Toulouse School of Economics

Final Project: Supervised Regression on Housing Dataset

Master 2 Data Science for Social Sciences

Emma Bastien, Coralie Sorbet

Teacher: Sébastien Gadat

Academic Year: 2024-2025

Contents

1 Introduction

The primary objective of this project is to predict housing prices using supervised regression techniques. Accurate prediction of housing prices is crucial for stakeholders such as real estate professionals, potential buyers, and policymakers. By employing statistical models and machine learning approaches, this project aims to provide robust and interpretable predictive insights, balancing accuracy and interpretability.

The dataset used for this analysis contains detailed housing information, including features such as the number of bedrooms, square footage, and neighbourhood characteristics. The target variable is the housing price, which is the model's dependent variable. With 2930 records and 80 features, the dataset provides a rich source of information for analysis. It was sourced from Kaggle, and its structure allows for comprehensive exploration and model development.

This report is structured to provide a logical flow through the project stages. The next section discusses the data preparation process, which includes handling missing values, addressing outliers, standardizing features, and applying feature engineering techniques to enhance predictive power. This explains the data-splitting strategy and resampling methods used to ensure reliable model evaluation. The core of the report focuses on regression methodologies, starting with statistical models such as linear regression and extending to machine learning techniques like Random Forest, Gradient Boosting, and Support Vector Regression (SVR).

Optimization techniques are discussed in a dedicated section, exploring strategies like Gradient Descent, hyperparameter tuning, and Monte Carlo cross-validation. Finally, the report evaluates the models' performance, comparing their predictive accuracy and interpretability using key metrics. By combining meticulous data preparation, diverse modelling approaches, and advanced optimization techniques, this project aims to provide a comprehensive solution for predicting housing prices.

2 Data Preparation and Cleaning

Data preparation and cleaning are crucial steps in ensuring the quality and reliability of the models. In this section, we address several key tasks to enhance the predictive power of the dataset. Each of these steps was performed to optimize the dataset for use in the regression models, ensuring that the final models produce accurate and interpretable results.

2.1 Handling Missing Values

First, in order to ensure that our models were as accurate as possible, we treated the missing values cases in our dataset.

When first inspecting for missing values in our dataset, we found that was a quite high percentage of missing values amongst 27 out of our 80 variables. However, when inspecting more the content of those variables, a lot of them were not actually NAs but corresponded to characteristics that specific houses did not have. For instance, for the variable "Garage Type", an "NA" value would actually correspond to a house that does not have a garage. So, after doing several checks on those variables, we concluded that for all those cases, no imputation was needed and we thus only re-encoded those modalities.

After this re-encoding, we still had one last variable containing almost 17% of actual missing values. To deal with this variable, we implemented a k Nearest Neighbors (kNN) imputation

consisting of imputing based on a notion of distance between individuals (here we chose k = 5 neighbours).

Consequently, our dataset after all these preprocessing steps was "missing value free".

2.2 Handling Outliers

Now, detecting and treating outliers in each one of our variables was also another important task. Indeed, for our models to be of good quality and to get good insights from it, we had to treat some of them because, for instance, in linear regression, outliers can distort the regression line, thus reducing predictive accuracy.

To do so and have a first idea of how many outliers we had at hand, we plotted a boxplot for each one of our variables.

For some of them, it was visually obvious that there were a lot of outliers needing to be treated, for instance for the variable **Lot Frontage**:

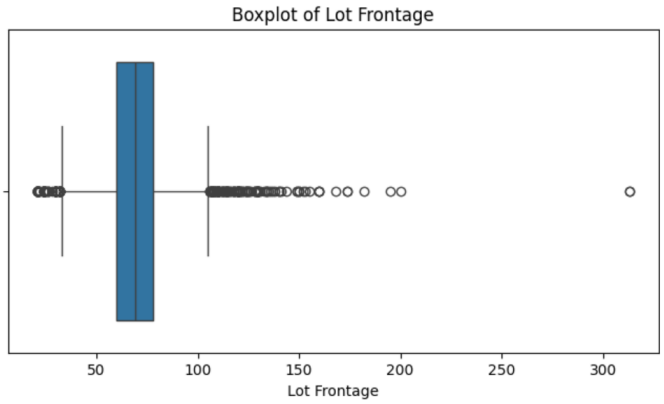

Figure 1: Boxplot of the Lot Frontage variable

However, for some variables having a boxplot like the one below (Pool Area), we made the choice of not treating their outliers. Indeed, in this example, the points outlined as outliers in the boxplot are not outliers but constitute the only houses that have a pool. There are so few houses with a pool in the dataset that we could think those houses are outliers when they are not:

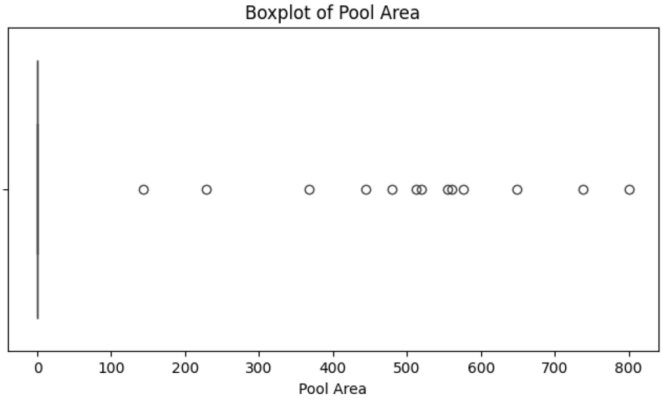

Figure 2: Boxplot of the Pool Area variable

Thus, to treat the outliers of the variables of the first type presented above, we used a statistical approach called winsorization. Instead of removing those outliers and thus losing a lot of

information, we replaced their values with the nearest values that are within a specified percentile range (here we only replaced the distributions' upper-end outliers since they were nearly no outliers in the lower-end of the distribution). All the values above the 97.5th percentile of the variable considered were replaced with the value at the 97.5th percentile.

For instance, for the **SalePrice** variable, the 97.5th percentile is equal to 389,475 dollars, thus all the houses more expensive than this value were replaced by 389,475.

2.3 Standardization of the Dataset

The dataset was standardized to ensure that all variables were on the same scale. Indeed, initially, there were significant discrepancies in the scales of the different variables. This step was essential because certain models, such as Lasso regression, are sensitive to the scale of the features. Without standardization, variables with larger ranges could dominate the optimization process, leading to biased results. By scaling the data to have a mean of 0 and a standard deviation of 1, we ensured that all features contributed equally to the model, improving the performance and reliability of models like Lasso regression.

2.4 Feature Engineering

Feature engineering plays a critical role in preparing the dataset for modelling by transforming raw data into meaningful input for machine learning algorithms. In this project, we applied one-hot encoding to the categorical variables to create dummy variables. The first dummy variable for each categorical feature was dropped to avoid multicollinearity, a common issue in regression models that occurs when two or more predictors are highly correlated, which can distort the estimation of model coefficients.

We also removed variables that were highly correlated (greater than 0.85) to prevent multicollinearity and improve the stability and interpretability of the models. Removing highly correlated features helps in reducing redundancy, which can lead to better model performance and more reliable results.

3 Resampling and Data Splitting

3.1 Resampling Strategy

In supervised regression tasks, resampling techniques are essential for obtaining reliable performance estimates and ensuring that the model generalizes well to unseen data. These methods help mitigate overfitting by training and testing the model on different subsets of the data, providing a more stable and unbiased evaluation.

For all models, we employed cross-validation (CV) to optimize the hyperparameters. Specifically, for the Lasso model, we used 5-fold cross-validation to determine the optimal value of the regularization parameter λ. This technique splits the data into 5 subsets, allowing the model to be trained and tested on each fold, ensuring a balanced evaluation of performance and minimizing the risk of overfitting by utilizing all data points for both training and testing.

For the other models, we first build them using cross-validation. We then tried to improve the model's performance and fine-tune the settings using grid search combined with a 5-fold cross-validation. This approach ensured a comprehensive exploration of possible parameter configurations and robust performance evaluation.

At the final stage of the project, we used Monte Carlo (MC) cross-validation to compare the performance of all models. Unlike k-fold cross-validation, which divides the data into a fixed number of subsets, MC cross-validation involves multiple random splits, offering greater flexibility. While k-fold ensures that each data point is used for both training and testing, it can be limited by the structure of the folds, potentially resulting in less diverse evaluations. In contrast, MC cross-validation generates a wide variety of training and testing sets, which provides a more comprehensive assessment of model performance across different data partitions. This flexibility allows for a more robust evaluation, as it is less sensitive to the specific choice of splits, leading to a more reliable estimate of the model's generalization ability. By using MC cross-validation, we ensured that our final model selection was based on a broader set of training/testing scenarios, enhancing the overall robustness of the results.

3.2 Data Splitting

For the data splitting process, we first define the target variable Y as the sale price ('SalePrice') and the feature matrix X as all the other variables in the dataset. To assess the model's performance, we split the data into training and testing sets. We used a train-test split with a test size of 0.2 (i.e., 20% of the data is reserved for testing), ensuring that the random state is set to 42 for reproducibility.

For certain models (Lasso regression, Gradient Boosting and SVR), which are sensitive to the scale of the features, we apply **standardization** to ensure all features have the same scale. StandardScaler is used to normalize the feature matrix X and is only performed on the dataset intended for models like Lasso, where feature scaling is required.

4 Regression Methodologies: Statistical and Machine Learning Approaches

This section provides an overview of the regression algorithms used in this analysis, combining statistical / econometrics methods for interpretability, that could be easily understood by a statistician (linear regression, lasso regression) and machine learning approaches for improved accuracy (random forest, gradient boosting, and support vector regression).

4.1 Statistical Modeling

4.1.1 Linear Regression Model

We first approach our problem of predicting house prices by using statistical regression models. The primary model is the ordinary least squares (OLS) linear regression that assumes a linear relationship between the predictors and the response variable, here being the price of the houses.

The model is defined as follows:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \epsilon, \tag{1}$$

where:

- y: The dependent variable (house price in our context).
- β_0: Intercept, the predicted value of y when all $x_i = 0$.

- β_i: Coefficients representing the contribution of each predictor x_i.
- ϵ: The residual error, is assumed to follow a normal distribution.

To estimate the betas coefficients, the method of OLS aims at minimizing an objective/loss function of the sum of squared residuals:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

and the solution in matrix form is defined as :

$$\hat{\beta} = (X^T X)^{-1} X^T Y.$$

Finally, assumptions of the model that will need to be checked in our results include:

- Linearity of relationships.
- Homoscedasticity (constant variance of errors).
- Normality: needed to have validity of inferential statistics, such as hypothesis testing and confidence intervals, for the estimated coefficients.
- No multicollinearity among predictors.

Residual analysis will be performed to confirm these assumptions.

Linear regression was chosen for this dataset due to its simplicity and interpretability, providing a clear framework to understand how predictors influence house prices through straightforward coefficients. It will also serve as a baseline model to compare its results to more complex algorithms.

4.1.2 Linear Regression Model with Lasso Penalty

To enhance the performance of linear regression, we incorporate regularization through the Lasso method. Lasso modifies the ordinary least squares (OLS) linear regression by adding a penalty term to the objective function, which encourages sparsity in the coefficients.

The model is still defined as follows:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \epsilon, \tag{2}$$

where:

- y: The dependent variable (house price in our context).
- β_0: Intercept, the predicted value of y when all $x_i = 0$.
- β_i: Coefficients representing the contribution of each predictor x_i, some of which may be set to zero by the Lasso penalty.
- ϵ: The residual error.

To estimate the β coefficients, Lasso minimizes a modified loss function that combines the OLS residual sum of squares with an ℓ_1-norm penalty:

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

The parameter λ controls the strength of the penalty where:

- Larger values of λ increase regularization, shrinking more coefficients to zero and simplifying the model.
- Smaller values of λ reduce the penalty, making Lasso behave more like standard OLS regression.

Finally, assumptions of the underlying linear regression model are still relevant ie.:

- Linearity of relationships.
- Homoscedasticity (constant variance of errors).
- Normality: needed to have validity of inferential statistics, such as hypothesis testing and confidence intervals, for the estimated coefficients.
- No multicollinearity among predictors.

and where residual analysis will also be performed to confirm these assumptions.

Lasso regression was chosen for this dataset due to its ability to handle high-dimensional data (indeed here, we have a lot of variables at hand) that simplifies the model by setting some coefficients to zero. This method reduces the risk of overfitting and helps identify the most relevant predictors of house prices. Additionally, it provides a balance between model complexity and predictive performance, making it a robust choice for improving the baseline linear regression model.

4.2 Machine Learning Approaches

On the other hand, machine learning models prioritize predictive power over interpretability. For this dataset, we implemented tree-based models such as random forest and gradient boosting and a support vector regression.

4.2.1 Random Forest

Random Forest (RF) is an ensemble learning method commonly used for both classification and regression tasks. It works by creating several decision trees, each trained on a different subset of the data. These trees are built in parallel, which helps reduce the model's variance and makes it less prone to overfitting. While individual trees might overfit, averaging their predictions (for regression) or using majority voting (for classification) helps smooth things out. Unlike Gradient Boosting, which focuses on minimizing a loss function, Random Forest improves accuracy by reducing variance, making it more stable and reliable than a single decision tree.

Mathematically, Random Forest can be expressed as:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x), \tag{3}$$

where:

- $\hat{y}$ is the predicted output,
- T is the number of trees in the forest,
- $f_t(x)$ is the prediction from the t-th tree for the input vector x.

Each tree in the forest is trained on a bootstrap sample (random sampling with replacement) from the original training data. Additionally, at each node, a random subset of features is considered for the split. This randomness helps reduce variance, preventing overfitting and makes the model robust, even when dealing with a large number of features.

Random Forest is especially suitable for datasets with numerous features and complex, nonlinear relationships, making it an excellent choice for predicting housing prices in this dataset. Since the dataset includes both numerical and categorical variables, as well as potential complex interactions among them, Random Forest can model these relationships effectively without requiring explicit feature engineering. Furthermore, Random Forest is resistant to overfitting and works well even when data contains noisy or missing values.

This method was chosen because it excels in high-dimensional spaces, handles nonlinearity, and is robust to overfitting. Additionally, it provides insights into feature importance, which can help identify the most influential variables in predicting housing prices.

Unlike linear models, Random Forest is non-parametric and does not rely on assumptions about the data distribution, linearity, or normality. It is also less sensitive to outliers and does not require any transformation of the data. However, while it can handle correlated features, performance is typically better when the predictors are not excessively correlated, as high correlation may lead to overfitting. Random Forest also requires a sufficiently large sample size and a sufficient number of trees to maintain its robustness and avoid both overfitting and underfitting.

4.2.2 Gradient Boosting

Gradient Boosting is an ensemble learning method that builds a model by combining multiple weak learners, typically decision trees, in a sequential manner. Each new tree is trained to correct the errors (residuals) of the previous trees, iteratively improving the model's predictive power. Unlike Random Forest, which constructs trees in parallel, Gradient Boosting builds trees sequentially. This approach can be more prone to overfitting but, when tuned correctly, it can yield a highly powerful model.

The model is trained to minimize a loss function L, commonly the mean squared error for regression tasks:

$$L(\hat{y}) = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2, \tag{4}$$

where y_i represents the actual value and $\hat{y}_i$ is the predicted value.

In Gradient Boosting, trees are fitted to the residuals (i.e., the difference between observed and predicted values) from previous trees. The key idea is to minimize these residuals iteratively by adding new decision trees. This iterative process can be expressed mathematically as:

$$\hat{y}_m = \hat{y}_{m-1} + \eta \cdot f_m(x), \tag{5}$$

where:

- $\hat{y}_m$ is the prediction after the m-th iteration,
- $\hat{y}_{m-1}$ is the prediction after the $(m-1)$-th iteration,
- $f_m(x)$ is the new weak learner (tree) added at the m-th iteration,
- η is the learning rate, which controls the contribution of each tree to the final model.

This process continues until a predefined number of trees is reached or the error no longer improves.

Gradient Boosting is particularly effective for regression tasks such as predicting housing prices because it captures complex nonlinear relationships between features. It excels in scenarios where interactions between features are crucial, easily handling both categorical and numerical data. Gradient Boosting fine-tunes the model by iteratively minimising residuals, delivering more accurate predictions, especially for difficult cases where other models might fail.

This method was chosen for this dataset due to its high accuracy and flexibility. Gradient Boosting excels when prediction accuracy is more important than model interpretability. It is less prone to overfitting than other boosting methods, particularly when combined with regularization techniques like early stopping and cross-validation.

Similar to Random Forest, Gradient Boosting is a non-parametric ensemble method, but it builds trees sequentially. It imposes fewer assumptions than linear models like Linear Regression or Lasso, but there are some important assumptions about the data:

- **No Assumptions on Data Distribution:** Gradient Boosting does not require the data to follow any specific distribution. It can handle various data distributions effectively.
- **Independence:** The observations are assumed to be independent.
- **Weak Learners:** It assumes that the weak learners (trees) are sufficiently simple and that each tree can correct the errors made by the previous trees in the sequence.
- **Sensitivity to Outliers and Noise:** Gradient Boosting can be sensitive to outliers and noise in the data, meaning that it may perform better with preprocessed or cleaned data.
- **Feature Importance:** Gradient Boosting assumes that the features contribute meaningfully to predicting the target. However, it does not require the features to be highly correlated with one another.

4.2.3 Support Vector Regression (SVR)

Support Vector Regression (SVR) is a robust regression technique that extends Support Vector Machines (SVM) to predict continuous values. SVR aims to find a function that approximates the relationship between the predictors and the target variable while allowing for a margin of tolerance, (ϵ), around the predicted values.

The principle can be explained with the following illustration :

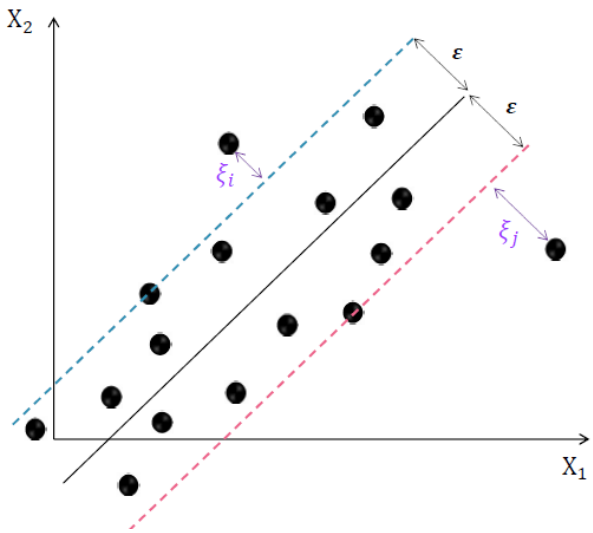

Figure 3: Illustration of the SVR principle (taken from Gil Casals, 2021)

Above, the idea is to fit a function such that the deviations between the actual and predicted values within a certain threshold (ϵ) are considered acceptable, meaning the model does not penalize these deviations. Points lying inside the "ϵ tube" are considered as correctly predicted and are thus not penalized while those lying outside are incorrectly predicted and hence contribute to the loss function. Consequently, SVR only uses a subset of the data points (support vectors) to define the optimal model. These support vectors are the points that lie outside the ϵ tube or on its boundaries and are the only points that affect the model's outcome. Indeed, as explained before, points inside the epsilon tube do not affect the model because their error is small enough to be tolerated.

The model is defined as follows:

$$y = \mathbf{w}^T \mathbf{x} + b, \tag{6}$$

where:

- y: The dependent variable (house price in our context).
- $\mathbf{w}$: The weight vector determines the hyperplane's orientation.
- $\mathbf{x}$: The feature vector (predictors).
- b: The bias term, shifting the hyperplane.

SVR optimizes the following loss function:

$$\min_{\mathbf{w},b,\xi_i,\xi_i^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i, \tag{7}$$

subject to:

$$y_i - (\mathbf{w}^T \mathbf{x}_i + b) \leq \epsilon + \xi_i, \quad (\mathbf{w}^T \mathbf{x}_i + b) - y_i \leq \epsilon + \xi_i^*, \quad \xi_i, \xi_i^* \geq 0. \tag{8}$$

Here:

- ϵ: The tolerance margin around the predicted value. Predictions within this margin are not penalized.
- ξ_i, ξ_i^*: Slack variables representing deviations from the ϵ-tube for data points outside the margin.
- C: Regularization parameter that controls the trade-off between margin width and tolerance for errors.

Key advantages of SVR include:

- Flexibility because SVR can model non-linear relationships by using kernel functions (e.g., radial basis function (RBF) kernel), which map the data into a higher-dimensional space where linear regression can be applied. However, in our case, we will see that using a linear kernel will suffice.
- Robustness by focusing only on data points near the decision boundary (support vectors), SVR is less sensitive to outliers.

SVR was chosen for this dataset because it provides flexibility to model complex relationships between predictors and house prices, making it a powerful alternative to linear regression. Its ability to balance model complexity (via ϵ and C) while maintaining robustness to outliers makes SVR a suitable method for predicting house prices with high accuracy.

5 Optimization Techniques

This section outlines the optimization strategies employed during model training. It focuses on mathematical principles, hyperparameter tuning, and evaluation techniques to ensure optimal model performance.

5.1 Hyperparameter Tuning

Hyperparameter tuning is about finding the best settings for parameters that aren't learned during training like regularization strength, tree depth, or learning rate to improve the model's performance. These parameters play a key role in controlling the learning process, and choosing them carefully helps the model strike a balance between underfitting and overfitting.

In this study, we used a **grid search strategy** to explore a range of possible hyperparameter values. The process is simple: define a grid of candidate values, train the model for each combination, and evaluate its performance on a validation set. To measure performance, we minimized the Root Mean Squared Error (RMSE), defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{9}$$

While grid search systematically tests all combinations, it can be computationally expensive when there are many parameters to tune. We have also combined the grid search with k-fold cross-validation to identify the optimal set of parameters while ensuring robust model evaluation.

By tuning the hyperparameters effectively, we ensured that the model achieves both accuracy and the ability to generalize well to unseen data, which is essential for robust predictions.

5.2 Monte Carlo Cross-Validation

Monte Carlo Cross-Validation (MCCV) was employed to evaluate the model's performance by repeatedly splitting the dataset into training and validation sets. Unlike k-fold cross-validation, where the splits are deterministic and systematic, MCCV generates multiple random partitions of the data. For each split, the model is trained on the training set and its performance is evaluated on the corresponding validation set.

The process is repeated multiple times, and the average performance metric, such as the Root Mean Squared Error (RMSE), is computed over all iterations. By averaging across several random splits, MCCV provides a robust and reliable estimate of the model's generalization error. This approach reduces the risk of performance estimates being overly influenced by a particular train-test split, making it particularly effective for assessing model stability and accuracy.

Furthermore, the flexibility of MCCV allows for greater control over the size of the training and validation sets, enabling us to adapt the resampling strategy to the specific requirements of the dataset. This ensures that the evaluation reflects the true predictive capability of the model when applied to unseen data.

5.3 Mathematical Optimization Challenges

The optimization problems addressed in this project focused on ensuring efficient and reliable model training by tackling key challenges.

A primary consideration was the **convexity** of the loss function, such as the mean squared error (MSE). Convexity ensures that optimization algorithms, particularly **Gradient Descent**, can reliably converge to a global minimum. Gradient Descent is an iterative optimization algorithm that minimizes a function by updating its parameters in the opposite direction of the gradient (slope) of the function regarding the parameters. For a given loss function $J(\beta)$, the parameter update rule can be expressed as:

$$\beta^{(t+1)} = \beta^{(t)} - \eta \nabla J(\beta), \tag{10}$$

where η is the learning rate, controlling the step size, and $\nabla J(\beta)$ is the gradient of the loss function. The choice of an appropriate learning rate is critical: if it is too large, the algorithm may overshoot the minimum or fail to converge; if it is too small, convergence can be excessively slow. In this project, proper tuning of the learning rate ensured stable and efficient convergence.

We also employed **regularization** techniques to mitigate overfitting, particularly in Lasso Regression. Regularization involves adding a penalty term to the loss function to control the complexity of the model. For Lasso, this is achieved through the L_1-norm:

$$J(\beta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \|\beta\|_1, \tag{11}$$

where λ determines the strength of the penalty. By driving less important coefficients toward zero, Lasso promotes sparsity in the model, which improves interpretability and generalizability.

Lastly, we carefully monitored **convergence**. For Gradient Descent and other iterative methods, ensuring convergence requires addressing challenges such as vanishing gradients, particularly in complex models. To mitigate these issues, we implemented proper initialization techniques and monitored the change in loss values over iterations, ensuring that the optimization process remained on track.

6 Results and Evaluation

6.1 The metrics used

To assess the results of our 5 models, we consider 5 different metrics: the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), the coefficient of determination R^2 and its adjusted version and finally the percentage of error, also called Mean Absolute Percentage Error (MAPE). They are more precisely defined as follows :

- **RMSE** is calculated as the square root of the average of the squared differences between predicted and actual values. It evaluates the model's prediction accuracy, expressed in the same units as the target variable. A lower RMSE indicates a better model fit :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{12}$$

  where, $\hat{y}_i$ is the predicted value and y_i is the actual value.

- **MAE** calculates the average absolute difference between predicted and actual values, with lower values indicating better accuracy :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{13}$$

- R^2 reflects the proportion of variance in the target variable explained by the model, with higher values indicating better explanatory power.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{14}$$

- Adjusted R^2 accounts for the number of predictors in the model, penalizing the addition of variables that do not improve model performance.

$$\text{Adjusted } R^2 = 1 - \left(1 - R^2\right) \cdot \frac{n-1}{n-p-1} \tag{15}$$

  where,

n is the number of data points (sample size),

p is the number of predictor variables (independent variables),

R^2 is the coefficient of determination.

- **Percentage of error** evaluates the relative error size in predictions compared to actual values. Lower percentages indicate better performance.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100 \tag{16}$$

Amongst those metrics, some are more adapted than others depending on the model we consider.

For instance, R^2 and the adjusted R^2 are the most adapted for Linear Regression and Linear Regression with Lasso while more complex machine learning algorithms like Random Forest, Gradient Boosting and SVR accuracies would be better assessed with RMSE or MAE.

6.2 Models Results

6.2.1 Linear Regression Model

Our linear regression model, aiming at predicting our dependent variable Y, the sale price of the houses, used as explanatory variables all of the other variables contained in our dataset.

For our very first baseline linear regression model, we obtained quite satisfying results with a coefficient of determination equal to 0.90 and an adjusted one equal to 0.73. However, this very high value of R^2 suggests that the model explains a large portion of the variance, but the substantial drop to an adjusted R^2 of 0.73 suggests that some variables may not be genuinely contributing to the model's predictive power and that overfitting might be happening.

However, all of the assumptions needed to be checked are indeed verified so that the use of this model is still adapted. We have that :

- Scatterplots of residuals vs. fitted values show no clear patterns, confirming the linearity assumption :

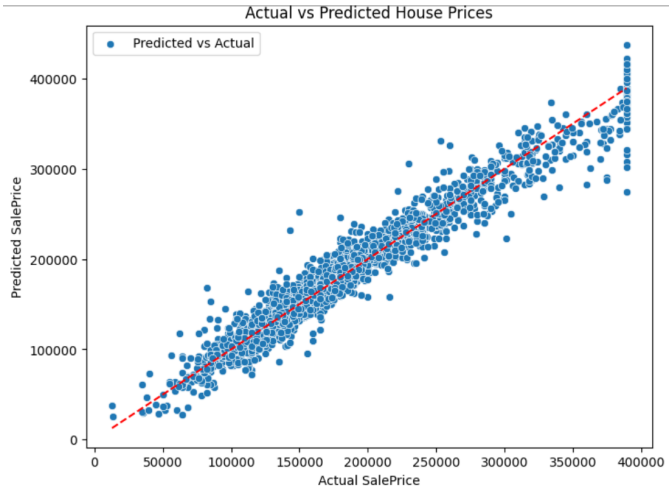

Figure 4: Scatterplot of actual vs predicted house prices with Linear Regression

- Residual Q-Q plots indicate approximate normality :
- And the variance of residuals remains roughly constant across fitted values.

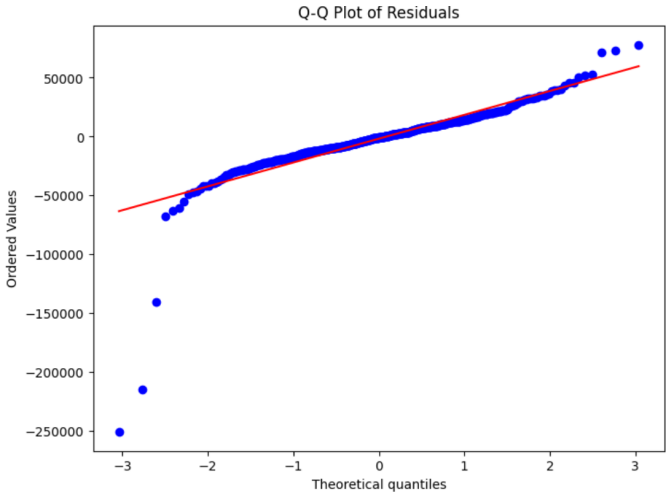

Figure 5: QQ plot of Linear Regression's residuals

As to go into more detail about which predictors contribute the most to house prices, let us now show the top 20 most important ones as follows:

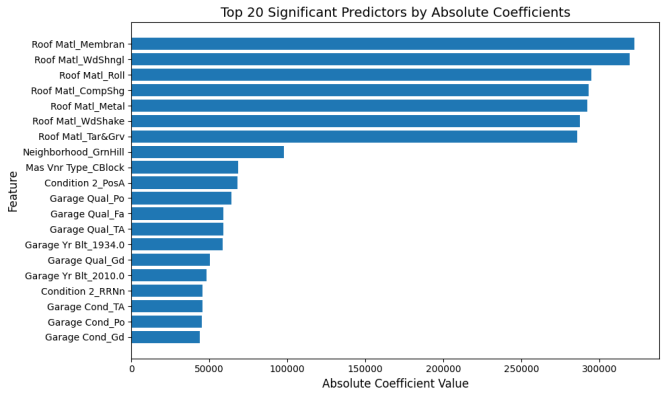

Figure 6: Top 20 Feature Importance in the Linear Regression model

Here, we have as first most important predictors and by far, variables linked to the roof materials of the houses, such as if the roof is made of membrane or wood etc... and also a lot of variables linked to the garage characteristics. We will see that those most important predictors denote compared to the ones we will find in the other models.

Improvements of this model, especially of the use of that numerous explanatory variables could be to either adopt a stepwise regression approach (backward or forward selection) of our features, ie. by getting rid one by one of the non-significant variables or use a regularization method ie. to add a penalty for the introduction of too many predictors as we did in our following section with the use of a Lasso penalty.

6.2.2 Linear Regression Model with Lasso Penalty

Now, as an improvement of our first Linear Regression model, we implement a first Linear Regression model by adding a Lasso penalty with an alpha parameter equal to 0.1, ie. quite small and slightly penalizing and we thus get very similar R^2 and adjusted R^2 to the ones in the simple linear regression of respectively 0.904 and 0.732.

To determine the optimal lambda value, we performed 5-fold cross-validation. This process identified the best alpha value as 387.03, which minimizes the Mean Squared Error (MSE), defined as the square of the RMSE. The evolution of the MSE across different alpha values is illustrated in Figure 7.

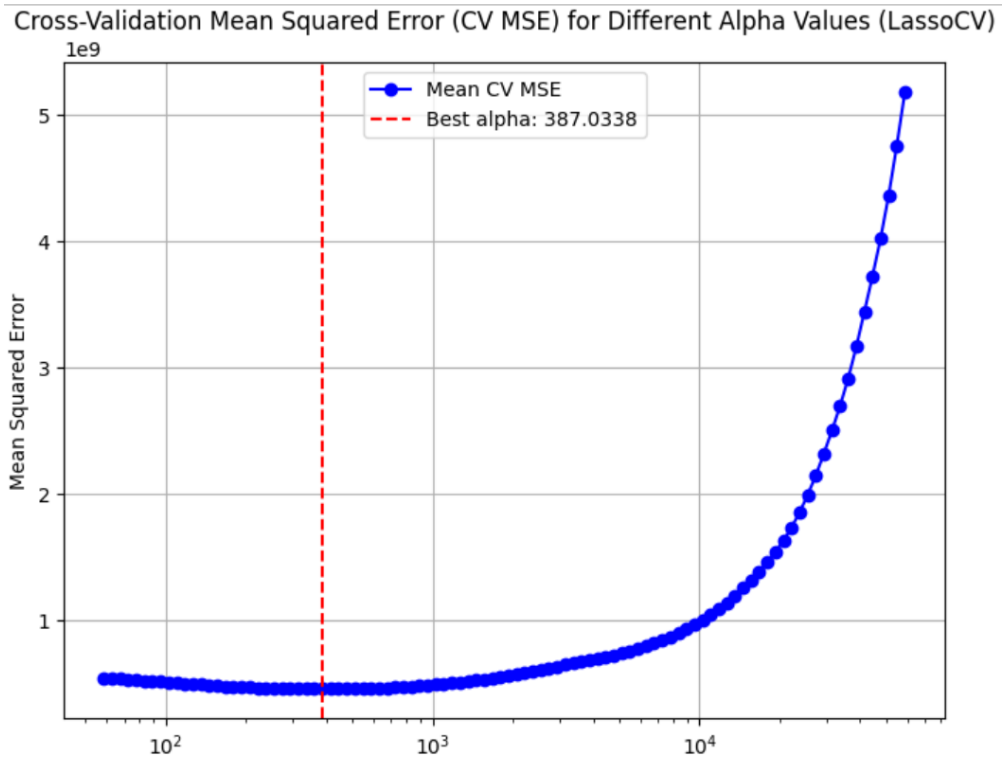

Figure 7: CV MSE for different alpha values tested

The plot demonstrates the MSE curve's convexity as an alpha function. Initially, the MSE decreases with increasing alpha, reflecting the benefits of regularization. At the optimal alpha of 387.03, marked by the red dashed line, the MSE reaches its minimum. Beyond this point, the MSE rises sharply as alpha increases further, revealing over-regularisation's negative impact. The convex nature of the curve guarantees a single global minimum, highlighting a clear optimal solution. This optimal alpha balances model complexity with predictive accuracy, effectively minimizing the MSE.

With this optimal value of alpha, we obtained a mean cross-validation value of R^2 equal to 0.908 (slightly higher) and a mean cross-validation adjusted R^2 equal to 0.733.

Let us now also show the top 20 most important predictors in this setting :

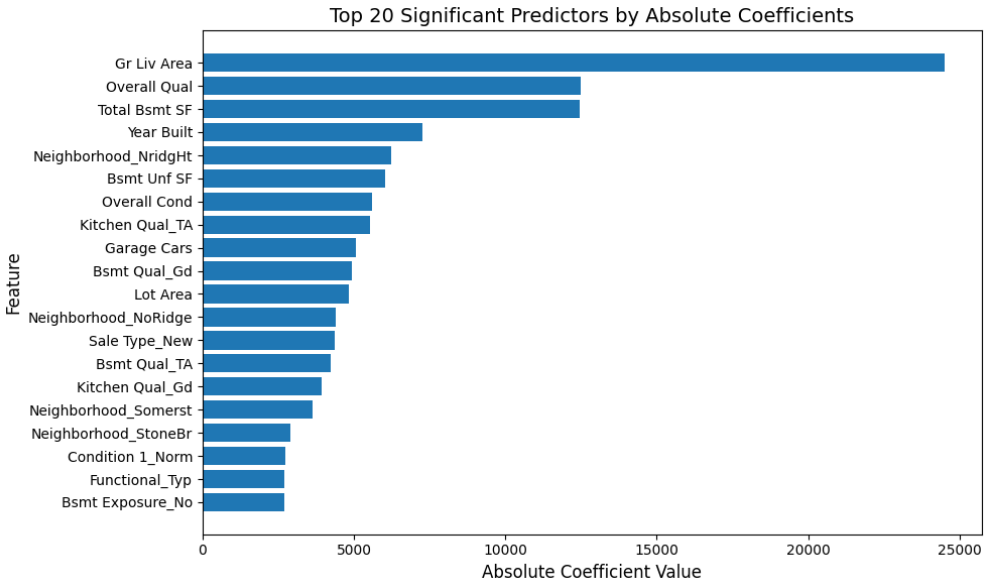

Figure 8: Top 20 Feature Importance in the Linear Regression with Lasso model (5-fold cross validation

The results indicate that the above-grade living area in square feet (**Gr Liv Area**) is the most influential feature, followed by the overall quality grade (**Overall Qual**) assigned to the house, and finally the total basement area in square feet (**Total Bsmt SF**). These features, for instance, are closely related to the size, functionality, and perceived quality of the house, which significantly influence its market value.

Those results show Linear Regression with Lasso's characteristic of doing effective feature selection by shrinking irrelevant coefficients to zero, thereby providing a meaningful interpretation of the most influential predictors. This process not only simplifies the model but also ensures that the retained features are directly tied to the target outcomes, enhancing interpretability.

As in Linear Regression, the analysis showed that the residuals exhibit normality, and there is a strong linear relationship between the actual and predicted values, validating Lasso's assumptions and ensuring robust predictive performance.

6.2.3 Random Forest Model

The Random Forest model was evaluated using 5-fold cross-validation to ensure a robust estimation of its performance. Initially, the model achieved a Cross-validation Mean Squared Error (MSE) of 519,518,252.96, a Mean Absolute Error (MAE) of 14,444.69, a Root Mean Squared Error (RMSE) of 23,236.83, and an R^2 score of 0.9027. The Adjusted R^2 was calculated as 0.7276, and the percentage of error was 8.82%.

To improve the model's performance, a Grid Search was combined with 5-fold cross-validation to optimize its hyperparameters. This approach slightly enhanced the results, yielding a Mean Absolute Error (MAE) of 14,210.16, a Root Mean Squared Error (RMSE) of 23,089.56, and an R^2 score of 0.9039. The Adjusted R^2 improved to 0.7310, and the percentage of error decreased to 8.73%.

The marginal improvement achieved through hyperparameter tuning highlights the value of combining cross-validation with grid search to fine-tune the model. This process not only enhanced predictive accuracy but also ensured the robustness of the Random Forest model.

To gain further insights into the predictive power of the features used in the Random Forest model, the top 20 most important predictors were identified and are visualized in Figure 9. Feature importance values provide a measure of how much each feature contributes to the model's predictive performance.

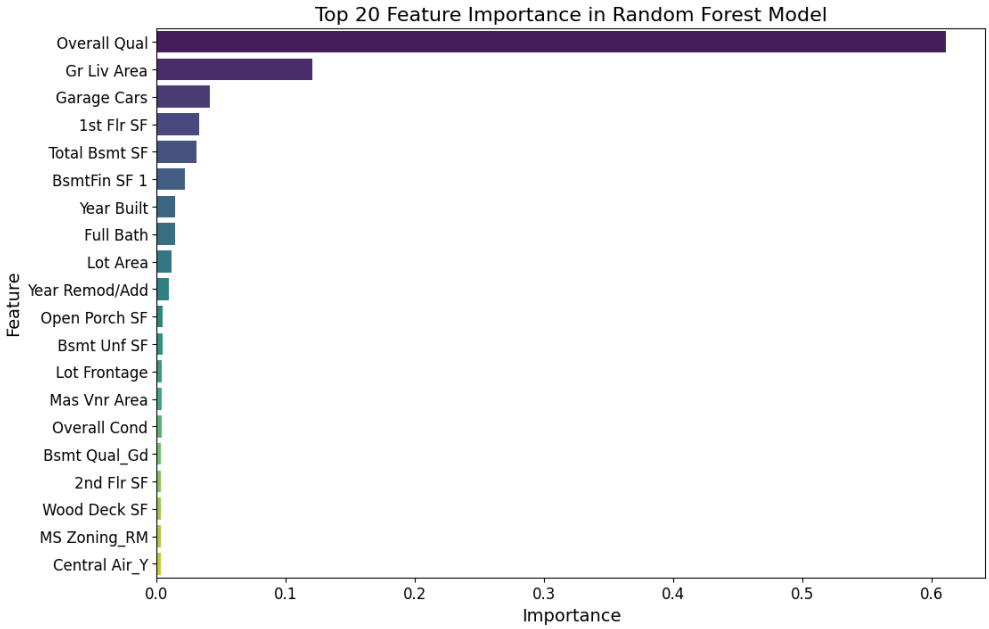

Figure 9: Top 20 Feature Importance in the Random Forest Model.

The results highlight that **Overall Quality** was by far the most influential feature, followed by **Gr Liv Area** and **Garage Cars**. These features are closely related to the quality and size of the property, aligning with expectations in predicting house prices. Other important predictors include **1st Flr SF**, **Total Bsmt SF**, and **Year Built**, which further emphasize the relevance of structural and size-related attributes in the model.

Features such as **Lot Area**, **Full Bath**, and **Year Remod/Add** also contributed significantly, suggesting that both the lot characteristics and recent renovations play a role in determining house values. Towards the lower end of the importance scale, features like **MS Zoning_RM** and **Central Air_Y** demonstrated a minor impact, indicating a more limited role in the overall prediction.

The ranking of feature importance underscores the Random Forest model's ability to capture complex relationships between property attributes and target outcomes, offering a meaningful interpretation of the predictors used.

The analysis confirms that the residuals exhibit normality, and there is a strong linear relationship between the actual and predicted values, validating the model's assumptions and ensuring reliable predictive performance.

6.2.4 Gradient Boosting Model

The Gradient Boosting model was also evaluated using 5-fold cross-validation. The results demonstrated strong predictive performance, with a cross-validation Mean Squared Error (MSE) of 447,887,231.01, a Mean Absolute Error (MAE) of 13,537.92, and a Root Mean Squared Error (RMSE) of 21,827.45. The coefficient of determination, R^2, was 0.9141, while the adjusted R^2 was 0.7596, indicating a robust fit. The percentage error of the model was calculated at 8.22%.

To further enhance the model's performance, a grid search was conducted in conjunction with 5-fold cross-validation. This optimization yielded improved metrics: the MAE decreased to 12,900.94, the RMSE dropped to 21,235.54, and R^2 increased to 0.9187, with an adjusted R^2 of 0.7725. The percentage error was also reduced to 7.76%, highlighting the benefit of hyperparameter tuning.

When compared to the Random Forest model, the Gradient Boosting model demonstrated slightly better predictive performance across all metrics. Specifically, Gradient Boosting achieved a lower MAE and RMSE, along with higher R^2 and adjusted R^2 values. These improvements, combined with a reduced percentage error, indicate that Gradient Boosting offers more accurate and reliable predictions than the Random Forest model, particularly after hyperparameter optimization through grid search.

A feature importance analysis for the Gradient Boosting model identified the top 20 predictors, as shown in Figure 10. The chart illustrates the relative contribution of each feature to the model, with the most significant predictors strongly influencing the outcome variable.

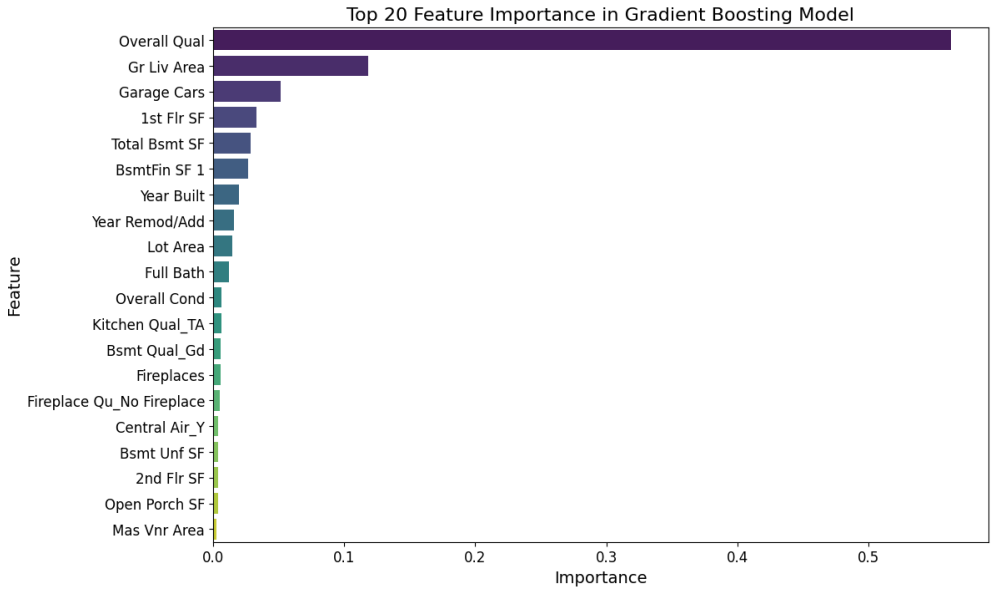

Figure 10: Top 20 Feature Importance in the Gradient Boosting Model.

The top 20 predictors for the Gradient Boosting model highlight the dominance of **Overall Quality**, which is by far the most influential feature. This is followed by **Gr Liv Area** (Above Ground Living Area) and **Garage Cars**, emphasizing the importance of living space and parking capacity. Other key structural variables, such as **1st Flr SF**, **Total Bsmt SF**, and **BsmtFin SF 1**, underline the significance of property size and additional usable spaces.

Chronological features like **Year Built** and **Year Remod/Add** highlight the importance of property age and renovations, while **Lot Area** and **Full Bath** reflect the land size and the number of bathrooms. Categorical features such as **Kitchen Qual_TA** and **Overall Condition** emphasize the quality and state of the property, with aesthetic attributes like **Open Porch SF** and **Mas Vnr Area** playing smaller but notable roles.

Compared to the Random Forest model, the Gradient Boosting model similarly prioritizes structural and quality-related features but places more weight on **Overall Quality**, while giving slightly less importance to **Gr Liv Area** and **Garage Cars**.

The residual analysis confirms the assumptions of normality, while the relationship between the actual and predicted values exhibits strong linearity, supporting the reliability of the gradient-boosting model's predictions.

6.2.5 Support Vector Regression Model

Finally, we implemented a first Support Vector Regression model with parameters chosen by default, except for the choice of the kernel that was obvious, ie. linear, we had as results the worst ones for now (very high MAE and RMSE, a coefficient of determination equal to only 0.45 and even a negative adjusted R squared).

Performing a grid search (with 5-fold cross-validation) to tune the hyperparameters of SVR was thus necessary.

By using the values of these best parameters, we obtained largely better results with an MAE equal to 13941.14, an RMSE equal to 23885.42, a R^2 equal to 0.90, an adjusted R^2 equal to 0.71 and a percentage of error equal to 8.30%.

When compared to the other models used until now, we get that, even by using a linear kernel, our SVR model does not have necessarily better predictive performance compared to Gradient Boosting for instance, even if it stays quite good and has close results. This might be since SVR modelization might be more adapted when modelling non-linear relationships.

This graph in Figure 11 displays the **permutation importance** of features in predicting house prices, which quantifies the impact of each feature on the model's prediction accuracy by measuring the change in model performance when the feature is randomly shuffled.

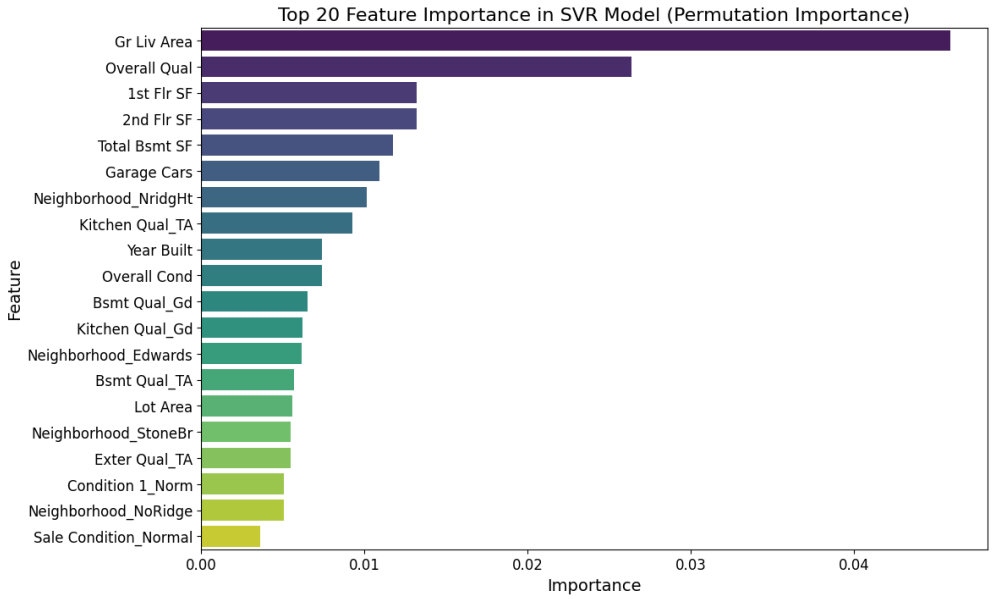

Figure 11: Top 20 Feature Importance in Support Vector Regression Model

The most important feature for predicting house prices is **Gr Liv Area** (Above Ground Living Area in square feet). Larger living areas tend to correlate with higher prices. **Overall Qual** (Overall Quality) indicates the overall material and finish quality of the house. Higher quality often leads to higher house values. The size of the first and second floors, **1st Flr SF** and **2nd Flr SF**, plays a significant role in determining the house price. Additionally, **Total Bsmt SF** (Total Basement Square Footage) is an important predictor, as a larger basement area adds usable space.

Garage Cars (Capacity of Garage in Terms of Cars) indicates that houses with larger garages tend to have higher values.

When considering neighbourhood features, **Neighborhood_NridgHt** (Northridge Heights) indicates that this neighbourhood is associated with higher property values, possibly due to location advantages. Kitchen quality also plays a key role in pricing, as **Kitchen Qual_TA** and **Kitchen Qual_Gd** (Typical and Good Kitchen Quality) reflect the importance of functional and aesthetic kitchen spaces. Furthermore, **Bsmt Qual** (Basement Quality) shows that high-quality basements contribute positively to the price.

In terms of other observations, features like **Year Built** and **Overall Condition** reflect the general upkeep and modernity of the house, while other neighborhood-related and exterior quality features also show some importance.

These results differ from the previous two models, where **Overall Qual** was significantly more important than other features; in this case, it ranks second in importance.

6.3 Results with the Monte Carlo Cross Validation

Finally, as explained previously, we implemented again all of our best-tuned models but by employing Monte Carlo Cross Validation and by computing for each one of our models the mean metrics obtained. This approach allows us to add some resampling strategy to our analysis and ensure that the evaluation of our models reflects their true predictive capabilities.

Here are the results we have found by using 10 different train-test splits :

Table 1: Final Results with Monte Carlo Cross Validation

Model	**MAE**	**RMSE**	**R^2**	**Adj. R^2**
Linear Regression	15166.4	22973.3	0.8991	0.7175
LassoCV	14182.7	21282.4	0.9129	0.7561
Random Forest	14708.6	22597.1	0.9021	0.7261
Gradient Boosting	**13112.1**	**20060.4**	**0.9230**	**0.7844**
SVR	14324.6	21901.3	0.9077	0.7418

Based on the Monte Carlo cross-validation results, the **Gradient Boosting Regression** outperforms other models across multiple metrics. It achieves the lowest Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), indicating better predictive accuracy. Additionally, it has the highest R^2 mean (0.9230) and Adjusted R^2 mean (0.7844), signifying that it explains the largest proportion of variance in the target variable **while accounting for model complexity**.

The combination of **high interpretability** and **robust predictive performance** makes **Gradient Boosting Regression the most suitable model** for our analysis.

7 Conclusion

To conclude, in our analysis, we have considered 5 statistical and machine learning approaches to predict the price of houses based on several of possible features.

The model that performed best for all the metrics we considered and when implementing Monte Carlo cross-validation was the Gradient Boosting Model, that here adapts well to the specific structure of our data, and is quite robust to overfitting and captures subtle dependencies between features and our target variable.

The strengths of our approach lie in its use of diverse models, such as Random Forest and Gradient Boosting, which help capture complex non-linear relationships in the data. However, there are also challenges, including the trade-off between model complexity and interpretability, particularly with ensemble methods that can be more difficult to explain compared to simpler models like linear regression.

In future work, improving model interpretability could be a priority, perhaps by incorporating feature selection techniques or simplifying the ensemble methods. Additionally, addressing issues related to overfitting and model tuning could further enhance the predictive performance.

To conclude, this work has highlighted the effectiveness of machine learning methods in predicting housing prices. The use of diverse techniques such as Random Forest, Gradient Boosting, and Lasso regression has demonstrated their applicability and flexibility in handling complex datasets. The combination of these methods brings originality and depth to the analysis, making the results highly relevant to both statisticians and data scientists interested in predictive modelling.

8 References

- Gadat, Sébastien (2022). *Supervised Learning Regression*. Mathematics of Machine and Deep Learning Algorithms Course.
- Gadat, Sébastien (2022). *ML Methods*. Mathematics of Machine and Deep Learning Algorithms Course.
- Gil Casals, Silvia (2021). *Support Vector Machines (SVM)*. Data Mining Course.
- Junaiddata35. 2024. *Ames Housing Dataset - House Price Prediction*. Kaggle.com. December 3, 2024. Ames Housing Dataset - House Price Prediction