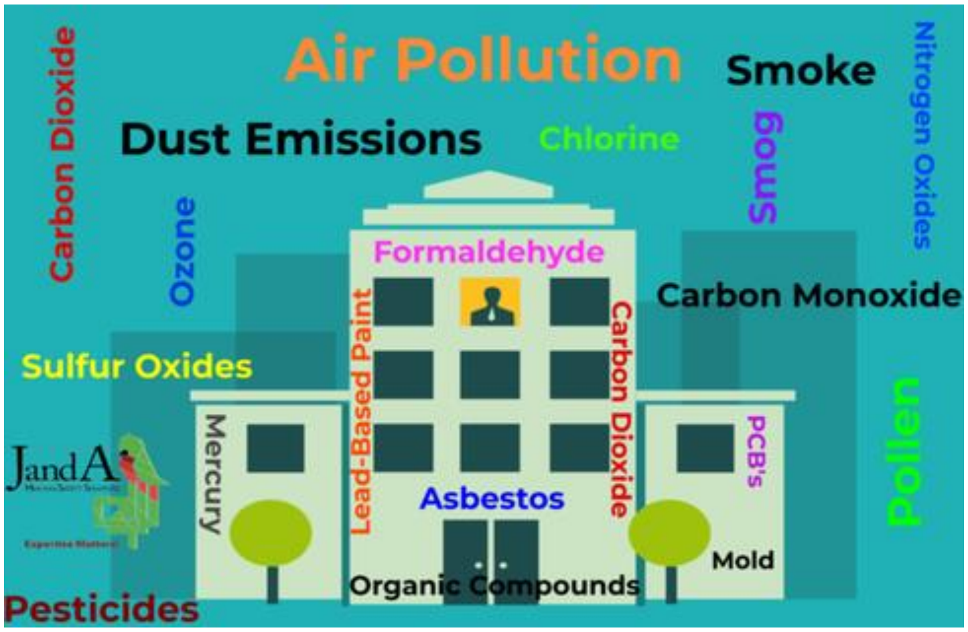
AIR Q ASSESSMENT TN USING MACHINE LEARNING



Introduction:

- Air quality assessment is a critical endeavor aimed at understanding and managing the complex issue of air pollution in the Indian state of Tamil Nadu. Located in the southern part of India, Tamil Nadu is known for its diverse landscapes, cultural heritage, and thriving urban centers. However, rapid industrialization, urbanization, and population growth have led to significant challenges concerning air quality and public health.

- The state's air quality is influenced by various factors, including industrial emissions, vehicular traffic, agricultural activities, meteorological conditions, and geographic features. As a result, assessing and improving air quality in Tamil Nadu is of paramount importance to safeguard the well-being of its residents and preserve its natural environment.

MODEL EVALUATION AND SELECTION:

- Split the data into training and testing sets.
- Choose an appropriate regression model (e.g., linear regression, random forest).
- Train the model on historical data.
- Evaluate the model's performance using metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE).
- Select the best-performing model for further analysis.

MODEL INTERPRETABILITY:

- Explain how to interpret feature importance from Gradient Boosting and XGBoost models.
- Discuss the insights gained from feature importance analysis and their relevance to air quality assessment in TN.
- Interpret feature importance from ensemble models like Random Forest and Gradient Boosting to understand the factors influencing the air quality assessment.

PROGRAM:

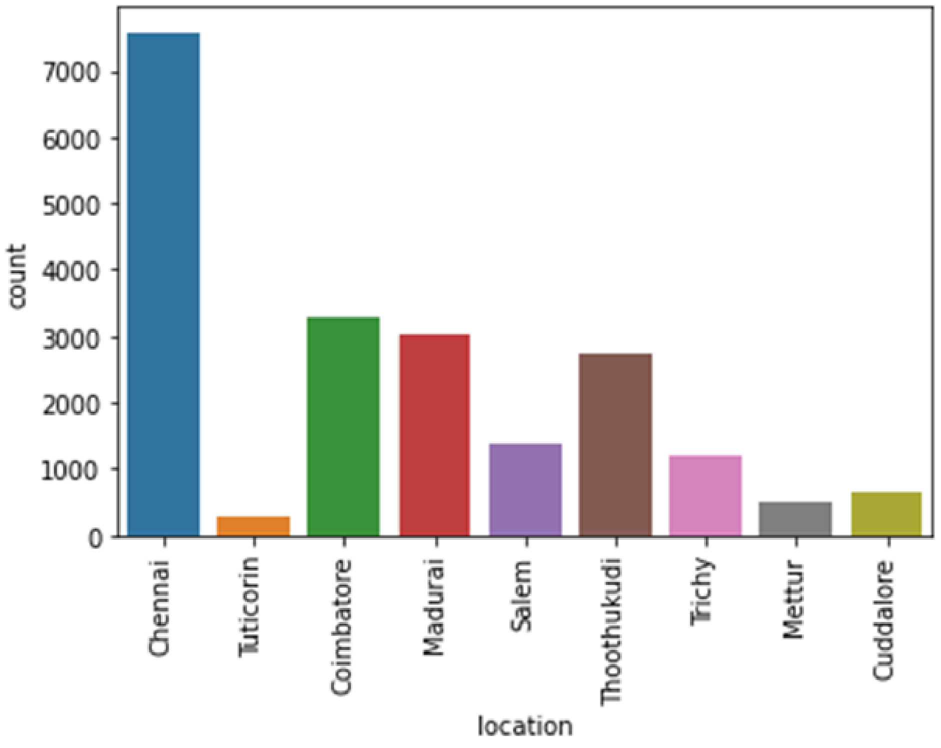
AIR QUALITY ASSESSMENT

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

print(os.listdir("../input"))
```

```
aq=pd.read_csv('../input/india-air-quality-data/data.csv',encoding="ISO-
8859-1")aq.tail(5)#Extracting Tamil Nadu state alonetn =
aq.query('state=="Tamil Nadu" ')tn.sample(2)tn.shapetn.describe(include
'all')
```

DATA VISUALIZATION:

```
datacount =sns.countplot(x ="location",data = tn);

datacount.set_xticklabels(datacount.get_xticklabels(), rotation=90);
```



In[1]:

```
loc = pd.pivot_table(tn, values=['so2','no2','spm'],index='location')

 ## Aggfunc: default-np.mean()

loc
Out[1]:
```

LOCATION	*no2*	*so2*	spm
Chennai	18.551330	11.905157	199.767056

Coimbatore	29.374767	5.832845	188.888683
Cuddalore	19.772657	9.110599	267.000000
Madurai	24.420616	11.153280	179.156298

LOCATION	***no2***	***so2***	*spm*
Mettur	24.039095	8.399177	267.000000
Salem	25.764407	8.190645	179.440385
Thoothukudi	16.948928	17.532772	210.858009
Trichy	18.211327	13.753170	267.000000
Tuticorin	14.505208	10.176389	51.322917

MODEL 1-POLYNOMIAL REGRESSION

```
In[1]:

from sklearn.preprocessing import PolynomialFeatures

from sklearn import linear_model poly = PolynomialFeatures(degree=2,
interaction_only=True)X_train2 = poly.fit_transform(X_train)X_test2 =
poly.fit_transform(X_test) poly_clf = linear_model.LinearRegression()
poly_clf.fit(X_train2, y_train) y_pred =
poly_clf.predict(X_test2)In[2]:print(poly_clf.score(X_train2,
y_train))0.5060498134379463In[3]:

print(poly_clf.score(X_test2, y_test))

0.5262979517606676


# Trying with higher degrees

In[4]:

poly = PolynomialFeatures(degree=3, interaction_only=True)

X_train2 = poly.fit_transform(X_train)

X_test2 = poly.fit_transform(X_test)
```

```
poly_clf = linear_model.LinearRegression()


poly_clf.fit(X_train2, y_train)


y_pred = poly_clf.predict(X_test2)
print(poly_clf.score(X_train2, y_train))
print(poly_clf.score(X_test2, y_test))

# degree = 3 has less scores than degree = 2
```

In [5]:

```
poly = PolynomialFeatures(degree=4, interaction_only=True)
X_train2 = poly.fit_transform(X_train)
X_test2 = poly.fit_transform(X_test)


poly_clf = linear_model.LinearRegression()


poly_clf.fit(X_train2, y_train)


y_pred = poly_clf.predict(X_test2)
print(poly_clf.score(X_train2, y_train))
print(poly_clf.score(X_test2, y_test))
```

```
0.4956914409476182
0.5164060494757956
```

```
# Nearly score to degree = 2. But still less than degree = 2
```

In [6]:

```
poly = PolynomialFeatures(degree=5, interaction_only=True)
X_train2 = poly.fit_transform(X_train)
X_test2 = poly.fit_transform(X_test)
```

```
poly_clf = linear_model.LinearRegression()


poly_clf.fit(X_train2, y_train)


y_pred = poly_clf.predict(X_test2)

print(poly_clf.score(X_train2, y_train))

print(poly_clf.score(X_test2, y_test))

0.4956914409476182

0.5164060494757956
```

MODEL 2-SIMPLE LINEAR REGRESSION

```
In[1]:from sklearn.linear_model import LinearRegression  In[2]:lin_mod =
LinearRegression()lin_mod.fit(X_train, y_train) Out[3]:LinearRegression()
In[4]:lin_mod.score(X_train, y_train ) Out[5]:0.4453601500506762
In[6]:lin_mod.score(X_test, y_test) Out[7]:0.46740661107915094
```

MODEL 3-DECISION TREE

```
In[1]:

from sklearn.tree import DecisionTreeRegressor

In[2]:

dTree=
DecisionTreeRegressor(criterion='mse',splitter='best',random_state=25,max_dep
th=5)


In[3]:

dTree.fit(X_train,y_train)

Out[3]:

DecisionTreeRegressor(max_depth=5, random_state=25)

In [4]:

print(dTree.score(X_train,y_train))

print(dTree.score(X_test,y_test))

0.6987590136971868
```

```
0.7490946656981097
```

#Trying with different "max_depth"

```
In [5]:

dTree=
DecisionTreeRegressor(criterion='mse',splitter='best',random_state=25,max_dep
th=14)

dTree.fit(X_train,y_train)

print(dTree.score(X_train,y_train))

print(dTree.score(X_test,y_test))

0.7320163141352926

0.7764637553626321
```

```
#No improvements in score after "max_depth = 14"

# Trying with different criteria

In [6]:

dTree=
DecisionTreeRegressor(criterion='mae',splitter='best',random_state=25,max_dep
th=20)

dTree.fit(X_train,y_train)

print(dTree.score(X_train,y_train))

print(dTree.score(X_test,y_test))

0.7152580836801676

0.7650663853334146
```

CONCLUSION AN(Phase 2):

Project Conclusion:

- In the Phase 2 conclusion,we will summarize the key findings and insight from the advanced regression techniques. The development of predictive models for estimating RSPM/PM10 levels based on SO2 and NO2 levels has proven successful. These models provide a valuable tool for forecasting air quality conditions and facilitating timely responses to mitigate pollution.