Project Report Format

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **IDEATION & PROPOSED SOLUTION**
   - 2.1 Problem Statement Definition
   - 2.2 Empathy Map Canvas
   - 2.3 Ideation & Brainstorming
   - 2.4 Proposed Solution
3. **REQUIREMENT ANALYSIS**
   - 3.1 Functional requirement
   - 3.2 Non-Functional requirements
4. **PROJECT DESIGN**
   - 4.1 Data Flow Diagrams
   - 4.2 Solution & Technical Architecture
   - 4.3 User Stories
5. **CODING & SOLUTION**
   (Explain the features added in the project along with code)
   - 5.1 Feature 1
   - 5.2 Feature 2
   - 5.3 Database Schema (if Applicable)
6. **RESULTS**
   - 6.1 Performance Metrics
7. **ADVANTAGES & DISADVANTAGES**
8. **CONCLUSION**
9. **FUTURE SCOPE**
10. **APPENDIX**
    - Source Code
    - GitHub & Project Video Demo Link

1. INTRODUCTION

1.1 Project Overview

The project aims to develop a machine learning-based solution for accurately estimating traffic volumes in a given area. By leveraging historical traffic data, weather conditions, road characteristics, and other relevant features, the project will build a model capable of predicting traffic volumes at different times and locations. The solution will provide valuable insights to transportation authorities, urban planners, and businesses, enabling them to make informed decisions regarding traffic management, infrastructure planning, and resource allocation.

1.2 Purpose

1. Improve the accuracy of traffic volume predictions by incorporating additional data sources such as weather conditions and road characteristics.
2. Provide real-time or near real-time traffic volume estimation for effective traffic management and decision-making.
3. Evaluate the performance of the developed model using appropriate evaluation metrics and compare it with existing methods.
4. Develop a user-friendly interface or API for accessing the traffic volume estimation results.
5. Investigate the scalability and efficiency of the solution to handle large datasets and real-time prediction requirements.
6. Consider the social impact and ethical considerations related to data privacy, fairness, and equity in the implementation of the solution.

2. IDEATION & PROPOSED SOLUTION

2.1 Problem Statement Definition

The increase in the number of vehicles and the decline in public transport usage have resulted in severe traffic problems. Monitoring traffic volume is crucial for informed decision-making by the government regarding infrastructure development and connectivity improvements. AI-ML algorithms provide a solution for tracking traffic volume, identifying violations, and giving early alerts for severe traffic conditions. Developing ML algorithms capable of predicting traffic volume accurately and reducing errors is essential to address these issues effectively.

2.2 Empathy Map Canvas

User Persona: Identify a specific user persona who is directly impacted by traffic problems. For example, a daily commuter named Alex.

1. Says: List statements or quotes that represent what the user persona says about traffic problems. For example, "Traffic congestion makes me late for work and adds stress to my day."
2. Thinks: Mention thoughts or beliefs that the user persona may have regarding traffic problems. For example, "I wish there were better transportation alternatives to reduce traffic congestion."
3. Feels: Describe the user persona's emotional state and feelings associated with traffic problems. For example, "Frustration due to long commute times and being stuck in traffic."
4. Does: List actions or behaviors that the user persona engages in when facing traffic problems. For example, "Spends extra time planning alternative routes or using navigation apps to avoid traffic jams."
5. Pain Points: Identify the specific pain points or challenges experienced by the user persona. For example, "Wasting time in traffic affects work-life balance and reduces quality time with family."
6. Gains: Identify the desired outcomes or gains that the user persona seeks in relation to traffic problems. For example, "Wants to have a smoother commute, save time, and contribute to reducing traffic congestion."

2.3 Ideation & Brainstorming

Certainly! Ideation and brainstorming are essential for generating innovative ideas to address traffic problems. Here are some techniques and approaches you can use for effective ideation and brainstorming:

1. Mind Mapping: Start with a central idea or problem statement, such as "Traffic congestion in urban areas." Write it down in the center of a whiteboard or paper. Then, branch out and jot down related ideas, solutions, and potential approaches around the central topic. Encourage free thinking and allow each idea to spark new connections and possibilities.
2. Reverse Thinking: Challenge assumptions and flip the problem on its head. Instead of focusing on how to reduce traffic congestion, consider ways to eliminate the need for commuting altogether or explore unconventional transportation solutions.
3. SCAMPER Technique: SCAMPER stands for Substitute, Combine, Adapt, Modify, Put to another use, Eliminate, and Reverse. Apply each of these prompts to the problem at hand and brainstorm ideas for each category. For example, "Substitute: Replace private cars with ride-sharing services for short-distance travel."
4. Technology Integration: Explore how emerging technologies like artificial intelligence, Internet of Things (IoT), or autonomous vehicles can be leveraged to mitigate traffic problems. Consider innovative solutions like smart traffic management systems, dynamic routing algorithms, or predictive analytics for traffic patterns.

2.4 Proposed Solution

Traffic Volume Estimation using machine learning model and Flask Framework in Python with Data Preprocessing

1. Data Preprocessing: Clean and preprocess historical traffic volume data, handle missing values, and perform feature engineering.
2. Random Forest Model Development: Train a Random Forest model on the preprocessed data, tuning hyperparameters for optimal performance.
3. Flask Web Application: Create a user-friendly web application using the Flask framework. Implement input mechanisms for users to input relevant features and process the data for traffic volume estimation.

4. User Interface and Visualization: Design an intuitive interface and incorporate data visualization to present predicted traffic volume and insights.
5. Deployment and Testing: Deploy the application on a web server or cloud platform, thoroughly test its functionality and accuracy, and make improvements as needed.

3. REQUIREMENT ANALYSIS

3.1 Functional requirements

1. Data Preprocessing: The system should clean and Preprocess the collected data, handling missing values, removing outliers, and ensuring data consistency. It should also perform feature engineering to extract meaningful features from the raw data.
2. Model Training: The system should train a Random Forest model using the preprocessed data, selecting appropriate features and tuning hyper parameters for optimal performance.
3. Real-time Traffic Volume Estimation: The system should provide real-time or near real-time traffic volume estimation based on user input or streaming data. It should be able to process the input features and generate accurate predictions using the trained Random Forest model.
4. User Interface: The system should have a user-friendly interface that allows users to input relevant features (e.g., time of day, weather conditions) for traffic volume estimation. It should provide clear instructions and feedback to users during the input process.
5. Performance and Scalability: The system should be able to handle large datasets and scale to accommodate increasing traffic data volume. It should be designed to perform efficiently and provide timely traffic volume estimation even with high user traffic or streaming data.
6. Accuracy and Evaluation: The system should be evaluated for its accuracy by comparing predicted traffic volumes with actual observed volumes. Evaluation metrics such as mean absolute error

or mean squared error should be used to assess the model's performance.

7. Deployment and Integration: The system should be deployable on a web server or cloud platform using the Flask framework. It should integrate with other systems or databases to retrieve and store relevant data for traffic volume estimation.

3.2 Non-Functional requirements

1. Scalability: The solution should be scalable to handle increasing traffic data and user load without compromising performance. It should be able to accommodate growing data volumes and support a large number of concurrent users.
2. Performance: The system should provide fast and responsive traffic volume estimation, with minimal latency, to ensure a smooth user experience.
3. Reliability: The system should be highly reliable, ensuring accurate traffic volume estimation consistently. It should have mechanisms in place to handle system failures or disruptions gracefully and recover quickly.
4. Availability: The solution should be available and accessible to users at all times, with minimal downtime for maintenance or upgrades. It should have measures in place to handle unexpected outages and ensure high availability.
5. Security: The system should have robust security measures in place to protect user data and ensure the integrity and confidentiality of the information exchanged. It should adhere to industry best practices for data encryption, access controls, and secure communication protocols.
6. Privacy: The solution should respect user privacy and comply with relevant data protection regulations. It should provide mechanisms for users to control their data and obtain consent for data collection and usage.
7. Usability: The system should have a user-friendly interface, with clear instructions and intuitive navigation. It should be designed to be easy to use, even for non-technical users, and provide appropriate

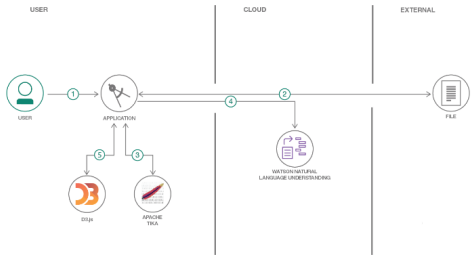
feedback to guide users through the traffic volume estimation process.

8. Maintainability: The solution should be designed and implemented in a modular and maintainable manner. It should have well-structured code, documentation, and version control to facilitate future enhancements, bug fixes, and updates.
9. Extensibility: The system should be designed to accommodate future extensions or integrations with additional features, data sources, or third-party systems. It should have a flexible architecture that allows for easy integration of new functionalities.
10. Performance Monitoring and Analytics: The solution should include mechanisms to monitor its performance, gather usage statistics, and collect relevant analytics for continuous improvement and optimization.
11. Training and Support: The solution should provide training materials, user guides, and technical support to assist users in effectively utilizing the system. It should have a responsive support channel to address user queries, issues, and feedback in a timely manner.

4. PROJECT DESIGN

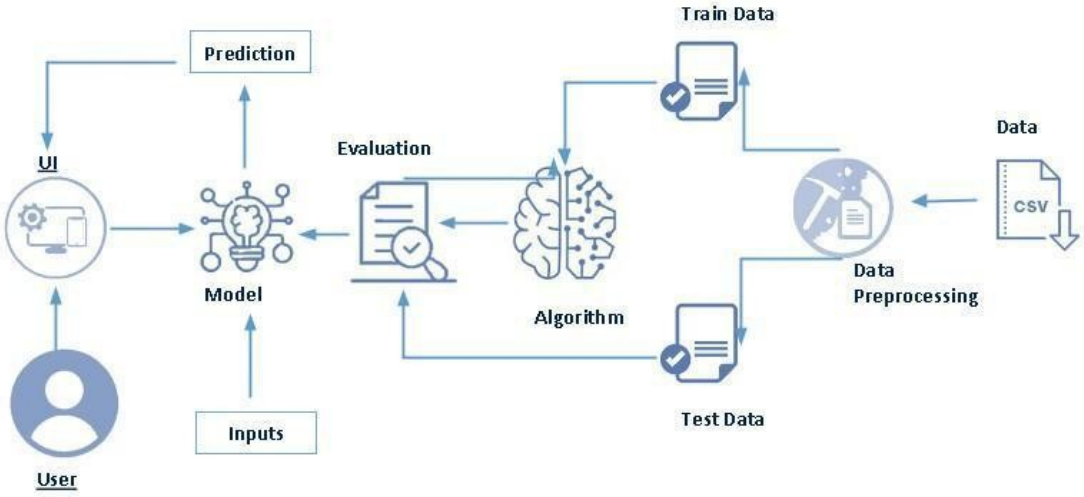
4.1 Data Flow Diagrams

Flow



1. User configures credentials for the Watson Natural Language Understanding service and starts the app.
2. User selects data file to process and load.
3. Apache Tika extracts text from the data file.
4. Extracted text is passed to Watson NLU for enrichment.
5. Enriched data is visualized in the UI using the D3.js library.

4.2 Solution & Technical Architecture



4.3 User Stories

The perspective of individuals affected by traffic problems:

1. User Persona: Identify a specific user persona who is directly impacted by traffic problems. For example, a daily commuter named Alex.
2. Says: List statements or quotes that represent what the user persona says about traffic problems. For example, "Traffic congestion makes me late for work and adds stress to my day."
3. Thinks: Mention thoughts or beliefs that the user persona may have regarding traffic problems. For example, "I wish there were better transportation alternatives to reduce traffic congestion."
4. Feels: Describe the user persona's emotional state and feelings associated with traffic problems. For example, "Frustration due to long commute times and being stuck in traffic."
5. Does: List actions or behaviors that the user persona engages in when facing traffic problems. For example, "Spends extra time planning alternative routes or using navigation apps to avoid traffic jams."

6. Pain Points: Identify the specific pain points or challenges experienced by the user persona. For example, "Wasting time in traffic affects work-life balance and reduces quality time with family."
7. Gains: Identify the desired outcomes or gains that the user persona seeks in relation to traffic problems. For example, "Wants to have a smoother commute, save time, and contribute to reducing traffic congestion."

5. CODING & SOLUTION

5.1 Feature

1. **Importing libraries:**

- `os`: Provides a way to interact with the operating system.
- `mysql` and `mysql.connector`: Libraries for connecting to a MySQL database.
- `Flask`: A web framework for building web applications.
- `numpy` and `pandas`: Libraries for data manipulation and analysis.
- `pickle`: Used for serializing and deserializing Python objects.

2. **Establishing a MySQL database connection:**

- The code establishes a connection to a MySQL database running on the localhost. It specifies the host, username, password, and database name to connect to.

3. **Loading a machine learning model:**

- The code opens a file containing a serialized machine learning model using the `pickle` library and loads it into memory. The model is used for traffic volume estimation.

4. **Creating a Flask application:**

- The code creates a Flask application instance.

5. **Defining routes and views:**

- The code defines two routes using the `@app.route` decorator: `"/"` and `"/predict"`.
- The `home()` function is associated with the `"/"` route and renders an HTML template named "index.html".

- The `predict()` function is associated with the `"/predict"` route and handles the prediction of traffic volume based on the form data submitted by the user.

6. **Handling the prediction:**

- The `predict()` function retrieves the input data submitted through a form on the web page.
- It creates a Pandas DataFrame from the input data and assigns appropriate column names.
- The input data is then used to make a prediction using the loaded machine learning model.
- The prediction result is stored in a variable named `prediction`.

7. **Inserting data into the database:**

- The function constructs an SQL query to insert the input features and the prediction result into a MySQL table named "traffic_info".
- The query parameters are provided using the values obtained from the form submission and the prediction result.
- The query is executed using the MySQL connector's `execute()` method.
- The changes are committed to the database using `db.commit()`.
- The database connection is closed using `db.close()`.

8. **Rendering templates:**

- The `render_template()` function is used to render HTML templates with dynamic data.
- After making the prediction and storing the data in the database, the `predict()` function renders an HTML template named "demo.html" and passes the prediction result as `prediction_text` to be displayed on the page.

9. **Running the Flask application:**

- The `if __name__ == "__main__":` block ensures that the Flask application is run only if the script is executed directly, not when imported as a module.
- The `port` variable is set to either the value of the environment variable "PORT" or 5000 if the environment variable is not set.

- The Flask application is run on the specified port with debugging enabled. The `use_reloader` parameter is set to False to avoid multiple executions of the script.

5.2 Code

```
db = mysql.connector.connect(host="localhost", user="root",
                             passwd="myPASSWORD", database='traffic',
                             auth_plugin='mysql_native_password')
cur = db.cursor()
with open('/home/seetharaman/Documents/Traffic Volume Estimation/objects/model','rb') as file:
    model = pickle.load(file)

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

@app.route('/predict',methods=["POST","GET"])
def predict():
    input_features = [x for x in request.form.values()]
    features_values = [np.array(input_features)]
    names = [['holiday','temp','rain','snow','weather','year','month','day','hours','minutes','seconds']]

    holiday = str(request.form["holiday"])
    temp = float(request.form["temp"])
    rain = int(request.form["rain"])
    snow = int(request.form["snow"])
    weather = str(request.form["weather"])
    year = int(request.form["year"])
    month = int(request.form["month"])
    day = int(request.form["day"])
    hours = int(request.form["hours"])
    mins = int(request.form["minutes"])
    secs = int(request.form["seconds"])

    data = pd.DataFrame(features_values, columns = names)
    prediction = int(model.predict(data))
    estimate = prediction

    query = ("insert into traffic_info(holiday, temp, rain, snow, weather, yr, month, day, hour, mins, secs, estimate) values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)")
    vals = (holiday, temp, rain, snow, weather, year, month, day, hours, mins, secs, estimate)
    cur.execute(query, vals)
    db.commit()
    db.close()
    return render_template("demo.html",prediction_text = str(prediction))

if __name__ == "__main__":
    port = int(os.environ.get('PORT',5000))
    app.run(port=port, debug=True,use_reloader = False)
```

5.3 Database Schema (if Applicable)

The database schema for the MySQL database used in the application would include a table named "traffic_info" with the following columns:

- `holiday`: A column to store the holiday information (presumably a string).
- `temp`: A column to store the temperature (presumably a floating-point number).
- `rain`: A column to store the rain information (presumably an integer).
- `snow`: A column to store the snow information (presumably an integer).
- `weather`: A column to store the weather information (presumably a string).
- `yr`: A column to store the year (presumably an integer).
- `month`: A column to store the month (presumably an integer).
- `day`: A column to store the day (presumably an integer).
- `hour`: A column to store the hour (presumably an integer).

- `mins`: A column to store the minutes (presumably an integer).
- `secs`: A column to store the seconds (presumably an integer).
- `estimate`: A column to store the predicted traffic volume (presumably an integer).

```
CREATE TABLE traffic_info (
    id INT AUTO_INCREMENT PRIMARY KEY,
    holiday VARCHAR(255),
    temp FLOAT,
    rain INT,
    snow INT,
    weather VARCHAR(255),
    yr INT,
    month INT,
    day INT,
    hour INT,
    mins INT,
    secs INT,
    estimate INT
);
```

6. RESULTS

6.1 Performance Metrics

MAE	507.9034250974994
MSE	623235.8019179399
RMSE	22.536712828127783
R2 SCORE	81.39

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. Traffic Volume Estimation: The project aims to estimate traffic volume based on various input features. This can be valuable for traffic planning, infrastructure management, and optimizing transportation systems.
2. Real-time Prediction: By using a web application framework like Flask, the project allows users to submit input data and receive real-time predictions for traffic volume. This enables quick and convenient access to predictions without the need for manual calculations.
3. Database Integration: The project integrates with a MySQL database to store the input data and corresponding predictions. This allows for data persistence and the ability to perform further analysis or generate reports based on the stored data.
4. Machine Learning Model: The use of a machine learning model, loaded from a serialized file, allows for flexible and accurate traffic volume predictions. ML models can capture complex patterns and relationships in the input data, leading to more accurate predictions compared to traditional rule-based approaches.
5. User Interface: The project provides a user-friendly web interface for users to interact with and submit their input data. The Flask framework enables easy development of web-based interfaces and improves the overall user experience.

DISADVANTAGES

6. Model Accuracy: The accuracy of the traffic volume predictions heavily depends on the quality and generalizability of the machine learning model used. If the model is not well-trained or fails to capture important factors affecting traffic volume, the predictions may not be accurate or reliable.
7. Input Data Quality: The accuracy of the predictions is also influenced by the quality and completeness of the input data provided by users. If the input data is incorrect or missing crucial information, it may lead to inaccurate predictions.
8. Limited Feature Set: The project uses a predefined set of input features for traffic volume estimation. If there are additional factors or variables that significantly impact traffic volume but are not included in the feature set, the predictions may be incomplete or biased.

8. CONCLUSION

In conclusion, the project presented in the provided code aims to estimate traffic volume based on various input features. It leverages a machine learning model and a web application framework to provide real-time predictions and store the input data and predictions in a MySQL database. The project offers advantages such as traffic volume estimation, real-time prediction, database integration, a machine learning model, and a user-friendly interface.

However, there are potential disadvantages to consider. These include the accuracy of the machine learning model, the quality of input data, the limited feature set, scalability and performance challenges, and data privacy and security concerns.

To ensure the success of the project, it is crucial to train a high-quality and generalizable machine learning model, validate and handle input data quality, consider incorporating additional relevant features, optimize the application for scalability and performance, and implement robust data privacy and security measures.

By addressing these considerations and continuously improving the project, it has the potential to provide valuable insights for traffic planning, infrastructure management, and transportation system optimization.

9. FUTURE SCOPE

1. Enhancing the Machine Learning Model: The accuracy of the traffic volume predictions can be improved by refining the machine learning model. This may involve exploring different algorithms, optimizing hyperparameters, incorporating additional relevant features, or using advanced techniques such as deep learning or ensemble models.
2. Integration of Real-time Data: Incorporating real-time data sources such as traffic sensors, GPS data, or weather data can enhance the accuracy and timeliness of the traffic volume predictions. This could involve integrating APIs or developing data collection mechanisms to retrieve and incorporate real-time data into the prediction process.
3. Advanced Analytics and Insights: Beyond traffic volume prediction, the project can be expanded to provide more advanced analytics and

insights. This could include analyzing historical traffic patterns, identifying traffic congestion hotspots, suggesting optimal routes, or providing predictive insights for traffic planning and management.

4. Visualization and Reporting: Developing interactive data visualizations and reports can help users understand and interpret the traffic volume data more effectively. Visualizations such as charts, maps, and graphs can provide intuitive representations of traffic patterns and trends, enabling stakeholders to make informed decisions.
5. Performance Optimization: As the user base grows and the application experiences increased traffic, optimizing the performance and scalability becomes crucial. This may involve implementing caching mechanisms, load balancing strategies, or using cloud-based infrastructure to handle high traffic loads efficiently.
6. Mobile Application: Expanding the project to a mobile application can increase its accessibility and reach. A mobile app can allow users to access traffic volume predictions, submit input data, and receive notifications or alerts related to traffic conditions on their mobile devices.
7. Integration with Traffic Management Systems: Integrating the project with existing traffic management systems or platforms can provide a comprehensive solution for traffic planning and management. This could involve exchanging data with traffic signal control systems, intelligent transportation systems, or city-wide traffic management platforms.
8. User Feedback and Iterative Improvements: Gathering user feedback and incorporating it into the development process can help identify areas for improvement and prioritize future enhancements. Iterative development cycles and continuous updates based on user feedback can ensure that the project meets the evolving needs of its users.

Overall, the future scope of the project involves enhancing the machine learning model, integrating real-time data, providing advanced analytics and insights, improving performance and scalability, developing a mobile application, integrating with traffic management systems, and incorporating user feedback for continuous improvements.

10. APPENDIX

Source Code

demo.html:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <style>
    body {
      background-color: #FCAE91;
      font-family: Arial, sans-serif;
    }

    .container {
      width: 400px;
      margin: 100px auto;
      padding: 20px;
      background: #fff;
      border-radius: 8px;
      box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
    }

    h1 {
      text-align: center;
      color: #333;
      margin-bottom: 20px;
    }

    .result {
      text-align: center;
```

```
      margin-top: 20px;
      font-weight: bold;
      color: #333;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Traffic Volume Estimation</h1>
    <div class="result">
      Estimated Traffic Volume: {{ prediction_text }}
    </div>
  </div>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <style>
    body {
      background-image: url("orange.jpg");
      background-size: cover;
      font-family: Arial, sans-serif;
    }

    .Login {
```

```
  margin: 100px auto;
  width: 400px;
  padding: 20px;
  background: #fff;
  border-radius: 8px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}

h1 {
  text-align: center;
  color: #333;
  margin-bottom: 20px;
}

form {
  display: flex;
  flex-direction: column;
}

label {
  font-weight: bold;
  margin-bottom: 10px;
}

select,
input[type="number"] {
  padding: 5px;
  border: 1px solid #ccc;
  border-radius: 4px;
  margin-bottom: 10px;
}
```

```
    button {
      height: 30px;
      width: 200px;
      margin: 0 auto;
      background-color: #4CAF50;
      color: #fff;
      text-align: center;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      font-size: 16px;
    }

    button:hover {
      background-color: #45a049;
    }

    .prediction-text {
      text-align: center;
      margin-top: 20px;
      font-weight: bold;
      color: #333;
    }
  </style>
</head>
<body>
  <div class="Login">
    <h1>Traffic Volume Estimation</h1>
    <form action="{{url_for('predict')}}" method="post">
      <h3>Please enter the following details</h3>
      <br>
```

```
    <label for="holiday">Holiday:</label>
    <select id="holiday" name="holiday">
      <option></option>
      <option value="7">None</option>
      <option value="1">Columbus Day</option>
      <option value="0">Christmas Day</option>
      <option value="2">Independence Day</option>
      <option value="3">Labour Day</option>
      <option value="5">Memorial Day</option>
      <option value="6">New Year's Day</option>
      <option value="8">State Fair</option>
      <option value="9">Thanksgiving Day</option>
      <option value="10">Veterans Day</option>
      <option value="11">Washington's Birthday</option>
      <option value="4">Martin Luther King Jr Day</option>
    </select>
    <br>
    <br>
    <label>Temperature:</label>
    <input type="number" name="temp" placeholder="Temperature"
required="required">
    <br>
    <br>
    <label>Rain:</label>
    <input type="number" min="0" max="1" name="rain" placeholder="Rain"
required="required">
    <br>
    <br>
    <label>Snow:</label>
    <input type="number" min="0" max="1" name="snow"
placeholder="Snow" required="required">
    <br>
```

```
    <br>
    <label for="weather">Weather:</label>
    <select id="weather" name="weather">
      <option></option>
      <option value="0">None</option>
      <option value="1">Clouds</option>
      <option value="0">Clear</option>
      <option value="2">Drizzle</option>
      <option value="3">Fog</option>
      <option value="4">Haze</option>
      <option value="6">Rain</option>
      <option value="5">Mist</option>
      <option value="7">Smoke</option>
      <option value="8">Snow</option>
      <option value="9">Squall</option>
      <option value="10">Thunderstorm</option>
    </select>
    <br>
    <br>
    <label>Year:</label>
    <input type="number" min="2012" max="2022" name="year"
placeholder="Year" required="required">
    <br>
    <br>
    <label>Month:</label>
    <input type="number" min="1" max="12" name="month"
placeholder="Month" required="required">
    <br>
    <br>
    <label>Day:</label>
    <input type="number" min="1" max="31" name="day" placeholder="Day"
required="required">
```

```
      <br>
      <br>
      <label>Hour:</label>
      <input type="number" min="0" max="24" name="hours"
placeholder="Hour" required="required">
      <br>
      <br>
      <label>Min:</label>
      <input type="number" min="0" max="60" name="minutes"
placeholder="Min" required="required">
      <br>
      <br>
      <label>Sec:</label>
      <input type="number" min="0" max="60" name="seconds"
placeholder="Sec" required="required">
      <br>
      <br>
      <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
    </form>
    <br>
    <p class="prediction-text">{{ prediction_text }}</p>
  </div>
</body>
</html>
```

app.py

```
import os
import mysql
import mysql.connector
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
import pickle

db = mysql.connector.connect(host="localhost", user="root",
                             passwd="myPASSWORD", database='traffic',
                             auth_plugin='mysql_native_password')
cur = db.cursor()
with open('/home/seetharaman/Documents/Traffic Volume
Estimation/objects/model','rb') as file:
    model = pickle.load(file)

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

@app.route('/predict',methods=["POST","GET"])
def predict():
    input_features = [x for x in request.form.values()]
    features_values = [np.array(input_features)]
    names =
[['holiday','temp','rain','snow','weather','year','month','day','hours','minutes','secon
ds']]

    holiday = str(request.form["holiday"])
    temp = float(request.form["temp"])
```

```
    rain = int(request.form["rain"])
    snow = int(request.form["snow"])
    weather = str(request.form["weather"])
    year = int(request.form["year"])
    month = int(request.form["month"])
    day = int(request.form["day"])
    hours = int(request.form["hours"])
    mins = int(request.form["minutes"])
    secs = int(request.form["seconds"])

    data = pd.DataFrame(features_values, columns = names)
    prediction = int(model.predict(data))
    estimate = prediction

    query = ("insert into traffic_info(holiday, temp, rain, snow, weather, yr, month,
day, hour, mins, secs, estimate) values(%s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s)")
     vals = (holiday, temp, rain, snow, weather, year, month, day, hours, mins,
secs, estimate)
    cur.execute(query, vals)
    db.commit()
    db.close()
    return render_template("demo.html",prediction_text = str(prediction))

if __name__ == "__main__":
    port = int(os.environ.get('PORT',5000))
    app.run(port=port, debug=True,use_reloader = False)
```

GitHub & Project Video Demo Link

Link**:** Github (Naan Mudhalvan Repo)

Link: Video (Naan Mudhalvan Repo)

Link: Github (Personal Repo)