Car Dheko - Used Car Price Prediction

Project Report

By

Durga Kannan

Objective:

The goal of this project is to develop a machine learning model that predicts the prices of used cars based on key attributes such as make, model, year, fuel type, and transmission. This model will be integrated into an interactive **Streamlit** web application, enabling both customers and sales representatives to quickly input car details and receive an accurate price estimate, streamlining the pricing process and improving the overall user experience.

Approach:

Part A- Data Preparation

1. **Import files and Data wrangling:**

• Load the datasets from multiple cities, which are in unstructured format (likely messy, containing text, JSON-like entries, or other formats) from Excel files.

• Use libraries like pandas to import data from each city's dataset for further processing.

• For unstructured data that includes JSON-like structures (e.g., dictionaries or lists embedded as strings), use ast.literal_eval to safely evaluate these strings into Python objects.

•Once JSON-like data is parsed, use pandas.json_normalize() to convert nested JSON objects into a flat, structured dataframe.

• This ensures that all nested values are properly spread across multiple columns, creating a clean and organized structure.

• After converting each city's dataset into a structured format, add a new column named ‘City’ and assign the respective city name as the value for all rows in the dataset.

• Use pandas.concat() to merge the structured datasets from all cities into a single dataframe.

• Ensure that columns are aligned and duplicate entries are handled nto create a unified dataset for model training and save dataframe to csv.

2. **Handling Missing Values and Data Cleaning:**

• Use pandas.dropna() to remove rows or columns containing missing data

• Remove symbols and units (₹, Lakh, Crore,kmpl,CC),Eliminate commas and whitespace

• Convert the Values to a Numerical Format suitable for machine learning models.

3. **Data Visualization:**

Exploratory Data Analysis (EDA) was performed to examine the relationships between various features and the target variable (price). This analysis helped uncover key patterns and identify potential outliers.

- Correlation Matrix: A heatmap of the correlation matrix highlighted significant correlations between dependent and independent features such as modelYear and km with price.

- Outlier Detection: Outliers in the price column were detected using the Interquartile Range (IQR) method to prevent them from affecting the model's performance

4. Feature Selection:
   - Categorical Features: Attributes such as fuel type, body type, and Brand, Insurance Validity, Color, city, transmission.

   - Numerical Features: Variables are Ownerno, Model year, kms driven, Milage, Engine, Seats were cleaned and converted to integers.

5. Encoding and Scaling:

• OneHot Encoding is used to convert categorical data into numerical form, which is often required by machine learning models.

• Standard Scaler ensures that all numerical features are on the same scale, preventing certain features from dominating others.

B. Model Development

1. Train-Test Split:

Split the dataset into training and testing sets to evaluate model performance. Common split ratios are 70-25,80-20

2.Model Selection:

i) Linear Regression: • Overview: Linear Regression was chosen as the baseline model due to its simplicity and ease of interpretation. • Regularization: Ridge and Lasso regression were applied to prevent overfitting.

ii) Gradient Boosting Regressor (GBR): • GradientBoostingRegressor is a machine learning technique used for regression tasks. It is part of the ensemble learning family • Gradient Boosting is based on the idea of fitting new models to the residuals (errors) of the previous models.

iii) Decision Tree Regressor: • Decision Trees were chosen for their interpretability and capability to model nonlinear relationships. • Pruning was applied to prevent overfitting by limiting the tree depth.

iv) Random Forest Regressor • A Random Forest is an ensemble of decision trees. In the case of regression, each decision tree predicts a value, and the final prediction is the average of all the trees' outputs. • Each tree is trained on a different random subset of the data (using bootstrapping). • For each node in a tree, only a random subset of features is considered for splitting.

3. Model Evaluation:

The models were evaluated using the following metrics:

- **Mean Absolute Error (MAE)**: **MAE** represents the average magnitude of the errors in the model's predictions, without considering their direction. The **lower the MAE**, the better. A smaller MAE indicates that the model's predictions are close to the actual values.

- **Mean Squared Error (MSE)**: **MSE** penalizes large errors more than smaller ones by squaring the differences. The **lower the MSE**, the better the model is at predicting the correct price without large errors.

- **Root Mean Squared Error (RMSE)**: **RMSE** is the square root of MSE and provides an error value in the same units as the target variable, which makes it easier to interpret. Lower RMSE means better model accuracy.

- **R^2 (R-Squared)**: **R^2** measures the proportion of the variance in the dependent variable (price) that is predictable from the independent variables (features). A higher **R^2** score indicates that the model is more accurate in predicting the target variable.

Evaluation metrics (MAE, MSE, RMSE, R^2) for different machine learning models:

	model	MAE	MSE	RMSE	R2
0	LinearRegression	2.439626e+13	4.575641e+28	2.139075e+14	-3.424396e+27
1	DecisionTreeRegressor	1.045080e+00	2.917630e+00	1.708107e+00	7.816455e-01
2	RandomForestRegressor	7.856467e-01	1.612333e+00	1.269777e+00	8.793335e-01
3	GradientBoostingRegressor	1.134004e+00	2.637002e+00	1.623885e+00	8.026476e-01
4	RidgeRegressor	2.124322e+00	9.191905e-01	1.457505e+00	8.410164e-01
5	LassoRegressor	2.155362e+00	9.220470e-01	1.468115e+00	8.386934e-01

Key Insights from above comparison:

1. **LinearRegression** has extremely large MAE, MSE, RMSE, and negative R^2, suggesting it's not performing well for this dataset.
2. **DecisionTreeRegressor** has moderate performance with a **R^2 of 0.78**, indicating a good fit but not perfect.
3. **RandomForestRegressor** performs best with **R^2 of 0.88**, the lowest MAE, and good overall results across metrics.
4. **GradientBoostingRegressor** also performs well but slightly worse than RandomForest, with an **R^2 of 0.80**.
5. **RidgeRegressor** and **LassoRegressor** are fairly close in performance, with an **R^2 of 0.84**, suggesting they are similar in predictive power but still below RandomForest.

Results after comparison:

Random Forest:

• When comparing the models based on their R^2 scores, the **RandomForestRegressor** emerged as the best model with an R^2.

4.Hyperparameter Tuning:

After running **RandomizedSearchCV** on the **RandomForestRegressor** with a broad hyperparameter search space, the following optimal parameters were found:

- **n_estimators** (Number of trees in the forest): 300
- **max_depth** (Maximum depth of the trees): 30
- **min_samples_split** (Minimum samples required to split a node): 10
- **min_samples_leaf** (Minimum samples required at a leaf node): 1
- **max_features** (Number of features to consider for splitting): None
- **bootstrap** (Whether to use bootstrap samples): True

- RandomizedSearchCV was effective in optimizing the hyperparameters of the **RandomForestRegressor**, leading to a model that is both accurate and generalizes well to unseen data.

- The tuning process significantly improved the model's performance, increasing the cross-validation score and fine-tuning the model's ability to make accurate predictions without overfitting.

5.Pipeline

- **Modular Structure**: The pipeline is designed with a modular architecture, promoting clear separation between data preprocessing and model training. This structure improves code readability and maintainability, making it easier to modify individual components (e.g., preprocessing or model) without disrupting the overall workflow.

- **Data Preprocessing**: The pipeline handles data preprocessing in two distinct steps: one for numerical features and another for categorical features. Numerical features are standardized using the StandardScaler, while categorical features are encoded using a specified encoder. This approach ensures each feature type is processed appropriately, enhancing the model's performance.

- **ColumnTransformer**: The use of ColumnTransformer allows for efficient application of different preprocessing techniques to specific columns in the dataset. This ensures that multiple feature types can be processed simultaneously, streamlining data preparation and ensuring the model receives the correct input format.

- **Model Integration**: The Random Forest regression model is seamlessly integrated into the pipeline. This means the model is directly trained on the preprocessed data, creating a smooth flow from data input to predictions. It also simplifies the process of applying the same preprocessing steps to new data when making predictions.

- **Reproducibility and Consistency**: By encapsulating the entire workflow—from preprocessing to model training—in a single pipeline, the process ensures reproducibility. The same preprocessing steps are consistently applied during both training and inference, which is crucial for reliable predictions and model validation.

C. Model Deployement- Streamlit

Streamlit is an open-source Python library for quickly building interactive web applications for data science and machine learning, making it ideal for deploying models.

1. **Features of the Application**:

- **User Input Interface**:Users can input car details (make, model, year, fuel type, etc.) via an intuitive interface with drop-down menus and sliders to minimize errors.

- **Price Prediction**:The trained Random Forest model predicts the car's price based on user inputs, displaying the result instantly for a seamless experience.

2. **Backend Implementation**:

- **Model Loading**:The trained Random Forest model, StandardScaler, and LabelEncoder are loaded using pickle, making them ready for use in predictions.

- **Data Preprocessing**:User inputs are preprocessed similarly to the training data to ensure consistency and accuracy in predictions.

D. Conclusion - Project Impact and Future Potential

- **Enhanced User Experience**: Deploying the predictive model through the Streamlit application transforms the CarDekho platform by offering fast and accurate price estimates for used cars, significantly improving the user experience.

- **Empowering Customers**: The real-time pricing tool provides customers with data-driven insights, enabling them to make informed decisions when buying or selling vehicles.

- **Streamlined Sales Process**: For sales representatives, the tool simplifies car valuation, saving time and ensuring consistency in pricing across the platform.

- **Opportunities for Future Growth**: This deployment also lays the foundation for future innovations, such as personalized recommendations and integration with real-time market data, further enhancing the accuracy and sophistication of predictions.