Major Project Synopsis

on

AIR QUALITY INDEX PREDICTOR

In partial fulfilment of requirements for the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

MANAY RAWAL [20100BTCSDSI07277]

RAHUL CHOUHAN [20100BTCSDSI07287]

DIVYANSH LASHKARI [20100BTCSDSI07269]

Under the guidance of

PROF. OM KANT SHARMA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SIIRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

SIIRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE

JUL.-DEC-2022

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

AIR QUALITY INDEX OF INDIA

How healthy is the air we breath?

Table of Contents

INTRODUCTION

Let’s see what you breath.

- Clean air is one of the basic requirement for good human health and well-being of the humanity. The problem of air pollution has become a new challenge for the world.
- Air pollution continues to be a well-known environmental problem worldwide
- The quality of the air is the result of complex interaction of many factors that involve the chemistry and the meteorology of the atmosphere, as well as the emissions of variety of pollutants both from natural and man-made sources
- This project uses supervised learning to predict Air Quality Index of India.
- The air quality is measured by the Air Quality Index. Let’s know what is the Air Quality Index (AQI) in India?

National Air Quality Index

91% of the World's Population Are Breathing in Polluted Air Every Day

- India uses the National Air Quality Index (AQI), Canada uses the Air Quality Health Index, Singapore uses the Pollutant Standards Index and Malaysia uses the Air Pollution Index.
- The National Air Quality Index (AQI) in India was launched on 17 September 2014 in New Delhi under the Swachh Bharat Abhiyan by the Environment Minister Shri PrakashJavadekar.
- The air quality index is composed of 8 pollutants ((PM10, PM2.5, NO2, SO2, CO, O3, NH3, and Pb).

Air Quality Index

The Quality of Air we breath

- AQI is the air quality index; it gives you the index value that what is the current pollution status in the city, how polluted the air currently is.
- The Air Quality Index (AQI) is a widely used concept to communicate with the public on air quality.
- The Air Quality Index is acquired by measuring emissions of eight major pollutants present in the air: Particulate matter (PM2.5 and PM10), Ozone (O3), Carbon Monoxide (CO), Nitrogen Dioxide (NO2), Sulphur Dioxide (SO2), Lead (Pb) and Ammonia (NH3) emissions.

6 Categories air of quality index

- The Air Quality Index measures the quality of air. It shows the amount and types of gases dissolved in the air. There are 6 categories of the air have been created in this air quality index.
- These categories are based on air quality. These categories are: good, satisfactory, moderate, poor, very poor and severe.

CENTRAL POLLUTION CONTROL'S BOARD

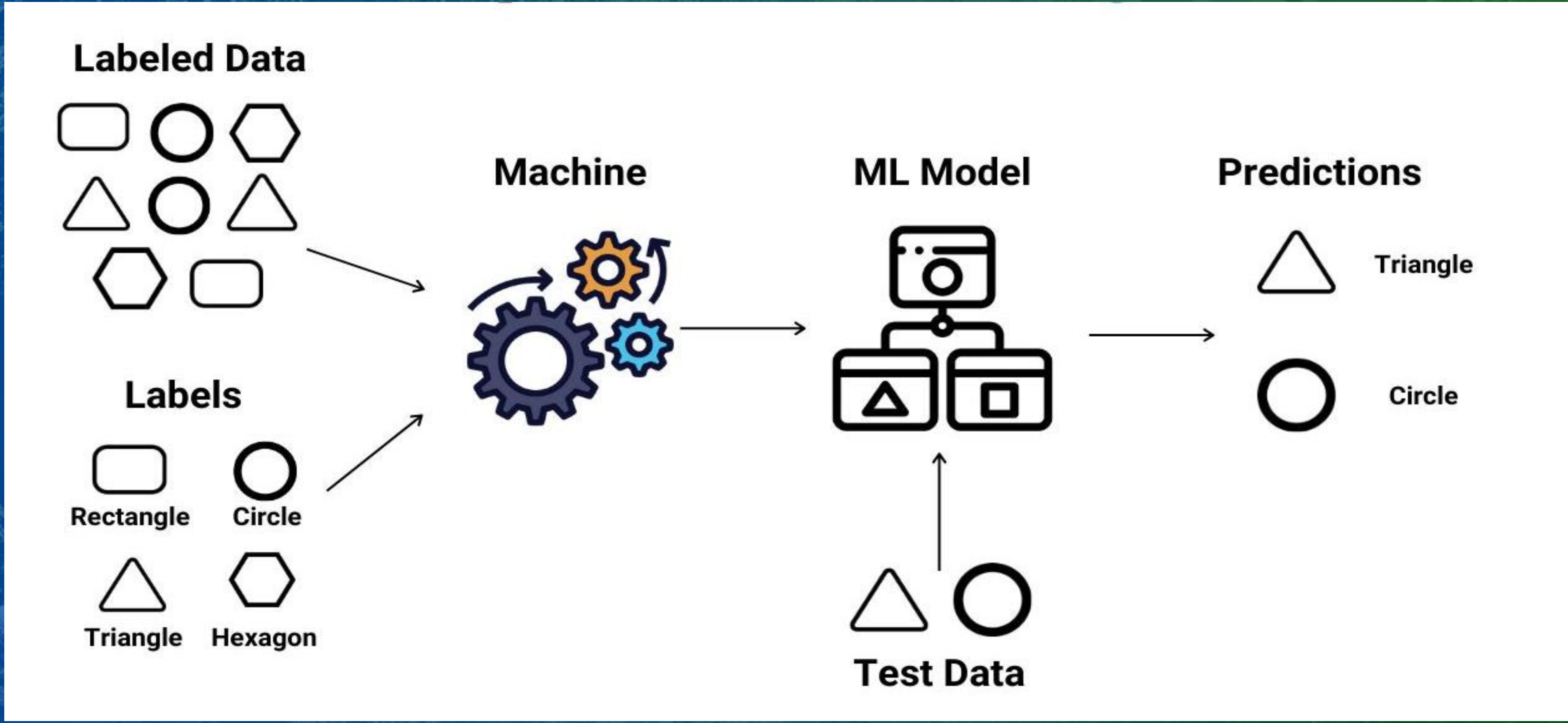
AIR QUALITY STANDARDS

AIR QUALITY INDEX	CATEGORY
0-50	Good
51-100	Satisfactory
101-200	Moderate
201-300	Poor
301-400	Very Poor
401-500	Severe

How our model works?

The Insides

Supervised Machine Learning



Preface

- “ Supervised learning means that a model learns from previous examples and is trained on labeled data. In other words, the dataset has tags that tell the model which patterns are related to fraud and which represent normal behavior ”.
- In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

Steps Involved in Supervised Learning:

1. Import the libraries
2. Read Dataset
3. Data Analysis
4. Label encoding of the categorical columns if have
5. Defining x and y
6. Splitting x and y into train and test data
7. Import the model
8. Train the model with x_train and y_train
9. Predict with x test and got predicted y
10. Evaluation with accuracy score and confusion matrix

Importing the libraries:

We can import libraries by following syntax:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Reading the dataset:

Reading the dataset by read_csv('File_name') function

```
df=pd.read_csv(io.BytesIO(uploaded['city_day.csv']))
```

```
city_day.csv
• city_day.csv(text/csv) - 2574056 bytes, last modified: 7/28/2020 - 100% done
Saving city_day.csv to city_day.csv
```

3. Data Analysis:

Extracting the first ten values from the Data Frame

```
print(df.head(10))
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10
5	Ahmedabad	2015-01-06	NaN	NaN	45.41	38.48	81.50	NaN	45.41
6	Ahmedabad	2015-01-07	NaN	NaN	112.16	40.62	130.77	NaN	112.16
7	Ahmedabad	2015-01-08	NaN	NaN	80.87	36.74	96.75	NaN	80.87
8	Ahmedabad	2015-01-09	NaN	NaN	29.16	31.00	48.00	NaN	29.16
9	Ahmedabad	2015-01-10	NaN	NaN	NaN	7.04	0.00	NaN	NaN

	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	18.59	36.08	4.43	10.14	1.00	NaN	NaN
4	39.33	39.31	7.01	18.89	2.78	NaN	NaN
5	45.76	46.51	5.42	10.83	1.93	NaN	NaN
6	32.28	33.47	0.00	0.00	0.00	NaN	NaN
7	38.54	31.89	0.00	0.00	0.00	NaN	NaN
8	58.68	25.75	0.00	0.00	0.00	NaN	NaN
9	8.29	4.55	0.00	0.00	0.00	NaN	NaN

Extracting the last ten values from the Data Frame

```
print(df.tail(10))
```

```
                City        Date  PM2.5    PM10    NO    NO2    NOx    NH3
29521  Visakhapatnam  2020-06-22  33.17  108.22  5.58  42.45  27.06  13.70
29522  Visakhapatnam  2020-06-23  25.40   83.38  2.76  34.09  19.92  13.13
29523  Visakhapatnam  2020-06-24  34.36   90.90  1.22  23.38  13.12  14.45
29524  Visakhapatnam  2020-06-25  13.45   58.54  2.30  21.60  13.09  12.27
29525  Visakhapatnam  2020-06-26   7.63   32.27  5.91  23.27  17.19  11.15
29526  Visakhapatnam  2020-06-27  15.02   50.94  7.68  25.06  19.54  12.47
29527  Visakhapatnam  2020-06-28  24.38   74.09  3.42  26.06  16.53  11.99
29528  Visakhapatnam  2020-06-29  22.91   65.73  3.45  29.53  18.33  10.71
29529  Visakhapatnam  2020-06-30  16.64   49.97  4.05  29.26  18.80  10.03
29530  Visakhapatnam  2020-07-01  15.00   66.00  0.40  26.85  14.05   5.20

         CO    SO2     O3  Benzene  Toluene  Xylene    AQI    AQI_Bucket
29521  0.73  13.65  34.85     3.99    10.24    2.32   95.0  Satisfactory
29522  0.54  10.40  43.27     2.88    12.03    1.33  100.0  Satisfactory
29523  0.56  10.92  35.12     2.99     3.15    1.60   86.0  Satisfactory
29524  0.41   8.19  29.38     1.28     5.64    0.92   77.0  Satisfactory
29525  0.46   6.87  19.90     1.45     5.37    1.45   47.0          Good
29526  0.47   8.55  23.30     2.24    12.07    0.73   41.0          Good
29527  0.52  12.72  30.14     0.74     2.21    0.38   70.0  Satisfactory
29528  0.48   8.42  30.96     0.01     0.01    0.00   68.0  Satisfactory
29529  0.52   9.84  28.30     0.00     0.00    0.00   54.0  Satisfactory
29530  0.59   2.10  17.05      NaN      NaN     NaN   50.0          Good
```

Describing the data frame statistically

```
print(df.describe())
```

```
              PM2.5          PM10            NO           NO2           NOx
count  24933.000000  18391.000000  25949.000000  25946.000000  25346.000000
mean      67.450578    118.127103     17.574730     28.560659     32.309123
std       64.661449     90.605110     22.785846     24.474746     31.646011
min        0.040000      0.010000      0.020000      0.010000      0.000000
25%       28.820000     56.255000      5.630000     11.750000     12.820000
50%       48.570000     95.680000      9.890000     21.690000     23.520000
75%       80.590000    149.745000     19.950000     37.620000     40.127500
max      949.990000   1000.000000    390.680000    362.210000    467.630000

                NH3            CO           SO2            O3       Benzene
count  19203.000000  27472.000000  25677.000000  25509.000000  23908.000000
mean      23.483476      2.248598     14.531977     34.491430      3.280840
std       25.684275      6.962884     18.133775     21.694928     15.811136
min        0.010000      0.000000      0.010000      0.010000      0.000000
25%        8.580000      0.510000      5.670000     18.860000      0.120000
50%       15.850000      0.890000      9.160000     30.840000      1.070000
75%       30.020000      1.450000     15.220000     45.570000      3.080000
max      352.890000    175.810000    193.860000    257.730000    455.030000

            Toluene        Xylene           AQI
count  21490.000000  11422.000000  24850.000000
mean       8.700972      3.070128    166.463581
std       19.969164      6.323247    140.696585
min        0.000000      0.000000     13.000000
25%        0.600000      0.140000     81.000000
50%        2.970000      0.980000    118.000000
75%        9.150000      3.350000    208.000000
max      454.850000    170.370000   2049.000000
```

Summing the null values column wise

```
print(df.isnull().sum())
```

```
City              0
Date              0
PM2.5          4598
PM10          11140
NO             3582
NO2            3585
NOx            4185
NH3           10328
CO             2059
SO2            3854
O3             4022
Benzene        5623
Toluene        8041
Xylene        18109
AQI            4681
AQI_Bucket     4681
dtype: int64
```

Calculating the relationship between each column in the data set.

```
print(df.corr())
```

```
            PM2.5      PM10        NO       NO2       NOx       NH3        CO  \
PM2.5    1.000000  0.846498  0.433491  0.350709  0.436792  0.275086  0.089912
PM10     0.846498  1.000000  0.502349  0.464380  0.527768  0.376816  0.112588
NO       0.433491  0.502349  1.000000  0.478070  0.794890  0.185621  0.212607
NO2      0.350709  0.464380  0.478070  1.000000  0.627627  0.234938  0.356521
NOx      0.436792  0.527768  0.794890  0.627627  1.000000  0.166224  0.226992
NH3      0.275086  0.376816  0.185621  0.234938  0.166224  1.000000  0.104891
CO       0.089912  0.112588  0.212607  0.356521  0.226992  0.104891  1.000000
SO2      0.132325  0.256974  0.170322  0.392233  0.238397 -0.038998  0.489697
O3       0.161238  0.244919  0.014580  0.293349  0.093170  0.094972  0.041736
Benzene  0.023911  0.022265  0.035771  0.025260  0.039121 -0.015650  0.061861
Toluene  0.117080  0.169335  0.150857  0.273926  0.189386  0.013227  0.277904
Xylene   0.114579  0.081700  0.094237  0.171701  0.087398 -0.019813  0.154889
AQI      0.659181  0.803313  0.452191  0.537071  0.486450  0.252019  0.683346

              SO2        O3   Benzene   Toluene    Xylene       AQI
PM2.5    0.132325  0.161238  0.023911  0.117080  0.114579  0.659181
PM10     0.256974  0.244919  0.022265  0.169335  0.081700  0.803313
NO       0.170322  0.014580  0.035771  0.150857  0.094237  0.452191
NO2      0.392233  0.293349  0.025260  0.273926  0.171701  0.537071
NOx      0.238397  0.093170  0.039121  0.189386  0.087398  0.486450
NH3     -0.038998  0.094972 -0.015650  0.013227 -0.019813  0.252019
CO       0.489697  0.041736  0.061861  0.277904  0.154889  0.683346
SO2      1.000000  0.162142  0.036110  0.296139  0.251195  0.490586
O3       0.162142  1.000000  0.020255  0.130209  0.111410  0.198991
Benzene  0.036110  0.020255  1.000000  0.739286  0.415427  0.044407
Toluene  0.296139  0.130209  0.739286  1.000000  0.421432  0.279992
Xylene   0.251195  0.111410  0.415427  0.421432  1.000000  0.165532
AQI      0.490586  0.198991  0.044407  0.279992  0.165532  1.000000
```

Labelling columns of the data frame

```
print(df.columns)
```

```
Index(['City', 'Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2',
       'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket'],
      dtype='object')
```

Replacing null values with “Good”.

```
df['AQI_Bucket'].fillna('Good')
```

```
0                Good
1                Good
2                Good
3                Good
4                Good
             ...     
29526            Good
29527    Satisfactory
29528    Satisfactory
29529    Satisfactory
29530            Good
Name: AQI_Bucket, Length: 29531, dtype: object
```

Printing the dataframe

```python
print(df)
```

```
                City        Date  PM2.5   PM10     NO    NO2    NOx    NH3  \
0          Ahmedabad  2015-01-01    NaN    NaN   0.92  18.22  17.15    NaN
1          Ahmedabad  2015-01-02    NaN    NaN   0.97  15.69  16.46    NaN
2          Ahmedabad  2015-01-03    NaN    NaN  17.40  19.30  29.70    NaN
3          Ahmedabad  2015-01-04    NaN    NaN   1.70  18.48  17.97    NaN
4          Ahmedabad  2015-01-05    NaN    NaN  22.10  21.42  37.76    NaN
...              ...         ...    ...    ...    ...    ...    ...    ...
29526  Visakhapatnam  2020-06-27  15.02  50.94   7.68  25.06  19.54  12.47
29527  Visakhapatnam  2020-06-28  24.38  74.09   3.42  26.06  16.53  11.99
29528  Visakhapatnam  2020-06-29  22.91  65.73   3.45  29.53  18.33  10.71
29529  Visakhapatnam  2020-06-30  16.64  49.97   4.05  29.26  18.80  10.03
29530  Visakhapatnam  2020-07-01  15.00  66.00   0.40  26.85  14.05   5.20

          CO    SO2      O3  Benzene  Toluene  Xylene   AQI    AQI_Bucket
0       0.92  27.64  133.36     0.00     0.02    0.00   NaN           NaN
1       0.97  24.55   34.06     3.68     5.50    3.77   NaN           NaN
2      17.40  29.07   30.70     6.80    16.40    2.25   NaN           NaN
3       1.70  18.59   36.08     4.43    10.14    1.00   NaN           NaN
4      22.10  39.33   39.31     7.01    18.89    2.78   NaN           NaN
...      ...    ...     ...      ...      ...     ...   ...           ...
29526   0.47   8.55   23.30     2.24    12.07    0.73  41.0          Good
29527   0.52  12.72   30.14     0.74     2.21    0.38  70.0  Satisfactory
29528   0.48   8.42   30.96     0.01     0.01    0.00  68.0  Satisfactory
29529   0.52   9.84   28.30     0.00     0.00    0.00  54.0  Satisfactory
29530   0.59   2.10   17.05      NaN      NaN     NaN  50.0          Good

[29531 rows x 16 columns]
```

4. . Label encoding of the categorical column if have:

- Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering. We do his by following syntax:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['AQI_Bucket']=le.fit_transform(df['AQI_Bucket'])
print(df['AQI_Bucket'].value_counts() )
```

```
1    8829
3    8224
6    4681
2    2781
5    2337
0    1341
4    1338
Name: AQI_Bucket, dtype: int64
```

5.Defining x and y

df.iloc helps us to select a specific row or column from the data set.

```
x=df.iloc[:,2:14]
y=df.iloc[:,0:16:15]
print(x)
print(y)
```

```
       PM2.5   PM10     NO    NO2    NOx    NH3     CO    SO2      O3
0        NaN    NaN   0.92  18.22  17.15    NaN   0.92  27.64  133.36
1        NaN    NaN   0.97  15.69  16.46    NaN   0.97  24.55   34.06
2        NaN    NaN  17.40  19.30  29.70    NaN  17.40  29.07   30.70
3        NaN    NaN   1.70  18.48  17.97    NaN   1.70  18.59   36.08
4        NaN    NaN  22.10  21.42  37.76    NaN  22.10  39.33   39.31
...      ...    ...    ...    ...    ...    ...    ...    ...     ...
29526  15.02  50.94   7.68  25.06  19.54  12.47   0.47   8.55   23.30
29527  24.38  74.09   3.42  26.06  16.53  11.99   0.52  12.72   30.14
29528  22.91  65.73   3.45  29.53  18.33  10.71   0.48   8.42   30.96
29529  16.64  49.97   4.05  29.26  18.80  10.03   0.52   9.84   28.30
29530  15.00  66.00   0.40  26.85  14.05   5.20   0.59   2.10   17.05

       Benzene  Toluene  Xylene
0         0.00     0.02    0.00
1         3.68     5.50    3.77
2         6.80    16.40    2.25
3         4.43    10.14    1.00
4         7.01    18.89    2.78
...        ...      ...     ...
29526     2.24    12.07    0.73
29527     0.74     2.21    0.38
29528     0.01     0.01    0.00
29529     0.00     0.00    0.00
29530      NaN      NaN     NaN

[29531 rows x 12 columns]
                City  AQI_Bucket
0          Ahmedabad           6
1          Ahmedabad           6
2          Ahmedabad           6
3          Ahmedabad           6
4          Ahmedabad           6
...              ...         ...
29526  Visakhapatnam           0
29527  Visakhapatnam           3
29528  Visakhapatnam           3
29529  Visakhapatnam           3
29530  Visakhapatnam           0

[29531 rows x 2 columns]
```

6. Splitting the x and y in train and test:

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. Splitting your dataset is essential for an unbiased evaluation of prediction performance.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size = .25)
print(x_train.shape, x_test.shape, y_train.shape,y_test.shape)
```

```
(22148, 12) (7383, 12) (22148, 2) (7383, 2)
```

7. Import the model :

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
print(lr)
```

```
LogisticRegression()
```

8. Train the model with x_train and y_train:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()

lr.fit(x_train,y_train)
```

9. Predict with x_test and get predicted y

```
y_pred_lr = lr.predict(x_test)
print(y_pred_lr[:5],y_test.values[:5] )

print(lr.score(x_train,y_train))
print(lr.score(x_test, y_test))
```

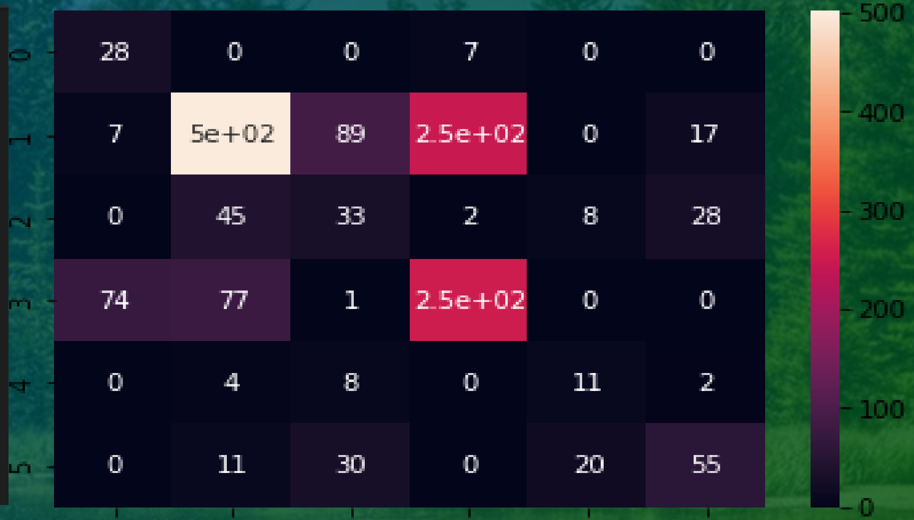
10. Evaluation

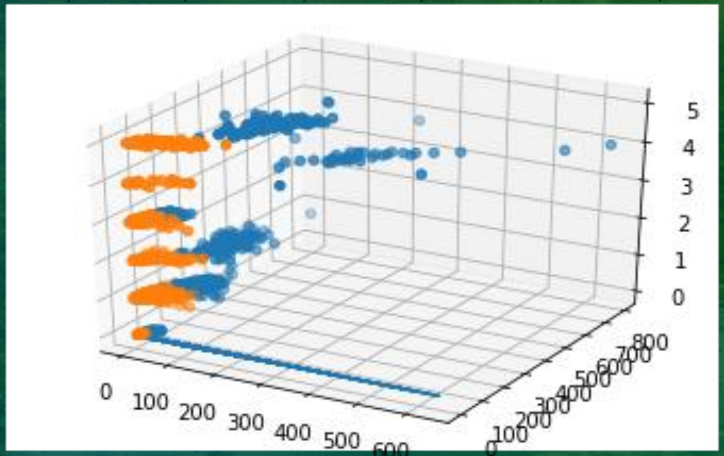
```
print(lr.score(x_train,y_train))
print(lr.score(x_test, y_test))

from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
print(confusion_matrix(y_pred_lr, y_test))
print(classification_report(y_pred_lr, y_test))
print(f'model_score- {lr.score(x_test,y_test)}')
print(f'accuracy_score- {accuracy_score(y_pred_lr, y_test)}')
```

```
0.5736583279880265
0.5567671584348942
[[ 32   1   0  11   0   0]
 [  3 464  71 248   3  16]
 [  0  61  45   2   4  35]
 [ 84  89   0 267   0   1]
 [  0   3   6   0  11   6]
 [  0  12  24   2   9  49]]
              precision    recall  f1-score   support

           0       0.27      0.73      0.39        44
           1       0.74      0.58      0.65       805
           2       0.31      0.31      0.31       147
           3       0.50      0.61      0.55       441
           4       0.41      0.42      0.42        26
           5       0.46      0.51      0.48        96

    accuracy                           0.56      1559
   macro avg       0.45      0.52      0.47      1559
weighted avg       0.59      0.56      0.57      1559

model_score- 0.5567671584348942
accuracy_score- 0.5567671584348942
```

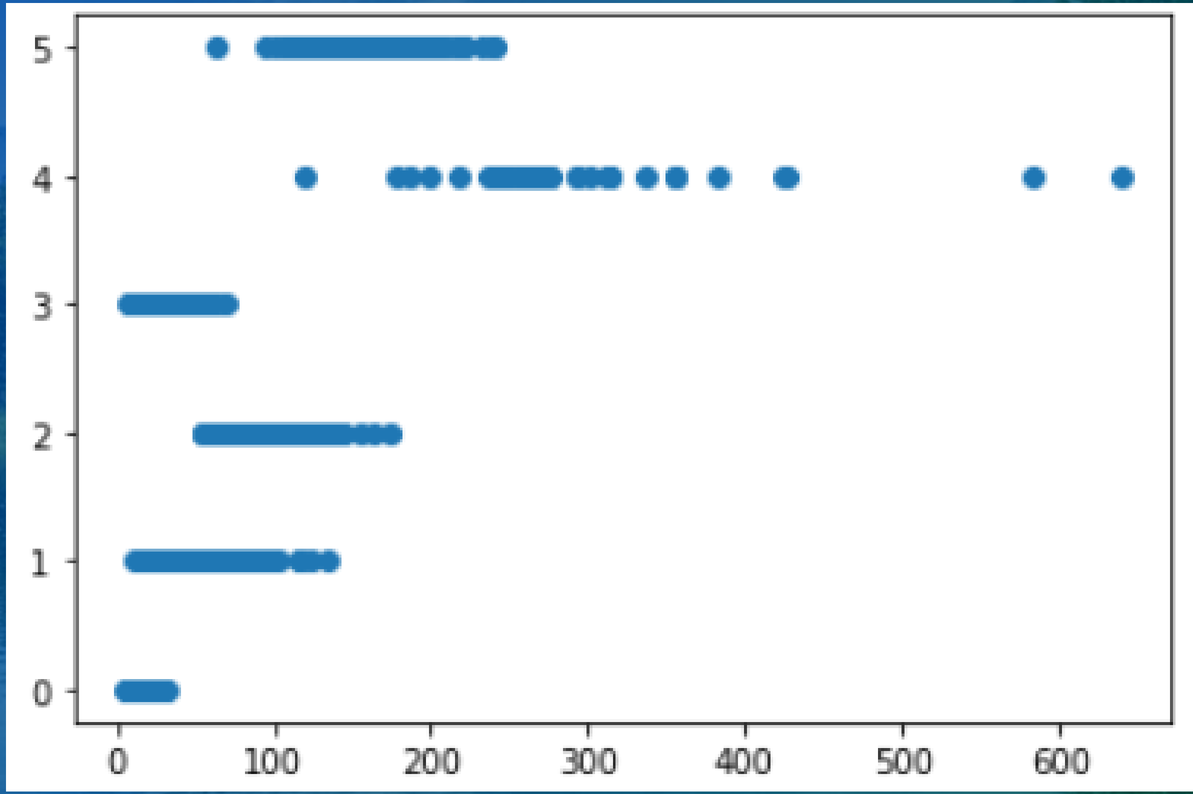
Confusion matrix and Accuracy

```
cm=confusion_matrix(y_pred_lr,y_test)
sns.heatmap(cm,annot=True)
plt.show()

ax = plt.axes(projection = '3d')
ax.scatter3D(x_test['PM2.5'],x_test['PM10'],y_test)
ax.scatter3D(x_test['NO'],x_test['NO2'],y_pred_lr, 'black')
plt.plot(x_test['PM2.5'], y_pred_lr)
plt.show()
```





```python
plt.scatter(x_test['PM2.5'], y_test)
plt.show()
```



• Evaluation

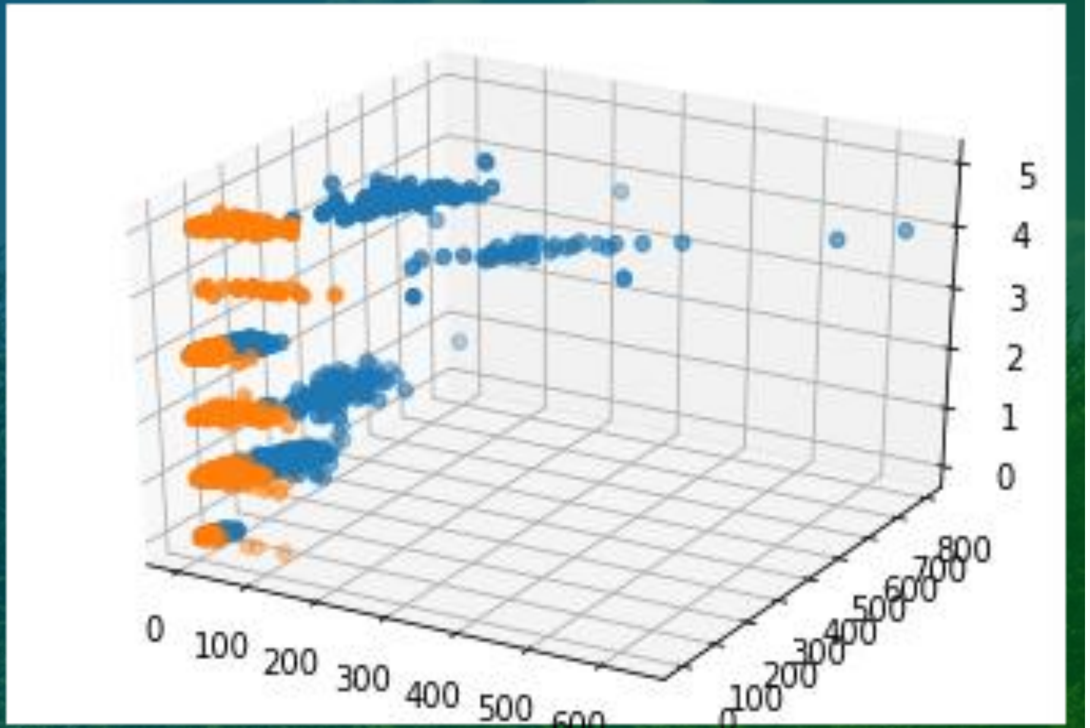
```
print(lr.score(x_train,y_train))
print(lr.score(x_test, y_test))
```

```
0.5736583279880265
0.5567671584348942
```

❖ Decision Tree
Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred_dtc=dtc.predict(x_test)
print(f'Predicted_y{y_pred_dtc[:5]} Actual_y{y_test.values[:5]}')
print(confusion_matrix(y_pred_dtc,y_test))

from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
cm=confusion_matrix(y_pred_dtc, y_test)
plt.figure(figsize=(7,5))
print(sns.heatmap(cm, annot=True))
plt.show()
print(classification_report(y_pred_dtc, y_test))
print(f'model_score- {dtc.score(x_test, y_test)} ')
print(f'accuracy_score- {accuracy_score(y_pred_dtc, y_test)}')
ax = plt.axes (projection ='3d')
ax.scatter3D(x_test['PM2.5'],x_test['PM10'],y_test)
ax.scatter3D(x_test['NO'],x_test['NO2'], y_pred_dtc,'black')
plt.show()
```

```
Predicted_y[0 4 0 3 1] Actual_y[[3]
 [4]
 [0]
 [3]
 [1]]
[[ 77   3   0  31   0   0]
 [  0 513  38  97   0   6]
 [  0  30  97   1   0  18]
 [ 32  90   0 381   0   0]
 [  0   0   1   0  23   0]
 [  0   2  25   0  16  78]]
AxesSubplot(0.125,0.125;0.62x0.755)
```

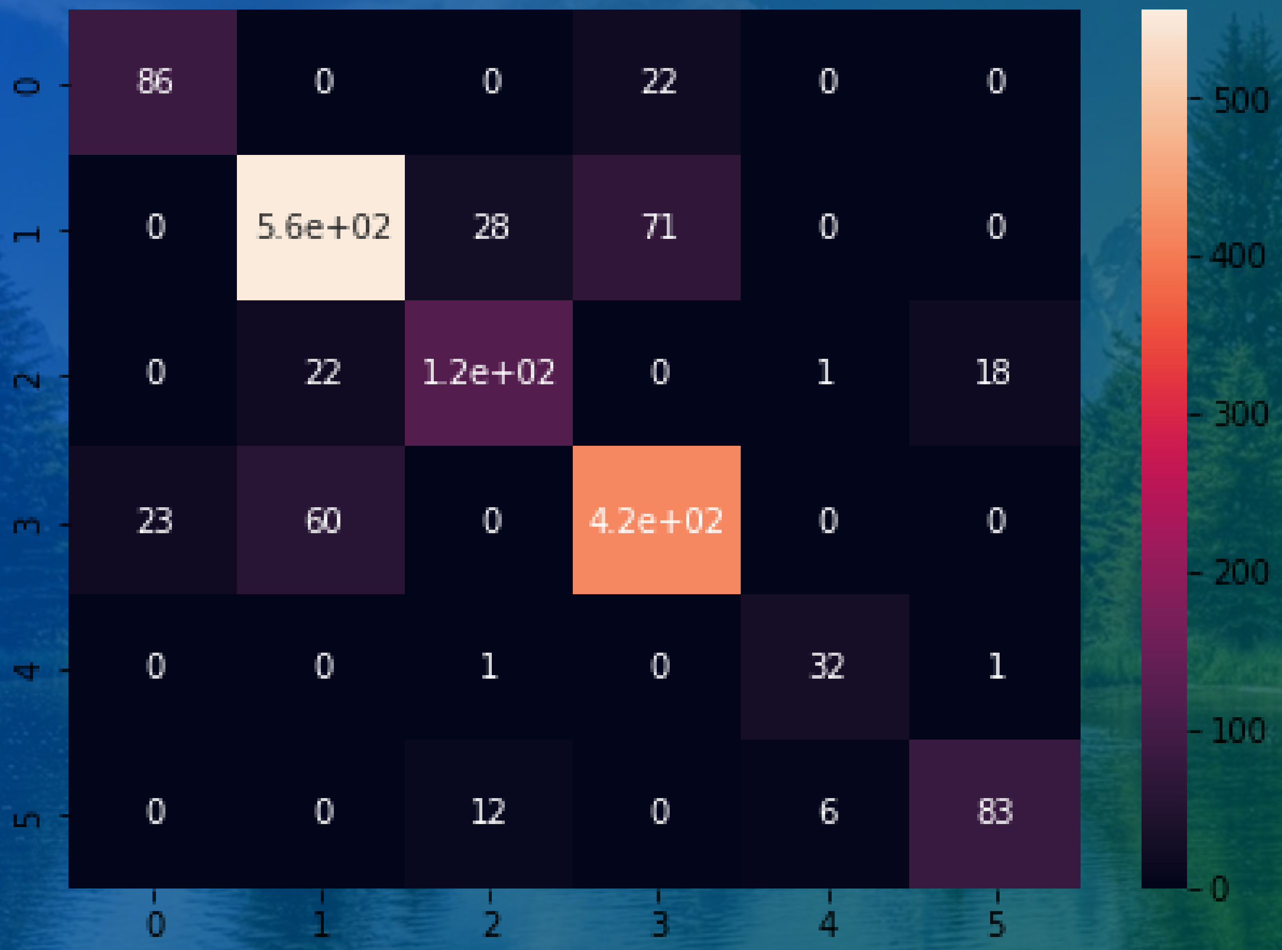


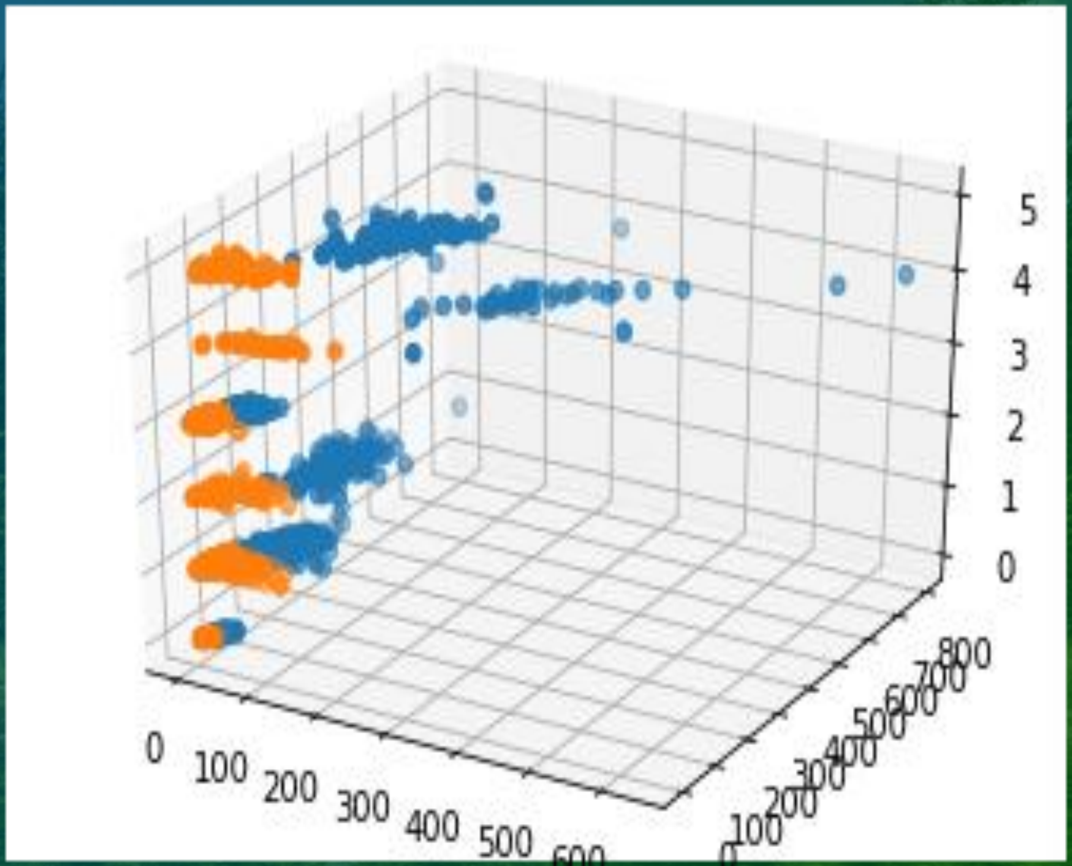
❖ Random forest
Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred_rfc = rfc.predict(x_test)
print(f'predicted_y-{y_pred_rfc} actual_y-{y_test.values}')
print(confusion_matrix(y_pred_rfc,y_test))
cm=confusion_matrix(y_pred_rfc,y_test)
plt.figure(figsize=(7,5))
sns.heatmap(cm, annot=True)
plt.show()
print(classification_report(y_pred_rfc, y_test))
print(f'model_score- {rfc.score(x_test,y_test)}')
print(f'accuracy_score- {accuracy_score(y_pred_rfc, y_test)}')
ax = plt.axes (projection ='3d')
ax.scatter3D(x_test['PM2.5'],x_test['PM10'],y_test)
ax.scatter3D(x_test['NO'],x_test['NO2'],y_pred_rfc,'black')
plt.show()
```

```python
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred_rfc = rfc.predict(x_test)
print(f'predicted_y-{y_pred_rfc} actual_y-{y_test.values}')
print(confusion_matrix(y_pred_rfc,y_test))
cm=confusion_matrix(y_pred_rfc,y_test)
plt.figure(figsize=(7,5))
sns.heatmap(cm, annot=True)
plt.show()
print(classification_report(y_pred_rfc, y_test))
print(f'model_score- {rfc.score(x_test,y_test)}')
print(f'accuracy_score- {accuracy_score(y_pred_rfc, y_test)}')
ax = plt.axes (projection ='3d')
ax.scatter3D(x_test['PM2.5'],x_test['PM10'],y_test)
ax.scatter3D(x_test['NO'],x_test['NO2'],y_pred_rfc,'black')
plt.show()
```

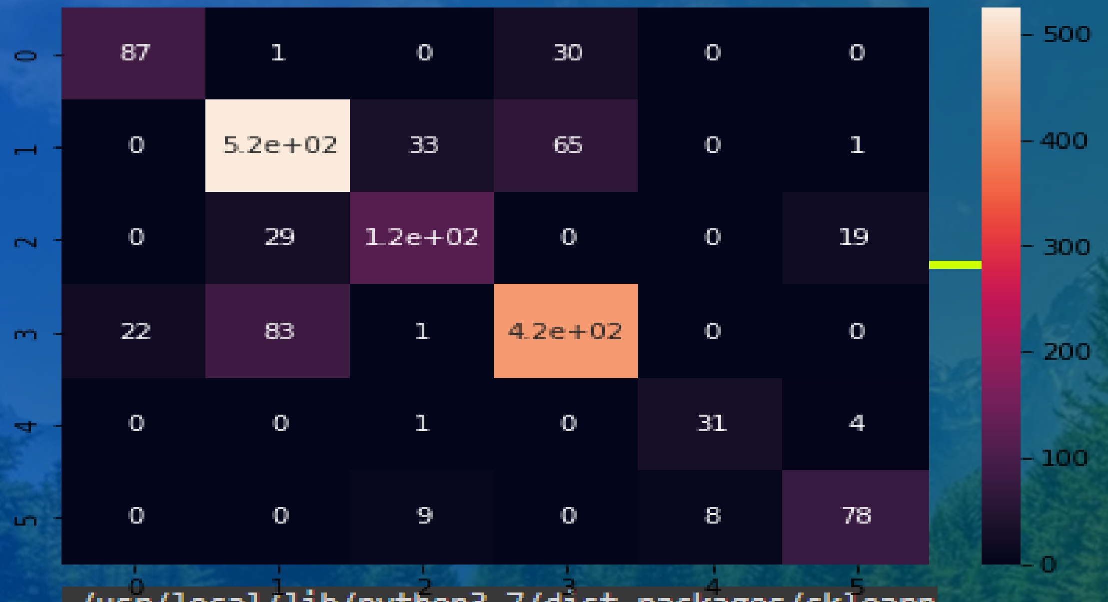
```
/usr/local/lib/python3.7/dist-packag
  This is separate from the ipykerne
predicted_y-[3 4 0 ... 3 0 3] actual
 [4]
 [0]
 ...
 [3]
 [0]
 [3]]
[[ 86   0   0  22   0   0]
 [  0 556  28  71   0   0]
 [  0  22 120   0   1  18]
 [ 23  60   0 417   0   0]
 [  0   0   1   0  32   1]
 [  0   0  12   0   6  83]]
```


86 0 0 22 0 0
0 5.6e+02 28 71 0 0
0 22 1.2e+02 0 1 18
23 60 0 4.2e+02 0 0
0 0 1 0 32 1
0 0 12 0 6 83
0 1 2 3 4 5
0 1 2 3 4 5
500 400 300 200 100 0

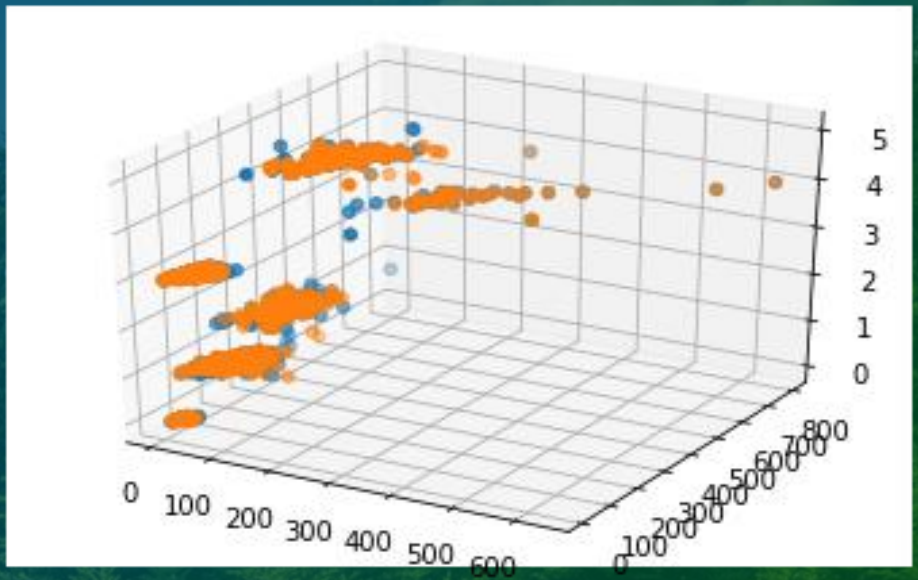

0 100 200 300 400 500 600
0 100 200 300 400 500 600 700 800
0 1 2 3 4 5

❖ K-Nearest Neighbors Classifier

```python
from sklearn.neighbors import KNeighborsClassifier
knc=KNeighborsClassifier()
knc.fit(x_train, y_train)
y_pred_knc=knc.predict(x_test)
print(f'Predicted_y{y_pred_knc[:5]} Actual_y{y_test.values[:5]}')
print(confusion_matrix(y_pred_knc,y_test))
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
cm=confusion_matrix(y_pred_knc,y_test)
plt.figure(figsize=(7,5))
print(sns.heatmap(cm, annot=True))
plt.show()
print(classification_report(y_pred_knc, y_test))
print(f'model_score- {dtc.score(x_test, y_test)} ')
print(f'accuracy_score- {accuracy_score(y_pred_knc, y_test)}')
ax = plt.axes(projection  = '3d')
ax.scatter3D(x_test['PM2.5'],x_test['PM10'],y_test)
ax.scatter3D(x_test['PM2.5'],x_test['PM10'],y_pred_knc, 'black')
plt.show()
```




```
/usr/local/lib/python3.7/dist-packages/sklearn
  return self._fit(X, y)
Predicted_y[0 4 0 3 1] Actual_y[[3]
 [4]
 [0]
 [3]
 [1]]
[[ 87   1   0  30   0   0]
 [  0 525  33  65   0   1]
 [  0  29 117   0   0  19]
 [ 22  83   1 415   0   0]
 [  0   0   1   0  31   4]
 [  0   0   9   0   8  78]]
AxesSubplot(0.125,0.125;0.62x0.755)
```

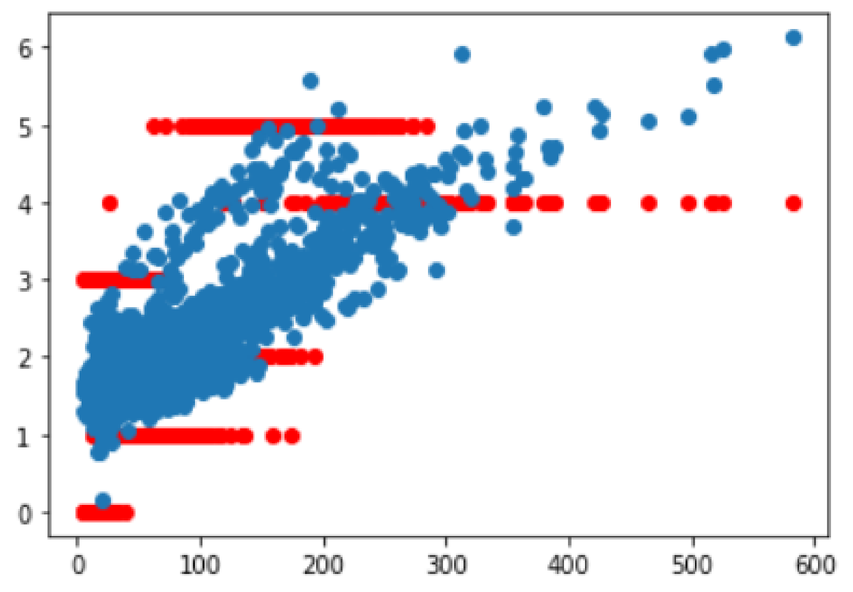



```
              precision    recall  f1-score   support

           0       0.80      0.74      0.77       118
           1       0.82      0.84      0.83       624
           2       0.73      0.71      0.72       165
           3       0.81      0.80      0.81       521
           4       0.79      0.86      0.83        36
           5       0.76      0.82      0.79        95

    accuracy                           0.80      1559
   macro avg       0.79      0.79      0.79      1559
weighted avg       0.80      0.80      0.80      1559

model_score- 0.7498396407953817
accuracy_score- 0.8037203335471456
```

❖ Simple linear Regressor

```
from sklearn.linear_model import LinearRegression
slr=LinearRegression()

slr.fit(x_train, y_train)
y_pred=slr.predict(x_test)

print(slr.score(x_train,y_train))
print(slr.score(x_test,y_test))

from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score

print("Mean Absolute Error:- ",mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:- ",mean_squared_error(y_test,y_pred))
print("r2_score:- ",r2_score(y_test,y_pred))

a=slr.coef_
b=slr.intercept_


plt.scatter(x_test['PM2.5'],y_test,color='r')
plt.scatter(x_test['PM2.5'],y_pred)

plt.show()
```

```
0.1961930096682737
0.18946593484141327
Mean Absolute Error:-  1.0708636884898894
Mean Squared Error:-  1.384272714993072
r2_score:-  0.18946593484141327
```



References

Dataset: https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india?resource=download

Google Drive: https://drive.google.com/drive/folders/1j5hfguD4NrOGAMA_rheI0RpP6-1Ux2gL

Google Colab 2: https://colab.research.google.com/drive/1Hi1ur89p4DdpjSAWjd0gJV-LGjHSG7e-#scrollTo=-teNaxQN-B01

Click on the link for the source codes

Google Colab 1: https://colab.research.google.com/drive/1GO0JBNjDj_WMdzNqXu0_kbI8TKzHRsyE

THANK YOU!

MANAY RAWAL [20100BTCSDSI07277]

RAHUL CHOUHAN [20100BTCSDSI07287]

DIVYANSH LASHKARI [20100BTCSDSI07269]