TRAFFIC MANAGEMENT SYSTEM-IOT

PHASE 3 PROJECT SUBMISSION

Start building the IoT Traffic Monitoring System by loading and Pre-processing dataset.

TRAFFIC MANAGEMENT SYSTEM-IOT

PHASE 3: Development part-1

INTRODUCTION:

- Traffic management systems play a vital role in urban and suburban environments, helping to mitigate congestion, reduce accidents, and improve the overall flow of traffic. With the advent of the Internet of Things (IoT), these systems have evolved to become more efficient, data-driven, and responsive. In this introduction, we will explore the concept of a Traffic Management System using IoT and its key components.

- A Traffic Management System using IoT, also known as an IoT-based Traffic Management System, is a smart transportation system that utilizes the Internet of Things (IoT) technology to monitor, control, and optimize traffic flow on roadways. It incorporates a network of interconnected devices and sensors to collect real-time data on traffic conditions and uses this data to manage and improve traffic operations.

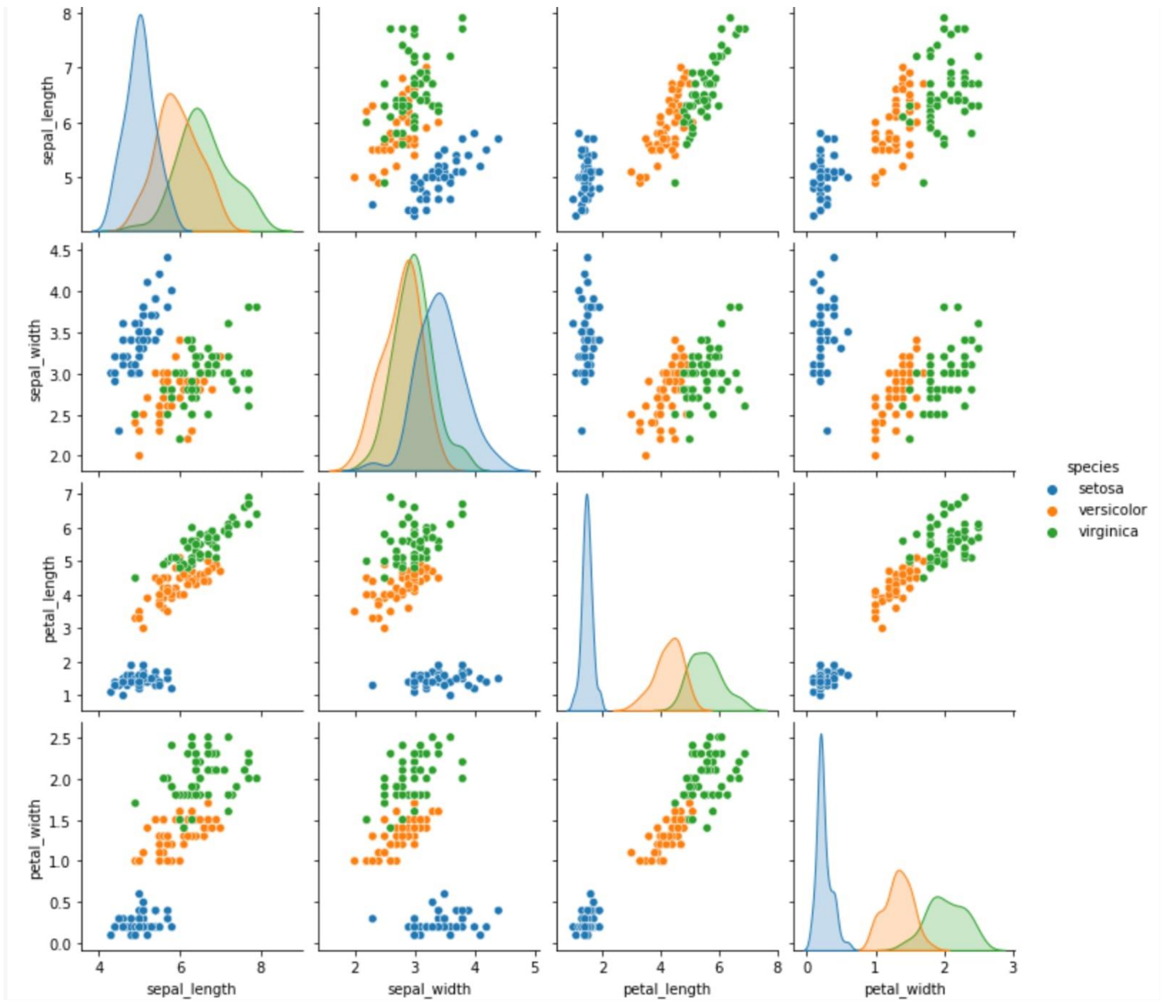
Introduction to IOT Traffic Management System With Python:

- An Internet of Things (IoT) traffic management system with Python is a modern solution that utilizes IoT devices, sensors, and the Python programming language to monitor, analyze, and manage traffic on roads and highways. It leverages real-time data from various sources to make intelligent decisions for optimizing traffic flow, reducing congestion, and improving overall road safety.

- Building an IoT traffic management system with Python requires expertise in hardware (sensors and IoT devices) and software development (data processing, algorithms). Python is a versatile programming language for building the software components of such systems due to its rich ecosystem of libraries and tools for data analysis, machine learning, and web development.

DATA PREPARATION:

- Data preparation for a traffic monitoring system is a critical step in ensuring that the system can provide accurate and reliable information. The data preparation process involves cleaning, transforming, and enriching the data to make it suitable for analysis.



PAIR PLOT OF DIFFERENT VARIABLES.

Loading and preprocessing datasets in traffic monitoring system using IoT:

1. Data Collection:

- Gather data from IoT sensors and devices deployed in the field.
- These sensors provide real-time data on various aspects of traffic, such as vehicle count, speed, and congest
- Common protocols for data transmission include MQTT, HTTP, or other IoT-specific protocols.

‘ import pandas as pd ’.

Load the dataset:

‘ dataset_ path = ‘ Your_ traffic_ data.csv ‘.

Load dataset into a pandas dataframe:

‘ df =pd. Read _csv (dataset_ path)’.

2. Data Preprocessing:

1. Data Cleaning:

- Data Validation: Validate the incoming data to identify and handle outliers, missing values, or inaccurate data points.
- Outlier Detector: Use statistical methods to detect and address outliers that may distort analysis.
- Missing Data Handling: Address missing data through imputation or removal of incomplete records.

Missing values:

‘missing_ values = df.isnull().sum()’

Default values:

‘ df = df.fillna(0)’

2. Noise Reduction:

- Apply smoothing and filtering techniques to reduce noise and fluctuations in the data. These techniques can help in improving data quality and reducing false alarms.

‘df[‘smoothed_Data’]=df[‘Vehicle_count’].rolling(window=window_size) .mean()’.

3. Data Compression:

- Compress data if storage or transmission resources are limited. This can be achieved through techniques like data differencing or data summarization.

‘original_size=len(speed_data_str)

Compressed_size=len(compressed_data)

Compression_ratio=original_size/compressed_size)’.

4. Data Quality Assessment:

- Data Quality Metrics: Define and monitor data quality metrics to assess accuracy, completeness, and timeliness of the data.
- Data Quality Monitoring: Continuously monitor data quality and set up alerts for anomalies or deviations from expected data patterns.

‘def access_data_quality(dataframe):

Print(“Data Quality Assessment:”)’.

5. Data Security and Privacy:

- Ensure that data is stored, transmitted, and accessed securely, adhering to security and privacy regulations and best practices.

4. Data Storage and Archiving:

1.Database Maintenance: Regularly maintain and optimize the database to ensure efficient storage and retrieval of data.
2.Data Archiving: Implement a strategy for archiving historical data that is no longer required for real-time analysis but may be needed for historical analysis or compliance purposes.
3.Backup and Disaster Recovery: Set up data backup mechanisms to protect against data loss in case of system failures or disasters.

Data is stored:

```
            'data_file = 'traffic_data.csv'
```

Archiving data:

```
            'archive_directory = 'data_archives'
```

# Create the archive directory if it doesn't exist.

```
            'if not os.path.exists(archive_directory):
                os.makedirs(archive_directory)'
```

# Function to save data to the CSV file

```
            'def save_data_to_csv(data):
                  with open(data_file, 'a', newline='') as file:
                        writer = csv.writer(file)
                        writer.writerow(data)'
```

# Function to archive data

```
           'def archive_data():
                  timestamp = time.strftime("%Y%m%d%H%M%S")
                  archive_filename =
f'{archive_directory}/traffic_data_{timestamp}.csv'
```

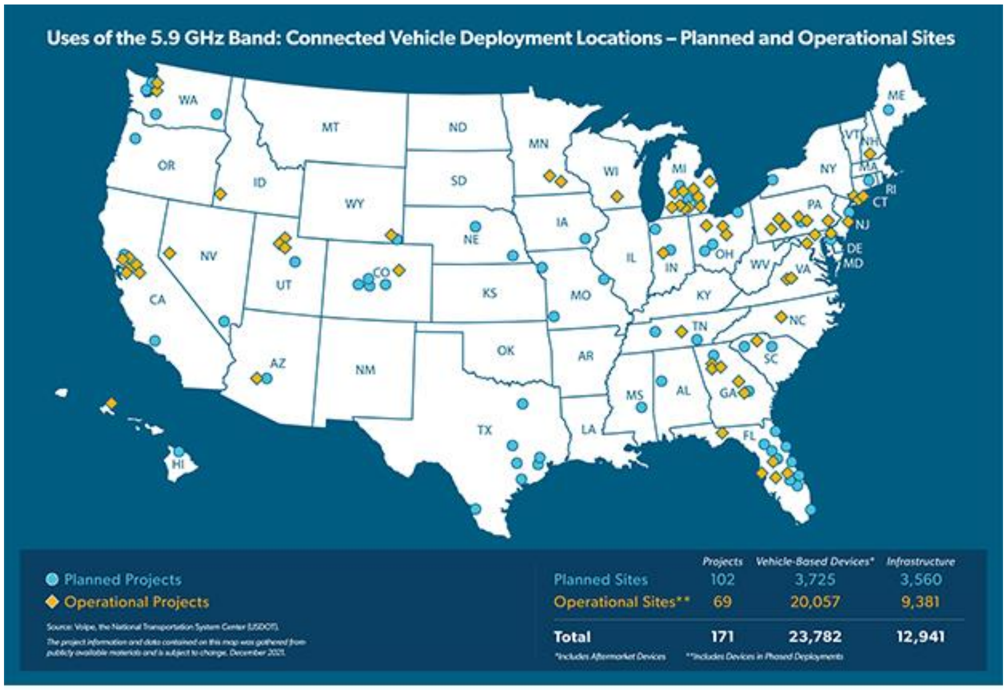
```
    try:
        copyfile(data_file, archive_filename)
        print(f"Archived data to {archive_filename}")
    except IOError as e:
        print(f"Error archiving data: {e}")

# Main data storage loop
try:
    while True:
         data_point = [time.strftime("%Y-%m-%d %H:%M:%S"), 'Sample data']  #
Replace with actual data
        save_data_to_csv(data_point)
        print(f"Stored data: {data_point}")
```

DATA ACQUISITION:

- Data acquisition in a traffic management system involves the process of collecting, gathering, and capturing data from various sources, such as sensors, cameras, detectors, and other IoT devices, to monitor and manage traffic on roads and highways. This data is a fundamental component of traffic management systems, as it provides valuable information for analysis, decision-making, and optimizing traffic flow.

- Data science projects typically start with the acquisition of data. Such data sets may consist of secondary data made available on the web by commercial or non-commercial organisations. This part of the tutorial explains how you can obtain such online data sets using code.
- Many data sets can be downloaded manually through your browser, for example, from data portals or repositories. Re3data is a large overview of repositories for research data.
- There can be good reasons for downloading data sets using a script. The manual acquisition of data may be tedious if the data collection consists of many files. In some cases, you may want to download files that are updated frequently.
- In this tutorial, we distinguish three methods of data acquisition: downloading data files, accessing data through APIs and webscraping. You

usually choose one of these methods to acquire your data, based on what the data provider offers.



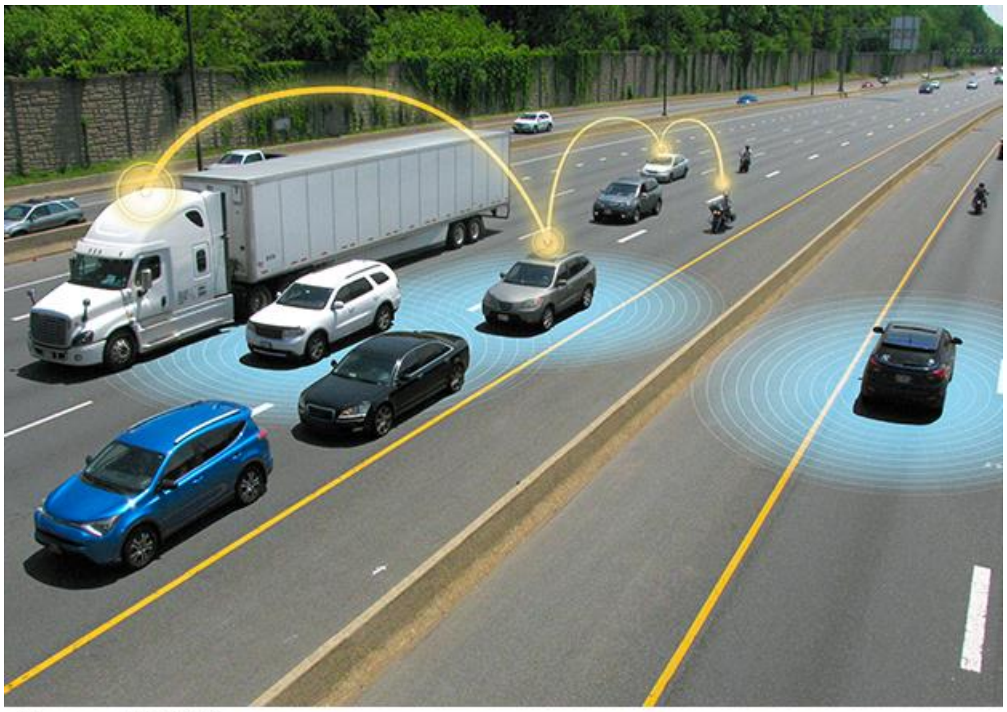
Source: Volpe, the National Transportation System Center (USDOT).
The project information and data contained on this map was gathered from publicly available materials and is subject to change. December 2021.

	Projects	*Vehicle-Based Devices**	*Infrastructure*
Planned Sites	102	3,725	3,560
Operational Sites**	69	20,057	9,381
Total	**171**	**23,782**	**12,941**

**Includes Aftermarket Devices* ***Includes Devices in Phased Deployments*

Source: USDOT.

DATA ACQUISITION


Source: USDOT.

PROGRAM FOR DATA ACQUISITION:

```
    import requests


contents = ""

response = requests.get('https://www.universiteitleiden.nl')

print( response.status_code )


if response.status_code == 200:

   response.encoding = 'utf-8'

   contents = response.text

   print (contents)

url = "https://www.gutenberg.org/files/98/98-0.txt"
```

```
response = requests.get(url)


if response:
    response.encoding = 'utf-8'
    print (response.text)
```

IMPORTANCE OF LOADING AND PREPROCESSING:

Loading and processing datasets are crucial for a traffic management system for several reasons:

1. **Data-Driven Decision Making:** Traffic management systems rely on data to make informed decisions. Loading and processing datasets provide the necessary information to make real-time decisions about traffic flow, congestion, and routing.
2. **Traffic Prediction:** Historical data can be used to predict traffic patterns, which helps in optimizing routes, traffic signals, and other traffic management strategies. This is essential for reducing congestion and improving traffic flow.
3. **Anomaly Detection:** Datasets enable the system to detect anomalies, such as accidents or road closures, in real-time. This information can be used to reroute traffic and dispatch emergency services promptly.
4. **Optimizing Signal Timing:** Traffic signal timings can be adjusted based on current traffic conditions, which reduces waiting times for vehicles and minimizes traffic congestion. This optimization relies on real-time data.
5. **Traffic Enforcement:** Datasets can be used to identify traffic violations through technologies like license plate recognition, which helps in enforcing traffic rules and ensuring road safety.
6. **Infrastructure Planning:** Long-term dataset analysis is vital for urban planning. It helps identify areas with chronic traffic issues, leading to informed decisions on infrastructure development and improvements.

7. **Public Information:** Traffic data can be shared with the public through apps or websites, helping commuters make informed choices about their routes and travel times.

8. **Emergency Response:** In the case of accidents or emergencies, the system can use traffic data to reroute emergency vehicles efficiently, potentially saving lives.

PROGRAM:

```
import paho.mqtt.client as mqtt
import json
import time

MQTT_BROKER = "mqtt.example.com"
MQTT_PORT = 1883
MQTT_TOPIC = "traffic_data"
MQTT_USERNAME = "your_username"
MQTT_PASSWORD = "your_password"
traffic_data = {
    "location": "City Center",
    "speed": 40,
    "congestion_level": "moderate",
    "timestamp": int(time.time())
}


def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe(MQTT_TOPIC)


def on_publish(client, userdata, mid):
    print(f"Published message with MID {mid}")


def main():
    client = mqtt.Client()
    client.username_pw_set(MQTT_USERNAME, MQTT_PASSWORD)
    client.on_connect = on_connect
    client.on_publish = on_publish
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    while True:
        traffic_data["timestamp"] = int(time.time())
        payload = json.dumps(traffic_data)
        client.publish(MQTT_TOPIC, payload)
        print(f"Published: {payload}")
        time.sleep(60)  # Send data every 60 seconds
if __name__ == "__main__":
    main()
```

In this script,

- Replace the placeholders for MQTT broker details (‘`MQTT_BROKER’`, ‘`MQTT_PORT’`, ‘`MQTT_TOPIC’`, ‘`MQTT_USERNAME’`, and ‘`MQTT_PASSWORD’`) with your actual MQTT broker information.
- Make sure you have the ‘`paho-mqtt’` library installed. You can install it using ‘`pip install paho-mqtt’`.
- Customize the ‘`traffic_data’` dictionary to match the format and content of your real traffic data.
- Run the script on your IoT device. It will continuously publish the simulated or real traffic data to the MQTT broker at the specified intervals.

OUTPUT:

The script you provided is designed to publish simulated traffic data to an MQTT broker. When you run this script, it will connect to the MQTT broker, publish the traffic data to the specified topic, and print a message indicating that the data has been published. Here's an example of what the output might look like when the script is running:

```
Connected with result code 0

Published: {"location": "City Center", "speed": 40, "congestion_level": "moderate", "timestamp": 1634558459}

Published: {"location": "City Center", "speed": 40, "congestion_level": "moderate", "timestamp": 1634558519}

Published: {"location": "City Center", "speed": 40, "congestion_level": "moderate", "timestamp": 1634558579}...
```

Here's an explanation of the output:

- "Connected with result code 0" indicates that the script successfully connected to the MQTT broker. Result code 0 typically means a successful connection.
- "Published: {...}" is the printed message showing the JSON data that was published to the MQTT topic. The content of the JSON object is the simulated traffic data, and the "timestamp" field is updated at regular intervals.
- The script repeats the process of updating the timestamp and publishing the data every 60 seconds, as specified in the `time.sleep(60)` line.

Visualization of analytics results:

- Data visualizations such as graphs and diagrams often make it easier for researchers to explore patterns and trends within large data sets, or to identify notable exceptions to the trends that can be observed.
- During the last number of years, a large number of visualization libraries have been developed for the Python language. Examples include Bokeh, Plotly, Altair and Folium.

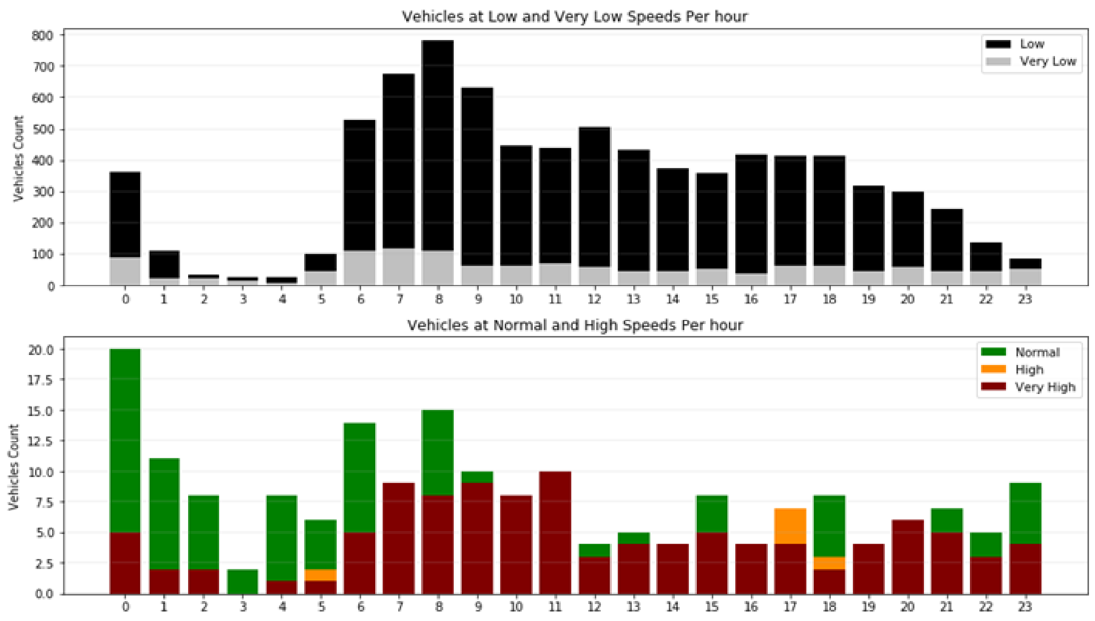
The data to be visualized:

- Data visualizations are obviously based on data. This tutorial makes use of a dataset named 'bli.csv'. This file contains data collected for the 2018 Better Life Index, which was created by the OECD to visuale some of the key factors that contribute to well-being in OECD countries, including education, income, housing and environment.

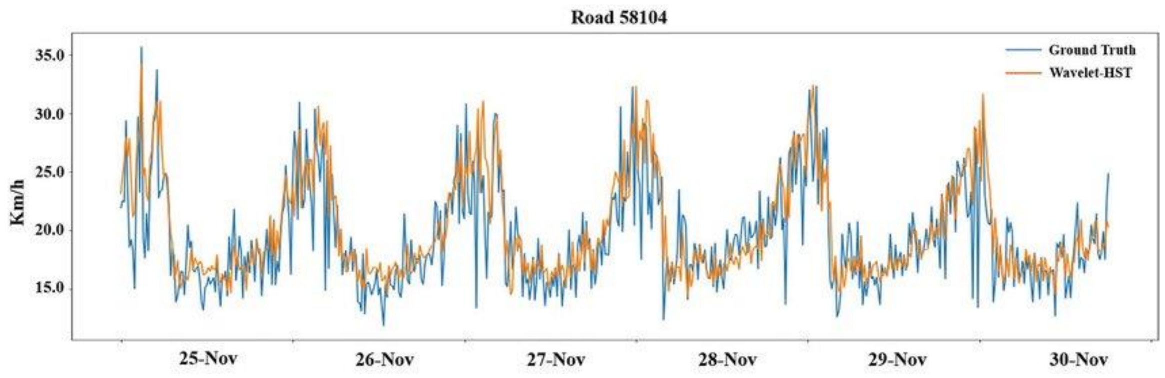
```
'import pandas as pd
import os

path_to_csv = os.path.join('Data','bli.csv')
df = pd.read_csv(path_to_csv)'
```

A bar plot

- Bar plots can be created using the `barplot()` method. It demands a `data` parameter, which need to refer to a `pandas` dataframe containing the values you want to visualize.



DATA ANALYSIS FOR TRAFFIC MANAGEMENT



TRAFFIC SPEED SERIES FORCASTING VISUALIZATION

Conclusion:

- The script efficiently processes traffic management system using IOT, is a smart transportation system that utilizes the Internet of Things (IoT) technology to monitor, control, and optimize traffic flow on roadways. decisions for optimizing traffic flow, reducing congestion, and improving overall road safety.

<u>THANK YOU!</u>