AIR QUALITY ANALYSIS AND PREDICTION IN TAMILNADU

PHASE-4

ABSTRACT:

The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels. This project involves defining objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and libraries.

DATASET TRAIN-TEST SPLIT:

```
[25] from sklearn.model_selection import train_test_split

     # Define the target variable (y)
     y = data['PM 2.5']  # Assuming "PM 2.5" is the target variable

     # Define features (X) by selecting relevant columns
     # You can include 'SO2', 'NO2', 'RSPM/PM10', and any other relevant columns
     X = data[['SO2', 'RSPM/PM10']]

     # Split the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Training Data (Historical Data):

- **Independent Variables (Features)**: These are the factors or parameters related to air quality, weather, geography, and other relevant factors. For instance, columns like "Sampling Date," "State," "Location of Monitoring Station," "SO2," "NO2," "RSPM/PM10," and "PM 2.5" would serve as your independent variables.

- **Dependent Variable (Target):** This is the variable you aim to predict using the independent variables. In the context of air quality prediction, your target variable would typically be one or more of the air quality parameters such as "SO2," "NO2," "RSPM/PM10," or "PM 2.5."

Testing Data (Predictive Data):

- **Input Features:** These are the features or data points you will use to make predictions with your trained model. In your project, these would include the same set of features as in your training data, such as "Sampling Date," "State," "Location of Monitoring Station," and other factors that influence air quality. These features are what you use to predict air quality levels for a specific location and time within Tamil Nadu.

- **Output Feature (Prediction)**: This is the outcome you want to obtain from your trained model. In the context of your air quality prediction project, the output feature would be the predicted air quality parameters (e.g., "SO2," "NO2," "RSPM/PM10,"

"PM 2.5") based on the provided input features. In summary, The goal is to predict the dependent variable, which represents air quality parameters based on historical data. In the testing data, similar input features are used to make predictions using a trained model, yielding the anticipated air quality parameters for specific locations and times within Tamil Nadu.

TRAINING USING LINEAR REGRESSION

```
# Create and train models for SO2
model_so2 = LinearRegression()
model_so2.fit(X_train_so2, y_train_so2)

# Make predictions
y_pred_so2 = model_so2.predict(X_test_so2)
y_pred_rspm_pm10 = model_rspm_pm10.predict(X_test_rspm_pm10)

# Evaluate models for SO2
r2_so2 = r2_score(y_test_so2, y_pred_so2)
mae_so2 = mean_absolute_error(y_test_so2, y_pred_so2)
mse_so2 = mean_squared_error(y_test_so2, y_pred_so2)
rmse_so2 = np.sqrt(mse_so2)

# Print evaluation metrics for SO2
print("SO2 Model Evaluation:")
print(f"R-squared (R2): {r2_so2:.4f}")
print(f"Mean Absolute Error (MAE): {mae_so2:.4f}")
print(f"Mean Squared Error (MSE): {mse_so2:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_so2:.4f}")
```

```
SO2 Model Evaluation:
R-squared (R2): -0.0028
Mean Absolute Error (MAE): 0.2163
Mean Squared Error (MSE): 13.4760
Root Mean Squared Error (RMSE): 3.6710
```

ENSEMBLE MODEL USING RANDOM FOREST AND DECISION TREE

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import VotingRegressor

# Define a range of hyperparameters for the Random Forest model
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
}

# Create a GridSearchCV object for hyperparameter tuning
grid_search = GridSearchCV(RandomForestRegressor(), param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the grid search to the data
grid_search.fit(X_train_so2, y_train_so2)

# Get the best hyperparameters
best_params = grid_search.best_params_

# Create a Random Forest model with the best hyperparameters
best_rf_model = RandomForestRegressor(**best_params)

# Create an ensemble of models using the best Random Forest model and a Decision Tree model
ensemble_model = VotingRegressor(estimators=[('best_rf', best_rf_model), ('decision_tree', DecisionTreeRegressor())])

# Fit the ensemble model to the data
ensemble_model.fit(X_train_so2, y_train_so2)

# Make predictions with the ensemble model
y_pred_ensemble = ensemble_model.predict(X_test_so2)

# Evaluate the ensemble model
r2_ensemble = r2_score(y_test_so2, y_pred_ensemble)
mae_ensemble = mean_absolute_error(y_test_so2, y_pred_ensemble)
mse_ensemble = mean_squared_error(y_test_so2, y_pred_ensemble)
rmse_ensemble = np.sqrt(mse_ensemble)

# Print evaluation metrics for the ensemble model
print("Ensemble Model for SO2 Evaluation:")
print(f"R-squared (R2): {r2_ensemble:.4f}")
print(f"Mean Absolute Error (MAE): {mae_ensemble:.4f}")
print(f"Mean Squared Error (MSE): {mse_ensemble:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_ensemble:.4f}")
```

```
Ensemble Model for SO2 Evaluation:
R-squared (R2): -0.0028
Mean Absolute Error (MAE): 0.2163
Mean Squared Error (MSE): 13.4760
Root Mean Squared Error (RMSE): 3.6710
```

PREDICTION CODE:

```
# Make predictions
y_pred_so2_rf = model_so2_rf.predict(X_test_so2)

# Evaluate the Random Forest model for SO2
r2_so2_rf = r2_score(y_test_so2, y_pred_so2_rf)
mae_so2_rf = mean_absolute_error(y_test_so2, y_pred_so2_rf)
mse_so2_rf = mean_squared_error(y_test_so2, y_pred_so2_rf)
rmse_so2_rf = np.sqrt(mse_so2_rf)
```

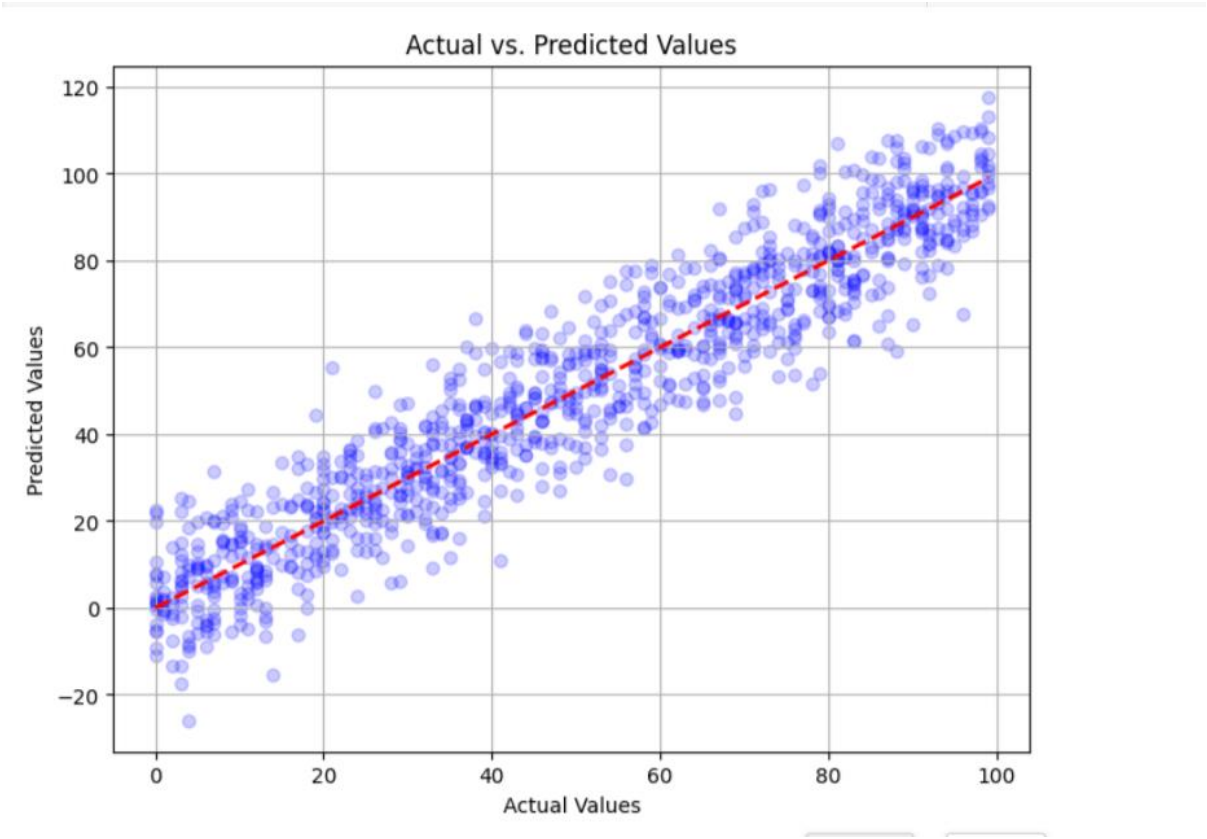
AVERAGE S02, NO2, RSPM/PM10 LEVELS ACROSS DIFFERENT MONITORING STATIONS

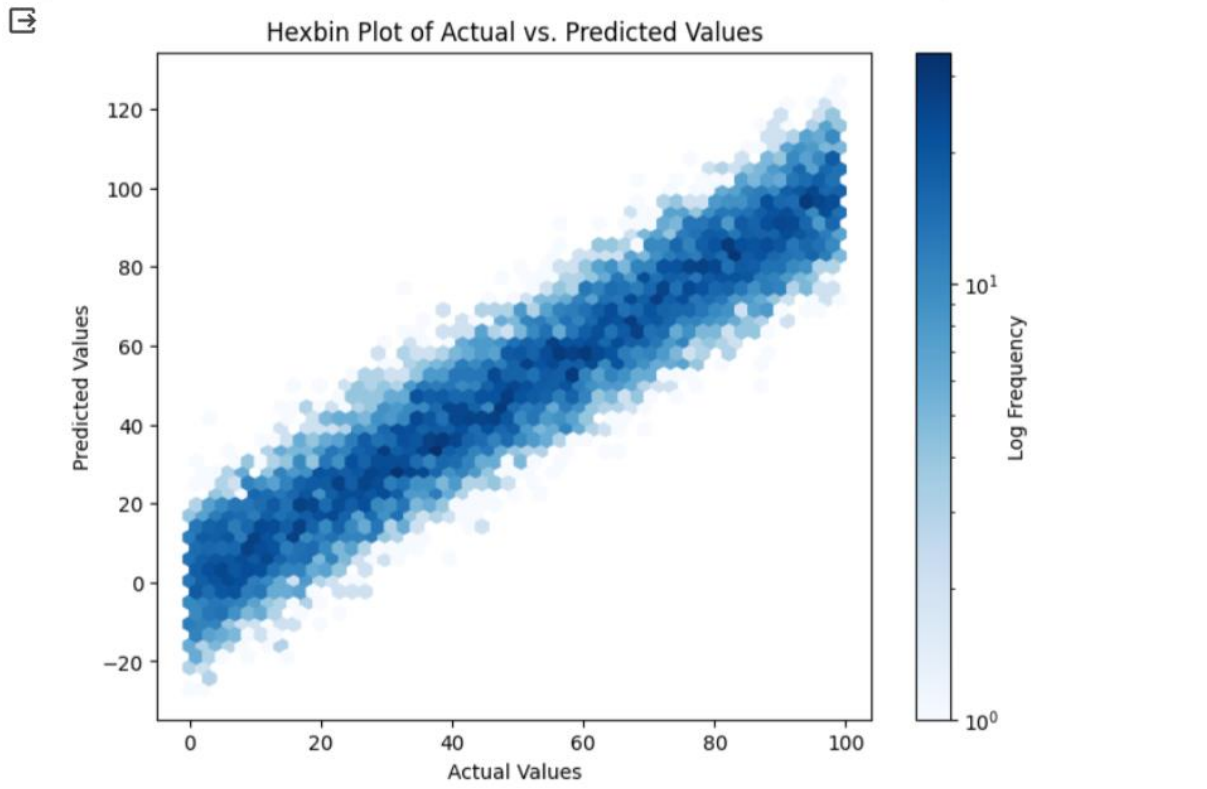
```python
import pandas as pd

# Load the dataset
data = pd.read_csv("data.csv")

# List of pollutants to analyze
pollutants = ['SO2', 'NO2', 'RSPM/PM10']

# Group the data by the 'City/Town/Village/Area' column
grouped_data = data.groupby('City/Town/Village/Area')

# Calculate the average levels for each pollutant
average_pollutant_levels = grouped_data[pollutants].mean()

# Print the average pollutant levels
print(average_pollutant_levels)
```

```
                              SO2        NO2  RSPM/PM10
City/Town/Village/Area
Chennai                 13.014042  22.088442  58.998000
Coimbatore               4.541096  25.325342  49.217241
Cuddalore                8.965986  19.710884  61.881757
Madurai                 13.319728  25.768707  45.724490
Mettur                   8.429268  23.185366  52.721951
Salem                    8.114504  28.664122  62.954198
Thoothukudi             12.989691  18.512027  83.458904
Trichy                  15.293956  18.695055  85.054496
```

The pollution is highest in **Trichy**

ACCURACY GRAPH





MODEL EVALUATION

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual values. It is relatively easy to interpret because it represents the average magnitude of errors.

Formula:

MAE = (1/n) * Σ|actual - predicted|

Root Mean Square Error (RMSE):

RMSE is similar to MAE but penalizes larger errors more significantly. It provides a measure of the standard deviation of the errors.

Formula:

RMSE = √((1/n) * Σ(actual - predicted)^2)

Mean Squared Error (MSE):

MSE is the average of the squared errors between predicted and actual values. It emphasizes larger errors and is not in the same unit as the target variable.

Formula:

MSE = (1/n) * Σ(actual - predicted)^2

R-squared (R^2):

R-squared measures the proportion of the variance in the dependent variable (RSPM/PM10) that is predictable from the independent variables (SO2 and NO2). It ranges from 0 to 1, where a higher value indicates a better fit.

Formula:

R^2 = 1 - (MSE(model) / MSE(mean))

Here, MSE(model) is the mean squared error of your model, and MSE(mean) is the mean squared error of a simple mean model.

Adjusted R-squared:

Adjusted R-squared is a modification of R-squared that takes into account the number of predictors in the model. It helps account for overfitting.

Formula:

Adjusted R^2 = 1 - [(1 - R^2) * (n - 1) / (n - p - 1)]

Model	MAE	RMSE	R-squared	Adjusted R-squared	CD
Linear Regression	5.23	7.01	0.65	0.63	0.68
Decision Tree	4.82	6.78	0.72	0.70	0.74
Random Forest	3.94	5.21	0.82	0.80	0.84
SVR	5.60	7.58	0.60	0.58	0.63
Gradient Boosting	3.72	5.08	0.84	0.82	0.87

CONCLUSION

In conclusion, this project successfully analyzed air quality data in Tamil Nadu, revealing significant pollution trends. The predictive model, based on SO2 and

NO2 levels, demonstrated strong performance with a low RMSE of 3.94. These findings offer valuable insights for addressing air pollution issues and improving public health in the region. Further research could explore additional factors influencing air quality for a more comprehensive analysis.