Deciphering House Price Dynamics Using Predictive Modeling on Housing Data

Allen Zou, Ting-An Lu, Zachary Chao, and Casey Hild

May 12, 2024

Abstract

Understanding the multifaceted dynamics of house price prediction in real estate markets is important for stakeholders involved in strategic decision-making. This research focuses on unraveling these complexities through predictive modeling using the Ames Housing dataset, aiming to identify the factors influencing house prices and develop a robust predictive model to estimate them accurately. Employing a systematic methodology, including decision tree modeling, forward feature selection, k-fold cross-validation, and grid search hyperparameter tuning, the research investigates the intricate interplay of various property characteristics. The findings reveal significant predictors of house prices, such as property size, amenities (like open porch square footage, above grade living area, and total basement square footage), and other features including the number of full bathrooms, garage size, and bedrooms above grade. While emphasizing practical insights for stakeholders, such as buyers, sellers, and investors, the study underscores the value of predictive modeling in strategic decision-making within the real estate domain.

1 Introduction

The real estate market is a complex ecosystem influenced by various factors, ranging from property characteristics to broader economic trends. Understanding the dynamics of house prices is essential for stakeholders involved in buying, selling, or investing in real estate. This research seeks to unravel the intricacies of house price dynamics through predictive modeling, offering insights that can inform strategic decision-making in the housing market.

In this study, we aim to address two fundamental research questions: What factors influence house prices, and can we construct a predictive model to estimate house prices based on these factors? By delving into these questions, we seek to unravel the complex interplay of factors shaping house price dynamics and explore the feasibility of developing a robust predictive model.

Based on the variables in the Ames Housing dataset, we hypothesize that certain factors such as location (i.e., the neighborhood variable, which describes the physical locations within Ames city limits), size of rooms (i.e., garage, basement, living area, etc. in square feet), and overall quality (i.e., categorical ratings for the aforementioned rooms such ”excellent,” ”good,” ”typical,” etc.), will have a significant impact on house prices. Specifically, they will have a positive relationship with house prices.

2 Dataset

The Ames Housing dataset used in this study was compiled by Dean De Cock and serves as a modernized and expanded version of the often-cited Boston Housing dataset [5]. It consists of two comma-separated values (CSV) files: train.csv and test.csv. The train.csv file comprises a tabular format with 1,460 rows and 81 columns, while the test.csv file contains 1,461 rows and 81 columns. Each row in both datasets represents a distinct property sale transaction, serving as a unique sample in the dataset.

The first column of each dataset is labeled ”Id” and contains a unique identifier for each property sale. The target variable, ”SalePrice,” is included as the last column, representing the sale price of each property. The features of each property sale transaction are captured in the columns between the first and last columns of the dataset.

The dataset encompasses a wide array of features that describe various aspects of residential properties. These features include categorical variables such as MSSubClass, which identifies the type of dwelling involved in the sale, and MSZoning, which indicates the general zoning classification of the sale. Additionally, numerical variables such as LotFrontage (linear feet of street connected to property) and LotArea (lot size in square feet) provide quantitative measurements of property characteristics. Other essential features cover aspects such as property configuration (LotShape, LotConfig), neighborhood information (Neighborhood), and conditions related to the property (Condition1, Condition2). Variables like OverallQual and OverallCond rate the overall material/finish and condition of the house, respectively, while variables like YearBuilt and YearRemodAdd provide information about the original construction and remodel dates.

Variable	**Description**
MSSubClass	Type of dwelling
MSZoning	Zoning classification
LotFrontage	Street connection length
LotArea	Lot size (sqare feet)
Street	Road access type
Alley	Alley access type
LotShape	Property shape
LandContour	Property flatness
Utilities	Available utilities
LotConfig	Lot configuration

Table 1: Example Variables and Descriptions

Providing detailed descriptions for each variable directly in the paper might lead to clutter. The Ames Housing dataset we utilized in this study includes a comprehensive data description file (data_description.txt), which contains detailed descriptions of all variables. Readers interested in understanding the specific definitions of features can refer to this file for more information. The dataset is publicly available on Kaggle, and interested readers can access it via the link in the bibliography section.

2.1 Data Pre-processing

The dataset initially comprised separate files for training and testing, with an unusual split of 1460:1461. To rectify this, the data were combined and shuffled, followed by an 80:20 train-test split. Due to the presence of numerous missing entries in at least half the rows and columns, preprocessing was necessary to ensure compatibility with scikit-learn, which does not support NaN values [8].

To address missing values, we first pruned columns with a substantial proportion of NaN values (approximately 20% or more). This step removed entries such as LotFrontage, Alley, FireplaceQu, PoolQC, Fence, and MiscFeature, resulting in a total of 73 out of 79 remaining columns (excluding SalePrice). One reason we chose to prune columns first was that no other column had anything close to 20% of its entries with NaN values. However, we chose not to prune rows because over half the data contained missing entries. Given that all entries had a unique ID and there are 2,335 entries, this would remove a substantial amount of our data.

For the remaining dataset, we used the IterativeImputer from scikit-learn to handle missing values in numerical columns. The IterativeImputer models each feature with missing values as a function of other features, then uses that estimate for imputation. This approach allows for more sophisticated and accurate filling of missing data compared to simpler methods like mean or median imputation. Categorical variables were imputed using the SimpleImputer with the strategy set to most_frequent, which replaces missing values with the most frequent value (mode) of each column.

Following imputation, the dataset was split back into training and testing sets. This preprocessing approach allowed us to retain 80% of the original dataset, providing a robust foundation for subsequent modeling tasks involving linear regression, decision trees, and support vector machines.

3 Methodology

In this section, we provide an overview of the methodologies used for model selection and evaluation, as well as the steps involved in developing reliable predictive models. More detail about how we applied these methodologies to select our model will be provided in Section 4 (Model Selection). It should be noted that all the strategies employed here involve the use of scikit-learn's packages [8]. We employ a systematic approach, beginning with an explanation of why we chose the decision tree model, followed by feature selection, k-fold cross-validation, and grid search hyperparameter tuning. Each subsection outlines a aspect of our methodology, covering forward feature selection, hyperparameter optimization, cross-validation, and model evaluation. However, before delving into these methodologies, we will discuss why decision trees were chosen as our primary modeling technique.

3.1 Decision Trees

Decision trees offer strengths that are beneficial to modeling our the Ames Housing dataset. One of their advantages is their capability to capture nonlinear relationships between predictors and the target variable, providing flexibility in modeling complex data [1]. Moreover, decision trees inherently rank predictors based on their importance in predicting the target variable, facilitating feature selection and offering insights into variable importance. Their high interpretability is another significant advantage, as decision trees provide intuitive visualizations of decision paths, making it easy to understand the model's decision-making process.

However, it's important to note that decision trees also come with notable weaknesses and how we address them. They are prone to overfitting, especially with deep trees that may capture noise in the data rather than true patterns. Additionally, decision trees can exhibit instability, as small variations in the data can lead to significantly different decision trees, impacting model robustness. Furthermore, decision trees produce piecewise constant predictions, which may not capture gradual changes in the target variable across predictor space. Despite these challenges, employing strategies such as grid search for hyperparameter tuning can effectively mitigate issues of overfitting and instability, ensuring the development of more reliable decision tree models.

3.2 Forward Selection Overview

Feature selection is aimed at identifying the subset of relevant features that contribute most to the model's predictive performance while reducing dimensionality and computational complexity [3]. In our methodology, we employed a forward feature selection approach to iteratively select features based on their impact on model performance.

The forward feature selection process begins with an empty set of selected features and iteratively adds one feature at a time, evaluating the model's performance at each step. At each iteration, the algorithm identifies the feature that results in the greatest improvement in model performance, as measured by a chosen evaluation metric. This metric, in our case, was the mean squared error (MSE), a common measure of regression model accuracy.

To evaluate the performance of each feature subset, we utilized k-fold cross-validation, a robust technique for estimating the model's performance on unseen data. In our implementation, we employed a 5-fold cross-validation strategy, partitioning the dataset into 5 equally sized folds, training the model on 4 folds, and evaluating it on the remaining fold [6]. Specifically, with 2,335 data entries in our dataset, this change results in each fold containing approximately 467 data entries for validation and 1,868 data entries for training. This process was repeated 5 times, with each fold serving as the validation set exactly once.

During each iteration of the forward feature selection process, we trained a decision tree regression model on the preprocessed training dataset using the selected features plus the feature under consideration. We then used grid search with cross-validation to tune the hyperparameters of the decision tree.

Normally, the feature selection process continued until adding additional features no longer resulted in a significant improvement in model performance or began to degrade performance, but for the purpose of visualization, we continued to run forward selection until it depletes all features.

Upon completion of the feature selection process, we stored the selected features, their corresponding mean squared errors, and the hyperparameters of the best-performing models each iteration for further analysis and model evaluation.

3.3 One-Hot Encoding

Before training the decision tree models, categorical variables were encoded using one-hot encoding [2]. This preprocessing step converts categorical variables into binary vectors, with each category represented as a binary feature. One-hot encoding is necessary for decision tree models because scikit-learn's decision trees don't natively accept strings from categorical variables. By converting categorical variables into a numerical format through one-hot encoding, decision trees can effectively process categorical features and capture their relationships with the target variable.

3.4 K-Fold Cross-Validation

K-fold cross-validation is a widely used technique to assess the performance and generalization ability of a predictive model. The main idea behind k-fold cross-validation is to partition the dataset into k subsets, or "folds," of approximately equal size [6]. The model is then trained k times, each time using k-1 folds as the training set and the remaining fold as the validation set. This process allows each data point to be used for validation exactly once. The performance metric, such as mean squared error (MSE) for regression tasks or accuracy for classification tasks, is computed for each fold, and the average performance across all folds is reported as the overall performance estimate of the model. K-fold cross-validation helps mitigate the risk of overfitting by providing a more reliable estimate of a model's performance on unseen data compared to a single train-test split. Common choices for the value of k include 5 or 10, although the choice may vary depending on the size and nature of the dataset.

Our decision to employ 5-fold cross-validation was driven by the characteristics of our preprocessed dataset, which comprises 2,335 rows and 73 columns. With a moderate-sized dataset and a relatively large number of features, 5-fold cross-validation strikes a balance between bias and variance in model evaluation. By partitioning the dataset into 5 equally sized folds, each fold contains approximately 467 samples, ensuring that the training sets remain sufficiently large for model fitting while still allowing for comprehensive evaluation across multiple iterations. This method further reduces the risk of overfitting by exposing the model to diverse subsets of the data, enhancing its robustness and reliability.

3.5 Optimizing Hyperparameters with Grid Search

In each iteration of the feature selection process, hyperparameters are optimized using scikit-learn's grid search to fine-tune the decision tree model [8]. Grid search is a systematic method for tuning hyperparameters by exhaustively searching through a predefined grid of parameter values and selecting the combination that yields the best model performance [2]. For the decision tree model, hyperparameters such as maximum depth, minimum samples split, minimum samples leaf, maximum features, and minimum impurity decrease are considered.

The grid search algorithm evaluates the model's performance using k-fold cross-validation as outlined in the previous subsection. By averaging the performance across all folds, grid search provides an estimate of the model's performance under various hyperparameter configurations. This process not only identifies the optimal hyperparameter values but also mitigates the risk of overfitting by ensuring that the chosen model parameters generalize well across different subsets of the data. The hyperparameter values that yield the lowest mean squared error (MSE) are selected as the optimal configuration for the decision tree model, ensuring that the model is both well-tuned and capable of generalizing to new data.

4 Model Selection

In the Model Selection section, we explore the application of methodologies described in the Methodology section to identify the optimal predictive model for our housing price prediction task. First, we detail the process of constructing a parameter grid tailored to our decision tree model. Subsequently, we elaborate on how we leveraged the outcomes derived from grid search cross-validation to select hyperparameters and a subset of features.

4.1 Grid Search Parameter Range

The param_grid dictionary for scikit-learn's grid search defines the range of values to explore during grid search. For each hyperparameter, such as max_depth, min_samples_split, min_samples_leaf, and min_impurity_decrease, we specified a list of potential values to test. We did not alter the options for max_features—['auto', 'sqrt', 'log2']—as these represent constant strings.

Due to the considerable computational time required for the forward selection process, we initially conducted a "dry" run with a smaller range of values for each hyperparameter. Increasing the value of k in k-fold drastically increases the computation time, so we sought to temporarily reduce the value of k for the "dry" run. Employing 2-fold instead of 10-fold cross-validation to reduce the computation time, we explored a broader range of values, including max_depth from 0 to 30 with increments of 5, min_samples_split and min_samples_leaf from 2 to 7 with increments of 1, and min_impurity_decrease from 0.1 to 0.7 with increments of 0.1.

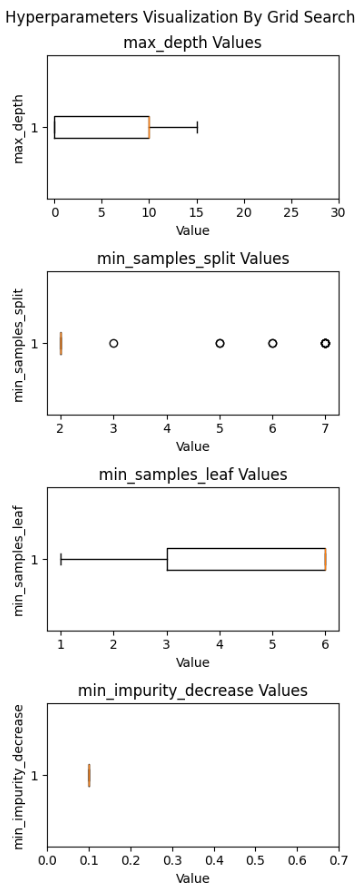

Figure 1: Hyperparameter Range for Grid Search

Based on the preliminary findings as depicted in Figure 1, we narrowed down the range of values to those between the 25th and 75th percentiles of performance for most variables. This visualization, generated from the "dry" run, provided insights into the distribution of model performance across different hyperparameter values. The plots for min_samples_split and min_impurity_decrease show a strange behavior where the range of values for almost every run is confined to a single value. Therefore, for these specific variables, we still increased the range of values around that single value. Subsequently, we selected the contracted ranges for each hyperparameter, including max_depth of 0, 5, and 10, min_samples_split of 2, 3, and 4 (we don't include the value of 1 because this variable requires at least 2 samples), min_samples_leaf of 1, 2, and 3, and min_impurity_decrease of 0.0, 0.1, and 0.2.

Once we determined the contracted ranges, we reinstated the k-fold cross-validation to 10 and executed a comprehensive grid search to identify the optimal hyperparameters for our decision tree model. This iterative process allowed us to strike a balance between computational efficiency and thorough exploration of hyperparameter space, ultimately leading to the selection of an effective predictive model.

4.2 Feature Subset and Hyperparameter Selection

This section illustrates how we visualize the model performance throughout the forward feature selection process using 5-fold cross-validation in each iteration of grid search and choose a model.

As described previously, the algorithm pinpoints the feature that yields the most substantial enhancement in model performance every iteration, gauged by MSE. Subsequently, based on the feature subset identified with the best-performing feature, we accumulate all the MSE values from the grid search on that particular model using the optimal feature subset. These aggregated MSE values form the basis for constructing a box and whisker plot, with each plot representing the MSE values from grid search on the best-performing feature subset for that iteratoin.

In Figure 2 (page 5), we present the box and whisker plots illustrating the MSE values of the model with the best-performing feature subset for all iterations of forward selection—comprising a total of 73 iterations, corresponding to the number of features in the preprocessed dataset. Notably, the initial iteration exhibits a markedly elevated box and whisker plot, indicating minimal improvement when incorporating a single feature. While the whiskers of nearly all other plots align approximately at the same lower bound, a pattern emerges in the mean of the box and whisker plots, notably at iteration 12, representing a subset of 12 features.

Given the challenge of interpreting the box and whisker plots in Figure 2, we provide a more focused view in Figure 3, zooming into iterations 8 to 16. Here, it becomes apparent that the box corresponding to iteration 12 is generally lower than those of neighboring plots, with the upper end of its whisker aligning closely with adjacent plots. Another observation is that the lower end of the whiskers for nearby plots is lower than the lower end of the whisker for iteration 12's plot. However, while these nearby whiskers technically fall below the lower whisker of iteration 12's plot, they do

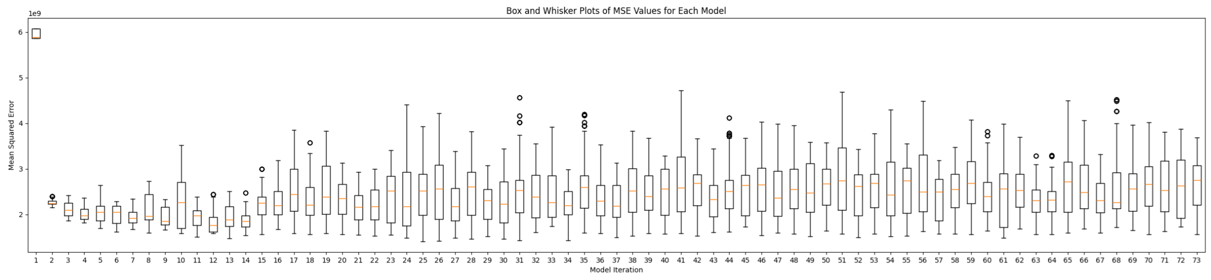

Figure 2: MSE values each iteration of grid search (full plot)

not drop substantially far down relative to it. All things considered, we opt for the model utilizing the subset of 12 features.

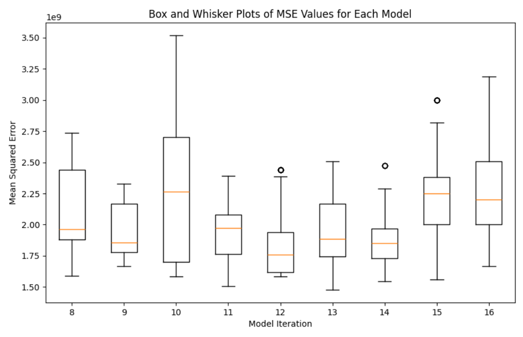

Figure 3: MSE values each iteration of grid search (partial plot)

Table 3: Selected Features

Feature	Description
OverallQual	House quality rating
GarageCars	Garage size in car capacity
Fireplaces	Number of fireplaces
BedroomAbvGr	Bedrooms above grade
Utilities	Available utilities
FullBath	Full bathrooms above grade
LandSlope	Property slope
2ndFlrSF	Second floor area
GarageType	Garage location
TotalBsmtSF	Total basement area
GrLivArea	Ground living area
OpenPorchSF	Open porch area

4.3 Final Hyperparameters and Features

Hyperparameter	Value
max_depth	10
max_features	'sqrt'
min_impurity_decrease	0.2
min_samples_leaf	2
min_samples_split	2

Table 2: Optimal hyperparameters for the decision tree model.

The hyperparameters listed in Table 2 represent the configuration determined by grid search to be the most optimal for the decision tree model chosen.

The selected features, as shown in Table 3, were determined through the forward selection algorithm employed during the model selection process. These features represent a subset of the original set of predictors and were iteratively chosen based on their contribution to improving the model's predictive performance.

5 Results

5.1 Performance Metrics

The performance of each model was evaluated using two common metrics: Mean Absolute Error (MAE) and Mean Squared Error (MSE). MAE measures the average magnitude of errors in predictions without considering their direction [2]. Conversely, MSE calculates the average of the squares of the errors, giving higher weights to larger errors and penalizing outliers more heavily.

After retraining the decision tree model with the hyperparameters shown in Table 2, we evaluated its performance on the testing dataset. The resulting Mean Squared Error (MSE) was 1,321,482,891, while the Mean Absolute Error (MAE) was 25,218. While the MSE provides a measure of the average squared difference between predicted and true values, the MAE offers a more interpretable metric, representing the average difference between predicted and true values. Given that house prices in the Ames Housing dataset are typically in the hundreds of thousands of dollars [5], an MAE of 25,218 indicates that the model's predictions are somewhat adequate.

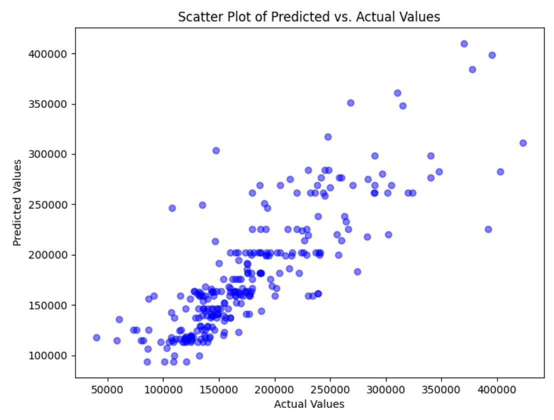

Figure 4: Scatterplot of predicted vs test values

However, further examination is warranted. Upon inspecting the scatterplot in Figure 4, which illustrates the relationship between predicted and true prices, while there's a positive correlation, the scatter suggests that the predictions lack precision, as evidenced by the somewhat scattered distribution of data points. Even though the Pearson's correlation coefficient, calculated to be 0.835, does indicate a strong positive linear relationship between predicted and true prices, the model's predictive accuracy may still be improved.

To recap our hypothesis, we anticipated that incorporating these factors into the decision tree model through forward selection, cross-validation, and grid search would enable accurate estimations of house prices. The results of this model somewhat align with this aspect of our hypothesis.

6 Interpretation and Analysis

In this section, we interpret the decision tree model's variables. While interpreting decision trees can provide valuable insights into the underlying patterns within the data, in our analysis, we chose not to visualize the decision tree. This decision was influenced by a technical constraint related to the handling of categorical variables within the scikit-learn library. As mentioned in earlier sections, scikit-learn's models do not handle categorical variables natively; they require encoding them in some manner [8].

6.1 Visualization Problem

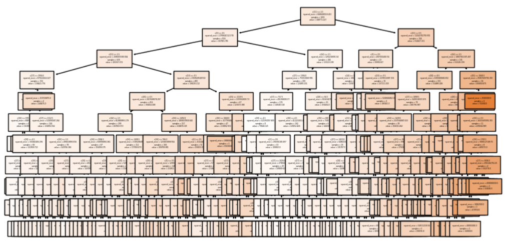

Figure 5: Visualization of Decision Tree

Because we utilized one-hot encoding to represent categorical variables, this approach results in the decision tree generating an extensive number of branches as shown in 5, rendering visualization challenging and interpretation cumbersome. Due to the sheer complexity and size of the resulting tree, attempting to visualize it would not provide clear or meaningful insights into the relationships between the predictor variables and the target variable. Therefore, instead of relying on visualizations, we examined the decision tree's feature importance to interpret it and extract actionable insights.

6.2 Feature Importances

Feature importance refers to the quantification of the relative contribution of each feature in a predictive model towards explaining the variability in the target variable [7]. It helps identify which features have the most significant influence on predictions, allowing for insights into the underlying relationships between input features and the target.

In this study, the importance value for each feature was computed using scikit-learn's decision tree implementation. The feature_importances attribute provides the importance of each feature in the decision tree model. This calculation is based on how much each feature contributes to reducing impurity (measured by Gini impurity) across all the nodes in the decision tree during the training process.

Impurity, in the context of decision trees, refers to the measure of uncertainty or disorder in a dataset. It quantifies how well a particular feature separates the data into classes. Gini impurity measures the probability of incorrectly classifying a randomly chosen element if it were randomly labeled according to the distribution of the classes in the node. It ranges from 0 to 0.5, where 0 represents perfect purity (all elements belong to the same class), and 0.5 represents maximum impurity (an equal distribution of classes). In decision tree algorithms, the goal is to minimize impurity when splitting nodes, meaning the feature that results in the greatest reduction in impurity (i.e., the feature that best separates the classes) is selected for splitting.

6.2.1 One-Hot Encoding Aggregation

When dealing with one-hot encoded features in the context of feature importance analysis, it's important to consider the aggregation of importance values back to the original categorical features [2]. One-hot encoding transforms categorical features into binary features, which can complicate the interpretation of feature importance.

Aggregating the importance values of the one-hot encoded features back to their original categorical features allows for a clearer understanding of the importance of the original categorical variables in the model. This process involves summing up the importance values of all binary features corresponding to each original categorical feature. By doing so, we can prioritize the impact of the categorical variables themselves rather than individual binary representations.

6.2.2 Interpreting Feature Importances

The feature importance table (Table 4) displays the feature importance of each variable. Importance values are expressed as percentages. For reference, ”SF” stands for ”square feet.”

The feature importance table derived from the decision tree model offers insights into the factors that significantly influence housing prices. According to the model, the most influential feature is ”Open Porch SF,” representing the square footage of open porch space, accounting for nearly half of the predictive power. Following this, ”Above Grade Living Area” and ”Total Basement SF” are also significant predictors, representing the living area and basement size respectively. Other features such as ”Full Baths” and ”Garage Cars” contribute to a much lesser extent. Interestingly, features like ”Overall Quality” and ”Utilities” show negligible importance, suggesting that they have minimal impact on housing prices in this model.

Table 4: Feature Importances from Decision Tree

Feature	Importance
Open Porch SF	48.74%
Above Grade Living Area	9.18%
Total Basement SF	7.44%
Full Baths	1.14%
Garage Cars	1.04%
Bedrooms Above Grade	0.25%
Second Floor SF	0.24%
Fireplaces	0.15%
Overall Quality	0.00%
Utilities	0.00%
Land Slope	0.00%
Garage Type	0.00%

7 Discussion

Interestingly, while our hypothesis anticipated that factors such as location, size of rooms, and overall quality would play pivotal roles, the model's findings provide a more nuanced perspective.

Specifically, variables such as ”Open Porch SF,” ”Above Grade Living Area,” and ”Total Basement SF” correspond to property characteristics, reflecting the importance of the size and amenities of the dwelling in determining its value. While the model did not highlight factors like ”Overall Quality” and ”Utilities” as significant predictors, this may suggest that other features, such as number of baths and number of cars that can fit in the garage, have a stronger influence on housing prices in this context. Nonetheless, the insights gleaned from this analysis enable a deeper understanding of the intricate interplay between various property characteristics and housing prices, aiding in refining our initial hypotheses and informing future research and decision-making processes in the real estate domain.

8 Research Limitations and Future Directions

While our study provides valuable insights into predicting house prices using machine learning models, several limitations and avenues for future research should be considered.

Firstly, it's important to recognize that Dean De Cock's dataset is specific to residential homes in Ames, Iowa. While the findings offer insights into housing market dynamics in this locality, generalizing the results to other geographical areas should be done cautiously. The characteristics of the Ames housing market, such as demographics, economic conditions, and housing policies, may differ significantly from other regions, impacting the relevance and applicability of the findings. Additionally, housing markets vary in terms of demand-supply dynamics, cultural preferences, and regulatory environments, which can influence property prices and market trends differently. Therefore, while our analysis provides valuable insights, it's essential to exercise caution when extrapolating findings to other housing markets or datasets.

Moreover, we acknowledge the limitations of geographic specificity. Some potential directions for validating the model with diverse datasets from different locations include external validation with datasets from other housing markets and exploring ways to adapt the model to different contexts.

Additionally, the peculiarities of the train-test split in our dataset may warrant reshuffling and redistributing the data or cross-validation for more robust model evaluation.

Secondly, given that only a subset of variables, such as specific areas within the house and overall quality, significantly influences the models, exploring dimensionality reduction techniques like Principal Component Analysis (PCA) could be beneficial. PCA can help identify latent variables and reduce the dimensionality of the feature space while preserving the most critical information, potentially improving model performance and interpretability [4].

9 Conclusion

In this study, we embarked on a journey to decipher the intricate dynamics of house prices by employing predictive modeling techniques on the Ames Housing dataset. Through preprocessing, feature selection, and model evaluation, we aimed to uncover the key factors influencing house prices and develop an accurate predictive model.

Preprocessing involved addressing missing data and ensuring compatibility with machine learning algorithms, while feature engineering encompassed selecting relevant predictors and transforming variables to enhance predictive power. We employed techniques such as forward feature selection, k-fold cross-validation, and grid search hyperparameter tuning to iteratively refine our model and optimize its performance. By systematically evaluating various models and configurations, we sought to identify the most influential predictors and construct a predictive framework capable of accurately estimating house prices.

We sought to identify the most influential predictors and construct a predictive framework capable of accurately estimating house prices. Despite our initial anticipation that traditional factors like overall quality and utilities would have significant impacts on housing prices, our findings revealed a more nuanced interplay between different property features and their influence on pricing dynamics. Certain property characteristics, such as the size of open porch space, above-grade living area, and total basement square footage, play pivotal roles in determining housing prices while traditional factors like overall quality and utilities showed minimal importance in our model.

Additionally, several limitations and avenues for future research remain. Addressing data loss due to missing entries, exploring advanced imputation techniques, and generalizing findings to other geographical areas are important steps for enhancing the robustness and applicability of predictive models. Furthermore, the exploration of dimensionality reduction techniques and the incorporation of additional datasets or features could offer further improvements in model performance and interpretability.

In conclusion, our study contributes to advancing the understanding of housing market dynamics and provides valuable insights for stakeholders involved in real estate decision-making. By leveraging predictive modeling techniques and data-driven approaches, we can empower individuals and organizations to make informed decisions in the dynamic and complex realm of real estate.

References

[1] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Taylor Francis, 1984.

[2] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[4] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[5] Kaggle. Ames Housing Dataset. https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data, 2016. Accessed: May 8, 2024.

[6] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, 2013.

[7] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Leanpub, 2023.

[8] Scikit-learn Contributors. Scikit-learn documentation, n.d.