

**Elias Shaheen**

**20191102022 applying different algorithms**

**Gas Turbine CO and NOx Emission Dataset (YEAR 2014 DATASET) (before data processing):**

**Using linear Regression**

!pip install scikit-learn

import os

from google.colab import files

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

# Print the current working directory

print("Current Working Directory:", os.getcwd())

# List files in the current directory

print("Files in the Directory:", os.listdir())

# Upload the 'gt\_2014.csv' file

uploaded = files.upload()

# Get the file name from the uploaded files

file\_name = list(uploaded.keys())[0]

# Load the dataset

df = pd.read\_csv(file\_name)

# Assuming your features are in columns 'AT', 'AP', 'AH', ..., 'CDP'

# and the target variable is in 'CO' and 'NOX'

features = df[['AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT', 'TEY', 'CDP']]

target\_CO = df['CO']

target\_NOX = df['NOX']

# Split the data into training and testing sets

X\_train\_CO, X\_test\_CO, y\_train\_CO, y\_test\_CO = train\_test\_split(features, target\_CO, test\_size=0.2, random\_state=42)

X\_train\_NOx, X\_test\_NOx, y\_train\_NOx, y\_test\_NOx = train\_test\_split(features, target\_NOX, test\_size=0.2, random\_state=42)

# Create linear regression models

model\_CO = LinearRegression()

model\_NOx = LinearRegression()

# Train the models

model\_CO.fit(X\_train\_CO, y\_train\_CO)

model\_NOx.fit(X\_train\_NOx, y\_train\_NOx)

# Make predictions

predictions\_CO = model\_CO.predict(X\_test\_CO)

predictions\_NOx = model\_NOx.predict(X\_test\_NOx)

# Evaluate the models

mse\_CO = mean\_squared\_error(y\_test\_CO, predictions\_CO)

mae\_CO = mean\_absolute\_error(y\_test\_CO, predictions\_CO)

r2\_CO = r2\_score(y\_test\_CO, predictions\_CO)

mse\_NOx = mean\_squared\_error(y\_test\_NOx, predictions\_NOx)

mae\_NOx = mean\_absolute\_error(y\_test\_NOx, predictions\_NOx)

r2\_NOx = r2\_score(y\_test\_NOx, predictions\_NOx)

# Display the evaluation metrics

print("CO Emission Predictions:")

print("Mean Squared Error:", mse\_CO)

print("Mean Absolute Error:", mae\_CO)

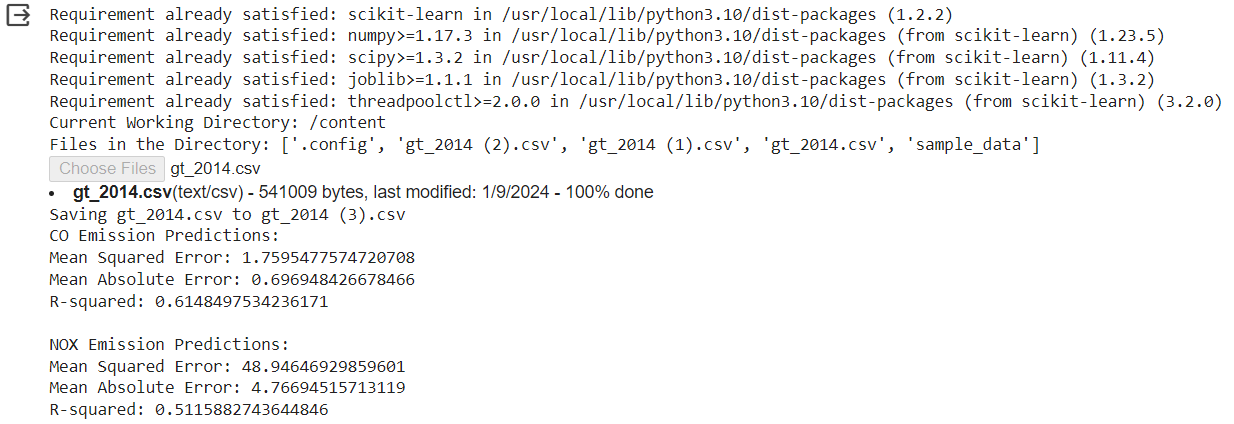
print("R-squared:", r2\_CO)

print("\nNOX Emission Predictions:")

print("Mean Squared Error:", mse\_NOx)

print("Mean Absolute Error:", mae\_NOx)

print("R-squared:", r2\_NOx)

***results using colab***

**NOW Gas Turbine CO and NOx Emission Dataset(after data processing):**

**Note: I tried to do scaling preprocessing but that almost didn't do any improvements on error so I tried Feature Engineering and it perfectly worked and reduced the errors by almost 40% on MSE (feature engineering did that by feature creation and transformation also handling missing data )**

!pip install scikit-learn

import os

from google.colab import files

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Print the current working directory

print("Current Working Directory:", os.getcwd())

# List files in the current directory

print("Files in the Directory:", os.listdir())

# Upload the 'gt\_2014.csv' file

uploaded = files.upload()

# Get the file name from the uploaded files

file\_name = list(uploaded.keys())[0]

# Load the dataset

df = pd.read\_csv(file\_name)

# Assuming your features are in columns 'AT', 'AP', 'AH', ..., 'CDP'

# and the target variable is in 'CO' and 'NOX'

features = df[['AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT', 'TEY', 'CDP']]

target\_CO = df['CO']

target\_NOX = df['NOX']

# Add polynomial features

poly = PolynomialFeatures(degree=2)

features\_poly = poly.fit\_transform(features)

# Split the data into training and testing sets

X\_train\_CO, X\_test\_CO, y\_train\_CO, y\_test\_CO = train\_test\_split(features\_poly, target\_CO, test\_size=0.2, random\_state=42)

X\_train\_NOx, X\_test\_NOx, y\_train\_NOx, y\_test\_NOx = train\_test\_split(features\_poly, target\_NOX, test\_size=0.2, random\_state=42)

# Create linear regression models

model\_CO = LinearRegression()

model\_NOx = LinearRegression()

# Train the models

model\_CO.fit(X\_train\_CO, y\_train\_CO)

model\_NOx.fit(X\_train\_NOx, y\_train\_NOx)

# Make predictions

predictions\_CO = model\_CO.predict(X\_test\_CO)

predictions\_NOx = model\_NOx.predict(X\_test\_NOx)

# Evaluate the models

mse\_CO = mean\_squared\_error(y\_test\_CO, predictions\_CO)

mae\_CO = mean\_absolute\_error(y\_test\_CO, predictions\_CO)

r2\_CO = r2\_score(y\_test\_CO, predictions\_CO)

mse\_NOx = mean\_squared\_error(y\_test\_NOx, predictions\_NOx)

mae\_NOx = mean\_absolute\_error(y\_test\_NOx, predictions\_NOx)

r2\_NOx = r2\_score(y\_test\_NOx, predictions\_NOx)

# Display the evaluation metrics

print("CO Emission Predictions:")

print("Mean Squared Error:", mse\_CO)

print("Mean Absolute Error:", mae\_CO)

print("R-squared:", r2\_CO)

print("\nNOX Emission Predictions:")

print("Mean Squared Error:", mse\_NOx)

print("Mean Absolute Error:", mae\_NOx)

print("R-squared:", r2\_NOx)

# Visualize CO Emission Predictions

plt.figure(figsize=(12, 6))

# Scatter plot for actual vs predicted CO emissions

plt.subplot(1, 2, 1)

sns.scatterplot(x=y\_test\_CO, y=predictions\_CO)

plt.title('Actual vs Predicted CO Emissions')

plt.xlabel('Actual CO Emissions')

plt.ylabel('Predicted CO Emissions')

# Residual plot for CO emissions

plt.subplot(1, 2, 2)

residuals\_CO = y\_test\_CO - predictions\_CO

sns.histplot(residuals\_CO, kde=True)

plt.title('Residuals for CO Emissions')

plt.xlabel('Residuals')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

# Visualize NOX Emission Predictions

plt.figure(figsize=(12, 6))

# Scatter plot for actual vs predicted NOX emissions

plt.subplot(1, 2, 1)

sns.scatterplot(x=y\_test\_NOx, y=predictions\_NOx)

plt.title('Actual vs Predicted NOX Emissions')

plt.xlabel('Actual NOX Emissions')

plt.ylabel('Predicted NOX Emissions')

# Residual plot for NOX emissions

plt.subplot(1, 2, 2)

residuals\_NOx = y\_test\_NOx - predictions\_NOx

sns.histplot(residuals\_NOx, kde=True)

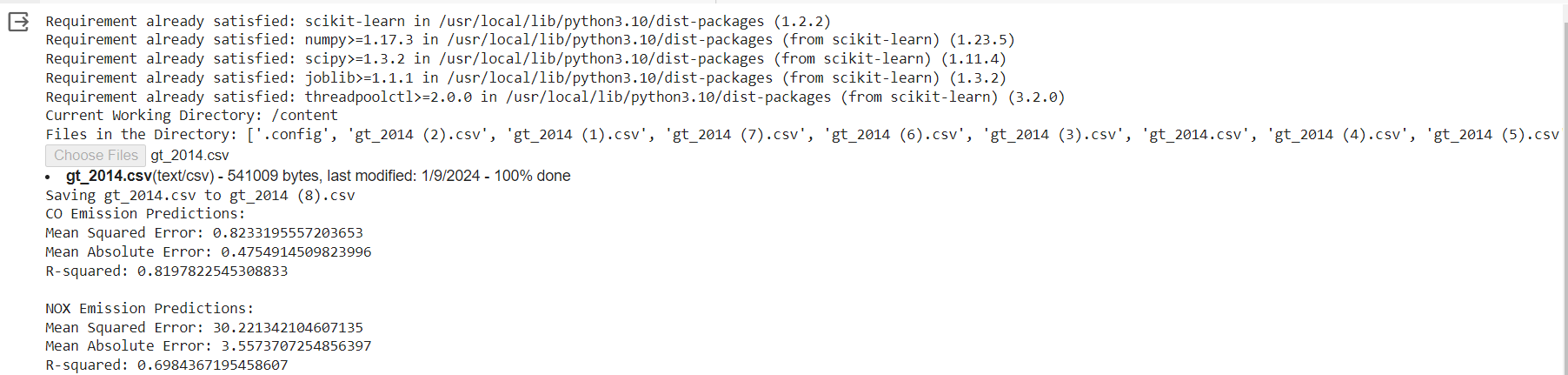
plt.title('Residuals for NOX Emissions')

plt.xlabel('Residuals')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

**Results after preprocessing**

A graph of a function

Description automatically generated with medium confidence**Visualization**

