House Prices Prediction

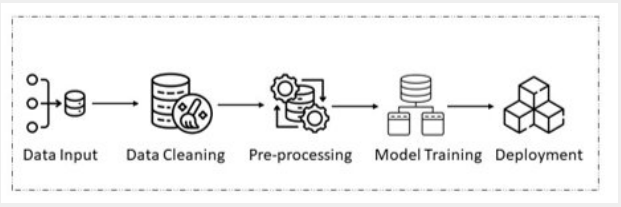
- Advanced Regression Techniques

Author:
Yug Bargaway,
Undergrad @ CSE,
IIT Kharagpur, India
(yugbargawayskills@gmail.com)
Github: House_Price_Notebook

Abstract:

This project tackles the regression problem of predicting house sale prices using a dataset provided by Kaggle's "House Prices - Advanced Regression Techniques" competition. The dataset contains diverse features ranging from structural attributes and neighborhood information to access the quality of the properties. The analysis involves a thorough data preprocessing phase. Exploratory Data Analysis (EDA) was conducted to uncover patterns and feature-target relationships. Multiple regression models, including regularized linear models, ensemble methods, and stacking techniques, were evaluated using cross-validation with logarithmic root mean squared error (RMSE) as the primary metric. This project demonstrates a complete machine learning pipeline and highlights the importance of feature engineering and model tuning in predictive modeling tasks.

Table of Contents:



1. Introduction to the Problem Statement:

The real estate market is **influenced** by a wide variety of **factors** ranging from property size and location to age, condition, and neighborhood amenities. This project focuses on solving the house **price prediction** problem. The goal is to **build** a machine learning model that can predict the final **sale price** of a house based on these features. This is a supervised regression problem where the target variable is continuous, and the model must generalize well on unseen data.

Through this project, we **aim** to understand:

- How different house features impact sale price.
- The effectiveness of different regression models.

2. Data Overview:

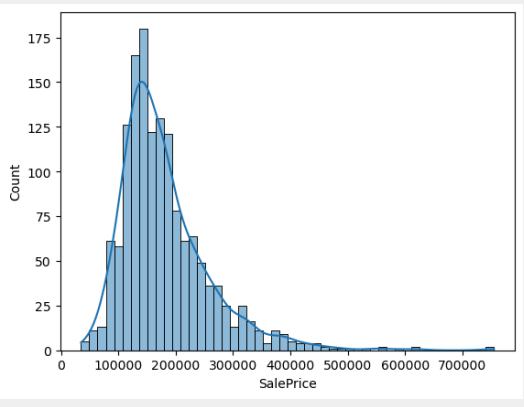
2.1. Understanding the Data:

The data we are provided with, has **81 columns and 1460 entries/rows** in it, containing **38 numerical values** (int and float) and **43 categorical values**. The data contains very few null values, but some columns, namely, Alley, Masvnrtype, Fireplacequ, PoolQC, Fence and MicsFeatures, contain a huge number of missing values which needs to be fixed before putting it into an algorithm. There are **no duplicate** entries. The correlation matrix of the data is demonstrated in the notebook file and cannot be displayed here due to the large number of columns. It shows some really strong correlations of SalesPrice with some columns.

3. Exploratory Data Analysis:

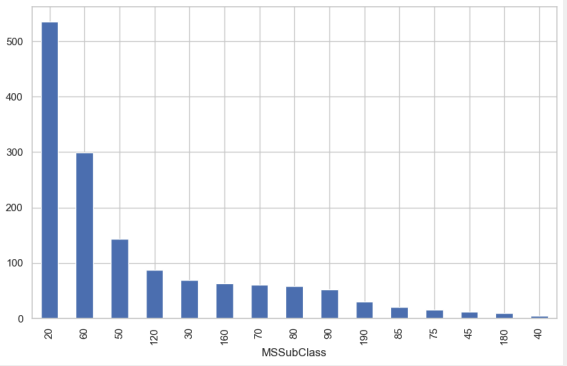
3.1. Univariate Analysis:

Histogram and Count plot of all input fields are available in the notebook file.

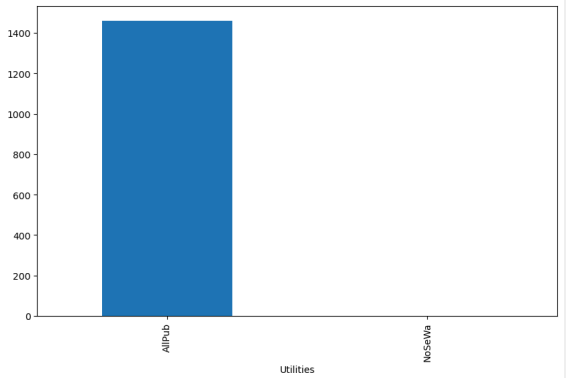


The analysis of **SalePrice** and plotting it represents, there are considerable sales of houses, priced in the middle range and the data is **skewed** to the left.

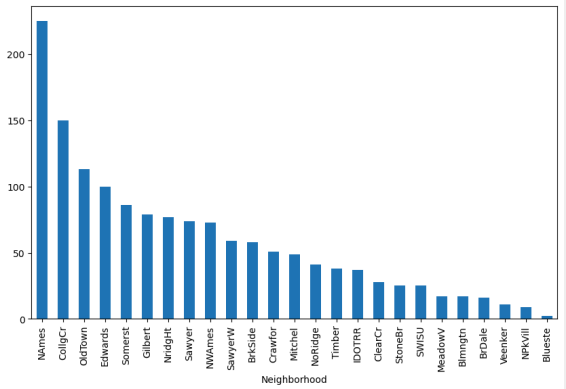
- The Analysis of “**Type of House**” represented by “MSSubClass” portrays that distribution is **skewed** towards houses in small buildings and apartments.



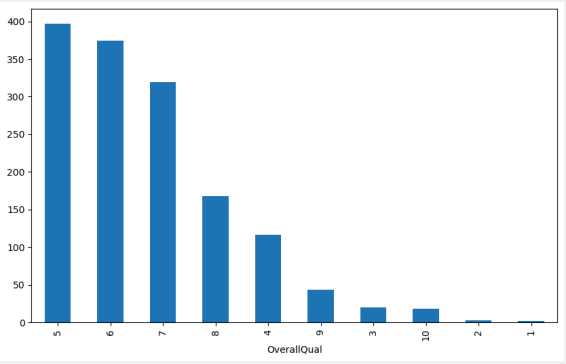
- Zoning Classification represents that **most** of the **houses sold** are in the **low density residential region** and there are no sales in Agricultural and Industrial regions.
- Distribution of **Plot Area** and **area of street** connecting the plot are both **highly skewed** towards the left.
- According to the data, mostly all the houses (>**99%**) are **connected** with the **paved roads** and thus this field will not be required for our analysis as it is biassed towards one side only.
- Most of the property has **no access to the alley** and very few plots (5 - 8 %) are connected with the alley, equally distributed on gravel and paved.
- Sales of the houses are **highly affected by the utilities provided** and mostly data is skewed towards all utilities and thus, this field is not very effective in sales price calculation.



- Type of building data shows, most of the houses sold are made as a single family house and are predominantly single or double story.
- Neighbourhood data represents, plots values do depend on neighbouring areas and are distributed nearly equally among areas.



- Overall **quality and condition** of the houses sold mostly lie in the range **5 - 8** (in the scale of 10).
- Overall trend shows that most of the houses have average external condition and material quality.
- Most of the houses sold contain **Gas forced** warm air **furnaces** as the primary heating source.
- Floor area represents mostly houses that are one storey only.



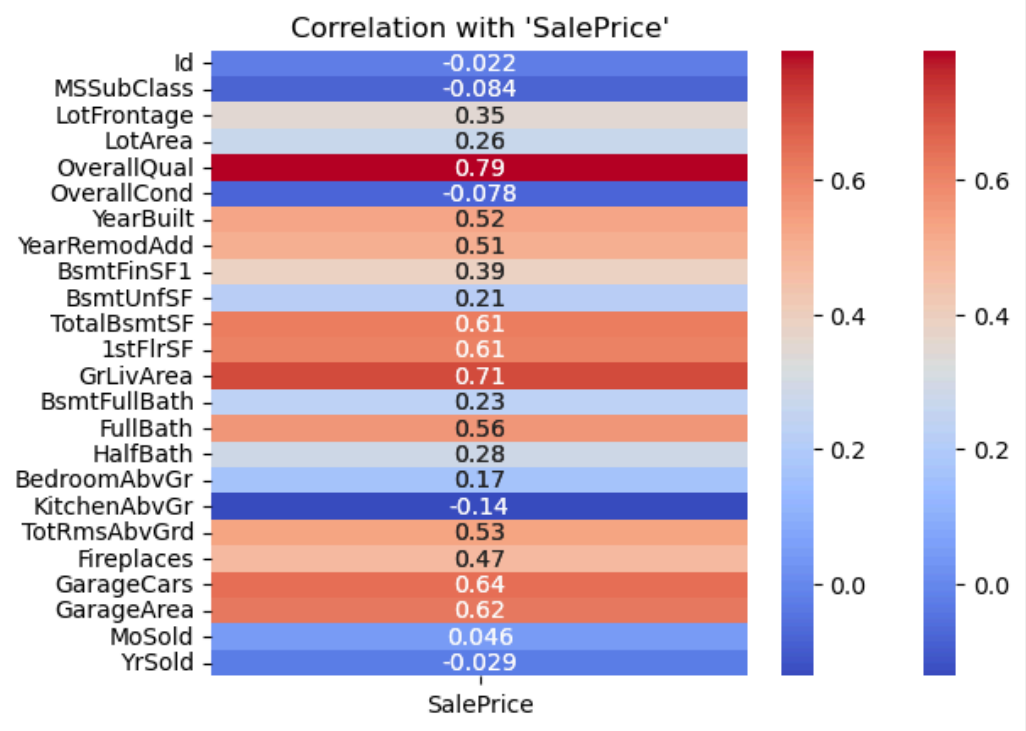
- The **Data as a whole represents** that most of the **regressive fields** are **skewed slightly to the left** and most of the **categorical** fields have **average** values as the dominating values.

Conclusively, we can obtain that there are **some columns** which are either **irrelevant** for our analysis or are **highly skewed** to be used in our model. Those columns are, thus, **removed** from our training data. Thus, we have 49 columns of data left with us, with **17 numerical** columns and **32 categorical** columns.

Column names removed are 'Street', 'LandContour', 'Utilities', 'LandSlope', 'Condition1', 'Condition2', 'RoofMatl', 'MasVnrArea', 'BsmtCond', 'BsmtFinType2', 'BsmtFinSF2', 'Heating', 'Electrical', '2ndFlrSF', 'LowQualFinSF', 'BsmtHalfBath', 'Functional', 'GarageYrBlt', 'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC', 'MiscVal', 'SaleType', 'SaleCondition'.

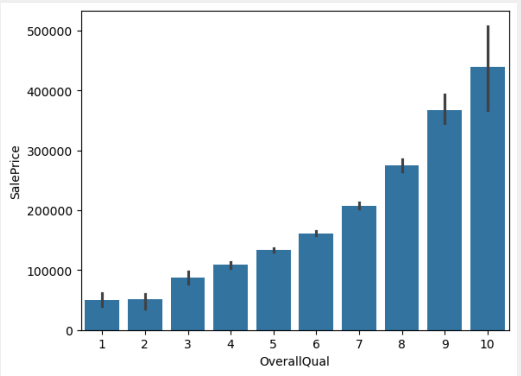
3.2. Bivariate and Multivariate Analysis:

In-depth analysis of different columns with **sales price** is done and represented in notebook file as well as report webpage, both are present in the github directory from where it can be accessed.



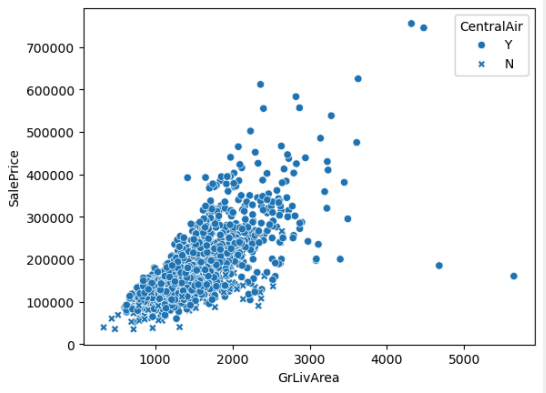
From the analysis of sales price with other attributes we have in the data, we can infer the following:

- Overall **Quality** and **Condition affects** the final price of the house **greatly.** In addition to this, Lot Area and Frontage have much effect.



- People **care less** about the **type** of the building, whether it is 1-storey, 2-storey or something else and more care about the year in which it is built.

- There is a **direct proportionality** between **sale price** and above **grade living area**.
- There is no correlation of sale price with the time when the property is sold, that is, the month and year.
- **Presence of a basement** in good quality and condition **increases sales price** as the size of the basement increases.



- The **availability** of a fireplace, number of **fireplaces**, availability of a **garage** and its size also **affect sales price**, which is intuitive and it is a good thing that our basic intuitions are aligning with the data, thus verifying the correctness of our data.
- Area of the **first floor** is **highly correlated** with SalePrice.

Some of the highly correlated row are:

Field	SalePrice Correlation
SalePrice	1
OverallQual	0.81
GrLivArea	0.731
YearBuilt	0.653
GarageArea	0.649
TotalBsmtSF	0.603

Some of the least correlated row are:

Field	SalePrice Correlation
MiscFeature	0
MSSubClass	0.007
KitchenAbvGr	0.046
MoSold	0.069
LotConfig	0.087
BldgType	0.088

With the help of **Bivariate analysis** and **correlation matrix**, we can conclude that these six fields are not affecting the sales price of our house and thus, it is safe to remove them from our data to further reduce the complexity of our analysis.

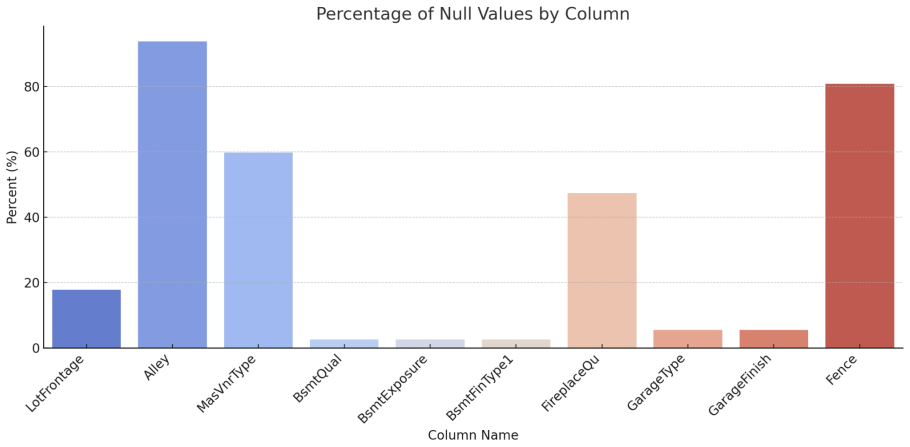
Thus, we have 43 columns of data left with us, with **17 numerical** columns and **26 categorical** columns.

4. Feature Engineering:

As part of the preprocessing, we have successfully eliminated numerous columns from our data to reduce the complexity of our model. We, furthermore, preprocess the data, add and transform several features, **standardize and normalize** the values of features, **handle missing values and outliers**, and if required, construct new features.

4.1. Handling Missing and Null Values:

With the univariate analysis, we concluded there are some columns which **contain missing** or null values which are needed to be catered before putting it in our model. The changes we perform to handle these cases are listed,



- Area of the front pathway (LotFrontage) contains **259 (17.7%)** null values. These values indicate there is **no proper front pathway** to the house. Thus, we **replaced** those with 0.
- Alley columns contained **1369 (93.8%)** null values. This indicates there is **no alley** present. Since it is a categorical column, we replaced null with None.
- The Masonry veneer type column has **872 (59.7%)** missing values. These represent there is **no masonry** present in the outer walls. Thus we replaced the null values with None.
- BsmtQual has **37 (2.5%)** missing values, BsmtExposure has **38 (2.6%)** missing values, BsmtFinType1 has **37 (2.5%)** missing values. This shows there is **no basement** in these houses.
- FireplaceQu had **690 (47.3%)** missing values, indicating **no fireplace** in the house.
- GarageType and GarageFinish, both, have **81 (5.5%)** missing values, indicating **no garage** presence.
- Fence has **1179 (80.8%)** missing values, indicating **no fencing** in the house.

Thus, we satisfactorily removed all the missing and null values from the dataset.

4.2. Encoding Categorical Data:

Categorical Data are object type data in which data is distributed in categories. It **needs** to be converted to **mathematical form** before giving it to the model. These are two ways by which we can perform the operation, ordinal encoding and one hot encoding.

4.2.1 Ordinal Encoding

There are some categorical values which follow a clearly defined hierarchical order which are converted according to the hierarchy.

The Columns, mentioned below, contain categorical values which follow **strict hierarchical** order of **poor to excellent**, which are scaled accordingly in the **scale** of integers. The scale provides excellent as the maximum value and poor as 0. All the other **intermediate** values are equally **distributed** in the range.

Ordinal Encoding is **performed** in the following **columns**: LotShape, ExterQual, ExterCond, BsmtQual, BsmtExposure, HeatingQC, CentralAir, KitchenQual, FireplaceQu and GarageFinish. The property they represent can be read from the data description.

4.2.2 One Hot Encoding

Apart from those processed under ordinal encoding, other categorical values are processed with one hot encoding method. These columns contain categorical values **without any order** and categories are independent to each other.

There are many columns which have many **small categories** which are of less significance as compared to larger categories. To control the complexity of our model, we took those categories and **clubbed** them as **others**.

- In MSZoning, “FV, RH and C (all)” contribute **only 4.4%, 1.1% and 0.7%** values respectively. They are clubbed as Others.
- In Alley, “Grvl and Pave” contribute only **3.4% and 2.8%** values respectively.
- Neighborhood column is **highly diversified** and thus there is no change in the entries of this column.
- Similarly in all other specified, if the column is not highly diversified, all values with less than 8% entries are classified as others and are **properly replaced**.

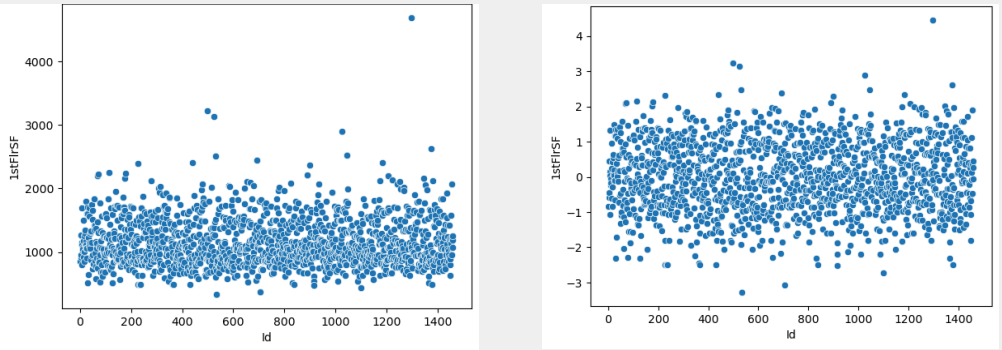
The columns, mentioned, are processed with one hot encoding and new columns of separate categories are modified in our dataset, replacing previous ones, thus, increasing a lot of columns and complexity of our model. To **avoid** the problem of **Multicollinearity**, we **removed the first column** of every bunch of encoded columns. We obtained a total of **94 columns** in our data.

One hot encoding is **performed** in the following columns: MSZoning, Alley, Neighborhood, HouseStyle, RoofStyle, Exterior1st, Exterior2nd, MasVnrType, Foundation, GarageType, BsmtFinType1 and Fence.

4.3. Feature Scaling: Standardization and Normalization:

We have a lot of columns in our dataset, each with a **different range** of values. We must standardize or normalize the data before putting in our model. Here, we are choosing the **approach of standardization** for our data.

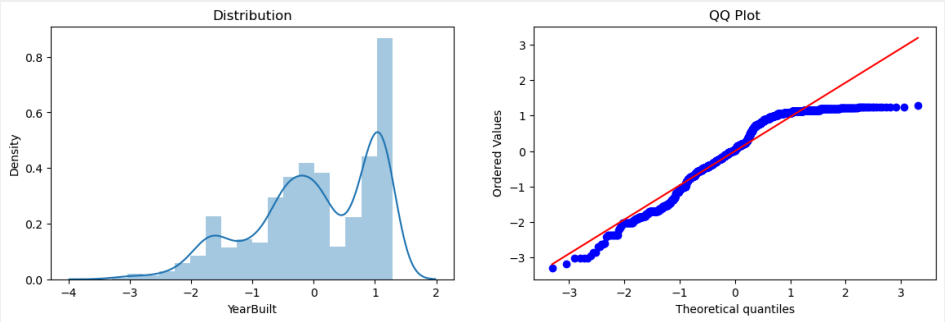
We standardized all the columns such that the values of the column, now, lie in the range -1 to +1 with **mean** equal to 0 and **standard deviation** equal to 1.



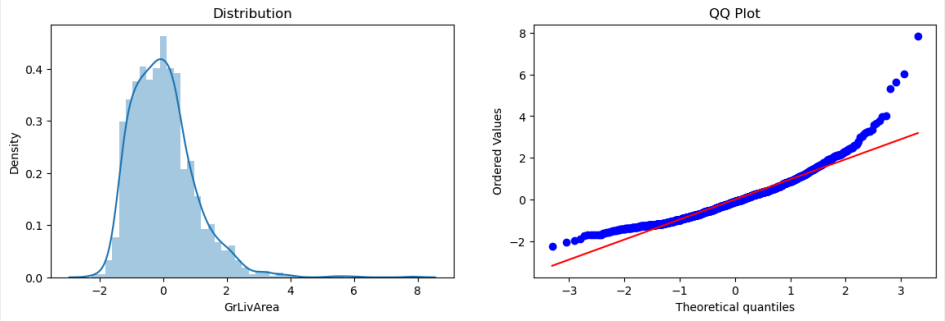
4.4. Feature Transformation:

Regression algorithms work **best** when probability **distribution** of the feature follows **gaussian** distribution. It is very important to transform the feature using some functional transformer so as to **shift** the probability distribution and **handle skewness** of the distribution.

The **QQ Plot** is a best way to pictorially represent closeness to gaussian distribution. The more the distribution aligns with the line, it is closer to gaussian distribution. QQ Plots of all tables are available in notebook file, a few are represented here for analysis purpose.



As shown, the distribution of year built shows skewness to the left. Proper **transformation** is **required** to make it closer to gaussian distribution.

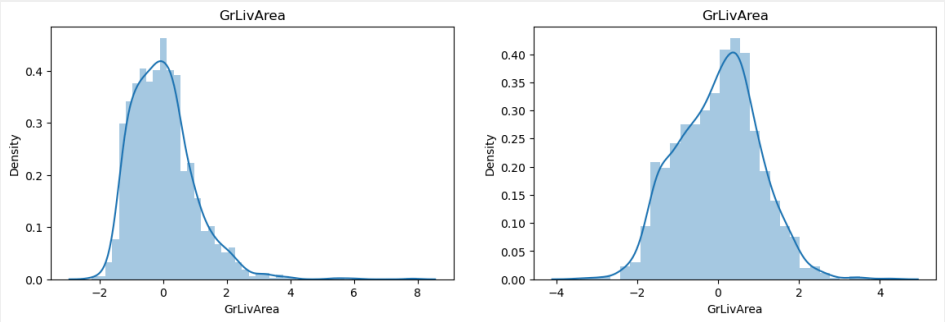


Probability distribution of Gross Living Area clearly shows **skewness to the right**. Applying **log** transformations are beneficial for these types of distributions.

We have already standardized the data and thus all columns now range in -1 to 1. Due to the occurrence of negative values, we will be applying **Yeo - Johnson transformation** for our model. We will raise the values of features by some power, ranging between -5 to +5 and thus, try to reduce skewness of our data.

Values of raised power of some features are represented beside. All other values can be accessed from the notebook. After applying the transformations, the old and new distributions are plotted and attached for reference and analysis purposes.

1	LotFrontage	1.030202
2	LotArea	-0.337527
3	LotShape	2.463230
4	OverallQual	0.864962

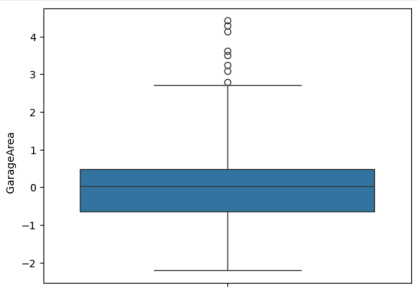


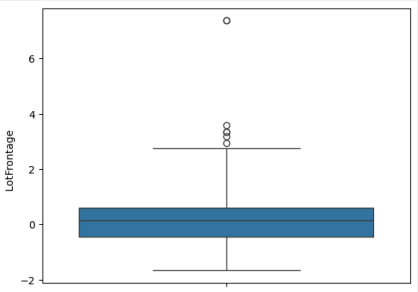
We can clearly see that this transformation caused the distribution to **behave, more gaussian** like. These comparison curves of other features are also available. We successfully added one more layer to our pipeline as a Yeo - Johnson transformer.

4.5. Outlier Handling:

Detection and handling of outliers are very **necessary** in machine learning modelling, especially when the **algorithm** used in the pipeline contains some form of **weights** to features.

Handling outliers contains various methods, mostly used are **trimming** and **capping**. We will be **using capping** in our dataset, as trimming can significantly reduce the size of our training data which is not very good for the model. As most of our regressive columns are either gaussian like or right skewed, we are using the IQR range method to detect outliers. Outliers in some features are demonstrated below. Others are accessible in notebook files.





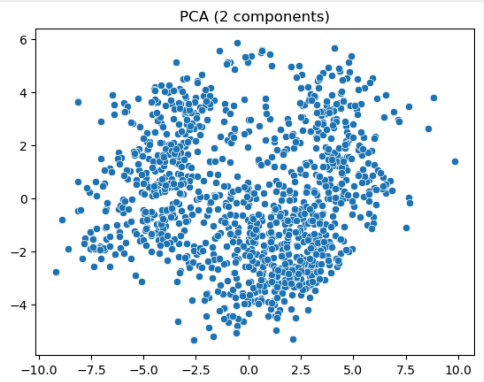
We have some outliers in our data, which we are fixing by the method of capping to the **value of 2.0 * IQR**. The features which are operated to handle outliers are: 'LotFrontage', 'BsmtFinSF1', 'BsmtUnfSF', '1stFlrSF', 'GarageArea'.

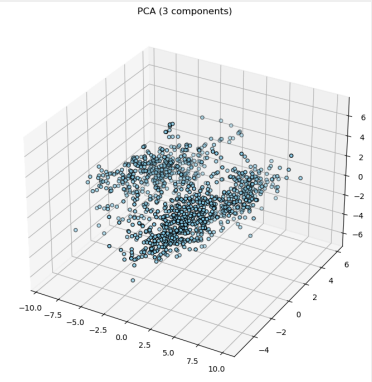
4.6. Feature Extraction:

A large number of features, that is, **high dimensional data** is not beneficial for our model and are very **computationally expensive**. From the set of our features, we are creating new features using Principal Component Analysis (PCA), thus lowering the dimension of our data and making our algorithm **run faster** and **enhancing** the **visualization** of features.

4.6.1. Principal Component Analysis

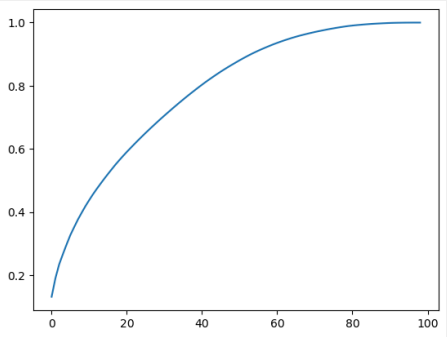
We have a lot of features in our dataset and now we are trying to **reduce its complexity** by extracting new features from existing ones. We are demonstrating below how our data behave when reduced to two dimensions or three dimensions.





We are extracting new features out of the given features. The number of features we need depend on how much variance of data we need to match. In general, **90% or more variance**

spread is taken for a comparable as well as efficient analysis. The below graph shows the **increase** of **matching covariance** when the number of components is increased.



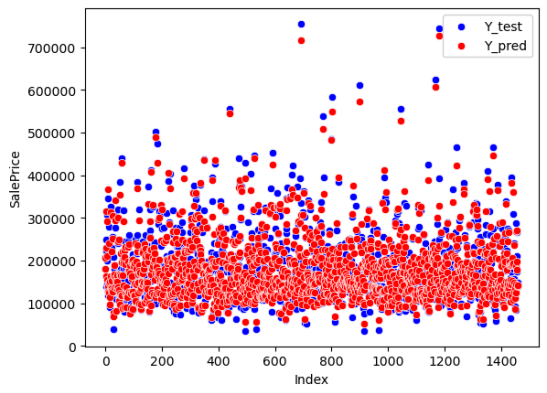
From the graph, we can conclude that to get a comparable analysis, we are required to take nearly 55 components. Thus, we obtained the data where there are **55 feature dimensions** from 91 dimensions.

5. Model Evaluation:

It is very important for our analysis to **choose the best possible algorithm** to use for the prediction. For this purpose, we need to test between various algorithms and try to find how well they predict our data using metrics.

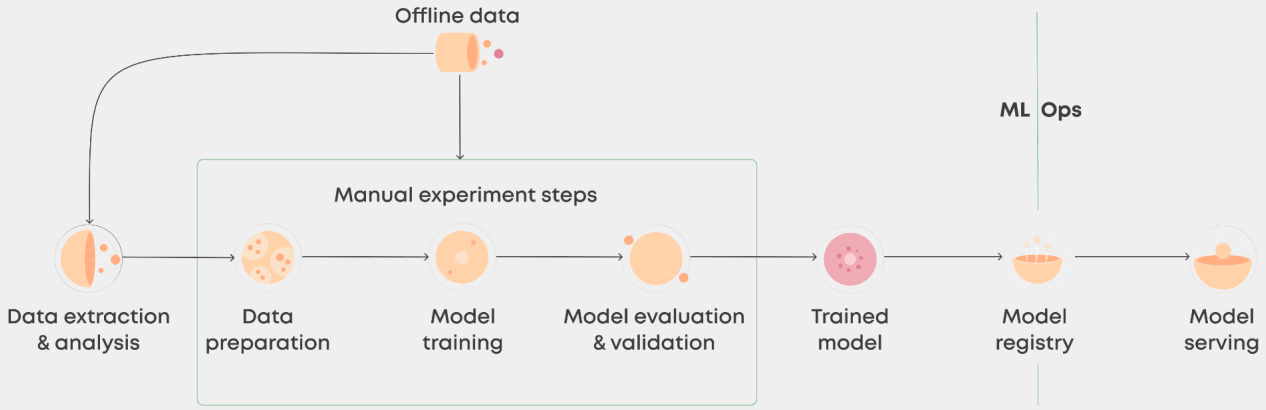
We will try to fit the following mentioned algorithms and then analyse them on the basics of **R2 Score**, **Mean Absolute Error (MAE)** and **Route Mean Squared Error (RMSE)**. We used the scikit-learn based implementations for the algorithms.

#	Model	R^2 Score	MAE	RMSE
1	Gradient Boosting	0.8765	19,250.86	30,773.96
2	Extra Trees	0.8705	19,470.81	31,521.17
3	Random Forest	0.8661	19,716.31	32,044.40
4	Lasso Regression	0.8300	21,611.72	36,107.25
5	Multiple Linear Regression	0.8294	21,600.30	36,176.11
6	Ridge Regression	0.8293	21,592.60	36,179.70
7	Decision Tree	0.8233	26,110.21	36,817.24
8	ElasticNet Regression	0.8166	21,097.91	37,505.79



From our evaluation of different ML based algorithms, we can clearly come to a conclusion that **Gradient Boosting Regressor** performs better for this dataset when transformed as we did. Now, we will move towards creation of the ML pipeline.

6. Machine Learning Pipeline:



We have **created** a c**omplete pipeline** for our test data to predict the sale price. The complete process as well as the pipeline is present in the notebook.

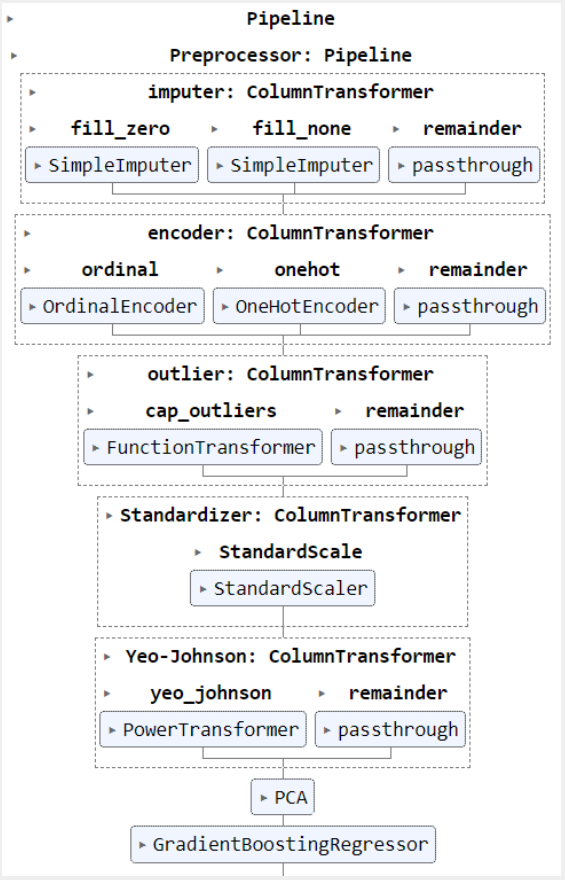
Creating a good ML Pipeline helps to **deploy models** in any **server**, in an effective manner with less errors and hassle free. We have created a pipeline with all the processing we have applied till now. The steps in the pipeline include:

- A **Preprocess pipeline** which contains the following transformers
    - A **Imputer Transformer** to impute any missing value in the testing value, which replaces categorical values with none and numerical values with mean.
    - A **Functional Transformer** to remove some rare categories from the columns.

    - A **Encoder Transformer** which performs ordinal encoding to the columns mentioned above and one hot encoding to other mentioned categorical columns.

- A **StandardScaler Transformer** which standardizes all the values in the dataset.
- A **Power Transformer** for Yeo-Johnson Transformation to make distribution look more gaussian like.
- A Transformer for **Principal Component Analysis** to reduce the number of features to 55.
- Finally, A **ML algorithm**, which is Gradient Boosting Regressor in our case to predict the sale price of the houses.

  The **visual representation** of our ML Pipeline is presented below:



The pipeline showcases the **end-to-end workflow** of the prediction algorithm, encapsulating all preprocessing steps and **model training** in a **structured and reproducible** manner. It is particularly **useful** for **deployment** on a **server** or **production environment,** ensuring that the same data transformations applied during training are consistently executed during inference. Additionally, the **complete deployment-ready code**, including the serialized model and preprocessing pipeline, is **provided** in the notebook for seamless integration and reuse.

7. Conclusion:

In this project, I built a complete Machine Learning **pipeline** to **predict house prices** for the Kaggle “House Prices: Advanced Regression Techniques" dataset. The process included **extensive data preprocessing, handling missing values, encoding categorical features, and experimenting with multiple regression models** including Linear Regression, Ridge, Lasso, Decision Trees, Random Forest, Gradient Boosting, and more. While the best-performing models currently show an R^2 score around **0.87** and a relative error of approximately **5 - 10%**, this performance reflects an **impressive** but **intentional baseline implementation** — **without heavy** feature engineering or hyperparameter **tuning** — to understand the fundamental relationships in the data and compare model behavior fairly. Key steps that contributed to this result include:

- Robust preprocessing (handling missing values, encoding, scaling).
- Target transformation using Yeo-Johnson to normalize skewness.
- Use of ensemble models like Random Forest and Gradient Boosting.
- Pipeline-based architecture for clean and reproducible modeling.

Despite the **modest scores**, the project **helped solidify** my understanding of the end-to-end **ML workflow**, including model comparison, evaluation metrics, and **pipeline design**. It also highlights the critical role of feature engineering and model **optimization** in real-world regression tasks.

Future improvements will focus on:

- Log-transforming skewed features like SalePrice and GrLivArea.
- Advanced feature engineering using domain knowledge (e.g., creating interaction terms).
- Hyperparameter tuning with cross-validation.
- Ensemble methods like XGBoost and model stacking.

This foundation positions the project for substantial **performance gains** in subsequent iterations while **demonstrating** a **structured** and **scalable approach** to regression modeling.

Acknowledgement:

I would like to express my sincere gratitude to the creators and contributors of the **"House Prices: Advanced Regression Techniques"** creator on Kaggle for providing such a valuable dataset and problem statement. This project has been an enriching opportunity to apply and deepen my knowledge of machine learning, data preprocessing, and regression analysis.