TRAFFIC VOLUME ESTIMATION

- Name of the Student: **B.ATCHYUT**
- Register Number: **22Q71A4206**

TRAFFIC VOLUME ESTIMATION

USING MACHINE LEARNING

INTRODUCTION

- Growth in the number of vehicles and degree of urbanization means that the annual cost of traffic jams is increasing in cities. This leads to a decrease in the quality of life among citizens through a considerable waste of time and excessive fuel consumption and air pollution in congested areas.
- Traffic congestion has been one of the major issues that most metropolises are facing despite measures being taken to mitigate and reduce it.
- . Early analysis of congestion events and prediction of traffic volumes is a crucial step to identify traffic bottlenecks, which can be utilized to assist traffic management centres

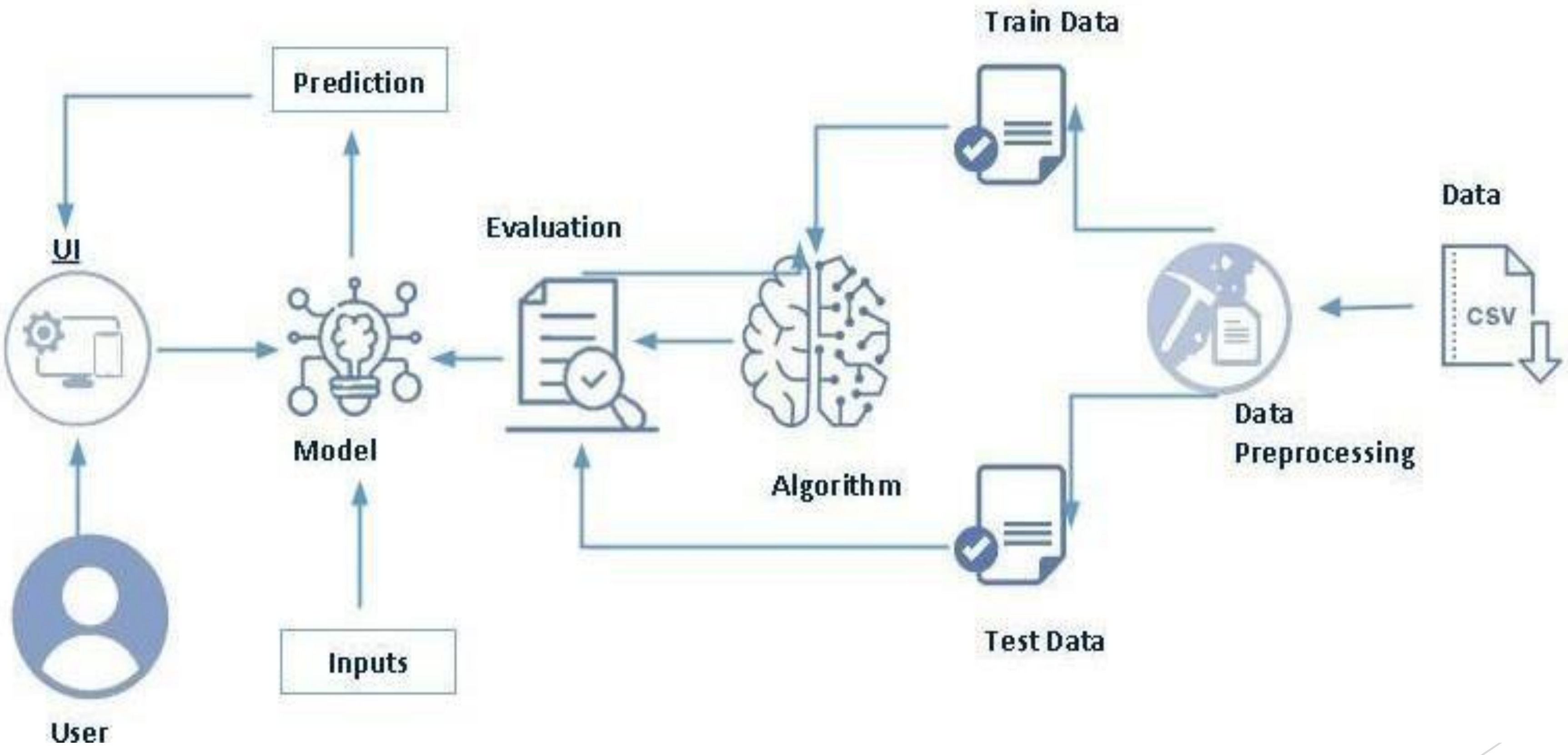
- Traffic jams on Urban Network are increasing day by day, because the traffic demand increases, and the speed of the vehicles is drastically reduced thus causing longer vehicular queuing and more such cases substantially hamper the traffic flow by giving rise to holdup.
- **MOTIVATION:** With the progress of urbanization and therefore the recognition of automobiles, transportation problems are becoming more and more challenging: the traffic volume flow is congested, wear n tear of vehicles, delays end in the late time of arrival at the meeting, accidents are frequent, and wastage of fuel while waiting in traffic.
- **PROBLEM DEFINITION:** Now? The question arises of how to improve the capacitor y of the road network. To solve this problem the first solution that occurs to most of us is to build more highways, expanding the number of lanes on the road.

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.
- You will able to analyze or get insights into data through visualization.
- Applying different algorithms according to a dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

SYSTEM ARCHITECTURE/IDEATION MAP



MODULE IMPLEMENTATION

PROJECT FLOW:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analysis the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data pre-processing ,Visualising and analysing data
- Model building
- Application Building

DATA PREPROCESSING

Data Pre-processing includes the following main tasks

- Import the Libraries.
- Importing the dataset and Analysing the Data
- Checking for Null Values.
- Data Visualization.
- Splitting the Dataset into Dependent and Independent Variables
- Feature Scaling.
- Splitting Data into Train and Test.

Import Necessary Libraries

It is important to import all the necessary libraries such as pandas, NumPy, matplotlib.

- **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
- **Pandas-** It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Matplotlib-** Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python
- **Sklearn** – which contains all the modules required for model building.

Importing the neccessary libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
```

Importing the Dataset and Analysing the Data

- You might have your data in .csv files, .excel files
- Let's load a .csv data file into pandas using read_csv() function.We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).
- If your dataset is in some other location, Then

Data=pd.read_csv(r"File_location/datasetname.csv")

Importing the Dataset

```
In [3]: data=pd.read_csv(r"C:\Users\ganir\OneDrive\Desktop\traffic volume.csv")
```

- **Note:** r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

- If the dataset is in the same directory of your program, you can directly read it, without giving raw as r.
- Our Dataset weatherAus.csv contains the following Columns
- Holiday - working day or holiday
- Temp- temperature of the day
- Rain and snow – whether it is raining or snowing on that day or not
- Weather = describes the weather conditions of the day
- Date and time = represents the exact date and time of the day
- Traffic volume – output column

- The output column to be predicted is Traffic volume.Based on the input variables we predict the volume of the traffic. The predicted output gives them a fair idea of the count of traffic

Analyse The Data

- **head()** method is used to return top n (5 by default) rows of a DataFrame or series.

```
data.head()
```

	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	None	288.28	0.0	0.0	Clouds	02-10-2012	09:00:00	5545
1	None	289.36	0.0	0.0	Clouds	02-10-2012	10:00:00	4516
2	None	289.58	0.0	0.0	Clouds	02-10-2012	11:00:00	4767
3	None	290.13	0.0	0.0	Clouds	02-10-2012	12:00:00	5026
4	None	291.14	0.0	0.0	Clouds	02-10-2012	13:00:00	4918

- **describe()** method computes a summary of statistics like count, mean, standard deviation, min, max, and quartile values.

```
data.describe()
```

	temp	rain	snow	traffic_volume
count	48151.000000	48202.000000	48192.000000	48204.000000
mean	281.205351	0.334278	0.000222	3259.818355
std	13.343675	44.790062	0.008169	1986.860670
min	0.000000	0.000000	0.000000	0.000000
25%	272.160000	0.000000	0.000000	1193.000000
50%	282.460000	0.000000	0.000000	3380.000000
75%	291.810000	0.000000	0.000000	4933.000000
max	310.070000	9831.300000	0.510000	7280.000000

- From the data, we infer that there are only decimal values and no categorical values.
  info() gives information about the data - paste the image here.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48204 entries, 0 to 48203
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   holiday         48204 non-null  object 
 1   temp            48151 non-null  float64
 2   rain            48202 non-null  float64
 3   snow            48192 non-null  float64
 4   weather         48155 non-null  object 
 5   date            48204 non-null  object 
 6   Time            48204 non-null  object 
 7   traffic_volume  48204 non-null  int64  
dtypes: float64(3), int64(1), object(4)
memory usage: 2.9+ MB
```

Checking for Null Values /Handling Missing Values

- The Most important step in data pre-processing is dealing with missing data, the presence of missing data in the dataset can lead to low accuracy.
- Check whether any null values are there or not. if it is present then the following can be done.

```
data.isnull().sum()
```

```
holiday            0
temp              53
rain               2
snow              12
weather           49
date               0
Time               0
traffic_volume     0
dtype: int64
```

There are missing values in the dataset, we will fill the missing values in the columns.

- We are using mean and mode methods for filling the missing values
  - Columns such as temp, rain, and snow are the numeric columns, when there is a numeric column you should fill the missing values with the mean/median method. so here we are using the mean method to fill the missing values.
  - Weather column has a categorical data type, in such case missing data needs to be filled with the most repeated/ frequent value. Clouds are the most repeated value in the column, so imputing with clouds value.

Handling the missing values

```
data['temp'].fillna(data['temp'].mean(),inplace=True)
data['rain'].fillna(data['rain'].mean(),inplace=True)
data['snow'].fillna(data['snow'].mean(),inplace=True)
```

```
from collections import Counter
```

```
print(Counter(data['weather']))
```

```
Counter({'Clouds': 15144, 'Clear': 13383, 'Mist': 5942, 'Rain': 5665, 'Snow': 2875, 'Drizzle': 1818, 'Haze': 1359, 'Thunderstor
m': 1033, 'Fog': 912, nan: 49, 'Smoke': 20, 'Squall': 4})
```

```
data['weather'].fillna('Clouds',inplace=True)
```

```
data.isnull().sum()
```

```
holiday           0
temp              0
rain              0
snow              0
weather           0
date              0
Time              0
traffic_volume    0
dtype: int64
```

Data Visualization

- Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.
- Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.
  - To visualize the dataset we need libraries called Matplotlib and Seaborn.
  - The Matplotlib library is a Python 2D plotting library that allows you to generate plots, scatter plots, histograms, bar charts etc.
- Let's visualize our data using Matplotlib and seaborn library.

  Before diving into the code, let's look at some of the basic properties we will be using when plotting.

  xlabel: Set the label for the x-axis.

  ylabel: Set the label for the y-axis.

  title: Set a title for the axes.

  Legend: Place a legend on the axes.

1. **data.corr()** gives the correlation between the columns

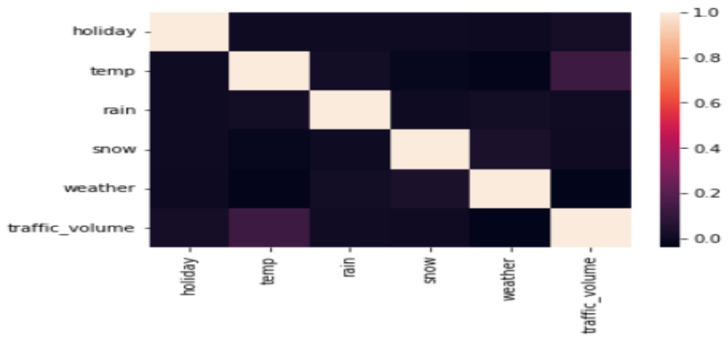
Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negative correlation.

```
data.corr()
```

	holiday	temp	rain	snow	weather	traffic_volume
holiday	1.000000	-0.000472	0.000066	0.000432	-0.004328	0.018676
temp	-0.000472	1.000000	0.009070	-0.019758	-0.033559	0.130034
rain	0.000066	0.009070	1.000000	-0.000090	0.009542	0.004714
snow	0.000432	-0.019758	-0.000090	1.000000	0.036662	0.000735
weather	-0.004328	-0.033559	0.009542	0.036662	1.000000	-0.040035
traffic_volume	0.018676	0.130034	0.004714	0.000735	-0.040035	1.000000

```
sns.heatmap(data.corr())
```

<AxesSubplot:>



- Correlation strength varies based on colour, lighter the colour between two variables, more the strength between the variables, darker the colour displays the weaker correlation
- We can see the correlation scale values on the left side of the above image

- **2. Pair Plot**: Plot pairwise relationships in a dataset.

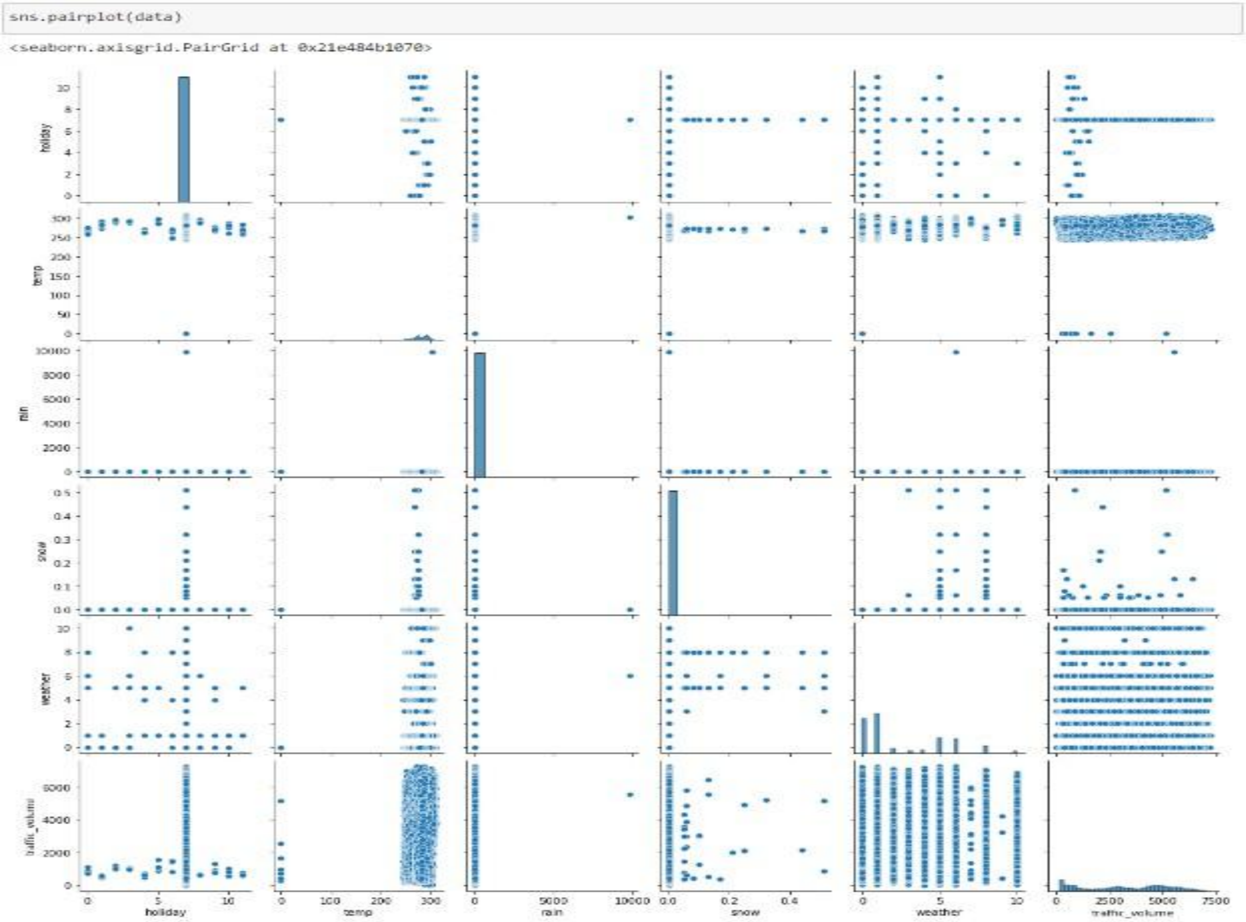
A pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or making a linear separation in our data-set.

  - By default, this function will create a grid of Axes such that each numeric variable in data will be shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.

  - We implement this using the below code.

Code:- sns.pairplot(data)

- Pair plot usually gives pairwise relationships of the columns in the dataset

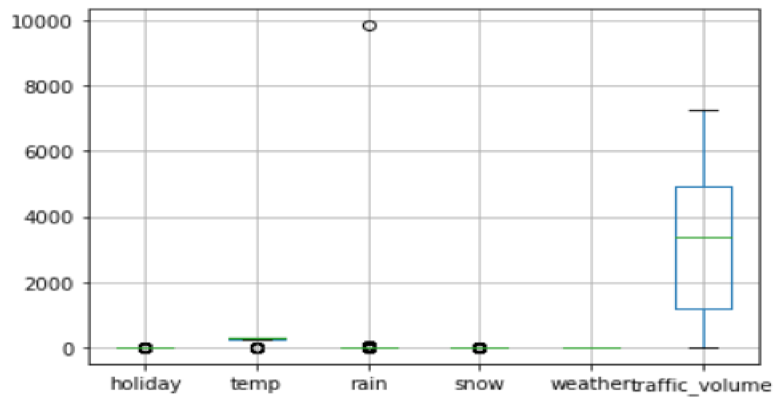


- From the above pair plot, we infer that
- 1.From the above plot we can draw inferences such as linearity and strength between the variables. how features are correlated(positive, neutral and negative)

- **3. Box Plot:**
  - Box-plot is a type of chart often used in explanatory data analysis. Box plots visually show the distribution of numerical data and skewness through displaying the data quartiles (or percentiles) and averages.
  - Box plots are useful as they show the average score of a data set. The median is the average value from a set of data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less. jupyter has a built-in function to create a boxplot called boxplot(). A boxplot plot is a type of plot that shows the spread of (

```
data.boxplot()
```

<AxesSubplot:>



From the above box plot, we infer how the data points are spread and the existence of the outliers

- **4. Data and time columns** need to be split into columns so that analysis and training of the model can be done in an easy way, so we use the split function to convert date into the year, month and day. time column into hours, minutes and seconds.

Splitting Date and Time

```
data[["day","month","year"]] = data["date"].str.split("-", expand = True)
```

```
data[["hours", "minutes", "seconds"]] = data["Time"].str.split(":", expand = True)
```

```
data.drop(columns=['date','Time'],axis=1,inplace=True)
```

```
data.head()
```

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
0	7	288.28	0.0	0.0	1	5545	02	10	2012	09	00	00
1	7	289.36	0.0	0.0	1	4516	02	10	2012	10	00	00
2	7	289.58	0.0	0.0	1	4767	02	10	2012	11	00	00
3	7	290.13	0.0	0.0	1	5026	02	10	2012	12	00	00
4	7	291.14	0.0	0.0	1	4918	02	10	2012	13	00	00

Splitting The Dataset Into Dependent And Independent Variable

- ► In machine learning, the concept of the dependent variable (y) and independent variables(x) is important to understand. Here, the Dependent variable is nothing but output in dataset and the independent variable is all inputs in the dataset.
- ► With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

To read the columns, we will use iloc of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

Let's split our dataset into independent and dependent variables.

y = data[traffic_volume] - independent

x = data.drop(traffic_volume,axis=1)

Splitting The Dataset Into Dependent And Independent Variable

```
y = data['traffic_volume']
x = data.drop(columns=['traffic_volume'],axis=1)
```

```
names = x.columns
```

Feature Scaling

- There is a huge disparity between the x values so let us use feature scaling.
- Feature scaling is a method used to normalize the range of independent variables or features of data.

Feature scaling

```
from sklearn.preprocessing import scale
```

```
x = scale(x)
```

```
x = pd.DataFrame(x,columns=names)
```

```
x.head()
```

	holiday	temp	rain	snow	weather	day	month	year	hours	minutes	seconds
0	0.015856	0.530485	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.345548	0.0	0.0
1	0.015856	0.611467	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.201459	0.0	0.0
2	0.015856	0.627964	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.057371	0.0	0.0
3	0.015856	0.669205	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	0.086718	0.0	0.0
4	0.015856	0.744939	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	0.230807	0.0	0.0

- After scaling the data will be converted into an array form
- Loading the feature names before scaling and converting them back to data frame after standard scaling is applied

Splitting The Data Into Train And Test

- When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test datasets. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
  - The train-test split is a technique for evaluating the performance of a machine learning algorithm.
  - Train Dataset: Used to fit the machine learning model.
  - Test Dataset: Used to evaluate the fit machine learning model.
  - In general you can allocate 80% of the dataset to the training set and the remaining 20% to test.
  - Now split our dataset into train set and test using train_test_split class from sci-kit learn library.

Splitting The Data Into Train And Test

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state =0)
```

- from sklearn import model_selection x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.2,random_state =0)

MODEL BULIDING

The model building includes the following main tasks

- Import the model building Libraries
- Initializing the model
- Training and testing the model
- Evaluation of Model
- Save the Model

Training And Testing The Model

- Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.
- There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are Regression algorithms.

1.Linear Regression

2.Decision Tree Regressor

3.Random Forest Regressor

4.KNN

5.svm

5.xgboost

- Steps in Building the model:-
  - **Initialize the model -**

Initializing the model

```
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
import xgboost
```

  - **Fit the models with x_train and y_train**

Fitting the models with x_train and y_train

```
lin_reg.fit(x_train,y_train)
Dtree.fit(x_train,y_train)
Rand.fit(x_train,y_train)
svr.fit(x_train,y_train)
XGB.fit(x_train,y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

- Predict the y_train values and calculate the accuracy

Predicting the y_train values and calculate the accuracy

```
p1 = lin_reg.predict(x_train)
p2 = Dtree.predict(x_train)
p3 = Rand.predict(x_train)
p4 = svr.predict(x_train)
p5 = XGB.predict(x_train)
```

- We're going to use the x-train and y-train obtained above in the train_test_split section to train our Random forest regression model. We're using the fit method and passing the parameters as shown below.
- We are using the algorithm from Scikit learn library to build the model as shown below,
- Once the model is trained, it's ready to make predictions. We can use the predict method on the model and pass x_test as a parameter to get the output as y_pred.
- Notice that the prediction output is an array of real numbers corresponding to the input array

Model Evaluation

- After training the model, the model should be tested by using the test data which is been separated while splitting the data for checking the functionality of the model.

- **Regression Evaluation Metrics**:

These model evaluation techniques are used to find out the accuracy of R-square_score model built in the Regression type of machine learning models. We have three types of evaluation methods.

- R-square_score
- RMSE – root mean squared error

1. R-squared _score –

Formula

$$R^2 = 1 - \frac{RSS}{TSS}$$

R^2 = coefficient of determination

RSS = sum of squares of residuals

TSS = total sum of squares

Regression Evaluation Metrics

```
from sklearn import metrics
```

R-squared _score

```
print(metrics.r2_score(p1,y_train))
print(metrics.r2_score(p2,y_train))
print(metrics.r2_score(p3,y_train))
print(metrics.r2_score(p4,y_train))
print(metrics.r2_score(p5,y_train))
```

```
-5.517285423636891
1.0
0.9748652589734118
-12.188104231382285
0.8349874938269883
```

```
p1 = lin_reg.predict(x_test)
p2 = Dtree.predict(x_test)
p3 = Rand.predict(x_test)
p4 = svr.predict(x_test)
p5 = XGB.predict(x_test)
```

```
print(metrics.r2_score(p1,y_test))
print(metrics.r2_score(p2,y_test))
print(metrics.r2_score(p3,y_test))
print(metrics.r2_score(p4,y_test))
print(metrics.r2_score(p5,y_test))
```

```
-5.399396398322208
0.6929568578898734
0.8025389977869495
-11.972215715232434
0.7922184852381723
```

- After considering both r squared values of test and train we concluded that random forest regressor is giving the better value, it is able to explain the 97% of the data in train values.
- Random forest gives the best r2-score, so we can select this model.

2. RMSE –Root Mean Square Error

Formula

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \hat{x}_i\right)^2}{N}}$$

RMSD = root-mean-square deviation

i = variable i

N = number of non-missing data points

x_i = actual observations time series

$\hat{x}_i$ = estimated time series

RMSE –Root Mean Square Error

```
MSE = metrics.mean_squared_error(p3,y_test)
```

```
np.sqrt(MSE)
```

799.8784771647161

- RMSE value for Random forest is very less when compared with other models, so saving the Random forest model and deploying using the following process

Save The Model

- After building the model we have to save the model.

- Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions or transport data over the network. wb indicates write method and rd indicates read method.

- This is done by the below code

Saving the Model

```
import pickle
```

```
pickle.dump(Rand,open("model.pkl",'wb'))
pickle.dump(le,open("encoder.pkl",'wb'))
```

Application Building

- In this section, we will be building a web application that is integrated into 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 server-side script

Build HTML Code

- In this HTML page, we will create the front-end part of the web page. On this page, we will accept input from the user and Predict the values.

  - In our project we have HTML files, they are

Spyder (Python 3.12)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Atchyut\Desktop\traffic_volume_project\app.py

app.py | train_model.py | output.html | index.html

```python
import numpy as np
import pickle
import pandas as pd
import os
from flask import Flask, request, render_template

app = Flask(__name__, template_folder='template')

try:
    with open("model.pkl", "rb") as f:
        model = pickle.load(f)
except Exception as e:
    model = None
    print(f"Error loading model: {e}")

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=["POST"])
def predict():
    if not model:
        return render_template("output.html", result="Model not loaded. Please check model.pkl"

    try:
        input_features = [float(x) for x in request.form.values()]
        feature_names = ['holiday', 'temp', 'rain', 'snow', 'weather',
                         'year', 'month', 'day', 'hours', 'minutes', 'seconds']
        data = pd.DataFrame([input_features], columns=feature_names)
        prediction = model.predict(data)[0]
        result_text = f"Estimated Traffic Volume is: {int(prediction)} units"
        return render_template("output.html", result=result_text)
    except Exception as e:
        return render_template("output.html", result=f"Error during prediction: {str(e)}")

if __name__ == "__main__":
    port = int(os.environ.get('PORT', 5000))
    app.run(port=port, debug=True, use_reloader=False)
```

C:\Users\Atchyut

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

Help | Variable Explorer | Plots | Files

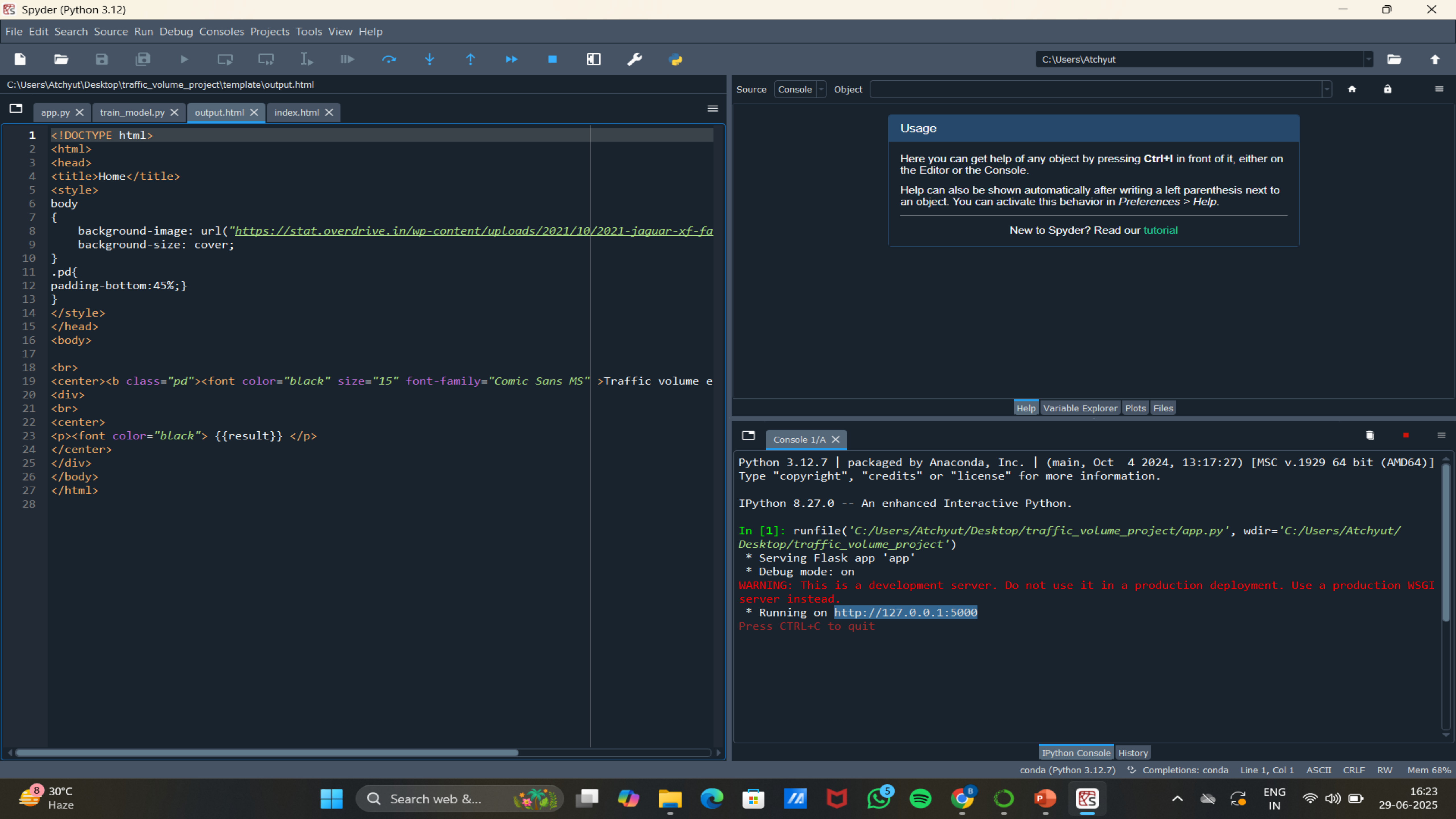
Console 1/A

```
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.27.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Atchyut/Desktop/traffic_volume_project/app.py', wdir='C:/Users/Atchyut/
Desktop/traffic_volume_project')
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

IPython Console | History

conda (Python 3.12.7) | Completions: conda | LSP: Python | Line 44, Col 1 | ASCII | CRLF | RW | Mem 67%

30°C Haze | Search web &... | ENG IN | 16:23 29-06-2025



Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Atchyut
C:\Users\Atchyut\Desktop\traffic_volume_project\template\output.html
app.py
train_model.py
output.html
index.html
<!DOCTYPE html>
<html>
<head>
<title>Home</title>
<style>
body
{
background-image: url("https://stat.overdrive.in/wp-content/uploads/2021/10/2021-jaguar-xf-fa
background-size: cover;
}
.pd{
padding-bottom:45%;}
}
</style>
</head>
<body>

<center><b class="pd"><font color="black" size="15" font-family="Comic Sans MS" >Traffic volume e
<div>

<center>
<p><font color="black"> {{result}} </p>
</center>
</div>
</body>
</html>
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help Variable Explorer Plots Files
Console 1/A
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.27.0 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/Atchyut/Desktop/traffic_volume_project/app.py', wdir='C:/Users/Atchyut/Desktop/traffic_volume_project')
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
IPython Console History
conda (Python 3.12.7) Completions: conda Line 1, Col 1 ASCII CRLF RW Mem 68%
30°C Haze
Search web &...
ENG IN
16:23
29-06-2025

Spyder (Python 3.12)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Atchyut

C:\Users\Atchyut\Desktop\traffic_volume_project\template\index.html

app.py | train_model.py | output.html | index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Traffic Volume Estimation</title>
</head>

<body background="https://cdn.vox-cdn.com/thumbor/voARJfEKvTp6iMSzW3ExPn06TDM=/0x78:3000x1766/1

  <div class="login">
    <center><h1>Traffic Volume Estimation</h1></center>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict') }}" method="post">
      <h1>Please enter the following details</h1>

      <label for="holiday">Holiday:</label>
      <select id="holiday" name="holiday">
        <option value="7">None</option>
        <option value="1">Columbus Day</option>
        <option value="10">Veterans Day</option>
        <option value="9">Thanksgiving Day</option>
        <option value="0">Christmas Day</option>
        <option value="6">New Year's Day</option>
        <option value="11">Washington's Birthday</option>
        <option value="5">Memorial Day</option>
        <option value="2">Independence Day</option>
        <option value="8">State Fair</option>
        <option value="3">Labor Day</option>
        <option value="4">Martin Luther King Jr Day</option>
      </select>
      <br><br>

      <label>Temperature (°C):</label>
      <input type="number" name="temp" placeholder="temp" required><br><br>

      <label>Rain (0 or 1):</label>
      <input type="number" min="0" max="1" name="rain" placeholder="rain" required><br><br>

      <label>Snow (0 or 1):</label>
      <input type="number" min="0" max="1" name="snow" placeholder="snow" required><br><br>

      <label for="weather">Weather:</label>
      <select id="weather" name="weather">
        <option value="1">Clouds</option>
```

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

Help | Variable Explorer | Plots | Files

Console 1/A

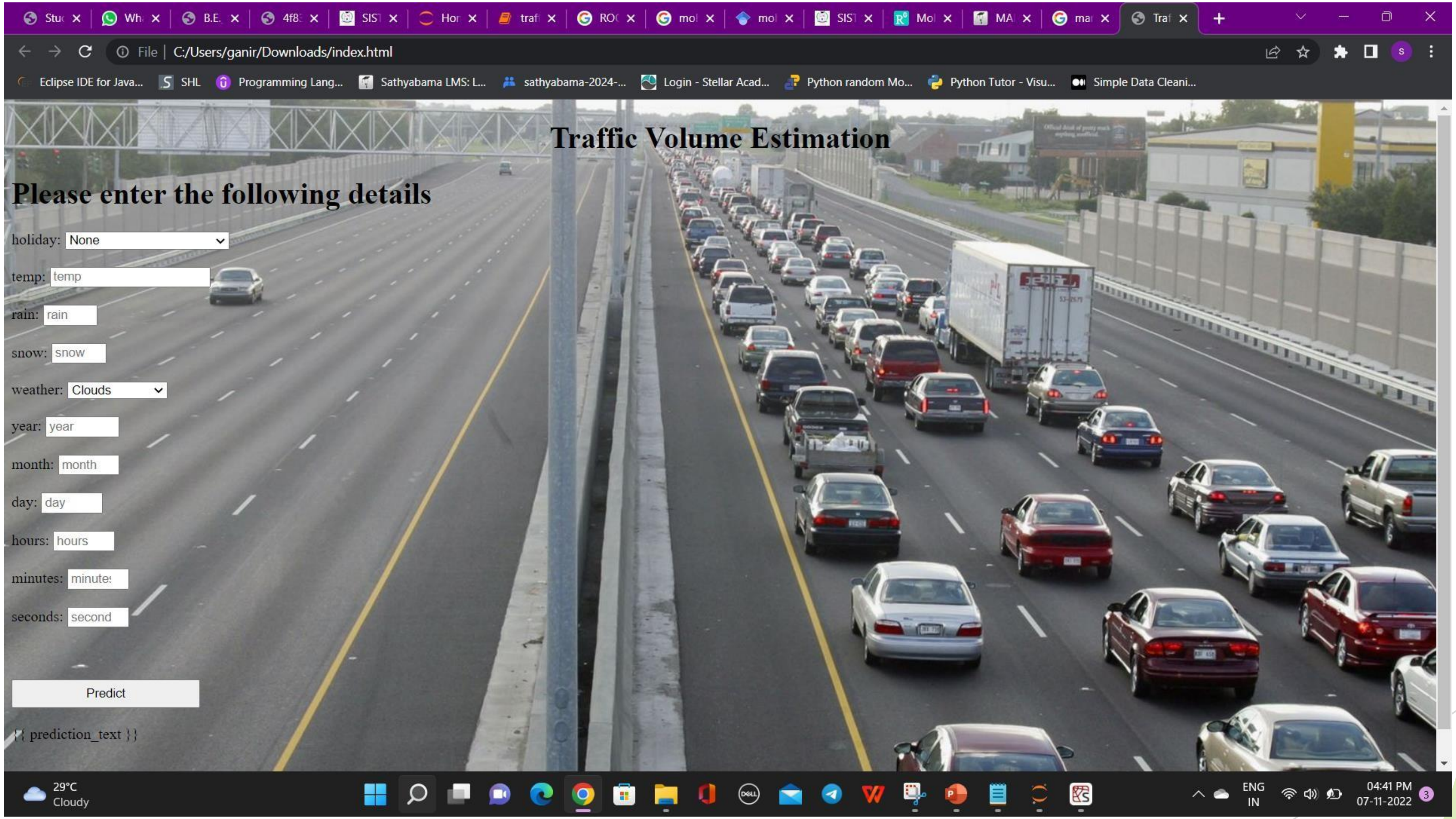
```
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.27.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Atchyut/Desktop/traffic_volume_project/app.py', wdir='C:/Users/Atchyut/
Desktop/traffic_volume_project')
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

IPython Console | History

conda (Python 3.12.7) Completions: conda Line 4, Col 25 UTF-8-GUESSED CRLF RW Mem 67%

30°C Haze Search web &... ENG IN 16:23 29-06-2025

- **The HTML page looks like this-**

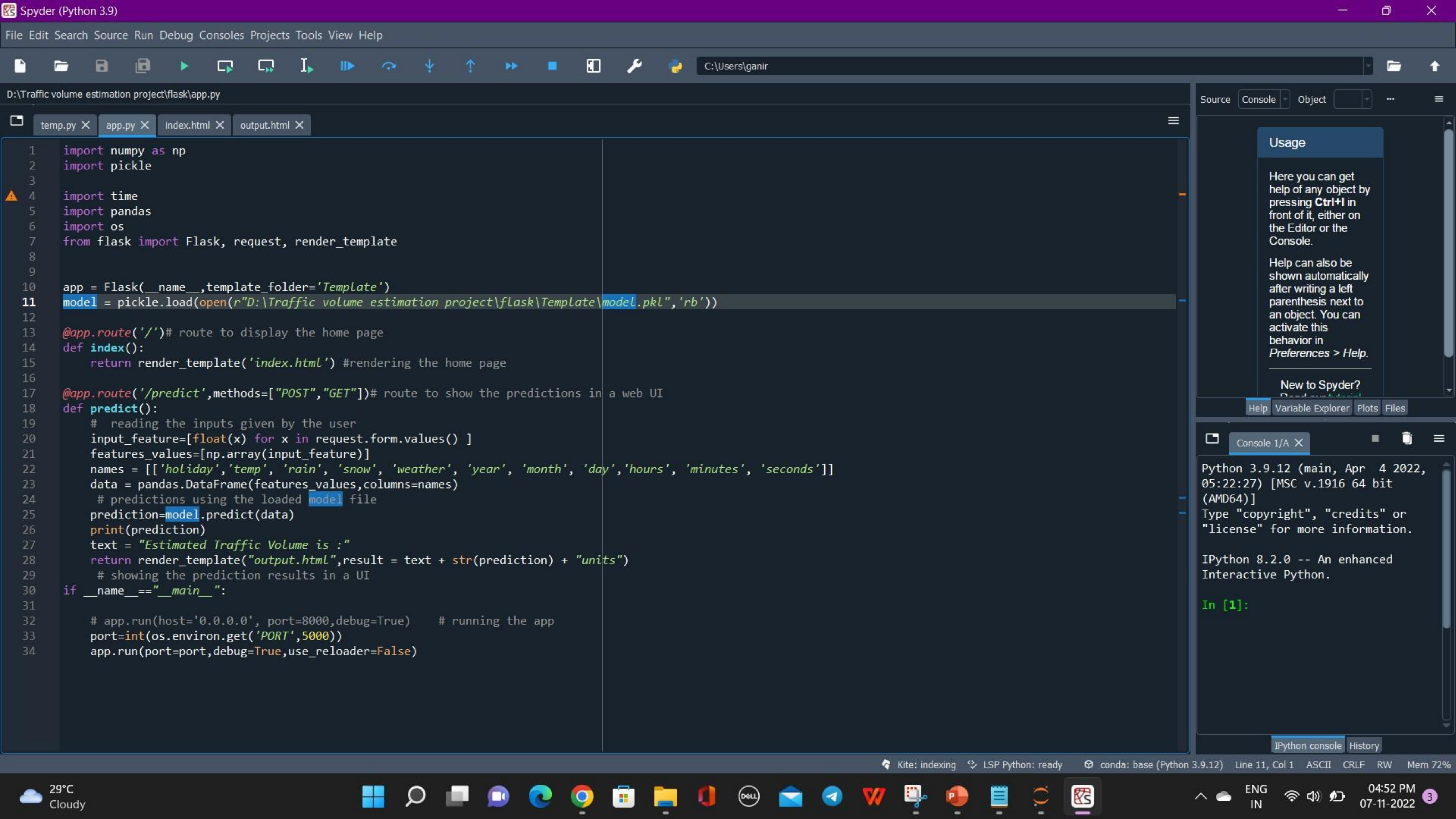


Main Python Script

- Let us build an app.py flask file which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.

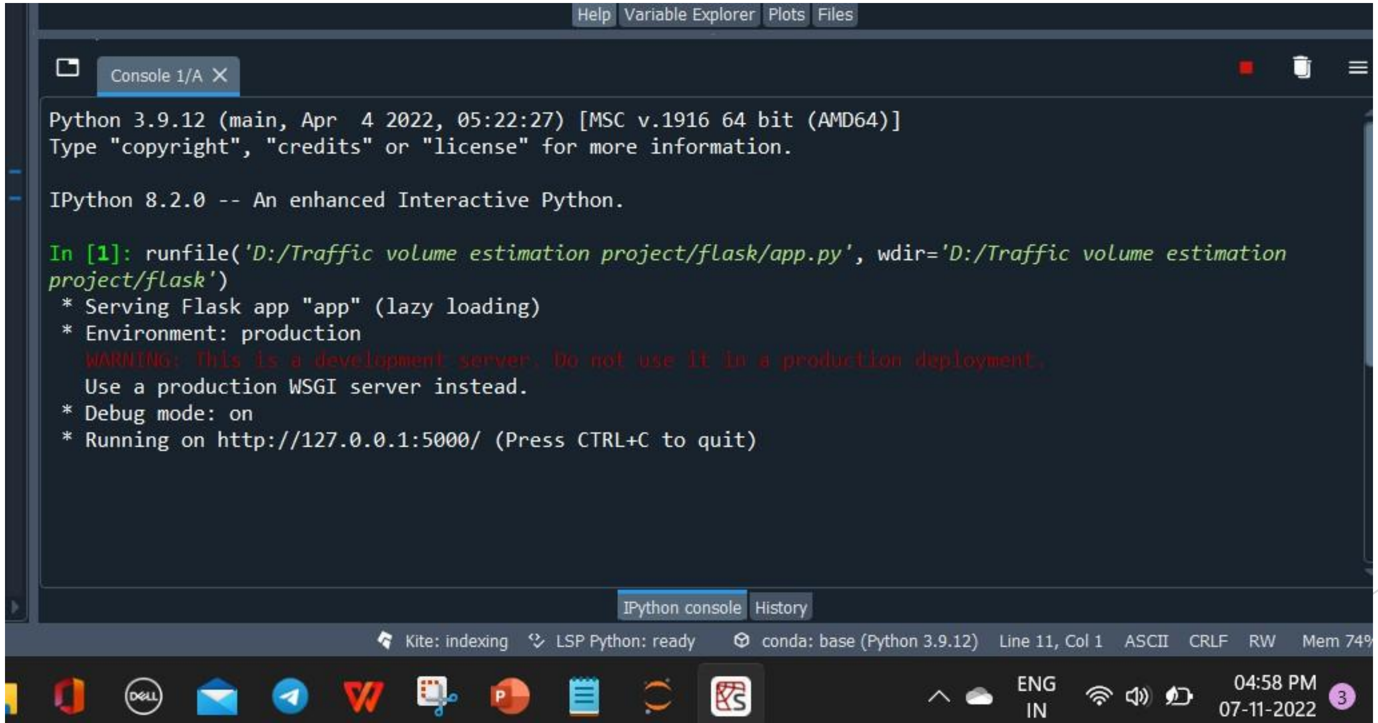
In order to develop web API with respect to our model, we basically use the Flask framework which is written in python.

- Line 1-9 We are importing necessary libraries like Flask to host our model request
- Line 12 Initialise the Flask application
- Line 13 Loading the model using pickle
- Line 16 Routes the API URL
- Line 18 Rendering the template. This helps to redirect to the home page. In this home page,we give our input and ask the model to predict
- In line 23 we are taking the inputs from the form
- Line 28 Feature Scaling the inputs
- Line 31 Predicting the values given by the user
- Line 32-35 if the output is false render no chance template If the output is True render chance template
- Line 36 The value of __name___is set to __main___when the module run as the main program otherwise it is set to the name of the module .


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\ganir
D:\Traffic volume estimation project\flask\app.py
temp.py app.py index.html output.html
import numpy as np
import pickle
import time
import pandas
import os
from flask import Flask, request, render_template
app = Flask(__name__,template_folder='Template')
model = pickle.load(open(r"D:\Traffic volume estimation project\flask\Template\model.pkl",'rb'))
@app.route('/')# route to display the home page
def index():
    return render_template('index.html') #rendering the home page
@app.route('/predict',methods=["POST","GET"])# route to show the predictions in a web UI
def predict():
    # reading the inputs given by the user
    input_feature=[float(x) for x in request.form.values()]
    features_values=[np.array(input_feature)]
    names = [['holiday','temp', 'rain', 'snow', 'weather', 'year', 'month', 'day','hours', 'minutes', 'seconds']]
    data = pandas.DataFrame(features_values,columns=names)
     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    text = "Estimated Traffic Volume is :"
    return render_template("output.html",result = text + str(prediction) + "units")
     # showing the prediction results in a UI
if __name__=="__main__":
    # app.run(host='0.0.0.0', port=8000,debug=True)   # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder?
Help Variable Explorer Plots Files
Console 1/A
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.2.0 -- An enhanced Interactive Python.
In [1]:
IPython console History
Kite: indexing LSP Python: ready conda: base (Python 3.9.12) Line 11, Col 1 ASCII CRLF RW Mem 72%
29°C Cloudy
ENG IN
04:52 PM 07-11-2022

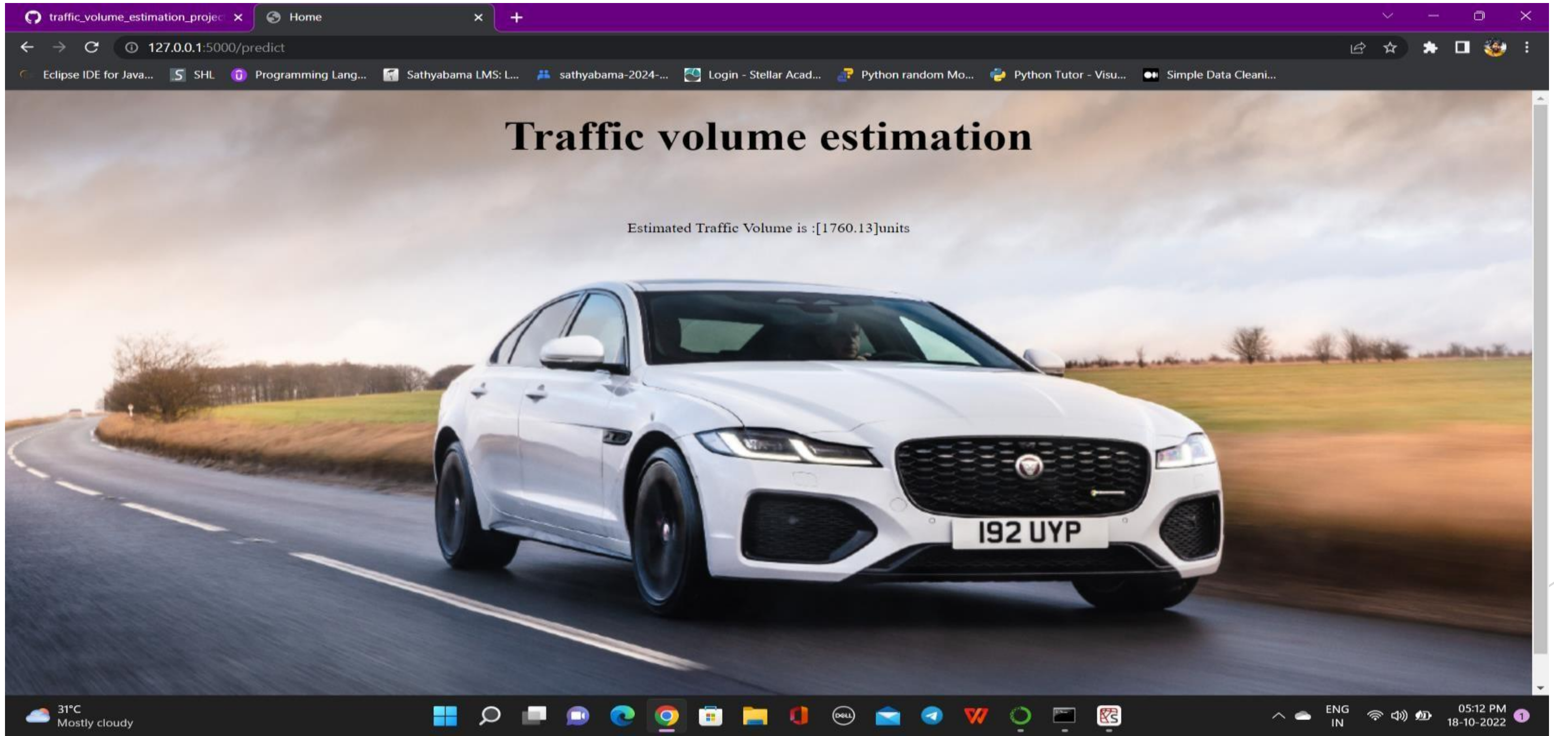
Run The App

- Open anaconda prompt from the start menu
  - Navigate to the folder where your python script is.
  - Now type the “python app.py” command
- Navigate to the localhost where you can view your web page, Then it will run on **local host:5000**



OUTPUT AND RESULTS

- Copy the HTTP link and paste it in google link tab, it will display the form page
- Enter the values as per the form and click on predict button
- It will redirect to the page based on prediction output
- The output will be displayed in the prediction text as Estimated Traffic volume is in units.



REFERENCES

- 1. **Desai et al. Traffic Forecasting for Pavement Design**. Report FHWA-TS-86-225. FHWA, U.S. Department of Transportation, March 1988, p. 121.
- 2. 1. **A. Deacon, J. G. Pigman**, and **J. G. Mayes. Estimation of Equivalent Axleloads**. Kentucky Transportation Research Program, University of Kentucky, Lexington, Dec. 1985, pp. 15- 16.
- 3. **Fatal Accident Reporting System**, 1988. NHTSA, U.S. Department of Transportation, 1988
- 4. **The Clean Air Act of 1990**, Title II, Part A Section 202 (C.3.C). In Environmental Statutes, 1990 ed., Government Institutes, Inc., Maryland.
- 5. **A. N. Johnson. Highway Traffic Capacity**. Public Roads, Vol. 13, No. 3, May 1932.
- 6. **1. Kinzer. Application of the Theory of Probability to Problems of Highway Traffic**. Proc., 5th Annual Meeting of the Institute of Traffic Engineers, ITE, New York, N.Y., 1934.
- 7. **W. F. Adams. Road Traffic Considered as a Random Series**. Journal of the Institution of Civil Engineers, London, 1936.
- 8. **W. A. Shelton. Dispersion of Highway Traffic by Time Periods**: Medium and Small Stations in Farm Area. HRB Proc., 1938.
- 9. **W. A. Shelton. Dispersion of Highway Traffic Volume by Time Periods**. HRB Proc., 1939, pp. 347-348.