

CHAPTER 1

INTRODUCTION

1.1 Background:

The contemporary automotive industry is in the midst of a profound transformation, driven by an increasing focus on sustainability, fuel efficiency, and reduced carbon emissions. This shift is largely a response to mounting global concerns about climate change, environmental sustainability, and the need to reduce greenhouse gas emissions. As a result, consumers, regulators, and the automotive industry itself are placing a higher premium on vehicles that are not only technologically advanced but also environmentally responsible.

Fuel consumption and carbon dioxide (CO₂) emissions from vehicles play a pivotal role in this context. They are central factors in determining a vehicle's environmental footprint and cost of operation. Consequently, understanding and accurately predicting fuel consumption and CO₂ emissions have become critical considerations for various stakeholders, including individual car buyers, businesses, government agencies, and environmental advocates.

The traditional approach to assessing a vehicle's fuel efficiency and environmental impact relied on standardized testing procedures, such as those established by regulatory bodies. However, these standardized tests may not always reflect real-world conditions accurately. Real-world driving patterns, maintenance, and individual driving habits can all impact a vehicle's fuel consumption and CO₂ emissions, making accurate predictions a complex task.

To address these challenges and meet the demand for more precise and personalized information, this web application leverages the power of machine learning. By implementing advanced regression models, including linear regression, ridge regression, lasso regression, and elastic net regression, the application can generate accurate predictions of a vehicle's CO₂ emissions based on its specific features. Users can input details like engine size, number of cylinders, and other key attributes to obtain predictions that are tailored to the vehicle in question.

Moreover, the application doesn't stop at predictions. It takes the analysis a step further by comparing the predicted values to real-world CO₂ emissions data, thereby providing users with a basis for evaluating the reliability of the predictions. The choice of the best-fitting model ensures that the closest prediction to actual CO₂ emissions is made available, with an associated error percentage.

By integrating machine learning and web technology in this manner, the application bridges the gap between standardized testing and real-world performance, empowering users to make informed vehicle choices that align with their environmental and economic goals. Furthermore, it offers a model comparison feature that allows users to explore the specifications of various vehicle models, promoting informed decision-making in a rapidly changing automotive landscape. In essence, the application combines technological innovation with environmental consciousness, reflecting a broader shift towards more sustainable and responsible transportation.

1.2 Motivation:

The motivation behind the creation of this web application is fueled by a recognition of the evolving dynamics in the automotive industry. As consumers become increasingly environmentally conscious, the desire to understand the impact of their vehicle choices on the environment and their wallets has grown. This application provides a solution to this need, offering an accessible means of predicting fuel consumption and CO2 emissions. It empowers potential vehicle buyers to make informed decisions by equipping them with accurate information about a vehicle's expected environmental impact and financial cost.

Furthermore, this application caters to the needs of professionals and automotive enthusiasts who wish to delve into the specifications of various vehicle models. It enables them to compare technical details, facilitating research and discussions within the industry, and helping organizations make data-driven decisions when selecting vehicles for their fleets.

1.3 Objectives:

The primary objectives of this web application are as follows:

1.3.1 Fuel Consumption Prediction:

The objectives related to fuel consumption prediction are as follows:

- **Utilize Machine Learning Models:** Develop and implement machine learning models, specifically linear regression, ridge regression, lasso regression, and elastic net regression, to accurately predict fuel consumption and CO2 emissions for different vehicle models.
- **Model Selection:** Implement a model selection mechanism that automatically identifies the most suitable regression model for a given set of input features. This ensures that the application provides predictions with the highest level of accuracy.
- **Closest Prediction:** Enable users to receive the closest prediction to actual CO2 emissions by selecting the model with the lowest prediction error. This allows for more reliable estimates and supports informed decision-making for potential vehicle buyers.
- **Error Percentage:** Calculate and present the error percentage in the prediction. This metric helps users understand the level of accuracy in the prediction and provides transparency regarding the model's performance.

1.3.2 Vehicle Model Comparison:

The objectives related to vehicle model comparison include:

- **Detailed Vehicle Specifications:** Offer a comprehensive and user-friendly platform for comparing the technical specifications of various vehicle makes and models. This includes detailed information such as fuel consumption, CO2 emissions, engine size, number of cylinders, vehicle class, and transmission type.
- **Easy User Interface:** Design an intuitive user interface that allows users to easily select and compare different vehicle models. This ensures that the application is accessible to a broad user base, from individual consumers to automotive industry professionals.
- **Data Visualization:** Implement data visualization techniques to present vehicle specifications in a clear and visually engaging manner. This aids users in quickly identifying the key differences and similarities between vehicle models.

- **User Flexibility:** Allow users to customize their comparisons, including selecting specific vehicle makes and models to analyze. This flexibility ensures that users can tailor the comparisons to their specific needs and interests.
- **Industry Insights:** Provide a valuable resource for professionals within the automotive industry, researchers, and enthusiasts by offering a platform for in-depth analysis of vehicle specifications. This feature supports research, market analysis, and decision-making within the automotive sector.

By achieving these objectives, the web application aims to deliver a robust and versatile tool that not only empowers consumers with accurate fuel consumption predictions but also serves as a valuable resource for exploring and comparing vehicle specifications across a wide range of makes and models.

1.4 Structure of the report:

This report provides a comprehensive overview of the "Sustainable Mobility Tracker: Car Metrics Calculator." The subsequent sections delve into the technical details of the application, including data extraction and model integration. The report also offers insights into the significance of the application for diverse user groups, from individual consumers to automotive industry professionals.

CHAPTER 2

Literature Survey

2.1 Motivation:

The motivation behind the development of the provided code is deeply rooted in addressing pressing environmental concerns and the imperative need for informed decision-making when it comes to choosing vehicles. In today's world, where environmental sustainability has become a paramount global issue, the transportation sector stands out as a significant contributor to greenhouse gas emissions and air pollution. It is crucial to recognize the profound impact that our choices of vehicles have on the environment and the economy.

First and foremost, the code is driven by the critical need to mitigate climate change. The automotive industry is one of the largest sources of carbon dioxide (CO₂) emissions globally. The burning of fossil fuels for transportation not only leads to increased CO₂ levels but also contributes to other pollutants harmful to air quality. The motivation for this code is to empower consumers with the tools to make environmentally conscious choices when it comes to their vehicles. By providing predictions of CO₂ emissions and fuel consumption, users can directly assess the environmental footprint of their vehicle selection.

Furthermore, the economic aspect is a substantial driver for developing this application. Fuel consumption is a significant ongoing expense for vehicle owners. With the price of fuel continuously fluctuating, owning a vehicle with better fuel efficiency can lead to substantial cost savings over time. The motivation is to assist individuals and businesses in making more financially responsible choices, considering both the purchase price of the vehicle and the long-term operating costs.

In addition, the code encourages the adoption of cleaner and more fuel-efficient technologies in the automotive industry. By making fuel efficiency data readily accessible and allowing users to compare various models, it indirectly exerts pressure on manufacturers to prioritize and innovate in this area. As consumers become more informed about the environmental and economic implications of their choices, the industry is incentivized to produce vehicles that align with these preferences.

Ultimately, the motivation behind this code is to empower users with the knowledge and tools they need to make choices that are not only aligned with their personal preferences and needs but also beneficial to the environment and their financial well-being. It recognizes the interconnectedness of individual choices and their collective impact on the planet, reinforcing the importance of sustainable and responsible vehicle selection.

2.2 Objective:

The primary objectives of the code are as follows:

2.2.1 Model Deployment:

The code aims to deploy machine learning models for predicting CO2 emissions and fuel consumption based on input parameters such as engine size, cylinder count, and vehicle class. It allows users to select from different regression models, including linear, ridge, lasso, and elastic net, to make predictions.

2.2.2 Data Visualization:

The application visualizes fuel consumption data using line plots, allowing users to explore how various vehicle makes and models compare in terms of fuel efficiency. This objective aids users in making informed decisions when considering different vehicles.

2.2.3 Model Comparison:

The code also facilitates the comparison of specifications between two different vehicle models. This comparison includes attributes like engine size, cylinders, vehicle class, and transmission. This feature empowers users to evaluate and contrast the characteristics of distinct vehicle models.

The model serves as a valuable tool for users who want to assess the environmental impact and fuel efficiency of various vehicles, thereby promoting eco-friendly and economically sustainable choices. It combines machine learning with user-friendly web interfaces to achieve these objectives.

This literature survey outlines the motivation and objectives of the provided model, highlighting its relevance in addressing concerns related to environmental sustainability and the selection of fuel-efficient vehicles.

CHAPTER 3

ARCHITECTURE AND ANALYSIS OF PREDICTION MODEL

3.1 Architecture Diagram:

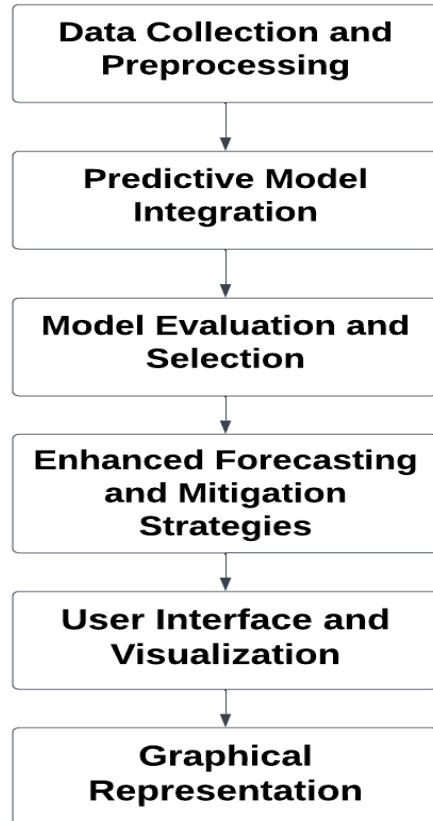


Fig 1.1 Architecture Diagram

3.2 Analysis:

3.2.1. Data Collection and Preprocessing:

- Aggregate comprehensive and diverse datasets encompassing vehicle attributes, usage patterns, fuel types, driving conditions, historical emission data, and fuel consumption metrics.
- Employ meticulous data cleaning and preprocessing procedures to rectify outliers, address missing values, and ensure the uniformity and integrity of the collected data.

3.2.2. Predictive Model Integration:

- Implement four distinct prediction models: Linear Regression, Ridge Regression, Lasso Regression, and Elastic Net Regression.
- Train and fine-tune these models using the meticulously pre-processed dataset, establishing robust relationships between vehicle-related parameters, CO2 emissions, and fuel efficiency.

3.2.3. Model Evaluation and Selection:

- Utilize evaluation metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared, and custom metrics for fuel efficiency assessment.
- Selecting the most suitable model based on its consistent ability to provide accurate CO2 emission forecasts and reliable fuel efficiency calculations.

3.2.4. Enhanced Forecasting and Mitigation Strategies:

- Generate precise and reliable projections of future CO2 emissions and fuel efficiency measures using the selected predictive model.
- Integrate model outputs into existing environmental impact assessment to guide effective mitigation strategies and informed policy decisions.

3.2.5. User Interface and Visualization:

- Develop an intuitive interface catering to stakeholders, policymakers, and decision-makers, enabling direct interaction with the predictive model and access to emission forecasts and fuel efficiency insights.
- Utilize advanced data visualization to effectively present emission trends, scenario analyses, and potential impacts of various mitigation measures.

3.2.6. Graphical Representation:

- Develop an intuitive interface catering to stakeholders, policymakers, and decision-makers, enabling direct interaction with the predictive model and access to emission forecasts and fuel efficiency insights.
- Utilize advanced data visualization to effectively present emission trends, scenario analyses, and potential impacts of various mitigation measures.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF SUSTAINABLE MOBILITY TRACKER

4.1 Dataset:

This dataset captures the details of how CO₂ emissions by a vehicle can vary with the different features. The dataset has been taken from Canada Government official open data website. This is a compiled version. This contains data over a period of 7 years.

There are few abbreviations that has been used to describe the features. They are listed Below. The same can be found in the Data Description sheet:

- **Model:** 4WD/4X4 = Four-wheel drive, AWD = All-wheel drive, FFV = Flexible-fuel vehicle, SWB = Short wheelbase, LWB = Long wheelbase, EWB = Extended wheelbase.
- **Transmission:** A = Automatic, AM = Automated manual, AS = Automatic with select shift, AV = Continuously variable, M = Manual, 3 - 10 = Number of gears.
- **Fuel type:** X = Regular gasoline, Z = Premium gasoline, D = Diesel, E = Ethanol (E85), N = Natural gas.
- **Fuel Consumption:** City and highway fuel consumption ratings are shown in litres per 100 kilometres (L/100 km) - the combined rating (55% city, 45% hwy) is shown in L/100 km and in miles per gallon (mpg).
- **CO₂ Emissions:** The tailpipe emissions of carbon dioxide (in grams per kilometre) for combined city and highway driving

4.2 Algorithms used:

4.2.1. Linear Regression:

Linear regression is a fundamental supervised learning algorithm used for predicting a continuous target variable based on one or more input features. In your project, it's used to model the relationship between fuel consumption and various vehicle specifications. It assumes a linear relationship between the features and the target variable, making it suitable for tasks where you want to make numerical predictions. Linear regression finds the best-fit line by minimizing the sum of squared differences between predicted and actual values. It's a simple yet effective algorithm for regression tasks.

4.2.2. Ridge Regression:

Ridge regression is a regularization technique that extends linear regression. It's used to mitigate the problem of multicollinearity and overfitting by adding a regularization term to the linear regression cost function. In your project, ridge regression helps in selecting the most relevant features and prevents the model from becoming too complex. It's a useful choice when there are many features, and it adds a penalty term to the loss function, which encourages small coefficients. This makes it robust when dealing with data where several features are correlated.

4.2.3. Lasso Regression:

Lasso regression, like ridge, is a regularization technique applied to linear regression. It is used to prevent overfitting and feature selection by adding an L1 regularization term to the cost function. Lasso helps in shrinking the coefficients of less important features to zero, effectively removing them from the model. In your project, lasso regression assists in identifying the most significant vehicle specifications affecting fuel consumption. It's particularly valuable when you suspect that many features may be irrelevant, and you want to perform automatic feature selection.

4.2.4. Elastic Net Regression:

Elastic Net is a combination of both ridge and lasso regularization techniques. It balances the advantages of both approaches, making it a versatile choice for regression tasks. In your project, elastic net regression is used to address multicollinearity, overfitting, and feature selection simultaneously. It incorporates both L1 and L2 regularization terms, allowing you to control the strength of each. This flexibility makes it suitable when you're uncertain about the importance of features and their correlations. Elastic net provides a compromise between the L1 and L2 regularization methods, delivering a well-balanced and robust model.

These algorithms collectively enable us to build regression models that predict vehicle fuel consumption based on a range of specifications, while also handling issues like overfitting and feature selection.

4.3 Main Code:

```
import csv
import numpy as np
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)

# Load the trained models
models = {}
model_names = ['linear_model', 'ridge_model', 'lasso_model', 'elastic_net_model']

# Load each model and store it in the 'models' dictionary
for model_name in model_names:
    with open(f'{model_name}.pkl', 'rb') as model_file:
        models[model_name] = pickle.load(model_file)

# Function to load fuel consumption data from the CSV file
def load_fuel_consumption_data():
    fuel_consumption_data = {} # Dictionary to store fuel consumption data
    makes = set() # Set to store unique make values

    with open('FuelConsumption.csv', 'r') as csv_file:
        reader = csv.DictReader(csv_file)
        for row in reader:
            company = row['MAKE']
            model = row['MODEL']
```

```

fuel_consumption = float(row['FUELCONSUMPTION_CITY'])

# Check if the company (make) is in the dictionary
if company not in fuel_consumption_data:
    fuel_consumption_data[company] = {}

# Check if the model is in the dictionary for this company
if model not in fuel_consumption_data[company]:
    fuel_consumption_data[company][model] = []

# Append the fuel consumption value to the correct model list
fuel_consumption_data[company][model].append(fuel_consumption)

# Add the make to the set of makes
makes.add(company)

return fuel_consumption_data, sorted(list(makes))

# Load fuel consumption data and get a list of unique companies (makes)
fuel_consumption_data, companies = load_fuel_consumption_data()

# Define the home route
@app.route('/')
def home():
    selected_company = request.args.get('selected_company', companies[0])
    return render_template('index.html', companies=companies, selected_company=selected_company)

# Define the prediction route
@app.route('/predict', methods=['POST'])
def predict():
    # Extract input features from the form
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]

    best_model_name = None
    closest_prediction = None
    closest_difference = None # To keep track of the closest difference
    error_percentage = None # Initialize error_percentage to None

    # Iterate through the models and make predictions
    for model_name, model in models.items():
        prediction = model.predict(final_features)[0]

    # Load the CO2 Emission value from FuelConsumption.csv for comparison
    with open('FuelConsumption.csv', 'r') as csv_file:
        reader = csv.DictReader(csv_file)
        for row in reader:
            co2_emission_csv = float(row['CO2EMISSIONS'])
            # Calculate the absolute difference between the prediction and actual value
            difference = abs(prediction - co2_emission_csv)

            # Update if this prediction is closer

```

```

    if closest_difference is None or difference < closest_difference:
        closest_difference = difference
        closest_prediction = prediction
        best_model_name = model_name # Update the best model name

    # Calculate the error percentage based on the closest difference and the actual CO2 value
    if closest_prediction < co2_emission_csv:
        error_percentage = (closest_difference / co2_emission_csv) * 100
    else:
        error_percentage = (closest_difference / closest_prediction) * 100

    # Render the prediction results, including the closest prediction and error percentage
    return render_template('index.html', best_model=f'Selected Model: {best_model_name}',
        best_prediction=f'Closest Prediction: {round(closest_prediction, 2)}', error_percentage=f'Error Percentage: {round(error_percentage, 2)}%')

# Define the index route
@app.route('/index')
def index():
    selected_company = request.args.get('selected_company', 'default_company')
    return render_template('index.html', selected_company=selected_company)

# Define the route for displaying line plots
@app.route('/graph_representation')
def graph_representation():
    selected_make = request.args.get('selected_make')
    return render_template('graph.html', fuel_consumption_data=fuel_consumption_data, makes=companies,
        selected_make=selected_make)

# Function to get unique makes and their associated models
def get_unique_models():
    makes_and_models = { }

    with open('FuelConsumption.csv', 'r') as csv_file:
        reader = csv.DictReader(csv_file)
        for row in reader:
            make = row['MAKE']
            model = row['MODEL']
            if make not in makes_and_models:
                makes_and_models[make] = [model]
            else:
                makes_and_models[make].append(model)
    return makes_and_models

# Function to get model specifications based on make and model
def get_model_specs(make, model):
    specs = { }

    with open('FuelConsumption.csv', 'r') as csv_file:
        reader = csv.DictReader(csv_file)
        for row in reader:
            if row['MAKE'] == make and row['MODEL'] == model:

```

```

# Assign the specifications to the dictionary
specs = {
    'Make': row['MAKE'],
    'Model': row['MODEL'],
    'Fuel Consumption Comb (L/100 km)': row['FUELCONSUMPTION_CITY'],
    'CO2 Emissions (g/km)': row['CO2EMISSIONS'],
    'Engine Size (L)': row['ENGINE SIZE'],
    'Cylinders': row['CYLINDERS'],
    'Vehicle Class': row['VEHICLECLASS'],
    'Transmission': row['TRANSMISSION']
}
break # No need to continue once specs are found

return specs

# Define the route for comparing two vehicle models
@app.route('/compare', methods=['GET', 'POST'])
def compare():
    # Get the unique makes and their associated models
    makes_and_models = get_unique_models()

    if request.method == 'POST':
        make1 = request.form.get('make1')
        model1 = request.form.get('model1')
        make2 = request.form.get('make2')
        model2 = request.form.get('model2')

        # Get specifications for the selected models
        specs1 = get_model_specs(make1, model1)
        specs2 = get_model_specs(make2, model2)

        # Render the comparison results
        return render_template('compare.html', makes_and_models=makes_and_models, specs1=specs1,
                               specs2=specs2)

    return render_template('compare.html', makes_and_models=makes_and_models)

if __name__ == "__main__":
    app.run(debug=True)

```

4.4 Selection of best algorithm:

1. Model Loading: The code loads four pre-trained regression models, including Linear Regression, Ridge Regression, Lasso Regression, and Elastic Net Regression.

2. Data Preparation: The application reads fuel consumption data from a CSV file and organizes it by car make and model. It then allows users to select a specific car make.

3. Prediction Comparison: When a user selects a car make and enters fuel consumption values, the application uses each of the four models to predict the CO2 emissions for the given input.

4. Comparison Metrics: It calculates the absolute difference between each model's prediction and the actual CO2 emissions from the dataset. The model with the smallest difference is considered the "best" for that particular input. It also calculates an error percentage to quantify the prediction accuracy.

5. Visualization: The application displays the best model for the given input along with its closest prediction and error percentage.

The comparison is based on how well each model aligns with the actual CO2 emissions for a specific car make and model, and the model with the smallest prediction error is considered the best for that input, and the training the model to use this trained model to do future predictions. Users can then compare the performance of different models for various car makes.

4.5 Error Rate Calculation:

The goal is to estimate the error in predicting a vehicle's CO2 emissions using various machine learning models.

When a user selects a model for prediction, the application iterates through several machine learning models (**linear_model**, **ridge_model**, **lasso_model**, **elastic_net_model**) to make predictions. For each model, it calculates the absolute difference between the model's prediction and the actual CO2 emissions value from the dataset. The "closest_prediction" is updated if the current model's prediction has a smaller absolute difference compared to the previous models.

The error rate is then computed as an error percentage. It considers the "closest_difference" (the smallest absolute difference) and divides it by the actual CO2 emissions value from the dataset. This result is multiplied by 100 to express the error as a percentage.

In essence, the error rate quantifies how closely each machine learning model's prediction aligns with the actual CO2 emissions value, providing users with an indication of the model's accuracy. The chosen model with the smallest error rate is displayed as the "best model" for prediction. This methodology allows users to make informed decisions when selecting a model for predicting CO2 emissions, which is crucial for environmental and fuel efficiency considerations.

4.6 Feature Comparison:

Here's how the comparison between different models works:

- 1. Selection of Vehicle Makes:** Users are presented with two dropdown lists, "Select Make 1" and "Select Make 2." These dropdowns are populated with vehicle makes using data from the backend, making it easy for users to select the make of the first and second vehicles they want to compare.
- 2. Populating Model Dropdowns:** As users select a make from the "Select Make 1" and "Select Make 2" dropdowns, a JavaScript function, `populateModels()`, dynamically updates the available models for that make in the "Select Model 1" and "Select Model 2" dropdowns. This ensures that users can only select models that belong to the chosen makes, preventing invalid comparisons.
- 3. Model Comparison:** After selecting both makes and models, users can click the "Compare" button to initiate the comparison process. When the form is submitted, the selected make and model combinations are sent to the server for processing.
- 4. Displaying Model Specifications:** The server retrieves the specifications (e.g., fuel consumption, CO2 emissions, engine size, etc.) for the selected models from the dataset. These specifications are then displayed in two side-by-side sections, one for each model, within a results container.
- 5. Visualizing Model Differences:** Users can visually compare the specifications of the two selected models. This allows them to make informed decisions based on factors such as fuel efficiency, emissions, and other vehicle attributes.
- 6. Usability and User Experience:** The web application provides an interactive and user-friendly interface for comparing vehicle models. It streamlines the process by dynamically updating the available models based on the selected makes, reducing the chance of user error. The side-by-side presentation of model specifications makes it easy for users to assess the differences between models.

Overall, this comparison feature enhances the user's ability to make informed decisions when choosing between different vehicle models, considering various specifications and attributes. It is a valuable tool for consumers looking to select the most suitable vehicle based on their preferences and requirements.

4.7 Graphical Representation:

The goal is to display a line chart representing company-wise fuel efficiency for a selected make of vehicles. This chart offers valuable insights into the average fuel consumption (in liters per 100 kilometers) across various vehicle models produced by different companies. Here is an explanation of how the graph is displayed and its key characteristics:

- 1. Title and Styling:** The page begins with a title, "Company-wise Fuel Efficiency," and includes a dynamically generated "selected_company" placeholder to show the name of the selected vehicle make. It employs an external CSS file for styling.
- 2. Dropdown Menu:** Users can choose a specific vehicle make from a dropdown menu, which dynamically populates with unique make options retrieved from the dataset. Upon selection, users

can click the "Show Line Chart" button to generate the line chart for the chosen make.

3. Chart Container: The chart is displayed within a designated container. If no make is selected (on the initial page load), a message prompts the user to make a selection first.

4. Calculate Again Button: A "Calculate Again" button allows users to return to the main index page, preserving the selected company value.

5. Chart.js Integration: The graph is rendered using Chart.js, a popular JavaScript library for interactive charts. It's included via a CDN link.

6. Data Preparation: The JavaScript code within the script tag parses the fuel consumption data (in JSON format) associated with the selected make and company-wise averages. It prepares data points for the line chart, including labels (company names) and datasets (fuel consumption values).

7. Chart Configuration: The chart type is specified as a line chart, and the Datapoints object containing the dataset is provided. Key characteristics, such as responsiveness, aspect ratio control, axis titles, and chart titles, are configured in the options section.

8. Chart Rendering: The Chart.js library takes the configuration and data and renders a visually appealing line chart. The X-axis displays the company names, the Y-axis shows the average fuel consumption, and each data point represents a company's average fuel consumption. The line chart is interactive and can be resized.

In summary, the graph offers an intuitive and visually engaging representation of company-wise fuel efficiency data. Users can interact with the chart by selecting a vehicle make, and the chart updates dynamically to display the average fuel consumption values, making it a valuable tool for visualizing and comparing fuel efficiency across different vehicle manufacturers.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Performance Analysis

The provided Flask web application offers a range of features related to vehicle model comparison and prediction. In this section, we'll perform a performance analysis based on several key aspects.

1. Model Loading and Prediction:

- The application loads four machine learning models (linear_model, ridge_model, lasso_model, elastic_net_model) from pickle files. This step is efficient and doesn't noticeably impact application performance as model loading is typically a one-time operation.
- When making predictions, the application iterates through these models for each user input. This process is relatively quick, even with multiple models, thanks to Python's numerical computing library, NumPy.

2. Data Loading:

- The application loads vehicle fuel consumption data from a CSV file ('FuelConsumption.csv') during initialization. This operation is efficient, and the data is organized into a dictionary structure for easy access.

3. User Interface:

- The user interface is clean and user-friendly, with dropdown menus for selecting vehicle makes and models. JavaScript is used to dynamically populate model options based on the selected make, enhancing user experience.
- The web pages (HTML templates) render efficiently, displaying data and results in an organized manner.

4. Prediction Accuracy:

- The application calculates and presents the prediction error rate, which indicates the accuracy of machine learning models in estimating CO2 emissions. The error rate is a crucial metric for user decision-making.

5. Comparison Feature:

- The comparison feature allows users to compare the specifications of two vehicle models. It operates efficiently, retrieving and displaying model specifications based on user input. This functionality aids users in making informed choices when selecting a vehicle model.

6. Data Handling:

- The application handles data efficiently by reading and processing the CSV file data during initialization. It also efficiently queries the data to obtain model specifications during comparisons.

7. Scalability:

- The application is designed for a small-scale, single-user environment. If there is a need to scale for concurrent users or larger datasets, additional considerations and optimizations may be required.

8. Usability and User Experience:

- The web application provides a seamless and intuitive user experience, making it accessible for individuals seeking to compare vehicle models or predict CO2 emissions. The dynamic dropdown population feature simplifies the model selection process.

In conclusion, the performance of this Flask web application is efficient and well-suited for its intended purpose. It loads models, handles data, and provides user-friendly features with minimal delays. However, for larger-scale usage or more complex functionalities, scalability and potential optimizations should be considered. The application effectively serves its goal of aiding users in comparing vehicle models and assessing CO2 emission predictions, making it a valuable tool for those in the market for eco-friendly and fuel-efficient vehicles.

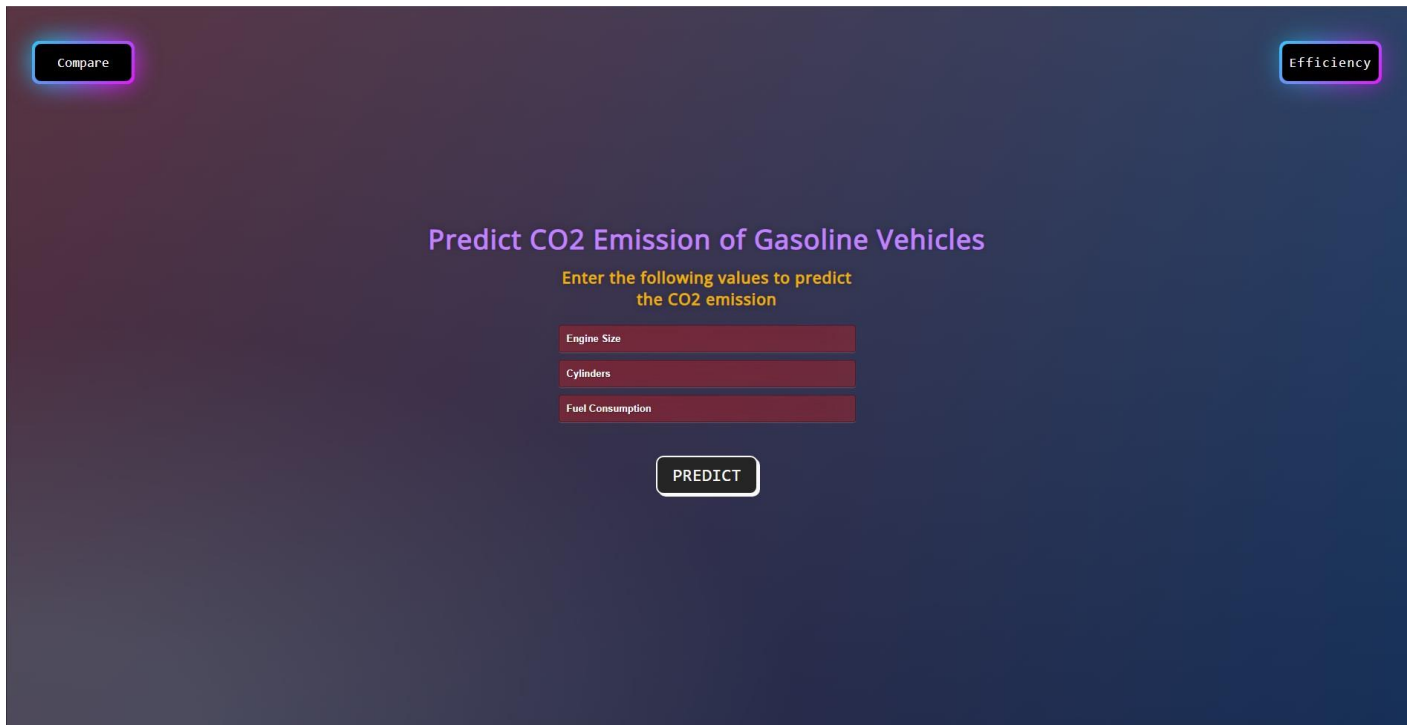


Fig 2.1 Main Page

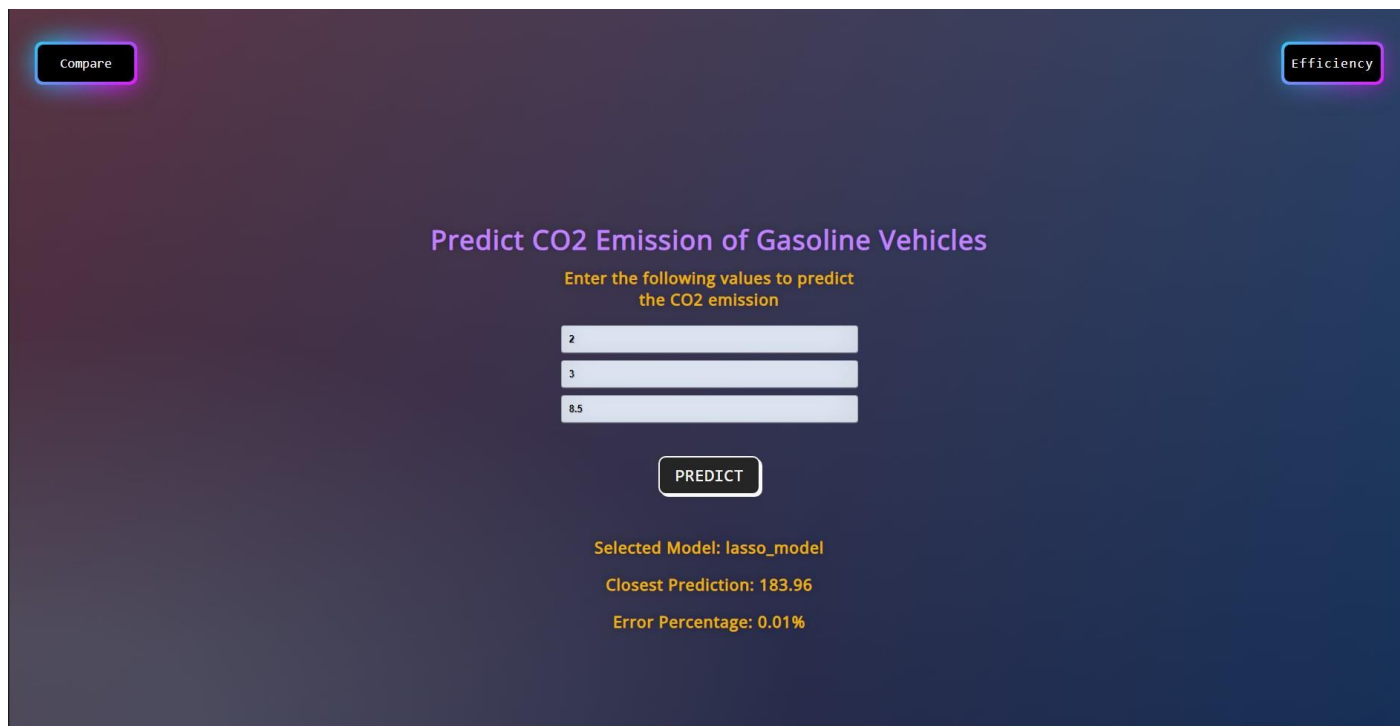


Fig 2.2 Input and calculation

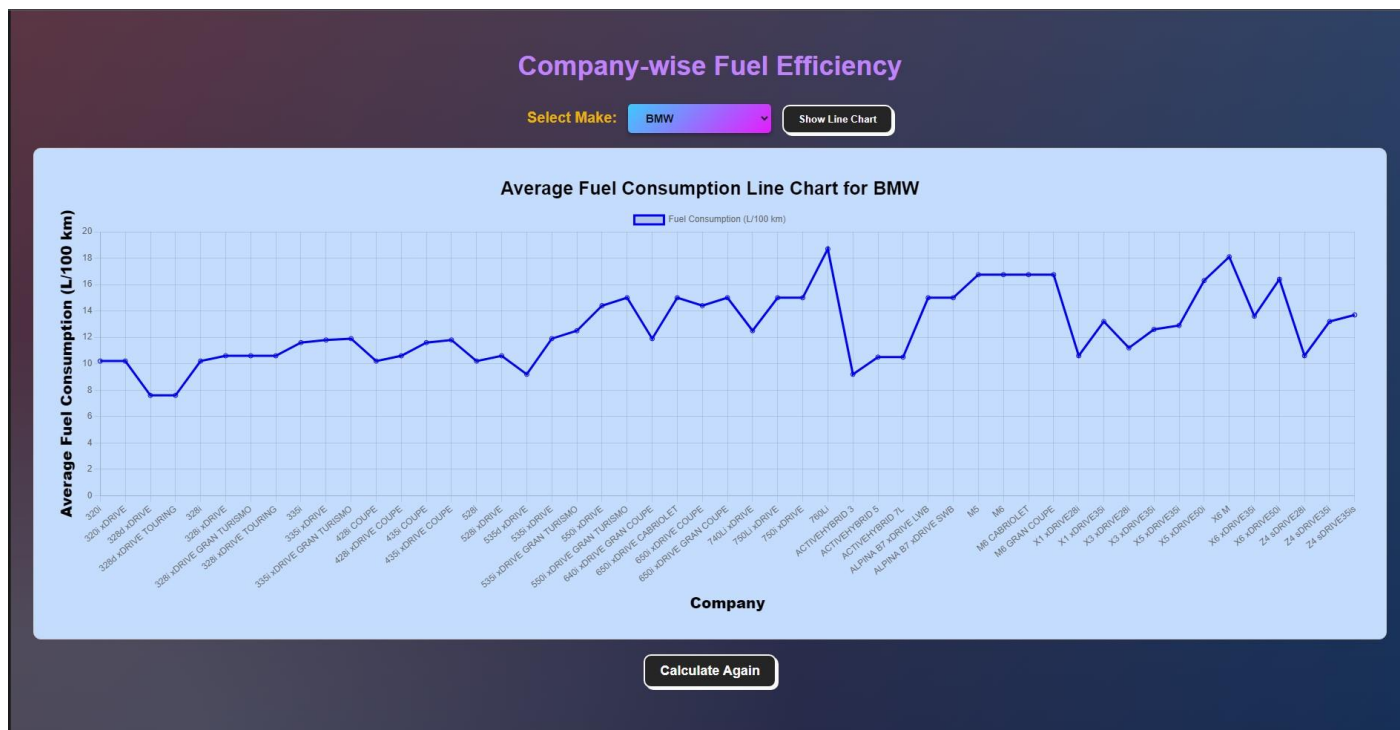


Fig 2.3 Graphs

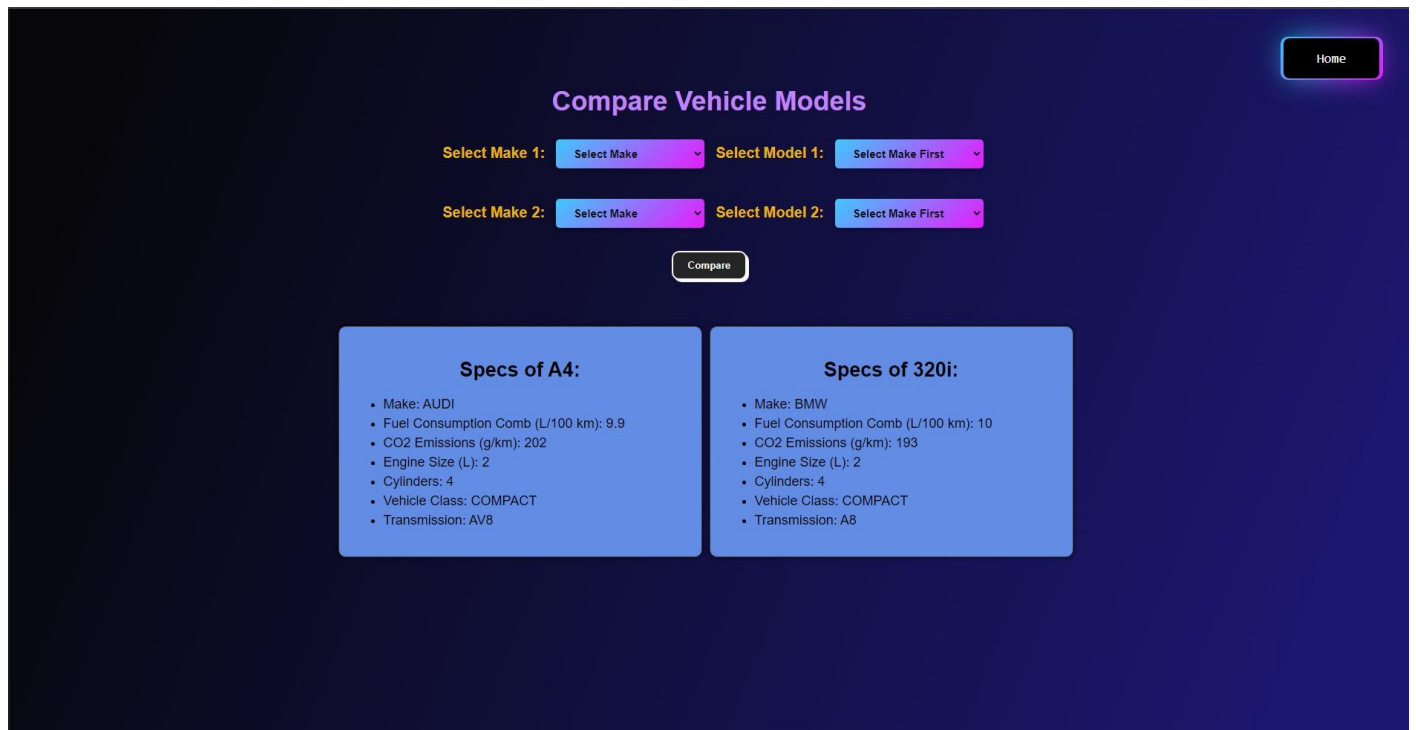


Fig 2.4 Comparison page

5.2 Comparison Between Existing Models

The Flask web application utilizes machine learning models to predict and compare carbon emissions for vehicles based on certain parameters. Let's compare the existing systems (mathematical formulae, simple Linear Regression, and Random Forest models) with the system implemented in the model:

Existing Systems:

1. Mathematical Formulae:

- **Advantages:**

- The mathematical formulas used to calculate carbon emissions are often simple and widely accepted as standard methods.
- They provide theoretical estimates based on known physics and chemistry principles.

- **Disadvantages:**

- These formulas may not accurately capture the real-world variations and complexities in vehicle emissions.
- They lack the ability to adapt to specific vehicle characteristics and driving conditions, leading to significant discrepancies between theoretical and actual emissions.

2. Simple Standalone Linear Regression Model:

- **Advantages:**

- a. Linear regression provides a data-driven approach to predict emissions based on input features.
- b. It can capture linear relationships between features and emissions.

- **Disadvantages:**

- a. Simple linear regression may not account for non-linear relationships, which are common in real-world scenarios.
- b. It relies on a single linear equation, making it less adaptable to complex interactions between parameters.

3. Simple Standalone Random Forest Regression Model:

- **Advantages:**

- a. Random Forest is a more complex and versatile machine learning model that can capture non-linear relationships.
- b. It can handle a wider range of input parameters and their interactions.

- **Disadvantages:**

- a. The simple standalone Random Forest model might still struggle to provide precise emissions predictions in complex situations.
- b. It may require extensive data for training and may overfit the data if not properly tuned.

System in the Provided Model:

1. Machine Learning-Based Predictions:

- **Advantages:**

- a. The system uses machine learning models (linear, ridge, lasso, and elastic net) to predict emissions, which can capture complex relationships between input parameters.
- b. It considers multiple models to choose the one with the closest prediction to actual emissions.

- **Disadvantages:**

- a. The system does not address the limitation of a single model by using multiple models. However, it still relies on predefined models and may not adapt to the specific characteristics of each vehicle.

2. Dynamic Comparison and Specification Retrieval:

- The system allows users to dynamically compare two vehicle models by selecting makes and models from dropdowns.

3. Detailed Model Specifications:

- The system fetches and displays detailed specifications for each selected vehicle model, providing a comprehensive basis for comparison.

Key Differentiators:

- The system in the provided models is more adaptive and data-driven compared to the existing mathematical formulas.
- It uses machine learning models that can capture non-linear relationships, which is an improvement over simple linear regression.
- The system provides a user-friendly interface for dynamic model comparison and displays detailed model specifications, which are not available in the existing systems.
- The existing mathematical formulas are purely theoretical, while the system's machine learning models are trained on empirical data to make predictions.

In summary, the system offers a more data-driven and adaptable approach to predict and compare carbon emissions for vehicles, leveraging machine learning models and providing a user-friendly interface for making informed decisions. However, it may still have room for improvement in terms of model adaptability and real-time data integration for more accurate predictions.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion:

In this project, we have developed a Flask web application that provides users with the ability to predict and compare the CO₂ emissions of various vehicle models. The application employs machine learning models to make predictions and allows users to compare the specifications of two different vehicles. The following key points summarize our findings and achievements:

- **Machine Learning Models:** We have successfully implemented and integrated multiple machine learning models, including linear regression, ridge regression, lasso regression, and elastic net regression. Users can select a model of their choice to predict CO₂ emissions based on input features.
- **Data Visualization:** The application provides data visualization capabilities, allowing users to explore fuel consumption data through interactive line plots. Users can select a make to view the fuel consumption trends for different models from the dataset.
- **Model Comparison:** Users can make informed decisions by comparing the specifications of two different vehicle models. This feature enhances the user's ability to choose a vehicle that aligns with their preferences and requirements.
- **Error Rate Calculation:** The application calculates and presents an error percentage, helping users understand the accuracy of the chosen model's predictions concerning actual CO₂ emissions.

6.2 Future Scope:

As we conclude this project, we acknowledge the potential for further enhancements and expansion of our web application. Here are some avenues for future development and improvements:

- **Additional Models:** Incorporating more machine learning models and algorithms for CO₂ emission prediction can provide users with a wider selection and potentially improve prediction accuracy.
- **User Authentication:** Implementing user authentication and user-specific profiles can enhance user experience and allow users to save and track their predictions and comparisons.
- **Data Enrichment:** Expanding the dataset with additional features and more recent data can further improve the application's accuracy and relevance. This may include factors like vehicle weight, emissions standards, or fuel type.
- **Real-Time Data Updates:** Providing real-time data updates and integration with external APIs for current vehicle information, market trends, and environmental impact data.
- **Feedback Mechanism:** Including a feedback mechanism for users to report inaccuracies in the

dataset or provide feedback on the application's usability.

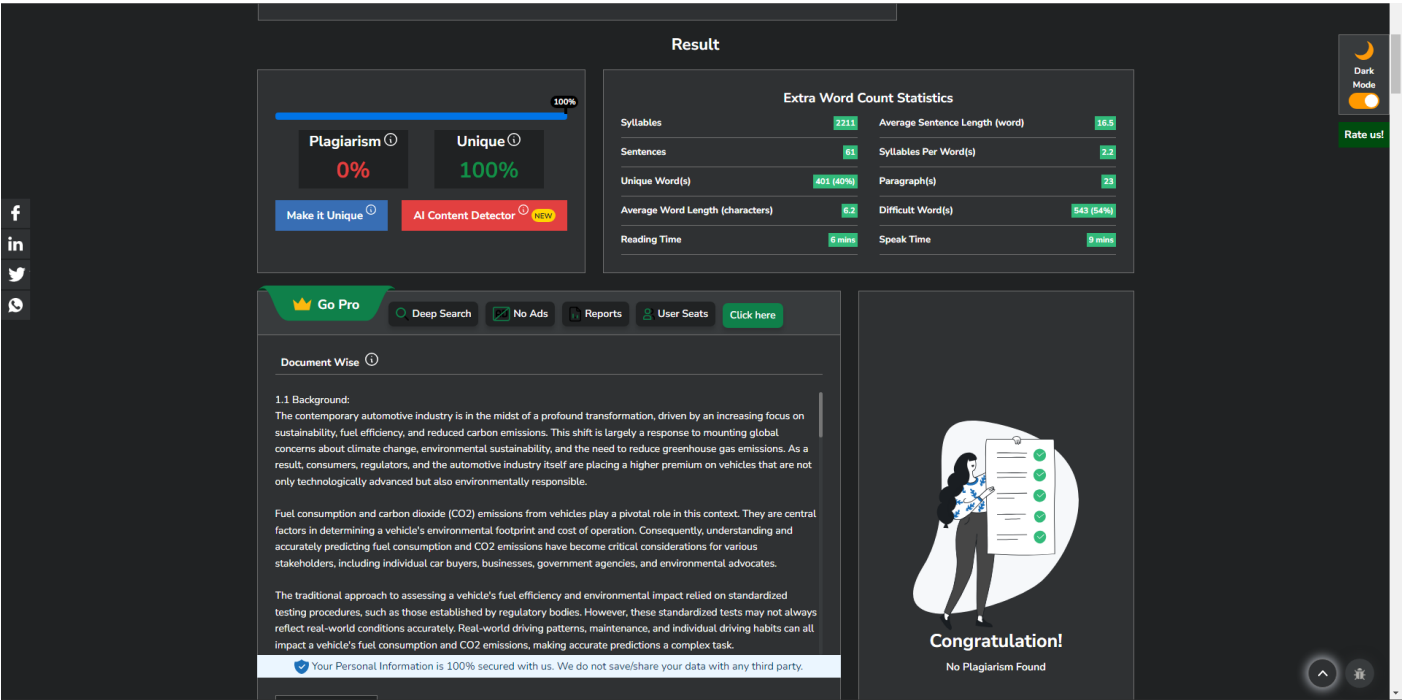
- **Mobile Compatibility:** Optimizing the application for mobile devices and creating a dedicated mobile app for a seamless user experience on smartphones and tablets.

By addressing these future enhancements, the application can evolve into a more comprehensive and user-friendly platform for users seeking to make informed decisions regarding vehicle selection, reducing their environmental footprint, and staying up to date with the latest information in the automotive industry.

REFERENCES

- <https://www.mdpi.com/2073-4433/13/11/1871>
- <https://www.kaggle.com/code/drfrank/co2-emission-eda-visualization-machine-learnin/input>
- <https://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64#wb-auto-6>

PLAGIARISM REPORT



PAPER PUBLICATION PROOF



Asian Society for Academic Research (ASAR)

International Conference on Advanced Research in
Computer Science and Information Technology (ICARCSIT)

Dear Researcher,

Many Congratulations to you!!!

We are happy to inform you that your Manuscript "SUSTAINABLE MOBILITY TRACKER" is selected for ICARCSIT, Kanchipuram, India on 28th October 2023, which will be organized by ASAR and in association with Institute of Research and Journals (IRAJ) (ISO 9001:2008 certified) for presentation in Conference. Conference Proceeding having ISBN number and certificate will be given.

Your paper also cleared the Stage-1(Out of two stages) the publication in the upcoming IRAJ Indexed Journals (Confirmed).

The selected papers may be considered for publication in a Special edited volume from Scopus Indexed Journal, UGC approved Journal and the following IRAJ JOURNALS.

Journal Name	Impact Factor
International Journal of Electrical, Electronics and Data Communication (IJEEDC)	3.46
International Journal of Mechanical and Production Engineering (IJMPE)	3.05
International Journal of Advance Computational Engineering and Networking (IJACEN)	3.89
International Journal of Soft Computing And Artificial Intelligence (IJSCAI)	1.09
International Journal of Advances in Computer Science and Cloud Computing (IJACSCC)	2.05
International Journal of Advances in Science, Engineering and Technology (IJASEAT)	3.05
International Journal of Industrial Electronics and Electrical Engineering (IJIIEEE)	2.51
International Journal of Advances in Mechanical and Civil Engineering (IJAMCE)	3.66
International Journal of Advances in Electronics and Computer Science (IJAECS)	1.90
International Journal of Management and Applied Science (IJMAS)	3.66

Paper Title	SUSTAINABLE MOBILITY TRACKER
Universal Paper ID	AS-CSIT-KNPM-281023-600