Co2 Emission By Countries

PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering Degree

in Computer Science and Engineering by

MADALA JAYA VIGNESH GOPI

BEN MATHEW

Internal Guide

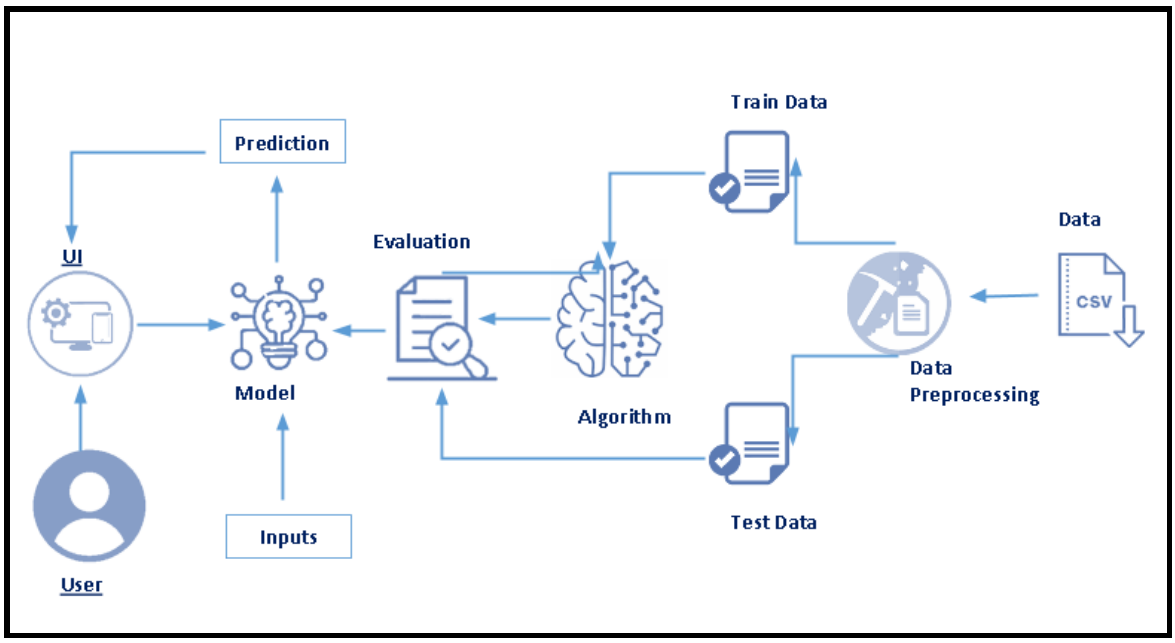
INDRA

Introduction:

Carbon emissions and environmental protection issues have brought pressure from the international community during Chinese economic development. Recently, Chinese Government announced that carbon emissions per unit of GDP would fall by 60–65% compared with 2005 and non-fossil fuel energy would account for 20% of primary energy consumption by 2030.The Beijing-Tianjin-Hebei region is an important regional energy consumption center in China, and its energy structure is typically coal-based which is similar to the whole country.

Therefore, forecasting energy consumption related to carbon emissions is of great significance to emissions reduction and upgrading of energy supply in the Beijing-Tianjin-Hebei region. Thus, this study thoroughly analyzed the main energy sources of carbon emissions including coal, petrol, natural gas, and coal power in this region.

THEORETICAL ANALYSIS:



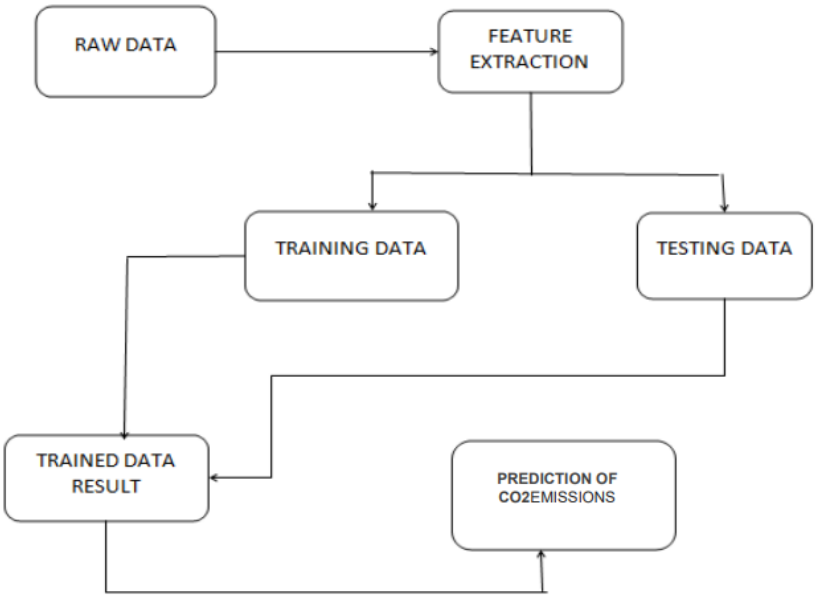


Fig: 4.1: System Architecture

Project Structure

Create a Project folder that contains files as shown below

- We are building a Flask Application which needs HTML pages **“index.html”** ,**“index1.html”** ,**“result.html”** stored in the templates folder and a python script **app.py** for server side scripting
- The model is built in the notebook **co2-emission.ipynb**

- We need the model which is saved and the saved model in this content is **co2.pickle.**
- The static folder will contain a css file which is used in the html file.
- The templates mainly used here are **“index.html”,”index1.html”, “result.html”** for showcasing the UI
- The flask app is denoted as **app.py**

- To complete the project, you need the following packages and libraries.
  1. Anaconda Navigator
  2. Jupyter
  3. Numpy
  4. Pandas
  5. Matplotlib
  6. Scikit-Learn

Data Pre-Processing

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn’t feasible for the analysis.
Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing Follow the following steps to process your Data

- Import the Libraries.
- Importing the dataset.
- Check unique values in the dataset
- Check the CO-2 Emissions of the countries.
- Understanding Data Type and Summary of features.
- Observing Target, Numerical and Categorical Columns
- Checking for Null Values.
- Data Visualization.
- Splitting Data into Train and Test.

Data Visualization

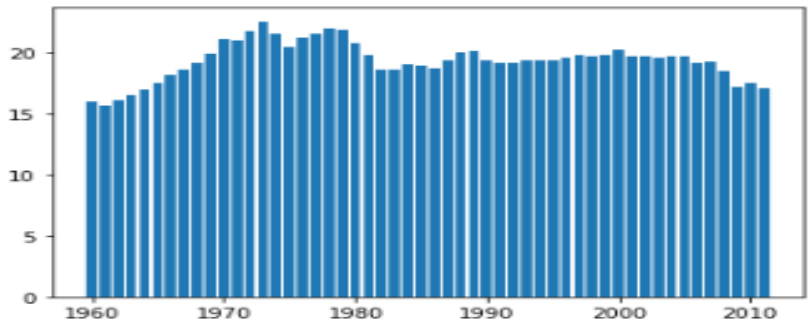
- Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine

learning models will perform poorly on data that wasn't visualized and understood properly.

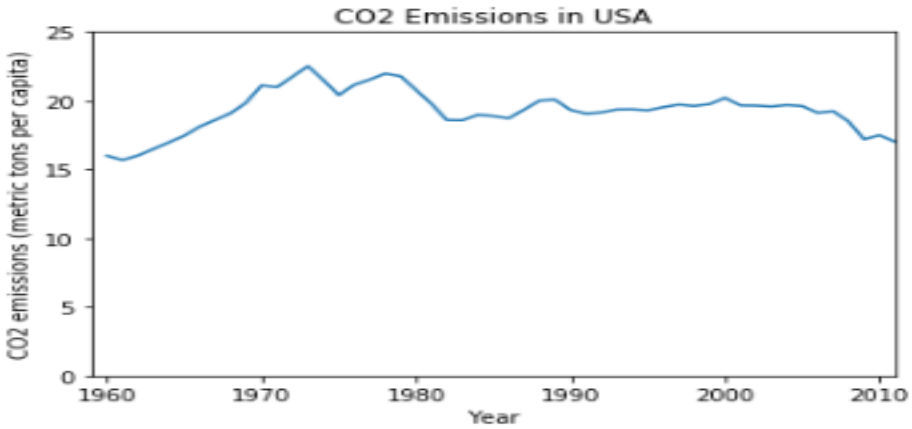
- To visualize the dataset we need libraries called Matplotlib and Seaborn.The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc.
- Let's visualize our data using Matplotlib and the Seaborn library.

- Get the years and co2 emission value in the particular year.

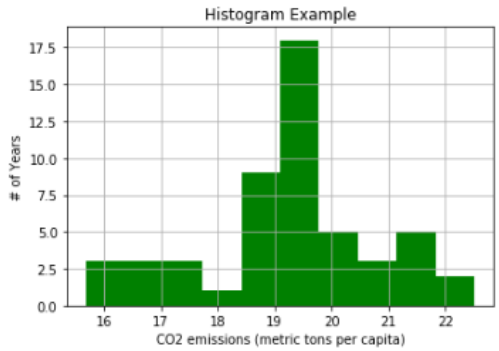
```
# get the years
years = stage['Year'].values
# get the values

co2 = stage['Value'].values

# create
plt.bar(years,co2)
plt.show()
```



- Plot the figure 'CO2 Emissions in USA' (Year vs Indicator Name).

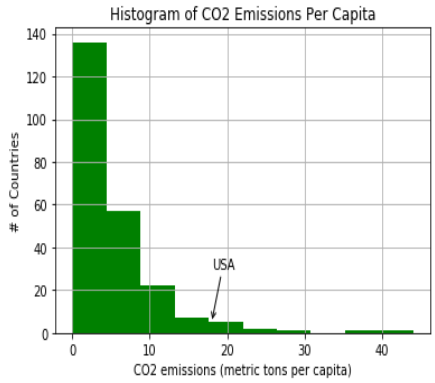


- A histogram is basically used to represent data provided in a form of some groups.It is accurate method for the graphical representation of numerical data distribution.It is a type of bar plot where X-axis represents the IndicatorName while Y-axis gives information about Years.



- Let's plot a histogram of the emissions per capita by country and subplots returns a tuple with the figure, axis attributes

- So the USA, at ~18 CO2 emissions (metric tons per capita) is quite high among all countries.
- An interesting next step, which we'll save for you, would be to explore how this relates to other industrialized nations and to look at the outliers with those values in the 40s!



So the USA, at ~18 CO2 emissions (metric tons per capital) is quite high among all countries.

An interesting next step, which we'll save for you, would be to explore how this relates to other industrialized nations and to look at the outliers with those values in the 40s!

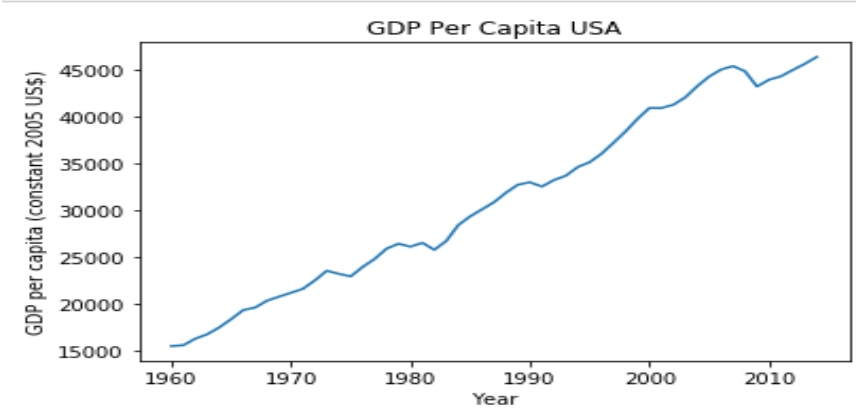
- Relationship between GDP and CO2 Emissions in USA

	CountryName	CountryCode	IndicatorName	IndicatorCode	Year	Value
22282	United States	USA	GDP per capita (constant 2005 US$)	NY.GDP.PCAP.KD	1960	15482.707760
48759	United States	USA	GDP per capita (constant 2005 US$)	NY.GDP.PCAP.KD	1961	15578.409657
77142	United States	USA	GDP per capita (constant 2005 US$)	NY.GDP.PCAP.KD	1962	16276.426685
105760	United States	USA	GDP per capita (constant 2005 US$)	NY.GDP.PCAP.KD	1963	16749.789436
134798	United States	USA	GDP per capita (constant 2005 US$)	NY.GDP.PCAP.KD	1964	17476.822248

```
stage.head(2)
```

	CountryName	CountryCode	IndicatorName	IndicatorCode	Year	Value
22232	United States	USA	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1960	15.999779
48708	United States	USA	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1961	15.681256

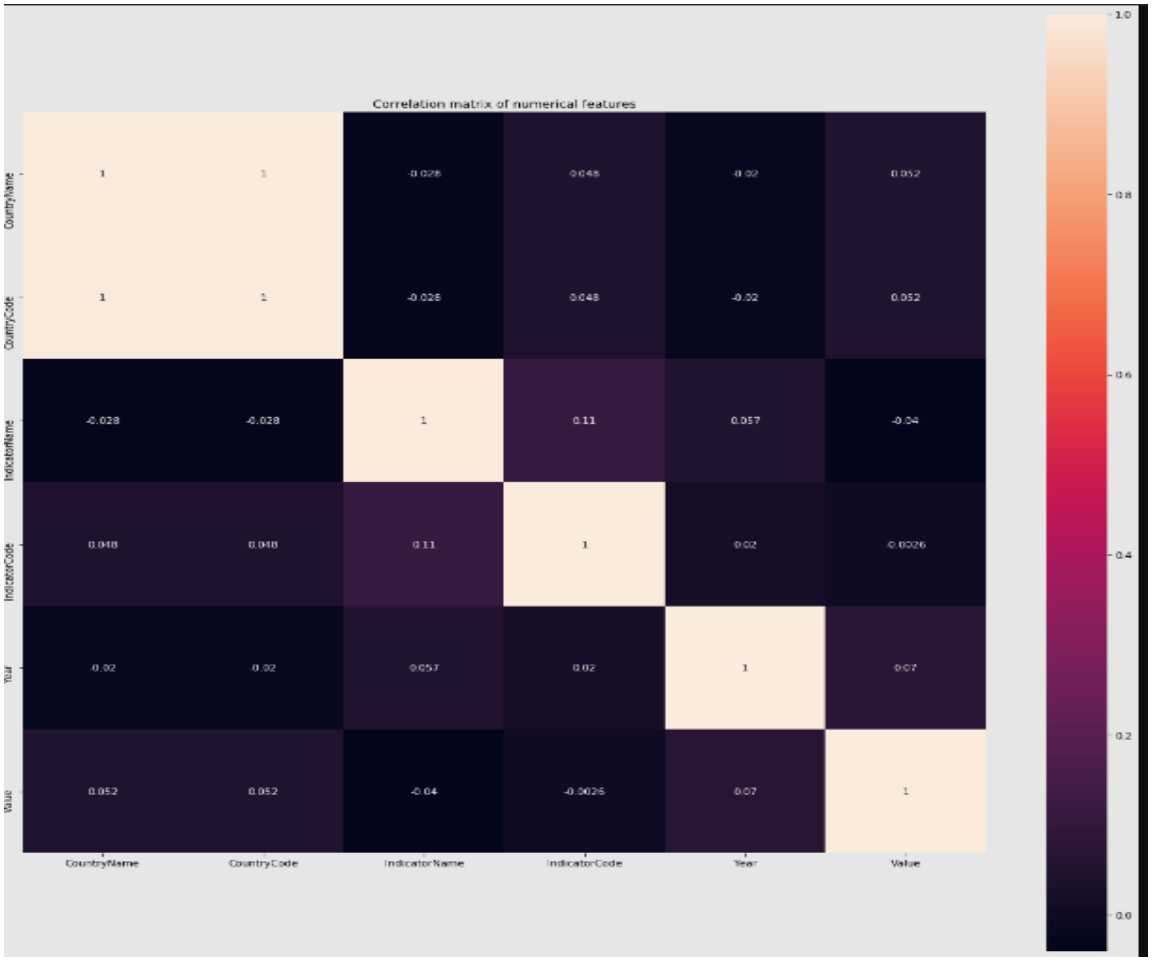
- So although we've seen a decline in the CO2 emissions per capita, it does not seem to translate to a decline in GDP per capita.



Finding correlation between the independent Columns

- Correlation is a statistical relationship between two variables and it could be positive, meaning both variables move in the same direction, or negative, meaning that when one variable's value increases, the other variables' values decrease.
- With the help of seaborn heatmap we will be plotting the heatmap and for finding the correlation between variables we have corr() available.

- If you observe the heatmap, lighter the colour the correlation between that two variables will be high.



- And correlation plays a very important role for extracting the correct features for building our model.

Splitting the Dataset into Dependent and Independent variables.

- In machine learning, the concept of dependent variable (y) and independent variables(x) is important to understand. Here, Dependent variable is nothing but output in the dataset and the independent variable is all inputs in the dataset. We can denote with any symbol (alphabets). In our dataset we can say that class is the dependent variable and all other columns are independent. But in order to select the independent columns we will be selecting only those columns which are highly correlated and of some value to our dependent column.
- With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.
- Let's create out independent and dependent variables:

- In the above code we are creating a DataFrame of the independent variable x with our selected columns and for dependent variable y we are only taking the class column.
- Where DataFrame is used to represent a table of data with rows and columns.

Split the dataset into Train set and Test set

- When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will have a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
- But the question is, how do you split the data? You can't possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is,'train_test_split.' Using this we can easily split the dataset into the training and the testing datasets in various proportions.
- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.We will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.
- There are a few other parameters that we need to understand before we use the class:
- test_size — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.4 as the value, the dataset will be split 40% as the test dataset
- train_size — you have to specify this parameter only if you're not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
- random_state — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which will become the number generator. If you don't pass anything, the Random_state instance used by np.random will be used instead.
- Now split our dataset into a train set and test using train_test_split class from scikit learn library.

Splitting dataset into train and test

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
print(x_train.shape)
print(x_test.shape)
```

```
(93977, 4)
(23495, 4)
```

Model Building

Train and Test the Model using Random Forest Regressor.

- There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have may be Classification algorithms are Regression algorithms.

Example: 1. Linear Regression.

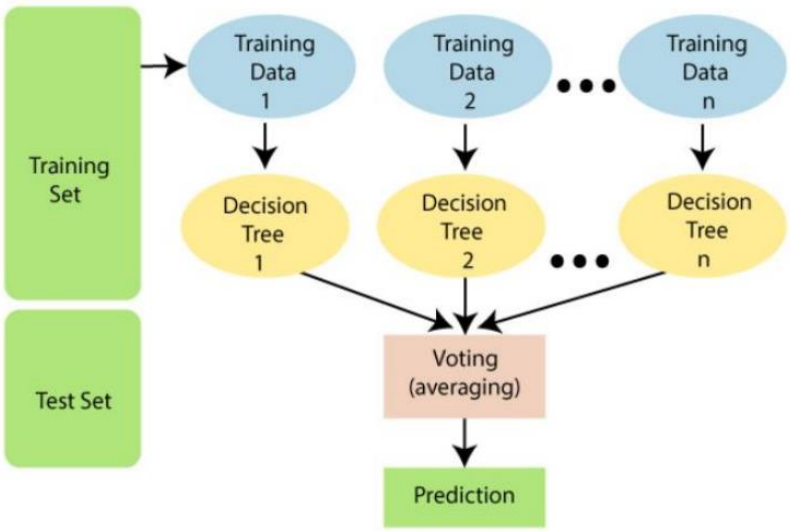
2. Logistic Regression.

3. Random Forest Regression / Classification.

4. Decision Tree Regression / Classification.

- You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.
- Now we apply the Random forest regressor algorithm on our dataset.
- A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

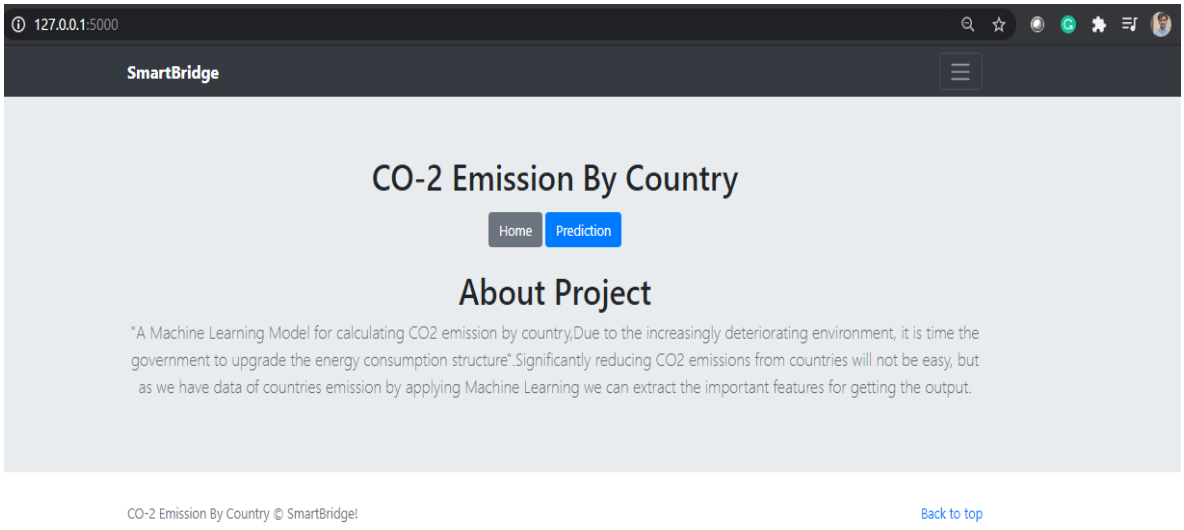
$$Y = a + bX$$



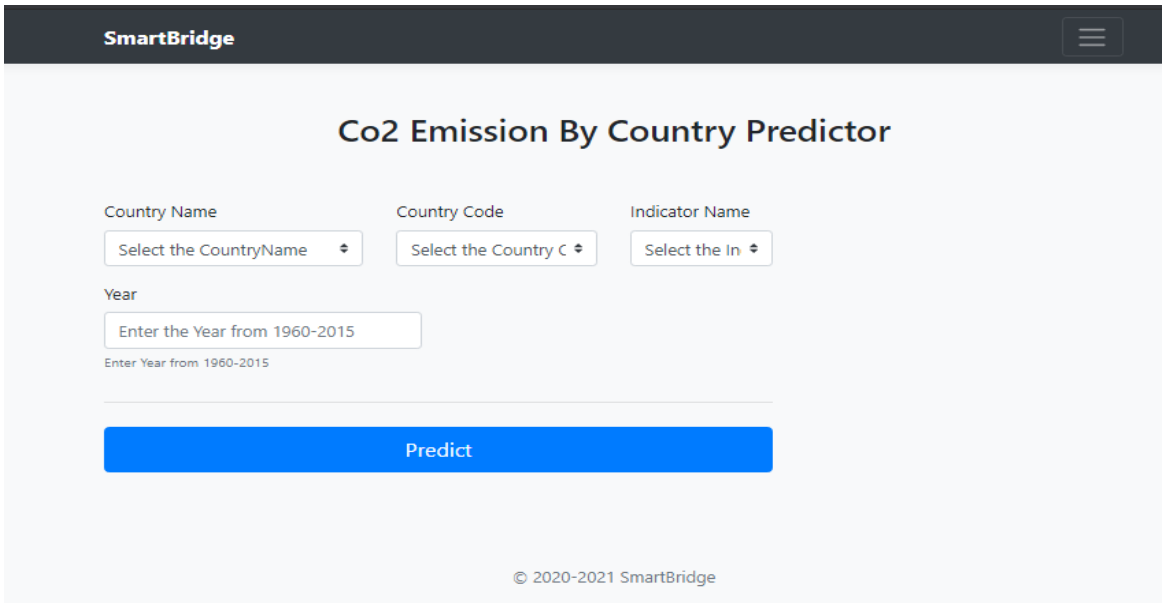
Output

Dashboard of the flask application.

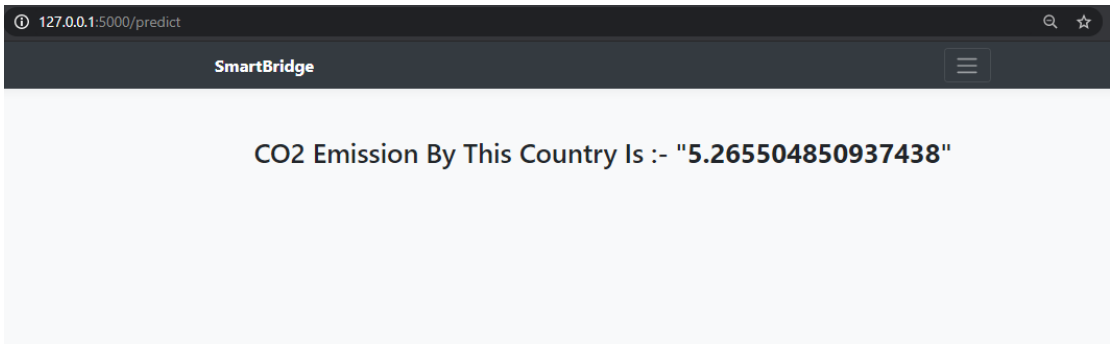
- This is the main page of C02-Emission By Country where you can know about the project and also from this page user can click onto the prediction button and they will redirect onto the prediction page for providing the inputs.



- The prediction page user gives the input for predicting the output where they can choose CountryName ,CountryCode,IndicatorName and input the Year in the range of 1960-2015 based on the dataset and click to predict the output.



- In the predict page user will get the output based on the inputs they provide in the prediction page.



PURPOSE

The purpose of the system is mainly to reduce the co2 emissions and find the factors that are most responsible for the carbon dioxide emissions and to reduce the usage of the factors effecting co2 emissions.

SCOPE

It can be further developed by including more features. The carbon dioxide emissions can be reduced by including some features like

- Switching to Electric Vehicles
- Usage of Solar panels and many more,
- therefore you can improve model accuracy.

OBJECTIVIES

- To predict the carbon dioxide emissions using ML model.
- To identify the features that are causing more carbon dioxide emissions.
- To reduce their usage in future.

ADVANTAGES

1. Accuracy: Machine learning models can provide accurate predictions and analysis of CO2 emissions by leveraging large datasets and complex algorithms, leading to more reliable results compared to traditional methods.
2. Real-time Monitoring: The proposed solution can continuously monitor CO2 emissions in real-time, allowing for prompt detection and response to changes or anomalies.
3. Scalability: Machine learning models can be easily scaled up to handle large volumes of data, making them suitable for monitoring CO2 emissions across multiple industries and sectors.
4. Efficiency: By automating the analysis process, the solution can significantly reduce the time and effort required to track and measure CO2 emissions, enabling organizations to make faster and more informed decisions.
5. Predictive Insights: Machine learning models can identify patterns and trends in CO2 emissions data, providing valuable insights that can help organizations proactively address potential environmental issue.

DIS-ADVANTAGES

1. Data Quality: The accuracy of the machine learning model heavily relies on the quality and reliability of the input data. Inaccurate or incomplete data can lead to biased or misleading results.
2. Data Availability: Access to comprehensive and up-to-date CO2 emissions data across various industries and sectors may be limited, hindering the model's effectiveness.
3. Model Complexity: Developing and maintaining a machine learning model for CO2 emissions requires specialized knowledge and expertise, making it challenging for organizations without adequate resources or technical capabilities.
4. Interpretability: Machine learning models can often be seen as "black boxes," making it difficult to understand how they arrive at their predictions. This lack of interpret

APPLICATIONS

1. Carbon Footprint Estimation: Machine learning models can analyze data from various sources, such as energy consumption, transportation, and industrial activities, to estimate carbon footprints for individuals, organizations, or regions. This information can help raise awareness and guide efforts to reduce emissions.
2. Energy Efficiency Optimization: Machine learning models can analyze energy consumption patterns and identify opportunities for optimizing energy usage. By providing recommendations on energy-efficient practices, these models can help reduce CO2 emissions in buildings, industrial processes, and transportation.
3. Renewable Energy Integration: Machine learning algorithms can analyze weather patterns, historical energy production data, and other variables to predict renewable energy generation. This information can be used to optimize the integration of renewable sources into the power grid, reducing reliance on fossil fuels and minimizing CO2 emissions.
4. Smart Grid Management: Machine learning models can analyze electricity demand patterns, consumer behavior, and grid infrastructure data to optimize the management and distribution of electricity. By improving grid efficiency, these models can reduce energy waste and associated CO2 emissions.
5. Emissions Monitoring and Compliance: Machine learning can assist in monitoring and predicting CO2 emissions from industrial facilities. By analyzing real-time sensor data and historical emissions records, these models can help identify potential non-compliance instances and enable proactive measures to mitigate emissions.

APPENDIX

1. Data Collection:
    - Data Sources: The model utilizes a diverse set of data sources to capture relevant information. These sources may include government databases, research publications, industry reports, and real-world driving data.
    - Data Preprocessing: Raw data is subjected to preprocessing techniques such as cleaning, normalization, feature engineering, and handling

missing values. These steps ensure data quality and enhance the model's performance.

2. Feature Selection:
    - Input Features: The model incorporates various features that influence CO_2 emissions. These features encompass vehicle attributes (engine size, weight, fuel type), driving behavior (average speed, acceleration patterns), and contextual factors (temperature, road type, traffic conditions). The specific set of features used can vary based on the model's design and data availability.
    - Feature Engineering: Advanced techniques like dimensionality reduction (e.g., principal component analysis) and feature scaling may be employed to enhance the model's effectiveness.
3. Model Architecture:
    - Regression Model: The CO_2 emission prediction is framed as a regression problem. Various machine learning algorithms can be used for this purpose, such as linear regression, decision trees, random forests, support vector regression, or neural networks.
    - Model Training: The model is trained on the collected and preprocessed data. The dataset is split into training and validation sets to assess the model's performance.
    - Hyperparameter Tuning: Hyperparameters of the chosen algorithm are fine-tuned using techniques like grid search, random search, or Bayesian optimization to optimize the model's performance.
4. Model Evaluation:
    - Evaluation Metrics: The model's performance is assessed using appropriate metrics for regression tasks, such as mean squared error (MSE), mean absolute error (MAE), or R-squared (R^2).
    - Cross-Validation: To obtain a robust estimate of the model's performance, k-fold cross-validation is often employed, where the dataset is divided into k subsets, and the model is trained and evaluated k times, with each subset used as the validation set once.
5. Deployment and Monitoring:
    - Deployment: Once the model is trained and evaluated, it can be deployed in a production environment. This may involve integrating it into an application, website, or other systems that can utilize the CO_2 emission predictions.
    - Monitoring: Continuous monitoring of the model's performance and data quality is crucial. This includes tracking prediction accuracy, retraining the model periodically using updated data, and addressing any issues or biases that may arise.

It is important to note that the specifics of the CO_2 emission machine learning model can vary depending on the chosen approach and available data. The provided

appendix outlines a general framework for developing such a model and should be adapted based on the specific requirements and constraints of the project.