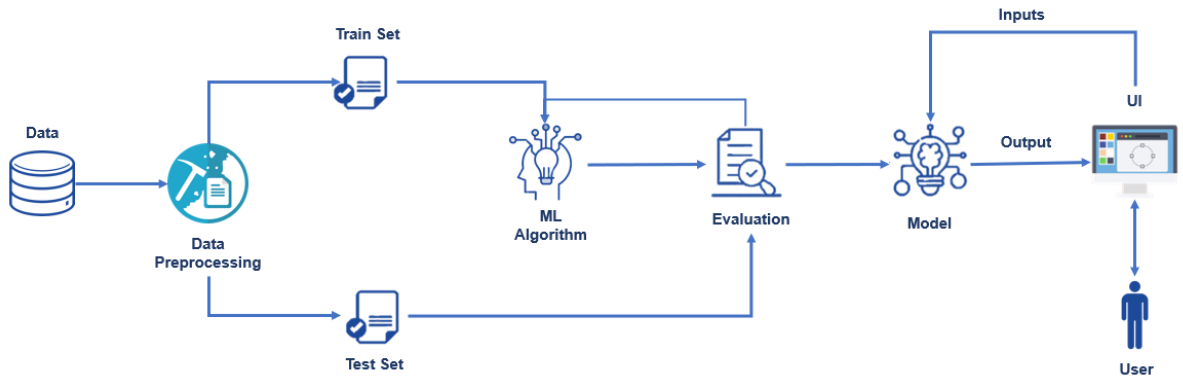
Predictive Modeling For Fleet Fuel Management using Machine Learning

Project Description:

Ability to model and predict the fuel consumption is vital in enhancing fuel economy of vehicles and preventing fraudulent activities in fleet management. Fuel consumption of a vehicle depends on several internal & external factors However, not all these factors may be measured or available for the fuel consumption analysis.

The main aim of the project is to build Machine Learning algorithm to predict the fuel consumption of fleet vehicles based on the gas type . A web application is built which is integrated with ML model

Technical Architecture:



Project Objectives:

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- Applying different algorithms according to the dataset
  You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework.

Project Flow:

- Download the dataset.
- Preprocess or clean the data.
- Analyze the pre-processed data.
- Train the machine with preprocessed data using an appropriate machine learning algorithm.
- Save the model and its dependencies.
- Build a Web application using flask that integrates with the model built.

Pre-Requisites:

In order to develop this project we need to install the following software/packages:

Anaconda Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video and click here to download anaconda IDE

Link: Click here to Watch video

Python packages:

NumPy: NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object.

Pandas: pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

Scikit-learn:

It is? ?a? ?free? ?machine learning library? ?for? ?Python.? ?It? ?features? ?various? ?algorithms? ?like? ?support? ?vector? ?machine,? ?random? ?forests,? ?and? ?k-neighbours,? ?and? ?it? ?also? ?supports? python? ?numerical? ?and? ?scientific? ?libraries? ?like? ?NumPy? ?and? ?SciPy?.

Flask: Web framework used for building Web applications

Link: Watch the video to install packages

If you are using anaconda navigator, follow the below steps to download the required packages:

- Open anaconda prompt.
- Type “pip install joblib” and click enter.

Prior Knowledge:

One should have knowledge of the following Concepts

Watch the below video to know about the types of machine learnings

Supervised and unsupervised learning:

Link: https://www.youtube.com/watch?v=kE5QZ8G_78c&feature=emb_logo

Linear Regression

Link:https://youtu.be/nk2CQITm_eo

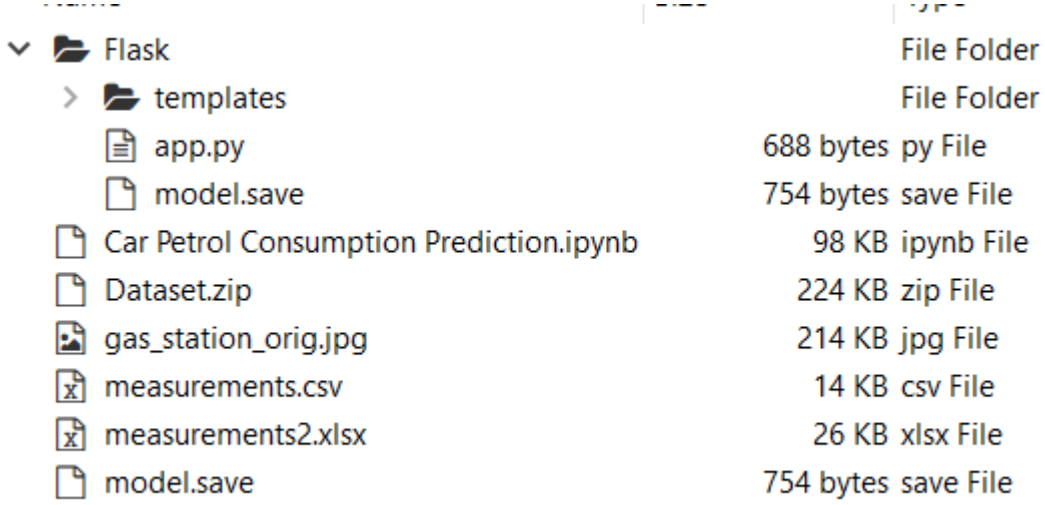
Jupyter Notebook:

Link:https://www.youtube.com/watch?v=HW29067qVWk

Flask:

Link: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:



- Car Petrol Consumption Prediction.ipynb is the jupyter notebook file where the model is built.
- Dataset.zip is the dataset file used in this project.

- model.save is the model file that generates when the notebook file is executed.
- Flask folder is the application folder where the web application and server-side program are present.
- Measurements.csv & measurements.xlsx are the dataset files

.

Milestone 1: Data Collection

For any Machine learning project, data is the primary source. Download the dataset and place it in the project folder.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link:https://www.kaggle.com/datasets/anderas/carconsume?select=measurements2.xlsx

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Milestone 1: Pre-Process The Data

As we have understood how the data is. Let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

1. Handling the null values.
2. Handling the categorical values if any.
3. Normalize the data if required.

4. Identify the dependent and independent variables.
5. Split the dataset into train and test sets.

Activity 1: Importing required libraries

Go to the project folder which you have created copy the project path and open anaconda prompt from the menu and go to the location of your project folder in anaconda prompt and type jupyter notebook. Now jupyter notebook will be opened and create a python file and start the programming.

```
In [1]:
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

.

Activity 2: Read the Dataset

The Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.
In pandas, we have a function called read_excel() to read the dataset. As a parameter, we have to give the directory of xlsx file.

```
4
5 df=pd.read_excel("measurements2.xlsx")
6
7 print(df.head())
```

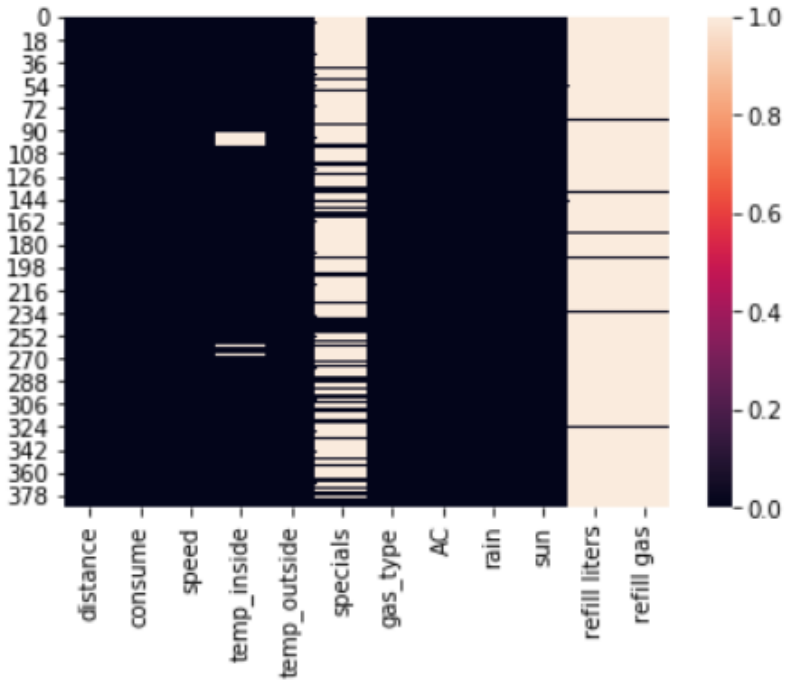
```
   distance  consume  speed  temp_inside  temp_outside specials gas_type  AC  \
0      28.0      5.0     26         21.5            12      NaN      E10   0
1      12.0      4.2     30         21.5            13      NaN      E10   0
2      11.2      5.5     38         21.5            15      NaN      E10   0
3      12.9      3.9     36         21.5            14      NaN      E10   0
4      18.5      4.5     46         21.5            15      NaN      E10   0

   rain  sun  refill liters refill gas
0     0    0           45.0        E10
1     0    0            NaN        NaN
2     0    0            NaN        NaN
3     0    0            NaN        NaN
4     0    0            NaN        NaN
```

Activity 3: Check Null Values

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. To visualize the null values heatmap() and barplot() from seaborn package is used.

In [3]:
```
1 import seaborn as sns
2 sns.heatmap(df.isnull())
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x2c192931c88>



In [3]:
```
1 df.isnull()
```

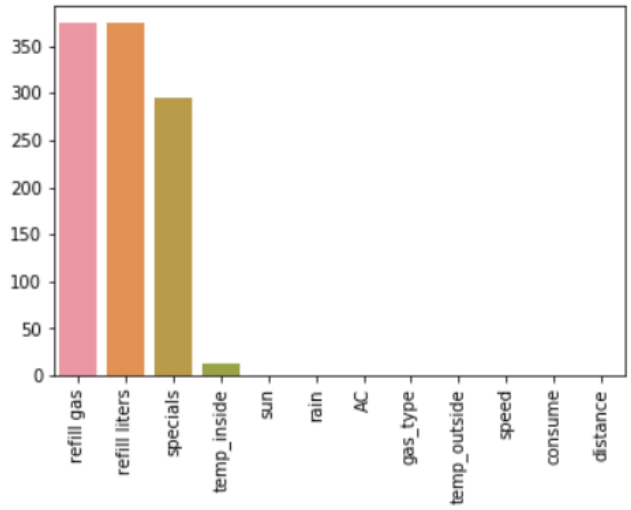
Out[3]:

	distance	consume	speed	temp_inside	temp_outside	specials	gas_type	AC	rain	sun	refill liters	refill gas
0	False	False	False	False	False	True	False	False	False	False	False	False
1	False	False	False	False	False	True	False	False	False	False	True	True
2	False	False	False	False	False	True	False	False	False	False	True	True
3	False	False	False	False	False	True	False	False	False	False	True	True
4	False	False	False	False	False	True	False	False	False	False	True	True
5	False	False	False	False	False	True	False	False	False	False	True	True
6	False	False	False	False	False	True	False	False	False	False	True	True
7	False	False	False	False	False	True	False	False	False	False	True	True
8	False	False	False	False	False	True	False	False	False	False	True	True

Plotting the variables which consist of maximum no of null values.

In [4]:

```
null_values=df.isnull().sum().sort_values(ascending=False)
ax=sns.barplot(null_values.index,null_values.values)
ax.set_xticklabels(ax.get_xticklabels(),rotation=90)
import matplotlib.pyplot as plt
plt.show()
```



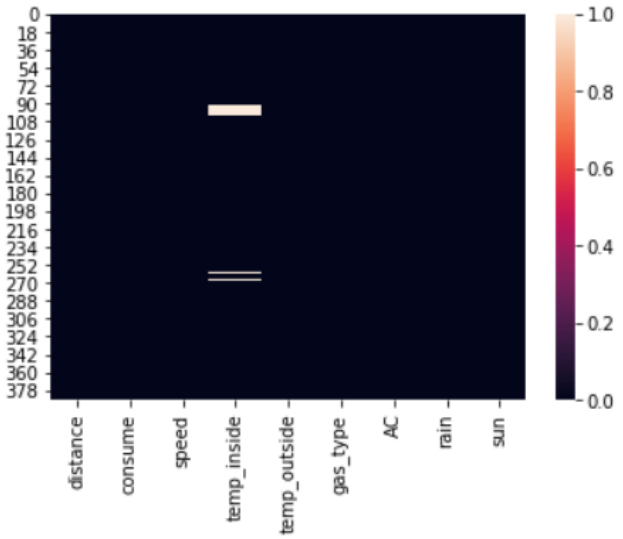
Activity 4: Removing null values

Refill gas, Refill liters, and specials columns are dropped using the drop() method from pandas. From the above image, we found these columns have many null values so it is dropped. Axis should be given as a parameter on the drop method.

In [5]:

```
df.drop(['refill gas','refill liters','specials'],axis=1,inplace=True)
sns.heatmap(df.isnull())
```

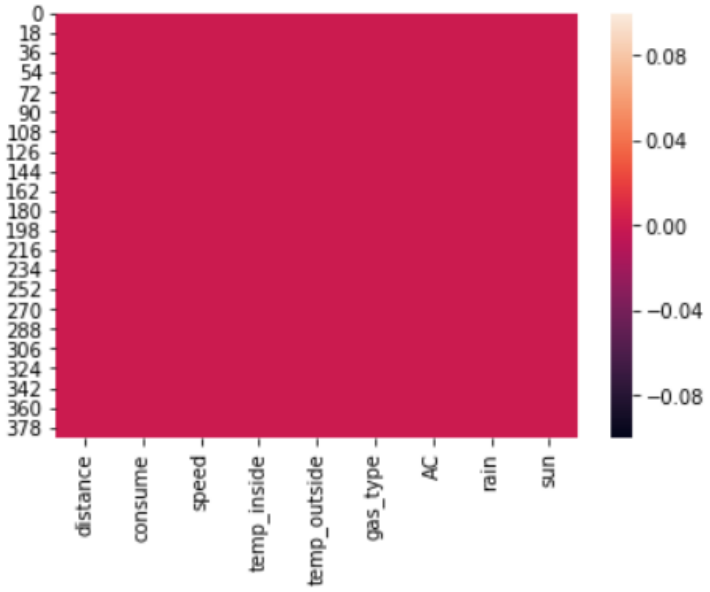
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x21743397400>



Activity 5:Handling null values

Here we are going to handle null values. From activity 3 we found we have null values in the 'temp_inside' column. So we are replacing the null value with its mean. Fillna() method from pandas is used to replace null values with their mean.

```
In [6]: 1 temp_inside_mean=np.mean(df['temp_inside'])

In [7]: 1 print(temp_inside_mean)

21.929521276595743

In [8]: 1 df['temp_inside'].fillna(temp_inside_mean,inplace=True)

In [9]: 1 sns.heatmap(df.isnull())

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x2174356b7f0>
```



Milestone 2: Model Building

Now our data is cleaned and it's time to build the model. We will be using the features to build the model by splitting them into dependent and independent variables.

Activity 1: Separating Independent and Dependent Values

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed.

```
In [10]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
l=LinearRegression()
```

```
In [11]:
x=df.drop(['consume','gas_type'],axis=1)
```

```
In [12]:
y=df['consume']
```

```
In [13]:
x.columns
```

```
Out[13]: Index(['distance', 'speed', 'temp_inside', 'temp_outside', 'AC', 'rain',
       'sun'],
      dtype='object')
```

```
In [14]:
x=x.values
y=y.values
```

Activity 2: Splitting Data into Train and Test

For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

For deep understanding refer this link: https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/

```
In [15]:
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

Activity 3: Applying Linear Regression

Now we are going to create our model with linear regression. As an initial step we have to initialize the linear model. Then train the model with fit() method. Now our

model is trained and to test the model predict() method is used. To find the loss of linear regression model mean_squared_error and mean_absolute_error are used.

Link: https://www.geeksforgeeks.org/ml-linear-regression/#:~:text=Linear%20Regression%20is%20a%20machine,relationship%20between%20variables%20and%20forecasting.

In [16]:
```
1 l.fit(x_train,y_train)
```

Out[16]:
```
LinearRegression()
```

In [17]:
```
1 x_train.shape
```

Out[17]:
```
(271, 7)
```

In [18]:
```
1 y_pred=l.predict(x_test)
```

In [19]:
```
1 print(l.coef_,l.intercept_)
```

```
[ 0.00523674 -0.02371772 -0.14711979 -0.03724498  0.41456804  0.61676684
 -0.06407861] 9.389308142257136
```

In [20]:
```
1 from sklearn import metrics
2 print(metrics.mean_squared_error(y_test,y_pred))
3 print(metrics.mean_absolute_error(y_test,y_pred))
4 print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
0.7424532609047081
0.6635761182069623
0.8616572757800564
```

In [21]:
```
1 dum1=pd.get_dummies(df['gas_type'])
2 print(dum1)
```

In [22]:
```
1 df=pd.concat([df,dum1],axis=1)
```

In [23]:
```
1 df.drop('gas_type',axis=1,inplace=True)
```

In [24]:
```
1 x1=df.drop('consume',axis=1)
```

In [25]:
```
1 y1=df['consume']
```

In [26]:
```
x1.columns
```

Out[26]:
```
Index(['distance', 'speed', 'temp_inside', 'temp_outside', 'AC', 'rain', 'sun',
       'E10', 'SP98'],
      dtype='object')
```

In [27]:
```
x1=x1.values
y1=y1.values
```

In [28]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
l=LinearRegression()
x_train,x_test,y_train,y_test=train_test_split(x1,y1,test_size=0.3,random_state=42)

```

In [29]:
```
l.fit(x_train,y_train)
```

Out[29]:
```
LinearRegression()
```

In [30]:
```
y_pred_1=l.predict(x_test)
print(y_pred_1)
```

```
[4.80398179 5.24631572 5.16373706 5.23299719 4.52776021 5.99062392
 5.73193936 5.23198354 5.8898096  4.94684204 4.0800537  4.78422755
 6.55357901 4.50083061 5.1268724  5.24267179 5.61167026 5.14823973
 5.48324723 5.36437201 4.13422549 5.30350959 4.94565881 5.23290799
 4.88631664 4.79418748 4.55506668 4.28205093 5.10144732 3.90735262
 4.97478302 5.29391251 4.75042548 4.56699402 5.53113778 5.02945576
 4.6453334  4.03415275 5.10287619 6.16080817 4.47545803 5.28255966
 5.37539962 4.41278157 4.69332325 4.39387259 5.10382269 5.1927726
 4.95992397 4.98995489 4.87121094 5.4268889  5.44648531 5.28120341
 4.61905757 4.90286809 6.70123899 5.3534319  4.71689758 4.78621524
 5.50574979 4.9290579  4.55311849 4.81518093 4.36022913 4.75672285
 5.55769604 4.34876836 4.82767226 4.91585314 4.28138845 4.6582407
 5.19170002 4.97280779 5.18528042 4.79819291 5.32165909 5.10687874
 5.38921307 5.15592614 5.26829591 5.45539801 4.47960294 5.3509791
 5.71243061 4.42243076 5.53113325 5.74565111 5.1678087  4.57634151
 4.81978083 4.50656632 5.10161474 3.96317992 4.30111744 5.47781482
 5.05321366 4.74406453 5.16373706 5.2337835  5.08221941 3.81421222
 4.58755104 4.49417409 5.39720411 4.50237128 4.34387901 4.53984859
 6.50203043 5.78353682 4.7085772  5.13955998 6.21742698 4.85512648
 4.7551128  5.46302901 4.8442509 ]
```

In [31]:
```
from sklearn import metrics
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred_1)))
```

```
0.864693406954018
```

In [32]:
```
x_train.shape
```

Out[32]:
```
(271, 9)
```

To save the model dump() method from joblib package is used.

```
In [32]:  1 x_train.shape
Out[32]: (271, 9)

In [33]:  1 x_train[0]
Out[33]: array([12.3, 62. , 21.5,  6. ,  0. ,  0. ,  0. ,  1. ,  0. ])

In [34]:  1 import joblib
          2 joblib.dump(l,'model.save')
Out[34]: ['model.save']
```

Milestone 3: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks

- Building HTML Pages
- Building serverside script

Activity 1: Build the python Flask app

Load the saved model. 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.

```
1 from flask import Flask, request,render_template
2 import joblib
3 app = Flask(__name__)
4 model = joblib.load("model.save")
5
6
7 app = Flask(__name__)
8
```

Load the home page

```
9 @app.route('/')
10 def predict():
11     return render_template('Manual_predict.html')
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with Manual_predict.html function. Hence, when the home page of the web server is opened in browser, the html page will be

rendered. Whenever you enter the values from the predict html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/y_predict',methods=['POST'])
def y_predict():
    x_test = [[float(x) for x in request.form.values()]]
    print('actual',x_test)
    pred = model.predict(x_test)

    return render_template('Manual_predict.html', \
                           prediction_text=('Car fuel Consumption(L/100km) \
                                            : ',pred[0]))


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Activity 2:Build An HTML Page

We Build an HTML page to take the values from the user in a form and upon clicking on the predict button we get the fuel consumption predicted.

```
<html>
<head>
<title>
    Prediction
</title>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
<style>
        * {
            box-sizing: border-box;
        }

        body {
            font-family: 'Montserrat' ;
        }

        .header {
            top:0;
            margin:0px;
            left: 0px;
            right: 0px;
            position: fixed;
            background-color: black;
            color: white;
            box-shadow: 0px 8px 4px grey;
            overflow: hidden;
            padding: 15px;
            font-size: 2vw;
            width: 100%;
            text-align: left;
            padding-left: 100px;
            opacity:0.9;
        }
        .header_text{
            font-size:40px;
            text-align:center;
        }
        .content{
        margin-top:100px;
        }
```

```
        .text{
            font-size:20px;
            margin-top:10px;
            text-align:center;
        }
        input[type=number], select {
  width: 50%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

input[type=submit] {
  width: 50%;
  background-color: #000000;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

input[type=submit]:hover {
  background-color: #5d6568;
  color:#ffffff;
  border-color:black;
}
form{
margin-top:20px;
}
.result{
color:black;
margin-top:30px;
margin-bottom:20px;
font-size:25px;
color:red;
}
```

```
</style>
</head>
<body align=center>
<div class="header">
        <div>Car Fuel Consumption </div>
</div>
<div class="content">
<div class="header_text">Car Fuel Consumption Prediction</div>
<div class="text">Fill in and below details to predict the consumption depending on the gas type.</div>
<div class="result">
  {{ prediction_text }}
  </div>
<form action="{{ url_for('y_predict') }}" method="POST">
    <input type="number" step= "any" id="distance" name="distance" placeholder="distance(km)">
    <input type="number" id="speed" name="speed" placeholder="speed(km/h)">
    <input type="number"  id="temp_inside" name="temp_insidet" placeholder="temp_inside(°C)">
    <input type="number"  id="temp_outside" name="temp_outside" placeholder="temp_outside(°C)">
    <input type="number"  id="AC" name="AC" placeholder="AC">
    <input type="number"  id="rain" name="rain" placeholder="rain">
    <input type="number"  id="sun" name="sun" placeholder="sun">
    <input type="number"  id="E10" name="E10" placeholder="E10">
    <input type="number"  id="SP98" name="SP98" placeholder="SP98">

    <input type="submit" value="Submit">
  </form>

</div>
</body>
</html>
```

Activity 3 : Run The Application

.

Step 1: Open anaconda prompt go to project folder and in that go to flask folder and run the python file by using the command "**python app.py**"

```
(base) D:\SmartBridge\MLAI\ML_Projects\guided projects feb\Car Fuel Consumption\Flask>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 301-111-576
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Output:

Car Fuel Consumption

Car Fuel Consumption Prediction

Fill in and below details to predict the consumption depending on the gas type.

distance(km)

speed(km/h)

temp_inside(°C)

temp_outside(°C)

AC

rain

sun

E10

SP98

Car Fuel Consumption

Car Fuel Consumption Prediction

Fill in and below details to predict the consumption depending on the gas type.

12.2

62

21

6

0

0

0

1

0

Submit

Car Fuel Consumption

Car Fuel Consumption Prediction

Fill in and below details to predict the consumption depending on the gas type.

('Car fuel Consumption(L/100km) : ', 4.707891280435151)

Milestone 4: Train The Model On IBM

In this milestone, you will learn how to build a Machine Learning Model and deploy it on the IBM Cloud.

Activity 1: Register For IBM Cloud

- Please click here to register for IBM
- Please click here to log in to IBM Account

Watch the below video to register and login into your IBM account

https://youtu.be/QuTDhYeJh0k

Activity 2: Train The ML Model on IBM

Watch the below video to train the Machine learning model on IBM Watson

https://youtu.be/TysuP3KgSzc

Activity 3:Integrate with Flask With Scoring End Point

Watch the below video to integrate the scoring endpoint to the flask

https://youtu.be/ST1ZYLmYw2U