TRAFFIC VOLUME ESTIMATION USING IBM WATSON MACHINE LEARNING

Developed By : B. Rajashekar, P. Nuthan Kumar, P. Swathi, K. Nishitha

Smart Bridge – Mini Project Report

1.INTRODUCTION

1.1 Overview

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.

1.2 Purpose

To overcome the problem of traffic congestion, the traffic prediction using machine learning which contains regression model and libraries like pandas, os, numpy, matplotlib.pyplot are used to predict the traffic. It is implemented so that the traffic congestion is controlled and can be accessed easily. Users can collect the traffic information of the traffic flow and can also check the congestion flow from the start of the day till the end of the day with the time span of one hour data.

2.LITERATURE SURVEY

2.1Existing problem and Existing approaches or methods

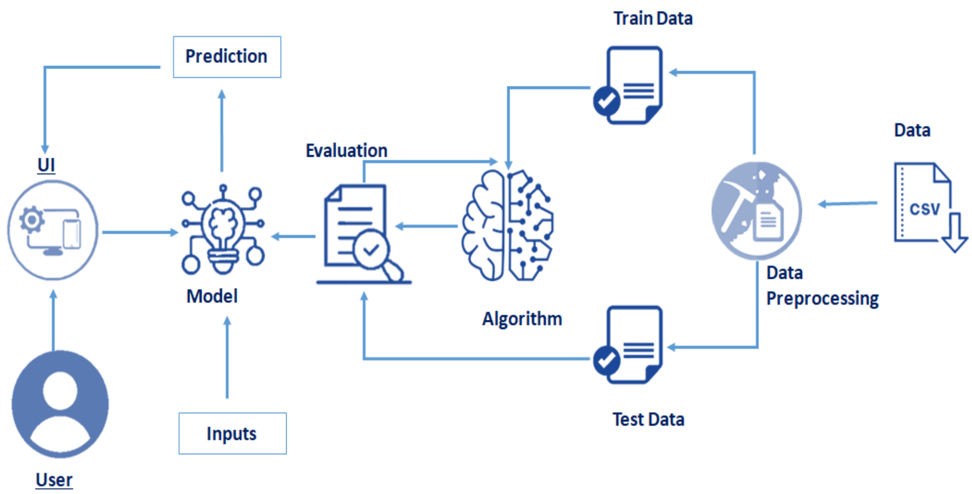
Traffic Volume Count can be done by various methods depending upon various factors like manpower available, budget, technology/instrument available, magnitude of traffic data required or to be collected which will then determine quality and type of vehicle classification to be adopted. Traffic counting falls in two main categories, namely: manual count and automatic count.Traffic data collection forms the integral part of traffic volume study as it provides the raw data and includes primary survey. The various types and methods used to collect traffic data not only provide a good and valuable coverage of the required traffic information.

2.2 Proposed solution

In this project we have shown use of 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.

3.THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware and Software requirements

Software Requirements:

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

Anaconda navigator

Python packages:

- numpy
- pandas.
- matplotlib.
- scikit-learn
- xgboost

Flask

Hardware Requirements

- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 4 GB
- Display : 14.1 "Color Monitor(LCD, CRT or LED)
- Clock Speed : 1.67 GHz

4.EXPERIMENTAL INVESTIGATION

For developing the project the team has completed several tasks:

Data Collection.

- Collect the dataset or Create the dataset

Data Pre-processing.

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Taking care of Missing Data.
- Feature Scaling.
- Splitting Data into Train and Test.

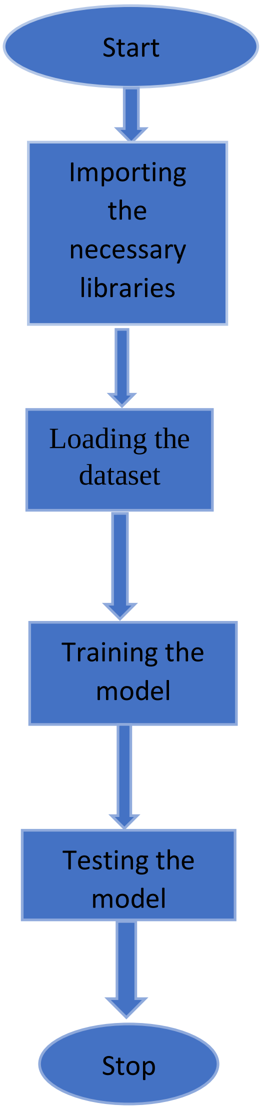
Model Building

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

Application Building

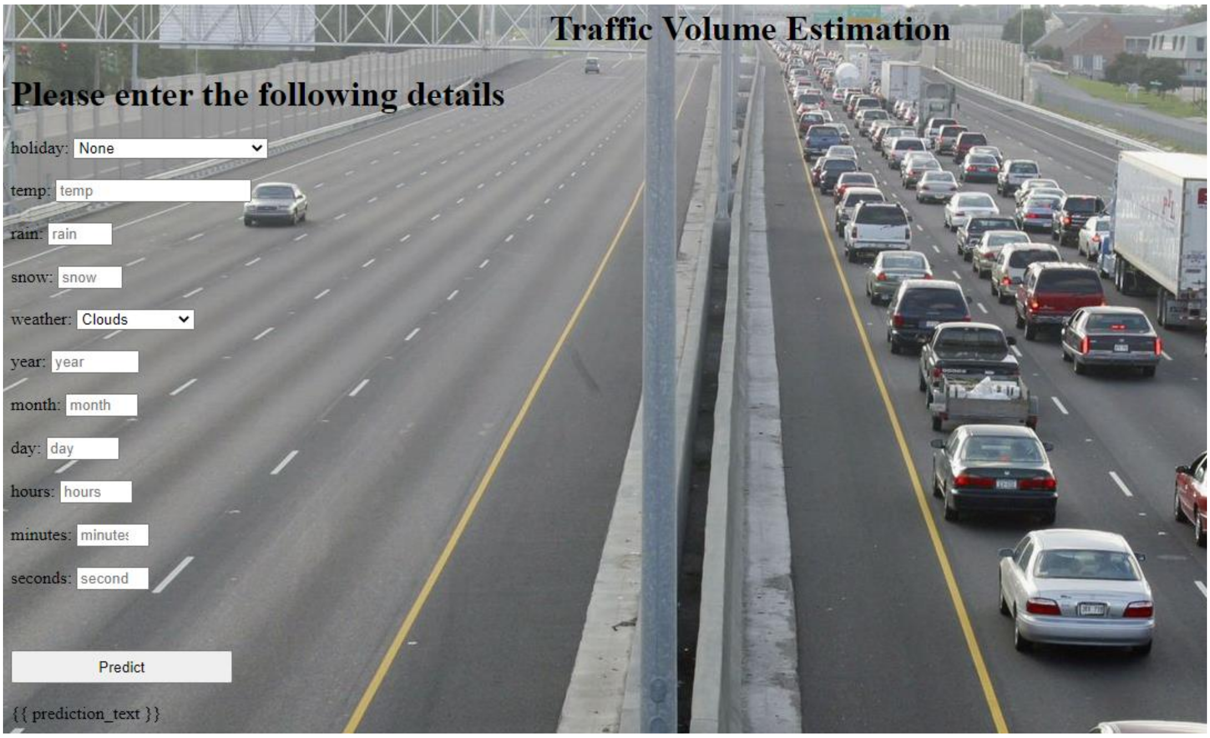
- Create an HTML file
- Build a Python Code
- Run the App

5.FLOW CHAT


Start
Importing the necessary libraries
Loading the dataset
Training the model
Testing the model
Stop

6. RESULTS

This is the HTML page where the user has to enter the details:



Final output of the project:

7.ADVANTAGES AND DISADVANTAGES

Advantages:

- Time complexity has been reduced
- Suitable for capturing the underlying relationships among different variables in an environment of uncertainty

Disadvantages:

- Interpretability of input variables (“black box”)
- Only predict within bounds of training – no extrapolation

8.APPLICATIONS

- We can estimate the volume of traffic by using machine learning techniques, which will cut down on the amount of time we spend in traffic.
- Smart cities can use this techniques.
- Structural design of roads.
- Planning and design of new streets.
- Planning, design and regulation for traffic.
- Establish properties and schedules for traffic improvements.

9.CONCLUSION

In the system, it has been concluded that we develop the traffic flow estimation system by using a machine learning algorithm and trained the model on IBM watson. By using regression model, the prediction is done. The public gets the benefits such as the current situation of the traffic flow, they can also check what will be the flow of traffic on the right after one hour of the situation. The weather conditions have been changing from years to years. The cost of fuel is also playing a major role in the transportation system. The forecasting or the prediction can help people or the users in judging the road traffic easier beforehand and even they can decide which way to go using their navigator and also this will prediction will be also helpful.

10.FUTURE SCOPE

These days, traffic prediction is extremely necessary for every part of the state and also worldwide. So, this method of prediction would be helpful in predicting the traffic earlier and decreasing the accidents which saves many lives. And it will also reduce the waste of time and excessive fuel consumption and air pollution in congested areas. For better congestion prediction, the grade and accuracy are prominent in traffic prediction. Within the future, the expectation are going to be the estimation of established order accuracy prediction with much easier and user-friendly methods. So people would find the prediction model useful and that they won't be wasting their time and energy to predict the information.

11. BIBILOGRAPHY

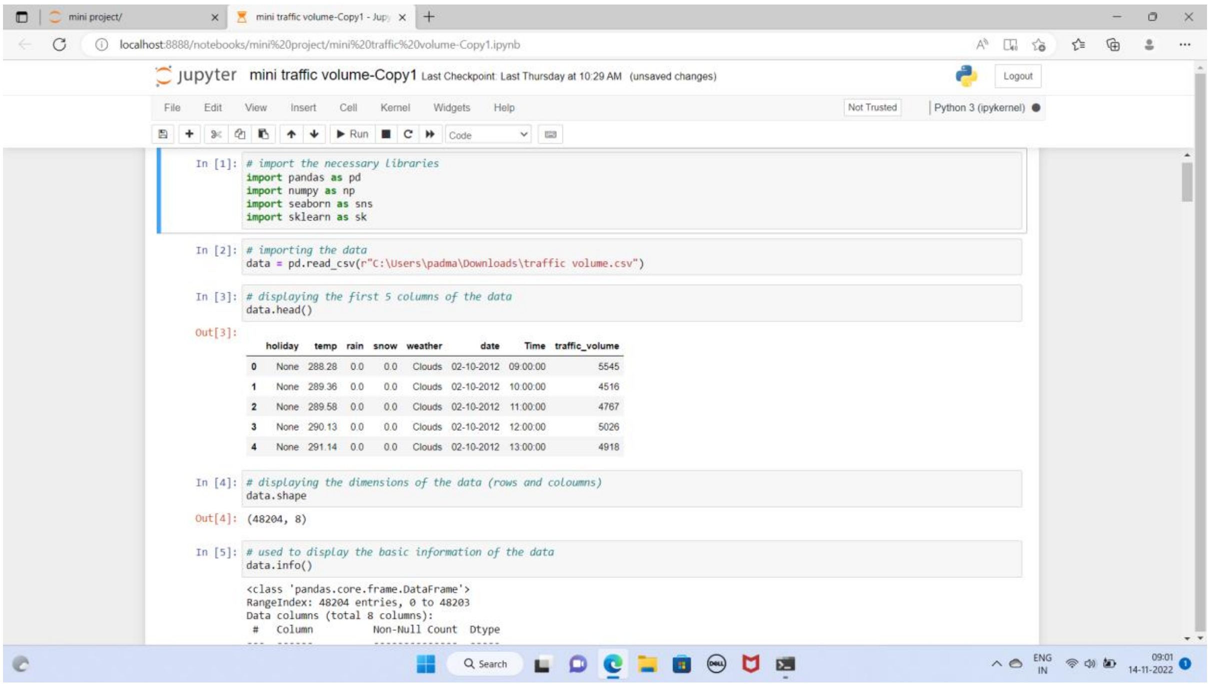
https://www.researchgate.net/publication/332407206_Traffic_Prediction_Using_Machine_Learning.

https://www.ibm.com/docs/en/watson-studio-local/1.2.3?topic=model.

www.engineeringenotes.com/transportation-engineering/traffic-engineering/traffic_engineering.

APPENDIX

Source Code



jupyter mini traffic volume-Copy1 Last Checkpoint: Last Thursday at 10:29 AM (unsaved changes)

In [1]:
```python
# import the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
```

In [2]:
```python
# importing the data
data = pd.read_csv(r"C:\Users\padma\Downloads\traffic volume.csv")
```

In [3]:
```python
# displaying the first 5 columns of the data
data.head()
```

Out[3]:

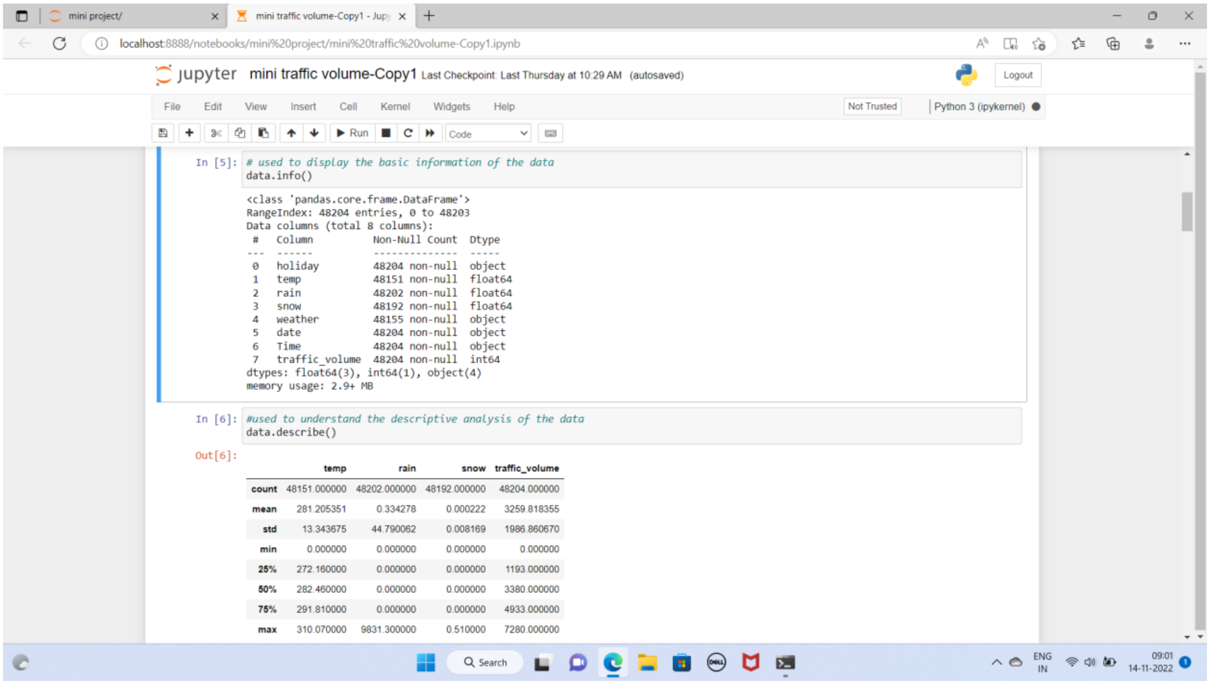
	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	None	288.28	0.0	0.0	Clouds	02-10-2012	09:00:00	5545
1	None	289.36	0.0	0.0	Clouds	02-10-2012	10:00:00	4516
2	None	289.58	0.0	0.0	Clouds	02-10-2012	11:00:00	4767
3	None	290.13	0.0	0.0	Clouds	02-10-2012	12:00:00	5026
4	None	291.14	0.0	0.0	Clouds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

In [4]:
```python
# displaying the dimensions of the data (rows and coloumns)
data.shape
```

Out[4]: (48204, 8)

In [5]:
```python
# 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
---  ------          --------------  -----
```



jupyter mini traffic volume-Copy1 Last Checkpoint: Last Thursday at 10:29 AM (autosaved)

In [5]:
```python
# 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 [6]:
```python
#used to understand the descriptive analysis of the data
data.describe()
```

Out[6]:

	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

```
In [7]: # used to display the null values of the data
        data.isnull().sum()

Out[7]: holiday           0
        temp             53
        rain              2
        snow             12
        weather          49
        date              0
        Time              0
        traffic_volume    0
        dtype: int64
```

```
In [8]: # used to display the data types of each column
        data.dtypes

Out[8]: holiday            object
        temp              float64
        rain              float64
        snow              float64
        weather            object
        date               object
        Time               object
        traffic_volume      int64
        dtype: object
```

Dealing with missing values of the data

```
In [9]: from collections import Counter
```

```
In [10]: print(Counter(data['rain']))
         print(Counter(data['snow']))

Counter({0.0: 44735, 0.25: 948, 0.51: 256, 1.02: 123, 0.3: 121, 0.76: 109, 0.38: 99, 1.78: 91, 1.52: 69, 0.64: 55, 1.27: 50, 0.
```

```
In [10]: print(Counter(data['rain']))
         print(Counter(data['snow']))

Counter({0.0: 44735, 0.25: 948, 0.51: 256, 1.02: 123, 0.3: 121, 0.76: 109, 0.38: 99, 1.78: 91, 1.52: 69, 0.64: 55, 1.27: 50, 0.
6: 32, 2.79: 29, 0.44: 26, 0.89: 25, 2.54: 23, 0.28: 23, 0.42: 21, 1.4: 21, 0.34: 20, 2.16: 19, 2.29: 19, 2.03: 19, 1.8: 16, 1.
09: 16, 3.05: 15, 0.32: 15, 1.2: 15, 0.9: 15, 0.98: 14, 0.68: 13, 0.81: 13, 4.57: 13, 7.11: 12, 0.85: 12, 0.7: 11, 2.1: 11, 0.5
5: 11, 5.59: 10, 1.86: 10, 8.4: 10, 1.15: 10, 0.47: 9, 5.08: 9, 1.21: 9, 0.43: 9, 6.1: 9, 5.84: 8, 1.66: 8, 0.79: 8, 0.4: 8, 1.
14: 8, 2.2: 8, 1.85: 8, 2.41: 8, 3.3: 8, 1.41: 7, 6.6: 7, 0.35: 7, 1.91: 7, 0.52: 7, 1.3: 7, 0.8: 7, 0.66: 7, 2.67: 7, 1.33: 7,
1.1: 7, 4.06: 7, 0.57: 6, 0.29: 6, 0.36: 6, 1.0: 6, 1.44: 6, 8.64: 6, 1.35: 6, 5.97: 6, 0.56: 6, 0.91: 6, 0.54: 6, 0.94: 6, 0.9
6: 6, 2.86: 6, 0.78: 6, 2.22: 6, 6.35: 6, 4.89: 6, 0.93: 6, 3.13: 6, 0.63: 6, 2.62: 6, 1.6: 6, 4.74: 6, 2.76: 6, 3.45: 6, 0.69:
5, 3.18: 5, 0.61: 5, 5.42: 5, 1.39: 5, 0.53: 5, 0.48: 5, 0.59: 5, 0.71: 5, 0.27: 5, 2.85: 5, 0.65: 5, 1.5: 5, 3.41: 5, 4.29: 5,
1.72: 5, 2.61: 5, 1.69: 5, 4.15: 5, 9.62: 5, 0.84: 4, 6.94: 4, 4.32: 4, 1.68: 4, 0.41: 4, 1.06: 4, 2.05: 4, 0.88: 4, 4.45: 4,
5.46: 4, 2.7: 4, 4.21: 4, 9.9: 4, 0.86: 4, 5.92: 4, 10.67: 4, 13.46: 4, 3.94: 4, 20.07: 4, 3.27: 4, 4.0: 4, 2.92: 4, 10.6: 4,
1.34: 3, 1.84: 3, 1.7: 3, 5.74: 3, 4.98: 3, 3.65: 3, 12.19: 3, 7.54: 3, 16.38: 3, 1.65: 3, 3.81: 3, 7.37: 3, 10.54: 3, 19.9: 3,
25.32: 3, 21.42: 3, 9.53: 3, 13.21: 3, 2.37: 3, 3.98: 3, 4.27: 3, 1.13: 3, 0.97: 3, 14.73: 3, 0.95: 3, 1.07: 3, 1.11: 3, 1.24:
3, 3.19: 3, 4.76: 3, 5.27: 3, 11.58: 3, 7.02: 3, 3.08: 3, 1.98: 3, 1.04: 3, 1.55: 3, 6.89: 3, 3.9: 3, 5.02: 3, 4.09: 3, 1.19:
3, 4.8: 3, 4.18: 3, 1.49: 3, 9.4: 3, 3.2: 3, 7.97: 3, 23.8: 3, 11.78: 3, 7.51: 3, 2.15: 3, 9.91: 3, 27.57: 3, 7.29: 3, 13.64:
3, 7.25: 3, 2.91: 3, 20.24: 3, 13.32: 3, 4.38: 3, 3.54: 3, 6.47: 3, 1.56: 3, 8.04: 3, 25.46: 3, 3.74: 3, 2.49: 3, 5.04: 3, 5.3
6: 3, 2.38: 3, 3.28: 3, 4.04: 3, 3.86: 3, 5.69: 3, 6.01: 3, 5.21: 3, 4.7: 3, 10.92: 3, 7.62: 3, 11.23: 3, 9.42: 3, 10.16: 3, 9.
15: 3, 3.75: 3, 1.82: 3, 5.62: 3, 3.1: 3, 2.6: 3, 1.45: 3, 2.26: 3, 2.48: 3, 5.12: 3, 1.01: 3, 4.79: 3, 5.19: 3, 3.39: 2, 0.46:
2, 9.14: 2, 3.56: 2, 16.0: 2, 1.96: 2, 4.39: 2, 28.7: 2, 0.83: 2, 0.72: 2, 0.31: 2, 0.26: 2, 0.58: 2, 1.08: 2, 1.71: 2, 2.21:
2, 0.62: 2, 1.12: 2, 1.46: 2, 1.32: 2, 0.87: 2, 7.39: 2, 1.83: 2, 0.5: 2, 0.77: 2, 2.98: 2, 4.43: 2, 6.45: 2, 1.76: 2, 7.77: 2,
8.89: 2, 15.41: 2, 5.25: 2, 8.02: 2, 12.7: 2, 1.03: 2, 5.86: 2, 7.87: 2, 1.67: 2, 3.09: 2, 1.51: 2, 7.72: 2, 4.64: 2, 2.06: 2,
2.4: 2, 2.96: 2, 6.48: 2, 5.89: 2, 2.39: 2, 2.88: 2, 3.4: 2, 1.29: 1, 44.45: 1, 55.63: 1, 18.8: 1, 0.37: 1, 0.67: 1, 1.87: 1,
0.33: 1, 2.13: 1, 1.63: 1, 1.38: 1, 2.35: 1, 2.11: 1, 2.53: 1, 0.92: 1, 1.22: 1, 1.05: 1, 2.31: 1, 3.17: 1, 2.14: 1, 2.34: 1,
1.61: 1, 5.58: 1, 5.11: 1, 5.1: 1, 4.53: 1, 1.25: 1, 4.5: 1, 3.47: 1, 0.45: 1, 2.18: 1, 2.84: 1, 2.93: 1, 2.87: 1, 2.8: 1, 0.7
4: 1, 1.28: 1, 1.47: 1, 4.66: 1, 2.08: 1, 3.12: 1, 1.53: 1, 3.25: 1, 1.9: 1, 12.45: 1, 1.37: 1, 2.78: 1, 1.31: 1, 3.44: 1, 2.7
5: 1, 2.19: 1, 1.59: 1, 5.73: 1, 5.93: 1, 3.91: 1, 18.03: 1, 1.88: 1, 3.01: 1, 2.12: 1, 0.73: 1, 11.59: 1, 2.33: 1, 5.52: 1, 1.
93: 1, 2.68: 1, 10.05: 1, 7.7: 1, 4.05: 1, 3.8: 1, 9831.3: 1, 16.51: 1, 12.83: 1, 18.42: 1, 5.06: 1, 1.95: 1, 9.0: 1, 8.86: 1,
5.99: 1, 8.0: 1, 31.75: 1, 5.41: 1, 2.83: 1, 15.75: 1, 3.64: 1, 7.13: 1, 1.16: 1, 7.05: 1, 2.73: 1, nan: 1, nan: 1})
Counter({0.0: 48129, 0.05: 14, 0.06: 12, 0.51: 6, 0.25: 6, 0.13: 6, 0.1: 6, 0.32: 5, 0.17: 3, 0.44: 2, 0.08: 2, nan: 1, nan: 1,
nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 0.21: 1, nan: 1})
```

```
In [11]: 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 [12]:
```python
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 [13]:
```python
data['weather'].fillna('Clouds',inplace=True)
```

In [14]:
```python
data.isnull().sum()
```

Out[14]:
```
holiday           0
temp              0
rain              0
snow              0
weather           0
date              0
Time              0
traffic_volume    0
dtype: int64
```

Encoding the data

In [15]:
```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['weather'] = le.fit_transform(data['weather'])
data['holiday'] = le.fit_transform(data['holiday'])
data.head()
```

Out[15]:

	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	7	288.28	0.0	0.0	1	02-10-2012	09:00:00	5545
1	7	289.36	0.0	0.0	1	02-10-2012	10:00:00	4516
2	7	289.58	0.0	0.0	1	02-10-2012	11:00:00	4767
3	7	290.13	0.0	0.0	1	02-10-2012	12:00:00	5026
4	7	291.14	0.0	0.0	1	02-10-2012	13:00:00	4918

In [16]:
```python
le = LabelEncoder()
```

In [17]:
```python
data['weather'] = le.fit_transform(data['weather'])
```

In [18]:
```python
data['holiday'] = le.fit_transform(data['holiday'])
```

In [19]:
```python
data.head()
```

Out[19]:

	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	7	288.28	0.0	0.0	1	02-10-2012	09:00:00	5545
1	7	289.36	0.0	0.0	1	02-10-2012	10:00:00	4516
2	7	289.58	0.0	0.0	1	02-10-2012	11:00:00	4767
3	7	290.13	0.0	0.0	1	02-10-2012	12:00:00	5026
4	7	291.14	0.0	0.0	1	02-10-2012	13:00:00	4918

In [20]:
```python
# splitting the data column into year,month,day
data[["day", "month", "year"]] = data["date"].str.split("-", expand = True)
```

In [21]:
```python
# splitting the data column into year,month,day
data[["hours", "minutes", "seconds"]] = data["Time"].str.split(":", expand = True)
```

In [22]:
```python
data.drop(columns=['date','Time'],axis=1,inplace=True)
```

In [23]:
```python
data.head()
```

Out[23]:

holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds

In [23]:
```
data.head()
```

Out[23]:

	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

scaling the data

In [24]:
```
y = data['traffic_volume']
x = data.drop(columns=['traffic_volume'],axis=1)
x
y
x.shape
y.shape
```

Out[24]: (48204,)

In [25]:
```
y = data['traffic_volume']
x = data.drop(columns=['traffic_volume'],axis=1)
```

In [26]:
```
names = x.columns
```

In [27]:
```
from sklearn.preprocessing import scale
```

In [28]:
```
x = scale(x)
```

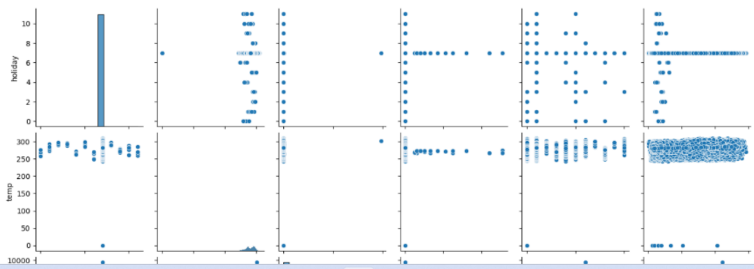
In [29]:
```
x  = pd.DataFrame(x,columns=names)
x.head()
```

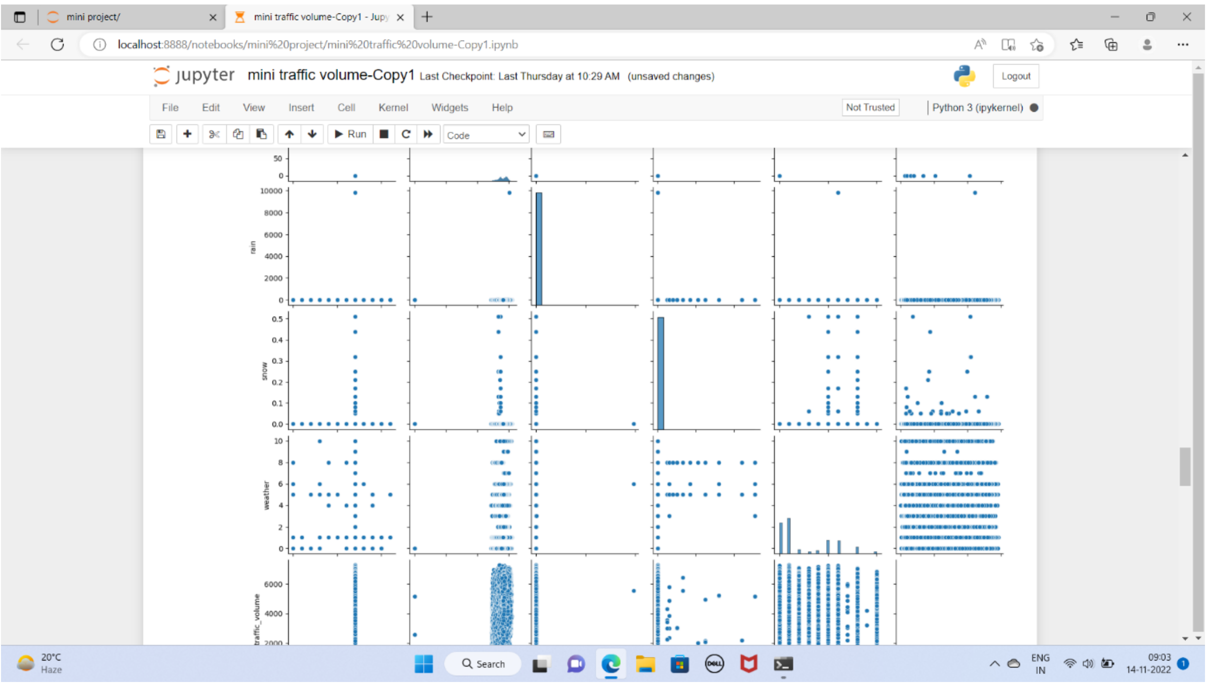
Out[29]:

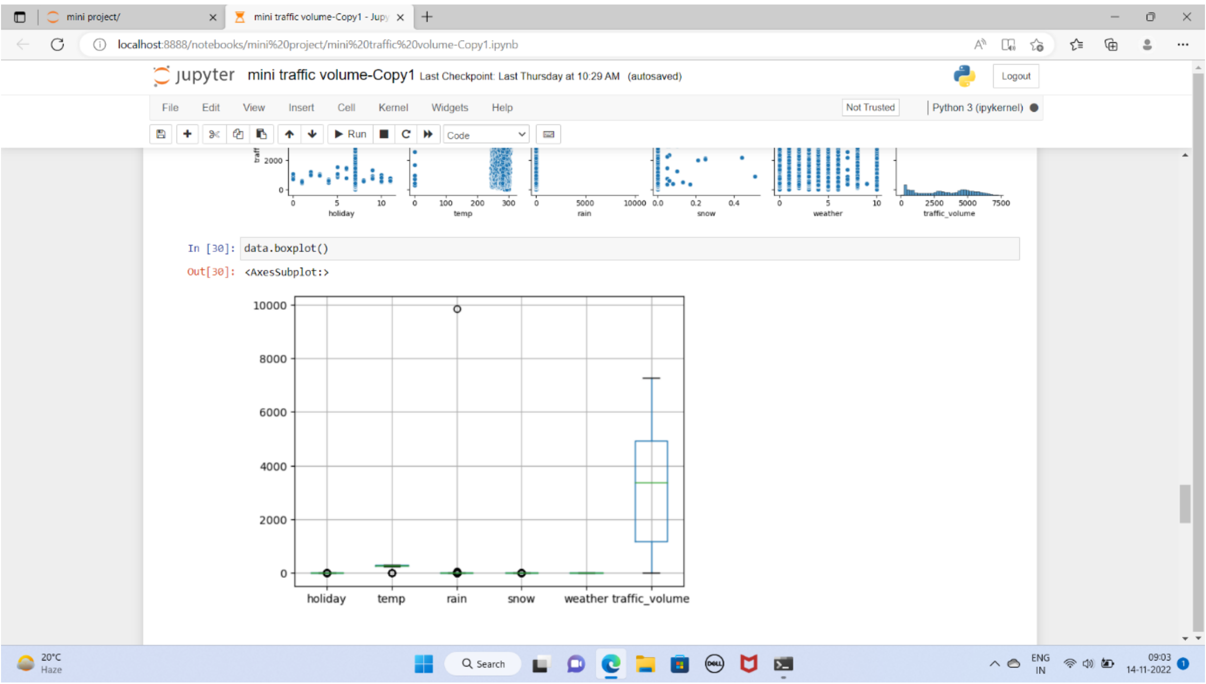
	holiday	temp	rain	snow	weather	day	month	year	hours	minutes	seconds
0	0.015856	0.530485	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.345548	0.0	0.0
1	0.015856	0.611467	-0.007463	-0.027235	-0.566452	-1.574903	1.02758	-1.855294	-0.201459	0.0	0.0
2	0.015856	0.627964	-0.007463	-0.027235	-0.566452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

In [30]:
```
sns.pairplot(data)
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x1b80cc45580>




mini project/
mini traffic volume-Copy1 - Jupy
localhost:8888/notebooks/mini%20project/mini%20traffic%20volume-Copy1.ipynb
jupyter mini traffic volume-Copy1 Last Checkpoint: Last Thursday at 10:29 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
rain
snow
weather
traffic_volume
20°C
Haze
Search
ENG
IN
09:03
14-11-2022


mini project/
mini traffic volume-Copy1 - Jupy
localhost:8888/notebooks/mini%20project/mini%20traffic%20volume-Copy1.ipynb
jupyter mini traffic volume-Copy1 Last Checkpoint: Last Thursday at 10:29 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
holiday
temp
rain
snow
weather
traffic_volume
In [30]: data.boxplot()
Out[30]: <AxesSubplot:>
10000
8000
6000
4000
2000
0
holiday
temp
rain
snow
weather traffic_volume
20°C
Haze
Search
ENG
IN
09:03
14-11-2022

mini traffic volume-Copy1 Last Checkpoint: Last Thursday at 10:29 AM (autosaved)

splitting the data

In [31]:
```
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)
```

In [32]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

model building

In [33]:
```
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
import xgboost
```

In [34]:
```
lin_reg = linear_model.LinearRegression()
Dtree = tree.DecisionTreeRegressor()
Rand = ensemble.RandomForestRegressor()
svr = svm.SVR()
XGB = xgboost.XGBRegressor()
```

In [35]:
```
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[35]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
```

In [40]:
```
cor = data.corr()
cor
```

Out[40]:

	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	0.006070	-0.019758	-0.033558	0.130034
rain	0.000066	0.006070	1.000000	-0.000090	0.009542	0.004714
snow	0.000432	-0.019758	-0.000090	1.000000	0.036662	0.000735
weather	-0.004328	-0.033558	0.009542	0.036662	1.000000	-0.040035
traffic_volume	0.018676	0.130034	0.004714	0.000735	-0.040035	1.000000

In [41]:
```
sns.heatmap(cor)
```

Out[41]: <AxesSubplot:>

```
In [38]: 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)

In [39]: from sklearn import metrics

In [40]: 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))

-5.5172854236368565
1.0
0.9748447506237436
-12.188104231382285
0.8349874938269883
```

with testing data finding the r-score

```
In [41]: 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 [42]: 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))
```

```
In [42]: 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))

-5.399396398322173
0.6939120482318348
0.8064302676608074
-11.972215715232434
0.7922184852381723
```

Randforest gives the best r-score value

```
In [43]: #RMSE values
         MSE = metrics.mean_squared_error(p3,y_test)

In [44]: np.sqrt(MSE)

Out[44]: 792.9086927553888
```

saving the model

```
In [45]: import pickle

In [46]: pickle.dump(Rand,open("model.pkl",'wb'))
         pickle.dump(le,open("encoder.pkl",'wb'))

In [47]: data.head()

Out[47]:
```

jupyter mini traffic volume-Copy1 Last Checkpoint: Last Thursday at 10:29 AM (unsaved changes)

In [43]:
```
#RMSE values
MSE = metrics.mean_squared_error(p3,y_test)
```

In [44]:
```
np.sqrt(MSE)
```

Out[44]: 792.9086927553888

saving the model

In [45]:
```
import pickle
```

In [46]:
```
pickle.dump(Rand,open("model.pkl",'wb'))
pickle.dump(le,open("encoder.pkl",'wb'))
```

In [47]:
```
data.head()
```

Out[47]:

	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

app.py

```
import numpy as np
import pickle

import time
import pandas
import os
from flask import Flask, request, render_template


app = Flask(__name__)
model = pickle.load(open(r'model.pkl','rb'))
scale = pickle.load(open('encoder.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. Data Frame (features_ values, columns=names)
     # predictions using the loaded model file
    prediction=model. predict(data)
    print(prediction)
    text = "Estimated Traffic Volume is :"
    return render_template("output.html",result = text + str(prediction) + "units")
     # showing the prediction results in a UI
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
```