AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF CSE

Programming in Python

Spring 2022-2023

Section: A

Final Term Project On

Classification Based Machine Learning Model Development

Based On

Global Air Pollution Dataset

<u>Supervised By:</u>

Dr. Akinul Islam Jony

Associate Professor & Head (UG), Computer Science

<u>Submitted By:</u>

Name	ID
1. Muhammad Shahriar Zaman	20-41840-1
2. Md. Raiyan Khan	19-41453-3
3. Nafiz Fahad	20-43416-1

Date of Submission: **May 2nd, 2023**

TABLE OF CONTENTS

Project Overview:

Air pollution is a major concern in the urban sphere of the 21st century. Man-made causes are degrading the quality of air in the environment and in return it is causing various ailments such as respiratory infections, lung diseases and heart conditions. Predicting air quality is an important step for identifying this problem.

For this project we will work on a dataset about global air pollution records. We will preprocess it, explore the data and then train machine learning models based on the datasets and finally analyze the outcomes.

Dataset Overview: Our dataset has 12 columns and 2291 records.

Dataset source->
https://www.kaggle.com/datasets/hasibalmuzdadid/global-air-pollution-dataset?fbclid=IwAR0gVSQlEEJG8zWPPSmnJyoKkjtw90Q0uxBz2oWOwDp2_hkw4QWh5Rl95PQ

1	Country	City	AQI	AQI_Category	CO_Value	CO_Category	Ozone_Value	Ozone_Category	NO2_Value	NO2_Category	PM_2.5_Value	PM_2.5_Category
2	Afghanistan	Kuhestan	151	Unhealthy	1	Good	41	Good	0	Good	151	Unhealthy
3	Afghanistan	Qunduz	117	Unhealthy for Sensitive Groups	1	Good	44	Good	0	Good	117	Unhealthy for Sensitive Groups
4	Afghanistan	Rostaq	113	Unhealthy for Sensitive Groups	1	Good	42	Good	0	Good	113	Unhealthy for Sensitive Groups
5	Afghanistan	Tokzar	77	Moderate	1	Good	40	Good	0	Good	77	Moderate
6	Albania	Gramsh	68	Moderate	1	Good	39	Good	1	Good	68	Moderate
7	Albania	Elbasan	71	Moderate	1	Good	44	Good	1	Good	71	Moderate
8	Algeria	Adrar	106	Unhealthy for Sensitive Groups	0	Good	45	Good	0	Good	106	Unhealthy for Sensitive Groups
9	Angola	Lubango	40	Good	1	Good	14	Good	0	Good	40	Good
10	Angola	Luena	66	Moderate	1	Good	22	Good	0	Good	66	Moderate
11	Angola	aluquemb	42	Good	2	Good	12	Good	0	Good	42	Good
12	Argentina	Chacabucc	17	Good	0	Good	17	Good	1	Good	17	Good
13	Argentina	Corrientes	13	Good	1	Good	12	Good	0	Good	13	Good
14	Argentina	Mercedes	14	Good	1	Good	14	Good	0	Good	3	Good
15	Argentina	Formosa	32	Good	0	Good	13	Good	0	Good	32	Good
16	Argentina	Necochea	23	Good	0	Good	23	Good	0	Good	17	Good
17	Argentina	Villeta	90	Moderate	2	Good	5	Good	3	Good	90	Moderate
18	Armenia	Ashtarak	39	Good	1	Good	29	Good	1	Good	39	Good
19	Armenia	Hrazdan	31	Good	0	Good	31	Good	0	Good	30	Good
20	Armenia	Dilijan	32	Good	1	Good	27	Good	0	Good	32	Good
21	Australia	Geraldton	29	Good	0	Good	29	Good	0	Good	13	Good
22	Australia	Ballarat	22	Good	0	Good	13	Good	6	Good	22	Good
23	Australia	Ballina	34	Good	0	Good	34	Good	1	Good	15	Good
24	Australia	Bundaberg	32	Good	0	Good	32	Good	0	Good	10	Good
25	Austria	osterneubu	59	Moderate	1	Good	30	Good	3	Good	59	Moderate
26	Austria	Leoben	63	Moderate	1	Good	39	Good	0	Good	63	Moderate
27	Austria	Dornbirn	29	Good	1	Good	25	Good	3	Good	29	Good
28	Austria	Feldkirch	34	Good	1	Good	25	Good	2	Good	34	Good
29	Austria	Enns	62	Moderate	1	Good	39	Good	1	Good	62	Moderate
30	Azerbaijan	Artyom	48	Good	1	Good	35	Good	0	Good	48	Good
31	Azerbaijan	Culfa	52	Moderate	1	Good	35	Good	0	Good	52	Moderate

Fig.: Dataset to be used in this project (Global Air Pollution)

Dataset Overview (Cont'd):

We have described each of the columns of our dataset below:

- **Country:** Name of the country of which the air will be studied
- **City:** Name of the city.
- **AQI Value:** Overall AQI**(Air Quality Index)** value of the city, the lesser the better
- **AQI Category:** Overall AQI category of the city
- **CO Value:** AQI value of Carbon Monoxide of the city
- **CO Category:** AQI category of Carbon Monoxide**(Pollutant)** of the city
- **Ozone Value:** AQI value of Ozone**(Pollutant)** of the city
- **Ozone Category:** AQI category of Ozone of the city
- **NO2 Value:** AQI value of Nitrogen Dioxide**(NO2, works as a pollutant)** of the city
- **NO2 Category:** AQI category of Nitrogen Dioxide of the city
- **PM2.5 Value:** AQI value of Particulate Matter with a diameter of 2.5**(Type of pollutant)** micrometers or less of the city.
- **PM2.5 Category:** AQI category of Particulate Matter with a diameter of 2.5 micrometers or less of the city.

Importing our dataset:

```
from sklearn import metrics
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import pandas as pd
#Using pandas dataframe
df = pd.read_csv('Project/airdata.csv')

print("\n\nShape of DataFrame:", df.shape)
```

```
Shape of DataFrame: (2291, 12)
PS C:\Users\Asus\Desktop\Final_Project>
```

Fig.: Shape of DataFrame df printed

Dataset Preprocessing: For training our machine learning model we have to work on a clean dataset, that is why we are preprocessing our dataframe df.

1. Data Cleaning:

	Country	City	AQI.Value	AQI.Category	CO.AQI.Value	CO.AQI.Category	Ozone.AQ
2259	Viet Nam	Ben Tre	77	Moderate	1	Good	
2260	Zambia	Kafue	36	Good	0	Good	
2261	Zimbabwe	Hwange	44	Good	0	Good	
2262		Granville	30	Good	1	Good	
2263		Kingston Upon Hull	33	Good	1	Good	
2264		New Waterford	20	Good	1	Good	
2265		Kingstown	163	Unhealthy	0	Good	
2266		Nanakuli	30	Good	0	Good	
2267		Lavagna	55	Moderate	1	Good	
2268		Ladispoli	48	Good	1	Good	
2269		Dong Hoi	55	Moderate	0	Good	
2270		Nettuno	53	Moderate	1	Good	
2271		Puebloviejo	71	Moderate	1	Good	
2272		Fiumicino	51	Moderate	2	Good	

Showing 2,258 to 2,272 of 2,291 entries, 12 total columns

Fig.: Dataset with missing values in country column

Here we can see that we have some missing values in our dataset, we will remove them by using the following code.

Deleting Rows with Missing Values:

```
# The notna() method detects the empty cells
# for non-empty cells it return True and vice-versa
df = df[df['Country'].notna()]
print(df)
```

Output:

2249	2248	Venezuela (Bolivarian Republic of)	Puerto Ayacucho	28	Good	1	Good
2250	2249	Venezuela (Bolivarian Republic of)	Araure	84	Moderate	3	Good
2251	2250	Viet Nam	Tay Ninh	45	Good	0	Good
2252	2251	Viet Nam	Thai Nguyen	150	Unhealthy	3	Good
2253	2252	Viet Nam	Tuyen Quang	167	Unhealthy	4	Good
2254	2253	Viet Nam	Chau Doc	57	Moderate	1	Good
2255	2254	Viet Nam	Cao Bang	47	Good	1	Good
2256	2255	Viet Nam	Bac Giang	179	Unhealthy	10	Good
2257	2256	Viet Nam	Quang Ngai	62	Moderate	1	Good
2258	2257	Viet Nam	Tuy Hoa	51	Moderate	1	Good
2259	2258	Viet Nam	Ben Tre	77	Moderate	1	Good
2260	2259	Zambia	Kafue	36	Good	0	Good
2261	2260	Zimbabwe	Hwange	44	Good	0	Good

Fig.: Dataframe after missing values have been removed

Now, let us visualize our dataset based on overall air quality parameters.

Code:

```
# Countplot will show barcharts based on categories
# of selected feature
import seaborn as sns
b=sns.countplot(x='AQI_Category', data=df)
b.set_xlabel("Air Quality Index Status",fontsize=15)
b.set_ylabel("No. of Records",fontsize=15)
plt.title("All gradings of AQI count",fontsize=20)
plt.show()
```

Output:

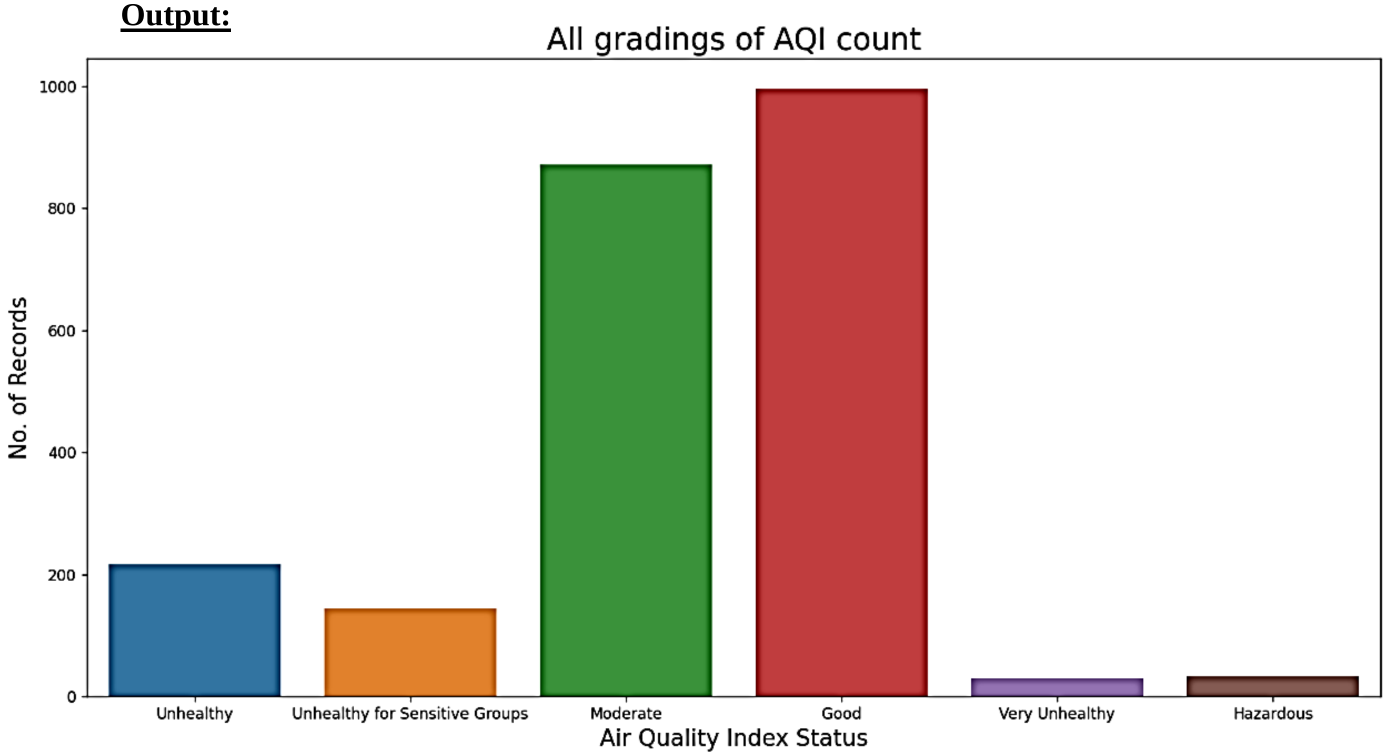


Fig.: Air quality of various countries of the world

Now, let us visualize our dataset based on air quality parameter(Ozone count in air).

Code:

```
# Countplot will show barcharts based on categories
# of selected feature
import seaborn as sns
b=sns.countplot(x='Ozone_Category', data=df)
b.set_xlabel("Ozone Count Status",fontsize=15)
b.set_ylabel("No. of Records",fontsize=15)
plt.title("All gradings of Ozone count",fontsize=20)
plt.show()
```

Output:

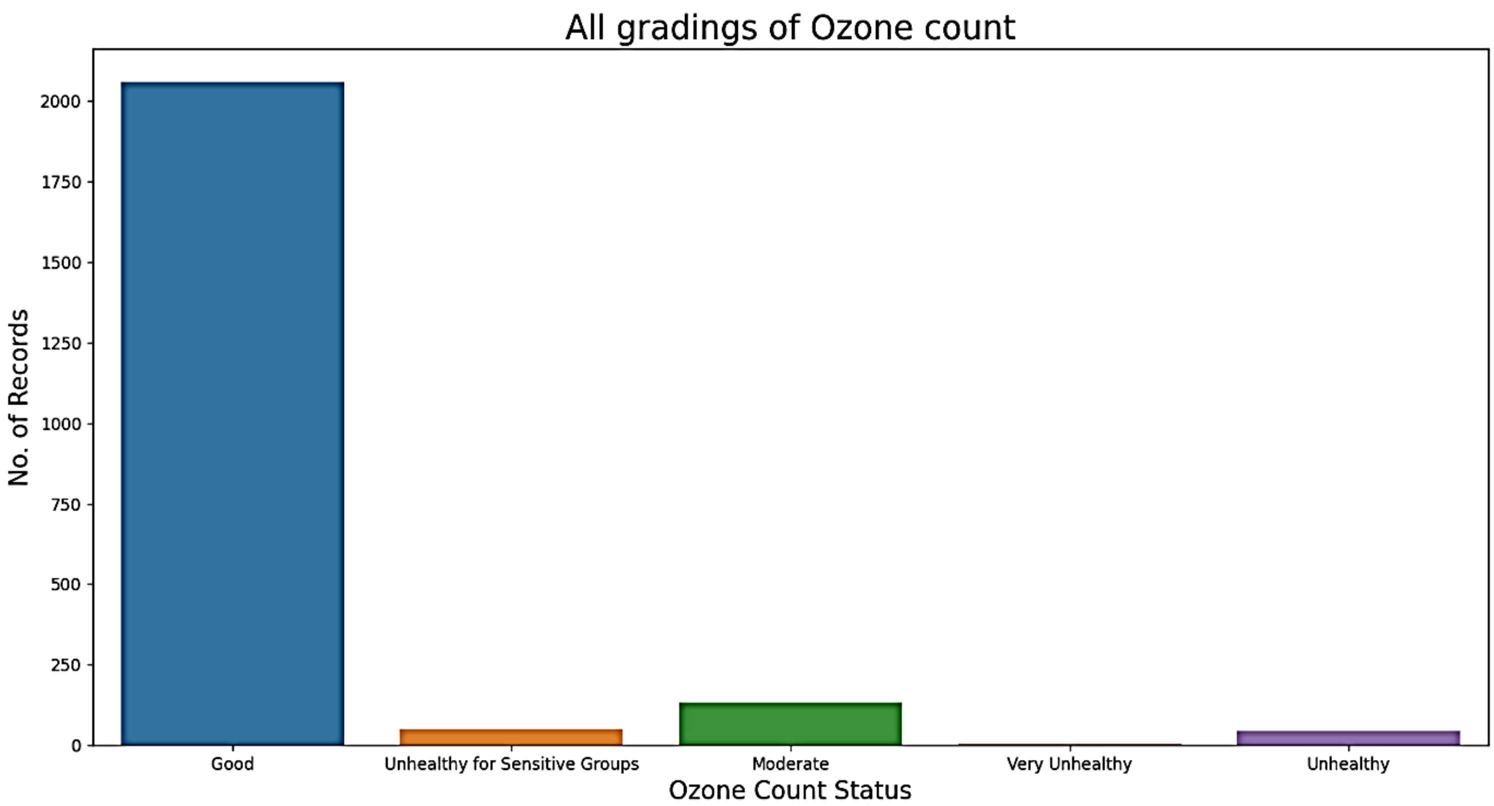


Fig.: Air quality (Ozone count) of various countries of the world

Now, let us visualize our dataset based on air quality parameter(Carbon Monoxide in air).

Code:

```
# Countplot will show barcharts based on categories
# of selected feature
import seaborn as sns
b=sns.countplot(x='CO_Category', data=df)
b.set_xlabel("CO Status of Countries",fontsize=18)
b.set_ylabel("No. of Records",fontsize=18)
plt.title("Gradings of Carbon Monoxide for all Countries",fontsize=22)
plt.show()
```

Output:

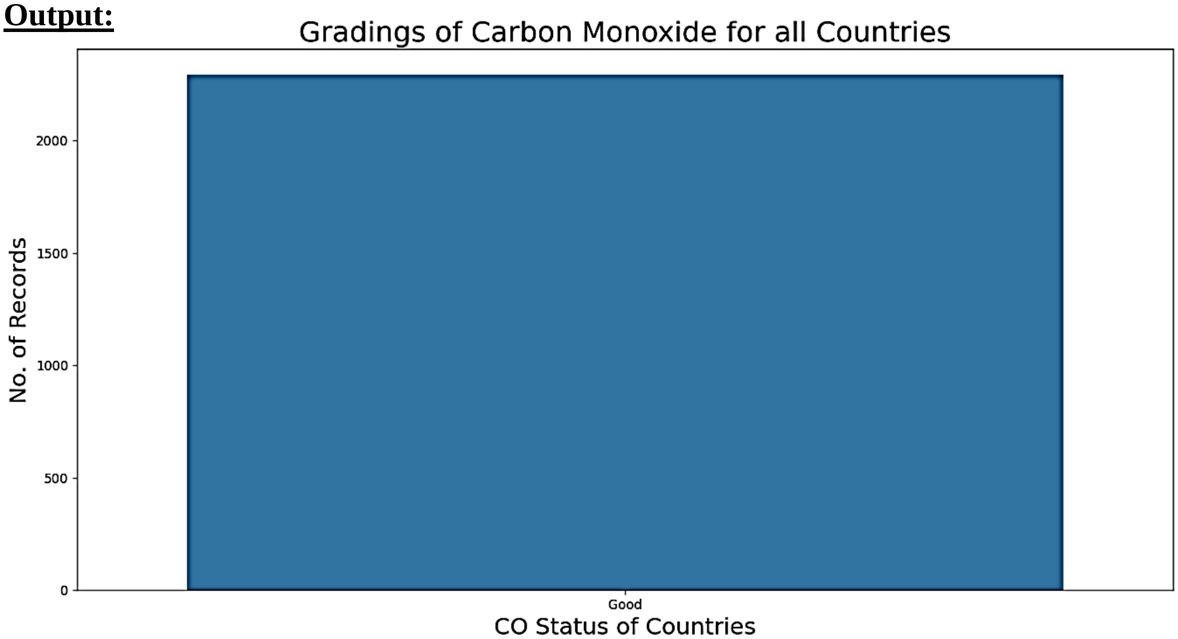


Fig.: Air quality (Carbon Monoxide) of various countries of the world

2. Data Reduction:

In our dataset the features Ozone_Category, CO_Category, NO2_Category and PM_2.5_Category are string values, and in addition to this they are quite imbalanced in their categories and they are derived from other existing features.

Similarly, the country and city name features are also redundant because their string values don't correlate to any of the features or target.

So we shall remove these features.

Code:

```
# We are deleting the following redundant features from df


del df['CO_Category'] # Deleting Carbon Monoxide Status
del df['Ozone_Category'] # Deleting Ozone Status
del df['NO2_Category']  # Deleting Nitrous Oxide Status
del df['PM_2.5_Category'] # Deleting Particulate Status

del df['Country']  #Deleting Country names
del df['City'] # Deleting City names
```

Output:

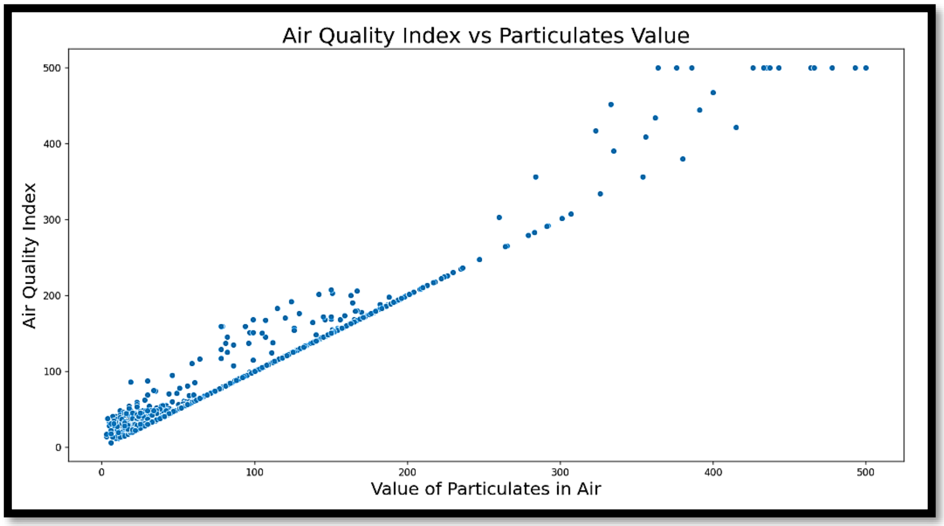
AQI	AQI_Category	CO_Value	Ozone_Value	NO2_Value	PM_2.5_Value
151	Unhealthy	1	41	0	151
117	Unhealthy for Sensitive Groups	1	44	0	117
113	Unhealthy for Sensitive Groups	1	42	0	113
77	Moderate	1	40	0	77
68	Moderate	1	39	1	68
71	Moderate	1	44	1	71
106	Unhealthy for Sensitive Groups	0	45	0	106
40	Good	1	14	0	40
66	Moderate	1	22	0	66
42	Good	2	12	0	42

Now, let us do some further visualization for exploratory data analysis

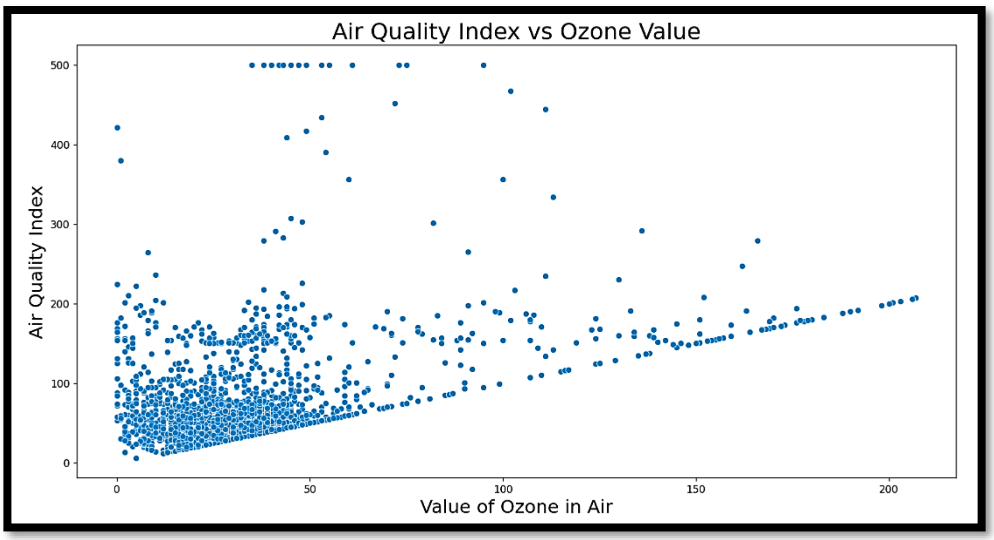
Fig.: Cleaned dataset, ready for training models

Code:

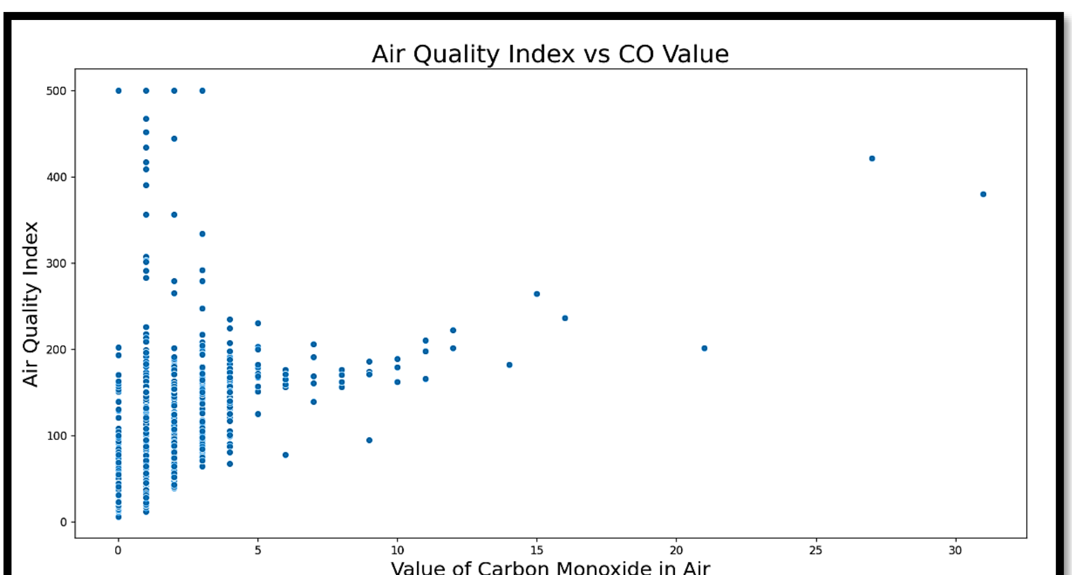
```
# Let us make a scatterplot to see the linear relationship
# between Air Quality Index and 2.5 micron Particulates Value
import seaborn as sns
plt.figure(figsize=(5,5))
b=sns.scatterplot(x=df['PM_2.5_Value'], y=df['AQI'])
b.set_xlabel("Value of Particulates in Air",fontsize=18)
b.set_ylabel("Air Quality Index",fontsize=18)
plt.title("Air Quality Index vs Particulates Value",fontsize=22)
plt.show()



# Let us make another scatterplot to see the linear relationship
# between Air Quality Index and Ozone Value
plt.figure(figsize=(5,5))
b=sns.scatterplot(x=df['Ozone_Value'], y=df['AQI'])
b.set_xlabel("Value of Ozone in Air",fontsize=18)
b.set_ylabel("Air Quality Index",fontsize=18)
plt.title("Air Quality Index vs Ozone Value",fontsize=22)
plt.show()


# Let us make yet another scatterplot to see the linear relationship
# between Air Quality Index and Carbon Monoxide Value
plt.figure(figsize=(5,5))
b=sns.scatterplot(x=df['CO_Value'], y=df['AQI'])
b.set_xlabel("Value of Carbon Monoxide in Air",fontsize=18)
b.set_ylabel("Air Quality Index",fontsize=18)
plt.title("Air Quality Index vs CO Value",fontsize=22)
plt.show()


# Let us make a last scatterplot to see the linear relationship
# between Air Quality Index and Nitrous oxide Value
plt.figure(figsize=(5,5))
b=sns.scatterplot(x=df['NO2_Value'], y=df['AQI'])
b.set_xlabel("Value of Nitrous oxide in Air",fontsize=18)
b.set_ylabel("Air Quality Index",fontsize=18)
plt.title("Air Quality Index vs NO2 Value",fontsize=22)
plt.show()
```



<u>Fig.:</u> Air Quality Index vs Particulate Count in Air



<u>Fig.:</u> Air Quality Index vs Ozone (O_3) Count in Air



<u>Fig.:</u> Air Quality Index vs Carbon-Monoxide Count in Air

Now, let us train our machine learning models. Here we shall use the classifiers SVM, KNN, Logistic Regression, Naïve Bayes and Decision Tree. Our target variable will be AQI_Category since our project requires us to demonstrate classification only.

<u>Code(Splitting Our Dataset):</u>

```
del df['AQI']
df.to_csv('clean_dataset.csv') # Saving Cleaned Dataset

y = df['AQI_Category'].to_numpy() # Air Quality Index is our Target Vector
del df['AQI_Category'] # Features will not include the target
X = df.to_numpy()
# importing train_test_split method from model_selection module
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 1)
# test_size = 0.2 means 80% of our rows will be used for training and
# 20% of our rows will be used for testing
# random_state = 1 guarantees that the split will always be the same
# (i.e.  reproducible results)

print("Data has been split!\n")
# Printing dimensions of training and testing data

print("X_train shape: ", X_train.shape)
print("X_test shape: ", X_test.shape)
print("y_train shape: ", y_train.shape)
print("y_test shape: ", y_test.shape)
```

<u>Output:</u>

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Data has been split!

X_train shape:  (1832, 5)
X_test shape:  (459, 5)
y_train shape:  (1832,)
y_test shape:  (459,)
```

<u>Fig.:</u> Shape of training and testing data

Code(Training our Models):

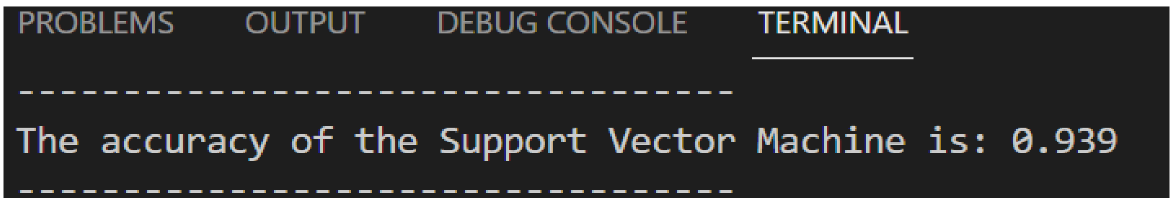
```
# Training SVM
# importing the necessary package to use the classification algorithm
from sklearn import svm
model_svm = svm.SVC(class_weight='balanced') #select the algorithm
model_svm.fit(X_train, y_train) #train the model with the training dataset



# Training KNN
# importing the necessary package to use the classification algorithm
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
#from sklearn.linear_model import LogisticRegression # for Logistic
#Regression algorithm
model_knn = KNeighborsClassifier(n_neighbors=48 )
# n=number of samples=2291 rows,
# n^0.5 neighbours chosen for putting the new data into a class
model_knn.fit(X_train, y_train) #train the model with the training dataset


# Training Logistic Regression
# importing the necessary package to use the classification algorithm
from sklearn.linear_model import LogisticRegression # for Logistic Regression
#algorithm
model_lr = LogisticRegression(max_iter=3000, class_weight='balanced')
model_lr.fit(X_train, y_train) #train the model with the training dataset


# Training Naïve Bayes
# importing the necessary package to use the classification algorithm
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB( )
model_nb.fit(X_train, y_train) #train the model with the training dataset



# Training Decision Tree
# importing the necessary package to use the classification algorithm
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree
Algoithm
model_dt = DecisionTreeClassifier( )
model_dt.fit(X_train, y_train) #train the model with the training dataset
```

<u>Code(Testing our Models):</u>

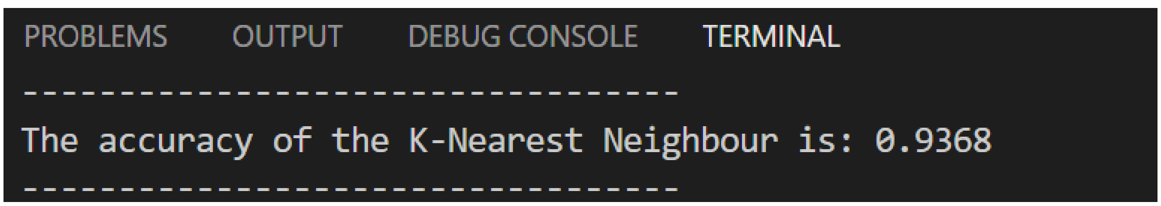
```
# Testing SVM
y_prediction_svm = model_svm.predict(X_test) # pass the testing data to the
#trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_svm = metrics.accuracy_score(y_prediction_svm, y_test).round(4)

print("----------------------------------")
print('The accuracy of the Support Vector Machine is: {}'.format(score_svm))
print("----------------------------------")

# Testing KNN
y_prediction_knn = model_knn.predict(X_test) #pass the testing data to the
trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_knn = metrics.accuracy_score(y_prediction_knn, y_test).round(4)
print("----------------------------------")
print('The accuracy of the K-Nearest Neighbour is: {}'.format(score_knn))
print("----------------------------------")

# Testing Logistic Regression
y_prediction_lr = model_lr.predict(X_test) #pass the testing data to the
trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_lr = metrics.accuracy_score(y_prediction_lr, y_test).round(4)
print("---------------------------------")
print('The accuracy of the Logistic Regression is: {}'.format(score_lr))
print("---------------------------------")

# Testing Naïve Bayes
y_prediction_nb = model_nb.predict(X_test) #pass the testing data to the
trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_nb = metrics.accuracy_score(y_prediction_nb, y_test).round(4)
print("---------------------------------")
print('The accuracy of the Naive Bayes Classifier is: {}'.format(score_nb))
print("---------------------------------")
```

<u>Code(Testing our Models)(Cont'd):</u>

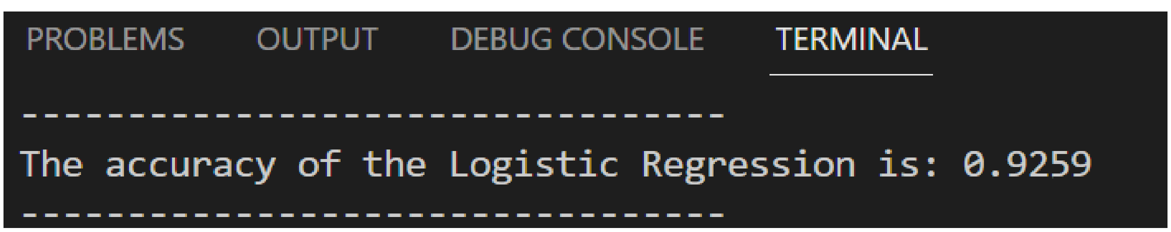
```
# Testing Decision Tree
y_prediction_dt = model_dt.predict(X_test)
#pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_dt = metrics.accuracy_score(y_prediction_dt, y_test).round(4)
print("---------------------------------")
print('The accuracy of the Decision Tree Classifier is: {}'.format(0.9622))
print("---------------------------------")
```



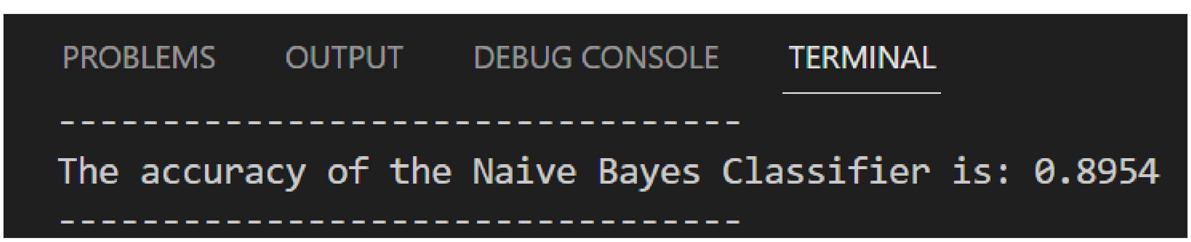
<u>Fig.:</u> Testing accuracy of SVM



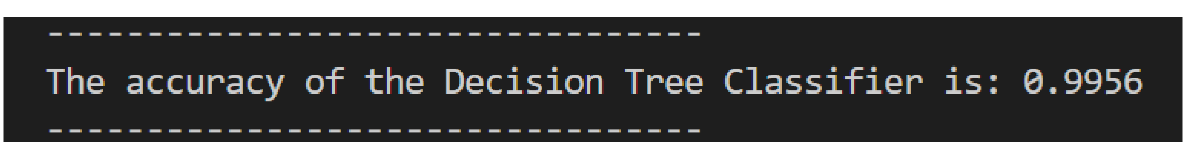
<u>Fig.:</u> Testing accuracy of KNN



<u>Fig.:</u> Testing accuracy of Logistic Regression



<u>Fig.:</u> Testing accuracy of Naïve Bayes



<u>Fig.:</u> Testing accuracy of Decision Tree

Code(Comparing Model Performances):

```
import numpy as np
#Making 2 numpy arrays for our model names and their classification
accuracies
model_names = np.array(['SVM', 'KNN', 'Logistic Regression', 'Naive Bayes',
'Decision Tree'])
model_accuracies = np.array([score_svm, score_knn, score_lr, score_nb,
score_dt])

model_names_axis = np.arange(len(model_names))#List of index, will be used in
labelling

# Making bar plot with unique color for each bar
plt.bar(model_names,model_accuracies, color=['purple', 'red', 'green',
'blue', 'orange'])

# Setting labels below each bar
plt.xticks(model_names_axis, model_names, fontsize=15)
plt.xlabel("Models",fontsize=20) # X axis label
plt.ylabel("Accuracy",fontsize=20) # Y axis label
plt.title("Classification Accuracy of Alll the Classifiers used",fontsize=22)
# Title

plt.show()
```

Output

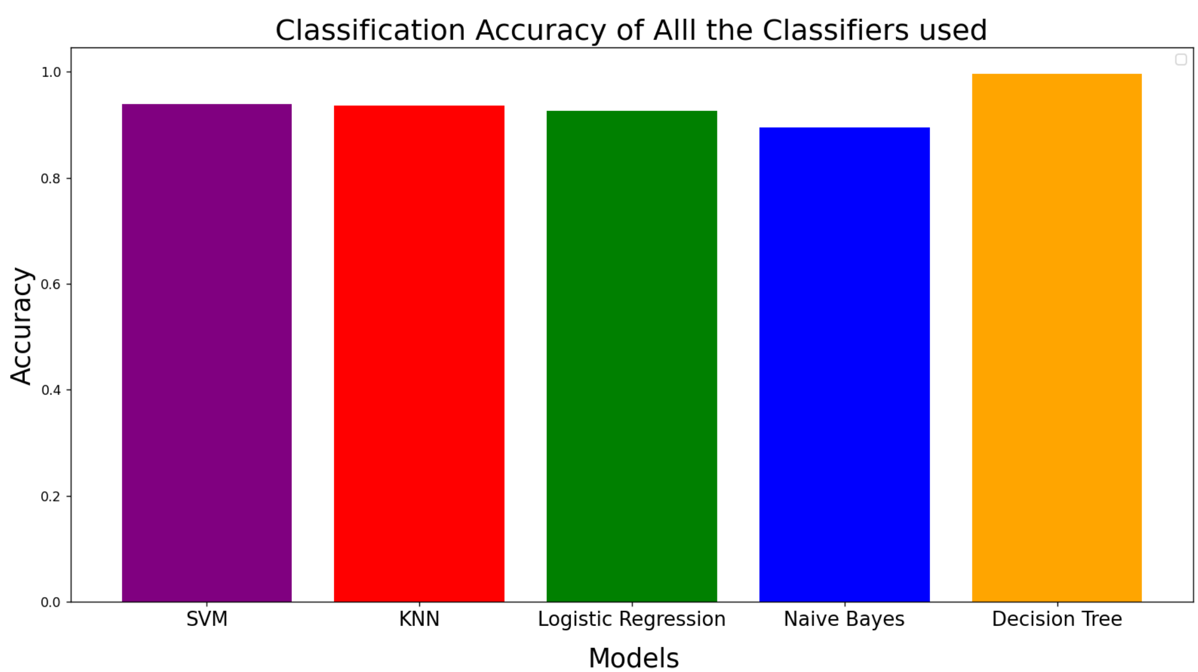


Fig.: Comparing classification accuracy of all the models

Code(Generating Confusion Matrices):

```
#KNN SVM matrix-------------------------------------------------------
cm_svm = confusion_matrix(y_test, y_prediction_svm)
sns.heatmap(cm_svm, annot=True, cmap='Greens',
xticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'],
yticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'])

plt.xlabel('Predicted labels(SVM)',fontsize=16)
plt.xticks(rotation=0,   fontsize=14)
plt.yticks(rotation=0,   fontsize=14)
plt.ylabel('True labels(SVM)',fontsize=16)
plt.title('Confusion Matrix(SVM)',fontsize=20)
plt.show()


#KNN confusion matrix--------------------------------------------------
cm_knn = confusion_matrix(y_test, y_prediction_knn)
sns.heatmap(cm_knn, annot=True, cmap='Purples',
xticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'],
yticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'])

plt.xlabel('Predicted labels(KNN)',fontsize=16)
plt.xticks(rotation=0,   fontsize=14)
plt.yticks(rotation=0,   fontsize=14)
plt.ylabel('True labels(KNN)',fontsize=16)
plt.title('Confusion Matrix(KNN)',fontsize=20)
plt.show()

#DT confusion matrix-----------------------------------------------------
cm_dt = confusion_matrix(y_test, y_prediction_dt)
sns.heatmap(cm_dt, annot=True, cmap='Greys',
xticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'],
yticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'])

plt.xlabel('Predicted labels(Decision Tree)',fontsize=16)
plt.xticks(rotation=0,   fontsize=14)
plt.yticks(rotation=0,   fontsize=14)
plt.ylabel('True labels(Decision Tree)',fontsize=16)
plt.title('Confusion Matrix(Decision Tree)',fontsize=20)
plt.show()
```

```
#Naive Bayes confusion matrix--------------------------------------------------
cm_nb = confusion_matrix(y_test, y_prediction_nb)
sns.heatmap(cm_nb, annot=True, cmap='YlOrBr',
xticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'],
yticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'])

plt.xlabel('Predicted labels(Naive Bayes)',fontsize=16)
plt.xticks(rotation=0,   fontsize=14)
plt.yticks(rotation=0,   fontsize=14)
plt.ylabel('True labels(Naive Bayes)',fontsize=16)
plt.title('Confusion Matrix(Naive Bayes)',fontsize=20)
plt.show()


#Logistic Regression confusion matrix----------------------------------------

cm_lr = confusion_matrix(y_test, y_prediction_lr)
sns.heatmap(cm_lr, annot=True, cmap='Reds',
xticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'],
yticklabels=['Good','Hazardous','Moderate','Unhealthy','UFSG','Very
Unhealthy'])

plt.xlabel('Predicted labels(Logistic Regression)',fontsize=16)
plt.xticks(rotation=0,   fontsize=14)
plt.yticks(rotation=0,   fontsize=14)
plt.ylabel('True labels(Logistic Regression)',fontsize=20)

plt.show()
```

Output(Confusion Matrix for each Model)

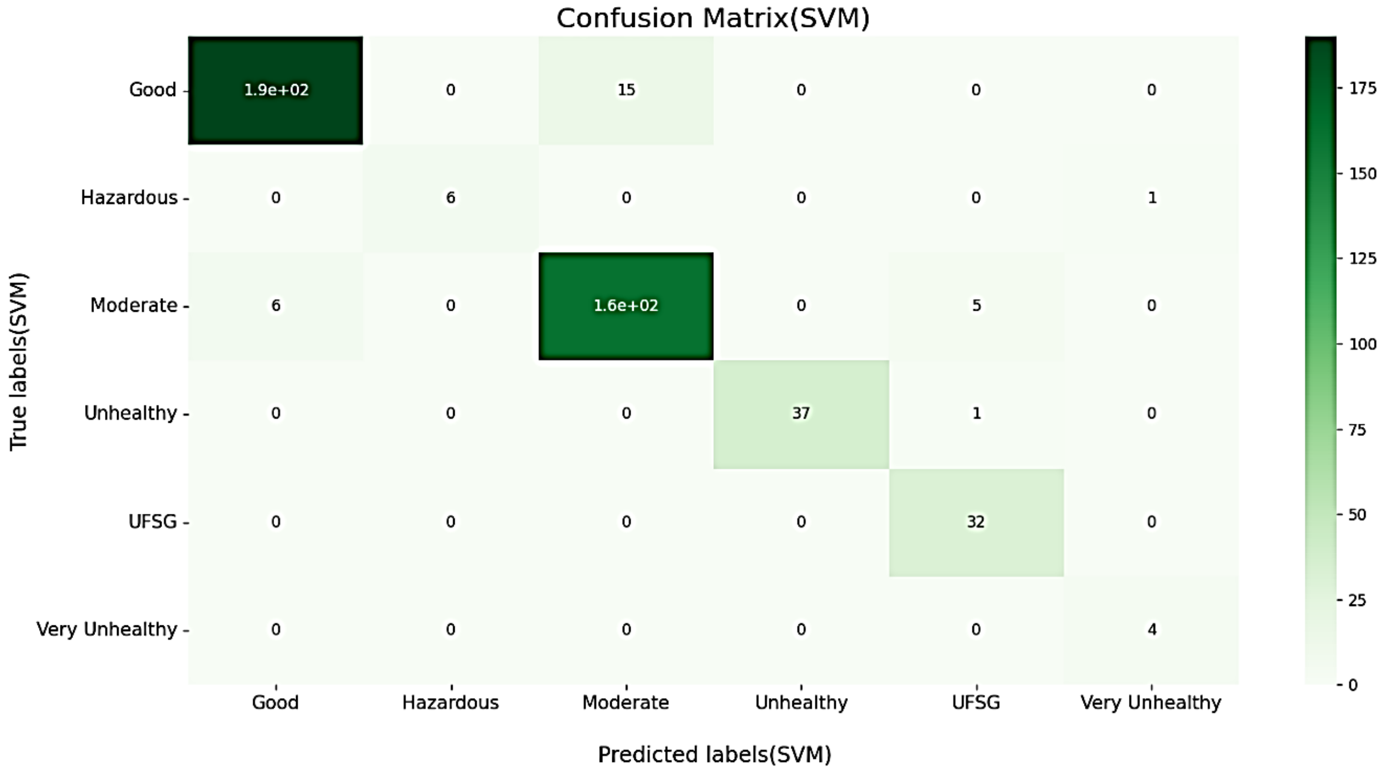


Fig.: Confusion matrix for SVM

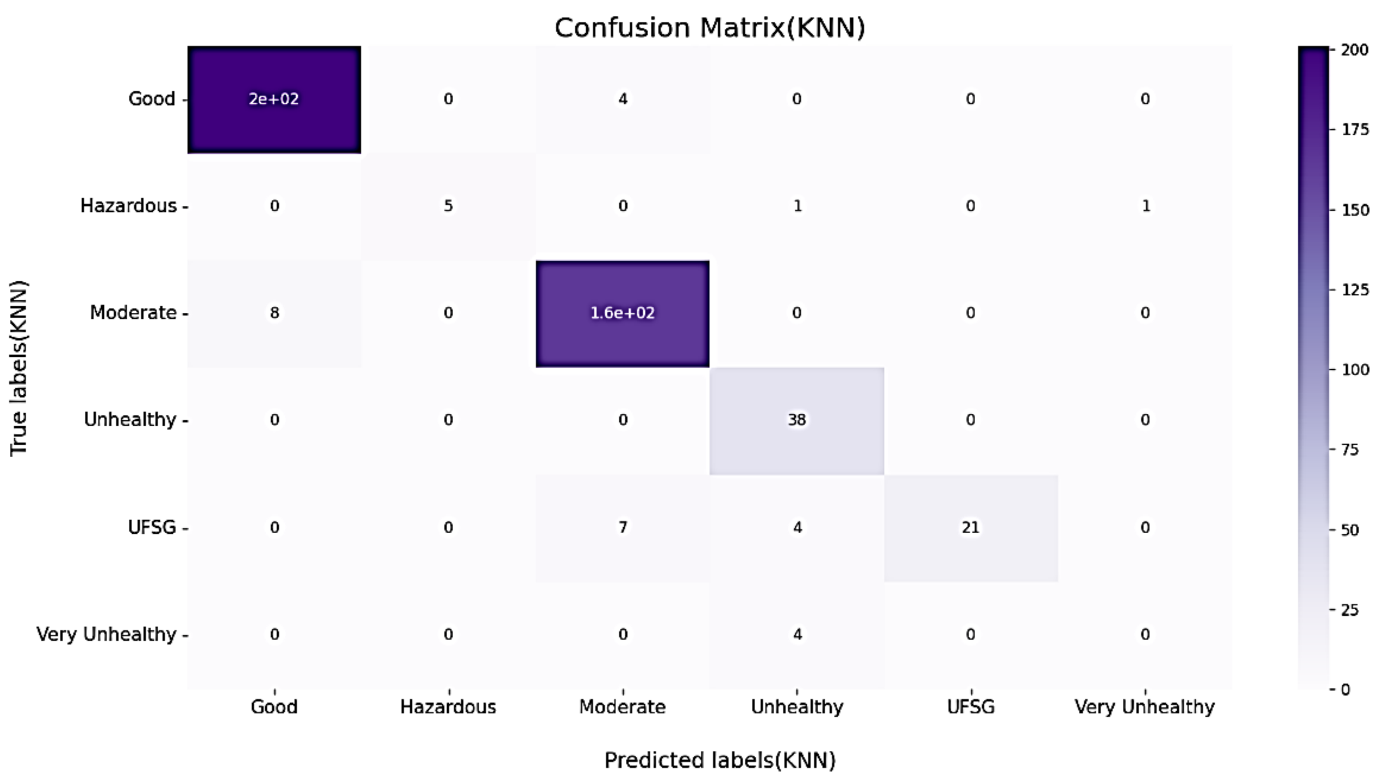


Fig.: Confusion matrix for KNN

Output(Confusion Matrix for each Model)

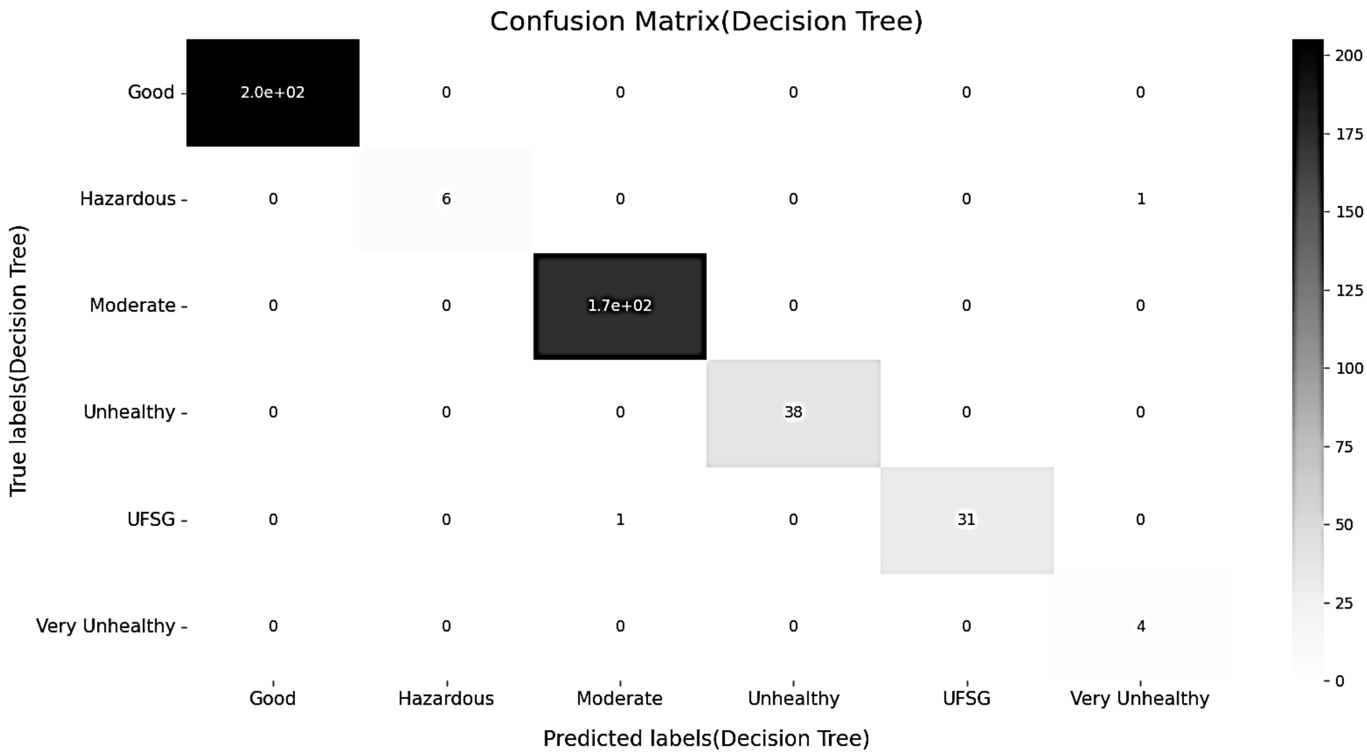


Fig.: Confusion matrix for Decision Tree (Max Accuracy)

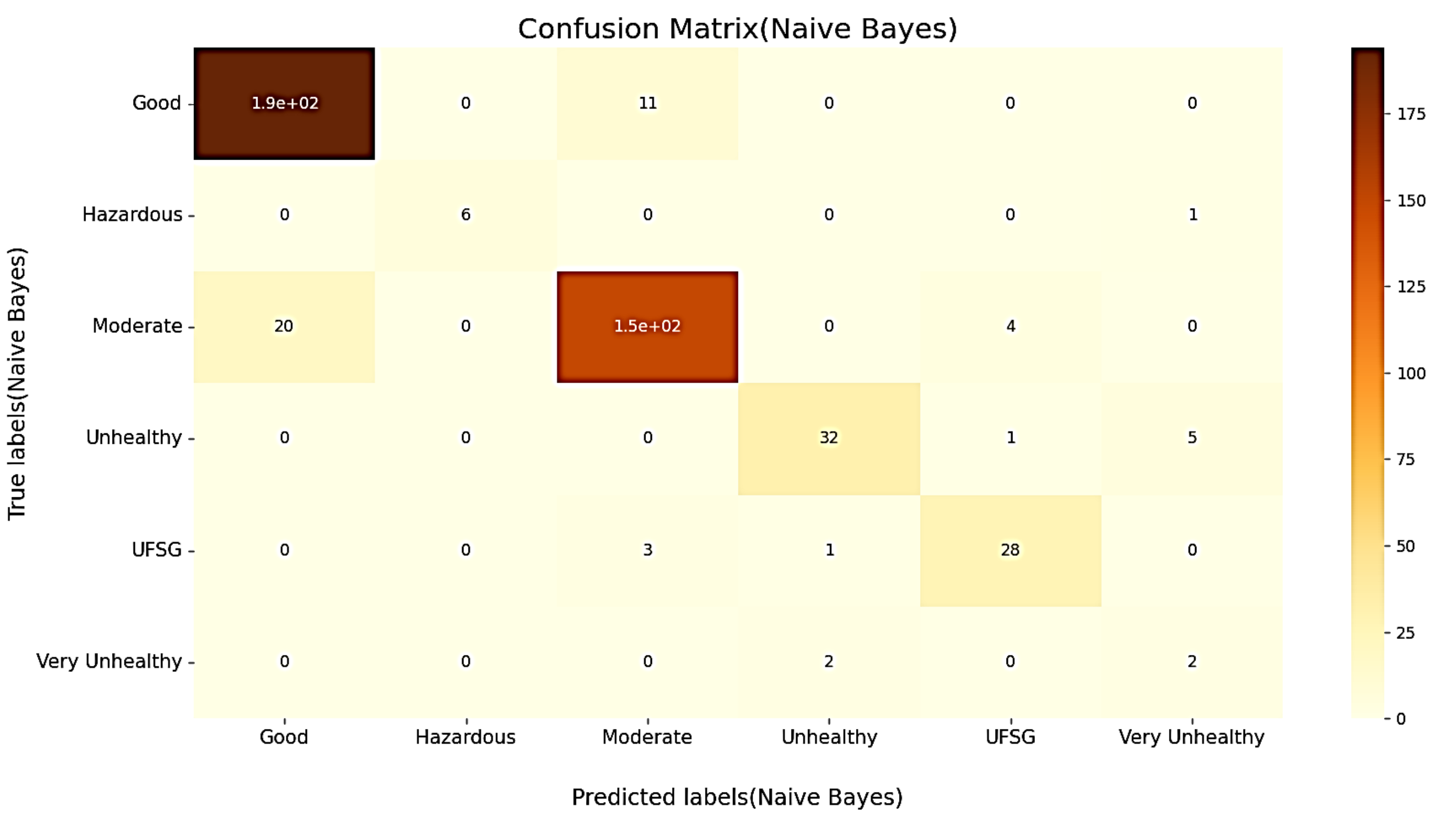


Fig.: Confusion matrix for Naïve Bayes

Output(Confusion Matrix for each Model)

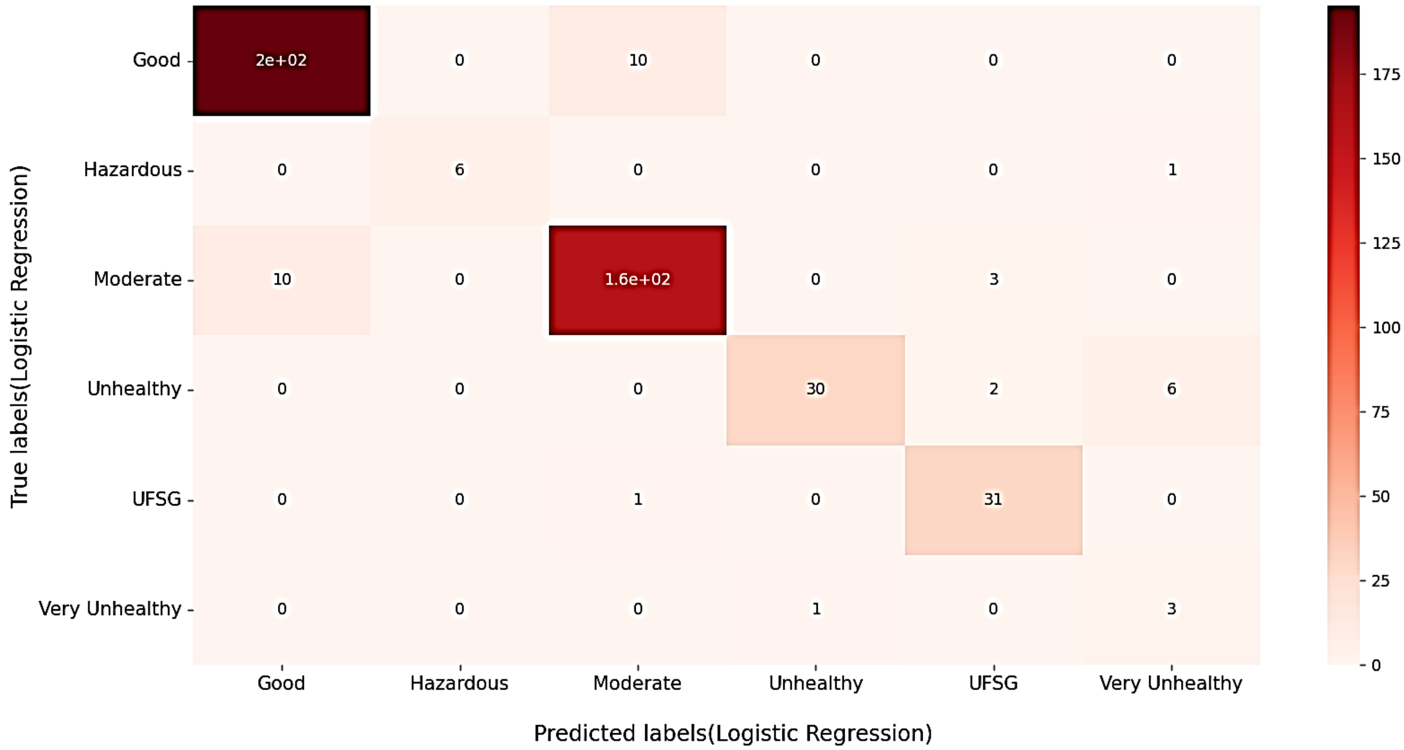


Fig.: Confusion matrix for Logistic Regression

Conclusion: The goal of this project was to analyze a dataset about global air pollution figures and using it to train machine learning classifiers. Here imported the dataset, preprocessed and visualized it and then trained 5 classifiers with it.

On testing all the models gave satisfactory results, with Decision Tree giving the best results at 99%. Thus we believe our project has met all the requirements and it is a success.

Note: Our project file and processed dataset can be downloaded from the following link: https://drive.google.com/drive/u/0/folders/1iJmL_KtmqmBqFyha3IuDubalmkA4WcFY