435643484-Project-Report

Bachelor's of computer application (Integral University)

Weather Prediction

Project 1 Report

Submitted in the partial fulfillment of requirement for the award of the degree of

Bachelor of Technology (B.Tech.)

in

Computer Science and Engineering

Under the Supervision of:

Er. Amit Sharma

Assistant Professor

Submitted By:

Gaurav (2316005)

Dhruv (2316017)

Atharva (2316047)

Aman (2316052)

Department of Computer Science and Engineering

Ambala College of Engineering and Applied Research

(Affiliated to Kurukshetra University, Kurukshetra)

DECEMBER-2019

CERTIFICATE

This is to certify that **Gaurav Sharma (2316005), Dhruv Goyal (2316017)**, **Atharva(2316047) and Aman Kumar(2316052)** studying in Ambala College of Engineering and Applied Research, Devsthali, (Batch: 2016-2020) have completed their project1 entitled **"Weather Prediction"** at Ambala College of Engineering and Applied Research, Devsthali under my supervision.

It is further certified that they had attended required number of practical classes at Ambala College of Engineering and Applied Research, Devsthali for the completion of their project1 during 7^{th} semester.

Er. Amit Sharma

Project Supervisor

DECLARATION OF STUDENT

We hereby declare that the work which is present in this project1 report entitled **"Weather Prediction"** in the partial fulfillment for the award of the degree of Bachelor of Technology and submitted to the Department of Computer Science and Engineering of Ambala College of Engineering and Applied Research, Devsthali affiliated to Kurukshetra University, Kurukshetra is an authentic record of us, carried out during a period of **AUG 2019 to NOV 2019,** under the supervision of **Er. Amit Sharma, Assistant Professor (CSE).**

The matter presented in this project report has not been submitted by us for the award of any other degree of this or any other institute/university.

Date:_____

Gaurav Sharma (2316005)

Dhruv Goyal (2316017)

Atharva (2316047)

Aman Kumar (2316052)

PREFACE

Weather prediction today is largely based on expert systems. This is undoubtedly, a great solution. But the intricacies involved in computing the network required to infer all the cases is astronomically vast. Hence, a Machine learning based solution is capable of using a statistically tuned ML-agent that harnesses the capabilities of traversing every case there realistically could be to create a fine tuned system wherein projections would be rather accurate for a much less developmental cost and computational cost alike.

Chapter 1 provides the details regarding the introduction of the area of project, Software tools and technology used like React and Python.

Chapter 2 provides problem statement, Objective in term of functional and non functional requirements provided by the project

Chapter 3 includes various UML diagrams like use case diagrams, activity diagram, description about tables, ER diagrams and flow charts.

Chapter 4 includes testing, where the various modules testing is done to check whether the is according to the user, Unit testing and Integration testing is done on modules using different test cases.

Chapter 5 includes results with proper screen shots and description about the screenshots like its purpose, how this screen will appear etc.

Chapter 6 clearly defines the steps required to make settings in the pc to run your project along with the snapshot. In short, anybody can install and run our project without our support by just referencing this chapter.

ACKNOWLEDGEMENT

Engineers in all disciplines must acquire knowledge of project making. Student, in particular, will find 'project making' as an integral part of their studies that will infuse the spirit of doing practical work in them.

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible whose constant guidance crowned our efforts with success.

We sincerely express our deep gratitude to the management of our college for giving us liberty to choose and to work on the most relevant project i.e. **"Weather Prediction"**. We are thankful to **Er. Manjit Singh** (HOD CSE) for ensuring that we have a smooth environment at the college and lab. At the very outset we would like to offer our never ending thanks to our project supervisor **Er. Amit Sharma** (Assistant Professor, CSE) who helped us with our project from the beginning till the end. His continuous surveillance over our work allowed us to work more efficiently.

Gaurav Sharma (2316005)

Dhruv Goyal (2316017)

Atharva (2316047)

Aman Kumar (2316052)

ABSTRACT

Weather prediction is the application of science and technology to predict the state of the atmosphere for a given location. Here this system will predict weather based on parameters such as temperature, humidity and wind. This system is a web application with effective graphical user interface. To predict the future's weather condition, the variation in the conditions in past years must be utilized. The probability that it will match within the span of adjacent fortnight of previous year is very high .We have proposed the use of linear regression for weather prediction system with parameters such as temperature, humidity and wind. It will predict weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1
Introduction

1.1 Introduction to Weather Forecasting

Weather forecasting is the task of predicting the state of the atmosphere at a future time and a specified location. Traditionally, this has been done through physical simulations in which the atmosphere is modeled as a fluid. The present state of the atmosphere is sampled, and the future state is computed by numerically solving the equations of fluid dynamics and thermodynamics. However, the system of ordinary differential equations that govern this physical model is unstable under perturbations, and uncertainties in the initial measurements of the atmospheric conditions and an incomplete understanding of complex atmospheric processes restrict the extent of accurate weather forecasting to a 10 day period, beyond which weather forecasts are significantly unreliable. Machine learning, on the contrary, is relatively robust to perturbations and doesn't require a complete understanding of the physical processes that govern the atmosphere. Therefore, machine learning may represent a viable alternative to physical models in weather forecasting.

Machine learning is the ability of computer to learn without being explicitly programmed. It allows machines to find hidden patterns and insights. In supervised learning, we build a model based on labeled training data. The model is then used for mapping new examples. So, based on the observed weather patterns from the past, a model can be built and used to predict the weather.

This project work focuses on solving the weather prediction anomalies and in-efficiency based on linear regression algorithms and to formulate an efficient weather prediction model based on the linear regression algorithms

1.2 Methodology used

In a developing country and an economy like India where major population is dependent on agriculture, weather conditions play an important and vital role in economic growth of the overall nation. So, weather prediction should be more precise and accurate. Weather parameters are collected from the open source . The data used in this project is of the years 2013-2019. The programming language used is 'Python'. Fig. 1.1 visualizes the system in the form of a block diagram.

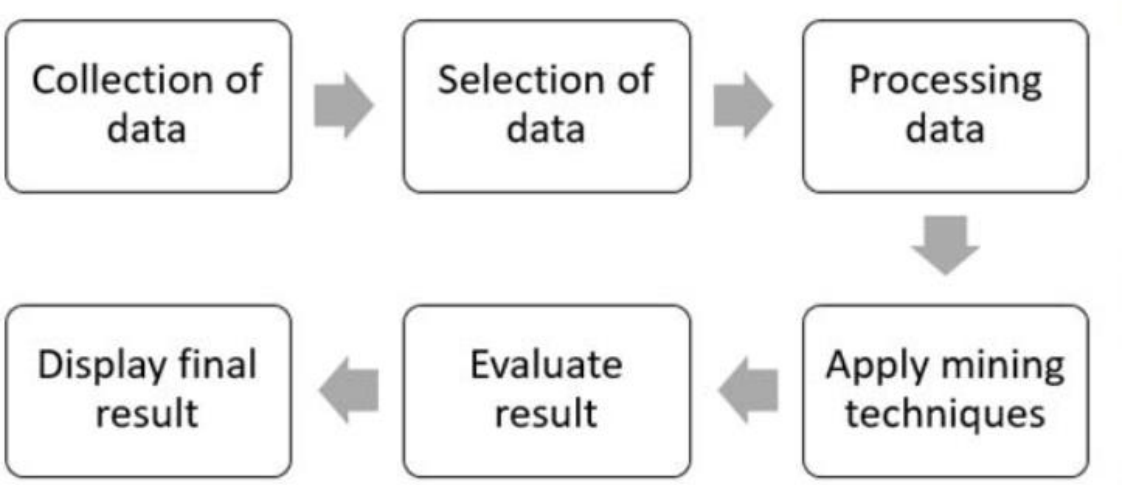


Fig 1.1 System Block Diagram

The weather is predicted using various indices like temperature, humidity and dew-point. Temperature is the measure of hotness or coldness, generally measured using thermometer. Units of temperature most frequently used are Celsius and Fahrenheit. We have used maximum and minimum temperature values along with normal temperature as different index values for prediction of the weather.

Humidity is the quantity of water vapor present in the atmosphere. It is a relative quantity.

Dew point is the temperature of the atmosphere (which varies according to pressure and humidity) below which water droplets begin to condense and dew is formed.

1.3 Technologies Used

1.3.1 Machine learning – Linear Regression

Linear regression is the most basic and frequently used predictive model for analysis. Regression estimates are generally used to describe the data and elucidate relationship between one or more independent and dependent variables. Linear regression finds the best-fit through the points, graphically. The best-fit line through the points is known as the regression line.

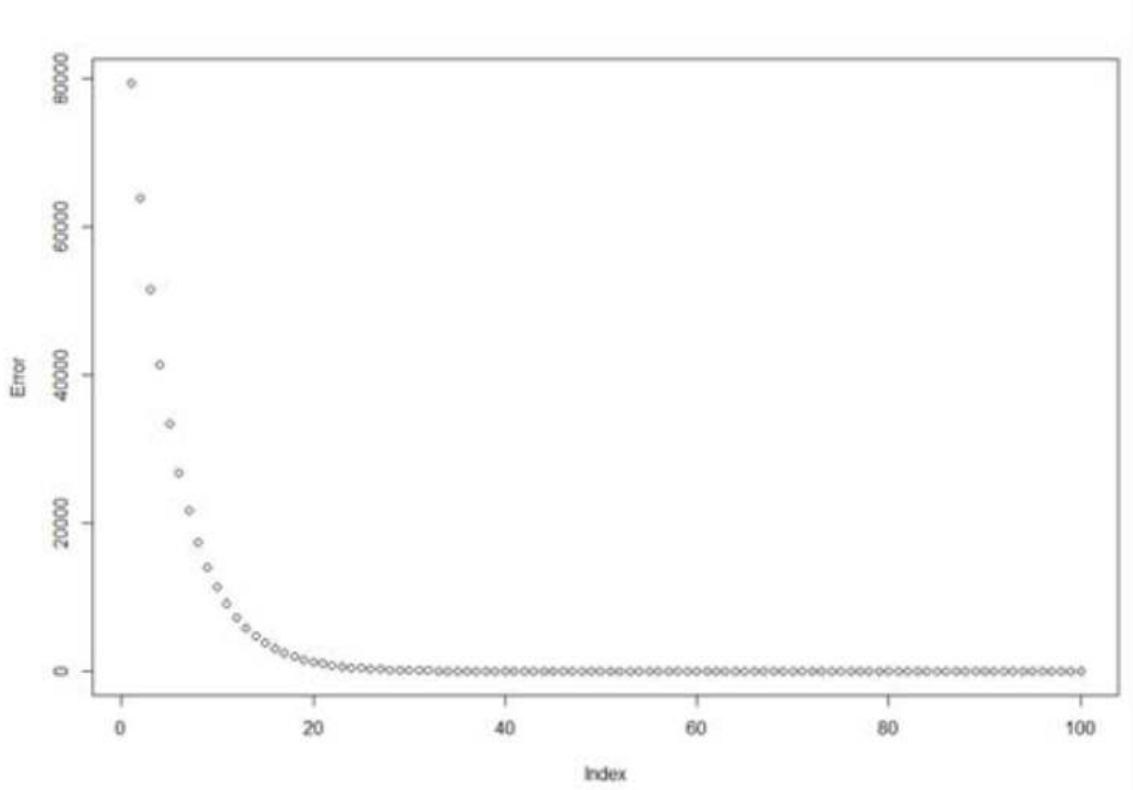


Fig. 1. Example of regression line

Fig. 1 is an example of the best-fit line. Here, the line can be straight or curved depending on the data. The best-fit line can also be a quadratic or polynomial which gives us better answer to our questions.

1.3.2 Javascript

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very

easy to implement because it is integrated with HTML. It is open and cross-platform. Once you learnt Javascript, it helps you developing great front-end as well as back-end softwares using different Javascript based frameworks like jQuery, Node.JS etc.

JavaScript is used to create interactive websites. It is mainly used for:

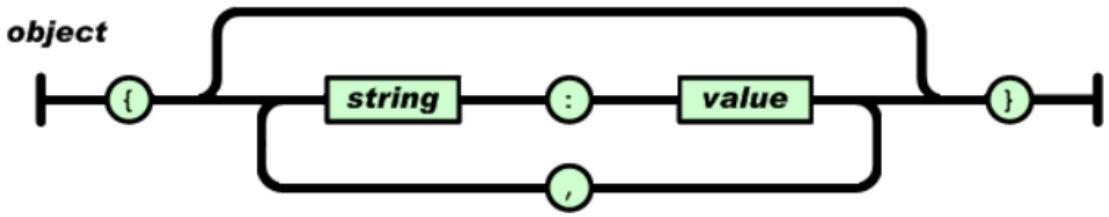
- Client-side validation
- Displaying pop-up windows and dialog boxes

1.3.3 Json

JSON (JavaScript Object Notation) is a lightweight format that is used for data interchanging. It is based on a subset of JavaScript language. It has been the preferred format because it is much more lightweight

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, structure, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.



1.3.4 ReactJS

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only

for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps.

1.3.5 Adobe After Effects

Adobe After Effects is a digital visual effect, motion graphics, and compositing application developed by Adobe Systems and used in the post-production process of film making and television production. Among other things, After Effects can be used for keying, tracking, compositing, and animation. It also functions as a very basic non-linear editor, audio editor, and media transcoder.

After Effects has extensive plug-in support; a broad range of third-party plug-ins are available. A variety of plug-in styles exist, such as particle systems for realistic effects for rain, snow, fire, etc.

With or without third-party plug-ins, After Effects can render 3D effects. Some of these 3D plug-ins use basic 2D layers from After Effects. In addition to 3D effects, there are plug-ins for making video look like film or cartoons; simulating fire, smoke, or water; particle systems; slow motion; creating animated charts, graphs, and other data visualization; calculating the 3D movement of a camera in a 2D video shot; eliminating flicker, noise, or rigging lines; translating timelines from FCP or Avid; adding high-end color correction; and other workflow improvements and visual effects.

1.3.6 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic

binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed

1.3.7 Weather API

Weather APIs are Application Programming Interfaces that allow you to connect to large databases of weather forecast and historical information. The weather API provides enough weather data for basic weather information (eg current weather, forecast, UV index data, and historical weather information). You can use geolocation and names to get a city location.

Chapter 2
Problem Identification

2.1 Problem Statement

Weather prediction is a useful tool for informing populations of expected weather conditions. Weather prediction is a complex topic and poses significant variation in practice. We will attempt to understand and implement a weather prediction application using the linear regression.

2.2 Project Scope

- Weather forecasts are made by collecting as much data as possible about the current state of the atmosphere (particularly the temperature, humidity and wind) to determine how the atmosphere evolves in the future.
- However, the chaotic nature of the atmosphere makes the forecasts less accurate as the range of the forecast increases.
- Traditional observations made at the surface of atmospheric pressure, temperature, wind speed, wind direction, humidity, precipitation are collected routinely from trained observers, automatic weather stations or buoys.During the data assimilation process, information gained from the observations is used In conjunction with a numerical model's most recent forecast for the time that observations were made to produce the meteorological analysis. The complicated equations which govern how the state of a fluid changes with time require supercomputers to solve them.
- The output from this model can be used the weather forecast as alternative.

2.3 Design and Implementation Constraints

The product is developed using API server. The back-end database are CSV files on the basis of that, the prediction takes place. The product is a general-purpose application software in which any user can gather the predicted information. The linear regression is used that predicts the weather based on the previous analysis of a data.

Major Components:

- Data Collection: The feeding historical data to the system, this could be from specific region.
- Data Cleaning: Under this component the data like the missing data, duplicated data is found and bad data is weed out
- Data selection: Under this stage, relevant data related to analysis is retrieved and classified under 6 attributes.

2.4 User Documentation

The product is a general-purpose application software where any user can access the software to gather information about the weather based on a current or a previous data analysis. In this product the software makes a request to the server to access the current dataset through which the data is analyzed using various machine learning algorithms.

2.5 Assumptions and Dependencies

- The software product is dependent on a dataset that is been retrieved from the server using various commands.
- The product will work based on the algorithm that has been discussed above.

2.6 System Features

2.6.1 Actors

- User
- Historical data provider
- Administrator

2.7 Functional Requirements

2.7.1 Accessing a database

- The system should allow administrator to add historical weather data.
- The system should be able to recognize patterns in temperature, humidity, and wind with use of historical data.

2.7.2 Prediction algorithm

- System should periodically apply prediction algorithms or models on obtained data and store results to central database.

- System shall obtain and display confidence value for each prediction given to user.

2.7.3 Actions performed by system

- System shall allow users to check weather for future three days.

2.8 Non-Functional Requirements

2.8.1 User Non-Functional Requirements

- System shall allow for users to get prediction for weather within almost two mouse clicks.
- System should ensure that features that do not require a user to be logged in.

2.8.2 System Non-Functional Requirements

- System should be able to run with core functionality from computer system.
- System should be able to show interactive animations to users regarding current and future climatic conditions.

2.8.3 Other Non-Functional Requirements

- System should textual prediction of climate conditions.

2.9 Other Requirements

2.9.1 Performance Requirements

The proposed software that we are going to develop will be used as the general-purpose application software. Therefore, it is expected that the database would perform functionally all the requirements that are specified by the user.

2.9.2 Safety Requirements

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup

2.9.3 Security Requirements

We are going to develop a secured database for the user. Software Quality Attributes. The Quality of the database is maintained in such a way so that it can be very user friendly to all the users.

2.9.4 Hardware Requirements

The system requires a database in order to store persistent data.

2.9.5 Software Constraints

The development of the system will be constrained by the availability of required software such as web servers, dataset and development tools.

Chapter 3
Design

Among modeling languages Unified Modeling Language (UML) has become most popular. UML is commonly used in the design and implementation of any system and software architectures. To achieve functional and non functional requirements of the system, UML model helps. In order to initiate the programming phase of building software, UML tools help in the creation of source code from UML diagram. The main objective of this paper to model a Weather Prediction System (Linear Regression approach) using UML. Weather prediction is a challenging area. The future weather conditions are predicted by trained regression model. In this chapter, we proposed a UML model for Weather Prediction using linear regression which provide a technique for predicting weather. This proposed enhanced method for weather prediction has advantages over other techniques

3.1 Introduction

In software industries, Object Oriented Development process is widely used. Object-Oriented Programming has heavily contributed toward a standardized method of modeling known as the Unified Modeling Language (UML). UML has become synonym for software modeling. UML is commonly used to model the software architecture as per the requirements and it includes a set of graphic notation techniques to create visual models of software-intensive systems. With the help of different UML diagrams for building the software, source code can be easily generated. The correctness of source code depends on the UML specification which needs to be standard, complete, precise, and unambiguous. A good UML specification leads to clearly defined semantics and an efficient code can be generated. The project is based upon the predicting Weather's condition.

3.1.1 Proposed approach

The proposed System using an enhanced approach is tested using the dataset of last 6 years from (2013-2019). The results are compared with previous methods results. The proposed enhanced method for weather prediction has advantages over the traditional techniques. This model produces the most accurate forecasts in comparison with previous techniques. This system can help the meteorologist to predict the future weather easily with accuracy.

3.1.2 Weather Prediction System Architecture

The system is developed in python along with javascript. Daily data sets of last 6 years (2013-2019) has been fetched to train our model. The system takes input from the datasets and produces the result.

The system building process consists of following sequential steps:

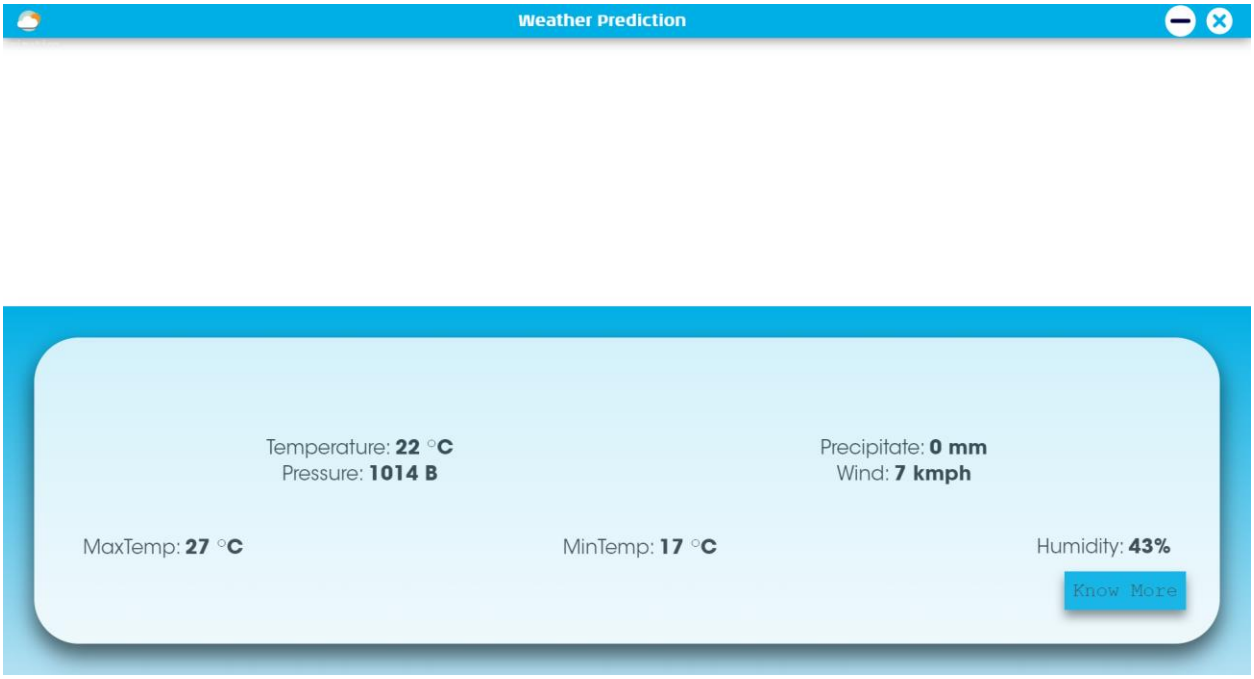
1. Fetching the dataset
2. Cleaning the dataset
3. Selection of the features of dataset
4. Train Model
5. Use the model to predict results.

3.2 Use-Case Diagram

An interaction between a user and a system is described by use case diagram. Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis is on what a system does rather than how. Use case diagrams are closely connected to scenarios. A scenario is an example of what happens when someone interacts with the system. A use case diagram is a collection of actors, use cases, and their communications.

For initial development we can use this use case. In this use case diagram we can see following use cases and actor. Use cases are self explanatory and they represent the main functions of Weather Prediction System.

3.3 Ui Design



Chapter 4
Testing

Testing is the process of evaluating a system or its component with the intent to find whether it satisfies the specified requirement or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. Systems should not be tested as a single, monolithic unit. The testing process should therefore proceed in the stages where testing is carried out incrementally in conjunction with system implementation. Errors in program components may come to light at a later stage of the testing process. The process is therefore an iterative one with information being fed back from later stage to earlier parts of the process. Following testings were done during the course of our project.

4.1 Unit Testing

Unit testing focuses verification efforts on the smaller unit of software design. Using the detailed design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of the test and the error detected as a result is limited by the constraint scope established for unit testing. The unit test is always white box oriented, and the step can be conducted in parallel for multiple modules

- Tested individual python file by debugging and using print statement
- Individual Component rendering

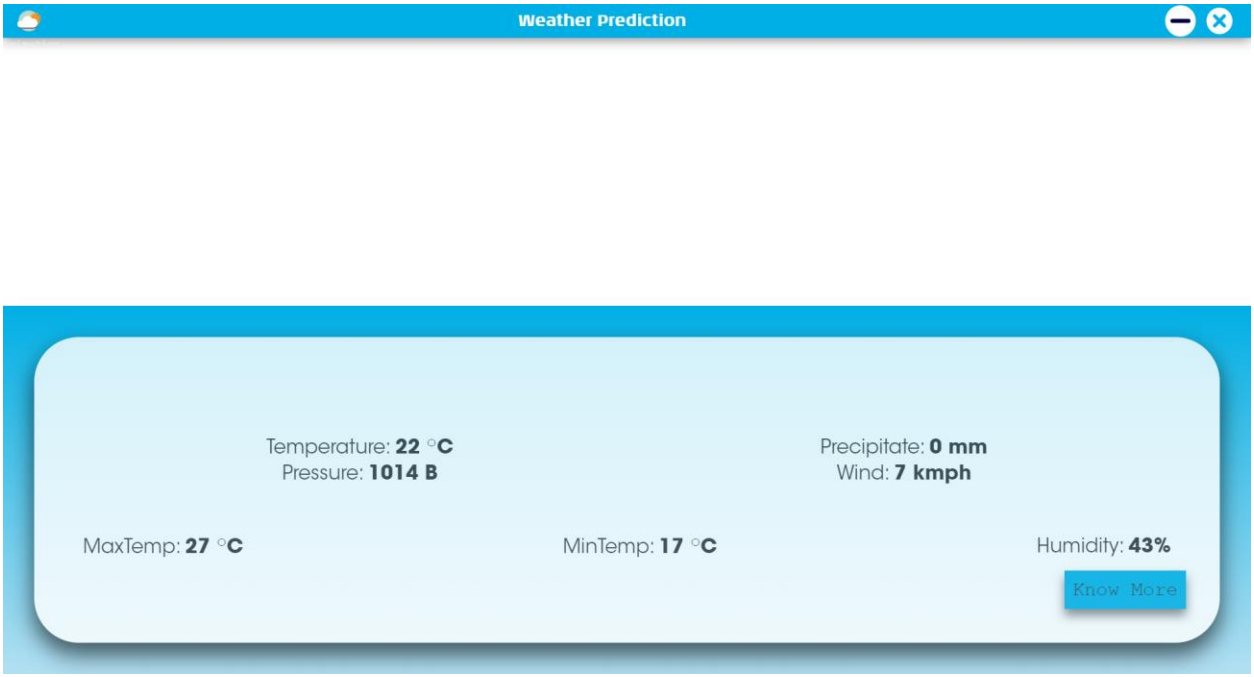
4.2 Integration Testing

With unit testing the modules may function properly, but at times they may have inadvertent affect on another, sub function when combined, may not produce the desired functions; individually acceptable impression may be signed to unacceptable levels then global data structure may present problems. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by the design.

Chapter 5
Results

1. Splash Screen

2. Main Page



Chapter 6
Deployment

6.1 Purpose of Deployment Phase

The deployment phase is the final phase of the software development life cycle (SDLC) and puts the product into production. After the project team tests the product and the product passes each testing phase, the product is ready to go live. This means that the product is ready to be used in a real environment by all end users of the product.

There are various phases of the deployment process the project team must follow to ensure the code and technology deploy appropriately. The phases include deployment preparation and procedures, product deployment, transferring ownership of the product, and closing the deployment phase.

6.2 Preparation and Procedures

In the preparation and procedures phase, the project team installs the software and conducts another test to ensure successful installation. Once the installation is complete, the project team creates operating procedures, which include instructions for how the software should work in the information technology environment. If there are issues with system functionality, the operating instructions also provide a mitigation plan to help the end user repair the issue.

6.3 Product Deployment

Under the product deployment phase, the project team implements the programming and coding to each system location. For example, say a company has two regional worksites in Noida and Kolkata and over 2,000 computer systems. The deployment phase includes pushing the program and coding to each regional site and each computer system.

6.4 Environment Variables:

- **Visual Studio Code** - Visual Studio Code is an IDE developed by Microsoft for Windows, Linux and macOS It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent completion, snippets, and code refactoring.

It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open source and released under the permissive MIT License. The compiled binaries are freeware and free for private or commercial use.

Working with Git and other SCM providers has never been easier. Review diffs, stage files, and make commits right from the editor. Push and pull from any hosted SCM service. If want to add new languages, themes, debuggers, and to connect to additional services just install the extensions. Extensions run in separate processes, ensuring they won't slow down the editor.

- **Adobe Illustrator**- Adobe Illustrator is a software application for creating drawings, illustrations, and artwork using a Windows or MacOS computer. Illustrator was initially released in 1987 and it continues to be updated at regular intervals, and is now included as part of the Adobe Creative Cloud. Illustrator is widely used by graphic designers, web designers, visual artists, and professional illustrators throughout the world to create high quality artwork. Illustrator includes many sophisticated drawing tools that can reduce the time need to create illustrations

  One of Adobe Illustrator's most important features is that the quality of artwork created using Illustrator is independent of the resolution at which it is displayed. This means that an image created in Illustrator can be enlarged or reduced without sacrificing image quality. This is an attribute of vector artwork, which uses mathematical relationships in describing lines, arcs, and other parts of an illustrator. By comparison, photographs edited using tools such as Adobe Photoshop are resolution-dependent, and image quality decreases when an image is enlarged

CONCLUSION

This research suggests and proposes an efficient and accurate weather prediction and forecasting model using linear regression concept. This concept is a part of machine learning. It is a very efficient weather prediction model and using the entities temperature, humidity and pressure, it can be used to make reliable weather predictions. This model also facilitates decision making in day to day life. It can yield even better results when applied to cleaner and larger datasets. Pre-processing of the datasets is effective in the prediction as unprocessed data can also affect the efficiency of the model.

FUTURE SCOPE

Scope of Weather Prediction

- Our system will only provide weather prediction of Amabla only.
- Prediction will be done based on historical weather activities like based on past temperature, wind, etc. pattern what will be the future weather.

Future Enhancement

- Mobile and IOS application Integration.
- Addition of new cities weather dataset to predict there future weather also.
- Addition of new Indices.
- Animation like snow and functions like notifications can also be added.

REFERENCES

1. https.wikipedia.com
2. https.w3schools.com
3. https.reactjs.org
4. https://dev.to/achowba/building-a-modal-in-react-15hg#:~:targetText=Open%20the%20Modal.js%20file,%7B%7B%20transform%3A%20props.show%20%3F

APPENDIX

Data Downloading

```
from wwo_hist import retrieve_hist_data
from backbone import getConfig


def getData(start_date=None, end_date=None, location=None):
    config, corePath = getConfig("dataset")
    config = config["api"]
    if start_date is None:
        start_date = config["start_date"]
    if end_date is None:
        end_date = config["end_date"]
    if location is None:
        location = config["location"]
    data = retrieve_hist_data(
        api_key=config["key"],
        location_list=location,
        start_date=start_date,
        end_date=end_date,
        frequency=config["frequency"],
        export_csv=False,
        store_df=True
    )
    dataset = {}
    for i in range(len(location)):
        loc = location[i]
        dataset[loc] = data[i].set_index(
            'date_time').drop(columns=['uvIndex'])

    return dataset
```

Data Processing

```
# dataProcessing.py
import pandas as pd
from dataDownload import getData


class DataProcessor:
    def __init__(self):
        self.loadConfig()

    def loadConfig(self):
        import os
        from backbone import getConfig
```

```
        datasetConfig, corePath = getConfig("dataset")
        datasetDir = os.path.join(corePath, datasetConfig["dataset_dir"])
        self.location = datasetConfig['api']['location'][0]
        self.labels = datasetConfig["labels"]
        self.predictionIndices = [label for label in self.labels]
        self.index = datasetConfig["index"]
        if self.index not in self.predictionIndices:
            self.predictionIndices.insert(0, self.index)
        self.nPrior = datasetConfig["n_prior"]
        self.rawDataFile = os.path.join(
            datasetDir, datasetConfig["raw_dataset_file"])
        self.dataFile = os.path.join(datasetDir, datasetConfig["dataset_file"])
        # print(self.labels)
        # print(self.predictionIndices)

    def __deriveNthPriorFeatures(self, data, feature, n):
        rows = data.shape[0]
        n = n * 24
        nthPriopData = [None]*n + [data[feature][i-n] for i in range(n, rows)]
        col_name = "{}_{}".format(feature, n)
        data[col_name] = nthPriopData

    def processRawData(self, rawDataFile=None):
        if rawDataFile is None:
            rawDataFile = self.rawDataFile
        try:
            data = pd.read_csv(rawDataFile, parse_dates=[self.index])[
                self.predictionIndices].set_index(self.index)
        except FileNotFoundError:
            data = getData()[self.location]
            # self.save(data, file=rawDataFile)

        for feature in self.predictionIndices:
            if feature != self.index:
                for n in range(1, self.nPrior+1):
                    self.__deriveNthPriorFeatures(data, feature, n)

        return data

    def getFeatureLabels(self, data):
        y = pd.DataFrame()
        for label in self.labels:
            y[label] = data.pop(label)
        # data = data.drop(columns=self.labels)
        return data, y
```

```
    def getLabels(self):
        return self.labels

    def save(self, data, file=None):
        if file is None:
            file = self.dataFile
        data = data.dropna()
        data.to_csv(file)

    def load(self, date=None, file=None):
        try:
            if file is None:
                file = self.dataFile
            dataset = pd.read_csv(file, parse_dates=[
                            self.index]).set_index(self.index)
            if date is not None:
                dataset = dataset.loc[pd.datetime(
                    date.year, date.month, date.day, date.hour)]
                dataset = pd.DataFrame(dataset).T
        except KeyError:
            from backbone import getCoreConfig, updateCoreConfig
            from datetime import datetime, timedelta
            start_date = getCoreConfig()['last_date']
            end_date = datetime.now().strftime('%d-%b-%Y')
            dataset = getData(start_date=start_date, end_date=end_date)[
                self.location]
            raw_data = pd.read_csv(self.rawDataFile).set_index(self.index)
            d = pd.concat([raw_data, dataset]).drop_duplicates()
            d.to_csv(self.rawDataFile)
            dataset = self.processRawData()
            self.save(dataset)
            next_date = datetime.strftime(
                datetime.now()+timedelta(1), "%d-%b-%Y")
            updateCoreConfig({"last_date": next_date})

        except FileNotFoundError:
            dataset = self.processRawData()
            self.save(dataset)
        if date is not None:
            dataset = dataset.loc[pd.datetime(
                date.year, date.month, date.day, date.hour)]
            dataset = pd.DataFrame(dataset).T
        return dataset


if __name__ == "__main__":
```

```
from datetime import datetime
dataProcessor = DataProcessor()
print(dataProcessor.load(datetime.now()))
```

Model Training

```
from backbone import getConfig
from dataProcessor import DataProcessor
import pandas as pd
from sklearn.linear_model import LinearRegression
import joblib

from os import path


class ModelTrain:

    def __init__(self):
        config, corePath = getConfig("dataset")
        self.index = config["index"]
        self.labels = config["labels"]
        self.corePath = corePath
        self.modelDir = config["models_dir"]
        self.datasetDir = config["dataset_dir"]

    def setUpDataset(self, train_size=0.8):
        dataProcessor = DataProcessor()
        data = dataProcessor.load()
        dataset = data.dropna()
        dataset_train = dataset.sample(frac=train_size, random_state=42)
        dataset_test = dataset.drop(dataset_train.index)

        X_train, y_train = dataProcessor.getFeatureLabels(dataset_train)
        X_test, y_test = dataProcessor.getFeatureLabels(dataset_test)
        # dataset_train = dataset_train.drop(columns=labels_train.columns)
        # dataset_test = dataset_test.drop(columns=labels_test.columns)

        return X_train, y_train, X_test, y_test

    def trainModel(self):
        try:
            X_train, y_train, X_test, y_test = self.setUpDataset(
                train_size=0.8)
        except FileNotFoundError as e:
            print(e)
        else:
```

```
        for label in self.labels:
            model = LinearRegression(n_jobs=-1)
            model.fit(X_train, y_train[label])
            joblib.dump(model, path.join(self.corePath,
                                self.modelDir, label+"_model.mdl"))
            print(model.score(X_test, y_test[label]))

        return X_test, y_test, self.labels
```

Model Prediction

```
from backbone import getConfig
import joblib
from os import path


class PredictionEngine:

    def __init__(self):
        self.models = {}
        config, corePath = getConfig("dataset")
        self.config = config
        for label in config["labels"]:
            try:
                self.models[label] = joblib.load(
                    path.join(corePath, config["models_dir"], label+"_model.mdl"))
            except FileNotFoundError:
                from modelTrain import ModelTrain
                modelTrain = ModelTrain()
                modelTrain.trainModel()
                self.models[label] = joblib.load(
                    path.join(corePath, config["models_dir"], label+"_model.mdl"))

    def predict(self, X, y, labels=None):
        if labels == None:
            labels = self.config["labels"]

        prediction = {}
        for label in labels:
            prediction[label] = self.models[label].predict(X)[0]
        return prediction
```

UI Components

```
import React, { Component } from "react";
// import Splash from "./components/Splash";
import Window from "./components/Window";
import Splash from "./components/Splash";
import axios from "axios";
const { remote } = window.require("electron");

export default class WeatherApp extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isLoading: true,
      data: null
    };
  }
  componentDidMount() {
    axios
      .get("http://localhost:5000/")
      .then(response => {
        const data = response.data;
        console.log(data);
        this.setState({
          isLoading: false,
          data: data
        });
      })
      .catch(error => {
        console.log(error);
      });
  }
  renderMainWindow() {
    const win = remote.getCurrentWindow();
    win.setSize(1900, 1080, true);
    win.setPosition(0, 0, true);
  }
  handleTheme() {
    const hr = new Date().getHours();
    let theme;
    if (hr >= 6 && hr <= 11) {
      theme = "morning";
    } else if (hr > 11 && hr < 16) {
      theme = "afternoon";
    } else if (hr >= 16 && hr < 19) {
      theme = "evening";
```

```
    } else {
      theme = "night";
    }
    return theme + "-theme";
  }
  render() {
    const theme = this.handleTheme();
    const { isLoading } = this.state;
    let ui = null;
    if (isLoading) {
      ui = (
        <div>
          <Splash />
        </div>
      );
    } else {
      this.renderMainWindow();
      ui = (
        <div className={theme}>
          <Window data={this.state.data} theme={theme} />
        </div>
      );
    }
    return <React.Fragment>{ui}</React.Fragment>;
  }
}
```

Window

```
import React, { Component } from "react";
import TitleBar from "./TitleBar";
import MainContent from "./MainContent";
import Modal from "./Modal";
import PredictionModal from "./PredictionModal";

export default class Window extends Component {
  constructor(props) {
    super(props);

    this.state = {
      showModal: false
    };
  }

  modalShow = () => {
    this.setState({
```

```
      showModal: true
    });
  };

  modalHide = () => {
    this.setState({
      showModal: false
    });
  };
  render() {
    const { prediction, real } = this.props.data;
    console.log(this.props.data);
    const modal = this.state.showModal ? (
      <Modal theme={this.props.theme}>
        <div className='modal'>
          <PredictionModal data={prediction} handleClose={this.modalHide} />
        </div>
      </Modal>
    ) : null;

    return (
      <div className='GridContainer'>
        <div className='TitleBarConatainer'>
          <TitleBar title='Weather Prediction' />
        </div>
        <div className='AnimationContainer'>Animation</div>
        <div className='ContentContainer'>
          <div className='MainContent'>
            <MainContent data={real} handleClick={this.modalShow} />
          </div>
        </div>
        {modal}
      </div>
    );
  }
}
```

Modal

```
import { Component } from "react";
import ReactDOM from "react-dom";

export default class Modal extends Component {
  constructor(props) {
    super(props);
```

```
    this.modalRoot = document.getElementById("modal-root");
    this.element = document.createElement("div");
    this.element.classList.add(this.props.theme);
  }

  componentDidMount() {
    this.modalRoot.appendChild(this.element);
  }
  componentWillUnmount() {
    this.modalRoot.removeChild(this.element);
  }

  render() {
    return ReactDOM.createPortal(this.props.children, this.element);
  }
}
```

GUIDE LINES

Chapter 1 Introduction	it provides the details regarding the introduction of the area to which project belong say image processing, defines the methodology used step by step to achieve the end product, may be diagrammatically. Software tools and technology used like Java introduction and features (at the most 5-6 pages).
Chapter 2 Problem Identification	problem statement, scope of the project, Objective in term of functional and non functional requirements provided by the project, top down or bottom up approach used for achieving the objective and its explanations, a formal SRS document etc. maximum number of pages 20.
Chapter 3 Detail design	it includes various UML diagrams like class diagram, use case diagrams etc., data base design, description about tables, ER diagrams, blue prints (for web based developments), flow charts, pseudo code, work breakdown structure etc.
Chapter 4 Testing	testing should be defined in tables including test case name, test case reference, description, precondition, post condition, result and remarks regarding black box testing and white box testing of unit testing, integration testing and system testing.
Chapter 5 Results	results may be in the form of tables, graphs etc along with proper screen shots and description about the screenshots like its purpose, how this screen will appear etc.
Chapter 6 Deployment	clearly defines the steps required to make settings in the pc to run your project along with the snapshot. Here by snapshot, we does not mean the project snapshot. Its about any setting made in operating system, path setting, directory where project will be stored. In short, anybody can install and run your project without your support by just referencing this chapter.
Conclusion	one page conclusion about your project that includes the summary of your work.
Future Scope	define future improvement, suggestion that helps your juniors to extend your idea and develop the next version.

References	details about links, books, research papers, journals that you refer during the project development.
Appendix A	hardware and software requirement
Appendix B	source code

Note:

- All the pages except front page must be properly numbered.
- All the pages before the chapter must be numbered like i, ii, iii, iv ……
- From chapters 1 onwards, pages must be numbered as 1, 2, 3, 4 …..
- Report must be of at least 50-60 pages excluding Preface, front page, table of content, conclusion, future scope, appendix and references.
- Chapter heading Font size 20, Bold, Right aligned, Times new roman, no spacing and a complete line below.
- Paragraph heading font size 14, Bold, Left aligned, Times new Roman, spacing 6 pts above and below, Line spacing 1.5, not underlined.
- Paragraph content font size 12, justified, Times new roman, 6 pts above and below, 1.5 line spacing, not underlined.
- Number of copies N+3 spiral bound, where N is number of students in group, 2 for department and 1 for project supervisor (in needed by him/her)
- Bring one copy during internal viva and must be checked and approved by supervisor.
- If approved then bring rest of the copies during External viva.
- Margin
    - Left : 1.25 inch
    - Right : 1 inch
    - Top : 1 inch
    - Bottom : 1 inch
- Figures and followed by their name must be center aligned.
- Figures must me numbered as per the chapter number and must be used in the chapter. For example, if there is a third figure of Computer Block diagram in chapter 2 then it must be named as

Figure 2.3: Computer Block Diagram

- Same is the case with any table used.
- Conclusion, future scope, references, appendix A and appendix B will not be mentioned as a chapter, Font Size 20, bold, center aligned, Times New Roman, spacing 12pts above and below, not underlined.

- Use all figures, tables, screen shots properly in explanation properly like as shown in figure 2.3.