car-dheko

Project Title:

Car Price Prediction using gradient boosting algorithm

1. Problem Statement

Predict the price of a car based on various features such as mileage, make, model, engine size, and year using a gradient boosting algorithm model.

2. Objective

To develop a gradient boosting algorithm regression model that predicts car prices based on historical sales data and car features.

---

3. Data Processing and extracting the data:

Overview:

The dataset contains car specifications in a nested JSON format, which has been transformed into a tabular format for easier analysis.

Extraction

3.1 Dataset Overview

- **Source:** Collected from various car listing datasets, from **car dheko**
- **Format:** The dataset is in CSV format, containing both numerical and categorical features related to car specifications and sales.

3.2 Features (Columns):

- **Numerical Features:**
  - `Year`: Year the car was manufactured.
  - `km`: Total miles the car has traveled.
  - `Ownerno`: Number of previous owners
- **Categorical Features:**
  - `bt`: The body type of the car (e.g., SUV, Hatchback).
  - `Model`: The specific model of the car (e.g., Corolla, Mustang).
  - `FuelType`: Type of fuel used by the car (e.g., Gasoline, Diesel, Electric).

3.3 Target Variable:

- **Price:** The selling price of the car (target variable for prediction).

---

4. Data Preparation

4.1 Converting Lists to DataFrames

- Car details were initially provided in list format. These lists were converted into pandas DataFrames to facilitate further processing.
  - Example:

```
car_list = [['Toyota', 'Corolla', 2015, 50000, 1.8],
['Ford', 'Mustang', 2018, 30000, 2.3]]
car_df = pd.DataFrame(car_list, columns=['Make', 'Mode
l', 'Year', 'Mileage', 'EngineSize'])
```

4.2 Converting Dictionaries to DataFrames

- Some data was provided as dictionaries, which were transformed into DataFrames for better accessibility and manipulation.
    - Example:

```
car_dict = {'Make': ['Toyota', 'Ford'], 'Model': ['Coro
lla', 'Mustang'], 'Year': [2015, 2018], 'Mileage': [500
00, 30000]}
car_df = pd.DataFrame(car_dict)
```

4.3 Data Cleaning

- **Missing Values:**
    - Numerical features like `Mileage` and `EngineSize` were filled with median values.
    - Categorical features like `Make` and `FuelType` were filled with the mode.

4.4 Feature Engineering

- **Encoding Categorical Variables:**
    - Used one-hot encoding for categorical features like `Make`, `Model`, and `FuelType`.

4.5 Data type conversion

- **EChange string to correct data type:**
    - Remove the unwanted carecters such as $, “,”,lack,cror from `price`, `gearbox`, and `km`.

4.6 Data Splitting

- **Train-Test Split:** The data was split into 80% for training and 20% for testing using `train_test_split()` from Scikit-learn.

---

5. Model Development

5.1 Model Selection

The **gradient boosting algorithm** algorithm was chosen for its ability to handle non-linear relationships and complex interactions between features, which makes it suitable for predicting car prices.

5.2 Model Training

The gradient boosting model was trained on the preprocessed data using the `gradient boosting algorithm` class from Scikit-learn.

5.3 Model Hyperparameters:

- n_estimators=300,
- random_state=42,
- learning_rate=0.2,
- max_depth=5,
- min_samples_leaf=4,
- min_samples_split=2

---

6. Model Evaluation

6.1 Metrics

The following regression metrics were used to evaluate the performance of the model:

- **Mean Absolute Error (MAE):** 67497
- **Mean Squared Error (MSE):** 10206216625
- **R-squared (R^2):** 0.92 (indicating that 92% of the variance in car prices is explained by the model).

6.2 Feature Importance

The top features influencing the car price prediction:

- `body type`: Most important feature.
- `Model year`: Significant impact on the price.
- `Transmision` and `KM`: Contribute to price variability.
- `Make`: Impact due to brand reputation.

7. Model Deployment

7.1 Deployment Method

- The model is deployed as a **Streamlit** web app, allowing users to input car features and get price predictions

8. Future Improvements

- Experiment with hyperparameter tuning (e.g., GridSearchCV) to further optimize model performance.
- Find the best parameter using grid search and build the model with the best parameter.

9. References

- Dataset Source:

  > cardheko

- Scikit-learn Documentation: Scikit-learn

Appendix

- **Model Artifacts:** The trained gradient boosting model and the preprocessor pipeline are saved as `.pkl` files.
- **Code Repository:** GitHub link to the project repository

model evaluation:

Model	MAE	MSE	R2

Linear model without outlier deduction	513927.97	815738600808.37	0.408650677001698
Linear model with outlier deduction	172222.18	5327287423 1.13	0.569453452865484
RandomForestRegressor	70480.07	12308297292.55	0.9017057596589749
Gradient Boosting model	67283.50	10587591749.03	0.915447339036797
DecisionTreeRegressor	91627.87	20550866984.03	0.8358804787916038

After Grid Search:

Model	MAE	MSE	R2
RandomForestRegressor	91964.03	18903219637.25	0.8536942703379551
Gradient Boosting model	67497.53	10206216625.31	0.921006685680578