Car Price Prediction Project

***Abstract*—The goal of this project is to use Linear Regression to predict the price of used cars based on data collected from various online sources.**

I. Introduction

The goal of this project is to predict the selling price of used cars by applying machine learning techniques, specifically Linear Regression. Data for this project was collected from online sources including websites like Oto, Bonbanh, and Xebiz. These sources provide detailed information on various factors that may influence a car's value, such as year of manufacture, mileage, vehicle condition, fuel type, and body type. This data is standardized and preprocessed to support building an accurate predictive model. Ultimately, this project aims to provide reliable estimates of used car prices based on these influential features.

II. Data Requirements

- **Name**: The model or specific name of the car.
- **Price**: The selling price of the used car.
- **Sale Date**: The date on which the car was sold.
- **Public Year**: The year the car was initially released or became publicly available.
- **Mileage**: The total distance the car has been driven, typically measured in kilometers or miles.
- **Origin**: The country or region where the car was manufactured.
- **Body Type**: The design classification of the car (e.g., sedan, SUV, hatchback).
- **City**: The city where the car is being sold or registered.
- **Year**: The specific manufacturing year of the car.
- **Fuel Type**: The type of fuel used by the car (e.g., gasoline, diesel, electric).
- **Manufacturer**: The company that produced the car.

Each of these fields captures important characteristics that can influence the car's price, helping to improve the predictive model's accuracy.

III. Data Collection

A. Libraries

The following libraries were used:

- **Selenium**: For web scraping.
- **Pandas**: For data handling and manipulation.
- **NumPy**: For mathematical operations.
- **Unidecode**: For encoding Vietnamese characters.

Method of Crawling

- Use Selenium to scrape the first page of each website. Since each website requires clicking a "show more" button to load more cars, configure Selenium to click the button and set up a waiting time to avoid being banned.
- After collecting links for each car, save them in a CSV file.
- Access each link, scrape all details listed above, and store them in a new CSV file.

B. Details

Xebiz: We gather all the data from the Overview table, which includes:

- Car Brand
- Origin
- Model
- Number of Engines
- Torque
- Gearbox
- Maximum Engine Power
- Fuel Capacity

The main goal is to use this information to fill in the missing data for two vehicle datasets. This method is referred to as *external inference*.

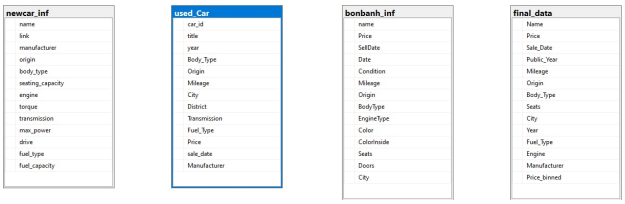


Fig. 1. Example of Xebiz car data

Bonbanh: From Bonbanh, we gather the following data from the car listings:

- **Basic Information:**
  - Name
  - City
- **Specification Table:**
  - Year of Manufacture
  - Km Travelled
  - Origin
  - Model
  - Gearbox
  - Engine
  - Exterior Colour
  - Interior Colour
  - Number of Seats
  - Number of Doors
  - Drive

These data points are crucial for understanding the technical specifications and condition of the cars listed on Bonbanh. Similar to the Xebiz data, we use this information to enrich and fill in missing values in our vehicle datasets, applying the external inference technique.

Oto.com: From Oto.com, we gather the following data from the car listings:

- **Basic Information:**
  - Year of Manufacture

Thông số kỹ thuật			
Năm sản xuất:	2021	Động cơ:	Dầu 2.0 L
Tình trạng:	Xe đã dùng	Màu ngoại thất:	Trắng
Số Km đã đi:	30,000 Km	Màu nội thất:	Đen
Xuất xứ:	Nhập khẩu	Số chỗ ngồi:	7 chỗ
Kiểu dáng:	SUV	Số cửa:	5 cửa
Hộp số:	Số tự động	Dẫn động:	RFD - Dẫn động cầu sau

Fig. 2. Example of Bonbanh car data

  - Model
  - Origin
  - Km Traveled
- **Location Information:**
  - Province
  - District
- **Car Details:**
  - Gearbox
  - Fuel Type

Năm SX	2023	Kiểu dáng	SUV	Tình trạng	Xe cũ
Xuất xứ	Nhập khẩu	Km đã đi	1.000 km	Tình thành	Hà Nội
Quận huyện	Cầu Giấy	Hộp số	Số tự động	Nhiên liệu	Máy xăng

Fig. 3. Example of oto.com car data

These details help in understanding the year, condition, and location-specific aspects of the cars. As with the other websites, we use this data to complete the missing fields in our datasets for further analysis and modeling.

IV. Data Storage

Using PyODBC and Pandas, the data was connected to a SQL server. Missing values were handled by replacing them with "None". Three tables were created: `newcar_inf`, `used_Car`, and `bonbanh_inf`.

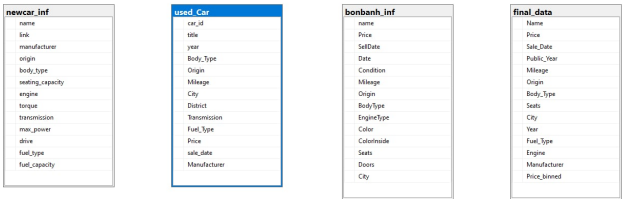


Fig. 4. All table we saved

V. Data Preprocessing

- Separate data into 3 dataframes (df).

New Car_inf Data Processing

  - Replace any 'None' values.
  - Adjust the fuel capacity for specific cars where it's not mentioned:
    - Vinfast Fadil 1.4 Tieu chuan CVT (ID: 21)
    - Toyota Venza 2.5 CVT (ID: 175)
    - Honda CR-V 1.5E 2021 (ID: 244)
    - Hyundai SantaFe 2.4 Xang (ID: 248)
    - Honda CR-V 1.5G 2021 (ID: 263)
    - Honda CR-V 1.5L 2021 (ID: 273)
    - Toyota Fortuner 2.4 TRD AT 4X2 (ID: 302)
    - BMW i8 1.5L Hybrid (ID: 552)
    - Rolls-Royce Cullinan V12 (ID: 585)
    - Rolls-Royce Dawn V12 (ID: 586)
  - Adjust the fuel type for specific cars:
    - Honda Civic 1.8 G (ID: 178)
    - Honda CR-V 1.5 E 2019 (ID: 245)
  - Drop the columns: `max_power`, `torque`, `link`.
  - Adjust the manufacturer for specific cars:
    - Land Rover Range Rover Evoque 2.0L I4 Turbocharged (ID: 464)
    - Replace 'Maybach' with 'Mercedes'.
    - Replace 'Rolls Royce' with 'Rolls-Royce'.
  - Replace 'Xe ban tai' with 'Ban tai' in the car body type.
  - Remove the term 'cho' from the seating capacity column.
  - Remove 'cc' from the engine column and round up engine values to the nearest integer.

Bonbanh_inf Data Processing

- Replace 'None' values in the dataframe with a null value.
- Separate values based on car condition:
  - `Xe moi` (new car) and `Xe cu` (used car).
- Format car names:
  - Split car name and version.
- Convert all values in the `Price` column to numeric format.
- Convert the `Date` column to integer format.
- Remove rows where `Date` is less than 2000.
- Rename the `Date` column to `Public Year`.
- **Mileage**:
  - Remove 'Km' and commas.
  - Change the column type to integer.
- **Origin**:
  - Replace 'Lap rap trong nuoc' with 'Lap rap'.
- **Body Type**:
  - Replace 'Van/Minivan' with 'Van'.
  - Replace 'Convertible/Cabriolet' with 'Convertible'.
  - Replace 'Ban tai / Pickup' with 'Ban tai'.
  - Remove all rows where body type is 'Truck'.
- **EngineType** and **Engine**:
  - Extract fuel types such as 'Xang', 'Dau', and 'Dien'.
  - Adjust values in the `Engine` column.
  - Change all data in the `Engine` column to integer format.
  - Drop the `EngineType` column.
- **Color**:
  - Drop the `Color` and `ColorInside` columns.
- **Seats** and **Doors**:

  - Remove the word 'cho' from the values.
  - Drop the `Doors` column.
- **Manufacturer**:
  - Split values where necessary.
  - Replace 'Rolls' with 'Rolls-Royce'.
  - Replace 'LandRover' with 'Land Rover'.

USED_CARS DATA PROCESSING

- Replace the value 'None' with NaN.
- Split the `title` column into `Public Year` and `Name` of the car:
  - Split the title and assign the first part as `Public Year` and the second part as `Name`.
  - Verify the results to ensure accuracy.
- Format the `Public Year`, `Origin`, `Mileage`, and `City` columns to align with other dataframes.
- Remove the `District` column as it has no impact on the model.
- Check the unique values for `Transmission` and `Fuel type` columns.
- Convert the `Price` column to a standard integer format.
- **Price Conversion Process**:
  - Remove the text "trieu" (Vietnamese for "million") and any surrounding whitespace.
  - Split the modified string to separate billions and millions (if present).
  - Convert prices based on the presence of "ti" (billion):
    - If `"ti"` is in the parts, multiply the billion part by 1,000 to convert to millions:
      - Example: "2 ti" is converted to 2,000 million.
      - If there are three parts, such as "2 ti 500," add the remaining millions to the converted billion value.
    - If there is no "ti," directly convert the remaining millions part.
  - Return the total price as an integer in millions.
- Format the `Manufacturer` column:
  - Replace "Mercedes-Benz" with "Mercedes."
- Fill the `Body type` column using the `fuzzywuzzy` library:
  - Use fuzzy matching to compare similar values across all 3 datasets.
  - Fill the `Body type` column if the similarity score is greater than 90%.

VI. DATA MERGING

DATA MERGING AND FORMATTING PROCESS

Formatting Column Names of `bonbanh_inf`

- Rename columns in `bonbanh_inf`:
  - `Bodytype` → `Body_Type`
  - `Selldate` → `Sell_Date`
  - `Fueltype` → `Fuel_Type`

Creating Merge Keys

Define the following merge keys:

- `Name`
- `Public Year`
- `Mileage`
- `Body_Type`
- `Origin`
- `City`
- `Fuel_Type`
- `Sell_Date`
- `Manufacturer`
- `Year`
- `Price`

Merging `bonbanh_inf` with `usedCar` by Merge Keys

- Use an outer merge: This includes all rows from both DataFrames, with NaN values for rows where there is no match in the other DataFrame.

Dropping Columns

- Drop the columns: `Seats`, `Engine`, and `Transmission`.

Data Formatting

Loop through Columns:

- Iterate through each column: `Origin`, `Body_Type`, `City`, `Fuel_Type`, and `Manufacturer`.
- Extract unique values from each column.

Binding Price Ranges:

- Define price ranges: 0, 300, 800, 2000.
- Bin the values in the `Price` column according to these ranges, using labels: `Low`, `Medium`, `High`, and `Luxury`.

Min-Max Scaling:

- **Scaling Function:** Define a function `min_max_scaling` that accepts a series (a column in a DataFrame) and applies min-max scaling.
- **Scaling Formula:**

$$\text{scaled value} = \frac{\text{value} - \min(\text{series})}{\max(\text{series}) - \min(\text{series})}$$

- Apply this scaling formula to the `Price` and `Mileage` columns in `final_data`, resulting in values between 0 and 1, where 0 represents the minimum and 1 the maximum.

One-Hot Encoding:

- **One-Hot Encoding:** Convert categorical columns in `final_data` into binary (0 or 1) columns. For each unique category in a column, a new binary column is created:
  - The value is 1 if the row matches that category.
  - The value is 0 otherwise.
- **Columns to Encode:**
  - `Origin`, `Body_Type`, `City`, `Fuel_Type`, `Manufacturer`, and `Price-binned`.
- **Result:** After encoding, `final_data_encoded` will have multiple new columns representing each unique category.

- **Convert Boolean Columns to Integers:** Identify columns of data type bool in `final_data_encoded` and convert them to integer type, where True becomes 1 and False becomes 0.

Store Data in SQL Server

- Save the processed `final_data_encoded` DataFrame back to the SQL server.

VII. Feature Engineering

A. Binning and Scaling

Price values were categorized into bins, and min-max scaling was applied to Price and Mileage.

B. Encoding

One-hot encoding was used for categorical columns, and Boolean columns were converted to integers.

VIII. Data Visualization and Exploratory Data Analysis

A. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is an essential step to understand the characteristics and patterns in the dataset before applying machine learning algorithms. In this section, we perform various EDA steps, including outlier detection, feature engineering, and visualizations.

- **Remove Outliers using the IQR Method:**
  - **Calculate the Quartiles:**
    - Q1: The 25th percentile of the 'Price' column.
    - Q3: The 75th percentile of the 'Price' column.
    - IQR (Interquartile Range): Calculated as $\text{IQR} = \text{Q3} - \text{Q1}$.
  - **Determine the Outlier Boundaries:**
    - *Lower Bound*: Defined as $\text{Q1} - 1.5 \times \text{IQR}$.
    - *Upper Bound*: Defined as $\text{Q3} + 1.5 \times \text{IQR}$.
- **Visualizing Histogram:**

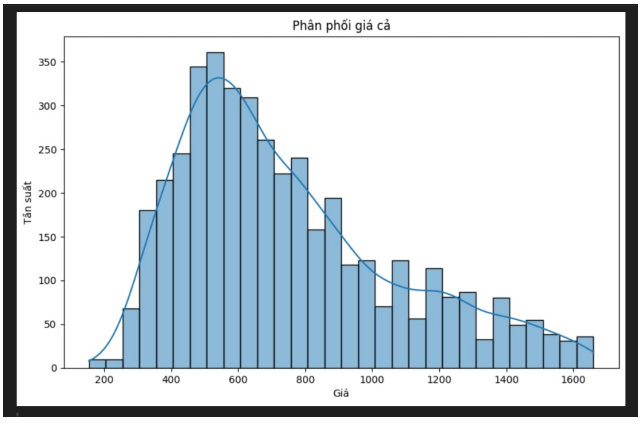


Fig. 5. Price distribution

  - **Shape of the Distribution:**
    - The histogram is right-skewed, with most of the data concentrated in the lower price range (around 400–800) and tapering off as the prices increase. This indicates that a majority of the entries in the dataset have lower prices, with fewer entries as the price rises.
  - **Frequency:**
    - The y-axis displays the frequency of prices within each bin. The highest frequency is around the 500–600 price range, with over 350 entries in that range, indicating this as a common price point.
  - **Distribution Curve:**
    - The smooth line over the histogram represents a density estimation, which aids in visualizing the shape of the distribution more clearly. The peak aligns with the bins of highest frequency, showing a general trend that most prices in this dataset are on the lower end.
- **Box Plot Analysis of All Numeric Columns:**

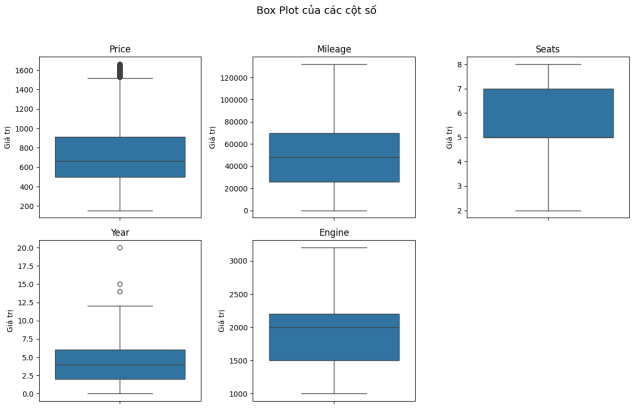


Fig. 6. Box plot of all numeric columns

  - **Price:**
    - The 'Price' variable is mostly concentrated between 400 and 1100, with a median around 700.
    - A notable number of high-value outliers appear above 1600, indicating a small subset of vehicles with significantly higher prices than the typical range.
  - **Mileage:**
    - The 'Mileage' column exhibits an extreme right skew, with several outliers extending up to around 8 million miles.
    - The majority of vehicles have lower mileage, suggesting a few high-mileage entries that may influence the overall distribution.
  - **Seats:**
    - The 'Seats' distribution is relatively balanced, spanning from 2 to 7 seats, with a median around 5–6 seats.
    - There are no significant outliers, indicating that the seating capacity is consistent across most vehicles in the dataset.
  - **Year:**
    - The 'Year' column shows a concentration of data points from earlier years, with a few high outliers representing newer models.

    - This suggests that the dataset contains mostly older vehicles, with some recent additions that stand out as outliers.
  - **Engine:**
    - The 'Engine' size variable has multiple high outliers, especially values above 3000, indicating a few vehicles with large engines.
    - The majority of engine sizes fall between 1000 and 2000, with a median around 1500, representing a typical range within the dataset.

- **Heatmap for Correlation Matrix of Numeric Columns:**

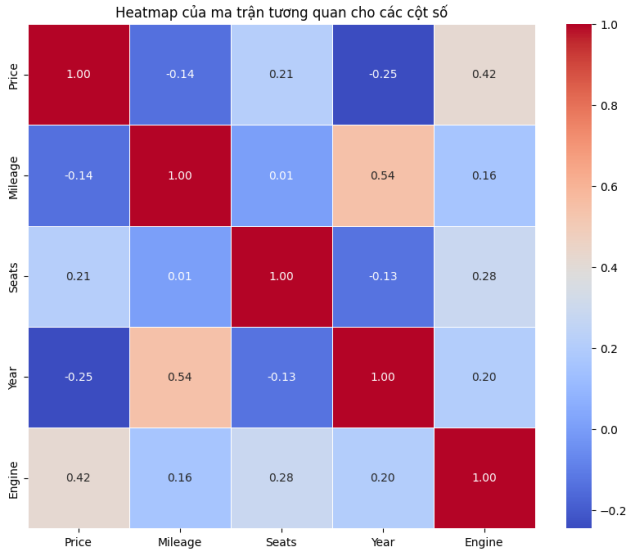


Fig. 7. Heatmap of all numeric columns

  - **Price:**
    - There is a moderate positive correlation between `Price` and `Engine` (0.37), suggesting that vehicles with larger engines tend to have higher prices.
    - `Price` also has a mild positive correlation with `Seats` (0.21), indicating that vehicles with a higher seating capacity may be priced higher.
    - A slight negative correlation with `Year` (-0.22) suggests that older vehicles tend to be less expensive.
  - **Mileage:**
    - `Mileage` shows a very low correlation with most other variables, indicating it is mostly independent in this dataset.
    - A slight positive correlation with `Year` (0.14) implies that older vehicles may have higher mileage, though the relationship is weak.
  - **Seats:**
    - `Seats` shows a mild positive correlation with `Engine` (0.27), implying that vehicles with larger engines often have more seating capacity.
    - `Seats` has negligible correlation with `Mileage` (0.00) and `Year` (-0.11), indicating that seating capacity is largely unaffected by a vehicle's age or mileage.
  - **Year:**
    - `Year` has a positive correlation with `Engine` (0.28), suggesting that newer vehicles tend to have larger engines.
    - There is also a slight negative correlation with `Price` (-0.22), indicating that older vehicles are generally priced lower.
  - **Engine:**
    - `Engine` shows positive correlations with `Price` (0.37), `Seats` (0.27), and `Year` (0.28), suggesting that vehicles with larger engines tend to be newer, have higher seating capacity, and are priced higher.

B. Visualizing

- **Scatter Plot of Mileage and Year:**

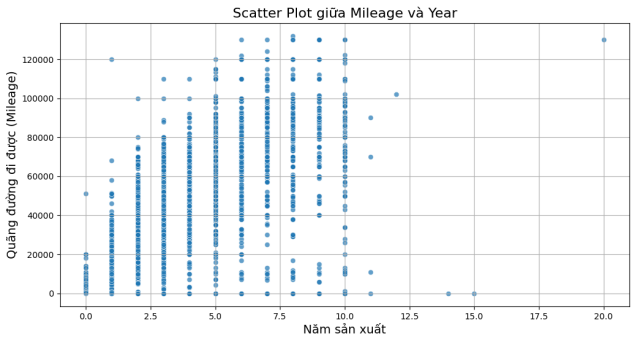


Fig. 8. Scatter plot of Mileage vs. Year of Manufacture

  - **Lack of Strong Linear Correlation:**
    - There is no discernible linear relationship between mileage and year of manufacture. This suggests that a vehicle's age is not a primary determinant of its total mileage.
  - **Year of Manufacture:**
    - Data points are relatively evenly distributed across the entire range of years, indicating that vehicles from various model years are included in the dataset.
  - **Mileage:**
    - The majority of data points cluster at lower mileage values, with a decreasing number of vehicles exhibiting higher mileage.
  - **Potential Factors Influencing Mileage:**
    - **Vehicle Type:** Different vehicle types (e.g., cars, trucks, SUVs) may have distinct usage patterns and mileage characteristics.
    - **Usage Patterns:** The intended use of a vehicle (e.g., daily commuting, long-distance travel, off-roading) significantly impacts its mileage accumulation.
    - **Maintenance Practices:** Regular maintenance and timely repairs can extend a vehicle's lifespan and influence its overall mileage.
    - **Environmental Factors:** Geographical location, climate conditions, and road quality can

affect a vehicle's wear and tear and, consequently, its mileage.

- **Vehicle Price Trends Over Time:**

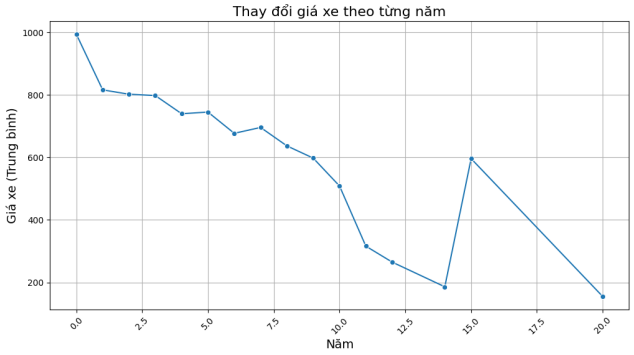


Fig. 9. Vehicle Price Trends Over Time

  - **Observations and Analysis:**
    - **Initial Price Decline:**
      - The plot shows a significant decline in average vehicle prices in the initial years. This could be attributed to various factors such as increased production, technological advancements, or changes in market dynamics.
    - **Stabilization and Fluctuations:**
      - After the initial decline, the average price stabilizes for a few years, followed by a period of fluctuations. This period reflect economic conditions, changes in consumer preferences, or shifts in fuel prices and 2011 was the big year for the KIA Morning model.
    - **Sharp Price Increase:**
      - A notable spike in average prices is observed around the 15th year. This could be due to factors like increased demand, supply shortages, or the introduction of new, high-priced models.
    - **Subsequent Decline:**
      - Following the peak, the average price experiences a sharp decline. This could be attributed to factors such as increased competition, economic downturns, or changes in consumer preferences.
  - **Limitations and Considerations:**
    - **Data Source and Quality:** The accuracy of the analysis depends on the quality and representativeness of the underlying data.
    - **External Factors:** Economic conditions, government policies, and technological advancements can significantly impact vehicle prices.

Average Price by City

- **Observations and Analysis:**
  - **Price Variation:** The chart highlights significant variations in average vehicle prices across different regions.

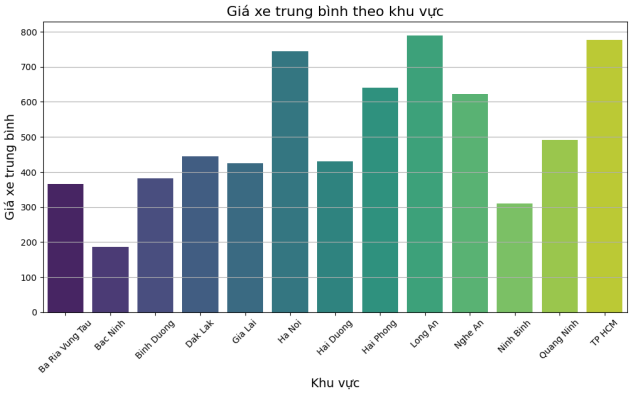


Fig. 10. Price by Location

  - **Highest Price Regions:** The highest average prices are observed in Ha Noi, Long An and TP HCM. This could be attributed to factors like higher income levels, increased demand, or the presence of luxury car dealerships in these areas.
  - **Lower Price Regions:** Regions like Ba Ria Vung Tau, Bac Ninh, and Binh Duong exhibit lower average vehicle prices. This could be due to lower income levels, lower demand, or a higher proportion of budget-friendly vehicle models.
  - **Regional Clusters:** The chart suggests potential regional clusters based on price levels. For instance, regions like Hai Duong, Hai Phong, and Long An show similar average prices.
- **Limitations and Considerations:**
  - **Data Source and Quality:** The accuracy of the analysis depends on the quality and representativeness of the underlying data.
  - **Economic Factors:** Economic factors like income levels, employment rates, and cost of living can influence vehicle prices.
  - **Vehicle Type:** The average price may vary depending on the types of vehicles prevalent in each region.

IX. Model Training and Evaluation

- Apply min-max scaling on the following columns: `Mileage`, `Engine`, `Price`, `Seats`, and `Year`.
- Drop the `Sell_Date` column.
- Use one-hot encoding on the categorical columns: `Origin`, `Body Type`, `City`, `Fuel_Type`, `Manufacturer`, and `Price_binned`.
- Convert all columns with the data type `bool` to `integer`.

Model Evaluation

Linear Regression

- **Mean Squared Error (MSE):** 7.465×10^{19}
- **R-squared (R^2):** -1.775×10^{21}
- **Observations:**
  - Poor Performance on Non-linear Data: The highly negative R^2 value indicates that linear regression struggled significantly with the non-linear relationships in the dataset.

- Sensitivity to Outliers: Linear regression's sensitivity to outliers may have drastically influenced the results, leading to the observed poor performance.

Huber Regressor

- **Mean Squared Error (MSE):** 0.00276
- **R-squared (R^2):** 0.9344
- **Observations:**
  - Robust to Outliers: Huber Regressor is less sensitive to outliers, improving performance on real-world data with anomalies.
  - Good Performance on Non-linear Data: The strong R^2 value (0.9344) indicates that it handles data that may not follow a strict linear trend effectively.

Random Forest Regressor

- **Mean Squared Error (MSE):** 0.00
- **R-squared (R^2):** 0.9675
- **Observations:**
  - Handling Non-linearity: Random forests can model complex, non-linear relationships by combining multiple decision trees.
  - Less Prone to Overfitting: Random forests reduce the risk of overfitting by averaging multiple trees.
  - Feature Importance: This model provides insights into feature importance, which is valuable for understanding the drivers of prediction.

Gradient Boosting Regressor

- **Mean Squared Error (MSE):** 0.00
- **R-squared (R^2):** 0.9426
- **Observations:**
  - High Accuracy: Gradient Boosting builds models sequentially to correct errors, which enhances predictive accuracy.
  - Feature Importance: Provides feature importance insights, similar to random forests.
  - Gradient Boosting achieves a high R^2 value of 0.9426, capturing much of the data's variability. However, it may require more tuning than Random Forests.

XGBoost Regressor

- **Mean Squared Error (MSE):** 0.00
- **R-squared (R^2):** 0.9683
- **Observations:**
  - Efficiency and Scalability: XGBoost is optimized for speed and performance.
  - Regularization: Includes regularization, which helps prevent overfitting and improves generalization.
  - High Accuracy: XGBoost achieves superior accuracy, with an R^2 score of 0.9683, making it the top-performing model.

Model Selection

- If computational resources and tuning are available, **XGBoost** is recommended for its predictive accuracy.
- For a balance between performance and interpretability, **Random Forest** is an excellent alternative, delivering high accuracy with less tuning complexity.
- **Huber Regressor** is a strong candidate if robustness to outliers and computational simplicity are priorities.

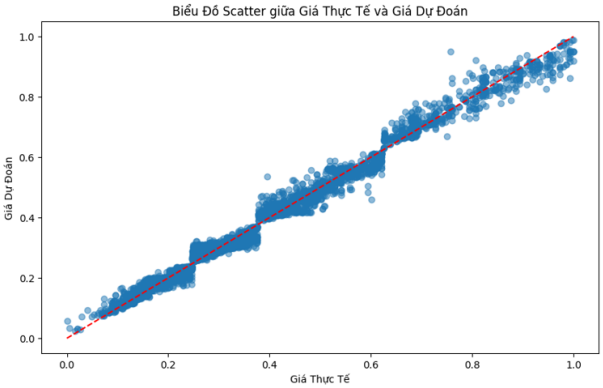
Chosen Model: Random Forest

X. Testing and Visualizing Results

A. Scatter Plot

Model Adjustments

- Lower the max-depth of the Random Forest function to avoid overfitting.
- **Scatter Plot between Actual and Predicted Prices**



Data Distribution

- Each point represents a pair of actual and predicted values for a particular data point in the test set.
- The majority of points are tightly clustered along the diagonal line, which represents perfect predictions (i.e., where actual values equal predicted values).

Performance Analysis

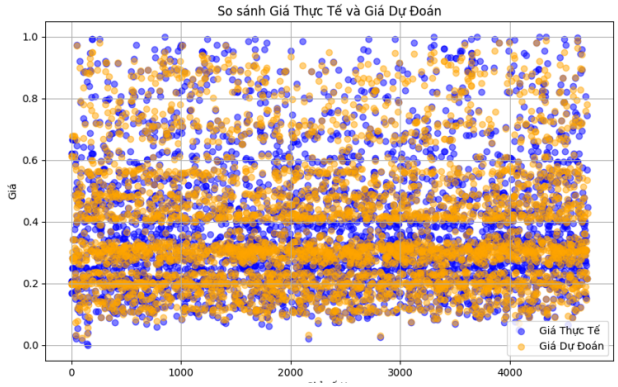
- The red dashed line on the plot represents the ideal line where predicted values would exactly match actual values. The fact that the points align closely with this line indicates that the model's predictions are accurate across a range of values.
- There are minor deviations from the line, suggesting some residual prediction errors, but these are relatively small, confirming the model's strong performance.

Outliers

- A few points deviate slightly from the diagonal line, especially in the middle range. These could be cases where the model struggled due to either outliers or complex feature interactions that it couldn't entirely capture. However, the number of such points is minimal and doesn't significantly affect overall performance.

B. Comparison Plot

- **Actual vs. Predicted Price Comparison Plot:**



Data Distribution

- **Blue Points:** Represent actual prices for each car index.
- **Orange Points:** Represent predicted prices made by the Random Forest model for each car index.
- The data appears uniformly distributed across various price levels, indicating a consistent range of values.

Model Performance

- The overlap between actual (blue) and predicted (orange) points at each car index suggests a good level of accuracy in predictions.
- The close alignment between the two sets of points shows that the model accurately captures the price patterns for most cars.

XI. Conclusion

The Random Forest Regressor was selected as the final model for predicting used car prices. Future work may include fine-tuning and advanced feature engineering.

References

[1] Oto, ”Mua bán xe,” https://oto.com.vn/mua-ban-xe.
[2] Bonbanh, ”Mua bán xe,” https://bonbanh.com/.
[3] Xebiz, ”Giá xe,” https://xebiz.vn/gia-xe/.