House Price Prediction Using Machine Learning

Author: Imasha Buddhini

Tools: Python, Pandas, NumPy, Scikit-Learn, Google Colab

Dataset: Ames Housing Dataset (Kaggle)

Introduction

Predicting house prices is a critical task in the real estate industry, helping buyers, sellers, and investors make informed decisions. With the availability of large structured datasets, machine learning techniques can be used to identify patterns and relationships that traditional methods may overlook.

Objectives

To build a machine learning model that predicts house sale prices based on various property features.

This project focuses on proper data preprocessing, feature engineering, model comparison, and ensemble learning to improve prediction accuracy.

Dataset Description

The dataset used in this project is the **Ames Housing Dataset**, obtained from Kaggle. It contains detailed information about residential properties, including structural attributes, location-based features, and quality indicators.

- Training samples: 1460 houses
- Test samples: 1459 houses
- Total features: 79 explanatory variables
- Target variable: SalePrice

The dataset includes both numerical and categorical features, with several columns containing missing values.

Data Preprocessing

The dataset contained missing values across both numerical and categorical features. Instead of applying a single imputation, missing values were handled based on feature meaning, data distribution, and real-world interpretation.

Numerical Features:

i) Neighborhood-Based Median Imputation (LotFrontage)

The feature LotFrontage, which represents the linear feet of street connected to the property, had a significant number of missing values. Since lot frontage strongly depends on neighborhood layout and planning, replacing missing values with the overall median would ignore local variations.

To address this, missing LotFrontage values were filled using the median frontage value within each neighborhood:

- Houses in the same neighborhood tend to have similar lot structures
- This avoids distortion caused by outliers or extreme values from other areas

ii) Median Imputation for Other Numerical Features

For other numerical variables where missing values did not indicate absence (e.g. garage year built), missing values were filled using the **median** of the feature.

iii) Zero Imputation for Absence-Based Features

Certain numerical features had missing values that explicitly represented non-existence, not unknown values. (Basement area, Basement bathrooms etc)

In such cases, missing values were filled with **0**, as this correctly reflects the real-world meaning:

This prevents the model from misinterpreting absence as an average or typical value.

Categorical Features:

- Filled with "None" when missing represented non-existence
- One-hot encoding was applied using pd.get_dummies()

Exploratory Data Analysis (EDA)

Before building models, exploratory data analysis was performed to understand the structure and quality of the data.

Key Observations:

- ✓ The target variable SalePrice was **right-skewed**, which could negatively impact model performance.
- ✓ Strong correlations were observed between house price and features related to area, quality, and overall size.

To reduce skewness, a log transformation was applied to the target variable.

Feature Engineering

To improve model performance, new meaningful features were created.

i) Total Square Footage

TotalSF = TotalBsmtSF + 1stFlrSF + 2ndFlrSF

ii) Total Bathrooms

TotalBath = FullBath + 0.5 × HalfBath + BsmtFullBath + 0.5 × BsmtHalfBath

These engineered features better capture the overall size and usability of the house compared to individual columns.

Low-informative were removed after encoding to reduce noise and dimensionality.

Model Selection and Training

Three regression models were trained and evaluated.

Models Used

- Ridge Regression (Linear Regression + regularization)
- Random Forest Regressor
- Gradient Boosting Regressor

Cross-validation was chosen to ensure robust evaluation and avoid overfitting.

Hyperparameter Tuning:

The Gradient Boosting Regressor showed the strongest baseline performance. Therefore, it was further optimized using tuned hyperparameters such as:

- Number of estimators
- Learning rate
- Tree depth
- Subsampling

After tuning, the model achieved a significantly improved RMSE score.

Ensemble Learning:

To further improve prediction accuracy, an ensemble model was created by combining predictions from all three models using weighted averaging.

Final Prediction = 0.2 × Linear Regression + 0.3 × Random Forest + 0.5 × Gradient Boosting

Conclusion

This project demonstrates a complete end-to-end machine learning workflow, from data cleaning to model deployment. Feature engineering and ensemble learning played a crucial role in improving prediction accuracy.

Key Takeaways

- Proper handling of missing values is critical
- Feature engineering can significantly boost model performance
- Ensemble models often outperform individual models