Neural Network–Based Price Prediction: Data, Modeling, and Methodology

Pragyan Nandan Pandey

7th December, 2026

1 Introduction

Accurate price estimation in high-dimensional environments is a central problem in applied economics, data science, and computational decision-making. Traditional parametric approaches often struggle to accommodate nonlinearities, interaction effects, and heterogeneous responses across observations, particularly when the feature space is large and structurally complex.

This project develops a **purely neural network–based framework** for price prediction, emphasizing robustness, interpretability of modeling choices, and computational efficiency. The approach integrates disciplined data preparation with modern deep learning techniques, leveraging **pretraining and model combination** to improve generalization while reducing training instability.

Price formation processes are inherently nonlinear and are shaped by complex interactions between observable characteristics, latent structural factors, and location-specific heterogeneity. While neural networks offer powerful universal function approximation capabilities, naïve implementations often suffer from overfitting, slow convergence, and sensitivity to initialization. This project addresses these challenges through a structured empirical pipeline encompassing exploratory analysis, statistically motivated preprocessing, representation learning, and controlled fine-tuning.

2 Exploratory Data Analysis (EDA)

The exploratory data analysis (EDA) stage is designed to understand the structure, quality, and statistical properties of the dataset prior to modeling. The objective is to identify distributional irregularities, assess relationships among variables, and guide feature engineering and model specification.

2.1 Data Structure and Quality

The dataset consists of structured tabular observations comprising continuous numerical variables, categorical identifiers, and temporal fields. Initial inspection confirms the absence of duplicate records and reveals consistent dimensionality across observations. Missing values are sparse and appear random, with no systematic patterns across variables.

Temporal variables required format standardization before analysis. Once corrected, temporal ordering behaved consistently, enabling reliable downstream processing.

2.2 Univariate and Multivariate Patterns

Univariate analysis indicates that most numerical variables exhibit mild skewness, while a subset displays pronounced right skew. These findings motivate nonlinear transformations in the preprocessing stage. The target variable (price) is strictly positive and continuous, with no abnormal mass points or zero inflation.

Correlation analysis reveals limited multicollinearity among raw features. Strong correlations arise primarily among transformed versions of the same variables, reflecting deliberate feature engineering rather than structural redundancy.

2.3 Spatial and Temporal Diagnostics

Location-based identifiers do not exhibit strong standalone linear effects but suggest indirect interaction-driven influences. Temporal diagnostics show weak autocorrelation and no persistent trends or seasonality, supporting treatment of the data as pooled cross-sectional rather than time-series.

2.4 Structural Insights

Preliminary clustering diagnostics suggest the presence of localized, non-spherical groupings, indicating that density-based clustering methods may be more appropriate than global partitioning approaches.

3 Data Preprocessing

The preprocessing pipeline transforms raw inputs into a statistically coherent, model-ready dataset. Each step is motivated by econometric validity, numerical stability, and compatibility with neural network optimization.

3.1 Feature Selection and Transformation

Features with near-zero variance, redundant semantic meaning, or extreme collinearity were removed or consolidated. Highly skewed numerical variables were transformed using logarithmic or square-root transformations to stabilize variance while preserving monotonicity.

3.2 Scaling and Encoding

All continuous features were standardized to zero mean and unit variance after transformation. Nominal categorical variables were encoded using one-hot encoding, avoiding artificial ordering assumptions and preserving interpretability.

3.3 Temporal Handling and Outliers

Temporal variables were converted to numeric representations where necessary but were not modeled as dynamic drivers due to weak temporal dependence. Extreme observations were retained unless they violated logical constraints, ensuring preservation of economically meaningful tail behavior.

3.4 Final Feature Matrix

The final dataset consists exclusively of numeric, scale-consistent features free of missing values, suitable for gradient-based optimization and neural network training.

4 Modeling Framework

The modeling strategy relies exclusively on neural networks. Two complementary models are trained and combined to exploit both representation learning and task-specific calibration.

4.1 Model 1: Pretrained Neural Network

Let $\mathbf{x}_i \in \mathbb{R}^d$ denote the feature vector for observation i. The pretrained neural network learns a nonlinear mapping

$$\mathbf{h}_i = f_\theta(\mathbf{x}_i), \tag{1}$$

where $f_\theta(\cdot)$ is a deep feedforward network and $\mathbf{h}_i$ is a latent representation. Pretraining improves initialization quality, accelerates convergence, and enhances generalization by capturing generic nonlinear structure.

4.2 Model 2: Task-Specific Neural Network

A second neural network maps the learned representation to prices:

$$\hat{y}_i^{(2)} = g_\phi(\mathbf{h}_i), \tag{2}$$

where $g_\phi(\cdot)$ is a shallow network trained to calibrate price predictions. This separation of representation learning and estimation improves robustness and reduces variance.

4.3 Multi-Modal Neural Network Architecture

This project employs two closely related multi-modal neural network models that jointly exploit structured tabular data and visual information for price prediction. Both models integrate pretrained convolutional neural networks for image feature extraction with fully connected networks for tabular representation learning. The final price estimate is obtained through nonlinear fusion of these modalities.

4.3.1 Notation

Let:

- $\mathbf{x}_i \in \mathbb{R}^d$ denote the tabular feature vector for observation i
- $\mathbf{I}_i \in \mathbb{R}^{224\times224\times3}$ denote the associated image input
- $y_i \in \mathbb{R}^+$ denote the observed price
- $\hat{y}_i$ denote the predicted price

—

4.3.2 Image Encoder (Pretrained CNN)

Both models use a pretrained EfficientNet architecture as a frozen feature extractor. Let $\Phi(\cdot)$ denote the pretrained convolutional mapping.

$$\mathbf{z}_i^{\text{img}} = \Phi(\mathbf{I}_i) \tag{3}$$

where:

$$\mathbf{z}_i^{\text{img}} \in \mathbb{R}^k, \quad k = \begin{cases} 256 & \text{(EfficientNetB3, Model 1)} \\ 1280 & \text{(EfficientNetB0, Model 2)} \end{cases}$$

The CNN parameters are frozen, ensuring:

- Reduced training time

- Improved generalization
- Stable feature representations

—

4.3.3 Tabular Encoder

The tabular encoder maps structured features into a dense latent space using fully connected layers with nonlinear activations.

$$\mathbf{z}_i^{\text{tab}} = f_\theta(\mathbf{x}_i) \tag{4}$$

where $f_\theta(\cdot)$ is defined as:

$$\mathbf{h}_i^{(1)} = \sigma(W_1\mathbf{x}_i + \mathbf{b}_1) \tag{5}$$
$$\mathbf{h}_i^{(2)} = \sigma(W_2\mathbf{h}_i^{(1)} + \mathbf{b}_2) \tag{6}$$
$$\mathbf{z}_i^{\text{tab}} = \mathbf{h}_i^{(2)} \tag{7}$$

with $\sigma(\cdot)$ denoting the ReLU activation function and dropout applied during training to control overfitting.

—

4.3.4 Multi-Modal Fusion

The latent representations from both modalities are concatenated to form a joint feature vector:

$$\mathbf{z}_i^{\text{fusion}} = \begin{bmatrix} \mathbf{z}_i^{\text{img}} \\ \mathbf{z}_i^{\text{tab}} \end{bmatrix} \in \mathbb{R}^{k+m} \tag{8}$$

This fused representation captures complementary visual and structural information relevant for price formation.

—

4.3.5 Prediction Head

The final prediction is produced via a fully connected regression head:

$$\mathbf{h}_i^{(3)} = \sigma(W_3\mathbf{z}_i^{\text{fusion}} + \mathbf{b}_3) \tag{9}$$
$$\mathbf{h}_i^{(4)} = \sigma(W_4\mathbf{h}_i^{(3)} + \mathbf{b}_4) \tag{10}$$
$$\hat{y}_i = W_5\mathbf{h}_i^{(4)} + b_5 \tag{11}$$

Batch normalization and dropout are applied between layers to stabilize training and improve out-of-sample robustness.

—

4.3.6 Loss Function

Both models are trained using the Mean Squared Error (MSE) loss:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{12}$$

MSE is appropriate for continuous price prediction and penalizes large deviations disproportionately.

—

4.3.7 Evaluation Metrics

Model performance is evaluated using:

Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{13}$$

Mean Absolute Percentage Error (MAPE)

$$\text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{14}$$

—

4.3.8 Optimization

All trainable parameters are optimized using the Adam optimizer:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta \mathcal{L} \tag{15}$$

with learning rate $\eta = 10^{-4}$ and early stopping based on validation loss.

—

Huber Loss The Huber loss combines the robustness of absolute loss with the smoothness of squared loss. For a prediction $\hat{y}_i$ and true value y_i, it is defined as:

$$\mathcal{L}_\delta(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right), & \text{otherwise} \end{cases} \tag{16}$$

where $\delta > 0$ is a threshold parameter controlling the transition between quadratic and linear loss regimes.

$$\mathcal{L}_{\text{Huber}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_\delta(y_i, \hat{y}_i) \tag{17}$$

The Huber loss is less sensitive to outliers than mean squared error while remaining differentiable, making it well-suited for price prediction tasks with heavy-tailed error distributions.

4.3.9 Model Interpretation

The architecture separates:

- **Representation learning** (via pretrained CNNs)
- **Contextual calibration** (via tabular encoders)

This modular design improves stability, interpretability, and computational efficiency while capturing complex nonlinear price determinants.

4.4 Loss Function and Optimization

Both models are trained using mean squared error (MSE):

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{18}$$

Optimization is performed using the Adam algorithm with learning-rate scheduling and early stopping to prevent overfitting.

4.5 Model Combination

Final price predictions are obtained by combining outputs from both models:

$$\hat{y}_i = \alpha\hat{y}_i^{(1)} + (1-\alpha)\hat{y}_i^{(2)}, \quad \alpha \in [0,1], \tag{19}$$

where α is selected based on validation performance. This weighted combination balances global nonlinear structure with local calibration accuracy.

5 Conclusion

The proposed framework integrates rigorous data diagnostics, disciplined preprocessing, and a carefully structured neural network modeling strategy. By leveraging pretraining and model combination, the approach achieves improved predictive accuracy, reduced training time, and enhanced robustness. While applied to price prediction, the methodology is broadly applicable to high-dimensional regression problems in applied economics and data science.