Project Report

On

Surprise Housing Price Prediction

Submitted To:

Ms. Khushboo Garg

(SME-FlipRobo)

Submitted By:

Pooja Jain(Int-34)

Acknowledgments

To Almighty, Flip Robo Team ,Data Trained Team and Family.. Thanks for making me push my limits and helping me in learning new skills.

(Pooja Jain)

INTRODUCTION

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.

- Our problem is related to one such housing company named ***Surprise Housing***.
- We are required to model the price of houses with the available independent variables.
- 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

A 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 data is provided in the CSV file. The company is looking at prospective properties to buy houses to enter the market.

You are 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:

1. Which variables are important to predict the price of a variable?
2. How do these variables describe the price of the house?

Review of literature

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.

Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the house.

Motivation for the Problem Undertaken

The objective of doing this project is to build a model to predict the house prices with the help of other supporting features. We are going to predict the price by using various Machine Learning algorithms. The sample data is provided to us from our client database. In order to improve the selection of customers, the client wants some predictions that could help them in further investment and

improvement in selection of customers. House Price Index is commonly used to estimate the changes in housing price. Since housing price is strongly correlated to other factors such as location, area, population, it requires other information apart from HPI to predict individual housing price.

There have been a considerably large number of papers adopting traditional machine learning approaches to predict housing prices accurately.

To explore various impacts of features on prediction methods, we will apply both traditional and advanced machine learning approaches to investigate the difference among several advanced models and also comprehensively validate multiple techniques in model implementation on regression and provide an optimistic result for housing price prediction.

Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of data is 1168 rows and 81columns.

There are 2 data sets that are given. One is training data and one is testing data.

1) Train file will be used for training the model, i.e., the model will learn from this file. It contains all the independent variables and the target variable. Size of training set: 1168 records.

2) Test file contains all the independent variables, but not the target variable. We will apply the model to predict the target variable for the test data. Size of test set: 292 records.

Data Preprocessing & EDA

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models.

- ✓ Loading the training dataset as dataframe
- ✓ Use pandas to set display.
- ✓ Check the number of rows and columns present in our training and testing dataset.
- ✓ Check for missing data and the number of rows with null values
- ✓ Verified the percentage of missing data in each column and decided to discard that have more than 50% of null values and dependency of 80% and above on single variable.
- ✓ Dropped all the unwanted columns and duplicate data present in our dataframe.
- ✓ Checked the unique values information in each column to get a gist for categorical data
- ✓ Performed imputation to fill missing data using mean on numeric data and mode for categorical data columns
- ✓ Used Pandas Profiling during the visualization phase along with pie plot, count plot, scatter plot and the others
- ✓ With the help of ordinal encoding technique converted all object datatype columns to numeric datatype
- ✓ Thoroughly checked for outliers and skew ness information
- ✓ With the help of heatmap, correlation bar graph was able to understand the Feature vs Label relativity and insights on multicollinearity among feature and target.
- ✓ Separate feature and label data to ensure feature scaling is performed avoiding any kind of biasness

- ✓ Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details
- ✓ Finally created a regression model function along with evaluation metrics to pass through various models.

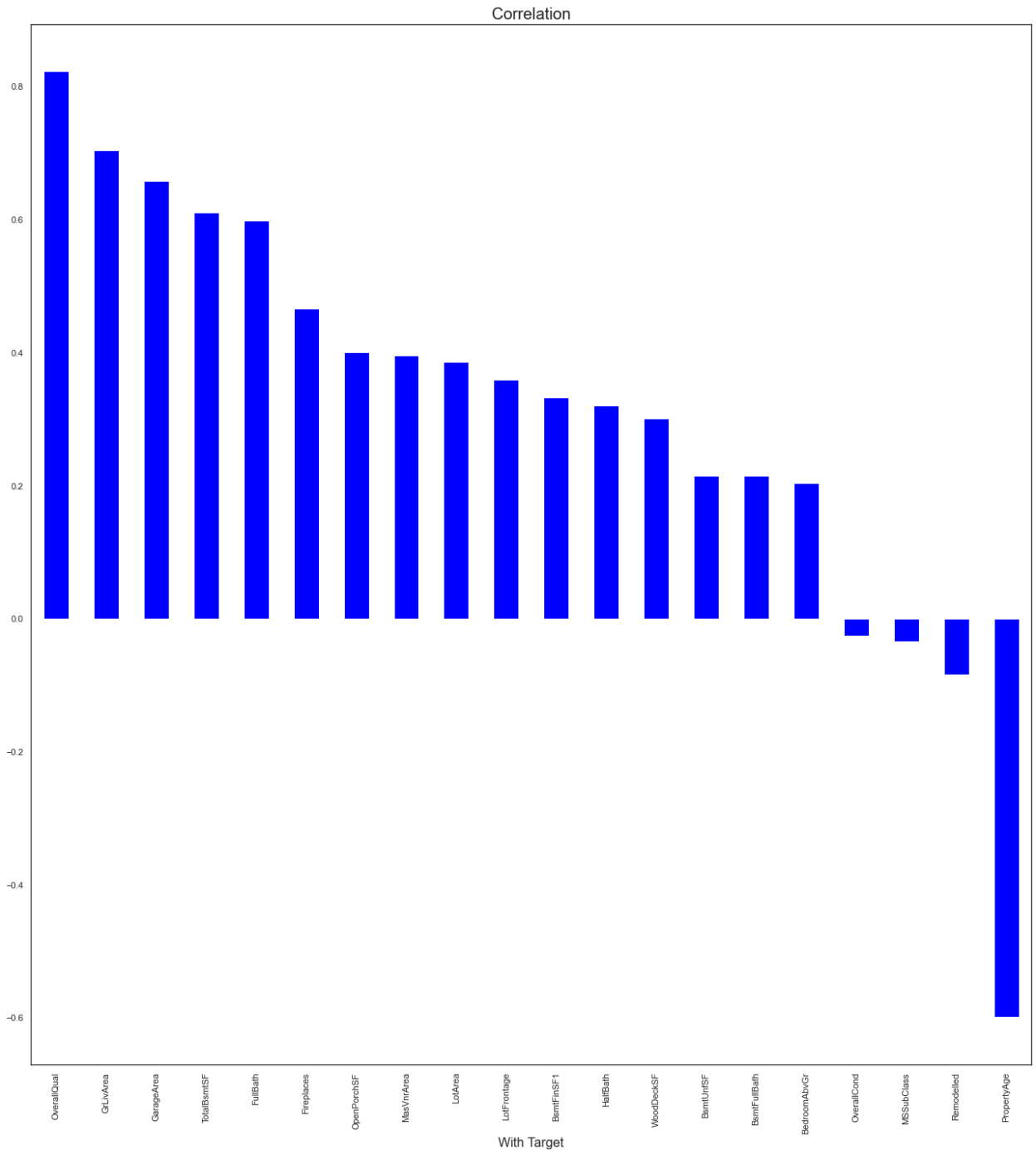
Data Inputs- Logic- Output Relationships

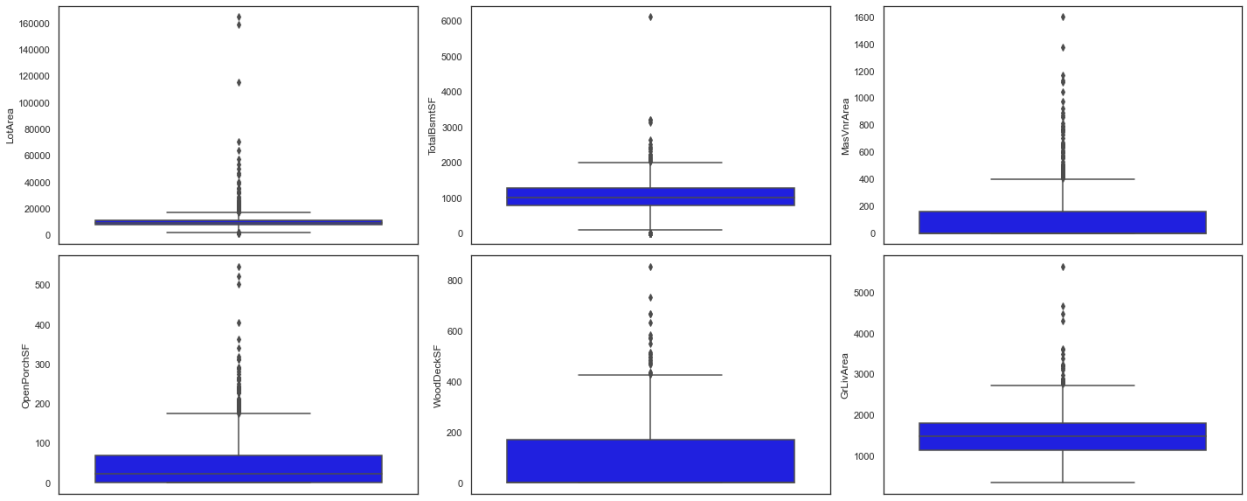
We had to go through various data preprocessing steps to understand what was given to us and what we were expected to predict for the project. When it comes to logical part the domain expertise of understanding how real estate works and how we are supposed to cater to the customers came in handy to train the model with the modified input data. In Data Science community there is a saying “Garbage In Garbage Out” therefore we had to be very cautious and spent almost 80% of our project building time in understanding each and every aspect of the data how they were related to each other as well as our target label.

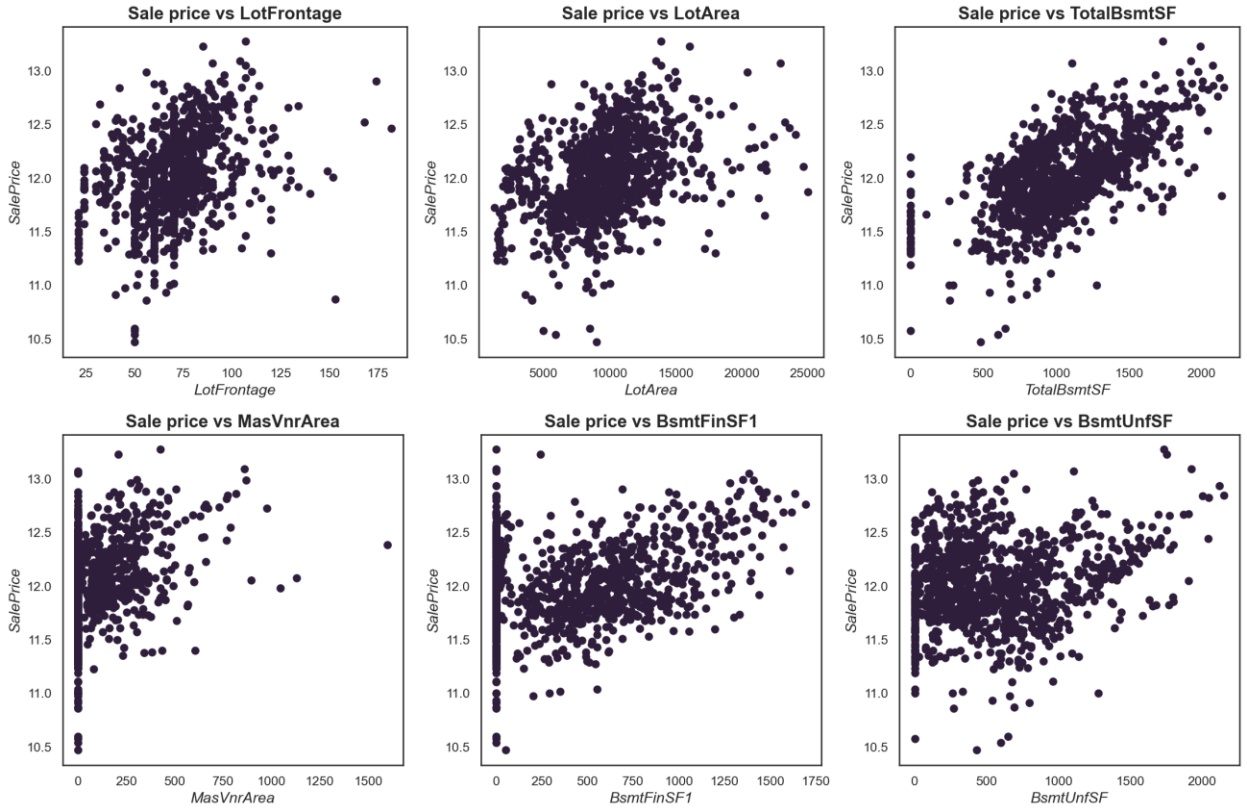
With the objective of predicting hosing sale prices accurately we had to make sure that a model was built that understood the customer priorities trending in the market and imposing those norms to generate relevant results.

Below are some heatmaps and scatterplots to understand the relationship between major variables.

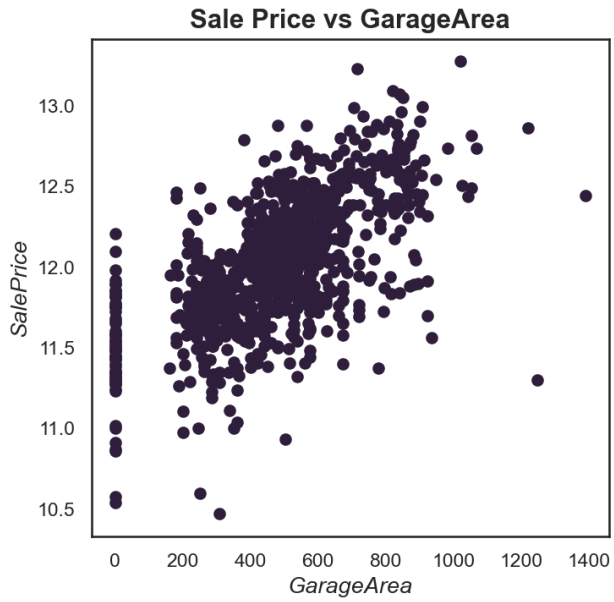
Also we have removed skewness from target to get better reults. We have done this using log transformation.

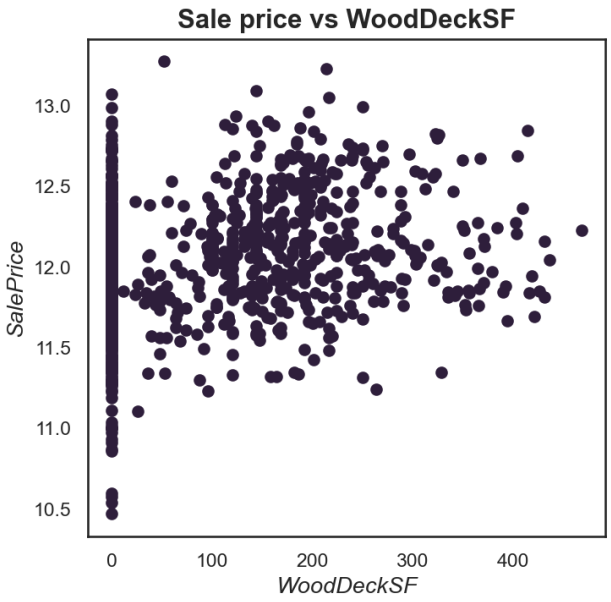

Correlation
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
OverallQual
GrLivArea
GarageArea
TotalBsmtSF
FullBath
Fireplaces
OpenPorchSF
MasVnrArea
LotArea
LotFrontage
BsmtFinSF1
HalfBath
WoodDeckSF
BsmtUnfSF
BsmtFullBath
BedroomAbvGr
OverallCond
MSSubClass
Remodelled
PropertyAge
With Target

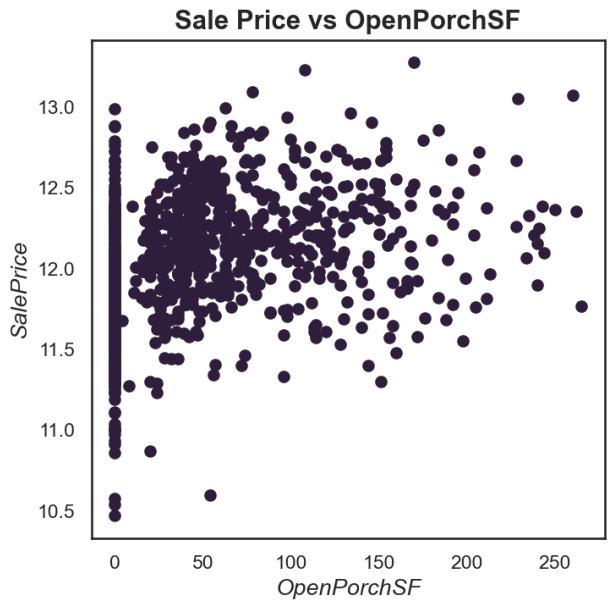

LotArea
TotalBsmtSF
MasVnrArea
OpenPorchSF
WoodDeckSF
GrLivArea


Sale price vs LotFrontage
Sale price vs LotArea
Sale price vs TotalBsmtSF
SalePrice
LotFrontage
LotArea
TotalBsmtSF

Sale price vs MasVnrArea
Sale price vs BsmtFinSF1
Sale price vs BsmtUnfSF
SalePrice
MasVnrArea
BsmtFinSF1
BsmtUnfSF


Sale Price vs GarageArea
SalePrice
GarageArea
13.0
12.5
12.0
11.5
11.0
10.5
0
200
400
600
800
1000
1200
1400

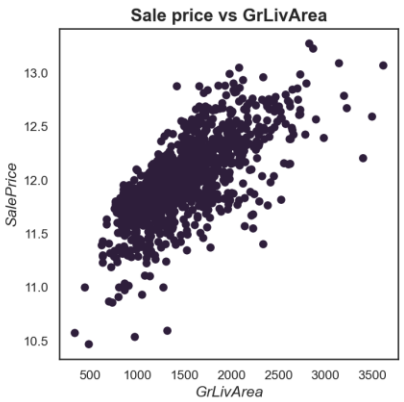

Sale price vs WoodDeckSF
SalePrice
WoodDeckSF
13.0
12.5
12.0
11.5
11.0
10.5
0
100
200
300
400

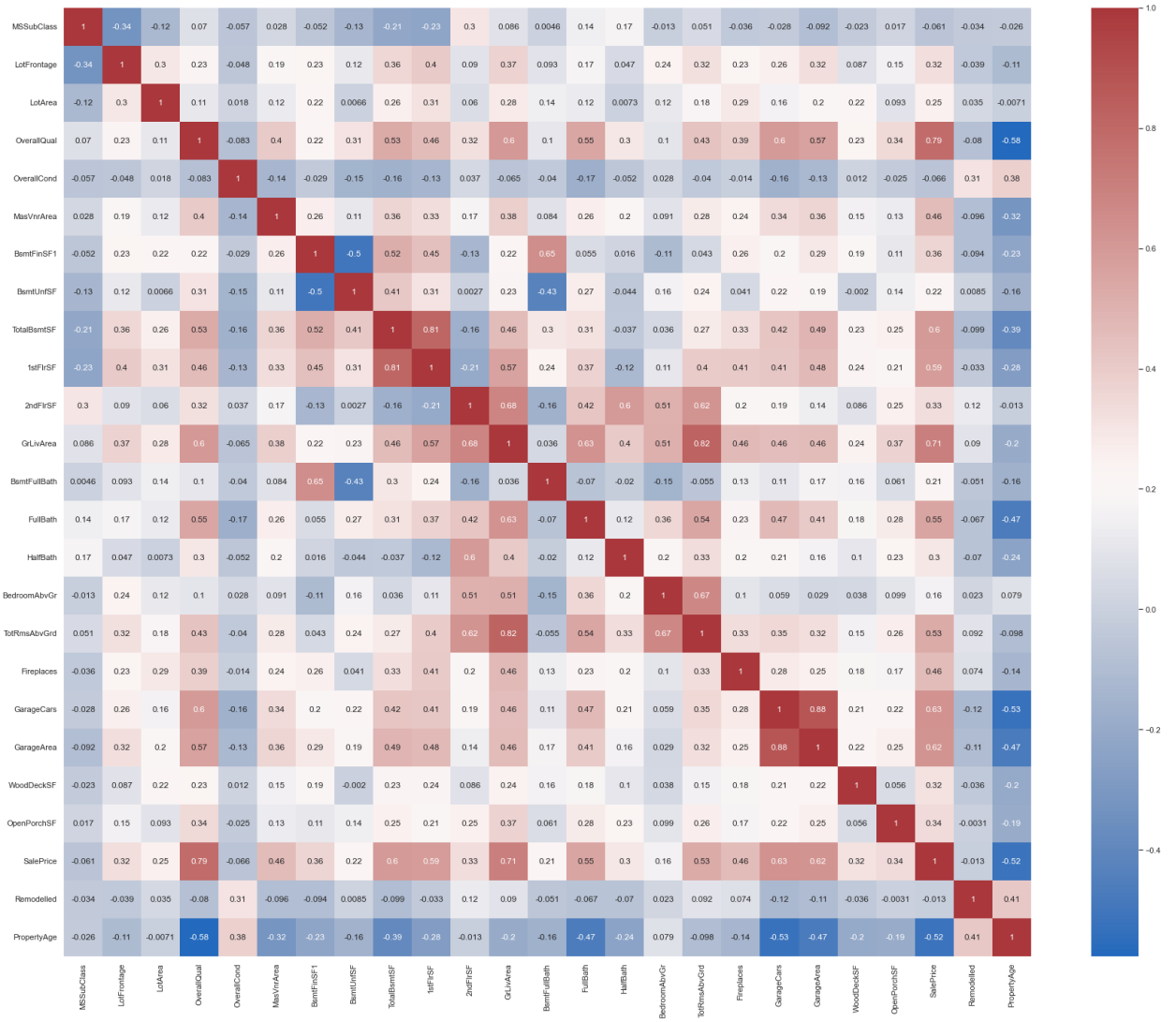

Sale Price vs OpenPorchSF
SalePrice
OpenPorchSF
13.0
12.5
12.0
11.5
11.0
10.5
0
50
100
150
200
250

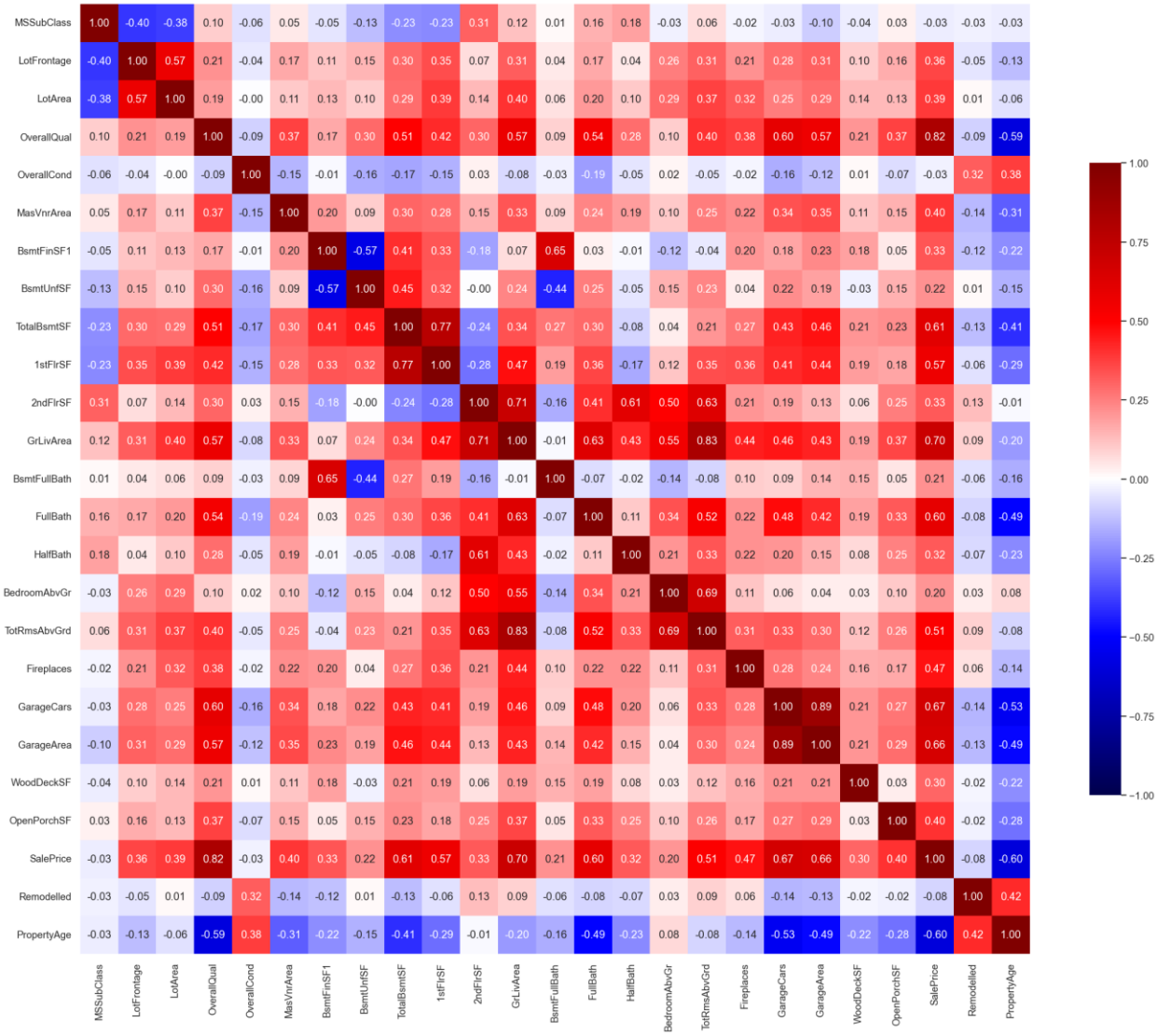

Sale price vs PropertyAge
SalePrice
PropertyAge
13.0
12.5
12.0
11.5
11.0
10.5
0
20
40
60
80
100
120
140

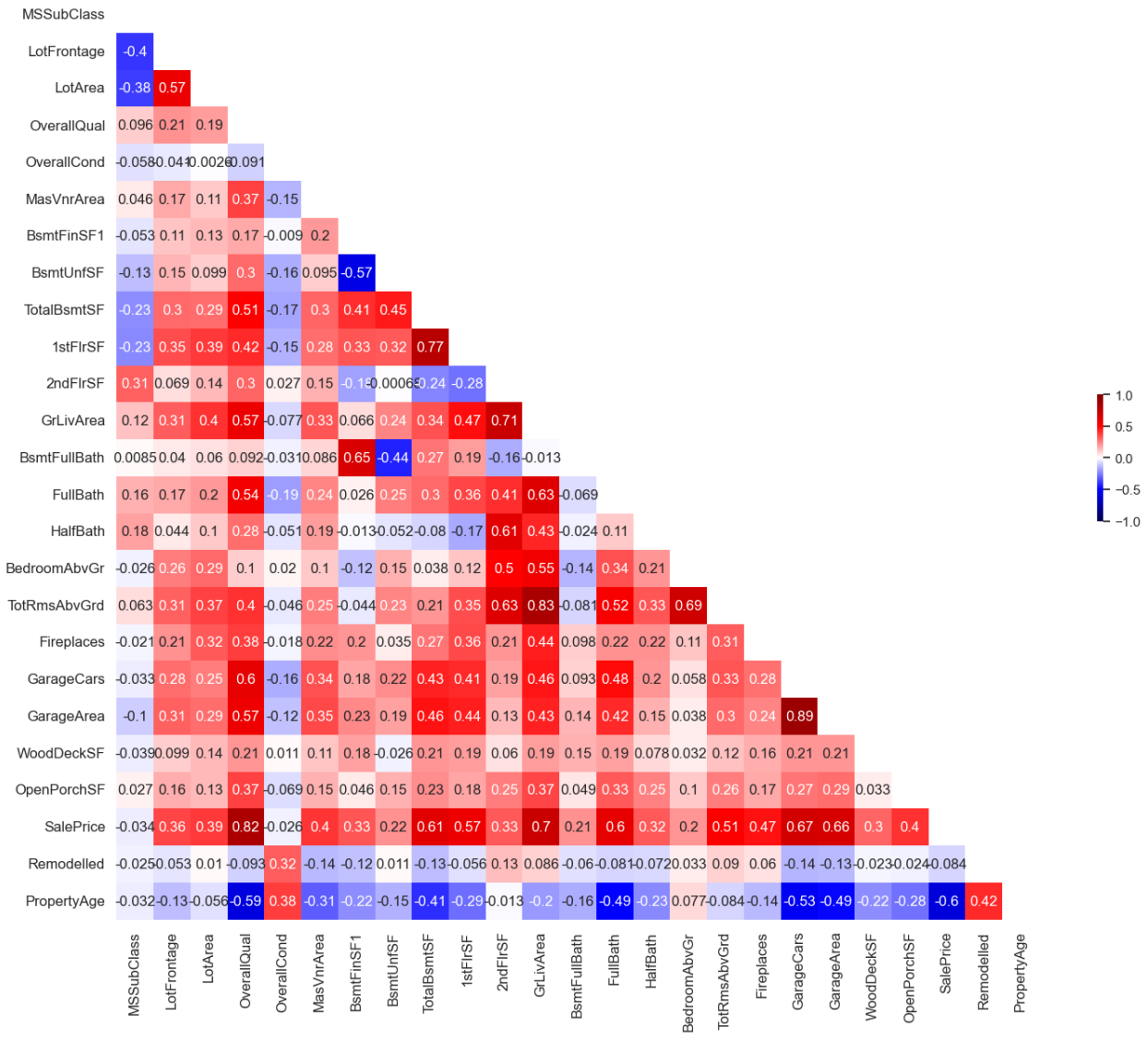

Sale price vs 1stFlrSF
SalePrice
1stFlrSF
13.0
12.5
12.0
11.5
11.0
10.5
500
1000
1500
2000
2500
3000

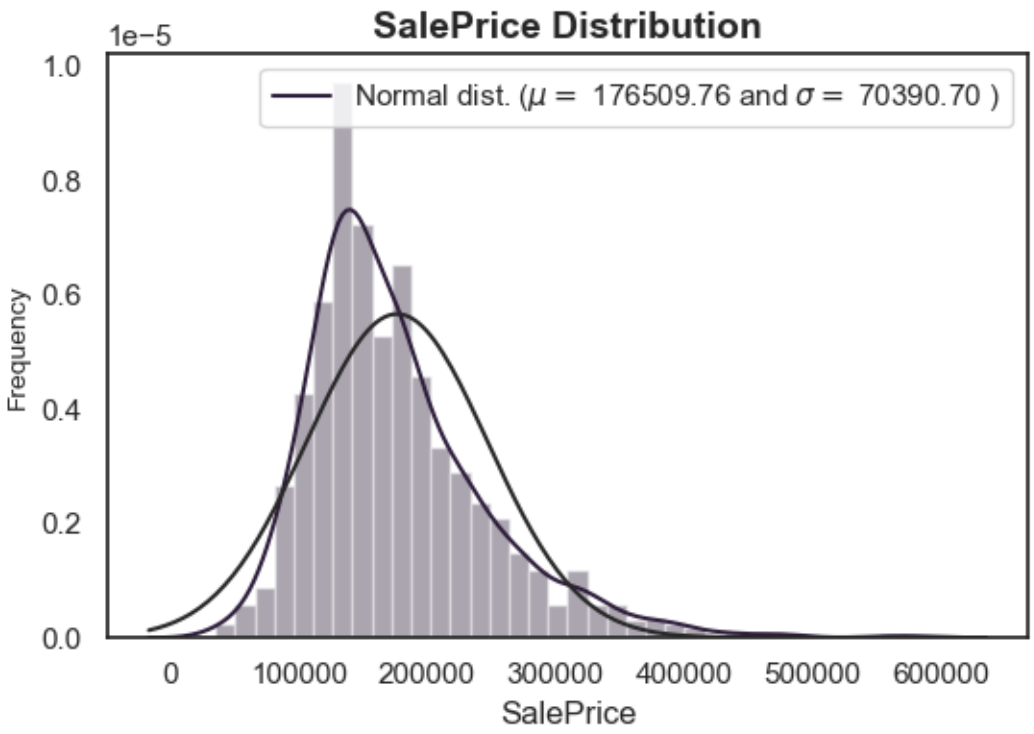

Sale price vs 2ndFlrSF
SalePrice
2ndFlrSF
13.0
12.5
12.0
11.5
11.0
10.5
0
250
500
750
1000
1250
1500
1750

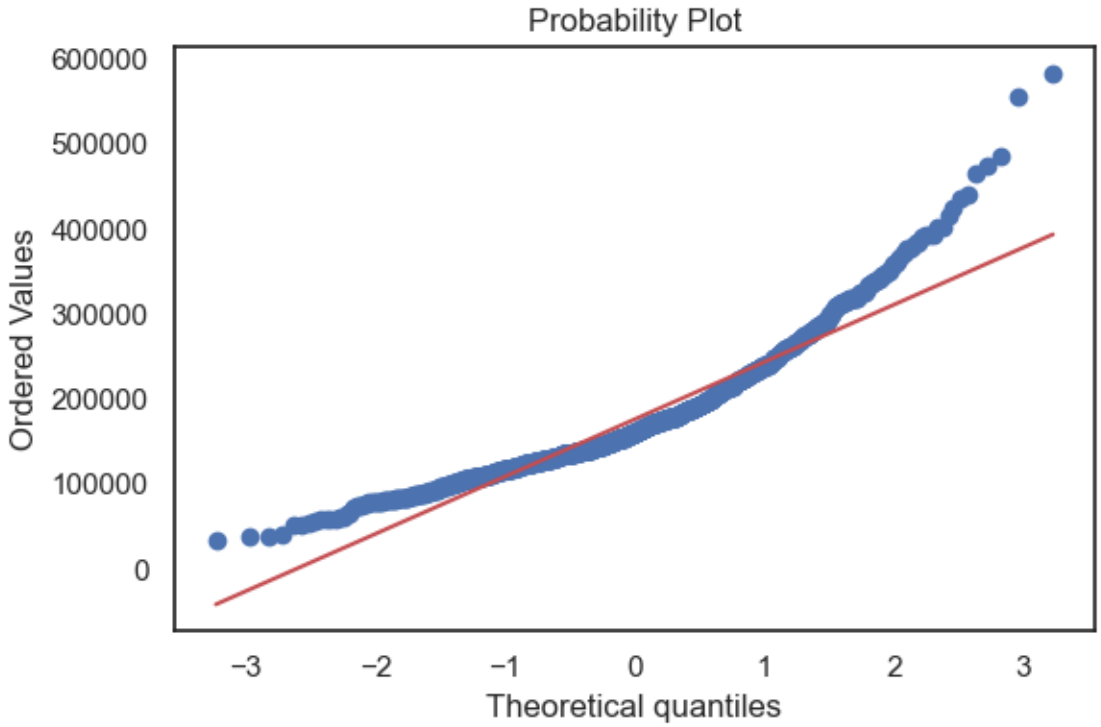

Sale price vs GrLivArea
SalePrice
GrLivArea
13.0
12.5
12.0
11.5
11.0
10.5
500
1000
1500
2000
2500
3000
3500

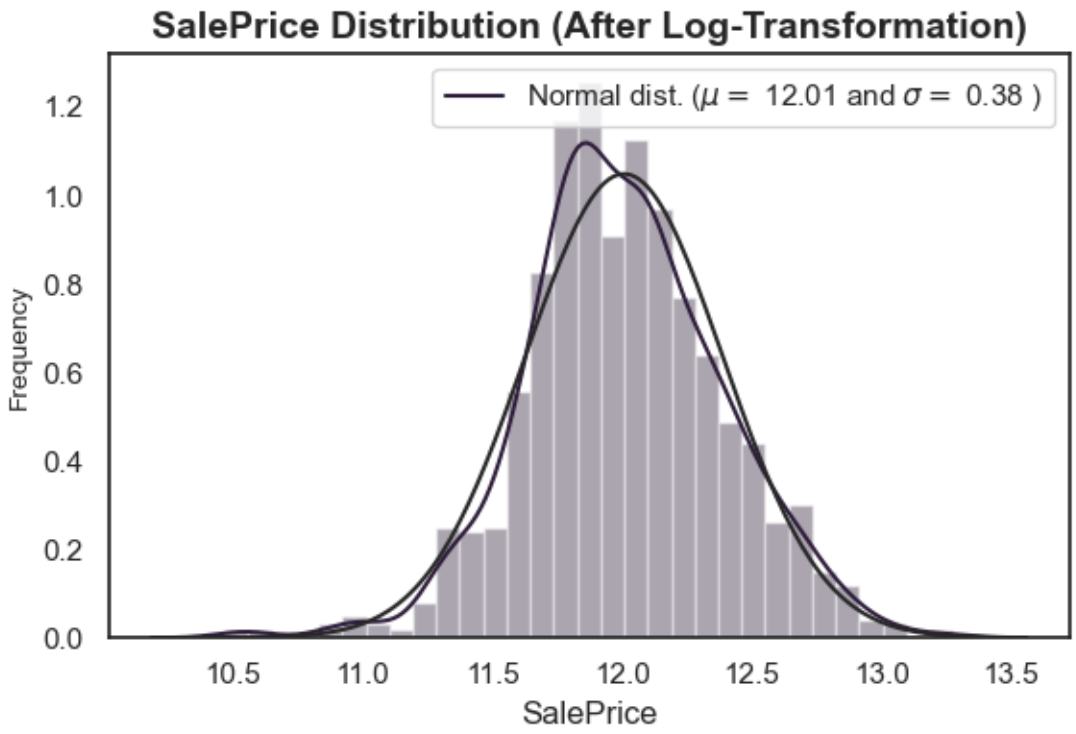

MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
MasVnrArea
BsmtFinSF1
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
GrLivArea
BsmtFullBath
FullBath
HalfBath
BedroomAbvGr
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
SalePrice
Remodelled
PropertyAge
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

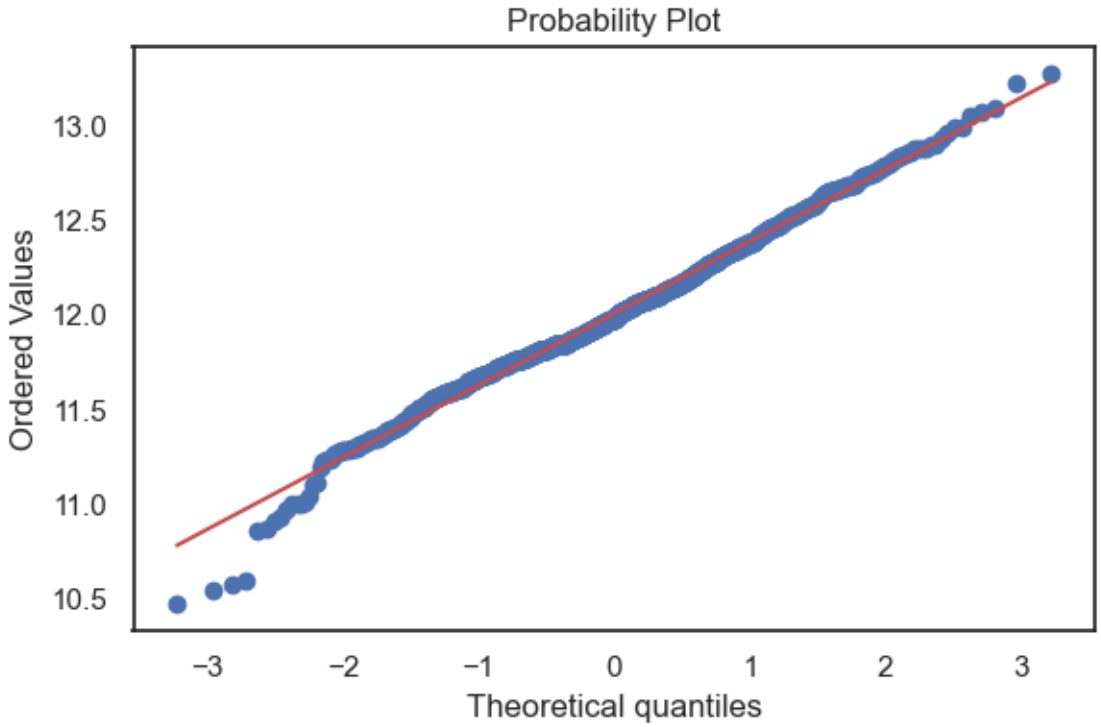

MSSubClass 1.00 -0.40 -0.38 0.10 -0.06 0.05 -0.05 -0.13 -0.23 -0.23 0.31 0.12 0.01 0.16 0.18 -0.03 0.06 -0.02 -0.03 -0.10 -0.04 0.03 -0.03 -0.03 -0.03
LotFrontage -0.40 1.00 0.57 0.21 -0.04 0.17 0.11 0.15 0.30 0.35 0.07 0.31 0.04 0.17 0.04 0.26 0.31 0.21 0.28 0.31 0.10 0.16 0.36 -0.05 -0.13
LotArea -0.38 0.57 1.00 0.19 -0.00 0.11 0.13 0.10 0.29 0.39 0.14 0.40 0.06 0.20 0.10 0.29 0.37 0.32 0.25 0.29 0.14 0.13 0.39 0.01 -0.06
OverallQual 0.10 0.21 0.19 1.00 -0.09 0.37 0.17 0.30 0.51 0.42 0.30 0.57 0.09 0.54 0.28 0.10 0.40 0.38 0.60 0.57 0.21 0.37 0.82 -0.09 -0.59
OverallCond -0.06 -0.04 -0.00 -0.09 1.00 -0.15 -0.01 -0.16 -0.17 -0.15 0.03 -0.08 -0.03 -0.19 -0.05 0.02 -0.05 -0.02 -0.16 -0.12 0.01 -0.07 -0.03 0.32 0.38
MasVnrArea 0.05 0.17 0.11 0.37 -0.15 1.00 0.20 0.09 0.30 0.28 0.15 0.33 0.09 0.24 0.19 0.10 0.25 0.22 0.34 0.35 0.11 0.15 0.40 -0.14 -0.31
BsmtFinSF1 -0.05 0.11 0.13 0.17 -0.01 0.20 1.00 -0.57 0.41 0.33 -0.18 0.07 0.65 0.03 -0.01 -0.12 -0.04 0.20 0.18 0.23 0.18 0.05 0.33 -0.12 -0.22
BsmtUnfSF -0.13 0.15 0.10 0.30 -0.16 0.09 -0.57 1.00 0.45 0.32 -0.00 0.24 -0.44 0.25 -0.05 0.15 0.23 0.04 0.22 0.19 -0.03 0.15 0.22 0.01 -0.15
TotalBsmtSF -0.23 0.30 0.29 0.51 -0.17 0.30 0.41 0.45 1.00 0.77 -0.24 0.34 0.27 0.30 -0.08 0.04 0.21 0.27 0.43 0.46 0.21 0.23 0.61 -0.13 -0.41
1stFlrSF -0.23 0.35 0.39 0.42 -0.15 0.28 0.33 0.32 0.77 1.00 -0.28 0.47 0.19 0.36 -0.17 0.12 0.35 0.36 0.41 0.44 0.19 0.18 0.57 -0.06 -0.29
2ndFlrSF 0.31 0.07 0.14 0.30 0.03 0.15 -0.18 -0.00 -0.24 -0.28 1.00 0.71 -0.16 0.41 0.61 0.50 0.63 0.21 0.19 0.13 0.06 0.25 0.33 0.13 -0.01
GrLivArea 0.12 0.31 0.40 0.57 -0.08 0.33 0.07 0.24 0.34 0.47 0.71 1.00 -0.01 0.63 0.43 0.55 0.83 0.44 0.46 0.43 0.19 0.37 0.70 0.09 -0.20
BsmtFullBath 0.01 0.04 0.06 0.09 -0.03 0.09 0.65 -0.44 0.27 0.19 -0.16 -0.01 1.00 -0.07 -0.02 -0.14 -0.08 0.10 0.09 0.14 0.15 0.05 0.21 -0.06 -0.16
FullBath 0.16 0.17 0.20 0.54 -0.19 0.24 0.03 0.25 0.30 0.36 0.41 0.63 -0.07 1.00 0.11 0.34 0.52 0.22 0.48 0.42 0.19 0.33 0.60 -0.08 -0.49
HalfBath 0.18 0.04 0.10 0.28 -0.05 0.19 -0.01 -0.05 -0.08 -0.17 0.61 0.43 -0.02 0.11 1.00 0.21 0.33 0.22 0.20 0.15 0.08 0.25 0.32 -0.07 -0.23
BedroomAbvGr -0.03 0.26 0.29 0.10 0.02 0.10 -0.12 0.15 0.04 0.12 0.50 0.55 -0.14 0.34 0.21 1.00 0.69 0.11 0.06 0.04 0.03 0.10 0.20 0.03 0.08
TotRmsAbvGrd 0.06 0.31 0.37 0.40 -0.05 0.25 -0.04 0.23 0.21 0.35 0.63 0.83 -0.08 0.52 0.33 0.69 1.00 0.31 0.33 0.30 0.12 0.26 0.51 0.09 -0.08
Fireplaces -0.02 0.21 0.32 0.38 -0.02 0.22 0.20 0.04 0.27 0.36 0.21 0.44 0.10 0.22 0.22 0.11 0.31 1.00 0.28 0.24 0.16 0.17 0.47 0.06 -0.14
GarageCars -0.03 0.28 0.25 0.60 -0.16 0.34 0.18 0.22 0.43 0.41 0.19 0.46 0.09 0.48 0.20 0.06 0.33 0.28 1.00 0.89 0.21 0.27 0.67 -0.14 -0.53
GarageArea -0.10 0.31 0.29 0.57 -0.12 0.35 0.23 0.19 0.46 0.44 0.13 0.43 0.14 0.42 0.15 0.04 0.30 0.24 0.89 1.00 0.21 0.29 0.66 -0.13 -0.49
WoodDeckSF -0.04 0.10 0.14 0.21 0.01 0.11 0.18 -0.03 0.21 0.19 0.06 0.19 0.15 0.19 0.08 0.03 0.12 0.16 0.21 0.21 1.00 0.03 0.30 -0.02 -0.22
OpenPorchSF 0.03 0.16 0.13 0.37 -0.07 0.15 0.05 0.15 0.23 0.18 0.25 0.37 0.05 0.33 0.25 0.10 0.26 0.17 0.27 0.29 0.03 1.00 0.40 -0.02 -0.28
SalePrice -0.03 0.36 0.39 0.82 -0.03 0.40 0.33 0.22 0.61 0.57 0.33 0.70 0.21 0.60 0.32 0.20 0.51 0.47 0.67 0.66 0.30 0.40 1.00 -0.08 -0.60
Remodelled -0.03 -0.05 0.01 -0.09 0.32 -0.14 -0.12 0.01 -0.13 -0.06 0.13 0.09 -0.06 -0.08 -0.07 0.03 0.09 0.06 -0.14 -0.13 -0.02 -0.02 -0.08 1.00 0.42
PropertyAge -0.03 -0.13 -0.06 -0.59 0.38 -0.31 -0.22 -0.15 -0.41 -0.29 -0.01 -0.20 -0.16 -0.49 -0.23 0.08 -0.08 -0.14 -0.53 -0.49 -0.22 -0.28 -0.60 0.42 1.00
MSSubClass LotFrontage LotArea OverallQual OverallCond MasVnrArea BsmtFinSF1 BsmtUnfSF TotalBsmtSF 1stFlrSF 2ndFlrSF GrLivArea BsmtFullBath FullBath HalfBath BedroomAbvGr TotRmsAbvGrd Fireplaces GarageCars GarageArea WoodDeckSF OpenPorchSF SalePrice Remodelled PropertyAge
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


MSSubClass
LotFrontage -0.4
LotArea -0.38 0.57
OverallQual 0.096 0.21 0.19
OverallCond -0.058 0.044 0.0026 0.091
MasVnrArea 0.046 0.17 0.11 0.37 -0.15
BsmtFinSF1 -0.053 0.11 0.13 0.17 -0.009 0.2
BsmtUnfSF -0.13 0.15 0.099 0.3 -0.16 0.095 -0.57
TotalBsmtSF -0.23 0.3 0.29 0.51 -0.17 0.3 0.41 0.45
1stFlrSF -0.23 0.35 0.39 0.42 -0.15 0.28 0.33 0.32 0.77
2ndFlrSF 0.31 0.069 0.14 0.3 0.027 0.15 -0.1 -0.00063 0.24 -0.28
GrLivArea 0.12 0.31 0.4 0.57 -0.077 0.33 0.066 0.24 0.34 0.47 0.71
BsmtFullBath 0.0085 0.04 0.06 0.092 -0.031 0.086 0.65 -0.44 0.27 0.19 -0.16 -0.013
FullBath 0.16 0.17 0.2 0.54 -0.19 0.24 0.026 0.25 0.3 0.36 0.41 0.63 -0.069
HalfBath 0.18 0.044 0.1 0.28 -0.051 0.19 -0.013 -0.052 -0.08 -0.17 0.61 0.43 -0.024 0.11
BedroomAbvGr -0.026 0.26 0.29 0.1 0.02 0.1 -0.12 0.15 0.038 0.12 0.5 0.55 -0.14 0.34 0.21
TotRmsAbvGrd 0.063 0.31 0.37 0.4 -0.046 0.25 -0.044 0.23 0.21 0.35 0.63 0.83 -0.081 0.52 0.33 0.69
Fireplaces -0.021 0.21 0.32 0.38 -0.018 0.22 0.2 0.035 0.27 0.36 0.21 0.44 0.098 0.22 0.22 0.11 0.31
GarageCars -0.033 0.28 0.25 0.6 -0.16 0.34 0.18 0.22 0.43 0.41 0.19 0.46 0.093 0.48 0.2 0.058 0.33 0.28
GarageArea -0.1 0.31 0.29 0.57 -0.12 0.35 0.23 0.19 0.46 0.44 0.13 0.43 0.14 0.42 0.15 0.038 0.3 0.24 0.89
WoodDeckSF -0.039 0.099 0.14 0.21 0.011 0.11 0.18 -0.026 0.21 0.19 0.06 0.19 0.15 0.19 0.078 0.032 0.12 0.16 0.21 0.21
OpenPorchSF 0.027 0.16 0.13 0.37 -0.069 0.15 0.046 0.15 0.23 0.18 0.25 0.37 0.049 0.33 0.25 0.1 0.26 0.17 0.27 0.29 0.033
SalePrice -0.034 0.36 0.39 0.82 -0.026 0.4 0.33 0.22 0.61 0.57 0.33 0.7 0.21 0.6 0.32 0.2 0.51 0.47 0.67 0.66 0.3 0.4
Remodelled -0.025 0.053 0.01 -0.093 0.32 -0.14 -0.12 0.011 -0.13 -0.056 0.13 0.086 -0.06 -0.081 -0.072 0.033 0.09 0.06 -0.14 -0.13 -0.023 -0.024 -0.084
PropertyAge -0.032 -0.13 -0.056 -0.59 0.38 -0.31 -0.22 -0.15 -0.41 -0.29 -0.013 -0.2 -0.16 -0.49 -0.23 0.077 -0.084 -0.14 -0.53 -0.49 -0.22 -0.28 -0.6 0.42
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
MasVnrArea
BsmtFinSF1
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
GrLivArea
BsmtFullBath
FullBath
HalfBath
BedroomAbvGr
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
SalePrice
Remodelled
PropertyAge
1.0
0.5
0.0
−0.5
−1.0


SalePrice Distribution
1e−5
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
Normal dist. (μ = 176509.76 and σ = 70390.70)
0
100000
200000
300000
400000
500000
600000
SalePrice


Probability Plot
600000
500000
400000
300000
200000
100000
0
Ordered Values
−3
−2
−1
0
1
2
3
Theoretical quantiles


SalePrice Distribution (After Log-Transformation)
Normal dist. (μ = 12.01 and σ = 0.38)
Frequency
SalePrice


Probability Plot
Ordered Values
Theoretical quantiles

Models Development & Evaluation

Our objective is to predict house price and analyses feature impacting Sale price. This problem can be solved using regression-based machine learning algorithm like linear regression.

For that purpose, we convert categorical variable into numerical features. Once data encoding is done then data is scaled using standard scalar.

Final model is built over this scaled data.

For building ML model before implementing regression algorithm, data is split in training & test data using train_test_split from model_selection module of sklearn library.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

After that model is train with various regression algorithm and 10-fold cross validation is performed.

Hyper-parameter tuning performed to build more accurate model out of best model.

The different regression algorithm used in this project to build ML model are:

- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- Ridge Regression
- XGB Regressor
- Extra Tree Regressor
- Lasso

Metrics used for evaluation:

1. ***Mean absolute error*** which gives magnitude of difference between the prediction of an observation and the true value of that observation.

2. ***Root mean square error*** is one of the most commonly used measures for evaluating the quality of predictions.

3. ***R2 score*** which tells us how accurate our model predict the results, is going to important evaluation criteria along with Cross validation score.

4. ***Cross Validation*** Score

Getting the maximum R2-Score and best random state

```
maxR2_score=0
maxRS=0
for i in range(50,500):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=i, test_size=0.2)
    lin_reg=LinearRegression()
    lin_reg.fit(X_train,Y_train)
    y_pred=lin_reg.predict(X_test)
    R2=r2_score(Y_test,y_pred)
    if R2>maxR2_score:
        maxR2_score=R2
        maxRS=i
print('Best R2 Score is', maxR2_score ,'on Random_state', maxRS)
```

```
Best R2 Score is 0.9377017379319479 on Random_state 389
```

Linear Regression

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
lin_reg= LinearRegression()
lin_reg.fit(X_train, Y_train)
y_pred = lin_reg.predict(X_test)
print('\033[1m'+ 'Error :'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+' R2 Score :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error :
Mean absolute error : 0.07432000129834601
Mean squared error : 0.008873609812502908
Root Mean squared error : 0.09419983976898744
 R2 Score :
93.77017379319479
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(lin_reg, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',lin_reg,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : LinearRegression() :

Mean CV Score : 0.8882344413647374
Difference in R2 & CV Score: 4.946729656721047
```

Random Forest Regressor

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
rfc = RandomForestRegressor()
rfc.fit(X_train, Y_train)
y_pred = rfc.predict(X_test)
print('\033[1m'+ 'Error of Random Forest Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Random Forest Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Random Forest Regressor:
Mean absolute error : 0.08371793122096458
Mean squared error : 0.012979245825869097
Root Mean squared error : 0.1139264930815879
R2 Score of Random Forest Regressor :
90.88776185801667
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(rfc, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',rfc,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : RandomForestRegressor() :

Mean CV Score : 0.8690939338599692
Difference in R2 & CV Score: 3.9783684720197527
```

Decision Tree Regressor

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
dtc = DecisionTreeRegressor()
dtc.fit(X_train, Y_train)
y_pred = dtc.predict(X_test)
print('\033[1m'+ 'Error of Decision Tree Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Decision Tree Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Decision Tree Regressor:
Mean absolute error : 0.14827231211530364
Mean squared error : 0.042658804950094324
Root Mean squared error : 0.20654008073517915
R2 Score of Decision Tree Regressor :
70.05086468252884
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(dtc, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',dtc,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : DecisionTreeRegressor() :

Mean CV Score : 0.7057804422145301
Difference in R2 & CV Score: -0.5271795389241731
```

Extra Tree Regressor

```
: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
  etc = ExtraTreesRegressor()
  etc.fit(X_train, Y_train)
  y_pred = etc.predict(X_test)
  print('\033[1m'+ 'Error of Extra Tree Regressor:'+ '\033[0m')
  print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
  print('Mean squared error :', mean_squared_error(Y_test, y_pred))
  print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
  print('\033[1m'+'R2 Score of Extra Tree Regressor :'+'\033[0m')
  print(r2_score(Y_test,y_pred)*100)
```

```
Error of Extra Tree Regressor:
Mean absolute error : 0.08289096308857113
Mean squared error : 0.012717187337500189
Root Mean squared error : 0.1127705073922264
R2 Score of Extra Tree Regressor :
91.07174322220244
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(etc, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',etc,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : ExtraTreesRegressor() :

Mean CV Score : 0.8684877195433108
Difference in R2 & CV Score: 4.222971267871358
```

Ridge Regression

```
from sklearn.linear_model import Ridge
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
rd = Ridge()
rd.fit(X_train, Y_train)
y_pred = rd.predict(X_test)
print('\033[1m'+ 'Error of Ridge Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Ridge Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Ridge Regressor:
Mean absolute error : 0.07430148126179226
Mean squared error : 0.008869698701417256
Root Mean squared error : 0.0941790778326973
R2 Score of Ridge Regressor :
93.77291963652732
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(rd, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',rd,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : Ridge() :

Mean CV Score : 0.888253856003127
Difference in R2 & CV Score: 4.94753403621462
```

XGB Regressor

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
xgb = XGBRegressor()
xgb.fit(X_train, Y_train)
y_pred = xgb.predict(X_test)
print('\033[1m'+ 'Error of XGB Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of XGB Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of XGB Regressor:
Mean absolute error : 0.08240442966733597
Mean squared error : 0.01250907594898242
Root Mean squared error : 0.11184398038778136
R2 Score of XGB Regressor :
91.21785036568939
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(xgb, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',xgb,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...) :

Mean CV Score : 0.8645505426544006
Difference in R2 & CV Score: 4.762796100249332
```

Lasso

```
from sklearn.linear_model import Lasso
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 389, test_size=0.2)
lasso = Lasso()
lasso.fit(X_train, Y_train)
y_pred = lasso.predict(X_test)
print('\033[1m'+ 'Error of Lasso:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Lasso :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Lasso:
Mean absolute error : 0.11392084805711976
Mean squared error : 0.02347483154001758
Root Mean squared error : 0.1532149847110836
R2 Score of Lasso :
83.5192076484723
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(lasso, X, Y, cv=10)
print('\033[1m'+'Cross Validation Score :',lasso,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : Lasso() :

Mean CV Score : 0.7909305855339235
Difference in R2 & CV Score: 4.42614909507995
```

Grid Search and Hyper Parameter Tuning of Ridge as it has the best R2-Score

```
from sklearn.model_selection import GridSearchCV
parameters = {"fit_intercept":[True,False],"copy_X":[True,False],
              "solver":['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs']}

clf = GridSearchCV(rd, param_grid=parameters, n_jobs=-1)
clf.fit(X_train, Y_train)

clf.best_params_
```

```
{'copy_X': True, 'fit_intercept': True, 'solver': 'svd'}
```

```
Final_model=Ridge(alpha=100,copy_X=True,fit_intercept=True,solver="svd" )
Final_model.fit(X_train,Y_train)
pred=Final_model.predict(X_test)
print('R2_Score:',r2_score(Y_test,pred)*100)
print('mean_squared_error:',mean_squared_error(Y_test,pred))
print('mean_absolute_error:',mean_absolute_error(Y_test,pred))
print("RMSE value:",np.sqrt(mean_squared_error(Y_test, pred)))
```

```
R2_Score: 93.81018322013375
mean_squared_error: 0.008816621377883258
mean_absolute_error: 0.07426832038291135
RMSE value: 0.09389686564461701
```

Conclusions & Findings

- 79.5% of House properties belongs to Low Density Residential Area followed by 14 % of properties belong to Medium Density Residential Area.
- Very Few properties (0.8%) belong to Commercial zone.
- Most of property for sale have overall condition rating of either 5 or 6.
- We already know of 80% of housing data belongs to Low density Residential Area
- Sale Price inside RL Zone is much higher than another remaining zone.
- House properties having Overall condition Rating of 8 & 9 have low price compare to others. This indicate that Overall Condition Rating is Not significant factor in determination of Sale price.
- 89.6% of House properties are near flat level surface.
- Also, price for Flat level surface house is much higher than other land contour.
- Around 72 % of house comes with inside Lot configuration.
- Cul-de-sac has maximum Mean Sale Price among all lot configuration.
- 63.4% house properties are regular in shape. Sale Price of property with slight irregular shape is higher than regular shape.
- There is No Significant relationship found between Sale price & Lot area. *As Overall Quality of House Increase the Sale Price of House also Increases*
- Cheapest Houses belong to Inside lot configuration while Costlier houses belongs to Corner Lot Configuration.
- More than 950 house properties are with building type Single-family Detached.

- More than 50% of house properties come with Overall Condition Rating of 5.
- More than 75% of house properties come with overall Quality Rating varies between 5 and 6.
- More than 500 House Properties comes with one story dwelling.
- Around 1000 sales happen by Conventional Warranty Deed.
- Home just constructed and sold category are exceptionally much costlier than anyone else.
- All loan-based sale is below 300000.
- We can see that Sale with condition like Abnorml, Family, Alloca, AdjLand are below the price of 300000.
- Maximum Base Price for House comes from Partial category- (associated with Uncompleted New Home) is higher than rest.
- Minimum base price comes from Normal condition sale and also highest sale price comes from this category.
- *As total floor area increases the sale price also get increases corresponding the overall quality of House.*
- More than 75% House properties come with Gable Roof Style followed by around 15 % house properties with Hip Style.
- Hip style Roof are much costlier than remaining roof style
- For High floor area construction mainly Hip style Roof is used and invariably high-cost properties mostly comes up with Hip Style Roof.
- More than 90% Properties in Data set made with roof material of Standard (Composite) Shingle.
- Wood Shingles is Costlier Material compare to rest.
- Around 60% of house properties come with Average Exterior quality and all of them below 400000.

- Very few House Properties comes with Excellent Exterior Quality.
- Costlier house properties come with Good & Excellent exterior quality.
- 44.2% Properties with CBlock Foundation & 43.9% housing property come with PConc Foundation.
- Pconc Foundation is mostly use in costly housing properties.

Limitations & Scope for future work

There is a large number of missing values presents in this data set, so we have to fill those missing values in correct manner. We can improve our model accuracy with some feature engineering and by doing some extensive hyper-parameter tuning on it.

Hardware and Software Used

Processor	AMD A8-7410 APU with AMD Radeon R5 Graphics 2.20 GHz
Installed RAM	4.00 GB (3.45 GB usable)

Windows specifications

Edition	Windows 10 Home Single Language
Version	22H2

Software Used

Anaconda

Jupyter Notebook

SCikit Learn and general data wrangling tools of Python