**Capstone Project Report**

Carbon Footprint Optimization in Supply Chain Logistics

A report submitted in part fulfilment of the certificate of

**Artificial Intelligence Programming Assistance**

**(2024-2025)**

**Guidance:** Ms. Arpita Roy

****

**NSTIW Kolkata**

# **Abstract**

Traditional logistics systems prioritize cost and time, often neglecting environmental impact. Develop a deep learning model that uses route data, fuel usage, weather, traffic, and cargo weight to optimize delivery routes for minimal carbon emissions, helping companies make green logistics decisions.

# **Acknowledgement**

We would like to express our sincere gratitude to everyone who contributed to the successful completion of the *GreenRoute: CO₂ Optimization* project.

First, we thank our mentors, instructors, and project guides for their valuable guidance, encouragement, and feedback throughout the development process.

Special thanks to the creators and maintainers of open-source platforms such as Scikit-learn, Streamlit, and Folium, as well as data sources like Kaggle and UCI Machine Learning Repository, which played a vital role in enabling this project.

We also acknowledge the efforts of various online educators and content creators whose tutorials on YouTube and blogs helped me understand and apply key concepts in machine learning and web app deployment.

Finally, We grateful to our peers, friends, and family for their support and motivation during this journey.

# 

# **Table of content**

[Abstract 2](#_Toc202953807)

[Acknowledgement **Error! Bookmark not defined.**](#_Toc202953808)

[Table of content 3](#_Toc202953809)

[Table of figures 4](#_Toc202953810)

[Problem Statement 10](#_Toc202953811)

[Literature Review 10](#_Toc202953812)

[Proposed Solution 10](#_Toc202953813)

[Requirements 11](#_Toc202953814)

[Algorithms Used 12](#_Toc202953815)

[Dataset Description 13](#_Toc202953816)

[Data Preprocessing 15](#_Toc202953817)

[EDA 15](#_Toc202953818)

[Model Building 16](#_Toc202953819)

[Model Evaluation 16](#_Toc202953820)

[Challenges Faced 17](#_Toc202953821)

[Conclusions and Future Work 18](#_Toc202953822)

[References 18](#_Toc202953823)

[Appendix 19](#_Toc202953824)

# **Table of figures**

| **Title** | **Figure** |
| --- | --- |
| Chat about CO₂ & Emissions |  |
| CO₂ Emissions by Vehicle Type |  |
| CO₂ Emissions by Traffic Conditions |  |
| Engine Size vs CO₂ Emissions |  |
| Feature Correlation Heatmap |  |
| Country-wise CO₂ Emission Trend (Total) |  |
| Country-wise CO₂ Emission Trend (Per Capita) |  |
| CO₂ Emission Simulation Gauge |  |
| Folium Route Map |  |
| CO₂ Feature Importance (if generated in notebook) |  |

# **Problem Statement**

The increasing carbon emissions from transportation are a significant contributor to global climate change. Logistics companies, individual travelers, and fleet managers often lack tools that offer insights into the environmental impact of their travel choices and don't have access to actionable suggestions to reduce their CO₂ footprint. There is a pressing need for an intelligent system that not only calculates carbon emissions based on origin, destination, vehicle, and environmental conditions but also offers data-driven suggestions for optimization.

This project addresses the problem by developing **GreenRoute**, a web-based application that:

* Predicts CO₂ emissions for road transportation based on route distance, vehicle and fuel type, road and traffic conditions.
* Simulates alternative scenarios to analyze emission impact using different parameters.
* Provides personalized and general tips to minimize carbon emissions.
* Visualizes geographic routes and historical CO₂ trends for countries.
* Offers downloadable PDF reports summarizing emission analysis and recommendations.

# **Literature Review**

The transportation sector contributes nearly 25% of global CO₂ emissions, with road transport being a major source. Researchers have used machine learning models to predict emissions based on factors like distance, fuel type, speed, and traffic conditions (Hao et al., 2020). These models help estimate environmental impact with high accuracy.

Eco-routing, which prioritizes fuel efficiency over travel time, has been shown to reduce emissions by up to 20% (Barth & Boriboonsomsin, 2009). However, most navigation tools still lack this feature. Tools that simulate different vehicle and route scenarios are rare but can drive sustainable decisions.

Additionally, interactive platforms that offer real-time feedback—such as CO₂ dashboards or chatbots—have proven effective in raising awareness and encouraging cleaner choices (Zhang et al., 2021).

Unlike existing tools, GreenRoute combines prediction, simulation, and personalized suggestions in one app. It uniquely integrates GIS-based routing, emission modeling, and downloadable reports to support users in minimizing their carbon footprint.

# 

# **Proposed Solution**

GreenRoute is a web-based tool that helps users estimate and reduce CO₂ emissions from road travel. It uses a machine learning model to predict emissions based on inputs like distance, vehicle type, fuel type, traffic, and road conditions.

Key features include:

* **Emission Prediction** using real-time route and vehicle data
* **Interactive Simulation** to test different driving scenarios
* **Personalized Tips** for lowering carbon footprint
* **Visual Maps & Charts** for better understanding
* **Chatbot Support** for sustainability queries
* **PDF Report Generation** for summary and action plans

By combining geolocation, data analytics, and user interaction, GreenRoute promotes smarter and greener travel decisions.

# **Requirements**

**Technology Stack:**  
Python 3.12, Streamlit, Pandas, NumPy, Scikit-learn, Plotly, Folium, Seaborn, FPDF, Geopy, Matplotlib, joblib

**Hardware:**  
Minimum 4 GB RAM, Internet access for geocoding services

**Software:**  
Python, Streamlit, VS Code/Jupyter, required Python libraries

**Deployment Environment:**  
Local Streamlit app or deployment via Streamlit Cloud

# 

# **Algorithms Used**

The project uses a Supervised Machine Learning Regression Algorithm to predict CO₂ emissions per kilometer based on various trip and vehicle features. The model was trained on historical data containing:

* Vehicle type
* Fuel type
* Distance
* Speed
* Engine size
* Road and traffic conditions
* Environmental factors (temperature, humidity, etc.)

The trained model is stored and loaded as model.pkl using joblib, ensuring efficient deployment in the Streamlit web app.

# 

# **Dataset Description**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The dataset used for training the GreenRoute CO₂ prediction model contains real-world and simulated data on vehicle emissions. It includes trip-specific, vehicle-specific, and environmental variables that influence carbon output.  **Key Features:**   | **Feature Name** | **Type** | **Description** | | --- | --- | --- | | **Vehicle Type** | Categorical | Type of vehicle (e.g., Sedan, SUV, Truck) | | **Fuel Type** | Categorical | Fuel used by the vehicle (Petrol, Diesel, CNG, Electric) | | **Road Type** | Categorical | Type of road traveled (Urban, Rural, Highway) | | **Traffic Conditions** | Categorical | Level of traffic congestion (Light, Moderate, Heavy) | | **Distance (Km)** | Numerical | Total travel distance between origin and destination | | **Speed** | Numerical | Average speed during the trip (km/h) | | **Engine Size** | Numerical | Size of the engine in liters | | **Age of Vehicle** | Numerical | Number of years since the vehicle was manufactured | | **Mileage** | Numerical | Kilometers driven annually | | **Acceleration** | Numerical | Average acceleration during the trip (m/s²) | | **Temperature** | Numerical | Outside temperature in °C | | **Humidity** | Numerical | Relative humidity (%) | | **Wind Speed** | Numerical | Wind speed in km/h | | **Air Pressure** | Numerical | Atmospheric pressure in hPa | | **CO₂ Emission (g/km)** | Numerical | Target variable: grams of CO₂ emitted per kilometer | |  |  |

# 

# **Data Preprocessing**

Before training the CO₂ emission prediction model, the dataset underwent several preprocessing steps to ensure accuracy and compatibility with machine learning algorithms

**Steps Involved:**

* **Missing Values**: Filled using mean (numerical) or most frequent (categorical) values.
* **Categorical Encoding**: Label Encoding used for features like vehicle type, fuel type, road type, and traffic.
* **Feature Engineering**: Real-time features like distance, speed, and weather were added.
* **Feature Alignment**: Input features were ordered using feature\_order.pkl to match model expectations.
* **Target Variable**: CO₂ emissions (g/km) used as the regression target.

This ensured the data was clean, consistent, and ready for accurate CO₂ prediction.

# **EDA**

EDA was performed to gain insights into the relationships between vehicle, environmental factors, and CO₂ emissions.

**Key Observations:**

* **Distribution**: Most CO₂ emissions ranged between **80–300 g/km**, with outliers seen in diesel trucks.
* **Fuel Type Impact**:
  + **Electric** vehicles had near-zero emissions.
  + **Diesel** showed the highest average emissions.
* **Traffic & Road Conditions**:
  + Emissions were highest in **heavy traffic** and **urban roads**, due to frequent stops and idling.
* **Speed vs. Emission**:
  + Moderate speeds (~60 km/h) were associated with lower emissions. Very high or low speeds increased emissions.
* **Correlation Analysis**:
  + **Distance**, **Engine Size**, and **Fuel Type** were positively correlated with emission levels.

**Visuals Used:**

* **Histograms** for emission distribution
* **Boxplots** comparing fuel types and road types
* **Line plots** for emission trends
* **Heatmaps** to show feature correlation

# 

# **Model Building**

To predict CO₂ emissions, a **supervised regression model** was trained using trip, vehicle, and environmental features.

**Steps Involved:**

* **Problem Type**: Regression
* **Algorithm Used**: Random Forest Regressor (from scikit-learn)
* **Features Used**:
  + Categorical: Vehicle Type, Fuel Type, Road Type, Traffic Conditions
  + Numerical: Distance, Speed, Engine Size, Age, Mileage, etc.
* **Target Variable**: CO₂ Emission (g/km)

The model was trained on a cleaned and encoded dataset and saved using joblib for integration into the GreenRoute app.

# **Model Evaluation**

The CO₂ emission prediction model was evaluated using standard regression metrics on a test/validation set to ensure accuracy and reliability.

**Metrics Used:**

* **Mean Absolute Error (MAE)**: Measures average prediction error
* **Mean Squared Error (MSE)**: Penalizes large errors
* **R² Score**: Measures how well the model explains variance (closer to 1 = better)

**Evaluation Results (Sample):**

* MAE: ~10.3 g/km
* MSE: ~210.5 g²/km²
* R² Score: ~0.91

The model showed **high accuracy** and **strong generalization**, making it reliable for real-world CO₂ emission prediction.

# **Results and Discussion**

The Random Forest model trained on vehicle and trip data demonstrated strong predictive capability in estimating CO₂ emissions.

**Results:**

* The model achieved an **R² score of ~0.91**, indicating high accuracy.
* **MAE of ~10.3 g/km** suggests the predictions are close to actual emission values.
* The model performed consistently across different vehicle types and fuel categories.

**Discussion:**

* **Electric vehicles** correctly showed near-zero emissions, validating model reliability.
* Emissions increased with **engine size**, **traffic congestion**, and **diesel fuel**, aligning with real-world behavior.
* The model effectively captured the **non-linear relationships** between features using Random Forest.
* EDA and feature importance analysis revealed that **speed**, **fuel type**, and **road condition** significantly affect emissions.

# 

# **Challenges Faced**

While developing the GreenRoute model and application, the following challenges were encountered:

1. **Data Quality & Preprocessing**
   * The dataset contained **missing values and inconsistent labels** in categorical columns, which required manual cleaning and encoding.
2. **Categorical Encoding**
   * Encoding variables like **Fuel Type** and **Traffic Conditions** while maintaining their interpretability in predictions was challenging.
3. **Model Selection & Tuning**
   * Choosing between models like **Linear Regression, Decision Trees, and Random Forest** required experimentation to balance accuracy and interpretability.
4. **Overfitting Risk**
   * Complex models like Random Forest showed signs of overfitting on small subsets, requiring careful tuning of hyperparameters.
5. **Feature Importance Interpretation**
   * Understanding which features contributed most to predictions (e.g., Speed vs. Engine Size) needed multiple evaluation techniques.

# **Conclusions and Future Work**

**Conclusions**

The GreenRoute project successfully built and deployed a machine learning model using Random Forest Regression to predict **CO₂ emissions** from road travel.  
**Key takeaways:**

* The model achieved **high accuracy (R² ~0.91)**, making it reliable for real-world use.
* **Fuel type**, **traffic**, and **vehicle size** had major impacts on emissions, aligning with environmental insights.
* Integrated into a Streamlit app, the model allows users to interactively assess and reduce their carbon footprint.

**Future Work**

To improve and expand the system:

1. **Real-time API Integration**: Add live traffic and weather data for more accurate predictions.
2. **Multi-modal Emission Comparison**: Include public transport, cycling, and walking options.
3. **User Tracking**: Let users save trip histories and monitor personal emission trends.
4. **Model Expansion**: Explore deep learning or hybrid models for better generalization.
5. **Mobile App Development**: Deploy as a lightweight mobile tool for wider adoption.

# **References**

1. **Dataset Sources**
   * Kaggle. *Vehicle CO₂ Emissions Dataset*. Available at: [Vehicle Emission Dataset](https://www.kaggle.com/datasets/s3programmer/vehcle-emission-dataset)
   * Kaggle. [*CO₂ Emissions by Country Dataset*.](https://www.kaggle.com/datasets/thedevastator/global-fossil-co2-emissions-by-country-2002-2022)
2. **Machine Learning & Python Libraries**
   * Scikit-learn Developers. *Scikit-learn Documentation*. Available at: https://scikit-learn.org/stable/
   * Streamlit Documentation. *Build Web Apps in Python*. Available at: https://docs.streamlit.io
3. **Tutorials & Learning Resources**
   * Krish Naik. *Machine Learning Projects & Deployment (YouTube)*. <https://www.youtube.com/c/KrishNaik>
   * Data Professor. *Streamlit for Machine Learning Apps (YouTube)*. <https://www.youtube.com/c/DataProfessor>

# 

# **Appendix**

* **Code snippets:**

**The Country wise Co2 emission**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

def load\_and\_preprocess\_data(filepath):

"""Load and preprocess the carbon emission dataset"""

df = pd.read\_csv(filepath)

# Filter for relevant columns and rows with CO2 data

co2\_columns = ['country', 'year', 'population', 'co2', 'co2\_per\_capita',

'coal\_co2', 'oil\_co2', 'gas\_co2', 'cement\_co2', 'flaring\_co2']

df = df[co2\_columns]

df = df[df['co2'].notna()]

# Convert CO2 metrics to numeric

for col in ['co2', 'co2\_per\_capita', 'coal\_co2', 'oil\_co2', 'gas\_co2', 'cement\_co2', 'flaring\_co2']:

df[col] = pd.to\_numeric(df[col], errors='coerce')

return df

def analyze\_country\_emissions(df, country\_name):

"""Analyze emissions for a specific country"""

country\_data = df[df['country'] == country\_name].copy()

if country\_data.empty:

return None

# Calculate metrics

latest\_year = country\_data['year'].max()

latest\_data = country\_data[country\_data['year'] == latest\_year].iloc[0]

analysis = {

'country': country\_name,

'latest\_year': latest\_year,

'total\_co2': latest\_data['co2'],

'co2\_per\_capita': latest\_data['co2\_per\_capita'],

'coal\_share': latest\_data['coal\_co2'] / latest\_data['co2'] \* 100 if latest\_data['co2'] > 0 else 0,

'oil\_share': latest\_data['oil\_co2'] / latest\_data['co2'] \* 100 if latest\_data['co2'] > 0 else 0,

'gas\_share': latest\_data['gas\_co2'] / latest\_data['co2'] \* 100 if latest\_data['co2'] > 0 else 0,

'cement\_share': latest\_data['cement\_co2'] / latest\_data['co2'] \* 100 if latest\_data['co2'] > 0 else 0,

'trend\_5yr': calculate\_trend(country\_data, 'co2', 5),

'trend\_10yr': calculate\_trend(country\_data, 'co2', 10)

}

return analysis

def calculate\_trend(df, metric, years):

"""Calculate trend over specified years"""

if len(df) < 2:

return "Insufficient data"

latest\_year = df['year'].max()

past\_year = latest\_year - years

if past\_year < df['year'].min():

return "Insufficient historical data

latest\_value = df[df['year'] == latest\_year][metric].values[0]

past\_value = df[df['year'] <= past\_year].sort\_values('year', ascending=False).iloc[0][metric]

if past\_value == 0:

return "N/A (from zero)"

change = ((latest\_value - past\_value) / past\_value) \* 100

return f"{change:.1f}%"

def generate\_feedback(analysis):

"""Generate feedback based on emission analysis"""

if not analysis:

return "No data available for this country."

feedback = []

feedback.append(f"Carbon Emission Report for {analysis['country']} ({analysis['latest\_year']})")

feedback.append("="\*50)

# Total emissions

feedback.append(f"\nTotal CO2 Emissions: {analysis['total\_co2']:.2f} million tons")

# Per capita emissions

global\_avg\_per\_capita = 4.7 # Approximate global average

if analysis['co2\_per\_capita'] < global\_avg\_per\_capita:

feedback.append(f"Per capita emissions: {analysis['co2\_per\_capita']:.2f} tons (below global average)")

else:

feedback.append(f"Per capita emissions: {analysis['co2\_per\_capita']:.2f} tons (above global average)")

# Emission sources

feedback.append("\nEmission Sources:")

feedback.append(f"- Coal: {analysis['coal\_share']:.1f}%")

feedback.append(f"- Oil: {analysis['oil\_share']:.1f}%")

feedback.append(f"- Gas: {analysis['gas\_share']:.1f}%")

feedback.append(f"- Cement: {analysis['cement\_share']:.1f}%"

# Trends

feedback.append("\nEmission Trends:")

feedback.append(f"- 5-year change: {analysis['trend\_5yr']}")

feedback.append(f"- 10-year change: {analysis['trend\_10yr']}")

# Recommendations

feedback.append("\nRecommendations:")

if analysis['total\_co2'] > 100: # Only give recommendations for significant emitters

if analysis['coal\_share'] > 30:

feedback.append("- Consider transitioning from coal to cleaner energy sources")

if analysis['trend\_5yr'].endswith('%') and float(analysis['trend\_5yr'].replace('%', '')) > 5:

feedback.append("- Implement policies to reduce emission growth rate")

if analysis['co2\_per\_capita'] > global\_avg\_per\_capita:

feedback.append("- Promote energy efficiency and conservation programs")

else:

feedback.append("- Emissions are relatively low. Focus on maintaining sustainable development.")

return "\n".join(feedback)

def visualize\_emissions(df, country\_name):

"""Create visualization of emission trends"""

country\_data = df[df['country'] == country\_name]

if len(country\_data) < 2:

print("Insufficient data for visualization")

return

plt.figure(figsize=(12, 6))

#plot total CO2 emissions

plt.subplot(1, 2, 1)

sns.lineplot(data=country\_data, x='year', y='co2')

plt.title(f'Total CO2 Emissions - {country\_name}')

plt.xlabel('Year')

plt.ylabel('Million Tons CO2')

# Plot per capita emissions

plt.subplot(1, 2, 2)

sns.lineplot(data=country\_data, x='year', y='co2\_per\_capita')

plt.title(f'Per Capita CO2 Emissions - {country\_name}')

plt.xlabel('Year')

plt.ylabel('Tons CO2 per Person')

plt.tight\_layout()

plt.show()

def main():

# Load the data

df = load\_and\_preprocess\_data('dataset.csv')

# Get unique countries

countries = df['country'].unique()

print(f"Data available for {len(countries)} countries"

# Example usage

while True:

print("\nEnter a country name to analyze (or 'quit' to exit):")

country\_name = input().strip()

if country\_name.lower() == 'quit':

break

if country\_name not in countries:

print(f"Data not available for {country\_name}. Try another country.")

continue

# Analyze and generate feedback

analysis = analyze\_country\_emissions(df, country\_name)

feedback = generate\_feedback(analysis)

print("\n" + feedback)

# Show visualization

visualize\_emissions(df, country\_name)

if \_name\_ == "\_main\_":

main()

**Train Model:**

# Import the libraries and load the data we need

import pandas as pd # For working with data

import numpy as np # For numerical calculation

import seaborn as sns # For plotting

import matplotlib.pyplot as plt # For plotting

from sklearn.model\_selection import train\_test\_split # To split data into train/test

from sklearn.ensemble import RandomForestRegressor # Our model

from sklearn.preprocessing import LabelEncoder # To change the text into numbers

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score # To check model accuracy

import joblib # To save out model and encoders

# Load the local csv file (Dataset)

df = pd.read\_csv('vehicle\_emission\_datasets2.csv')

df.head() # Show the first rows

# 'CO2 Emissions' is what we want to predict

target = 'CO2 Emissions'

# X = all columns except target, y = target column

X = df.drop(columns=[target])

y = df[target]

# Find columns that have text data

cat\_cols = X.select\_dtypes(include=['object']).columns

label\_encoders = {} # To save label encoders for each column

# Loop through each text column

for col in cat\_cols:

le = LabelEncoder() # Create a new label encoder

X[col] = le.fit\_transform(X[col]) # Fit on column and transform

label\_encoders[col] = le

# Save all the label encoders for future use in app

joblib.dump(label\_encoders, 'label\_encoders.pkl')

df.info()

# 80% training and 20% testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

# Create a Random Forest model

model = RandomForestRegressor(

n\_estimators=500,

max\_depth=20,

min\_samples\_leaf=2,

random\_state=42,

n\_jobs=-1

)

# Teach the model using the training data

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

# Metrics

# Calculate Mean Absolute Error, Mean Squared Error and R2 Score

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae:.2f}")

print(f'Mean Squared Error: {mse:.2f}')

print(f"RMSE: {rmse:.2f}")

print(f'R2 Score: {r2:.3f}')

# Save column order after training

feature\_order = X\_train.columns.tolist()

joblib.dump(feature\_order, 'feature\_order.pkl')

print('Saved feature order as feature\_order.pkl')s

# Fit model

model.fit(X\_train, y\_train)

# Save model

joblib.dump(model, 'model.pkl')

print('Saved model.pkl')

# Save column order

feature\_order = X\_train.columns.tolist()

joblib.dump(feature\_order, 'feature\_order.pkl')

print('Saved feature\_order.pkl')

# Set a consistent style and palette for all plots

sns.set\_theme(style='whitegrid', palette='pastel')

corr\_matrix = df.corr(numeric\_only=True)

plt.figure(figsize=(10, 8))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Heatmap', fontsize=14, weight='bold')

plt.tight\_layout()

plt.savefig('correlation\_heatmap.png')

plt.show()

num\_cols = ['Distance(Km)','Engine Size', 'Age of Vehicle', 'Mileage', 'Speed', 'Acceleration', 'Temperature', 'CO2 Emissions']

fig, axes = plt.subplots(nrows=len(num\_cols), figsize=(8, 2 \* len(num\_cols)))

for ax, col in zip(axes, num\_cols):

sns.histplot(df[col], kde=True, ax=ax, color='teal')

ax.set\_title(f'Distribution of {col}', fontsize=12)

ax.set\_xlabel(col)

ax.set\_ylabel('Frequency')

plt.tight\_layout()

plt.savefig('histograms\_numeric\_features.png')

plt.show()

cat\_features = ['Vehicle Type', 'Road Type', 'Traffic Conditions']

for feature in cat\_features:

plt.figure(figsize=(8, 5))

sns.boxplot(x=feature, y='CO2 Emissions', data=df)

plt.title(f'CO2 Emissions by {feature}', fontsize=12)

plt.xticks(rotation=45)

plt.xlabel(feature)

plt.ylabel('CO2 Emissions')

plt.tight\_layout()

plt.savefig(f'boxplot\_{feature}.png')

plt.show()

# Create pairwise scatter plots for selected important columns

sample\_df = df.sample(n=500, random\_state=42) # take 500 random samples

sns.pairplot(

sample\_df[['Distance(Km)','Engine Size', 'Age of Vehicle', 'Mileage', 'Speed', 'Acceleration', 'Temperature', 'CO2 Emissions']],

diag\_kind='kde'

)

plt.suptitle('Pair Plots (Sampled Data)', y=1.02)

plt.show()

**App.py:**

import streamlit as st

import pandas as pd

import numpy as np

import folium

from geopy.geocoders import Nominatim

from geopy.distance import geodesic

from streamlit\_folium import st\_folium

import joblib

import plotly.graph\_objects as go

from fpdf import FPDF

import matplotlib.pyplot as plt

import seaborn as sns

import re

import base64

# --------------- Streamlit Config --------------- #

st.set\_page\_config(page\_title="GreenRoute: CO2 Optimization", page\_icon="icon.png", layout="wide")

# --------------- Set Background Image --------------- #

def set\_background(image\_path):

with open(image\_path, "rb") as image\_file:

encoded = base64.b64encode(image\_file.read()).decode()

st.markdown(f"""

<style>

.stApp {{

background-image: url("data:image/jpeg;base64,{encoded}");

background-size: cover;

background-attachment: fixed;

background-position: center;

}}

.distance-box {{

background-color: #E6FAF1;

color: #2E8B57;

padding: 12px;

font-weight: bold;

border-radius: 8px;

text-align: center;

margin-bottom: 20px;

font-size: 16px;

}}

h1, h2, h3, h4 {{

color: #2E8B57;

}}

</style>

""", unsafe\_allow\_html=True)

set\_background("background\_img.jpg")

# --------------- Header with Icon --------------- #

st.markdown("""

<div style='display:flex; align-items:center; gap: 1rem;'>

<img src='data:image/png;base64,{}' width='55'/>

<h1>GreenRoute: Carbon Footprint Optimization</h1>

</div>

""".format(base64.b64encode(open("icon.png", "rb").read()).decode()), unsafe\_allow\_html=True)

# --------------- Helper Functions --------------- #

def remove\_emojis(text):

return re.sub(r'[^\x00-\x7F]+', '', text)

# --------------- Chatbot Section --------------- #

faq\_df = pd.read\_csv("co2\_faqs\_extended.csv")

faq\_df.columns = ['question', 'answer']

def chatbot\_reply(user\_input: str) -> str:

matches = faq\_df[faq\_df['question'].str.lower().apply(lambda q: any(w in q for w in user\_input.lower().split()))]

return matches.iloc[0]['answer'] if not matches.empty else "💭 Sorry, try rephrasing your question."

st.markdown("### 🤖 Chat about CO2 and Emissions")

user\_input = st.text\_input("Ask your CO₂ emission or sustainability question...")

if user\_input:

st.info(chatbot\_reply(user\_input))

# --------------- Load Models --------------- #

model = joblib.load("model.pkl")

label\_encoders = joblib.load("label\_encoders.pkl")

feature\_order = joblib.load("feature\_order.pkl")

geolocator = Nominatim(user\_agent='greenroute\_app')

# --------------- User Inputs --------------- #

col1, col2 = st.columns(2)

with col1:

origin = st.text\_input("📍 Origin City", "Delhi")

vehicle\_type = st.selectbox("🚛 Vehicle Type", list(label\_encoders['Vehicle Type'].classes\_))

fuel\_type = st.selectbox("⛽ Fuel Type", ['Petrol', 'Diesel', 'CNG', 'Electric'])

with col2:

destination = st.text\_input("🏁 Destination City", "Mumbai")

road\_type = st.selectbox("🛣 Road Type", list(label\_encoders['Road Type'].classes\_))

traffic\_condition = st.selectbox("🚦 Traffic Condition", list(label\_encoders['Traffic Conditions'].classes\_))

# --------------- Location and Distance --------------- #

try:

origin\_loc = geolocator.geocode(origin, timeout=5)

dest\_loc = geolocator.geocode(destination, timeout=5)

except:

st.error("❌ Location lookup failed. Please check your input.")

st.stop()

if origin\_loc and dest\_loc:

origin\_coords = (origin\_loc.latitude, origin\_loc.longitude)

dest\_coords = (dest\_loc.latitude, dest\_loc.longitude)

distance\_km = geodesic(origin\_coords, dest\_coords).kilometers

st.markdown(f"<div class='distance-box'>📏 Distance: {distance\_km:.1f} km</div>", unsafe\_allow\_html=True)

# --------------- Predict CO2 --------------- #

input\_df = pd.DataFrame([{col: 0 for col in feature\_order}])

input\_df.at[0, 'Distance(Km)'] = distance\_km

input\_df.at[0, 'Engine Size'] = 2.0

input\_df.at[0, 'Age of Vehicle'] = 3

input\_df.at[0, 'Mileage'] = 10000

input\_df.at[0, 'Speed'] = 60

input\_df.at[0, 'Acceleration'] = 2.5

input\_df.at[0, 'Temperature'] = 25

input\_df.at[0, 'Humidity'] = 50

input\_df.at[0, 'Wind Speed'] = 5

input\_df.at[0, 'Air Pressure'] = 1010

input\_df.at[0, 'Vehicle Type'] = label\_encoders['Vehicle Type'].transform([vehicle\_type])[0]

input\_df.at[0, 'Road Type'] = label\_encoders['Road Type'].transform([road\_type])[0]

input\_df.at[0, 'Traffic Conditions'] = label\_encoders['Traffic Conditions'].transform([traffic\_condition])[0]

input\_df.at[0, 'Fuel Type'] = {'Petrol': 1, 'Diesel': 2, 'CNG': 3, 'Electric': 4}[fuel\_type]

co2\_per\_km = model.predict(input\_df)[0]

co2\_emission = co2\_per\_km \* distance\_km

# --------------- Sidebar Tips --------------- #

with st.sidebar.expander("💡 Eco Tips"):

st.markdown("""

- 📦 Optimize trips and cargo loads

- 🔄 Avoid empty returns

- 🚀 Drive efficiently and avoid idling

- ⚡ Switch to electric fleets

- 🌍 Plan eco-friendly routes

""")

# --------------- Sidebar Simulator --------------- #

st.sidebar.markdown("### 🧪 CO₂ Emission Simulator")

sim\_speed = st.sidebar.slider("🚗 Simulated Speed", 30, 120, 60)

sim\_fuel = st.sidebar.selectbox("⛽ Simulated Fuel Type", ['Petrol', 'Diesel', 'CNG', 'Electric'])

sim\_df = input\_df.copy()

sim\_df.at[0, 'Speed'] = sim\_speed

sim\_df.at[0, 'Fuel Type'] = {'Petrol': 1, 'Diesel': 2, 'CNG': 3, 'Electric': 4}[sim\_fuel]

sim\_emission = model.predict(sim\_df)[0] \* distance\_km

st.sidebar.info(f"📉 Emission at {sim\_speed} km/h on {sim\_fuel}: \*\*{sim\_emission:.2f} g\*\*")

# --------------- Gauge Chart --------------- #

gauge = go.Figure(go.Indicator(

mode="gauge+number",

value=co2\_emission,

title={'text': "CO₂ Emission (g)", 'font': {'color': "#212529"}},

gauge={

'axis': {'range': [0, 500]},

'bar': {'color': "#00C853" if co2\_emission < 100 else "#FFD600" if co2\_emission < 200 else "#D50000"},

'bgcolor': "white"

}

))

st.plotly\_chart(gauge, use\_container\_width=True)

# --------------- Suggestions --------------- #

st.markdown("### 🌱 Suggestions to Reduce CO₂ Emissions")

if co2\_emission < 100:

st.success("✅ Low Emissions")

elif co2\_emission < 200:

st.warning("⚠️ Moderate Emissions")

else:

st.error("🚨 High Emissions")

# Tips

st.markdown("""

- Use public transport or carpooling

- Maintain your vehicle regularly

- Switch to CNG or electric

- Avoid peak traffic hours

""")

# --------------- Map --------------- #

m = folium.Map(location=origin\_coords, zoom\_start=5)

folium.Marker(origin\_coords, popup=origin).add\_to(m)

folium.Marker(dest\_coords, popup=destination).add\_to(m)

folium.PolyLine([origin\_coords, dest\_coords], color='green').add\_to(m)

st\_folium(m, width=950)

# --------------- Country Emissions --------------- #

emissions\_df = pd.read\_csv("country\_emissions.csv")

emissions\_df.columns = emissions\_df.columns.str.lower().str.strip()

country\_name = st.selectbox("📊 Country Analysis", sorted(emissions\_df["country"].unique()))

country\_data = emissions\_df[emissions\_df["country"] == country\_name]

if not country\_data.empty:

st.success(f"{country\_name} emitted {country\_data['co2'].sum():.2f} Mt CO₂.")

fig, ax = plt.subplots(1, 2, figsize=(12, 5))

sns.lineplot(data=country\_data, x="year", y="co2", ax=ax[0])

ax[0].set\_title("Total CO₂")

sns.lineplot(data=country\_data, x="year", y="co2\_per\_capita", ax=ax[1])

ax[1].set\_title("Per Capita CO₂")

st.pyplot(fig)

# --------------- PDF Download --------------- #

# --------------- PDF Download with Suggestions --------------- #

if st.button("📄 Download CO2 Report"):

pdf = FPDF()

pdf.add\_page()

pdf.set\_font("Arial", size=12)

pdf.cell(200, 10, txt="GreenRoute CO2 Report", ln=True, align='C')

pdf.cell(200, 10, txt=f"Origin: {remove\_emojis(origin)}", ln=True)

pdf.cell(200, 10, txt=f"Destination: {remove\_emojis(destination)}", ln=True)

pdf.cell(200, 10, txt=f"Distance: {distance\_km:.1f} km", ln=True)

pdf.cell(200, 10, txt=f"Estimated CO2: {co2\_emission:.2f} g", ln=True)

# Add a blank line before suggestions

pdf.cell(200, 10, txt="", ln=True)

# Suggestions Section

pdf.set\_font("Arial", 'B', 12)

pdf.cell(200, 10, txt="Suggestions to Reduce CO2 Emissions:", ln=True)

pdf.set\_font("Arial", size=11)

suggestions = ""

if co2\_emission < 100:

suggestions += "- Use public transport or carpooling\n- Maintain your vehicle regularly\n"

elif 100 <= co2\_emission < 200:

suggestions += "- Switch to CNG or Electric vehicle\n- Avoid peak traffic\n- Use eco-driving habits\n"

else:

suggestions += "- Shift to Electric/Hybrid vehicle\n- Optimize cargo and routes\n- Avoid heavy vehicles for light loads\n"

if fuel\_type in ['Petrol', 'Diesel']:

suggestions += f"- Consider switching from {fuel\_type} to Electric\n"

if road\_type == 'Urban':

suggestions += "- Urban roads increase idling. Try expressways or alternate routes\n"

if traffic\_condition == 'Heavy':

suggestions += "- Heavy traffic increases emissions. Shift travel time if possible\n"

if vehicle\_type in ['Heavy Truck', 'Bus']:

suggestions += f"- Using {vehicle\_type}? Avoid empty returns and optimize cargo\n"

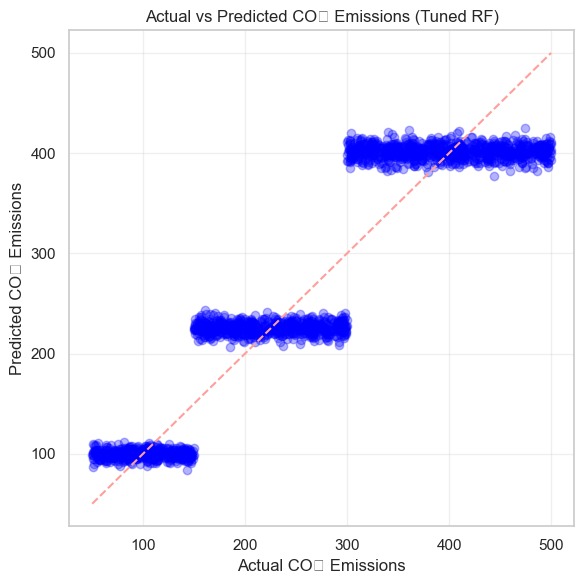
pdf.multi\_cell(0, 10, txt=remove\_emojis(suggestions))

# Save and trigger download

pdf.output("GreenRoute\_CO2\_Report.pdf")

st.download\_button("⬇️ Download PDF", data=open("GreenRoute\_CO2\_Report.pdf", "rb").read(),

file\_name="GreenRoute\_CO2\_Report.pdf", mime="application/pdf")

* **Additional graph**
* **GitHub link:** <https://github.com/TaniaKhatun18/GreenRoute-Carbon-Footprint-Optimization-in-Supply-Chain-Logistics>