing. This helps in reducing the number of lines of code that are written by the programmer. This makes the development environment extremely productive. While languages like C and C++ use either procedural programming technique or object-oriented paradigm, python enables the usage of both. This language helps us in developing computer vision, image processing, classification, regression, and various other artificial intelligence systems.

3.3.2 Hardware Interface

Effective hardware plays an essential role in the effectiveness of a machine learning system especially when the scale of the model is huge and

the amount of data involved is vast. The following are the hardware requirements for the proposed system.

<u>Hardware</u>	<u>Minimum Requirement</u>
Disk Space	32 GB or more, 10 GB or more
Processor	1.4 GHz and 64 bit
Memory	512 MB
Display	(800 × 600) Capable video adapter and monitor

3.4 SUMMARY

This chapter clearly outlined the system requirements for the proposed system. It stated that the language used for this air quality prediction system was python and the Jupyter notebook was the interface on which the programming was done. Supervised learning method was used for training the machine learning model. This chapter also stated the various libraries in python which are dependencies for this implementation. Some of those libraries were keras, pandas, NumPy, sklearn etc. Along with all these system specification details we also came across the design construction which was outlined clearly, giving a high level overview of the entire proposed system.

CHAPTER 4

CONCEPTS AND DESIGN

4.1 INTRODUCTION

This chapter focusses on providing a clear outline of the proposed system architecture. The architecture is described in detail and the conceptual base for the entire system is also outlined in the sections that follow. Based on the concept used in the proposed system we also highlight the reasons why the existing system must be replaced with the system proposed in this thesis.

4.1.1 Concept

This air quality prediction system uses, a set of factors like ambient pollutant levels of the area under consideration to predict the air quality. Data mining techniques like exploratory data analysis were used to determine which features or pollution causing factors were essential for the intended prediction. Statistical components are calculated from the data available which helps in unifying the multimodal data. This eases the end user's data inquiry process from the data net. This part comprises of logic creation and aligning the related data fields with each other. Once the information model is formed, analytics and mining techniques are used on the same. The air quality predictions are generated.

The network used in this system is Long Short-Term Memory network. This is a type of Recurrent Neural Network and its capable of learning order dependence. This is one of the extremely efficient networks used in time series prediction problems. Unlike traditional feed forward networks, RNN has a set of internal states. These states are responsible for storing information about past inputs for a specific amount of time which is decided depending on the

input and the weights associated. LSTM has trainable parameters, noise resistance and as mentioned above they can also store information for an arbitrary time period.

In this proposed system we used Backpropagation to train the LSTM net. BPTT algorithm uses chain rule like that in calculus in order to train the neural network to work efficiently. This chain rule means that everytime there is a single step progress in the forward direction along the LSTM network there will be one step in the reverse direction alongside the adjustment of the training parameters and weights. In other words, backpropagation aims to minimize the cost function by adjusting network's weights and biases.

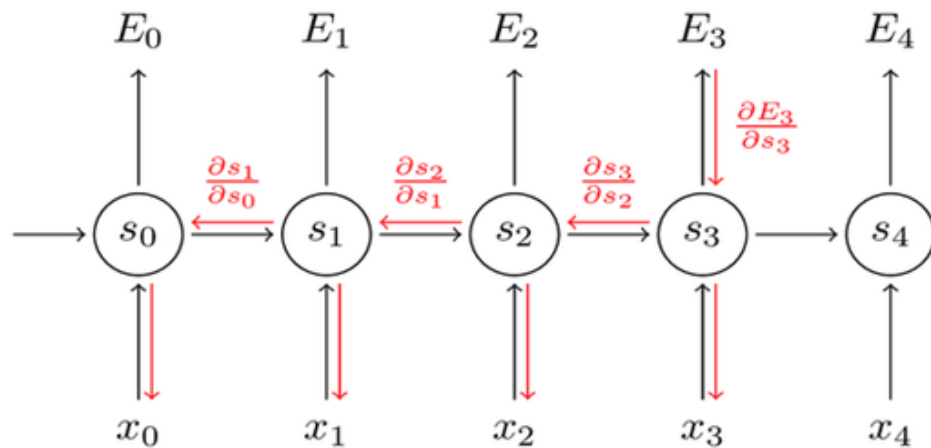


Fig 4.1 Backpropagation Through Time

4.1.2 Existing System Defenceless

It has been observed that LSTM's have solved two crucial problems which existed namely vanishing gradient problem and exploding gradient problem. The loss function in our network has a value that approaches zero when excessive layers are added to the network. This problem is solved in LSTM since it can use Backpropagation algorithm to train.

One of the key problems faced in most RNN networks is that of exploding gradients in which errors accumulate over time. This huge accumulated error results in high updates to the network when we update the weights in order to reduce the error. These large updates to the weights lead to instability in the prediction model. This reduces the capability of the model to learn and get trained. There is something called a forget gate which is not there in RNN but is present in LSTM. This forget gate never gets activated thereby ensuring that the gradient explosion doesn't occur at all. Therefore we can say that LSTM is effective in learning long range dependencies.

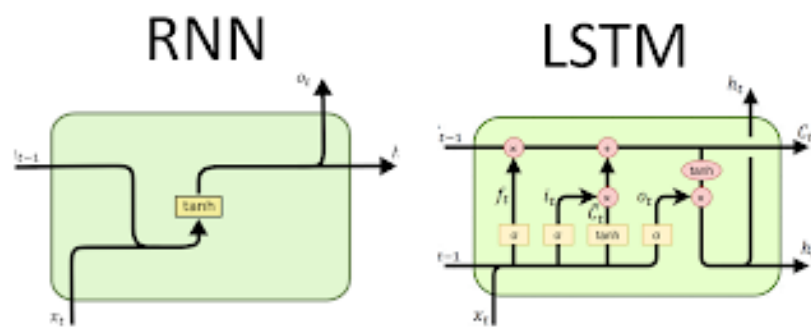


Fig 4.2 RNN vs LSTM

Many of the existing models have the issue of high computational cost which reduces the efficiency of the prediction net. Another common issue in the existing systems is the less accuracy of the results they generate and the fact that they don't correspond well with dynamically changing data which in the case of air quality prediction is quite a common thing that might be faced. The few existing systems which use RNN's also prove to be slightly inefficient as they encounter the vanishing and exploding gradient problems. Hence, due to all the above stated facts, the existing systems are rendered defenseless.

4.2 SYSTEM ARCHITECTURE

4.2.1 Proposed Architecture

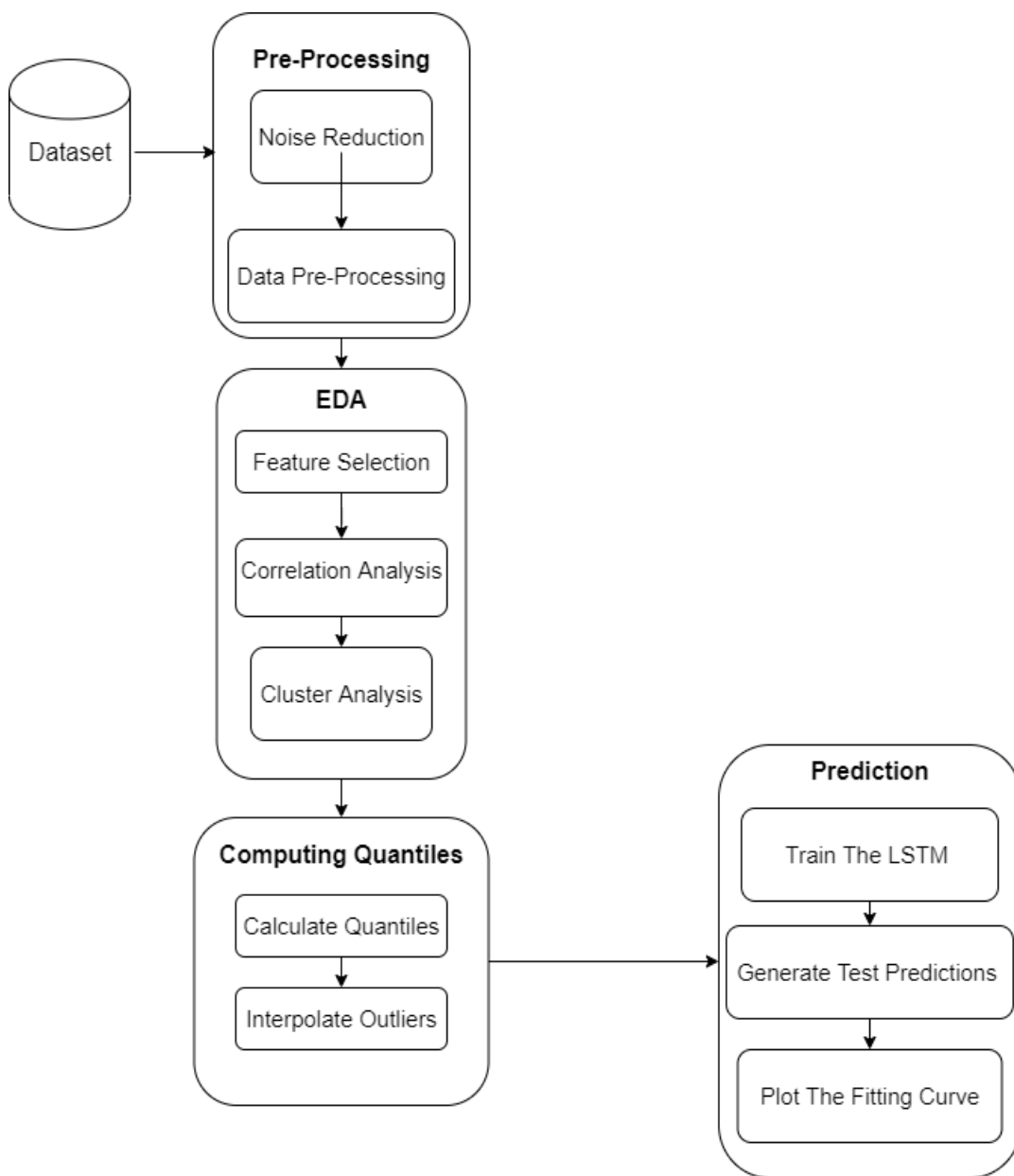


Fig 4.3 System Architecture

4.2.2 Description

1. The dataset with pollution statistics from the meteorological stations are taken as the input.
2. This data needs to be refined to us for our system, therefore it undergoes noise reduction and pre-processing.
3. The pre-processed data is then sent for Exploratory Data Analysis.
4. EDA is done in three phases in which the first is feature selection, in this phase the characteristics from the dataset which are essential for our analysis are determined.
5. Once these essential features are identified we move to the next phase of correlation analysis. Correlation analysis is like a preliminary technique used to identify the relationships, linear or non-linear, among the features selected in the previous step.
6. We then perform cluster analysis which gives us a clear idea about the patterns in air pollutant levels which in turn will help us in predicting the air quality at a future point in time.
7. After EDA, with the selected features, the data moves into the next step where we calculate quantiles. This enables us to split the distribution that we are using in a uniform manner.
8. The quantiles computed in the previous step are used to find the outliers in the dataset for each feature and these outliers are then interpolated.
9. We move on to predict the air quality which is done using LSTM network which is trained using the Backpropagation Through Time (BPTT) algorithm.
10. This model is now used to generate predictions on the test data following which we compare the test RMS error value with the Standard deviation computed before to see if the model is good or not.
11. Post this we plot the fitting curve for the feature under consideration which shows the original dataset, train prediction and the test prediction.

4.3 SUMMARY

This chapter clearly outlined the underlying principle and idea of the proposed system. It says that the network created in the proposed system was an LSTM which was trained with the help of Backpropagation. The system architecture is also shown in this chapter. This diagram distinctly outlines the four main phases of the system development process. These were, data pre-processing, exploratory data analysis, computing quantiles and the final LSTM prediction phase. Having a clear idea about the design and concepts for the proposed system, we can now proceed with the detailed description of the modules in the next chapter.

CHAPTER 5

METHODOLOGY

5.1 INTRODUCTION

The proposed air quality monitoring model which combines exploratory data analysis, LSTM network and backpropagation through time algorithm is based on the monitoring data of air pollution obtained from Shijiazhuang air quality monitoring stations. In this project we perform the air quality prediction over three steps or phases. The first is using exploratory data analysis to find the characteristics from our data set that are essential for predicting the air quality. This is followed by calculation of quantiles which basically helps in dividing the data we have into equal sized adjacent subgroups. The next phase is where we do the prediction. This is done using Long Short-Term Memory network. The LSTM helps us to predict the next set of values in a sequence by learning from a series of observations from the past.

The Air Quality Prediction system aims at creating an efficient air quality speculation model which couples knowledge discovery in databases and Back Propagation Through Time algorithm. The technique is used to evaluate and analyse the factors that impact the study. It also aids the process of excluding or ignoring those factors which are either irrelevant or do not contribute much to the end result. By doing so the model enhances the accuracy levels of the prediction. It also reduces the number the computations that are required to be carried out arrive the desired efficiency.

After data model is created, we use multiple analytics and statistical methods mining air quality. We perform the air quality prediction over three steps or phases. The first is using exploratory data analysis to find the characteristics from our data set that are essential for predicting the air quality. This is followed by calculation of quantiles which basically helps in dividing the data we have into equal sized adjacent subgroups. The next phase is where we do the prediction. This is done using Long

Short-Term Memory network. The LSTM helps us to predict the next set of values in a sequence by learning from a series of observations from the past.

5.2 EXPLORATORY DATA ANALYSIS

The proposed system uses exploratory data analysis or the EDA method to analyze the air pollution data set and summarize the main and essential characteristics from the data. Prior to this data analysis phase, pre-processing of the available data and noise reduction were done in order to ensure high accuracy in the results that obtained. In data analysis this is the first step to be implemented before making use of formal statistical techniques. This technique was used to find the pollutants which significantly impact the pollution levels over certain time periods.

The first phase of exploratory data analysis deals with feature selection. The features such as SO₂, NO₂, NO, CO₂, and levels of various particulate matters suspended in air were found to be essential characteristics. Sulphur dioxide when inhaled causes coughing, wheezing, difficulty in breathing and in some extreme cases premature deaths. Nitrogen dioxide at 25-50 ppm levels causes bronchitis and pneumonia while at higher levels i.e. above 100 ppm it causes deaths due to asphyxiation. Once these essential features are identified, the next phase of correlation analysis is performed. Correlation analysis is like a preliminary technique used to identify the relationships, linear or non-linear, among the features selected in the previous step. Correlation is calculated as the ratio between the covariance of two features to the product of their standard deviation. By doing so a correlation coefficient is obtained whose value ranges between +1 and -1.

The next step was to perform cluster analysis which gives a clear idea about the patterns in air pollutant levels which in turn will help in predicting the air quality at a future point in time. Using EDA methods proves to be extremely useful. Since all the data in the dataset is numerical, it gives a statistical evaluation of the pollutant levels and identifies those pollutants might have significant impact on the ambient air quality in the times to come.

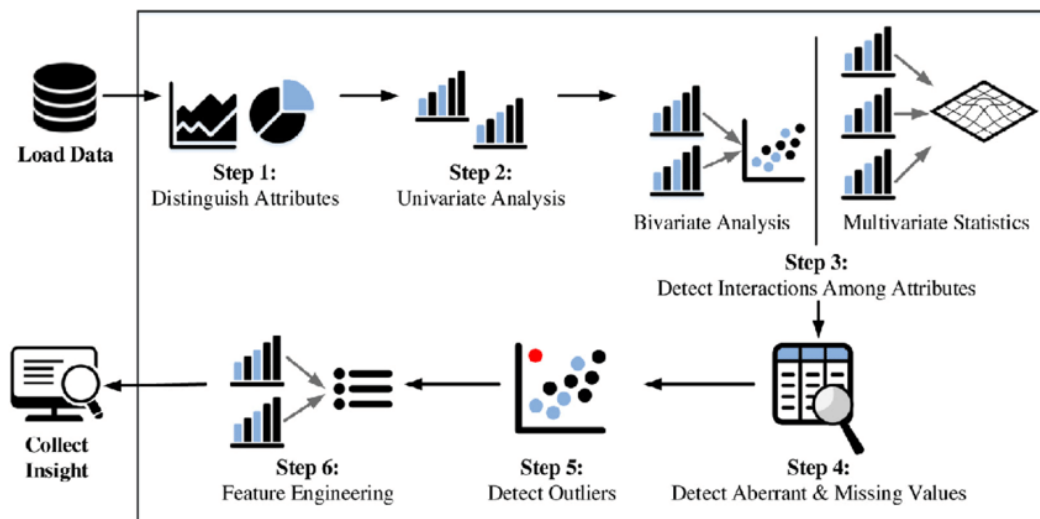


Fig 5.1 Steps in EDA

5.2.1 Algorithm

- Read the data from the csv file that contains the dataset.
- Convert the time and date columns into a single column with datetime in order to ease the process of sorting the data in the dataset.
- The data is now sorted into hourly records using the datetime column which was generated.
- EDA is done in three phases in which the first is feature selection, in this phase the characteristics from the dataset which are essential for our analysis are determined.
- Once these essential features are identified we move to the next phase of correlation analysis. Correlation analysis is like a preliminary technique used to identify the relationships, linear or non-linear, among the features selected in the previous step.
- This is followed by cluster analysis which marks the end of exploratory data analysis.

5.3 CALCULATING QUANTILES

The next phase after EDA is the calculation of quantiles. Quantiles are quantities used to split the distribution that is being working with in a uniform manner. It divides

the data into equal sized adjacent groups which eases the process of representing the statistical information as well as the prediction process. The calculation of quantiles helps to find the minimum and maximum values for all features, in turn helping in finding the high and low outliers in the data. It starts with the number of equal sized divisions needed for the data distribution. Then the quantile that will enable such division is identified. Some of the quantiles that are used commonly are addressed with specific names like median is the two quantile which divides the distribution into two equal halves, quartiles are the four quantile and so on.

The quantiles that were calculated were data count, mean, standard deviation, minimum, maximum, 25%, 50%, 75% values. This is done using the quantile function provided by Pandas. Pandas dataframe.quantile() function return values at the given quantile over requested axis, a NumPy percentile. Segregating the data and grouping it together with the help of quantiles enables easier prediction and mining of the data. The minimum and maximum quantile values are used for computing the outliers in the available data. The first and last records are removed for all features in order to prevent missing value extrapolation issues.

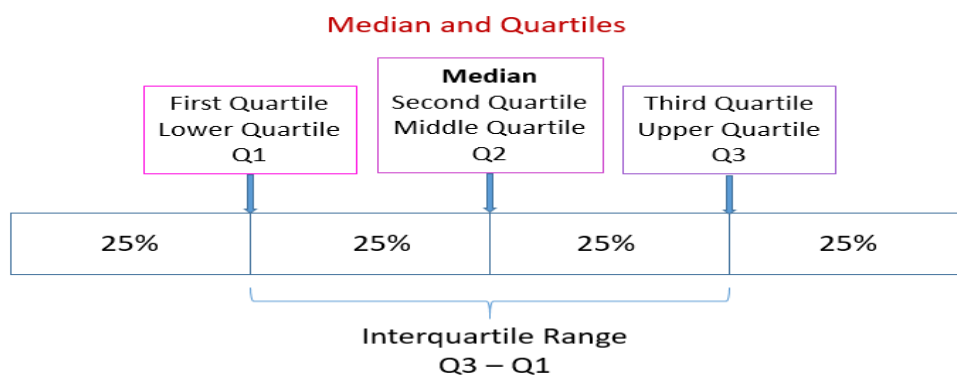


Fig 5.2 Quartiles

After computing these quantiles and performing further calculations we perform interpolation. Interpolate () function provided by Pandas library is basically used to fill the values in the series which are not applicable or in other words, weren't available in the data that was collected. A variety of interpolation methodologies are used by this method provided by Pandas to fill all the blanks in the dataset instead of just hard coding any value like 0 or 1. This takes care of the missing records in the dataset which might otherwise be an issue while predicting the air quality.

5.3.1 Algorithm

- Compute the 25% and 75% quantiles for each attribute using the `dataframe.quantile()` function provided by Pandas.
- Find the interquartile range (IQR), which is nothing but the difference between the 25% and 75% quantile values.
- After this the min and max values for every attribute in the dataset is computed.
- Using minimum, maximum, 25% quantile, 75% quantile and the IQR computed before, we proceed to find the minimum interquartile range and maximum interquartile range values.
- In the next step with all the quantiles computed above, find the low and high outliers and interpolate the same using the Pandas `interpolate()` function.

5.4 LSTM PREDICTION MODULE

LSTM network was used for predicting the air quality. LSTM is expanded as Long Short-Term Memory network. Recurrent Neural Networks are better performing than normal ANN's and LSTM is one, which makes it much more efficient than previously used networks. It has the capability to learn long term dependencies. This proves to be an advantage while are predicting the air quality for a certain point in the future as the network will work based on the memory of the past. Backpropagation Through Time (BPTT) is the algorithm that is used to train the LSTM network. BPTT is a gradient based algorithm which is generally used for Recurrent Neural Networks. The input training data for the LSTM network was provided as a matrix. This algorithm updates network weights to reduce the error. It is a supervised learning algorithm.

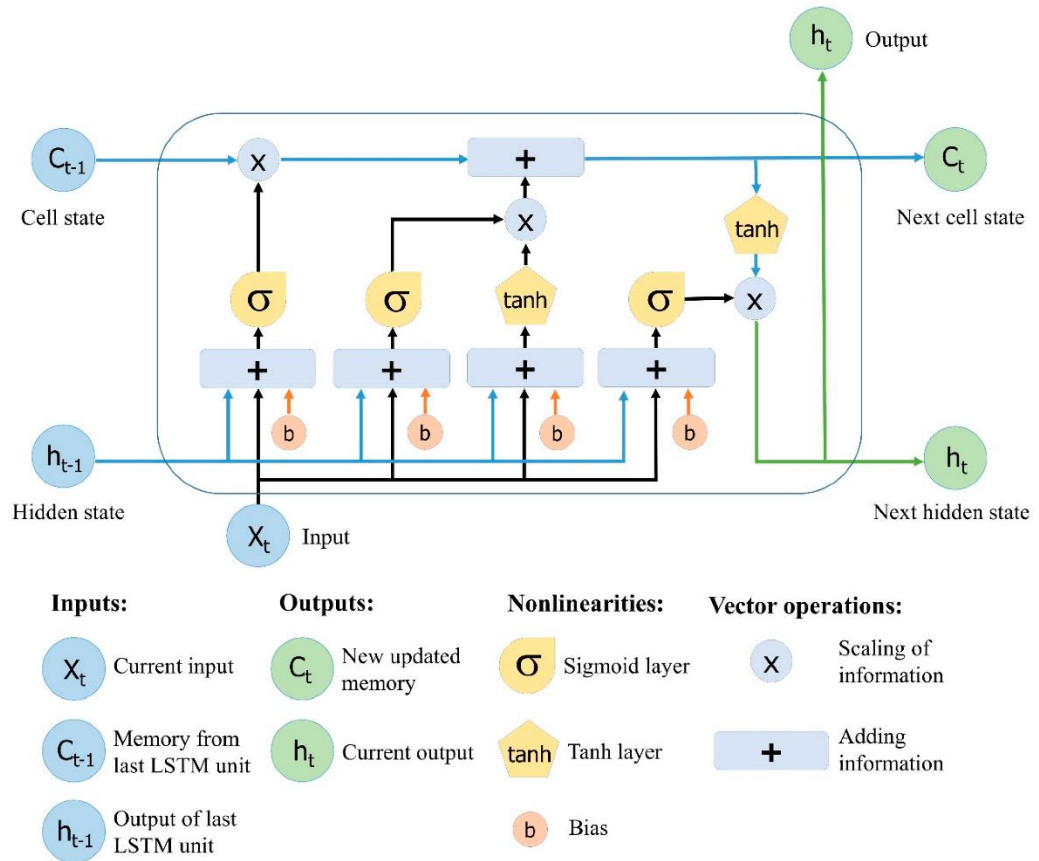


Fig 5.3 LSTM network and its components

Feeding time steps initiates the process. The LSTM is given one input for each timestep and it produces one output. The lookback for the LSTM was set to one and 60 hidden layers were used. The mean squared errors or loss is computed and accumulated over successive timesteps and the network is then rolled back to update the weights. The optimization was done using Adam optimizer in which the learning rate was customized to 0.0005 and ReduceLROnPlateau provided by keras was used for reducing the learning rate when it comes across a plateau in the training and the patience was set to 5 for this. Over each epoch the root mean squared loss was calculated. Every timestep has a copy of the network, an input timestep and an output. For each timestep error calculation is done and accumulate it and then unrolling is done and weights are updated. By doing so it is observed that there is a subsequent decrease in RMS error in consecutive epochs leading to better accuracy. The testing is then done and if the loss in testing is less than the Standard Deviation for the dataset

that was calculated along with the quantiles, then the model is good enough for the prediction.

5.4.1 Algorithm

- Create a variable for the model and assign a sequential type model to it.
- Set the model hyperparameters like the number of epochs, batch size, verbose etc.
- Pass the records in the training set as the input and generate predictions.
- Compute the loss value and the root mean squared error for the training and call this the train score or the training error.
- Repeat the same over successive epochs to reduce the error and obtain accurate predictions.
- Now pass the test inputs through the model and generate test predictions and compute its loss and RMS error.
- Compare this test error value with the standard deviation and if its lesser then the model is effective.
- If not, then tweak the hyperparameters successively till the desired accuracy is obtained.
- Once the model is satisfactory, plot the fitting curve.

5.5 SUMMARY

Summarizing the contents of this chapter we see that the proposed system is developed over three main modules which are exploratory data analysis, calculating quantiles and prediction. The dataset is a file with csv or comma separated values as its records and this data has temporal information about the pollutant levels. This is fed as the input and we obtain future predictions as the output. The predictions are then used to plot a fitting curve. This fitting curve shows the original dataset, which is the expected outcome and the predicted data which is the result produced by the proposed system.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the most crucial phase of the project where all theoretical ideas take proper form. This is the stage that leads us to experimental results, which help in concluding on the output of the system that was created in the implementation stage. All this while the focus was on the previously existing systems and why they were inefficient. But, now the focus shifts entirely to planning and effective execution of the proposed system to obtain desired results. The more complex the system is, the more efforts go into realising the designs and dependencies outlined previously.

6.2 SAMPLE CODE

This contains the complete code for the proposed air quality prediction system. This code is written in python and uses a dataset of 9232 pollution records in comma separated format spanning from 19/05/2019 to 18/08/2018 was used to train the entire pipeline of the air quality prediction system. From the dataset we combine the data from the time and date fields into a single string in datetime format. We then proceed to group the same and form hourly data. This is followed by feature selection. The minimum, maximum values for each of these attributes from the previous step was identified and using this the outliers were interpolated. Thus, the graphical representations of the predictions were obtained with the help of the following code.

```
#IMPORT NECESSARY PACKAGES
```

```
import datetime, warnings, scipy
```

```
import pandas as pand
```

```
import numpy as np
```

```
import seaborn as sea
```

```

import matplotlib as mplot
import matplotlib.pyplot as pplot
pplot.rcParams["patch.force_edgecolor"] = True
pplot.style.use('fivethityeight')
mplot.rc('patch', edgecolor = 'gray', linewidth=1)
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "last_expr"
pand.options.display.max_columns = 50
%matplotlib inline
warnings.filterwarnings("ignore")
import math
import keras
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import mean_squared_error
df = pand.read_csv('2019-05-19.csv', sep=' ')
df.head()

# CONVERT TIME TO DATETIME FORMAT:
def combine_date(df, tab_name):
    list_tab = []
    for i in range(df.shape[0]):
        list_tab.append(df.loc[i, 'Tanggal'] + 'T' + df.loc[i, tab_name][0:2])
    return np.array(list_tab, dtype='datetime64')

# CONVERT TIME TO DATETIME FORMAT:
df['Datetime'] = combine_date(df, 'Jam')

# CONVERT INTO HOURLY DATA
df = df[['Datetime','O3','CO','NO2','SO2','NO','CO2','VOC','PM1','PM2.5','PM4','PM10','TSP','TEMP','HUM','WS','WD','ISPU']]
df2 = df.groupby(['Datetime']).mean()
df2.head()

```

```
# descriptive statistics
```

```
df2.describe()
```

```
# CALCULATE QUANTILE FROM EACH ATTRIBUTES
```

```
def calculate_quantile (i, df2):
```

```
    QUART1 = df2[[i]].quantile(0.25)[0]
```

```
    QUART3 = df2[[i]].quantile(0.75)[0]
```

```
    IQR = QUART3 - QUART1
```

```
    min = df2[[i]].min()[0]
```

```
    max = df2[[i]].max()[0]
```

```
    min_IQR = QUART1 - 1.5*IQR
```

```
    max_IQR = QUART3 + 1.5*IQR
```

```
    return QUART1, QUART3, min, max, min_IQR, max_IQR
```

```
df2.drop(index=[df2.index[0], df2.index[df2.shape[0]-1]], inplace=True)
```

```
# FIND AND INTERPOLATE THE OUTLIERS
```

```
for i in df2.columns:
```

```
    print('\nAttribute-',i,':')
```

```
    QUART1, QUART3, min, max, min_IQR, max_IQR = calculate_quantile(i, df2)
```

```
    print('QUART1 = %.2f' % QUART1)
```

```
    print('QUART3 = %.2f' % QUART3)
```

```
    print('min IQR = %.2f' % min_IQR)
```

```
    print('max IQR = %.2f' % max_IQR)
```

```
    if (min < min_IQR):
```

```
        print('---> Low outlier is found = %.2f' % min)
```

```
    if (max > max_IQR):
```

```
        print('---> High outlier is found = %.2f' % max)
```

```
def convert_nan (x, max_IQR=max_IQR, min_IQR=min_IQR):
```

```
    if ((x > max_IQR) | (x < min_IQR)):
```

```
        x = np.nan
```

```
    else:
```

```
        x = x
```

```

    return x
def convert_nan_HUM (x, max_IQR=100.0, min_IQR=min_IQR):
    if ((x > max_IQR) | (x < min_IQR)):
        x = np.nan
    else:
        x = x
    return x
if (i == 'HUM'):
    df2[i] = df2[i].map(convert_nan_HUM)
    df2[i] = df2[i].interpolate(method='linear')
if (i != 'HUM'):
    df2[i] = df2[i].map(convert_nan)
    df2[i] = df2[i].interpolate(method='linear')
if (len(df2[df2[i].isnull()][i]) == 0):
    print('##### Outliers have been interpolated #####')
df2.head()

```

#PREDICTION FOR TEMPERATURE ATTRIBUTE

```

dataset = np.log1p(df2[['TEMP']].values)
dataset.shape
dist_df = pd.DataFrame({'TEMP' : df2['TEMP'].values, 'log_TEMP' : dataset[:,0]})
plt.figure(figsize=(12,5))
dist_df.hist();
train_size = int(len(dataset) * 0.75)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
print(len(train), len(test))

```

#CREATING THE DATAFRAME FOR TRAINING AND TESTING

```

def dataset_creation(dataset, lookback=1):
    dataX, dataY = [], []

```



```

    for i in range(len(dataset)-lookback-1):
        s = dataset[i:(i+lookback), 0]
        dat axis.append(s)
        dataY.append(dataset[i + lookback, 0])
    return np.array(dataX), np.array(dataY)

# SPLIT THE DATASET INTO TRAIN AND TEST SETS
lookback = 1
trX, trY = dataset_creation(train, lookback)
tsX, tsY = dataset_creation(test, lookback)
print('Dimensions of trX :', trX.shape)
print('Dimensions of trY :', trY.shape)
print('Dimensions of tsX :', tsX.shape)
print('Dimensions of tsY :', tsY.shape)
trX = np.reshape(trX, (trX.shape[0], 1, trX.shape[1]))
tsX = np.reshape(tsX, (tsX.shape[0], 1, tsX.shape[1]))
print('Dimensions of trX :', trX.shape)
print('Dimensions of tsX :', tsX.shape)

# TRAIN THE LSTM MODEL WITH DESIRED PARAMETERS
model = Sequential()
net.add(LSTM(4, input_shape=(1, lookback)))
net.add(Dense(1))
net.compile(loss='mean_squared_error', optimizer='adam')
net.fit(trX, trY, epochs=500, batch_size=32, verbose=2)

# COMPUTE ROOT MEAN SQUARED ERROR FOR TRAINING AND TESTING
trainsetPrediction = net.predict(trX)
testsetPrediction = net.predict(tsX)
trainsetPrediction = np.expml(trainsetPrediction)
trY = np.expml(trY)

```

```

testsetPrediction = np.expm1(testsetPrediction)
tsY = np.expm1(tsY)
trainError = math.sqrt(mean_squared_error(trY, trainsetPrediction[:,0]))
print('Train Score: %.2f RMSE' % (trainError))
testError = math.sqrt(mean_squared_error(tsY, testsetPrediction[:,0]))
print('Test Score: %.2f RMSE' % (testError))

# EVALUATE THE PERFORMANCE OF THE MODEL
test_series = pd.Series(tsY)
if testError < test_series.std():
    print('\n[ Model performance is GOOD enough ]')
    print('\nRMSE of test prediction < Standard deviation of test dataset')
    print('%.2f' % (testError), '<', ' %.2f' % (test_series.std()))
else:
    print('\n[ Model performance is NOT GOOD enough ]')
    print('\nRMSE of test prediction > Standard deviation of test dataset')
    print('%.2f' % (testError), '>', ' %.2f' % (test_series.std()))

# PLOT THE FITTING CURVE
trainsetPredictionionPlot = np.empty_like(dataset)
trainsetPredictionionPlot[:, :] = np.nan
trainsetPredictionionPlot[lookback:len(trainsetPrediction)+lookback, :] = trainsetPrediction
testsetPredictionionPlot = np.empty_like(dataset)
testsetPredictionionPlot[:, :] = np.nan
testsetPredictionionPlot[len(trainsetPrediction)+(lookback*2)+1:len(dataset)-1, :] = testsetPrediction

time_axis = np.linspace(0, dataset.shape[0]-1, 15)
time_axis = np.array([int(i) for i in time_axis])
time_axisLab = np.array(df2.index, dtype='datetime64[D]')
fig = plt.figure()
ax = fig.add_axes([0, 0, 2.1, 2])

```

```

axis.plot(np.expm1(dataset), label='Original Dataset')
axis.plot(trainsetPredictionionPlot, color='orange', label='Trainset Predictions')
axis.plot(testsetPredictionionPlot, color='red', label='Testset Predictions')
axis.set_xticks(time_axis)
axis.set_xticklabels(time_axisLab[time_axis], rotation=45)
axis.set_xlabel('\nDate', fontsize=27, fontweight='bold')
axis.set_ylabel('TEMP', fontsize=27, fontweight='bold')
axis.legend(loc='best', prop= {'size':20})
axis.tick_params(size=10, labels=15)
axis.set_xlim([-1,1735])
ax1 = fig.add_axes([2.3, 1.3, 1, 0.7])
ax1.plot(np.expm1(dataset), label='Original Dataset')
ax1.plot(testsetPredictionionPlot, color='red', label='Testset Predictions')
ax1.set_xticks(time_axis)
ax1.set_xticklabels(time_axisLab[time_axis], rotation=45)
ax1.set_xlabel('Date', fontsize=27, fontweight='bold')
ax1.set_ylabel('TEMP', fontsize=27, fontweight='bold')
ax1.tick_params(size=10, labels=15)
ax1.set_xlim([1360,1735]);

```

#PREDICTION FOR PM10 ATTRIBUTE

```

dataset = np.log1p(df2[['PM10']].values)
dataset.shape
dist_df = pd.DataFrame({'PM10' : df2[['PM10']].values, 'log_PM10' : dataset[:,0]})
plt.figure(figsize=(12,5))
dist_df.hist();
train_size = int(len(dataset) * 0.75)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
print(len(train), len(test))
lookback = 1
trX, trY = dataset_creation(train, lookback)

```

```

tsX, tsY = dataset_creation(test, lookback)
print('Dimensions of trX :', trX.shape)
print('Dimensions of trY :', trY.shape)
print('Dimensions of tsX :', tsX.shape)
print('Dimensions of tsY :', tsY.shape)
trX = np.reshape(trX, (trX.shape[0], 1, trX.shape[1]))
tsX = np.reshape(tsX, (tsX.shape[0], 1, tsX.shape[1]))
print('Dimensions of trX :', trX.shape)
print('Dimensions of tsX :', tsX.shape)
model = Sequential()
net.add(LSTM(4, input_shape=(1, lookback)))
net.add(Dense(1))
net.compile(loss='mean_squared_error', optimizer='adam')
net.fit(trX, trY, epochs=1500, batch_size=32, verbose=2)
trainsetPrediction = net.predict(trX)
testsetPrediction = net.predict(tsX)
trainsetPrediction = np.expm1(trainsetPrediction)
trY = np.expm1(trY)
testsetPrediction = np.expm1(testsetPrediction)
tsY = np.expm1(tsY)
trainError = math.sqrt(mean_squared_error(trY, trainsetPrediction[:,0]))
print("Train Score: %.2f RMSE" % (trainError))
testError = math.sqrt(mean_squared_error(tsY, testsetPrediction[:,0]))
print("Test Score: %.2f RMSE" % (testError))
test_series = pd.Series(tsY)
if testError < test_series.std():
    print("\n[ Model performance is GOOD enough ]")
    print("\nRMSE of test prediction < Standard deviation of test dataset")
    print("%.2f" % (testError), '<', "%.2f" % (test_series.std()))
else:
    print("\n[ Model performance is NOT GOOD enough ]")
    print("\nRMSE of test prediction > Standard deviation of test dataset")

```

```

print('%0.2f' % (testError), '>', '%0.2f' % (test_series.std()))
trainsetPredictionionPlot = np.empty_like(dataset)
trainsetPredictionionPlot[:, :] = np.nan
trainsetPredictionionPlot[lookback:len(trainsetPrediction)+lookback, :] = trainsetPrediction
testsetPredictionionPlot = np.empty_like(dataset)
testsetPredictionionPlot[:, :] = np.nan
testsetPredictionionPlot[len(trainsetPrediction)+(lookback*2)+1:len(dataset)-1, :] = testsetPrediction
time_axis = np.linspace(0, dataset.shape[0]-1, 15)
time_axis = np.array([int(i) for i in time_axis])
time_axisLab = np.array(df2.index, dtype='datetime64[D]')
fig = plt.figure()
ax = fig.add_axes([0, 0, 2.1, 2])
ax.plot(np.expml(dataset), label='Original Dataset')
ax.plot(trainsetPredictionionPlot, color='orange', label='Trainset Predictions')
ax.plot(testsetPredictionionPlot, color='red', label='Testset Predictions')
ax.set_xticks(time_axis)
ax.set_xticklabels(time_axisLab[time_axis], rotation=45)
ax.set_xlabel('\nDate', fontsize=27, fontweight='bold')
ax.set_ylabel('PM10', fontsize=27, fontweight='bold')
ax.legend(loc='best', prop= {'size':20})
ax.tick_params(size=10, labels=15)
ax.set_xlim([-1,1735])
ax1 = fig.add_axes([2.3, 1.3, 1, 0.7])
ax1.plot(np.expml(dataset), label='Original Dataset')
ax1.plot(testsetPredictionionPlot, color='red', label='Testset Predictions')
ax1.set_xticks(time_axis)
ax1.set_xticklabels(time_axisLab[time_axis], rotation=45)
ax1.set_xlabel('Date', fontsize=27, fontweight='bold')
ax1.set_ylabel('PM10', fontsize=27, fontweight='bold')
ax1.tick_params(size=10, labels=15)

```

```
ax1.set_xlim([1360,1735]);
```

#PREDICTION FOR HUMIDITY ATTRIBUTE

```
dataset = np.log1p(df2[['HUM']].values)
dataset.shape
dist_df = pd.DataFrame({'HUM' : df2['HUM'].values, 'log_HUM' : dataset[:,0]})
plt.figure(figsize=(12,5))
dist_df.hist();
train_size = int(len(dataset) * 0.75)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
print(len(train), len(test))
lookback = 1
trX, trY = dataset_creation(train, lookback)
tsX, tsY = dataset_creation(test, lookback)
print('Dimensions of trX :', trX.shape)
print('Dimensions of trY :', trY.shape)
print('Dimensions of tsX :', tsX.shape)
print('Dimensions of tsY :', tsY.shape)
model = Sequential()
net.add(LSTM(50, input_shape=(1, lookback)))
net.add(Dense(1))
rlrop = ReduceLROnPlateau(monitor='validationSetLoss', factor=0.1, patience=5,
min_delta=1E-4)
adam=keras.optimizers.Adam(learning_rate=0.00005, beta_1=0.9, beta_2=0.999,
amsgrad=False)
net.compile(loss='mean_squared_error', optimizer= adam)
net.fit(trX, trY, validation_data=(tsX, tsY), epochs=300, batch_size=32, verbose=2,
callbacks=[rlrop])
trainsetPrediction = net.predict(trX)
testsetPrediction = net.predict(tsX)
trainsetPrediction = np.expm1(trainsetPrediction)
```

```

trY = np.expm1(trY)
testsetPrediction = np.expm1(testsetPrediction)
tsY = np.expm1(tsY)
trainError = math.sqrt(mean_squared_error(trY, trainsetPrediction[:,0]))
print('Train Score: %.2f RMSE' % (trainError))
testError = math.sqrt(mean_squared_error(tsY, testsetPrediction[:,0]))
print('Test Score: %.2f RMSE' % (testError))
test_series = pd.Series(tsY)

if testError < test_series.std():
    print('\n[ Model performance is GOOD enough ]')
    print('\nRMSE of test prediction < Standard deviation of test dataset')
    print('%.2f' % (testError), '<', ' %.2f' % (test_series.std()))
else:
    print('\n[ Model performance is NOT GOOD enough ]')
    print('\nRMSE of test prediction > Standard deviation of test dataset')
    print('%.2f' % (testError), '>', ' %.2f' % (test_series.std()))

trainsetPredictionionPlot = np.empty_like(dataset)
trainsetPredictionionPlot[:, :] = np.nan
trainsetPredictionionPlot[lookback:len(trainsetPrediction)+lookback,:] =
trainsetPrediction
testsetPredictionionPlot = np.empty_like(dataset)
testsetPredictionionPlot[:, :] = np.nan
testsetPredictionionPlot[len(trainsetPrediction)+(lookback*2)+1:len(dataset)-1, :] =
testsetPrediction
time_axis = np.linspace(0, dataset.shape[0]-1, 15)
time_axis = np.array([int(i) for i in time_axis])
time_axisLab = np.array(df2.index, dtype='datetime64[D]')
fig = plt.figure()
axis = fig.add_axes([0, 0, 2.1, 2])
axis.plot(np.expm1(dataset), label='Original Dataset')

```

```

axis.plot(trainsetPredictionionPlot, color='orange', label='Trainset Predictions')
axis.plot(testsetPredictionionPlot, color='red', label='Testset Predictions')
axis.set_xticks(time_axis)
axis.set_xticklabels(time_axisLab[time_axis], rotation=45)
axis.set_xlabel('\nDate', fontsize=27, fontweight='italic')
axis.set_ylabel('HUM', fontsize=27, fontweight='italic')
axis.legend(loc='best', prop= {'size':20})
axis.tick_params(size=10, labels=17)
axis.set_xlim([-1,1735])
axis1 = fig.add_axes([2.3, 1.3, 1, 0.7])
axis1.plot(np.expm1(dataset), label='Base Dataset')
axis1.plot(testsetPredictionionPlot, color='red', label='Testset Predictions')
axis1.set_xticks(time_axis)
axis1.set_xticklabels(time_axisLab[time_axis], rotation=45)
axis1.set_xlabel('Date', fontsize=27, fontweight='bold')
axis1.set_ylabel('HUM', fontsize=27, fontweight='bold')
axis1.tick_params(size=10, labels=15)
axis1.set_xlim([1360,1735]);

```

6.3 OUTPUT

A dataset of 9232 pollution records in comma separated format spanning from 19/05/2019 to 18/08/2018 was used in the proposed system. The data from the time and date fields from the dataset was combined into a single string in datetime format. This dataset was cleaned and sorted into hourly data. This resulted in a dataset with 1735 records. After this the quantiles were computed, and the outliers were interpolated. The minimum, maximum values for each of these attributes was identified and using this the outliers were interpolated. 75% of this dataset which is 1301 records was used to train the entire pipeline of the air quality prediction system and 25% i.e. 434 records were used for testing. This is followed by feature selection, in which Temperature, Humidity, PM10, O3, CO, NO2, SO2, NO, CO2, TSP were identified as the key features or attributes. At this point the data available has been

cleaned and the quantiles have been calculated. This data can now be used in training and testing the prediction net. Using the temperature and logarithmic temperature values dataframes are created and histograms are plotted of the same to obtain a clear picture of the data at hand.

<Figure size 864x360 with 0 Axes>

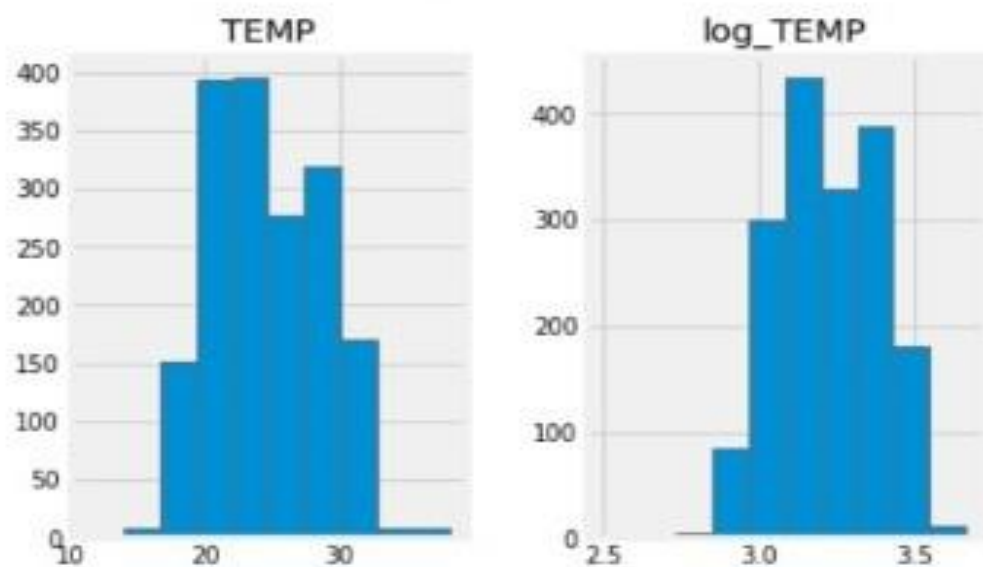


Fig 6.1. Histogram for temperature dataframe

<Figure size 864x360 with 0 Axes>

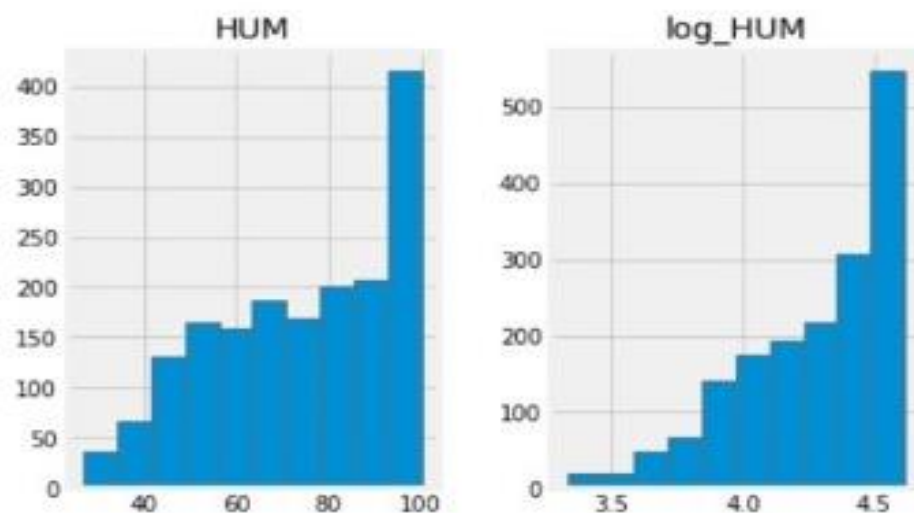


Fig 6.2 Histogram for humidity dataframe

<Figure size 864x360 with 0 Axes>

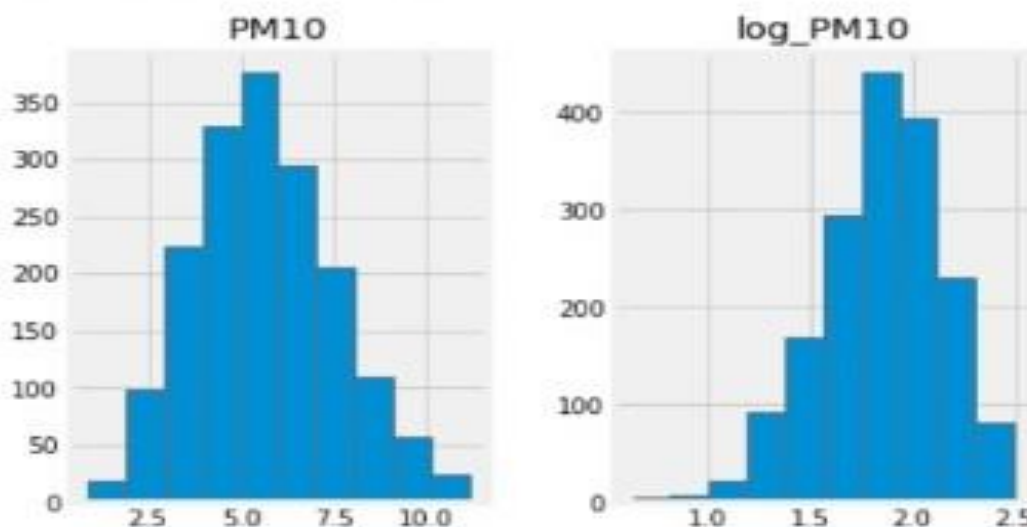


Fig 6.3 Histogram for particulate matter dataframe

With this training data of 1301 records, the LSTM is trained using the BPTT algorithm over 500 epochs in batches of 32. Over these 500 epochs the RMS Error is calculated, which consistently decreases leading up to a minimum loss in the end. Once the network has been trained, test data is used to generate predictions. The train score and test score which are basically the loss in training and the loss in testing, were generated. The air quality prediction model is then evaluated using a consecutive set of experiments. The loss computed during testing is compared with the standard deviation and the system concludes whether the model is good enough or not. The results were visualized using a fitting curve that portrays the effectiveness of the model on both train and test data as well as a comprehensive overview of the performance aspect of the entire system.

The first curve obtained is for temperature predictions and the next are for PM10 predictions followed by humidity predictions, which are some of the major influential factors in air pollution predictions. Fig 6.4 shows the plot for temperature predictions for which the loss at the end of training was 0.0088, and post testing the training RMSE was computed as 2.36 and the testing RMSE was 3.16. The standard deviation was 4.40 for the temperature data and the testing RMSE was less than this

value, hence the model was good. After this the fitting curve for temperature was plotted.

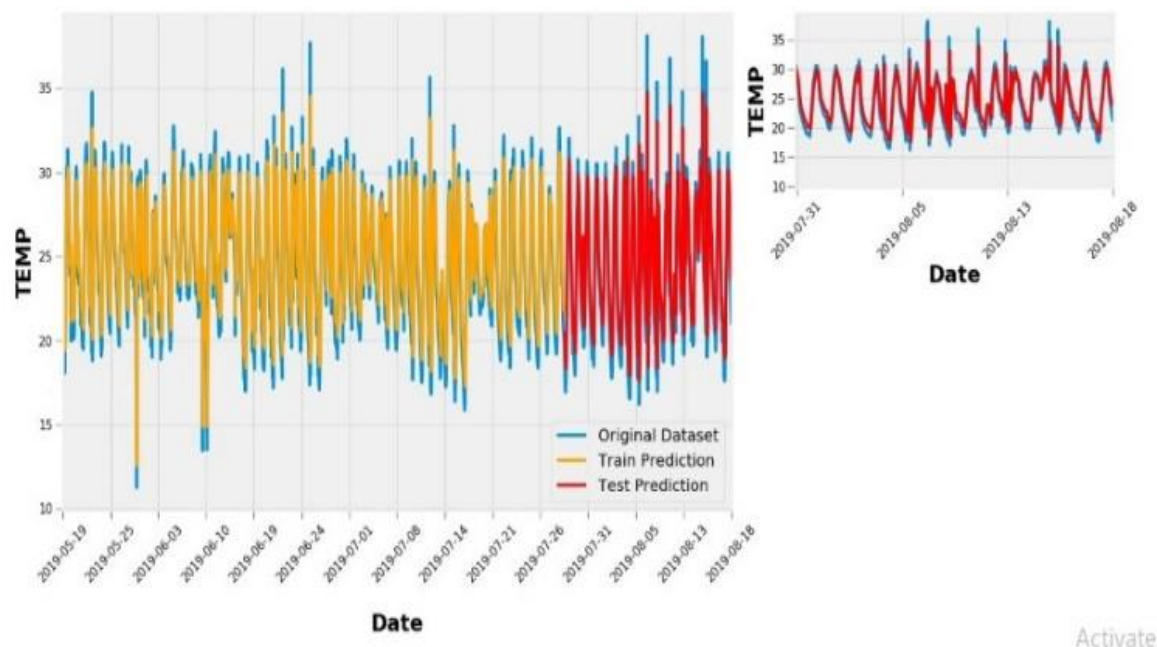


Fig 6.4 Fitting curve for temperature predictions

The same steps were carried out for PM10. Fig 6.5 shows the plot for PM10 predictions for which the loss at the end of training was 0.0131, and post testing the training RMSE was computed as 7.85 and the testing RMSE was 7.18. The standard deviation was 19.32 for the PM10 data and the testing RMSE was significantly less than this value, hence the model was good. After this the fitting curve for PM10 was plotted.

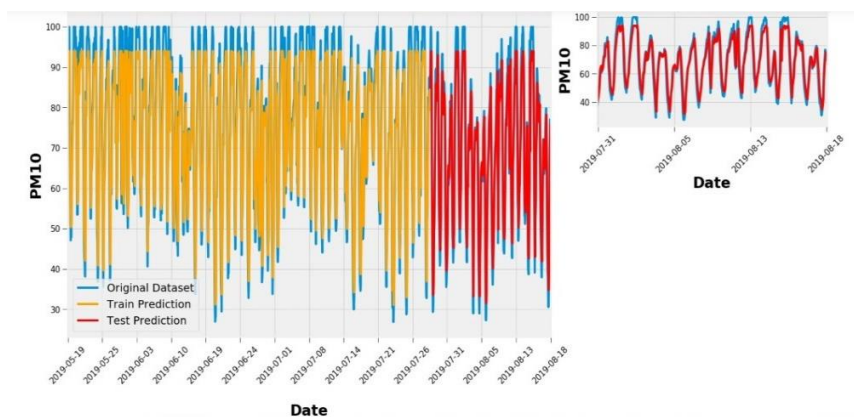


Fig 6.5 Fitting curve for PM10 predictions

The training and testing for humidity data was done. Fig 6.6 shows the plot for humidity predictions for which the loss at the end of training was 0.0156, and post testing the training RMSE was computed as 8.67 and the testing RMSE was 7.83. The standard deviation was 18.43 for the humidity data and the testing RMSE was significantly less than this value, hence the model was good. After this the fitting curve was plotted for the same. These plots help to conclude that the predictions of the proposed model are nearly consistent with the actual values.

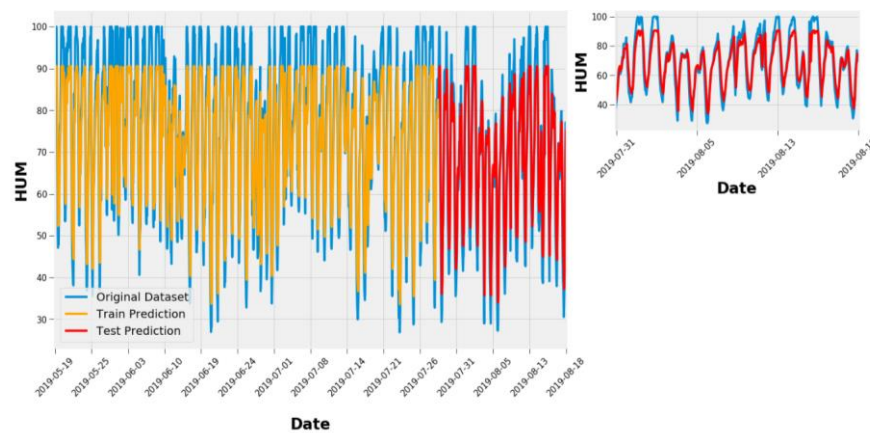


Fig 6.6 Fitting curve for Humidity predictions

Apart from the results that were obtained, we also made certain other observations. These predictions were arrived at after tweaking a number of parameters. In the process of experimenting we observed that an LSTM network consists of hidden network layers which can be increased or decreased in number. When the number of hidden layers are increased the accuracy of the prediction generally increases. The number of epochs through which we train the system is another aspect of the model which we can tweak and again like the hidden layers, an increase in number of epochs also increases the accuracy considerably. While both the above mentioned facts are true, it is to be noted that increasing any of these hyper parameters beyond a point causes overshooting beyond the most accurate result. To avoid this, when we set a learning rate, we use the function that helps in reducing learning rate value when a plateau is encountered in the learning process thereby shortening the leaps taken.

6.4 IMPLEMENTATION SCREENSHOTS

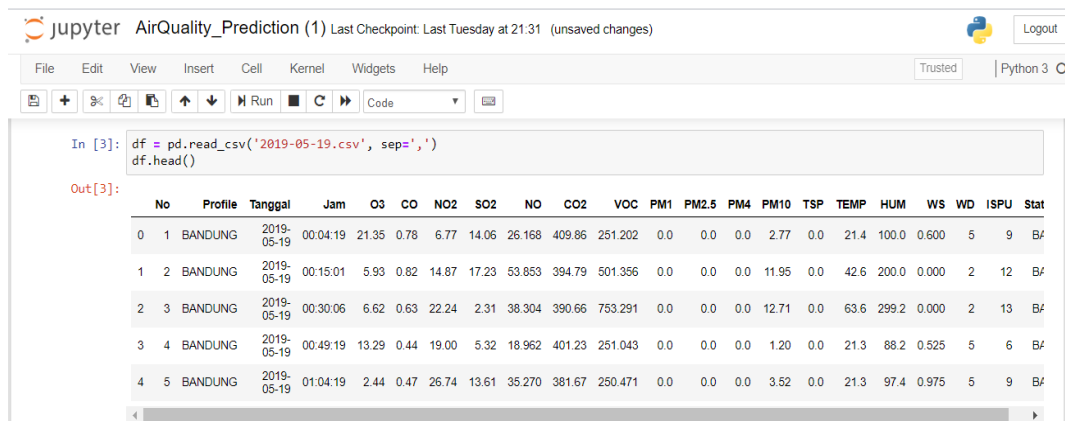


Fig 6.7 Reading the dataset

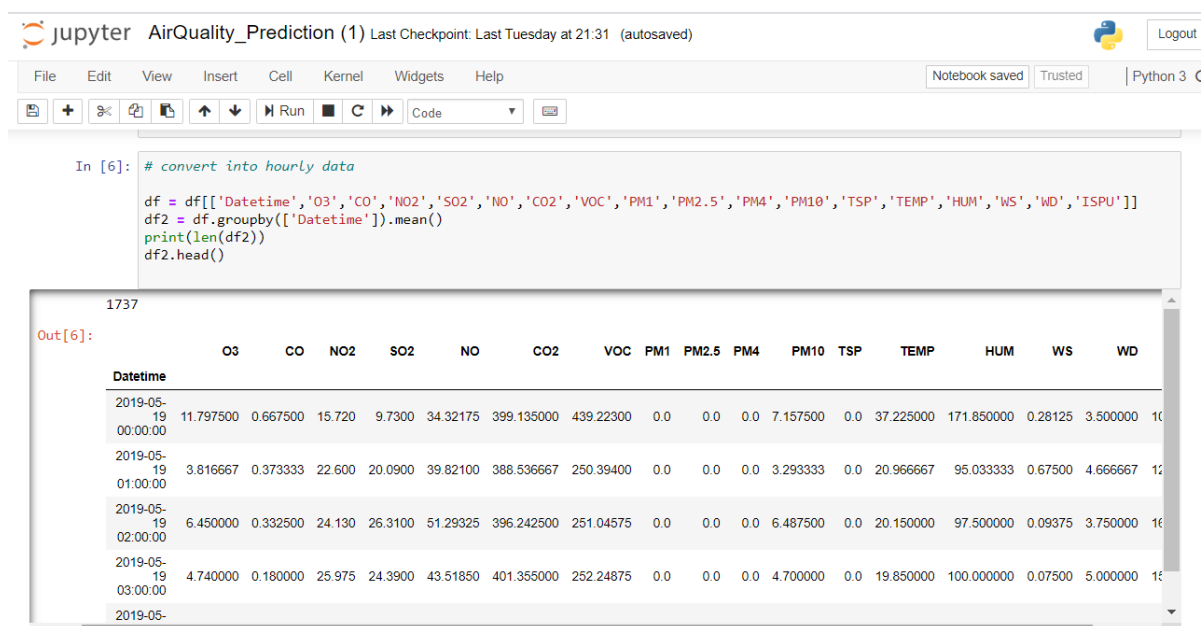
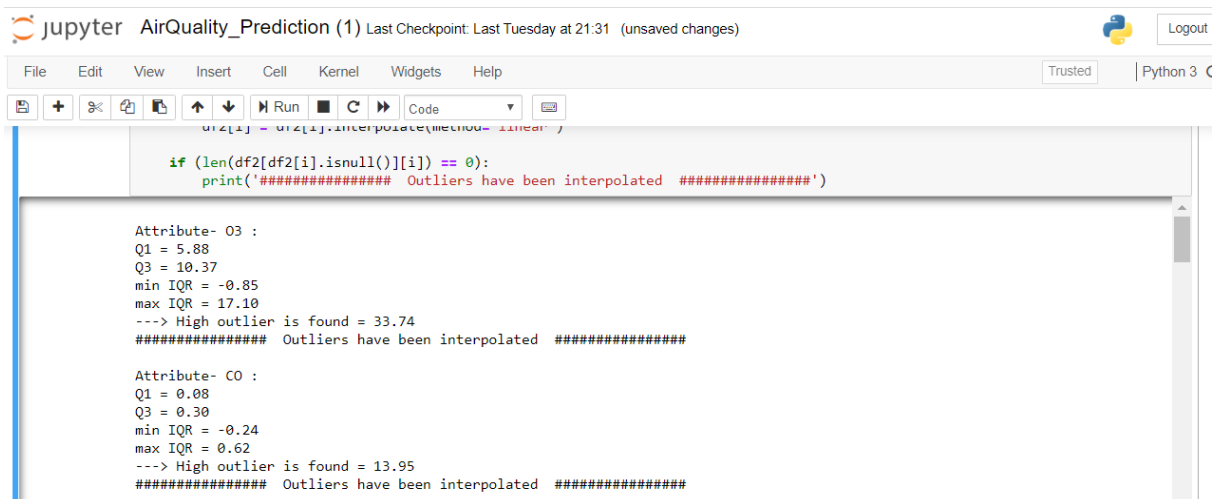


Fig 6.8 Condense dataset to hourly format



The Jupyter Notebook interface shows a code cell with the following content:

```

df2[i] = df2[i].interpolate(method='linear')

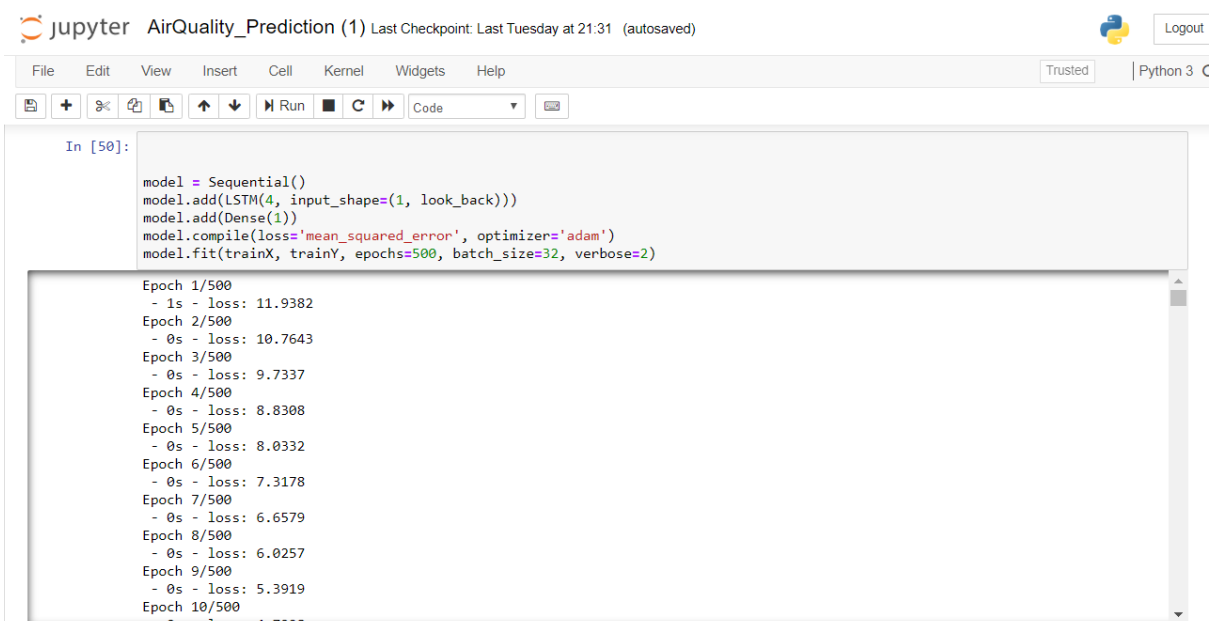
if (len(df2[df2[i].isnull()][i]) == 0):
    print('##### Outliers have been interpolated #####')

Attribute- O3 :
Q1 = 5.88
Q3 = 10.37
min IQR = -0.85
max IQR = 17.10
--> High outlier is found = 33.74
##### Outliers have been interpolated #####

Attribute- CO :
Q1 = 0.08
Q3 = 0.30
min IQR = -0.24
max IQR = 0.62
--> High outlier is found = 13.95
##### Outliers have been interpolated #####

```

Fig 6.9 Computing quantiles



The Jupyter Notebook interface shows a code cell with the following content:

```

In [50]:
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=500, batch_size=32, verbose=2)

```

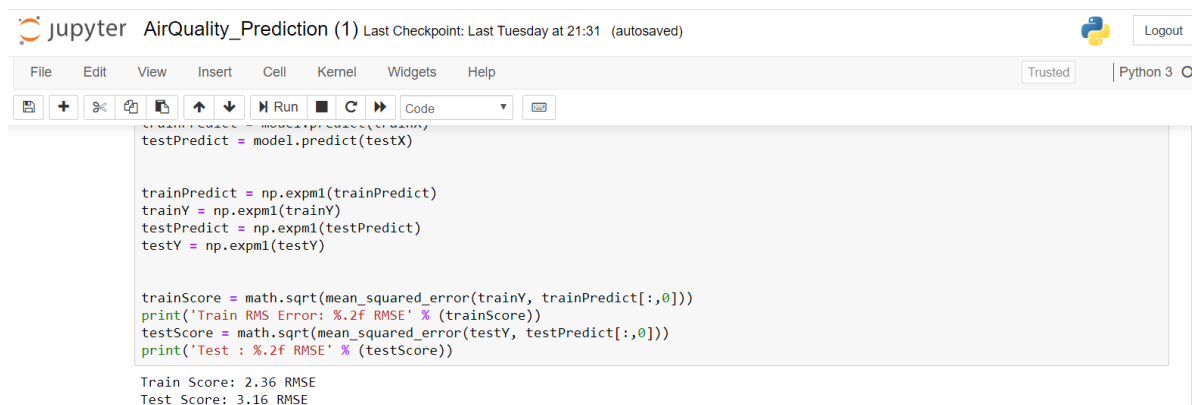
The output of the training process is displayed below the code cell:

```

Epoch 1/500
- 1s - loss: 11.9382
Epoch 2/500
- 0s - loss: 10.7643
Epoch 3/500
- 0s - loss: 9.7337
Epoch 4/500
- 0s - loss: 8.8308
Epoch 5/500
- 0s - loss: 8.0332
Epoch 6/500
- 0s - loss: 7.3178
Epoch 7/500
- 0s - loss: 6.6579
Epoch 8/500
- 0s - loss: 6.0257
Epoch 9/500
- 0s - loss: 5.3919
Epoch 10/500
- 0s - loss: 4.7208

```

Fig 6.10 Training for temperature attribute



The image shows a Jupyter Notebook titled "AirQuality_Prediction (1)". The code in the cell calculates the Root Mean Square Error (RMSE) for both training and testing data. It uses the `model.predict()` function to generate predictions for the test set. The RMSE values are calculated using the `math.sqrt(mean_squared_error())` function. The output shows a Train Score of 2.36 RMSE and a Test Score of 3.16 RMSE.

```

testPredict = model.predict(testX)

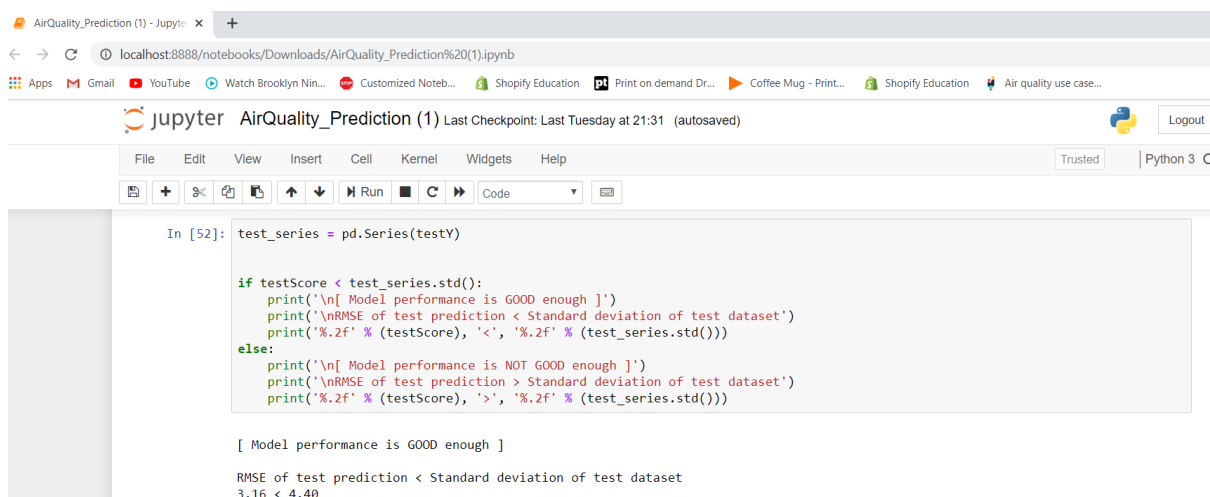
trainPredict = np.expml(trainPredict)
trainY = np.expml(trainY)
testPredict = np.expml(testPredict)
testY = np.expml(testY)

trainScore = math.sqrt(mean_squared_error(trainY, trainPredict[:,0]))
print('Train RMS Error: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY, testPredict[:,0]))
print('Test : %.2f RMSE' % (testScore))

Train Score: 2.36 RMSE
Test Score: 3.16 RMSE

```

Fig 6.11 Train and Test RMSE values for temperature



The image shows a Jupyter Notebook titled "AirQuality_Prediction (1)". The code in the cell generates test predictions for the temperature variable. It uses the `pd.Series()` function to create a series from the test data. The code then checks if the test score is less than the standard deviation of the test dataset. If it is, it prints a message indicating that the model performance is good enough. Otherwise, it prints a message indicating that the model performance is not good enough. The output shows that the RMSE of test prediction (3.16) is less than the standard deviation of test dataset (4.40), so the model performance is good enough.

```

In [52]: test_series = pd.Series(testY)

if testScore < test_series.std():
    print('\n[ Model performance is GOOD enough ]')
    print('\nRMSE of test prediction < Standard deviation of test dataset')
    print('%.2f' % (testScore), '<', '%.2f' % (test_series.std()))
else:
    print('\n[ Model performance is NOT GOOD enough ]')
    print('\nRMSE of test prediction > Standard deviation of test dataset')
    print('%.2f' % (testScore), '>', '%.2f' % (test_series.std()))

[ Model performance is GOOD enough ]

RMSE of test prediction < Standard deviation of test dataset
3.16 < 4.40

```

Fig 6.12 Generating test predictions for Temperature

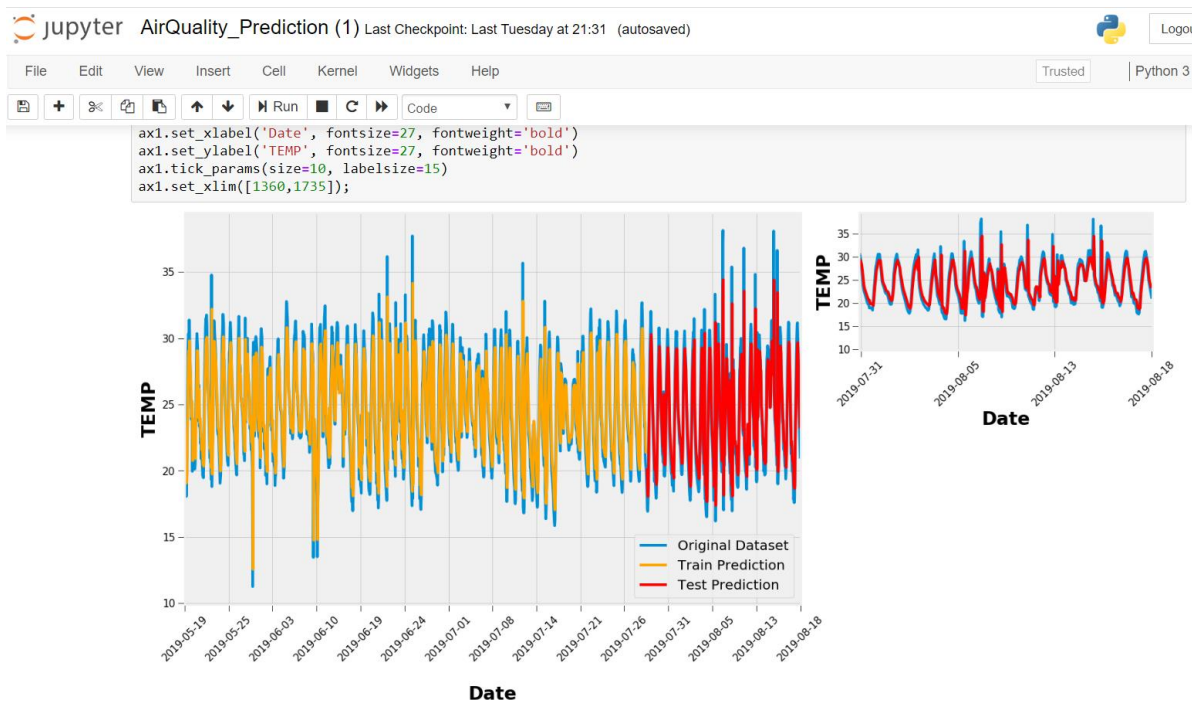


Fig 6.13 Final plot for temperature

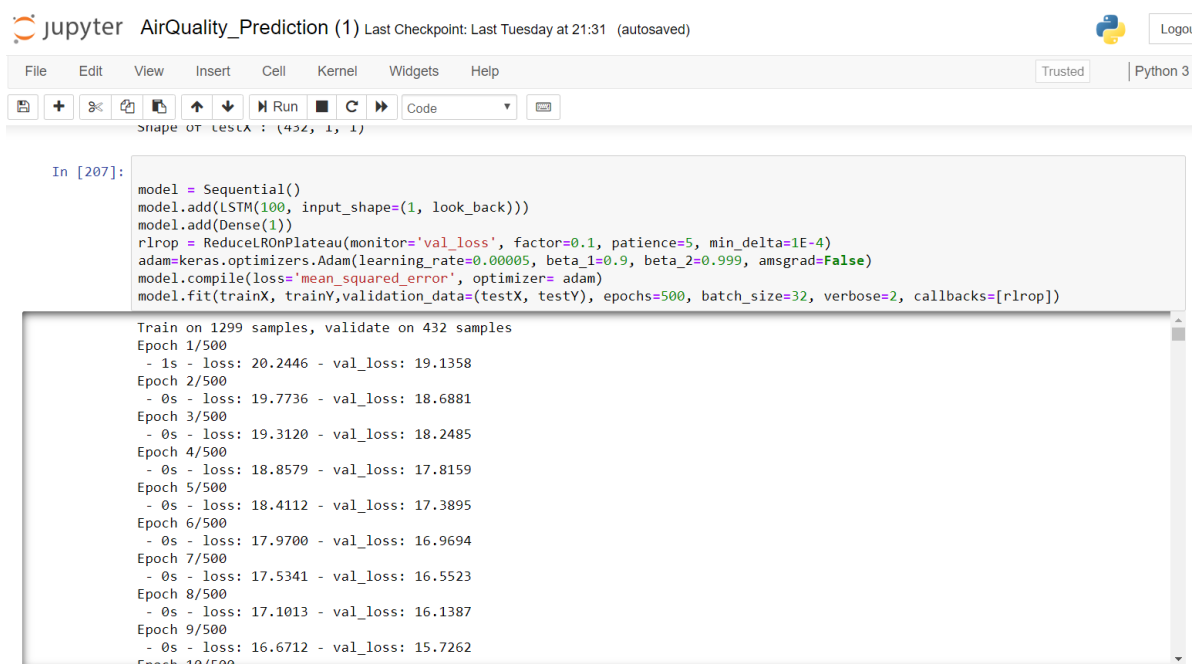
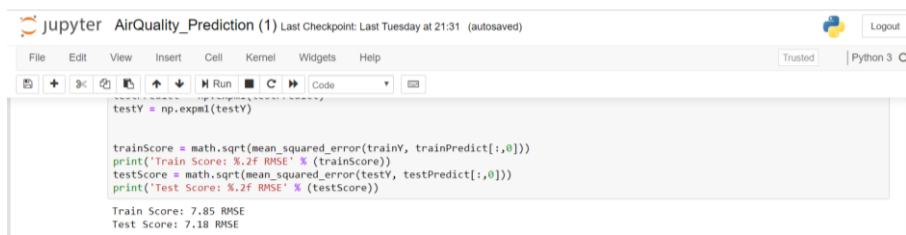


Fig 6.14 Training for PM10 attribute



```

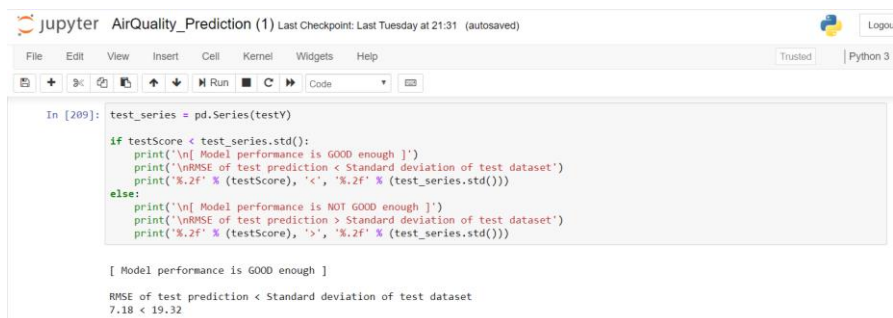
testY = np.expml(testY)

trainScore = math.sqrt(mean_squared_error(trainY, trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY, testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))

Train Score: 7.85 RMSE
Test Score: 7.18 RMSE

```

Fig 6.15 Train and Test RMSE values for PM10



```

In [209]: test_series = pd.Series(testY)

if testScore < test_series.std():
    print('\n[ Model performance is GOOD enough ]')
    print('\nRMSE of test prediction < Standard deviation of test dataset')
    print('%.2f' % (testScore), '<', ' %.2f' % (test_series.std()))
else:
    print('\n[ Model performance is NOT GOOD enough ]')
    print('\nRMSE of test prediction > Standard deviation of test dataset')
    print('%.2f' % (testScore), '>', ' %.2f' % (test_series.std()))

[ Model performance is GOOD enough ]

RMSE of test prediction < Standard deviation of test dataset
7.18 < 19.32

```

Fig 6.16 Generating test predictions for PM10

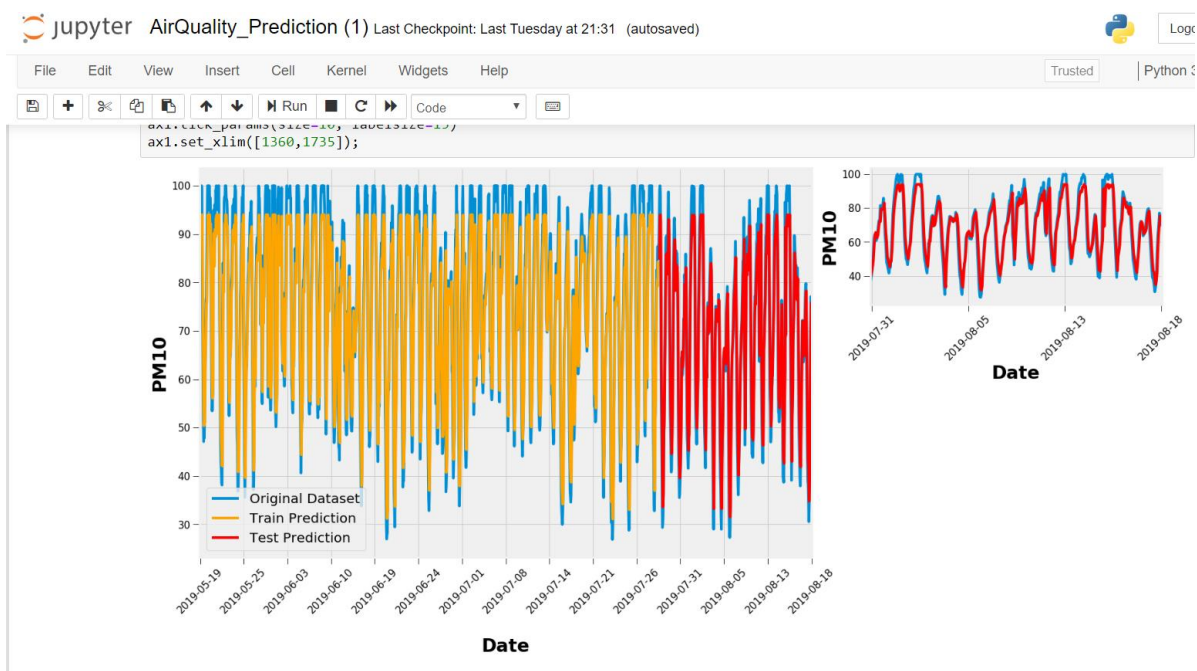


Fig 6.17 Final plot for PM10

```

Shape of trainX : (1299, 1, 1)
Shape of testX : (432, 1, 1)

In [26]: model = Sequential()
          model.add(LSTM(100, input_shape=(1, look_back)))
          model.add(Dense(1))
          r1rop = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_delta=1E-4)
          adam=keras.optimizers.Adam(learning_rate=0.00005, beta_1=0.9, beta_2=0.999, amsgrad=False)
          model.compile(loss='mean_squared_error', optimizer=adam)
          model.fit(trainX, trainY, validation_data=(testX, testY), epochs=100, batch_size=32, verbose=2, callbacks=[r1rop])

Epoch 92/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 93/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 94/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 95/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 96/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 97/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 98/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 99/100
- 0s - loss: 0.0125 - val_loss: 0.0122
Epoch 100/100
- 0s - loss: 0.0125 - val_loss: 0.0122

Out[26]: <keras.callbacks.callbacks.History at 0x23ac8553e48>

```

Fig 6.18 Training for Humidity Attribute

```

In [27]: trainPredict = model.predict(trainX)
          testPredict = model.predict(testX)

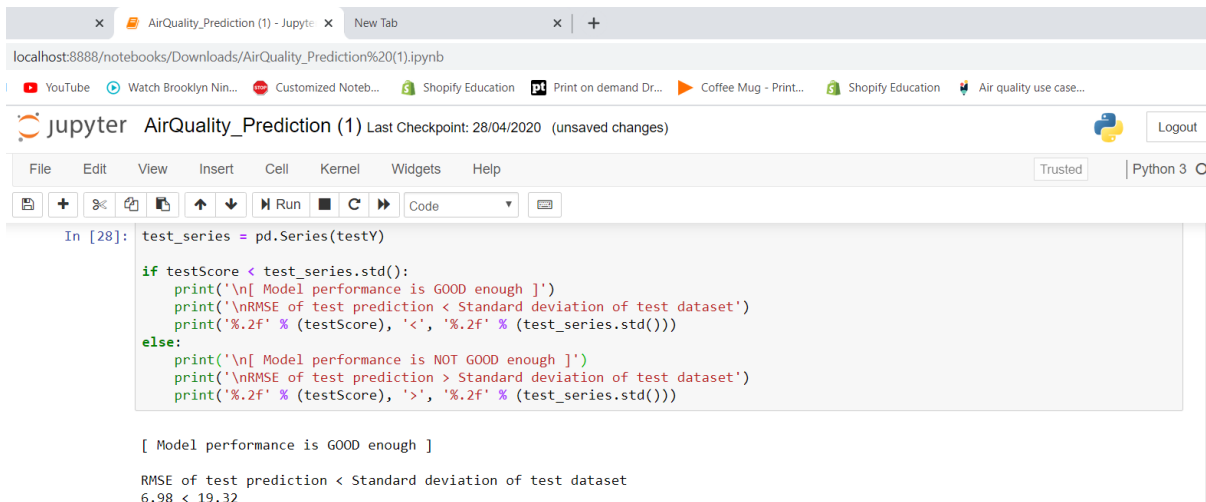
          trainPredict = np.expml(trainPredict)
          trainY = np.expml(trainY)
          testPredict = np.expml(testPredict)
          testY = np.expml(testY)

          trainScore = math.sqrt(mean_squared_error(trainY, trainPredict[:,0]))
          print('Train Score: %.2f RMSE' % (trainScore))
          testScore = math.sqrt(mean_squared_error(testY, testPredict[:,0]))
          print('Test Score: %.2f RMSE' % (testScore))

Train Score: 7.58 RMSE
Test Score: 6.98 RMSE

```

Fig 6.19 Train and Test RMSE for Humidity



The image shows a Jupyter Notebook window titled "AirQuality_Prediction (1)". The code in the cell is as follows:

```
In [28]: test_series = pd.Series(testY)

if testScore < test_series.std():
    print('\n[ Model performance is GOOD enough ]')
    print('\nRMSE of test prediction < Standard deviation of test dataset')
    print('%2f' % (testScore), '<', '%2f' % (test_series.std()))
else:
    print('\n[ Model performance is NOT GOOD enough ]')
    print('\nRMSE of test prediction > Standard deviation of test dataset')
    print('%2f' % (testScore), '>', '%2f' % (test_series.std()))

[ Model performance is GOOD enough ]

RMSE of test prediction < Standard deviation of test dataset
6.98 < 19.32
```

Fig 6.20 Generating Test Predictions for Humidity

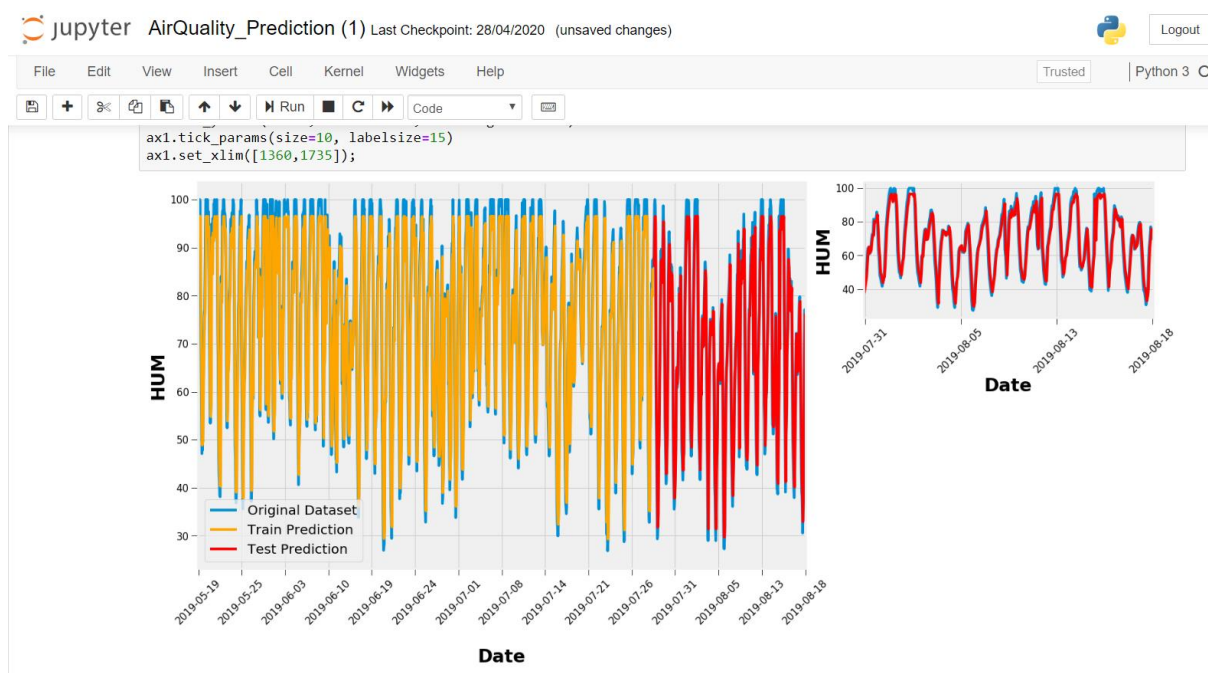


Fig 6.21 Final Plot for Humidity

6.5 SUMMARY

In this chapter we discussed the complete implementation of the air quality prediction system. The entire code along for the process of reading data, analysing this data, calculating quantiles, and making the predictions was outlined. The code gave a

detailed picture of the system and how the complete design and structure discussed in the previous stages was implemented. This code was written in python and the dataset used was a csv file. The output obtained was also explored and explained in detail. The results were explained with the screenshots. The predictions obtained were plotted to form a fitting curve. This gave a clear visualisation of the predictions and the accuracy of the model.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

A reliable and efficient Air quality forecasting model was created by the proposed system. This model comprises of a combination of the RNN LSTM trained with the BPTT algorithm and other effective data mining methodologies. Using data mining methods, the pollutants having a relatively high impact on air quality over time and lesser interrelation were identified. This led to the creation of a model which provides high accuracy predictions coupled with lesser number of calculations for efficiency. The prediction model proposed herein is bound to prove to be an authentic way for predictions by environmental departments. The technique used here to evaluate and analyse the factors that impact the study involves various data mining algorithms. It also aids the process of excluding or ignoring those factors which are either irrelevant or do not contribute much to the result. By doing so the model enhances the accuracy levels of the prediction. The proposed atmospheric quality forecasting model established herein can meet the needs of practical application, because it has higher forecasting accuracy. The inference is based on valid data points and efficient algorithms and hence, can be used by organisations of significance in the field of air quality management to restore the atmosphere to its original purity.

7.2 FUTURE WORK

For future work we aim to reduce the loss further and improve the efficiency of the system. There could also be a further extension of the system to accommodate more attributes. The cumulative report generation and providing a set of suggestions for air quality improvement depending on the pollutant levels is another area where there is scope for research and analysis. The scope for some varied outcomes obtained by tweaking more parameters in the model can also be explored.

REFERENCES

- W. Yu, “Spatial co-location pattern mining for location-based services in road networks,” *Expert Systems with Applications*, vol. 46, pp. 324–335, 2016.
- C. Xue, W. Song, L. Qin, Q. Dong, and X. Wen, “A spatiotemporal mining framework for abnormal association patterns in marine environments with a time series of remote sensing images,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 38, pp. 105–114, 2015.
- H. Nguyen, W. Liu, and F. Chen, “Discovering congestion propagation patterns in spatio-temporal traffic data,” *IEEE Transactions on Big Data*, pp. 1–1, 2016.
- Aggarwal and D. Toshniwal, “Spatio-temporal frequent itemset mining on web data,” in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 1160–1165.
- G. Atluri, A. Karpatne, and V. Kumar, “Spatio-temporal data mining: A survey of problems and methods,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 83, 2018.
- Y. Shao, B. Liu, S. Wang, and G. Li, “A novel software defect prediction based on atomic class-association rule mining,” *Expert Systems with Applications*, vol. 114, pp. 237–254, 2018.
- S. A. Aljawarneh, R. Vangipuram, V. K. Puligadda, and J. Vinjamuri, “Gspamine: An approach to discover temporal association patterns and trends in internet of things,” *Future Generation Computer Systems*, vol. 74, pp. 430–443, 2017.
- C.-H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh, “Algorithms for frequent itemset mining: a literature review,” *Artificial Intelligence Review*, pp. 1–19, 2018.
- M. Antonelli, P. Ducange, F. Marcelloni, and A. Segatori, “A novel associative classification model based on a fuzzy frequent pattern mining algorithm,” *Expert Systems with Applications*, vol. 42, no. 4, pp. 2086–2097, 2015.
- N. Aryabarzan, B. Minaei-Bidgoli, and M. Teshnehlab, “negfin: An efficient algorithm for fast mining frequent itemsets,” *Expert Systems with Applications*, vol. 105, pp. 129–143, 2018.

- U. Turdukulov, A. O. Calderon Romero, O. Huisman, and V. Retsios, “Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach,” *International Journal of Geographical Information Science*, vol. 28, no. 10, pp. 2013–2029, 2014.
- L. Szathmary, “Finding frequent closed itemsets with an extended version of the eclat algorithm,” in *Annales Mathematicae et Informaticae*, vol. 48. EKF LICEUM KIADO ESZTERHAZY TER 1, EGER, 3300, HUNGARY, 2018, pp. 75–82.
- Zhang, P. Tian, X. Zhang, Q. Liao, Z. L. Jiang, and X. Wang, “Hasheclat: an efficient frequent itemset algorithm,” *International Journal of Machine Learning and Cybernetics*, pp. 1–14, 2019.
- S. Qin, F. Liu, C. Wang, Y. Song, and J. Qu, “Spatial-temporal analysis and projection of extreme particulate matter (pm₁₀ and pm_{2.5}) levels using association rules: A case study of the jing-jin-ji region, china,” *Atmospheric Environment*, vol. 120, pp. 339–350, 2015.
- S. Skakun, N. Kussul, A. Y. Shelestov, M. Lavreniuk, and O. Kussul, “Efficiency Assessment of Multitemporal C-Band Radarsat-2 Intensity and Landsat-8 Surface Reflectance Satellite Imagery for Crop Classification in Ukraine,” *IEEE J. of Select. Topics in Applied Earth Obser. and Rem. Sens.*, vol. 9, no. 8, pp. 3712–3719, 2016.

sowmya

by Sowmya Sowmya

Submission date: 11-May-2020 07:59AM (UTC+0530)

Submission ID: 1321261893

File name: sowmys.pdf (2.22M)

Word count: 13618

Character count: 72291

ORIGINALITY REPORT

8%

SIMILARITY INDEX

6%

INTERNET SOURCES

6%

PUBLICATIONS

6%

STUDENT PAPERS

PRIMARY SOURCES

1

machinelearningmastery.com

Internet Source

1%

2

keras.io

Internet Source

1%

3

www.codegist.net

Internet Source

1%

4

Submitted to University of Sydney

Student Paper

1%

5

www.scribd.com

Internet Source

<1%

6

Submitted to University of Strathclyde

Student Paper

<1%

7

Min Huang, Tao Zhang, Jingyang Wang, Likun Zhu. "A new air quality forecasting model using data mining and artificial neural network", 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2015

Publication

<1%

8	Dongdong Gui, Sheng-hua Zhong, Zhong Ming. "Chapter 15 Implicit Affective Video Tagging Using Pupillary Response", Springer Science and Business Media LLC, 2018 Publication	<1 %
9	prwatech.in Internet Source	<1 %
10	datascience.foundation Internet Source	<1 %
11	Submitted to University of Oxford Student Paper	<1 %
12	vivelaruta.es Internet Source	<1 %
13	Changhui Yu. "Research of time series air quality data based on exploratory data analysis and representation", 2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics), 2016 Publication	<1 %
14	Submitted to University of Southern California Student Paper	<1 %
15	hadooptutorial.info Internet Source	<1 %
16	Vipul Raheja, K.S. Rajan. "Comparative Study of Association Rule Mining and MiSTIC in	<1 %

Extracting Spatio-temporal Disease Occurrences Patterns", 2012 IEEE 12th International Conference on Data Mining Workshops, 2012

Publication

17

"Table of contents", 2012 IEEE 12th International Conference on Data Mining Workshops, 2012.

Publication

<1 %

18

Submitted to University of Stirling

Student Paper

<1 %

19

"Program", 2018 27th Wireless and Optical Communication Conference (WOCC), 2018

Publication

<1 %

20

Himanshu Singh, Yunis Ahmad Lone. "Deep Neuro-Fuzzy Systems with Python", Springer Science and Business Media LLC, 2020

Publication

<1 %

21

Submitted to University of London External System

Student Paper

<1 %

22

gchester.com

Internet Source

<1 %

23

Dipanjan Sarkar, Raghav Bali, Tushar Sharma. "Practical Machine Learning with Python", Springer Science and Business Media LLC,

<1 %

2018

Publication

-
- | | | |
|-----------|---|----------------|
| 24 | Submitted to Birkbeck College
Student Paper | <1 % |
|-----------|---|----------------|
-
- | | | |
|-----------|---|----------------|
| 25 | Submitted to CVC Nigeria Consortium
Student Paper | <1 % |
|-----------|---|----------------|
-
- | | | |
|-----------|--|----------------|
| 26 | Submitted to Universiti Putra Malaysia
Student Paper | <1 % |
|-----------|--|----------------|
-
- | | | |
|-----------|--|----------------|
| 27 | Haripriya Ayyalasomayajula, Edgar Gabriel, Peggy Lindner, Daniel Price. "Air Quality Simulations Using Big Data Programming Models", 2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService), 2016
Publication | <1 % |
|-----------|--|----------------|
-
- | | | |
|-----------|--|----------------|
| 28 | citeseerx.ist.psu.edu
Internet Source | <1 % |
|-----------|--|----------------|
-
- | | | |
|-----------|--|----------------|
| 29 | es.scribd.com
Internet Source | <1 % |
|-----------|--|----------------|
-
- | | | |
|-----------|---|----------------|
| 30 | Submitted to Heriot-Watt University
Student Paper | <1 % |
|-----------|---|----------------|
-
- | | | |
|-----------|--|----------------|
| 31 | www.ukessays.com
Internet Source | <1 % |
|-----------|--|----------------|
-
- | | | |
|-----------|--|----------------|
| 32 | cs.uwindsor.ca
Internet Source | <1 % |
|-----------|--|----------------|
-

Exclude quotes Off

Exclude bibliography On

Exclude matches < 10 words