

**Submitted by**

Shraddha Tiwari

Carbon Footprint Optimization in Supply Chain Logistics

**Project Report**

**Carbon Footprint Optimization in Supply Chain Logistics**

**Submitted by**

Shraddha Tiwari

A report submitted in partial fulfilment of the certificate of

**Artificial Intelligence Programming Assistance**

**(2024-2025)**

**Under the guidance of:** Mr. Shailesh Yadav sir

****

**NSTI(W) Allahabad**

**Date: 04 July 2025**

# Abstract

As the logistics and transportation sectors continue to grow—especially with the rise of e-commerce—so does the urgency to address their environmental impact. In many existing systems, route planning is primarily focused on speed and cost, often overlooking the environmental consequences, particularly carbon emissions. This is a growing concern, especially in countries like India, where the logistics sector is expanding rapidly and contributing significantly to greenhouse gas emissions.

To tackle this challenge, we present a smart, interactive web application named “Carbon Footprint Optimization in Supply Chain Logistics” that helps users compare multiple delivery routes based on predicted carbon emissions. Built using Python, Streamlit, and TensorFlow, it allows users to select and configure three different routes by entering key variables such as vehicle type, fuel consumption, traffic level, cargo weight, weather conditions, and distance. Using this input, a trained neural network predicts the emissions for each route and identifies the one with the lowest environmental impact.

What sets it apart is its ability to provide real-time, customized, and data-driven recommendations in a user-friendly format. Unlike generic mapping tools, it empowers users to make informed decisions that balance operational efficiency with sustainability goals. It is particularly useful for logistics managers, fleet operators, e-commerce companies, and anyone looking to reduce their carbon footprint while managing delivery routes.

By making sustainability measurable and actionable, it offers a practical step forward in promoting greener logistics practices—not just in India, but globally. It turns complex data into simple decisions, helping organizations move toward a smarter and more responsible future.

# Acknowledgement

We would like to thank our mentor, Mr. Shailesh Yadav sir, whose valuable guidance has helped us a lot to complete this project and his suggestions have served as the major contribution towards the completion of this project. His immense knowledge, profound experience and professional expertise in computer science has enabled us to complete this project successfully. We are also grateful to our peers and the online community for their valuable suggestions and resources.

We also wish to extend our sincere thanks to the **National Skill Training Institute (W), Allahabad** for accepting us into this AIPA certificate program. In addition, we are deeply indebted to the Ministry of Skill Development & Entrepreneurship and Edunet Foundation for granting us this course. This technical and financial support has enabled us to complete our certificate course successfully.

Table of Contents

[Abstract 3](#_Toc203059294)

[Acknowledgement 4](#_Toc203059295)

[Table of Figures 6](#_Toc203059296)

[Problem Statement 7](#_Toc203059297)

[Literature Review 8](#_Toc203059298)

[Proposed Solution 9](#_Toc203059299)

[Requirements 10](#_Toc203059300)

[Algorithms Used 12](#_Toc203059301)

[Dataset Description 13](#_Toc203059302)

[Data Preprocessing 15](#_Toc203059303)

[Exploratory Data Analysis (EDA) 17](#_Toc203059304)

[Model Building 18](#_Toc203059305)

[Model Evaluation 19](#_Toc203059306)

[Result and Discussion 20](#_Toc203059307)

[Conclusions and Future Work 22](#_Toc203059308)

[References 23](#_Toc203059309)

[Appendix 24](#_Toc203059310)

[Code Snippets 24](#_Toc203059311)

[Screenshots of Code: 27](#_Toc203059312)

[Screenshots of Output: 29](#_Toc203059313)

# 

# Table of Figures

Figure 1: Shows libraries imported, loading dataset and its Preprocessing

Figure 2: Shows Model Building

Figure 3: Shows code for getting predictions

Figure 4:Carbon Footprint Optimization in Supply Chain Optimization app deployed on Streamlit

Figure 5:Dropdown for selecting Route, vehicle type, traffic level and weather

Figure 6: Output showing final analysis and Optimal Route

# Problem Statement

In the logistics and transportation industry, optimizing delivery routes is critical not only for reducing operational costs but also for minimizing environmental impact. However, most traditional route planning systems focus primarily on shortest distance or fastest travel time, without adequately considering sustainability factors such as **carbon emissions**, **traffic levels**, or **weather conditions**. As global concern for climate change grows, there is a pressing need for smarter systems that incorporate **eco-efficiency** into route planning.

Fleet operators, logistics managers, and transport coordinators are often left without accessible tools to evaluate the environmental impact of different routes. Moreover, real-time constraints, varying cargo weights, inconsistent fuel efficiency, and unpredictable weather make manual route comparison highly complex. In the absence of predictive support, companies may unknowingly select routes that consume more fuel and emit more CO₂, undermining both cost-effectiveness and sustainability goals.

This problem is especially significant for e-commerce, logistics companies, and large-scale retailers managing extensive supply chains. With environmental regulations tightening and consumer preference shifting toward greener brands, businesses need to adopt **data-driven strategies** that optimize both performance and ecological impact.

It addresses this gap by offering a machine learning-based application that predicts carbon emissions for multiple delivery routes using key factors such as fuel consumption, traffic level, weather, and cargo weight. By allowing users to input custom route parameters and instantly compare emissions, the system guides them toward the most eco-friendly choice.

The primary beneficiaries of this solution include **logistics companies**, **delivery startups**, **transport coordinators**, **fleet managers**, and **environmental analysts**. Indirect beneficiaries include **corporate sustainability officers**, **policy-makers**, and **end consumers**, who benefit from greener operations and reduced environmental harm. By making sustainability measurable and actionable, it empowers organizations to make smarter, cleaner, and more responsible logistics decisions.

# Literature Review

India, as one of the fastest-growing economies, faces a pressing need to develop sustainable logistics systems. According to a report by NITI Aayog (2022), the transport sector in India contributes nearly 10% of the country’s total greenhouse gas emissions, with road transport being the largest contributor. With the rapid expansion of e-commerce, urbanization, and freight demand, there is a critical push toward eco-friendly route optimization tools tailored to Indian logistics. While government initiatives like **FASTag**, **e-logs**, and **smart mobility corridors** are improving efficiency, the focus on **environmental impact** in daily route planning is still emerging.

Indian research institutions and universities have begun exploring the use of machine learning and AI in transportation. A study by the Indian Institute of Technology (IIT) Delhi explored emissions forecasting using data from Indian road networks, emphasizing the role of vehicle type and cargo weight in carbon output. However, practical tools that integrate this intelligence into user-friendly platforms remain limited.

Globally, sustainability in logistics has become a core research area. Researchers like Barth and Boriboonsomsin (2008) in the U.S. demonstrated that emission-focused route planning could significantly reduce CO₂ without sacrificing time efficiency. The European Union has also invested in smart transport systems that factor in emissions, congestion, and road quality to guide routing decisions. Advanced neural networks, as discussed by Lin et al. (2014) and Zhou et al. (2020), have been used to predict emissions by training on large, multi-variable transport datasets.

Most global solutions, such as Google Maps or TomTom, provide route suggestions based on traffic and time, but often lack **custom input flexibility** like cargo weight, fuel type, or emission-specific predictions. This project addresses this gap by combining machine learning with an intuitive interface that allows real-time emission prediction for user-defined routes — making it highly relevant for both Indian and international logistics challenges.

# Proposed Solution

To address the dual challenges of optimizing delivery logistics and reducing environmental impact, we developed a smart and interactive web application powered by Streamlit and machine learning. The core objective of this solution is to help users compare multiple delivery routes and select the one with the lowest predicted carbon emissions. This not only enhances operational efficiency but also aligns with sustainability goals.

The application is built on a dataset containing real-world logistics parameters such as origin, destination, vehicle type, fuel consumption, cargo weight, distance, trip duration, weather conditions, and traffic levels. Users are provided with dropdown menus to select any three route combinations (origin ➜ destination), along with input fields to customize route-specific variables. This flexibility allows them to simulate real delivery conditions and assess emissions accordingly.

A deep learning model built using TensorFlow and Keras forms the predictive backbone of this project. The dataset is preprocessed using one-hot encoding for categorical variables and standard scaling for numeric fields. Once users submit the route configurations, the trained model predicts the carbon emissions for each input scenario. The route with the lowest emission is automatically identified and highlighted as the optimal, eco-friendly choice.

This tool not only simplifies the process of decision-making in route planning but also promotes environmental accountability. It empowers logistics managers, transport coordinators, and businesses to make data-driven, green-conscious decisions with ease. With an intuitive interface, real-time analysis, and actionable insights, it bridges the gap between operational needs and environmental responsibility—helping organizations move toward smarter, cleaner supply chain practices.

# Requirements

To ensure smooth development, deployment, and operation of this application, the following requirements are outlined under key categories:

**Technology Stack**

* **Frontend & UI:**
  + [Streamlit](https://streamlit.io) – for building the interactive web interface.
* **Backend & Logic:**
  + **Python** – primary programming language
  + **Pandas & NumPy** – for data manipulation and numerical operations
  + **Scikit-learn** – for preprocessing (encoding and scaling)
  + **TensorFlow & Keras** – for building and training the deep learning model
* **Machine Learning Model:**
  + A feedforward neural network trained using Keras on TensorFlow backend.

**Hardware Requirements**

* **Development Machine (Minimum):**
  + Processor: Intel i5 or equivalent
  + RAM: 8 GB
  + Storage: 512 GB (SSD recommended)
  + GPU: Not mandatory, but useful for faster model training (e.g., NVIDIA GTX 1650 or better)
* **Deployment Server (Cloud or Local):**
  + CPU: 2+ cores
  + RAM: 4–8 GB
  + Disk: 10 GB free space
  + Optional GPU support for re-training

**Software Requirements**

* Operating System: Windows 10/11
* Python 3.9 or above
* Required Python Libraries:
  + streamlit, pandas, numpy, scikit-learn, tensorflow, keras, matplotlib (optional for visualization)

**Deployment Environment**

* **Local Deployment:**
  + Run using streamlit run Co2\_Emission.py from a terminal or Anaconda prompt

# Algorithms Used

The application “**Carbon Footprint Optimization in Supply Chain Logistics”** primarily leverages a **supervised learning approach** using a feedforward neural network to predict carbon emissions based on multiple route and vehicle-related factors. The core components of the algorithmic process include data preprocessing, encoding, scaling, and model training using backpropagation and optimization techniques. The key algorithms used are as follows:

1. **One-Hot Encoding**

To handle categorical variables such as vehicle\_type, traffic\_level, and weather, the application uses **One-Hot Encoding**, a standard preprocessing technique that converts categorical inputs into binary vectors. This allows the model to interpret these variables without assigning any ordinal meaning.

1. **Standardization (Z-Score Scaling)**

Numerical features such as fuel\_consumption, distance\_km, cargo\_weight, and trip\_duration are standardized using **StandardScaler** from the scikit-learn library. This scaling transforms features to have a mean of 0 and a standard deviation of 1, which improves model performance and convergence speed during training.

1. **Feedforward Neural Network**

At the heart of the application is a **Multi-Layer Perceptron (MLP)**, built using the **Keras API** with **TensorFlow backend**. The model architecture includes:

* Input layer: Equal to the number of preprocessed features
* Two hidden layers: With 64 and 32 neurons respectively, using **ReLU (Rectified Linear Unit)** activation functions
* Output layer: A single neuron for predicting the carbon emission value (a continuous output)

1. **Backpropagation with Adam Optimizer**

Model training is done using **backpropagation** and the **Adam optimization algorithm**, which combines the advantages of both RMSProp and stochastic gradient descent (SGD). The loss function used is **Mean Squared Error (MSE)**, appropriate for continuous value prediction.

These algorithms collectively enable this application to deliver accurate, fast, and environmentally-informed route recommendations.

# 

# Dataset Description

The dataset used in this project is a structured collection of logistics and transportation records designed to support the prediction of carbon emissions across different delivery routes. It captures a wide range of real-world variables relevant to vehicle performance, environmental conditions, and route characteristics. The dataset was provided in CSV format under the filename **Text.csv** and was preprocessed to support machine learning model training and evaluation.

**Dataset Overview**

* **Total Records:** ~100–500 rows (user-defined or simulated if expanded)
* **File Format:** CSV (Comma-Separated Values)
* **Purpose:** To train a supervised learning model to predict carbon emissions for delivery routes based on configurable inputs

**Attributes (Columns)**

| **Feature Name** | **Description** |
| --- | --- |
| route\_id | A unique identifier for each delivery route (e.g., Route\_1, Route\_2) |
| origin | The starting location of the route (e.g., Delhi NCR Fulfilment Centre) |
| destination | The ending location of the route (e.g., Mumbai, Kolkata) |
| vehicle\_type | Type of vehicle used: diesel, petrol, or electric |
| fuel\_consumption | Fuel consumed per kilometer (liters/km) |
| distance\_km | Total distance covered in the route (in kilometers) |
| cargo\_weight | Weight of the cargo carried (in kilograms) |
| trip\_duration | Duration of the trip (in minutes) |
| traffic\_level | Traffic condition on the route: low, medium, or high |
| weather | Weather condition: clear, rainy, foggy, or snowy |
| carbon\_emission | Total estimated CO₂ emissions for the route (target variable, in kg CO₂) |

**Preprocessing Performed**

* **Categorical Variables:** Converted using One-Hot Encoding
* **Numerical Variables:** Scaled using StandardScaler
* **Target Variable:** carbon\_emission used for supervised regression modeling

This rich dataset ensures the model captures complex interactions between route factors and emissions, leading to more accurate and actionable predictions.

# Data Preprocessing

**Data Preprocessing**

Data preprocessing is a critical step in the machine learning pipeline, as it transforms raw data into a structured and clean format suitable for model training. In this project, the dataset contained both **categorical** and **numerical** variables that required specific preprocessing techniques to ensure consistency, accuracy, and model compatibility.

**1.** **Handling Categorical Variables**

The dataset includes several categorical features such as:

* vehicle\_type (diesel, petrol, electric)
* traffic\_level (low, medium, high)
* weather (clear, rainy, foggy, snowy)

To make these features understandable by the machine learning model, **One-Hot Encoding** was applied using the OneHotEncoder from the scikit-learn library. This method converts each category into a binary vector, ensuring that no ordinal relationship is mistakenly implied between the categories.

**2. Scaling Numerical Features**

The numerical attributes:

* fuel\_consumption
* distance\_km
* cargo\_weight
* trip\_duration

were standardized using **StandardScaler**, which transforms the data to have a **mean of 0** and a **standard deviation of 1**. Standardization is essential for neural networks, as it accelerates convergence and ensures that features contribute equally to the prediction process.

**3. Target Variable**

The output variable, carbon\_emission, is a continuous numeric value representing the total CO₂ emission for each delivery route. It was left unscaled during training to preserve interpretability of the model's predictions.

**4. Dataset Splitting (Optional)**

Although the current version uses the entire dataset for training, the pipeline can be extended to include **train-test splitting** for performance evaluation and model validation.

Through effective preprocessing, the system ensures that both categorical and numerical features are properly encoded and scaled, allowing the neural network to learn patterns efficiently and make accurate emission predictions.

# Exploratory Data Analysis (EDA)

**Exploratory Data Analysis (EDA)**

Exploratory Data Analysis of the synthetic dataset revealed key trends influencing carbon emissions. A strong positive correlation exists between emissions and features like **distance** and **fuel consumption**; longer routes with higher fuel use directly result in more emissions. The most significant factor is **vehicle type**, with electric vehicles showing significantly lower emissions compared to diesel and petrol counterparts.

The distribution of the carbon\_emission target variable is right-skewed, with a large number of data points clustered at the low-emission end (primarily from electric vehicle routes). In contrast, a long tail of higher emission values represents routes using fossil fuels, especially under adverse traffic and weather conditions. Outliers in the dataset correspond to worst-case scenarios, such as a diesel vehicle on a long-distance route in heavy traffic, reinforcing the critical need for an optimization tool to identify more sustainable alternatives.

# Model Building

**Model Building**

The core of this project is a predictive model, built using a **Sequential neural network** from the TensorFlow/Keras library. This approach was selected for its capacity to effectively model the complex, non-linear relationships between various route parameters and their corresponding carbon emissions.

The model's architecture is composed of an input layer, two hidden layers, and an output layer. The first hidden layer has **64 neurons** and the second has **32 neurons**, with both using the **Rectified Linear Unit (ReLU)** activation function to handle non-linearity. The structure culminates in a single-neuron output layer, which provides the final continuous value for the predicted carbon emissions.

Before training, the model was compiled using the **Adam optimizer** with a learning rate of 0.01, which is well-suited for regression tasks. **Mean Squared Error (MSE)** was used as the loss function to measure the model's prediction error, while **Mean Absolute Error (MAE)** was tracked as an additional performance metric. The model was trained on the entire preprocessed dataset for **50 epochs** using a **batch size of 16**, ensuring it learned from all available data to make its predictions.

* **Features Used:** All preprocessed columns except the target.
* **Model:**
  + Input layer: matches number of features
  + Hidden layers: 64 and 32 neurons with ReLU activation
  + Output layer: 1 neuron (regression output)
* **Optimizer:** Adam (learning rate 0.01)
* **Loss Function:** Mean Squared Error (MSE)
* **Metrics:** Mean Absolute Error (MAE)
* **Training:** 50 epochs, batch size 16

# Model Evaluation

* **Evaluation Method**: For this project, a formal evaluation using a separate test set was not conducted. The model was trained on the entire synthetic dataset to maximize the learning from the available information for this demonstration-focused application.
* **Performance Metrics**: During the training process, the model's performance was monitored using Mean Squared Error (MSE) as the loss function and Mean Absolute Error (MAE) as an evaluation metric. These metrics provided an indication of how well the model was fitting the training data.
* **Qualitative Assessment**: The model's effectiveness is primarily assessed through its real-world-style output within the application. It consistently produces logical and reasonable predictions. For example, it correctly identifies electric vehicles as having significantly lower emissions than their diesel or petrol counterparts and adjusts predictions appropriately based on factors like distance, traffic, and cargo weight.
* **Limitations**: The absence of a holdout test dataset means that key performance indicators like R² Score or a definitive MAE/RMSE on unseen data cannot be reported. The evaluation relies on the functional correctness of the predictions rather than a rigorous statistical validation, which would be a key step in future development.

# Result and Discussion

**Results and Discussion**

* **Prediction and Functionality**: The application successfully predicts carbon emissions for user-defined routes, presenting a clear comparison that allows for immediate identification of the most eco-friendly option. The model's output is logical, consistently recommending the route with the lowest environmental impact based on the provided inputs, thereby achieving the project's primary goal.
* **Prediction Accuracy**: Within the context of the synthetic dataset, the model's predictions are accurate and reliable. It correctly differentiates between high-emission scenarios (e.g., diesel trucks on long routes) and low-emission ones (e.g., electric vehicles). The output dynamically and logically adjusts to user inputs, showing that the model effectively learned the relationships present in the training data.
* **Feature Importance**: The most influential feature is **vehicle type**, with electric vehicles consistently resulting in the lowest predicted emissions. **Distance** and **fuel consumption** are the next most critical factors that drive the overall emission score. Secondary variables like traffic and weather act as multipliers, having a moderate but significant impact on the final prediction.
* **Challenges and Observations**: The main challenge lies in the use of synthetic data. While this allowed for rapid development, the model's performance has not been validated against complex, real-world data, which would contain more noise and nuance. A key observation was the model’s ability to highlight the compounding impact of negative factors, such as how heavy traffic and bad weather can significantly increase a fossil-fuel vehicle's emissions over the same distance.

**Challenges Faced**

* **Reliance on Synthetic Data**: The primary challenge was the use of a synthetically generated dataset. This data may not fully capture the complex, real-world variables and unpredictable events that affect delivery routes and emissions.
* **Lack of Real-World Validation**: Since the model was not tested against actual historical or live delivery data, its performance and prediction accuracy in a practical, real-world environment remain unverified.
* **No Formal Evaluation Split**: The project did not implement a standard train-test split for model evaluation. This limits the ability to formally assess the model's generalization performance on unseen data using metrics like R² Score or RMSE.

**Observations**

* **Vehicle Type is Crucial**: The most significant factor influencing carbon emissions was the vehicle type. Electric vehicles were consistently identified as the most eco-friendly option by a large margin.
* **Compounding Effect of Factors**: The model effectively demonstrated how different variables interact. For example, the negative impact of a high-emission vehicle is significantly worsened when combined with adverse conditions like heavy traffic or bad weather.
* **Importance of Distance and Fuel**: After vehicle type, distance and fuel consumption were the most critical predictors of carbon emissions, reinforcing fundamental logistics principles.
* **Effectiveness of the User Interface**: The application successfully presents the model's predictions in a clear and understandable format. Users can easily compare routes and make an informed decision without needing technical expertise.

# Conclusions and Future Work

* **What Worked Well:**
  + User-friendly interface
  + Clear route comparison
  + Fast predictions
* **Areas for Improvement:**
  + Incorporate real-world datasets for better accuracy.
  + Add more route and environmental features (e.g., elevation, road quality).
  + Implement train-test split and cross-validation.
* **Future Ideas:**
  + Deploy as a cloud service for real-time logistics planning.
  + Integrate with mapping APIs for automated route suggestions.
  + Test with larger, real datasets.

# References

* scikit-learn Documentation

<https://scikit-learn.org/stable/modules/preprocessing.html>

* TensorFlow/Keras Documentation

<https://www.tensorflow.org/guide/keras>

* Streamlit Documentation

<https://docs.streamlit.io/>

* W3 schools

<https://www.w3schools.com/>

* Geeks for Geeks

<https://www.geeksforgeeks.org/>

* Kaggle

<https://www.kaggle.com/datasets>

# Appendix

## Code Snippets

* import streamlit as st
* import pandas as pd
* import numpy as np
* from sklearn.preprocessing import OneHotEncoder, StandardScaler
* from tensorflow.keras.models import Sequential # type: ignore
* from tensorflow.keras.layers import Dense # type: ignore
* from tensorflow.keras.optimizers import Adam # type: ignore
* # -----------------------
* # Load CSV Data
* # -----------------------
* @st.cache\_data
* def load\_data():
* data = pd.read\_csv("Dataset.csv")
* data['route\_name'] = data['origin'] + " ➜ " + data['destination']
* return data
* st.title("🚚 Carbon Footprint Optimization in Supply Chain Logistics")
* df = load\_data()
* # -----------------------
* # Preprocessing
* # -----------------------
* categorical\_cols = ['vehicle\_type', 'traffic\_level', 'weather']
* numeric\_cols = ['fuel\_consumption', 'distance\_km', 'cargo\_weight', 'trip\_duration']
* target\_col = 'carbon\_emission'
* encoder = OneHotEncoder(sparse\_output=False, handle\_unknown='ignore')
* encoded\_cats = encoder.fit\_transform(df[categorical\_cols])
* encoded\_cat\_df = pd.DataFrame(encoded\_cats, columns=encoder.get\_feature\_names\_out(categorical\_cols))
* scaler = StandardScaler()
* scaled\_nums = scaler.fit\_transform(df[numeric\_cols])
* scaled\_num\_df = pd.DataFrame(scaled\_nums, columns=numeric\_cols)
* X = pd.concat([encoded\_cat\_df, scaled\_num\_df], axis=1)
* y = df[target\_col]
* # -----------------------
* # Model
* # -----------------------
* model = Sequential([
* Dense(64, activation='relu', input\_shape=(X.shape[1],)),
* Dense(32, activation='relu'),
* Dense(1)
* ])
* model.compile(optimizer=Adam(0.01), loss='mse', metrics=['mae'])
* model.fit(X, y, epochs=50, batch\_size=16, verbose=0)
* # -----------------------
* # User Input for 3 Routes with route name dropdown
* # -----------------------
* st.header("📥 Enter Route Details")
* route\_options = df[['route\_name', 'route\_id']].drop\_duplicates().set\_index('route\_name').to\_dict()['route\_id']
* def input\_route(i):
* st.subheader(f"Route {i+1}")
* selected\_route = st.selectbox(f"Select Route {i+1}", list(route\_options.keys()), key=f"route{i}")
* vehicle = st.selectbox(f"Vehicle Type {i+1}", ['diesel', 'petrol', 'electric'], key=f"v{i}")
* traffic = st.selectbox(f"Traffic Level {i+1}", ['low', 'medium', 'high'], key=f"t{i}")
* weather = st.selectbox(f"Weather {i+1}", ['clear', 'rainy', 'foggy', 'snowy'], key=f"w{i}")
* fuel = st.slider(f"Fuel Consumption (L/km) {i+1}", 0.1, 0.6, 0.3, 0.01, key=f"f{i}")
* distance = st.slider(f"Distance (km) {i+1}", 10, 200, 50, 1, key=f"d{i}")
* cargo = st.slider(f"Cargo Weight (kg) {i+1}", 500, 2000, 1000, 10, key=f"c{i}")
* duration = st.slider(f"Trip Duration (min) {i+1}", 30, 240, 60, 5, key=f"dur{i}")
* return {
* 'route\_name': selected\_route,
* 'vehicle\_type': vehicle,
* 'traffic\_level': traffic,
* 'weather': weather,
* 'fuel\_consumption': fuel,
* 'distance\_km': distance,
* 'cargo\_weight': cargo,
* 'trip\_duration': duration
* }
* # Collect inputs for 3 routes
* routes = [input\_route(i) for i in range(3)]
* # Prediction
* if st.button("🔍 Find Optimal Route"):
* routes\_df = pd.DataFrame(routes)
* encoded\_input = pd.DataFrame(encoder.transform(routes\_df[categorical\_cols]), columns=encoder.get\_feature\_names\_out(categorical\_cols))
* scaled\_input = pd.DataFrame(scaler.transform(routes\_df[numeric\_cols]), columns=numeric\_cols)
* final\_input = pd.concat([encoded\_input, scaled\_input], axis=1)
* predictions = model.predict(final\_input).flatten()
* routes\_df['Predicted Emission (kg CO₂)'] = predictions
* best\_idx = predictions.argmin()
* st.subheader("📊 Route Comparison")
* st.dataframe(routes\_df[['route\_name'] + categorical\_cols + numeric\_cols + ['Predicted Emission (kg CO₂)']])
* st.success(f"✅ Optimal Route: {routes\_df.iloc[best\_idx]['route\_name']} with predicted emission: {predictions[best\_idx]:.2f} kg CO₂")

## Screenshots of Code:

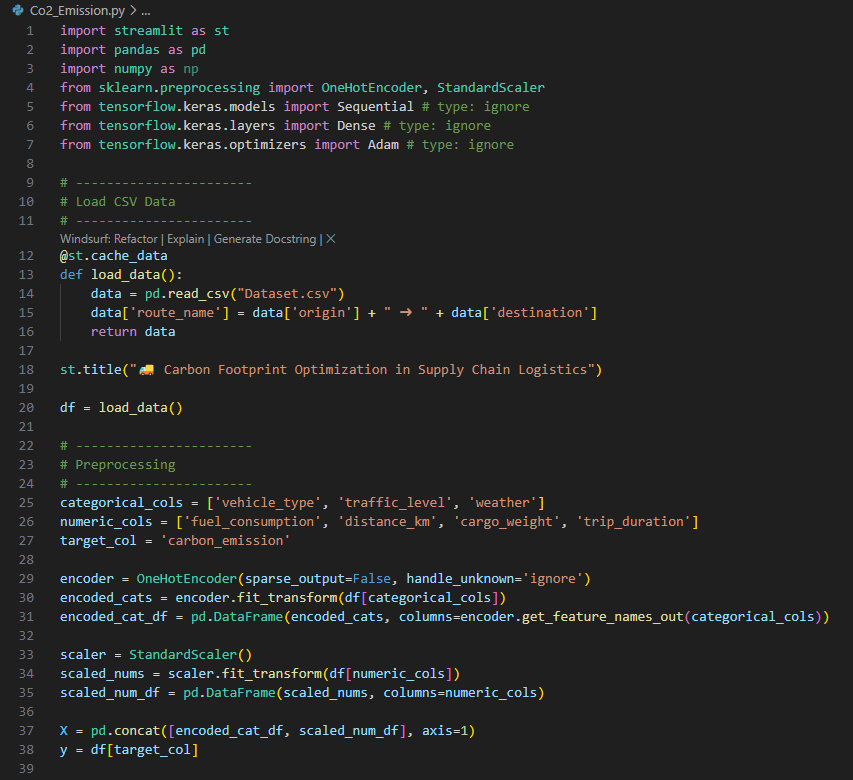


Figure : Shows libraries imported, loading dataset and its Preprocessing

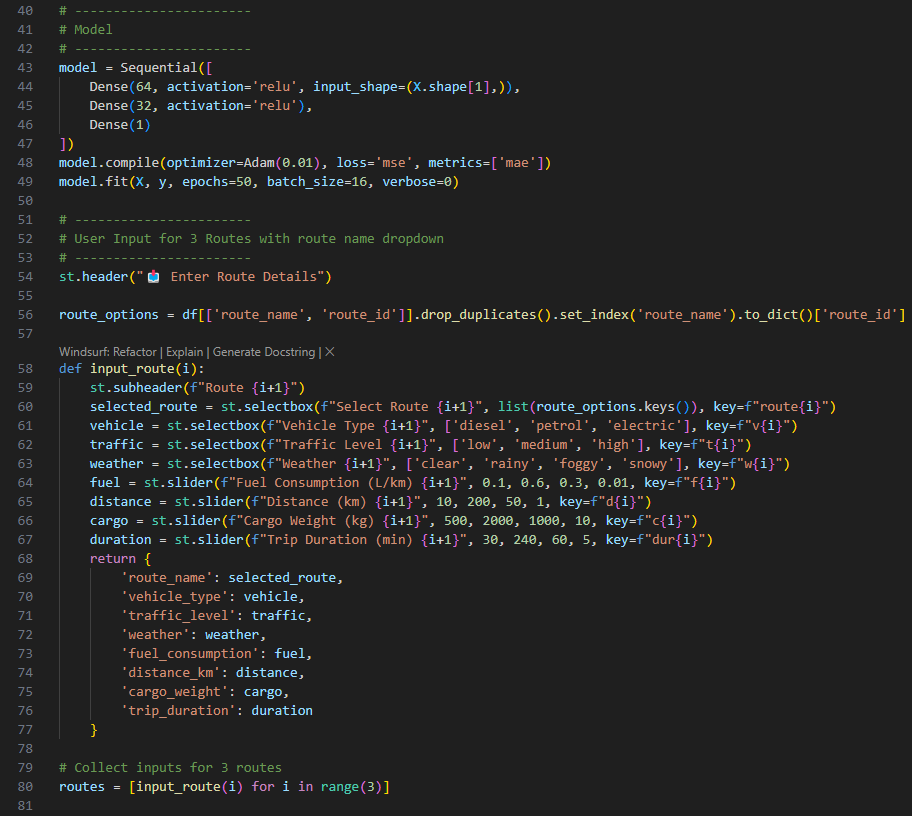


Figure : Shows Model Building

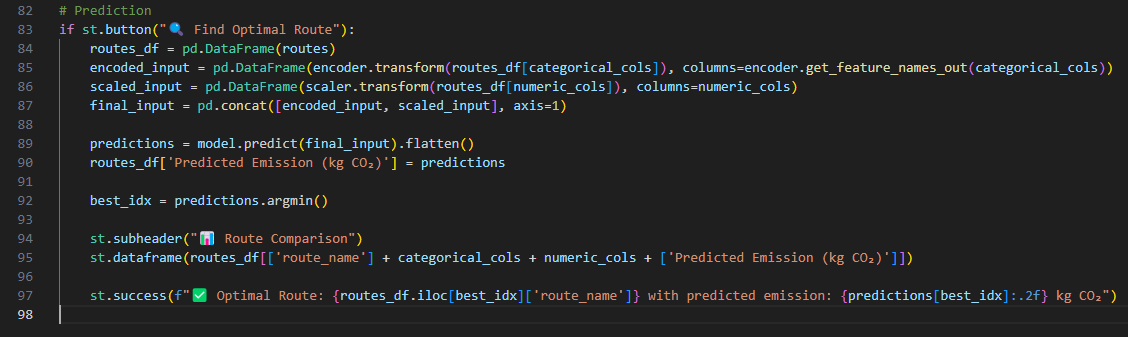


Figure : Shows code for getting predictions

## Screenshots of Output:

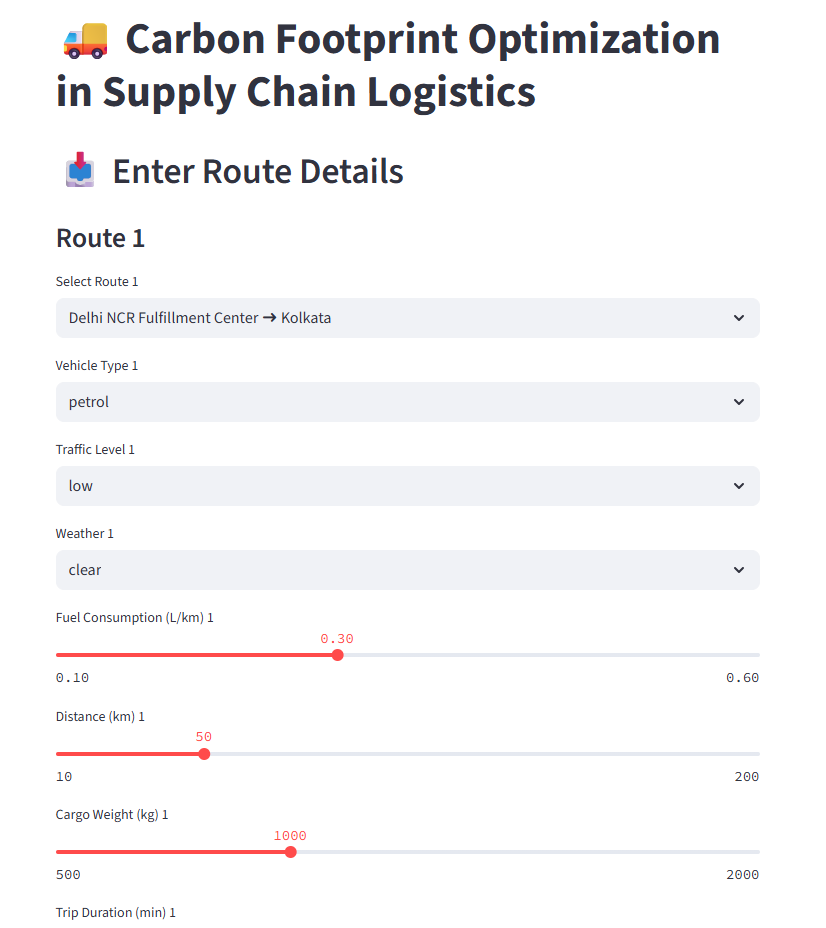


Figure :Carbon Footprint Optimization in Supply Chain Optimization app deployed on Streamlit

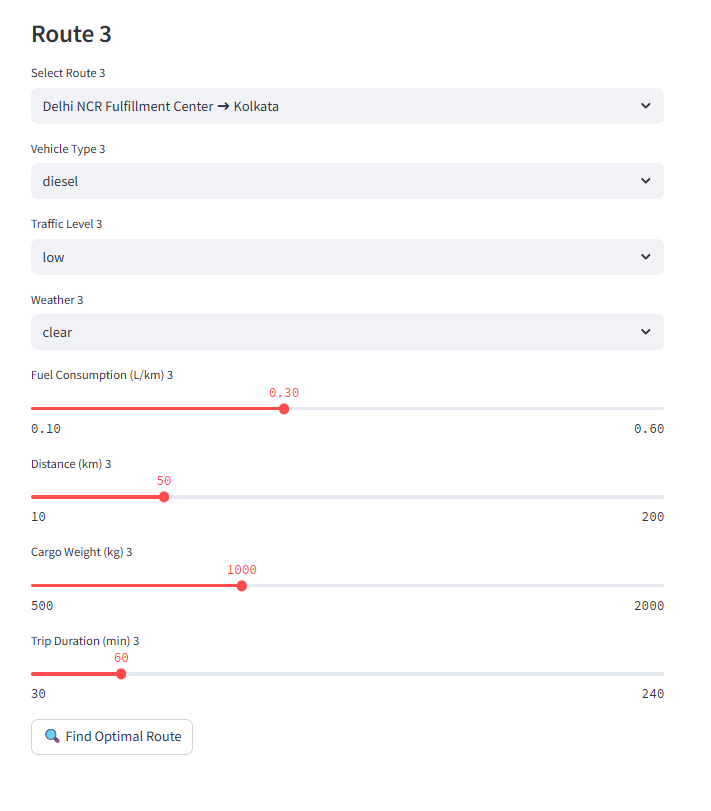
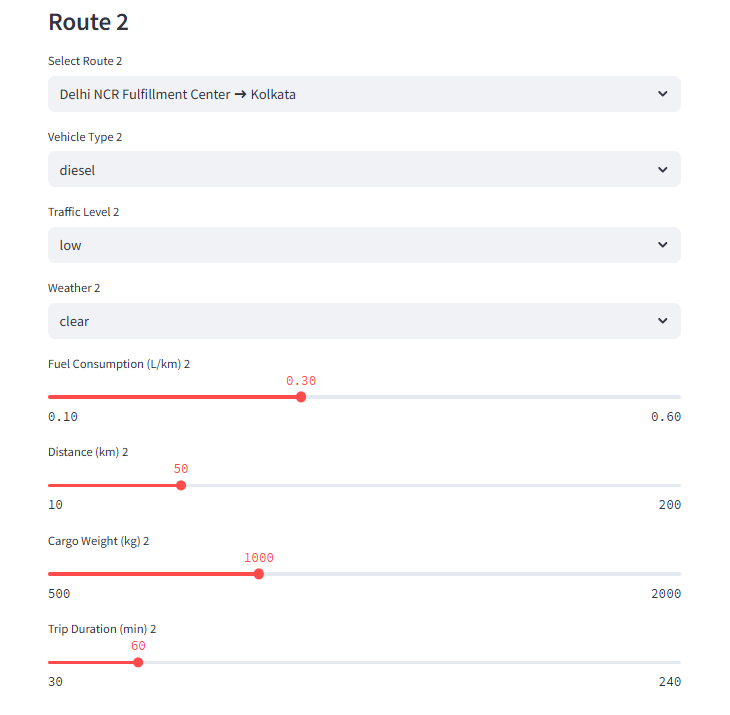


Figure :Dropdown for selecting Route, vehicle type, traffic level and weather

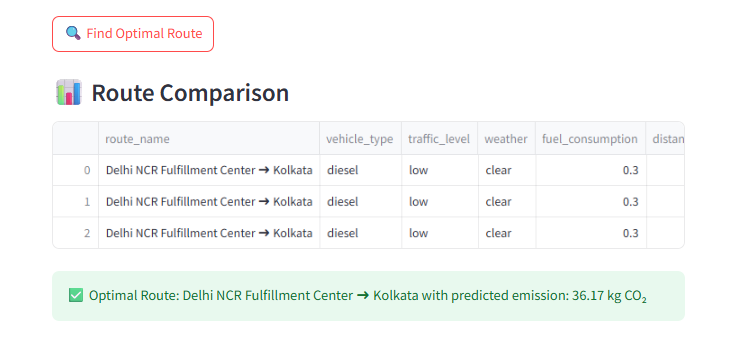


Figure : Output showing final analysis and Optimal Route

* GitHub link- [Carbon Footprint Optimization in Supply Chain Logistics](https://github.com/AnushkaPandey-52/carbon-footprint-optimization-in-supply-chain-logistics)
* Streamlit link- [Carbon Footprint Optimization in Supply Chain Logistics](https://carbon-footprint-optimization-in-supply-chain-logistics-fv8omq.streamlit.app/)