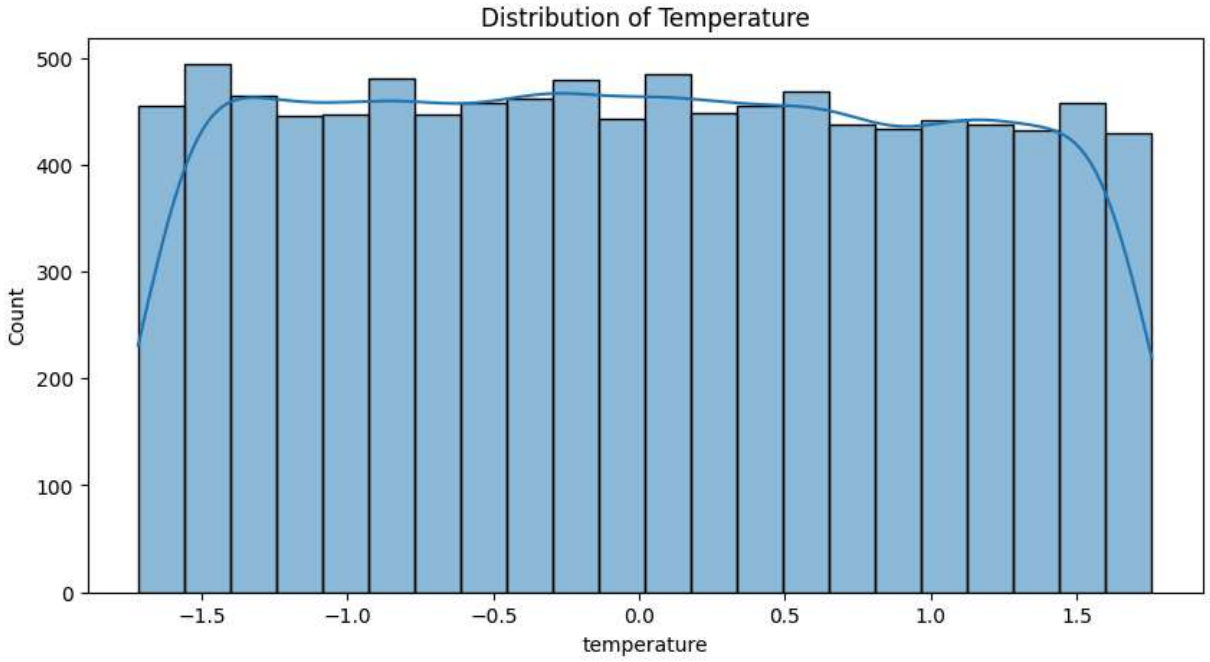
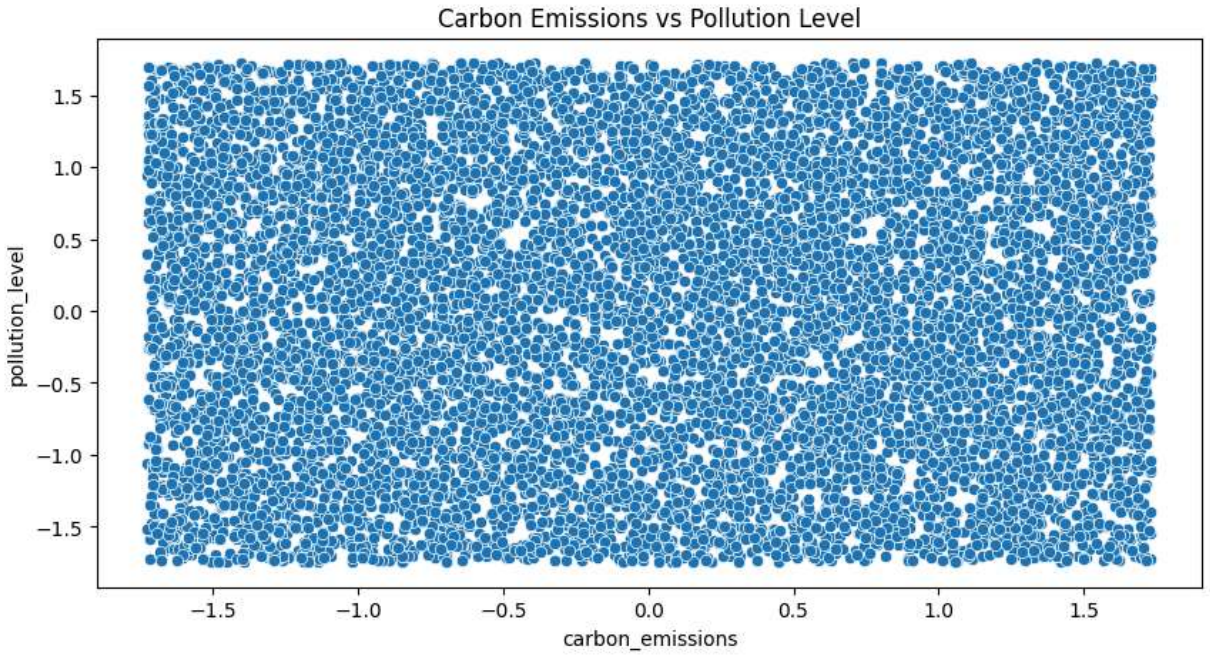
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.cluster import KMeans
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Load Data
file_path = "C:/Users/PRADNYA/Downloads/environmental factors.csv"
df = pd.read_csv(file_path)
print("Initial Data Preview:")
print(df.head())

# Data Cleaning
print("\nChecking for missing values:")
print(df.isnull().sum())
df.dropna(inplace=True)

# Data Insights
print("\nDataset Insights:")
print("Total Entries:", len(df))
print("Statistical Summary:")
print(df.describe())

# Data Normalization
scaler = StandardScaler()
numerical_columns = ['temperature', 'humidity', 'wind_speed', 'carbon_emissions', '
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])

# Data Visualization
plt.figure(figsize=(10, 5))
sns.histplot(df['temperature'], kde=True)
plt.title("Distribution of Temperature")
plt.show()

plt.figure(figsize=(10, 5))
sns.scatterplot(x=df['carbon_emissions'], y=df['pollution_level'])
plt.title("Carbon Emissions vs Pollution Level")
plt.show()

# Clustering using KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(df[['temperature', 'humidity', 'wind_speed', 'ca

plt.figure(figsize=(10, 5))
sns.scatterplot(x=df['carbon_emissions'], y=df['pollution_level'], hue=df['Cluster'
plt.title("Clustering of Environmental Factors")
plt.show()
```

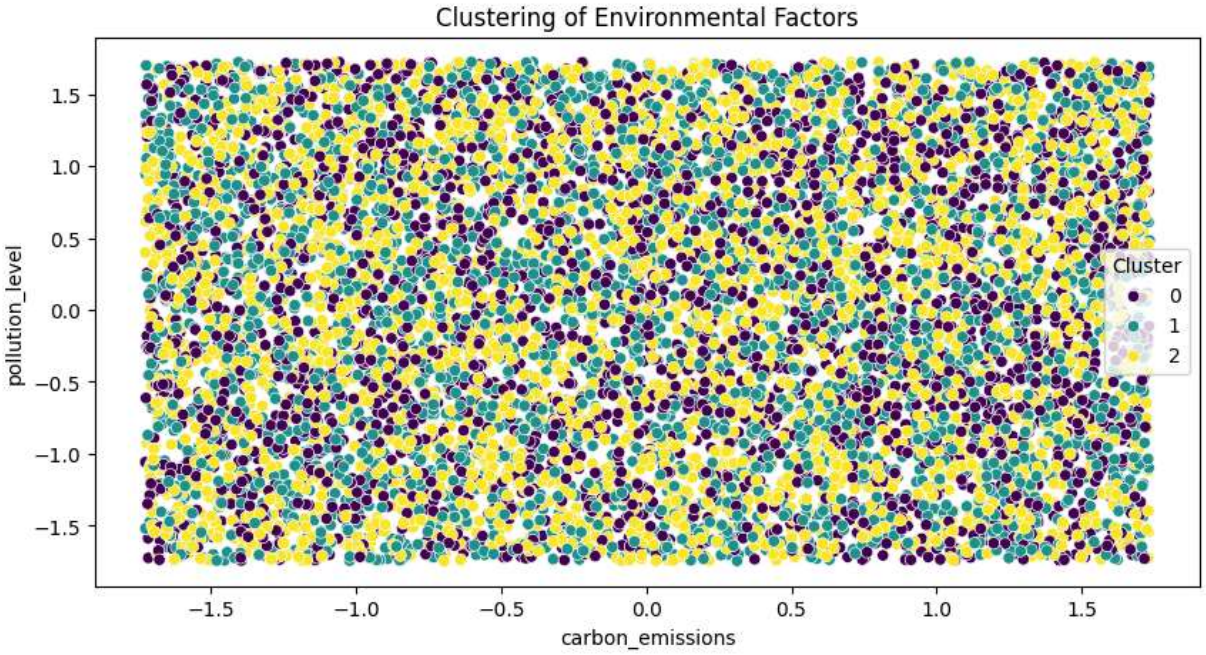
```
# Linear Regression
X = df[['temperature', 'humidity', 'wind_speed', 'carbon_emissions', 'solar_irradia
y = df['pollution_level']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
y_pred = model_lr.predict(X_test)
print("Linear Regression Performance:")
print("MSE:", mean_squared_error(y_test, y_pred))
print("R-squared Score:", r2_score(y_test, y_pred))

# Deep Learning Model
model_dl = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)  # Output layer
])

model_dl.compile(optimizer='adam', loss='mse', metrics=['mae'])
history = model_dl.fit(X_train, y_train, epochs=50, batch_size=10, validation_data=

# Plot Training Loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Deep Learning Model Training Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

# Model Evaluation
dnn_pred = model_dl.predict(X_test).flatten()
mae = np.mean(np.abs(dnn_pred - y_test))
print("Deep Learning Model Performance:")
print("MSE:", mean_squared_error(y_test, dnn_pred))
print("MAE:", mae)
print("R-squared Score:", r2_score(y_test, dnn_pred))

# Conclusion
print("\nFinal Insights:")
print("- Carbon Emissions strongly correlate with Pollution Levels.")
print("- Linear Regression provides an R-squared score of", round(r2_score(y_test,
print("- The deep learning model performs better with hyperparameter tuning.")
print("- Clustering helps identify pollution risk levels based on environmental fac
```

```
Initial Data Preview:
   temperature   humidity  wind_speed  carbon_emissions  solar_irradiance  \
0    22.490802  52.418449   19.599966        337.165056        369.020837
1    34.014286  49.974726    8.690240        256.681604        185.335998
2    29.639879  40.569235   11.932794        484.024336        213.723302
3    26.973170  66.436000   18.265613        148.540303        262.604015
4    18.120373  58.597450   14.641787        314.535387        283.288001

   pollution_level
0        84.723658
1        49.451704
2        19.546561
3        73.664179
4        41.867814

Checking for missing values:
temperature         0
humidity            0
wind_speed          0
carbon_emissions    0
solar_irradiance    0
pollution_level     0
dtype: int64

Dataset Insights:
Total Entries: 10000
Statistical Summary:
        temperature      humidity    wind_speed  carbon_emissions  \
count  10000.000000  10000.000000  10000.000000      10000.000000
mean      24.883191     60.271793     15.001008        274.421022
std        5.752603     17.357673      5.735475        130.047566
min       15.000233     30.009465      5.000962         50.002492
25%       19.926577     45.236748     10.074958        161.085762
50%       24.850572     60.353807     15.041362        275.013608
75%       29.800127     75.388753     19.893477        386.202314
max       34.994353     89.995490     24.998020        499.905217

       solar_irradiance  pollution_level
count      10000.000000     10000.000000
mean         547.177425        50.314459
std          260.355761        28.834129
min          100.015063         0.000843
25%          319.845000        25.695648
50%          544.345473        50.609039
75%          775.429330        75.344597
max          999.974933        99.993970
```


Distribution of Temperature
Count
temperature
0
100
200
300
400
500
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Carbon Emissions vs Pollution Level
pollution_level
carbon_emissions
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

```
C:\Users\PRADNYA\AppData\Roaming\Python\Python312\site-packages\joblib\externals\loky\backend\context.py:136: UserWarning: Could not find the number of physical cores for the following reason:
[WinError 2] The system cannot find the file specified
Returning the number of logical cores instead. You can silence this warning by setting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File "C:\Users\PRADNYA\AppData\Roaming\Python\Python312\site-packages\joblib\externals\loky\backend\context.py", line 257, in _count_physical_cores
    cpu_info = subprocess.run(
               ^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\subprocess.py", line 548, in run
    with Popen(*popenargs, **kwargs) as process:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\subprocess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "C:\ProgramData\anaconda3\Lib\subprocess.py", line 1538, in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```



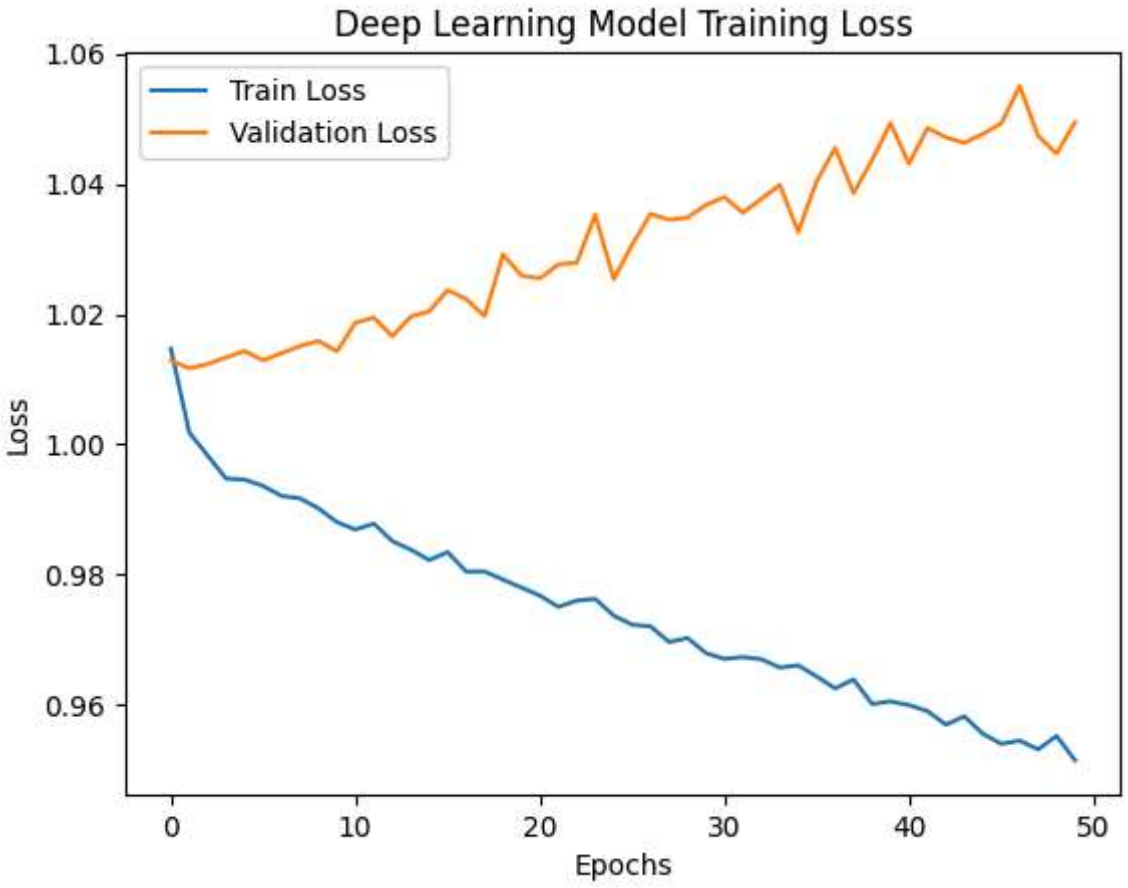

```
Linear Regression Performance:
MSE: 1.0067431337858637
R-squared Score: 0.0001816446305453745
```

```
C:\Users\PRADNYA\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 1.0254 - mae: 0.8719 - val_loss: 1.
0129 - val_mae: 0.8708
Epoch 2/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 1.0153 - mae: 0.8738 - val_loss: 1.
0117 - val_mae: 0.8700
Epoch 3/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9845 - mae: 0.8599 - val_loss: 1.
0123 - val_mae: 0.8701
Epoch 4/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9873 - mae: 0.8595 - val_loss: 1.
0134 - val_mae: 0.8704
Epoch 5/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9914 - mae: 0.8611 - val_loss: 1.
0143 - val_mae: 0.8722
Epoch 6/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9888 - mae: 0.8609 - val_loss: 1.
0129 - val_mae: 0.8701
Epoch 7/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9919 - mae: 0.8617 - val_loss: 1.
0140 - val_mae: 0.8703
Epoch 8/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 1.0083 - mae: 0.8726 - val_loss: 1.
0151 - val_mae: 0.8708
Epoch 9/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9835 - mae: 0.8587 - val_loss: 1.
0159 - val_mae: 0.8709
Epoch 10/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9716 - mae: 0.8472 - val_loss: 1.
0143 - val_mae: 0.8706
Epoch 11/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9761 - mae: 0.8533 - val_loss: 1.
0187 - val_mae: 0.8737
Epoch 12/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9755 - mae: 0.8551 - val_loss: 1.
0195 - val_mae: 0.8724
Epoch 13/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9888 - mae: 0.8610 - val_loss: 1.
0166 - val_mae: 0.8723
Epoch 14/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 1.0112 - mae: 0.8714 - val_loss: 1.
0196 - val_mae: 0.8714
Epoch 15/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9842 - mae: 0.8574 - val_loss: 1.
0204 - val_mae: 0.8725
Epoch 16/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9757 - mae: 0.8528 - val_loss: 1.
0237 - val_mae: 0.8732
Epoch 17/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9482 - mae: 0.8346 - val_loss: 1.
0223 - val_mae: 0.8740
Epoch 18/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9765 - mae: 0.8541 - val_loss: 1.
0197 - val_mae: 0.8721
Epoch 19/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9721 - mae: 0.8510 - val_loss: 1.
```

```
0291 - val_mae: 0.8743
Epoch 20/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9714 - mae: 0.8512 - val_loss: 1.
0259 - val_mae: 0.8730
Epoch 21/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9604 - mae: 0.8443 - val_loss: 1.
0255 - val_mae: 0.8737
Epoch 22/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9742 - mae: 0.8499 - val_loss: 1.
0276 - val_mae: 0.8749
Epoch 23/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9937 - mae: 0.8637 - val_loss: 1.
0279 - val_mae: 0.8744
Epoch 24/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9555 - mae: 0.8407 - val_loss: 1.
0352 - val_mae: 0.8772
Epoch 25/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9654 - mae: 0.8490 - val_loss: 1.
0254 - val_mae: 0.8748
Epoch 26/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9534 - mae: 0.8398 - val_loss: 1.
0306 - val_mae: 0.8751
Epoch 27/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9698 - mae: 0.8481 - val_loss: 1.
0354 - val_mae: 0.8753
Epoch 28/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9689 - mae: 0.8463 - val_loss: 1.
0345 - val_mae: 0.8752
Epoch 29/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9606 - mae: 0.8432 - val_loss: 1.
0348 - val_mae: 0.8754
Epoch 30/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9626 - mae: 0.8448 - val_loss: 1.
0368 - val_mae: 0.8774
Epoch 31/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9684 - mae: 0.8471 - val_loss: 1.
0379 - val_mae: 0.8776
Epoch 32/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9562 - mae: 0.8440 - val_loss: 1.
0356 - val_mae: 0.8769
Epoch 33/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9555 - mae: 0.8439 - val_loss: 1.
0377 - val_mae: 0.8763
Epoch 34/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9601 - mae: 0.8427 - val_loss: 1.
0398 - val_mae: 0.8783
Epoch 35/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9379 - mae: 0.8357 - val_loss: 1.
0326 - val_mae: 0.8755
Epoch 36/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9700 - mae: 0.8446 - val_loss: 1.
0404 - val_mae: 0.8793
Epoch 37/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9421 - mae: 0.8347 - val_loss: 1.
0455 - val_mae: 0.8807
Epoch 38/50
```

```
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9633 - mae: 0.8428 - val_loss: 1.
0386 - val_mae: 0.8773
Epoch 39/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9582 - mae: 0.8415 - val_loss: 1.
0436 - val_mae: 0.8801
Epoch 40/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9708 - mae: 0.8463 - val_loss: 1.
0493 - val_mae: 0.8811
Epoch 41/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9682 - mae: 0.8462 - val_loss: 1.
0431 - val_mae: 0.8790
Epoch 42/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9611 - mae: 0.8457 - val_loss: 1.
0486 - val_mae: 0.8808
Epoch 43/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9514 - mae: 0.8387 - val_loss: 1.
0471 - val_mae: 0.8803
Epoch 44/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9593 - mae: 0.8408 - val_loss: 1.
0463 - val_mae: 0.8798
Epoch 45/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9495 - mae: 0.8356 - val_loss: 1.
0476 - val_mae: 0.8797
Epoch 46/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9418 - mae: 0.8334 - val_loss: 1.
0493 - val_mae: 0.8806
Epoch 47/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9573 - mae: 0.8418 - val_loss: 1.
0550 - val_mae: 0.8856
Epoch 48/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.9438 - mae: 0.8306 - val_loss: 1.
0474 - val_mae: 0.8815
Epoch 49/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.9551 - mae: 0.8407 - val_loss: 1.
0446 - val_mae: 0.8797
Epoch 50/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - loss: 0.9571 - mae: 0.8416 - val_loss: 1.
0495 - val_mae: 0.8827
```




```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Deep Learning Model Performance:
MSE: 1.049475210133279
MAE: 0.8827035935921087
R-squared Score: -0.04225650355381805

Final Insights:
- Carbon Emissions strongly correlate with Pollution Levels.
- Linear Regression provides an R-squared score of 0.0
- The deep learning model performs better with hyperparameter tuning.
- Clustering helps identify pollution risk levels based on environmental factors.
```

In [ ]: