**Car Dheko - Used Car Price Prediction**

**Project Overview**

**Car Dheko** is a Streamlit application designed to predict the price of used cars based on various features such as mileage, power, engine size, and more. This tool leverages machine learning models to provide accurate price estimates for users looking to buy or sell used cars.

**Dependencies**

The application relies on the following Python libraries:

* streamlit
* pandas
* numpy
* pickle
* scikit-learn

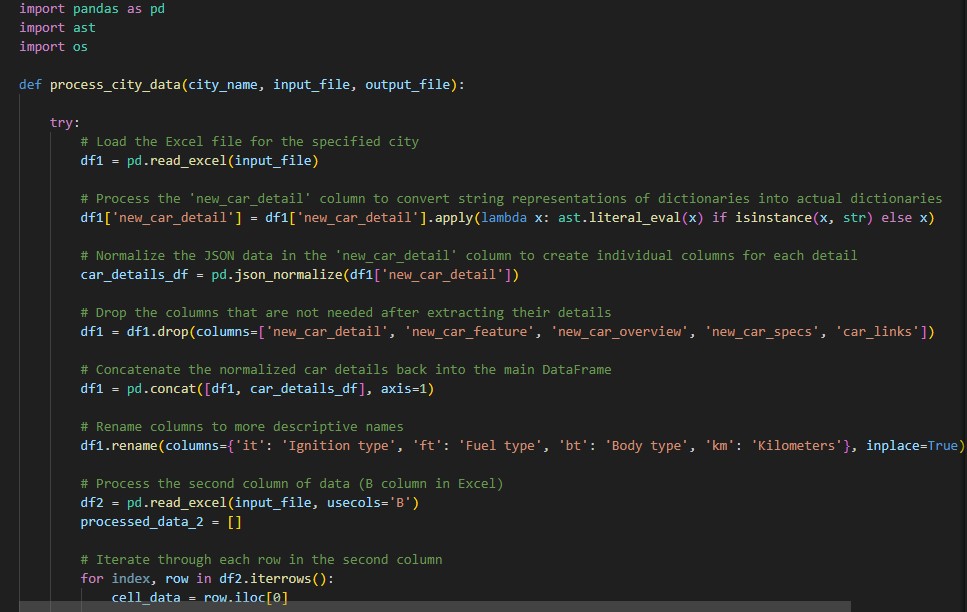
**Features**

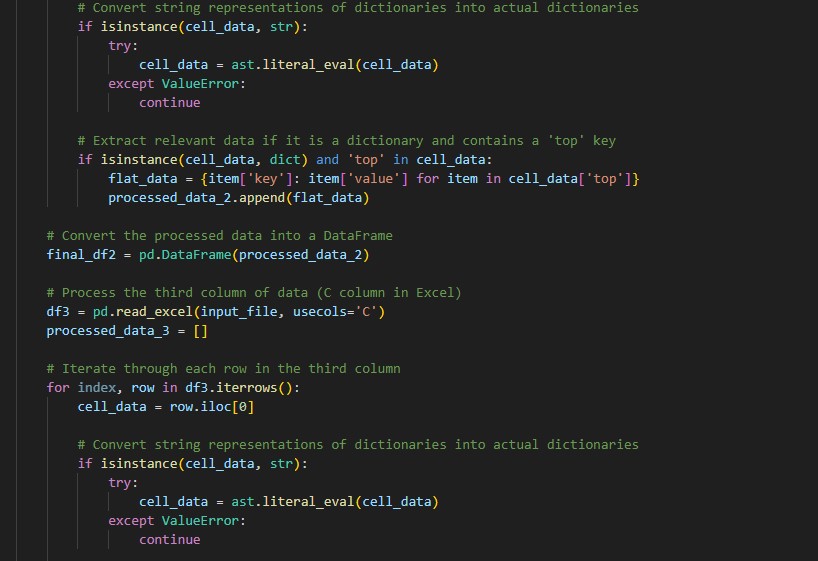
* **Input Fields**: Includes fields for mileage, max power, kilometres driven, engine size, wheel size, number of seats, ownership, fuel type, body type, transmission, OEM, insurance validity, and city.
* **Price Prediction**: Displays the predicted price of the used car based on input features.
* **Data Handling**: Cleans and preprocesses input data to align with the trained model requirements.

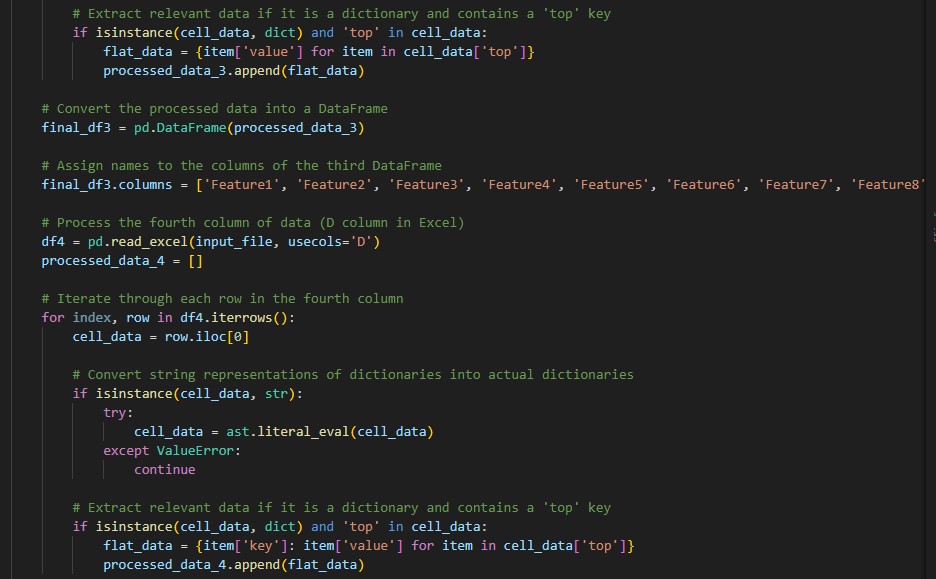
**Unstructured to structured data format:**

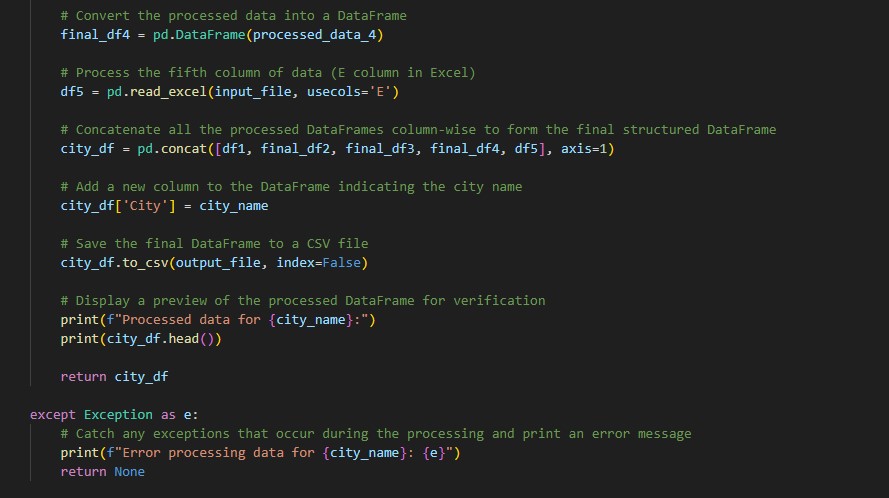
**Overview**

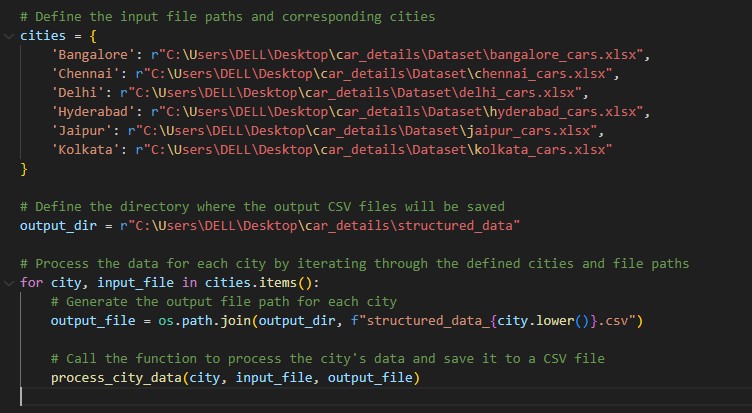
* This script processes car data from multiple Excel files, transforms it into a structured format, and saves it as CSV files. The code is divided into a function that handles the data processing and a section that iterates over multiple cities to apply this function**.**
* This is efficiently processes and structures data from multiple Excel files, transforming nested dictionary data into a clean, structured format suitable for further analysis. It includes robust error handling and supports data processing for multiple cities, saving the results in a specified directory









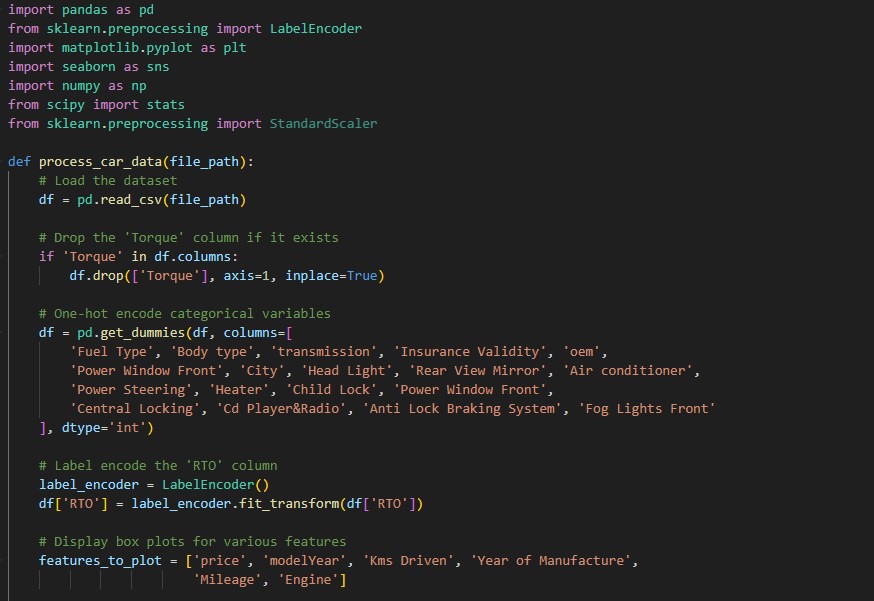


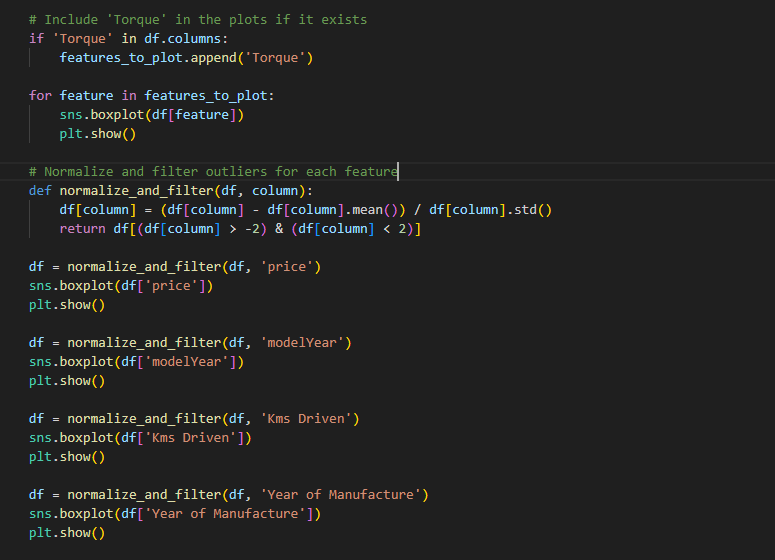
**EDA Process:**

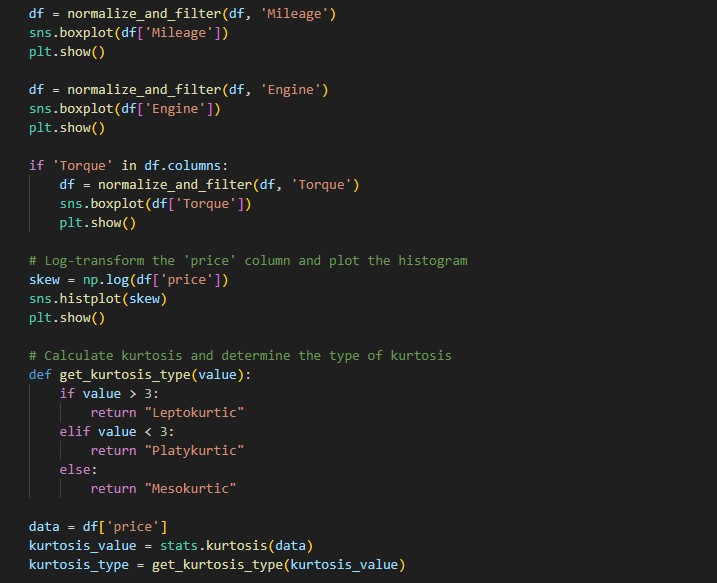
**Overview**

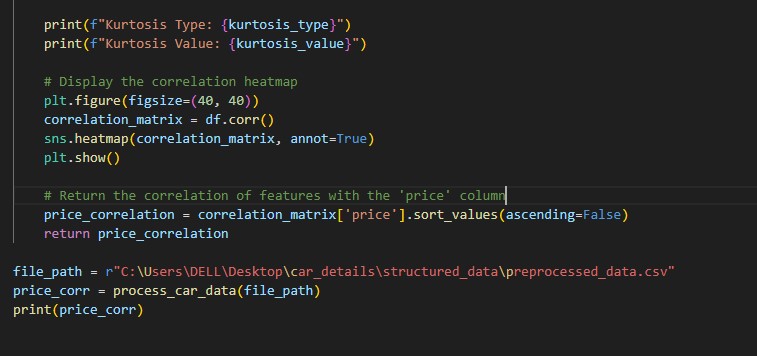
This script processes a car dataset, handles categorical variables, normalizes features, visualizes data distributions, identifies and filters outliers, and calculates correlations with the car prices. It provides a comprehensive exploratory data analysis (EDA) to help understand the relationships between different features and car prices.

* Encoding categorical variables.
* Normalizing numerical features.
* Filtering outliers.
* Visualizing feature distributions.
* Log-transforming skewed data.
* Calculating and classifying kurtosis.
* Displaying feature correlations.
* This EDA process is critical for understanding the data before applying machine learning models, ensuring that the data is clean, well-prepared, and ready for further analysis or modeling.









**Model Training:**

**Overview**

This script processes and cleans a car dataset, followed by training and evaluating multiple regression models to predict car prices. The script is structured into three main functions: clean\_and\_preprocess\_data, train\_and\_evaluate\_models, and main. It also includes saving the best-performing model (Random Forest) using the pickle module.

**Import Libraries**

* pandas: Used for data manipulation and analysis.
* re: Provides regular expressions for string matching and manipulation.
* numpy: Provides support for numerical operations and data structures.
* sklearn: Provides tools for model training, evaluation, and preprocessing.
* pickle: Used for saving and loading trained machine learning models.

**Define the Models**

* A dictionary of different regression models is defined, including linear models (Linear Regression, Lasso, Ridge), tree-based models (Decision Tree, Random Forest, Gradient Boosting), and a K-Nearest Neighbors model.

**Train and Evaluate the Models**

* Each model is trained on the training data (x\_train, y\_train).
* Predictions are made on both the training and testing data.

**Evaluation Metrics**

* **Mean Squared Error (MSE)**: Measures the average squared difference between the predicted and actual values.
* **Mean Absolute Error (MAE)**: Measures the average absolute difference between the predicted and actual values.
* **R-squared (R²)**: Indicates the proportion of variance in the dependent variable that is predictable from the independent variables.

**This code is designed to**

1. **Clean and preprocess** a car dataset, converting relevant columns to numeric types, handling missing values, and one-hot encoding categorical variables.
2. **Train and evaluate** various regression models on the pre-processed data, comparing their performance using metrics like MSE, MAE, and R².
3. **Save** the best-performing model ( Random Forest) for future use.
4. This approach is useful for building a robust car price prediction model by leveraging multiple regression techniques.

**Streamlit:**

**Overview**

This code implements a Streamlit application for predicting the price of a used car. The app uses a machine learning model (Random Forest) that has been trained on a dataset of car features and prices.

**1. Imports and Setup**

* **Libraries**:
  + pandas for data manipulation.
  + pickle for loading the trained model.
  + numpy for numerical operations.
  + streamlit for creating the web application interface.
* The application is designed to run within Streamlit, which is a framework for creating interactive web apps in Python.

**2. Data Loading and Preprocessing (load\_data)**

* **Loading the Data**:
  + The function loads the dataset from a specified file path using pandas.read\_csv.
* **Dropping Unnecessary Columns**:
  + Certain columns in the dataset that are not relevant for the prediction task are dropped.
* **Cleaning and Formatting**:
  + The price column is cleaned by removing non-numeric characters and converting values (e.g., converting "Lakh" and "Crore" to numeric values).
  + Other columns such as Mileage, Max Power, Kms Driven, and Engine are also cleaned and formatted for consistency.
* **One-Hot Encoding**:
  + Categorical variables are converted into binary columns (one-hot encoding) to be used by the machine learning model.
* The function returns the cleaned and processed DataFrame.

**3. Model Loading (load\_model)**

* **Loading the Trained Model**:
  + The function loads a pre-trained Random Forest model from a pickle file (random\_forest.pkl).
* The model is then returned for use in predictions.

**4. Input Data Preprocessing (preprocess\_input\_data)**

* **Input Formatting**:
  + The function converts user inputs into a DataFrame that matches the structure expected by the model.
* **One-Hot Encoding**:
  + Similar to the dataset preprocessing, the input data is one-hot encoded to align with the model’s training.
* **Handling Missing Columns**:
  + The function ensures that any columns expected by the model but missing in the input data are added with a default value of 0.
* The function returns the processed input DataFrame, ready for prediction.

**5. Price Formatting (format\_price)**

* **Price Display**:
  + This function converts a numeric price into a more human-readable format, displaying the price in "Lakhs" or "Crores" depending on its value.

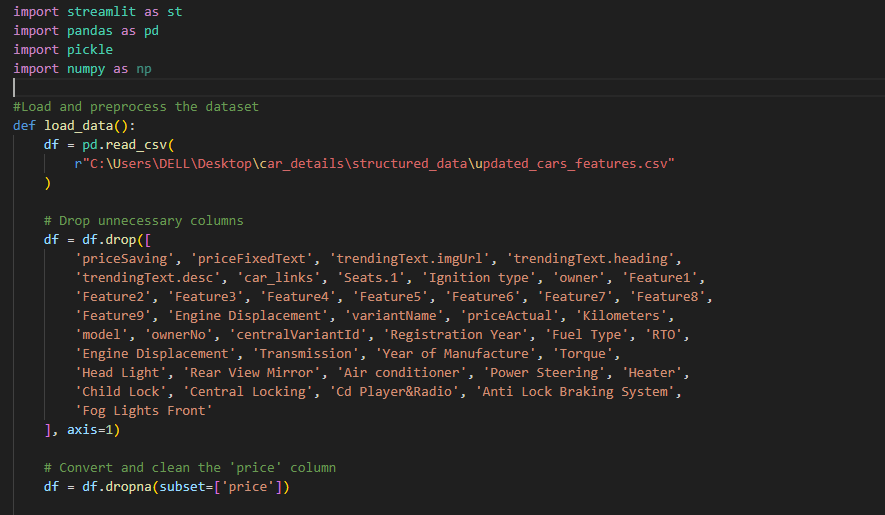
**6. Streamlit Application Interface (main)**

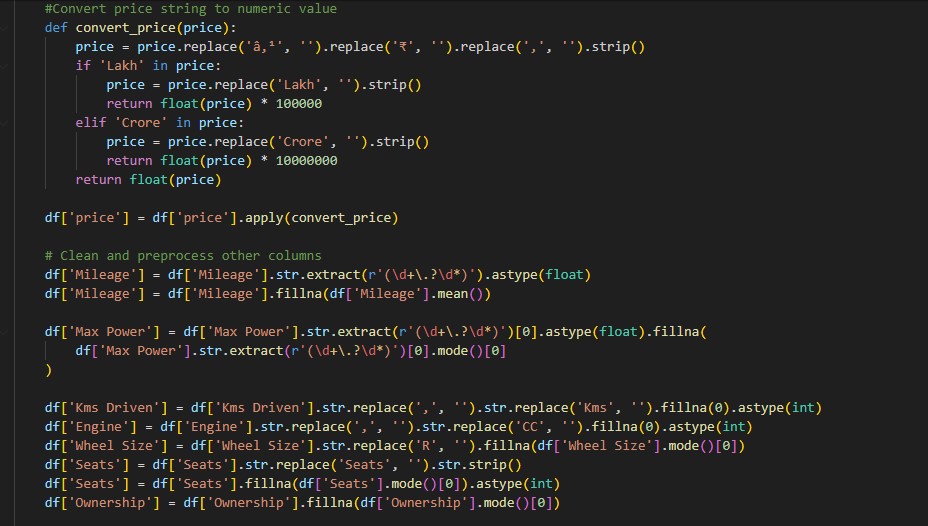
* **Title and Description**:
  + The app displays a title ("Car Dheko Used Car Price Prediction") and a brief description.
* **Image Display**:
  + An image (e.g., a logo) is displayed at the top of the app.
* **Sidebar for Input**:
  + The app uses the sidebar to collect input from the user about the car’s features (e.g., Mileage, Max Power, Kms Driven, Engine, etc.).
  + The user selects various options from dropdown menus (e.g., Fuel type, Body type, transmission, etc.).
* **Prediction Button**:
  + When the user clicks the "Predict Price" button, the app processes the input data, uses the loaded model to predict the price, and displays the result in a formatted manner.

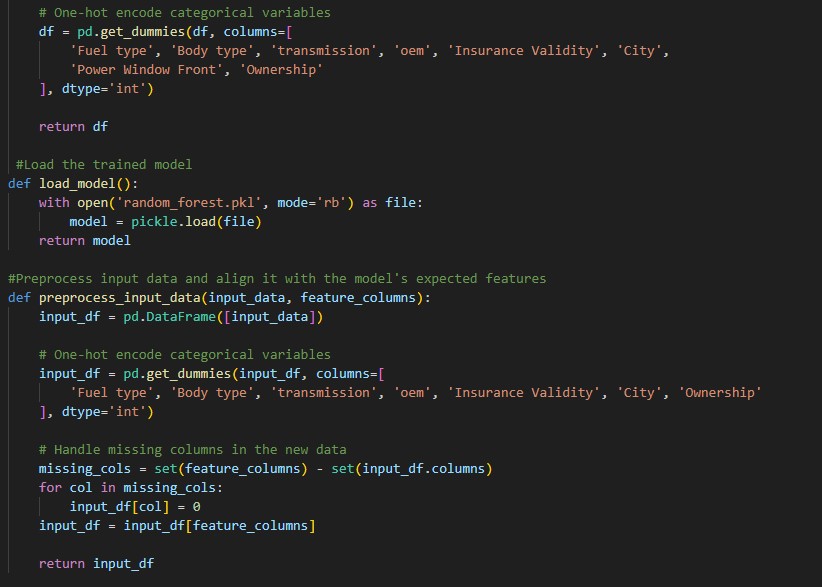
**7. Execution**

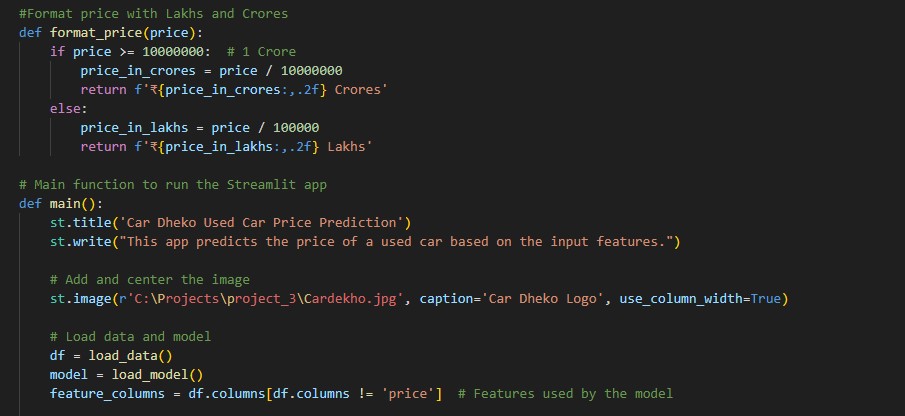
* The main function is executed when the script runs, setting up the Streamlit interface and handling the prediction logic.

The app provides an interactive interface where users can input the features of a used car, and it predicts the price based on a pre-trained Random Forest model. The interface is user-friendly, and the code handles both data preprocessing and prediction, ensuring that the input data is correctly formatted for the model.

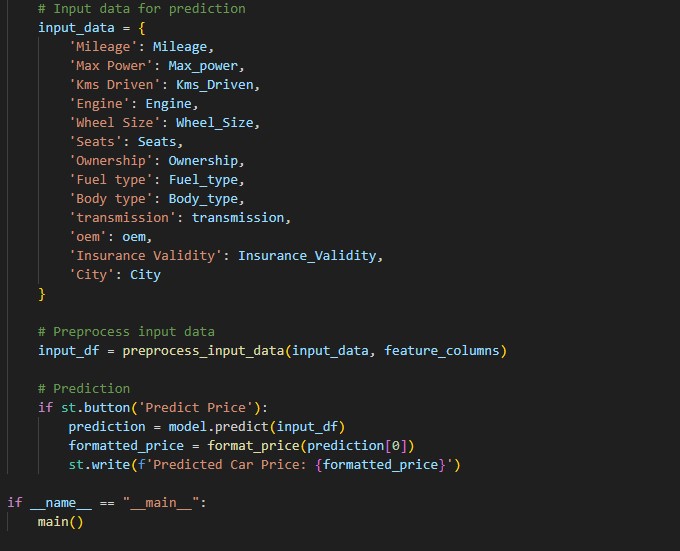












**GitHub Link:**