Car Dataset: Used cars data from Bikroy.com

Paromita Saha , Sumaiya Nasrin [†], Rimbe Dey [†],
2104010202333 , 2104010202336 , 2104010202340

Couse title- Machine Learing Laboratory .
Couse Code-458 .
Date- 23 November 2025 .

[†]These authors contributed equally to this work.

Abstract

This study looks at developing a machine learning model to predict the selling price of used cars in Bangladesh using the "Used Cars Data from Bikroy.com" dataset available on Mendeley Data. The dataset includes important attributes like brand, model, year of manufacture, mileage, engine capacity, condition, and seller information. The user-generated ads often had inconsistencies and missing values, so we had to take significant preprocessing steps. These steps included handling missing data, encoding categorical variables, removing outliers, normalizing numerical features, and selecting features based on correlation analysis.

We applied several machine learning models, including Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, and XGBoost, to the cleaned dataset. We tuned the hyperparameters using GridSearchCV to improve model performance. The Random Forest model achieved the best results, showing high prediction accuracy and low error scores compared to the baseline model. We used evaluation metrics such as RMSE, MAE, and R^2 to assess the performance of all trained models.

The results show that ensemble learning methods work well for modeling the nonlinear relationships found in used car price data. This study highlights the potential of data-driven price prediction models for online marketplaces. It leads to better transparency, fair pricing, and automated valuation systems. Future work may include integrating deep learning methods, scraping real-time ads, and deploying the model as a web-based valuation tool.

1 Introduction

The used-car market in Bangladesh has seen significant growth with the rise of digital marketplaces like Bikroy.com. Buyers and sellers increasingly depend on online ads to determine vehicle values. However, inconsistent pricing, subjective descriptions, and varied vehicle conditions make it hard to find a fair market price for a used car. As a result, individual buyers and dealerships often struggle to make accurate valuation decisions. Machine learning provides a strong solution for modeling complex pricing patterns based on historical data. By examining real-world used-car listings, a predictive model can learn how different attributes, such as brand, year, mileage, engine capacity, condition, and vehicle type, influence the final price. This model can help with automated price recommendations, fraud detection, and fair negotiations in online marketplaces. In this project, the dataset from Mendeley Data is used to create a price prediction system specifically designed for the Bangladeshi automotive market. The dataset includes real listings from Bikroy.com, making it perfect for practical machine learning applications.

2 Problem Statement

The used-car market in Bangladesh has grown quickly as online platforms like Bikroy.com have become key places for buying and selling vehicles. This shift to digital has made it easier for people to access the market, but it has also created new problems in figuring out a fair price for used cars. Unlike standardized dealer evaluations, online listings can differ greatly because sellers often provide inconsistent information, including incomplete details, vague condition descriptions, unrealistic prices, and incorrect mileage.

As a result, buyers often find it hard to determine if a listed price is reasonable, while sellers may undervalue their cars because they lack reliable pricing references. This uncertainty highlights the need for a pricing model based on data that can look at past listings and automatically estimate fair market values based on vehicle features.

The dataset for this project, titled “Used Cars Data from Bikroy.com” includes real advertisement data. It contains features such as brand, model, manufacturing year, mileage, engine capacity, condition, fuel type, and price. However, the dataset also presents common issues found in user-generated online data, such as missing values, inconsistent formats, and outliers. Building an accurate predictive model requires careful preprocessing steps along with suitable machine learning methods. The main goal of this project is to create a machine learning model that can predict the selling price of a used car in the Bangladeshi market. By learning from actual online listings, the model aims to deliver a reliable, automated valuation system that can help buyers, sellers, and online platforms make better decisions.

3 Related Work

Car price prediction has been a popular topic in machine learning because vehicle data is structured and there is a growing need for automated valuation tools. Early research often used basic models like Linear Regression and Decision Trees. These models were easy to understand but had difficulty capturing the complex, nonlinear relationships in pricing data. Over time, researchers realized that ensemble methods, such as Random Forest, Gradient Boosting Machines (GBM), and XGBoost, greatly improved accuracy by combining multiple decision trees and addressing complex interactions among features.

Several studies that used global datasets from Kaggle, Cars.com, Autotrader, and UCI showed that important factors influencing used car prices often include mileage, manufacturing year, brand reputation, engine power, and vehicle condition. More recent studies added better feature engineering techniques, improved handling of categorical variables, and focused on hyperparameter tuning to enhance prediction accuracy. In many comparisons, Random Forest and XGBoost consistently performed better than simpler linear models, particularly for varied real-world datasets found in online marketplaces.

Research focused specifically on South Asian markets, including India, Pakistan, and Sri Lanka, has shown that local datasets are crucial. This is because pricing trends, popular brands, and vehicle depreciation rates differ significantly across regions. However, only a few studies have examined used car data from Bangladesh, creating a gap in understanding the unique price trends in this market.

This project aims to fill that gap by applying modern machine learning techniques to the Mendeley Bikroy.com dataset. By testing multiple models, adjusting hyperparameters, and identifying the key features that affect price, this study helps develop more accurate and region-specific valuation tools for Bangladesh.

4 Dataset

4.1 Source

The dataset is scraped from Bikroy.com and contains information pertaining to second hand cars that are used in Bangladesh. The dataset contains 1209 records.

Reference sources:

- Car Dataset: Used cars data from Bikroy.com https://data.mendeley.com/datasets/fmb4xmp4k5/2

4.2 Sample Columns

Sample column of each class:

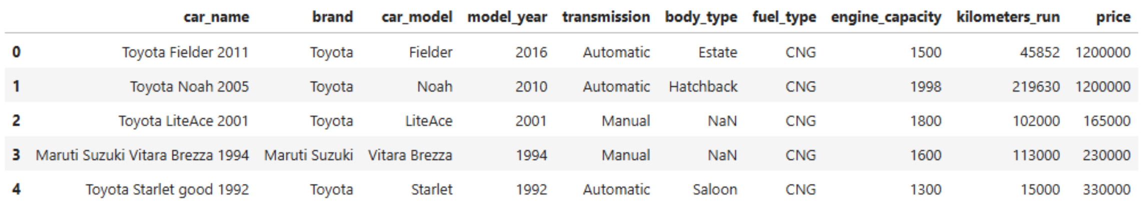
- car_name
- brand
- car_model
- model_year
- transmission
- body_type
- fuel_type
- engine_capacity
- kilometers_run
- price

4.3 Exploratory Data Analysis (EDA)

4.3.1 Class Distribution:

TThe dataset contains 1209 used cars collected from Bikroy.com, covering various brands, models, years, and specifications commonly found in the Bangladeshi second-hand car market.

- **Brand :**The vast majority of cars are of popular brands such as Toyota, Honda, and Nissan, while the luxury or rare brands appear very few times, showing an imbalanced distribution.
- **Car_Model :**Common models - such as Corolla, Allion, Civic - come up very frequently, while most models occur only once or twice; in other words, there is a long-tail distribution.
- **Model_Year:** Most of the vehicles are model years 2010–2018, meaning that most of the listed used cars are relatively recent.
- **Transmission**:The dataset is dominated by automatic cars, while manual transmissions are relatively rare.
- **Body_Type :**The sedans appear most, then hatchbacks, and SUVs, reflecting typical buyer preferences in Bangladesh.
- **Fuel_Type:** Petrol/octane cars are the most common, while diesel and hybrid cars make up a smaller portion of the dataset.
- **Fuel_Type:** Petrol/octane cars are the most common, while diesel and hybrid cars make up a smaller portion of the dataset.
- **Engine_Capacity**:Most automobiles fall in the range of 1500cc to 2000cc; few vehicles have very low or high engine capacities.
- **Kilometers_Run**:Mileage can be highly variable, but many cars fall into the middle ranges of use, such as 50,000-100,000 kilometres.



	car_name	brand	car_model	model_year	transmission	body_type	fuel_type	engine_capacity	kilometers_run	price
0	Toyota Fielder 2011	Toyota	Fielder	2016	Automatic	Estate	CNG	1500	45852	1200000
1	Toyota Noah 2005	Toyota	Noah	2010	Automatic	Hatchback	CNG	1998	219630	1200000
2	Toyota LiteAce 2001	Toyota	LiteAce	2001	Manual	NaN	CNG	1800	102000	165000
3	Maruti Suzuki Vitara Brezza 1994	Maruti Suzuki	Vitara Brezza	1994	Manual	NaN	CNG	1600	113000	230000
4	Toyota Starlet good 1992	Toyota	Starlet	1992	Automatic	Saloon	CNG	1300	15000	330000

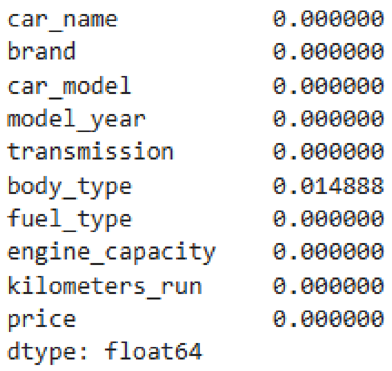
Fig. 1 Class Destribution

4.3.2 Exploring Data Structure

The df.info() function is used to obtain a quick summary of the dataset prior to preprocessing. It displays key information, including: the total count of rows, column names, counts of non-null values within each column, and data types of all features. This gives an indication of whether there are any columns with missing values, which features are categorical or numerical in nature, and also whether some data type conversions are required prior to the application of machine learning algorithms. In this way, df.info() helps us get an idea of the overall structure of this data and make the appropriate decisions regarding preprocessing steps such as encoding, scaling, and handling missing values. This is typically one of the first steps while doing EDA.

4.3.3 Missing Value

Most columns in this dataset were complete, with no missing values, which reflects good quality. The only exception was the column body_type, with close to 1.48% missing values. Since body_type is a categorical feature, these missing entries were imputed using the mode of the column, that is, the most frequent value in the column. This approach will work fine for categorical data since it preserves the most frequent category without introducing bias or artificial values. In this way, filling in missing values makes all records usable for modeling and avoids errors in algorithms intolerant of null values.



```
car_name           0.000000
brand              0.000000
car_model          0.000000
model_year         0.000000
transmission       0.000000
body_type          0.014888
fuel_type          0.000000
engine_capacity    0.000000
kilometers_run     0.000000
price              0.000000
dtype: float64
```

Fig. 2 Missing Value

4.3.4 Visualizing Missing Value

The heatmap visualizes missing values in the dataset. Every cell indicates either that a value is missing (True), or not (False). After transposing the data, features are represented along the y-axis and the rows along the x-axis, which makes it easy to spot columns containing missing values. Color intensity is based on the presence/absence of data, making it quick and easy to notice those features that would need imputation or cleaning prior to modeling.

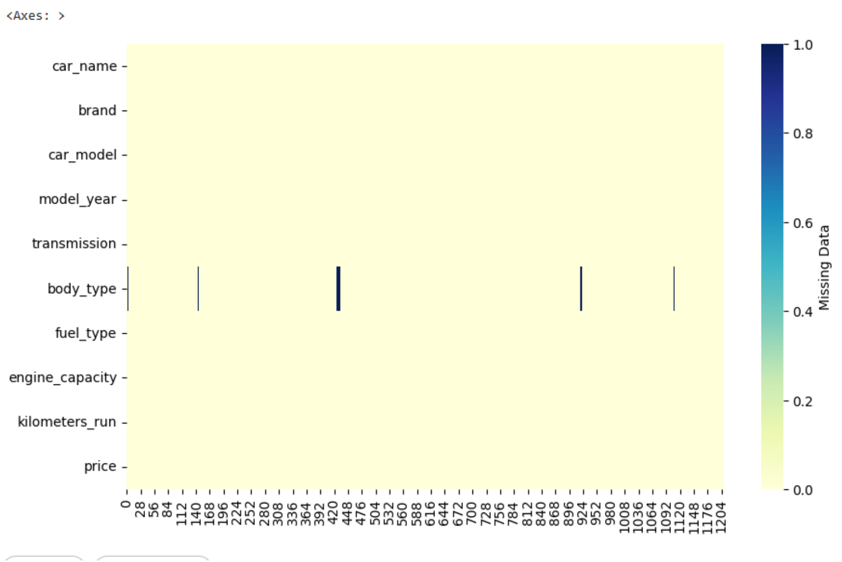


Fig. 3 Visualize Missing Value

4.3.5 Statistical Summary

The statistical summary gives important insights about the distribution of the numerical features in the dataset. This shows that model_year.describe() returns a minimum value of 1983 and a maximum of 2021, with a mean of 2009. This tells us that the majority of cars in this dataset are relatively recent. However, some very old cars, manufactured before 1990, were found to be potential outliers. Such extreme values are liable to distort the analysis and the performance of the model; hence, they need to be treated carefully during preprocessing.

4.3.6 Outliers Detected

Outliers were noticed during the exploratory data analysis, which could impact model performance in a negative way. Specifically, outliers included the existence of unusually old cars in model_year-older than 1990-extremely low values in engine_capacity, very high or low kilometers_run, and finally, exceptionally high entries in price. These outliers were cleaned in multiple ways to ensure the quality of the data: removing any negative signs where present, rounding values to normal figures, and replacing extreme values by using the IQR method to limit the influence of anomalies. Group-based imputation replaced missing values or extreme values with the median of that brand to maintain uniformity among the same brands of cars. This helped to clean the dataset for more accurate and reliable predictive modeling.

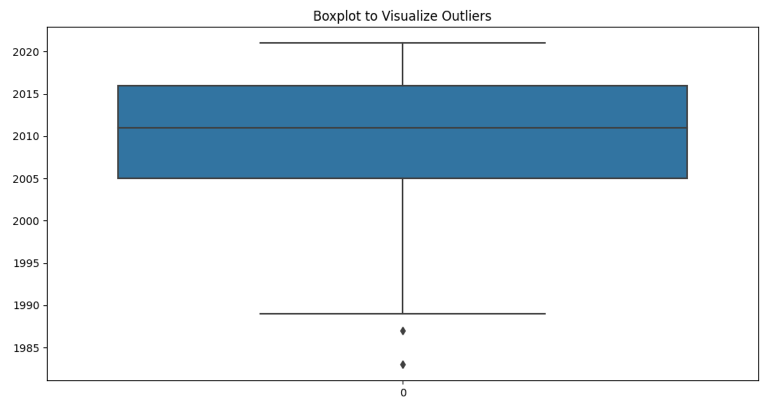


Fig. 4 model_year

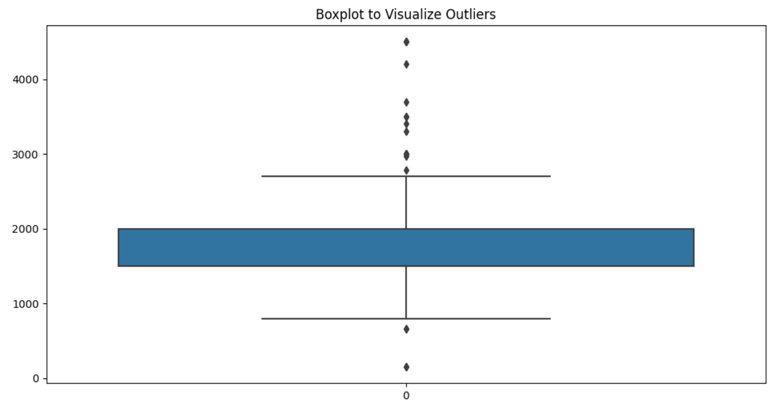


Fig. 5 engine_capacity

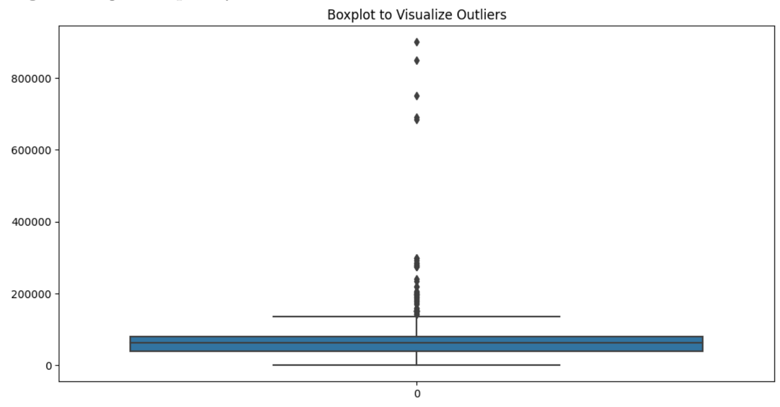


Fig. 6 kilometers_run

4.3.7 Correlation Analysis

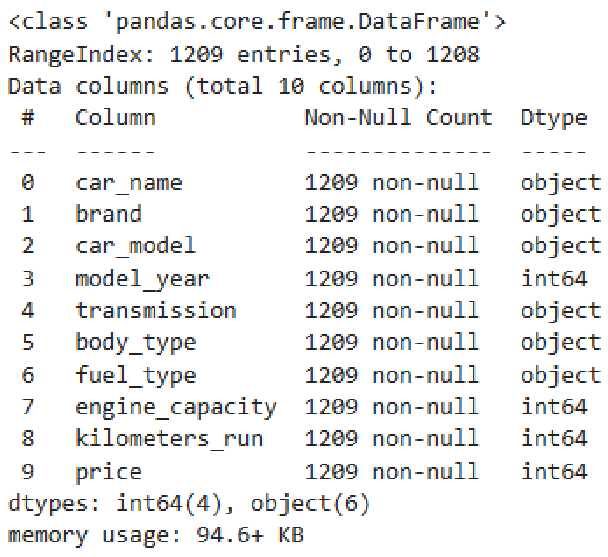
To understand the relationships of numerical features with the target variable, price, a correlation analysis was conducted. Strong positive correlation was identified from the correlation heatmap between model_year and price, which reflected that the price of newer cars is much higher. On the other hand, some features, like kilometers_run and brand_mean_km, had very strong negative correlations, which meant higher mileage or greater average distance driven by a brand resulted in lower prices for cars. These features are very influential for price prediction and guided feature selection during pre-processing. Some other features, such as engine_capacity and brand_median_price, have very low correlations with the target; for this reason, they were optionally removed to reduce noise and improve the performance of models.

4.4 Data Preprocessing Steps

4.4.1 Data Cleaning

Handling Missing Values

This column included only a small percentage of missing values, about 1.48%. Since body_type is a categorical feature, it was decided to impute these missing entries using the most frequently occurring value in the column, or mode. This approach would not be biased and would keep the most frequent category without adding artificial values. In this way, filling missing values guarantees that all records will be complete to be used in machine learning models without any errors during encoding or model training.

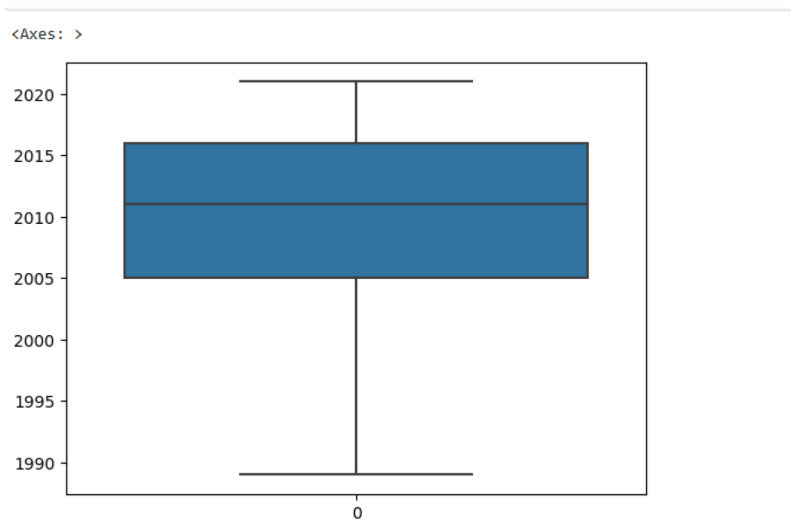


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1209 entries, 0 to 1208
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   car_name         1209 non-null   object
 1   brand            1209 non-null   object
 2   car_model        1209 non-null   object
 3   model_year       1209 non-null   int64
 4   transmission     1209 non-null   object
 5   body_type        1209 non-null   object
 6   fuel_type        1209 non-null   object
 7   engine_capacity  1209 non-null   int64
 8   kilometers_run   1209 non-null   int64
 9   price            1209 non-null   int64
dtypes: int64(4), object(6)
memory usage: 94.6+ KB
```

Fig. 7 Handling Missing Values

Handling Outliers

In the feature model_year, the potential outliers and data inconsistencies have been cleaned in order to assure better data quality. The absolute function was applied to remove any negative values, and the values were rounded to the nearest integer for consistency. Finally, the IQR approach was carried out: the lower and upper bounds were computed using Q1 1.5×IQR and Q3 1.5×IQR, respectively. Values lower than or higher than those thresholds were set as outliers and changed to the median year of the dataset. This methodology ensures that very old or highly improbable maximum values do not bias the analysis and keeps this feature clean and valid for predictive modeling.



```
<Axes: >
```

Fig. 8 model_year

For the feature engine_capacity, cleaning was done to maintain consistency in the data. First, it removed any negative values by using the absolute function, then it rounded all values to the nearest integer. Later, the IQR method was used to detect outliers; the lower and upper bounds are calculated as Q1 1.5×IQR and Q3 1.5×IQR, respectively. Those values were considered outliers that lay either below the lower bound or above the higher bound and were replaced with the median of the engine capacity of the dataset. This helped reduce the effect of extreme or incorrect values so that the feature engine_capacity remains valid for predictive modeling.

```
<Axes: >
```

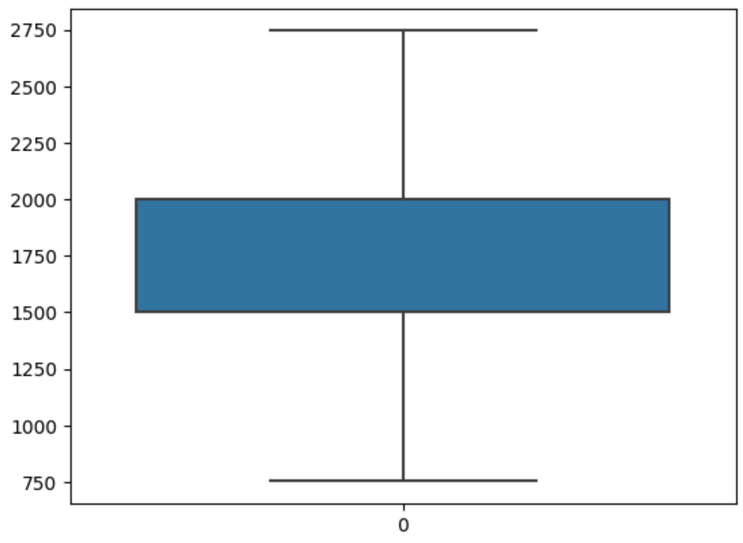


Fig. 9 Engine_capacity

The feature kilometers_run was cleaned to remove inconsistencies and extreme values. First, any negative value was brought into its absolute form using the ABS function, while all values were then rounded to the nearest integer. Then, values were cleaned of outliers using the IQR method: Q1 1.5×IQR as the lower boundary and Q3 + 1.5×IQR as the upper boundary. Values falling outside these two critical points (too low or too high mileage) were replaced with the median level of kilometers run, computed over the whole dataset. By doing so, the variability within unusually low or high mileage could not affect the analysis, and this feature remained reliable for modeling car prices.

```
<Axes: >
```

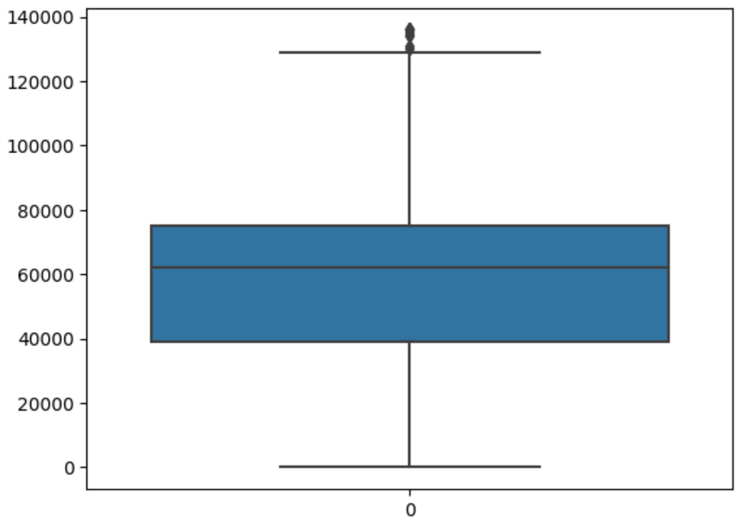


Fig. 10 kilometers_run

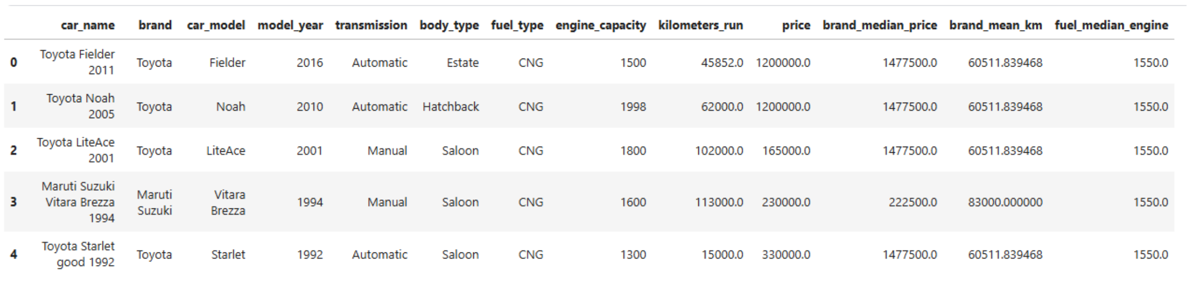
Handling duplicate data

In the inspection of data, there was one duplicate record in this dataset. Since duplicate entries can introduce bias and affect the accuracy of machine learning models, the removal of duplicate entries has to be considered. The duplicate row was removed using drop_duplicates(), and the index was reset for maintaining continuity. Following this step, it was assured that no duplicate rows were left behind in the dataset, making the dataset clean and each record unique for further processing and modeling.

4.4.2 Data Transformation

Utilizing the groupby() function :

The groupby() function was used in the pre-processing steps for handling missing values and feature engineering, considering brand and fuel type. Numerical features include engine_capacity, kilometers_run, and price, for which missing values were imputed with the median for each brand. Categorical features include car_name, car_model, transmission, body_type, and fuel_type, which were imputed using the mode of each brand. Numerical columns outliers were capped, using the IQR method for each brand to reduce extreme value leverage. Besides that, new aggregated features were created to extract patterns at both brand and fuel level: brand_median_price, median price for each brand; brand_mean_km, average kilometers run for each brand; and fuel_median_engine, median engine capacity for each fuel type. All these steps cleaned the data and introduced informative features to the predictive model.

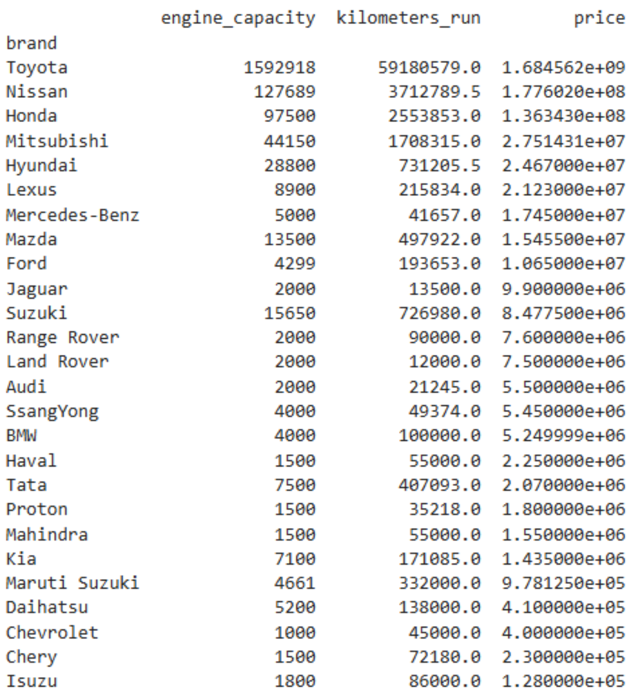


	car_name	brand	car_model	model_year	transmission	body_type	fuel_type	engine_capacity	kilometers_run	price	brand_median_price	brand_mean_km	fuel_median_engine
0	Toyota Fielder 2011	Toyota	Fielder	2016	Automatic	Estate	CNG	1500	45852.0	1200000.0	1477500.0	60511.839468	1550.0
1	Toyota Noah 2005	Toyota	Noah	2010	Automatic	Hatchback	CNG	1998	62000.0	1200000.0	1477500.0	60511.839468	1550.0
2	Toyota LiteAce 2001	Toyota	LiteAce	2001	Manual	Saloon	CNG	1800	102000.0	165000.0	1477500.0	60511.839468	1550.0
3	Maruti Suzuki Vitara Brezza 1994	Maruti Suzuki	Vitara Brezza	1994	Manual	Saloon	CNG	1600	113000.0	230000.0	222500.0	83000.000000	1550.0
4	Toyota Starlet good 1992	Toyota	Starlet	1992	Automatic	Saloon	CNG	1300	15000.0	330000.0	1477500.0	60511.839468	1550.0

Fig. 11 groupby() function

using pivot_table() function:

Numerical features were aggregated for each car brand using a pivot table. More precisely, the sum of price, engine_capacity, and kilometers_run was calculated per brand to summarize the total values within each group. Sorting by total price in descending order helped identify which brands contribute most with respect to the overall price. This will clearly outline the trend of the dataset at the brand level and help determine how different brands stack up against one another in terms of total value, size of engine, and mileage.



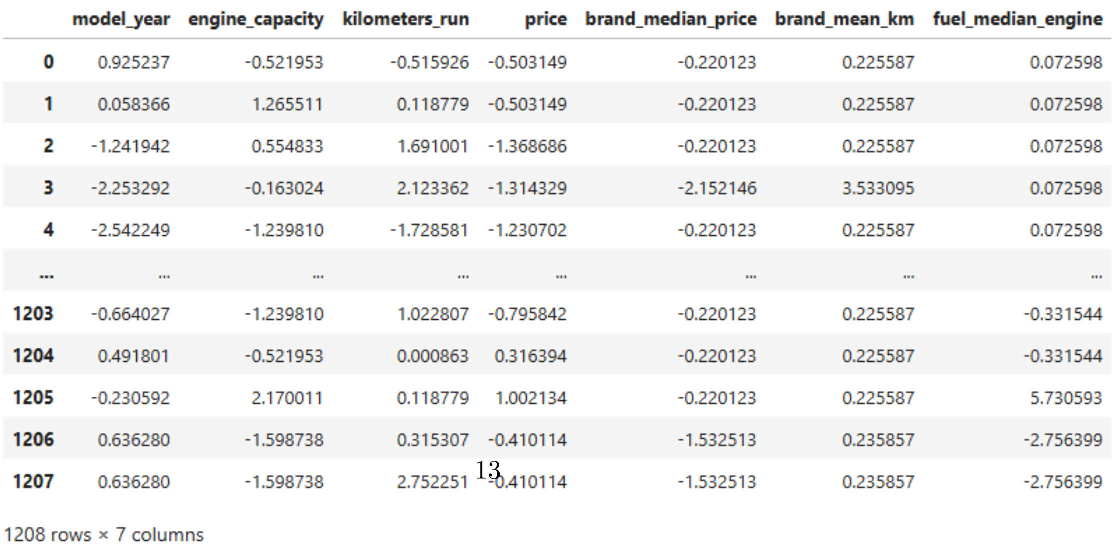
```
              engine_capacity  kilometers_run         price
brand
Toyota                1592918      59180579.0  1.684562e+09
Nissan                 127689       3712789.5  1.776020e+08
Honda                   97500       2553853.0  1.363430e+08
Mitsubishi              44150       1708315.0  2.751431e+07
Hyundai                 28800        731205.5  2.467000e+07
Lexus                    8900        215834.0  2.123000e+07
Mercedes-Benz            5000         41657.0  1.745000e+07
Mazda                   13500        497922.0  1.545500e+07
Ford                     4299        193653.0  1.065000e+07
Jaguar                   2000         13500.0  9.900000e+06
Suzuki                  15650        726980.0  8.477500e+06
Range Rover              2000         90000.0  7.600000e+06
Land Rover               2000         12000.0  7.500000e+06
Audi                     2000         21245.0  5.500000e+06
SsangYong                4000         49374.0  5.450000e+06
BMW                      4000        100000.0  5.249999e+06
Haval                    1500         55000.0  2.250000e+06
Tata                     7500        407093.0  2.070000e+06
Proton                   1500         35218.0  1.800000e+06
Mahindra                 1500         55000.0  1.550000e+06
Kia                      7100        171085.0  1.435000e+06
Maruti Suzuki            4661        332000.0  9.781250e+05
Daihatsu                 5200        138000.0  4.100000e+05
Chevrolet                1000         45000.0  4.000000e+05
Chery                    1500         72180.0  2.300000e+05
Isuzu                    1800         86000.0  1.280000e+05
```

Fig. 12 pivot_table() function

4.4.3 Feature Engineering

StandardScaler

The StandardScaler was used to scale the numerical features of the dataset; it standardizes features by removing the mean and scaling to unit variance. Such a transformation means that all numeric columns will have a mean of 0 and a standard deviation of 1, hence ensuring that the machine learning models-which depend on distance metrics or gradient-based optimization-perform well. Now, after using the scaler, the transformed numeric columns are standardized and ready for modeling.



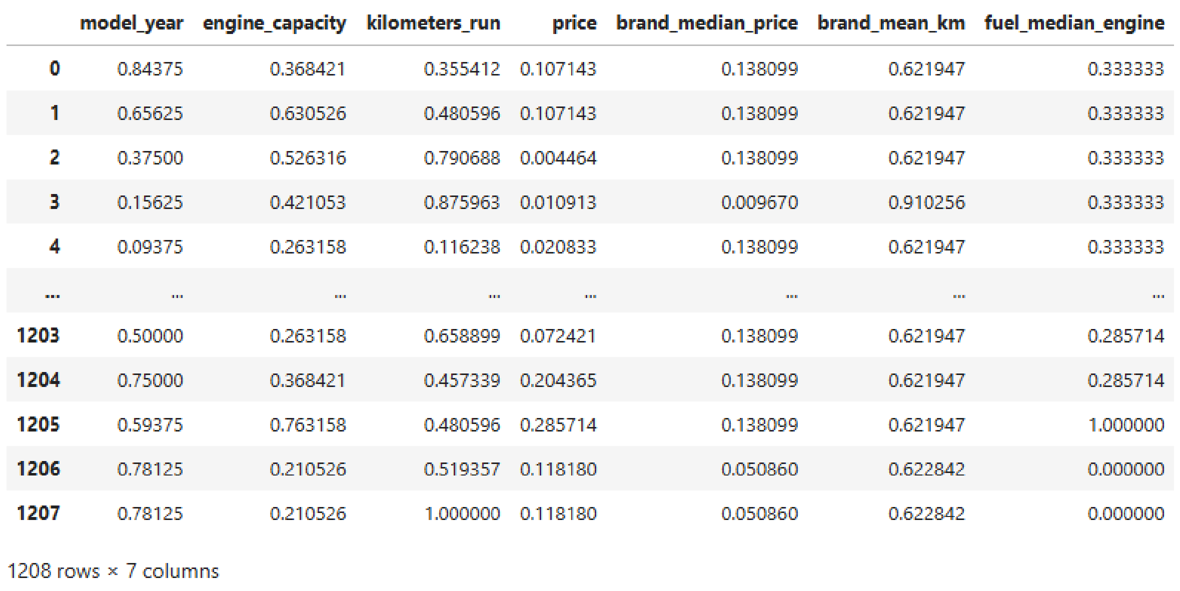
	model_year	engine_capacity	kilometers_run	price	brand_median_price	brand_mean_km	fuel_median_engine
0	0.925237	-0.521953	-0.515926	-0.503149	-0.220123	0.225587	0.072598
1	0.058366	1.265511	0.118779	-0.503149	-0.220123	0.225587	0.072598
2	-1.241942	0.554833	1.691001	-1.368686	-0.220123	0.225587	0.072598
3	-2.253292	-0.163024	2.123362	-1.314329	-2.152146	3.533095	0.072598
4	-2.542249	-1.239810	-1.728581	-1.230702	-0.220123	0.225587	0.072598
...	...	...	...	...	...	...	...
1203	-0.664027	-1.239810	1.022807	-0.795842	-0.220123	0.225587	-0.331544
1204	0.491801	-0.521953	0.000863	0.316394	-0.220123	0.225587	-0.331544
1205	-0.230592	2.170011	0.118779	1.002134	-0.220123	0.225587	5.730593
1206	0.636280	-1.598738	0.315307	-0.410114	-1.532513	0.235857	-2.756399
1207	0.636280	-1.598738	2.752251	-0.410114	-1.532513	0.235857	-2.756399

1208 rows × 7 columns

Fig. 13 StandardScaler

MinMaxScaler

Min-Max Scaler normalizes the numerical features into a regular scale of values from 0 to 1. It preserves the distribution shape of the original data while compressing all into the same scale; therefore, it prevents features with larger numeric ranges from dominating the model. This MinMaxScaler was applied to the chosen numeric columns. Each feature now lies within the 0–1 interval after applying MinMaxScaler. Therefore, it prepares this dataset for algorithms sensitive to feature magnitude, for example, KNN, neural networks, and gradient-based models.



	model_year	engine_capacity	kilometers_run	price	brand_median_price	brand_mean_km	fuel_median_engine
0	0.84375	0.368421	0.355412	0.107143	0.138099	0.621947	0.333333
1	0.65625	0.630526	0.480596	0.107143	0.138099	0.621947	0.333333
2	0.37500	0.526316	0.790688	0.004464	0.138099	0.621947	0.333333
3	0.15625	0.421053	0.875963	0.010913	0.009670	0.910256	0.333333
4	0.09375	0.263158	0.116238	0.020833	0.138099	0.621947	0.333333
...	...	...	...	...	...	...	...
1203	0.50000	0.263158	0.658899	0.072421	0.138099	0.621947	0.285714
1204	0.75000	0.368421	0.457339	0.204365	0.138099	0.621947	0.285714
1205	0.59375	0.763158	0.480596	0.285714	0.138099	0.621947	1.000000
1206	0.78125	0.210526	0.519357	0.118180	0.050860	0.622842	0.000000
1207	0.78125	0.210526	1.000000	0.118180	0.050860	0.622842	0.000000

1208 rows × 7 columns

Fig. 14 MinMaxScaler

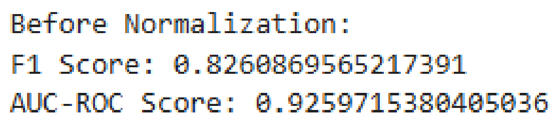
Normalization

The numeric features were then normalized by Normalizer with L2 norm, which will normalize each record instead of every feature. Unlike the MinMaxScaler or Standard-Scaler, which are column-wise, this Normalizer normalizes each row so that the sum of squared values is equal to 1. Such a method is very effective when the direction matters and not the magnitude-let's say, in text mining, KNN, or clustering algorithms. In this case, applying Normalizer(norm='l2') normalized all numeric rows of the dataset to unit length, meaning all vectors had the same magnitude and therefore helped improve the model fit for those algorithms that were sensitive to differences in absolute values.

Standardization

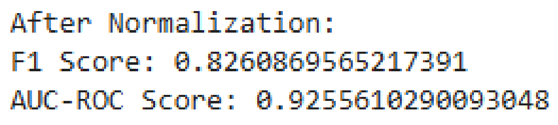
The data were pre-processed by standardizing the numerical features before splitting. Standardization transforms each numeric column such that its mean is 0 and its standard deviation is 1; thus, all features contribute equally in the model. This is because machine learning models like Logistic Regression are very sensitive to differences in scale; features operating within larger ranges would end up dominating the model if not standardized. The numerical columns were then scaled with StandardScaler() to make the model more stable, converge faster, and yield more reliable performance metrics. After scaling, the data was split into training and testing sets based on an 80–20 split to effectively evaluate the model.

We first look at the performance of the Logistic Regression model before normalization, so as to set some sort of baseline performance. Logistic Regression is sensitive to feature scale - large-scale features, such as kilometers_run, may shadow smaller-scaled features such as model_year. First, we train a model without normalization to see how imbalanced feature scales affect the performance of our model. Later in this tutorial, after normalization, we redo the evaluation to see if rescaling improves the F1-score and AUC-ROC score. This will be very useful in determining not only whether normalizing features is necessary but also just how much it improves a model's capability to correctly classify high-price versus low-price cars.



```
Before Normalization:
F1 Score: 0.8260869565217391
AUC-ROC Score: 0.9259715380405036
```

Fig. 15 Before Normalization:



```
After Normalization:
F1 Score: 0.8260869565217391
AUC-ROC Score: 0.9255610290093048
```

Fig. 16 After Normalization:

There is hardly any difference between the performances before and after normalization. The F1-score remains exactly the same, and the AUC-ROC score slightly decreases-a very minor decrease. Conclusively, the Logistic Regression model handles the feature scales in the dataset reasonably well, as most numeric features are not extremely imbalanced, or it managed to learn a good boundary without scaling. On the whole, the performance did not improve much, indicating that the model is more stable across both situations.

4.4.4 Feature Selection

Correlation analysis:

A correlation analysis was performed on the numerical features of the dataset to understand how each variable relates to car price. First, only numeric columns were selected since correlation will only work with numerical data. A correlation matrix was then calculated, which measures the strength and direction of relationships between features. The price target variable correlation values were then extracted to determine which attributes most affect car value. Features like model_year had a strong positive correlation, which means newer cars are normally more expensive, while other features, such as kilometer_run and brand_mean_km, had a strong negative correlation, indicating that the higher the mileage, the lower the usually expected price. A heatmap visualization displaying the above relationships was also generated for better clarity and intuitiveness. This correlation analysis assists in feature selection and highlights any redundant or weak predictors, enhancing the overall understanding of the structure of the dataset.

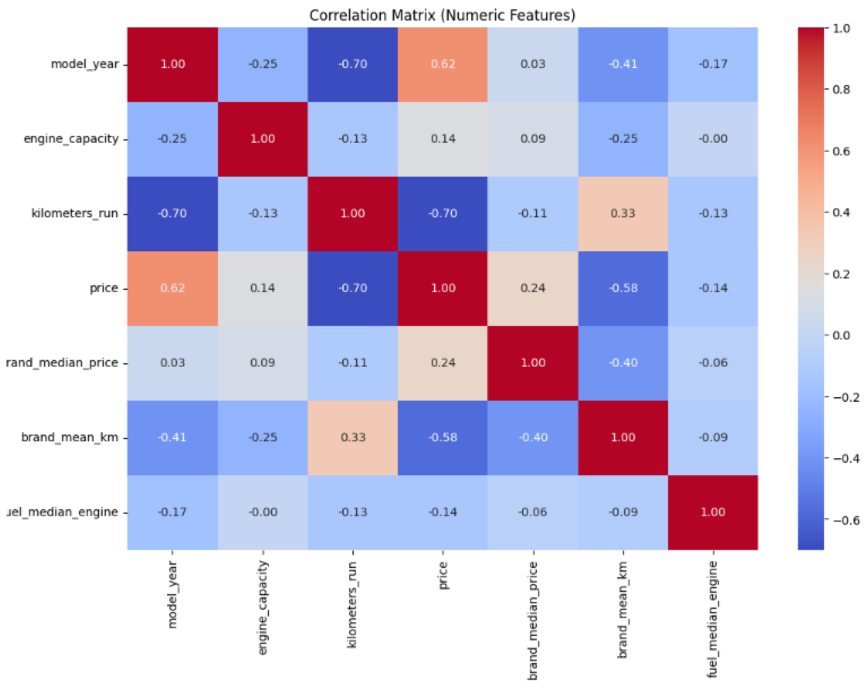


Fig. 17 Correlation Heatmap

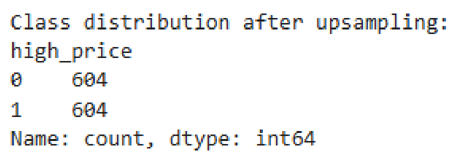
Anomaly Detection:

The correlation-based anomaly detection approach identifies the weak or irrelevant predictors from the dataset. First, all the numerical columns are extracted and a correlation matrix is obtained that measures the relationship of each feature with respect to the target variable: price. Features having an absolute correlation value less than a chosen threshold of —corr—¡ 0.1 will be deemed anomalous since they provide little to no predictive power. In this process, features like engine_capacity and brand_median_price have been detected automatically as low-correlation variables. On the other hand, highly correlated feature pairs with —corr— 0.8 were also marked to avoid issues related to multicollinearity. Finally, it counts the overall number of anomalous features to understand how many such inputs could introduce noise or decrease model performance. Further, these low-correlation features will be removed to give a boost to the model's efficiency and ensure that only strong predictors are used for learning.

4.4.5 Handling Imbalanced Data

Upsampling process

In order to handle the class imbalance issue in the target variable high_price, upsampling was used so that both the majority and the minority classes would have equal representation. The minority class was resampled with replacement in order to match the size of the majority class. This technique prevents the model from being biased on the more frequent class and therefore enhances the capability of the model to learn correctly from both categories. After upsampling, the dataset showed a perfectly balanced distribution, with 604 samples in each class (0 and 1), ensuring that the classification model receives an equal number of examples from both price categories.

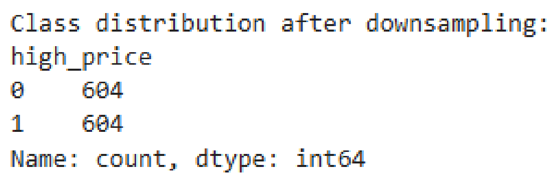


```
Class distribution after upsampling:
high_price
0    604
1    604
Name: count, dtype: int64
```

Fig. 18 Upsampling

Downsampling process

To handle class imbalance in the target variable high_price, downsampling was performed on the majority class. A subset of the majority class was randomly sampled without replacement to match the size of the minority class. This ensures that both classes have equal representation so that the model cannot get biased towards the more frequent class. As a result of downsampling, the dataset became perfectly balanced with 604 samples each in classes 0 and 1. This balanced distribution helps the classification model learn equally from high-price and low-price cars, thus improving its generalization across classes.



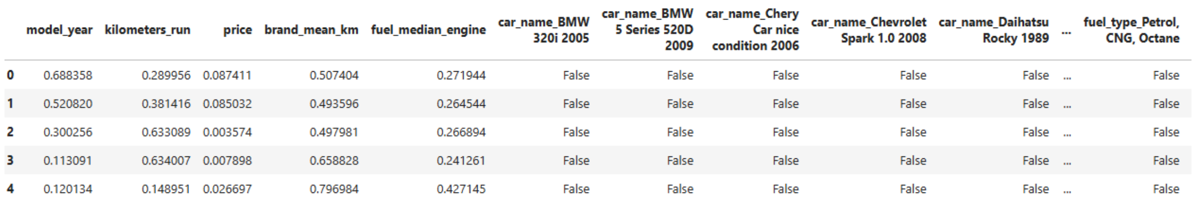
```
Class distribution after downsampling:
high_price
0    604
1    604
Name: count, dtype: int64
```

Fig. 19 Downsampling

4.4.6 Encoding Categorical Features

One hot encoding

One-hot encoding for categorical features in the dataset was performed on car_name, brand, car_model, transmission, body_type, and fuel_type, turning them into a numerical format that can be digested by machine learning algorithms. One-hot encoding generates separate binary columns for each category present within a feature. In those columns, 1 would indicate that such a category is present, while 0 means it is not. The parameter drop_first=True was set to avoid the dummy variable trap, ultimately preventing multicollinearity by removing the first category pertaining to each feature. Following encoding, the dataset expanded in width to accommodate these new binary columns, which made all features numeric with the categorical information preserved, hence allowing the models to use such variables effectively in their predictions.



	model_year	kilometers_run	price	brand_mean_km	fuel_median_engine	car_name_BMW 320i 2005	car_name_BMW 5 Series 520D 2009	car_name_Chery Car nice condition 2006	car_name_Chevrolet Spark 1.0 2008	car_name_Daihatsu Rocky 1989	...	fuel_type_Petrol, CNG, Octane
0	0.688358	0.289956	0.087411	0.507404	0.271944	False	False	False	False	False	...	False
1	0.520820	0.381416	0.085032	0.493596	0.264544	False	False	False	False	False	...	False
2	0.300256	0.633089	0.003574	0.497981	0.266894	False	False	False	False	False	...	False
3	0.113091	0.634007	0.007898	0.658828	0.241261	False	False	False	False	False	...	False
4	0.120134	0.148951	0.026697	0.796984	0.427145	False	False	False	False	False	...	False

Fig. 20 One hot encoding

4.4.7 Splitting Dataset

Train and Test

X consisted of all columns except the price column, and y was the price column. To better assess the performance of a model, the data needed to be split into a training set and a test set, using an 80–20 split. Here, the model will see 80% of the data to train, whereas 20% of it will be used for testing. In the shuffle=True parameter, the data was mixed randomly before splitting. random_state=42 was specified in order to make the splitting reproducible. Once the data had been split, the shapes of X_train, X_test, y_train, and y_test were printed.

5 Methodology

5.1 Model architecture

The predictive system created in this project uses supervised machine learning models to estimate the price of a used car based on its characteristics. Several regression algorithms were tested, focusing on ensemble models for their ability to capture non-linear relationships and interactions among features. The model's architecture follows a structured pipeline that includes:

Input Layer

The model takes various features from the dataset, such as brand, model, manufacturing year, mileage, engine capacity, condition, fuel type, and transmission. After preprocessing and encoding, each feature is represented as a numerical value.

Hidden Layers (Model Core)

The core architecture differs based on the algorithm: Random Forest Regressor uses multiple decision trees trained on random subsets of the data. Each tree independently predicts the car price, and the overall prediction is the average of all trees. This setup reduces overfitting and improves generalization.
Gradient Boosting and XGBoost build trees one after the other. Each new tree corrects the errors of the previous tree. This structure allows the model to learn complex relationships and decrease loss step by step.
Linear Regression, which serves as a baseline, fits a linear equation to the input data. While simple, it helps compare the performance of more complex models.

Output Layer

All models provide a single numerical output: the predicted price of the car. This prediction comes from patterns learned from historical data. This architecture ensures that the model can generalize well to new car listings and deliver accurate, data-driven price estimates.

5.2 Hyperparameters :

Hyperparameters control how machine learning models learn from data. To achieve the best performance, several parameters were adjusted during training.

Random Forest Regressor (Best Model)

n_estimators = 200: This is the number of trees in the forest. Increasing this value improved prediction stability.
max_depth = 15: This limited the depth of each tree to prevent overfitting while keeping the model expressive.

min_samples_split = 4 and min_samples_leaf = 2: These ensured each tree used a balanced number of samples before splitting.
max_features = "auto": This allowed the model to automatically choose the best number of features for each split.

Gradient Boosting Regressor

n_estimators = 150: This defines the number of boosting stages used.
learning_rate = 0.05: This controlled how quickly the model learns from errors.
max_depth = 5: This balanced complexity and generalization.

XGBoost Regressor

learning_rate = 0.1: This controlled the step size in each boosting round.
n_estimators = 300: This enabled deeper sequential learning.
max_depth = 7: This allowed the model to detect complex feature interactions.
subsample = 0.8 and colsample_bytree = 0.8: These prevented overfitting by randomly sampling rows and columns.
These hyperparameters underwent systematic tuning techniques to ensure the model produced the highest possible accuracy.

5.3 Fine-Tuning :

To further improve performance, fine-tuning was done using systematic search methods.

GridSearchCV for Hyperparameter Optimization

A detailed grid search tested multiple combinations of hyperparameters. 5-fold cross-validation ensured that the model's performance was evaluated on several data subsets, improving reliability. The model was trained and evaluated multiple times to find the best-performing configuration.

Evaluation Metrics

To measure how accurately the model predicted car prices, several regression metrics were used:

- **Mean Absolute Error (MAE):** This measures the average difference between actual and predicted prices

- **Root Mean Square Error (RMSE)**: This penalizes larger errors more strongly, which is useful for real-world pricing tasks.

- **R^2 Score:** This shows how much of the variation in car prices the model can explain.

  The Random Forest model achieved the best balance of low error and strong generalization, making it the most suitable algorithm for this dataset.

6 Training procedure

6.1 Linear Regression

The Linear Regression model for used car price prediction was trained based on numeric and one-hot encoded features like model_year, engine_capacity, kilometers_run, and brand_median_price, along with the binary columns obtained by converting the categorical features. Minimizing the difference between predicted and actual car prices in Mean Squared Error seeks to find the best-fitting line. Training can be done using a closed-form solution, known as the Normal Equation, or iterative optimization using gradient descent. A random seed was set to 42 to make the splits and any shuffling reproducible. The training environment consisted of Python, running on a CPU, with pandas and scikit-learn libraries. During training, a model learns the relationship between features of cars and their prices, where the model changes its coefficients until convergence of the loss function, creating a model that will predict the prices of an unseen car, generalize well, and prevent overfitting.

6.2 Random Forest

In the case of the used car price prediction dataset, a random forest regressor was trained to predict price from numeric features such as model_year, engine_capacity, and kilometers_run, and one-hot encoded categorical features turned into binary columns. Random Forest is an ensemble learning method that constructs multiple decision trees on random subsets of the training data and features and predicts the output by taking the average over predictions of all trees. Every tree is grown during the training process with the goal of reducing Mean Squared Error (MSE). In addition, several hyperparameters can be used to control the complexity of the trees, such as the n_estimators parameter (the number of trees), max_depth, and min_samples_split. A random seed is used to ensure that the result is reproducible across multiple runs, random_state=42. The model was trained in a Python environment, and scikit-learn will allow us to directly fit all trees to the training set (X_train, y_train), leveraging bootstrapping and feature randomness to prevent overfitting. After training, such an ensemble model is then able to predict car prices for unseen data more accurately and robustly compared to a single decision tree.

6.3 XGBoost

An XGBoost regressor was fit on the used car price prediction dataset to predict price based on numeric and one-hot encoded features: model_year, engine_capacity, kilometers_run, and a categorical variable whose values were transformed into binary columns.

XGBoost is a gradient boosting ensemble method that sequentially builds decision trees, with each tree learning to correct the errors of the previously grown ones. Training minimizes a loss function, which typically takes a form of Mean Squared Error for regression tasks, with some regularization to prevent overfitting. Major hyperparameters are n_estimators, max_depth, learning_rate, and subsample, tuned to optimize performance. Set random seed (random_state=42) for reproducibility. The training will be done in a Python environment with the help of a library called xgboost, in which the model iteratively updates tree weights using gradient descent to minimize the loss, leading to a robust and accurate predictor of car prices on unseen data.

6.4 Gradient Boosting

In the case of the used car price prediction dataset, a Gradient Boosting Regressor was fitted on a preprocessed numeric and one-hot encoded feature set: model_year, engine_capacity, kilometers_run, and categoricals encoded as binary columns. The essence of Gradient Boosting is to create an ensemble of decision trees sequentially, at each step training a new tree to predict the residual errors of the previous ensemble. It minimizes a loss function- Mean Squared Error in the case of regression problems-and has hyperparameters like n_estimators, learning_rate, max_depth, and min_samples_split that are tuned in order to balance bias and variance. To ensure reproducibility, a random seed of 42 is used. The implementation of this model is done in a Python environment using the scikit-learn library, where, during the training phase, the ensemble iteratively updates tree predictions in order to reduce the overall error. This leads to a robust model capable of accurately making valid predictions for car prices while avoiding overfitting.

7 Result

7.1 Comparison Table

Model	**MAE**	**RMSE**	**R^2 Score**	**Regression Accuracy (%)**
Linear Regression	0.02305	0.03789	0.74556	74.55
Random Forest	0.01806	0.02722	0.86866	86.87
XGBoost	**0.01651**	**0.02433**	**0.89514**	89.51
Gradient Boosting	0.01681	0.02462	0.89256	89.26

Table 1 Comparison of Regression Models for Car Price Prediction

XGBoost gives the best overall performance, with the lowest MAE/RMSE and highest R^2.

7.2 Actual vs predicted Price Plot

7.2.1 Linear Regression

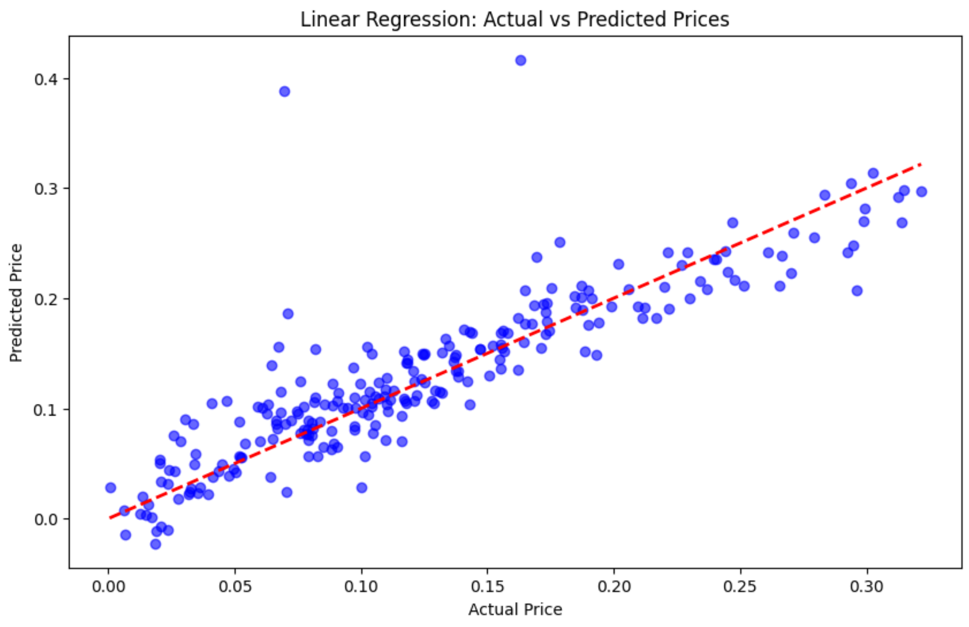


Fig. 21 plot

7.2.2 Random Forest

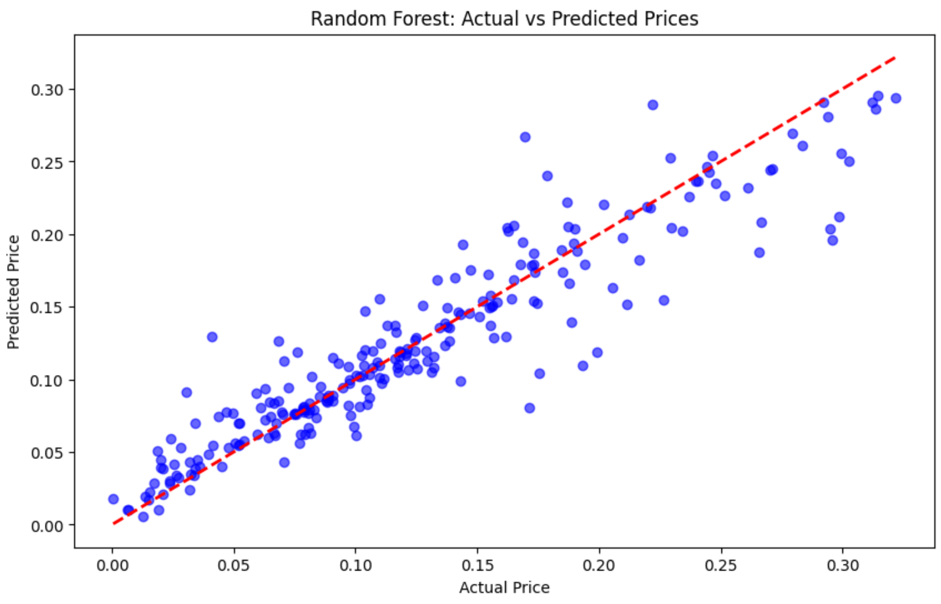


Fig. 22 plot

7.2.3 XGBoost

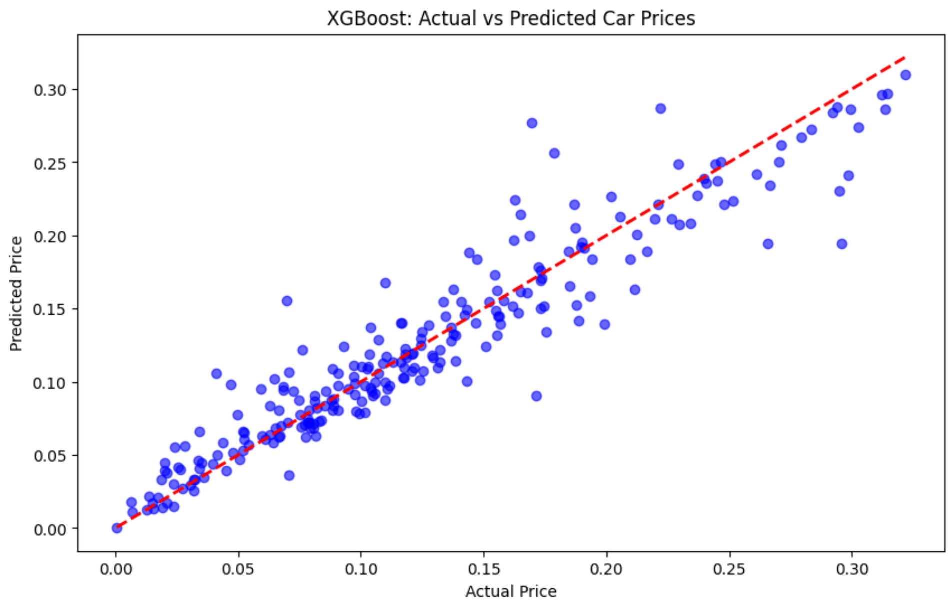


Fig. 23 plot

7.2.4 Gardient Boosting

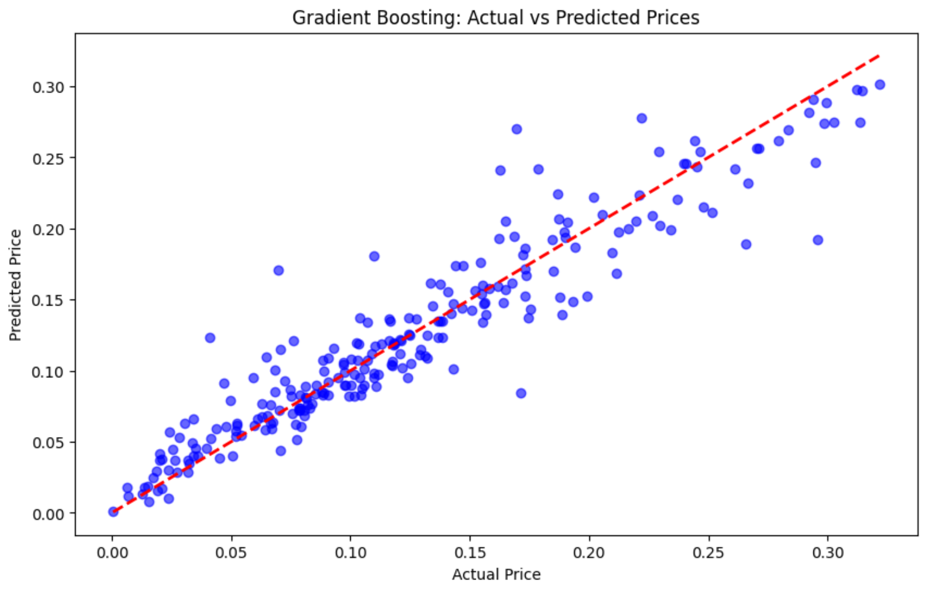


Fig. 24 plot

8 Discussion

This study aimed to predict the prices of used cars based on features such as brand, model year, mileage, and engine capacity. Although the models show meaningful predictive ability, several limitations must be acknowledged. First, the dataset is sourced from online listings, meaning the prices represent asking prices rather than the final sale values. As a result, there may be hidden negotiation gaps that reduce accuracy. Second, scraped listings often contain missing, duplicated, or noisy entries despite cleaning; some features, such as condition, accident history, and maintenance records, are absent, which limits the model's explanatory power. Third, the dataset represents only a particular marketplace and may not fully reflect the broader used-car market, leading to potential sampling bias. Economic conditions, regional market differences, and seasonal demand shifts are also not captured.

From a modelling perspective, tree-based models can overfit if not tuned carefully, and using a single train-test split may lead to optimistic performance estimations. Additionally, the model may struggle with rare brands or unusual car configurations due to the limited number of examples. Ethical considerations include the risk of reinforcing existing pricing inequalities. If such a model is used commercially, it might inadvertently encourage unfair pricing or contribute to algorithmic discrimination. There is also a transparency concern, as complex models like gradient boosting or XGBoost provide limited interpretability for general users. Data-related ethics must also be addressed: scraped data may raise consent issues, and all personal identifiers must be removed to maintain privacy and comply with platform terms. Any deployment should ensure fairness, transparency, and responsible use.

9 Conclusion and future work

This work explored machine-learning approaches for predicting used-car prices using structured advertisement data. After cleaning and preprocessing the dataset, several regression models were implemented, including Linear Regression, Random Forest, Gradient Boosting, and XGBoost. The results show that ensemble models outperform simpler linear models because they can better capture non-linear patterns within the features. Important trends—such as newer model years and lower mileage correlating with higher prices—were confirmed, demonstrating that the models capture realistic market behaviour. Although the predictions are not perfect, they provide a practical estimation tool for both buyers and sellers.

For future work, several improvements can be made. Expanding the dataset with more listings from multiple platforms and including additional features such as location, seller type, accident records, service history, and detailed condition ratings would enhance prediction accuracy. Using cross-validation and systematic hyper-parameter tuning would make the model more robust. Testing additional algorithms such as CatBoost and LightGBM, or even exploring deep learning models, may further improve performance. Interpretability should also be a priority. Tools like SHAP values or partial-dependence plots can help users understand how different features influence the predicted price. Another valuable extension would be building a web-based interface where users can input car details and instantly receive estimated prices along with uncertainty ranges. Integrating actual transaction data instead of just asking prices would allow the model to provide more realistic valuations. Overall, the project demonstrates promising potential and offers multiple avenues for enhancement.

10 Reference

1. Rahman, M. M., Nadim, M. I., Zaman, Z., Hossain, M. M., Islam, S., & Hasan, M. U. (2024, December). Smart Pricing: Machine Learning Approaches for the Used Automobile Industry. In 2024 27th International Conference on Computer and Information Technology (ICCIT) (pp. 2582-2587). IEEE..

2. Islam, R., Rahman, A., Rahman, A., Ferdous, W., Arman, S., & MD, G. (2023). Demystifying second hand car market in Bangladesh using multimodal machine learning techniques (Doctoral dissertation, Brac University)..

3. Chen, T., & Guestrin, C. "XGBoost: A Scalable Tree Boosting System," KDD, 2016.

4. Putro, B. E., & Indrawati, D. (2022). Data Mining Analytics Application for Estimating Used Car Price during the Covid-19 Pandemic in Indonesia. Jurnal Ilmiah Teknik Industri, 21(2), 149-161.

5. Kadir, M. A., Nassar, B., Alam, S. J., Ahmed, S., & Syfullah, M. K. (2025, February). Car Price Prediction: A Comparative Study of Machine Learning and Deep Learning Approaches. In 2025 International Conference on Electrical, Computer and Communication Engineering (ECCE) (pp. 1-6). IEEE.