ABSTRACT

The transportation industry is one of the largest and most important industries in the modern economy. Organizations in this industry operate more than a thousand fleet vehicles on average. Thus it becomes harder for organizations to manage the expenditures of the vehicles. The goal of this project is to provide a platform for users like individuals and organizations, which allows them to predict the fuel consumption of vehicles using Machine Learning. The project consists of two prediction sections, one for single-sample prediction and the other for multi-sample prediction. This allows organizations to manage expenditures and prevent fraudulent activities easily. This project is also designed to be highly extensible, which allows developers to use these features in their applications. A sample Excel sheet is provided, on which the user can enter multiple vehicle details and upload them to find the fuel consumption of all the vehicles in the Excel sheet. Users can also view their prediction history. The project also allows users to download detailed reports of the vehicles, which enables more detailed insight into the performance of the vehicles. Therefore it becomes easy to manage the fuel expenses for any organization or individual thus making it a better platform to use.

TABLE OF CONTENTS: -

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 (heavyduty 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

Trip-based modeling of fuel consumption in modern fleet vehicles using machine learning addresses several existing problems:

1. Inefficient Fuel Use: Traditional methods may not capture the nuances of fuel consumption across different trips and driving conditions. This leads to inefficient fuel use and higher operational costs.
2. Lack of Real-time Insights: Fleet managers often lack real-time data and predictive insights, making it difficult to optimize routes and driving behaviors dynamically to save fuel.
3. Inaccurate Predictions: Existing models might not accurately predict fuel consumption due to insufficient consideration of factors like traffic conditions, driver behavior, vehicle load, and environmental conditions.
4. Scalability Issues: Manual data analysis and traditional modeling approaches can be cumbersome and do not scale well with large fleets, leading to less efficient management and decision-making processes.
5. Environmental Impact: Inefficient fuel consumption leads to higher emissions, contributing to environmental pollution and not meeting regulatory standards for emissions.
6. Maintenance and Wear: Inaccurate fuel consumption models can lead to poor maintenance scheduling, which affects vehicle health and longevity.

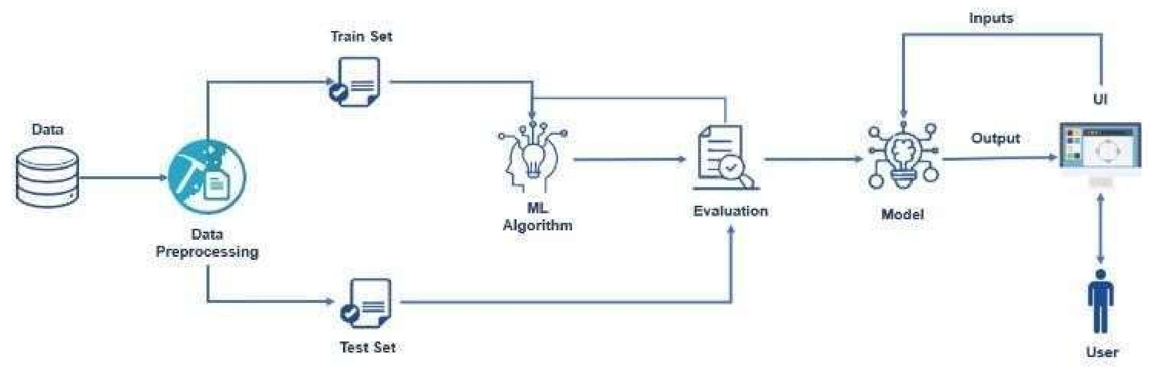
Using machine learning for trip-based fuel consumption modeling can provide more accurate predictions by analyzing vast amounts of data, considering multiple variables, and learning patterns that traditional methods may overlook. This leads to better fuel efficiency, cost savings, and a reduced environmental footprint.

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 bestmodel, the model was adjusted, and classification was conducted based on the modification of the network.

3.THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING AND HARDWARE

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 opensource, 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.

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.

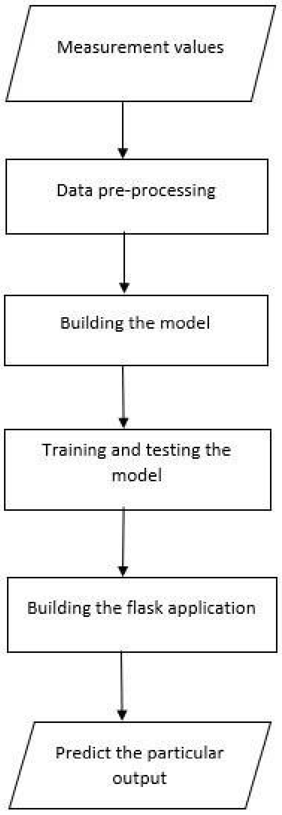
HARDWARE REQUIREMENTS:

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

4.EXPERIMENTAL INVESTIGATION

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


Measurement values
Data pre-processing
Building the model
Training and testing the model
Building the flask application
Predict the particular output

6.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. Machine learning models can analyze vast amounts of data from various sources (e.g., GPS, sensors, historical data) to predict fuel consumption more accurately than traditional methods.
2. These models can process data in real-time, providing fleet managers with immediate insights into fuel consumption, enabling prompt decision-making.
3. Machine learning models can be tailored to individual vehicles and driving behaviors, allowing for more precise fuel consumption predictions.
4. By identifying patterns in fuel consumption data, machine learning can help predict when a vehicle might need maintenance, reducing downtime and improving overall efficiency.
5. Advanced algorithms can optimize routes and driving behaviors to minimize fuel consumption, thus reducing operational costs and environmental impact.

DISADVANTAGES:

1. High-quality, extensive datasets are crucial for training effective machine learning models. Incomplete or poor-quality data can lead to inaccurate predictions.
2. Developing and maintaining machine learning models can be complex and require specialized knowledge and skills, potentially increasing operational costs.
3. Machine learning models, especially deep learning, can be resource-intensive, requiring significant computational power and storage.
4. The initial cost of implementing machine learning solutions can be high, including costs for hardware, software, and skilled personnel.
5. Machine learning models, particularly deep learning models, can be seen as "black boxes" where the decision-making process is not easily interpretable, making it challenging to understand how specific predictions are made.

7.CONCLUSION

In conclusion, trip-based modeling of fuel consumption in modern fleet vehicles using machine learning offers a powerful approach to optimize fuel efficiency and reduce operational costs. By analyzing various trip parameters and vehicle data, machine learning algorithms can predict fuel consumption with high accuracy. This method enables fleet managers to make informed decisions on route planning, vehicle maintenance, and driver behavior modifications. Consequently, it contributes to significant fuel savings and environmental benefits. Overall, the integration of machine learning in fuel consumption modeling is a promising advancement for the future of fleet management.

8.FUTURE SCOPE

1. 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.
2. Internet of Things (IoT): Integrating IoT devices to collect comprehensive data on various factors affecting fuel consumption.
3. Emissions Reduction: Developing models that not only predict fuel consumption but also optimize it to reduce emissions, contributing to environmental sustainability.
4. Predictive Maintenance: Using fuel consumption data to predict and prevent maintenance issues, ensuring vehicles operate efficiently.
5. Dashboards and Visualizations: Creating intuitive dashboards and visualization tools to help fleet managers understand and act on fuel consumption data.

9.BIBILOGRAPHY

1. Lin, W., & Qin, R. (2018). Vehicle Fuel Consumption Prediction Based on Machine Learning Algorithms. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 4915-4917). IEEE.

2. Li, Y., Zhang, J., Yu, B., & Liu, X. (2019). Research on Predictive Model of Fuel Consumption for Vehicle Based on Machine Learning. In 2019 5th International Conference on Control, Automation and Robotics (ICCAR) (pp. 321-324). IEEE.

3. Yu, J., Ye, Z., & Yu, C. (2017). A hybrid fuel consumption model for commercial vehicles. Journal of Traffic and Transportation Engineering (English Edition), 4(3), 252-262.

4. Liao, Y., Xu, Q., Wang, S., Li, J., & Liu, C. (2019). Modeling and prediction of vehicle fuel consumption based on machine learning. Journal of Cleaner Production, 229, 597606.

5. Kim, K., Lee, J., & Jo, J. (2018). Prediction of Fuel Consumption for Vehicles Using Machine Learning. Energies, 11(11), 3162.

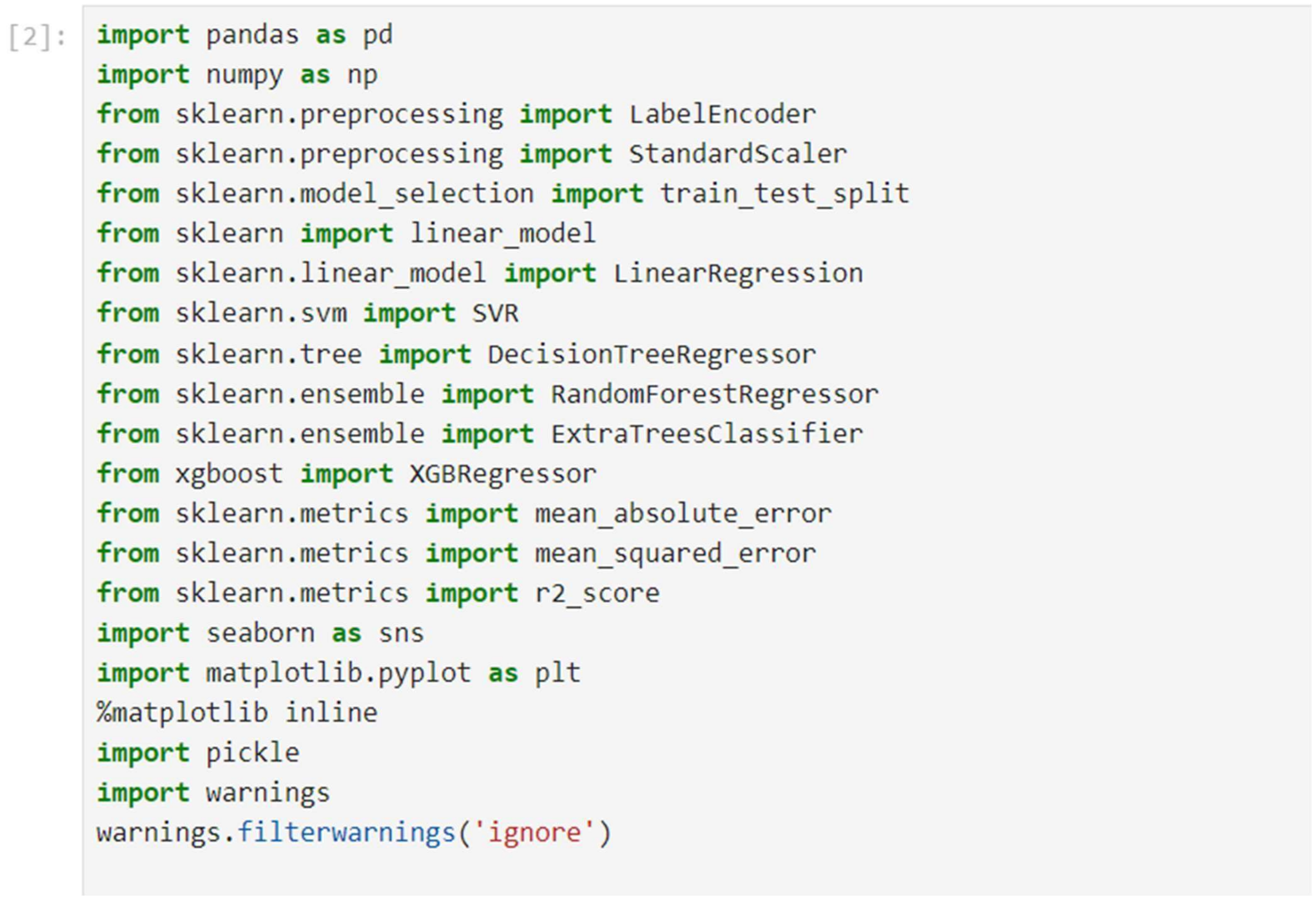
6. Khatami, A., Khatami, S., Akyel, C., & Balaban, E. (2019). A data-driven model for fuel consumption prediction in passenger cars. Energies, 12(15), 2961.

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.



```
[2]: import pandas as pd
     import numpy as np
     from sklearn.preprocessing import LabelEncoder
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn import linear_model
     from sklearn.linear_model import LinearRegression
     from sklearn.svm import SVR
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.ensemble import ExtraTreesClassifier
     from xgboost import XGBRegressor
     from sklearn.metrics import mean_absolute_error
     from sklearn.metrics import mean_squared_error
     from sklearn.metrics import r2_score
     import seaborn as sns
     import matplotlib.pyplot as plt
     %matplotlib inline
     import pickle
     import warnings
     warnings.filterwarnings('ignore')
```

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.

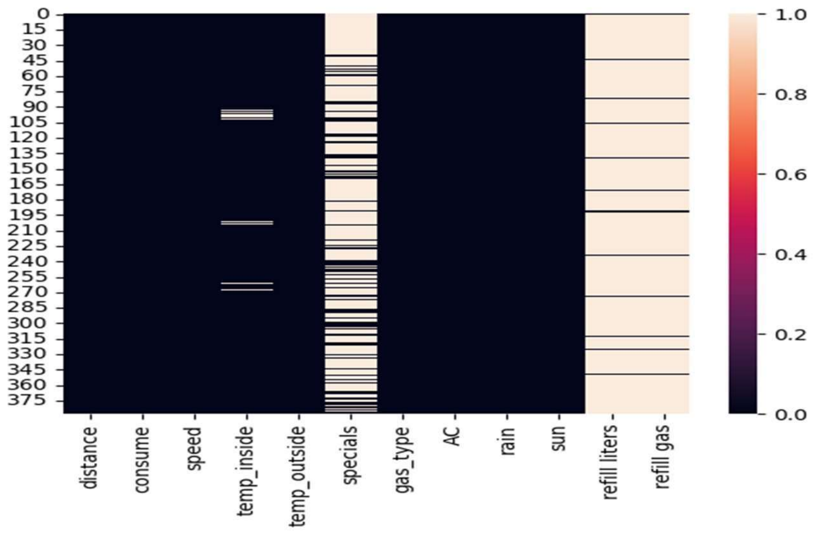
In pandas, we have a function called read_excel() to read the dataset. As a parameter, we have to give the directory of xlsx file.

```
[4]: df = pd.read_excel(r'C:\Users\akhil\OneDrive\Desktop\miniproject523\measurements2.xlsx')
     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
```

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.

```
[6]: import seaborn as sns
     sns.heatmap(df.isnull())

[6]: <Axes: >
```



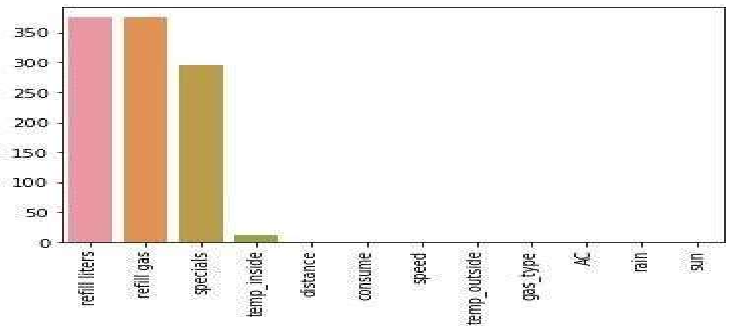
```
In [2]: df.shape

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
```

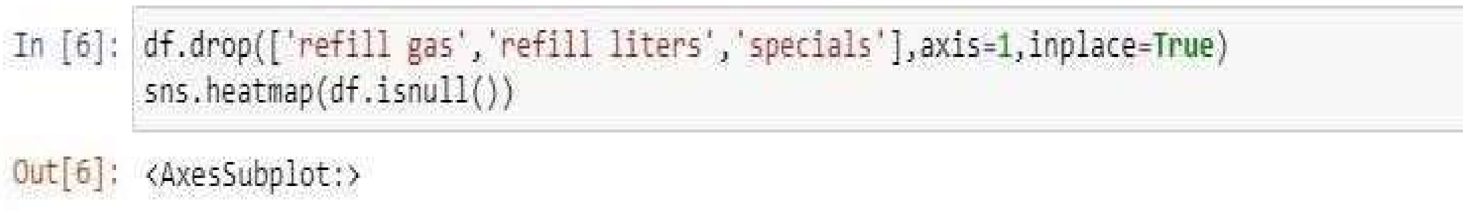
• Plotting the variables which consist of maximum no of null values.

```
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()
```

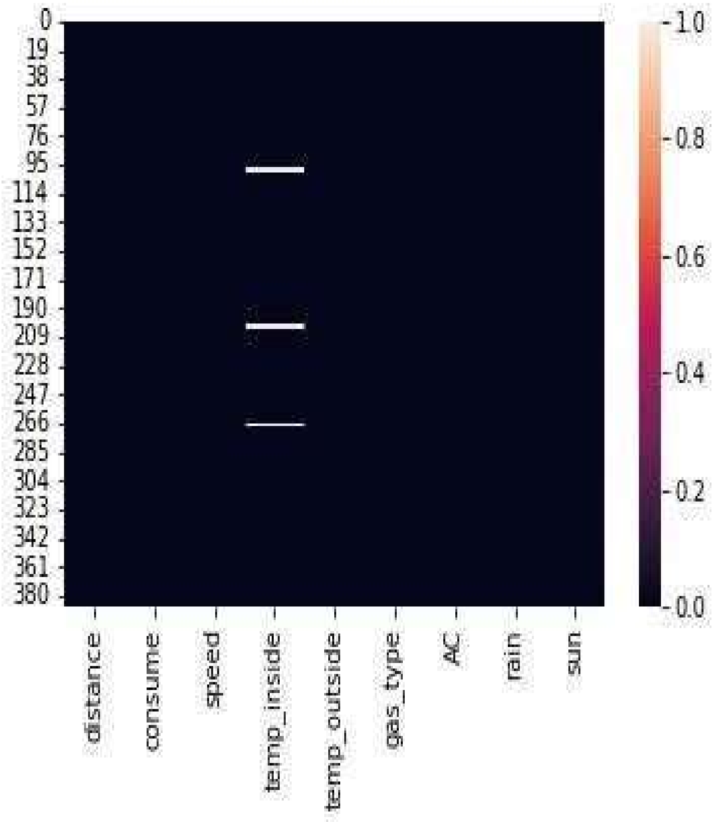


Removing Null Values

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.

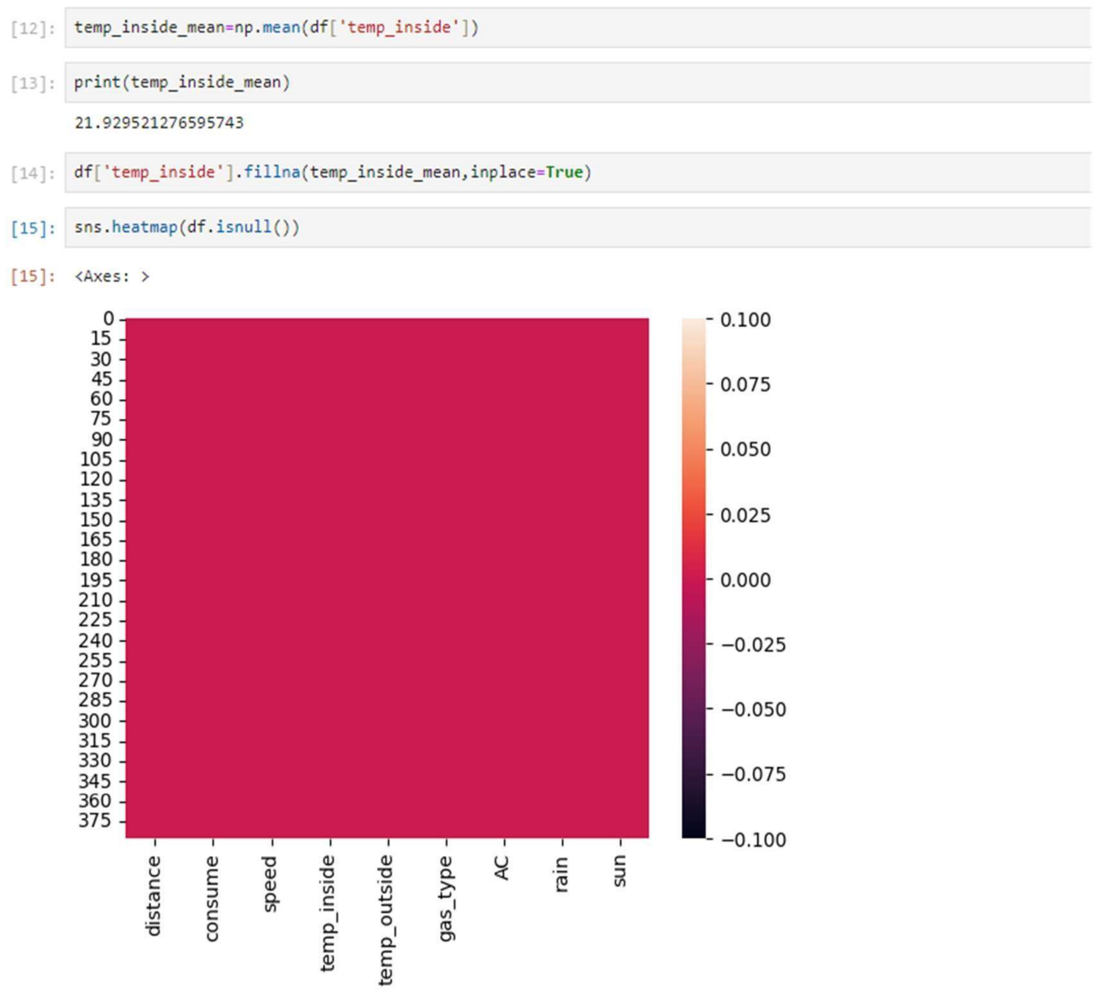


```
In [6]: df.drop(['refill gas','refill liters','specials'],axis=1,inplace=True)
        sns.heatmap(df.isnull())

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

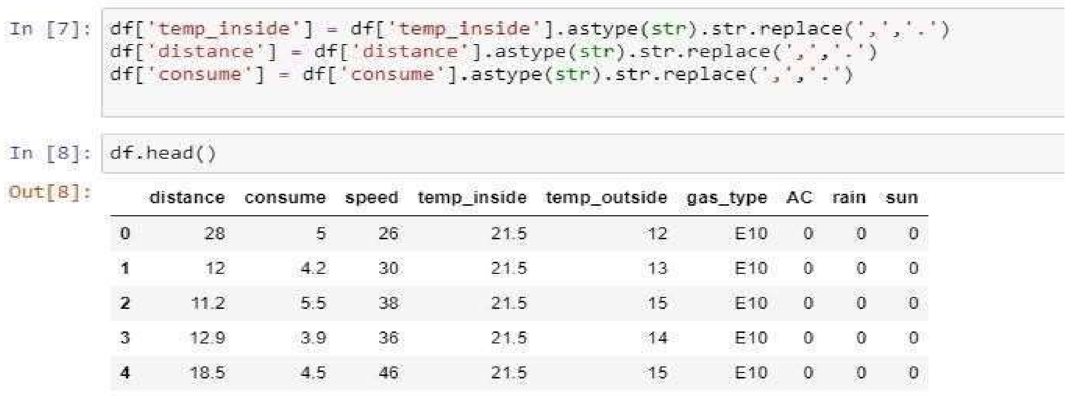


Handling Null Values

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. isnall() method from pandas is used to replace null values with their mean.



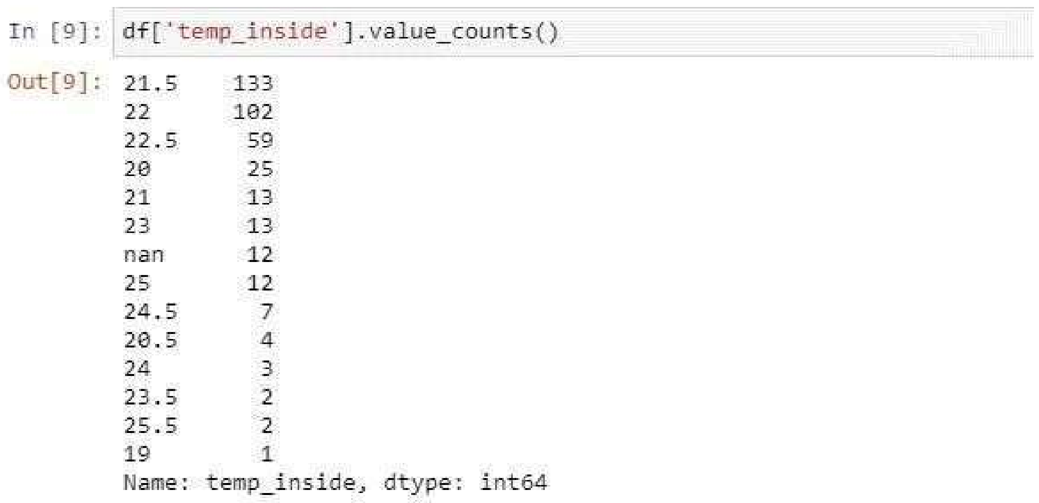
```
[12]: temp_inside_mean=np.mean(df['temp_inside'])

[13]: print(temp_inside_mean)

      21.929521276595743

[14]: df['temp_inside'].fillna(temp_inside_mean,inplace=True)

[15]: sns.heatmap(df.isnull())

[15]: <Axes: >
```



```
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(',','.')
```

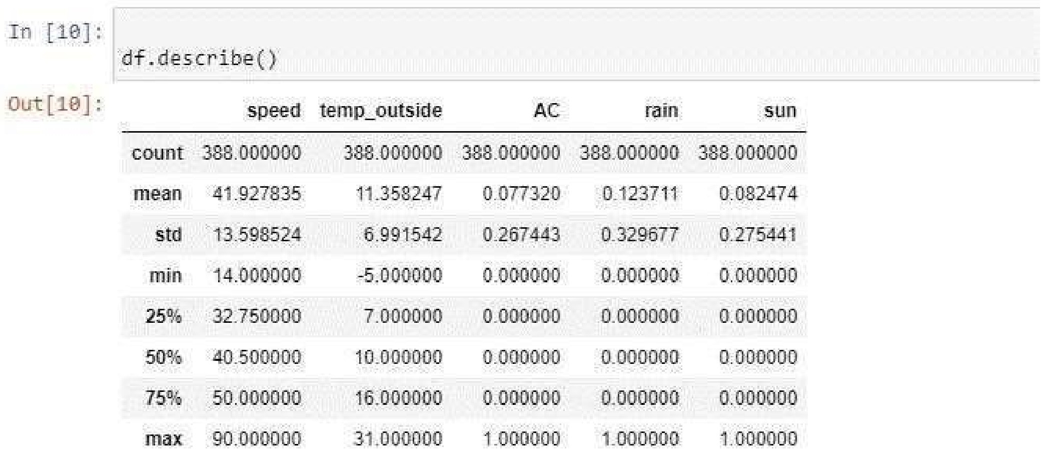
```
In [8]: df.head()
```

```
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
```



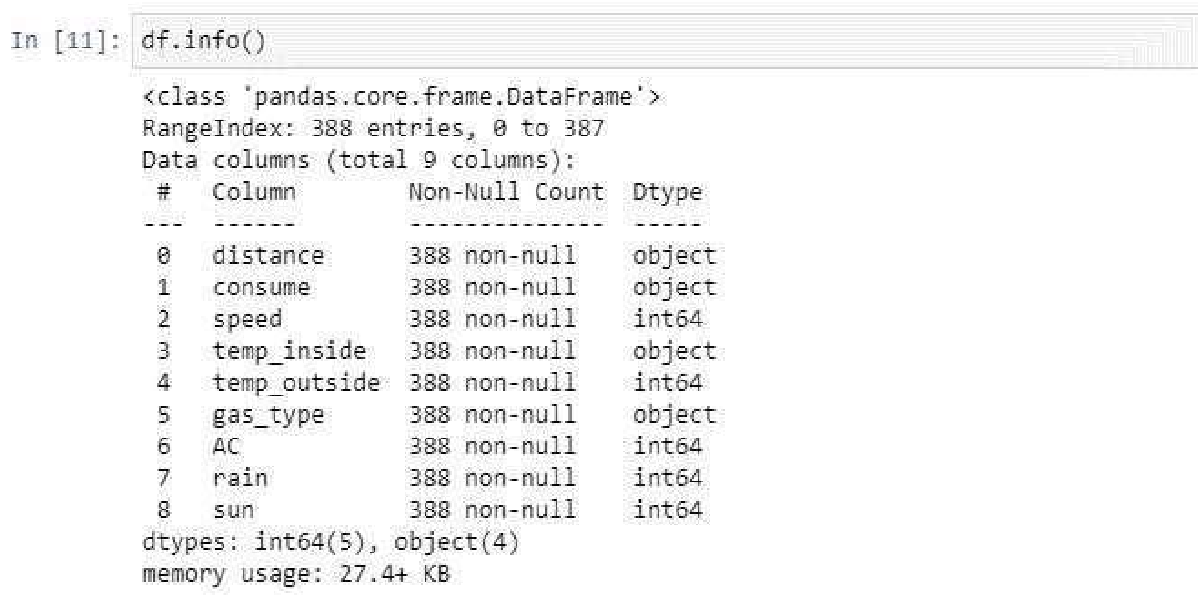
```
In [9]: df['temp_inside'].value_counts()
```

```
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
```

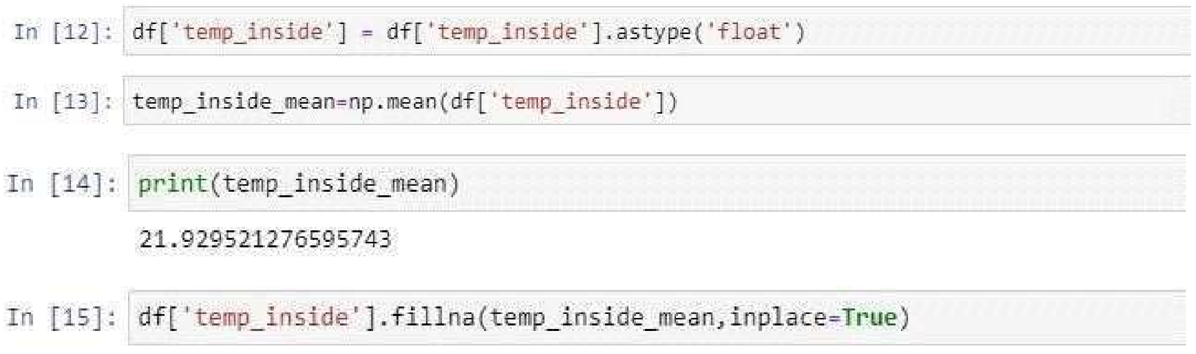


```
In [10]:
         df.describe()
```

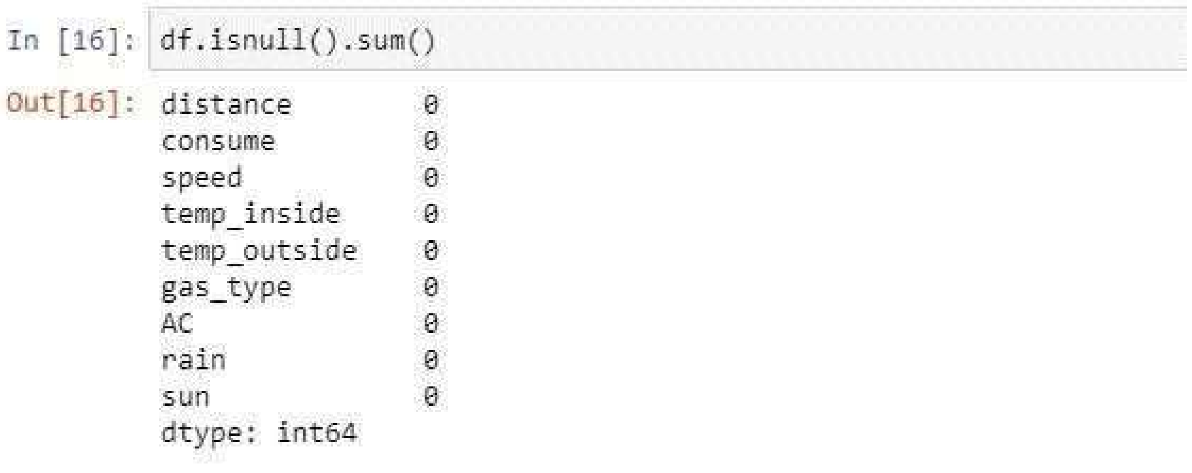
```
Out[10]:
             speed  temp_outside          AC        rain         sun
  count 388.000000    388.000000  388.000000  388.000000  388.000000
   mean  41.927835     11.358247    0.077320    0.123711    0.082474
    std  13.598524      6.991542    0.267443    0.329677    0.275441
    min  14.000000     -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)
```

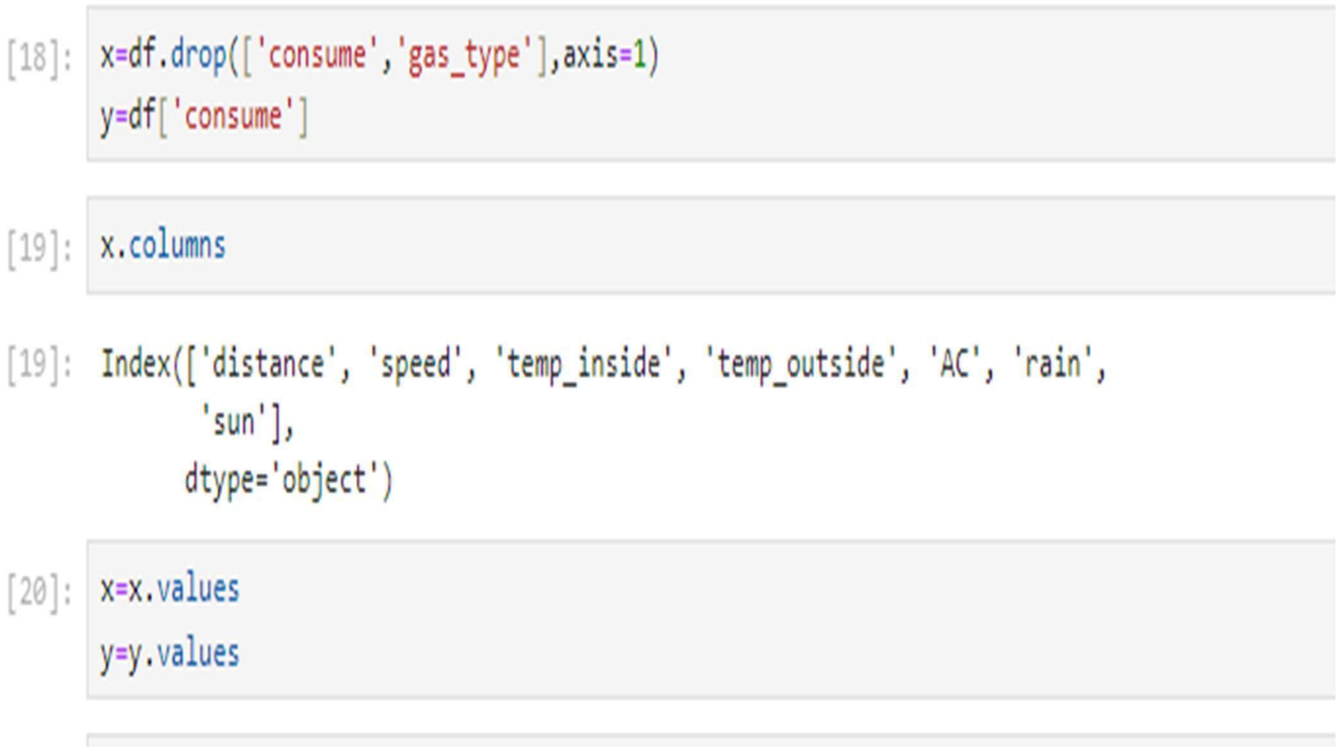


```
In [16]: df.isnull().sum()

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
```

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.

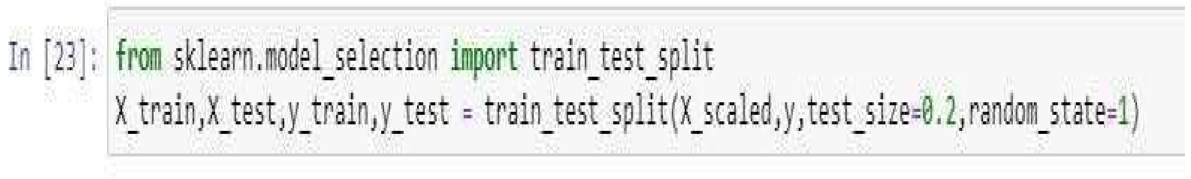
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed.



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

[19]: x.columns

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

[20]: x=x.values
      y=y.values
```

Splitting Data Into Train And Test

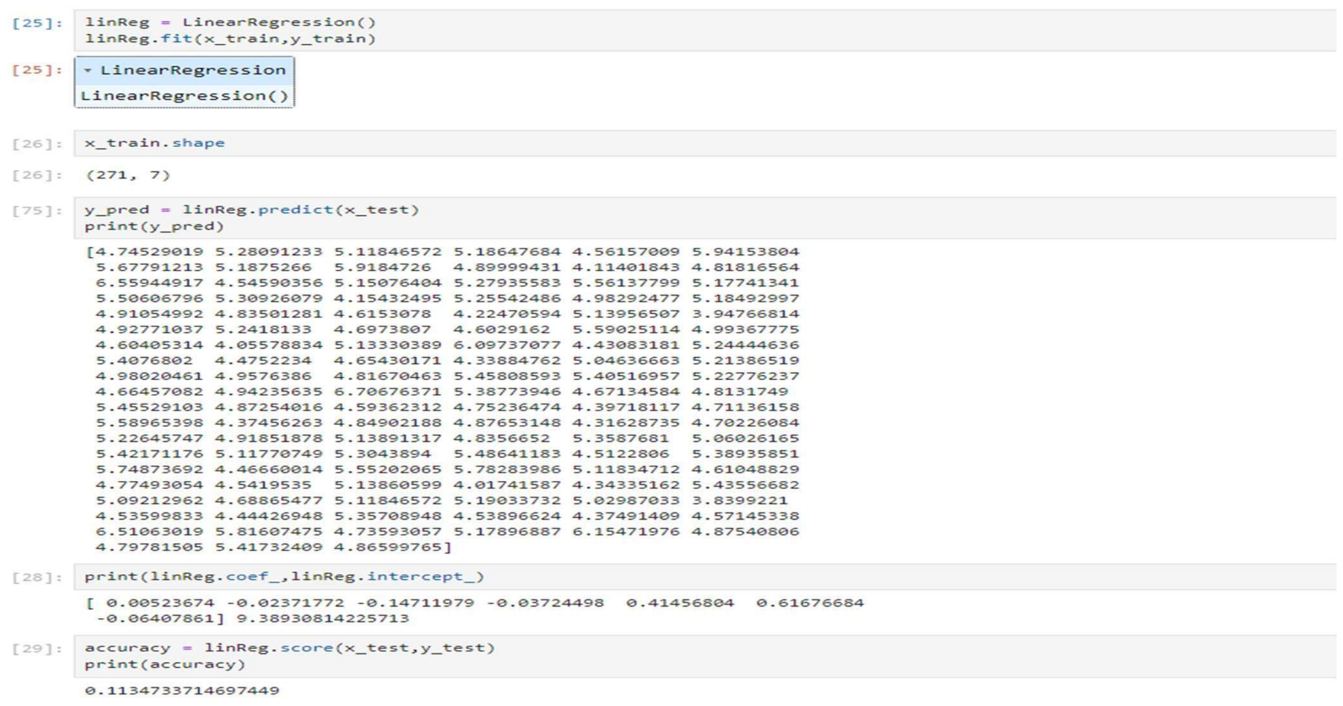
For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



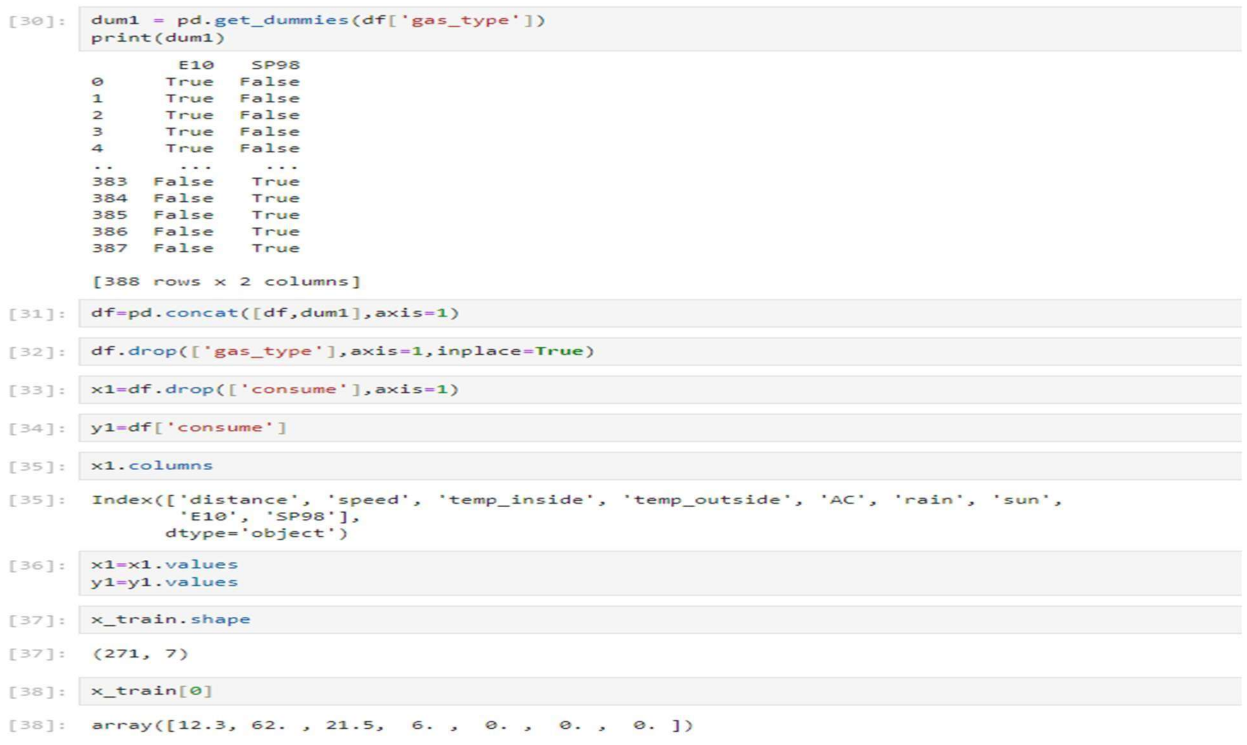
```
In [23]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.2,random_state=1)
```

TRAINING THE MODEL IN MULTIPLE ALGORITHMS:

1.Linear Regression:

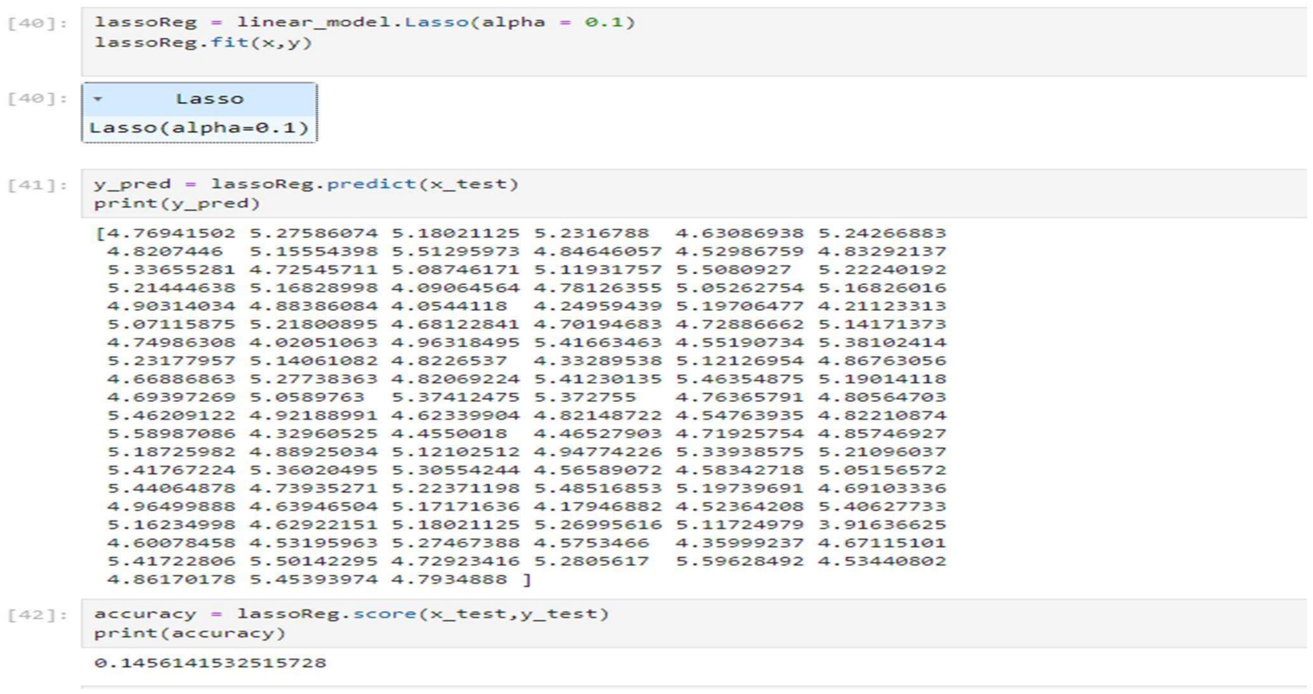


```
[25]: linReg = LinearRegression()
      linReg.fit(x_train,y_train)

[25]: ▾ LinearRegression
      LinearRegression()

[26]: x_train.shape

[26]: (271, 7)

[75]: y_pred = linReg.predict(x_test)
      print(y_pred)

      [4.74529019 5.28091233 5.11846572 5.18647684 4.56157009 5.94153804
       5.67791213 5.1875266  5.9184726  4.89999431 4.11401843 4.81816564
       6.55944917 4.54590356 5.15076404 5.27935583 5.56137799 5.17741341
       5.50606796 5.30926079 4.15432495 5.25542486 4.98292477 5.18492997
       4.91054992 4.83501281 4.6153078  4.22470594 5.13956507 3.94766814
       4.92771037 5.2418133  4.6973807  4.6029162  5.59025114 4.99367775
       4.60405314 4.05578834 5.13330389 6.09737077 4.43083181 5.24444636
       5.4076802  4.4752234  4.65430171 4.33884762 5.04636663 5.21386519
       4.98020461 4.9576386  4.81670463 5.45808593 5.40516957 5.22776237
       4.66457082 4.94235635 6.70676371 5.38773946 4.67134584 4.8131749
       5.45529103 4.87254016 4.59362312 4.75236474 4.39718117 4.71136158
       5.58965398 4.37456263 4.84902188 4.87653148 4.31628735 4.70226084
       5.22645747 4.91851878 5.13891317 4.8356652  5.3587681  5.06026165
       5.42171176 5.11770749 5.3043894  5.48641183 4.5122806  5.38935851
       5.74873692 4.46660014 5.55202065 5.78283986 5.11834712 4.61048829
       4.77493054 4.5419535  5.13860599 4.01741587 4.34335162 5.43556682
       5.09212962 4.68865477 5.11846572 5.19033732 5.02987033 3.8399221
       4.53599833 4.44426948 5.35708948 4.53896624 4.37491409 4.57145338
       6.51063019 5.81607475 4.73593057 5.17896887 6.15471976 4.87540806
       4.79781505 5.41732409 4.86599765]

[28]: print(linReg.coef_,linReg.intercept_)

      [ 0.00523674 -0.02371772 -0.14711979 -0.03724498  0.41456804  0.61676684
       -0.06407861] 9.38930814225713

[29]: accuracy = linReg.score(x_test,y_test)
      print(accuracy)

      0.1134733714697449
```



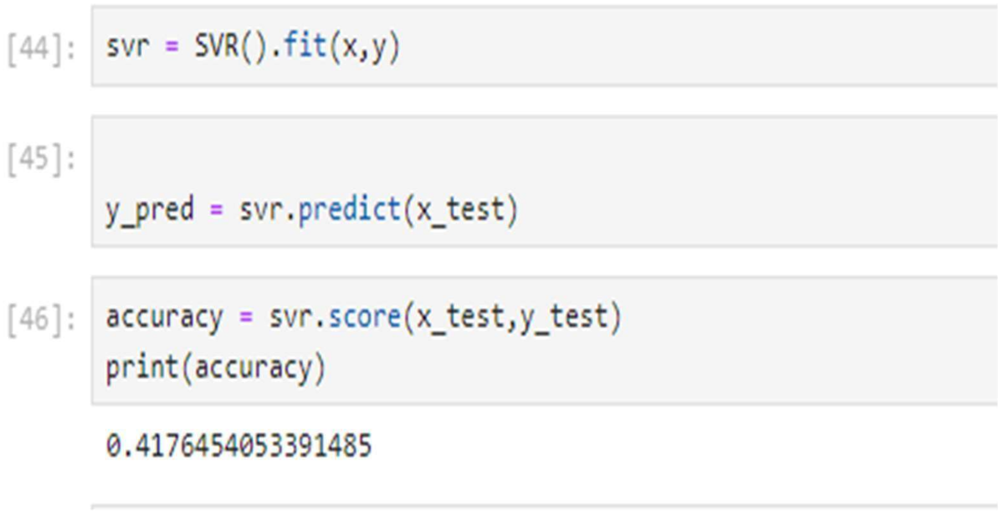
```
[30]: dum1 = pd.get_dummies(df['gas_type'])
      print(dum1)

             E10   SP98
      0     True  False
      1     True  False
      2     True  False
      3     True  False
      4     True  False
      ..     ...    ...
      383  False   True
      384  False   True
      385  False   True
      386  False   True
      387  False   True

      [388 rows x 2 columns]

[31]: df=pd.concat([df,dum1],axis=1)

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

[33]: x1=df.drop(['consume'],axis=1)

[34]: y1=df['consume']

[35]: x1.columns

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

[36]: x1=x1.values
      y1=y1.values

[37]: x_train.shape

[37]: (271, 7)

[38]: x_train[0]

[38]: array([12.3, 62. , 21.5,  6. ,  0. ,  0. ,  0. ])
```

2. Lasso Regression:



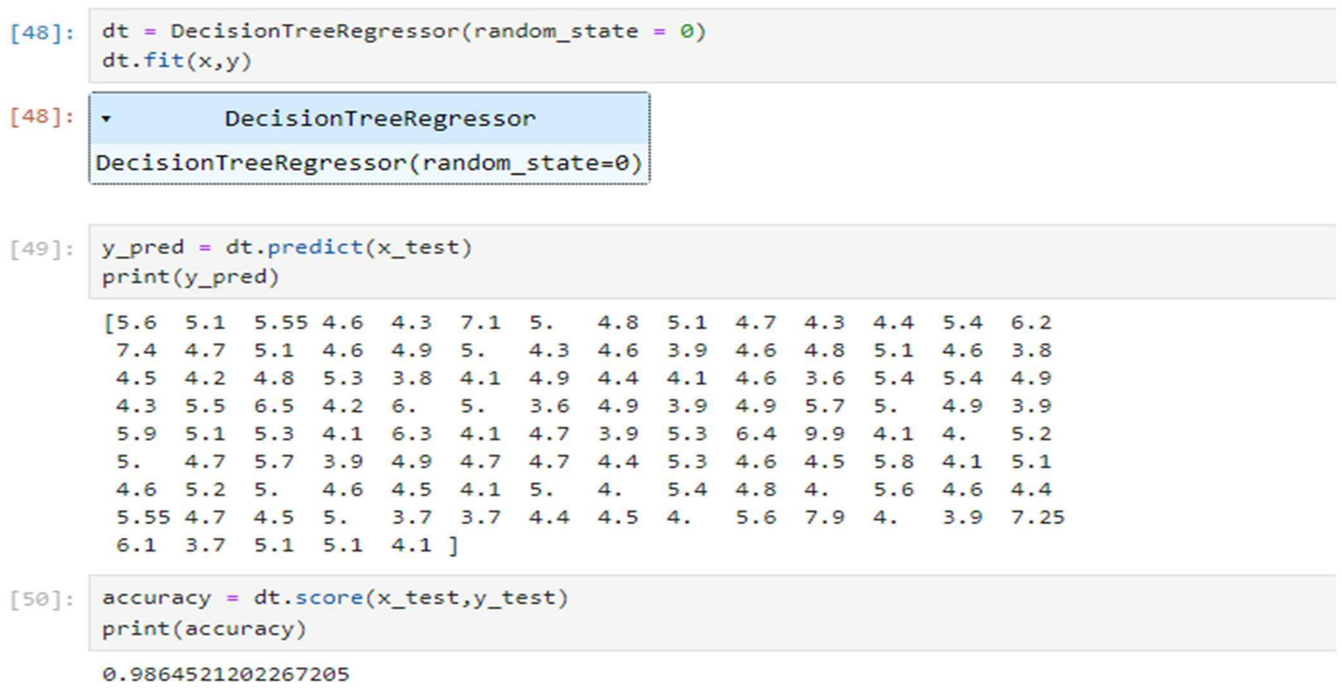
```
[40]: lassoReg = linear_model.Lasso(alpha = 0.1)
      lassoReg.fit(x,y)

[40]: Lasso
      Lasso(alpha=0.1)

[41]: y_pred = lassoReg.predict(x_test)
      print(y_pred)

[4.76941502 5.27586074 5.18021125 5.2316788  4.63086938 5.24266883
 4.8207446  5.15554398 5.51295973 4.84646057 4.52986759 4.83292137
 5.33655281 4.72545711 5.08746171 5.11931757 5.5080927  5.22240192
 5.21444638 5.16828998 4.09064564 4.78126355 5.05262754 5.16826016
 4.90314034 4.88386084 4.0544118  4.24959439 5.19706477 4.21123313
 5.07115875 5.21800895 4.68122841 4.70194683 4.72886662 5.14171373
 4.74986308 4.02051063 4.96318495 5.41663463 4.55190734 5.38102414
 5.23177957 5.14061082 4.8226537  4.33289538 5.12126954 4.86763056
 4.66886863 5.27738363 4.82069224 5.41230135 5.46354875 5.19014118
 4.69397269 5.0589763  5.37412475 5.372755   4.76365791 4.80564703
 5.46209122 4.92188991 4.62339904 4.82148722 4.54763935 4.82210874
 5.58987086 4.32960525 4.4550018  4.46527903 4.71925754 4.85746927
 5.18725982 4.88925034 5.12102512 4.94774226 5.33938575 5.21096037
 5.41767224 5.36020495 5.30554244 4.56589072 4.58342718 5.05156572
 5.44064878 4.73935271 5.22371198 5.48516853 5.19739691 4.69103336
 4.96499888 4.63946504 5.17171636 4.17946882 4.52364208 5.40627733
 5.16234998 4.62922151 5.18021125 5.26995616 5.11724979 3.91636625
 4.60078458 4.53195963 5.27467388 4.5753466  4.35999237 4.67115101
 5.41722806 5.50142295 4.72923416 5.2805617  5.59628492 4.53440802
 4.86170178 5.45393974 4.7934888 ]

[42]: accuracy = lassoReg.score(x_test,y_test)
      print(accuracy)

0.1456141532515728
```

3. SVR Regression:



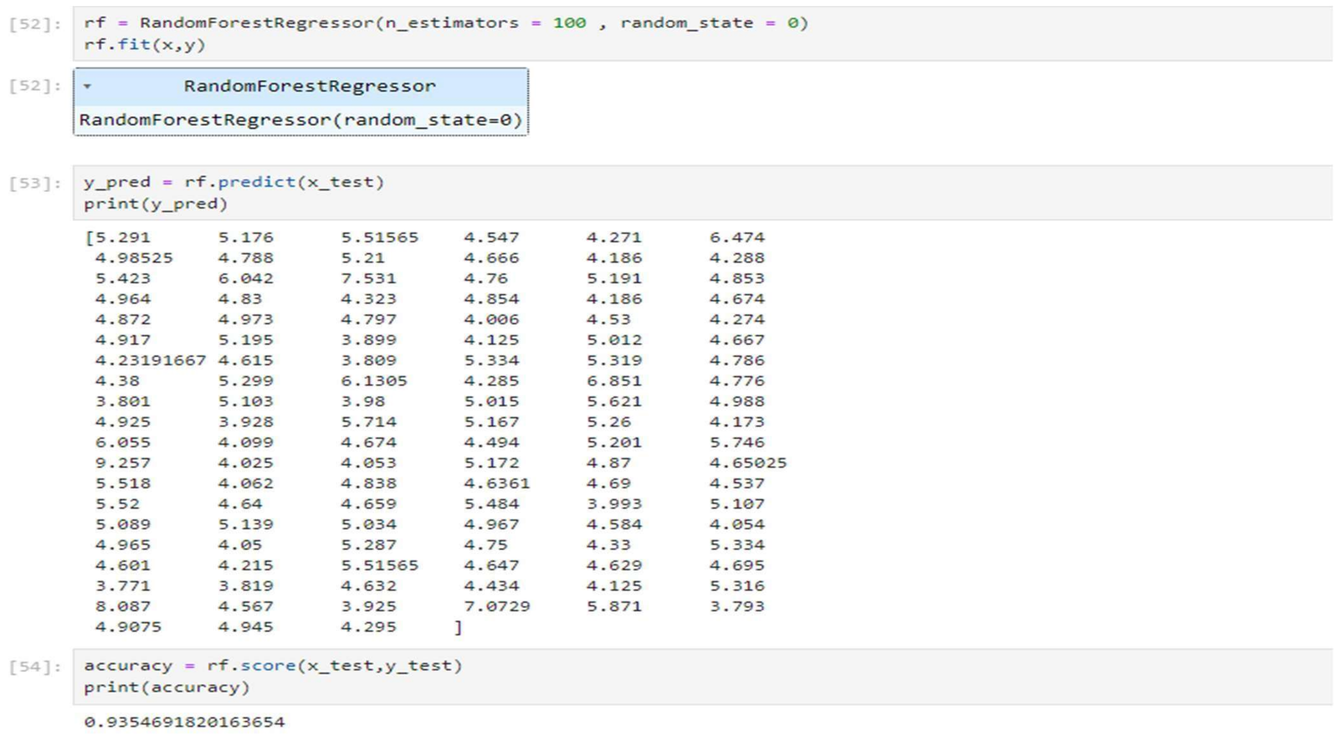
```
[44]: svr = SVR().fit(x,y)

[45]: y_pred = svr.predict(x_test)

[46]: accuracy = svr.score(x_test,y_test)
      print(accuracy)

0.4176454053391485
```

4. Decision Tree:



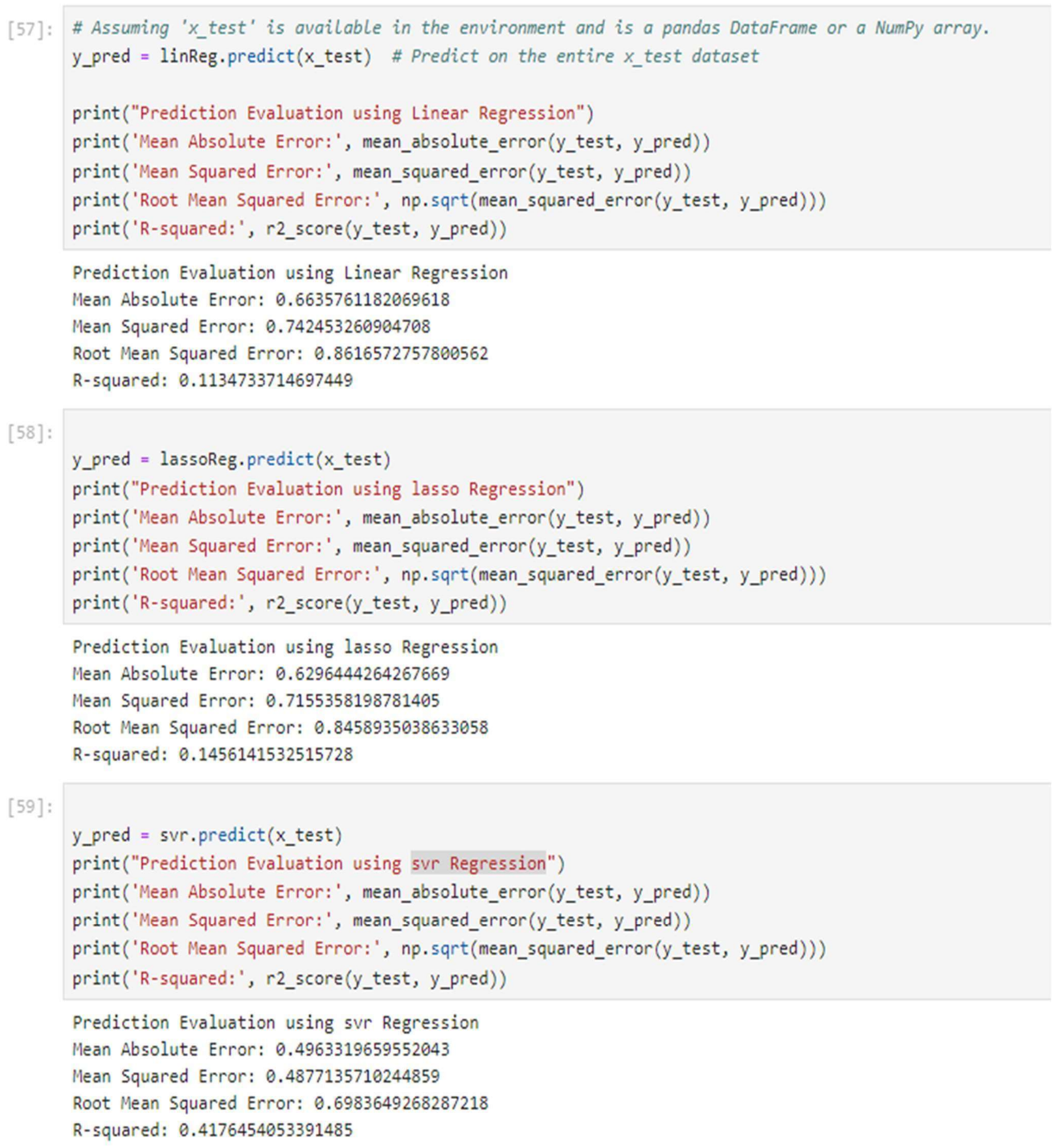
```
[48]: dt = DecisionTreeRegressor(random_state = 0)
      dt.fit(x,y)

[48]: ▾         DecisionTreeRegressor
      DecisionTreeRegressor(random_state=0)

[49]: y_pred = dt.predict(x_test)
      print(y_pred)

      [5.6  5.1  5.55 4.6  4.3  7.1  5.   4.8  5.1  4.7  4.3  4.4  5.4  6.2
       7.4  4.7  5.1  4.6  4.9  5.   4.3  4.6  3.9  4.6  4.8  5.1  4.6  3.8
       4.5  4.2  4.8  5.3  3.8  4.1  4.9  4.4  4.1  4.6  3.6  5.4  5.4  4.9
       4.3  5.5  6.5  4.2  6.   5.   3.6  4.9  3.9  4.9  5.7  5.   4.9  3.9
       5.9  5.1  5.3  4.1  6.3  4.1  4.7  3.9  5.3  6.4  9.9  4.1  4.   5.2
       5.   4.7  5.7  3.9  4.9  4.7  4.7  4.4  5.3  4.6  4.5  5.8  4.1  5.1
       4.6  5.2  5.   4.6  4.5  4.1  5.   4.   5.4  4.8  4.   5.6  4.6  4.4
       5.55 4.7  4.5  5.   3.7  3.7  4.4  4.5  4.   5.6  7.9  4.   3.9  7.25
       6.1  3.7  5.1  5.1  4.1 ]

[50]: accuracy = dt.score(x_test,y_test)
      print(accuracy)

      0.9864521202267205
```

5. Random Forest:



```
[52]: rf = RandomForestRegressor(n_estimators = 100 , random_state = 0)
      rf.fit(x,y)

[52]: ▾         RandomForestRegressor
      RandomForestRegressor(random_state=0)

[53]: y_pred = rf.predict(x_test)
      print(y_pred)

      [5.291      5.176      5.51565    4.547      4.271      6.474
       4.98525    4.788      5.21       4.666      4.186      4.288
       5.423      6.042      7.531      4.76       5.191      4.853
       4.964      4.83       4.323      4.854      4.186      4.674
       4.872      4.973      4.797      4.006      4.53       4.274
       4.917      5.195      3.899      4.125      5.012      4.667
       4.23191667 4.615      3.809      5.334      5.319      4.786
       4.38       5.299      6.1305     4.285      6.851      4.776
       3.801      5.103      3.98       5.015      5.621      4.988
       4.925      3.928      5.714      5.167      5.26       4.173
       6.055      4.099      4.674      4.494      5.201      5.746
       9.257      4.025      4.053      5.172      4.87       4.65025
       5.518      4.062      4.838      4.6361     4.69       4.537
       5.52       4.64       4.659      5.484      3.993      5.107
       5.089      5.139      5.034      4.967      4.584      4.054
       4.965      4.05       5.287      4.75       4.33       5.334
       4.601      4.215      5.51565    4.647      4.629      4.695
       3.771      3.819      4.632      4.434      4.125      5.316
       8.087      4.567      3.925      7.0729     5.871      3.793
       4.9075     4.945      4.295     ]

[54]: accuracy = rf.score(x_test,y_test)
      print(accuracy)

      0.9354691820163654
```

Testing All Models With Multiple Evaluation Metrics:



```
[57]: # Assuming 'x_test' is available in the environment and is a pandas DataFrame or a NumPy array.
      y_pred = linReg.predict(x_test)  # Predict on the entire x_test dataset

      print("Prediction Evaluation using Linear Regression")
      print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
      print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
      print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
      print('R-squared:', r2_score(y_test, y_pred))
```

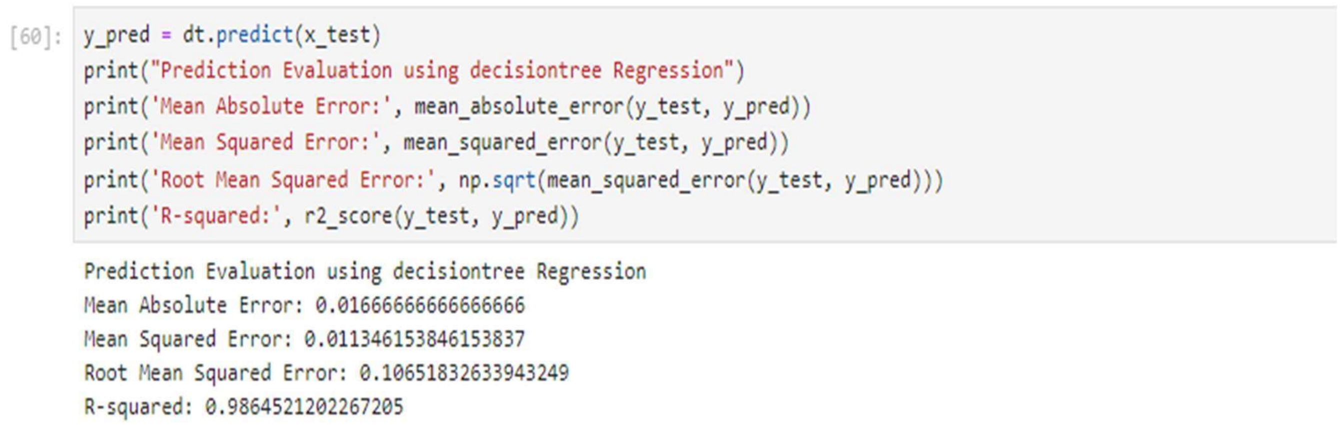
```
Prediction Evaluation using Linear Regression
Mean Absolute Error: 0.6635761182069618
Mean Squared Error: 0.742453260904708
Root Mean Squared Error: 0.8616572757800562
R-squared: 0.1134733714697449
```

```
[58]:
      y_pred = lassoReg.predict(x_test)
      print("Prediction Evaluation using lasso Regression")
      print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
      print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
      print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
      print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using lasso Regression
Mean Absolute Error: 0.6296444264267669
Mean Squared Error: 0.7155358198781405
Root Mean Squared Error: 0.8458935038633058
R-squared: 0.1456141532515728
```

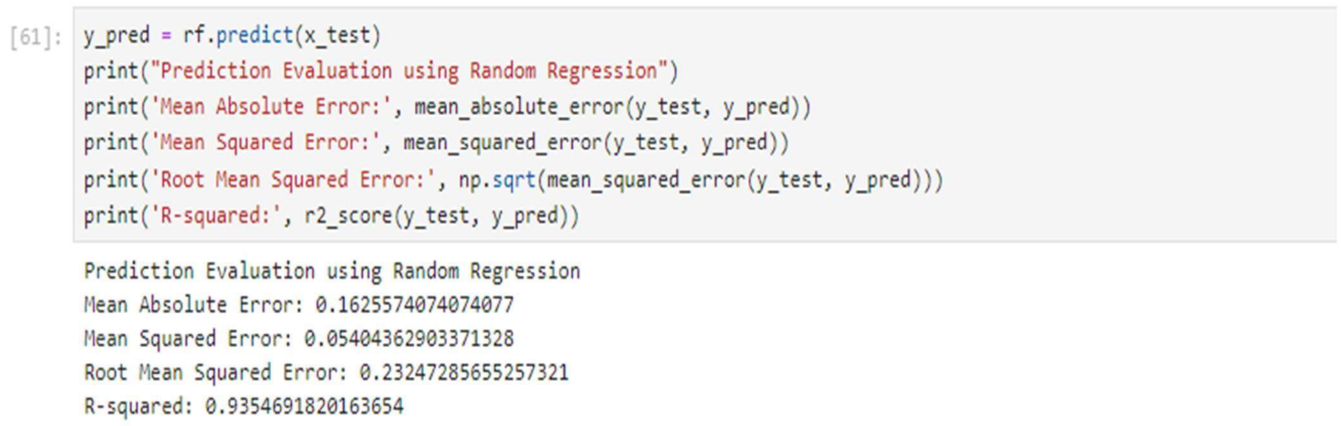
```
[59]:
      y_pred = svr.predict(x_test)
      print("Prediction Evaluation using svr Regression")
      print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
      print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
      print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
      print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using svr Regression
Mean Absolute Error: 0.4963319659552043
Mean Squared Error: 0.4877135710244859
Root Mean Squared Error: 0.6983649268287218
R-squared: 0.4176454053391485
```



```
[60]: y_pred = dt.predict(x_test)
      print("Prediction Evaluation using decisiontree Regression")
      print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
      print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
      print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
      print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using decisiontree Regression
Mean Absolute Error: 0.016666666666666666
Mean Squared Error: 0.011346153846153837
Root Mean Squared Error: 0.10651832633943249
R-squared: 0.9864521202267205
```

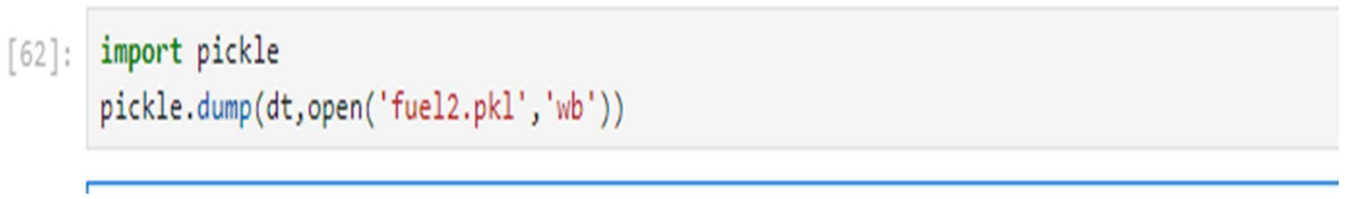


```
[61]: y_pred = rf.predict(x_test)
      print("Prediction Evaluation using Random Regression")
      print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
      print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
      print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
      print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using Random Regression
Mean Absolute Error: 0.1625574074074077
Mean Squared Error: 0.05404362903371328
Root Mean Squared Error: 0.23247285655257321
R-squared: 0.9354691820163654
```

Applying Decision Tree :

Now we are going to create our model with decision tree. As an initial step we have to initialize the
decision 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 decision tree model mean squared error and mean absolute error are used.



```
[62]: import pickle
      pickle.dump(dt,open('fuel2.pkl','wb'))
```

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.

<u>Index.html</u>:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Fuel Predictor Website</title>
   <link rel="stylesheet" href="../static/style.css">
   <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,F
ILL,GRAD@20..48,100..700,0..1,-50..200">
</head>
<body>
   <header>
      <nav class="navbar">
         <a href="{{url_for('home')}}" class="logo">TBFCP<span>.</span></a>
         <ul class="menu-links">
          <span id="close-menu-btn"class="material-symbols-outlined">close</span>
          <li><a href="{{url_for('home')}}">Home</a></li>
          <li><a href="{{url_for('y_predict')}}">Fuel predictor</a></li>
          <li><a href="{{url_for('about')}}">About us</a></li>
          <li><a href="{{url_for('contact')}}">Contact</a></li>
         </ul>
         <span id="menu-btn"class="material-symbols-outlined">menu</span>
      </nav>
   </header>

   <section class="hero-section">
      <div class="content">
          <h1>Know your vechile's fuel consumption for a trip</h1>
          <p>
             For different kinds of high fleet vechiles the fuel consumption is different based
on trip you take
```

```
          </p>
           <button><a href="{{url_for('y_predict')}}">Predict Now</button>
        </div>
    </section>

</body></html>
```

About.html:

```
 <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>About Us page</title>
   <link rel="stylesheet" href="../static/style2.css">
</head>
<body>
   <section class="hero">
      <header>
         <nav class="navbar">
            <a href="{{url_for('home')}}" class="logo">TBFCP<span>.</span></a>
            <ul class="menu-links">
             <li><a href="{{url_for('home')}}">Home</a></li>
             <li><a href="{{url_for('y_predict')}}">Fuel predictor</a></li>
             <li><a href="{{url_for('about')}}">About us</a></li>
             <li><a href="{{url_for('contact')}}">Contact</a></li>
            </ul>
         </nav>
      </header>
      <div class="heading">
         <h1> About Us</h1>
      </div>
      <div class="container">
         <div class="hero-content">
            <h2>Our Service</h2>
            <p>We predict the fuel consumption .<br>
of your vechile based on the trip you take<br>
```

```
we use the machine learning model which calculate the
fuel<br>                    based on the several parameters and
provide accurate results</p>
                <button class="button"> Learn more</button>
            </div>
            <div class="hero-img">
                <img src="../static/fuelstats.jpg" alt="" width="500" height="400">
</div>
            <div class="our-team">
                <h2>Our Team</h2>
                <p>Meet our team members who help you out<br>
with the prediction of fuel consumption for your vechile
</p>
           </div>
           <div class="team">
            <ul class="team-mem">
                <li>
                    <b>Ramya Sree Kasula</b><br>
                    <p>21UK1A0523</p>
                    <p>ramyasree.kasula1729@gmail.com</p>
                </li>
                <li>
                    <b>Akhil Boinapelli</b><br>
                    <p>21UK1A0521</p>
                    <p>akhilboinapelli@gmail.com</p>
                </li>
                <li>
                    <b>Chellamalla Manikanta</b><br>
                    <p>21UK1A0559</p>
                    <p>chellamalla.manikanta28@gmail.com</p>
                </li>
                <li>
                    <b>Bhukya Koushik</b><br>
                    <p>22UK5A0502</p>
                    <p>koushikbhukya8688@gmail.com</p>
                </li>
               </ul>
```

```
          </div>
        </div>
    </section>     <script>         const header =
document.querySelector("header");         const menuBtn =
document.querySelector("#menu-btn");
        const closeMenubtn = document.querySelector("#close-menu-btn");

        menuBtn.addEventListener("click",() => {
header.classList.toggle("show-mobile-menu");
        });

        closeMenubtn.addEventListener("click",() => {
menuBtn.click();
        });
    </script>
</body>
</html>
```

Contact.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Us</title>
    <link rel="stylesheet" href="../static/style3.css">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-
free@6.4.2/css/fontawesome.min.css" integrity="sha384-
BY+fdrpOd3gfeRvTSMT+VUZmA728cfF9Z2G42xpaRkUGu2i3DyzpTURDo5A6C
aLK" crossorigin="anonymous">
</head>
<body>
    <section class="contact">
        <header>
```

```
          <nav class="navbar">
            <a href="{{url_for('home')}}" class="logo">TBFCP<span>.</span></a>
            <ul class="menu-links">
             <li><a href="{{url_for('home')}}">Home</a></li>
             <li><a href="{{url_for('y_predict')}}">Fuel predictor</a></li>
             <li><a href="{{url_for('about')}}">About us</a></li>
             <li><a href="{{url_for('contact')}}">Contact</a></li>
            </ul>
          </nav>
        </header>
        <div class="content">
          <h2>Contact us</h2><br>
          <p>Feel to contact us any time at your service.</p>
        </div>
        <div class="container">
          <div class="contactInfo">
             <div class="box">
                <div class="icon"><i class="fa fa-map-marker" aria-
hidden="true"></i></div>
                <div class="text">
                   <h3>Address</h3>
                   <p>Vaagdevi Engineering college ,<br> Bollikuntaa , <br>
506002</p>
                </div>
             </div>
             <div class="box">
                <div class="icon"><i class="fa fa-phone" aria-hidden="true"></i></div>
                <div class="text">
                   <h3>phone</h3>
                   <p>9160678963 <br>8919572315 <br> 8919816641 </p>
                </div>
             </div>
             <div class="box">
```

```
                    <div class="icon"><i class="fa fa-envelope" aria-
hidden="true"></i></div>
                    <div class="text">
                        <h3>Email</h3>
                        <p> ramyasree.kasula1729@gmail.com<br>
akhilboinapelli@gmail.com <br> chellamalla.manikanta28@gmail.com</p>
                    </div>
                </div>
                <div class="contactForm">
                  <form>
                     <h2>Send Message</h2>
                     <div class="inputBox">
                        <input type="text" name="" required>
                        <span>Full Name</span>
                     </div>
                     <div class="inputBox">
                        <input type="text" name="" required>
                        <span>Email</span>
                     </div>
                     <div class="inputBox">
                        <textarea required="required"></textarea>
                        <span>Enter your Message...</span>
                     </div>
                     <div class="inputBox">
                        <input type="submit" name="" value="send">
                     </div>
                  </form>
                </div>
            </div>
        </div>
    </section>
    </body>
</html>
```

<u>Y_predict.html</u>:

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fuel predictor</title>
    <link rel="stylesheet" href="../static/style1.css">
</head>
<body>

    <div class="container">
        <header>
            <nav class="navbar">
               <a href="{{url_for('home')}}" class="logo">TBFCP<span>.</span></a>
               <ul class="menu-links">
                <span id="close-menu-btn"class="material-symbols-
outlined">close</span>
                <li><a href="{{url_for('home')}}">Home</a></li>
                <li><a href="{{url_for('y_predict')}}">Fuel predictor</a></li>
                <li><a href="{{url_for('about')}}">About us</a></li>
                <li><a href="{{url_for('contact')}}">Contact</a></li>
               </ul>
               <span id="menu-btn"class="material-symbols-outlined">menu</span>
            </nav>
        </header>
    <div class="super">
       <div class="main">
         <div class="sub">
            <h1>Fuel Consumption Predictor</h1>
        <form action = "{{ url_for('predict')}}" method = "POST">
            <div class="input-group">
```

```
              <div class="input-field">
                 <input type = "number" name="dist" min="1" placeholder =
"distance(km) "required>
              </div>
              <div class="input-field">
                 <input type = "number" name="speed" min="1" placeholder =
"speed(km/h)" required>
              </div>
              <div class="input-field">
                 <input type = "number " name="ti" min="1" placeholder =
"temparature_inside(C)" required>
              </div>
              <div class="input-field">
                 <input type = "number" name="to"min="1" placeholder =
"temparature_outside(C)" required>
              </div>
              <div class="input-field">
                 <input type = "number" name="ac" min="1" placeholder = "Ac"
required>
              </div>
              <div class="input-field">
                 <input type = "number" name="rain"  min="1" placeholder = "rain"
required>
              </div>
              <div class="input-field">
                 <input type = "number"  name="sun" min="1" placeholder = "sun"
required>
              </div>
           </div>
           <div class="btn-field">
              <button type="submit">Submit</button>
           </div>
        </form>
         </div>
        </div>
    </div>
    </div>
</body>
</html>
```

Result.html:

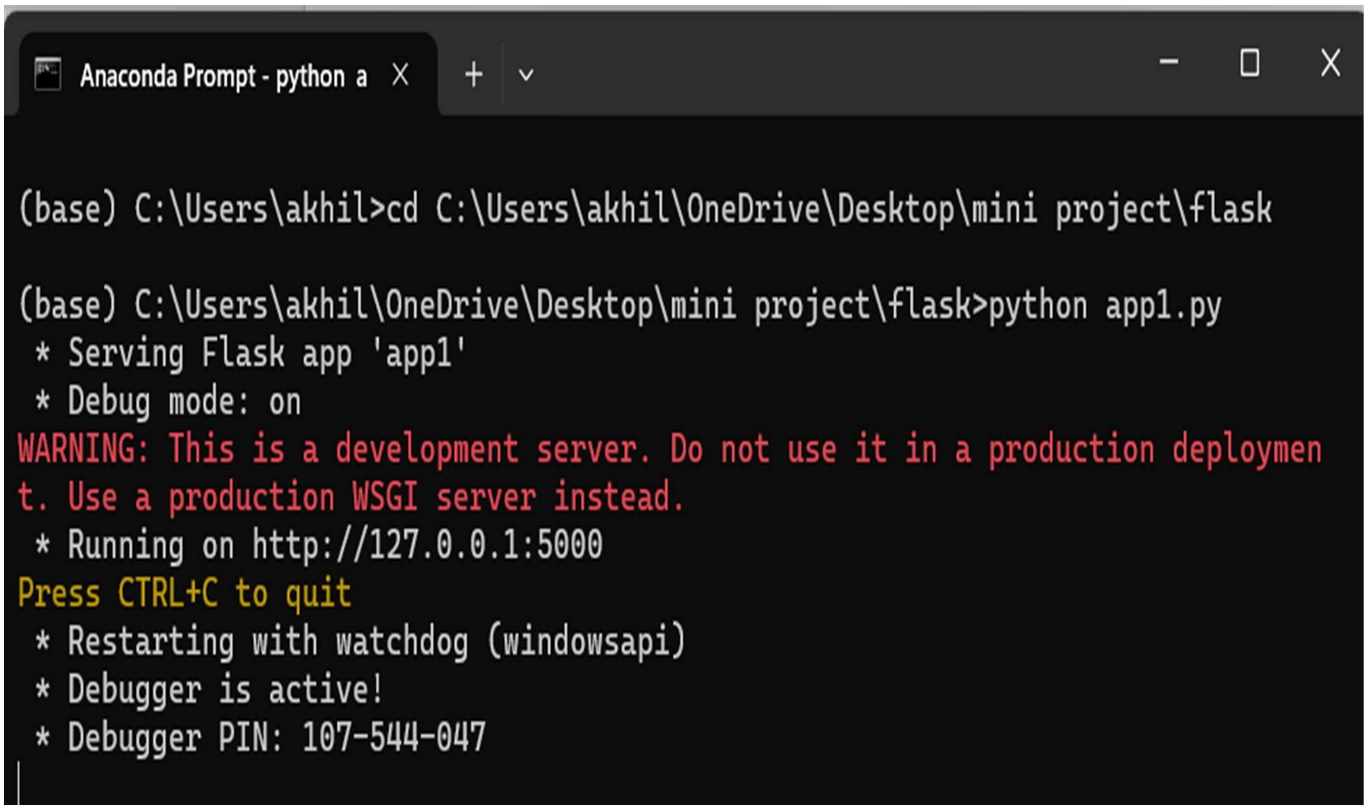
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>result</title>
    <link rel="stylesheet" href="../static/style1.css">
</head>
<body>
    <div class="mini">
        <header>
            <nav class="navbar">
               <a href="{{url_for('home')}}" class="logo">TBFCP<span>.</span></a>
<ul class="menu-links">

                <li><a href="{{url_for('home')}}">Home</a></li>
                <li><a href="{{url_for('y_predict')}}">Fuel predictor</a></li>
                <li><a href="{{url_for('about')}}">About us</a></li>
                <li><a href="{{url_for('contact')}}">Contact</a></li>
               </ul>

            </nav>
        </header>
       <div class="form-box">
        <div class = "result">
            <h2>{{prediction_text}}</h2>
            </div>
           </div>
         </div>
      </body>
      </html>
```

Build The Python Flask App (app1.py):

```
from flask import Flask, render_template, request import pickle model =
pickle.load(open(r'C:\Users\akhil\OneDrive\Desktop\mini project\fuel2.pkl','rb'))
app=Flask(__name__) @app.route('/') def home():
   return render_template('index.html')
@app.route('/about') def
about():
   return render_template('about.html')
@app.route('/contact') def
contact():
   return render_template('contact.html')
@app.route('/y_predict',methods=['POST','GET'])
def y_predict():
   return render_template('y_predict.html')
@app.route('/result', methods=['GET','POST'])
def predict():
   print(request.method)     if
request.method == 'POST' :
      x_test=[[float(x) for x in request.form.values()]]
print('actual',x_test)        pred=model.predict(x_test)        return
render_template('result.html', \
prediction_text=('Car fuel Consumption(L/100km) \
                                    : ',pred[0]))
   return render_template('result.html')

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

Run The Application:

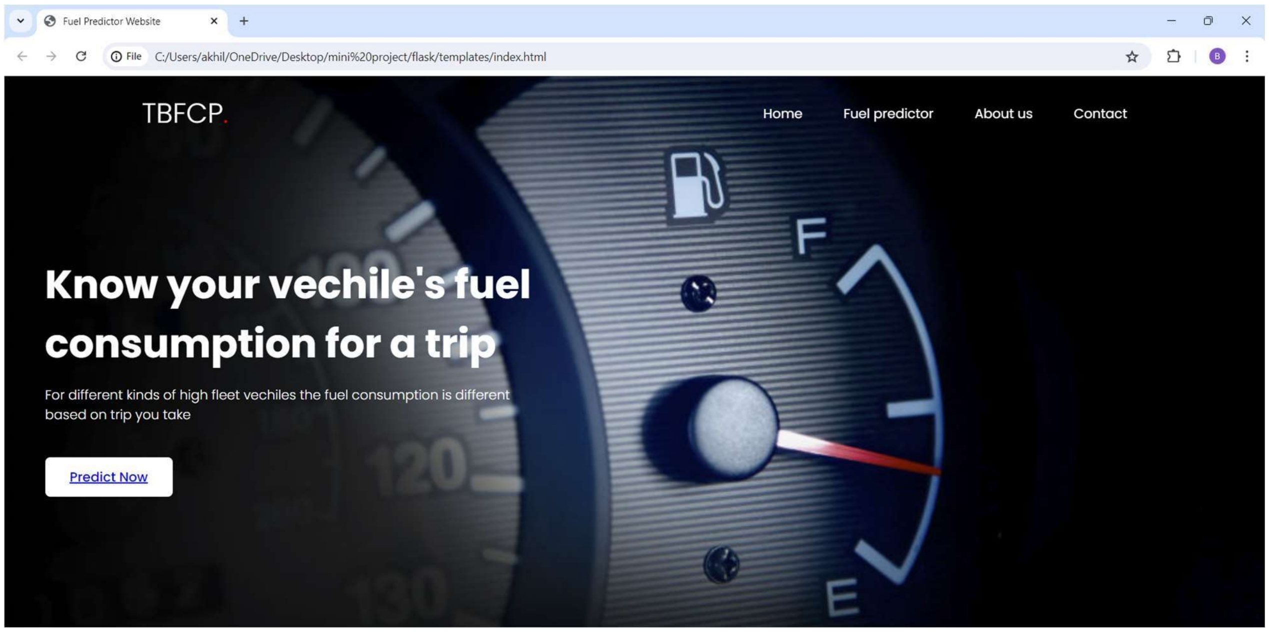
Open anaconda prompt go to project folder and in that go to flask folder and run the python file by using the command “python app1.py”



```
(base) C:\Users\akhil>cd C:\Users\akhil\OneDrive\Desktop\mini project\flask

(base) C:\Users\akhil\OneDrive\Desktop\mini project\flask>python app1.py
 * Serving Flask app 'app1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deploymen
t. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 107-544-047
```

11. RESULT

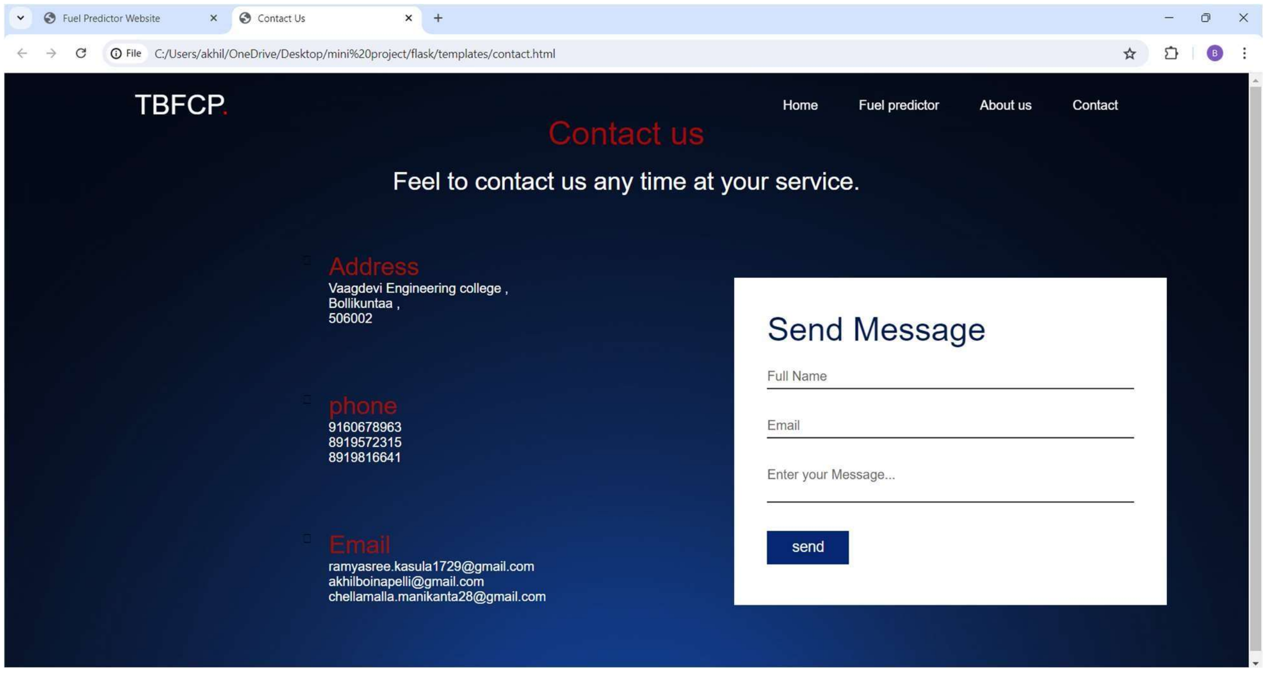
HOME PAGE:


Fuel Predictor Website
C:/Users/akhil/OneDrive/Desktop/mini%20project/flask/templates/index.html
TBFCP.
Home
Fuel predictor
About us
Contact
Predict Now

Know your vechile's fuel consumption for a trip

For different kinds of high fleet vechiles the fuel consumption is different based on trip you take

CONTACT PAGE:



Contact us

Feel to contact us any time at your service.

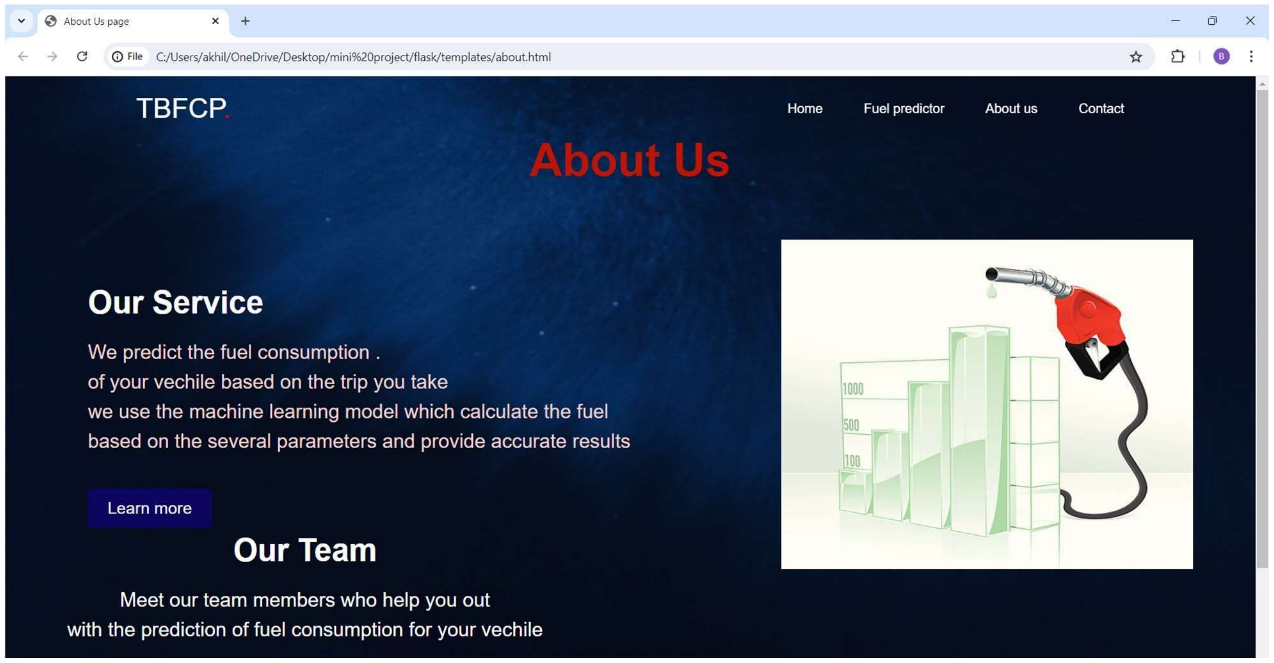
Address
Vaagdevi Engineering college ,
Bollikuntaa ,
506002

phone
9160678963
8919572315
8919816641

Email
ramyasree.kasula1729@gmail.com
akhilboinapelli@gmail.com
chellamalla.manikanta28@gmail.com

Send Message

ABOUT US PAGE:


About Us page

C:/Users/akhil/OneDrive/Desktop/mini%20project/flask/templates/about.html

TBFCP.

Home Fuel predictor About us Contact

About Us

Our Service

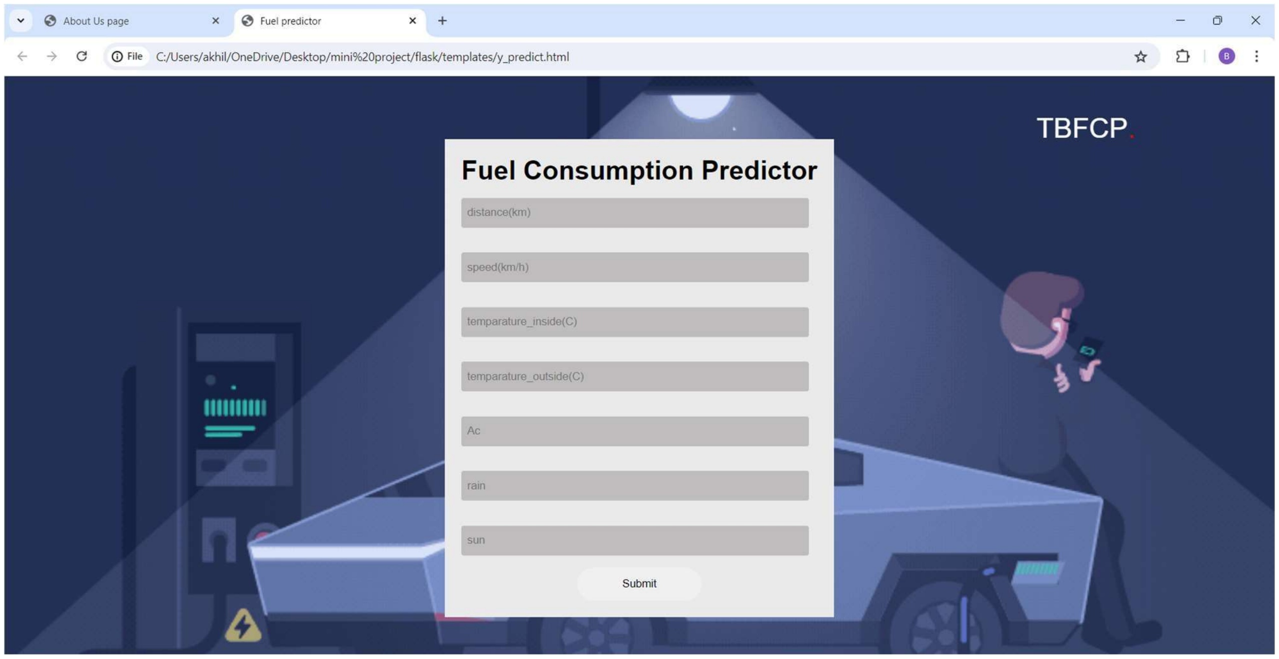
We predict the fuel consumption .
of your vechile based on the trip you take
we use the machine learning model which calculate the fuel
based on the several parameters and provide accurate results

Learn more

Our Team

Meet our team members who help you out
with the prediction of fuel consumption for your vechile

PREDICT PAGE:


About Us page Fuel predictor

C:/Users/akhil/OneDrive/Desktop/mini%20project/flask/templates/y_predict.html

TBFCP.

Fuel Consumption Predictor

distance(km)

speed(km/h)

temparature_inside(C)

temparature_outside(C)

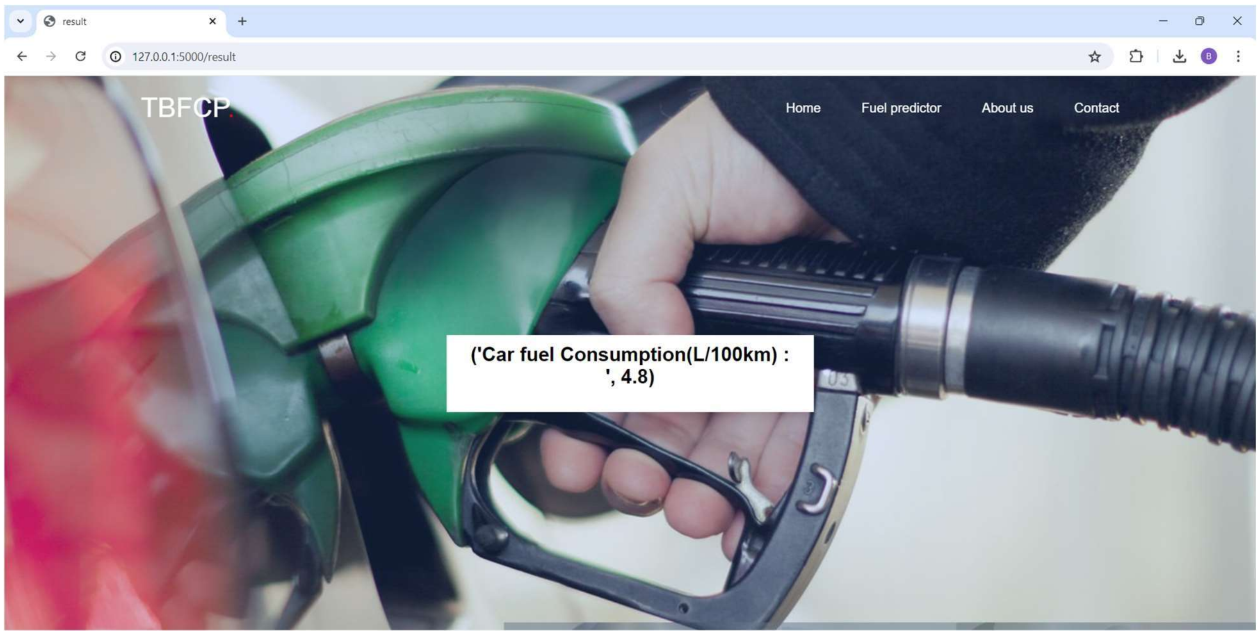
Ac

rain

sun

Submit

RESULT PAGE:



('Car fuel Consumption(L/100km) : ', 4.8)