# **ABSTRACT**

**Traffic Telligence** is an advanced traffic volume estimation system that leverages the power of machine learning to accurately predict vehicular flow based on environmental and temporal factors. With the rapid urbanization and increasing vehicle population, traffic congestion has become a persistent challenge in modern cities. Accurate forecasting of traffic volume is crucial for smart traffic management, infrastructure planning, and reducing commute times.

This project utilizes historical traffic data with key features such as temperature, rainfall, snowfall, weather conditions, holidays, and timestamps (date and time) to train a robust machine learning model. The model is designed to analyze complex patterns and predict traffic volume for a given set of input conditions. Techniques like data preprocessing, feature engineering, model optimization, and scaling are applied to enhance accuracy and efficiency.

The system is deployed using a user-friendly **Flask web interface**, allowing users to input real-time conditions and receive immediate traffic volume predictions. Visualizations and analytical tools are integrated for better interpretability of results. Traffic Telligence demonstrates how AI can be effectively integrated into urban systems to enable

***Key Words***: Traffic Volume, Random Forest, Machine Learning, RSME, Flask

# **CHAPTER 1. INTRODUCTION**

# **CHAPTER 2. AIM AND SCOPE OF THE PRESENT INVESTIGATION**

# **CHAPTER 3. EXPERIMENTAL OR MATERIALS AND METHODS;**

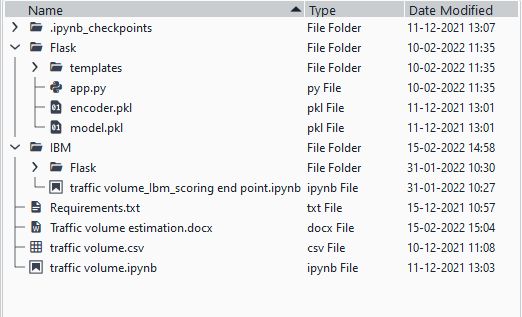
# **ALGORITHMS USED**

# **CHAPTER 4. RESULTS AND DISCUSSION, PERFORMANCE**

**ANALYSIS**

**CHAPTER 5. SUMMARY AND CONCLUSIONS**

**Project Structure**



# 

# **CHAPTER 1. 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. The safe and time-efficient movement of the people and goods is dependent on Traffic flow, which is directly connected to the traffic characteristics. 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.

* 1. ***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, the traffic environment is becoming worse, to unravel this problem and to assist society, we've chosen our topic as traffic volume prediction.

***1.2 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. However, according to the study done by scholars, expanding the road capacity will cause more serious traffic conditions. Therefore, traffic volume prediction is one of the most famous.

# **CHAPTER 2. AIM AND SCOPE OF THE PRESENT INVESTIGATION**

***2.1 AIM:***

We will be using Regression algorithms such as Linear Regression, Decision tree, Random forest, and xgboost to predict the count of traffic volume. We will train and test the data with these algorithms. From this best model is selected and saved in .pkl (Pickle) format.Once the model is saved, we integrate it with flask application and also deploy the model in IBM.

***2.2 SCOPE:***

The objective of this study is to seek out a traffic volume predictor suitable for real implications. This predictor must be accurate in terms of computation cost and power consumption. Within the go after such a predictor, we've included the subsequent contributions: We compare existing schemes to seek out their effectiveness for real-time applications.

# **CHAPTER 3. EXPERIMENTAL OR MATERIALS AND METHODS;**

# **ALGORITHMS USED**

***3.1 Pre Requisites***

To complete this project, you must require the following software’s,  concepts, and packages

* **Anaconda navigator**
* **Packages**   
   Open anaconda prompt as administrator.
* Type “pip install numpy” and click enter
* Type “pip install pandas” and click enter.
* Type “pip install matplotlib” and click enter.
* Type “pip install scikit-learn” and click enter.
* Type “pip install Flask” and click enter.
* Type “pip install xgboost” and click enter.

***3.2 Project Objectives***

By the end of this project:

* You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
* You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
* 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.

***3.3 Project Flow***

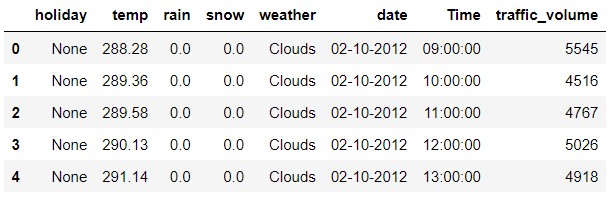
* User interacts with the UI (User Interface) to enter the input values.
* Entered input values are analyzed by the model which is integrated.
* Once the model analyses the input the prediction is showcased on the UI.

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

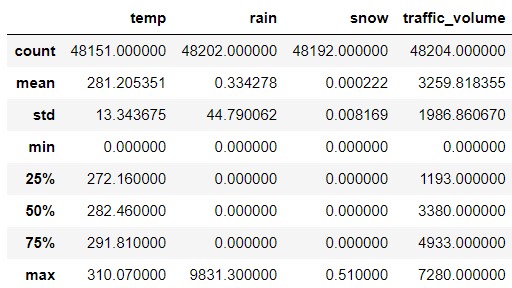
* Data Collection.
  + Collect the dataset or Create the dataset
* Data Pre-processing.
  + Import the Libraries.
  + Importing the dataset.
  + Checking for Null Values.

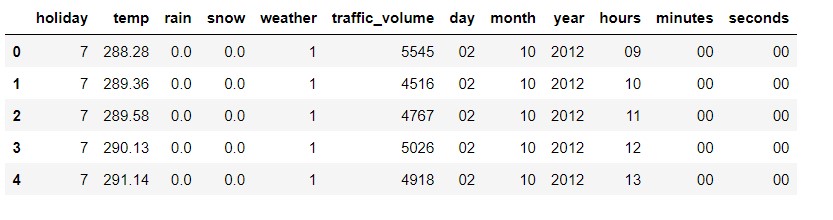
***3.4 Data Collection***

* ML depends heavily on data, without data, it is impossible for an “AI” model to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training **data set**. It is the actual **data set** used to train the model for performing various actions.



**3.1** **DATA HEADER**



 **3.2** **DESCRIBE THE DATA**

**3.4 DATA HEAD AFTER SPLITTING IN DATE AND TIME**

***3.5 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.

# 

# **CHAPTER 4. RESULTS AND DISCUSSION, PERFORMANCE**

**ANALYSIS**

***4.1 Project Structure:***

* Flask files consist of template folder which has HTML pages, app.py file and .pkl files which are used for application building
* IBM folder has flask files and scoring endpoint.ipynb- model training code file.
* We need the model which is saved and the saved model in this content is Traffic volume. Pkl
* Templates folder which contains index.HTML file, chance.HTML file, noChance.HTML file.
* Scale.pkl for scaling, encoder.pkl file for encoding the categorical data, imputer.pkl file for filling out the missing values

***4.2 Importing The Dataset***

* You might have your data in .csv files, .excel files
* Let’s load a .csv data file into pandas usingread\_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”)**
* 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.

***4.3 Handling the Missing values***

1. 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.

2. Check whether any null values are there or not. if it is present then the following can be done.

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

3. 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.

***4.4 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.

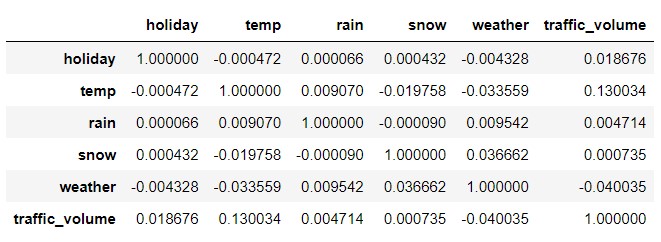
Let’s visualize our data using Matplotlib and seaborn library.  
  
Before diving into the code, let's look at some of the basic properties we will be using when plotting.

**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.

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

**4.1 Correlation of the Data**

**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.

* 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

**4.1 PAIR PLOT**

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

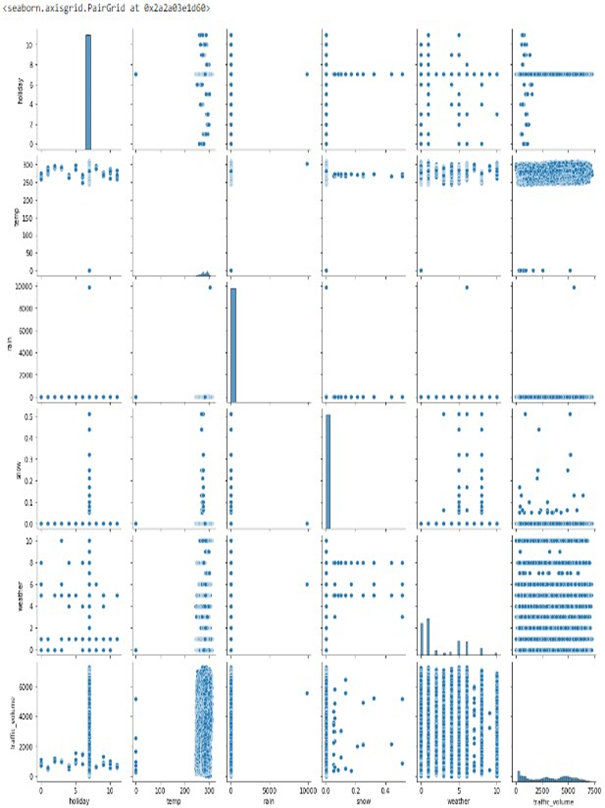
A pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or making a linear separation in our data-set.

* 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.

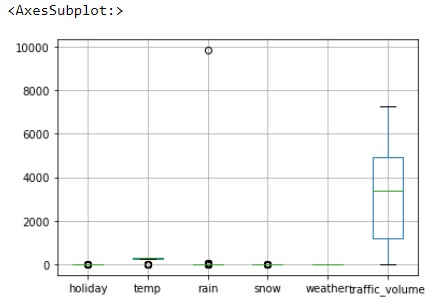
Pair plot usually gives pairwise relationships of the columns in the dataset

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)



**4.2 SEABORN AXIS GRID**

* ****BOXPLOT:**

**4.3** **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 in all the quartiles.

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.

***4.5 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)

### ***Feature Scaling***

**4.1 FEATURE SCALING OF X-GRID**

* 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

***4.7 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.

***4.8 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

We’re going touse 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.

***4.9 Model Evaluation***

Duration: 0.5 Hrs

Skill Tags:

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 models built in the Regression type of machine learning models. We have three types of evaluation methods.
* R-square\_score
* RMSE – root mean squared error
* **R-squared \_score** -

It is the ratio of the number of correct predictions to the total number of input samples. Calculating the r2 score value using for all the models.

* 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.

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.

### ***4.10 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.

# 

# **CHAPTER 5. SUMMARY AND CONCLUSIONS**

***5.1 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.

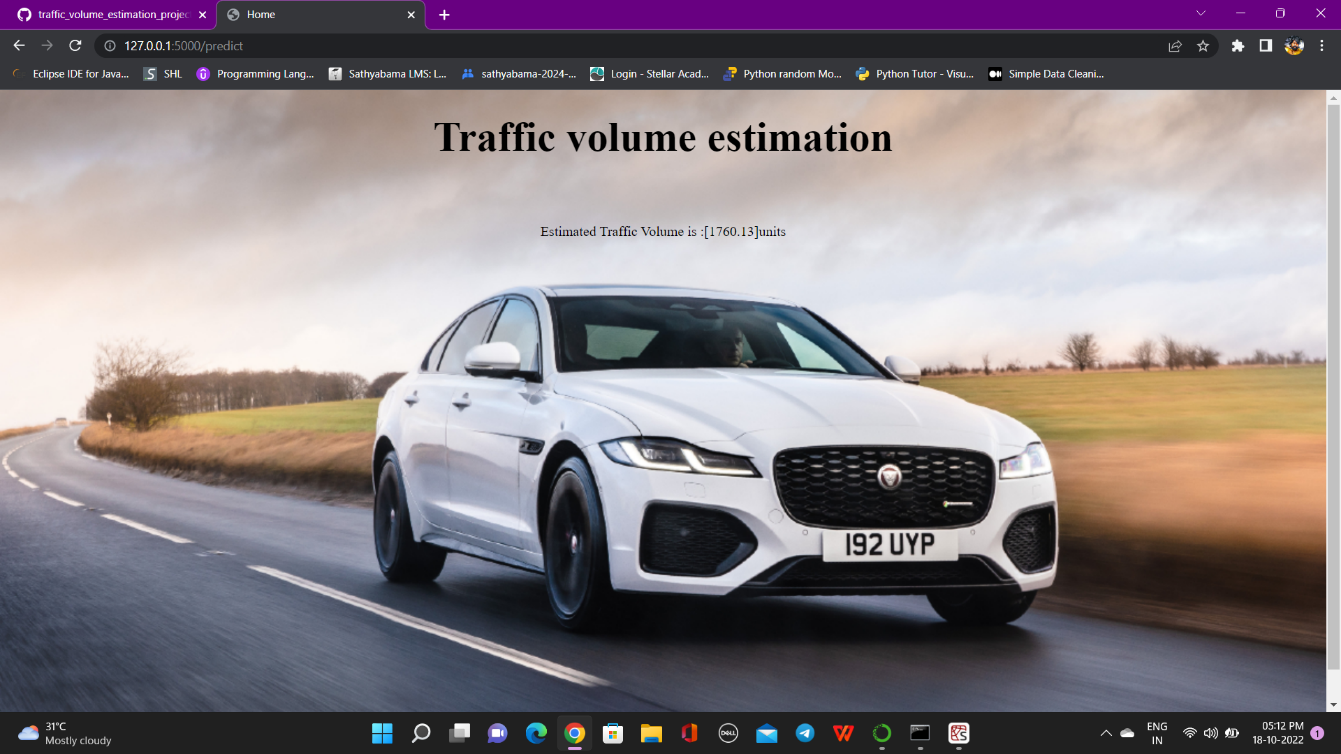
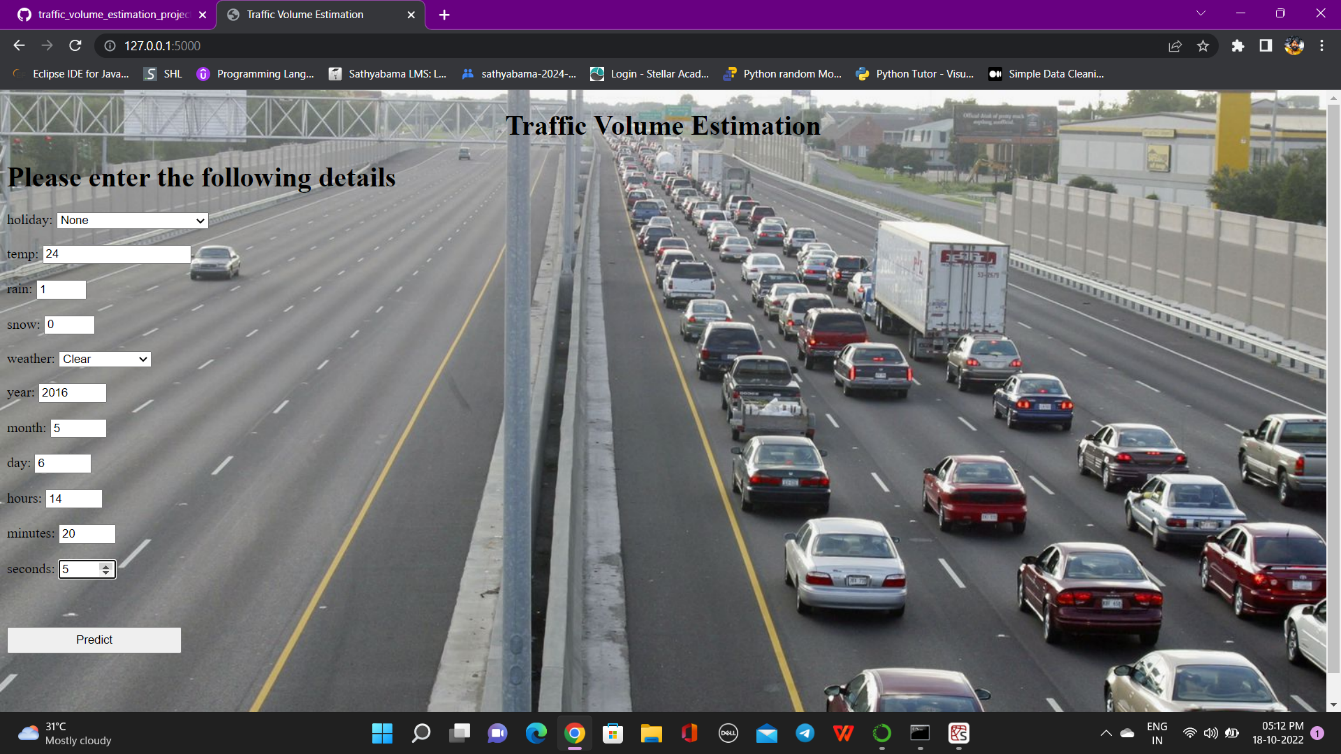
6. 1. Kinzer. Application of the Theory of Probability to Problems of Highway Traffic. Proc., 5th Annual Meeting of the Institute of Traffic Engineers, ITE, New York, N.Y., 1934.

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.

**B. INPUT/OUTPUT :**

****

**B. SOURCE CODE**

* **PYTHON CODE USED IN JUPYTER NOTEBOOK**

# 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

data=pd.read\_csv(r"C:\Users\palag\Downloads\traffic volume data set.csv")

# Analysing the Data

data.head()

data.describe()

data.info()

# Checking the null values

data.isnull().sum()

# Handling the missing values

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

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

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

from collections import Counter

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

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

data.isnull().sum()

# Encoding the data

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

data['weather'] = le.fit\_transform(data['weather'])

data['holiday'] = le.fit\_transform(data['holiday'])

import matplotlib.pyplot as plt

data.head()

sns.pairplot(data)

data.boxplot()

# 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()

# 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

from sklearn.preprocessing import scale

x = scale(x)

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

x.head()

# 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)

# Training And Testing The Model

# Initializing the model

from sklearn import linear\_model

from sklearn import tree

from sklearn import ensemble

from sklearn import svm

import xgboost

# Fitting the models with x\_train and y\_train

lin\_reg = linear\_model.LinearRegression()

Dtree = tree.DecisionTreeRegressor()

Rand = ensemble.RandomForestRegressor()

svr = svm.SVR()

XGB = xgboost.XGBRegressor()

# 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)

# 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)

# 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))

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))

# RMSE –Root Mean Square Error

MSE = metrics.mean\_squared\_error(p3,y\_test)

np.sqrt(MSE)

# Saving the Model

import pickle

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

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

* **PYTHON CODE USED FOR APP BUILDING**
* import numpy as np
* import pickle
* import joblib
* import matplotlib.pyplot as plt
* import time
* import pandas
* import os
* from flask import Flask, request, jsonify, render\_template
* app=Flask(\_\_name\_\_)
* model=pickle.load(open("model.pkl",'rb'))
* scale = pickle.load(open('encoder.pkl','rb'))
* @app.route("/")
* def home():
* return render\_template("home.html")
* @app.route("/predict")
* def predict():
* return render\_template("web\_page.html")
* @app.route("/about")
* def about():
* return render\_template("about.html")
* @app.route("/result",methods=["POST"])
* def result():
* input\_feature=[float(x)for x in request.form.values()]
* features\_values=[np.array(input\_feature[0:11])]
* # features = features\_values[:7]
* # print(features)
* names = [['holiday','temp','rain','snow','weather','year','month','day','hours','minutes','seconds']]
* data = pandas.DataFrame(features\_values, columns=names)
* # data = scale.fit\_transform(data)
* # data = pandas.DataFrame(data, columns = names)
* prediction=model.predict(data)
* print(prediction)
* text = "Estimated Traffic Volume is :"
* return render\_template("result.html" ,prediction\_text = text + str(prediction))
* if \_\_name\_\_=="\_\_main\_\_":
* # port=8000, debug=True)
* app.run(debug=True,use\_reloader=False)
* # \* running the app
* # %%
* # %%
* import os
* os.getcwd()
* # %%

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 .

* **HTML CODES USED**
* **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=/0x78:3000x1766/1600x900/cdn.vox-cdn.com/uploads/chorus\_image/image/44219366/72499026.0.0.jpg" text="black">

<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>

</style></head>

<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 Years Day</option>

<option value=11>Washingtons 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> &nbsp;&nbsp;<br>

<br> <label>temp:</label>

<input type="number" name="temp" placeholder="temp " required="required" /><br>

<br>

<label>rain:</label>

<input type="number" min="0" max="1" name="rain " placeholder="rain" required="required" /><br>

<br>

<label>snow:</label>

<input type="number" min="0" max="1" name="snow " placeholder="snow " required="required" /><br>

<br>

<label for="weather">weather:</label>

<select id="weather" name="weather">

<option value=1>Clouds</option>

<option value=0>Clear</option>

<option value=6>Rain</option>

<option value=2>Drizzle</option>

<option value=5>Mist</option>

<option value=4>Haze</option>

<option value=3>Fog</option>

<option value=10>Thunderstorm</option>

<option value=8>Snow</option>

<option value=9>Squall</option>

<option value=7>Smoke</option><

</select> &nbsp;&nbsp;<br>

<br>

<label>year:</label>

<input type="number" min="2012" max="2022" name="year " placeholder="year " required="required" /><br>

<br>

<label>month:</label>

<input type="number" min="1" max="12" name="month " placeholder="month " required="required" /><br>

<br>

<label>day:</label>

<input type="number" min="1" max="31" name="day " placeholder="day " required="required" /><br>

<br>

<label>hours:</label>

<input type="number" min="0" max="24" name="hours " placeholder="hours " required="required" /><br>

<br>

<label>minutes:</label>

<input type="number" min="0" max="60" name="minutes " placeholder="minutes " required="required" /><br>

<br>

<label>seconds:</label>

<input type="number" min="0" max="60" name="seconds " placeholder="seconds " required="required" /><br>

<br>

<br><br>

<button type="submit" class="btn btn-primary btn-block btn-large" style="height:30px;width:200px">Predict</button>

</form>

<br>

{{ prediction\_text }}

<br>

<br>

<img src="data:image/png;base64,{{url\_3}}" alt="Submit Form" height="180" width="233" onerror="this.style.display='none'"/>

<img src="data:image/png;base64,{{url\_1}}" alt="Submit Form" height="180" width="233" onerror="this.style.display='none'"/>

<img src="data:image/png;base64,{{url\_4}}" alt="Submit Form" height="180" width="233" onerror="this.style.display='none'"/>

<br>

<br>

<img src="data:image/png;base64,{{url\_2}}" alt="Submit Form" height="150" width="711" onerror="this.style.display='none'"/>

</div>

</body>

</html>

* **Output.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-facelift-india-01.jpg");

background-size: cover;

}

.pd{

padding-bottom:45%;}

}

</style>

</head>

<body>

<br>

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

<div>

<br>

<center>

<p><font color="black"> {{result}} </p>

</center>

</div>

</body>

</html>