PROJECT PROFILE

❖ Project Profile

Project Name: TRAFFIC PREDICTION FOR INTELLIGENT TRANSPORTATION SYSTEM USING MACHINE LEARNING

Type of Application: WEB APPLICATION

Project Description: Traffic congestion is defined as the state on transport which is characterized by slower speeds of vehicles this is also because of the bad condition of the roads, weather, concern zone, temperature, etc. This traffic flow prediction is mainly based on the real time dataset which is collected with the help of various cameras and sensors. In recent day the deep learning concepts has dragged the attention for the detection of traffic flow predictions.

In this project some of the common and familiar machine learning concepts like Random Forest (RF) and SVR (Support Vector Regression) are applied on the dataset for the traffic flow predictions. The important attributes of weather, temperature, zone name, and day are used to predict the traffic flow of the particular zone. The performance of the proposed system can be evaluated by using accuracy.

Team Size: Two

Front End: Tkinter, SUMO

Back End: Python

Tools used: ANACONDA NAVIGATOR

INTRODUCTION TO TOOLS

❖ Introduction to Tools

- **Front End Tool:**

1. Tkinter:

Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it’s the only framework built into the Python standard library.

GUI is nothing but a desktop app that provides you with an interface that helps you to interact with the computers and enriches your experience of giving a command (command-line input) to your code. They are used to perform different tasks in desktops, laptops, and other electronic devices, etc

This Python framework provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of ‘graphical control elements’, aka widgets, for building application interfaces.

For example, a button widget can accept mouse clicks, and can also be programmed to perform some kind of action, such as exiting the application.

2. <u>SUMO (Simulation of Urban Mobility):</u>

SUMO is an open source traffic simulation package including net import and demand modelling components. We describe the current state of the package as well as future developments and extensions. SUMO helps to investigate several research topics e.g. route choice and traffic light algorithm or simulating vehicular communication. Therefore the framework is used in different projects to simulate automatic driving or traffic management strategies.

SUMO road networks can be either generated using an application named “netgen” or generated by importing a digital road map. The road network importer “netconvert” allows to read networks from other traffic simulators as VISUM, Vissim, or MATsim. It also reads other common formats, as shapefiles or Open Street Map. Due to the lack of applications, the support for TIGER networks was dropped. But TIGER networks are also available as shapefiles and were included in the OSM data base. Besides these formats, “netconvert” is also capable to read less known formats, as RoboCup network format, or openDRIVE.

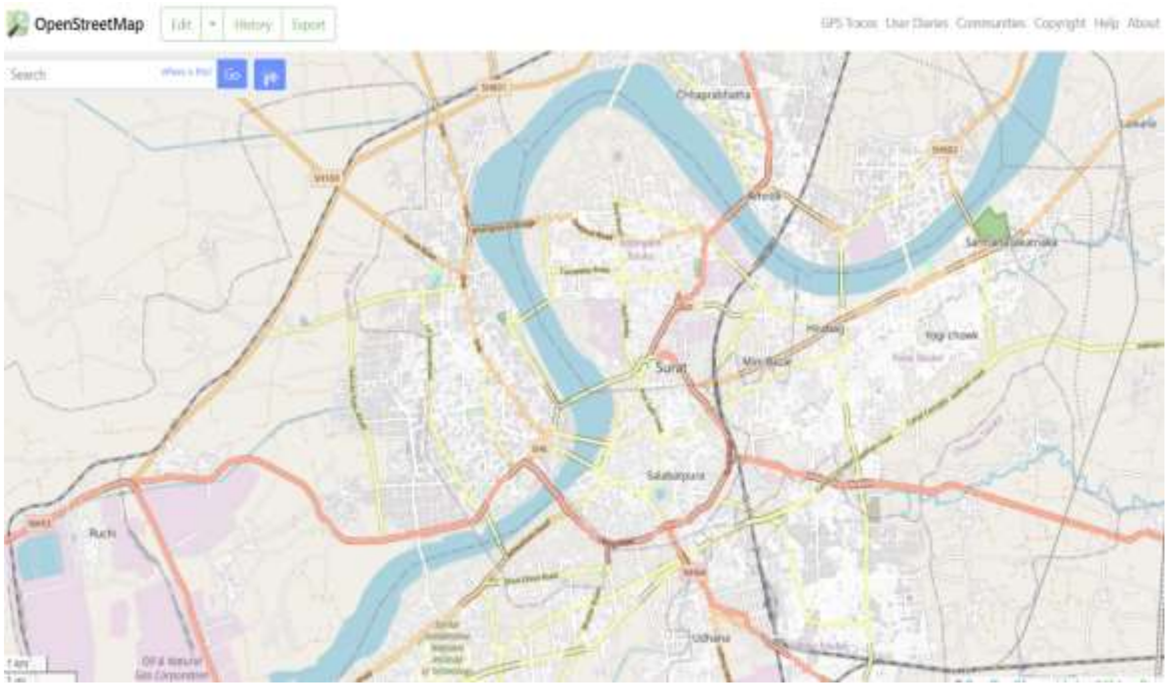
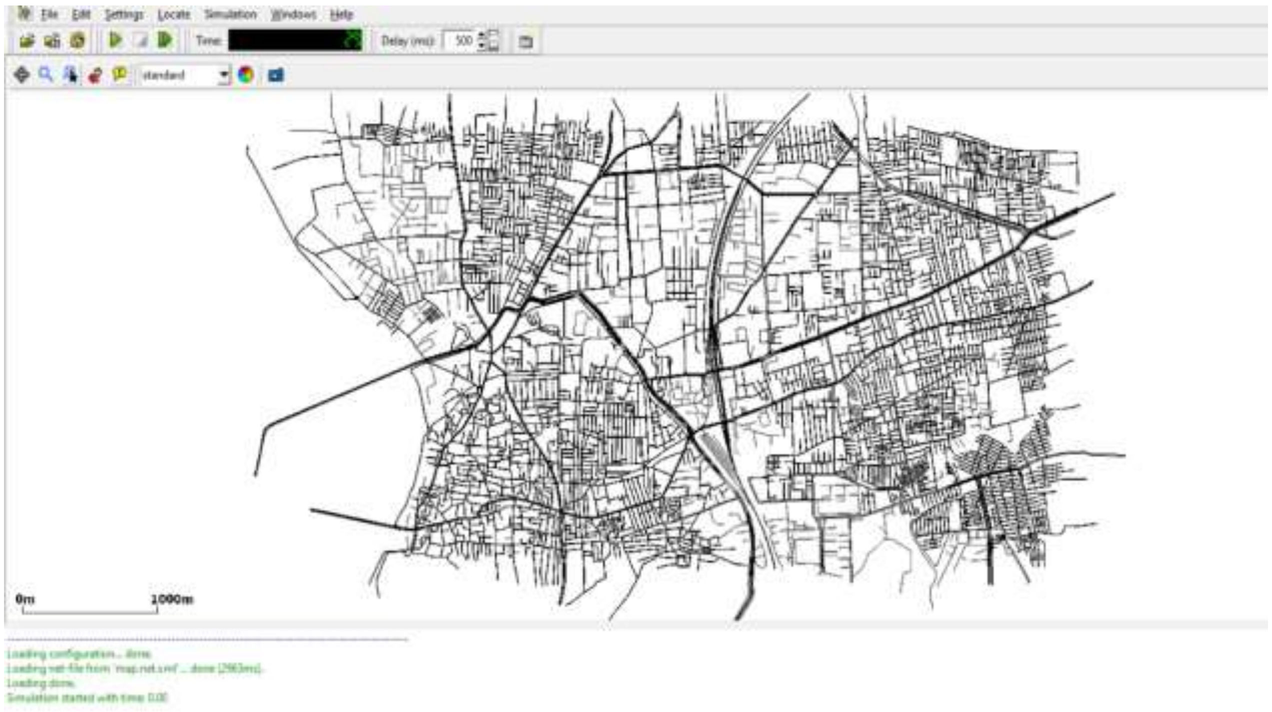


Figure 2.1



```
Loading configuration... done.
Loading net-file from 'map.net.xml' ... done (2963ms).
Loading done.
Simulation started with time: 0.00
```

Figure 2.2

➢ Back End Tool:

1. Python:

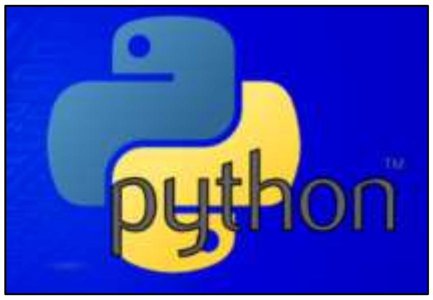
Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++.

The language founded in the year 1991 by the developer Guido Van Rossum has the programming easy and fun to do.

The Python is widely used in bigger organizations because of its multiple programming paradigms. They usually involve imperative and object-oriented functional programming. It has a comprehensive and large standard library that has automatic memory management and dynamic features.

Python is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools/libraries.

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Many developers have also used Python to build productivity tools, games, and desktop apps, so there are plenty of resources to help you learn how to do those as well.



SYSTEM STUDY

1. Existing System :

- In existing system, regardless of vehicles will increase on roads, the traffic too will increase and therefore the accessible road network capability isn't possible to handle this more load. There are 2 attainable approaches to resolve this issue. Traffic flow refers to the quantity of vehicles passing through a given purpose on the route during a bound amount of your time. Most of the time, because of factors like geographical location, traffic conditions, driving time, atmosphere and private circumstances of the driver, every vehicle on the route can have a speed that's somewhat totally different from those around it.

2. Proposed System:

❖ It is very important problem in data analysis. Here we are using machine learning and genetic algorithm. These proposed algorithm gives the much more efficiency than the above system. From the dataset it modifies the different issues. It’s also helpful in the perspective of environment friendliness to reduce carbon emission, communication technology to provide traveller information to increase the safety and efficiency of the road transportation systems.

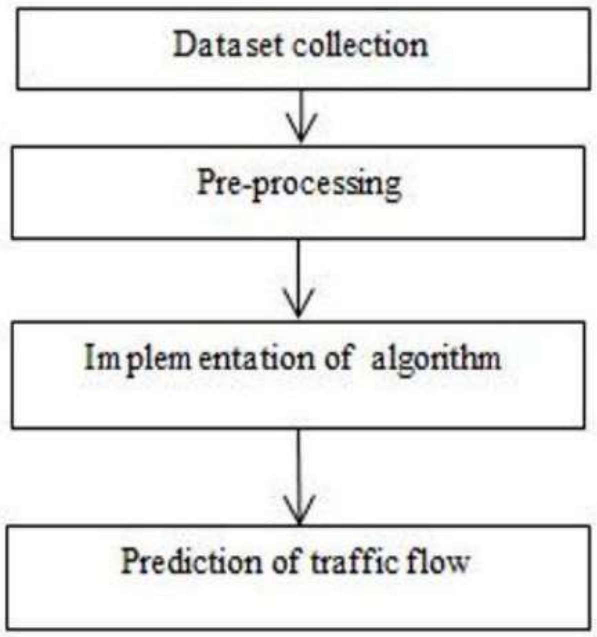


Figure 2.1

3. Scope of the Proposed System:

- In general, traffic prediction studies can be categorized into three major categories: naïve methods, parametric methods and non-parametric methods. Naïve methods are usually simple non-model baseline predictors, which can sometimes return good results. Parametric models are based on traffic flow theory and are researched separately and in parallel to non-parametric, more data-driven machine learning methods. A strong movement towards non-parametric methods can be observed in the recent years, probably because of increased data availability, the progress of computational power and the development of more sophisticated algorithms.

- Non-parametric does not mean models are without parameters; but refers to model's parameters, which are flexible and not fixed in advance. The model's structure as well as model parameters are derived from data. One significant advantage of this approach is that less domain knowledge is required in comparison to parametric methods, but also more data is required to determine a model. This also implies that successful implementation of data-driven models is highly correlated to the quality of available data.

- In contrast, traffic in urban areas can be much more dynamic and non-linear, mainly because of the presence of many intersections and traffic signs. In such environments, data-driven machine-learning approaches, such as neural Networks, Random Forests and kNN, can be more appropriate, due to their ability to model highly nonlinear relationships and dynamic processes.

4. Aim and Objective of the Proposed System

4.1 Aim:

4.1.1 The aim is to research different machine learning algorithms capable of producing accurate traffic flow prediction.

- ➢ we have tested three different data driven methods, from a well-known machine learning library scikit-learn (version 0.20) :

1. k Nearest Neighbors (kNN):

❖ K-Nearest Neighbour is one amongst the only Machine Learning algorithms supported supervised Learning technique. K-NN algorithmic program assumes the similarity between the new case/data and offered cases and place the new case into the class that's most like the offered classes. K-NN algorithmic program stores all the offered knowledge and classifies a replacement information supported the similarity. this suggests once new knowledge seems then it are often simply classified into a well suite class by victimization K- NN algorithmic program. K-NN algorithmic program are often used for Regression moreover as for Classification however largely it's used for the Classification issues. K-

NN could be a non-parametric algorithmic program, which suggests it doesn't create any assumption on underlying knowledge.

2. Random Forests:

- The type of supervised learning that Random Forest belongs to is supervised machine learning. it's straightforward and pliant to use, leading to inflated and improved accuracy. Random forest incorporates a wide selection of applications in classification and regression. It achieves the most effective results by combining call trees and reducing error because of biased and variation. Train the info set and apply the classification technique supported it within the projected model. Outliers & null values were eliminated from the dataset once it had been collected. the info preprocessing was completed as a result of this. The dataset is separated into 2 sections once pre-processing, one for train and also the different for testing. The dataset is split into 3 parts: thirty three % for coaching and seventy seven % for testing. The accuracy it had been anticipated by analysing the strategy with coaching and testing sets. The Random Forest rule is tested mistreatment Google Colab and also the Python programing language.

4.1.2 Which technique is best?

1. How does the KNN algorithm work ?

➢ The below image shows the circles and rounded rectangles spread all over the graph. These represent two dominant classifications of data records in a dataset. For the sake of simplicity, let us assume that these are the only two classifications available in the dataset. Now, what if you have a test data record, the red diamond, and you want to find out which of these classes does this test sample belong to?

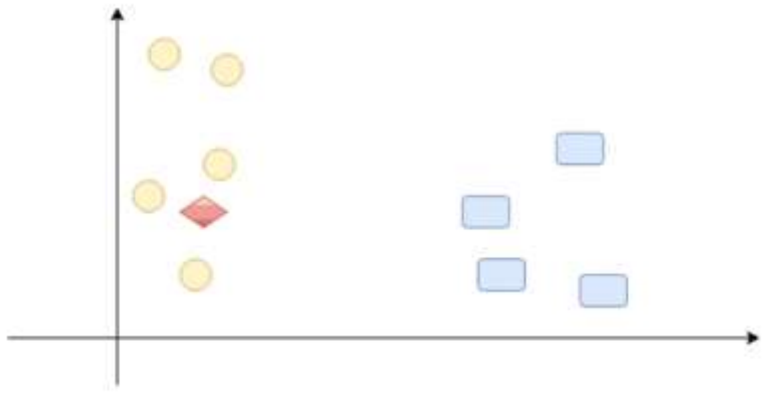


Figure 4.1.2

The “k” in k-NN algorithm is the count of nearest neighbors we wish to take a vote from. Let’s say k = 3. Hence, we make a circle with the diamond as the centre and just big enough to enclose only three data points on the plane.

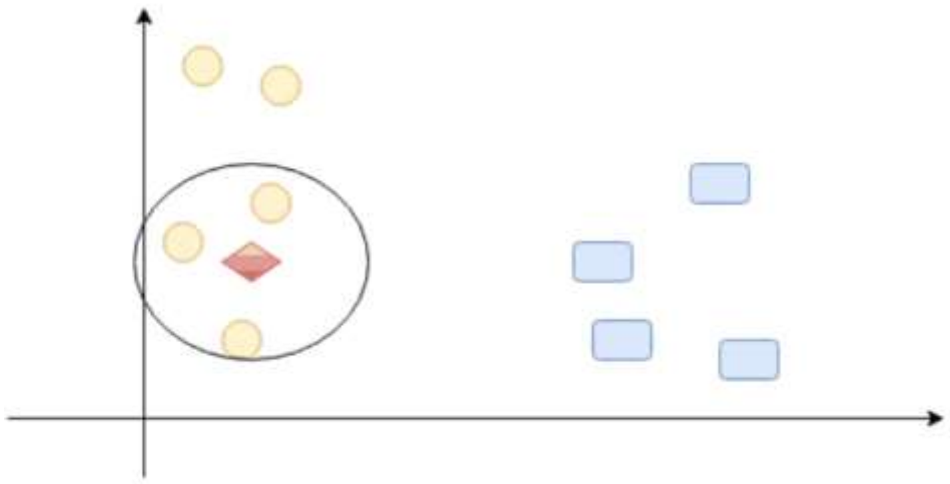


Figure 4.1.2

The three closest points to diamond are all circles, hence with a good level of confidence, we can say that the diamond representing the test sample falls under circles class.

2. How does the Random Forest algorithm work?

- Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

- One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Below you can see how a random forest would look like with two trees:

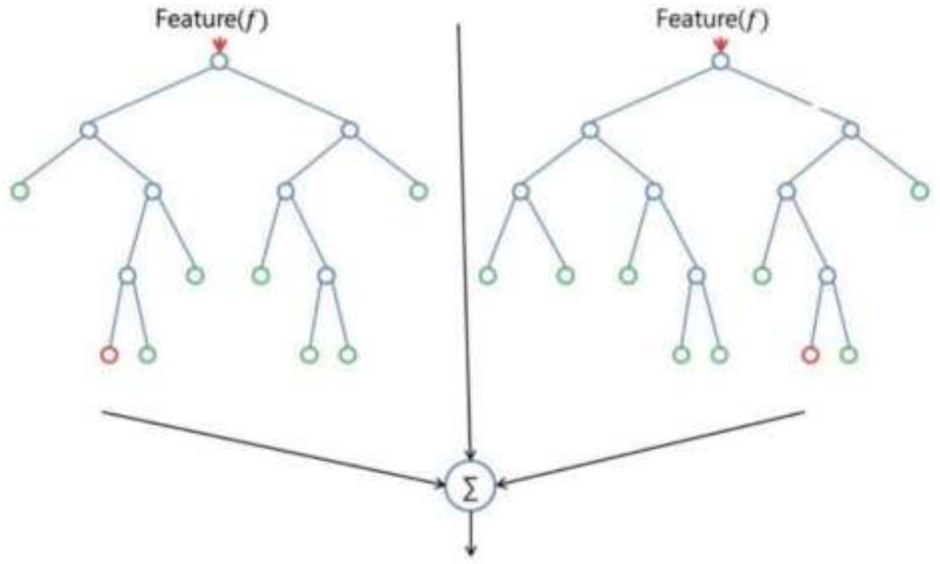


Figure 4.2.1

Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

4.1.3 What is proposed method?

- One such event, which impacts our daily lives, is the act of commuting. As typical city dwellers, most of us commute every day for work, and often we run against the strict schedule. As an example, you need to head to the office tomorrow to attend an important early morning meeting. So, you must start early and also ensure that the duration of commute, or the travel time, to your office is within that safe limit that ensures that you reach in time, neither late, nor too early. What you need is a recommendation system that can suggest you the most favorable moment to leave from home so that you are assured of reaching the office, just in time.

4.1 <u>Objective:</u>

- India is a country of huge population. The Road traffic in all cities of India is of greater concern. There is always a long wait for the people on the roads of the cities. India is among the top countries with large traffic index in the worldand, it is also 4th among the traffic index rankings of 2019 [3]. With high time index and also the C02 (Carbon di oxide) percent among all the cities. So it is important to find effective solutions through ML to solve traffic problem.

5 Feasibility Study :

5.1 **Collection Of Real-Time Data:**

- The type and quality of the data available in real-time plays a crucial role in dictating the possible approaches to our prediction problem. As suggest before, main sources of real-time information include induction loop detectors and surveillance systems, GPSenabled public vehicles and data collected from users in real-time, for instance through smartphones. Cabs and buses often incorporate location systems which can also be used for planning and traffic control. In addition to a source of real-time data, historic information is also essential regardless of the approach. While data driven methods rely on the network's history to predict its evolution, model-driven methods require it in order to calibrate the parameters used in the traffic simulation

5.1 Data Pre-Processing:

- ❖ Pre-processing refers to the transformations applied to our information before feeding it to the algorithm. Data Pre-processing is a method that is used to convert the raw information into a clean data set. In other words, whenever the data is gathered from different origin it is collected in raw format which is not feasible for the analysis. When it comes to creating a Machine Learning model, data pre-processing is the earliest step marking the initiation of the process. Typically, real-world data is incomplete, inconsistent, (contains errors or mistakes), and often lacks specific attribute values/trends. This is where data pre-processing comes the scenario – it helps to clean, format, and organize the raw information, thereby making it ready-to-go for Machine Learning models.

5.2 Data Availability:

- ❖After pre-processing, data availability can be divided in three different groups:

  Static observations: real-time data obtained in fixed locations in the network, e.g. from induction loops or surveillance systems. This often includes metrics such as traffic flow, density, volume, avg velocity, occupancy or queue length at specific points in the network.

  Route observations: usually obtained from GPS enabled cabs and buses, smartphones or traffic operators. This includes metrics such as travel duration, queue length and accidents that are obtained from a random point in the network. Global observations: usually get from secondary source, this data can be as varied as weather reports, special events (e.g. cricket match) or other monitoring which should be considered for prediction purposes, such as the day of the week.

5.3 Prediction Metrics and Targets:

- ❖ Depending on the selected approach and desired output, a large no of variables can be predicted. In its simplest form, the target is the direct prediction of an input variable at some point in the future. Common targets try to define the state of the network at some specific points in terms of vehicle density, traffic flow and avg velocity. Other common yet often derived targets include travel duration and queue length. Other different outputs could be useful for road managers. It is often important to predict propagation on road networks caused by traffic condition changes, weather incidents or road works.

5.4 Measuring Prediction Accuracy :

❖ In order to assess the standard of the traffic prediction systems, it is essential to ascertain metrics that allow the comparison of the various methods. This evaluation must consist on a comparison between the traffic forecast result and the actual traffic conditions at that point in time. The following metrics are the foremost common:

1) Mean Relative Error: Also called Mean Absolute Percent Error (MAPE), Percentage Absolute Error (PAE), Ratio of Absolute Error and accuracy (1 − MAP E). This metric corresponds to the avg absolute percentage change between forecasted and true value, relative to the true value.

2**) Mean Absolute Error (MAE):** This metric corresponds to the average absolute difference between predicted and true values, also called Mean Absolute Deviation.

3) Root Mean Square Error (RMSE): This metric corresponds to the square root of the mean of the square difference between observed and forecasted values.

SYSTEM ANALYSIS

5. SYSTEM ARCHITECTURE:

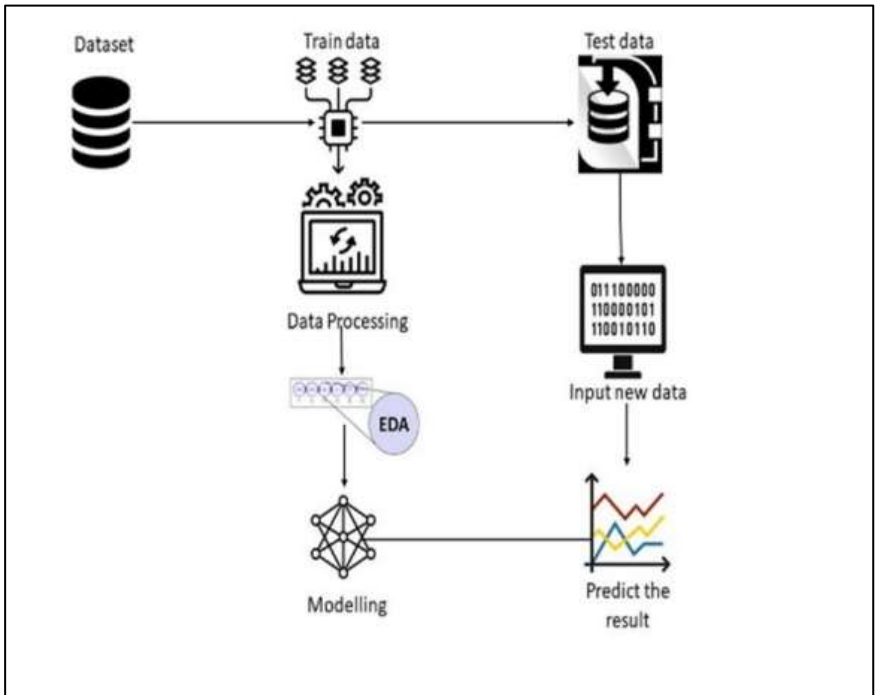


Fig: System Architecture

Figure 5.1

5.1 <u>Data Flow Diagram :</u>

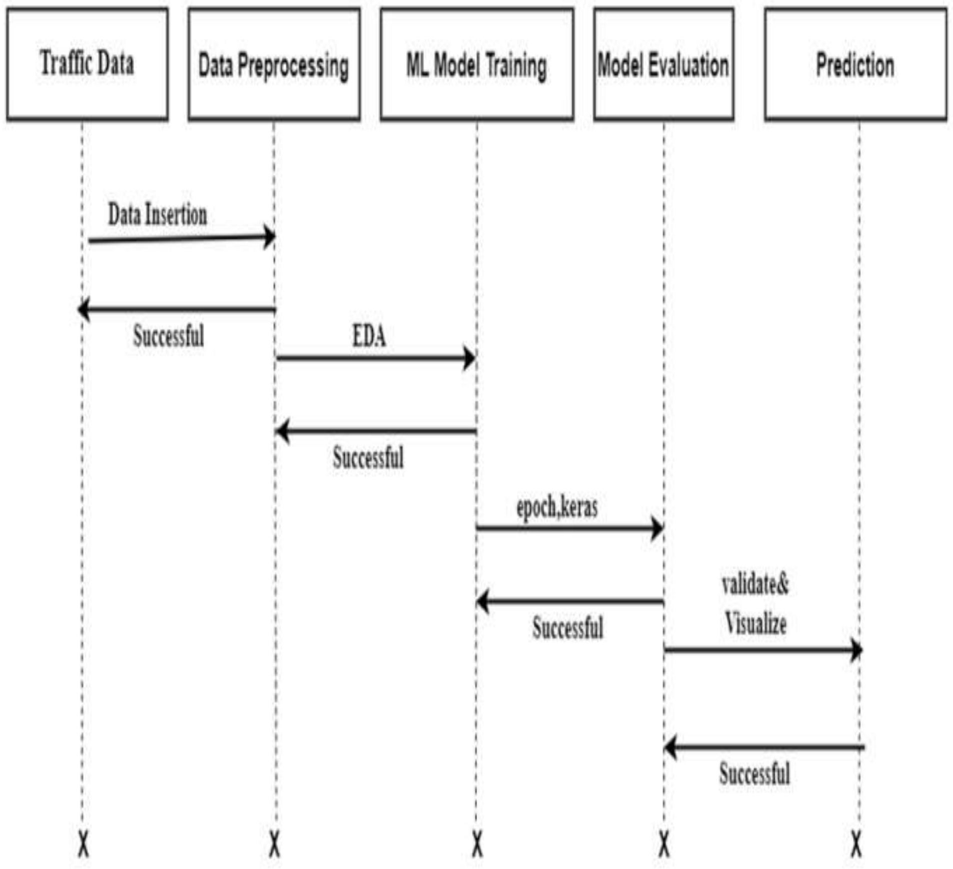


Fig: Flow System

Figure 5.1.1

1. Data types and sources:

❖ Traffic is influenced by many factors, and you should consider all of them to make accurate predictions. So, there are several main groups of data that you'll have to obtain.

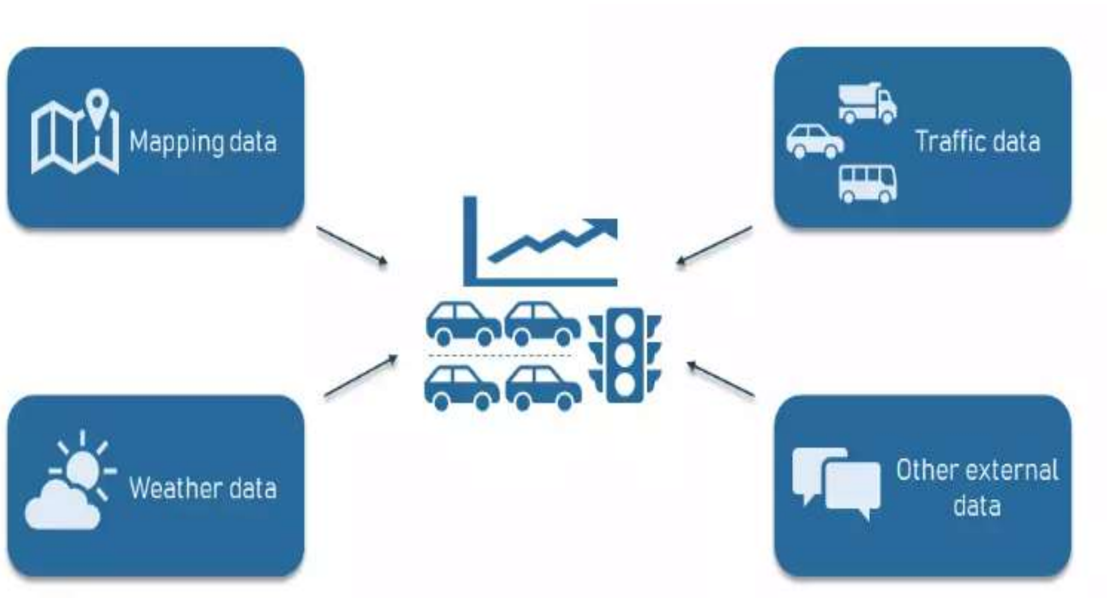


Figure 1.1

2. Mapping data :

❖ First of all, you need to have a detailed map with road networks and related attributes. Connecting to such global mapping data providers as Google Maps, TomTom, HERE, or OSM is a great way to obtain complete and up-to-date information.

3. Traffic information:

❖ Then, you'll have to collect both historical and current traffic-related information such as the number of vehicles passing at a certain point, their speed, and type (trucks, light vehicles, etc.).

4. Weather information:

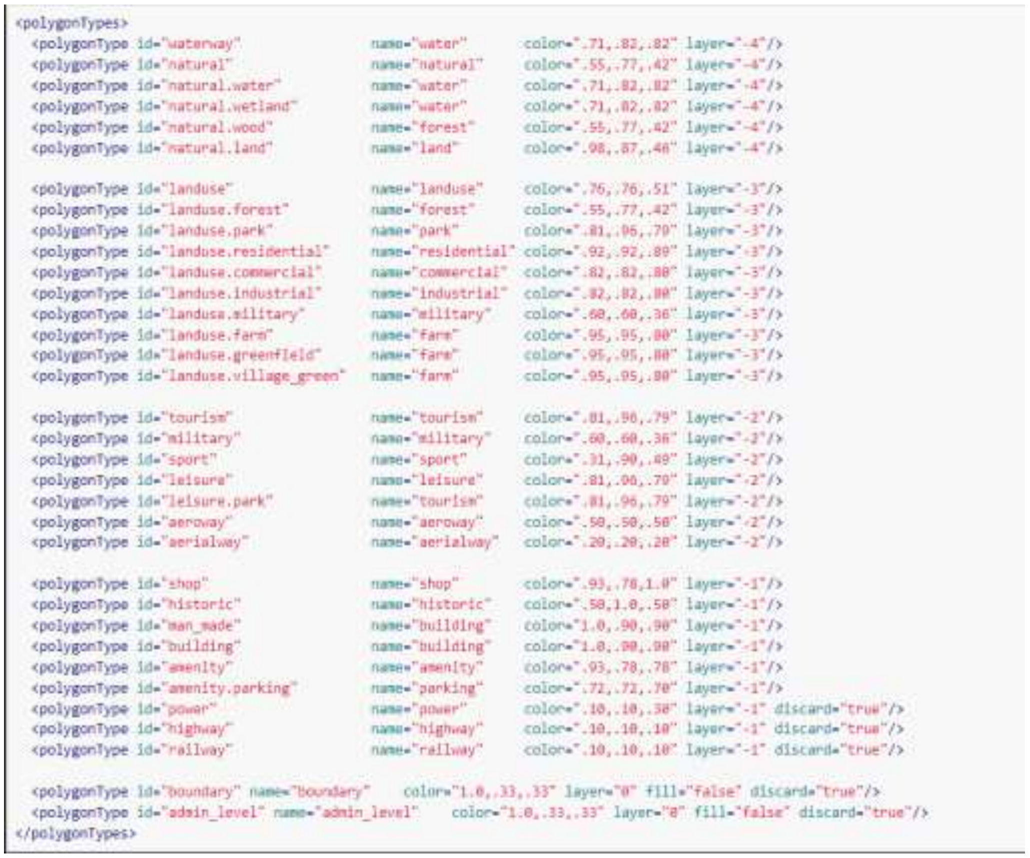
❖ Weather data (historical, current, and forecasted) is also necessary as meteorological conditions impact the road situation and driving speed.

5. **Additional data on road conditions:**
There are external data sources that can provide important information that impacts traffic. Think social media posts about sports events in the area, local news about civil protests, or even police scanners about crime scenes, accidents, or road blockages.

SYSTEM DESIGN

- Execution steps to generate random traffic:

1. Make two folders on your desktop named ”sumo” and ”map”
2. Go to http://sumo.dlr.de/wiki/Networks/Import/OpenStreetMap
3. Scroll down to “Importing additional polygons”

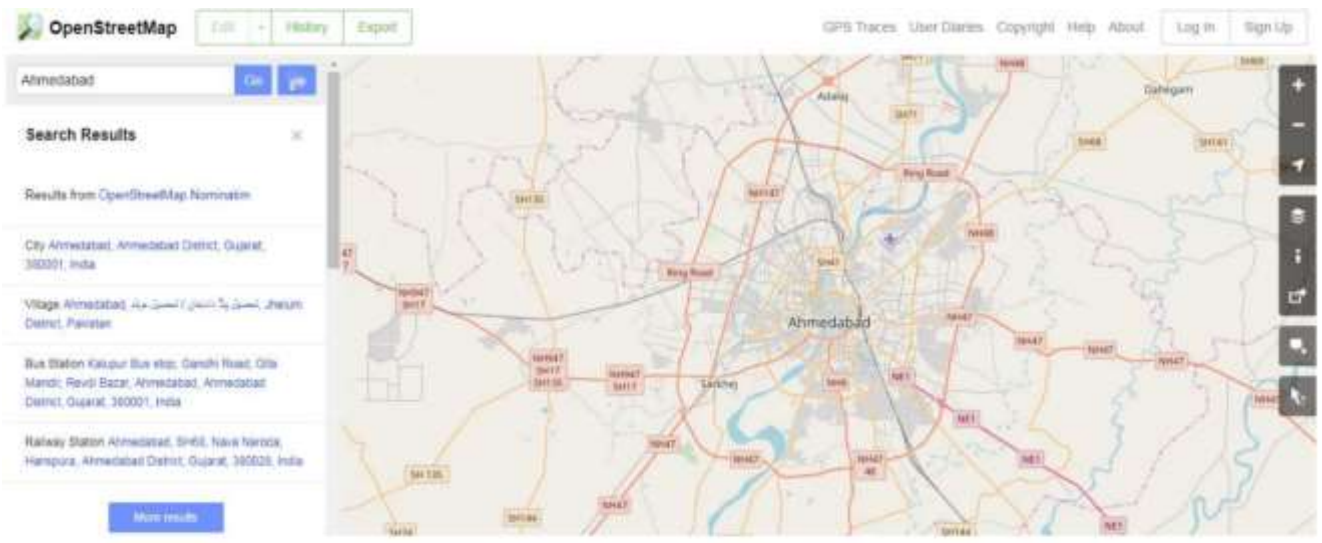


```
<polygonTypes>
  <polygonType id="waterway"              name="water"       color=".71,.82,.82" layer="-4"/>
  <polygonType id="natural"               name="natural"     color=".55,.77,.42" layer="-4"/>
  <polygonType id="natural.water"         name="water"       color=".71,.82,.82" layer="-4"/>
  <polygonType id="natural.wetland"       name="water"       color=".71,.82,.82" layer="-4"/>
  <polygonType id="natural.wood"          name="forest"      color=".55,.77,.42" layer="-4"/>
  <polygonType id="natural.land"          name="land"        color=".98,.87,.46" layer="-4"/>

  <polygonType id="landuse"               name="landuse"     color=".76,.76,.51" layer="-3"/>
  <polygonType id="landuse.forest"        name="forest"      color=".55,.77,.42" layer="-3"/>
  <polygonType id="landuse.park"          name="park"        color=".81,.96,.79" layer="-3"/>
  <polygonType id="landuse.residential"   name="residential" color=".92,.92,.89" layer="-3"/>
  <polygonType id="landuse.commercial"    name="commercial"  color=".82,.82,.80" layer="-3"/>
  <polygonType id="landuse.industrial"    name="industrial"  color=".82,.82,.80" layer="-3"/>
  <polygonType id="landusa.military"      name="military"    color=".60,.60,.36" layer="-3"/>
  <polygonType id="landuse.farm"          name="farm"        color=".95,.95,.80" layer="-3"/>
  <polygonType id="landuse.greenfield"    name="farm"        color=".95,.95,.80" layer="-3"/>
  <polygonType id="landuse.village_green" name="farm"        color=".95,.95,.80" layer="-3"/>

  <polygonType id="tourism"               name="tourism"     color=".81,.96,.79" layer="-2"/>
  <polygonType id="military"              name="military"    color=".60,.60,.36" layer="-2"/>
  <polygonType id="sport"                 name="sport"       color=".31,.90,.49" layer="-2"/>
  <polygonType id="leisure"               name="leisure"     color=".81,.96,.79" layer="-2"/>
  <polygonType id="leisure.park"          name="tourism"     color=".81,.96,.79" layer="-2"/>
  <polygonType id="aeroway"               name="aeroway"     color=".50,.50,.50" layer="-2"/>
  <polygonType id="aerialway"             name="aerialway"   color=".20,.20,.20" layer="-2"/>

  <polygonType id="shop"                  name="shop"        color=".93,.78,1.0" layer="-1"/>
  <polygonType id="historic"              name="historic"    color=".50,1.0,.50" layer="-1"/>
  <polygonType id="man_made"              name="building"    color="1.0,.90,.90" layer="-1"/>
  <polygonType id="building"              name="building"    color="1.0,.90,.90" layer="-1"/>
  <polygonType id="amenity"               name="amenity"     color=".93,.78,.78" layer="-1"/>
  <polygonType id="amenity.parking"       name="parking"     color=".72,.72,.70" layer="-1"/>
  <polygonType id="power"                 name="power"       color=".10,.10,.30" layer="-1" discard="true"/>
  <polygonType id="highway"               name="highway"     color=".10,.10,.10" layer="-1" discard="true"/>
  <polygonType id="railway"               name="railway"     color=".10,.10,.10" layer="-1" discard="true"/>

  <polygonType id="boundary" name="boundary"    color="1.0,.33,.33" layer="0" fill="false" discard="true"/>
  <polygonType id="admin_level" name="admin_level"    color="1.0,.33,.33" layer="0" fill="false" discard="true"/>
</polygonTypes>
```

Figure 3.1

4. Copy the script into Notepad++
5. Remove ”Power” feature from the script
6. Save as ”typemap.xml”

7. Go to https://www.openstreetmap.org: Choose your city or area in which you want to generate traffic.



Ahmedabad

Search Results

Results from OpenStreetMap Nominatim

City Ahmedabad, Ahmedabad District, Gujarat, 380001, India

Village Ahmedabad, [illegible], Jhelum District, Pakistan

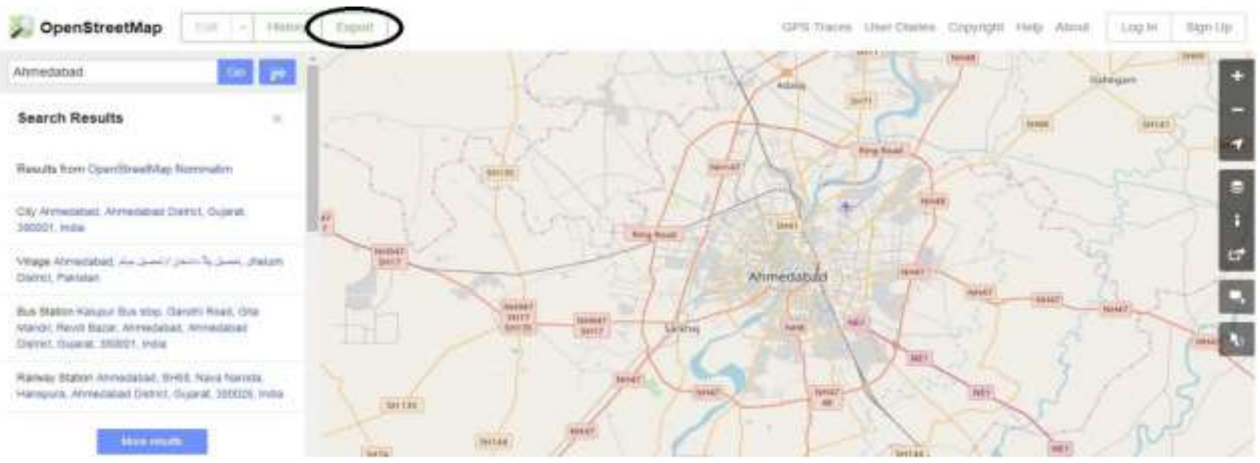
Bus Station Kalupur Bus stop, Gandhi Road, Gita Mandir, Revdi Bazar, Ahmedabad, Ahmedabad District, Gujarat, 380001, India

Railway Station Ahmedabad, SH68, Nava Naroda, Hanspura, Ahmedabad District, Gujarat, 380026, India

More results

Figure 7.1

8. Click on Export.



Ahmedabad

Search Results

Results from OpenStreetMap Nominatim

City Ahmedabad, Ahmedabad District, Gujarat, 380001, India

Village Ahmedabad, [illegible], Jhelum District, Pakistan

Bus Station Kalupur Bus stop, Gandhi Road, Gita Mandir, Revdi Bazar, Ahmedabad, Ahmedabad District, Gujarat, 380001, India

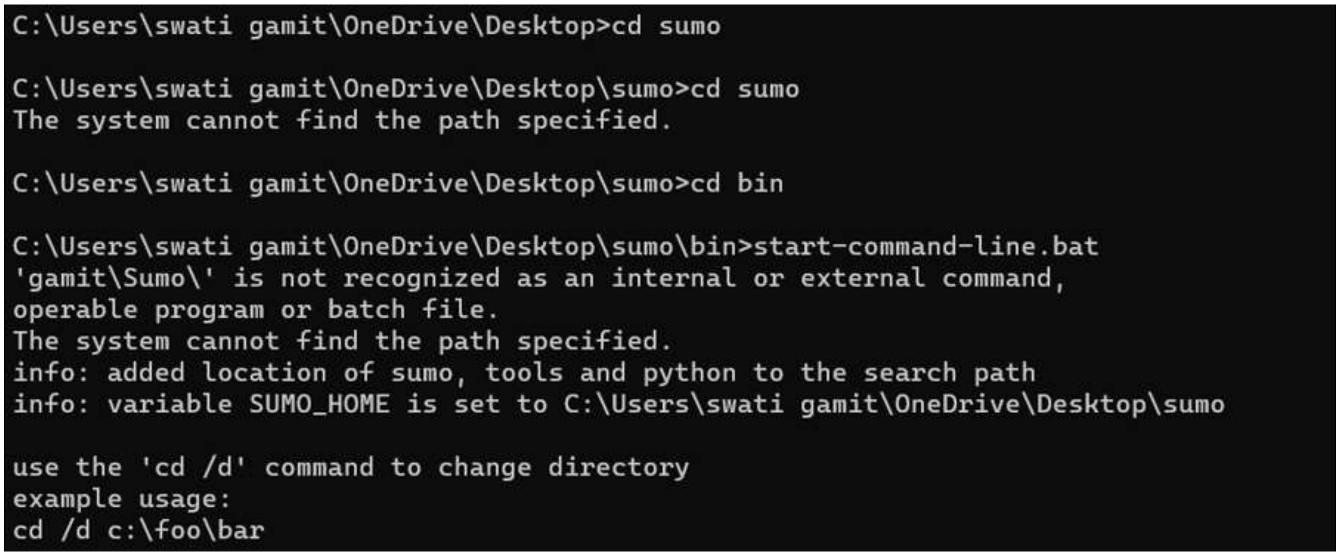
Railway Station Ahmedabad, SH68, Nava Naroda, Hanspura, Ahmedabad District, Gujarat, 380026, India

More results

Figure 8.1

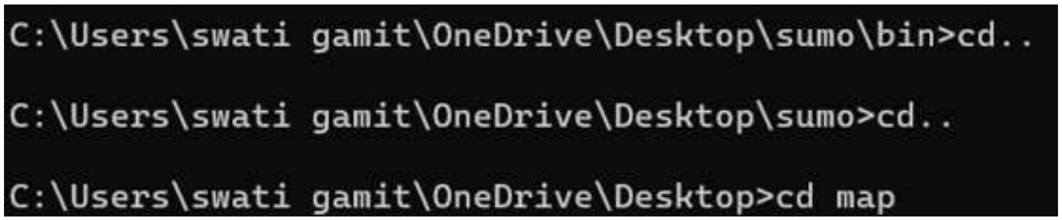
9. Save as “map.osm”

10. Go to the sumo directory: and write: run start-command-line.bat



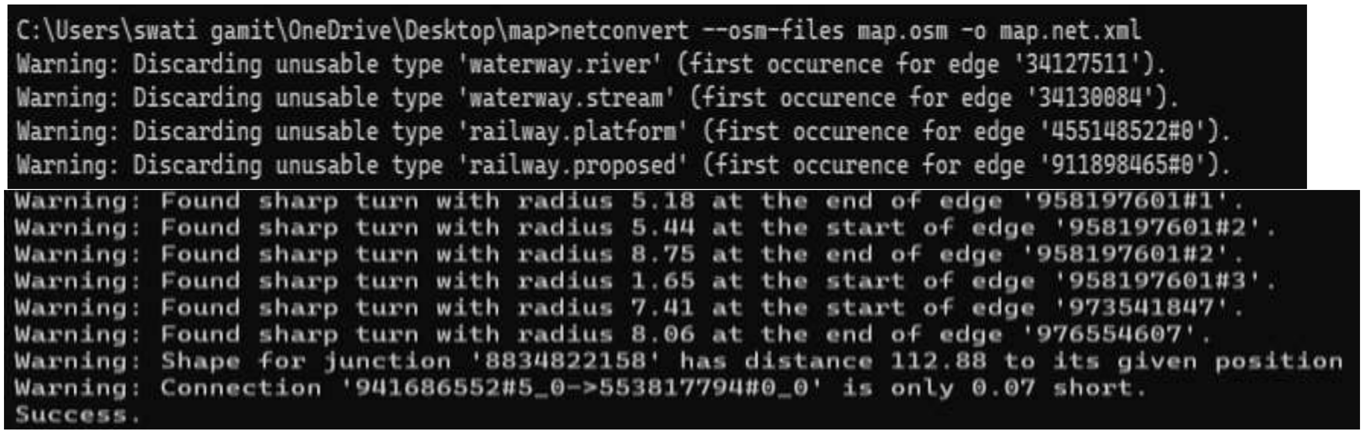
```
C:\Users\swati gamit\OneDrive\Desktop>cd sumo

C:\Users\swati gamit\OneDrive\Desktop\sumo>cd sumo
The system cannot find the path specified.

C:\Users\swati gamit\OneDrive\Desktop\sumo>cd bin

C:\Users\swati gamit\OneDrive\Desktop\sumo\bin>start-command-line.bat
'gamit\Sumo\' is not recognized as an internal or external command,
operable program or batch file.
The system cannot find the path specified.
info: added location of sumo, tools and python to the search path
info: variable SUMO_HOME is set to C:\Users\swati gamit\OneDrive\Desktop\sumo

use the 'cd /d' command to change directory
example usage:
cd /d c:\foo\bar
```

11. Now leave the “SUMO” directory and Go to the map directory.



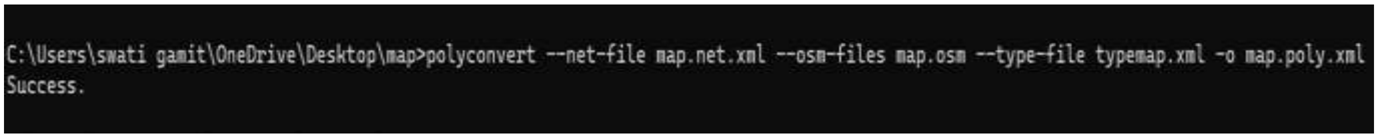
```
C:\Users\swati gamit\OneDrive\Desktop\sumo\bin>cd..

C:\Users\swati gamit\OneDrive\Desktop\sumo>cd..

C:\Users\swati gamit\OneDrive\Desktop>cd map
```

12. Generate “netconvert” file. Type: netconvert –osm-files map.osm -o map.net.xml



```
C:\Users\swati gamit\OneDrive\Desktop\map>netconvert --osm-files map.osm -o map.net.xml
Warning: Discarding unusable type 'waterway.river' (first occurence for edge '34127511').
Warning: Discarding unusable type 'waterway.stream' (first occurence for edge '34130084').
Warning: Discarding unusable type 'railway.platform' (first occurence for edge '455148522#0').
Warning: Discarding unusable type 'railway.proposed' (first occurence for edge '911898465#0').
Warning: Found sharp turn with radius 5.18 at the end of edge '958197601#1'.
Warning: Found sharp turn with radius 5.44 at the start of edge '958197601#2'.
Warning: Found sharp turn with radius 8.75 at the end of edge '958197601#2'.
Warning: Found sharp turn with radius 1.65 at the start of edge '958197601#3'.
Warning: Found sharp turn with radius 7.41 at the start of edge '973541847'.
Warning: Found sharp turn with radius 8.06 at the end of edge '976554607'.
Warning: Shape for junction '8834822158' has distance 112.88 to its given position
Warning: Connection '941686552#5_0->553817794#0_0' is only 0.07 short.
Success.
```

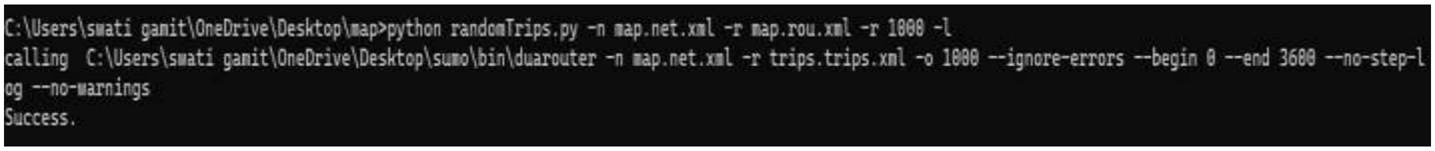
13. Generate ”Polyconvert” file Type: polyconvert –net-file map.net.xml –osm-files map.osm –type-file typemap.xml -o map.poly.xml



```
C:\Users\swati gamit\OneDrive\Desktop\map>polyconvert --net-file map.net.xml --osm-files map.osm --type-file typemap.xml -o map.poly.xml
Success.
```

14.We are looking for random traffic, Go to your sumo folder and search ”randomTraffic.py” and Note the parameters like (Number of vehicles, Simulation time, Route length etc..)

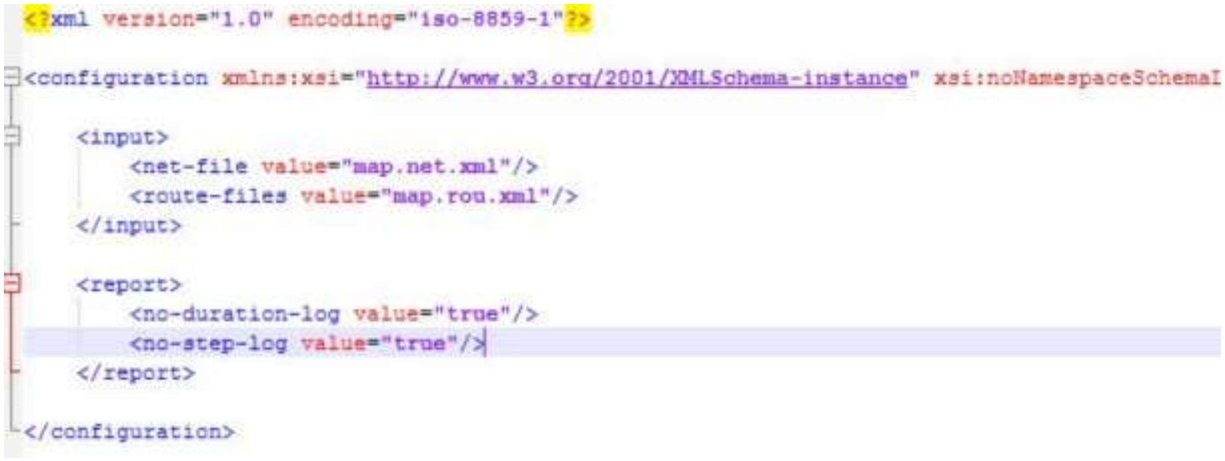
15. Type: python randomTrips.py -n map.net.xml -r map.rou.xml -e 100 -l to generate route file



```
C:\Users\swati gamit\OneDrive\Desktop\map>python randomTrips.py -n map.net.xml -r map.rou.xml -r 1000 -l
calling  C:\Users\swati gamit\OneDrive\Desktop\sumo\bin\duarouter -n map.net.xml -r trips.trips.xml -o 1000 --ignore-errors --begin 0 --end 3600 --no-step-l
og --no-warnings
Success.
```

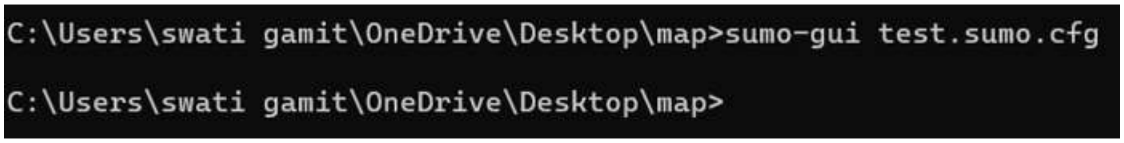
16. Go to the SUMO folder and search for “test.sumo.cfg” and copy that file in “map” folder.

17. Edit "test.sumo.cfg”



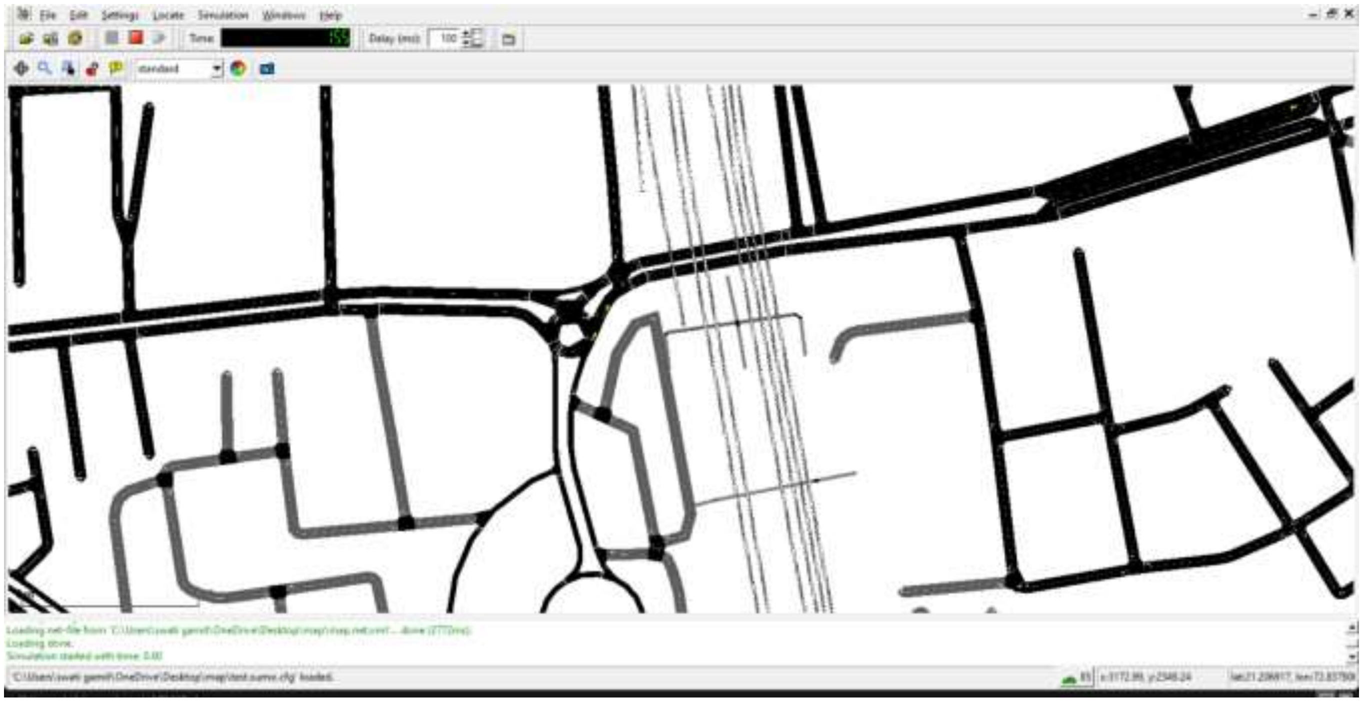
```
<?xml version="1.0" encoding="iso-8859-1"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaL

    <input>
        <net-file value="map.net.xml"/>
        <route-files value="map.rou.xml"/>
    </input>

    <report>
        <no-duration-log value="true"/>
        <no-step-log value="true"/>
    </report>

</configuration>
```

18. Go to command prompt and type: sumo-gui map.sumo.cfg



```
C:\Users\swati gamit\OneDrive\Desktop\map>sumo-gui test.sumo.cfg

C:\Users\swati gamit\OneDrive\Desktop\map>
```

19. Press Enter and SUMO will open



```
Loading net-file from 'C:\Users\swati gamit\OneDrive\Desktop\map\map.net.xml' ... done (2770ms).
Loading done.
Simulation started with time: 0.00
```

```
'C:\Users\swati gamit\OneDrive\Desktop\map\test.sumo.cfg' loaded.
x:3172.99, y:2348.24    lat:21.206817, lon:72.83790
```

20. Note your simulation parameters and save files.

SYSTEM TESTING

6.1 Dataset:

We collected are from the GitHub site for the traffic prediction implementations of machine learning methods to show outputs in this prediction system. The data set have the number of vehicles passes on the road with the help of the road sensors. The traffic data collected by the sensors has recorded every hour from 2012 – 2018 also have the climate data with the help of the temperature. The total number of data set which is collected is 1439.

6.2 Pre-processing and Feature Extraction:

Data pre-processing is the most important phase of a machine learning project. Data that shows the important components that are needed to solve an issue is provided in a way that makes machine learning perform best. Data transformation, data reduction, feature selection, and feature scaling techniques used in feature engineering help reorganize raw data into a format suitable for specific kinds of algorithms. By doing this, the computing power and time needed to train a machine learning or artificial intelligence algorithm can be greatly reduced.

- the observation made for traffic prediction is that many of them used different algorithms like K-Nearest Neighbour (KNN), Recurrent Neural Network/ Artificial Neural Networks (RNN/ANN) and Radial Basis Function (RBF). Among these algorithms the best accuracy achieved by these algorithms for the traffic prediction efficiency is around 87%.

6.3 Random Forest Regression:

- Random Forest Regression Algorithm is the next phase of the Decision tree algorithm where the accuracy of the problem will be enhanced with better methods. As, decision tree algorithm only work with a single tree expansion to solve the problem. But Random Forest Regression will create N number of trees and gives the outputs. Further those output will be taken average to get the best accuracy of the problem. The Random Forest Predictor formula is given in equation

$$f_{rf}^{N}(x) = \frac{1}{N}\sum_{n=1}^{N} T(x;\Theta_n)$$

Where,

N = total number data used for prediction

x = number of iterations done by the rf algorithm

rf = Random Forest

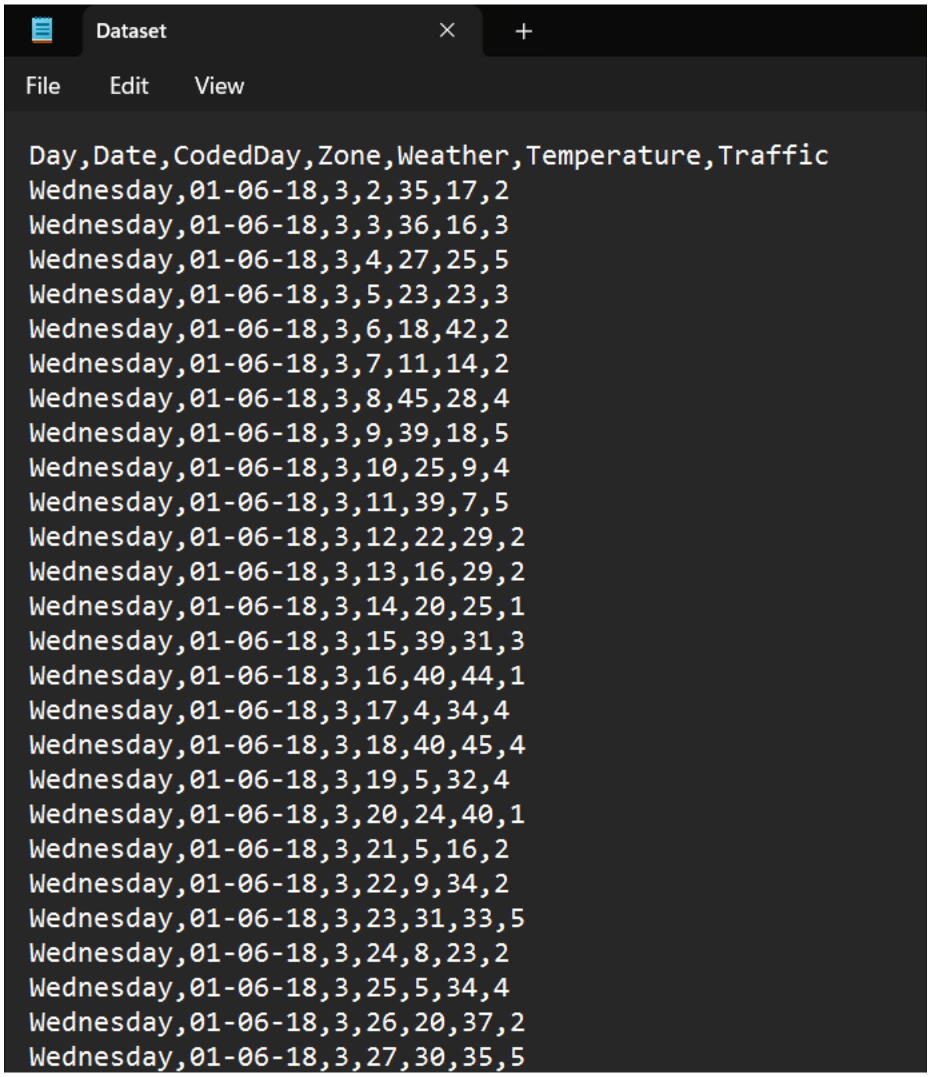
⊖n = characterizes the Nth random forest in terms of

splitting into different decision trees.

- The pre-processed data is divided into training data and test data. Next, the ML algorithm is chosen along with training data. The training data will go under the algorithm process and gives us the output. This predicted output will be compared with the test data. In the final step, the comparison result of the algorithm will give the efficiency of the prediction.

6.4 Screen Layouts:

- Data collection :
  This data is collected from 1st June 2018 to 10th June 2018.



```
Day,Date,CodedDay,Zone,Weather,Temperature,Traffic
Wednesday,01-06-18,3,2,35,17,2
Wednesday,01-06-18,3,3,36,16,3
Wednesday,01-06-18,3,4,27,25,5
Wednesday,01-06-18,3,5,23,23,3
Wednesday,01-06-18,3,6,18,42,2
Wednesday,01-06-18,3,7,11,14,2
Wednesday,01-06-18,3,8,45,28,4
Wednesday,01-06-18,3,9,39,18,5
Wednesday,01-06-18,3,10,25,9,4
Wednesday,01-06-18,3,11,39,7,5
Wednesday,01-06-18,3,12,22,29,2
Wednesday,01-06-18,3,13,16,29,2
Wednesday,01-06-18,3,14,20,25,1
Wednesday,01-06-18,3,15,39,31,3
Wednesday,01-06-18,3,16,40,44,1
Wednesday,01-06-18,3,17,4,34,4
Wednesday,01-06-18,3,18,40,45,4
Wednesday,01-06-18,3,19,5,32,4
Wednesday,01-06-18,3,20,24,40,1
Wednesday,01-06-18,3,21,5,16,2
Wednesday,01-06-18,3,22,9,34,2
Wednesday,01-06-18,3,23,31,33,5
Wednesday,01-06-18,3,24,8,23,2
Wednesday,01-06-18,3,25,5,34,4
Wednesday,01-06-18,3,26,20,37,2
Wednesday,01-06-18,3,27,30,35,5
```

- Time of the day (Zone column):

  - ➢ A number code representing a 10-minute interval time-zone, splitting the 24 hours of a day into 144 zones, (For example, the 10-minute duration from 00:00 to 00:10 Hrs is coded as 1 and 00:10 to 00:20 Hrs is coded as 2, and so on)

- Day of the week (CodedDay column):
  - ➢ Week day in a coded number, 7 weekdays converted into to 7 numbers starting from 1(Sunday) to 7(Saturday).

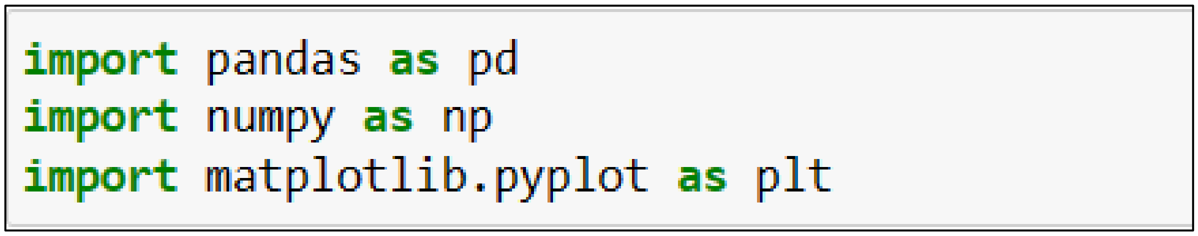
- Weather Conditions (CodedWeather column):
  - ➢ Weather in a coded number. Check out the codes representing weather conditions that are used in this training set.

- Temperature (Temperature column):
  - ➢ Average temperature during the day, in Fahrenheit.

6.4.1Model Construction

(i) Importing Modules

To make use of the functions in a module, you'll need to import the module with an import statement. An import statement is made up of the import keyword along with the name of the module. In a Python file, this will be declared at the top of the code, under any shebang lines or general comments.

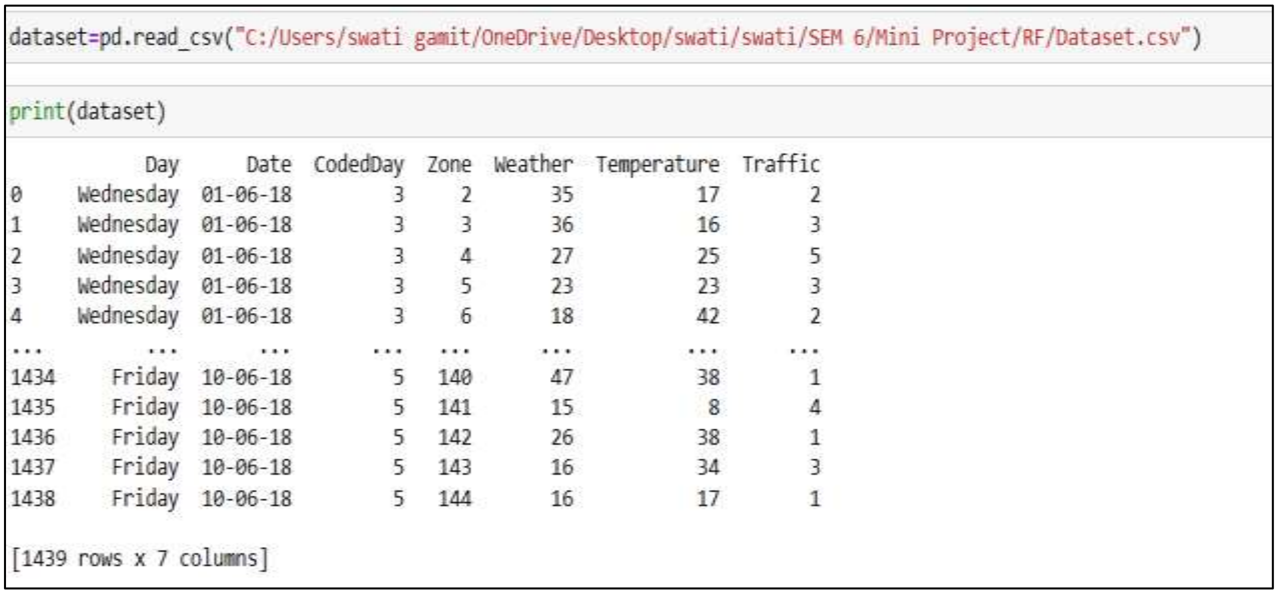


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Install and Load pandas Package

Pandas is a powerful data analysis package. It makes data exploration and manipulation easy. It has several functions to read data from various sources.

(ii)

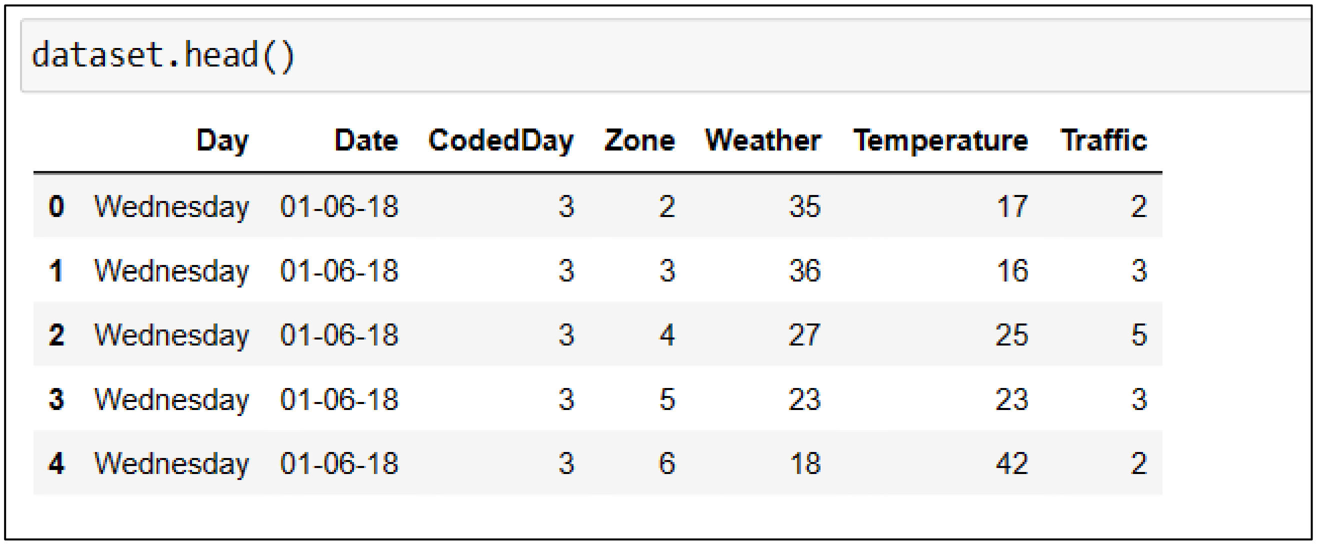


```
dataset=pd.read_csv("C:/Users/swati gamit/OneDrive/Desktop/swati/swati/SEM 6/Mini Project/RF/Dataset.csv")
```

```
print(dataset)
```

```
            Day      Date  CodedDay  Zone  Weather  Temperature  Traffic
0     Wednesday  01-06-18         3     2       35           17        2
1     Wednesday  01-06-18         3     3       36           16        3
2     Wednesday  01-06-18         3     4       27           25        5
3     Wednesday  01-06-18         3     5       23           23        3
4     Wednesday  01-06-18         3     6       18           42        2
...         ...       ...       ...   ...      ...          ...      ...
1434     Friday  10-06-18         5   140       47           38        1
1435     Friday  10-06-18         5   141       15            8        4
1436     Friday  10-06-18         5   142       26           38        1
1437     Friday  10-06-18         5   143       16           34        3
1438     Friday  10-06-18         5   144       16           17        1

[1439 rows x 7 columns]
```

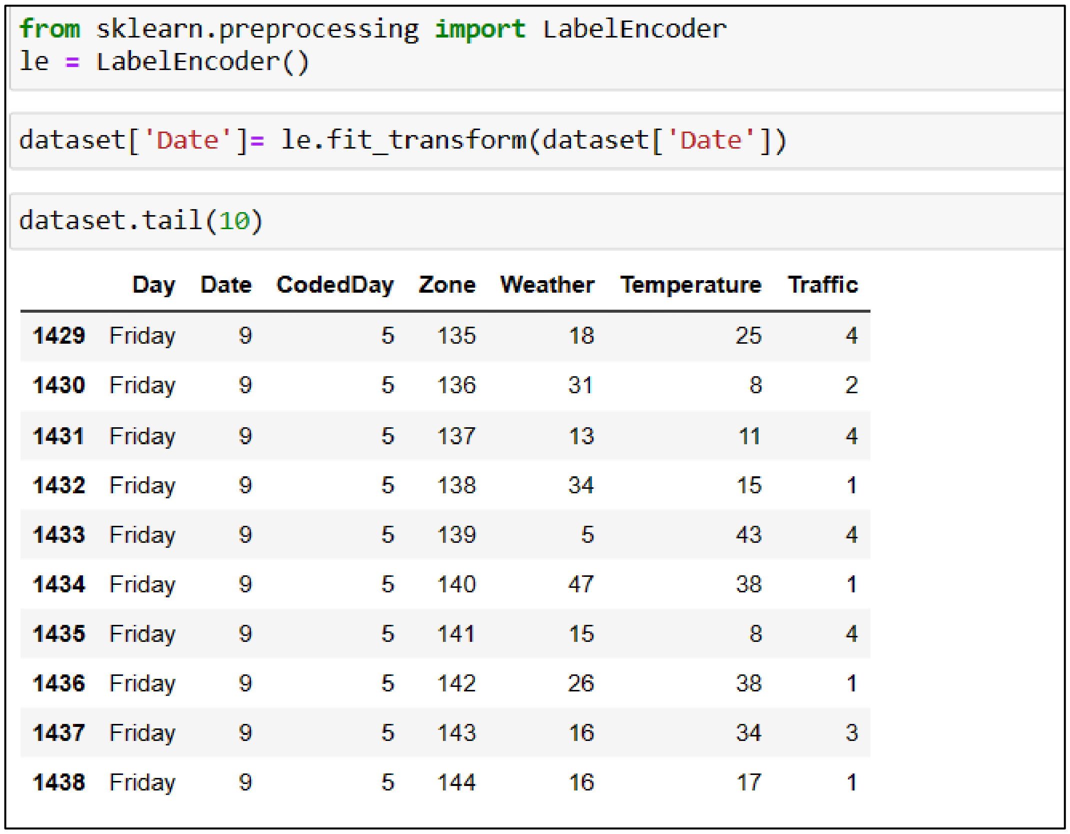
(iii)



```
dataset.head()
```

	Day	Date	CodedDay	Zone	Weather	Temperature	Traffic
0	Wednesday	01-06-18	3	2	35	17	2
1	Wednesday	01-06-18	3	3	36	16	3
2	Wednesday	01-06-18	3	4	27	25	5
3	Wednesday	01-06-18	3	5	23	23	3
4	Wednesday	01-06-18	3	6	18	42	2

(iv)



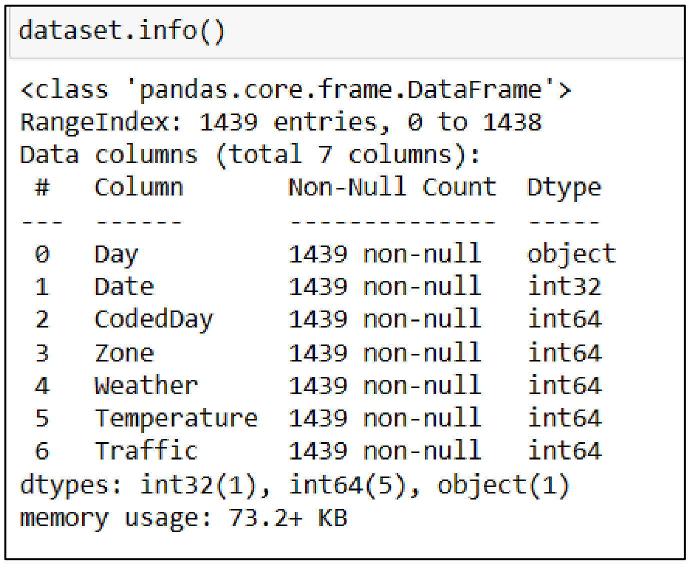
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
dataset['Date']= le.fit_transform(dataset['Date'])
```

```
dataset.tail(10)
```

	Day	Date	CodedDay	Zone	Weather	Temperature	Traffic
1429	Friday	9	5	135	18	25	4
1430	Friday	9	5	136	31	8	2
1431	Friday	9	5	137	13	11	4
1432	Friday	9	5	138	34	15	1
1433	Friday	9	5	139	5	43	4
1434	Friday	9	5	140	47	38	1
1435	Friday	9	5	141	15	8	4
1436	Friday	9	5	142	26	38	1
1437	Friday	9	5	143	16	34	3
1438	Friday	9	5	144	16	17	1

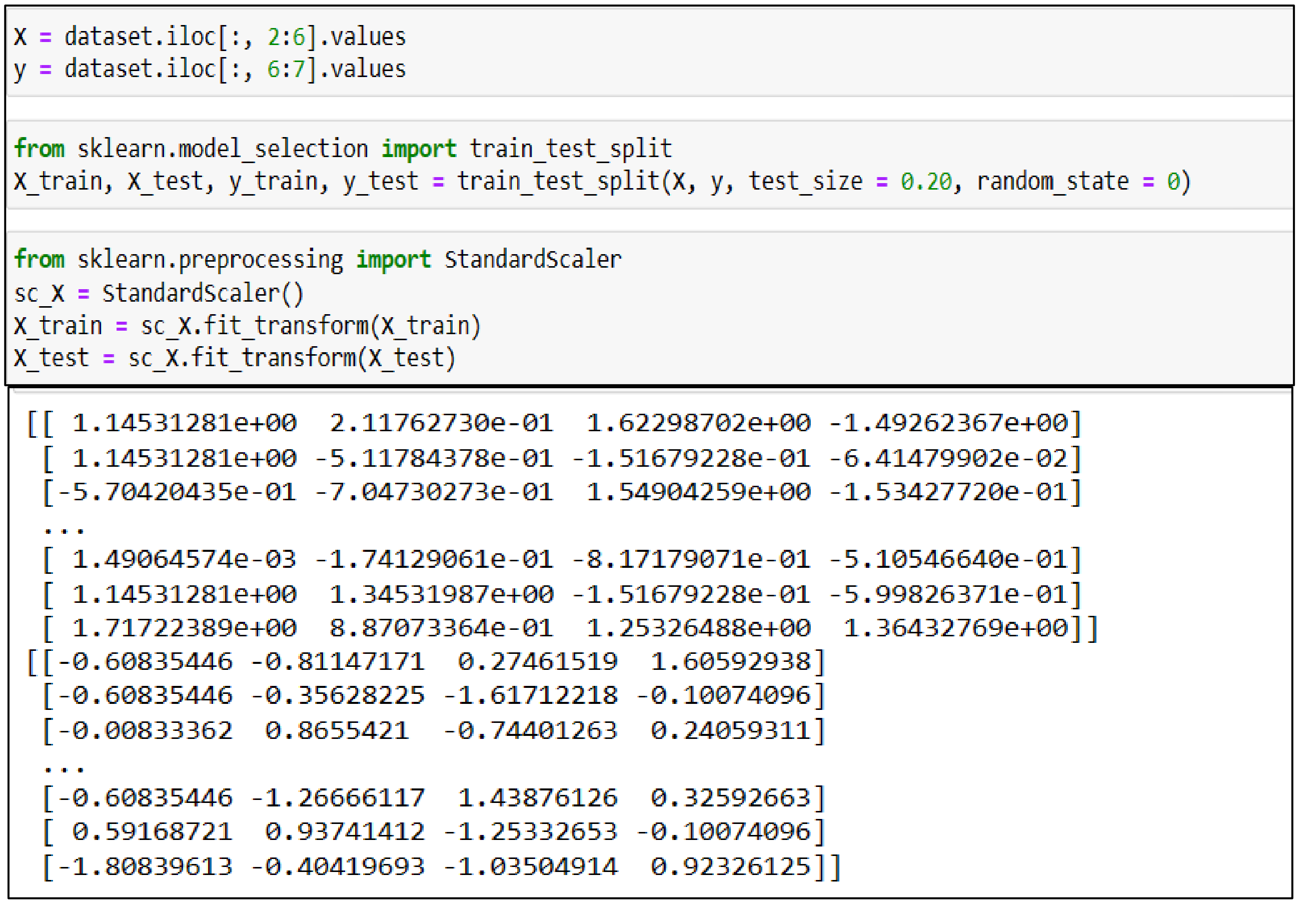
(v)



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1439 entries, 0 to 1438
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Day          1439 non-null   object
 1   Date         1439 non-null   int32 
 2   CodedDay     1439 non-null   int64 
 3   Zone         1439 non-null   int64 
 4   Weather      1439 non-null   int64 
 5   Temperature  1439 non-null   int64 
 6   Traffic      1439 non-null   int64 
dtypes: int32(1), int64(5), object(1)
memory usage: 73.2+ KB
```

(vi)



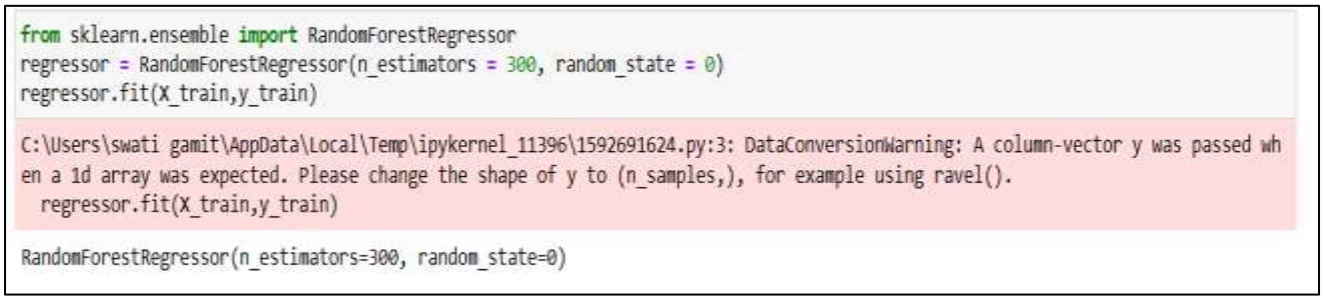
```
X = dataset.iloc[:, 2:6].values
y = dataset.iloc[:, 6:7].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

```
[[ 1.14531281e+00  2.11762730e-01  1.62298702e+00 -1.49262367e+00]
 [ 1.14531281e+00 -5.11784378e-01 -1.51679228e-01 -6.41479902e-02]
 [-5.70420435e-01 -7.04730273e-01  1.54904259e+00 -1.53427720e-01]
 ...
 [ 1.49064574e-03 -1.74129061e-01 -8.17179071e-01 -5.10546640e-01]
 [ 1.14531281e+00  1.34531987e+00 -1.51679228e-01 -5.99826371e-01]
 [ 1.71722389e+00  8.87073364e-01  1.25326488e+00  1.36432769e+00]]
[[-0.60835446 -0.81147171  0.27461519  1.60592938]
 [-0.60835446 -0.35628225 -1.61712218 -0.10074096]
 [-0.00833362  0.8655421  -0.74401263  0.24059311]
 ...
 [-0.60835446 -1.26666117  1.43876126  0.32592663]
 [ 0.59168721  0.93741412 -1.25332653 -0.10074096]
 [-1.80839613 -0.40419693 -1.03504914  0.92326125]]
```

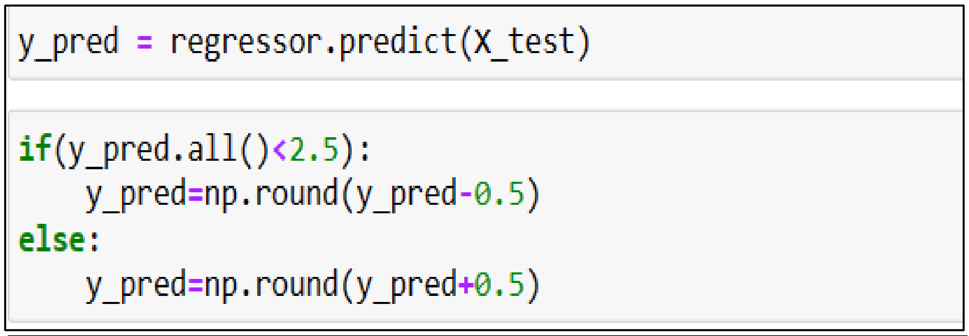
(vii)



```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
regressor.fit(X_train,y_train)
```

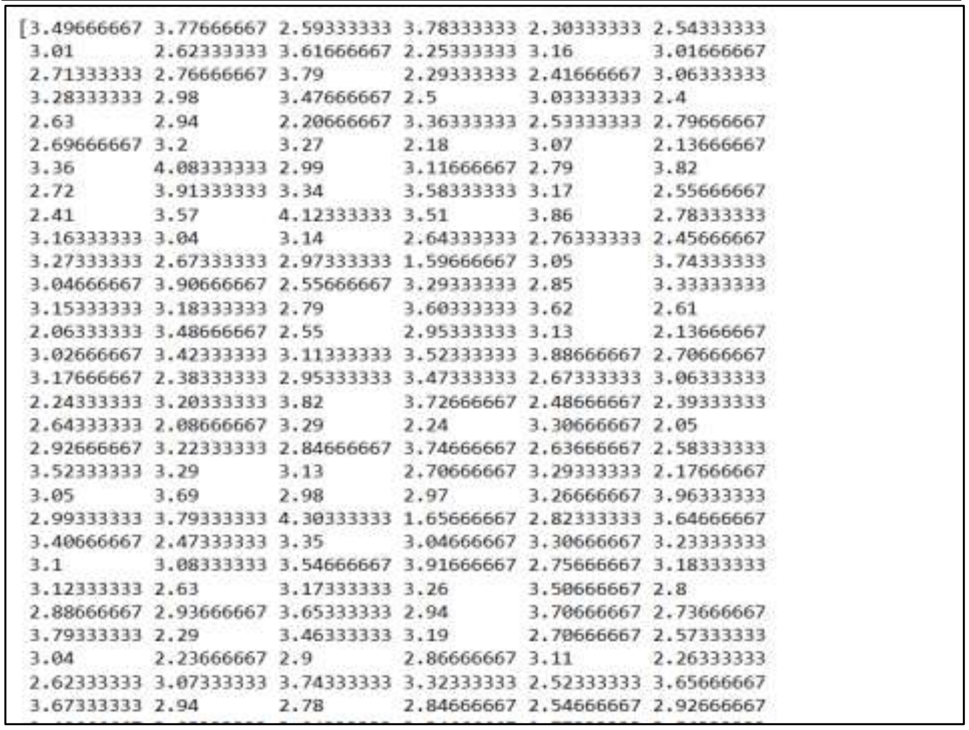
```
C:\Users\swati gamit\AppData\Local\Temp\ipykernel_11396\1592691624.py:3: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  regressor.fit(X_train,y_train)

RandomForestRegressor(n_estimators=300, random_state=0)
```

(viii)



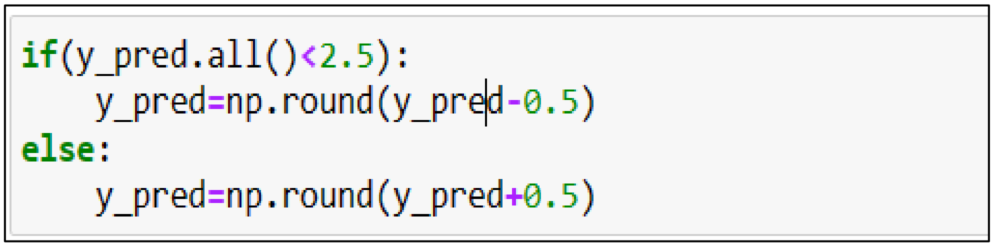
```
y_pred = regressor.predict(X_test)
```

```
if(y_pred.all()<2.5):
    y_pred=np.round(y_pred-0.5)
else:
    y_pred=np.round(y_pred+0.5)
```



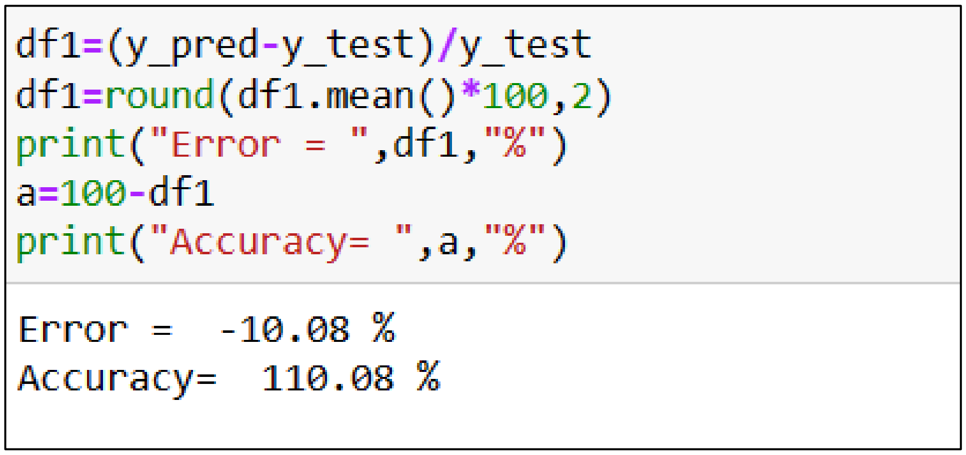
```
[3.49666667 3.77666667 2.59333333 3.78333333 2.30333333 2.54333333
 3.01       2.62333333 3.61666667 2.25333333 3.16       3.01666667
 2.71333333 2.76666667 3.79       2.29333333 2.41666667 3.06333333
 3.28333333 2.98       3.47666667 2.5        3.03333333 2.4
 2.63       2.94       2.20666667 3.36333333 2.53333333 2.79666667
 2.69666667 3.2        3.27       2.18       3.07       2.13666667
 3.36       4.08333333 2.99       3.11666667 2.79       3.82
 2.72       3.91333333 3.34       3.58333333 3.17       2.55666667
 2.41       3.57       4.12333333 3.51       3.86       2.78333333
 3.16333333 3.04       3.14       2.64333333 2.76333333 2.45666667
 3.27333333 2.67333333 2.97333333 1.59666667 3.05       3.74333333
 3.04666667 3.90666667 2.55666667 3.29333333 2.85       3.33333333
 3.15333333 3.18333333 2.79       3.60333333 3.62       2.61
 2.06333333 3.48666667 2.55       2.95333333 3.13       2.13666667
 3.02666667 3.42333333 3.11333333 3.52333333 3.88666667 2.70666667
 3.17666667 2.38333333 2.95333333 3.47333333 2.67333333 3.06333333
 2.24333333 3.20333333 3.82       3.72666667 2.48666667 2.39333333
 2.64333333 2.08666667 3.29       2.24       3.30666667 2.05
 2.92666667 3.22333333 2.84666667 3.74666667 2.63666667 2.58333333
 3.52333333 3.29       3.13       2.70666667 3.29333333 2.17666667
 3.05       3.69       2.98       2.97       3.26666667 3.96333333
 2.99333333 3.79333333 4.30333333 1.65666667 2.82333333 3.64666667
 3.40666667 2.47333333 3.35       3.04666667 3.30666667 3.23333333
 3.1        3.08333333 3.54666667 3.91666667 2.75666667 3.18333333
 3.12333333 2.63       3.17333333 3.26       3.50666667 2.8
 2.88666667 2.93666667 3.65333333 2.94       3.70666667 2.73666667
 3.79333333 2.29       3.46333333 3.19       2.70666667 2.57333333
 3.04       2.23666667 2.9        2.86666667 3.11       2.26333333
 2.62333333 3.07333333 3.74333333 3.32333333 2.52333333 3.65666667
 3.67333333 2.94       2.78       2.84666667 2.54666667 2.92666667
```

(ix)



```
if(y_pred.all()<2.5):
    y_pred=np.round(y_pred-0.5)
else:
    y_pred=np.round(y_pred+0.5)
```

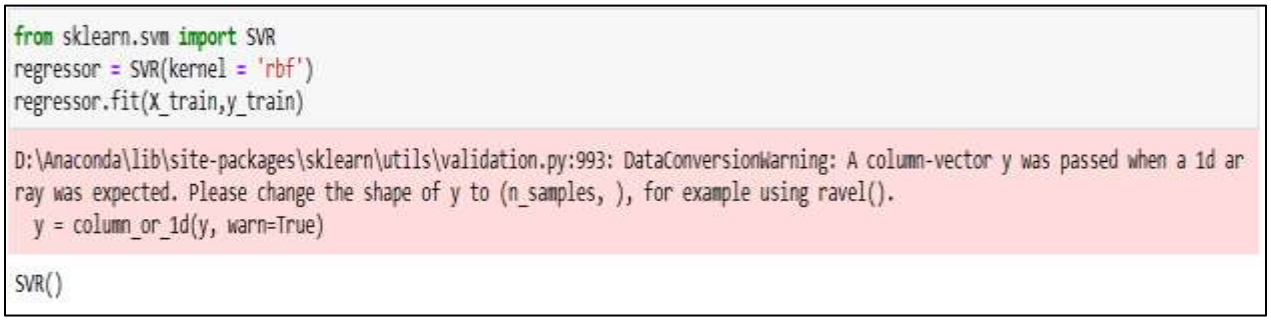
(x)



```
df1=(y_pred-y_test)/y_test
df1=round(df1.mean()*100,2)
print("Error = ",df1,"%")
a=100-df1
print("Accuracy= ",a,"%")
```

```
Error =  -10.08 %
Accuracy=  110.08 %
```

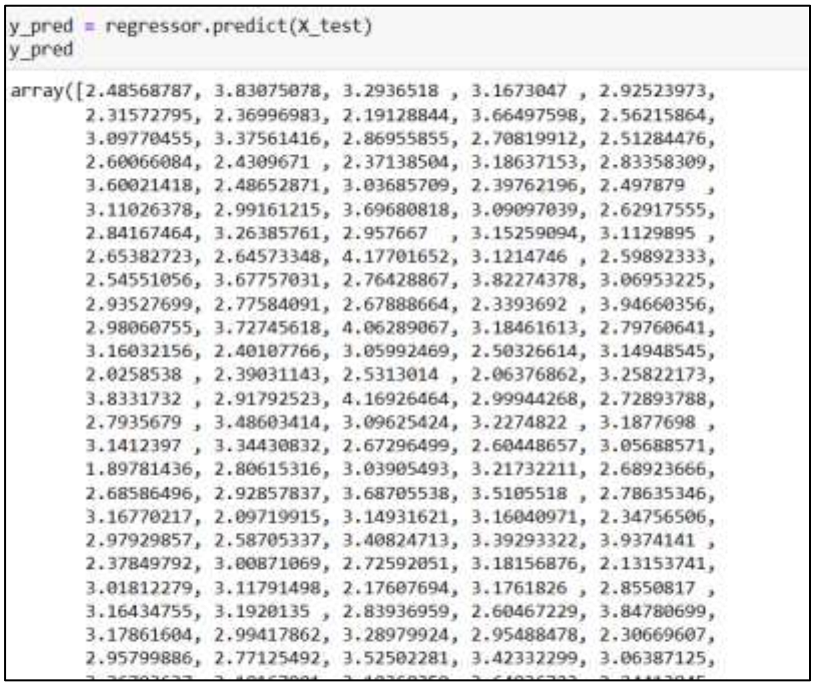
(xi)



```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train,y_train)
```

```
D:\Anaconda\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d ar
ray was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

SVR()
```

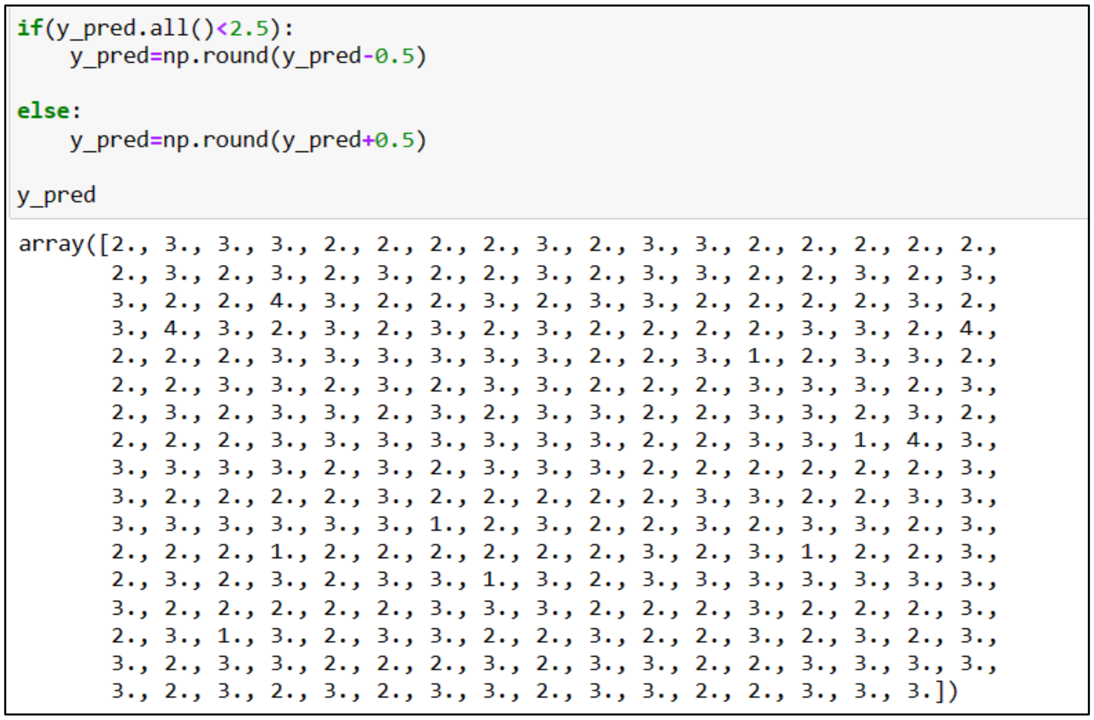
(xii)



```
y_pred = regressor.predict(X_test)
y_pred
```

```
array([2.48568787, 3.83075078, 3.2936518 , 3.1673047 , 2.92523973,
       2.31572795, 2.36996983, 2.19128844, 3.66497598, 2.56215864,
       3.09770455, 3.37561416, 2.86955855, 2.70819912, 2.51284476,
       2.60066084, 2.4309671 , 2.37138504, 3.18637153, 2.83358309,
       3.60021418, 2.48652871, 3.03685709, 2.39762196, 2.497879  ,
       3.11026378, 2.99161215, 3.69680818, 3.09097039, 2.62917555,
       2.84167464, 3.26385761, 2.957667  , 3.15259094, 3.1129895 ,
       2.65382723, 2.64573348, 4.17701652, 3.1214746 , 2.59892333,
       2.54551056, 3.67757031, 2.76428867, 3.82274378, 3.06953225,
       2.93527699, 2.77584091, 2.67888664, 2.3393692 , 3.94660356,
       2.98060755, 3.72745618, 4.06289067, 3.18461613, 2.79760641,
       3.16032156, 2.40107766, 3.05992469, 2.50326614, 3.14948545,
       2.0258538 , 2.39031143, 2.5313014 , 2.06376862, 3.25822173,
       3.8331732 , 2.91792523, 4.16926464, 2.99944268, 2.72893788,
       2.7935679 , 3.48603414, 3.09625424, 3.2274822 , 3.1877698 ,
       3.1412397 , 3.34430832, 2.67296499, 2.60448657, 3.05688571,
       1.89781436, 2.80615316, 3.03905493, 3.21732211, 2.68923666,
       2.68586496, 2.92857837, 3.68705538, 3.5105518 , 2.78635346,
       3.16770217, 2.09719915, 3.14931621, 3.16040971, 2.34756506,
       2.97929857, 2.58705337, 3.40824713, 3.39293322, 3.9374141 ,
       2.37849792, 3.00871069, 2.72592051, 3.18156876, 2.13153741,
       3.01812279, 3.11791498, 2.17607694, 3.1761826 , 2.8550817 ,
       3.16434755, 3.1920135 , 2.83936959, 2.60467229, 3.84780699,
       3.17861604, 2.99417862, 3.28979924, 2.95488478, 2.30669607,
       2.95799886, 2.77125492, 3.52502281, 3.42332299, 3.06387125,
```

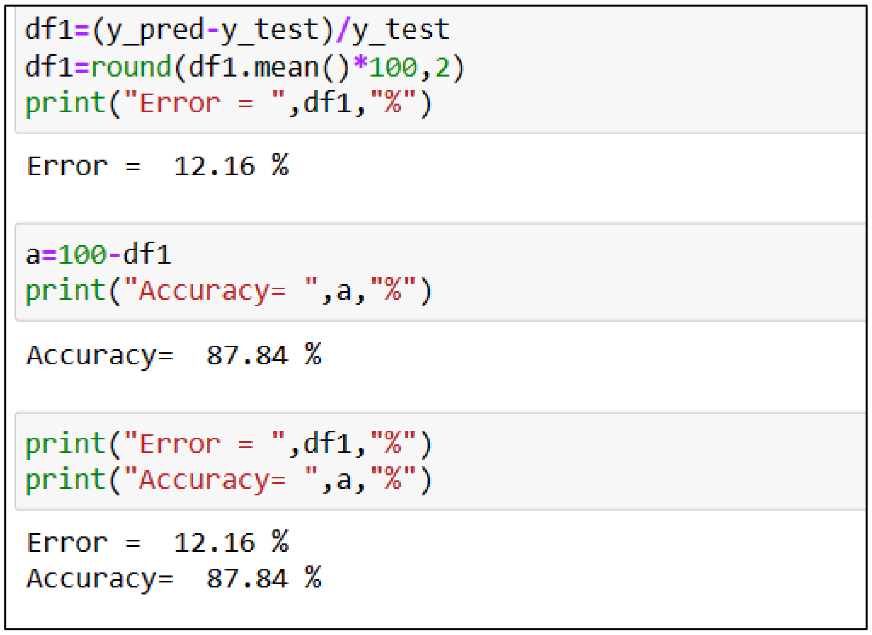
(xiii)



```
if(y_pred.all()<2.5):
    y_pred=np.round(y_pred-0.5)

else:
    y_pred=np.round(y_pred+0.5)

y_pred
```

```
array([2., 3., 3., 3., 2., 2., 2., 2., 3., 2., 3., 3., 2., 2., 2., 2., 2.,
       2., 3., 2., 3., 2., 3., 2., 2., 3., 2., 3., 3., 2., 2., 3., 2., 3.,
       3., 2., 2., 4., 3., 2., 2., 3., 2., 3., 3., 2., 2., 2., 2., 3., 2.,
       3., 4., 3., 2., 3., 2., 3., 2., 3., 2., 2., 2., 2., 3., 3., 2., 4.,
       2., 2., 2., 3., 3., 3., 3., 3., 3., 2., 2., 3., 1., 2., 3., 3., 2.,
       2., 2., 3., 3., 2., 3., 2., 3., 3., 2., 2., 2., 3., 3., 3., 2., 3.,
       2., 3., 2., 3., 3., 2., 3., 2., 3., 3., 2., 2., 3., 3., 2., 3., 2.,
       2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 2., 2., 3., 3., 1., 4., 3.,
       3., 3., 3., 3., 2., 3., 2., 3., 3., 3., 2., 2., 2., 2., 2., 2., 3.,
       3., 2., 2., 2., 2., 3., 2., 2., 2., 2., 2., 3., 3., 2., 2., 3., 3.,
       3., 3., 3., 3., 3., 3., 1., 2., 3., 2., 2., 3., 2., 3., 3., 2., 3.,
       2., 2., 2., 1., 2., 2., 2., 2., 2., 2., 3., 2., 3., 1., 2., 2., 3.,
       2., 3., 2., 3., 2., 3., 3., 1., 3., 2., 3., 3., 3., 3., 3., 3., 3.,
       3., 2., 2., 2., 2., 2., 3., 3., 3., 2., 2., 2., 3., 2., 2., 2., 3.,
       2., 3., 1., 3., 2., 3., 3., 2., 2., 3., 2., 2., 3., 2., 3., 2., 3.,
       3., 2., 3., 3., 2., 2., 2., 3., 2., 3., 3., 2., 2., 3., 3., 3., 3.,
       3., 2., 3., 2., 3., 2., 3., 3., 2., 3., 3., 2., 2., 3., 3., 3.])
```

(xiv)



```
df1=(y_pred-y_test)/y_test
df1=round(df1.mean()*100,2)
print("Error = ",df1,"%")
```

```
Error =  12.16 %
```

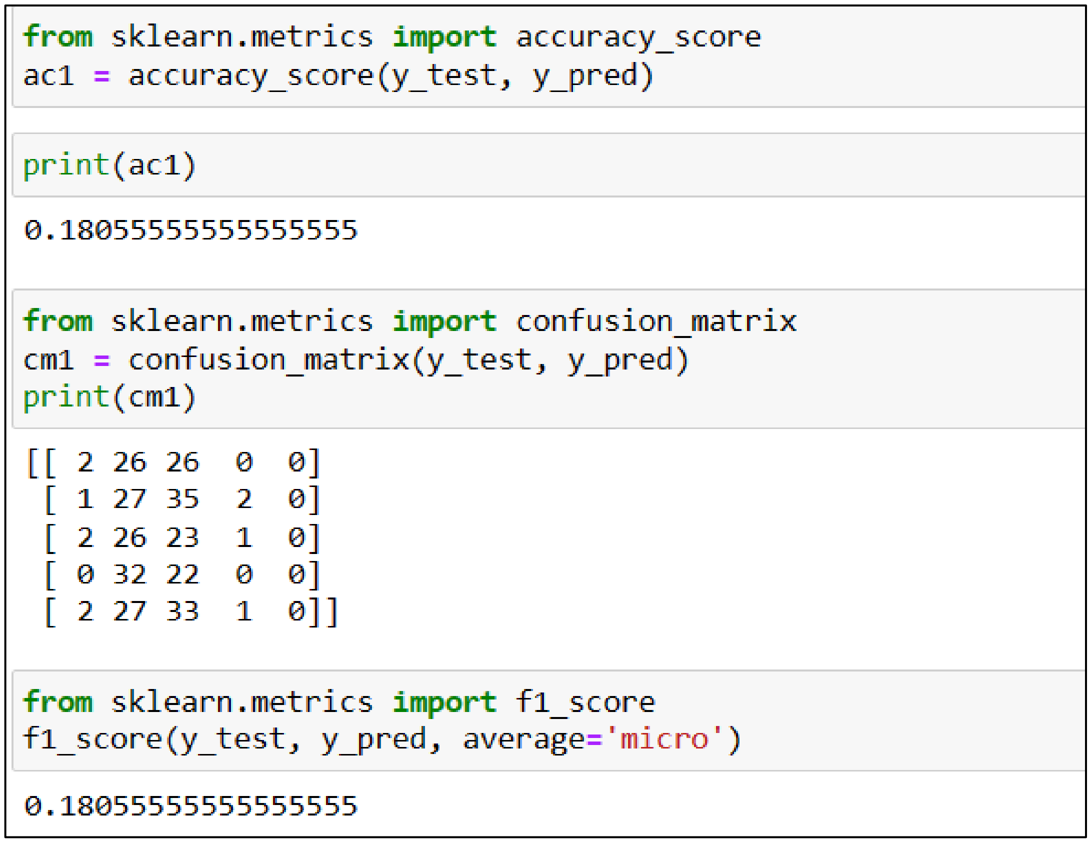
```
a=100-df1
print("Accuracy= ",a,"%")
```

```
Accuracy=  87.84 %
```

```
print("Error = ",df1,"%")
print("Accuracy= ",a,"%")
```

```
Error =  12.16 %
Accuracy=  87.84 %
```

(xv)



```
from sklearn.metrics import accuracy_score
ac1 = accuracy_score(y_test, y_pred)
```

```
print(ac1)
```

```
0.18055555555555555
```

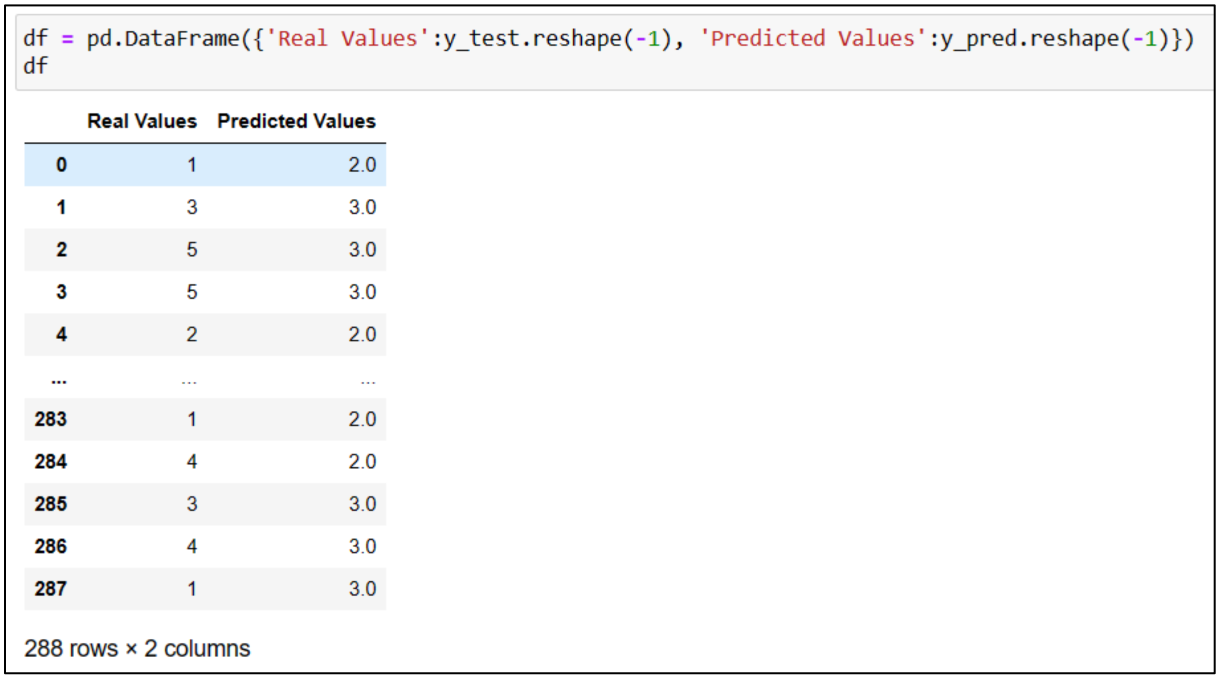
```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, y_pred)
print(cm1)
```

```
[[ 2 26 26  0  0]
 [ 1 27 35  2  0]
 [ 2 26 23  1  0]
 [ 0 32 22  0  0]
 [ 2 27 33  1  0]]
```

```
from sklearn.metrics import f1_score
f1_score(y_test, y_pred, average='micro')
```

```
0.18055555555555555
```

(xvi)



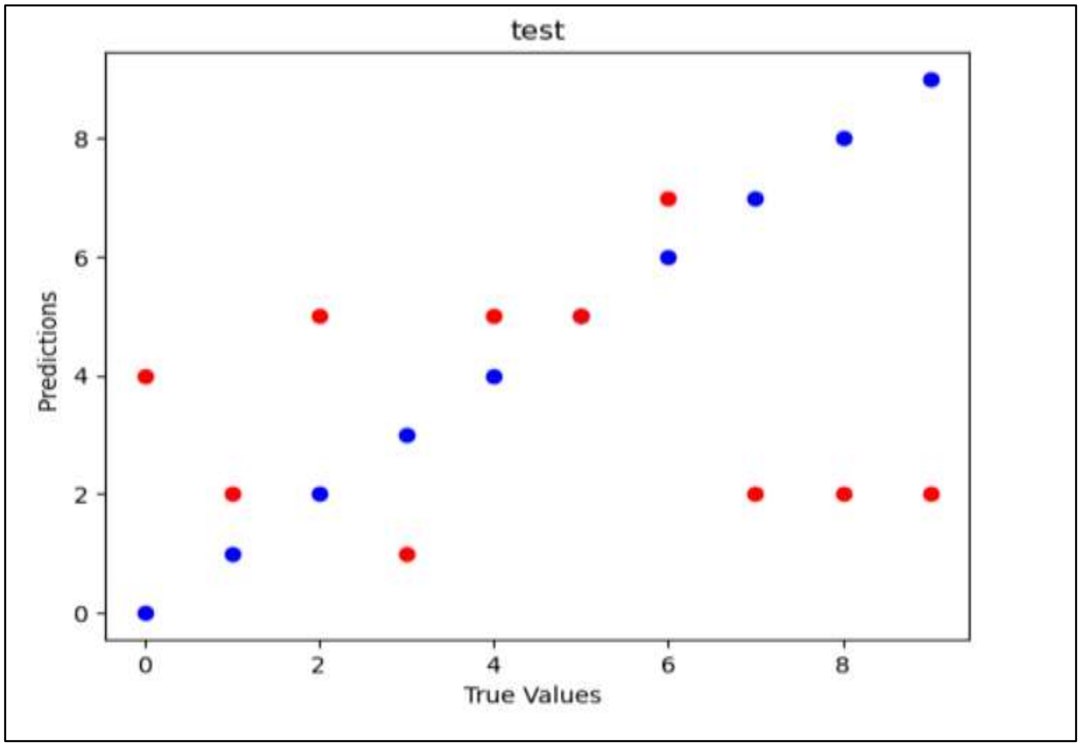
```
df = pd.DataFrame({'Real Values':y_test.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
df
```

	Real Values	Predicted Values
0	1	2.0
1	3	3.0
2	5	3.0
3	5	3.0
4	2	2.0
...	...	...
283	1	2.0
284	4	2.0
285	3	3.0
286	4	3.0
287	1	3.0

288 rows × 2 columns

(xvii)

```python
import matplotlib.pyplot as plt
import numpy as np


def plotGraph(y_test,y_pred,regressorName):
    if max(y_test) >= max(y_pred):
        my_range = int(max(y_test))
    else:
        my_range = int(max(y_pred))
    plt.scatter(range(len(y_test)), y_test, color='blue')
    plt.scatter(range(len(y_pred)), y_pred, color='red')
    plt.xlabel('True Values ')
    plt.ylabel('Predictions ')
    plt.title(regressorName)
    plt.show()
    return


y_test = range(10)
y_pred = np.random.randint(0, 10, 10)

plotGraph(y_test, y_pred, "test")
```



FUTURE ENHANCEMENT

We have the performed of machine learning method Random Forests used for predicting traffic flow. The results show that simple naïve methods, such as historical average, are surprisingly effective when making long-term predictions (more than one hour into the future), while using current traffic measurements as naïve method for prediction works well when making more short-term predictions (less than 1h). This is to be expected, since current traffic situation effect more on traffic in the nearby future, then on traffic in a few hours or days.

By using less complex models, optimal model parameters are found more readily, the models run a lot faster, and they are easier to understand and maintain. We also state that the main disadvantage of models presented in this research, is its inability to predict unusual traffic events. Even though common traffic status is informative for a commuter in a new environment, unusual traffic is the most informative information for local commuter who is aware of usual traffic. The main reason for this disadvantage is that current models uses only historical traffic data.

Since, some of unusually traffic events are caused by other related events (such as nearby traffic accidents, bad weather, holidays, etc.), we believe that by including additional data sources in the model, prediction of such events could be significantly improved.

Therefore, our future plan is to collect several quality traffic related data sources (such as traffic alerts, special days statuses, bigger social events, etc.) and fuse them with loop counters data in order to generate better traffic prediction models.

BIBLIOGRAPHY/ REFERENCES

[1] G. Meena, D. Sharma and M. Mahrishi, "Traffic Prediction for Intelligent Transportation System using Machine Learning," 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), 2020, pp. 145-148, doi:10.1109/ICETCE48199.2020.9091758

[2] X. Kong, Z. Xu, G. Shen, J. Wang, Q. Yang, and B. Zhang, "Urban traffic congestion estimation and prediction based on floating car trajectory data," Future Generation Computer Systems, vol. 61, pp. 97–107, 2016.

[3] Yuan, H., Li, G. A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. Data Sci. Eng. 6, 63–85 (2021). https://doi.org/10.1007/s41019-020-00151-z

[4] https://www.openstreetmap.org/export#map=15/21.2223/72.8374

[5] https://github.com/Nupurgopali/Traffic-Prediction-using-SVR-and-RFR/blob/master/svr.py

[6] Yue Dai et al., "Dynamic prediction of drivers' personalroutes through machine learning," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1-8, doi10.1109/SSCI.2016.7850094.

[7] https://www.slideshare.net/OmSuryawanshi9/traffic-prediction-for-intelligent-transportation-system-using-machine-learning?from_action=save

[8] Yiming He, Mashrur Chowdhury, Yongchang Ma, and PierluigiPisu. Merging mobility and energy vision with hybrid electric vehicles and vehicle infrastructure integration. Energy Policy, 41:599–609, 2012.

[9] Jason Brownlee. Bagging and random forest ensemble algorithms for machine learning. Machine Learning Algorithms, pages 4–22, 2016