Car Dekho - Used Car Price Prediction (Regression Model)

Project Report

By: Soundhararajan S

Introduction

This project aims to enhance customer experience and optimize the pricing process for used cars by developing a machine learning model. The model leverages historical car price data and incorporates various features such as make, model, year, fuel type, and transmission type.

The primary objective is to accurately predict the prices of used cars and integrate the model into an interactive web application using **Streamlit**. This application allows customers and sales representatives to input car details and receive real-time price predictions, improving the efficiency of the car buying and selling process.

A. Approach

1. Data Preparation and Processing

Importing and Cleaning Data

- Load datasets from multiple cities (often in unstructured formats like text, JSON-like entries, or Excel files).
- Utilize **pandas** to import, process, and standardize datasets.
- Parse JSON-like data using ast.literal_eval() and normalize nested structures with pandas.json_normalize().
- Merge datasets from all cities into a single, structured dataframe.
- Add a **'City'** column to differentiate records from different locations.

Handling Missing Values and Data Cleaning

- Use dropna() to remove rows/columns with missing data.
- Remove currency symbols (₹, Lakh, Crore) and units (kmpl, CC).
- Eliminate unnecessary whitespace and commas.
- Convert textual numerical values into machine-readable formats.

2. Exploratory Data Analysis (EDA)

- **Correlation Matrix**: Used heatmaps to identify relationships between features and price (e.g., model year and mileage).
- **Outlier Detection**: Applied the **Interquartile Range (IQR) method** to remove price anomalies that could impact model accuracy.

3. Feature Engineering

Categorical Features

- Fuel Type, Body Type, Brand, Insurance Validity, Color, City, and Transmission.

Numerical Features

- Owner Count, Model Year, Kilometers Driven, Mileage, Engine Capacity, and Number of Seats.

Encoding and Scaling

- **One-Hot Encoding**: Converted categorical variables into numerical representations.
- **Standard Scaling**: Ensured all numerical features were normalized to prevent dominant influences.

B. Model Development

1. Train-Test Split

- The dataset was split into **training (70%)** and **testing (30%)** subsets.

2. Model Selection

i) Linear Regression

- Selected as a baseline model due to its simplicity and interpretability.
- Applied **Ridge** and **Lasso** regression to mitigate overfitting.

ii) Gradient Boosting Regressor (GBR)

- Used ensemble learning to fit new models to residual errors.
- Helps improve predictive accuracy by sequentially refining weak learners.

iii) Decision Tree Regressor

- Chosen for its ability to model non-linear relationships.
- Applied **pruning** to prevent overfitting.

iv) Random Forest Regressor

- An ensemble of multiple decision trees, averaging predictions to enhance accuracy.
- Utilized **bootstrap aggregation (bagging)** for model stability.

- Optimized feature selection to improve generalization.

3. Model Evaluation

The models were evaluated using:

- **Mean Squared Error (MSE)**: Measures the average squared difference between actual and predicted values.
- **Mean Absolute Error (MAE)**: Measures average absolute differences.
- **R^2 Score**: Represents the proportion of variance explained by the model.

Model Performance Comparison

Model	MAE	MSE	RMSE	R^2
Linear Regression	1.93	5.75	2.39	-4.83
Decision Tree Regressor	1.06	2.87	1.69	0.76
Random Forest Regressor	0.83	1.73	1.31	0.85
Gradient Boosting Regressor	1.04	2.23	1.49	0.81
Ridge Regressor	2.58	1.12	1.61	0.78
Lasso Regressor	2.59	1.12	1.61	0.78

4. Results

- **Random Forest Regressor** achieved the best performance with the highest R^2 score and lowest MSE/MAE.
- **Hyperparameter Tuning**: Applied **Grid Search** to optimize parameters such as n_estimators and max_depth.

5. Model Pipeline

- **Modular Structure**: Separates data preprocessing, training, and prediction to improve maintainability.
- **ColumnTransformer Usage**: Applies different transformations to categorical and numerical features simultaneously.
- **Ensures Reproducibility**: Guarantees consistent preprocessing during both training and inference.

C. Model Deployment - Streamlit

1. Features of the Application

i) User Input Interface

- A user-friendly interface where customers enter car details (Make, Model, Year, Fuel Type, Transmission, etc.).
- Interactive elements like dropdowns and sliders minimize user errors.

ii) Real-Time Price Prediction

- The **Random Forest model** predicts the price instantly based on user input.

iii) Backend Implementation

- The trained **Random Forest model**, **StandardScaler**, and **LabelEncoder** are loaded using the **pickle** library.
- Ensures preprocessing is consistent with training data.

D. Conclusion & Impact

- The **Streamlit application** improves user experience at **Car Dekho** by offering **instant and reliable price predictions**.
- Customers gain **data-driven insights**, facilitating better buying and selling decisions.
- Sales representatives benefit from **standardized valuations**, reducing pricing discrepancies.
- **Future enhancements**: Incorporating **real-time market trends** and **personalized recommendations** to refine predictions.

Final Thoughts

This project effectively bridges **machine learning** and **real-world business applications** by transforming car valuation into a **data-driven, automated**, and **efficient** process. With further iterations, the model can become even more precise and intelligent, contributing significantly to the used car industry.
