Housing Price Prediction Project

Submitted by:

Rajashri Sadafule Darveshi

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to the sources Medium, Analytics Vidhya, TowardsDataScience, Data Training Projects and youtube channel’s which helped me to for this project. I would like to thank FlipRobo for giving me this opportunity to develop and accomplish this project.

INTRODUCTION

Problem Statement:

Houses are one of the necessary need 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 modelling, Market mix modelling, 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.

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 below.

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:

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

Problem Understanding:

- ✓ 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.
- ✓ House Price prediction, is important to drive Real Estate efficiency. As earlier, House prices were determined by calculating the acquiring and selling price in a locality.
- ✓ Therefore, the House Price prediction model is very essential in filling the information gap and improve Real Estate efficiency.
- ✓ The aim is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities.

- ✓ By analysing previous market trends and price ranges, and also upcoming developments future prices will be predicted. Cost of property depending on number of attributes considered.
- ✓ Now as a data scientist our work is to analyse the dataset and apply our skills towards predicting house price.

What is Housing Price Prediction?

- ✓ Prediction house prices are expected to help people who plan to buy a house so they can know the price range in the future, then they can plan their finance well. In addition, house price predictions are also beneficial for property investors to know the trend of housing prices in a certain location.

Importance of Housing Price Prediction:

- ✓ House Price prediction, is important to drive Real Estate efficiency. As earlier, House prices were determined by calculating the acquiring and selling price in a locality. Therefore, the House Price prediction model is very essential in filling the information gap and improve Real Estate efficiency

Data Sources:

The training data and testind data for this project are available in csv file.

About the data: Details of dataset are as follows

- ➢ Number of data points in train data:1168
- ➢ Number of features in train data: 81
- ➢ Number of data points in test data: 292
- ➢ Number of features in test data: 80

We will understand all features are their details of the dataset are as follows:

1. MSSubClass: Identifies the type of dwelling involved in the sale.
2. MSZoning: Identifies the general zoning classification of the sale.
3. LotFrontage: Linear feet of street connected to property
4. LotArea: Lot size in square feet
5. Street: Type of road access to property
6. Alley: Type of alley access to property
7. LotShape: General shape of property
8. LandContour: Flatness of the property
9. Utilities: Type of utilities available
10. LotConfig: Lot configuration

11. LandSlope: Slope of property
12. Neighborhood: Physical locations within Ames city limits
13. Condition2: Proximity to various conditions (if more than one is present)
14. BldgType: Type of dwelling
15. HouseStyle: Style of dwelling
16. OverallQual: Rates the overall material and finish of the house
17. OverallCond: Rates the overall condition of the house
18. YearBuilt: Original construction date
19. YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
20. RoofStyle: Type of roof
21. RoofMatl: Roof material
22. Exterior1st: Exterior covering on house
23. Exterior2nd: Exterior covering on house (if more than one material)
24. MasVnrType: Masonry veneer type
25. MasVnrArea: Masonry veneer area in square feet
26. ExterQual: Evaluates the quality of the material on the exterior
27. ExterCond: Evaluates the present condition of the material on the exterior
28. Foundation: Type of foundation
29. BsmtQual: Evaluates the height of the basement
30. BsmtCond: Evaluates the general condition of the basement
31. BsmtExposure: Refers to walkout or garden level walls
32. BsmtFinType1: Rating of basement finished area
33. BsmtFinSF1: Type 1 finished square feet
34. BsmtFinType2: Rating of basement finished area (if multiple types)
35. BsmtFinSF2: Type 2 finished square feet
36. BsmtUnfSF: Unfinished square feet of basement area
37. TotalBsmtSF: Total square feet of basement area
38. Heating: Type of heating
39. HeatingQC: Heating quality and condition
40. CentralAir: Central air conditioning
41. Electrical: Electrical system
42. 1stFlrSF: First Floor square feet
43. 2ndFlrSF: Second floor square feet
44. LowQualFinSF: Low quality finished square feet (all floors)
45. GrLivArea: Above grade (ground) living area square feet
46. BsmtFullBath: Basement full bathrooms
47. BsmtHalfBath: Basement half bathrooms

48. FullBath: Full bathrooms above grade
49. HalfBath: Half baths above grade
50. Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
51. Kitchen: Kitchens above grade
52. KitchenQual: Kitchen quality
53. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
54. Functional: Home functionality (Assume typical unless deductions are warranted)
55. Fireplaces: Number of fireplaces
56. FireplaceQu: Fireplace quality
57. GarageType: Garage location
58. GarageYrBlt: Year garage was built
59. GarageFinish: Interior finish of the garage
60. GarageCars: Size of garage in car capacity
61. GarageArea: Size of garage in square feet
62. GarageQual: Garage quality
63. GarageCond: Garage condition
64. PavedDrive: Paved driveway
65. WoodDeckSF: Wood deck area in square feet
66. OpenPorchSF: Open porch area in square feet
67. EnclosedPorch: Enclosed porch area in square feet
68. 3SsnPorch: Three season porch area in square feet
69. ScreenPorch: Screen porch area in square feet
70. PoolArea: Pool area in square feet
71. PoolQC: Pool quality
72. Fence: Fence quality
73. MiscFeature: Miscellaneous feature not covered in other categories
74. MiscVal: $Value of miscellaneous feature
75. MoSold: Month Sold (MM)
76. YrSold: Year Sold (YYYY)
77. SaleType: Type of sale
78. SaleCondition: Condition of sale

```
df=pd.read_csv('house_train.csv')
df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

Housing Price Prediction dataset having 1168 rows and 81 features.

Where **SalePrice** is the resultant feature

Features names are as follow.

```
df.keys()
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Exploratory Data Analysis:

Dataset contains Categorical and Numericle type data.

Above details features details we get the datatypes of features. This gives the information about the dataset which includes indexing type, column type, contains null values and memory usage.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
 14  Condition2     1168 non-null   object
 15  BldgType       1168 non-null   object
 16  HouseStyle     1168 non-null   object
 17  OverallQual    1168 non-null   int64
 18  OverallCond    1168 non-null   int64
 19  YearBuilt      1168 non-null   int64
 20  YearRemodAdd   1168 non-null   int64
 21  RoofStyle      1168 non-null   object
 22  RoofMatl       1168 non-null   object
 23  Exterior1st    1168 non-null   object
 24  Exterior2nd    1168 non-null   object
 25  MasVnrType     1161 non-null   object
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object
 28  ExterCond      1168 non-null   object
 29  Foundation     1168 non-null   object
 29  Foundation     1168 non-null   object
 30  BsmtQual       1138 non-null   object
 31  BsmtCond       1138 non-null   object
 32  BsmtExposure   1137 non-null   object
 33  BsmtFinType1   1138 non-null   object
 34  BsmtFinSF1     1168 non-null   int64
 35  BsmtFinType2   1137 non-null   object
 36  BsmtFinSF2     1168 non-null   int64
 37  BsmtUnfSF      1168 non-null   int64
 38  TotalBsmtSF    1168 non-null   int64
 39  Heating        1168 non-null   object
 40  HeatingQC      1168 non-null   object
 41  CentralAir     1168 non-null   object
 42  Electrical     1168 non-null   object
 43  1stFlrSF       1168 non-null   int64
 44  2ndFlrSF       1168 non-null   int64
 45  LowQualFinSF   1168 non-null   int64
 46  GrLivArea      1168 non-null   int64
 47  BsmtFullBath   1168 non-null   int64
 48  BsmtHalfBath   1168 non-null   int64
 49  FullBath       1168 non-null   int64
 50  HalfBath       1168 non-null   int64
 51  BedroomAbvGr   1168 non-null   int64
 52  KitchenAbvGr   1168 non-null   int64
 53  KitchenQual    1168 non-null   object
 54  TotRmsAbvGrd   1168 non-null   int64
 55  Functional     1168 non-null   object
 56  Fireplaces     1168 non-null   int64
 57  FireplaceQu    617 non-null    object
 58  GarageType     1104 non-null   object
 59  GarageYrBlt    1104 non-null   float64
 60  GarageFinish   1104 non-null   object
 61  GarageCars     1168 non-null   int64
 62  GarageArea     1168 non-null   int64
 63  GarageQual     1104 non-null   object
 64  GarageCond     1104 non-null   object
 65  PavedDrive     1168 non-null   object
 66  WoodDeckSF     1168 non-null   int64
 67  OpenPorchSF    1168 non-null   int64
 68  EnclosedPorch  1168 non-null   int64
 69  3SsnPorch      1168 non-null   int64
 70  ScreenPorch    1168 non-null   int64
 71  PoolArea       1168 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          237 non-null    object
 74  MiscFeature    44 non-null     object
 75  MiscVal        1168 non-null   int64
 76  MoSold         1168 non-null   int64
 77  YrSold         1168 non-null   int64
 78  SaleType       1168 non-null   object
 79  SaleCondition  1168 non-null   object
 80  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 739.2+ KB
```

The dataset contains the details of the employees who are working in an organization. The dataset contains both dependent and independent variables and also contains both categorical and numerical data. In this dataset "**SalesPrice**"" is our target variable which has continous data. So this is a "**Regression type**" problem in which we need to predict the house price for given independanat features.

We an see number of unique contain present in each feature.

```
pd.set_option('display.max_rows',None)
df.nunique()
```

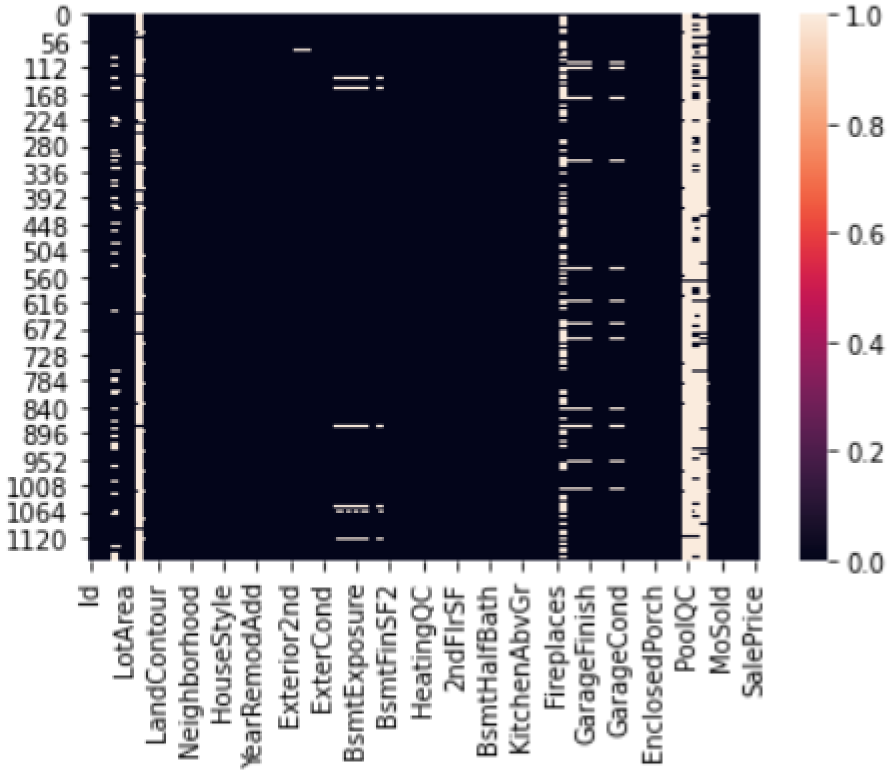
```
Id               1168
MSSubClass         15
MSZoning            5
LotFrontage       106
LotArea           892
Street              2
Alley               2
LotShape            4
LandContour         4
Utilities           1
LotConfig           5
LandSlope           3
Neighborhood       25
Condition1          9
Condition2          8
BldgType            5
HouseStyle          8
OverallQual        10
OverallCond         9
YearBuilt         110
YearRemodAdd       61
RoofStyle           6
RoofMatl            8
Exterior1st        14
Exterior2nd        15
MasVnrType          4
MasVnrArea        283
ExterQual           4
ExterCond           5
Foundation          6
BsmtQual            4
BsmtCond            4
BsmtExposure        4
BsmtFinType1        6
BsmtFinSF1        551
BsmtFinType2        6
BsmtFinSF2        122
BsmtUnfSF         681
TotalBsmtSF       636
Heating             6
HeatingQC           5
CentralAir          2
Electrical          5
1stFlrSF          669
2ndFlrSF          351
LowQualFinSF       21
GrLivArea         746
BsmtFullBath        4
BsmtHalfBath        3
FullBath            4
HalfBath            3
BedroomAbvGr        8
KitchenAbvGr        4
KitchenQual         4
TotRmsAbvGrd       12
Functional          7
Fireplaces          4
FireplaceQu         5
GarageType          6
GarageYrBlt        97
GarageFinish        3
GarageCars          5
GarageArea        392
GarageQual          5
GarageCond          5
PavedDrive          3
WoodDeckSF        244
OpenPorchSF       176
EnclosedPorch     106
3SsnPorch          18
ScreenPorch        65
PoolArea            8
PoolQC              3
Fence               4
MiscFeature         4
MiscVal            20
MoSold             12
YrSold              5
SaleType            9
SaleCondition       6
SalePrice         581
dtype: int64
```

Detect the missing values:

The dataset has missing values we can see with isnull().sum() function and with heatmap graph.

```
sns.heatmap(df.isnull())  #checking null values
```

```
<AxesSubplot:>
```



In dataset ID is just for serial number not giving any information so we will drop ID column. Then 'Alley', 'MiscFeature', 'PoolQC' these columns are having more than 80% NA data so we will drop these columns

Statistical Analysis of dataset:

We will use describe() method for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. As our dataset having both numeric and object series and also the DataFrame column sets of mixed data types. Describe methode uses columns contain continuous type of data

```python
# Get unique and top values for the dataset
df.describe()
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeck
count	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	1168.000000	...	1168.0000
mean	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	46.647260	...	96.2063
std	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.664785	163.520016	...	126.1589
min	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000	...	0.0000
25%	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000	...	0.0000
50%	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000	...	0.0000
75%	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000	...	171.0000
max	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	...	857.0000

8 rows × 37 columns

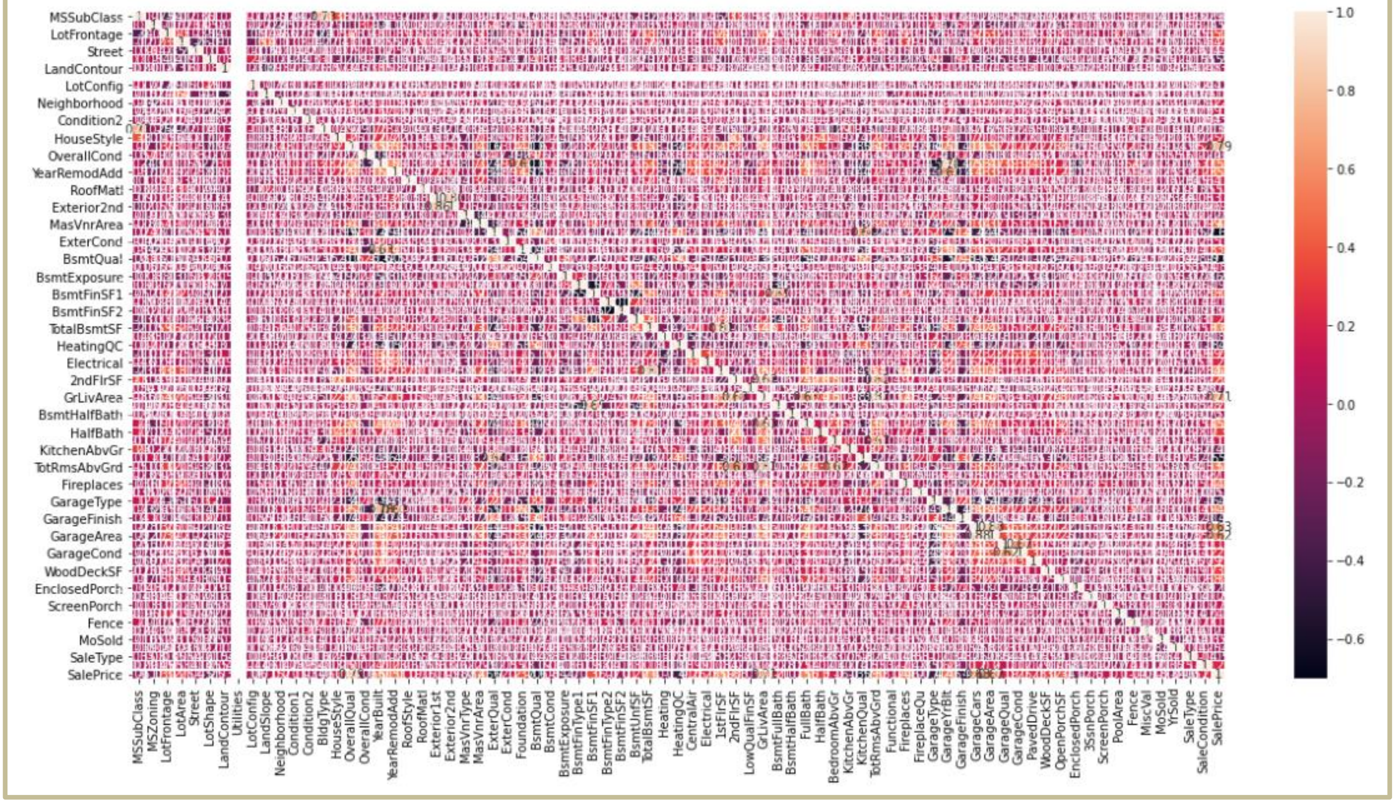
We can observe the following things.

- While checking the info of the datasets, found some columns with more than 80% null values
- Features LotFrontage, MasvnrArea, GarageyrBlt will fill NA with mean values
- Now remaining null catogoricle features values will replace with NA
- While checking for null values I found null values in most of the columns and I have used imputation method to replace those null values (mode for categorical column and mean for numerical columns).

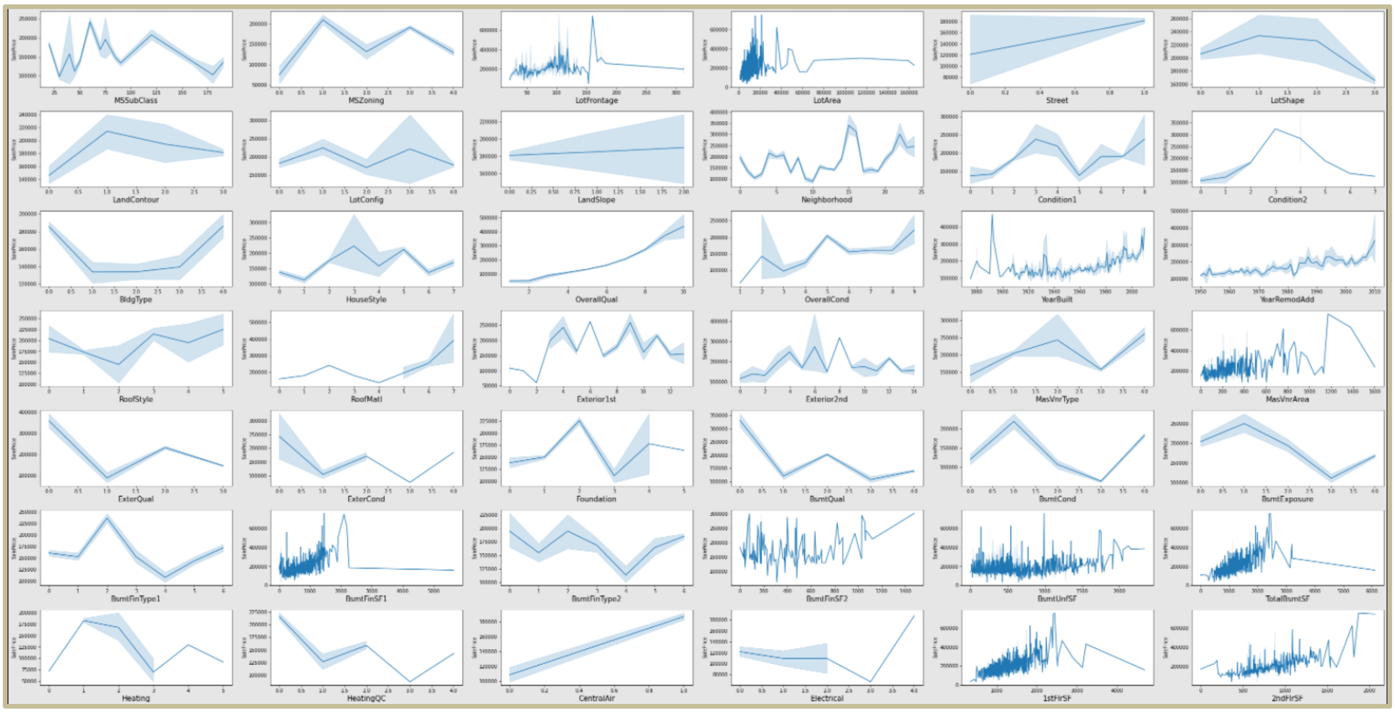
```python
df['BsmtQual'] = df['BsmtQual'].fillna('NA')
df['BsmtCond'] = df['BsmtCond'].fillna('NA')
df['BsmtExposure'] = df['BsmtExposure'].fillna('NA')
df['BsmtFinType1'] = df['BsmtFinType1'].fillna('NA')
df['BsmtFinType2'] = df['BsmtFinType2'].fillna('NA')
df['FireplaceQu'] = df['FireplaceQu'].fillna('NA')
df['GarageType'] = df['GarageType'].fillna('NA')
df['GarageYrBlt'] = df['GarageYrBlt'].fillna(0)
df['GarageFinish'] = df['GarageFinish'].fillna('NA')
df['GarageQual'] = df['GarageQual'].fillna('NA')
df['GarageCond'] = df['GarageCond'].fillna('NA')
df['Fence'] = df['Fence'].fillna('NA')
df['MasVnrType'] = df['MasVnrType'].fillna('NA')
```

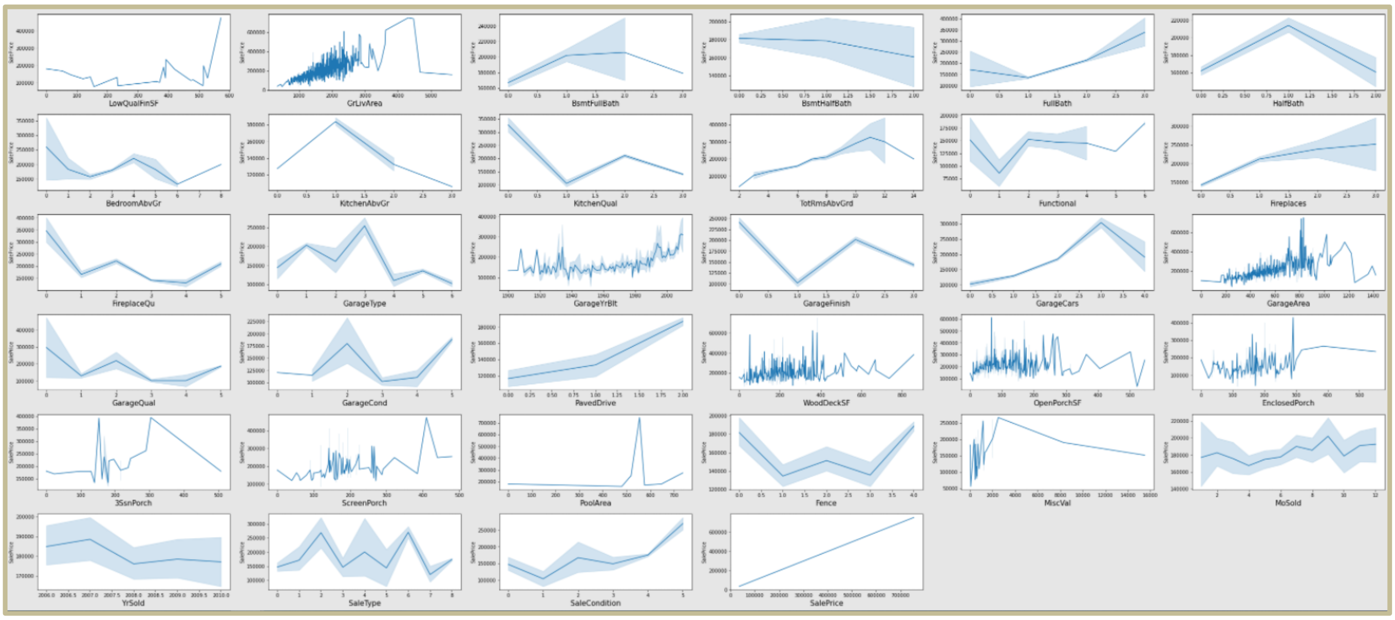
Correlation heatmap is graphical representation of correlation matrix representing correlation between different variables. As in this dataset more than 75 features present so with heatmap its difficult to correlate the features.

```
# Visualizing the correlation matrix by plotting heat map.
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),linewidths=.1, annot = True)
plt.yticks(rotation=0);
```

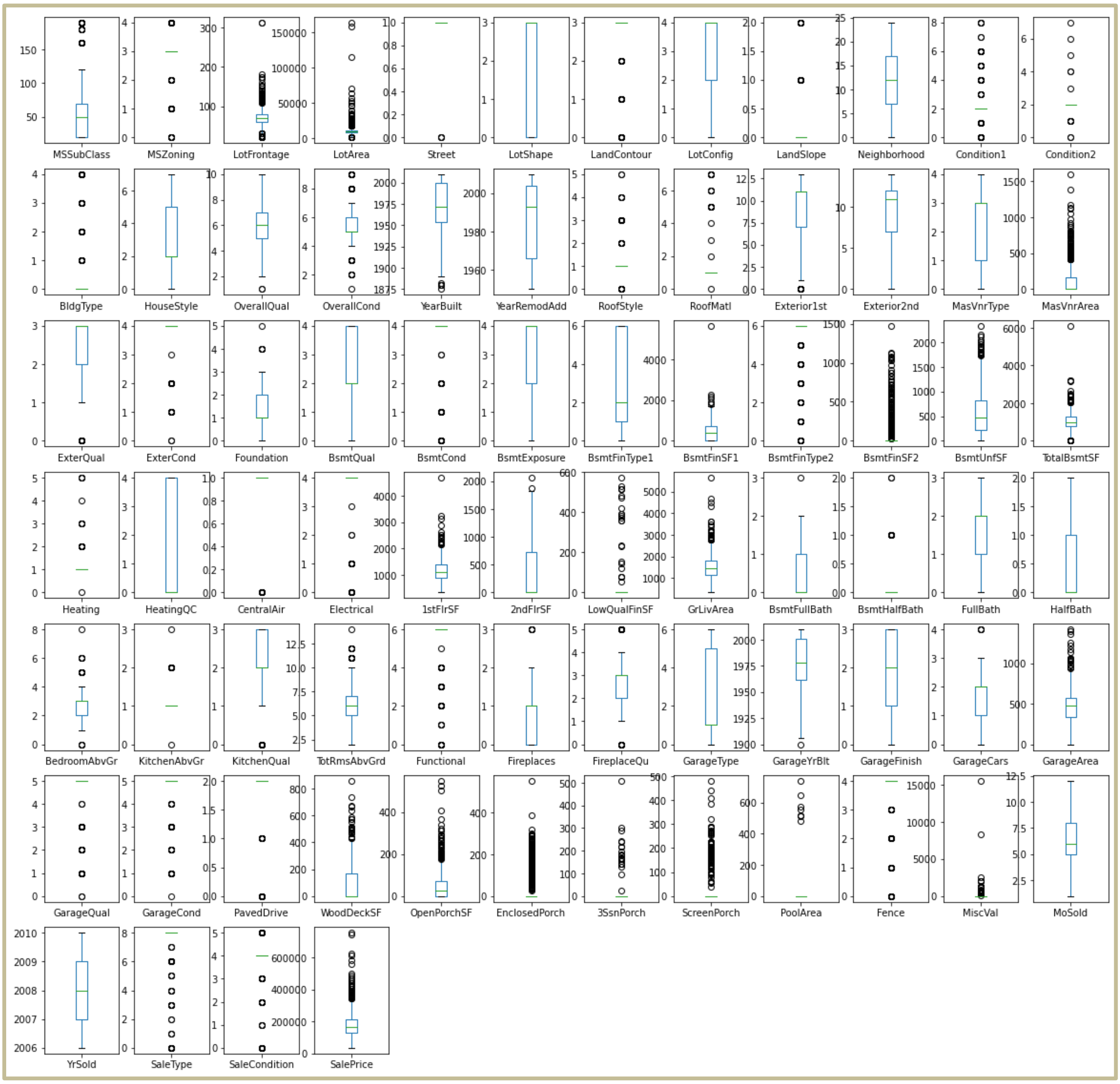


Visualizing the outliers using the lineplot: Bivariate Analysis we can do analysis with salesprice label





Visualizing the outliers using the boxplot:



- ✓ Box plot for each pair of categorical features that shows the relation with the median sale price for all the sub categories in each categorical feature. And also for continuous numerical variables I have used reg plot to show the relationship between continuous numerical variable and target variable.

- ✓ Found that there is a linear relationship between continuous numerical variable and SalesPrice.

- ✓ we can observe these features are having outliers 'LotFrontage', 'LotArea','MasVnrArea', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', 'LowQualFinSF', 'GrLivArea', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'Fence', 'MiscVal', 'SaleType', 'SaleCondition' we will try to remove outliers with zscore

Removing outliers by Zscore and IQR Methode

```
# removing outliers by Zscore
features=['LotFrontage','LotArea','MasVnrArea','BsmtFinSF2','BsmtUnfSF','1stFlrSF','LowQualFinSF',
         'GrLivArea','GarageArea','WoodDeckSF','OpenPorchSF','EnclosedPorch','3SsnPorch','ScreenPorch','Fence','MiscVal','SaleType','Sal
eCondition']
from scipy.stats import zscore
z=np.abs(zscore(df))
df1=df[(z<8).all(axis=1)]
df1.head(5)
```

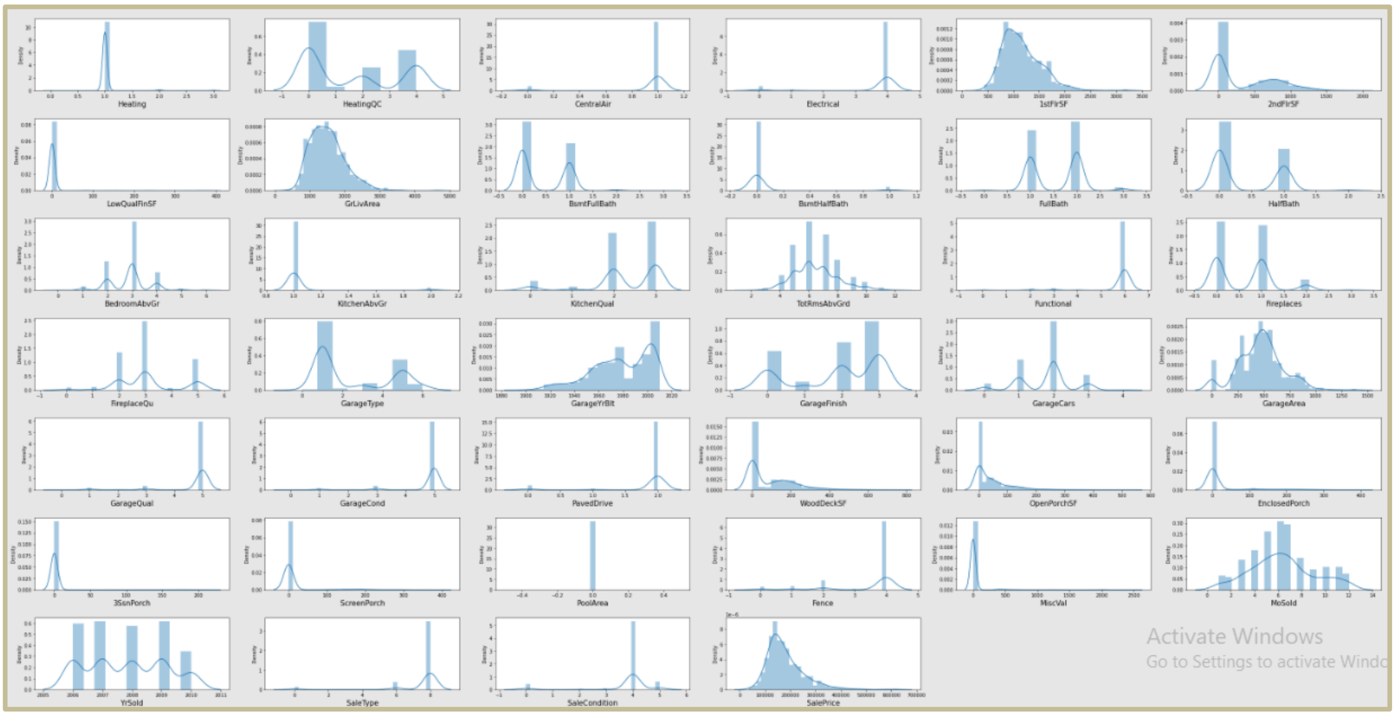
Dataloss of 4.2% with zscore.

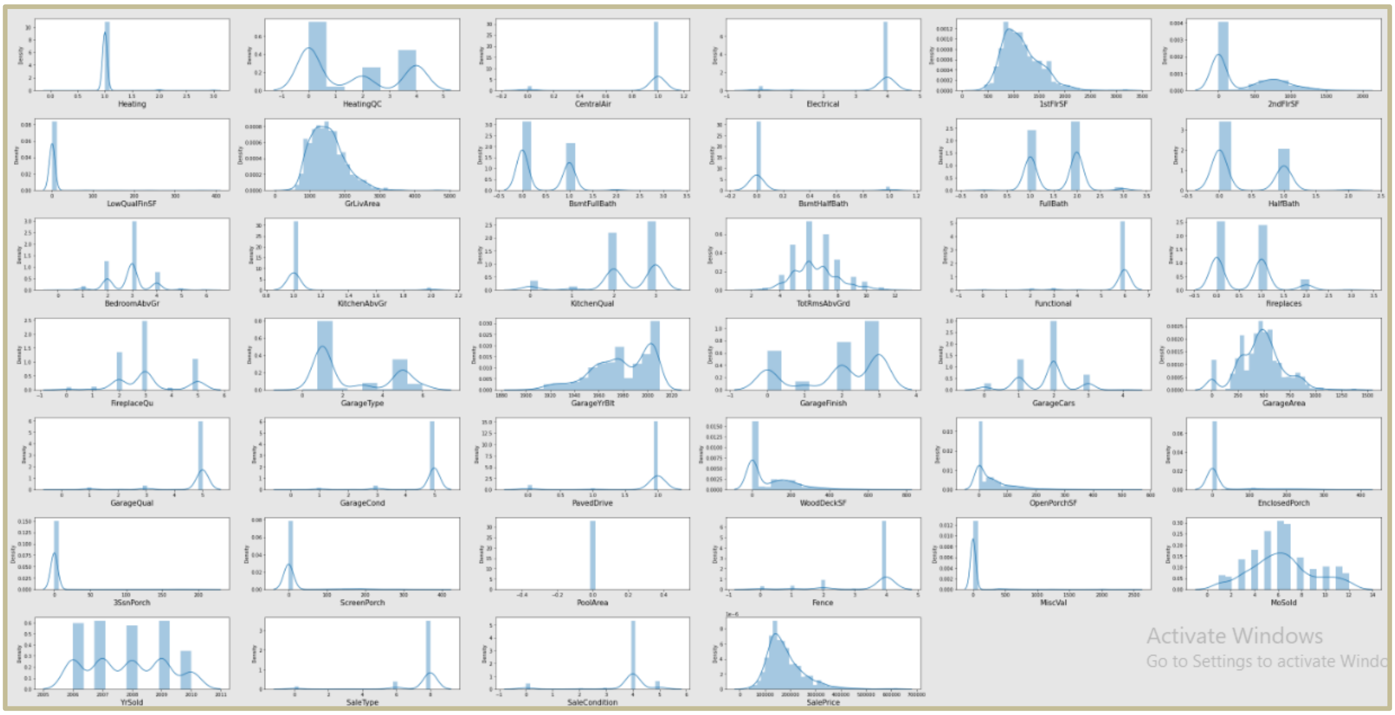
```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3 - Q1

df_1=df[~((df < (Q1 - 8 * IQR)) |(df > (Q3 + 8 * IQR))).any(axis=1)]
```

Dataloss of 86% with IQR which is very high.

We removed outliers with dataloss of 4.2% with zscore.which is less than 5% using zscore.

Distplot before Removing outliers by Zscore:





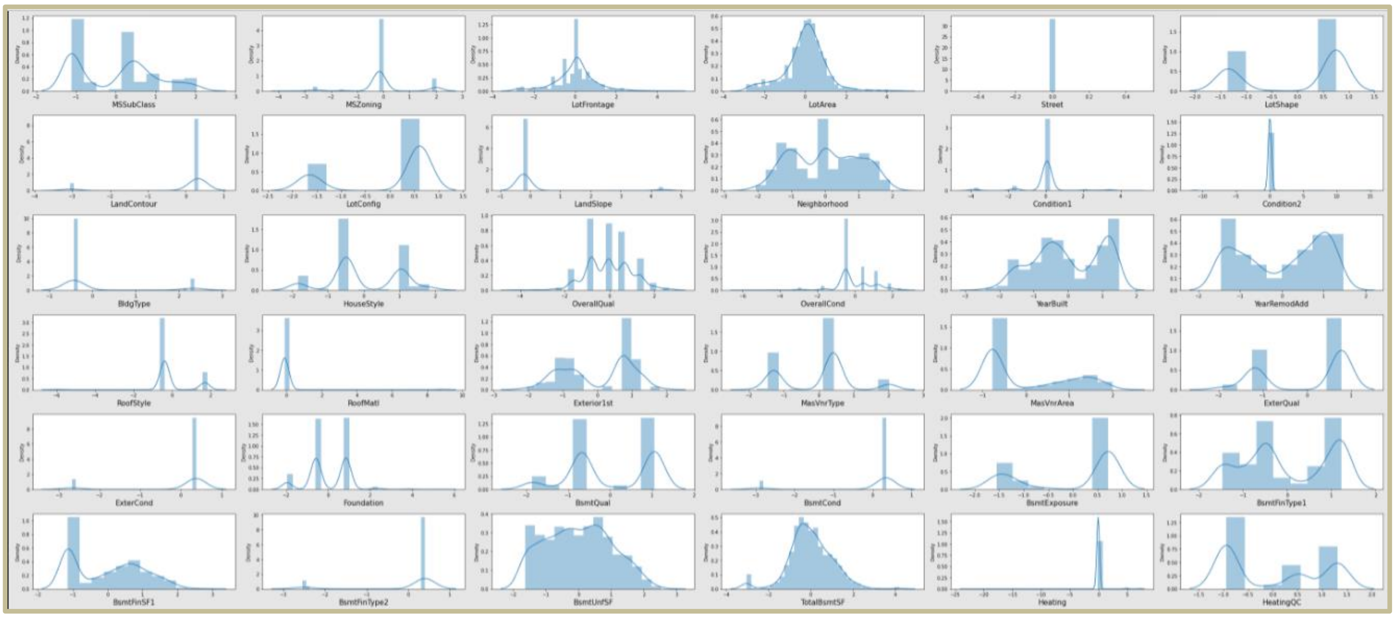
Still some feature are having skewness or outlier present.

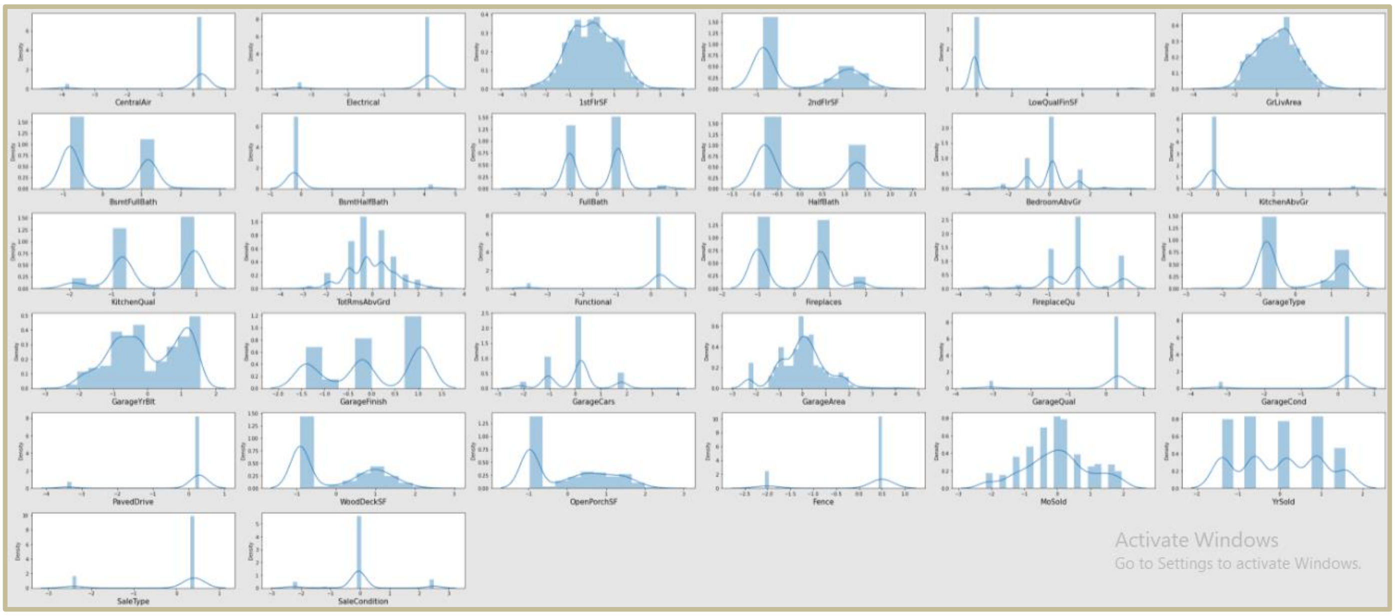
```
df1.skew()

MSSubClass        1.422060
MSZoning         -1.705609
LotFrontage       0.716133
LotArea           3.269748
Street            0.000000
LotShape         -0.622543
LandContour      -3.267184
LotConfig        -1.162836
LandSlope         5.040971
Neighborhood      0.057895
Condition1        3.116130
Condition2        7.673385
BldgType          2.314447
HouseStyle        0.285028
OverallQual       0.157192
OverallCond       0.601976
YearBuilt        -0.589899
YearRemodAdd     -0.510156
RoofStyle         1.509426
RoofMatl          9.545839
Exterior1st      -0.615743
Exterior2nd      -0.596100
MasVnrType       -0.550699
MasVnrArea        2.542936
ExterQual        -1.831179
ExterCond        -2.557072
Foundation       -0.020299
BsmtQual         -0.477206
BsmtCond         -2.846851
BsmtExposure     -0.987132
BsmtFinType1      0.096360
BsmtFinSF1        0.775786
BsmtFinType2     -3.191550
BsmtFinSF2        4.251484
BsmtUnfSF         0.929644
TotalBsmtSF       0.603531
Heating           8.613725
HeatingQC         0.461825
CentralAir       -3.649493
Electrical       -3.192577
1stFlrSF          0.993199
2ndFlrSF          0.772313
LowQualFinSF     11.747851
GrLivArea         0.916873
BsmtFullBath      0.600000
BsmtHalfBath      4.048667
FullBath          0.086564
HalfBath          0.630449
BedroomAbvGr      0.057747
KitchenAbvGr      4.746909
KitchenQual      -1.416179
TotRmsAbvGrd      0.511928
Functional       -3.968382
Fireplaces        0.650740
FireplaceQu       0.294934
GarageType        0.647939
GarageYrBlt      -0.664807
GarageFinish     -0.637047
GarageCars       -0.335474
GarageArea        0.130258
GarageQual       -3.421536
GarageCond       -3.700621
PavedDrive       -3.363156
WoodDeckSF        1.344931
OpenPorchSF       2.247901
EnclosedPorch     2.801933
3SsnPorch         8.571167
ScreenPorch       3.719086
PoolArea          0.000000
Fence            -1.981077
MiscVal           9.662815
MoSold            0.227570
YrSold            0.104185
SaleType         -3.635717
SaleCondition    -2.718697
SalePrice         1.585115
dtype: float64
```

Observation: After removing applying Zscore method data loss is 4.2%. Which is less than 5%. Still some feature are having skewness or outlier present. Now we will do **power transformation technique to treat the skewness in the data yeo-johnson**

yeo-johnson method: Removed the skewness using yeo-johnson method.

Distplot power transformation technique 'yeo-Johnson'





- ✓ The looks normal compare to the old data but still in some features skewness is present. After applying power transformation technique to treat the skewness in the data yeo-Johnson, skewness is decresed but some of columns still having lots of outlier so we will drop them.
- ✓ These are some columns MiscVal, PoolArea, ScreenPorch, 3SsnPorch, EnclosedPorch, BsmtFinSF2, Exterior2nd.
- ✓ After droping MiscVal, PoolArea, ScreenPorch, 3SsnPorch, EnclosedPorch, BsmtFinSF2, Exterior2nd these columns now our data is cleaned.

Pearson's correlation coefficient :

Pearson's correlation coefficient to check the correlation between dependent and independent features

```
data_corr = df.corr()
data_corr['SalePrice'].
sort_values(ascending = False)
```

SalePrice	1.000000
OverallQual	0.789185
GrLivArea	0.707300
GarageCars	0.628329
GarageArea	0.619000
TotalBsmtSF	0.595042
1stFlrSF	0.587642
FullBath	0.554988
TotRmsAbvGrd	0.528363
YearBuilt	0.514408
YearRemodAdd	0.507831
MasVnrArea	0.463626
Fireplaces	0.459611
GarageYrBlt	0.458007
Foundation	0.374169
BsmtFinSF1	0.362874
OpenPorchSF	0.339500
2ndFlrSF	0.330386
LotFrontage	0.323779
WoodDeckSF	0.315444
HalfBath	0.295592
LotArea	0.249499
GarageCond	0.249340
CentralAir	0.246754
Electrical	0.234621
PavedDrive	0.231707
SaleCondition	0.217687
BsmtUnfSF	0.215724
BsmtFullBath	0.212924
HouseStyle	0.205502
Neighborhood	0.198942
RoofStyle	0.192654
GarageQual	0.192392
RoofMatl	0.159865
BedroomAbvGr	0.158281
Fence	0.143922
Functional	0.118673
ExterCond	0.115167
Exterior1st	0.108451
Condition1	0.105820
PoolArea	0.103280
ScreenPorch	0.100284
Exterior2nd	0.097541
BsmtCond	0.084121
MoSold	0.072764
BsmtFinType2	0.069657
3SsnPorch	0.060119
Street	0.044753
Condition2	0.033956
LandContour	0.032836
LandSlope	0.015485
BsmtFinSF2	-0.010151
BsmtHalfBath	-0.011109
MiscVal	-0.013071
LowQualFinSF	-0.032381
YrSold	-0.045508
SaleType	-0.050851
LotConfig	-0.060452
MSSubClass	-0.060775
OverallCond	-0.065642
BldgType	-0.066028
FireplaceQu	-0.076951
MasVnrType	-0.082168
BsmtFinType1	-0.099860
Heating	-0.100021
EnclosedPorch	-0.115004
KitchenAbvGr	-0.132108
MSZoning	-0.133221
LotShape	-0.248171
BsmtExposure	-0.267635
HeatingQC	-0.406604
GarageType	-0.415370
GarageFinish	-0.424922
KitchenQual	-0.592468
BsmtQual	-0.601307
ExterQual	-0.624820
Utilities	NaN

Name: SalePrice, dtype: float64

Observation: from corelation we can observe these are features are positively corelated with Sales price ,OverallQual, GrLivArea, GarageCars, GarageArea, TotalBsmtSF, 1stFlrSF, FullBath, TotRmsAbvGrd, YearBuilt, YearRemodAdd, MasVnrArea, Fireplaces, GarageYrBlt,Foundation, BsmtFinSF1, penPorchSF, 2ndFlrSF, LotFrontage, WoodDeckSF, HalfBath, LotArea, GarageCond, CentralAir, Electrical, PavedDrive, SaleCondition,BsmtUnfSF, BsmtFullBath, HouseStyle

And negatively corelated with Sales price these features are LotShape, BsmtExposure, HeatingQC, GarageType, GarageFinish, KitchenQual, BsmtQual, ExterQual And Utilities having all NAN so we will drop Utilities

Standard scaler

Scaled the data using standard scalarizaion method to overcome with the issue of data biasness.

```
from sklearn.preprocessing import StandardScaler
scal = StandardScaler()
sc = scal.fit_transform(x)
x = pd.DataFrame(sc, columns = x.columns)
```

Observation of Exploratory Data Analysis:

- ✓ Statistical analysis like checking shape, nunique, value counts, info describe etc
- ✓ While checking the info of the datasets I found some columns with more than 80% null values, so these columns will create skewness in datasets so I decided to drop those columns.
- ✓ Then while looking into the value counts I found some columns with more than 85% zero values this also creates skewness in the model and there are chances of getting model bias so I have dropped those columns with more than 85% zero values.
- ✓ While checking for null values I found null values in most of the columns and I have used imputation method to replace those null values (mode for categorical column and mean for numerical columns).
- ✓ In Id and Utilities column the unique counts were 1168 and 1 respectively, which means all the entries in Id column are unique and ID is the identity number given for perticular asset and all the entries in Utilities column were same so these two column will not help us in model building. So I decided to drop those columns.
- ✓ And all these steps were performed to both train and test datasets separately and simultaneously.

Model Preparation

For model preparation we will Separate the features and label variables into x and y

```
x = df1.drop(columns = 'SalePrice')
y = df1['SalePrice']
```

Encoding the categorical columns using label encoder

```
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
en = OrdinalEncoder()
for i in df.columns:
    if df[i].dtypes == 'object':
        df[i] = en.fit_transform(df[i].values.reshape(-1,1))
```

Training and Testing Data

Separate data into a training set and a test set . This is a very standard approach in Machine Learning. The random_state parameter is simply a seed for the algorithm to use (if we didn't specify one, it would create different training and test sets every time we run it) Find for which state we are getting best accuracy with LinearRegression. Below we can see at 40 random state we are getting best accuracy score 89.7%.

Now with 40 random state we done train test split for taining and testing data.

```
MaX_r2_score=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.20,random_state=i)
    lr = LinearRegression()
    lr.fit(x_train,y_train)
    y_pred = lr.predict(x_test)
    r2_scores = r2_score(y_test,y_pred)
    if r2_scores>MaX_r2_score:
        MaX_r2_score = r2_scores
        random_state = i
rm_st= random_state
print("MaX R2 score corresponding to random state",random_state,"is",MaX_r2_score)
```

```
MaX R2 score corresponding to random state 40 is 0.8971348404039902
```

- ✓ Using Linear Regression we find R2 score and its corresponding random state
- ✓ Done the train_test_split data with 75 training and 25 testing data

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25,random_state=rm_st)
```

Building Machine Learning Models:

- ✓ Since SalePrice was my target and it was a continuous column so this particular problem was regression problem.
- ✓ Using Linear Regression we find R2 score and its corresponding random state
- ✓ Done the train_test_split data with 75 training and 25 testing data
- ✓ Now we will check accuracy with following Repressor algorithm and finalize one model
  - ➢ DecisionTreeRegressor()
  - ➢ Linear Regression
  - ➢ RandomForestRegressor()
  - ➢ KNeighborsRegressor()
  - ➢ AdaBoostRegressor()
  - ➢ Lasso()
  - ➢ Ridge()
  - ➢ ExtraTreesRegressor()
  - ➢ XGBRegressor()
  - ➢ GradientBoostingRegressor()

Fitting the data to various model and checking the accuracy:

```
kf = KFold(n_splits=5, random_state=rm_st, shuffle=True)
train=[]
test=[]
Mse=[]
cv=[]
for m in model:
    m.fit(x_train,y_train)
    pred_train=m.predict(x_train)
    pred_test=m.predict(x_test)
    train_score=r2_score(y_train,pred_train)
    train.append(train_score*100)
    test_score=r2_score(y_test,pred_test)
    test.append(test_score*100)
    mse = mean_squared_error(y_test,pred_test)
    Mse.append(mse)
    score=cross_val_score(m,x,y,cv=kf)
    cv.append(score.mean()*100)

Performance={'Model':['Linear Regression','DecisionTree','RandomForest','KNN','AdaBoost','GradientBoos
             'Training Score':train,'Test Score':test,'Mean Square Error':Mse,'Cross Validation Score'
Performance=pd.DataFrame(data=Performance)
Performance
```

Here with following comparision table of Training score, Test Score, Mean Square Error and Cross Validation Score

	Model	Training Score	Test Score	Mean Square Error	Cross Validation Score
0	Linear Regression	84.927534	88.445052	6.208591e+08	82.531217
1	DecisionTree	100.000000	73.010746	1.450160e+09	71.111381
2	RandomForest	97.754714	89.350274	5.722206e+08	85.830321
3	KNN	85.971443	82.320600	9.499322e+08	79.860453
4	AdaBoost	87.428574	82.025052	9.658123e+08	79.347513
5	GradientBoosting	97.038297	90.559087	5.072699e+08	86.101995
6	Lasso	84.927758	88.437130	6.212848e+08	82.523758
7	Ridge	84.927504	88.436850	6.212998e+08	82.547934
8	Extra Tree	100.000000	90.054847	5.343632e+08	85.826853
9	XGBRegressor	99.996111	88.169368	6.356719e+08	82.298194

Observation:

- Following are the result over all 9 algorithms with respect to Training Score,Test Score,Mean Square Error and the Cross Validation Score
- DecisionTree having 100% accuracy which is showing over fitting and maximum difference in test score and CV score. XGBRegressor is also somewhat showing overfitting.
- Linear Regression, Ridge and Lasso are underfitting model as training score is less than testing score.
- We can see GradientBoosting, Extra Tree and RandomForest regressor having comparative less Mean Square Error.
- GradientBoosting, Extra Tree and RandomForest regressor Also test score and Cross Validation Score difference is also less.
- So we will go for Hyper parameter tuning for GradientBoosting, Extra Tree and RandomForest regressor model and will choose best model out of them

Hyper Parameter Tunning RandomForest Regressor:

```
gcv.best_params_
```

```
{'criterion': 'mse',
 'max_depth': 13,
 'min_samples_split': 4,
 'n_estimators': 400}
```

```
Finalmod_max= RandomForestRegressor(criterion = 'mse',max_depth = 13, min_samples_split = 4, n_estimators = 400)
Finalmod_max.fit(x_train,y_train)
pred_test=Finalmod_max.predict(x_test)
R2=r2_score(y_test,pred_test)
scores=cross_val_score(Finalmod_max,x,y,cv=kf)
MSE = mean_squared_error(y_test,pred_test)
print('RandomForestRegressor Performance')
print('-----------------------------------------------------')
print('Accuracy Score', R2*100)
print('Cross Validation score',scores.mean()*100)
print('Mean Square Error',MSE)
```

```
RandomForestRegressor Performance
---------------------------------------------------
Accuracy Score 89.30143826541472
Cross Validation score 86.02295617933564
Mean Square Error 574844617.6756921
```

- ✓ **RandomForest Regressor**, after tuning the model with best parameters we can see the decreased accuracy from 89.30% to 88.87% and Cross Validation Score almost same Also Mean Square Error values has increased which means error has increased so we will not go for this model

Hyper Parameter Tunning ExtraTreesRegressor:

```
et.best_params_
```

```
{'bootstrap': True,
 'max_depth': 15,
 'min_samples_split': 4,
 'n_estimators': 200,
 'n_jobs': -2}
```

```
Finalmod_et= ExtraTreesRegressor(n_estimators=200,max_depth=15,min_samples_split=4,bootstrap='True',n_jobs=-2)
Finalmod_et.fit(x_train,y_train)
pred_test=Finalmod_et.predict(x_test)
R2=r2_score(y_test,pred_test)
scores=cross_val_score(Finalmod_et,x,y,cv=kf)
MSE = mean_squared_error(y_test,pred_test)
print('ExtraTreesRegressor Performance')
print('----------------------------------------------------')
print('Accuracy Score', R2*100)
print('Cross Validation score',scores.mean()*100)
print('Mean Square Error',MSE)
```

```
ExtraTreesRegressor Performance
----------------------------------------------------
Accuracy Score 89.28417315649365
Cross Validation score 86.0143290759634
Mean Square Error 575772289.5612291
```

- ✓ **Extra Tree Regressor**, after tuning the model with best parameters we can see the decresed accuracy from 90.05% to 89.28% and Cross Validation Score almost same Also Mean Square Error values has increased which means error has increased so we will not go for this model.

Hyper Parameter Tunning GradientBoosting Regressor:

```
gbr1.best_params_
```

```
{'learning_rate': 0.01,
 'max_depth': 4,
 'n_estimators': 2000,
 'random_state': 1,
 'subsample': 0.5}
```

```
Finalmod_gbr1= GradientBoostingRegressor(n_estimators=2000,max_depth=4,learning_rate= 0.01,random_state= 1,subsample= 0.5)
Finalmod_gbr1.fit(x_train,y_train)
pred_test=Finalmod_gbr1.predict(x_test)
R2=r2_score(y_test,pred_test)
scores=cross_val_score(Finalmod_gbr1,x,y,cv=kf)
MSE = mean_squared_error(y_test,pred_test)
print('GradientBoostingRegressor Performance')
print('----------------------------------------------------')
print('Accuracy Score', R2*100)
print('Cross Validation score',scores.mean()*100)
print('Mean Square Error',MSE)
```

```
GradientBoostingRegressor Performance
----------------------------------------------------
Accuracy Score 91.38653046870057
Cross Validation score 87.21272324159783
Mean Square Error 462810490.0843456
```

✓ Finally we selected **GradientBoosting Regressor**, after tunning the model with best parameters we can see the incresed accuracy from 90.55% to 91.39% and Cross Validation Score from 86.10% to 87.21% Also Mean Square Error values has reduced which means error has reduced.

Saving the model and predictions using saved model:

✓ Save best model using .pkl as follows.

✓ Now after saving the best model, loading my saved model and predicting the test values.

✓ Predicted the SalePrice for test dataset(25% of train dataset) using saved model of train dataset, and the predictions look good.

✓ Also Predicted the SalePrice for test dataset using saved model of train dataset.

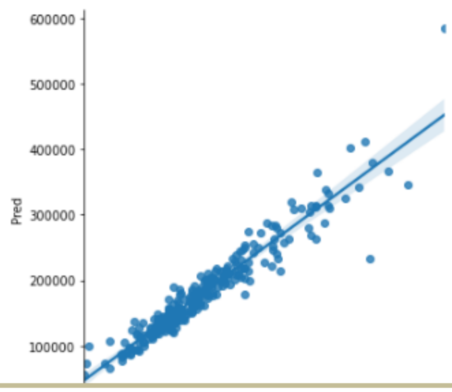
```
import pickle
filename='HusePricePredict.pkl'
pickle.dump(Finalmod_gbr1,open(filename,'wb'))
```

Best Model Saving:

✓ Predicted the SalePrice for test dataset(25% of train dataset) using saved model of train dataset, and the predictions look good.

```
res=pd.DataFrame()
res['Actual']=y_test
pred_lr=Finalmod_gbr1.predict(x_test)

data = pd.DataFrame({'Y Test':y_test , 'Pred':pred_lr},columns=['Y Test','Pred'])
sns.lmplot(x='Y Test',y='Pred',data=data,palette='rainbow')
data.head()
```

	Y Test	Pred
703	290000	263791.074864
1112	236000	217979.098092
135	123000	126480.652957
542	135000	130302.787214
166	180000	206630.782290



Conclusion::Best Model

- ✓ In this project report, we have used machine learning algorithms to predict the house prices.
- ✓ We have mentioned the step by step procedure to analyze the dataset and finding the correlation between the featuers. Thus we can select the features which are not correlated to each other and are independent in nature.
- ✓ Those feature sets were then given as an input to nine algorithms
- ✓ Hence we calculated the performance of each model using different performance metrics and compared them based on these metrics. Then we have also saved the dataframe of predicted prices of test dataset.
- ✓ To conclude, the application of machine learning in property research is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to property appraisal, and presenting an alternative approach to the valuation of housing prices.
- ✓ Future direction of research may consider incorporating additional property transaction data from a larger geographical location with more features, or analysing other property types beyond housing development.

We can observe both original and predicted attrition values are same. Conclusion is **GradientBoosting** as best model.

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_test, color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual SalePrice')
plt.ylabel('Predicted SalePrice')
plt.title('GradientBoosting Regressor')
plt.show()
```

Thank You