FLIP ROBO

PROJECT REPORT ON :

“PFA Housing Project”

SUBMITTED BY:

CHRIS CHHOTAI

ACKNOWLEDGMENT

I'd want to express my heartfelt gratitude to the "Flip Robo" team for providing me with the opportunity to work with such a beautiful dataset and for helping me develop my data analysis skills. And I'd like to express my heartfelt gratitude to Ms. Swati Mahaseth (SME Flip Robo), who has guided me through all of the challenges I've encountered while working on the project who has inspired me in many ways and encouraged me greatly with his wise words and unwavering support, resulting in a beautiful project.

Thank you so much to my "Data trained" academic team, who are the reason I am where I am now. Last but not least, my parents, who have supported me throughout my life. Thank you also to the many other people who have assisted me in completing the project, whether directly or indirectly.

1.INTRODUCTION

Business Problem Framing:

This is a real estate issue in which Surprise Housing, a US-based housing company, has opted to invest in the Australian market. Their plan is to buy houses in Australia for less than their market value and then sell them for a profit at a higher price. To accomplish so, this organisation employs data analytics to choose which properties to invest in.

The company has gathered information on previously sold houses in Australia, and using this information, they hope to determine the value of potential properties in order to determine whether or not they are acceptable for investment.

To determine the worth of properties, the company has provided data for us to analyse and extract information about features that are significant in predicting house prices. They seek a machine learning model that can predict house prices as well as the importance of each essential feature in house prediction, such as how and to what extent each variable influences house prices.

Conceptual Background of the Domain Problem:

Property value frequently rises with time in real estate, as demonstrated in numerous countries. One of the reasons for this is that the population is growing. The value of a property is also determined by its closeness, size, and location, as well as the target market for whom it is being sold. For instance, if the audience is primarily interested in commercial purposes. Then, in comparison to the property located in a remote location, the property located in a heavily populated area will be sold quickly and at a high price. Similarly, if the audience is solely interested in a place to live, property in a less populated region with a vast area and all services will sell for a higher price.

The business is seeking for potential properties to purchase in order to enter the market. We must use Machine Learning to create a model that will estimate the actual worth of potential properties and help us decide whether or not to invest in them.

Review of Literature:

Houses are one of the most basic necessities of every person on the planet, and hence the housing and real estate markets are one of the most important contributors to the global economy.Surprise Housing, a housing company located in the United States, has decided to enter the Australian market. The company employs data analytics to buy houses for less than their true value and then resell them for more. With its great weather, cosmopolitan cities, diverse natural landscapes and relaxed lifestyle, it’s no wonder that Australia remains a top pick for expats.

Living cost in Australia for one person: $2,835 per month. Average living expenses for a couple: $4,118 per month. Average monthly living expenses for a family of 4: $5,378. Australia currently has the 16th highest cost of living in the world, with the USA and UK well behind at 21st and 33rd place respectively. Sydney and Melbourne are popular choices for expats moving to Australia. House pricing in some of the top Australian cities:-

Sydney - median house price A$1,142,212

Adelaide- median house price A$542,947

Hobbart (smaller city)- median house price A$530,570.

Motivation for the Problem Undertaken:

To get a better understanding of real-world challenges where Machine Learning and Data Analysis can be used to assist organisations in various areas in making better decisions that will allow them to generate profit or avoid losses than would otherwise be achievable without the use of data. Real estate is one of these domainsHouses are a basic need for everyone on the planet, so the housing and real estate sector is one of the most important contributors to the global economy. It's a huge market with a lot of different companies operating in it. Data science is a critical tool for resolving difficulties in the area and assist businesses in increasing total income and profits by enhancing marketing techniques and focusing on changing trends in home sales and purchases. Machine learning approaches like as predictive modelling, market mix modelling, and recommendation systems are employed by housing firms to achieve their business objectives. One such housing company is the source of our difficulty.

2.Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem:

We examined the description or statistical summary of the data using describe, checked the correlation using corr, and visualised it using heatmap in this project. Then we plotted outliers and removed them using Z-Score.

In [8]:
```
#Lets get a general idea about the dataset
df.describe()
```

Out[8]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeck
count	1168.000000	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	...	1168.000
mean	724.136130	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	...	96.206
std	416.159877	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.664785	...	126.158
min	1.000000	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	...	0.000
25%	360.500000	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	...	0.000
50%	714.500000	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	...	0.000
75%	1079.500000	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	...	171.000
max	1460.000000	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.000

8 rows × 38 columns

From this statistical analysis we make some of the interpretations that,

- Maximum standard deviation of 8957.44 is observed in LotArea column.

- Maximum SalePrice of a house observed is 755000 and minimum is 34900.

- In the columns Id, MSSubclass, LotArea, MasVnrArea,

  BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, Miscval, salePrice mean is considerably greater than median so the columns are positively skewed.

- In the columns FullBath, BedroomAbvGr, Fireplaces, Garagecars, GarageArea, YrSold Median is greater than mean so the columns are negatively skewed.

- In the columns Id, MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, SalePrice there is considerable difference between the 75 percentile and maximum so outliers are present.

Data Sources and their formats:

The variable features of this problem statement are as :

- MSSubClass: Identifies the type of dwelling involved in the sale

- MSZoning: Identifies the general zoning classification of the sale

- LotFrontage: Linear feet of street connected to property

- LotArea: Lot size in square feet
- Street: Type of road access to property
- Alley: Type of alley access to property
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to various conditions
- Condition2: Proximity to various conditions (if more than one is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling

- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Evaluates the quality of the material on the exterior
- ExterCond: Evaluates the present condition of the material on the exterior
- Foundation: Type of foundation

- BsmtQual: Evaluates the height of the basement
- BsmtCond: Evaluates the general condition of the basement
- BsmtExposure: Refers to walkout or garden level walls
- BsmtFinType1: Rating of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Rating of basement finished area (if multiple types)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet

- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
- Kitchen: Kitchens above grade
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality (Assume typical unless deductions are warranted)
- Fireplaces: Number of fireplaces

- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet

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

Data Preprocessing Done:

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split,GridSearchCV,KFold,cross_val_score
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.feature_selection import RFE
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Loading and reading the data
df= pd.read_csv("train_data.csv")
df
```

Out[2]:

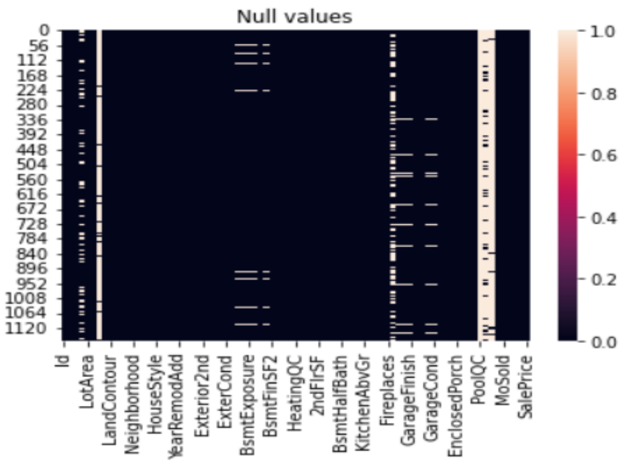
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

Feature Engineering has been used for cleaning of the data. Some unused columns have been deleted and even some columns have been bifurcated which was used in the prediction. We first done data cleaning. We first looked percentage of values missing in columns then we imputed missing values.

```
In [13]: # Let's check the missing values of top 30 columns
         df.isnull().sum().sort_values(ascending = False).head(30)
```

```
Out[13]: PoolQC          1161
         MiscFeature     1124
         Alley           1091
         Fence            931
         FireplaceQu      551
         LotFrontage      214
         GarageYrBlt       64
         GarageFinish      64
         GarageType        64
         GarageQual        64
         GarageCond        64
         BsmtExposure      31
         BsmtFinType2      31
         BsmtQual          30
         BsmtCond          30
         BsmtFinType1      30
         MasVnrType         7
         MasVnrArea         7
         Id                 0
         Functional         0
         Fireplaces         0
         KitchenQual        0
         KitchenAbvGr       0
         BedroomAbvGr       0
         HalfBath           0
         FullBath           0
         BsmtHalfBath       0
         BsmtFullBath       0
         TotRmsAbvGrd       0
         GarageCars         0
         dtype: int64
```

```
In [9]: sns.heatmap(df.isnull())
        plt.title('Null values')
        plt.show()
```



In [16]:
```python
# Let's check the percentage of missing values of each column

def missing_values_table(df):
    mis_val = df.isnull().sum()
    mis_val_percent = 100 * df.isnull().sum() / len(df)
    mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
    mis_val_table_ren_columns = mis_val_table.rename(
    columns = {0 : 'Missing Values', 1 : '% of Total Values'})
    mis_val_table_ren_columns = mis_val_table_ren_columns[
        mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
    '% of Total Values', ascending=False).round(1)
    print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
        "There are " + str(mis_val_table_ren_columns.shape[0]) +
            " columns that have missing values.")
    return mis_val_table_ren_columns
missing_values_table(df)
```

```
Your selected dataframe has 81 columns.
There are 18 columns that have missing values.
```

Out[16]:

	Missing Values	% of Total Values
PoolQC	1161	99.4
MiscFeature	1124	96.2
Alley	1091	93.4
Fence	931	79.7
FireplaceQu	551	47.2
LotFrontage	214	18.3

In [17]:
```python
# Let's fill the missing values in categorical columns as NA

columns = ["FireplaceQu", "GarageType", "GarageFinish", "GarageQual", "GarageCond", "BsmtExposure", "BsmtFinType2", "BsmtCond", '
df[columns] = df[columns].fillna('NA')
```

In [18]:
```python
# Let's fill the missing values in MasVnrType with None
df['MasVnrType'] = df['MasVnrType'].fillna('None')
```

In [19]:
```python
# Let's fill the missing values in GarageYrBlt with 0

df['GarageYrBlt'] = df['GarageYrBlt'].fillna('0')
```

In [20]:
```python
# Let's Imputing the missing values and replace it with the median

df['LotFrontage'].fillna(df['LotFrontage'].median(),inplace=True)
df['MasVnrArea'].fillna(df['MasVnrArea'].median(),inplace=True)
```

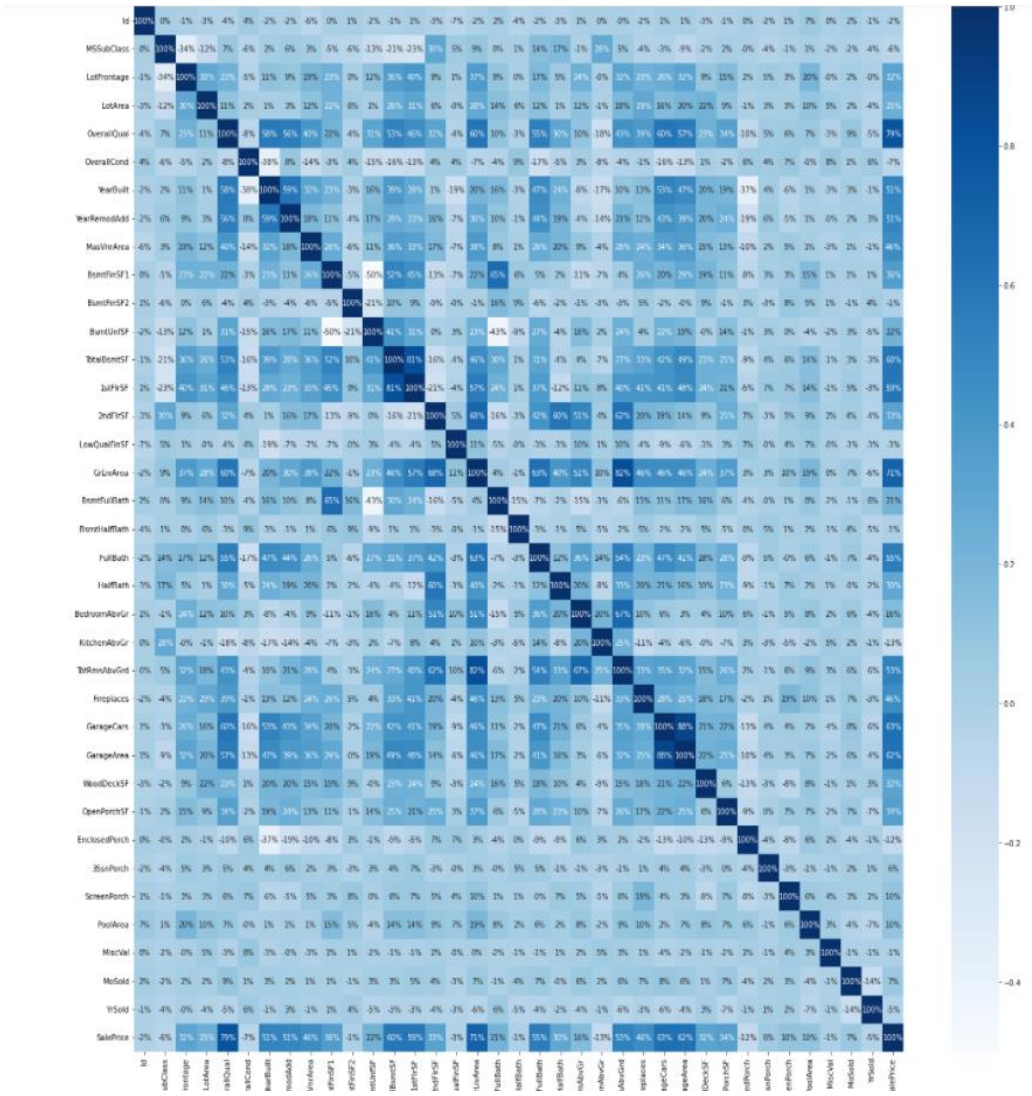
In [11]:
```python
# Let's explore the categorical columns

for column in df.columns:
    if df[column].dtypes == object:
        print(str(column) + ' : ' + str(df[column].unique()))
        print(df[column].value_counts())
        print('\n')
```

```
MSZoning : ['RL' 'RM' 'FV' 'RH' 'C (all)']
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64


Street : ['Pave' 'Grvl']
Pave    1164
Grvl       4
Name: Street, dtype: int64


Alley : [nan 'Grvl' 'Pave']
Grvl    41
Pave    36
Name: Alley, dtype: int64
```

We saw that Utilities only has one unique value, so we'll be removing this field. Then, using dummy variables, we

converted all of the categorical columns to numerical columns.
Then we checked the corelation with the heatmap.

```
In [23]: # Let's plot the heat map

         plt.figure(figsize=(24,24))
         sns.heatmap(df_cor,annot=True,fmt='.0%',cmap='Blues')
         plt.show()
```



While observing the heatmap we observed that:

• The columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea are all significantly positively associated.

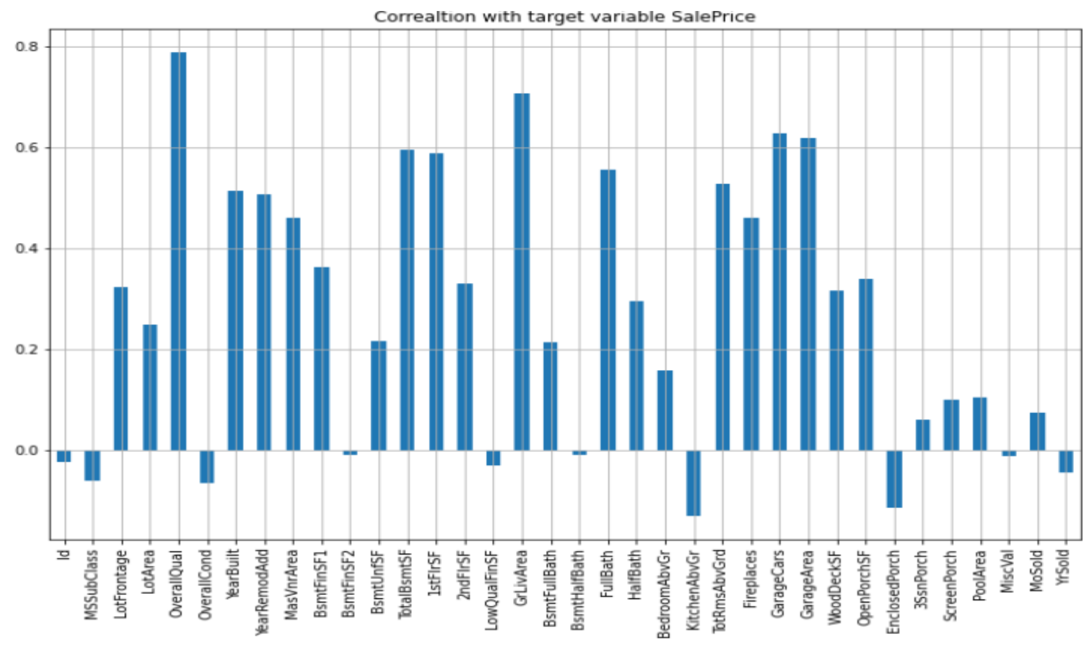
• OverallCond, KitchenAbvGr, Encloseporch, and YrSold are all adversely linked with SalePrice.

• Because we notice multicollinearity between columns, we'll use Principal Component Analysis (PCA).

• Because there is no correlation between the column Id and the other columns, this column will be removed.

Data Inputs- Logic- Output Relationships:

Here, we examine the relationship between all of our feature variables and the label of the target variable.



1. The most favourable correlation between OverallQual and SalePrice is in the column OverallQual.

2. The highest negative correlation between KitchenAbvGrd and SalePrice is in the column KitchenAbvGrd. a set of assumptions about the problem being studied

We assumed it was a Regression problem after looking at the target variable label.Because we noticed multicollinearity across columns, we decided to use Principal Component Analysis (PCA).We also saw that the Utilities column had only one unique value, thus we anticipated that these columns will be removed.

3.DATA ANALYSIS AND VISUALIZATION:

1.Identification of possible problem-solving approaches (methods):

To check out, we transformed all of our categorical variables to numeric variables using dummy variables and removed the columns that we considered were unneeded.

We noticed skewness in the data and attempted to reduce it by using the winsorization technique to address outliers.

We used sklearn's StandardScaler package to scale the feature variables on a single scale because the data was improperly scaled.

We used Principal Component Analysis (PCA) to minimise the number of feature variables in the data from 256 to 100 by showing

Eigenvalues and using the number of nodes as our number of feature variables.

2.Testing of Identified Approaches (Algorithms)

The algorithms we used for the training and testing are as follows:-

- Linear Regression
- Lasso
- Ridge
- Elastic Net
- SVR
- KNeighbors Regressor
- Decision Tree Regressor
- Random Forest Regressor
- Ada Boost Regressor
- Gradient Boosting Regressor

RUN AND EVALUATE SELECTED MODELS:

```
In [78]: model=[LinearRegression(),
               DecisionTreeRegressor(),
               KNeighborsRegressor(),
               SVR(),
               Lasso(),
               Ridge(),
               ElasticNet(),
               RandomForestRegressor(),
               AdaBoostRegressor(),
               GradientBoostingRegressor()
              ]
         for m in model:
             m.fit(x_train,y_train)
             print('score of',m,'is:',m.score(x_train,y_train))
             predm=m.predict(x_test)
             print('Error:')
             print('Mean absolute error:',mean_absolute_error(y_test,predm))
             print('Mean squared error:',mean_squared_error(y_test,predm))
             print('Root Mean Squared Error:',np.sqrt(mean_squared_error(y_test,predm)))
             print("r2_score:",r2_score(y_test,predm))
             print('**************************************************************')
             print('\n')
```

```
score of LinearRegression() is: 0.8261885623849852
Error:
Mean absolute error: 20912.728925586456
Mean squared error: 976364504.7347167
Root Mean Squared Error: 31246.831915167284
r2_score: 0.8418204242613713
************************************************************


score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 30810.15811965812
Mean squared error: 2155863483.269231
Root Mean Squared Error: 46431.27699373808
r2_score: 0.6507312899227278
************************************************************


score of KNeighborsRegressor() is: 0.7872613942929165
Error:
Mean absolute error: 25140.612820512823
Mean squared error: 1335919943.1006837
Root Mean Squared Error: 36550.23861892948
r2_score: 0.7835693034766205
************************************************************


score of SVR() is: -0.04904428034111108
Error:
Mean absolute error: 58204.53456411457
Mean squared error: 6657851394.448467
Root Mean Squared Error: 81595.65793869467
r2_score: -0.07863006469170974
************************************************************
score of Lasso() is: 0.8261885527142644
Error:
Mean absolute error: 20910.564704916495
Mean squared error: 976180174.7789334
Root Mean Squared Error: 31243.88219762284
r2_score: 0.8418502873238447
************************************************************


score of Ridge() is: 0.826188489973443
Error:
Mean absolute error: 20906.190339336816
Mean squared error: 975786667.9889657
Root Mean Squared Error: 31237.58422139852
r2_score: 0.8419140388600644
************************************************************


score of ElasticNet() is: 0.8180009506545038
Error:
Mean absolute error: 19807.457331821464
Mean squared error: 896085304.2865978
Root Mean Squared Error: 29934.68396837685
r2_score: 0.8548263557612776
************************************************************


score of RandomForestRegressor() is: 0.973456342125331
Error:
Mean absolute error: 20234.690085470087
Mean squared error: 907991185.6714469
Root Mean Squared Error: 30132.89208940036
r2_score: 0.8528974990104257
************************************************************
```

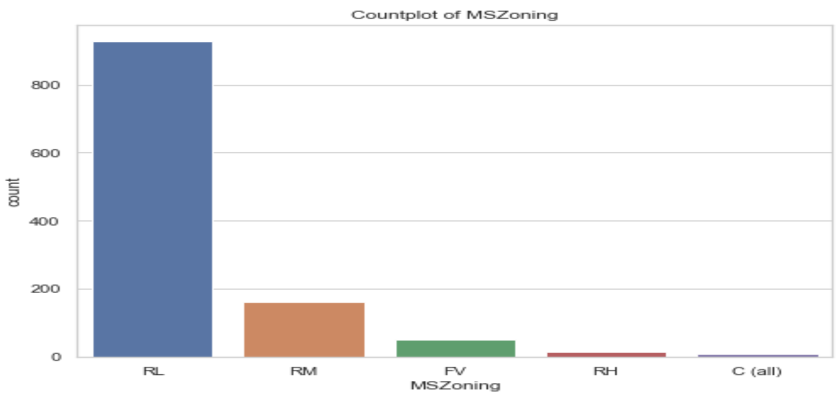
```
score of AdaBoostRegressor() is: 0.8475582998072446
Error:
Mean absolute error: 30186.52890240819
Mean squared error: 1633822387.0466282
Root Mean Squared Error: 40420.568860997344
r2_score: 0.7353065061643878
*************************************************************


score of GradientBoostingRegressor() is: 0.9733791310413415
Error:
Mean absolute error: 21260.90909090528
Mean squared error: 1021886566.0368047
Root Mean Squared Error: 31966.960537980533
r2_score: 0.8344454528151606
*************************************************************
```

3.Key Metrics for success in solving problem under consideration:

We chose the Ridge Regressor model that gave us the greatest (least) RMSE score using the measure Root Mean Squared Error.

4.Visualizations:

In [26]:
```
# Let's check the column MsZoning

plt.subplots(figsize=(8,6))
sns.countplot(x="MSZoning", data=df)
plt.title("Countplot of MSZoning")
plt.xlabel('MSZoning')
plt.ylabel("count")
plt.show()

df['MSZoning'].value_counts()
```

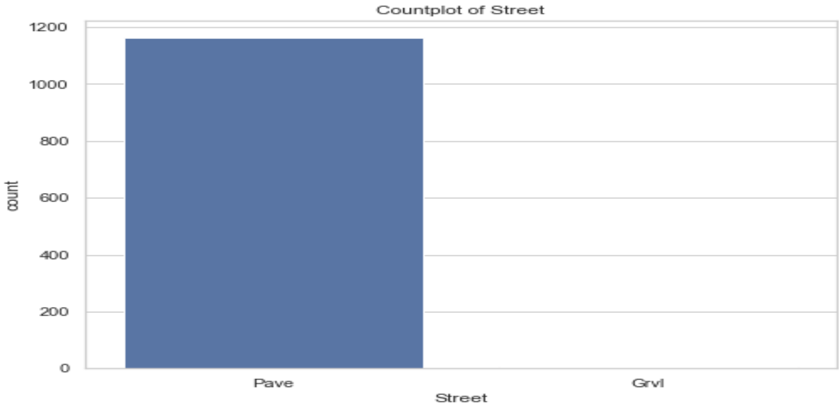


Out[26]:
```
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
```

Observation:

Maximum of 928 number of MSZoning are RL.

In [27]:
```python
# Let's check the column Street

plt.subplots(figsize=(8,6))
sns.countplot(x="Street", data=df)
plt.title("Countplot of Street")
plt.xlabel('Street')
plt.ylabel("count")
plt.show()

df['Street'].value_counts()
```

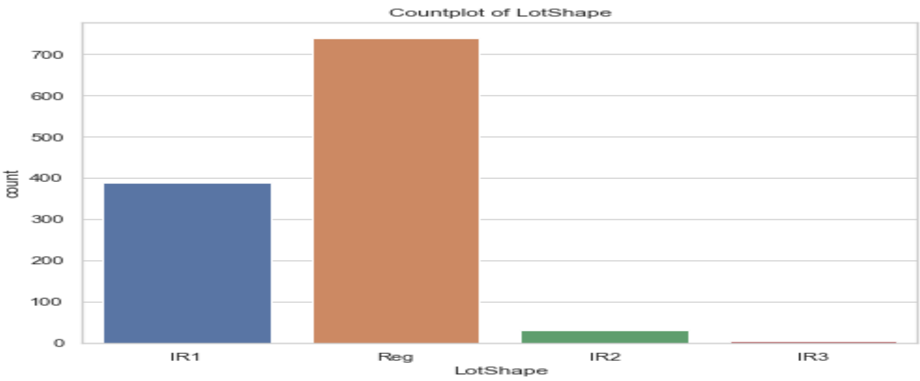


Out[27]:
```
Pave    1164
Grvl       4
Name: Street, dtype: int64
```

Observation:

Maximum of 1164 number of Street are Pave where as only 4 are Grvl.

In [28]:
```python
# Let's check the column LotShape

plt.subplots(figsize=(8,6))
sns.countplot(x="LotShape", data=df)
plt.title("Countplot of LotShape")
plt.xlabel('LotShape')
plt.ylabel("count")
plt.show()

df['LotShape'].value_counts()
```

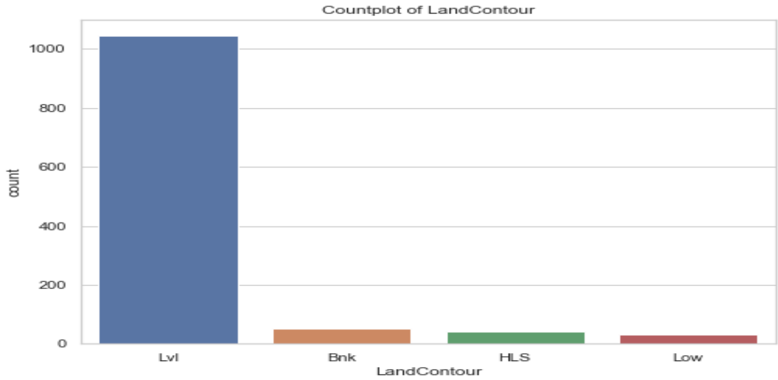


Out[28]:
```
Reg    740
IR1    390
IR2     32
IR3      6
Name: LotShape, dtype: int64
```

Observation:

Maximum of 740 number of LotShape are Reg.

In [29]:
```
# Let's check the column LandContour

plt.subplots(figsize=(8,6))
sns.countplot(x="LandContour", data=df)
plt.title("Countplot of LandContour")
plt.xlabel('LandContour')
plt.ylabel("count")
plt.show()

df['LandContour'].value_counts()
```

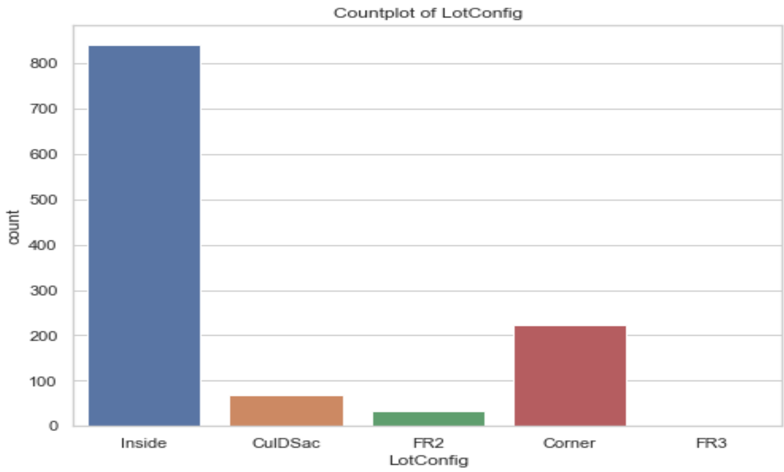


Out[29]:
```
Lvl      1046
Bnk        50
HLS        42
Low        30
Name: LandContour, dtype: int64
```

Observation:

Maximum, 1046 number of LandContour are Lvl.

In [30]:
```
# Let's check the column LotConfig

plt.subplots(figsize=(8,6))
sns.countplot(x="LotConfig", data=df)
plt.title("Countplot of LotConfig")
plt.xlabel('LotConfig')
plt.ylabel("count")
plt.show()

df['LotConfig'].value_counts()
```

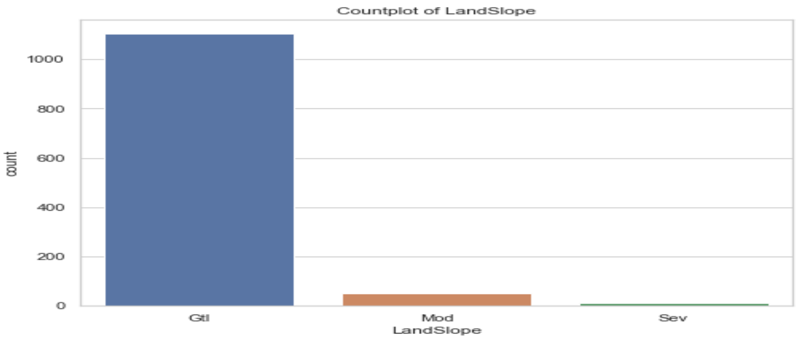


Out[30]:
```
Inside     842
Corner     222
CulDSac     69
FR2         33
FR3          2
Name: LotConfig, dtype: int64
```

Observation: ¶

Maximum of 842 number of LotConfig are Inside

In [31]:
```
# Let's check the column LandSlope

plt.subplots(figsize=(8,6))
sns.countplot(x="LandSlope", data=df)
plt.title("Countplot of LandSlope")
plt.xlabel('LandSlope')
plt.ylabel("count")
plt.show()

df['LandSlope'].value_counts()
```

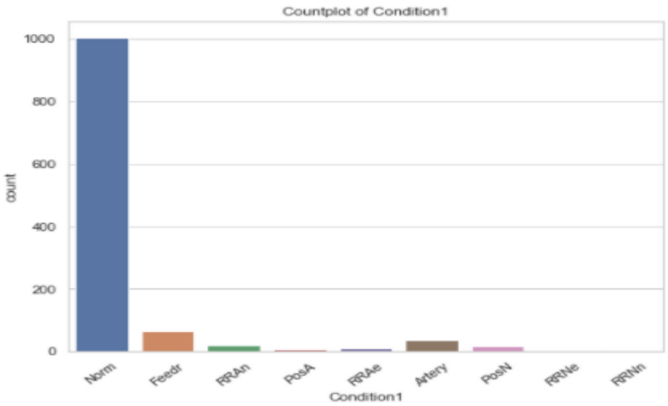


Out[31]:
```
Gtl     1105
Mod       51
Sev       12
Name: LandSlope, dtype: int64
```

Observation:

Maximum of 1105 number of LandSlope are Gtl.

In [33]:
```
# Let's check the column Condition1

plt.subplots(figsize=(8,6))
sns.countplot(x="Condition1", data=df)
plt.title("Countplot of Condition1")
plt.xticks(rotation=40)
plt.xlabel('Condition1')
plt.ylabel("count")
plt.show()

df['Condition1'].value_counts()
```

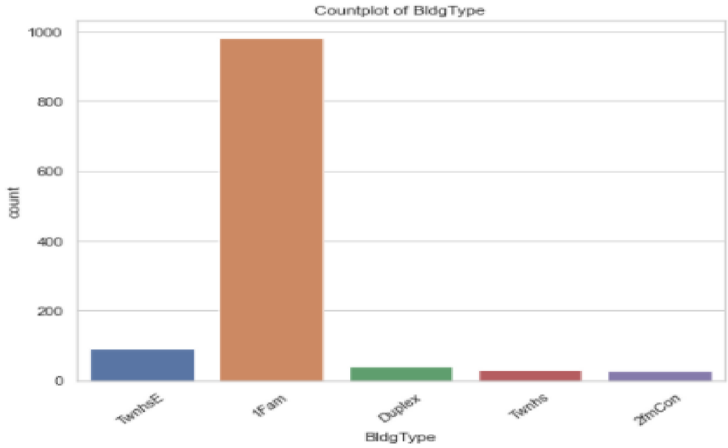


Out[33]:
```
Norm      1005
Feedr       67
Artery      38
RRAn        20
PosN        17
RRAe         9
PosA         6
RRNn         4
RRNe         2
Name: Condition1, dtype: int64
```

Observation:

Maximum of 1005 number of Condition1 is Norm.

In [34]:
```python
# Let's check the column BldgType

plt.subplots(figsize=(8,6))
sns.countplot(x="BldgType", data=df)
plt.title("Countplot of BldgType")
plt.xticks(rotation=40)
plt.xlabel('BldgType')
plt.ylabel("count")
plt.show()

df['BldgType'].value_counts()
```

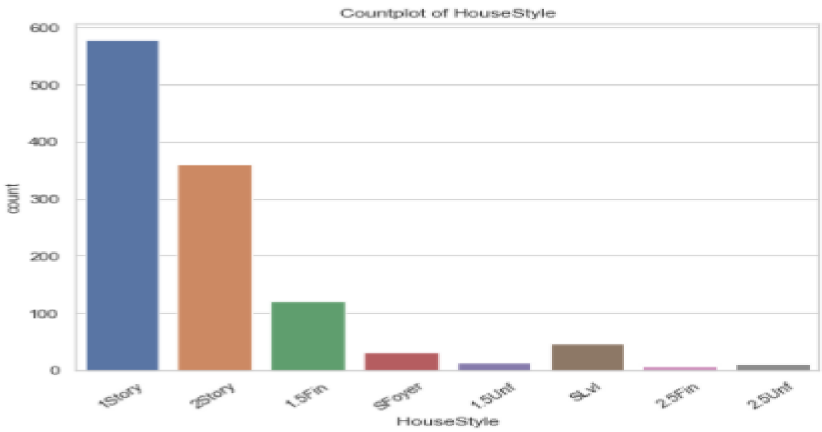


Out[34]:
```
1Fam      981
TwnhsE     90
Duplex     41
Twnhs      29
2fmCon     27
Name: BldgType, dtype: int64
```

Observation:

Maximum of 981 number of BldgType are 1Fam.

In [35]:
```python
# Let's check the column HouseStyle

plt.subplots(figsize=(8,6))
sns.countplot(x="HouseStyle", data=df)
plt.title("Countplot of HouseStyle")
plt.xticks(rotation=40)
plt.xlabel('HouseStyle')
plt.ylabel("count")
plt.show()

df['HouseStyle'].value_counts()
```

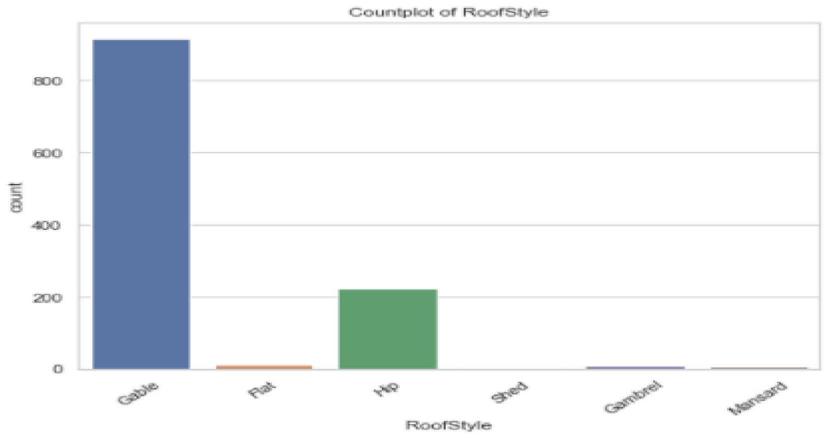


Out[35]:
```
1Story    578
2Story    361
1.5Fin    121
SLvl       47
SFoyer     32
1.5Unf     12
2.5Unf     10
2.5Fin      7
Name: HouseStyle, dtype: int64
```

Observation:

1Story has highest number of count followed by 2Story, 1.5Fin, SlvL etc.

In [36]:
```python
# Let's check the column RoofStyle

plt.subplots(figsize=(8,6))
sns.countplot(x="RoofStyle", data=df)
plt.title("Countplot of RoofStyle")
plt.xticks(rotation=40)
plt.xlabel('RoofStyle')
plt.ylabel("count")
plt.show()

df['RoofStyle'].value_counts()
```

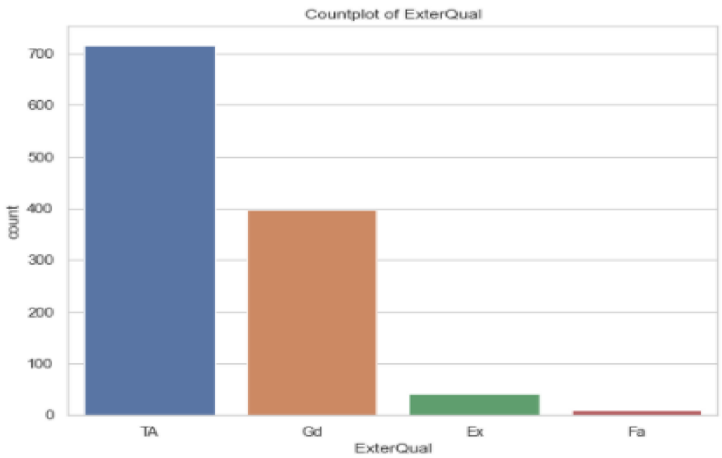


Out[36]:
```
Gable      915
Hip        225
Flat        12
Gambrel      9
Mansard      5
Shed         2
Name: RoofStyle, dtype: int64
```

Observation:

Maximum of 915 number of RoofStyle are Gable.

In [37]:
```python
# Let's check the column ExterQual

plt.subplots(figsize=(8,6))
sns.countplot(x="ExterQual", data=df)
plt.title("Countplot of ExterQual")
plt.xlabel('ExterQual')
plt.ylabel("count")
plt.show()

df['ExterQual'].value_counts()
```

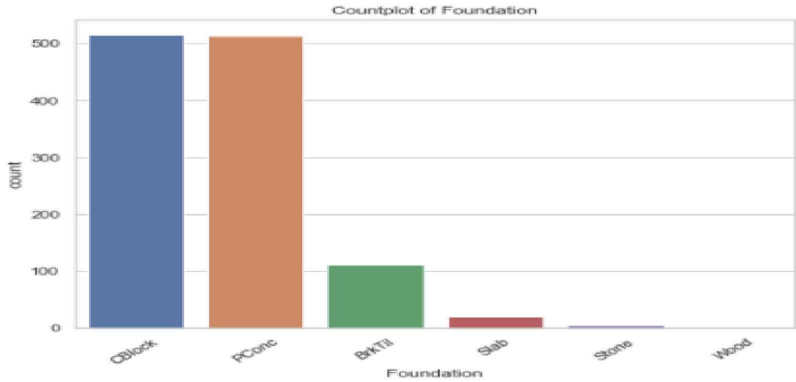


Out[37]:
```
TA    717
Gd    397
Ex     43
Fa     11
Name: ExterQual, dtype: int64
```

Observation:

Maximum of 717 number of ExterQual is TA.

In [38]:
```
# Let's checking the column Foundation

plt.subplots(figsize=(8,6))
sns.countplot(x="Foundation", data=df)
plt.title("Countplot of Foundation")
plt.xticks(rotation=40)
plt.xlabel('Foundation')
plt.ylabel("count")
plt.show()

df['Foundation'].value_counts()
```

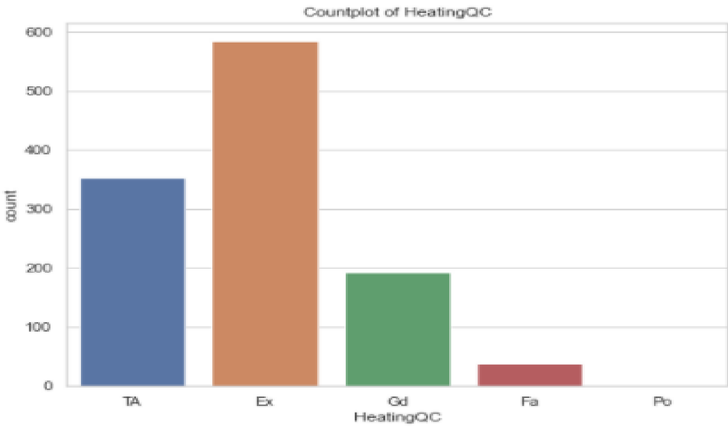


Out[38]:
```
CBlock     516
PConc      513
BrkTil     112
Slab        21
Stone        5
Wood         1
Name: Foundation, dtype: int64
```

Observation:

Maximum of 516 number of Foundation are CBlock.

In [39]:
```
# Let's check the column HeatingQC

plt.subplots(figsize=(8,6))
sns.countplot(x="HeatingQC", data=df)
plt.title("Countplot of HeatingQC")
plt.xlabel('HeatingQC')
plt.ylabel("count")
plt.show()

df['HeatingQC'].value_counts()
```

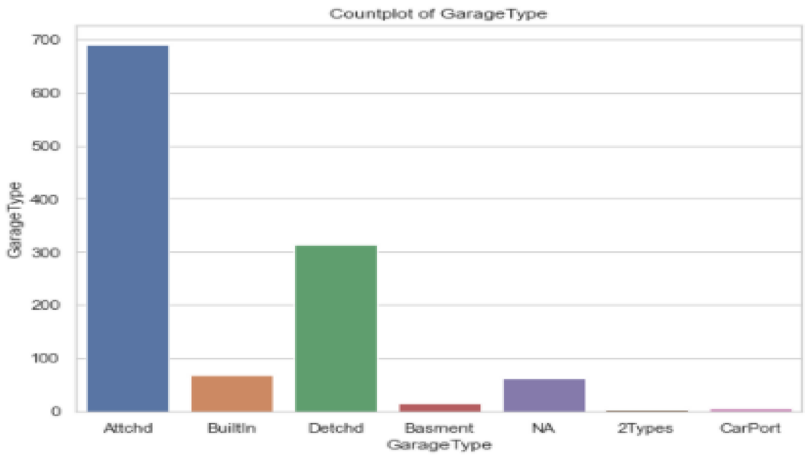


Out[39]:
```
Ex     585
TA     352
Gd     192
Fa      38
Po       1
Name: HeatingQC, dtype: int64
```

Observation:

Maximum of 585 number of HeatingQC is Ex.

In [40]:
```python
# Let's check the column GarageType

plt.subplots(figsize=(8,6))
sns.countplot(x="GarageType", data=df)
plt.title("Countplot of GarageType")
plt.xlabel('GarageType')
plt.ylabel("GarageType")
plt.show()

df['GarageType'].value_counts()
```

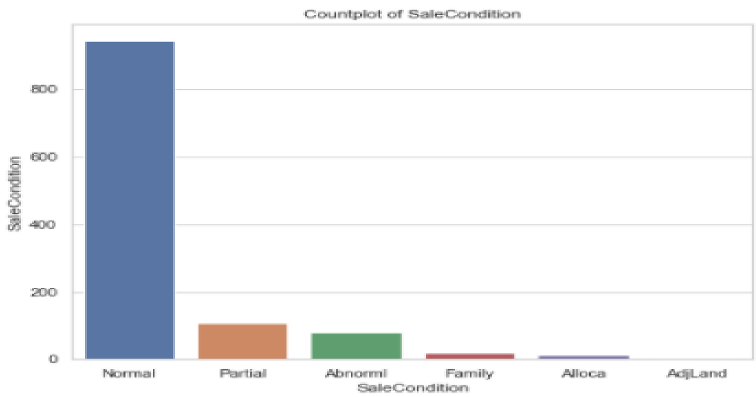


Out[40]:
```
Attchd     691
Detchd     314
BuiltIn     70
NA          64
Basment     16
CarPort      8
2Types       5
Name: GarageType, dtype: int64
```

Observation:

Maximum of 691 number of GarageType are Attchd.

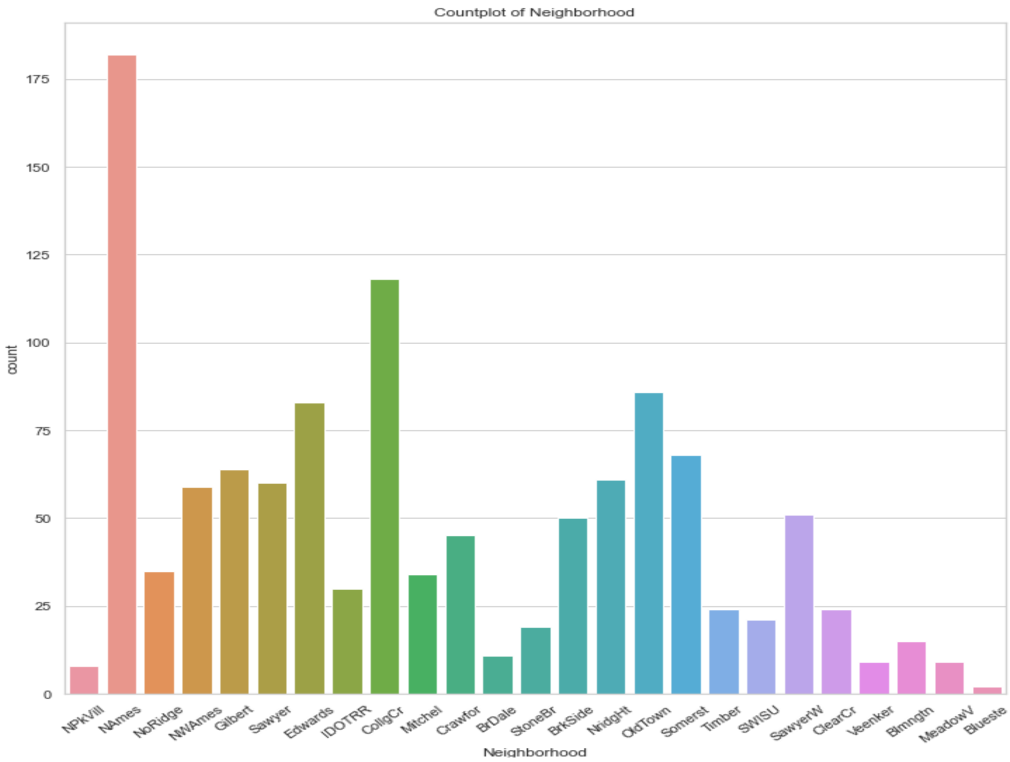
In [41]:
```python
# Let's check the column SaleCondition

plt.subplots(figsize=(8,6))
sns.countplot(x="SaleCondition", data=df)
plt.title("Countplot of SaleCondition")
plt.xlabel('SaleCondition')
plt.ylabel("SaleCondition")
plt.show()

df['SaleCondition'].value_counts()
```



Out[41]:
```
Normal     945
Partial    108
Abnorml     81
Family      18
Alloca      12
AdjLand      4
Name: SaleCondition, dtype: int64
```

Observation:

Maximum of 945 number of SaleCondition is normal.




```
Out[32]: NAmes      182
         CollgCr    118
         OldTown     86
         Edwards     83
         Somerst     68
         Gilbert     64
         NridgHt     61
         Sawyer      60
         NWAmes      59
         SawyerW     51
         BrkSide     50
         Crawfor     45
         NoRidge     35
         Mitchel     34
         IDOTRR      30
         Timber      24
         ClearCr     24
         SWISU       21
         StoneBr     19
         Blmngtn     15
         BrDale      11
         MeadowV      9
         Veenker      9
         NPkVill      8
         Blueste      2
         Name: Neighborhood, dtype: int64
```

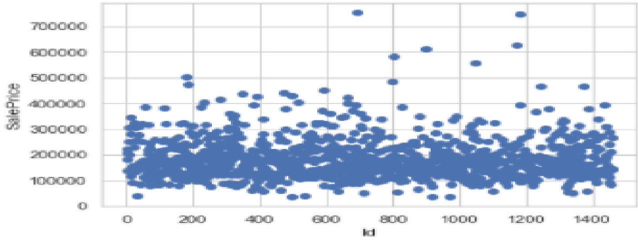
Observation:

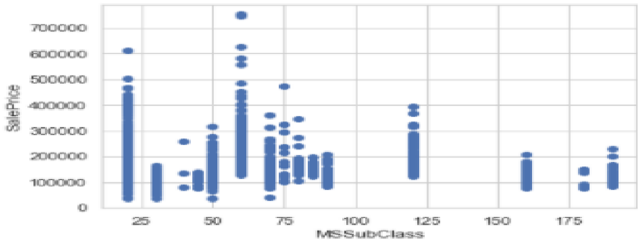
Maximum of 182 number of Neighborhood are Names.

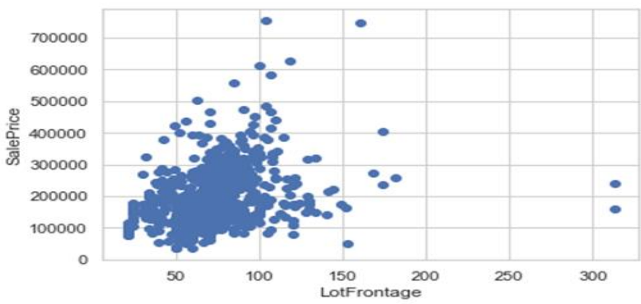
Bivariate Analysis:

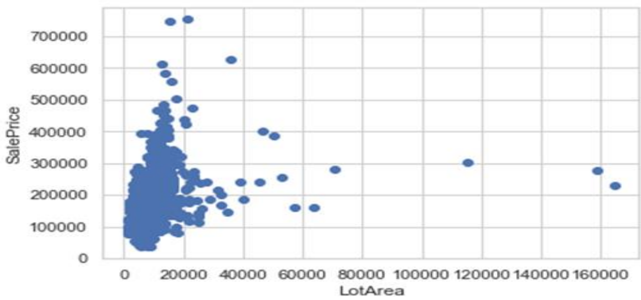
Bivariate Analysis:

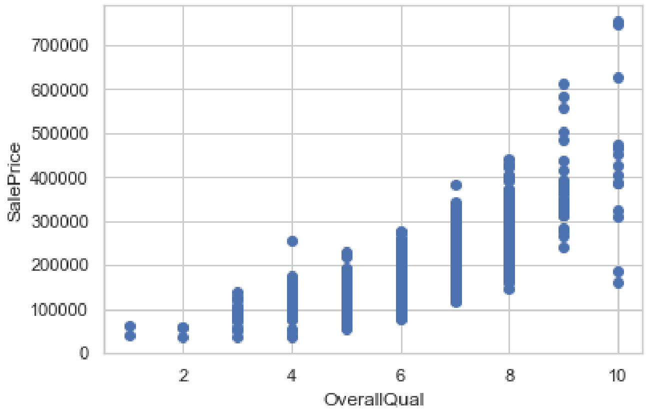
In [43]:
```
# Let's plot the Scatter plot between all feature variables and target variable

for col in df.describe().columns:
    data=df.copy()
    plt.scatter(data[col],data['SalePrice'])
    plt.xlabel(col)
    plt.ylabel('SalePrice')
    plt.show()
```

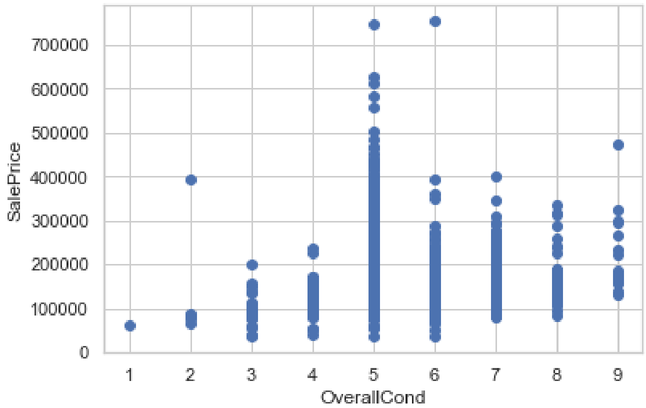


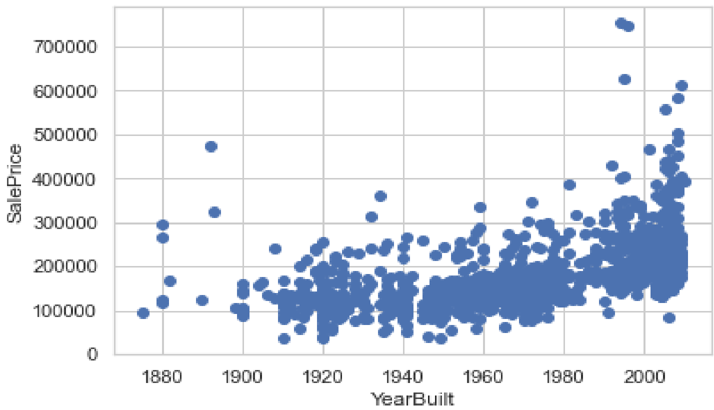


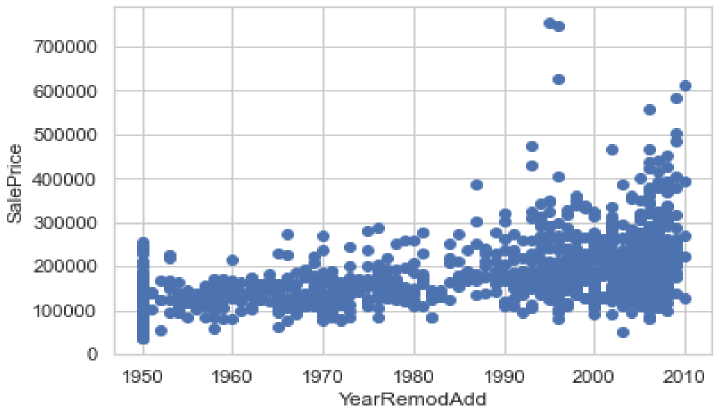


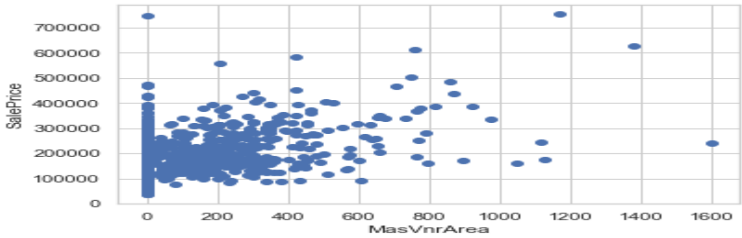


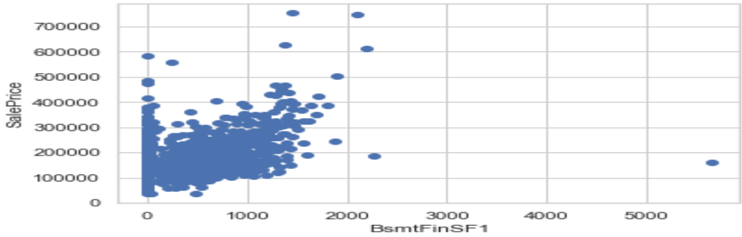

SalePrice
OverallQual
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10

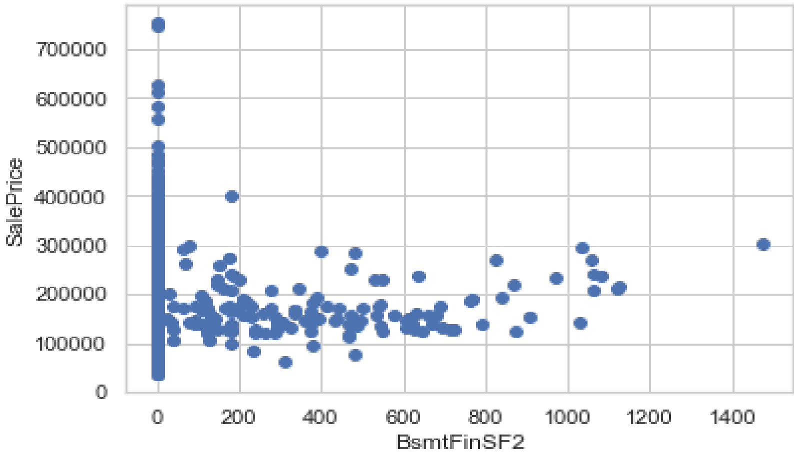

SalePrice
OverallCond
0
100000
200000
300000
400000
500000
600000
700000
1
2
3
4
5
6
7
8
9

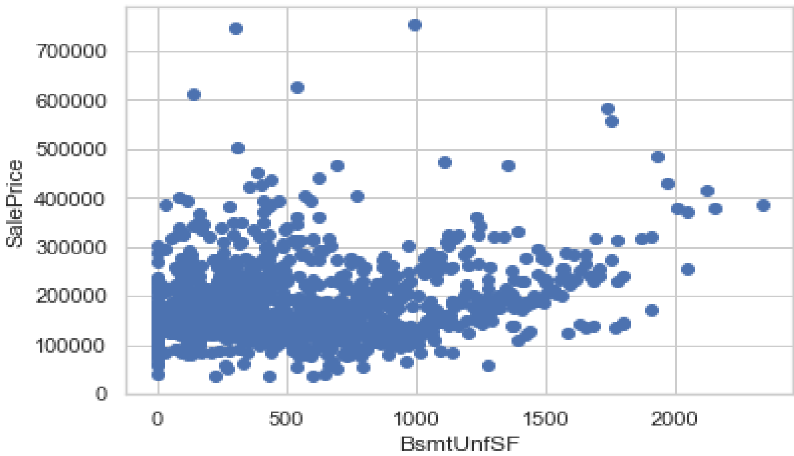

SalePrice
YearBuilt
0
100000
200000
300000
400000
500000
600000
700000
1880
1900
1920
1940
1960
1980
2000

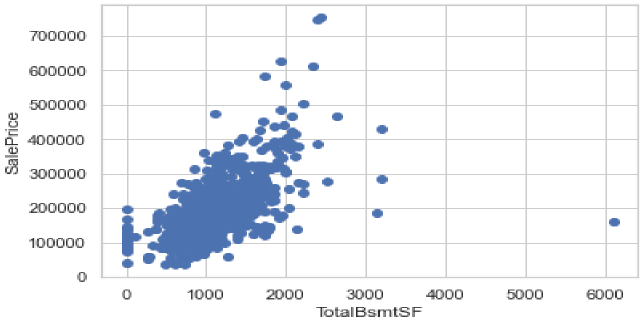

SalePrice
YearRemodAdd
0
100000
200000
300000
400000
500000
600000
700000
1950
1960
1970
1980
1990
2000
2010

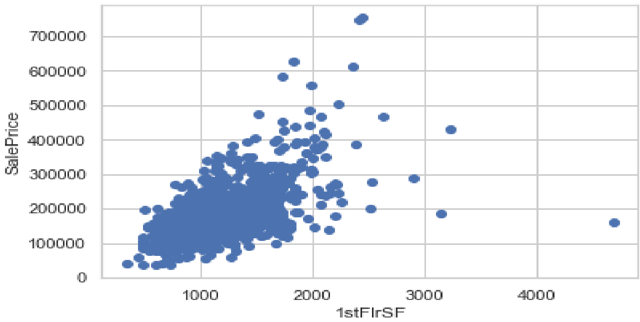

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

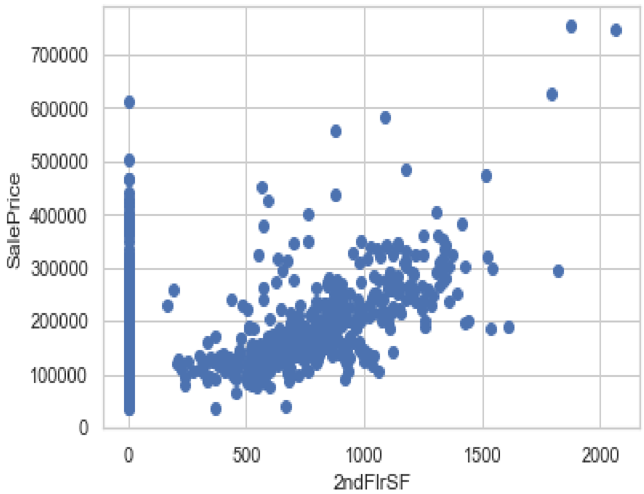

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
BsmtFinSF1

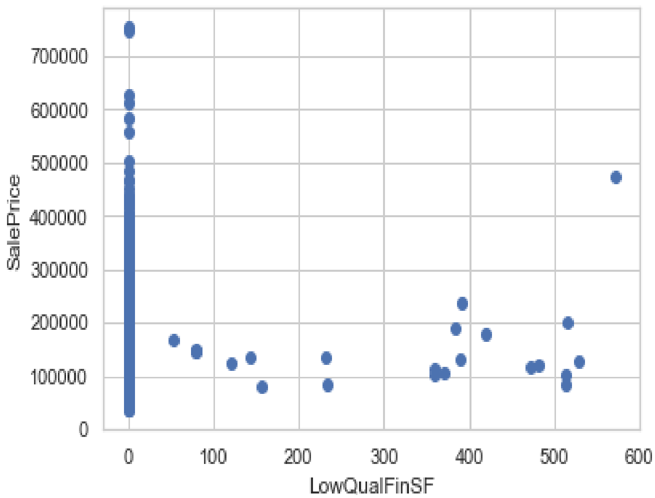

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800
1000
1200
1400
BsmtFinSF2

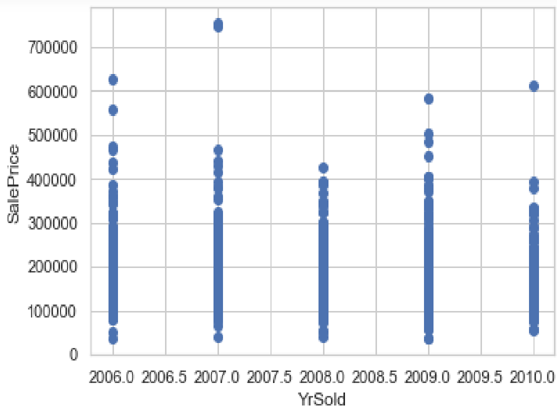

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
500
1000
1500
2000
BsmtUnfSF

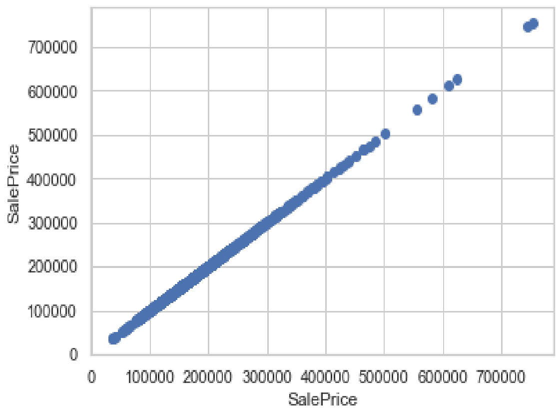

SalePrice
TotalBsmtSF
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
6000


SalePrice
1stFlrSF
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000


SalePrice
2ndFlrSF
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000

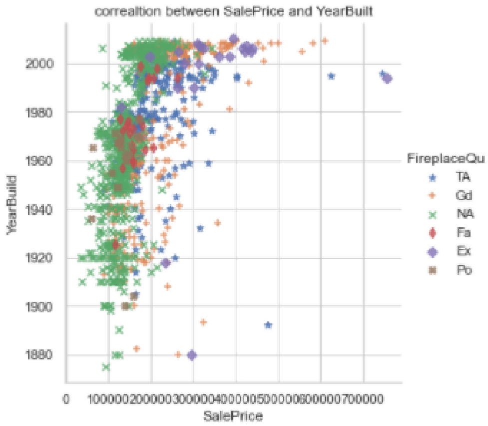

SalePrice
LowQualFinSF
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600





In [57]:
```
# Let's plot the scatter plot between SalePrice and OverallCond with respect to MSZoning

plt.figure(figsize=(14,14))
sns.lmplot(x='SalePrice',y='YearBuilt',fit_reg=False,data=df,hue='FireplaceQu',markers=['*','+','x','d','D','X'])
plt.xlabel('SalePrice')
plt.title('correaltion between SalePrice and YearBuilt')
plt.ylabel('YearBuild')
plt.show()
```

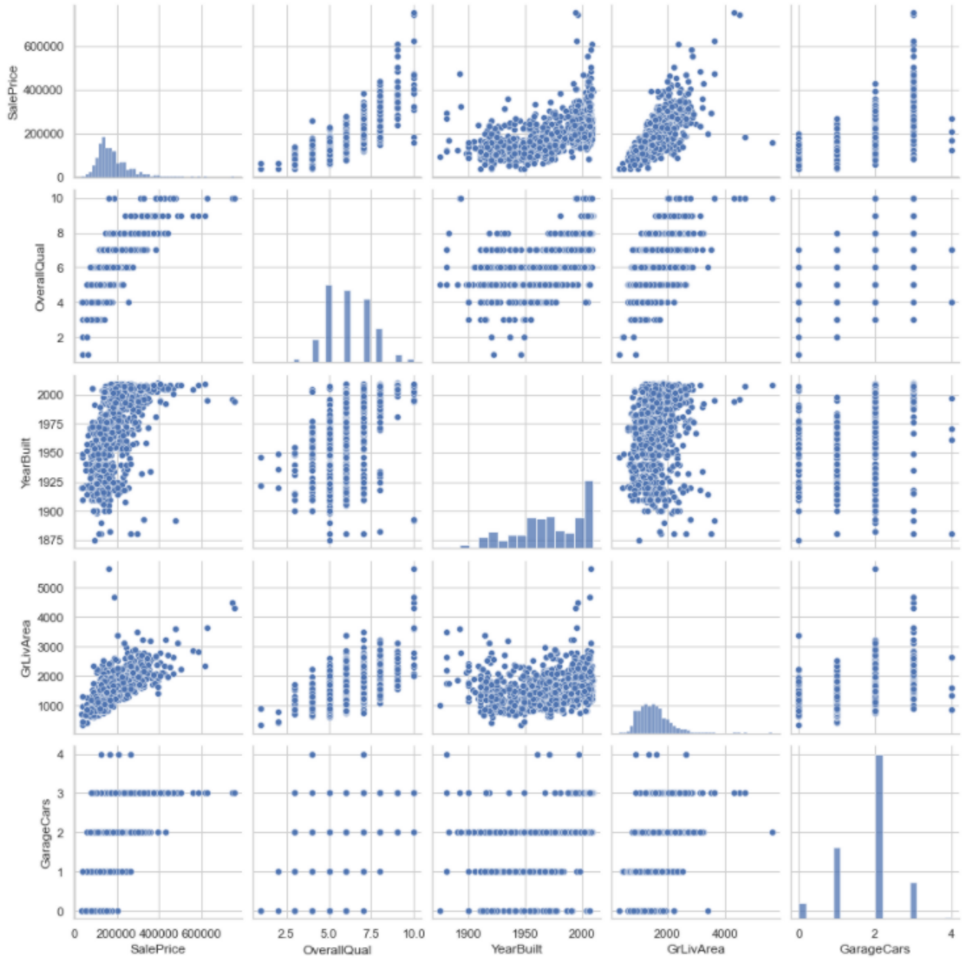
```
<Figure size 1008x1008 with 0 Axes>
```



Observation:

As the YearBuilt is increasing SalePrice is also increasing.

In [58]:
```
# Let's plot the pairplot

sns.pairplot(df, vars=['SalePrice','OverallQual','YearBuilt','GrLivArea','GarageCars']);
```



Observation:

SalePrice is highly positively correlated with GrLivArea and OverallQual.

Interpretation of the Results:

Hyper Parameter Tuning:

We chose the Ridge Regressor model that gave us the greatest (least) RMSE score using the measure Root Mean Squared Error.

Hyperparameter Tuning

In [79]:
```
# Let's Use the GridSearchCV to find the best paarameters in Ridge Regressor

parameters={'alpha': [25,10,4,2,1.0,0.8,0.5,0.3,0.2,0.1,0.05,0.02,0.01]}
rg=Ridge()

reg=GridSearchCV(rg,parameters,n_jobs=-1)
reg.fit(x,y)
print(reg.best_params_)
```

```
{'alpha': 25}
```

In [80]:
```
# Let's use the Ridge Regressor with its best parameters

RG=Ridge(alpha=25)
RG.fit(x_train,y_train)
print('Score:',RG.score(x_train,y_train))
y_pred=RG.predict(x_test)
print('\n')
print('Mean absolute error:',mean_absolute_error(y_test,y_pred))
print('Mean squared error:',mean_squared_error(y_test,y_pred))
print('Root Mean Squared error:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('\n')
print("r2_score:",r2_score(y_test,y_pred))
print('\n')
```

```
Score: 0.8261453142743175


Mean absolute error: 20755.9300607753
Mean squared error: 962984816.822721
Root Mean Squared error: 31031.99666187661


r2_score: 0.8439880505394382
```

After hyperparameter optimization, we found that Ridge Regressor works well with respect to our model, with a minimum RMSE of 32302.

CONCLUSION :

Key Findings and Conclusions of the Study:

In this research, we attempted to demonstrate how housing prices fluctuate and the elements that influence their fluctuation. The best(minimum) RMSE score was obtained by GridSearchCV utilising the optimal settings of the Ridge Regressor, however the Lasso Regressor model also performed well.

Learning Outcomes of the Study in respect of Data Science:

This experiment has highlighted the importance of effective sampling, modelling, and data prediction.We were able to analyse and comprehend several hidden insights about the data using various advanced visualisation tools.We were able to remove extraneous columns and outliers from our dataset using data cleaning, which would have caused our model to overfit or underfit.

The few challenges while working on this project where:-

- Improper scaling
- Too many features
- Missing values
- Skewed data due to outliers

We used sklearns's package StandardScaler to scale the data to a single scale because it was improperly scaled.Because there were too many features in the data (256), we used Principal Component Analysis (PCA) to get the Eigenvalues and then reduced our features to 90 columns based on the number of nodes.There were numerous missing values in various columns, which we interpolated based on our knowledge.Due to the presence of outliers, the columns were skewed, which we corrected using the winsorization procedure.

Limitations of this work and Scope for Future Work:

While we were unable to achieve our aim of a minimum RMSE in house price prediction without allowing the model to overfit, we did create a system that can go very near to that goal given enough time and data. There is always space for improvement in any endeavour. Because of the nature of this project, multiple algorithms can be merged as modules and their findings mixed to improve the accuracy of the final output. More algorithms can be added to this model to improve it even further. These algorithms' output, however, must be in the same format as the others. The modules are simple to add once that criterion is met, as seen in the code. The project gains a lot of modularity and versatility as a result of this.