1.INTRODUCTION

Air pollution which is detrimental to people's health is a widespread problem across many countries around the world. With the development of the economy and society all over the world, most metropolitan cities are experiencing elevated concentrations of ground-level air pollutants, especially in fast-developing countries like India and China.

Exposure to air pollution can affect everyone, but it can be particularly harmful to people with heart disease or a lung condition both short and long-term exposure to air pollutants has been associated with health impacts. More severe impacts affect people building an early warning system, which provides precise forecasts and also alerts health alarms to local inhabitants will provide valuable information to protect humans from damage by air pollution.

The combined effects of ambient (outdoor) and household air pollution cause about 7million premature deaths every year. This research aims to predict the level of Air Pollution with a set of data used to make predictions. Through them and to obtain the best prediction using several models and compares them to find the appropriate solutions.

To develop robust application using Machine learning algorithms and different techniques using large datasets and find out the optimum solution for Air Quality that helps the human being and It is used to predict the future concentrations of air pollutants in accordance with methodological variables.

The major pollutants area unit oxide (NO), monoxide (CO), stuff (PM), SO2, etc. monoxide is made thanks to the deficient Oxidization of propellant like rock oil, gas, etc. Nitrogen Oxide is made thanks to the ignition of thermal fuel; Carbon monoxide causes headaches, vomiting; aromatic hydrocarbon is made due to smoking, it causes metabolic process problems; gas oxides cause vertigo, nausea; stuff with a diameter of 2.5 micrometres or but that affects additional to human health.

Measures should be taken to reduce air pollution within the atmosphere. Air Quality Index (AQI), is used to measure the standard of air. Earlier classical ways like probability, statistics were accustomed predict the standard of air, but those ways area units terribly complicated to predict the standard of air. Due to the advancement of technology, currently, it's terribly straightforward to fetch the data regarding the pollutants of air exploitation sensors. Assessment of data to notice the pollutants wants vigorous analysis.

PAGE Convolution Neural networks, algorithmic neural www.ijcrt.org © 2021 IJCRT | Volume 9, Issue 7 July 2021 | ISSN: 2320-2882 IJCRT2107446 International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org e211 networks, Deep Learning, Machine learning algorithms assure in accomplishing the prediction of future AQI in order that measures can be taken befittingly.

Machine learning that comes under computing has 3 sorts of learning algorithms, they're supervised Learning, unsupervised learning, reinforcement learning. Within the projected work we tend to have used the supervised

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

The existing system aims to predict air quality using machine learning techniques. It utilizes historical air pollutant data collected from sensors and pre-processes the data by cleaning and normalizing it. The dataset is divided into training and testing sets, and feature selection is applied to identify relevant attributes. Various supervised machine learning algorithms, including Linear Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest, are employed to predict air quality levels. The system's architecture involves data extraction, pre-processing, feature selection, training and testing using machine learning algorithms, and prediction. The goal is to develop a model that accurately predicts air quality levels for improved pollution management and public health protection.

DISADVANTAGES OF EXISTING SYSTEM:

The accuracy of air quality predictions heavily relies on the quality and coverage of sensor data. If the sensor network is sparse or data quality is poor, it can affect the reliability of the predictions.

2.2 PROPOSED SYSTEM:

The proposed system aims to enhance air quality prediction by incorporating advanced machine learning algorithms like Random Forest, decision tree, svm It includes ensemble learning, time-series analysis, and real-time monitoring for adapting to changing data. The system integrates external factors, improves model interpretability, and features a user-friendly interface

ADVANTAGES OF PROPOSED SYSTEM:

By incorporating advanced machine learning algorithms and ensemble techniques, the proposed system can provide more accurate predictions of air quality levels, helping authorities and individuals make informed decisions.

3. SYSTEM STUDY

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the

users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4. SOFTWARE ENVIRONMENT

4.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

4.1.1. Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

Hello, Python!

4.1.2. Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result −

```
Hello, Python!
```

4.1.3. Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
Starting an identifier with a single leading underscore indicates that the identifier is private.
Starting an identifier with two leading underscores indicates a strongly private identifier.
If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

4.1.4. Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not
assert finally or
break for pass
class from print
continue global raise
def if return
del import try
elif in while
else is with
except lambdayield

4.1.5. Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.
The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
else:
```

```
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
import sys
try:
# open file stream
file = open (file_name, "w")
except IO Error:
print "There was an error writing to", file_name
sys. exit()
print "Enter '", file_ finish,
print "' When finished"
while file_ text != file_ finish:
file_ text = raw_ input("Enter text: ")
if file_text == file_finish:
# close the file
file.close
break
file.write(file_text)
file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
print "Next time please enter something"
sys.exit()
try:
file = open(file_name, "r")
except IOError:
print "There was an error reading file"
sys.exit()
```

```
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline
comment.
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

4.1.6. Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

4.1.7. Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara
```

dict['Age']: 7

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section.

4.1.8. Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
```

```
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python
tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup; NameError: name 'tup' is not defined
```

4.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

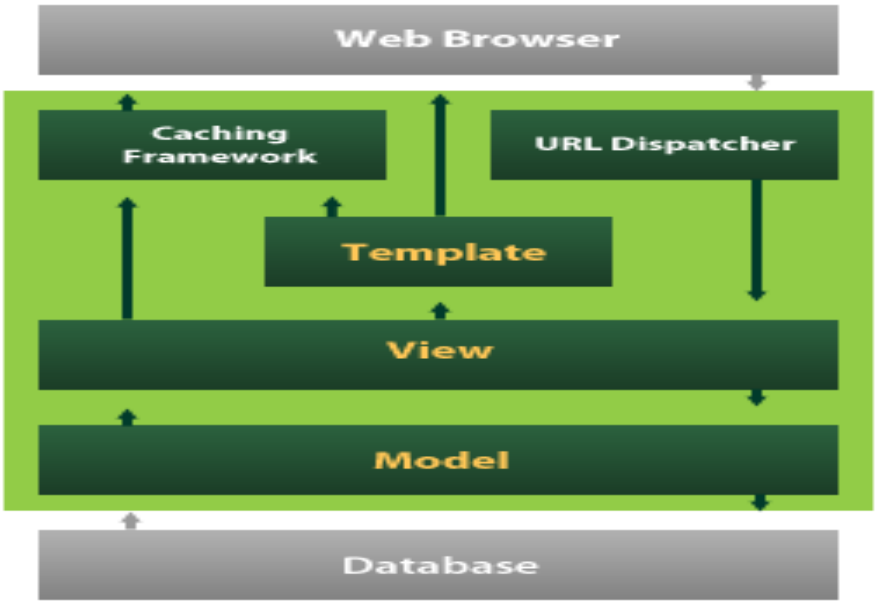


Fig.4.1.Django

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

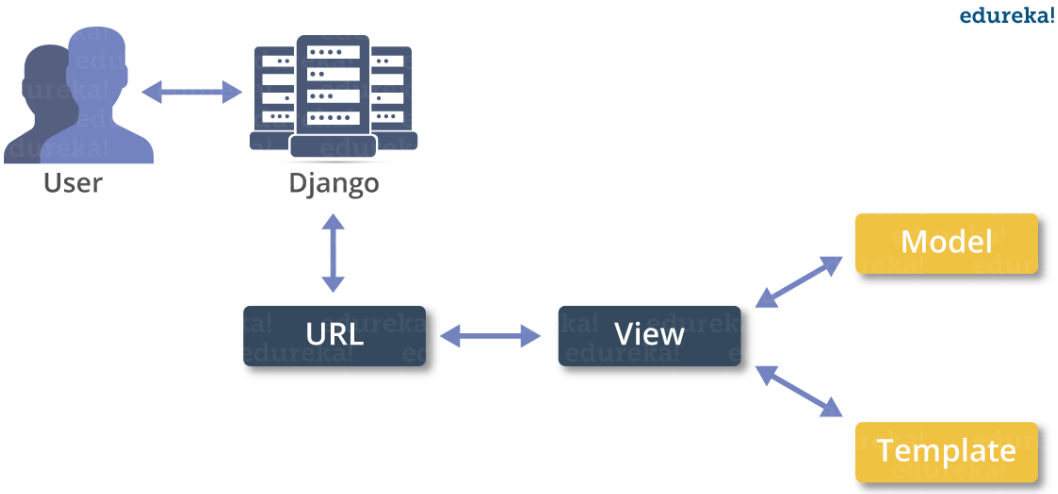


Fig.4.2.Django

4.2.1 Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin start project my project

This will create a "my project" folder with the following structure −

My project/

manage.py

my project/

__init__.py

settings.py

urls.py

wsgi.py

The Project Structure

The “my project” folder is just your project container, it actually contains two elements –

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “my project” subfolder − This folder is the actual python package of your project. It contains four files −

__init__.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder my project/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database. SQL',
    'USER': '',
    'PASSWORD': '',
    'HOST': '',
    'PORT': '',
  }
}
```

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)

PostgreSQL (django.db.backends.postgresql_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

$ python manage.py run server

You will get something like the following on running the above code −

```
Validating models...
0 errors found
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'my project. Settings'
Starting development server at http://127.0.0.1:8000/
```

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

4.2.2 Create an Application

We assume you are in your project folder. In our main “my project” folder, the same folder then manage.py −

$ python manage.py start app my app

You just created my app application and like project, Django create a “my app” folder with the application structure −

My app/

__init__.py

admin.py

models.py

tests.py

views.py

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "my app" application, now we need to register it with our Django project "my project". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

INSTALLED_APPS = (

'Django. contrib. admin',

'Django. contrib. auth',

'Django. contrib. contenttypes',

'Django. contrib .sessions',

'Django. contrib. messages',

'Django. contrib. staticfiles',

'my app',

)

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in my app folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from Django import forms
class Login Form(forms. Form):
user = forms. Char Field(max_length = 100)
password = forms. Char Field(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: Date Input for dates, Checkbox Input for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request. POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from my app forms import LoginForm
def login(request):
username = "not logged in"
if request. method == "POST":
#Get the posted form
My LoginForm = Login Form(request .POST)
if My Login Form. is_ valid():
username = My Login Form. cleaned _data['username']
else:
My Login Form = Login form()
return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
<body>
```

```
<form name = "form" action = "{% URL "myapp.views.login" %}"
method = "POST" >{% csrf_token %}
<div style = "max-width:470px;">
<center>
<input type = "text" style = "margin-left:20%;"
placeholder = "Identifiant" name = "username" />
</center>
</div>
<br>
<div style = "max-width:470px;">
<center>
<input type = "password" style = "margin-left:20%;"
placeholder = "password" name = "password" />
</center>
</div>
<br>
<div style = "max-width:470px;">
<center>
<button style = "border:0px; background-color:#4285F4; margin-top:8%;
height:35px; width:80%;margin-left:19%;" type = "submit"
value = "Login" >
<strong>Login</strong>
</button>
</centre>
</div>
</form>
</body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

<html>
<body>
You are : {{username}}
</body>
</html>
Now, we just need our pair of URLs to get started: myapp/urls.py
from django.conf.urls import patterns, url
from django.views.generic import Template View
URL patterns = patterns('myapp.views',
url(r'^connection/',Template View. as_ view (template name = 'login.html')),
url(r'^login/', 'login', name = 'login'))
When accessing "/myapp/connection", we will get the following login.html template rendered –
Setting Up Sessions
In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −
'django.contrib.sessions.middleware.SessionMiddleware'
And INSTALLED_APPS should have −
'django.contrib.sessions'
By default, Django saves session information in database (Django session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.
When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.
Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.
For this, first lets change our login view to save our username cookie server side −
def login(request):
username = 'not logged in'

```
if request. Method == 'POST':
MyLoginForm = LoginForm(request. POST)
if My Login Form. is_ valid():
username = My Login Form. Cleaned _data['username']
request. session['username'] = username
else:
MyLoginForm = LoginForm ()
return render(request, 'loggedin.html', {"username" : username}
```

Then let us create form View view for the login form, where we won't display the form if cookie is set −

```
def form View(request):
if request.session.has_key('username'):
username = request. session['username']
return render(request, 'loggedin.html', {"username" : username})
else:
return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import Template View
URL patterns = patterns('myapp.views',
  url(r'^connection/','form View', name = 'login form'),
  url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

5.SYSTEM DESIGN

5.1 DATA FLOW

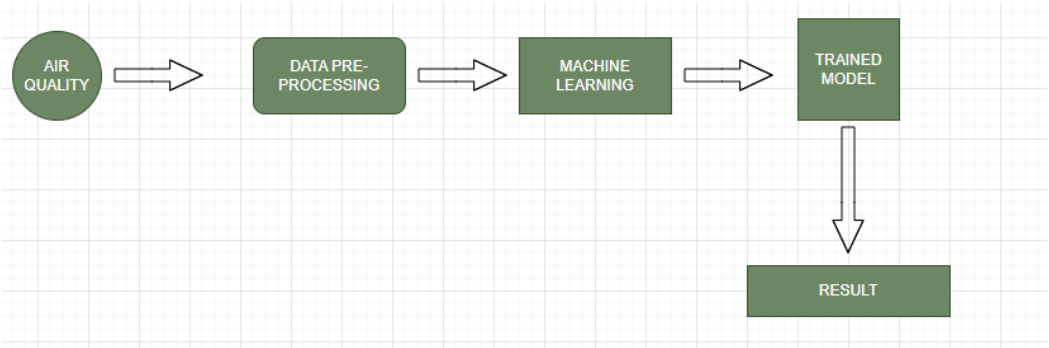


Fig:5.1.1Data Flow Diagram

5.2 USE CASE:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagramis to show what system functions are performed for which actor. Roles of the can be despite.

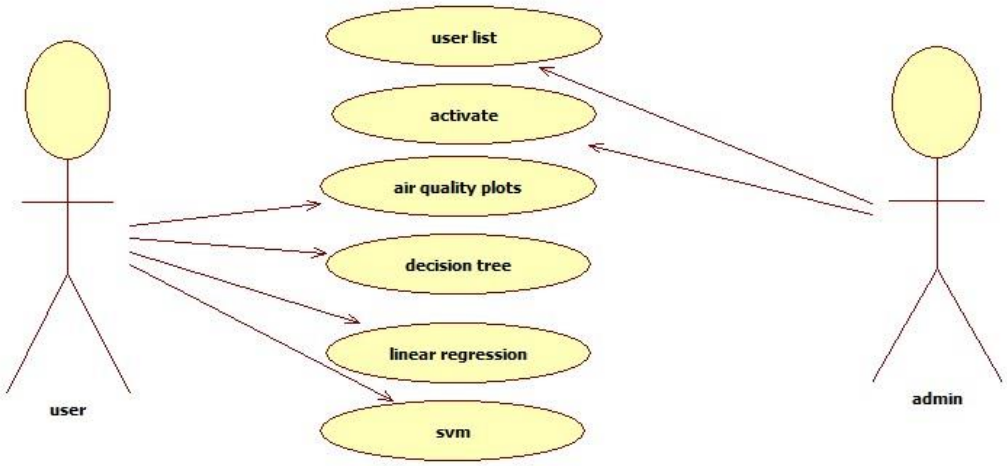
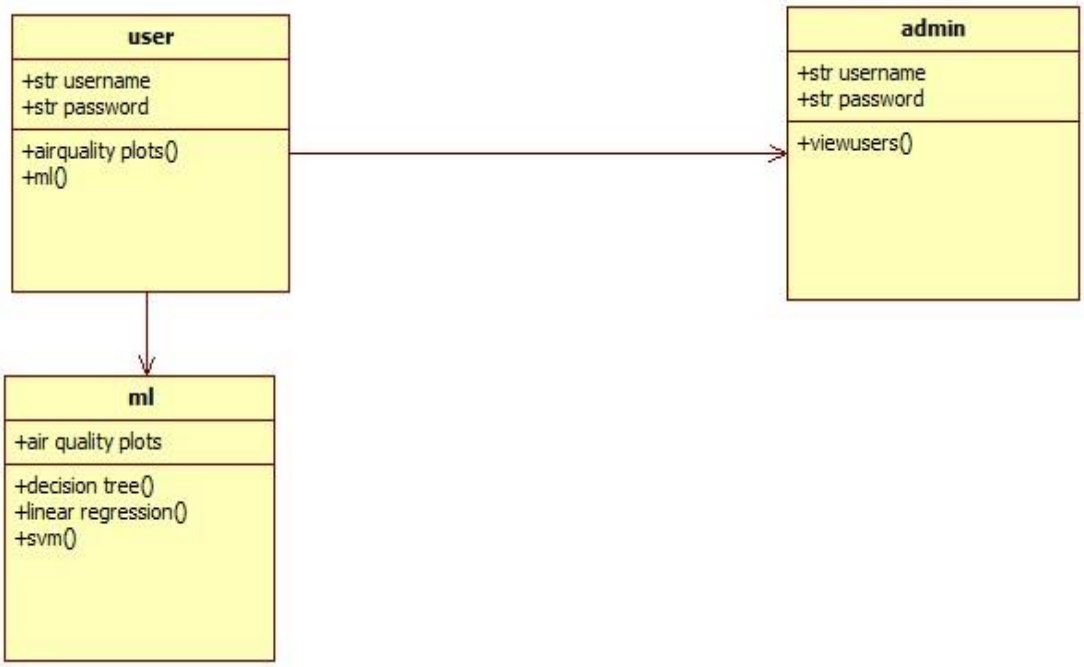


Fig:5.1.2 use case

5.3 CLASS DIAGRAMS:

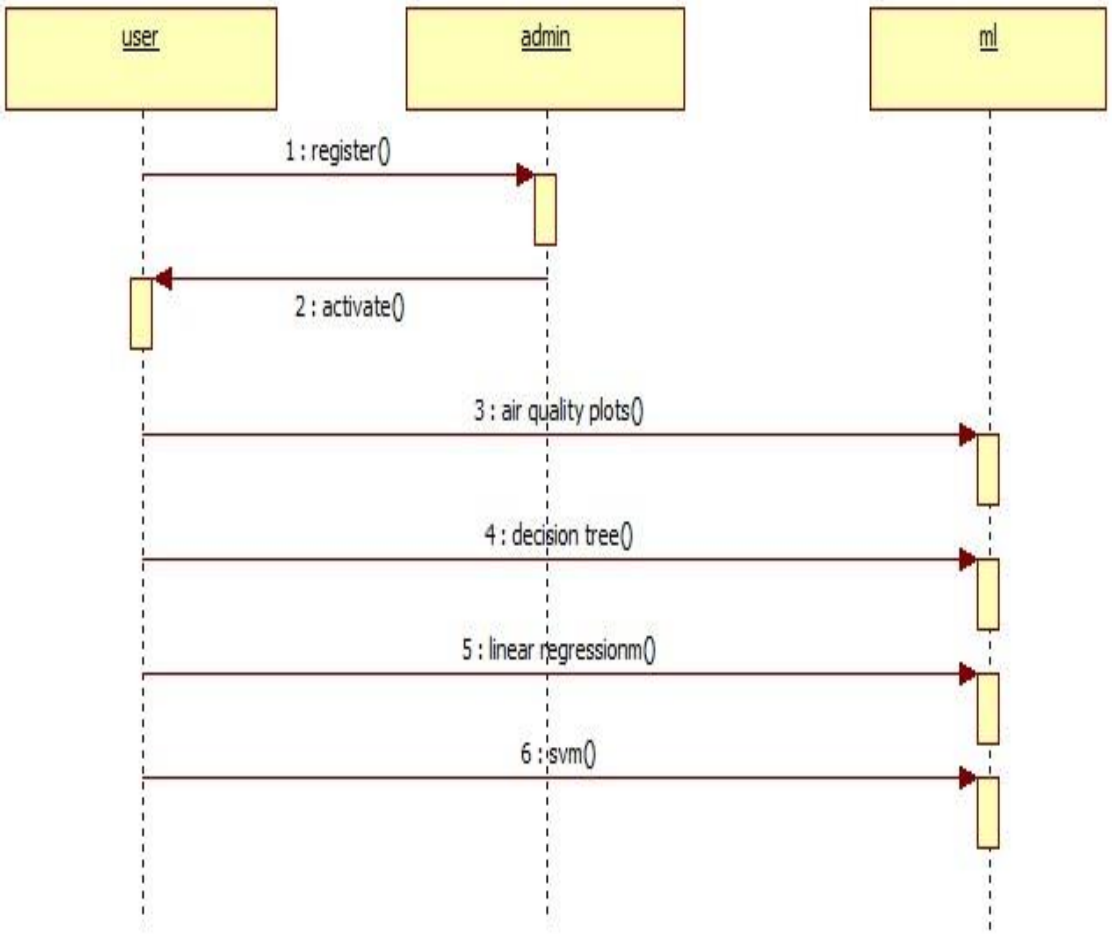
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



5.1.3 CLASS DIAGRAM

5.4 SEQUENCE DIAGRAM:

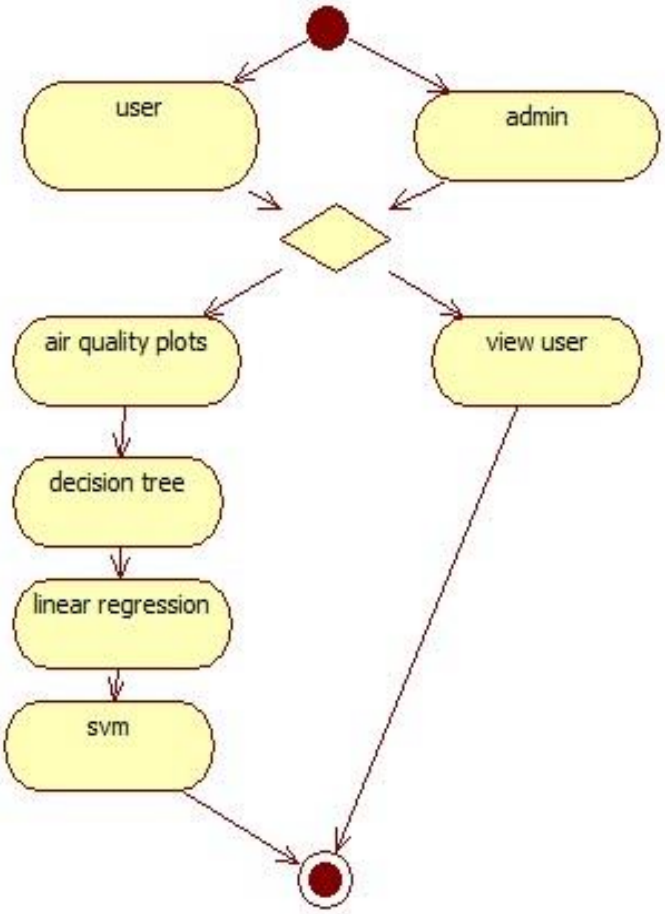
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



5.1.4 SEQUENCE DIAGRAM

5.5 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Anactivity diagram shows the overall flow of control.



5.1.5 ACTIVITY DIAGRAM

6. INPUT AND OUTPUT DESIGN

6.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

6.2OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements

Select methods for presenting information. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

7. SYSTEM REQURIMENTS

7.1 HARDWARE REQUIREMENTS:

- **System** : Intel i5
- **Hard Disk** : 256GB
- **Monitor** : 15'' LED
- **Mouse** : Optical Mouse.
- **Ram** : 8GB.

7.2 SOFTWARE REQUIREMENTS:

- **Operating system** : Windows 10.
- **Coding Language** : Python.
- **Designing** : Html, CSS, Django.
- **Data Base** : SQLite.

HARDWARE AND SOFTWARE REQUIREMENTS

REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

REQUIREMENT SPECIFICATION

Functional Requirements

- Graphical User interface with the User.

Software Requirements

For developing the application the following are the Software Requirements:

1. Python
2. Django

Operating Systems supported

1. Windows 10 64 bit OS

Technologies and Languages used to Develop

1. Python

Debugger and Emulator

- Any Browser (Particularly Chrome)

Hardware Requirements

For developing the application the following are the Hardware Requirements:

- Processor: Intel i9
- RAM: 32 GB
- Space on Hard Disk: minimum 1 TB

8. SAMPLE SOURCE CODE

Source code :

```
from django.shortcuts import render, HTTP Response
from django.contrib import message
from django.http import JsonResponse
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import Label Encoder
# views.py
from .algorithms. Algorithm import process_data, calculate_mse
from django.http import JsonResponse
from sklearn.ensemble import RandomForestClassifier
# Define your view function for the decision tree model training and metrics calculation
from .algorithms. Algorithm import decision
from sklearn.model_selection import train_test_split
import subprocess
import os
from .forms import UserRegistrationForm
from .models import UserRegistrationModel
from django.conf import settings
import numpy as np
import matplotlib.pyplot as plt
from  sklearn. metricsimport
confusion_matrix,accuracy_score,precision_score,recall_score,roc_auc_score,f1_score
# Create your views here.
def UserRegisterActions(request):
if request.method == 'POST':
form = UserRegistrationForm(request.POST)
if form.is_valid():
print('Data is Valid')
form. Save()
messages. Success(request, 'You have been successfully registered')
```

```
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})
else:
messages. Success(request, 'Email or Mobile Already Existed')
print("Invalid form")
else:
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})
def UserLoginCheck(request):
if request.method == "POST":
login id = request.POST. Get('loginid')
pswd = request.POST. Get('pswd')
print("Login ID = ", loginid, ' Password = ', pswd)
try:
check = UserRegistrationModel.objects.get(
loginid=loginid, password=pswd)
status = check. Status
print('Status is = ', status)
if status == "activated":
request.session['id'] = check.id
request.session['logged user'] = check.name
request.session['loginid'] = loginid
request.session['email'] = check .email
print("User id At", check.id, status)
return render(request, 'users/UserHomePage.html', {})
else:
messages .success(request, 'Your Account Not at activated')
return render(request, 'UserLogin.html')
except Exception as e:
print('Exception is ', str(e))
pass
messages. success(request, 'Invalid Login id and password')
return render(request, 'UserLogin.html', {})
def User Home(request):
```

```
return render(request, 'users/UserHomePage.html', {})
from sklearn.metrics import mean_squared_error
def predict_ aqi_ bucket(request):
if request. Method == 'POST':
# Assuming you have a form where users input the feature values
# For simplicity, I'll assume the form fields are 'pm25', 'pm10', 'no', etc.
pm25 = float(request. POST['pm25'])
pm10 = float(request. POST['pm10'])
no = float(request. POST['no'])
no2 = float(request. POST['no2'])
nox = float(request. POST['nox'])
nh3 = float(request. POST['nh3'])
co = float(request. POST['co'])
so2 = float(request. POST['so2'])
o3 = float(request. POST['o3'])
# Create a Data Frame with the input data
input data = pd. Data Frame({
'PM2.5': [pm25],
'PM10': [pm10],
'NO': [no],
'NO2': [no2],
'NOx': [nox],
'NH3': [nh3],
'CO': [co],
'SO2': [so2],
'O3': [o3]
})
# Load the trained model and Label Encoder
model, le, X_train, X_test, y_train, y_test = process_data(request)
# Make predictions on the new data
X_new = input_data.values
predicted_bucket = model.predict(X_new)
predicted_bucket_label = le.inverse_transform(predicted_bucket.astype(int))[0]
# Calculate the Mean Squared Error on the training data
```

```
y_train_pred = model.predict(X_train)
mse_train = mean_squared_error(y_train, y_train_pred)
# Optionally, calculate the Mean Squared Error on the test data
y_test_pred = model.predict(X_test)
mse_test = mean_squared_error(y_test, y_test_pred)
return render(request, 'users/result.html', {
'predicted_bucket': predicted_bucket_label,
'mse_train': mse_train,
'mse_test': mse_test
})
return render(request, 'users/prediction_form.html')
import pandas as pd
import seaborn as sns
import plotly.express as px
from django.shortcuts import render
def air_quality_pair_plots(request):
# Load the air quality data into a Data Frame
path = settings.MEDIA_ROOT + "//" + 'city_day.csv'
data = pd.read_csv(path)
# Assuming 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3' are columns of interest
columns_of_interest = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3']
# Create a subset of the data with only the columns of interest
subset_data = data[columns_of_interest]
# Create pair plots using seaborn
sns.set(style='ticks')
pair_plot = sns.pairplot(subset_data)
# Convert the pair plots to a Plotly figure
fig = px.imshow(pair_plot.data)
# Get the plot's HTML representation
plot_html = fig.to_html(full HTML=False, include_plotlyjs='cdn')
# Render the template with the interactive plot
return render(request, 'users/pair_plots.html', {'plot_html': plot_html})
def decision tree(request):
```

```
from sklearn import metrics
path = settings.MEDIA_ROOT + "//" + 'city_day.csv'
d = pd.read_csv(path)
# Your data preprocessing code
pmean = d["PM2.5"].mean()
d["PM2.5"].fillna(pmean, inplace=True)
pmmean = d["PM10"].mean()
d["PM10"].fillna(pmmean, inplace=True)
nmean = d["NO"].mean()
d["NO"].fillna(nmean, inplace=True)
nomean = d["NO2"].mean()
d["NO2"].fillna(nomean, inplace=True)
noxmean = d["NOx"].mean()
d["NOx"].fillna(noxmean, inplace=True)
   nhmean = d["NH3"].mean()
d["NH3"].fillna(nhmean, inplace=True)
cmean = d["CO"].mean()
d["CO"].fillna(cmean, inplace=True)
smean = d["SO2"].mean()
d["SO2"].fillna(smean, inplace=True)
omean = d["O3"].mean()
d["O3"].fillna(omean, inplace=True)
# Drop the 'Benzene' and 'Toluene' columns if they are present in the DataFrame
if 'Benzene' in d.columns:
d.drop('Benzene', axis=1, inplace=True)
if 'Toluene' in d.columns:
d.drop('Toluene', axis=1, inplace=True)
if 'Xylene' in d.columns:
d.drop('Xylene', axis=1, inplace=True)
le = LabelEncoder()
d['AQI_Bucket'] = le.fit_transform(d['AQI_Bucket'])
# Separate features (X) and target variable (y)
X = d.iloc[:, 2:11].values
y = d['AQI_Bucket'].values
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_ size=0.3, random state=0)
Regressor = DecisionTreeRegressor()
Regressor. fit(X_train, Y_train)
prediction = Regressor predict(X_test)
mae = metrics.mean_absolute_error(Y_test, prediction)
mse = metrics.mean_squared_error(Y_test, prediction)
rmse = np.sqrt(mse)
return render(request, 'users/dt.html', {
'mae': mae,
'mse': mse,
'rmse': rmse,
})
def lr(request):
from sklearn import metrics
from sklearn.linear_model import Linear Regression
path= settings.MEDIA_ROOT + "//" + 'city_day.csv'
d = pd.read_csv(path)
# Your data preprocessing code
pmean = d["PM2.5"].mean()
d["PM2.5"].fillna(pmean, inplace=True)
pmmean = d["PM10"].mean()
d["PM10"].fillna(pmmean, inplace=True)
nmean = d["NO"].mean()
d["NO"].fillna(nmean, inplace=True)
nomean = d["NO2"].mean()
d["NO2"].fillna(nomean, inplace=True)
noxmean = d["NOx"].mean()
d["NOx"].fillna(noxmean, inplace=True)
nhmean = d["NH3"].mean()
d["NH3"].fillna(nhmean, inplace=True)
cmean = d["CO"].mean()
d["CO"].fillna(cmean, inplace=True)
smean = d["SO2"].mean()
d["SO2"].fillna(smean, inplace=True)
```

```
omean = d["O3"].mean()
d["O3"].fillna(omean, inplace=True)
# Drop the 'Benzene' and 'Toluene' columns if they are present in the Data Frame
if 'Benzene' in d.columns:
d.drop('Benzene', axis=1, inplace=True)
if 'Toluene' in d.columns:
d.drop('Toluene', axis=1, inplace=True)
if 'Xylene' in d.columns:
d.drop('Xylene', axis=1, inplace=True)
le = Label Encoder()
d['AQI_Bucket'] = le.fit_transform(d['AQI_Bucket'])
# Separate features (X) and target variable (y)
X = d.iloc[:, 2:11].values
y = d['AQI_Bucket'].values
X_ train, X_ test, Y_ train, Y_ test = train _test_ split (X, y, test_ size=0.3, random state=0)
Regressor = Linear Regression()
Regressor. fit(X_ train, Y_ train)
prediction = Regressor .predict(X_ test)
mae = metrics.mean_absolute_error(Y_ test, prediction)
mse = metrics.mean_squared_error(Y_ test, prediction)
rmse = np.sqrt(mse)
return render(request, 'users/lr.html', {
'mae': mae,
'mse': mse,
'rmse': rmse,
})
def svm(request):
from sklearn import metrics
from sklearn.svm import SVR
path = settings.MEDIA_ROOT + "//" + 'city_day.csv'
d = pd.read_csv(path)
# Your data preprocessing code
pmean = d["PM2.5"].mean()
d["PM2.5"].fillna(pmean, inplace=True)
```

```
pmmean = d["PM10"].mean()
d["PM10"].fillna(pmmean, inplace=True)
nmean = d["NO"].mean()
d["NO"].fillna(nmean, inplace=True)
nomean = d["NO2"].mean()
d["NO2"].fillna(nomean, inplace=True)
noxmean = d["NOx"].mean()
d["NOx"].fillna(noxmean, inplace=True)
nhmean = d["NH3"].mean()
d["NH3"].fillna(nhmean, inplace=True)
cmean = d["CO"].mean()
d["CO"].fillna(cmean, inplace=True)
smean = d["SO2"].mean()
d["SO2"].fillna(smean, inplace=True)
omean = d["O3"].mean()
d["O3"].fillna(omean, inplace=True)
# Drop the 'Benzene' and 'Toluene' columns if they are present in the Data Frame
if 'Benzene' in d.columns:
d.drop('Benzene', axis=1, inplace=True)
if 'Toluene' in d.columns:
d.drop('Toluene', axis=1, inplace=True)
if 'Xylene' in d.columns:
d.drop('Xylene', axis=1, inplace=True)
le = Label Encoder()
d['AQI_Bucket'] = le.fit_transform(d['AQI_Bucket'])
# Separate features (X) and target variable (y)
X = d.iloc[:, 2:11].values
y = d['AQI_Bucket'].values
X_ train, X_ test, Y_ train, Y_ test = train _test _split (X, y, test_ size=0.3, random_ state=0)
Regressor = SVR()
Regressor. fit (X _train, Y_ train)
prediction = Regressor .predict(X_ test)
mae = metrics. mean_absolute_error(Y_ test, prediction)
mse = metrics. mean_squared_error(Y_ test, prediction)
```

```
rmse = np.sqrt(mse)
return render(request, 'users/svm.html', {
'mae': mae,
'mse': mse,
'rmse': rmse,
})
```

9.MODULES:

- User
- Admin
- svm
- decision tree

MODULES DESCRIPTION:

User:

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user registers, then admin can activate the customer. Once the admin activates the customer then the customer can login into our system. After login he can add the data to predict the traffic prediction . After adding the data we can find the prediction of the algorithm. First we can find the svm algorithm and then we can find the random forest algorithm.

Admin:

Admin can login with his credentials. Once he logs in he can activate the users. The activated user only login in our applications. The admin can set the predictions of algorithms.Admin can predict random forest algorithms and also predict the support vector machine algorithm. The admin can add new data to the dataset. .

svm:

Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you

have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well

Decision Trees:

Versatile machine learning tools for classification and regression tasks. They represent decisions as a tree structure, guiding through feature-based choices to predict outcomes. Decision Trees are interpretable, prone to overfitting (addressed through pruning), and can be enhanced via ensemble methods like Random Forest and Gradient Boosting.

10. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement

.

10.1. TYPES OF TESTS

10.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

10.1.2. Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

10.1.3. Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

10.1.4. System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

10.1.5. White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

10.1.6. Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

10.1.7. Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

10.2 Test strategy and approach

- Field testing will be performed manually and functional tests will be written in detail.

10.2.1 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

10.2.2 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

10.2.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

10.2.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

11. SAMPLE TEST CASES

S.no	Test Case	Excepted Result	Result	Remarks (IF fails)
1.	User Register	If User registration successfully.	Pass	If already user email exists then it fails.
2.	User Login	If the Username and password is correct then it will be a valid page.	Pass	Un Register Users will not log in.
3.	Admin Add the Data	A new record will be added to our dataset.	Pass	According to India meteorological repository the data must be float fields otherwise it fails.
4.	user add data	user will add the data to find out location prediction based on details	Pass	we couldn't get the required data to predict...
5.	random forest	will get the random forest accuracy results.	Pass	won't get accuracy
6.	svm	will get the svm result	Pass	won't get accuracy
7.	Linear regression	Will get linear regression accuracy	pass	won't get accuracy
8.	Decision tree	Will get decision tree accuracy	Pass	won't get accuracy
9.	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allow here
10.	Admin can activate the register users	Admin can activate the register user id	Pass	If the user did not find it then it won't login.

12.SCREEN SHOTS

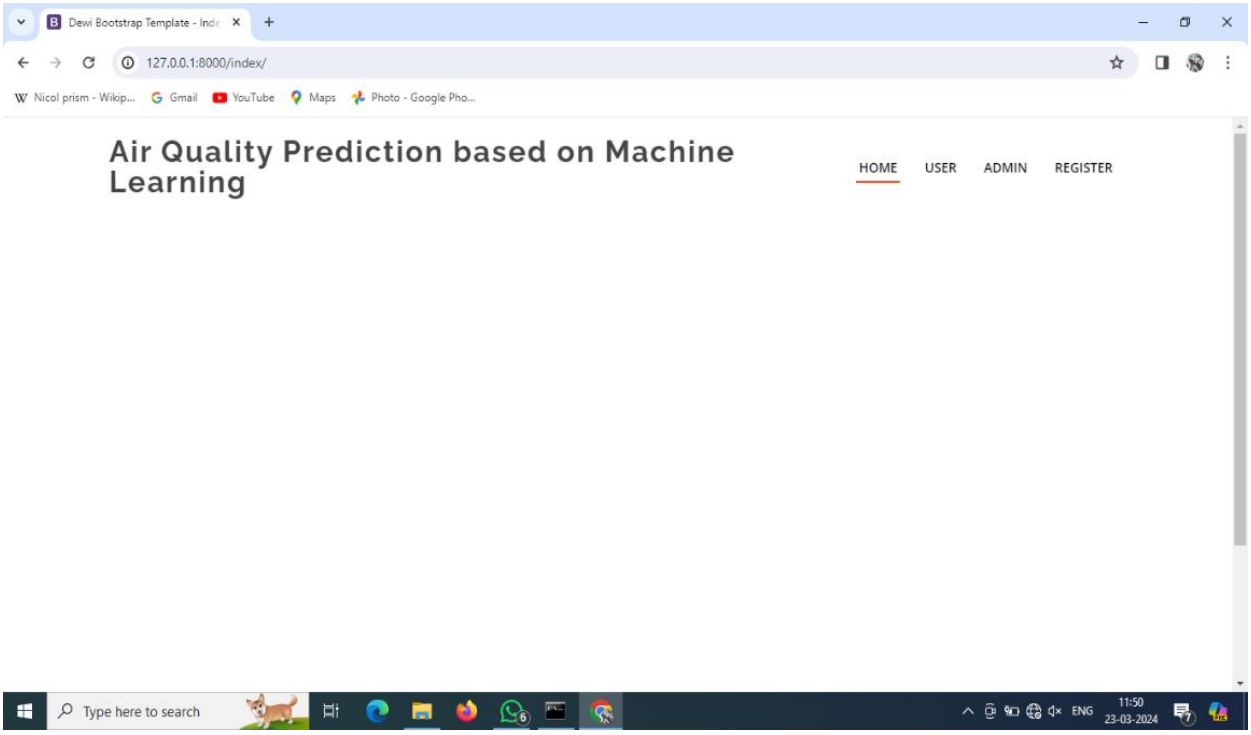


Fig:12.1. In this home page we want to login or Register

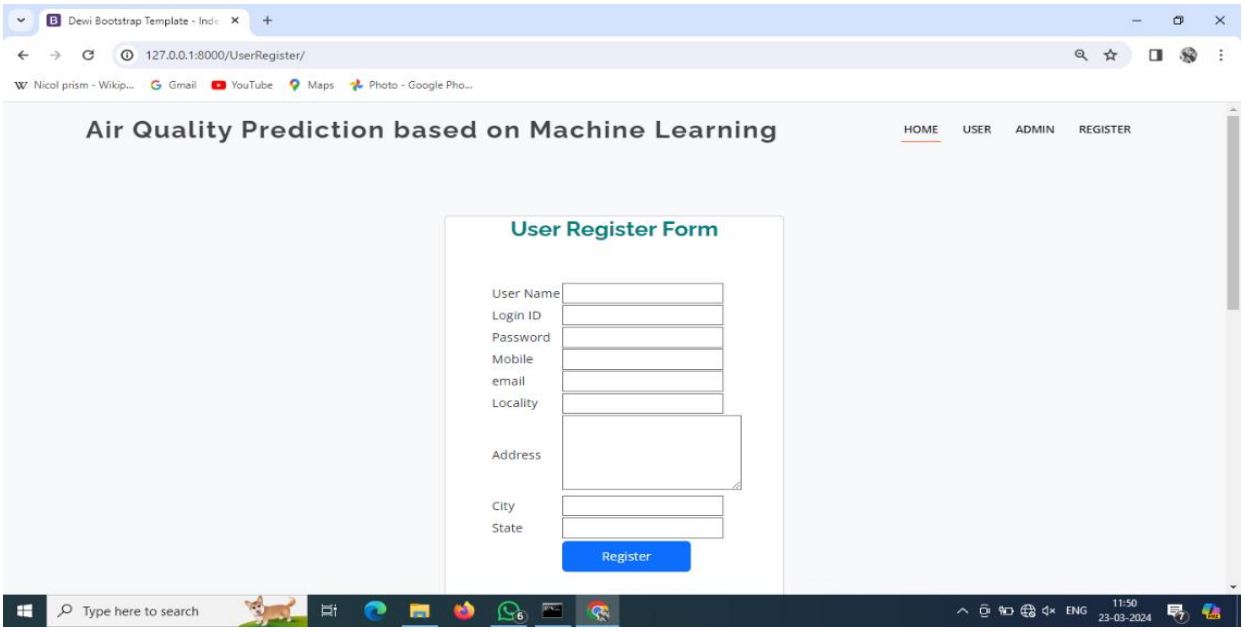


Fig:12.2 In this form to add our details to resister

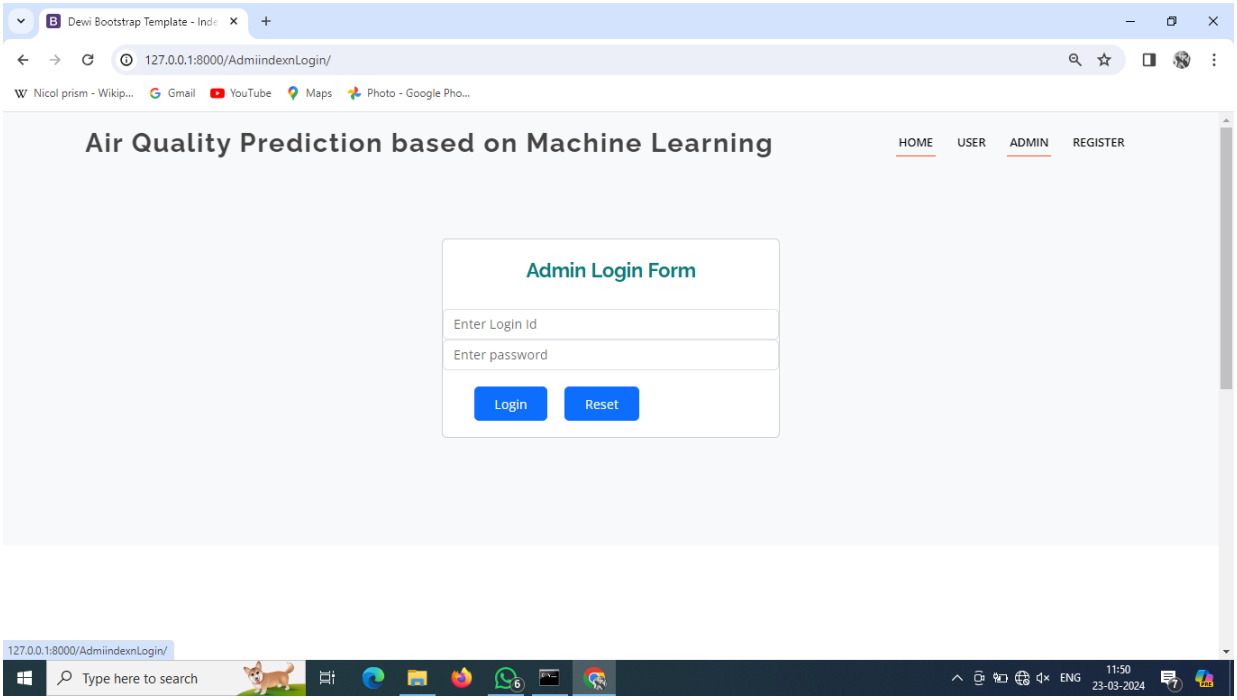


Fig:12.3 In this admin to login him details

Admin home:

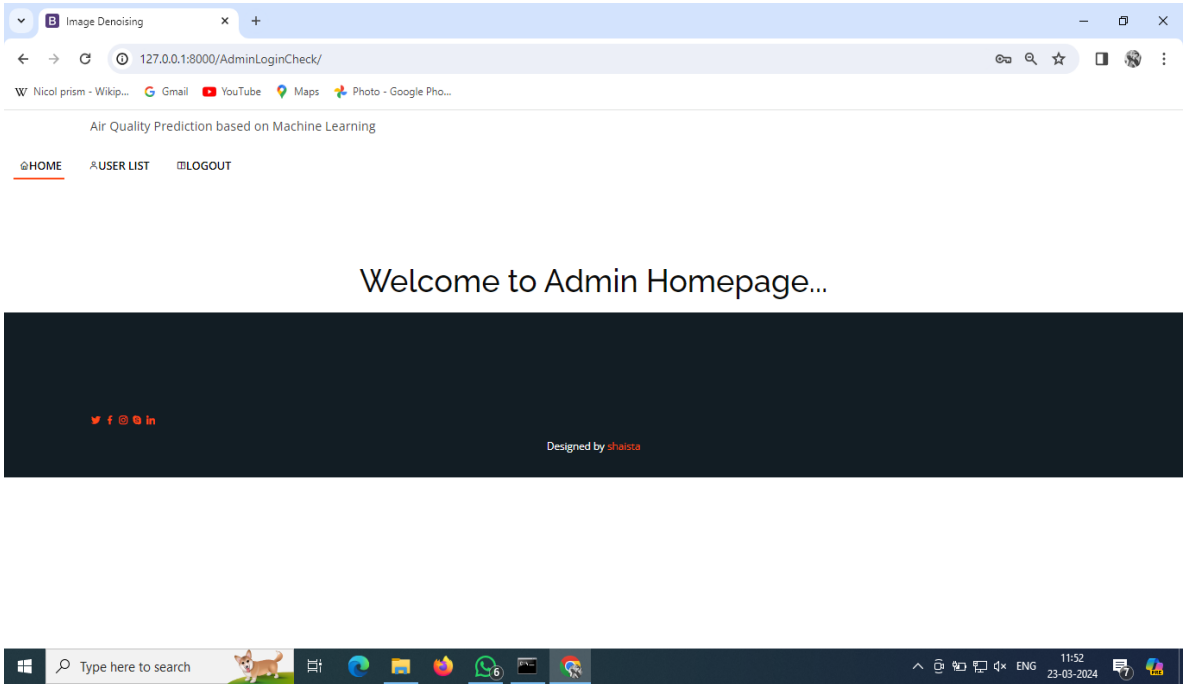


Fig:12.4 In this admin home page to activate the user details

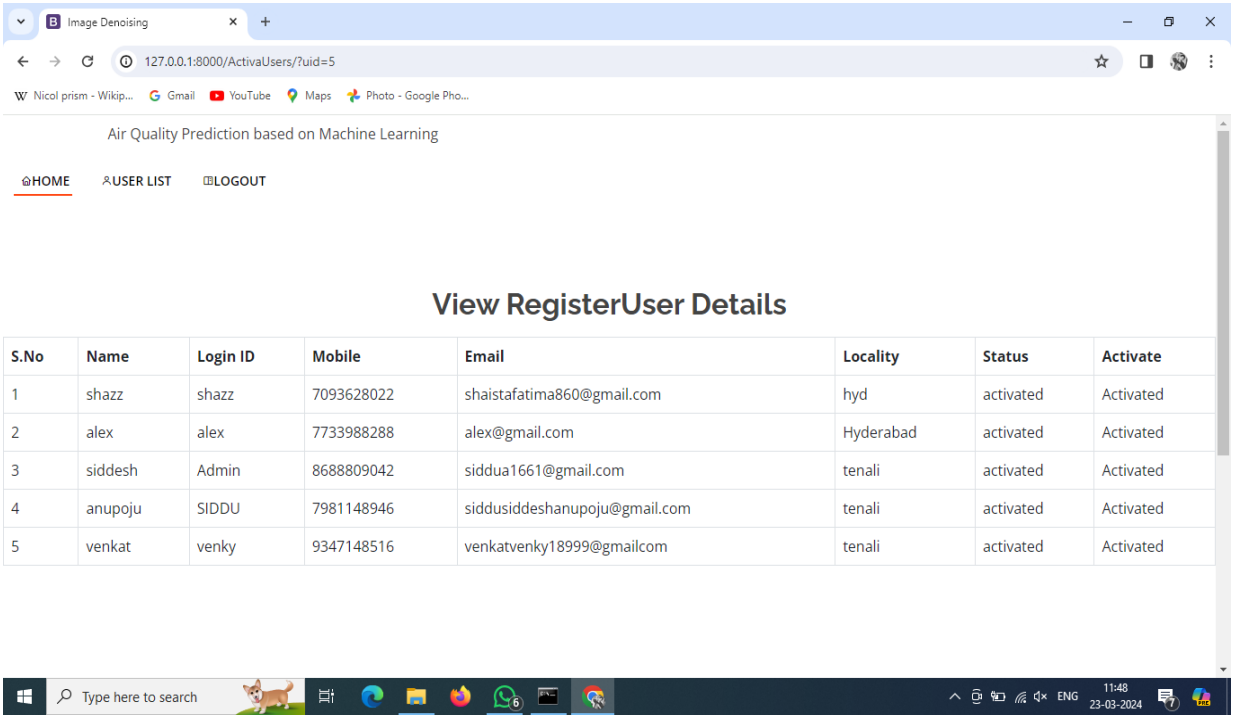


Fig12.5 In this view user details to activate

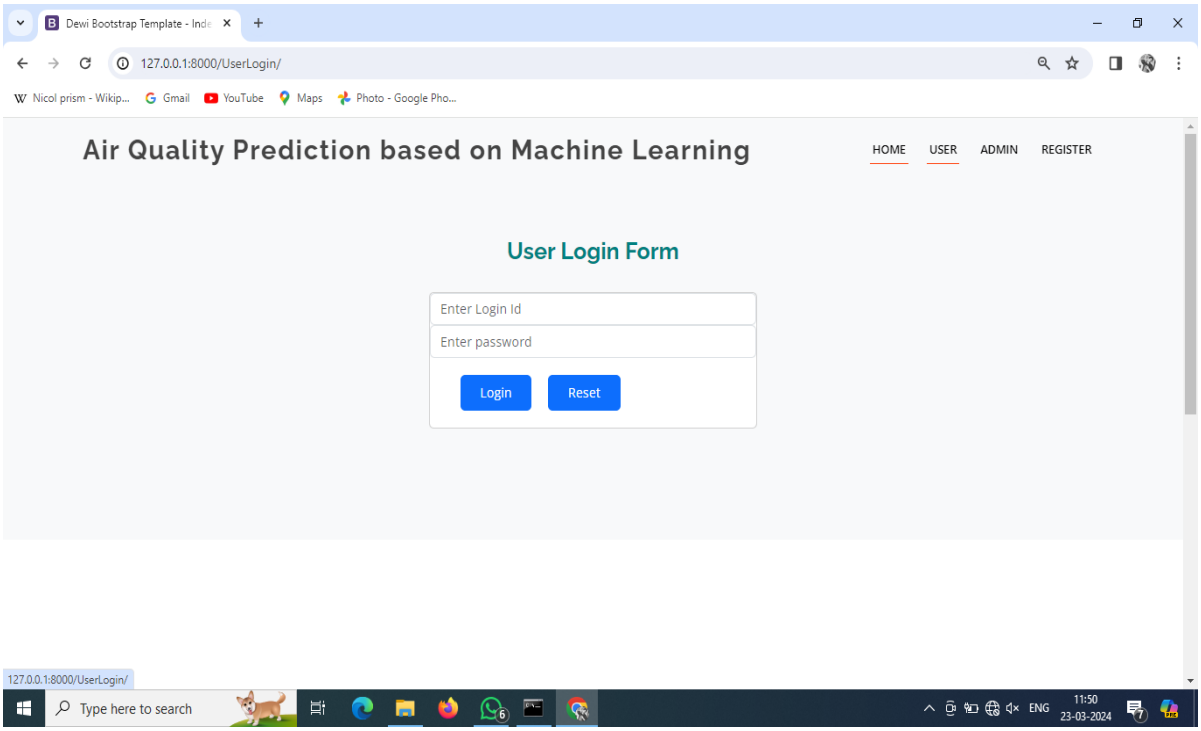


Fig:12.6 In this user login page to check the view

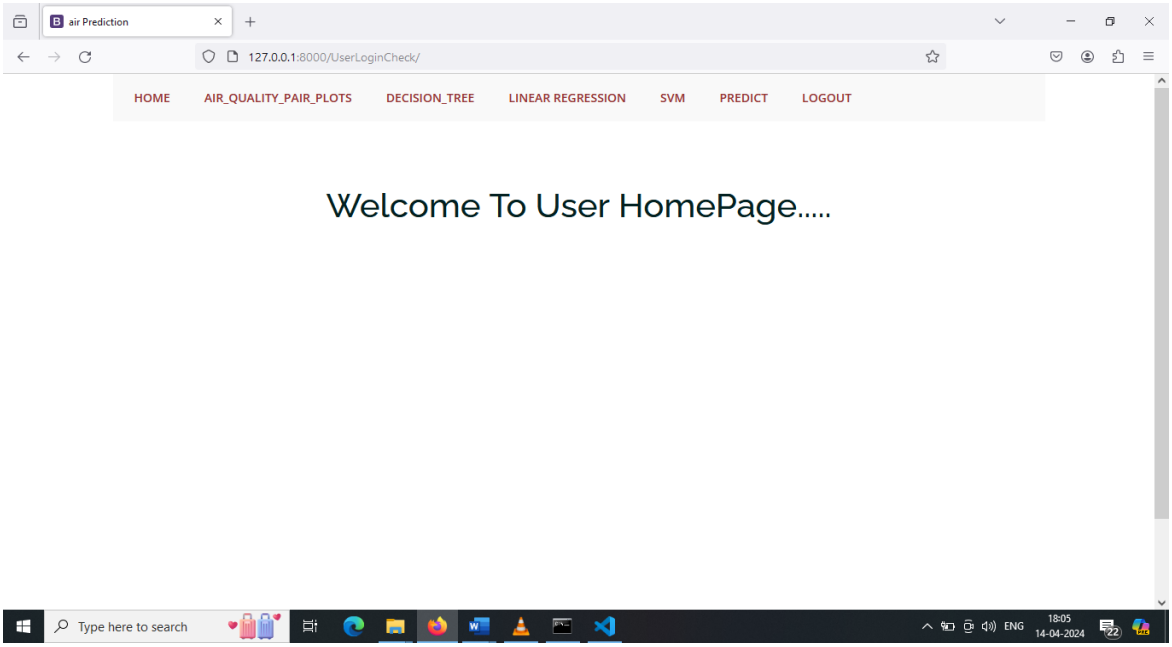


Fig:12.7 User home page to start the value predict

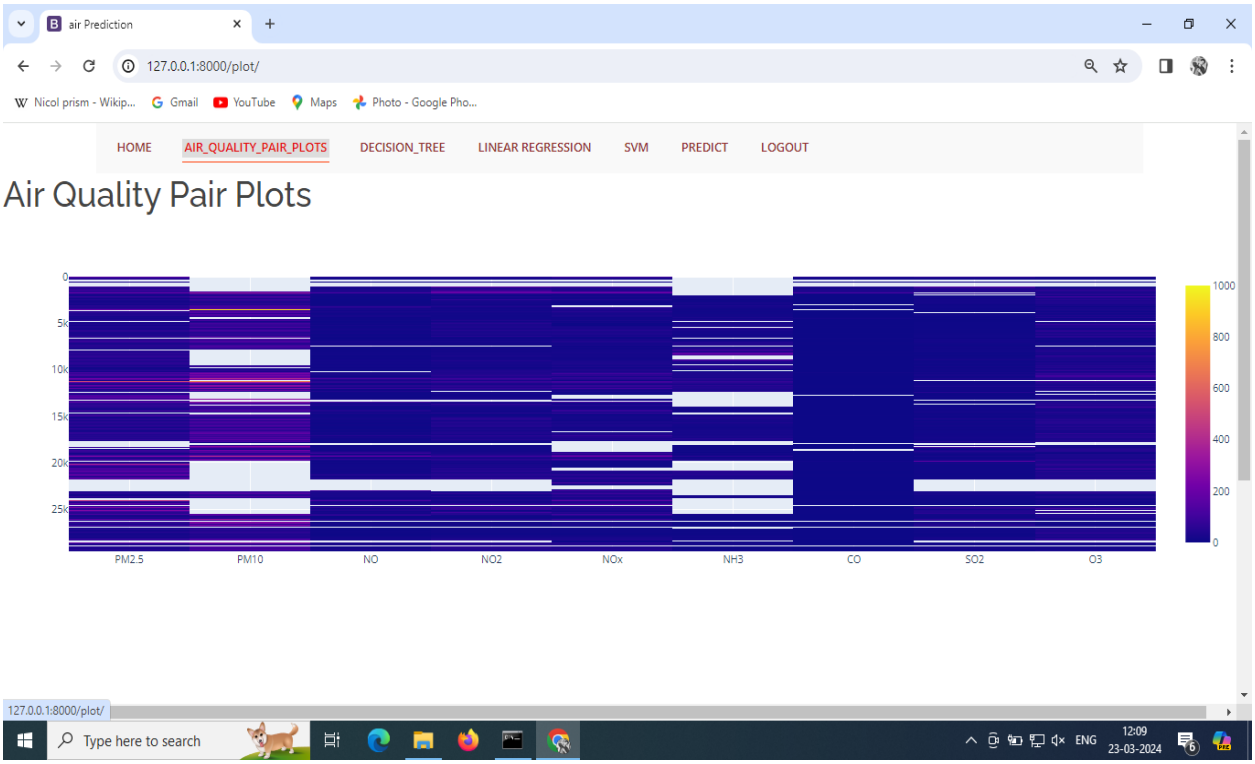


Fig:12.8 Air quality plots values checking

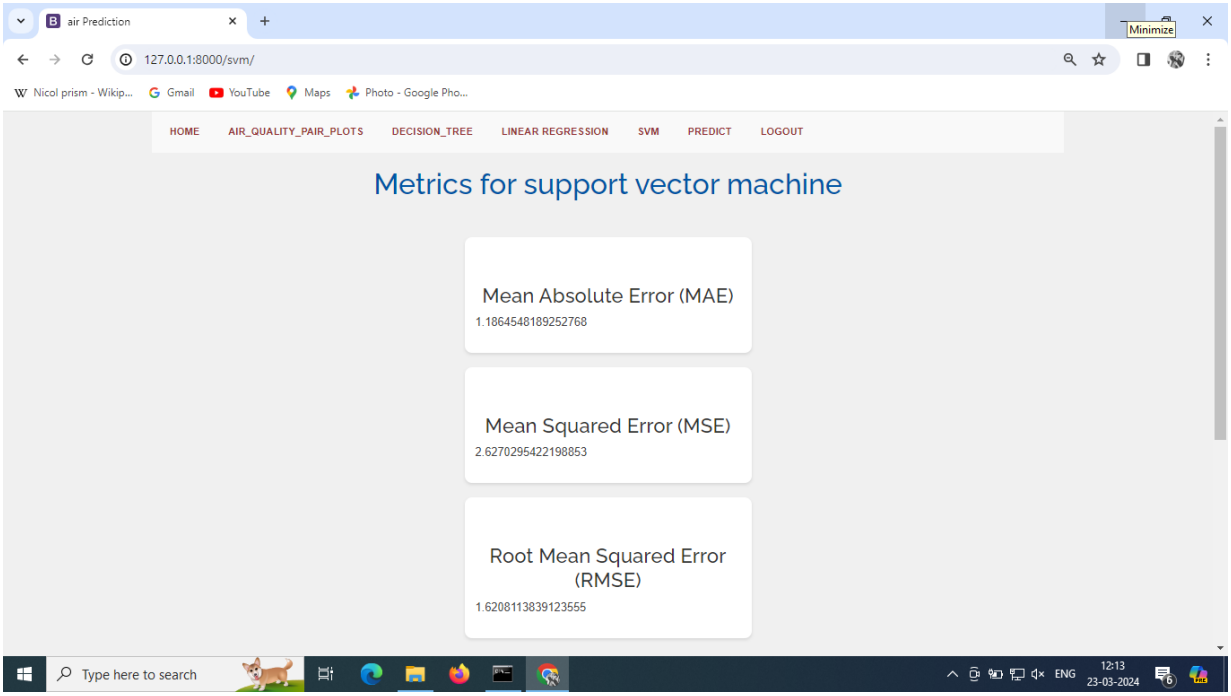


Fig:12.9 Svm by using the formula

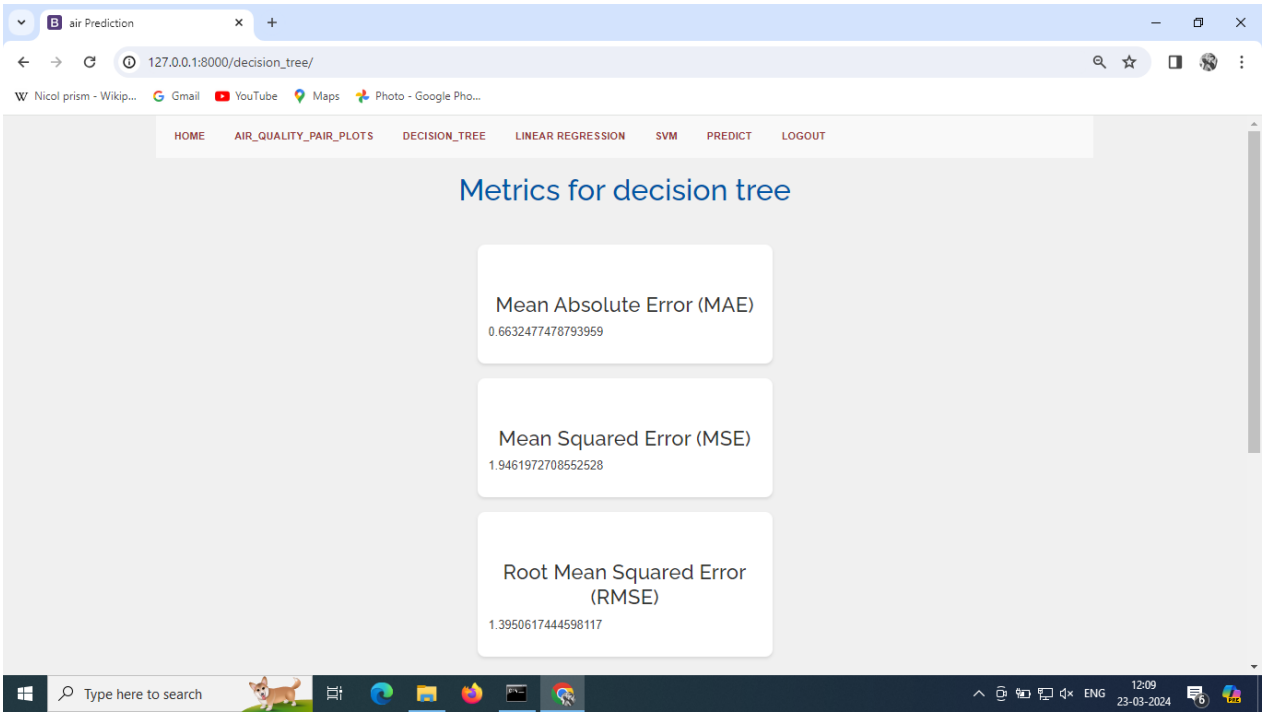


Fig:12.10 decision tree by using formula

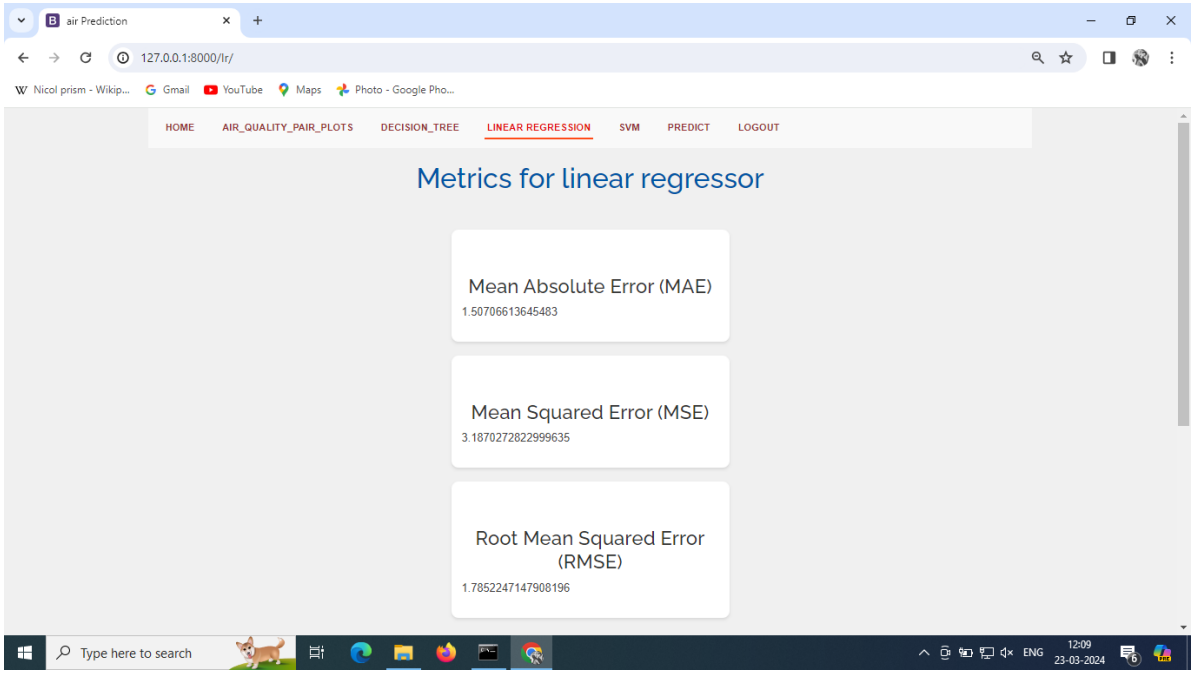


Fig:12.11 linear regressor by using formula

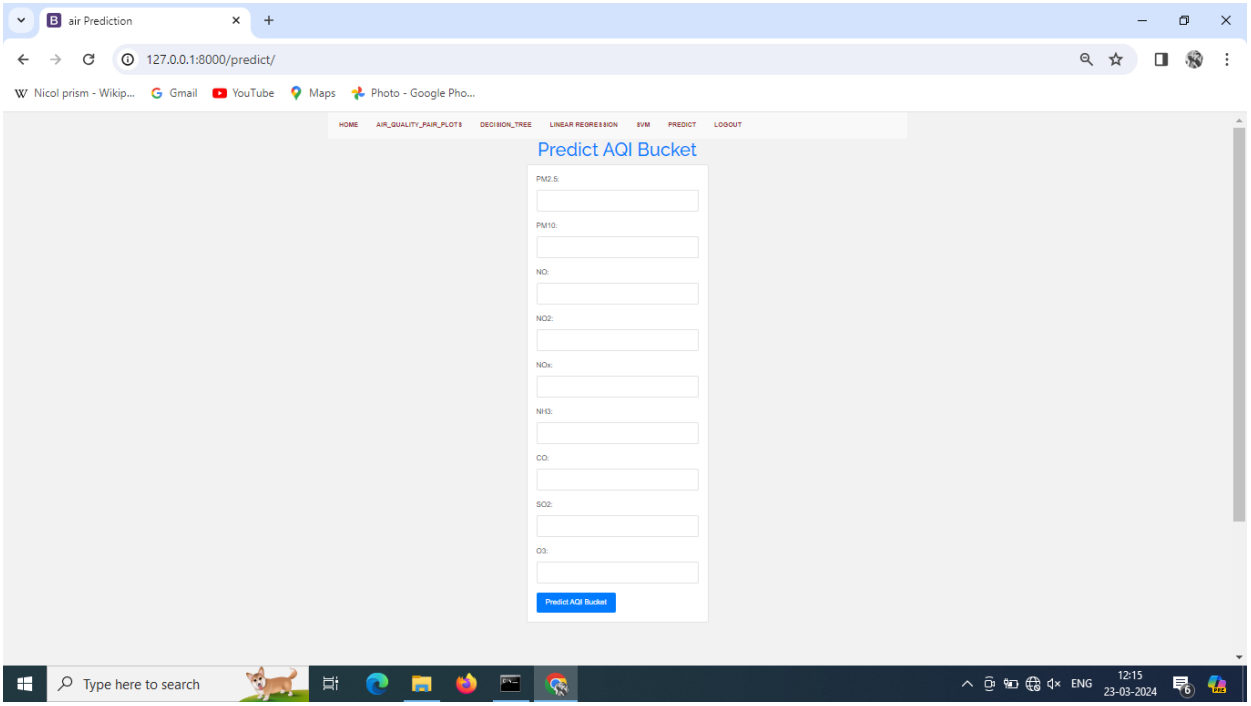


Fig:12.12 Prediction form using formula

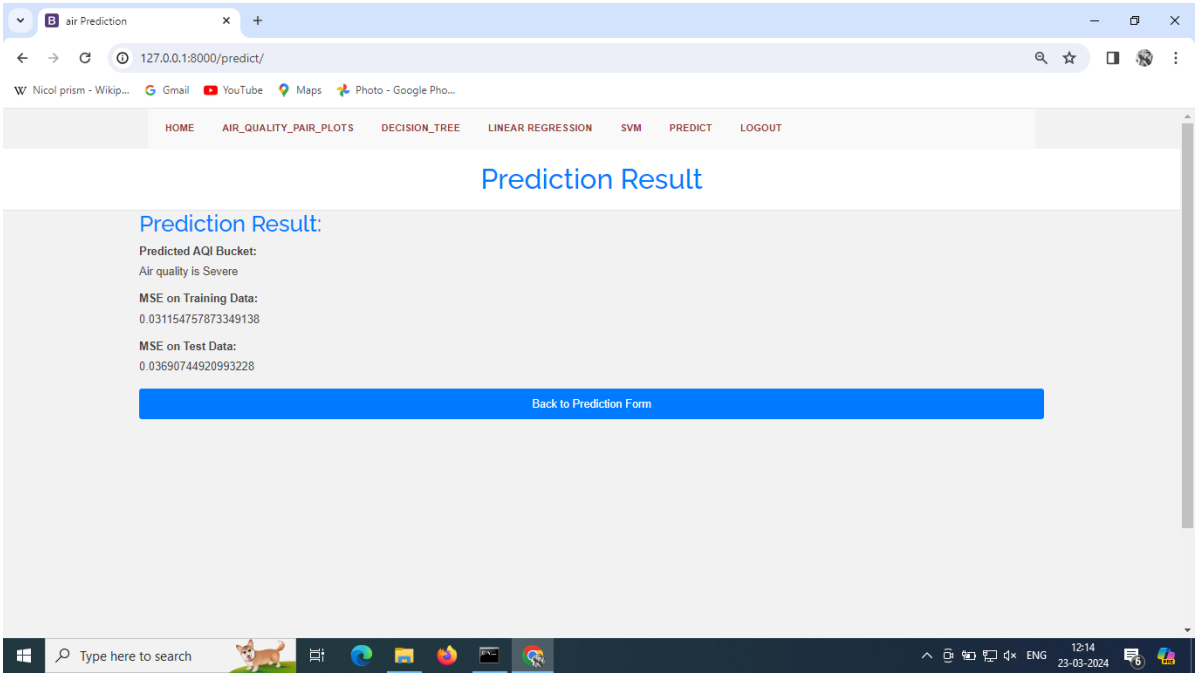


Fig:12.13 To view the prediction result

13. FUTURE ENHANCEMENT

The Project on air quality prediction using machine learning could benefit from several enhancements. These include exploring more advanced feature selection techniques, incorporating advanced machine learning models like gradient boosting and neural networks, considering ensemble learning and hybrid models for increased accuracy, accounting for temporal and spatial factors, implementing real-time monitoring and feedback loops, addressing missing data with augmentation and imputation methods, focusing on model interpretability, leveraging external data sources like meteorological data, developing a user-friendly interface, and rigorously validating and testing the enhanced model against existing benchmarks. These improvements would contribute to more accurate and adaptable air quality predictions.

14. CONCLUSION

This project aims to develop a robust module for predicting of Pollution and Risk prediction on Air Quality. The features used for prediction are considered. The prediction model has been built for prediction of Air Quality with a maximum accuracy. Few highly correlated features are used for analysing and prediction of risk factor and calculating Air Quality using Machine Learning Algorithm and Techniques

15.REFERENCES

[1]Verma, Ishan, Rahul Ahuja, Hardik Meisheri, and Lipika Dey. ”Air pollutant severity reduction using Bi-directional LSTM Network.” In 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 651-654. IEEE, 2018.

[2] Figures Zhang, Chao, Baoxian Liu, Jun chi Yan, Jing hai Yan, Lingjun Li, Dawei Zhang, Xiaoguang Rui, and Rongfang Bie. ”Hybrid Measurement of Air Quality as a 5 Fig. 8. RH w.r.t tin oxide Fig. 9. RH w.r.t C6H6 Mobile Service: An Image Based Approach.” In 2017 IEEE International Conference on Web Services (ICWS), pp. 853- 856. IEEE,2017.

[3] Yang, Ruijun, Feng Yan, and Nan Zhao. ”Urban air quality based on Bayesian network.” In 2017 IEEE 9th Fig. 10. RH w.r.t NO Fig. 11. RH w.r.t NO2 International Conference on Communication Software and Networks (ICCSN), pp. 1003-1006. IEEE,2017.

[4] Ayele, TemeseganWalelign, and RutvikMehta.”Air pollution monitoring and prediction using IoT.” In 2018 Second International Conference on Inventive Communication 6 Fig. 12. RH w.r.t Temperature Fig. 13. RH w.r.t CO and Computational Technologies (ICICCT), pp. 1741-1745. IEEE,2018.

[5] Djebbri, Nadjet, and MouniraRouainia. ”Artificial neural networks based air pollution monitoring in industrial sites.” In 2017 International Conference on Engineering and Technology (ICET), pp. 1-5. IEEE,2017. www.ijcrt.org © 2021 IJCRT | Volume 9, Issue 7 July 2021 | ISSN: 2320-2882 IJCRT2107446 International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org e213

[6] Kumar, Dinesh. ”Evolving Differential evolution method with random forest for prediction of Air Pollution.” Procedia computer science 132 (2018): 824-833.

[7] Jiang, Ningbo, and Matthew L. Riley. ”Exploring the utility of the random forest method for forecasting ozone pollution in SYDNEY.” Journal of Environment Protection and Sustainable Development 1.5 (2015): 245-254.

[8] Svetnik, Vladimir, et al. ”Random forest: a classification and regression tool for compound classification and QSAR modeling.” Journal of chemical information and computer sciences 43.6 (2003): 1947-1958.

[9] Biau, GA˜ Srard. ”Analysis of a random forest model.” ˇJournal of Machine Learning Research 13.Apr (2012): 1063- 1095.

[10] Biau, Gerard, and ErwanScornet. ”A random forest ´ guided tour.” Test 25.2 (2016): 197-227.

[11] Grimm, Rosina, et al. ”Soil organic carbon concentrations and stocks on Barro Colorado Island— Digital soil mapping using Random Forests analysis.” Geoderma 146.1- 2 (2008): 102- 113.

[12] Strobl, Carolin, et al. ”Conditional variable importance for random forests.” BMC bioinformatics 9.1 (2008): 307.

[13] Svetnik, Vladimir, et al. ”Random forest: a classification and regression tool for compound classification and QSAR modeling.” Journal of chemical information and computer sciences 43.6 (2003): 1947-1958.

[14] Verikas, Antanas, AdasGelzinis, and MarijaBacauskiene. ”Mining data with random forests: A survey and results of new tests.” Pattern recognition 44.2 (2011): 330-349.

[15] Ramasamy Jayamurugan,1 B. Kumaravel,1 S. Palanivelraja,1 and M.P.Chockalingam2 International Journal of Atmospheric Sciences Volume 2013, Article ID 264046, 7 pages http://dx.doi.org/10.1155/2013/264046