Classification of AQI Between Good, Moderate and Poor

Group 20

Prepared By:

Shakti Singh Rathore

Bablu Prajapati

Problem Statement

To classify the AIR QUALITY INDEX in Good, Moderate and Poor Based on the feature data collected form Government of India

SUMMARY

To do data analysis on India Air Quality data and predict the value of Air Quality Index based on given features of concentration of Sulphur dioxide, nitrogen dioxide, respirable suspended particulate matter, suspended particulate matter and classify the Air Quality as good, moderate, poor.

The data is combined (across the years and states) and largely clean version of the Historical Daily Ambient Air Quality Data released by the Ministry of Environment and Forests and Central Pollution Control Board of India under the National Data Sharing and Accessibility Policy (NDSAP) year 1990-2015

Since industrialization, there has been an increasing concern about environmental pollution. As mentioned in the WHO report, 7 million premature deaths annually are linked to air pollution. Air pollution is the world's largest single environmental risk.

it has been found that India's air pollution is deadlier than even China’s. Using this dataset, one can explore India's air pollution levels at a more granular scale.

Detailed Description

Dataset:

The dataset used in this story is publicly available and was created by the Government of India. To create the data Government of India used different sample of different gases like Sulphur dioxide, nitrogen dioxide, respirable suspended particulate matter, suspended particulate matter

We can observe that the data set contains 2132 rows and 13 columns. 'AQI_LABEL' is the column which we are going to predict, which says if AQI 0 means the pollution category is good ,1 means moderate and 2 means poor. Here we deal with multiclass data and we have to use
different techniques to balance dataset, hot encoding and various preprocessing techniques like Standard Scaler, SMOTE for minority class to generate synthetic samples so we can balance the class so that model is not biases towards the class which ha higher number of observation and various hyper parameter tuning for different models.

Features information:

- stn_code- code of the observing station
- sampling_date date of sample taken
- State name of state
- Location location
- Agency responsible agency
- Type type of area like residential or industrial
- so2 concentration of so2
- no2 concentration of no2
- rspm concentration of rspm
- spm concentration of spm
- location_monitoring_station location of that station
- pm2_5 concentration of pm 2.5
- year year of sample taken

Analysis

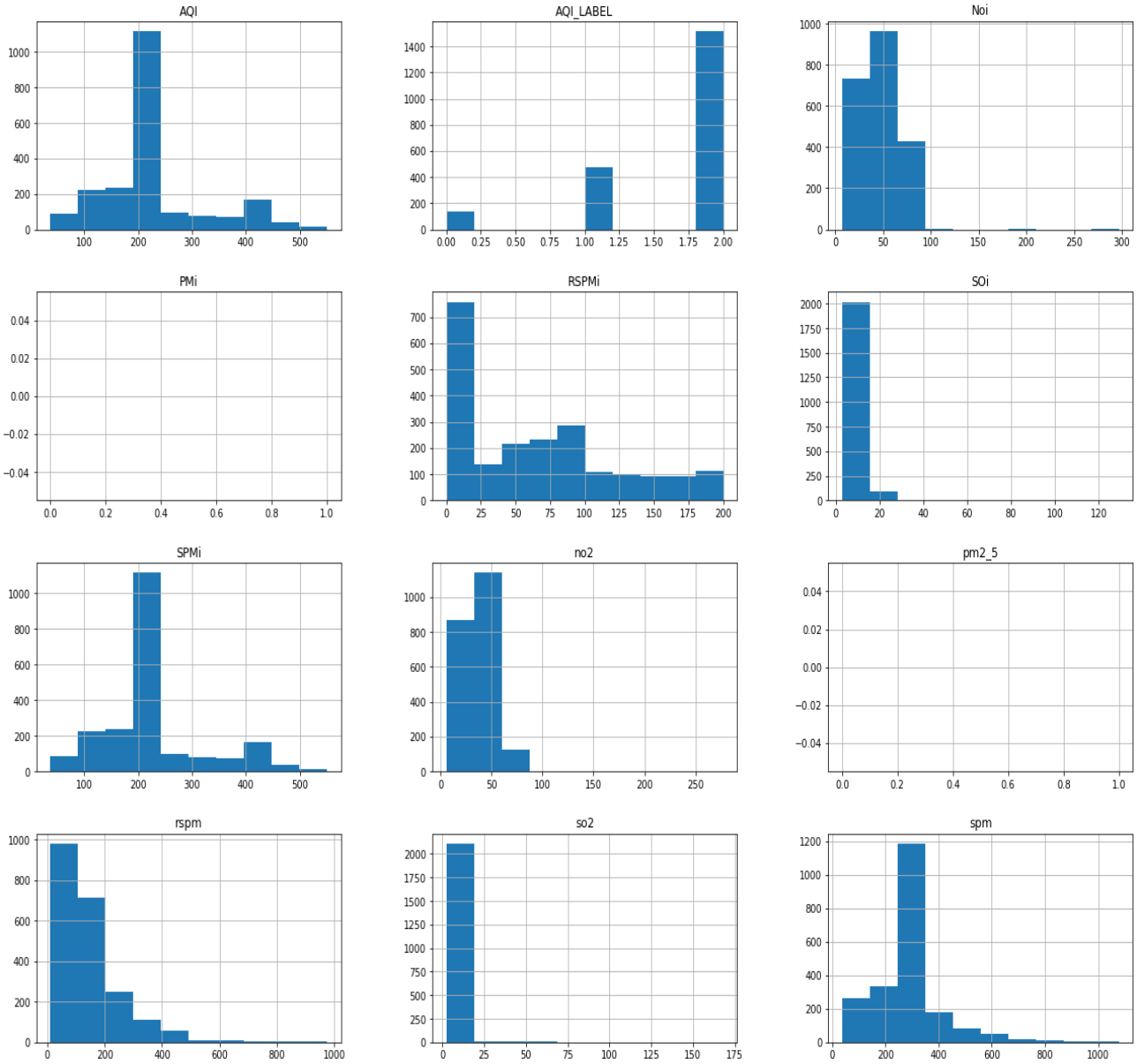
We use python language for analysis of our data. We will use Different libraries to help us in model building some of them listed below.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
%matplotlib inline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report
```

Visualization:

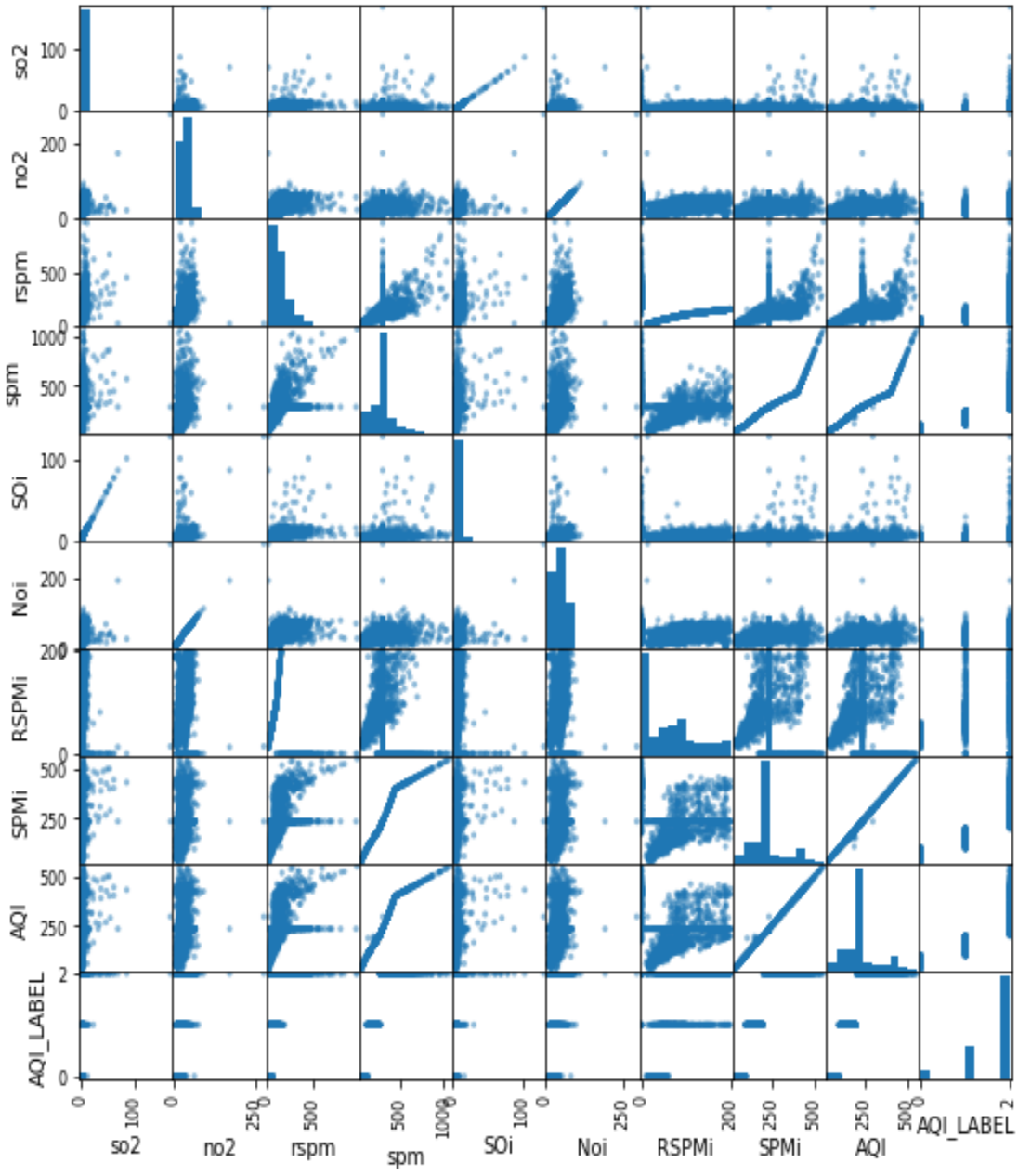
Visualization of data is an imperative aspect of data science. It helps to understand data and also to explain the data to another person.

Histogram:

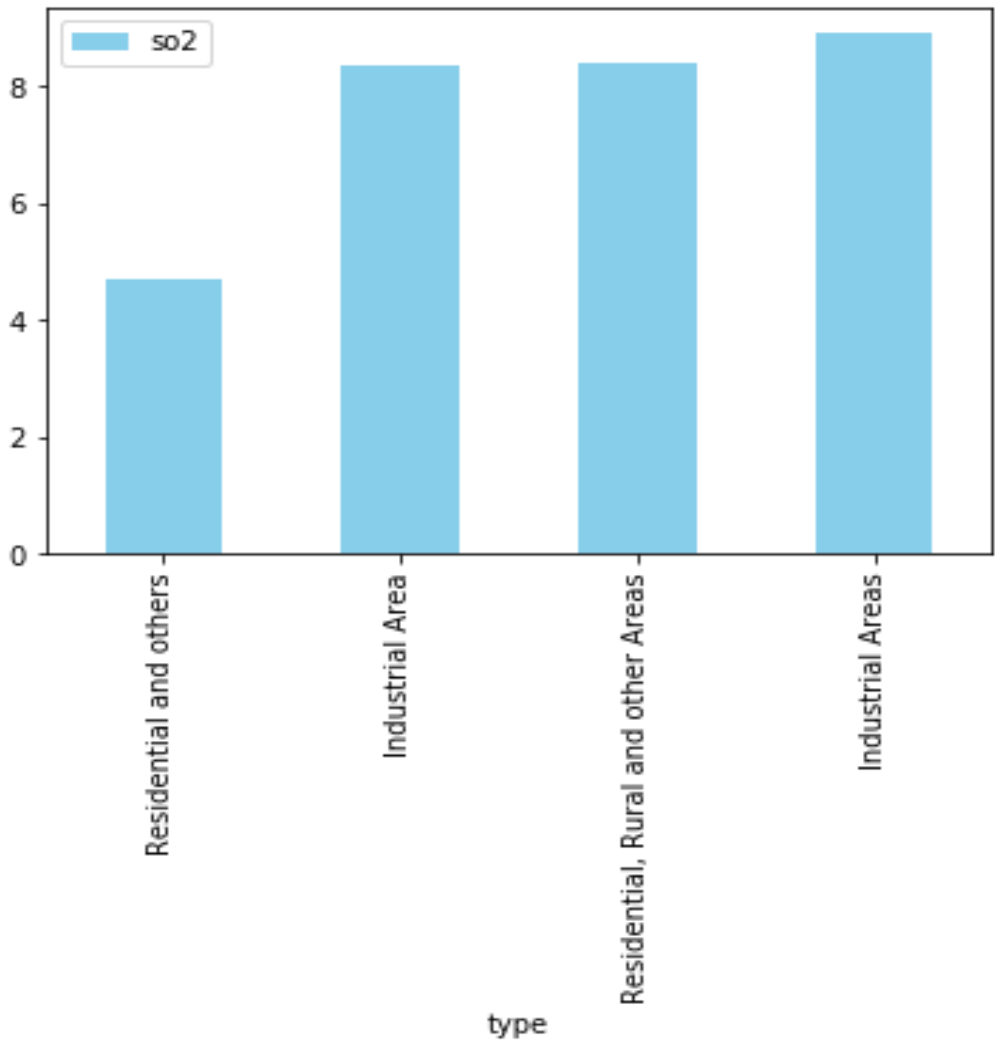


Scatter Plot Matrix:

```
scatter_matrix(df5,figsize=(9,9))
plt.show
```

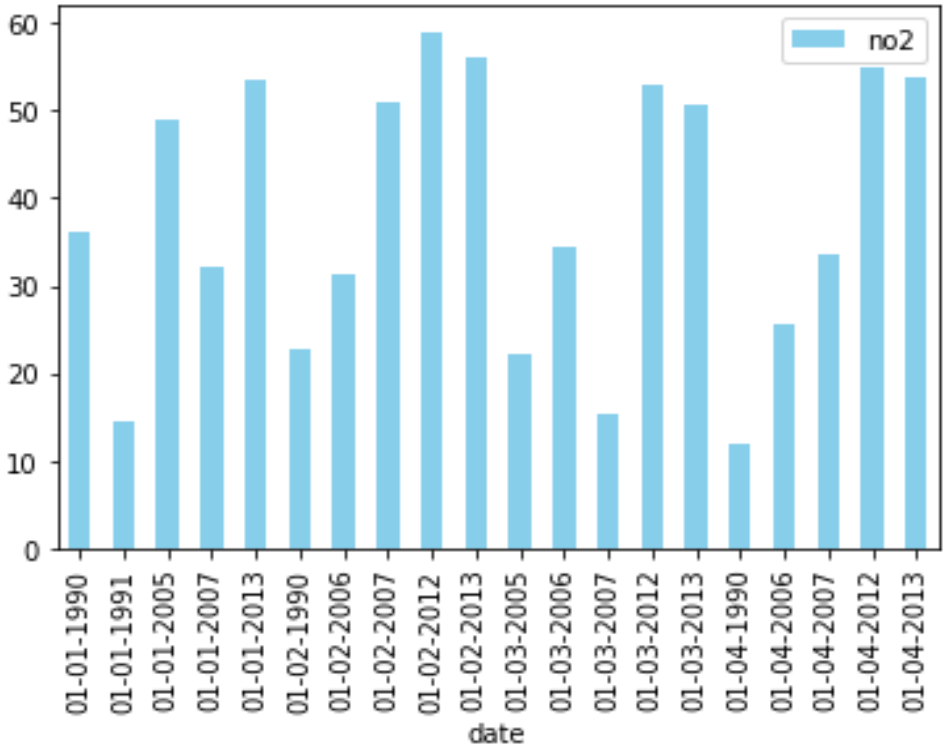


visualization of concentration of gases in different types of areas and from the graph we can clearly see that emissions of gases in industrial areas are higher than any other area and residential areas emit less gases because there are no industrial activities.



The below graph shows that emissions of gases are increasing year by year.

```
df[['no2','date']].groupby(["date"]).mean().sort_values(by='date').head(20).plot.bar(color='skyblue')
plt.show()
```



Algorithms

1. Logistic Regression:

Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Uses:

• Logistic regression deals with this problem by using a logarithmic transformation on the outcome variable which allows us to model a nonlinear association in a linear way.

• To predict an outcome variable that is categorical from predictor variables that are continuous and/or categorical.

Model:

Imported Logistic Regression from sklearn libraries for the algorithm to fit the model. The score of the classifier is 0.91%

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
X1 = df3[["SOi","Noi","RSPMi","SPMi"]]
Y1 = df3['AQI_LABEL']
```

```
x_train, x_test, y_train, y_test = train_test_split(X1, Y1, test_size=0.3, random_state=10)
```

```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

A confusion matrix table is used to describe the performance of a classification model on test data for which the true values known.

```
from sklearn.metrics import confusion_matrix
con_mat=confusion_matrix(y_test,pred)
print(con_mat)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

The accuracy of this model is 0.91%

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.65	0.72	43
1	0.83	0.83	0.83	157
2	0.96	0.97	0.97	440
accuracy			0.92	640
macro avg	0.86	0.82	0.84	640
weighted avg	0.92	0.92	0.92	640

2. Logistic Regression with SMOTE (Synthetic Minority Over-sampling Technique)

Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

here we use SMOTE technique to balance the class i.e. SMOTE generate synthetic sample in the minority class

```
from imblearn.over_sampling import SMOTE
a = df3[["SOi","Noi","RSPMi","SPMi"]]
b = df3['AQI_LABEL']
```

```
x_train0, x_test0, y_train0, y_test0 = train_test_split(a, b, test_size=0.3, random_state=10)
```

Standard Scaling:

Here Standard Scaling is used to scale the feature.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
x_train0 = sc_X.fit_transform(x_train0)
x_test0 = sc_X.transform(x_test0)
```

```
pred=model.predict(x_test)
```

```
model.score(x_test,y_test)
```

Applying SMOTE

```
from imblearn.over_sampling import SMOTE
smt = SMOTE()
x_train0, y_train0 = smt.fit_sample(x_train0, y_train0)
```

Fitting the model:

```
model2 = LogisticRegression()
model2.fit(x_train0,y_train0)
```

```
np.bincount(y_train0)
y_pred0=model2.predict(x_test0)
```

```
model2.score(x_test0,y_test0)
```

```
confusion_matrix(y_test0, y_pred0)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test0, y_pred0))
```

The accuracy of this model is 97%

Classification Report:

	precision	recall	f1-score	support
0	0.93	1.00	0.97	43
1	0.99	0.97	0.98	157
2	1.00	1.00	1.00	440
accuracy			0.99	640
macro avg	0.98	0.99	0.98	640
weighted avg	0.99	0.99	0.99	640

3. K-Nearest Neighbors

k-Nearest-Neighbor algorithm is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

Uses:

- Easier to use and therefore known as, lazy learner.
- It makes highly accurate predictions.

Model:

Imported K-Neighbors Classifier from SKLearn libraries for the algorithm and used five neighbor values to fit the model. As we see below, the score of this model is 0.979%.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
X2 = df3[["SOi","Noi","RSPMi","SPMi"]]
Y2 = df3['AQI_LABEL']
```

Standard Scaling:

Standard Scaling ensures that all features are mapped to the same range of values; it scales all the feature in a specific range and increase the performance of our model.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
x_train2 = sc_X.fit_transform(x_train1)
x_test2 = sc_X.transform(x_test1)
```

```
y_predict = model3.predict(x_test2)
```

```
accuracy_score(y_test1,y_predict)
```

```
confusion_matrix(y_test1, y_predict)
```

```
print(classification_report(y_test1,y_predict))
```

The accuracy of this model is .979%

Classification Report:

	precision	recall	f1-score	support
0	0.91	1.00	0.96	43
1	0.97	0.94	0.96	157
2	0.99	0.99	0.99	440
accuracy			0.98	640
macro avg	0.96	0.98	0.97	640
weighted avg	0.98	0.98	0.98	640

4.Support Vector Classification:

Support Vector Classification (SVC) are capable of performing multi-class classification on a dataset. SVC implements the “one-against-one” approach for multi- class classification.

Uses:

• Effective in high dimensional spaces.

• SVM are an effective tool in real-value function estimation

Model:

Imported Support Vector Classification from sklearn libraries for the algorithm to fit the model. The linear kernel maps a single vector to a vector of higher dimensionality. The classifier score is 0.98%

```
from sklearn.svm import SVC
```

```
X3 = df3[["SPMi","SOi","Noi","RSPMi"]]
Y3 = df3['AQI_LABEL']
```

```
x_train3, x_test3, y_train3, y_test3 = train_test_split(X3, Y3, test_size=0.3, random_state=10)
```

Standard Scaling

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
x_train4 = sc_X.fit_transform(x_train3)
x_test4 = sc_X.transform(x_test3)
```

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 30)
classifier.fit(x_train4, y_train3)
```

```
Y_Pred = classifier.predict(x_test4)
```

```
accuracy_score(y_test3,Y_Pred)
```

```
cm = confusion_matrix(y_test3, Y_Pred)
cm
```

```
print(classification_report(y_test3,Y_Pred))
```

The accuracy of this model is .98%

Classification Report:

	precision	recall	f1-score	support
0	0.88	1.00	0.93	43
1	0.94	0.96	0.95	157
2	1.00	0.98	0.99	440
accuracy			0.98	640
macro avg	0.94	0.98	0.96	640
weighted avg	0.98	0.98	0.98	640

Conclusion

Model	Accuracy	Avg. Precision	Avg. Recall
Support Vector Classifier	**.98**	**.94**	**.98**
KNN Classifier	**0.979**	**.95**	**.97**
Logistic Regression with SMOTE	**0.978**	**.97**	**.99**
Logistic Regression	**0.92**	**.86**	**.81**

According to above data we can say that Logistic Regression with SMOTE is the best model which gives higher precision and recall and help to identify the category of pollution.

Lessons learnt from the project:

learn various types of machine learning algorithm and EDA process and cleaning of dataset, various types of encoding and how to handle class imbalance using SMOTE etc.

Certificate of originality: All the content of report/code/are created by Shakti and Bablu.