Project Report On

Housing*: Price Prediction*

Submitted By:

Arun Joshva Stephenson

ACKNOWLEDGMENT

I would like to express my sincere thanks of gratitude to my mentors from Data Trained academy and Flip Robo Technologies Bangalore for letting me work on this project. Their suggestions and directions have helped me in the completion of this project successfully. All the required information & the dataset are provided by Flip Robo Technologies.

References:

- https://www.academia.edu/38358601/House Price Prediction
- https://www.researchgate.net/publication/342302491HousingMarket Prediction Problem using Different Machine Learning Algorithms A Case Study
- https://scholarworks.sjsu.edu/cgi/viwecontent.cgi?article=1540&context=etd projects

INTRODUCTION

Thousand of houses are sold every day. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price? Also is it the location? Is it the overall quality of the house? Is it the size? Could it be sold at a good price in future? All these questions come in to our mind when we decide to purchase a house.

In this study, a machine learning model is proposed to predict a house price based on data related to the house (its size, the year it was built in, etc.). During the development and evaluation of our model, we will show the code used for each step followed by its output. This will facilitate the reproducibility of our work.

Business Problem Framing:

Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and

purchases. Predictive modeling, Market mix modeling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.

The project endeavours to extensive data analysis and implementation of different machine learning techniques in python for having the best model with most important features of a house on insight of both business value and realistic perspective.

Business goal: with the help of available independent variables, we need to model the price of the houses. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Conceptual Background of the Domain Problem:

House prices increase every year, so there is a need for a system to predict house prices in the future. House price prediction can help

the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house.

The problem statement is related to the US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia.

The company is looking at prospective properties to buy houses to enter the market. It is required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of house?
- How do these variables describe the price of the house?

In this section, we evaluate widely used regression technologies like Linear Regression, Regularization, Bagging and Boosting and many more ensemble techniques to predict the house sale price result.

Review of Literature:

The relationship between house prices and the economy is an important motivating factor for predicting house prices (Pow, Janulewicz, & Liu, 2014). There is no accurate measure of house prices

(Pow, Janulewicz, & Liu, 2014). Pow states that Real Estate property prices are linked with economy (Pow, Janulewicz, & Liu, 2014). He also states there is no accurate measure of house prices. A property's value is important in real estate transactions. Pow tries to predict the sold and asking prices of real estate values without bias to help both buyers and sellers make their decisions. A property's value is important in real estate transactions.

Housing market is important for economic activities (khamis & kamarudin, 2014). Traditional housing price prediction is based on cost and sale price comparison. So, there is a need for building a model to efficiently predict the house price.

Based on the sample data provided to us from our client database where we have understood that the company is looking at prospective properties to buy houses to enter the market. The data set explains it is a regression problem as we need to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not.

House prices trends are not only the concerns for buyers and sellers, but they also indicate the current economic situations. Therefore, it is important to predict the house prices without bias to help both buyers and sellers make their decisions.

Motivation for the Problem Undertaken:

I have gone through many projects before, but this project has given me an idea to handle large number of attributes. By doing this project I have got an idea about how to deal with data exploration where I have used all my analyzation skills to predict the house price using ML models. The model will be a good way for both buyers and sellers to understand the pricing dynamic of a new market.

The main objectives of this study are as follows:

- ✓ To apply data pre-processing and preparation techniques in order to obtain clean data.
- ✓ To build machine learning models able to predict house price based on house features.
- ✓ To analyse and compare models' performance in order to choose the best model.

By processing the above objects, I will be able to find which variables are important to predict the price of house? And how do these variables describe the price of the house? The relation between house prices and the economy is an important factor for predicting house prices.

ANALYTICAL PROBLEM FRAMING

Mathematical/Analytical Modeling of the Problem:

The house price model is based on a demand function for housing services and a standard life-cycle model of utility for a representative household. This is a common approach in academic research into house prices. The study is to predict the sale price of the house and analyzing which features are important and how they contribute in the prediction.

There are two datasets. One is train dataset which is supervised and another one is test dataset which is unsupervised. The target variable is “SalePrice” and it is a regression type problem. I have used train dataset to build machine learning models and then by using this model I made prediction for the test dataset.

I have observed some columns having more than 85% of zero entries and 70% of null values so, I decided to drop those columns. I have performed both univariate and bivariate analysis to analyze the sale price of the house. I have analyzed the categorical and numerical features using categorical plots and numerical plots respectively to get better insights from the data. In this project I have done various mathematical and statistical analysis such as describing the statistical summary of the columns, feature engineering, treating null values,

removing outliers, skewness, encoding the data etc. Checked for correlation between the features and visualized it using heat map. Also, I built many regression algorithms while building machine learning models, used hyper tuning method for best model and saved the best model. Finally, I predicted the sale price of the house using the saved trained model.

Data Sources and their formats

- ✓ A US-based housing company named Surprise Housing has collected the dataset from the sale of houses in Australia and the data is provided by Flip Robo Company and it is in csv format. There are 2 data sets:
  - Train dataset
  - Test dataset
- ✓ Train dataset will be used for training the machine learning models. The dataset contains 1168 rows and 81 columns, out of 81 columns, 80 are independent variables and remaining 1 is dependent variable (Sale Price).
- ✓ Test dataset contains all the independent variables, but not the target variable. We will apply the trained model to predict the target variable for the test data. The dataset contains 292 rows and 80 columns.
- ✓ The dataset contains both numerical and categorical data. Numerical data contains both nominal and ordinal variables.
- ✓ I can concatenate both train and test data, but this may cause data leakage so I decided to process both the data separately.

Data Pre-processing Data

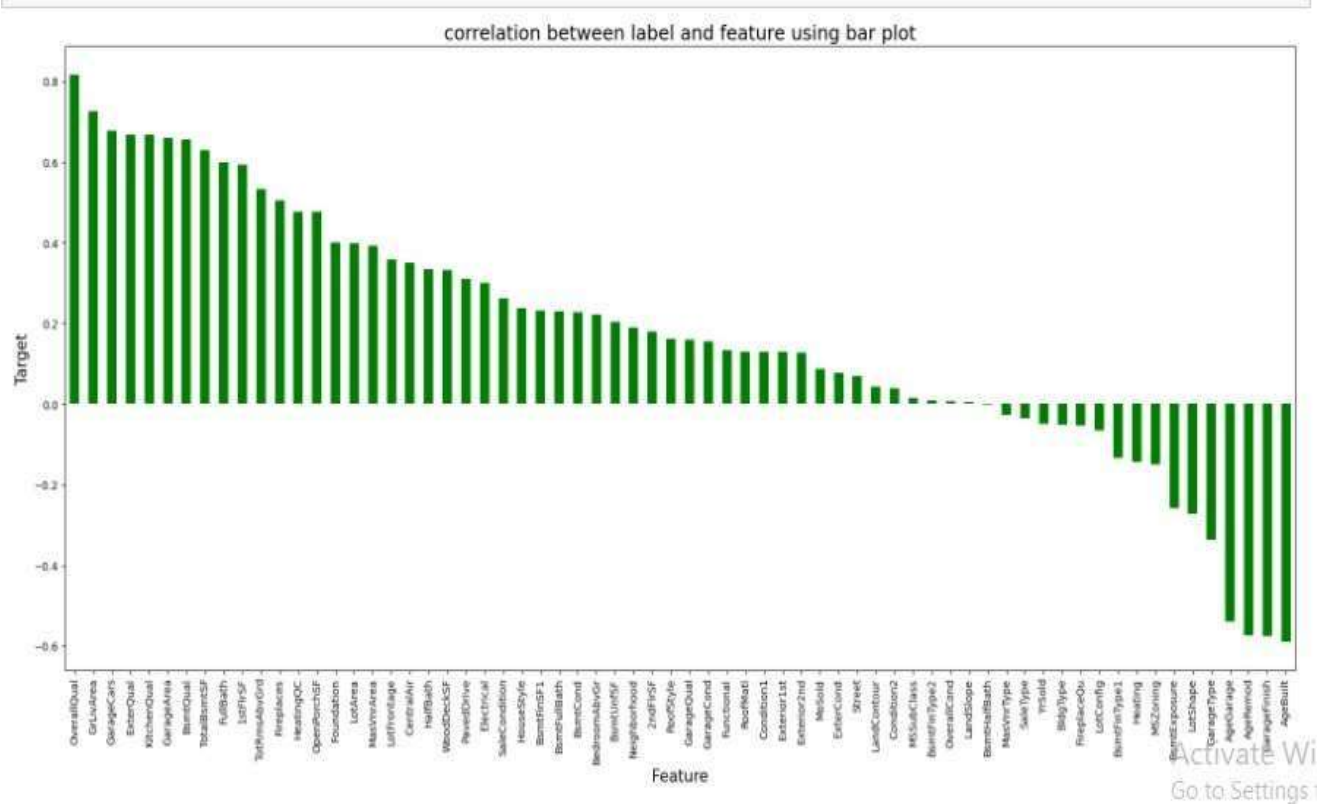
- Firstly, I have imported the necessary libraries and imported both train and test datasets which were in csv format. And process both datasets simultaneously.
- I have done some statistical analysis like checking shape, nunique, column names, data types of the features, info about the features, value counts etc for both train and test data.
- I have dropped the columns “Id” and “Utilities” from both the datasets. Since Id is the unique identifier which contains unique value throughout the data also all the entries in Utilities column were unique. They had no significance impact on the prediction.
- While looking into the value count function I found some of the columns having more than 85% of zero values so, I dropped those columns from both the datasets as they might create skewness which will impact my model.
- Also, I have done some feature extraction as the datasets contained some time variables like YearBuilt, YearRemodAdd, GarageYrBit and YrSold. Converting them into age seem more meaningful as they offer more information about the longevity of the features. It is analogous to the fact that, the statement “Mr. X died at the age of 66 years” holds more information for us than the statement “Mr. X died in the year 2019”. So, I have extracted age information from the datetime variables by taking the difference in year between the year the house was built and year the house was sold and dropped the year columns.

- Used Pearson's correlation coefficient to check the correlation between label and features.
- While checking the correlation I came across multicollinearity problem, I checked VIF values and removed GrLivArea to overcome with the multicollinearity issue.
- Scaled both the datasets using Standard Scalar method and used regression algorithms to build ML models.
- All these steps were performed to both train and test datasets simultaneously.

Data Inputs- Logic- Output Relationships

- To analyse the relation between features and target I have done EDA where I analysed the relation using many plots like bar plot, reg plot, scatter plot, line plot, swarm plot, strip plot, violin plot etc. And found some of the columns like OverallQual, TotalRmsAbvGrd, FullBath, GarageCars etc have strong positive linear relation with the label.

- I have checked the correlation between the target and features using heat map and bar plot. Where I got the positive and negative correlation between the label and features.



From the above bar plot I can notice the positive and negative correlation between the features and label SalePrice. Below are the correlated features.

Important features that affect SalePrice positively and negatively

Features having high Positive correlation with label

- OverallQual
- GrLivArea
- ExterQual
- KitchenQual
- BsmtQual
- GarageCars
- GarageArea

- TotalBsmtSF
- 1stFirSF
- FullBath
- TotRmsAbvGrd

Features having high Negative correlation with label

- Heating
- MSZoning
- LotShape
- BsmtExposure
- GarageType
- AgeRemod
- AgeGarage
- AgeBuilt
- GarageFinish

Hardware & Software Requirements & Tools Used:

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:

- Anaconda 3- language used Python3

Libraries required:

```
# preprocessing
import numpy as np
import pandas as pd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt

import os
import scipy as stats

import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

With the above sufficient library, we can perform pre-processing and data cleaning. For building my ML models Below libraries are required.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn import metrics
```

```
from sklearn.model_selection import GridSearchCV
```

MODEL/S DEVELOPMENT & EVALUATION

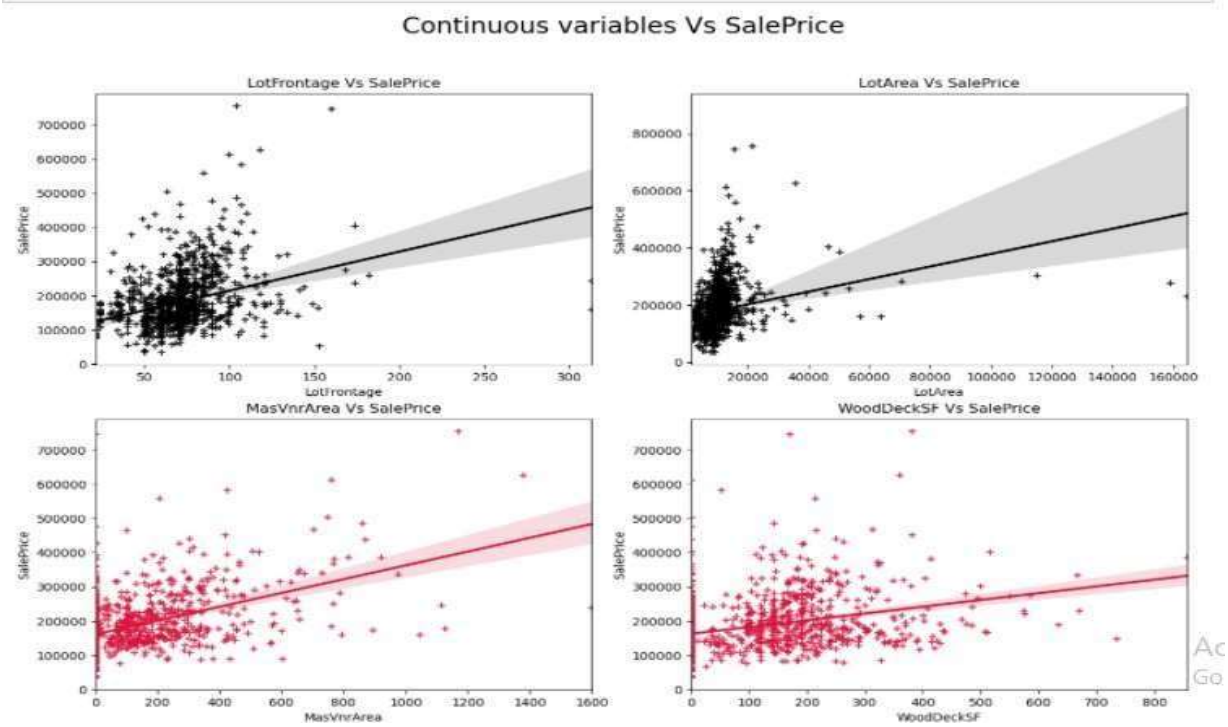
Identification of possible Problem-solving approaches(Methods):

- I have used imputation methods to treat the null values.
- Used percentile method to remove outliers.
- Removed skewness using power transformation (yeo-johnson) method.
- Encoded the object type data into numerical using Ordinal Encoder.
- I have used Pearson's correlation coefficient method to check the correlation between the dependent and independent variables.
- I have scaled the data using Standard Scalar method to overcome with the data biasness.
- Used many Machine Learning models to predict the sale price of the house.

Visualizations:

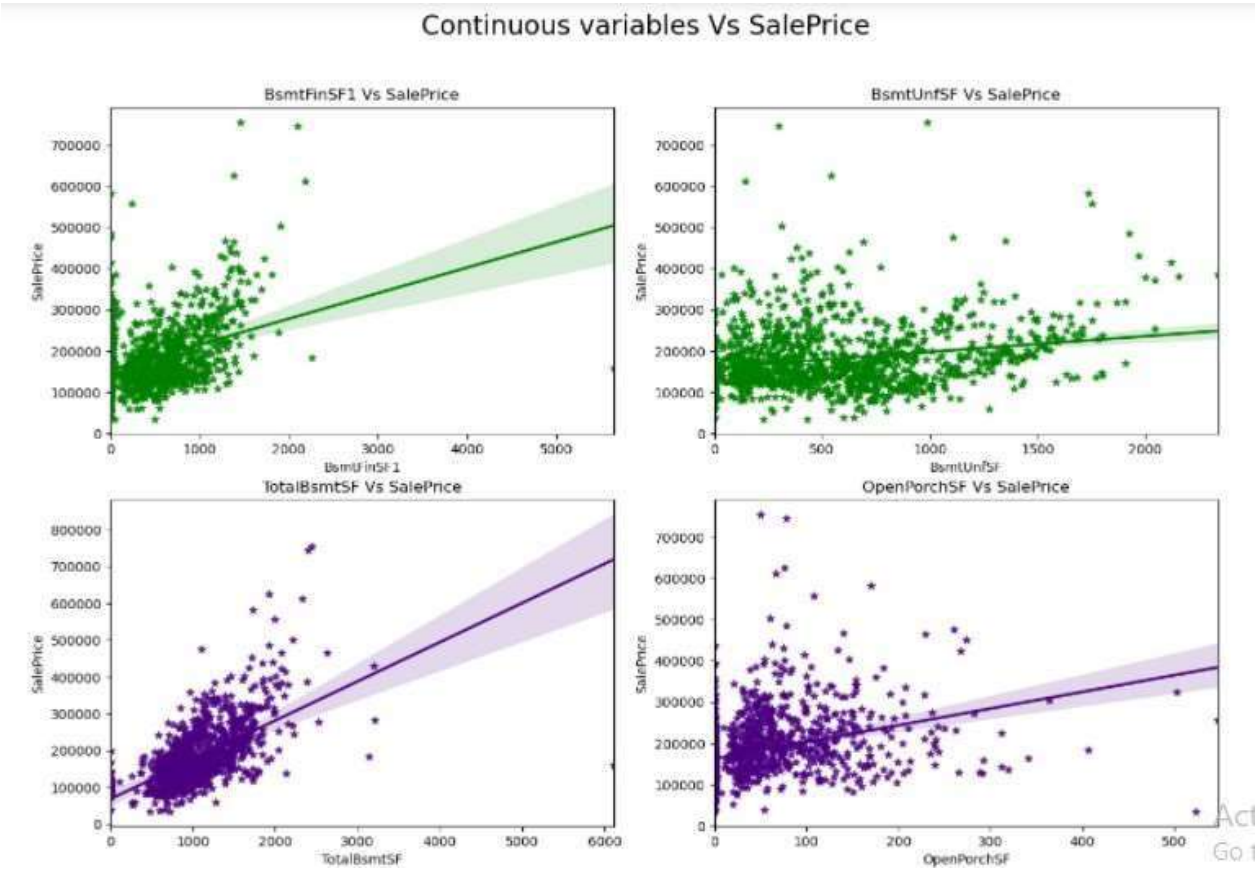
I have analysed the data using both univariate and bivariate analysis to visualize the data. In univariate analysis I have used pie plots, count plots and distribution plot and in bivariate analysis I have used bar plots for categorical columns and used reg plots, scatter plots, strip plots, swarm plots, violin plots and line plot to visualize numerical

columns. These plots have given good pattern. Here I will be showing only bivariate analysis to get the better insights of relation between label and the features.



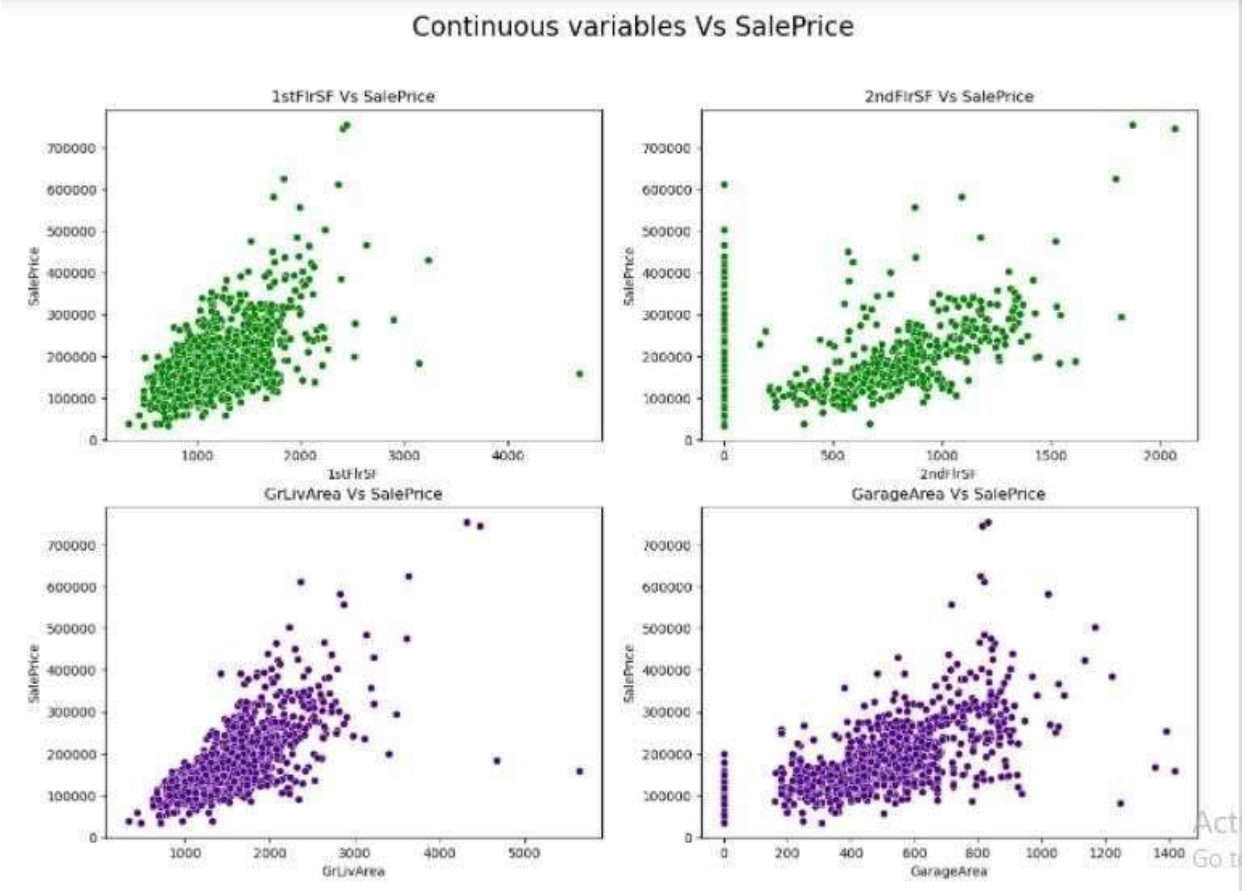
Observations:

- ***SalePrice vs LotFrontage***: From the plot we can observe there is no much linear relation between the label and feature. If the linear feet of street connected to property is more, the sale price is also high.
- ***SalePrice vs LotArea***: There is weakly positive linear relation between the label and feature. But the sale price is high when lot size has around 20000 square feet area. Also as the lot size increases the price is also increasing moderately.
- ***SalePrice vs MasVnrArea***: There is bit positive linear relation between feature and target. Also the sale price is high when Masonry veneer area has around 50-400 square feet. So as the masonry veneer area in square feet increases sale price is also increasing.
- ***SalePrice vs WoodDeckSF***: There is weakly positive linear relation between the feature and target. As the Wood deck area increases, sale price is also increases.



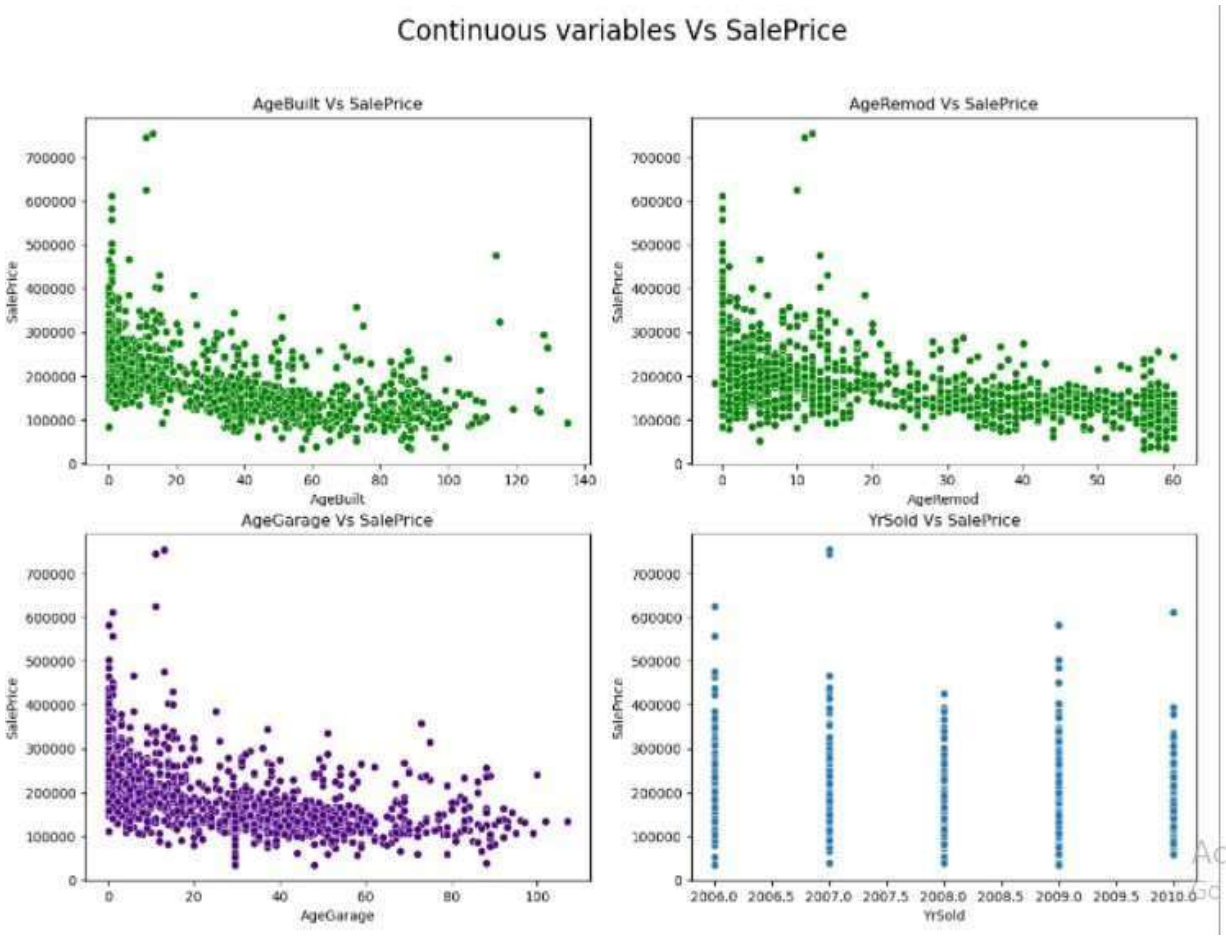
Observations:

- **SalePrice vs BsmtFinSF1**: There is weakly positive linear relation between feature and label. The sale price is high that is 100000-300000 when basement square feet lie upto 1500 square feet. So as the type 1 basement finished square feet increases, sale price is also increases.
- **SalePrice vs BsmtUnfSF:** There is positive linear relation between the target and BsmtUnfSF. When the unfinished basement area is below 1000 square feet, the sale price is high.
- **SalePrice vs TotalBsmtSF**: There is positive linear relation between sale price and TotalBsmtSF. As total basement area increases, sale price also increases.
- **SalePrice vs OpenPorchSF**: There is a linear relation between the label and feature. The sale price is high when Open porch area is below 200sf. Here also as the Open porch area increases, sale price is also increases.



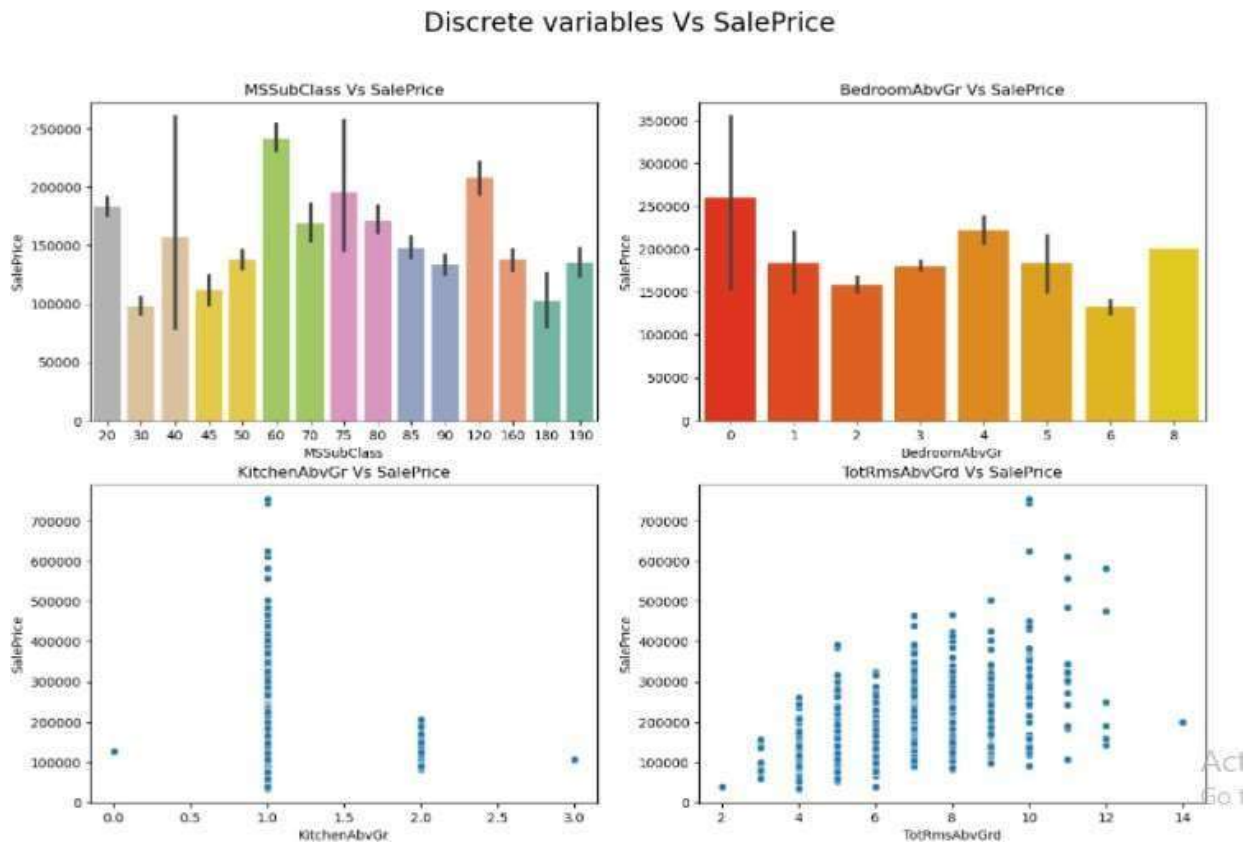
Observation:

- ***SalePrice vs 1stFlrSF:*** There is a linear relation between the label and feature. As we can observein the plot, the sale price is high when the first floor area lies between 500-2000 square feet. So as the 1st floor area increases, sales price also increases moderately.
- ***SalePrice vs 2ndFlrSF:*** There is a positive correlation between SalePrice and 2ndFlrSF. So it is obvious that the sale price increases based on the floors.
- ***SalePrice vs GrLivArea:*** Most of the houses have above grade living area. There is a positive correlation between the label and feature. Here as the above grade living area increases, sale price also increases.
- ***SalePrice vs GarageArea:*** Similar to 2nd floorSF, here also positive linear relation between the label and feature. As size of garage area increases, sale price also increases. The sale price is high when size of garage area is between 200-800 square feet.



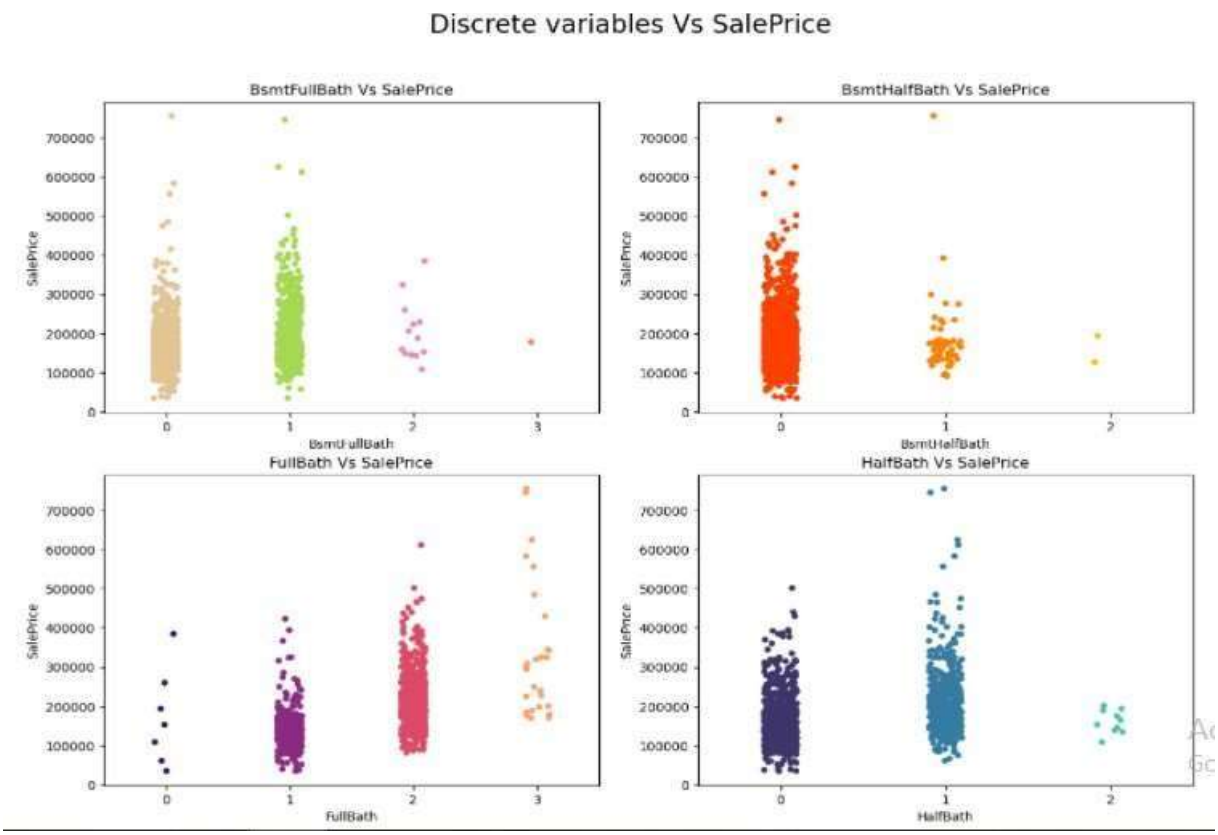
Observations:

- ***SalePrice vs AgeBuilt:*** From the plot I can notice there is negative linear relation between sale price and AgeBuilt. The buildings which have built long back are having less sales price compare to new buildings. Also there are presence of outliers in the data.
- ***SalePrice vs AgeRemod:*** Similar to AgeBuilt, there is a negative linear relation between the label and features. As if Building modification has done long back then the price is less compared to new one. As the age increases, sale price decreases.
- ***SalePrice vs AgeGarage:*** There is negative linear relation and houses which are having recently built garages, they have high sale price. As the age of the garage was built increases, the sale price decreases.
- ***SalePrice vs YrSold:*** Almost all the buildings sold in the recent years and all of them have same sale price. There is no significance difference.



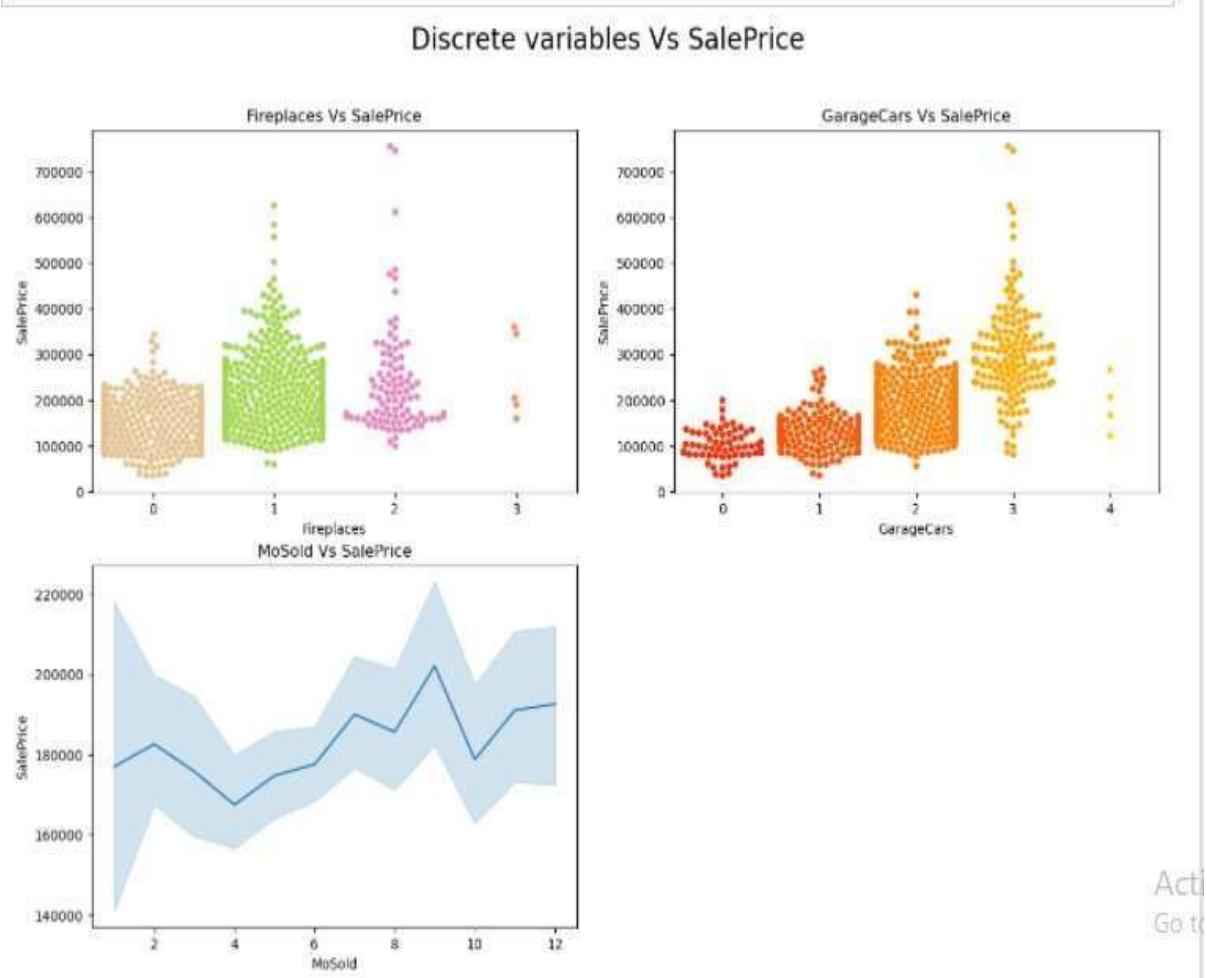
Observations:

- ***SalePrice vs MSSubClass:*** The sale price is high for the MSSubClass 60,120 and 20.
- ***SalePrice vs BedroomAbvGr:*** Many houses are having 0 and 4 bedrooms have high sales price also houses having 8 bedrooms also have high sales price. Other bedroom grades have average sale price.
- ***SalePrice vs KitchenAbvGr:*** Most of the houses have single kitchen and few houses have 2 kitchens. The sale price is also high in case of the houses having single kitchen.
- ***SalePrice vs TotRmsAbvGrd:*** We can observe some linear relation between Total rooms above grade and Sale Prices as the number of rooms increases the sale price also increases.



Observations:

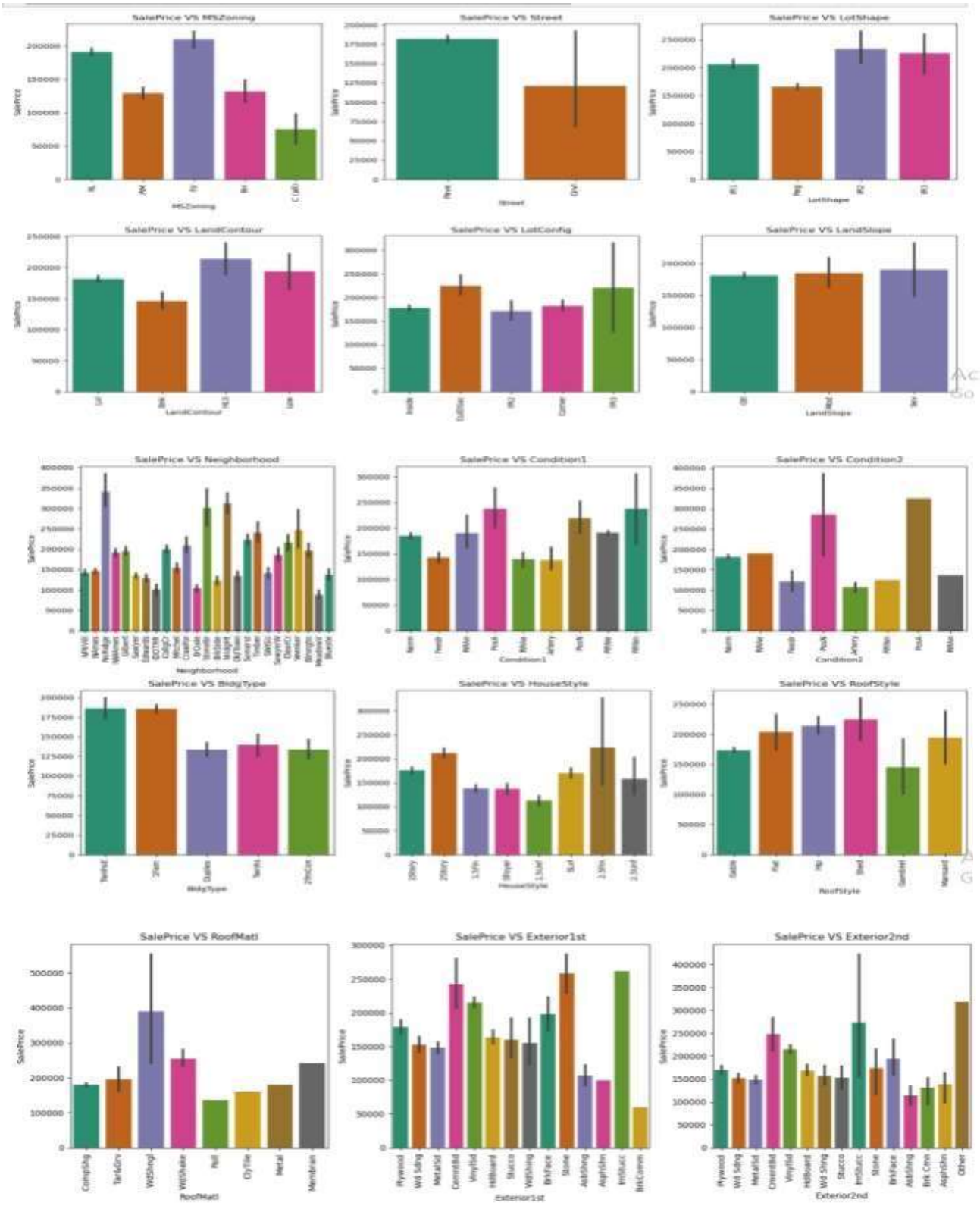
- ***SalesPrice vs BsmtFullBath***: Most of the houses have basement full bathrooms as 0 and 1 which means some of the houses have single basement bathrooms and some of the houses have no basement bathrooms. And sales price is also high in these cases.
- ***SalesPrice vs BsmtHalfBath:*** The houses do not have any single basement bathrooms and those houses have average sale price.
- ***SalesPrice vs FullBath:*** There is positive linear relation between the sale price and full bathrooms above grade. Large number of houses have 1-2 full bathrooms. As the full bathrooms grades increases, sale price is also increasing slightly.
- ***SalesPrice vs HalfBath:*** Some of the houses have no half bathrooms and also somr of the houses have single half bathroom and very few houses have 2 half bathrooms. The houses with 0-1 half bathrooms have average sale price.



Observation:

- ***SalesPrice vs Fireplaces:*** Some houses have no fire places and some houses have 1-2 fire places. The sales price is high for houses having single fireplaces.
- ***SalesPrice vs GarageCars:*** There is positive linear relation between target and feature. As size of garage in car capacity increases, sales price also increases.
- ***SalesPrice vs MoSold:*** Monthly sold have no significance impact on sale price.

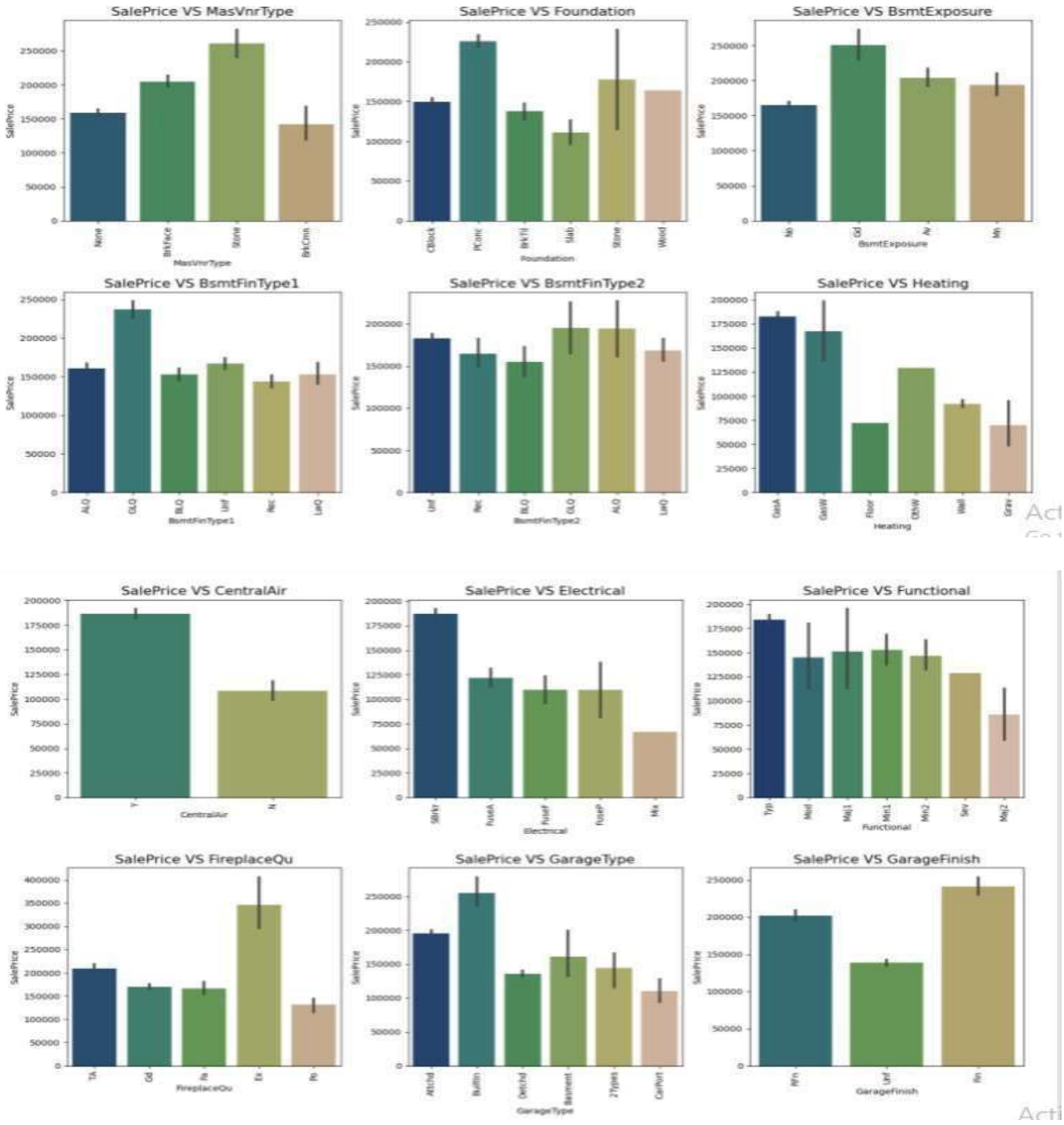
Visualizing Nominal Variables vs SalePrice:



Observation:

- ***SalePrice Vs MSZoning***: Most of the houses are belongs to Floating Village Residential followed by Residential Low Density. The houses from this zone are have high saleprice compared to other zones.
- ***SalePrice vs Street:*** By observing the bar plot, it is obvious that the property of house with Paved type of road have high SalePrice and the houses in gravel roads have very less sale price.
- ***SalePrice vs LotShape***: Most of the houses having moderately irregular and irregular shape of property have high sale price and houses with regullar type of property have less sale price compared to others.
- ***SalePrice vs LandContour:*** The houses having the hiliside and depression property flatness have high sale price compared to others.
- ***SalePrice vs LotConfig***: Most of the houses having moderately irregular and irregular shape of property have high sale price and houses with regular type of property have less sale peice compared to others.
- ***SalePrice vs LandSlope:*** There is no significance difference between the stope of the property. As we can observe the houses having Gentle slope, Moderate Slope and Severe Slope have same sale price.
- ***SalePrice vs Neighborhood***: The houses which are located near Nothridge have high sale price compared to others.
- ***SalePrice vs Condition1:*** The houses having the conditions adjacent to positive off-site teature and houses within 200 of North-South Railroad have high sale price compared to others.
- ***SalePrice vs Condition2***: The houses having the conditions near positive off-site feature park, greenbelt, etc and adjacent to positive off-site feature have high sale price.
- ***SalePrice vs BldgType***: Most of the houses are Single-family Detached and Townhouse End Unit and they have higher sale price compared to other categories.
- ***SalePrice vs HouseStyle:*** Houses which are having styleof dwelling 2nd level finished and Two story have high sale price compared to other types.
- ***SalePrice vs RoofStyle:*** The houses having the roof style Flat, Hip and Shed have high sale price and the houses having gabrei roof style have less sale price.
- ***SalePrice vs RoofMatl***: Houses with Wood Shingles root materials have high sale prices.

- ***SalePrice vs Exterior1st***: Houses having imitation Stucco, Stone and Cement Board as 1st exterior cover have high sale price.
- ***SalePrice vs Exterior2nd:*** Houses having imitation Stucco and other as 2nd cover have high sale price.

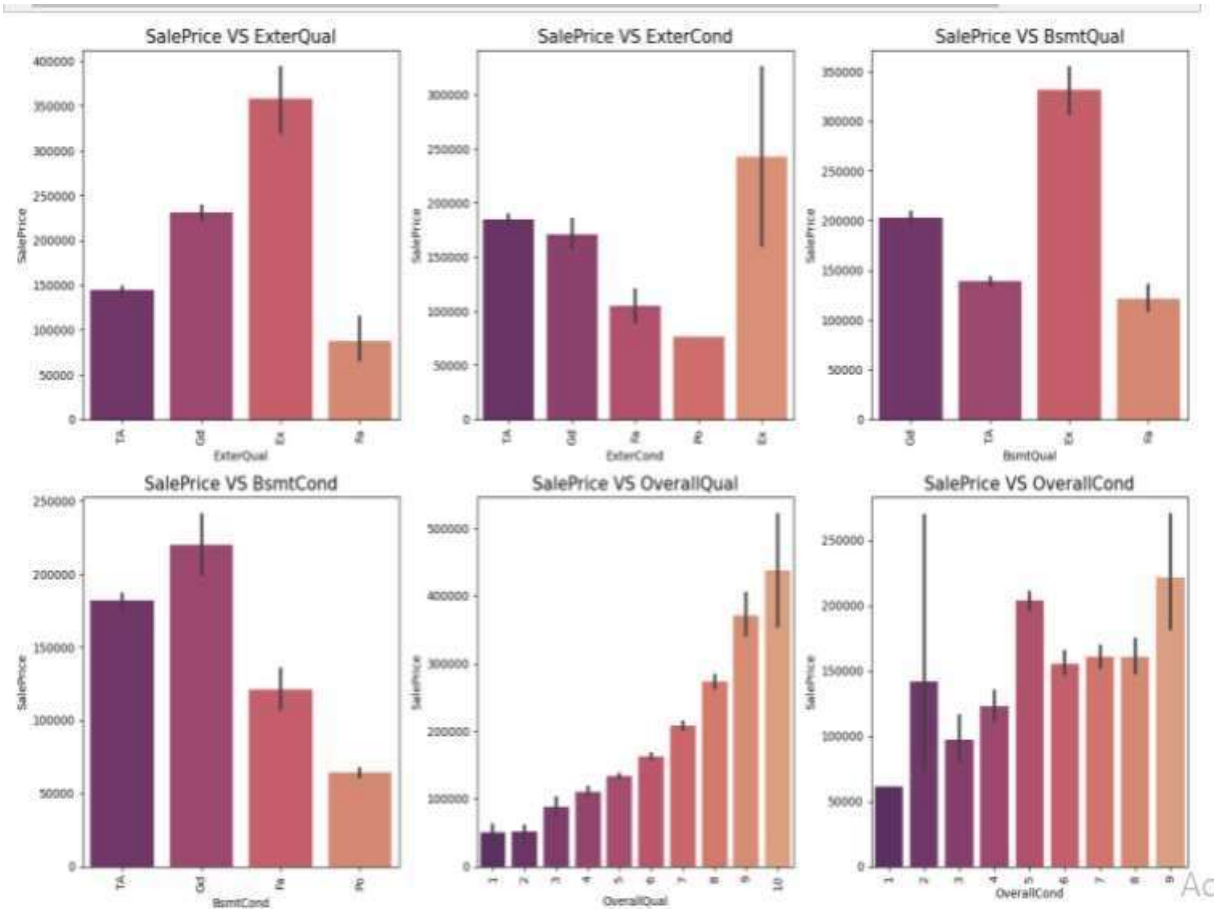


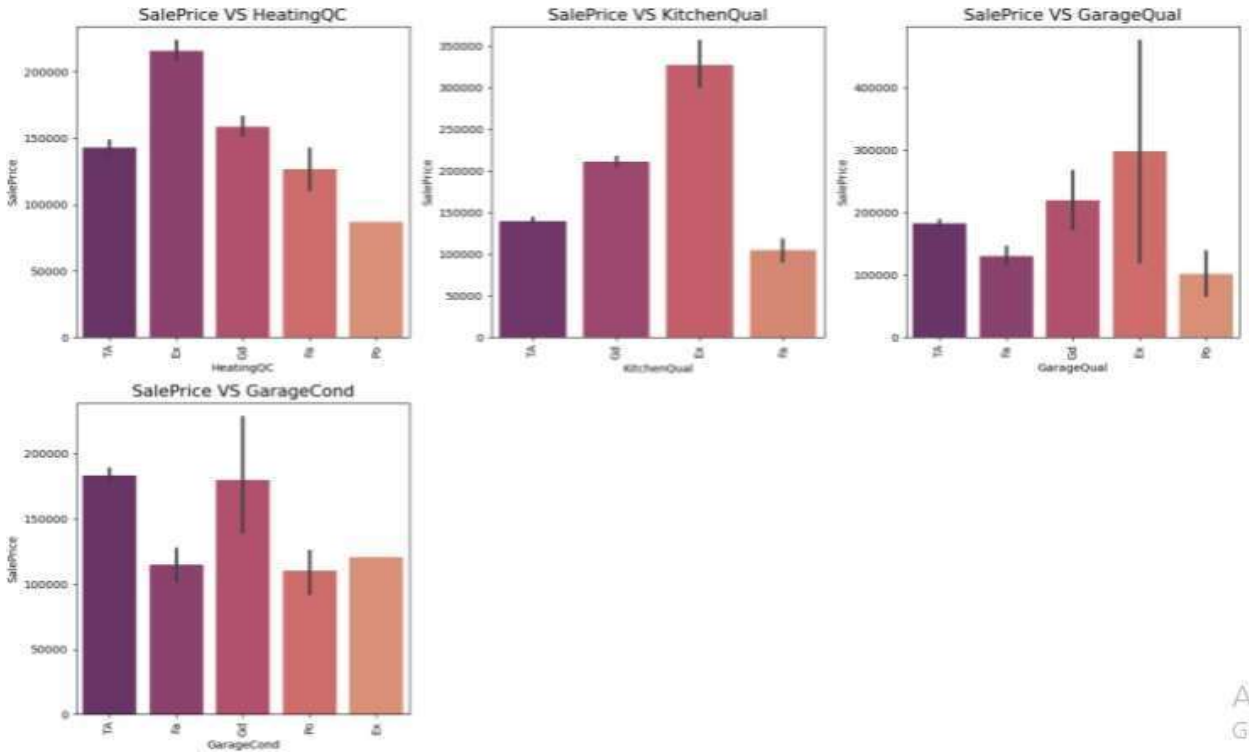
Observation:

- ***SalePrice vs MasVnrType:*** Houses having Stone Masonry veneer type have high sale price than other types.
- ***SalePrice vs Foundation***: Houses having Poured Contrete as foundation type have high sale price compared to other types.
- ***SalePrice vs BsmtExposure:*** Houses having good walkout or garden level walls have high sale price compared to others.
- ***SalePrice vs BsmtFinType1:*** The sale price is high for th ehouses containing good living quarters basement finished area.
- ***SalePrice vs BsmtFinType2***: The sale price is moderately high for the houses having good living quarters and average living quarters.
- ***SalePrice vs Heating:*** The houses having the heating type gas forced warm air furnace and gas hot water or steam heat have high sale price.
- ***SalePrice vs CentralAir***: Most of the houses have central air conditioning so it is ovvious that these houses have high sale price.
- ***SalePrice vs Electrical:*** Most of the houses having standard circuit breakers & romex have high sale price compared to others.
- ***SalePrice vs Functional:*** The houses having the typical functionality have maximum sales price and others have average sale price.
- ***SalePrice vs FireplaceQu:*** The houses having excellent exceptional masonry fireplace quality have high sale price and the houses having poor fireplace quality have very less sale price compared to others.
- ***SalePrice vs GarageType:*** The houses having built-in garage have high sale price compared to others.

- ***SalePrice vs GarageFinish:*** Garages located inside the house which is got finished have high sale price.
- ***SalePrice vs PavedDrive***: Houses having paved drive ways have high sale price.
- ***SalePrice vs SaleType:*** Many houses having sale types as just constructed and sold and Contract 15% Down payment regular terms have high sale price.
- ***SalePrice vs SaleCondition:*** Houses having partial sale condition that is home was not completed when last assessed have high sale price.

Visualizing Ordinal Variables vs SalePrice:





Observation:

- **SalePrice vs ExterQual:** Houses having excellent quality of the material on the exterior have high sale price and houses having fair quality have very less sale price.
- **SalePrice vs ExterCond**: Houses having excellent condition of the material on the exterior have high sale price and the houses having poor condition of the material on the extr=erior have very less sale price compared to others.
- **SalePrice vs BsmtQual**: The houses which evaluates the excellent quality of height of the basement have high sale price compared to others.
- **SalePrice vs BsmtCond**: The houses which evaluates the good quality of general condition of the basement have high sale price compared to others.
- **SalePrice vs OverallQual:** The houses which have very excellent overall quality like material and finish of the house have high sale price. Also we can observe from the plot as the overall quality of the house increases, the sale price also increases. That is there is good linear relation between SalePrice and OverallQual.

- **SalePrice vs OverallCond:** The houses having overall condition as excellent and average have very high sale price compared to others.
- **SalePrice vs HeatingQC**: Most of the houses having excellent heating quality and condition have high sale price.
- **SalePrice vs KitchenQual:** Houses having excellent quality of the kitchen have high sale price compared to others.
- **SalePrice vs GarageQual**: The sale price of the house is high for the houses having excellent garage quality.
- **SalePrice vs GarageCond:** Houses having typical/average garage condition have high sale price and the houses having good garage condition also have high sales price compared to others.

Testing of Identified Approaches(Algorithms):

In this problem SalePrice is my target variable which is continuous in nature, from this I can conclude that it is a regression type problem hence I have used following regression algorithms to predict the sale price of the house. After the pre-processing and data cleaning I left with 67 columns including target and I used these features for prediction.

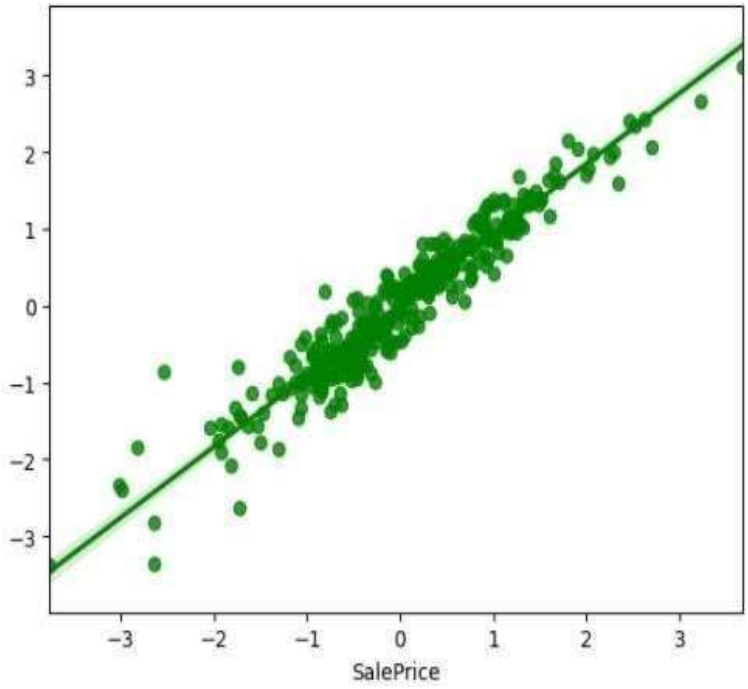
- Linear Regression
- Lasso Regressor
- Ridge Regressor
- Random Forest Regressor
- Extra Trees Regressor
- Gradient Boosting Regressor
- Bagging Regressor

Run and Evaluate Selected Models:

- ***Linear Regression:*** Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables. Linear regression algorithm shows a linear relationship between a dependent (Y) and one or more independent (y) variables. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

```
# checking r2 score for linear regression
LR = LinearRegression()
LR.fit(x_train,y_train)

# prediction
predLR=LR.predict(x_test)
print('R2_score:',r2_score(y_test,predLR))

# mean Absolute error (MAE)
print('MAE:',metrics.mean_absolute_error(y_test, predLR))

# mean Squared error (MSE)
print('MSE:',metrics.mean_squared_error(y_test, predLR))

# root mean Squared error (RMSE)
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predLR)))

# checking cross validation score for Linear Regression
print("Cross Validation Score:",cross_val_score(LR,x,y,cv=5).mean())

# visualizing the predicted values
sns.regplot(y_test,predLR,color="g")
plt.show()
```

```
R2_score: 0.9135263036327237
MAE: 0.22164526960806763
MSE: 0.08934981498358738
RMSE: 0.2989143940722617
Cross Validation Score: 0.865012291420177
```



Created linear regression model and getting 91.35% R2 score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

Regularization Techniques:

Regularization is a form of regression, that constrains/regularizes or shrinks the coefficient estimates towards zero. In other words, it is used to avoid the risk of overfitting. In this project I have used 2 regularization techniques. Lasso and Ridge regressors.

- ***Lasso Regressor:***

  It performs L1 regularization and it is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. This particular type of regression is well-suited for models showing high levels of multicollinearity. It

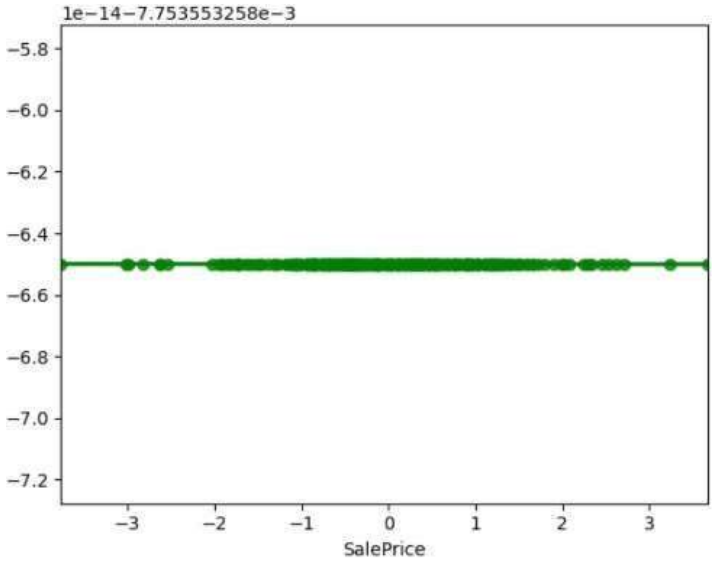
is used when we have greater number of features because it automatically performs feature selection.

```
# checking best alpha value
parameters={'alpha':[1e-15,1e-10, 1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
lasso=Lasso()
clf=GridSearchCV(lasso,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'alpha': 0.01}
```

```
# checking r2score for Lasso Regressor
laso=Lasso(alpha=0.01)

# predictions
lasso.fit(x_train,y_train)
lasso.score(x_train,y_train)
pred_lasso=lasso.predict(x_test)

print('R2_Score:',r2_score(y_test,pred_lasso))
print('MAE:',metrics.mean_absolute_error(y_test, pred_lasso))
print('MSE:',metrics.mean_squared_error(y_test, pred_lasso))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, pred_lasso)))

# checking cv score for lasso regression
print("Cross Validation Score:",cross_val_score(lasso,x,y,cv=5).mean())

# Visualizing the predicted values
sns.regplot(y_test,pred_lasso,color="g")
plt.show()
```

```
R2_Score: -0.0006442632621941335
MAE: 0.7851151818514971
MSE: 1.0339257316713824
RMSE: 1.0168213863168802
Cross Validation Score: -0.01496753017094452
```



Created Lasso regressor model and getting -ve R2 score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

- ***Ridge Regressor:***

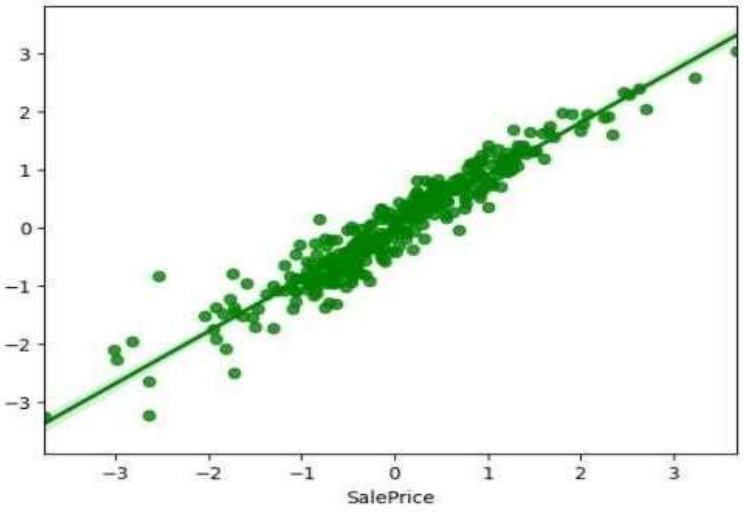
  Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. It is the original cost function of linear regressor we add a regularized term which forces the learning algorithm to fit the data and helps to keep the weights lower as possible.

```
# checking best alpha value
paramsRidge = {'alpha':[1e-15,1e-10, 1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
ridge=Ridge()
clf=GridSearchCV(ridge,paramsRidge)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'alpha': 55}
```

```
# checking r2score for Ridge Regressor
ridge=Ridge(alpha=100)

# predictions
ridge.fit(x_train,y_train)
ridge.score(x_train,y_train)
pred_ridge=ridge.predict(x_test)

print('R2_Score:',r2_score(y_test,pred_ridge))
print('MAE:',metrics.mean_absolute_error(y_test, pred_ridge))
print('MSE:',metrics.mean_squared_error(y_test, pred_ridge))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, pred_ridge)))

# checking cv score for Ridge regression
print("Cross Validation Score:",cross_val_score(ridge,x,y,cv=5).mean())

# Visualizing the predicted values
sns.regplot(y_test,pred_ridge,color="g")
plt.show()
```

```
R2_Score: 0.9130979646314432
MAE: 0.22158072332648257
MSE: 0.08979240055726417
RMSE: 0.29965380117272694
Cross Validation Score: 0.8704185066242779
```



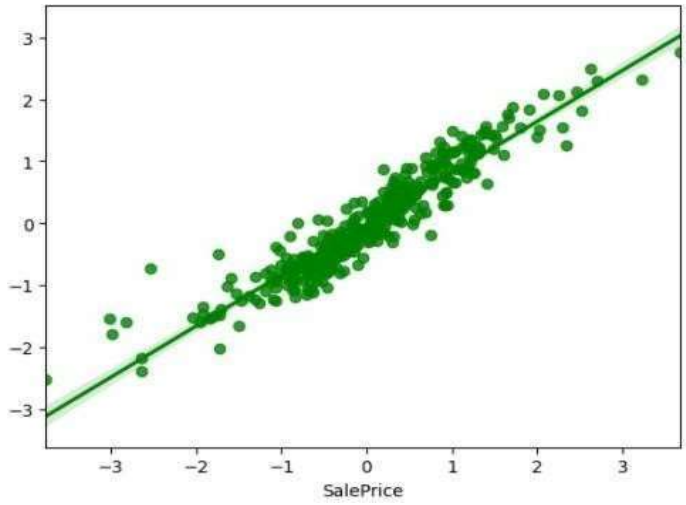
Created Ridge regressor model and getting 91.30% R2 score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

- ***Random Forest Regressor:***

Random forest is an ensemble technique capable of performing both regression and classification tasks with use of multiple decision trees and a technique called Bootstrap Aggregation. It improves the predictive accuracy and control over-fitting.

```
# checking r2score for Random Forest Regressor
RFR=RandomForestRegressor()
RFR.fit(x_train,y_train)

# predictions
predRFR=RFR.predict(x_test)
print('R2_Score:',r2_score(y_test,predRFR))

# metric evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predRFR))
print('MSE:',metrics.mean_squared_error(y_test, predRFR))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predRFR)))

# checking cv score for Random Forest regression
print("Cross Validation Score:",cross_val_score(RFR,x,y,cv=5).mean())

# Visualizing the predicted values
sns.regplot(y_test,predRFR,color="g")
plt.show()
```

```
R2_Score: 0.8860757027414723
MAE: 0.24694013396121609
MSE: 0.1177134239636445
RMSE: 0.34309389963047215
Cross Validation Score: 0.8514681734225704
```



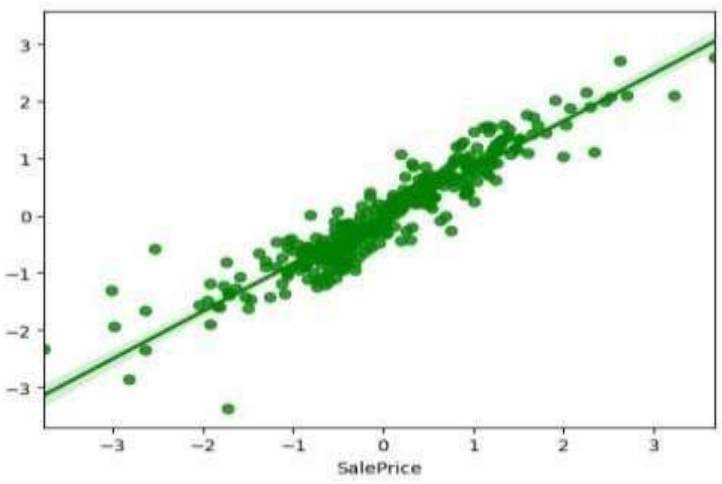
Regressor model and getting 88.60% score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

- ***Extra Trees Regressor:***

The extra trees implements a metaestimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
# checking r2score for Extra Trees Regressor
XT=ExtraTreesRegressor()
XT.fit(x_train,y_train)

# predictions
predXT=XT.predict(x_test)
print('R2_Score:',r2_score(y_test,predXT))

# metric evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predXT))
print('MSE:',metrics.mean_squared_error(y_test, predXT))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predXT)))

# checking cv score for Extra Trees regression
print("Cross Validation Score:",cross_val_score(XT,x,y,cv=5).mean())

# Visualizing the predicted values
sns.regplot(y_test,predXT,color="g")
plt.show()
```

```
R2_Score: 0.8680813252929117
MAE: 0.2543781497831872
MSE: 0.13630629512928777
RMSE: 0.3691968243759523
Cross Validation Score: 0.8505378002977132
```



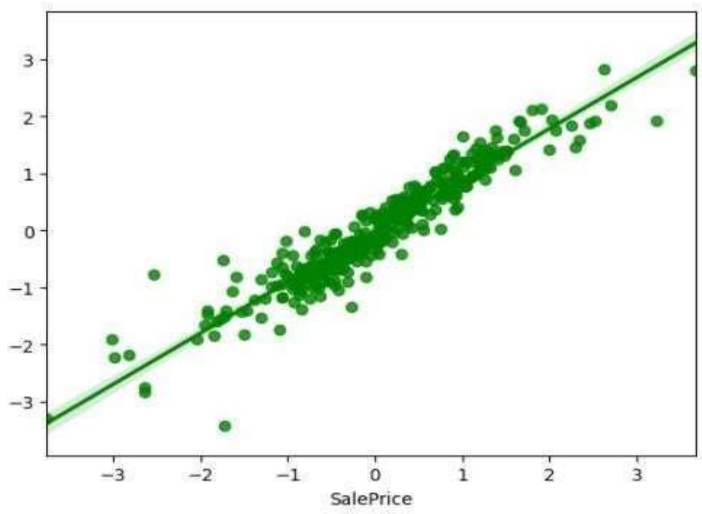
Created ExtraTrees Regressor model and getting 86.80% R2 score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

- ***Gradient Boosting Regressor:***

  Gradient Boosting Regressor is also works for both numerical as well as categorical output variables. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

```
# checking r2score for GradientBoosting Regressor
GB=GradientBoostingRegressor()
GB.fit(x_train,y_train)

# predictions
predGB=GB.predict(x_test)
print('R2_Score:',r2_score(y_test,predGB))

# metric evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predGB))
print('MSE:',metrics.mean_squared_error(y_test, predGB))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predGB)))

# checking cv score for Gradient Boosting regression
print("Cross Validation Score:",cross_val_score(GB,x,y,cv=5).mean())

# Visualizing the predicted values
sns.regplot(y_test,predGB,color="g")
plt.show()
```

```
R2_Score: 0.8946555624188998
MAE: 0.2317646698305891
MSE: 0.1088481978085453
RMSE: 0.3299215024949803
Cross Validation Score: 0.8649771347670235
```



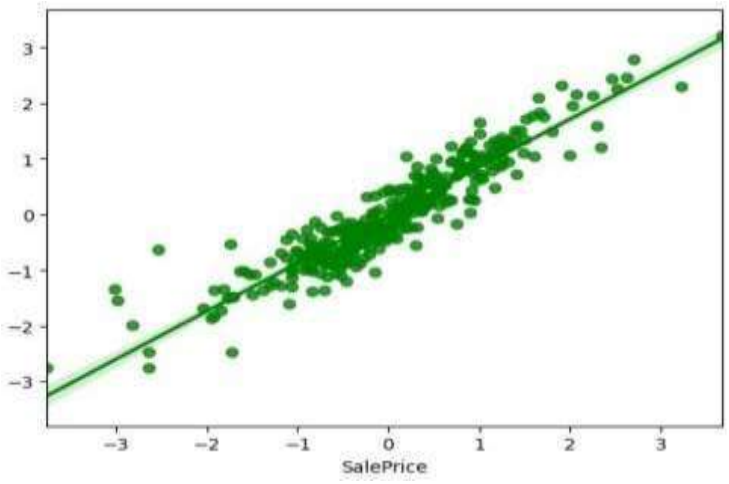
Created GradientBoosting Regressor model and getting 89.46% R2 score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

- ***Bagging Regressor:***

A Bagging regressor is an ensemble metaestimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions to form a final prediction.

```
# checking r2score for Bagging Regressor
BR=BaggingRegressor()
BR.fit(x_train,y_train)

# predictions
predBR=BR.predict(x_test)
print('R2_Score:',r2_score(y_test,predBR))

# metric evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predBR))
print('MSE:',metrics.mean_squared_error(y_test, predBR))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predBR)))

# checking cv score for Bagging regression
print("Cross Validation Score:",cross_val_score(BR,x,y,cv=5).mean())

# Visualizing the predicted values
sns.regplot(y_test,predBR,color="g")
plt.show()
```

```
R2_Score: 0.8694637428672988
MAE: 0.26703932312250445
MSE: 0.1348778982908211
RMSE: 0.3672572644493518
Cross Validation Score: 0.8317019324670822
```



Created Bagging Regressor model and getting 86.94% R2 score using this model. From the above plot we can observe the sales price of the house. The best fit line shows there is strong linear relation between test data of trained model and predicted values.

From the difference between R2 score and Cross validation score I can conclude that Bagging Regressor as my best fitting model as it is giving less difference compare to other models. Let's perform Hyperparameter tuning to increase the model accuracy.

Hyper Parameter Tuning

```
# Let's use the GridSearchCV to find the best parameters in Bagging Regressor
from sklearn.model_selection import GridSearchCV

# Bagging Regressor
parameters = {'n_estimators': [10,50,100,200,500],
              'max_samples':[1.0,5.0,6.0,0.008],
              'max_features':[1.0,10.0,0.0001,5.68],
              'bootstrap':[True,False],
              'oob_score':[True,False],
              'n_jobs':[-1,-2,-3,-4]}
```

```
GCV = GridSearchCV(BaggingRegressor(),parameters,cv=5)
```

Running GridSearchCV for the model Bagging Regressor.

```
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=BaggingRegressor(),
             param_grid={'bootstrap': [True, False],
                         'max_features': [1.0, 10.0, 0.0001, 5.68],
                         'max_samples': [1.0, 5.0, 6.0, 0.008],
                         'n_estimators': [10, 50, 100, 200, 500],
                         'n_jobs': [-1, -2, -3, -4],
                         'oob_score': [True, False]})
```

```
# getting best parameters
GCV.best_params_
```

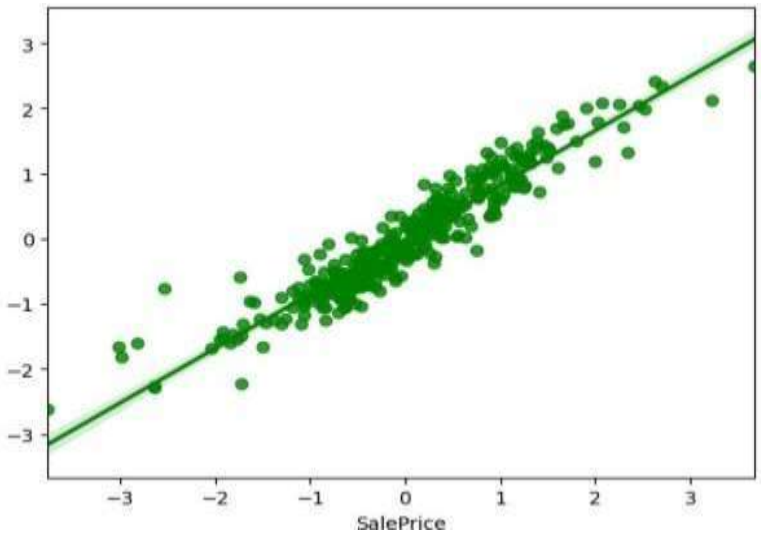
```
{'bootstrap': True,
 'max_features': 1.0,
 'max_samples': 1.0,
 'n_estimators': 200,
 'n_jobs': -4,
 'oob_score': False}
```

I have used GridSearchCV to get the best parameters of Bagging Regressor. And used all the obtained parameters to get the accuracy of final model.

Creating final model:

```
# creating final model
Final_Model = BaggingRegressor(bootstrap=True , max_features=1.0 , max_samples=1.0 , n_estimators=200 , n_jobs=-4 , oob_score=Fa

# prediction
Final_Model.fit(x_train, y_train)
pred = Final_Model.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred)*100)

# metric evaluation
print('Mean absolute error: ',metrics.mean_absolute_error(y_test, pred))
print('Mean squared error: ',metrics.mean_squared_error(y_test, pred))
print('Root Mean Squared error:',np.sqrt(metrics.mean_squared_error(y_test, pred)))

# visualizing the predicted values
sns.regplot(y_test,pred,color="g")
plt.show()
```

```
R2_Score:  88.95340936754856
Mean absolute error:  0.24500950665139257
Mean squared error:  0.11414000680818295
Root Mean Squared error: 0.33784612889329213
```

The R2 score of Bagging Regressor has been increased % after tuning the model. It is giving R2 score as % which is very good. The plot gives some strong linear between test and predicted values.

Saving the final model and predicting the saved model

- I have saved my final best model using joblib library in .pkl format.

```
# saving the model using .pkl
import joblib
joblib.dump(Final_Model,"Housing_SalePrice_Prediction.pkl")
```

```
['Housing_SalePrice_Prediction.pkl']
```

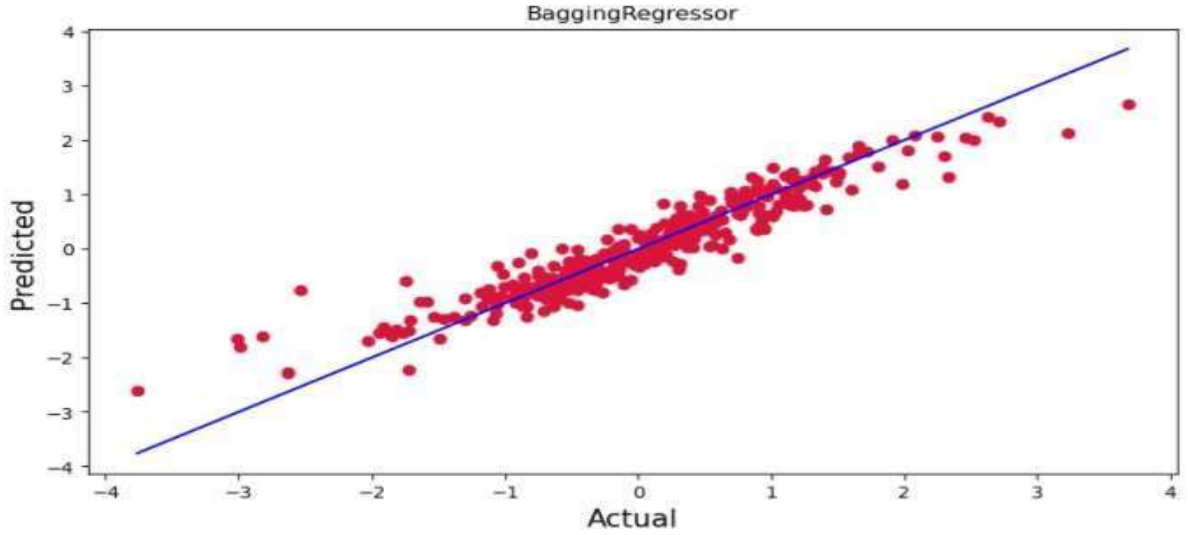
- Loading saved model and predicting the sale price

```
# Loading the saved model
Model=joblib.load("Housing_SalePrice_Prediction.pkl")

# prediction
prediction = Model.predict(x_test)
prediction
```

```
array([-0.62715656, -0.74735437, -1.65419152,  0.17331808, -0.38785913,
        0.36739113, -0.84084215, -0.79764927, -0.93656716, -0.14209635,
       -0.57046769, -0.19252582, -0.23014205,  1.40136646, -0.76948284,
       -0.30964257,  0.00968298,  0.04108274, -0.80194618,  0.3314038 ,
       -1.30564872, -1.60567967,  0.34432445, -0.35146932, -0.27444782,
        0.79397974,  0.16818197, -0.55271554,  1.08716147,  0.54057613,
       -0.10264455, -0.02121226, -0.22890164, -0.62634825,  1.78220055,
        0.35668621,  1.4374627 , -0.43707301,  0.05640856, -0.26711982,
       -0.65447724, -0.02574074,  1.64042725, -0.58876891,  0.52699475,
       -0.43765949,  1.50542644,  1.69474725,  0.18531012, -1.49330256,
        0.33084974,  2.09436062, -0.74955522, -0.1458354 , -0.19964081,
       -0.53604242,  1.25441063, -0.79678309, -0.08672407,  2.12845598,
       -0.88718812,  0.09227802, -0.17621189, -0.84701693, -0.64308654,
       -0.89912155, -0.8272114 ,  1.26335886,  0.47632155, -1.03980529,
        2.04121199, -0.55355078, -0.13225865,  0.84475632,  0.27056324,
        1.30730368, -0.69557546, -0.60736066,  1.26399375, -1.25679206,
        0.42809221,  0.55950653,  1.06751378,  0.12686528, -0.79603463,
       -1.31907347, -0.17074931, -0.69633051, -0.58531421, -0.01632785,
       -0.55060803,  0.20211494, -0.96157077,  0.69103308, -0.24332383,
       -0.6913375 ,  0.28632904, -0.02103002, -0.43373395, -0.86794432,
       -1.55291355, -0.98560926, -0.36167448,  1.20274756,  1.33183194,
       -0.37109695,  1.80354298,  1.16938496, -0.39560517, -0.79381953,
        1.2452307 , -0.70138833, -0.99654632, -0.6561017 , -0.56945367,
       -0.09556829, -0.2867795 , -0.79352626, -1.8103595 , -1.54245313,
        1.19522586,  0.01896936,  0.95516051,  0.69377572, -0.19791683,
       -0.46590264, -0.61358127,  0.34031702,  0.78533097,  0.21884839,
        0.46250103, -0.00427841,  0.76629069, -0.51633883,  1.4875825 ,
       -1.31512588, -0.1800339 ,  0.15072435,  0.00847516, -0.16874924,
        0.99740832, -0.76368467, -0.7669477 , -0.30807351, -2.22534009,
        0.50724332,  0.17574331, -0.66241633, -0.75299921, -0.63374905,
        0.33109834,  0.97329933, -1.06416468,  0.4513675 , -0.24925259,
       -0.51747904, -0.86606367,  1.99441782,  1.0979664 , -0.04394183,
       -0.98623012, -0.93549995,  0.89510009,  1.39410916, -0.49396106,
        0.3007666 , -0.89457862, -0.934952  , -0.7301473 , -0.51558911,
       -0.37530332, -0.89460848,  0.82735855, -0.53026183, -0.36400709,
       -1.52100185,  0.37508622, -1.18084765, -0.72522976,  0.46068614,
```

```
-0.40784998, -0.68812066, -1.05301597,  0.93805859, -0.21240776,
 0.93343933,  0.42193473, -2.26320674,  0.26436864,  0.96507767,
 0.25030495,  1.32991535,  0.35197053,  0.11895984,  0.22635672,
-0.70584717, -0.15463913,  1.21017762, -0.15282186,  1.21214219,
-0.71048195,  0.3317502 ,  0.35771143, -0.35442967, -0.47626736,
 1.02050713,  1.0765286 ,  0.65214607,  1.12229167,  1.49812098,
-0.23905155, -2.61680629, -0.48247745,  0.78142239,  0.01519552,
 0.69841314, -0.90670874, -0.22015194, -0.42914249,  0.9609283 ,
-0.48202654,  0.66774967, -0.35409356, -0.41769715, -0.64286104,
-0.27485225,  0.23243039,  1.71627326,  0.06548659, -0.33066923,
-0.58480665, -0.1427296 ,  0.28707035,  0.93441935,  0.01861281,
 2.06185674,  0.60729246, -0.41106349, -1.23424027,  1.24620237,
 1.05691141, -0.82366045, -0.97989948,  0.56602899,  0.65076112,
 0.54048845, -0.06047782,  2.42785709,  2.0088771 ,  0.10122052,
-0.28456441, -0.53420988,  0.59300727, -0.84265943,  1.16295388,
-0.50860668, -0.22137085, -0.51748206,  0.78204998, -1.04880759,
 1.1208815 , -1.68669007, -0.26737298,  0.5457568 ,  0.27195023,
 0.7876065 , -0.92903437,  0.71767205, -0.31883846, -1.607482  ,
-1.25438598,  0.39596102,  0.16061445,  0.93611202,  0.0626388 ,
-1.05039228,  0.41248536,  1.05109395, -1.23867383,  0.78583056,
 1.43337247, -1.1327701 ,  1.34496314, -0.75014995,  0.80642602,
 1.76054339, -0.91386615,  1.88638148,  0.62939387,  0.43696509,
 1.19317808,  1.41007705, -0.76478273, -0.69446701, -0.55414642,
-0.23831707, -0.51368973, -0.64898222, -0.39731386,  0.0593391 ,
-0.51728508,  0.92157975,  0.62994054,  0.45335292,  1.27303883,
 2.34717209,  2.66290904,  0.59994225,  1.36990751,  0.09916279,
 0.52054192,  0.81473074, -1.28746061,  1.10064633, -0.91928327,
-0.69955959, -0.73610505, -1.47530726,  0.45519314, -1.02775327,
-0.71483678,  0.356534  , -0.67579652, -1.43581175,  0.20240396,
-0.20262715, -0.71990339,  0.62374102,  0.68068087,  0.50782171,
-0.62898175,  0.09334048,  0.41714518, -0.22744493,  0.81302122,
-0.095649  ,  0.93016123,  0.46131459,  0.53125169,  0.1920465 ,
-1.05492985, -0.72772007, -0.67476345, -0.51932892, -1.65491323,
-2.29304829,  0.10234443,  1.11052134, -0.0782178 , -0.32604811,
-0.66996386])
```

```
plt.figure(figsize=(10,5))
plt.scatter(y_test, prediction, c='crimson')
p1 = max(max(prediction), max(y_test))
p2 = min(min(prediction), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('Actual', fontsize=15)
plt.ylabel('Predicted', fontsize=15)
plt.title("BaggingRegressor")
plt.show()
```



- The plot gives the linear relation between predicted and actual sale price of the house. The best fitting line gives the actual values and red dots gives the predicted values.

Predicting SalePrice of house for test dataset using saved trained model

```
# predicting the house sale price from the features of the testing data
Predicted_SalePrice = Model.predict(x_test)
Predicted_SalePrice
```

```
array([-0.62715656, -0.74735437, -1.65419152,  0.17331808, -0.38785913,
        0.36739113, -0.84084215, -0.79764927, -0.93656716, -0.14209635,
       -0.57046769, -0.19252582, -0.23014205,  1.40136646, -0.76948284,
       -0.30964257,  0.00968298,  0.04108274, -0.80194618,  0.3314038 ,
       -1.30564872, -1.60567967,  0.34432445, -0.35146932, -0.27444782,
        0.79397974,  0.16818197, -0.55271554,  1.08716147,  0.54057613,
       -0.10264455, -0.02121226, -0.22890164, -0.62634825,  1.78220055,
        0.35668621,  1.4374627 , -0.43707301,  0.05640856, -0.26711982,
       -0.65447724, -0.02574074,  1.64042725, -0.58876891,  0.52699475,
       -0.43765949,  1.50542644,  1.69474725,  0.18531012, -1.49330256,
        0.33084974,  2.09436062, -0.74955522, -0.1458354 , -0.19964081,
       -0.53604242,  1.25441063, -0.79678309, -0.08672407,  2.12845598,
       -0.88718812,  0.09227802, -0.17621189, -0.84701693, -0.64308654,
       -0.89912155, -0.8272114 ,  1.26335886,  0.47632155, -1.03980529,
        2.04121199, -0.55355078, -0.13225865,  0.84475632,  0.27056324,
        1.30730368, -0.69557546, -0.60736066,  1.26399375, -1.25679206,
        0.42809221,  0.55950653,  1.06751378,  0.12686528, -0.79603463,
       -1.31907347, -0.17074931, -0.69633051, -0.58531421, -0.01632785,
       -0.55060803,  0.20211494, -0.96157077,  0.69103308, -0.24332383,
       -0.6913375 ,  0.28632904, -0.02103002, -0.43373395, -0.86794432,
       -1.55291355, -0.98560926, -0.36167448,  1.20274756,  1.33183194,
       -0.37109695,  1.80354298,  1.16938496, -0.39560517, -0.79381953,
        1.2452307 , -0.70138833, -0.99654632, -0.6561017 , -0.56945367,
       -0.09556829, -0.2867795 , -0.79352626, -1.8103595 , -1.54245313,
        1.19522586,  0.01896936,  0.95516051,  0.69377572, -0.19791683,
       -0.46590264, -0.61358127,  0.34031702,  0.78533097,  0.21884839,
        0.46250103, -0.00427841,  0.76629069, -0.51633883,  1.4875825 ,
       -1.31512588, -0.1800339 ,  0.15072435,  0.00847516, -0.16874924,
        0.99740832, -0.76368467, -0.7669477 , -0.30807351, -2.22534009,
        0.50724332,  0.17574331, -0.66241633, -0.75299921, -0.63374905,
        0.33109834,  0.97329933, -1.06416468,  0.4513675 , -0.24925259,
       -0.51747904, -0.86606367,  1.99441782,  1.0979664 , -0.04394183,
       -0.98623012, -0.93549995,  0.89510009,  1.39410916, -0.49396106,
        0.3007666 , -0.89457862, -0.934952  , -0.7301473 , -0.51558911,
       -0.37530332, -0.89460848,  0.82735855, -0.53026183, -0.36400709,
```

```
-1.52100185,  0.37508622, -1.18084765, -0.72522976,  0.46068614,
-0.40784998, -0.68812066, -1.05301597,  0.93805859, -0.21240776,
 0.93343933,  0.42193473, -2.26320674,  0.26436864,  0.96507767,
 0.25030495,  1.32991535,  0.35197053,  0.11895984,  0.22635672,
-0.70584717, -0.15463913,  1.21017762, -0.15282186,  1.21214219,
-0.71048195,  0.3317502 ,  0.35771143, -0.35442967, -0.47626736,
 1.02050713,  1.0765286 ,  0.65214607,  1.12229167,  1.49812098,
-0.23905155, -2.61680629, -0.48247745,  0.78142239,  0.01519552,
 0.69841314, -0.90670874, -0.22015194, -0.42914249,  0.9609283 ,
-0.48202654,  0.66774967, -0.35409356, -0.41769715, -0.64286104,
-0.27485225,  0.23243039,  1.71627326,  0.06548659, -0.33066923,
-0.58480665, -0.1427296 ,  0.28707035,  0.93441935,  0.01861281,
 2.06185674,  0.60729246, -0.41106349, -1.23424027,  1.24620237,
 1.05691141, -0.82366045, -0.97989948,  0.56602899,  0.65076112,
 0.54048845, -0.06047782,  2.42785709,  2.0088771 ,  0.10122052,
-0.28456441, -0.53420988,  0.59300727, -0.84265943,  1.16295388,
-0.50860668, -0.22137085, -0.51748206,  0.78204998, -1.04880759,
 1.1208815 , -1.68669007, -0.26737298,  0.5457568 ,  0.27195023,
 0.7876065 , -0.92903437,  0.71767205, -0.31883846, -1.607482  ,
-1.25438598,  0.39596102,  0.16061445,  0.93611202,  0.0626388 ,
-1.05039228,  0.41248536,  1.05109395, -1.23867383,  0.78583056,
 1.43337247, -1.1327701 ,  1.34496314, -0.75014995,  0.80642602,
 1.76054339, -0.91386615,  1.88638148,  0.62939387,  0.43696509,
 1.19317808,  1.41007705, -0.76478273, -0.69446701, -0.55414642,
-0.23831707, -0.51368973, -0.64898222, -0.39731386,  0.0593391 ,
-0.51728508,  0.92157975,  0.62994054,  0.45335292,  1.27303883,
 2.34717209,  2.66290904,  0.59994225,  1.36990751,  0.09916279,
 0.52054192,  0.81473074, -1.28746061,  1.10064633, -0.91928327,
-0.69955959, -0.73610505, -1.47530726,  0.45519314, -1.02775327,
-0.71483678,  0.356534  , -0.67579652, -1.43581175,  0.20240396,
-0.20262715, -0.71990339,  0.62374102,  0.68068087,  0.50782171,
-0.62898175,  0.09334048,  0.41714518, -0.22744493,  0.81302122,
-0.095649  ,  0.93016123,  0.46131459,  0.53125169,  0.1920465 ,
-1.05492985, -0.72772007, -0.67476345, -0.51932892, -1.65491323,
-2.29304829,  0.10234443,  1.11052134, -0.0782178 , -0.32604811,
-0.66996386])
```

- These are the predicted sale price of the house for test data set.

- I have created data frame for the predicted result which is as below.

Creating DataFrame and Saving the Predictions

```
# creating dataframe for the predicted results
Prediction = pd.DataFrame()
Prediction['SalePrice']= Predicted_SalePrice
Prediction
```

	SalePrice
0	-0.627157
1	-0.747354
2	-1.654192
3	0.173318
4	-0.387859
...	...
346	0.102344
347	1.110521
348	-0.078218
349	-0.326048
350	-0.669964

351 rows × 1 columns

```
# saving the predictions
Prediction.to_csv("Predicted_House_SalePrice_Data.csv", index=False)
```

I have predicted the SalePrice for test dataset using saved model of train dataset and getting good predictions. I have saved my predictions in csv format for further analysis.

Key Metrics for success in solving problem under consideration

Interpretation of the results:

Visualizations: I have used distribution plot to visualize the target variable SalePrice, which was almost normally distributed. From the scatter plot we noticed most of the features like OverallQual, TotalRmsAbvGrd , FullBath, GarageCars etc had some strong linear relation with target as we observed as the quality or area increased, the sale price also tends to increase.

The heat map and bar plot helped to understand the correlation between target and features. Also, with the help of heat map I found multicollinearity problem and I have done feature selection to overcome with the issue. Detected outliers and skewness using box plots and distribution plots. And I found some of the features skewed to right. I got to know the count of each column using count plots and pie plots.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed many processing steps which I have already mentioned in the pre-processing step.

Modelling: After cleaning and processing both train and test data, I performed train test split to build the model. I have built multiple regression models to get the accurate R2 score, and evaluation metrics. I got Bagging regressor as best model which gives 87% R2 score. This is due to over-fitting , so I checked the cross-validation score. After tuning the best model Bagging regressor I got 88% R2 score and even got minimum MAE, MSE and RMSE values. Less error means no over fitting.

And finally, I saved my final model and got the good predictions results for test dataset.

CONCLUSION

Key Findings and Conclusions of the Study

In this study, we have used multiple machine learning models to predict the house sale price. We have gone through the data analysis by performing feature engineering, finding the relation between features and label through visualizations. And got the important feature and we used these features to predict the price by building ML models. We have got good prediction results. After using hyper parameter tuning, the best model increased and the R2 score was 88% also the errors decreased which means no over-fitting issue.

Findings: Which variables are important to predict the price of variable?

- ✓ Overall Quality is the most contributing and highest positive impacting feature for prediction. Also, the features like GarageArea, LotArea, 1stFlrSF, TotalBsmtSF etc have somewhat linear relation with the price variable.

How do these variables describe the price of the house?

- ✓ The houses which have very excellent overall quality like material and finish of the house have high sale price. Also we have observed from the plot that as the overall quality of the house increases, the sale price also increases. That is there is good linear relation between SalePrice and OverallQual. So, if the seller builds the house according to these types of qualities that will increase the sale price of the house.
- ✓ There is a linear relation between the SalePrice and 1stFlrSF. As we have seen as the 1st floor area increases, sales price also increases moderately. So, people like to live in the houses which have only 1-2 floors and the cost of the house also increases in this case.
- ✓ Also, we have seen the positive linear relation between the SalePrice and GarageArea. As size of garage area increases, sale price also increases.
- ✓ There is positive linear relation between sale price and TotalBsmtSF. As total basement area increases, sale price also increases.

Learning Outcomes of the Study in respect of Data Science:

While working on this project I learned more things about the housing market and how the machine learning models have helped to predict the price of house which indeed helps the sellers and buyers to understand the future price of the house. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This graphical representation helped me to understand which features are important and how these features describe the sale price. Data cleaning was one of the important and crucial things in this project where I replaced all the null values with imputation methods and dealt with features having zero values and time variables.

Finally, our aim is achieved by predicting the house price for the test data, I hope this will be further helps for sellers and buyers to understand the house marketing. The machine learning models and data analytic techniques will have an important role to play in this type of problems. It helps the customers to know the future price of the houses.

Limitations of this work and scope for future work

Limitations:

- ✓ In case of processing train and test dataset, I felt concatenation is not suitable as it causes data leakage. The dataset contains some irrelevant columns, zero values, null values, so it is need to increase the dataset size by filling these values.

- ✓ The dataset has many limitations, the main limitation is that we have no information potential buyers and environment of the sale. The factors such as auctions can have an influence on the price of the house.

- ✓ The dataset does not capture many economic factors. Collecting more accurate and important details about the houses from the buyers will help to analyse the data more clearly.

Future work:

- ✓ One of the major future scopes is adding estate database of more cities which will provide the user to explore more estates and reach an accurate decision.

- ✓ As a recommendation, I advise to use this model by the people who want to buy a house in the area covered by the dataset to have an idea about the actual price. The model can be used also with datasets that cover different cities and areas provided that they contain the same features. I also suggest that people take into consideration the features that were deemed as most important as seen in this study might help them estimate the house price better

Thank You

****