Car Price Prediction Project

1 Project Goal

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.

1.1 Sources

- `https://oto.com.vn/mua-ban-xe`
- `https://bonbanh.com/`
- `https://xebiz.vn/gia-xe/`

2 Data Requirements

The required data fields include:

- Name
- Price
- Sale date
- Public year
- Mileage
- Origin
- Body type
- City
- Year
- Fuel type
- Manufacturer

3 Collecting Data

3.1 Libraries Used

- Selenium: for web scraping
- Pandas: for handling dataframes
- Numpy: for mathematical operations

3.2 Crawling Method

- Use Selenium to navigate to the first page of each site. For sites that require clicking “Show more” to load additional data, automate the clicking and wait to avoid bans.
- Gather links for each car and save them in a CSV file.
- Use the links to collect detailed information for each car and store it in a new CSV file.

4 Storing Data

- Use PyODBC and Pandas to connect to the SQL server and read CSV files.
- Rename columns with incorrect formats and handle missing values by filling them with ”None”.
- Convert certain data types to strings as needed.
- Create three tables: `newcar_inf`, `used_Car`, and `bonbanh_inf` to store data and close connections after insertion.

5 Preprocessing Data

- Retrieve data from the SQL server and split into three dataframes.
- Clean and adjust specific fields in each dataset, such as fuel capacity, manufacturer names, and body types.
- Replace inconsistent or missing values, convert strings to numerical types, and standardize formats across dataframes.

6 Data Cleaning for Each Dataset

6.1 New Car Data

- Replace ”None” with appropriate values.
- Adjust fuel capacity for specific cars and fuel types where missing.
- Drop unnecessary columns like `max_power`, `torque`, and `link`.
- Standardize manufacturer names and seating capacity.

6.2 Bonbanh_inf Data

- Replace ”None” with NULL values.
- Convert price columns to numerical values and ensure date columns are integers, renaming ”Date” to ”Public Year”.
- Standardize formats for mileage, origin, body type, engine, and manufacturer.

6.3 Used Cars Data

- Replace ”None” values with NaN and format fields for merging.
- Convert the price to integer values after removing unnecessary text.
- Use Fuzzy Wuzzy to check and fill similar values across datasets if a similarity score is above 90%.

7 Data Merging

- Format column names for merging.
- Define merge keys: Name, Public Year, Mileage, Body_Type, Origin, City, Fuel_Type, Sale_Date, Manufacturer, Year, Price.
- Use an outer merge to include all rows from both datasets. Drop redundant columns.

8 Feature Engineering

8.1 Binning

Set price ranges and categorize prices into bins: Low, Medium, High, and Luxury.

8.2 Scaling

Define a min-max scaling function:

$$\text{scaled\_value} = \frac{\text{value} - \text{min}}{\text{max} - \text{min}}$$

Apply scaling to the Price and Mileage columns, setting values between 0 and 1.

8.3 Encoding

- Apply one-hot encoding to categorical columns such as Origin, Body_Type, City, Fuel_Type, Manufacturer, and Price-binned.
- Convert Boolean columns to integers (1 for True, 0 for False).

9 Data Visualization and EDA

9.1 Outlier Removal (using IQR)

- Calculate Quartiles (Q1 and Q3) and determine the Interquartile Range (IQR).
- Define bounds as $Q1 - 1.5 \times \text{IQR}$ and $Q3 + 1.5 \times \text{IQR}$.
- Filter data within these bounds.

9.2 Histograms

Describe price distribution with a right-skewed histogram, showing most data in the lower price range.

9.3 Box Plot of Numeric Columns

- Summarize price, mileage, seats, year, and engine distributions.
- Highlight high-value outliers in Price and Engine columns.

9.4 Correlation Matrix Heatmap

Show correlations among numeric columns, highlighting:

- Positive correlation between Price and Engine size.
- Negative correlation between Year and Price, indicating older vehicles are generally cheaper.

10 Model Training and Evaluation

10.1 Preprocessing for Model

- Scale Mileage, Engine, Price, Seats, and Year using min-max scaling.
- Drop the `Sell_Date` column.
- Apply one-hot encoding and format Boolean columns as integers.

10.2 Model Comparison

- **Linear Regression**: Poor performance, high sensitivity to non-linear relationships and outliers.
- **Huber Regressor**: High robustness to outliers, good performance with MSE = 0.00276, R^2 = 0.9344.
- **Random Forest**: Excellent performance on non-linear data, with MSE = 0.00, R^2 = 0.9675.
- **Gradient Boosting**: High accuracy but more sensitive to tuning, MSE = 0.00, R^2 = 0.9426.
- **XGBoost**: Best performance, MSE = 0.00, R^2 = 0.9683, but computationally intensive.

10.3 Final Model Selection

For a balance between accuracy and computational efficiency, Random Forest is chosen as the final model.

11 Testing and Visualizing Results

11.1 Scatter Plot

Create a scatter plot between actual prices and predicted prices:

- Most points lie close to the diagonal, indicating accurate predictions.
- Minor deviations indicate residual prediction errors, but they are minimal.

Figure 1: Scatter plot between actual and predicted prices

11.2 Actual vs. Predicted Price Comparison Plot

Compare actual prices (blue points) and predicted prices (orange points) for each car index, showing a consistent distribution across various price levels.

Figure 2: Comparison of actual vs. predicted prices by car index

12 Conclusion

The Random Forest Regressor was selected as the final model for predicting used car prices due to its balance of accuracy and interpretability. Future improvements could involve fine-tuning and exploring advanced feature engineering.