University of Sydney

COMP5310 Principles of Data Science

COMP5310 Project Stage 2

Group Number: 197

Group Members:

SID: 540373930 **Unikey:** yiwu0182
SID: 550077552 **Unikey:** tche0080
SID: 540917332 **Unikey:** awan0950

15 May 2025

Group Component 1

1. Topic and Research Question

The central research question of this project is:

"Can we accurately predict the market price of used vehicles based on vehicle specifications and usage conditions?"

Pricing used vehicles is challenging due to the interplay of factors such as brand, mileage, fuel type, engine size, and age. Buyers often find it difficult to judge price fairness, while sellers face uncertainty in setting competitive yet profitable prices—contributing to inefficiencies across the second-hand vehicle market.

In this context, predictive modelling based on historical vehicle data becomes a valuable tool. By training models on cleaned and structured data, we aim to generate accurate price estimates that reflect real market dynamics. This modelling serves multiple stakeholders:

Consumers can assess whether a vehicle is overpriced or a good deal, increasing their bargaining power and reducing the risk of overpaying.

Dealers and resellers can improve pricing strategies, manage inventory turnover more effectively, and identify undervalued assets for acquisition.

Online vehicle platforms can offer automated pricing recommendations to enhance listing quality and platform credibility.

Insurance and loan providers may use price prediction tools to improve risk assessment, residual value estimation, and credit evaluation processes.

2. Dataset

This project uses a cleaned dataset of 16,026 used vehicle listings with 18 features, each representing technical specifications, usage conditions, and sale-related information for individual cars. The target variable is Price, which is continuous and represents the final listed price in AUD.

The dataset contains a mix of numerical and categorical features. Numerical variables include Mileage, Horsepower, Displacement, Weight, Age, Previous_Owners, Gears, and Cons_Comb. Categorical features include Make_Model, Body_Type, Fuel, Gearing_Type, Drive_Chain, and Vat.. The most frequently occurring brands are Audi, BMW, and Mercedes, with Benzine as the dominant fuel type and Manual as the dominant gearing type.

Challenges and Biases in the Data

Distributional Skew and Outliers

Variables like Price, Mileage, and Horsepower are right-skewed with extreme values (e.g., >300,000 km mileage, >1000 horsepower), which may lead to overfitting on rare high-end cases.

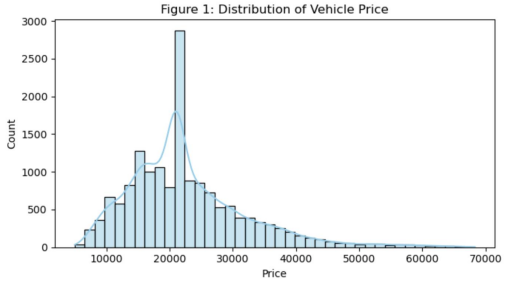


Figure 1: Distribution of Vehicle Price

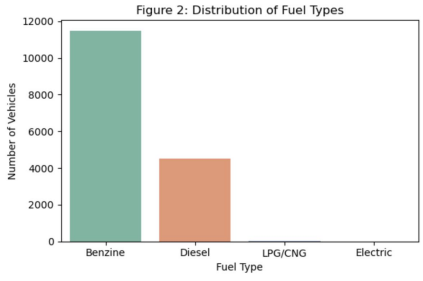


Figure 2: Distribution of Fuel Types

Category Imbalance

Over 70% of cars use Benzine, and Manual transmissions dominate, potentially biasing the model against underrepresented fuel types and gearings.

High Cardinality in Categorical Variables

Make_Model has 9 families, but some like Audi A3 have thousands of entries, while others are rare. This uneven distribution can lead to overfitting if not encoded carefully.

Potential Multicollinearity

Strong correlations exist between features like Horsepower, Displacement, and Weight, which may destabilize linear models and inflate feature importance.

Impact on Modelling Process

These issues can reduce model generalization, cause bias, or impair interpretability. For instance, outliers can skew loss functions, and unbalanced classes can degrade performance on minority categories.

To mitigate these effects, we applied tailored preprocessing steps: label and frequency encoding for categorical variables, outlier capping, and stratified train/test splitting. These steps ensure model fairness and robustness across diverse vehicle types.

3. Set up

3.1 Modelling agreements

In this study, we decided to use the clean dataset from assignment 1 directly, because we have already handled some issues, such as missing values and some useless features etc. However, our individual components may contain other preprocessing steps (such as label encoding) to suit our models. And the preprocessing will not influence the fairness of the comparison.

We will predict the price of vehicles (numerical) according to features related to second-hand cars (such as age, type, gears, etc.). We decided to choose RMSE, MAE, and R-squared as metrics to measure the performance of models. These metrics were chosen because they go beyond standard accuracy and allow us to evaluate the models from multiple views:

RMSE (Root Mean Squared Error): The unit of RMSE is consistent with the original target variable and is easier to interpret. It pays more attention to large errors because squaring it and then rooting it makes the large prediction deviation larger. However, it is sensitive to outliers, and it cannot directly show 'how much variance is explained'.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

MAE (Mean Absolute Error): MAE is the average of the absolute differences between the predicted values and the actual values. It is not sensitive to outliers. MAE has the same weight for each error. And it has a more direct explanation: The unit is the same as the target variable. So, it is suitable for checking the overall average deviation.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

R-squared: R-squared measures how much variance the model explains, which is a normalised measurement indicator. It has strong interpretability: 0 means the model is not better than the simple average, and 1 indicates perfect prediction. R-squared is used to measure the overall interpretive ability of the model, and it is especially suitable for comparing the performance of multiple models on the same dataset.

$$R^2 = 1 - \frac{\sum(\hat{y}_i - y_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

3.2 Data division

We split the dataset into the **training set (70%)** and the **test set (30%)**. This is a compromise choice between training and assessment, because the model cannot learn the pattern when the training set is too small, and the evaluation results will be unstable when the test set is too small. Then we add a random state (= 42 here) to guarantee the same results in each running time. Here, the validation set is not included because it will be split by the cross-validation method we chose.

Moreover, we decided to use **Repeated K-Fold Cross-Validation** (n_splits = 5, n_repeats = 3). The Repeated K-Fold Cross-Validation is based on the common K-Fold Cross-Validation and obtains more stable and reliable evaluation results by repeated splitting and training. 5-fold is one of the most commonly used folding numbers, and it has a better balance between assessment error and time cost than a small fold (such as 2-fold) or a large fold (such as 10-fold). Repeating three times is a balance point between effect and efficiency, because the more repetitions, the more times the model is trained, and the computing cost rises sharply.

Stratified sampling was not applied, as the target variable Price is continuous; this technique is more appropriate for classification tasks. The price distribution (Figure 1) is moderately right-skewed, which is typical in used vehicle markets. Since the skewness is not severe and no extreme outliers were observed, we did not perform **transformations or outlier removal**. The distribution is reasonable for this context, and the evaluation metrics used are robust to such skewness.

COMP5310 Stage 2 Report: XGBoost

Unikey: yiwu0182

1. Predictive model

1.1. Model Description

The model used in this section is Extreme Gradient Boosting (XGBoost), a scalable and efficient ensemble learning algorithm based on gradient-boosted decision trees. It builds multiple weak learners in sequence, each correcting the residuals of the previous one, and combines them for final prediction.

XGBoost does not assume linearity or normality, making it suitable for our dataset, which includes non-linear interactions, mixed feature types, and skewed distributions such as in Mileage and Price. It handles multicollinearity and outliers well and does not require feature scaling.

This model is particularly appropriate for predicting used vehicle prices, as it can capture complex relationships—such as how Horsepower and Age jointly affect price—which simpler models may miss. It also supports parallel computation, enabling efficient hyperparameter tuning.

The main limitation is reduced interpretability compared to linear models, and tuning can be computationally intensive. However, given the structure and complexity of the data, XGBoost offers a strong balance of flexibility and performance, aligning well with the goal of accurate price prediction.

1.2. Model Algorithm

XGBoost is a stage-wise boosting algorithm that builds an ensemble of decision trees to iteratively reduce prediction errors. At each boosting round, the model minimizes the following regularized objective function (m0_58475958, 2024):

$$\text{obj}^{(t)} \approx \sum_{i=1}^{m} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \gamma T + \frac{1}{2} \lambda \|w\|^2$$

Figure 1. Function in XGBoost

```
Input:
    - Training data: (X, y)
    - Number of boosting rounds: T
    - Learning rate: η
    - Base prediction: F₀(x) = mean(y)

For t = 1 to T do:
    1. Compute gradients (gᵢ) and Hessians (hᵢ) for each data point
    2. Fit a regression tree fₜ(x) using gᵢ and hᵢ
    3. Add fₜ(x) to the model with weight η:
            Fₜ(x) = Fₜ₋₁(x) + η * fₜ(x)

Output:
    Final prediction function F_T(x)
```

Figure 2. Pseudocode of the XGBoost Algorithm

Where: 1) g_i is the first-order derivative (gradient) of the loss with respect to the prediction for sample i. 2) h_i is the second-order derivative (Hessian) of the loss for sample i. 3) $f_t(x_i)$ is the prediction of the new tree f_t for sample x_i. 4) T is the number of leaf nodes in the tree f_t. 5) w is the vector of leaf weights (predicted values at each leaf). 6) $\|w\|^2$ represents the L2 norm (squared) of the leaf weight vector. 7) γ and λ are regularization parameters controlling model complexity.

In this project, we implemented the model using XGBRegressor from the xgboost library. We performed hyperparameter tuning using GridSearchCV with RepeatedKFold (5 folds × 3 repeats), optimizing parameters such as max_depth, learning_rate, n_estimators, subsample, and colsample_bytree.

While explicit regularization terms were not tuned, the algorithm's internal structure still penalizes overly complex trees, offering strong generalization. These characteristics are well-suited to modeling the non-linear and hierarchical relationships in vehicle pricing. XGBoost also supports recent adaptations such as early stopping, monotonic constraints, and automatic handling of missing values, although these were not applied in this implementation.

1.3. Model Development

The model development process began with basic preprocessing. While no advanced techniques such as feature scaling or dimensionality reduction (e.g., PCA) were used—since XGBoost is a tree-based model and does not require normalized or orthogonalized inputs—categorical features such as Make_Model and Fuel were transformed using label encoding to allow compatibility with the algorithm's structure.

Hyperparameter tuning was carried out using GridSearchCV, with a search space covering max_depth, learning_rate,

n_estimators, subsample, and colsample_bytree. Each parameter was selected based on its theoretical impact on model performance:

max_depth{3, 5} controls tree complexity and risk of overfitting.

learning_rate{0.05, 0.1} adjusts the contribution of each tree, balancing learning speed and stability.

n_estimators{100, 200} sets the number of boosting rounds.

subsample{0.8, 1.0} and **colsample_bytree**{0.8, 1.0} introduce randomness to reduce variance and overfitting, especially important in our context with high-cardinality features.

The Python functions used include:

LabelEncoder: Encodes categorical variables (e.g., Make_Model, Fuel) into numerical form for compatibility with tree-based models like XGBoost.

XGBRegressor: Initializes the XGBoost regression model that implements scalable gradient boosting decision trees.

RepeatedKFold: Implements repeated K-fold cross-validation to reduce evaluation variance and improve parameter selection robustness.

GridSearchCV: Performs an exhaustive search over specified hyperparameter values using cross-validation to identify the optimal combination.

fit: Trains the model or hyperparameter searcher on the training dataset.

predict: Generates predictions from the trained model for a given input dataset (typically the test set).

mean_squared_error, **r2_score**, and **mean_absolute_error**: Evaluate final model performance.

sns.scatterplot: Creates scatterplots for visualizing relationships such as actual vs predicted prices or residuals vs predictions.

plot_importance: Visualizes feature importance based on the trained XGBoost model to identify which features had the most predictive power.

plt.figure, plt.plot, plt.axhline: Used to configure figure size, draw reference lines, and manage plot layout in model evaluation visualizations.

In the context of used vehicle pricing, parameters such as subsample and colsample_bytree helped prevent overfitting caused by dominant categories like Make_Model, while max_depth controlled how much interaction between features (e.g., Horsepower × Mileage) could be captured.

2. Model Evaluation and Optimization

2.1. Model Evaluation

The model's performance was assessed using MAE, RMSE, and R^2 on the test set. The tuned model outperformed the initial version across all metrics:

Metric	Initial Model	Final Model
MAE	4903.50	4770.90
RMSE	6471.21	6223.66
R^2	0.5283	0.5537

These results demonstrate consistent improvement after hyperparameter tuning. Residuals appear randomly distributed with no clear pattern, indicating that the model generalizes well and does not suffer from strong systematic bias.

Figure 3 shows that predictions align closely with actual prices, particularly in the mid-price range ($10,000 – $40,000). Residuals in Figure 4 are centered around zero with no clear trend, indicating low bias. However, error variance increases at higher price ranges, likely due to fewer samples and greater diversity.

Although XGBoost is less interpretable than linear models, it effectively captures nonlinear relationships. Some categorical variables, like Make_Model, are highly imbalanced, and minor noise in fields such as Mileage may affect performance.

Future improvements could include engineered features (e.g., age-adjusted mileage), SHAP-based interpretation, or

model ensembling to enhance both accuracy and explainability.

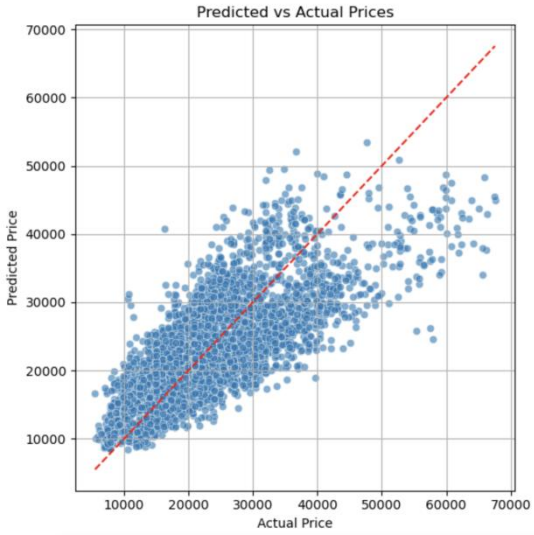


Figure 3: Predicted vs Actual Prices

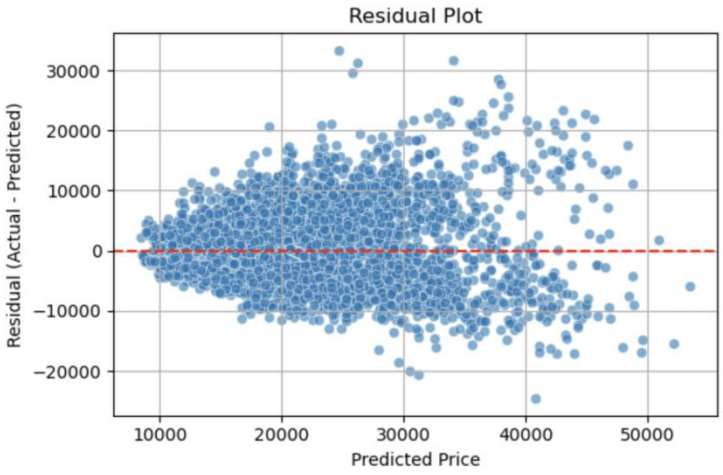


Figure 4: Residual Plot of XGBoost Predictions

2.2. Model Optimisation

The final model was selected via GridSearchCV with repeated 5-fold cross-validation (3 repeats, 15 total folds). The optimal hyperparameters were:

Maximum depth: 5

Learning rate: 0.05

Number of estimators: 200

Subsample ratio: 0.8

Column sample by tree: 0.8

This configuration achieved the best cross-validated performance, with test RMSE reduced from 6471.21 to 6223.66 and R^2 improved from 0.5283 to 0.5637.

Each parameter played a key role: max_depth controlled model complexity, learning_rate adjusted learning speed, and subsample and colsample_bytree introduced randomness to enhance generalization. These changes reduced overfitting and improved predictive accuracy.

Feature importance analysis (Figure 5) shows that Mileage, Cons_Comb, and Horsepower are the most influential variables — consistent with domain knowledge that vehicle condition and efficiency drive pricing. This supports the research goal of accurately predicting used car prices using vehicle characteristics.

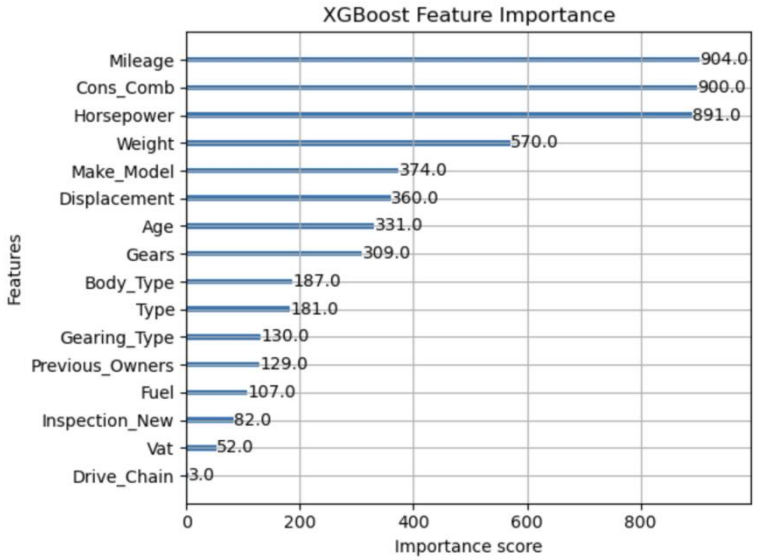


Figure 5: Feature importance

TabNet Deep Learning Model for Price Prediction in Used Vehicles

Unikey:tche0080

1. Predictive model – TabNet Deep Learning Model

1.1 Model Description: The predictive model used in this section is TabNet. It is a deep learning architecture designed for tabular data, introduced by Arik and Pfister (2021) from Google Cloud AI. The use of Tabnet is well-suited to our cleaned dataset in this assignment. TabNet combines the strengths of gradient boosting decision trees and neural networks by using sequential attention mechanisms to choose which features to reason from at each decision step (Arik & Pfister, 2021). Tabnet is not like traditional models like XGBoost or Random Forest. This model performs feature selection and interprets feature importance through "attention masks"(Arik & Pfister, 2021)**.**

Assumptions: In TabNet architecture, one of the key assumptions is that different samples of the dataset will benefit from focusing on different subsets of features. This assumption is performed by using instance-wise attention masks, and it is generated by the "Sparsemax function". Another important assumption is that feature selection and transformation can be done simultaneously and sequentially over multiple decision steps(Arik & Pfister, 2021).

Suitability for our Research Problem: TabNet is well-suited for our research problem, which aims to accurately predict the market price of used vehicles based on their technical and categorical features. Tabnet's ability is to handle both numerical and categorical features without requiring complicated preprocessing. It can automatically learn complex relationships within the data. Moreover, TabNet provides built-in interpretability through feature importance masks, which allows us to understand which attributes are the most important to predictions. Compared to XGBoost or LightGBM, TabNet performs end-to-end learning. This means that it can train all parts of the model in one process. This feature makes it particularly effective for capturing important patterns in real-life tabular datasets, such as the dataset we used in this assignment (Arik & Pfister, 2021).

Strengths: One of the advantages of using TabNet in this dataset is its interpretability. TabNet allows visualization of feature importance at each decision step and provides insights into how decisions are made for each sample. It also reduces the need for extensive preprocessing, such as binary encoding. TabNet's attention mask selects which features to focus on at each step, helping the model adapt to complex patterns and ignore less important features, which can reduce overfitting. This attention architecture also improves efficiency and accuracy, especially in high-dimensional data. According to Arik & Pfister (2021), TabNet demonstrates comparably higher accuracy than XGBoost, CatBoost, or MLP models.

Limitations: Tabnet requires careful tuning of hyperparameters. For example, it includes the number of decisions, the steps, attention deep sizes or other related coefficients. This limitation can make the model training time longer and computationally expensive (Arik & Pfister, 2021). Another disadvantage is that Tabnet model's training time is longer than other traditional models, especially on the smaller datasets. Traditional models might perform better in smaller datasets with shorter training time. Lastly, since Tabnet is released in 2021, the open-source and documentation of Tabnet might not be mature enough compared to other already well-established models. This limitation reflects the challenges in debugging or implementing (Arik & Pfister, 2021).

1.2 Model Algorithm: TabNet is composed of feature transformers, decision steps, and attention masks(Arik & Pfister, 2021). At each decision step, a subset of features is selected and passed through transformation blocks to generate predictions. The key innovation is its use of attention masks to guide instance-wise feature selection in an interpretable manner (Arik & Pfister, 2021).

Step-by-Step Execution:

1. **Input Features:** All the numerical and categorical features are processed directly without extensive preprocessing (one-hot encoding e.g.)
2. **Share Feature Transformer Block:** This block learns key feature representations and is reused across all decision steps. It ensures a consistent starting point for each. It also consists of fully connected (linear) layers with non-linear activations, which is typically in two layers. It also integrates Ghost Batch Normalization to simulate smaller batch sizes and promote generalization. In addition, Gated Linear Units (GLUs) are used instead of ReLU to allow the model to control how much information is passed through. GLU enhances the model's learning flexibility.
3. **Attentive Transformer and Sparsemax:** In each decision step, the attentive mask will be learned using Sparsemax. Sparsemax is basically an alternative to Softmax activation function. Sparsemax function outputs sparse probability, and it assigns zeros to some elements. In Softmax, no value should ever be zero and it turns inputs into probabilities of sum of 1, but in Sparsemax there's value of 0 (feature becomes sparse) and turns inputs into probabilities with sum of 1. With the feature of having probability of 0 in Sparsemax, we can tell which feature is more important obviously.
4. **Step-Specific Feature:** The selected important features will then be passed through a step-specific feature transformer. This block has the same structure as shared feature transformer block, but it will not be reused. For step specific transformers, a new one at each decision step to further refine the input for that step.
5. **Decision Step Output:** Every decision step will then output a contribution to the final prediction. One part of the transformed data will also be passed to the next step. This allows the model to accumulate information step-by-step.
6. **Aggregation:** The outputs will now "aggregated" or sum up to produce the final prediction. Each prediction step adds its partial output to a running total. Once all the steps are completed, this step sums up the partial output and becomes the final prediction.
7. **Loss Function:** This model mainly uses the main loss (MSE for regression model and cross entropy for classification). In addition, this model contains an additional sparsity regularization loss, which is the λ_sparse

hyperparameter. This loss encourages the attention of the sparse, which ensures that only important features are selected in each step of the training.

Overall Workflow(Arik & Pfister, 2021): (a) The TabNet "encoder" includes feature transformers, attentive transformers, and masks to select important features at each step. The output is split part goes to the next step: final prediction. **(b)** The "decoder" demonstrates the encoder and is used for tasks like reconstruction in self-supervised learning.**(c)** Each feature transformer block has four layers: two shared across steps, and two specific to each step. Layers include fully connected units, batch normalization, and gated linear units (GLU) for flexible learning.**(d)** The "attentive transformer" uses prior feature usage and Sparsemax to create sparse masks.

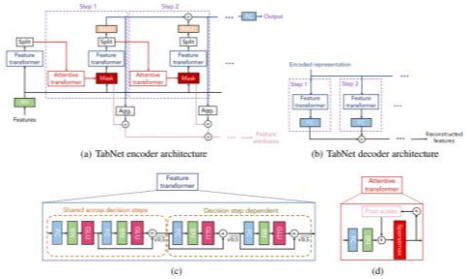


Figure1:The Overall Workflow of TabNet Model(Arik & Pfister, 2021)

Recent Advancements or Adaptation: Researchers have experienced Entmax function as an alternative to Sparsemax function to soften sparsity (Peters et al., 2019). More integration with modern optimizers such as RAdam has been discovered to improve convergence stability on imbalanced data. Moreover, TabNet has been included in libraries like PyTorch TabNet (DreamQuark, 2021) and Google Cloud AutoML for easy access.

Evolution of Tabnet: TabNet was inspired by the success of gradient boosting models on tabular data, adopting sequential decision refinement like boosting. The use of attention mechanisms and Sparsemax draws from advances in NLP models like Transformers (Arik & Pfister, 2021).

Applicability to the Research Question with Specific Example: Tabnet is increasingly used in industries such as finance, healthcare, and retail nowadays. These intersections of industries are commonly centered within the structured tabular data, which is like the dataset we used for this assignment. A study suggests that TabNet achieved good accuracy and interpretability compared to traditional models like XGBoost in insurance risk pricing tasks (Wang et al., 2023). This indicates TabNet's capability to handle structured data with complex feature datasets effectively just like our dataset.

1.3 Model Development:

Data Preprocessing:

Feature Scaling: All the numerical features were standardized using StandardScaler() from sklearn.preprocessing. This function ensures each feature had a mean of 0 and standard deviation of 1. This step avoids dominance by features with larger scales.

Label Encoding for Categorical Variables: LabelEncoder() function is used to allow Tabnet learn categorical embeddings. For example, a feature with values ["diesel", "electric", "petrol"] might be encoded as [0, 1, 2]. It enables handling them efficiently while maintaining semantic meaning through learned embeddings.

Target Normalisation: The function of min-max normalization on our target variable "price" scales it to the 0 to 1 range. The TabNet model was implemented using the TabNetRegressor class from the pytorch_tabnet package.

Key Hyperparameters:

1. **n_d(number of features in the decision layer):** This parameter controls the size of the number of features used to make predictions. Larger values indicate more patterns and more computation.
2. **n_a(number of features in attention transformer):** Controls the size of attention transformer.
3. **n_steps(number of decision steps):** Determines how many times the model refines its representation before the output of final prediction. 3–5 steps are often sufficient.
4. **Gamma(feature reusage relaxation parameter):**Higer values(e.g., 1.5–2.0) indicate more feature reuse through the steps. Lower values increase sparsity.
5. **lambda_sparse(weight of sparsity regularization loss):**controls how aggressively the model pushes feature masks. Higher value means that fewer features were selected in each step.
6. **optimizer_params(parameters of the optimizer):** Includes the learning rate of training. Adam optimizer was used with a learning rate (lr) tuned via Optuna, an advanced hyperparameter optimization tool. 1e-3 and 1e-1 in this case. Learning rate directly affects **convergence speed and stability**.
7. **mask_type(Type of feature mask activation):**Sparsemax activation is set in this parameter. Alternative tuning option is "entmax" activation function.

2. Model Evaluation and Optimization

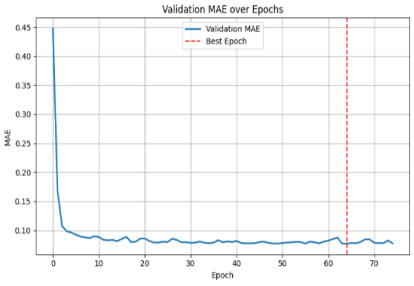
2.1 Model Evaluation:

The TabNet model's performance was evaluated using mainly three metrics: Mean Absolute Error (MAE), Root Mean Squared Error(RMSE), and R^2 Score. By using these metrics, we can evaluate the model's performance under some conditions that include class imbalance and data noise.

MAE: MAE averages the absolute difference between predicted and actual prices. It is more sensitive to outliers and noise than RMSE. This evaluation metric is suitable for car pricing where some listings may contain inconsistent values (Willmott & Matsuura, 2005).

RMSE: RMSE is the square root to MAE. RMSE penalizes larger errors and prevents large prediction mistakes on certain

examples. (very old vehicles or luxury cars) This metric can significantly reveal the price imbalance(Chai & Draxler, 2014), which is important in our context, since instances could skew performance if that is not detected.
R^2: This metric indicates the proportion of the variance for the price in this case. Also, it provides an overall sense of fitness. However, sometimes it misleads the data imbalance. Therefore, it is necessary to include MAE and RMSE for a more fair and balanced evaluation process(Kuhn & Johnson, 2013).
Results: The Initial TabNet model achieved the following results on the test set(30% of the dataset):
RMSE: 0.1133 MAE: 0.0829 R^2: 0.4193
The final TabNet model achieved the following results on the test set(30% of the dataset):
RMSE: 0.1068 MAE: 0.0771 R^2: 0.4844 Adjusted R^2: 0.4827
These metrics reflect a **low average error**. While RMSE and MAE indicate accurate predictions, the R^2 value reveals that less than half of the variance in price is captured. It suggests opportunities for further improvement. This outcome is reasonable given the dataset's complexity, heterogeneity (varied car models, brands, ages), and likely class imbalance, since there's fewer examples of high-end or uncommon vehicles. From a practical perspective, the model is effective at predicting prices for typical cases but struggles with edge cases. This is the behavior of many machine learning models on real tabular data where data imbalance and noise distort performance.



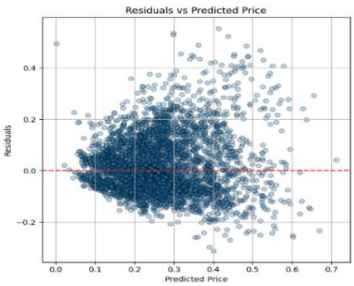


Figure2:Validation MAE over Epochs Figure3:Actual vs Predicted Prices Figure4:Residuals vs Predicted Prices
Results' Plots Explanation: Figure 2 demonstrates the validation MAE across epochs. It shows rapid convergence within the first 10 epochs and plateauing afterward. The low and stable MAE suggests that the model has effectively learned without overfitting. Figure 3 shows a positive linear correlation between actual and predicted prices. Most predictions cluster near the ideal line. However, a slight underestimation is observed in higher price ranges, possibly due to the skewed distribution of vehicle prices. Figure 4 shows residuals scattered symmetrically around zero with no visible trend. This pattern suggests that the model's errors are not systematically biased. It supports the model's generalization ability.
Recommendation for Future Work: Data augmentation for tabular data can be used in future work. Recently emerged for tabular data. Applying methods such as Gaussian note injection could possibly reduce prediction bias. Additionally, exploring segmented models for different vehicle categories may improve accuracy and interpretability.

2.2 Model Optimisation:

To improve model performance, we implemented the advanced hyperparameter optimization, Optuna framework to select the best parameters for our final model training. It is a powerful alternative to the traditional grid search. Optuna is support for efficient Bayesian search by "Tree-structured Parzen Estimators". Bayesian search has been proved that it performs better than traditional grid search in both context of speed and search quality(Akiba et al., 2019). In addition, all the trials are trained using 5- fold cross validation, and the mean MAE was used as minimized. During hyperparameters training, repeated k-fold contains 5 splits and 3 repeats for 30 trials.
Repeated K-Fold: It was chosen over standard K-Fold to reduce variance in validation scores and provide a more robust estimate of model generalization (Kuhn & Johnson, 2013). 5 folds and 3 repeats resulted in 30 train-test combinations per trial improving reliability during hyperparameter tuning.
Critical TabNet Hyperparameters for Optimization with Justification:

1. **n_d ,n_a:**[8,16,32,64]. Smaller values such as 8 or 16 reduce the model training time. Larger values such as 32 or 64 allow the model to detect more complex feature interactions, which is important in used car listings datasets.
2. **n_steps(number of decision steps):**3 to 5. Fewer than 3 steps may be underfit. However, 5 steps may bring longer training time. The range of 3-5 is a good combination between model depth and efficiency (Arik & Pfister, 2021).
3. **Gamma(feature reusage relaxation parameter):**1.0 to 2.0. Value closer to 1.0 will improve sparsity and gerneralisation, since each step selects mostly different features. On the other hand, 2.0 will allow more feature overlap, which improves performance in the scenario of certain features are important. The range of 1 and 2 is recommended by the authors of TabNet(Arik & Pfister, 2021).
4. **lambda_sparse(weight of sparsity regularization loss):** 1e-5 to 1e-1(log scale). Lower values will apply lower penalties. Higher values (1e-1) force the stricter feature selection. This parameter range is also recommended by the author of TabNet(Arik & Pfister, 2021).
5. **optimizer_params(parameters of the optimizer):** 1e-3 to 1e-1 (log-scale). A log-scale range from 0.001 to 0.1 is commonly used in deep learning. This range also gives Optuna flexibility to find the best value to stabilize training.
6. **mask_type(Type of feature mask activation):** ["Sparsemax", "entmax"]. Sparsemax is a default attention mask in this model. In recent, a newer alternative, Entmax has been discovered to stabilize training on noisy data(Peters et al., 2019). Including both activation functions adapt between hard and soft attention mechanisms.

Outcome Summary: Through hyperparameter tuning and evaluation, the final TabNet model shows strong local accuracy in terms of global generalization (R^2). It is a well-performing, interpretable, and practical for used car price prediction.

Unikey: awan0950 Model: CatBoost

1. Predictive model

1.1. Model Description

CatBoost is a machine learning algorithm developed by Yandex (a Russian company). It belongs to a type of Gradient Boosting Decision Tree (GBDT) and the family of ensemble methods. It performs better in solving categorical features and avoiding overfitting using the techniques 'ordered boosting' and 'target statistics'.

Assumptions: CatBoost does not rely on usual assumptions such as linearity, normal distribution of features, or homoscedasticity, etc. Hence, it is suitable for real-world datasets with mixed feature types (numerical and categorical). This aligns well with our dataset, which includes both types of variables, such as vehicle mileage (numerical), weight (numerical), gearing type (categorical), and fuel type (categorical).

Strengths and weaknesses: As I showed above, CatBoost can handle categorical data without one-hot encoding and reduce overfitting to some degree via ordered boosting. However, it also has limitations. Firstly, it is less interpretable than linear models. For example, CatBoost builds an ensemble of decision trees, so it does not have explicit mathematical equations (such as linear regression), making the overall learning process more complex and less transparent. Secondly, the performance of CatBoost largely depends on parameter tuning (such as learning_rate, depth, iterations, and l2_leaf_reg). Overall, CatBoost is well-suited to this regression task as it can learn from complex features and achieve robust performance without heavy preprocessing.

1.2. Model Algorithm

Algorithm principle: CatBoost is similar to other GBDT models, and the basic idea is: build decision trees one by one, and each tree tries to correct errors of the previous model. In each round, the model calculates the "gradient of the current error" (direction), and then the new tree is trained based on this gradient. The results of all the trees are weighed and combined into the final prediction.

CatBoost aims to minimize the loss function (such as RMSE , Logloss, or MAE):

$$F(x) = \sum_{m=1}^{M} \eta \cdot h_m(x)$$

Where $h_m(x)$ is the prediction result of the mth tree, η is the learning rate, M is the total number of trees.

Training process and pseudocode:

Step 1: Initialize the model. First, calculate an initial value (such as the mean value of the label) using the training set as the initial predicted value $F_0(x)$.

Step 2: Compute the gradient (the derivative of the loss function with respect to the model output). This indicates 'how much the prediction of the current model deviates from the true value'.

$$g_i = \frac{\partial \mathcal{L}(y_i,\ F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}$$

Step 3: Build a new tree, $h_m(x)$ to correct errors.

Step 4: Update the model. And the model will be closer and closer to the real one.

$$F_m(x) = F_{m-1}(x) - \eta \cdot h_m(x)$$

The following pseudocode presents the process:

```
Input: Training data D
Initialize model F_0(x) = average(y)

For m = 1 to M (iterations, each round of boosting):
    1. Compute gradients for each sample
    2. Build a decision tree
    3. Update the  model

Return final model F_M(x)

Output: Final predictive model F_M(x)
```

Important hyperparameters:

parameters	Illustration
iterations	Number of training rounds. More iterations can improve accuracy but increase training time and risk of overfitting.
learning_rate	The contribution ratio of each tree to the total prediction. Deeper trees can capture more complex patterns but may overfit.
depth	The depth of the decision tree. A lower value (e.g., 0.01 or 0.05) leads to slower learning but may have better generalization.
l2_leaf_reg	L2 regularization term coefficient. Higher values make the model more conservative.
loss_function	Loss function type. RMSE and MAE are alternatives for the model (may depend on whether robustness to outliers was more important).
eval_metric	Evaluation metric.

Potential variations with CatBoost: There are three other parameters in the model that have different values to choose:

parameters	choice	influence
loss_function	'RMSE', 'MAE', 'Quantile', etc.	Different tasks with different goals have different choices.
bootstrap_type	'Bayesian', 'Bernoulli', 'MVS'	Control the data sampling method and affect robustness and speed.
grow_policy	'SymmetricTree', 'Depthwise', 'Lossguide'	Control the growth mode of the tree and affect the fitting ability.

1.3. Model Development

Data preprocessing: In this model, I did not apply any techniques for data preprocessing. The following three reasons are given. First of all, I use the dataset that has been preprocessed in the previous assignment. Hence, there are no missing values and outliers in the dataset. Furthermore, CatBoost can natively handle categorical features, which means it is not necessary to transfer categorical features to numeric representations (CatBoost internally transforms them using its own 'target statistics with permutation' technique). Finally, CatBoost is a tree-based model and therefore insensitive to the scale of numerical features. Consequently, feature normalization or standardization is not required. Overall, I use the clean dataset in assignment 1 directly without any additional preprocessing.

Building an initial model and parameter selections: Due to our regression problem (predicting price), the class ***CatBoostRegressor()*** from ***catboost*** is used to create a CatBoost regression model object. Here, four parameters are set. I choose **iterations = 100**, because it is a reasonable starting point (without early stopping), which is sufficient to achieve initial results, and it allows the training time to be short, being helpful for parameter tuning. The **learning_rate** is set to be **0.05**, for the reason that this learning rate will neither be too slow nor too easy to overfit with previous small iterations (100). Then, **depth = 6** and **l2_leaf_reg=3** are classical and moderate value choices for these two parameters. They can avoid overfitting to some degree. (Notice: the **loss function** is not included here because I use the default '**RMSE**', which is suitable for this price prediction task)

2. Model Evaluation and Optimization

2.1. Model Evaluation

The initial model set in the previous section is trained, and I evaluate the three metrics (RMSE, MAE, and R-squared) in the test set of this model to see its performance in this regression task. The Repeated K-fold Cross-Validation with n_splits = 5 and n_repeats = 3 is applied to calculate the average values of these three metrics, and it provides robust generalization performance estimation for the model. The initial CatBoost model got an average MAE of 4887.58 and an RMSE of 6420.60, indicating that the model's price predictions deviated from the actual values by around 4,800-6,300 units on average. The R-squared, with a score of 0.5344, suggests that approximately 53% of the variance in the car prices is explained by the model, which means that a moderate level of prediction is achieved. While the model captures some important patterns, the gap between RMSE and MAE shows the existence of some large prediction errors. Although class imbalance is not a problem considered in this regression task, the results suggest potential areas for improvement. Further hyperparameter tuning may help the model to improve performance.

2.2. Model Optimization

Grid-Search: In order to enhance the performance of the initial CatBoost model, we applied hyperparameter tuning using grid search to go through all the parameter combinations. We defined a search space for the following three important hyperparameters based on theoretical considerations and prior experiences:

depth: [6, 8] — to control tree complexity and capture deeper feature interactions.

learning_rate: [0.03, 0.05, 0.07] — to balance learning speed and convergence stability (as I said before, smaller learning rates are chosen because of the small iterations, and this can avoid overfitting).

l2_leaf_reg: [1, 3, 5] — to regularize the model and prevent overfitting.

We generated all possible parameter combinations using the grid setting and evaluated each configuration by Repeated K-Fold Cross-Validation (5 splits, 3 repeats). The performance was assessed based on MAE, which is less sensitive to outliers and capable of providing a more stable measurement of overall deviation in vehicle price prediction. After testing all combinations, the best parameter should with the minimized average MAE. Hence, from this, we can see an improvement over the initial model.

Results of final model: The grid search gives the best parameter combination with depth = 8, learning_rate = 0.07, l2_leaf_reg = 1 (achieved a minimum average MAE of 4827.93 across cross-validation folds). Overall, models with deeper trees (depth=8) have better performance than those with shallower ones, implying the complex relationships in the data. A relatively higher learning rate (0.07) leads to better convergence without overfitting. For regularization, lighter settings (l2_leaf_reg = 1 or 3) performed better, which means slight regularization is more appropriate, while excessive regularization (=5) may make the model overly conservative. The final tuned CatBoost model slightly outperformed the initial model on the test set. The MAE decreased from 4887.58 to 4855.83, and RMSE dropped marginally from 6420.60 to 6419.98. However, the R^2 score remained unchanged at 0.5344, indicating that the variance explained by the model did not increase. These results suggest that while hyperparameter tuning improved the accuracy of predictions to a small level, further improvement might require additional strategies such as increasing iterations and setting early stopping.

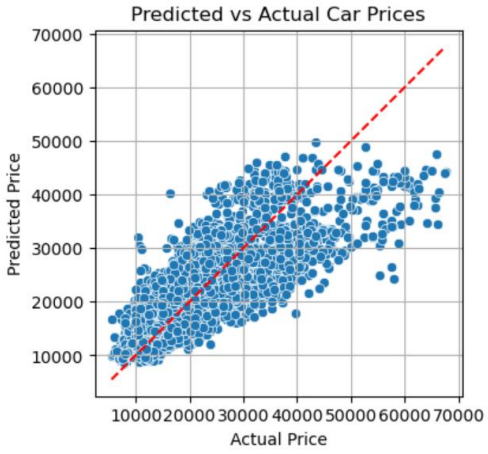


Figure1: Predicted vs Actual Car Prices

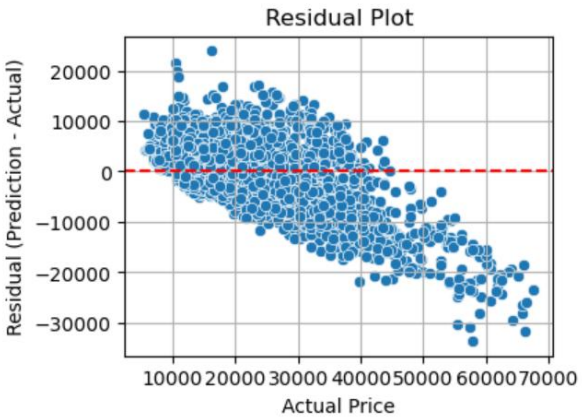


Figure2: Residual Plot

As shown in Figure 1, the predicted values follow the general trend of the actual prices, and they are close to the diagonal reference line, indicating that the model has reasonable predictive power. However, at the range of higher prices (>50,000), the predictions tend to underestimate the actual prices, which may suggest the model is difficult to capture the characteristics of more expensive second-hand vehicles. The residual plot (Figure 2) shows a clear negative trend, in which residuals decrease as actual prices increase. This indicates that the model overestimates low-priced vehicles and underestimates high-priced ones, implying that the model might fail to capture some of the nonlinearities and features relevant to predicting higher-end car prices. Ideally, residuals should be randomly scattered around zero without any clear trend. The model's prediction results suggest several possible directions for further improvement. First, using a log transformation on the price variable may help reduce the effect of extreme values and improve prediction for prices of expensive vehicles. Additionally, using sample weighting might balance the model's focus across different price ranges. Finally, analyzing samples with high errors may help recognize specific patterns that the current model fails to capture.

Group Component 2

1. Discussion:

Overview of all models:

This project evaluated three advanced regression models, including CatBoost, XGBoost, and TabNet. They are selected for their proven effectiveness in handling structured data. CatBoost and XGBoost are powerful gradient boosting algorithms known for their high performance in tabular datasets. TabNet offers a deep learning approach that utilizes attention mechanisms to dynamically select relevant features. The combination of these models enables a comprehensive comparison of tree-based versus neural network-based methods in this task.

Strength and Limitations:

Strengths and limitations of CatBoost:

CatBoost has several significant advantages. Firstly, it natively supports categorical variables, allowing it to directly process features such as fuel type or vehicle type without manual encoding, which simplifies data preprocessing. Secondly, it includes techniques to prevent overfitting, such as 'ordered boosting' and 'target encoding', which help avoid information leakage and improve generalization. Thirdly, feature scaling or normalization is not required, making the model easier and more effective to build.

However, the model also has several limitations. It is a 'black box' model with weak interpretability (unlike linear regression), because it cannot clearly explain how each feature affects the prediction. Additionally, CatBoost is sensitive to hyperparameters, and its performance depends heavily on a proper combination of settings like depth, learning_rate, and l2_leaf_reg. The time cost of tuning can also be high, especially when using Grid Search with Repeated K-fold Cross-Validation. Furthermore, based on the residual plots, the model shows limited ability to handle extreme values (it tends to underestimate high-priced vehicles), indicating difficulty in learning from skewed distributions (Prokhorenkova et al., 2018).

Strengths and limitations of XGBoost:

XGBoost is well suited to our task due to its ability to model complex, non-linear relationships and handle mixed data types without extensive preprocessing (Chen & Guestrin, 2016). It captures interactions between features like Mileage and Horsepower, and its regularization mechanisms help prevent overfitting—especially valuable with high-cardinality features such as Make_Model.

However, its black-box nature limits interpretability, and tuning can be computationally expensive. Feature importance scores alone do not fully explain predictions, which may hinder transparency in practical use.

While XGBoost offers strong predictive power for this task, its complexity and reduced transparency highlight the trade-off between accuracy and interpretability.

Strengths and Limitations of TabNet:

TabNet is a deep learning architecture specifically designed for tabular data. It uses a sequential attention mechanism to learn which features to focus on at each decision step. It enables both feature selection and interpretability in a neural network context. This makes TabNet particularly appealing for structured datasets such as car price prediction.

One of TabNet's key strengths is its ability to model non-linear relationships effectively without requiring manual feature pre-processing. The use of feature-wise attention allows the model to adaptively focus on the most useful features during training. Unlike tree-based models, TabNet can provide end-to-end gradient learning, which can benefit from large datasets and GPU acceleration. Moreover, it can produce sparse masks that offer a form of interpretability by capturing which features contribute to prediction.

However, TabNet also presents limitations. Its performance is highly sensitive to hyperparameter tuning, including learning rate, number of decision steps, and virtual batch size. Without careful tuning, the model may underperform compared to more established methods like XGBoost. TabNet also demands significantly more computational resources and training time, which can be a barrier for practical use in limited environments (Arik & Pfister, 2021).

Quantitative comparison (with MAE & R^2 charts)

Summary Table of Model Performance:

The table below summarizes the performance of three predictive models—CatBoost, XGBoost, and TabNet—based on Mean Absolute Error (MAE) and R^2 Score.

Table 1: Summary Table of Model Performance

Model	MAE	R^2 Score
CatBoost	4855.83	0.5344
XGBoost	4770.90	0.5634
TabNet	4885.83	0.4824

Bar Chart Visualisation for Crucial Metrics

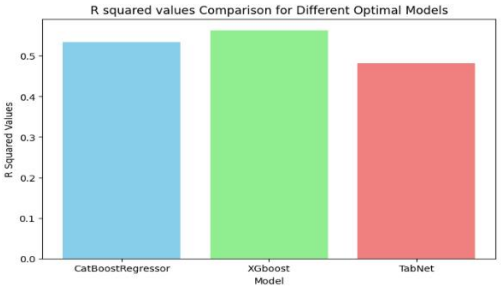


Figure 1: R^2 Comparison of Different Models

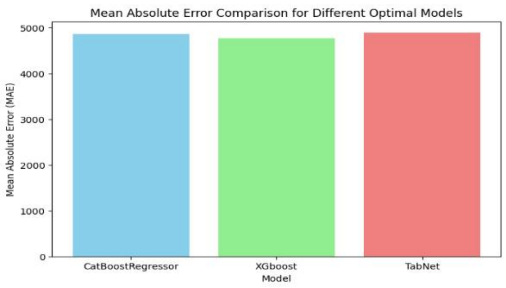


Figure 2: MAE Comparison of Different Models

Interpretation, Analysis, and Reasoning

Among the three models, XGBoost achieved the lowest MAE (4770.90) and highest R^2 score (0.5634). This indicates its well-performing ability to minimize prediction error and explain variance in the target variable. CatBoost performed slightly worse with an MAE of 4855.83 and R^2 of 0.5344, while TabNet had the lowest predictive performance (MAE = 4885.83, R^2 = 0.4844). These results align with the qualitative observation that TabNet may require larger datasets and more tuning to match the accuracy of tree-based models. Overall, XGBoost's consistency across both metrics supports its selection as the optimal model.

XGBoost likely performed best due to its ability to capture feature interactions through gradient boosting, especially in a dataset with mixed variable types and moderate non-linearity. Its regularization parameters (e.g., gamma and lambda) helped control overfitting. CatBoost's performance was slightly behind but consistent, likely due to its original treatment of categorical features such as vehicle type. For TabNet, although it is architecturally sophisticated, underperformed possibly due to insufficient data volume and its reliance on deep attention layers. This can cause overfitting on smaller tabular datasets. These observations support the conclusion that tree-based models remain more practical for tabular regression under current dataset constraints.

For example, CatBoost accurately differentiated price ranges between "hybrid" and "diesel" vehicles without requiring encoding, improving stability and reducing preprocessing. XGBoost performed particularly well on mid-range cars like a 5-year-old Toyota Corolla with 60,000km. It captures non-linear interactions between age and mileage. In contrast, TabNet tended to underestimate luxury car prices, such as predicting a $120,000 vehicle at around $100,000. This example reflects its difficulty with outliers and high prices.

Border Implications in terms of Analysis

The choice of predictive model has direct implications for both the accuracy and the practical implementation of car pricing systems. XGBoost's high accuracy and well performance make it well-suited for real-time pricing engines used in online marketplaces. However, its lack of transparency may cause challenges while predicting price estimates to users. CatBoost offers comparable accuracy with better handling of categorical data. It makes it more appealing in systems where fewer preprocessing is acceptable. TabNet, while promising for complex, non-linear relationships, requires substantial computational resources and large datasets to reach optimal performance. This makes it less ideal for lightweight or mobile applications at this stage. Therefore, model selection should not be only based on accuracy but also consider stakeholder needs, deployment environments, and ethical transparency.

2. Conclusion

In response to our research question "*Can we accurately predict the market price of used vehicles based on vehicle specifications and usage conditions?*", our findings show that this is achievable using tree-based ensemble models like XGBoost, which balance accuracy, speed, and effectiveness. Quantitatively, XGBoost demonstrated the strongest performance. XGBoost achieves the lowest Mean Absolute Error (4770.90) and the highest R^2 score (0.5634). This indicates superior predictive accuracy and variance explanation in car price prediction. CatBoost has the results of an MAE of 4855.83 and R^2 of 0.5344. CatBoost benefits from its native handling of categorical features and less pre-processing process. TabNet, while conceptually ideal and sophisticated based on its attention-based deep learning structure. As a result, TabNet underperformed (MAE = 4885.83, R^2 = 0.4844). It is likely due to its sensitivity to data size and sensitivity of the outliers.

Given these findings, XGBoost is recommended as the most effective predictive model. This recommendation is justified by empirical results, its ability to capture nonlinear interactions between features (e.g., mileage and age), and domain-aligned strengths such as scalability and performance stability. While it lacks full transparency, its advantages in performance, speed, and generalization make it ideal for practical deployment in automated car valuation systems.

For future work, we propose several aspects of the research:

- Data Collection: Expanding the dataset to include additional vehicle attributes such as accident history or dealership location to improve feature richness.
- Model Refinement: Conducting advanced hyperparameter optimization using Optuna or Bayesian Search. Moreover, we can possibly experiment with ensemble stacking to combine XGBoost and CatBoost outputs.
- Interdisciplinary Collaboration: Working with domain experts to better understand market factors influencing pricing, which could guide the inclusion of external variables.

These directions strongly support our goal of building an accurate and scalable car price prediction model in this research.

References

Arik, S. O., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8), 6679–6687. https://doi.org/10.1609/aaai.v35i8.16776

Peters, M. E., Niculae, V., Martins, A. F. T., & Smith, N. A. (2019). Sparse Sequence Modeling with Entmax. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

DreamQuark. (2021). *PyTorch TabNet*. GitHub repository. https://github.com/dreamquark-ai/tabnet

Wang, Y., Chen, X., Li, D., & Liu, J. (2023). An interpretable machine learning approach for insurance premium prediction: A TabNet model-based framework. *Expert Systems with Applications*, 213, 119543. https://doi.org/10.1016/j.eswa.2023.119543

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1), 79–82. https://doi.org/10.3354/cr030079

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), 1247–1250. https://doi.org/10.5194/gmd-7-1247-2014

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer. https://doi.org/10.1007/978-1-4614-6849-3

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. Advances in Neural Information Processing Systems, 31. https://arxiv.org/abs/1706.09516

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

m0_58475958. (2021, July 25). Machine Learning: Introduction to XGBoost and Formula Derivation [Blog post, in Chinese]. CSDN. https://blog.csdn.net/m0_58475958/article/details/119077906

Appendix

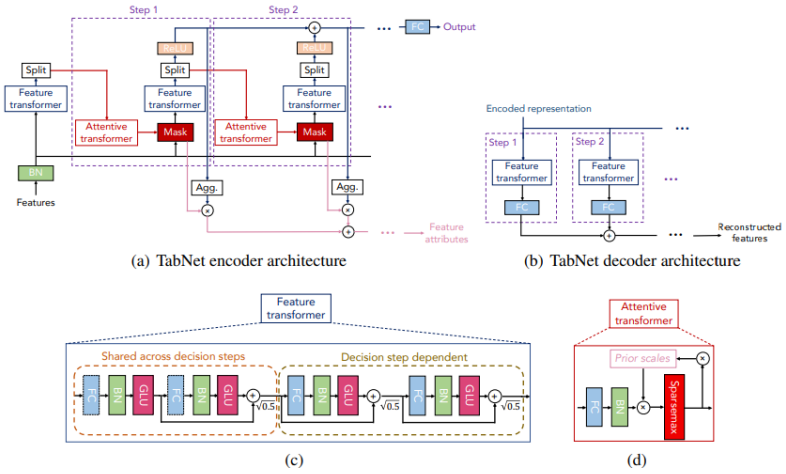


Figure1:The Overall Workflow of TabNet Model(Arik & Pfister, 2021)

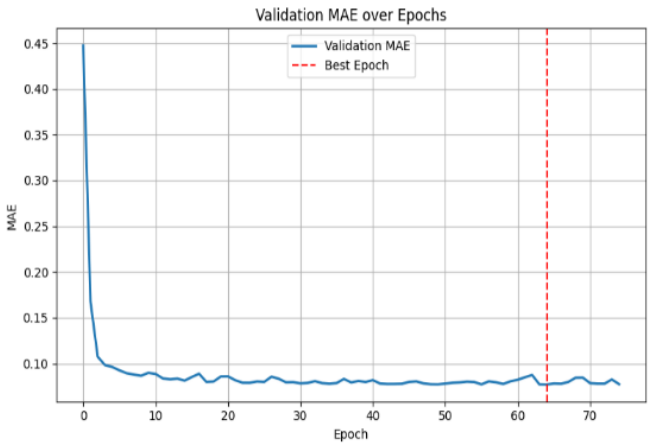


Figure 2:TabNet Validation MAE over Epochs

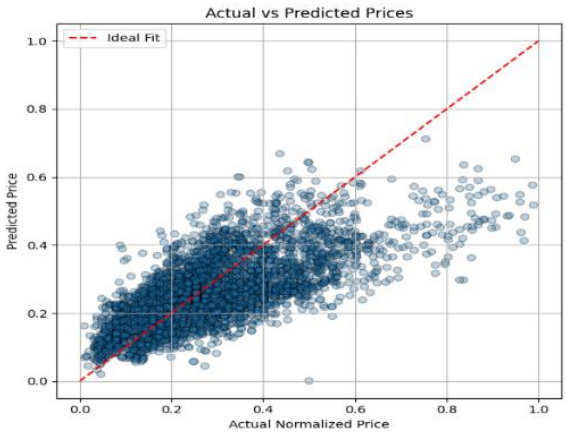


Figure3: TabNet Actual vs Predicted Prices

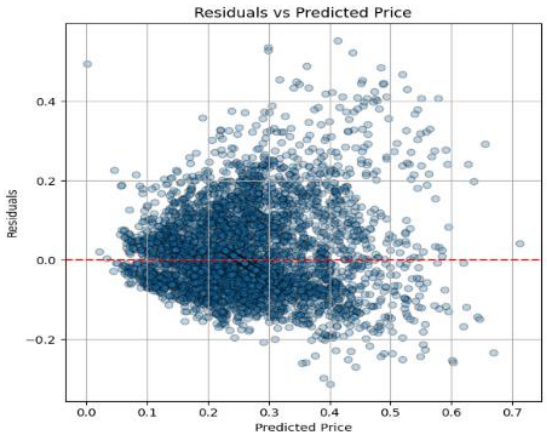


Figure4: TabNet Residuals vs Predicted Prices

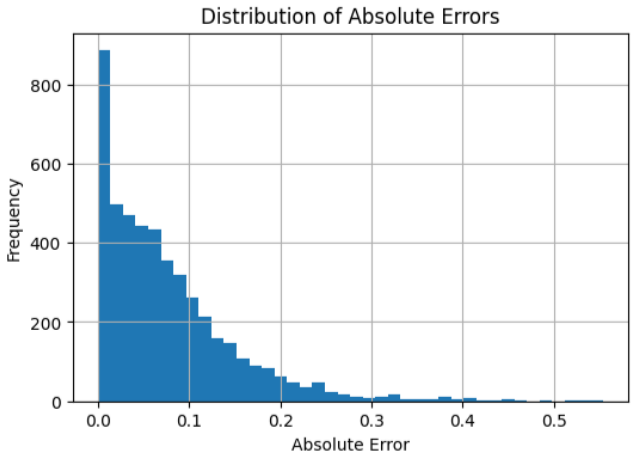


Figure 5: TabNet Distribution of Absolute Errors

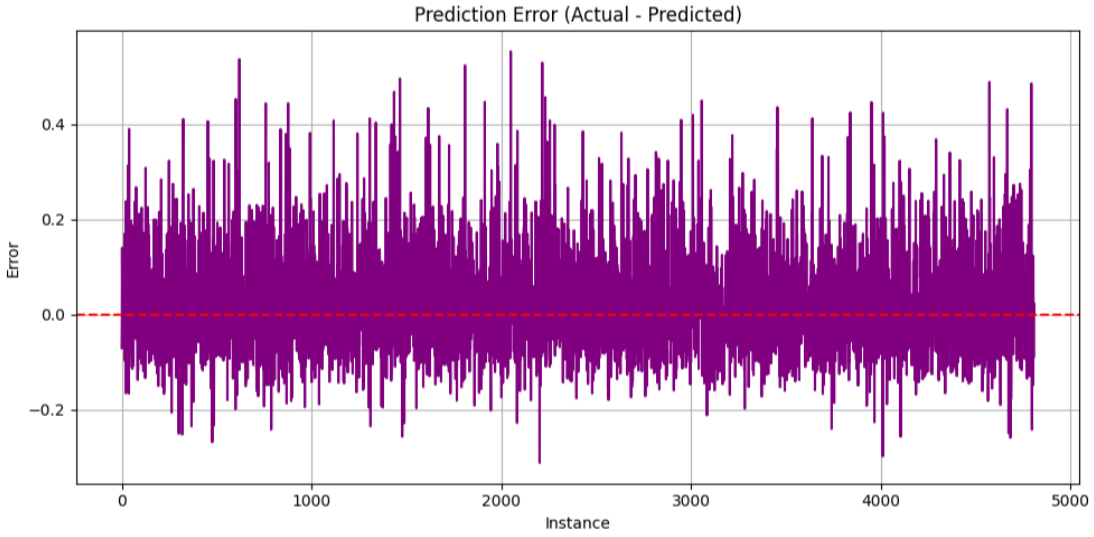


Figure 6: TabNet Prediction Error Frequency