FLIP ROBO

"A project report On Housing - Price Prediction"

Submitted by

Himaja Ijjada

ACKNOWLEDGMENT

I express my sincere gratitude to FlipRobo Technologies for giving me the opportunity to work on the Housing Price prediction projects using machine learning algorithms. I would also like to thank FlipRobo Technologies for providing me with the requisite datasets to work with. And I would like to express my gratitude to Mr. Mohd Kashif (SME FlipRobo) and Ms. Sapna Verma (SME FlipRobo) for being of a great help in completion of the project.

I acknowledge my indebtedness to the authors of papers titled: “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” for providing me with valuable insights and knowledge of the dynamic relationships that exist in the economics of real estate and housing markets.

CONTENTS

INTRODUCTION

Business Problem Framing

Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the major contributors in the world's economy. It has a huge chunk of market and there are various companies working in this domain. Data science becomes a very important tool to solve problems in this field 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 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?

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 is crucial in understanding their mutually inclusive relationships with their pricing.

Review of 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.

From studying the papers and analyzing the research work it, is learnt that locational attributes and structural attributes are prominent factors in predicting house prices. Studies suggest that there exists a close relationship between House pricing and locational attributes such as distance from the closest shopping center, train station, position offering views of hills or shore, the neighborhood in which the property is situated etc.

Structural attributes of the house like lot size, lot shape, quality and condition of the house, garage capacity, rooms, Lot frontage, number of bedrooms, bathrooms, overall finishing of the house etc. play a big role in influencing the house price. Neighborhood qualities can be included in deciding house price. Factors like efficiency of public education, community social status, and the socio-cultural demographics improve the worth of a property.

The demand side of the housing market is also a necessary component. Although population growth is widely known as a driver in housing demand, the key issue lies in the proportion of people with abundant financial resources.

Variables representing land value such as rents and material costs also demonstrate their influence in explaining house prices, which are positively related to housing prices. Multiple regression analysis models allow to ascertain price predictions by capturing independent and dependent variable data. In using multiple regression modelling techniques, we can describe changes brought to a dependent variable with changes in the independent variables.

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 that best fits their budget and will be of tremendous benefit, especially to housing developers and researchers, to ascertain the most significant attributes to determine house prices and to acknowledge the best machine learning model to be used to conduct a study in this field.

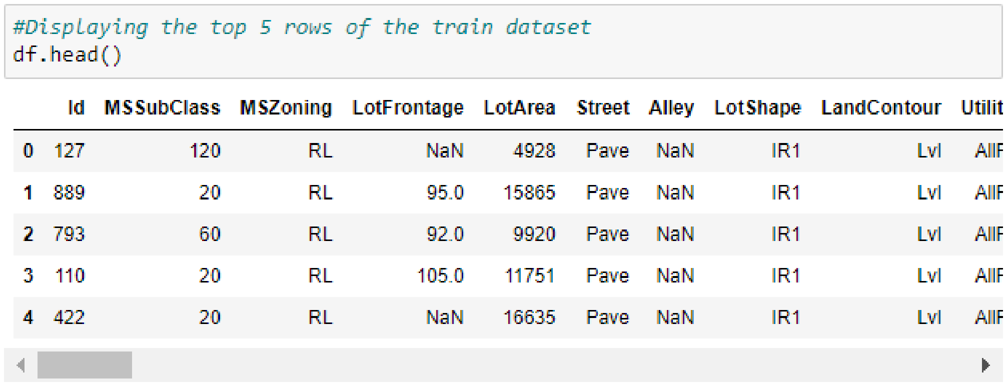
Analytical problem framing

Mathematical/ Analytical Modeling of the Problem

Various Regression analysis techniques were used to build predictive models to understand the relationships that exist between Housing sales prices and various Housing property attributes. The Regression analysis models were used to predict the Sale price value for changes in Housing property attributes. Regression modelling techniques were used in this Problem since Sales Price data distribution is continuous in nature. In order to forecast house price, predictive models such as ridge regression Model, Random Forest Regression model, Decision tree Regression Model, Support Vector Machine Regression model, Extreme Gradient Boost Regression were used to describe how the values of Sale Price depended on the independent variables of various Housing property attributes.

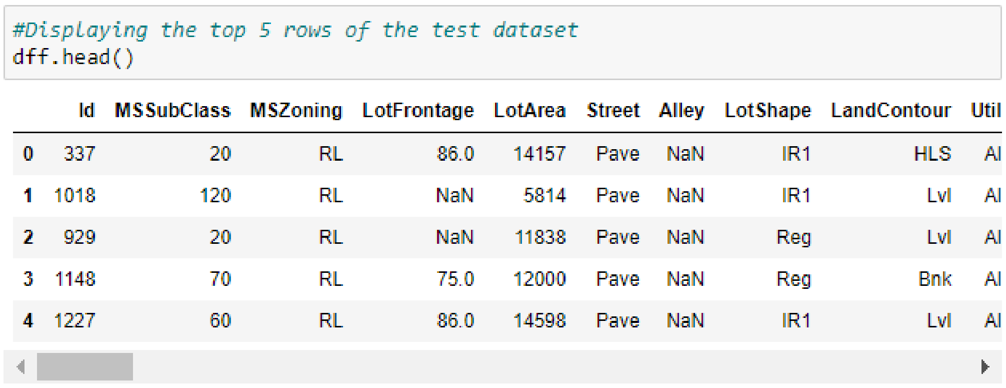
Data Sources and their formats

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 2 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).



```
#Displaying the top 5 rows of the train dataset
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilit
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllF
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllF
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllF
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllF
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllF



```
#Displaying the top 5 rows of the test dataset
dff.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Util
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	Al
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	Al
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	Al
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	Al
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	Al

Training Dataset contains 1168 entries and 81 variables, while Test Dataset contains 292 entries and 80 variables.

Dataset Description

The Independent Feature columns are as follows

- 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) livin g 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

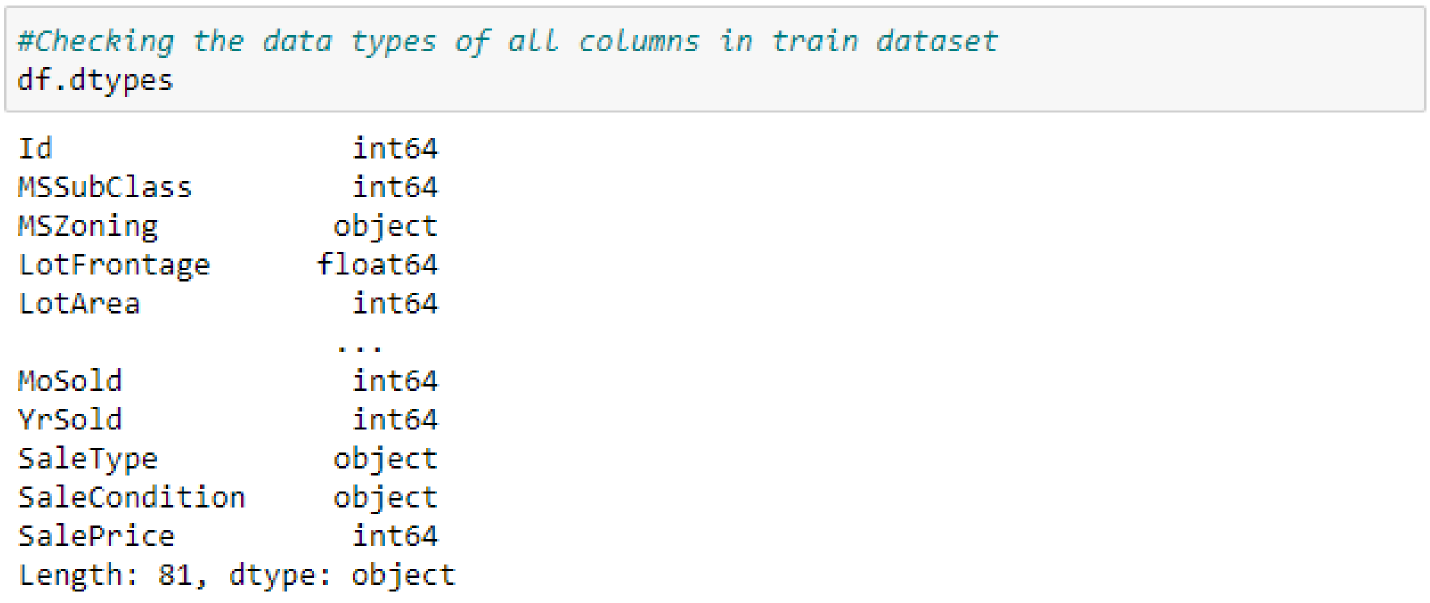
Target Column:

- SalePrice

Data Preprocessing Done

I. Train Dataset

Checking the data types of all columns in train dataset

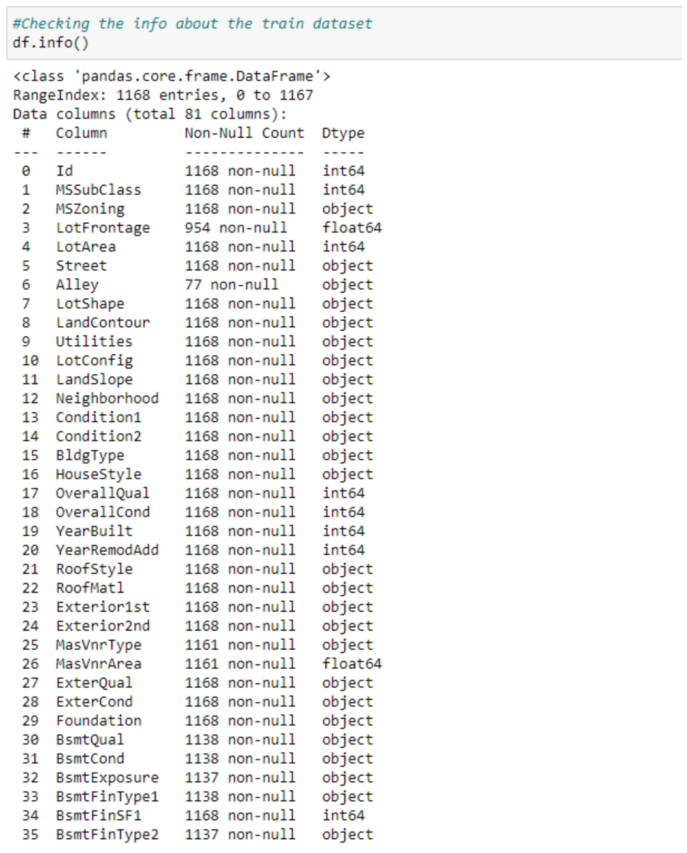


```
#Checking the data types of all columns in train dataset
df.dtypes
```

```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

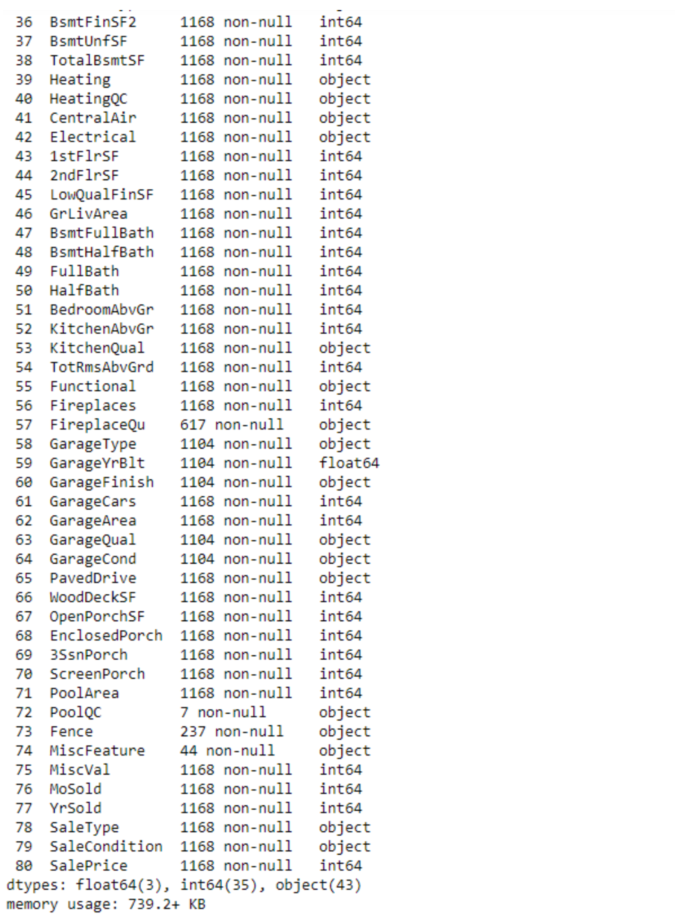
# Observations - In the train dataset we have int, object as well as flaot data types.

Checking the info about the train dataset



```
#Checking the info about the train dataset
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 
 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
```

Observations

Above is the info about train dataset from which we can observe that

- There are some missing values in the dataset, which needs to be filled using imputation techniques.
- And in Alley, PoolQC, Fence and MiscFeature has more than 80% null values so we may drop these columns in the further steps

Checking unique values of each column in train dataset

```
#Checking unique values of each column in train dataset
df.nunique()
```

```
Id               1168
MSSubClass         15
MSZoning            5
LotFrontage       106
LotArea           892
                 ...
MoSold             12
YrSold              5
SaleType            9
SaleCondition       6
SalePrice         581
Length: 81, dtype: int64
```

Observations

- In Id column the unique count is 1168 which means all the values in the column are unique and ID is the identity number given for perticular asset so this ID has no purpose in developing a model and training. Hence it can be dropped.
- In Utilities column unique value count is 1 which means all the entries are same this also has no purpose in model building so this column can also be dropped.

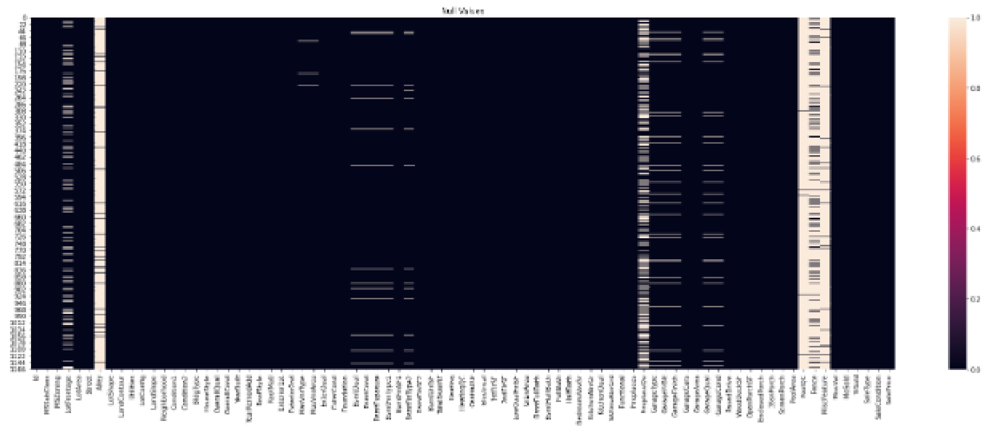
Checking null values in the train dataset

```
#Checking null values in the train dataset
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
```

Visualizing null values in train dataset

```
#Visualizing null values in train dataset
plt.figure(figsize=[30,10])
sns.heatmap(df.isnull())
plt.title("Null Values")
plt.show()
```



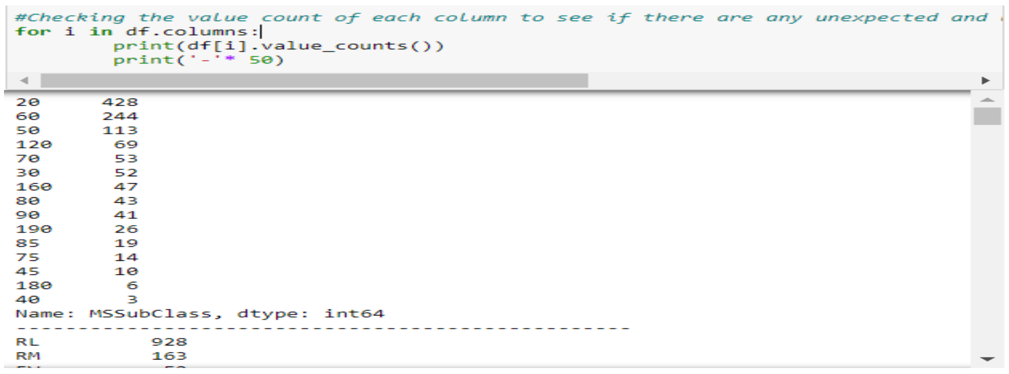
Observations

- Those columns with more than 50% (584 entries) null values can be dropped to avoid unncessary issue for model development.
- Those columns with less than 50% (584 entries) null values can be treated using imputation techniques.
- The column FireplaceQu has 551 null values, which accounts to 50% approximately can also be dropped

Dropping all the unnecessary columns from the dataset

```
#Dropping all the unnecessary columns from the dataset
df = df.drop(["Alley"],axis=1)
df = df.drop(["PoolQC"],axis=1)
df = df.drop(["Fence"],axis=1)
df = df.drop(["MiscFeature"],axis=1)
df = df.drop(["Id"],axis=1)
df = df.drop(["Utilities"],axis=1)
df = df.drop(["FireplaceQu"],axis=1)
```

Checking the value count of each column to see if there are any unexpected and unwanted entries present in the column in train dataset.



```
#Checking the value count of each column to see if there are any unexpected and
for i in df.columns:
        print(df[i].value_counts())
        print('-'* 50)
```

```
20     428
60     244
50     113
120     69
70      53
30      52
160     47
80      43
90      41
190     26
85      19
75      14
45      10
180      6
40       3
Name: MSSubClass, dtype: int64
--------------------------------------------------
RL        928
RM        163
```

Observations

- There are no unnecessary or duplicate entries in any column of the train dataset.
- There are zero values as entries in some columns which are below 60%, hence they are acceptable and reasonable.
- But there are 85% zero values as entries in the following columns
    - BsmtFinSF2
    - LowQualFinSF
    - EnclosedPorch
    - 3SsnPorch
    - ScreenPorch
    - PoolArea
    - MiscVal

So lets drop these columns.

Dropping unnecessary columns in train dataset

```
#Dropping unnecessary columns in train dataset
df.drop(columns = ['BsmtFinSF2','LowQualFinSF','EnclosedPorch','3SsnPorch','Scre
```

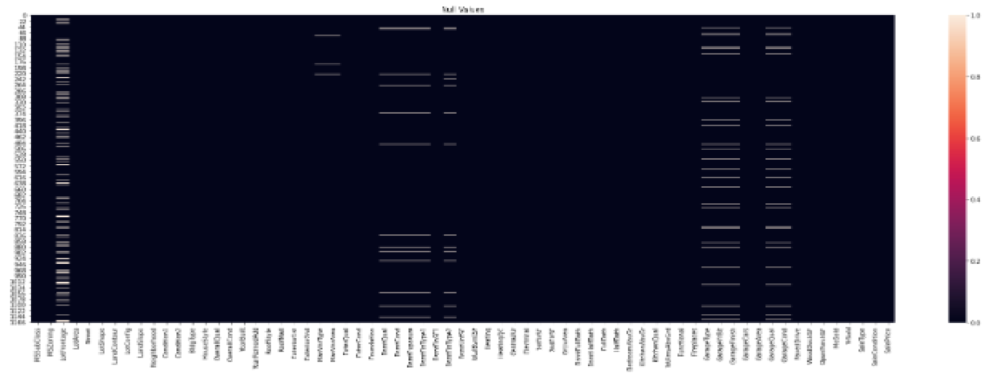
Checking for null values in the train dataset again

```
#Checking for null values in the train dataset again
df.isnull().sum()
```

```
MSSubClass         0
MSZoning           0
LotFrontage      214
LotArea            0
Street             0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 67, dtype: int64
```

Visualizing null values in train dataset

```
#Visualizeing null values in train dataset
plt.figure(figsize=[30,10])
sns.heatmap(df.isnull())
plt.title("Null Values")
plt.show()
```



Observations

We can observe that there are null values in most of the columns of the train dataset. We need to treat them using appropriate imputation techniquesand fill them with their respective values.

Creating a list of categorical and numerical datatypes in train dataset

```
#Creating a list of categorical and numerical datatypes in train dataset
df_categorical=[]
df_numerical=[]
for col in df.columns:
    if (df[col].dtype=='object'):
          df_categorical.append(col)
    else:
        df_numerical.append(col)
```

Replacing null values of categorical column with mode of that column in train dataset

```
#Replacing null values of categorical column with mode of that column in train d
catcol=df.columns.values
for i in range(0,len(catcol)):
    if df[catcol[i]].dtype == "object":
        df[catcol[i]].fillna(df[catcol[i]].mode()[0], inplace=True)
```

Replacing null values of numerical column with mean of that column in train dataset.

```
#Replacing null values of numerical column with mean of that column in train dat
numcol=df.columns.values
for i in range(0,len(numcol)):
    if df[numcol[i]].dtype != "object":
        df[numcol[i]].fillna(df[numcol[i]].mean(), inplace=True)
```

Replaced all the null values in Numerical columns and categorical coumns of the train dataset.

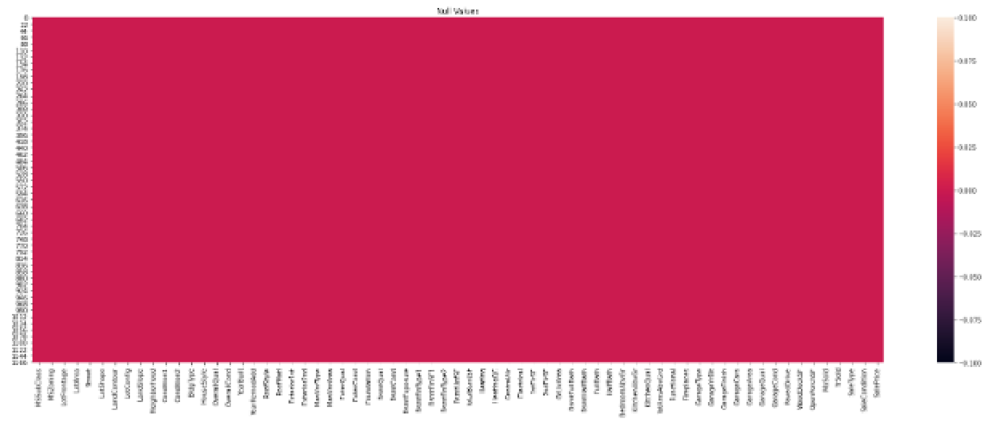
Checking for null values again in train dataset

```
#Checking for null values again in train dataset
df.isnull().sum()
```

```
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
Street           0
                ..
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 67, dtype: int64
```

Visualizing null values again after imputation in train dataset

```
#Visualizeing null values again after imputation in train dataset
plt.figure(figsize=[30,10])
sns.heatmap(df.isnull())
plt.title("Null Values")
plt.show()
```



Checking for empty observations in target column

```
#Checking for empty observations in target column
df.loc[df['SalePrice'] == " "]
```

```
MSSubClass  MSZoning  LotFrontage  LotArea  Street  LotShape  LandContour  LotConfig  LandS
```

There are no empty observations in the target column of train dataset.

Converting years column to age column in train dataset

```
# Converting years column to age column in train dataset
df['Year_SinceBuilt'] = df['YearBuilt'].max() - df['YearBuilt']
df['Year_SinceRemodAdded'] = df['YearRemodAdd'].max() - df['YearRemodAdd']
df['Year_SinceSold'] = df['YrSold'].max() - df['YrSold']
df['GarageAge'] = df['GarageYrBlt'].max() - df['GarageYrBlt']
```

Dropping old columns in train dataset

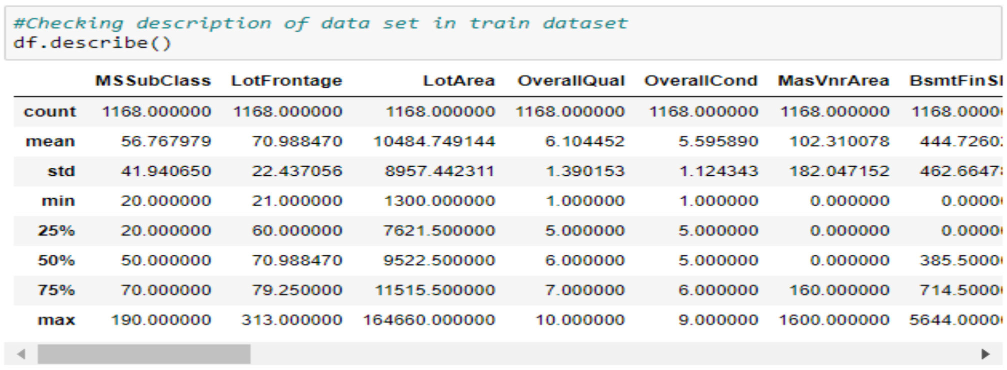
```
# Dropping old columns in train dataset
df.drop(['YearBuilt','YearRemodAdd','YrSold','GarageYrBlt'], axis=1, inplace = T
```

We have converted all the year columns to their respective age, as age helps us more than year Built in the dataset.

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 analyzed 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 Sale Price. Whereas Age of House Remodeling age Garage age have a linear negative relationship and therefore increase in their values leads to a decrease in Sale Price.

Checking description of data set in train dataset



```
#Checking description of data set in train dataset
df.describe()
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSI
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.0000
mean	56.767979	70.988470	10484.749144	6.104452	5.595890	102.310078	444.7260
std	41.940650	22.437056	8957.442311	1.390153	1.124343	182.047152	462.6647
min	20.000000	21.000000	1300.000000	1.000000	1.000000	0.000000	0.0000
25%	20.000000	60.000000	7621.500000	5.000000	5.000000	0.000000	0.0000
50%	50.000000	70.988470	9522.500000	6.000000	5.000000	0.000000	385.5000
75%	70.000000	79.250000	11515.500000	7.000000	6.000000	160.000000	714.5000
max	190.000000	313.000000	164660.000000	10.000000	9.000000	1600.000000	5644.0000

Assumptions /Observations

- Big difference between max value and 75% in SalePrice, MSSubClass, LotFrontage, LotArea, BsmtFinSF1, BsmtF inSF2, etc. indicates presence of outliers.

- A higher std than mean in columns:

    - MasVnrArea

    - BsmtFinSF1

    - BsmtFinSF2

    - WoodDeckSF

- OpenPorchSF, EnclosedPorch, 3SsnPorch etc. indicates presence of skewness.

- An Anomaly is displayed in the relationship between age of house and SalePrice. There is a general negative relationship between House age and Sale Price, ie. Increase in age leads to a decrease in SalePrice. However, houses built between 1880 and 1900 sold for the highest. The assumption made in this regard is that those houses were sold for the highest amount because of their antiquity value.

Hardware and Software Requirements and Tools Used

Hardware Used:

- Processor AMD Ryzen 9 5900HX(8 Cores 16 Logical Processors)
- Physical Memory: 16.0GB (3200MHz)
- GPU: Nvidia RTX 3060 (192 bits), 6GB DDR6 VRAM, 3840 CUDA cores.

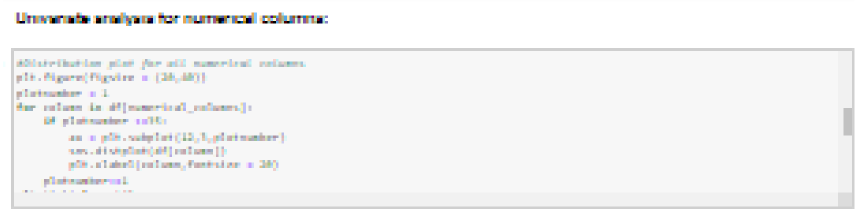
Software Used:

- Windows 10 Operating System
- Anaconda Package and Environment Manager: Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data- science packages suitable for Windows and provides a host of tools and environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects.
- Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.
- Python3: It is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used for Data Analytics and Data science projects/application. Python provides numerous libraries to deal with mathematics, statistics and scientific function.
- Python Libraries used:
  - **Pandas:** For carrying out Data Analysis, Data Manipulation, and Data Cleaning etc.
  - **Numpy:** For performing a variety of operations on the datasets.
  - **matplotlib.pyplot, Seaborn**: For visualizing Data and various relationships between Feature and Label Columns
  - **Scipy:** For performing operations on the datasets
  - **Statsmodels:** For performing statistical analysis
  - **sklearn** for Modelling Machine learning algorithms, Data Encoding, Evaluation metrics, Data Transformation, Data Scaling, Component analysis, Feature selection etc.

Model/s Development and evaluation

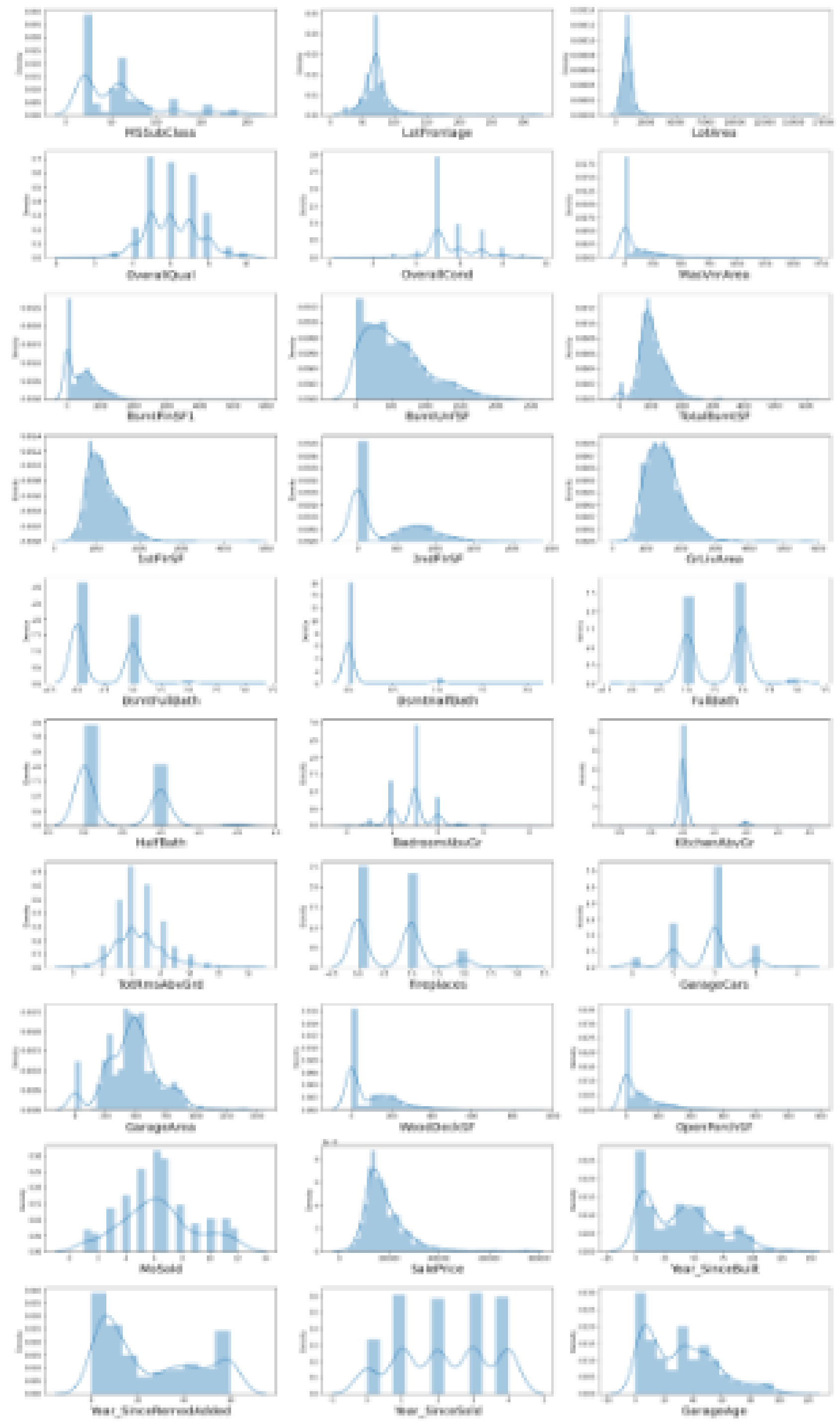
Visualization

Univariate Analysis:

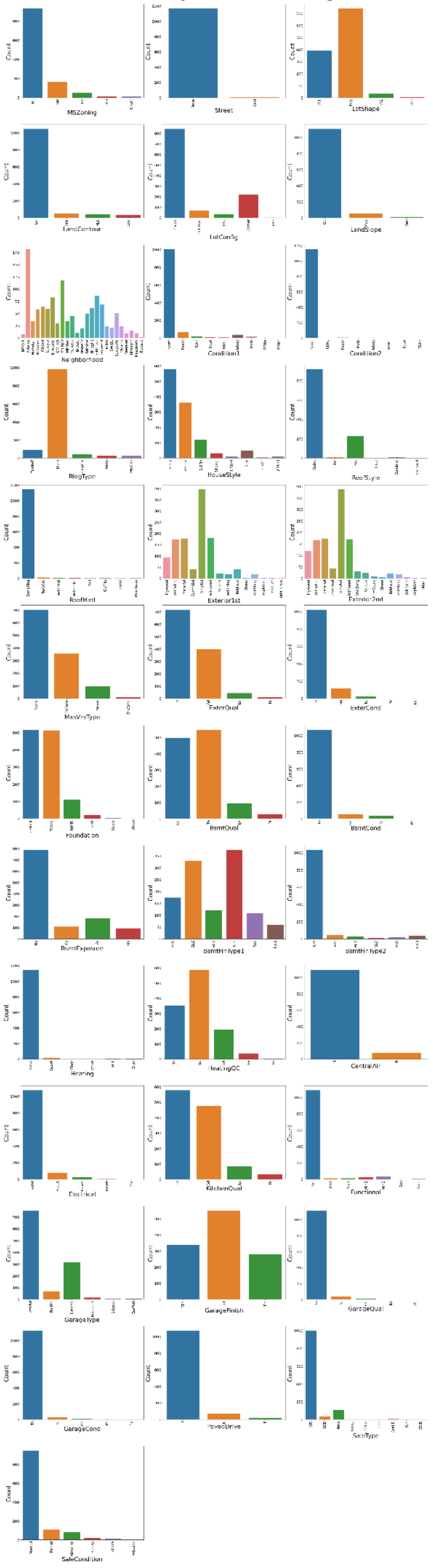


Univariate analysis for numerical columns:

```
#Distribution plot for all numerical columns
plt.figure(figsize = (20,40))
plotnumber = 1
for column in df[numerical_columns]:
    if plotnumber <=35:
        ax = plt.subplot(12,3,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize = 20)
    plotnumber+=1
```



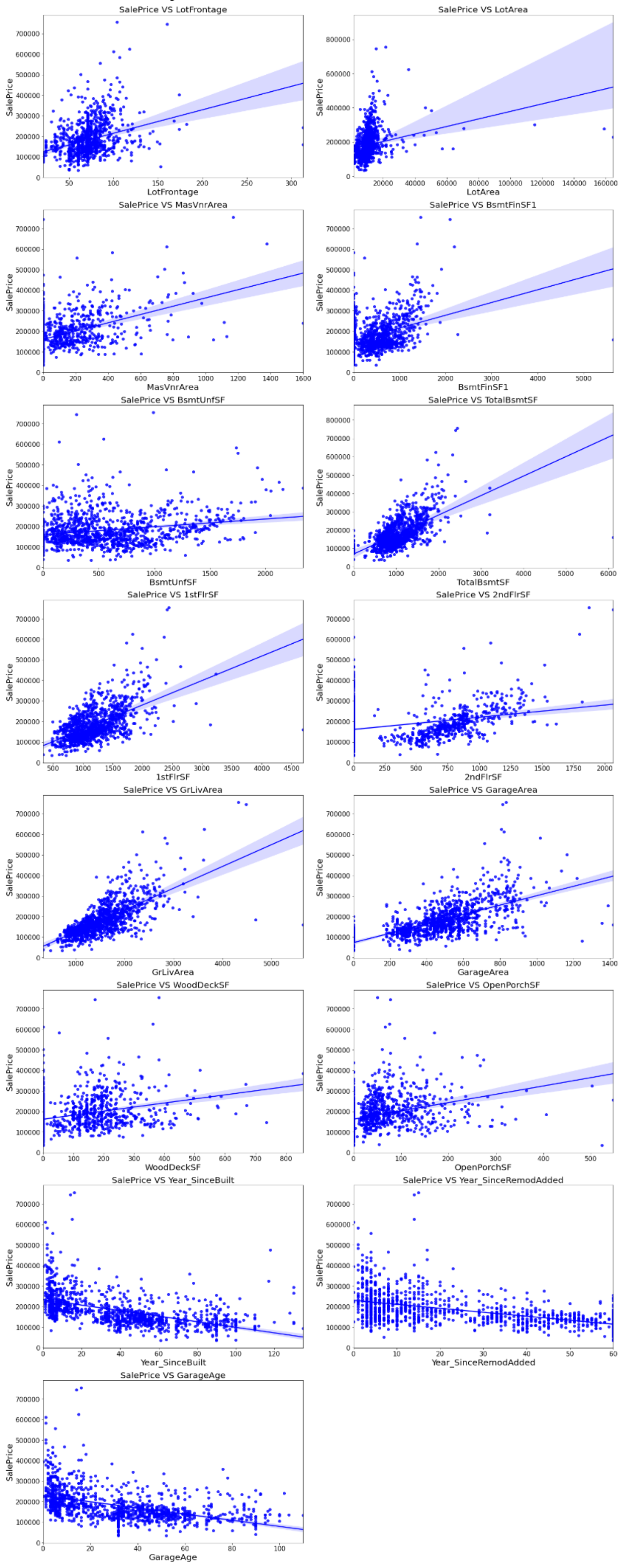
Univariate analysis for categorical columns:



Observations:

- It is found that Residential Low Density zoning has maximum count, for the feature general zoning classification of the sale(MSZoning).
- In Paved streets we can observe maximum count, for the feature Type of road access to property(Street).
- Regular shaped property has maximum count, for the feature General shape of property(LotShape).
- Near Flat/Level property has maximum count, for the feature Flatness of the property(LandContour).
- Inside lot configured property has maximum count, for the feature Lot configuration(LotConfig).
- Gentle sloped property has maximum count, for the feature Slope of property(LandSlope).
- If the property is located in North Ames then count is good compared to other locations, for the feature Physical locations within Ames city limits(Neighborhood).
- If the Proximity to various conditions-1 is normal then count is high for the feature Proximity to various conditions(Condition1).
- If the Proximity to various conditions-2 is normal then count is high for the feature Proximity to various conditions (if more than one is present)(Condition2).
- Single-family Detached dwelling has maximum count for the feature Type of dwelling(BldgType).
- One story dwelling housestyle has maximum count for the feature Style of dwelling(HouseStyle).
- For Gable roof style the count is high for the feature Type of roof(RoofStyle).
- For Standard (Composite) Shingle roof material the count is high for the feature Roof material(RoofMatl).
- For Vinyl Siding exterior-1 covering on house has maximum counts for the feature Exterior covering on house(Exterior1st).
- For Vinyl Siding exterior-2 covering on house has maximum counts for the feature Exterior covering on house (if more than one material)(Exterior2nd).
- For Masonry veneer type(MasVnrType) None has maximum count.
- For Typical/Average(TA) quality of the material on the exterior has maximum count, for the feature Evaluates the quality of the material on the exterior (ExterQual).
- For Typical/Average(TA) condition of the material on the exterior has maximum count for the feature Evaluates the present condition of the material on the exterior(ExterCond).
- For Cinder Block and Poured Contrete foundations the count is maximum for the feature Type of foundation(Foundation).
- For unfinished Rating of basement finished area-1 the count is maximum for the feature Rating of basement finished area(BsmtFinType1).
- For unfinished Rating of basement finished area-2 the count is maximum for the feature Rating of basement finished area (if multiple types)(BsmtFinType2).
- For Gas forced warm air furnace type of heating the count is maximum for the feature Type of heating(Heating).
- For Excellent Heating quality and condition the count is high for the feature Heating quality and condition(HeatingQC).
- For Central air conditioning-yes has maximum count for the feature Central air conditioning(CentralAir).
- For Standard Circuit Breakers & Romex Electrical system the count is high for the feature Electrical system(Electrical).
- For Typical/Average(TA) and good Kitchen quality the count is maximum for the feature Kitchen quality(KitchenQual).
- Typical Functionality has highest count for Home functionality (Assume typical unless deductions are warranted)(Functional).
- For good Fireplace quality the count is high for the feature Fireplace quality(FireplaceQu).
- If Garage location Attached to home then the count is high, for the feature Garage location(GarageType).
- For Unfinished Interior of the garage the count is maximum, for the feature Interior finish of the garage(GarageFinish).
- For Typical/Average(TA) Garage quality the count is high, for the feature Garage quality(GarageQual).
- For Typical/Average(TA) Garage condition the count is high, for the feature Garage condition(GarageCond).
- For Paved driveway the count is maximum, for the feature Paved driveway(PavedDrive).
- For Warranty Deed - Conventional type of sales the count is maximum, for the feature Type of sale(SaleType).
- For Normal sales condition the count is high, for the feature Condition of sale(SaleCondition).

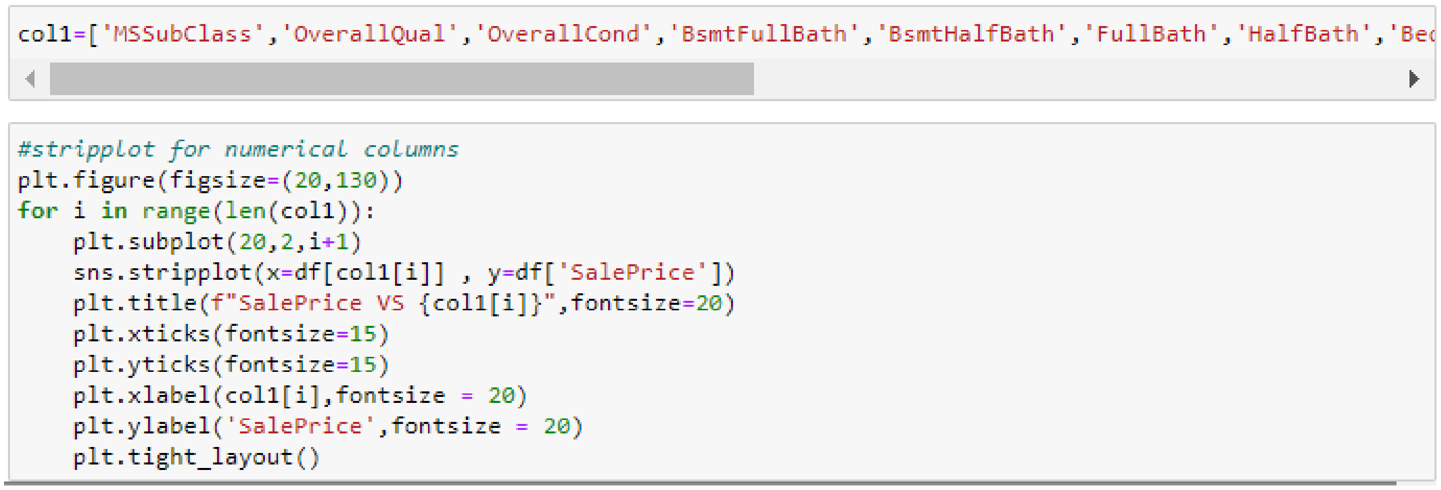
Bivariate Analysis for numerical columns:



Observations:

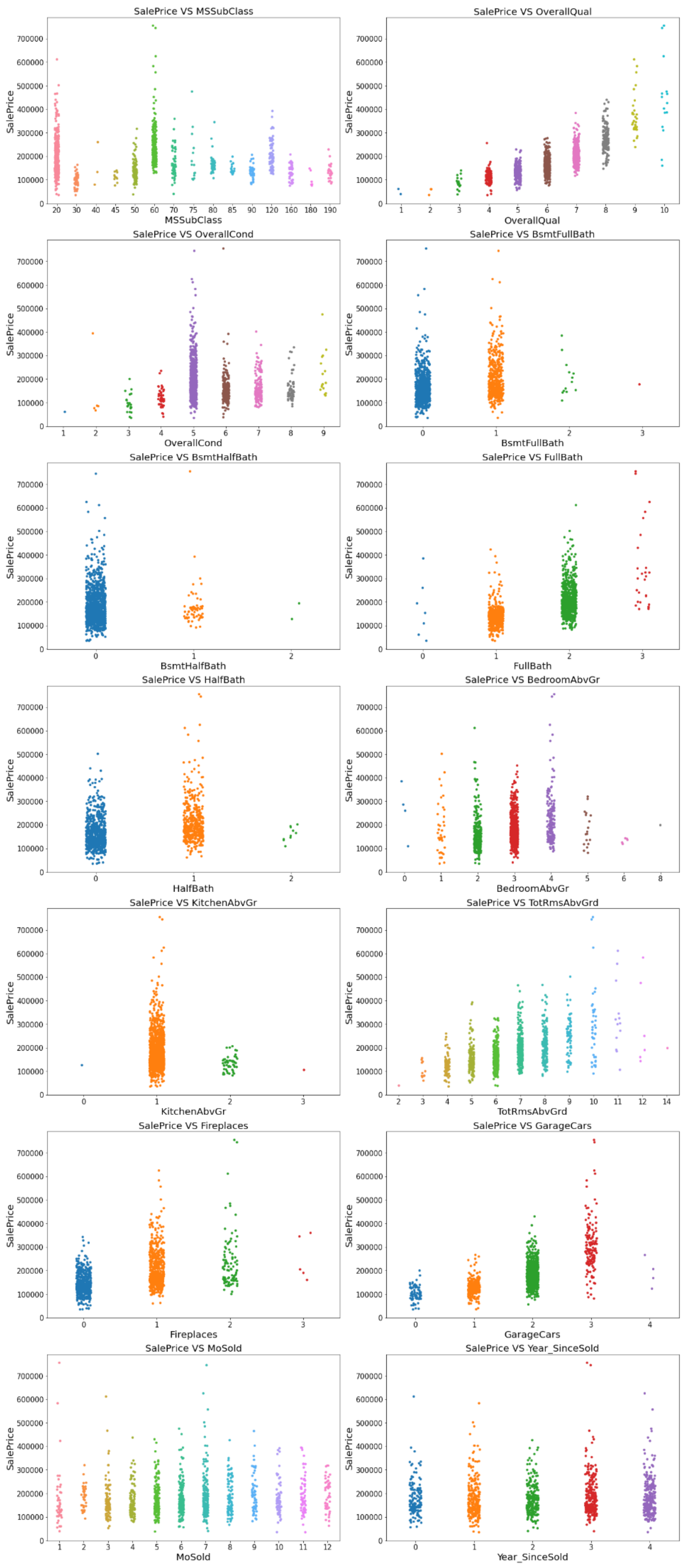
- As Linear feet of street connected to property(LotFrontage) is increseing sales is decreasing and the SalePrice is rangeing between 0-3 lakhs.
- As Lot size in square feet(LotArea) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As Masonry veneer area in square feet(MasVnrArea) is increasing sales is decreasing and saleprice is rangeing between 0-4 lakhs.
- As Type 1 finished square feet(BsmtFinSF1) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As Unfinished square feet of basement area(BsmtUnfSF) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs. There are some outliers also.
- As Total square feet of basement area(TotalBsmtSF) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As First Floor square feet(1stFlrSF) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As Second floor square feet(2ndFlrSF) is increseing sales is increasing in the range 500-1000 and the saleprice is in between 0-4 lakhs.
- As Above grade (ground) living area square feet(GrLivArea) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As Size of garage in square feet(GarageArea) is increseing sales is increseing and the saleprice is in between 0-4 lakhs.
- As Wood deck area in square feet(WoodDeckSF) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As Open porch area in square feet(OpenPorchSF) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.
- As Year_SinceBuilt is increseing sales is decreasing and the saleprice is high for newly built building and the sales price is in between 0-4 lakhs.
- As Since Remodel date (same as construction date if no remodeling or additions)(Year_SinceRemodAdded) is increseing sales is decreasing and the saleprice is in between 1-4 lakhs.
- As Since Year garage was built(GarageAge) is increseing sales is decreasing and the saleprice is in between 0-4 lakhs.

Plotting strip plot for the numerical columns



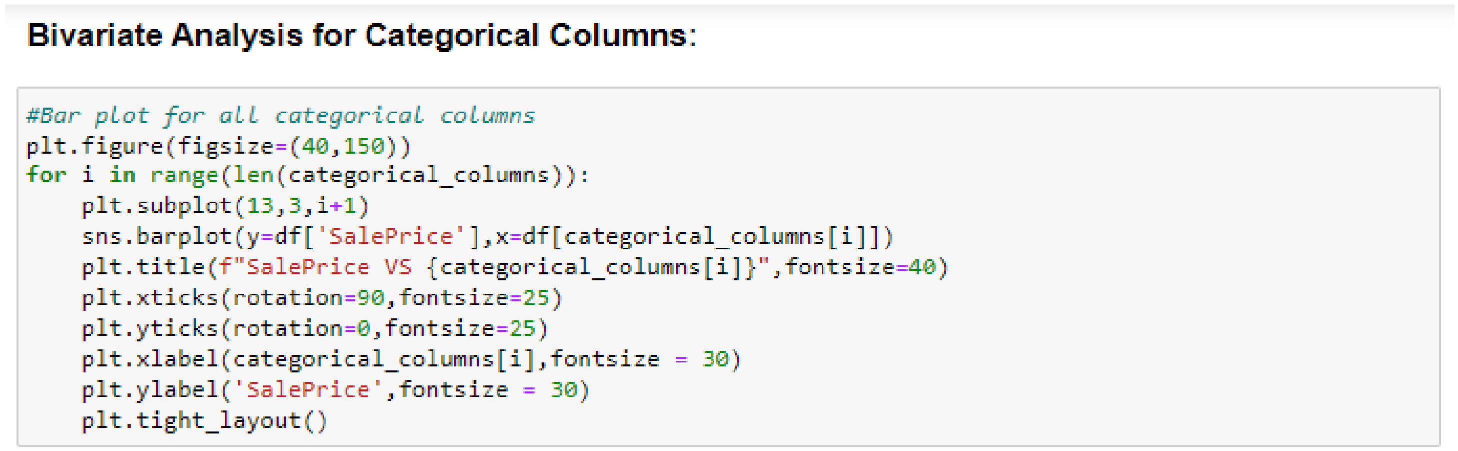
```
col1=['MSSubClass','OverallQual','OverallCond','BsmtFullBath','BsmtHalfBath','FullBath','HalfBath','Be
```

```
#stripplot for numerical columns
plt.figure(figsize=(20,130))
for i in range(len(col1)):
    plt.subplot(20,2,i+1)
    sns.stripplot(x=df[col1[i]] , y=df['SalePrice'])
    plt.title(f"SalePrice VS {col1[i]}",fontsize=20)
    plt.xticks(fontsize=15)
    plt.yticks(fontsize=15)
    plt.xlabel(col1[i],fontsize = 20)
    plt.ylabel('SalePrice',fontsize = 20)
    plt.tight_layout()
```


SalePrice VS MSSubClass
SalePrice
MSSubClass
SalePrice VS OverallQual
SalePrice
OverallQual
SalePrice VS OverallCond
SalePrice
OverallCond
SalePrice VS BsmtFullBath
SalePrice
BsmtFullBath
SalePrice VS BsmtHalfBath
SalePrice
BsmtHalfBath
SalePrice VS FullBath
SalePrice
FullBath
SalePrice VS HalfBath
SalePrice
HalfBath
SalePrice VS BedroomAbvGr
SalePrice
BedroomAbvGr
SalePrice VS KitchenAbvGr
SalePrice
KitchenAbvGr
SalePrice VS TotRmsAbvGrd
SalePrice
TotRmsAbvGrd
SalePrice VS Fireplaces
SalePrice
Fireplaces
SalePrice VS GarageCars
SalePrice
GarageCars
SalePrice VS MoSold
SalePrice
MoSold
SalePrice VS Year_SinceSold
SalePrice
Year_SinceSold

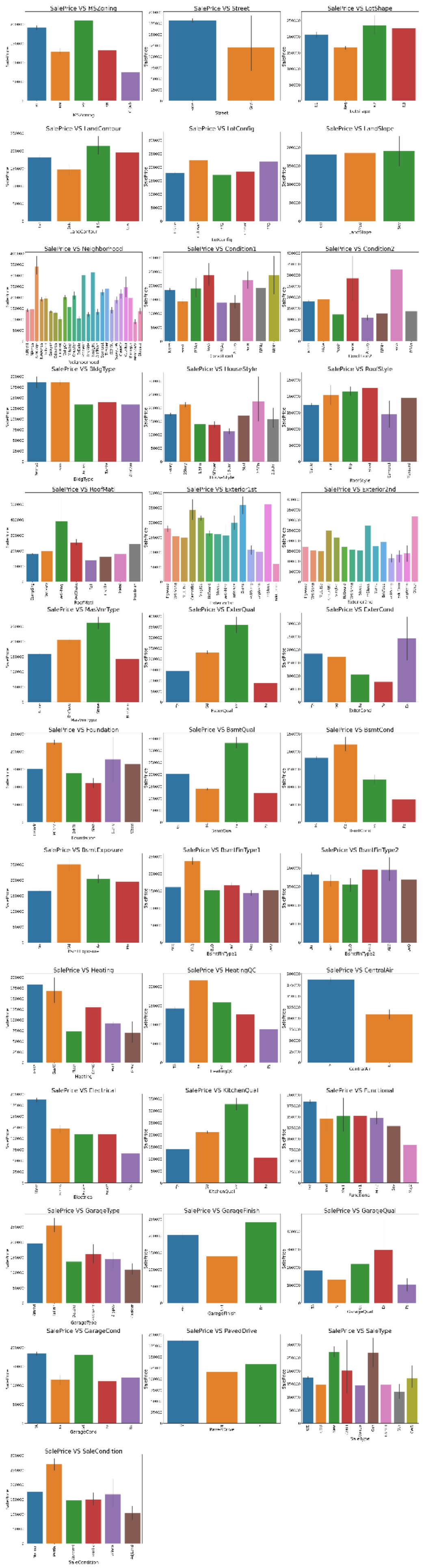
Observations:

- For 1-STORY 1946 & NEWER ALL STYLES (20) and 2-STORY 1946 & NEWER (60) types of dwelling (MSSuubClass) the sales is good and SalePrice is also high.
- As Rates the overall material and finish of the house (OverallQual) is increasing linearly sales is also increasing And SalePrice is also increasing linearly.
- For 5(Average) overall condition of the house(OverallCond) the sales is high and SalePrice is also high.
- For 0 and 1 Basement full bathrooms(BsmtFullBath) the sales as well as SalePrice is high.
- For 0 Basement half bathrooms(BsmtHalfBath) the sales as well as SalePrice is high.
- For 1 and 2 Full bathrooms above grade(FullBath) the sales as well as SalePrice is high.
- For 0 and 1 Half baths above grade(HalfBath) the sales as well as SalePrice is high.
- For 2, 3 and 4 Bedrooms above grade (does NOT include basement bedrooms)(BedroomAbvGr) the sales as well as SalePrice is high.
- For 1 Kitchens above grade(KitchenAbvGr) the sales as well as SalePrice is high.
- For 4-9 Total rooms above grade (does not include bathrooms)(TotRmsAbvGrd) the sales as well as SalePrice is high.
- For 0 and 1 Number of fireplaces(Fireplaces) the sales as well as SalePrice is high.
- For 1 and 2 Size of garage in car capacity(GarageCars) the sales is high and for 3 Size of garage in car capacity(GarageCars) the SalePrice is high.
- In between april to august for Month Sold(MoSold) the sales is good with SalePrice.
- For all the Year_SinceSold the salePrice and sales both are same.



Bivariate Analysis for Categorical Columns:

```
#Bar plot for all categorical columns
plt.figure(figsize=(40,150))
for i in range(len(categorical_columns)):
    plt.subplot(13,3,i+1)
    sns.barplot(y=df['SalePrice'],x=df[categorical_columns[i]])
    plt.title(f"SalePrice VS {categorical_columns[i]}",fontsize=40)
    plt.xticks(rotation=90,fontsize=25)
    plt.yticks(rotation=0,fontsize=25)
    plt.xlabel(categorical_columns[i],fontsize = 30)
    plt.ylabel('SalePrice',fontsize = 30)
    plt.tight_layout()
```


SalePrice VS MSZoning
SalePrice VS Street
SalePrice VS LotShape
SalePrice VS LandContour
SalePrice VS LotConfig
SalePrice VS LandSlope
SalePrice VS Neighborhood
SalePrice VS Condition1
SalePrice VS Condition2
SalePrice VS BldgType
SalePrice VS HouseStyle
SalePrice VS RoofStyle
SalePrice VS RoofMatl
SalePrice VS Exterior1st
SalePrice VS Exterior2nd
SalePrice VS MasVnrType
SalePrice VS ExterQual
SalePrice VS ExterCond
SalePrice VS Foundation
SalePrice VS BsmtQual
SalePrice VS BsmtCond
SalePrice VS BsmtExposure
SalePrice VS BsmtFinType1
SalePrice VS BsmtFinType2
SalePrice VS Heating
SalePrice VS HeatingQC
SalePrice VS CentralAir
SalePrice VS Electrical
SalePrice VS KitchenQual
SalePrice VS Functional
SalePrice VS GarageType
SalePrice VS GarageFinish
SalePrice VS GarageQual
SalePrice VS GarageCond
SalePrice VS PavedDrive
SalePrice VS SaleType
SalePrice VS SaleCondition

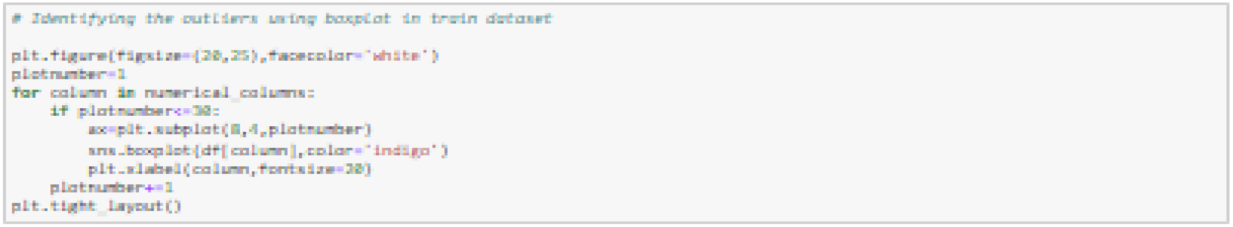
Observations:

- For Floating Village Residential(FV) and Residential Low Density(RL) zoning classification of the sale(MSZoning) the saleprice is high.
- For paved type of road access to property(Street) the SalePrice is high.
- For Slightly irregular(IR1), Moderately Irregular(IR2) and Irregular(IR3) shape of property(LotShape) the SalePrice is high.
- For Hillside - Significant slope from side to side(HLS) Flatness of the property(LandContour) the SalePrice is High.
- For Cul-de-sac(CulDSac) Lot configuration(LotConfig) the SalePrice is High.
- For all types of Slope of property(LandSlope) i.e.,Gentle slope(Gtl), Moderate Slope(Mod) and Severe Slope(Sev) the SalePrice is High.
- For Northridge(NoRidge) locations within Ames city limits(Neighborhood) the SalePrice is High.
- For Within 200' of North-South Railroad(RRNn), Adjacent to postive off-site feature(PosA) and Near positive off-site feature--park, greenbelt, etc.(PosN) Proximity to various conditions(Condition1) has the maximum SalePrice.
- For Adjacent to postive off-site feature(PosA) and Near positive off-site feature--park, greenbelt, etc.(PosN) Proximity to various conditions (if more than one is present)(Condition2) has maximum SalePrice.
- For Single-family Detached(1Fam) and Townhouse End Unit(TwnhsE) type of dwelling(BldgType) the SalePrice is high.
- For 2Story and Two and one-half story: 2nd level finished(2.5Fin) Style of dwelling(HouseStyle) the SalePrice is high.
- For Shed Type of roof(RoofStyle) the SalePrice is high.
- For Wood Shingles(WdShngl) Roof material(RoofMat1) the SalePrice is high.
- For Cement Board(CemntBd), Imitation Stucco(ImStucc) and Stone type of Exterior covering on house(Exterior1st) the SalePrice is high.
- For Cement Board(CemntBd), Imitation Stucco(ImStucc) and other Exterior covering on house (if more than one material)(Exterior2) has maximum SalePrice.
- For Stone Masonry veneer type(MasvnrType) the SalePrice is high.
- For Excellent(Ex) quality of the material on the exterior(ExterQual) the SalePrice is high.
- For Excellent(Ex) present condition of the material on the exterior(ExterCond) the SalePrice is high.
- For Poured Contrete(PConc) Type of foundation(Foundation) the SalePrice is high.
- For Excellent(100+ inches)(Ex) height of the basement(BsmtQual) the SalePrice is high.
- For Good(Gd) general condition of the basement(BsmtCond) the SalePrice is high.
- For Good Exposure(Gd) of walkout or garden level walls(BsmtExposure) has maximum SalePrice.
- For Good Living Quarters(GLQ) of basement finished area(BsmtFinType1) has maximum SalePrice.
- For Good Living Quarters(GLQ) and Average Living Quarters(ALQ) of basement finished area (if multiple types)(BsmtFinType2) has maximum SalePrice.
- For Gas forced warm air furnace(GasA) and Gas hot water or steam heat(GasW) Type of heating(Heating) has high SalePrice.
- For Excellent(Ex) Heating quality and condition(HeatingQC) the SalePriceis high.
- For building having Central air conditioning(CentralAir) the SalePrice is high.
- For Standard Circuit Breakers & Romex(Sbrkr) of Electrical system(Electrical) the SalePrice is Maximum.
- For Excellent(Ex) Kitchen quality(KitchenQual) the SalePrice is high.
- For Typical Functionality(Typ) type of Home functionality (Assume typical unless deductions are warranted)(Functional) the SalePrice is high.
- For Excellent - Exceptional Masonry Fireplace(Ex) of Fireplace quality(FireplaceQual) has highest SalePrice.
- For Built-In (Garage part of house - typically has room above garage)(BuiltIn) Garage location(GarageType) the SalePrice is maximum.
- For Completely finished(Fin) Interior of the garage(GarageFinish) the SalePrice is high.
- For Excellent(Ex) Garage quality(GarageQual) the SalePrice is high.
- For Typical/Average(TA) and Good(Gd) Garage condition(GarageCond) the SalePrice is high.
- For having Paved driveway(PavedDrive) the SalePriceis high.
- For Home just constructed and sold(New) and Contract 15% Down payment regular terms(Con) of type of sale(SaleType) has highest SalePrice.

- For Home was not completed when last assessed (associated with New Homes)(Partial) Condition of sale(SalesCondition) the SalePrice is maximum.

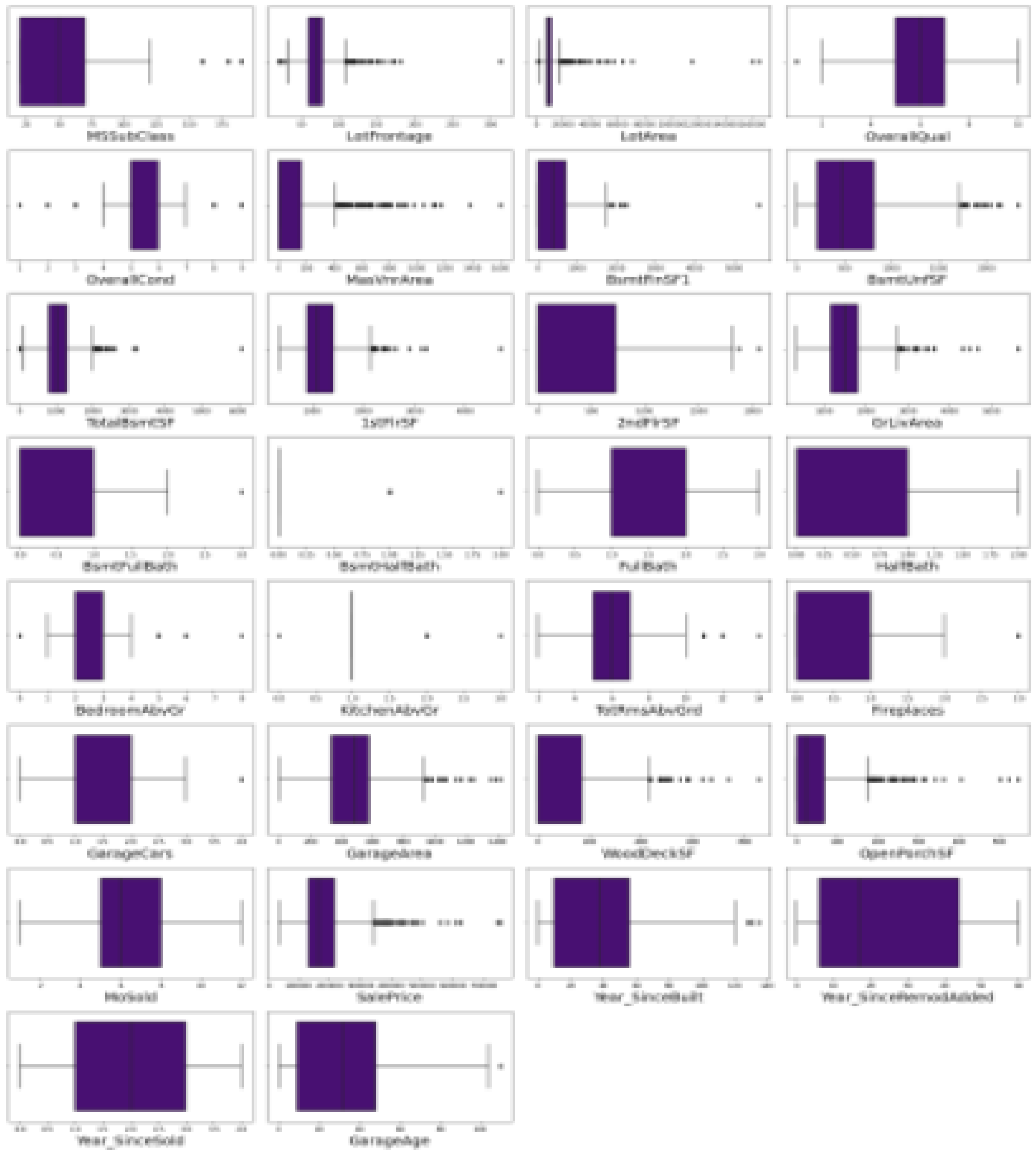
Identification of possible problem-solving approaches (methods)

Checking for outliers:



```
# Identifying the outliers using boxplot in train dataset

plt.figure(figsize=(20,25),facecolor='white')
plotnumber=1
for column in numerical_columns:
    if plotnumber<=30:
        ax=plt.subplot(8,4,plotnumber)
        sns.boxplot(df[column],color='indigo')
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.tight_layout()
```



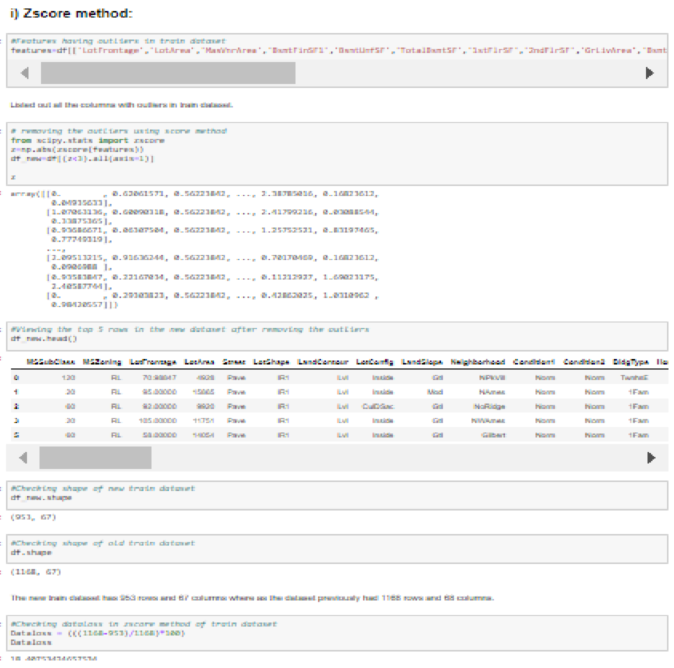
Observations

The dataset has outliers present in the following columns -

- MSSubClass
- LotFrontage
- LotArea
- OverallQual
- OverallCond
- MasVnrArea
- BsmtFinSF1
- BsmtUnfSF
- TotalBsmtSF
- 1stFlrSF
- 2ndFlrSF
- GrLivArea
- BsmtFullBath
- BsmtHalfBath
- BedroomAbvGr
- KitchenAbvGr
- TotRmsAbvGrd
- Fireplaces
- GarageCars
- GarageArea
- WoodDeckSF
- OpenPorchSF
- Year_SinceBuilt
- GarageAge
- SalePrice

Since SalePrice is the target we need not remove outliers from this column.And some of the columns like MSSubClass, OverallQual and OverallCond are categorical so we need not remove outliers in those categorical columns.

Removing Outliers in train dataset:



i) Zscore method:

```
#Features having outliers in train dataset
features=df[['LotFrontage','LotArea','MasVnrArea','BsmtFinSF1','BsmtUnfSF','TotalBsmtSF','1stFlrSF','2ndFlrSF','GrLivArea','Bsmt
```

Listed out all the columns with outliers in train dataset.

```
# removing the outliers using zcore method
from scipy.stats import zscore
z=np.abs(zscore(features))
df_new=df[(z<3).all(axis=1)]

z
```

```
array([[0.        , 0.63061571, 0.56221842, ..., 2.38785016, 0.16823612,
        0.04915633],
       [1.07063136, 0.60090318, 0.56221842, ..., 2.41799216, 0.03088544,
        0.33875365],
       [0.93686671, 0.06307504, 0.56221842, ..., 1.25752531, 0.83197465,
        0.77749319],
       ...,
       [2.09511215, 0.91636244, 0.56221842, ..., 0.70170469, 0.16823612,
        0.0986988 ],
       [0.93583847, 0.22167034, 0.56221842, ..., 0.11312937, 1.69823175,
        2.40587744],
       [0.        , 0.29103823, 0.56221842, ..., 0.42862025, 1.0310962 ,
        0.98420557]])
```

```
#Viewing the top 5 rows in the new dataset after removing the outliers
df_new.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType
0	120	RL	70.98847	4928	Pave	IR1	Lvl	Inside	Gtl	NPkVill	Norm	Norm	TwnhsE
1	20	RL	95.00000	15865	Pave	IR1	Lvl	Inside	Mod	NAmes	Norm	Norm	1Fam
2	60	RL	92.00000	9920	Pave	IR1	Lvl	CulDSac	Gtl	NoRidge	Norm	Norm	1Fam
3	20	RL	105.00000	11751	Pave	IR1	Lvl	Inside	Gtl	NWAmes	Norm	Norm	1Fam
5	60	RL	58.00000	14054	Pave	IR1	Lvl	Inside	Gtl	Gilbert	Norm	Norm	1Fam

```
#Checking shape of new train dataset
df_new.shape
```

```
(953, 67)
```

```
#Checking shape of old train dataset
df.shape
```

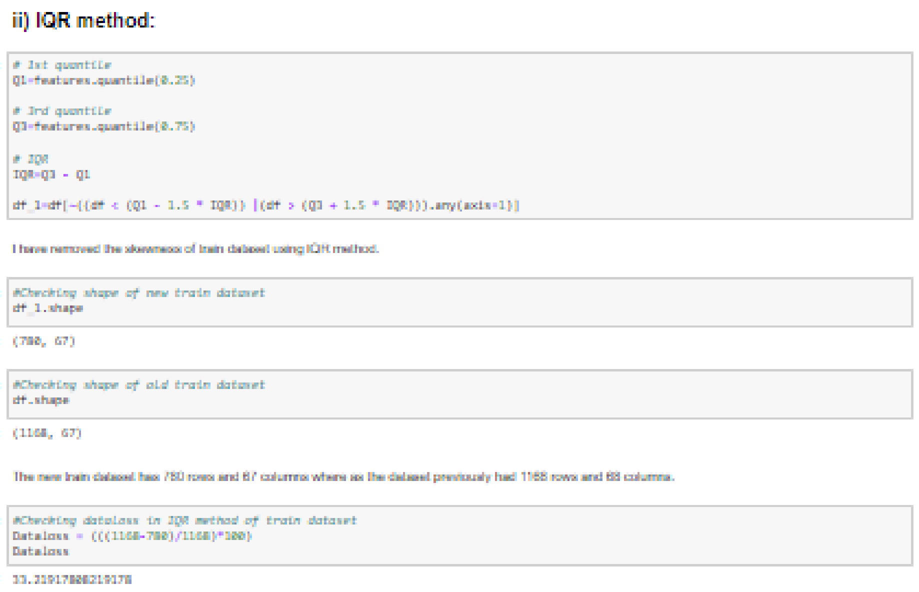
```
(1168, 67)
```

The new train dataset has 953 rows and 67 columns where as the dataset previously had 1168 rows and 68 columns.

```
#Checking dataloss in zscore method of train dataset
Dataloss = (((1168-953)/1168)*100)
Dataloss
```

```
18.40753424657534
```

Using Z-score method for trian dataset the data loss is more than 10% so let us have a look into IQR method to remove outliers.



ii) IQR method:

```
# 1st quantile
Q1=features.quantile(0.25)

# 3rd quantile
Q3=features.quantile(0.75)

# IQR
IQR=Q3 - Q1

df_1=df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

I have removed the skewness of train dataset using IQR method.

```
#Checking shape of new train dataset
df_1.shape
```

```
(780, 67)
```

```
#Checking shape of old train dataset
df.shape
```

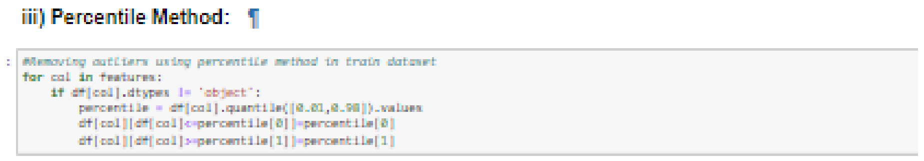
```
(1168, 67)
```

The new train dataset has 780 rows and 67 columns where as the dataset previously had 1168 rows and 68 columns.

```
#Checking dataloss in IQR method of train dataset
Dataloss = (((1168-780)/1168)*100)
Dataloss
```

```
33.21917808219178
```

The new train dataset has 780 rows and 67 columns where as the dataset previously had 1168 rows and 68 columns. In IQR method of train dataset the data loss is more than 10% so let us have a look into percentile method to remove outliers.

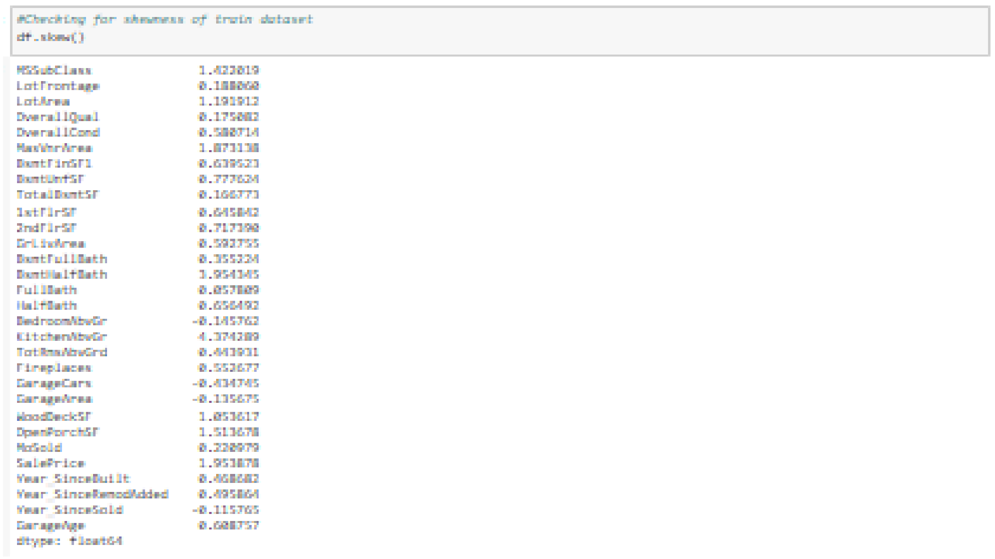


iii) Percentile Method: ¶

```
#Removing outliers using percentile method in train dataset
for col in features:
    if df[col].dtypes != 'object':
        percentile = df[col].quantile([0.01,0.98]).values
        df[col][df[col]<=percentile[0]]=percentile[0]
        df[col][df[col]>=percentile[1]]=percentile[1]
```

We have successfully removed outliers in train dataset using percentile method.

Checking for skewness:



```
#Checking for skewness of train dataset
df.skew()
```

```
MSSubClass               1.422019
LotFrontage              0.188060
LotArea                  1.191912
OverallQual              0.175082
OverallCond              0.580714
MasVnrArea               1.873138
BsmtFinSF1               0.639523
BsmtUnfSF                0.777624
TotalBsmtSF              0.166773
1stFlrSF                 0.645842
2ndFlrSF                 0.717390
GrLivArea                0.592755
BsmtFullBath             0.355224
BsmtHalfBath             3.954345
FullBath                 0.057809
HalfBath                 0.656492
BedroomAbvGr            -0.145762
KitchenAbvGr             4.374289
TotRmsAbvGrd             0.443931
Fireplaces               0.552677
GarageCars              -0.434745
GarageArea              -0.135675
WoodDeckSF               1.053617
OpenPorchSF              1.513678
MoSold                   0.220979
SalePrice                1.953878
Year_SinceBuilt          0.468682
Year_SinceRemodAdded     0.495864
Year_SinceSold          -0.115765
GarageAge                0.608757
dtype: float64
```

Observations

We can observe that the following columns have skewness present in train dataset

- MSSubClass
- LotArea
- OverallCond
- MasVnrArea
- BsmtFinSF1
- BsmtUnfSF
- 1stFlrSF
- 2ndFlrSF
- GrLivArea
- BsmtHalfBath
- HalfBath
- KitchenAbvGr
- Fireplaces
- WoodDeckSF
- OpenPorchSF
- SalePrice
- GarageAge

SalePrice is the target we need not remove skewness in this column.And MSSubClass and OverallCond are seems to be categorical so let us ignore these columns.

Dropping unnecessary column in train dataset

```
#Dropping unnecessary column in train dataset
df = df.drop(["GarageAge"],axis=1)
```

Removing skewness using yeo-johnson method for train dataset:

```
#Creating a list of skewed features in train dataset
fea=['LotArea','MasVnrArea','BsmtFinSF1','BsmtUnfSF','1stFlrSF','2ndFlrSF','GrLivArea','BsmtHalfBath',
```

Taking a list as fea with all the columns with skewness in train dataset.

```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
'''
parameters:
method = 'box_cox' or 'yeo-johnson'
'''
```

```
"\nparameters:\nmethod = 'box_cox' or 'yeo-johnson'\n"
```

Using yeo_johnson method we have removed the skewness in train dataset.

```
df[fea] = scaler.fit_transform(df[fea].values)
```

```
#Checking skewness again in train dataset
df[fea].skew()
```

```
LotArea         0.077861
MasVnrArea      0.415092
BsmtFinSF1     -0.418554
BsmtUnfSF      -0.304290
1stFlrSF       -0.000731
2ndFlrSF        0.279883
GrLivArea      -0.005974
BsmtHalfBath    3.954345
HalfBath        0.498003
KitchenAbvGr    0.000000
Fireplaces      0.076595
WoodDeckSF      0.110387
OpenPorchSF    -0.010092
dtype: float64
```

After removing skewness we have high skewness in BsmtHalfBath .let us drop this column from the train dataset

```
#Dropping unnecessary column
df = df.drop(["BsmtHalfBath"],axis=1)
```

Ordinal Encoding:

```
#Replacing ratings with suitable numbers in required columns in train dataset
column = ['ExterQual','ExterCond','BsmtQual','BsmtCond','HeatingQC','KitchenQual','GarageQual','Garage
for i in column:
    df[i] = df[i].replace({'Ex':5, 'Gd':4, 'TA':3, 'Fa':2, 'Po':1, 'None':0})
```

```
#Replacing ratings with suitable numbers in required columns in test dataset
column = ['ExterQual','ExterCond','BsmtQual','BsmtCond','HeatingQC','KitchenQual','GarageQual','Garage
for i in column:
    dff[i] = dff[i].replace({'Ex':5, 'Gd':4, 'TA':3, 'Fa':2, 'Po':1, 'None':0})
```

We have replaced all rating entries with required numbers.

```
#Ordinal encoding for train dataset
from sklearn.preprocessing import OrdinalEncoder
OE = OrdinalEncoder()
for i in df.columns:
    if df[i].dtypes=='object':
        df[i]=OE.fit_transform(df[i].values.reshape(-1,1))
```

```
#Ordinal encoding for test dataset
from sklearn.preprocessing import OrdinalEncoder
OE = OrdinalEncoder()
for i in dff.columns:
    if dff[i].dtypes=='object':
        dff[i]=OE.fit_transform(dff[i].values.reshape(-1,1))
```

We have encoded all my categorical columns in train and test datasets using Ordinal encoder.

Correlation:

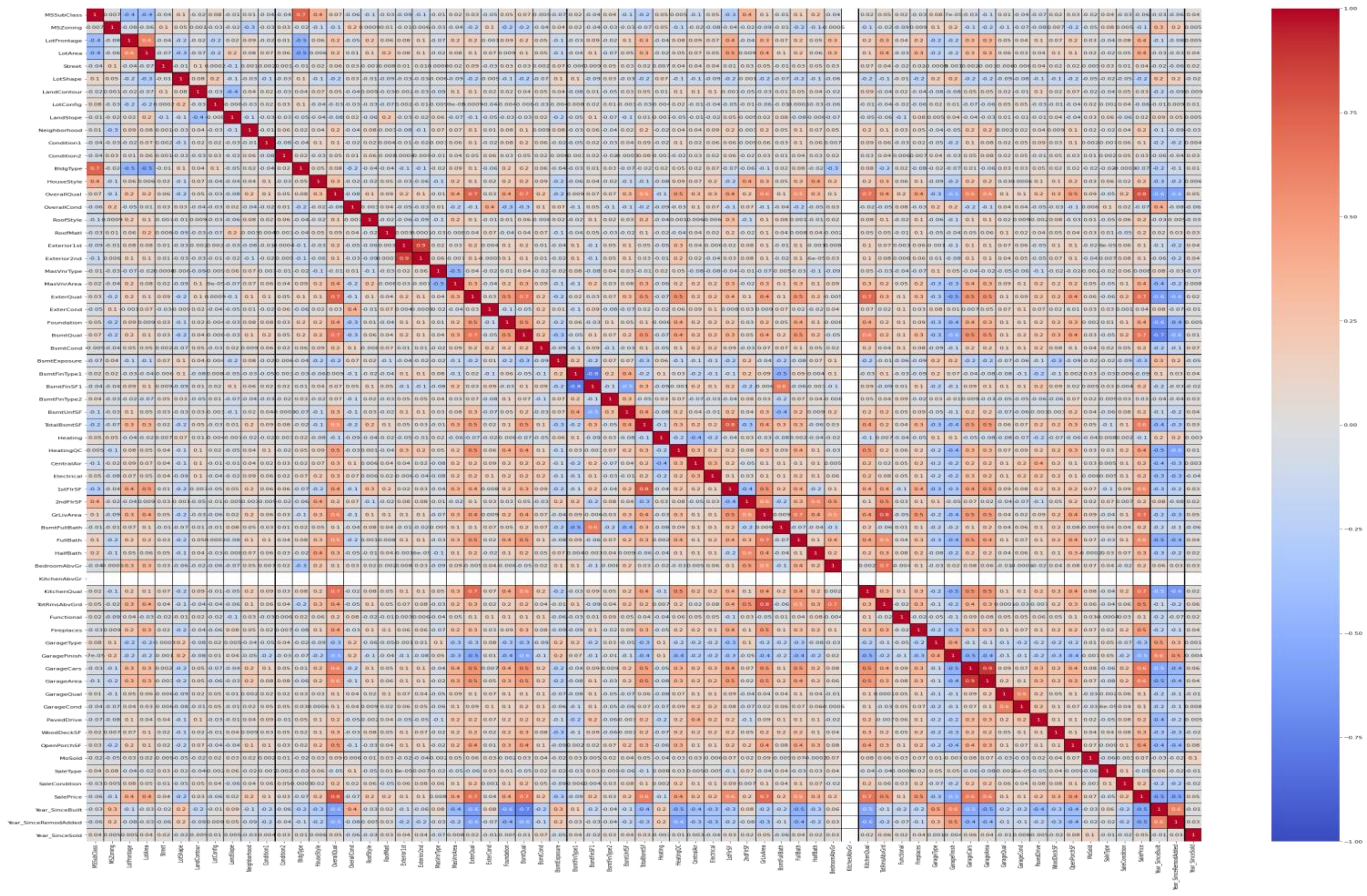
```
#Correlation of train dataset
cor=df.corr()
cor
```

```
                      MSSubClass  MSZoning  LotFrontage   LotArea    Street  LotShape  LandContour  LotConfig  LandS
MSSubClass              1.000000  0.007478    -0.391424 -0.400825 -0.035981  0.104485    -0.021387   0.076880  -0.01
MSZoning                0.007478  1.000000    -0.084834 -0.058912  0.140215  0.053655     0.001175  -0.027246  -0.02
LotFrontage            -0.391424 -0.084834     1.000000  0.596143 -0.044573 -0.157341    -0.016620  -0.201691   0.02
LotArea                -0.400825 -0.058912     0.596143  1.000000 -0.072669 -0.287003    -0.074834  -0.198998   0.17
Street                 -0.035981  0.140215    -0.044573 -0.072669  1.000000 -0.012941     0.105226   0.000153  -0.14
...                          ...       ...          ...       ...       ...       ...          ...        ...
SaleCondition          -0.028981  0.004501     0.076587  0.046115  0.014176 -0.054905     0.047715   0.043692  -0.06
SalePrice              -0.060775 -0.133221     0.358470  0.394343  0.044753 -0.248171     0.032836  -0.060452   0.01
Year_SinceBuilt        -0.031787  0.296752    -0.130022 -0.028547 -0.021386  0.231550    -0.158435  -0.009537   0.08
Year_SinceRemodAdded   -0.056618  0.174586    -0.080871 -0.028632 -0.057866  0.155428    -0.086936   0.009281   0.04
Year_SinceSold          0.038595  0.004964     0.005074  0.039843  0.019635 -0.021421    -0.009499   0.009817   0.00

65 rows × 65 columns
```

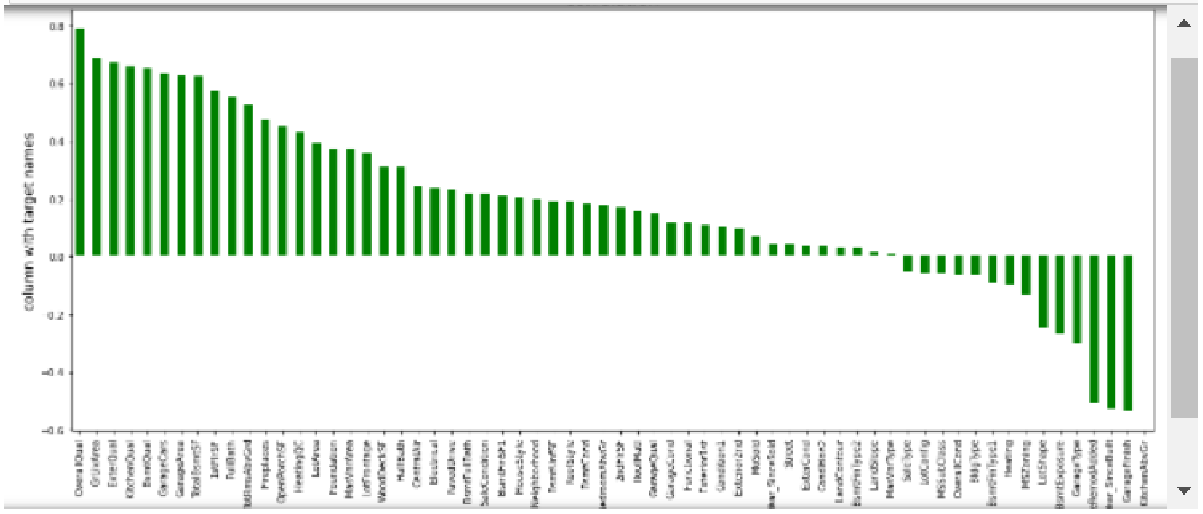
Above are the correlations of all the pair of features of train dataset.To get better visualization on the correlation of features,let us plot it using heat map.

Visualizing the correlation matrix by plotting heat map for train dataset



We can clearly observe a multicollinearity issue in some of the features of train dataset so we have to check VIF and Let us plot a bar graph to get better insight on targets correlation with other features.

```
plt.figure(figsize=(20,8))
df.corr()['SalePrice'].sort_values(ascending=False).drop(['SalePrice']).plot(kind='bar',color='g')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



Observations

We can observe the correlation of the target with the other features in the train dataset

- Here most of the features have positive correlation where as few of the features are negatively correlated.

Separating features and label in train dataset:

```
x = df.drop("SalePrice",axis=1)
y = df["SalePrice"]
```

We have separated the target and independent columns.

Scaling the train data using standard scaler:

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

We have scaled the train data using standard scaler.

```
X.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	C
0	1.508301	-0.021646	0.039092	-1.306083	0.058621	-1.373107	0.318473	0.606420	-0.226126	0.142224	
1	-0.877042	-0.021646	1.321126	1.356458	0.058621	-1.373107	0.318473	0.606420	3.295414	-0.024227	
2	0.077095	-0.021646	1.160948	0.113089	0.058621	-1.373107	0.318473	-1.220661	-0.226126	0.475125	
3	-0.877042	-0.021646	1.855050	0.530989	0.058621	-1.373107	0.318473	0.606420	-0.226126	0.308675	
4	-0.877042	-0.021646	0.039092	1.497522	0.058621	-1.373107	0.318473	-0.611634	-0.226126	0.308675	

This is the train data of independent variables after scaling.

Checking for multicolinearity issue in train dataset using VIF:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	5.068462	MSSubClass
1	1.351449	MSZoning
2	2.025904	LotFrontage
3	2.619867	LotArea
4	1.104743	Street
...	...	...
59	1.111644	SaleType
60	1.179899	SaleCondition
61	7.514182	Year_SinceBuilt
62	2.994873	Year_SinceRemodAdded
63	1.086942	Year_SinceSold

64 rows × 2 columns

We can observe that GrLivArea has high VIF. We can drop this columnn to reduce the impact of multicollinearity in the dataset

```
#Droping high VIF columns
X = X.drop(["GrLivArea"],axis=1)
```

We have dropped the GrLivArea column

Checking VIF again

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

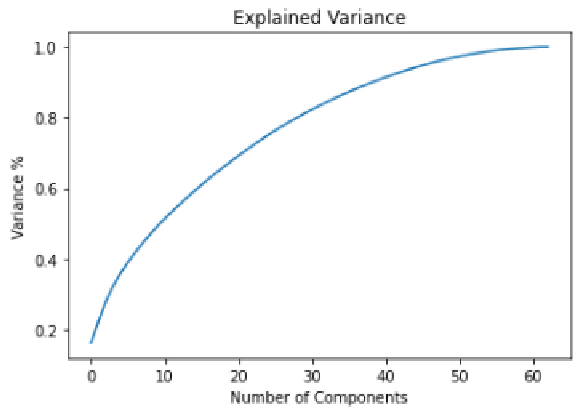
	vif_Features	Features
0	5.062507	MSSubClass
1	1.350491	MSZoning
2	2.022069	LotFrontage
3	2.617317	LotArea
4	1.099038	Street
...	...	...
58	1.109443	SaleType
59	1.179661	SaleCondition
60	7.372528	Year_SinceBuilt
61	2.992307	Year_SinceRemodAdded
62	1.086491	Year_SinceSold

63 rows × 2 columns

Although there are few columns in the dataset with high VIF than expected, we cannot drop them from the dataset without knowing their importance for the target/ dependent variable, so the multicolinearity issue is solved in train dataset upto some extent.

Principle Component Analysis on train dataset

```
from sklearn.decomposition import PCA
pca = PCA()
principleComponents = pca.fit_transform(X)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance %')
plt.title('Explained Variance')
plt.show()
```



60 components od the dataset explain around 95% variance in output/target

Selecting Kbest Features

```
from sklearn.feature_selection import SelectKBest, f_classif
bestfeat = SelectKBest(score_func = f_classif, k = 'all')
fit = bestfeat.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

```
fit = bestfeat.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
dfcolumns.head()
featureScores = pd.concat([dfcolumns,dfscores],axis = 1)
featureScores.columns = ['Feature', 'Score']
print(featureScores.nlargest(65,'Score'))
```

```
         Feature     Score
14   OverallQual  5.303071
22     ExterQual  3.404921
25      BsmtQual  2.882415
44   KitchenQual  2.809819
50    GarageCars  2.611142
..           ...       ...
30  BsmtFinType2  0.823425
10    Condition1  0.803848
46    Functional  0.797839
11    Condition2  0.782965
58      SaleType  0.778011

[62 rows x 2 columns]
```

We can see the importance of each feature in the above output

Dropping some least important columns from the dataset

```
#Dropping some least important columns from the dataset
X = X.drop(['SaleType'],axis=1)
X = X.drop(["Condition2"],axis=1)
```

We have dropped the above columns from the dataset

II. Test Dataset

The EDA, Feature extraction, Feature engineering, Data cleaning, fixing multicollinearity has been done to the test data set in the similar process as we have done to the train dataset.

Next I have checked for the shape of both the train data and the test data after cleaning and feature engineering

```
#Checking the shape of X in train data
X.shape
```

```
(1168, 61)
```

```
#Checking the shape of X in test data
X_1.shape
```

```
(292, 61)
```

Both have equal no of columns so we can proceed further with model training

Model building using train dataset:

Testing of Identified Approaches (Algorithms)

Finding Best Random State and Accuracy:

```
#importing necessary libraries
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestRegressor
maxAccu=0
maxRS=0
for i in range(1,200):
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.30, random_state =i)
    mod = RandomForestRegressor()
    mod.fit(X_train, y_train)
    pred = mod.predict(X_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.8970199276146686  on Random_state  135
```

We got the best accuracy and random state.

Creating train test split.

```
#Creating train test split.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.30,random_state=maxRS)
```

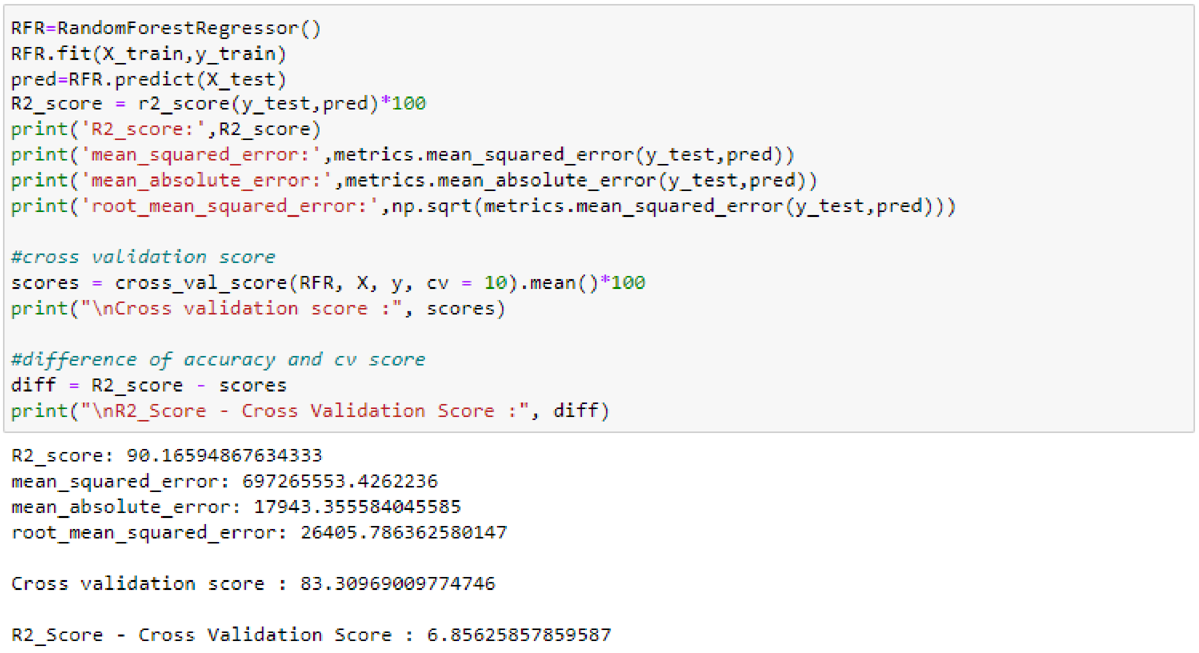
Created train test split.

Run and Evaluate selected models

Regression Algorithms:

```
#importing necessary libraries
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor as KNN
from sklearn.linear_model import SGDRegressor
from xgboost import XGBRegressor
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingRegressor
from sklearn import metrics
```

i) RandomForestRegressor:



```
RFR=RandomForestRegressor()
RFR.fit(X_train,y_train)
pred=RFR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(RFR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

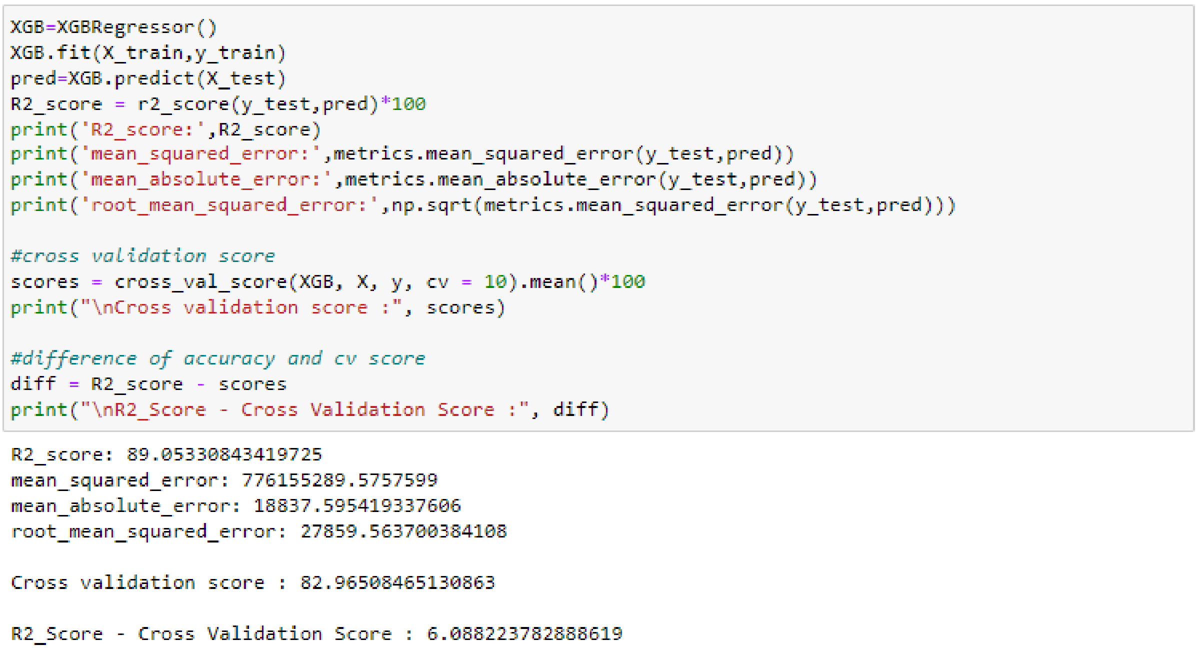
```
R2_score: 90.16594867634333
mean_squared_error: 697265553.4262236
mean_absolute_error: 17943.355584045585
root_mean_squared_error: 26405.786362580147

Cross validation score : 83.30969009774746

R2_Score - Cross Validation Score : 6.85625857859587
```

RandomForestRegressor is giving 90.16% r2_score.

ii) XGBRegressor:



```
XGB=XGBRegressor()
XGB.fit(X_train,y_train)
pred=XGB.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(XGB, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

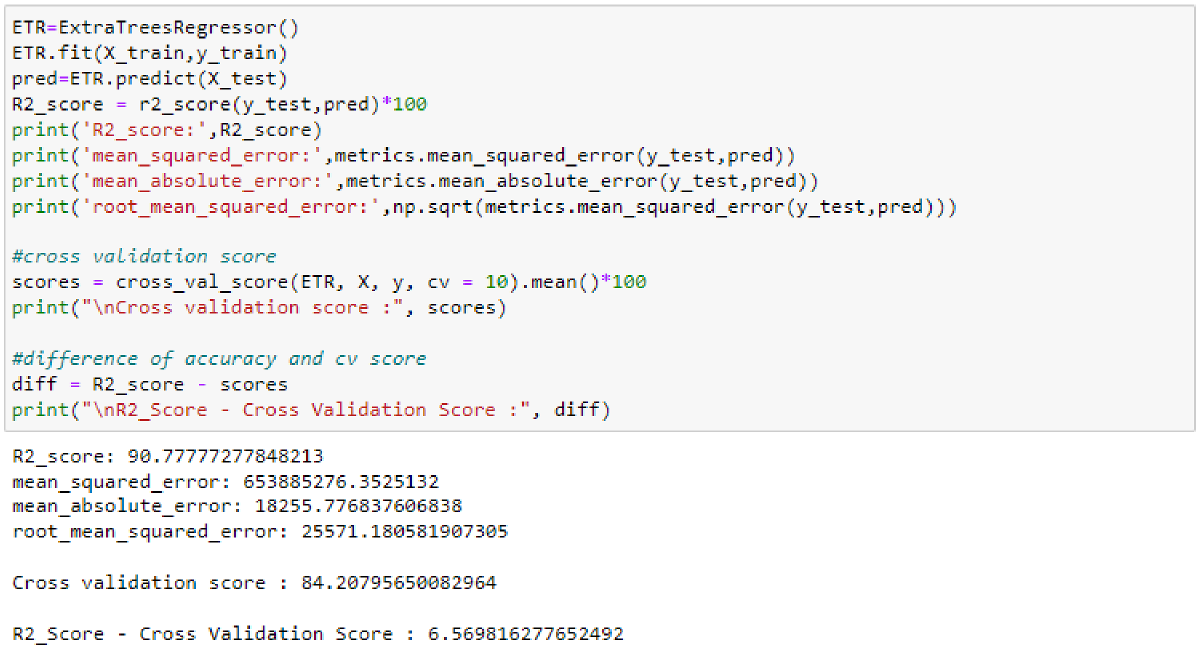
```
R2_score: 89.05330843419725
mean_squared_error: 776155289.5757599
mean_absolute_error: 18837.595419337606
root_mean_squared_error: 27859.563700384108

Cross validation score : 82.96508465130863

R2_Score - Cross Validation Score : 6.088223782888619
```

XGBRegressor is giving me 89.05% r2_score.

iii) ExtraTreesRegressor:



```
ETR=ExtraTreesRegressor()
ETR.fit(X_train,y_train)
pred=ETR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(ETR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

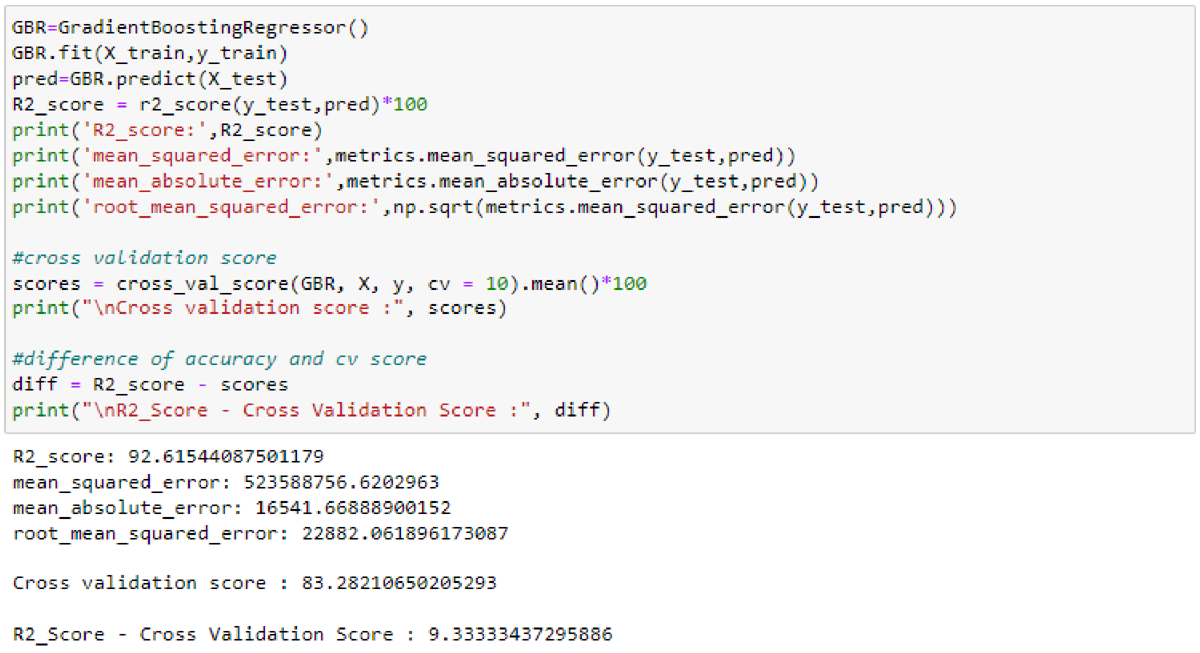
```
R2_score: 90.77777277848213
mean_squared_error: 653885276.3525132
mean_absolute_error: 18255.776837606838
root_mean_squared_error: 25571.180581907305

Cross validation score : 84.20795650082964

R2_Score - Cross Validation Score : 6.569816277652492
```

ExtraTreesRegressor is giving me 90.77% r2_score.

iv) GradientBoostingRegressor:



```
GBR=GradientBoostingRegressor()
GBR.fit(X_train,y_train)
pred=GBR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(GBR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 92.61544087501179
mean_squared_error: 523588756.6202963
mean_absolute_error: 16541.66888900152
root_mean_squared_error: 22882.061896173087

Cross validation score : 83.28210650205293

R2_Score - Cross Validation Score : 9.33333437295886
```

GradientBoostingRegressor is giving me 92.61% r2_score.

v) DecisionTreeRegressor:

```
DTR=DecisionTreeRegressor()
DTR.fit(X_train,y_train)
pred=DTR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))

#cross validation score
scores = cross_val_score(DTR, X, y, cv = 10).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 73.7654809357102
mean_squared_error: 1860110940.2621083
mean_absolute_error: 29436.76923076923
root_mean_squared_error: 43129.003469383664

Cross validation score : 62.99222177821948

R2_Score - Cross Validation Score : 10.77325915749072
```

DecisionTreeRegressor is giving me 73.76% r2_score.

After observing the difference of model accuracy and cross validation score we can see RandomForestRegressor as the best model.

Key Metrics for success in solving problem under consideration

Hyper parameter tunning for best model:

```
#importing necessary libraries
from sklearn.model_selection import GridSearchCV
```

```
parameter = {'n_estimators':[30,60,80],
             'max_depth': [10,20,40],
             'min_samples_leaf':[1,2,5,10,20,30],
             'min_samples_split':[5,10,20],
             'criterion':['mse','mae'],
             'max_features':["auto","sqrt","log2"]}
```

Giving RFR parameters and Running grid search CV for RFR

```
#Running grid search CV for RFR.
GCV.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 972 candidates, totalling 4860 fits

GridSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'criterion': ['mse', 'mae'], 'max_depth': [10, 20, 40],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 5, 10, 20, 30],
                         'min_samples_split': [5, 10, 20],
                         'n_estimators': [30, 60, 80]},
             verbose=1)
```

Tuning the model using GCV.

Getting the best parameters

```
#Getting the best parameters
GCV.best_params_
```

```
{'criterion': 'mae',
 'max_depth': 10,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 60}
```

Got the best parameters for RFR.

Interpretation of the Results

Assigning a variable for the best model

```
# Assigning a variable for the best model
Best_mod=RandomForestRegressor(criterion='mae',max_features='sqrt',min_samples_split=5,n_estimators=60,
Best_mod.fit(X_train,y_train)
pred=Best_mod.predict(X_test)
print('R2_Score:',r2_score(y_test,pred)*100)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print("RMSE value:",np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
R2_Score: 89.24134955874074
mean_squared_error: 762822575.0660958
mean_absolute_error: 18537.213319088318
RMSE value: 27619.242840202838
```

This is the model r2_score after tuning. We have 89.24% as r2_score which is good to proceed with.

Saving the model:

```
# Saving the model using .pkl
import joblib
joblib.dump(Best_mod,"House_Price.pkl")
```

```
['House_Price.pkl']
```

We have saved the model as House_Price.Using .pkl

Predicting House Price for test dataset using Saved model of train dataset

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

#Prediction
prediction = model.predict(X_test)
prediction
```

```
array([145841.125     , 153632.6      , 169143.16666667, 261018.09166667,
       103341.075     , 267955.39166667, 184268.3      , 124148.81666667,
       139742.08333333, 113145.34166667,  89805.83333333, 143810.025     ,
       252080.35833333, 162553.53333333, 139033.33333333, 181588.66666667,
       184987.775     , 170839.25833333, 288342.58333333, 164203.03333333,
       137665.175     , 154953.40833333, 133481.01666667, 290673.96666667,
       121808.98333333, 145160.025     , 240658.84166667, 229211.53333333,
       106231.26666667, 156118.3      , 178086.71666667, 129372.28333333,
       195948.38333333, 121137.15     , 165014.56666667, 159123.58333333,
       275134.09166667, 126673.48333333, 117400.96666667, 150461.95833333,
       297883.69166667, 183427.83333333, 405768.23333333,  80559.58333333,
       109345.61666667, 281538.975    , 133815.41666667, 298696.78333333,
       239190.53333333, 137661.94166667, 150739.66666667, 116842.01666667,
       191861.91666667, 111308.25     , 206845.10833333, 153407.04166667,
       219597.125     , 127186.86666667, 375809.6      , 293947.63333333,
       137646.95     , 164074.16666667, 189651.16666667, 137350.24166667,
       178699.21666667, 188758.75     , 228629.41666667, 234886.66666667,
       200557.36666667, 111298.09166667, 123513.54166667, 107758.68333333,
       359464.36666667, 149740.15833333, 167026.64166667, 302276.05     ,
       178214.91666667, 278845.325    , 304799.65     , 153884.64166667,
       119955.4      , 162233.89166667, 126021.66666667, 124848.54166667,
       278846.475    , 105225.91666667, 128671.60833333, 197305.125    ,
       270233.09166667, 263657.80833333, 152734.725    , 108106.16666667,
       161724.51666667,  98449.14166667, 107383.9      , 218425.83333333,
        99751.66666667, 135024.95     , 118304.86666667, 280358.375    ,
```

Creating a data frame of actual vs predicted values

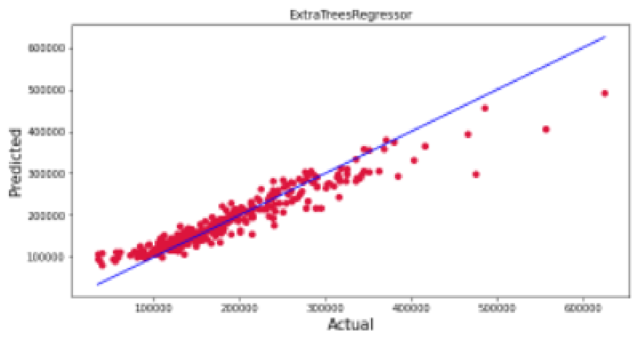
```
pd.DataFrame([model.predict(X_test)[:],y_test[:]],index=["Predicted","Actual"])
```

	0	1	2	3	4	5	6	7	8	9	1
Predicted	145841.125	153632.6	169143.166667	261018.091667	103341.075	267955.391667	184268.3	124148.816667	139742.083333	113145.341667	89805.83333
Actual	120000.000	140000.0	172500.000000	244600.000000	88000.000	252000.000000	176000.0	124900.000000	120000.000000	87000.000000	37900.0000

Above are the predicted values and the actual values.They look similar.

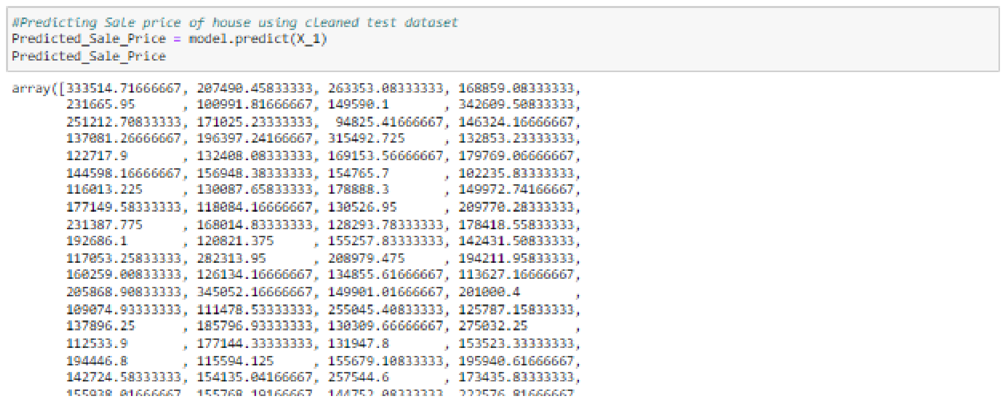
Visualizing the actual vs predicted values

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



Plotting Actual vs Predicted. To get better insights. Blue line is the actual line and red dots are the predicted values. They are similar apart from few exceptions

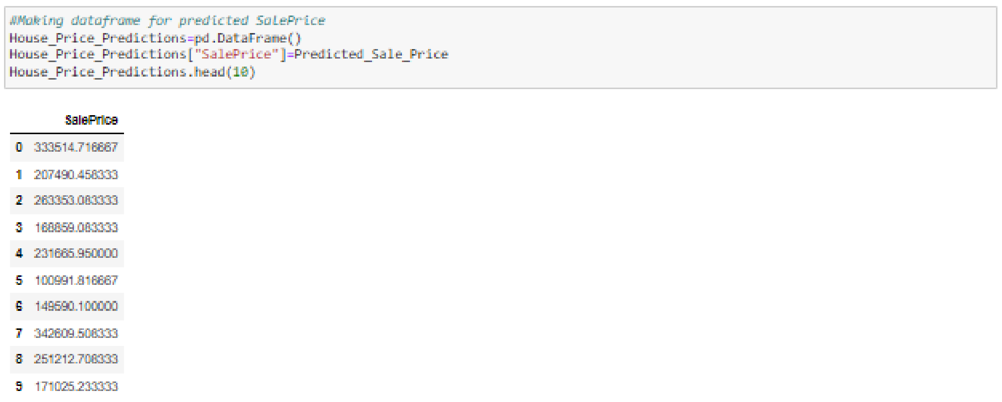
Predicting Sale price of house using cleaned test dataset



```
#Predicting Sale price of house using cleaned test dataset
Predicted_Sale_Price = model.predict(X_1)
Predicted_Sale_Price
```

```
array([333514.71666667, 207490.45833333, 263353.08333333, 168859.08333333,
       231665.95      , 100991.81666667, 149590.1       , 342609.50833333,
       251212.70833333, 171025.23333333,  94825.41666667, 146324.16666667,
       137081.26666667, 196397.24166667, 315492.725     , 132853.23333333,
       122717.9       , 132408.08333333, 169153.56666667, 179769.06666667,
       144598.16666667, 156948.38333333, 154765.7       , 102235.83333333,
       116013.225     , 130087.65833333, 178888.3       , 149972.74166667,
       177149.58333333, 118084.16666667, 130526.95      , 209770.28333333,
       231387.775     , 168014.83333333, 128293.78333333, 178418.55833333,
       192686.1       , 120821.375     , 155257.83333333, 142431.50833333,
       117053.25833333, 282313.95      , 208979.475     , 194211.95833333,
       160259.00833333, 126134.16666667, 134855.61666667, 113627.16666667,
       205868.90833333, 345052.16666667, 149901.01666667, 201000.4       ,
       109074.93333333, 111478.53333333, 255045.40833333, 125787.15833333,
       137896.25      , 185796.93333333, 130309.66666667, 275032.25      ,
       112533.9       , 177144.33333333, 131947.8       , 153523.33333333,
       194446.8       , 115594.125     , 155679.10833333, 195940.61666667,
       142724.58333333, 154135.04166667, 257544.6       , 173435.83333333,
       155938.01666667  155768.19166667  144752.08333333  222576.81666667
```

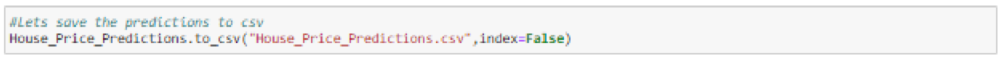
Making dataframe for predicted SalePrice



```
#Making dataframe for predicted SalePrice
House_Price_Predictions=pd.DataFrame()
House_Price_Predictions["SalePrice"]=Predicted_Sale_Price
House_Price_Predictions.head(10)
```

	SalePrice
0	333514.716667
1	207490.458333
2	263353.083333
3	168859.083333
4	231665.950000
5	100991.816667
6	149590.100000
7	342609.508333
8	251212.708333
9	171025.233333

Saving the predictions to csv



```
#Lets save the predictions to csv
House_Price_Predictions.to_csv("House_Price_Predictions.csv",index=False)
```

We have saved the predicted values as csv file.

Conclusion

Key Findings and Conclusions of the Study

Based on the in-depth analysis of the Housing Project, The Exploratory analysis of the datasets, and the analysis of the Outputs of the models the following observations are made:

● Structural attributes of the house Structural attributes of the house like lot size, lot shape, quality and condition of the house, garage capacity, rooms, Lot frontage, number of bedrooms, bathrooms, overall finishing of the house etc play a big role in influencing the house price.

● Neighborhood qualities can be included in deciding house price.

● Various plots like Barplots, stripplots and distplots helped in visualising the Feature-label relationships which corroborated the importance of structural and locational attributes for estimating Sale Prices.

● Due to the Training dataset being very small, the outliers had to be retained for proper training of the models.

● Therefore, Random Forest Regressor, being robust to outliers and being indifferent to nonlinear features, performed well despite having to work on small dataset.

Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle as it contains all types of data in it. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove missing value and to replace null value and zero values with their respective mean, median or mode. This study is an exploratory attempt to use five machine learning algorithms in estimating housing prices, and then compare their results.

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.

Limitations of this work and Scope for Future Work

- First drawback is the data leakage when we merge both train and test datasets.
- Followed by huge number of outliers and skewness which will reduce our model accuracy and efficiency.
- Also, we have tried best to deal with outliers, skewness, null values and zero values. So that we have achieved an accuracy of 90.13% even after dealing with these drawbacks.
- Also, this study will not cover all regression algorithms instead, it is focused on the chosen algorithm, starting from the basic regression techniques to the advanced ones.
- This model doesn't predict future prices of the houses mentioned by the customer. Due to this, the risk in investment in an apartment or an area increases considerably. To minimize this error, customers tend to hire an agent which again increases the cost of the process.
- While features that focus on structural and locational attributes of housing properties are crucial for estimating the Sale Price of Housing properties, they aren't the only factors that influence the value in the housing market. Data on Demographics (Age, Income, Regional preferences of buyers, purpose of buying a property) is very important for understanding the Housing market. Interest Rates too impact the price and demand of houses. Economic cycles also influence Real Estate prices. Government Policies, Regulations, Legalizations are also important factors that may influence the sales of houses. The availability of data on above features would help build a predictive model that would more accurately understand the relationship between the features and target variable and yield more accurate predictions.
- To conclude, the application of machine learning in property research is still at an early stage. We hope this study can provide some methodological improvements and contributions to property calculation and presenting an alternative to the valuation of housing prices. Future direction of research may consider incorporating additional property transaction data including more features.

Thank you