Iteration 4

Xiaoyang Fei, Jack Carpini, Maalolan Bharaniraj

Mar 19, 2025

1 Objectives and Hypothesis

Our objective, and subsequent model training, was based around three possible datasets that we created. Our three datasets consisted of the raw data with no inflation adjustment, a column-wise inflation adjustment, and a row-wise inflation adjustment. Our task was to figure out which dataset performed the best during modeling so we would know which one to move forward with. Our hypothesis here was that model performance was related to dataset creation methods. To test this, we tested the same models on each dataset and compared the performance using RMSE. Digging down to the individual datasets, each dataset was clustered using KMeans. Next, all models were tested on each cluster and RMSE was stored for performance comparison.

2 Experimental Setup

All code for our project was written in Python, writing and testing code in Jupyter notebooks with Google Colab. All model training was conducted on our local machine setups on CPU. No GPU was used to train any models during our experimentation. For our data preprocessing, clustering, model training, and evaluation, we utilized a variety of libraries in Python, ensuring efficiency and reproducibility. Below is a breakdown of the key libraries used in our project:

2.1 Data Processing and Manipulation

- **pandas** – Used for handling and manipulating structured data, including merging datasets and performing inflation adjustments.
- **numpy** – Provided numerical operations and matrix computations, essential for feature transformations.

2.2 Data Preprocessing

- **scikit-learn** – Used for:

  - Train-test split (train_test_split) to ensure proper dataset partitioning.
  - Feature scaling (StandardScaler, MinMaxScaler) for normalizing numerical columns.
  - KMeans clustering (KMeans) for segmenting the dataset into distinct groups.
  - Principal Component Analysis (PCA) for visualizing cluster distributions.

2.3 Machine Learning Models

- **scikit-learn** – For traditional regression models:
  - Lasso regression (Lasso) for feature selection and regularized linear modeling.
  - Ridge regression (Ridge) to handle collinearity in the dataset.
- **LightGBM** – Used for training LightGBM, which emerged as the best-performing model.
- **XGBoost** – Utilized for XGBoost models, compared against LightGBM.
- **Pytorch** – Applied for developing and training feedforward neural networks as a deep learning alternative.

2.4 Model Evaluation and Metrics

- **scikit-learn** – For computing model evaluation metrics:
  - Root Mean Squared Error (RMSE) (mean_squared_error)
  - R-squared (R^2 score) (r2_score)
  - Cross-validation (KFold, cross_val_score) to assess generalization.

2.5 Data Visualization

- **matplotlib** – Used for plotting RMSE comparisons, cluster visualizations, and model residuals.
- **seaborn** – Provided advanced statistical plots, such as pair plots and correlation heatmaps, to analyze feature relationships.
- **plotly** – Used for interactive visualizations, especially for PCA-based cluster representations.

3 Dataset Processing

We used the `train_test_split` function from Scikit-Learn to partition each dataset into **80% training** and **20% testing**. Initially, all models were run with base configurations to evaluate performance before hyperparameter tuning.

3.1 Handling High-Dimensional Data

For the dataset without inflation adjustment, we encountered memory limitations when working with the raw merged data. To address this:

- **Downcasting Numeric Columns**: Reduced memory usage by converting numerical columns to lower precision types.
- **Merging Data**: The dataset was created by merging two datasets with different formats and columns, leading to **high dimensionality and cardinality**.
- **Sampling & Memory Reduction**: Due to its large size, we applied sampling techniques and memory-efficient transformations.

3.2 Clustering Approach

- **Unadjusted Dataset**: We applied **KMeans clustering** using the **elbow method** to determine the optimal number of clusters. The best value for k was between **3 and 5**, with **5 clusters** yielding the best performance.
- **Inflation-Adjusted Dataset**: Standard **KMeans clustering** was used to group the data into **3 clusters**.

3.3 Regression Models

For both datasets, we trained **Lasso** and **Ridge regression** models:

- **Preprocessing**: Categorical columns were **encoded**, and numerical columns were **scaled**.
- **Ridge Regression Optimization**: After an initial run, the **alpha** parameter was incrementally adjusted to improve performance:
  - **Unadjusted dataset**: Alpha was **tuned manually**.
  - **Inflation-adjusted dataset**: Alpha was increased from **1 to 10** to prevent ill-conditioned matrix warnings.

3.4 LightGBM Training

- **Unadjusted Dataset**: LightGBM was trained for each cluster with:
  - `n_estimators` = 100
  - `learning_rate` = 0.1
- **Inflation-Adjusted Dataset**: LightGBM was trained with:
  - `num_leaves` = 50
  - `n_estimators` = 50
  - `learning_rate` = 0.5

3.5 Deep Learning Approach

For the inflation-adjusted dataset, we implemented a **feedforward neural network** with:

- **Three Hidden Layers**:
  - (10, 128)
  - (128, 64)
  - (64, 1)
- **Activation Function**: ReLU

3.6 Inflation Adjustment for Price Normalization

To ensure price comparisons across different years, we adjusted selling prices for each **year, make, model, and trim** to reflect **2024-equivalent prices**. After adjustment:

- **KMeans clustering** segmented the dataset into **two clusters** based on numerical attributes.
- **XGBoost & LightGBM models** were trained separately for each cluster.

3.7 Hyperparameter Tuning

To improve model performance, we tuned the following parameters:

- **XGBoost**:
  - Learning rate
  - Maximum depth
  - Gamma regularization

- **LightGBM**:
  - `num_leaves`
  - `learning_rate`
  - Boosting rounds (grid search optimization)

4 Results

4.1 Raw unadjusted merged dataset results

Test Result:

Cluster	**Model**	**RMSE**	$\mathbf{R^2}$
0	Lasso Regression	7412.82	0.29
0	Ridge Regression	7412.82	0.29
1	Lasso Regression	1737.58	0.97
1	Ridge Regression	1737.58	0.97
4	Lasso Regression	9866.98	-0.37
4	Ridge Regression	9764.99	-0.34
0	LightGBM	5133.30	0.66
1	LightGBM	1733.18	0.97
4	LightGBM	8388.16	0.01

Table 1: Raw Unadjusted Merged Dataset Results

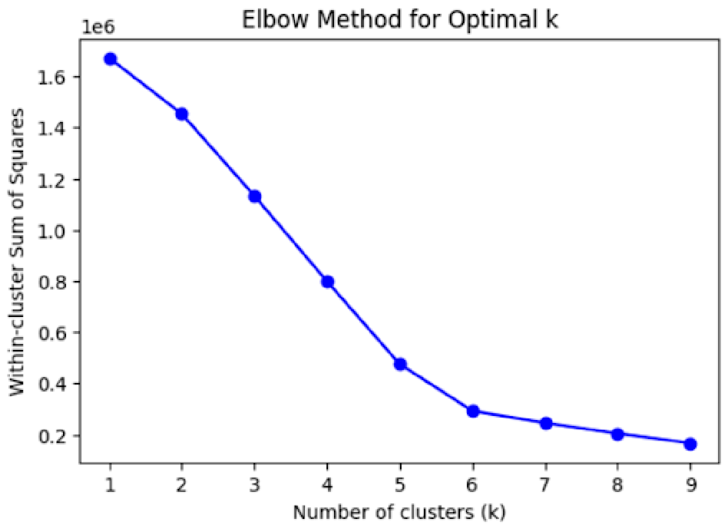

Figure 1: KMeans Clusters (PCA)

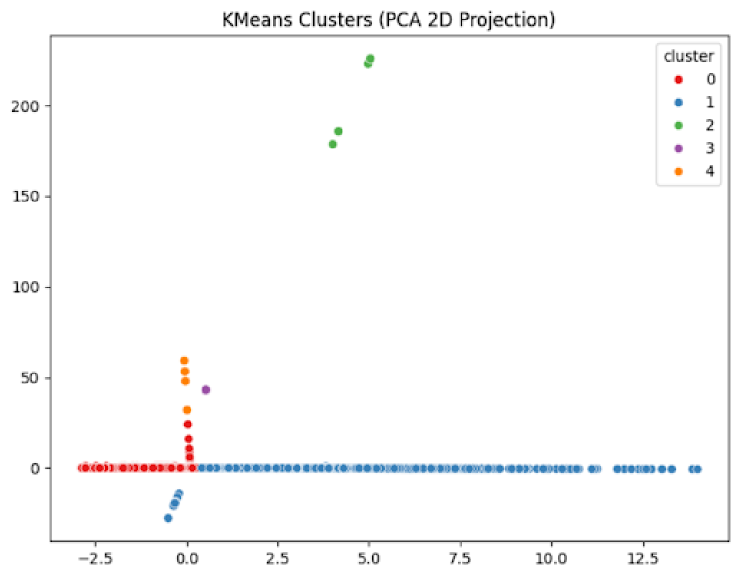

Figure 2: Elbow Method for Optimal k

4.2 Column Adjusted Inflation Results

Test Result:

Cluster	**Model**	**RMSE**	**R^2**
0	Lasso Regression	2520.36	0.961
0	Ridge Regression	2520.34	0.961
0	LightGBM	2472.27	0.962
1	Lasso Regression	17302.984	0.0037
1	Ridge Regression	17302.988	0.0037
1	LightGBM	11013.39	0.596
2	Lasso Regression	2085.54	0.935
2	Ridge Regression	2085.56	0.935
2	LightGBM	1665.10	0.958

Table 2: Results for Column Adjusted Inflation Dataset

After running K-Fold Cross Validation for 5 folds, the following results were yielded:

Cluster	Mean RMSE of K-Fold Scores
0	2327.85
1	9248.50
2	1778.62

Table 3: Mean RMSE of K-Fold Scores for Each Cluster

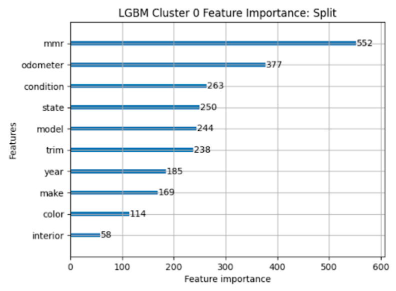

Figure 3: Caption for Image 1

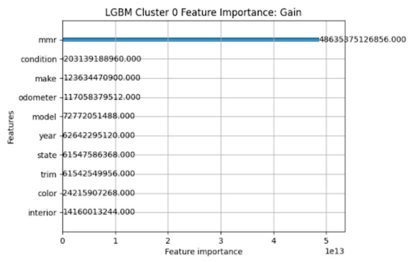

Figure 4: Caption for Image 2

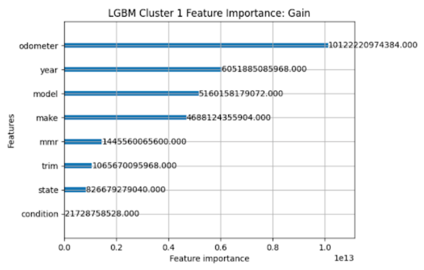

Figure 5: Split and Gain plots

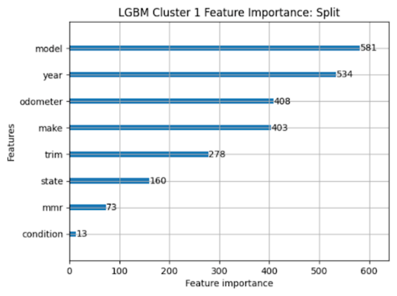

Figure 6: Split and Gain plots

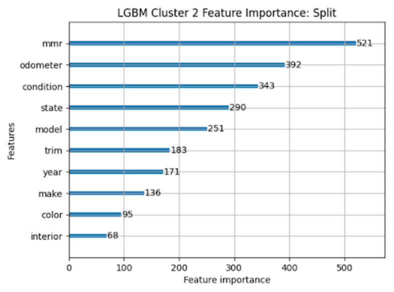

Figure 7: Split and Gain plots

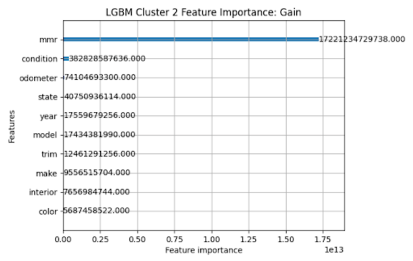

Figure 8: Split and Gain plots

4.3 Row-wise Inflation Adjustment Results

Test Results:

Cluster R²	Model	RMSE
0 0.912	XGBoost	2525.35
0 0.965	LightGBM	2520.34
1 0.872	XGBoost	2084.13
1 0.921	LightGBM	2587.11
2 0.932	XGBoost	2084.13
2 0.933	LightGBM	2087.20

Table 4: Results for Row-wise Inflation Adjustment Dataset

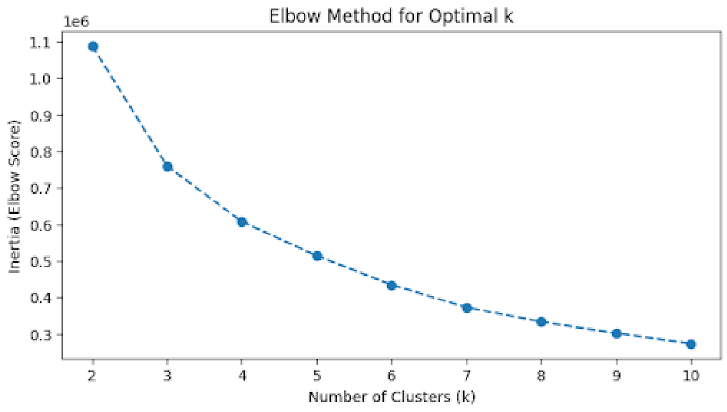

Figure 9: Split and Gain plots

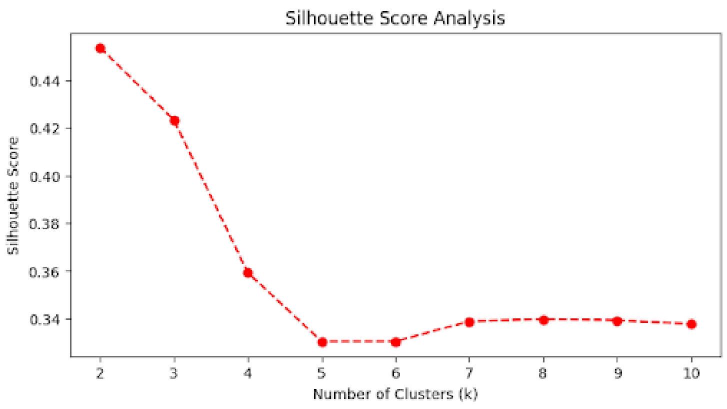

Figure 10: Split and Gain plots

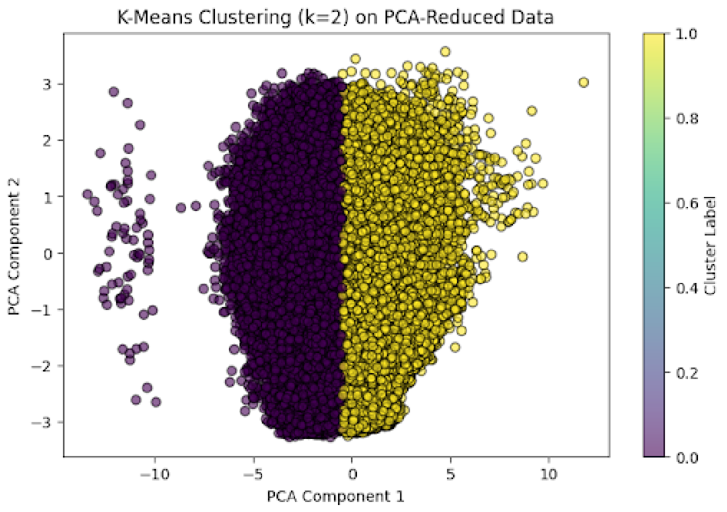

Figure 11: Split and Gain plots

5 Evaluation and Metrics

For all models, we used root mean squared error (RMSE) for comparison. This allowed us to easily compare linear and non-linear model performance while also providing a metric that is easy to interpret in the context of our problem. Additionally, we included R^2 to measure the variation in Y explained by X. For validation, we applied K-fold cross-validation to evaluate how well our models generalize to new test data.

6 Discussion

Initially, our dataset contained a price column reflecting the original sale price in the year the car was sold. This posed a challenge, as inflation generally causes prices to increase over time. Cars sold in 2014, for example, tend to have lower prices than those sold in 2024. To address this issue, we created three versions of the dataset:

- **Raw Unadjusted Dataset**: Contains all features with unadjusted prices.
- **Column Adjusted Inflation Dataset**: Includes a new column with inflation-adjusted 2024 sale prices, with the original price column removed.
- **Row-wise Inflation Dataset**: Adjusts the price for each row to reflect 2024-equivalent values.

Once we finalized these datasets, we began modeling and tuning our models. We first implemented baseline versions of three models: **Lasso Regression**, **Ridge Regression**, and **LightGBM**. After obtaining initial test results, we tuned model parameters to improve performance.

6.1 Raw Unadjusted Dataset

For the original raw merged dataset, we adjusted clustering and modeling parameters to enhance performance. Initially, clustering with $k = 3$ led to poor model performance, with some clusters producing NaN results. Using the elbow method, we determined that $k = 5$ yielded the best results. Some clusters achieved 90% accuracy, while two continued to return NaN, and others remained below 30% accuracy.

Lasso Regression adjustments made no difference, so it was kept as is. **Ridge Regression** alpha was increased from 1 to 10, resulting in only minor improvements, with R^2 increasing from approximately 20% to 29%. **LightGBM** showed better potential—after increasing the number of leaves from 25 to 50, accuracy for one cluster improved from 30% to 60%, although two clusters still returned NaN values. Ultimately, we determined that this dataset was not ideal for accurately predicting car prices, leading us to explore better data adjustments.

6.2 Column Adjusted Inflation Dataset

For the column-adjusted inflation dataset, after running baseline models, we fine-tuned parameters to improve performance:

- **Lasso Regression**: Performed well on two out of three clusters with default parameters, so no tuning was necessary.
- **Ridge Regression**: The alpha parameter was incrementally increased from 1 to 10 to prevent an ill-conditioned matrix warning and improve performance, aligning it with Lasso Regression results.
- **LightGBM**: Performed well on two clusters with default parameters. The underperforming cluster was optimized, reducing RMSE to 8879 and increasing R^2 to 0.65.

Further hyperparameter tuning is ongoing, alongside the development of a deep learning model for testing.

6.3 Row-wise Inflation Dataset

For the row-wise inflation dataset, we refined model hyperparameters to improve performance:

XGBoost Adjustments

- Learning rate: $0.03 \rightarrow 0.05$
- Max depth: $6 \rightarrow 8$
- Gamma: $0 \rightarrow 0.1$ (added regularization)
- Subsample: 0.8 (to prevent overfitting)
- Colsample by tree: 0.7 (to improve generalization)

LightGBM Adjustments

- Num leaves: $31 \rightarrow 50$ (for better feature interactions)
- Learning rate: 0.05
- Max depth: 7
- Boosting rounds: 150 (to optimize performance)

7 Conclusion

After evaluating multiple approaches, we concluded that the **2024 price column-adjusted method with clustering** is the most effective strategy for predicting used car prices. The reasons are as follows:

7.1 1. Standardization of Prices Across Time

Car prices fluctuate over time due to inflation and market conditions. The raw dataset contained vehicles sold in different years with original sale prices, making direct comparisons between old and new transactions unreliable. By adjusting all sale prices to 2024-equivalent values, we ensured that the model learned from real price trends rather than outdated values. This adjustment reduced noise and created a more normalized price column, which likely improved model performance.

7.2 2. Improved Model Accuracy with Cluster-Specific Predictions

Car pricing is influenced by numerous factors, such as brand, mileage, and condition. A single regression model struggles to capture these complex relationships. To address this, we clustered data using numerical attributes (e.g., mileage, age, condition) before training separate models for each cluster.

This clustering approach offered two key advantages:

- **More Homogeneous Groups**: Each cluster contained similar vehicles, simplifying the modeling process.
- **Better Predictive Performance**: Training specialized models per cluster minimized RMSE and improved R^2 across all subgroups.

7.3 3. LightGBM as the Best Performing Model

Extensive testing showed that **LightGBM** consistently outperformed other models (XGBoost, Lasso, Ridge Regression) by:

- Achieving lower RMSE and higher R^2.
- Demonstrating stronger generalization across clusters.
- Being computationally efficient compared to deep learning models.
- Handling outliers and non-linear relationships more effectively.

7.4 4. Handling Outlier Clusters

Certain clusters exhibited higher price variability due to rare car models or extreme mileage values. Using a clustering-based approach allowed us to fine-tune models for underperforming clusters separately, improving overall predictive accuracy.