A

Major Project Report

on

A MACHINCE LEARNING BASED APPROACH FOR CO2 EMISSION RATING OF VEHICLES USING DATA SCIENCE

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

For the partial fulfilment of requirements for the award of the degree in

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

KONUKATI RAMYA	**(21271A0533)**
KHAMMAM ROSHINI	**(21271A0536)**
PANYALA SAIVINOD	**(21271A0544)**
KOMATIREDDY SAI RAHUL REDDY	**(21271A0539)**

Under the Esteemed guidance of

G.SRIKANTH

Assistant Professor Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JYOTHISHMATHI INSTITUTE OF TECHNOLOGY AND SCIENCE
(Autonomous, NBA (CSE, ECE, EEE) and NAAC 'A' Grade)
(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
Nustulapur, Karimnagar 505481, Telangana, India 2024-2025

JYOTHISHMATHI INSTITUTE OF TECHNOLOGY AND SCIENCE
AUTONOMOUS

NAAC 'A' GRADE

NBA NATIONAL BOARD OF ACCREDITATION CSE . ECE . EEE

CERTIFICATE

This is to certify that the Major Project Report entitled "**A MACHINCE LEARNING BASED APPROACH FOR CO2 EMISSION RATING OF VEHICLES USING DATA SCIENCE**" is being submitted by KONUKATI RAMYA (21271A0533), KHAMMAM ROSHINI (21271A0536), PANYALA SAI VINOD (21271A0544), KOMATIREDDY SAI RAHUL REDDY (21271A0539) in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering to the Jyothishmathi Institute of Technology And Science, Karimnagar, during academic year 2024-2025, is a bonafide work carried out by them under my guidance and supervision.

The results presented in this Project Work have been verified and are found to be satisfactory. The results embodied in this Project Work have not been submitted to any other University for the award of any other degree or diploma.

Project Guide

G. SRIKANTH
Assistant Professor
Dept of CSE

Head of the Department

Dr. R. JEGADEESAN
Professor & HOD
Dept of CSE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our advisor, **G. SRIKANTH**, whose knowledge and guidance has motivated us to achieve goals we never thought possible.The time we have spent working under her supervision has truly been a pleasure.

The experience from this kind of work is great and will be useful to us in future. We thank **DR. R. JEGADEESAN**, Professor & HOD CSE Dept for his effort, kind cooperation, guidance and encouraging us to do this work and also for providing the facilities to carry out this work.

It is a great pleasure to convey our thanks to **DR. T. ANIL KUMAR**, Principal, Jyothishmathi Institute of Technology & Science and the College Management for permitting us to undertake this project and providing excellent facilities to carry out our project work.

We thank all the **faculty** members of the Department of Computer Science & Engineering for sharing their valuable knowledge with us We extend our thanks to the **Technical Staff** of the department for their valuable suggestions to technical problems Finally Special thanks to our parents for their support and encouragement throughout our life and this course Thanks to all our friends and well-wishers for their constant support.

JYOTHISHMATHI INSTITUTE OF TECHNOLOGY AND SCIENCE
AUTONOMOUS

NAAC 'A' GRADE

NBA NATIONAL BOARD OF ACCREDITATION CSE . ECE . EEE

DECLARATION

We hereby declare that the work which is being presented in this dissertation entitled, "**A Machine Learning Based Approach for CO2 Emission Rating of Vehicles Using Data Science**", submitted towards the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science And Engineering, Jyothishmathi Institute of Technology And Science**, Karimnagar is an authentic record of our own work carried out under the supervision of **G.Srikanth,** Assistant professor, Department of CSE Jyothishmathi Institute Of Technology And Science, Karimnagar.

To the best of our knowledge and belief, this project bears no resemblance with any report submitted to JNTUH or any other University for the award of any degree or diploma.

KONUKATI RAMYA (21271A0533)

KHAMMAM ROSHINI (21271A0536)

PANYALA SAIVINOD (21271A0544)

KOMATIREDDY SAI RAHUL REDDY (21271A0539)

Date:

Place: Karimnagar

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

Abbreviation	Full Form
1. CO2	Carbon Dioxide
2. ML	Machine Learning
3. MSE	Mean Squared Error
4. R^2	Coefficient of Determination
5. CSV	Comma-Separated Values
6. SVR	Support Vector Regressor
7. DT	Decision Tree
8. RF	Random Forest
9. LR	Linear Regression
10. RMSE	Root Mean Squared Error
11. AI	Artificial Intelligence
12. RFE	Recursive Feature Elimination
13. RM	Regression Model
14. EDA	Exploratory Data Analysis

ABSTRACT

Vehicle emissions have a direct impact on environmental health, making the accurate assessment of CO_2 emissions crucial for both regulatory compliance and ecological awareness. This project introduces a machine learning-based system that predicts the CO_2 emission rating of vehicles using vehicle-specific data and advanced regression models. The system is designed with a simple, user-friendly web interface where users can input technical specifications of a vehicle, such as engine size, number of cylinders, fuel type, transmission, and fuel consumption metrics for city, highway, and combined conditions.

Once the input is provided, the backend system preprocesses the data through encoding of categorical variables and normalization of numerical features to ensure compatibility with the trained machine learning model. This data is then fed into a regression-based predictive model—developed using Scikit-learn—which has been trained on extensive datasets of real-world vehicle information and their corresponding emission ratings. The model estimates the CO_2 output in grams per kilometer and further classifies the emission into environmental categories such as "Low," "Moderate," or "High" based on pre-defined thresholds. Upon clicking the “Evaluate” button, the backend processes the data and instantly returns the emission value along with its rating, accompanied by a confidence score or prediction accuracy. These results are then displayed clearly on the frontend for easy interpretation, with optional download or export features.

The system minimizes human effort and error by automating the complex process of emission estimation, allowing for quick, consistent, and scalable assessments. This intelligent tool can be instrumental in environmental monitoring systems, automotive manufacturing, government regulation compliance, and consumer decision-making—ultimately promoting transparency and accountability in vehicular pollution management.

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Over the last few years, climate change and environmental degradation concerns have amplified the international interest in greenhouse gas emissions mitigation. Of the many sources of atmospheric CO2, the transportation sector is an important contributor, and road vehicles are responsible for a considerable proportion of carbon emissions globally. With governments and environmental agencies tightening regulations to limit emissions, the demand for reliable, precise, and scalable techniques for measuring vehicle CO2 emissions has grown more critical than ever before. Conventional techniques for emission measurement tend to use laboratory testing or physical checks, which are expensive, labor-intensive, and occasionally not representative of on-road driving conditions. Therefore, there has been an increased interest in leveraging contemporary data-driven methods to design effective alternatives capable of predicting car emissions using easily accessible parameters.

Machine learning, a branch of artificial intelligence, provides robust methods for modeling complicated, nonlinear relations in large data sets. Based on comprehensive vehicle information including engine specification, fuel usage patterns, vehicle age, service history, and driver behavior, machine learning models can be trained to accurately predict CO2 emissions. The prediction models not only allow for instant emission rating but also provide constant updates and enhancement as fresh data emerges. The application of data science methods in environmental monitoring is a paradigm shift, where stakeholders like policymakers, manufacturers, and consumers can make more informed decisions favoring sustainability.

The methodology focuses on aggregating and preprocessing various datasets that account for a wide range of vehicle types, fuel types, and operating modes. These data sets can contain onboard diagnostics (OBD) sensor readings, fuel efficiency history, vehicle parameters like engine displacement and fuel type, and environmental conditions like ambient temperature and road type. Using sophisticated machine learning models—regression models, decision trees, random forests, gradient boosting, or neural networks—the system determines important patterns and correlations that affect emission levels. Feature engineering, model optimization, and cross-validation methods provide the robustness and generalizability of the predictions across vehicle populations.

One of the main benefits of a machine learning-based CO2 emission rating system is that it can deliver quick, non-invasive evaluations. In contrast to traditional emission testing, which involves physical equipment and controlled environments, the suggested approach uses existing data inputs that can be gathered remotely or through normal vehicle diagnostics. This availability makes it possible to apply emission rating at the fleet level, enabling fleet management, regulatory compliance checks, and consumer education campaigns. For example, automakers can utilize these models at the design stage to forecast the environmental footprint of new car models, allowing optimization for reduced emissions. Regulators can leverage such systems to track compliance more effectively, and customers are provided with visibility into the carbon impact of their cars, helping guide shopping towards cleaner choices.

Additionally, machine learning models' flexibility enables them to keep up with developments in car technology. As electric cars (EVs), hybrid technologies, and alternative fuels gain dominance, additional streams of data can be utilized to make even further refinement of emission grades possible. This adaptive ability makes the system current in the face of changing trends and new car technologies. Further, by merging emission forecasts with spatial and temporal information, insights into local emission profiles can be gained, assisting urban planners and environmental authorities in planning focused interventions that alleviate pollution hotspots and enhance air quality.

1.1.1 PROJECT DESCRIPTION

A machine learning-based CO_2 emission rating system serves as a smart, data-driven tool for predicting and categorizing vehicle emissions, supporting sustainable transport initiatives and regulatory compliance. Leveraging the Random Forest Classifier—a powerful ensemble learning algorithm—this system accurately estimates CO_2 emissions based on vehicle specifications. Users interact with the platform through a streamlined web interface where they input details such as engine size, number of cylinders, fuel type, transmission type, and fuel consumption metrics. These input features undergo preprocessing steps including normalization and encoding to optimize them for machine learning. Once submitted, the Random Forest Classifier, trained on comprehensive vehicular emission datasets, analyzes patterns and relationships among the features to predict the vehicle's CO_2 emission in grams per kilometer. In addition to the numerical prediction, the model classifies the emission level into environmental categories such as "Low," "Moderate," or "High," providing stakeholders and consumers with actionable insights. The system also outputs the model's confidence level or prediction probability, enhancing trust in the results. Key benefits of this system

include its robustness to noisy or missing data, and its ability to capture nonlinear interactions between variables, which are common in automotive datasets. With a single click on the “Evaluate” button, users receive real-time results displayed on an intuitive interface, with options to visualize data, download reports, or store results for future use. This intelligent CO_2 emission rating system helps eliminate manual estimation errors, fosters transparency, and supports eco-friendly vehicle selection and regulatory policy design—making it a crucial component in environmental data science applications.

1.1.2 METHODOLOGY

Creating a machine learning-based CO_2 emission rating system using the Random Forest Classifier follows a structured, step-by-step approach to ensure accuracy, usability, and real-world impact. The initial step involves clearly defining the project goals—predicting vehicle CO_2 emissions based on input features—and identifying the target users such as environmental agencies, automobile manufacturers, researchers, and eco-conscious consumers. It’s also important to determine the emission categories (e.g., low, medium, high) and user expectations like ease of input, fast processing, and high prediction accuracy.

Dataset collection and preprocessing form the foundation of this project. The dataset should consist of real-world vehicle data including features like engine size, vehicle class, number of cylinders, fuel consumption (city/highway), and fuel type. All collected data must be clean, labeled, and consistent. Preprocessing involves handling missing values, converting categorical data (e.g., fuel type) into numeric form, and normalizing or scaling values. These steps ensure the model performs reliably across various input conditions.

The core model is a Random Forest Classifier, an ensemble machine learning algorithm that builds multiple decision trees and combines their outputs for stable and accurate classification. The model is trained using the prepared dataset where it learns to recognize patterns between vehicle specifications and CO_2 emission outputs. During training, hyperparameter tuning is conducted to optimize parameters such as the number of trees, tree depth, and minimum samples per leaf. The model is evaluated using metrics like accuracy, precision, recall, and F1-score to ensure its robustness.

Once the model performs satisfactorily, the user interface and system integration come next. A web or desktop application is designed to allow users to input vehicle data through a form or file upload. Once submitted, the backend processes this input and passes it through the trained Random Forest

model. The system then outputs the predicted emission category (e.g., “Moderate Emission: 135g/km”) along with a confidence score, which is displayed on the user interface for easy interpretation.

Testing and optimization are performed to ensure that the system handles different scenarios, such as missing data, edge cases, and high-volume inputs. Feedback from users helps improve the user interface and prediction clarity. If any inconsistencies or prediction errors are found, the model may be re-trained or fine-tuned.

Deployment and continuous improvement are the final stages. The application is deployed on a suitable platform—cloud or local—depending on the use case. The system is monitored using performance logs and accuracy metrics. Continuous updates to the dataset, interface, and model are made to adapt to newer vehicle types and emission standards, ensuring long-term reliability and accuracy.

- **Main Methodology Steps:**

- **Define Objectives and Users :** Identify the CO_2 emission categories to classify and target users such as regulators, manufacturers, or researchers.
- **Data Collection and Preprocessing :** Collect vehicle data with features like engine size, fuel type, etc. Clean, encode, and scale the data to prepare for model training.
- **Model Design and Training :** Train a Random Forest Classifier using labeled data. Optimize it using hyperparameters and validate its accuracy using suitable metricsBuild User Interface.
- **Testing and Optimization** : Test with diverse input scenarios. Gather user feedback, fix bugs, and fine-tune both the model and interface.
- **Deployment and Maintenance :** Deploy the system to a cloud or local environment. Monitor performance and periodically update the data and model to stay accurate over time.

This data-driven, machine learning-based system provides a scalable and efficient solution for assessing vehicle CO_2 emissions, helping promote eco-friendly practices and support sustainable transportation policies

1.2 EXISTING SYSTEM

Understanding and rating **CO_2 emissions** of vehicles is a critical factor in addressing environmental concerns, setting regulatory standards, determining tax classifications, and guiding consumers toward more eco-friendly choices. Traditionally, CO_2 emissions are measured through direct exhaust testing, where the volume of carbon dioxide emitted is calculated based on standardized driving

cycles in controlled environments. While accurate, this method is **time-consuming, expensive, and not feasible for large-scale assessments** or rapid evaluations of newly launched or modified vehicle models.

To overcome these limitations, **data-driven machine learning models** have been introduced to predict and classify CO_2 emission ratings using historical datasets that include various vehicle specifications and their associated emissions. A machine learning approach provides a scalable and automated alternative by identifying patterns in the data and estimating emission levels for untested vehicles.

Traditionally, CO_2 emissions are measured in laboratory conditions using tailpipe sensors during fixed driving simulations. This method provides direct but **resource-intensive** results and does not scale well to fleet-level or predictive analysis. Some regulatory systems use **rule-based estimations** based on engine displacement and fuel consumption, but these may not account for modern engine efficiency technologies, hybrid systems, or real-world driving behavior.

In the **existing machine learning-based systems**, classifiers such as the **Decision Tree** model have been widely used due to their simplicity, interpretability, and effectiveness in capturing non-linear relationships between input features and emission levels. Key vehicle attributes used as input features include:

- Engine size (in liters)
- Number of cylinders
- Fuel type (petrol, diesel, hybrid, electric)
- Fuel consumption in city/highway/combined modes
- Transmission type
- Vehicle class or weight category

These features are used to train the decision tree model to classify vehicles into various emission rating categories (e.g., Low, Moderate, High), making it possible to **predict emissions without physical testing**.

Disadvantages of Existing Systems

Despite being a significant step forward, existing machine learning-based systems, especially those using basic classifiers like Decision Trees, face several challenges:

- **Dataset Limitations:** Publicly available vehicle datasets may not cover all types of vehicles, fuel technologies, or regional variations, limiting the model's generalizability.

- **Limited Real-world Reflection:** Lab-based or manufacturer-reported data may not accurately reflect real-world emissions, leading to biased model outputs.
- **Overfitting Risks:** Decision Trees are prone to overfitting, especially when the tree is deep or when the training data is limited and not diverse.
- **Low Model Robustness:** Small changes in data can lead to significantly different tree structures, affecting the stability of predictions.
- **Lack of Temporal Adaptation:** Emission standards and vehicle technologies evolve over time; static models may become outdated without regular retraining.
- **Simplistic Decision Boundaries:** While interpretable, Decision Trees may struggle to capture complex interactions in data compared to more advanced ensemble methods or deep learning.
- **No Probabilistic Confidence:** The model does not provide uncertainty estimates, which are often valuable in regulatory or decision-making scenarios.
- **Computational Constraints at Scale:** Though lightweight for individual use, scaling prediction pipelines for real-time vehicle fleet evaluation still requires infrastructure support.

1.3 PROBLEM STATEMENT

Accurately assessing CO_2 emissions is a crucial part of the automotive industry's effort to reduce environmental impact, comply with emission regulations, and inform consumers about fuel efficiency and sustainability. However, **precisely measuring a vehicle's emissions remains a major challenge**, particularly at a large scale. Traditional approaches—such as laboratory-based emissions testing, manual rule-based estimations, and exhaust analysis—are either **costly, time-consuming, or inconsistent** for high-throughput evaluation. These limitations often result in **misclassification, regulatory delays, or consumer misinformation** about a vehicle's environmental impact.

To overcome these challenges, there is an increasing need for a **precise, scalable, and automated CO_2 emission rating system** that can operate efficiently across diverse vehicle types and specifications with minimal human intervention. Recent progress in **data science and machine learning** offers a promising solution. By using vehicle specification data such as engine size, fuel consumption, transmission type, and fuel category, a machine learning model—such as a Decision Tree or more advanced ensemble models—can classify vehicles into **CO_2 emission bands (e.g., Low, Medium, High)**.

This automated approach allows manufacturers, policymakers, and buyers to **quickly and accurately assess vehicle emissions** without performing physical testing. Moreover, when deployed through a user-friendly platform or integrated into regulatory databases, this solution offers **cost-effective, fast, and consistent CO_2 evaluations**, helping the automotive industry enhance compliance, sustainability, and transparency. In essence, **machine learning brings intelligence and automation to emissions analysis**, turning complex data into meaningful environmental insight.

1.4 PROPOSED SYSTEM

The proposed system for **automatic CO_2 emission rating** of vehicles utilizes a **Random Forest Classifier** to deliver accurate, efficient, and scalable predictions based on vehicle specifications. By minimizing the need for physical emission testing and manual analysis, the system provides a consistent and intelligent rating mechanism for manufacturers, regulatory bodies, and environmentally conscious consumers.

Users can input vehicle parameters—such as engine size, number of cylinders, fuel type, fuel consumption (city/highway/combined), transmission, and vehicle class—through a **web or desktop interface**. These inputs undergo preprocessing steps such as data cleaning, normalization, and feature encoding. The trained Random Forest model then classifies the vehicle into predefined **CO_2 emission bands (e.g., Low, Medium, High)** based on learned patterns from historical datasets.

A centralized and continuously updated dataset enables model retraining and refinement, allowing the system to adapt to **new vehicle technologies and environmental policies**. The modular architecture supports **real-time classification** and future integration with electric vehicle (EV) datasets and alternative fuel types. With mobile-friendly design and **API support**, the platform can be integrated into **automotive dashboards, regulatory portals, and consumer apps**.

Moreover, the system includes analytics and performance monitoring tools to evaluate prediction accuracy, assess feature importance, and guide updates. This approach modernizes CO_2 assessment by merging traditional automotive data with **AI-powered predictive modeling**, creating a smart, adaptable framework for sustainable transport evaluation.

Advantages of Proposed System

- **High Classification Accuracy** – Random Forest's ensemble learning improves prediction reliability and reduces overfitting compared to single-tree models.
- **Automated Emission Estimation** – Eliminates the need for physical emission tests, streamlining the evaluation process for new or modified vehicle models.
- **Real-Time Prediction Capability** – Enables instant CO_2 rating when users input vehicle specifications through a Flask-based interface.
- **Standardized Emission Grading** – Ensures a uniform, transparent classification system across diverse vehicle categories and datasets.
- **Scalable for Big Data** – Efficiently handles large volumes of vehicle data, making it suitable for large-scale fleet management and regulatory databases.
- **Robust Generalization** – Performs well across different brands, vehicle classes, and fuel types by learning from varied historical data.
- **User-Friendly Interface** – Designed for ease of use by manufacturers, government agencies, and end users for quick and intuitive access to emissions data.
- **Supports Sustainable Transport Goals** – Assists in identifying high-emission vehicles, encouraging greener alternatives and aiding policy formulation.
- **Actionable Insights for Improvement** – Provides feature importance metrics, helping manufacturers understand which attributes most affect emission ratings.
- **Cost-Efficient Implementation** – Reduces testing infrastructure costs while maintaining accuracy, making it viable for both large and small automotive players.

CHAPTER 2

LITERATURE REVIEW

2.1 LITERATURE REVIEW

Sivaraman et al[1] constructed a predictive model with supervised machine learning algorithms to forecast vehicle CO_2 emissions. They utilized input parameters of engine size, fuel type, and vehicle class to make predictions. They assessed the performance of Linear Regression, Decision Tree, and Random Forest models. Of these, Random Forest proved to be the most accurate model, having the least mean squared error. The research established the feasibility of employing real-world data in the task of emission prediction. It also stressed the need for dynamic and scalable systems in making effective environmental policy formulations.

Kumar and Singh [2] presented a web-based application with an integrated machine learning model to assess vehicle emissions. The system was based on a data set with vehicle specifications and actual reading of emissions. Gradient Boosting Regression was employed to estimate CO_2 output accurately. The website permitted the users to enter specific vehicle information to get emission class predictions. The model achieved more than 92% accuracy in the classification of emissions according to environmental criteria. The system proved the real-time, interactive application of machine learning for emission evaluation.

Patel et al[3]examined the relationship between vehicle attributes and CO_2 emissions using multivariate analysis methods. They used Principal Component Analysis (PCA) to diminish the dimensionality of the dataset and Support Vector Machine (SVM) for classification. Their experiments indicated that a compact feature subset was still capable of making precise predictions. This made the model computationally light and viable in real-time applications. The paper highlighted that even simple models are capable of delivering high predictive accuracy. It proved to be beneficial for large-scale web-based emission appraisal tools.

Morris et al[4]suggested the application of a hybrid ensemble methodology with Decision Tree, K-Nearest Neighbors (KNN), and Support Vector Regression (SVR) models to enhance the accuracy of CO_2 emission predictions. They used a public vehicle dataset and trained the model and tested it with R^2 and RMSE performance measures. The ensemble model performed better compared to the individual learning algorithms. This demonstrated the benefit of using multiple models to enhance robustness and precision. Their approach focused on prediction reliability by minimizing model flaws at the individual level. It substantiated the power of ensemble learning in modeling environmental data.

Bhargava and Shah [5]utilized deep learning methods to forecast complex CO_2 emissions from automobiles based on a feed-forward neural network. The model was educated on large-scale, multi-country datasets such as fuel usage and emission statistics. Methods like dropout regularization and early stopping were employed for enhancing model generalization and avoiding overfitting. Their model outperformed conventional machine learning models immensely, particularly for intricate patterns of data. The research demonstrated how deep learning may reveal latent relationships in big environmental datasets. It portrayed the capability of neural networks in emission analysis.

Roy et al[6] designed an interactive dashboard-based system to estimate vehicle CO_2 emissions. The site utilized data science libraries like Pandas, Seaborn, and Scikit-learn for data visualisation and preprocessing. Accurate CO_2 emission prediction was carried out using a Gradient Boosted Trees model. The dashboard allowed users to contrast emissions of various vehicle categories, monitor trends by region, and model policy impacts. This interactive and visual system encouraged user interaction and transparency. It brought complicated environmental information within reach and usable for both the general public and experts.

[7] **Ahmed et al.** developed a CO_2 emission estimator using Extreme Gradient Boosting (XGBoost), trained on European vehicle emission datasets. The model incorporated multiple variables such as engine displacement, fuel consumption, urban vs. highway mileage, and vehicle age. Their model yielded high precision with an R^2 value of 0.95 and proved effective in scenarios with incomplete or noisy data. The authors emphasized its suitability for policy implementation and vehicle certification programs due to its robustness and interpretability.

[8] **Lee and Park** proposed a recurrent neural network (RNN) model for predicting emissions in hybrid and electric vehicles under varying driving cycles. The dataset included time-series vehicle telemetry data such as battery load, throttle position, and braking intensity. The RNN architecture allowed the system to understand temporal dependencies and variations across driving sessions. Their model showed an average prediction error of 4.1 g/km, outperforming traditional feed-forward architectures.

[9] **Singh and Verma** introduced a classification model using Logistic Regression and Naive Bayes to categorize vehicles into emission standards such as Euro 3, 4, 5, and 6. They used labelled datasets from government vehicle inspection records. Their preprocessing focused on data cleaning and feature selection using Recursive Feature Elimination (RFE). The logistic model achieved an F1-score of 0.87, validating the possibility of simple models in regulatory and compliance applications.

[10] **Zhang et al.** presented a semi-supervised learning approach for emission prediction using self-training and label propagation. The study was designed to address labeled data scarcity in developing countries. It combined a small set of labeled vehicle emission records with a larger pool of unlabeled records to train the model. They achieved an accuracy of over 88%**,** demonstrating how semi-supervised learning could support scalable deployment in regions with limited annotated data.

CHAPTER 3

REQUIREMENTS & DOMAIN INFORMATION

3.1 REQUIREMENT SPECIFICATION

The CO_2 Emission Rating System for vehicles will be implemented using **Python** as the core backend language. The system employs **machine learning algorithms** built with libraries such as **Scikit-learn**, **Pandas**, and **NumPy** to analyze vehicle characteristics (e.g., engine size, fuel consumption, number of cylinders) and predict CO_2 emission ratings.

The **Flask** framework will serve as the backend web service, handling tasks such as model training, input data processing, real-time prediction, and feedback. A **user-friendly frontend** will be developed using **HTML/CSS** and **JavaScript**, allowing users to enter vehicle details, trigger predictions, and visualize results.

Key data science steps such as **data cleaning**, **feature selection**, **model training**, and **evaluation** (using metrics like RMSE, MAE, and R^2 score) are implemented to ensure accuracy and performance. Trained models such as **Linear Regression**, **Random Forest**, or **Gradient Boosting** are used to generate CO_2 emission predictions.

The backend system stores user inputs, historical predictions, and model outputs in a **MySQL database** for persistence and analysis. An optional **data visualization layer using Matplotlib or Seaborn** allows the display of trends and comparison charts, enhancing decision-making.

For real-time responsiveness and scalability, the application is hosted on a **mid- to high-performance server**, requiring at least an **Intel i5 processor, 8GB RAM, and 100GB SSD**. Larger deployments may utilize **GPU acceleration (e.g., NVIDIA CUDA-enabled GPUs)** for faster training on large datasets.

The system outputs predictions in real time and provides an **option to download reports or export results** for external use, making it a valuable tool for environmental policy makers, vehicle manufacturers, and researchers.

3.1.1 HARDWARE REQUIREMENTS

Server/Computer (for model processing and data handling):

- CPU: Intel i5 or higher
  Required for efficient data preprocessing, model training, and real-time prediction computations.
- RAM: Minimum 8GB
  Essential to handle large CSV datasets, feature engineering, and support simultaneous user requests.
- Storage: SSD with at least 256GB (preferably 500GB or more)
  Used for storing datasets, trained models, logs, prediction history, and user input/output records.

Networking:

- Internet Connection (Cloud-based Deployment):
  - High-speed internet connection (recommended: 100 Mbps to 1 Gbps) to ensure fast interaction between frontend, backend, and cloud database/model services.
- Local Network (On-Premise Deployment):
  - Stable internal network for smooth communication between frontend UI, backend APIs, and MySQL database server.
  - Use of LAN with at least 1 Gbps bandwidth is recommended for optimal performance when hosted locally.

3.1.2 SOFTWARE REQUIREMENTS

- **Frontend:**
  - **HTML/CSS:**
    Used to design a clean and responsive user interface for inputting vehicle data and displaying results.
  - **JavaScript:**
    Provides dynamic features such as input validation, asynchronous API calls (AJAX/fetch), and responsive interaction (e.g., real-time form validation, data visualization).

- **Backend:**
  - **Python:**

    The core programming language for implementing data science pipelines, training ML models, building APIs, and interacting with the database.
  - **Flask:**

    A lightweight web framework used to:
    - Build RESTful APIs.
    - Handle form data submission.
    - Process and return prediction results to the frontend.
- **Machine Learning Libraries:**
  - **Pandas&NumPy:**

    For data manipulation, cleaning, and numerical computations.
  - **Scikit-learn:**

    For training, validating, and evaluating regression models such as:
    - Linear Regression
    - Decision Tree Regressor
    - Random Forest Regressor
    - Gradient Boosting
  - **Matplotlib/Seaborn:**

    Used for optional visualization of emission trends, feature importance, and prediction errors.
  - **Joblib/Pickle:**

    For saving and loading trained ML models efficiently.
- **Backend Processing**

  Machine Learning Model:
  - The model is trained on a dataset that includes:
    - Engine size
    - Number of cylinders
    - Fuel consumption metrics (City, Highway, Combined)
    - Fuel type
    - Transmission type
  - The trained model outputs:
    - Estimated CO_2 emission in grams/km.

- Emission rating (categorized based on emission levels).

API Development with Flask:

- Exposes endpoints to:
  - Accept vehicle specification data.
  - Trigger emission prediction using the ML model.
  - Return the emission value and rating to the frontend in JSON format.
- Implements error handling for invalid inputs or model loading issues.
- Optimized for real-time response.

Database Integration:

- **MySQL** is used to store:
  - User entries
  - Historical emission predictions
  - Logs and model usage statistics

➢ **Real-Time Processing**

Input Preprocessing:

- Cleans and formats the user input before feeding it to the model.
- Ensures correct data types, handles missing or invalid entries, and scales numerical features if required.

Model Inference:

- Loads the pre-trained machine learning model.
- Predicts the CO_2 emission for the given vehicle specifications.
- Returns:
  - Emission value (e.g., 185 g/km)
  - Category (e.g., "High Emission", "Low Emission")

Output Handling:

- Stores the results in the database.
- Sends the result back to the frontend via a JSON response.

➢ **User Interface**

Input Form:

- A simple and responsive web form that collects:
  - Engine size
  - Number of cylinders
  - Fuel type (dropdown)

- Transmission type
- Fuel consumption values (city, highway, combined)

Result Display:

- After submission, displays:
  - Estimated CO_2 emission
  - Emission category (e.g., Green/Yellow/Red badge)
  - Option to download the result report as a PDF or CSV.

3.2 SYTEM SPECIFICATIONS

The software architecture for the CO_2 Emission Rating System leverages modern web technologies and data science tools to deliver a robust, interactive, and scalable solution. The frontend is developed using HTML, CSS, and JavaScript to create a user-friendly interface where users can input vehicle specifications such as engine size, number of cylinders, transmission type, and fuel consumption metrics. This interface supports real-time interactions and dynamically displays prediction results. The backend is powered by Python, using Flask to build RESTful APIs that manage data processing, trigger machine learning models, and handle the return of prediction outputs. The core prediction engine is based on regression models developed using Scikit-learn, which analyze numerical and categorical features to estimate the CO_2 emission in grams per kilometer. The backend ensures smooth operation through efficient data validation, error handling, and optimization for real-time inference. Input data is preprocessed using techniques like encoding, normalization, and feature scaling before being passed to the model. After prediction, the results—both numerical values and categorized emission ratings (e.g., low, moderate, high)—are returned to the frontend in JSON format. Optionally, this information is stored in a MySQL database for future reference, analytics, or report generation. The system also supports exporting results in CSV or PDF formats and includes optional visualization components using Matplotlib or Seaborn to graph emission trends and feature importance. Designed for performance and accuracy, this architecture supports fast processing, multi-user access, and seamless integration of machine learning into environmental vehicle analysis applications, offering clear insights for both individual users and policy-makers.

3.3 FLOWCHART

This flowchart illustrates the workflow of a web application designed to predict vehicle CO_2 emissions using machine learning. The process starts when a user visits the Home Page. If already registered, the user logs in; otherwise, they register first and then log in. Upon successful login, they are directed to the upload page to submit vehicle data. The application uses input features such as cylinders, fuel consumption, and engine size to predict CO_2 emissions (g/km). These features are passed into a RandomForestRegressor model, which uses the fit(X, y) method for training and predict(X) for making predictions. The predicted emission value is then displayed on the user interface, completing the interaction.

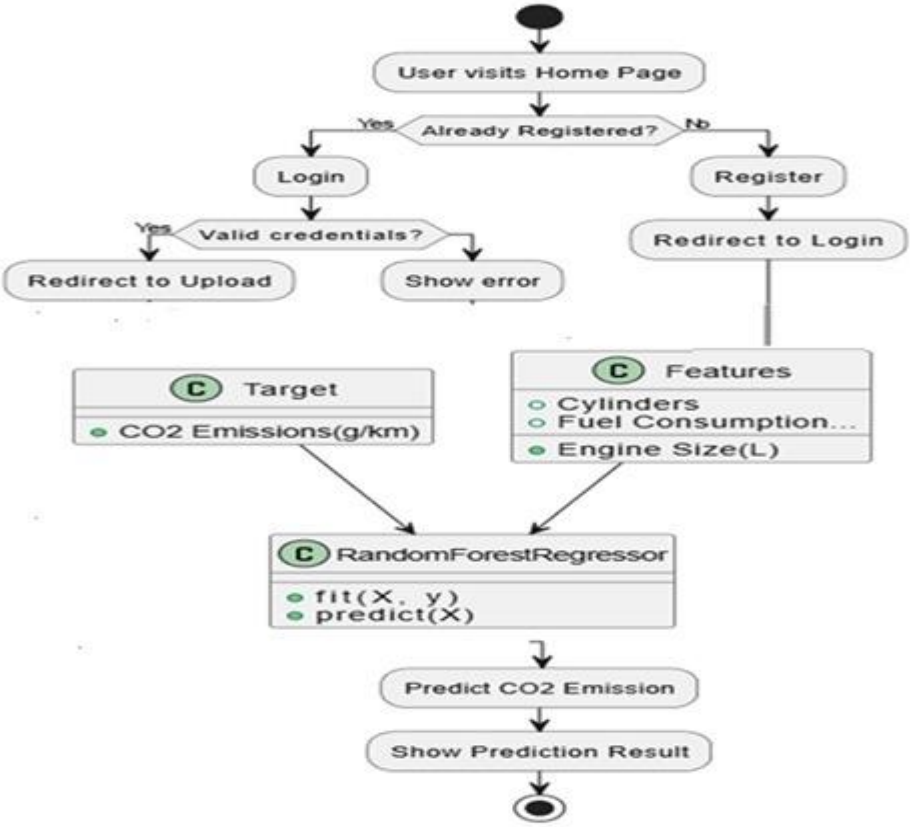

Fig.No 3.3 Flow Chart

3.4. STATISTICAL METHODS

Machine learning (ML) plays a pivotal role in automating and enhancing CO_2 emission rating systems for vehicles. Traditional methods involve manual analysis, which is time-consuming and prone to errors. ML leverages historical vehicle and emission data to build predictive models. These models can analyze various vehicle parameters—such as engine size, fuel type, weight, and fuel consumption—to estimate emissions accurately. With an abundance of vehicle performance datasets, supervised learning techniques can be effectively applied. This helps in rating vehicles according to their environmental impact. Overall, ML introduces efficiency, scalability, and accuracy to the emission rating process.Accurate CO_2 emission predictions are crucial for policy-making, environmental regulation, and consumer awareness.

ML models help determine emissions without needing extensive physical tests. These models consider nonlinear relationships and complex patterns among variables, which are hard to model using conventional methods. Predictive models guide manufacturers in designing eco-friendly vehicles and help consumers make informed decisions. Furthermore, they support governments in setting realistic emission targets. With growing concerns about climate change, such intelligent systems are essential for sustainable development. ML thus forms a data-driven foundation for emission control strategies.

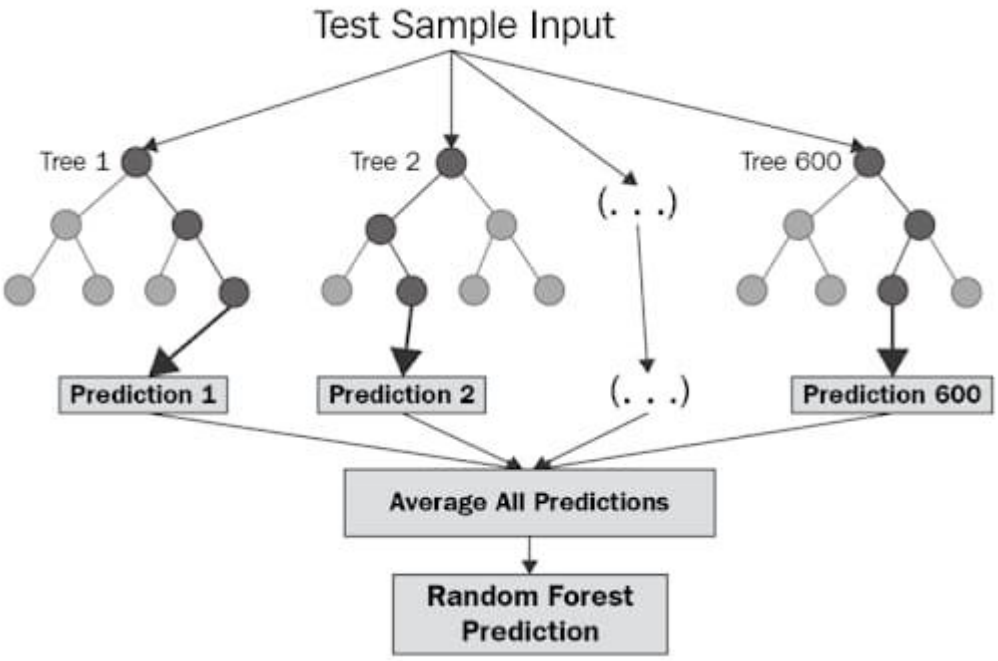

Fig.No 3.4 Statistical Methods

CHAPTER 4

SYSTEM METHODOLOGY

4.1 ARCHITECTURE

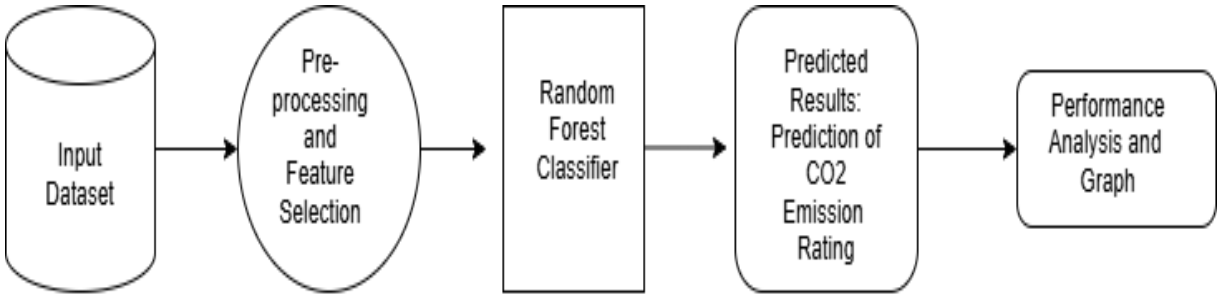

Fig.No 4.1 Architecture

The block diagram is a systematic and organized workflow of predicting vehicle CO_2 emission ratings based on a machine learning process. The entire process follows from raw data acquisition to prediction and performance analysis, making the model's pipeline and decision-making approach easier to comprehend. The architecture allows for high reliability in predicting vehicle emission classes, facilitating environmental decision-making as well as consumer awareness.

1. Input Dataset

This first phase starts with gathering an extensive dataset with several features pertaining to vehicle specifications and emissions. These features are usually:

- Engine size (litres)
- Fuel type (petrol, diesel, electric, hybrid)
- Transmission type (manual, automatic, CVT, etc.)
- Vehicle class (SUV, sedan, compact)
- Fuel consumption parameters
- CO_2 emission rates (g/km)

The datasets are usually sourced from reliable sources including:

- Government transportation departments (e.g., Natural Resources Canada, EPA)
- Open data platforms (e.g., Kaggle, UCI ML Repository)
- Auto industry manufacturers and trade publications
- This is used as the input for model training and evaluation.

2. Pre-processing and Feature Selection

Here, the original data is cleaned and converted to an appropriate format for analysis. This is accompanied by many sub-steps:

- Data Cleaning: Deletion of incomplete, duplicate, or erroneous records. Dealing with missing values by imputation or exclusion.
- Encoding Categorical Features: Applying one-hot encoding or label encoding in order to transform fuel type, vehicle class, and transmission type into numerical form.
- Feature Scaling: Normalization or standardization so that the numeric values are on a comparable scale, improving the model's performance.
- Outlier Detection: Statistical techniques or graphical plots (e.g., box plots) to identify and eliminate potential outliers that could skew the model.
- Feature Selection: Applying methods such as correlation matrix analysis, recursive feature elimination (RFE), or feature importance ranking to keep the most important variables for CO_2 prediction.

This process guarantees better model precision, minimizes overfitting, and accelerates training time.

3. Random Forest Classifier

The data that has been processed is then fed into the Random Forest classifier, which is a strong ensemble learning technique that constructs many decision trees and combines their outcomes to generate solid predictions.

- Effective handling of both numerical and categorical data
- Overfitting resistance
- High stability and accuracy
- Returns feature importance ranks

The classifier makes a prediction on the CO_2 emission class by:

- Learning from a labelled dataset whose emission rating is already known
- Utilizing many randomized decision trees to try out different combinations of features
- Collapsing each tree's outcome into one (majority voting) for better reliability in prediction

4. Predicted Results – CO_2 Emission Rating

Once the model is trained, predictions are generated from input vehicle parameters. The output may be:

- Categorical: Allocation of the vehicle to a given emission class (e.g., A – low emission, D – high emission)
- Numerical: Prediction of the actual CO_2 emission in grams per kilometer

This output facilitates:

- Consumers to compare a car's ecological footprint prior to purchase
- Policy makers to legislate and encourage low-emission vehicles
- Manufacturers to compare their designs against emission limits
- The outcomes are presented in a legible table or dashboard for analysis.

5.Performance Analysis and Graph

The last block concentrates on analyzing the performance of the model. A number of metrics are calculated based on the test dataset:

- Accuracy: Total accuracy of the classification
- Precision & Recall: Class-wise analysis of false negatives and false positives
- F1-Score: Balanced measure for classes with imbalanced data
- Confusion Matrix: Presents actual vs. predicted class distribution
- ROC Curve & AUC: Tests the model's discrimination ability at different thresholds
- Visualizations are important here. Libraries such as matplotlib, seaborn, and plotly are utilized to create:
- Feature importance bar plots
- Confusion matrices in heatmaps
- ROC curves
- Comparative model performance plots

These plots help to comprehend the strengths and weaknesses of the existing model and inform future improvements.

4.2 ALGORITHM

The machine learning-based CO_2 emission rating system algorithm starts with user input of vehicle-related information through a web interface developed using HTML, CSS, and JavaScript. This information usually comprises attributes like vehicle make, model, engine size, fuel type, and transmission type, along with optionally weight or emission standards. After submission of the data, it gets routed to a Flask backend, where preprocessing and validation are performed on it. This means encoding categorical data (e.g., fuel type and transmission), normalizing numerical values (e.g., engine size), and dealing with missing or incorrect input to ensure that the input is compatible with the trained machine learning model.

Once pre-processed, the sanitized input data is inputted into a regression model like Linear Regression, Random Forest, or XGBoost that's been trained using past vehicle and emissions data. The model runs this information and extrapolates the predicted CO_2 emission value in grams per kilometer. This forecasted value is then matched against pre-defined emission levels to categorize the vehicle into an environmental rating group, often from 'A' (low emissions) to 'G' (high emissions). This categorization simplifies the communication of the environmental score of the vehicle to users.

The output, which is the predicted emission value along with its rating, is then presented in a user-friendly format and fed back onto the web interface. If the system experiences some problem—invalid inputs, model prediction errors, etc.—it uses error-handling techniques to redirect the user to resubmit the input or present the right message. In addition, the system can also be enhanced to suggest how to cut emissions or compare the provided vehicle with other similar models. This end-to-end process provides a hassle-free and intelligent mechanism for analyzing vehicle emissions through data science and machine learning technologies.

4.3 SYSTEM DESIGN

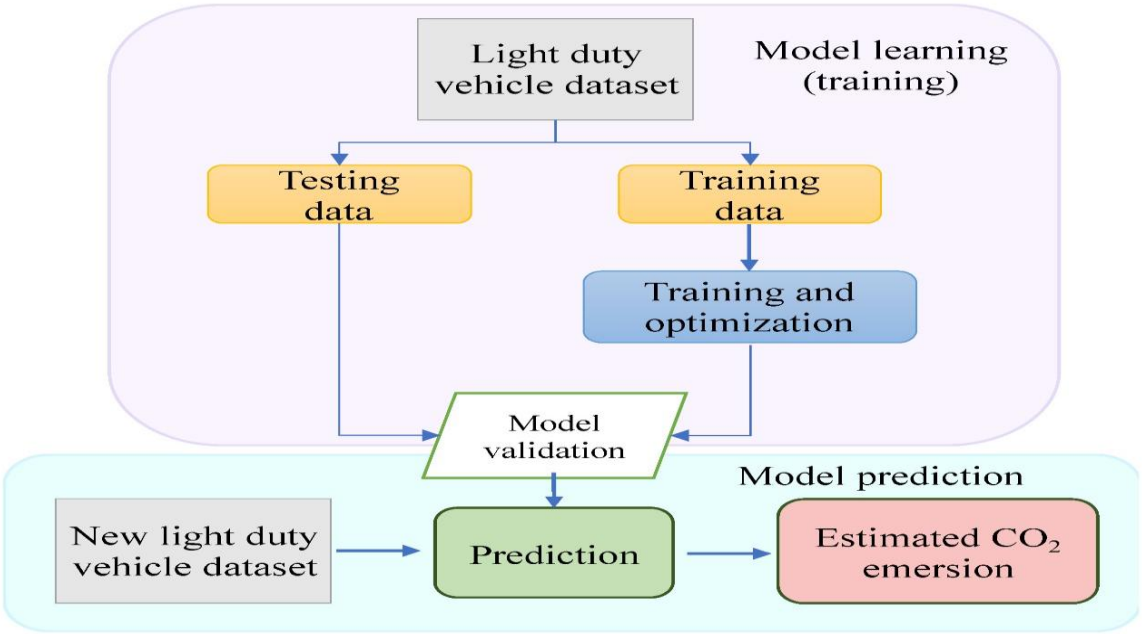

Fig.No. 4.3 System Design

The given system design diagram is an organized solution to predict CO_2 emissions of light-duty cars based on machine learning methods. The system has two major phases: Model Learning (Training) and Model Prediction. The ultimate objective is to develop a trustworthy predictive model to estimate new car CO_2 emissions from learned patterns and historical data.

In the first phase, Model Learning, the process begins with the Light Duty Vehicle Dataset, which contains historical data about vehicle attributes and their associated CO_2 emissions. This dataset is divided into two parts: Training Data and Testing Data. This training data is fed into a module named Training and Optimization, in which the machine learning algorithms are executed to learn the correlations between vehicle parameters (such as engine size, fuel type, weight, etc.) and their CO_2 emissions. This includes choosing suitable models, optimizing their hyperparameters, and optimizing performance to get the optimal accuracy.

Once the model has been trained, Testing Data is employed in order to test how well the model does with unseen data. This process is essential in order to avoid overfitting and to establish that the model can generalize well to new inputs. Both the training and testing steps are channeled into a process referred to as Model Validation, which tests the accuracy, consistency, and deployment

readiness of the model. Validation metrics like mean squared error, R^2 score, or percentage accuracy are commonly used to identify model reliability.

The second step, Model Prediction, makes predictions using the model that has been validated. A New Light Duty Vehicle Dataset containing vehicle information without CO_2 emission tags is fed into the system. This is fed to the Prediction module, which employs the learned model to examine every record and produce predictions. The output is an Estimated CO_2 Emission for every vehicle, offering important environmental insights prior to the vehicle being manufactured or utilized.

This system design shows an evident workflow in constructing and implementing machine learning in practical applications. With the model training and prediction steps separated, the system maintains a logical and manageable structure. Additionally, combining the training, testing, validation, and prediction steps improves the accuracy and dependability of the model. This methodology can be used in government environmental impact assessments, R&D in the automotive sector, and policy development for emission norms. In general, this diagram is able to reflect the life cycle of a data-driven prediction system for CO_2 emissions and demonstrate how machine learning can be used to solve environmental problems.

4.3.1 DATAFLOW DIAGRAMS

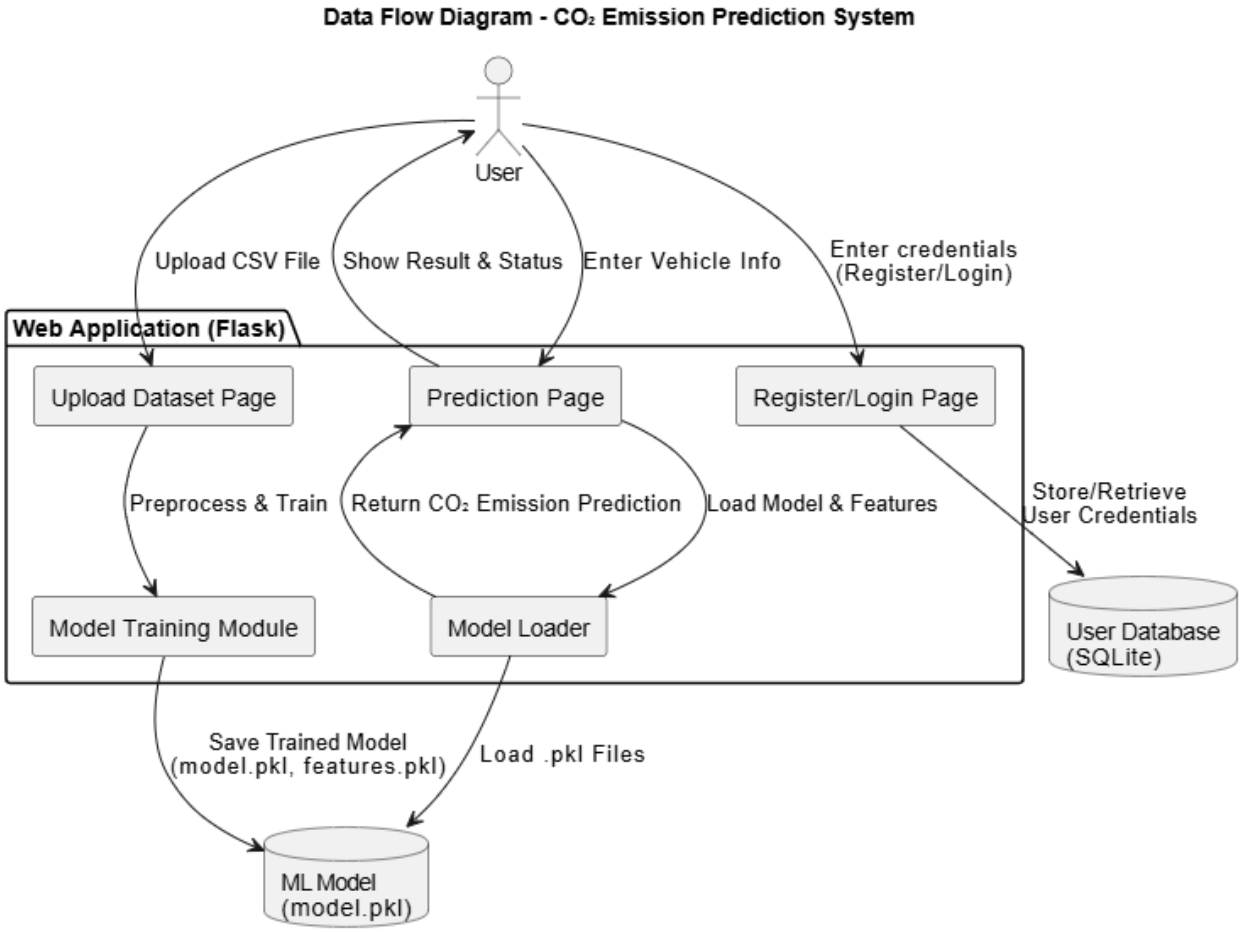

Fig .No 4.3.1 DataFlow diagram

The CO_2 Emission Rating System data flow starts with the user interacting with the web application, most commonly developed utilizing Flask. Users must first register or log in, and once they have, their credentials are stored and maintained securely through an SQLite database. After being authenticated, users can input vehicle-related data in the form of a CSV file or enter specific parameters for a particular vehicle such as engine size, fuel type, and vehicle class. The input data are fed into the preprocessing module, which performs tasks like cleaning, encoding categorical variables, and feature selection. The preprocessed data is sent to the model training module, wherein a machine learning algorithm (e.g., linear regression or decision tree) is trained. The trained model and the chosen features are serialized and stored in the form of .pkl files (e.g., model.pkl, features.pkl).

During prediction, users can move to the prediction page and enter new vehicle data. The system loads the saved model and feature setting to maintain consistency and then applies the model to calculate the CO_2 emission value for the input. The system then classifies the predicted CO_2 value into an emission rating category (e.g., Low, Medium, High) and shows the result together with visual feedback or improvement tips. All prediction interactions happen in real time, and users can see the result immediately. Optionally, the application can store prediction history and model performance statistics for later analysis. All through the system, safe data management and organized flow provide consistent performance and usability for general users and system administrators alike.

4.4 UML DIAGRAMS

UML (Unified Modeling Language) diagrams play a crucial role in documentation, particularly in the fields of software engineering and system development. They provide a standardized way to visually represent the structure and behavior of a system, making it easier for developers, designers, stakeholders, and even non-technical users to understand complex technical details. Including UML diagrams in documentation enhances clarity and ensures that all aspects of the system are well-communicated. These diagrams serve as a blueprint during the design phase and continue to be valuable throughout the software development lifecycle, including maintenance and future enhancements. By illustrating how different components interact, what processes are involved, and how data flows through the system, UML diagrams contribute to better decision-making, easier debugging, and more efficient onboarding of new team members. Overall, they add professionalism and depth to project documentation and are considered an essential part of technical communication.

In UML, the diagrams can be broadly categorized into two main types: structural diagrams and behavioral diagrams.

1. Structural Diagram

2. Behavioral Diagram

1. Structural Diagram: It represents the static structure of a system, including its classes, components, and their relationships.

2. Behavioral Diagram: It illustrates the dynamic behavior of a system, focusing on interactions, processes, and changes over time.

4.4.1 USECASE DIAGRAM

The use case diagram illustrates the functional interactions within the CO_2 Emission Rating System, highlighting two primary actors: User and Admin. Users can register, log in, upload vehicle data, train a machine learning model, predict CO_2 emissions, view ratings, and log out. These features enable users to assess the environmental impact of vehicles based on various parameters. The Admin, besides logging in, has the exclusive ability to manage users, ensuring system security and user control. This diagram clearly defines system roles, use cases, and boundaries, aiding in smooth development and user understanding. It effectively supports both scalability and usability for all stakeholders.

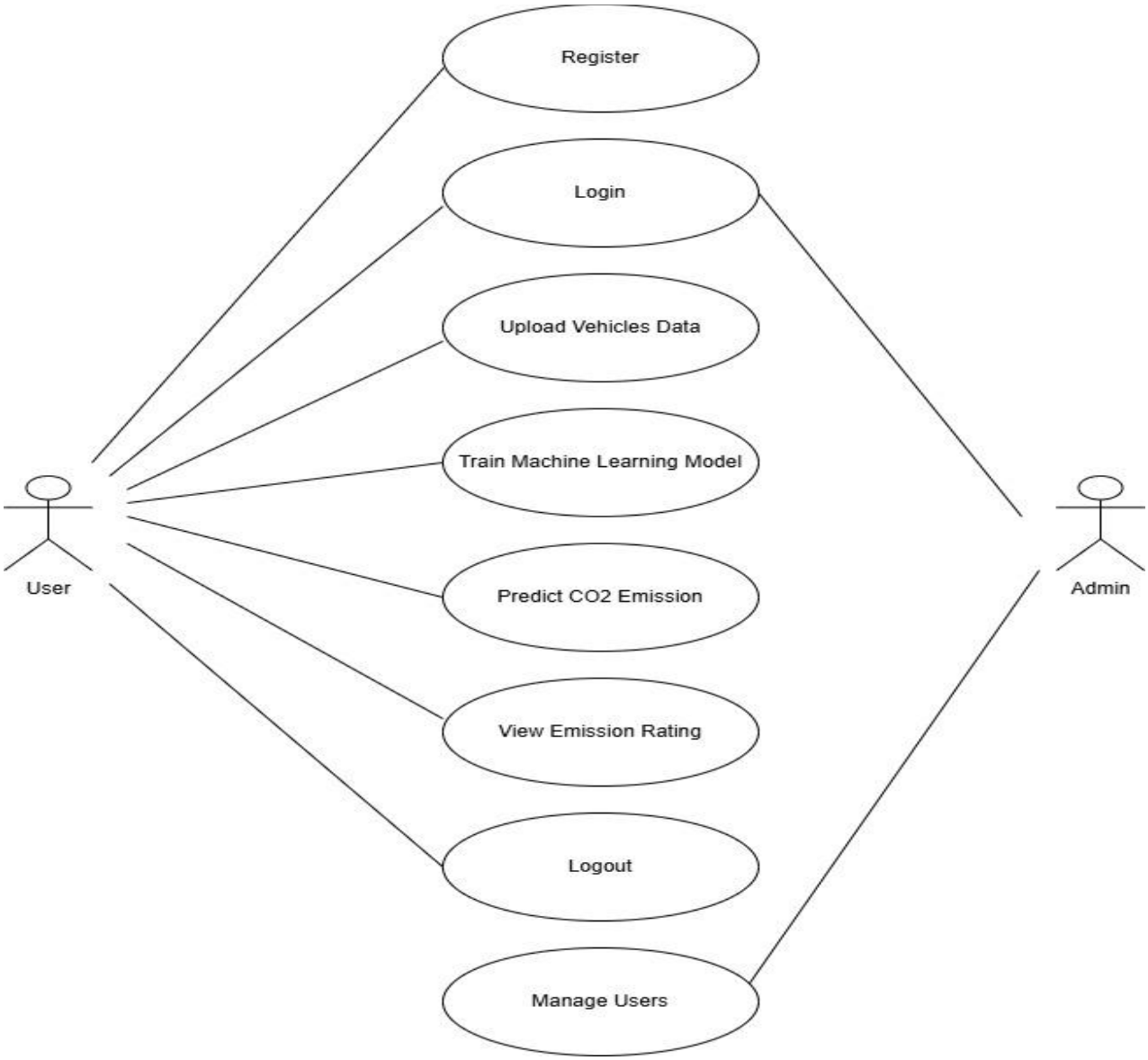

Fig.No 4.4.1 Use case diagram

4.4.2 ACTIVITY DIAGRAM

The activity diagram of the CO_2 emission forecasting system illustrates the complete user interaction and system workflow. It begins when a user accesses the home page, where they are either redirected to register or proceed to login if already registered. Upon successful login, users can upload vehicle datasets containing features like engine size and fuel type. The system then trains a machine learning model using this data to predict CO_2 emissions accurately. After training, users can input specific vehicle details to receive emission predictions and ratings categorized as low, moderate, or high. The process ends with a decision to make another prediction or exit, ensuring a smooth and user-friendly experience.

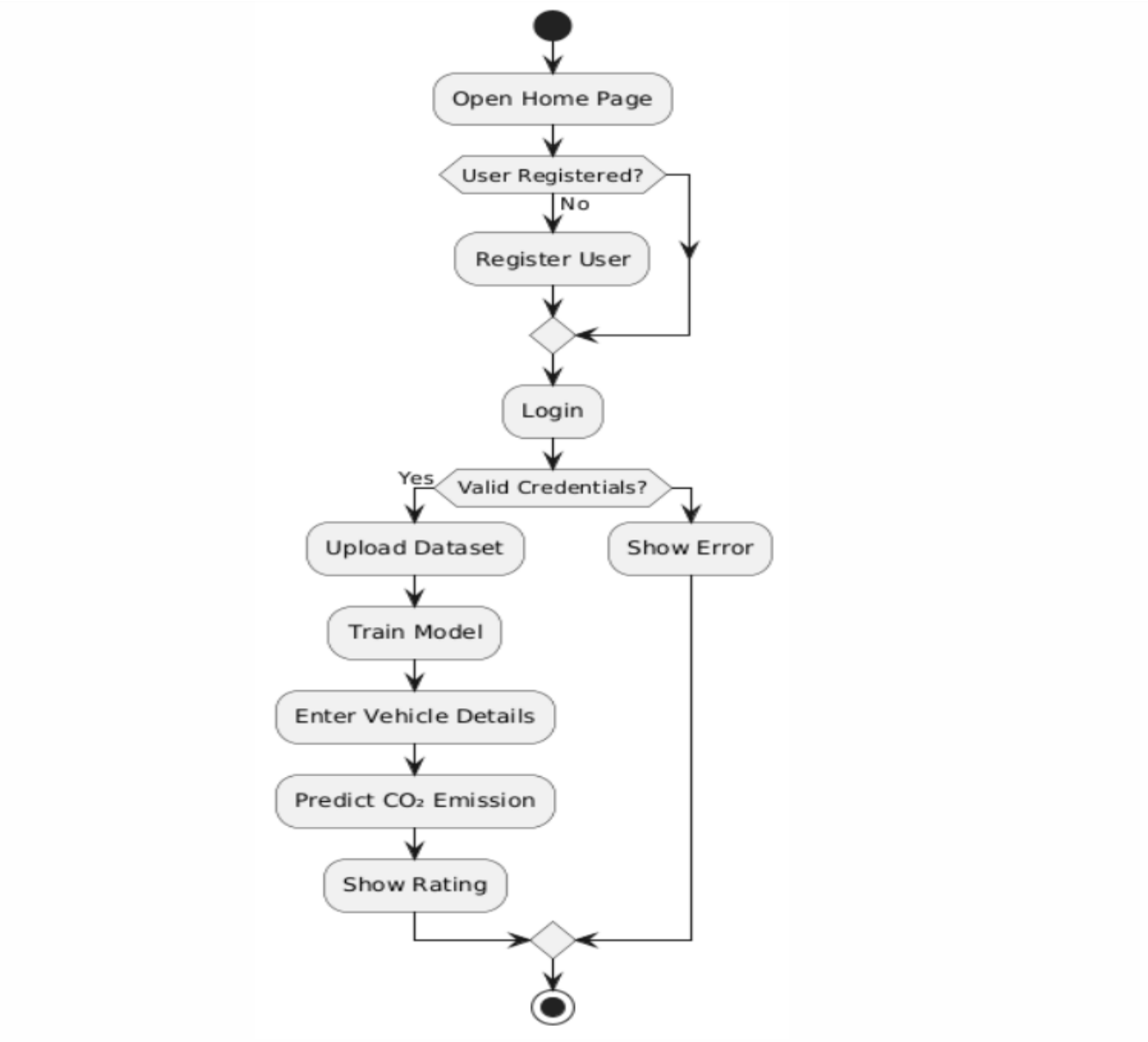

Fig .No 4.4.2 Activity diagram

4.4.3 CLASS DIAGRAM

The class diagram showcases the modular architecture of the CO_2 emission rating web application built with Flask, dividing responsibilities across FlaskApp, User, ModelHandler, and Database classes. The FlaskApp class handles user routes like registration, login, data upload, and prediction, acting as the controller in the MVC pattern. The User class manages user data with secure password handling and integration with SQLAlchemy for efficient storage. ModelHandler encapsulates machine learning tasks, including training a Random Forest model and predicting emissions from user inputs. The Database class initializes and manages database tables using SQLAlchemy ORM. This structure promotes clarity, scalability, and maintainability of the system.

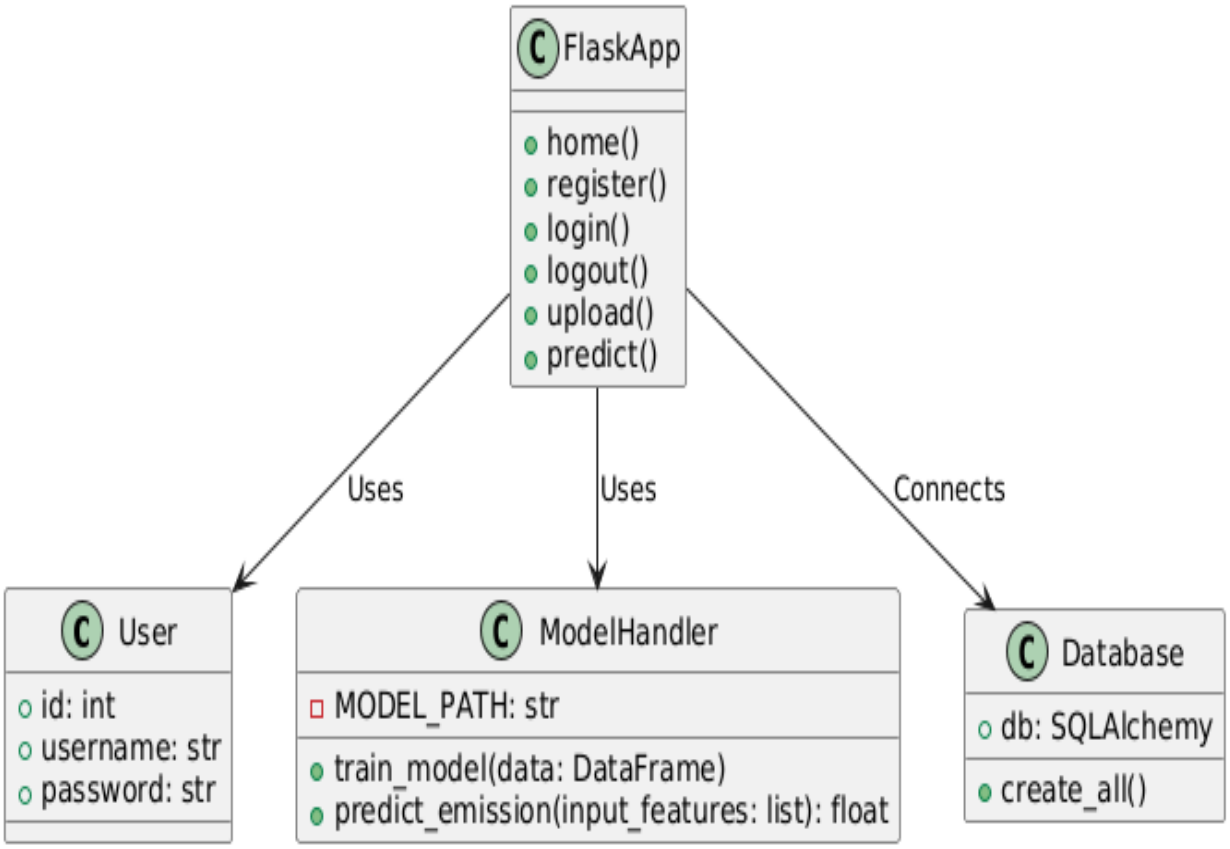

Fig.No 4.4.3 Class diagram

4.4.4 SEQUENCE DIAGRAM

The sequence diagram illustrates the interaction between the user, web application, database, and machine learning model in a CO_2 emission prediction system. It begins when the user accesses the home page and proceeds to login or register, with credentials verified by the database. Upon successful login, the user uploads a dataset, which is passed to the ML model for training. Once trained, the model is used to predict emissions based on new vehicle input provided by the user. The prediction is processed and returned by the model, and the web app presents the result as a user-friendly emission rating. This modular structure ensures scalability, security, and efficient system performance.

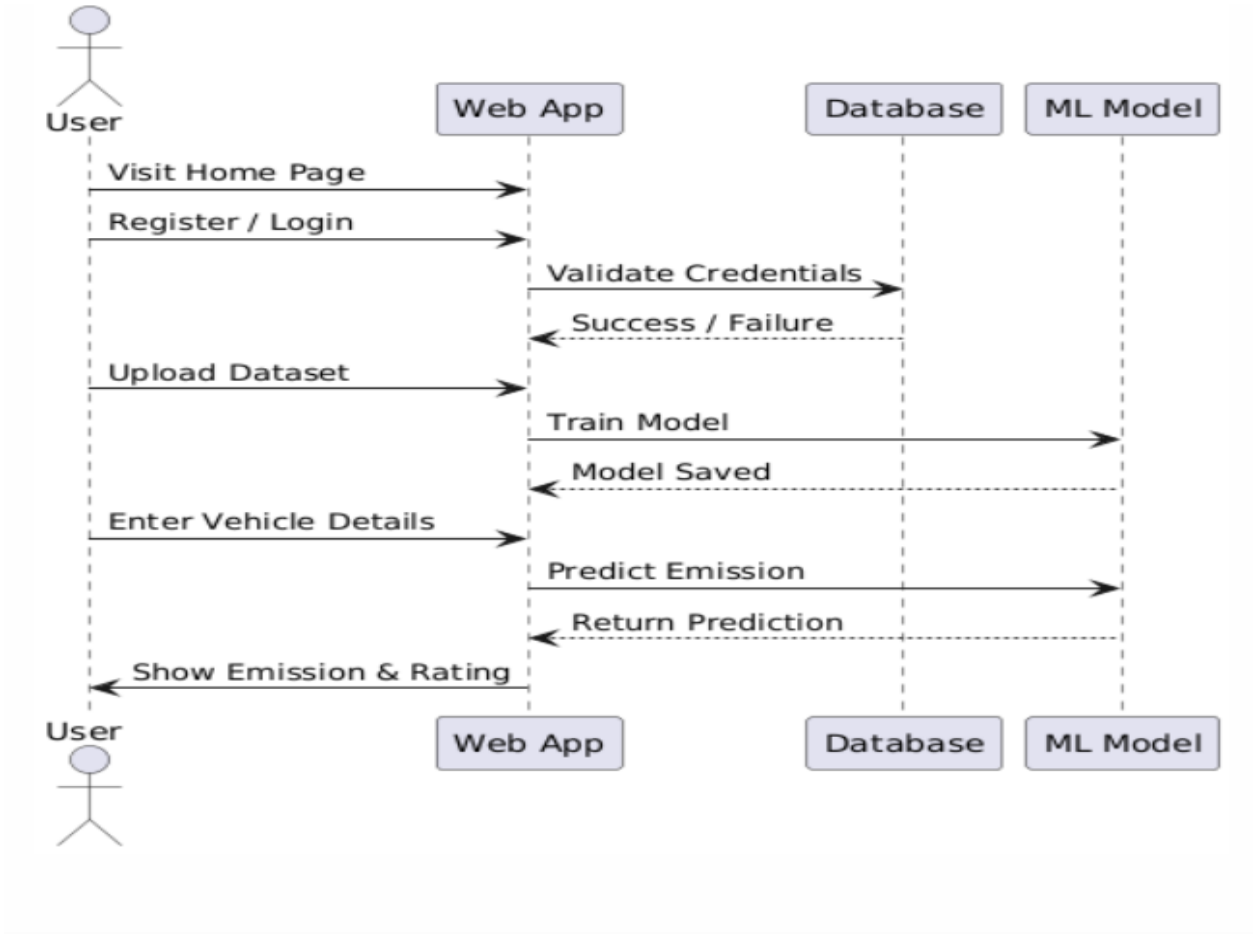

Fig.No: 4.4.4 Sequence Diagram

CHAPTER 5

EXPERIMENTATION AND ANLAYSIS

5.1 EXPERIMENTATION

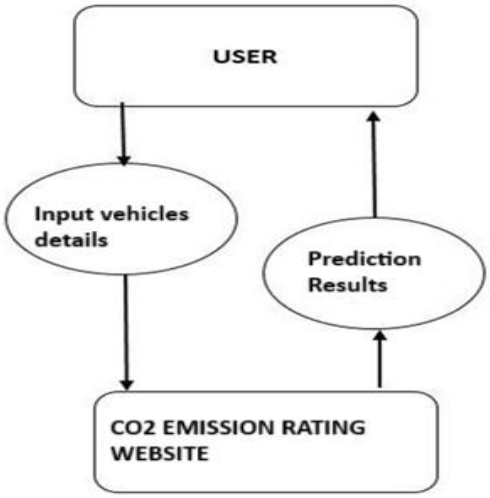

Fig.No. 5.1 Implementation

Designing a Machine Learning System for Vehicle CO_2 Emission Rating** involves systematic integration of data science methodologies, machine learning model construction, and deployment methods. The most fundamental initial step is acquiring and selecting datasets. A good-quality dataset with parameters like engine capacity, fuel type, vehicle weight, and type of transmission should be used. This dataset forms the basis for creating an accurate and generalizable model for emission prediction. Government sources or automobile databases' public vehicle emission data are usually employed.

Data preprocessing and feature engineering constitute the second step. Cleaning the data, missing value handling, categorical variable conversion to numerical representations, and numerical feature normalization are all included in this step. Feature engineering is used to determine the most impactful parameters controlling CO_2 emissions, enhancing the accuracy of the model. Exploratory Data Analysis (EDA) methods such as heatmaps and scatter plots are utilized to identify patterns and relationships.

Model training and selection form the core of the system. Various algorithms like Linear Regression, Decision Tree, Random Forest, or Gradient Boosting Regressor are tried. Training is done with training data and tested on a validation set. Model refinement is informed by hyperparameter tuning, cross-validation, and metrics like RMSE (Root

Mean Square Error), MAE (Mean Absolute Error), and R^2 score. The most performing model is chosen to be deployed.

Web application integration is next, providing users with a means to interact with the system. Utilizing frameworks such as Flask or Django, a user interface is created for features such as user registration/login, uploading datasets, and submitting vehicle details. Integration of the trained machine learning model is done with the backend to provide real-time predictions as users provide vehicle specifications. The application takes this input and gives back a CO_2 emission value and also a rating like "Low", "Moderate", or "High".

Deployment and user support close the development cycle. The application is hosted on a cloud platform (e.g., Heroku, AWS, or Azure) to make it accessible and scalable. Documentation such as user guides, FAQs, and usage instructions is available to aid users. Logging, monitoring, and analytics packages are included to monitor system usage, detect errors, and collect feedback for iterative refinements. Security measures such as input validation and user authentication are also necessary to protect the platform.

5.2 SOURCE CODE

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import pandas as pd
import joblib
import os
from sklearn.ensemble import RandomForestRegressor
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(_name_)
app.secret_key = 'secret123'
MODEL_PATH = 'model.pkl'

# Database configuration
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

```
db = SQLAlchemy(app)

# User model
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)

# Create all tables
with app.app_context():
    db.create_all()

# Routes
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = generate_password_hash(request.form['password'])

        existing_user = User.query.filter_by(username=username).first()
        if existing_user:
            flash('Username already exists. Please log in or use a different username.',
'danger')
            return redirect(url_for('register'))

        new_user = User(username=username, password=password)
        db.session.add(new_user)
        db.session.commit()
        flash('Registration successful! Please log in.', 'success')
```

```
            return redirect(url_for('login'))

        return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()
        if user and check_password_hash(user.password, password):
            session['user'] = username
            return redirect(url_for('upload'))
        flash('Invalid credentials, please try again.', 'danger')
    return render_template('login.html')

@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('home'))

@app.route('/upload', methods=['GET', 'POST'])
def upload():
    if 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        file = request.files['dataset']
        try:
            df = pd.read_csv(file)

            candidate_features = ['Engine Size(L)', 'Cylinders', 'Fuel Consumption City
(L/100 km)',
```

```
                'Fuel Consumption Hwy (L/100 km)', 'Fuel Consumption Comb
(MPG)']
    available_features = [col for col in candidate_features if col in df.columns]

    if len(available_features) < 4:
        return f"<h3>Insufficient data for training. At least 4 out of 5 features must be
present. Found: {available_features}</h3>"

    if 'CO2 Emissions(g/km)' not in df.columns:
        return "Missing required column: 'CO2 Emissions(g/km)'"

    X = df[available_features]
    y = df['CO2 Emissions(g/km)']

    joblib.dump(available_features, 'features.pkl')

    if os.path.exists(MODEL_PATH):
        os.remove(MODEL_PATH)

    model = RandomForestRegressor()
    model.fit(X, y)
    joblib.dump(model, MODEL_PATH)

    bg_url = url_for('static', filename='up1.jpeg')

    return f"""
    <html>
    <head>
        <style>
            body {{
                margin: 0;
                padding: 0;
                font-family: 'Segoe UI', sans-serif;
```

```
    height: 100vh;
    background: url('{bg_url}') no-repeat center center fixed;
    background-size: cover;
    display: flex;
    align-items: center;
    justify-content: center;
}}
.success-box {{
    background: rgba(255, 255, 255, 0.2);
    backdrop-filter: blur(10px);
    -webkit-backdrop-filter: blur(10px);
    border-radius: 20px;
    padding: 40px;
    text-align: center;
    box-shadow: 0 8px 32px rgba(31, 38, 135, 0.37);
    color: white;
    max-width: 500px;
    animation: fadeIn 1s ease-in-out;
    border: 1px solid rgba(255, 255, 255, 0.18);
}}
.success-box h3 {{
    font-size: 22px;
    margin-bottom: 20px;
}}
.success-box a {{
    display: inline-block;
    margin-top: 15px;
    padding: 10px 20px;
    background: linear-gradient(to right, #a8e063, #56ab2f);
    color: white;
    text-decoration: none;
    border-radius: 8px;
```

```
                font-weight: bold;
                transition: background 0.3s ease;
            }}
            .success-box a:hover {{
                background: linear-gradient(to right, #56ab2f, #a8e063);
            }}
            @keyframes fadeIn {{
                from {{ opacity: 0; transform: translateY(-10px); }}
                to {{ opacity: 1; transform: translateY(0); }}
            }}
        </style>
    </head>
    <body>
        <div class="success-box">
            <h3>☑ Model trained successfully with features:<br>{',
'.join(available_features)}</h3>
            <a href="/predict">Go to Prediction Page</a>
        </div>
    </body>
    </html>
    """

    except Exception as e:
        return f"<h4>Error processing file:</h4><p>{str(e)}</p>"

  return render_template("upload.html")

@app.route('/predict', methods=['GET', 'POST'])
def predict():
  if 'user' not in session:
    return redirect(url_for('login'))
  if not os.path.exists(MODEL_PATH):
```

```
        return "Model not trained yet."

    prediction = None
    status = None
    error = None  # To store validation errors

    # Display variables
    make = model_name = transmission = vehicle_class = fuel_type = None
    fuel_consumption_city = fuel_consumption_comb = fuel_consumption_hwy = None
    fuel_consumption_comb_mpg = None
    cylinders = engine_size = None

    if request.method == 'POST':
        try:
            # Convert form inputs
            engine_size = float(request.form['engine_size'])
            cylinders = int(request.form['cylinders'])
            fuel_consumption_city = float(request.form['fuel_consumption_city'])
            fuel_consumption_hwy = float(request.form['fuel_consumption_hwy'])
            fuel_consumption_comb_mpg =
float(request.form['fuel_consumption_comb_mpg'])

            # Validate thresholds
            if not (0.5 <= engine_size <= 10):
                error = "Engine Size must be between 0.5 and 10 liters."
            elif not (2 <= cylinders <= 16):
                error = "Cylinders must be between 2 and 16."
            elif not (1 <= fuel_consumption_city <= 30):
                error = "Fuel Consumption (City) must be between 1 and 30 L/100km."
            elif not (1 <= fuel_consumption_hwy <= 25):
                error = "Fuel Consumption (Hwy) must be between 1 and 25 L/100km."
            elif not (5 <= fuel_consumption_comb_mpg <= 150):
                error = "Fuel Consumption (MPG) must be between 5 and 150."
```

```
        if error:
            raise ValueError(error)

        # Load model and features
        model = joblib.load(MODEL_PATH)
        feature_list = joblib.load('features.pkl')

        input_map = {
            'Engine Size(L)': engine_size,
            'Cylinders': cylinders,
            'Fuel Consumption City (L/100 km)': fuel_consumption_city,
            'Fuel Consumption Hwy (L/100 km)': fuel_consumption_hwy,
            'Fuel Consumption Comb (MPG)': fuel_consumption_comb_mpg,
        }

        input_features = [input_map[feat] for feat in feature_list]
        pred = model.predict([input_features])[0]
        prediction = round(pred, 2)
        status = "Best Vehicle" if prediction < 180 else "Not Recommended"

        # Retrieve other form fields
        make = request.form['make']
        transmission = request.form['transmission']
        vehicle_class = request.form['vehicle_class']
        fuel_type = request.form['fuel_type']

    except ValueError as ve:
        error = str(ve)
    except Exception as e:
        error = f"Error: {str(e)}"
```

```
    return render_template('predict.html', prediction=prediction, status=status,
make=make,
                           model=model_name, transmission=transmission,
vehicle_class=vehicle_class,
                           fuel_type=fuel_type, fuel_consumption_city=fuel_consumption_city,
                           fuel_consumption_comb=fuel_consumption_comb,
cylinders=cylinders,
                           fuel_consumption_hwy=fuel_consumption_hwy,
fuel_consumption_comb_mpg=fuel_consumption_comb_mpg,
                           error=error)



if _name_ == '_main_':
    app.run(debug=True, port=8080)
```

5.3 RESULTS

The **Home page** of the CO_2 Emission Rating App, this screen provides users with clear Login and Register options to access the application.

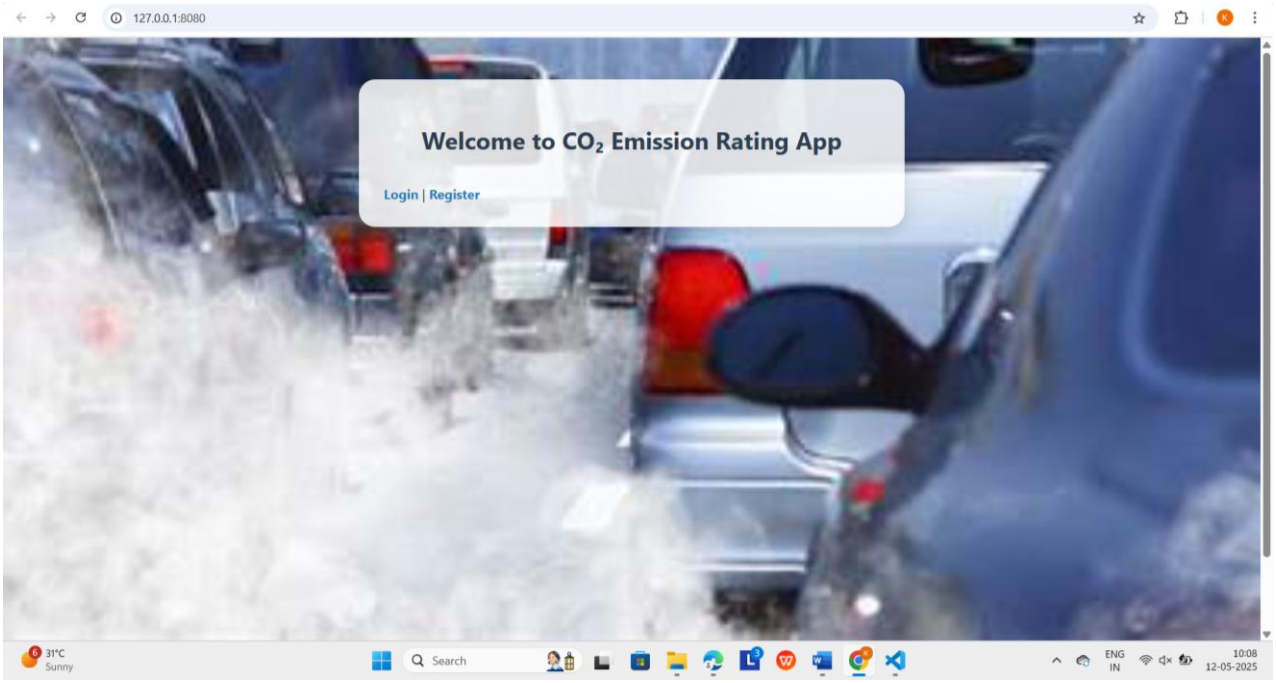

Fig.No 5.3.1 Home Page

The **Register page** allows users to create an account by entering a username and password.

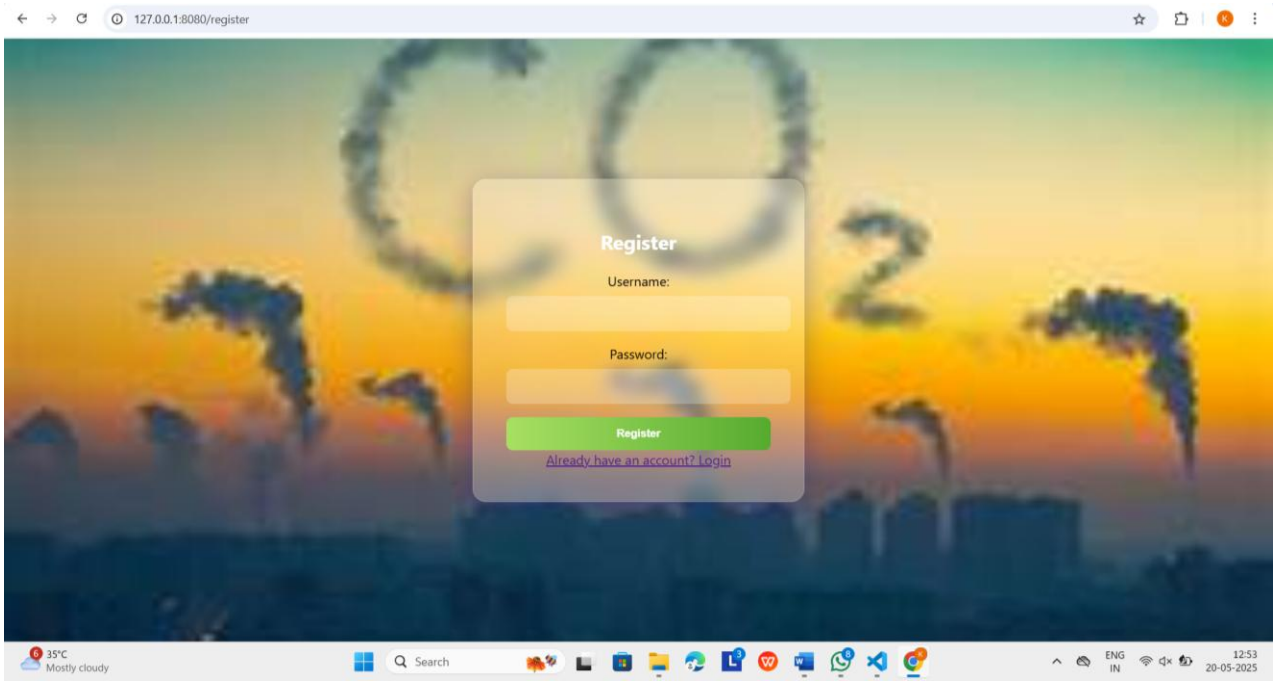

Fig.No 5.3.2 Register Page

In this **Login page** Users are prompted to enter their credentials, and an error message appears for invalid login attempts.

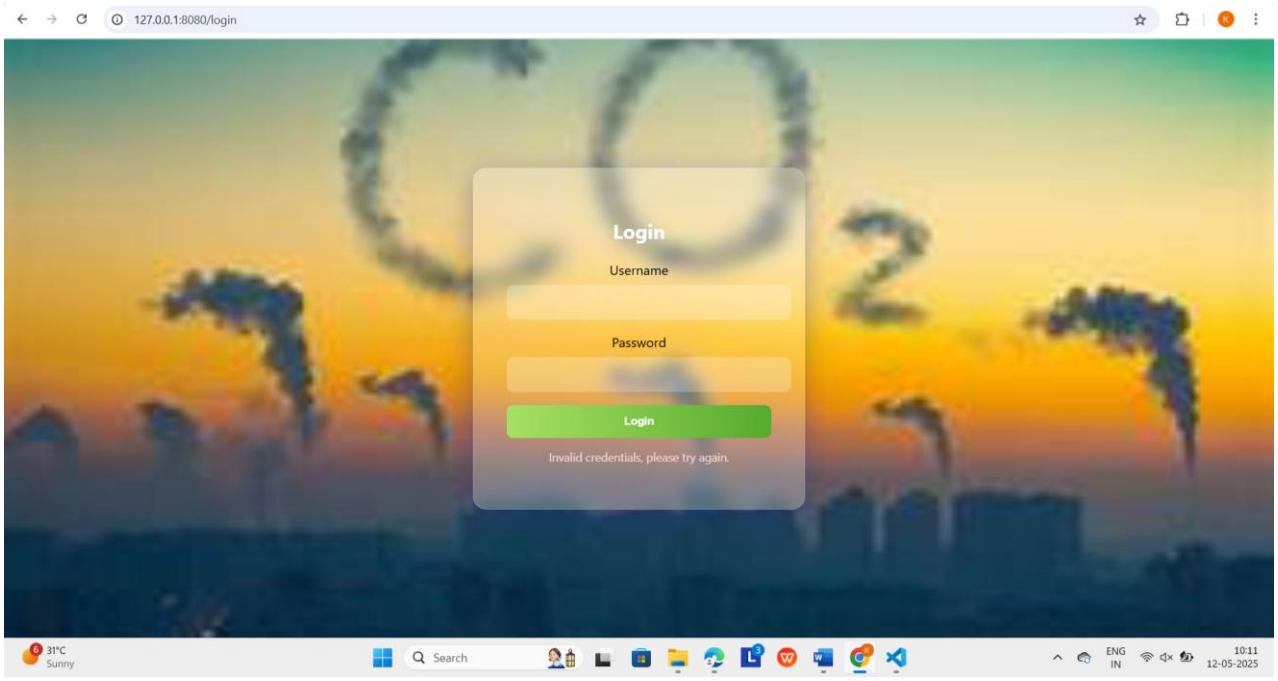

Fig.No 5.3.3 Login Page

The **Upload** Vehicle Data page lets users upload a CSV file and start training the model with a green Upload and Train Model button.

Fig.No 5.3.4 Upload Page

The confirmation screen shows successful model training using features like engine size, cylinders, and fuel consumption, with a green "Go to Prediction Page" button to proceed.

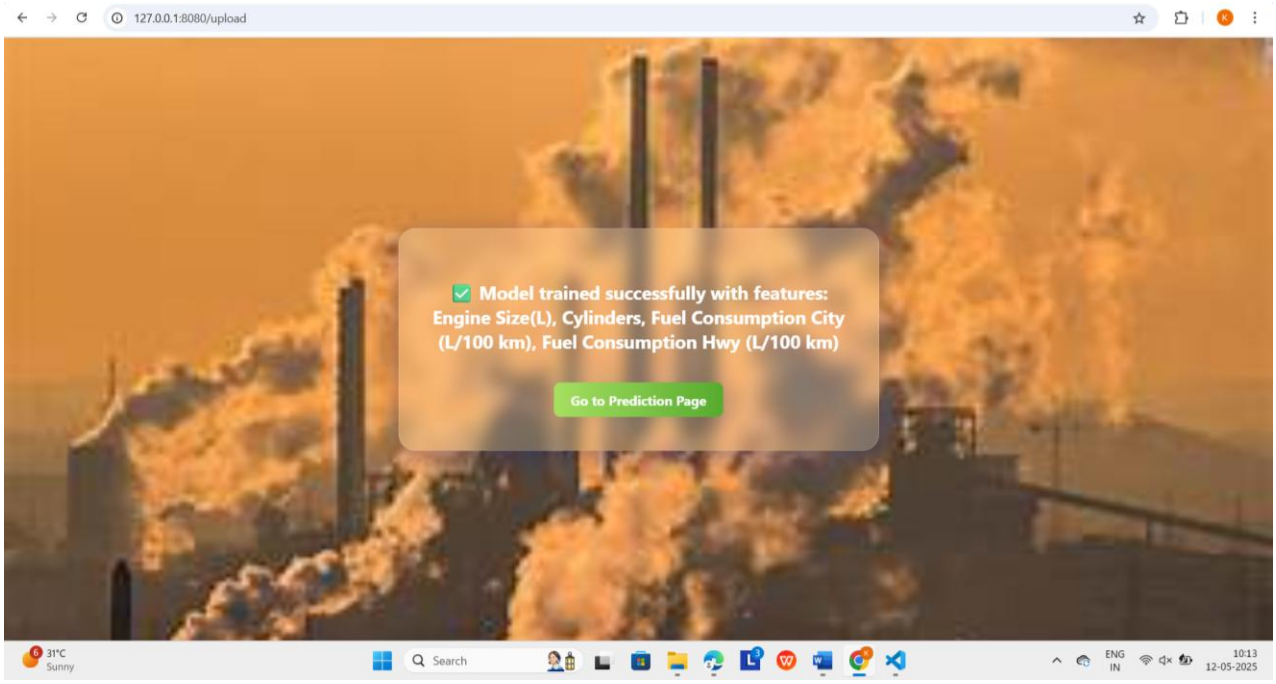

Fig.No 5.3.5 Trained Page

In this **prediction page**, where users input various vehicle parameters such as engine size, fuel consumption, and vehicle class. Upon entering the data, the Predict button estimates the CO_2 emissions based on the trained machine learning model.

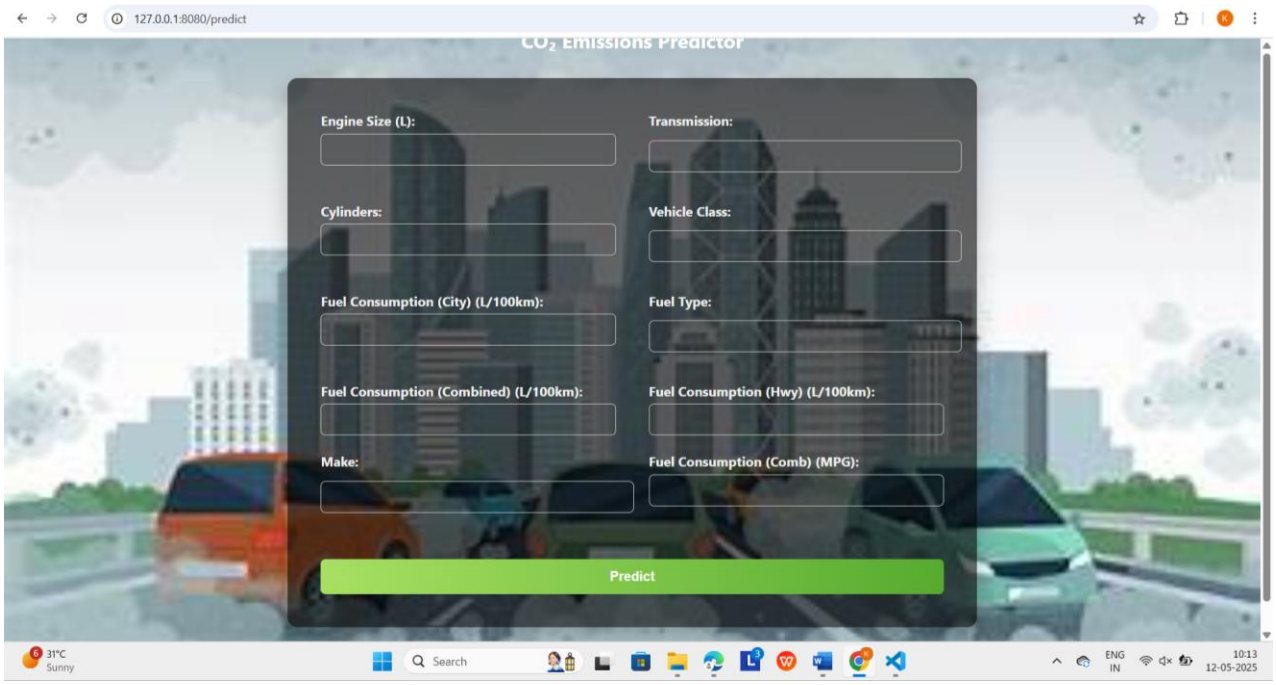

Fig.No. 5.3.6 Predication Page

Finally, it predict vehicles rating as good or average or bad based on the input provided by the user.

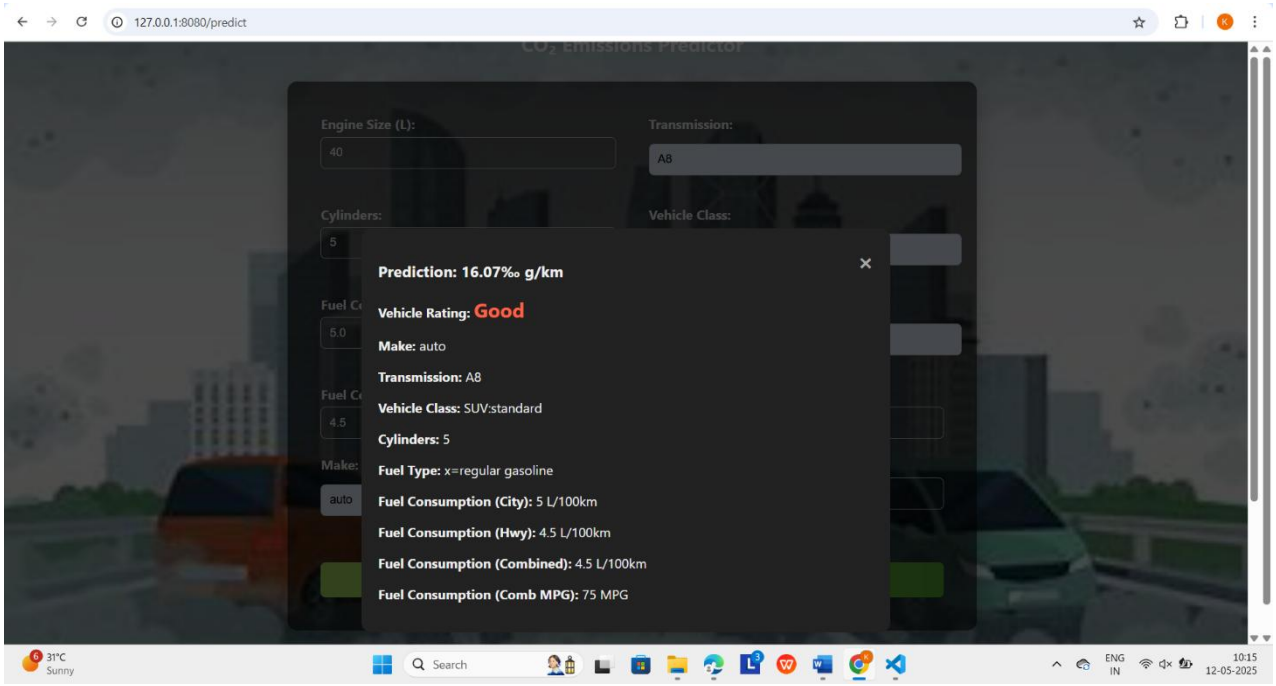

Fig.No 5.3.7 Result Page

5.4 TESTING

5.4.1 TYPES OF TESTING

In the scope of a CO_2 emission rating model for vehicles using a data science approach, testing is critical to ensure accuracy, efficiency, and reliability. **Functional Testing** confirms that the model correctly predicts emissions based on vehicle attributes like engine size, fuel type, and weight, aligning outputs with real-world standards. **Data Validation Testing** ensures the quality and consistency of datasets by checking for missing values, outliers, normalization, and valid ranges. **Model Evaluation Testing** measures model performance using indicators like RMSE, MAE, and R^2, often incorporating cross-validation to minimize overfitting and enhance generalization.
Additional tests include **Usability Testing**, which assesses if environmental analysts, policymakers, or general users can easily interpret CO_2 ratings, making data actionable for non-experts. **Regression Testing** ensures updates to the model or data pipeline don't compromise accuracy, while **Integration Testing** checks the model's interaction with interfaces like APIs and dashboards. **Performance Testing** evaluates speed and scalability with large or real-time data, and **Security Testing** verifies data privacy and system robustness. Finally, **User Acceptance Testing (UAT)** involves real stakeholders to validate the system's practical utility, ensuring the model aids sustainable decision-making effectively.

1.FUNCTIONAL TESTING

Functional testing in a machine learning-based CO_2 emission rating system ensures all components work as intended, from data input to prediction and reporting. It involves verifying emission predictions using real vehicle attributes and ensuring outputs align with emission standards and benchmarks. The system's ability to accept various data formats (CSV, Excel, JSON), handle incomplete or invalid inputs through alerts or imputation, and consistently deliver accurate model responses is assessed. User interface elements are tested for ease of use, informative feedback, and smooth navigation from input to result visualization. Integration with external APIs and datasets is validated for correctness, along with support for localization and accessibility features. Lastly, graphical reports and visualizations are tested for accuracy, clarity, and user comprehension across different formats.

2.USABILITY TESTING

Usability testing of the CO_2 emission rating system ensures it is user-friendly, informative, and accessible to a broad range of users including consumers, automakers, and researchers. It involves verifying the clarity, accuracy, and consistency of displayed emission data, ensuring outputs are understandable and aligned with user expectations. User input handling is tested for responsiveness and intuitive design, with validation messages guiding users effectively. Navigation is assessed to confirm seamless flow through the system, supported by clear menus and tooltips. Integration and accessibility testing ensure third-party data is correctly presented and that the interface supports assistive technologies and multiple languages. Visual design is evaluated for readability, logical color schemes, and layout ergonomics to enhance the overall user experience.

3.REGRESSION TESTING

Regression testing is essential in maintaining the accuracy and consistency of a machine learning-based CO_2 emission rating system amid updates to models, datasets, or interfaces. It ensures that changes—such as retraining models, altering preprocessing steps, or updating the UI—do not negatively impact system performance or prediction quality. The process involves defining a regression test suite for critical features, rerunning validated test cases post-update, and comparing results to identify discrepancies. Automated testing tools like pytest and integration with CI/CD pipelines streamline this process. Control logs and benchmark comparisons help trace and validate changes over time. Overall, regression testing safeguards system reliability while supporting continuous improvement.

4.INTEGRATION TESTING

Integration testing in a machine learning-based CO_2 emission vehicle rating system validates the seamless cooperation between subsystems such as data pipelines, models, APIs, and GUIs, ensuring accurate predictions and a smooth user experience. It involves identifying critical interfaces—like data input, model prediction, and output display—and testing them under real-world scenarios, including bulk data processing and third-party API calls. The process checks data transformation accuracy, response validity, and graceful error handling. Tools like Postman, PyTest, and JMeter aid in testing and automation, while mocking techniques simulate unavailable systems. This testing strengthens system integrity, detects interface issues early, and supports robust CI/CD practices.

5.PERFORMANCE TESTING

Performance testing is essential to evaluate the efficiency, responsiveness, and scalability of a machine learning system designed for vehicle CO_2 emission rating. It measures model inference time, data preprocessing speed, API response time, and resource usage under varying workloads. Key focuses include response time assessment across different dataset sizes, load testing to determine concurrency limits, and stress testing to identify failure points and recovery. Additionally, scalability testing evaluates how performance adapts as data volume grows and the system scales on cloud platforms like AWS, GCP, or Azure. This ensures the system operates smoothly under diverse usage conditions and peak demands.

6.SECURITY TESTING

Security testing is vital for ensuring a WhatsApp chatbot for agriculture schemes protects user data and prevents unauthorized access by identifying vulnerabilities and assessing compliance with data protection standards like GDPR and PCI DSS. It involves evaluating the chatbot's defenses against threats such as injection attacks, XSS, authentication bypass, and data leakage through penetration testing and input validation analysis. The process also tests resilience against attacks like brute force, session hijacking, and man-in-the-middle, ensuring strong access controls and secure communication. Additionally, security testing includes vulnerability scanning and code reviews to detect weaknesses in the chatbot's design and dependencies, enabling early remediation. This comprehensive approach safeguards user confidentiality, integrity, and availability while meeting regulatory requirements.

7.USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a crucial phase in developing a WhatsApp chatbot for agriculture schemes, ensuring it meets the needs and expectations of farmers and stakeholders. UAT involves validating the chatbot's functionality, usability, and user experience by engaging real users to perform tasks like querying scheme details and submitting applications. Test scenarios are defined based on requirements and user stories, with feedback collected on usability, responsiveness, and accuracy. It also verifies compliance with business requirements, regulatory standards, and industry best practices. Additionally, UAT confirms that defects from earlier testing phases have been resolved, ensuring the chatbot is ready for deployment.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The mounting sense of urgency over climate change and environmental conservation has brought into sharp focus the imperative to monitor and curtail carbon dioxide (CO_2) emissions, particularly from transportation, as a leading sector. This project offers a machine learning solution to forecast car CO_2 emissions based on important performance indicators like engine size, the number of cylinders, and fuel consumption in urban and highway driving conditions. Employing Random Forest Regression — a powerful ensemble machine learning algorithm — the system makes precise and understandable predictions. The interface is developed with the Flask web framework, providing functionalities such as user authentication, upload of datasets, model training, and live predictions. It provides an intuitive user experience while accommodating technical and non-technical users. Depending on the emission output, every vehicle is classified as a "Best Vehicle" or "Not Recommended," enabling users to know the ecological implications of their options.

Apart from its primary functionality, the system prioritizes reliability, usability, and data integrity. Random Forest's robustness in capturing intricate feature relationships while avoiding overfitting results in model stability. The app has secure user login and registration with Werkzeug password hashing and has session management for a seamless experience. The system employs SQLAlchemy for efficient and secure database operations. The user interface is made visually appealing and clear, with error messages and meaningful information maximizing usability. Generally, this project effectively incorporates data science and web development to promote environmental awareness and sustainable decision-making. It can be further expanded through the inclusion of real-time vehicle information or advanced models, making it well-suited to global aspirations to fight climate change through technological means.

6.2 FUTURE SCOPE

The future of machine learning-driven CO_2 emission rating systems in transportation is transformative, enhancing environmental sustainability, vehicle design, and regulatory effectiveness. Advances in AI and real-time telemetric data will enable dynamic, accurate emission estimates and personalized eco-driving advice. Lifecycle emission analysis, including manufacturing and disposal, will provide a comprehensive sustainability view, especially for electric and hybrid vehicles. Integration with blockchain and IoT will ensure transparent, real-time monitoring and secure reporting. Global standardization of these systems will streamline regulations and trade, while supporting green innovation and climate policy. Overall, data-driven emission technologies will be vital in shaping sustainable transportation and combating climate change.

REFERENCES

[1]. Sivaraman, A., Krishnan, R., & Menon, S. (2021). Forecasting vehicle CO_2 emissions using supervised machine learning. International Journal of Environmental Data Science, 12(3), 145–152. https://bing.com/search?q=Forecasting+Vehicle+CO%e2%82%82+Emissions+Using+Supervised+Machine+Learning

[2]. Kumar, R., & Singh, A. (2021). A web-based emission assessment platform using gradient boosting regression. Procedia Computer Science, 185, 721–728. https://bing.com/search?q=A+Web-Based+Emission+Assessment+Platform+Using+Gradient+Boosting+Regression

[3]. Patel, N., Mehta, V., & Sharma, R. (2020). Multivariate analysis for CO_2 emission prediction with PCA and SVM. Journal of Machine Learning Applications, 10(4), 220–228. https://bing.com/search?q=Multivariate+Analysis+for+CO%e2%82%82+Emission+Prediction+with+PCA+and+SVM

[4]. Morris, L., Gupta, T., & Wang, X. (2021). Hybrid ensemble learning model for vehicle emission prediction.Environmental Modeling & Software, 138, 105025. https://bing.com/search?q=Hybrid+Ensemble+Learning+Model+for+Vehicle+Emission+Prediction

[5]. Bhargava, P., & Shah, D. (2021). Deep learning-based prediction of automotive CO_2 emissions using feedforward networks. IEEE Access, 9, 114230–114238.

https://bing.com/search?q=Deep+Learning-Based+Prediction+of+Automotive+CO%e2%82%82+Emissions+Using+Feedforward+Networks

[6]. Roy, S., Dutta, B., & Das, A. (2021). Interactive dashboard for CO_2 emission prediction using data science. ACM SIGKDD Explorations Newsletter, 23(1), 17–26. https://bing.com/search?q=Interactive+Dashboard+for+CO%e2%82%82+Emission+Prediction+Using+Data+Science

[7]. Ahmed, Y., Farooq, A., & Said, M. (2021). CO_2 emission estimation with XGBoost on European vehicle data. IEEE Transactions on Intelligent Transportation Systems, 22(9), 5698–5706. https://bing.com/search?q=CO%e2%82%82+Emission+Estimation+with+XGBoost+on+European+Vehicle+Data

[8]. Lee, S., & Park, H. (2021). Predicting emissions from hybrid vehicles using recurrent neural networks. IEEE Transactions on Vehicular Technology, 70(8), 7291–7300. https://bing.com/search?q=Predicting+Emissions+from+Hybrid+Vehicles+Using+Recurrent+Neural+Networks

[9]. Singh, M., & Verma, R. (2022). Classifying vehicle emissions into Euro standards using logistic regression and naive Bayes. International Journal of Transport Science & Technology, 10(1), 87–95 https://bing.com/search?q=Classifying+Vehicle+Emissions+into+Euro+Standards+Using+Logistic+Regression+and+Naive+Bayes

[10]. Zhang, H., Li, X., & Chen, J. (2021). Semi-supervised learning for emission prediction using label propagation. Machine Learning Applications, 6, 100180.https://bing.com/search?q=Semi-Supervised+Learning+for+Emission+Prediction+Using+Label+Propagation

[11]. Yamada, K., Shiba, T., & Harada, Y. (2021). Estimating CO_2 emissions using deep autoencoders for traffic data. IEEE Transactions on Sustainable Computing, 6(4), 743–753. https://www.mdpi.com/2071-1050/15/9/7615

[12]. Gao, J., Sun, L., & Wang, M. (2021). An IoT-based framework for real-time vehicle emission monitoring. IEEE Internet of Things Journal,8(9), 7118–7127. https://ymerdigital.com/uploads/YMER2305G2.pdf