Course: Machine Learning

Exercise 1: Regression Problem

Name: Konstantinos Vardakas

Student ID: 522

Email: pcs0522@uoi.gr

Dataset Description

This project is based on the well-known 'House Prices: Advanced Regression Techniques' competition, which was hosted on the Kaggle platform. The dataset relates to houses in Ames, Iowa, and includes characteristics that can be used to predict a property's final sale price. For this project, the train.csv file was used, containing 79 features (80 if the record ID is included) and the sale price (SalePrice), the latter being the target feature.

The variables in the dataset cover various categories of information, such as location and plot, condition and quality, characteristics related to the rooms of the house, materials used in construction, and time-related information about the construction of the house. These variables are either numerical (integer or decimal) or categorical (nominal or ordinal). Regarding the target feature, values range from $34,900 to $755,000, and show a right skew, as shown in the histogram in Figure 1.

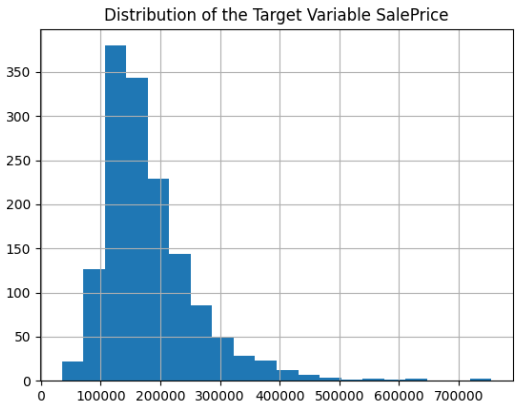

Figure 1: Distribution of the target variable SalePrice

Data Preprocessing

Data preprocessing is a critical step in the development of machine learning models, as it directly affects not only the performance and generalization of the models, but also their speed and convergence during training. Especially in the case of linear models, careful handling of variables is necessary to satisfy basic assumptions such as linearity, homoscedasticity, and independence of errors. In addition, linear models are more sensitive to outliers, which makes feature cleaning, transformation, and encoding necessary steps. This chapter describes some important preprocessing steps that were performed. For the sake of brevity, not all individual steps are presented in detail.

Initially, features with extremely low variance (quasi-constant features) were identified and removed, i.e., variables that have almost the same value in more than 99% of observations. Such features have minimal predictive value and may introduce noise or cause overfitting . Some of these features are PoolQC and PoolArea (no pool in 99.5% of cases), Street (99.6% cobblestone) and Utilities (99.9% AllPub).

One of the characteristics of the dataset, MSSubClass, represents the type of residence based on style and use (e.g., single-family home, duplex, split-level, etc.), but in the form of numerical codes, without implying any quantitative or hierarchical relationship between the values. For reasons of simplification and better interpretation, the feature was transformed to express the number of floors in the residence. The mapping was based on a categorization of dwelling types according to the following correspondence:

1-Story	1.5-Story	2-Story	2.5-Story	Split/Multi-Level	Multi-Family
0	1	2	3	4	5

Table 1: Mapping of the MSSubclass feature, based on floors

Although the new feature apparently represents ordinal categories, its linear correlation with the sale price proved negligible (Pearson coefficient = -0.034). Therefore, one-hot encoding was chosen for its incorporation into the model to avoid

imposing an artificial hierarchy that does not reflect the actual patterns in the data. The absence of a linear correlation can also be seen in the boxplots below.

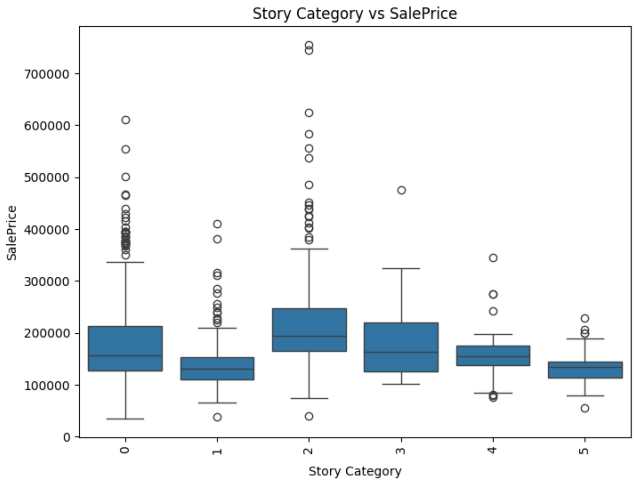

Figure 2: Box plots showing the absence of a linear relationship between the Story Level feature and the target feature

The LotArea feature, which represents the area of the plot, presented significant outliers that could disproportionately affect the training of the model, especially when using cost functions based on squared error . For this reason, a logarithmic transformation was applied to reduce the asymmetry of the distribution and limit the effect of outliers, making the variable more suitable for modeling.

Then, as with any other numerical feature, normalization was applied (transformation to a distribution with mean ± std = 0 ± 1). This process facilitates the convergence of training algorithms, especially in models based on optimization through derivatives (such as linear models), and reduces the risk of getting stuck in local minima when finding the model parameters. In addition, scaling the features to the same range of values allows conclusions to be drawn from the feature coefficients regarding their importance in the final prediction.

In the following plot, which highlights the relationship between LotArea and SalePrice, the confidence interval widens as LotArea increases. These high and extreme values can significantly affect the behavior of the model, since MSE gives greater weight to large deviations.

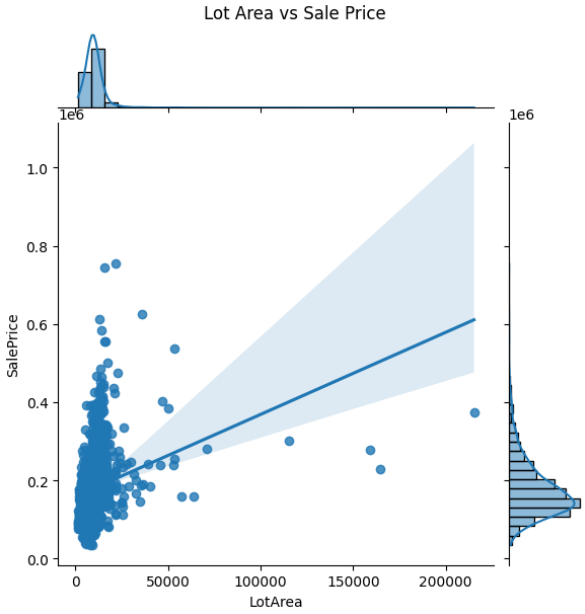

Figure 3: Correlation of the LotArea attribute with the target attribute SalePrice

Between the OverallQual (overall quality) and OverallCond (overall condition) characteristics, it was decided to use only OverallQual, as it shows a clearer linear correlation with SalePrice, as shown in the following figure.

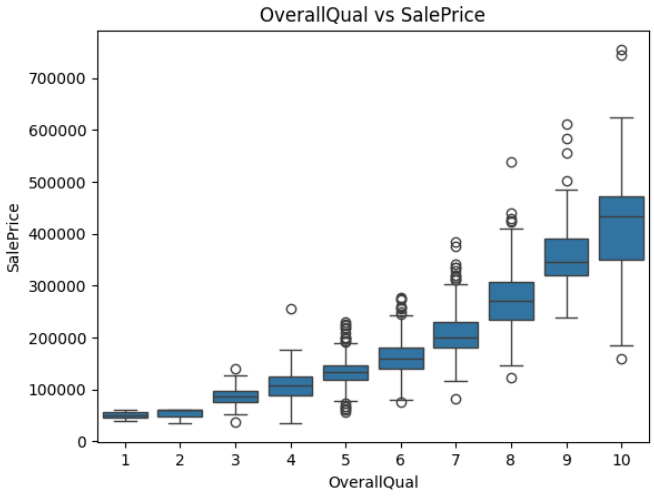

Figure 4: Correlation of the OverallQual attribute with the target attribute SalePrice

Furthermore, OverallCond concentrates most of the values in category 5, which also has a high std (Saleprice with OverallCond = 5: std = 85.117 compared to std of the dataset = 79.442). The OverallQual feature that was retained was normalized with min-max scaling, since the minimum and maximum permissible values of the variable (1 to 10) are known and constant.

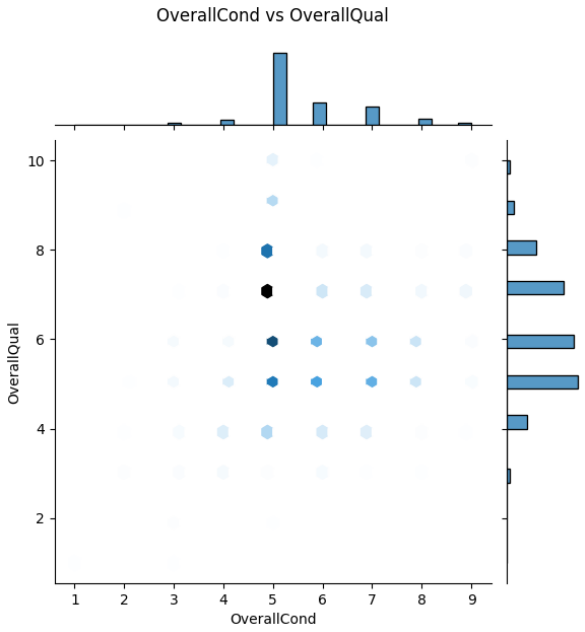

Figure 5: Differences in the distribution of OverallQual and OverallCond characteristics

Regarding chronological characteristics (such as YearBuilt, YearRemodAdd, GarageYrBlt), a transformation was applied to relevant time differences in relation to YrSold, so that the data would reflect the age of each characteristic at the time of sale. For example, YearBuilt was transformed into YrSold - YearBuilt.

Although the original features were replaced by the corresponding time values, YrSold was retained, as the year of sale may contain information relevant to market trends for each year. The three chronological features that reflect the chronological difference (YearBuiltDiff, YearRemodDiff, GarageYrBltDiff) were then normalized through standard scaling. In contrast, YrSold was encoded with one-hot encoding in order to capture any non-linear trends per year in the real estate market.

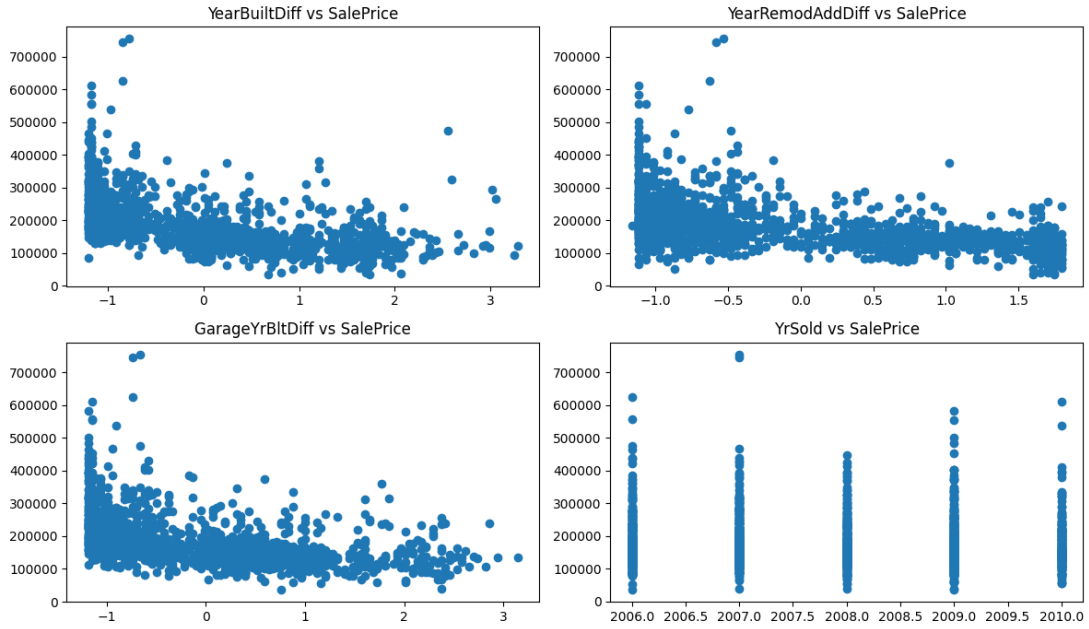

Figure 6: Correlation of time characteristics with the target characteristic

The GarageYrBltDiff attribute, which expresses the difference between the year of sale and the year of construction of the garage, shows missing values in cases where there is no garage. These values were replaced with 0, as the absence of a garage is represented by the GarageCars attribute, to which category 0 was added for properties without a garage.

Among the characteristics GarageCars (number of vehicles that can fit in the garage) and GarageArea (garage area), a strong correlation was observed with a Pearson correlation coefficient = 0.88, which makes them redundant characteristics. However, GarageCars was preferred, as it showed higher predictive power in simple linear regression with SalePrice ($R^2 = 0.419$ vs. 0.385 for GarageArea).

The last time-related feature is the month of sale (MoSold). To reduce the complexity of the features and better capture seasonality, we chose to group months into seasons. Instead of simply categorizing by quarter or four-month period, a more flexible approach was taken using circular transformation and unsupervised learning.

The months were transformed into a circular form in order to maintain the natural periodicity of time. Specifically, each month was initially converted into an angle using the formula:

$$\vartheta = \frac{2 \cdot \pi \cdot month}{12}$$

Then, the sine and cosine functions were applied, creating two new features. With this transformation, months such as December and January are depicted as temporally close, which is not the case in the original linear space.

Taking advantage of this conversion, clustering was applied to the three-dimensional space containing the values cos(x), sin(x), and SalePrice, with the aim of grouping months that are both temporally close and have similar average sales prices. To avoid the dominance of SalePrice in the result, it was normalized before applying the clustering algorithm. Finally, the number of clusters was set to four, and the final seasonal categories were encoded using one-hot encoding. The resulting grouping is shown in the diagram below.

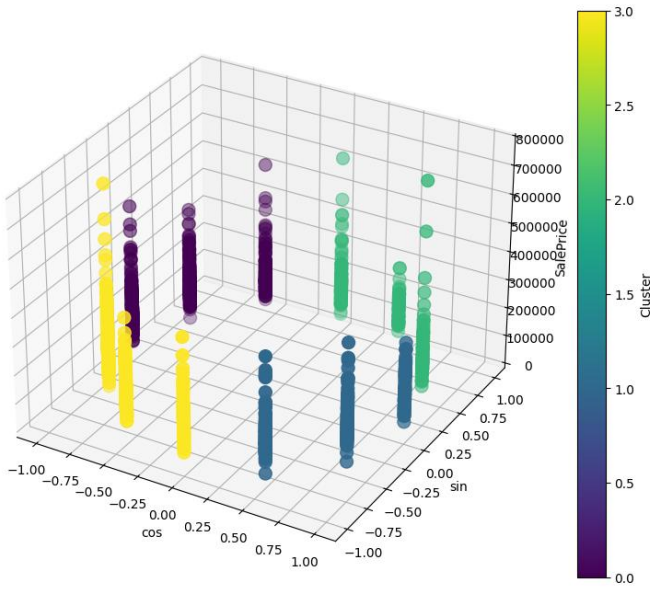

Figure 7: Grouping of seasons based on property sale prices.

The chart below also shows the grouping in the initial space, using the average Standardized Sale Price for each month.

Figure 8: Grouping of seasons based on the sale price of properties in the original location

About the remaining characteristics, additional selection and processing techniques were applied with the aim of reducing dimensionality and improving the quality of the inputs to the model. Specifically, numerical features that showed low linear correlation with the sales price (Pearson correlation less than 0.4) were removed, as it was considered that their contribution to the prediction was limited and they might introduce noise or reinforce overfitting.

At the same time, categorical characteristics were grouped based on either conceptual relevance (e.g., similar construction materials were classified together) or economic approach (e.g., areas were grouped into "high," "medium," and "low" cost, depending on average property prices).

Finally, categorical characteristics that had a classified scale of values (e.g., quality or condition rating from 1 to 5) and showed a relatively stable linear correlation with the sale price were coded as ordinal variables in order to maintain the hierarchy of categories.

Feature Selection

At this stage, the features have been fully preprocessed and are now suitable for comparative evaluation and selection. Missing data values have been filled in, the categorical features have been grouped based on the information they convey, either in terms of their significance or in relation to the target (SalePrice), and the numerical features have been transformed to present, as far as possible, a linear correlation with the sale price.

At the same time, the effects of extreme values have been identified and reduced, while all variables have been normalized, resulting in them moving on a similar scale. This is particularly important as it enables consistent comparison between numerical and categorical features and allows for the correct application of feature selection techniques without being disproportionately affected by differences in scale.

The first step in feature selection was based on the use of Pearson's correlation coefficient. Specifically, all possible pairs of features were examined, and those with an absolute correlation greater than 0.8 were selected. In each pair with a high correlation, only the feature with the highest Pearson coefficient in relation to the sale price (SalePrice) was retained, while the other was removed as redundant.

The choice of this method was based on the fact that Pearson's coefficient measures the linear dependence between variables, which is particularly useful for linear models such as linear regression. However, it is important to note that this approach does not identify non-linear relationships (e.g., logarithmic or exponential), which limits its use when selecting features in more complex models, such as neural networks or Gaussian Processes with RBF kernels.

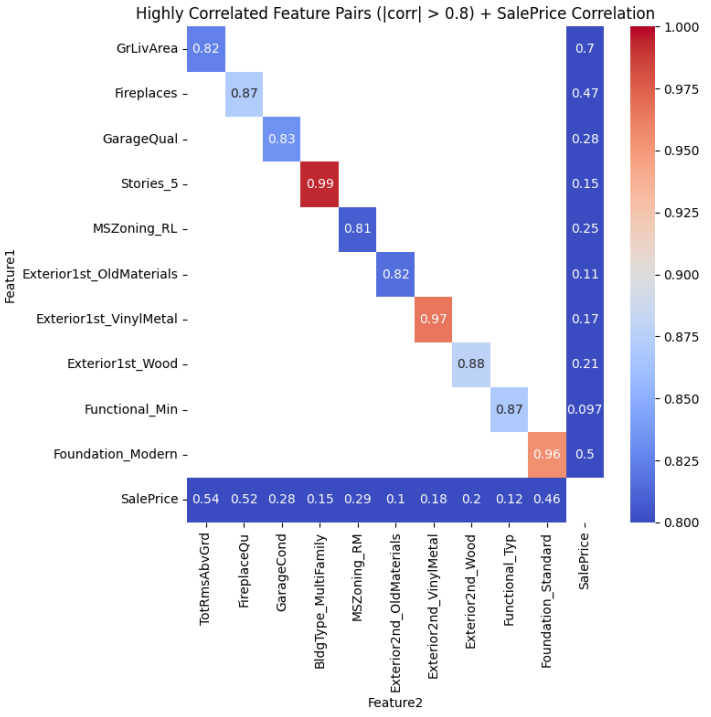

Figure 9: Pairs of characteristics with high absolute Pearson Correlation coefficient values and their correlation with the target characteristic

To further reduce dimensionality and highlight the features with the greatest predictive power, the Sequential Feature Selection (SFS) method was then applied in a forward direction, using linear regression as an estimator. The selection was made from the complete set of features and evaluated using 5-fold cross-validation, based on performance in terms of the R^2 coefficient.

The following diagram shows the change in the cross-validation score in relation to the number of selected features. Maximum performance is achieved with 34 features, while from this point onwards, the addition of new features leads to a decrease in accuracy, possibly due to the introduction of noise or redundant information.

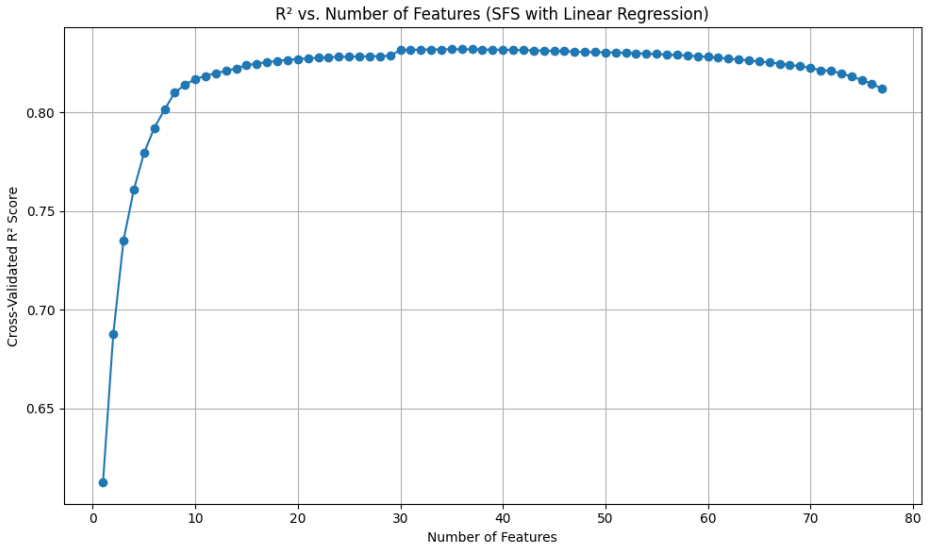

Figure 10: R^2 plot as a function of the number of characteristics

It is worth noting that, as in the case of Pearson correlation, the ideal subset of features may differ depending on the nature of the model (linear or non-linear).

An interesting observation concerns the addition of the one-hot encoded values of the SaleCondition feature. Specifically, with the addition of the SaleCondition_Normal feature (29th in order), the improvement in the score is marginal (~5*10-5). However, when the information for the remaining two categories, SaleCondition_NonNormal and SaleCondition_AllocaLand, is also incorporated, the representation of the categorical variable is completed and a significant increase in R^2 is observed (~2*10-3, a 100-fold increase).

Training and Analysis of Predictive Models

For each of the following models, a common training split and testing split were used to ensure comparability between results. In addition, the same KFold cross-validation class object (with 10 folds) was applied to the training split so that the initial training set was divided into the same ten subsets each time and thus trained and validated on the same data.

Linear Regression

The first model examined is the Linear Regression model. This model attempts to fit a hyperplane to the data, minimizing the mean square error between the predicted and actual values.

After training the linear model on the training set with the 34 selected features, the average cross-validation $R^2 = 0.836 \pm 0.044$ (10-fold CV) was obtained, indicating stable performance on the training set. Next, the model was fitted to the entire training set and evaluated on the testing set, resulting in $R^2 = 0.87$, demonstrating good generalization ability.

In the histogram of residuals (difference between actual and predicted values), it can be seen that they generally follow a normal distribution, with some outliers in both directions. These extreme errors reach values of ±100,000, which suggests that the model may be sensitive to individual cases with extremely high or low sales prices.

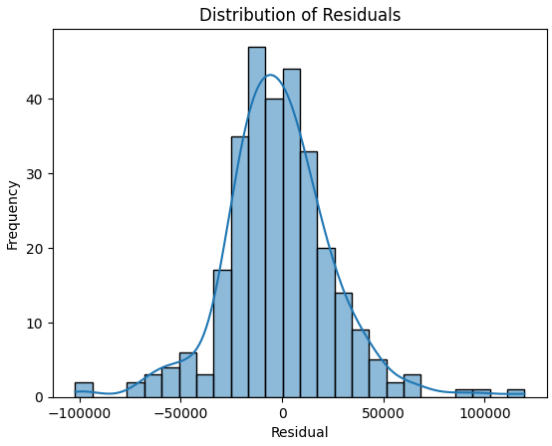

Figure 11: Distribution of residuals

The Q-Q plot reinforces this observation, as deviations from the normality line appear mainly at the extremes of the distribution, confirming the presence of outliers. The existence of these values can have a significant effect on the model coefficients, as already mentioned.

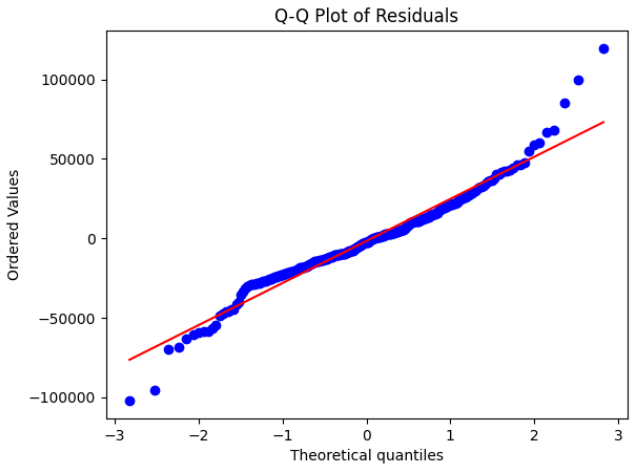

Figure 12: Q-Q plot of residuals

In order to limit the influence of outliers on the model coefficients, a logarithmic transformation was performed on the target. This transformation normalizes the distribution of the target variable, which now more closely approximates a normal distribution. As a result, outliers better conform to the overall trend of the data and their effect on model training is significantly reduced, leading to more stable and representative coefficients.

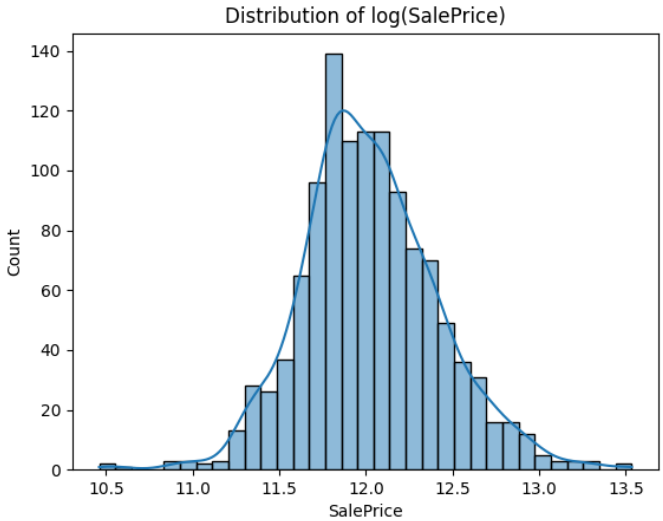

Figure 13: Distribution of the target characteristic after logarithmic transformation

After applying the logarithmic transformation and training the model, the cross-validation R^2 improved to 0.889 ± 0.026, i.e., an increase of 0.047, while the standard deviation decreased by 0.018, indicating more stable results. The test R^2 also improved, reaching 0.897, recording an increase of 0.02 compared to the R^2 before the transformation. This improvement demonstrates the effectiveness of the logarithmic transformation in enhancing the model's generalization ability.

Below is the histogram of residuals after logarithmic transformation of the target. The outliers on the left side of the distribution have been significantly reduced, while on the right side there are still deviations.

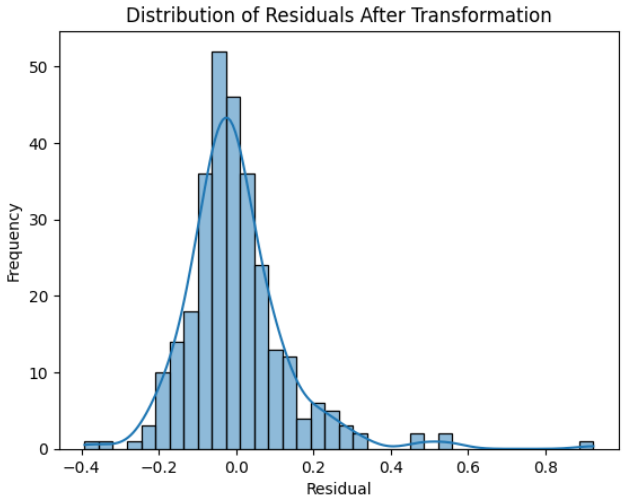

Figure 14: Histogram of residuals after logarithmic transformation

It is worth noting that, following the logarithmic transformation of SalePrice, the model now predicts the logarithmic sale price.

Polynomial Regression

The application of polynomial regression entails an exponential increase in the number of features, especially when high-degree polynomials (e.g., up to 10th degree) are considered. For this reason, instead of using the 34 original features, Principal Component Analysis (PCA) was applied, with the aim of reducing dimensionality and limiting computational complexity.

Specifically, the first 5 principal components were selected, which retain the largest percentage of information from the original dataset. The data were then

transformed into polynomial features up to degree 10, and a linear model was trained in this enriched feature space.

Thus, the number of features resulting from the application of a polynomial transformation of degree d=10 is calculated based on the formula:

$$\binom{n+d}{d} = \frac{(n+d)!}{d!}$$

When n = 34, the result is 2481256778, while when n = 5, the result is only 3003 features. After generating the polynomial features for each degree of polynomial, it is crucial to apply normalization so that all features are on the same scale, and thus can converge to some minimum of the training set.

When applying PCA, the original dimensions of the data are transformed into a new orthogonal space, where each new dimension is a linear combination of the original features. The principal components are ranked in descending order according to the amount of variance they explain in the data. Specifically, the first principal component is the one that aligns with the maximum variance of the data and therefore captures the most important part of the total information (without necessarily being related to the target feature). This means that this direction contains the dominant pattern of the data, while the subsequent components capture secondary variations.

The following figure shows the target feature as a function of the principal component (PCA component 1), with the predictions obtained for each degree of the polynomial. Initially, it can be observed that there is linearity between the principal component and the target, so it appears that important information for predicting the target feature has been retained.

It can be observed that when the degree of the polynomial is small, the model appears to be capable of generalizing and approximating the actual values of the test split. As the degree of the polynomial increases, the model is given more flexibility, and thus learns the noise of the dataset rather than the information, i.e., it overfits.

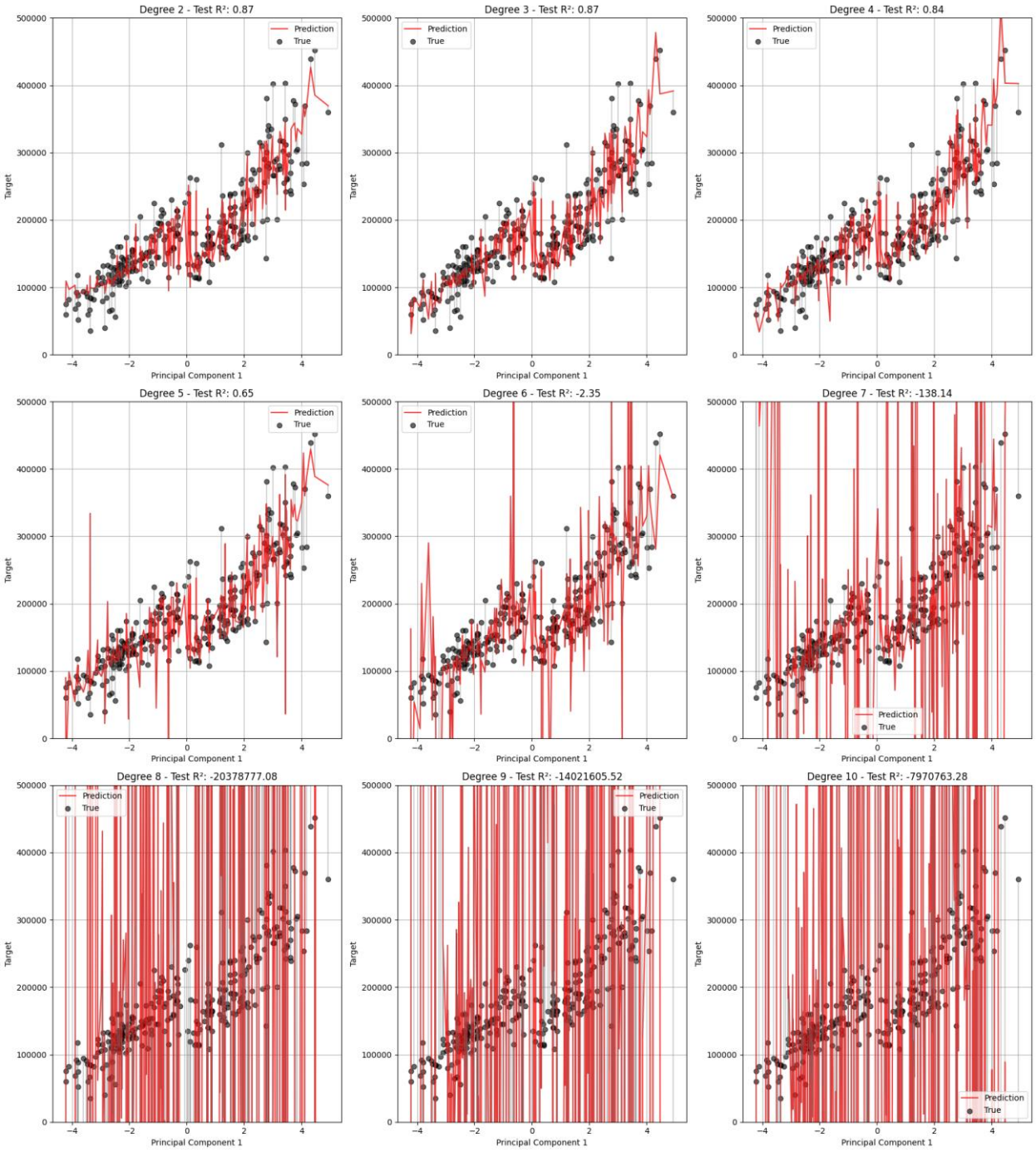

Figure 15: Line of the function f(x) for each set of characteristics x, compared to the PCA component 1 vs Saleprice diagram

This flexibility of the model with the increase in the polynomial degree is also reflected in the table below. Specifically, the model zeroes the error in the training set, increases the error in the test set, and increases the variance in the cross-validation sets, thus preventing the model from generalizing to new data.

Degree	2	3	4	5	6	7	8	9	10
train R^2	0.844	0.862	0.897	0.928	0.955	0.983	1.0	1.0	1.0
test R^2	0.867	0.865	0.838	0.653	-2.34	$-1e^2$	$-2e^7$	$-1e^7$	$-8e^6$
cv mean	0.819	0.771	0.651	0.052	-32.3	$-2e^4$	$-1e^7$	$-5e^{18}$	$-9e^{18}$
cv std	0.135	0.230	0.446	1.168	59.3	$5.6e^5$	$2.9e^7$	$1.5e^{19}$	$2.8e^{19}$

Table 2: Metrics of the model for each degree of polynomial used

Using the logarithmic value of the target characteristic, no difference was observed.

Lasso Regression

Lasso Regression is a method used to improve the accuracy of the model on unknown data by adding a regularization term to the error term. The loss function for Lasso has the form:

$$Loss = \sum_{i=1}^{M} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

The first term is the same as the term in the Linear Regression model, which represents the squared error between the observations and the predictions. The second term is the regularization term, which adds a penalty for the absolute value of the weights wj. The hyperparameter λ determines the intensity of the regularization and, consequently, the weight of the regularization term in relation to the total error.

Lasso Regression is particularly useful when we have a large number of features, as through normalization it can automatically select the most important features, eliminating the less important ones and thus avoiding overfitting. In this way, it enhances the generalization of the model to unknown data.

As in the case of the Linear Regression model, it was observed that predictions improve when the logarithmic values of SalePrice are used. Therefore, logarithmic values will be used to present and evaluate the results. Also, since Lasso tends to zero out feature weights that either introduce noise or provide redundant information, the full set of features will be used. The selection of the most important features will result

from the training process, through the zeroing of the corresponding weight coefficients by the model itself.

Under these conditions, the range of values studied is from $1e^{-5}$ to $1e^{2}$, and the results are shown in the figure below.

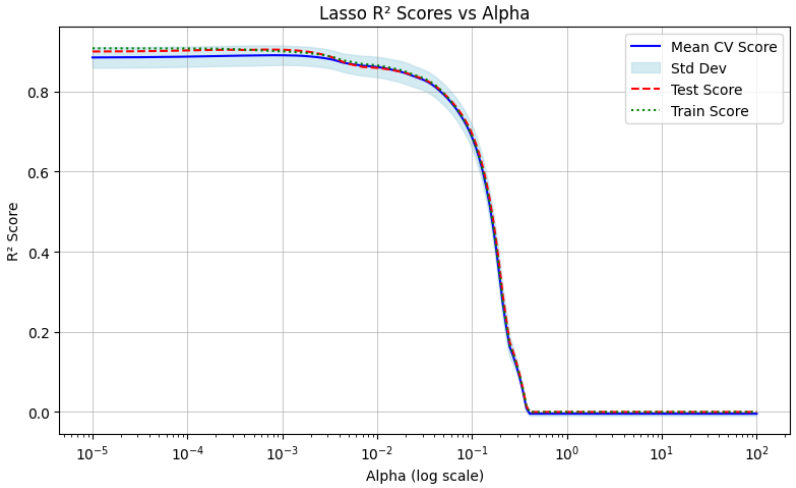

Figure 16: Plot of the R^2 metric as a function of penalty value

The test score appears to improve up to a certain point (approximately for a value of $\lambda \approx 10-3$), achieving the best generalization ability (test score = 0.905, CV score = 0.891 ± 0.024). Beyond that point, the normalization term begins to dominate, reducing even coefficients that contribute significantly to the prediction of the target variable. This results in a reduction in both the train score and the test score. Finally, for very large values of λ, the coefficients are zeroed and the prediction is due to the bias term of the model, which takes the value of the mean of SalePrice, leading to R2 = 0.

The following figure shows the coefficients of the best model that emerged. The complexity of the model appears to be low, as most coefficients have already been zeroed out using a fairly small λ (0.0008).

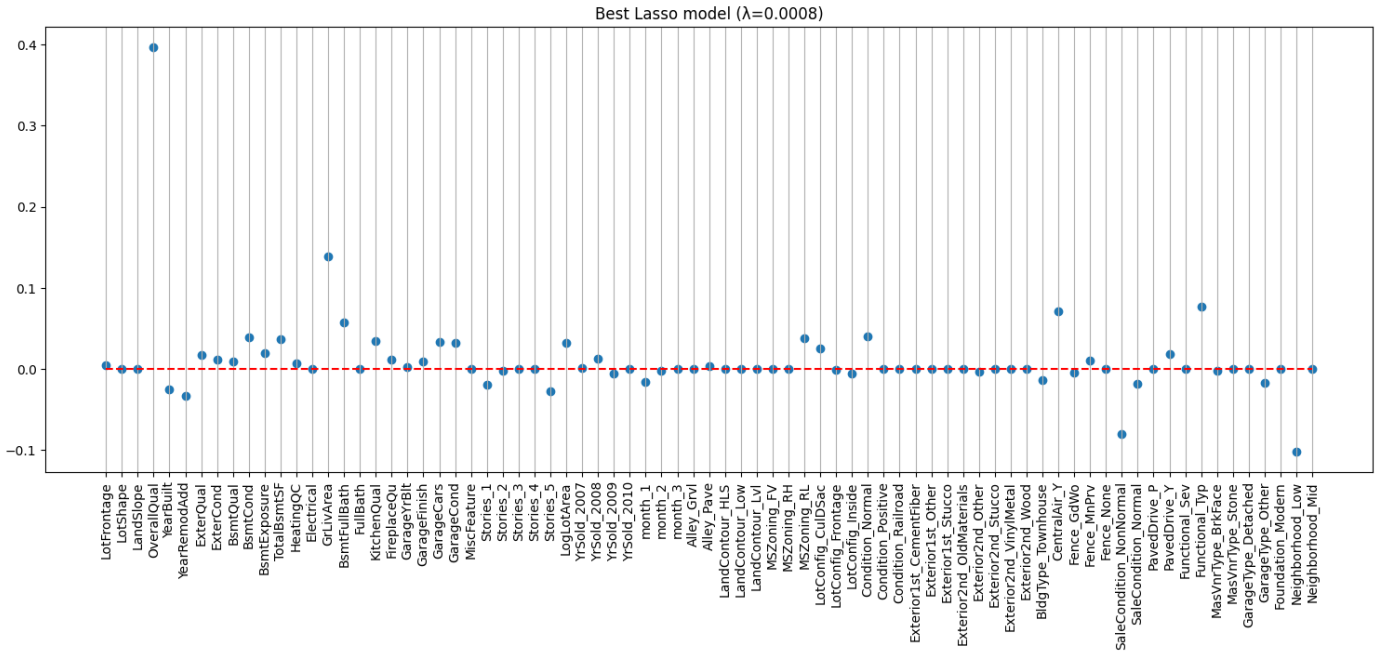

Figure 17: Coefficients of the features for the λ value that gives the best model, based on the test score

The Overall Quality feature appears to have the greatest impact on the model, as it adds approximately 0.4 logarithmic units to log(SalePrice) when it changes from minimum to maximum (i.e., from 0 to 1 on the min-max scale). To convert the change from a logarithmic value to a normal price unit, we apply the inverse of the logarithm, i.e. $\exp(0.4) \approx 1.5$. This means that a property with Overall Quality at the maximum of the scale (1 after normalization) is predicted to have approximately 49% higher price than a property with Overall Quality at the minimum of the scale (0), holding all other characteristics constant

The temporal characteristics appear to have a negative correlation with the property price, as the older the date of construction or renovation, the lower the predicted sale price. This suggests that newer homes tend to be valued higher. In addition, the Neighborhood_Low feature has a strongly negative effect, suggesting that this location is associated with significantly lower prices compared to other areas.

The following diagrams show the evolution of the coefficients as the value of the normalization parameter increases. It can be seen that as the penalty increases, the coefficients gradually shrink and eventually become zero, revealing the feature selection process applied by Lasso.

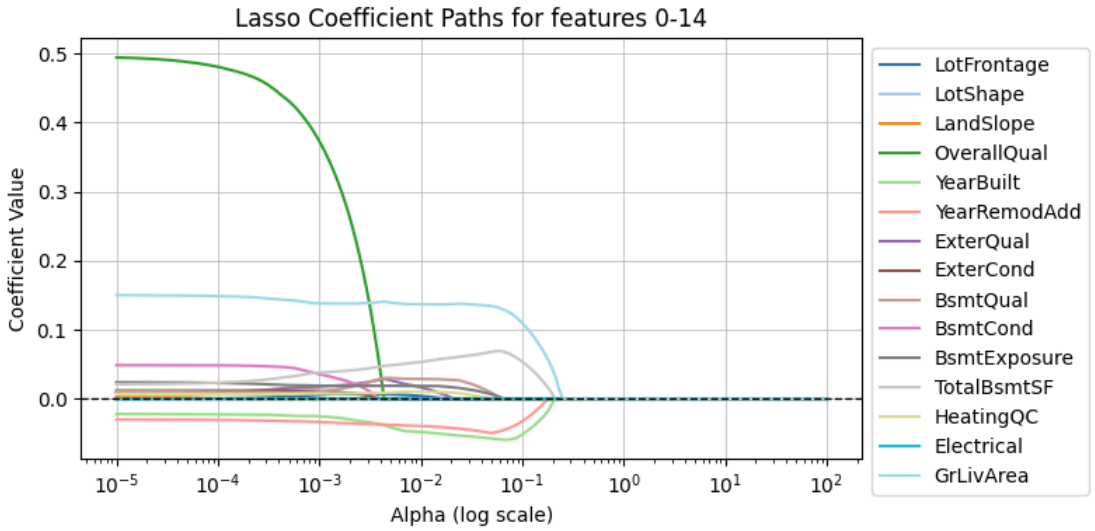

Figure 18: Coefficients of characteristics 0-14 as a function of log(penalty) values

This diagram also shows the characteristic that had the greatest weight in the best model, namely Overall Quality. It can be seen that, for small values of the normalization parameter, its coefficient is significantly greater than that of the other characteristics. However, as the penalty increases, its coefficient becomes zero earlier than others. This zeroing seems to be accompanied by the strengthening of the coefficients of other features, which probably convey similar (if not redundant) information. Many of these characteristics are shown in the following diagram. The diagrams for the remaining characteristics (30-76) are not presented in the report for reasons of brevity.

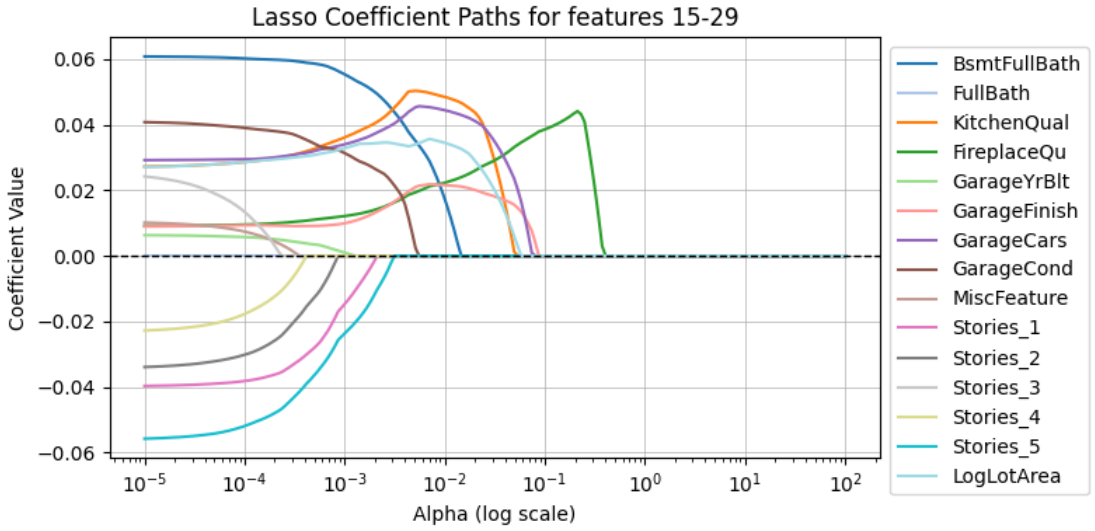

Figure 19: Coefficients of characteristics 15-29 as a function of log(penalty) values

Multilayer Neural Networks

1 Hidden Layer

Multi-layer neural networks are powerful machine learning models capable of capturing complex non-linear relationships in data. Unlike simpler linear models, neural networks have hidden layers, in which feature engineering is performed automatically. In this way, the network is able to identify complex, non-linear relationships in the data without the need for this process to be done manually. Thus, the features of layer n are derived from nonlinear transformations of combinations of the features of layer n−1, allowing the network to gradually "build" more abstract and descriptive representations of the data.

To implement non-linearity, the ReLU (Rectified Linear Unit) activation function was used. ReLU introduces non-linearity at a very low computational cost, as its derivative is simple: zero for values less than zero and equal to the input for positive values. In this context, two different architectures were applied: one with one and one with two hidden layers, in order to study the effect of depth on the accuracy and generalizability of the model.

Initially, the first model tested was the one with a single hidden layer containing 10 neurons, using the non-logarithmic values of the target. The model achieved a CV score of 0.8706 ± 0.0522, while in the test set it scored a final R2=0.8885, demonstrating good generalization ability in this case as well.

The following heatmap illustrates the feature engineering performed at the hidden layer by optimizing the coefficients during training on the training set. The activations of the neurons in the hidden layer result from nonlinear combinations of the original features, through the trained weights and the ReLU activation function. Since the features have been normalized beforehand, resulting in a comparable scale, the coefficients can be interpreted as importance indicators: the greater the absolute value of a weight, the greater the effect of the corresponding feature on the activation of that neuron.

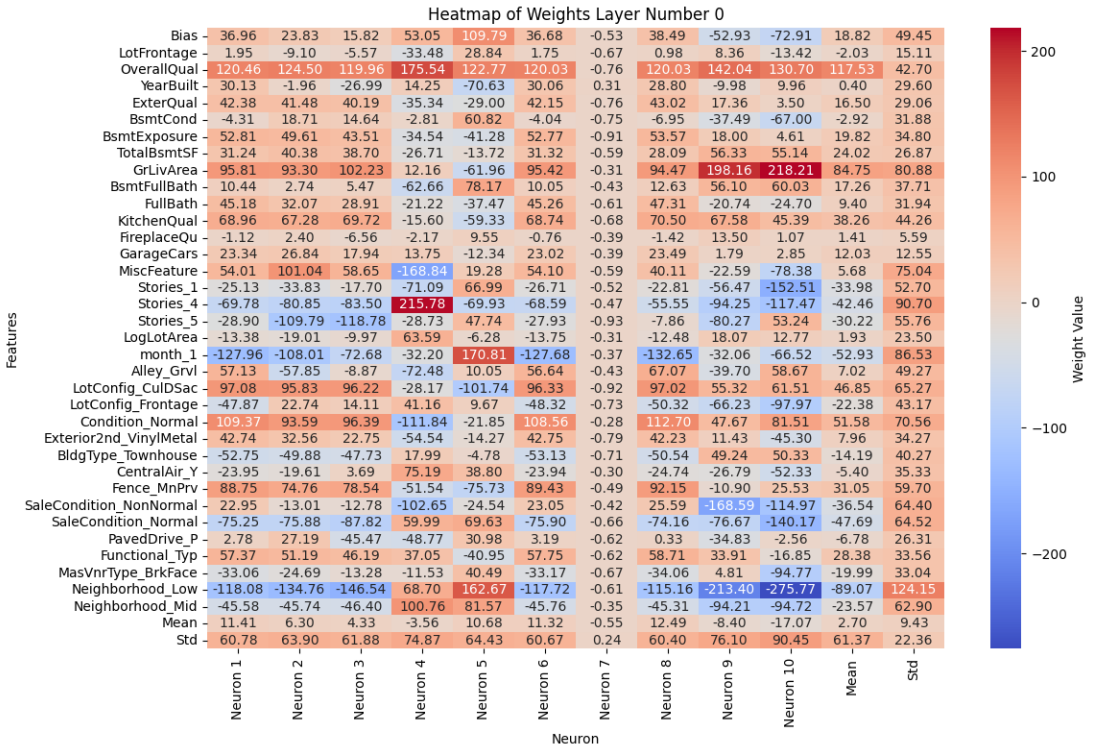

Figure 20: Heatmap with the coefficients of each neuron for each feature.

With regard to the features, the picture is similar to that of the analysis of the Lasso Regressor coefficients. The OverallQual feature emerges as the most important, as it contributes significantly to the estimation of a property's value. Specifically, it has the highest average coefficient in all hidden units, with a value of approximately 117.53 ± 42.70. The Neighborhood_Low feature, on the other hand, appears to be the one that most negatively affects the value of a property, as its average coefficient is 89.07. However, it shows a large variation (124.15), as in neuron number 5 it has a very large and positive value (162.67). Also, the feature extracted from the KMeans clustering, season, seems to be important, as in spring (feature month1), it has a very low coefficient (-52.93).

With regard to feature engineering, neurons 4, 5, and 10 are of particular interest, as they have the highest coefficients (in absolute value) towards the model output: 179.29, 192.11, and 122.90, respectively, as shown in Figure 21.

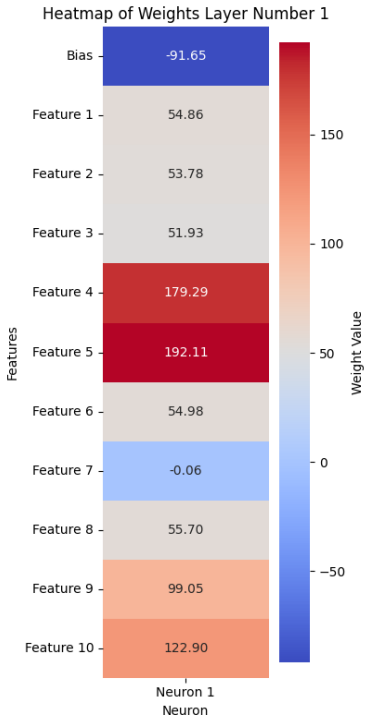

Figure 21: Weights and coefficient of the model's output layer

However, although neuron 10 has a smaller coefficient in comparison, its activation by the input features is significantly higher in absolute terms (-17.07 ± 90.45), which could enhance its contribution to the final prediction. However, since ReLU is used as the activation function, it makes the contribution of this neuron zero in cases where a negative number is extracted. Also, neuron 4 has smaller weights on average than the input features (-3.56), so its total contribution to the final prediction is expected to be smaller. It is worth noting that neuron 7 has a negligible effect, as both its activations from the input features and the output coefficient are close to zero.

Neuron number 5 appears to play an important role in the overall prediction, as it has the largest coefficient in the final output layer. However, its focus on features related to lower-value properties, such as Neighborhood_Low and Month_1, is counterintuitive, as these are usually associated with low prices. Nevertheless, the overall average effect of these features on all neurons, as already mentioned, is small, indicating that their influence is limited, possibly due to low activations (ReLU) or compensation from other features.

It is worth noting that the above analysis is based on the assumption that all features are on the same scale, which is true due to the normalization that was applied.

However, there is also an implicit assumption that the distributions of the features are uniform, which would mean that each feature activates the neurons with similar frequency. For this reason, a more empirical analysis follows, presenting the outputs of each neuron when the training set passes through the network.

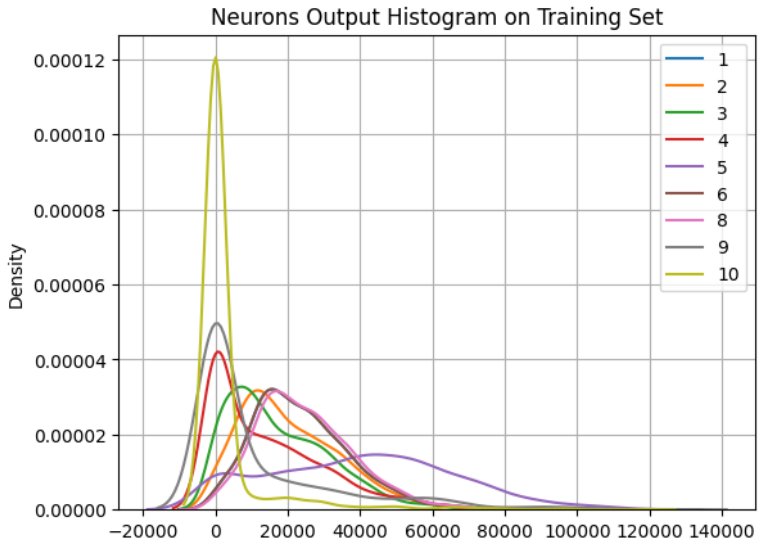

Figure 22: Distributions of the outputs of each neuron when the training set is input

no Neuron	1	2	3	4	5	6	7	8	9	10
out value (mean ± std)	23621 ± 13207	20666 ± 14019	17375 ± 13865	14299 ± 15933	40094 ± 25615	23584 ± 13201	-92 ± 0	24833 ± 13323	12082 ± 21641	3256 ± 11963

Table 3: Mean±std values for the output of each neuron in the training set

As can be seen from the above results, the neuron with the greatest influence on the final prediction is indeed neuron number 5. The high variance of its outputs is probably due both to the large final weight it carries (192.11), which amplifies the effects of the incoming features, as well as to the significant contribution of the OverallQual feature, which has a wide range of values and a high weight. On the other hand, neuron number 7 has zero variance and a mean value equal to its final bias, which suggests that its output after ReLU is constantly zero, therefore, its only contribution is through the bias, which is added consistently to the final sum. Regarding the effect of neuron number 10, the initial assumption that its contribution to the final prediction is very limited is confirmed.

2 Hidden Layers

The model with two hidden layers has the ability to extract more complex and abstract features, combining the already transformed features of the first layer into even more complex nonlinear relationships. In this way, it can "learn" deeper structures within the data, potentially offering better adaptation to complex patterns that cannot be captured with a single hidden layer. For brevity, the results will be described more generally.

The architecture used was 2 hidden layers with 10 neurons each, and ReLU activation function. The model achieved a slightly better test score R2 = 0.8897 and a marginally lower CV R2 = 0.8706 ± 0.0522. The difference in performance is practically negligible, especially when considering the additional computational cost: the network includes 10×10 more weights and 10 additional biases. This indicates that, for this particular problem, a single hidden layer is sufficient to capture the patterns of the dataset.

In general, it can be observed that the final coefficients of the model with two hidden layers are smaller in size. This is to be expected, since starting from the same initial features, the additional hidden layer "spreads" the influence of each feature over more intermediate steps. The activations now go through an additional transformation stage, resulting in the final values ending up at the same desired outputs, but with more dispersed contributions from each feature.

In both cases, training R2 is slightly better than the test set, indicating a small degree of overfitting (1 hidden layer: 0.027, 2 hidden layers: 0.034).

Gaussian Processes

Gaussian Processes (GP) are a powerful nonparametric regression approach, which instead of directly learning the weights w of a model, expresses the solution as a linear combination of observations through a vector α, i.e., $f(x) = \sum a_i k(x, x_i)$. In this way, direct projection into high dimensions (as was the case with polynomial regression) is avoided, and the problem is converted into a calculation between samples

using the kernel trick, thus achieving an indirect reduction in dimensionality and better handling of complexity.

Another important advantage of Gaussian Processes is that, in addition to predictions, they also provide the covariance matrix of the predictions. Taking the diagonal elements of this matrix, i.e., the covariance of each prediction with itself, we obtain the variance of each prediction, from which we can easily extract the corresponding standard deviation as a measure of uncertainty.

The function $k(x, x_i)$ mentioned above is the kernel function used to calculate the similarity between points x and xi. In this case, the linear kernel and the Gaussian kernel (RBF) are used.

As expected, the linear kernel produced results equivalent to the Linear Regression model, as both methods are based on linear relationships between variables. Specifically, when the logarithmic value of SalePrice was used, the test R^2 was 0.8966 and the cross-validation R^2 σε 0.8889 ± 0.0260, was 0.8889 ± 0.0260, while without the logarithmic transformation, the corresponding values were 0.8707 for the test R^2 nd 0.8363 ± 0.043 for the cross-validation.

The following chart shows the model's prediction (transformed SalePrice value) as a function of GrLivArea for 50 data points from the test data set. The confidence interval (±2 standard deviations) has been multiplied by 105 to make it more visible in the chart. It can be seen that the range of the confidence interval is significantly smaller in areas with high point density, while in areas with fewer points it is larger.

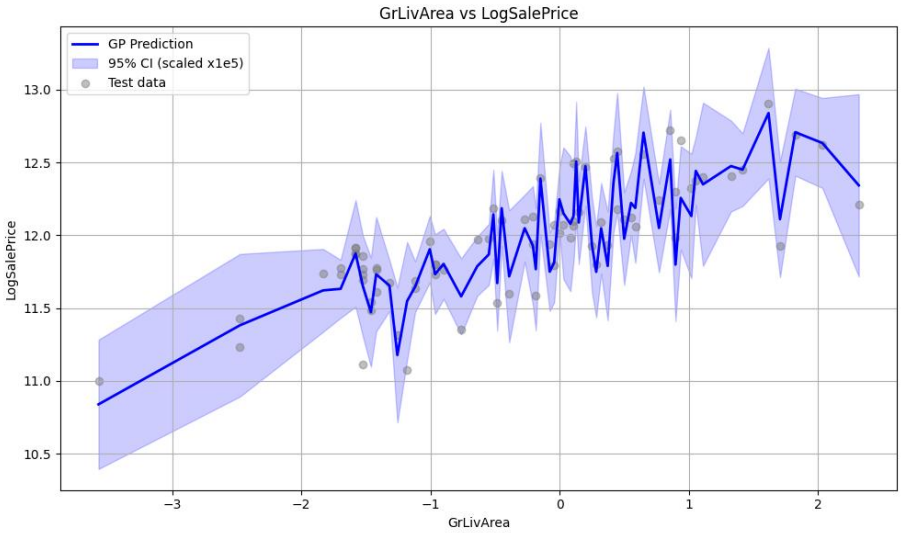

Figure 23: Predictions of the model function of a continuous characteristic, GrLivArea

With regard to the RBF kernel, it was observed that without the addition of noise, the model overfitted the noise in the data, resulting in an inability to generalize correctly. To address this, noise following a normal distribution was added, and when the noise was 1, the model's performance was optimal in terms of the testing score. Furthermore, the training score was similar to the testing score, indicating that the model was no longer overfitting.

With regard to the parameter σ, no sensitivity was observed when the noise was sufficient for the model to produce reliable predictions. On the contrary, when the artificial noise is very small, it was observed that small values of σ (which maximize the effect of nearby neighboring points) lead to better results. The following table shows the train and test scores for each sigma and Noise Level (train R^2| test R^2).

		Noise Level					
		0	1e-5	1e-3	1e-1	1	10
sigma	1e-2	1|0.5	1|0.53	1|0.68	0.98|0.85	0.89|0.89	0.76|0.84
	1e-1	1|-5.8	1|0.53	1|0.68	0.98|0.85	0.89|0.89	0.76|0.84
	1	1|-0.7	1|0.53	1|0.68	0.98|0.85	0.89|0.89	0.76|0.84
	1e1	1|-5.9	1|-5.9	1|-5.9	0.98|0.85	0.89|0.89	0.76|0.84
	1e2	1|-5.9	1|-5.9	1|-5.9	0.98|0.85	0.89|0.89	0.76|0.84

Table 4: Model metrics for each sigma and noise value

The following diagram shows the predictions in the test set in 50 examples, with the confidence interval multiplied by 2e4 so that it is distinguishable in the diagram.

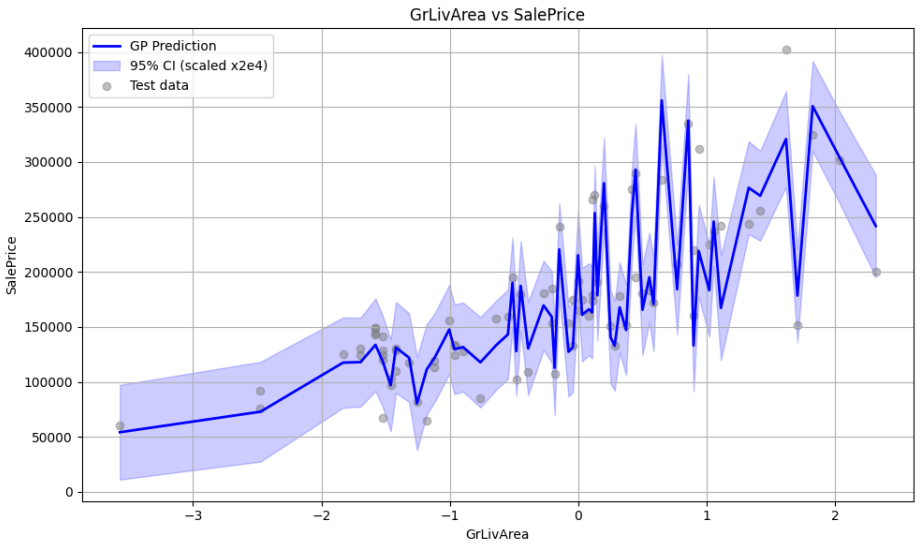

Figure 24: Prediction of the model as a function of GrLivArea

Models Comparison and Conclusions

In this project, different regression models were applied and compared for predicting real estate sales prices, starting with simple linear methods and moving on to more complex approaches, such as neural networks and Gaussian processes.

The simplest model initially used was Linear Regression, which produced excellent results, especially when the target feature was logarithmized. This transformation converted the initially skewed distribution of the target (Figure 1) into a Gaussian distribution (Figure 13), reducing the effect of extreme values on the final weights and the total error, since MSE was used as the cost function.

The Polynomial Regression model showed good generalization ability when low-degree polynomials were used. The features used were derived from projecting the original features of Linear Regression onto a 5-dimensional subspace, with the aim of reducing dimensionality and enabling the use of up to 10 degrees. As the degree of the polynomial increased, overfitting was observed, as expected, with the R^2 of the training set approaching unity, while the test score and the average cross-validation score decreased significantly, with the latter showing increasing variance. Compared to

Linear Regression, no improvement was observed in terms of generalization to unknown data.

With regard to the Lasso Regressor, optimal performance was achieved using the complete set of features, with the penalty value gradually leading to the elimination of redundant features. As expected, the best predictions were achieved when the target was logarithmic, as in the case of Linear Regression. The model's performance on the test set was better than any other model used.

The two neural networks used showed similar results. The network with two hidden layers had comparable performance to the one with a single hidden layer, despite the fact that it included 10x10 more weights and 10 additional biases between the first and second hidden layers.

Finally, Gaussian Processes models showed equally good generalization ability. With a linear kernel, the performance was expectedly similar to that of Linear Regression, both in the logarithmic and untransformed form of the target. In contrast, with the RBF kernel, the best results were achieved in the untransformed target, but without the addition of artificial noise, the model exhibited severe overfitting for each value of sigma. With the addition of noise, overfitting was reduced, but the best model ultimately scored a lower R^2 in the testing set compared to the corresponding linear kernel model.

The table below shows the results of the models and the Target that provided these metrics. For the Polynomial Regression model, the model with a polynomial degree of 2 is shown.

Model	Train R^2	CV R^2	Test R^2	Target
Linear Regression	0.897	0.889 ± 0.026	0.897	log(SalePrice)
Polynomial Regression	0.844	0.819 ± 0.135	0.867	SalePrice
Lasso Regression	0.902	0.891 ± 0.024	0.905	log(SalePrice)
NN 1 Hidden Layer	0.916	0.871 ± 0.052	0.889	SalePrice
NN 2 Hidden Layer	0.924	0.861 ± 0.087	0.890	SalePrice
GP Linear Kernel	0.897	0.889 ± 0.026	0.897	log(SalePrice)
GP RBF Kernel	0.892	0.855 ± 0.039	0.889	SalePrice

Table 5: Metrics for each model and format of the target used

It should be noted that there were some outliers in the training set, while the test set was cleaner, which explains why the test score is better than the cross-validation score, specifically in the case of non-logarithmic target values, where the outliers were left as they were. This was also observed in some batches, which had a large difference in R^2 from the rest during cross-validation (as can be seen in the table above, where models using SalePrice have a higher std in the R^2 score).

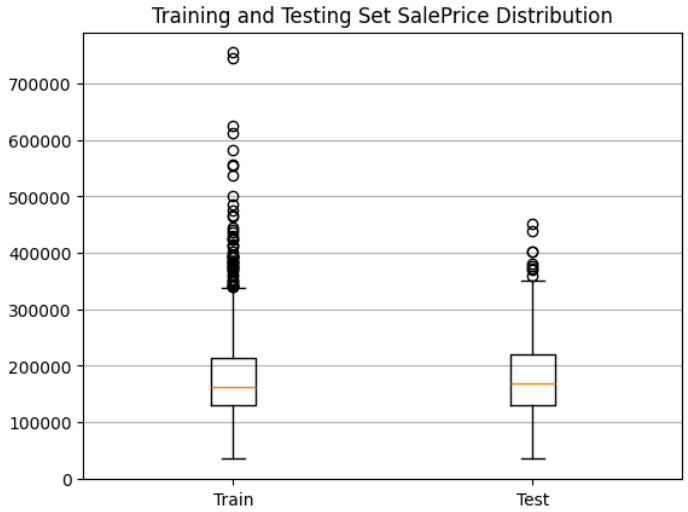

Figure 25: Boxplot of the training and the test set

Notes on the code

Two code files are included: a preprocess.ipynb file, where data preprocessing takes place, and a train.ipynb file, where basic feature selection is initially performed, followed by model fitting. Also included is train.csv from the Kaggle website, which is read in preprocess.ipynb and finally exported from the same file as data_preprocessed.csv. The latter is read from the train.ipynb file and the processes mentioned above are performed.