TabTransformer-Based Vehicle Mileage Prediction

Author: Petar Prlina

1. Motivation

Mileage is a primary signal of vehicle wear and directly influences resale price, financing terms, and insurance premiums. In many transactional datasets, however, mileage is missing, self-reported, or even deliberately manipulated. Platforms and dealerships therefore need predictive models that can infer mileage from other observable vehicle attributes to flag anomalies and fill gaps. The broader motivation for this project is twofold:

Business impact. Accurate mileage estimation helps marketplaces maintain data quality, prevents fraudulent listings, and supports more precise valuation models. Even reducing error by tens of thousands of kilometers narrows negotiation ranges and improves consumer trust.

Scientific curiosity. The rise of attention-based architectures for tabular data promises richer modeling of categorical interactions than classical linear models or tree ensembles. We investigate whether the TabTransformer (Huang et al., 2020) can improve upon traditional baselines for a real-world automotive dataset that exhibits heavy-tailed targets, mixed data types, and substantial heterogeneity across brands and vehicle segments.

In short, the project examines whether a deep learning model designed for tabular data can produce actionable mileage predictions that meet a stringent requirement: test-set RMSE below 40k km.

2. Research questions

The work revolves around a central question and several supporting inquiries, all derived from the project brief.

Q1 (Primary). Can a TabTransformer-based regression pipeline predict vehicle mileage (km_driven) from heterogeneous tabular features with RMSE < 30k km.

Q2 (Feature engineering). Which transformations—particularly around age-normalized ratios, log transforms, and categorical abstractions—yield the most informative feature space for the transformer encoder?

Q3 (Robustness). How should we treat extreme mileage outliers so that the model remains useful both for the dense middle of the distribution and the sparse high-mileage tail?

Q4 (Optimization). Which hyperparameters (learning rate, weight decay, dropout, transformer depth) balance convergence speed, regularization, and generalization in this setting?

Dataset summary. The project uses the public "Vehicle Dataset from CarDekho" (Car details v3.xls) comprising 8,128 listings with 13 raw columns. Five features are categorical (name, fuel, seller_type, transmission, owner). The rest are numeric but often encoded as strings (mileage "23.4 kmpl", torque "194 Nm at 3750 rpm"). The target, km_driven, spans from 1 km to 2,360,000 km with median ~65,000 km, producing a long-tailed response variable. Missing values affect mileage, engine, max_power, torque, and seats. The analysis focuses on automotive market data, which includes both personal vehicles and commercial taxis, increasing target variability.

3. Related work

Historically, used-car analytics relied on linear regression and hedonic pricing models (Court, 1939) that treat price as a sum of individual attribute contributions. Modern datasets shifted focus to tree-based ensembles (Random Forest, Gradient Boosting, XGBoost) because of their ability to model non-linear interactions and handle missing data. Kaggle competitions on car price prediction often crown gradient boosting as the winning approach due to its robustness on small-to-medium tabular datasets.

Deep learning approaches for tabular data gained momentum with the introduction of the Transformer (Vaswani et al., 2017) and its derivatives. et al. (2020) proposed TabTransformer, which applies multi-head self-attention to categorical embeddings, contextualizing them before combining with numerical features. Gorishniy et al. (2021) evaluated multiple neural architectures on a range of tabular benchmarks, concluding that with careful preprocessing and regularization, attention-based models can rival or outperform boosting methods, especially when categorical cardinality is high or interactions complex.

In the automotive domain, most research examines price prediction or demand forecasting, often augmenting tabular features with textual data. Mileage prediction specifically appears as a sub-task within fraud detection or history verification systems but receives less independent study. This project contributes by applying an attention-based tabular architecture to mileage estimation, and evaluating performance under strict target clipping rules.

4. Methodology

It comprises of six stages: data ingestion, cleaning, outlier handling, feature engineering, model architecture, and training/evaluation protocol.

4.1 Data ingestion and initial checks

- Load the Excel-formatted dataset using pandas, infer column types, and normalize column names to snake_case.
- Perform sanity checks: number of unique brands, presence of duplicate rows (43 duplicates removed), distribution of key features, and ratio of categorical to numeric values.
- Create an 80/20 train-test split stratified on seller_type (three classes) to maintain proportionate representation in the holdout set. Random seed 42 ensures reproducibility.

4.2 Data cleaning

- **Parsing text-encoded numerics.** Mileage, engine, max_power, and torque columns contain units embedded in strings. Regular expressions extract numeric components, convert to floats, and harmonize units (e.g., torque expressed in kgm converted to Newton-meters via multiplication by 9.80665). Torque also yields an auxiliary rpm feature when present.
- **Handling missing values.** For training data, numeric columns use median imputation after splitting; categorical columns introduce `<UNK>` tokens for unseen categories.
- **Duplicate removal.** A small set of completely duplicated listings is dropped to prevent biasing the model toward frequently repeated samples.

4.3 Outlier detection and target treatment

- **Isolation Forest.** Trained solely on numeric predictors from the training partition, using contamination=0.055 (5.5%). Observations flagged as outliers are removed before further processing.
- **LOESS soft clipping.** For each numeric feature, a LOESS regression is fit against vehicle age (frac=0.2). Values deviating beyond three median absolute deviations (MAD) from the local LOESS fit are clipped to

that boundary. This technique suppresses extreme values while preserving continuity and attribute relationships.

- **Target clipping.** Mileage values above the 99.5th percentile (~270,000 km) are clipped *only within the training set*. This decision keeps optimization stable yet ensures the evaluation uses the original, unmodified targets. Approximately 35 training examples are affected, reducing the influence of extreme commercial vehicles during gradient descent.

4.4 Feature engineering

Feature engineering aims to encode domain knowledge and mitigate skew.

- **Temporal features.** Compute `age = 2025 - year`, `age_sq`, and `log_age`.
- **Usage ratios.** Derive `km_per_year`, `price_per_year`, `price_per_km`, `torque_per_liter`, `power_per_liter`, `price_per_power`, and `price_per_engine`.
- **Log transforms.** Apply `log1p` to selling_price, engine, max_power, mileage, and torque to reduce skewness.
- **Categorical abstraction.** Extract brand (first token of name) and a simplified model identifier (first two tokens). Categoricals include name_simple, brand, fuel, seller_type, transmission, owner. Unseen categories map to a shared `<UNK>` token during inference (no additional rare-category bucket).
- **Missingness flags.** Binary indicators mark whether a numeric value was originally missing before imputation.
- **Scaling.** Numeric features use RobustScaler (median center, IQR scaling) fit on the training data to resist the impact of long tails.

4.5 Model architecture

The TabTransformer configuration is implemented in PyTorch:

- **Categorical embeddings.** Each categorical feature embeds into a 32-dimensional vector (including a learned embedding for `<UNK>`). A `[CLS]` token summarizing categorical context is prepended.
- **Transformer encoder.** Four encoder layers with eight attention heads (d_model=128) process the categorical tokens. Layer normalization and dropout (0.1) mitigate overfitting. No positional encoding is applied because the categorical tokens are unordered.
- **Fusion head.** The `[CLS]` output from the transformer concatenates with the normalized numeric features, followed by an MLP (input `d_model + numeric_dim` → 256 → 128 → 1) producing the standardized log-transformed mileage prediction.

4.6 Training, validation

- **Target transformation.** Training targets undergo `log1p` transformation followed by standardization (subtract mean, divide by standard deviation). The inverse transform (exp, multiply, add) reconstructs predictions in kilometers.
- **Cross-validation.** A 3-fold stratified cross-validation (on seller_type) evaluates the hyperparameter grid defined in the configuration. In the current setup the grid contains a single combination—learning_rate=1e-4, weight_decay=1e-4, dropout=0.1, but before it had tested combinations of learning_rate=[5e-5, 1e-4, 2e-4], weight_decay=[1e-5, 5e-5, 1e-4], dropout=[0.05, 0.1, 0.15]
- **Optimization.** AdamW optimizer.
- **Final model.** After selecting the best hyperparameters, the model is retrained on the full training set (post CV) for up to 24 epochs with early stopping. Evaluation on the holdout test set uses the original, unclipped mileage values.

5.Discussion

5.1 Hyperparameter search outcomes

Cross-validation results identify the configuration `{learning_rate=1e-4, weight_decay=1e-4, dropout=0.1}` as the most optimal. Alternative settings (explored in earlier experiments) either slowed convergence (lower learning rates) or caused instability (higher learning rates); adjusting dropout away from 0.1 consistently degraded validation performance.

5.2 Final evaluation

- **Test RMSE:** 27,388 km

5.3 Practical implications

The trained model can be integrated into a data quality pipeline that compares reported mileage with predicted mileage. Listings deviating by more than 30k km for example could trigger manual review.

5.4 Limitations and future directions

- **Tail coverage**: Training-time clipping curtails the model's ability to generalize beyond 270k km. Future work should explore adaptive clipping thresholds or loss functions less sensitive to outliers (Huber, quantile pinball).
- **Model ensembles:** Combining TabTransformer with gradient boosting might reduce variance and capture complementary patterns.
- **Multimodal features**: Incorporating text descriptions or image embeddings could improve predictions, especially for luxury segments where appearance and maintenance condition matter.
- **Temporal drift**: The dataset spans multiple years, but the pipeline treats year solely via derived features. Modeling temporal drift explicitly (e.g., with time-based cross-validation) might enhance robustness as market preferences change.

6. References

Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2021). Revisiting Deep Learning Models for Tabular Data. *Advances in Neural Information Processing Systems*, 34, 18932–18943.

Huang, X., Khetan, A., Cvitkovic, M., & Karnin, Z. (2020). TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *Advances in Neural Information Processing Systems*, 33.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, 30.