# **CAR PRICE PREDICTION**

Project Goal: Use Linear Regression to Predict the old car price based on collected data

Sources:<https://oto.com.vn/mua-ban-xe> [https://bonbanh.com](https://bonbanh.com/)/

<https://xebiz.vn/gia-xe/>

Data requirements: Name, price, sale date, public year, mileage, origin, body type, city, year, Fuel-type, manufacturer

Collecting data:

Library used:

Selenium: To crawdata

Pandas: Working on dataframe

Numpy: Solve math problems

· Method of crawling:

o Use selenium to look up to the first page of the site, since all sites require to press the ‘show more’ button to appear more car, we must guide the cursor to press and set up a waiting time to not get banned, then crawl all the like of each car and save into a CSV file

o Then look up each link saved crawl every detail mentioned above and store it in a new CSV file.

Storing data:

· Pyodbc, pandas

· Use pandas to read file csv

· Adjust the name for which column has the wrong format

· Handle missing values by filling all with ‘None’

· Convert some data type to String

· Use Pyodbc to set connection and create 3 tables to store newcar\_inf, used\_Car, and bonbanh\_inf

· Insert data into the table and close connections

Preprocessing data:

· Retrieving data from the SQL server

· Separate into 3 df

· New car:

o Replace the ‘None’ value

o Adjust Fuel capacity for some specific cars since it's not mentioned

§ 21 Vinfast Fadil 1.4 Tieu chuan CVT

§ 175 Toyota Venza 2.5 CVT

§ 244 Honda CR-V 1.5E 2021

§ 248 Hyundai SantaFe 2.4 Xang

§ 263 Honda CR-V 1.5G 2021

§ 273 Honda CR-V 1.5L 2021

§ 302 Toyota Fortuner 2.4 TRD AT 4X2

§ 552 BMW i8 1.5L Hybrid

§ 585 Rolls-Royce Cullinan V12

§ 586 Rolls-Royce Dawn V12

o Adjust Fuel type for some specific cars:

§ 178 Honda Civic 1.8 G

§ 245 Honda CR-V 1.5 E 2019

o Drop the ‘max\_power, torque, link’ column

o Adjust the manufacturer for some specific car

§ 464 Land Rover Range Rover Evoque 2.0L I4 Turbocha…

§ Replace ‘Maybach’ to ‘Mercedes’

§ Replace ‘Rolls Royce’ to ‘Rolls-Royce’

o Replace ‘Xe ban tai’ with ‘Ban tai’

o Remove the digit ‘cho’ in the Seating capacity column

o Remove the digit ‘cc’ in the Engine column

o round up the value for each car in the Engine column

Bonbanh\_inf:

* replace ‘None’ in the data frame with a Null value
* separate value:
  + ‘Xe moi’ and ‘Xe cu’
* format name of cars:
  + split name and version
* convert all Price column’s values to a number
* change the Date columns to integer
* remove dates that are less than ‘2000’
* rename the ‘Date’ column to ‘Public Year’
* Mileage:
  + remove ‘Km’ and ‘,’
  + change column type to Integer
* Origin:
  + replace ‘Lap rap trong nuoc’ to ‘Lap rap’
* Body type:
  + replace ‘Van/Minivan’ with ‘Van’
  + ‘Convertible/Cabriolet’ with ‘Convertible’
  + ‘Ban tai / Pickup’ with ‘Ban tai’
  + remove all the value that is ‘Truck’
* EngineType, Engine:
  + extract fuel and engine type: ‘Xang’, ‘Dau’, ‘Dien’
  + adjust some value of the Engine column
  + change all data type to integer
  + drop EngineType column
* Color
  + drop all Color and ColorInside column
* Seats, Doors:
  + remove ‘cho’
  + drop column Doors
* Manufacturer:
  + split all value
  + replace ‘Rolls’ to ‘Rolls-Royce’
  + replace ‘LandRover’ to ‘Land Rover’

Used\_cars:

* Replace the value ‘None’ to nan
* Split the title to Public year and Name of the car
  + Split the title and take the first part as year and second part as name
  + Check again to make sure everything is correct
* Format the Public year, origin, Mileage, City value to merge with others dataframe
* Clear the District because it has no impact to the model
* Check the unique value of Transmission and Fuel type
* Convert the price to normal integer

**Removing Text**: It removes the text "trieu" (Vietnamese for "million") from the string price\_str and trims any surrounding whitespace.

**Splitting the Price**: It splits the modified string into a list parts, where each word or number is separated by spaces. This helps handle cases where the price includes both billions ("ti") and millions.

**Converting Price**:

* If "ti" (meaning "billion") is in the list, it indicates the presence of a higher unit. The function handles two cases:
  + If there are two parts (e.g., "2 ti"), it multiplies the billion part by 1,000 to convert it to millions (so "2 ti" becomes 2000 millions).
  + If there are three parts (e.g., "2 ti 500"), it converts the billion part to millions and adds the remaining millions directly.
* If there's no "ti", it simply converts the remaining millions part directly.

**Return the Total**: The final total is returned as an integer, representing the entire value in millions.

* Format the manufacture
  + Replace Mercedes-Benz with Mercedes
* Fill the Body type column in the fuzzy wuzzy library
  + Use fuzzy wuzzy to check the similar score will all 3 datasets.
  + Fill the columns if a similar score > 90%
* Fill the Body type columns

Merge data:

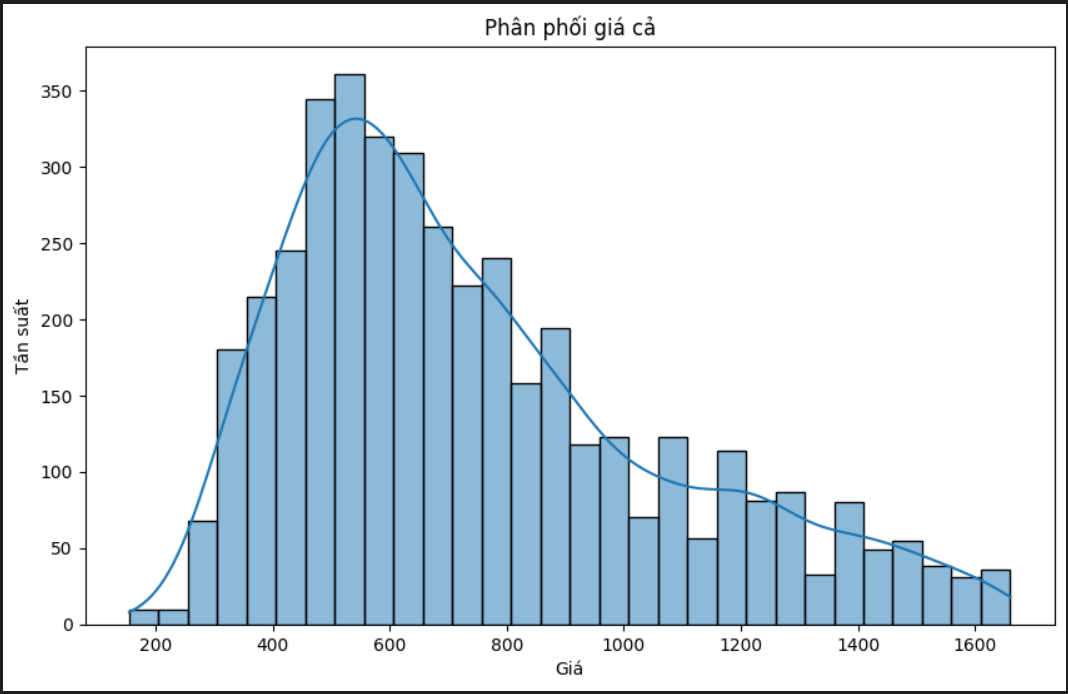
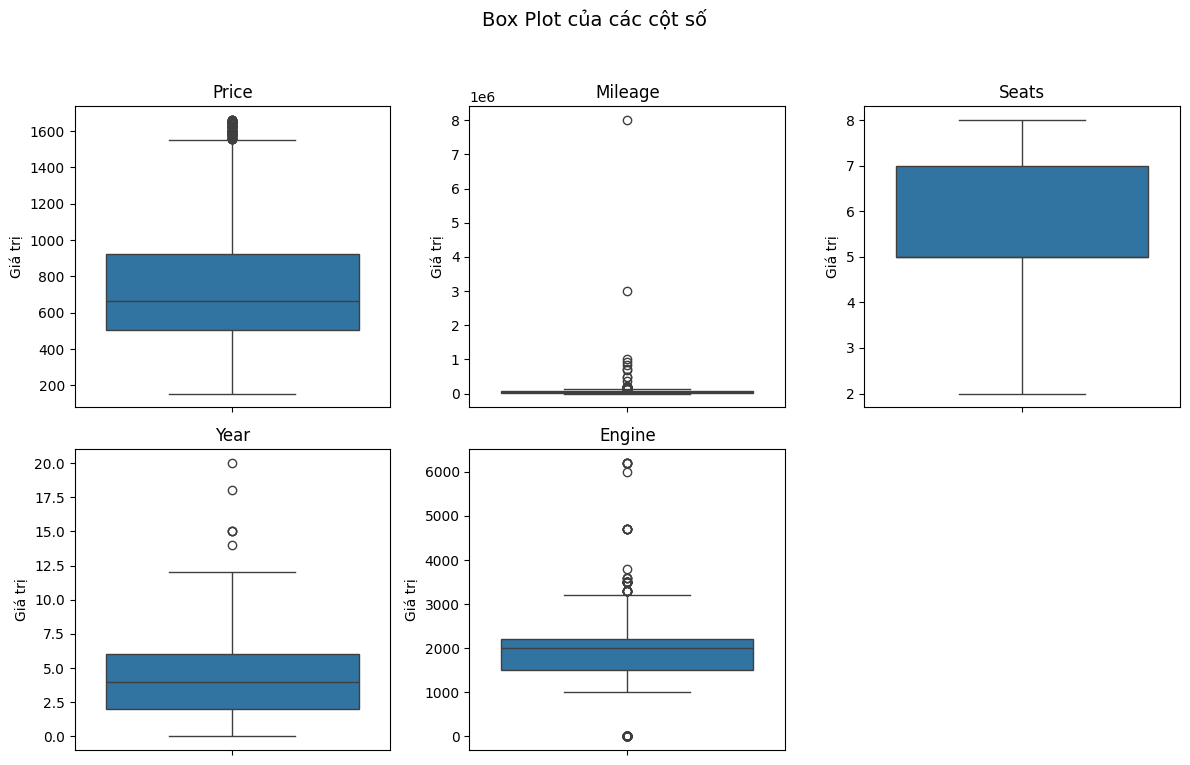
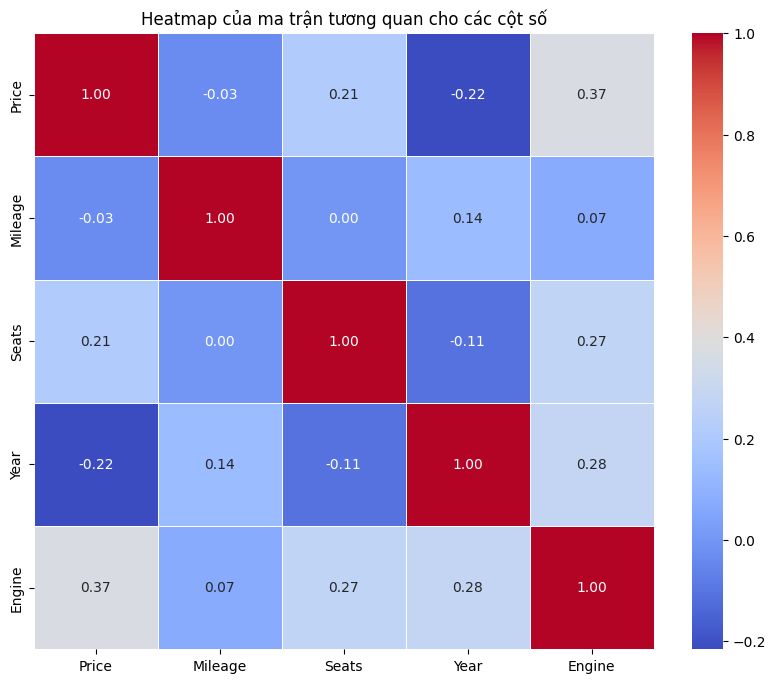
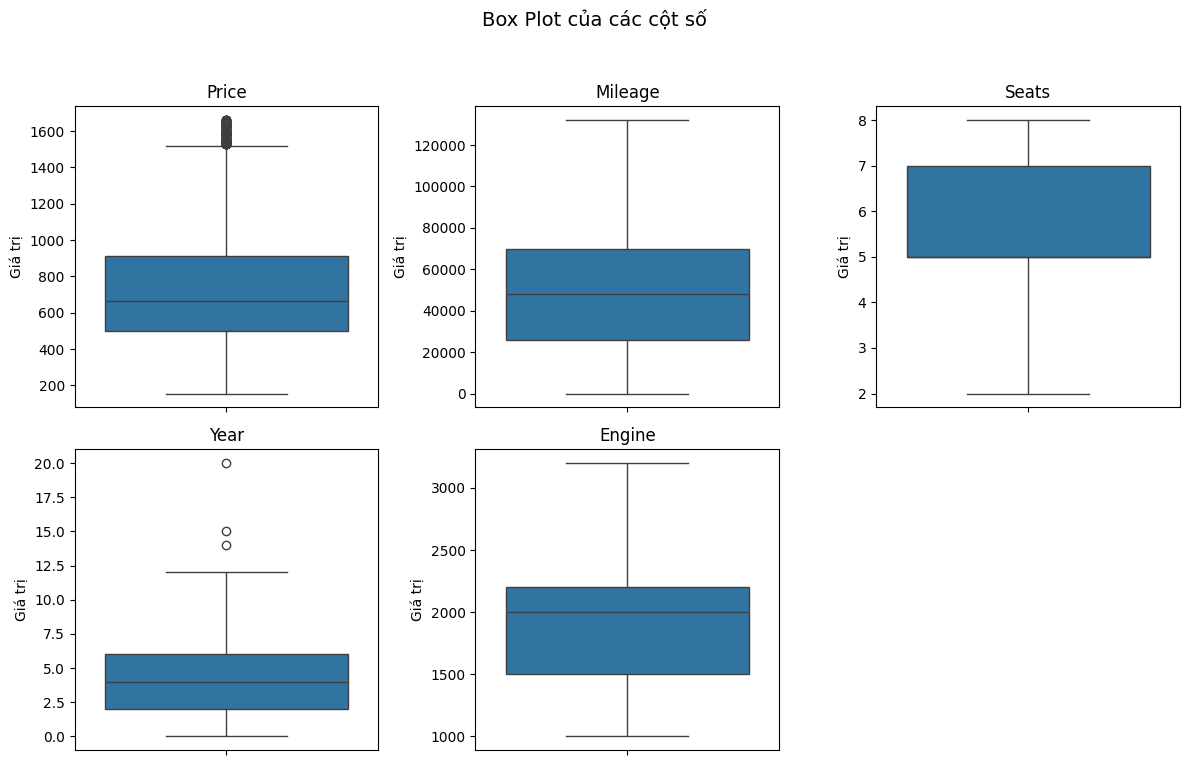
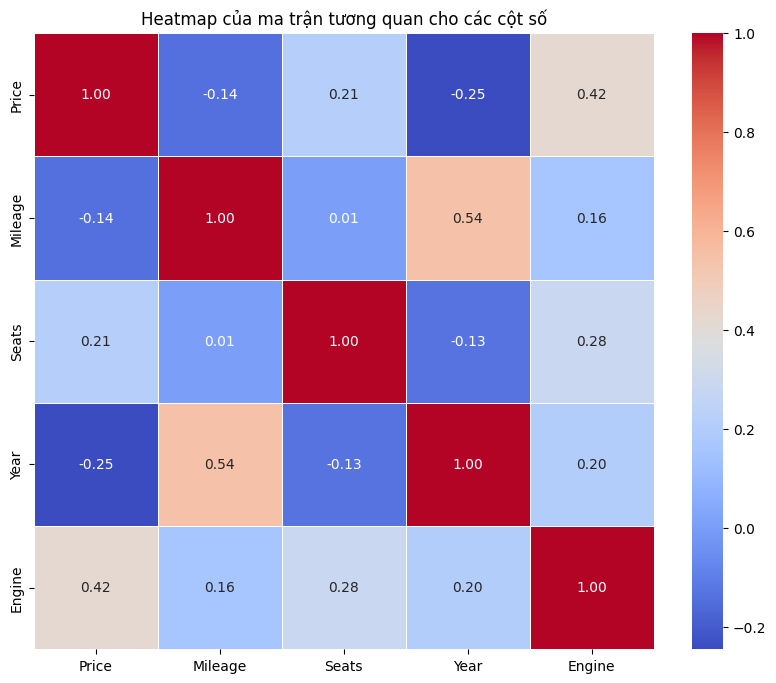
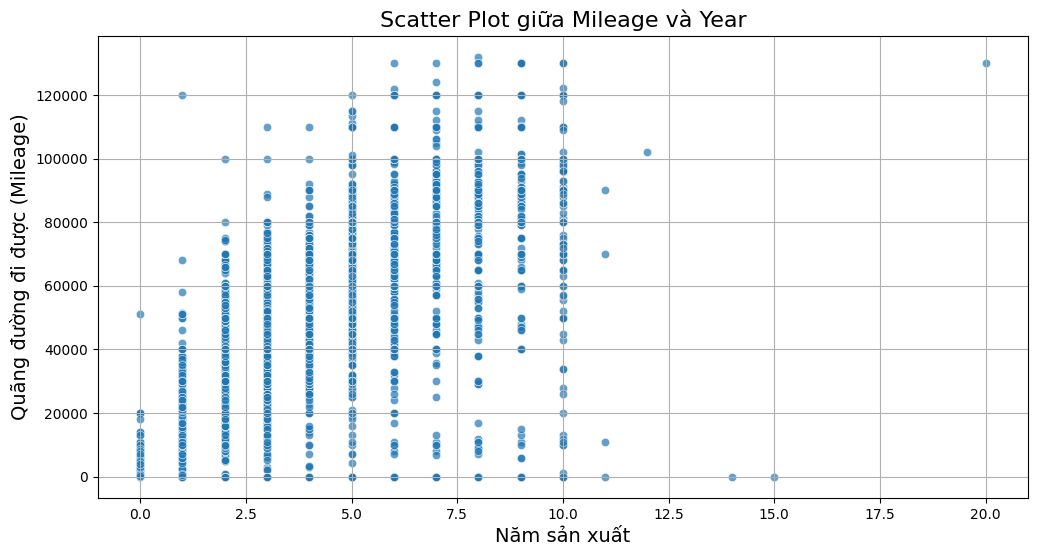
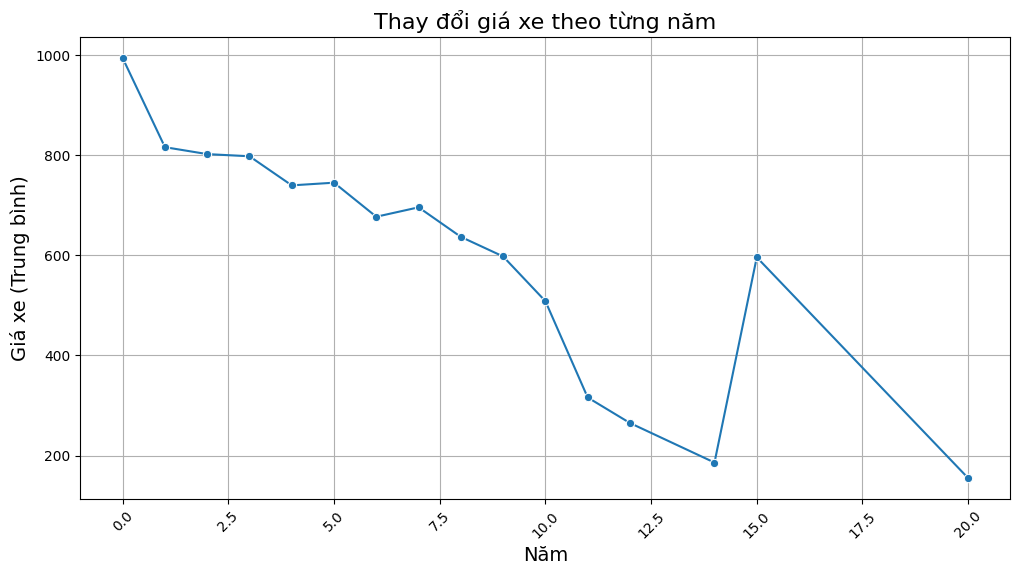
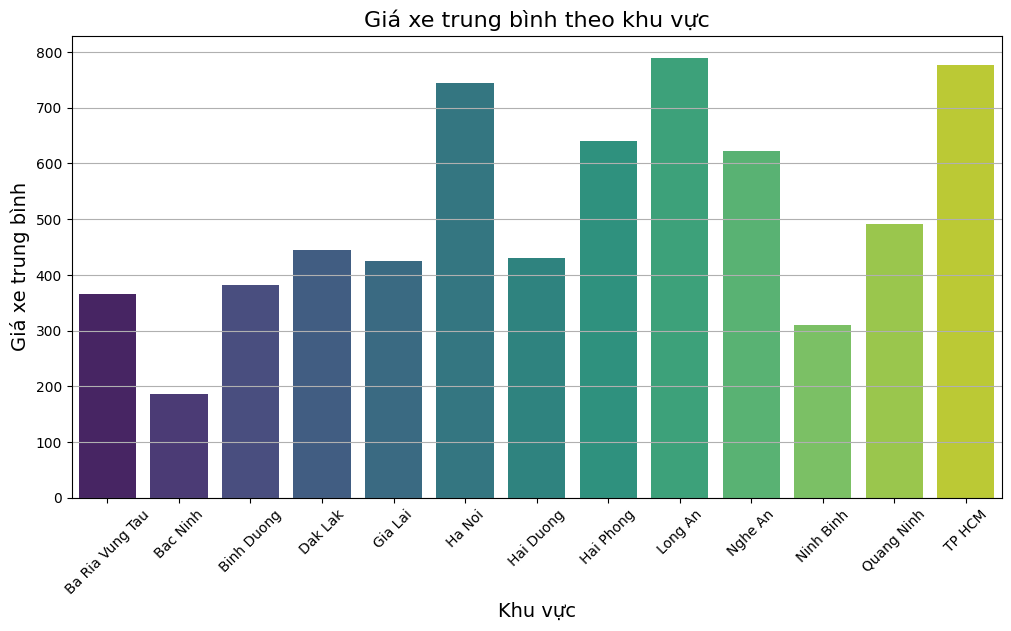
* format the column name of bonbanh\_inf:
  + ‘Bodytype’: ‘Body\_Type’
  + ‘Selldate’: ‘Sell\_Date’
  + Fueltype’: ‘Fuel\_Type’
* Create merge Keys:
  + Name
  + Public Year
  + Mileage
  + Body\_Type
  + Origin
  + City
  + Fuel\_Type
  + Sale\_Date
  + Manufacturer
  + Year
  + Price
* Merge the bonbanh\_inf data frame to usedCar by merge keys:
  + Use outer merge: all rows from both DataFrames will be included in final\_data. Rows that do not find a match in the other data frame will have NaN values in place of missing information.
* drop the ‘Seats’, ‘Engine’, and ‘Transmission’ column

Formatting

* Loop through each column 'Origin', 'Body\_Type', 'City', 'Fuel\_Type', 'Manufacturer'
* Extract unique values
* Binding:
  + set the price range: 0, 300, 800, 2000
  + binding the values in the Price column to each range following labels=['Low', 'Medium', 'High', 'Luxury'])
* Min-Max Scaling:
  + **Define the scaling function**: The function min\_max\_scaling accepts a series (a column in a data frame) and scales it.
  + **Scaling Formula**: (series - series.min()) / (series.max() - series.min())
    - This formula subtracts the minimum value of the series from each data point and divides it by the range (the difference between the maximum and minimum values).
    - The result is that values in the column are scaled between 0 and 1, where 0 represents the minimum and 1 represents the maximum.
  + **Apply Scaling**: This operation applies the min\_max\_scaling function to the Price and Mileage columns of final\_data. After this operation, both columns will contain values between 0 and 1.
* One-hot Encoding:
  + **One-Hot Encoding**: This line applies one-hot encoding to the specified categorical columns in final\_data. One-hot encoding converts categorical values into binary (0 or 1) columns. For each unique value in a categorical column, it creates a new column where:
    - The value is 1 if the row matches that category.
    - The value is 0 if it does not.
  + **Columns to Encode**:
    - Origin, Body\_Type, City, Fuel\_Type, Manufacturer, and Price-binned.
    - Each unique value in these columns will become its own column in the resulting final\_data\_encoded DataFrame.
  + **Result**: After encoding, final\_data\_encoded will have many new columns representing each category in these original columns.
  + **Select Boolean Columns**: This line identifies columns of data type bool within final\_data\_encoded. After one-hot encoding, some of the new columns may have a boolean data type. Storing them as integers can improve performance in some machine learning algorithms.
  + **Convert Boolean Columns to Integers**: This line converts each boolean column to an integer type, where True becomes 1 and False becomes 0.

Store data back to SQL server

visualizing data:

* EDA
* Remove outliers: (using IQR)
  + **Calculate the Quartiles**:
    - Q1: The 25th percentile of 'Price'.
    - Q3: The 75th percentile of 'Price'.
    - IQR: The interquartile range, calculated as Q3−Q1.
  + **Determine the Outlier Boundaries**:
    - lower\_bound: The lower threshold, set at Q1−1.5×IQR
    - upper\_bound: The upper threshold, set at Q3+1.5×IQR
  + **Filter the Data**:
    - The line final\_data[(final\_data['Price'] >= lower\_bound) & (final\_data['Price'] <= upper\_bound)] filters final\_data to retain only rows where the 'Price' is within the calculated bounds.
* Drop column ‘Public\_Year’
* Visualizing Histogram:
  + 
    - **Shape of the Distribution**: The histogram is right-skewed, with most of the data concentrated on the lower price range (around 400–800) and tapering off as the prices increase. This suggests that there are more entries in the dataset with lower prices, and fewer entries as the price rises.
    - **Frequency**: The y-axis shows the frequency (Tần suất) of prices within each bin. The highest frequency appears to be around the 500–600 price range, with over 350 entries in that range.
    - **Distribution Curve**: The smooth line over the histogram represents a density estimation, helping to visualize the distribution's shape more clearly. The peak aligns with the highest frequency bins, showing a general trend that prices in this dataset are mostly on the lower end.
* Box plot of all numeric columns:
  + 
    - **Price**:
      * The 'Price' variable is mostly concentrated between 400 and 1100, with a median of 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 has 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 the 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 in earlier years, with a few high outliers representing newer models.
      * This suggests that the dataset contains mostly older vehicles, with some recent additions that may stand out as outliers.
    - **Engine**:
      * The 'Engine' size variable has multiple high outliers, particularly 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 for the dataset.
* heatmap for correlation matrix of 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 show a mild positive correlation with Engine (0.27), implying that vehicles with larger engines often have more seating capacity.
      * Seats have 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 be priced higher.
* Use IQR to remove outliers for ‘Mileage’
* Remove all data that is < 5000 in ‘Engine’
* Use IQR to remove outliers for ‘Engine’
* Box plot after removing outliers:
  + 
* Heatmap after removing outliers:
  + 
* Visualizing:
* Scatter plot of Mileage and Year:
  + 
    - **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.
    - **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:
  + 
    - 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 might reflect economic conditions, changes in consumer preferences, or shifts in fuel prices.
      * **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.
      * **Highest Price Regions:** The highest average prices are observed in **TP HCM** and **Quang Ninh**. 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.

Training:

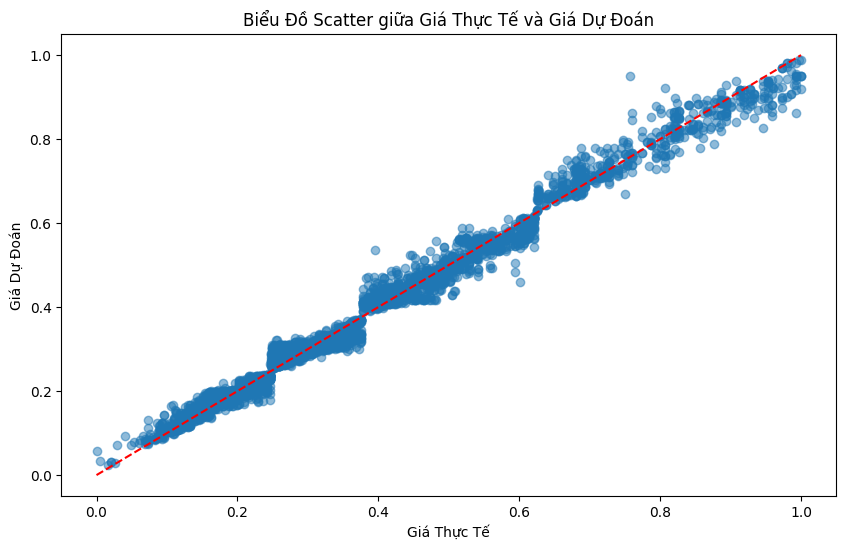
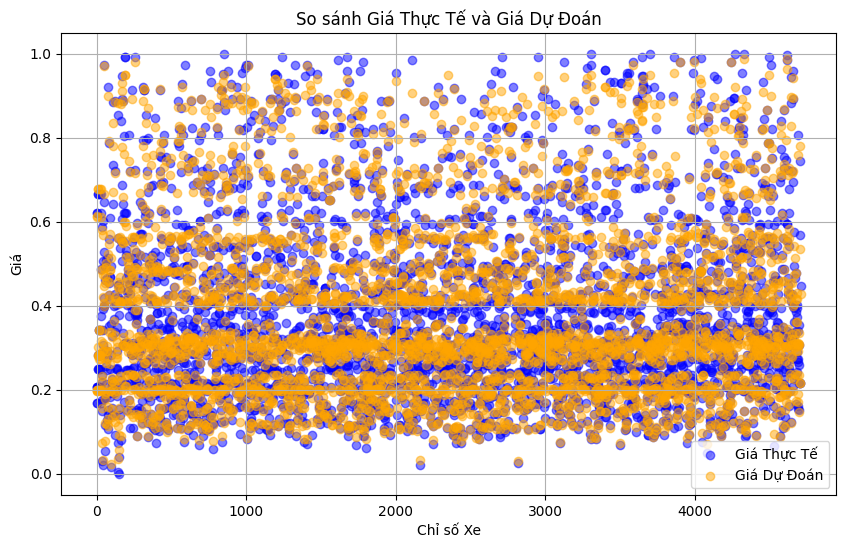
* Use min-max scaling on ‘Mileage’, ‘Engine’, ‘Price’, ‘Seats’, ‘Year’
* Drop column Sell\_Date
* Use one-hot encoding on ‘Origin’, ‘Body Type’, ‘City’, ‘Fuel\_Type’, ‘Manufacturer’, ‘Price\_binned’
* **Format all columns with the data type ‘bool’ to ‘integer’**
* Linear regression:
  + **Mean Squared Error (MSE)**: 7.465×10^19
  + **R-squared (R²)**: −1.775×10^21
    - **Poor Performance on Non-linear Data**: Given the highly negative R² value, it appears that linear regression struggled significantly with the non-linear relationships in the dataset.
    - **Sensitivity to Outliers**: Linear regression is sensitive to outliers, which could drastically influence the results, leading to the poor performance observed.
* Huber regressor:
  + **MSE**: 0.00276
  + **R-squared (R²)**: 0.9344
    - **Robust to Outliers**: Less sensitive to outliers. This robustness often improves performance on real-world data, which frequently contains anomalies.
    - **Good Performance on Non-linear Data**: It performs well on data that may not strictly follow a linear trend, as seen by the strong R² value (0.9344).
  + With a high R² value and a very low MSE, the Huber Regressor performs significantly better than linear regression on this dataset. Its robustness to outliers helps in improving predictive accuracy.
* Random Forest Regressor
  + **MSE**: 0.00
  + **R-squared (R²)**: 0.9675
    - **Handling Non-linearity**: Random forests can model complex, non-linear relationships by aggregating multiple decision trees.
    - **Less Prone to Overfitting**: Random forests reduce the risk of overfitting compared to individual decision trees by averaging multiple trees.
    - **Feature Importance**: This model can provide insights into feature importance, which can be valuable for understanding which factors drive the prediction.
  + With an R² score of 0.9675, random forests deliver excellent predictive power for this dataset, capturing non-linear relationships effectively. This model balances robustness and predictive accuracy but at the cost of interpretability and computation.
* Gradient Boosting Regressor:
  + **MSE**: 0.00
  + **R-squared (R²)**: 0.9426
    - **High Accuracy**: Gradient Boosting builds models sequentially to correct errors, which can result in high predictive accuracy.
    - **Feature Importance**: Similar to random forests, gradient boosting provides feature-importance insights, which can be valuable for understanding key drivers in the data.
  + Gradient Boosting provides a high R² value of 0.9426, indicating it captures much of the data’s variability. However, it may be more sensitive to tuning than Random Forests, which makes it slightly less practical in situations with limited computational resources.
* XGBoost Regressor**:**
  + **MSE:** 0.00
  + **R-squared (R²):** 0.9683
    - **Efficiency and Scalability:** XGBoost is optimized for speed and performance, often faster than other boosting techniques.
    - **Regularization:** XGBoost includes regularization, which helps prevent overfitting and improves generalization on test data.
    - **High Accuracy:** XGBoost often yields superior accuracy, making it a popular choice in machine learning competitions.
  + XGBoost achieved the best R² score (0.9683), making it the top-performing model for this dataset. However, its computational demands and need for tuning make it more suitable when prediction accuracy is prioritized over simplicity and computational efficiency.

Based on the results:

1. **If computational resources and tuning are available**, XGBoost provides the best predictive accuracy and should be selected for the final model.
2. **For a balance between performance and interpretability**, Random Forest is an excellent alternative, delivering high accuracy with less tuning complexity.
3. **Huber Regressor** is also a strong candidate if robustness to outliers and computational simplicity are priorities.

=> **Choose Random Forest**

Testing and visualizing results:

* Lower the max-depth learning of the Random Forest function to avoid **Overfitting**
* Scatter plot between **actual prices** 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**:
    - he 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.
* 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.