Predicting Prices for Used Cars

Julios Fotiou
Andreas Hadjoullis
Computer Science
University of Cyprus

I. Goals

The automotive market has a wide range of car prices depending on factors such as brand, age, mileage, and vehicle type. Predicting car prices accurately is valuable for both buyers and sellers. This project aims to build a predictive model using structured automotive data.

Our dataset is derived from Kaggle in which we have a comprehensive collection of valuable data about used cars, and provides insight into how the cars are being sold, what price they are being sold for, and all the details about their condition.

This project focuses on predicting car prices using machine learning models trained on our real-world dataset. To improve prediction quality, extreme values outside the practical range [1,000 – 200,000] are excluded. Various regression models are evaluated using standardized preprocessing and feature selection pipelines. We have also tested various classification algorithms on our dataset, after splitting our dataset into bins, decided by their price.

The main goals of the project are:

- Understand our dataset.
- Clean and preprocess raw car listing data.
- Select relevant features and remove noise.
- Train and compare multiple regression and classification models.
- Evaluate model performance using robust statistical metrics.

II. Approach

A. Exploratory Data Analysis (EDA)

First of all we need to understand our dataset. Before doing that however, we get rid of all rows/instances in which the price does not belong in the range of [1,000 – 200,000]. Our rezoning behind this choice, is that we only find our model practical for values that lie in said range, and all other rows would make it harder for the models to predict accurately prices in our desired target range. After removing said rows, we end up with 288,023 rows instead of the original 371,528.

We start by checking for features that have little to no deviation. For example 'offerType' only has two unique values, and one of them appears four times. Meaning this feature has no impact on the target of our dataset. The same applies for 'seller' and 'nrOfPictures'. So we remove the aforementioned features after also removing the insignificantly few rows that had a different value. We are now only working with 17 features/columns.

Some features, such as, 'vehicleType', 'gearbox', 'model', 'fuelType' and 'notRepairedDamage', have a significant number of missing values into the tens of thousands.

1) Target Variable - Price: We start examining our target variable. This is the distribution of Car Prices:

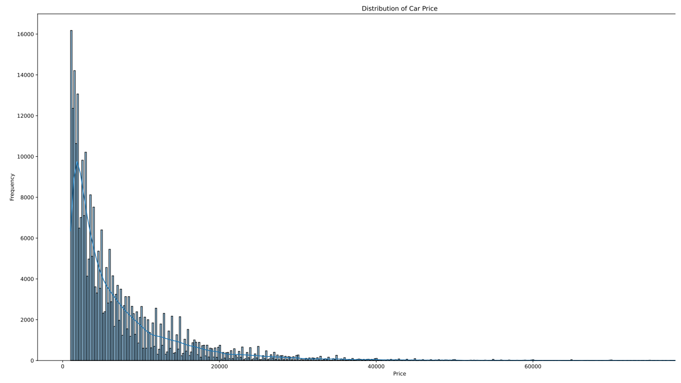

Fig. 1: Distribution of car prices in the dataset(whole graph is not shown)

Our plot showcases that we have rather a lot of extreme outliers. We can also assume that the price distribution is not 'normal' and it is skewed. An assumption that we confirm by running various normality tests such as 'Anderson-Darling', 'Kolmogorov-Smirnov', 'Jarque-Bera' and others. In addition the skeweness of the distribution is found to be '5.13', which confirms that our distribution is extremely right skewed.

2) Categorical Features: We start examining our target variable. This is the distribution of Car Prices:

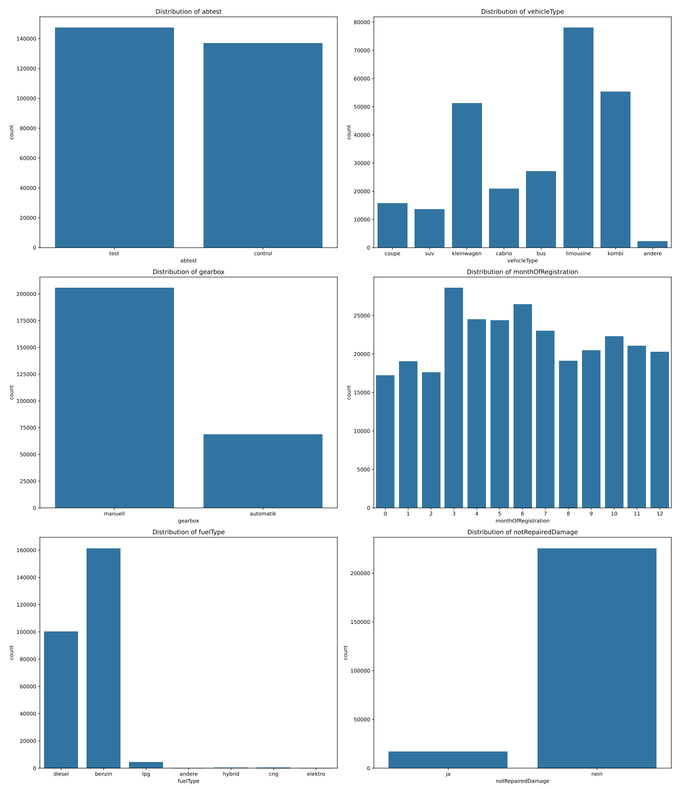

Fig. 2: Distribution of Categorical features in the dataset

1) **ABtest (abtest)**
   - Both the *test* and *control* groups are similarly distributed, with the *test* group appearing slightly more frequently.
   - This feature may be valuable for analyzing the impact of A/B testing on vehicle pricing.
2) **Vehicle Type (vehicleType)**
   - The distribution is dominated by three types: *kleinwagen*, *limousine*, and *kombi*.
   - Other types still appear frequently enough to be considered relevant.
   - The *andere* category is underrepresented and may benefit from oversampling during model training.
3) **Gearbox (gearbox)**
   - Most vehicles have a manual (*manuell*) gearbox, although a significant number are automatic (*automatik*).
   - Gearbox type could be an important predictor of vehicle price.
4) **Month of Registration (monthOfRegistration)**
   - Car registrations are relatively evenly distributed across the months.
   - The presence of a value like 0 suggests potential placeholder values for missing or erroneous data, though the dataset documentation does not clarify this.
5) **Fuel Type (fuelType)**
   - *Benzin* is by far the most common fuel type.
   - *Diesel* also appears frequently enough to influence modeling.
   - Less common fuel types—such as *lpg*, *hybrid*, *cng*, *elektro*, and *andere*—occur too infrequently and might be better grouped into a single `other` category.
6) **Unrepaired Damage (notRepairedDamage)**
   - The majority of entries report *nein* (no damage).
   - A smaller portion reports *ja* (damage present), which could still provide useful information during training.

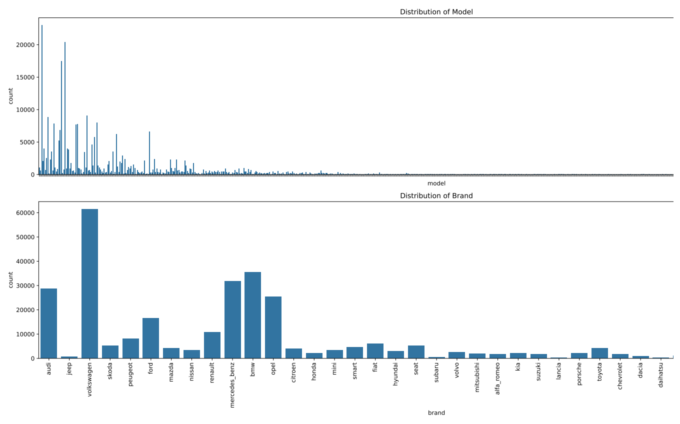

Fig. 3: Distribution of Model and Brand features in the dataset (whole graph is not shown)

Both 'model' and 'brand' are categorical with a very large amount of possible values, dealing with these columns might prove difficult.

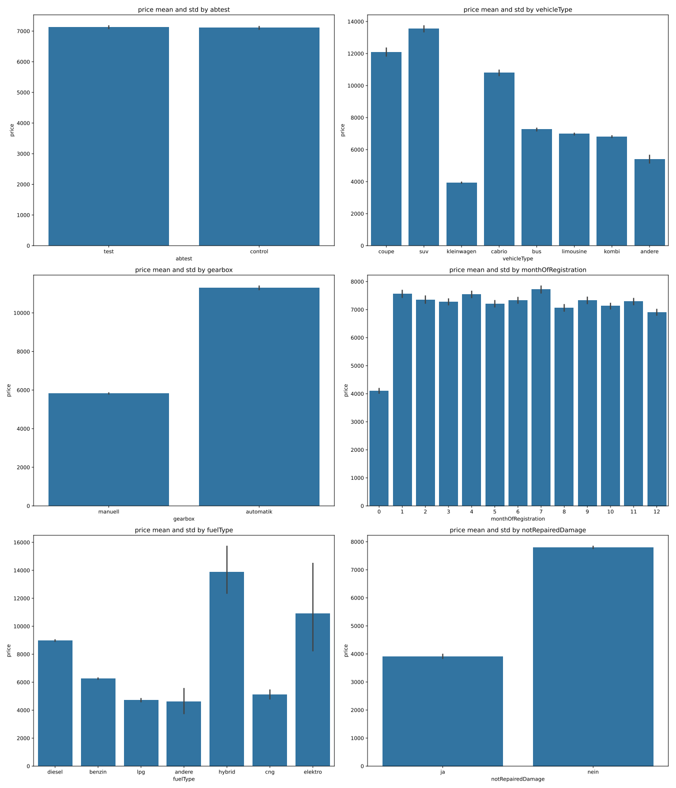

Fig. 4: Target mean and std by categorical features

General Observations:

- Most correlations identified align with our expectations and provide strong signals for price prediction.
- The behavior of the `monthOfRegistration` feature, particularly the `0` value, remains unclear and requires further clarification.

1) **ABtest (abtest)**
   - Both categories of `abtest` exhibit nearly identical average prices.
   - This suggests that the feature has no meaningful correlation with the target variable.
   - Given its apparent lack of influence, it may be beneficial to remove this feature from the model.
2) **Type of Vehicle (vehicleType)**
   - Each vehicle category shows a distinct average price, though with relatively small variance.
   - This indicates a moderate correlation between vehicle type and price, supporting its inclusion as a predictive feature.
3) **Gearbox (gearbox)**
   - The average price differs substantially between gearbox types.
   - This strong association implies high predictive value for this feature.
4) **Month of Registration (monthOfRegistration)**
   - Most months yield similar mean prices.
   - However, entries labeled as month `0` differ significantly, warranting further investigation to understand whether this represents missing data or another issue.
5) **Fuel Type (fuelType)**
   - The fuel type shows a strong correlation with vehicle price, as different categories yield significantly different averages.
   - Categories such as *hybrid* and *elektro* should not be grouped into an `other` category due to their distinct pricing patterns.
   - It may be worth discussing whether to retain or exclude low-frequency categories like *hybrid* and *elektro* depending on modeling objectives.
   - Less frequent types such as *lpg*, *andere*, and *cng* could potentially be grouped into an `other` category.
6) **Unrepaired Damage (notRepairedDamage)**
   - A strong correlation exists between the presence of unrepaired damage and lower prices.
   - This suggests high predictive value for this feature.

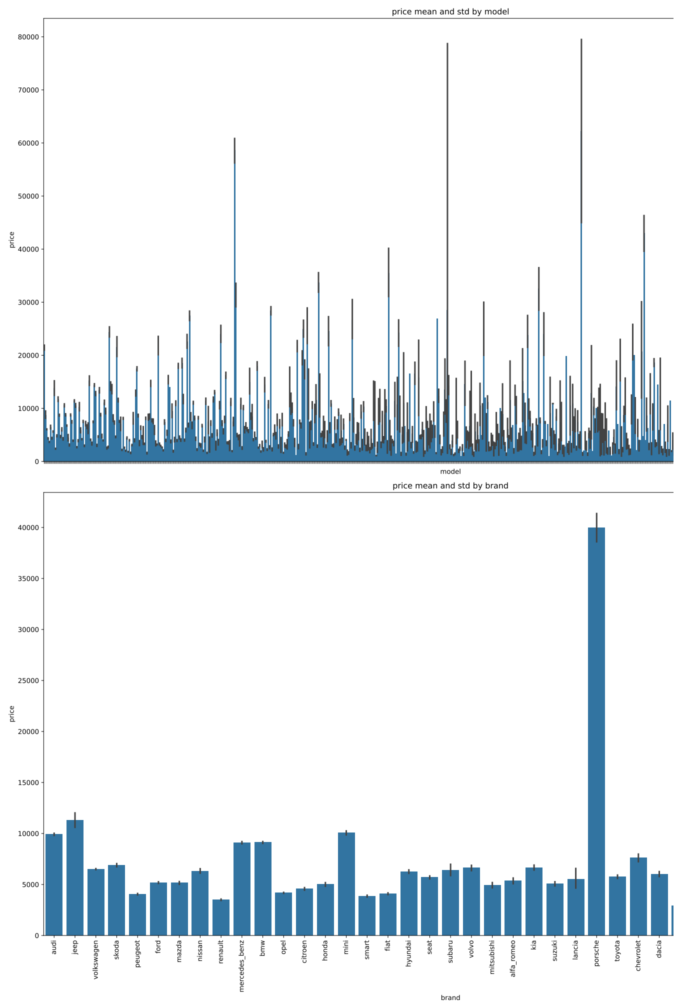

Fig. 5: Target mean and std by model and brand features

1) **Model (model)**

- The `model` feature shows a strong correlation with `price`.
- Many models exhibit high variance in price, indicating that this feature captures a wide range of vehicle characteristics.
- Different models within the same brand often have significantly different price points. For example, both the `911` and `Cayenne` belong to Porsche, yet target different segments with distinct pricing.

2) **Brand (brand)**

- The `brand` feature also shows a strong correlation with `price`.
- While most brands show relatively low variation in price, a few have high deviation, which may merit further investigation.

General Observations:

- Variability within a brand's price distribution may be explained by the diversity of models it offers.
- Even within the same model, vehicles may differ in important ways—such as damage status, age, or other correlated features—that contribute to price outliers.
- Outliers are present in most brands and models, with some categories exhibiting more extreme deviations than others.

name	model	brand
Golf_3_1.6	golf	volkswagen
A5_Sportback_2.7_Tdi	NaN	audi
Jeep_Grand_Cherokee_"Overland"	grand	jeep
GOLF_4_1_4__3TÜRER	golf	volkswagen
Skoda_Fabia_1.4_TDI_PD_Classic	fabia	skoda
BMW_316i___e36_Limousine___Bastlerfahrzeug__Export	3er	bmw
Peugeot_206_CC_110_Platinum	2_reihe	peugeot
VW_Derby_Bj_80__Scheunenfund	andere	volkswagen
Ford_C___Max_Titanium_1_0_L_EcoBoost	c_max	ford
VW_Golf_4_5_tuerig_zu_verkaufen_mit_Anhaengerkupplung	golf	volkswagen

TABLE I: Sample of car name, model, and brand from dataset

A difficulty arises when dealing with the feature 'name'. As shown from the Table I, each value is distinct and does not follow any specific convention. This is important since 'model' can be derived from 'name' and model has 12,148 missing values. We manage to derive the model of the cars from their name by applying fuzzy match of the name into known models of the corresponding brand. This is important since the price of a car within the same brand can vary largely due to the specific model, as shown here

3) Numerical Features: Distributions of numerical features follow.

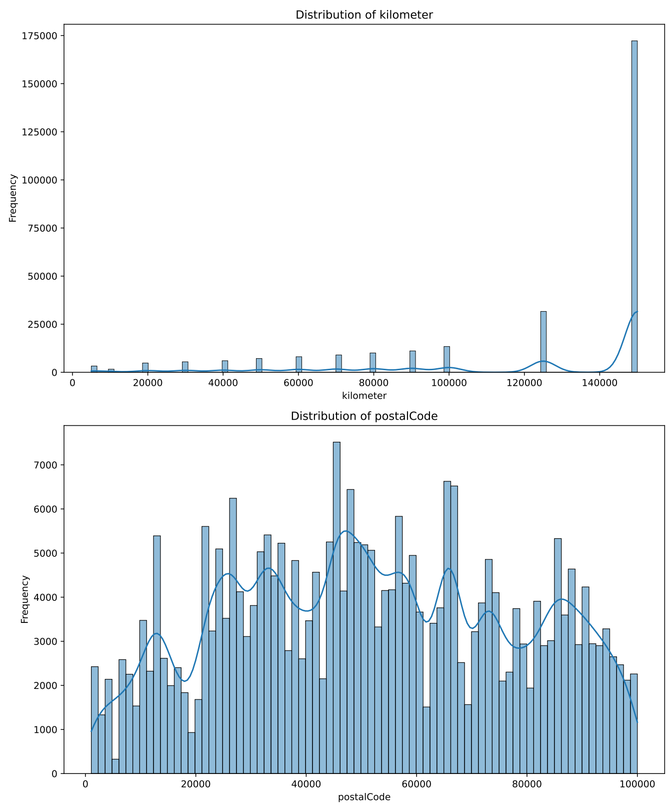

Fig. 6: Distribution of numerical features

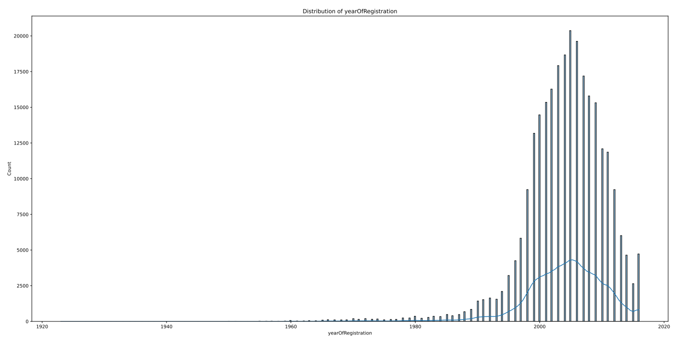

Fig. 7: Distribution of yearOfRegistration

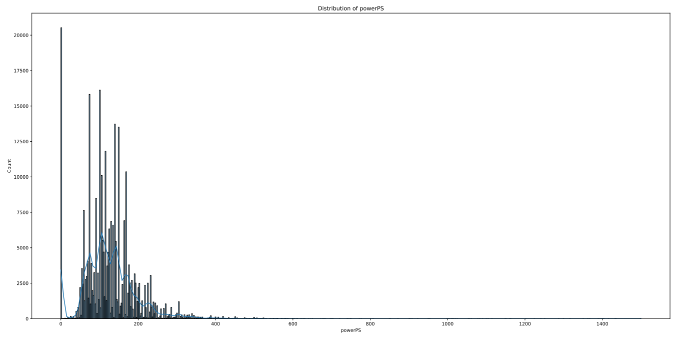

Fig. 8: Distribution of powerPS

Every feature except 'postalCode' is largely-extremely skewed, and no feature follows normal distribution. It needs to be stated that 'yearOfRegistration' and 'powerPS' had a large number of erroneous values. For example dozens of rows had 'yearOfRegistration' values less than 1920 or greater than 2016 (dataset was collected in 2016) and 'powerPS' had values greater than 1500.

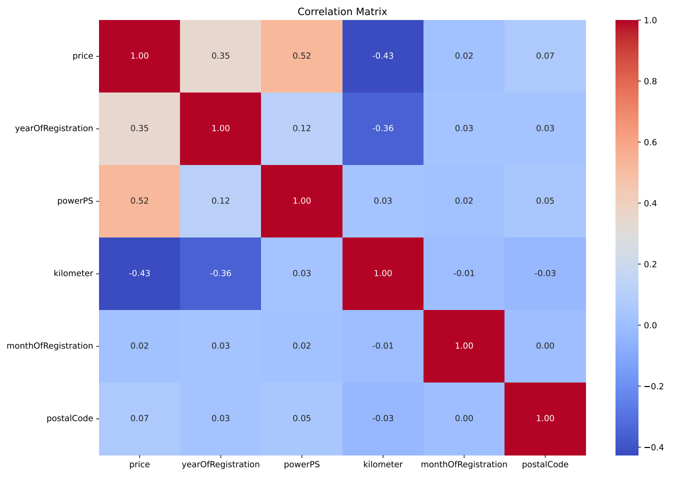

Fig. 9: Heatmap of numerical features

Moderate Correlation:

- `yearOfRegistration` shows a moderate negative correlation with `kilometer`, which aligns with expectations—older vehicles tend to have higher mileage.
- `price` exhibits a moderate positive correlation with both `kilometer` and `powerPS`. These relationships are intuitive: vehicles with more mileage typically depreciate in value, while higher engine power tends to increase price.

No Significant Correlation:

- The remaining numerical features display negligible to no correlation with the target variable or each other.

4) Dataset Split: Before splitting our dataset, first we replace all erroneous values with `NaN`, in order to later treat them as missing values, and fill them in via an imputer. We also combine the values 'lpg' and 'cng' into an `other` category as mentioned before. We also create two new features that we believe will help our model:

1) `adLifespan`: we subtract the columns 'lastSeen' and 'dateCrawled' and we now have how long it took for the car to be sold. It's likely that its higher priced counterparts tend to remain listed longer than cheaper listings which quickly disappear after being seen often enough by members in related markets searching those platforms for new vehicles up for sale at any given time within certain parameters established such as location or age amongst others. This is counted in days.
2) `carAge`: we subtract the columns 'dateCreated' and 'yearOfRegistration' (we also add the 'monthOfRegistration'), we count this in days so it's in the same scale as 'adLifespan'.
3) Parameters or configurations

We now split our dataset 80/20 for train/test respectively. Rows with 'fuelType' values 'hybrid' or 'elektro' are artificially increased (oversampling) in order to better learn these instances. It's important to do this after data splitting, as to avoid data leakage.

We currently have the following panda frames: `X_train`, `X_test` and `y_train, y_test`. And we start creating various versions of `X_train`, in which each version is using different techniques. We use the following imputers:

- SimpleImputer with 'mean' strategy for numerical features
- IterativeImputer for numerical features
- SimpleImputer with 'most_frequent' strategy for categorical features
- SimpleImputer with 'constant' strategy for categorical features

We use the following encoders:

- LabelEncoder
- OneHotEncoding for numerical features

We use the following scalers:

- Normilizer with 'l2' norm for each row
- MinMaxScaler
- RobustScaler

We also use, SFS (Sequential Forward Selection), unskewing (both on X and y). Keep in mind, this is not an exhaustive list of all ideas and concepts that were tried.

III. Experimental Setup

We tested each dataset version we created with a variety of regression algorithms such as 'RandomForest', 'AdaBoost', 'XGBoost', 'CatBoost' and others with their default parameters without altering their hyper parameters to save time (the algorithm 'SVR' was excluded due to impractically long training times). We ran each algorithm both with and without unskewing the target variable.

After finding the two algorithms that performed best along with the corresponding dataset version, we used GridSearchCV pipeline in order to find the best version of those algorithms. Performance was evaluated using the `R2` score.

We performed the same process again but for a classification problem, where now the target variable was not `price` but `price-range` where the following groups exist 'low', 'low average', 'middle average' and 'high'. The only major difference is that we were also able to use the LDA (Linear Discriminant Analysis) dimensionality reduction algorithm. Here we evaluated performance using the `f1_score_weighted`.

IV. Results and Evaluation

For regression, we found that the algorithms that performed best without tweaking them were:

- CatBoost with the original target with score '0.856'.

- RandomForest with the unskewed target with score '0.853'.

Also, the best dataset Versions where:

- V7 with target unskewed with score '0.777'.
- V8 with target unskewed with score '0.776'.

From the pipeline, we found that 'CatBoost' with dataset V8 performed the best with an 'R2' score of '0.853' on the test dataset.

For classification, we found that the algorithms that performed best without tweaking them were:

- RandomForest with score '0.873'.
- XGBoost with score '0.864'.

Also, the best dataset Versions where:

- V10 '0.817'.
- V8 with score '0.811'.

From the pipeline, we found that 'XGBoost' with dataset V8 performed the best with an 'f1_score_weighted' score of '0.875' on the test dataset.

Extra information:

Version V7: This version includes the following preprocessing steps:

- Feature selection using a k-best subset from the original training and test datasets decided by SFS.
- Numerical features were imputed using the `mean` strategy.
- Categorical features were imputed using the `most frequent` strategy.
- Categorical features were encoded using `Ordinal Encoding`, with unknown categories handled by assigning a special value.
- All numerical features were scaled using the `RobustScaler` to mitigate the impact of outliers.

Version V8: Built upon V7 with additional preprocessing:

- Applied `PowerTransformer` (Yeo-Johnson method) to unskew numerical features: `kilometer`, `carAge`, `powerPS`, and `adLifespan`.

Version V10: An alternate preprocessing pipeline starting from the same selected features:

- Numerical features were imputed using the `mean` strategy.
- Categorical features were imputed using the `most frequent` strategy.
- Categorical features were encoded using `One-Hot Encoding`, with unknown categories handled robustly.
- All numerical features were scaled using the `RobustScaler`.
- Applied `Linear Discriminant Analysis (LDA)` on the one-hot encoded categorical features, reducing them to 3 components.
- Final dataset consists of scaled numerical features and the LDA components.
- `PowerTransformer` (Yeo-Johnson) applied to the same set of unskewed features: `kilometer`, `carAge`, `powerPS`, and `adLifespan`.