PFA Housing Project

Submitted by:

ROHIT KATTEWAR

ACKNOWLEDGEMENT

I would like to thank and express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project named ‘PFA Housing Project’ using Machine Learning algorithms.

Primarily, I would like to thank to the author of the paper titled: “Housing Price Prediction Project” for providing me invaluable knowledge and insights in the form of dataset. A survey of literature and the impact of housing quality on house prices in eight capital cities depicts the dynamic relationship that exist in the economics of the estate and housing markets.

Finally, I will thank my mentors, under whose guidance I learned a lot about Machine Learning, Natural Language Processing and much more.

INTRODUCTION

❖ Business Problem Framing

A United States of America 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. 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. 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.

❖ Conceptual Background of the Domain Problem

Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Hedonic Characteristics of Housing Price: A Hedonic approach is preferred for predicting the sale prices in the housing market because the market displays resilience, flexibility and spatial fixity. Housing Attributes: Studying the structural, locational, and economic attributes of housing properties are crucial in understanding their mutually inclusive relationships with the pricing.

❖ Review of the Literature

Two research papers, namely: “House Price Prediction using a Machine Learning Model: A Survey of Literature” and “The impact of housing quality on house prices in eight capital cities, Australia” were reviewed and evaluated to gain insights into all the attributes that influence the price of house. In this research, various models were built in which the house Sale Price is projected as separate and dependent variable while locational, structural and various other attributes of housing properties were treated as independent variables. Therefore, the house price is set as a target or dependency variable, while other attributes are set as independent variables to determine the main variables by identifying the correlation coefficient of each attribute.

❖ Motivation for the Problem Undertaken

There is a steady rise in house demand with every passing year, and consequently the house prices are rising every year. The problem arises when there are numerous variables such as location and property demand that influence the pricing. Therefore, buyers, sellers, developers and the real estate industry are keen to know the most important factors influencing the house price to help investors make sound decisions and help house builders set the optimal house price. There are many benefits that home buyers, property investors, and house builders can reap from the house-price model. This model aims to serve as a repository of such information and gainful insights to home buyers, property investors and house builders, that will help them determine best house prices. This model can be useful for potential buyers in deciding the characteristics of a house they want.

ANALYTICAL PROBLEM FRAMING

❖ Mathematical/Analytical Modeling of the Problem:

In order to forecast house price, predictive models such as Random Forest Regression model, Decision tree Regression Model, Support Vector Machine Regression model, Gradient Boost Regression were used to describe how the values of Sale Price depended on the independent variables of various Housing property attributes. Regression modelling techniques were used in this Problem since Sales Price data distribution is continuous in nature.

❖ Data Sources and their Formats:

The dataset was compiled by a US-based housing company named Surprise Housing. The company has collected a data set from the sale of houses in Australia. The dataset was made available in .csv file format. There are two datasets: One for training the predictive machine learning models and the second one to be used by the models for predicting the SalePrice (target variable).

```
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
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	

5 rows × 81 columns

```
df_test.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeatu
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	Na
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	Na
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	Na
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	Na
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	Na

5 rows × 80 columns

❖ Dataset Description:

- MSSubClass: Identifies the type of dwelling involved in the sale.
- MSZoning: Identifies the general zoning classification of the sale.
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access to property
- Alley: Type of alley access to property
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to various conditions
- Condition2: Proximity to various conditions (if more than one is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Evaluates the quality of the material on the exterior
- ExterCond: Evaluates the present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Evaluates the height of the basement
- BsmtCond: Evaluates the general condition of the basement
- BsmtExposure: Refers to walkout or garden level walls
- BsmtFinType1: Rating of basement finished area

- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Rating of basement finished area (if multiple types)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
- Kitchen: Kitchens above grade
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality (Assume typical
- unless deductions are warranted)
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet

- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold (MM)
- YrSold: Year Sold (YYYY)
- SaleType: Type of sale
- SaleCondition: Condition of sale

❖ Data Pre-processing Done

```
df.isnull().sum()
```

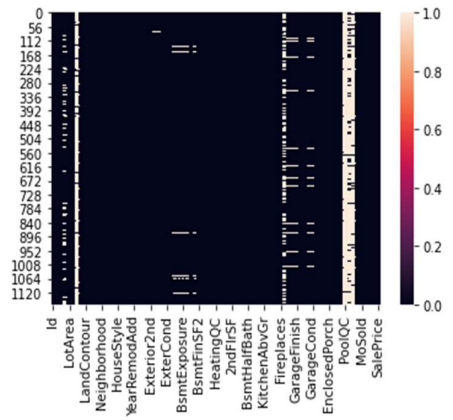
```
Id               0
MSSubClass       0
MSZoning         0
LotFrontage    214
LotArea          0
                ...
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 81, dtype: int64
```

```
df_test.isnull().sum()
```

```
Id               0
MSSubClass       0
MSZoning         0
LotFrontage     45
LotArea          0
                ..
MiscVal          0
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
Length: 80, dtype: int64
```

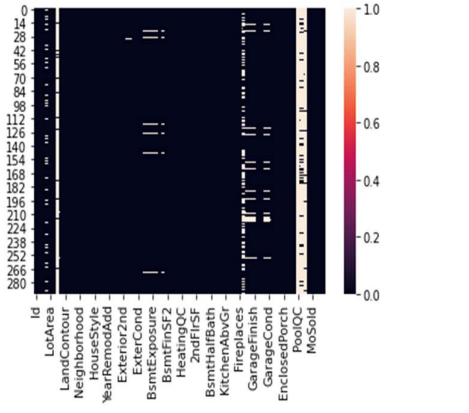
```
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```




```
sns.heatmap(df_test.isnull())
```

```
<AxesSubplot:>
```



Treating Missing Values

The columns Alley, PoolQC, MiscFeature, FireplaceQu, Fence have extremely sparse data with and very high percentage of null values. Hence, we are dropping these columns from both the datasets.

```
# Displaying the columns having null values
null_values=df.isnull().sum()
null_values=null_values[null_values>0]

# Displaying the columns having null values in percentage format
null_percent = null_values * 100 / df.shape[0]

# Concatenating the number and perecentage of missing values into one dataframe and sorting it in Descending order
pd.concat([null_values,null_percent], axis=1,keys=['Null Values', 'Null Percentage']).sort_values(by="Null Values",ascending=Fals
```

	Null Values	Null Percentage
PoolQC	1161	99.400685
MiscFeature	1124	96.232877
Alley	1091	93.407534
Fence	931	79.708904
FireplaceQu	551	47.174658
LotFrontage	214	18.321918
GarageType	64	5.479452
GarageYrBlt	64	5.479452
GarageFinish	64	5.479452
GarageQual	64	5.479452
GarageCond	64	5.479452
BsmtExposure	31	2.654110
BsmtFinType2	31	2.654110
BsmtCond	30	2.568493
BsmtFinType1	30	2.568493
BsmtQual	30	2.568493
MasVnrArea	7	0.599315
MasVnrType	7	0.599315

```
# Replacing null values in MiscFeature with No Feature
df['MiscFeature'].fillna('No feature', inplace=True)

df['Alley'].fillna('No Alley', inplace=True)
df['Fence'].fillna('No Fence', inplace=True)
df['FireplaceQu'].fillna('No Fireplace', inplace=True)
df['LotFrontage'].fillna(0, inplace=True)
```

```
# Replacing null values of numerical values with their mean of the respective columns
for column in df.columns:
    if df[column].dtypes == 'float64':
            df[column]=df[column].fillna(df[column].mean())


# Replacing null values of object datatypes their mode of the respective columns
for column in df.columns:
    if df[column].dtypes == 'object':
        df[column]=df[column].fillna(df[column].mode()[0])
```

```
df.isnull().sum().any()
```

```
False
```

Hence, we removed all the null values from the train dataset.

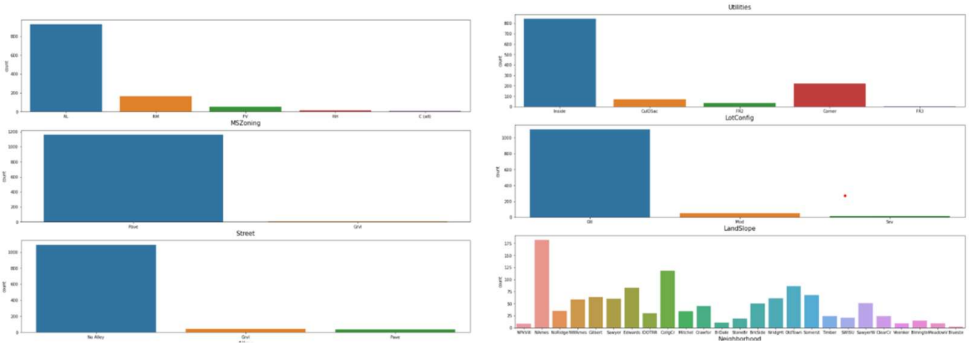
Plotting a heatmap of null values revealed that in both training and testing datasets, Columns titled: Alley, PoolQC, MiscFeature, FireplaceQu, Fence have extremely sparse data with overwhelmingly high percentage of null values.

❖ Data Inputs- Logic- Output Relationships:

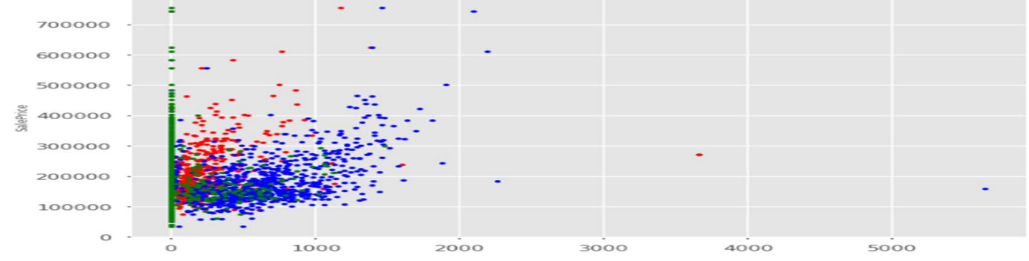
The Datasets consist mainly of object data type variables and a few float and int data type variables. The relationships between the independent variables and dependent variable were analysed. Features like Lot area, Lot Frontage, Overall Quality, Overall Condition, Basement Finishing, Total Basement Surface Area, first and 2nd Floor square feet, Garage capacity, Total rooms have a positive linear relationship, therefore increase in their values leads to increase in SalePrice. Whereas Age of Housem Remodellling age Garrage age have a linear negative relationship and therefore increase in their values leads to a decrease in SalePrice.

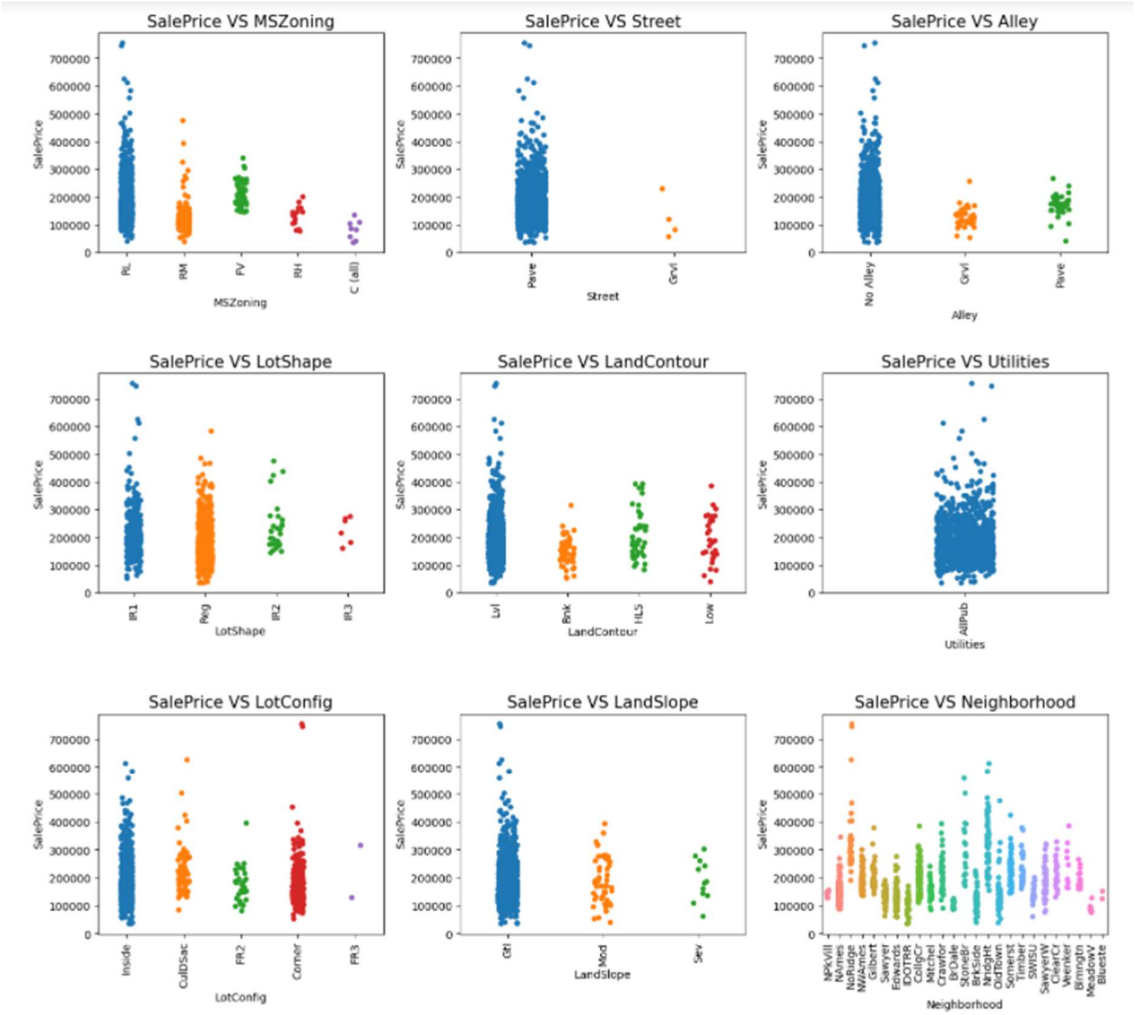
❖ Exploratory Data Analysis:

Visualizations: Bar plots, Count plots, Box plots, Swarm plots, Scatter plots were used to visualise the data of all the columns and their relationships with Target variable.




```
plt.style.use('ggplot')
ax1 = df.plot(kind='scatter', x='MasVnrArea', y='SalePrice', color='r')
ax2 = df.plot(kind='scatter', x='BsmtFinSF1', y='SalePrice', color='b', ax=ax1)
ax3 = df.plot(kind='scatter', x='BsmtFinSF2', y='SalePrice', color='g', ax=ax1)
```





From the graphs above the following observations are made:

1. Most of the houses belongs to Residential Low-Density zone and many houses from this zone are having higher prices than other zones.

2. Almost all houses are having paved streets and few are having gravel streets.

3. More number of houses are having General shape of property slightly irregular or regular. Few of them are having irregular shape.

4. SalePrice vs LandContour plot shows most of the houses are nearly Flat/Leveled.

5. Nearly all houses has kind of utilities.

6. More number of lots are inside or at corners.

7. Most of the houses are having gentle slopes.

8. Houses with sever slopes are having slightly lower prices.

9. Houses located in Northridge are having more prices compared to other locations.

10. SalePrice vs Condition1 shows most number of houses bearing normal conditions.

11. SalePrice vs Condition2 shows most houses having normal conditions and very few with other conditions.

12. Most houses are Single-family detached and are having higher sale prices than other categories.

13. SalePrice vs HouseStyle shows the houses which are having style of dweling 1-story and 2-story are having higher prices than other types.

14. Many houses are having roof style with gable and hip. and very less number of houses are having shed.

15. many houses are having roof material as standard (Composite) Shingle and houses with roof material as Standard (Composite) Shingle and Wood Shingles are having higher prices.

16. Most houses are having Vinyl Siding as 1st and 2nd covering on the house and are also having higher prices, houses with hard board and cement shilding are also having higher prices.

17. The houses with four Masonry veneer types that are, Brick Common, Brick Face, Cinder Block, Stone. Houses with Brick Common are having lower price.

18. The prices of houses are higher when material used for exterior are good or excellent.

19. Many houses are having cinder block and Poured Concrete foundation and very less houses are having wood foundations, houses with Poured Concrete foundations are having higher prices.

20. Basement quality is mostly average or good and the houses with excellent basement quality are having more prices.

21. Basement exposure is not strongly related to the sale price.

22. Most of the houses are having Heating type as Gas forced warm air furnace and Sale price of houses are higher whenever the quality of heating is excellent.

23. Most houses are having central air conditioning and are having more prices than that of houses which are without air conditioning.

24. Most of the houses with Standard Circuit Breakers & Romex electrical system and are having higher sale prices as well. Very a smaller number of houses are with Mixed type of electrical systems.

25. Most houses are with good and average kitchen quality, houses are having higher prices when kitchen quality is excellent.

26. In very rare cases fire place are prefabricated fireplace in basement and ben franklin Stove and these houses are having lower prices.

27. In most of the cases garage is attached to the house only. The houses with attached garage or Built-In the house prices are higher.

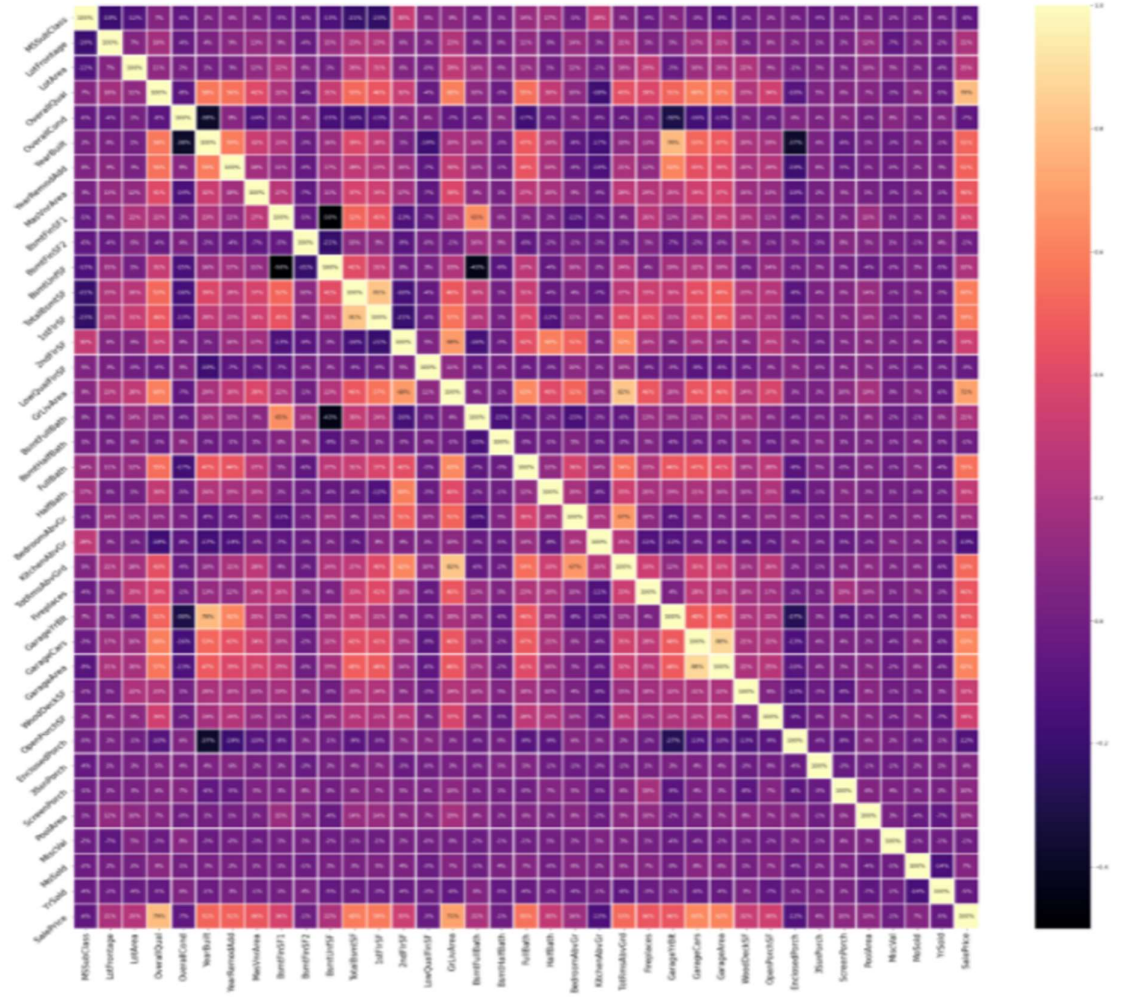
28. Many houses are having Sale type of Warranty Deed Conventional and just constructed and sold and are having higher prices.

Correlation:

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
MSSubClass	1168.0	56.767979	41.940650	20.0	20.00	50.000000	70.00	190.0
LotFrontage	1168.0	57.982021	35.471226	0.0	41.75	64.000000	79.25	313.0
LotArea	1168.0	10484.749144	8957.442311	1300.0	7621.50	9522.500000	11515.50	164660.0
OverallQual	1168.0	6.104452	1.390153	1.0	5.00	6.000000	7.00	10.0
OverallCond	1168.0	5.595890	1.124343	1.0	5.00	5.000000	6.00	9.0
YearBuilt	1168.0	1970.930651	30.145255	1875.0	1954.00	1972.000000	2000.00	2010.0
YearRemodAdd	1168.0	1984.758562	20.785185	1950.0	1966.00	1993.000000	2004.00	2010.0
MasVnrArea	1168.0	102.310078	182.047152	0.0	0.00	0.000000	160.00	1600.0
BsmtFinSF1	1168.0	444.726027	462.664785	0.0	0.00	385.500000	714.50	5644.0
BsmtFinSF2	1168.0	46.647260	163.520016	0.0	0.00	0.000000	0.00	1474.0
BsmtUnfSF	1168.0	569.721747	449.375525	0.0	216.00	474.000000	816.00	2336.0
TotalBsmtSF	1168.0	1061.095034	442.272249	0.0	799.00	1005.500000	1291.50	6110.0
1stFlrSF	1168.0	1169.860445	391.161983	334.0	892.00	1096.500000	1392.00	4692.0
2ndFlrSF	1168.0	348.826199	439.696370	0.0	0.00	0.000000	729.00	2065.0
LowQualFinSF	1168.0	6.380137	50.892844	0.0	0.00	0.000000	0.00	572.0
GrLivArea	1168.0	1525.066781	528.042957	334.0	1143.25	1468.500000	1795.00	5642.0
BsmtFullBath	1168.0	0.425514	0.521615	0.0	0.00	0.000000	1.00	3.0
BsmtHalfBath	1168.0	0.055651	0.236699	0.0	0.00	0.000000	0.00	2.0
FullBath	1168.0	1.562500	0.551882	0.0	1.00	2.000000	2.00	3.0
HalfBath	1168.0	0.388699	0.504929	0.0	0.00	0.000000	1.00	2.0
BedroomAbvGr	1168.0	2.884418	0.817229	0.0	2.00	3.000000	3.00	8.0
KitchenAbvGr	1168.0	1.045377	0.216292	0.0	1.00	1.000000	1.00	3.0
TotRmsAbvGrd	1168.0	6.542808	1.598484	2.0	5.00	6.000000	7.00	14.0
Fireplaces	1168.0	0.617295	0.650575	0.0	0.00	1.000000	1.00	3.0
GarageYrBlt	1168.0	1978.193841	24.198559	1900.0	1962.00	1978.193841	2001.00	2010.0
GarageCars	1168.0	1.776541	0.745554	0.0	1.00	2.000000	2.00	4.0
GarageArea	1168.0	476.860445	214.466769	0.0	338.00	480.000000	576.00	1418.0
WoodDeckSF	1168.0	96.206336	126.158988	0.0	0.00	0.000000	171.00	857.0
OpenPorchSF	1168.0	46.559932	66.381023	0.0	0.00	24.000000	70.00	547.0
EnclosedPorch	1168.0	23.015411	63.191089	0.0	0.00	0.000000	0.00	552.0
3SsnPorch	1168.0	3.639555	29.088867	0.0	0.00	0.000000	0.00	508.0
ScreenPorch	1168.0	15.051370	55.080816	0.0	0.00	0.000000	0.00	480.0
PoolArea	1168.0	3.448630	44.896939	0.0	0.00	0.000000	0.00	738.0
MiscVal	1168.0	47.315068	543.264432	0.0	0.00	0.000000	0.00	15500.0
MoSold	1168.0	6.344178	2.686352	1.0	5.00	6.000000	8.00	12.0
YrSold	1168.0	2007.804795	1.329738	2006.0	2007.00	2008.000000	2009.00	2010.0
SalePrice	1168.0	181477.005993	79105.586863	34900.0	130375.00	163995.000000	215000.00	755000.0

```
# Correlation using Heatmap
plt.figure(figsize=(35,35))
p = sns.heatmap(df.corr(), annot=True, lw=1.5, fmt='.0%', cmap='magma')
xlabel = p.set_xticklabels(p.get_xticklabels(),fontdict={'fontsize':16}, rotation=90)
ylabel = p.set_yticklabels(p.get_yticklabels(),fontdict={'fontsize':16}, rotation=45)
```



OverallQual, GrLiveArea, GarageCars, GarageArea, TotalBsmtSF, 1st FlrSF, Full Bath, TotRmsAbvGrd, MasVnrArea, FirePlaces have the strongest positive correlation with SalePrice while BsmtQual, ExterQual, KitchenQual, GarageFinish, House_age, Remod_age, HeatingQC, Garage_age have the strongest negative correlation with SalePrice.

❖ Model Building:

Feature Importance:

Feature Importance

```
from sklearn.feature_selection import SelectKBest, chi2
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
#concat two dataframes for better visualization
featurescores = pd.concat([dfcolumns,dfscores],axis=1)
featurescores.columns = ['Feature','Score'] #naming the dataframe columns
ficolumns = featurescores.nlargest(44,'Score')
ficolumns
```

	Feature	Score
3	LotArea	609026.012005
43	2ndFlrSF	330619.731363
33	BsmtFinSF1	263234.492234
36	BsmtUnfSF	197183.993004
45	GrLivArea	108197.738196
25	MasVnrArea	103021.612804
65	WoodDeckSF	76296.607606
42	1stFlrSF	71902.164801
37	TotalBsmtSF	71398.909813
61	GarageArea	66550.228477
66	OpenPorchSF	37539.943378
2	LotFrontage	10382.598743
0	MSSubClass	9216.158805
11	Neighborhood	1869.539194
39	HeatingQC	1276.959175
57	GarageType	858.330229
23	Exterior2nd	706.134379
32	BsmtFinType1	677.968481
6	LotShape	657.369287

2	LotFrontage	10382.598743
0	MSSubClass	9216.158805
11	Neighborhood	1869.539194
39	HeatingQC	1276.959175
57	GarageType	858.330229
23	Exterior2nd	706.134379
32	BsmtFinType1	677.968481
6	LotShape	657.369287
15	HouseStyle	628.190051
75	MoSold	586.965625
22	Exterior1st	510.454671
9	LotConfig	489.678971
55	Fireplaces	452.800185
49	HalfBath	413.043438
59	GarageFinish	401.969828
46	BsmtFullBath	361.087288
18	YearBuilt	330.799504
16	OverallQual	286.368600
60	GarageCars	245.415578
28	Foundation	234.497748
53	TotRmsAbvGrd	213.934303
58	GarageYrBlt	209.872413
19	YearRemodAdd	159.136873
48	FullBath	151.136877
24	MasVnrType	114.445650
50	BedroomAbvGr	101.506706
52	KitchenQual	80.704603
31	BsmtExposure	78.826953
29	BsmtQual	75.373559
26	ExterQual	71.056433
17	OverallCond	62.340517
56	FireplaceQu	58.596235
76	YrSold	0.484220

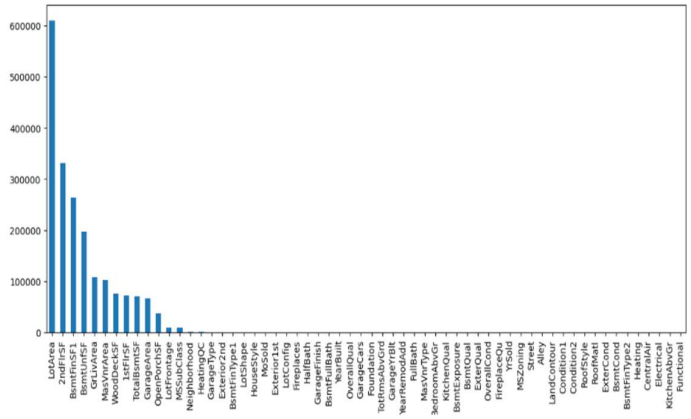
```
# Getting the best features for the model

bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)

# Filtering out the selected features for model.
filtered_features = [i[0] for i in sorted([[list(df.columns)[x], i] for x, i in enumerate(list(bestfeatures.scores_)) if i > 0.0
print(filtered_features)
```

```
['LotArea', '2ndFlrSF', 'BsmtFinSF1', 'BsmtUnfSF', 'GrLivArea', 'MasVnrArea', 'WoodDeckSF', '1stFlrSF', 'TotalBsmtSF', 'GarageA
rea', 'OpenPorchSF', 'LotFrontage', 'MSSubClass', 'Neighborhood', 'HeatingQC', 'GarageType', 'Exterior2nd', 'BsmtFinType1', 'Lo
tShape', 'HouseStyle', 'MoSold', 'Exterior1st', 'LotConfig', 'Fireplaces', 'HalfBath', 'GarageFinish', 'BsmtFullBath', 'YearBui
lt', 'OverallQual', 'GarageCars', 'Foundation', 'TotRmsAbvGrd', 'GarageYrBlt', 'YearRemodAdd', 'FullBath', 'MasVnrType', 'Bedro
omAbvGr', 'KitchenQual', 'BsmtExposure', 'BsmtQual', 'ExterQual', 'OverallCond', 'FireplaceQu', 'YrSold']
```

```
plt.figure(figsize=(14,6))
feat_importances = pd.Series(bestfeatures.scores_, index=x.columns)
feat_importances.nlargest(60).plot(kind='bar')
plt.show()
```



Identification of possible problem-solving approaches (methods) The whole problem-solving approach includes the following steps:

- **Problem Framing:** It includes understanding the problem that is whether the problem is of regression or classification. The present project is of Classification type.

- **Data Understanding:** Data understanding means having an intimate grasp of both the distributions of variables and the relationships between variables. It also includes summary statistics and data visualization.

- **Data Cleaning:** The process of identifying and repairing issues with the data is termed as data cleaning. Statistical methods are used for data cleaning. Some of them are outlier detection and imputation. There are many outliers in the present dataset, so we have imputed outliers with the upper bridge values. We can also remove outliers which may lead to loss of valuable data.
- **Data Selection:** The process of reducing the scope of data to those elements that are most useful for making predictions is called data selection.
- **Data Preparation:** It includes the data to identify the features to be selected and removed. In the present dataset, we have set "Id" column as index and included only 44 features. Before passing the data into the model, we should convert all the categorical data into numerical. So, we Label encoder and convert. In addition to this, the skewness is mitigated by "log transformation" method.
- **Model Evaluation:** Model evaluation consists of identifying input and output variables and splitting the dataset into train and test datasets. The output variable is "label" and the remaining features are input variables. In this project, we have split into 80:20 train and test respectively.
- **Model Configuration:** Hyperparameter tuning the models will get the best fit parameters of each and every model. In this project we use GridSearchCV for knowing the best fit parameters. This can be seen detail in the further sections with a snapshot of it.
- **Model Selection:** The process of selecting one method as the solution is called model selection. It includes the regression model performance metrics.

Feature Selection:

Before passing the features into the model, we select some features which is of high importance. The feature selection is shown in below figure.

Testing of Identified Approaches (Algorithms):

The Machine Learning Algorithms used in this project for training and testing to predict the prices of the houses are namely:

- Linear Regression
- Lasso Regression
- Ridge Regression
- Decision Tree Regressor

In addition to the above algorithms, ensembling techniques are also use. They are:

- Random Forest Regressor
- Gradient Boosting Regressor

Linear Regression:

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x). When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, it refers to the method as multiple linear regression.

Lasso Regression:

Lasso regression is a type linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models. This particular type of regression is well-suited for models showing high levels of multicollinearity.

Ridge Regression:

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

Decision Tree Regressor:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Random Forest Regressor:
A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Gradient Boosting Regressor:
Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Run and Evaluate selected models:

The above-mentioned algorithms have been run in the jupyter notebook and the performance metrics are found as shown in further below figures:
To check whether the model is overfitting/underfitting GridSearchCV is used and cross validated the models as shown in below figures.

```
#Importing required library

from sklearn import linear_model
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.model_selection import GridSearchCV,cross_val_score
```

```
# Getting the best Random State value with R2 score

from sklearn.model_selection import train_test_split
model_r = [LinearRegression(),Lasso(),Ridge(),DecisionTreeRegressor(),RandomForestRegressor(),GradientBoostingRegressor()]
for m in model_r:
    max_r_score=0
    for r_state in range (42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=0.20)
        m.fit(x_train,y_train)
        y_pred=m.predict(x_test)
        r2_scr=r2_score(y_test,y_pred)
        if r2_scr>max_r_score:
            max_r_score=r2_scr
            final_r_state=r_state
    print('max r2 scoring corresponding to',m,final_r_state,'is',max_r_score)
```

```
max r2 scoring corresponding to LinearRegression() 56 is 0.7821752759405631
max r2 scoring corresponding to Lasso() 56 is 0.7821590631267854
max r2 scoring corresponding to Ridge() 56 is 0.782134356778311
max r2 scoring corresponding to DecisionTreeRegressor() 62 is 0.7055819901647828
max r2 scoring corresponding to RandomForestRegressor() 62 is 0.8166562572320405
max r2 scoring corresponding to GradientBoostingRegressor() 72 is 0.8364943254648449
```

Gradient Boosting Regressor gave the best R2 score with random state of 72.

Finding Best Parameters Using GridSearchCV

```
model_params = {

    'LinearRegression':{
        'model':LinearRegression(),
        'params': {
            'fit_intercept':['True','False'],
            'n_jobs':[1,2,3,4]
        }
    },

    'Lasso':{
        'model':Lasso(),
        'params': {
            'alpha':[1.0,2.0,3.0,4.0,5.0],
            'selection':['cyclic','random']
        }
    },

     'Ridge' : {
        'model': Ridge(),
        'params': {
            'max_iter': [1,2,3,4],
            'solver':['auto','svd','sag','cholesky']
        }
    },

        'Decision Tree Regressor': {
        'model':DecisionTreeRegressor(),
        'params':{
           'criterion':['mse','mae','poisson'],
            'splitter':['best','random']
       }
    },

        'random Forest Regressor': {
        'model': RandomForestRegressor(),
        'params' : {
            'criterion':['mse','mae'],
            'n_estimators': [50,70,100,120]
        }
    },


        'Gradient Boosting Regressor':{
        'model':GradientBoostingRegressor(),
        'params':{
            'criterion':['mse','mae','friedman_mse'],
            'n_estimators': [50,70,100,120]
        }
    }
}
```

```
scores = []
from sklearn.model_selection import GridSearchCV,cross_val_score
for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5,scoring="r2", return_train_score=False)
    clf.fit(x, y)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
```

```
# Making a dataframe for the best value model, best_score & best_parameters
best_df= pd.DataFrame(scores,columns=['model','best_score','best_params'])
best_df
```

	model	best_score	best_params
0	LinearRegression	0.702125	{'fit_intercept': 'True', 'n_jobs': 1}
1	Lasso	0.702183	{'alpha': 5.0, 'selection': 'cyclic'}
2	Ridge	0.702223	{'max_iter': 1, 'solver': 'svd'}
3	Decision Tree Regressor	0.498620	{'criterion': 'mae', 'splitter': 'random'}
4	random Forest Regressor	0.751544	{'criterion': 'mae', 'n_estimators': 70}
5	Gradient Boosting Regressor	0.748493	{'criterion': 'mae', 'n_estimators': 120}

After passing the best parameters into the models we got the following results:

```
score of LinearRegression(n_jobs=1) is: 0.7312981450400451
Error:
Mean absolute error: 22895.82762845471
Mean squared error: 1098143993.4096744
Root Mean Squared Error: 33138.255738793414
r2_score: 0.7245871389963043
Cross Validation Score: 0.6961236977384623

--- -:- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ---

score of Lasso(alpha=5.0) is: 0.7312973820347134
Error:
Mean absolute error: 22887.33192964424
Mean squared error: 1098170530.5950317
Root Mean Squared Error: 33138.65613743309
r2_score: 0.7245804835101513
Cross Validation Score: 0.6963018643741454

--- -:- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ---

score of Ridge(max_iter=1) is: 0.7312973489626469
Error:
Mean absolute error: 22887.18228031719
Mean squared error: 1098118458.478414
Root Mean Squared Error: 33137.87045780724
r2_score: 0.7245935431186382
Cross Validation Score: 0.6963175740762482

--- -:- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ---


score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 30314.465811965812
Mean squared error: 2050769618.6196582
Root Mean Squared Error: 45285.423909020195
r2_score: 0.48567006575358107
Cross Validation Score: 0.4823906928658407

--- -:- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ---

score of RandomForestRegressor(criterion='mae', n_estimators=70) is: 0.9603850381370981
Error:
Mean absolute error: 20894.61800976801
Mean squared error: 934226824.1708225
Root Mean Squared Error: 30565.12431139161
r2_score: 0.7656973183704381
Cross Validation Score: 0.7454017232810541

--- -:- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ---

score of GradientBoostingRegressor(criterion='mae') is: 0.8934415888500671
Error:
Mean absolute error: 21000.72653056197
Mean squared error: 977168886.013036
Root Mean Squared Error: 31259.700670560425
r2_score: 0.7549275138818299
Cross Validation Score: 0.7362250322557888

--- -:- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ----- --:-- ---
```

From the figure, we can say that Gradient Boosting Regressor works best.

Key Metrics for success in solving problem under consideration:
Evaluating a model is a major part of building an effective machine learning model. The performance metrics of a Regression model are R2 score, mean absolute error, mean squared error, Root Mean Squared Error. Among these R2 score should be in between 0 & 1.

Train Test Split the data:

Train Test Split the data:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=62,test_size=0.20)

rfr.fit(x_train,y_train)
print('score of',rfr,'is:',rfr.score(x_train,y_train))
predrfr=rfr.predict(x_test)
print('Mean absolute error:', mean_absolute_error(y_test,predrfr))
print('Mean squared error:', mean_squared_error(y_test, predrfr))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, predrfr)))
print('r2_score:',r2_score(y_test, predrfr))
```

```
score of RandomForestRegressor(criterion='mae', n_estimators=70) is: 0.9611349760660248
Mean absolute error: 18533.018192918193
Mean squared error: 807888879.8452312
Root Mean Squared Error: 28423.386143196083
r2_score: 0.7955304745684761
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=72,test_size=0.20)

gbr.fit(x_train,y_train)
print('score of',gbr,'is:',gbr.score(x_train,y_train))
predgbr=gbr.predict(x_test)
print('Mean absolute error:', mean_absolute_error(y_test,predgbr))
print('Mean squared error:', mean_squared_error(y_test, predgbr))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, predgbr)))
print('r2_score:',r2_score(y_test, predgbr))
```

```
score of GradientBoostingRegressor(criterion='mae') is: 0.8980675052844312
Mean absolute error: 15125.516022488813
Mean squared error: 474965515.71526
Root Mean Squared Error: 21793.703579595185
r2_score: 0.8407726608897461
```

From the figure, we can say that Gradient Boosting Regressor works best. The performance metrics obtained are good.
All the data pre-processing steps are done in test dataset which is similar to train dataset and is executed in a single cell.

The predictions made on the test dataset are shown in below figure:

```
# Making predictions on test set splitted from train dataset
pred=gbr.predict(x_test)
print('predicted result:',pred)
print('actual:',y_test)
```

```
predicted result: [149190.66375986 128740.5754663   77317.74792998 158821.28058124
 140178.20479188 194499.39820127 122375.46816183 123857.01896334
 128103.27762322 163760.02571295 134995.33051082 166191.61473757
 151186.86448545 250250.67411094 126147.45552933 128505.6101041
 180908.53241732 178717.21762886 137429.80051452 176221.44756996
  87040.34600966  86657.46168693 188951.00141307 145215.99868646
 150802.96692973 263912.96873163 191965.2026099  143051.37720469
 245002.91766045 199832.46037638 157621.11992846 166512.73244234
 154826.06026731 124104.97210869 251914.8598479  181436.43705639
 271166.81063403 138926.56739401 169253.06788121 145712.32120104
 141933.88298121 177233.87654144 249410.69919975 132671.97366018
 207584.38128217 148017.60744473 182861.54562414 232569.56983253
 187565.16272342 106772.48193191 222140.29030455 193648.42955425
 133277.91570844 158132.90034702 151215.2680376  135220.89161599
 247699.55603168 122418.22504735 153476.65186972 228055.71921936
 118379.88664932 175823.80034573 148715.08192354 123876.92081645
 147768.70228341 116506.05689056 117664.36445521 218425.68126392
 191869.22584764 104445.53139045 219722.41096058 133508.23169438
 165055.61991589 200642.15951731 197376.65693104 293498.64656971
 128598.9979466  114369.20769385 256155.25913703 117467.97937327
 205136.22320214 202983.75724216 239747.55584123 171926.08448112
 128572.0095157   92870.96971426 146810.84308914 128998.26915049
 151574.2443077  145980.94342258 131253.20160758 191068.1708838
 117716.97956517 205498.09084444 154102.48141388 127700.37646207
 182268.95589186 171288.073631   143737.1563194  128073.36073922
  84053.36735908 108050.94224756 144107.74730168 246319.44188364
 248201.28289508 174754.04945025 257870.23695285 266731.1729534
 146573.08872632 117370.21839618 275087.90712283 120760.97979027
 115777.71791441 151798.39238018 143469.42988485 168020.23213255
 157370.30200683 140420.11832751  63909.26367172 105247.49010989
 262387.73691389 181569.02413328 227176.77129159 194293.10707095
 150060.52261513 145458.65445009 130030.51309577 198672.45817906
 245561.29263929 179740.51895109 182178.58012199 176643.66805887
 205513.78728102 133631.60529188 284615.59122322 104644.86775029
 152179.90715761 183257.77370519 186130.05356386 161123.70849969
 263906.10071967 121684.97028811 134737.40394393 169888.22283936
 118845.64585343 206390.42102791 172429.23590011 128813.67965475
 125533.12390086 134998.91758564 240057.75631706 208777.0672514
 117261.25380942 185532.95621151 144340.22861707 123056.87898071
```

```
# Making predictions on test dataset
gbr.predict(df_test)
```

```
array([204035.79482463, 192322.83445885, 194076.96659084, 207612.8486183 ,
       181222.3076001 , 248852.52548756, 185841.76810361, 184923.84720178,
       175399.30592687, 185841.76810361, 238421.26958738, 179230.42818029,
       221772.50912361, 195720.7765985 , 214584.36734042, 165801.23804728,
       217233.98913816, 208371.72952573, 159685.6989704 , 186480.35167786,
       205423.49752064, 227717.00240293, 155971.73760877, 229330.60121934,
       195308.20963996, 221318.28422801, 184319.77465752, 189192.67056308,
       163926.63895684, 211334.3017933 , 193712.56872866, 185841.76810361,
       215987.32327053, 214423.89981822, 238421.26958738, 231555.21266164,
       183128.44370715, 230942.39766914, 158323.07706281, 220559.40332058,
       210124.06742173, 159685.6989704 , 169756.86718726, 189262.87944829,
       211316.90512749, 209716.48981632, 226267.75125095, 223876.15032162,
       204035.79482463, 204035.79482463, 168946.31486499, 181503.54896094,
       217010.09339589, 223876.15032162, 222954.58667521, 190737.8092508 ,
       217010.09339589, 184319.77465752, 215184.73278012, 205403.82239289,
       179339.02220067, 144473.73306066, 193204.56844194, 213725.43627087,
       204035.79482463, 214425.85187269, 189262.87944829, 211628.64278177,
       208556.7892022 , 208301.52064052, 164934.41698613, 155053.81670694,
       219704.52112809, 203273.81637179, 212717.65824158, 157732.29052863,
       220865.1041458 , 216649.64263903, 183186.28245776, 187073.11093189,
       198864.89392327, 202364.13060257, 208486.58031699, 208301.52064052,
       190855.05447103, 199770.80273625, 223876.15032162, 177879.72569143,
       207612.8486183 , 160714.90685051, 208639.8740611 , 226267.75125095,
       219704.52112809, 236317.62838936, 131713.48403182, 185841.76810361,
       207386.6972841 , 161156.24120082, 177879.72569143, 211907.69166167,
       161156.24120082, 217945.67092121, 221957.56880007, 231555.21266164,
       198864.89392327, 211642.1821372 , 186480.35167786, 202458.64679113,
       208525.32371195, 217600.87993007, 188440.16109546, 165710.26319678,
       187014.20238972, 163079.5958391 , 184910.5611917 , 219704.52112809,
       188935.74508064, 163291.86999155, 224078.4001533 , 188104.02751827,
       208371.72952573, 214584.36734042, 175883.09845733, 241227.88840877,
       202364.13060257, 183719.23024134, 193204.56844194, 188255.101419  ,
       194801.30842219, 259373.08788138, 179230.42818029, 148690.33019188,
       212636.42081106, 182963.32756499, 173782.78261745, 204035.79482463,
       204035.79482463, 169756.86718726, 198864.89392327, 214238.84014176,
       191606.00387847, 176281.16112795, 195392.09495637, 222954.58667521,
       223823.23067144, 162373.94908972, 235557.44175933, 205423.49752064,
       187513.24098408, 190264.26793684, 180898.66331284, 226267.75125095,
       186480.35167786, 181313.67732904, 251002.8685466 , 185472.57364857,
```

Conclusion:

Key Findings and Conclusions of the Study:

The goal is to achieve the system which will reduce the human effort to find a house having reasonable price. The proposed system House Price Prediction model approximately tries to achieve the same one. We have managed out how to prepare a model that gives users for a best approach with future lodging value predictions. Proposed system focused on predict the house price according to the area and machine learning methods are used. The experimental results

showed that this technique that is used while developing system will give best prediction of house price.

Learning Outcomes of the Study in respect of Data Science:

Data Visualization made the project problem easy to understand easy and every feature. While doing the research of the problem, we have clearly identified and imputed the columns with missing values. All the Data pre-processing steps made the problem easier to clean the data. New treatments of outliers are learnt. I have always dropped the outliers but, in this project imputed with other values without losing data.

Five Algorithms are used, in which "Gradient Boosting Regressor" has the best performance metrics with R2 score. We couldn't reach predicting the house price with R2 score of 1 in practical is the limitation of this work.

----- --:-- ----- ----- --:-- ----- ❄ ----- --:-- ---------- --:-- -----