FLIP ROBO

Housing Prediction Project

Submitted by:

PUNEET JAWA

ACKNOWLEDGMENT

I want to thanks our SME Miss. Khushboo Garg for providing the Dataset and helping us to solve the problem and addressing out our Query in right time.

INTRODUCTION

Business Problem Framing

This is a real estate problem where a US based housing company named Surprise Housing has decided to invest in Australian Market. Their agenda is to buy houses in Australia at prices below their actual value in the market and sell them at high prices to gain profit. To do this this company uses data analytics to decide in which property they must invest. Company has collected the data of previously sold houses in Australia and with the help of this data they want to know to the value of prospective properties to decide whether it will suitable to invest in the properties or not. To know the value of Properties Company has provided data to us to do data analysis and to extract the information of attributes which are important to predict the price of the houses. They want a machine learning model which can predict the price of houses and also the significance of each important attribute in house prediction i.e, how and to what intensity each variable impacts the price of the house.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

In real estate the value of property usually increases with time as seen in many countries. One of the causes for this is due to rising population. The value of property also depends on the proximity of the property, its size its neighbourhood and audience for which the property is subjected to be sold. For example, if audience is mainly concerned of commercial purpose. Then the property which is located in densely populated area will be sold very fast and at high prices compared to the one located at remote place. Similarly, if audience is concerned only on living place then property with less dense area having large area with all services will be sold at higher prices. The company is looking at prospective properties to buy houses to enter the market. We 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.

REVIEW OF LITERATURE

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 markets which is one of the major contributors in the world's economy. 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. We 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.

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 understand real world problems where Machine Learning and Data Analysis can be applied to help organizations in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data .One of such domain is Real Estate. Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

In this project we have performed various mathematical and statistical analysis such as we checked description or statistical summary of the data using describe, checked correlation using corr and also visualized it using heatmap. Then we have used Z-Score to plot outliers and remove them.

```
In [17]: # Let's check the statistical summary of our dataset

HPP.describe()
```

Out[17]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	724.136130	56.767979	70.807363	10484.749144	6.104452	5.595890	1970.930651	1984.758562	101.696918	444.726027	46.647260
std	416.159877	41.940650	22.440317	8957.442311	1.390153	1.124343	30.145255	20.785185	182.218483	462.664785	163.520016
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000
25%	360.500000	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000
50%	714.500000	50.000000	70.000000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000
75%	1079.500000	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000
max	1460.000000	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000

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

In [6]:
```
# Let's check the information of our dataset

HPP.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
```

```
In [5]: # Let's check the data types of our columns

        HPP.dtypes
```

```
Out[5]: Id                 int64
        MSSubClass         int64
        MSZoning          object
        LotFrontage      float64
        LotArea            int64
                          ...
        MoSold             int64
        YrSold             int64
        SaleType          object
        SaleCondition     object
        SalePrice          int64
        Length: 81, dtype: object
```

DATA PREPROCESSING DONE

After loading all the required libraries we loaded the data into our jupyter notebook.

In [1]:
```python
# Let's import all the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.pandas.set_option('display.max_columns',None)

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from scipy import stats

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV,cross_val_score
from sklearn.model_selection import GridSearchCV

#importing warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
# Let's load our dataset

HPP=pd.read_csv(r"D:\flip WS\Project-Housing\Project-Housing_splitted\train.csv")
HPP
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Edwards	Feedr	
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	NPkVill	Norm	
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	Inside	Gtl	IDOTRR	Feedr	
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	

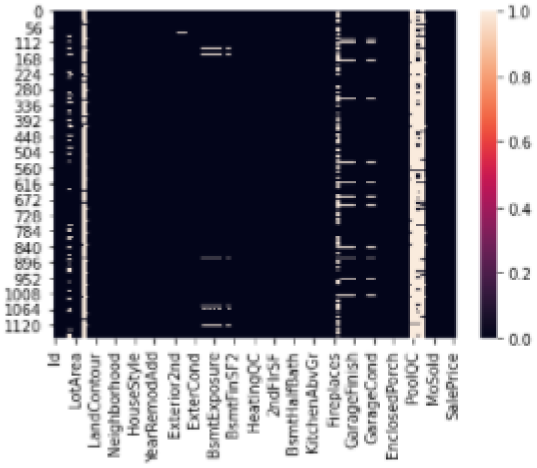
1168 rows × 81 columns

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 [9]:
```
# Let's check the missing values of top 30 columns

HPP.isnull().sum().sort_values(ascending = False).head(30)
```

Out[9]:
```
PoolQC          1161
MiscFeature     1124
Alley           1091
Fence            931
FireplaceQu      551
LotFrontage      214
GarageType        64
GarageCond        64
GarageYrBlt       64
GarageFinish      64
GarageQual        64
BsmtExposure      31
BsmtFinType2      31
BsmtFinType1      30
BsmtCond          30
BsmtQual          30
MasVnrArea         7
MasVnrType         7
Exterior2nd        0
Exterior1st        0
OverallCond        0
ExterQual          0
ExterCond          0
Foundation         0
RoofMatl           0
RoofStyle          0
YearRemodAdd       0
YearBuilt          0
SalePrice          0
OverallQual        0
dtype: int64
```

In [11]:
```
# Let's plot the heat map for our missing values

sns.heatmap(HPP.isnull());
```



In [12]:
```python
# Let's check the percentage of missing values of each column

def missing_values_table(HPP):
    mis_val = HPP.isnull().sum()
    mis_val_percent = 100 * HPP.isnull().sum() / len(HPP)
    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(HPP.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(HPP)
```

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

Out[12]:

	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
GarageType	64	5.5
GarageYrBlt	64	5.5
GarageFinish	64	5.5
GarageQual	64	5.5
GarageCond	64	5.5
BsmtExposure	31	2.7
BsmtFinType2	31	2.7
BsmtCond	30	2.6
BsmtFinType1	30	2.6
BsmtQual	30	2.6
MasVnrArea	7	0.6
MasVnrType	7	0.6

In [13]:
```python
# Let's fill the missing values in categorical columns as NA

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

In [14]:
```python
# Let's fill the missing values in MasVnrType with None

HPP['MasVnrType'] = HPP['MasVnrType'].fillna('None')
```

In [15]:
```python
# Let's fill the missing values in GarageYrBlt with 0

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

In [16]:
```python
# Let's Imputing the missing values and replace it with the median

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

```
In [7]: # Let's explore the categorical columns

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

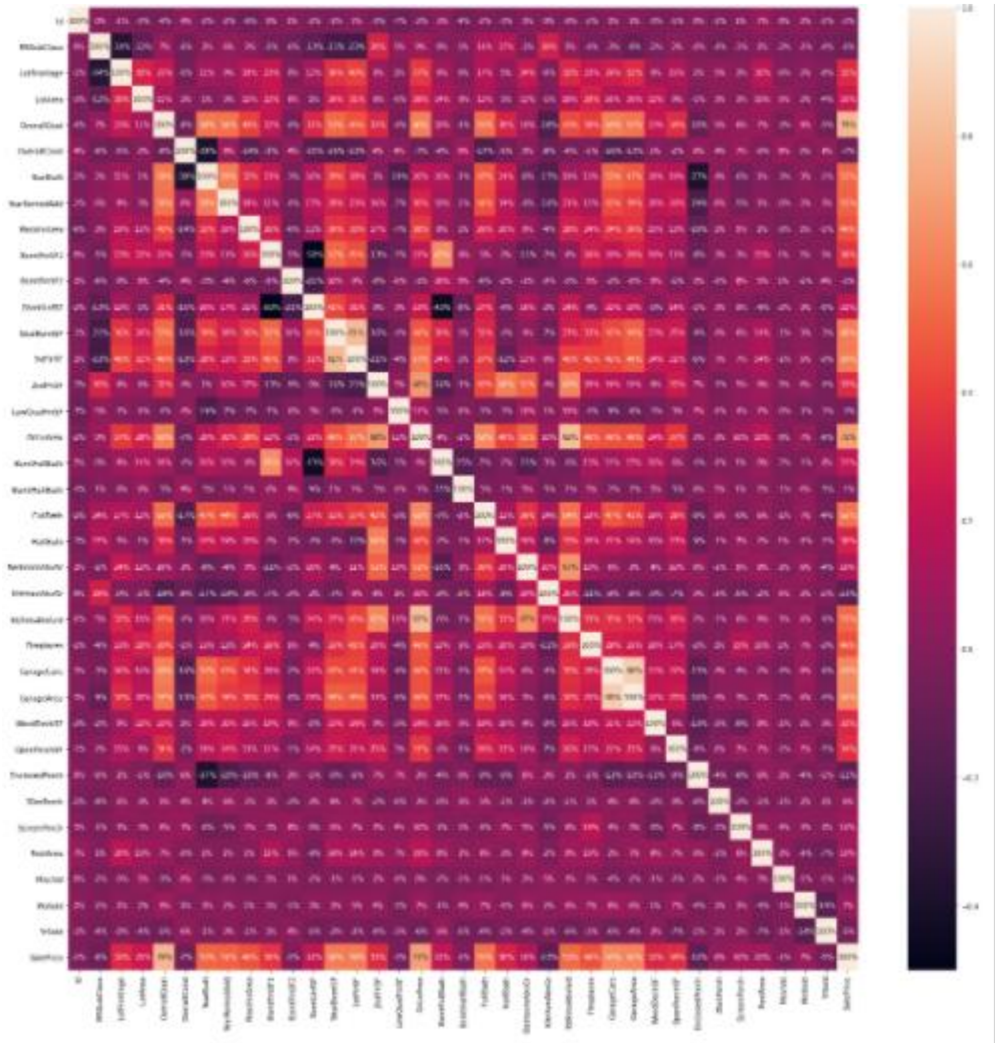
We observed that there is only one unique value present in Utilities so will be dropping this column. Then we encoded all the categorical columns into numerical columns using dummy variables.


```
In [7]: # Let's explore the categorical columns

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


Then we checked the correlation with the help of heatmap

```
In [19]: # Let's plot the heat map

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



While checking the heatmap of correlation we observed that:

• SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea.

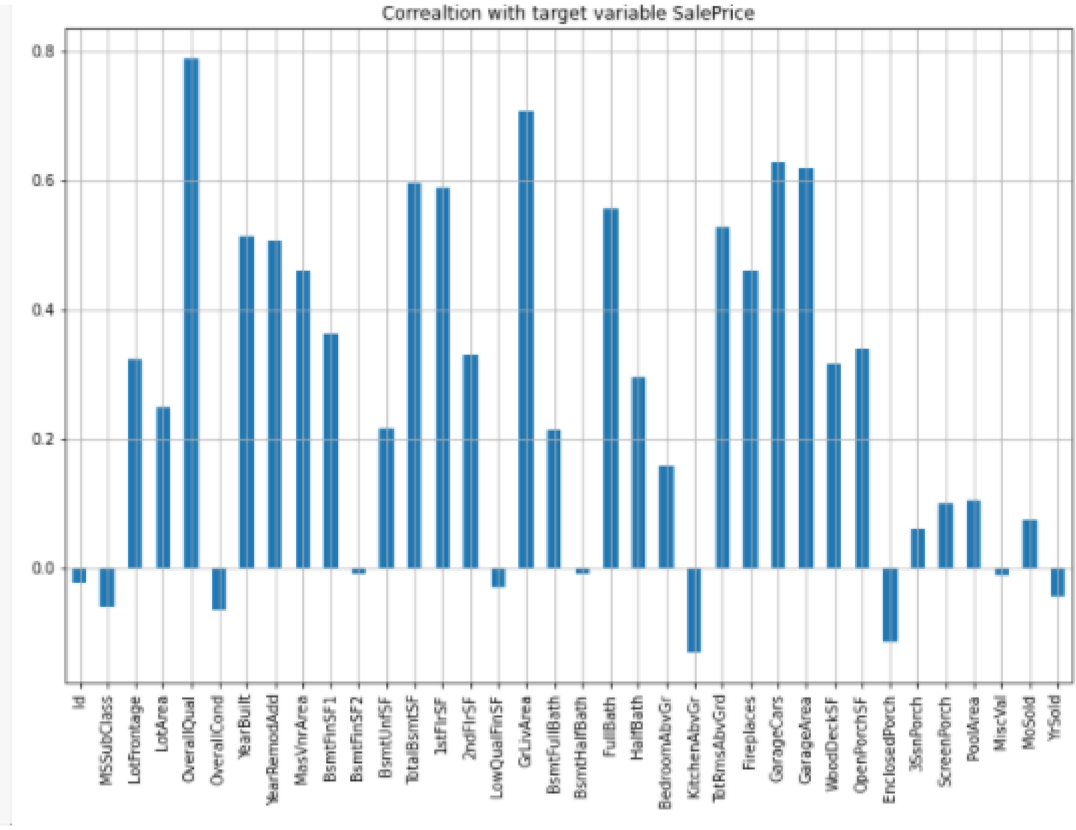
• SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch, YrSold.

• We observe multicollinearity in between columns so we will be using Principal Component Analysis(PCA).

• No correlation has been observed between the column Id and other columns so we will be dropping this column.

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

Here we check the correlation



between all our feature variables with target variable label

```
In [20]: # Let's check the correlation with target variable 'SalePrice'

         plt.figure(figsize=(12,8))
         HPP.drop('SalePrice', axis=1).corrwith(HPP['SalePrice']).plot(kind='bar',grid=True)
         plt.xticks(rotation='vertical')
         plt.title("Correaltion with target variable SalePrice");
```

1. The column OverallQual is most positively correlated with SalePrice.
2. The column KitchenAbvGrd is most negatively correlated with SalePrice.

Set of assumptions related to the problem under consideration

By looking into the target variable label we assumed that it was a Regression type of problem

We observed multicollinearity in between columns so we assumed that we will be using Principal Component Analysis (PCA).

We also observed that only one single unique value was present in Utilities column so we assumed that we will be dropping these columns.

LIBRARIES:

The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition pca, sklearn standardscaler, GridSearchCV, joblib

```
# Let's import all the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.pandas.set_option('display.max_columns',None)

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from scipy import stats

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV,cross_val_score
from sklearn.model_selection import GridSearchCV
```

Model Training

```
In [63]: HPP_x=HPP_cap.drop(columns=['SalePrice'],axis=1)
         y=HPP_cap['SalePrice']
```

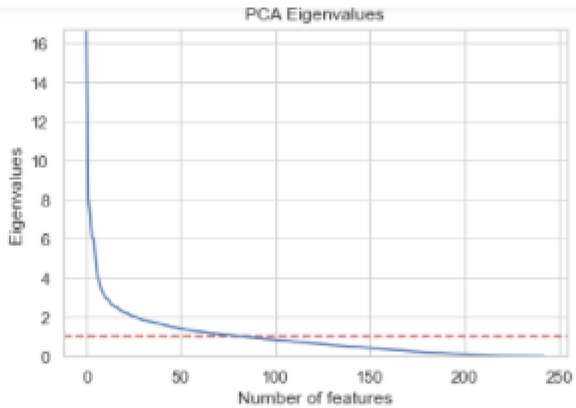
```
In [64]: #Scaling input variables

         sc=StandardScaler()
         x=sc.fit_transform(HPP_x)
         x=pd.DataFrame(x,columns=HPP_x.columns)
```

PCA

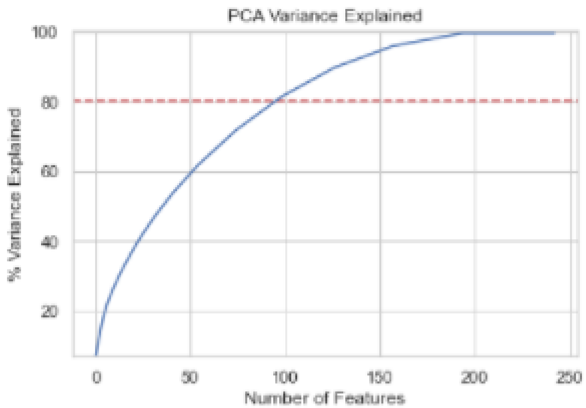
In [65]:
```
# Let's explore the PCA

covar_matrix = PCA(n_components = len(x.columns))
covar_matrix.fit(x)
```

Out[65]:
```
PCA(n_components=243)
```

In [66]:
```
# Let's plot the PCA componenets

plt.ylabel('Eigenvalues')
plt.xlabel('Number of features')
plt.title('PCA Eigenvalues')
plt.ylim(0,max(covar_matrix.explained_variance_))
plt.style.context('seaborn-whitegrid')
plt.axhline(y=1, color='r', linestyle='--')
plt.plot(covar_matrix.explained_variance_)
plt.show()
```




```
variance = covar_matrix.explained_variance_ratio_
var=np.cumsum(np.round(covar_matrix.explained_variance_ratio_, decimals=3)*100)

plt.ylabel('% Variance Explained')
plt.xlabel('Number of Features')
plt.title('PCA Variance Explained')
plt.ylim(min(var),100.5)
plt.style.context('seaborn-whitegrid')
plt.axhline(y=80, color='r', linestyle='--')
plt.plot(var)
plt.show()
```



In [68]:
```
pca=PCA(n_components=90)
xpca=pca.fit_transform(x)
x=xpca
```

In [69]:
```
pd.DataFrame(data=x)
```

Out[69]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	0.024203	-1.896922	0.132794	0.813523	-2.206812	-1.805705	1.036301	1.146785	0.747099	1.907002	2.588056	-1.809343	5.003214	3.043748	0.4
1	-2.247523	-4.219013	2.433476	2.469569	5.428540	2.217286	4.359156	-0.564535	-2.471974	0.703819	3.234971	5.223782	-1.644465	0.047039	-3.
2	-3.177185	-0.067083	0.034339	-0.530294	1.284232	-2.884103	1.488415	0.125741	0.740127	-1.442289	-1.459694	-0.113960	-1.068124	1.423319	-0.
3	-2.108227	-3.530378	1.215795	2.012101	1.144993	0.328716	-3.079922	-0.169071	1.556493	0.784475	1.017221	-0.783351	2.492130	2.439852	0.
4	-3.131148	-1.375622	0.344989	1.783482	0.115215	-0.337047	-0.857577	1.615359	-0.120048	-1.227154	1.756361	0.986905	0.270941	-0.923624	2.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	3.795603	-2.918648	-1.471846	-0.272985	-2.503341	0.282378	-1.206302	-0.261547	0.680012	0.524374	-1.350808	0.858826	-1.812401	1.273234	-0.
1164	4.015039	2.373317	10.993826	-4.930223	-3.243416	0.557147	0.472628	-1.427026	-1.053117	-0.060521	-0.836034	2.615087	-1.269243	-0.148323	1.4
1165	0.639938	-1.219592	-0.937026	-1.445349	-1.285664	-5.677711	0.848913	3.372065	1.121146	2.763902	4.436689	-3.371795	4.887224	1.984750	0.
1166	6.935101	2.136047	-2.251870	-2.370323	2.506504	1.338087	-0.219983	-0.662462	0.992586	-0.607953	0.007823	1.109680	1.092486	1.481054	0.4
1167	-3.748662	1.996965	-0.459527	-0.736009	-0.689898	-2.326322	1.362739	-1.770849	-0.709047	-0.560495	-0.129719	-0.468546	-0.274639	-0.351496	-0.

1168 rows × 90 columns

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

We first converted all our categorical variables to numeric variables with the help of dummy variables to checkout and dropped the columns which we felt were unnecessary.

We observed skewness in data so we tried to remove the skewness through treating outliers with winsorization technique.

The data was improper scaled so we scaled the feature variables on a single scale using sklearn's StandardScaler package.

There were too many (256) feature variables in the data so we reduced it to 100 with the help of Principal Component Analysis(PCA) by plotting Eigenvalues and taking the number of nodes as our number of feature variables.

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

```
model=[LinearRegression(),
       DecisionTreeRegressor(),
       KNeighborsRegressor(),
       SVR(),
       Lasso(),
       Ridge(),
       ElasticNet(),
       RandomForestRegressor(),
       AdaBoostRegressor(),
       GradientBoostingRegressor()
      ]
```

RUN AND EVALUATE SELECTED MODELS

```
In [72]: 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: 0.8273754247760959
Error:
Mean absolute error: 19625.799143842883
Mean squared error: 817812691.1519995
Root Mean Squared Error: 28597.42455452937
r2_score: 0.8537555186743268
**************************************************************


score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 24994.649572649574
Mean squared error: 1139709879.3760684
Root Mean Squared Error: 33759.58944323921
r2_score: 0.7961925976762323
**************************************************************


score of KNeighborsRegressor() is: 0.8022257430731456
Error:
Mean absolute error: 24354.53675213675
Mean squared error: 1474688918.5179489
Root Mean Squared Error: 38401.67858984746
r2_score: 0.736290328655108
**************************************************************


score of SVR() is: -0.046457743904219084
Error:
Mean absolute error: 51982.49977534046
Mean squared error: 5756951928.23989
Root Mean Squared Error: 75874.58025083164
r2_score: -0.0294807819334717
**************************************************************
```

```
score of Lasso() is: 0.8273754157832334
Error:
Mean absolute error: 19624.423182771403
Mean squared error: 817747339.8823696
Root Mean Squared Error: 28596.281924095827
r2_score: 0.8537672050453435
**************************************************************
```

```
score of Ridge() is: 0.8273753703888294
Error:
Mean absolute error: 19621.33208348357
Mean squared error: 817692776.1169267
Root Mean Squared Error: 28595.327872170423
r2_score: 0.8537769623524417
**************************************************************
```

```
score of ElasticNet() is: 0.8205255279535884
Error:
Mean absolute error: 18826.68624399566
Mean squared error: 826427770.0167346
Root Mean Squared Error: 28747.656774365707
r2_score: 0.8522149363945651
**************************************************************
```

```
score of RandomForestRegressor() is: 0.9690172427945005
Error:
Mean absolute error: 18936.740085470083
Mean squared error: 798801174.0261666
Root Mean Squared Error: 28263.07085272523
r2_score: 0.8571552329259668
**************************************************************
```

```
score of AdaBoostRegressor() is: 0.8434305821369604
Error:
Mean absolute error: 29121.03492192892
Mean squared error: 1509471811.9866998
Root Mean Squared Error: 38851.92159966737
r2_score: 0.73007031486787
**************************************************************
```

```
score of GradientBoostingRegressor() is: 0.9715662911419614
Error:
Mean absolute error: 18767.389040285903
Mean squared error: 740617571.1087067
Root Mean Squared Error: 27214.289832893064
r2_score: 0.867559853595691
**************************************************************
```

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

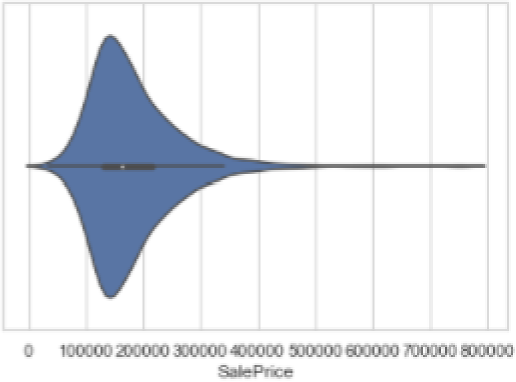
We used the metric Root Mean Squared Error by selecting the Ridge Regressor model which was giving us best(minimum) RMSE score

VISUALIZATIONS

Data Visualization

Univatriate Analysis

.]:
```
# Let's Check the target variable

sns.set(style='whitegrid')
sns.violinplot(HPP['SalePrice'])
plt.show()

HPP['SalePrice'].value_counts()
```

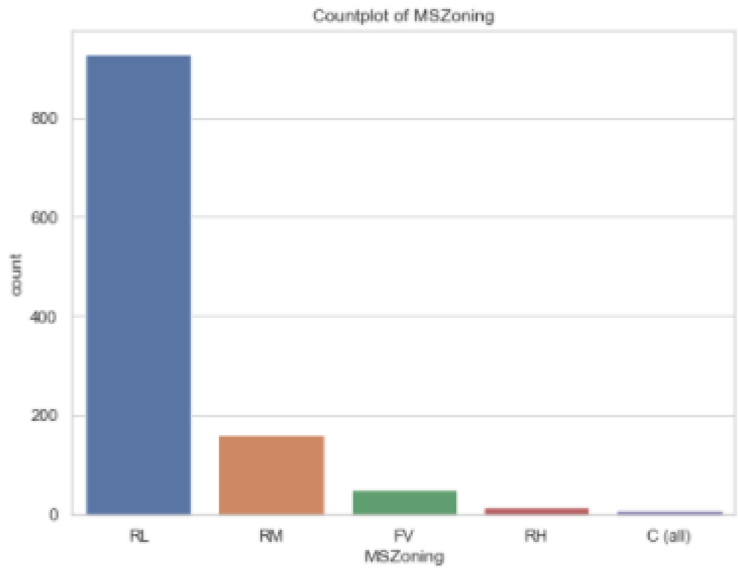


.]:
```
140000    18
135000    16
155000    12
139000    11
160000    11
          ..
126175     1
204000     1
186000     1
369900     1
105500     1
Name: SalePrice, Length: 581, dtype: int64
```

Observation:

```
Maximum number of SalePrice lies between 140000 and 230000.
```

```
22]: # Let's check the column MsZoning

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

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

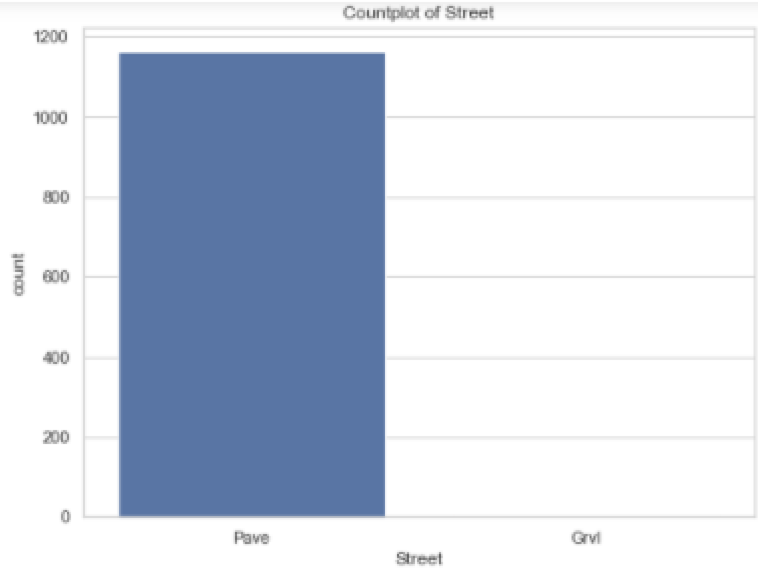



```
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
```

Observation:

```
Maximum, 928 number of MSZoning are RL.
```

```
# Let's check the column Street

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

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

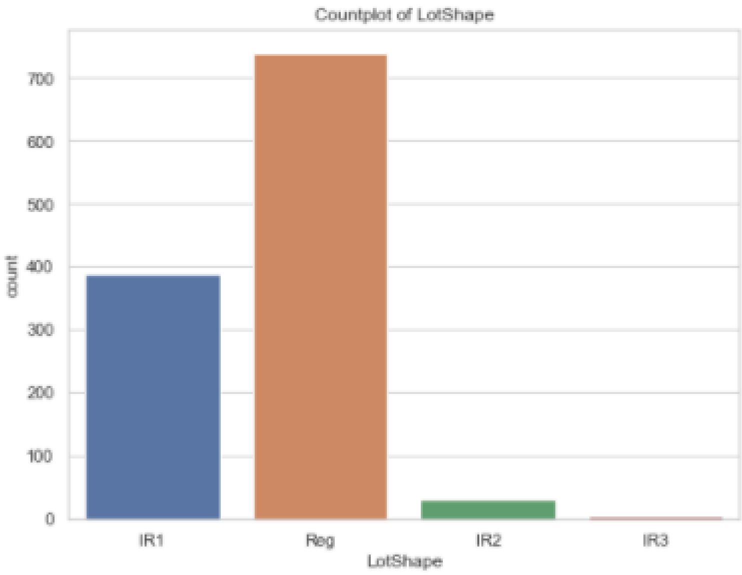



```
Pave    1164
Grvl       4
Name: Street, dtype: int64
```

Observation:

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

```
# Let's check the column LotShape

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

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

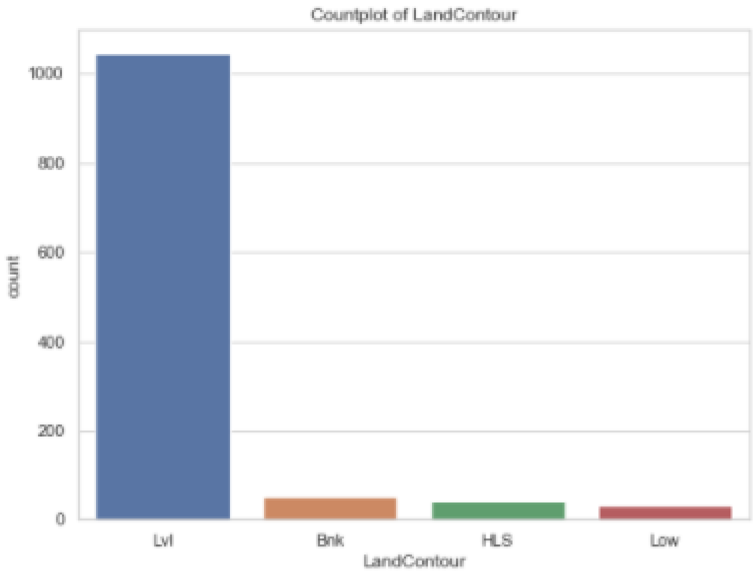



```
Reg     740
IR1     390
IR2      32
IR3       6
Name: LotShape, dtype: int64
```

Observation:

    Maximum, 740 number of LotShape are Reg.

]:
```
# Let's check the column LandContour

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

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

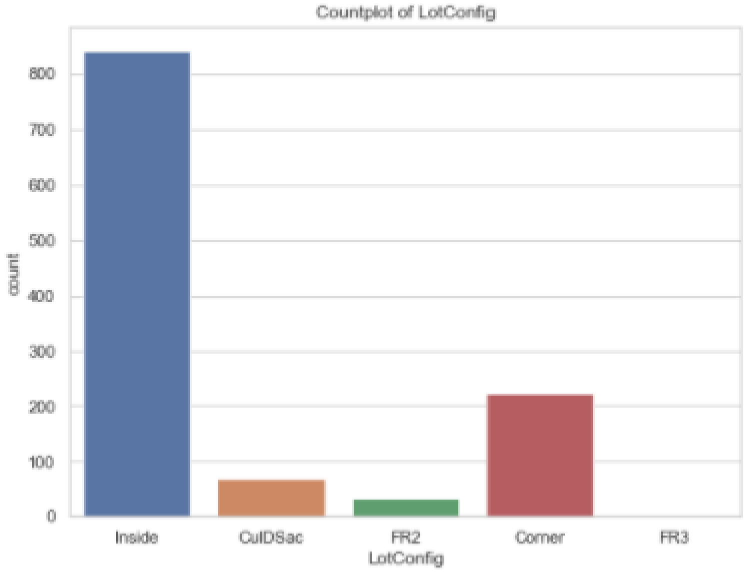


]:
```
Lvl    1046
Bnk      50
HLS      42
Low      30
Name: LandContour, dtype: int64
```

Observation:

    Maximum, 1046 number of LandContour are Lvl.

```
# Let's check the column LotConfig

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

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

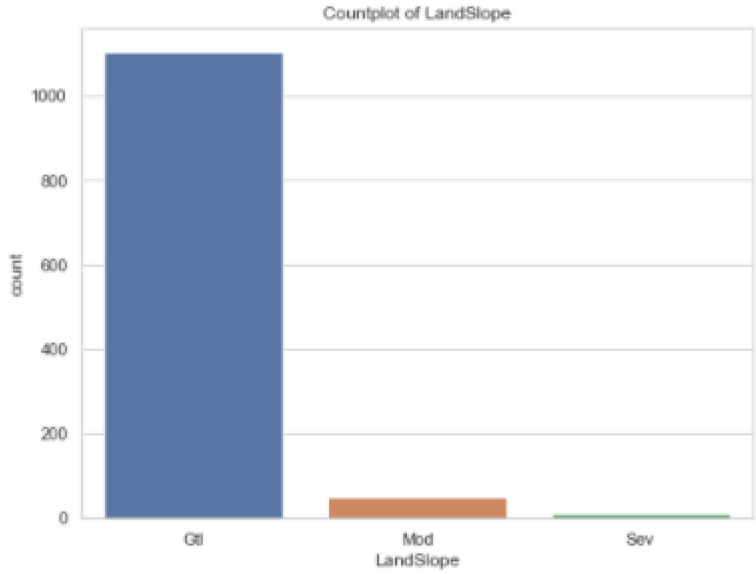



```
Inside     842
Corner     222
CulDSac     69
FR2         33
FR3          2
Name: LotConfig, dtype: int64
```

Observation:

    Maximum, 842 number of LotConfig are Inside.

```
# Let's check the column LandSlope

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

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

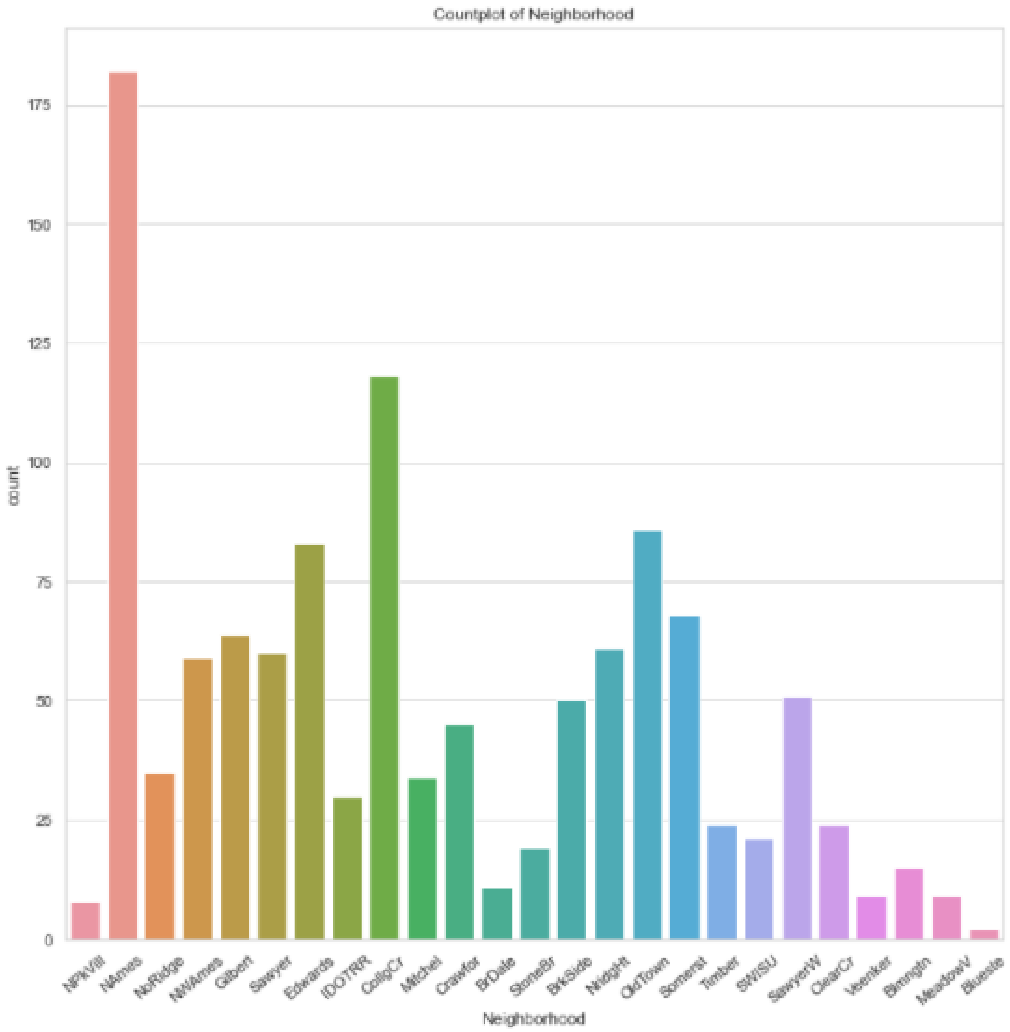



```
Gtl     1105
Mod       51
Sev       12
Name: LandSlope, dtype: int64
```

Observation:

    Maximum, 1105 number of LandSlope are Gtl.

```
# Let's check the column Neighborhood

plt.subplots(figsize=(12,12))
sns.countplot(x="Neighborhood", data=HPP)
plt.title("Countplot of Neighborhood")
plt.xticks(rotation=40)
plt.xlabel('Neighborhood')
plt.ylabel("count")
plt.show()

HPP['Neighborhood'].value_counts()
```



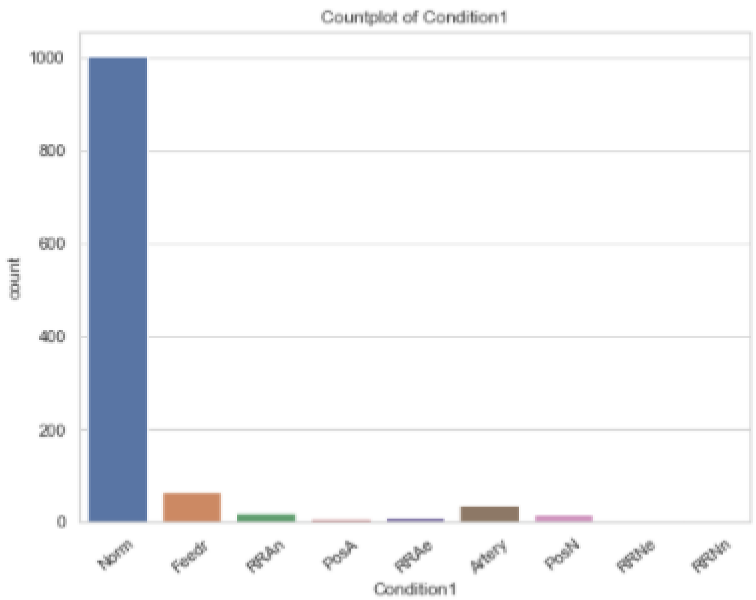
Countplot of Neighborhood
count
175
150
125
100
75
50
25
0
NPkVill
NAmes
NoRidge
NWAmes
Gilbert
Sawyer
Edwards
IDOTRR
CollgCr
Mitchel
Crawfor
BrDale
StoneBr
BrkSide
NridgHt
OldTown
Somerst
Timber
SWISU
SawyerW
ClearCr
Veenker
Blmngtn
MeadowV
Blueste
Neighborhood

```
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
Veenker      9
MeadowV      9
NPkVill      8
Blueste      2
Name: Neighborhood, dtype: int64
```

Observation:

```
Maximum, 182 number of Neighborhood are Names.
```

```python
# Let's check the column Condition1

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

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

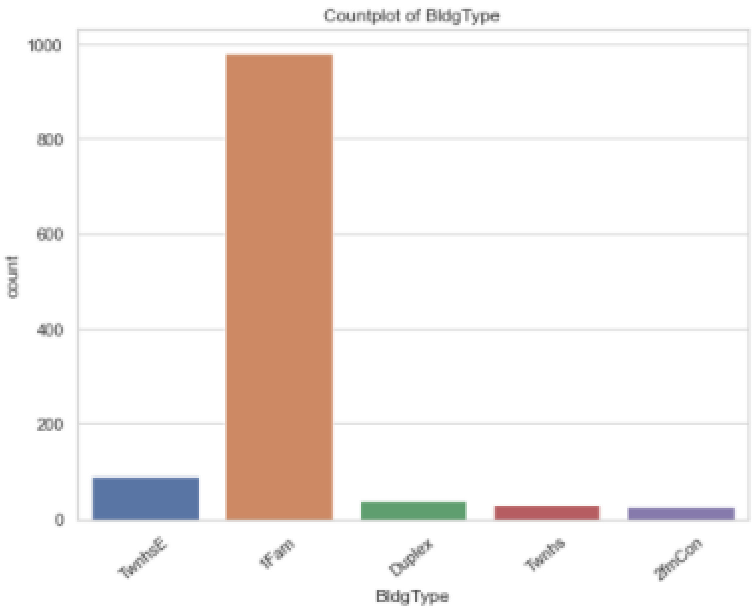


```
Norm      1005
Feedr       67
Artery      38
RRAn        20
PosN        17
RRAe         9
PosA         6
RRNn         4
RRNe         2
Name: Condition1, dtype: int64
```

Observation:

```
Maximum, 1005 number of Condition1 is Norm.
```

```
# Let's check the column BldgType

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

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

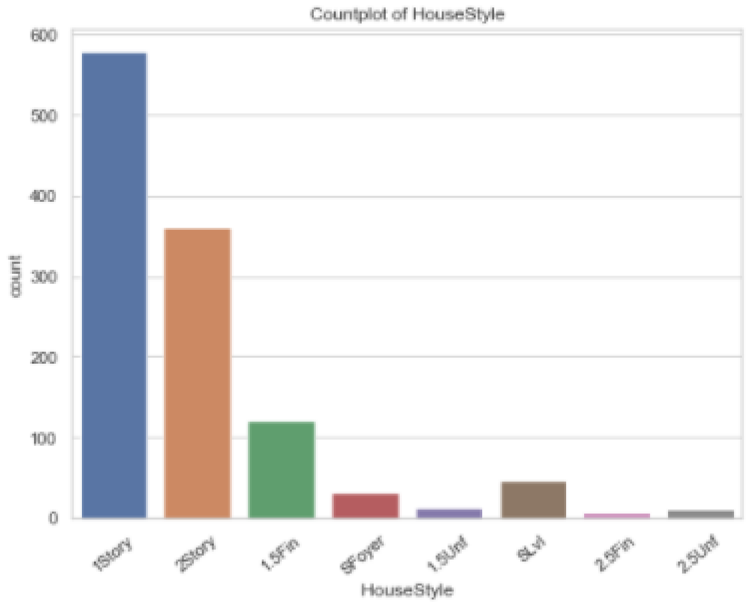



```
1Fam      981
TwnhsE     90
Duplex     41
Twnhs      29
2fmCon     27
Name: BldgType, dtype: int64
```

Observation:

```
Maximum, 981 number of BldgType are 1Fam.
```

```python
# Let's check the column HouseStyle

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

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

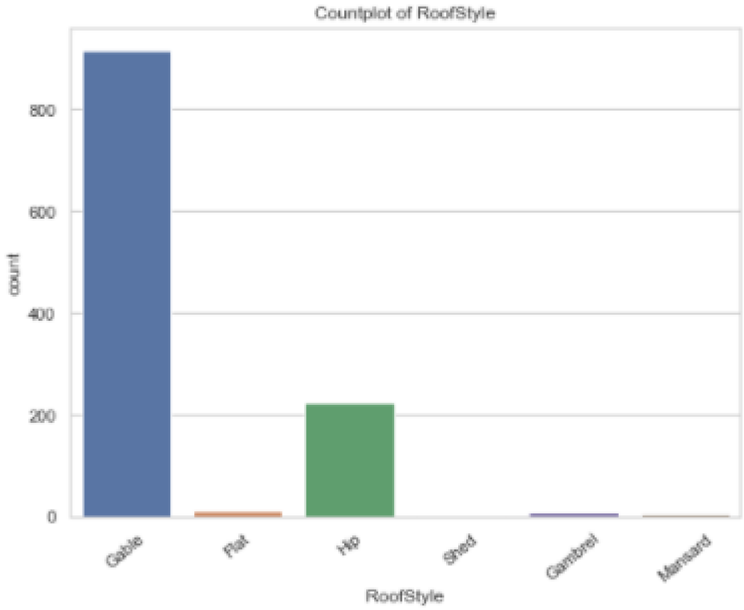



```
1Story    578
2Story    361
1.5Fin    121
SLvl       47
SFoyer     32
1.5Unf     12
2.5Unf     10
2.5Fin      7
Name: HouseStyle, dtype: int64
```

Observation:

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

```python
# Let's check the column RoofStyle

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

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

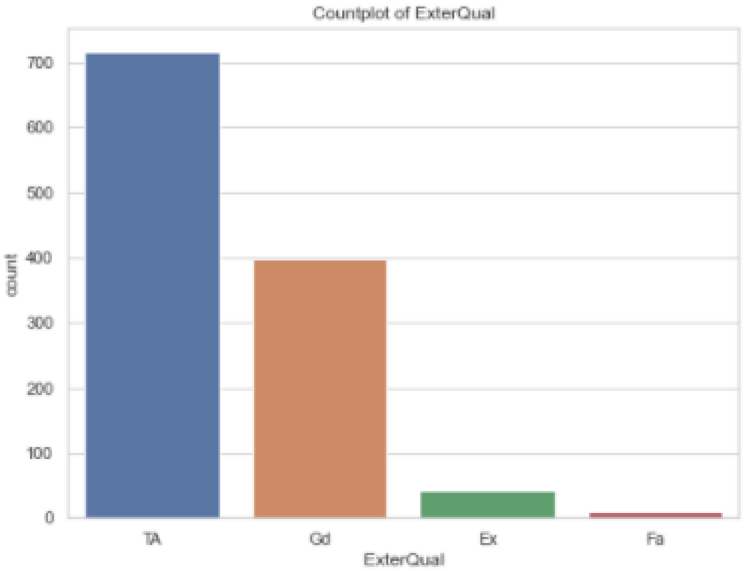



```
Gable      915
Hip        225
Flat        12
Gambrel      9
Mansard      5
Shed         2
Name: RoofStyle, dtype: int64
```

Observation:

    Maximum, 915 number of RoofStyle are Gable.

```
# Let's check the column ExterQual

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

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

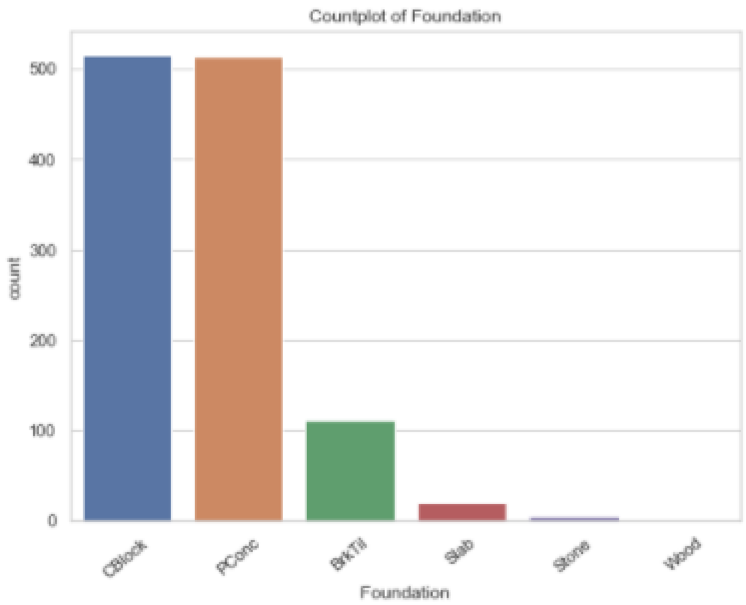



```
TA    717
Gd    397
Ex     43
Fa     11
Name: ExterQual, dtype: int64
```

Observation:

```
Maximum, 717 number of ExterQual is TA.
```

```
# Let's checking the column Foundation

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

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

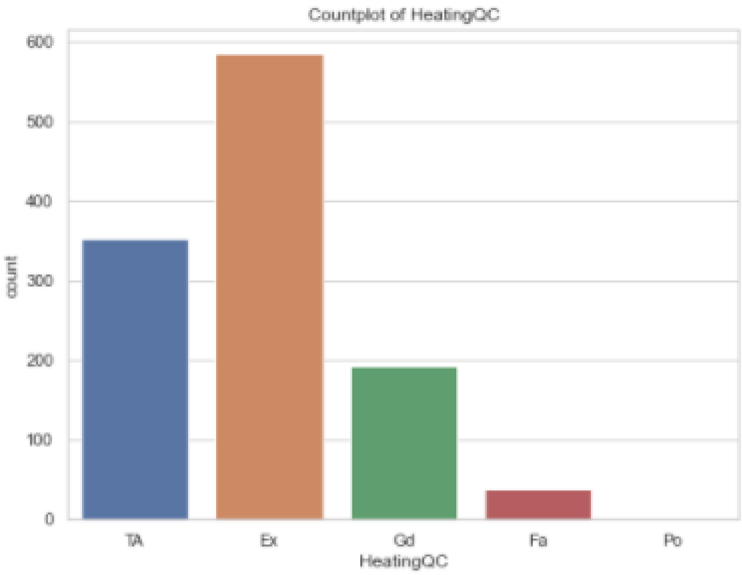



```
CBlock    516
PConc     513
BrkTil    112
Slab       21
Stone       5
Wood        1
Name: Foundation, dtype: int64
```

Observation:

    Maximum, 516 number of Foundation are CBlock.

```
]: # Let's check the column HeatingQC

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

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

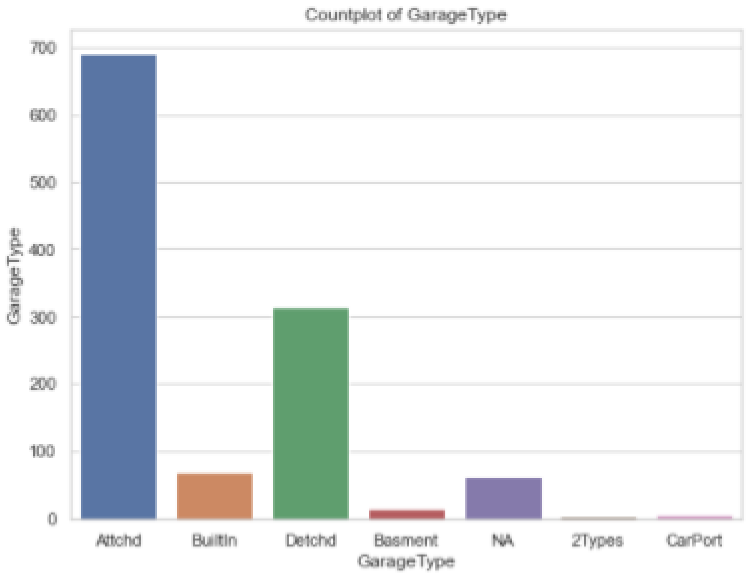



```
Ex    585
TA    352
Gd    192
Fa     38
Po      1
Name: HeatingQC, dtype: int64
```

Observation:

    Maximum, 585 number of HeatingQC is Ex.

```
# Let's check the column GarageType

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

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

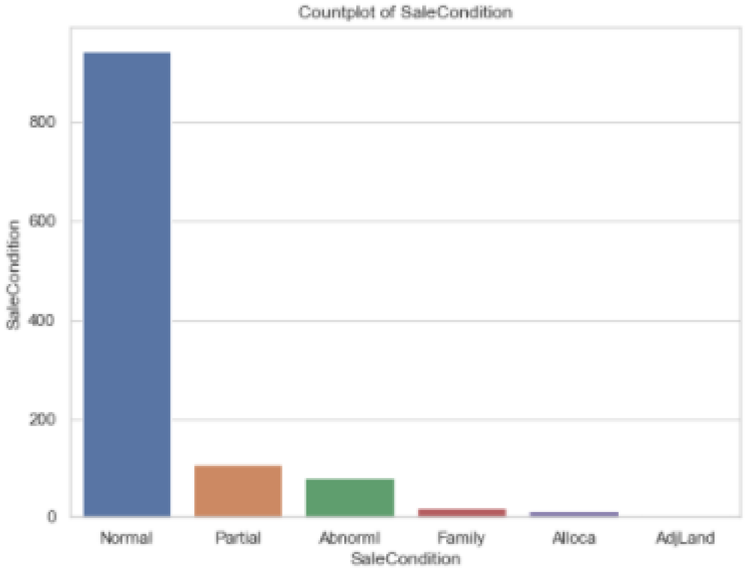



```
Attchd     691
Detchd     314
BuiltIn     70
NA          64
Basment     16
CarPort      8
2Types       5
Name: GarageType, dtype: int64
```

Observation:

```
Maximum, 691 number of GarageType are Attchd.
```

```python
# Let's check the column SaleCondition

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

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

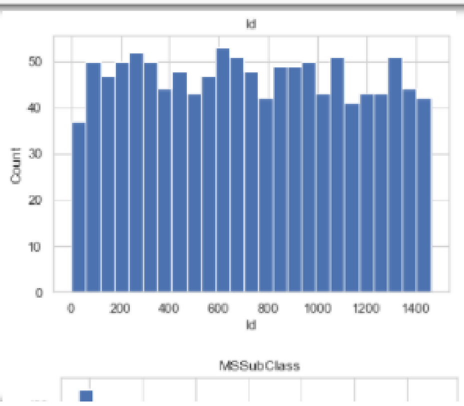



```
Normal     945
Partial    108
Abnorml     81
Family      18
Alloca      12
AdjLand      4
Name: SaleCondition, dtype: int64
```

Observation:

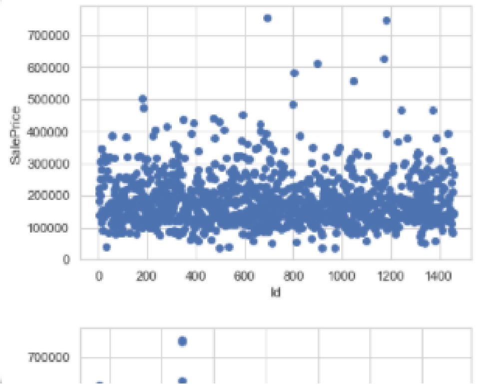
```
Maximum, 945 number of SaleCondition is normal.
```

```python
# Let's plot the histogram of every numerical column

for col in HPP.describe().columns:
    data=HPP.copy()
    data[col].hist(bins=25)
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.title(col)
    plt.show()
```



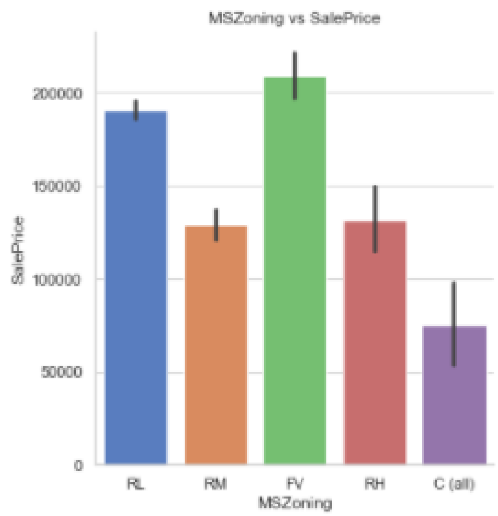
Bivariate Analysis

```
# Let's plot the Scatter plot between all feature variables and target variable

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




```
# Let's plot the Factor plot of MSZoning vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='MSZoning',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('MSZoning vs SalePrice')
plt.ylabel('SalePrice')
plt.show()
print(HPP.groupby('SalePrice')['MSZoning'].value_counts());
```

```
<Figure size 576x432 with 0 Axes>
```



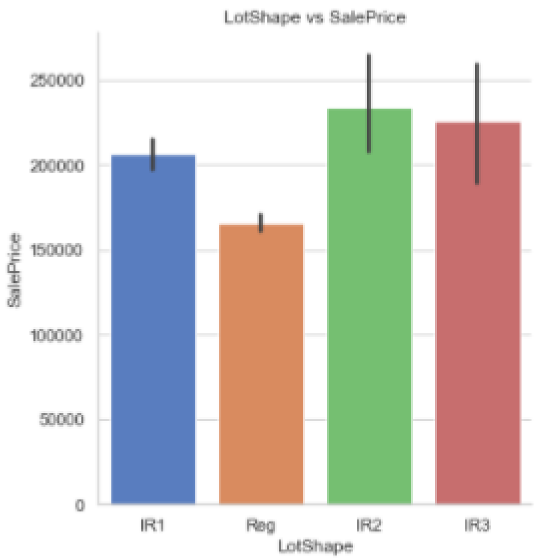
```
SalePrice  MSZoning
34900      C (all)     1
35311      C (all)     1
37900      RM          1
39300      RL          1
40000      C (all)     1
                      ..
582933     RL          1
611657     RL          1
625000     RL          1
745000     RL          1
755000     RL          1
Name: MSZoning, Length: 697, dtype: int64
```

Observation:

    SalePrice is maximum with FV MSZOning.

```python
# Let's plot the Factor plot of LotShape vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='LotShape',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('LotShape vs SalePrice')
plt.ylabel('SalePrice')
plt.show();
print(HPP.groupby('SalePrice')['LotShape'].value_counts());
```

```
<Figure size 576x432 with 0 Axes>
```



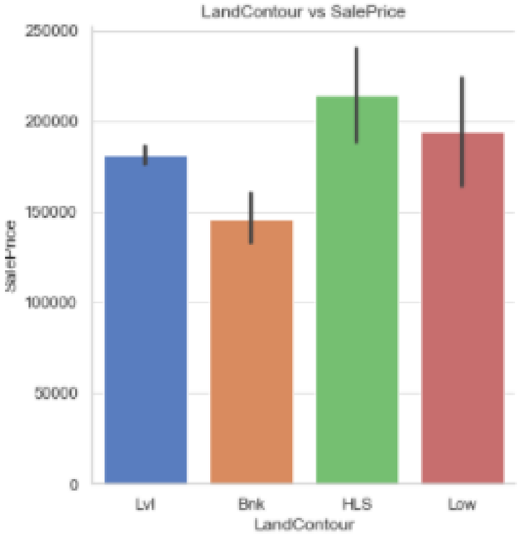

```
SalePrice  LotShape
34900      Reg         1
35311      Reg         1
37900      Reg         1
39300      Reg         1
40000      Reg         1
                      ..
582933     Reg         1
611657     IR1         1
625000     IR1         1
745000     IR1         1
755000     IR1         1
Name: LotShape, Length: 733, dtype: int64
```

bservation:

    SalePrice is maximum with IR2 LotShape.

```
 Let's plot the Factor plot of LandContour vs SalePrice

lt.figure(figsize=(8,6))
ns.factorplot(x='LandContour',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
lt.title('LandContour vs SalePrice')
lt.ylabel('SalePrice')
lt.show()
rint(HPP.groupby('SalePrice')['LandContour'].value_counts())
```

```
Figure size 576x432 with 0 Axes>
```



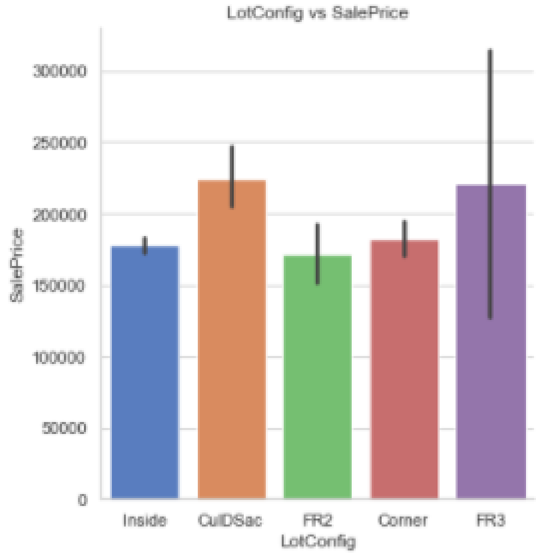

```
SalePrice  LandContour
34900      Lvl            1
35311      Lvl            1
37900      Lvl            1
39300      Low            1
40000      Lvl            1
                         ..
582933     Lvl            1
611657     Lvl            1
625000     Lvl            1
745000     Lvl            1
755000     Lvl            1
Name: LandContour, Length: 655, dtype: int64
```

Observation:

    SalePrice is maximum with HLS LandContour.

```
# Let's plot the Factor plot of LotConfig vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='LotConfig',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('LotConfig vs SalePrice')
plt.ylabel('SalePrice')
plt.show()

print(HPP.groupby('SalePrice')['LotConfig'].value_counts())
```

<Figure size 576x432 with 0 Axes>



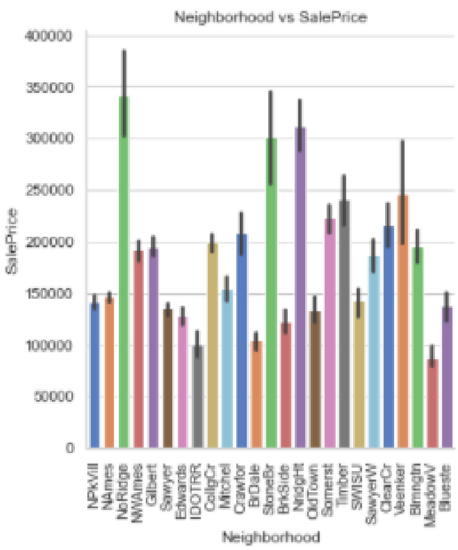

```
SalePrice  LotConfig
34900      Inside       1
35311      Inside       1
37900      Inside       1
39300      Inside       1
40000      Inside       1
                       ..
582933     Inside       1
611657     Inside       1
625000     CulDSac      1
745000     Corner       1
755000     Corner       1
Name: LotConfig, Length: 743, dtype: int64
```

Observation:

    SalePrice is maximum with CulDsac LotConfig.

```
# Let's plo the Factor plot of Neighborhood vs SalePrice

plt.figure(figsize=(16,16))
sns.factorplot(x='Neighborhood',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('Neighborhood vs SalePrice')
plt.xticks(rotation='vertical')
plt.ylabel('SalePrice')
plt.show()

print(HPP.groupby('SalePrice')['Neighborhood'].value_counts())
```

```
<Figure size 1152x1152 with 0 Axes>
```

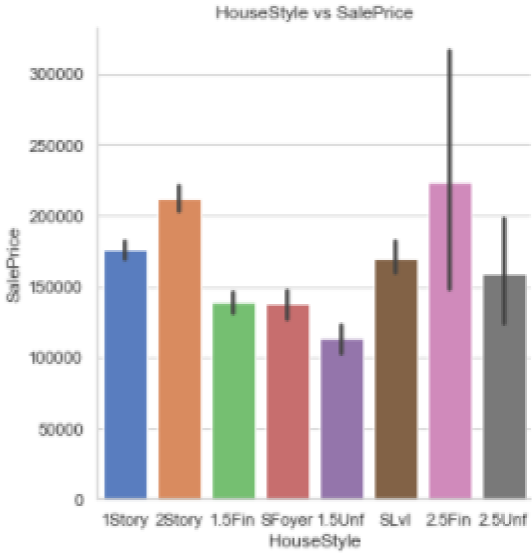



```
SalePrice  Neighborhood
34900      IDOTRR          1
35311      IDOTRR          1
37900      OldTown         1
39300      BrkSide         1
40000      IDOTRR          1
                          ..
582933     NridgHt         1
611657     NridgHt         1
625000     NoRidge         1
745000     NoRidge         1
755000     NoRidge         1
Name: Neighborhood, Length: 1013, dtype: int64
```

Observation:

    SalePrice is maximum with NoRidge Neighborhood.

```
# Let's plot the Factor plot of HouseStyle vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='HouseStyle',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('HouseStyle vs SalePrice')
plt.ylabel('SalePrice')
plt.show()

print(HPP.groupby('SalePrice')['HouseStyle'].value_counts())
```



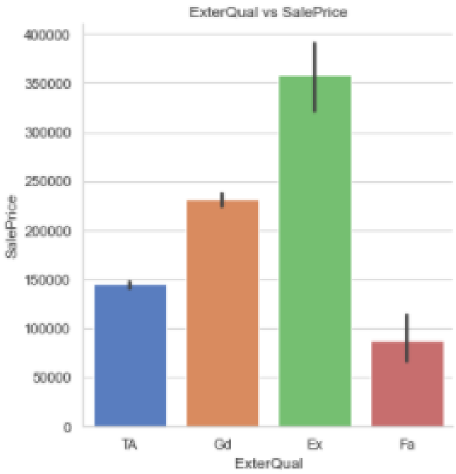

```
SalePrice  HouseStyle
34900      1Story        1
35311      1Story        1
37900      1.5Fin        1
39300      1Story        1
40000      2Story        1
                        ..
582933     2Story        1
611657     1Story        1
625000     2Story        1
745000     2Story        1
755000     2Story        1
Name: HouseStyle, Length: 840, dtype: int64
```

Observation:

    SalePrice is maximum with 2.5Fin HouseStyle.

```python
# Let's plot the Factor plot of ExterQual vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='ExterQual',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('ExterQual vs SalePrice')
plt.ylabel('SalePrice')
plt.show()

print(HPP.groupby('SalePrice')['ExterQual'].value_counts())
```

<Figure size 576x432 with 0 Axes>

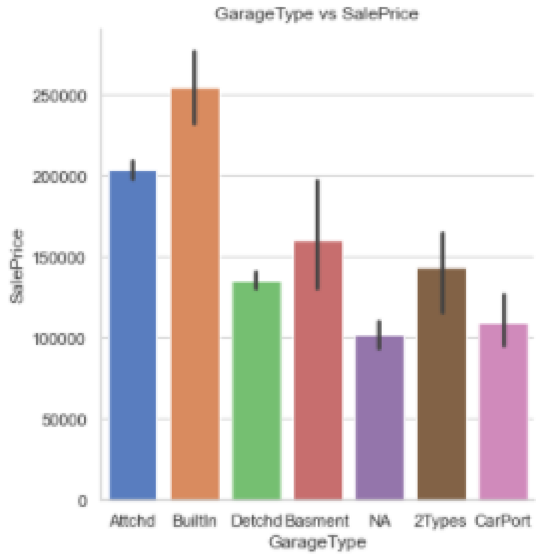



```
SalePrice  ExterQual
34900      TA           1
35311      TA           1
37900      TA           1
39300      Fa           1
40000      TA           1
                       ..
582933     Ex           1
611657     Ex           1
625000     Gd           1
745000     Gd           1
755000     Ex           1
Name: ExterQual, Length: 679, dtype: int64
```

Observation:

    SalePrice is maximum with Ex ExterQual.

```python
# Let's plot the Factor plot of GarageType vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='GarageType',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('GarageType vs SalePrice')
plt.ylabel('SalePrice')
plt.show()


print(HPP.groupby('SalePrice')['GarageType'].value_counts())
```



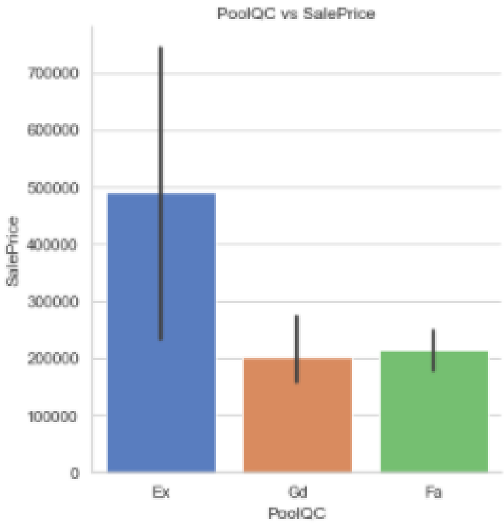

```
SalePrice  GarageType
34900      NA            1
35311      Detchd        1
37900      NA            1
39300      NA            1
40000      Detchd        1
                        ..
582933     BuiltIn       1
611657     Attchd        1
625000     Attchd        1
745000     Attchd        1
755000     Attchd        1
Name: GarageType, Length: 762, dtype: int64
```

Observation:

    SalePrice is maximum with Builtin GarageType.

```python
# Let's plot the Factor plot of PoolQC vs SalePrice

plt.figure(figsize=(8,6))
sns.factorplot(x='PoolQC',y='SalePrice',data=HPP,kind='bar',size=5,palette='muted',aspect=1)
plt.title('PoolQC vs SalePrice')
plt.ylabel('SalePrice')
plt.show()

print(HPP.groupby('SalePrice')['PoolQC'].value_counts())
```

```
<Figure size 576x432 with 0 Axes>
```




```
SalePrice  PoolQC
160000     Gd        1
171000     Gd        1
181000     Fa        1
235000     Ex        1
250000     Fa        1
274970     Gd        1
745000     Ex        1
Name: PoolQC, dtype: int64
```

Observation:

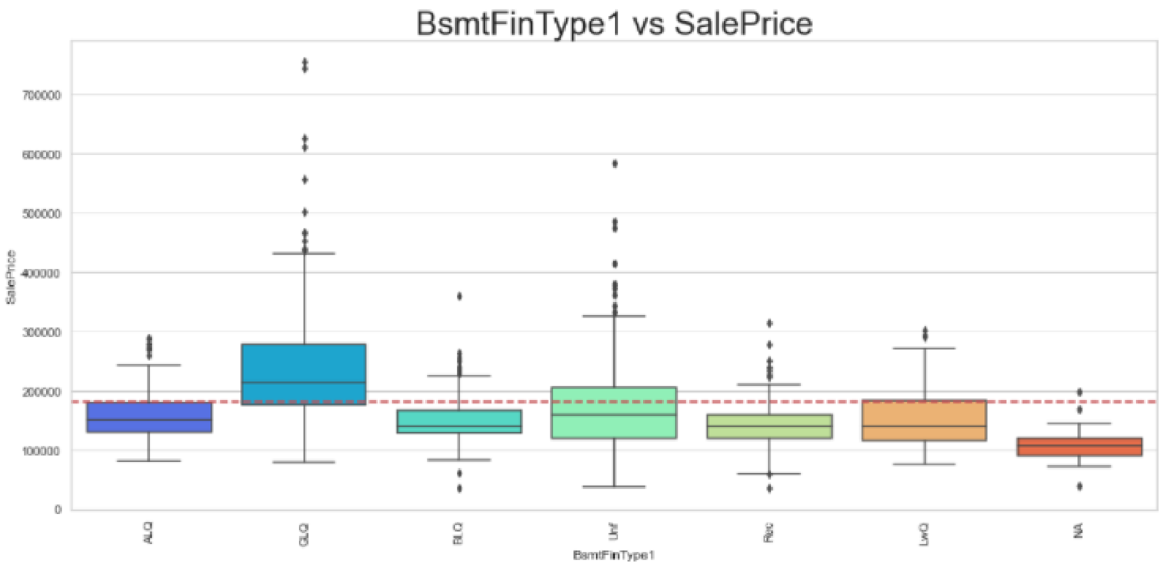
    SalePrice is maximum with Ex PoolQC.

```python
# Let's plot the Foundation vs SalePrice plot

plt.figure(figsize=(18,8))
mean_price=np.mean(HPP['SalePrice'])
sns.boxplot(y='SalePrice',x='Foundation',data=HPP,palette="rainbow")
plt.axhline(mean_price,color='r',linestyle='dashed',linewidth=2)
plt.title("Foundation vs SalePrice",fontsize=30)
plt.xticks(rotation='vertical')
plt.show()
```

Observation:

```
SalePrice is maximum with PConc.
```

```
# Let's plot the BsmtFinType1 vs SalePrice plot

plt.figure(figsize=(18,8))
mean_price=np.mean(HPP['SalePrice'])
sns.boxplot(y='SalePrice',x='BsmtFinType1',data=HPP,palette="rainbow")
plt.axhline(mean_price,color='r',linestyle='dashed',linewidth=2)
plt.title("BsmtFinType1 vs SalePrice",fontsize=30)
plt.xticks(rotation='vertical')
plt.show()
```



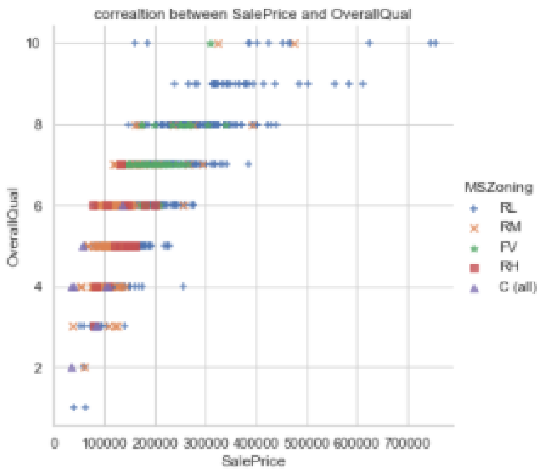
Observation:

    SalePrice is maximum with GLQ BsmtFinType1.

Multivariate Analysis

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

plt.figure(figsize=(14,14))
sns.lmplot(x='SalePrice',y='OverallQual',fit_reg=False,data=HPP,hue='MSZoning',markers=['+','x','*','s','^'])
plt.xlabel('SalePrice')
plt.title('correaltion between SalePrice and OverallQual')
plt.ylabel('OverallQual')
plt.show()
```

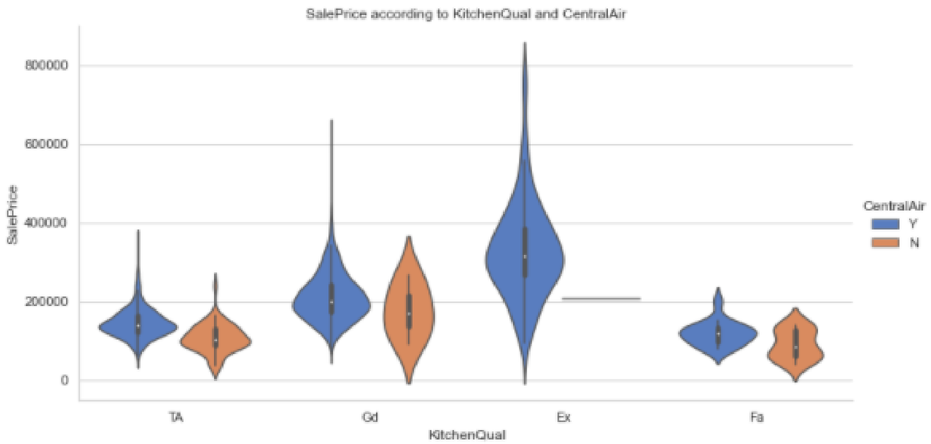
<Figure size 1008x1008 with 0 Axes>



Observation:

    With MSZoning RL and increase in OverallQual the SalePrice of a house increases.

```
]: # Let's plot the GarageType and GarageCond with respect to SalePrice plot

sns.factorplot(x='KitchenQual',y='SalePrice',hue='CentralAir',data=HPP,kind='violin',size=5,palette='muted',aspect=2)
plt.title('SalePrice according to KitchenQual and CentralAir')
plt.xticks()
plt.ylabel('SalePrice')
plt.show()
```



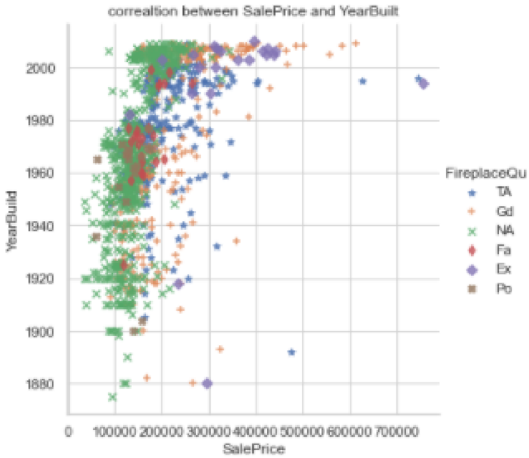
Observation:

    SalePrice is maximum with Ex kitchenQual and CentralAir.

]:

```
# 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=HPP,hue='FireplaceQu',markers=['*','+','x','d','D','X'])
plt.xlabel('SalePrice')
plt.title('correaltion between SalePrice and YearBuilt')
plt.ylabel('YearBuild')
plt.show()
```

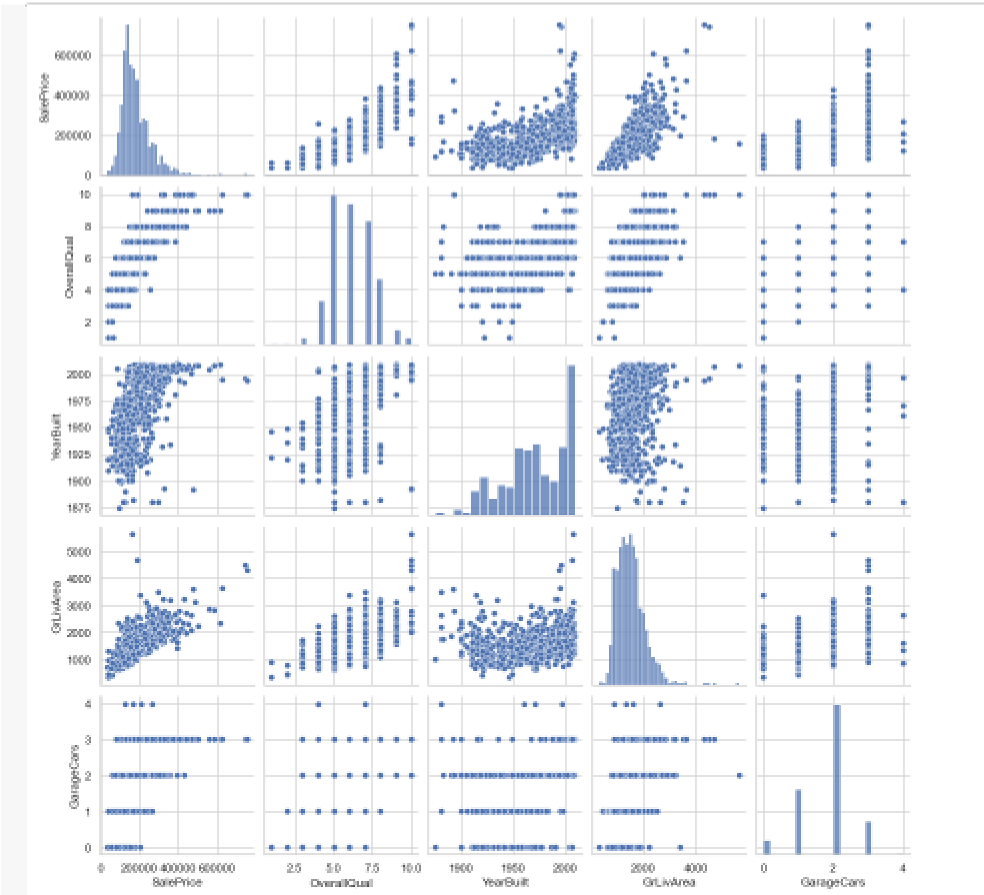
<Figure size 1008x1008 with 0 Axes>



Observation:

As the YearBuilt is increasing SalePrice is also increasing.

```
# Let's plot the pairplot

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



Observation:

SalePrice is highly positively correlated with GrLivArea and OverallQual.

INTERPRETATION OF THE RESULTS

From the visualization we interpreted that the target variable SalePrice was highly positively correlated with the columns GrLivArea, YearBuilt, OverallQual, GarageCars, GarageArea.

From the preprocessing we interpreted that data was improper scaled.

Hyperparameter tuning

```
# 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}
```

```
# 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: 0.8273426054003025


Mean absolute error: 19519.415803482367
Mean squared error: 815131516.6463553
Root Mean Squared error: 28550.50816791805


r2_score: 0.8542349768426382
```

From the modeling we interpreted that after hyperparameter tuning Ridge Regressor works best with respect to our model with minimum RMSE of 28550

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

In this project we have tried to show how the house prices vary and what are the factors related to the changing of house prices. The best(minimum) RMSE score was achieved using the best parameters of Ridge Regressor through GridSearchCV though Gradient Boosting Regressor model performed well too.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

This project has demonstrated the importance of sampling effectively, modelling and predicting data. Through different powerful tools of visualization we were able to analyse and interpret different hidden insights about the data. Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

The few challenges while working on this project where:-

- Improper scaling
- Too many features

- Missing values
- Skewed data due to outliers

The data was improper scaled so we scaled it to a single scale using sklearns's package StandardScaler.

There were too many(256) features present in the data so we applied Principal Component Analysis(PCA) and found out the Eigenvalues and on the basis of number of nodes we were able able to reduce our features upto 90 columns.

There were lot of missing values present in different columns which we imputed on the basis of our understanding.

The columns were skewed due to presence of outliers which we handled through winsorization technique.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

While we couldn't reach out goal of minimum RMSE in house price prediction without letting the model to overfit, we did end up creating a system that can with enough time and data get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.

THANKYOU