Used Car Price Prediction Project

Introduction

This project focuses on enhancing the car-buying experience and optimizing pricing for used vehicles by building a machine learning model. Based on historical data, this model predicts prices by considering variables such as make, model, year, fuel type, and transmission type. The goal is to enable accurate price estimation for used cars and integrate it into an interactive web application using **Streamlit** and **Django**. This application allows users and sales representatives to input car details and obtain real-time price predictions, simplifying decision-making in the car market while providing a seamless and user-friendly interface.

Approach

1.Data Import and Wrangling

- **Dataset Loading**: Imported datasets from various cities in unstructured formats (e.g., JSON-like entries) from Excel files using libraries like pandas and flattened them into structured dataframes using pandas.json_normalize().

- **Data Merging:** Added a new column, "City," to identify the origin of each dataset and combined all datasets into a unified structure using pandas.concat().

- **Data Export:** Saved the unified dataframe as a CSV file to streamline access for model training.

2. Data Cleaning

- **Handling Missing Values:** Addressed missing data by removing incomplete entries with pandas.dropna().

- **Symbol and Unit Removal:** Removed symbols like ₹, Lakh, and kmpl and reformatted values for numerical processing.

3. Data Visualization and Exploratory Data Analysis (EDA)

- **Correlation Analysis:** Created a correlation matrix to identify relationships between features and price, revealing trends such as mileage and year impacting price.

- **Outlier Handling:** Identified and addressed outliers using the Interquartile Range (IQR) method to enhance model performance.

4. Feature Selection

- **Categorical Features:** Incorporated variables like fuel type, body type, brand, insurance validity, and transmission type.

- **Numerical Features:** Cleaned and standardized variables such as mileage, engine size, and seating capacity.

5. Encoding and Scaling

- **One-Hot Encoding:** Transformed categorical variables into numerical representations.

- **Standard Scaling:** Normalized numerical features to ensure no single feature dominated the model's learning process.

Model Development

1.Train-Test Split

Divided the dataset into training and test sets using a 70-30 ratio to ensure effective model evaluation.

2. Model Selection and Training

- Linear Regression: Used as a baseline model, incorporating Ridge and Lasso regularization techniques to mitigate overfitting.

- Gradient Boosting Regressor (GBR): Utilized to capture intricate relationships in the data by iteratively focusing on residual errors.

Decision Tree Regressor: Implemented for its ability to model non-linear relationships, with pruning applied to control tree depth and prevent overfitting.

- Random Forest Regressor: Selected for final deployment due to its ensemble approach, which combines multiple decision trees and averages their outputs for enhanced prediction accuracy.

Evaluation Metrics

Model	MSE	RMSE	R^2
Linear Regression	4.9141	2.2167	0.6378
Decision Tree Regressor	2.8316	1.6827	0.7913
Random Forest Regressor	1.5058	1.2271	0.7913
Gradient Boosting Regressor	2.0081	1.4170	0.8890

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values**.**
- **Root Mean Squared Error (RMSE):** Highlights the prediction error in the same units as the target variable.
- **R^2 Score: Indicates the proportion of variance explained by the model.**

Selected Model: Random Forest Regressor

- **Hyperparameter Tuning**: Optimized parameters like n_estimators and max_depth using Grid Search.

Deployment

Streamlit Integration:

- Developed an interactive web application using Streamlit, enabling users to input car details and instantly receive predictions based on the trained Random Forest model.

- The app offers a simple and intuitive interface for real-time interactions.

Django Backend:

- Integrated Django to manage user authentication, database operations, and overall application logic.

- Developed REST APIs to connect the backend with the Streamlit front-end, enabling seamless data transfer and interaction.

Conclusion and Project Impact

Deploying the model via a Streamlit app enhances customer experience by providing quick, data-driven price estimates. This tool aids decision-making for both customers and sales representatives, offering instant insights. The integration with Django ensures a seamless backend for smooth operation. Future enhancements, like incorporating market trends and additional data, can further improve accuracy and user satisfaction, making the tool more valuable and effective.
