```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Load the dataset
data = pd.read_excel('/content/New DataSet.xlsx')
```

```
print(data.head())
```

```
                     WHO Region ISO3 WHO Country Name City or Locality  \
0  Eastern Mediterranean Region  AFG      Afghanistan            Kabul   
1               European Region  ALB          Albania           Durres   
2               European Region  ALB          Albania           Durres   
3               European Region  ALB          Albania          Elbasan   
4               European Region  ALB          Albania          Elbasan   

   Measurement Year  PM2.5 (µg/m3)  PM10 (µg/m3)  NO2 (µg/m3)  \
0              2019         119.77           NaN          NaN   
1              2015            NaN         17.65        26.63   
2              2016          14.32         24.56        24.78   
3              2015            NaN           NaN        23.96   
4              2016            NaN           NaN        26.26   

   PM25 temporal coverage (%)  PM10 temporal coverage (%)  \
0                        18.0                         NaN   
1                         NaN                         NaN   
2                         NaN                         NaN   
3                         NaN                         NaN   
4                         NaN                         NaN   

   NO2 temporal coverage (%)  \
0                        NaN   
1                  83.961187   
2                  87.932605   
3                  97.853881   
4                  96.049636   

                                           Reference  \
0  U.S. Department of State, United States Enviro...   
1   European Environment Agency (downloaded in 2021)   
2   European Environment Agency (downloaded in 2021)   
3   European Environment Agency (downloaded in 2021)   
4   European Environment Agency (downloaded in 2021)   

  Number and type of monitoring stations  Version of the database  Status  
0                                    NaN                     2022     NaN  
1                                    NaN                     2022     NaN  
2                                    NaN                     2022     NaN  
3                                    NaN                     2022     NaN  
4                                    NaN                     2022     NaN  
```

```
# Check for duplicate rows
duplicates = data.duplicated()

# Count the number of duplicate rows
num_duplicates = duplicates.sum()
print(f'Number of duplicate rows: {num_duplicates}')

# Display the duplicate rows (if any)
if num_duplicates > 0:
    print("Duplicate rows:")
    print(df[duplicates])
else:
    print("No duplicate rows found.")
```

```
Number of duplicate rows: 0
No duplicate rows found.
```

```
data = data[['WHO Country Name','Measurement Year', 'PM2.5 (µg/m3)', 'PM10 (µg/m3)', 'NO2 (µg/m3)', 'PM25 temporal coverage (%)', 'PM10 temp
```

```
# Check for missing values
print(data.isnull().sum())
```

```
WHO Country Name                 0
Measurement Year                 0
PM2.5 (µg/m3)                17143
PM10 (µg/m3)                 11082
NO2 (µg/m3)                   9991
PM25 temporal coverage (%)   24916
PM10 temporal coverage (%)   26810
NO2 temporal coverage (%)    12301
dtype: int64
```

```
# Drop rows with missing NO2 values
data = data.dropna(subset=["PM2.5 (µg/m3)","PM10 (µg/m3)","NO2 (µg/m3)","PM25 temporal coverage (%)","PM10 temporal coverage (%)","NO2 tempc
```

```
# Select relevant features for the model
features = ['Measurement Year', 'PM2.5 (µg/m3)', 'PM10 (µg/m3)', 'NO2 (µg/m3)', 'PM25 temporal coverage (%)', 'PM10 temporal coverage (%)',
X = data[features]
y = data['NO2 (µg/m3)']
```

```
# Split the data 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)
```

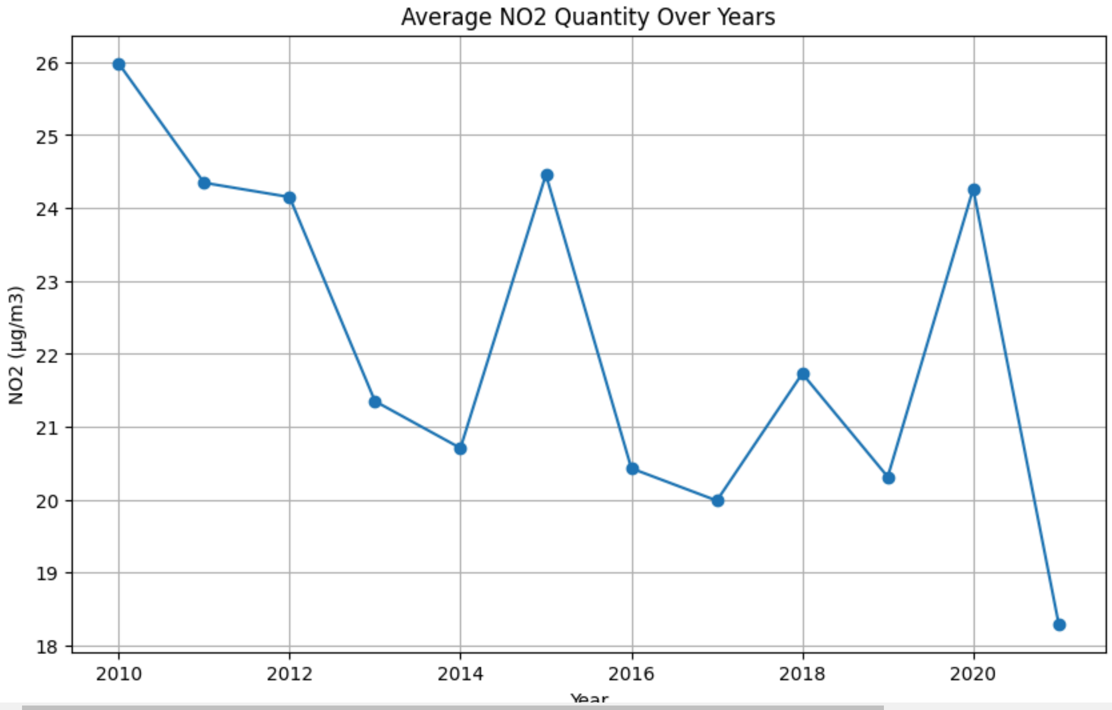
```
# Initialize the Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)

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

```
▾     RandomForestRegressor     ⓘ ?
RandomForestRegressor(random_state=42)
```

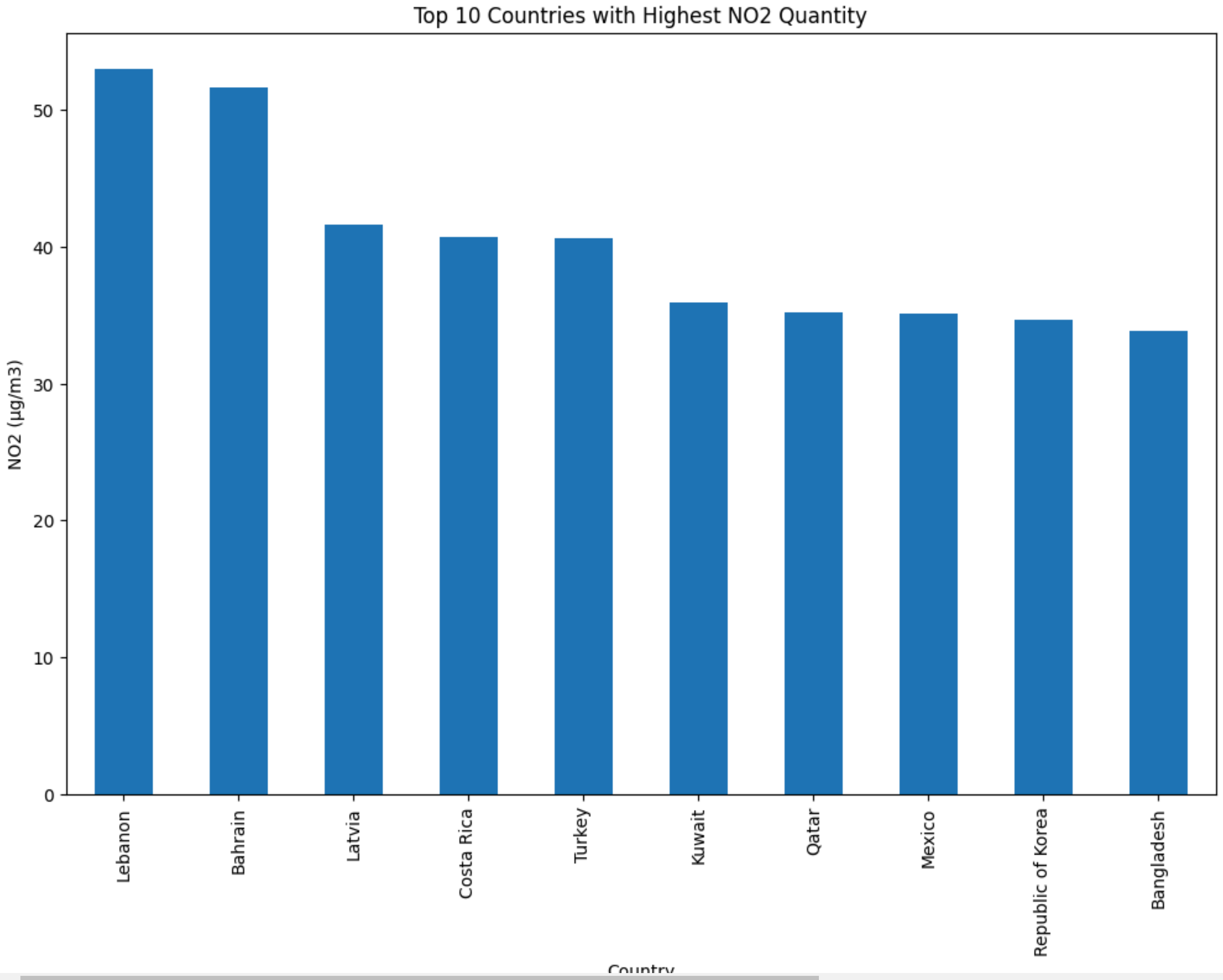
```
# Make predictions on the test set
y_pred = model.predict(X_test)

# 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^2 Score: {r2}')
```

```
Mean Squared Error: 0.0054520471987953715
R^2 Score: 0.9999634027124203
```

```
# Group by year and calculate the mean NO2 quantity
NO2_by_year = data.groupby('Measurement Year')['NO2 (µg/m3)'].mean()

# Plot the NO2 quantity over years
plt.figure(figsize=(10, 6))
plt.plot(NO2_by_year.index, NO2_by_year.values, marker='o')
plt.title('Average NO2 Quantity Over Years')
plt.xlabel('Year')
plt.ylabel('NO2 (µg/m3)')
plt.grid(True)
plt.show()
```



```
# Group by country and calculate the mean NO2 quantity
no2_by_country = data.groupby('WHO Country Name')['NO2 (µg/m3)'].mean().sort_values(ascending=False)

# Plot the top 10 countries with the highest NO2 quantity
plt.figure(figsize=(12, 8))
no2_by_country.head(10).plot(kind='bar')
plt.title('Top 10 Countries with Highest NO2 Quantity')
plt.xlabel('Country')
plt.ylabel('NO2 (µg/m3)')
plt.show()
```



```
# Select the country you want to analyze
selected_country = "Bangladesh"  # Replace with the desired country name

# Filter the dataset for the selected country
country_data = data[data['WHO Country Name'] == selected_country]

# Check if the country exists in the dataset
if country_data.empty:
    print(f"No data available for {selected_country}.")
else:
    print(f"Data for {selected_country}:")
    # Select only the relevant columns
    filtered_data = country_data[['WHO Country Name', 'Measurement Year', 'NO2 (µg/m3)']]
    print(filtered_data.head())
```

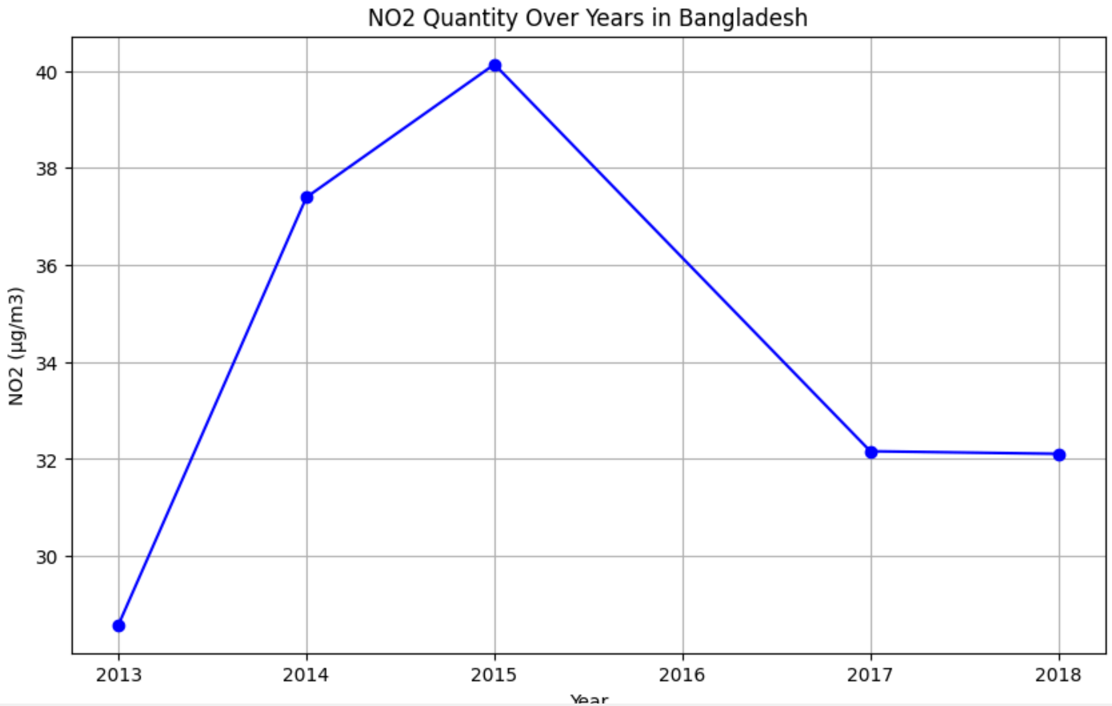
```
Data for Bangladesh:
     WHO Country Name  Measurement Year  NO2 (µg/m3)
1736       Bangladesh              2013         6.11
1737       Bangladesh              2014         5.87
1738       Bangladesh              2015         7.52
1740       Bangladesh              2017        17.79
1741       Bangladesh              2018        41.17
```

```
# Group by 'Measurement Year' and calculate the mean NO2 for the selected country
NO2_by_year = country_data.groupby('Measurement Year')['NO2 (µg/m3)'].mean()

# Display the NO2 data by year
print(NO2_by_year)
```

```
Measurement Year
2013    28.580000
2014    37.395714
2015    40.137143
```

```
2017    32.160000
2018    32.107143
Name: NO2 (µg/m3), dtype: float64
```

```
# Plot the NO2 quantity over years for the selected country
plt.figure(figsize=(10, 6))
plt.plot(NO2_by_year.index, NO2_by_year.values, marker='o', linestyle='-', color='b')
plt.title(f'NO2 Quantity Over Years in {selected_country}')
plt.xlabel('Year')
plt.ylabel('NO2 (µg/m3)')
plt.grid(True)
plt.show()
```



```
# Group by 'WHO Country Name' and calculate statistics for NO₂ levels
no2_stats_by_country = data.groupby('WHO Country Name')['NO2 (µg/m3)'].agg(
    mean='mean',
    median='median',
    std='std',
    min='min',
    max='max'
).reset_index()

# Calculate the range (max - min)
no2_stats_by_country['range'] = no2_stats_by_country['max'] - no2_stats_by_country['min']

# Display the statistics
print(no2_stats_by_country)
```

```
       WHO Country Name       mean  median        std    min    max  range
0             Australia   8.979787   8.270   4.130846   3.50  20.46  16.96
1               Austria  27.486364  31.360  11.005448   7.37  39.61  32.24
2               Bahrain  51.625000  51.625   5.932626  47.43  55.82   8.39
3            Bangladesh  33.875676  35.700  17.394212   5.87  73.70  67.83
4               Belgium  25.269474  28.660   9.131774   8.20  40.02  31.82
5                Brazil  24.506250  19.975  12.739226   7.55  45.34  37.79
6              Bulgaria  25.191429  28.910  12.593077   1.17  40.93  39.76
7                Canada  13.574156  13.160   6.145999   2.00  27.00  25.00
8                 Chile  16.460000  15.570   9.728876   7.10  43.77  36.67
9              Colombia  26.040000  24.050  10.282758  13.10  42.48  29.38
10           Costa Rica  40.710000  40.710        NaN  40.71  40.71   0.00
11              Croatia  11.760000  11.760        NaN  11.76  11.76   0.00
12               Cyprus  17.716000  17.230  10.828468   3.02  33.60  30.58
13              Czechia  18.072132  17.000   6.377071   2.54  38.72  36.18
14              Denmark  33.570000  33.570   5.317443  29.81  37.33   7.52
15              Estonia   8.793684   9.295   4.507057   1.82  17.14  15.32
16              Finland  20.992000  20.925   8.384330   5.66  35.73  30.07
17               France  26.910500  26.355   9.161944   2.99  49.09  46.10
18              Germany  29.848846  30.870  14.755895   5.91  69.84  63.93
```

```
19              Greece  21.040000  21.040        NaN  21.04  21.04   0.00
20             Hungary  27.607500  28.925   6.934293  18.19  34.39  16.20
21             Iceland   7.940000   7.940   6.392245   3.42  12.46   9.04
22               India  21.043047  19.000  10.911389   5.00  73.67  68.67
23             Ireland  28.890000  28.890        NaN  28.89  28.89   0.00
24               Italy  31.876857  31.790  12.524409   3.32  66.59  63.27
25              Jordan  27.040000  27.040        NaN  27.04  27.04   0.00
26              Kuwait  35.970000  30.460  16.420680  20.39  71.99  51.60
27              Latvia  41.660000  41.660        NaN  41.66  41.66   0.00
28             Lebanon  53.000000  53.000   8.485281  47.00  59.00  12.00
29           Lithuania  19.557333  21.060   3.713188  13.33  24.50  11.17
30          Luxembourg  30.723333  28.830  16.720590  15.03  48.31  33.28
31               Malta  17.894000  13.700  15.956243   3.02  34.85  31.83
32              Mexico  35.073056  29.810  15.026164  23.57  83.94  60.37
33         Netherlands  29.660625  34.870   9.855485  10.32  41.03  30.71
34              Norway  16.767500  17.235  14.294530   0.35  51.57  51.22
35              Poland  20.660370  21.010  10.478133   3.39  50.45  47.06
36            Portugal  17.572500  12.480  13.371794   2.69  37.56  34.87
37               Qatar  35.200000  30.000   8.700575  28.00  47.00  19.00
38   Republic of Korea  34.669412  34.780   9.700730  15.04  60.16  45.12
39             Romania  27.323000  24.430   8.801751  16.77  44.84  28.07
40             Senegal  24.676667  21.215   8.421588  15.68  36.16  20.48
41           Singapore  26.000000  26.000        NaN  26.00  26.00   0.00
42            Slovakia  26.703333  32.920  12.554323   8.08  37.97  29.89
43            Slovenia  14.856000   1.990  17.874649   1.59  34.70  33.11
44        South Africa  21.407308  18.505  13.104694   3.70  70.69  66.99
45               Spain  14.578226  10.995  10.670379   1.51  46.94  45.43
46              Sweden  10.608571   7.410  10.296860   0.48  37.57  37.09
47            Thailand  20.014583  21.000   9.383786   6.00  36.83  30.83
48 Trinidad and Tobago  12.400000  12.400   5.444722   8.55  16.25   7.70
49              Turkey  40.660000  39.810  18.196632  13.90  64.24  50.34
50      United Kingdom  32.685625  33.460  12.719567   8.68  58.80  50.12
```

```
import joblib

# Save the model to a file
joblib.dump(model, 'no2_quantity_model.pkl')
```