

Project Development phase

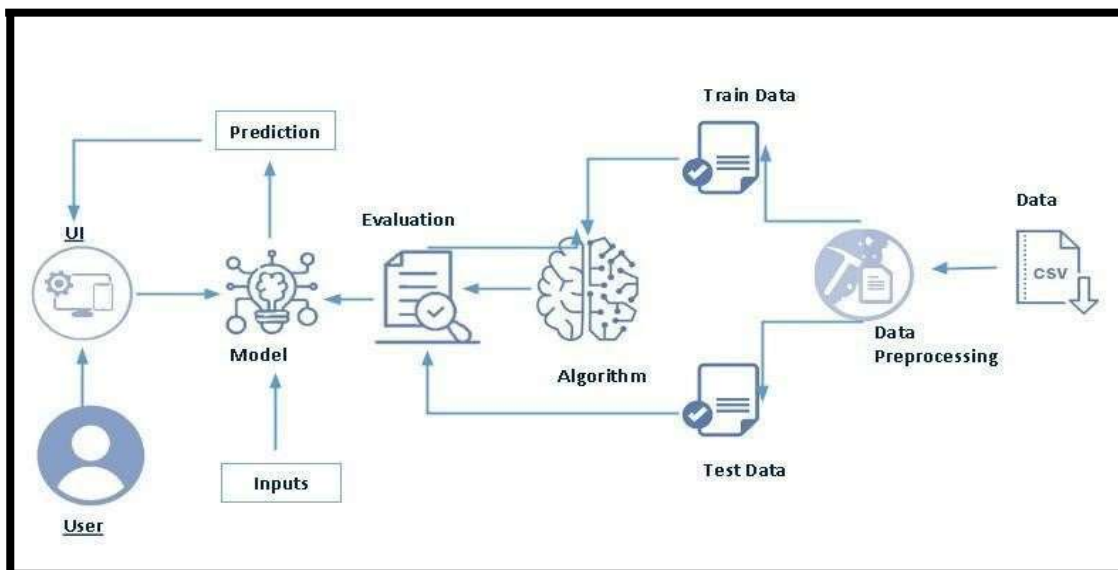
Project Manual

TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning

Team Id: Team - 591643

Traffic problem is one of the major problem now a days, In the increase in no of vehicles and non –usage of public transport leading to traffic related issues, Making a eye on count of traffic at each level enables the government to take the further decisions such as building new roads, increasing infrastructure ,developing multi-channel connectivity .To address such problems to tracking the vehicle count in each and every place AI-ML has given a solution to such kind of traffic related issues, which are able to measure the volume of traffic, identify the violations of traffic rules etc.ML models could give early alerts of severe traffic to help prevent issues related to traffic problems. Hence, there is needs to develop ML algorithms capable in predicting Traffic volume with acceptable level of precision and in reducing the error in the dataset of the projected Traffic volume from model with the expected observable Traffic volume.

Technical Architecture:



Project Flow:

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

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

- Define Problem / Problem Understanding
 - Specify the business problem
 - Business requirements
 - Literature Survey
 - Social or Business Impact.
- Data Collection & Preparation
 - Collect the dataset
 - Data Preparation
- Exploratory Data Analysis
 - Descriptive statistical
 - Visual Analysis
- Model Building
 - Training the model in multiple algorithms
 - Testing the model
- Performance Testing & Hyperparameter Tuning
 - Testing model with multiple evaluation metrics
 - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
 - Save the best model
 - Integrate with Web Framework
- Project Demonstration & Documentation
 - Record explanation Video for project end to end solution
 - Project Documentation-Step by step project development procedure

Project Structure

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Traffic problem is one of the major problem now a days, In the increase in no of vehicles and non –usage of public transport leading to traffic related issues, Making an eye on count of traffic at each level enables the government to take the further decisions such as building new roads, increasing infrastructure ,developing multi-channel connectivity .To address such problems to tracking the vehicle count in each and every place AI-ML has given a solution to such kind of traffic related issues, which are able to measure the volume of traffic, identify the violations of traffic rules etc.ML models could give early alerts of severe traffic to help prevent issues related

to traffic problems. Hence, there is needs to develop ML algorithms capable in predicting Traffic volume with acceptable level of precision and in reducing the error in the dataset of the projected Traffic volume from model with the expected observable Traffic volume.

Activity 2: Business requirements

Traffic volume estimation using machine learning can provide valuable insights to businesses in several ways. Some of the key business requirements that can be addressed through traffic volume estimation using machine learning are:

Resource Planning: Accurate traffic volume estimation can help businesses plan their resources better. By predicting traffic volume in advance, businesses can plan their staffing needs, allocate resources efficiently, and optimize their operations to meet customer demand.

Real-time Monitoring: Businesses require real-time monitoring of traffic data to make informed decisions. This includes monitoring traffic flow, congestion, accidents, weather conditions, and other factors that may impact traffic

Activity 3: Literature Survey

Traffic volume estimation is a crucial component of transportation planning and management. Accurate and reliable estimation of traffic volume is essential for optimizing traffic flow, reducing congestion, and improving safety on roadways. Machine learning (ML) techniques have been widely used in recent years for traffic volume estimation due to their ability to handle complex and large-scale data sets. In this literature survey, we will explore the various ML techniques used for traffic volume estimation.

Activity 4: Social or Business Impact.

Social Impact: -

- **Improved Safety:** Accurate traffic volume estimation can help identify high-risk areas where accidents are more likely to occur, allowing authorities to take appropriate safety measures and reduce the risk of accidents.
- **Reduced Congestion:** Traffic volume estimation can help optimize traffic flow and reduce congestion, which can lead to reduced travel time, improved air quality, and fewer frustrations for drivers.

Business Model/Impact: -

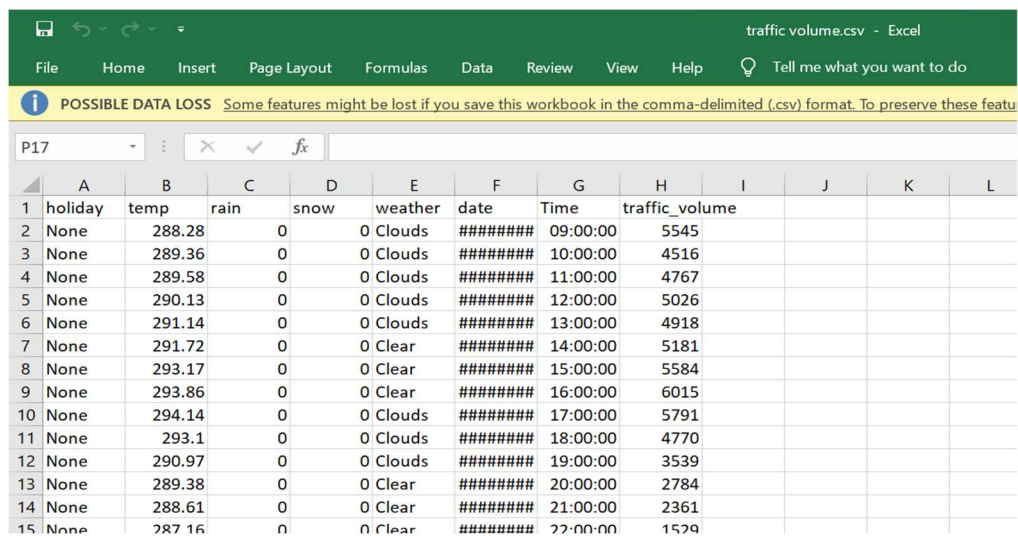
- **Improved Logistics:** Traffic volume estimation can help logistics companies optimize their delivery routes and schedules, reducing transportation costs and improving delivery times.
- **Improved Retail Sales:** Accurate traffic volume estimation can help retailers determine the best location for their stores, as well as the best times to stock their inventory and offer promotions.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project we have used traffic volume.csv data. This data is downloaded from kaggle.com.



	A	B	C	D	E	F	G	H	I	J	K	L
1	holiday	temp	rain	snow	weather	date	Time	traffic_volume				
2	None	288.28	0	0	Clouds	#####	09:00:00	5545				
3	None	289.36	0	0	Clouds	#####	10:00:00	4516				
4	None	289.58	0	0	Clouds	#####	11:00:00	4767				
5	None	290.13	0	0	Clouds	#####	12:00:00	5026				
6	None	291.14	0	0	Clouds	#####	13:00:00	4918				
7	None	291.72	0	0	Clear	#####	14:00:00	5181				
8	None	293.17	0	0	Clear	#####	15:00:00	5584				
9	None	293.86	0	0	Clear	#####	16:00:00	6015				
10	None	294.14	0	0	Clouds	#####	17:00:00	5791				
11	None	293.1	0	0	Clouds	#####	18:00:00	4770				
12	None	290.97	0	0	Clouds	#####	19:00:00	3539				
13	None	289.38	0	0	Clear	#####	20:00:00	2784				
14	None	288.61	0	0	Clear	#####	21:00:00	2361				
15	None	287.16	0	0	Clear	#####	22:00:00	1529				

Activity 1.1: Import Necessary Libraries

Data Collection

```
In [121]: #importing the necessary 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
import xgboost
```

Activity 1.2: Read the Dataset

```
In [123]: #importing the data
data=pd.read_csv(r"C:\Project\IBM\traffic volume.csv")
```

```
In [124]: #displaying first 5 columns of the data
data.head()
```

```
Out[124]:
```

	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

- Our Dataset trafficvolume.csv contains 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 = represent 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 count of traffic

Activity 2: Data Preparation

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Data and time column
- Checking descriptive statistics
- Checking correlation

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

Data Preparation

```
In [125]: #used to display the basic information of the data
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
```

```
In [126]: #used to find shape of the data
data.shape
```

```
Out[126]: (48204, 8)
```

Handling missing values

```
In [127]: #used to display the null values of the data
data.isnull().sum()

Out[127]: holiday            0
temp            53
rain            2
snow           12
weather        49
date            0
Time            0
traffic_volume  0
dtype: int64

In [128]: #filling NaN values with mean, median and mode using fillna() method
data['temp'].fillna(data['temp'].mean(), inplace=True)
data['rain'].fillna(data['rain'].mean(), inplace=True)
data['snow'].fillna(data['snow'].mean(), inplace=True)

In [129]: from typing import Counter
print(Counter(data['weather']))

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

In [130]: data['weather'].fillna('Clouds', inplace=True)
```

Activity 2.2: Handling Data and time column

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

Handling data and time column

```
In [131]: #splitting the date column into year, month, day
data[['day', 'month', 'year']] = data['date'].str.split("-", expand=True)

In [132]: #splitting the time column into year, month, day
data[['hours', 'minutes', 'seconds']] = data['Time'].str.split(":", expand=True)

In [133]: data.drop(columns=['date', 'Time'], axis=1, inplace=True)
data.head()

Out[133]:
```

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
0	None	288.28	0.0	0.0	Clouds	5545	02	10	2012	09	00	00
1	None	289.36	0.0	0.0	Clouds	4516	02	10	2012	10	00	00
2	None	289.58	0.0	0.0	Clouds	4767	02	10	2012	11	00	00
3	None	290.13	0.0	0.0	Clouds	5026	02	10	2012	12	00	00
4	None	291.14	0.0	0.0	Clouds	4918	02	10	2012	13	00	00

Activity 2.3: checking descriptive statistics

```
In [134]: #used to understand the descriptive analysis of the data
data.describe()
```

```
Out[134]:
```

	temp	rain	snow	traffic_volume
count	48204.000000	48204.000000	48204.000000	48204.000000
mean	281.205351	0.334278	0.000222	3259.818355
std	13.336338	44.789133	0.008168	1986.860670
min	0.000000	0.000000	0.000000	0.000000
25%	272.180000	0.000000	0.000000	1193.000000
50%	282.429000	0.000000	0.000000	3380.000000
75%	291.800000	0.000000	0.000000	4933.000000
max	310.070000	9831.300000	0.510000	7280.000000

Activity 2.4: checking correlation

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.

Checking Correlation

```
In [135]: import matplotlib.pyplot as plt
data.corr()
```

C:\Users\nikhi\AppData\Local\Temp\ipykernel_12608\2463821536.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

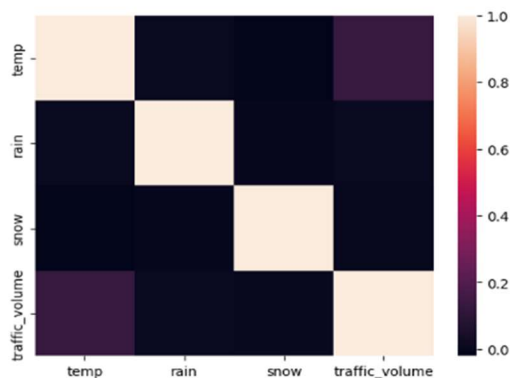
```
Out[135]:
```

	temp	rain	snow	traffic_volume
temp	1.000000	0.009070	-0.019758	0.130034
rain	0.009070	1.000000	-0.000090	0.004714
snow	-0.019758	-0.000090	1.000000	0.000735
traffic_volume	0.130034	0.004714	0.000735	1.000000

```
In [136]: sns.heatmap(data.corr())
```

C:\Users\nikhi\AppData\Local\Temp\ipykernel_12608\1794321463.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
Out[136]: <Axes: >
```



- 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 left side of the above image

Milestone 3: Exploratory Data Analysis

Activity 1: Univariate Analysis

Univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

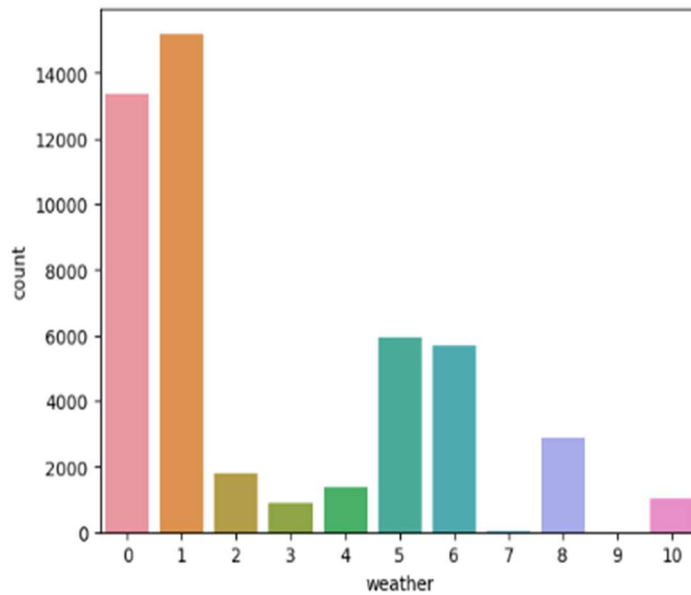
Activity 2: Bivariate Analysis

To find the relation between two features we use bivariate analysis.

Univariate Analysis

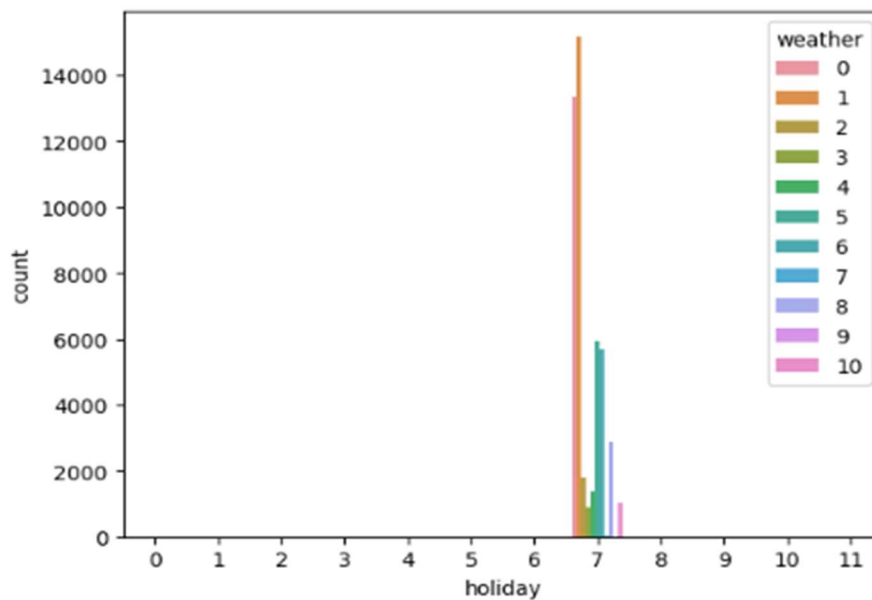
```
In [143]: sns.countplot(x=data['weather'])
```

```
Out[143]: <Axes: xlabel='weather', ylabel='count'>
```



Bivariate Analysis

```
In [144]: sns.countplot(x=data['holiday'], hue=data['weather'])  
plt.show()
```



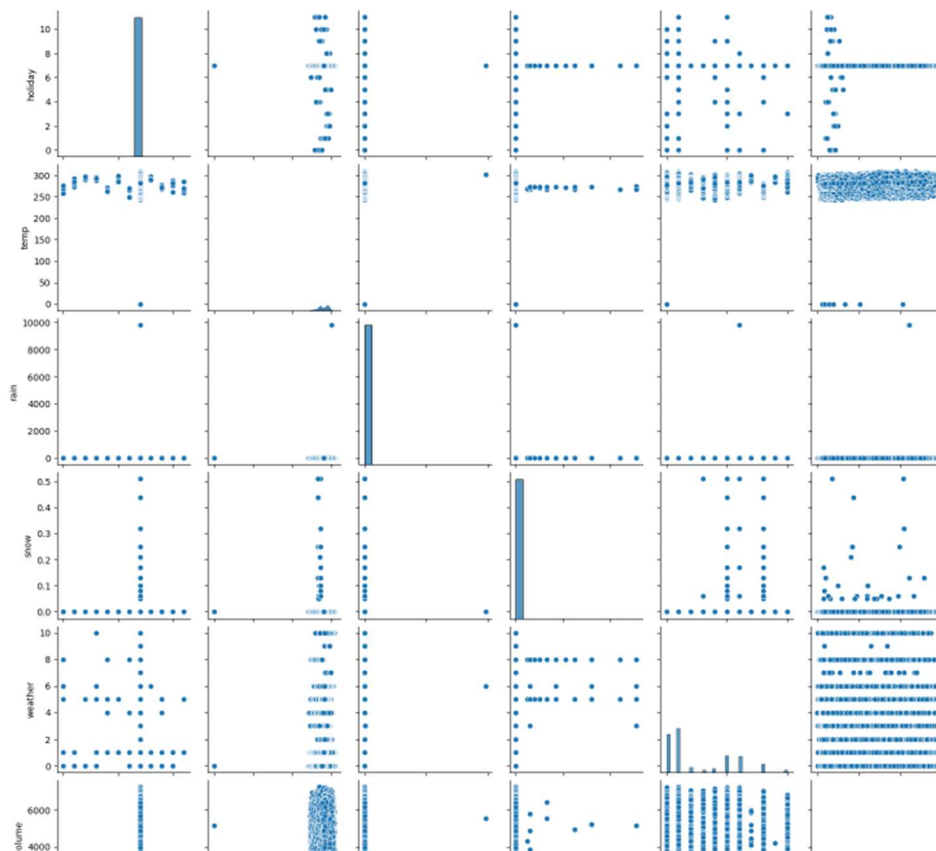
Activity 3: Multivariate Analysis

Pair Plot: Plot pairwise relationships in a dataset.

Multivariate Analysis

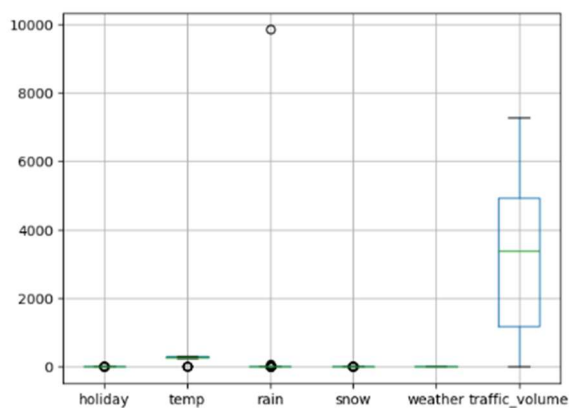
```
In [145]: sns.pairplot(data)
```

```
Out[145]: <seaborn.axisgrid.PairGrid at 0x256816b9210>
```



```
In [146]: data.boxplot()
```

```
Out[146]: <Axes: >
```



Splitting data into X and y

In machine learning, the concept of dependent variable (y) and independent variables(x) is important to understand. Here, Dependent variable is nothing but output in dataset and independent variable is all inputs in the dataset.

y = data[traffic_volume] - independent

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

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data.

Splitting data into X and Y

```
In [147]: y=data['traffic_volume']
In [148]: x=data.drop(columns=['traffic_volume'],axis=1)
In [149]: names=x.columns

In [150]: from sklearn.preprocessing import scale
x=scale(x)
x=pd.DataFrame(x,columns=names)
x.head()

Out[150]:
```

	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 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

- 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 training set and the remaining 20% to test
- Now split our dataset into train set and test using train_test_split class from scikit learn library.

Splitting the data into Train and Test

```
In [151]: 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)
```

Milestone 4: Model Building

Activity 1: Training the 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.svm
- 5.xgboost

Model Building

Training the model

```
In [152]: from sklearn import linear_model
          from sklearn import tree
          from sklearn import ensemble
          from sklearn import svm
          import xgboost

In [153]: lin_reg=linear_model.LinearRegression()
          Dtree=tree.DecisionTreeRegressor()
          Rand=ensemble.RandomForestRegressor()
          svr=svm.SVR()
          XGB=xgboost.XGBRegressor()

In [154]: 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)

Out[154]: XGBRegressor(base_score=None, booster=None, callbacks=None,
                        colsample_bylevel=None, colsample_bynode=None,
                        colsample_bytree=None, device=None, early_stopping_rounds=None,
                        enable_categorical=False, eval_metric=None, feature_types=None,
                        gamma=None, grow_policy=None, importance_type=None,
                        interaction_constraints=None, learning_rate=None, max_bin=None,
                        max_cat_threshold=None, max_cat_to_onehot=None,
                        max_delta_step=None, max_depth=None, max_leaves=None,
                        min_child_weight=None, missing=nan, monotone_constraints=None,
                        multi_strategy=None, n_estimators=None, n_jobs=None,
                        num_parallel_tree=None, random_state=None, ...)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [165]: 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)
```

Activity 2: Testing the model

Testing the model

```
In [156]: 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)

In [157]: from sklearn import metrics
          from sklearn.metrics import classification_report

In [158]: !pip install --upgrade scikit-learn

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in c:\users\nikhi\appdata\roaming\python\python311\site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\nikhi\appdata\roaming\python\python311\site-packages (from scikit-learn) (1.26.2)
Requirement already satisfied: scipy>=1.5.0 in c:\users\nikhi\appdata\roaming\python\python311\site-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)

In [159]: 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))

-5.399396398322171
0.696246876366159
0.8042583868615718
-11.972215715232423
0.8065613781045625
```

Milestone 5: Performance Testing & Hyperparameter Tuning

After predicting we will find the r-square value of each model

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses.

```
from sklearn import metrics

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))

-5.517285423636865
1.0
0.9747969952887571
-12.188104231382285
0.8349874938269883
```

Predict the y_test values and calculate the r-square value.

After considering the 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

We're going to use x_train and y_train obtained above in train_test_split section to train our Random forest regression model. We're using the fit method and passing the parameters as shown below.

Activity 2: 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 models built in Regression type of machine learning models. We have three types of evaluation methods.

- R-square_score
- RMSE – root mean squared error

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

It is the ratio of number of correct predictions to the total number of input samples.

```

from sklearn import metrics

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))

-5.517285423636865
1.0
0.9747969952887571
-12.188104231382285
0.8349874938269883

```

Select the model, which gives the best accuracy of all, and generate predictions and find the accuracy with training and testing data

1. RMSD –Root Mean Square deviation

Formula

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

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

Randomforest gives best r-score value ¶

```

In [160]: MSE=metrics.mean_squared_error(p3,y_test)

In [161]: np.sqrt(MSE)

Out[161]: 796.5192672088438

```

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

Milestone 6: Model Deployment

Activity 1: 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

```

In [162]: import pickle

In [163]: pickle.dump(Rand,open("model.pkl","wb"))
           pickle.dump(le,open("encoder.pkl","wb"))

```

Activity 2: Integrate with Web Framework

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

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Build HTML Code

index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Traffic Volume Estimation</title>
7   </head>
8   <body background="C:\Users\nikhi\Downloads\traffic jam.jpeg" text="black">
9     <div class="Login">
10      <center><h1>Traffic Volume Estimation</h1></center>
11      <form action="{{ url_for('predict')}}" method="post">
12        <h1>Please enter the following details</h1>
13
14      </style></head>
15
16      <label for="holiday">holiday:</label>
17      <select id="holiday" name="holiday">
18        <option value=7>None</option>
19        <option value=1>Columbus Day</option>
20        <option value=10>Veterans Day</option>
21        <option value=9>Thanksgiving Day</option>
22        <option value=0>Christmas day</option>
23        <option value=6>New Years Day</option>
24        <option value=11>Washingtons Birthday</option>
25        <option value=5>Memorial Day</option>
26        <option value=2>Independence day</option>
27        <option value=8>State Fair</option>
28        <option value=3>Labor Day</option>
29        <option value=4>Martin Luther King Jr Day</option>
```

```

57 <br> <label>temp:</label>
58 <input type="number" name="temp" placeholder="temp" required="required" /><br>
59 <label>rain:</label>
60 <input type="number" min="0" max="1" name="rain" placeholder="rain" required="required"/> <br>
61 <br>
62 <label>snow:</label>
63 <br>
64 <label>snow:</label>
65 <input type="number" min="0" max="1" name="snow " placeholder="snow " required="required" /><br>
66 <br>
67 <label for="weather">weather:</label>
68 <select id="weather" name="weather">
69 <option value=1>Clouds</option>
70 <option value=0>Clear</option>
71 <option value=6>Rain</option>
72 <option value=2>Drizzle</option>
73 <option value=5>Mist</option>
74 <option value=4>Haze</option>
75 <option value=3>Fog</option>
76 <option value=10>Thunderstorm</option>
77 <option value=8>Snow</option>
78 <option value=9>Squall</option>
79 <option value=7>Smoke</option>
80 </select> &nbsp;&nbsp;&nbsp;<br>
81 <br>
82 <label>year:</label>
83 <input type="number" min="2012" max="2022" name="year " placeholder="year " required="required" /><br>
84 <br>
85 <label>year:</label>
86 <input type="number" min="2012" max="2022" name="year " placeholder="year " required="required" /><br>
87 <br>
88 <label>month:</label>
89 <input type="number" min="1" max="12" name="month " placeholder="month" required="required" /><br>
90 <br>
91 <label>day:</label>
92 <input type="number" min="1" max="31" name="day " placeholder="day" required="required" /><br>
93 <br>
94 <label>hours:</label>
95 <input type="number" min="0" max="24" name="hours " placeholder="hours " required="required" /><br>
96 <br>
97 <label>minutes:</label>
98 <input type="number" min="0" max="60" name="minutes " placeholder="minutes " required="required" /><br>
99 <br>
100 <label>seconds:</label>
101 <input type="number" min="0" max="60" name="seconds " placeholder="seconds " required="required" /><br>
102 <br>
103 <br><br>
104 <button type="submit" class="btn btn-primary btn-block btn-Large" style="height:30px;width:200px">Predict</button>
105 </form>
106 <br>
107 <div>
108 <div>
109 <div>
110 <div>
111 <div>
112 <div>
113 <div>
114 <div>
115 <div>
116 <div>
117 <div>
118 <div>
119 <div>
120 <div>
121 <div>
122 <div>
123 <div>
124 <div>
125 <div>
126 <div>
127 <div>
128 <div>
129 <div>
130 <div>
131 <div>
132 <div>
133 <div>
134 <div>
135 <div>
136 <div>
137 <div>
138 <div>
139 <div>
140 <div>
141 <div>
142 <div>
143 <div>
144 <div>
145 <div>
146 <div>
147 <div>
148 <div>
149 <div>
150 <div>
151 <div>
152 <div>
153 <div>
154 <div>
155 <div>
156 <div>
157 <div>
158 <div>
159 <div>
160 <div>
161 <div>
162 <div>
163 <div>
164 <div>
165 <div>
166 <div>
167 <div>
168 <div>
169 <div>
170 <div>
171 <div>
172 <div>
173 <div>
174 <div>
175 <div>
176 <div>
177 <div>
178 <div>
179 <div>
180 <div>
181 <div>
182 <div>
183 <div>
184 <div>
185 <div>
186 <div>
187 <div>
188 <div>
189 <div>
190 <div>
191 <div>
192 <div>
193 <div>
194 <div>
195 <div>
196 <div>
197 <div>
198 <div>
199 <div>
200 <div>
201 <div>
202 <div>
203 <div>
204 <div>
205 <div>
206 <div>
207 <div>
208 <div>
209 <div>
210 <div>
211 <div>
212 <div>
213 <div>
214 <div>
215 <div>
216 <div>
217 <div>
218 <div>
219 <div>
220 <div>
221 <div>
222 <div>
223 <div>
224 <div>
225 <div>
226 <div>
227 <div>
228 <div>
229 <div>
230 <div>
231 <div>
232 <div>
233 <div>
234 <div>
235 <div>
236 <div>
237 <div>
238 <div>
239 <div>
240 <div>
241 <div>
242 <div>
243 <div>
244 <div>
245 <div>
246 <div>
247 <div>
248 <div>
249 <div>
250 <div>
251 <div>
252 <div>
253 <div>
254 <div>
255 <div>
256 <div>
257 <div>
258 <div>
259 <div>
260 <div>
261 <div>
262 <div>
263 <div>
264 <div>
265 <div>
266 <div>
267 <div>
268 <div>
269 <div>
270 <div>
271 <div>
272 <div>
273 <div>
274 <div>
275 <div>
276 <div>
277 <div>
278 <div>
279 <div>
280 <div>
281 <div>
282 <div>
283 <div>
284 <div>
285 <div>
286 <div>
287 <div>
288 <div>
289 <div>
290 <div>
291 <div>
292 <div>
293 <div>
294 <div>
295 <div>
296 <div>
297 <div>
298 <div>
299 <div>
300 <div>
301 <div>
302 <div>
303 <div>
304 <div>
305 <div>
306 <div>
307 <div>
308 <div>
309 <div>
310 <div>
311 <div>
312 <div>
313 <div>
314 <div>
315 <div>
316 <div>
317 <div>
318 <div>
319 <div>
320 <div>
321 <div>
322 <div>
323 <div>
324 <div>
325 <div>
326 <div>
327 <div>
328 <div>
329 <div>
330 <div>
331 <div>
332 <div>
333 <div>
334 <div>
335 <div>
336 <div>
337 <div>
338 <div>
339 <div>
340 <div>
341 <div>
342 <div>
343 <div>
344 <div>
345 <div>
346 <div>
347 <div>
348 <div>
349 <div>
350 <div>
351 <div>
352 <div>
353 <div>
354 <div>
355 <div>
356 <div>
357 <div>
358 <div>
359 <div>
360 <div>
361 <div>
362 <div>
363 <div>
364 <div>
365 <div>
366 <div>
367 <div>
368 <div>
369 <div>
370 <div>
371 <div>
372 <div>
373 <div>
374 <div>
375 <div>
376 <div>
377 <div>
378 <div>
379 <div>
380 <div>
381 <div>
382 <div>
383 <div>
384 <div>
385 <div>
386 <div>
387 <div>
388 <div>
389 <div>
390 <div>
391 <div>
392 <div>
393 <div>
394 <div>
395 <div>
396 <div>
397 <div>
398 <div>
399 <div>
400 <div>
401 <div>
402 <div>
403 <div>
404 <div>
405 <div>
406 <div>
407 <div>
408 <div>
409 <div>
410 <div>
411 <div&gt
```

Activity 2.2: Build Python code:

Import the libraries


```

1  import numpy as np
2  import pickle
3  import joblib
4  import matplotlib
5  import matplotlib.pyplot as plt
6  import time
7  import pandas
8  import os
9  from flask import Flask, request, jsonify, render_template
10
11 app=Flask(__name__)
12 model=pickle.load(open(r"C:\Users\nikhi\Desktop\project deployment\model.pkl", 'rb'))
13 @app.route('/')
14 def home():
15     return render_template('index.html')
16 @app.route('/predict', methods=["POST", "GET"])
17 def predict():
18     input_feature=[float(x) for x in request.form.values()]
19     features_values=np.array(input_feature)
20     names=[['holiday', 'temp', 'rain', 'snow', 'weather', 'year', 'month', 'day', 'hours', 'minutes', 'seconds']]
21     data=pandas.DataFrame(features_values, columns=names)
22
23     prediction=model.predict(data)
24     print(prediction)
25     text="Estimates Traffic Volume is :"
26     return render_template("index.html", prediction_text= text + str(prediction))
27 if __name__=="__main__":
28     port=int(os.environ.get('PORT', 5000))
29     app.run(port=port, debug=True, use_reloader=False)

```

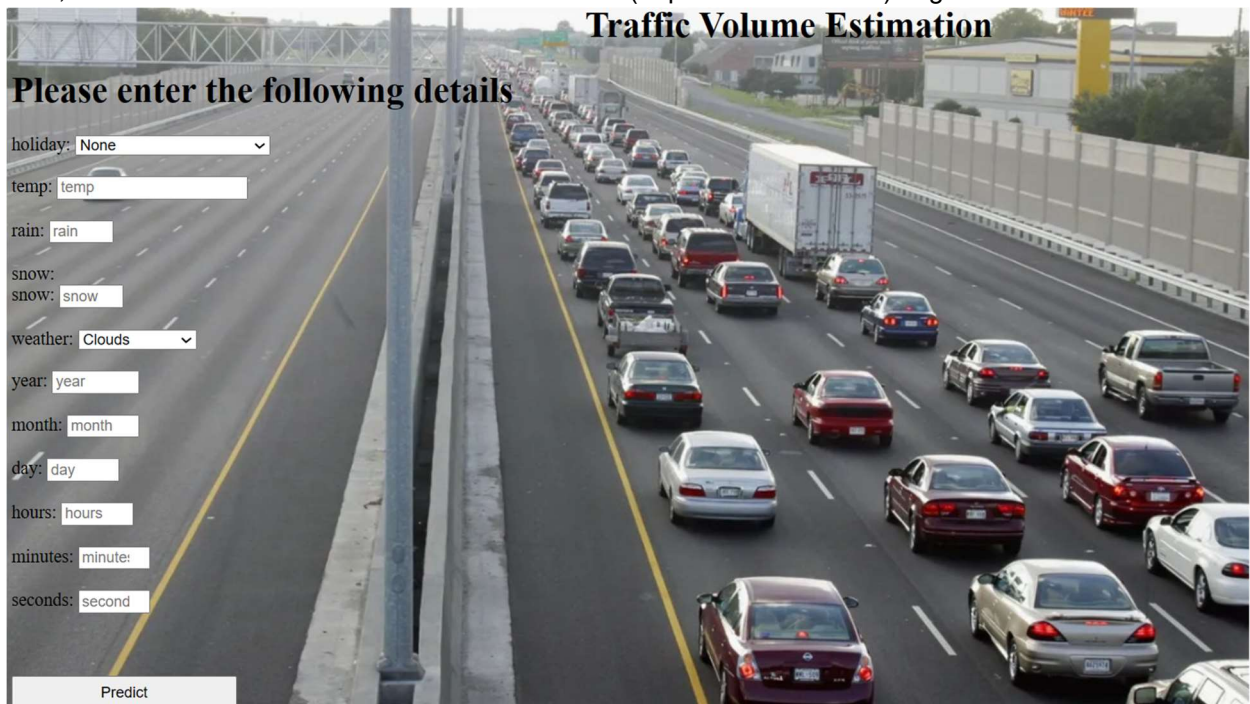
Activity 2.3: Run the web application

```

PS C:\Users\nikhi\Desktop\project deployment> c:: cd 'c:\Users\nikhi\Desktop\project deployment'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\nikhi\.vscode\extensions\ms-python.python-2023.22.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55715' '--' 'c:\Users\nikhi\Desktop\project deployment\ap deployment\app.py'
* 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

```

Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result



Traffic Volume Estimation

Please enter the following details

holiday:

temp:

rain:

snow:

weather:

year:

month:

day:

hours:

minutes:

seconds:

It will display all the input parameters and the prediction text will display the output value of the data given by user.