Car Price Prediction Project

Introduction

Car prices are influenced by various factors such as brand, model, fuel type, year of purchase, and kilometers driven. Predicting car prices based on these parameters can assist buyers and sellers in making informed decisions. This project utilizes machine learning techniques to develop a predictive model for estimating car prices based on relevant features extracted from a dataset of used cars.

Objectives

1. Develop a machine learning model to predict car prices accurately.
2. Provide an intuitive and user-friendly web interface for users to input details and obtain predictions.
3. Implement data cleaning and preprocessing techniques to prepare the dataset for model training.
4. Ensure scalability and usability through Flask-based web deployment.

Methodology

Data Cleaning and Preprocessing

1. **Dataset Overview**:
    - The dataset (quikr_car.csv) contains information about used cars, including attributes such as name, company, year, price, kilometers driven, and fuel type.
2. **Cleaning Steps**:
    - **Year Column**:
        - Removed non-numeric entries and converted the data type to integers.
    - **Name Column**:
        - Limited the name to the first three words for standardization.
    - **Price Column**:
        - Removed entries like "Ask for Price" and converted the column to numeric by removing commas.

- **Kilometers Driven**:
    - Removed non-numeric characters and converted the column to integers.
- **Fuel Type**:
    - Removed rows with missing values.
- **Outlier Removal**:
    - Filtered out entries where the price exceeded 8 million.

3. **Deduplication and Resetting Index**:
    - Removed duplicate entries and reset the index for consistency.

Model Development

1. **Feature and Target Selection**:
    - Features: name, company, fuel_type, year, and kms_driven.
    - Target: Price.
2. **Data Splitting**:
    - Split the dataset into training and testing sets with an 80:20 ratio.
3. **Pipeline Creation**:
    - Used OneHotEncoder to encode categorical variables (name, company, and fuel_type).
    - Combined the encoder and the regression model into a pipeline using make_pipeline.
4. **Model Training and Evaluation**:
    - Utilized LinearRegression as the predictive model.
    - Iteratively evaluated the model using r2_score over 1000 random seeds to select the best-performing split.
5. **Model Persistence**:
    - Saved the trained model as a LinearRegressionModel.pkl file using the pickle module for future use.

Web Application

1. **Framework**:
    - Used Flask to build a web application for car price prediction.
2. **Frontend**:
    - Built a responsive and user-friendly interface using HTML, CSS (via Bootstrap), and JavaScript.
    - Dynamic dropdown menus for company and car model selections.
3. **Backend**:
    - Loaded the pre-trained model and served predictions based on user inputs.
    - Routes:
        - /: Rendered the main interface with input forms.
        - /predict: Processed user inputs and returned predicted prices.
4. **JavaScript Integration**:
    - Implemented dynamic updates for car models based on the selected company.
    - Managed asynchronous requests to the backend using XMLHttpRequest for predictions.

Tools Used

1. **Python Libraries**:
    - pandas: Data manipulation and cleaning.
    - numpy: Numerical computations.
    - sklearn: Machine learning model development.
    - pickle: Model serialization.
2. **Web Development**:
    - **Flask**: Backend framework.
    - **HTML/CSS**: Frontend structure and styling.
    - **Bootstrap**: Responsive design.
    - **JavaScript**: Frontend interactivity.

3. **Environment**:
    - Jupyter Notebook: Exploratory data analysis and model training.

Conclusion

This project successfully combines machine learning with web development to provide a practical solution for car price prediction. The trained model demonstrated satisfactory performance, and the web interface ensures that users can interact with the model seamlessly. Future enhancements could include:

1. Incorporating additional features such as location and previous ownership.
2. Experimenting with advanced models like Gradient Boosting or Neural Networks for improved accuracy.
3. Enhancing the frontend with real-time validations and richer visualizations.