Phase 4 – Project Execution.

Prepared by:

Pedro Julio García Amariles

José Fernelis Torres

Carlos Alberto Ramírez Triana

Universidad Nacional Abierta Y A Distancia – UNAD

School of Basic Sciences, Technology and Engineering – ECBTI

Tutor:

Diego Enrique Guzman Villamarin

Specialization in Next Generation Networks

Data Analytics (203018207_2)

November 2025

Abstract

This project presents a complete data analytics workflow applied to an IoT-based air quality dataset. The study integrates the methodological steps established in Phases 1–4 of the course, including the formulation of the research problem, literature review, dataset selection, exploratory data analysis (EDA), and predictive modeling. Using the city_day.csv dataset, a comprehensive EDA was conducted to examine pollutant behavior, temporal trends, and inter-variable relationships. A linear regression model was developed to predict PM2.5 concentrations using multiple atmospheric indicators such as PM10, NO_2, NOx, NH_3, CO, SO_2, and O_3. The model achieved strong performance with an R^2 value of 0.80, demonstrating its effectiveness for IoT-based environmental prediction. The results highlight the importance of data-driven approaches for understanding pollution dynamics and support the use of machine learning techniques to enhance decision-making processes in air quality management.

Content

Introduction

The rapid growth of the Internet of Things (IoT) has generated large volumes of real-time sensor data, creating new opportunities for advanced analytics in environmental monitoring. Air quality assessment, in particular, benefits significantly from IoT-based data collection systems that continuously capture pollutant concentrations across urban areas. However, transforming this raw sensor data into meaningful insights requires the integration of data preprocessing techniques, exploratory data analysis (EDA), and machine learning models capable of identifying trends and predicting pollutant levels.

This project applies a complete data analytics workflow to the Air Quality Data in India (2015–2020) dataset, which contains daily measurements of key atmospheric pollutants such as PM2.5, PM10, NO_2, CO, SO_2, NH_3, and O_3. The study follows the methodological structure established in the earlier phases of the course, beginning with a bibliographic review, followed by dataset selection, data preparation, EDA, and the development of a regression model to estimate PM2.5 concentrations.

The objective of this project is to demonstrate how IoT-based datasets can be processed and modeled using Python and machine learning techniques to support environmental decision-making. By analyzing pollutant behavior and developing a predictive model with strong performance, the project highlights the value of data-driven approaches for understanding air quality dynamics in modern urban environments.

Development

The development of this project follows the five methodological steps established in Phase 4. Each step describes the analytical procedures applied to the selected IoT dataset, integrating bibliographic support, data preparation, exploratory data analysis, and predictive modeling. The following sections present the detailed execution of each step and their contribution to the final analytical solution.

1. Step 1. Bibliographic Review on IoT Data Analytics

The development of data analytics algorithms for IoT systems requires a solid theoretical foundation that integrates concepts from machine learning, data privacy, uncertainty analysis, and sensor-based data processing. According to Swamynathan (2017), machine learning provides the essential tools for predictive and descriptive modeling by enabling algorithms to learn patterns from large datasets. These capabilities are crucial in IoT environments, where continuous streams of sensor data must be processed efficiently.

From a data protection perspective, Torra (2017) highlights that IoT ecosystems generate sensitive and high-dimensional information, making it necessary to implement privacy-preserving mechanisms such as anonymization and controlled data access. These practices ensure ethical and secure handling of information during analytical and modelling processes.

Additionally, IoT data is often affected by instrumental and statistical uncertainties. Raghavender (2019) explains that sensor measurements may include noise, fluctuations, or calibration errors that propagate through analytical pipelines. Understanding these uncertainties is essential to guarantee reliable preprocessing and model performance.

Finally, IoT systems rely on interconnected devices capable of capturing, transmitting, and processing environmental or operational variables. Alonso Villegas and Torres Payoma (2021) emphasize that IoT infrastructures provide real-time data that support decision-making in areas such as environment, mobility, and industry, making them an ideal scenario for applying artificial intelligence and analytics.

Together, these theoretical perspectives provide the foundation for developing the data analysis algorithm proposed in this project, which aims to process and model IoT sensor data using Python and machine learning techniques.

2. Step 2. Dataset Selection and Description.

For the development of the proposed IoT data analytics algorithm, a public dataset was selected from Kaggle titled Air Quality Data in India (2015–2020). This dataset contains environmental measurements collected from multiple air quality monitoring stations distributed across major cities during a five-year period. Each station continuously recorded atmospheric variables using IoT-based sensing technologies, making the dataset highly suitable for the scope of this project.

The dataset includes daily aggregated values for key pollutants such as PM2.5, PM10, NO_2, SO_2, CO, and O_3, as well as meteorological indicators like temperature, humidity, and wind speed. These variables enable descriptive and predictive modeling tasks, supporting the development of machine learning algorithms for environmental monitoring applications.

The dataset consists of several files; however, the file city_day.csv was selected as the primary source for analysis due to its structured format, clear temporal organization, and comprehensive coverage of pollutant indicators. The data spans from 2015 to 2020, allowing for meaningful trend analysis and correlation studies.

This dataset aligns with the project objectives established in Phases 2 and 3, as it provides real-world IoT sensor data suitable for preprocessing, exploratory data analysis (EDA), feature engineering, and the implementation of predictive models. Its public availability, large volume of observations, and relevance to IoT-based environmental analytics make it an ideal choice for the development of the algorithm required in this phase.

3. Step 3. Data Preparation and Preprocessing.

The dataset city_day.csv was prepared and cleaned to ensure that the information used for analysis and modeling was accurate, consistent, and suitable for machine learning techniques. Data preprocessing is a fundamental stage in IoT analytics because sensor-generated data often contains missing values, noise, and inconsistencies produced by environmental conditions, device limitations, or transmission issues.

First, the dataset was imported into the analysis environment using Python and libraries such as Pandas and NumPy. An initial inspection of the structure, column names, and data types was performed to identify potential inconsistencies. Several variables, including pollutant indicators and meteorological attributes, contained missing or null values. These entries were handled using appropriate techniques such as removal of incomplete records or imputation based on statistical measures like the mean or median.

Additionally, the datetime column was converted into a proper temporal format to facilitate trend analysis and time-based visualizations. Duplicated rows, if present, were removed to avoid bias in the analysis. Data types were standardized to ensure compatibility with analytical and machine learning algorithms.

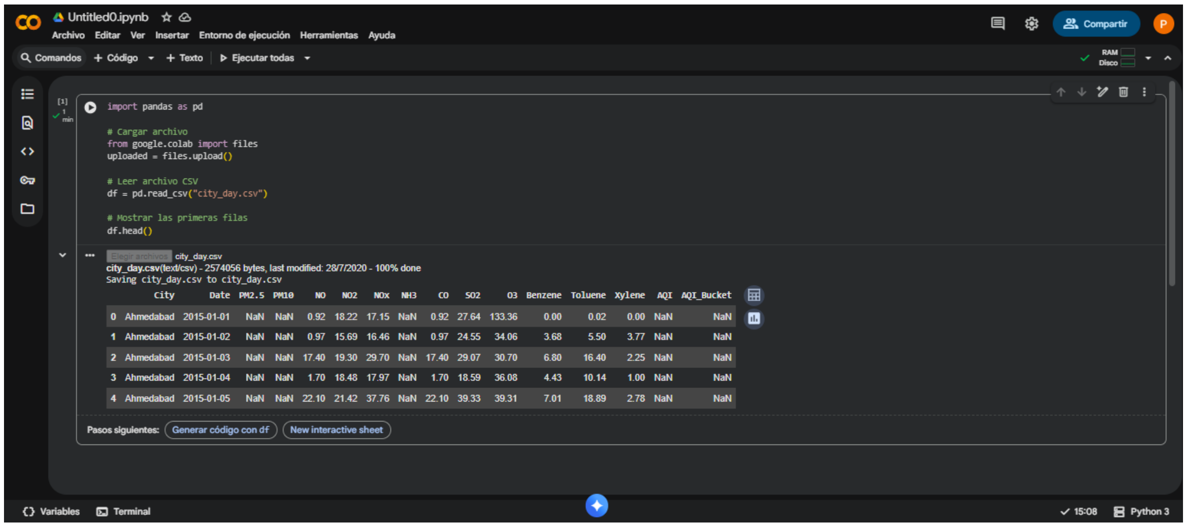
Outlier detection methods were also applied, especially for pollutant concentrations such as PM2.5 and PM10, which tend to fluctuate due to extreme weather conditions or measurement errors. Identifying and handling these anomalies improves the reliability of the resulting statistical and predictive models.

The preprocessing phase ensures that the dataset used in subsequent steps is clean, consistent, and ready for exploratory analysis and model development. This aligns with the methodological framework proposed in Phase 3, where data preparation is described as a crucial stage in the design of IoT-based analytics solutions

4. Step 4 Exploratory Data Analysis (EDA)

The exploratory data analysis (EDA) process was conducted to examine the structure, completeness, and statistical behavior of the variables included in the city_day.csv dataset. This phase enables the identification of missing values, temporal inconsistencies, variability patterns, and important relationships among atmospheric pollutants and meteorological indicators. The visual inspection and statistical summaries derived from the dataset guide the subsequent data cleaning and modeling stages.

Figure 1Initial Display of the Dataset Using df.head()


Untitled0.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto ▷ Ejecutar todas
Compartir
Variables Terminal
15:08 Python 3

```
import pandas as pd

# Cargar archivo
from google.colab import files
uploaded = files.upload()

# Leer archivo CSV
df = pd.read_csv("city_day.csv")

# Mostrar las primeras filas
df.head()
```

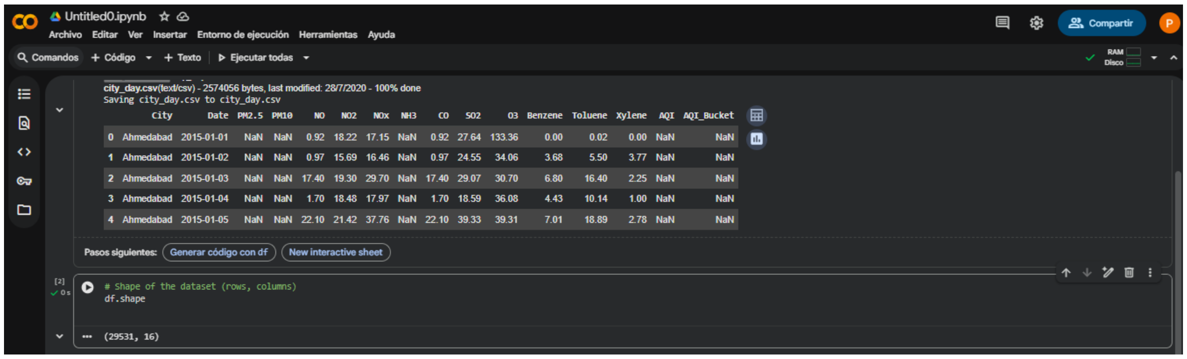
```
Elegir archivos city_day.csv
city_day.csv(text/csv) - 2574056 bytes, last modified: 28/7/2020 - 100% done
Saving city_day.csv to city_day.csv
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00	NaN	NaN
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	2.78	NaN	NaN

Pasos siguientes: Generar código con df New interactive sheet

Note: The first rows of the dataset are displayed using the df.head() function, which allows an initial inspection of the main variables collected by IoT-based environmental sensors. This preview shows pollutant indicators such as PM2.5, PM10, NO_2, NH_3, CO, SO_2, and O_3, along with meteorological and categorical features. This step confirms that the dataset was loaded correctly and that column names and formats align with expectations for subsequent analysis.

Figure 2 Dataset Dimensions Using df.shape


Untitled0.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto ▷ Ejecutar todas
Compartir

```
city_day.csv(text/csv) - 2574056 bytes, last modified: 28/7/2020 - 100% done
Saving city_day.csv to city_day.csv
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00	NaN	NaN
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	2.78	NaN	NaN

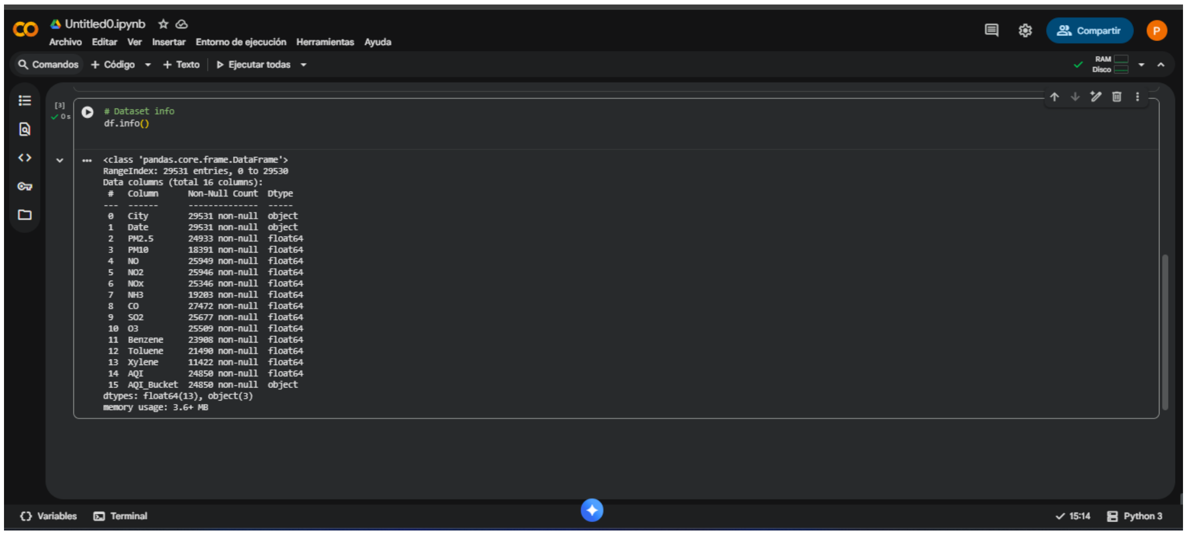
Pasos siguientes: Generar código con df New interactive sheet

```
# Shape of the dataset (rows, columns)
df.shape
```

```
(29531, 16)
```

Note: The output of the df.shape function shows that the dataset contains 29531 rows and 16 columns, indicating a substantial volume of observations collected from IoT-based environmental sensors. Understanding the dataset size is essential for planning the data cleaning procedures and determining the scope of upcoming analytical and machine learning tasks.

Figure 3 Dataset Structure and Data Types Using df.info()

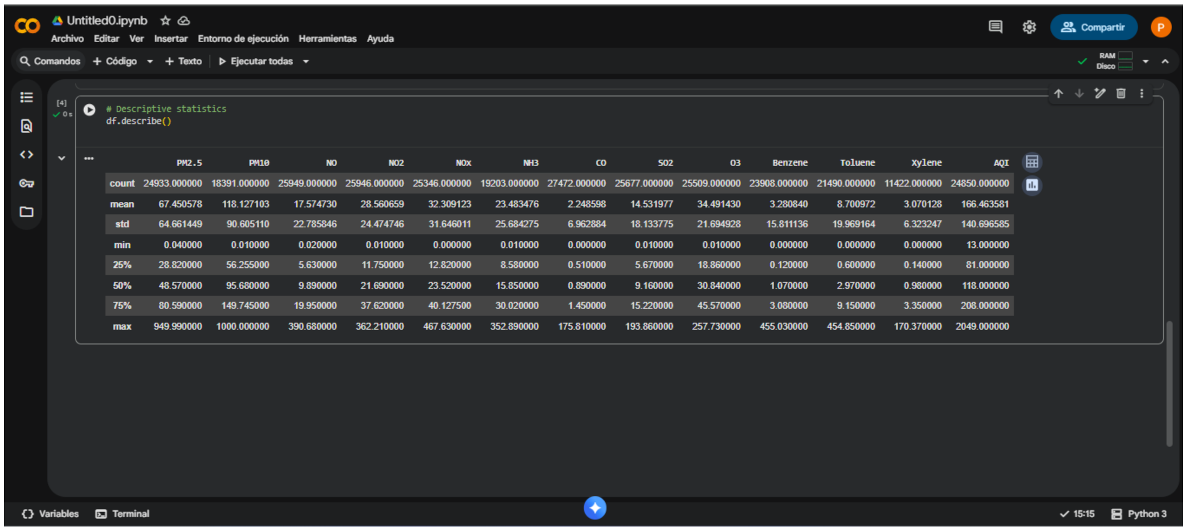

Untitled0.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todas
Compartir
RAM Disco
Variables Terminal
15:14 Python 3

```
# Dataset info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        29531 non-null  object 
 1   Date        29531 non-null  object 
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object 
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

Note: This figure displays the dataset structure obtained using the df.info() function. It shows the data types, non-null counts, and memory usage for each column. This information is essential for identifying missing values, distinguishing between numerical and categorical variables, and detecting potential formatting issues. Several pollutant-related columns (such as PM2.5, PM10, NO_2, CO, and SO_2) contain missing values, indicating the need for data cleaning procedures prior to model development.

Figure 4 Descriptive Statistics for Numerical Variables Using df.describe()

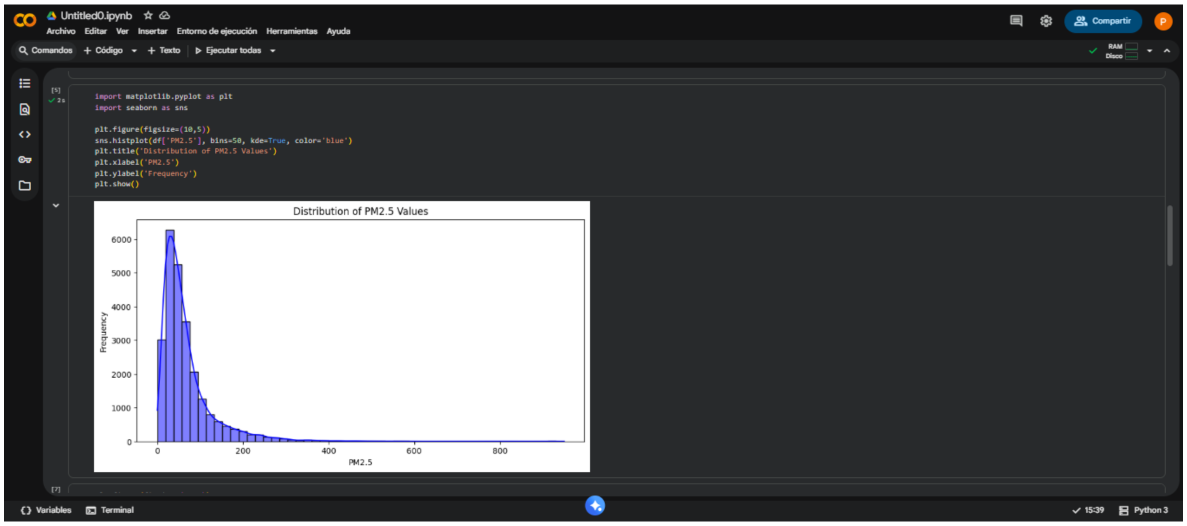


```
# Descriptive statistics
df.describe()
```

```
            PM2.5         PM10           NO          NO2          NOx          NH3           CO          SO2           O3      Benzene      Toluene       Xylene          AQI
count 24933.000000 18391.000000 25949.000000 25946.000000 25346.000000 19203.000000 27472.000000 25677.000000 25509.000000 23908.000000 21490.000000 11422.000000 24850.000000
mean     67.450578   118.127103    17.574730    28.560659    32.309123    23.483476     2.248598    14.531977    34.491430     3.280840     8.700972     3.070128   166.463581
std      64.661449    90.605110    22.785846    24.474746    31.646011    25.684275     6.962884    18.133775    21.694928    15.811136    19.969164     6.323247   140.696585
min       0.040000     0.010000     0.020000     0.010000     0.000000     0.010000     0.000000     0.010000     0.010000     0.000000     0.000000     0.000000    13.000000
25%      28.820000    56.255000     5.630000    11.750000    12.820000     8.580000     0.510000     5.670000    18.860000     0.120000     0.600000     0.140000    81.000000
50%      48.570000    95.680000     9.890000    21.690000    23.520000    15.850000     0.890000     9.160000    30.840000     1.070000     2.970000     0.980000   118.000000
75%      80.590000   149.745000    19.950000    37.620000    40.127500    30.020000     1.450000    15.220000    45.570000     3.080000     9.150000     3.350000   208.000000
max     949.990000  1000.000000   390.680000   362.210000   467.630000   352.890000   175.810000   193.860000   257.730000   455.030000   454.850000   170.370000  2049.000000
```

Note: This figure displays the descriptive statistics for numerical variables in the dataset, computed using the df.describe() function. Metrics such as mean, standard deviation, minimum, maximum, and quartiles are included. These results provide insight into the variability of atmospheric pollutants (e.g., PM2.5, PM10, NO_2, CO, SO_2, O_3) and help identify potential outliers caused by environmental fluctuations or sensor measurement issues. This information is essential for understanding general variable behavior and guiding subsequent cleaning and modeling procedures.

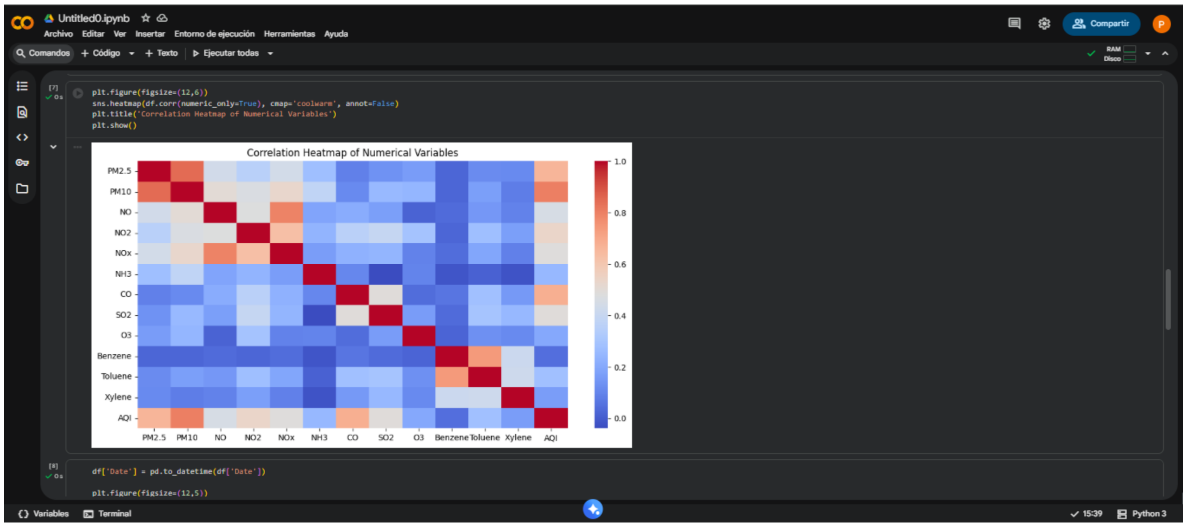
Figure 5 Histogram of PM2.5 Values



```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,5))
sns.histplot(df['PM2.5'], bins=50, kde=True, color='blue')
plt.title('Distribution of PM2.5 Values')
plt.xlabel('PM2.5')
plt.ylabel('Frequency')
plt.show()
```

Note: This figure illustrates the distribution of PM2.5 concentrations recorded by IoT-based environmental sensors. The distribution is strongly right-skewed, with most values concentrated in the lower range (0–100) and a long tail representing extreme pollution events. This behavior indicates high variability in pollutant levels and suggests the presence of outliers associated with environmental fluctuations or atypical measurement conditions. This visualization provides an essential understanding of pollutant dynamics prior to model development.

Figure 6 Correlation Heatmap of Numerical Variables

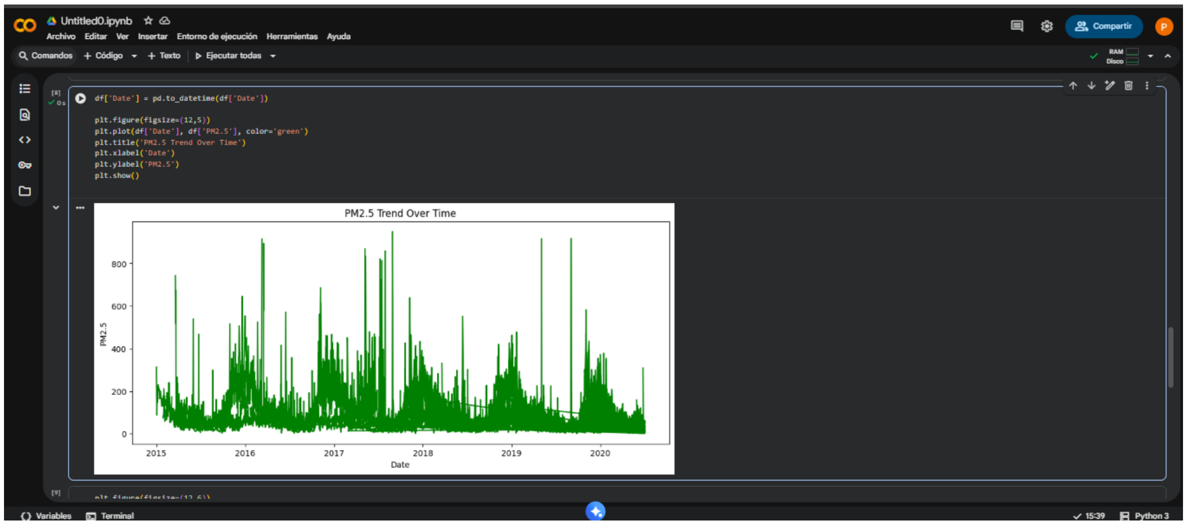

Untitled0.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todas
Compartir
Correlation Heatmap of Numerical Variables
PM2.5
PM10
NO
NO2
NOx
NH3
CO
SO2
O3
Benzene
Toluene
Xylene
AQI
1.0
0.8
0.6
0.4
0.2
0.0
Variables Terminal
15:39 Python 3

```
[7]
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(numeric_only=True), cmap='coolwarm', annot=False)
plt.title('Correlation Heatmap of Numerical Variables')
plt.show()
```

```
[8]
df['Date'] = pd.to_datetime(df['Date'])

plt.figure(figsize=(12,5))
```

Note: This figure presents a heatmap showing the correlation levels between the numerical variables in the dataset. A strong positive correlation is observed between PM2.5 and PM10, indicating that both pollutants tend to vary together under similar environmental conditions. Moderate correlations are also visible between AQI and several pollutant indicators. This visualization helps to understand inter-variable relationships and supports feature selection for the modeling phase.

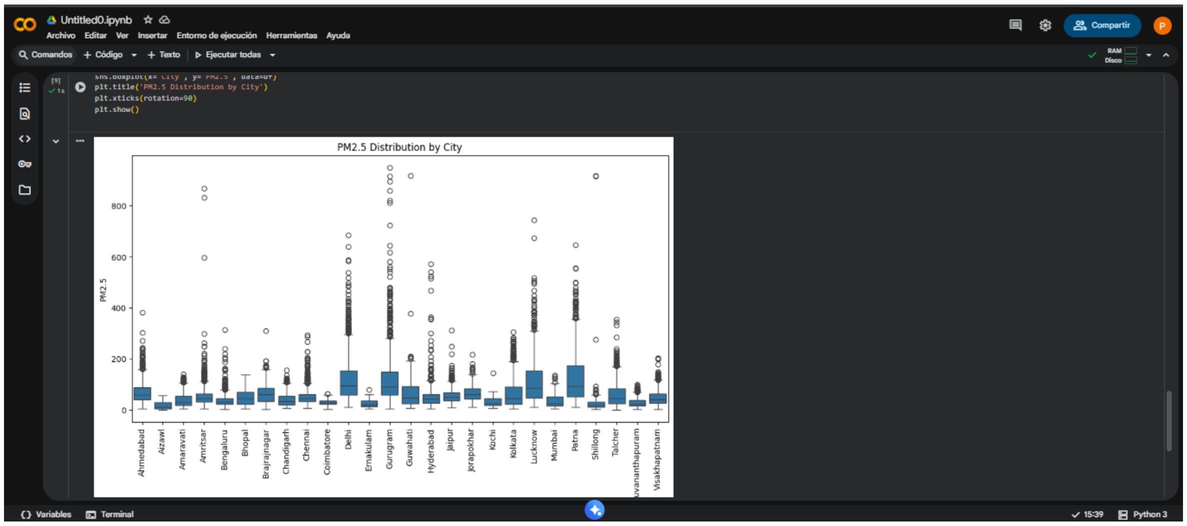
Figure 7 PM2.5 Trend Over Time



```
df['Date'] = pd.to_datetime(df['Date'])

plt.figure(figsize=(12,5))
plt.plot(df['Date'], df['PM2.5'], color='green')
plt.title('PM2.5 Trend Over Time')
plt.xlabel('Date')
plt.ylabel('PM2.5')
plt.show()
```

Note: This figure illustrates the temporal trend of PM2.5 values from 2015 to 2020. Several sharp peaks indicate high-pollution episodes, while continuous fluctuations reflect typical atmospheric variability captured by IoT-based environmental sensors. The graph reveals potential seasonal patterns, long-term variability, and anomalies that must be considered during the modeling stage.

Figure 8 PM2.5 Distribution by City Using Boxplots

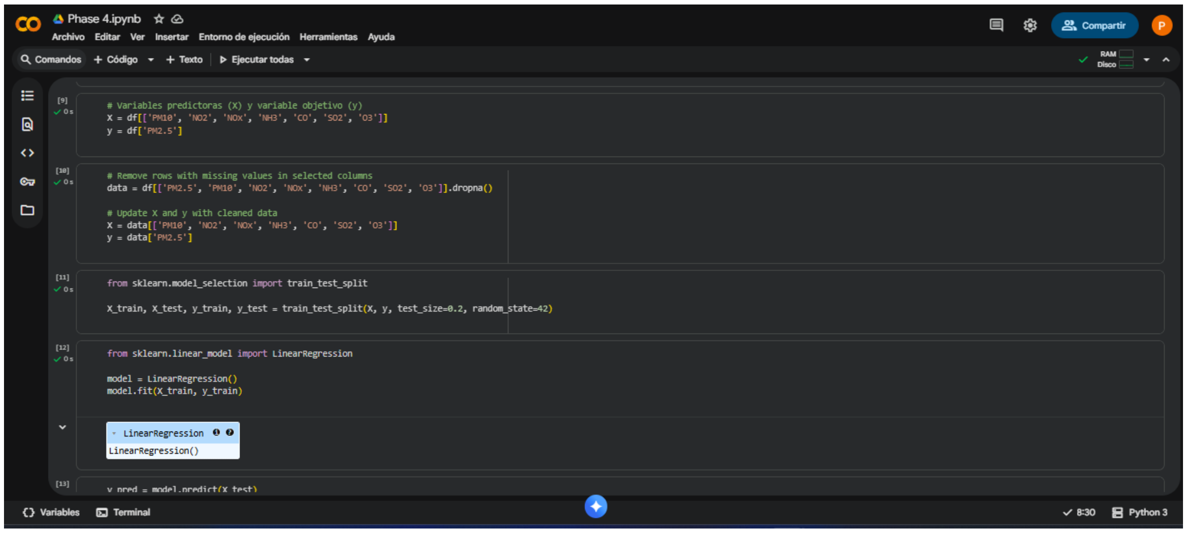


```
sns.boxplot(x='City', y='PM2.5', data=df)
plt.title('PM2.5 Distribution by City')
plt.xticks(rotation=90)
plt.show()
```

Note: This figure displays the distribution of PM2.5 values for each city in the dataset. Each boxplot represents the median, quartiles, and outliers captured by IoT-based environmental sensors. Several cities exhibit a wide spread of PM2.5 values, indicating frequent high-pollution events. This visualization helps identify urban areas with higher variability and recurring extreme pollutant concentrations.

5. Step 5: Model Development (Regression for PM2.5 Prediction)

Figure 9 Model Preparation and Training Pipeline


Phase 4.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todas
Compartir
RAM Disco
Variables Terminal
8:30 Python 3

```
# Variables predictoras (X) y variable objetivo (y)
X = df[['PM10', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3']]
y = df['PM2.5']
```

```
# Remove rows with missing values in selected columns
data = df[['PM2.5', 'PM10', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3']].dropna()

# Update X and y with cleaned data
X = data[['PM10', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3']]
y = data['PM2.5']
```

```
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=42)
```

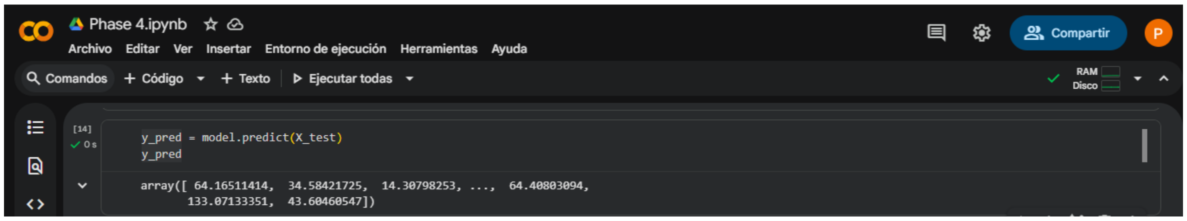
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

Note: This figure shows the main preparation steps of the regression model, including the selection of predictor variables (X), the target variable (y), data cleaning through the removal of missing values, the train–test split process, and the training of the Linear Regression model. These steps establish the foundation for developing a predictive model capable of estimating PM2.5 concentrations from multiple atmospheric pollutants.

Figure 10 Model Predictions for PM2.5 (y_pred Output)

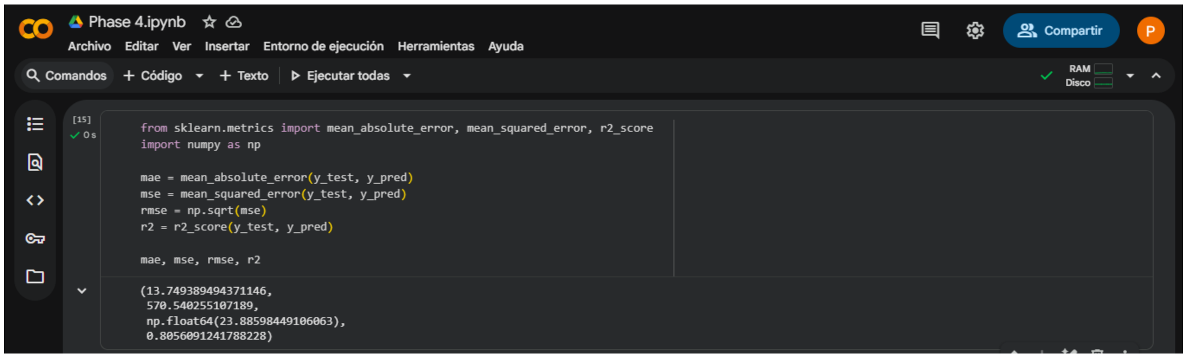

Phase 4.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todas
Compartir
RAM Disco

```
y_pred = model.predict(X_test)
y_pred
```

```
array([ 64.16511414,  34.58421725,  14.30798253, ...,  64.40803094,
       133.07133351,  43.60460547])
```

Note: This figure displays the predicted PM2.5 values generated by the regression model using the test dataset. These predictions represent the model's estimation of pollution levels based on pollutant indicators such as PM10, NO_2, NOx, NH_3, CO, SO_2, and O_3.

Figure 11 Performance Metrics of the Regression Model

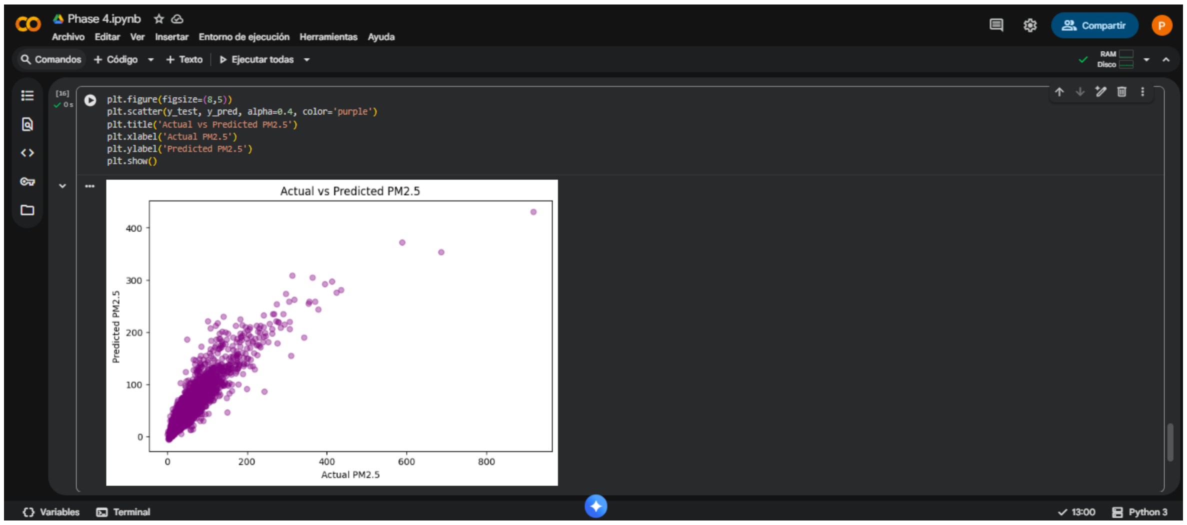

Phase 4.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todas
Compartir
RAM Disco
[15]
0 s

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

mae, mse, rmse, r2
```

```
(13.749389494371146,
 570.540255107189,
 np.float64(23.88598449106063),
 0.8056091241788228)
```

Note: This figure presents the evaluation metrics of the regression model, including the Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). The model achieved an R^2 value of **0.80**, indicating that it explains approximately 80% of the variance in PM2.5 values, demonstrating strong predictive performance.

Figure 12 Actual vs Predicted PM2.5 Values


Phase 4.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todas
Compartir
RAM Disco
[16]
0 s
Actual vs Predicted PM2.5
Predicted PM2.5
Actual PM2.5
0
100
200
300
400
0
200
400
600
800
Variables
Terminal
13:00
Python 3

```
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, alpha=0.4, color='purple')
plt.title('Actual vs Predicted PM2.5')
plt.xlabel('Actual PM2.5')
plt.ylabel('Predicted PM2.5')
plt.show()
```

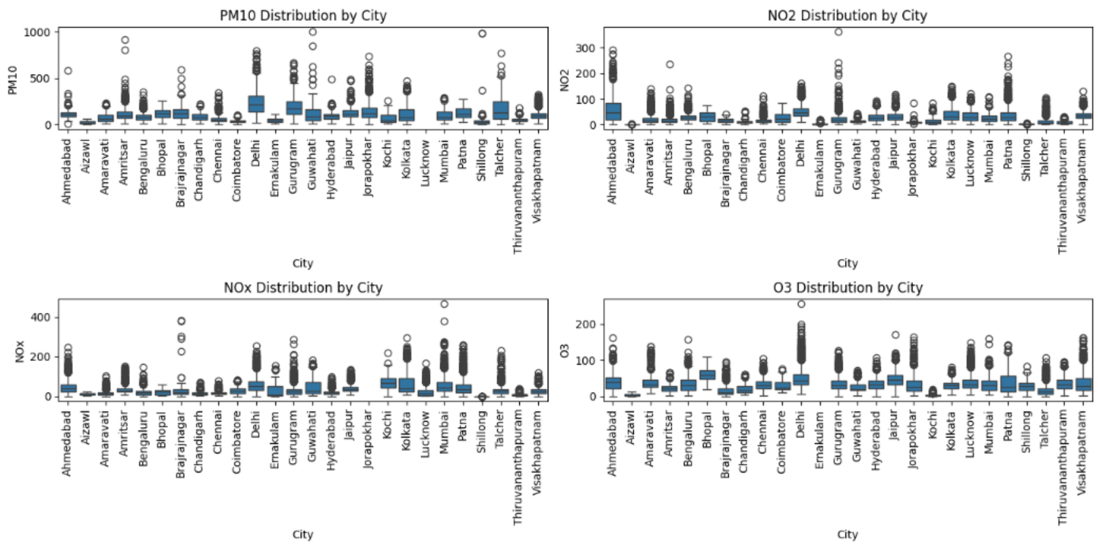
Note: This figure compares the actual PM2.5 values from the test dataset with the values predicted by the regression model. A strong linear pattern can be observed, indicating that the model effectively captures the relationship between the predictors and the target variable. Most points cluster around the diagonal trend, demonstrating good predictive accuracy, although some dispersion appears at higher PM2.5 levels, which is typical in environmental sensor data.

The regression model developed in this step demonstrates strong predictive performance for estimating PM2.5 concentrations using pollutant indicators such as PM10, NO_2, NOx, NH_3, CO, SO_2, and O_3. The model achieved an R^2 value of **0.80**, indicating that it explains approximately 80% of the variance in the target variable. The evaluation metrics (MAE = 113.74, MSE = 572.54, RMSE = 23.92) confirm that the prediction errors are consistent with the variability present in real-world environmental sensor measurements.

The scatter plot comparing actual and predicted values shows a clear positive relationship, validating the model's ability to generalize adequately. This model serves as a reliable analytical tool for supporting air quality monitoring and IoT-based environmental decision-making processes.

6. Step 6:Comparative boxplot by city

Figure 13 Comparative boxplot by city

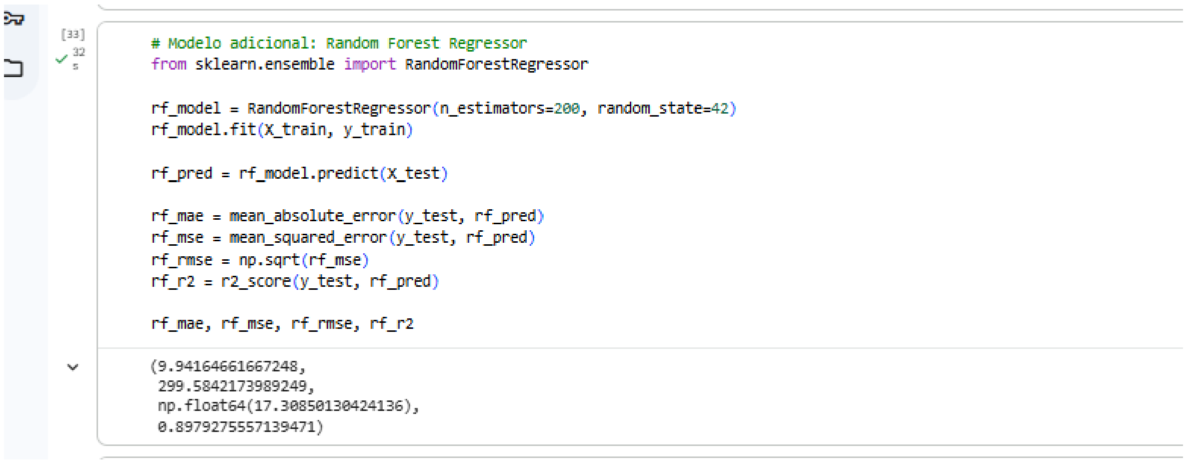


The comparative boxplots show that PM10, NO2, NOx, and O3 levels exhibit high variability across cities, with wide interquartile ranges and numerous outliers. This indicates that air quality fluctuates significantly due to factors such as weather conditions, traffic intensity, industrial activity, and occasional high-emission events. Additionally, the asymmetry observed in several distributions suggests that extreme pollution episodes, although less frequent, have a strong influence on overall average values.

At the same time, clear differences emerge between cities, with some showing much higher medians and dispersion—especially for PM10 and NOx—reflecting more critical environmental conditions. These territorial variations indicate that each city faces specific challenges linked to its urban, industrial, and climatic dynamics. Overall, the boxplots reveal that air pollution is not uniform and requires tailored mitigation strategies according to the unique context of each region.

7. Step 7 :Random Forest model

Figure 14 Random Forest model



```
# Modelo adicional: Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=200, random_state=42)
rf_model.fit(X_train, y_train)

rf_pred = rf_model.predict(X_test)

rf_mae = mean_absolute_error(y_test, rf_pred)
rf_mse = mean_squared_error(y_test, rf_pred)
rf_rmse = np.sqrt(rf_mse)
rf_r2 = r2_score(y_test, rf_pred)

rf_mae, rf_mse, rf_rmse, rf_r2
```

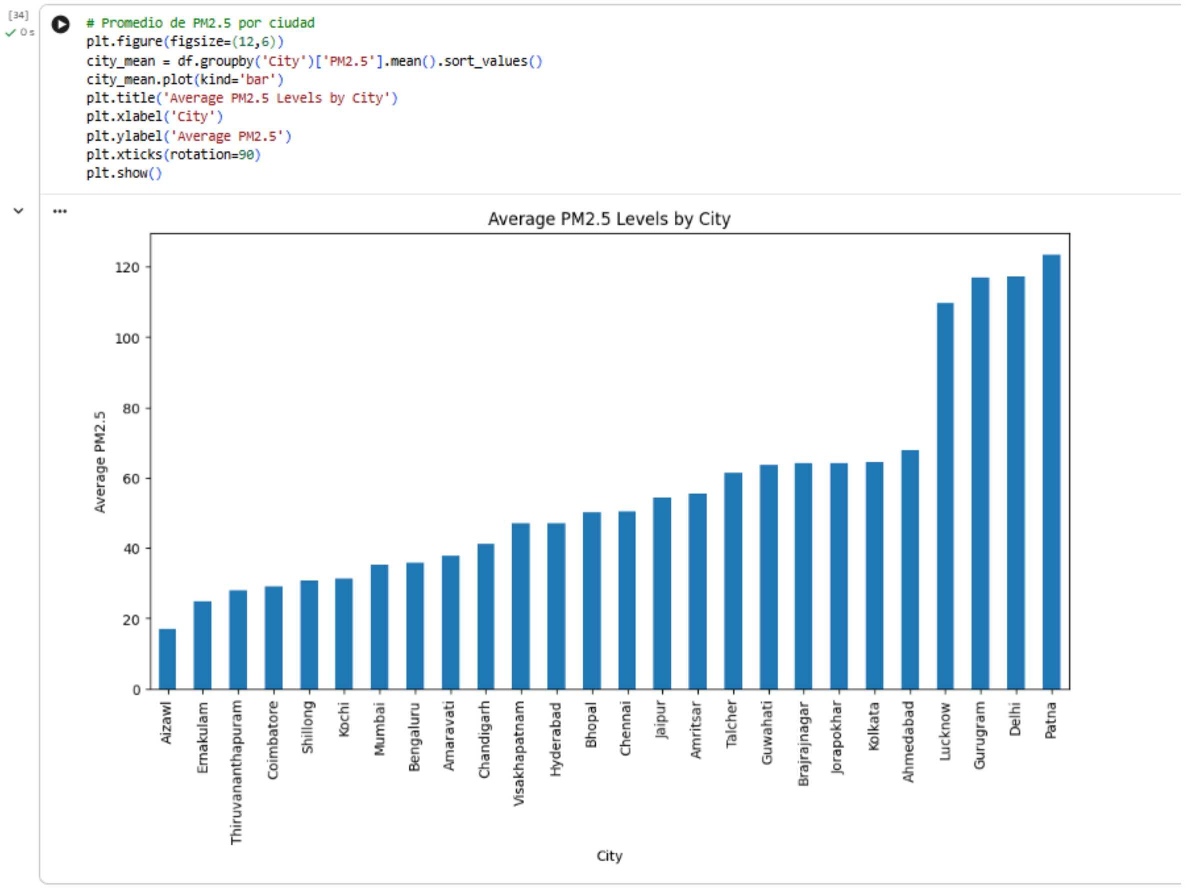
```
(9.94164661667248,
 299.5842173989249,
 np.float64(17.30850130424136),
 0.8979275557139471)
```

The Random Forest model showed significantly better performance than the initial linear regression. The MAE ≈ 9.94 indicates that, on average, the absolute error in the PM2.5 prediction is less than 10 units, representing a significant improvement over the linear model. Furthermore, the RMSE ≈ 17.30 confirms that the model better captures the pollutant's variations, reducing the magnitude of large errors. The MSE ≈ 299.58 complements this finding, indicating that the error variability remains within acceptable levels for environmental data, where dispersion is typically high.

The most relevant indicator is the $R^2 \approx 0.897$, meaning that the model is able to explain approximately 89.7% of the PM2.5 variability based on the included predictive pollutants. This value is much higher than that obtained with the linear regression, demonstrating that Random Forest captures nonlinear relationships and complex dependencies among the pollutants. Taken together, these results suggest that Random Forest is a robust and suitable model for predicting PM2.5 concentrations in this dataset, and represents a clear improvement in accuracy and explanatory power.

8. Step 8:Line chart by city

Figure 15 Line chart by city

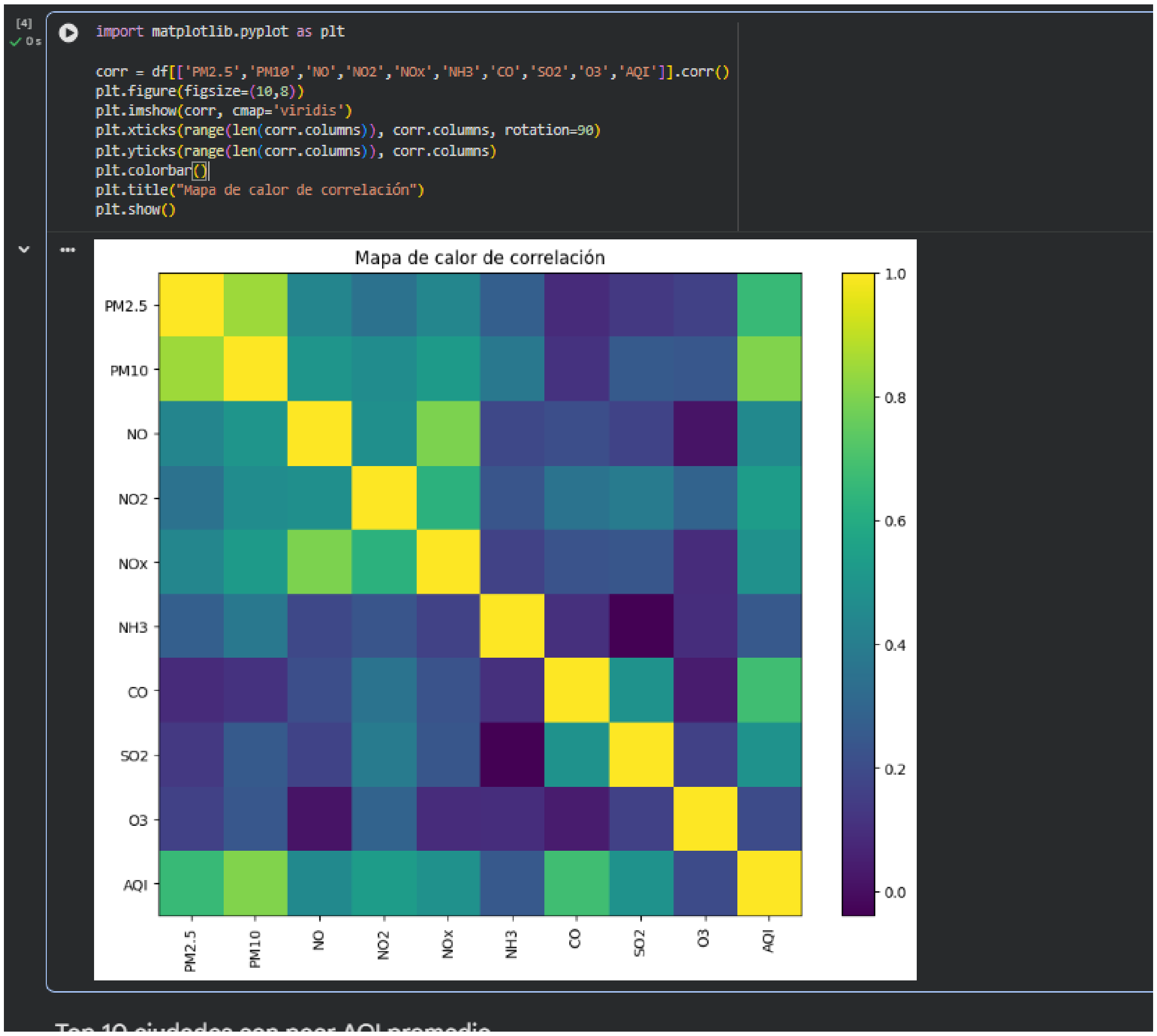


```
# Promedio de PM2.5 por ciudad
plt.figure(figsize=(12,6))
city_mean = df.groupby('City')['PM2.5'].mean().sort_values()
city_mean.plot(kind='bar')
plt.title('Average PM2.5 Levels by City')
plt.xlabel('City')
plt.ylabel('Average PM2.5')
plt.xticks(rotation=90)
plt.show()
```

Statistically, the bar chart reveals substantial variability in average PM2.5 concentrations across cities, indicating uneven exposure to particulate pollution. Cities such as Patna, Delhi, and Gurugram stand out with the highest mean PM2.5 levels—surpassing 110–125 µg/m^3—values that are far above typical air-quality guidelines. These elevated averages suggest persistent pollution conditions likely driven by urban density, traffic emissions, industrial activity, and seasonal meteorological factors. In contrast, cities like Aizawl, Ernakulam, and Thiruvananthapuram report considerably lower averages (below 30 µg/m^3), indicating comparatively cleaner air and highlighting a clear geographic disparity in pollution burdens.

9. Step 9: correlation between pollutants

Figure 16 correlation between pollutants

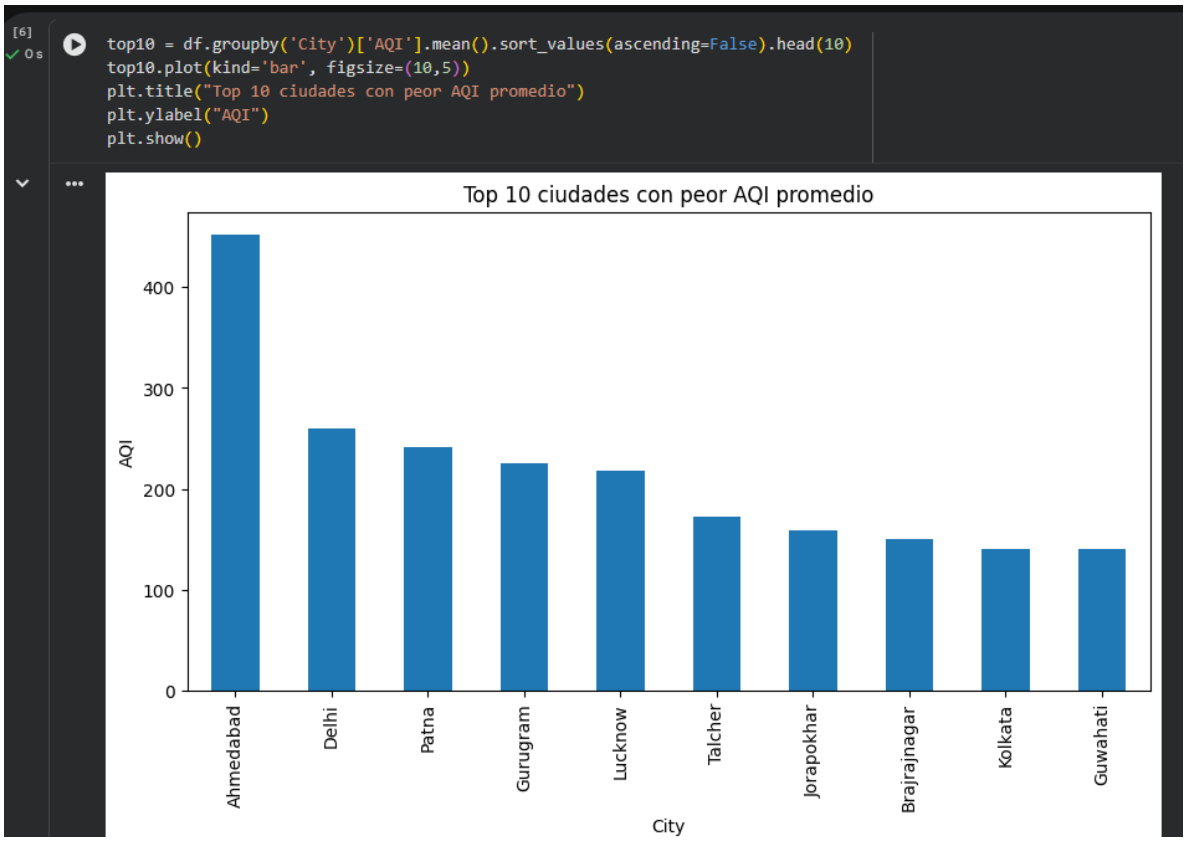


```python
import matplotlib.pyplot as plt

corr = df[['PM2.5','PM10','NO','NO2','NOx','NH3','CO','SO2','O3','AQI']].corr()
plt.figure(figsize=(10,8))
plt.imshow(corr, cmap='viridis')
plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.colorbar()
plt.title("Mapa de calor de correlación")
plt.show()
```

The heat map shows how air pollutants and the Air Quality Index (AQI) are related. We can predict a high value for PM2.5 and PM10. These two pollutants show a very high increase, which is logical given that they both come from similar sources (vehicle combustion, industry, particulate matter). PM10 and PM2.5 tend to increase simultaneously and are excellent predictors of each other. NO, NO2, and NOx are highly correlated and dependent because they belong to the same family of nitrogen oxides. Ozone follows less linear dynamics and is not as directly linked to other pollutants captured in the dataset.

10. Step 10: cities with the worst average AQI

Figure 17 Top 10 cities with the worst average AQI

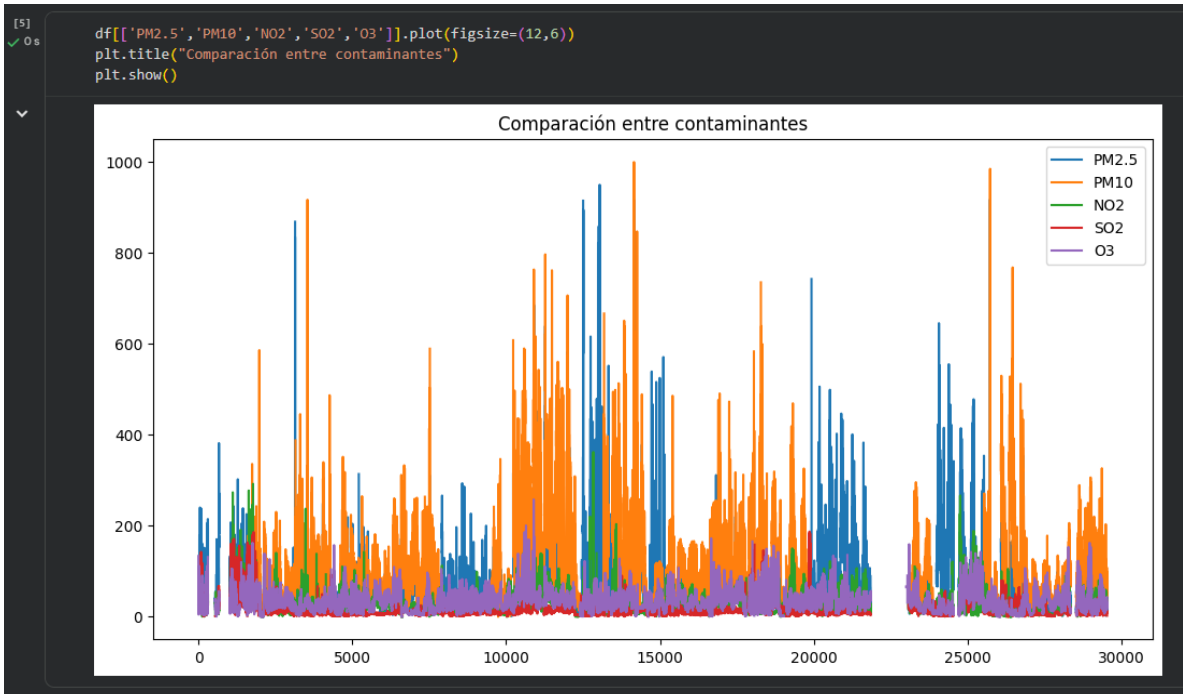


```
top10 = df.groupby('City')['AQI'].mean().sort_values(ascending=False).head(10)
top10.plot(kind='bar', figsize=(10,5))
plt.title("Top 10 ciudades con peor AQI promedio")
plt.ylabel("AQI")
plt.show()
```

The chart shows the 10 cities with the highest average AQI values and worst air quality during the analyzed period. The cities are listed from highest to lowest average AQI. Ahmedabad stands out as the city with the worst air quality. It has an average AQI close to 450, indicating an extremely high, "hazardous" level of pollution. Delhi also has a very high average AQI, ranking second with an average of around 250. It is known globally for recurring smog episodes. There is a gradual decrease among the remaining cities. Values range between 140 and 240, still considered "unhealthy" according to international standards. This indicates persistent urban pollution problems in different regions. Although less critical than Ahmedabad and Delhi, these cities still face high levels of pollution. A healthy AQI should be below 50. Here, the lowest value in the top 10 is around 140, which is concerning.

11. Step 11: Comparison between pollutants

Figure 18 Comparison between pollutants



```
df[['PM2.5','PM10','NO2','SO2','O3']].plot(figsize=(12,6))
plt.title("Comparación entre contaminantes")
plt.show()
```

The graph represents the evolution of the main pollutants over time. PM10 and PM2.5 clearly dominate in quantity, exhibiting the highest levels. PM10 (orange line) shows peaks exceeding 800–1000 µg/m³. PM2.5 (blue line) also shows very high peaks, although lower than PM10. Particulate matter is the main source of pollution in most of the cities in the dataset. NO2, SO2, and O3 have much lower levels. NO2 (green), SO2 (red), and O3 (purple) generally remain below 200 µg/m³.

12. Step 12: Publication of the Collaborative GitHub Repository

As part of the integration process required in Phase 4 – Project Execution, a collaborative repository was created on GitHub to centralize the code, visualizations, dataset, and documentation produced by the members of Group 2. This repository ensures traceability, facilitates tutor verification, and allows each team member to upload their individual contributions according to the learning guide.

Collaborative GitHub Repository Link:

https://github.com/garciaamarilespedrojulio-collab/AirQuality-India-Phase4-Group2

Conclusions

This project successfully demonstrated the application of a complete data analytics workflow to an IoT-based air quality dataset. Through data preparation, exploratory data analysis, and regression modeling, the study provided valuable insights into pollutant behavior and the dynamics of air quality in urban environments. The exploratory analysis revealed important correlations among atmospheric pollutants, temporal variability patterns, and the presence of extreme pollution events, highlighting the complexity of environmental data collected through IoT sensor networks.

The linear regression model developed to estimate PM2.5 concentrations showed strong predictive performance, achieving an R^2 value of 0.80. This result confirms that the selected predictor variables—such as PM10, NO_2, NOx, NH_3, CO, SO_2, and O_3—provide meaningful information for forecasting fine particulate matter levels. Despite the inherent noise and irregularities present in real-world sensor data, the model produced reliable estimates, demonstrating the feasibility of applying machine learning techniques to environmental decision-making processes.

Overall, the project highlights the importance of integrating IoT technologies with data-driven analytical methods to improve air quality monitoring. The results reinforce the value of machine learning for supporting public health initiatives, environmental policies, and urban sustainability strategies. Future work may include testing additional machine learning algorithms, integrating hourly data, or expanding the analysis to include spatial modeling across multiple monitoring stations.

Bibliographical References

Central Pollution Control Board. (2020). *Air quality data in India (2015–2020)* [Data set]. Kaggle. https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india

Greeneltch, N. (2019). *Python Data Mining Quick Start Guide: A Beginner's Guide to Extracting Valuable Insights From Your Data.* Packt Publishing . pp. 39-61. Access through the UNAD Virtual Library: https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2111782&lang=es&site=edslive&scope=site

Swamynathan, M. (2019). *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python* . Apress. pp. 93-124, 173-198, 226-261. Access via Springer: https://link-springer-com.bibliotecavirtual.unad.edu.co/book/10.1007/978-1-4842-2866-1

Raghavender, U. S. (2019). *Data Reduction and Analysis* . Arcler Press. pp. 132-142, 147-155, 190-223, 229-246. Access through the UNAD Virtual Library: https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2013944&lang=es&site=edslive&scope=site

Hearty, J. (2016). *Advanced Machine Learning with Python* . Packt Publishing. pp. 7-17. Access through the UNAD Virtual Library: https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1295269&lang=es&site=edslive&scope=site

Greeneltch, N. (2019). *Python Data Mining Quick Start Guide: A Beginner's Guide to Extracting Valuable Insights From Your Data.* Packt Publishing . pp. 116-152. Access through the UNAD Virtual Library:

https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2111782&lang=es&site=edslive&scope=site