Report
Machine Learning
(AIML202)

In partial fulfillment ofthe requirements for the award ofthe degree
of
Bachelors ofTechnology in
Computer Science and Engineering(Data Science)

By
ShubhRawat
A023109522008

Under the guidance of
Dr. Dhruv Sharma

Department ofComputer Science & Engineering
AMITY SCHOOLOFENGINEERING & TECHNOLOGY
AMITY UNIVERSITY UTTAR PRADESH, NOIDA (U.P.)

Table of Content

Abstract

The House Price Prediction Model is a machine learning-based approach designed to estimate real estate values using structured property data. Traditional valuation methods, such as comparative market analysis, rely on subjective assessments and limited data. In contrast, this project employs HistGradientBoostingRegressor, a state-of-the-art gradient boosting technique, to enhance prediction accuracy.

The model is trained on the House Prices: Advanced Regression Techniques dataset, which includes a mix of numerical and categorical features influencing house prices. A structured data preprocessing pipeline was implemented, including missing value imputation, feature selection, categorical encoding, and outlier removal, ensuring the dataset was optimized for machine learning.

HistGradientBoostingRegressor was selected for its speed, efficiency, and ability to handle missing values. The model was evaluated using Root Mean Squared Error (RMSE) and R^2 Score, demonstrating its predictive strength in estimating house prices. Cross-validation further ensured the model's robustness and reliability.

The results indicate that gradient boosting effectively captures complex feature interactions in real estate pricing. This project contributes to automated property valuation, providing a data-driven alternative to traditional appraisal methods. Future improvements include hyperparameter tuning, integration of economic indicators, and deployment as a web-based tool for real-world applications.

1. Introduction

1.1 Objective of the Project

The real estate market is complex and dynamic, with house prices being influenced by multiple factors such as location, house size, year built, quality of materials, and neighborhood trends. Traditional methods for estimating house prices rely on manual assessments by real estate agents, which can be subjective and inconsistent.

The objective of this project is to develop a data-driven machine learning model using HistGradientBoostingRegressor, which can:

- Predict house prices based on given property features.
- Reduce manual errors in property valuation.
- Help buyers and sellers make informed decisions. ● Analyze market trends based on historical data.

1.2 Importance of House Price Prediction

A reliable house price prediction system has several real-world applications:

- For Home Buyers: Helps in determining whether a property is fairly priced.
- For Home Sellers: Assists in setting an optimal selling price.
- For Real Estate Investors: Aids in making profitable investment decisions.
  - For Banks & Mortgage Lenders: Supports risk assessment for home loans.

1.3 Machine Learning in Real Estate

Traditional valuation models depend on comparative market analysis (CMA), which involves comparing a property to similar recently sold homes. However, this approach has limitations:

- Limited Data Scope: CMA considers only a few nearby properties.
- Human Bias: Agents may overvalue/undervalue homes.
- Market Fluctuations: Cannot adapt to rapid price changes.
- Machine learning models, on the other hand, analyze thousands of data points, learn patterns from past sales, and predict future prices more accurately.

2. Dataset Overview

2.1 Data Source and Description

The dataset used in this project is from Kaggle's House Prices: Advanced Regression Techniques competition. It consists of numerical and categorical variables that describe a property's characteristics and its final sale price.

2.2 Features of the Dataset

The dataset includes 79 explanatory variables, which can be grouped as:

2.2.1 Numerical Features (Continuous Variables)

These are measurable quantities that directly impact house prices:

- LotArea: Total land area of the property (square feet).
- YearBuilt: Construction year of the house.
- GrLivArea: Above-ground living area in square feet.
- TotalBsmtSF: Basement area in square feet.
- OverallQual: Overall material and finish quality (Scale 1-10).

In [9]:
```
df.head(6)
```

Out[9]:

LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	B
65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	
80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	
68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	
60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	
84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	
85.0	14115	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	Norm	Norm	

2.2.2 Categorical Features (Discrete Variables)

These are non-numeric attributes describing house characteristics:

- Neighborhood: The location of the house (e.g., Downtown, Suburban).
- HouseStyle: Architectural style (e.g., 1-story, 2-story).
- RoofStyle: Type of roof (e.g., Gable, Hip).
- Heating: Type of heating system (e.g., Gas, Electric).

2.2.3 Target Variable (Dependent Variable)

The feature that we are trying to predict:

SalePrice – The final selling price of the house (USD).

3. Data Preprocessing

Before training the machine learning model, data preprocessing is necessary to clean and transform the dataset.

3.1 Handling Missing Values

Missing values can cause errors during training. Our approach to handle them:

- Drop columns with more than 20% missing values.
- Mean Imputation for numerical features (replacing missing values with the column mean).
- Mode Imputation for categorical features (replacing missing values with the most common category).

```
In [19]: col_for_drop = null_percent[null_percent > 20].keys() # if the null value % 20 or > 20 so need to drop it
```

```
In [20]: df = df.drop(col_for_drop, axis=1)
```

```
df['Electrical'] = df['Electrical'].fillna(df['Electrical'].mode()[0])
df['Exterior1st'] = df['Exterior1st'].fillna(df['Exterior1st'].mode()[0])
df['Exterior2nd'] = df['Exterior2nd'].fillna(df['Exterior2nd'].mode()[0])
df['Functional'] = df['Functional'].fillna(df['Functional'].mode()[0])
df['KitchenQual'] = df['KitchenQual'].fillna(df['KitchenQual'].mode()[0])
df['MSZoning'] = df['MSZoning'].fillna(df['MSZoning'].mode()[0])
df['SaleType'] = df['SaleType'].fillna(df['SaleType'].mode()[0])
df['Utilities'] = df['Utilities'].fillna(df['Utilities'].mode()[0])

df.columns = df.columns.str.strip()
```

3.2 Feature Selection

Feature selection helps improve model efficiency by removing irrelevant variables.

We used correlation analysis to select features with a correlation of ≥ 0.5 or ≤ 0.5 with SalePrice.

```
In [23]: # Describe the target
         train["SalePrice"].describe()
```

```
Out[23]: count      1460.000000
         mean     180921.195890
         std       79442.502883
         min       34900.000000
         25%      129975.000000
         50%      163000.000000
         75%      214000.000000
         max      755000.000000
         Name: SalePrice, dtype: float64
```

3.3 Encoding Categorical Variables

Machine learning models cannot work directly with text-based categorical data.

We used One-Hot Encoding to convert categorical features into numerical form.

3.4 Outlier Detection and Removal

Outliers were detected using boxplots and removed to prevent bias in predictions.

4. Model Selection and Justification

4.1 What is Gradient Boosting?

Gradient Boosting is a powerful ensemble learning technique that combines multiple weak models (decision trees) into a strong predictive model.

4.2 Why Use HistGradientBoostingRegressor?

- Optimized for Speed: Faster than traditional Gradient Boosting methods.
- Handles Missing Values: Works even when some data is missing.
- Better Generalization: Reduces overfitting compared to regular Gradient Boosting.

In [94]:
```
from sklearn.experimental import enable_hist_gradient_boosting  # Enables HGBR
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.impute import SimpleImputer
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, r2_score

# Step 1: Impute missing values in X_train
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)

# Step 2: Initialize the model
hbgr = HistGradientBoostingRegressor(random_state=42)

# Step 3: Cross-validation to calculate R² scores on the training set
r2 = make_scorer(r2_score)
r2_scores = cross_val_score(hbgr, X_train_imputed, y_train, cv=5, scoring=r2)  # 5-fold cross-validation

# Step 4: Calculate average R² score
average_r2 = r2_scores.mean()
print(f"Average R² score (training accuracy) from cross-validation: {average_r2}")
```

```
Average R² score (training accuracy) from cross-validation: 0.889444799471551
```

4.3 Model Implementation

1. Splitting Data:

   - 80% Training Set
   - 20% Test Set

2. Feature Imputation: Missing values were filled with SimpleImputer(strategy='mean').

3. Mode Initialization: HistGradientBoostingRegressor(random_state=42) was chosen.
4. Training: The model was trained using X_train_imputed and y_train.

```
In [93]:
from sklearn.experimental import enable_hist_gradient_boosting  # Enables HGBR
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error
import numpy as np

# Assuming `train_len` is defined and corresponds to the length of the training set
# X_train and X_test are parts of the dataset based on `train_len`
X_train = df[:train_len]  # Features for training
X_test = df[train_len:]   # Features for testing
y_train = SalePrice       # Target variable for training

# Print shapes of the data
print(f"Training set shape: {X_train.shape}")
print(f"Test set shape: {X_test.shape}")
print(f"Length of y_train: {len(y_train)}")

# Step 1: Handle missing values using SimpleImputer
imputer = SimpleImputer(strategy='mean')  # You can change the strategy if needed
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Step 2: Initialize the model
hbgr = HistGradientBoostingRegressor(random_state=42)

# Step 3: Train the model on the training data
hbgr.fit(X_train_imputed, y_train)

# Step 4: Make predictions on the test data
y_pred = hbgr.predict(X_test_imputed)

# Step 5: Evaluate the model (using the training set for this example)
train_predictions = hbgr.predict(X_train_imputed)
mse = mean_squared_error(y_train, train_predictions)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE) on the training set: {rmse}")

# You can use the predictions `y_pred` to evaluate on actual test data if available
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/experimental/enable_hist_gradient_boosting.py:15: UserWarning: Since vers
ion 1.0, it is not needed to import enable_hist_gradient_boosting anymore. HistGradientBoostingClassifier and HistGradientBoo
stingRegressor are now stable and can be normally imported from sklearn.ensemble.
  warnings.warn(
Training set shape: (1460, 497)
Test set shape: (1459, 497)
Length of y_train: 1460
Root Mean Squared Error (RMSE) on the training set: 0.04788436790159733
```

5. Model Evaluation

5.1 Root Mean Squared Error (RMSE)

- RMSE quantifies the error between predicted and actual house prices.
  - Formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{actual} - y_{predicted})^2}$$

- Lower RMSE = better model performance.

In [97]:
```python
from sklearn.experimental import enable_hist_gradient_boosting  # Enables HGBR
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.impute import SimpleImputer
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import make_scorer, r2_score

# Define the model
hbgr = HistGradientBoostingRegressor()

# Impute missing values in X_train
imputer = SimpleImputer(strategy='mean')  # You can change the strategy to 'median', 'most_frequent', etc.
X_train_imputed = imputer.fit_transform(X_train)

# Now run cross-validation
cv = KFold(n_splits=3, shuffle=True, random_state=45)
r2 = make_scorer(r2_score)
r2_val_score = cross_val_score(hbgr, X_train_imputed, y_train, cv=cv, scoring=r2)

# Check mean R^2 score
score = r2_val_score.mean()
print(f"Mean R^2 score from cross-validation: {score}")
```

```
Mean R^2 score from cross-validation: 0.8765418060323135
```

5.2 R^2 Score (Coefficient of Determination)

- Measures how well the independent variables explain the variance in house prices.
- Formula:

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}}$$

- Higher R^2 = better model accuracy.

```
In [99]: from sklearn.ensemble import HistGradientBoostingRegressor

# Initialize the model
HGBR = HistGradientBoostingRegressor()

# Perform cross-validation without imputing NaN values
cross_validation = cross_val_score(estimator=HGBR, X=X_train, y=y_train, cv=10)

print("Cross validation accuracy of HGBR model = ", cross_validation)
print("Cross validation mean accuracy of HGBR model = ", cross_validation.mean())

Cross validation accuracy of HGBR model =  [0.88066348 0.93256081 0.91381916 0.84375237 0.88257827 0.90430409
 0.88122108 0.92427662 0.89348091 0.88144689]
Cross validation mean accuracy of HGBR model =  0.8938103668353966
```

6. Results and Discussion

- The model performed well on training data, achieving a low RMSE and a high R^2 score.
- Feature selection improved model accuracy by removing irrelevant attributes.
- Gradient boosting helped capture complex relationships between house features and price.
-

7. Conclusion and Future Work

7.1 Key Takeaways

- HistGradientBoostingRegressor is effective for house price prediction.
- Feature selection played a crucial role in improving accuracy.
- The model can be used in real estate applications for property valuation.

7.2 Future Enhancements

- Hyperparameter tuning to optimize model parameters.
- External market factors (e.g., interest rates) could be included. ● Exploring Deep Learning models for better predictions.

8. References

1. Scikit-Learn Documentation: https://scikit-learn.org/
2. Kaggle House Prices Dataset: https://www.kaggle.com/c/house-pricesadvanced-regression-techniques