FLIP ROBO

HOUSING: PRICE PREDICTION

Submitted by:

R A J A S E K A R

ACKNOWLEDGMENT

I would like to acknowledge the contribution of following people without whose help and guidance this report would not have been completed .

I and support of **Flip Robo** Staff **Ms. Sapna Verma (SME) and Mr. Mohd Khasifh , Mr. Arunkumar** and **Data Trained Institute** with respect and gratitude whose expertise, guidance and support has made this report possible. I am indeed proud and fortunate to be supported by him/her.

INTRODUCTION

- Business Problem Framing

A US-based housing company named **Surprise Housing** has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below. The company is looking at the prospective properties to enter into the market.

1. We are required to build a model using machine learning in order to predict the actual values of the prospective properties.
2. And to decide whether to invest in them or not.

For this company wants to know the :-

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

- Conceptual Background of the Domain Problem

House prices will get increase based on many factors those are

1. Neighborhood Comps (Comparative house) .
2. Location :-
    (a) Quality of local schools.
    (b) Employment opportunities.
    (c) Proximity to shopping entertainment and recreational centres.
3. House Price and Useable data.

4. Age and Condition.
5. Interest Rates.
6. Updates and Upgrades.

- Review of Literature :-

Following are the extension of above topic which deals with the factors influence the House price.

1. Neighborhood Comps :-

   Sale prices of similar home's in our neighborhood that have sold recently is one of the best indicators of the house values. These comparable homes are often refer to as "Comps". Whether it's a house appraisal a comparative market analysis done by an agent, or an open door evaluation, Most real estate experts will rely on comps to estimate house values. Luckily, computers are really good at this task. For example, we combine a robust data model that can analyze hundreds of pairs of comps for any given address with insights from local pricing experts. We're then able to provide home sellers with a competitive, cash offer in 24 hours so they can simplify the process and move on their own timeline.

2. Location :-

   When appraisers determine how much value to assign based on the location of the house, they're looking at three primary indicators, according to Inman:

   - Quality of local schools.
   - Employment Opportunities.
   - Proximity to shopping entertainment and recreational centre.

In addition, a location's proximity to highways, utility lines, and public transit can all impact a house overall value. When it comes to calculating a house's value, location can be more important than even the “size” and “condition” of the house.

3. House Price and Usable Prices :-

   Size is an important element to consider while estimating the house’s value since bigger homes can positively impact its valuation. The value of house is roughly estimated in price per square foot.

   In addition to square footage, a home’s usable space matters when determining its value. Garages, attics, and unfinished basements are generally not counted in usable square footage. So if you have a 2,000-square-foot home with a 600-square-foot garage, that’s only 1,400-square-feet of livable space.

   Livable space is what is most important to buyers and appraisers. Bedrooms and bathrooms are most highly valued, so the more beds and baths your home offers, the more your home is generally worth. However these trends are very locally specific.

4. Age and Condition :-

   Typically, homes that are newer appraise at a higher value. The fact that critical parts of the house, like plumbing, electrical, the roof, and appliances are newer and therefore less likely to break down, can generate savings for a buyer. For example, if a roof has a 20-year warranty, that’s money an owner will save over the next two decades, compared to an older home that may need a roof replaced in just a few years. Many buyers will pay top-dollar for a move-in-ready home. This is why most buyers require an inspection contingency in their contract — they want to negotiate repairs to avoid any major expenses following the sale.

5. Updates and Upgrades :-

   Updates and upgrades can add value to your home, especially in older homes that may have outdated features. However, not all home improvement projects are created equally. Additionally, some projects like adding a pool or wood floors tend to have bigger increases for more expensive homes, while projects like a kitchen remodel or adding a full bathroom tend to have a bigger increase for less expensive homes.

6. The Local Market :-

   Even if your home is in excellent condition, in the best location, with premium upgrades, the number of other properties for sale in your area and the number of buyers in the market can impact your home value.

   - If there are a lot of buyers competing for fewer homes it's a seller's market.
   - A Market with few buyers but many homes on the market is referred to as a buyer's market.

   Additionally, market conditions can affect how long it takes your home to sell. In a seller's market, homes tend to sell quickly, whereas in a buyer's market it's typical for homes to see longer days on market (DOM).

7. Economic Indicators :-

   The broader economy often impacts a person's ability to buy or sell a home, so in slower economic conditions, the housing market can struggle. For example, if employment or wage growth slows, then fewer people might be able to afford a home or there may also be less opportunity to relocate for new opportunities.

Analytical Problem Framing

1. What is Analytical problem framing?

Analytic problem framing involves translating the business problem into terms that can be addressed analytically via data and modeling. It's at this stage that you work backwards from the results / outputs you want to the data/inputs you're going to need, where you identify potential drivers and hypotheses to test, and where you nail down your assumptions.
Analytic problem framing is the antithesis of merely working with the ready-to-hand data and seeing what comes of it, hoping for something insightful. Typically, the process moves on from here to data collection, cleansing and transformation,Methodology selection and model building, never to return. But
if you're willing to borrow and use a concept from complex
adaptive systems – maps and models – you can make repeat use
of this stage to improve your overall outcome.

2. Hardware Requirements

A mid level computer that runs on Intel i3/i5/i7 or A10/A11/M1 or ryzen 3/5 or any other equivalent chipset and a suitable processor.

3. Software Requirements

Windows / Linux /Mac OS

4. Tools, Libraries and Packages used

Tool:

1. Anaconda Navigator
2. Jupyter Notebook

Libraries and Packages:

1. Numpy.
2. Pandas.
3. Matplotlib.
4. Seaborn.

- ## **Data Sources and their formats :-**

  1. The Data-set seems to be divided into 2 sets, one for training and one for testing.
  2. The train data-set seems to have 1168 rows and 81 columns.
  3. The test data-set seems to have 292 rows and 80 columns.
  4. Both the files were given as csv files.
  5. The data-set contains 3- float data types, 35-int data types and 43-object data types.

- ## Data Preprocessing Steps :-
  1. The columns which didn't give any feature importance have been dropped.
  2. The Null values are found and visualized using missing package as follows.
  3. After dropping the unnecessary column we can have only 41 columns previously 81 columns.

```
Out[10]: MSSubClass        int64
         MSZoning         object
         Street           object
         LandContour      object
         LandSlope        object
         BldgType         object
         HouseStyle       object
         OverallCond       int64
         YearBuilt         int64
         Exterior1st      object
         Exterior2nd      object
         ExterCond        object
         BsmtQual         object
         BsmtCond         object
         BsmtFinSF1        int64
         BsmtUnfSF         int64
         TotalBsmtSF       int64
         1stFlrSF          int64
         2ndFlrSF          int64
         GrLivArea         int64
         FullBath          int64
         BedroomAbvGr      int64
         KitchenAbvGr      int64
         KitchenQual      object
         TotRmsAbvGrd      int64
         GarageYrBlt     float64
         GarageFinish     object
         GarageCars        int64
         GarageArea        int64
         GarageQual       object
         GarageCond       object
         OpenPorchSF       int64
         EnclosedPorch     int64
         3SsnPorch         int64
         ScreenPorch       int64
         PoolArea          int64
         PoolQC           object
         MiscFeature      object
         MiscVal           int64
         YrSold            int64
         SalePrice         int64
         dtype: object
```

a-data/DATA---ANALYS × New Tab × Home Page - Select or create a n × H

localhost:8888/notebooks/Housing%20Price%20-%20Flip%20Robo.ipynb

Maps News Gmail ICC Women's ODI B... Selecting, Slicing an... Download image wi...

jupyter Housing Price - Flip Robo Last Checkpoint: Last Thursday a

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

In [13]:
```
#Checking for null values in the dataset
df_train.isnull().sum()
```

Out[13]:
```
MSSubClass       0
MSZoning         0
Street           0
LandContour      0
LandSlope        0
BldgType         0
HouseStyle       0
OverallCond      0
YearBuilt        0
Exterior1st      0
Exterior2nd      0
ExterCond        0
BsmtQual        30
BsmtCond        30
BsmtFinSF1       0
BsmtUnfSF        0
TotalBsmtSF      0
1stFlrSF         0
```

pha-data/DATA---ANALYS × New Tab ×

localhost:8888/notebooks/Housing%20Price%20-%20Flip%20

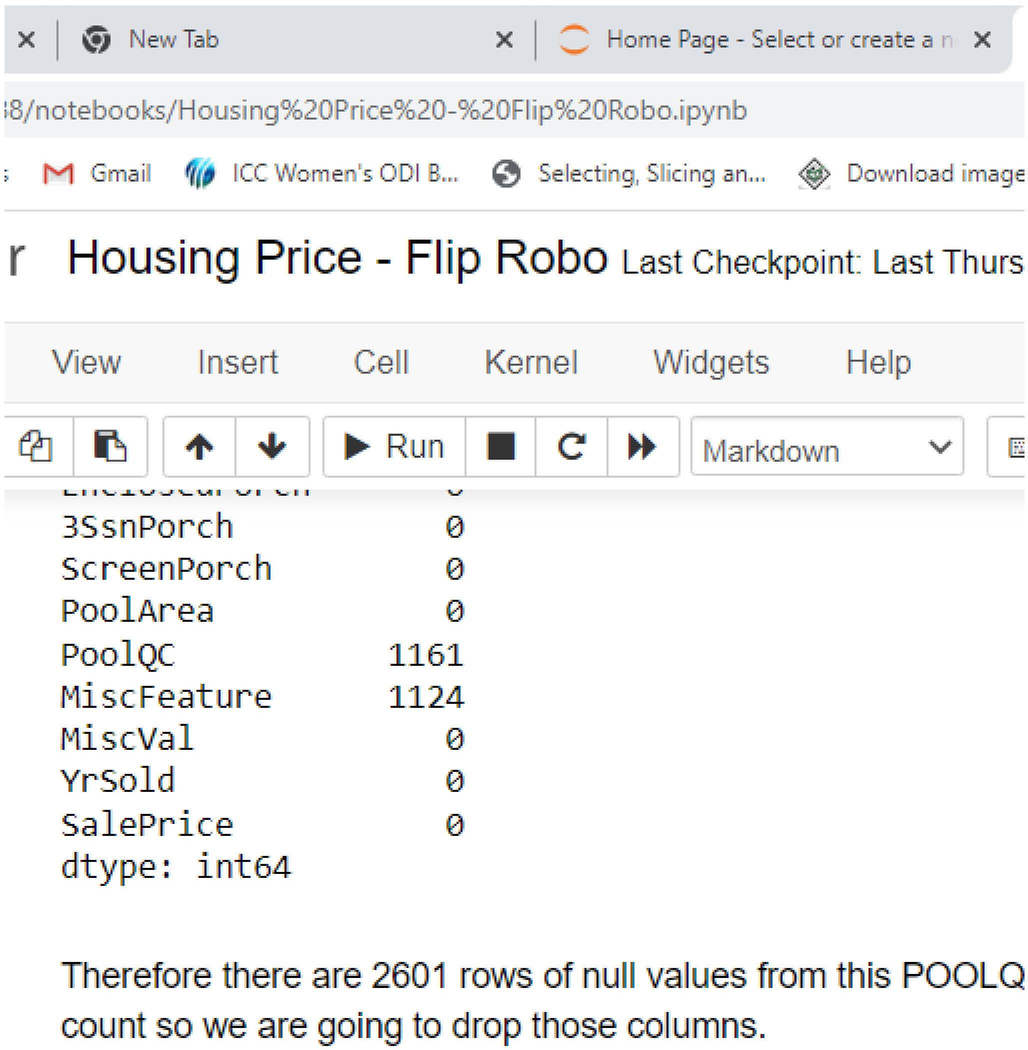
Maps News Gmail ICC Women's ODI B... Selecti

jupyter Housing Price - Flip Robo

File Edit View Insert Cell Kernel

Run

```
BsmtFinSF1         0
BsmtUnfSF          0
TotalBsmtSF        0
1stFlrSF           0
2ndFlrSF           0
GrLivArea          0
FullBath           0
BedroomAbvGr       0
KitchenAbvGr       0
KitchenQual        0
TotRmsAbvGrd       0
GarageYrBlt       64
GarageFinish      64
GarageCars         0
GarageArea         0
GarageQual        64
GarageCond        64
OpenPorchSF        0
EnclosedPorch      0
3SsnPorch          0
ScreenPorch        0
PoolArea           0
PoolQC          1161
```



3. Therefore there are 2601 rows of null values from this “POOLQC” and “Misc Feature” have more null values nearly equal to total count so we are going to drop those columns.
4. There are 17 OBJECT columns and 13 INT64 columns and 1 FLOAT64 column.
5. After dropping Misc feature and Poll QC then we can have 39 columns only.

- Data Preprocessing Done :-

1. Data cleaning is usually a necessary steps to build a good dataset for a model for predictions.
2. ".dropna" function is used to remove the null values.
3. "LABEL ENCODING" is next step we are implementing here with a Proper code and those steps are useful for transforming the object Data-type to integer data-type.

ha-data/DATA---ANALYS × New Tab × Home Page - Select or cr

localhost:8888/notebooks/Housing%20Price%20-%20Flip%20Robo.ipynb

Maps News Gmail ICC Women's ODI B... Selecting, Slicing an... Dc

jupyter Housing Price - Flip Robo Last Checkpoint:

File Edit View Insert Cell Kernel Widgets H

Run Markdown

```
YrSold           0
SalePrice        0
dtype: int64
```

In [19]:
```
df_train.dropna(inplace=True)
```

In [20]:
```
df_train.shape
```

Out[20]:
```
(1080, 39)
```

In [21]:
```
df_train.isnull().sum()
```

Out[21]:
```
MSSubClass      0
MSZoning        0
Street          0
LandContour     0
LandSlope       0
BldgType        0
HouseStyle      0
OverallCond     0
YearBuilt       0
Exterior1st     0
```

ha-data/DATA---ANALYS ✕ New Tab ✕

localhost:8888/notebooks/Housing%20Price%20-%20Flip%

Maps News Gmail ICC Women's ODI B... Se

jupyter Housing Price - Flip Rol

File Edit View Insert Cell Kerne

Run

```
HouseStyle       0
OverallCond      0
YearBuilt        0
Exterior1st      0
Exterior2nd      0
ExterCond        0
BsmtQual         0
BsmtCond         0
BsmtFinSF1       0
BsmtUnfSF        0
TotalBsmtSF      0
1stFlrSF         0
2ndFlrSF         0
GrLivArea        0
FullBath         0
BedroomAbvGr     0
KitchenAbvGr     0
KitchenQual      0
TotRmsAbvGrd     0
GarageYrBlt      0
GarageFinish     0
GarageCars       0
GarageArea       0
```

pha-data/DATA---ANALYS × | New Tab × | Home Page - S

localhost:8888/notebooks/Housing%20Price%20-%20Flip%20Robo.ipynb

Maps News Gmail ICC Women's ODI B... Selecting, Slicing an...

jupyter Housing Price - Flip Robo Last Chec

File Edit View Insert Cell Kernel Widget

Run Marl

```
GarageYrBlt      0
GarageFinish     0
GarageCars       0
GarageArea       0
GarageQual       0
GarageCond       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
MiscVal          0
YrSold           0
SalePrice        0
dtype: int64
```

O B S E R V A T I O N S :-

1. Therefore there are 2601 rows of null val
   total count so we are going to drop those
2. We have filled the null values of garage y
3. Therefore we have removed the null valu

A---ANALYS × | New Tab × | Home Page - Select or create a n × | Hou

:alhost:8888/notebooks/Housing%20Price%20-%20Flip%20Robo.ipynb

News | Gmail | ICC Women's ODI B... | Selecting, Slicing an... | Download image wi...

ıpyter **Housing Price - Flip Robo** Last Checkpoint: Last Thursday at

Edit View Insert Cell Kernel Widgets Help

Run | Markdown

```
df_train['GarageFinish']=le.fit_transform(df_train['Gar
```

In [54]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1080 entries, 0 to 1167
Data columns (total 39 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   MSSubClass      1080 non-null   int64
 1   MSZoning        1080 non-null   int32
 2   Street          1080 non-null   int32
 3   LandContour     1080 non-null   int32
 4   LandSlope       1080 non-null   int32
 5   BldgType        1080 non-null   int32
 6   HouseStyle      1080 non-null   int32
 7   OverallCond     1080 non-null   int64
 8   YearBuilt       1080 non-null   int64
 9   Exterior1st     1080 non-null   int32
 10  Exterior2nd     1080 non-null   int32
 11  ExterCond       1080 non-null   int32
 12  BsmtQual        1080 non-null   int32
 13  BsmtCond        1080 non-null   int32
```

pha-data/DATA---ANALYS | New Tab | Home Page - Select or create a n

localhost:8888/notebooks/Housing%20Price%20-%20Flip%20Robo.ipynb

Maps News Gmail ICC Women's ODI B... Selecting, Slicing an... Download imag

jupyter Housing Price - Flip Robo Last Checkpoint: Last Thurs

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

```
 10  Exterior2nd    1080 non-null   int32
 11  ExterCond      1080 non-null   int32
 12  BsmtQual       1080 non-null   int32
 13  BsmtCond       1080 non-null   int32
 14  BsmtFinSF1     1080 non-null   int64
 15  BsmtUnfSF      1080 non-null   int64
 16  TotalBsmtSF    1080 non-null   int64
 17  1stFlrSF       1080 non-null   int64
 18  2ndFlrSF       1080 non-null   int64
 19  GrLivArea      1080 non-null   int64
 20  FullBath       1080 non-null   int64
 21  BedroomAbvGr   1080 non-null   int64
 22  KitchenAbvGr   1080 non-null   int64
 23  KitchenQual    1080 non-null   int32
 24  TotRmsAbvGrd   1080 non-null   int64
 25  GarageYrBlt    1080 non-null   float64
 26  GarageFinish   1080 non-null   int32
 27  GarageCars     1080 non-null   int64
 28  GarageArea     1080 non-null   int64
 29  GarageQual     1080 non-null   int32
 30  GarageCond     1080 non-null   int32
 31  OpenPorchSF    1080 non-null   int64
```

ha-data/DATA---ANALYS × New Tab × Home Page - Select or create a n

localhost:8888/notebooks/Housing%20Price%20-%20Flip%20Robo.ipynb

Maps News Gmail ICC Women's ODI B... Selecting, Slicing an... Download ir

jupyter Housing Price - Flip Robo Last Checkpoint: Last Th

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

```
 19  GrLivArea      1080 non-null   int64
 20  FullBath       1080 non-null   int64
 21  BedroomAbvGr   1080 non-null   int64
 22  KitchenAbvGr   1080 non-null   int64
 23  KitchenQual    1080 non-null   int32
 24  TotRmsAbvGrd   1080 non-null   int64
 25  GarageYrBlt    1080 non-null   float64
 26  GarageFinish   1080 non-null   int32
 27  GarageCars     1080 non-null   int64
 28  GarageArea     1080 non-null   int64
 29  GarageQual     1080 non-null   int32
 30  GarageCond     1080 non-null   int32
 31  OpenPorchSF    1080 non-null   int64
 32  EnclosedPorch  1080 non-null   int64
 33  3SsnPorch      1080 non-null   int64
 34  ScreenPorch    1080 non-null   int64
 35  PoolArea       1080 non-null   int64
 36  MiscVal        1080 non-null   int64
 37  YrSold         1080 non-null   int64
 38  SalePrice      1080 non-null   int64
dtypes: float64(1), int32(15), int64(23)
memory usage: 274.2 KB
```

- **Mathematical/Analytics modeling done**

  - First of all we have transforming the given dataset from .CSV files into the Data Frame by the command,

    df=pd.read_csv("Housing _train.csv").
  - Before that we have importing the required libraries and are listed below.

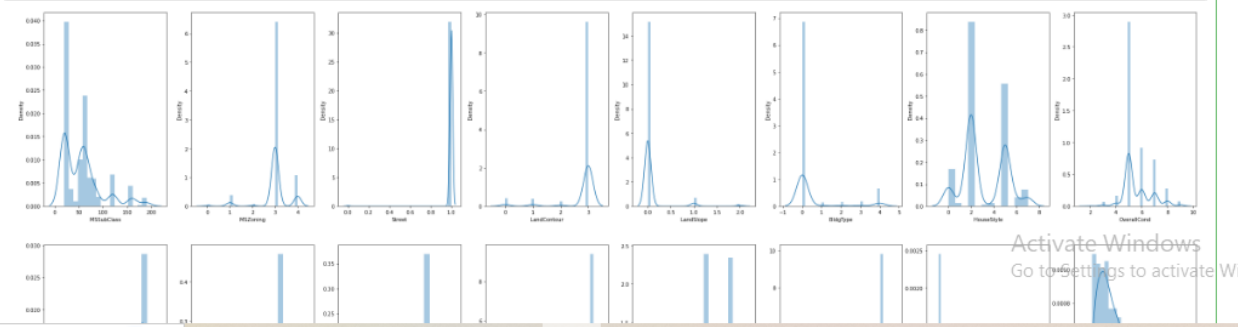
    1. **Pandas** 2. **Numpy** 3. **Sklearn** etc;
  - After that we have splitted the dataset into Independent and Dependent variables, By **Drop method** after knowing the column names.
  - **Describe method** shows the Statistical summary of the given dataset and it's include the **total count, mean, Std Deviation, minimum and maximum value, 1st 2nd 3rd quartile.**
  - From quartiles we can found **Inter-Quartile range, Quartile range (**if necessary).
  - After some preprocessing we will know the correlation of the variables by using the function **.corr() in the suffix of dataframe name,** so that we will come to know the positive and negative correlation between the variables to understand the necessary variable which influence the house price etc.

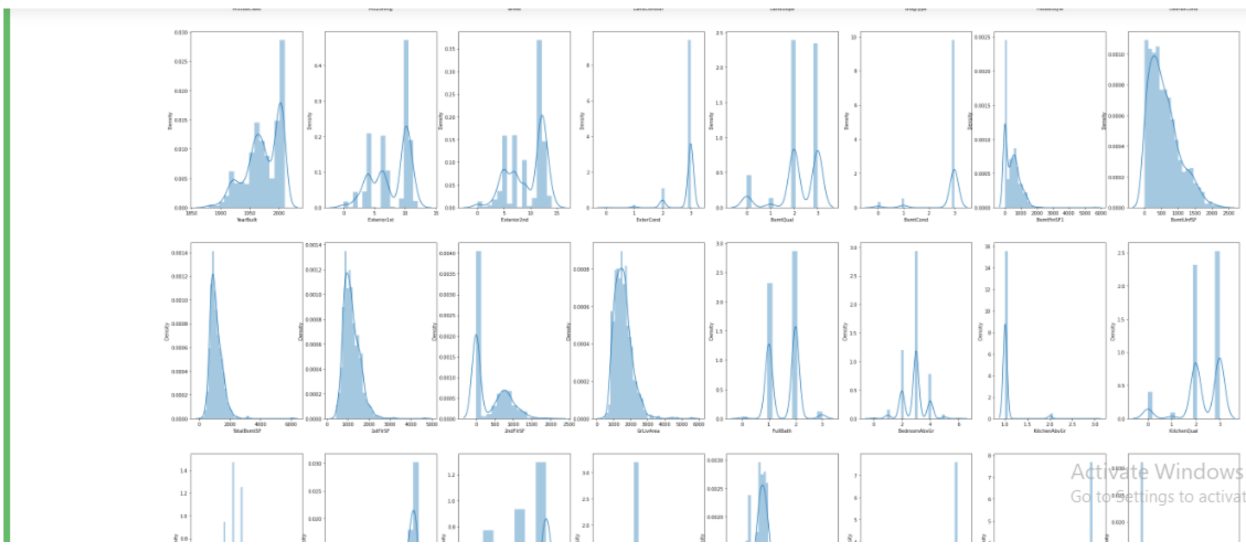
Model Development and Evaluation

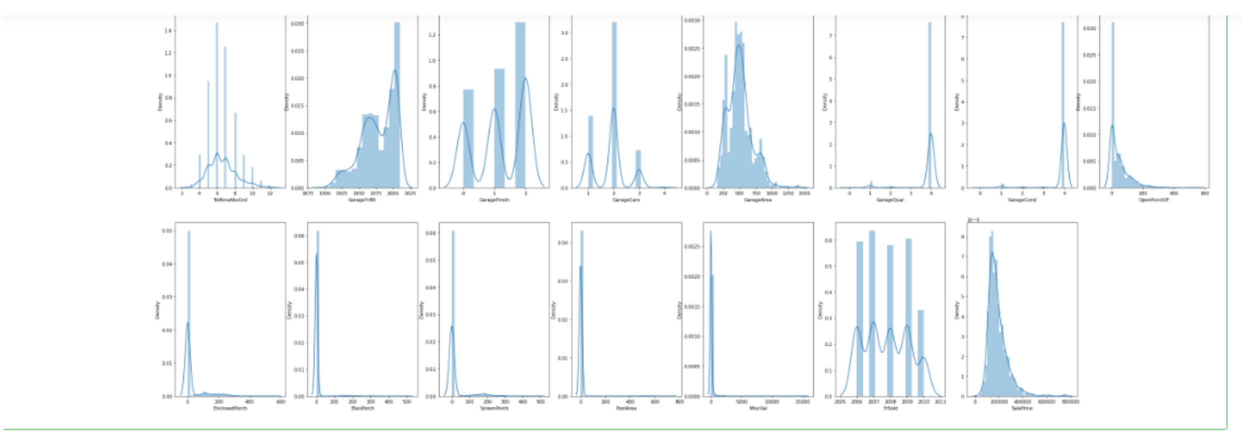
- **VISUALIZATIONS :-**

  1. Univariate Analysis

```python
# Univariate Analysis
collist=df_train.columns.values
ncol=8
nrow=5
plt.figure(figsize=(40,40))
for i in range(0,len(collist)):
    plt.subplot(nrow,ncol,i+1)
    sns.distplot(df_train[collist[i]])
```

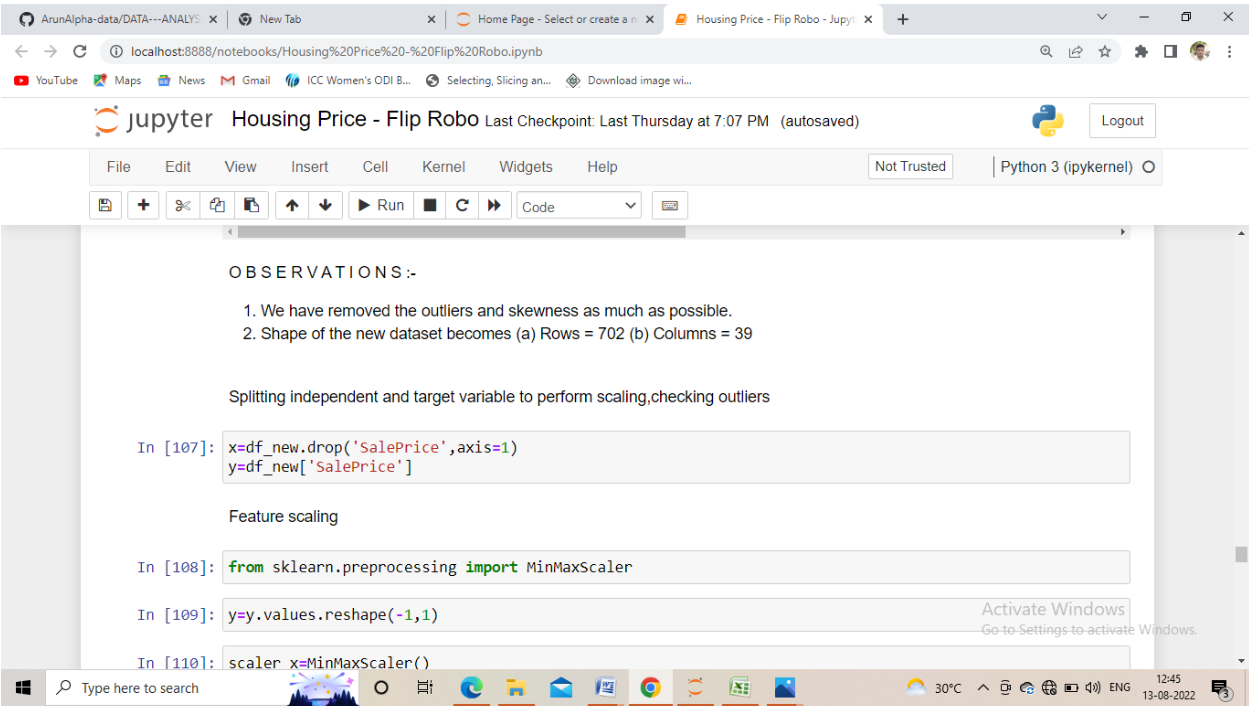






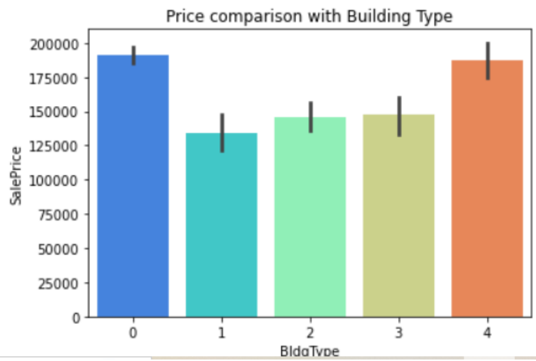
The above plot shows that more number of columns are skewed.Therefore let's see the skewness through the datas

2. Bivariate Analysis

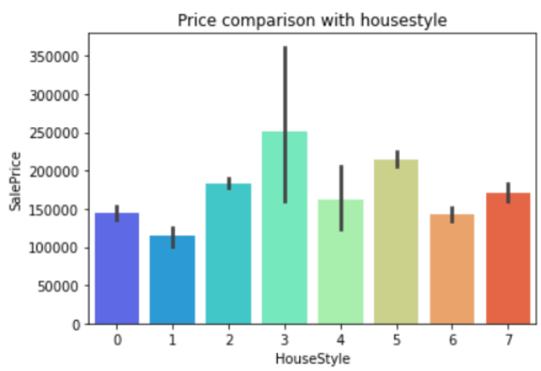


1 = Graval. Therefore gravel Street prices are higher than pavement Street prices

In [61]:
```
sns.barplot(x='BldgType',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison wit
plt.show()
```

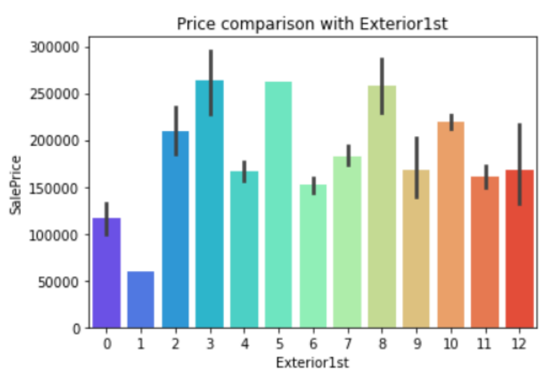


In [63]:
```
sns.barplot(x='HouseStyle',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison w
plt.show()
```



3-SplitLevel and 5-one and one-half storey 2nd level unfinished housestyle have the higher sale price

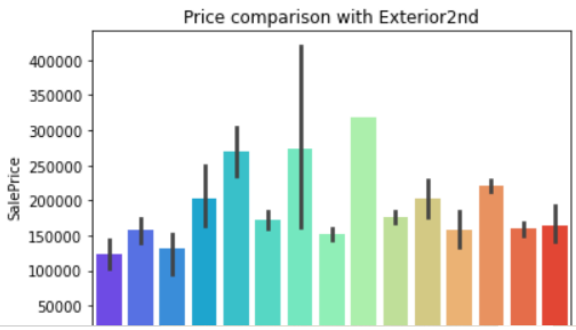
In [64]:
```
sns.barplot(x='Exterior1st',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison
plt.show()
```



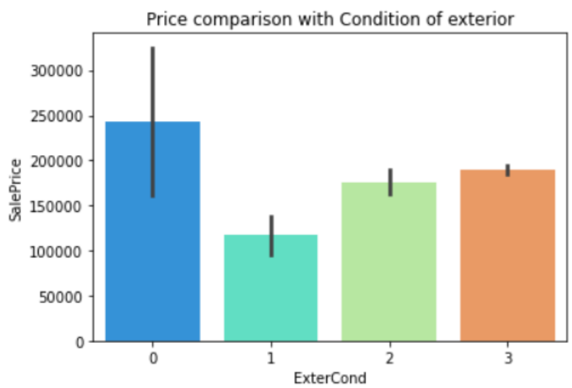
"Wood Siding" "Wood Shingles" AND "CEMENT BOARD" EXTERIOR COVER HOUSES HAVE THE HIGHER SALE PRICE

"Wood Siding" "Wood Shingles" AND "CEMENT BOARD" EXTERIOR COVER HOUSES HAVE THE HIGHER SALE PRICE

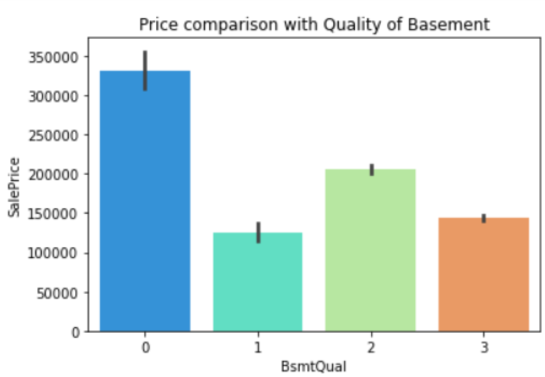
In [65]:
```
sns.barplot(x='Exterior2nd',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison
plt.show()
```



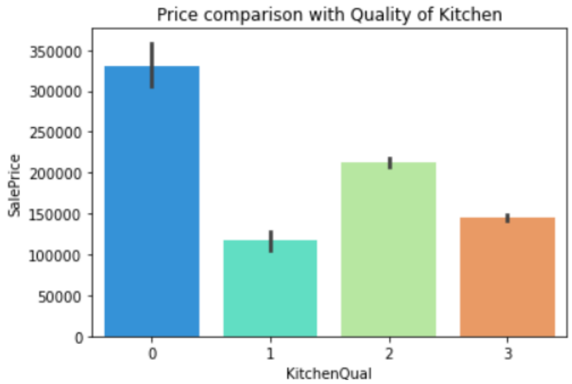
In [66]:
```
sns.barplot(x='ExterCond',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison wi
plt.show()
```



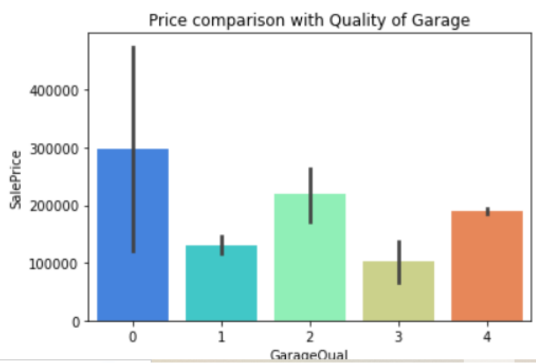
In [67]:
```
sns.barplot(x='BsmtQual',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison wit
plt.show()
```



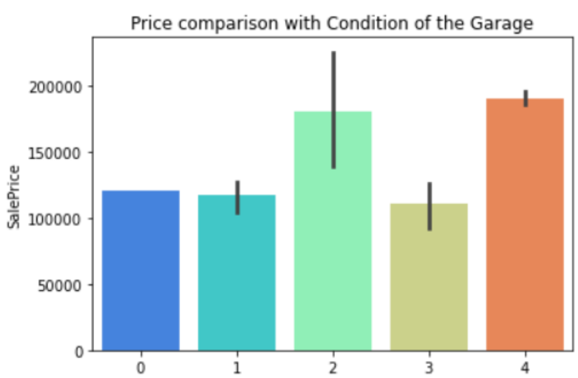
In [69]:
```
sns.barplot(x='KitchenQual',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison
plt.show()
```



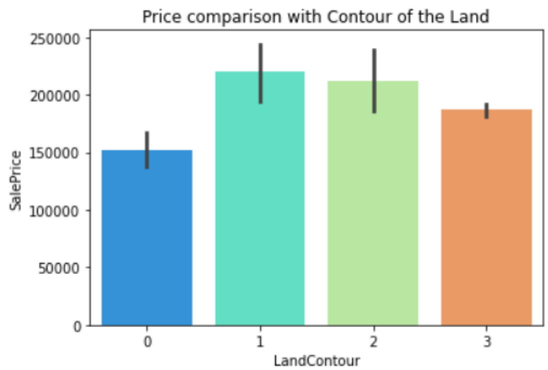
In [70]:
```
sns.barplot(x='GarageQual',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison w
plt.show()
```



In [72]:
```
sns.barplot(x='GarageCond',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison w
plt.show()
```

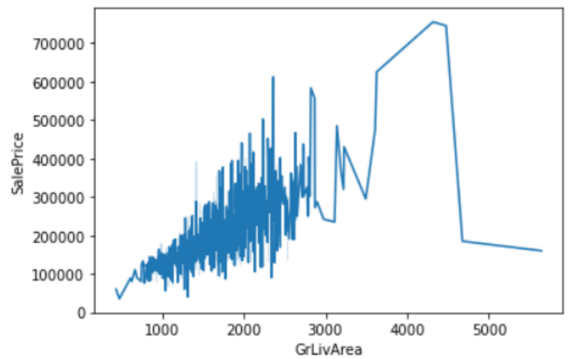


In [76]:
```
sns.barplot(x='LandContour',y='SalePrice',data=df_train,palette='rainbow').set_title('Price comparison
plt.show()
```



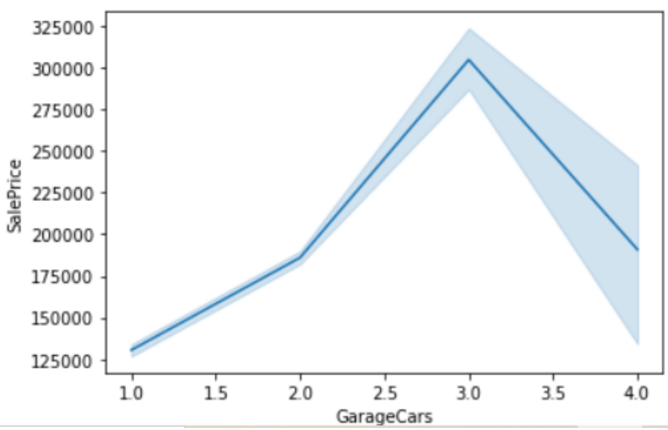
In [78]:
```
sns.lineplot(x='GrLivArea',y='SalePrice',data=df_train,palette='Rainbow')
```

Out[78]: <AxesSubplot:xlabel='GrLivArea', ylabel='SalePrice'>



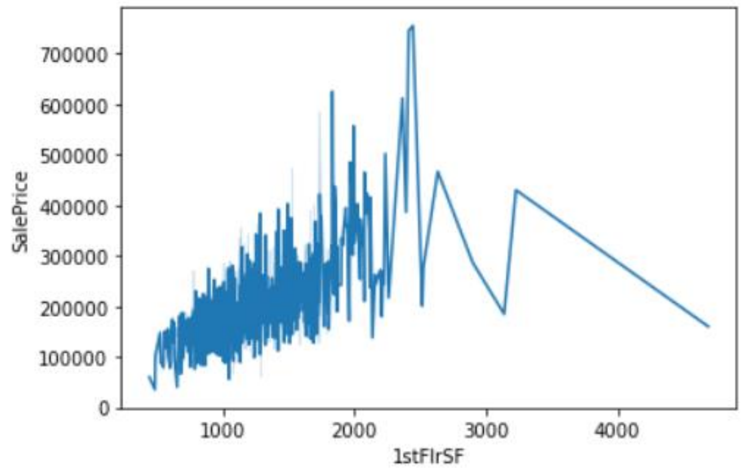
In [79]:
```
sns.lineplot(x='GarageCars',y='SalePrice',data=df_train,palette='Rainbow')
```

Out[79]: <AxesSubplot:xlabel='GarageCars', ylabel='SalePrice'>



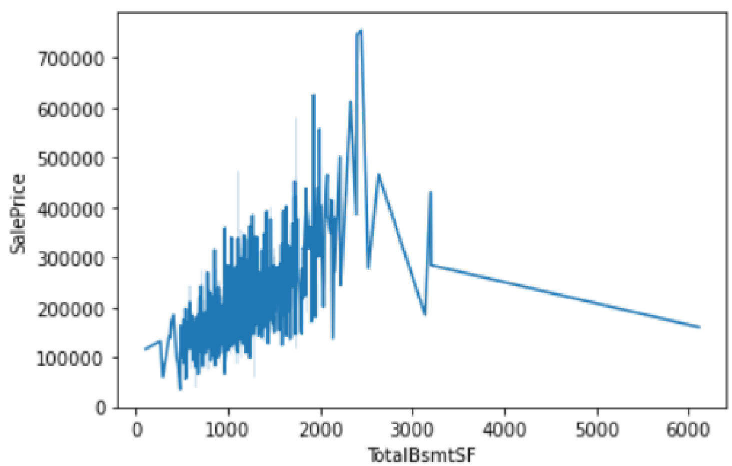
```
In [80]: sns.lineplot(x='1stFlrSF',y='SalePrice',data=df_train,palette='Rainbow')
```

```
Out[80]: <AxesSubplot:xlabel='1stFlrSF', ylabel='SalePrice'>
```

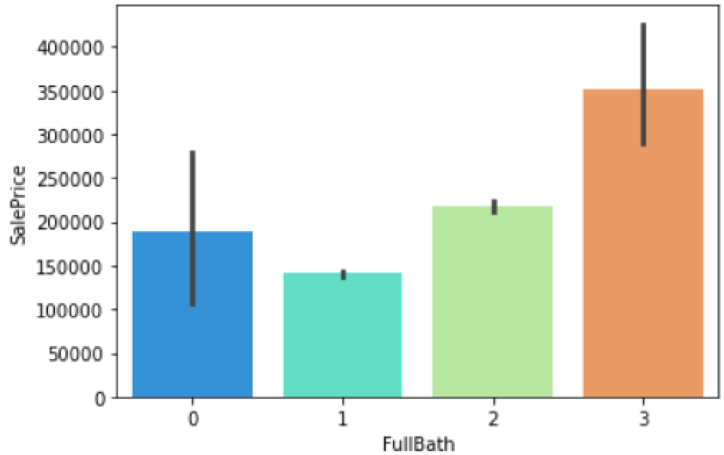



```
In [81]: sns.lineplot(x='TotalBsmtSF',y='SalePrice',data=df_train,palette='Rainbow')
```

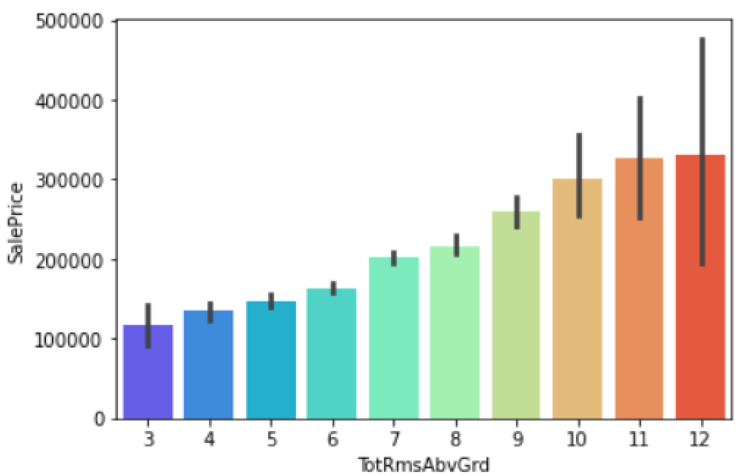
```
Out[81]: <AxesSubplot:xlabel='TotalBsmtSF', ylabel='SalePrice'>
```



```
In [85]: sns.barplot(x='FullBath',y='SalePrice',data=df_train,palette='rainbow')
Out[85]: <AxesSubplot:xlabel='FullBath', ylabel='SalePrice'>
```




```
In [86]: sns.barplot(x='TotRmsAbvGrd',y='SalePrice',data=df_train,palette='rainbow')
Out[86]: <AxesSubplot:xlabel='TotRmsAbvGrd', ylabel='SalePrice'>
```



3. Multivariate Analysis :-

In [92]:

```
corr_matrix=df_train.corr()
corr_matrix['SalePrice'].sort_values(ascending=False)
```

Out[92]:

```
SalePrice        1.000000
GrLivArea        0.705625
GarageCars       0.622357
GarageArea       0.599835
1stFlrSF         0.586065
TotalBsmtSF      0.579411
FullBath         0.557725
TotRmsAbvGrd     0.541545
YearBuilt        0.496005
GarageYrBlt      0.470217
OpenPorchSF      0.356271
BsmtFinSF1       0.340220
2ndFlrSF         0.315742
BsmtUnfSF        0.191700
HouseStyle       0.182870
BedroomAbvGr     0.152119
GarageCond       0.151381
```



```
GarageCond       0.151381
Exterior1st      0.104592
GarageQual       0.102456
PoolArea         0.101854
Exterior2nd      0.099962
ExterCond        0.094413
ScreenPorch      0.082817
3SsnPorch        0.058384
BsmtCond         0.047767
Street           0.044146
LandSlope        0.027086
LandContour      0.021908
MiscVal         -0.016295
YrSold          -0.042647
MSSubClass      -0.057658
BldgType        -0.059403
OverallCond     -0.102803
KitchenAbvGr    -0.112070
EnclosedPorch   -0.112289
MSZoning        -0.133894
```

```
ExterCond        0.094413
ScreenPorch      0.082817
3SsnPorch        0.058384
BsmtCond         0.047767
Street           0.044146
LandSlope        0.027086
LandContour      0.021908
MiscVal         -0.016295
YrSold          -0.042647
MSSubClass      -0.057658
BldgType        -0.059403
OverallCond     -0.102803
KitchenAbvGr    -0.112070
EnclosedPorch   -0.112289
MSZoning        -0.133894
GarageFinish    -0.507284
KitchenQual     -0.620717
BsmtQual        -0.623149
Name: SalePrice, dtype: float64
```

Observations :-

- From Univariate analysis we can clearly see that only few columns doesn’t have the skewness. Therefore we should remove that skewness using proper transformation method.
- From Bivariate analysis we can see the relation between the Saleprice and the other variable
    1. RM - Residential Medium Density and RH - Residential High Density have the higher sale price.
    2. 1 = Graval. Therefore gravel Street prices are higher than pavement Street prices.
    3. 1fam = Single family detached and 2fmcon = Two family Conversion(originally built as one-family dwelling) building types have the higher sale price.
    4. 3-SplitLevel and 5-one and one-half storey 2nd level unfinished housestyle have the higher sale price.

5. "Wood Siding" "Wood Shingles" AND "CEMENT BOARD" EXTERIOR COVER HOUSES HAVE THE HIGHER SALE PRICE.
6. "Brick Face" , "Wood Shingles" AND "Plywood" Exterior covering on house have the higher sale price.
7. AVERAGE/TYPICAL present condition of exterior material have the higher sale price.
8. Good Quality of height of the basement have the higher sale price.
9. GOOD condition of the basement have the higher sale price.

10. TYPICAL/AVERAGE Quality of the Kitchen have the higher Sale Price.

11. AVERAGE/TYPICAL quality of the garage have the higher sale price.

12. GOOD AND EXCELLENT condition of the garage have the higher sale price.

13. UNFINISHED INTERIOR of the Garage have the higher sale price.

14. Bnk - Quick AND Significant rise form the street grade to building, HLS - Significant slope from side to side,These Land Contour have the Higher Sale Price.

15. (a) GrLivArea - LIVING AREA Square Feet above Ground | GarageCars = Size of Garage in Car Capacity | TotRmsAbvGrd = TOTAL Number OF Rooms ABOVE GROUND | FullBath = FULL BATHROOMS ABOVE GROUND.

(b) IF these variables increases then Value of the houses also increses.

1STFlrSF - 1st floor square feet | TotalBsmtSF - TOTAL BASEMENT AREA IN SQUARE FEET

(C) THESE Variables has the steep increse in square feet upto 2000 - 2500 BUT after that there is a SLUMP in the Sale price of the houses

(D) This shows that People are not preffering the LARGER AREAS IN BASEMENT AND 1ST FLR AREA.

- From this correlation values we can clearly see that **“GrLivArea”** has **strong positive correlation** with **SalesPrice** follwed by “**GarageCars**”,”**GarageArea”,”1stFlrSF”**,”**TotalBsmtSF**”,”**FullBath”, ”TotRmsAbvGrd”**. ”**LandContour**” has weak positive correlation with “**SalesPrice**”.
- **” MiscVal”** has strong negative correlation with **SalesPrice** and **“BsmtQual”** has weak negative correlation with **SalesPrice**.

- **.describe()** mini statistical analysis.

```
In [93]: df_train.describe()
Out[93]:
```

	MSSubClass	MSZoning	Street	LandContour	LandSlope	BldgType	HouseStyle	OverallCond	YearBuilt
count	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000
mean	55.888889	3.009259	0.997222	2.785185	0.063889	0.456481	3.101852	5.617593	1972.794444
std	40.786938	0.629744	0.052656	0.690303	0.283287	1.189993	1.902341	1.094711	29.598649
min	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	1880.000000
25%	20.000000	3.000000	1.000000	3.000000	0.000000	0.000000	2.000000	5.000000	1955.000000
50%	50.000000	3.000000	1.000000	3.000000	0.000000	0.000000	2.000000	5.000000	1975.000000
75%	70.000000	3.000000	1.000000	3.000000	0.000000	0.000000	5.000000	6.000000	2001.000000
max	190.000000	4.000000	1.000000	3.000000	2.000000	4.000000	7.000000	9.000000	2010.000000

8 rows × 39 columns

OUR next step is to remove the skewness and outliers from our dataset, for this we are using ”**LOG transformation**” method for removing skewness and and “**ZSCORE”** for removing outliers.

In [87]:
```
df_train.skew().sort_values(ascending=False)
```

Out[87]:
```
MiscVal          22.361068
PoolArea         12.727509
3SsnPorch         9.670196
KitchenAbvGr      5.953687
LandSlope         4.824263
ScreenPorch       3.921742
EnclosedPorch     3.163829
TotalBsmtSF       2.520182
BldgType          2.404457
OpenPorchSF       2.251192
SalePrice         2.030236
BsmtFinSF1        1.873921
1stFlrSF          1.546817
GrLivArea         1.511250
MSSubClass        1.412610
BsmtUnfSF         0.910951
GarageArea        0.824807
OverallCond       0.804600
2ndFlrSF          0.769988
TotRmsAbvGrd      0.627432
```

```
GarageArea        0.824807
OverallCond       0.804600
2ndFlrSF          0.769988
TotRmsAbvGrd      0.627432
HouseStyle        0.250283
GarageCars        0.184877
YrSold            0.129521
FullBath          0.008685
BedroomAbvGr     -0.098087
GarageFinish     -0.331681
Exterior1st      -0.539605
Exterior2nd      -0.584295
YearBuilt        -0.649796
GarageYrBlt      -0.671329
BsmtQual         -1.283297
KitchenQual      -1.400961
MSZoning         -1.791581
ExterCond        -3.194854
LandContour      -3.212075
BsmtCond         -3.324956
GarageQual       -4.381879
GarageCond       -5.274506
Street          -18.920806
```

In [95]:
```
df_train['MSSubClass']=np.log(df_train['MSSubClass'])

df_train['OverallCond']=np.log(df_train['OverallCond'])

df_train['TotalBsmtSF']=np.log(df_train['TotalBsmtSF'])

df_train['1stFlrSF']=np.log(df_train['1stFlrSF'])
```

In [96]:
```
df_train['YrSold']=np.log(df_train['YrSold'])
```

In [97]:
```
df_train['GarageArea']=np.log(df_train['GarageArea'])
```

Let's Check For Skewness

```
In [98]: df_train.skew().sort_values(ascending=False)

Out[98]: MiscVal          22.361068
         PoolArea         12.727509
         3SsnPorch         9.670196
         KitchenAbvGr      5.953687
         LandSlope         4.824263
         ScreenPorch       3.921742
         EnclosedPorch     3.163829
         BldgType          2.404457
         OpenPorchSF       2.251192
         SalePrice         2.030236
         BsmtFinSF1        1.873921
         GrLivArea         1.511250
         BsmtUnfSF         0.910951
         2ndFlrSF          0.769988
         TotRmsAbvGrd      0.627432
         HouseStyle        0.250283
         MSSubClass        0.207909
         GarageCars        0.184877
```

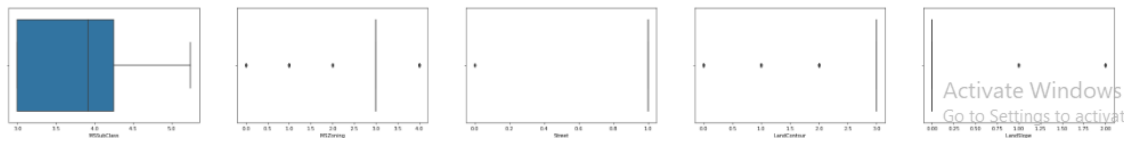
1. We haven't removed the negative skewness as it affects the dataset and we have removed only a particular skewness because of comparing the skewness variables with correlation and the variables with negative correlation and has high skewness has been removed from our training dataset.

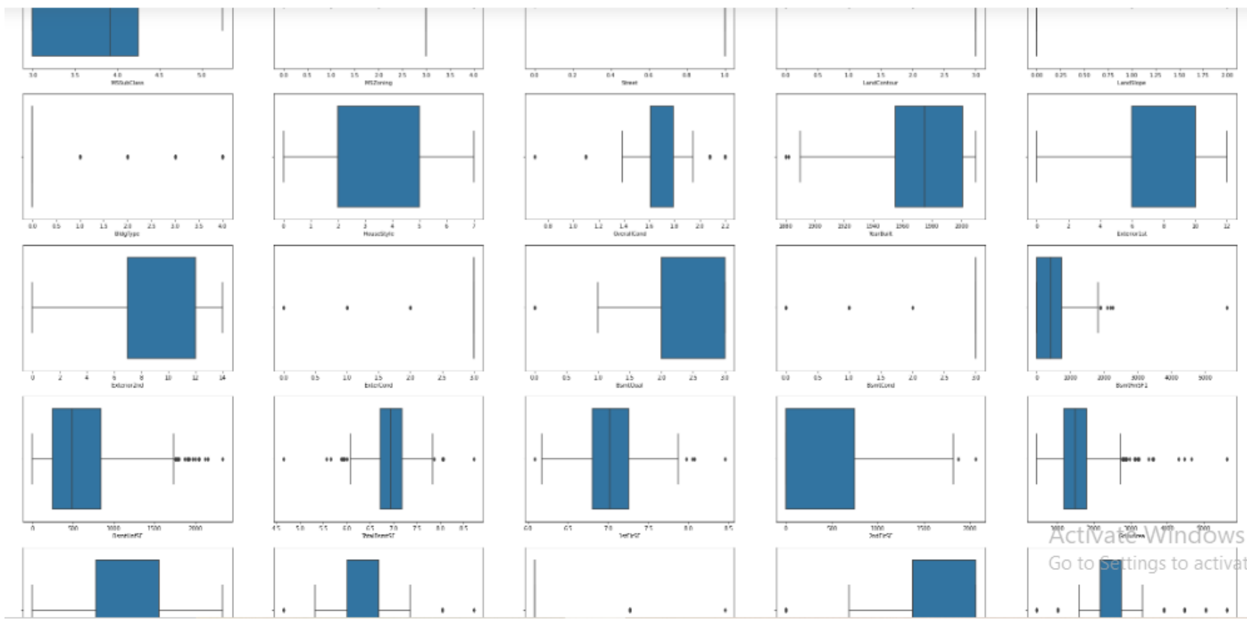
Plotting Outliers :-

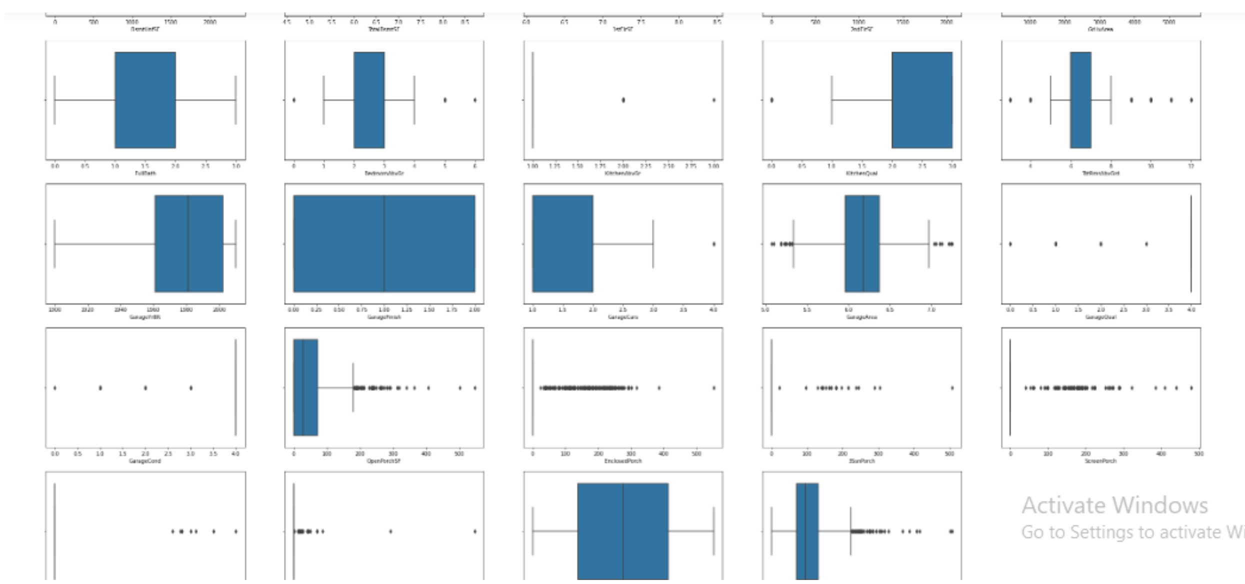
Plotting Outliers

In [100]:
```
import seaborn as sns
```

In [101]:
```
collist=df_train.columns.values
ncol=5
nrow=8
plt.figure(figsize=(40,40))
for i in range(0,len(collist)):
    plt.subplot(nrow,ncol,i+1)
    sns.boxplot(df_train[collist[i]])
```







1. Housestyle , FullBath , GarageFinish , YrSold , GarageCars, MSSubclass , GarageFinish, GarageYrBlt, KitchenQual, 2ndFlrSF, Exterior2nd, BsmtQual, Exterior1st, YearBuilt, MSSUBCLASS these columns doesn't have the outliers.

2. Let's remove outliers in upcoming steps.

Removing Outliers :-

Let's Quantify

In [102]:
```
from scipy.stats import zscore
```

In [103]:
```
z=np.abs(zscore(df_train))
z
```

Out[103]:

	MSSubClass	MSZoning	Street	LandContour	LandSlope	BldgType	HouseStyle	OverallCond	YearBuilt	Exterior1st	...
0	1.448155	0.014710	0.052778	0.311333	0.225631	2.979143	0.579477	0.511656	0.108351	0.239045	...
1	1.129558	0.014710	0.052778	0.311333	3.305988	0.383778	0.579477	0.439171	0.094455	1.095160	...
2	0.450959	0.014710	0.052778	0.311333	0.225631	0.383778	0.998258	0.511656	0.784370	0.572596	...
3	1.129558	0.014710	0.052778	0.311333	0.225631	0.383778	0.579477	0.439171	0.142152	0.239045	...
4	1.129558	0.014710	0.052778	0.311333	0.225631	0.383778	0.579477	1.243083	0.142152	1.573250	...
...	...	...	...	...	...	...	...	...	...	...	...
1162	0.546236	0.014710	0.052778	0.311333	0.225631	0.383778	0.579477	3.175666	1.987310	0.428058	...
1163	1.129558	0.014710	0.052778	0.311333	0.225631	0.383778	0.579477	0.511656	0.195858	0.572596	...
1165	1.862028	0.014710	0.052778	0.311333	0.225631	2.138413	0.998258	0.439171	0.108351	0.239045	...

In [104]:
```
threshold=3
df_new=df_train[(z<3).all(axis=1)]
df_new
```

Out[104]:

	MSSubClass	MSZoning	Street	LandContour	LandSlope	BldgType	HouseStyle	OverallCond	YearBuilt	Exterior1st	...
0	4.787492	3	1	3	0	4	2	1.609438	1976	7	...
2	4.094345	3	1	3	0	0	5	1.609438	1996	6	...
3	2.995732	3	1	3	0	0	2	1.791759	1977	7	...
4	2.995732	3	1	3	0	0	2	1.945910	1977	3	...
5	4.094345	3	1	3	0	0	5	1.609438	2006	10	...
...	...	...	...	...	...	...	...	...	...	...	...
1158	2.995732	3	1	3	0	0	2	1.791759	1977	7	...
1161	4.094345	3	1	3	0	0	5	1.609438	1999	10	...
1163	2.995732	3	1	3	0	0	2	1.609438	1967	6	...
1165	5.075174	3	1	3	0	3	5	1.791759	1976	7	...

702 rows × 39 columns

```
In [105]: print(df_train.shape)
          print(df_new.shape)

(1080, 39)
(702, 39)
```

- After removing the outliers our dataset becomes
  1. Rows = 702
  2. Columns = 39
- Scaling the dataset is being used to fit the values equally so that we can perform our analysis better.
- Objective behind using this Scaling techniques is that the range of data's in our dataset is varying high. Example if we take the target variable the range starts in lakhs and ends in lakhs but if we see other independent variable its range is within 20.
- Here **MIN-MAX scaling** technique is used because it's a Regression problem not a classification.

Splitting independent and target variable to perform scaling,checking outliers

```
In [107]: x=df_new.drop('SalePrice',axis=1)
          y=df_new['SalePrice']
```

Feature scaling

```
In [108]: from sklearn.preprocessing import MinMaxScaler

In [109]: y=y.values.reshape(-1,1)

In [110]: scaler_x=MinMaxScaler()
```


- Testing of Identified Approaches (Algorithms)

  Train_test_split() is used to split the dataset into train and test for model building.

  Test_size=0.20 and random_state = 1.

- **Run and Evaluate selected models :-**

Models using here are listed below

1. Linear Regression.
2. Random Forest Regressor.
3. Decision Tree Regressor.
4. Support Vector Regressor
5. SGD Regressor.

Metrics which are used here is as follows :-

1. Mean Squared Error.
2. Mean Absolute Error.
3. R2 score.
4. Root Mean Squared Error.

```
print('Mean Squared Errors :',mean_squared_error(y_test,y_pred1))
print('Mean Absolute Errors :',mean_absolute_error(y_test,y_pred1))
print('Root Mean Squared Error :',np.sqrt(mean_squared_error(y_test,y_pred1)))
```

```
Errors :
Mean Squared Errors : 0.005988964108216794
Mean Absolute Errors : 0.05438965788401245
Root Mean Squared Error : 0.0773883977623054
```

In [130]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import r2_score
```

In [131]:
```
lr=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
lr.fit(x_train,y_train)
y_pred1=lr.predict(x_test)
r2score=r2_score(y_test,y_pred1)
cvscore=cross_val_score(LinearRegression(),x_train,y_train,cv=5).mean()
print( f"Accuracy={r2score*100},cross_value_score={cvscore*100},and difference={(r2score*100)-(cvscore*
```

```
Accuracy=84.03239233377249,cross_value_score=85.8470046773879,and difference=-1.8146123436154085
```

```
In [130]: from sklearn.model_selection import cross_val_score
          from sklearn.metrics import r2_score
```

```
In [131]: lr=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
          lr.fit(x_train,y_train)
          y_pred1=lr.predict(x_test)
          r2score=r2_score(y_test,y_pred1)
          cvscore=cross_val_score(LinearRegression(),x_train,y_train,cv=5).mean()
          print( f"Accuracy={r2score*100},cross_value_score={cvscore*100},and difference={(r2score*100)-(cvscore*

          Accuracy=84.03239233377249,cross_value_score=85.8470046773879,and difference=-1.8146123436154085
```

Here, with Linear regressor model has error pretty much close to 0. Therefore it is considered has very good model.Let's see the other model performance

2. Random Forest Regressor model

```
In [132]: from sklearn.ensemble import RandomForestRegressor

In [133]: rf=RandomForestRegressor()

In [134]: rf.fit(x_train,y_train)

Out[134]: RandomForestRegressor()

In [135]: y_pred2=rf.predict(x_test)

In [136]: comp_df=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred2})

In [137]: print('Errors :')
          print('Mean Squared Errors :',mean_squared_error(y_test,y_pred2))
          print('Mean Absolute Errors :',mean_absolute_error(y_test,y_pred2))
```



```
In [137]: print('Errors :')
          print('Mean Squared Errors :',mean_squared_error(y_test,y_pred2))
          print('Mean Absolute Errors :',mean_absolute_error(y_test,y_pred2))
          print('Root Mean Squared Error :',np.sqrt(mean_squared_error(y_test,y_pred2)))

          Errors :
          Mean Squared Errors : 0.007214578445548584
          Mean Absolute Errors : 0.05287240239006035
          Root Mean Squared Error : 0.08493867461615223

In [138]: r2score=r2_score(y_test,y_pred2)
          cvscore=cross_val_score(rf,x_train,y_train,cv=5).mean()
          print( f"Accuracy={r2score*100},cross_value_score={cvscore*100},and difference={(r2score*100)-(cvscore*

          Accuracy=80.76469385787624,cross_value_score=86.02803412392872,and difference=-5.263340266052481
```

3. Decision Tree Regressor Model

```
In [139]: from sklearn.tree import DecisionTreeRegressor

In [140]: dt=DecisionTreeRegressor(max_depth=4,min_samples_leaf=0.1,random_state=1)

In [141]: dt.fit(x_train,y_train)

Out[141]: DecisionTreeRegressor(max_depth=4, min_samples_leaf=0.1, random_state=1)

In [142]: y_pred3=dt.predict(x_test)

In [143]: comp_df=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred3})

In [144]: comp_df
```



```
In [145]: print('Errors :')
          print('Mean Squared Errors :',mean_squared_error(y_test,y_pred3))
          print('Mean Absolute Errors :',mean_absolute_error(y_test,y_pred3))
          print('Root Mean Squared Error :',np.sqrt(mean_squared_error(y_test,y_pred3)))

Errors :
Mean Squared Errors : 0.012850587803869886
Mean Absolute Errors : 0.07610273877738784
Root Mean Squared Error : 0.11336043314962185

In [146]: r2score=r2_score(y_test,y_pred3)
          cvscore=cross_val_score(dt,x_train,y_train,cv=5).mean()
          print( f"Accuracy={r2score*100},cross_value_score={cvscore*100},and difference={(r2score*100)-(cvscore*

Accuracy=65.73812421899814,cross_value_score=66.48509965753698,and difference=-0.7469754385388399
```


4. Support Vector Regressor

```
In [147]: from sklearn.svm import SVR

In [148]: svr=SVR()

In [149]: svr.fit(x_train,y_train)

Out[149]: SVR()

In [150]: y_pred4=svr.predict(x_test)

In [151]: comp_df=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred4})

In [152]: comp_df
```



```
r2score=r2_score(y_test,y_pred4)

cvscore=cross_val_score(svr,x_train,y_train,cv=5).mean()

print( f"Accuracy={r2score*100},cross_value_score={cvscore*100},and difference={(r2score*100)-(cvscore*100)}")

Accuracy=85.32513356452107,cross_value_score=84.16648161246837,and difference=1.1586519520526934
```

5. SGD Regressor Model

```
In [155]: from sklearn.linear_model import SGDRegressor
```

```
In [156]: sgd=SGDRegressor()
```

```
In [157]: sgd.fit(x_train,y_train)
```

```
Out[157]: SGDRegressor()
```

```
In [158]: y_pred5=sgd.predict(x_test)
```

```
In [159]: comp_df=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred5})
```



```
In [160]: comp_df
```

```
In [161]: print('Errors :')
          print('Mean Squared Error :',mean_squared_error(y_test,y_pred5))
          print('Mean Absolute Error :',mean_absolute_error(y_test,y_pred5))
          print('Root Mean Sqquared Error :',np.sqrt(mean_squared_error(y_test,y_pred5)))
```

```
Errors :
Mean Squared Error : 0.009206508283809494
Mean Absolute Error : 0.06405809281416541
Root Mean Sqquared Error : 0.09595055124286413
```

```
In [162]: r2score=r2_score(y_test,y_pred5)
          cvscore=cross_val_score(sgd,x_train,y_train,cv=5).mean()
          print( f"Accuracy={r2score*100},cross_value_score={cvscore*100},and difference={(r2score*100)-(cvscore*
```

```
Accuracy=75.45386654596027,cross_value_score=77.26368606154642,and difference=-1.809819515586156
```

SAVING THE MODEL

SAVING THE MODEL

```
In [178]: import joblib
```

```
In [179]: joblib.dump(svr,"Housing_trained.pkl")
```

```
Out[179]: ['Housing_trained.pkl']
```

Predictions on Test Data

```
In [243]: df_pred=pd.DataFrame({"Sale Price" : predictions})
          df_pred
Out[243]:
```

	Sale Price
0	0.580052
1	0.408699
2	0.465011
3	0.353451
4	0.452521
5	0.333407
6	0.544501
7	0.440617
8	0.394011
9	0.245388


```
In [239]: fitted_model=pickle.load(open("Housing_price_trained.pkl","rb"))

In [240]: fitted_model

Out[240]: SVR()

In [241]: predictions=fitted_model.predict(df_new1)

In [242]: predictions

Out[242]: array([0.58005176, 0.40869863, 0.4650105 , 0.35345149, 0.45252123,
                 0.33340681, 0.54450098, 0.44061688, 0.3940112 , 0.24538751,
                 0.29885233, 0.30051707, 0.37720414, 0.49886785, 0.32912272,
                 0.2821844 , 0.37287342, 0.39305815, 0.3145974 , 0.30204173,
                 0.29095602, 0.2834587 , 0.26886452, 0.36270839, 0.31758178,
                 0.33645929, 0.27899304, 0.35254563, 0.36668561, 0.45178938,
                 0.30754985, 0.20788522, 0.34216989, 0.41210288, 0.29135542,
                 0.33354456, 0.31354085, 0.25323474, 0.56010926, 0.39758131,
                 0.37776264, 0.3117685 , 0.32153855, 0.30766427, 0.31347214,
                 0.38036021, 0.60759571, 0.30969119, 0.39489564, 0.30518325,
                 0.23012896, 0.46073284, 0.34655812, 0.32176385, 0.39310412,
```


Conclusion Remarks :-

1. We could see that using "Support Vector Regression" gives a most promising predictions when compared with all other Models.
2. Variables which influence the Sales Prices are “**GrLivArea”, “GarageArea”** , “**GarageCars”, “1stFlrSF”, “TotalBsmtSF” ,“FullBath” , “TotRmsAbvGrd” , “LandContour”.**