

# **DAYANANDA SAGAR UNIVERSITY**

**KUDLU GATE, BANGALORE – 560068**



**Bachelor of Technology  
In  
COMPUTER SCIENCE AND ENGINEERING**

## **Major Project Phase-II Report**

### **PERFORMANCE IMPROVEMENT OF ML ALGORITHMS IN WEATHER FORECASTING**

By

**Suvik Sharma M S - ENG19CS0327**

**Varsha R - ENG19CS0354**

**Vignesh S - ENG19CS0360**

**Yashodha C R - ENG19CS0368**

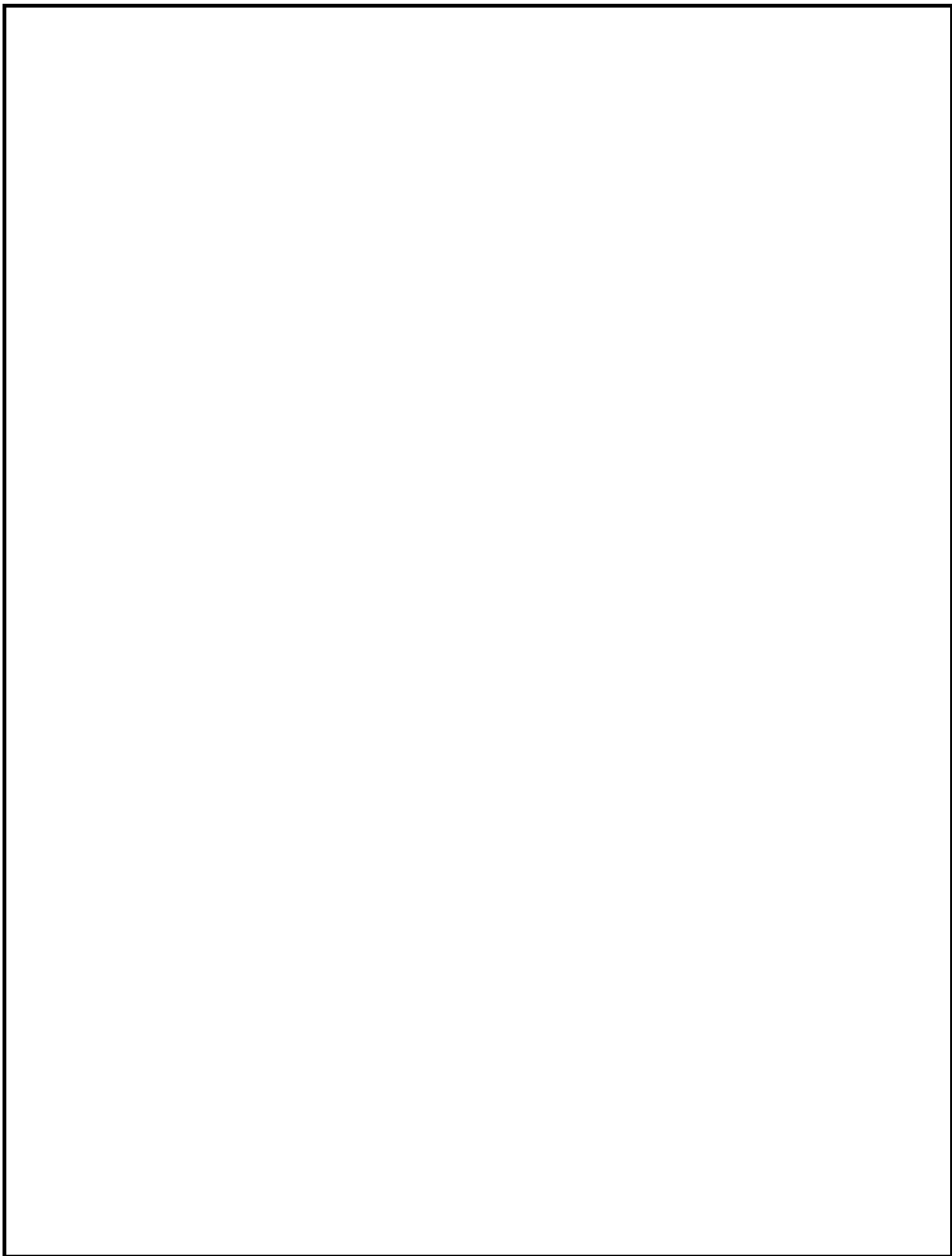
**Under the supervision of**

**Ms. Vedashree L V**

**Assistant Professor, Dept. of CSE**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,  
SCHOOL OF ENGINEERING  
DAYANANDA SAGAR UNIVERSITY,  
BANGALORE**

**2022-2023**



## ACKNOWLEDGEMENT

*It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.*

*First, we take this opportunity to express our sincere gratitude to the School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.*

*We would like to thank **Dr. Udaya Kumar Reddy K R, Dean, School of Engineering & Technology, Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Department Chairman, Computer Science and Engineering, Dayananda Sagar University**, for providing the right academic guidance that made our task possible.*

*We would like to thank our guide **Ms. Vedashree L V, Assistant Professor, Dept. of Computer Science and Engineering, Dayananda Sagar University**, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

*We would like to thank our **Project Coordinator Dr. Meenakshi Malhotra and Dr. Pramod Kumar Naik** as well as all the staff members of Computer Science and Engineering for their support.*

*We are also grateful to our family and friends who provided us with every requirement throughout the course.*

*We would like to thank one and all who directly or indirectly helped us in the Project work.*

## DECLARATION

We, **Suvik Sharma M S (ENG19CS0327), Varsha R (ENG19CS0354) , Vignesh S (ENG19CS0360), Yashodha C R (ENG19CS0358)** are students of eighth semester B. Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Major Project Stage-II titled “**PERFORMANCE IMPROVEMENT OF ML ALGORITHMS IN WEATHER FORECASTING**” has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2022-2023**.

**Student**

**Signature**

**Suvik Sharma M S  
ENG19CS0327:**

**Varsha R  
ENG19CS0354:**

**Vignesh S  
ENG19CS0360:**

**Yashodha C R  
ENG19CS0368:**

**Place : Bangalore**

**Date :**



**DAYANANDA SAGAR UNIVERSITY**

**School of Engineering**  
**Department of Computer Science & Engineering**

Kudlu Gate, Bangalore – 560068  
Karnataka, India

**CERTIFICATE**

This is to certify that the Phase-II project work titled “**Performance Improvement of ML Algorithms in Weather Forecasting**” is carried out by **Suvik Sharma M S (ENG19CS0327)**, **Varsha R (ENG19CS0354)**, **Vignesh S (ENG19SC0360)**, **Yashodha C R (ENG19CS0368)** bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, and Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2022-2023**.

Prof. Vedashree L V  
Assistant Professor  
Dept. of CSE  
School of Engineering  
Dayananda Sagar University

Date:-

**Name of the Examiner**

**Signature of Examiner**

- 1.
- 2.



**DAYANANDA SAGAR UNIVERSITY**

**School of Engineering**  
**Department of Computer Science & Engineering**

Kudlu Gate, Bangalore – 560068  
Karnataka, India

**CERTIFICATE**

This is to certify that the Phase-II project work titled “**Performance Improvement of ML Algorithms in Weather Forecasting**” is carried out by **Suvik Sharma M S (ENG19CS0327)**, **Varsha R (ENG19CS0354)**, **Vignesh S (ENG19SC0360)**, **Yashodha C R (ENG19CS0368)** bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, and Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2022-2023**.

Prof . Vedashree L V	Dr. Girisha G S	Dr. Udaya Kumar Reddy K R
Assistant Professor Dept. of CSE School of Engineering Dayananda Sagar University	Chairman CSE School of Engineering Dayananda Sagar University	Dean School of Engineering Dayananda Sagar University
Date:-	Date :-	Date :-

**Name of the Examiner**

**Signature of Examiner**

- 1.
- 2.

# TABLE OF CONTENTS

<b>LIST OF ABBREVIATIONS.....</b>	
<b>LIST OF FIGURES .....</b>	
<b>ABSTRACT.....</b>	
<b>CHAPTER 1 INTRODUCTION.....</b>	<b>1</b>
1.1. HOW IS IT USEFUL? .....	1
1.2. FIGURES & STATISTICS.....	2
1.3 SCOPE .....	5
1.3.1 APPLICATIONS OF WEATHER FORECASTING.....	5
1.3.2 ENVIRONMENTAL IMPACT .....	6
<b>CHAPTER 2 PROBLEM DEFINITION .....</b>	<b>7</b>
<b>CHAPTER 3 LITERATURE SURVEY.....</b>	<b>8</b>
<b>CHAPTER 4 PROJECT DESCRIPTION.....</b>	<b>10</b>
4.1. PROPOSED DESIGN .....	10
4.2. ASSUMPTIONS AND DEPENDENCIES.....	13
<b>CHAPTER 5 REQUIREMENTS.....</b>	<b>14</b>
5.1. FUNCTIONAL REQUIREMENTS.....	14
5.2. NON-FUNCTIONAL REQUIREMENTS .....	14
5.3 SOFTWARE / SYSTEM REQUIREMENTS.....	15
5.3.1 HARDWARE REQUIREMENTS.....	15
5.3.2 SOFTWARE REQUIREMENTS.....	15
<b>CHAPTER 6 METHODOLOGY.....</b>	<b>16</b>
<b>CHAPTER 7 EXPERIMENTATION.....</b>	<b>20</b>
<b>CHAPTER 8 TESTING AND RESULTS .....</b>	<b>30-36</b>
<b>REFERENCES... ..</b>	<b>37</b>

## LIST OF ABBREVIATIONS

ML	Machine Learning
SVM	Support Vector Machine
RNN-LSTM	Recurrent Neural Network - Long Short Term Memory
NWP	Numerical Weather Prediction
MNB	Multinomial Naive Bayes
iJade	intelligent Java Development Environment
LR	Logistic Regression
ANN	Artificial Neural Network
RMS	Root Mean Square
CPU	Central Processing Unit
RAM	Random Access Memory
OS	Operating System
CSV	Comma Separated Values
XLSX	Excel File Format
LMS	Least Mean Squared
RMS	Residual Root Mean Squares
OLS	Ordinary Least Squares



## LIST OF FIGURES

Fig. No.	Description of the figure	Page No.
4.11	Proposed Design	12
6.1	Methodology	16
7.1	Importing Necessary Packages	21
7.2	All the features in the dataset	22
7.3	Checking the columns in the data frame	22
7.4	Modelling the algorithm into dataset	23
7.5	Showcasing only implemented features	23
7.6	Splitting of data into test and train dataset	24
7.8	Implementing Decision Tree Regression for Prediction of mean and variance score	25
7.9	Implementing Random Forest Regression and calculating mean and Variance	25
7.10	Calculating R-Squared Score for Linear Regression	26
7.11	Calculating R-squared Score for Decision Tree	26
7.12	Calculating R-squared Score for Random Forest Regression	26
7.13	Gaussian Naïve Bayes	27
7.14	Multinomial Naïve Bayes	27
7.15	Complement Naïve Bayes	28
7.16	Bernoullis Naïve Bayes	28
7.17	Categorical Naïve Bayes	29
8.1	Importing Packages	31
8.2	Acid Rain Dataset	31
8.3	Drop of Columns in which has null values	31
8.5	Splitting of testing and training data	32
8.6	Implementing KNN classifier	33
8.7	Calculating Precision score	33
8.8	Calculating F1 Score	34
8.9	Balanced Accuracy Score	34
8.10	Feature Selection and visualization of the dataset	35
8.11	Calculating R-Squared Score of Linear Regression	35

8.12	Calculating R-Squared Score of Decision Tree	36
8.13	Calculating R-Squared Score of Random Forest	36

# ABSTRACT

Weather plays an important role in our everyday life. The climate is changing at a drastic rate nowadays, which makes the old prediction method less effective and hectic. We plan to approach the everyday unpredictable weather with a model that helps us forecast weather before-hand with available data. A comparatively reliable application model that forecasts weather better than the conventional method of the same.

With the rise in technology and data driven industries, the need for an accurate & precise weather forecasting system is a highly challengeable task. The traditional method of forecasting weather just by looking up to the sky proves to be unreliable most of the time, but with loads of data to use and put into a machine learning model makes it easier to forecast the weather based on certain parameters and conditions. Dealing with extraneous data can be an overwhelming and tedious task, and in expertise might lead to wrong weather forecasts and predictions which in turn affects the most vital sectors the industry, mining, construction & logistics.

In this project, we present to you a model that will be used for weather forecasting with better and higher accuracy. By using methods and techniques used to increase the model's performance, the efficiency of the forecasting system is drastically increased. The proposed model will be to use various techniques used to increase the performance of the model

# TABLE OF CONTENTS

<b>LIST OF ABBREVIATIONS.....</b>	<b>i</b>
<b>LIST OF FIGURES .....</b>	<b>ii</b>
<b>ABSTRACT.....</b>	<b>iii</b>
<b>CHAPTER 1 INTRODUCTION.....</b>	<b>1</b>
1.1. HOW IS IT USEFUL? .....	1
1.2. FIGURES & STATISTICS.....	2
1.3 SCOPE .....	5
1.3.1 APPLICATIONS OF WEATHER FORECASTING.....	5
1.3.2 ENVIRONMENTAL IMPACT .....	6
<b>CHAPTER 2 PROBLEM DEFINITION .....</b>	<b>7</b>
<b>CHAPTER 3 LITERATURE SURVEY.....</b>	<b>8</b>
<b>CHAPTER 4 PROJECT DESCRIPTION.....</b>	<b>10</b>
4.1. PROPOSED DESIGN .....	10
4.2. ASSUMPTIONS AND DEPENDENCIES.....	13
<b>CHAPTER 5 REQUIREMENTS.....</b>	<b>14</b>
5.1. FUNCTIONAL REQUIREMENTS.....	14
5.2. NON-FUNCTIONAL REQUIREMENTS .....	14
5.3 SOFTWARE / SYSTEM REQUIREMENTS.....	15
5.3.1 HARDWARE REQUIREMENTS.....	15
5.3.2 SOFTWARE REQUIREMENTS.....	15
<b>CHAPTER 6 METHODOLOGY.....</b>	<b>16</b>
<b>CHAPTER 7 EXPERIMENTATION.....</b>	<b>20</b>
<b>CHAPTER 8 TESTING AND RESULTS .....</b>	<b>30-36</b>
<b>REFERENCES... ..</b>	<b>37</b>

## LIST OF ABBREVIATIONS

ML	Machine Learning
SVM	Support Vector Machine
RNN-LSTM	Recurrent Neural Network - Long Short Term Memory
NWP	Numerical Weather Prediction
MNB	Multinomial Naive Bayes
iJade	intelligent Java Development Environment
LR	Logistic Regression
ANN	Artificial Neural Network
RMS	Root Mean Square
CPU	Central Processing Unit
RAM	Random Access Memory
OS	Operating System
CSV	Comma Separated Values
XLSX	Excel File Format
LMS	Least Mean Squared
RMS	Residual Root Mean Squares
OLS	Ordinary Least Squares

## LIST OF FIGURES

Fig. No.	Description of the figure	Page No.
4.11	Proposed Design	12
6.1	Methodology	16
7.1	Importing Necessary Packages	21
7.2	All the features in the dataset	22
7.3	Checking the columns in the data frame	22
7.4	Modelling the algorithm into dataset	23
7.5	Showcasing only implemented features	23
7.6	Splitting of data into test and train dataset	24
7.8	Implementing Decision Tree Regression for Prediction of mean and variance score	25
7.9	Implementing Random Forest Regression and calculating mean and Variance	25
7.10	Calculating R-Squared Score for Linear Regression	26
7.11	Calculating R-squared Score for Decision Tree	26
7.12	Calculating R-squared Score for Random Forest Regression	26
7.13	Gaussian Naïve Bayes	27
7.14	Multinomial Naïve Bayes	27
7.15	Complement Naïve Bayes	28
7.16	Bernoullis Naïve Bayes	28
7.17	Categorical Naïve Bayes	29
8.1	Importing Packages	31
8.2	Acid Rain Dataset	31
8.3	Drop of Columns in which has null values	31
8.5	Splitting of testing and training data	32
8.6	Implementing KNN classifier	33
8.7	Calculating Precision score	33
8.8	Calculating F1 Score	34
8.9	Balanced Accuracy Score	34
8.10	Feature Selection and visualization of the dataset	35
8.11	Calculating R-Squared Score of Linear Regression	35

8.12	Calculating R-Squared Score of Decision Tree	36
8.13	Calculating R-Squared Score of Random Forest	36

## **ABSTRACT**

Weather plays an important role in our everyday life. The climate is changing at a drastic rate nowadays, which makes the old prediction method less effective and hectic. We plan to approach the everyday unpredictable weather with a model that helps us forecast weather before-hand with available data. A comparatively reliable application model that forecasts weather better than the conventional method of the same.

With the rise in technology and data driven industries, the need for an accurate & precise weather forecasting system is a highly challengeable task. The traditional method of forecasting weather just by looking up to the sky proves to be unreliable most of the time, but with loads of data to use and put into a machine learning model makes it easier to forecast the weather based on certain parameters and conditions. Dealing with extraneous data can be an overwhelming and tedious task, and in expertise might lead to wrong weather forecasts and predictions which in turn affects the most vital sectors the industry, mining, construction & logistics.

In this project, we present to you a model that will be used for weather forecasting with better and higher accuracy. By using methods and techniques used to increase the model's performance, the efficiency of the forecasting system is drastically increased. The proposed model will be to use various techniques used to increase the performance of the model



# CHAPTER 1 INTRODUCTION

The first attempt to predict weather numerically has always required a large workforce. With the development of powerful machines, and modeling techniques, weather prediction has made a comeback. Machine learning is one of the steps that the world is taking towards a large mission of the technology world. Since the input weather data is used to forecast the future weather at a geographical location, it has to be done with extreme caution and precision.

Machine learning helps with data analytics, and prediction for applications such as Weather forecasting, Stock market prediction, Natural Language Processing, Image recognition, Human action recognition and so on.

To overcome the traditional problems, we can combine classifiers, use a dataset that is better in quality, a model that can deal with extraneous data, feature selection, feature engineering and importantly algorithm tuning.

By using datasets that are authentic, precise & verifiable, we can use machine learning models such as Ridge regression & RNN, we aim to build a model that is accurate, and performance enhanced for weather forecasting.

## 1.1. HOW IS IT USEFUL?

Many primary industry sectors depend immensely on weather forecasting for their daily conventional working. Industries such as Mining, Travel, Logistics, Farming, Construction and Turf-related sports such as Football, Rugby, Hockey, Cricket, Track events, etc. Machine learning provides the capability to the system that learns and improves from experience without being programmed dynamically.

## 1.2. FIGURES & STATISTICS

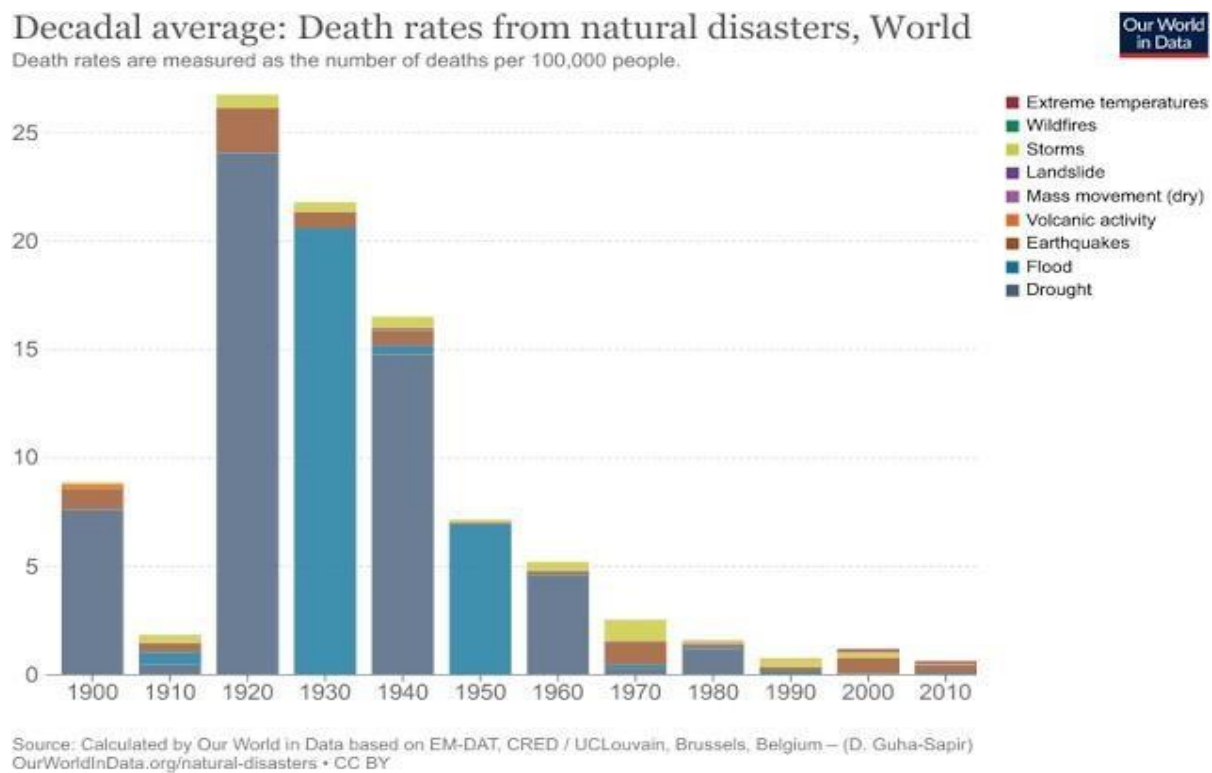


Fig 1.2.1 Estimated death rates from natural disasters

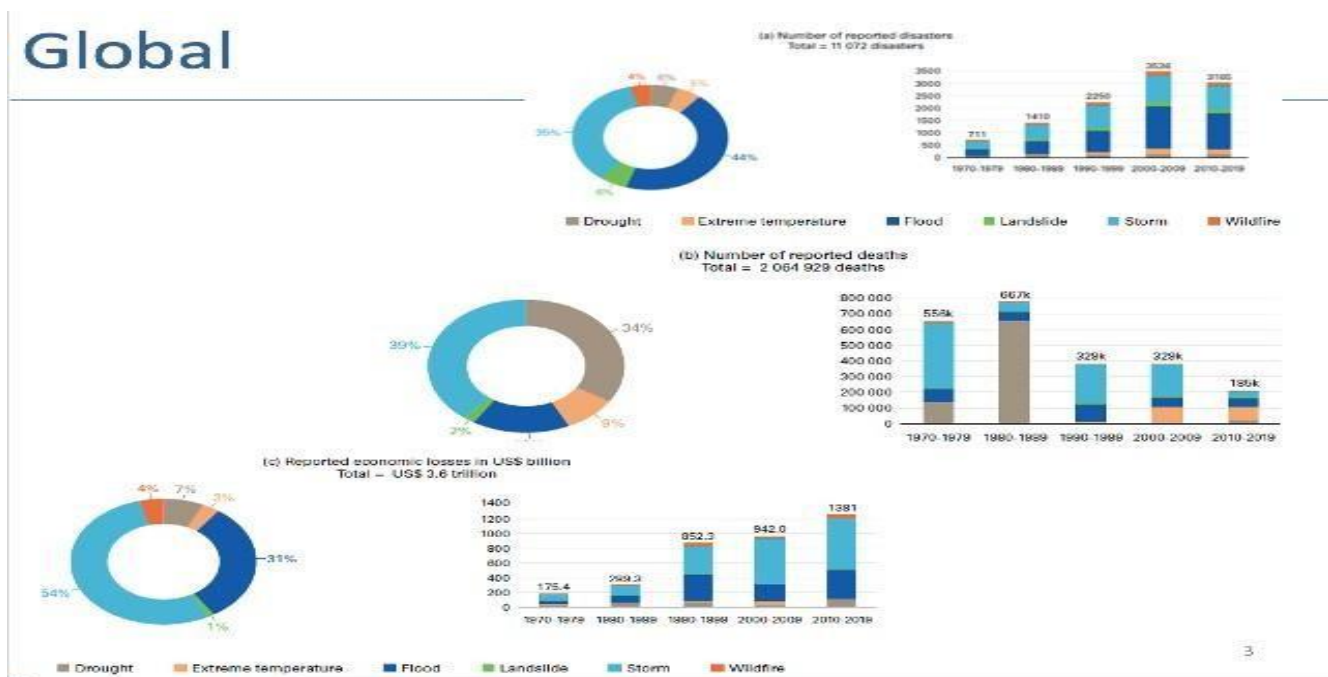
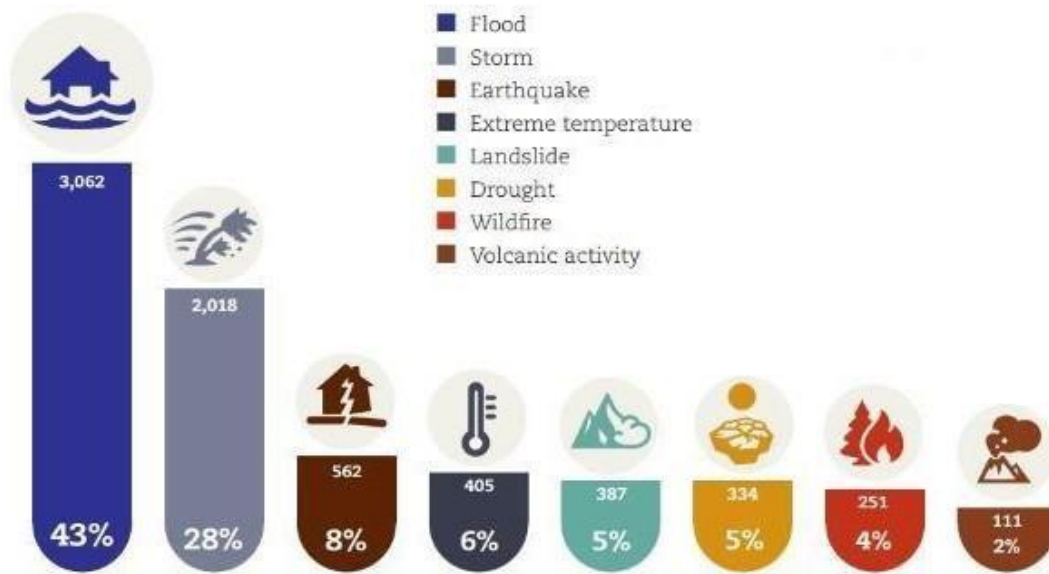


Fig 1.2.2 Global report of natural disasters from 1970 to 2019



Source: UN/CRED

Fig 1.2.3 distribution of types of natural disasters from around the world in the last 5 years

Disaster Type	Events	Events/Year	Percent Frequency	Total Costs	Percent of Total Costs	Cost/Event	Cost/Year	Deaths	Deaths/Year
Drought	30	0.7	8.9%	\$309.4B <sup>CI</sup>	13.5%	\$10.3B	\$7.2B	4,256 <sup>†</sup>	99 <sup>†</sup>
Flooding	37	0.9	10.9%	\$174.9B <sup>CI</sup>	7.6%	\$4.7B	\$4.1B	676	16
Freeze	9	0.2	2.7%	\$34.4B <sup>CI</sup>	1.5%	\$3.8B	\$0.8B	162	4
Severe Storm	162	3.8	47.9%	\$374.1B <sup>CI</sup>	16.3%	\$2.3B	\$8.7B	1,982	46
Tropical Cyclone	59	1.4	17.5%	\$1,194.4B <sup>†CI</sup>	52.0% <sup>†</sup>	\$21.0B <sup>†</sup>	\$27.8B <sup>†</sup>	6,864	160
Wildfire	21	0.5	6.2%	\$126.9B <sup>†CI</sup>	5.5% <sup>†</sup>	\$6.3B <sup>†</sup>	\$3.0B <sup>†</sup>	435	10
Winter Storm	20	0.5	5.9%	\$83.4B <sup>CI</sup>	3.6%	\$4.2B	\$1.9B	1,314	31
All Disasters	338	7.9	100.0%	\$2,297.5B <sup>†CI</sup>	100.0% <sup>†</sup>	\$6.9B <sup>†</sup>	\$53.4B <sup>†</sup>	15,689	365

Fig 1.2.4 Estimated events, total costs, cost per event, deaths &amp; deaths per year tabular statistical data of natural disasters

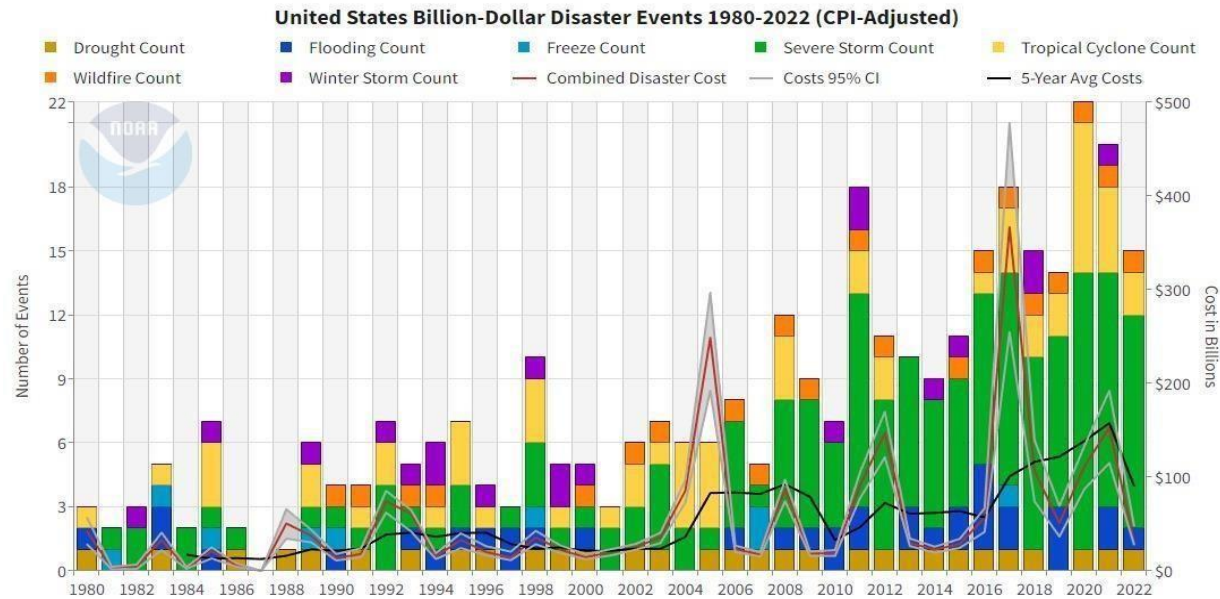


Fig 1.2.5 Estimated cost from disasters occurred in the USA from 1980 to 2022

(a)	Disaster type	Year	Country	Deaths
1	Drought	1983	Ethiopia	300 000
2	Storm ( <i>Bhola</i> )	1970	Bangladesh	300 000
3	Drought	1983	Sudan	150 000
4	Storm ( <i>Gorky</i> )	1991	Bangladesh	138 866
5	Storm ( <i>Nargis</i> )	2008	Myanmar	138 366
6	Drought	1973	Ethiopia	100 000
7	Drought	1981	Mozambique	100 000
8	Extreme temperature	2010	Russian Federation	55 736
9	Flood	1999	Bolivarian Republic of Venezuela	30 000
10	Flood	1974	Bangladesh	28 700
(b)	Disaster type	Year	Country	Economic losses (in US\$ billion)
1	Storm ( <i>Katrina</i> )	2005	United States	163.61
2	Storm ( <i>Harvey</i> )	2017	United States	96.94
3	Storm ( <i>Maria</i> )	2017	United States	69.39
4	Storm ( <i>Irma</i> )	2017	United States	58.16
5	Storm ( <i>Sandy</i> )	2012	United States	54.47
6	Storm ( <i>Andrew</i> )	1992	United States	48.27
7	Flood	1998	China	47.02
8	Flood	2011	Thailand	45.46
9	Storm ( <i>Ike</i> )	2008	United States	35.63
10	Flood	1995	Democratic People's Republic of Korea	25.17

Fig 1.2.6 Top 10 disasters ranked according to reported deaths and economic losses

### **1.3. SCOPE**

The prediction / forecasts offered by the model is backed by machine learning techniques and procedural steps taken to carefully preprocess the numerical data. With exploration, we have found out a few ways to deal with extraneous data, missing data and null values in the dataset. The objective is to clean the given data and use the parameters that mostly affect the performance and accuracy of the model. The follow-through action would be to use various methods / algorithms to increase the performance and accuracy of the weather forecasting model.

Potential end users of a weather forecasting system are:

- General Public
- Shipping Mercantile & Naval
- Port officials
- Defense services
- Agriculture
- Flood forecasting

#### **1.3.1. APPLICATIONS OF WEATHER FORECASTING**

The importance of accurate weather forecasts cannot be over emphasized as the needs for them are always craved in a virtual aspect of life. Few of the fields that are end users of Weather Forecasting Services are:

- Severe weather alerts and advisories
- Air traffic
- Marine
- Agriculture
- Utility companies
- Military applications
- Logistics

### 1.3.2. ENVIRONMENTAL IMPACT

Weather forecasts are made by collecting quantitative data about the current state of the atmosphere and scientific understanding of the environment processes. The chaotic nature of the atmosphere, and an incomplete understanding of the atmosphere results in inaccurate forecasting and prediction of daily weather.

There are a variety of end users to weather forecasting. Weather warnings are important forecasts because they are used to protect life and property. Forecasts based on temperature, precipitation are important to agriculture and its entire sector. This in turn affects the logistics side of the commodity markets.

The traditional method of forecasting the weather at a geographic location is simply inefficient and not very reliable, with the help of our model we would be able to forecast weather at a location that is way more accurate, reliable and performance enhanced just by a glance of our system.

A weather forecasting system is also very helpful in the following domains:

- Agricultural sector.
- Manufacturing activities with outdoor machinery planted for.
- Industry level forecasting.
- Government funded constructions & Commercial construction work.
- Mining of minerals.
- Aviation & Military services.
- Commercial flight takeoffs & landing.

## **CHAPTER 2 PROBLEM DEFINITION**

Weather forecasting is the application of scientific techniques and technology to predict the conditions of the atmosphere at a certain location and time. A lot of the primary sector's activities depend on weather forecasting for production & entertainment i.e., farming, mining, construction, logistics and turf - related sports.

To overcome difficulties in prediction, improved and reliable weather prediction systems are required, and the application uses historical weather data to forecast the weather. Machine learning algorithms such as Linear Regression, Decision Tree, Random Forest Regression, KNN, Naïve Bayes and its epitomes are used for the same.



## CHAPTER 3 LITERATURE REVIEW

Title: Dires Negash Fente, Dheeraj Kumar Singh, ‘Weather Forecasting Using Artificial Neural Network’, IEEE, International conference 2018.

Artificial neural networks are inspired by biological neuron models. In artificial neural networks, numbers of highly nonlinear neurons are interconnected so as to form a network. This paper summarizes the use of Ann and combines the RNN network which provides relevance towards the historical weather data. This paper demonstrates the use of RNN algorithm and provides better accuracy on the whole.

Title: Pradeep Hewage, Marcello Trovati, Ella Pereira and Ardhendu Behera ‘Deep learning-based effective fine-grained weather forecasting model’, Elsevier Journal 2020.

NWP numerical weather prediction is focused here; the NWP these models define a coordinate system, which divides the earth into a 3-dimensional grid. For short term weather prediction MIMO is taken into authorization. . MISO-LSTM, MISO-TCN, MIMO-LSTM, MIMO-TCN are taken into comparison. In this article, it demonstrates that the proposed lightweight deep model can be utilized for weather forecasting up to 12 h for 10 surface weather parameters. The model outperformed the state-of-the-art WRF model for up to 12 h.

Title: Kartika Purwandari, Join W. C.Sigalingging , Tjeng Wawan Cenggoro , Bens Pardamean, ‘Multi-class Weather Forecasting from Twitter Using Machine Learning Approaches’, Elsevier , Conference , 4 sep 2020

This paper mainly comprises the big data that mainly lays its foundation on (SVM), Multinomial Naive Bayes (MNB), and Logistic Regression (LR) method. The experimental results show that SVM substantially outperforms various other machine learning algorithms for the task of text. Here substantially the SVM

Title: Raymond Lee and James Liu, ‘ iJADE WeatherMAN: A Weather Forecasting SystemUsing Intelligent Multi Agent-Based Fuzzy Neural Network’ , IEEE, Journal , August 2020.

Java Agent Development Environment (iJADE), to provide an integrated and intelligent agent-basedplatform in the e-commerce environment. In addition to the facilities found in contemporary agent development platforms, which focus on the autonomy and mobility of the multi agents. Here the model is compared with the ANN model this ijade model gathers the data from the mobile networks and gets access data for prediction.



Title: K Geetha Rani, Dr D C Joy Winnie, S Sufiyah Begum, S Nirosha ‘Designing a model for weather forecasting using machine learning’ , IEEE Journal, August 2020

The technique used for weather forecasting is an advanced machine learning model known as Hidden Markov Model. In this research paper, there exists a recurring function that acts in every step of data processing, hence forecasting weather on a mathematical model. HMM can also be used for time series data reading

Title: Bogdan Bochenek, Zbigniew Ustrnul, ‘Machine learning in weather prediction and climate analyses – applications and perspectives’, MDPI Atmosphere Journal, January 2022

The techniques used are SVM, ANN & RNN. The author discusses comparing the prediction models of weather data with a windowed duration of 8 weeks. The author also infers that calculating RMS error, and implementing RNN is implied to be a better model for weather forecasting

## CHAPTER 4 PROJECT DESCRIPTION

Our aim is to improve an already existing model of weather forecasting. The objective of a project such as this is to improve performance than the conventional method of the system itself. The workflow of numerical data demonstrates to us the importance of development and improvement of machine learning, and its applications in everyday lives. With the introduction of the machine learning concept, there is a great ease for data analysis and prediction.

Climate anticipating stations are frameworks that permit determining of day by day, week after week or month to month climate conditions. To foresee the climate of a specific area over a fixed term is a dull procedure. The early methodologies couldn't deal with unstructured information and missing information and subsequently came about in misjudged yield.

Our goal is to build a model that predicts the weather with given appropriate input data from a dataset. An application that is easy to use, has a simple GUI that represents the weather report / forecasting in a more visualized and appealing manner. With sufficient data points, the weather prediction can be made more serviceable & accurate.

### 4.1 Proposed Design

This design involves in combining the best practices of preprocessing data and making the best use of available features, libraries & machine learning model i.e., Ridge Regression to be able to perform efficient weather forecasting rather considering and comparing the previous approach Linear regression, Random forest , Decision tree not just stopping up to predicting the weather .Acid rain prediction has also been integrated with the KNN and QDA approach a User interactive interface has been developed in order to showcase the best and accurate data.

Computing the accuracy, performance & precision of the already existing weather forecasting model & improving its performance with an eminent result of outcome.

To increase the accuracy, performance & precision, there are multiple techniques that are considered. Methods include, increasing number of parameters, normalizing the existing data from the obtained dataset, increasing the quality & quantity of data, handling extraneous and erroneous input numerical data.

The addition of a classifier or addition of a layered network to computing numeric input weather data, enhances the performance of the weather forecasting system drastically.

The most common approach to enhance the performance of any machine learning model is:

- Add more data
- Treat missing and outlier values
- Feature engineering
- Feature selection
- Multiple algorithms, layers or classifiers
- Algorithm tuning
- Cross validation

The overall objective of the project is to bring up the best model for the purpose of weather forecasting and enhance its performance for a better, accurate & precise model.

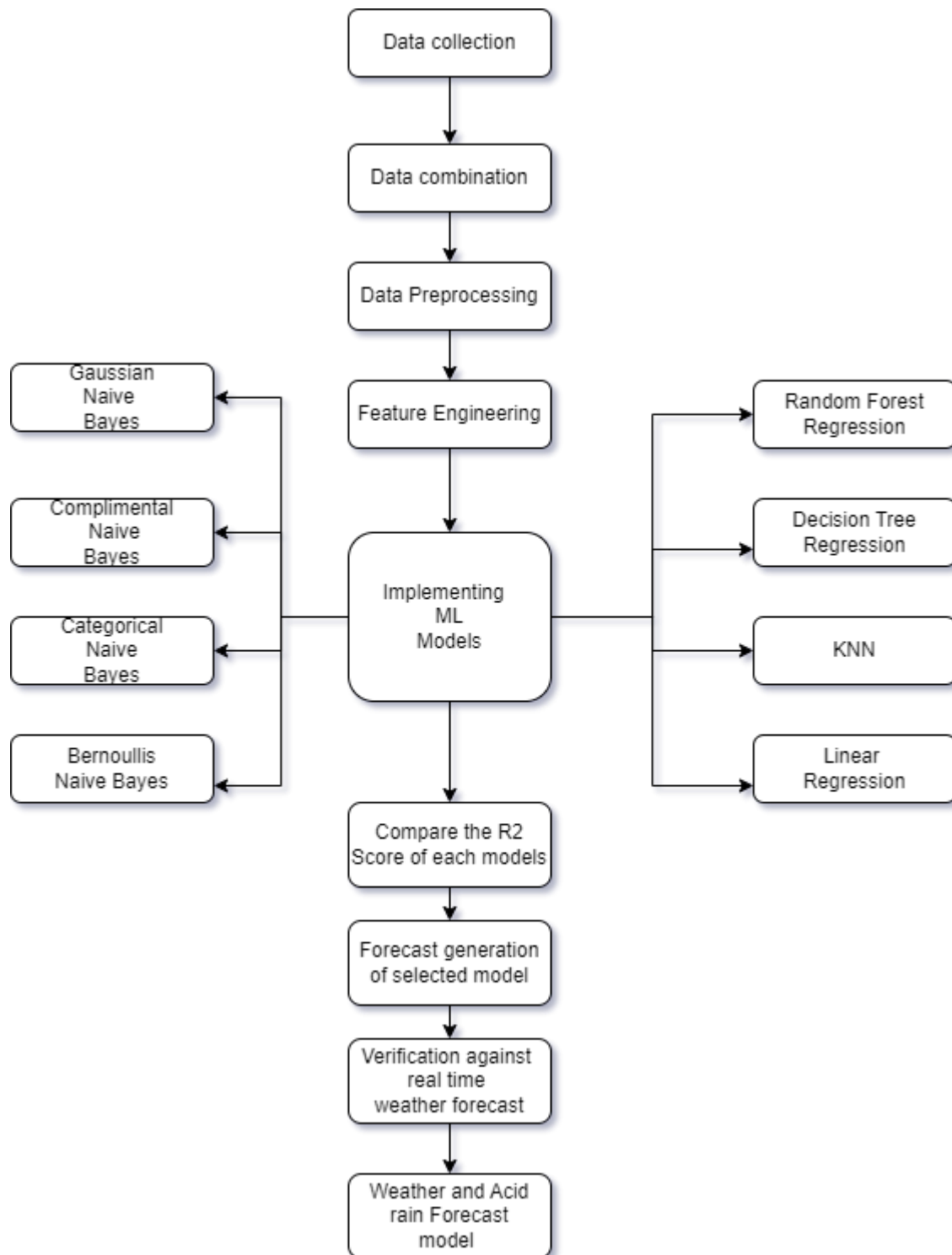


Fig 4.1.1

Design and workflow of weather forecasting model

## 4.2 Assumptions and Dependencies

- We train our numerically obtained weather data with our model.
- In the process of data collection from various different sources, we also need to make sure the authenticity of the numeric data obtained from the right source.
- Weather data released by the Airport, Data collected from a trusted weather station, are examples of authentic sources of numeric weather data collection.
- The data values range in the obtained dataset play a vital role in using it to forecast a weather report.
- The parameters in the dataset play an integral role, more the parameters to forecast weather from, the better it is for the model to be trained and then tested.

## **CHAPTER 5 REQUIREMENTS**

### **5.1 Functional Requirements**

- The system requires a clean and complete dataset of weather inputs.
- Preprocessing of data from dataset.
- Feature extraction of a particular place to predict weather better.
- Machine learning model & algorithm.
- Predicting weather using supervised learning

### **5.2 Non-functional requirements**

- Correctness: The model should perform well under stress conditions & give appropriate output
- Performance of the model: The model must be able to deal with extraneous data
- Portability: Model should be accessible on different systems rather than the source system alone.
- Reliability: The model should be able to service the user without failing under any condition.
- Availability: The model should be available at any point of time to predict the weather.
- Efficiency: The model should be able to predict weather with efficient usage of time & data.
- Maintainability: Code should be easy to implement, use & modify.

## **5.3 Software / System requirements**

### **5.3.1 Hardware requirements**

- 1) Intel i5 CPU / AMD Ryzen 5 or i7 recommended
- 2) 8GB RAM or 16GB recommended.

### **5.3.2 Software requirements**

- 1) Windows 10 OS/ Unix/ Mac OS
- 2) Python 3 or Python 2 version
- 3) Anaconda Prompt
- 4) Jupyter Notebook

## CHAPTER 6 METHODOLOGY

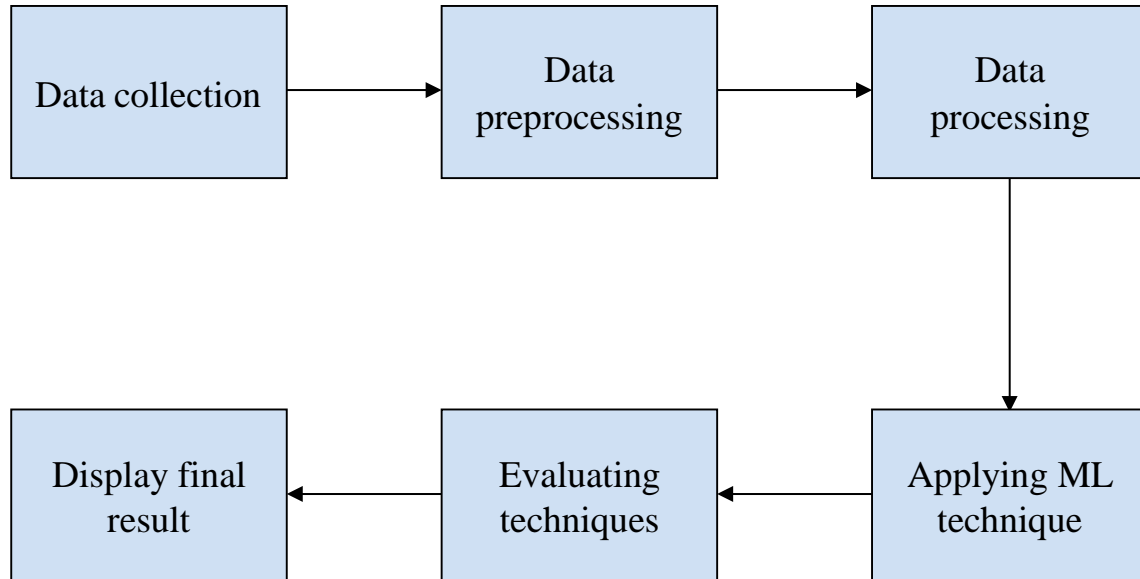


Fig 6.1

Data Flow

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. Steps Invoked in Data Preprocessing are -

Data Cleaning (Ignore the tuples, Fill the Missing values, Regression, Clustering), Data Transformation (Normalization, Attribute Selection, Concept Hierarchy Generation, Discretization), Data Reduction (Data Cube Aggregation, Attribute Subset Selection, Numerosity Reduction, Dimensionality Reduction).

Numpy is used to do mathematical operations on matrices using python. Matrices such as table data and whatsoever we read on to the program. Here in this file, we are using a CSV (comma-separated values) and XLSX (Excel) file.

The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, It is used to download files from the computer to the program or save files from the program to the systems folder from any OS.

Panda - Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.



Sklearn is a free machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

### **Linear Regression**

Linear regression analysis is a statistical method used to estimate the value of one variable based on another variable's value. The variable that is being forecasted is called the dependent variable, while the variable used to make predictions is known as the independent variable

For any type of regression machine learning model, the first step is to standardize the variables by subtracting their means and dividing by their standard deviations.

### **Decision Tree**

The decision tree is a supervised learning technique that can handle both regression and classification tasks without any parameters. It has a hierarchical arrangement with a root node, internal nodes, branches, and leaf nodes.

### **Random Forest Regression**

Random Forest utilizes ensemble learning to tackle complex problems by aggregating the outputs of multiple weaker classifiers. At the core of this system lie three fundamental parts: the root node, decision node, and leaf node. The root node serves as the starting point for dividing the population, and each subsequent split creates a decision node.

Ultimately, the splitting process culminates in a leaf node, signifying the conclusion of the process.

### **KNN**

Leveraging the proximity of data points, K-nearest neighbours (KNN or k-NN) performs supervised learning classification, and as a non-parametric method, it predicts the group classification or outcome for an individual data point. The prevalent method for computing distance is Euclidean distance in KNN.

The KNN algorithm falls into the category of "lazy learning" models, which implies that it only preserves a training dataset and does not undergo a training stage. All computations are carried out during prediction or classification, and the technique is referred to as instance-based or memory-based learning due to its reliance on memory to store all the training data.

In order to identify the data points that are closest to a given query point, computing the distance between the query point and other data points is crucial. The resulting distance metrics play a pivotal role in forming decision boundaries that demarcate query points into distinct regions. Decision boundaries are frequently depicted using Voronoi diagrams.

## **Bayes Theorem**

### **Gaussian Naïve Bayes**

The probabilistic approach and Gaussian distribution-based classification technique, known as Gaussian Naive Bayes (GNB), is employed in Machine Learning (ML). It operates on the premise that each parameter (also called predictors or features) has an independent ability to predict the output variable. By combining the predictions of all the parameters, the model provides the ultimate prediction, which reflects the probability of the dependent variable being classified in each group. The group with the highest probability is selected as the final classification.

### **Compliment Naïve Bayes**

Complement Naive Bayes is an updated version of the Multinomial Naive Bayes algorithm that improves its ability to manage unstable data. The Multinomial Naive Bayes algorithm faces challenges in processing imbalanced datasets, where one class has significantly more instances than others, resulting in an uneven distribution of examples. The analysis of such data can be problematic as models may over fit to prioritize the class with a larger number of instances.

### **Categorical Naïve Bayes**

The Naive Bayes classifier that is based on categories is ideal for classifying discrete features with categorical distributions. The categories for each feature are obtained from a distribution that is categorical in nature.

### **Bernoulli's Naïve Bayes**

Bernoulli Naive Bayes is a variant of the Naive Bayes algorithm that is limited to binary values. The algorithm is commonly used to determine if a particular word appears in a document or not. This approach simplifies the model and can be useful in situations where the frequency of word occurrences is less important. Instead, the focus is on counting the occurrences of binary features, specifically whether a word is present in the document or not.

Next step is to conduct the Exploratory Data Analysis. It is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

Sequentially, we arrange the core dataset into its chronological order and start training our model for machine learning and we also make sure that we have the right data types to process our data before ingestion into the machine learning model.

When we fit a model, we are asking it to learn a set of coefficients that best fit over the training distribution as well as hope to generalize on test data points as well. Learning those coefficients can be done in various ways and to reduce the error in coefficients as well. The reduction techniques are Least Mean Squared (LMS) & Residual Sum of Squares. (RMS).

## CHAPTER 7 EXPERIMENTATION

**Data cleaning:** the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

### **Data Preprocessing:**

Acquire the dataset

Import all the crucial libraries

Import the dataset

Identifying and handling the missing values

Encoding the categorical data

Splitting the dataset

Feature scaling

### **Data visualization:**

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Additionally, it provides an excellent way for employees or business owners to present data to non-technical audiences without confusion.

**1. Acquire:** Obtain the data, whether from a file on a disk or a source over a network.

Data can be collected from many resources such as books, files and digital documents. This is the beginning and fundamental step of data visualization.

**2. Parse:** Provide some structure for the data's meaning, and order it into categories.

You may have collected immense data, but it is necessary to restructure the collected data. This structure will make it easier to convey to others what data you have by format, tags, names, and indices.

### 3. **Filter:** Remove all but the data of interest.

Not all data is useful. Filter out the data that cannot serve your goal. If you are focusing on the data of a specific period, remove the data of other periods.

### 4. **Mine:** Apply methods from statistics or data mining as a way to discern patterns or place the data in mathematical context.

Data visualization is to help viewers seek for insights that may not be gained from raw data or statistics. This step helps get a basic understanding of the data that is significant for the whole process.

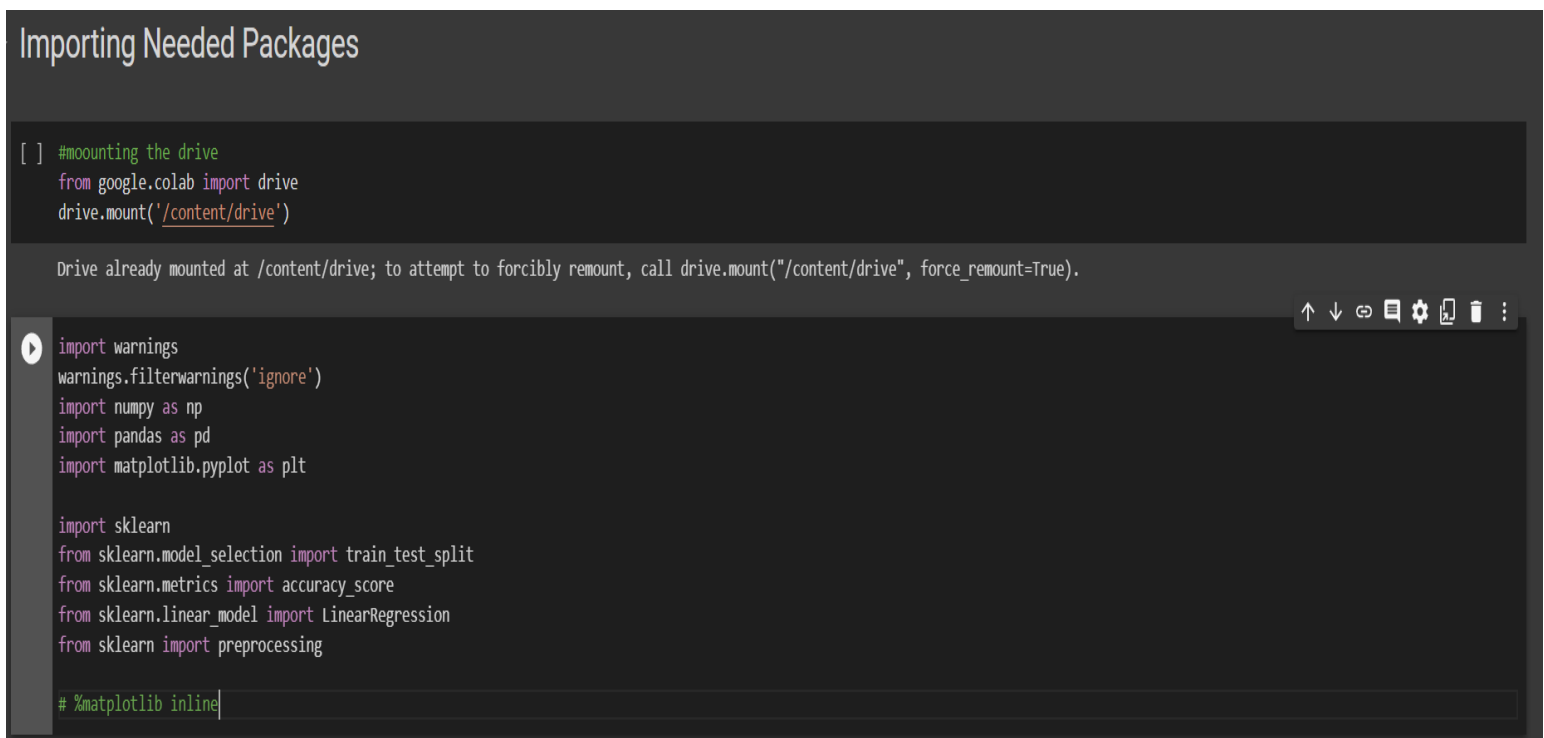
### 5. **Represent:** Choose a basic visual model, such as a bar graph, list, or tree.

Various visual models are available. You need to select the suitable type that best suits your needs. Edraw data visualization software helps you produce over 200 kinds of visuals instantly.

### 6. **Refine:** Improve the basic representation to make it clearer and more visually engaging.

Polish your work according to some basic color and graphic design theory. You can also rely on Edraw by just choosing one theme which includes harmonious color, font, and line style.

### 7. **Interact:** Add methods for manipulating the data or controlling what features are visible.



```
[ ] #mounting the drive
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing

# %matplotlib inline
```

Fig 7.1

Importing necessary packages

```
weather_df = pd.read_csv('/content/drive/MyDrive/ACIDDDD AKA vignesh folder/kanpur.csv', parse_dates=['date_time'], index_col='date_time')
weather_df.head(5)
```

date_time	maxtempC	mintempC	totalSnow_cm	sunHour	uvIndex	uvIndex.1	moon_illumination	moonrise	moonset	sunrise	...	WindChillC	WindGustKmph	cloudcover	humidity	precipMM	pr
2009-01-01 00:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	11	21	17	50	0.0	
2009-01-01 01:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	12	22	11	52	0.0	
2009-01-01 02:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	12	23	6	55	0.0	
2009-01-01 03:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	12	23	0	57	0.0	
2009-01-01 04:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	14	19	0	54	0.0	

5 rows x 24 columns

Fig 7.2  
All the features in the dataset

### Checking columns in our dataframe

```
weather_df.columns
```

```
Index(['maxtempC', 'mintempC', 'totalSnow_cm', 'sunHour', 'uvIndex',
      'uvIndex.1', 'moon_illumination', 'moonrise', 'moonset', 'sunrise',
      'sunset', 'DewPointC', 'FeelsLikeC', 'HeatIndexC', 'WindChillC',
      'WindGustKmph', 'cloudcover', 'humidity', 'precipMM', 'pressure',
      'tempC', 'visibility', 'winddirDegree', 'windspeedKmph'],
      dtype='object')
```

Fig 7.3  
Checking columns in the data frame

## Checking is there any null values in dataset

```
weather_df.isnull().any()
```

```
maxtempC      False
mintempC      False
totalSnow_cm  False
sunHour       False
uvIndex       False
uvIndex.1     False
moon_illumination  False
moonrise      False
moonset       False
sunrise       False
sunset        False
DewPointC     False
FeelsLikeC    False
HeatIndexC    False
WindChillC    False
WindGustKmph  False
cloudcover    False
humidity      False
precipMM      False
pressure      False
tempC         False
visibility     False
winddirDegree False
windspeedKmph False
dtype: bool
```

Fig 7.4  
Modelling the algorithm into dataset

```
weather_df_num=weather_df.loc[:,['maxtempC','mintempC','cloudcover','humidity','tempC','sunHour','HeatIndexC','precipMM','pressure','windspeedKmph']]
weather_df_num.head()
```

```
maxtempC  mintempC  cloudcover  humidity  tempC  sunHour  HeatIndexC  precipMM  pressure  windspeedKmph
date_time
2009-01-01 00:00:00    24      10        17      50     11      8.7        12      0.0    1015        10
2009-01-01 01:00:00    24      10        11      52     11      8.7        13      0.0    1015        11
2009-01-01 02:00:00    24      10         6      55     11      8.7        13      0.0    1015        11
2009-01-01 03:00:00    24      10         0      57     10      8.7        13      0.0    1015        12
2009-01-01 04:00:00    24      10         0      54     11      8.7        14      0.0    1016        11
```

Fig 7.5  
Showcasing only implemented features

```
[ ] train_X,test_X,train_y,test_y=train_test_split(weather_x,weather_y,test_size=0.2,random_state=4)
```

```
▶ train_X.shape
```

```
↳ (77145, 9)
```

```
[ ] train_y.shape
```

```
(77145,)
```

Fig 7.6

Splitting of data into test and train data set

Linear regression

```
[ ] model=LinearRegression()  
    model.fit(train_X,train_y)
```

```
▼ LinearRegression  
LinearRegression()
```

```
[ ] prediction = model.predict(test_X)
```

```
[ ] #calculating error  
    np.mean(np.absolute(prediction-test_y))
```

```
1.2004735794096797
```

```
[ ] print('Variance score: %.2f' % model.score(test_X, test_y))
```

```
Variance score: 0.96
```

Fig 7.7

Implementing Linear Regression model and predicting mean and variance



## Decision Tree Regression

```
[ ] from sklearn.tree import DecisionTreeRegressor  
regressor=DecisionTreeRegressor(random_state=0)  
regressor.fit(train_X,train_y)
```

```
DecisionTreeRegressor  
DecisionTreeRegressor(random_state=0)
```

```
[ ] prediction2=regressor.predict(test_X)  
np.mean(np.absolute(prediction2-test_y))
```

```
0.5630130830784121
```

```
[ ] print('Variance score: %.2f' % regressor.score(test_X, test_y))
```

```
Variance score: 0.98
```

Fig 7.8

Implementing Decision Tree Regression for prediction of mean and variance score

## Random Forest Regression

```
[ ] from sklearn.ensemble import RandomForestRegressor  
regr=RandomForestRegressor(max_depth=90,random_state=0,n_estimators=100)  
regr.fit(train_X,train_y)
```

```
RandomForestRegressor  
RandomForestRegressor(max_depth=90, random_state=0)
```

```
[ ] prediction3=regr.predict(test_X)  
np.mean(np.absolute(prediction3-test_y))
```

```
0.4749165453503998
```

```
[ ] print('Variance score: %.2f' % regr.score(test_X, test_y))
```

```
Variance score: 0.99
```

Fig 7.9

Implementing Random Forest Regression and calculating mean and variance

## Calculating R2-score for Multiple Linear Regression

```
[ ] print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
    print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
    print("R2-score: %.2f" % r2_score(test_y, prediction) )
```

```
Mean absolute error: 1.20
Residual sum of squares (MSE): 2.51
R2-score: 0.96
```

Fig7.10  
Calculating R -squared Score for Linear Regression

## Calculating R2-score for Decision Tree Regression

```
print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction2 - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction2 - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y, prediction2) )
```

```
Mean absolute error: 0.56
Residual sum of squares (MSE): 1.12
R2-score: 0.98
```

Fig 7.11  
Calculating R-Squared score for Decision Tree Regression

## Calculating R2-score for Random Forest Regression

```
[ ] from sklearn.metrics import r2_score

print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction3 - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction3 - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y, prediction3) )
```

```
Mean absolute error: 0.47
Residual sum of squares (MSE): 0.63
R2-score: 0.99
```

Fig 7.12  
Calculating R -squared score for Random forest algorithm

## Guassain Naive Bayes

```
[ ] from sklearn.naive_bayes import GaussianNB
    gnb = GaussianNB()
    y_pred = gnb.fit(train_X,train_y).predict(test_X)

    #Calculating the R-2 Score for the Naive Bayes
    # print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
    # print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
    # print("R2-score: %.2f" % r2_score(test_y,prediction ) )
    from sklearn.metrics import precision_score , accuracy_score , f1_score
    precision = precision_score(test_y , y_pred, average='macro')
    accuracy = accuracy_score(test_y, y_pred)
    f1 = f1_score(test_y, y_pred, average='macro')
    print("Accuracy:", accuracy)
    print("F1 score:", f1)
    print("Precision score:", precision)
```

Accuracy: 0.15689324415409342  
F1 score: 0.16618640464154924  
Precision score: 0.18385951913975518

Fig 7.13  
Gaussian Naïve Bayes

## Multi-nominal Niave Bayes

```
[ ] from sklearn.naive_bayes import MultinomialNB

    clf = MultinomialNB(force_alpha=True)
    y_pred = clf.fit(train_X,train_y).predict(test_X)

    #Calculating the R2- Score for the Muti-nominal Naive bayes
    # print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
    # print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
    # print("R2-score: %.2f" % r2_score(test_y,prediction ) )
    from sklearn.metrics import precision_score , accuracy_score , f1_score
    precision = precision_score(test_y , y_pred, average='macro')
    accuracy = accuracy_score(test_y, y_pred)
    f1 = f1_score(test_y, y_pred, average='macro')
    print("Accuracy:", accuracy)
    print("F1 score:", f1)
    print("Precision score:", precision)
```

Accuracy: 0.07103230155026702  
F1 score: 0.06334048751654787  
Precision score: 0.08430329810077584

Fig 7.14  
Multinomial Naïve

## Complement Naive Bayes

```
[ ] from sklearn.naive_bayes import ComplementNB
clf = ComplementNB(force_alpha=True)
y_pred = clf.fit(train_X,train_y).predict(test_X)

#Calculate the R-2 score for the complement Naive Bayes
# print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
# print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
# print("R2-score: %.2f" % r2_score(test_y,prediction ) )
from sklearn.metrics import precision_score , accuracy_score , f1_score
precision = precision_score(test_y , y_pred, average='macro')
accuracy = accuracy_score(test_y, y_pred)
f1 = f1_score(test_y, y_pred, average='macro')
print("Accuracy:", accuracy)
print("F1 score:", f1)
print("Precision score:", precision)
```

```
Accuracy: 0.07963913516876653
F1 score: 0.023960654651783413
Precision score: 0.023475394754889977
```

Fig 7.15  
Implementing complement Naïve Bayes

## Bernoulli NB

```
▶ from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB(force_alpha=True)
y_pred= clf.fit(train_X,train_y).predict(test_X)

# print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
# print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
# print("R2-score: %.2f" % r2_score(test_y,prediction ) )
from sklearn.metrics import precision_score,accuracy_score, f1_score
precision = precision_score(test_y , y_pred, average='macro')
accuracy = accuracy_score(test_y, y_pred)
f1 = f1_score(test_y, y_pred, average='macro')
print("Precision score:", precision)
print("Accuracy:", accuracy)
print("F1 score:", f1)
```

```
☐ Precision score: 0.0021664988804729465
Accuracy: 0.05573702493907814
F1 score: 0.004017334015996879
```

Fig 7.16  
Implementing Bernoulli's Naïve Bayes

## Categorical NB

```
from sklearn.naive_bayes import CategoricalNB
clf = CategoricalNB(force_alpha=True)
y_pred = clf.fit(train_X,train_y).predict(test_X)
# gnb = GaussianNB()
# gnb.fit(train_X, train_y)
# y_pred = gnb.predict_proba(test_X)[: , 1]

# print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
# print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
# print("R2-score: %.2f" % r2_score(test_y,prediction ) )
from sklearn.metrics import precision_score , accuracy_score , f1_score

precision = precision_score(test_y , y_pred, average='macro')
accuracy = accuracy_score(test_y, y_pred)
f1 = f1_score(test_y, y_pred, average='macro')

print("Accuracy:", accuracy)
print("F1 score:", f1)
print("Precision score:", precision)
```

```
Accuracy: 0.2671747809405299
F1 score: 0.24187558832123787
Precision score: 0.24397573483792967
```

Fig 7.17  
Categorical Naïve Bayes

## **CHAPTER 8 RESULTS AND DISCUSSION**

After training the machine learning model using the local weather dataset, we evaluate it using evaluating metrics. Given a larger dataset and training data and parameters, a higher accuracy trained model will be obtained thus this provides a reason to compare with the Linear Regression, KNN , Random forest and decision tree algorithm to find and analyze the data, integrate it with the acid rain forecast to create a front end. The parameters used are obtained in real time with sensors from organizations that have high accuracy of data collection, like Airport, Weather stations, etc.

To fully compute a highly accurate and efficient model for weather forecasting and to enhance the performance of the same is to have better classifiers than the ones discussed in Experimentation. (Chapter 7 )

```
[ ] import numpy as np
import os
import pandas as pd
import sklearn

from sklearn import preprocessing
from google.colab import drive
drive.mount('/content/drive/')
data1 = pd.read_csv('/content/drive/MyDrive/Major pjt/weatherHistory.csv')
data1.head()
```

Fig 8.1  
Importing sklearn for preprocessing

Mounted at /content/drive/

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

Fig 8.2  
Dataset for the acid rain

```
data1 = data1.drop(['Summary', 'Precip Type'], axis=1)
data1.head(100)]
```

	Formatted Date	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...
95	2006-04-12 23:00:00.000 +0200	7.855556	6.122222	0.72	9.8049	11.0	15.0052	0.0	1006.56	Foggy overnight and breezy in the morning.
96	2006-04-13 00:00:00.000 +0200	7.316667	6.211111	0.75	6.6654	326.0	15.8746	0.0	1007.07	Overcast throughout the day.
97	2006-04-13 01:00:00.000 +0200	7.244444	6.005556	0.75	7.1162	309.0	15.8746	0.0	1007.37	Overcast throughout the day.
98	2006-04-13 02:00:00.000 +0200	5.438889	5.438889	0.88	3.7191	193.0	9.9820	0.0	1012.23	Overcast throughout the day.
99	2006-04-13 03:00:00.000 +0200	7.200000	4.550000	0.76	14.4739	273.0	15.8263	0.0	1007.28	Overcast throughout the day.

Fig 8.3  
Drop of columns in which has null values

```
[ ] label_encoder = preprocessing.LabelEncoder()
data1["Daily Summary"] = label_encoder.fit_transform(data1["Daily Summary"])
data1["Daily Summary"].unique()

array([197, 111, 33, 36, 143, 209, 132, 57, 35, 204, 170, 94, 118,
       166, 156, 188, 205, 187, 155, 95, 198, 120, 43, 60, 42, 119,
       47, 51, 162, 59, 63, 52, 41, 175, 169, 147, 91, 37, 150,
       83, 39, 112, 58, 97, 45, 148, 29, 154, 152, 144, 34, 18,
       183, 96, 84, 195, 98, 157, 128, 165, 193, 194, 61, 146, 28,
       82, 23, 149, 89, 79, 72, 80, 174, 171, 75, 40, 64, 48,
       62, 49, 181, 179, 20, 105, 202, 71, 19, 109, 3, 88, 208,
       2, 99, 104, 191, 164, 192, 85, 153, 159, 200, 178, 176, 103,
       110, 22, 76, 142, 190, 115, 121, 158, 134, 90, 182, 203, 16,
       160, 92, 107, 21, 108, 137, 136, 81, 126, 207, 201, 186, 15,
       55, 125, 32, 124, 54, 25, 106, 177, 74, 93, 56, 117, 50,
       44, 173, 53, 86, 141, 163, 167, 127, 6, 129, 180, 7, 151,
       66, 69, 65, 168, 14, 102, 38, 1, 130, 185, 184, 77, 172,
       9, 206, 87, 11, 73, 101, 100, 131, 0, 10, 139, 27, 8,
       199, 161, 114, 213, 145, 70, 46, 24, 13, 122, 78, 196, 26,
       4, 12, 17, 5, 113, 123, 116, 135, 140, 138, 133, 189, 68,
       30, 31, 210, 212, 67, 211])
```

Fig 8.4  
Transformation of data by label encoder

```
[ ] data1["Daily Summary"].value_counts()

111    20085
197     9981
209     6169
170     5184
35      4201
...
21         24
107         24
160         24
211         24
53          23
Name: Daily Summary, Length: 214, dtype: int64

[ ] # Stratified Sampling using Scikit-learn's Stratified Shuffle Split Class
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.25, random_state=42)
for train_index, test_index in split.split(data1, data1["Daily Summary"]):
    strat_train_set1 = data1.loc[train_index]
    strat_test_set1 = data1.loc[test_index]

train_set1 = strat_train_set1.drop("Daily Summary", axis=1)
train_set1 = strat_train_set1.drop("Formatted Date", axis=1)
train_labels = strat_train_set1["Daily Summary"].copy()
test_set1 = strat_test_set1.drop("Daily Summary", axis=1)
test_set1 = strat_test_set1.drop("Formatted Date", axis=1)
test_labels = strat_test_set1["Daily Summary"].copy()
```

Fig 8.5  
Splitting of testing and training data



```
[ ] from sklearn.metrics import matthews_corrcoef
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
parameters = {'n_neighbors':[1,3,5,7,10,13,15], 'weights':['uniform', 'distance'], 'metric':['euclidean', 'manhattan'], 'algorithm':['auto', 'ball_tree', 'kd_tree', 'brute']}
neigh1 = KNeighborsClassifier()
clf = GridSearchCV(neigh1, parameters, scoring = 'balanced_accuracy', cv=10)
res1=clf.fit(train_set, train_labels)
res1.best_estimator_

KNeighborsClassifier
KNeighborsClassifier(metric='manhattan', n_neighbors=1)

[ ] res1.best_params_

{'algorithm': 'auto',
 'metric': 'manhattan',
 'n_neighbors': 1,
 'weights': 'uniform'}

[ ] y_test_pred=res1.best_estimator_.predict(test_set)

[ ] matthews_corrcoef(test_labels, y_test_pred)

0.8482237010443743

from sklearn.metrics import accuracy_score
accuracy_score(test_labels, y_test_pred)

0.8593348262420171
```

Fig 8.6  
Implementing the KNN classifier

```
from sklearn.metrics import precision_score
precision_score(test_labels, y_test_pred, average=None)

array([[1.         , 1.         , 0.83333333, 1.         , 1.         ,
        1.         , 0.85714286, 1.         , 0.875         , 1.         ,
        0.55555556, 0.85714286, 1.         , 1.         , 0.83333333,
        0.8         , 1.         , 0.71428571, 1.         , 1.         ,
        1.         , 1.         , 1.         , 1.         , 0.83333333,
        0.85714286, 1.         , 1.         , 0.875         , 0.9         ,
        0.66666667, 0.83333333, 0.77777778, 0.66666667, 0.5         ,
        0.85779398, 0.5         , 0.67540984, 0.83333333, 0.67605634,
        0.53571429, 0.75         , 0.7826087 , 0.85028249, 1.         ,
        0.76842105, 1.         , 0.65277778, 0.74626866, 0.57142857,
        1.         , 0.74074074, 0.72972973, 1.         , 0.85714286,
        0.875         , 0.83333333, 0.91208791, 0.69230769, 0.54166667,
        0.74103586, 0.67105263, 0.59615385, 0.8620178 , 0.60583942,
        1.         , 1.         , 1.         , 1.         , 1.         ,
        0.625         , 0.81818182, 0.9047619 , 0.8         , 0.76923077,
        0.8         , 1.         , 1.         , 0.71428571, 0.5         ,
        0.78571429, 1.         , 0.52631579, 0.87714988, 0.83333333,
        0.75         , 0.8         , 0.8         , 0.8         , 0.8125         ,
        1.         , 0.73584906, 0.57142857, 1.         , 0.71428571,
        0.79464286, 0.71428571, 0.75         , 0.8         , 0.89473684,
        0.78125         , 1.         , 0.85714286, 0.82142857, 0.95384615,
        0.85185185, 1.         , 0.75         , 0.85185185, 0.6         ,
        1.         , 0.97311514, 0.51515152, 1.         , 0.8         ,
        0.84         , 0.66666667, 1.         , 0.90909091, 1.         ,
        0.80952381, 0.72727273, 1.         , 1.         , 0.83333333,
        1.         , 1.         , 0.85714286, 1.         , 0.83333333,
        1.         , 0.42857143, 0.96819338, 0.66666667, 1.         ,
        0.85714286, 0.85714286, 0.88888889, 0.5         , 1.         ,
        1.         , 1.         , 1.         , 0.96107383, 0.77272727,
        0.85714286, 1.         , 0.80487805, 0.77083333, 0.51111111,
        0.57352941, 0.66666667, 0.67676768, 1.         , 0.61235955,
        0.38317757, 0.69724771, 0.88888889, 0.58333333, 0.33333333])
```

Fig 8.7  
Calculating precision score

```

from sklearn.metrics import f1_score
f1_score(test_labels, y_test_pred, average=None)

array([1.          , 1.          , 0.83333333, 0.66666667, 1.          ,
       1.          , 0.92307692, 1.          , 0.82352941, 0.90909091,
       0.66666667, 0.92307692, 0.90909091, 1.          , 0.83333333,
       0.72727273, 1.          , 0.76923077, 1.          , 1.          ,
       1.          , 0.90909091, 0.8          , 0.90909091, 0.83333333,
       0.92307692, 0.8          , 1.          , 0.7          , 0.90393013,
       0.44444444, 0.83333333, 0.73684211, 0.66666667, 0.25          ,
       0.87657196, 0.5          , 0.66131621, 0.83333333, 0.67132867,
       0.65217391, 0.69863014, 0.67924528, 0.83611111, 0.66666667,
       0.76439791, 0.90909091, 0.68115942, 0.71942446, 0.55172414,
       0.8          , 0.74074074, 0.71052632, 0.8          , 0.92307692,
       0.7          , 0.83333333, 0.92017738, 0.66666667, 0.48148148,
       0.79657388, 0.57303371, 0.58490566, 0.86265776, 0.61710037,
       1.          , 0.5          , 1.          , 0.90909091, 1.          ,
       0.71428571, 0.7826087          , 0.84444444, 0.72727273, 0.8          ,
       0.72727273, 1.          , 0.73684211, 0.76923077, 0.57142857,
       0.6875          , 0.8          , 0.57142857, 0.88916563, 0.83333333,
       0.75          , 0.72727273, 0.72727273, 0.72727273, 0.76470588,
       1.          , 0.72897196, 0.42105263, 0.90909091, 0.65326633,
       0.81651376, 0.72727273, 0.78947368, 0.77714286, 0.79069767,
       0.80645161, 0.90909091, 0.92307692, 0.79310345, 0.90510949,
       0.80701754, 0.90909091, 0.6          , 0.73015873, 0.75          ,
       1.          , 0.98375185, 0.45333333, 0.90909091, 0.72727273,
       0.76363636, 0.44444444, 0.90909091, 0.82644628, 0.69090909,
       0.87179487, 0.69565217, 1.          , 0.90909091, 0.83333333,
       1.          , 0.66666667, 0.63157895, 0.8          , 0.83333333,
       0.8          , 0.46153846, 0.97564103, 0.66666667, 0.66666667,
       0.92307692, 0.92307692, 0.88888889, 0.4          , 0.90909091,
       0.5          , 1.          , 0.90909091, 0.96561025, 0.73913043,
       0.92307692, 1.          , 0.79518072, 0.64912281, 0.46464646,
       0.609375          , 0.66666667, 0.647343          , 0.28571429, 0.63556851,
       0.40394089, 0.68778281, 0.76190476, 0.58333333, 0.22222222])

```

Fig 8.8  
Calculating F1score

```

[ ] from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(test_labels,y_test_pred)

0.7155539210932479

[ ]

```

Fig 8.9  
Balanced accuracy score

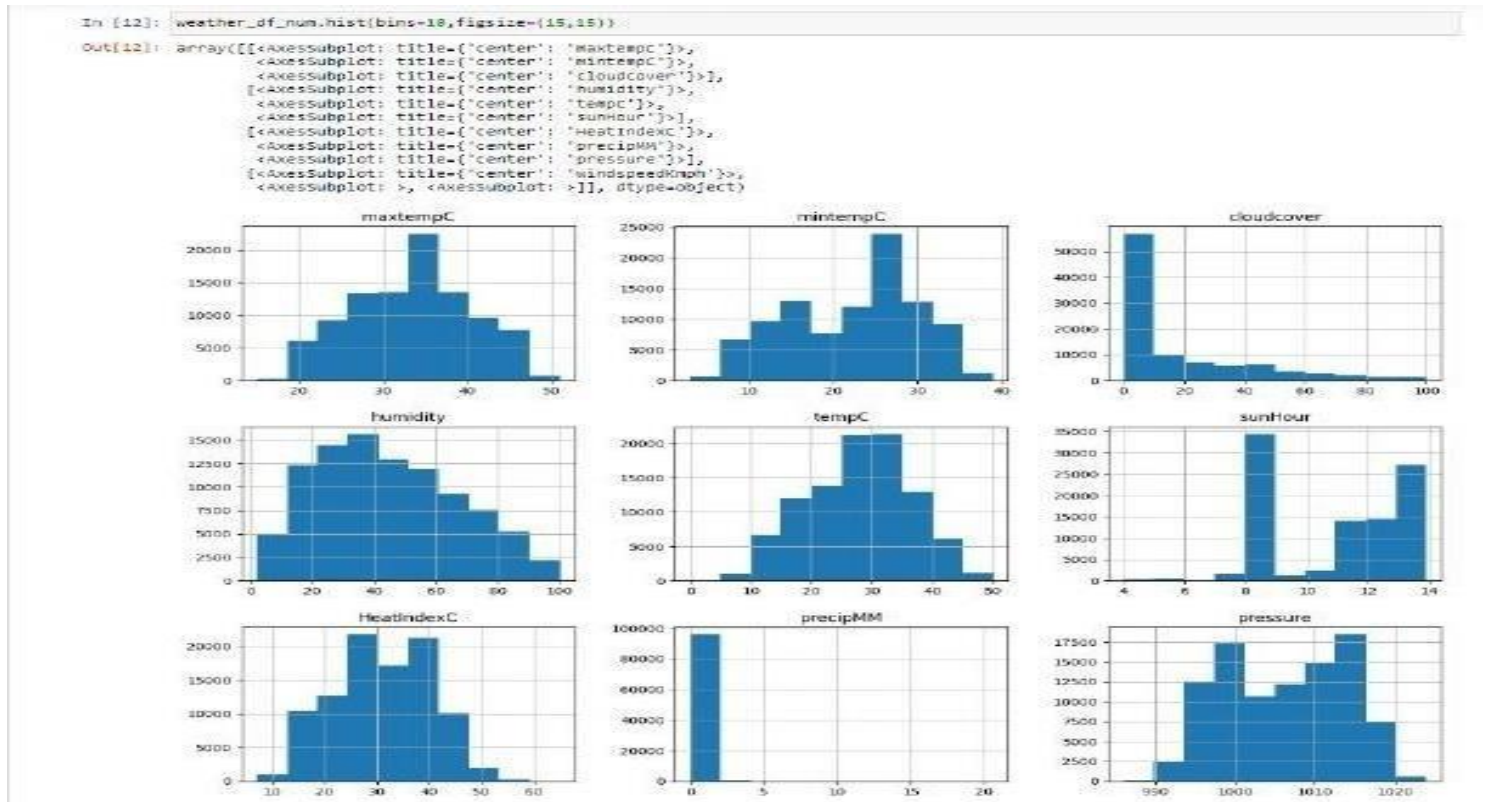


Fig 8.10  
Feature selection through data visualization

## Calculating R2-score for Multiple Linear Regression

```
[ ] print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y,prediction) )
```

```
Mean absolute error: 1.20
Residual sum of squares (MSE): 2.51
R2-score: 0.96
```

Fig 8.11  
Calculating R-squared Score

## Calculating R2-score for Decision Tree Regression

```
print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction2 - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction2 - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y, prediction2) )
```

```
Mean absolute error: 0.56
Residual sum of squares (MSE): 1.12
R2-score: 0.98
```

Fig 8.12  
Calculating R-Squared score for Decision Tree Regression

## Calculating R2-score for Random Forest Regression

```
[ ] from sklearn.metrics import r2_score

print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction3 - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction3 - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y, prediction3) )
```

```
Mean absolute error: 0.47
Residual sum of squares (MSE): 0.63
R2-score: 0.99
```

Fig 8.13  
Calculating R-squared Score for Random Forest Regression

## REFERENCES

- [1]. K Geetha Rani, Dr D C Joy Winnie, S Sufiyah Begum, S Nirosha ‘Designing a model for weather forecasting using machine learning’ , IEEE Journal, August 2020
- [2]. Bogdan Bochenek, Zbigniew Ustrnul, ‘Machine learning in weather prediction and climate analyses – applications and perspectives’, MDPI Atmosphere Journal, January 2022
- [3]. A H M Jakaria, Md Mosharaf Hossain, Mohammad Ashiqur Rahman, ‘Smart weather forecasting using machine learning: a case study in Tennessee’, Researchgate Journal, August 2020
- [4]. Nitin Singh, Saurabh Chaturvedi, Shamim Akhter, ‘Weather forecasting using machine learning’, IEEE Journal, December 2019.
- [5]. Siddharth Singh, Mayank Kaushik, Ambuj Gupta, Anil Kumar Malviya, ‘Weather forecasting using Machine learning techniques’, Galgotias University, February 2020.
- [6]. Dires Negash Fente, Dheeraj Kumar Singh, ‘Weather Forecasting Using Artificial Neural Network’, IEEE, International conference 2018.
- [7]. Pradeep Hewage, Marcello Trovati, Ella Pereira and Ardhendu Behera ‘Deep learning based effective fine-grained weather forecasting model’, Elsevier Journal 2020.
- [8]. Kumar Abhishek, M.P.Singh, Saswata Ghosh, Abhishek Anand, ‘Weather forecasting model using Artificial Neural Network’ , Elsevier Journal ,2012.
- [9]. Imran Maqsood, Muhammad Riaz Khan, Ajith Abraham , ‘An ensemble of neural networks for weather forecasting’ , Springer , Journal , May 2020.
- [10]. Kartika Purwandari, Join W. C.Sigalingging , Tjeng Wawan Cenggoro , Bens Pardamean, ‘Multi-class Weather Forecasting from Twitter Using Machine Learning Approaches’ , Elsevier , Conference , 4 sep 2020.
- [11]. Raymond Lee and James Liu, ‘ iJADE WeatherMAN: A Weather Forecasting SystemUsing Intelligent Multi Agent-Based Fuzzy Neuro Network’ , IEEE , Journa , August 2020.

