# Emission of CO2 from Cars

A typical passenger vehicle emits about 4.6 metric tons of carbon dioxide per year. This assumes the average gasoline vehicle on the road today has a fuel economy of about 22.0 miles per gallon and drives around 11,500 miles per year.

Significantly reducing CO2 emissions from transport will not be easy, as the rate of emission reductions has slowed. Other sectors have cut emissions since 1990, but as more people become more mobile, CO2 emissions from transport are increasing.

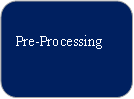
Efforts to improve the fuel efficiency of new cars are also slowing. After a steady decline, newly registered cars emitted on average 0.4 grammes of CO2 per kilometre more in 2017 than the year before.

**Machine Learning has following four steps:**

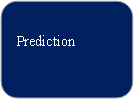
1. Data Collection
2. Data Pre-processing
3. Model Building
4. Application Building













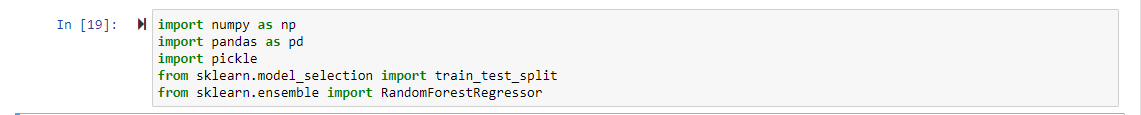


**Data Collection:**

The given dataset is related to Emission of Co2 from cars. Dataset was downloaded from kaggle.com website.

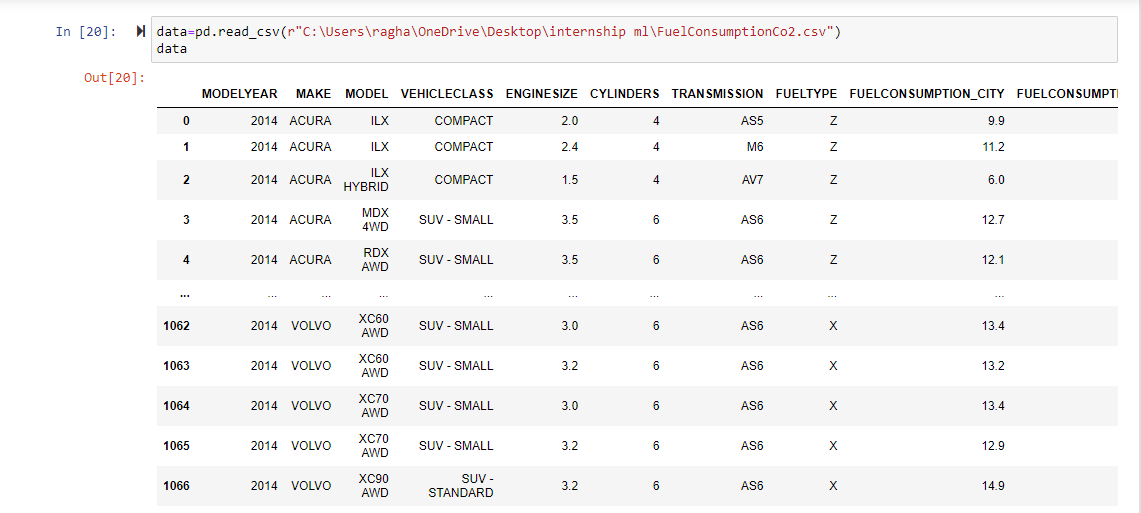
**Data Pre-Processing**

Importing the required packages



1. **Pandas:** It is a python library mainly used for data manipulation.
2. **Numpy:** This python library is used for numerical analysis.
3. **pickle:**
4. **Train\_test\_split:** used for splitting data arrays into training data and for testing data.

**Importing the dataset:**

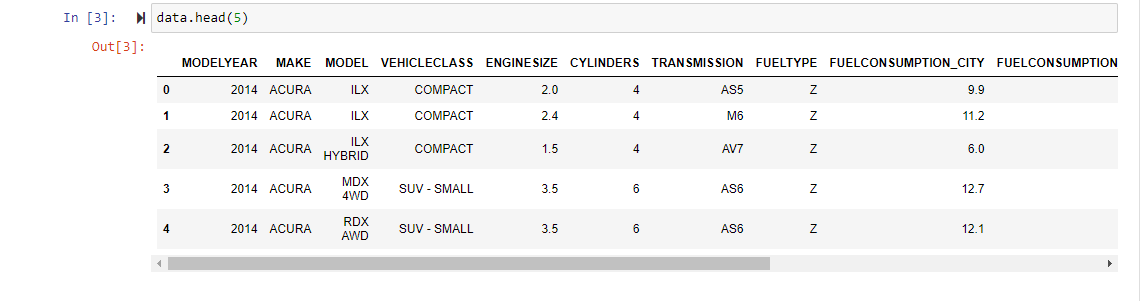


* We might have the data in .csv files, .excel files or .tsv files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that’s compatible with pandas.
* Let’s load a .csv data file into pandas. There is a function for it, called **read\_csv().**We will need to locate the directory of the CSV file at first (it’s more efficient to keep the dataset in the same directory as your program).
* here, the data displays the all the values in the dataset.

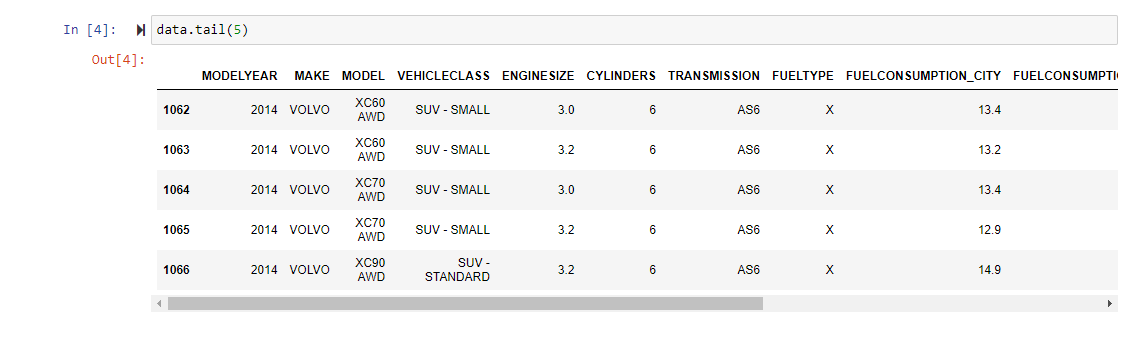
**Data Visualization:**

Exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods and used for determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

* To check first five rows of dataset, we have a function call head( ).This returns the first 5 values of the dataset.



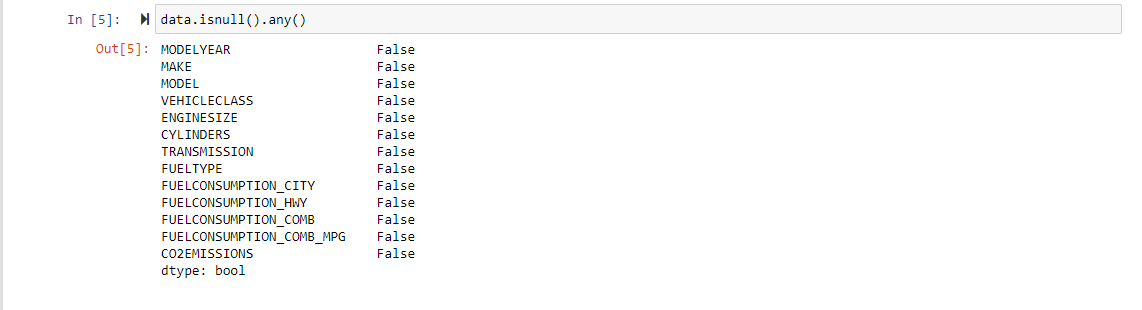
* To check last 5 values of the dataset, we use the tail().



**Taking care of Missing Data:**

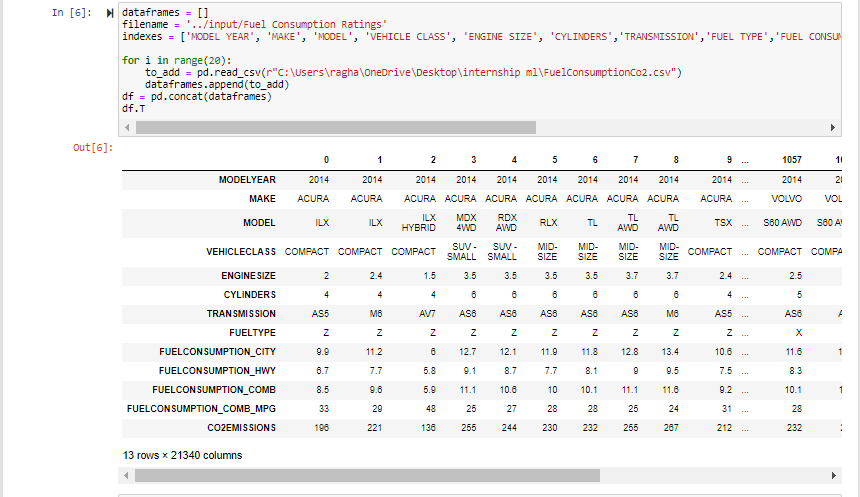
Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously, you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values for continuous and for categorical we make use of mode values and replace the missing data.

* We will be using isnull().any() method to see which column has missing values.



* Since there are no missing values in the dataset, no need to execute this step.

**Converting the data into dataframes:**

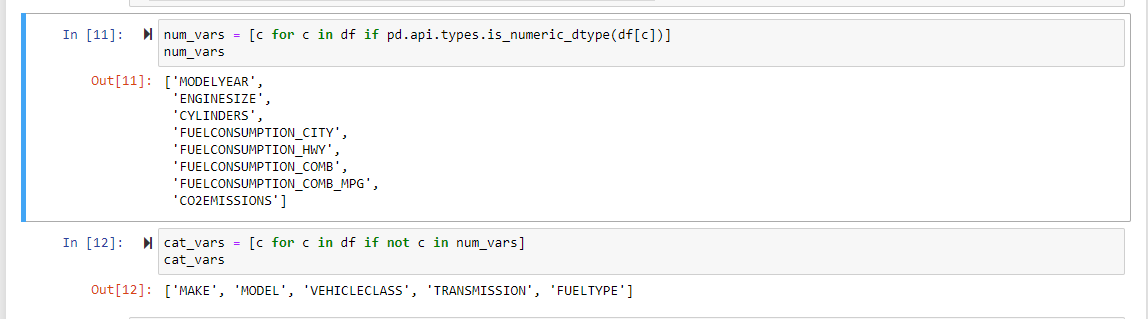


**Dropping the duplicate values:**

* we use this syntax to drop the duplicate values in the datasset drop\_duplicates()



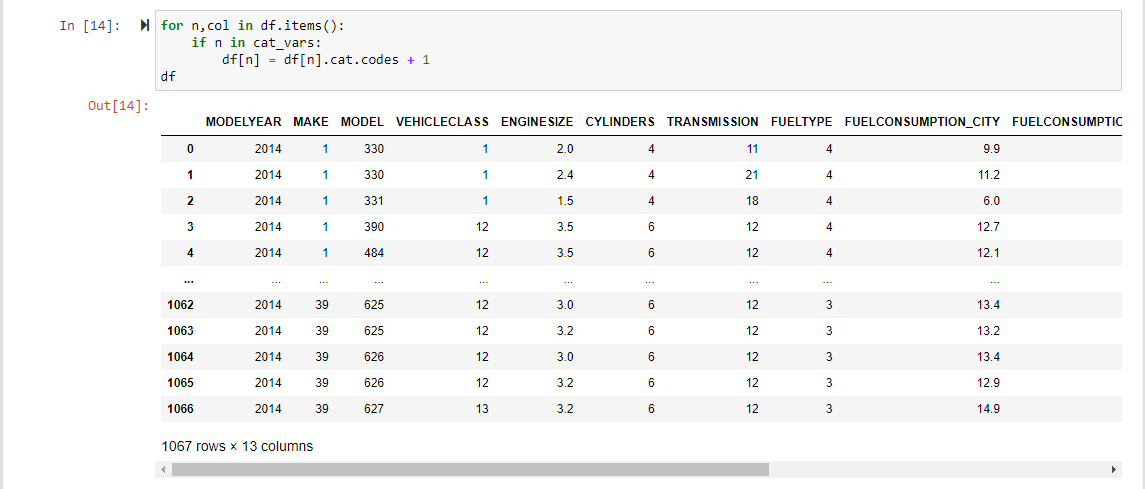
**Checking which are numeric data and which are not:**



**Converting the categorical data into numbers**



**After Label encoding the dataset displaying the data:**



**Feature Scaling:**

1. Splitting Data into Train and Test:

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 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.

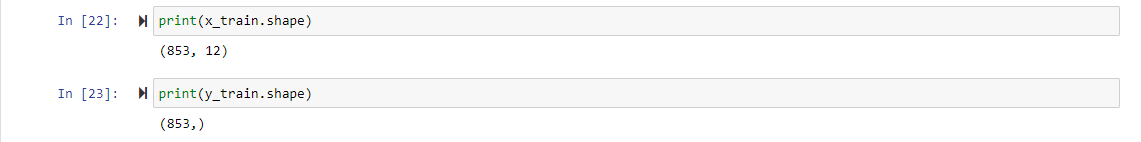
In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.

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:

1. **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 0.5 as the value, the dataset will be split 50% as the test dataset
2. **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.
3. **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 train set and test using train\_test\_split class from scikit learn library.



**3. Model Building**

Training and testing the model:

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 it may be Classification algorithms are Regression algorithms.

Example: 1. Linear Regression.

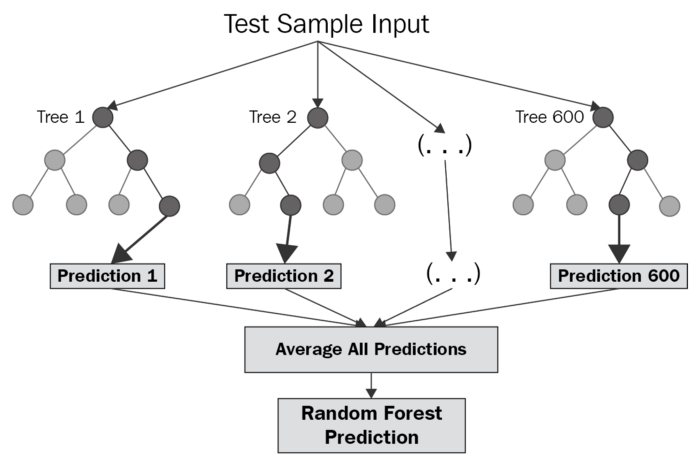
2. Logistic Regression.

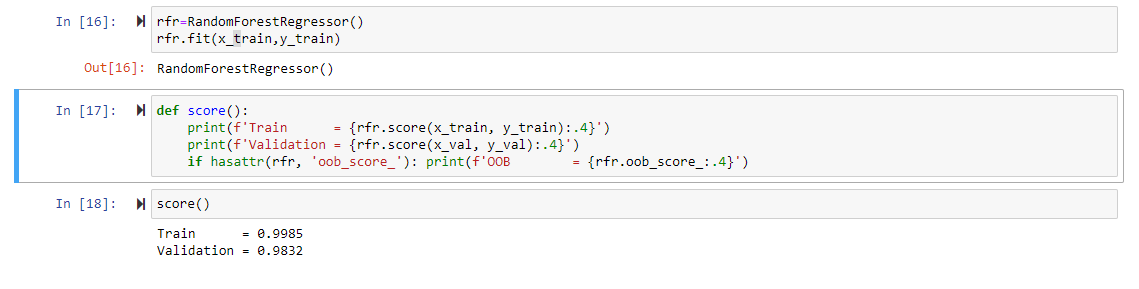
3. Random Forest Regression / Classification.

4. Decision Tree Regression / Classification.

**Now we apply Random Forest Regression algorithm on our dataset.**

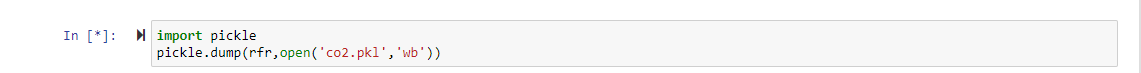
Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression.





**Saving a model:**

Model is saved so it can be used in future and no need to train it again.



**4. Application Building**

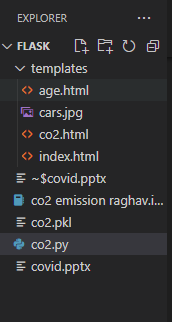
Creating a HTML File and flask application.

Build python code

* 1. Importing Libraries
  2. Routing to the html Page
  3. Showcasing prediction on UI
  4. Run The app in local browser

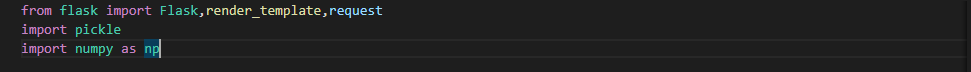
**Project Structure:**

Create a Project folder that contains files as shown below



* We are building a Flask Application that needs HTML pages stored in the templates folder
* Templates folder contains co2.html
* And in the other folder we have the python file which has connection to the HTML page named as co2.py
* Model is saved so it can be used in future and no need to train it again and is named as co2.pkl

**Task 1: Importing Libraries**



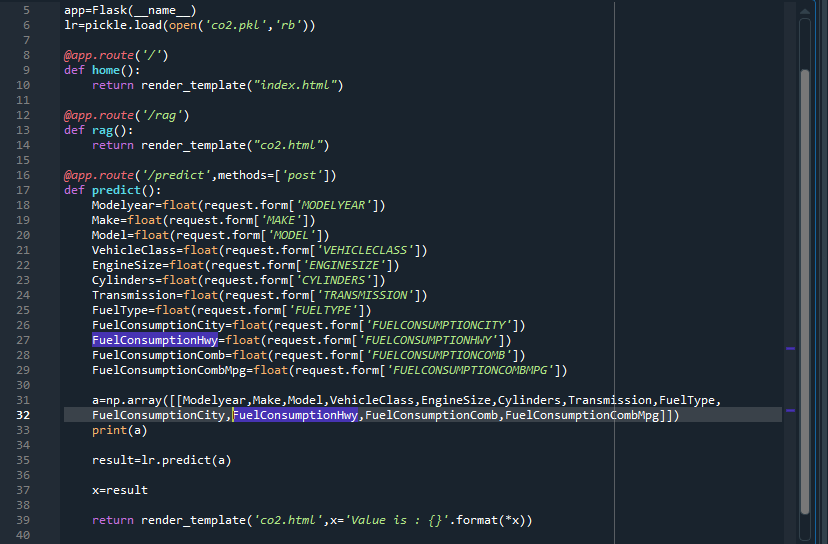
Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument pickle library to load the model file.

**Task 2: Routing to the html Page**

Here, declared constructor is used to route to the HTML page created earlier.

In the above example, ‘/’ URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page is rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Here, “co2.html” is rendered when home button is clicked on the UI



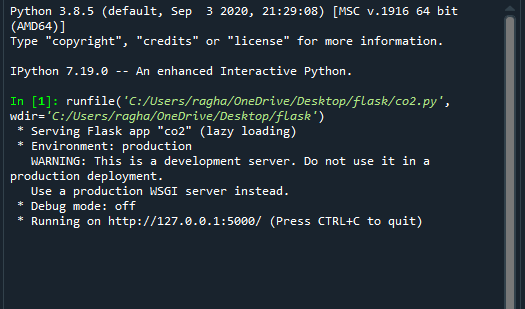
**Task 3: Main Function**

This is used to run the application in a local host.

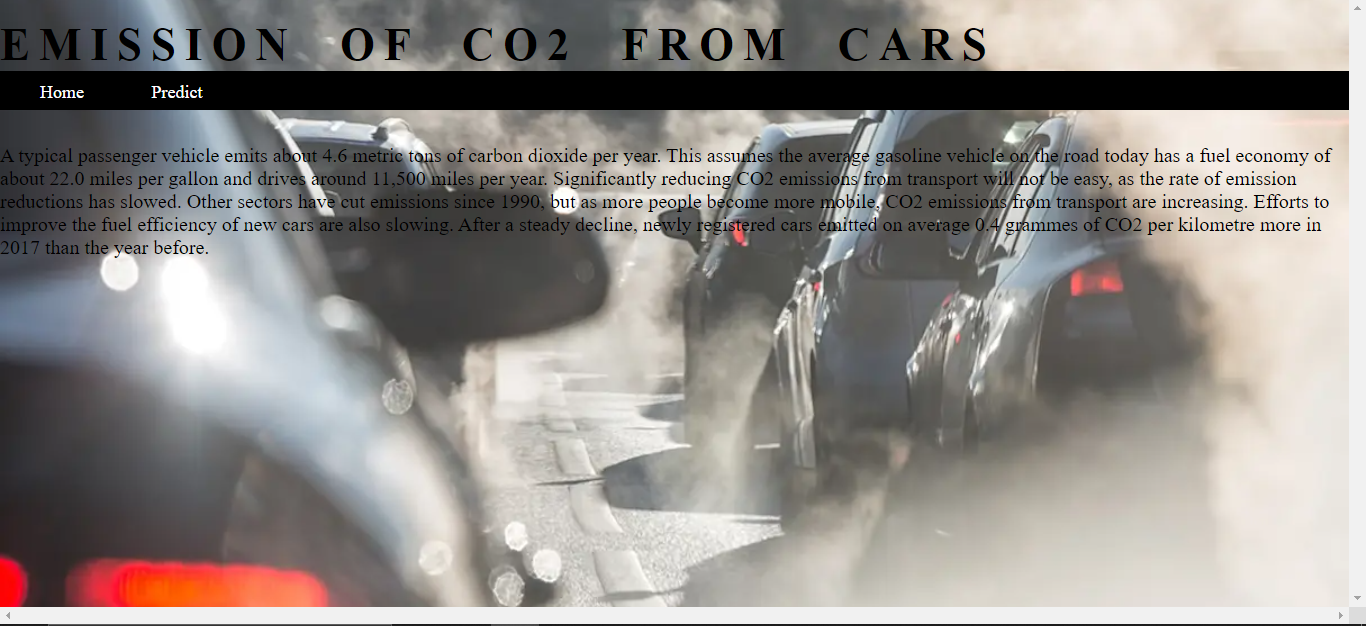


**Activity 3: Run the application**

1. Open the anaconda prompt from the start menu.
2. Navigate to the folder where your app.py resides.
3. Now type “python co2.py” command.
4. It will show the local host where your app is running on **http://127.0.0.1:5000/**
5. Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
6. Enter the values, click on the predict button and see the result/prediction on the web page.

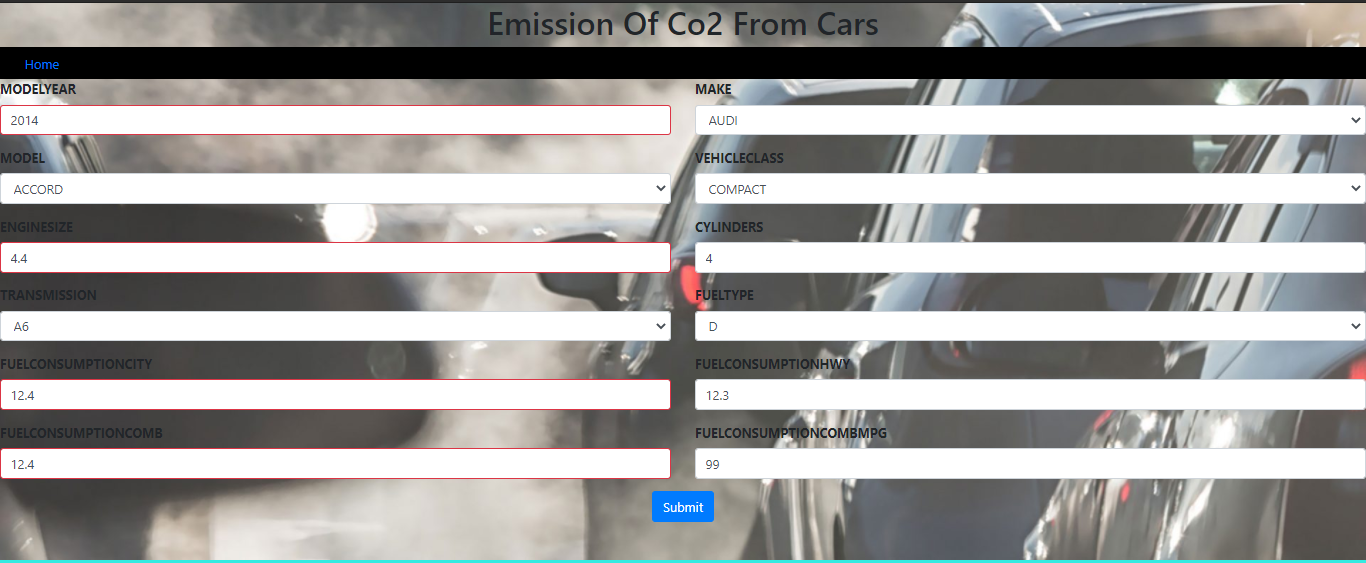


**OUTPUT SCREEN:**



Here when the predict button is pressed it is redirected to Predict page which is named as “**co2.html**”.

Here we are inserting the values in it.



After, entering the values we can see the value is 247.77

