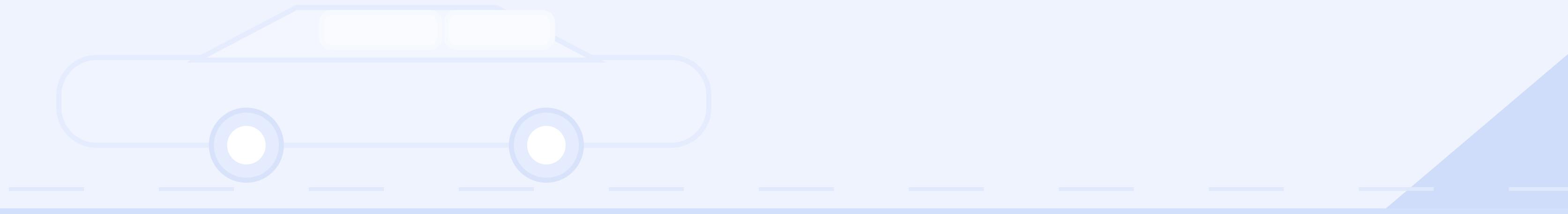




Vehicle Price Predictor

Authors: Salih ATAY, Ramazan EDIZ

Date: December 25, 2025



1) Dataset & Feature Landscape (What We're Predicting)

Goal: predict `selling_price` from numeric + categorical listing features.

Key idea: price is mainly driven by depreciation, performance segment, and market preference.

Core signals:

We drop `name` (high-cardinality; tends to overfit).

- `year`, `km_driven` (depreciation & usage)
- `engine`, `max_power` (segment/performance)
- `fuel`, `transmission`, `seller_type`, `owner` (market preference & trust)

2) Key Features: Why These Matter (and Why We Chose Them)

We select features that are high-signal, widely available, and business-explainable.

In short: fewer but stronger drivers -> more stable predictions.

High-signal drivers:

- `year` -> depreciation
- `km_driven` -> usage
- `engine`, `max_power` -> segment/performance
 - `fuel`, `transmission` -> demand
 - `seller_type`, `owner` -> trust/risk

3) Business Objective & Evaluation Metrics

Goal: consistent price estimates for valuation & negotiation.

Priority: avoid big mistakes on expensive cars.

Metrics:

- R²
- MAE
- RMSE (penalizes big errors)

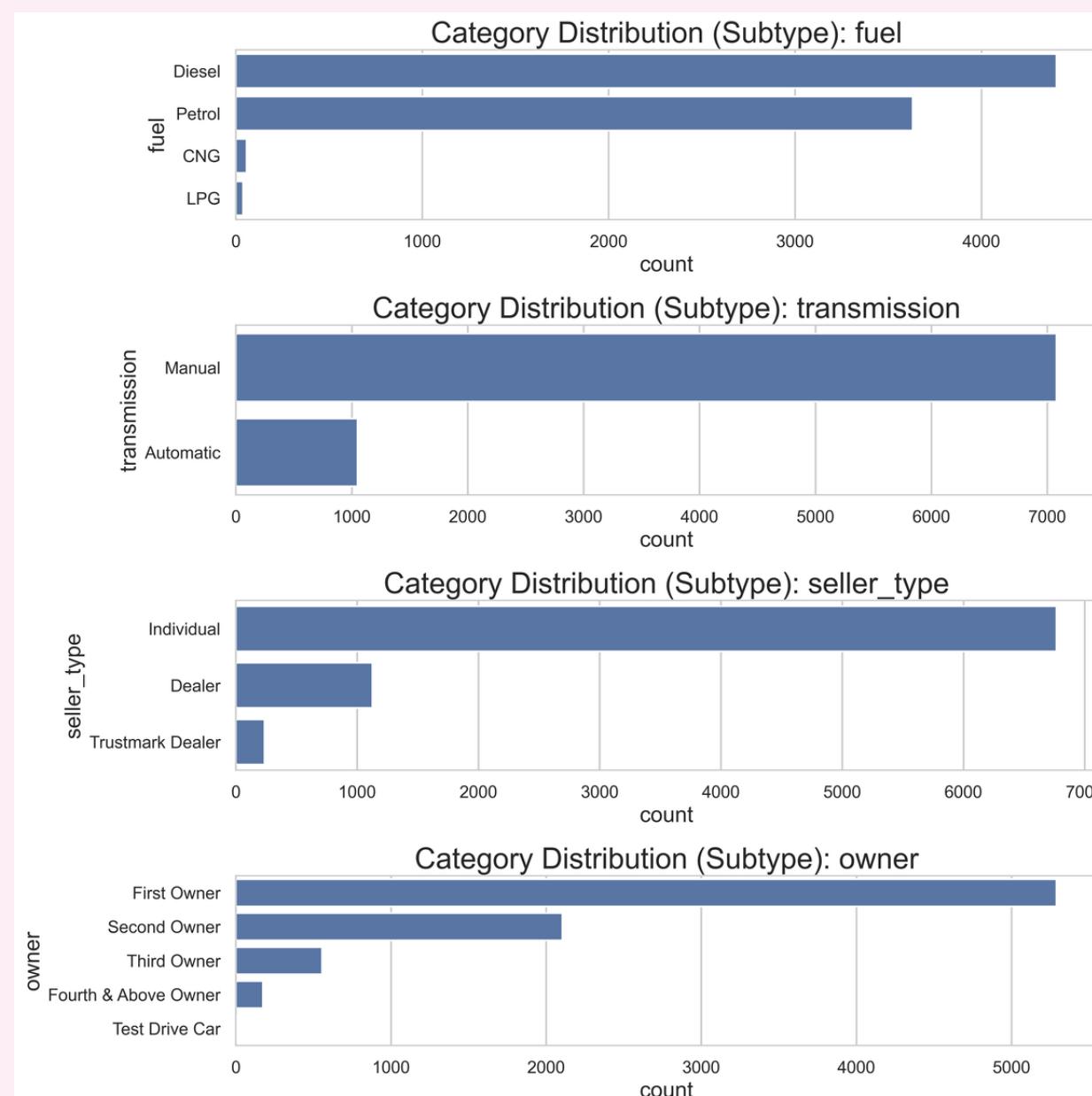
4) Categorical “Subtypes”

5/20

We interpret “subtypes” as category groups that create distinct pricing segments. These variables often encode market preference and perceived risk, which can shift the entire price level even when numeric specs are similar.

Key subtype columns in this dataset:

- `fuel` (petrol/diesel/CNG, etc.)
- `transmission` (manual/automatic)
- `seller_type` (individual/dealer/trust signals)

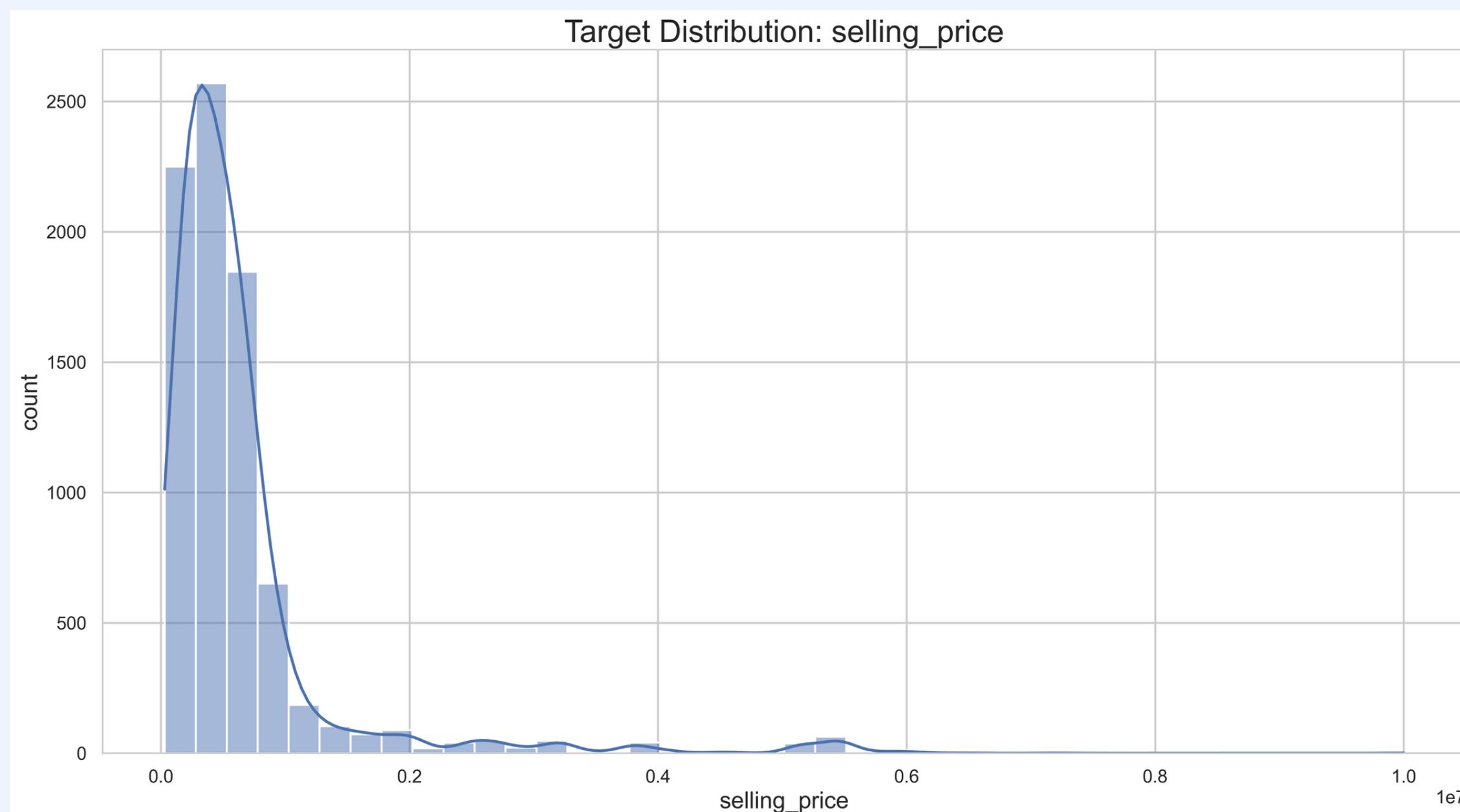


5) Target Distribution (Raw)

The raw target is right-skewed: most cars are in a mid-range, with a relatively small number of very expensive listings. This is typical for real marketplaces.

Why it matters:

- Many models optimize an error that is sensitive to large values
- A few high-priced outliers can dominate RMSE
- Models may underfit the long tail unless the preprocessing and features help



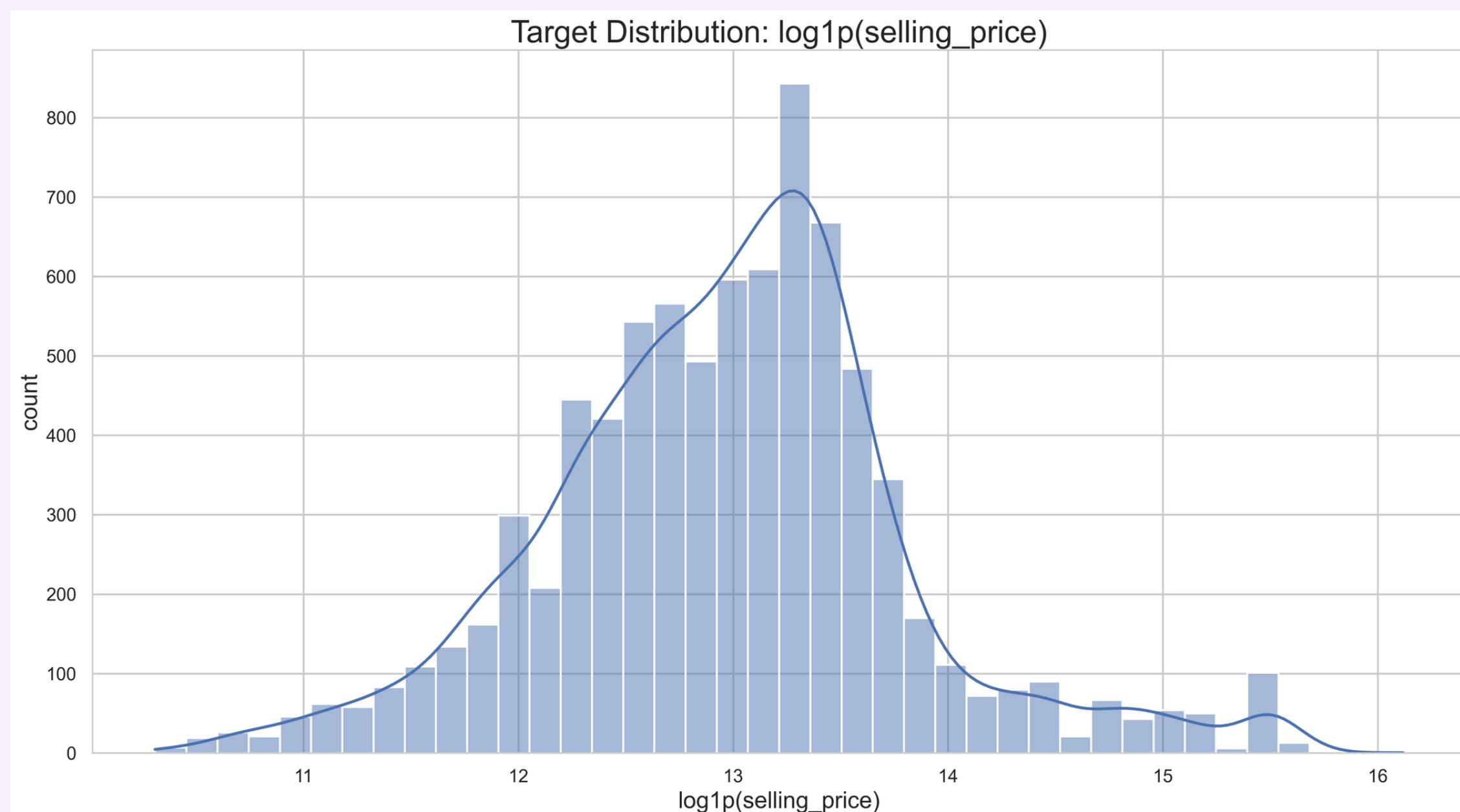
6) Target Distribution (Log Transform)

7/20

To reduce skew and stabilize learning, we also look at a log-transformed target: `log1p(selling_price)` . This compresses the long tail and makes the distribution more “normal-like”.

Practical benefits:

- Improves stability for models that are sensitive to target shape (e.g., SVR)
- Reduces the dominance of extreme prices in the loss
- Often yields more uniform relative error across cheap vs expensive cars

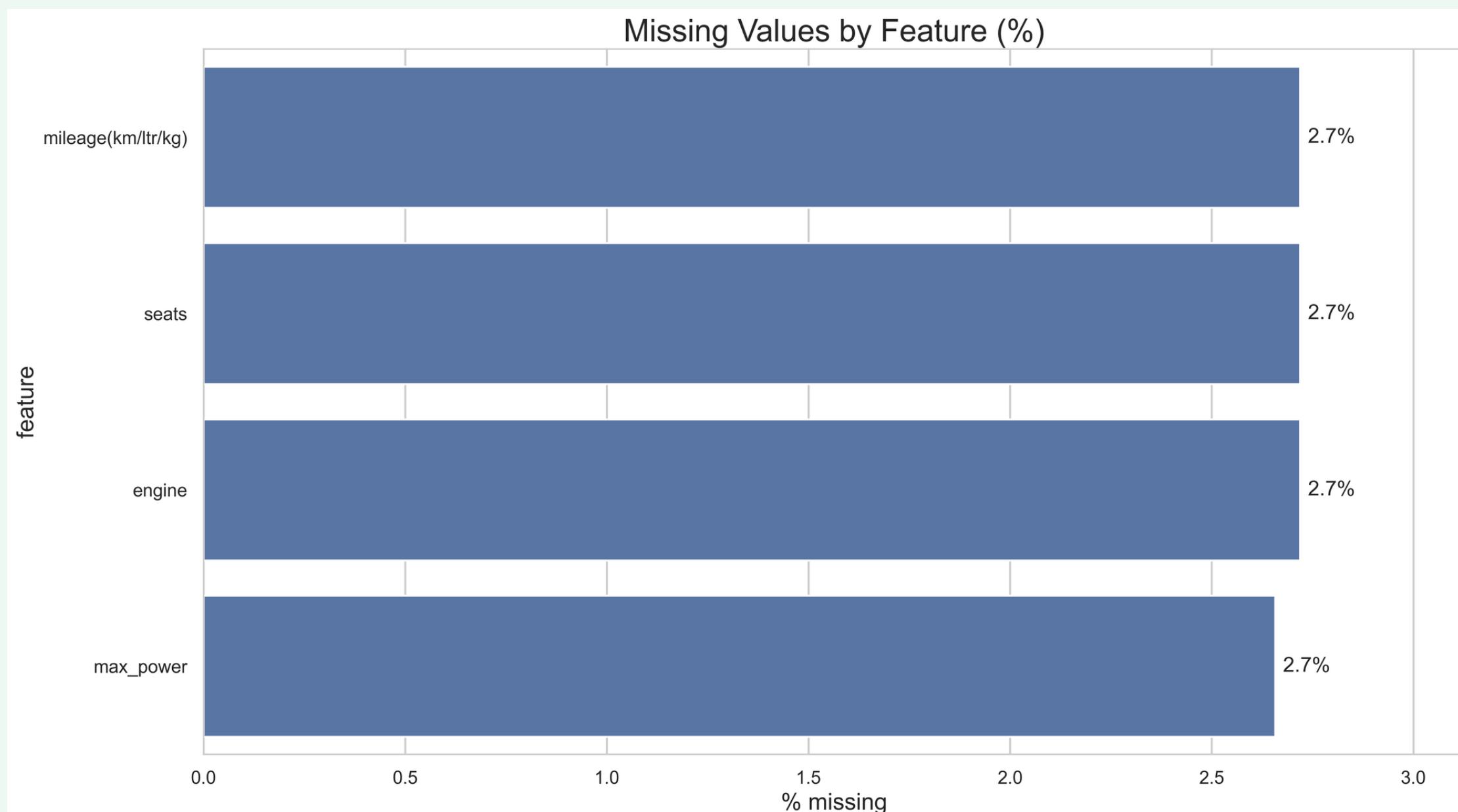


7) Missing Values & Cleaning

Real listing data is messy. Some fields are stored as text that contains numbers, and some numeric values are missing after parsing. If we skip cleaning, the model either fails (non-numeric inputs) or learns unstable patterns.

What we do:

- Parse numeric-in-text columns (e.g., “1248 CC” -> 1248, “74 bhp” -> 74)
- Median-impute missing numeric values (robust to outliers)
- One-hot encode categorical variables to create a consistent numeric matrix



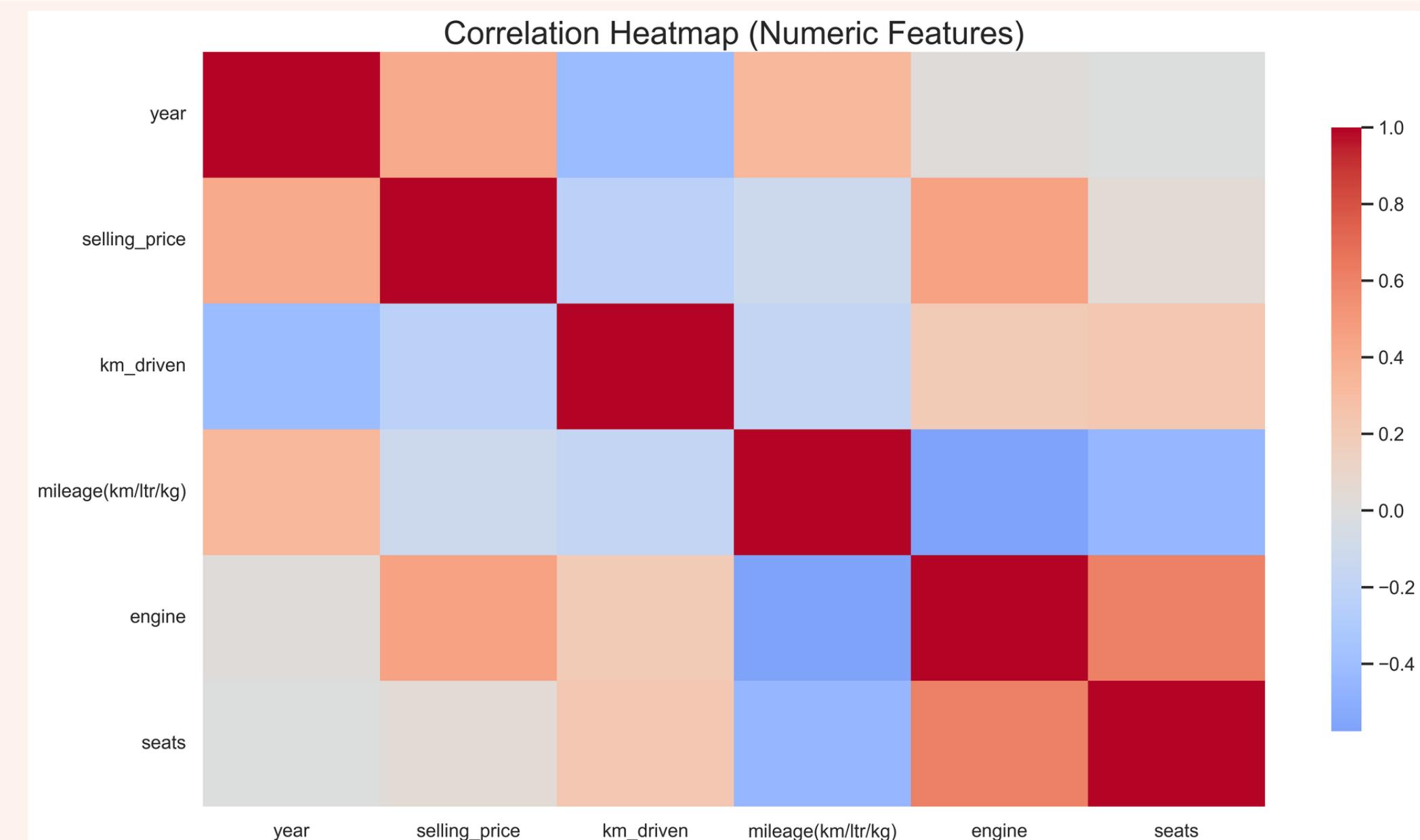
8) Numeric Relationships (Correlation)

9/20

We use correlation analysis as a quick sanity check: do numeric variables show sensible relationships with price and with each other? This step helps catch obvious issues (wrong parsing, swapped units, unexpected missing patterns).

What we look for:

- `year` often correlates positively with price (newer → more expensive)
- `km_driven` often correlates negatively (more usage → cheaper)
- `engine`/`max_power` can correlate with price but also with each other (collinearity)



9) Model Family & Rationale

We compare models with different bias/variance profiles. The point is not just to “pick the winner”, but to understand how much performance we gain when we move from linear to nonlinear and from bagging to boosting.

Models:

- Ridge: fast baseline; good for checking if the feature set contains enough signal
- RandomForest: captures nonlinear interactions and is robust; provides feature importance
- XGBoost: strong boosting approach; often improves accuracy with good regularization
- SVR (log-target): benefits from scaling and a more regular target distribution

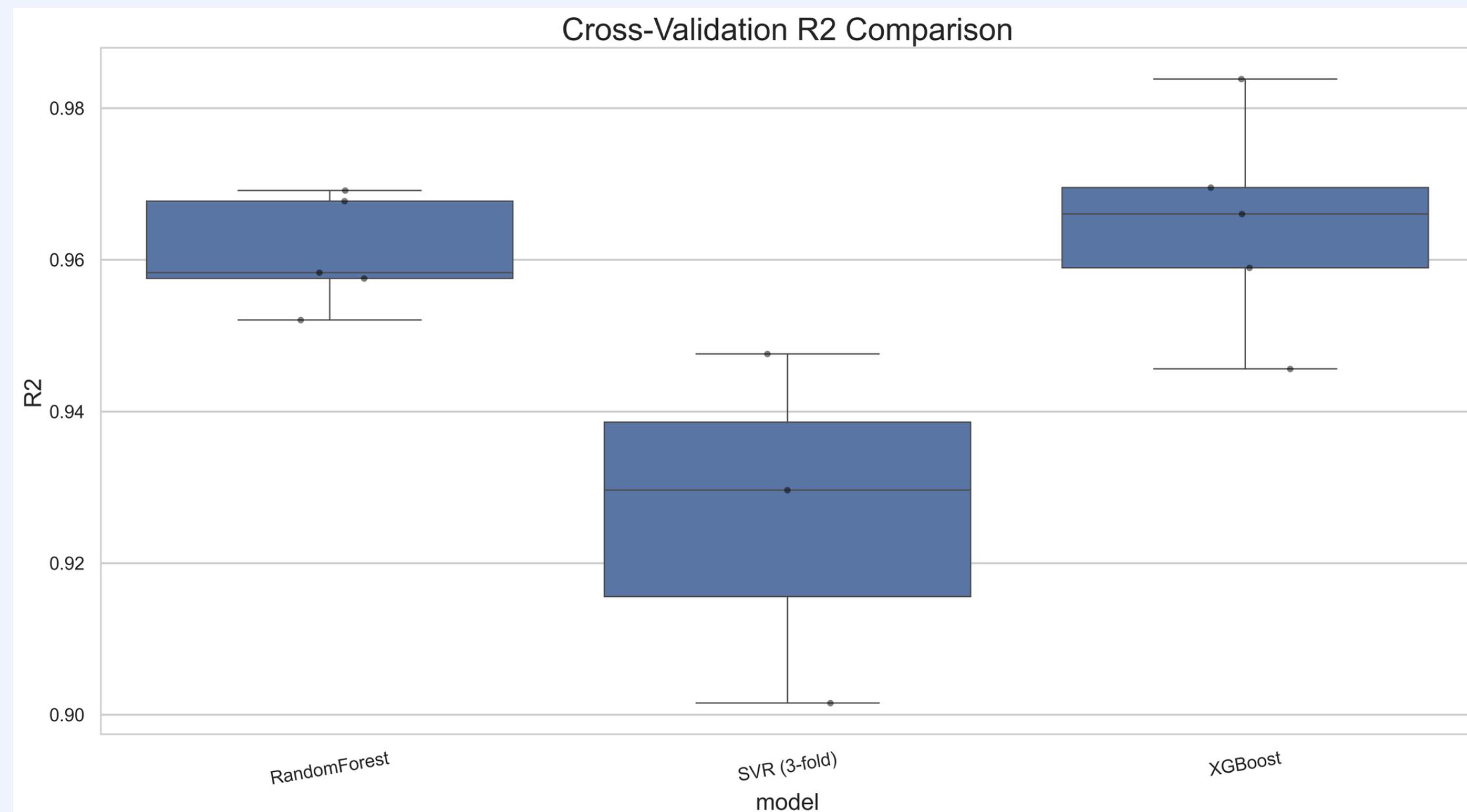
10) Cross-Validation (Why It Matters)

11/20

Cross-validation reduces the risk of over-interpreting a single train/test split. Especially with skewed targets, one split can contain more expensive cars than another, changing metrics substantially.

Why we use CV:

- More stable performance estimate across different folds
- Better comparison across models (same folds)
- Helps detect models that look great on one split but are unstable overall



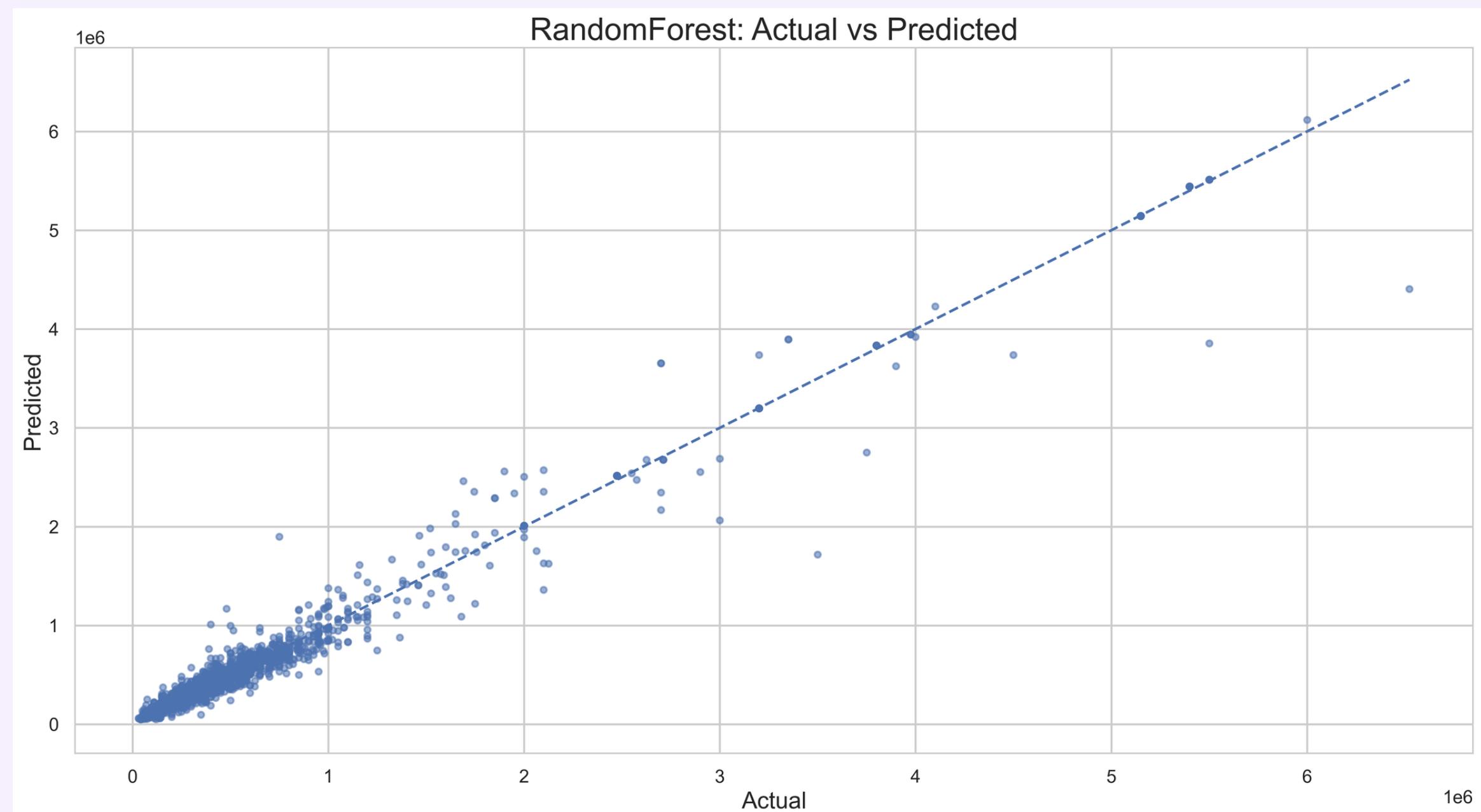
11) RandomForest: Actual vs Predicted

12/20

This plot is a fast visual quality check. If predictions track the diagonal line well across the full range, the model is capturing both low and high prices reasonably.

What to watch:

- Systematic underprediction at the top end (common with long-tail targets)
- Fan-shaped spread (heteroscedasticity), meaning errors grow with price



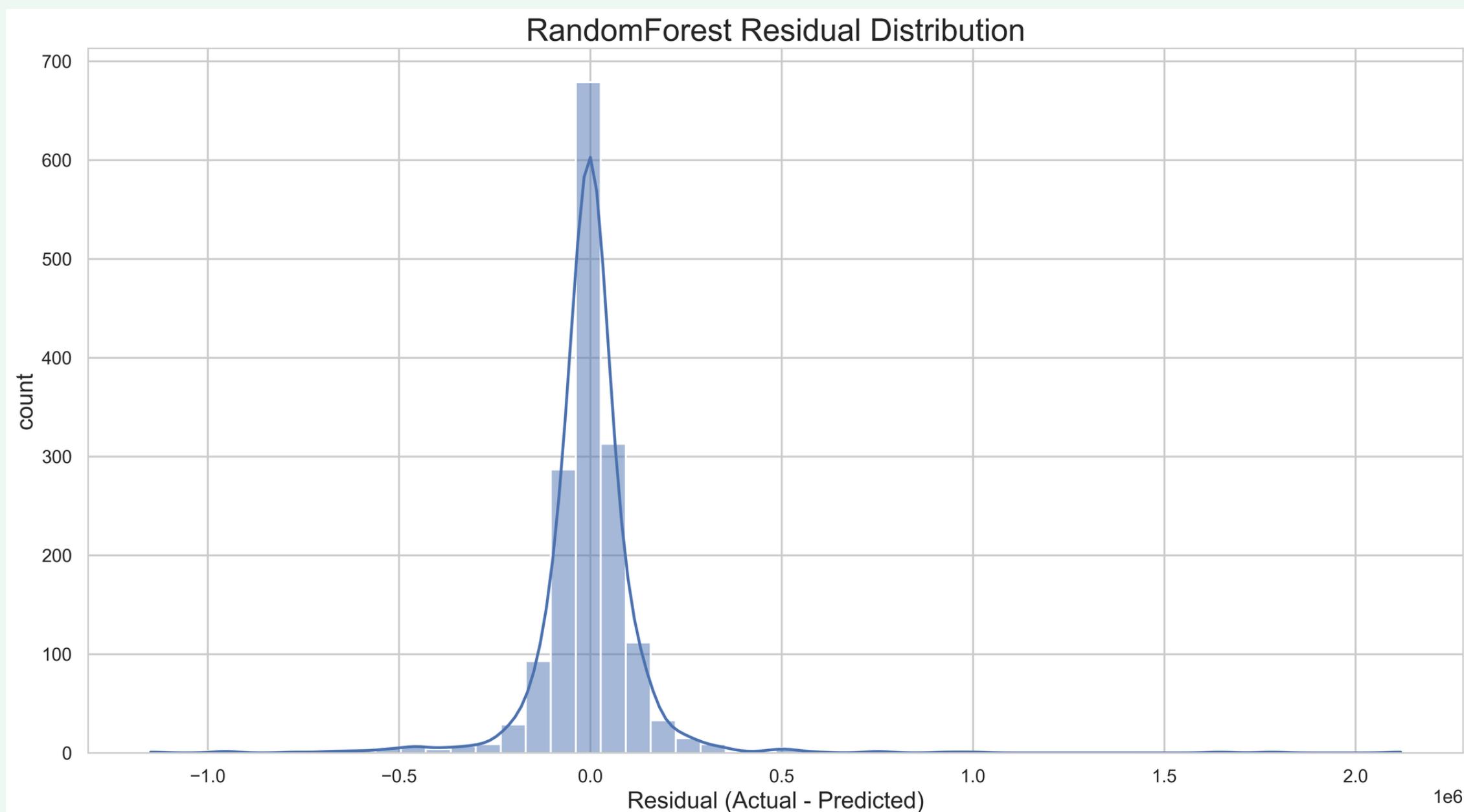
12) RandomForest: Residual Analysis

13/20

Residuals summarize “how wrong” the model is and in which direction.

Interpretation:

- Centered around 0 → no strong bias overall
- Wide tails → occasional big mistakes (often outliers or rare categories)
- Skewed residuals → systematic under/overpricing in some region



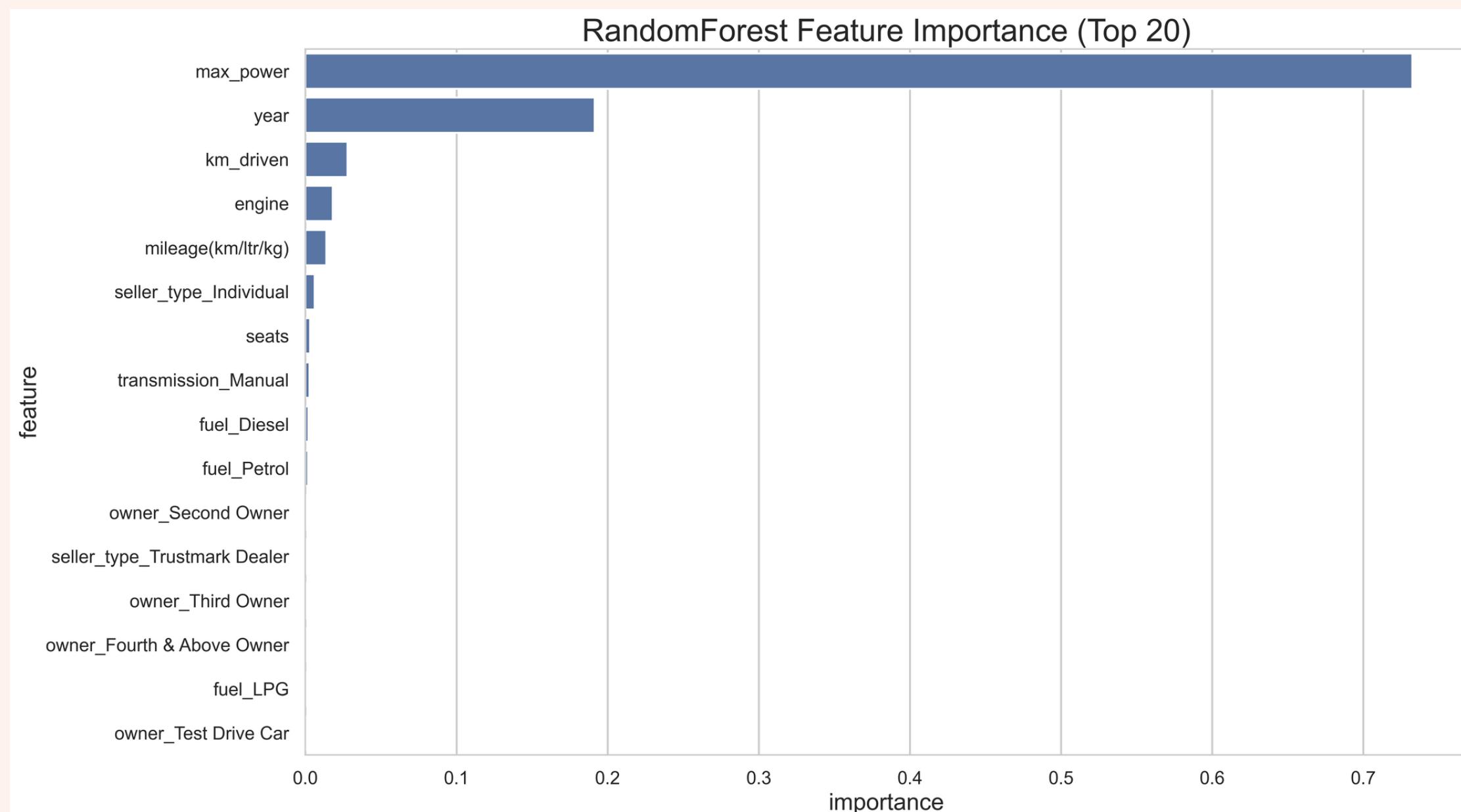
13) RandomForest: Feature Importance (Top 20)

14/20

Feature importance provides a narrative: which signals the model relies on most. For RandomForest, the default importance is impurity-based and can favor features with many split opportunities.

Use it for:

- Debugging (are top drivers sensible?)
- Communication (explain what the model “looks at”)
- Feature iteration (what to improve next)



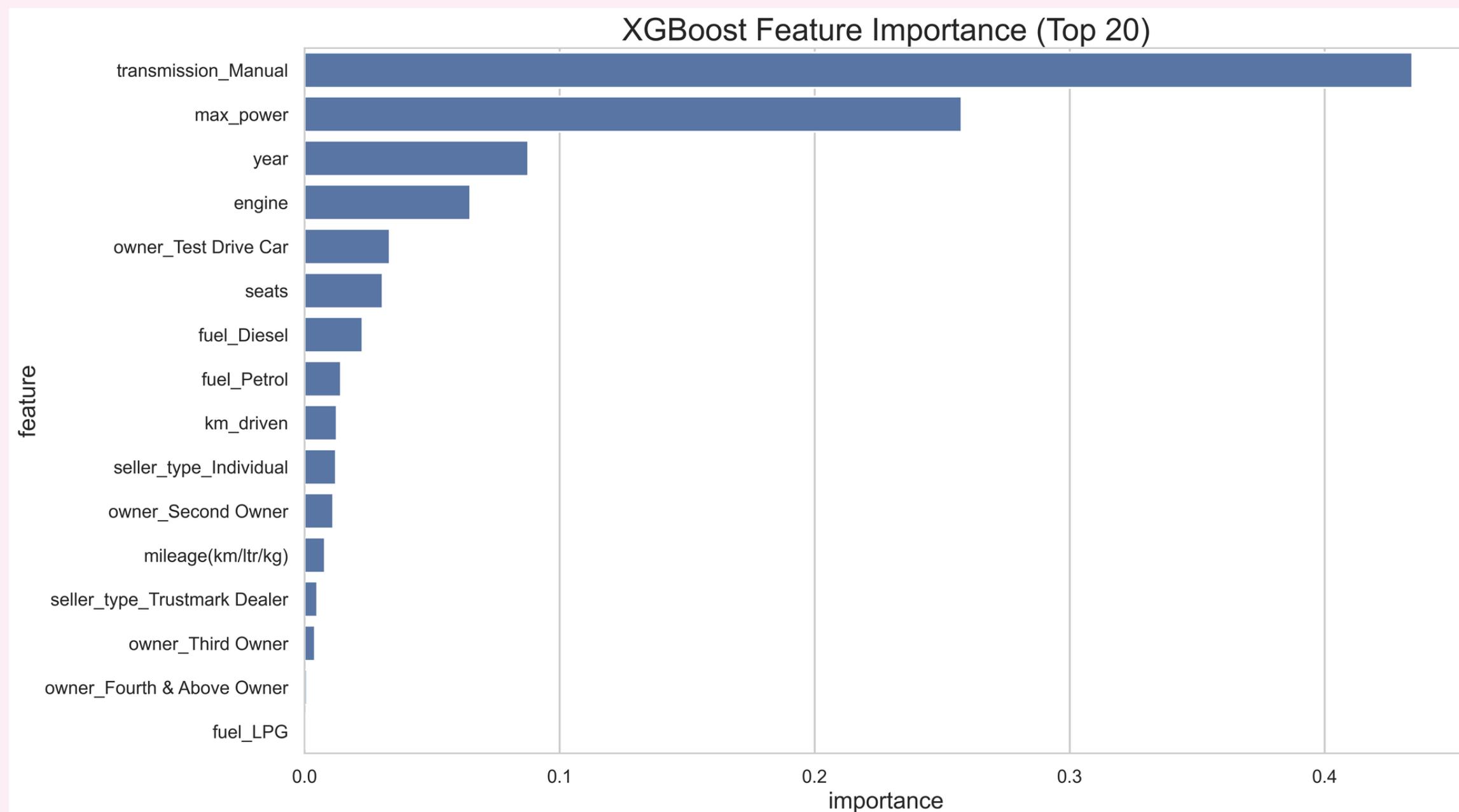
14) XGBoost: Feature Importance (Top 20)

15/20

XGBoost importance is model-specific (often gain/weight/cover). It is useful for within-model understanding, but the absolute scale differs from RandomForest.

Practical guidance:

- Compare ranking within XGBoost to see which features dominate
- Use it alongside error plots; importance alone does not guarantee correctness



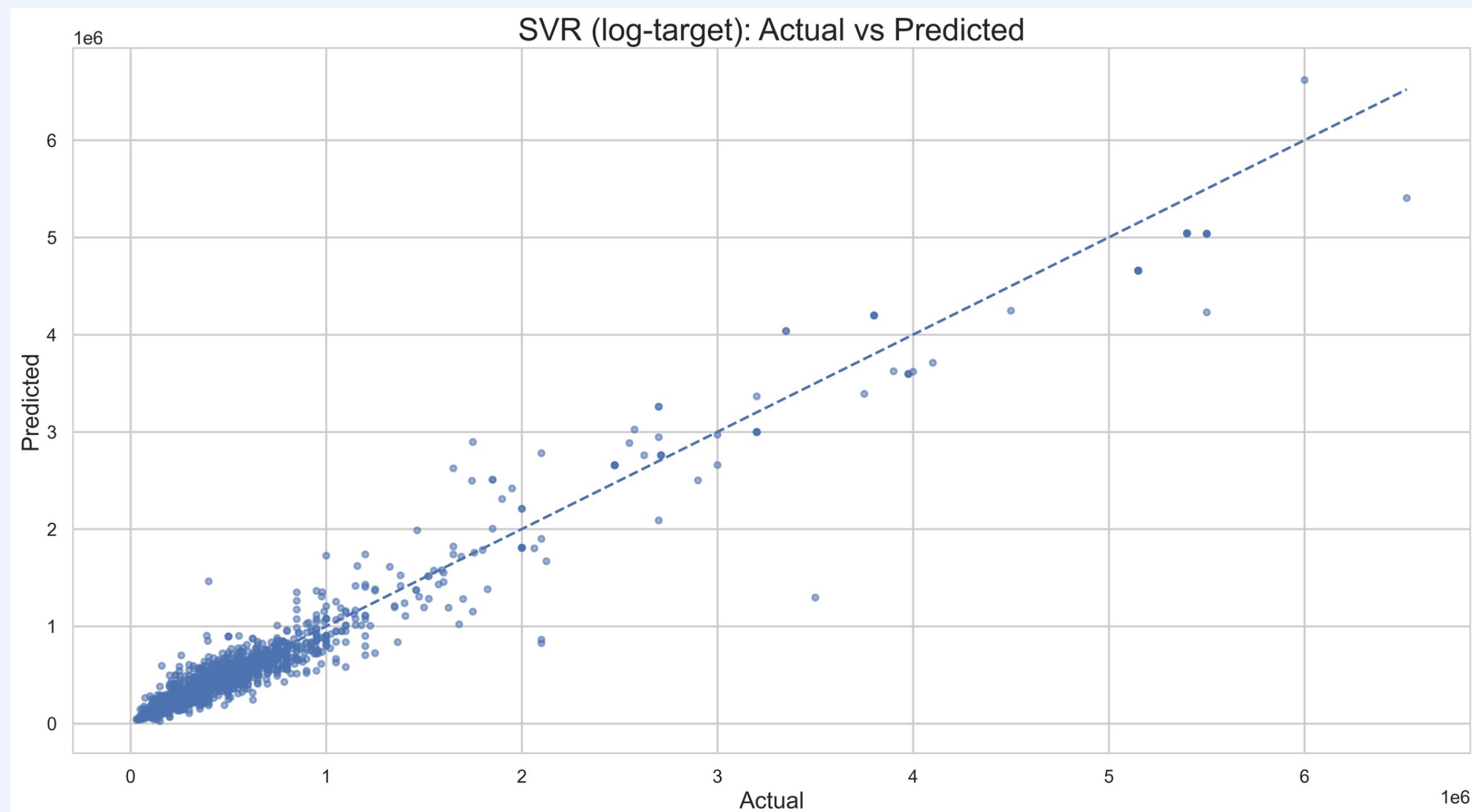
15) SVR (Log-Target): Actual vs Predicted

16/20

SVR can perform well when the feature space is properly scaled and the target distribution is well-behaved. Using a log-target often makes SVR less sensitive to extreme prices.

What we expect:

- Smoother behavior on the long tail
- Potentially higher bias (smoother fit) but fewer catastrophic errors

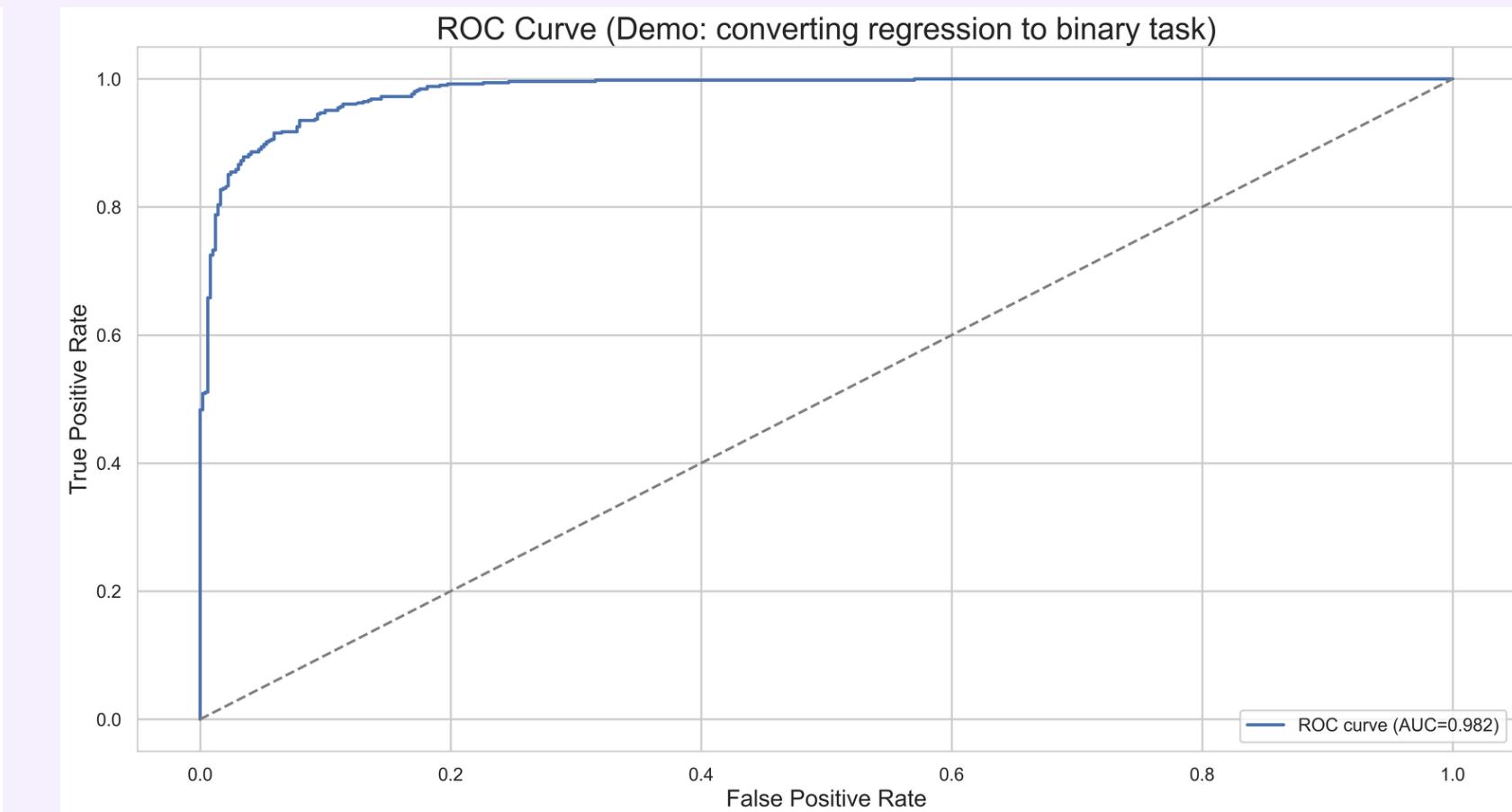
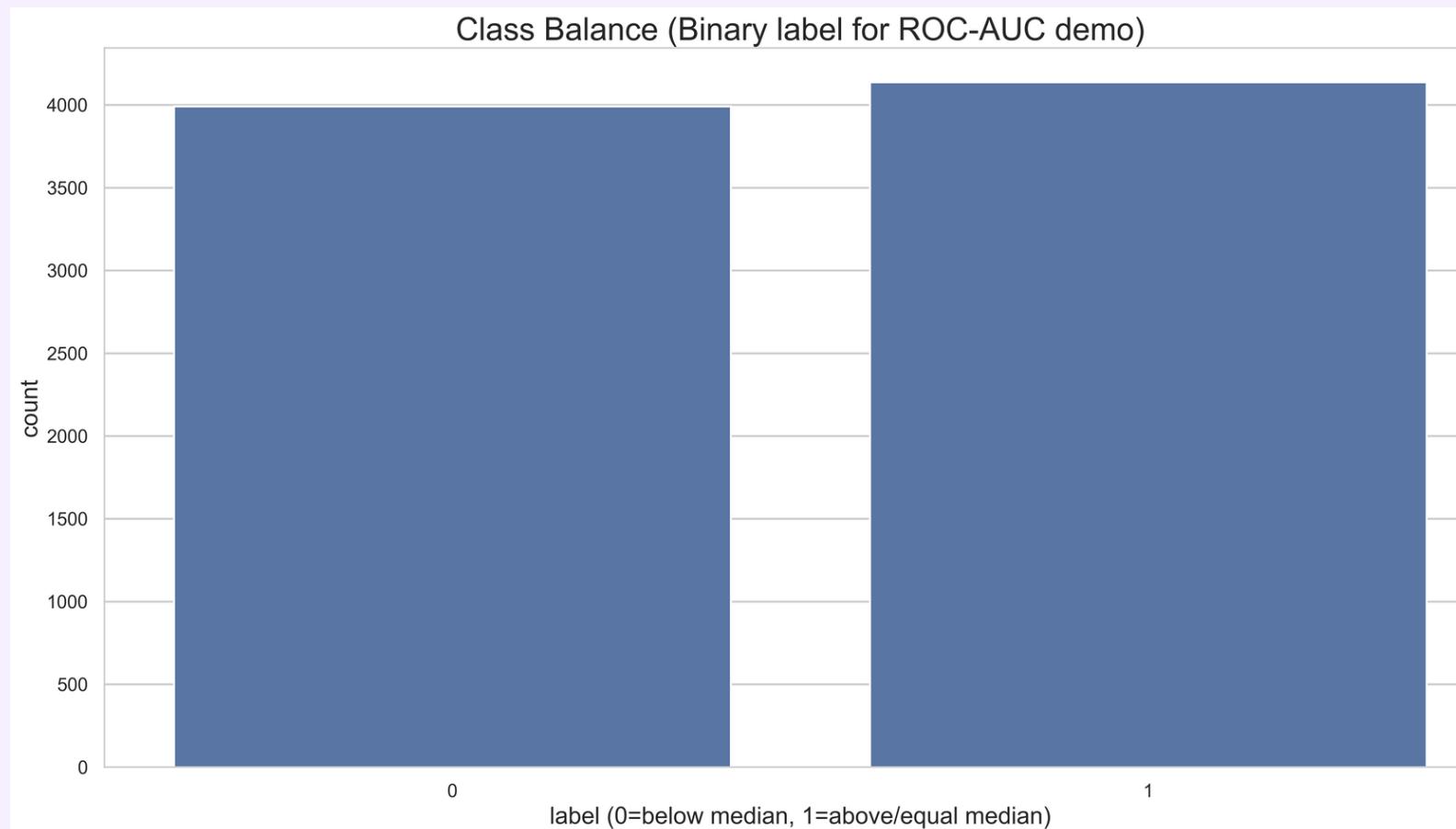


16) “Imbalance” and ROC-AUC (Clarification)

17/20

This project is regression, so “class imbalance” is not directly applicable. However, the target is skewed, which is sometimes loosely described as “imbalance” in reports.

To clarify the concept, we demonstrate ROC-AUC by converting the regression target into a binary label (above/below median). This is purely educational and not the main evaluation for the pricing model.



17) Test Results (Metric Summary)

These results summarize performance on the evaluation split used in the reporting script. All three models are strong; the trade-off is typically between overall fit (R2) and error characteristics (MAE/RMSE).

Metrics:

How to read this:

Note: values are reported from `metrics_summary.csv` and will vary with data/splits.

- RandomForest → R2: 0.966, MAE: 73,736, RMSE: 148,216
 - XGBoost → R2: 0.963, MAE: 69,000, RMSE: 155,985
 - SVR (log-target) → R2: 0.955, MAE: 99,261, RMSE: 171,838
 - XGBoost has the best MAE here (lower typical error)
- RandomForest has slightly higher R2 (overall explained variance)
- SVR is competitive but less accurate in absolute error on this split



18) Key Insights

Key takeaways from the analysis:

Practically, this suggests focusing on robust preprocessing and periodically validating on fresh data as the market changes.

- High-cardinality fields like `name` are tempting but risky; they often act like IDs
- Feature importance is useful for storytelling and debugging, but should be backed by error plots
- The target's right-skew means we must pay attention to tail behavior (expensive cars)

Project Notes & Data Overview

Supervised and Unsupervised Learning:

This project uses supervised learning methods only. The model learns from input data along with the correct output labels (targets).

Unsupervised learning (finding patterns or clusters without labels) is not used in this project.

Why was the 'name' column dropped?

'name' column has very high cardinality and can cause overfitting. It often acts as an ID and may allow the model to memorize rather than generalize.

For a more robust and generalizable model, this column was removed during preprocessing.

Number of Samples and Features:

The dataset contains 8,130 rows (car listings) and 11 features. After dropping the 'name' column, the model is trained with 10 features.

Thank you for listening