Machine learning-based optimization for catalytic sulfur removal: Computational modeling and analysis of fuel purification for reduction of environmental impacts

Nandini Priya

210107055

Submission Date: April 25, 2024

Final Project submission

Course Name : Applications of Al and ML in chemical engineering

Course Code: CL653

Contents

1 Executive Summary

The project focuses on optimizing the catalytic sulfur removal process in petroleum refining to mitigate environmental impacts caused by sulfur compounds. The problem at hand involves improving the efficiency of the **Hydrodesulfurization (HDS) process**, which separates sulfur compounds from petroleum-based products. The proposed solution utilizes machine learning-based optimization techniques to enhance the performance of the HDS process.

The methodologies employed include the development and application of multiple machine learning models, namely **Multi-Linear Regression, Multi-layered Perceptron (MLP), Gaussian Process Regression (GPR), and Multi-Task Lasso (MTL).** These models are trained and optimized through hyperparameter tuning for improved accuracy and efficiency.

The project utilizes a dataset containing **input parameters such as temperature, pressure, catalyst dosage, and initial sulfur content**, along with **target outputs including sulfur concentration, SO2 emission percentage, and HDS cost**. By training the machine learning models on this dataset, the project aims to predict and optimize the target outputs based on the input parameters.

The expected outcomes of the project include highly accurate predictive models for sulfur concentration, emission percentage, and HDS cost. These models will enable process engineers to make informed decisions to optimize the HDS process, reducing environmental impacts and operational costs associated with sulfur removal from petroleum products. Additionally, the project provides insights into the relationships between input parameters and target outputs, facilitating further process optimization and design improvements. Overall, the project contributes to the advancement of fuel purification techniques and environmental sustainability in the petroleum industry.

2 Introduction

Background: In the realm of Chemical Engineering, the removal of sulfur compounds from petroleum-based products is a critical process due to both operational and environmental concerns. Sulfur compounds present in fuels can lead to the emission of harmful gases such as sulfur dioxide (SO2), contributing to air pollution and environmental degradation. Furthermore, sulfur compounds can also cause corrosion in refinery equipment and degrade the performance of catalysts, leading to increased operational costs and reduced efficiency in petroleum refining processes.

Importance of the Problem:Given the significant environmental and economic implications of sulfur compounds in petroleum products, optimizing the removal of sulfur through processes like Hydrodesulfurization (HDS) is paramount. Efficient sulfur removal not only reduces emissions of harmful pollutants but also enhances the quality and performance of refined fuels. Therefore, improving the performance of HDS processes holds great importance in the field of Chemical Engineering, particularly in the petroleum refining industry.

Problem Statement:The specific problem addressed in this project is the optimization of the HDS process to enhance the removal of sulfur compounds from petroleum-based products. The project aims to develop machine learning-based models to predict and optimize key parameters of the HDS process, such as sulfur concentration, SO2 emission percentage, and HDS cost. By leveraging computational modeling and analysis, the project seeks to improve the efficiency and effectiveness of sulfur removal, thereby reducing environmental impacts and operational costs associated with petroleum refining.

References:

The problem statement draws inspiration from the following reference article:
https://www.sciencedirect.com/science/article/pii/S2214157X23011413#bib12

Objectives:The main objectives of the project include:

1. Develop machine learning models, including Multi-Linear Regression, Multi-layered Perceptron (MLP), Gaussian Process Regression (GPR), and Multi-Task Lasso (MTL), for optimizing the HDS process.

2. Train and optimize the machine learning models using hyperparameter tuning to accurately predict sulfur concentration, SO2 emission percentage, and HDS cost based on input parameters.

3. Evaluate the performance of the developed models through statistical analysis and comparison with experimental data.

4. Provide insights into the relationships between input parameters and target outputs to facilitate further process optimization and design improvements.

5. Contribute to the advancement of fuel purification techniques and environmental sustainability in the petroleum industry through enhanced sulfur removal processes.

3 Methodology

Data Source: The dataset that I have used here has been generated synthetically using AI Tools such as ChatGPT .I have used the 40 point dataset(Appendix-1) given in the research paper as a reference to set the range for each variable to generate a 1000 points dataset. The below mentioned code generates a set of linearly spaced data bound in the intervals that resonate with the sample data. Additionally, 4% noise has been added to the linearly spaced data to generate the data points that can be used for development of the model and analyzing its performance.

```
# Define the percentage of noise (8%)
noise_percentage = 0.08

# Define ranges for various parameters
Temperature_min, Temperature_max = 50, 450
Pressure_min, Pressure_max = 20, 100
Dosage_min, Dosage_max = 0.2, 1
sulfur_min, sulfur_max = 2000, 4000
emission_min, emission_max = 0, 75
cost_min, cost_max = 10, 50

# Generate synthetic data with noise
data = {
    'Temperature (◦C)': np.linspace(Temperature_min, Temperature_max, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points
    'Pressure (bar)': np.linspace(Pressure_min, Pressure_max, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points)),
    'Dosage (g)': np.linspace(Dosage_min, Dosage_max, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points)),
    'Initial sulfur (ppm)': np.linspace(sulfur_min, sulfur_max, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points)),
    'Emission (%)': np.linspace(emission_min, emission_max, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points)),
    'HDS cost ($)': np.linspace(cost_min, cost_max, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points)),
    'Sulfur concentration (ppm)': np.linspace(1000, 1900, num_data_points) * (1 + noise_percentage * np.random.randn(num_data_points))
}



# Create a DataFrame with column order rearranged
df = pd.DataFrame(data, columns=['Temperature (◦C)', 'Pressure (bar)', 'Dosage (g)', 'Initial sulfur (ppm)', 'Sulfur concentration (ppm)',
              'Emission (%)', 'HDS cost ($)'])
```

```python
# Create a DataFrame with column order rearranged
df = pd.DataFrame(data, columns=['Temperature (°C)', 'Pressure (bar)', 'Dosage (g)', 'Initial sulfur (ppm)', 'Sulfur concentration (ppm)',
             'Emission (%)', 'HDS cost ($)'])

# Display the first few rows of the DataFrame
print(df.head())

# Save the DataFrame to a CSV file
df.to_csv('final_data.csv', index=False)
```

Data Preprocessing:

1. Handling null and missing values: Since the data is generated synthetically, there are no null or missing values. I have still added the code to check for null values.

```python
# Check for null values
null_values = nandini_df.isnull().sum()

# Display the number of null values in each column
print("Number of null values in each column:")
print(null_values)
```

```
Number of null values in each column:
Temperature (°C)              0
Pressure (bar)                0
Dosage (g)                    0
Initial sulfur (ppm)          0
Sulfur concentration (ppm)    0
Emission (%)                  0
HDS cost ($)                  0
```

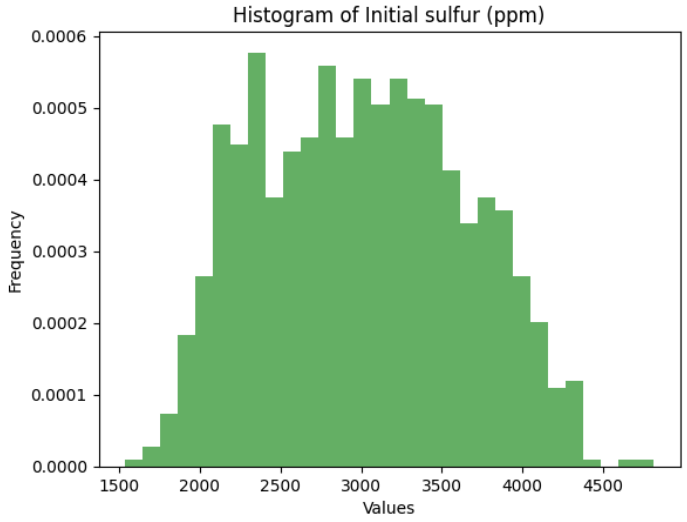
2. Outlier Detection and Removal:I have used the **IQR method** to detect outliers from the dataset. The detected outliers are then removed from the dataset.

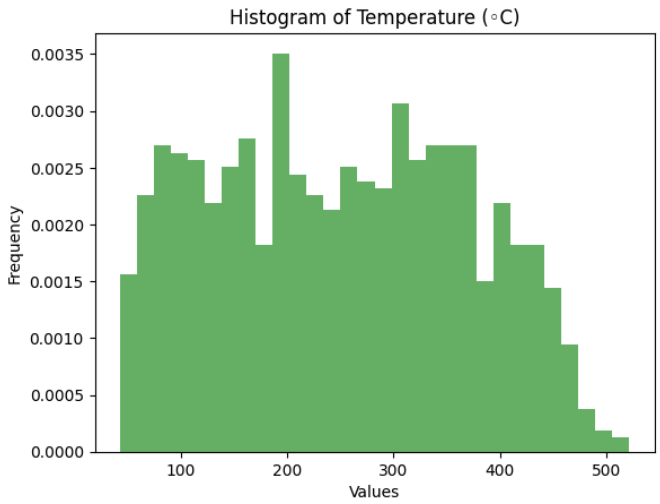
```python
# Define a function to detect outliers using the IQR method
def detect_outliers(df):
    outliers = pd.DataFrame(columns=df.columns)
    for col in df.columns:
        if df[col].dtype != 'object':
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR
            col_outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
            outliers = pd.concat([outliers, col_outliers])
    return outliers

# Detect outliers in the dataset
outliers = detect_outliers(nandini_df)

# Display the outliers
print("Outliers in the dataset:")
print(outliers)
```

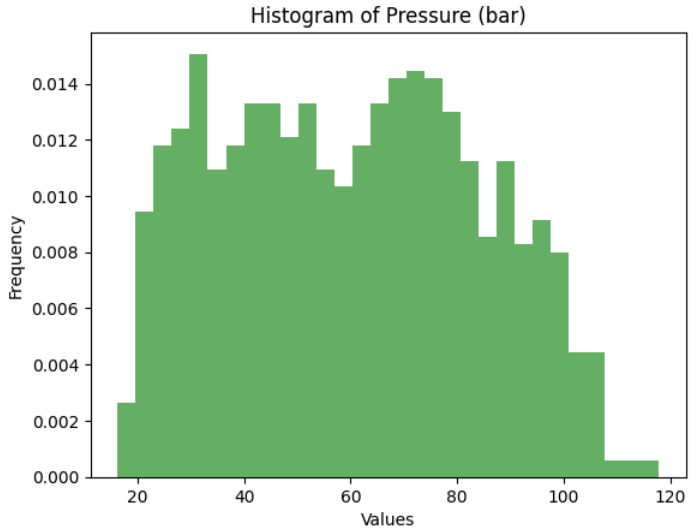
```
Outliers in the dataset:
     Temperature (°C)  Pressure (bar)  Dosage (g)  Initial sulfur (ppm)  \
953        388.394044       92.067413     1.00004           5264.426145

     Sulfur concentration (ppm)  Emission (%)  HDS cost ($)
953                 1799.999313     71.524729     48.995623
```

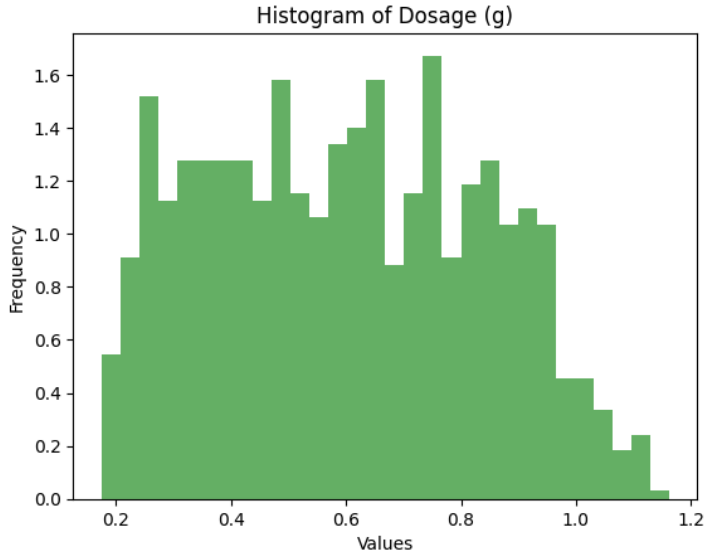
3. Visual Plots: I have drawn various plots such as histograms, QQ Plots and Box Plots to visually identify the patterns in the dataset, identify the relations between features and labels and draw proper conclusions.

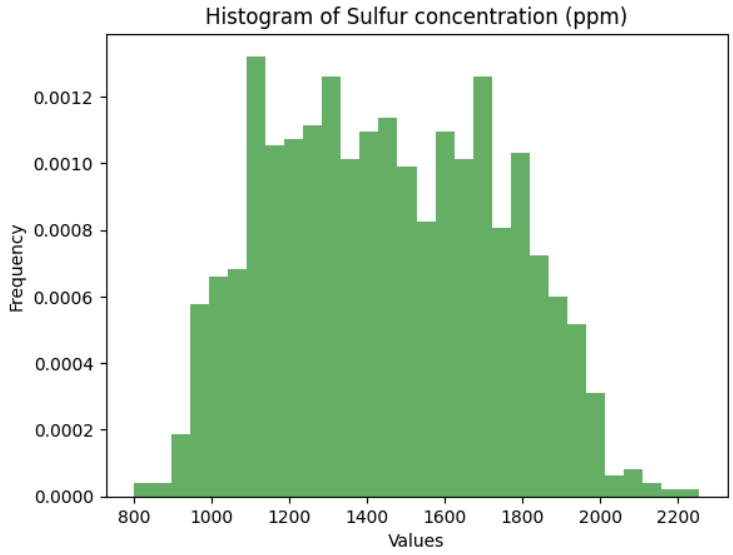
HISTOGRAM:

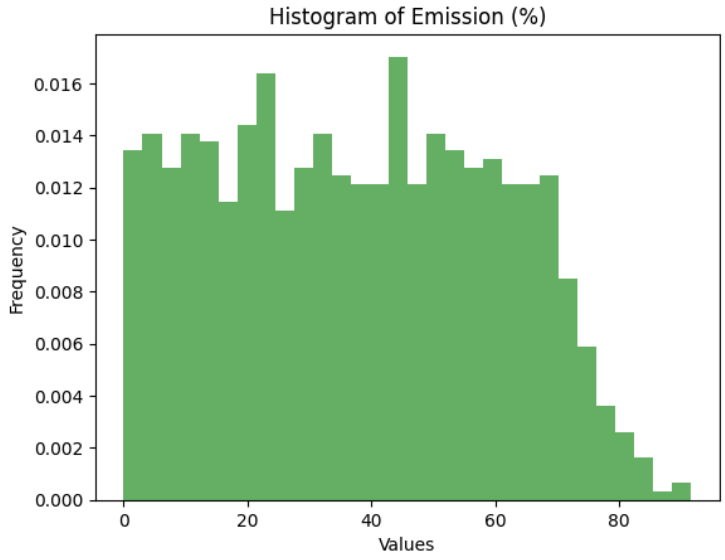


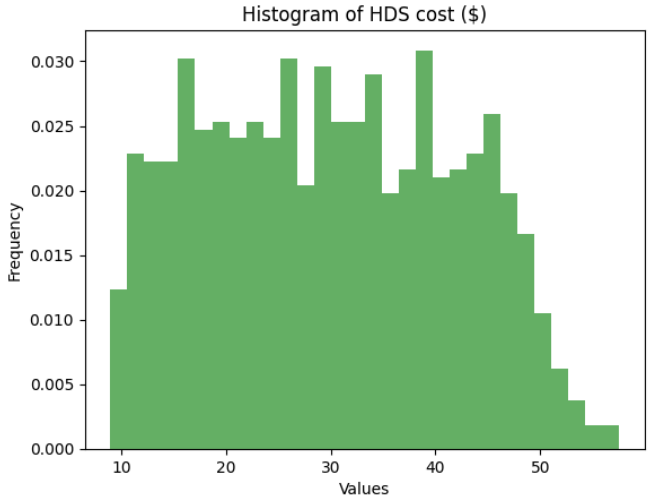

Histogram of Temperature (◦C)
Frequency
Values
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
100
200
300
400
500


Histogram of Pressure (bar)
Frequency
Values
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
20
40
60
80
100
120

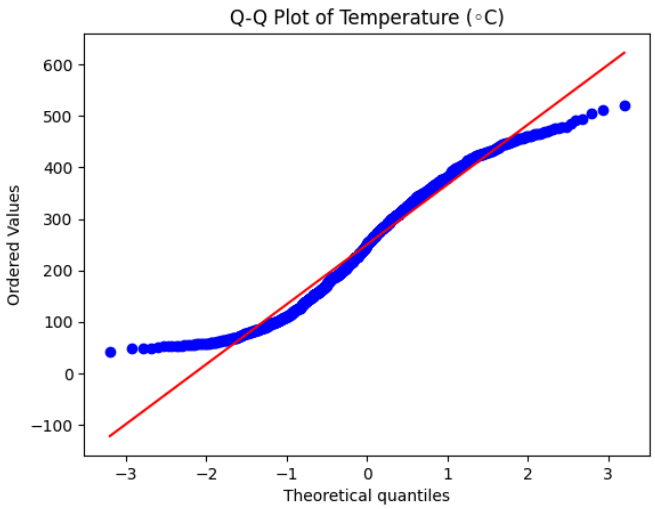

Histogram of Dosage (g)
Frequency
Values
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

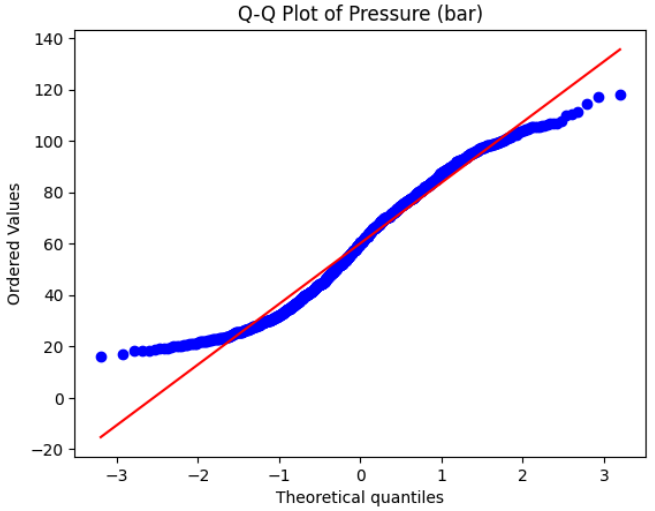


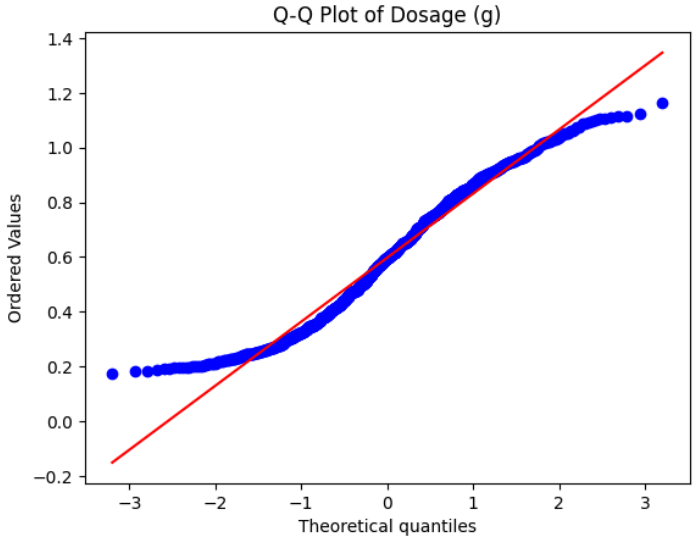


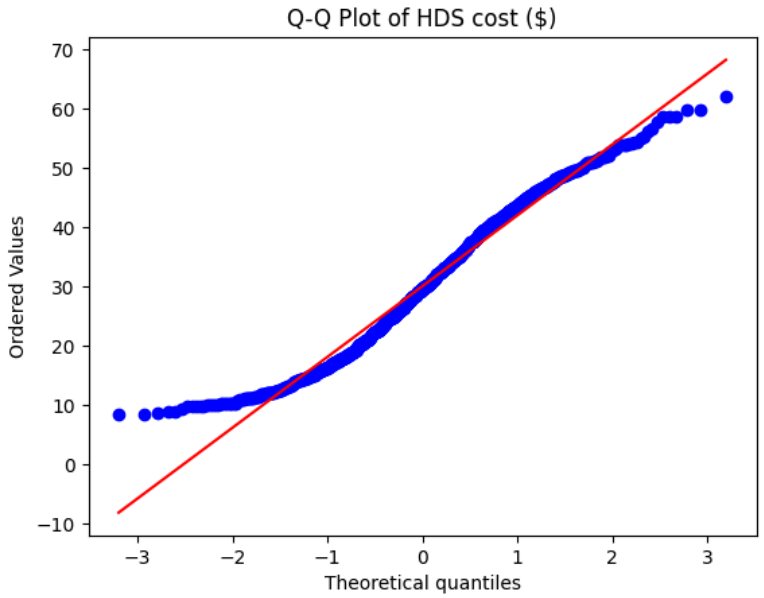


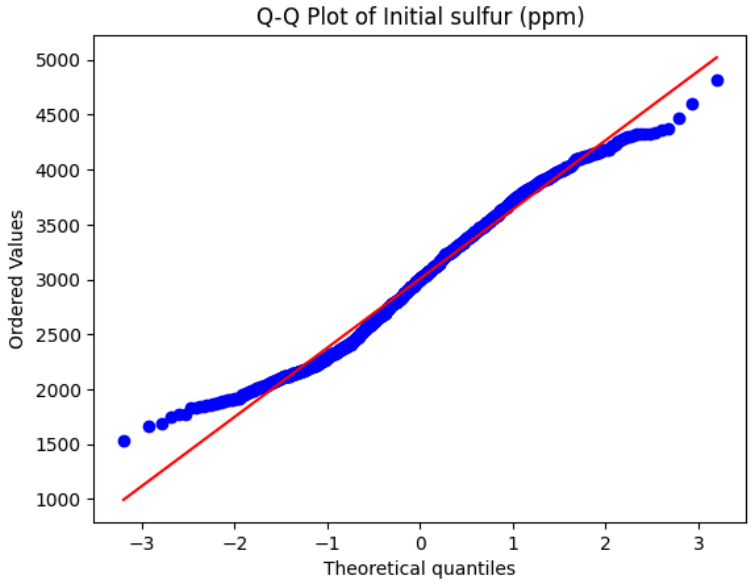
QQ Plots: This is plotted to see the gaussian distribution of the dataset.

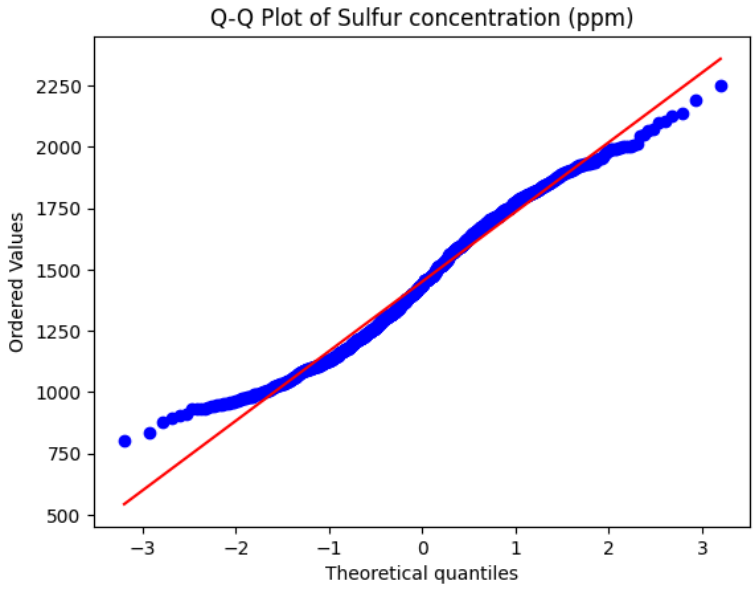

Q-Q Plot of Temperature (◦C)
Ordered Values
Theoretical quantiles
600
500
400
300
200
100
0
−100
−3
−2
−1
0
1
2
3

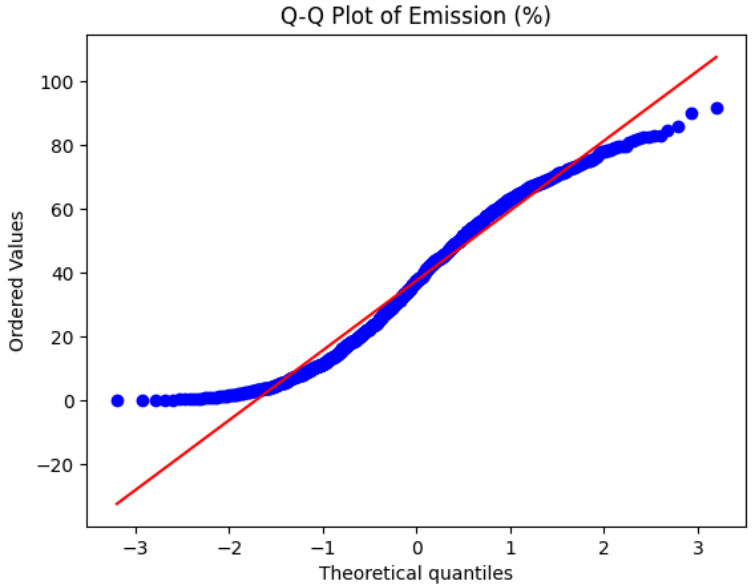

Q-Q Plot of Pressure (bar)
Ordered Values
Theoretical quantiles
140
120
100
80
60
40
20
0
−20
−3
−2
−1
0
1
2
3

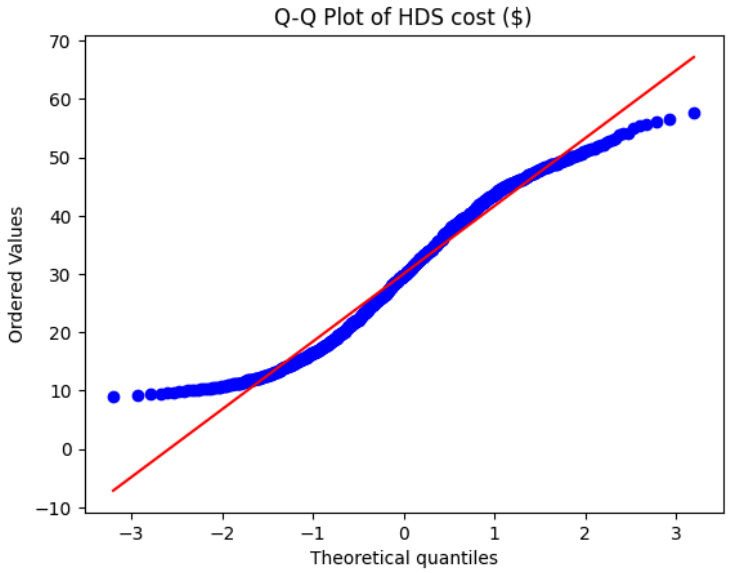

Q-Q Plot of Dosage (g)
Ordered Values
Theoretical quantiles
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−3
−2
−1
0
1
2
3


Q-Q Plot of HDS cost ($)
Ordered Values
Theoretical quantiles
70
60
50
40
30
20
10
0
−10
−3
−2
−1
0
1
2
3

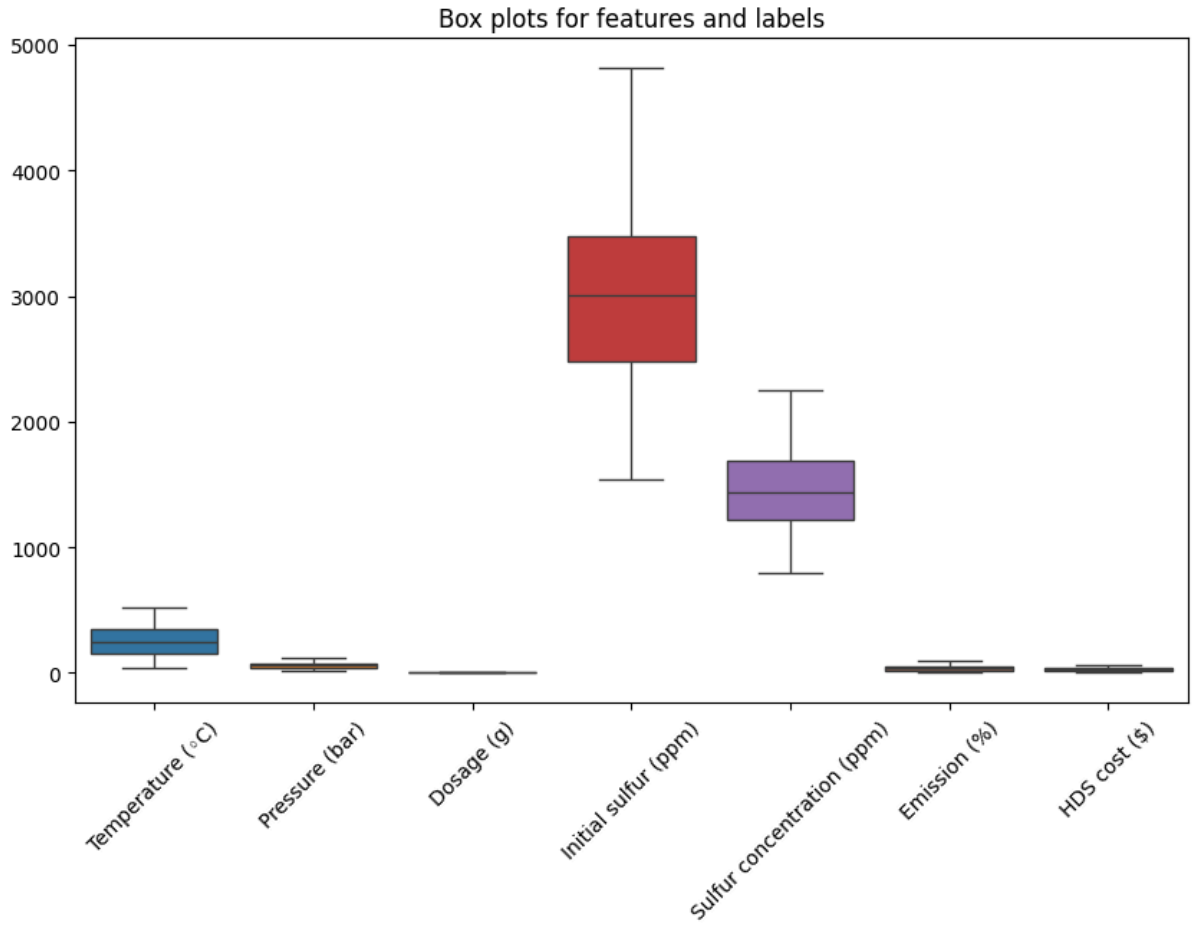

Q-Q Plot of Initial sulfur (ppm)
Ordered Values
Theoretical quantiles
5000
4500
4000
3500
3000
2500
2000
1500
1000
−3
−2
−1
0
1
2
3


Q-Q Plot of Sulfur concentration (ppm)
Ordered Values
Theoretical quantiles
2250
2000
1750
1500
1250
1000
750
500
−3
−2
−1
0
1
2
3


Q-Q Plot of Emission (%)
Ordered Values
Theoretical quantiles
100
80
60
40
20
0
−20
−3
−2
−1
0
1
2
3



BOXPLOT:



4. Feature Scaling: The features and labels have been scaled using Standardization (scaling features having a mean of 0 and standard deviation of 1).

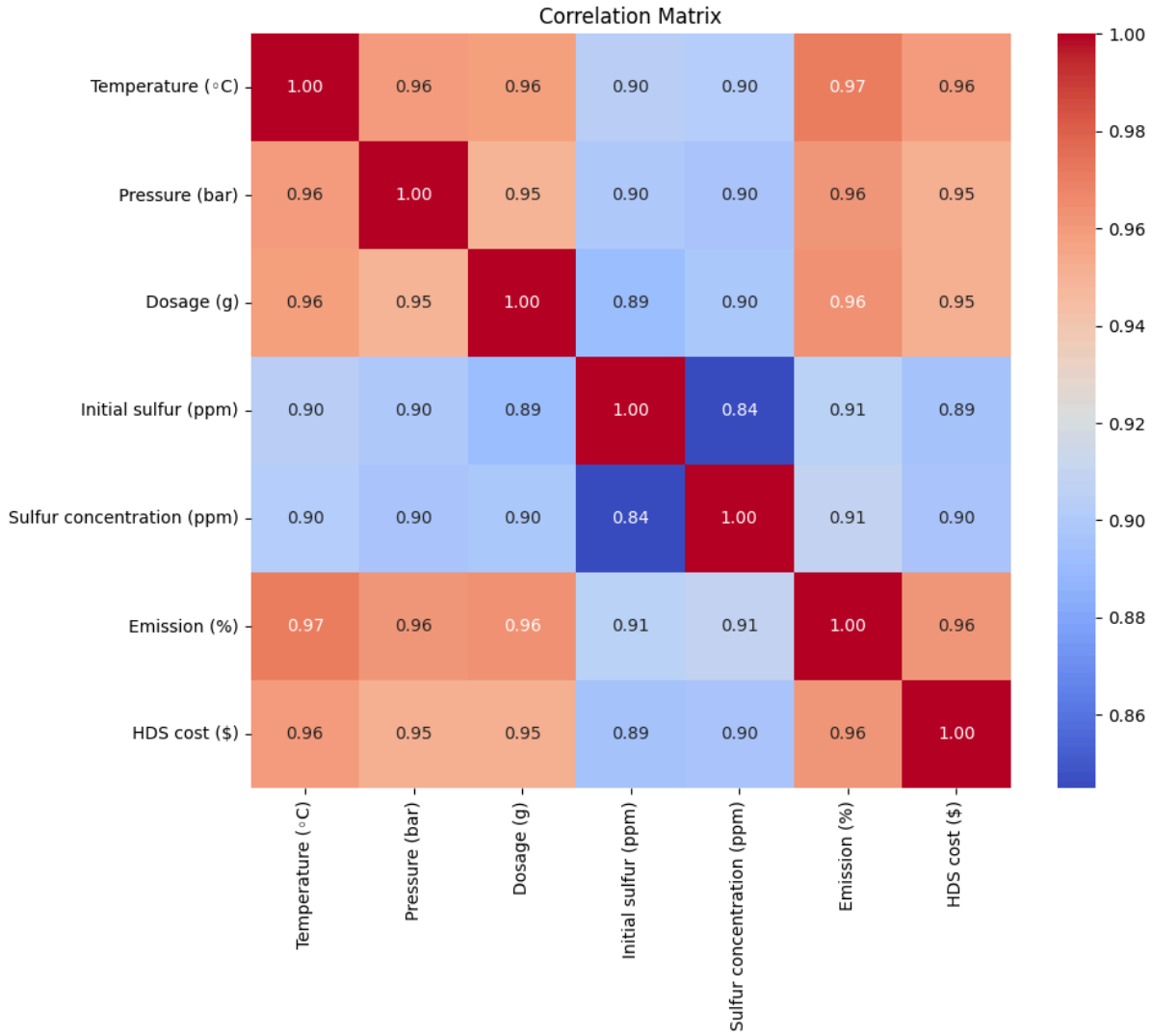
```python
from sklearn.preprocessing import StandardScaler


# Extract the feature variables
features = ['Temperature (◦C)', 'Pressure (bar)', 'Dosage (g)', 'Initial sulfur (ppm)']
X = nandini_df[features]

# Extract the label variables
labels = ['Sulfur concentration (ppm)', 'Emission (%)', 'HDS cost ($)']
y = nandini_df[labels]

# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler to your data and transform both features and labels
nandini_df_scaled = nandini_df.copy()  # Create a copy of the original DataFrame
nandini_df_scaled[features] = scaler.fit_transform(X)
nandini_df_scaled[labels] = scaler.fit_transform(y)

# Now nandini_df_scaled contains the scaled features and labels
```

5. Feature Selection: Since we just have four features and three labels, we do not require to do PCA here and we use all the features.

FEATURES: **Temperature (◦C), Pressure (bar) , Dosage (g) , Initial sulfur (ppm)**

LABELS: **Sulfur concentration (ppm), Emission (%), HDS cost ($)**

The correlation plot has been drawn:



6. Data Splitting: The data is split into two fractions - 80% as training set and 20% as the test set.

Model Architecture:

The proposed AI/ML model architecture for optimizing the catalytic sulfur removal process involves the integration of multiple machine learning algorithms, including Multi-layered Perceptron (MLP), Gaussian Process Regression (GPR), Multi-Task Lasso (MTL), and Multilinear Regression. Each algorithm is tailored to address specific aspects of the problem and contributes to the overall optimization of the Hydrodesulfurization (HDS) process.

Reasons for Choosing this Architecture:

- The chosen architecture combines different machine learning algorithms to leverage their respective strengths and address different aspects of the problem.
- MLP is well-suited for capturing **nonlinear relationships**, while GPR provides **probabilistic predictions** and uncertainty estimates.
- MTL performs feature selection and **regularization** across multiple tasks, enhancing model interpretability and generalization.
- Multilinear regression offers simplicity and **interpretability**, serving as a baseline for comparison.
- By integrating these diverse algorithms, the model architecture aims to provide accurate predictions for optimizing the catalytic sulfur removal process while ensuring robustness and interpretability.

Tools and Technologies:

1. Python:
   - Python is used as the primary programming language for implementing machine learning algorithms, data preprocessing, and analysis.
   - Libraries such as NumPy, Pandas, and Scikit-learn are utilized for numerical computations, data manipulation, and machine learning tasks.

2. Google Colab:
   - Google Colab is used for interactive development and documentation of the project.
   - It allows for the creation of executable code cells, visualizations, and markdown text, making it ideal for exploratory data analysis and model prototyping.

3. TensorFlow / PyTorch:
   - TensorFlow/PyTorch is used for implementing neural network architectures, particularly the Multi-layered Perceptron (MLP) model.
   - These deep learning frameworks provide efficient tools for building, training, and deploying neural networks, enabling the development of complex models for optimizing the HDS process.

4. Scikit-learn:

- Scikit-learn is utilized for implementing machine learning algorithms such as Gaussian Process Regression (GPR) and Multi-Task Lasso (MTL).
- It offers a wide range of machine learning models, preprocessing techniques, and evaluation metrics, facilitating the development and evaluation of predictive models.

5. Matplotlib and Seaborn:

- Matplotlib and Seaborn are used for data visualization, allowing for the creation of informative plots and charts to visualize relationships between variables and model performance.

6. Pandas and NumPy:

- Pandas and NumPy are employed for data manipulation, handling datasets, and performing numerical computations.
- These libraries provide efficient data structures and functions for data cleaning, transformation, and analysis, essential for preprocessing the HDS dataset.

4 Implementation Plan

Development Phases:

Data Collection and Preprocessing:

- Gathered the HDS process dataset and performed initial exploratory data analysis.
- Cleaned the data by handling missing values, and removing outliers.

Model Development:

- Implemented machine learning algorithms including MLP, GPR, MTL, and Multilinear Regression using Python libraries.
- Fine-tuned hyperparameters using GridSearch.
- Developed a pipeline for training and evaluating each model.

Model Evaluation and Optimization:

- Evaluated the performance of each model using appropriate metrics such as R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).
- Optimize the models further by adjusting parameters and features based on evaluation results.

Validation and Testing:

- Validating the trained models using cross-validation techniques to ensure generalization to unseen data.
- Testing the final models on a separate test dataset to assess their performance and robustness.

Model Training:

Training the machine learning models for optimizing the catalytic sulfur removal process involved several strategies and techniques to ensure accuracy and robustness. Here are the key components of the model training process:

Algorithm Selection:

- For this project, I have selected algorithms such as Multi-layered Perceptron (MLP), Gaussian Process Regression (GPR), Multi-Task Lasso (MTL), and Multilinear based on their suitability for capturing different aspects of the HDS process.

Data Splitting:

- I had split the dataset into training and testing sets to evaluate model performance effectively.
- I used a typical split of **80% for training** and **20% for testing**.

Hyperparameter Tuning:

- I did hyperparameter tuning for each model(**except for Multi-Linear Regression**) to optimize their performance and improve accuracy.

Model Training:

- I trained each model using the training dataset and the selected algorithm and then predicted the values for the test dataset. I also plotted the **Actual vs Predicted Curves** and **Residual Plots** for each label to understand the performance of each model.
- I have shown results for each model for both cases- with and without hyperparameter tuning.

By following these strategies for model training, the project aimed to develop accurate and reliable machine learning models for optimizing the catalytic sulfur removal process, contributing to environmental sustainability and process efficiency in the petroleum industry.

Model Evaluation:

Performance is evaluated using metrics such as **Coefficient of Determination (R^2), Mean Absolute Error (MAE)** and **Root Mean Square Error (RMSE)** to assess prediction accuracy.

5 Testing and Deployment

Testing Strategy:

1. Test Dataset:

- Using a separate test dataset, distinct from the training and validation sets, to evaluate the model's performance against unseen data.
- Ensuring that the test dataset represents a diverse range of scenarios and conditions to assess the model's generalization ability.

2. Evaluation Metrics:

- Evaluating the model's performance using appropriate metrics such as R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).
- Additionally, analyzing any discrepancies between predicted and actual values to identify areas for improvement.

3. Cross-Validation:

- Employing cross-validation techniques, such as k-fold cross-validation, to validate the model's performance across multiple subsets of the data.
- Cross-validation provides a robust estimate of the model's generalization ability and helps assess its reliability under different conditions.

Deployment Strategy:

1. Scalability:

- Ensuring that the deployed model can handle varying workloads and datasets, scaling effectively as demand increases.
- Considering deploying the model on scalable cloud platforms such as AWS, Google Cloud, or Azure to accommodate fluctuations in usage.

2. Performance Optimization:

- Optimizing the model's performance for real-time inference by leveraging techniques such as model quantization, pruning, or compression.
- Implementing caching mechanisms or distributed computing strategies to minimize latency and improve response times.

3. API Development:

- Developing an API (Application Programming Interface) to expose the model's functionality to other software applications and systems.
- Using RESTful API standards for interoperability and ease of integration with existing infrastructure.

4. Monitoring and Maintenance:

- Implementing monitoring tools to continuously track the model's performance and health in production.
- Establishing protocols for regular maintenance, including model retraining, updating, and versioning to ensure ongoing reliability and accuracy.

Ethical Considerations:

1. Bias and Fairness:

- Evaluating the model for biases that may result in unfair outcomes and implementing fairness-aware algorithms or mitigation techniques to address bias and promote equitable decision-making.

2. Transparency and Interpretability:

- Ensuring transparency in the model's decision-making process, providing explanations or interpretations for its predictions.
- Using interpretable models or techniques such as SHAP (SHapley Additive exPlanations) values to enhance transparency and trustworthiness.

3. Privacy and Data Security:

- Safeguarding sensitive data and ensuring compliance with privacy regulations such as GDPR or HIPAA.
- Implementing encryption, anonymization, or differential privacy techniques to protect individuals' privacy and confidentiality.

4. Accountability and Governance:

- Establishing clear accountability frameworks and governance structures for model development, deployment, and usage.
- Documenting model decisions, assumptions, and limitations to facilitate accountability and mitigate potential risks or harms.

By addressing these considerations, the project aims to deploy the model responsibly, ensuring ethical use and minimizing potential negative impacts on stakeholders.

6 Results and Discussion

Findings: **MODEL RESULTS**

- MULTILINEAR REGRESSION:

  Metrics for Sulfur concentration (ppm):

  R^2: 0.8433

  RMSE: 0.3889

  MAE: 0.3004

  Metrics for Emission (%):

  R^2: 0.9629

  RMSE: 0.1906

  MAE: 0.1290

  Metrics for HDS cost ($):

  R^2: 0.9428

  RMSE: 0.2395

  MAE: 0.1833

  Equation for Sulfur concentration (ppm):

Sulfur concentration (ppm) = 0.00 + 0.26 * Temperature (◦C) + 0.27 * Pressure (bar) + 0.31 * Dosage (g) + 0.09 * Initial sulfur (ppm)

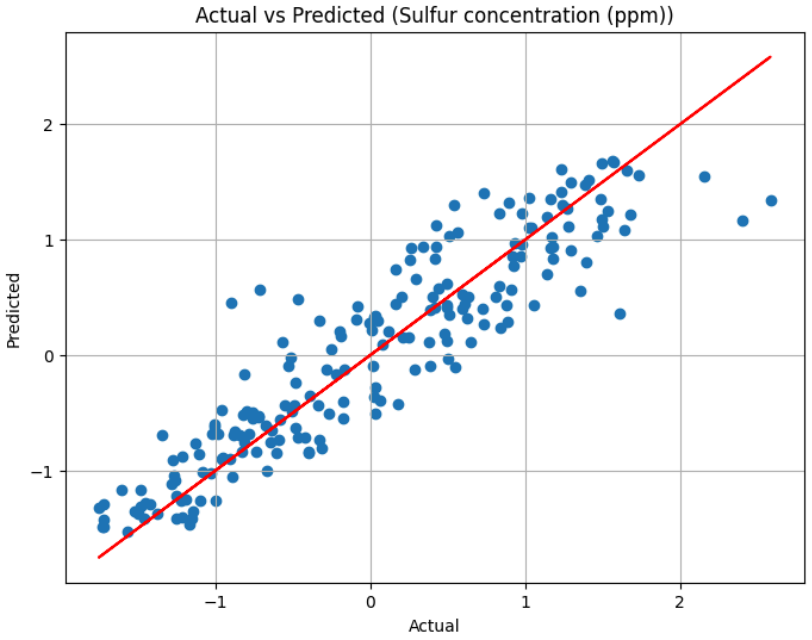
Equation for Emission (%):

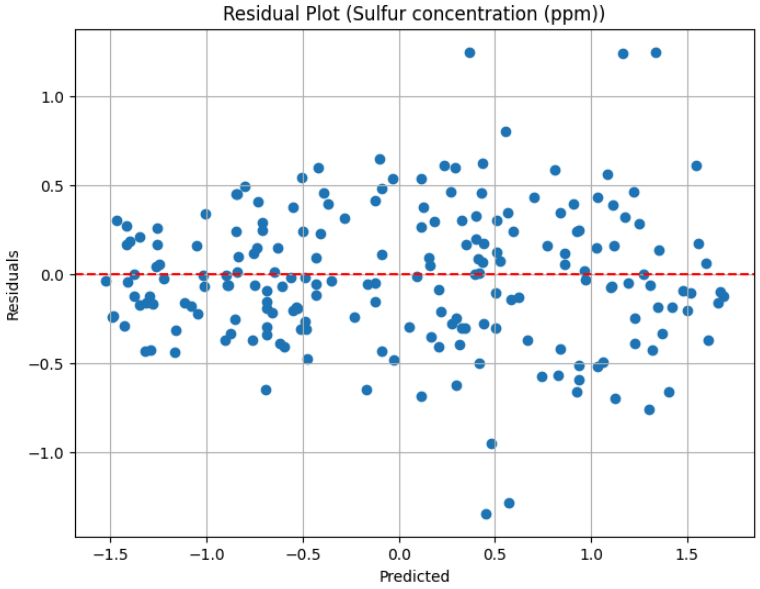
Emission (%) = 0.00 + 0.41 * Temperature (◦C) + 0.22 * Pressure (bar) + 0.30 * Dosage (g) + 0.07 * Initial sulfur (ppm)

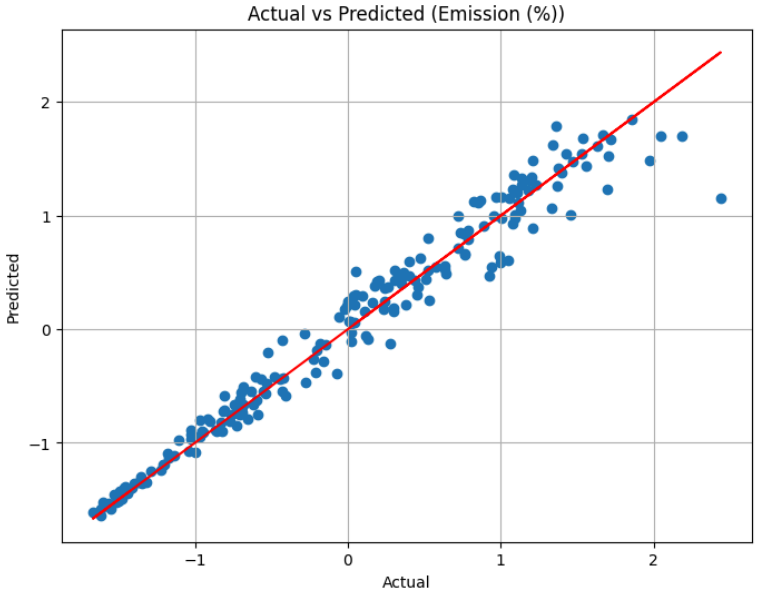
Equation for HDS cost ($):

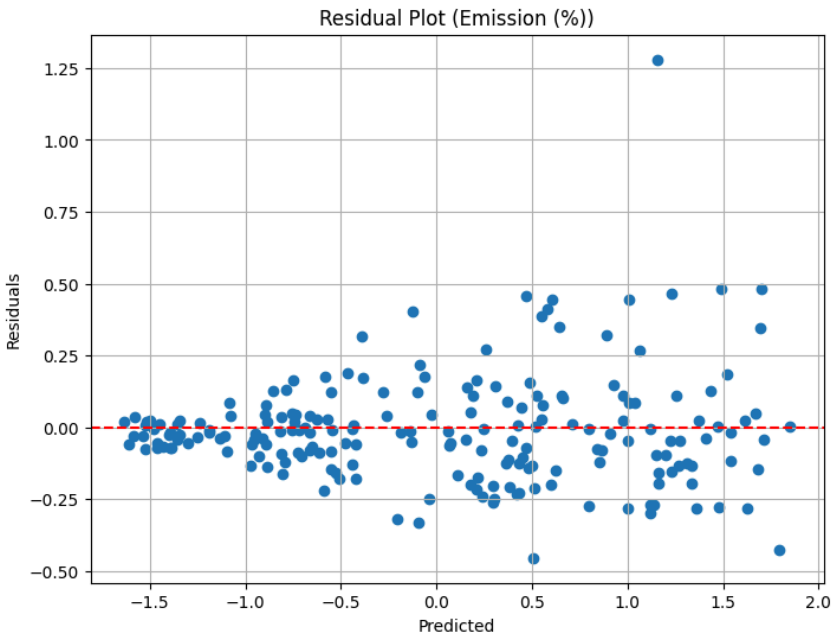
HDS cost ($) = -0.01 + 0.39 * Temperature (◦C) + 0.23 * Pressure (bar) + 0.29 * Dosage (g) + 0.07 * Initial sulfur (ppm)

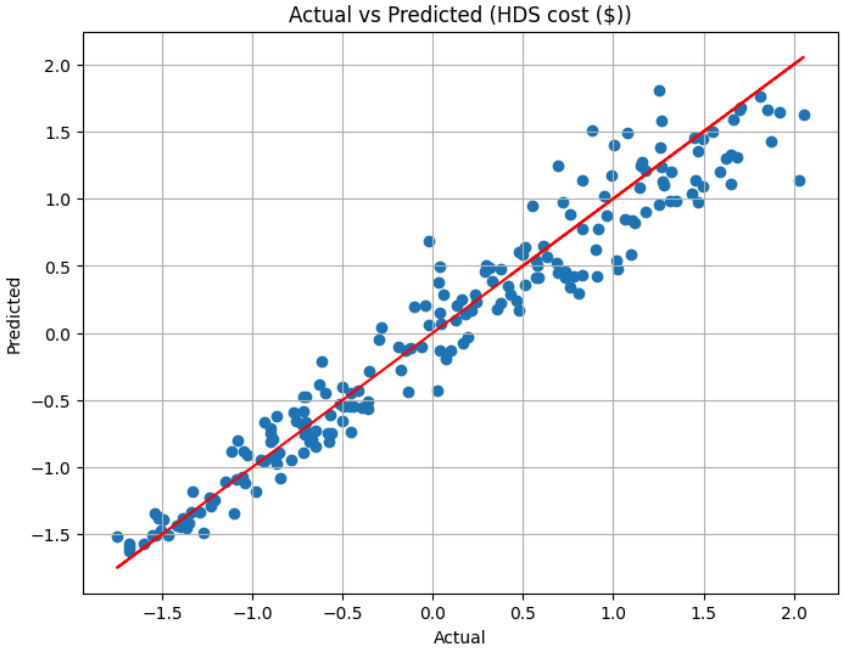
PLOTS:

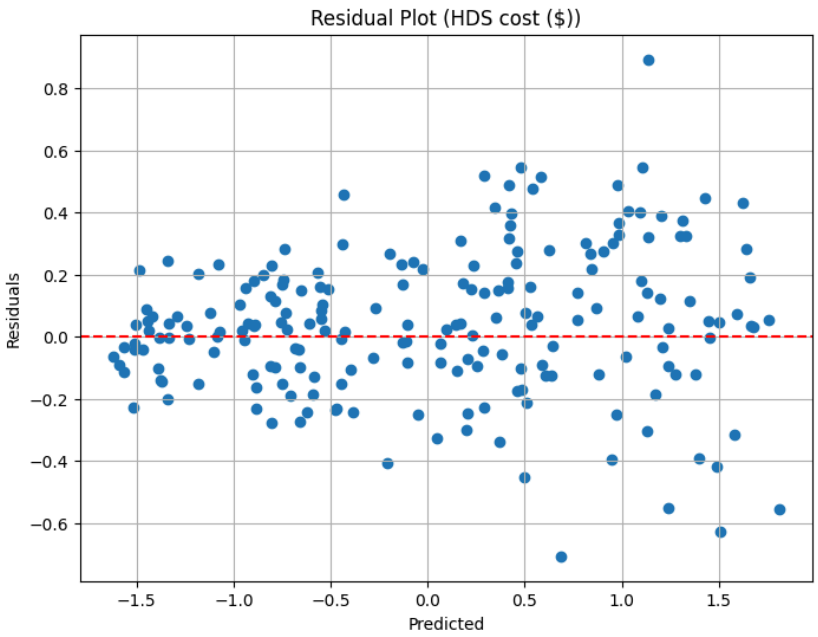



Residual Plot (Sulfur concentration (ppm))
Residuals
Predicted
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Actual vs Predicted (Emission (%))
Predicted
Actual
2
1
0
−1
−1
0
1
2


Residual Plot (Emission (%))
Residuals
Predicted
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Actual vs Predicted (HDS cost ($))
Predicted
Actual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



- MULTI-LAYER PERCEPTRON(Without Hyperparameter Tuning):

  Metrics for Sulfur concentration (ppm):

  R^2: 0.7957

  RMSE: 0.4439

  MAE: 0.3320

  Metrics for Emission (%):

  R^2: 0.9617

  RMSE: 0.1883

  MAE: 0.1416

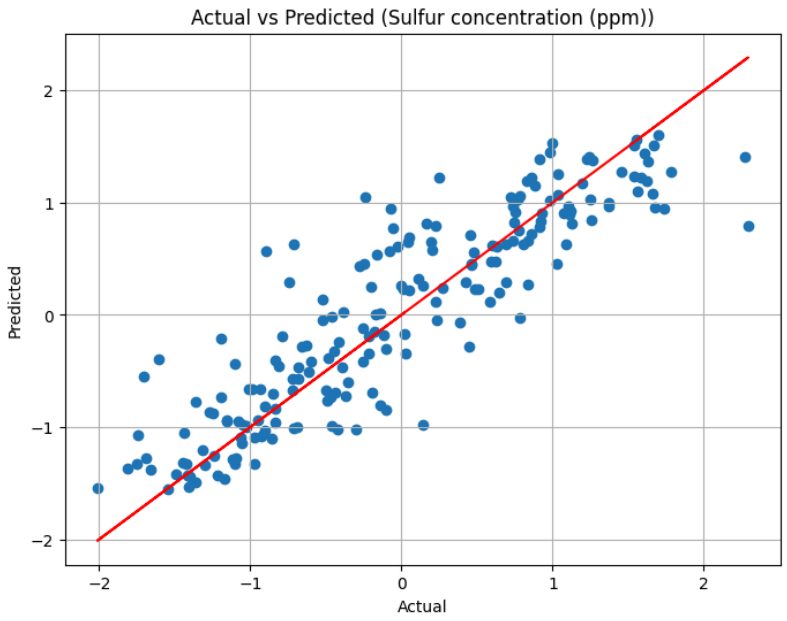
  Metrics for HDS cost ($):

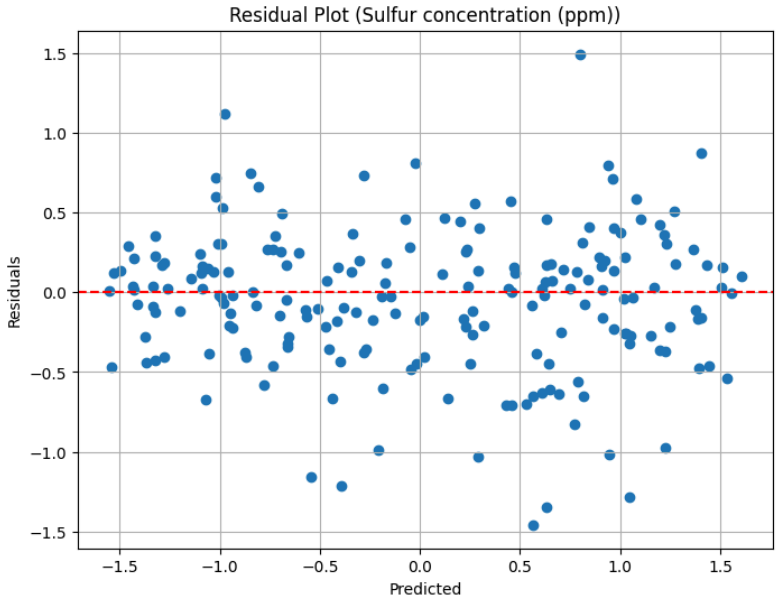
  R^2: 0.9432

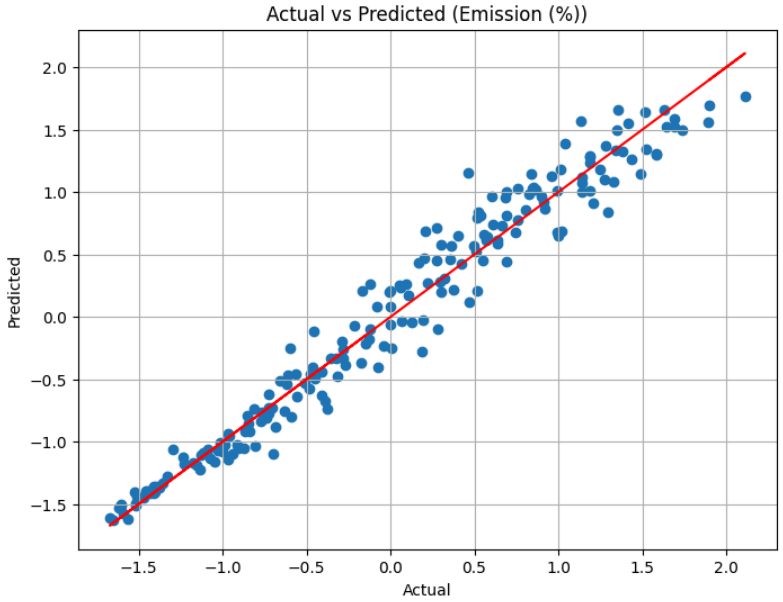
  RMSE: 0.2313

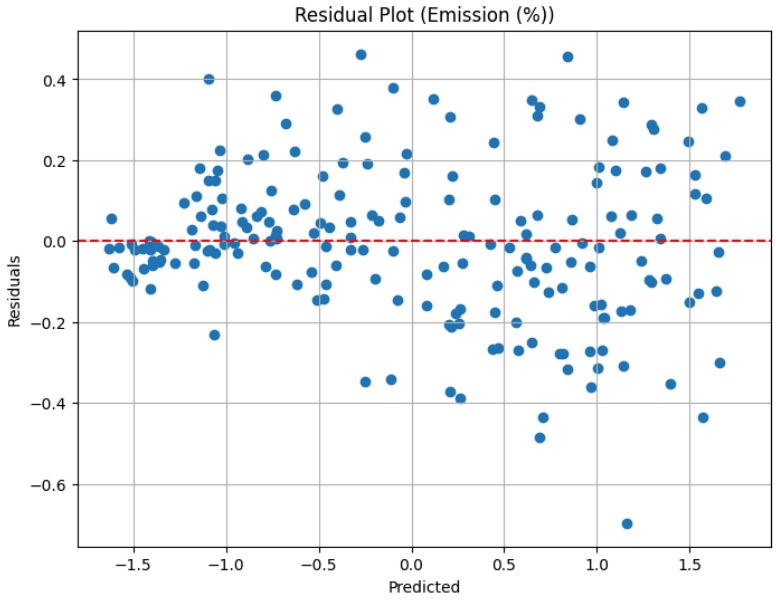
  MAE: 0.1773

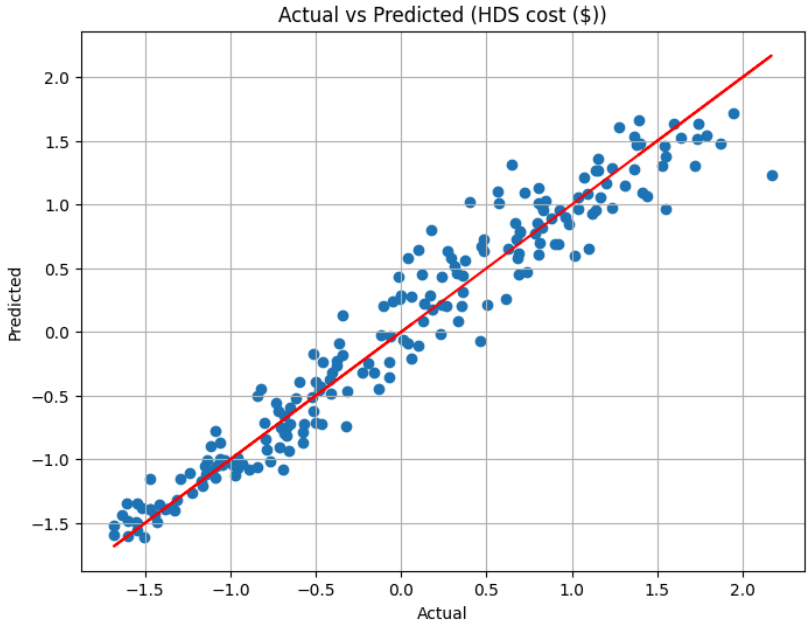
  PLOTS:

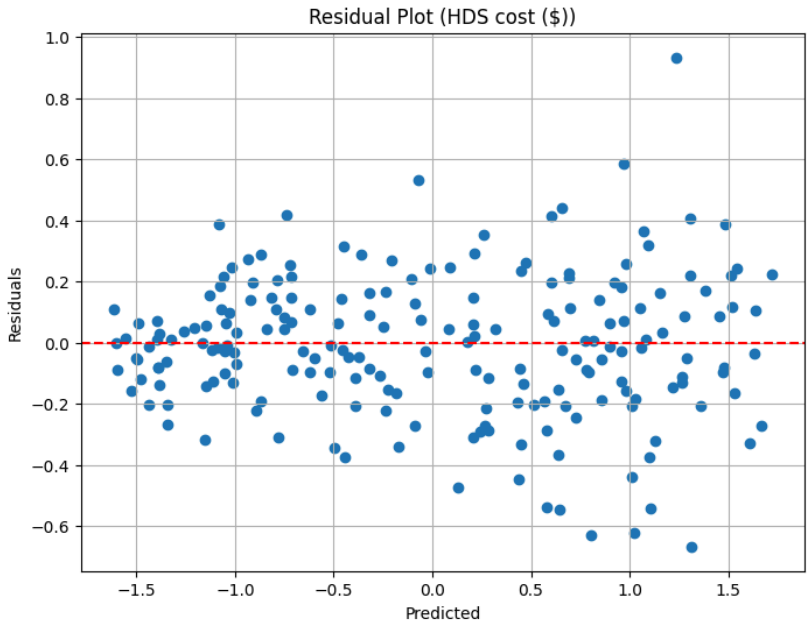

Actual vs Predicted (Sulfur concentration (ppm))
Predicted
Actual
2
1
0
−1
−2
−2
−1
0
1
2


Residual Plot (Sulfur concentration (ppm))
Residuals
Predicted
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Actual vs Predicted (Emission (%))
Predicted
Actual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Residual Plot (Emission (%))
Residuals
Predicted
0.4
0.2
0.0
−0.2
−0.4
−0.6
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5





- MULTI-LAYER PERCEPTRON(With Hyperparameter Tuning):

  Metrics for Sulfur concentration (ppm):

  R^2: 0.8388

  RMSE: 0.3944

  MAE: 0.2979

  Metrics for Emission (%):

  R^2: 0.9622

  RMSE: 0.1924

MAE: 0.1303

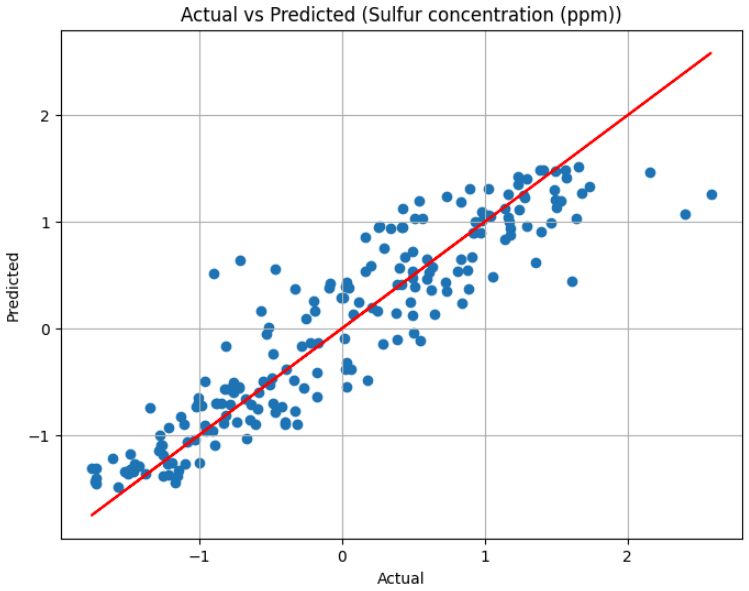
Metrics for HDS cost ($):

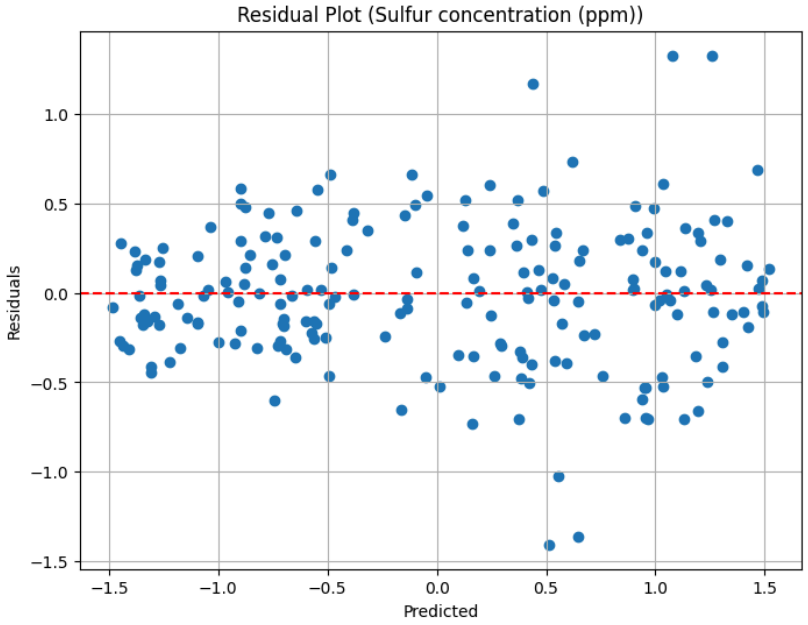
R^2: 0.9427

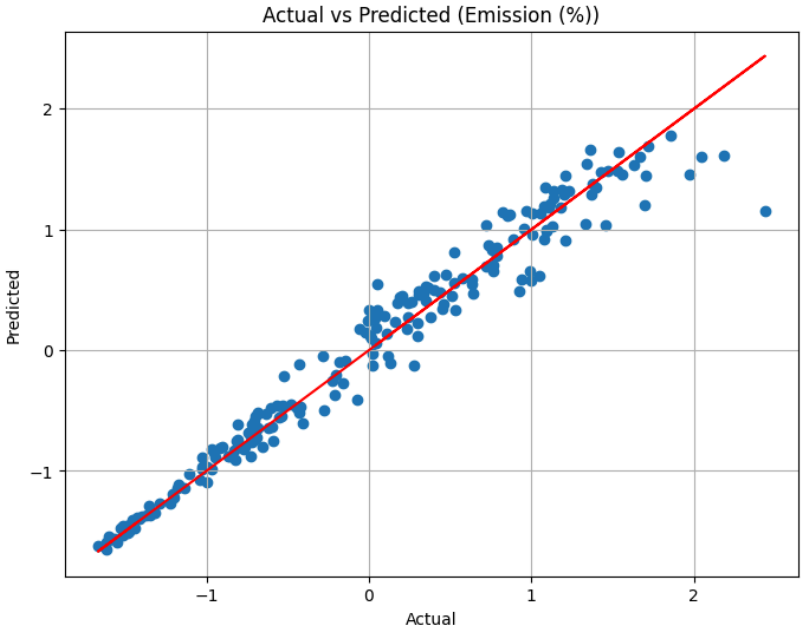
RMSE: 0.2396

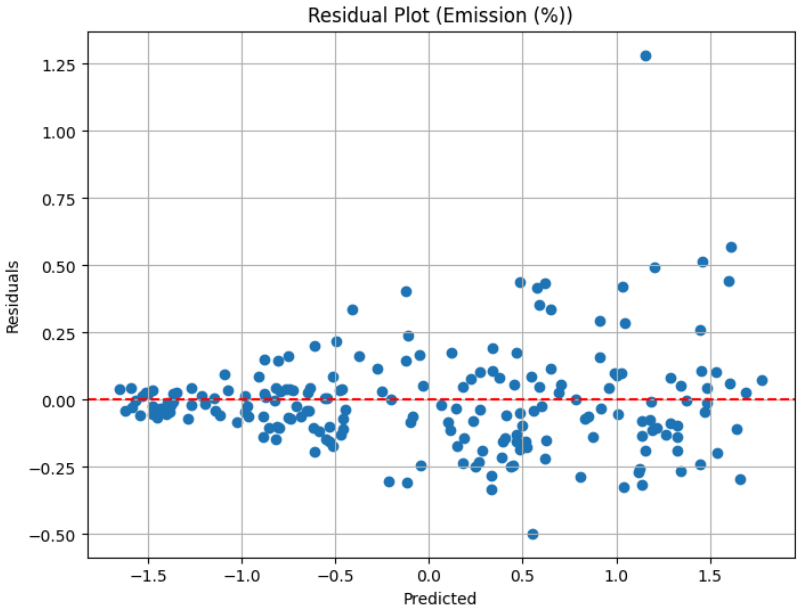
MAE: 0.1861

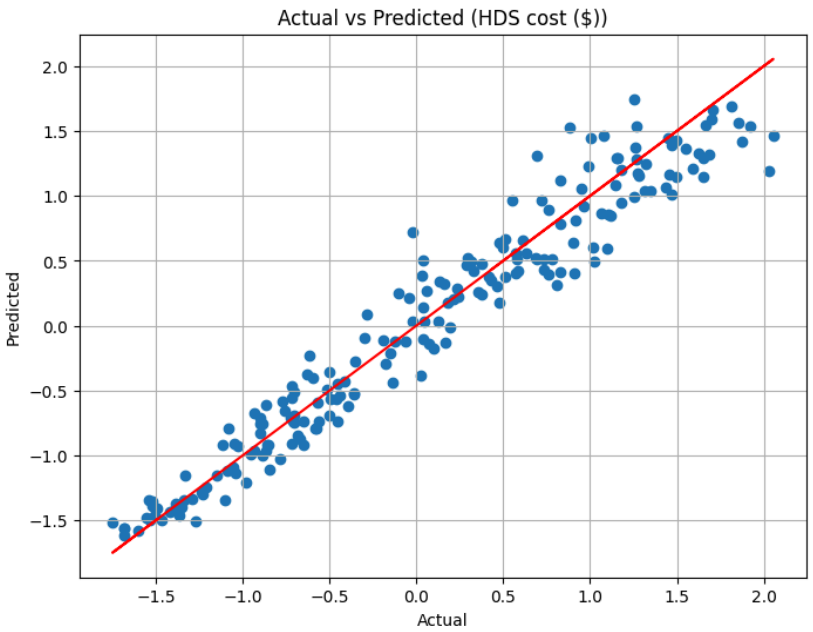
PLOTS:

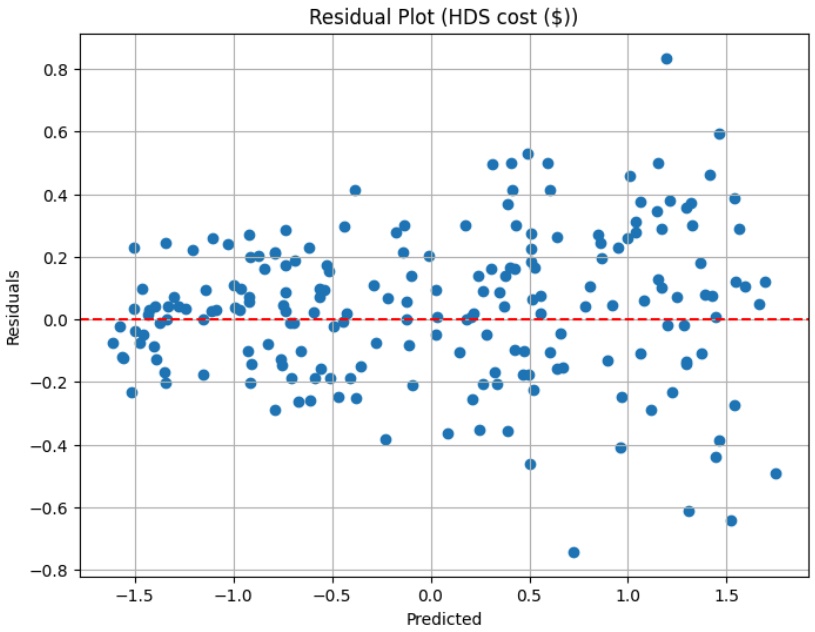





Actual vs Predicted (Emission (%))
Predicted
Actual
2
1
0
−1
−1
0
1
2


Residual Plot (Emission (%))
Residuals
Predicted
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5





- GAUSSIAN PROCESS REGRESSION(Without Hyperparameter Tuning):

  Metrics for Sulfur concentration (ppm):

  R^2: 0.5380

  RMSE: 0.6677

  MAE: 0.5303

  Metrics for Emission (%):

  R^2: 0.7953

  RMSE: 0.4475

MAE: 0.3139

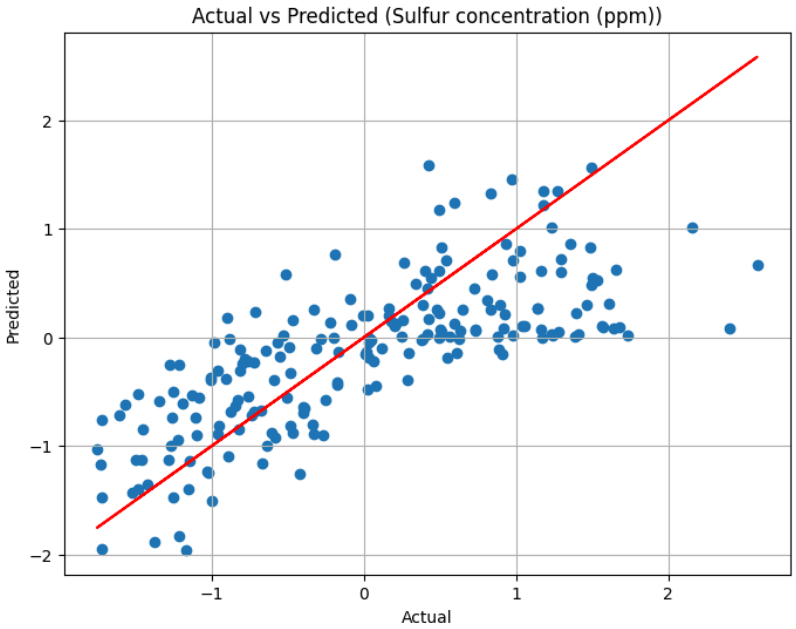
Metrics for HDS cost ($):

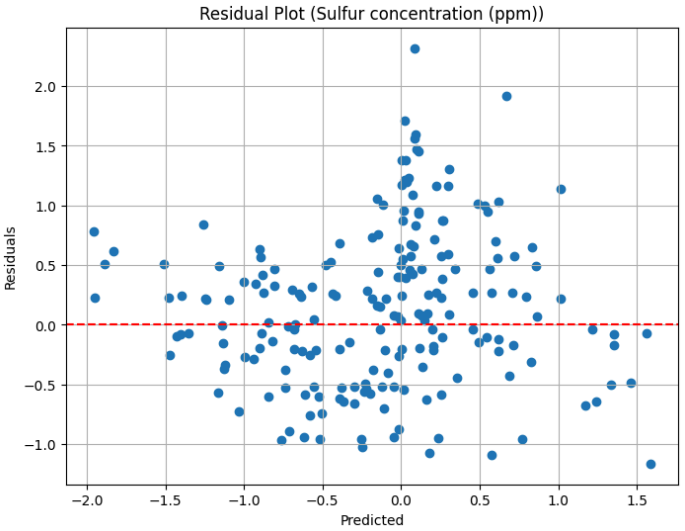
R^2: 0.7114

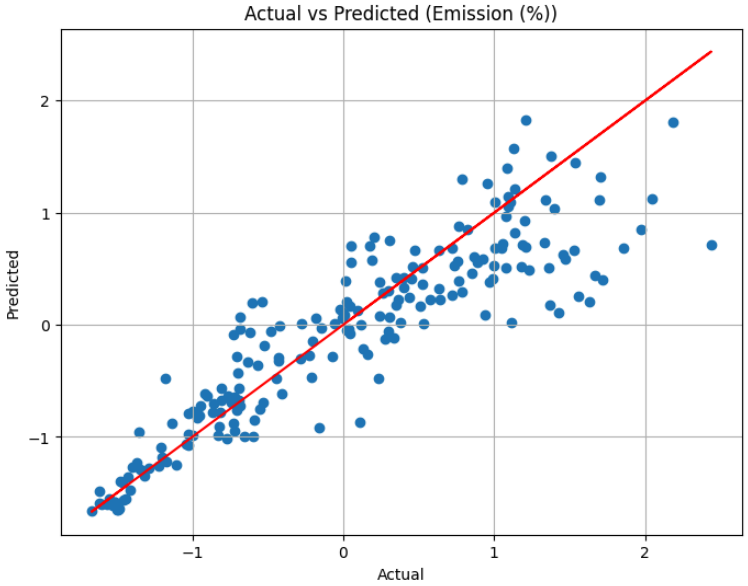
RMSE: 0.5377

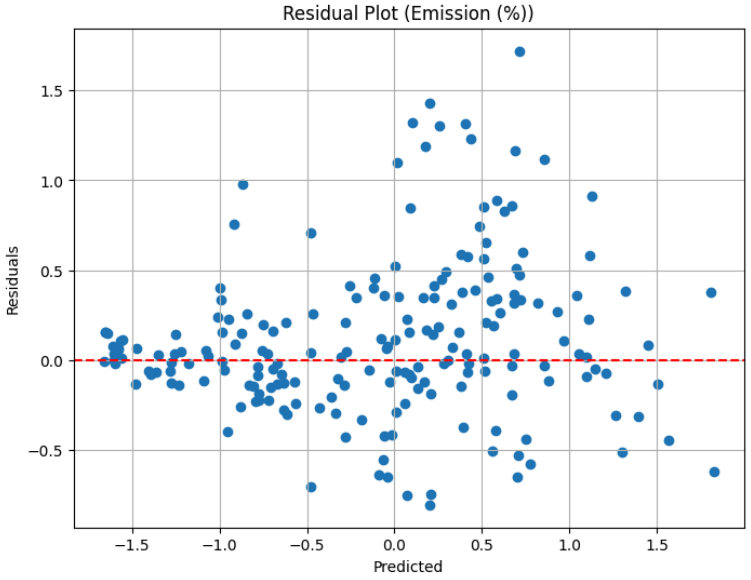
MAE: 0.3988

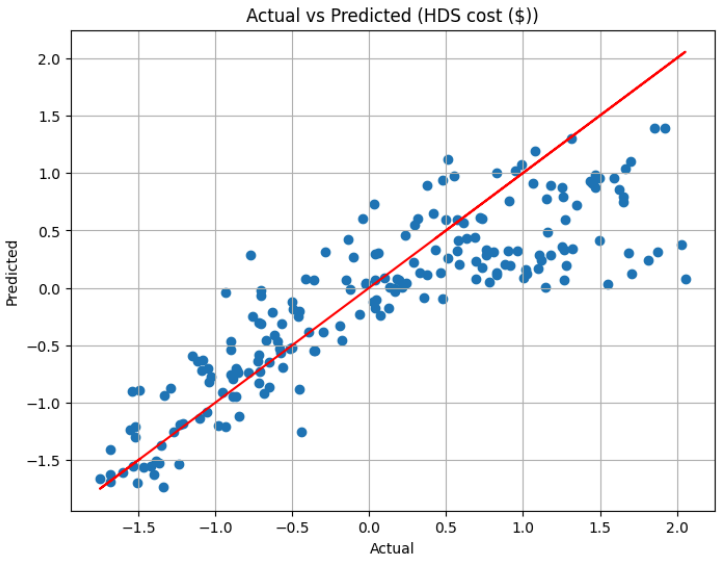
PLOTS:

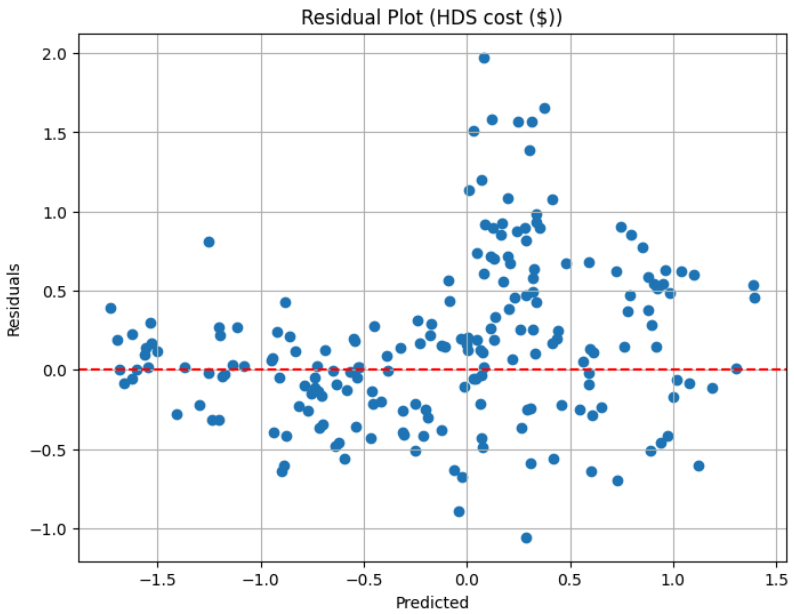





Actual vs Predicted (Emission (%))
Predicted
Actual
2
1
0
−1
−1
0
1
2


Residual Plot (Emission (%))
Residuals
Predicted
1.5
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Actual vs Predicted (HDS cost ($))
Predicted
Actual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



- GAUSSIAN PROCESS REGRESSION(With Hyperparameter Tuning):

  Metrics for Sulfur concentration (ppm):

  R^2: 0.5380

  RMSE: 0.6677

  MAE: 0.5303

  Metrics for Emission (%):

  R^2: 0.7953

  RMSE: 0.4475

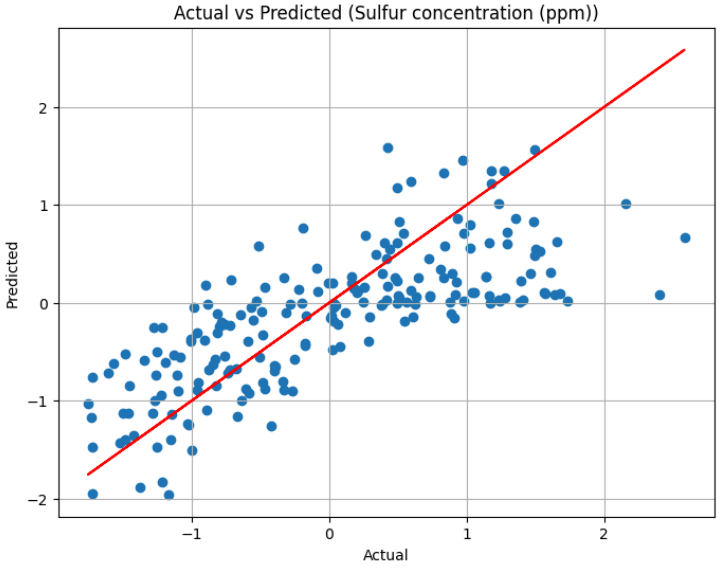
  MAE: 0.3139

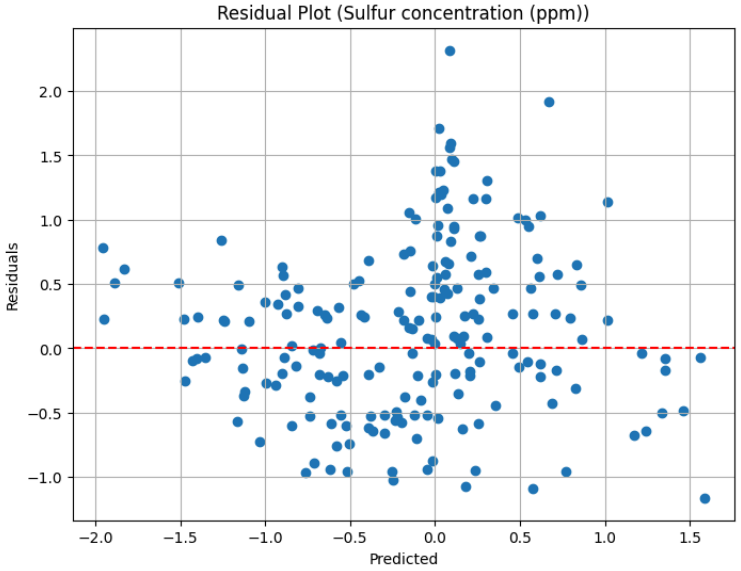
  Metrics for HDS cost ($):

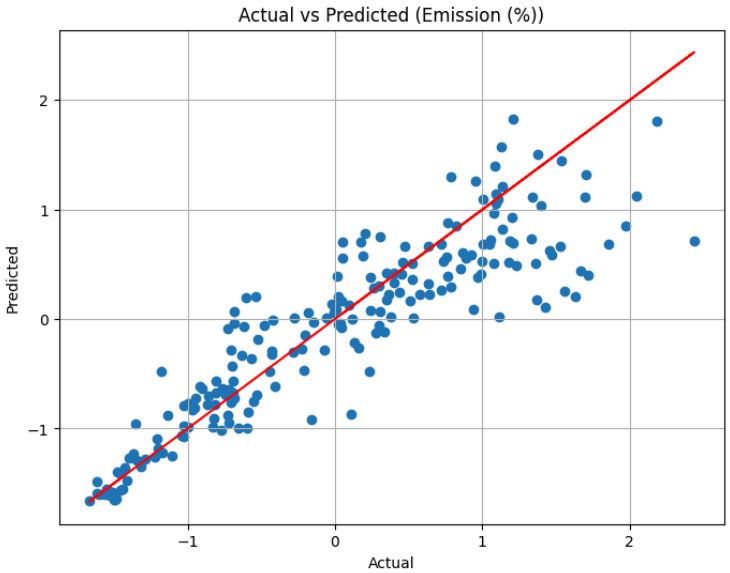
  R^2: 0.7114

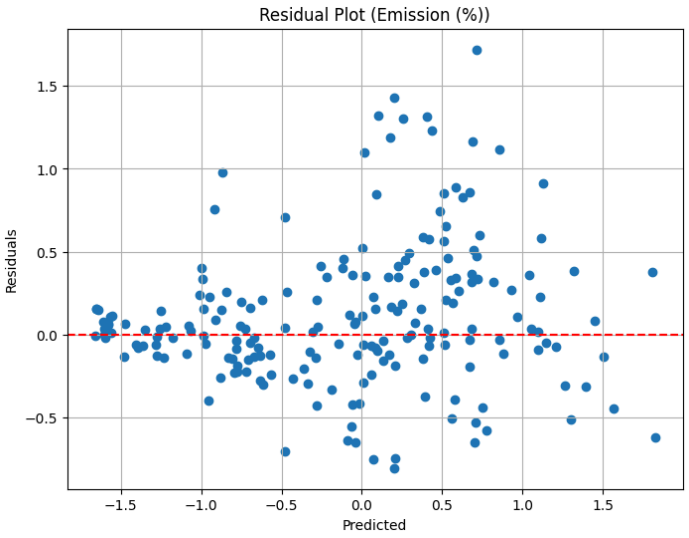
  RMSE: 0.5377

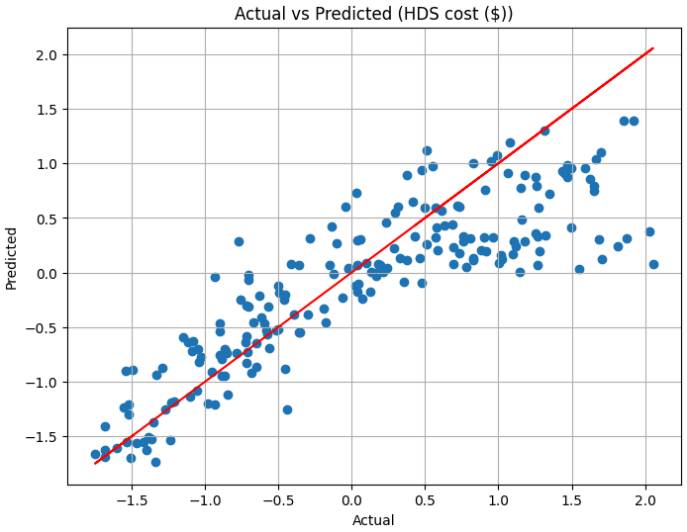
  MAE: 0.3988

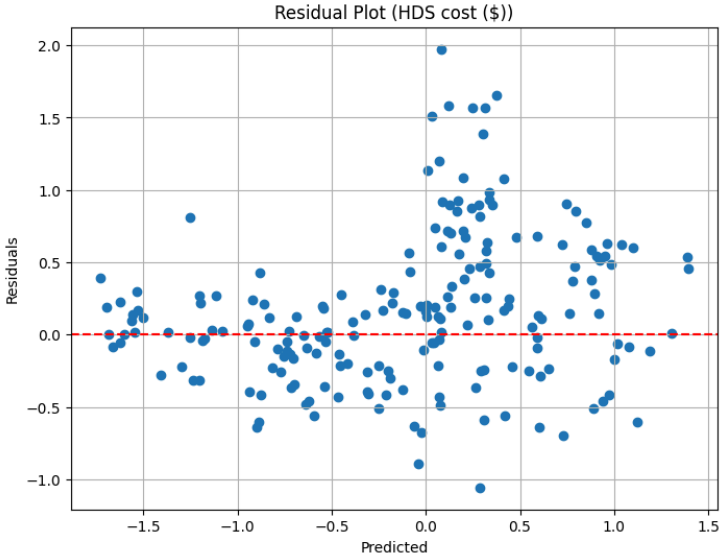

Actual vs Predicted (Sulfur concentration (ppm))
Predicted
Actual
2
1
0
−1
−2
−1
0
1
2


Residual Plot (Sulfur concentration (ppm))
Residuals
Predicted
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Actual vs Predicted (Emission (%))
Predicted
Actual
2
1
0
−1
−1
0
1
2


Residual Plot (Emission (%))
Residuals
Predicted
1.5
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Actual vs Predicted (HDS cost ($))
Predicted
Actual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Residual Plot (HDS cost ($))
Residuals
Predicted
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

- MULTI-TASK LASSO(Without Hyperparameter Tuning):

Metrics for Sulfur concentration (ppm):

R^2: 0.5067

RMSE: 0.6897

MAE: 0.5864

Metrics for Emission (%):

R^2: 0.6141

RMSE: 0.5976

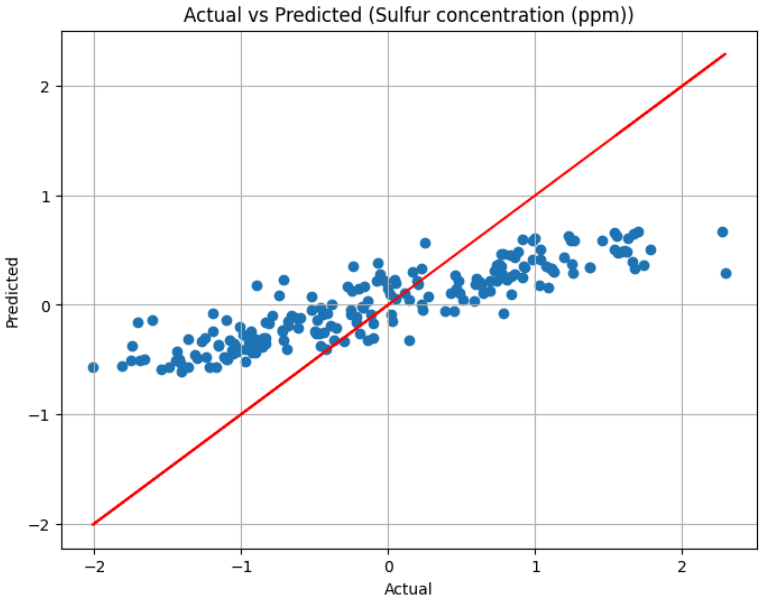
MAE: 0.5091

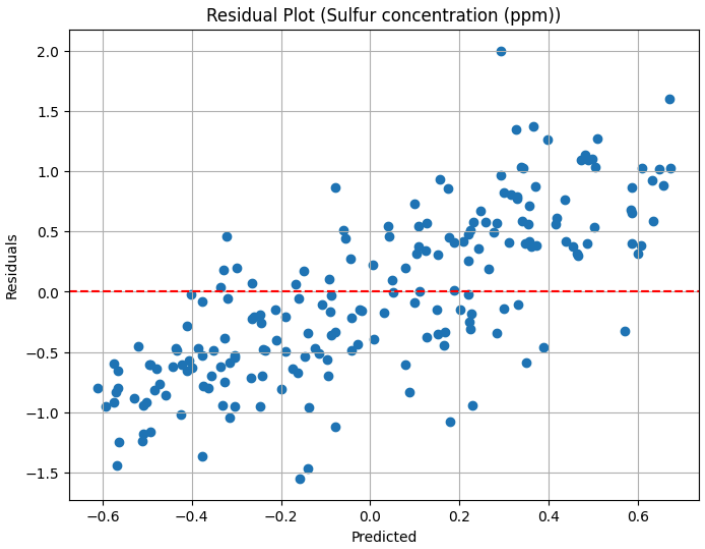
Metrics for HDS cost ($):

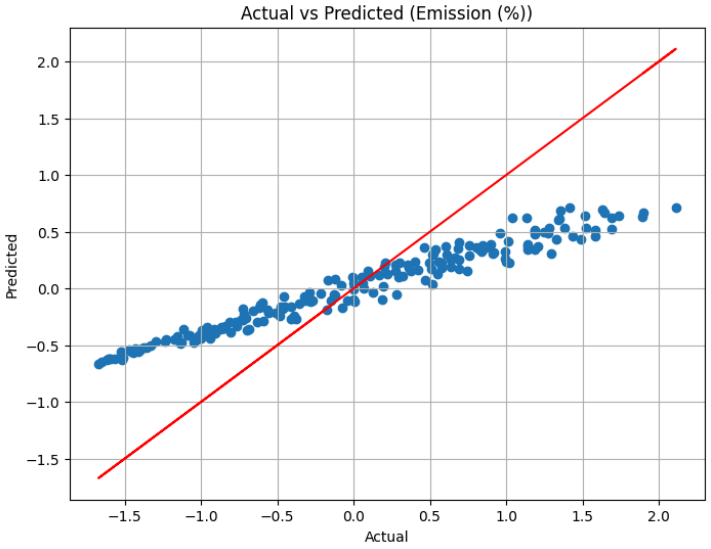
R^2: 0.5955

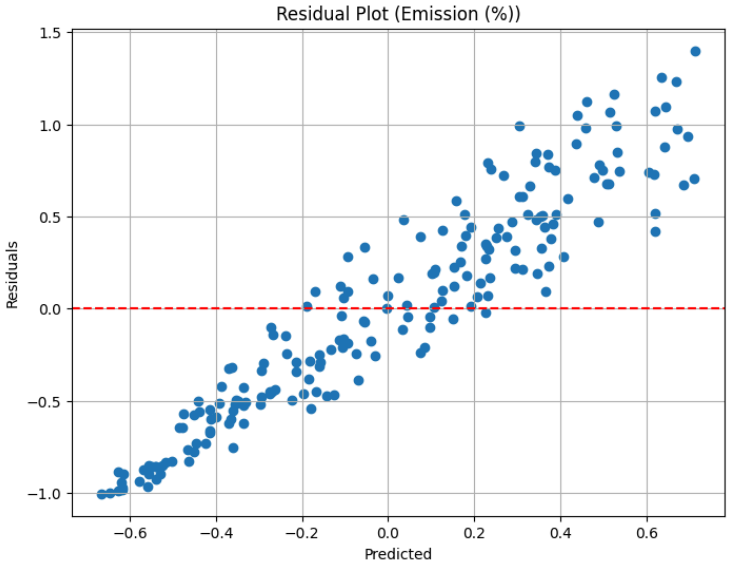
RMSE: 0.6176

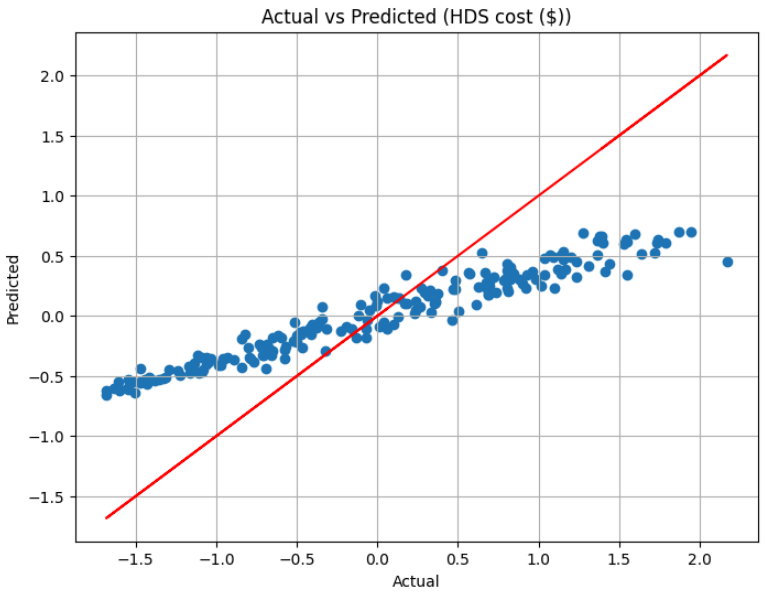
MAE: 0.5272

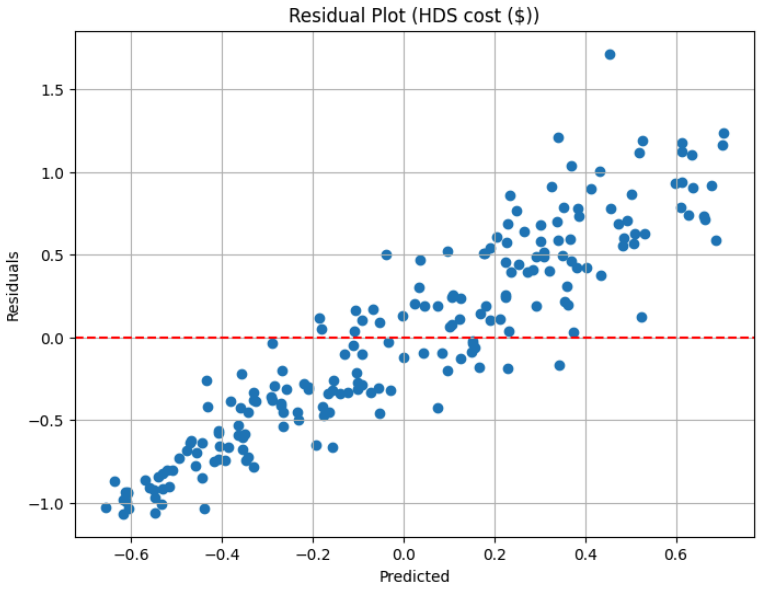



Residual Plot (Sulfur concentration (ppm))
Residuals
Predicted


Actual vs Predicted (Emission (%))
Predicted
Actual


Residual Plot (Emission (%))
Residuals
Predicted





- MULTI-TASK LASSO(With Hyperparameter Tuning):

  Metrics for Sulfur concentration (ppm):

  R^2: 0.8001

  RMSE: 0.4390

  MAE: 0.3339

  Metrics for Emission (%):

  R^2: 0.9646

  RMSE: 0.1809

  MAE: 0.1370

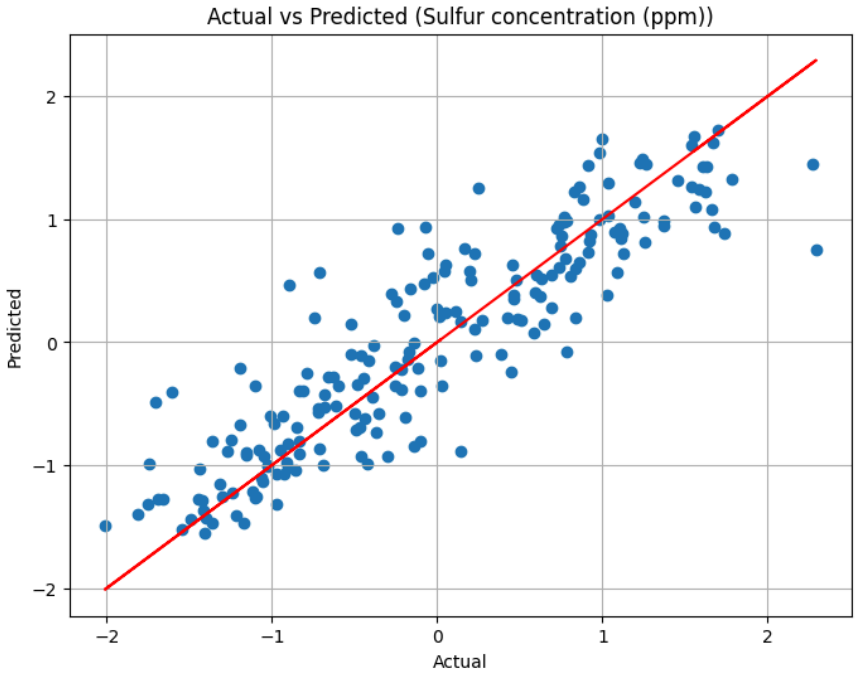
Metrics for HDS cost ($):

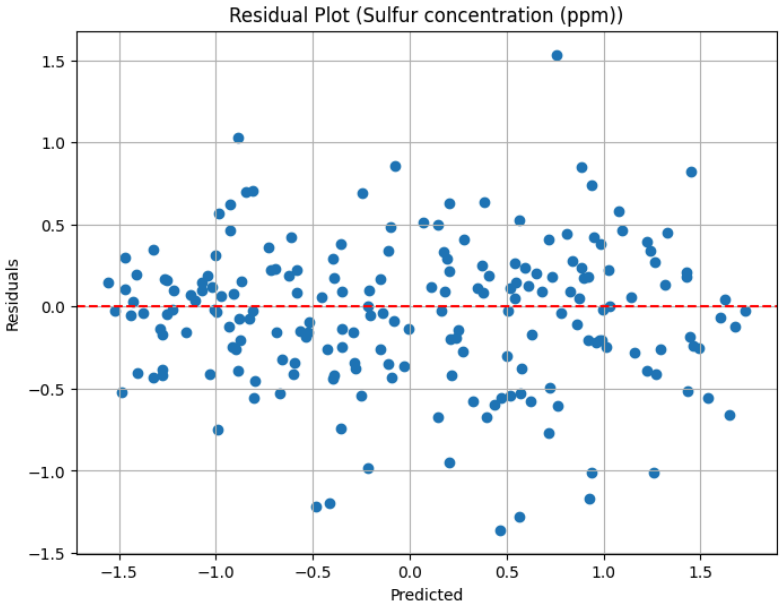
R^2: 0.9444

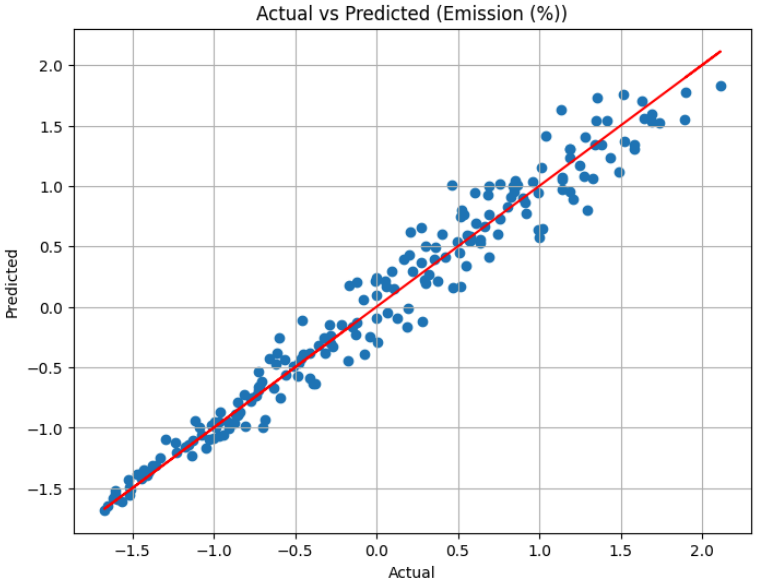
RMSE: 0.2289

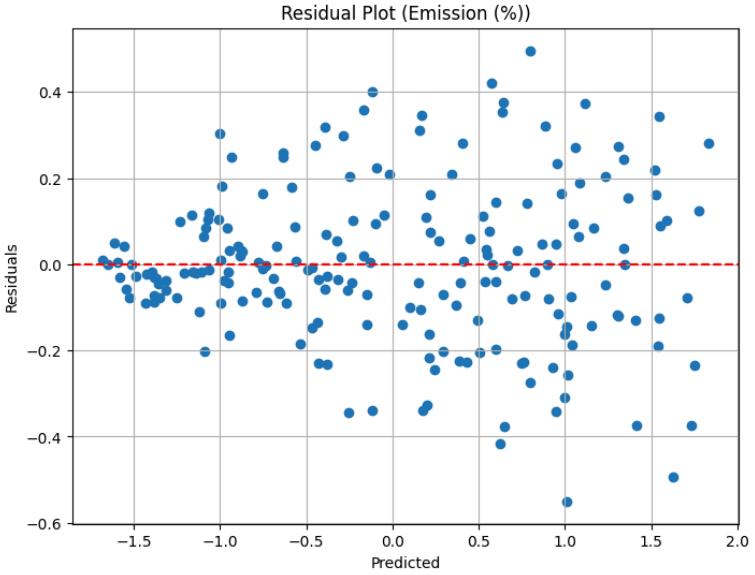
MAE: 0.1730

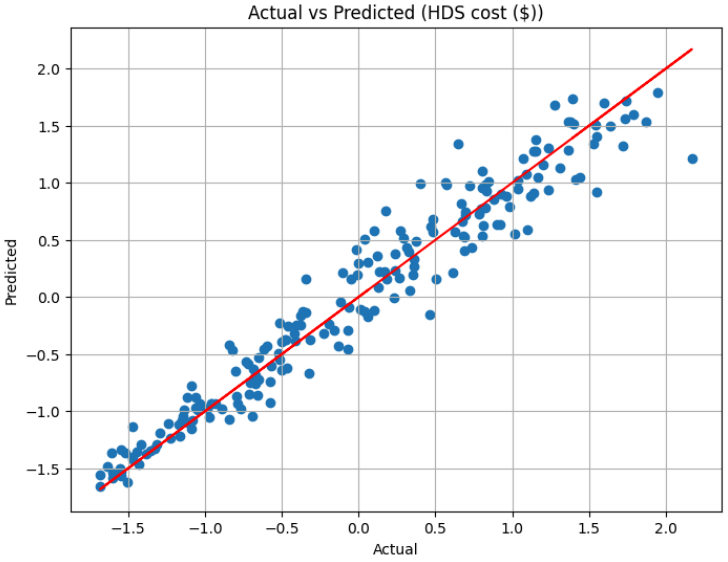
PLOTS:

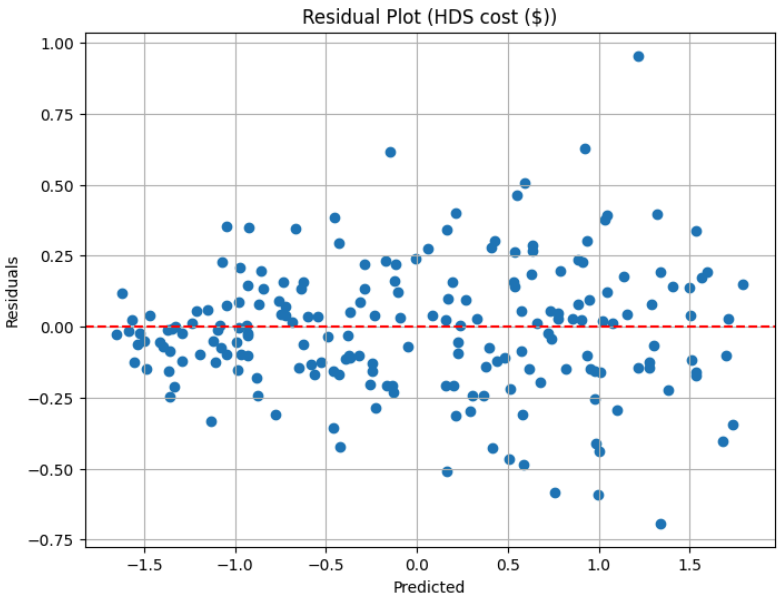





Actual vs Predicted (Emission (%))
Predicted
Actual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Residual Plot (Emission (%))
Residuals
Predicted
0.4
0.2
0.0
−0.2
−0.4
−0.6
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Actual vs Predicted (HDS cost ($))
Predicted
Actual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



RESULTS SUMMARY:

The performance of different machine learning models in predicting sulfur concentration (ppm), emission percentage (%), and HDS cost ($) for the catalytic sulfur removal process was evaluated using various metrics.

Multilinear Regression: Achieved **moderate performance** across all metrics, with R-squared values ranging from 0.8433 to 0.9428. It showed **good predictability for sulfur concentration and emission percentage but slightly lower performance for HDS cost.**

Multi-Layer Perceptron (Without Hyperparameter Tuning): Demonstrated **competitive performance** with R-squared values ranging from 0.7957 to 0.9617. It performed consistently **well across all metrics** and showed promising results for sulfur concentration, emission percentage, and HDS cost prediction.

Multi-Layer Perceptron (With Hyperparameter Tuning): **Improved performance compared to the untuned version,** achieving R-squared values ranging from 0.8388 to 0.9622. The tuning process optimized the model's hyperparameters, resulting in slightly better performance.

Gaussian Process Regression: Showed mixed performance, with R-squared values ranging from 0.5380 to 0.7953. It performed **relatively poorer** compared to other models, particularly for sulfur concentration prediction.

Multi-Task Lasso (Without Hyperparameter Tuning): Displayed **moderate performance**, with R-squared values ranging from 0.5067 to 0.6141. It showed **lower predictability** compared to other models, especially for sulfur concentration and HDS cost.

Multi-Task Lasso (With Hyperparameter Tuning): Demonstrated **improved performance after hyperparameter tuning**, achieving R-squared values ranging from 0.8001 to 0.9646. The tuning process enhanced the model's performance, particularly for sulfur concentration and emission percentage prediction.

It can be seen that performance of all models improve with hyperparameter tuning.

Best Model Selection:

Based on the performance metrics and overall consistency, the **Multi-Layer Perceptron (MLP) with hyperparameter tuning** appears to be the **best model for predicting sulfur concentration, emission percentage, and HDS cost**. It consistently achieved ***high R-squared values*** and showed promising results across all metrics. Therefore, the MLP with hyperparameter tuning should be selected for deployment in practical applications for catalytic sulfur removal optimization.

Comparative Analysis: COMPARISON WITH THE RESULTS MENTIONED IN RESEARCH PAPER

Following are the results of the original research paper.

Output	Sulfur concentration		Emission		HDS cost	
Models	Train	Test	Train	Test	Train	Test
MLP	0.999	0.983	0.999	0.971	1.0	1.0
GPR	0.999	0.942	0.999	0.864	0.999	0.990
MTL	0.757	0.790	0.818	0.738	1.0	0.999

Table 6. Error rates final models.

Output	Sulfur concentration		Emission		HDS cost	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE
MLP	7.69×10^{1}	9.05×10^{1}	3.41	4.23	5.57×10^{-1}	6.58×10^{-1}
GPR	1.23×10^{2}	1.59×10^{2}	7.58	9.27	1.69×10^{-4}	1.58×10^{-4}
MTL	3.11×10^{2}	3.76×10^{2}	8.10	9.58	1.38×10^{-2}	1.63×10^{-2}

Based on these results, the MLP model has clearly shown the best accuracy for all three outputs as indicated by the statistical parameters shown above. But due to the higher complexity of this model, this model is somewhat suspicious of overfitting due to the higher error rate at the cost of HDS. Therefore, due to the relative closeness of the R2 score of this model with GPR, as well as the lower error rate of **GPR**, this model has been selected for the analysis of the process.The datset used in the research paper contained only 40 datapoints obtained from real experiments. As already mentioned in the research paper, GPR has been chosen as the best model .

But in our analysis, **MLP** has been chosen as the best model as it performed comparatively better.

This difference has arisen as the datasets being used in both the analyses are different. Though I have considered the original dataset while generating the dataset, my dataset is still generated randomly, hence giving me different results.

Challenges and Limitations: The major challenge was generating the dataset that followed a normal gaussian distribution. Also since our dataset was random, the results deviated from real time experiments.

7 Conclusion and Future Work

Summary:

The project developed machine learning models to optimize sulfur removal in petroleum refining, specifically HDS. Models like MLP, GPR, MTL, and Multilinear Regression were effective in predicting sulfur concentration, SO2 emissions, and HDS cost, aiding in process optimization and environmental impact reduction.

The models improve HDS efficiency, reduce sulfur emissions, and lower operational costs, benefiting refineries in decision-making and environmental sustainability.

Based on the performance metrics and overall consistency, the **Multi-Layer Perceptron (MLP) with hyperparameter tuning** appears to be the **best model for predicting sulfur concentration, emission percentage, and HDS cost**. Therefore, the MLP with hyperparameter tuning should be selected for deployment in practical applications for catalytic sulfur removal optimization.

Future Directions:

1. Integration with Process Control Systems: Integrate models for dynamic process optimization.

2. Advanced Feature Engineering: Explore advanced techniques for better insights.

3. Model Ensemble Approaches: Combine models for improved accuracy and stability.

4. Incorporation of External Factors: Include external factors for comprehensive modeling.

5. Ethical and Social Implications: Investigate ethical implications and ensure fairness and transparency in decision-making.

8 References

https://www.sciencedirect.com/science/article/pii/S2214157X23011413#bib12

9 Appendices

APPENDIX - 1: ORIGINAL 40 POINT DATASET

10 Auxiliaries

Data Source:https://github.com/Nandiniii10/CL653-Project/blob/main/final_data.csv

Python file:https://colab.research.google.com/drive/1p-LPysML9AhgoRy_uPbioG3QZo3fIQCq?usp=sharing