RASHTRIYA ISPAT NIGAM LIMITED VISAKHAPATNAM STEEL PLANT

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

Submitted by:

PADALA NAVYANTH REDDY (100026308)

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:

KOLUPOTI SAI PRAKASH (100026639)

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:

JATHIN R (100026685)

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:

THATOLU MANI ABHINAV(100026269)

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:

SUSHMITHA KOLUKULURU (100026686)

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:

KODAVALLA SREE VENKAT (100026232)

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:

MALLAMPATI BHAVISHYA (100026606)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

PADALA NAVYANTH REDDY (100026308)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

KOLUPOTI SAI PRAKASH (100026639)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

JATHIN R (100026685)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

THATOLU MANI ABHINAV(100026269)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

SUSHMITHA KOLUKULURU (100026686)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

KODAVALLA SREE VENKAT (100026232)

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 .Kameswara Rao** for their guidance and valuable advices throughout the development of the project. We are happy to express our profound sense of thanks to our Guide **T. 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,

MALLAMPATI BHAVISHYA (100026606)

CERTIFICATE

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

CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON

PADALA NAVYANTH REDDY (100026308)

This is to certify that PADALA NAVYANTH REDDY Third year student of Computer Science in Btech from vellore instiitute of technology has completed a project “CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON” at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd 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

KOLUPOTI SAI PRAKASH (100026639)

This is to certify that KOLUPOTI SAI PRAKASH Third year student of Computer Science in Btech from vellore instituite of technology has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd 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

JATHIN R (100026685)

This is to certify that JATHIN R Third year student of Computer Science in Btech from vellore instiiute of technology has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd 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

THATOLU MANI ABHINAV(100026269)

This is to certify that THATOLU MANI ABHINAV Third year student of Computer Science in Btech from vellore instituite of technology(CHENNAI) has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd December 2023. But the Project done by him was found to be Excellent.

DATE : 02/12/2023

SUBMITTED TO:

PLACE:VISAKHPATNAM

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

SUSHMITHA KOLUKULURU (100026686)

This is to certify that SUSHMITHA KOLUKULURU Third year student of Computer Science in Btech from vellore instituite of technology has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd 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

KODAVALLA SREE VENKAT (100026232)

This is to certify that KODAVALLA SREE VENKAT Third year student of Computer Science in Btech from vellore instituite of technology has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd 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

MALLAMPATI BHAVISHYA (100026606)

This is to certify that MALLAMPATI BHAVISHYA Third year student of Computer Science in Btech from vellore instiitute of technology(A.P) has completed a project "CO_CO2 RATIO PREDICTION USING MACHINE LEARNING BY PYTHON" at RINL, VISAKHAPATNAM STEEL PLANT for 4 weeks from 6th November 2023 to 2nd 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 CO2 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 CO2 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 CO2 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 CO2 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 &amp; 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.

2.1.1 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 (CO2) 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.

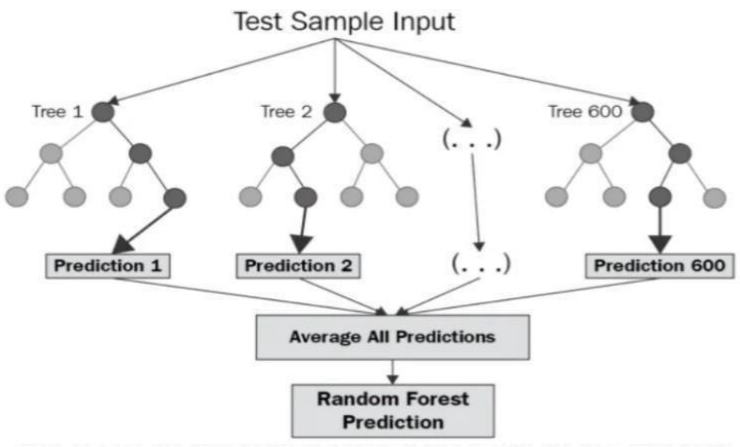
The saved model can be loaded later for making predictions on new data.

3.1 RANDOM FOREST REGRESSOR:

Random Forest Regression is a supervised learning algorithm that uses **ensemble learning** method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

3.2 RANDOM FOREST REGRESSOR MODEL

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



Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Decision trees

Since the random forest model is made up of multiple decision trees, it would be helpful to start by describing the decision tree algorithm briefly. Decision trees start with a basic question, such as, “Should I surf?” From there, you can ask a series of questions to determine an answer, such as, “Is it a long period swell?” or “Is the wind blowing offshore?”. These questions make up the decision nodes in the tree, acting as a means to split the data. Each question helps an individual to arrive at a final decision, which would be denoted by the leaf node. Observations that fit the criteria will follow the “Yes” branch and those that don’t will follow the alternate path. Decision trees seek to find the best split to subset the data, and they are typically trained through the Classification and Regression Tree (CART) algorithm. Metrics, such as Gini impurity, information gain, or mean square error (MSE), can be used to evaluate the quality of the split.

This decision tree is an example of a classification problem, where the class labels are "surf" and "don't surf."

While decision trees are common supervised learning algorithms, they can be prone to problems, such as bias and overfitting. However, when multiple decision trees form an ensemble in the random forest algorithm, they predict more accurate results, particularly when the individual trees are uncorrelated with each other.

Ensemble methods

Ensemble learning methods are made up of a set of classifiers—e.g. decision trees—and their predictions are aggregated to identify the most popular result. The most well-known ensemble methods are bagging, also known as bootstrap aggregation, and boosting. In 1996, Leo Breiman introduced the bagging method; in this method, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. After several data samples are generated, these models are then trained independently, and depending on the type of task—i.e. regression or classification—the average or majority of those predictions yield a more accurate estimate. This approach is commonly used to reduce variance within a noisy dataset.

Random forest algorithm

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. Feature randomness, also known as feature bagging or “the random subspace method” generates a random subset of features, which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features.

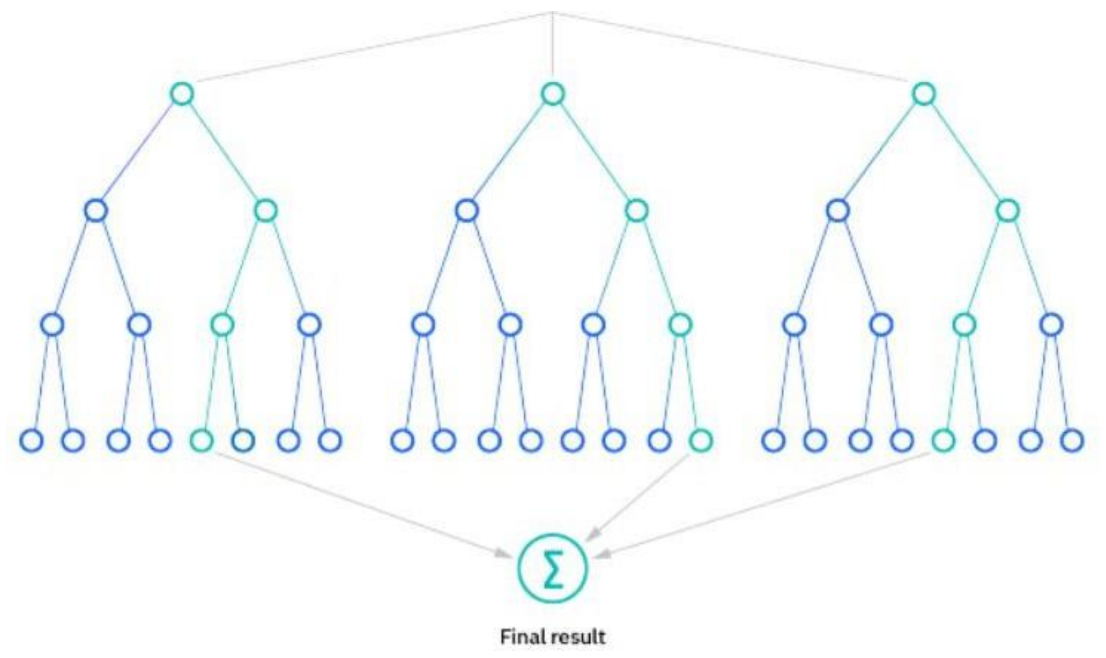
If we go back to the “should I surf?” example, the questions that I may ask to determine the prediction may not be as comprehensive as someone else’s set of questions. By accounting for all the potential variability in the data, we can reduce the risk of overfitting, bias, and overall variance, resulting in more precise predictions.
Featured products

SPSS Modeler

How it works

Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve for regression or classification problems.

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we’ll come back to later. Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary. For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.



Benefits and challenges of random forest

There are a number of key advantages and challenges that the random forest algorithm presents when used for classification or regression problems. Some of them include:

Key Benefits

- Reduced risk of overfitting: Decision trees run the risk of overfitting as they tend to tightly fit all the samples within training data. However, when there's a robust number of decision trees in a random forest, the classifier won't overfit the model since the averaging of uncorrelated trees lowers the overall variance and prediction error.
- Provides flexibility: Since random forest can handle both regression and classification tasks with a high degree of accuracy, it is a popular method among data scientists. Feature bagging also makes the random forest classifier an effective tool for estimating missing values as it maintains accuracy when a portion of the data is missing.
- Easy to determine feature importance: Random forest makes it easy to evaluate variable importance, or contribution, to the model. There are a few ways to evaluate feature importance. Gini importance and mean decrease in impurity (MDI) are usually used to measure how much the model's accuracy decreases when a given variable is excluded. However, permutation importance, also known as mean decrease accuracy (MDA), is another importance measure. MDA identifies the average decrease in accuracy by randomly permutating the feature values in oob samples.

Key Challenges

- Time-consuming process: Since random forest algorithms can handle large data sets, they can be provide more accurate predictions, but can be slow to process data as they are computing data for each individual decision tree.
- Requires more resources: Since random forests process larger data sets, they’ll require more resources to store that data.
- More complex: The prediction of a single decision tree is easier to interpret when compared to a forest of them.

Random forest applications

The random forest algorithm has been applied across a number of industries, allowing them to make better business decisions. Some use cases include:

- Finance: It is a preferred algorithm over others as it reduces time spent on data management and pre-processing tasks. It can be used to evaluate customers with high credit risk, to detect fraud, and option pricing problems.
- Healthcare: The random forest algorithm has applications within computational biology .allowing doctors to tackle problems such as gene expression classification, biomarker discovery, and sequence annotation. As a result, doctors can make estimates around drug responses to specific medications.
- E-commerce: It can be used for recommendation engines for cross-sell purposes.

Why we used random forest for the project is

Certainly! Let's dive into more detail for each of the mentioned machine learning algorithms:

1. **Linear Regression:**

   - *Description:* Linear regression models the relationship between the dependent variable and one or more independent variables by fitting a linear equation to observed data.

   - *Use Cases:* Predicting house prices, sales forecasting, and other scenarios where a linear relationship is assumed.

2. **Decision Trees:**

   - *Description:* Decision trees recursively split the data based on the most significant feature at each node, forming a tree-like structure. Each leaf node represents a decision or prediction.

   - *Use Cases:* Classification and regression tasks. Decision trees are intuitive and easy to visualize.

3. **Support Vector Machines (SVM):**

   - *Description:* SVMs find the hyperplane that best separates classes in a high-dimensional space. They can handle non-linear relationships using kernel functions.

   - *Use Cases:* Classification tasks, such as image recognition, and regression tasks.

4. **Neural Networks:**

   - *Description:* Neural networks consist of layers of interconnected nodes (neurons) that learn to map input data to output. Deep learning involves neural networks with multiple hidden layers.

   - *Use Cases:* Image and speech recognition, natural language processing, and complex pattern recognition.

5. **K-Nearest Neighbors (KNN):**

   - *Description:* KNN classifies data points based on the majority class of their k-nearest neighbors. It's a simple algorithm with no training phase.

   - *Use Cases:* Classification and regression tasks, especially when localized patterns are important.

6. **Gradient Boosting (e.g., XGBoost, LightGBM):**

   - *Description:* Gradient boosting builds a series of weak learners (typically decision trees) and combines them to create a strong predictive model. Each tree corrects the errors of the previous ones.

   - *Use Cases:* Regression and classification tasks where high accuracy is required.

7. **Ensemble Methods (Bagging, Boosting):**

   - *Description:* Ensemble methods combine multiple models to improve overall performance. Random Forests use bagging (bootstrap aggregating) to build multiple decision trees and average their predictions.

   - *Use Cases:* Any task where combining diverse models can lead to better generalization and robustness.

8. **Principal Component Regression (PCR):**

   - *Description:* PCR combines principal component analysis (PCA) and linear regression. It reduces dimensionality using PCA and then applies linear regression on the principal components.

   - *Use Cases:* Useful when dealing with multicollinearity in high-dimensional datasets.

Each algorithm has its strengths and weaknesses, making them suitable for different scenarios. The choice depends on factors such as the nature of the data, the problem complexity, interpretability requirements, and computational resources. Experimenting with various algorithms and understanding their characteristics is crucial in selecting the most appropriate model for a given task and random forest regressor is best for our project

About :

Certainly! Random Forest is an ensemble learning method that combines multiple decision trees to create a more robust and accurate predictive model. Here's a detailed explanation of how Random Forest works:

### 1. **Decision Trees:**

   - Random Forest is built upon the foundation of decision trees. A decision tree is a flowchart-like structure where each internal node represents a decision based on the value of a particular feature, and each leaf node represents the outcome or the class label.

### 2. **Bootstrap Aggregating (Bagging):**

- Random Forest employs a technique called Bootstrap Aggregating or Bagging. It involves creating multiple subsets of the training dataset by sampling with replacement (bootstrapping). Each subset is used to train a different decision tree.

### 3. **Random Feature Selection:**

- During the construction of each decision tree, a random subset of features is considered at each split. This introduces diversity among the trees, making the Random Forest less prone to overfitting. The number of features to consider at each split is typically the square root of the total number of features.

### 4. **Building Multiple Trees:**

- Random Forest builds a predefined number of decision trees, and each tree is constructed independently using a subset of the data and a subset of features. The randomness introduced in both data and feature selection helps in capturing different aspects of the underlying patterns in the data.

### 5. **Voting or Averaging:**

- For classification tasks, the final prediction is made by a majority vote among the trees. For regression tasks, the final prediction is the average of the predictions from individual trees.

### 6. **Advantages of Random Forest:**

- **High Accuracy:** Random Forest tends to provide high accuracy because it leverages the wisdom of multiple trees.

- **Robust to Overfitting:** The ensemble approach and random feature selection make Random Forest less prone to overfitting compared to individual decision trees.

   - **Handles Missing Values:** Random Forest can handle missing values in the dataset.

### 7. **Out-of-Bag (OOB) Error:**

   - Since each tree is trained on a different subset of data, there are data points that are not included in the training of a particular tree. These out-of-bag samples can be used to estimate the model's performance without the need for a separate validation set.

### 8. **Feature Importance:**

   - Random Forest provides a measure of feature importance based on how much each feature contributes to the overall accuracy of the model. This can be valuable for feature selection and understanding the importance of different variables in making predictions.

### 9. **Limitations:**

   - **Interpretability:** While powerful, the ensemble nature of Random Forests can make them less interpretable compared to individual decision trees.

   - **Computational Complexity:** Training multiple decision trees can be computationally expensive, especially for large datasets.

Random Forests are widely used for both classification and regression tasks across various domains. They are particularly effective when dealing with high-dimensional data, noisy datasets, and complex relationships between features and the target variable. The ability to handle diverse datasets and provide robust predictions makes Random Forest a popular choice in many machine learning applications.

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.

4.2 GRAPHICAL REPRESENTATION

Representing the predicted values and their differences from the actual values in graphical form has several benefits:

1. **Visual Comparison:** Graphical representations allow for a quick and intuitive comparison between the predicted values and the actual values. It provides a visual means to assess how well the model's predictions align with the ground truth. By plotting the data points or lines, any discrepancies or patterns can be easily identified and interpreted.
2. **Pattern Visualization:** Graphs can reveal patterns or trends in the data that may not be apparent from numerical values alone. By visualizing the data, it becomes easier to observe any systematic variations, cyclic patterns, or other relationships between the predicted and actual values. This can provide insights into the performance of the model and potential areas of improvement.
3. **Communicating Results:** Graphical representations are often more accessible and easier to interpret for stakeholders or non-technical audiences. By presenting the data in a visual form, the results of the model's predictions can be effectively communicated, facilitating better understanding and decision-making.
4. **Error Analysis:** Graphs can aid in error analysis by highlighting regions where the model performs well or poorly. For example, by plotting the

absolute differences between predicted and actual values, it becomes evident which areas have higher or lower prediction errors. This information can guide further investigation into the underlying factors contributing to the errors and inform model refinement strategies. Visualizing the predicted values and their differences in graphical form enhances the interpretability, understanding, and communication of the model's performance. It allows for quick insights, pattern recognition, and error analysis, ultimately facilitating better decision-making and guiding further improvements in the model. In this project the variation among the actual values and the predicted values is compared using a Line Graph. This graphical representation is obtained using “matplotlib.pyplot” package in python.

More info about Mean Squared Error (MSE)

Certainly! Mean Squared Error (MSE) is a common metric used in machine learning to evaluate the performance of a regression model. It quantifies the average squared difference between the predicted values and the actual values. Here's a more detailed explanation:

1. **Definition:**

   The Mean Squared Error is calculated as the average of the squared differences between predicted and actual values. For a dataset with \(n\) data points, where \(y_i\) is the actual value and \(\hat{y}_i\) is the predicted value for the \(i\)-th data point, the MSE is given by:

   \[MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \]

2. **Interpretation:**

   - A smaller MSE indicates that the predictions are closer to the actual values.

   - Squaring the differences penalizes larger errors more than smaller ones.

3. **Calculation Steps:**

   - Subtract the predicted value from the actual value for each data point.

   - Square each difference.

   - Calculate the average of these squared differences.

4. **Use in Model Evaluation:**

   - MSE is often used during the training phase to optimize models by adjusting parameters to minimize this error.

   - It's also used on a separate test dataset to assess how well the model generalizes to new, unseen data.

5. **Sensitivity to Outliers:**

   - MSE is sensitive to outliers, meaning that large errors for a few data points can significantly impact the overall score.

6. **Normalization:**

   - Sometimes, MSE is normalized by the range of the target variable or other factors to make it more interpretable and comparable across different datasets.

7. **Other Metrics:**

   - While MSE is widely used, it's not the only metric for regression. Depending on the context, other metrics like Mean Absolute Error (MAE) or \(R^2\) (coefficient of determination) may also be considered.

8. **Implementation in Python:**

   - In Python, you can easily calculate MSE using libraries like NumPy:

```python
import numpy as np
from sklearn.metrics import mean_squared_error

y_true = np.array([1, 2, 3, 4])
y_pred = np.array([2, 2.5, 3.5, 3.9])

mse = mean_squared_error(y_true, y_pred)
print("Mean Squared Error:", mse)
```

Remember, while MSE is valuable, it's essential to consider the specific characteristics of your problem and dataset when choosing an evaluation metric.

More info about Accuracy

Accuracy is a common metric used in machine learning to evaluate the performance of classification models. It represents the ratio of correctly predicted instances to the total instances in the dataset. Here's more information about accuracy:

1. **Definition:**

- Accuracy is calculated as the ratio of correctly predicted instances to the total instances in the dataset. It is expressed as a percentage.

- For a binary classification problem, where there are two classes (positive and negative), accuracy is given by:

{Accuracy} = [(Number of Correct Predictions)/(Total Number of Predictions) X 100%]

2. **Interpretation:**

- A higher accuracy indicates better model performance.

- It is a straightforward metric that is easy to understand and communicate.

3. **Limitations:**

- Accuracy might not be the best metric in all scenarios, especially when dealing with imbalanced datasets.

- In imbalanced datasets, where one class significantly outnumbers the other, a high accuracy might be achieved by simply predicting the majority class.

4. **Use in Model Evaluation:**

- Accuracy is often used during the training phase to optimize models by adjusting parameters to maximize accuracy.

- It's commonly used on a separate test dataset to assess how well the model generalizes to new, unseen data.

5. **Sensitivity to Imbalance:**

   - In cases of imbalanced datasets, where one class is rare compared to the other, accuracy alone might be misleading. For example, if 95% of instances belong to class A and 5% to class B, a model that predicts all instances as class A would still achieve 95% accuracy.

6. **Complementary Metrics:**

   - In addition to accuracy, other metrics such as precision, recall, and F1 score are often considered to provide a more comprehensive evaluation of a classifier's performance, especially in imbalanced scenarios.

7. **Implementation in Python:**

   - In Python, you can calculate accuracy using scikit-learn:

```python
from sklearn.metrics import accuracy_score
y_true = [1, 0, 1, 1, 0, 1]
y_pred = [1, 0, 1, 0, 1, 1]
acc = accuracy_score(y_true, y_pred)
print("Accuracy:", acc)
```

8. **Adjusting for Imbalance:**

   - If dealing with imbalanced datasets, techniques such as resampling, using different evaluation metrics, or employing specialized algorithms like cost-sensitive learning can help address the issue.

While accuracy is a valuable metric, it's crucial to consider the specific characteristics of your problem, especially if the dataset is imbalanced, and to explore other metrics for a more nuanced evaluation of your model's performance.

More info about Matplotlib

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in Python. `matplotlib.pyplot` is a collection of functions that make `matplotlib` work like MATLAB. It is often used for creating 2D plots and charts. Here's some information about `matplotlib.pyplot`:

1. **Installation:**
   - You can install Matplotlib using the following pip command:
     ```
     pip install matplotlib
     ```

2. **Basic Usage:**
   - To use `matplotlib.pyplot`, you typically import it with the alias `plt`:
     ```python
     import matplotlib.pyplot as plt
     ```

3. **Creating Simple Plots:**
   - You can create basic line plots, scatter plots, histograms, and more using the `plt.plot()`, `plt.scatter()`, `plt.hist()`, etc. functions.

4. **Plot Customization:**
   - Matplotlib allows extensive customization of plots. You can set axis labels, titles, legends, colors, line styles, markers, and more.
     ```python
     plt.plot(x, y, label='My Data', color='blue', linestyle='--', marker='o')
     plt.xlabel('X-axis Label')
     ```

   plt.ylabel('Y-axis Label')

   plt.title('My Plot')

   plt.legend()

5. **Subplots:**

   - You can create multiple plots in the same figure using `plt.subplots()`.

   ```python

   fig, ax = plt.subplots(nrows=2, ncols=2)

   ax[0, 0].plot(x1, y1)

   ax[0, 1].scatter(x2, y2)

   ```

6. **Saving Plots:**

   - You can save the generated plots in various formats (e.g., PNG, PDF, SVG) using `plt.savefig()`.

7. **Common Types of Plots:**

   - Line plots: `plt.plot()`

   - Scatter plots: `plt.scatter()`

   - Bar plots: `plt.bar()`

   - Histograms: `plt.hist()`

   - Box plots: `plt.boxplot()`

   - Pie charts: `plt.pie()`

8. **Additional Functionality:**

   - `matplotlib.pyplot` provides additional functionalities like color maps, annotations, 3D plotting, and more.

9. **Documentation and Resources:**

   - Matplotlib has extensive documentation and a gallery of examples on its official website: [Matplotlib Documentation](https://matplotlib.org/stable/contents.html)

10. **Interactive Mode:**

    - `matplotlib.pyplot` can be used interactively in Jupyter notebooks or other interactive environments, allowing you to dynamically modify plots.

11. **Integration with Pandas:**

    - `matplotlib.pyplot` can be easily integrated with Pandas for visualizing DataFrames.

Here's a simple example of a line plot using `matplotlib.pyplot`:

```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a line plot
plt.plot(x, y, label='Linear Function')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Simple Line Plot')
plt.legend()
plt.show()
```

This example demonstrates the basic usage of `matplotlib.pyplot` to create a line plot with labeled axes and a legend. You can customize the plot further based on your specific requirements.

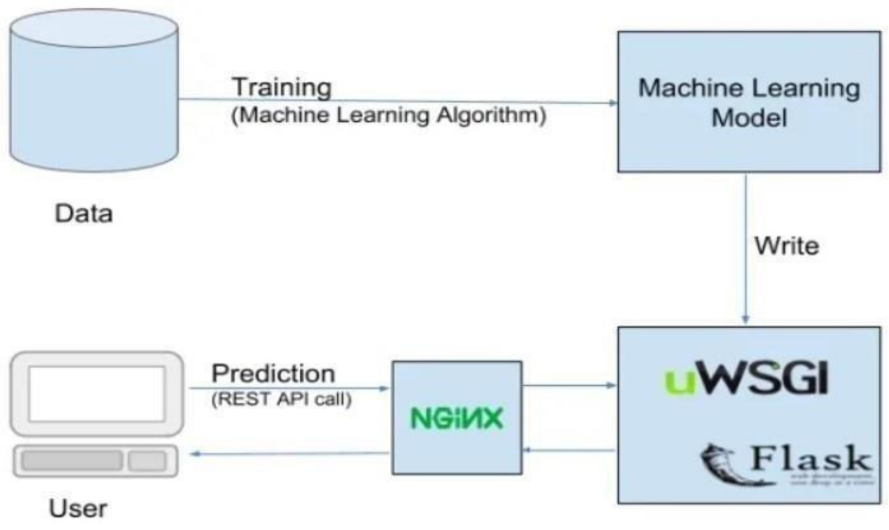
V.IMPLEMENTATION AND INTEGRATION

5.1 INTEGRATION INTO A FLASK APPLICATION

The machine learning model was developed using a Random Forest Regressor algorithm to predict the CO:CO2 ratios after 1, 2, 3, and 4 hours. Once the model was trained and evaluated, it was integrated into a Flask web application. The Flask code served as the backend of the system, handling HTTP requests and generating responses.

The trained machine learning model (Random Forest Regressor) was loaded within the Flask application. The Flask routes were set up to handle incoming requests, gather the required input data, and make predictions using the loaded model. The predictions were then used to generate the desired output, which included the predicted ratios, visualizations, and other relevant information.

By integrating the machine learning model with the Flask code, the system was able to provide real-time predictions based on user input. This allowed users to interact with the model through a user-friendly web interface, making it more accessible and practical for practical use.



5.2 WHAT IS FLASK?

Flask is a popular web framework written in Python that allows developers to build web applications quickly and efficiently. It follows a minimalist approach and provides the necessary tools and libraries to handle web development tasks.

1. **Microframework:** Flask is often referred to as a "microframework" because it focuses on simplicity and minimalism. It provides a basic set of features and functionality, allowing developers to choose and add additional components as needed.
2. **Routing and Request Handling:** Flask uses a routing mechanism to map URLs to specific functions or view handlers. Developers can define routes and specify the HTTP methods (GET, POST, etc.) that each route should respond to. Flask also provides request and response handling utilities to process incoming requests and generate appropriate responses.
3. **Template Engine:** Flask comes with a built-in template engine called Jinja2, which allows developers to separate the application logic from the presentation layer. Templates can be used to dynamically generate HTML pages by inserting data and applying conditional logic or loops. This makes it easier to create dynamic and interactive web pages.
4. **HTTP and RESTful APIs**: Flask supports various HTTP features and provides tools for building RESTful APIs. Developers can define routes with specific URL patterns, handle different request methods, and serialize data into various formats like JSON or XML for API responses.
5. **Extensions and Modularity:** Flask follows a modular design, allowing developers to add functionality through various extensions. These extensions provide additional features like database integration, authentication, form handling, and more. Flask's modular architecture makes it easy to customize and extend applications according to specific requirements.
6. **Development and Testing:** Flask includes a built-in development server, allowing developers to run and test their applications locally during development. It also provides a convenient debugger and error handling mechanism to help identify and fix issues. Flask encourages unit testing and provides tools for writing and executing tests to ensure application quality.
7. **Community and Ecosystem:** Flask has a large and active community of developers. This vibrant community has contributed a wide range of thirdparty extensions, tutorials, and documentation, making it easier to find resources and solutions to common web development challenges.

Flask is known for its simplicity, flexibility, and ease of use, making it a popular choice for building web applications and APIs using Python.

Flask is a lightweight and versatile web framework for Python that is widely used for developing web applications. It is designed to be simple and easy to use, making it an excellent choice for both beginners and experienced developers. Here are some key features and reasons why Flask is often considered a popular choice:

1. **Micro Framework:** Flask is often referred to as a "micro" framework because it provides the essentials needed for web development without imposing a rigid structure. This allows developers to choose the components they need and build upon them as per their requirements.

2. **Modularity:** Flask follows a modular design, allowing developers to choose and use only the components they need. This makes it highly customizable and flexible, enabling developers to integrate various extensions for specific functionalities.

3. **Ease of Learning:** Flask has a simple and straightforward syntax, making it easy for developers to pick up and start building applications quickly. Its minimalistic approach also means there is less boilerplate code, resulting in faster development.

4. **Werkzeug and Jinja2:** Flask is built on top of the Werkzeug WSGI toolkit and uses the Jinja2 template engine. Werkzeug provides a set of utilities for handling HTTP requests, while Jinja2 allows developers to create dynamic HTML templates.

5. **Extensibility:** Flask has a vibrant ecosystem of extensions that can be easily integrated to add various features to your application. Whether you need database integration, authentication, or other functionalities, there's likely a Flask extension available for it.

6. **Community and Documentation:** Flask has a large and active community, which means there are plenty of resources, tutorials, and

documentation available. This can be immensely helpful for developers who are just starting with Flask or facing challenges during development.

7. **RESTful Support:** Flask provides excellent support for building RESTful APIs. With the help of extensions like Flask-RESTful, developers can create APIs with minimal effort.

8. **Scalability:** While Flask is often chosen for small to medium-sized projects, it is also scalable and can be used for larger applications. Its flexibility allows developers to structure their code in a way that suits the project's size and complexity.

It's important to note that the choice of a backend framework depends on various factors, including the specific requirements of a project, the developer's familiarity with a particular framework, and the project's long-term goals. Flask's simplicity, flexibility, and ease of learning make it an attractive option for many developers, but other frameworks like Django or Express.js may be better suited for different use cases. Ultimately, the "best" backend tool depends on the specific needs of the project and the preferences of the development team.

IMPLEMENTATION AND SOURCE CODE :

FLASK :

App.py :

```
from flask import Flask, render_template, request,
redirect, url_for import pickle
import numpy as np
import matplotlib.pyplot as
plt import io
import base64
from sklearn.metrics import

mean_squared_error app = Flask(_

name_)

# Load the trained machine learning model
model = pickle.load(open("D:\ml\model.pkl", "rb"))



@app.route('/'
) def home():
   return render_template('index.html')



@app.route('/predict',
methods=['POST']) def predict():
   # Get the input values from the form
   date_time =
   request.form.get("DATE_TIME")
   cb_flow =
   float(request.form.get("CB_FLOW"))
   cb_press =
   float(request.form.get("CB_PRESS"))
   cb_temp =
   float(request.form.get("CB_TEMP"))
   steam_flow =
   float(request.form.get("STEAM_FLOW"))
   steam_temp =
   float(request.form.get("STEAM_TEMP"))
   steam_press =
   float(request.form.get("STEAM_PRESS"))
   o2_press =
   float(request.form.get("O2_PRESS"))
   o2_flow =
   float(request.form.get("O2_FLOW"))
   o2_per =
float(request.form.get("O2_PER"))
pci = float(request.form.get("PCI"))
atm_humid =
float(request.form.get("ATM_HUMID"))
hb_temp =
float(request.form.get("HB_TEMP"))
hb_press =
float(request.form.get("HB_PRESS"))
top_press =
float(request.form.get("TOP_PRESS"))
top_temp1 =
float(request.form.get("TOP_TEMP1"))
top_spray =
float(request.form.get("TOP_SPRAY"))
top_temp =
float(request.form.get("TOP_TEMP"))
top_press_1 =
float(request.form.get("TOP_PRESS_1")) h2
= float(request.form.get("H2"))
skin_temp_avg =
float(request.form.get("SKIN_TEMP_AVG")) co =
float(request.form.get("CO"))
co2 = float(request.form.get("CO2"))

# Create a numpy array of the input
values input_data = np.array([
   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, skin_temp_avg, co, co2
  ]).reshape(1, -1)

  # Make the prediction using the
  model prediction =
  model.predict(input_data)

  # Prepare the prediction results
  prediction_text = f"Prediction:
  {prediction}"

  # Calculate CO:CO2 ratios for
  different hours hours_range =
  range(1, 5)
  co_ratios = [co / (co2 * hours) for hours in hours_range]

  # Create a bar plot to visualize the
  ratios plt.bar(hours_range, co_ratios)
  plt.xlabel('Hours')
  plt.ylabel('CO:CO2 Ratio')
  plt.title('CO:CO2 Ratio After Different Hours')
  plt.xticks(hours_range)

  # Save the plot to a BytesIO
  object img_bytes = io.BytesIO()
  plt.savefig(img_bytes,
  format='jpg') img_bytes.seek(0)

  # Encode the image bytes as base64
  img_base64 = base64.b64encode(img_bytes.read()).decode('utf-8')

  # Calculate the CO:CO2 ratios for different hours
  ratio_labels = [f"After {hours} hour(s), CO:CO2 ratio is {ratio:.2f}" for
    hours, ratio in zip( hours_range, co_ratios)]

  # Calculate the mean squared error
  actual_values = [1.000367533, 1.004055304, 1.01027027,
            1.002263991]  # Actual CO:CO2 ratios
  as floats mse =
  mean_squared_error(actual_values, co_ratios)

  # Calculate the accuracy
  accuracy = 1 - mse

  return redirect(url_for('result', prediction_text=prediction_text,
img_base64=img_base64, ratio_labels=ratio_labels, mse=mse, accuracy=accuracy))
```

```
@app.route('/resul
t') def result():
    prediction_text =
    request.args.get('prediction_text')
    img_base64 =
    request.args.get('img_base64')
    ratio_labels =
    request.args.getlist('ratio_labels') mse =
    float(request.args.get('mse'))
    accuracy = float(request.args.get('accuracy'))

    # Render the result page with the prediction result, image, and ratios
    return render_template('result.html', prediction_text=prediction_text,
img_base64=img_base64, ratio_labels=ratio_labels, mse=mse, accuracy=accuracy)
```

```
if _name_== '_main_
  ':
  app.run(debug=True
  )
```

Explanation:

The provided code is a Flask application that serves as the backend for a web-based prediction system. Here's an overview of how the code works:

- **Importing Required Libraries:** The necessary libraries such as Flask, pickle, numpy, matplotlib, io, and base64 are imported.
- **Loading the Trained Model:** The pre-trained machine learning model is loaded into memory using the `pickle.load()` function. This model will be used to make predictions based on the input data.
- **Defining Routes:** The Flask application defines two routes using the `@app.route` decorator:
  - The `'/'` route corresponds to the home page of the web application. It renders the `index.html` template.
  - The `'/predict'` route is used to handle the form submission when the user submits input data. It extracts the input values from the form, creates a numpy array with the input data, and makes a prediction using the loaded model.
- **Making Predictions:** The `predict()` function handles the form submission and predicts the CO:CO2 ratios based on the input data. It prepares the prediction result and calculates the CO:CO2 ratios for different hours.
- **Plotting the Ratios:** The CO:CO2 ratios for different hours are plotted using a bar chart from the `matplotlib` library. The plot is saved as a PNG image in memory and then encoded as base64 to be displayed in the result page.
- **Calculating Metrics:** The mean squared error (MSE) is calculated by comparing the predicted CO:CO2 ratios with the actual values. The accracy is calculated as 1 minus the MSE.

- **Redirecting to the Result Page:** The `predict()` function redirects the user to the `'result'` route with the prediction result, image, ratio labels, MSE, and accuracy as arguments.

- **Rendering the Result Page:** The `result()` function receives the prediction result, image, ratio labels, MSE, and accuracy as arguments. It renders the `result.html` template, passing these values to be displayed in the result page.

- **Running the Application:** The Flask application is run by calling `app.run(debug=True)`. It starts the web server, allowing the application to be accessed and interacted with through a web browser.

This Flask application takes user input, passes it through the loaded machine learning model to make predictions, and displays the results along with additional visualizations and metrics on a web page.

<u>Index.html :</u>

```
<!DOCTYPE html>
<html>


<head>
  <title>SP Projects</title>
  <style>
    body {
      background-image: url("https://dizz.com/wp-
content/uploads/2021/10/modern-blast- furnace-under-repair-renovation-
38945923-transformed.webp");
      background-position: center
      center; background-repeat:
      no-repeat; background-size:
      100% 100%;
      font-family: 'Times New Roman',
      sans-serif; color: #333;
    }
```

```
.container {
   max-width:

   800px; margin: 0

   auto;
```

```
  padding: 20px;
}

.header {
  text-align: center;
  margin-bottom:
  40px;
}

h1 {
  font-size: 36px;
  color: black;
  font-family: 'Times New Roman',
  "cursive"; font-weight: bolder;
  margin: 0;
}

h2 {
  font-family: 'Times New Roman', "cursive";
}

.form-container {
  background-color:
  #fff; border-
  radius: 5px;
  box-shadow: 0 0 10px rgba(0, 0, 0,
  0.1); padding: 30px;
  padding-right:
```

```
  50px; padding-
  bottom: 10px;
}

ul {
  font-family: 'Times New Roman',
  "cursive"; color: darkgreen;
```

```
}

.form-container h2
   { font-size: 24px;
   color: darkred;
   margin-top: 0;
   margin-bottom:
   20px; text-align:
   center;
}

.form-container input[type="text"],
.form-container
   input[type="number"] { display:
   block;
   width: 100%;
   margin-bottom: 10px;
   padding: 10px;
   border: 1px solid
   #ccc; border-
   radius: 5px;
   font-family: 'Times New Roman',
   "cursive"; color: black;
}

.form-container
   input[type="submit"] {
   display: block;
```

```
width: 200px;

margin: 20px

auto; padding:

10px; border-

radius: 20px;

border: none;

background-color:

black; color: #fff;

font-size: 16px;
```

```
        cursor: pointer;
        font-family: 'Times New Roman', "cursive";
    }

    .prediction {
        text-align:
        center;
        margin-top:
        30px;       font-
        size: 20px;
        font-family:  'Times  New  Roman',
        "cursive"; color: darkblue;
    }

    .ratio-list {
        margin-top:
        30px;
        padding-left:
        0;
        list-style-type: none;
    }

    .ratio-list li {
        margin-bottom:
        5px; color:
        darkgreen;
    }
</style>
```

```
</head>


<body>
  <div class="container">
    <div class="header">
      <h1>CO_CO2 RATIO PREDICTION USING MACHINE
      LEARNING</h1>
    </div>
    <div class="form-container">
      <h2>Input Values</h2>
```

```
          <form action="{{ url_for('predict') }}" method="post">
            <input type="text" name="DATE_TIME"
            placeholder="DATE_TIME">
            <input type="number" step="any" name="CB_FLOW"
            placeholder="CB_FLOW">
            <input type="number" step="any" name="CB_PRESS"
            placeholder="CB_PRESS">
            <!-- Add other input fields as needed -->
            <input type="submit" value="Predict">
          </form>
          <br>
        </div>
    </div>
</body>
  </html>
```

Explanation:

The provided code is an HTML template file for a web-based user interface using Flask and Jinja2 templating. It represents a form for inputting values and displaying the prediction results for the CO:CO2 ratio prediction project.

The template file begins with the necessary HTML structure and includes a `<head>` section for defining the title and CSS styles. The body of the template is divided into a container `<div>` that holds the form and prediction results.

Within the form-container `<div>`, the user can input various values related to the project. Each input field corresponds to a specific feature required for making predictions. The input fields are named accordingly and have placeholders to guide the user.

Upon submitting the form, the action attribute directs the request to the `/predict` route in the Flask application. The method is set to `POST`, ensuring that the form data is sent securely.

The prediction result is displayed in the `<div>` with the class "prediction". The prediction text is dynamically rendered using the `prediction_text` variable, which is passed from the Flask route.

If there are CO:CO2 ratio results available, they are displayed in an unordered list `<ul>` with the class "ratio-list". Each ratio value is iterated through and rendered as a list item `<li>` using the `ratios` variable passed from the Flask route.

Overall, the HTML template file provides a user-friendly interface for entering input values, displaying prediction results, and presenting CO:CO2 ratio values if available.

Result.html :

```
<!DOCTYPE html>
<html>

<head>
  <title>Result</title>
  <style>
    body {
      background: rgb(0, 128, 128);
      background: linear-gradient(0deg, rgba(0, 128, 128, 1) 9%, rgba(17, 2, 24, 1) 93%);
    }

    .container {
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
      font-family: "Comic Sans MS", "cursive";
    }

    h1 {
      color: white;
      text-align: center;
```

```
        font-size: 36px;
        margin-top: 0;
        margin-bottom: 30px;
    }

    h2 {
        color: #73ecfa;
    }

    li {
        font-weight: bold;
    }

    p {
        color: #e7ecf4;
        font-size: 18px;
        text-align: center;
        margin-top: 0;
    }

    ul {
        list-style-type: none;
        padding-left: 0;
        margin-top: 30px;
    }

    li {
        color: rgb(254, 250, 41);
        font-size: 16px;
        margin-bottom: 10px;
    }
</style>
```

```
</head>

<body>
   <div class="container">
       <div class="header">
          <h1>CO_CO2 RATIO PREDICTION RESULTS</h1>
       </div>
       <div class="prediction">
          <h2>{{ prediction_text }}</h2>
       </div>
       <div class="ratio-chart">
          <h2>CO:CO2 Ratios</h2>
          <img src="data:image/png;base64,{{ img_base64 }}" alt="Ratio Chart">
       </div>
       <div class="ratio-list">
          <ul>
             {% for ratio in ratio_labels %}
             <li>{{ ratio }}</li>
             {% endfor %}
          </ul>
       </div>
   </div>
</body>
</html>
```

Explanation:

The template file starts with the necessary HTML structure and includes a `<head>` section for defining the title and CSS styles. The body of the template is divided into a container `<div>` that holds the result content.

Within the container `<div>`, there is a header `<h1>` that displays the title of the result page.

The prediction result is displayed in a `<div>` with the class "prediction". The result text is dynamically rendered using the `prediction_text` variable, which is passed from the Flask route.

Below the prediction result, there is a `<div>` with the class "ratio-chart" that contains a title `<h2>` for the CO:CO2 ratios. The chart itself is displayed as an `<img>` tag, where the image source is set to a base64-encoded representation of the chart image. The `img_base64` variable is passed from the Flask route.

Further down, there is a `<div>` with the class "ratio-list" that displays the CO:CO2 ratios as a list. The ratios are iterated through using a `for` loop, and each ratio is displayed as a list item `<li>`.

Overall, the HTML template file provides a visually appealing result page that showcases the prediction result, a chart visualization of the CO:CO2 ratios, and a list of individual ratio values.

Certainly! HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies used in web development to create and design web pages. Let's delve into each of them:

### HTML (Hypertext Markup Language):

1. **Overview:**

   - HTML is the standard markup language for creating web pages.

   - It structures content on the web, defining elements such as headings, paragraphs, links, images, tables, and more.

2. **Basic Structure:**

   - HTML documents have a basic structure:

    ```html
    <!DOCTYPE html>
    <html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <!-- Content goes here -->
    </body>
    </html>
    ```

3. **Elements and Tags:**

   - Elements are the building blocks of HTML, defined by tags (e.g., `<p>` for paragraphs, `<h1>` for headings).

   - Tags often come in pairs (opening and closing), encapsulating content.

4. **Attributes:**

   - HTML elements can have attributes that provide additional information.

   - Example: `<img src="image.jpg" alt="Description">`

5. **Links and Navigation:**

   - `<a>` tag is used for hyperlinks, connecting pages.

   - `<nav>` tag is used for navigation menus.

6. **Lists:**

   - `<ul>` for unordered lists (bullets), `<ol>` for ordered lists (numbers), `<li>` for list items.

7. **Forms:**

   - `<form>` tag is used for creating input forms.

   - Input types include text, password, checkbox, radio, etc.

8. **Semantic HTML:**

   - Semantic elements like `<header>`, `<footer>`, `<article>`, and `<section>` provide meaning to the structure.

### CSS (Cascading Style Sheets):

1. **Overview:**

   - CSS is used for styling HTML and controlling the layout of web pages.

   - It separates content from presentation, enhancing the visual appeal of websites.

2. **Selectors:**

   - Selectors target HTML elements to apply styles.

   - Example: `h1 { color: blue; }` selects all `<h1>` elements and sets their color to blue.

3. **Properties and Values:**

   - CSS rules consist of properties and values.

   - Example: `font-size: 16px;`, where `font-size` is the property and `16px` is the value.

4. **Box Model:**

   - Every HTML element is a box with content, padding, border, and margin.

   - `margin`, `padding`, and `border` properties control spacing and borders.

5. **Layout:**

   - CSS provides layout options like `float`, `position`, and `display` for creating responsive designs.

   - Flexbox and Grid layout systems offer advanced control over layout.

6. **Colors and Backgrounds:**

   - Colors can be specified using names, HEX codes (#RRGGBB), RGB values, or HSL values.

   - Background properties control background color and images.

7. **Transitions and Animations:**

   - CSS allows for smooth transitions between states and keyframe animations.

   - Example: `transition: color 0.3s ease;` transitions color changes over 0.3 seconds with ease-in-out timing.

8. **Media Queries:**

   - Used for responsive design, adjusting styles based on the device's characteristics.

   - Example: `@media screen and (max-width: 600px) { /* styles for small screens */ }`

9. **Vendor Prefixes:**

   - Some CSS features may require vendor prefixes for cross-browser compatibility.

   - Example: `-webkit-border-radius: 5px;`

These are the foundational concepts of HTML and CSS. As web development evolves, new features and best practices emerge, but understanding these basics provides a solid foundation for creating web pages and applications.

How HTML and CSS works in Flask

In Flask, a web framework for Python, HTML and CSS are used to build the frontend (user interface) of web applications. Flask allows you to integrate HTML templates and serve static CSS files alongside your Python code. Here's how you can use HTML and CSS in a Flask application:

### 1. **HTML Templates:**

Flask uses the Jinja templating engine to render dynamic content within HTML templates. These templates allow you to embed Python code within HTML files. Here's a basic example:

```python
# app.py

from flask import Flask, render_template
```

```
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html', name='John')
```

```html
<!-- templates/index.html -->

<!DOCTYPE html>
<html>
<head>
    <title>Flask App</title>
    <!-- You can link to external CSS files here -->
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Hello, {{ name }}!</h1>
</body>
</html>
```

In this example, the `render_template` function is used to render the `index.html` template. The `{{ name }}` syntax is Jinja templating, allowing you to inject dynamic content.

### 2. **Static Files (CSS):**

You can store your CSS files in a `static` folder within your Flask project. For example:

```
project/
|-- app.py
|-- templates/
|   |-- index.html
|-- static/
    |-- style.css
```

Link your CSS file in your HTML templates as shown in the example above. The `url_for('static', filename='style.css')` function generates the correct URL for the static file.

### 3. **Dynamic Content:**

Flask allows you to pass dynamic data from your Python code to your HTML templates. In the example above, the `name` variable is passed to the template and displayed dynamically.

### 4. **Form Handling:**

HTML forms are used to collect user input. Flask can handle form submissions, validate data, and perform actions based on user input. You can use Flask-WTF or other form-handling libraries for more advanced form handling.

### 5. **Routing:**

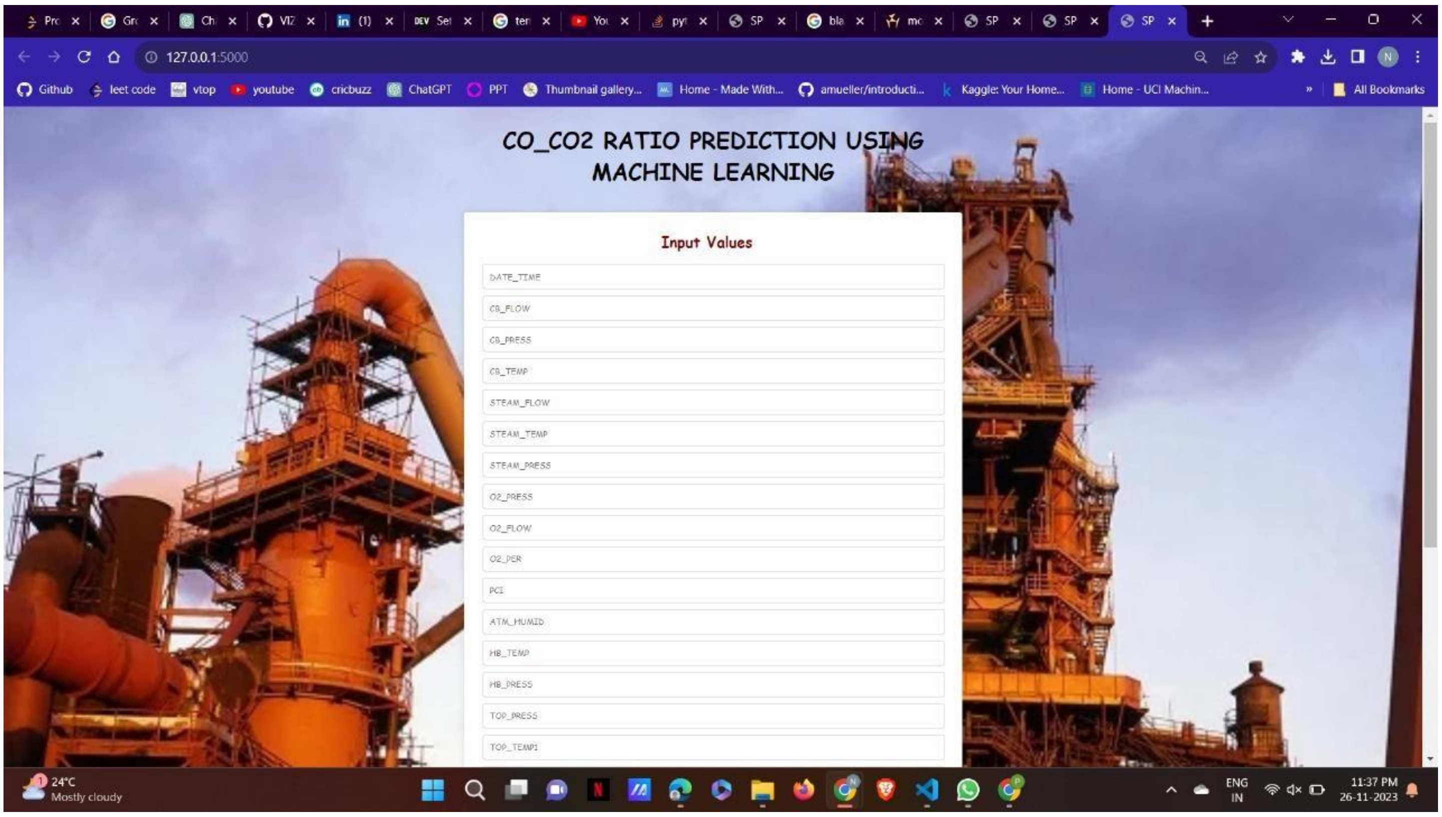
Flask uses routes to map URLs to functions in your Python code. Decorators like `@app.route('/')` define what function to call when a specific URL is accessed.

### 6. **Integration with Frontend Frameworks:**

You can use Flask as a backend API with frontend frameworks (e.g., React, Vue, Angular) to create single-page applications. Flask serves as the backend, handling data and logic, while the frontend framework handles the user interface.

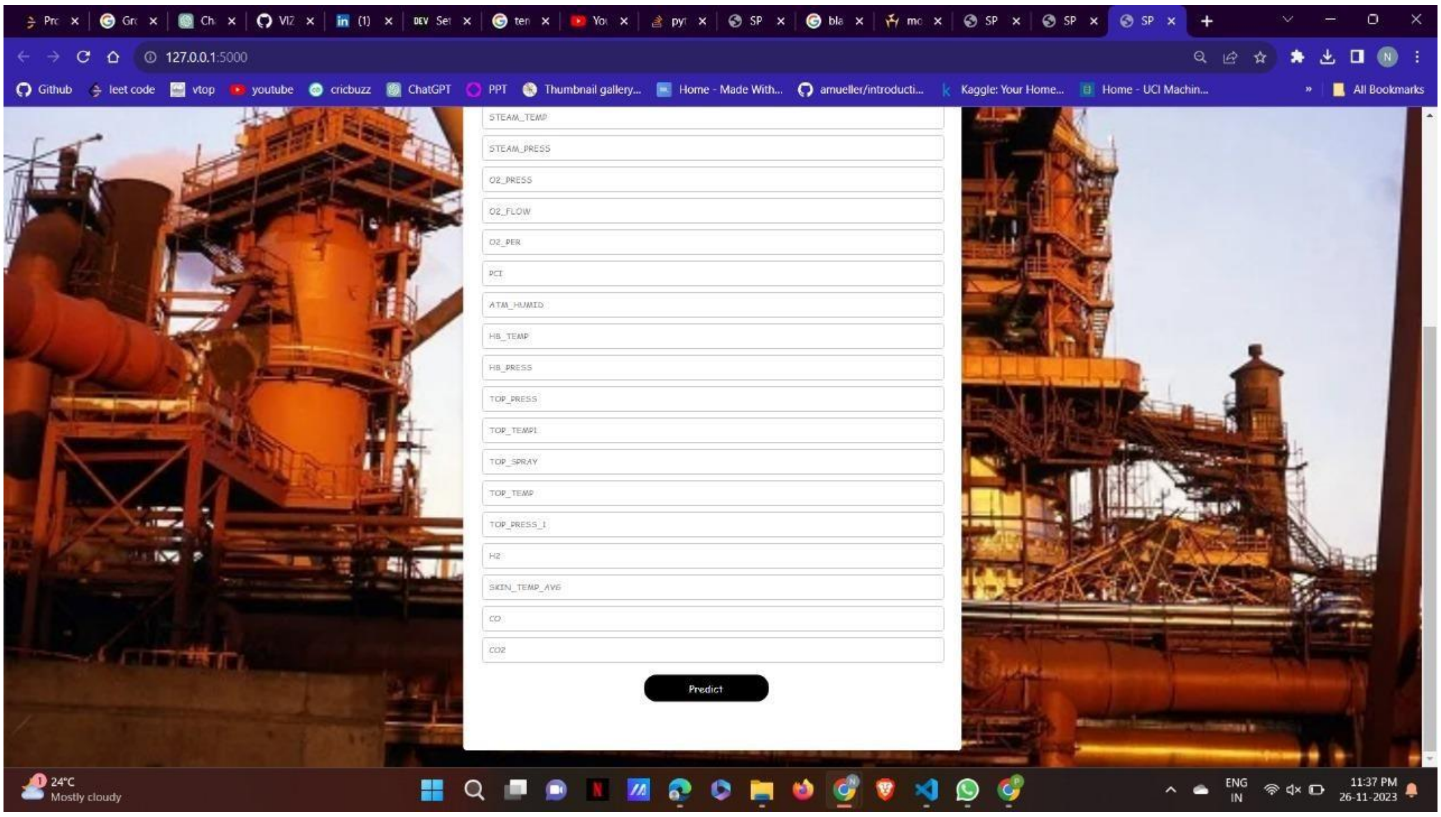
By combining these elements, you can create dynamic web applications using Flask with HTML and CSS for the frontend presentation.

RESULTS


CO_CO2 RATIO PREDICTION USING MACHINE LEARNING

Input Values

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

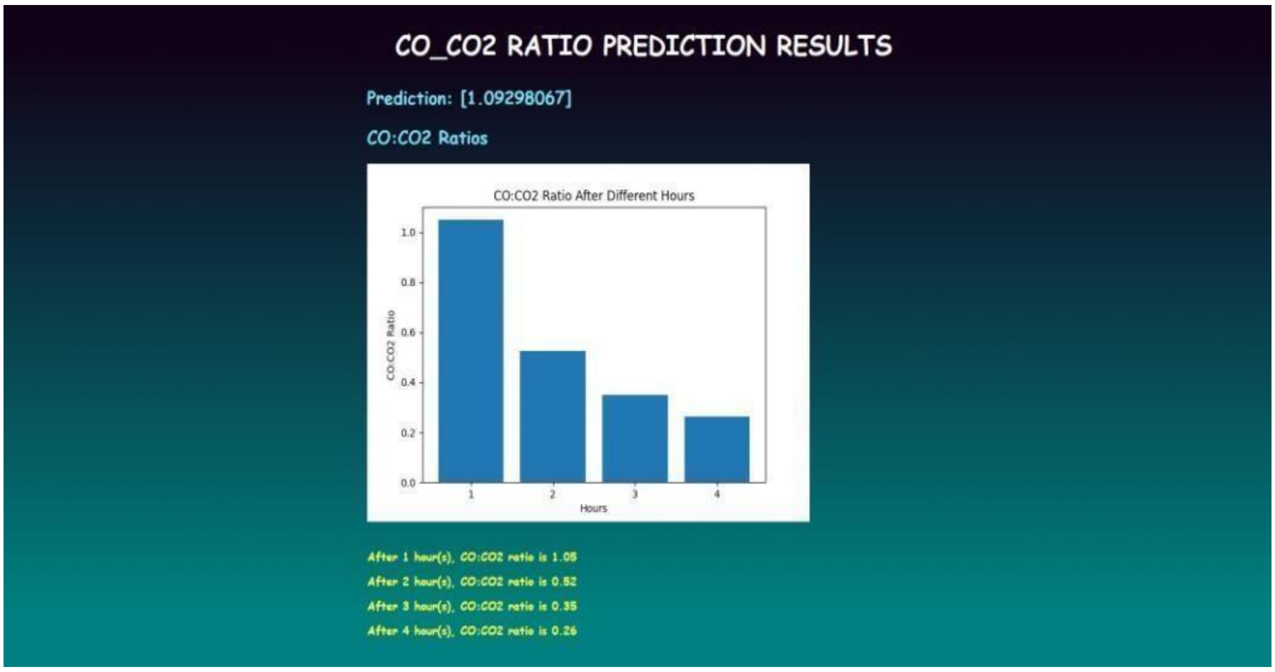

STEAM_TEMP
STEAM_PRESS
O2_PRESS
O2_FLOW
O2_PER
PCI
ATM_HUMID
HB_TEMP
HB_PRESS
TOP_PRESS
TOP_TEMP1
TOP_SPRAY
TOP_TEMP
TOP_PRESS_1
H2
SKIN_TEMP_AVG
CO
CO2

This represents the web page which takes the Blast furnace parameters as the input and using those input values along with the trained Machine Learning model it calculates the predicted co and co2 ratio values after certain time intervals.

The prediction values are clearly shown in the result page using a graphical representation. Since the ratios are decreases as the time interval increases, the accuracy is good for the model.

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.


CO_CO2 RATIO PREDICTION RESULTS

Prediction: [1.09298067]

CO:CO2 Ratios

After 1 hour(s), CO:CO2 ratio is 1.05
After 2 hour(s), CO:CO2 ratio is 0.52
After 3 hour(s), CO:CO2 ratio is 0.35
After 4 hour(s), CO:CO2 ratio is 0.26

The carbon monoxide (CO) to carbon dioxide (CO_2) ratio in a blast furnace is a critical parameter that influences the efficiency of the iron-making process. The blast furnace is a large, vertical furnace used in the production of iron from iron ore. The reduction of iron ore to molten iron occurs in the presence of carbon monoxide, which is produced by the combustion of coke (a carbon-rich material derived from coal) with hot air in the lower part of the furnace.

The overall chemical reaction in the blast furnace can be simplified as follows:

$Fe_2O_3 + 3CO \rightarrow 2Fe + 3CO_2$

In this equation, iron ore (Fe_2O_3) is reduced to iron (Fe) by carbon monoxide (CO), resulting in the production of carbon dioxide (CO_2). The ratio of CO to CO_2 is crucial for several reasons:

1. **Reduction of Iron Ore:** The primary role of carbon monoxide is to reduce iron ore to metallic iron. This reaction is only effective if there is enough CO available. A high CO/CO_2 ratio is desired to maximize the reduction of iron ore.

2. **Temperature Control:** The ratio of CO to CO_2 also influences the temperature within the blast furnace. A higher CO/CO_2 ratio generally leads to higher temperatures, which is beneficial for the reduction reactions.

3. **Energy Efficiency:** The blast furnace process requires a significant amount of energy. A balanced CO to CO_2 ratio contributes

to the overall energy efficiency of the process. It helps in utilizing the heat generated by the combustion reactions effectively.

4. **Carbon Consumption:** The carbon in the form of coke is a critical reductant in the blast furnace. A well-controlled CO to CO2 ratio ensures optimal carbon consumption, avoiding excessive coke usage.

5. **Hot Metal Quality:** The quality of the hot metal produced in the blast furnace, in terms of its carbon content and impurities, is influenced by the CO to CO2 ratio. Maintaining the right ratio helps in achieving the desired hot metal composition.

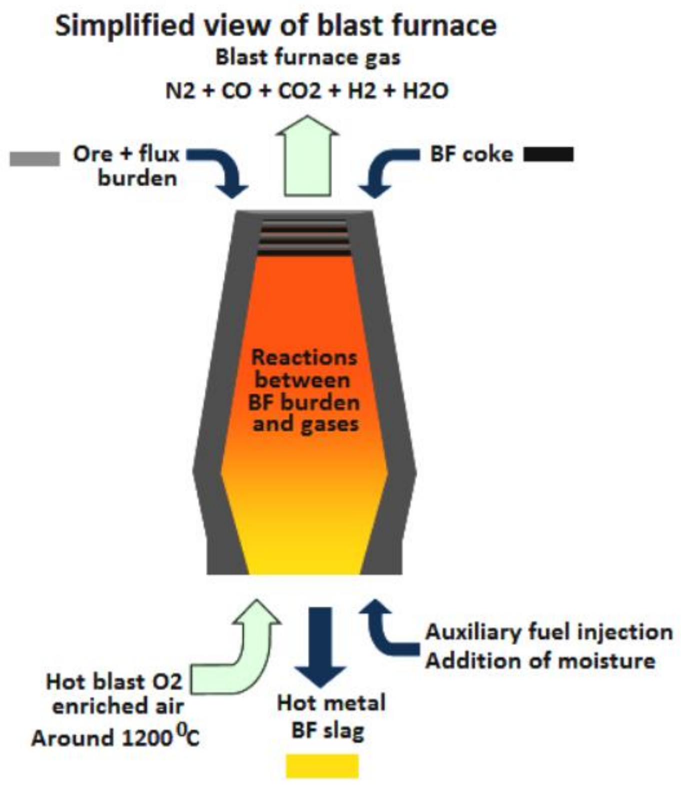
Achieving the desired CO to CO2 ratio involves careful control of various operating parameters within the blast furnace, including the composition of the charge materials, the blast temperature, and the distribution of air and fuel. Advanced process control technologies are often employed to optimize these parameters and maintain efficient operation.

It's worth noting that while a high CO/CO2 ratio is generally desired, there are practical limits and trade-offs that need to be considered to ensure stable and safe furnace operation. The goal is to balance the need for efficient iron reduction with the overall requirements of the iron-making process.

Blast furnace (BF) process is the leading technology for the production of hot metal (HM) required for steelmaking as well as for the production of the pig iron. HM is the main product of the BF. During the production of the HM, BF gas is produced simultaneously. BF gas is the name given to the by-product which is continuously produced from the upward gaseous rise of blast air through the burden in the BF during its operation.

Though the purpose of partial combustion of carbon in the BF is to remove the oxygen (O2) from the ore burden but the volume of gas generated in the BF makes the BF also a gas producer. BF gas is an important source of chemical energy consumed outside the BF process and has a major impact on the gas balance of an integrated steel plant. First of all, the surplus of BF gas is consumed in different furnaces of the steel plant and also in the power plant boiler together with other by-product gases such as coke oven gas and converter gas. The main parameter which has a decisive say about the usefulness of BF gas is its calorific value.

An illustrative simple view of blast furnace operation, showing the BF gas coming out from the furnace top is shown in Fig 1.



During the prc nace, hot air
blast is blown ontained
within the hot f coke) to
produce carbon di-0xide (CO2) and carbon mono oxide (CO), as per the equations (i) C + O2 = CO2, and (ii) CO2 + C = 2CO. The gas

produced by this reaction moves up the furnace shaft which has been charged with ores, fluxes and coke. After a number of chemical reactions as described below and a travel of around 25 m to 30 m the BF gas comes out of the furnace as a heated, dust laden and lean calorific value (CV) combustible gas.

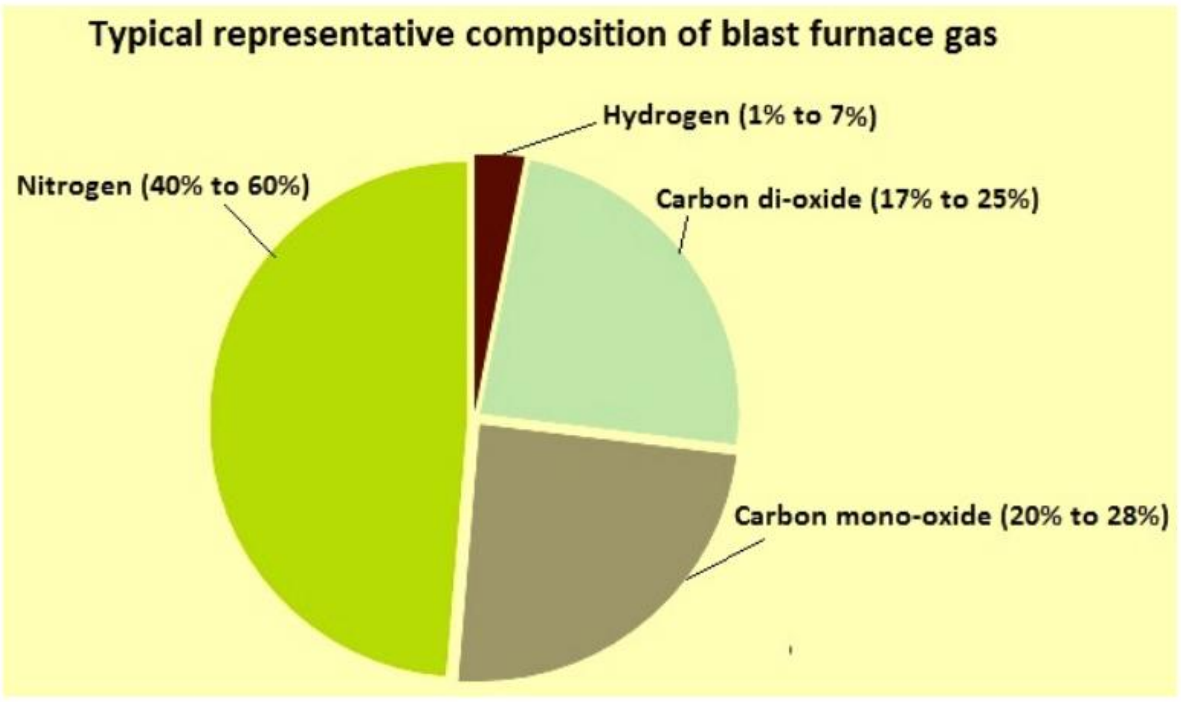
Both carbon (C) in the coke and CO are reducing agents for the ore burden consisting of hematite (Fe_2O_3), wustite (FeO), and magnetite (Fe_3O_4). These oxides are reduced to form Fe and CO_2. For example, the reduction mechanisms of hematite are given by the equations (i) $Fe_2O_3 + 2C = 2Fe + CO + CO_2$, and (ii) $Fe_2O_3 + 3CO = 2Fe + 3CO_2$.

A further source of gaseous release results from the decomposition of limestone and dolomite used as basic fluxes for the removal of the impurities. These reactions are (i) $CaCO_3 = CaO + CO_2$, and (ii) $MgCO_3 = MgO + CO_2$.

All of these changes are happening in the reaction zone of the furnace, and importantly from the perspective of BF gas composition, chemical equilibrium for the gases released is governed by the Boudouard reversible reaction ($2CO = CO_2 + C$) as a set ratio is reached between CO and CO_2 for a given temperature. The operational result is for large quantities of hot CO_2, CO, and N_2 to ascend through the furnace as fresh burden travels down into the reaction zone.

However, there can be further constituents added to the gaseous composition depending on systematic variables. As an example, additional reductants can be injected in the BF in order to reduce coke requirement in the burden, such as pulverized coal, oil, natural gas, or recycled plastics, and thereby improve furnace efficiency. However, burden integrity is to be maintained, necessitating the injection of steam or O2 alongside any additional reductants. These additions lead to fluctuating levels of H2 and H2O in the hot air blast, and subsequently affect the water-gas shift reversible reactions namely (i) $C + H_2O = CO + H_2$, and (ii) $CO + H_2O = CO_2 + H_2$.

The overall chemical composition of BF gas is hence dynamic and is dependent on the furnace operating parameters. A dry volumetric composition of the BF gas representative of typical operation is given in the Fig 2.



The specific volume of the BF gas (cum/ton of HM) generated, its chemical composition, and its CV is dependent on the operating parameters of BF, such as (i) characteristics of the burden materials, (ii) amount of fluxes charged in the BF, (iii) distribution of burden materials in the BF stack, (iv) grade of hot metal being made, (v) quantity of auxiliary fuel injected into the BF, (vi) the temperature of hot blast, and (vii) the O2 content in the blast. Hence, the operating parameters are of practical importance from the point of view of energy management of the integrated steel plant. The quantity of BF gas transferred to other consumers depends on the amount of gas produced in the BF and on the amount of BF gas consumed in the hot blast stoves of the BF.

The total quantity of CO + CO2 gases by volume in the BF gas at the furnace top ranges from around 37 % to 53 % of the total gas volume. The CO/CO2 ratio can vary in a blast furnace from 1.25:1 to 2.5:1. Higher percentage of CO in the gas makes the BF gas hazardous. The hydrogen (H2) content of the BF gas can vary from 1 % to 7 % depending upon the type and amount of fuel injected in the tuyeres of the BF. The balance component of the BF gas is nitrogen (N2). Methane (CH4) can also be present in the BF gas upto a level of 0.2 %.

In the BF, some hydro-cyanide (HCN) and cyanogen gas (CN2) can also form due to the reaction of N2 in the hot air blast and C of the coke. This reaction is catalyzed by the alkali oxides. These gases are highly poisonous. BF gas can contain these cyano compounds in the range of 200 milligrams per cubic metre (mg/cum) to 2000 mg/cum.

BF gas leaves the BF top at a temperature of around 120 deg C to 370 deg C and a pressure which can range from around 350 mm to 2,500 mm mercury gauge pressure. It carries at this stage around 20 grams per cubic metre (g/cum) to 115 g/cum of water vapour and 20 g/cum to 40 g/cum of dust commonly known as ‘flue dust’. The particle size of the flue dust can vary from a few microns to 6 mm.

BF gas is almost colourless (mild whitish), and an odourless gas. Other main characteristics of the BF are (i) very low CV usually in the range of around 700 kilo calories per cubic meter (kcal/cum) to 850 kcal/cum, (ii) relatively a high density usually in the range of around 1.250 kilograms per cubic meter (kg/cum) at is 0 deg C and 1 atmosphere pressure which the standard temperature and pressure (STP), (iii) low theoretical flame temperature which is around 1455 deg C, (iv) low rate of flame propagation which is usually lower than any other common gaseous fuel, (v) burns with a non luminous flame, (vi) auto ignition point of around 630 deg C, and (vii) has a lower explosive limit (LEL) of 27 % and an upper explosive limit (UEL) of 75 % in an air-gas mixture at normal temperature and pressure. The density of the BF gas is highest amongst all the gaseous fuel. Since the density is higher than the density of air, it settles in the bottom in case of a leakage. High concentration of CO gas in the BF gas makes the gas hazardous.

The high top pressure of BF gas is utilized to operate a generator (top gas pressure recovery turbine, i.e. TRT in short). TRT can generate electrical energy (power) up to 35 kWh/ ton of hot metal without burning any fuel. Dry type of TRT can produce more power then wet type.

cleaning of bf gas

BF gas coming out of the furnace top contains 20 g/cum to 40 g/cum of flue dust and cannot be used as such. This dust contains fine particles of coke, burden materials and chemical compounds which are formed due to the reactions taking place inside the BF. This dirty BF gas is cleaned in gas cleaning plant in two stages namely (i) primary gas cleaning stage, and (ii) secondary gas cleaning stage.

Primary gas cleaning consists of dust catchers, cyclones, or a combination of both. The gravity principle is used for the removal of large particles (coarser than 0.8 mm) of the dust. In this stage, the BF gas is normally passed through a dust catcher where all the coarser particles are removed. The dust catcher is a large cylindrical structure normally 20 m to 30 m in diameter and with a height of 20 m to 30 m. It is usually lined to insulate it and prevent the condensation of moisture present in the BF gas so that the dust remains dry and does not ball up and flow freely into the conical portion of the dust catcher at its bottom for its periodical removal.

The BF gas is sent to the dust catchers by a single down comer and enters through the top of the dust catcher by a vertical pipe which carries the gas downward inside the dust catcher. This pipe flares at its lower extremity like an inverted funnel, so that as the gas passes downward its velocity (and thus its dust carrying potential) decreases, and most of the coarser dust (coarser than 0.8 mm) drops out of the gas stream and is deposited in the cone at the bottom of the dust catcher. Because the bottom of the dust catcher is closed, and the gas outlet is near the top, the direction of the travel of the gas gets reversed 180 degrees. This sudden reversal in the direction of flow causes more of the dust to get settle down.

After dust catcher the gas is sent to secondary gas cleaning stage. Here BF gas is cleaned either by dry type gas cleaning system or wet type gas cleaning plants. In dry type gas cleaning plants bag filters are used for removal of fines particles of dust while in the wet type gas cleaning plant BF gas is washed of dust in scrubbers in several stages.

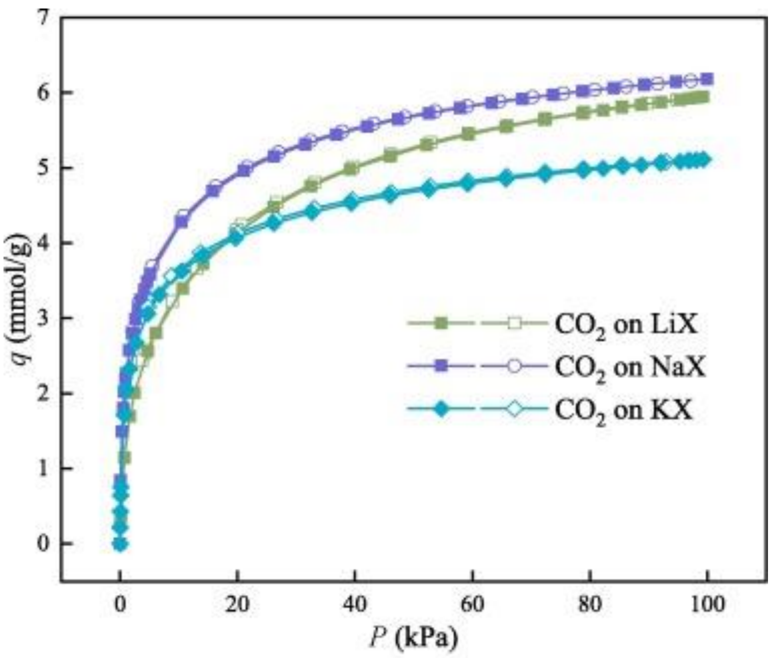
Uses of BF gas

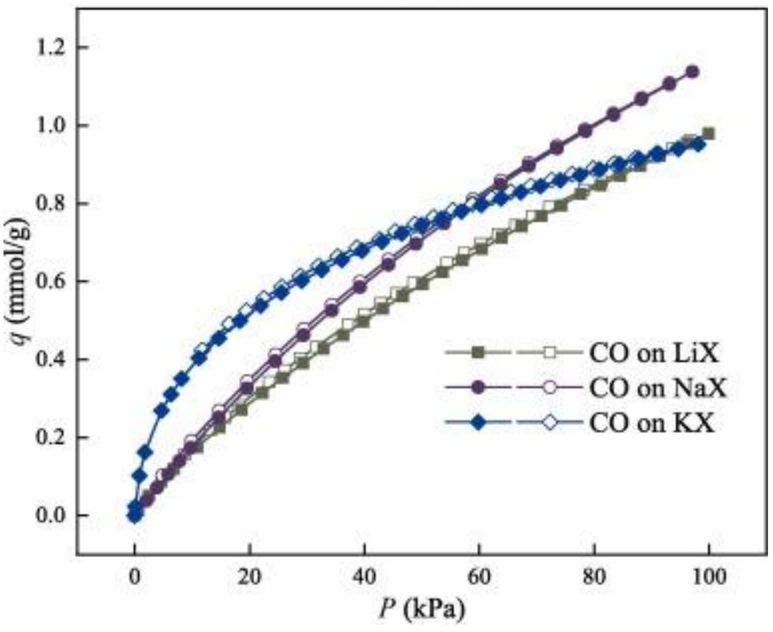
The sensible heat in the BF top gases was first utilized in 1832 to transfer heat to the cold blast. Originally, this heat exchanger was mounted on the top of the furnace. In 1845, the first attempts were made to make use of heat of combustion of BF gas, but the burning of BF gas was not successful till 1857. It is probable that the progress in the utilization of BF gas was delayed due to its high dust content, the problems of cleaning and handling, and the low cost of solid fuel. Increasing cost of other fuels and competition forced its use.

In the past BF gas use was restricted to the heating of hot blast stoves in the blast furnaces and using it in multi fuel boilers. It was not considered to be economical for other uses because of its various characteristics. However in the recent years, several factors have contributed to its enlarged use. The factors which have contributed to the enlarged use of gas are (i) increase in the cost of the purchased fuels, (ii) technical improvement in gas cleaning thus improving the cleanliness of the gas, and (iii) technology development for BF gas preheating.

In integrated steel plants, BF gas is normally being used mixed with either coke oven gas or converter gas or both. The mixed gas is used as a fuel in various furnaces of the integrated steel plant. BF gas without mixing and without preheat can be used in BF stoves, soaking pits, normalizing and annealing furnaces, foundry core ovens, gas engines for blowing, boilers for power generation, gas turbines for power generation. With recent advances in the technology, BF gas is also being directly used in the sinter plant furnace.

The thermal advantage of using BF gas in gas engines for blowing and for power generation has to overcome the heavy investment and maintenance expense required for such equipment. The modern boiler house utilizes high steam pressure and temperature with efficient turbo-blowers and generators. This has sufficiently reduced the thermal advantage of gas engines and hence their use has become difficult to get justified. Some steel plants in Asia and Europe have been successful in the use of direct connected gas turbines for driving generators. Preheated BF gas along with preheated air has been used successfully in coke-oven heating, soaking pits, and reheating furnaces.

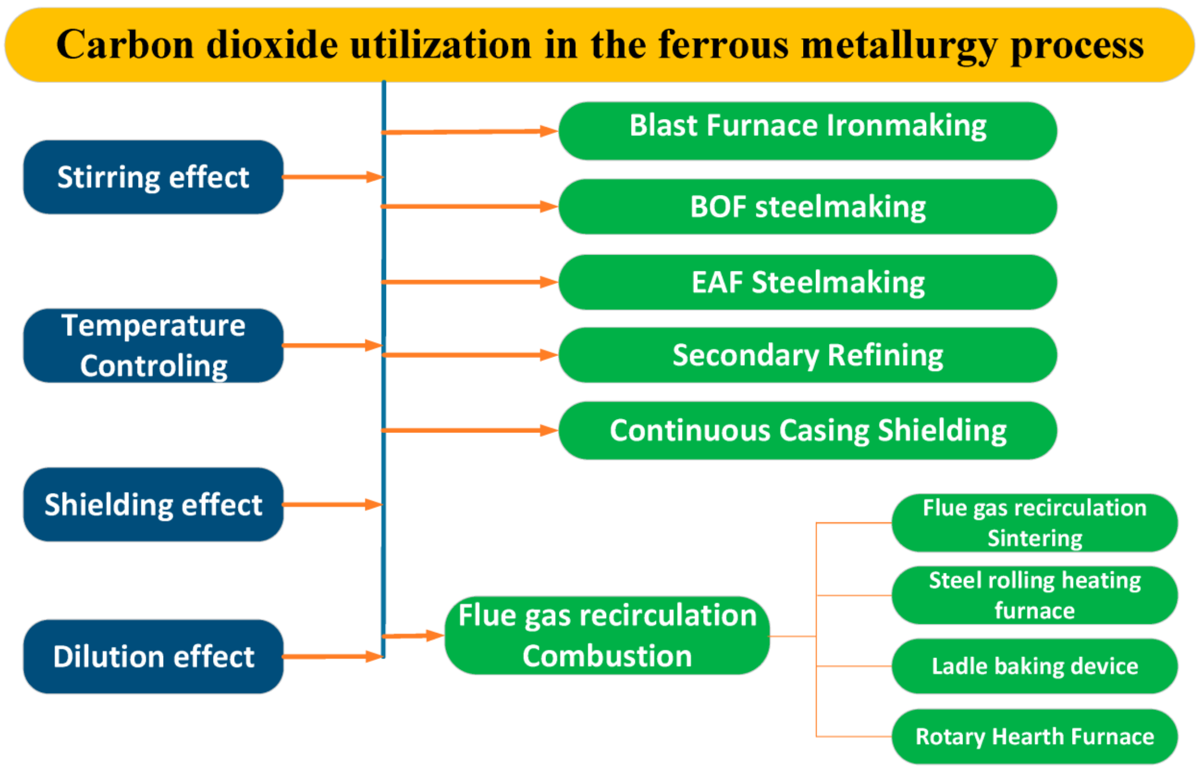




When BF gas is preheated, it is to have a minimum cleanliness of 0.023 g/cum and in all cases where this gas is used, extra precautions is needed to prevent the escape of unburned BF gas into the surroundings since it contains a large percentage of toxic CO gas.

In blast furnace operations, where the BF gas has a heating value approaching a low value of 700 kcal/cum, it becomes necessary to mix the BF gas with other fuel gases to obtain very high temperature of the hot air blast from the stove.

Certainly! Let's delve deeper into the roles of carbon monoxide (CO) and carbon dioxide (CO2) in the steelmaking processes, specifically in Basic Oxygen Steelmaking (BOS) and Electric Arc Furnace (EAF):


Carbon dioxide utilization in the ferrous metallurgy process
Stirring effect
Temperature Controling
Shielding effect
Dilution effect
Blast Furnace Ironmaking
BOF steelmaking
EAF Steelmaking
Secondary Refining
Continuous Casing Shielding
Flue gas recirculation Combustion
Flue gas recirculation Sintering
Steel rolling heating furnace
Ladle baking device
Rotary Hearth Furnace

### Basic Oxygen Steelmaking (BOS):

1. **Oxidation of Impurities:**

- In the BOS process, the initial molten metal is often pig iron, which contains impurities like carbon, silicon, and manganese.

- The introduction of a high-purity oxygen stream into the molten pig iron facilitates the oxidation of these impurities. Carbon in the pig iron reacts with oxygen to form carbon monoxide (\(CO\)):

$$C + O_2 \rightarrow CO$$

- The carbon monoxide produced then acts as a reducing agent in further reactions, particularly with iron oxides present in the pig iron:

$$2Fe_2O_3 + 3CO \rightarrow 4Fe + 3CO_2$$

- This reduction reaction is crucial for extracting iron from the iron ore in the pig iron.

2. **Temperature Control:**

- The combustion of carbon in the form of coke (used as a source of carbon) also contributes to the overall heat balance of the process.

- The heat generated by the combustion of carbon helps to maintain the high temperatures needed for the molten state of the metal and facilitates the fluidity of the molten metal.

### Electric Arc Furnace (EAF):

1. **Decarburization:**

   - In the EAF process, which primarily uses scrap steel as the input material, carbon content control is crucial.

   - Oxygen is injected into the molten steel bath, leading to the combustion of carbon to form carbon dioxide:

$$C + O_2 \rightarrow CO_2$$

   - The carbon dioxide produced can further react with additional carbon to form carbon monoxide:

$$CO_2 + C \rightarrow 2CO$$

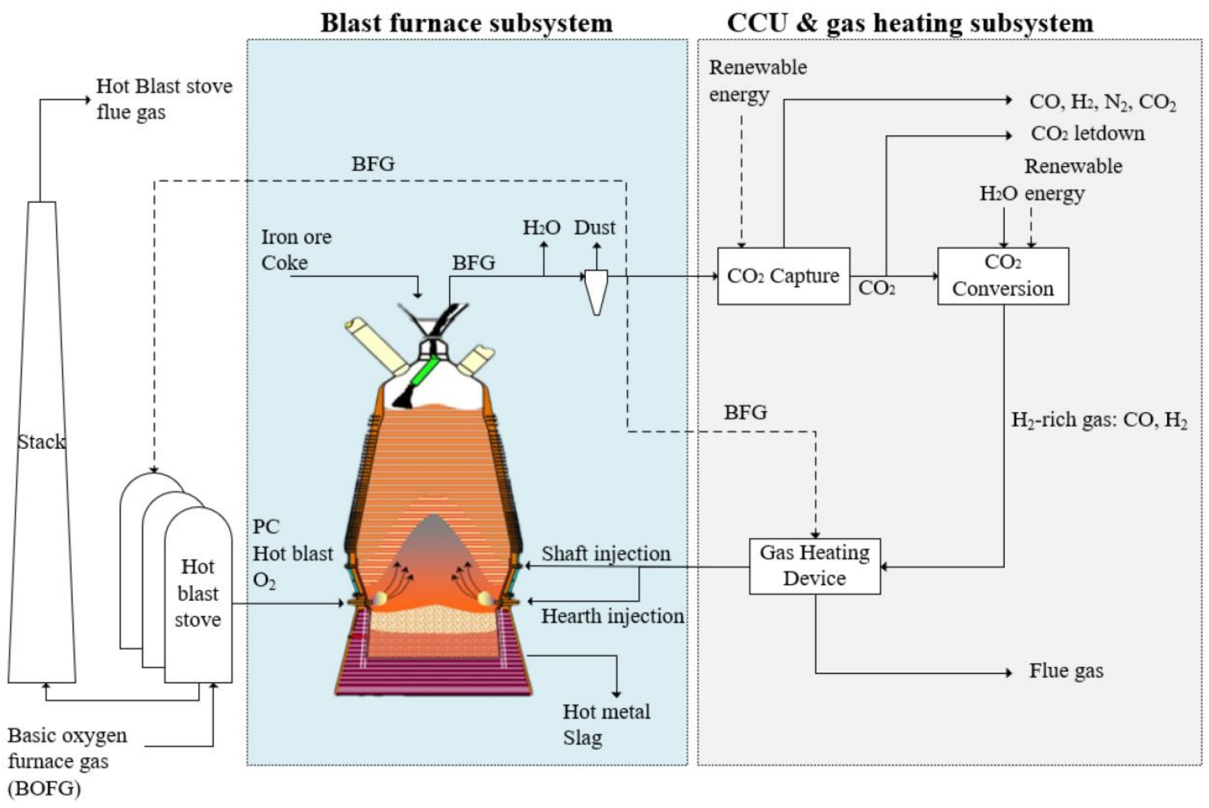
   - This controlled combustion helps in reducing the carbon content of the molten steel, a critical step in achieving the desired steel composition.

2. **Heat Generation:**

   - Similar to BOS, the combustion reactions in the EAF process generate significant heat.

   - The heat contributes to maintaining the high temperatures required for melting the scrap steel and facilitating the chemical reactions involved in the steelmaking process.

In summary, the roles of CO and CO2 in steelmaking are intertwined with the oxidation and reduction reactions that occur during the refining of molten metal. These reactions are fundamental to the removal of impurities, particularly excess carbon, and contribute to the overall energy balance required for efficient steel production. The control of these gases is a crucial aspect of achieving the desired steel composition and quality.



Pattern used in this project :

The prediction of the CO:CO2 ratio in various processes, including those related to industrial activities like steelmaking or combustion, can benefit from machine learning techniques. Here's an overview of how machine learning is applied in the prediction of CO:CO2 ratios:

1. **Data Collection:**

   - The first step in applying machine learning is collecting relevant data. This may include historical data from industrial processes, laboratory experiments, or simulations. The data should cover a range of input variables that influence the CO:CO2 ratio, such as temperature, pressure, composition of input materials, and other process parameters.

2. **Feature Engineering:**

   - Once the data is collected, the next step is feature engineering. This involves selecting and transforming the input variables (features) to create a dataset suitable for training a machine learning model. For CO:CO2 ratio prediction, features may include temperature, pressure, chemical composition, and any other relevant parameters.

3. **Model Selection:**

   - Various machine learning models can be applied to predict the CO:CO2 ratio. Common models include regression algorithms such as linear regression, support vector regression, or more complex models like decision trees, random forests, or neural networks. The choice of the model depends on the complexity of the relationship between the input features and the CO:CO2 ratio.

4. **Training the Model:**

   - The selected model is trained using the prepared dataset. During training, the model learns the patterns and relationships between the input features and the target variable (CO:CO2 ratio). The model adjusts its parameters to minimize the difference between predicted and actual values in the training dataset.

5. **Validation and Tuning:**

- After training, the model needs to be validated using a separate dataset not seen during training. This helps assess the model's generalization performance. Model hyperparameters may be tuned to optimize performance, and techniques like cross-validation may be employed to ensure robustness.

6. **Prediction and Deployment:**

- Once the model is trained and validated, it can be used to predict CO:CO2 ratios for new, unseen data. In an industrial setting, this could be applied in real-time to optimize process control. The model can be integrated into control systems to make predictions and adjustments based on incoming data.

7. **Continuous Improvement:**

- Machine learning models can be periodically retrained with new data to adapt to changes in the process or to improve prediction accuracy over time. This helps ensure that the model remains relevant and effective in dynamic environments.

Machine learning, in this context, enables the creation of predictive models that can capture complex relationships between input variables and the CO:CO2 ratio. The application of machine learning contributes to better process optimization, efficiency, and control in industrial settings.

More details about them

Certainly, let's elaborate on each section to provide a more detailed understanding:

### 1. Data Collection:

In the context of predicting CO:CO2 ratios, data collection involves gathering information related to the process or system under consideration. This data might include historical records of CO:CO2 ratios in different scenarios, along with associated process parameters. These parameters could encompass a wide range of variables such as temperature, pressure, gas flow rates, and chemical composition. The more diverse and representative the dataset, the better the machine learning model can learn the underlying patterns.

### 2. Feature Engineering:

Feature engineering is the process of selecting, transforming, or creating features from the raw data to better represent the problem to the machine learning algorithm. For CO:CO2 ratio prediction, features might include not only the direct input parameters like temperature and pressure but also derived features or interactions between variables. Proper feature engineering enhances the model's ability to capture the complex relationships within the data.

### 3. Model Selection:

Choosing an appropriate machine learning model depends on the characteristics of the problem. Linear regression models assume a linear relationship between input features and the target variable, while more complex models like decision trees or neural networks can capture nonlinear relationships. The choice may involve experimentation and evaluation of different algorithms to find the one that best fits the data and the nature of the CO:CO2 prediction problem.

### 4. Training the Model:

During the training phase, the selected model is fed with the prepared dataset. The model adjusts its internal parameters iteratively to minimize the difference between its predictions and the actual CO:CO2 ratios in the training data. This process involves optimization algorithms that guide the model toward the most accurate representation of the underlying patterns in the data.

### 5. Validation and Tuning:

The trained model is then validated using a separate dataset that it has not seen during training. This step helps assess how well the model generalizes to new, unseen data. If necessary, model hyperparameters (settings external to the model that can be fine-tuned) are adjusted to optimize performance. Techniques like cross-validation may be employed to ensure the model's robustness.

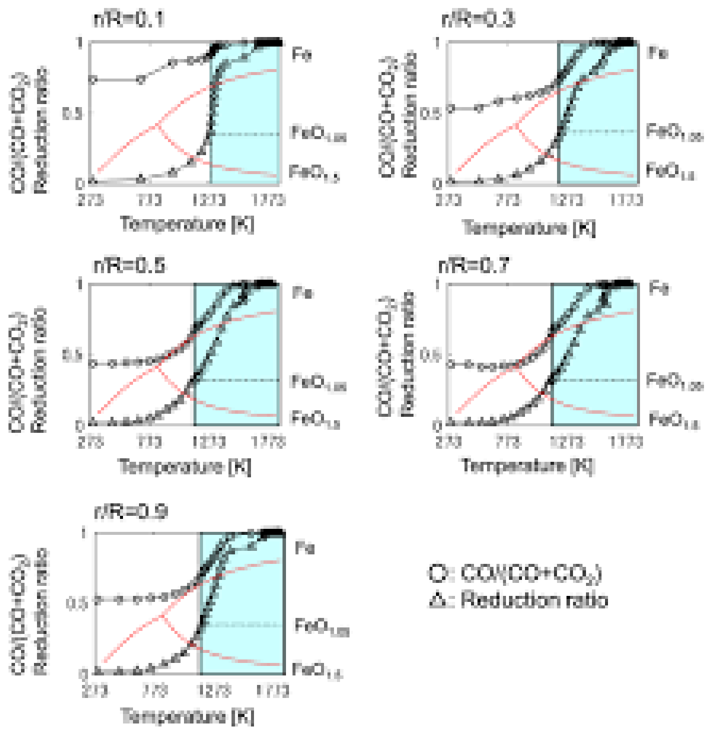
### 6. Prediction and Deployment:

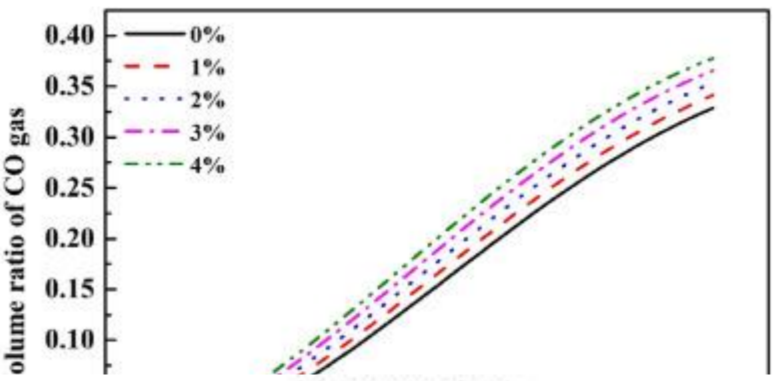
Once the model is trained and validated, it can be deployed to make predictions on new, real-time data. In an industrial context, this might involve integrating the model into control systems. The model can continuously receive incoming data, predict CO:CO2 ratios, and potentially make recommendations or adjustments to optimize the process.

### 7. Continuous Improvement:

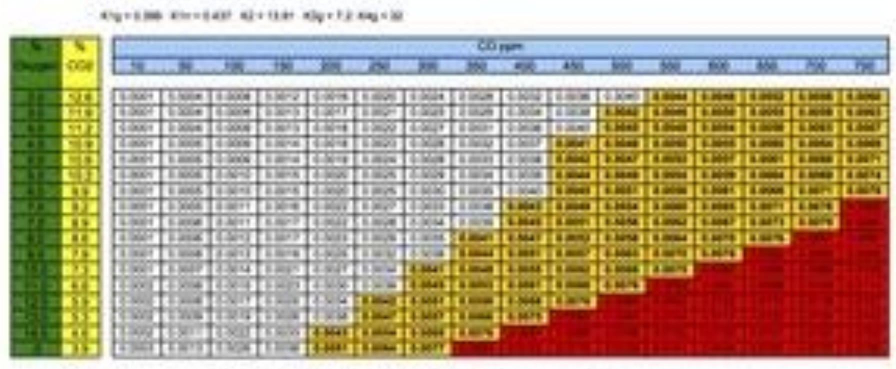
Machine learning models benefit from continuous improvement. Periodic retraining with new data ensures that the model remains accurate and relevant over time. This adaptability is crucial in dynamic environments where processes or conditions may change. Continuous improvement may involve updating the model architecture, re-evaluating features, and incorporating new knowledge into the training process.

In summary, the application of machine learning to predict CO:CO2 ratios involves a systematic process, starting from data collection and culminating in the deployment of a trained model for real-time predictions. Each step contributes to the model's ability to understand and capture the underlying patterns in the data, leading to more accurate predictions and improved process optimization.





CO/CO2 RATIO for LPG



[illegible]

% Oxygen	% CO2	CO ppm															
		10	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750
[illegible]	[illegible]	0.0001	[illegible]	0.0008	0.0012	0.0016	0.0020	0.0024	0.0028	0.0032	0.0036	0.0040	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	11.9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

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), 251-256.
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