

# HOUSE PRICE PREDICTION — ANN

Kummithi Adhi Keshava Reddy

2023BCS0169

## 1. Introduction

The real estate market is influenced by many variables: property area, condition, age of the house, neighborhood, structural quality, amenities — all of which may non-linearly affect the sale price.

The Kaggle “House Prices: Advanced Regression Techniques” dataset provides a large and detailed record of residential homes with ~79 explanatory variables capturing many aspects of a home and its surroundings.

In this project, we aim to build an Artificial Neural Network (ANN) — a regression model — to predict house sale prices using this dataset. The model’s goal is to learn complex, non-linear relationships among features and output a reliable price estimate.

## 2. Dataset Description & Problem Statement

- The dataset consists of two CSV files: **train.csv** (with sale prices) and **test.csv** (without sale prices, for prediction).
- The training set contains **1460** houses, with **81 columns** (features + SalePrice + Id). The test set contains around **1459** houses.
- There are **79 explanatory (input) variables** describing aspects such as: living area (GrLivArea), basement size (TotalBsmtSF), garage capacity (GarageCars / GarageArea), number of rooms above ground (TotRmsAbvGrd), year built (YearBuilt), house quality metrics (OverallQual, OverallCond), and many categorical variables (neighborhood, type of construction, exterior material, roof style, condition, etc.) that capture structural and locational properties.
- The problem: Given these features (for a house), predict the SalePrice, i.e. the final sale price in USD. This is a regression task.

### Why this dataset is good:

- Large number of features — captures many facets of a house (size, condition, structural attributes, neighborhood, etc.), which allows deep models (like ANN) to leverage non-linearity.
- Realistic dataset — reflects real-world housing data complexity: numerical + categorical data, missing values, varied distributions.

- Opportunity for feature engineering: combining, transforming, encoding, handling missing values, outliers, etc. to improve model performance.

## 3. Objectives

- Load and explore the Kaggle dataset to understand feature distributions and relationships with sale price.
- Preprocess data: handle missing values, encode categorical variables, scale numerical features, manage outliers.
- Engineer useful features (e.g. total living area combining basement + above ground, total bathrooms, quality indicators, etc.) to improve predictive power.
- Build an ANN regression model to predict house prices, using the cleaned and engineered data.
- Evaluate the model with appropriate regression metrics and validate performance (train/validation split or cross-validation).
- (Optional but desirable) Compare ANN performance with baseline models (e.g. linear regression, random forest, gradient boosting) to gauge relative effectiveness.
- Prepare final predictions for unseen test data.

## 4. Methodology & Data Preprocessing

This project follows a structured data-preprocessing pipeline implemented fully in the Jupyter Notebook. The Kaggle *House Prices: Advanced Regression Techniques* dataset contains a mix of numerical and categorical features, requiring careful cleaning, transformation, and preparation before training the ANN model.

### 4.1 Importing Libraries

The notebook begins by importing essential Python libraries for data handling, numerical operations, visualization, preprocessing, and neural network modeling. These include libraries such as Pandas, NumPy, visualization tools, Scikit-learn preprocessing utilities, and TensorFlow/Keras for ANN development.

### 4.2 Loading the Dataset

Both the training and test datasets provided by Kaggle were loaded into the notebook. The training dataset contains features along with the target variable *SalePrice*, while the test dataset contains only features.

The *Id* column, which serves only as an identifier, was removed since it does not contribute to the prediction model.

## 4.3 Exploratory Data Analysis (EDA)

An initial exploration of the dataset was performed to understand patterns, trends, and relationships.

Key steps included:

- Examining data types, missing values, and summary statistics
- Visualizing feature distributions to identify skewness
- Identifying highly correlated numerical features with *SalePrice*
- Observing strong relationships with important variables such as overall material quality, above-ground living area, basement area, garage capacity, and year built
- Detecting outliers that may negatively affect model training

This analysis helped determine which features required transformation or special handling.

## 4.4 Handling Missing Values

The dataset contains missing values across several numerical and categorical features.

To maintain data integrity, the following methods were applied:

### Numerical Features

Missing values in numerical features were filled using appropriate statistical measures such as the median. This prevents distortion compared to mean imputation when distributions are skewed.

### Categorical Features

Categorical features with missing values were assigned meaningful placeholder categories such as “None” or “No”, especially when absence itself represents useful information (e.g., no garage, no basement, no fence).

### Columns with Too Many Missing Values

Some features had excessively high percentages of missing data and were removed from the dataset to avoid introducing noise. Examples include pool-related features or rarely observed amenities.

### Target Variable Transformation

Since the distribution of *SalePrice* is positively skewed, a logarithmic transformation was applied to smooth the distribution, improving the performance of regression models.

## 4.5 Encoding Categorical Variables

The dataset contains many categorical features describing neighborhood, construction material, heating type, roof style, etc.

These features were converted into numerical values using encoding techniques so that the neural network could interpret them. The most commonly used technique was One-Hot Encoding, which converts each category into separate binary columns.

## 4.6 Feature Scaling

As neural networks are sensitive to the scale of input values, all numerical features were standardized.

Scaling ensures that features such as square footage, age, and lot area contribute equally during model training and prevents larger-valued attributes from dominating the learning process.

## 4.7 Splitting the Dataset

The cleaned and transformed dataset was divided into two parts:

- A training set for learning patterns
- A validation/test set for evaluating model performance

This ensures that the model is evaluated on unseen data and helps detect overfitting.

## 4.8 Outlier Detection and Removal

A small number of extreme outliers were identified, particularly large properties with unusually low sale prices.

These were removed to prevent the neural network from learning misleading patterns and to improve model stability.

## 4.9 Final Prepared Dataset

After completing all preprocessing steps, the final dataset consisted of:

- No missing values
- Fully encoded categorical variables
- Scaled numerical fields
- A balanced and cleaned feature set suitable for ANN training

- A transformed target variable representing the log of the original *SalePrice*

The dataset was then ready for use in building and training the Artificial Neural Network model.

## 5. ANN Model Architecture & Training

- Use an Artificial Neural Network (ANN) suitable for regression. Input layer size = number of features after preprocessing.
- Hidden layers: 1–3 dense layers with ReLU activation (or other as required) — allow model to capture non-linear relationships.
- Output layer: single neuron, linear activation — predicts continuous sale price.
- Loss function: Mean Squared Error (MSE) (or Mean Absolute Error, depending on preference).
- Optimizer: for example Adam.
- Training hyperparameters: epochs (e.g. 50–200), batch\_size (e.g. 32), with early stopping on validation loss to avoid overfitting.
- Optional: regularization (dropout, weight decay), batch normalization, learning rate tuning, etc.

## 6. Model Evaluation

Evaluate model using metrics such as:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE) / Root Mean Squared Error (RMSE)
- R<sup>2</sup> Score (coefficient of determination) to know how well the variance of sale prices is explained by the model

Optionally, compare with baseline models (linear regression, decision tree, random forest, gradient boosting) to see how much gain the ANN gives. Many people use such baseline models on this dataset.

Also perform cross-validation to get more stable performance estimates.

## 7. Results & Discussion

Explain what you observed: e.g.

- How well the ANN performed on validation data (with MAE, MSE, R<sup>2</sup>).
- Which features were most influential (if you applied methods to check feature importance / perform feature selection).

- Whether data preprocessing and feature engineering helped — e.g. did log-transform or aggregate features improve results.
- Compare with baseline models: did ANN outperform simpler models? By how much?
- Challenges: missing data, categorical variables, outliers — and how you addressed them.
- Limitations: dataset is from Ames, Iowa; distribution may not generalize to other cities/countries; model may overfit to local structural/market patterns.

## 8. Conclusion

Summarize that using the rich Kaggle dataset and proper preprocessing + feature engineering + ANN modeling, one can build an effective house-price prediction system. With careful handling of missing data, outliers, and categorical features, the model can generalize reasonably well. This work demonstrates the power of neural networks for regression on complex real-world datasets.

## 9. References

- Kaggle — House Prices: Advanced Regression Techniques dataset.
- Guide / tutorials for dataset and modeling on Kaggle / DataScience portfolios.
- Studies/projects by other practitioners using feature engineering, ensemble, baseline and neural networks on this dataset.