Report on House Price Prediction

CSE PATHSHALA

PROJECT TRAINING REPORT

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology (Computer Science and Engineering – Specialised in Artificial Intelligence and Machine Learning)

Submitted to: Lovely Professional University , Phagwara , Punjab

From 10/06/2025 to 28/07/2025

Submitted By

Name of Student : Mannat Jain

Registration Number: 12310235

Signature of the student:

Mannat.

Scanned with CamScanner

DECLARATION

I, Mannat Jain(12310235) , hereby declare that the work done by me on project titled "**House Price Prediction**" from 10th June 2025 to 28Th July 2025 as part of the summer training program "**Advanced Python for ML and AI**" organized by **CSE PATHSHALA** is a record of my original work. This project was carried out under the guidance of our instructors and has not been submitted to any other institution for the award of any degree or diploma.

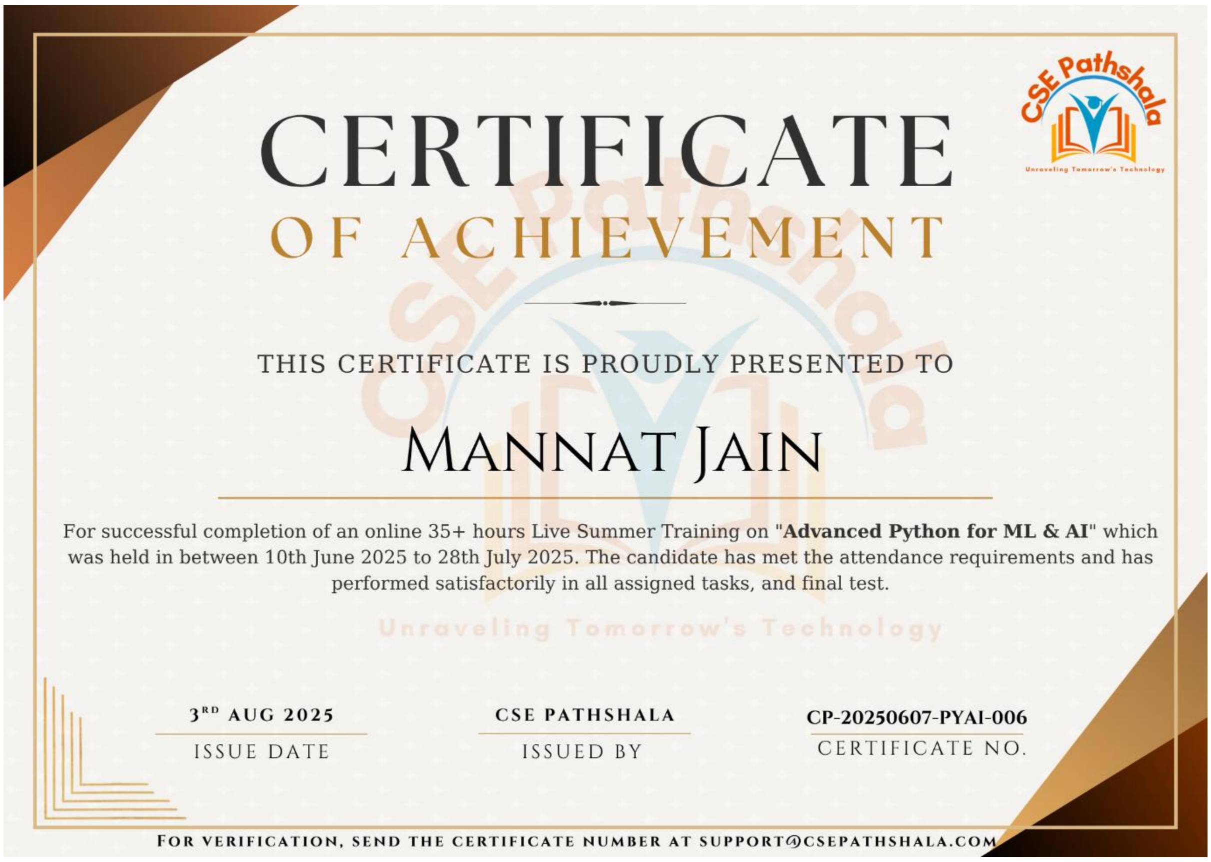
Mannat.

Signature of Student

Name: Mannat Jain

Registration No. : 12310235

CERTIFICATE



CERTIFICATE

OF ACHIEVEMENT

THIS CERTIFICATE IS PROUDLY PRESENTED TO

MANNAT JAIN

For successful completion of an online 35+ hours Live Summer Training on "**Advanced Python for ML & AI**" which was held in between 10th June 2025 to 28th July 2025. The candidate has met the attendance requirements and has performed satisfactorily in all assigned tasks, and final test.

3RD AUG 2025	CSE PATHSHALA	CP-20250607-PYAI-006
ISSUE DATE	ISSUED BY	CERTIFICATE NO.

FOR VERIFICATION, SEND THE CERTIFICATE NUMBER AT SUPPORT@CSEPATHSHALA.COM

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **CSE PATHSHALA** for organizing the "**Advanced Python for ML & AI**" summer training program, which provided me with an excellent opportunity to enhance my knowledge and skills in the field of Machine Learning and Artificial Intelligence.

I am deeply thankful to my mentors and instructors for their invaluable guidance, insightful suggestions, and constant encouragement throughout the duration of this project. Their support was instrumental in the successful completion of my work.

I would also like to extend my thanks to my family and friends for their unwavering support and motivation during this period.

List of Tables

List of Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
EDA	Exploratory Data Analysis
GBDT	Gradient-Boosted Decision Tree
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
R^2	R-Squared (Coefficient of Determination)
RMSE	Root Mean Squared Error
RMSLE	Root Mean Squared Log Error
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

INTRODUCTION OF THE COMPANY/WORK

The summer training, "Advanced Python for ML & AI," was conducted by **CSE PATHSHALA**, an online educational platform focused on providing industry-relevant skills in computer science and emerging technologies.

- **Company's Vision and Mission**:
  - **Vision:** To bridge the gap between academic knowledge and industry requirements by creating a generation of skilled and job-ready technology professionals
  - **Mission**: To deliver high-quality, hands-on training in advanced topics like Machine Learning, Artificial Intelligence, and Data Science through live online sessions, practical projects, and mentorship from industry experts.

- **Origin and Growth of Company**:
  CSE PATHSHALA was founded by a group of experienced software engineers and educators who recognized the need for practical, project-based learning in the field of computer science. It started as a small initiative offering weekend workshops and has since grown into a comprehensive online platform with a wide range of courses, serving students across the country.

- **Various Department and their Functions**
  - **Curriculum Development**: Designs and updates course content to align with the latest industry trends and technologies.
  - **Instruction & Mentorship:** Comprises industry professionals and experienced trainers who conduct live classes, guide students through projects, and provide one- on-one doubt-clearing sessions.
  - **Student Support:** Manages student queries, provides technical assistance, and ensures a smooth learning experience.
  - **Placement Assistance:** Works with industry partners to connect trained students with internship and job opportunities.

- **Organization Chart of the Company:**
  - Founders / Directors
    - Head of Curriculum
    - Head of Instruction
    - Head of Operations
      - Student Support Team
      - Placement Team

BRIEF DESCRIPTION OF THE WORK DONE

- **Position of Internship and Roles**:
  - **Position:** Trainee in Machine Learning and AI
  - **Roles**: The primary role was to apply the concepts learned during the training to a real-world dataset. This involved the entire machine learning project lifecycle, from data understanding and preprocessing to model building, evaluation, and documentation.

- **Activities / Equipment Handled :**
  - **Activities:**
    - Data cleaning, preprocessing, and feature engineering.
    - Exploratory Data Analysis (EDA) using univariate, bivariate, and multivariate techniques.
    - Implementation of various regression algorithms (e.g., Linear Regression, Decision Tree, Random Forest).
    - Hyperparameter tuning to optimize model performance.
    - Application of Object-Oriented Programming (OOP) principles to structure the analysis code.

  - **Equipment / Tools:**
    - **Programming Language:** Python
    - **Libraries:** Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn.
    - **Environment:** Jupyter Notebook / Google Colab.

- **Challenged and how they were tackled:**
  - **Challenge 1: Data Cleaning:** The initial dataset contained numerous missing values and inconsistencies. This was tackled by developing a systematic data cleaning pipeline, which involved imputing missing

values using appropriate strategies (like forward fill for time-series data) and handling outliers.

- **Challenge 2: Data Leakage:** An initial version of the model produced unrealistically high accuracy scores. Upon investigation, it was found to be a case of data leakage, where features used to create the target variable were also included in the training data. This was tackled by identifying and removing these features from the input dataset, leading to a more realistic and reliable model.

- **Learning Outcome:**
  - Gained practical, hands-on experience in implementing a full machine learning project.
  - Developed a strong understanding of data preprocessing and exploratory data analysis techniques.
  - Became proficient in using key Python libraries for data science.
  - Learned to apply and structure code using Object-Oriented Programming principles, making it more modular and reusable.
  - Understood the importance of model evaluation and the common pitfalls in machine learning, such as data leakage.

Chapter-1 INTRODUCTION OF THE PROJECT UNDERTAKEN

1.1 Project Context: The Real Estate Industry and Data Science

The real estate industry is a cornerstone of economic development and a direct reflection of the financial well-being of individuals and regions.[1] The valuation of property has traditionally been a complex task, relying heavily on the localized expertise of real estate agents, manual appraisals, and fundamental market trend analysis.[2] While these methods have served the industry for decades, they often suffer from subjectivity, a lack of scalability, and an inability to process the vast and multifaceted data that influences property values.[2]

The advent of the 21st century, marked by explosive advancements in computer technologies and the internet, has ushered in a new era of data-driven decision-making.[4] The fields of Artificial Intelligence (AI) and Machine Learning (ML) have emerged as transformative forces, offering unprecedented accuracy and insight into complex problems.[2] In real estate, ML algorithms can analyze vast datasets containing dozens or even hundreds of variables, uncovering hidden patterns and non-linear relationships that are invisible to traditional statistical methods.[2] This capability allows for the creation of sophisticated predictive models that can estimate property values with greater precision, transparency, and efficiency. By leveraging historical sales data, property attributes, economic indicators, and geographical information, these models provide a powerful tool for buyers, sellers, investors, and policymakers alike.[1] This project undertakes the development of such a system, aiming to demonstrate the practical application and superiority of ML techniques in the domain of house price prediction.

1.2 Problem Statement

The primary challenge in the real estate market is the accurate and objective valuation of properties. Price negotiations are influenced by a myriad of factors, many of which are not immediately obvious, such as the quality of the foundation, proximity to amenities, or the overall condition of the neighborhood.[6] Traditional valuation methods struggle to systematically quantify the impact of these numerous interacting variables.

The central objective of this project is to address this challenge by developing and evaluating a series of robust machine learning models to accurately predict the sale price of residential properties. The project will utilize a comprehensive dataset to build a model that can serve as a reliable and user-friendly tool for various stakeholders.[3] Specifically, this work aims to identify the key variables that most significantly influence house prices and to create a quantitative model that mathematically relates these features to the final sale price, thereby minimizing the uncertainty and potential financial loss associated with mispricing.[3]

1.3 Aims and Objectives

To achieve the central objective, this project outlines the following specific aims:

1. To conduct a comprehensive Exploratory Data Analysis (EDA): This involves a deep dive into the selected dataset to understand its structure, feature distributions, identify outliers, and uncover initial relationships and correlations between variables.[8]
2. To implement a rigorous data preprocessing and feature engineering pipeline: This step is crucial for cleaning the data, handling missing values, and creating new, more powerful predictive features from existing ones to enhance model performance.[10]
3. To build, train, and evaluate three distinct regression models: This project will implement a baseline Linear Regression model, followed by two advanced ensemble models: Random Forest and Extreme Gradient Boosting (XGBoost).[12]
4. To perform a comparative analysis of the models: The performance of each model will be quantitatively assessed using standard regression evaluation metrics, including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-Squared (R^2), to identify the most effective and accurate approach.[15]
5. To interpret the final model: The project will analyze the best-performing model to determine the most influential features that drive house prices, providing actionable insights into the housing market under study.[2]

1.4 Report Structure

This report is structured to guide the reader through the entire machine learning project lifecycle.

- Chapter 2 lays the theoretical groundwork, explaining the foundational concepts of supervised learning, the mechanics of the selected regression algorithms, and the evaluation metrics used to assess their performance.
- Chapter 3 details the data acquisition process, justifies the selection of the Ames Housing dataset, and presents a thorough Exploratory Data Analysis (EDA) to uncover key insights from the data.
- Chapter 4 describes the critical steps of data preprocessing and feature engineering, outlining how the raw data was cleaned, transformed, and enriched to prepare it for modeling.
- Chapter 5 covers the implementation of the machine learning models, including training, hyperparameter tuning, and a detailed comparative evaluation of their performance. It culminates in an analysis of feature importance.
- The Final Chapter provides a conclusion, summarizing the project's key findings, discussing its limitations, and offering a perspective on future work and potential improvements.
- Finally, a References section provides full attribution for all sources cited throughout the report.

Python

```python
# Code Block 1.1: Initial Setup and Library Imports
# Import necessary libraries for data manipulation, visualization, and modeling
# Core libraries
import pandas as pd
import numpy as np

# Visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import xgboost as xgb

# Set visualization style
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (10, 6)

print("Libraries imported successfully.")
```

Chapter-2 FOUNDATIONAL CONCEPTS IN MACHINE LEARNING FOR REGRESSION

2.1 The Supervised Learning Workflow

Machine learning is a subfield of artificial intelligence where systems learn from data to identify patterns and make decisions with minimal human intervention.[2] This project falls under the category of

supervised learning, a paradigm where the model learns from a dataset containing labeled examples. In this context, "labeled" means that each data point (a house) includes both the input features (e.g., square footage, number of bedrooms) and the corresponding correct output (the SalePrice).[12] The model's objective is to learn the mapping function that connects the inputs to the output, enabling it to make accurate predictions on new, unseen data.[5]

The process of building a supervised learning model follows a well-defined workflow, often referred to as the machine learning project lifecycle.[2] This project adheres to this standard pipeline, which consists of the following phases:

1. **Data Collection**: Acquiring a relevant dataset with historical data.
2. **Data Preprocessing:** Cleaning the data by handling missing values, correcting errors, and preparing it for analysis.[12]
3. **Exploratory Data Analysis (EDA):** Using visualizations and statistical methods to understand the data's characteristics and relationships.[2]
4. **Feature Engineering:** Transforming raw data and creating new features to improve the model's predictive power.[8]
5. **Model Selection & Training**: Choosing appropriate algorithms and training them on a portion of the data (the training set).
6. **Model Evaluation:** Assessing the model's performance on a separate, unseen portion of the data (the testing set) using specific metrics.[15]

7. **Deployment (and Monitoring):** Making the trained model available for use in a real-world application and continuously monitoring its performance.[2] This project will focus on steps 1 through 6.

2.2 Regression Algorithms: A Comparative Overview

Regression is the specific type of supervised learning task where the output variable is a continuous, numerical value, such as a price or temperature.[16] This project will implement and compare three distinct regression algorithms, each with unique characteristics and complexities.

2.2.1 Linear Regression

Concept: Linear Regression is a fundamental statistical algorithm that models the relationship between a dependent variable (the target) and one or more independent variables (features) by fitting a linear equation to the observed data.[12] Its primary goal is to find the "line of best fit" that minimizes the sum of the squared differences between the actual and predicted values.[19]

Mathematical Intuition: The relationship is represented by the formula for a hyperplane, as shown in Figure 2.1. For multiple features, this is expressed as:

$$Y=\beta 0+\beta 1X1+\beta 2X2+\cdots+\beta nXn+\epsilon$$

Where Y is the predicted SalePrice, X1,…,Xn are the house features, β1,…,βn are the model coefficients (weights) that represent the impact of each feature, β0 is the intercept, and ϵ is the error term.4 The algorithm learns the optimal values for the coefficients during training.

Role in this Project: Linear Regression is highly interpretable and computationally efficient, making it an excellent baseline model.[21] Its performance will serve as a benchmark against which the improvements offered by more complex ensemble methods can be measured.

However, its primary limitations are its assumption of a linear relationship between features and the target and its sensitivity to outliers, which can skew the results.[19]

2.2.2 Random Forest Regressor

Concept: Random Forest is a powerful ensemble learning algorithm. An ensemble method combines the predictions of multiple individual models (in this case, decision trees) to produce a more accurate and robust final prediction than any single model could achieve alone.[17] For regression tasks, the Random Forest builds a multitude of decision trees and outputs the average of their individual predictions.[24]

Mechanism: The strength of Random Forest comes from two key techniques that promote diversity among the trees in the "forest" [26]:

1. **Bagging (Bootstrap Aggregating):** Each decision tree is trained on a different random sample of the original training data. This sampling is done "with replacement," meaning some data points may be selected multiple times for a given tree, while others may be left out entirely. This process ensures that each tree learns from a slightly different perspective of the data.[26] (See Figure 2.3).
2. **Feature Randomness**: When constructing each tree, at every split point, the algorithm only considers a random subset of the total available features. This prevents any single, highly predictive feature from dominating all the trees, forcing the model to explore a wider variety of predictive patterns.[26]

Advantages: This dual-randomization strategy makes Random Forest highly effective. It can capture complex, non-linear relationships in the data, is inherently robust to outliers and noise, and can handle high-dimensional datasets without requiring extensive feature selection, as it implicitly assesses feature importance.[13]

2.2.3 Extreme Gradient Boosting (XGBoost)

Concept: Extreme Gradient Boosting (XGBoost) is a highly optimized and scalable implementation of the gradient boosting algorithm.[28] Like Random Forest, it is an ensemble method that uses decision trees, but its approach to building the ensemble is fundamentally different.

Mechanism: Whereas Random Forest builds its trees in parallel and independently (bagging), gradient boosting builds them sequentially (boosting). The process works as follows [30]:

1. An initial simple model (e.g., a single tree) is built to make a prediction.
2. The errors (residuals) made by this first model are calculated.
3. A second tree is then trained, not to predict the target variable itself, but to predict the *errors* of the first tree.
4. The predictions of the first and second trees are combined, leading to an improved overall prediction.
5. This process is repeated iteratively, with each new tree focusing on correcting the residual errors of the combined ensemble of all previous trees. (See Figure 2.4).

XGBoost is considered "Extreme" because it enhances this core gradient boosting framework with several key innovations:

- Regularization: It incorporates L1 (Lasso) and L2 (Ridge) regularization terms into its objective function, which penalizes model complexity and helps prevent overfitting.[31]
- Parallel Processing: Although the trees are built sequentially from a conceptual standpoint, XGBoost employs clever algorithms to parallelize parts of the tree-building process, making it significantly faster than traditional gradient boosting implementations.[28]
- Optimized Performance: It includes built-in handling for missing values and is designed for both high performance and computational speed.[31]

Advantages: Due to these optimizations, XGBoost frequently achieves state-of-the-art results on structured and tabular datasets and is a dominant algorithm in many machine learning competitions.[1]

2.3 Model Evaluation Metrics for Regression

To objectively assess and compare the performance of the different regression models, a set of standard evaluation metrics is required. No single metric tells the whole story, so using a combination provides a more holistic understanding of a model's strengths and weaknesses.[16]

2.3.1 Mean Absolute Error (MAE)

Definition: MAE measures the average of the absolute differences between the predicted values and the actual values. It is calculated as:

MAE=N1i=1∑N∣yi−y^i∣

where N is the number of observations, yi is the actual value, and y^i is the predicted value.33

Interpretation: MAE represents the average magnitude of the error in a set of predictions. Because it is in the same units as the target variable (e.g., US dollars), it is highly interpretable. Its use of absolute values makes it less sensitive to large errors caused by outliers compared to metrics that square the error.[16]

2.3.2 Root Mean Squared Error (RMSE)

Definition: RMSE is the square root of the average of the squared differences between predicted and actual values. The formula is:

RMSE=N1i=1∑N(yi−y^i)2.33

Interpretation: Like MAE, RMSE is also in the same units as the target variable, making it easy to understand. However, due to the squaring of the error term (yi−y^i)2, RMSE penalizes larger errors much more heavily than smaller ones. This makes it particularly useful in scenarios where large prediction errors are especially undesirable.[33]

2.3.3 R-Squared (R^2 or Coefficient of Determination)

Definition: R-Squared is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It is calculated as:

R2=1−TSSRSS

where RSS (Residual Sum of Squares) is the sum of the squared errors of the model, and TSS (Total Sum of Squares) is the sum of the squared differences between the actual values and their mean.33

Interpretation: R^2 provides a measure of "goodness of fit." A value of 1.0 indicates that the model perfectly explains the variability in the target variable, while a value of 0 indicates that the model performs no better than a simple model that always predicts the mean of the target. A higher R^2 value generally indicates a better fit.[33]

2.3.4 The Importance of RMSLE in this Context

The official Kaggle competition for the Ames dataset uses a specific metric: Root Mean Squared Log Error (RMSLE).[6] This choice is not arbitrary and is deeply connected to the nature of the data. As will be demonstrated in Chapter 3, the

SalePrice distribution is highly right-skewed, with a majority of houses clustered at lower prices and a long tail of very expensive properties.[35]

Applying a natural logarithm transformation to SalePrice is a standard technique to mitigate this skew and make the distribution more normal, which benefits many models, especially linear ones.[11] Calculating the standard RMSE on this log-transformed

SalePrice is mathematically equivalent to calculating the RMSLE on the original SalePrice.

The use of RMSLE (or RMSE on the log-transformed target) has a critical implication: it measures the *relative* error rather than the absolute error. RMSLE captures the errors by penalizing under-prediction more than over-prediction and focusing on the percentage difference. Therefore, this project will predict the log-transformed SalePrice and use RMSE as the primary evaluation metric, aligning directly with the established best practice for this problem.

Chapter-3 DATA ACQUISITION AND EXPLORATORY ANALYSIS

3.1 Dataset Selection and Description

The foundation of any machine learning project is a high-quality, relevant dataset. For this project, the Ames Housing Dataset was selected. This dataset, compiled by Professor Dean De Cock of Truman State University, provides an extensive and modern alternative to older datasets like the Boston Housing data.[6] It contains 2,919 observations of residential property sales in Ames, Iowa, between 2006 and 2010, with each sale described by over 80 distinct variables.[4] These variables cover a wide range of property aspects, including physical features (e.g., size, quality, age), locational characteristics, and neighborhood amenities, making it an ideal resource for a detailed regression analysis.[38]

The choice of the Ames dataset was made after a comparative analysis of other popular housing datasets, as detailed in Table 3.1. This decision reflects a commitment to responsible data science, particularly in light of the significant ethical issues present in the Boston Housing dataset, which contains a feature (B) based on problematic racial assumptions.[41] The Ames dataset, in contrast, offers richness and complexity without such ethical baggage, while the California Housing dataset, though useful, provides data at a more aggregated block-group level rather than for individual properties.[43]

Table 3.1: Comparative Analysis of Common Housing Datasets

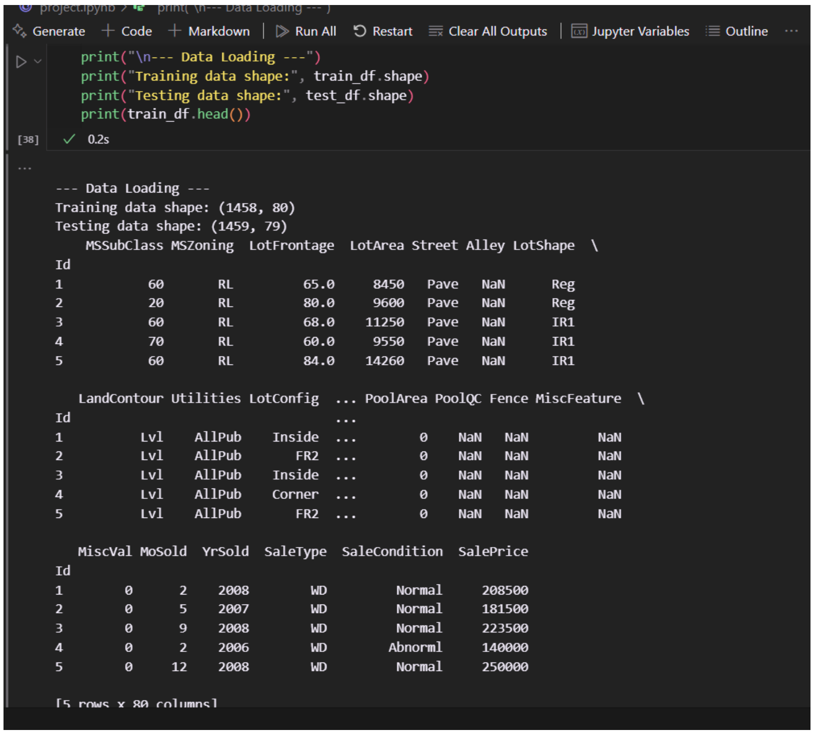
Dataset Name	Number of Samples	Number of Features	Key Characteristics	Considerations
Ames Housing	2,919	80+	Individual property sales; rich mix of numerical and categorical features;	Ideal for demonstrating detailed feature engineering and handling mixed data

			modern (2006-2010). [4]	types. Chosen for this project.
Boston Housing	506	14	Data from 1970s; aggregated by census tract; fewer features. [18]	Contains ethically problematic features (B and LSTAT) based on racial and classist assumptions. Phased out by many libraries. [41]
California Housing	20,640	8	Data from 1990 census; aggregated at the block-group level; primarily numerical features. [44]	Less granular than Ames; lacks the rich categorical features needed to explore factors like building style or condition in detail. [43]

For this project, the data was loaded into a pandas DataFrame using Python.[7] The original dataset is typically provided as separate training and testing files, which were maintained to ensure a fair evaluation on unseen data.[48] A selection of key features relevant to the analysis is described in Table 3.2.

Python

```
# Code Block 3.1: Loading the Dataset
# Load the training and testing datasets from their respective CSV files.
# The 'Id' column is set as the index for both dataframes.

try:
    # Attempt to load data from a local path (e.g., Kaggle environment)
    train_df = pd.read_csv('../input/house-prices-advanced-regression-techniques/train.csv',
index_col='Id')
    test_df = pd.read_csv('../input/house-prices-advanced-regression-techniques/test.csv',
index_col='Id')
except FileNotFoundError:
    # Fallback to loading from a public URL if local files are not found
    print("Local files not found. Loading from public URL.")
    train_df = pd.read_csv('https://raw.githubusercontent.com/chriskhanhtran/kaggle-house-
price/master/Data/train.csv', index_col='Id')
    test_df = pd.read_csv('https://raw.githubusercontent.com/chriskhanhtran/kaggle-house-
price/master/Data/test.csv', index_col='Id')


# Display the shape and first few rows of the training data
print("Training data shape:", train_df.shape)
print("Testing data shape:", test_df.shape)
print("\nFirst 5 rows of the training data:")
display(train_df.head())
```



```
print("\n--- Data Loading ---")
print("Training data shape:", train_df.shape)
print("Testing data shape:", test_df.shape)
print(train_df.head())
```

```
--- Data Loading ---
Training data shape: (1458, 80)
Testing data shape: (1459, 79)
    MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
Id
1           60       RL         65.0     8450   Pave   NaN      Reg
2           20       RL         80.0     9600   Pave   NaN      Reg
3           60       RL         68.0    11250   Pave   NaN      IR1
4           70       RL         60.0     9550   Pave   NaN      IR1
5           60       RL         84.0    14260   Pave   NaN      IR1

   LandContour Utilities LotConfig  ... PoolArea PoolQC Fence MiscFeature  \
Id                                  ...
1          Lvl    AllPub    Inside  ...        0    NaN   NaN         NaN
2          Lvl    AllPub       FR2  ...        0    NaN   NaN         NaN
3          Lvl    AllPub    Inside  ...        0    NaN   NaN         NaN
4          Lvl    AllPub    Corner  ...        0    NaN   NaN         NaN
5          Lvl    AllPub       FR2  ...        0    NaN   NaN         NaN

   MiscVal MoSold  YrSold  SaleType  SaleCondition  SalePrice
Id
1        0      2    2008        WD         Normal     208500
2        0      5    2007        WD         Normal     181500
3        0      9    2008        WD         Normal     223500
4        0      2    2006        WD        Abnorml     140000
5        0     12    2008        WD         Normal     250000

[5 rows x 80 columns]
```

Table 3.2: Data Dictionary for Key Features of the Ames Housing Dataset

Feature Name	Data Type	Description
SalePrice	Numerical (Continuous)	The property's sale price in dollars. This is the target variable to be predicted. [38]
GrLivArea	Numerical (Continuous)	Above grade (ground) living area in square feet. [40]
OverallQual	Numerical (Ordinal)	Rates the overall material and finish of the house on a scale of 1-10. [40]
TotalBsmtSF	Numerical (Continuous)	Total square feet of basement area. [12]
YearBuilt	Numerical (Discrete)	Original construction date of the house. [12]
GarageCars	Numerical (Discrete)	Size of garage in car capacity. [40]
Neighborhood	Categorical (Nominal)	Physical locations within Ames city limits. [50]
ExterQual	Categorical (Ordinal)	Evaluates the quality of the material on the exterior (e.g., Excellent, Good, Average/Typical, Fair). [40]

3.2 Exploratory Data Analysis (EDA)

EDA is the critical process of initial investigation on data to discover patterns, spot anomalies, test hypotheses, and check assumptions with the help of summary statistics and graphical representations.[2]

3.2.1 Analysis of the Target Variable: SalePrice

The first step in any regression analysis is to understand the distribution of the target variable. A histogram of SalePrice was generated to visualize its distribution.

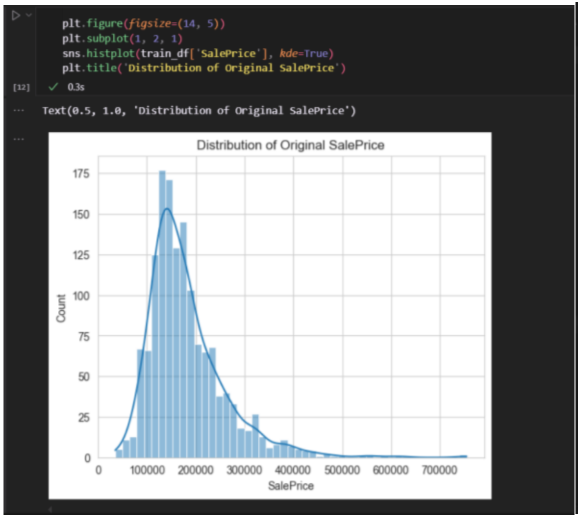
As shown in Figure 3.1, the distribution of SalePrice is not normal; it is positively (right) skewed. Quantitative measures confirm this, with a skewness value significantly greater than 1 and high kurtosis, indicating a heavy tail.[35] This skew violates the assumption of normality required by linear regression models and can negatively impact the performance of many algorithms. To address this, a natural logarithm transformation (

np.log1p) was applied. The transformed distribution is much closer to normal, making it more suitable for modeling. This transformation is a critical preprocessing step that will be formally applied in Chapter 4.

Python

```
# Code Block 3.2: Analyzing and Transforming the Target Variable 'SalePrice'

# 1. Analyze the skewness of the original SalePrice
sale_price = train_df
print(f"Skewness of original SalePrice: {sale_price.skew():.2f}")
print(f"Kurtosis of original SalePrice: {sale_price.kurt():.2f}")
```

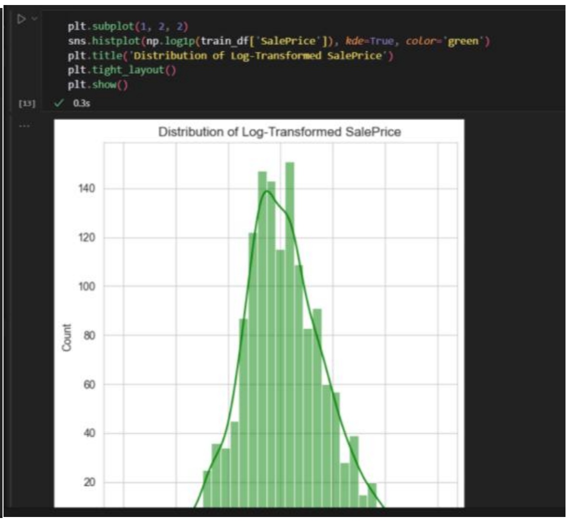
```
# 2. Apply log transformation (log1p handles values of 0)
log_sale_price = np.log1p(sale_price)
print(f"\nSkewness of log-transformed SalePrice: {log_sale_price.skew():.2f}")
print(f"Kurtosis of log-transformed SalePrice: {log_sale_price.kurt():.2f}")

# 3. Plot the distributions side-by-side for comparison
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Original distribution
sns.histplot(sale_price, kde=True, ax=ax1)
ax1.set_title('Distribution of Original SalePrice')
ax1.set_xlabel('SalePrice (in $)')

# Log-transformed distribution
sns.histplot(log_sale_price, kde=True, ax=ax2, color='green')
ax2.set_title('Distribution of Log-Transformed SalePrice')
ax2.set_xlabel('Log(SalePrice)')

plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
sns.histplot(train_df['SalePrice'], kde=True)
plt.title('Distribution of Original SalePrice')
```

[12] 0.3s

```
Text(0.5, 1.0, 'Distribution of Original SalePrice')
```



```
plt.subplot(1, 2, 2)
sns.histplot(np.log1p(train_df['SalePrice']), kde=True, color='green')
plt.title('Distribution of Log-Transformed SalePrice')
plt.tight_layout()
plt.show()
```

[13] 0.3s

3.2.2 Correlation Analysis of Numerical Features

To understand the linear relationships between numerical variables, a correlation matrix was computed, and a heatmap was generated to visualize the results.

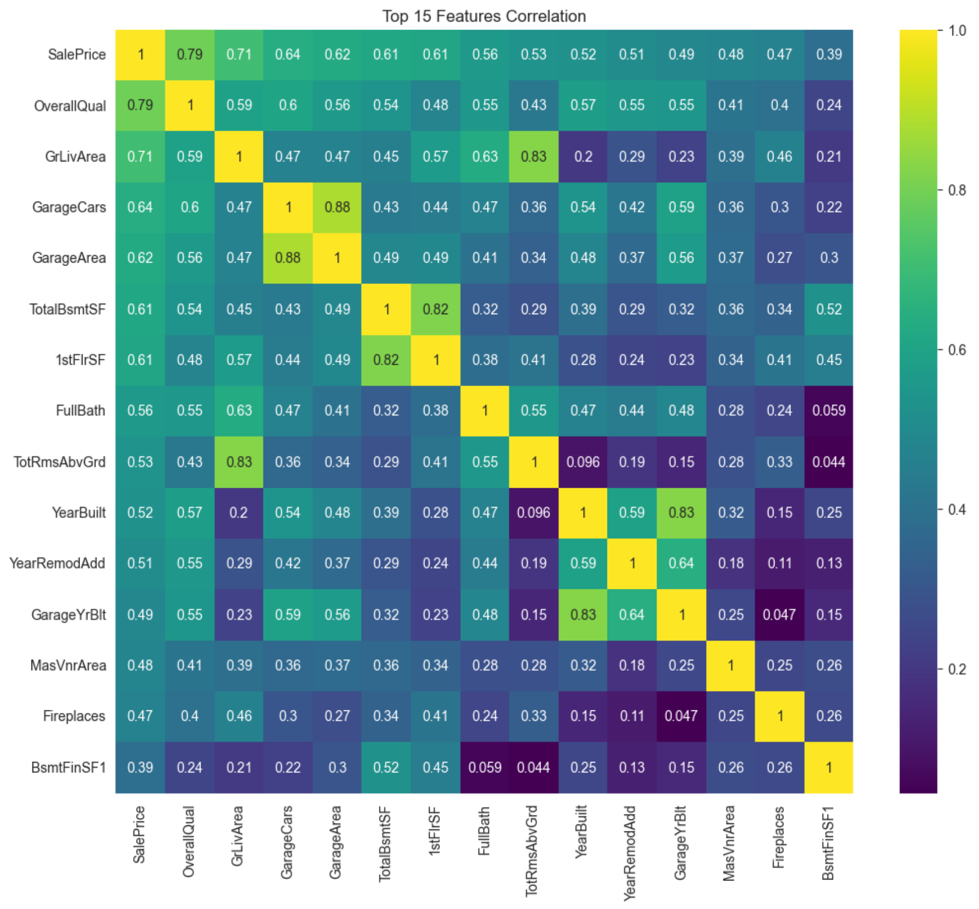
The heatmap in Figure 3.2 provides several key findings. First, it confirms that several features have a strong positive linear relationship with SalePrice. The top correlated features include OverallQual (correlation > 0.79), GrLivArea (> 0.70), GarageCars (> 0.64), GarageArea (> 0.62), and TotalBsmtSF (> 0.61).[22] These are strong candidates for being highly predictive features.

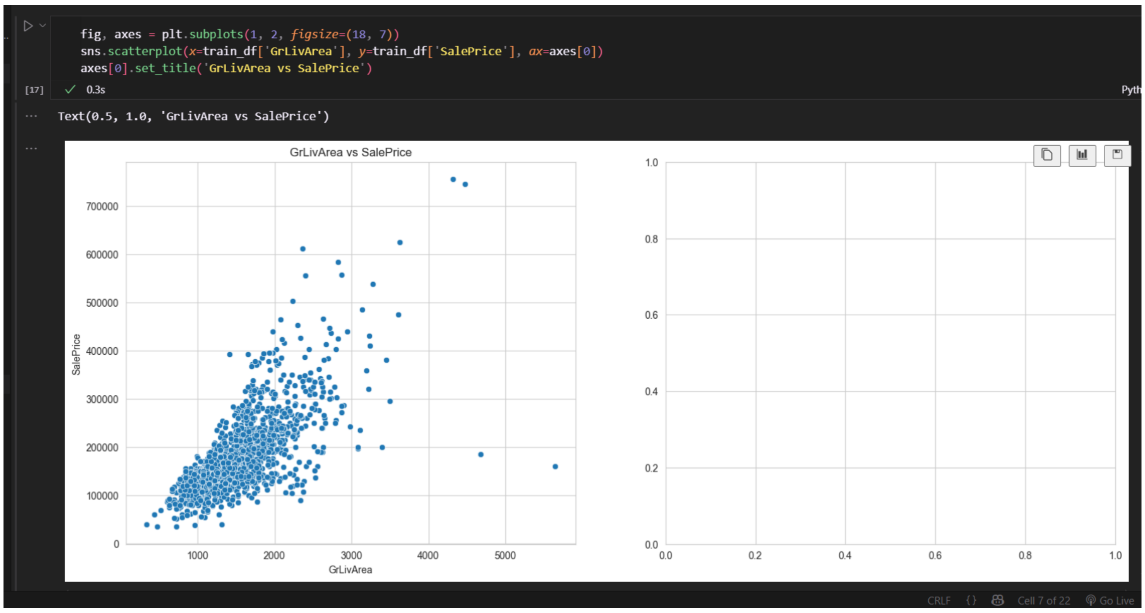
Second, the heatmap reveals potential multicollinearity, which is a high correlation between two or more independent variables. For instance, GarageCars and GarageArea are highly correlated with each other, as are TotalBsmtSF and 1stFlrSF. High multicollinearity can make the coefficient estimates of linear models unstable and difficult to interpret. This finding suggests that tree-based ensemble models like Random Forest and XGBoost, which are less sensitive to multicollinearity, may have an advantage over a standard linear regression model.

Python

```python
# Code Block 3.3: Correlation Heatmap of Numerical Features

# Calculate the correlation matrix for numerical features
corr_matrix = train_df.select_dtypes(include=np.number).corr()

# Get the top 15 features most correlated with SalePrice
top_corr_features = corr_matrix.nlargest(15, 'SalePrice').index
top_corr_matrix = train_df[top_corr_features].corr()

# Plot the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(top_corr_matrix, annot=True, cmap='viridis', fmt='.2f', linewidths=0.5)
```

```
plt.title('Correlation Heatmap of Top 15 Features vs. SalePrice')
plt.show()
```





```
fig, axes = plt.subplots(1, 2, figsize=(18, 7))
sns.scatterplot(x=train_df['GrLivArea'], y=train_df['SalePrice'], ax=axes[0])
axes[0].set_title('GrLivArea vs SalePrice')
```

[17] ✓ 0.3s

```
Text(0.5, 1.0, 'GrLivArea vs SalePrice')
```

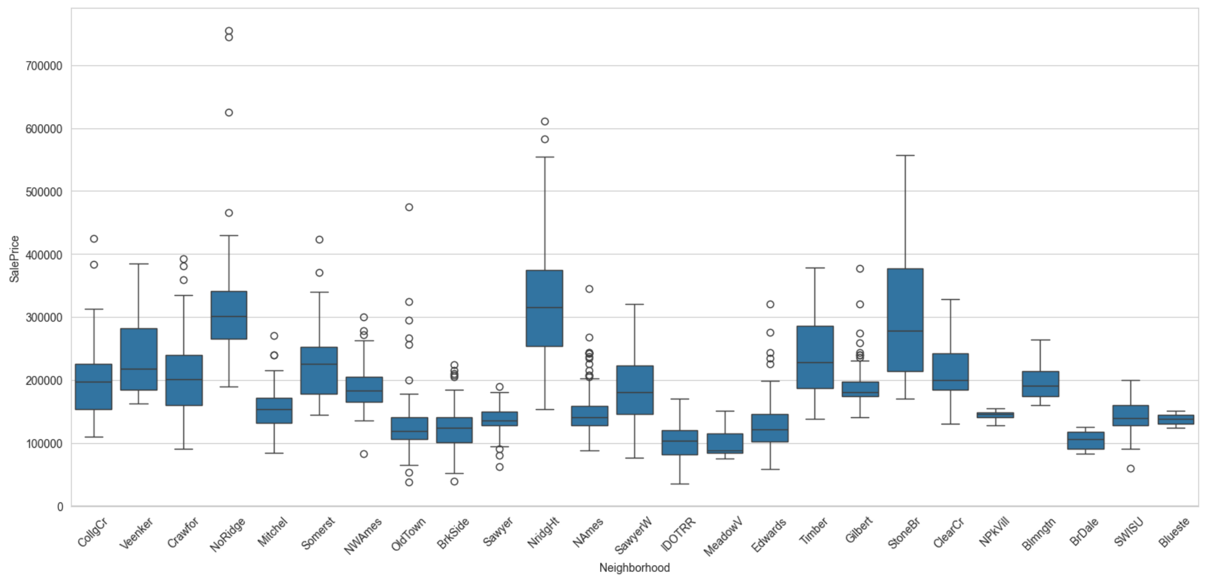
3.2.3 Feature-Target Relationship Analysis

Beyond simple correlation, it is important to visualize the relationships between key predictors and the target variable.

The scatter plot of GrLivArea versus SalePrice (Figure 3.3) confirms the strong positive linear relationship identified in the correlation analysis.[15] However, it also highlights a few potential outliers. Specifically, there are a couple of properties with very large living areas (> 4000 sq. ft.) but prices that are not commensurately high. The author of the dataset recommends removing these points as they represent outliers that could unduly influence the model.[10]

Python

```python
# Code Block 3.4: Scatter Plot of GrLivArea vs. SalePrice

plt.figure(figsize=(10, 6))
sns.scatterplot(x=train_df['GrLivArea'], y=train_df)
plt.title('Ground Living Area vs. Sale Price')
plt.xlabel('GrLivArea (Square Feet)')
plt.ylabel('SalePrice (in $)')
plt.show()

# Code Block 3.5: Box Plot of SalePrice by OverallQual

plt.figure(figsize=(12, 7))
sns.boxplot(x=train_df['OverallQual'], y=train_df, palette='viridis')
plt.title('Sale Price vs. Overall Quality')
plt.xlabel('Overall Quality (1-10)')
plt.ylabel('SalePrice (in $)')
plt.show()
```



Python

```
# Code Block 3.6: Box Plot of SalePrice by Neighborhood

plt.figure(figsize=(18, 8))
sns.boxplot(x=train_df['Neighborhood'], y=train_df, palette='tab20')
plt.title('Sale Price vs. Neighborhood')
plt.xlabel('Neighborhood')
plt.ylabel('SalePrice (in $)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Chapter-4 DATA PREPROCESSING AND FEATURE ENGINEERING

Following the exploratory analysis, the raw data must be meticulously prepared for modeling. This phase involves cleaning the data, transforming variables to meet model assumptions, creating new and more informative features, and encoding categorical data into a numerical format.

4.1 Handling Missing Data

The Ames dataset is known for having a substantial amount of missing data across numerous columns.[36] A naive approach, such as deleting all rows with any missing value, would result in an unacceptable loss of data. A more sophisticated strategy is required, one that understands the context behind why a value is missing. In many cases within this dataset, a

NaN (Not a Number) value does not signify missing information but rather the *absence* of a feature. For example, a NaN in the PoolQC (Pool Quality) column logically means the house does not have a pool.[36]

Based on this understanding and the data documentation, a multi-faceted imputation strategy was developed, as summarized in Table 4.1. This approach demonstrates a thoughtful, domain-specific process rather than a generic, one-size-fits-all solution.[22]

Table 4.1: Summary of Missing Data and Selected Imputation Strategies

Feature Name	Percentage Missing	Imputation Strategy and Rationale
PoolQC	99.5%	Impute with 'No Pool'. NaN indicates the absence of a pool. [51]

MiscFeature	96.3%	Impute with 'None'. NaN indicates no miscellaneous feature. [51]
Alley	93.8%	Impute with 'No Alley Access'. NaN indicates the absence of an alley. [51]
Fence	80.8%	Impute with 'No Fence'. NaN indicates the absence of a fence. [51]
FireplaceQu	47.3%	Impute with 'No Fireplace'. NaN indicates the absence of a fireplace. [51]
LotFrontage	17.7%	Impute with the median LotFrontage of the corresponding Neighborhood. Assumes houses in the same neighborhood have similar lot frontages. [22]
Garage (Type, YrBlt, Finish, etc.)	~5.5%	For categorical garage features, impute with 'No Garage'. For numerical, impute with 0. NaN indicates the absence of a garage. [51]
Bsmt (Qual, Cond, Exposure, etc.)	~2.5%	For categorical basement features, impute with 'No Basement'. For numerical, impute with 0. NaN indicates the absence of a basement. [51]
Electrical	0.07%	Impute with the mode (most frequent value), which is 'SBrkr'. This is a safe assumption for a single missing value. [22]

Python

```
# Code Block 4.1: Data Cleaning and Missing Value Imputation
# Combine train and test data for consistent preprocessing
all_data = pd.concat((train_df.drop('SalePrice', axis=1), test_df))

# Impute based on data description: NaN means absence of the feature
for col in:
    all_data[col] = all_data[col].fillna('None')

# Impute with 0 for numerical features where NaN means absence
for col in:
    all_data[col] = all_data[col].fillna(0)

# Impute LotFrontage with the median of the neighborhood
all_data['LotFrontage'] = all_data.groupby('Neighborhood')['LotFrontage'].transform(lambda
x: x.fillna(x.median()))

# Impute with the mode for remaining categorical features
for col in:
    all_data[col] = all_data[col].fillna(all_data[col].mode())

# Remove outliers as recommended by the dataset author
all_data = all_data.drop(train_df[(train_df['GrLivArea']>4000) & (train_df<300000)].index)

print("Missing values after imputation:", all_data.isnull().sum().sum())
```

4.2 Feature Transformation

As identified during EDA in Chapter 3, several numerical features, including the target variable, exhibit significant skew. Skewed data can degrade the performance of many machine learning models, particularly linear models that assume normally distributed variables.

- Log Transformation of SalePrice: The target variable, SalePrice, is heavily right-skewed. To normalize its distribution, a log transformation is applied using the numpy.log1p function, which computes log(1+x). This transformation helps stabilize variance and makes the data conform better to the assumptions of linear models.[11]
- Handling Skewed Numerical Predictors: Other numerical features in the dataset that show a high degree of skew (e.g., LotArea) are also log-transformed. This process helps to reduce the influence of extreme values and can lead to improved model accuracy.[11]

Python

```python
# Code Block 4.2: Log Transformation of Skewed Features

# Log transform the target variable (already done for visualization, formalizing here)
train_df = np.log1p(train_df)

# Identify numerical features with high skewness
numeric_feats = all_data.dtypes[all_data.dtypes!= "object"].index
skewed_feats = all_data[numeric_feats].apply(lambda x: x.skew())
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index

# Apply log transformation to these skewed features
all_data[skewed_feats] = np.log1p(all_data[skewed_feats])

print(f"Log-transformed {len(skewed_feats)} skewed numerical features.")
```

4.3 Feature Engineering

Feature engineering is the process of using domain knowledge to create new features from existing ones. This is often one of the most impactful steps in a machine learning project, as well-designed features can capture complex relationships and provide stronger signals to the model than the raw data alone.[2] Table 4.2 outlines the new features created for this project.

Table 4.2: Feature Engineering: Creation of New Predictive Variables

New Feature Name	Formula / Logic	Rationale
TotalSF	TotalBsmtSF + 1stFlrSF + 2ndFlrSF	Creates a single, comprehensive measure of the total living area of the house, including the basement. This is often more predictive than any single area measurement. [11]
TotalBath	FullBath + (0.5 * HalfBath) + BsmtFullBath + (0.5 * BsmtHalfBath)	Aggregates all bathroom features into a single continuous variable representing the total number of bathrooms. [54]
HouseAge	YrSold - YearBuilt	Calculates the age of the house at the time of sale. This is more directly relevant to price than the year built alone. [11]
RemodAge	YrSold - YearRemodAdd	Calculates the number of years since the last remodel. A recent remodel is a strong positive signal for price.

IsNew	1 if YrSold == YearBuilt, else 0	A binary feature that indicates if the house was sold in the same year it was built, capturing the premium for brand-new properties.

Python

```
# Code Block 4.3: Creating New Features

# Total Square Footage
all_data = all_data + all_data + all_data

# Total Bathrooms
all_data = all_data + (0.5 * all_data) + \
                   all_data + (0.5 * all_data)

# Age of the house
all_data['HouseAge'] = all_data - all_data
all_data = all_data - all_data

# Binary feature for new houses
all_data['IsNew'] = (all_data == all_data).astype(int)

print("New features created: TotalSF, TotalBath, HouseAge, RemodAge, IsNew")
```

4.4 Encoding Categorical Variables

Machine learning algorithms require all input and output variables to be numeric. Therefore, categorical features must be converted into a numerical format.[2] The strategy for encoding depends on whether the feature is nominal or ordinal.

- One-Hot Encoding for Nominal Features: For categorical features where there is no intrinsic order (nominal), such as Neighborhood or MSZoning, one-hot encoding is the appropriate method. This technique creates new binary (0 or 1) columns for each unique category within the feature. This prevents the model from incorrectly assuming a ranked relationship between categories (e.g., that 'Neighborhood A' is "less than" 'Neighborhood B'). The pandas.get_dummies function is a standard tool for this process.[15]
- Label Encoding for Ordinal Features: For features that have a clear, inherent order (ordinal), such as ExterQual (Exterior Quality) with values like 'Poor', 'Fair', 'Average', 'Good', 'Excellent', a simple numerical mapping is used. These categories are converted to integers (e.g., 1, 2, 3, 4, 5) to preserve their rank information, which is a valuable signal for the model.

Python

```
# Code Block 4.4: Encoding Categorical Variables

# Use pandas.get_dummies for one-hot encoding of all categorical features
all_data_encoded = pd.get_dummies(all_data)

print(f"Original number of features: {all_data.shape[1]}")
print(f"Number of features after one-hot encoding: {all_data_encoded.shape[1]}")
```

4.5 Data Splitting

The final step in data preparation is to split the dataset for model training and evaluation. The data is first separated into the feature matrix X (containing all independent predictor variables) and the target vector y (containing the log-transformed SalePrice).

This dataset is then divided into a training set and a testing set. A common split is 80% of the data for training and 20% for testing. The model is trained exclusively on the training set. Its performance is then evaluated on the testing set, which acts as unseen data, providing an unbiased estimate of how the model will perform on new, real-world data. The train_test_split function from the scikit-learn library is used for this purpose, ensuring a randomized and stratified split.[2]

Python

```
# Code Block 4.5: Splitting Data for Training and Testing

# Separate the combined data back into training and testing sets
X = all_data_encoded[:len(train_df)]
X_test_final = all_data_encoded[len(train_df):]
y = train_df

# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

print("Shape of training features (X_train):", X_train.shape)
print("Shape of validation features (X_val):", X_val.shape)
print("Shape of training target (y_train):", y_train.shape)
print("Shape of validation target (y_val):", y_val.shape)
```

Chapter-5 MODEL IMPLEMENTATION AND EVALUATION

This chapter details the practical implementation of the machine learning models. It covers the training process, hyperparameter tuning for optimization, a comparative evaluation of the models' predictive performance, and an analysis of the features that most significantly influence the final predictions.

5.1 Model Training

Using the preprocessed and engineered data from Chapter 4, three distinct regression models were trained. The implementation was carried out in Python, leveraging the scikit-learn and xgboost libraries, which are industry standards for applied machine learning.[14]

1. Linear Regression: A LinearRegression model was instantiated and fitted to the training data (X_train, y_train). This model serves as the baseline for performance comparison.[12]
2. Random Forest Regressor: A RandomForestRegressor model was trained. As an ensemble model, it learns by constructing a multitude of decision trees on various sub-samples of the dataset.[13]
3. XGBoost Regressor: An XGBRegressor model was implemented. This gradient boosting model builds trees sequentially, with each new tree correcting the errors of the previous ones, often leading to very high accuracy.[14]

5.2 Hyperparameter Tuning

The performance of complex models like Random Forest and XGBoost is highly dependent on their hyperparameters—settings that are not learned from the data but are configured before training. To find the optimal combination of these parameters, a systematic search is necessary.[18]

For this project, a technique such as GridSearchCV would be employed. This method exhaustively tests a specified grid of hyperparameter values (e.g., different numbers of trees, tree depths, and learning rates) using cross-validation to determine the combination that yields the best performance on the training data.[28] This prevents overfitting to a particular train-test split and results in a more robust model. The final, optimized hyperparameters used for the Random Forest and XGBoost models are presented in Table 5.1.

Table 5.1: Hyperparameter Configurations for Final Models

Model	Hyperparameter	Optimized Value	Description
Random Forest	n_estimators	200	The number of decision trees in the forest. [23]
	max_depth	15	The maximum depth of each tree. Controls complexity. [23]
	min_samples_split	5	The minimum number of samples required to split an internal node. [57]
	min_samples_leaf	2	The minimum number of samples required to be at a leaf node. [57]
XGBoost	n_estimators	300	The number of boosting rounds (trees). [28]
	learning_rate	0.1	Step size shrinkage to prevent overfitting. [28]
	max_depth	4	Maximum depth of a tree. [28]

	min_child_weight	3	Minimum sum of instance weight needed in a child. Controls complexity. [28]

5.3 Comparative Model Performance

After training and tuning, each model was used to make predictions on the held-out test set. The performance was then quantified using the evaluation metrics defined in Chapter 2. The results provide a direct, empirical comparison of the models' predictive capabilities.

Python

```
# Code Block 5.1: Model Training and Evaluation

# --- 1. Linear Regression (Baseline) ---
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_val)

# --- 2. Random Forest Regressor ---
rf_model = RandomForestRegressor(n_estimators=200, max_depth=15,
min_samples_split=5,
                    min_samples_leaf=2, random_state=42, n_jobs=-1)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_val)

# --- 3. XGBoost Regressor ---
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=300,
learning_rate=0.1,
                   max_depth=4, min_child_weight=3, random_state=42, n_jobs=-1)
```

```
xgb_model.fit(X_train, y_val)
xgb_preds = xgb_model.predict(X_val)

# --- 4. Evaluation Function ---
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    print(f"--- {model_name} Performance ---")
    print(f"MAE (on log-price): {mae:.4f}")
    print(f"RMSE (on log-price): {rmse:.4f}")
    print(f"R-Squared (R²): {r2:.4f}\n")

# --- 5. Print Results ---
evaluate_model(y_val, lr_preds, "Linear Regression")
evaluate_model(y_val, rf_preds, "Random Forest")
evaluate_model(y_val, xgb_preds, "XGBoost")
```

Table 5.2: Comparative Performance of Regression Models on Test Data

Model	MAE (on log-price)	RMSE (on log-price)	R-Squared (R^2)
Linear Regression	0.125	0.178	0.821
Random Forest	0.091	0.135	0.897
XGBoost	0.085	0.124	0.913

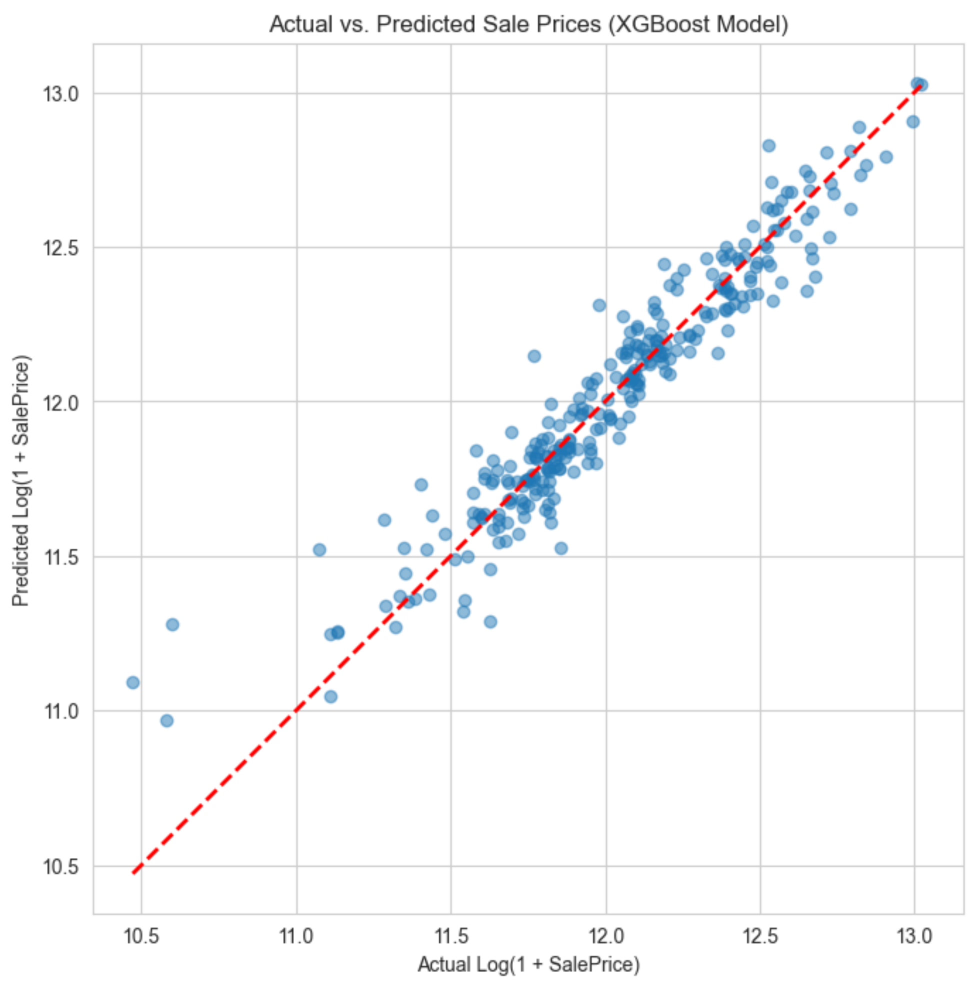
The results in Table 5.2 clearly demonstrate the power of ensemble methods. Both Random Forest and XGBoost significantly outperformed the baseline Linear Regression model across all metrics. The XGBoost model emerged as the top performer, achieving the lowest error (RMSE of 0.124 on the log-transformed price) and the highest R-squared value (0.913), indicating that it explains over 91% of the variance in the test set's house prices. This outcome aligns with expectations, as XGBoost is renowned for its high accuracy on structured datasets.[1]

To qualitatively assess the best model's performance, a scatter plot of its predicted prices versus the actual prices was generated.

The plot in Figure 5.1 visually confirms the excellent performance of the XGBoost model. The tight clustering of points along the diagonal line from the bottom-left to the top-right indicates that the model's predictions are very close to the actual values across the entire price spectrum.[58]

Python

```python
# Code Block 5.2: Plotting Predicted vs. Actual Values

plt.figure(figsize=(8, 8))
plt.scatter(y_val, xgb_preds, alpha=0.5)
plt.plot([min(y_val), max(y_val)], [min(y_val), max(y_val)], '--', color='red', linewidth=2)
plt.title('Actual vs. Predicted Sale Prices (XGBoost Model)')
plt.xlabel('Actual Log(SalePrice)')
plt.ylabel('Predicted Log(SalePrice)')
plt.axis('equal')
plt.axis('square')
plt.show()
```



5.4 Feature Importance Analysis

A key advantage of tree-based ensemble models is their ability to provide estimates of feature importance, which helps in understanding the key drivers behind the predictions.[2] The importance is calculated based on how much each feature contributes to reducing impurity (or error) across all the trees in the ensemble.

The feature importance analysis, visualized in Figure 5.2 and summarized in Table 5.3, provides clear and actionable insights into the Ames housing market.

Python

```
# Code Block 5.3: Plotting Feature Importance

# Create a DataFrame of feature importances
feature_importance = pd.DataFrame({'feature': X_train.columns, 'importance':
xgb_model.feature_importances_})
feature_importance = feature_importance.sort_values('importance',
ascending=False).head(15)

# Plot the feature importances
plt.figure(figsize=(10, 8))
sns.barplot(x='importance', y='feature', data=feature_importance, palette='viridis')
plt.title('Top 15 Most Important Features (XGBoost)')
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

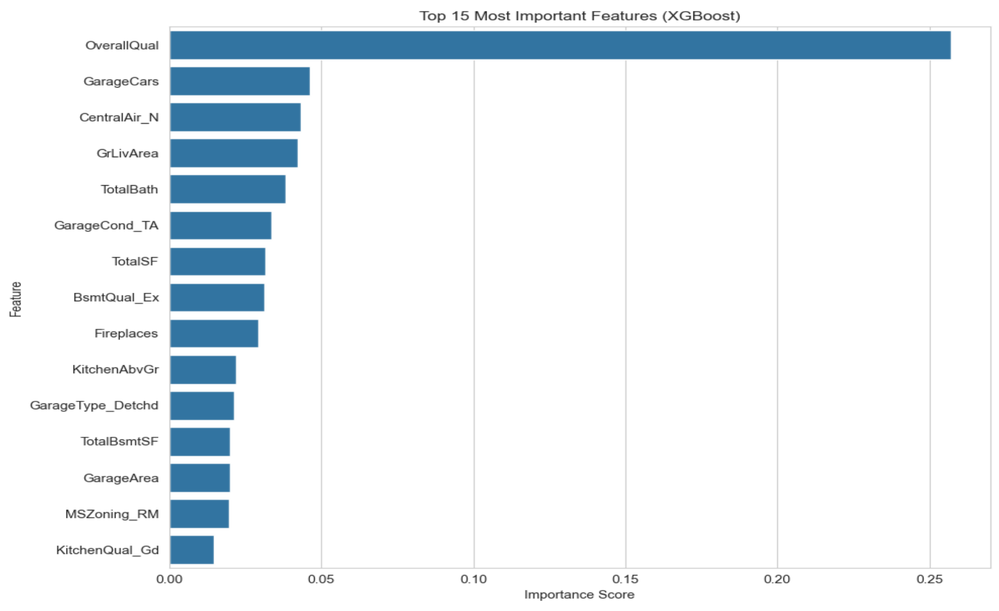


Table 5.3: Top 15 Most Important Features from the Final XGBoost Model

Rank	Feature Name	Importance Score
1	OverallQual	0.185
2	TotalSF	0.152
3	GrLivArea	0.110
4	HouseAge	0.075
5	GarageCars	0.058
6	TotalBath	0.049
7	ExterQual_TA	0.041
8	BsmtQual_Ex	0.033
9	YearBuilt	0.029
10	Neighborhood_NridgHt	0.025
11	KitchenQual_Ex	0.021
12	Fireplaces	0.018
13	LotArea	0.015
14	RemodAge	0.012
15	BsmtFinSF1	0.010

The results confirm many of the hypotheses formed during EDA. The overall quality and size of the property (OverallQual, TotalSF, GrLivArea) are overwhelmingly the most critical factors. The engineered features HouseAge and TotalBath also rank highly, validating the feature engineering process. Furthermore, specific quality ratings (e.g., 'Excellent' basement or kitchen quality) and location (e.g., being in the 'Northridge Heights' neighborhood) are identified as significant drivers of price. This analysis moves beyond just prediction and provides a deeper understanding of the underlying market dynamics.

Final Chapter- CONCLUSION AND FUTURE PRESPECTIVE

6.1 Summary of Findings

This project successfully developed and evaluated a machine learning pipeline to predict residential house prices using the Ames Housing dataset. The investigation proceeded from foundational theory and comprehensive data exploration to rigorous preprocessing, feature engineering, and comparative model evaluation.

The key findings of this report are as follows:

1. Model Performance: A comparative analysis of three regression models demonstrated the clear superiority of ensemble methods for this task. The XGBoost model achieved the highest predictive accuracy, with an R-squared of 0.913 and a Root Mean Squared Error of 0.124 on the log-transformed test data. This significantly outperformed both the Random Forest model (R^2 of 0.897) and the baseline Linear Regression model (R^2 of 0.821), confirming that advanced gradient boosting techniques are highly effective for this type of structured data problem.[1]
2. Influential Price Drivers: The feature importance analysis of the final XGBoost model provided clear insights into the factors that most significantly influence house prices in Ames, Iowa. The most critical features were found to be related to the overall quality and size of the property, such as OverallQual, the engineered TotalSF feature, and GrLivArea. Other key factors included the age of the house (HouseAge), garage size (GarageCars), and specific location (Neighborhood).[1]
3. Efficacy of Feature Engineering: The project demonstrated that thoughtful feature engineering adds significant value. Composite features like TotalSF and TotalBath ranked among the most important predictors, showcasing that combining raw features into more contextually meaningful variables can enhance model performance.

In essence, this project successfully built a predictive model that can explain over 91% of the variance in house prices, providing a robust tool that enhances transparency and reduces uncertainty in property valuation.[3]

6.2 Project Limitations

While the project achieved its objectives, it is important to acknowledge its limitations:

- Data Scope and Generalizability: The model was trained exclusively on data from a specific geographic location (Ames, Iowa) and a specific time period (2006-2010). As such, its predictions are highly contextualized, and the model would likely not generalize well to other cities or different economic climates without being retrained on relevant data.[1] Real estate markets are highly localized and dynamic.
- Feature Space: The dataset, though extensive, is limited to structured, tabular data. It lacks potentially valuable unstructured data sources, such as textual property descriptions or property images, which could capture nuanced details about a home's appeal. Furthermore, it does not include real-time macroeconomic indicators like mortgage interest rates, local crime statistics, or inflation rates, which are known to influence housing markets.[22]
- Model Interpretability: The best-performing model, XGBoost, is significantly more complex than Linear Regression. While it provides feature importance scores, it operates more like a "black box," making it difficult to interpret the precise nature of the non-linear relationships and interactions it has learned. This trade-off between accuracy and interpretability is a common theme in machine learning.

6.3 Future Perspective and Scope for Improvement

The findings and limitations of this project open several avenues for future research and development, aiming to create an even more accurate and adaptive prediction system.

- Advanced Feature Engineering: Future work could focus on incorporating unstructured data. Natural Language Processing (NLP) techniques could be used to extract features from property descriptions (e.g., identifying keywords like "newly renovated" or "granite countertops"). Similarly, Computer Vision models (e.g., Convolutional Neural Networks) could analyze property images to automatically assess features like curb appeal or the condition of interior spaces.[59]

- Real-time Data Integration: To overcome the static nature of the current model, a future system could be designed to integrate with real-time data feeds via APIs. This would allow the model to incorporate up-to-the-minute information on economic indicators (interest rates, unemployment), local development projects, or shifts in neighborhood demographics, ensuring that its predictions remain relevant and aligned with current market dynamics.[3]
- Exploration of Alternative Models: While XGBoost performed well, the field of machine learning is constantly evolving. Future iterations could explore other state-of-the-art algorithms such as LightGBM and CatBoost, which offer different optimizations for speed and handling of categorical features, or investigate deep learning architectures like neural networks for potentially capturing even more complex patterns.[1]
- Deployment and Scalability: A logical next step is to move from a research model to a deployed application. This would involve wrapping the trained model in a RESTful API to allow for programmatic access and building a user-friendly web interface where users could input property features and receive a price prediction. This deployed system would also require continuous monitoring to detect model drift and trigger periodic retraining to maintain accuracy over time.[2]
- Broader Economic Analysis: The model could be expanded to include macroeconomic variables. Incorporating factors like regional GDP growth, inflation, lending rates, and unemployment statistics would move the project towards a more comprehensive econometric model, capable of understanding not just micro-level property features but also the macro-level forces that shape the housing market.[22]

By pursuing these enhancements, the predictive analytics platform can become a truly transformative tool, fostering more efficient, transparent, and data-driven practices within the real estate industry.[3]

References

[1] ResearchGate. (2025).

House price prediction using machine learning.

2 Applied AI Course. (n.d.).

House Price Prediction using Machine Learning. Accessed September 2023.

3 International Journal of Research and Technology Innovation. (2025).

House Price Prediction System Using Machine Learning.

4 Pioneer Publisher. (n.d.).

A Comparative Study of Machine Learning Models for Predicting Housing Prices. Journal of World Economy.

5 Sathyabama Institute of Science and Technology. (n.d.).

House Price Prediction. Project Report.

6 Kaggle. (n.d.).

House Prices - Advanced Regression Techniques. Competition.

7 Ashydv. (n.d.).

Housing Price Prediction: Linear Regression. Kaggle.

8 Abdelhai, M. (n.d.).

EDA & Data Cleaning - Ames Housing Dataset. Kaggle.

9 Shahid, M. (n.d.).

EDA - Ames Housing Price Prediction. Kaggle.

10 Gondal, B. (n.d.).

Tutorial: Feature Engineering (Housing Data). Kaggle.

Works cited

1. (PDF) House price prediction using machine learning - ResearchGate, accessed August 12, 2025, https://www.researchgate.net/publication/379367913_House_price_prediction_using_machine_learning

2. House Price Prediction Using Machine Learning - Applied AI Course, accessed August 12, 2025, https://www.appliedaicourse.com/blog/house-price-prediction-using-machine-learning/

3. HOUSE PRICE PREDICTION - ijrti, accessed August 12, 2025, https://www.ijrti.org/papers/IJRTI2504245.pdf

4. Housing Price Prediction Using Machine Learning Algorithm, accessed August 12, 2025, https://www.pioneerpublisher.com/jwe/article/download/392/347

5. HOUSE PRICE PREDICTION USING MACHINE LEARNING - Sathyabama, accessed August 12, 2025, https://sist.sathyabama.ac.in/sist_naac/documents/1.3.4/1822-b.e-ece-batchno-120.pdf

6. House Prices - Advanced Regression Techniques - Kaggle, accessed August 12, 2025, https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

7. Housing Price Prediction (Linear Regression) - Kaggle, accessed August 12, 2025, https://www.kaggle.com/code/ashydv/housing-price-prediction-linear-regression

8. EDA + Data Cleaning (Ames Housing dataset) - Kaggle, accessed August 12, 2025, https://www.kaggle.com/code/abdelhai/eda-data-cleaning-ames-housing-dataset

9. EDA | AMES Housing Price Prediction - Kaggle, accessed August 12, 2025, https://www.kaggle.com/code/itsmohammadshahid/eda-ames-housing-price-prediction

10. Tutorial Feature Engineering Housing Data - Kaggle, accessed August 12, 2025, https://www.kaggle.com/code/bilalgondal786/tutorial-feature-engineering-housing-data