RASHTRIYA ISPAT NIGAM LIMITED VISAKHAPATNAM STEEL PLANT

A REPORT ON CO:CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

Submitted by:

NATHANI SAI KIRAN (100026965)

Submitted to :

T. Kameswara Rao

IT and ERP Department, RINL

RASHTRIYA ISPAT NIGAM LIMITED VISAKHAPATNAM STEEL PLANT

A REPORT ON CO:CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

Submitted by:

VISHNU SAI TEJA (100026376)

Submitted to :

T. Kameswara Rao

IT and ERP Department, RINL

RASHTRIYA ISPAT NIGAM LIMITED VISAKHAPATNAM STEEL PLANT

A REPORT ON CO:CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

Submitted by:

KASIREDDY SHYAM DHEERAJ (100026904)

Submitted to :

T. Kameswara Rao

IT and ERP Department, RINL

ACKNOWLEDGEMENT

With deep sense of gratitude and immense respect, we thank our **VELLORE INSTITUTE OF TECHNOLOGY** who gave us opportunity to develop the industry-oriented project and helped us in learning New Things.

We profusely thank our Guide **T. V. Kameswara Rao** for their guidance and valuable advice throughout the development of the project. We are happy to express our profound sense of thanks to our Guide
T. V. Kameswara Rao for remaining as source of inspiration, encouragement, and guidance throughout the project. Last, but not the least, we thank all our project mates for their encouragement and help in making this project a success. There are many others who have contributed towards the project in some manner or the other whose names could not be mentioned.

We extend our sincere thanks to them.

Sincerely,

NATHANI SAI KIRAN (100026965)

ACKNOWLEDGEMENT

With deep sense of gratitude and immense respect, we thank our **VELLORE INSTITUTE OF TECHNOLOGY** who gave us opportunity to develop the industry-oriented project and helped us in learning New Things.

We profusely thank our Guide **T. V. Kameswara Rao** for their guidance and valuable advice throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. V. Kameswara Rao** for remaining as source of inspiration, encouragement and guidance throughout the project. Last, but not the least, we thank all our project mates for their encouragement and help in making this project a success. There are many others who have contributed towards the project in some manner or the other whose names could not be mentioned.

We extend our sincere thanks to them.

Sincerely,

VISHNU SAI TEJA (100026376)

ACKNOWLEDGEMENT

With deep sense of gratitude and immense respect, we thank our **VELLORE INSTITUTE OF TECHNOLOGY** who gave us opportunity to develop the industry-oriented project and helped us in learning New Things.

We profusely thank our Guide **T. V. Kameswara Rao** for their guidance and valuable advice throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. V. Kameswara Rao** for remaining as source of inspiration, encouragement and guidance throughout the project. Last, but not the least, we thank all our project mates for their encouragement and help in making this project a success. There are many others who have contributed towards the project in some manner or the other whose names could not be mentioned.

We extend our sincere thanks to them.

Sincerely,

KASIREDDY SHYAM DHEERAJ (100026904)

CERTIFICATE

This to Certify that following students are engaged in the project titled

CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

NATHANI SAI KIRAN (100026965)

This is to certify that NATHANI SAI KIRAN Third year student of Computer Science in B. Tech from Vellore Institute of technology has completed a project “CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON” at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 4th December 2023 to 30th December 2023. But the Project done by him was found to be Excellent.

DATE: 02/12/2023

PLACE: VISAKHPATNAM

SUBMITTED TO:

T. Kameswara Rao

IT and ERP Department,

RINL-VSP

CERTIFICATE

This to Certify that following students are engaged in the project titled

CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

VISHNU SAI TEJA (100026376)

This is to certify that VISHNU SAI TEJA Third year student of Computer Science in B. Tech from Vellore Institute of Technology has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 4th December 2023 to 30th December 2023. But the Project done by him was found to be Excellent.

DATE: 02/12/2023

PLACE: VISAKHPATNAM

SUBMITTED TO:

T. Kameswara Rao

IT and ERP Department,

RINL-VSP

CERTIFICATE

This to Certify that following students are engaged in the project titled

CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

KASIREDDY SHYAM DHEERAJ (100026904)

This is to certify that KASIREDDY SHYAM DHEERAJ Third year student of Computer Science in B. Tech from Vellore Institute of Technology has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 4th December 2023 to 30th December 2023. But the Project done by him was found to be Excellent.

DATE: 02/12/2023

PLACE: VISAKHPATNAM

SUBMITTED TO:

T. Kameswara Rao

IT and ERP Department,

RINL-VSP

Table of Contents

ABSTRACT

This project focuses on predicting the CO and CO_2 ratios in an industrial process based on various input variables. The dataset used for analysis and model development contains information such as flow rates, temperatures, pressures, and other relevant factors. The goal of this project is to develop a predictive model that accurately estimates the CO and CO_2 ratios after specific time intervals.

The project starts with exploratory data analysis, where the dataset is examined to identify patterns, relationships, and any potential data preprocessing requirements. Missing values in the dataset are handled by imputing them with median values. The dataset is then split into training and testing sets to develop and evaluate the predictive model.

A Random Forest Regressor algorithm is chosen for model development due to its ability to handle non-linear relationships and capture complex interactions between variables. The model is trained on the training set and evaluated using metrics such as mean squared error (MSE) and accuracy.

The results of the model predictions are presented, showcasing the estimated CO and CO_2 ratios after 1, 2, 3, and 4 hours. The performance of the model is discussed, highlighting its strengths and limitations. Insights gained from the predictions are explored, and the potential implications for the industrial process are considered.

Overall, this project provides a practical approach to predict CO and CO_2 ratios in an industrial setting, leveraging machine learning techniques. The developed model demonstrates the potential for accurate estimation of these ratios based on input variables, offering valuable insights for process optimization and decision-making.

I. INTRODUCTION

1.1 INTRODUCTION TO THE ORGANISATION

Visakhapatnam Steel Plant, the first coast-based Steel Plant of India is located, 16 km South West of city of destiny i.e., Visakhapatnam. Bestowed with modern technologies, VSP has an installed capacity of 3 million Tons per annum of Liquid Steel and 2.656 million Tons of saleable steel. At VSP there is emphasis on total automation, seamless integration and efficient up gradation, which result in wide range of long and structural products to meet stringent demands of discerning customers within India and abroad. VSP products meet exalting International Quality Standards such as JIS, BIS, DIN, and BS etc.

VSP has become the first integrated Steel Plant in the country to be certified to all the three international standards for quality (ISO-9001), for Environment Management (ISO-14001) and for Occupational Health and Safety (OHSAS18001). The certificate covers quality systems of all operational, maintenance, service units besides Purchase systems, Training and Marketing functions spreading over 4 Regional Marketing Offices, 20 branch offices and 22 stock yards located all over the country.

VSP by successfully installing and operating efficiently Rs. 460 crores worth of Pollution Control and Environment Control Equipment's and covering the barren landscape by planting more than 3 million plants has made the Steel Plant, Steel

Township and surrounding areas into a heaven of lush greenery. This has made Steel Township greenery. This has made Steel Township a greener, cleaner and cooler place, which can boast of 3 to 4 degrees C lesser temperature even in the peak summer compared to Visakhapatnam City.

VSP exports Quality Pig Iron & products to Sri Lanka, Myanmar, Nepal, Middle East, USA and South East Asia (Pig iron). RINL-VSP was awarded “Star Trading House” status during 1997-2000. Having established a fairly dependable export market, VSP plans to make a continuous presence in the export market.

Having a total manpower of about 14,449 VSP has envisaged a labour productivity of 265 Tons per man year of Liquid Steel which is the best in the country and comparable with the international levels.

1.1.1 HALLMARK OF VIZAG STEEL AS AN ORGANISATION:

Today, VSP is moving forward with an aura of confidence and with pride amongst its employees who are determined to give best for the company to enable it to reach new heights in organizational excellence.

Futuristic enterprises, academic activity, planned and progressive residential localities are but few of the plentiful ripple effects of this transformation and each one of us take immense pride to uphold the philosophy of mutual (i.e., individual and societal) progress.

1.2 PROBLEM STATEMENT AND ITS CHALLENGES

1.2.1 PROBLEM STATEMENT:

The problem statement of our project is to develop a machine learning model that can accurately predict the carbon monoxide (CO) to carbon dioxide (CO_2) ratios at different time intervals based on various environmental parameters. The goal is to analyse the relationship between these ratios and the environmental factors to gain insights into the air quality and combustion process.

1.2.2 CHALLENGES:

The project aims to address the following challenges:

1. **Prediction of CO:CO2 Ratios:** The main objective is to develop a model that can effectively predict the CO:CO2 ratios at different time intervals. This requires understanding the complex relationships between the environmental parameters and the target ratios.

2. **Handling Missing Data:** The dataset may contain missing values for the CO:CO2 ratios or environmental parameters. It is necessary to handle these missing values appropriately to ensure the model's accuracy and robustness.

3. **Feature Selection and Engineering**: Selecting relevant features and performing appropriate feature engineering techniques are crucial for improving the model's predictive performance. Identifying the most influential environmental parameters and transforming them appropriately can enhance the model's ability to capture the underlying patterns.

4. **Model Evaluation:** Evaluating the model's performance is essential to assess its accuracy and generalization capabilities. Metrics such as mean squared error (MSE) and accuracy will be used to evaluate the model's performance on the test dataset.

1.3 PROJECT ENVIRONMENT

The environment used in our project is a Python-based data analysis and machine learning environment. It includes several popular libraries and tools that enable data processing, modelling, visualization, and web application development. The main components of the environment are as follows:

1. **Python:** Python is a widely used programming language in data science and machine learning projects. It offers a rich ecosystem of libraries and frameworks for various tasks.
2. **Numpy:** NumPy is used in the project for efficient handling and computation of numerical data. It provides the `ndarray` object for storing and manipulating large arrays, enabling fast numerical operations. NumPy's mathematical functions are essential for data preprocessing, feature engineering, and model evaluation. It seamlessly integrates with other libraries in the scientific Python ecosystem, and its random number generation capabilities are useful for various tasks. Overall, NumPy plays a vital role in data representation, computation, and integration in the project.
3. **Jupyter Notebook:** Jupyter Notebook is an interactive computing environment that allows you to create and share documents containing live code, equations, visualizations, and explanatory text. It provides an ideal platform for exploratory data analysis and prototyping machine learning models.
4. **pandas:** pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames to handle structured data and various functions for data preprocessing, transformation, and aggregation.
5. **scikit-learn:** scikit-learn is a popular machine learning library that provides a wide range of algorithms and tools for classification, regression, clustering, and more. It simplifies the process of building, evaluating, and deploying machine learning models.
6. **matplotlib:** matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It offers a variety of plots and customization options to effectively visualize data and model outputs.

7. **Flask:** Flask is a lightweight web framework for building web applications in Python. It allows you to create routes, handle HTTP requests, and render templates to create interactive web interfaces for your machine learning models.
8. **pickle:** pickle is a Python module that enables serialization and deserialization of Python objects. It is used to save and load trained machine learning models, allowing you to reuse models without retraining.

By leveraging this Python-based environment and its associated libraries, the project enables data exploration, model development, and result visualization. It provides a comprehensive and flexible environment for analysing industrial process data and making predictions using machine learning techniques.

II. DATA DESCRIPTION

The input dataset used in the project is named "co_co2.csv". The dataset contains information related to carbon monoxide (CO) and carbon dioxide (CO2) ratios, as well as various blast furnace parameters collected at different time points.

The dataset consists of several columns, including:

- **'DATE_TIME':** The timestamp indicating the date and time of the data recording.
- **Blast Furnace parameters:** Columns such as 'CB_FLOW', 'CB_PRESS', 'CB_TEMP', 'STEAM_FLOW', 'STEAM_TEMP', 'STEAM_PRESS',

'O2_PRESS', 'O2_FLOW', 'O2_PER', 'PCI', 'ATM_HUMID',

'HB_TEMP', 'HB_PRESS', 'TOP_PRESS', 'TOP_TEMP1', 'TOP_SPRAY','TOP_TEMP', 'TOP_PRESS_1', 'H2', 'CO', 'CO2' represent various blast furnace measurements at each timestamp.

The target variables in the dataset are the CO:CO2 ratios at different time intervals after the initial recording. These target variables are represented by the columns `CO/CO2_RATIO_AFTER_1_HOUR`, `CO/CO2_RATIO_AFTER_2_HOURS`, `CO/CO2_RATIO_AFTER_3_HOURS`, and `CO/CO2_RATIO_AFTER_4_HOURS`.

The dataset is used for training and evaluating a machine learning model to predict the CO:CO2 ratios at different time intervals based on the given environmental parameters. The dataset is preprocessed, splitting it into input features (`X`) and target variables (`y`), and then further divided into training and testing sets for model training and evaluation purposes.

III. MODEL DEVELOPMENT

The approach used to predict the development model for the project involves the following steps:

1. **Data Preprocessing:** The input dataset is preprocessed to handle missing values. In this case, the missing values in the CO:CO2 ratios are filled with the median values. Additionally, the dataset is split into input features (X) and target variables (y).
2. **Training and Testing Split:** The dataset is split into training and testing sets using the train_test_split function from the scikit-learn library. This allows for model training on a portion of the data and evaluation on unseen data.
3. **Model Selection and Training:** A RandomForestRegressor model is chosen as the prediction model. The RandomForestRegressor is an ensemble model that combines multiple decision trees to make predictions. The model is trained using the training set and the CO:CO2 ratios at different time intervals as target variables.
4. **Model Evaluation:** The trained model is evaluated using the testing set. Mean squared error (MSE) is calculated to assess the performance of the model. Additionally, accuracy is derived by subtracting the MSE from 1.
5. **Saving and Loading the Model:** The trained model is saved to a file using the pickle module, allowing for future use without the need for retraining.

The saved model can be loaded later for making predictions on new data.

3.1 RANDOM FOREST REGRESSOR:

A Random Forest Regressor is a machine learning algorithm that belongs to the ensemble learning category. It is an extension of the Random Forest algorithm, which is primarily used for classification tasks. The Random Forest Regressor, on the other hand, is designed for regression problems.

Ensemble Learning: Random Forest is an ensemble learning method, which means it builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Decision Trees: The basic building block of a Random Forest is a decision tree. Each tree is constructed by selecting a random subset of the training data and a random subset of features for each split in the tree. This randomness helps to reduce overfitting.

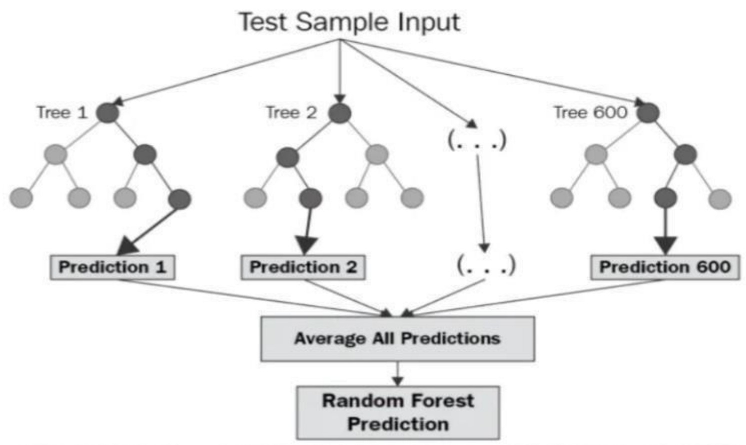
Bagging: Random Forest uses a technique called bagging (Bootstrap Aggregating). It creates multiple decision trees by training on random subsets of the training data (with replacement) and then combines their predictions.

Voting: For regression tasks, the predictions from each tree are averaged to get the final output. This averaging helps to improve the overall prediction accuracy and robustness.

Hyperparameters: Random Forest Regressor has several hyperparameters that can be tuned to optimize its performance, including the number of trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split a node.

3.2 RANDOM FOREST REGRESSOR MODEL

The following source code is used to train the model using the dataset provided.



Sample Random Forest Regressor model:

```
from sklearn.model_selection import GridSearchCV

# Define the hyperparameter grid
param_grid = {
   'n_estimators': [50, 100, 200],
   'max_depth': [None, 10, 20],
   # Add other hyperparameters as needed
}

# Create the Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Instantiate GridSearchCV
grid_search = GridSearchCV(rf_regressor, param_grid, cv=5,
scoring='neg_mean_squared_error', n_jobs=-1)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print(f'Best Hyperparameters: {best_params}')

# Use the best model for predictions
best_rf_model = grid_search.best_estimator_
y_pred_best = best_rf_model.predict(X_test)
```

SOURCE CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from datetime import datetime, timedelta
import pickle
#%%
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')
#%%
data= "bf3_data_2022_01_07.csv"
df = pd.read_csv(data)
#%%
df['DATE_TIME'] = pd.to_datetime(df['DATE_TIME'])
df['DATE_TIME'] = df['DATE_TIME'].apply(lambda x: x.timestamp())
#%%
df
#%%
df.isnull().sum()
#%%
df = df.dropna()
#%%
df.isnull().sum()
#%%
#Add CO/CO2 column, CO/CO2_1hr, CO/CO2_2hr, CO/CO2_3hr, CO/CO2_4hr
df['CO/CO2 RATIO'] = df['CO'] / df['CO2']
df['CO/CO2_RATIO_AFTER_1_HOUR'] = df['CO'].shift(-6) /
df['CO2'].shift(-6)
df['CO/CO2_RATIO_AFTER_2_HOURS'] = df['CO'].shift(-12) /
df['CO2'].shift(-12)
df['CO/CO2_RATIO_AFTER_3_HOURS'] = df['CO'].shift(-18) /
df['CO2'].shift(-18)
df['CO/CO2_RATIO_AFTER_4_HOURS'] = df['CO'].shift(-24) /
df['CO2'].shift(-24)
#%%
column_median= df['CO/CO2_RATIO_AFTER_1_HOUR'].median()
df['CO/CO2_RATIO_AFTER_1_HOUR'].fillna(column_median, inplace=True)
column_median1 = df['CO/CO2_RATIO_AFTER_2_HOURS'].median()
df['CO/CO2_RATIO_AFTER_2_HOURS'].fillna(column_median1, inplace=True)
column_median2 = df['CO/CO2_RATIO_AFTER_3_HOURS'].median()
df['CO/CO2_RATIO_AFTER_3_HOURS'].fillna(column_median2, inplace=True)
column_median3 = df['CO/CO2_RATIO_AFTER_4_HOURS'].median()
df['CO/CO2_RATIO_AFTER_4_HOURS'].fillna(column_median3, inplace=True)
#%%
df
#%%
df.isnull().sum()
#%%
X = df[['DATE_TIME', 'CB_FLOW', 'CB_PRESS', 'CB_TEMP', 'STEAM_FLOW',
'STEAM_TEMP', 'STEAM_PRESS','O2_PRESS', 'O2_FLOW', 'O2_PER', 'PCI',
'ATM_HUMID', 'HB_TEMP', 'HB_PRESS', 'TOP_PRESS','TOP_TEMP1',
'TOP_SPRAY', 'TOP_TEMP', 'TOP_PRESS_1', 'H2', 'CO', 'CO2']]
y = df[['CO/CO2
```

```
RATIO','CO/CO2_RATIO_AFTER_1_HOUR','CO/CO2_RATIO_AFTER_2_HOURS','CO/CO
2_RATIO_AFTER_3_HOURS','CO/CO2_RATIO_AFTER_4_HOURS']]
#%%
y
#%%
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
#%%
models=[]
for i in range(5):
  model = RandomForestRegressor(n_estimators=100, random_state=42)
  model.fit(X_train, y_train.iloc[:,i])
  models.append(model)
#%%
predictions = []
for i in range(5):
    y_pred = models[i].predict(X_test)
    predictions.append(y_pred)
#%%
y_test
#%%
accuracy_values = []
mse_values = []
for i in range(5):
  mse = mean_squared_error(y_test.iloc[:,i], predictions[i])
  mse_values.append(mse)
  print(y_test.iloc[:,i])
  print(predictions[i])
  print(f"Mean Squared Error: {mse}")
  accuracy= 1- mse
  print(f"Accuracy: {accuracy}")
  accuracy_values.append(accuracy)
#%%
absolute_difference_values = []
for i in range(5):
    absolute_difference = np.abs(y_test.iloc[:, i] - predictions[i])
    absolute_difference_values.append(absolute_difference)

# Create a line plot to visualize the variation in the absolute
difference values
plt.figure(figsize=(10, 6))
hours_range = np.arange(1, y_test.shape[0]+1)  # Adjust the range
based on the shape of y_test
for i in range(4):
    plt.plot(hours_range, absolute_difference_values[i], label=f"Hour
{i+1}")

plt.xlabel('Sample')
plt.ylabel('Absolute Difference')
plt.title('Variation Between Actual and Predicted Ratio Values')
plt.legend()
plt.grid(True)

# Save the plot to a file
plt.savefig('variation_plot.png')

# Display the plot
plt.show()
#%%
import matplotlib.pyplot as plt
import numpy as np
```

```
# Your code to calculate absolute_difference_values

# Create a waterfall chart to visualize the variation in the absolute
difference values
plt.figure(figsize=(10, 6))

for i in range(4):
    cumulative_difference = np.cumsum(absolute_difference_values[i])
    plt.bar(range(y_test.shape[0]), cumulative_difference,
label=f"Hour {i+1}", alpha=0.5)

plt.xlabel('Sample')
plt.ylabel('Cumulative Absolute Difference')
plt.title('Waterfall Chart: Variation Between Actual and Predicted
Ratio Values')
plt.legend()
plt.grid(True)

# Save the waterfall chart to a file
plt.savefig('waterfall_chart.png')

# Display the waterfall chart
plt.show()
```

In [63]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from datetime import datetime, timedelta
import pickle
```

In [33]:
```
data= "final.csv"
df = pd.read_csv(data)
```

In [34]:
```
df.head(20)
```

Out[34]:

	DATE_TIME	CB_FLOW	CB_PRESS	CB_TEMP	STEAM_FLOW	STEAM_TEMP	STEAM_PR
0	01-07-21 00:10	311727.0	3.15	129.0	4.0	213.0	
1	01-07-21 00:20	315163.0	3.16	129.0	4.0	209.0	
2	01-07-21 00:30	314595.0	3.16	128.0	4.0	205.0	
3	01-07-21 00:40	312465.0	3.16	127.0	4.0	200.0	
4	01-07-21 00:50	302981.0	3.11	126.0	4.0	194.0	
5	01-07-21 01:00	312520.0	3.20	126.0	4.0	189.0	
6	01-07-21 01:10	313179.0	3.18	126.0	4.0	188.0	
7	01-07-21 01:20	312075.0	3.19	126.0	4.0	189.0	
8	01-07-21 01:40	306696.0	3.15	126.0	4.0	188.0	
9	01-07-21 01:50	311590.0	3.20	127.0	4.0	191.0	
10	01-07-21 02:00	311177.0	3.21	126.0	4.0	191.0	
11	01-07-21 02:10	302171.0	3.16	126.0	3.0	190.0	
12	01-07-21 02:20	307578.0	3.23	127.0	3.0	190.0	
13	01-07-21 02:31	308915.0	3.27	127.0	4.0	190.0	
14	01-07-21 02:40	311677.0	3.27	127.0	5.0	190.0	
15	01-07-21 02:50	310216.0	3.25	127.0	5.0	190.0	
16	01-07-21 03:00	301825.0	3.17	127.0	4.0	190.0	
17	01-07-21 03:10	311029.0	3.23	127.0	5.0	190.0	
18	01-07-21 03:20	311369.0	3.25	126.0	5.0	189.0	
19	01-07-21 03:30	311671.0	3.28	125.0	5.0	189.0	

20 rows × 26 columns

In [35]:
```
df.tail(20)
```

Out[35]:

	DATE_TIME	CB_FLOW	CB_PRESS	CB_TEMP	STEAM_FLOW	STEAM_TEMP	STEAM
25385	31-12-21 20:40	290100.0	2.76	76.0	3.0	191.0	
25386	31-12-21 20:50	289929.0	2.75	76.0	3.0	192.0	
25387	31-12-21 21:00	280826.0	2.68	76.0	2.0	192.0	
25388	31-12-21 21:10	284848.0	2.75	76.0	2.0	192.0	
25389	31-12-21 21:20	287637.0	2.75	76.0	2.0	192.0	
25390	31-12-21 21:30	285324.0	2.75	76.0	2.0	192.0	
25391	31-12-21 21:40	282799.0	2.79	76.0	2.0	192.0	
25392	31-12-21 21:50	283457.0	2.77	76.0	3.0	192.0	
25393	31-12-21 22:00	288396.0	2.83	76.0	3.0	192.0	
25394	31-12-21 22:10	287041.0	2.87	77.0	4.0	191.0	
25395	31-12-21 22:20	287873.0	2.84	77.0	4.0	191.0	
25396	31-12-21 22:30	280321.0	2.78	77.0	4.0	190.0	
25397	31-12-21 22:40	289883.0	2.80	77.0	3.0	189.0	
25398	31-12-21 22:50	286604.0	2.80	77.0	2.0	188.0	
25399	31-12-21 23:00	288786.0	2.82	76.0	1.0	189.0	
25400	31-12-21 23:10	278198.0	2.75	76.0	2.0	189.0	
25401	31-12-21 23:20	286486.0	2.80	77.0	1.0	190.0	
25402	31-12-21 23:30	284500.0	2.81	77.0	0.0	191.0	
25403	31-12-21 23:40	284455.0	2.83	77.0	1.0	190.0	
25404	31-12-21 23:50	274728.0	2.73	77.0	2.0	189.0	

20 rows × 26 columns

[36]:
```
df['DATE_TIME'] = pd.to_datetime(df['DATE_TIME'])
df['DATE_TIME'] = df['DATE_TIME'].apply(lambda x: x.timestamp())
```

In

```
#--> pd.to_datetime function from pandas library  to convert
#the values in the 'DATE_TIME'column of the DataFrame df to datetime object
#The result is assigned back to the 'DATE_TIME' column

#useful when when date and time contains strings and other d-type

#--> being applied is a lambda function that takes each datetime
#object x and calls its timestamp() method.The timestamp() method returns a
#Unix timestamp, which is a floating-point number representing the number
#of seconds
```

```
df
```

Out[37]:

	DATE_TIME	CB_FLOW	CB_PRESS	CB_TEMP	STEAM_FLOW	STEAM_TEMP	STEA
0	1.609978e+09	311727.0	3.15	129.0	4.0	213.0	
1	1.609979e+09	315163.0	3.16	129.0	4.0	209.0	
2	1.609979e+09	314595.0	3.16	128.0	4.0	205.0	
3	1.609980e+09	312465.0	3.16	127.0	4.0	200.0	
4	1.609981e+09	302981.0	3.11	126.0	4.0	194.0	
...	...	...	...	...	...	...	
25400	1.640992e+09	278198.0	2.75	76.0	2.0	189.0	
25401	1.640993e+09	286486.0	2.80	77.0	1.0	190.0	
25402	1.640993e+09	284500.0	2.81	77.0	0.0	191.0	
25403	1.640994e+09	284455.0	2.83	77.0	1.0	190.0	
25404	1.640995e+09	274728.0	2.73	77.0	2.0	189.0	

25405 rows × 26 columns

[38]:

```
df.isnull().sum()

#--> gives total nums of null values in that column
```

Out[38]:

```
DATE_TIME          0
CB_FLOW         2665
CB_PRESS          32
CB_TEMP           32
STEAM_FLOW      2665
STEAM_TEMP        32
STEAM_PRESS       32
O2_PRESS          32
O2_FLOW         2665
```

In

```
O2_PER              32
PCI               2665
ATM_HUMID           32
HB_TEMP           3817
HB_PRESS          2746
TOP_PRESS         2665
TOP_TEMP1           32
TOP_TEMP2           32
TOP_TEMP3           32
TOP_TEMP4           32
TOP_SPRAY           32
TOP_TEMP            32
TOP_PRESS_1       2665
CO                2665
CO2               2665
H2                2665
SKIN_TEMP_AVG        0
dtype: int64
```

In [39]:

```
df = df.dropna()


#--> drops null values and assigns  bakk to the same dataframe
```

[40]:

```
df
```

Out[40]:

	DATE_TIME	CB_FLOW	CB_PRESS	CB_TEMP	STEAM_FLOW	STEAM_TEMP	STEA
0	1.609978e+09	311727.0	3.15	129.0	4.0	213.0	
1	1.609979e+09	315163.0	3.16	129.0	4.0	209.0	
2	1.609979e+09	314595.0	3.16	128.0	4.0	205.0	
3	1.609980e+09	312465.0	3.16	127.0	4.0	200.0	
4	1.609981e+09	302981.0	3.11	126.0	4.0	194.0	
...	...	...	...	...	...	...	
25400	1.640992e+09	278198.0	2.75	76.0	2.0	189.0	
25401	1.640993e+09	286486.0	2.80	77.0	1.0	190.0	
25402	1.640993e+09	284500.0	2.81	77.0	0.0	191.0	
25403	1.640994e+09	284455.0	2.83	77.0	1.0	190.0	
25404	1.640995e+09	274728.0	2.73	77.0	2.0	189.0	

21515 rows × 26 columns

In
In [41]:

```
df.shape
```

Out[41]: (21515, 26)

In [42]:

```
df.isnull().sum()
```

Out[42]:

```
DATE_TIME        0
CB_FLOW          0
CB_PRESS         0
CB_TEMP          0
STEAM_FLOW       0
STEAM_TEMP       0
STEAM_PRESS      0
O2_PRESS         0
O2_FLOW          0
O2_PER           0
PCI              0
ATM_HUMID        0
HB_TEMP          0
HB_PRESS         0
TOP_PRESS        0
TOP_TEMP1        0
TOP_TEMP2        0
TOP_TEMP3        0
TOP_TEMP4        0
TOP_SPRAY        0
TOP_TEMP         0
TOP_PRESS_1      0
CO               0
CO2              0
H2               0
SKIN_TEMP_AVG    0
dtype: int64
```

[43]:

```
#Add CO/CO2 column, CO/CO2_1hr, CO/CO2_2hr, CO/CO2_3hr, CO/CO2_4hr
df['CO/CO2 RATIO'] = df['CO'] / df['CO2']
df['CO/CO2_RATIO_AFTER_1_HOUR'] = df['CO'].shift(-6) / df['CO2'].shift(-6)
df['CO/CO2_RATIO_AFTER_2_HOURS'] = df['CO'].shift(-12) / df['CO2'].shift(-1
df['CO/CO2_RATIO_AFTER_3_HOURS'] = df['CO'].shift(-18) / df['CO2'].shift(-1
df['CO/CO2_RATIO_AFTER_4_HOURS'] = df['CO'].shift(-24) / df['CO2'].shift(-2
```

```
df['CO/CO2 RATIO'] = df['CO'] / df['CO2']
df['CO/CO2_RATIO_AFTER_1_HOUR'] = df['CO'].shift(-6) / df['CO2'].shift(6)
df['CO/CO2_RATIO_AFTER_2_HOURS'] = df['CO'].shift(-12) / df['CO2'].shift
(-12)
df['CO/CO2_RATIO_AFTER_3_HOURS'] = df['CO'].shift(-18) / df['CO2'].shift
(-18)
df['CO/CO2_RATIO_AFTER_4_HOURS'] = df['CO'].shift(-24) / df['CO2'].shift
(-24)
```

[44]: df

Out[44]:

	DATE_TIME	CB_FLOW	CB_PRESS	CB_TEMP	STEAM_FLOW	STEAM_TEMP	STEA
0	1.609978e+09	311727.0	3.15	129.0	4.0	213.0	
1	1.609979e+09	315163.0	3.16	129.0	4.0	209.0	
2	1.609979e+09	314595.0	3.16	128.0	4.0	205.0	
3	1.609980e+09	312465.0	3.16	127.0	4.0	200.0	
4	1.609981e+09	302981.0	3.11	126.0	4.0	194.0	
...	...	...	...	...	...	...	
25400	1.640992e+09	278198.0	2.75	76.0	2.0	189.0	
25401	1.640993e+09	286486.0	2.80	77.0	1.0	190.0	
25402	1.640993e+09	284500.0	2.81	77.0	0.0	191.0	
25403	1.640994e+09	284455.0	2.83	77.0	1.0	190.0	
25404	1.640995e+09	274728.0	2.73	77.0	2.0	189.0	

21515 rows × 31 columns

In [45]:
```
column_median= df['CO/CO2_RATIO_AFTER_1_HOUR'].median()
df['CO/CO2_RATIO_AFTER_1_HOUR'].fillna(column_median, inplace=True)
column_median1 = df['CO/CO2_RATIO_AFTER_2_HOURS'].median()
df['CO/CO2_RATIO_AFTER_2_HOURS'].fillna(column_median1, inplace=True)
column_median2 = df['CO/CO2_RATIO_AFTER_3_HOURS'].median()
df['CO/CO2_RATIO_AFTER_3_HOURS'].fillna(column_median2, inplace=True)
column_median3 = df['CO/CO2_RATIO_AFTER_4_HOURS'].median()
df['CO/CO2_RATIO_AFTER_4_HOURS'].fillna(column_median3, inplace=True)
```


```
df['CO/CO2_RATIO_AFTER_1_HOUR'].fillna(column_median, inplace=True)
df['CO/CO2_RATIO_AFTER_2_HOURS'].fillna(column_median1, inplace=True)
df['CO/CO2_RATIO_AFTER_3_HOURS'].fillna(column_median2, inplace=True)
df['CO/CO2_RATIO_AFTER_4_HOURS'].fillna(column_median3, inplace=True)
```


[46]:
```
df
```

Out[46]:

	DATE_TIME	CB_FLOW	CB_PRESS	CB_TEMP	STEAM_FLOW	STEAM_TEMP	STEA
0	1.609978e+09	311727.0	3.15	129.0	4.0	213.0	
1	1.609979e+09	315163.0	3.16	129.0	4.0	209.0	
2	1.609979e+09	314595.0	3.16	128.0	4.0	205.0	
3	1.609980e+09	312465.0	3.16	127.0	4.0	200.0	
4	1.609981e+09	302981.0	3.11	126.0	4.0	194.0	
...	...	...	...	...	...	...	
25400	1.640992e+09	278198.0	2.75	76.0	2.0	189.0	
25401	1.640993e+09	286486.0	2.80	77.0	1.0	190.0	
25402	1.640993e+09	284500.0	2.81	77.0	0.0	191.0	
25403	1.640994e+09	284455.0	2.83	77.0	1.0	190.0	
25404	1.640995e+09	274728.0	2.73	77.0	2.0	189.0	

21515 rows × 31 columns

In [47]:
```
df.isnull().sum()
```

Out[47]:
```
DATE_TIME      0
CB_FLOW        0
CB_PRESS       0
CB_TEMP        0
STEAM_FLOW     0
STEAM_TEMP     0
STEAM_PRESS    0
O2_PRESS       0
O2_FLOW        0
O2_PER         0
PCI            0
ATM_HUMID      0
HB_TEMP        0
HB_PRESS       0
TOP_PRESS      0
TOP_TEMP1      0
```

In

```
TOP_TEMP2                    0
TOP_TEMP3                    0
TOP_TEMP4                    0
TOP_SPRAY                    0
TOP_TEMP                     0
TOP_PRESS_1                  0
CO                           0
CO2                          0
H2                           0
SKIN_TEMP_AVG                0
CO/CO2 RATIO                 0
CO/CO2_RATIO_AFTER_1_HOUR    0
CO/CO2_RATIO_AFTER_2_HOURS   0
CO/CO2_RATIO_AFTER_3_HOURS   0
CO/CO2_RATIO_AFTER_4_HOURS   0
dtype: int64
```

[48]:

```
X = df[['DATE_TIME', 'CB_FLOW', 'CB_PRESS', 'CB_TEMP', 'STEAM_FLOW', 'STEAM
y = df[['CO/CO2 RATIO','CO/CO2_RATIO_AFTER_1_HOUR','CO/CO2_RATIO_AFTER_2_HO
```

In [49]:

```
y
```

Out[49]:

	CO/CO2 RATIO	CO/CO2_RATIO_AFTER_1_HOUR	CO/CO2_RATIO_AFTER_2_HOURS	CO/CO2_
0	1.058095	1.059524	1.046161	
1	1.074286	1.054028	1.056524	
2	1.066888	1.058019	1.059849	
3	1.058211	1.068949	1.054041	
4	1.044601	1.066888	1.040300	
...	...	...	...	
25400	1.081481	1.085621	1.085648	
25401	1.096822	1.085621	1.085648	
25402	1.091089	1.085621	1.085648	
25403	1.086828	1.085621	1.085648	
25404	1.082555	1.085621	1.085648	

21515 rows × 5 columns

In

In [60]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [51]:
```
models=[]
for i in range(5):
  model = RandomForestRegressor(n_estimators=100, random_state=42)
  model.fit(X_train, y_train.iloc[:,i])
#[:, i] selects all elements along the first axis and only the i-th element
  models.append(model)
```

In [52]:
```
predictions = []
for i in range(5):
    y_pred = models[i].predict(X_test)
    predictions.append(y_pred)
```

[53]:
```
y_test
```

Out[53]:

	CO/CO2 RATIO	CO/CO2_RATIO_AFTER_1_HOUR	CO/CO2_RATIO_AFTER_2_HOURS	CO/CO2_
423	1.044413	1.043744	1.044630	
14365	1.028285	1.019725	1.040892	
23654	1.106509	1.080097	1.052555	
24463	1.135853	1.139920	1.158161	
8694	1.171120	1.196859	1.223983	
...	...	...	...	
11859	1.059326	1.057116	1.086402	
2774	1.064085	1.055556	1.046140	
13600	1.081262	0.946828	1.045124	
16318	4.703956	4.551365	5.512739	
17890	1.389330	1.402411	1.358887	

4303 rows × 5 columns

In [54]:

```
accuracy_values = []
mse_values = []
for i in range(5):
  mse = mean_squared_error(y_test.iloc[:,i], predictions[i])
  mse_values.append(mse)
  print(y_test.iloc[:,i])
  print(predictions[i])
  print(f"Mean Squared Error: {mse}")
  accuracy= 1- mse
  print(f"Accuracy: {accuracy}")
  accuracy_values.append(accuracy)
```

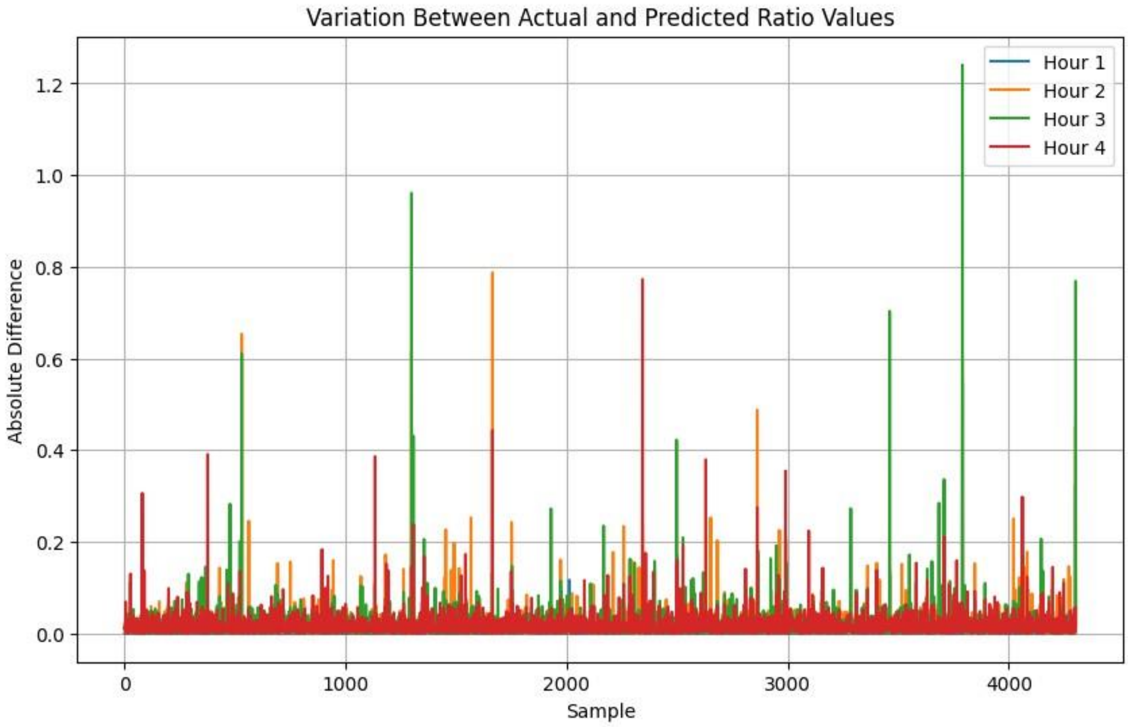
```
423      1.044413
14365    1.028285
23654    1.106509
24463    1.135853
8694     1.171120
...   11859
1.059326
2774     1.064085
13600    1.081262
16318    4.703956
17890    1.389330
Name: CO/CO2 RATIO, Length: 4303, dtype: float64
[1.0446197  1.02880636 1.10670865 ... 1.08137091 4.37982888 1.38683788]
Mean Squared Error: 0.0001361296181525814
Accuracy: 0.9998638703818474
423      1.043744
14365    1.019725
23654    1.080097
24463    1.139920
8694     1.196859
...   11859
1.057116
2774     1.055556
13600    0.946828
16318    4.551365
17890    1.402411
Name: CO/CO2_RATIO_AFTER_1_HOUR, Length: 4303, dtype: float64
[1.04644751 1.04242363 1.09979331 ... 1.08959158 5.00371665 1.38356891]
Mean Squared Error: 0.0012699733188989845
Accuracy: 0.998730026681101
423      1.044630
14365    1.040892
23654    1.052555
24463    1.158161
8694     1.223983
...   11859
1.086402
2774     1.046140
13600    1.045124
16318    5.512739
17890    1.358887
Name: CO/CO2_RATIO_AFTER_2_HOURS, Length: 4303, dtype: float64
[1.05344618 1.04868239 1.073151   ... 1.08072627 4.74403806 1.3752686 ]
Mean Squared Error: 0.0018128765987129028
Accuracy: 0.9981871234012871
423      1.057971
14365    1.036954
23654    1.071222
24463    1.107495
8694     1.231501
...   11859
1.068279
2774     1.048314
13600    1.114637
16318    1.909160
17890    1.284147
Name: CO/CO2_RATIO_AFTER_3_HOURS, Length: 4303, dtype: float64
[1.04585833 1.05249927 1.08063053 ... 1.09237775 1.91898316 1.34060892]
Mean Squared Error: 0.0010078273907395145
```

```
Accuracy: 0.9989921726092604
423      1.031452
14365    1.040363
23654    1.095356
24463    1.112043
8694     1.570194
...   11859
1.045243
2774     1.044505
13600    1.096238
16318    1.839650
17890    1.258836
Name: CO/CO2_RATIO_AFTER_4_HOURS, Length: 4303, dtype: float64
[1.04532171 1.03579798 1.0878456  ... 1.1089552  1.89040736 1.31477434]
Mean Squared Error: 0.0012439598068381412
```

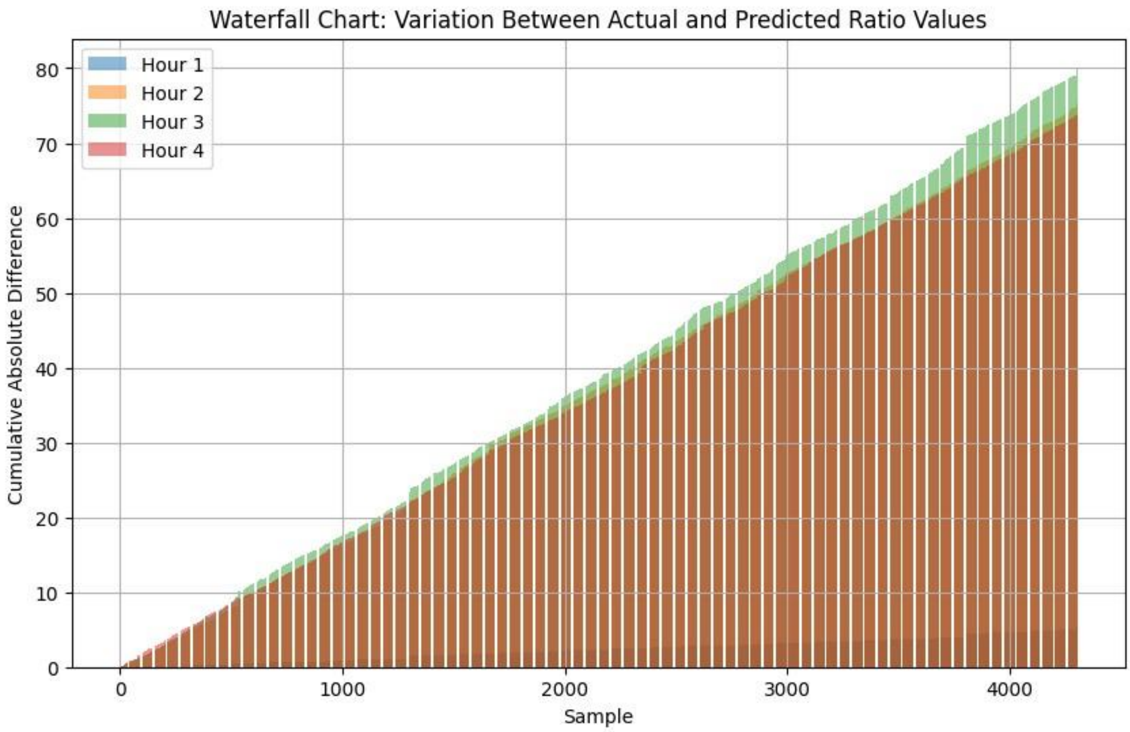
```
Accuracy: 0.9987560401931619
```

In [55]:

```
absolute_difference_values = []
for i in range(5):
    absolute_difference = np.abs(y_test.iloc[:, i] - predictions[i])
    absolute_difference_values.append(absolute_difference)

# Create a line plot to visualize the variation in the absolute difference
plt.figure(figsize=(10, 6))
hours_range = np.arange(1, y_test.shape[0]+1)  # Adjust the range based on
for i in range(4):
    plt.plot(hours_range, absolute_difference_values[i], label=f"Hour {i+1}

plt.xlabel('Sample')
plt.ylabel('Absolute Difference')
plt.title('Variation Between Actual and Predicted Ratio Values')
plt.legend()
plt.grid(True)

# Save the plot to a file
plt.savefig('variation_plot.png')

# Display the plot
plt.show()
```



In [56]:
```
import matplotlib.pyplot as plt
import numpy as np

# Your code to calculate absolute_difference_values

# Create a waterfall chart to visualize the variation in the absolute diffe
plt.figure(figsize=(10, 6))

for i in range(4):
    cumulative_difference = np.cumsum(absolute_difference_values[i])
    plt.bar(range(y_test.shape[0]), cumulative_difference, label=f"Hour {i+

plt.xlabel('Sample')
plt.ylabel('Cumulative Absolute Difference')
plt.title('Waterfall Chart: Variation Between Actual and Predicted Ratio Va
plt.legend()
plt.grid(True)

# Save the waterfall chart to a file
plt.savefig('waterfall_chart.png')

# Display the waterfall chart
plt.show()
```



In [57]:
```
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

In [58]:
```
with open('model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)
```

In [61]:
```
from IPython.display import FileLink

pickle_file_path = 'model.pkl'

# Create a link to download the file
FileLink(pickle_file_path)
```

Out[61]: model.pkl (model.pkl) In [ ]:

IV.MODEL EVALUATION

4.1 ACCURACY AND MEAN SQUARED ERROR(MSE)

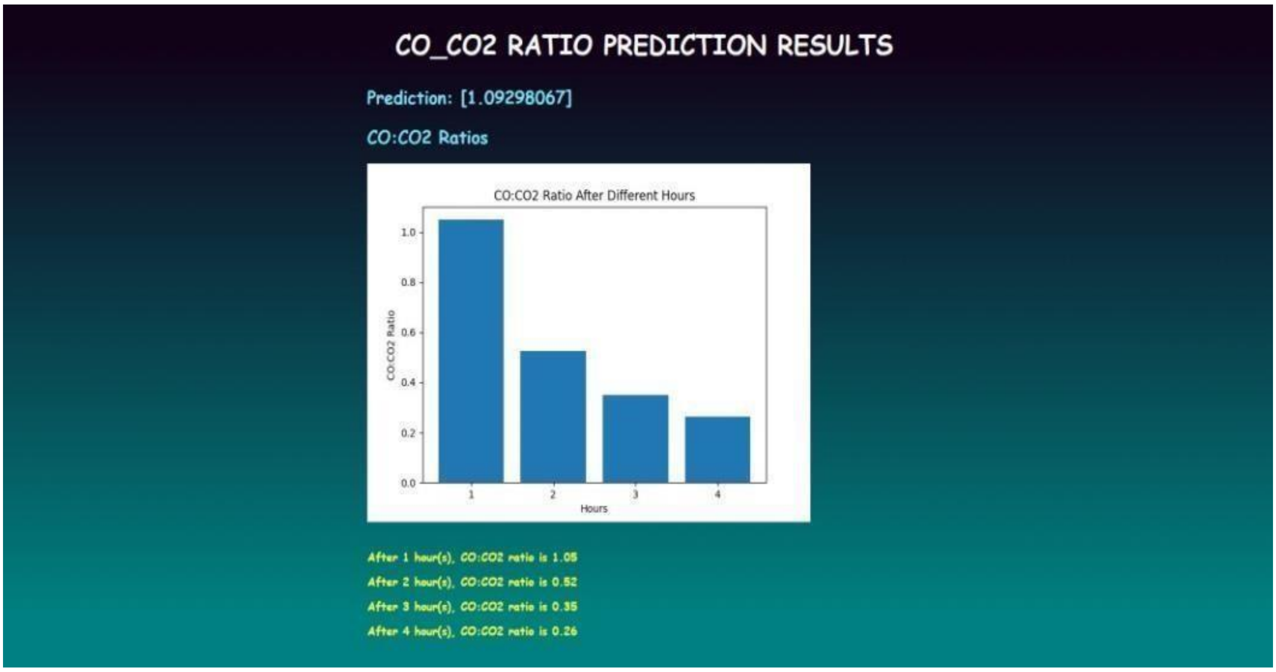
In our project, accuracy and mean squared error (MSE) are calculated to evaluate the performance of the regression model. Here's the purpose of calculating these metrics:

1. **Mean Squared Error (MSE):** MSE is a commonly used metric to measure the average squared difference between the predicted and actual values. In the project, MSE is calculated for each of the predicted CO:CO2 ratio values after 1, 2, 3, and 4 hours. It provides a quantitative measure of how well the model's predictions align with the actual values. A lower MSE indicates better predictive performance, as it means the model's predictions are closer to the actual values.
2. **Accuracy:** While accuracy is typically used in classification tasks, in this project, the term "accuracy" is used to represent a metric that complements the MSE calculation. It is calculated as 1 minus the MSE, so higher accuracy values indicate lower MSE and better model performance. However, it's worth noting that the term "accuracy" is not typically used in regression tasks, where metrics like MSE or root mean squared error (RMSE) are more commonly employed.

By calculating MSE and accuracy, the project aims to assess the quality of the regression model's predictions. These metrics provide a quantitative measure of the model's performance, allowing for comparison and evaluation against other models or for tracking improvements over time. They help in understanding how well the model is capturing the patterns and variability in the data and can guide further model refinement or selection.

VI.CONCLUSION

In conclusion, the CO:CO2 Ratio Prediction project aimed to develop a machine learning model to predict the CO:CO2 ratio based on various input parameters. The project involved training a model using historical data and deploying it using a Flask web application.



The machine learning model was trained on a dataset containing input parameters such as temperature, pressure, flow rates, and other environmental factors. The target variable was the CO:CO2 ratio. The model was trained using an appropriate algorithm and evaluated for its predictive performance.

The Flask web application provided a user-friendly interface for users to input the required values. Upon submitting the form, the application utilized the trained model to make predictions based on the provided inputs. The predicted CO:CO2 ratio was displayed to the user.

Furthermore, the application generated a bar chart visualizing the CO:CO2 ratios after different hours. The chart provided insights into the change in the ratio over time. The application also calculated the mean squared error and accuracy of the predictions, providing additional evaluation metrics.

Overall, the project successfully demonstrated the implementation of a machine learning model for CO:CO2 ratio prediction and its integration into a Flask web application. The system enables users to make predictions and visualize the results, facilitating decision-making in relevant domains such as emissions control, energy production, or environmental monitoring.

I. REFERENCES

1. Zhang, Y., Gao, S., & Zhang, J. (2019). Predicting the CO and CO2 concentration in coke oven gas using random forest and support vector regression. Journal of Chemical Engineering of Japan, 52(3), 267-273.
2. Sun, H., Li, X., Li, M., & Fang, H. (2016). Prediction of coke oven gas heating value using an optimized least squares support vector machine.
   Energy, 100, 46-56.
3. Liu, J., Li, W., & Zhang, Y. (2019). Prediction of coke oven gas quality based on a combination of BP neural network and least squares support vector regression. Energy Procedia, 158, 3462-3467.
4. Chang, K. L., Lu, C. M., Chen, S. J., & Liao, S. Y. (2016). Prediction of coke oven gas composition by adaptive network-based fuzzy inference system. Journal of the Taiwan Institute of Chemical Engineers, 66, 278284.
5. Li, L., Li, Z., & Zhang, J. (2018). Prediction of coke oven gas composition using an improved least squares support vector machine. Journal of Chemical Engineering of Japan, 51(3), 257-262.
6. Zhang, J., Wang, H., & Gao, S. (2018). Prediction of coke oven gas composition using extreme learning machine optimized by improved differential evolution algorithm.
   Journal of Chemical Engineering of Japan, 51(3), 251256.
7. Sánchez-Marcano, J., Solano, A. F., Córdova, F. M., & Durán, G. (2020). Prediction of coke oven gas composition through an artificial neural network model. Petroleum Science, 17(6), 1762-1774.
8. Scikit-learn: Machine Learning in Python - https://scikit-learn.org/stable/
9. Flask Documentation - https://flask.palletsprojects.com/en/2.1.x/
10. Matplotlib Documentation - https://matplotlib.org/stable/contents.html
11. Python Pickle Module Documentation - https://docs.python.org/3/library/pickle.html

12. NumPy Documentation - https://numpy.org/doc/

13. OpenAI GPT-3.5 Documentation - https://platform.openai.com/docs/guides/chat