## horizontal line



House Pricing Regression Analysis

16.08.2024

**─**

Abdulmujeeb Sakibu

ALT/SOD/023/1147

Capstone Project

# Overview

The point of this project is to look through and be capable of predicting what the average price of a house, would go for in the area. This project could benefit a city planning committee in helping to monitor and decide the price of housing properties. It could also help a potential house buyer to figure out, how much the kind of house he/she is seeking and plan towards that goal.

# Goals

1. Prepare the dataset and improve on it so as to make the model as accurate as possible
2. Utilize Multiple models and train the data on the best one
3. Drop as few columns as possible from the dataset

# Methodology

## Importing dependencies

First is the imports, I imported basic dependencies like Numpy and Pandas for manipulating the dataset, Seaborn and Matplotlib for visualizations, and different functions from the Scikit-learn library for scaling the data training the model.

## Downloading the Dataset

The dataset is from Kaggle and was downloaded to my Google Drive. This means I can download everything I need online directly to my notebook for the whole Project.

The dataset was a mix of different datatypes and had a shape of (1460, 80).

I also check basic information on the dataset using the Pandas modules like .head .describe .shape etc.

1. Missing data

There were 19 columns with varying amounts of missing data. For columns with missing data above 70%, I dropped it directly since it would be hard to do anything with it.

For the others, I either filled it up with ‘None’ or 0 since that was what the dataset was trying to convey. I did use the mean to complete the Frontage column and I dropped a row for the electricity column since it only has a single missing column.

1. Explorative Data Analysis

I made the decision to focus my EDA on the target column, in this case, SalePrice. This is because it was my goal and the only column i am preparing the model for.

I drew up a histplot on the SalePrice column and realized that the prices were heavily skewed with outliers on the high side of the sales. I decided to use Natural Log to standardize the column instead of capping it because the prices of houses, while averaged at 150,000, can still definitely go up as high as a million.

By standardizing the column, i could keep the original prices (albeit now in their Natural Log form) while also preventing the model from heavily skewing the data on training.

I also checked how the other columns feared with each other and while there was some skewing seen on some of the numeric columns, it was not as bad.

In total, all the columns had a positive relationship with the SalePrice column which is great for our training

1. Feature Engineering

My feature engineering focused a bit on the categorical columns, especially the ones with data type ‘object’. This is to avoid the model throwing errors while trying to read through the dataset for training

I had to first look through and drop (20) columns that had too many bais on their categories (one category carrying 80% on the column).

Then for the other 19, I used the get dummies function to split them into dummies before adding them back, alongside the other numeric columns, to a new dataframe.

My dataset now has a shape of (1459, 159)

1. Scaling and Splitting

I used the Scikit-Learn standard scaler function to scale the independent variables before using the train\_test\_split function to split them with a test split of 20%.

1. Model Training and Test

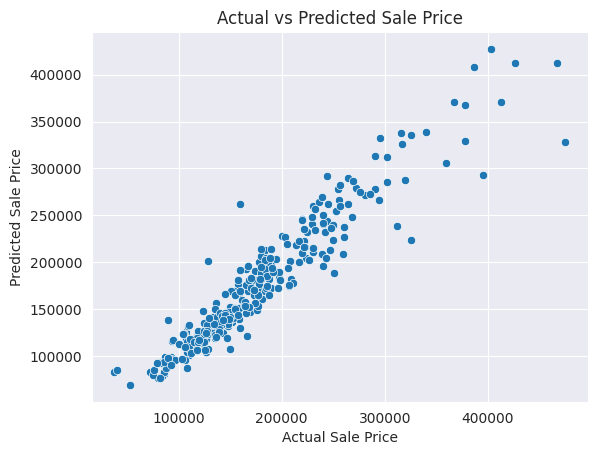
For the models, I got the:

* Linear Regression model
* Gradient Boosting Regression model
* Support Vector Regression model
* Decision Tree Regression model
* Random Forest Regression model, and
* XGBoost Regression model

All these are linear regression algorithms and I evaluated them all with cross-validation test functions on my new dataset. The best-performing algorithm was the Gradient Boosting Regression model and this is due to the high amount of columns my dataset now has after the feature engineering.

1. Model Evaluation

After training the model, i used the function MSE function to get both the MSE and the RMSE. and they were very low at 0.01687 and 0.12988 respectively. I also calculated the R2 which is used to know how accurate a regression model is and got a score of 0.8925.

Lastly, i ploted my predictions against the original result (y\_test) and got a pretty linear graph, showing that my results were very close 

1. Conclusion

I was able to achieve most of my goals in the end by creating a pretty accurate dataset. Although I ended up discarding 24 of the 77 columns, that was a pretty good result considering how much work the dataset needed.

I also ended up adding more than 100 columns to the already wide dataset.

With a system built around this dataset, Users can almost accurately predict what a house might go for on the market and if possible, save towards or make decisions based on it that would save them a lot of time waste in the future