In [36]:
```
# Core Libraries
import pandas as pd
import numpy as np

# Visualization Libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Statistical and Machine Learning Libraries
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# Time-Series Analysis Libraries (if needed)
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA

# Geographic Data Libraries (if needed)
import geopandas as gpd
import folium
from shapely.geometry import Point, Polygon

# Data Cleaning and Transformation
from datetime import datetime
```

In [33]:
```
!pip install pandas numpy matplotlib seaborn scikit-learn statsmodels plotly geopandas
!pip install folium
```

```
Requirement already satisfied: pandas in c:\users\aadesh\anaconda3\lib\site-packages (2.1.4)
Requirement already satisfied: numpy in c:\users\aadesh\anaconda3\lib\site-packages (1.26.4)
Requirement already satisfied: matplotlib in c:\users\aadesh\anaconda3\lib\site-packages (3.8.0)
Requirement already satisfied: seaborn in c:\users\aadesh\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: scikit-learn in c:\users\aadesh\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: statsmodels in c:\users\aadesh\anaconda3\lib\site-packages (0.14.0)
Requirement already satisfied: plotly in c:\users\aadesh\anaconda3\lib\site-packages (5.9.0)
Requirement already satisfied: geopandas in c:\users\aadesh\anaconda3\lib\site-packages (0.10.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\aadesh\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\aadesh\anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\aadesh\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aadesh\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: joblib>=1.1.1 in c:\users\aadesh\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\aadesh\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in c:\users\aadesh\anaconda3\lib\site-packages (from statsmodels) (1.11.4)
Requirement already satisfied: patsy>=0.5.2 in c:\users\aadesh\anaconda3\lib\site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\aadesh\anaconda3\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: pyogrio>=0.3.0 in c:\users\aadesh\anaconda3\lib\site-packages (from geopandas) (0.10.0)
Requirement already satisfied: pyproj>=3.3.0 in c:\users\aadesh\anaconda3\lib\site-packages (from geopandas) (3.7.0)
Requirement already satisfied: shapely>=1.8.0 in c:\users\aadesh\anaconda3\lib\site-packages (from geopandas) (2.0.6)
Requirement already satisfied: six in c:\users\aadesh\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
Requirement already satisfied: certifi in c:\users\aadesh\anaconda3\lib\site-packages (from pyogrio>=0.7.2->geopandas) (2024.2.2)
Collecting folium
  Downloading folium-0.19.2-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting branca>=0.6.0 (from folium)
  Downloading branca-0.8.1-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: jinja2>=2.9 in c:\users\aadesh\anaconda3\lib\site-packages (from folium) (3.1.3)
Requirement already satisfied: numpy in c:\users\aadesh\anaconda3\lib\site-packages (from folium) (1.26.4)
Requirement already satisfied: requests in c:\users\aadesh\anaconda3\lib\site-packages (from folium) (2.31.0)
Requirement already satisfied: xyzservices in c:\users\aadesh\anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\aadesh\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\aadesh\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\aadesh\anaconda3\lib\site-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\aadesh\anaconda3\lib\site-packages (from requests->folium) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\aadesh\anaconda3\lib\site-packages (from requests->folium) (2024.2.2)
Downloading folium-0.19.2-py2.py3-none-any.whl (110 kB)
   ---------------------------------------- 0.0/110.1 kB ? eta -:--:--
   ---------- ----------------------------- 30.7/110.1 kB 660.6 kB/s eta 0:00:01
   ---------- ----------------------------- 102.4/110.1 kB 2.0 MB/s eta 0:00:01
   ---------- ----------------------------- 110.1/110.1 kB 1.3 MB/s eta 0:00:00
Downloading branca-0.8.1-py3-none-any.whl (26 kB)
Installing collected packages: branca, folium
Successfully installed branca-0.8.1 folium-0.19.2
```

In [111]:
```
df = pd.read_csv(r"C:\Users\Aadesh\Downloads\Air_Quality.csv")
```

In [6]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18025 entries, 0 to 18024
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unique ID       18025 non-null  int64
 1   Indicator ID    18025 non-null  int64
 2   Name            18025 non-null  object
 3   Measure         18025 non-null  object
 4   Measure Info    18025 non-null  object
 5   Geo Type Name   18025 non-null  object
 6   Geo Join ID     18016 non-null  float64
 7   Geo Place Name  18016 non-null  object
 8   Time Period     18025 non-null  object
 9   Start_Date      18025 non-null  object
 10  Data Value      18025 non-null  float64
 11  Message         0 non-null      float64
dtypes: float64(3), int64(2), object(7)
memory usage: 1.7+ MB
None
```

In [7]:
```
print(df.isnull().sum())
```

```
Unique ID            0
Indicator ID         0
Name                 0
Measure              0
Measure Info         0
Geo Type Name        0
Geo Join ID          9
Geo Place Name       9
Time Period          0
Start_Date           0
Data Value           0
Message          18025
dtype: int64
```

In [14]:
```
# Remove rows where "Geo Join ID" or "Geo Place Name" is null
df_cleaned = df.dropna(subset=["Geo Join ID", "Geo Place Name"])
df_cleaned = df.drop(columns=["Message"])

# Verify the updated dataset
print(f"Rows before cleaning: {len(df)}")
print(f"Rows after cleaning: {len(df_cleaned)}")
print(f"Columns before cleaning: {df.columns.tolist()}")
print(f"Columns after cleaning: {df_cleaned.columns.tolist()}")

# Save the cleaned dataset to a new file (optional)
df_cleaned.to_csv(r"C:\Users\Aadesh\Downloads\Air_Quality_Cleaned.csv", index=False)
```

```
Rows before cleaning: 18016
Rows after cleaning: 18016
Columns before cleaning: ['Unique ID', 'Indicator ID', 'Name', 'Measure', 'Measure Info', 'Geo Type Name', 'Geo Join ID', 'Geo Place Name', 'Time Period', 'Start_Date', 'Data Value', 'Message']
Columns after cleaning: ['Unique ID', 'Indicator ID', 'Name', 'Measure', 'Measure Info', 'Geo Type Name', 'Geo Join ID', 'Geo Place Name', 'Time Period', 'Start_Date', 'Data Value']
```

In [15]:
```
df = pd.read_csv(r"C:\Users\Aadesh\Downloads\Air_Quality_Cleaned.csv")
```

In [17]:
```
#Standardize Date Format
df['Start_Date'] = pd.to_datetime(df['Start_Date'], format='%m/%d/%Y')
```

In [22]:
```
# Standardize Text Column Strip whitespace and convert to lowercase
df['Geo Place Name'] = df['Geo Place Name'].str.strip().str.lower()
```

In [59]:
```
df = df[df['Time Period'].str.contains("Annual Average")]
```

In [60]:
```
df['Time Period'] = df['Time Period'].str.replace("Annual Average", "").str.strip()
```

In [61]:
```
print(df.describe())
```

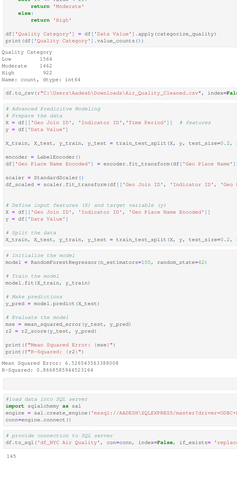
```
          Unique ID  Indicator ID   Geo Join ID  \
count   3948.000000   3948.000000  3.948000e+03
mean    418589.597518    370.000000  7.622145e+05
min     187509.000000    365.000000  1.000000e+00
25%     176565.750000    365.000000  2.030000e+02
50%     370808.500000    370.000000  3.030000e+02
75%     645204.250000    375.000000  4.040000e+02
max     826379.000000    375.000000  1.051656e+08
std     236987.987911      5.008633  8.820152e+06

                      Start_Date   Data Value          Month
count                       3948  3948.000000  3948.000000
mean   2015-06-19 10:17:08.571428352   14.340502    6.500000
min              2008-12-01 00:00:00    5.000000    1.000000
25%              2011-12-01 00:00:00    8.400000    1.000000
50%              2015-07-02 00:00:00   12.100000    6.500000
75%              2019-01-01 00:00:00   19.700000   12.000000
max              2022-01-01 00:00:00   44.800000   12.000000
std                              NaN    7.004176    5.500697
```

In [63]:
```
#Unique Values in Categorical Column
print(df['Geo Type Name'].unique())
print(df['Name'].unique())
```

```
['UHF42' 'Borough' 'UHF34' 'CD' 'Citywide']
['Fine particles (PM 2.5)' 'Nitrogen dioxide (NO2)']
```

In [64]:
```
df['Data Value'].hist(bins=50, range=(0,50) )
plt.title("Distribution of Air Quality Values")
plt.xlabel("Data Value")
plt.ylabel("Frequency")
plt.show()
```



In [65]:
```
# Create a line plot for seasonal trends
seasonal_trends = df.groupby('Month')['Data Value'].mean()
seasonal_trends.plot(kind='line', marker='o', linestyle='-', color='green', title="Seasonal Air Quality Trends")

plt.xlabel("Month")
plt.ylabel("Average Data Value")
plt.xticks(ticks=seasonal_trends.index)  # Ensures proper month labeling
plt.grid(True)  # Add grid lines for better readability
plt.show()
```


In [68]:
```
# Correlation heatmap
# Encode the "Name" column
le = LabelEncoder()
df['Name_encoded'] = le.fit_transform(df['Name'])

# Re-run heatmap with numeric columns
df = df.select_dtypes(include=['float64', 'int64'])
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```


In [ ]:
```

```

In [66]:
```
# Create column "Air Quality category" (low, moderate, high)
def categorize_quality(value):
    if value < 10:
        return "Low"
    elif 10 <= value < 20:
        return "Moderate"
    else:
        return "High"

df['Quality Category'] = df['Data Value'].apply(categorize_quality)
print(df['Quality Category'].value_counts())
```

```
Quality Category
Low         1566
Moderate    1462
High         922
Name: count, dtype: int64
```

In [67]:
```
df.to_csv(r"C:\Users\Aadesh\Downloads\Air_Quality_Cleaned.csv", index=False)
```

In [70]:
```
# Advanced Predictive Modeling
# Prepare the data
X = df[['Geo Join ID', 'Indicator ID','Time Period']]  # Features
y = df['Data Value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

encoder = LabelEncoder()
df['Geo Place Name Encoded'] = encoder.fit_transform(df['Geo Place Name'])

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[['Geo Join ID', 'Indicator ID', 'Geo Place Name Encoded']])


# Define input features (X) and target variable (y)
X = df[['Geo Join ID', 'Indicator ID', 'Geo Place Name Encoded']]
y = df['Data Value']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [71]:
```
# Initialize the model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-Squared: {r2}")
```

```
Mean Squared Error: 6.526543563380008
R-Squared: 0.8668585944523164
```

In [ ]:
```

```

In [45]:
```
#Load Data into SQL server
import sqlalchemy as sal
engine = sal.create_engine('mssql://AADESH\SQLEXPRESS2/master?driver=ODBC+DRIVER+17+FOR+SQL+SERVER')
conn=engine.connect()
```

In [47]:
```
# provide connection to SQL server
df.to_sql('df_NYC Air Quality', con=conn, index=False, if_exists= 'replace')
```

Out[47]:
```
145
```