User Guide

Functionality #1

Input the number of passengers, the distance of the trip, and the amount of luggage you have for the trip. Then, enter how important each category is to you, on a scale of one to five. The program uses this information to compare vehicle options and will generate the best overall vehicle type for your situation, along with a few alternative options.

Functionality #2

Enter a body style, drivetrain configuration, and your assumption about the fuel efficiency of your combination. Using actual data, the program will analyze vehicles that match your input and determine their average fuel efficiency. It will also compare this group to others to show whether your assumption is accurate or not.

Functionality #3

Enter the body style, number of cylinders, and engine displacement (in liters) into their respective boxes. The program will use real data patterns to estimate the vehicle's city and highway fuel efficiency (in miles per gallon) and display the predicted results.

Functionality #4

Enter the distance you plan to travel and the total budget you have for the trip. Instead of selecting a vehicle first, the program works in reverse and calculates the maximum number of passengers and amount of luggage that can be supported within your budget for that distance.

Documentation on page 2

Documentation

Design

The basic design of our project is to use separate classes for data handling, logic, and user interface in Vaadin. The core of our project contains several main components. We have a DataLoader class which handles reading and cleaning the data from our CSV file. Next, we have a Vehicle Java class which represents each vehicle as an object with attributes such as MPG, class, and displacement, along with additional calculated values used throughout the program. We also have a TripRequest class which takes the user's inputs and stores them so they can be used across the program. After that, we built a VehicleRecommender class which handles filtering vehicles and applying scoring logic to determine the best matches. We also implemented a VehicleAnalysis class to support dataset-level comparisons and summary statistics. We created a RegressionModel class to handle the calculations and training for the regression model. It also holds the logic to calculate MSE and R^2 for both highway and city mpg. Also, we have a BudgetTripPlanner class which takes user input for the trip distance, along with the budget. It calculates the estimated fuel cost for the journey, along with number of passengers and amount of luggage that can be taken on the trip. Finally, we have the UI layer using Vaadin, which currently uses a mix of ComboBoxes and text inputs to gather user input and buttons to trigger the different functionalities and display results.

Technical Decisions

We chose to store the dataset using an ArrayList of Vehicle objects because it allows for simple iteration, filtering, and compatibility with our overall program structure and Vaadin interface. Instead of relying on advanced libraries or data structures, we kept the implementation straightforward using loops and conditionals to keep it easy to follow/readable and alignment with our goals. We also made the decision to clean and validate data during loading to avoid issues later in calculations. For user preferences, we implemented a weighted scoring system where each category (fuel efficiency, comfort, range, and speed) is scaled and normalized so that no single category dominates due to magnitude differences. Additionally, we separated the recommendation logic and analysis logic into different classes to clearly distinguish between decision-making functionality and statistical comparison functionality. For regression, we chose to implement a manual solution instead of using an external library, allowing us to keep the implementation consistent with the rest of the project.

Models

Our project uses three primary models corresponding to the first three functionalities. The first model is a weighted decision model that filters vehicles based on passenger count and luggage constraints, then ranks good options using a weighted scoring system. This scoring system incorporates trip-adjusted MPG, estimated range, comfort (approximated from vehicle class), and speed (approximated from engine displacement and cylinders). The second model is an analysis model that groups vehicles by body style and drivetrain and computes average, minimum, and maximum MPG. It compares these values across groups and against the overall dataset to show whether common assumptions about fuel efficiency are supported. The third model is a multiple linear regression model that predicts city and highway MPG based on vehicle class, number of cylinders, and engine displacement. These inputs are first converted into numeric form and normalized using z-score normalization, and then regression coefficients are computed using the normal equation. The final functionality extends the first model by reversing the logic, using distance and budget constraints to estimate the maximum passenger and luggage capacity for a trip.