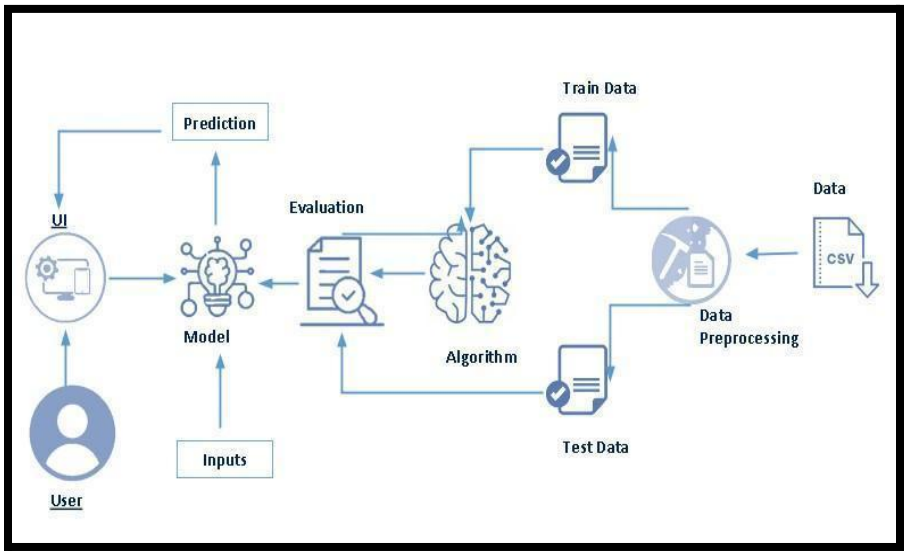
TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning

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 mutli-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,

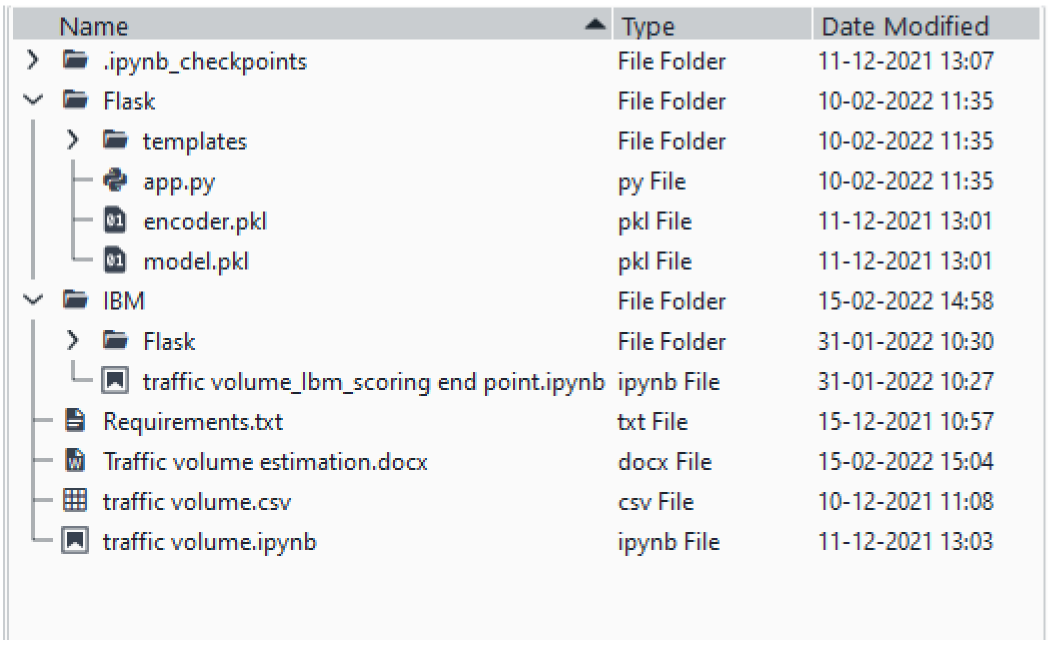
- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- 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

Prior Knowledge

• ML Concepts o Supervised learning:
https://www.javatpoint.com/supervised-machine-learning o Unsupervised learning:
https://www.javatpoint.com/unsupervised-machine-learning
• Decision tree:
https://www.javatpoint.com/machine-learning-decision-tree-classificationalgorithm
• Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
• KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
• Xgboost:
https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-tounderstand-the-math-behind-boost/
• Evaluation metrics:
https://www.analyticsvidhya.com/blog/2019/08/11-important-modelevaluation-error-metrics/
• Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure

Create the Project folder which contains files as shown below



Name	Type	Date Modified
.ipynb_checkpoints	File Folder	11-12-2021 13:07
Flask	File Folder	10-02-2022 11:35
templates	File Folder	10-02-2022 11:35
app.py	py File	10-02-2022 11:35
encoder.pkl	pkl File	11-12-2021 13:01
model.pkl	pkl File	11-12-2021 13:01
IBM	File Folder	15-02-2022 14:58
Flask	File Folder	31-01-2022 10:30
traffic volume_Ibm_scoring end point.ipynb	ipynb File	31-01-2022 10:27
Requirements.txt	txt File	15-12-2021 10:57
Traffic volume estimation.docx	docx File	15-02-2022 15:04
traffic volume.csv	csv File	10-12-2021 11:08
traffic volume.ipynb	ipynb File	11-12-2021 13:03

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

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 (Student Will Write)

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 .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link:

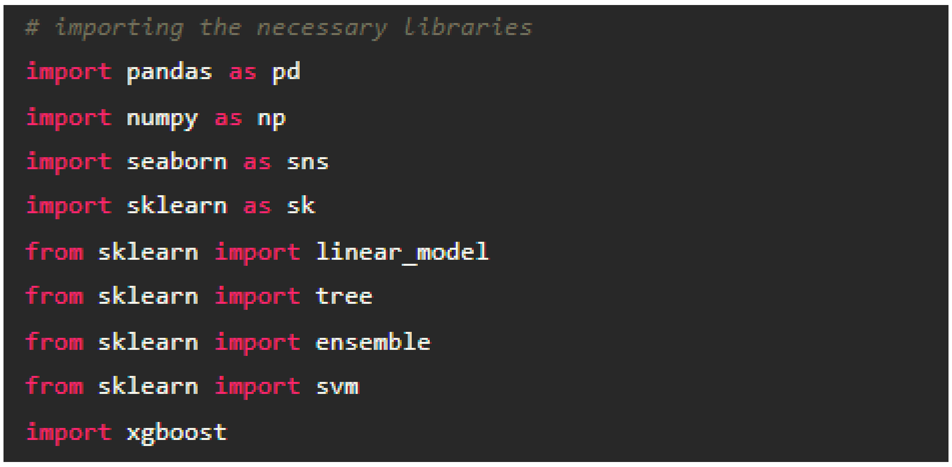
Please refer to the link given below to download the data set and to know about the dataset :

https://drive.google.com/file/d/1iV5PfYAmI6YP0_0S4KYy1ZahHOqMgDbM/view

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a few techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Import Necessary Libraries

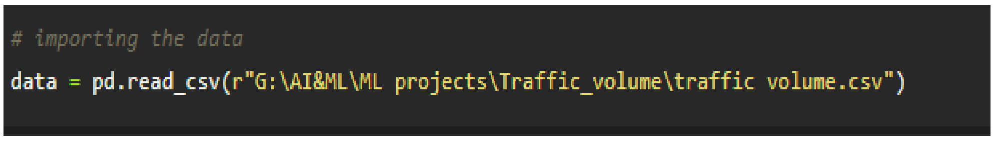


```
# importing the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
import xgboost
```

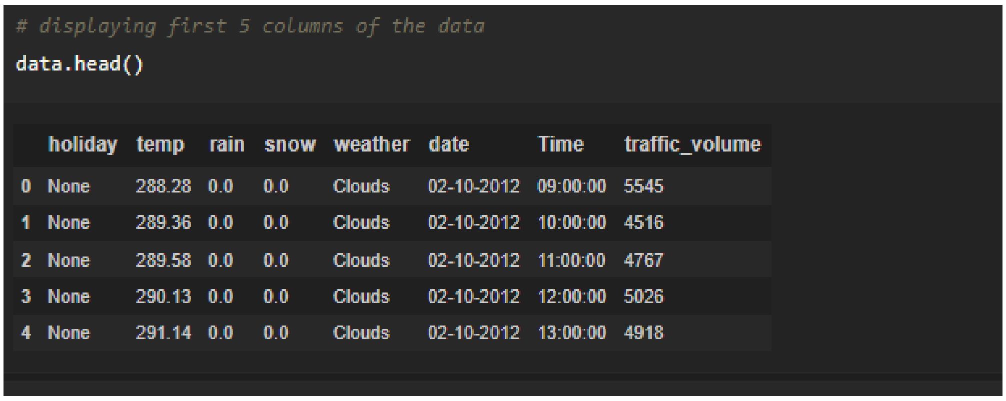
Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

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



```
# importing the data
data = pd.read_csv(r"G:\AI&ML\ML projects\Traffic_volume\traffic volume.csv")
```



```
# displaying first 5 columns of the data
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
```

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 in same directory of your program, you can directly read it, without giving raw as r.

- Our Dataset weatherAus.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 = 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 count of traffic

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

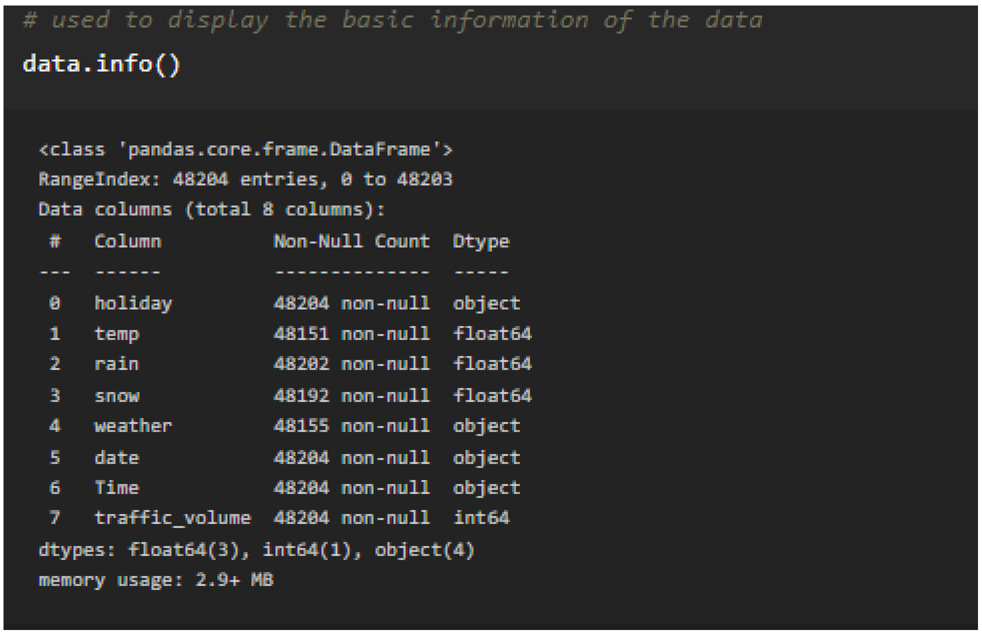
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- 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

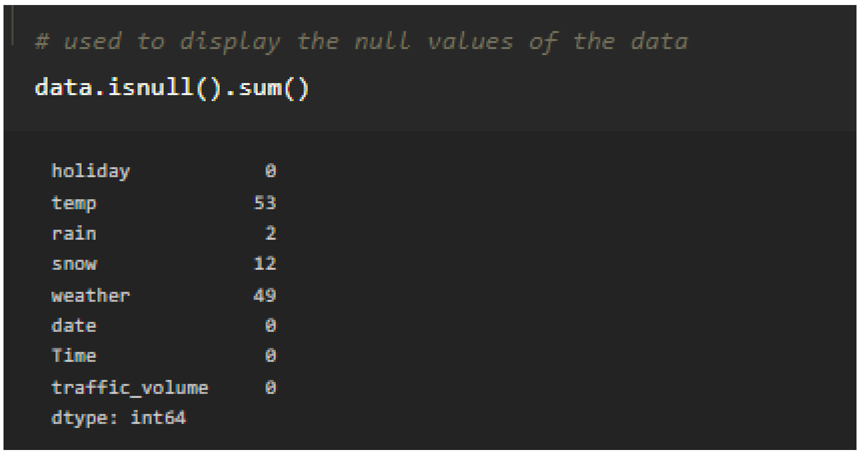
- Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.



```
# 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
```

- For checking the null values, df.isnull() function is used. To sum those null values we use. sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

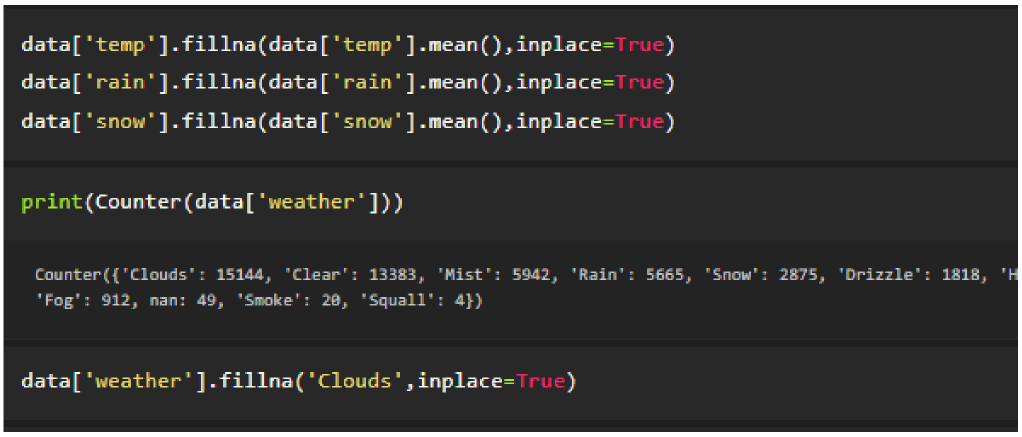


```
# used to display the null values of the data
data.isnull().sum()
```

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

Imputing data using Imputation method in sklearn.Simpleimputer

a.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)
```

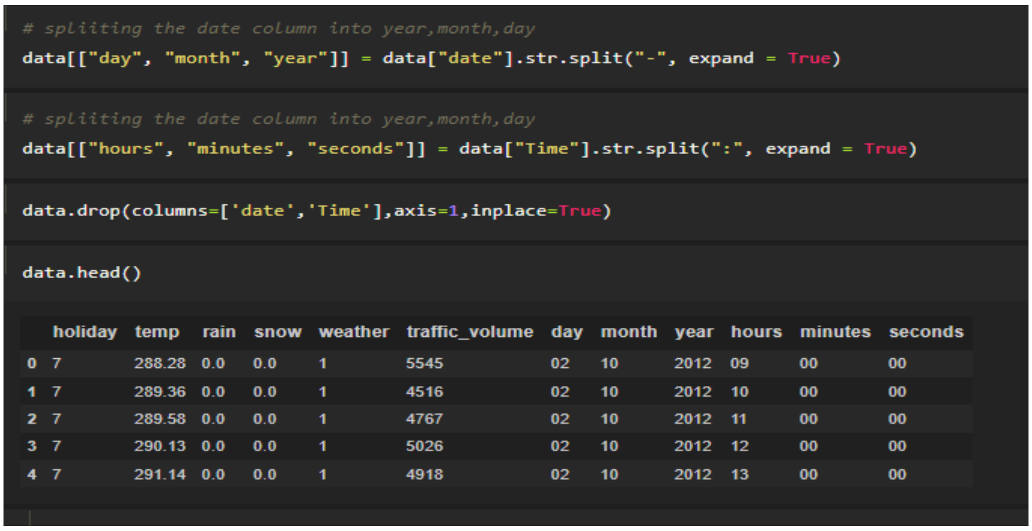
```
print(Counter(data['weather']))
```

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

```
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.



```
# spliiting the date column into year,month,day
data[["day", "month", "year"]] = data["date"].str.split("-", expand = True)
```

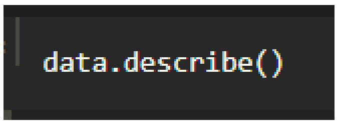
```
# spliiting the date column into year,month,day
data[["hours", "minutes", "seconds"]] = data["Time"].str.split(":", expand = True)
```

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

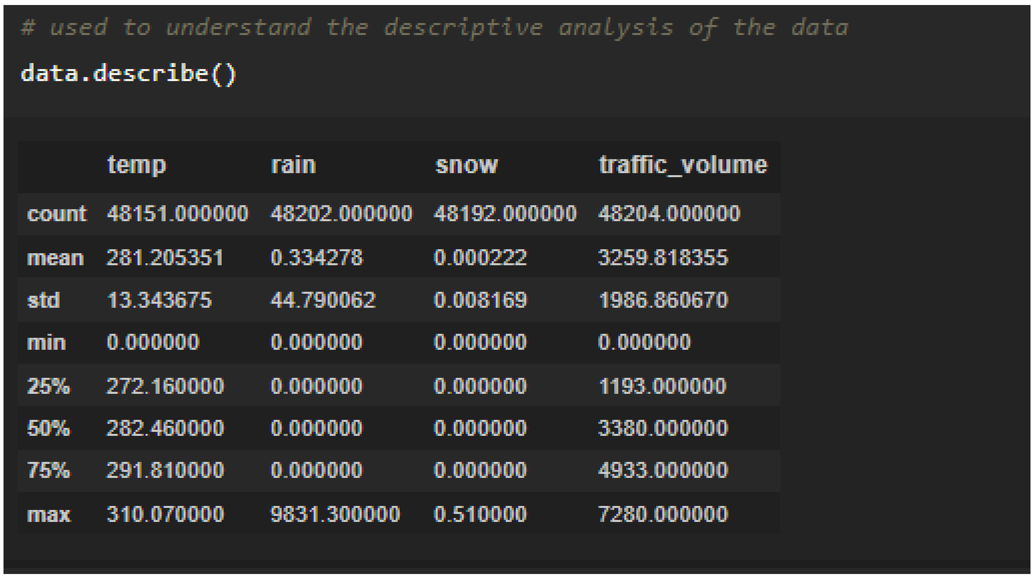
```
data.head()
```

```
   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
```

Activity 2.3: checking descriptive statistics



```
data.describe()
```



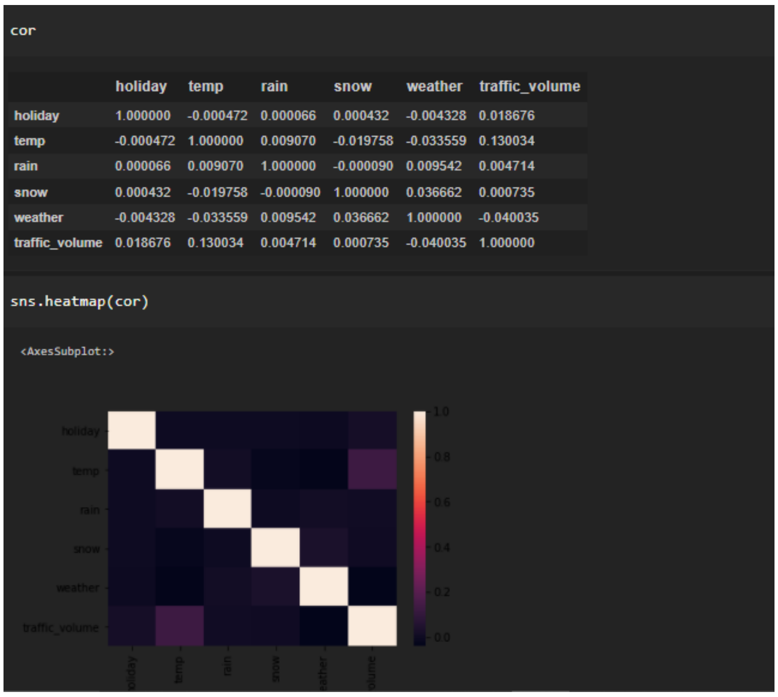
```
# used to understand the descriptive analysis of the data
data.describe()
```

```
       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
```

Activity 2.4: checking correlation

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.



```
cor
```

```
               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(cor)
```

```
<AxesSubplot:>
```

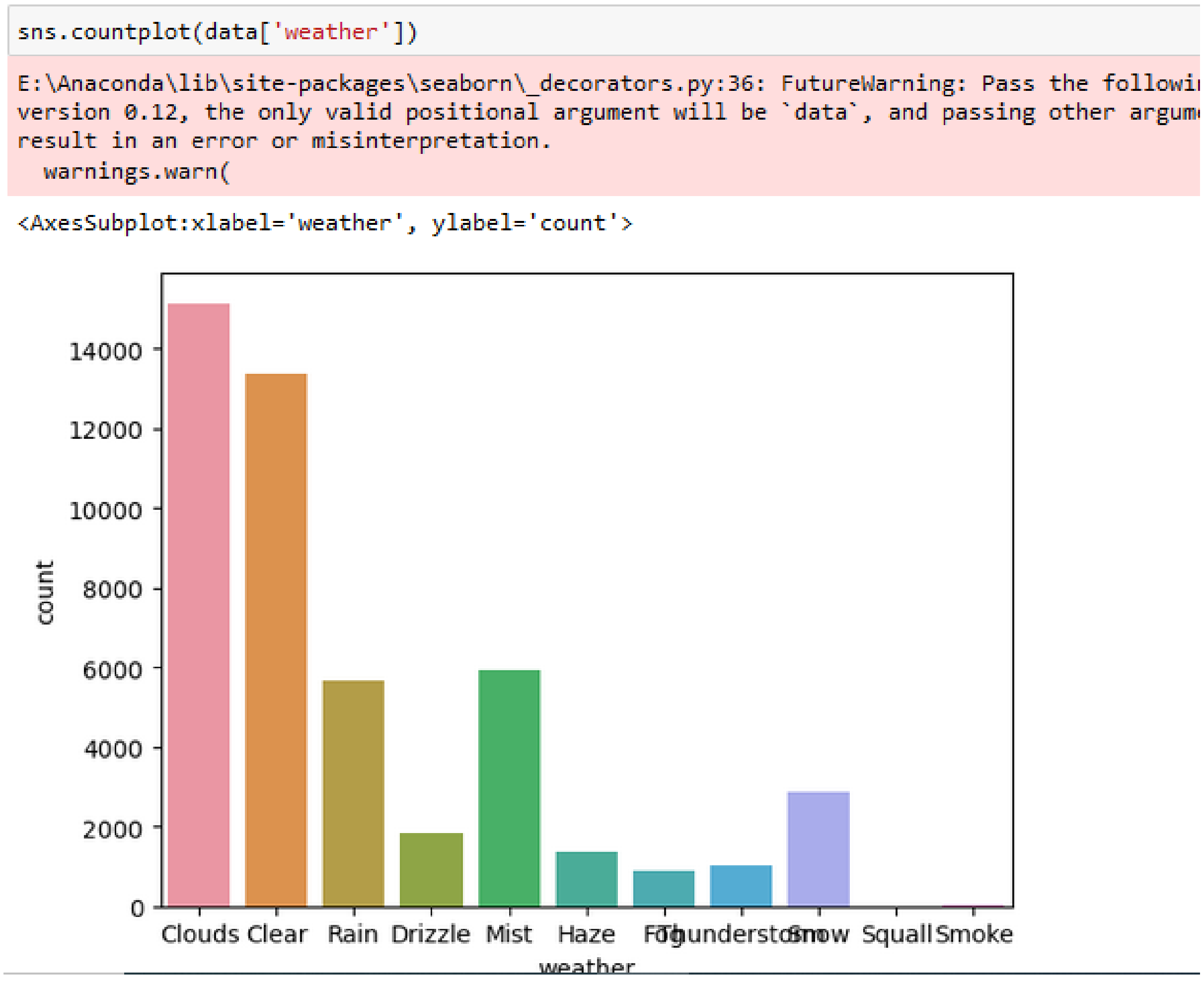
- Correlation strength varies based on colour, lighter the colour between two variables, more the strength between the variables, darker the colour displays the weaker correlation
- We can see the correlation scale values on left side of the above image

Milestone 3: Exploratory Data Analysis

Activity 1: Univariate Analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



```
sns.countplot(data['weather'])
```

```
E:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followi
version 0.12, the only valid positional argument will be `data`, and passing other argum
result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='weather', ylabel='count'>
```

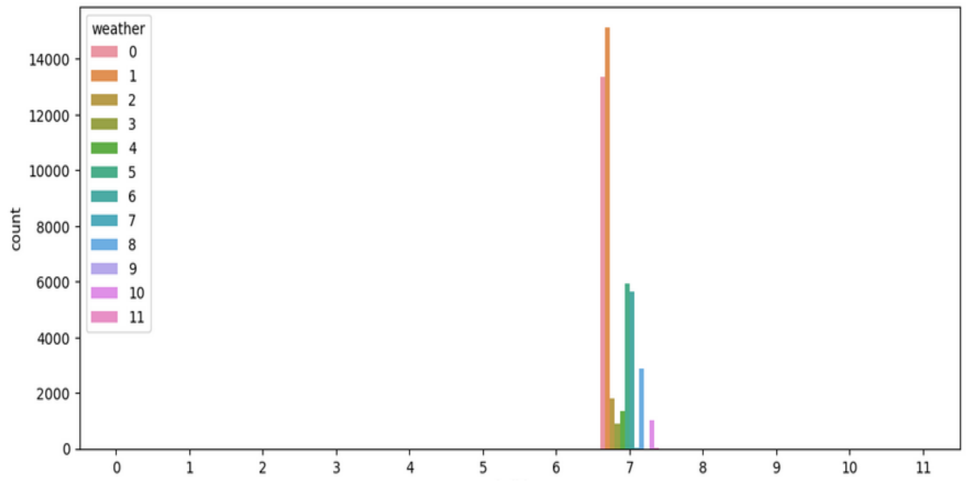
Activity 2: Bivariate Analysis

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

Count plot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```
sns.countplot(data['holiday'],hue=data['weather'])

plt.show()
```

```
E:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



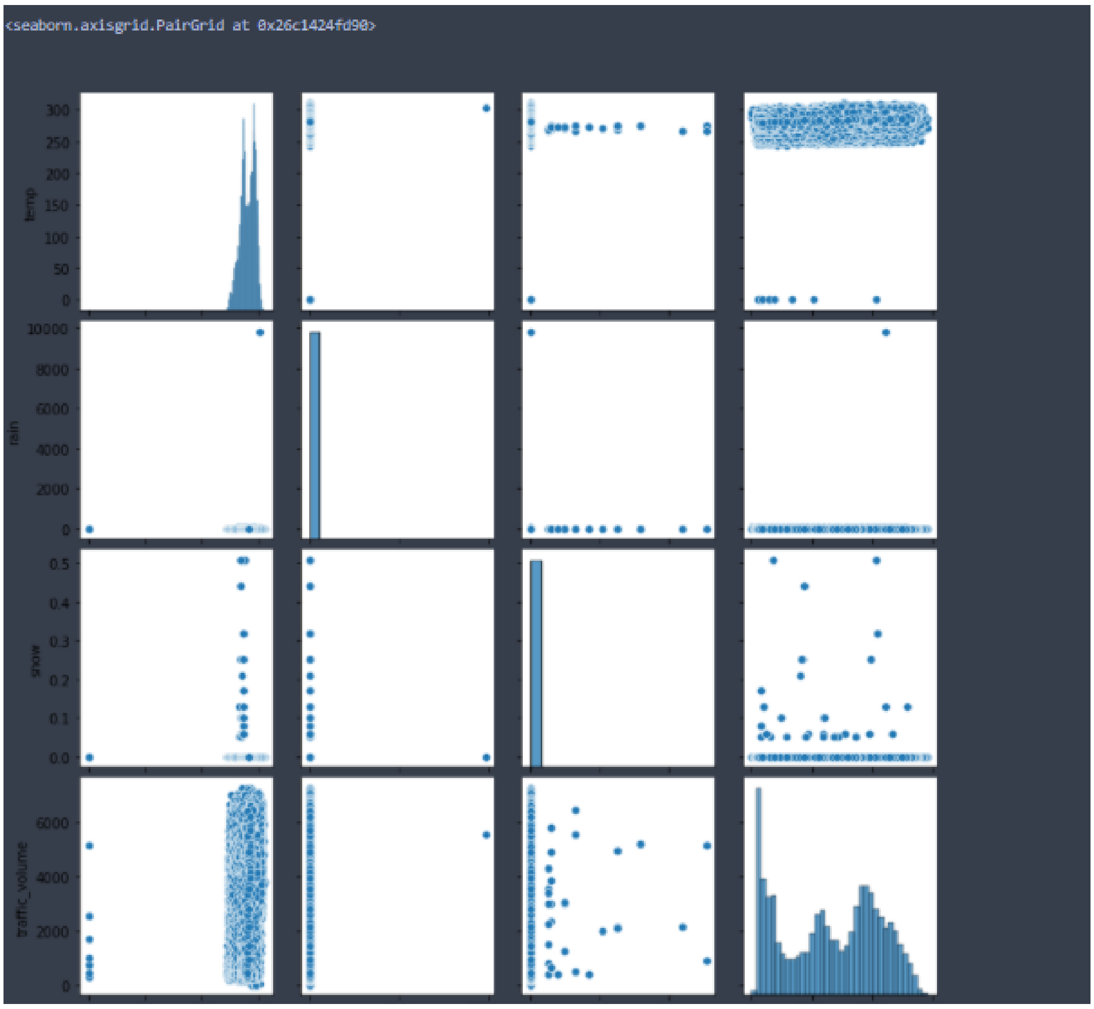
Activity 3: Multivariate Analysis

Pair Plot: Plot pairwise relationships in a dataset.

- By default, this function will create a grid of Axes such that each numeric variable in data will by 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)

  The output is as shown below



```
<seaborn.axisgrid.PairGrid at 0x26c1424fd90>
```

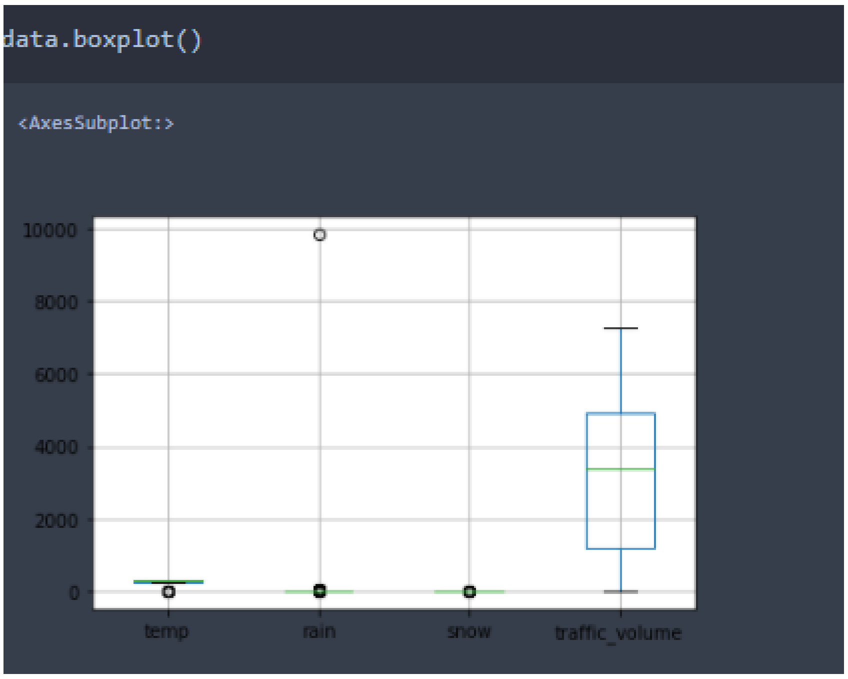
Pair plot usually gives pair wise relationships of the columns in the dataset

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

2.how features are correlated (positive, neutral and negative)

3.Box Plot: jupyter has a built-in function to create boxplot called boxplot(). A boxplot plot is a type of plot that shows the spread of data in all the quartiles



```
data.boxplot()
```

```
<AxesSubplot:>
```

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

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.
- 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].

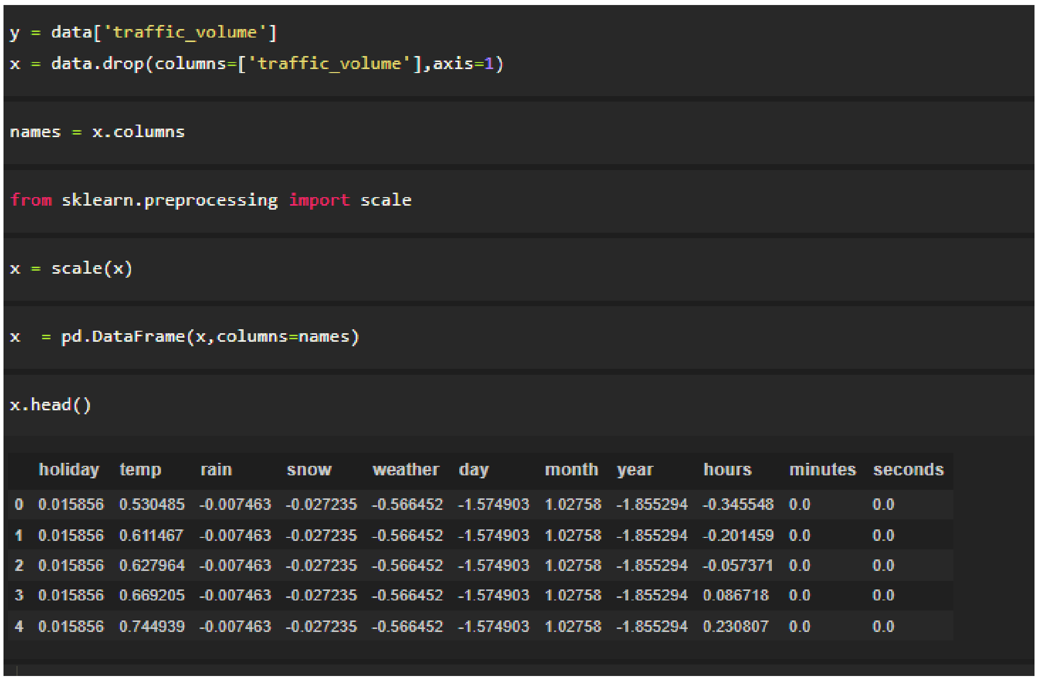
Let's split our dataset into independent and dependent variables.

y = data[traffic_volume] - independent
x = data.drop(traffic_volume,axis=1)

Feature Scaling

There is 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.



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

```
names = x.columns
```

```
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

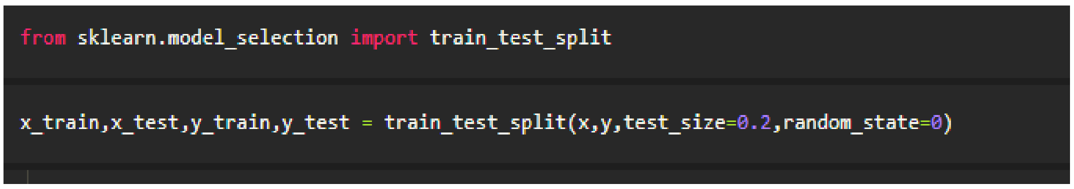
- After scaling the data will be converted into array form
- Loading the feature names before scaling and converting them back to data frame after standard scaling is applied

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 dataset. 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 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.

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)



```
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

Model building includes the following main tasks

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

Activity 1: Training 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

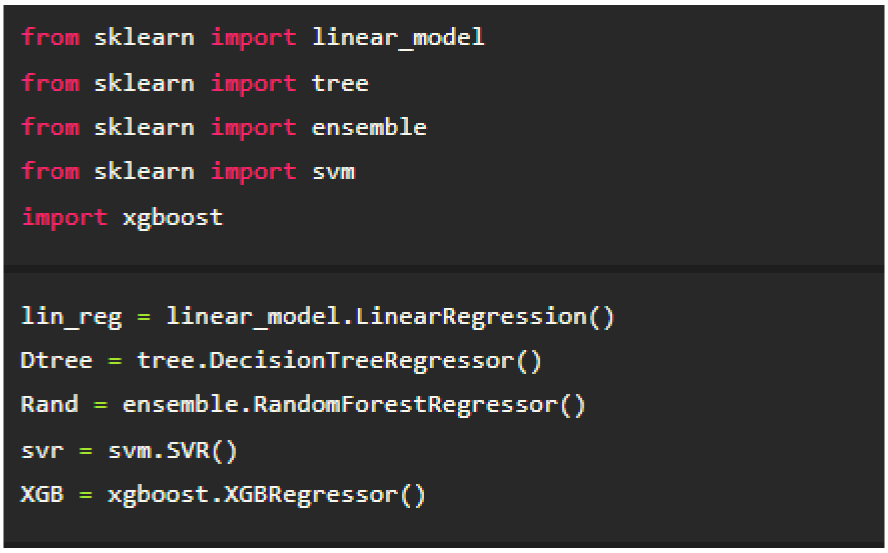
3.Random Forest Regressor

4.svm

5.xgboost

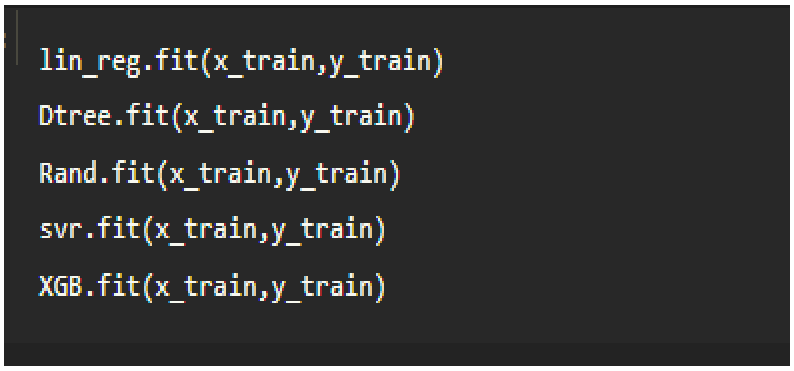
Steps in Building the model:-

- Initialize the model



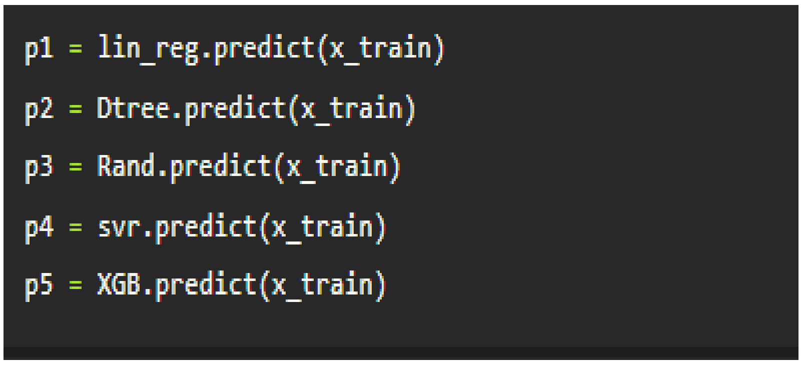
```python
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
import xgboost

lin_reg = linear_model.LinearRegression()
Dtree = tree.DecisionTreeRegressor()
Rand = ensemble.RandomForestRegressor()
svr = svm.SVR()
XGB = xgboost.XGBRegressor()
```

- Fit the models with x_train and y_train



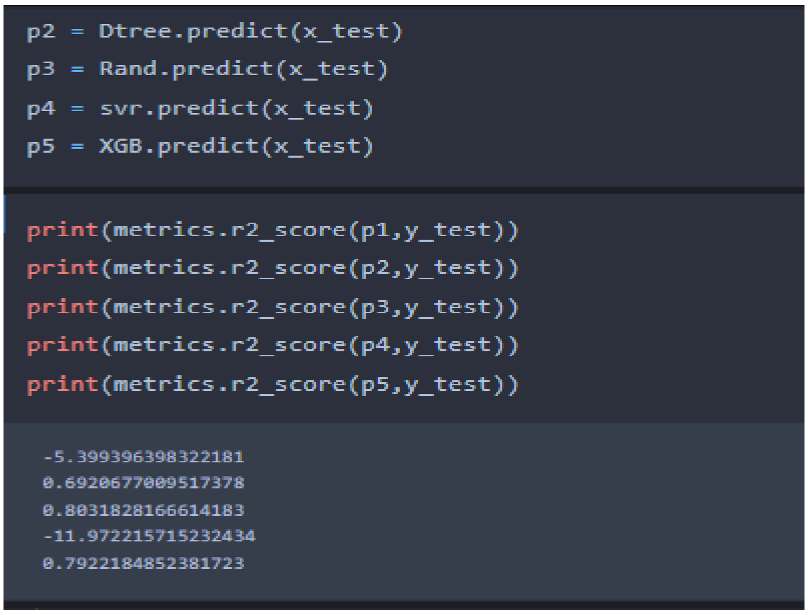
```
lin_reg.fit(x_train,y_train)
Dtree.fit(x_train,y_train)
Rand.fit(x_train,y_train)
svr.fit(x_train,y_train)
XGB.fit(x_train,y_train)
```

- Predict 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)
```

Activity 2: Testing the model



```
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))
```

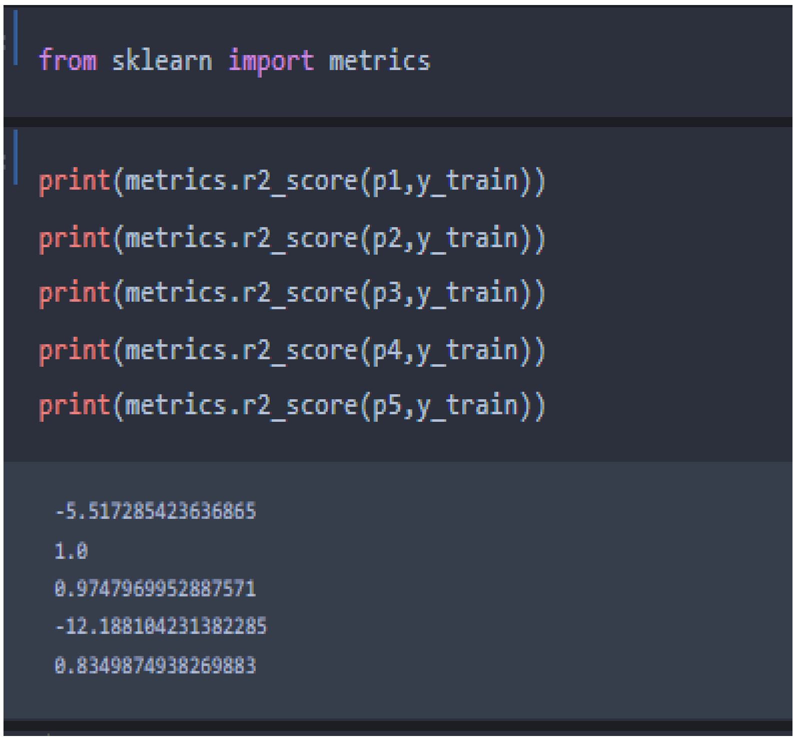
```
-5.399396398322181
0.6920677009517378
0.8031828166614183
-11.972215715232434
0.7922184852381723
```

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.

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.

Activity 2: Model Evaluation

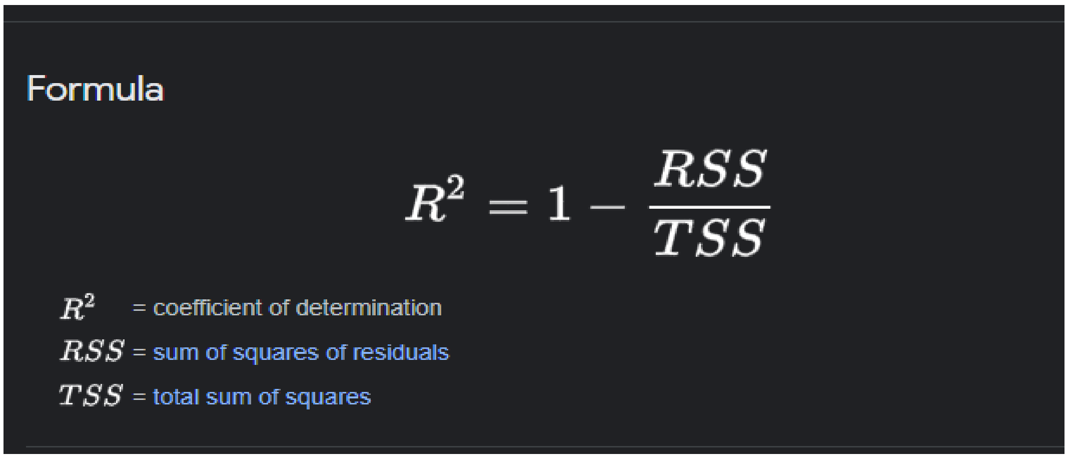
After training the model, the model should be tested by using the test data which is been separated while splitting the data for checking the functionality of the model.

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

1. R-squared _score



Formula

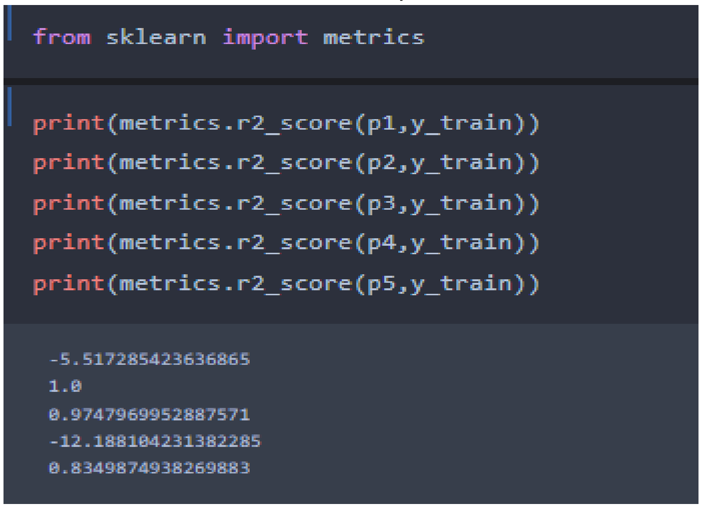
$$R^2 = 1 - \frac{RSS}{TSS}$$

R^2 = coefficient of determination

RSS = sum of squares of residuals

TSS = total sum of squares

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



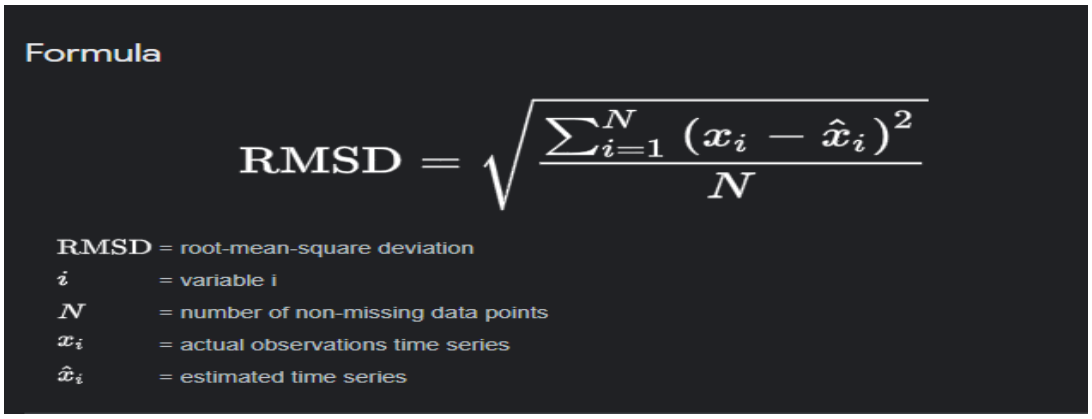
```
from sklearn import metrics
```

```
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
```

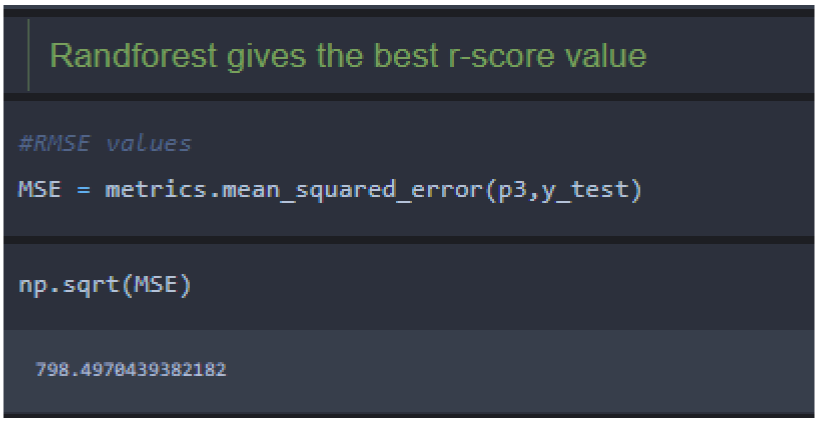
Select the model,which gives the best accuracy of all,and generate predictions and find the accuracy with training and testing data

2. RMSD –Root Mean Square deviation


Formula

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

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


Randforest gives the best r-score value

```
#RMSE values
MSE = metrics.mean_squared_error(p3,y_test)
```

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

```
798.4970439382182
```

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

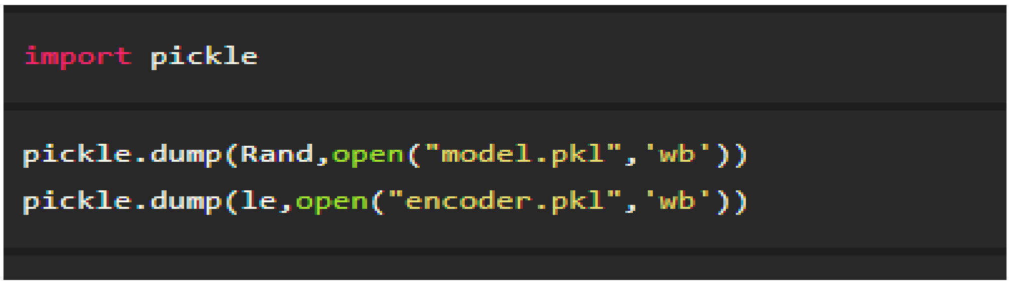
Milestone 6: Model Deployment

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


```
import pickle
```

```
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.

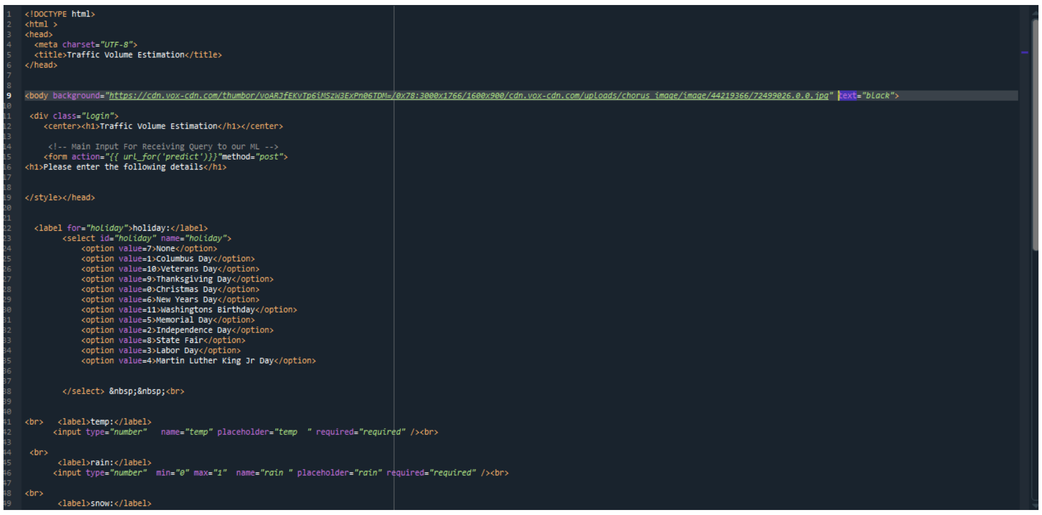
This section has the following tasks

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

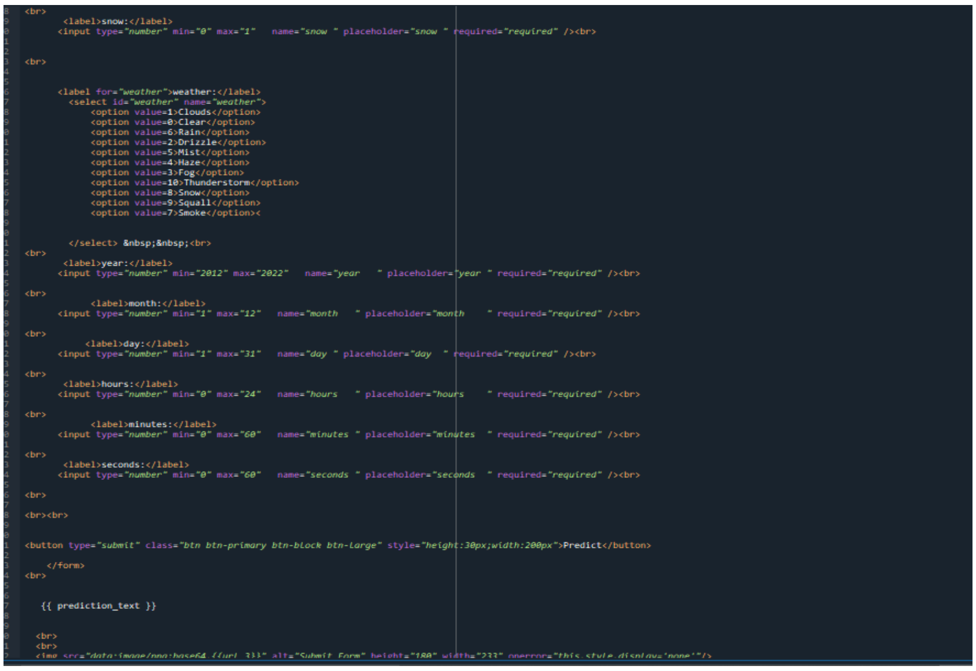
Activity 2.1: Build HTML Code

In this HTML page, we will create the front end part of the web page. In this page we will accept input from the user and Predict the values.
For more information regarding HTML
https://www.w3schools.com/html/
In our project we have HTML files ,they are
1.index.html

index.html



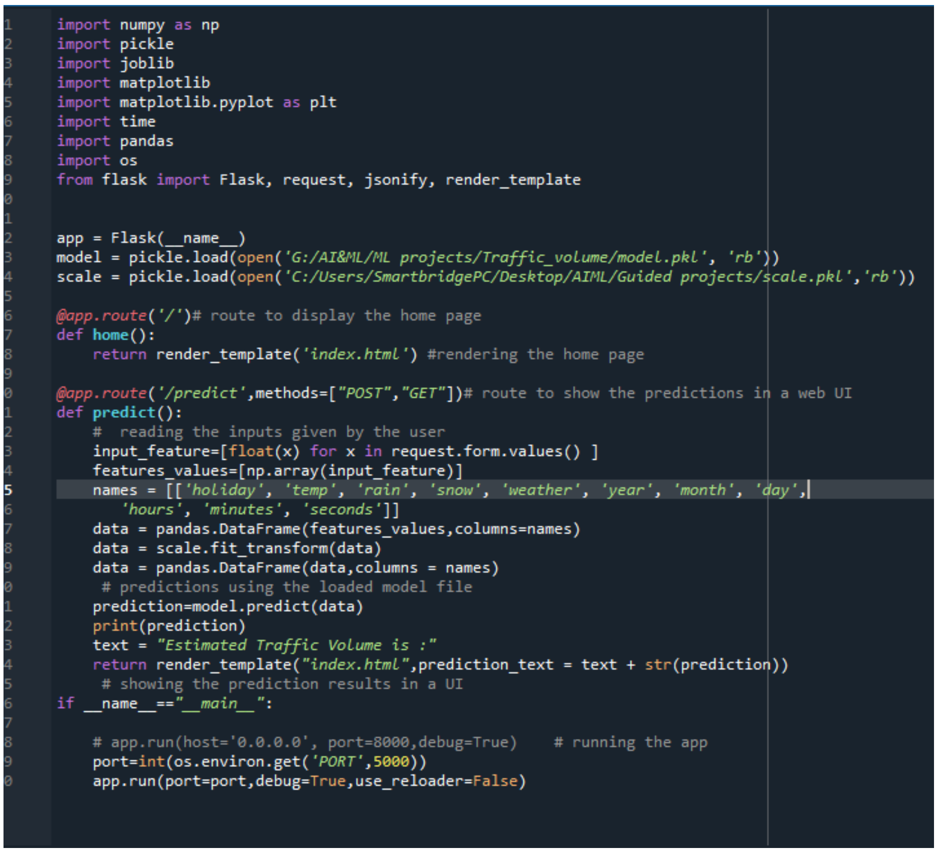
```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Traffic Volume Estimation</title>
</head>


<body background="https://cdn.vox-cdn.com/thumbor/voARJfEKvTp6iMSzW3ExPn06TDM=/0x78:3000x1766/1600x900/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>
```



```html
<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'"/>
```

Activity 2.2: Build Python code:

Import the libraries

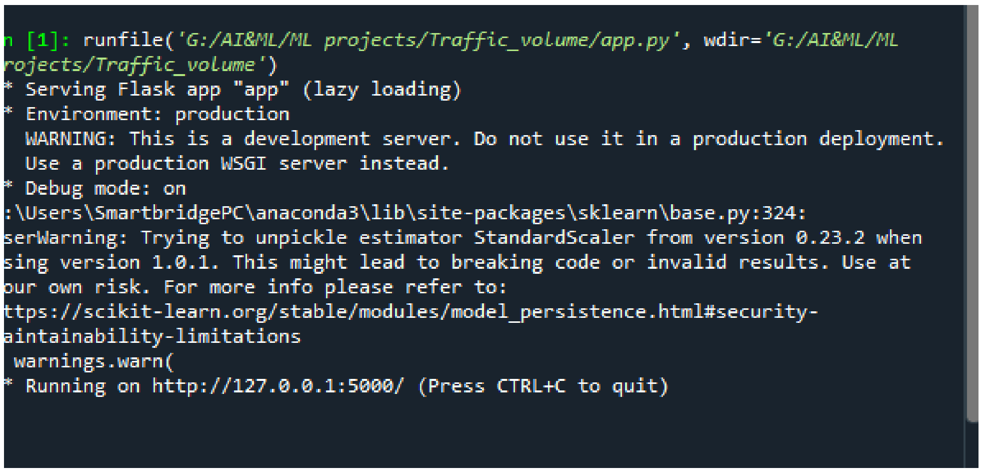


```python
import numpy as np
import pickle
import joblib
import matplotlib
import matplotlib.pyplot as plt
import time
import pandas
import os
from flask import Flask, request, jsonify, render_template


app = Flask(__name__)
model = pickle.load(open('G:/AI&ML/ML projects/Traffic_volume/model.pkl', 'rb'))
scale = pickle.load(open('C:/Users/SmartbridgePC/Desktop/AIML/Guided projects/scale.pkl','rb'))

@app.route('/')# route to display the home page
def home():
    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)
    data = scale.fit_transform(data)
    data = pandas.DataFrame(data,columns = names)
     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    text = "Estimated Traffic Volume is :"
    return render_template("index.html",prediction_text = text + str(prediction))
     # 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)
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

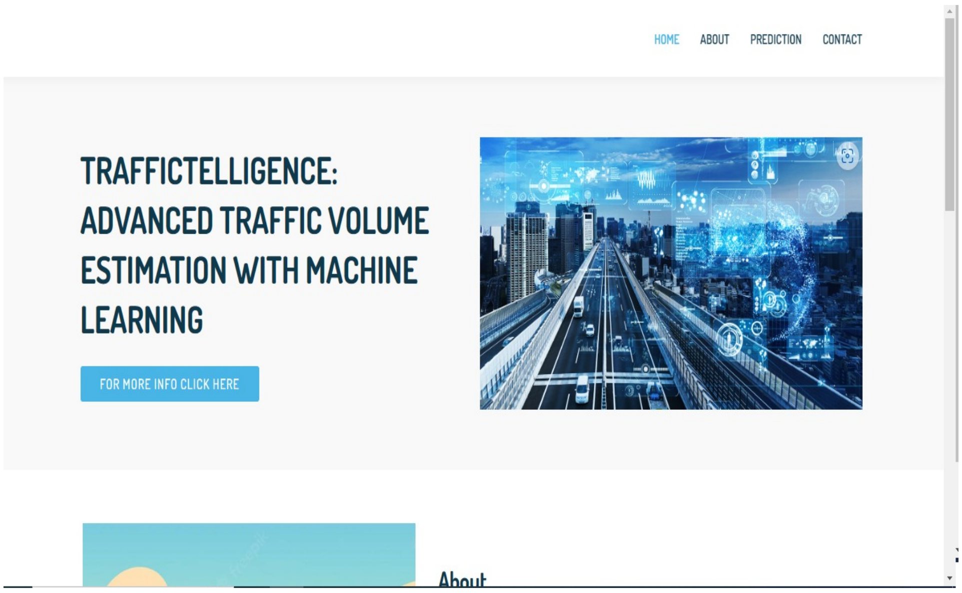
Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



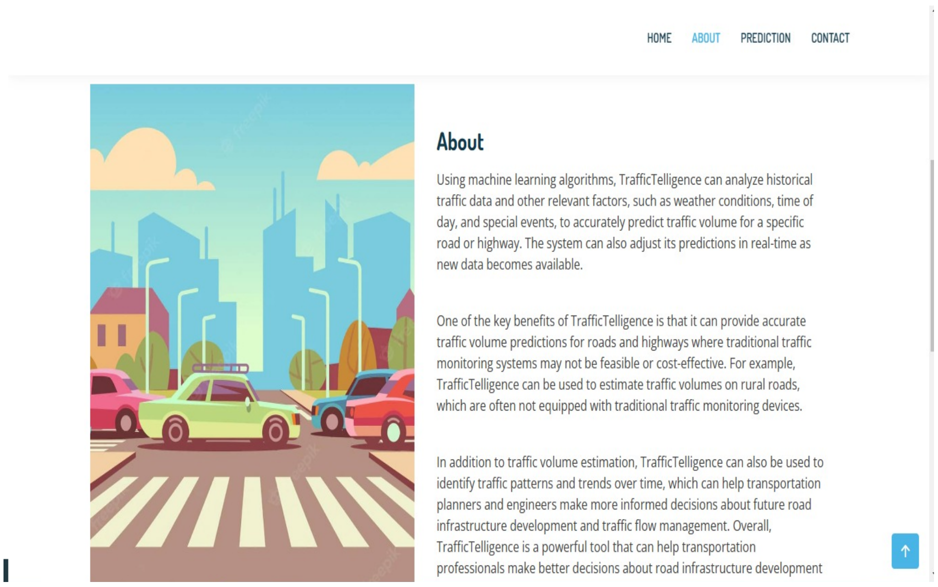
```
n [1]: runfile('G:/AI&ML/ML projects/Traffic_volume/app.py', wdir='G:/AI&ML/ML
rojects/Traffic_volume')
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
:\Users\SmartbridgePC\anaconda3\lib\site-packages\sklearn\base.py:324:
serWarning: Trying to unpickle estimator StandardScaler from version 0.23.2 when
sing version 1.0.1. This might lead to breaking code or invalid results. Use at
our own risk. For more info please refer to:
ttps://scikit-learn.org/stable/modules/model_persistence.html#security-
aintainability-limitations
 warnings.warn(
* 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



TRAFFICTELLIGENCE: ADVANCED TRAFFIC VOLUME ESTIMATION WITH MACHINE LEARNING

About

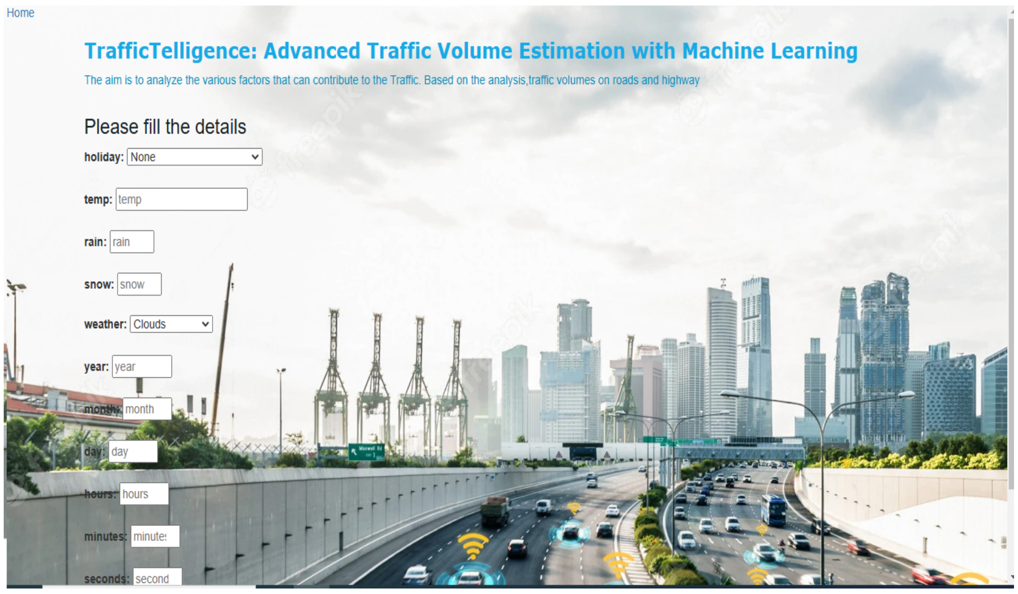


About

Using machine learning algorithms, TrafficTelligence can analyze historical traffic data and other relevant factors, such as weather conditions, time of day, and special events, to accurately predict traffic volume for a specific road or highway. The system can also adjust its predictions in real-time as new data becomes available.

One of the key benefits of TrafficTelligence is that it can provide accurate traffic volume predictions for roads and highways where traditional traffic monitoring systems may not be feasible or cost-effective. For example, TrafficTelligence can be used to estimate traffic volumes on rural roads, which are often not equipped with traditional traffic monitoring devices.

In addition to traffic volume estimation, TrafficTelligence can also be used to identify traffic patterns and trends over time, which can help transportation planners and engineers make more informed decisions about future road infrastructure development and traffic flow management. Overall, TrafficTelligence is a powerful tool that can help transportation professionals make better decisions about road infrastructure development



TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning

The aim is to analyze the various factors that can contribute to the Traffic. Based on the analysis,traffic volumes on roads and highway

Please fill the details

holiday: None

temp: temp

rain: rain

snow: snow

weather: Clouds

year: year

month: month

day: day

hours: hours

minutes: minutes

seconds: second

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

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided