Predicting Housing Prices Using the Ames Housing Prices Competition Dataset from Kaggle

Tabasum Hamdard
Dec 4, 2024

Abstract

This comprehensive study uses advanced machine learning techniques to predict housing prices using Ames Housing Prices dataset taken from kaggle. The dataset encompasses 80 explanatory variables that describe several characteristics of residential properties in Ames, Iowa. The primary goal of our study is to achieve the most accurate predictions of housing prices that have the lowest average of errors through a combination of preprocessing, model training, and validation.

We explored various linear and nonlinear predictive models, including Ordinary Least Squares (OLS), Ridge Regression, Lasso Regression, Elastic Net, Principal Component Regression (PCR), Partial Least Squares (PLS), Multivariate Adaptive Regression Splines (MARS), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Neural Networks (NN). Steps involved in our data preprocessing was handling missing values, centering and scaling, and removing uncorrelated predictors.

We used cross-validation to train and evaluate our models and compare their predictive power on our test dataset. We analyzed relationships between predictors and target variables as a feature selection and, ultimately, identified the best-performing model. This provided us with the understanding of the factors that influenced housing prices and showed the importance of machine learning strategies in real estate price predictions.

Background

The *Ames Housing dataset* was first introduced in 2011 by Dean De Cock and became a stable data for enthusiast and student who wanted to learn machine learning (Kaggle, n.d.). The dataset consists of a variety of detailed information about the housing properties in Ames, Iowa. Since this was real-world data, it became an interesting and fun challenge for data scientists. The dataset contained different proportions of numerical and categorical data, missing values, and other real world problems as well.

Predicting housing prices is an interesting challenge that can be used for market analysis, investment options, and even state policy in the future. An accurate prediction of the model can help sellers, buyers, and developers to make informed decisions. Machine learning can be applied to explore this problem in its entirety, from traditional regression models to advanced models such as neural networks to see which one of these approaches perform best in tackling this issue.

Our main goal in this assignment is to use as many of these rich features as this dataset offers to make our models as precise as possible for housing price prediction. To achieve this goal we use a step-by-step approach, starting with importing the dataset, processing it according to our needs to highlight the dataset's strong features, and transforming it into a suitable format (Kuhn , 2013). Then we will process it with Ordinary Least Squares, Ridge, Lasso, Elastic Net, Support Vector Machines, Neural Networks, and other corresponding models. After that, we will compare the results meticulously to identify the predictive models that handle this dataset the best and explore ways to fine-tune our models to get the best result.

This exercise not only shows us the usage of predictive models in real life but also shows how real-world problems can be analyzed using advanced algorithm tools, particularly, in the real estate domain.

Variable Introduction and Definitions

The Ames Housing dataset, originally compiled in 2011, consists of 2,930 residential property records, providing 80 explanatory variables that contains detailed information of these homes. These variables offer a comprehensive view of residential properties, covering physical properties, location characteristics, building quality, and additional property features that collectively influence housing prices. They can be broadly categorized into numerical, categorical, and ordinal types, reflecting diverse data formats and levels of measurement. Numerical variables consist of measurable aspects, such as lot size or living area, while categorical variables classify properties into distinct groups, such as zoning designations or roof styles. Ordinal variables capture ranked qualities, such as overall material quality or condition, offering valuable information about property hierarchy and desirability of the houses.

For the purpose of this project, we are using only the training dataset, which was provided in the kaggle website, constituting approximately half of the total records (1460 sample size). This provided a manageable yet representative subset for building and evaluating predictive models.

We had a total number of 81 variables listed as follows:

Variable Name	Description
SalePrice	The property's sale price in dollars (target variable).
MSSubClass	The building class.
MSZoning	The general zoning classification.
LotFrontage	Linear feet of street connected to property.
LotArea	Lot size in square feet.
Street	Type of road access.
Alley	Type of alley access.
LotShape	General shape of property.
LandContour	Flatness of the property.
Utilities	Type of utilities available.
LotConfig	Lot configuration.
LandSlope	Slope of the property.
Neighborhood	Physical locations within Ames city limits.
Condition1	Proximity to main road or railroad.
Condition2	Proximity to main road or railroad (if a second is present).
BldgType	Type of dwelling.
HouseStyle	Style of dwelling.
OverallQual	Overall material and finish quality.
OverallCond	Overall condition rating.
YearBuilt	Original construction date.
YearRemodAdd	Remodel date.
RoofStyle	Type of roof.
RoofMatl	Roof material.
Exterior1st	Exterior covering on house.
Exterior2nd	Exterior covering on house (if more than one material).
MasVnrType	Masonry veneer type.
MasVnrArea	Masonry veneer area in square feet.
ExterQual	Exterior material quality.

ExterCond	Present condition of the material on the exterior.
Foundation	Type of foundation.
BsmtQual	Height of the basement.
BsmtCond	General condition of the basement.
BsmtExposure	Walkout or garden level basement walls.
BsmtFinType1	Quality of basement finished area.
BsmtFinSF1	Type 1 finished square feet.
BsmtFinType2	Quality of second finished area (if present).
BsmtFinSF2	Type 2 finished square feet.
BsmtUnfSF	Unfinished square feet of basement area.
TotalBsmtSF	Total square feet of basement area.
Heating	Type of heating.
HeatingQC	Heating quality and condition.
CentralAir	Central air conditioning.
Electrical	Electrical system.
1stFlrSF	First floor square feet.
2ndFlrSF	Second floor square feet.
LowQualFinSF	Low quality finished square feet (all floors).
GrLivArea	Above grade (ground) living area square feet.
BsmtFullBath	Basement full bathrooms.
BsmtHalfBath	Basement half bathrooms.
FullBath	Full bathrooms above grade.
HalfBath	Half baths above grade.
Bedroom	Number of bedrooms above basement level.
Kitchen	Number of kitchens.
KitchenQual	Kitchen quality.
TotRmsAbvGrd	Total rooms above grade (does not include bathrooms).
Functional	Home functionality rating.
Fireplaces	Number of fireplaces.
FireplaceQu	Fireplace quality.
GarageType	Garage location.
GarageYrBlt	Year garage was built.
GarageFinish	Interior finish of the garage.
GarageCars	Size of garage in car capacity.

GarageArea	Size of garage in square feet.
GarageQual	Garage quality.
GarageCond	Garage condition.
PavedDrive	Paved driveway.
WoodDeckSF	Wood deck area in square feet.
OpenPorchSF	Open porch area in square feet.
EnclosedPorch	Enclosed porch area in square feet.
3SsnPorch	Three-season porch area in square feet.
ScreenPorch	Screen porch area in square feet.
PoolArea	Pool area in square feet.
PoolQC	Pool quality.
Fence	Fence quality.
MiscFeature	Miscellaneous features not covered in other categories.
MiscVal	Value of miscellaneous features in dollars.
MoSold	Months sold.
YrSold	Year sold.
SaleType	Type of sale.
SaleCondition	Condition of sale.

Preprocessing

Prior to preprocessing it is always essential to remove the response variable from the dataset so there is no data leakage later for model building (BrownLee, 202). Therefore, we removed the target variable as well as the ID column because it was not necessary for further steps. This process left us with 79 predictors for the preprocessing step.

Missing Values

The dataset has 6965 missing values across several different variables. Some predictors have more than 50% missing from their total, which was removed to improve the reliability of our dataset. The removed columns are as follows:

1. **PoolQC:** Pool quality
2. **MiscFeatures:** Miscellaneous features not covered in other categories
3. **Alley:** Type of alley access
4. **Fence:** Fence quality

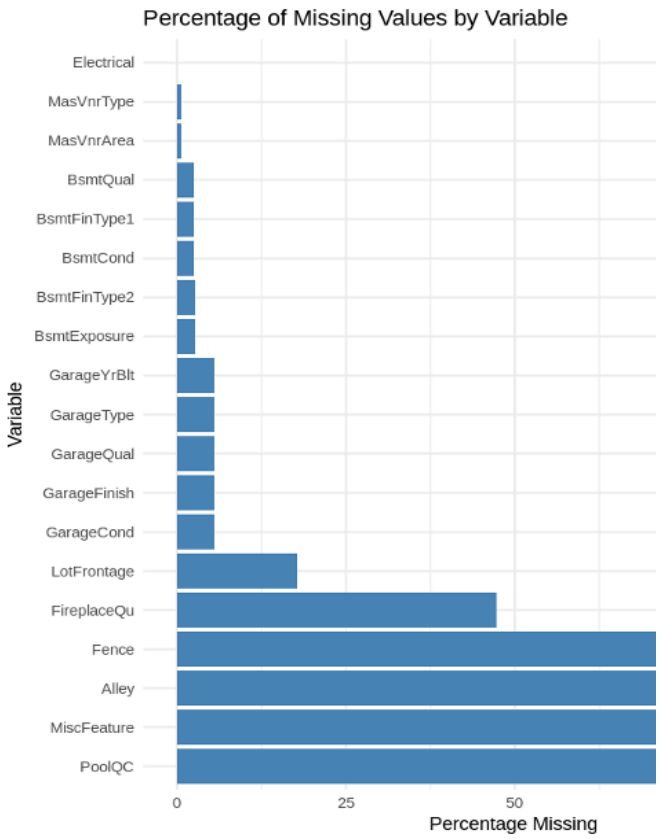


Figure 1. Missing Values by Percentage

As shown in figure 1, variables with more than half missing values were eliminated because a large amount of missing values in a predictor would significantly complicate the preprocessing and imputation steps. These steps are important in decreasing bias in our model. After this step, we have only 75 predictors.

For the rest of the missing values, we used the imputation method using K-nearest neighbors (KNN) with a tuning parameter of k=5. This means we estimate missing values from its 5 nearest neighbors in the same dataset and use the related feature values to fill in the gaps. This method ensures that the value reflects the pattern of the data and keeps the relationship with the dataset, while helping us keep the dataset whole for subsequent analysis.

Dataset Distribution

The dataset contains 44 categorical and 37 continuous predictors with our target variable being continuous (Sales price). The distribution of categorical and continuous variables are shown in the graph below

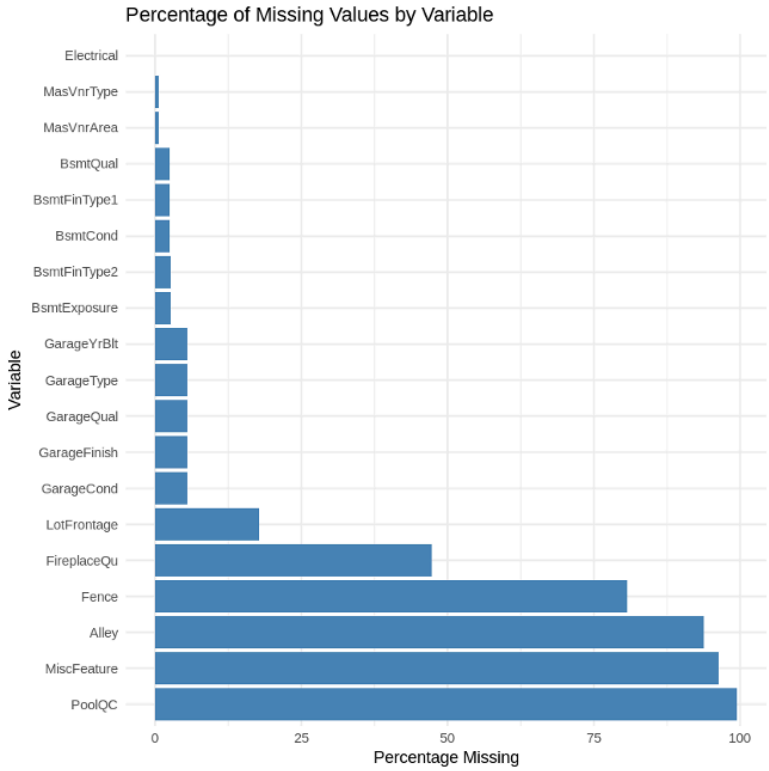


Figure 2. Distribution of Variable Types

Dummy Variables and Date Formats

Before applying dummy variables and converting the categorical variables to numeric, we took a series of steps to handle and transform temporal variables into simpler forms and observe their relevance with the target variable (SalePrice).

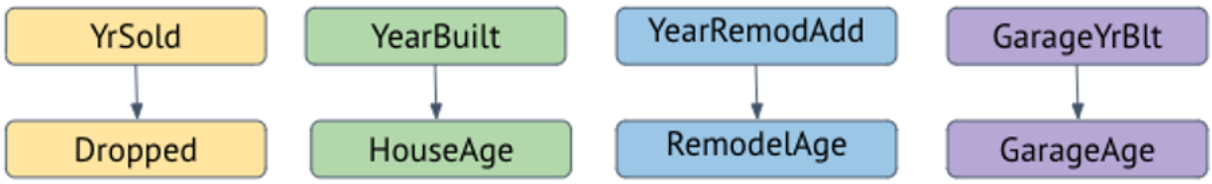


Figure 2.1 Transformation of Temporal Variables

These steps are as follows:

1. **Correlation Analysis**:
   We calculated the correlations between our temporal variables (YrSold, YearBuilt, YearRemodAdd, and GarageYrBlt) and the target variable using the cor() function to assess their relationship. Variables with weak or negative correlations means that they are less predictive and are better to be removed.
2. **Removing Irrelevant Variable**:
   Based on figure 2.1, YrSold (year the house was sold) showed a **negative correlation** (-0.0289) with the target variable and its entire column was removed from our dataset. This suggests that the year of sale does not positively contribute to the prediction of the target variable.
3. **Transforming Temporal Variables**:
   The remaining variables that contain dates are transformed into **age-related features** to simplify them and ensure they reflect their potential impact on housing prices in an easier way when it is used later in our model building process:
   - **HouseAge**: This was calculated by taking the difference between the YrSold and YearBuilt to get the age of the house at the time of sale as an output.
   - **RemodelAge**: Same calculation was performed to find the difference between YrSold and YearRemodAdd to get the time passed since the last remodeling.
   - **GarageAge**: Similarly the difference of YrSold and GarageYrBlt was calculated to represent the age of the garage at the time of sale.

   This step was crucial as it makes the variables more numerically meaningful by focusing on the age of the features rather than having to deal with date structured variables.

4. **Dropping Original Yearly Columns**:
   Once the age-related features are created, we no longer need the original columns (YearBuilt, YearRemodAdd, GarageYrBlt, and YrSold), therefore, they are removed from the dataset using the select() function from the dplyr package. This step prevents redundancy and reduces dimensionality.

Our dataset also contained predictors that were numerical but had to be treated as categorical variables. These predictors were as follows:

Figure 2.2 Predictors Converted to Numeric

As shown in Figure 2.2, the above predictors had to be handled correctly so they are treated as categorical variables, converted to numeric variables and then to dummy variables to ensure consistency across the dataset.

The total number of predictors increased to 273 after converting them to dummy variables.

Near-Zero Variance

For the predictors that show very little variation across observations, as they have a huge number of 0 values, we apply near-zero variance. This step ensures that the predictors that are problematic and provide almost no useful information are handled to avoid negative performance impact later in the model building. After removing 159 predictors that contained near-zero variance we were only left with 114 predictors.

Correlation Matrix

To handle multicollinearity among our predictors, the first step taken was to evaluate the relationships between all pairs of predictors in the dataset by computing a correlation matrix.

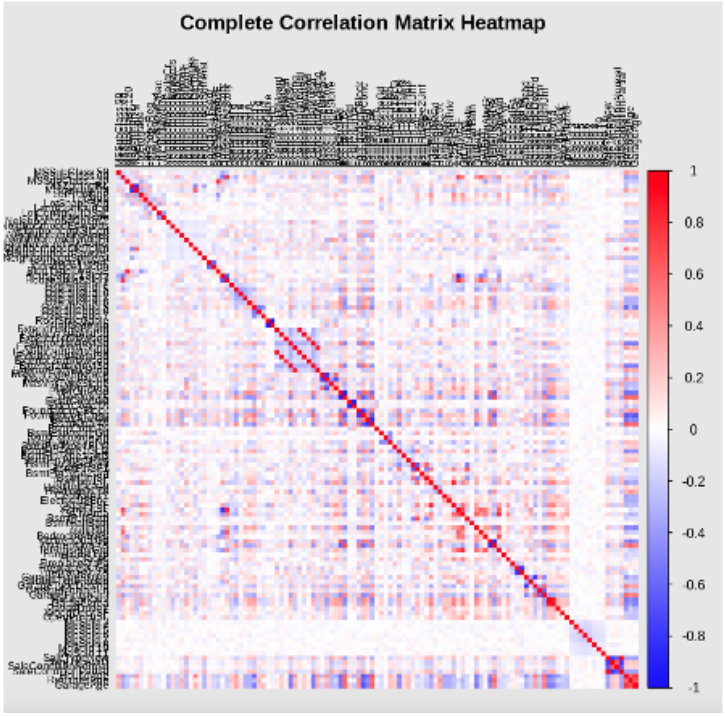


Figure 3. Correlation Matrix Heatmap

Using a correlation coefficient of 0.9, we identified those highly correlated predictors. We can observe these predictors from Figure 3 colored in red, indicating they have the highest collinearity. The following predictors were identified:

1. RoofStyleHip
2. ExterQualTA
3. Exterior1stMetalSd
4. Exterior1stVinylSd
5. SaleTypeNew

These highly correlated predictors were removed, leaving us with a final set of 110 predictors. This step improves the model's interpretability and ensures that no variable excessively influences the model due to redundancy.

Data Splitting and Resampling

To ensure a robust evaluation of model performance, the data was split into training and testing sets using a random 80/20 split ratio because our target predictor is continuous. This division resulted in 1168 samples for training and 292 samples for testing. The training set was used for model development and cross-validation, while the test set was reserved for final performance evaluation.

Resampling Strategy:

To further validate the models and minimize overfitting, 10-fold cross-validation was used on the training data. This method divides the training data into 10 subsets, using 9 for training and 1 for validation in each iteration, which ensures comprehensive evaluation of our model.

Distribution of the Target Variable:

After inspection of the target variable, we identified that it was heavily rightly skewed, this could significantly impact the predictive performance of our models.

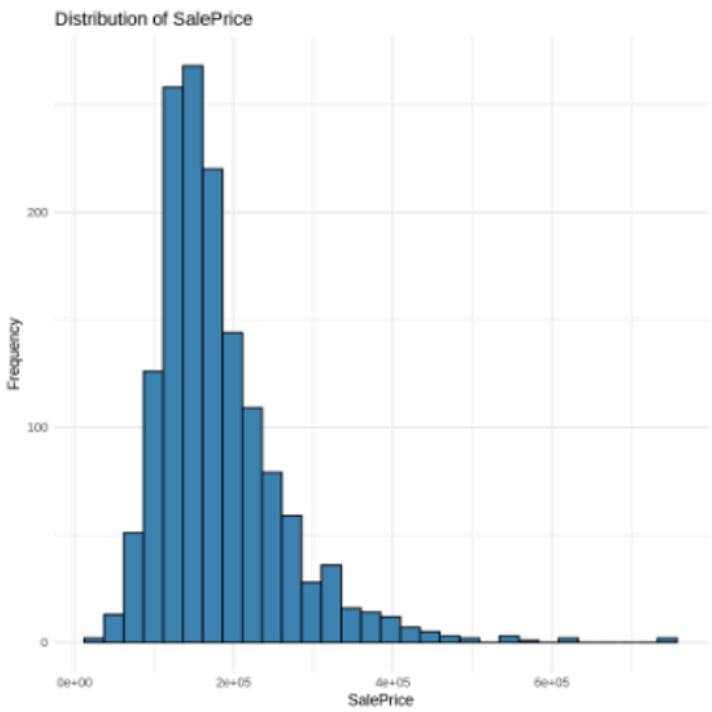


Figure 4. Distribution of SalePrice

Box-Cox transformation for the Target Variable

To handle the skewness issue, we applied Box-Cox transformation and created a more normally distributed target variable. First step to reach this goal was to use a linear model and fit it using all the predictors from the training dataset without eliminating highly correlated variables. This was necessary to understand the relationship between the predictors and the target for our transformation process.

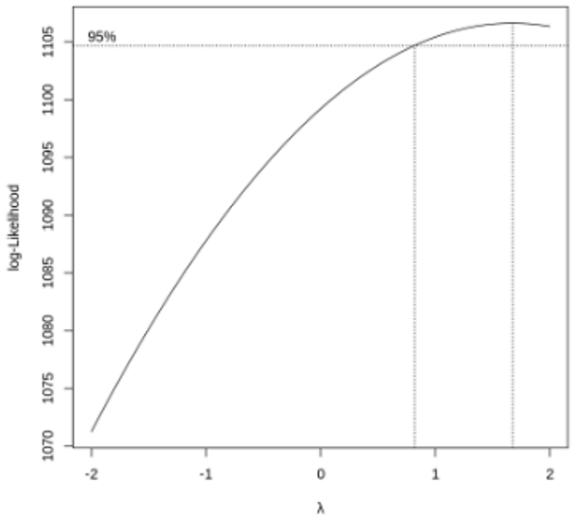


Figure 5. Box-Cox Transformation Log-Likelihood

Box-Cox method explores lambda range from -2 to 2, to find the optimal transformation parameters to minimize skewness of the linear model. The method that secures the transformation enhances the assumption of linear regression, normality of errors and homoscedasticity by methodically testing these lambda values.

The model residuals with Box-Cox applied show reduced skewness and better homogeneity of variance, so the assumptions of normality and homoscedasticity are met much better.
After applying the above transformations, the skewness of our response variable was changed from a value of around 1.88 to -0.219, which indicates a relatively normal distribution. Additionally, our models didn't require BoxCox or spatial sign transformation for the rest of the predictors, therefore, it is worth mentioning that we skipped this preprocessing step.

Models

To predict housing prices, a range of both linear and nonlinear predictive models were explored. This approach allowed for the evaluation of diverse modeling techniques to find the most accurate and robust method for price estimation.

Linear Predictive Models:

1. **Ordinary Least Squares (OLS)**: A baseline linear regression model assuming no multicollinearity among predictors.
2. **Ridge Regression**: Incorporates regularization to address multicollinearity and improve generalization by penalizing large coefficients.

3. **Lasso Regression**: Adds regularization with feature selection by shrinking less relevant coefficients to zero.
4. **Elastic Net (E-NET)**: Combines Ridge and Lasso penalties to balance regularization and feature selection.
5. **Principal Component Regression (PCR)**: Reduces dimensionality by using principal components of predictors for regression.
6. **Partial Least Squares Regression (PLS)**: Balances predictor reduction and target correlation to improve regression accuracy.

Non-Linear Predictive Models:

1. **Multivariate Adaptive Regression Splines (MARS)**: Captures non-linear relationships by creating piecewise linear fits.
2. **Support Vector Machine (SVM)**: Maps data into higher dimensions to capture complex patterns with kernel functions.
3. **K-Nearest Neighbors (KNN)**: Predicts target values by averaging outcomes of the nearest neighbors based on distance.
4. **Neural Network (NN)**: Models complex relationships using layers of interconnected nodes and non-linear activation functions.

Each model was evaluated using cross-validation on the dataset, and the best-performing model was selected based on prediction accuracy and their ability to generalize.

Models Results

Linear Model Performance

The performance of six linear predictive models was evaluated on both the training and test datasets. The metrics used to assess these models include Root Mean Squared Error (RMSE) and R-squared () values. A summary of the results is as follows:

Models	RMSE (Training Set)	R-squared (Training Set)	RMSE (Test Set)	R-squared (Test Set)
OLS	0.1634916	0.8348211	0.1585286	0.8606519
Ridge	0.1616538	0.8404443	0.1521457	0.870059
Lasso	0.1607747	0.8402121	0.1532793	0.8719271
Elastic Net	0.1600687	0.8424046	0.1512562	0.8786557
PCR	0.1793181	0.8007603	0.1687404	0.8445183
PLS	0.1613994	0.8384435	0.1507725	0.8778174

Table 1. Comparison Table of Linear Models

- **PLS** and **Elastic Net** models demonstrated the best overall performance, with the lowest RMSE and highest values on the test dataset.
- **Lasso** also performed competitively, balancing regularization and predictive accuracy.
- **PCR** showed relatively higher RMSE and lower values, suggesting less optimal performance compared to other models.
- Regularized models like **Ridge**, **Lasso**, and **Elastic Net** provided significant improvements over the baseline **OLS**model by reducing overfitting and improving generalization.

In conclusion, we achieved the best performing linear models as PLS and Elastic Net, and the worst performing linear regression model as PCR, particularly.

Non-Linear Model Performance

The performance of four non-linear predictive models was evaluated using Root Mean Squared Error (RMSE) and R-squared () metrics on both the training and test datasets. The results are summarized as follows:

Models	RMSE (Training Set)	R-squared (Training Set)	RMSE (Test Set)	R-squared (Test Set)
MARS	0.1649153	0.8321994	0.1673047	0.8430930
SVM	0.1307137	0.8903392	0.1460841	0.8828786
KNN	0.1818747	0.7971668	0.1907653	0.8181342
Neural Network	0.1460579	0.8632528	0.1437893	0.8855455

Table 2. Comparison Table of Non-Linear Models

- **SVM** achieved the best overall performance, with an RMSE of 0.1307137 and an of 0.8903392 on the test set, showcasing its ability to capture complex patterns in the data.
- **Neural Network** also performed well, with the lowest RMSE 0.1460579 and the highest value 0.8632528 on the test set, indicating strong predictive accuracy.
- **MARS** demonstrated moderate performance, with a balance between training and test results, but it underperformed compared to Neural Network and SVM.
- **KNN** had the highest RMSE and the lowest values among all models, suggesting it is less suited for this dataset's predictive tasks.

Best Two performing Models

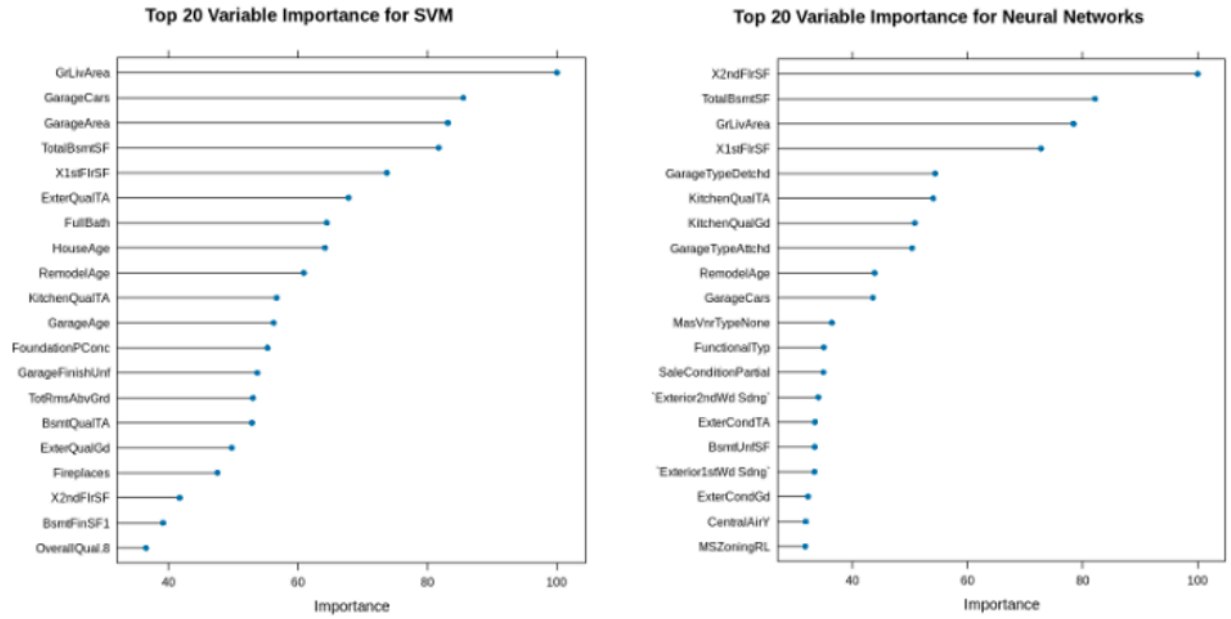
Models	RMSE (Training Set)	R-squared (Training Set)	RMSE (Test Set)	R-squared (Test Set)
SVM	0.1307137	0.8903392	0.1460841	0.8828786
Neural Network	0.1460579	0.8632528	0.1437893	0.8855455

Table 3. Top Two Performing Models

Both models effectively capture the intricate relationships in the dataset and are highly suited for the task. Among the two, the Neural Network slightly outperformed SVM on the test data, establishing itself as the optimal model for this analysis.

Importance Plots

The image shows two plots highlighting the top 20 most important variables for predicting housing prices using two different machine learning models—**Neural Networks** and **Support Vector Machines (SVM)**—based on the Ames Housing dataset.



Neural Networks (NN)

This plot ranks the variables in terms of their importance as determined by a trained neural network model. Neural networks often derive feature importance based on how much a variable contributes to the predictive performance across the hidden layers. We observed the following important features from the Neural Network's Importance Plot:

- **X2ndFlrSF** (Second Floor Square Footage) and **TotalBsmtSF** (Total Basement Square Footage) are the top contributors, indicating that the size of these areas is critical in determining housing prices.
- Variables like **GrLivArea** (Above Ground Living Area) and **GarageTypeDetchd** (Detached Garage) also play a significant role.

- Features like **KitchenQualTA** (Kitchen Quality) and **FunctionalTyp** (Functionality) highlight that condition and quality measures are important in this model.

Support Vector Machines (SVM)

This plot shows the importance of variables as identified by the SVM model. SVM uses support vectors and kernels to separate classes or predict outcomes, so the importance is derived from how these variables influence decision boundaries. Observed features from the Importance Plot of SVMs are as follows:

- **GrLivArea** (Above Ground Living Area) emerges as the most important feature, highlighting its strong relationship with housing prices.
- Variables like **GarageCars** (Number of Cars that Fit in Garage) and **GarageArea** (Garage Size) are prominent, showing the importance of garage-related features.
- Quality-related variables like **ExterQualGd** (Exterior Quality) and **KitchenQualTA** also feature prominently.

Comparison Between Models:

- Both models emphasize **GrLivArea**, **GarageCars**, and **KitchenQualTA**, indicating their consistent influence across different predictive methods.
- Neural networks seem to focus more on detailed area-specific variables (e.g., **X2ndFlrSF**, **TotalBsmtSF**), while SVM places slightly more emphasis on quality and condition metrics (e.g., **ExterQualGd**, **OverallQualB**).

Summary

These plots provide the importance of Neural Network and SVM models in capturing the intricate relationships within the dataset that provides interesting and valuable insights into the factors influencing housing prices in Ames. The analysis also highlights the importance of regularization techniques and the role of feature selection in improving model performance.

References

Brownlee, J. (2020a, August 17). How to avoid data leakage when performing data preparation.

MachineLearningMastery.com.
https://machinelearningmastery.com/data-preparation-without-data-leakage

Kaggle. (n.d.). Home Data for ML Course. Kaggle. Retrieved December 9, 2024, from
https://www.kaggle.com/competitions/home-data-for-ml-course

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.
http://appliedpredictivemodeling.com/