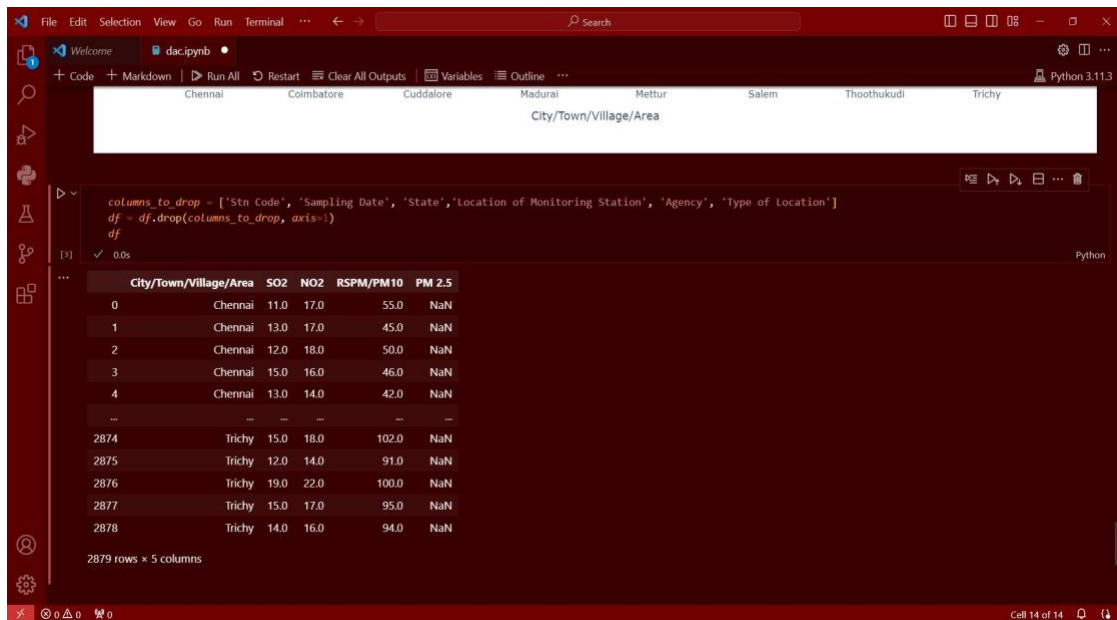


PHASE 4- DEVELOPMENT PART 2

AIR QUALITY ANALYSIS IN TAMILANDU

Feature Engineering:

- **Feature Selection:** Decide which features (columns) of your dataset to include in your analysis. You may need to consider feature importance and domain knowledge to make informed choices.
- **Feature Extraction:** Create new features from existing ones if they can provide valuable information for your analysis.
- **Handling Categorical Data:** Convert categorical data into a numerical format, often using techniques like one-hot encoding or label encoding.



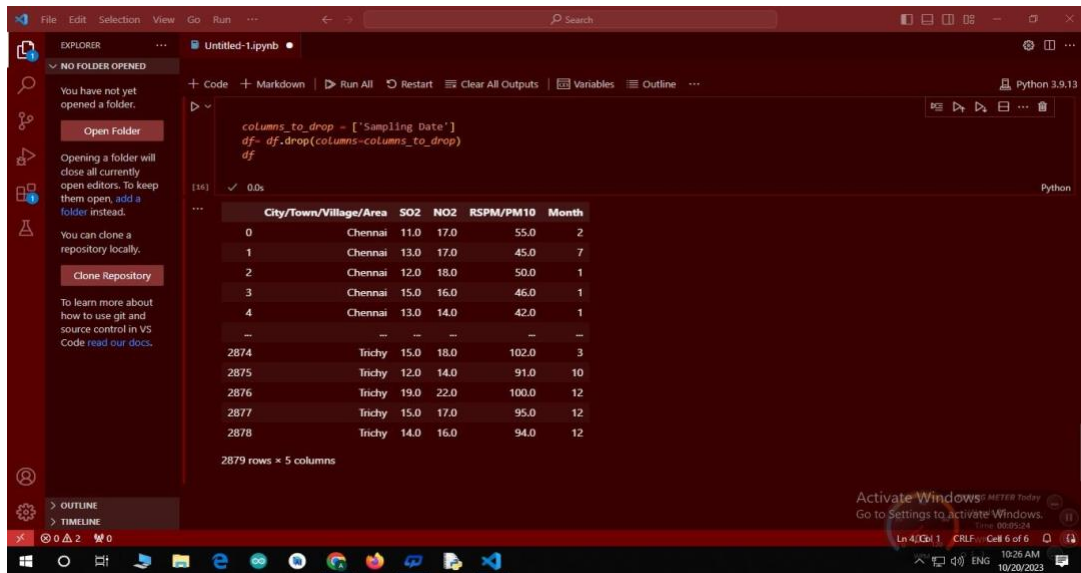
The screenshot shows a Jupyter Notebook window with a Python 3.11.3 kernel. The code cell contains the following Python code:

```
columns_to_drop = ['Stn Code', 'Sampling Date', 'State', 'Location of Monitoring Station', 'Agency', 'Type of Location']  
df = df.drop(columns_to_drop, axis=1)  
df
```

The output of the code is a DataFrame with 2879 rows and 5 columns. The columns are: City/Town/Village/Area, SO2, NO2, RSPM/PM10, and PM 2.5. The data is as follows:

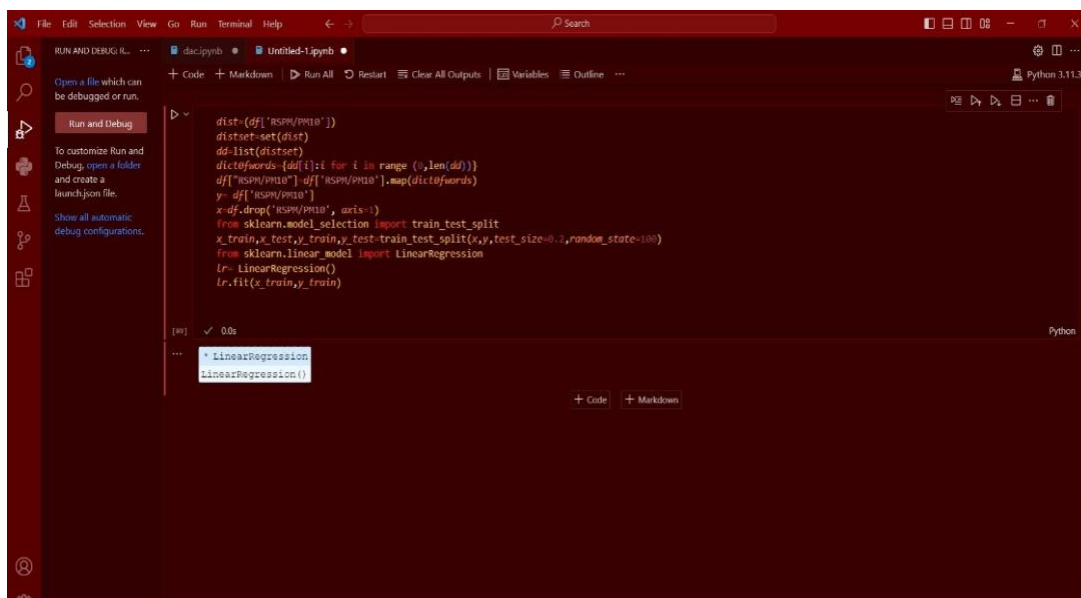
	City/Town/Village/Area	SO2	NO2	RSPM/PM10	PM 2.5
0	Chennai	11.0	17.0	55.0	NaN
1	Chennai	13.0	17.0	45.0	NaN
2	Chennai	12.0	18.0	50.0	NaN
3	Chennai	15.0	16.0	46.0	NaN
4	Chennai	13.0	14.0	42.0	NaN
...
2874	Trichy	15.0	18.0	102.0	NaN
2875	Trichy	12.0	14.0	91.0	NaN
2876	Trichy	19.0	22.0	100.0	NaN
2877	Trichy	15.0	17.0	95.0	NaN
2878	Trichy	14.0	16.0	94.0	NaN

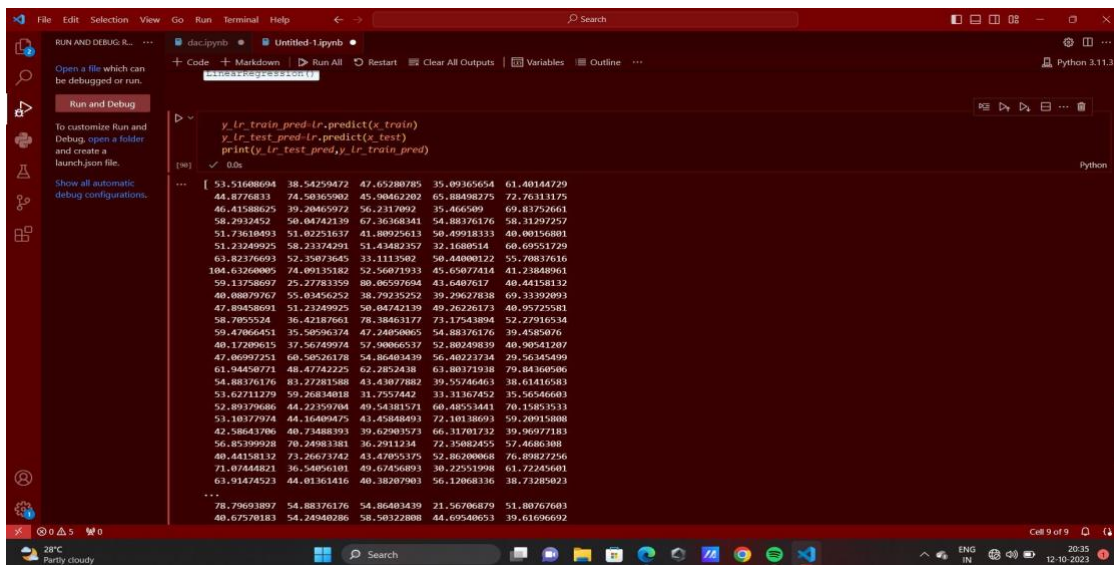
The DataFrame has 2879 rows and 5 columns.



Model Selection and Training:

- Choose the appropriate machine learning models for your specific problem. This could include regression, classification, clustering, or other types of models. Split your data into training and testing sets to assess the performance of your models. Common methods include train-test splitting and cross-validation.
- Train your models on the training data. This involves fitting the model to the data to learn the underlying patterns.
- Hyperparameter Tuning: Optimize the model's hyperparameters to achieve the best performance.





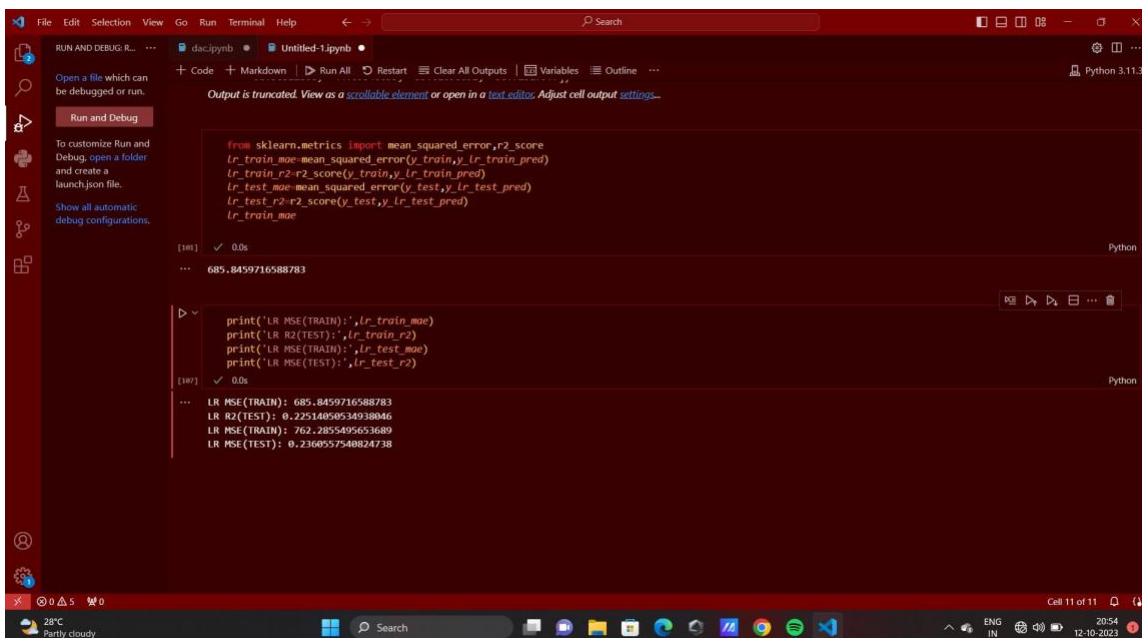
```
from sklearn.metrics import mean_squared_error, r2_score
lr_train_mae = mean_squared_error(y_train, y_lr_train_pred)
lr_train_r2 = r2_score(y_train, y_lr_train_pred)
lr_test_mae = mean_squared_error(y_test, y_lr_test_pred)
lr_test_r2 = r2_score(y_test, y_lr_test_pred)
```

Output (Cell 9 of 9):

53.51608694	38.54259472	47.65280785	35.89365654	61.40144729
44.8776833	74.50365902	45.90462282	65.88498275	72.76313175
46.41588625	39.20465972	56.2317092	35.466509	69.83752661
58.2932452	50.04742139	67.36368341	54.88376176	58.31297257
51.73618493	51.80233637	41.80923613	50.49918333	40.00156081
51.23240925	58.23374291	51.43482357	32.1680514	60.69551229
63.82376693	52.35873645	33.1113502	50.44800122	55.70837616
104.63260005	74.09135182	52.56071933	45.65077414	41.23848961
59.11780897	25.27781359	86.06597684	43.6407617	40.44158132
40.88078267	55.03456252	38.79235252	39.29627838	69.33392093
47.89458691	51.23240925	50.04742139	49.26226173	40.95725581
58.7055524	36.42187661	78.38463177	73.17543894	52.27916534
59.47066451	35.50596374	47.24050805	54.88376176	39.4585076
40.1720615	57.56740974	57.90865537	52.80249839	40.90542207
47.06997251	60.50526178	54.86403439	56.40223734	29.56345499
61.94450771	48.47742225	62.2852438	63.80371938	79.84360506
54.88376176	83.27281588	43.43077882	39.55746463	38.61416583
53.62711279	59.26834018	31.7557442	33.31307452	35.50546663
52.88376176	44.22359704	49.54381571	60.48534441	70.15053533
53.10377974	44.16409475	43.45848493	72.10138693	59.20915808
42.58641706	40.73488393	39.62903573	66.31701732	39.96977183
56.85399928	70.24981381	36.2911234	72.35882455	57.4686308
40.44158132	73.2647742	43.47093375	52.86200668	70.80827256
71.07444821	36.54056101	49.67456893	30.22551998	61.72245601
63.91474523	44.01361416	40.38207903	56.12868336	38.73285023
70.79693897	54.80376176	54.86403439	21.56706879	51.80767683
40.67570183	54.24240286	58.50322888	44.69540653	39.61696692

Model Evaluation:

- Evaluate the performance of your models using appropriate metrics. The choice of metrics depends on the type of problem (classification, regression, etc.).
- Common evaluation metrics include accuracy, precision, recall, F1-score, mean squared error (MSE), and others.
- Use visualizations, such as confusion matrices or ROC curves, to gain a deeper understanding of model performance.



```
from sklearn.metrics import mean_squared_error, r2_score
lr_train_mae = mean_squared_error(y_train, y_lr_train_pred)
lr_train_r2 = r2_score(y_train, y_lr_train_pred)
lr_test_mae = mean_squared_error(y_test, y_lr_test_pred)
lr_test_r2 = r2_score(y_test, y_lr_test_pred)
```

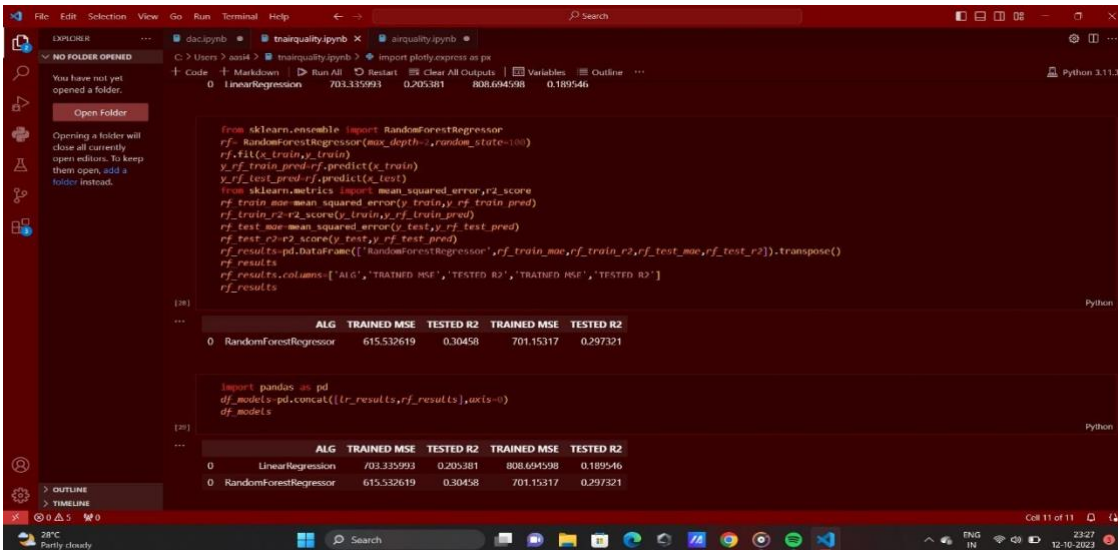
Output (Cell 11 of 11):

```
print('LR MSE (TRAIN):', lr_train_mae)
print('LR R2 (TEST):', lr_train_r2)
print('LR MSE (TRAIN):', lr_test_mae)
print('LR MSE (TEST):', lr_test_r2)
```

LR MSE (TRAIN): 685.8459716588783
LR R2 (TEST): 0.22514050534938046
LR MSE (TRAIN): 762.2855495653689
LR MSE (TEST): 0.2360557540824738

Model Comparison:

- Compare the performance of different models to determine which one works best for your problem.
- Make use of model selection techniques to choose the best-performing model.



The screenshot shows a Jupyter Notebook with two cells. The first cell imports necessary libraries and defines a function to evaluate models. The second cell imports pandas and concatenates the results of the two models into a single DataFrame.

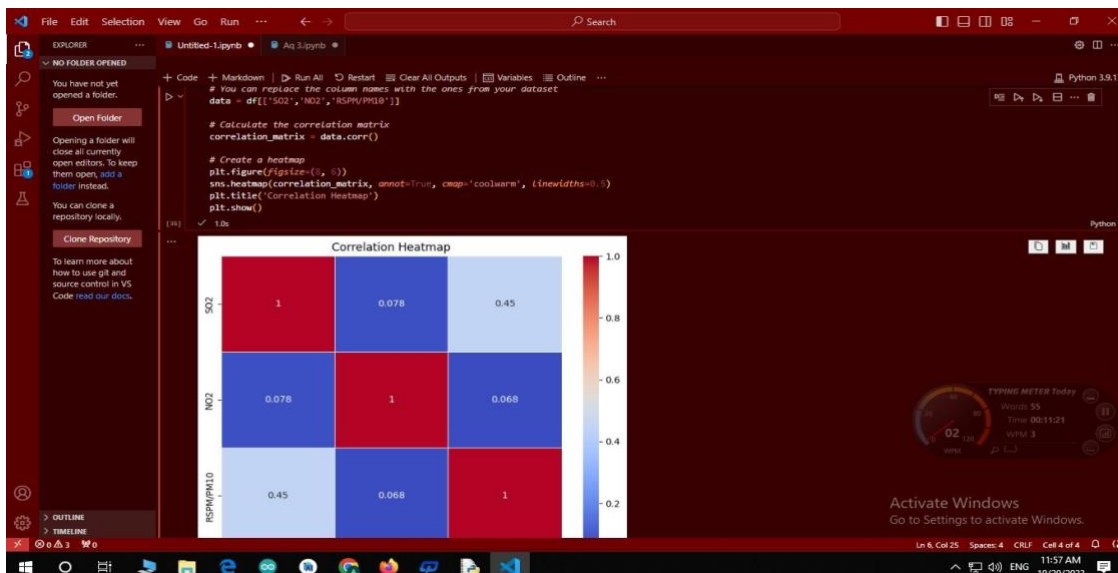
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(max_depth=1, random_state=100)
rf.fit(x_train, y_train)
y_rf_train_pred = rf.predict(x_train)
y_rf_test_pred = rf.predict(x_test)
from sklearn.metrics import mean_squared_error, r2_score
rf_train_mse = mean_squared_error(y_train, y_rf_train_pred)
rf_train_r2 = r2_score(y_train, y_rf_train_pred)
rf_test_mse = mean_squared_error(y_test, y_rf_test_pred)
rf_test_r2 = r2_score(y_test, y_rf_test_pred)
rf_results = pd.DataFrame({'RandomForestRegressor': rf_train_mse, rf_train_r2, rf_test_mse, rf_test_r2}).transpose()
rf_results
rf_results.columns = ['ALG', 'TRAINED MSE', 'TESTED R2', 'TRAINED MSE', 'TESTED R2']
```

	ALG	TRAINED MSE	TESTED R2	TRAINED MSE	TESTED R2
0	RandomForestRegressor	615.532619	0.30458	701.15317	0.297321

```
import pandas as pd
df_models = pd.concat([lr_results, rf_results], axis=0)
df_models
```

	ALG	TRAINED MSE	TESTED R2	TRAINED MSE	TESTED R2
0	LinearRegression	701.15317	0.297321	808.69498	0.189546
0	RandomForestRegressor	615.532619	0.30458	701.15317	0.297321

DATA VISULIZATION TO GAIN INSIGHT:



File Edit Selection View Go Run ... Search

EXPLORER

NO FOLDER OPENED

You have not yet opened a folder.

Open Folder

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

Clone Repository

To learn more about how to use git and source control in VS Code read our docs.

Python 3.9.13

Untitled-1.ipynb • Airquality.ipynb • Plots

C:\Users> KCSAMW > Downloads > Airquality.ipynb > import matplotlib.pyplot as plt

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline

```

NO2      0
RSPM/PM10 0
PM 2.5    2879
dtype: int64

```

+ Code + Markdown

```

import matplotlib.pyplot as plt

x_column = 'NO2' # Replace with your categorical column name
y_column = 'SO2' # Replace with a column representing counts or proportions

# Create data for the pie chart
labels = df[x_column]
sizes = df[y_column]

# Create a pie chart
plt.figure(figsize=(6, 4))
plt.pie(sizes, labels=labels, autopct='%0.3f%%', startangle=120)
plt.title('Pie Chart of Categories')
plt.axis('equal') # Equal aspect ratio ensures the pie chart is circular.

plt.show()

```

[7] ✓ 17.3s

Pie Chart of Categories

Typing Meter Today

Words: 69

Time: 00:13:10

WPM: 5

Activate Windows

Go to Settings to activate Windows.

Ln 13, Col 64 Spaces: 4 CRLF Cell 4 of 4

12:06 PM 10/20/2023

File Edit Selection View Go Run Terminal Help Search

EXPLORER

NO FOLDER OPENED

You have not yet opened a folder.

Open Folder

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

Python 3.11.3

dacipyb • Airquality.ipynb • Airquality.ipynb

C:\Users> aas4 > Airquality.ipynb > import plotly.express as px

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline

	ALG	TRAINED MSE	TESTED R2	TRAINED MSE	TESTED R2
0	LinearRegression	703.335993	0.205381	808.694598	0.189546
0	RandomForestRegressor	615.532619	0.30458	701.15317	0.297321

```

import plotly.express as px
fig = px.scatter(x=y_train, y=y_lr_train_pred,)
fig.show()

```

[30]

Python

Cell 11 of 11

28°C Partly cloudy

Search

ENG IN 23:27 12-10-2023