TRIP BASED MODELLING OF FUEL CONSUMPTION IN MODERN FLEET VECHILES USING IBM WATSON

MINI PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

MATETI NIKHIL	**19UK1A05E1**
DANABOINA ASHWINI	**19UK1A05K5**
CHITTIREDDY SREEJA	**19UK1A05F6**
UPPULA THARUN	**19UK1A05G8**

Under the esteemed guidance of

Mr. MAHESH

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
(Affiliated to JNTUH, Hyderabad)
Bollikunta, Warangal – 506005

2019– 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING VAAGDEVI ENGINEERING COLLEGE

BOLLIKUNTA, WARANGAL – 506005

2019 – 2023

CERTIFICATE OF COMPLETION

MINI PROJECT

This is to certify that the Mini project entitled **"TRIP BASED MODELLING OF FUEL CONSUMPTION IN MODERN FLEET VEHCLES USING IBM WATSON"**is being submitted by **M.NIKHIL(H.NO:19UK1A05E1),D.ASHWINI(19UK1A05K5),CH.SREEJA (H.NO:19UK1A05F6),U.THARUN(H.NO:19UK1A05G8)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2022-23, is a record of work carried out by them under the guidance and supervision.

Project Guide

Mr. E.Mahesh
(Assistant Professor)

Head of the Department

Dr. R. Naveen Kumar
(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr.P.PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this Mini Project in the institute.

We extend our heartfelt thanks to **Dr.R.NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the Mini Project.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the Mini Project and for their support in completing the Mini Project.

We express heartfelt thanks to the guide, **Mr.E.Mahesh** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this Mini Project.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

M.NIKHIL **(19UK1A05E1)**

D.ASHWINI **(19UK1A05K5)**

CH.SREEJA **(19UK1A05F6)**

U.THARUN **(19UK1A05G8)**

TABLE OF CONTENTS: -

Trip Based Modelling of Fuel Consumption in modern Fleet Vehicles using IBM Watson

1.INTRODUCTION

1.1OVERVIEW

The fuel efficiency of fleet vehicles can be beneficial not only for the automotive and transportation industry but also for a country's economy and the global environment . The cost of fuel consumed contributes to approximately 30% of a fleet vehicles life cycle cost. Reduction in fuel consumption by just a few percent can significantly reduce costs for the transportation industry . The effective and accurate estimation of fuel consumption (fuel consumed in L/km) can help to analyze emissions as well as prevent fuel-related fraud. As per Environmental Protection Agency (EPA) reports, 28% of total greenhouse gas emissions come from transportation (heavy-duty vehicles and passenger cars) . The United States Environmental Protection Agency (US EPA) has introduced Corporate Average Fuel Economy (CAFÉ) standards enforcing automotive manufacturers to be compliant with standards to regulate fuel consumption . US EPA regulations enacting fuel economy improvements in freights released in 2016 target truck fuel efficiency, which is predicted to improve by 11–14% by 2021 . Most states have now mandated that truck fleets update their vehicle inventory with modern vehicles due to air quality regulation.

1.2 PURPOSE

Several studies have been presented in the past for evaluating the fuel efficiency of fleet vehicles using simulation-based models and data-driven models. A simulation model was developed based on engine capacity, fuel injection, fuel specification, aerodynamic drag, grade resistance, rolling resistance, and atmospheric conditions, with simulated dynamic driving conditions to predict fuel consumption . A statistical model which is fast and simple compared to the physical load-based approach was developed to predict vehicle emissions and fuel consumption . The impact of road infrastructure , traffic conditions , drivers' behaviour , weather conditions , and the ambient temperature on fuel consumption were studied, and it was determined that fuel consumption can be reduced by 10% with eco-driving influences. The era of big data has enabled the modeling of huge volumes of data for companies to reduce emissions and fuel consumption. Machine learning techniques such as support vector machine (SVM) , random forest (RF) , and IBM watson are widely applied to turn data into meaningful insights and solve complex problems.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

In recent years, deep learning has been used in various applications including the classification of ship targets in inland waterways for enhancing intelligent transport systems. Various researchers introduced different classification algorithms, but they still face the problems of low accuracy and misclassification of other target objects. Hence, there is still a need to do more research on solving the above problems to prevent collisions in inland waterways.

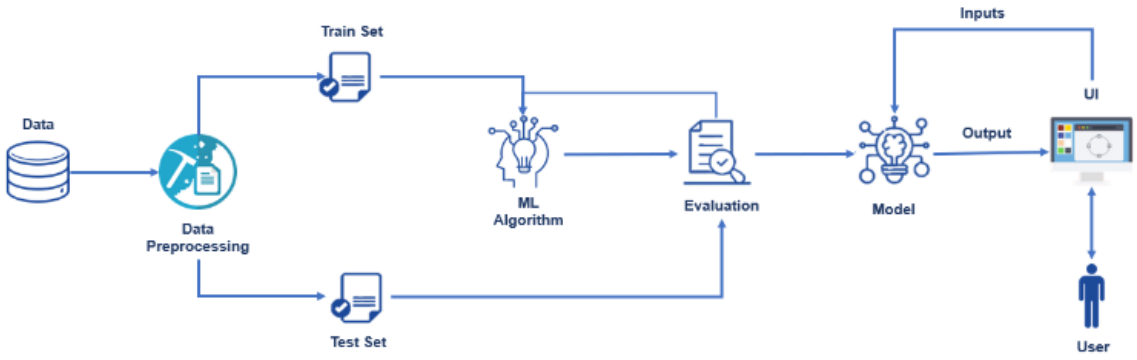
2.2 PROPOSED SYSTEM

In order to solve the problems for the accuracy of the classification system, we proposed a new classification model. First, based on the pretrained models, the models were fine-tuned with the public dataset we used.

Based on their performance, the best model was selected in order to further adjust the performance for high accuracy in classifying ships in inland river waterways. After selecting the best model, the model was adjusted, and classification was conducted based on the modification of the network.

3.THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE AND SOFTWARE DESIGNING

Software requirements:

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum , and first released on February 20, 1991. Its high-level built in data structures, combined with dynamic typing and dynamic binding , make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

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 mac OS.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.

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Spyder

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. Spyder is extensible with first-party and third party plugins includes support for interactive tools for data inspection and embeds Python specific code. Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

Flask

Web framework used for building. It is a web application framework written in python which will be running in local browser with a user interface. In this application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the user.

Hardware Requirements:

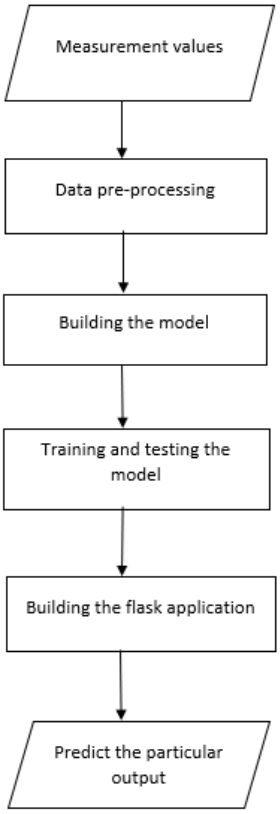
o Operating system: window7 and above with 64bit
o Processor Type -Intel Core i3-3220
o RAM: 4Gb and above
o Hard disk: min 100gb

4.EXPERIMENTALINVESTIGATION

The text data need to be organized before proceeding with the project. The original dataset has a single

folder. We will be using the measurements.csv file to fetch the text data of training data. The data need to be unique and all fields need to be filled. The dataset images are to be pre-processed before giving to the model. We will create a function that uses the pre-trained model for predicting custom outputs. Then we have to test and train the model. After the model is build, we will be integrating I to a web application.

5.FLOWCHART



6.ADVANTAGES

- Data modeling can help to identify the trend in instantaneous fuel consumption and to calculate the total fuel consumed by the vehicle for each trip, which can further help in diagnosing vehicle performance in the case of abnormalities.
- The effective and accurate estimation of fuel consumption (fuel consumed in L/km) can help to analyze emissions as well as prevent fuel-related fraud

7.CONCLUSION

In conclusion, the study demonstrates the modeling of fuel consumption in modern fleet vehicles with an IBM Watson using very few technical parameters. An attempt was made to develop a model using very few parameters collected under different conditions. Data from modern fleet vehicles with the same make and model, driven by different persons on various routes under different external conditions, were used for training the IBM Watson.

8.FUTURE SCOPE

- In future works, the proposed method will be improved in order to classify the people in different countries with extra features using more advanced technology.

9.BIBILOGRAPHY

- https://researchrepository.wvu.edu/faculty_publications/3072/?utm_source=researchrepository.wvu.edu%2Ffaculty_publications%2F3072&utm_medium=PDF&utm_campaign=PDFCoverPages

10.APPENDIX

SOURCE CODE:

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

Read The Datasets

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.

```
In [2]: df=pd.read_csv("measurements.csv")

        df.head()
```

Out[2]:

	distance	consume	speed	temp_inside	temp_outside	specials	gas_type	AC	rain	sun	refill liters	refill gas
0	28	5	26	21,5	12	NaN	E10	0	0	0	45	E10
1	12	4,2	30	21,5	13	NaN	E10	0	0	0	NaN	NaN
2	11,2	5,5	38	21,5	15	NaN	E10	0	0	0	NaN	NaN
3	12,9	3,9	36	21,5	14	NaN	E10	0	0	0	NaN	NaN
4	18,5	4,5	46	21,5	15	NaN	E10	0	0	0	NaN	NaN

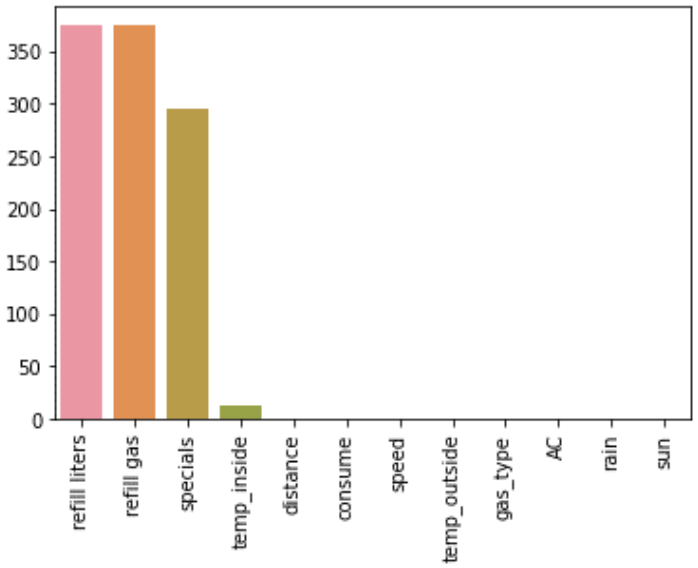
Check Null Values

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 [2]: df.shape
Out[2]: (388, 12)

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388 entries, 0 to 387
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   distance       388 non-null    object
 1   consume        388 non-null    object
 2   speed          388 non-null    int64
 3   temp_inside    376 non-null    object
 4   temp_outside   388 non-null    int64
 5   specials       93 non-null     object
 6   gas_type       388 non-null    object
 7   AC             388 non-null    int64
 8   rain           388 non-null    int64
 9   sun            388 non-null    int64
 10  refill liters  13 non-null     object
 11  refill gas     13 non-null     object
dtypes: int64(5), object(7)
memory usage: 36.5+ KB

In [4]: df.isnull().sum()
Out[4]: distance          0
consume           0
speed             0
temp_inside      12
temp_outside      0
specials        295
gas_type          0
AC                0
rain              0
sun               0
refill liters   375
refill gas      375
dtype: int64
```

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

```
In [5]:
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()
```

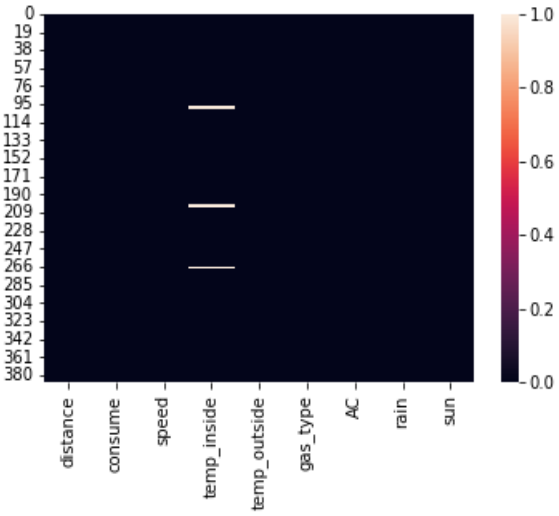


Removing Null Values

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 [6]:
df.drop(['refill gas','refill liters','specials'],axis=1,inplace=True)
sns.heatmap(df.isnull())
```

```
Out[6]: <AxesSubplot:>
```



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 [7]:
```
df['temp_inside'] = df['temp_inside'].astype(str).str.replace(',','.')
df['distance'] = df['distance'].astype(str).str.replace(',','.')
df['consume'] = df['consume'].astype(str).str.replace(',','.')
```

In [8]:
```
df.head()
```

Out[8]:

	distance	consume	speed	temp_inside	temp_outside	gas_type	AC	rain	sun
0	28	5	26	21.5	12	E10	0	0	0
1	12	4.2	30	21.5	13	E10	0	0	0
2	11.2	5.5	38	21.5	15	E10	0	0	0
3	12.9	3.9	36	21.5	14	E10	0	0	0
4	18.5	4.5	46	21.5	15	E10	0	0	0

In [9]:
```
df['temp_inside'].value_counts()
```

Out[9]:
```
21.5    133
22      102
22.5     59
20       25
21       13
23       13
nan      12
25       12
24.5      7
20.5      4
24        3
23.5      2
25.5      2
19        1
Name: temp_inside, dtype: int64
```

In [10]:
```
df.describe()
```

Out[10]:

	speed	temp_outside	AC	rain	sun
count	388.000000	388.000000	388.000000	388.000000	388.000000
mean	41.927835	11.358247	0.077320	0.123711	0.082474
std	13.598524	6.991542	0.267443	0.329677	0.275441
min	14.000000	-5.000000	0.000000	0.000000	0.000000
25%	32.750000	7.000000	0.000000	0.000000	0.000000
50%	40.500000	10.000000	0.000000	0.000000	0.000000
75%	50.000000	16.000000	0.000000	0.000000	0.000000
max	90.000000	31.000000	1.000000	1.000000	1.000000

```
In [11]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388 entries, 0 to 387
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   distance      388 non-null    object
 1   consume       388 non-null    object
 2   speed         388 non-null    int64
 3   temp_inside   388 non-null    object
 4   temp_outside  388 non-null    int64
 5   gas_type      388 non-null    object
 6   AC            388 non-null    int64
 7   rain          388 non-null    int64
 8   sun           388 non-null    int64
dtypes: int64(5), object(4)
memory usage: 27.4+ KB
```

```
In [12]: df['temp_inside'] = df['temp_inside'].astype('float')
```

```
In [13]: temp_inside_mean=np.mean(df['temp_inside'])
```

```
In [14]: print(temp_inside_mean)
```

```
21.929521276595743
```

```
In [15]: df['temp_inside'].fillna(temp_inside_mean,inplace=True)
```

```
In [16]: df.isnull().sum()
```

```
Out[16]: distance        0
         consume         0
         speed           0
         temp_inside     0
         temp_outside    0
         gas_type        0
         AC              0
         rain            0
         sun             0
         dtype: int64
```

Model building:

Seperating Independent And Dependent Variables

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

```
In [18]: y=df['consume']
```

```
In [19]: y = y.astype('float')
```

```
In [20]: x.head()
```

Out[20]:

	distance	speed	temp_inside	temp_outside	AC	rain	sun
0	28	26	21.5	12	0	0	0
1	12	30	21.5	13	0	0	0
2	11.2	38	21.5	15	0	0	0
3	12.9	36	21.5	14	0	0	0
4	18.5	46	21.5	15	0	0	0

```
In [21]: from sklearn.preprocessing import scale
         X_scaled=pd.DataFrame (scale(x), columns=x.columns)
         X_scaled.head()
```

Out[21]:

	distance	speed	temp_inside	temp_outside	AC	rain	sun
0	0.368714	-1.172804	-0.432382	0.091908	-0.28948	-0.375735	-0.299813
1	-0.338044	-0.878274	-0.432382	0.235123	-0.28948	-0.375735	-0.299813
2	-0.373381	-0.289216	-0.432382	0.521552	-0.28948	-0.375735	-0.299813
3	-0.298288	-0.436480	-0.432382	0.378338	-0.28948	-0.375735	-0.299813
4	-0.050923	0.299843	-0.432382	0.521552	-0.28948	-0.375735	-0.299813

```
In [22]: y.dtype
```

```
Out[22]: dtype('float64')
```

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.

```
In [23]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.2,random_state=1)
```

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.

In [24]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
l=LinearRegression()
```

In [25]:
```
l.fit(X_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
X_train.shape
```

Out[26]:
```
(310, 7)
```

In [27]:
```
y_pred=l.predict(X_test)
```

In [28]:
```
print(l.coef_,l.intercept_)
```

```
[ 0.06098318 -0.25848807 -0.08457254 -0.29910799  0.14843329  0.16317931
 -0.03078575] 4.938915956072435
```

In [29]:
```
from sklearn import metrics
print(metrics.mean_squared_error(y_test,y_pred))
print(metrics.mean_absolute_error(y_test,y_pred))
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
0.5465365506923656
0.5951773803147186
0.73928110397356
```

In [30]:
```
import joblib
joblib.dump(l,'model.save')
```

Out[30]:
```
['model.save']
```

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>
```

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.

In [34]:
```
from flask import Flask, request,render_template
import joblib
app = Flask(__name__)
model = joblib.load("model.save")
```

In [35]:
```
app = Flask(__name__)
```

Load the home page

In [36]:
```
@app.route('/')
def predict():
    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.

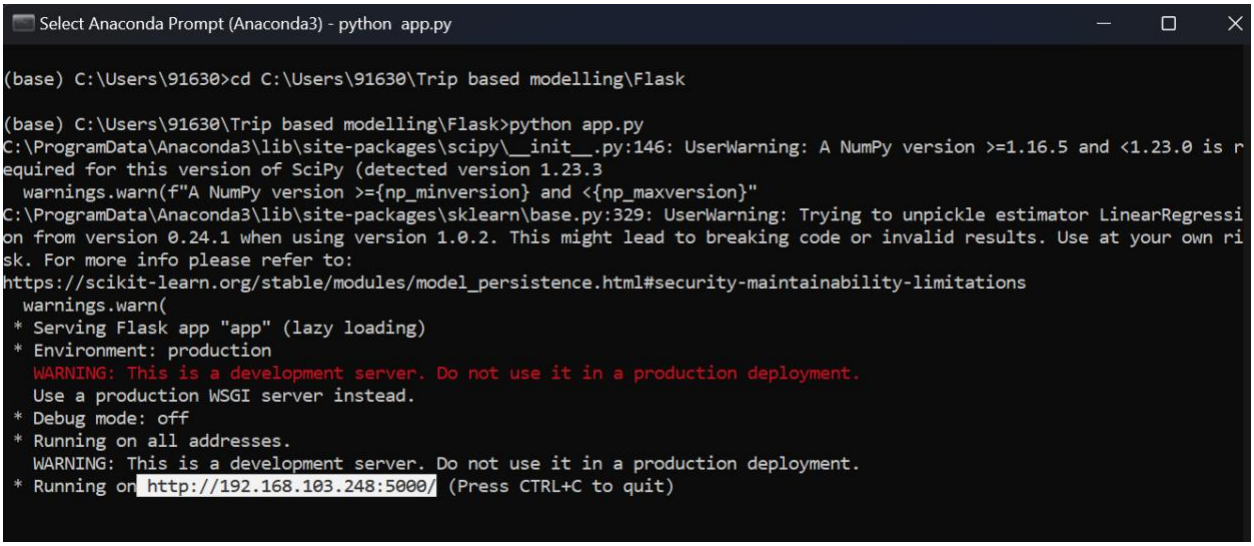
```
In [*]: @app.route('/y_predict',methods=['POST'])
        def y_predict():
            x_test = [[float(x) for x in request.form.values()]]
            print('actual',x_test)
            pred = model.predict(x_test)

            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=False)
```

```
 * Serving Flask app "__main__" (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: off
```

```
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://192.168.103.248:5000/ (Press CTRL+C to quit)
```

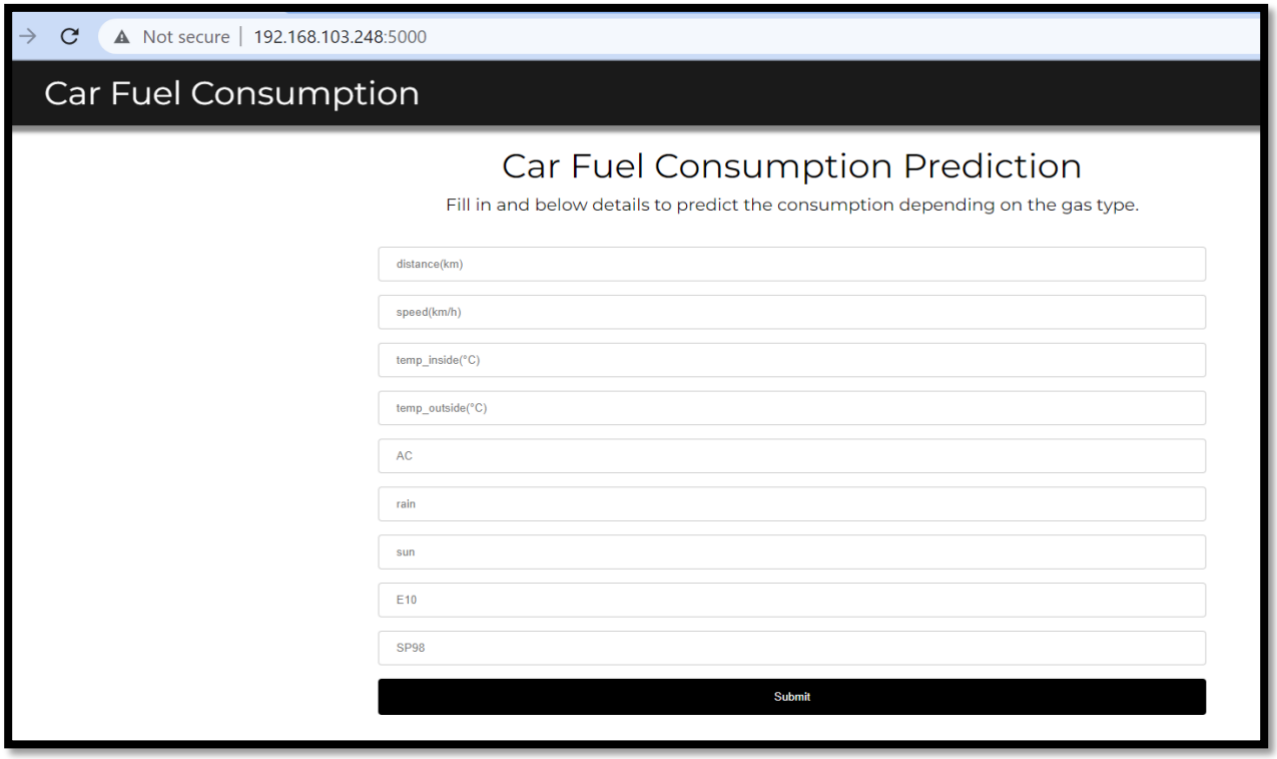
Run The Application:

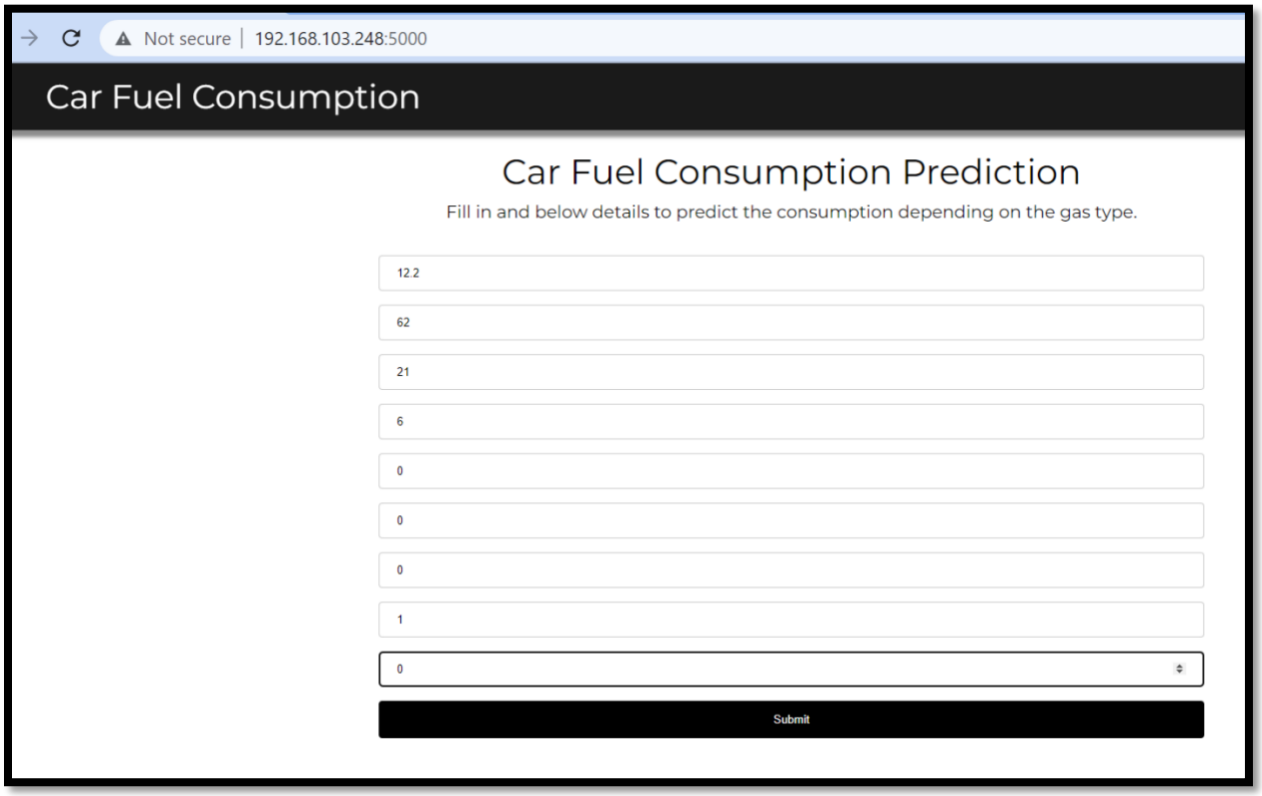
Step 1: Open anaconda prompt go to project folder and in that go to flask folder and run the python file by using the command "**python app.py**"

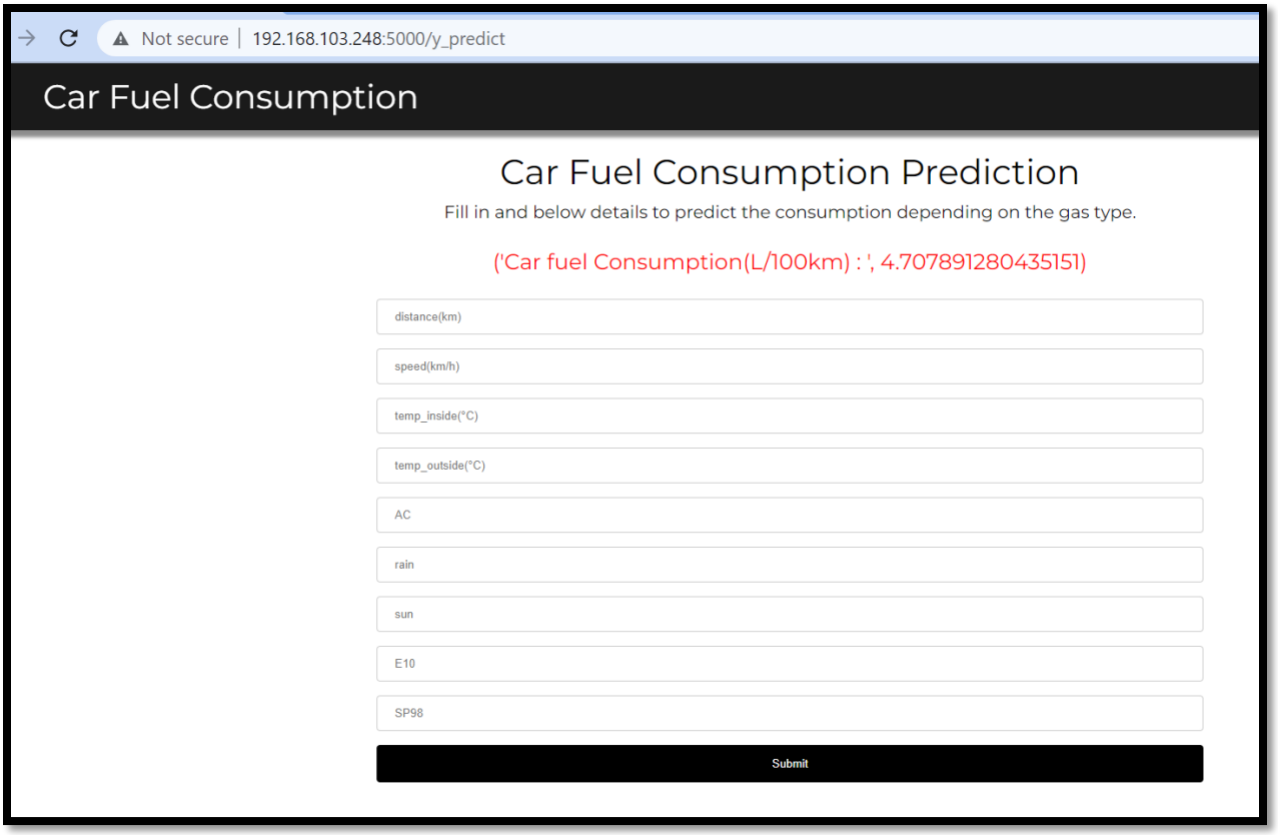


11. RESULT

OUTPUT:






Not secure | 192.168.103.248:5000/y_predict
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)
distance(km)
speed(km/h)
temp_inside(°C)
temp_outside(°C)
AC
rain
sun
E10
SP98
Submit