Name : Shivam Bhatt

Enrollment No: 92301733046

Code:

```
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, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.impute import SimpleImputer # For handling potential missing values

# --- Configuration ---
FILE_PATH_METHANE = '/content/ethylene_methane.txt'
FILE_PATH_CO = '/content/ethylene_CO.txt'

# --- 1. Create Mock Data (Replace with your actual file loading) ---
# Assuming 'Ethylene' is the target, 'Methane'/'CO' are key features, plus others.
# Let's simulate some relationships.

# Mock Data for ethylene_methane.txt
np.random.seed(42)
num_samples = 100

data_methane = {
    'Methane_Concentration': np.random.rand(num_samples) * 100 + 50, # 50-150 ppm
    'Temperature_C': np.random.rand(num_samples) * 30 + 10,      # 10-40 C
    'Humidity_Percent': np.random.rand(num_samples) * 50 + 30,    # 30-80 %
```

```
    'Pressure_kPa': np.random.rand(num_samples) * 20 + 90,       # 90-110 kPa
    'Ethylene_Concentration': (
        5 + 0.8 * (np.random.rand(num_samples) * 100 + 50)  # Methane effect
        + 0.5 * (np.random.rand(num_samples) * 30 + 10)  # Temperature effect
        - 0.2 * (np.random.rand(num_samples) * 50 + 30)  # Humidity effect
        + np.random.randn(num_samples) * 5 # Noise
    )
}
df_methane = pd.DataFrame(data_methane)
# Introduce some missing values for demonstration
df_methane.loc[df_methane.sample(frac=0.05).index, 'Methane_Concentration'] = np.nan
df_methane.loc[df_methane.sample(frac=0.03).index, 'Temperature_C'] = np.nan

print("--- Mock df_methane Head ---")
print(df_methane.head())
print("\n--- Mock df_methane Info ---")
df_methane.info()
print("\n")

# Mock Data for ethylene_CO.txt
data_co = {
    'CO_Concentration': np.random.rand(num_samples) * 20 + 1,    # 1-21 ppm
    'Light_Intensity_Lux': np.random.rand(num_samples) * 1000 + 100, # 100-1100 Lux
    'Flow_Rate_LPM': np.random.rand(num_samples) * 5 + 1,        # 1-6 LPM
    'Ethylene_Concentration': (
        2 + 1.2 * (np.random.rand(num_samples) * 20 + 1)  # CO effect
        + 0.01 * (np.random.rand(num_samples) * 1000 + 100) # Light effect
        + np.random.randn(num_samples) * 3 # Noise
    )
}
df_co = pd.DataFrame(data_co)
```

```
# Introduce some missing values for demonstration
df_co.loc[df_co.sample(frac=0.07).index, 'CO_Concentration'] = np.nan

print("--- Mock df_co Head ---")
print(df_co.head())
print("\n--- Mock df_co Info ---")
df_co.info()
print("\n")

# --- Function to process and model each dataset ---
def analyze_dataset(df, dataset_name, target_column='Ethylene_Concentration'):
    print(f"\n--- Analyzing Dataset: {dataset_name} ---")

    # --- 1. Data Understanding & Preprocessing ---
    print("\n1. Data Understanding & Preprocessing")
    print("\nInitial Data Info:")
    df.info()
    print("\nInitial Data Description:")
    print(df.describe())
    print("\nMissing Values Before Imputation:")
    print(df.isnull().sum())

    # Drop rows where the target variable is missing (critical for training)
    df.dropna(subset=[target_column], inplace=True)

    # Impute missing values for features (using mean for numerical)
    # This imputer will be fit on training data and transform train/test
    imputer = SimpleImputer(strategy='mean')
    # Identify numerical columns for imputation, excluding the target if it's already handled
    numerical_cols = df.select_dtypes(include=np.number).columns.tolist()
    if target_column in numerical_cols:
```

```
    numerical_cols.remove(target_column)

# Apply imputer to the numerical columns
df[numerical_cols] = imputer.fit_transform(df[numerical_cols])

print("\nMissing Values After Imputation:")
print(df.isnull().sum())

# Visualize distributions
plt.figure(figsize=(15, 5))
for i, col in enumerate(df.drop(columns=[target_column]).columns):
    plt.subplot(1, len(df.drop(columns=[target_column]).columns), i + 1)
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(df[target_column], kde=True)
plt.title(f'Distribution of {target_column}')
plt.show()

# Correlation Matrix
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title(f'Correlation Matrix for {dataset_name}')
plt.show()

# Define features (X) and target (y)
X = df.drop(columns=[target_column])
y = df[target_column]
```

```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"\nTrain set size: {X_train.shape[0]} samples")
print(f"Test set size: {X_test.shape[0]} samples")

# Feature Scaling (Standardization)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Convert scaled arrays back to DataFrames for easier handling if needed (optional)
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns, index=X_train.index)
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X_test.columns, index=X_test.index)

# --- 2. Model Implementation & 3. Performance Evaluation ---
print("\n2. Model Implementation & 3. Performance Evaluation")

models = {
    'Linear Regression': LinearRegression(),
    'Lasso Regression': Lasso(alpha=0.1, random_state=42), # Alpha is a hyperparameter
    'Ridge Regression': Ridge(alpha=1.0, random_state=42), # Alpha is a hyperparameter
    'Decision Tree Regressor': DecisionTreeRegressor(random_state=42, max_depth=5), #
max_depth is a hyperparameter
    'Random Forest Regressor': RandomForestRegressor(n_estimators=100, random_state=42,
max_depth=8) # n_estimators, max_depth are hyperparameters
}

results = {}

for name, model in models.items():
    print(f"\n--- Training {name} ---")
```

```
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)

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)

results[name] = {'MAE': mae, 'MSE': mse, 'RMSE': rmse, 'R2': r2}

print(f"{name} Performance:")
print(f"  MAE: {mae:.4f}")
print(f"  MSE: {mse:.4f}")
print(f"  RMSE: {rmse:.4f}")
print(f"  R-squared: {r2:.4f}")

# Plot Actual vs. Predicted values
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title(f'{name}: Actual vs. Predicted ({dataset_name})')
plt.grid(True)
plt.show()

# Plot Residuals
residuals = y_test - y_pred
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred, y=residuals, alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--')
```

```
        plt.xlabel("Predicted Values")
        plt.ylabel("Residuals")
        plt.title(f'{name}: Residual Plot ({dataset_name})')
        plt.grid(True)
        plt.show()

    # --- 4. Analysis & Insights ---
    print("\n4. Analysis & Insights")
    results_df = pd.DataFrame(results).T
    print("\n--- All Model Performance Summary ---")
    print(results_df.sort_values(by='R2', ascending=False))

    best_model_name = results_df['R2'].idxmax()
    print(f"\nBest performing model based on R-squared: {best_model_name}")

    # Feature Importance (for tree-based models)
    if best_model_name in ['Decision Tree Regressor', 'Random Forest Regressor']:
        print(f"\n--- Feature Importance for {best_model_name} ---")
        model = models[best_model_name]
        feature_importances = pd.Series(model.feature_importances_,
index=X.columns).sort_values(ascending=False)
        print(feature_importances)

        plt.figure(figsize=(10, 6))
        sns.barplot(x=feature_importances.values, y=feature_importances.index)
        plt.title(f'Feature Importance for {best_model_name} ({dataset_name})')
        plt.xlabel('Importance')
        plt.ylabel('Feature')
        plt.show()
    elif best_model_name in ['Linear Regression', 'Lasso Regression', 'Ridge Regression']:
        print(f"\n--- Coefficients for {best_model_name} ---")
```

```
        model = models[best_model_name]
        coefficients = pd.Series(model.coef_, index=X.columns).sort_values(ascending=False)
        print(coefficients)

        plt.figure(figsize=(10, 6))
        sns.barplot(x=coefficients.values, y=coefficients.index)
        plt.title(f'Feature Coefficients for {best_model_name} ({dataset_name})')
        plt.xlabel('Coefficient Value')
        plt.ylabel('Feature')
        plt.show()

    print("\n--- General Insights ---")
    print(f"The {best_model_name} generally performed best for {dataset_name} based on R-squared.")
    print("Consider further hyperparameter tuning (e.g., using GridSearchCV or RandomizedSearchCV) for each model to optimize performance.")
    print("The correlation matrix and feature importance plots give insights into which factors are most influential on Ethylene concentration.")
    print("Residual plots can indicate if the model is missing any patterns (e.g., non-linearity). A good residual plot should show no discernible pattern.")

    print(f"\n--- End of Analysis for {dataset_name} ---")

# --- Run Analysis for each dataset ---

# Step 1: Replace mock data with actual file loading
# For ethylene_methane.txt
try:
    # Assuming your file is space-separated, comma-separated, or tab-separated
    # Adjust `sep` parameter if your delimiter is different
    df_methane_actual = pd.read_csv(FILE_PATH_METHANE, sep='\s+') # Example for space-separated
    print(f"\nSuccessfully loaded {FILE_PATH_METHANE}")
```

```
    analyze_dataset(df_methane_actual.copy(), "Ethylene_Methane Dataset") # Use .copy() to avoid modifying original df
except FileNotFoundError:
    print(f"\nWarning: {FILE_PATH_METHANE} not found. Using mock data for Ethylene_Methane analysis.")
    analyze_dataset(df_methane.copy(), "Ethylene_Methane Dataset")
except Exception as e:
    print(f"\nError loading {FILE_PATH_METHANE}: {e}. Using mock data for Ethylene_Methane analysis.")
    analyze_dataset(df_methane.copy(), "Ethylene_Methane Dataset")

# For ethylene_CO.txt
try:
    df_co_actual = pd.read_csv(FILE_PATH_CO, sep='\s+') # Example for space-separated
    print(f"\nSuccessfully loaded {FILE_PATH_CO}")
    analyze_dataset(df_co_actual.copy(), "Ethylene_CO Dataset")
except FileNotFoundError:
    print(f"\nWarning: {FILE_PATH_CO} not found. Using mock data for Ethylene_CO analysis.")
    analyze_dataset(df_co.copy(), "Ethylene_CO Dataset")
except Exception as e:
    print(f"\nError loading {FILE_PATH_CO}: {e}. Using mock data for Ethylene_CO analysis.")
    analyze_dataset(df_co.copy(), "Ethylene_CO Dataset")
```

```
/tmp/ipython-input-2264976319.py:242: SyntaxWarning: invalid escape sequence '\s'
  df_methane_actual = pd.read_csv(FILE_PATH_METHANE, sep='\s+') # Example for space-separated
/tmp/ipython-input-2264976319.py:255: SyntaxWarning: invalid escape sequence '\s'
  df_co_actual = pd.read_csv(FILE_PATH_CO, sep='\s+') # Example for space-separated
--- Mock df_methane Head ---
   Methane_Concentration  Temperature_C  Humidity_Percent  Pressure_kPa  \
0              87.454012      10.942876         62.101582     91.033634
1             145.071431      29.092312         34.206998    100.627093
2             123.199394      19.430679         38.081436    100.812702
3             109.865848      25.257121         74.927709    102.748598
4              65.601864      37.226994         60.321453    104.521827

   Ethylene_Concentration
0               57.773804
1              121.696768
2               82.990932
3              119.513720
4               83.700041

--- Mock df_methane Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Methane_Concentration   95 non-null     float64
 1   Temperature_C           97 non-null     float64
 2   Humidity_Percent        100 non-null    float64
 3   Pressure_kPa            100 non-null    float64
 4   Ethylene_Concentration  100 non-null    float64
dtypes: float64(5)
memory usage: 4.0 KB


--- Mock df_co Head ---
   CO_Concentration  Light_Intensity_Lux  Flow_Rate_LPM  \
0         17.741580           436.370464       5.543507
1          9.743868           622.519192       1.005601
2         19.310274           832.271807       3.780909
3         15.430657           103.346362       4.783179
4         13.208093           567.310767       2.293442
```

```
   Ethylene_Concentration
0               17.177199
1               14.373411
2               30.698374
3               16.551221
4               32.825451

--- Mock df_co Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   CO_Concentration       93 non-null     float64
 1   Light_Intensity_Lux    100 non-null    float64
 2   Flow_Rate_LPM          100 non-null    float64
 3   Ethylene_Concentration  100 non-null    float64
dtypes: float64(4)
memory usage: 3.3 KB


Successfully loaded /content/ethylene_methane.txt

--- Analyzing Dataset: Ethylene_Methane Dataset ---

1. Data Understanding & Preprocessing

Initial Data Info:
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 4178504 entries, (np.float64(0.0), np.float64(0.0), np.float64(0.0), np.float64(-41.98), np.float64(2067.64), np.float64(-37.13), np.float64(2.28)) to (np.float64(41790.19), np.floa
Data columns (total 12 columns):
 #   Column      Dtype
---  ------      -----
 0   Time        float64
 1   (seconds),  float64
 2   Methane     float64
 3   conc        float64
 4   (ppm),      float64
 5   Ethylene    float64
 6   conc.1      float64
 7   (ppm),.1    float64
 8   sensor      float64
 9   readings    float64
 10  (16         float64
 11  channels)   float64
dtypes: float64(12)
memory usage: 600.2 MB
```

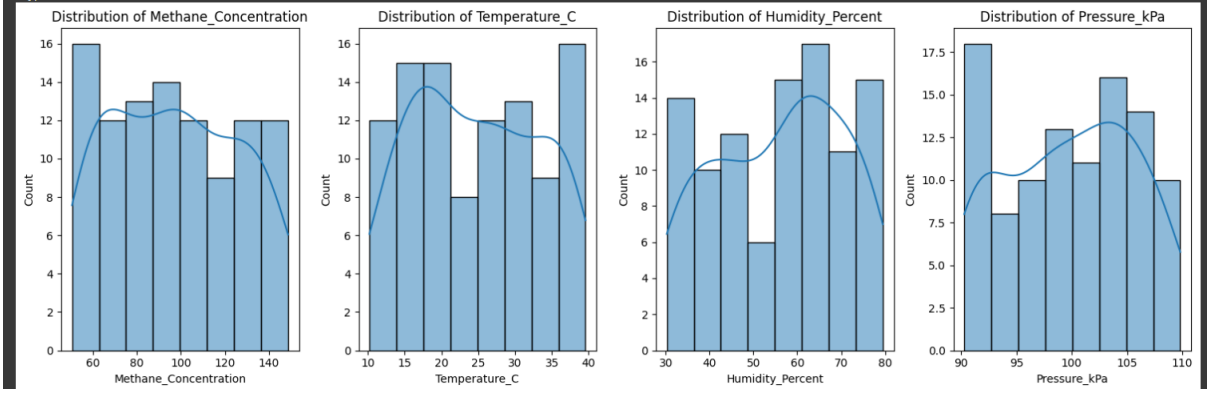
```
dtypes: float64(12)
memory usage: 600.2 MB

Initial Data Description:
                Time    (seconds),      Methane          conc       (ppm),  \
count  4.178504e+06  4.178504e+06  4.178504e+06  4.178504e+06  4.178504e+06
mean   1.863258e+03  2.386329e+03  2.689914e+03  2.978962e+03  3.541804e+03
std    1.104965e+03  1.425092e+03  1.102780e+03  1.229724e+03  2.607058e+02
min   -1.268000e+01 -4.198000e+01 -1.528000e+01 -1.187000e+01  2.976530e+03
25%    8.197600e+02  1.061500e+03  1.533260e+03  1.660440e+03  3.344630e+03
50%    1.393180e+03  1.688830e+03  2.785620e+03  3.136350e+03  3.481370e+03
75%    2.813350e+03  3.605260e+03  3.610690e+03  4.083020e+03  3.708470e+03
max    4.420840e+03  5.707530e+03  5.304140e+03  5.820370e+03  4.436430e+03

           Ethylene        conc.1     (ppm),.1        sensor      readings  \
count  4.178504e+06  4.178504e+06  4.178504e+06  4.178504e+06  4.178504e+06
mean   2.823842e+03  2.301598e+03  2.024606e+03  1.687490e+03  1.806279e+03
std    2.002925e+02  9.875315e+02  8.246309e+02  9.865598e+02  1.102382e+03
min    2.367650e+03  6.898700e+02  5.817900e+02  4.433000e+02  4.239800e+02
25%    2.672740e+03  1.409680e+03  1.201660e+03  7.593200e+02  7.685600e+02
50%    2.782960e+03  2.174530e+03  2.072150e+03  1.266570e+03  1.332660e+03
75%    2.943730e+03  3.140840e+03  2.701560e+03  2.543380e+03  2.746090e+03
max    3.519340e+03  4.849910e+03  4.062070e+03  4.540980e+03  5.108820e+03

                (16     channels)
count  4.178504e+06  4.178504e+06
mean   2.309401e+03  1.862996e+03
std    9.652522e+02  7.658043e+02
min    7.713900e+02  5.890900e+02
25%    1.424110e+03  1.113520e+03
50%    2.165370e+03  1.839750e+03
75%    3.123440e+03  2.506360e+03
max    4.699970e+03  3.764370e+03

Missing Values Before Imputation:
Time          0
(seconds),    0
Methane       0
conc          0
(ppm),        0
Ethylene      0
conc.1        0
(ppm),.1      0
sensor        0
readings      0
(16           0
channels)     0
dtype: int64
```

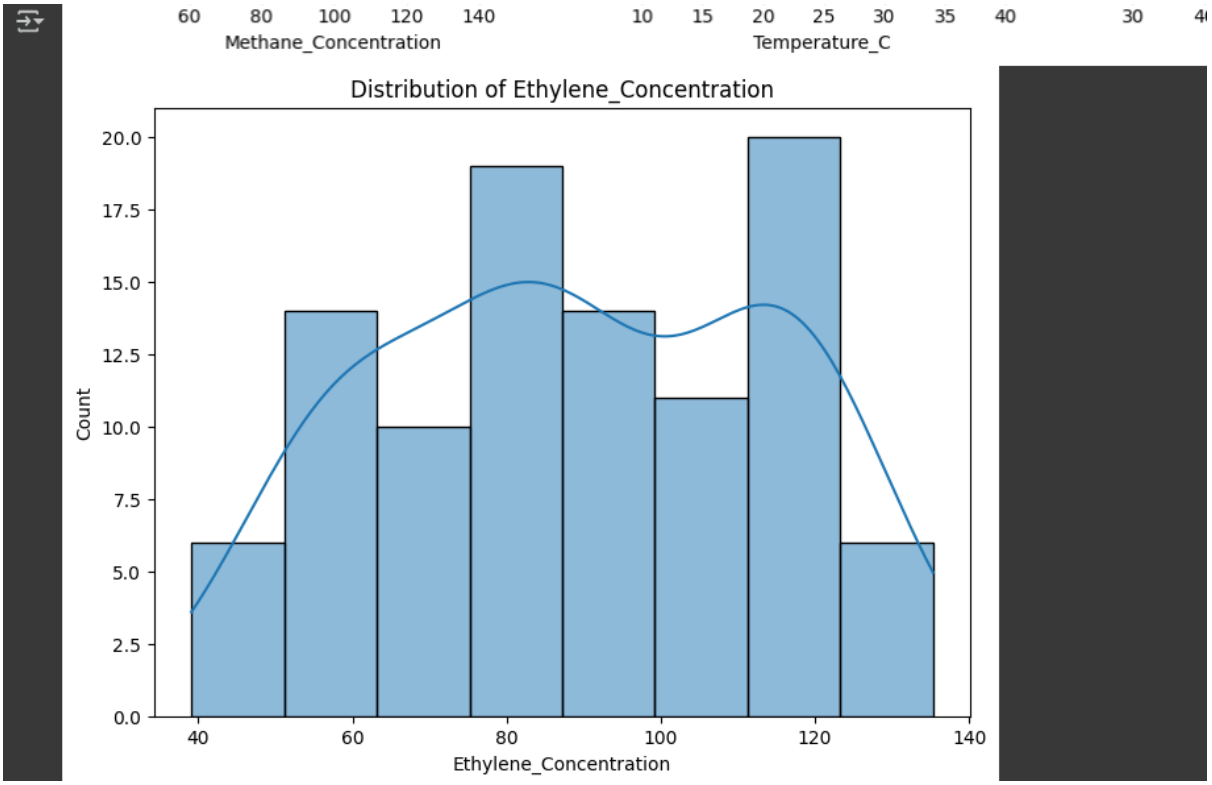
What can I help you build?

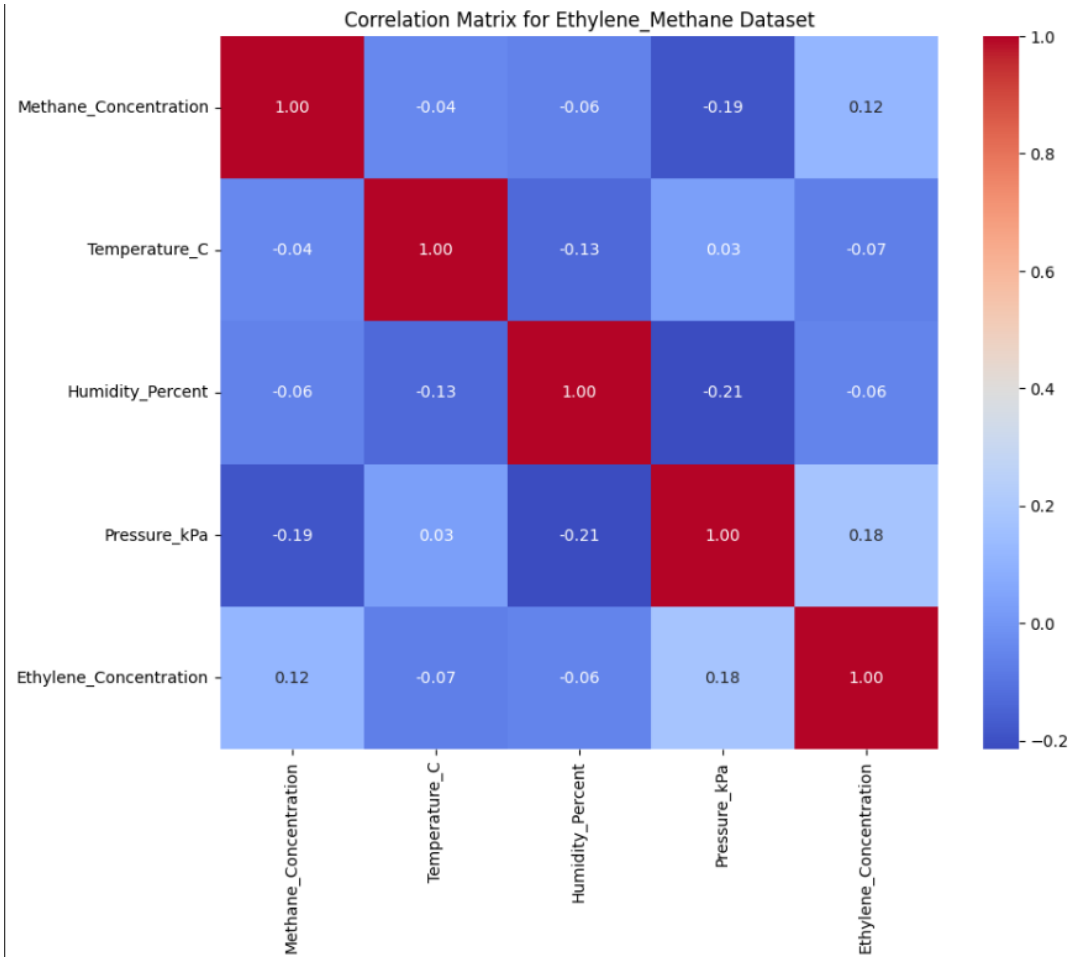
```
--- Analyzing Dataset: Ethylene_Methane Dataset ---

1. Data Understanding & Preprocessing

Initial Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Methane_Concentration   95 non-null     float64
 1   Temperature_C           97 non-null     float64
 2   Humidity_Percent        100 non-null    float64
 3   Pressure_kPa            100 non-null    float64
 4   Ethylene_Concentration  100 non-null    float64
dtypes: float64(5)
memory usage: 4.0 KB

Initial Data Description:
       Methane_Concentration  Temperature_C  Humidity_Percent  Pressure_kPa  \
count              95.000000      97.000000        100.000000    100.000000
mean               97.424045      24.744484         55.880067     99.822979
std                30.033847       8.756821         14.671312      5.869044
min                50.552212      10.208564         30.253079     90.287870
25%                69.041866      17.255569         43.843993     94.992298
50%                97.221493      25.080371         58.127747    100.194366
75%               124.638950      32.666534         67.618347    104.715550
max               148.688694      39.569514         79.502693    109.810103

       Ethylene_Concentration
count              100.000000
mean                89.233523
std                 25.078254
min                 39.105161
25%                 68.023823
50%                 87.804644
75%                112.609187
max                135.336428

Missing Values Before Imputation:
Methane_Concentration     5
Temperature_C             3
Humidity_Percent          0
Pressure_kPa              0
Ethylene_Concentration    0
dtype: int64

Missing Values After Imputation:
```

What can I help you build?

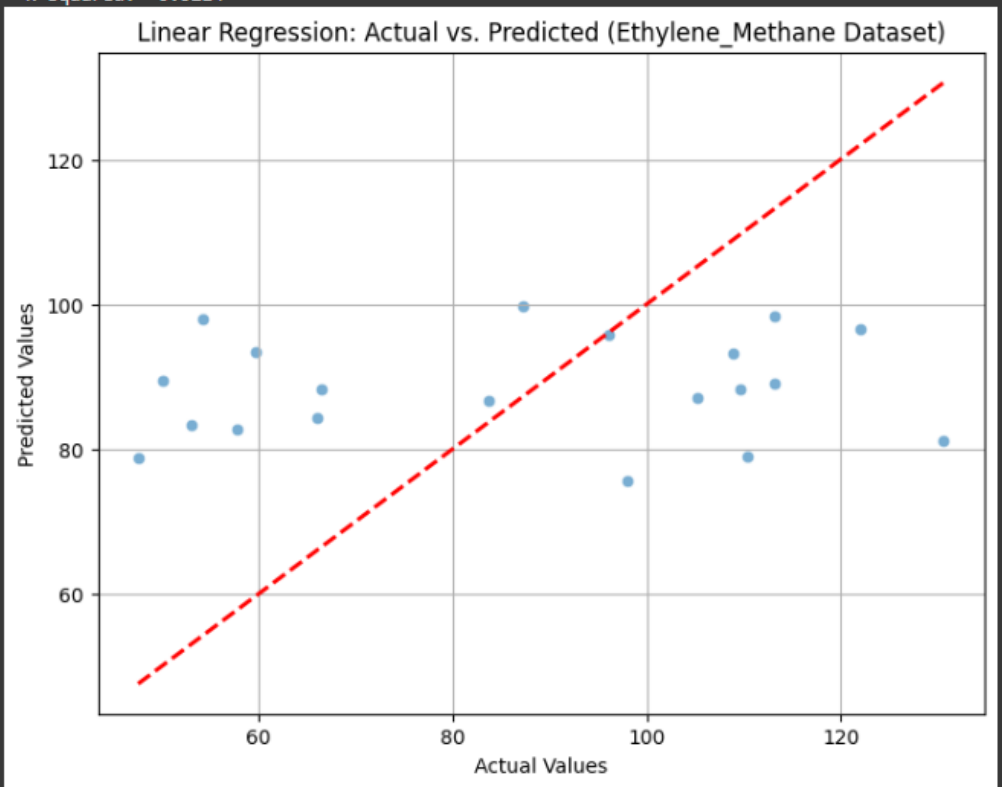
```
Missing Values After Imputation:
Methane_Concentration     0
Temperature_C             0
Humidity_Percent          0
Pressure_kPa              0
Ethylene_Concentration    0
dtype: int64
```

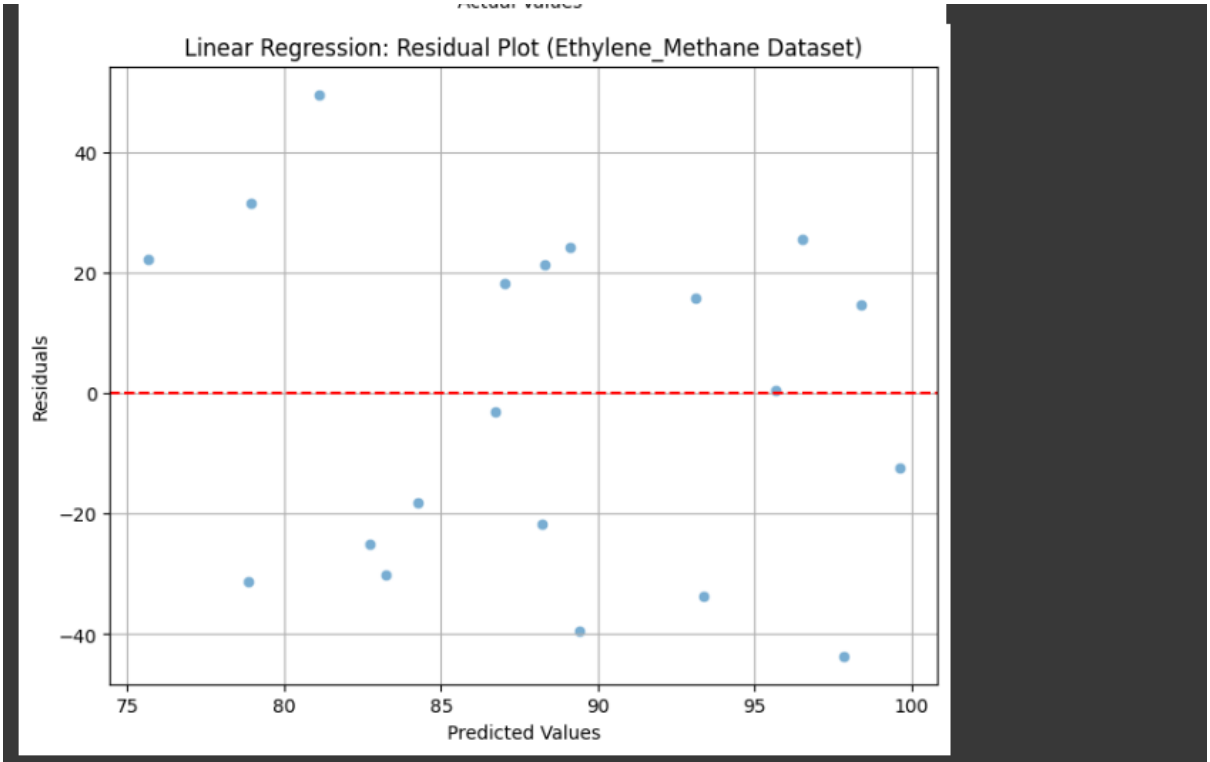




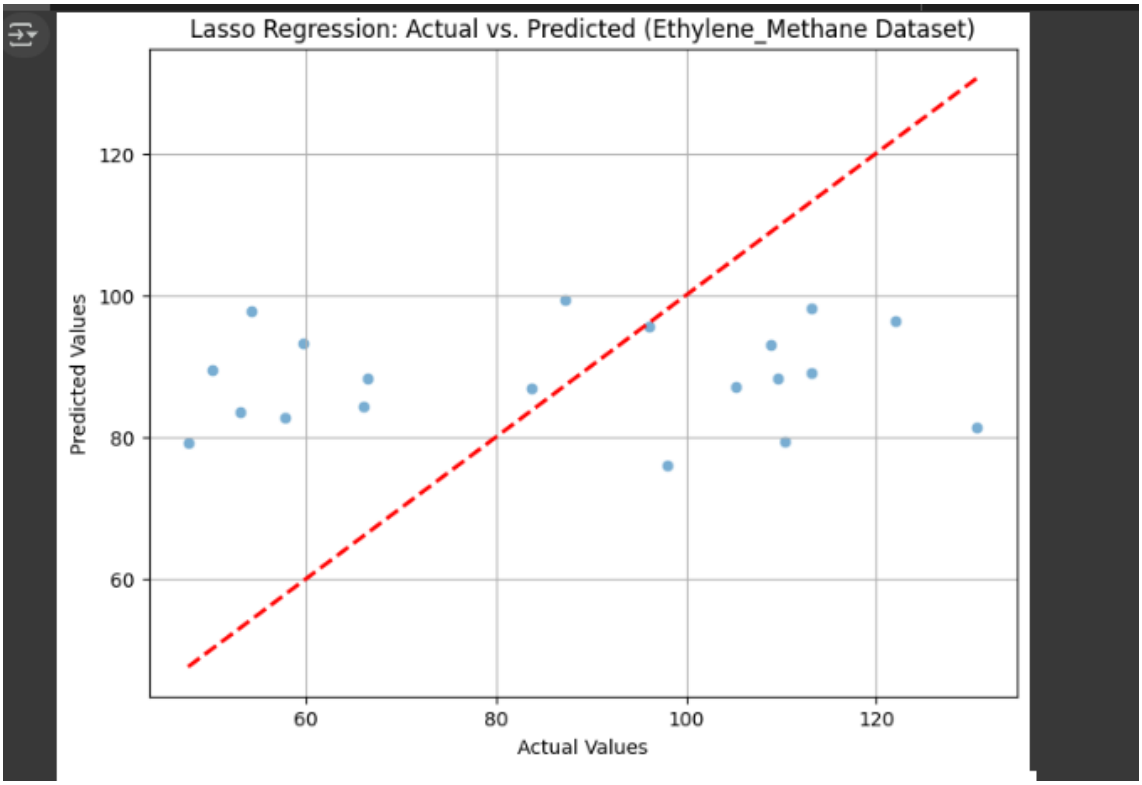

Correlation Matrix for Ethylene_Methane Dataset
Methane_Concentration
Temperature_C
Humidity_Percent
Pressure_kPa
Ethylene_Concentration
1.00
-0.04
-0.06
-0.19
0.12
-0.04
1.00
-0.13
0.03
-0.07
-0.06
-0.13
1.00
-0.21
-0.06
-0.19
0.03
-0.21
1.00
0.18
0.12
-0.07
-0.06
0.18
1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

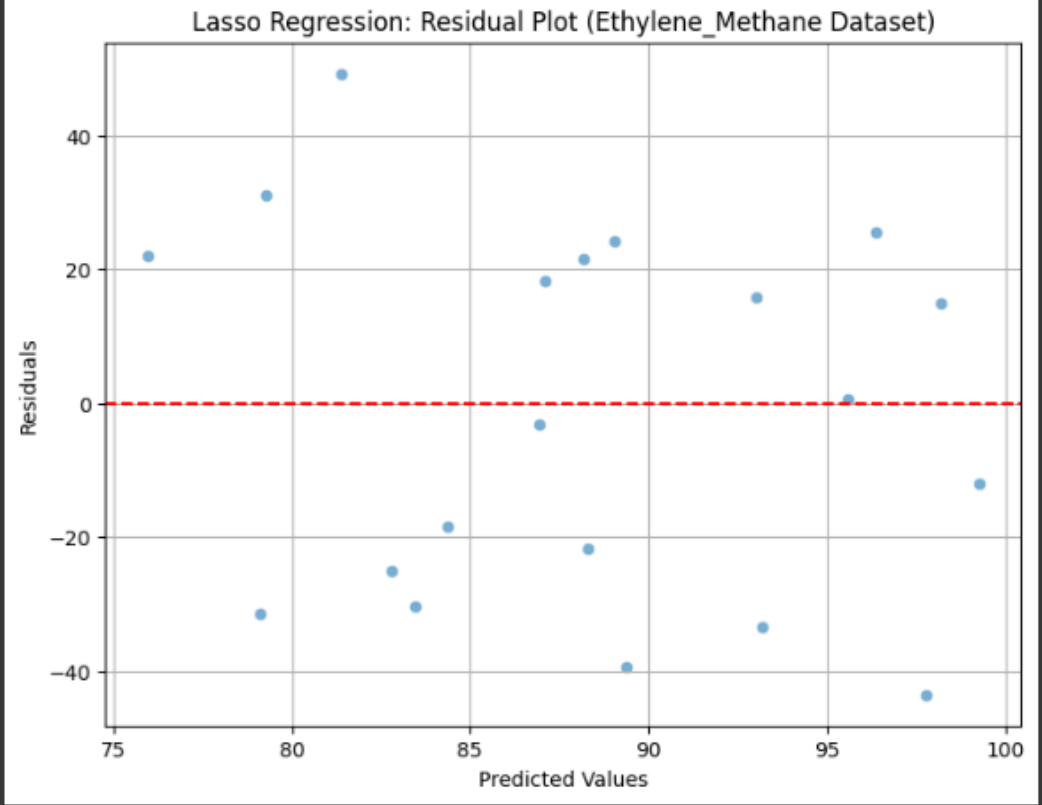
```
Train set size: 80 samples
Test set size: 20 samples

2. Model Implementation & 3. Performance Evaluation

--- Training Linear Regression ---
Linear Regression Performance:
  MAE: 24.1004
  MSE: 724.8669
  RMSE: 26.9234
  R-squared: -0.0214
```





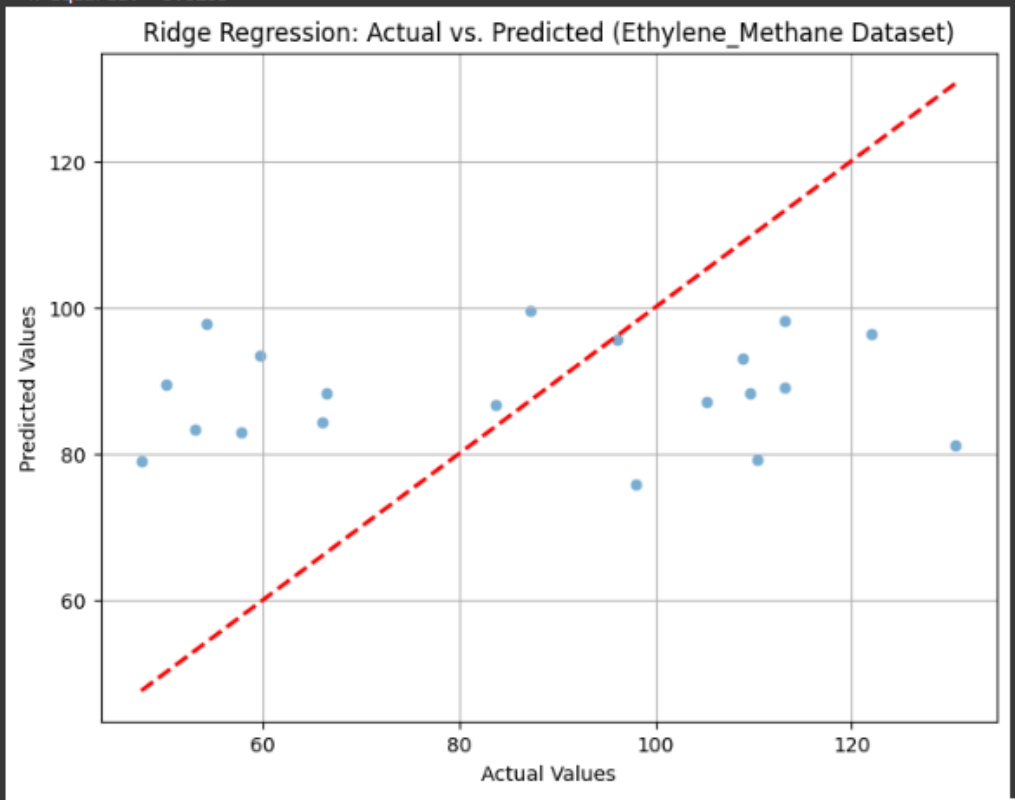

```
--- Training Lasso Regression ---
Lasso Regression Performance:
  MAE: 24.0977
  MSE: 723.2340
  RMSE: 26.8930
  R-squared: -0.0191
```

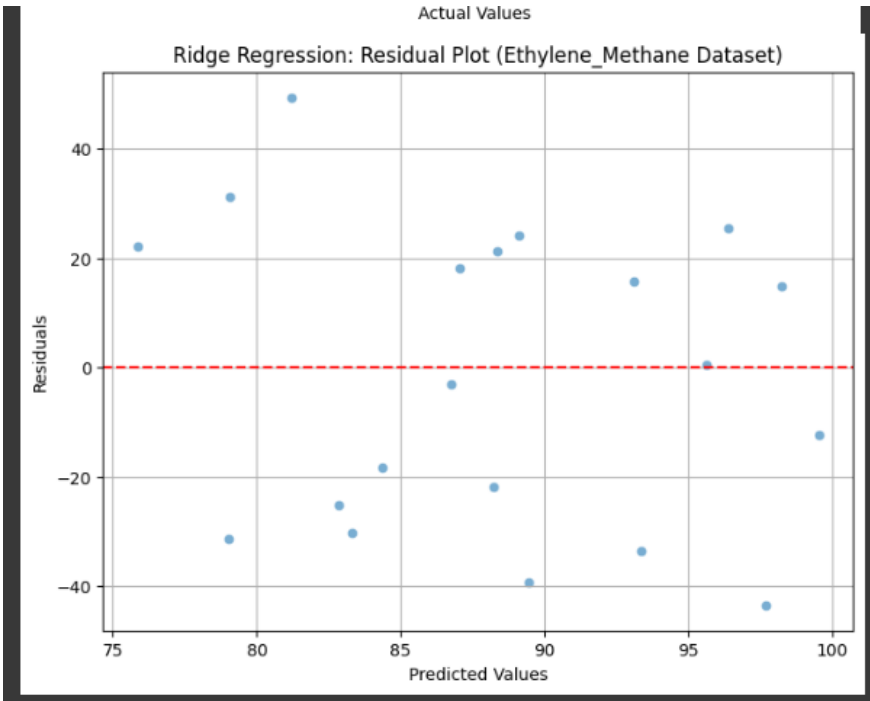




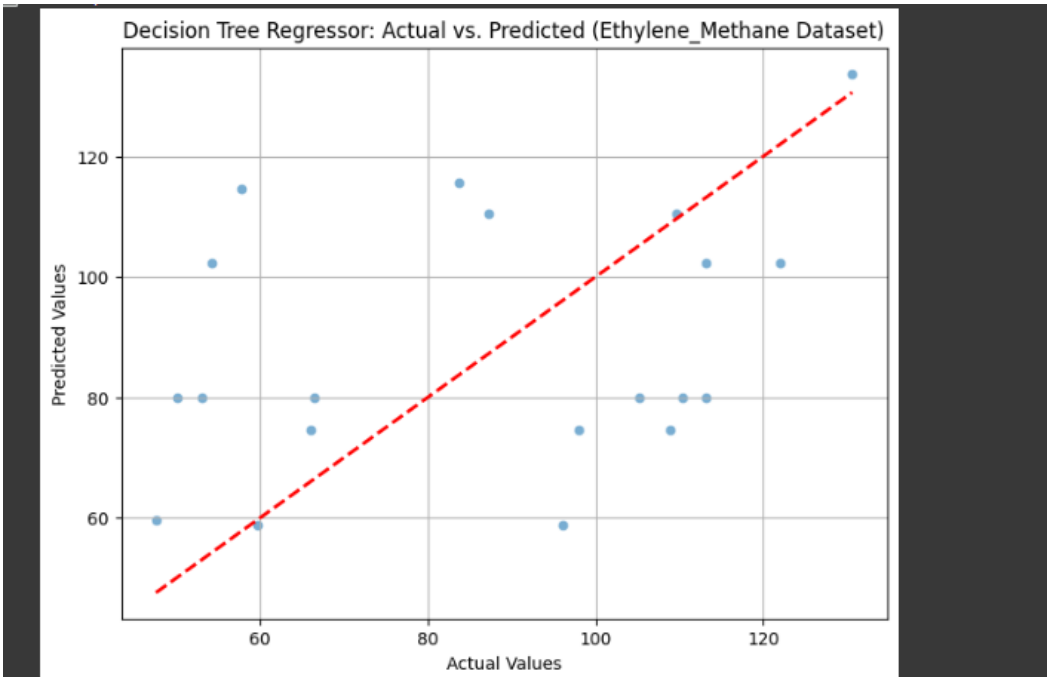

```
--- Training Ridge Regression ---
Ridge Regression Performance:
  MAE: 24.1033
  MSE: 724.2852
  RMSE: 26.9125
  R-squared: -0.0205
```

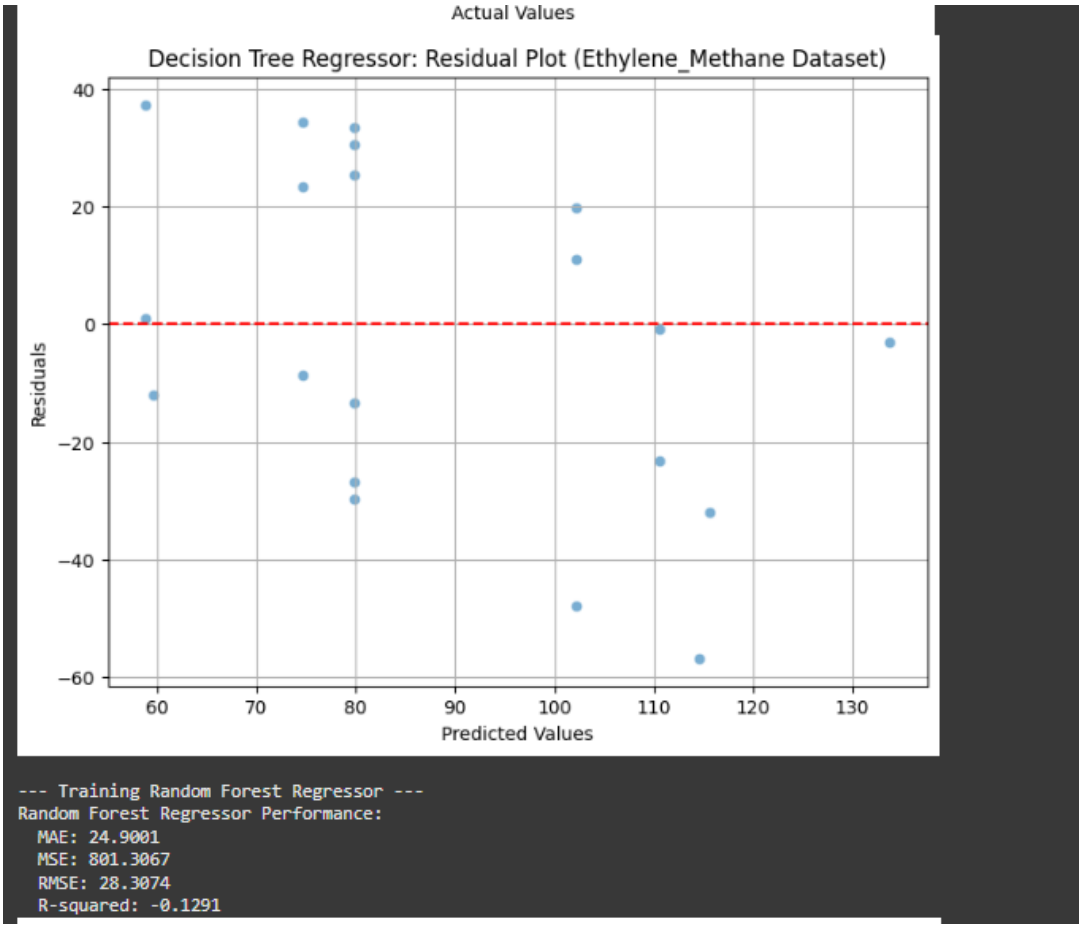
```
--- Training Ridge Regression ---
Ridge Regression Performance:
  MAE: 24.1033
  MSE: 724.2852
  RMSE: 26.9125
  R-squared: -0.0205
```



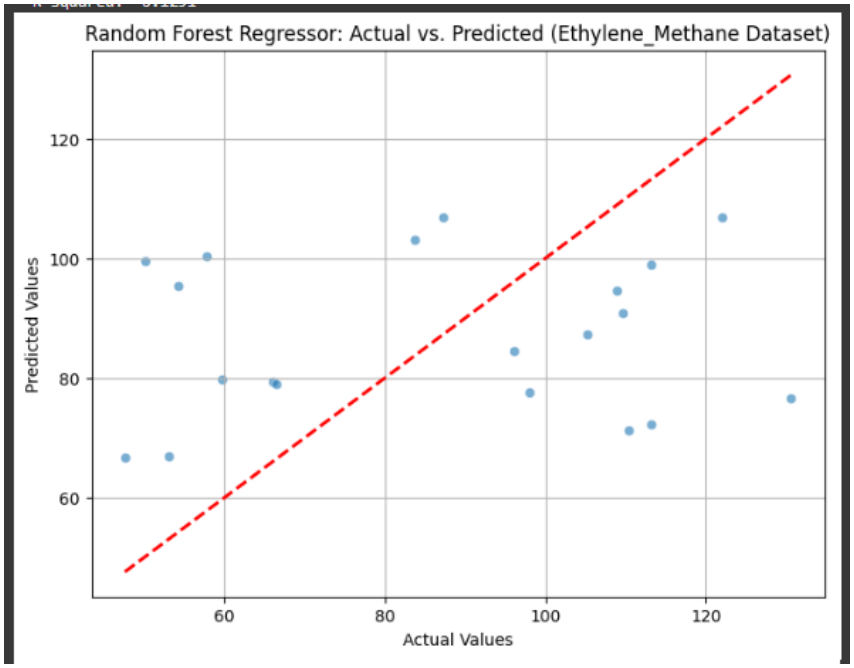


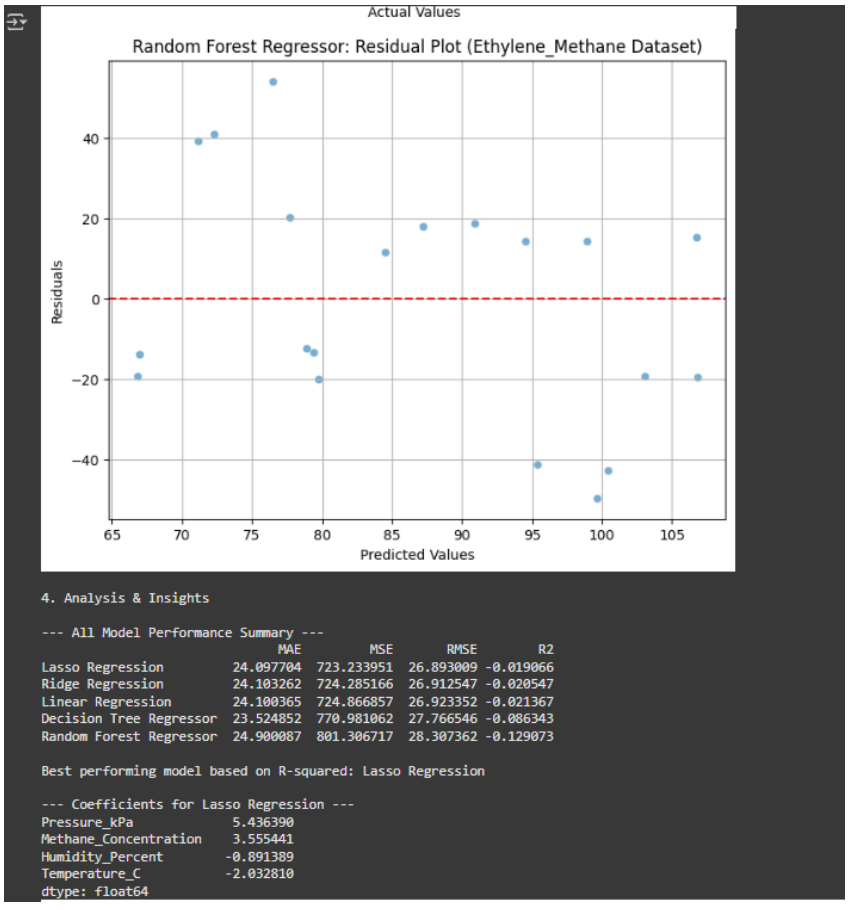

```
--- Training Decision Tree Regressor ---
Decision Tree Regressor Performance:
  MAE: 23.5249
  MSE: 770.9811
  RMSE: 27.7665
  R-squared: -0.0863
```



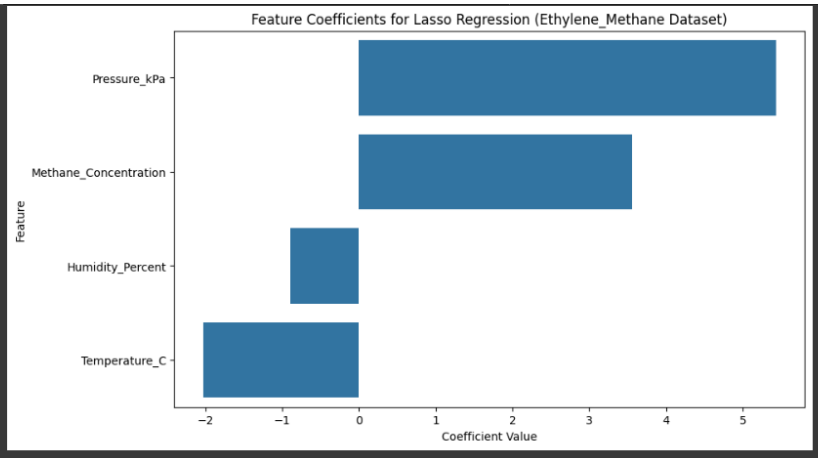



```
--- Training Random Forest Regressor ---
Random Forest Regressor Performance:
  MAE: 24.9001
  MSE: 801.3067
  RMSE: 28.3074
  R-squared: -0.1291
```






```
4. Analysis & Insights

--- All Model Performance Summary ---
                            MAE         MSE       RMSE        R2
Lasso Regression         24.097704  723.233951  26.893009 -0.019066
Ridge Regression         24.103262  724.285166  26.912547 -0.020547
Linear Regression        24.100365  724.866857  26.923352 -0.021367
Decision Tree Regressor  23.524852  770.981062  27.766546 -0.086343
Random Forest Regressor  24.900087  801.306717  28.307362 -0.129073

Best performing model based on R-squared: Lasso Regression

--- Coefficients for Lasso Regression ---
Pressure_kPa             5.436390
Methane_Concentration    3.555441
Humidity_Percent        -0.891389
Temperature_C           -2.032810
dtype: float64
```

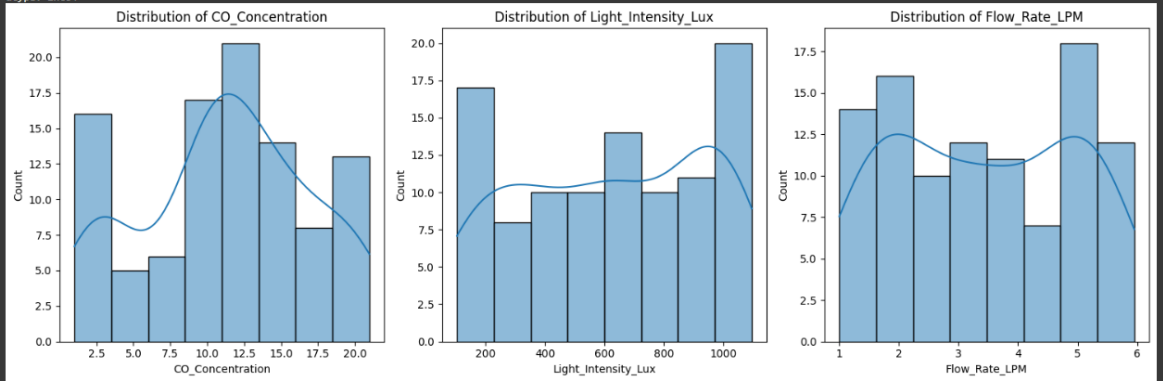


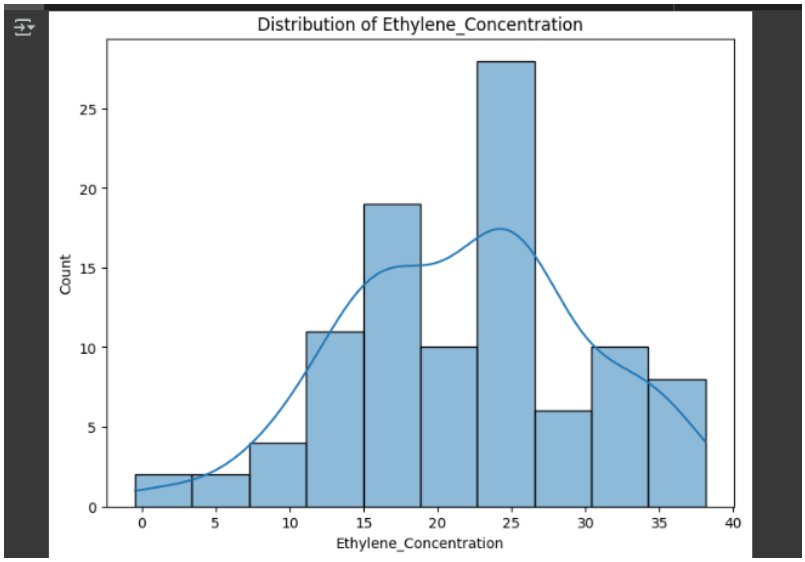

```
--- General Insights ---
The Lasso Regression generally performed best for Ethylene_Methane Dataset based on R-squared.
Consider further hyperparameter tuning (e.g., using GridSearchCV or RandomizedSearchCV) for each model to optimize performance.
The correlation matrix and feature importance plots give insights into which factors are most influential on Ethylene concentration.
Residual plots can indicate if the model is missing any patterns (e.g., non-linearity). A good residual plot should show no discernible pattern.

--- End of Analysis for Ethylene_Methane Dataset ---

Successfully loaded /content/ethylene_CO.txt

--- Analyzing Dataset: Ethylene_CO Dataset ---
```

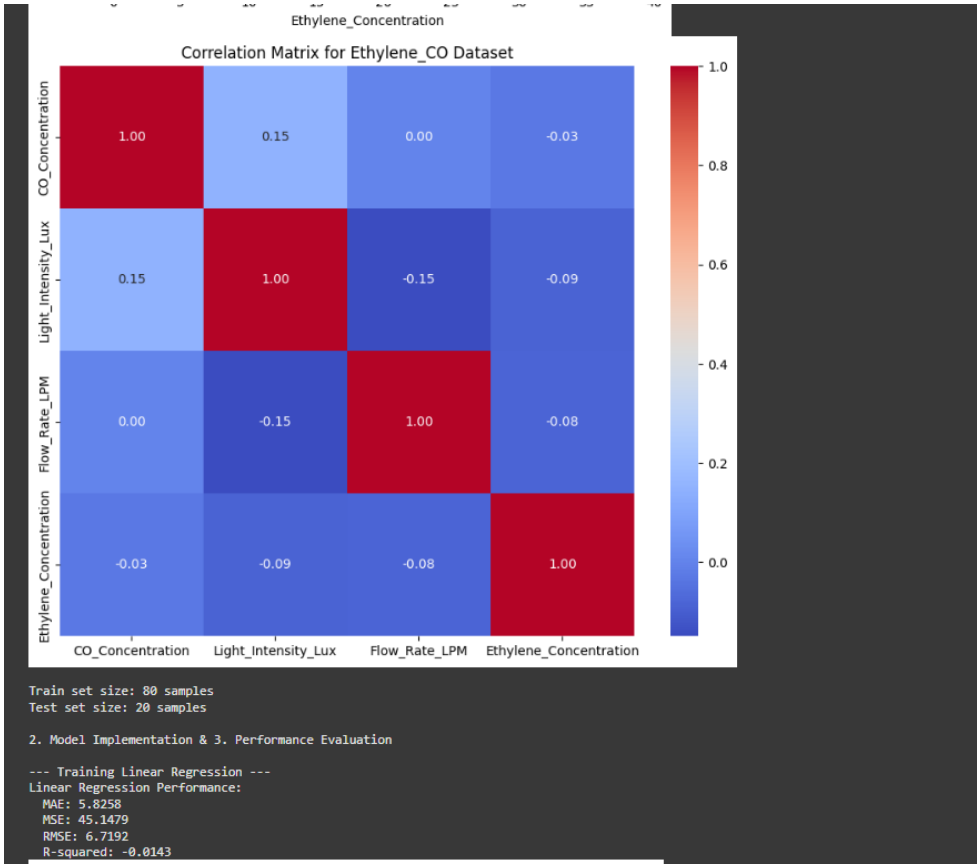
```
1. Data Understanding & Preprocessing

Initial Data Info:
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 4208261 entries, (np.float64(0.0), np.float64(0.0), np.float64(0.0), np.float64(-50.85), np.float64(-1.95), np.float64(-41.82), np.float64(1.3)) to (np.float64(42087.55), np.float64(0.0), np.float64(0.0), np.float64(1130.23), np.fl
Data columns (total 12 columns):
 #   Column       Dtype
---  ------       -----
 0   Time         float64
 1   (seconds),   float64
 2   CO           float64
 3   conc         float64
 4   (ppm),       float64
 5   Ethylene     float64
 6   conc.1       float64
 7   (ppm),.1     float64
 8   sensor       float64
 9   readings     float64
 10  (16          float64
 11  channels)    float64
dtypes: float64(12)
memory usage: 612.7 MB

Initial Data Description:
               Time    (seconds),            CO          conc       (ppm),  \
count  4.208261e+06  4.208261e+06  4.208261e+06  4.208261e+06  4.208261e+06
mean   1.894957e+03  2.214692e+03  5.086639e+03  5.385428e+03  1.186241e+03
std    4.395373e+02  4.353987e+02  1.966451e+03  2.072383e+03  4.531719e+02
min   -1.219000e+01 -4.214000e+01 -2.322000e+01 -1.382000e+01  1.597400e+02
25%    1.503460e+03  1.856650e+03  3.241950e+03  3.448620e+03  9.343300e+02
50%    1.984550e+03  2.272780e+03  5.224880e+03  5.537910e+03  1.140320e+03
75%    2.262740e+03  2.566470e+03  6.719280e+03  7.104200e+03  1.543660e+03
max    2.904040e+03  3.582110e+03  9.883580e+03  1.052104e+04  5.513995e+04

            Ethylene       conc.1      (ppm),.1        sensor      readings  \
count  4.208261e+06  4.208261e+06  4.208261e+06  4.208261e+06  4.208261e+06
mean   1.219252e+03  4.871079e+03  3.926250e+03  9.281877e+02  1.035772e+03
std    4.308765e+02  1.707681e+03  1.364496e+03  1.677561e+02  1.741085e+02
min    2.153100e+02  8.622200e+02  8.654900e+02  6.095700e+02  6.738900e+02
25%    9.419200e+02  3.435680e+03  2.728930e+03  8.083400e+02  9.102500e+02
50%    1.176600e+03  5.016130e+03  4.120240e+03  9.298600e+02  1.060700e+03
75%    1.557780e+03  6.126830e+03  4.967480e+03  1.033490e+03  1.161370e+03
max    5.066950e+04  9.626260e+03  9.762620e+03  2.454402e+04  2.142068e+04

              (16    channels)
count  4.208261e+06  4.208261e+06
mean   5.289323e+03  4.309685e+03
std    1.984057e+03  1.584038e+03
min    5.467300e+02  4.885800e+02
25%    3.642120e+03  2.805070e+03
50%    5.320550e+03  4.543040e+03
75%    6.739320e+03  5.565070e+03
max    1.066530e+04  7.874490e+03
```

```
                (16     channels)
count  4.208261e+06  4.208261e+06
mean   5.289323e+03  4.309685e+03
std    1.984057e+03  1.584038e+03
min    5.467300e+02  4.885800e+02
25%    3.642120e+03  2.805070e+03
50%    5.320550e+03  4.543040e+03
75%    6.739320e+03  5.565070e+03
max    1.066530e+04  7.874490e+03

Missing Values Before Imputation:
Time          0
(seconds),    0
CO            0
conc          0
(ppm),        0
Ethylene      0
conc.1        0
(ppm),.1      0
sensor        0
readings      0
(16           0
channels)     0
dtype: int64

Error loading /content/ethylene_CO.txt: ['Ethylene_Concentration']. Using mock data for Ethylene_CO analysis.

--- Analyzing Dataset: Ethylene_CO Dataset ---

1. Data Understanding & Preprocessing

Initial Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CO_Concentration        93 non-null     float64
 1   Light_Intensity_Lux     100 non-null    float64
 2   Flow_Rate_LPM           100 non-null    float64
 3   Ethylene_Concentration  100 non-null    float64
dtypes: float64(4)
memory usage: 3.3 KB

Initial Data Description:
       CO_Concentration  Light_Intensity_Lux  Flow_Rate_LPM  \
count         93.000000           100.000000     100.000000
mean          11.233177           626.222201       3.430229
std            5.857866           317.076876       1.528956
min            1.004750           103.346362       1.005147
25%            7.291931           350.808204       1.992763
50%           11.448692           623.775346       3.350180
75%           15.620473           941.417685       4.854433
max           20.987070          1096.335776       5.950742
```

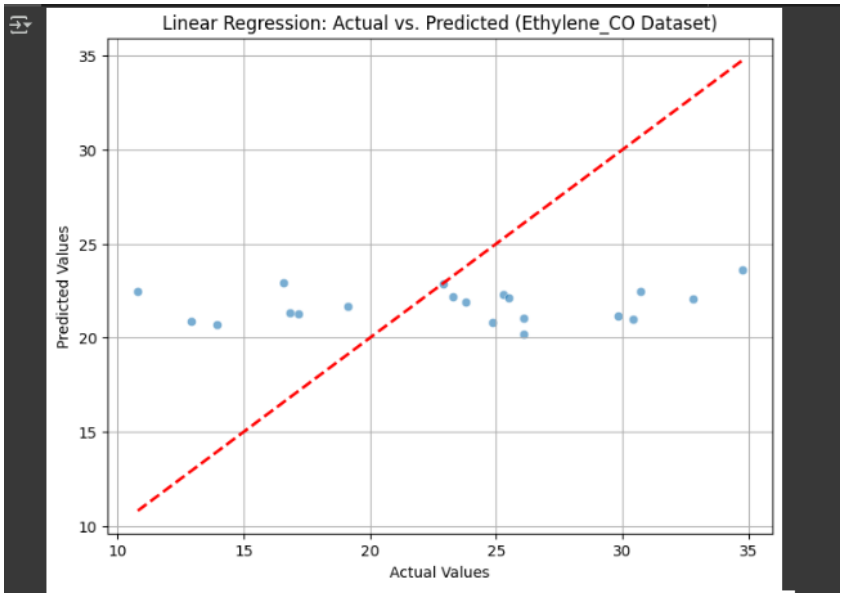
```
       Ethylene_Concentration
count              100.000000
mean                21.941004
std                  8.169020
min                 -0.442647
25%                 16.440078
50%                 23.022896
75%                 26.377408
max                 38.106330

Missing Values Before Imputation:
CO_Concentration          7
Light_Intensity_Lux       0
Flow_Rate_LPM             0
Ethylene_Concentration    0
dtype: int64

Missing Values After Imputation:
CO_Concentration          0
Light_Intensity_Lux       0
Flow_Rate_LPM             0
Ethylene_Concentration    0
dtype: int64
```

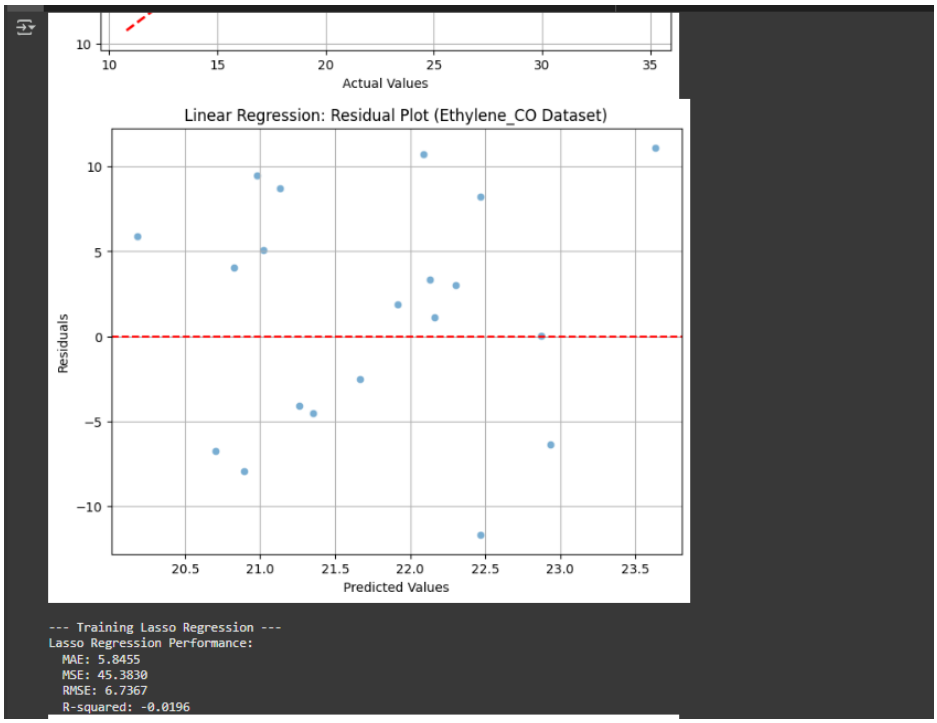




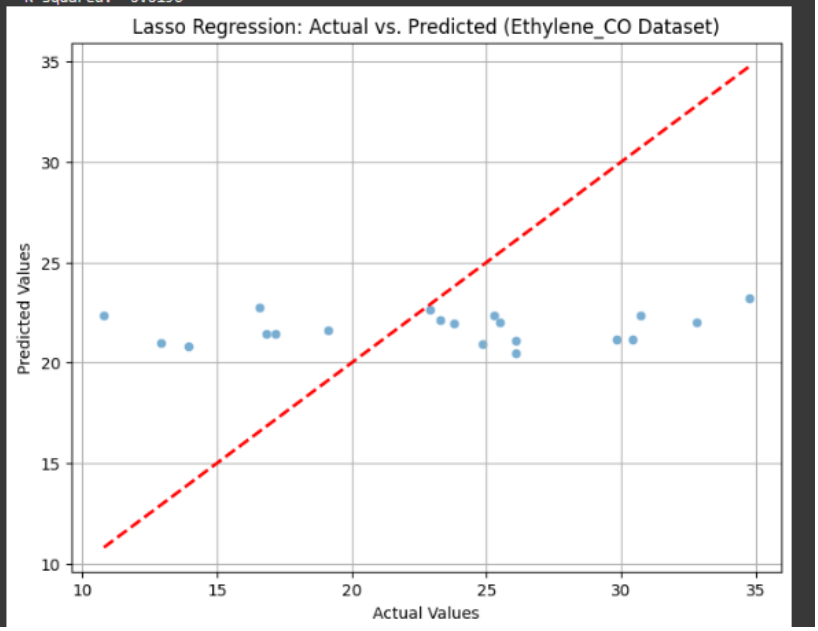


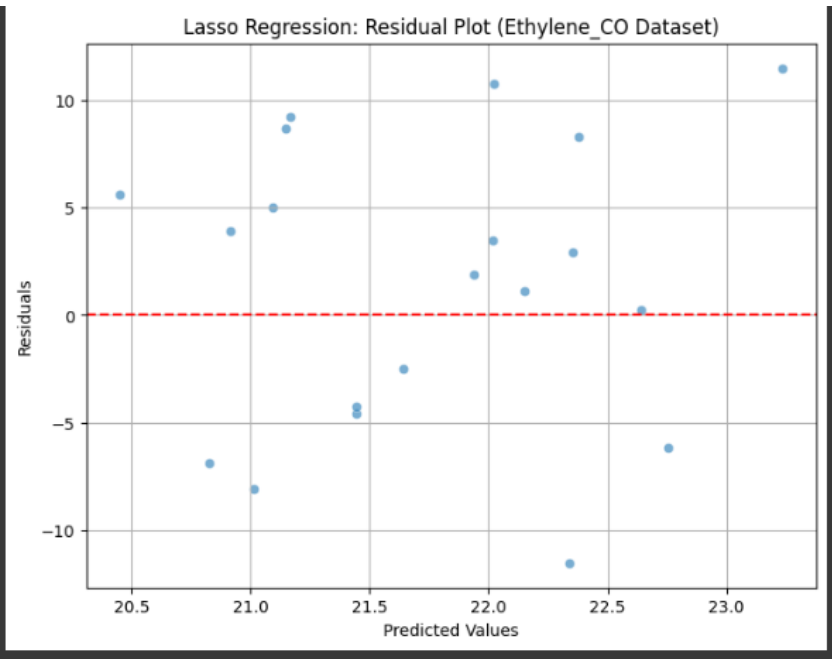

```
Train set size: 80 samples
Test set size: 20 samples

2. Model Implementation & 3. Performance Evaluation

--- Training Linear Regression ---
Linear Regression Performance:
  MAE: 5.8258
  MSE: 45.1479
  RMSE: 6.7192
  R-squared: -0.0143
```


Linear Regression: Actual vs. Predicted (Ethylene_CO Dataset)
Predicted Values
Actual Values
10
15
20
25
30
35

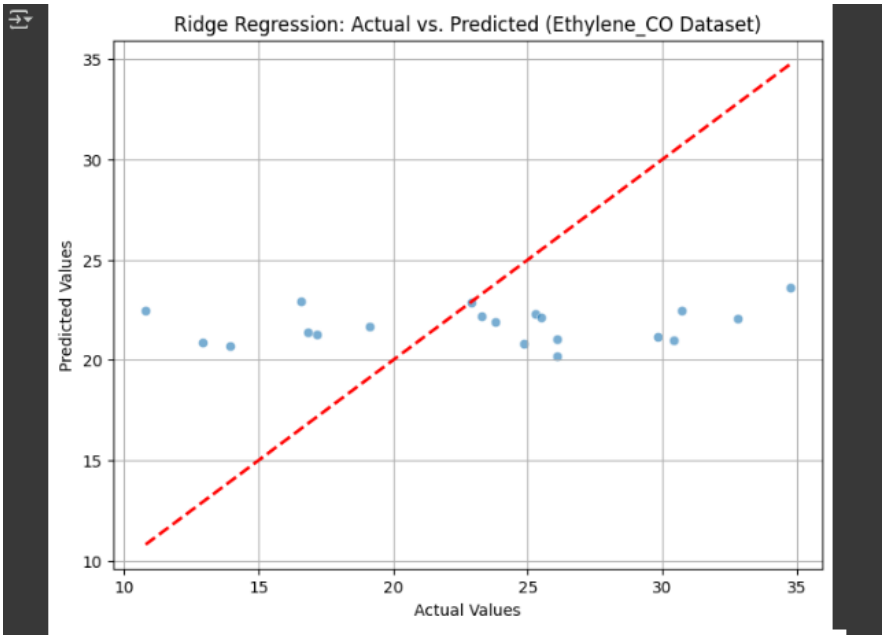

Actual Values
Linear Regression: Residual Plot (Ethylene_CO Dataset)
Residuals
Predicted Values
--- Training Lasso Regression ---
Lasso Regression Performance:
MAE: 5.8455
MSE: 45.3830
RMSE: 6.7367
R-squared: -0.0196

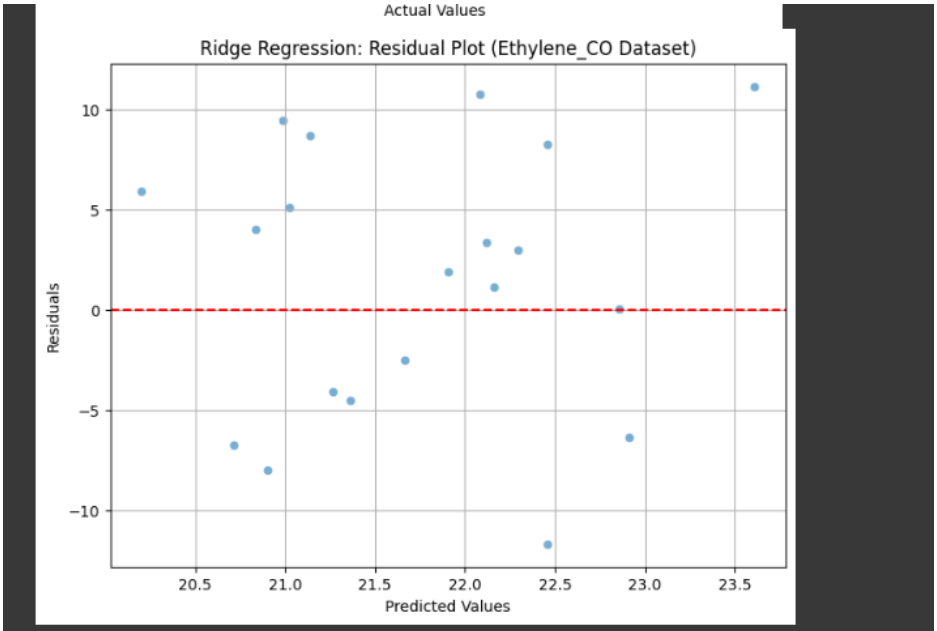
```
RMSE: 6.7367
R-squared: -0.0196
```





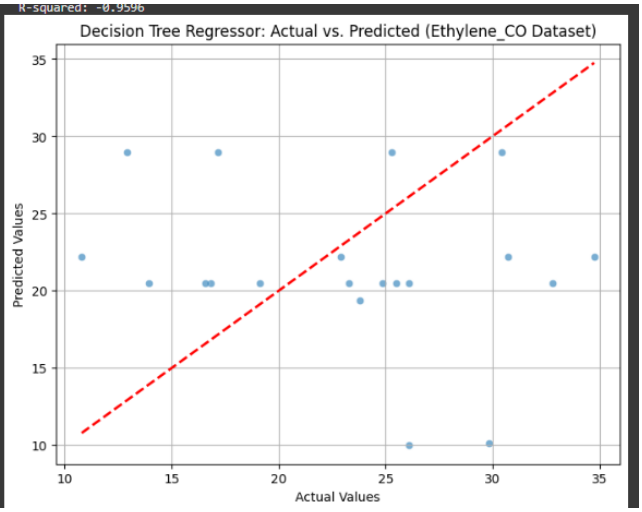

```
--- Training Ridge Regression ---
Ridge Regression Performance:
  MAE: 5.8277
  MSE: 45.1640
  RMSE: 6.7204
  R-squared: -0.0147
```

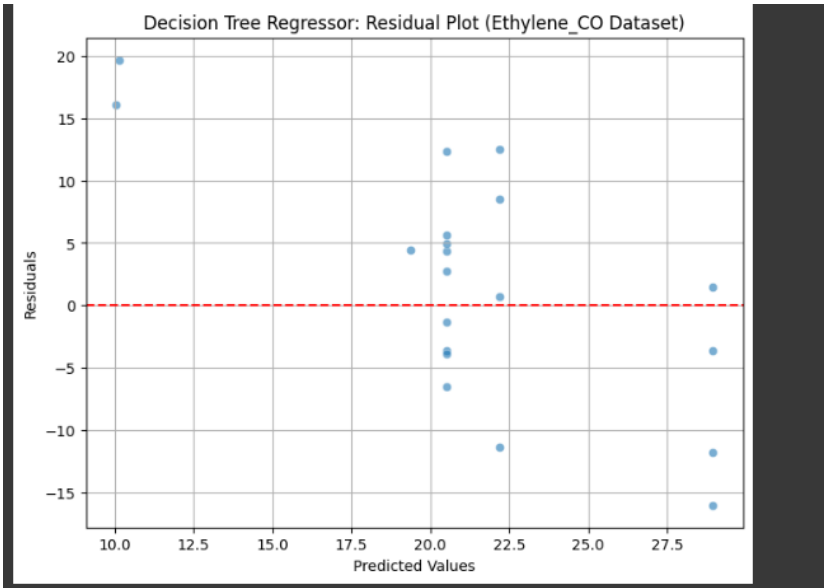




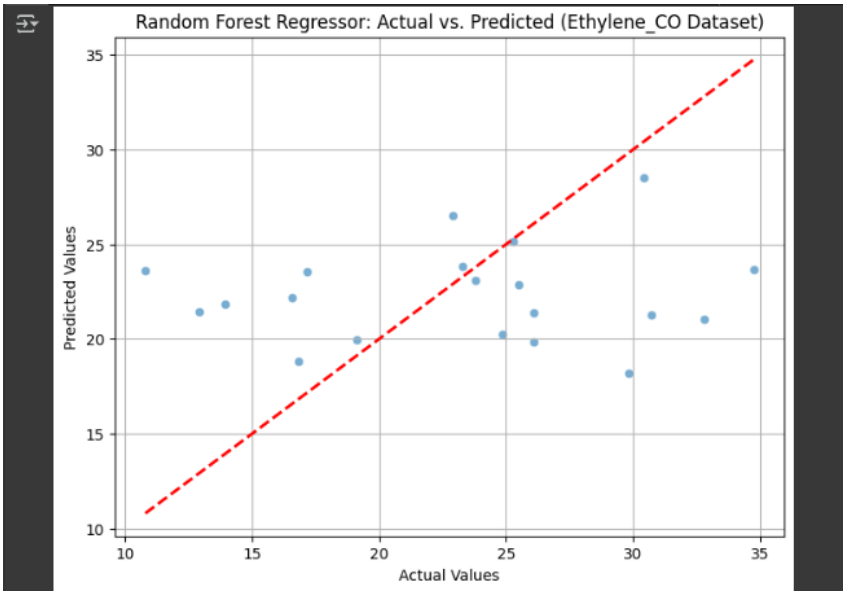

```
--- Training Decision Tree Regressor ---
Decision Tree Regressor Performance:
  MAE: 7.5893
  MSE: 87.2242
  RMSE: 9.3394
  R-squared: -0.9596
```

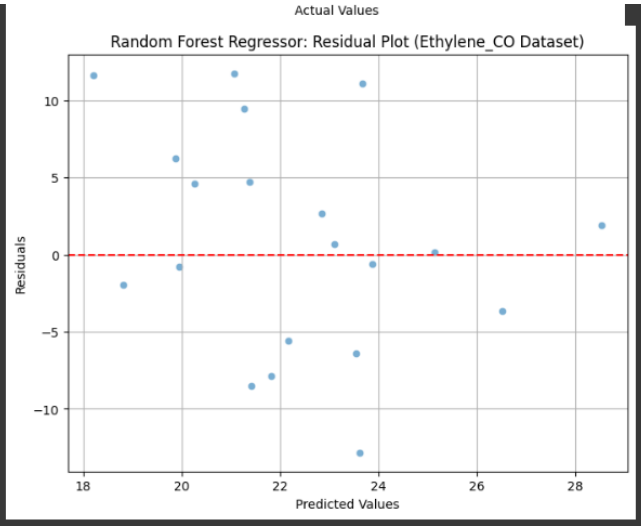
R-squared: -0.9596



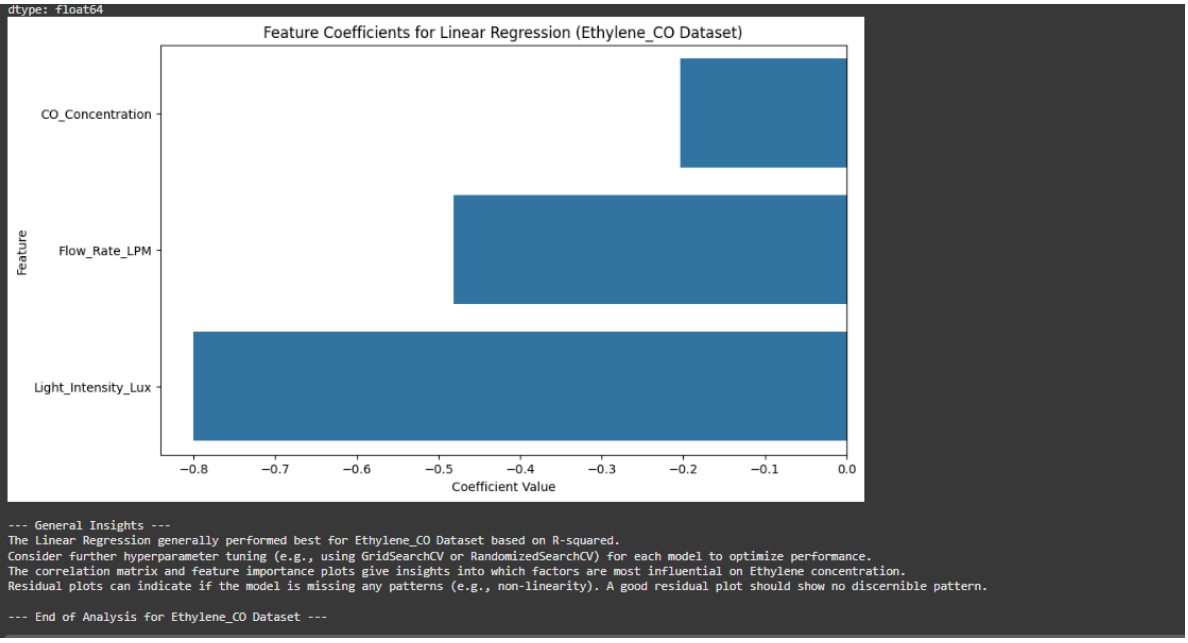



```
--- Training Random Forest Regressor ---
Random Forest Regressor Performance:
  MAE: 5.6473
  MSE: 48.3348
  RMSE: 6.9523
  R-squared: -0.0859
```






```
4. Analysis & Insights

--- All Model Performance Summary ---
                            MAE        MSE      RMSE        R2
Linear Regression      5.825760  45.147882  6.719217 -0.014326
Ridge Regression       5.827678  45.164028  6.720419 -0.014688
Lasso Regression       5.845541  45.383023  6.736692 -0.019609
Random Forest Regressor  5.647294  48.334754  6.952320 -0.085924
Decision Tree Regressor  7.589271  87.224190  9.339389 -0.959643

Best performing model based on R-squared: Linear Regression

--- Coefficients for Linear Regression ---
CO_Concentration      -0.203601
Flow_Rate_LPM         -0.480900
Light_Intensity_Lux   -0.800616
dtype: float64
```




```
--- General Insights ---
The Linear Regression generally performed best for Ethylene_CO Dataset based on R-squared.
Consider further hyperparameter tuning (e.g., using GridSearchCV or RandomizedSearchCV) for each model to optimize performance.
The correlation matrix and feature importance plots give insights into which factors are most influential on Ethylene concentration.
Residual plots can indicate if the model is missing any patterns (e.g., non-linearity). A good residual plot should show no discernible pattern.

--- End of Analysis for Ethylene_CO Dataset ---
```