CHAPTER 1

INTRODUCTION

Growth in the number of vehicles and degree of urbanization means that the annual cost of traffic jams is increasing in cities. This leads to a decrease in the quality of life among citizens through a considerable waste of time and excessive fuel consumption and air pollution in congested areas. Traffic congestion has been one of the major issues that most metropolises are facing despite measures being taken to mitigate and reduce it. The safe and time-efficient movement of the people and goods is dependent on Traffic flow, which is directly connected to the traffic characteristics. Early analysis of congestion events and prediction of traffic volumes is a crucial step to identify traffic bottlenecks, which can be utilized to assist traffic management centres.Traffic jams on Urban Network are increasing day by day, because the traffic demand increases, and the speed of the vehicles is drastically reduced thus causing longer vehicular queuing and more such cases substantially hamper the traffic flow by giving rise to holdup.

1.1 Motivation

With the progress of urbanization and therefore the recognition of automobiles, transportation problems are becoming more and more challenging: the traffic volume flow is congested, wear n tear of vehicles, delays end in the late time of arrival at the meeting, accidents are frequent, and wastage of fuel while waiting in traffic, the traffic environment is becoming worse, to unravel this problem and to assist society, we've chosen our topic as traffic volume prediction.

1.2 Problem Definition

Now? The question arises of how to improve the capacitor y of the road network. To solve this problem the first solution that occurs to most of us is to build more highways, expanding the number of lanes on the road. However, according to the study done by scholars, expanding the road capacity will cause more serious traffic conditions. Therefore, traffic volume prediction is one of the most famous.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM:

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

2.2 SCOPE:

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

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS;ALGORITHMS USED

3.1 Pre Requisites

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

- **Anaconda navigator**
- **Packages**

  Open anaconda prompt as administrator.

  - Type "pip install numpy" and click enter
  - Type "pip install pandas" and click enter.
  - Type "pip install matplotlib" and click enter.
  - Type "pip install scikit-learn" and click enter.
  - Type "pip install Flask" and click enter.
  - Type "pip install xgboost" and click enter.

3.2 Project Objectives

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- You will able to analyze or get insights into data through visualization.
- Applying different algorithms according to a dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

3.3Project Flow

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

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

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

3.4Data Collection

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

3.5Import Necessary Libraries

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

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

CHAPTER 4

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 Project Structure:

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

4.2 Importing The Dataset

- You might have your data in .csv files, .excel files
- Let’s load a .csv data file into pandas usingread_csv() function.We will need to locate the directory of the CSV file at first (it’s more efficient to keep the dataset in the same directory as your program).

- If your dataset is in some other location, Then
- **Data=pd.read_csv(r"File_location/datasetname.csv")**

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

The output column to be predicted is Traffic volume Based on the input variables we predict the volume of the traffic.

Analyse the Data

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

4.3 Handling the Missing values

1. The Most important step in data pre-processing is dealing with missing data, the presence of missing data in the dataset can lead to low accuracy.

2. Check whether any null values are there or not. if it is present then the following can be done.
There are missing values in the dataset, we will fill the missing values in the columns.

3. We are using mean and mode methods for filling the missing values

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

4.4 Data Visualization:

Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.

Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.

➢ To visualize the dataset we need libraries called Matplotlib and Seaborn.

➢ The Matplotlib library is a Python 2D plotting library that allows you to generate plots, scatter plots, histograms, bar charts etc.

Let's visualize our data using Matplotlib and seaborn library.

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

xlabel: Set the label for the x-axis.

ylabel: Set the label for the y-axis.

title: Set a title for the axes.

Legend: Place a legend on the axes.

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

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

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

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

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

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

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

From the above pair plot, we infer that

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

4.2 BOX PLOT

Box-plot is a type of chart often used in explanatory data analysis. Box plots visually show the distribution of numerical data and skewness through displaying the data quartiles (or percentiles) and averages.

Box plots are useful as they show the average score of a data set. The median is the average value from a set of data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.

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

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

4.5 Splitting The Dataset Into Dependent And Independent Variable

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

- With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

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

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

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

4.6 Feature Scaling of X-GRID

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

4.7 Splitting the Data Into Train And Test

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test datasets. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to the training set and the remaining 20% to test.
- Now split our dataset into train set and test using train_test_split class from sci-kit learn library.

4.8 Training and Testing The Model

- Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.

- There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be classification algorithms are regression algorithms.

  1.Linear Regressor
  2.DecisionTreeRegressor
  3.RandomForestRegressor
  4.KNN
  5.svm
  5.xgboost

We're going touse the x-train and y-train obtained above in the train_test_split section to train our Random forest regression model. We're using the fit method and passing the parameters as shown below.

We are using the algorithm from Scikit learn library to build the model as shown below,

Once the model is trained, it's ready to make predictions. We can use the **predict** method on the model and pass **x_test** as a parameter to get the output as **y_pred.**

Notice that the prediction output is an array of real numbers corresponding to the input array.

4.9 Model Evaluation

Duration: 0.5 Hrs

Skill Tags:

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

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

- R-square_score
- RMSE – root mean squared error

❍ **R-squared _score -**

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

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

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

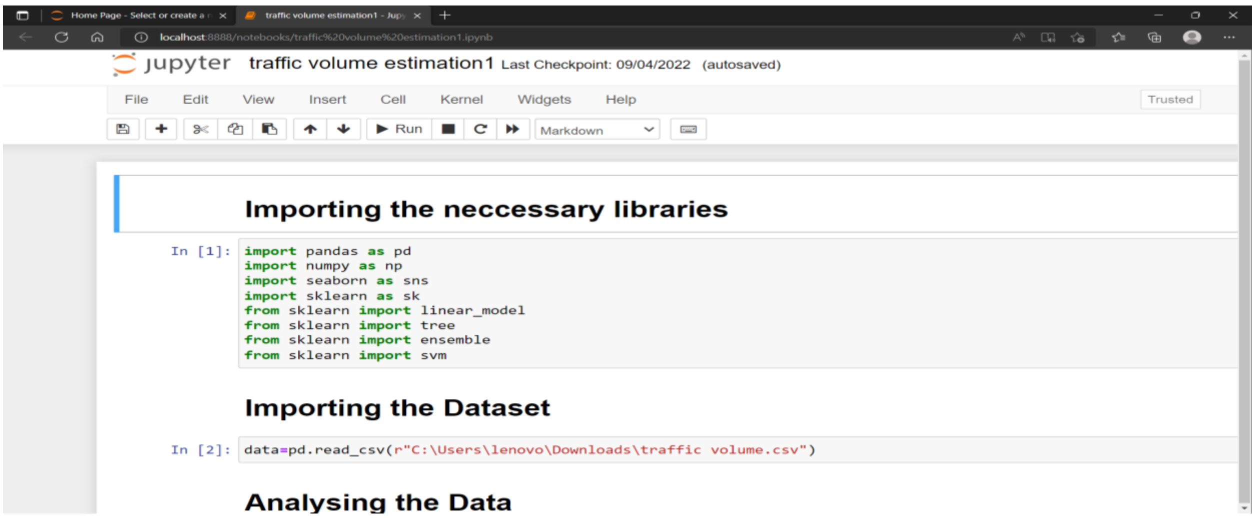
4.10 Save the Model

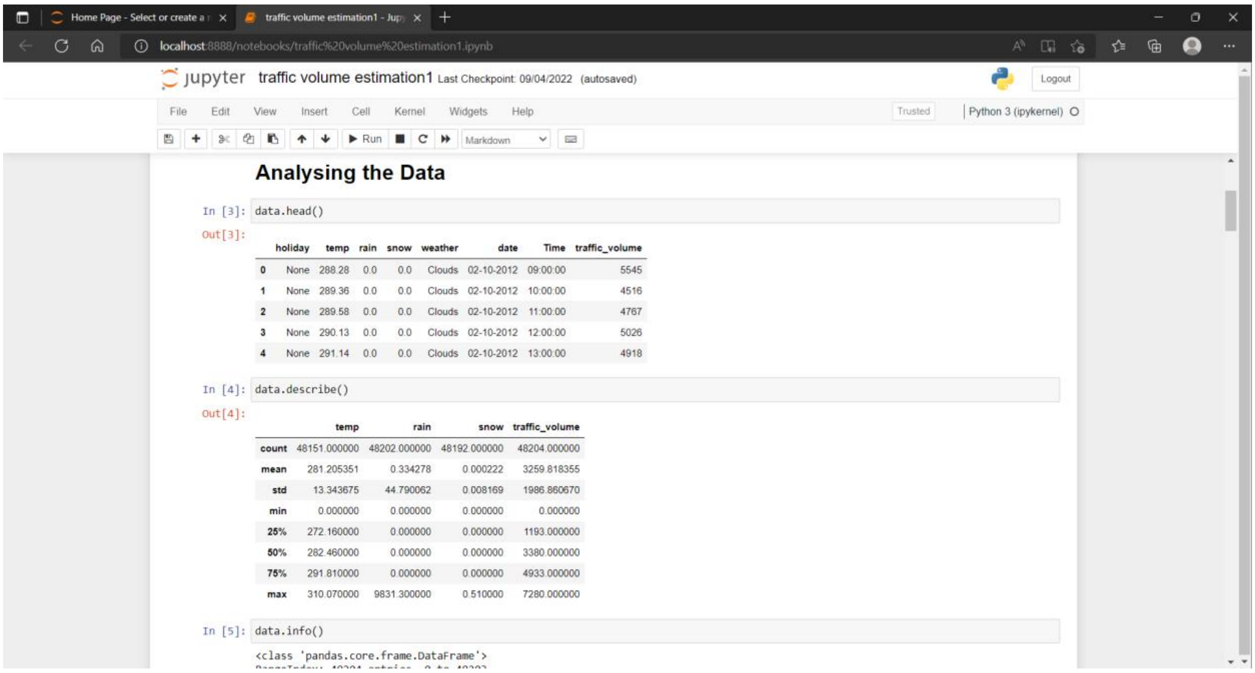
After building the model we have to save the model.

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

4.11 APPENDIX

A.SCREENSHOTS





Analysing the Data

In [3]:
```
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

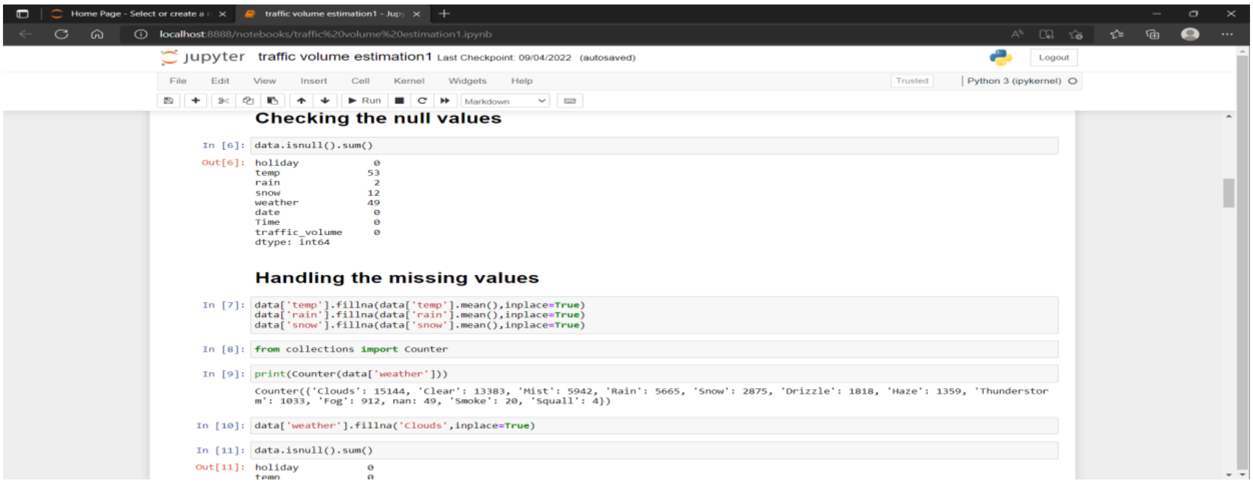
In [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```



Checking the null values

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
holiday           0
temp             53
rain              2
snow             12
weather          49
date              0
Time              0
traffic_volume    0
dtype: int64
```

Handling the missing values

In [7]:
```
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 [8]:
```
from collections import Counter
```

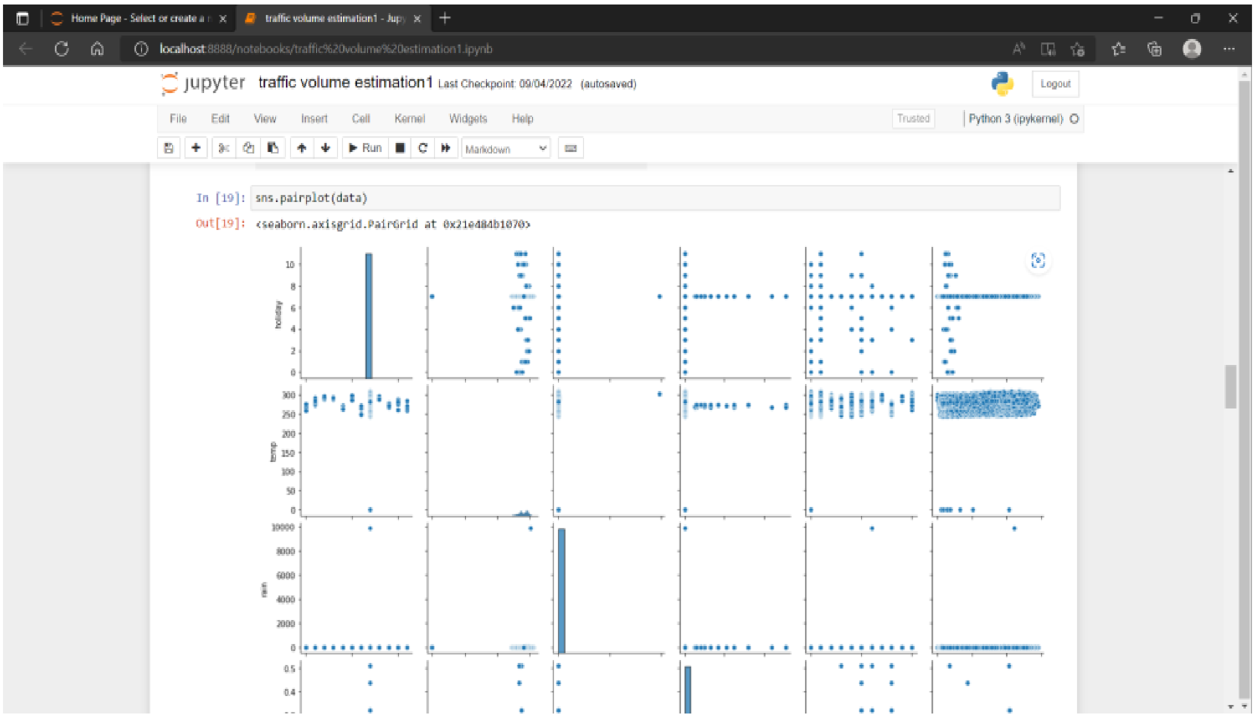
In [9]:
```
print(Counter(data['weather']))
```

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

In [10]:
```
data['weather'].fillna('Clouds',inplace=True)
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
holiday           0
```



In [19]:
```
sns.pairplot(data)
```

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x21e484b1070>
```

localhost:8888/notebooks/traffic%20volume%20estimation1.ipynb

jupyter traffic volume estimation1 Last Checkpoint: 09/04/2022 (autosaved)

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

Splitting Date and Time

```
In [22]: data[["day","month","year"]] = data["date"].str.split("-", expand = True)

In [23]: data[["hours", "minutes", "seconds"]] = data["Time"].str.split(":", expand = True)

In [24]: data.drop(columns=['date','Time'],axis=1,inplace=True)

In [25]: data.head()
```

Out[25]:

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

Splitting The Dataset Into Dependent And Independent Variable

Splitting The Dataset Into Dependent And Independent Variable

```
In [26]: y = data['traffic_volume']
         x = data.drop(columns=['traffic_volume'],axis=1)

In [27]: names = x.columns
```


localhost:8888/notebooks/traffic%20volume%20estimation1.ipynb

2	7	289.58	0.0	0.0	1	4767	02	10	2012	11	00	00
3	7	290.13	0.0	0.0	1	5026	02	10	2012	12	00	00
4	7	291.14	0.0	0.0	1	4918	02	10	2012	13	00	00

Splitting The Dataset Into Dependent And Independent Variable

```
In [26]: y = data['traffic_volume']
         x = data.drop(columns=['traffic_volume'],axis=1)

In [27]: names = x.columns
```

Feature scaling

```
In [28]: from sklearn.preprocessing import scale

In [29]: x = scale(x)

In [30]: x = pd.DataFrame(x,columns=names)

In [31]: x.head()
```

Out[31]:

	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

Splitting The Data Into Train And Test

In [32]:
```
from sklearn.model_selection import train_test_split
```

In [33]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state =0)
```

Home Page - Select or create a | traffic volume estimation1 - Jupy

localhost:8888/notebooks/traffic%20volume%20estimation1.ipynb

jupyter traffic volume estimation1 Last Checkpoint: 09/04/2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Initializing the model

In [34]:
```
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
import xgboost
```

Fitting the models with x_train and y_train

In [35]:
```
lin_reg = linear_model.LinearRegression()
Dtree = tree.DecisionTreeRegressor()
Rand = ensemble.RandomForestRegressor()
svr = svm.SVR()
XGB = xgboost.XGBRegressor()
```

Fitting the models with x_train and y_train

In [36]:
```
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[36]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
```

Predicting the y_train values and calculate the accuracy

In [37]:
```
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)
```

Home Page - Select or create a | traffic volume estimation1 - Jupy

localhost:8888/notebooks/traffic%20volume%20estimation1.ipynb

jupyter traffic volume estimation1 Last Checkpoint: 09/04/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

Run Code

Regression Evaluation Metrics

In [38]:
```
from sklearn import metrics
```

R-squared _score

In [39]:
```
print(metrics.r2_score(p1,y_train))
print(metrics.r2_score(p2,y_train))
print(metrics.r2_score(p3,y_train))
print(metrics.r2_score(p4,y_train))
print(metrics.r2_score(p5,y_train))
```

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

In [40]:
```
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 [41]:
```
print(metrics.r2_score(p1,y_test))
print(metrics.r2_score(p2,y_test))
print(metrics.r2_score(p3,y_test))
print(metrics.r2_score(p4,y_test))
print(metrics.r2_score(p5,y_test))
```

```
-5.399396398322208
0.6929568578898734
0.8025389977869495
```

RMSE –Root Mean Square Error

In [42]:
```
MSE = metrics.mean_squared_error(p3,y_test)
```

In [43]:
```
np.sqrt(MSE)
```

Out[43]:
```
799.8784771647161
```

Saving the Model

In [44]:
```python
import pickle
```

In [45]:
```python
pickle.dump(Rand,open("model.pkl",'wb'))
pickle.dump(le,open("encoder.pkl",'wb'))
```

In [ ]:
```python

```