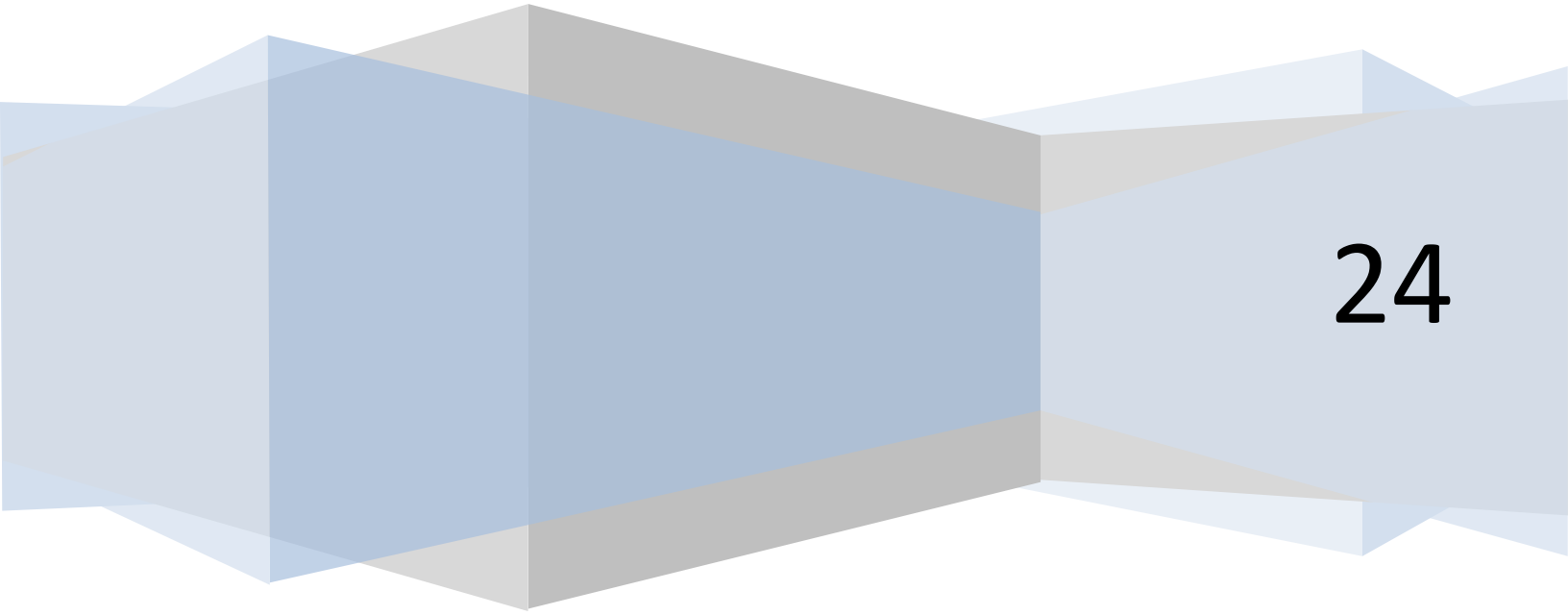
Capstone_3

CarDheko – Optimizing Used Car Price Predictions

(Regression Model Project Report)

Mohammed Riyaskhan



24

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 Streamlit web application, allowing users and sales representatives to input car details and obtain real-time price predictions, simplifying decision-making in the car market.

A. Approach

1. Data Import and Wrangling

- **Loading Datasets:** Import datasets from various cities in unstructured formats (e.g., JSON-like entries) from Excel files using libraries like `pandas`.
- **Data Parsing and Cleaning:** Parse JSON-like structures and use `pandas.json_normalize()` to flatten them into a structured dataframe.
- **Data Merging:** After structuring each city's data, a new column titled 'City' is added, and datasets are merged using `pandas.concat()` for a unified structure.

- **Data Export:** Save the unified dataframe as a CSV file for easier access in model training..

2. Data Cleaning

- **Missing Values**: Used `pandas.dropna()` to handle missing entries.
- **Symbol Removal**: Eliminated units (₹, Lakh, kmpl) and formatted values for numerical processing.

3. Data Visualization and EDA

- **Correlation Matrix**: Highlighted relationships between features and price to uncover trends, such as mileage and year affecting price.
- **Outlier Detection**: Detected and handled outliers using the IQR method to improve model accuracy.

4. Feature Selection

- **Categorical Features**: Included variables such as fuel type, body type, brand, insurance validity, and transmission.
- **Numerical Features**: Cleaned and standardized variables like mileage, engine size, and seats.

5. Encoding and Scaling

- **OneHot Encoding**: Converted categorical variables to numerical format.
- **Standard Scaler**: Normalized numerical data to prevent certain features from dominating the model.

B. Model Development

1. Train-Test Split

Split data into training and test sets with a 70-30 ratio for model evaluation

2. Model Selection and Training

i) **Linear Regression**: Applied as a baseline model, with Ridge and Lasso regularization to reduce overfitting.
ii) **Gradient Boosting Regressor (GBR)**: Employed to capture complex relationships by focusing on model residuals.
iii) **Decision Tree Regressor**: Used for interpretable, non-linear relationships, with pruning to limit depth.
iv) **Random Forest Regressor**: Chosen for final deployment due to its ensemble nature, averaging outputs for more accurate predictions.

3. Model Evaluation

Evaluated models based on Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 score

Evaluation Metrics:

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values.
- **Mean Absolute Error (MAE):** Provides clear prediction accuracy.
- **R^2 Score:** Indicates the variance explained by the model.

Model Performance Comparison

Model	MAE	MSE	RMSE	R2
LinearRegression	1935936589	5.75E+20	23976305799	-4.83E+19
DecisionTreeRegressor	1.055584958	2.87338617	1.695106536	0.758623737
RandomForestRegressor	0.825905429	1.728796798	1.314837176	0.854773954
GradientBoostingRegressor	1.043791703	2.229969888	1.49330837	0.812673352
RidgeRegressor	2.583592841	1.123174422	1.607355854	0.782967569
LassoRegressor	2.585866437	1.12115329	1.608062946	0.782776577

4. Selected Model - Random Forest Regressor

Hyperparameter Tuning: Used Grid Search to optimize parameters like `n_estimators` and `max_depth`.

C. Model Pipeline

- **Data Preprocessing**: Created a modular pipeline for both numerical scaling and categorical encoding.
- **Column Transformer**: Applied to preprocess distinct data types simultaneously, improving consistency.
- **Model Integration**: Integrated the Random Forest model, streamlining the workflow from input to prediction.

•

D. Deployment

Streamlit Application

Streamlit: Developed an interactive web app for users to input car details and receive instant predictions using the trained Random Forest model.

Application Features

5. Model Deployment with Streamlit

Using Streamlit, an open-source Python library, a custom web application was developed to make model predictions accessible:

1. **User Interface:**
    1. Drop-downs and sliders for easy data entry.
    2. Enables user input for key features like car make, model, year, and city.
2. **Price Prediction:**
    1. Provides real-time price predictions powered by the trained Random Forest model.
3. **Backend:**
    1. Utilizes `pickle` to load the model and preprocesses user input consistently with training data.

Conclusion and Project Impact

Deploying this model through a Streamlit app enhances customer experience by offering quick, data-driven price estimates. This tool aids decision-making for both customers and sales representatives, laying a foundation for potential future features like market data integration to boost accuracy and user satisfaction.

Thankyou!