Predicting Used Car Prices With Regression

Elum Nyabando
University of Minnesota - Twin Cities
CSCI 4521 - Applied Machine Learning
Shelby Ziccardi
May 9th, 2025

Introduction

The central objective of this project was to develop predictive models that estimate the selling price of used cars based on their various features. With the rise in online vehicle resale platforms, accurate car price predictions are essential for both buyers and sellers to make informed decisions. The driving question guiding this work was: *How accurately can we predict the resale value of a used car given its attributes such as year, km, fuel type, transmission type, and seller type?* To address this, I implemented a series of linear, nonlinear, and neural network models to evaluate and compare their predictive performance, using threshold-based accuracy metrics and conventional error measurements such as RMSE and R^2. The aim was not only to benchmark simple regression methods but also to explore how much gain could be achieved by introducing interaction terms, nonlinearity, and machine learning-based approaches.

Vehicle Dataset

The dataset used in this project was sourced from the publicly available “CAR DETAILS FROM CAR DEKHO.csv” file from Kaggle, which contains historical listings of used cars posted on the CarDekho platform, an Indian online vehicle marketplace. The dataset included essential attributes for each vehicle such as the year of manufacture, kilometers driven, fuel type, transmission type, seller type, and the target variable: selling price.

To prepare the data for modeling, several preprocessing and feature engineering steps were conducted. First, a new feature called car_age was created by subtracting the car’s manufacturing year from 2025, converting the 'year' column into a more directly interpretable measure of depreciation. Continuous variables such as km_driven and car_age were standardized using z-score normalization to ensure that their scales did not skew model performance.

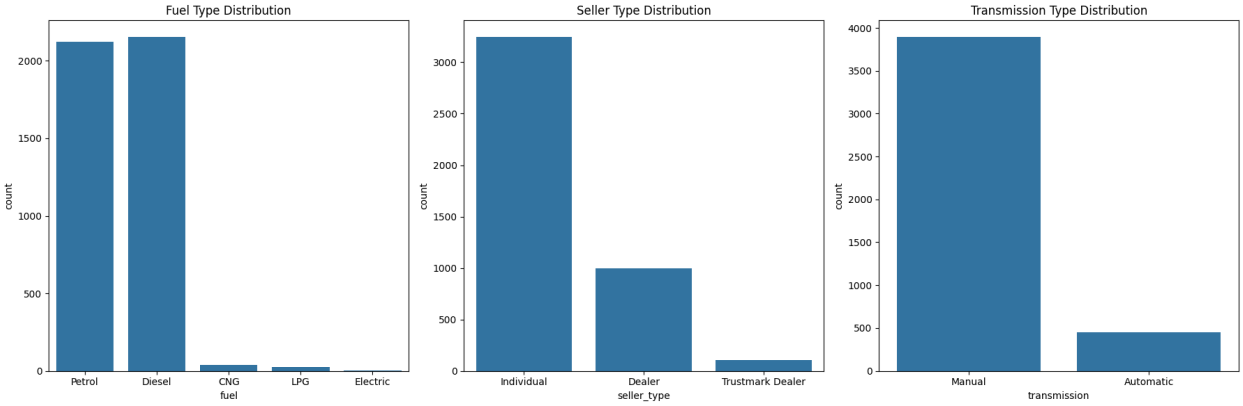
Categorical variables including fuel, transmission, and seller_type were encoded into numeric format using .astype('category').cat.codes to facilitate compatibility with regression models and neural networks. Feature selection was informed by domain knowledge and exploratory data analysis, including visual distribution checks (histograms and count plots), and a heatmap of the correlation matrix for numeric features. This analysis confirmed a moderate correlation between car age and selling price, among other relationships. While class imbalance was not a concern due to the regression nature of the task, I applied quantile-based grouping of selling prices during model evaluation to better visualize prediction accuracy across the price spectrum.

Recent advances in car price prediction have shifted toward machine learning techniques like Gradient Boosting and deep learning models, which can better capture complex, nonlinear relationships in large datasets. Industry leaders often use proprietary models enhanced with additional features such as vehicle condition, geographic trends, and user behavior data. While this project remains focused on regression and basic neural networks, the implementation of a PyTorch-based model reflects a simplified adaptation of these cutting-edge methods.

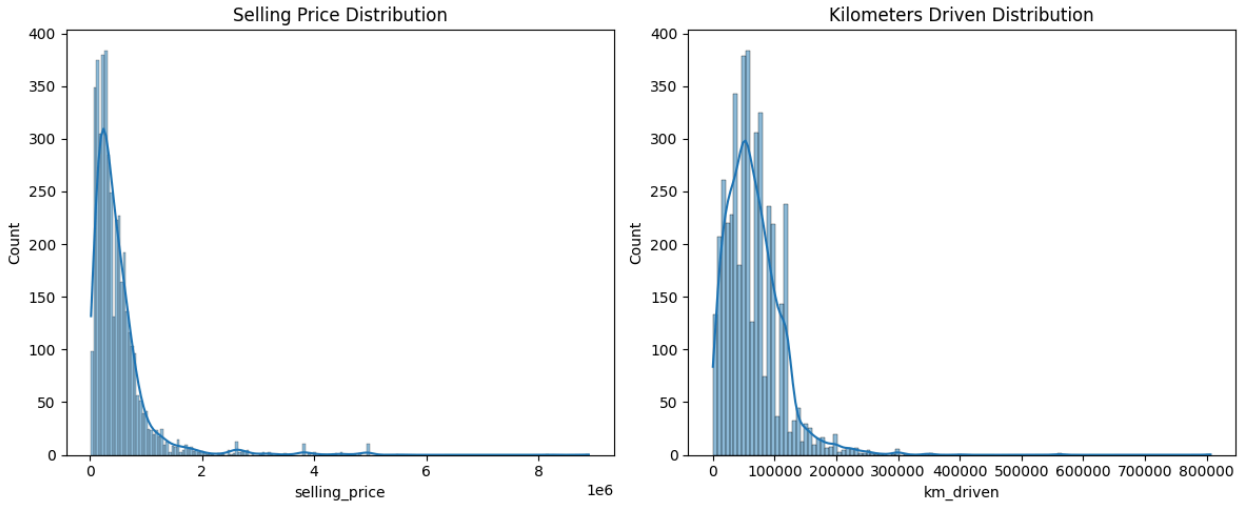
Visualization

Several visualizations were used throughout the project to explore the data, guide feature engineering, and compare model predictions:

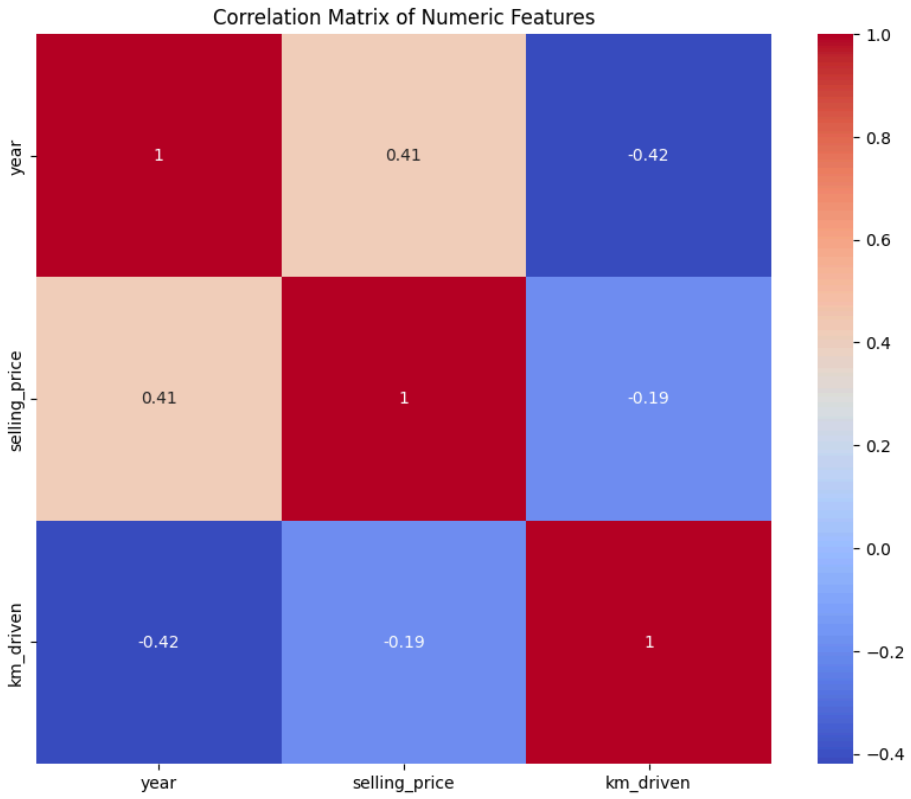
Count plots were created for fuel, seller_type, and transmission to understand the distribution of these key categorical variables. These visualizations revealed that most cars used petrol and diesel fuel, were listed by individual sellers, and had manual transmissions.



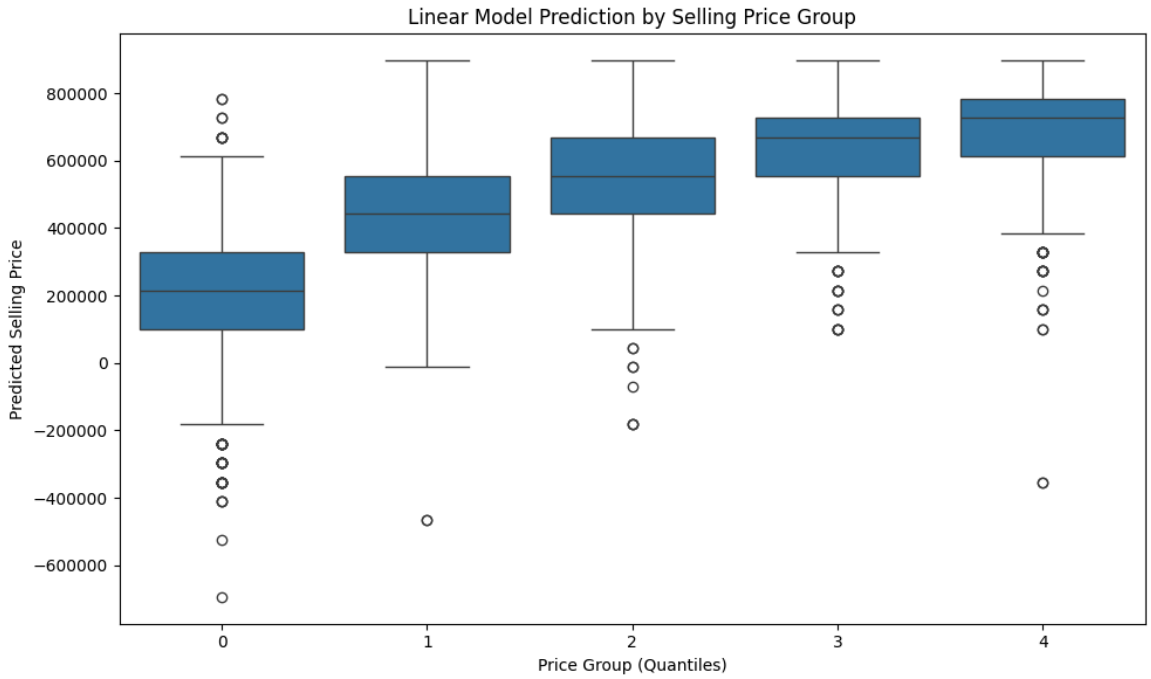
Histograms for selling_price and km_driven showed both features were right-skewed, with a long tail of higher values. This highlighted the need for normalization to stabilize variance and improve model training.

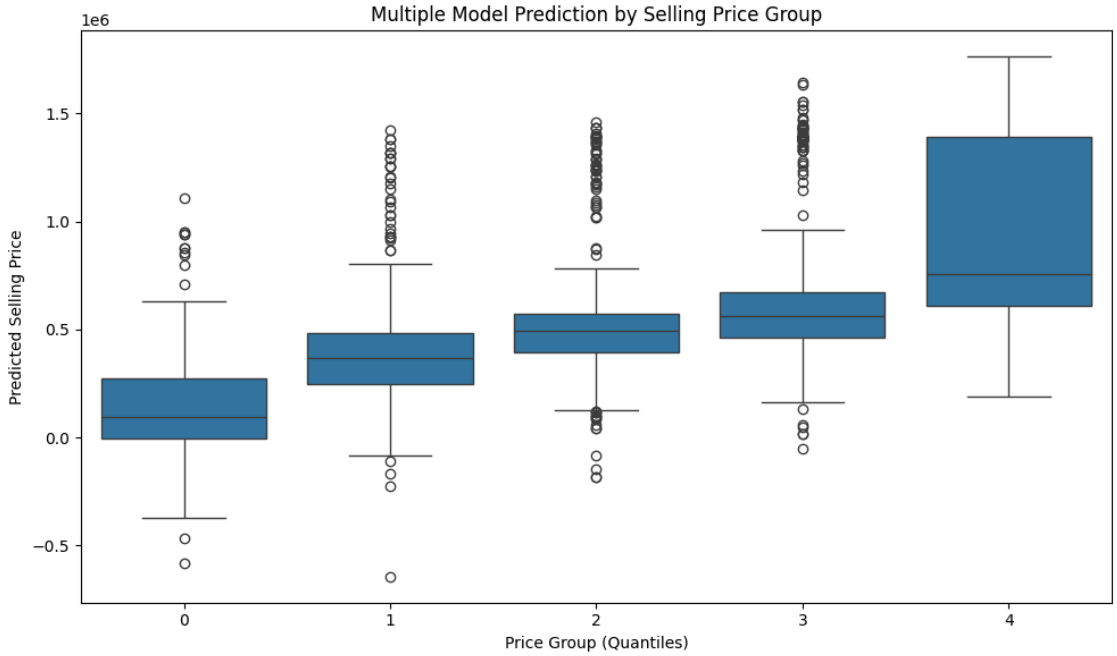


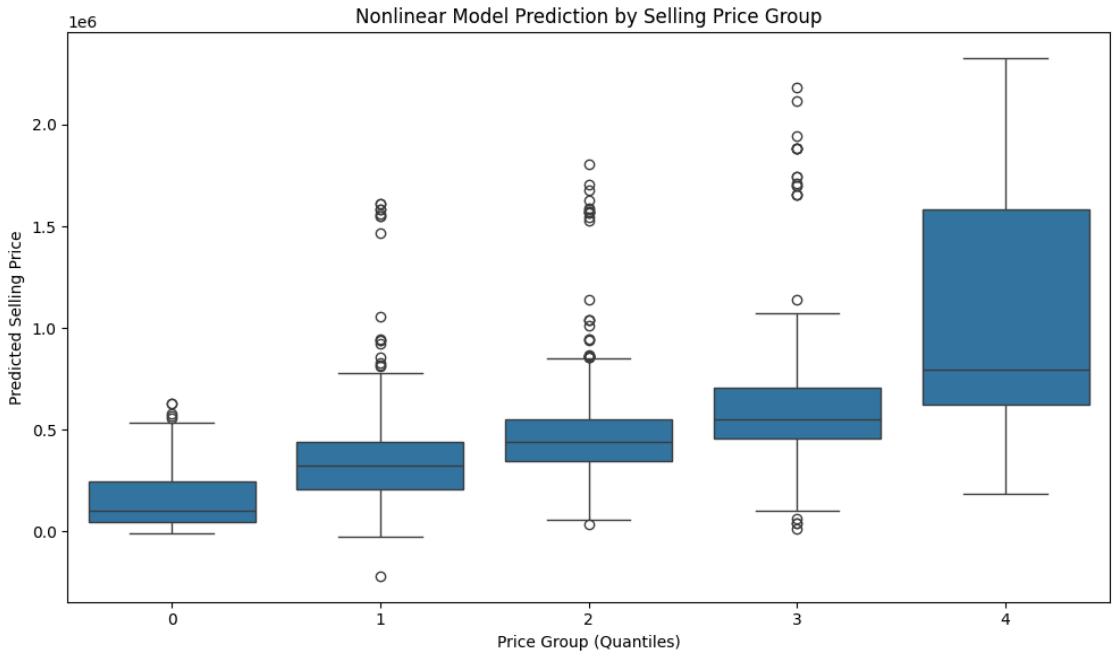
A heatmap of Pearson correlation coefficients among numeric variables helped confirm that car_age and km_driven were moderately correlated with selling_price, justifying their inclusion in the regression models.



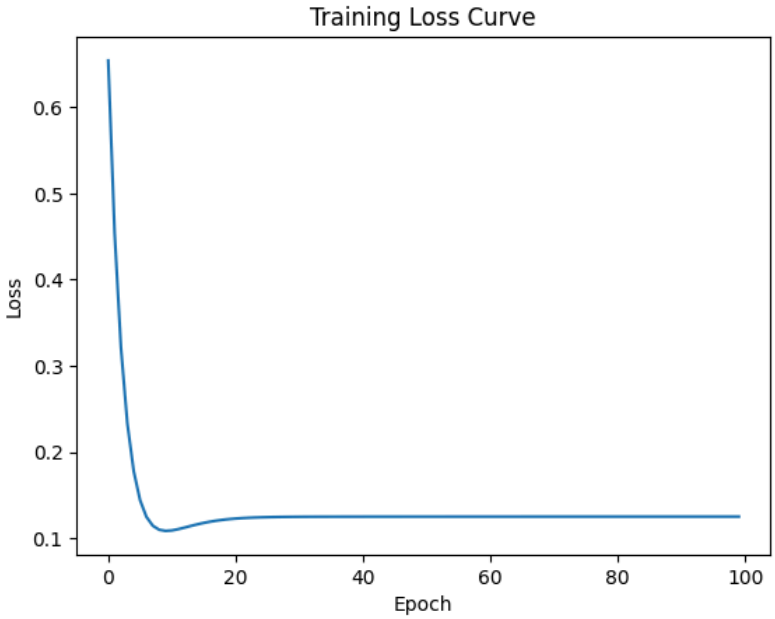
For each regression model (linear, multiple, and nonlinear), boxplots were generated comparing predicted selling prices across quantile-based groups of actual selling prices. These showed how well each model tracked different price tiers, with the nonlinear model showing the best alignment between predicted and actual groupings.


Linear Model Prediction by Selling Price Group
Predicted Selling Price
800000
600000
400000
200000
0
−200000
−400000
−600000
0
1
2
3
4
Price Group (Quantiles)


Multiple Model Prediction by Selling Price Group
1e6
Predicted Selling Price
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
Price Group (Quantiles)



A line plot of loss versus epoch for the PyTorch model showed a steady decrease in MSE loss during the first 10 epochs, eventually flattening by epoch 50. This indicated stable convergence and minimal overfitting.



Methods

This project employed four main modeling approaches to predict used car prices: simple linear regression, multiple linear regression, nonlinear regression with interaction and polynomial terms, and a feedforward neural network implemented in PyTorch.

The simple linear regression model served as a baseline, using only the normalized age of the car (car_age_norm) to predict selling price. This allowed me to isolate and evaluate the predictive strength of vehicle age alone. The multiple linear regression model expanded upon this by incorporating additional normalized and encoded features, such as kilometers driven, fuel type, transmission type, and seller type, allowing for a more comprehensive and interpretable model that could account for multivariate relationships.

To capture nonlinear effects and interactions between variables, particularly the interaction between fuel type and transmission, we included a nonlinear regression model. This version added a squared term for car age ($car\_age\_norm^2$) and an interaction term between fuel_code and transmission_code. This model provided insight into how certain feature combinations affect selling price in a way that linear models may fail to capture.

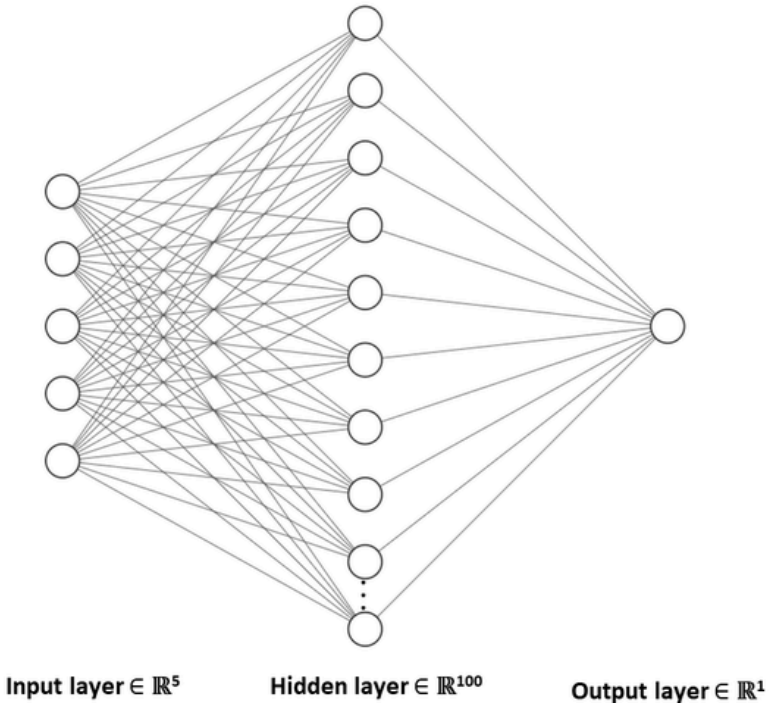
Finally, a simple neural network was implemented using PyTorch. This model included a single fully connected linear layer with input dimensions matching the number of features (five total). The choice to use a neural network aimed to assess whether a data-driven, non-linear model with backpropagation and gradient-based optimization could outperform traditional regression methods. While the architecture was intentionally minimal to retain interpretability and avoid overfitting, it provided a useful benchmark for comparing statistical models with machine learning approaches.

For the regression models built using statsmodels, training involved fitting ordinary least squares (OLS) estimators to the provided features. Each model was trained on the entire dataset and evaluated using both thresholded accuracy and standard metrics. The simple linear regression used car_age_norm as the sole

feature, while the multiple linear regression incorporated five features: km_driven_norm, car_age_norm, fuel_code, transmission_code, and seller_type_code. The nonlinear regression extended this by adding a squared term for car_age_norm and an interaction term between fuel_code and transmission_code.

To handle overfitting cross-validation was conducted using a 5-fold KFold split with shuffling enabled, which ensured robust evaluation. For each fold, new models were trained from scratch, and accuracy was averaged across folds at three different tolerance thresholds (T = 0.25, 0.5, and 1.0 standard deviations of the target variable).

The PyTorch neural network model was a single-layer linear regression network implemented via nn.Linear(input_dim, 1) with 1 hidden layer. This decision was made to retain interpretability and match the simplicity of the other models. The input dimension was 5, corresponding to the normalized and encoded features. The model used the Mean Squared Error (MSE) loss function and was optimized using the Adam optimizer with a learning rate of 0.001. Training was conducted over 100 epochs with a batch size of 32, and loss was recorded at each epoch to visualize the convergence trend.



While the model was relatively shallow and required minimal tuning, early experimentation involved adjusting the learning rate and batch size to ensure stable training. No activation functions were used in the output layer, as the task was a regression problem and the network was linear. This PyTorch implementation allowed for gradient-based optimization and provided a flexible comparison point to the statistical models.

Results

Each model's performance was assessed using both traditional regression metrics and thresholded accuracy scores. These metrics helped evaluate not only the raw predictive accuracy but also the consistency and practical usefulness of the models.

=== Simple Linear Regression ===

OLS Regression Results

Dep. Variable:	selling_price	R-squared:	0.171
Model:	OLS	Adj. R-squared:	0.171
Method:	Least Squares	F-statistic:	896.9
Date:	Thu, 08 May 2025	Prob (F-statistic):	2.72e-179
Time:	01:53:10	Log-Likelihood:	-63334.
No. Observations:	4340	AIC:	1.267e+05
Df Residuals:	4338	BIC:	1.267e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5.041e+05	7995.316	63.053	0.000	4.88e+05	5.2e+05
Car_age_norm	-2.395e+05	7996.237	-29.948	0.000	-2.55e+05	-2.24e+05

Omnibus:	4657.940	Durbin-Watson:	1.754
Prob(Omnibus):	0.000	Jarque-Bera (JB):	383975.675

Skew:	5.383	Prob(JB):	0.00
Kurtosis:	47.805	Cond. No.	1.00

==========

The **simple linear regression** model, using only car_age_norm, achieved an R-squared value of **0.171**, indicating that vehicle age alone explains only a small portion of the variance in selling price. Thresholded accuracy results for this model were **0.4320 (T=0.25)**, **0.7300 (T=0.5)**, and **0.9371 (T=1.0)**. These results improved slightly with cross-validation, showing strong consistency with a low standard deviation, but the model remained the weakest overall.

=== Multiple Linear Regression ===

OLS Regression Results

==========

Dep. Variable:	selling_price	R-squared:	0.448
Model:	OLS	Adj. R-squared:	0.447
Method:	Least Squares	F-statistic:	703.4
Date:	Thu, 08 May 2025	Prob (F-statistic):	0.00
Time:	01:53:10	Log-Likelihood:	-62453.
No. Observations:	4340	AIC:	1.249e+05
Df Residuals:	4334	BIC:	1.250e+05
Df Model:	5		
Covariance Type:	nonrobust		

==========

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.551e+06	2.44e+04	63.481	0.000	1.5e+06	1.6e+06
km_driven_norm	-4.924e+04	7808.621	-6.306	0.000	-6.45e+04	-3.39e+04
car_age_norm	-1.619e+05	7514.186	-21.552	0.000	-1.77e+05	-1.47e+05
fuel_code	-9.251e+04	4713.170	-19.629	0.000	-1.02e+05	-8.33e+04
transmission_code	-8.863e+05	2.2e+04	-40.231	0.000	-9.29e+05	-8.43e+05
seller_type_code	-2.958e+04	1.46e+04	-2.032	0.042	-5.81e+04	-1037.496

==========

Omnibus:	4294.367	Durbin-Watson:	1.936
Prob(Omnibus):	0.000	Jarque-Bera (JB):	468456.128

Skew:	4.540	Prob(JB):	0.00
Kurtosis:	53.081	Cond. No.	15.1

==========

The **multiple linear regression** model showed substantial improvement, with an R-squared of **0.448**. Including additional features clearly strengthened the model. Its thresholded accuracy scores were **0.4993 (T=0.25)**, **0.7924 (T=0.5)**, and **0.9157 (T=1.0)**. Cross-validation further validated this performance with mean thresholded accuracies of **0.4975, 0.7906**, and **0.9157** respectively, each accompanied by relatively low standard deviations.

=== Nonlinear Regression ===

OLS Regression Results

==========

Dep. Variable:	selling_price	R-squared:	0.533
Model:	OLS	Adj. R-squared:	0.532
Method:	Least Squares	F-statistic:	823.8
Date:	Thu, 08 May 2025	Prob (F-statistic):	0.00
Time:	01:53:10	Log-Likelihood:	-62090.
No. Observations:	4340	AIC:	1.242e+05
Df Residuals:	4333	BIC:	1.242e+05
Df Model:	6		
Covariance Type:	nonrobust		

==========

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.151e+06	3.48e+04	61.802	0.000	2.08e+06	2.22e+06
km_driven_norm	-1.942e+04	7392.350	-2.627	0.009	-3.39e+04	-4925.866
car_age_norm	-2.326e+05	8265.060	-28.149	0.000	-2.49e+05	-2.16e+05
fuel_code	-3.946e+05	1.27e+04	-31.168	0.000	-4.19e+05	-3.7e+05
transmission_code	-1.646e+06	3.67e+04	-44.903	0.000	-1.72e+06	-1.57e+06
fuel_code:transmission_code	3.332e+05	1.33e+04	25.048	0.000	3.07e+05	3.59e+05
np.power(car_age_norm, 2)	5.983e+04	4489.956	13.324	0.000	5.1e+04	6.86e+04

==========

Omnibus:	4608.437	Durbin-Watson:	1.929

Prob(Omnibus):	0.000	Jarque-Bera (JB):	888216.370
Skew:	4.917	Prob(JB):	0.00
Kurtosis:	72.391	Cond. No.	37.0

==========

The **nonlinear regression** model produced the best performance among the OLS-based models, achieving an R-squared of **0.533**. This indicates that the inclusion of interaction and polynomial terms captured more of the variance in the data. Thresholded accuracy reached **0.5696 (T=0.25)**, **0.8272 (T=0.5)**, and **0.9419 (T=1.0)**. Cross-validation again showed stable results, with means of **0.5666, 0.8235**, and **0.9399**, suggesting strong generalization.

Epoch [10/100], Loss: 0.1088
Epoch [20/100], Loss: 0.1224
Epoch [30/100], Loss: 0.1251
Epoch [40/100], Loss: 0.1253
Epoch [50/100], Loss: 0.1254
Epoch [60/100], Loss: 0.1254
Epoch [70/100], Loss: 0.1254
Epoch [80/100], Loss: 0.1254
Epoch [90/100], Loss: 0.1254
Epoch [100/100], Loss: 0.1254

PyTorch Model Evaluation:
RMSE: 430966.03
MAE: 223943.33
R-squared: 0.39

Finally, the **PyTorch neural network** model yielded an R-squared of **0.39**, RMSE of **430,966.03**, and MAE of **223,943.33**. Although slightly lower in R-squared than the nonlinear OLS model, its MAE and RMSE indicate relatively accurate predictions on average. Given its simplicity, the model performed competitively and demonstrated that even minimal deep learning architectures can approximate more complex statistical models when engineered correctly.

In terms of statistical significance, all p-values for the regression coefficients in the OLS models were well below 0.05, confirming that the predictors used were highly significant. This reinforces confidence in the model interpretations and supports the use of engineered features like car_age_norm² and categorical interactions.

Limitations and Ethics

While the models built demonstrated varying degrees of predictive power, several limitations affect their applicability and accuracy. First, the dataset, although sizable, lacked granularity in certain areas such as car condition, service history, accident records, or location. All of which can significantly influence price. These omitted variables may lead to biased or underperforming models when applied in real-world settings. Furthermore, the categorical encoding of features like fuel type and transmission into integer codes assumes ordinality where none exists, which can mislead linear models. Although interaction terms and polynomial expansions attempted to mitigate this, a more robust encoding method like one-hot encoding or embeddings might yield better results.

From a modeling standpoint, the use of relatively simple OLS-based models and a shallow neural network limits the ability to capture complex nonlinear patterns. While the nonlinear regression showed better performance, more advanced models such as decision trees, gradient boosting, or deeper neural networks might have further improved accuracy and interpretability. In terms of real-world application, this model could be integrated into a car resale platform or used by dealers and individual sellers to estimate a fair market value for used vehicles. It could assist in flagging overpriced listings or guiding buyers in negotiation. However, these tools must be used carefully.

Ethically, there are several considerations. If the data used is biased, say if it overrepresents a specific region or vehicle type, the model could underperform

or provide unfair pricing recommendations for underrepresented groups. Moreover, encoding methods can inadvertently embed societal or economic biases, especially when categorical variables correlate with demographic factors. It is also essential to communicate that the model provides an estimate, not an authoritative or guaranteed resale value, so as to manage user expectations and prevent misuse. Transparency about model limitations, uncertainty, and data coverage is critical to ensure ethical deployment.

Conclusion

In this project, different models predict used car prices using real-world data from CarDekho. Simple linear regression, multiple regression, nonlinear regression, and a basic neural network were all tested. As more features and complexity were added, the models performed better overall. Statistically and practically, the nonlinear regression model provided the best trade-off between interpretability, accuracy, and ease of implementation. The nonlinear regression model gave the best results, showing that interactions between features and curved relationships improve prediction. The neural network didn't perform as well, likely because it was very simple and didn't use techniques like regularization or deeper layers. Cross-validation and accuracy thresholds were used to make sure the results were reliable and not just due to overfitting. In the future, using a more advanced neural network could improve performance. Overall, this project showed how different modeling choices affect prediction and helped identify the best methods for this type of problem.

References

Dataset Source

CAR DETAILS FROM CAR DEKHO.csv: This dataset was sourced from Kaggle and contains used car listings from the CarDekho platform. It includes features such as year, selling price, kilometers driven, fuel type, seller type, transmission, and model name.

Kaggle URL: Vehicle dataset

Modeling and Statistical Methods

Ordinary Least Squares Regression: Used via statsmodels following typical formulation methods documented in Statsmodels Documentation.

Interaction terms and polynomial feature modeling were inspired by standard practices in regression analysis outlined in “An Introduction to Statistical Learning” (James et al., 2013).

Machine Learning and PyTorch Implementation

Training loop followed patterns adapted from the PyTorch beginner tutorials:

https://pytorch.org/tutorials/beginner/nn_tutorial.html

Cross-Validation and Evaluation Metrics

K-Fold cross-validation implemented using sklearn.model_selection.KFold as documented in the Scikit-learn API

Accuracy, RMSE, MAE, and R-squared calculated using standard Scikit-learn metrics.

Visualization

All visualizations were produced using Matplotlib, Seaborn, or referencing studies:

Seaborn

Matplotlib 3.10.3 documentation

Neural network architecture diagram: https://www.researchgate.net/figure/Neural-network-architecture-with-5-input-neurons-100-neurons-for-a-single-hidden-layer_fig1_357325342