FLIP ROBO

HOUSING PRICE PREDICTION PROJECT

Submitted by:

Niti Marodia

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to all the Mentors who have taught me Machine Learning because of the knowledge they had provided to me I am able to complete this project.

INTRODUCTION

- ## Business Problem Framing

  Housing and Real estate Markets are one of the major contributors in a country's economy.It is very large market and various companies are working in this domain. Data Science can play a vital role in solving problems related to this domain and can help the countries in their overall revenue, profits and improving their marketing strategies. Machine learning techniques can be used for achieving business goals for this housing companies. Our Problem is related to one of such U.S based housing company named Surprise Housing which want to enter Australian Market. The company want to use Data Analytics to purchase houses at a price below their actual values and flip them at a higher price. The company has collected a dataset from the sale of houses in Australia. The company is looking at prospective properties to buy houses to enter the market. we will build a model using Machine Learning to predict the actual value of the prospective properties and it will help the company to decide whether to invest in property or not.

- ## Conceptual Background of the Domain Problem

  Trends in housing prices indicate the current economic situation and also are a concern to the buyers and sellers. There are many factors that have an impact on house prices, such as the number of bedrooms and bathrooms. House price depends upon its location as well. A house with great accessibility to highways, schools, malls, employment opportunities, wouldhave a greater price as compared to a house with no such accessibility. Predicting house prices manually is a difficult task and generally not very accurate, hence there are many systems developed for house price prediction.

- ## Review of Literature

  The world is shifting from manual to automated systems. The objective of our project is to reduce the problems faced by the customer. In the present situation, the customer visits a real estate agent so that he/she can suggest suitable showplaces for his investments. But theabove method is risky as the agent may forecast wrong prices to the customer and that will lead to loss of customer's investment. This manual technique which is currently used in the market is outdated and has a high risk. So as to overcome the drawback, there is a need for an updated and automated system. So we are using machine learning techniques where we will be using different algorithms for this project to get the accurate prediction for the price of the house.

  Machine learning is a form of artificial intelligence which compose available computers with the efficiency to be trained without being veraciously programmed. Machine learning interest on the extensions of computer programs which is capable enough to modify when unprotected to new-fangled data. Machine learning algorithms are broadly classified into three divisions, namely; Supervised learning, Unsupervised learning and Reinforcement learning.

  Supervised learning is a learning in which we teach or train the machine using data which is well labelled that means some data is already tagged with correct answer. After that, machine is provided with new set of examples so that supervised learning algorithm analyses the training data and produces a correct outcome from labelled data

- Undertaken

Growing unaffordability of housing has become one of the major challenges for countries around the world. In order to gain a better understanding of the commercialized housing market we are currently facing; we want to figure out what are the top influential factors of the housing price. Apart from the more obvious driving forces such as the inflation and the scarcity of land, there are also a number of variables that are worth looking into. Therefore, we choose to study the house price prediction., which enables us to dig into the variables in depth and to provide a model that could more accurately estimate house prices. In this way, people could make better decision when it comes to home investment.

Our objective is to discuss the major factors that affect housing price and make precise prediction for it. We use 80 explanatory variables including almost every aspect of residential homes in Australia. Methods of both statistical, regression models and machine learning models are applied and further compared according to their performance to better estimate the final price of each house. The model provides price prediction based on similar comparable of people's dream house, which allow both buyers and sellers to better negotiate home prices according to market trend.

Analytical Problem Framing

- Data Sources and their formats

In this project we are given two CSV file containing train and test dataset of sales of houses in Australia.

There are 80 columns and our target variable is to predict Sale price.
Below are the descriptions of the columns:

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

limitsCondition1: 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 conditionCentralAir: 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 feetGarageQual: 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

qualityFence: 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

- # Mathematical/ Analytical Modeling of the Problem

First of all we will load necessary libraries and then will load our HOUSING.csv file

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df=pd.read_csv("train.csv")
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
df.head(10)
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
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
5	1197	60	RL	58.0	14054	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm
6	561	20	RL	NaN	11341	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm
7	1041	20	RL	88.0	13125	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	Sawyer	Norm
8	503	20	RL	70.0	9170	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	Edwards	Feedr
9	576	50	RL	80.0	8480	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm

Now we will check the columns and shape of the dataset

```
df.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

```
df.shape
```

```
(1168, 81)
```

Simultaneously with the same process we will check our test data set.

```
df.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition'],
      dtype='object')
```

```
df.shape
```

```
(292, 80)
```

From the above visualization we can see that we have 1168 rows and 81 columns in traindata set and 292 rows and 80 columns in test data set.

Let’s check null values in both the data set.

```
# Here we will check the percentage of nan values present in each feature
features_with_na=[features for features in df.columns if df [features].isnull().sum()>1]

for feature in features_with_na:
    print(feature, np.round(df[feature].isnull().mean(),4), ' % missing values')
```

```
LotFrontage 0.1832  % missing values
Alley 0.9341  % missing values
MasVnrType 0.006  % missing values
MasVnrArea 0.006  % missing values
BsmtQual 0.0257  % missing values
BsmtCond 0.0257  % missing values
BsmtExposure 0.0265  % missing values
BsmtFinType1 0.0257  % missing values
BsmtFinType2 0.0265  % missing values
FireplaceQu 0.4717  % missing values
GarageType 0.0548  % missing values
GarageYrBlt 0.0548  % missing values
GarageFinish 0.0548  % missing values
GarageQual 0.0548  % missing values
GarageCond 0.0548  % missing values
PoolQC 0.994  % missing values
Fence 0.7971  % missing values
MiscFeature 0.9623  % missing values
```

Here we can see that there are many null values present in train dataset. Some columns contains more that 80% of the null values.

Let’s go with test dataset.

```
# Here we will check the percentage of nan values present in each feature
features_with_na=[features for features in test_df.columns if test_df [features].isnull().sum()>1]

for feature in features_with_na:
    print(feature, np.round(test_df[feature].isnull().mean(),4), ' % missing values')
```

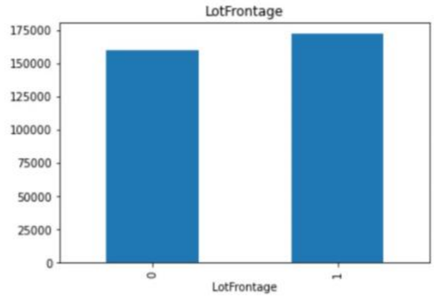
```
LotFrontage 0.1541  % missing values
Alley 0.9521  % missing values
BsmtQual 0.024  % missing values
BsmtCond 0.024  % missing values
BsmtExposure 0.024  % missing values
BsmtFinType1 0.024  % missing values
BsmtFinType2 0.024  % missing values
FireplaceQu 0.476  % missing values
GarageType 0.0582  % missing values
GarageYrBlt 0.0582  % missing values
GarageFinish 0.0582  % missing values
GarageQual 0.0582  % missing values
GarageCond 0.0582  % missing values
PoolQC 1.0  % missing values
Fence 0.8493  % missing values
MiscFeature 0.9658  % missing values
```

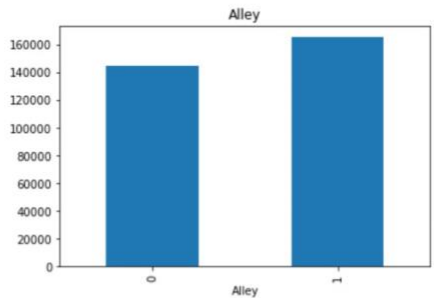
The same can be seen in test data set also.

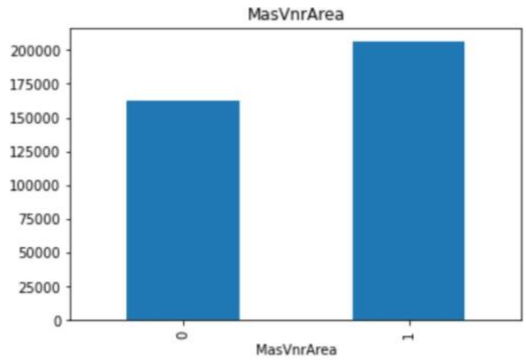
- # Visualization

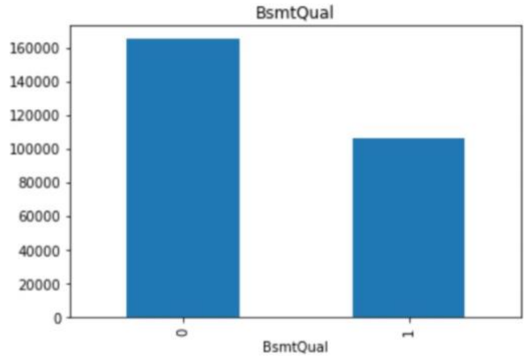
Since their are many missing values, we need to find the relationship betweenmissing values and Sales price

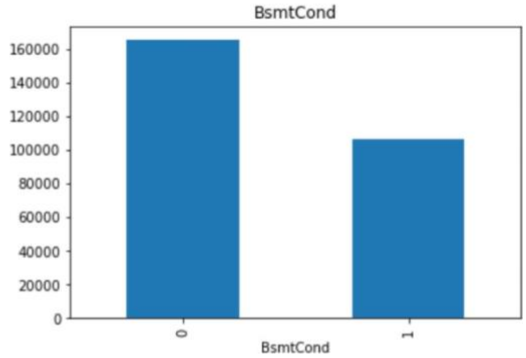
```
for feature in features_with_na:
    data= df.copy()

    #Let's make a variable that indicate 1 if the observation was missing or zero if its not
    data[feature] = np.where(data[feature].isnull(),1 ,0)

    #Let's calculate the mean Sale Price where the information is missing or present
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.title(feature)
    plt.show()
```









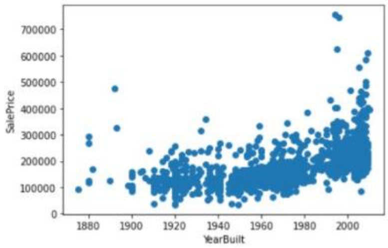


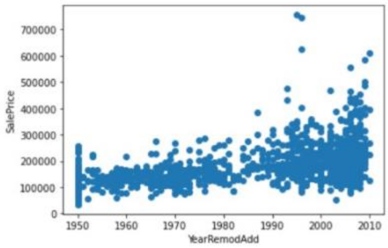
Here variable 1 indicates missing and 0 indicates its not

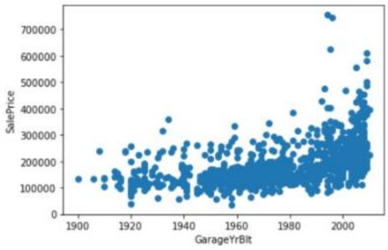
With the relation between the missing values and the dependent variable is clearly visible.So We need to replace these nan values with something meaningful.

we can see that features like lotfrontage, alley has nan values because of this the mediansale price is increasing so we will replace it with some meaningful later on.

```
#Here we will compare the difference between All years features with Sale Price

for feature in year_feature:
    if feature!='YrSold':
        data=df.copy()
        # We will capture the difference between year variable and year the house was sold for
        df[feature]=df['YrSold']-data[feature]

        plt.scatter(data[feature],data['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalePrice')
        plt.show()
```



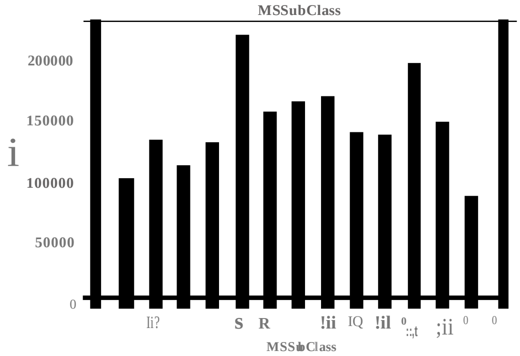


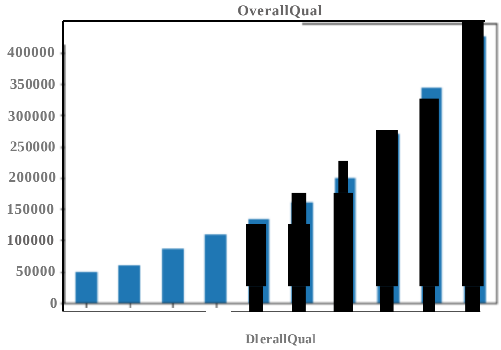


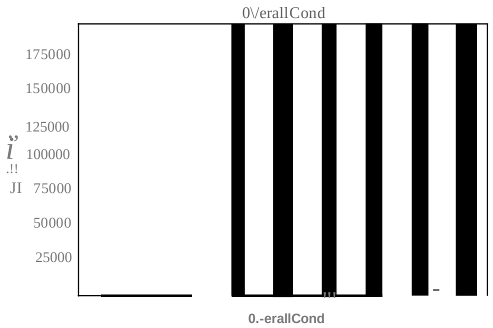
Here we can see that year built, The house which was built earlier has less price than therecent one.
Year of remodification, if the year is 60 the price is very less as compared to 0 to 10years.
Same thing happening with garage built year.

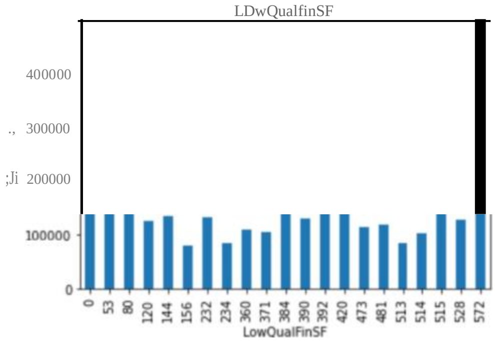
#Lets ind the relation between discrete feature and sales price

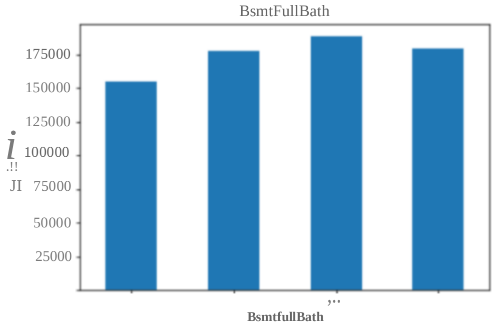
```
for feature in discrete_feature:data=df.copy()
    data.groupby(feature)['SalePrice'].median().plot.bar() plt.xlabel(feature)
    plt.ylabel('SalePrice')
    plt.title(feature) plt.show()
```

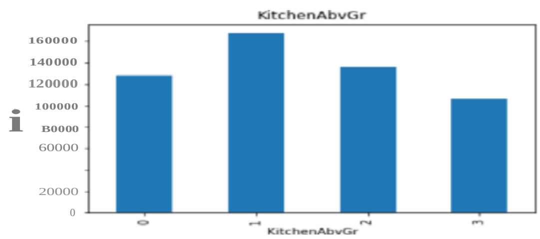


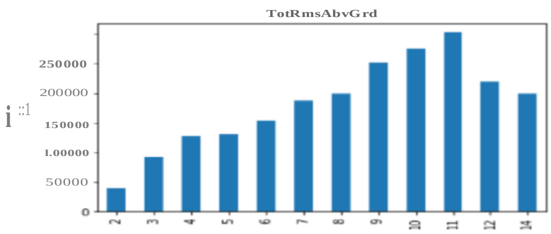


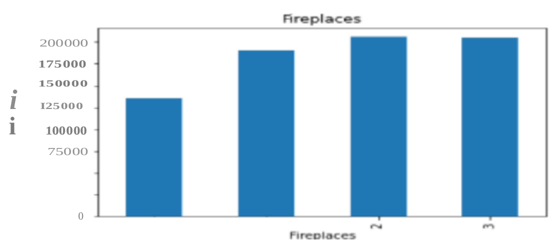

0VerallCond
175000
150000
125000
100000
75000
50000
25000
0.-erallCond


LDwQualfinSF
400000
300000
200000
100000
0
0 53 80 120 144 156 232 234 360 371 384 390 392 420 473 481 513 514 515 528 572
LowQualFinSF


BsmtFullBath
175000
150000
125000
100000
75000
50000
25000
BsmtfullBath


KitchenAbvGr
160000
140000
120000
100000
60000
20000
0
0 1 2 3
KitchenAbvGr


TotRmsAbvGrd
250000
200000
150000
50000
0
2 3 4 5 6 7 8 9 10 11 12 14

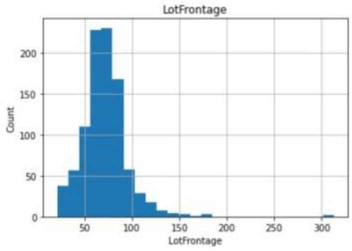

Fireplaces
200000
175000
150000
125000
100000
75000
0
2 3
Fireplaces

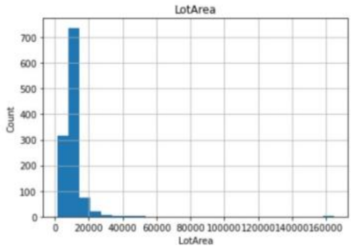
As we can see the overall quality is increasing the price is high.
If the overall condition of the house is average the price is more that others.As there is increase in no of fire places the price of house is rising.
As the house has a pool area of 555 sq feet the price of the house is more as comparedto others.

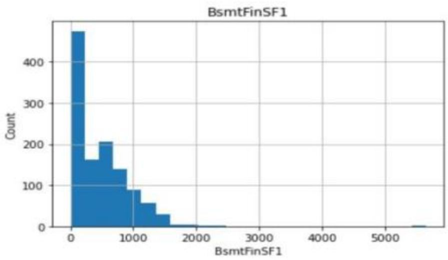
```
In [18]: continuous_feature=[feature for feature in numerical_features if feature not in discrete_feature+year_feature+['Id']]
         print("Continuous feauture count {}".format(len(continuous_feature)))
```

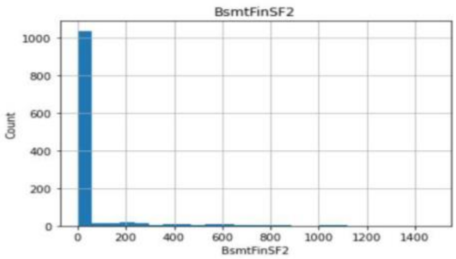
```
Continuous feauture count 16
```

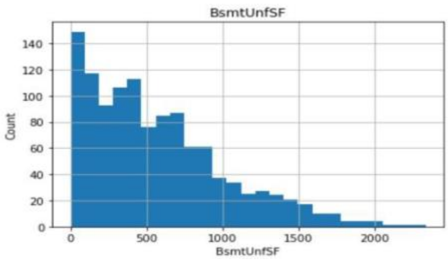
```
In [19]: #Lets analyse the continuous values by creating histogram to understand the distribution

         for feature in continuous_feature:
             data=df.copy()
             data[feature].hist(bins=25)
             plt.xlabel(feature)
             plt.ylabel("Count")
             plt.title(feature)
             plt.show()
```





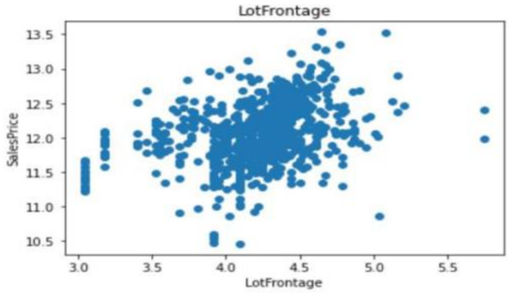


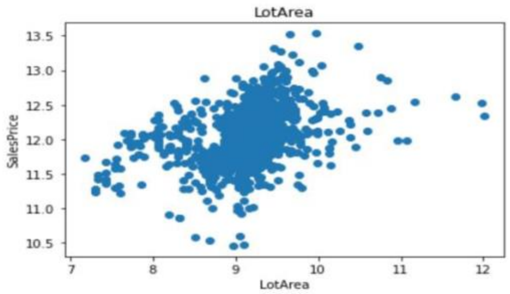


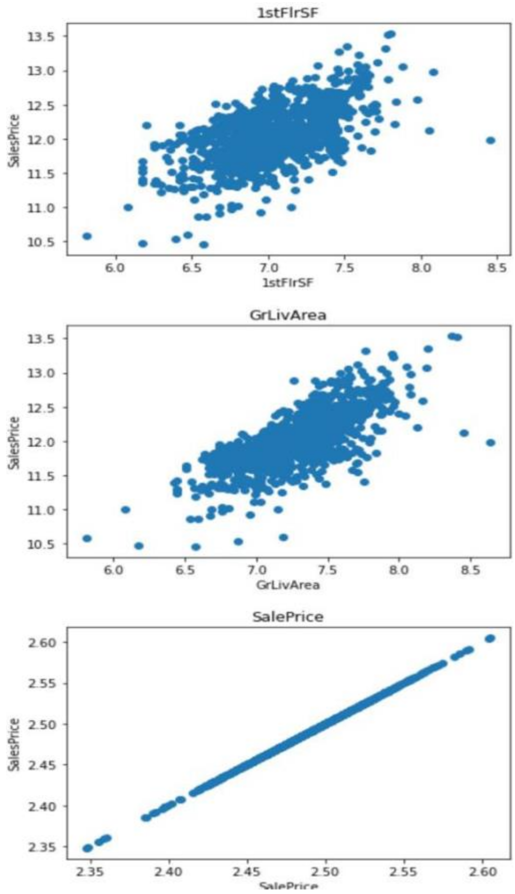


Here you can see that the data is skewed so we will perform log transformation toreduced skewness of the data.

```python
## We will be using logarithmic transformation


for feature in continuous_feature:
    data=df.copy()
    if 0 in data[feature].unique():
        pass
    else:
        data[feature]=np.log(data[feature])
        data['SalePrice']=np.log(data['SalePrice'])
        plt.scatter(data[feature],data['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalesPrice')
        plt.title(feature)
        plt.show()
```

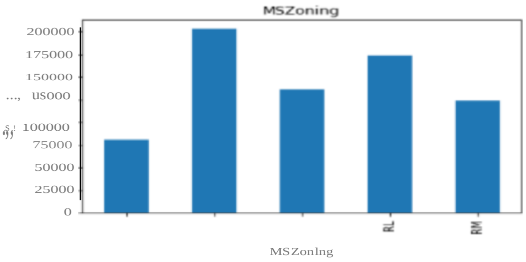


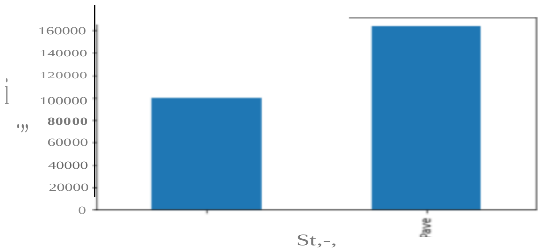


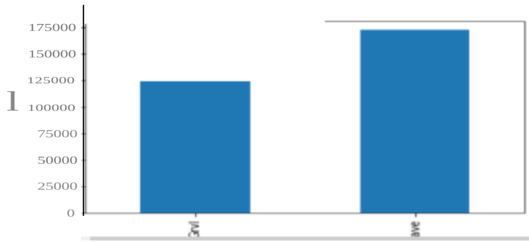


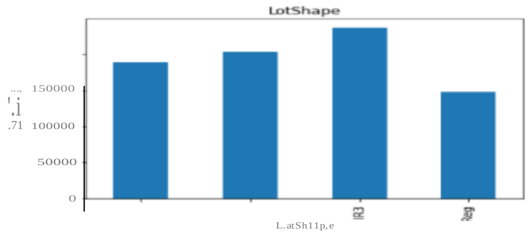
So after applying log transformation its giving monotonic relationship, as lotfrontage, lotarea, Grlivarea is increasing the price is increasing.

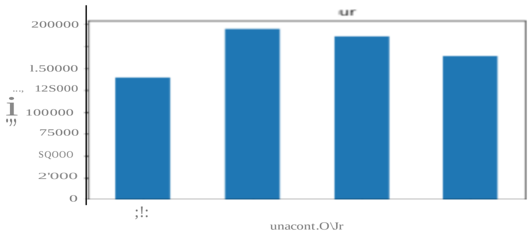
Lets’s see the relationship between categorical feature and dependent variable.

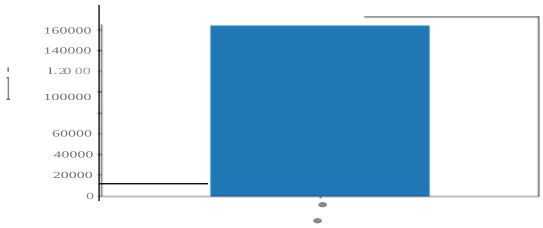

MSZoning
200000
175000
150000
100000
75000
50000
25000
0
RL
RM

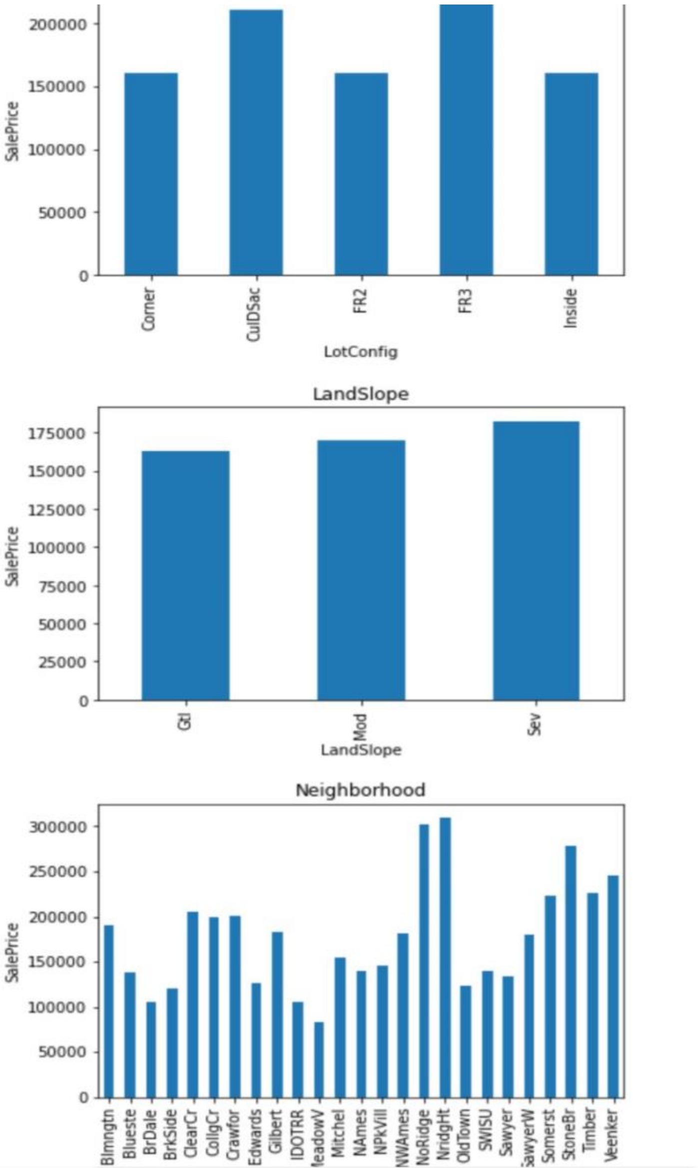

160000
140000
120000
100000
80000
60000
40000
20000
0
Pave


175000
150000
125000
100000
75000
50000
25000
0
Grvl


LotShape
150000
100000
50000
0
IR3
Reg

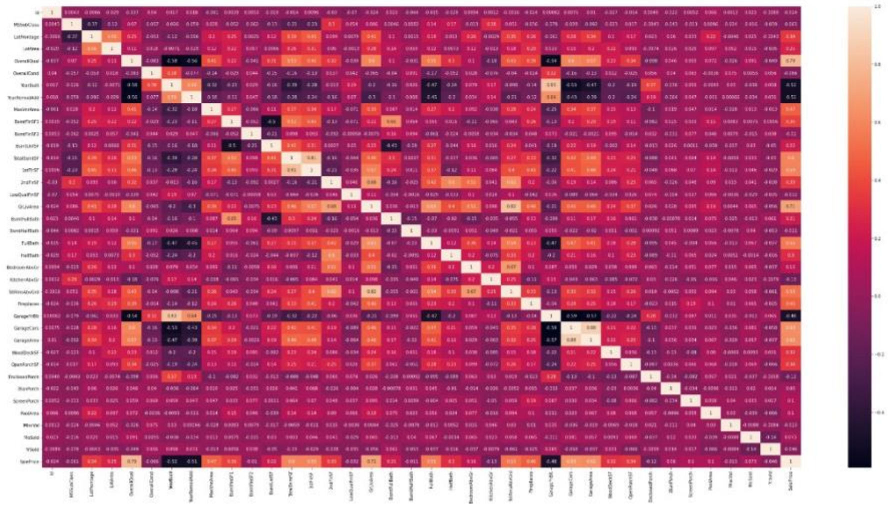

200000
75000
0


160000
140000
100000
60000
40000
20000
0



Let’s do some statistical analysis

```
plt.figure(figsize=(40,20))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In this we found that Variables like OverallQual (overall material and finish of the house) , Year Built , TotRmsAbvGrd (Total rooms above grade (does not include bathrooms) , GarageCars (Size of garage in car capacity) ,GarageArea (Size of garage in square feet) , GrLivArea (Above grade (ground) living area square feet) , FullBath (Full bathrooms abovegrade) have positive relationship with the sales Price.
YearBuilt, YearRemodAdd, GarageYrBuilt are negatively related with sale price.

- # Data Preprocessing Done

As above we have seen there is lot of missing data so we will deal with these missing values .

First of all we will drop columns Alley , MiscFeature , PoolQC , Fence and GarageYrBlt because more than 80 % data in these columns are missing if we replace these missing data with some data it can give us wrong prediction in the final model thus making our model less effective so better to drop these columns.

```
In [30]: df.drop(['Id','Alley','GarageYrBlt','PoolQC','Fence','MiscFeature'],axis=1,inplace=True)
```

Now in the other columns GarageType, GarageFinish , GarageQual , GarageCond , BsmtFinType2 , BsmtExposure, Bsmtclond which have missing data of 10-20 % we willreplace the missing data with the mode value .

```
## Filling Missing Values
df['LotFrontage']=df['LotFrontage'].fillna(df['LotFrontage'].mean())
df['BsmtCond']=df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtQual']=df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['FireplaceQu']=df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0])
df['GarageType']=df['GarageType'].fillna(df['GarageType'].mode()[0])
df['GarageFinish']=df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual']=df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond']=df['GarageCond'].fillna(df['GarageCond'].mode()[0])
```

```
df['MasVnrType']=df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])
df['MasVnrArea']=df['MasVnrArea'].fillna(df['MasVnrArea'].mode()[0])
```

```
df['BsmtExposure']=df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtFinType2']=df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0])
```

Same we will do with our test data set.

Now we will call our Finalised Test data set and append train and test data set .

Importing Finalised Test data

```
test_final=pd.read_csv('finaltest.csv')
test_final.head(5)
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	Bl
0	20	RL	86.000000	14157	Pave	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm	Norm	1F
1	120	RL	66.425101	5814	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm	Norm	Tv
2	20	RL	66.425101	11838	Pave	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1F
3	70	RL	75.000000	12000	Pave	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm	Norm	1F
4	60	RL	86.000000	14598	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr	Norm	1F

```
# Appending Both Dataset
df1=df.append(test_final)
df1.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	Bl
0	120	RL	70.98847	4928	Pave	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	Norm	Tv
1	20	RL	95.00000	15865	Pave	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	Norm	1F
2	60	RL	92.00000	9920	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	Norm	1F
3	20	RL	105.00000	11751	Pave	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	Norm	1F
4	20	RL	70.98847	16635	Pave	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	Norm	1F

Now we will use label encoder to convert categorical data into numbers

Using label Encoder

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['MSZoning','Street','LotShape','LandContour','Utilities','LotConfig','LandSlope','Neighborhood','Condition1','Condition2','BldgTyp
e',
       'HouseStyle','RoofStyle','RoofMatl','Exterior1st','Exterior2nd','MasVnrType','ExterQual','ExterCond','Foundation','BsmtQual',
       'BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','Heating','HeatingQC','CentralAir','Electrical','KitchenQual',
       'Functional','GarageType','GarageFinish','GarageCond','GarageQual','PavedDrive','SaleType','SaleCondition','FireplaceQu']
for val in list1:
    df1[val]=le.fit_transform(df1[val].astype(str))
```

- # Data Inputs – Logic - Output Relationships

Now we will divide the data into input and output, the output will be ‘SalePrice’ and all the other remaining columns will be input.

```
# Splitting Test and Train database.
Train=df1[0:1138]
Test=df1[1138:]
Test=Test.drop(['SalePrice'],axis=1)
```

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

```
x.tail()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
1162	30	3	45.00000	8212	1	3	3	0	4	0	7	2	2
1163	20	3	70.98847	9819	1	0	3	0	4	0	19	2	2
1165	160	3	24.00000	2280	1	3	3	0	2	0	13	2	2
1166	70	0	50.00000	8500	1	3	3	0	4	0	9	1	2
1167	60	3	70.98847	7861	1	0	3	0	4	0	8	2	2

```
y.tail()
```

```
1162     58500.0
1163    122000.0
1165    148500.0
1166     40000.0
1167    183200.0
Name: SalePrice, dtype: float64
```

- ## Hardware and Software Requirements and Tools Used

  Hardware: 8GB RAM, 64-bit, 7th gen i7 processor.
  Software: MS-Excel, Jupyter Notebook, python 3.6

Importing Necessary libraries

```python
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.linear_model import LinearRegression
```

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

  As we know that it is a Regression problem in which our output is 'Sale Price' so we will use the regression model like linear Regression , KNeighbors Regressor , Decision Tree Regressor,Gradient Boosting Regressor , Random Forest Regressor , Ada Boost Regressor etc we will train our training data using these algorithms and then we will test on the finalised test data set for final house price prediction . The algorithm which is giving better accuracy and Cross value score will be chosen as final model.

- ## Testing of Identified Approaches (Algorithms)

Here we will use here KNeighborsRegressor , Decision Tree Regressor , Gradient Boosting Regressor , Lasso , Random Forest Regressor ,Ada Boost Regressor And Linear Regressionalgorithms.

- # Run and Evaluate selected models

Now we will find best parameters of different algorithms.

```
def maxr2_score(regr,x,y):
    max_r_score=0
    for i in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state= i ,test_size=0.20)
        regr.fit(x,y)
        y_pred=regr.predict(x_test)
        r2_scr=r2_score(y_test,y_pred)
        if r2_scr>max_r_score:
            max_r_score=r2_scr
            final_i=i
    print(" We are getting maximum  r2 score corresponding to",final_i,"is",max_r_score)
    return final_i
```

```
lreg=LinearRegression()
i=maxr2_score(lreg,x,y)
```

```
 We are getting maximum  r2 score corresponding to 80 is 0.9016128009289508
```

```
print("Mean r2 score for Linear Regression:",cross_val_score(lreg,x,y,cv=5,scoring="r2").mean())
print("standard deviation in r2 score for Linear Regression",cross_val_score(lreg,x,y,cv=5,scoring="r2").std())
```

```
Mean r2 score for Linear Regression: 0.7313019398098832
standard deviation in r2 score for Linear Regression 0.13647787720419355
```

```
# using Grid search CV to find best  parameters of different algorithim .
#Best parameters for  KNeighborsRegressor
knn=KNeighborsRegressor()
parameters={"n_neighbors" :range(1,10) , 'algorithm' :['auto', 'ball_tree', 'kd_tree', 'brute']}
gd=GridSearchCV(knn,parameters)
gd.fit(x,y)
print(" Best parameters of KNeighborsRegressor is :-")
print(gd.best_params_)
print("\n")

#Best parameters for  DecisionTreeRegressor
dtr=DecisionTreeRegressor()
parameters={"criterion" :( 'mse', 'friedman_mse', 'mae') ,'max_features' : ['auto', 'sqrt', 'log2']}
gd=GridSearchCV(dtr,parameters)
gd.fit(x,y)
print(" Best parameters of DecisionTreeRegressor is :-")
print(gd.best_params_)
print("\n")
```

```
#Best parameters for  GradientBoostRegressor
gbr=GradientBoostingRegressor()
parameters={"learning_rate":[0.001,0.01,0.1,1],"n_estimators":[25,50,100,120,150]}
gd=GridSearchCV(gbr,parameters)
gd.fit(x,y)
print(" Best parameters of GradientBoostingRegressor is :-")
print(gd.best_params_)
print("\n")


#Best parameters for  LassoRegressor
lsreg=Lasso()
parameters={"alpha":[0.001,0.01,0.1,1] , 'selection' : ['cyclic', 'random']}
gd=GridSearchCV(lsreg,parameters)
gd.fit(x,y)
print(" Best parameters of Lasso is :-")
print(gd.best_params_)
print("\n")

#Best parameters for  RandomForestRegressor
rfr=RandomForestRegressor()
parameters={"n_estimators":[10,50,100,120,150],"max_features": ["auto", "sqrt", "log2"]}
gd=GridSearchCV(rfr,parameters,cv=5)
gd.fit(x,y)
print(" Best parameters of RandomForestRegressor is :-")
print(gd.best_params_)
print("\n")


#Best parameters for  AdaBoostRegressor
ada=AdaBoostRegressor()
parameters={"learning_rate":[0.001,0.01,0.1,1],"n_estimators":[25,50,100,150,200]}
gd=GridSearchCV(ada,parameters)
gd.fit(x,y)
print(" Best parameters of AdaBoostRegressor is :-")
print(gd.best_params_)
print("\n")
```

```
 Best parameters of KNeighborsRegressor is :-
{'algorithm': 'auto', 'n_neighbors': 8}


 Best parameters of DecisionTreeRegressor is :-
{'criterion': 'mae', 'max_features': 'auto'}


 Best parameters of GradientBoostingRegressor is :-
{'learning_rate': 0.1, 'n_estimators': 100}


 Best parameters of Lasso is :-
{'alpha': 1, 'selection': 'random'}


 Best parameters of RandomForestRegressor is :-
{'max_features': 'sqrt', 'n_estimators': 150}


 Best parameters of AdaBoostRegressor is :-
{'learning_rate': 1, 'n_estimators': 100}
```

Now we will find the r2 score , cross value score and standard deviation of different algorithm.

```python
# Finding best r2 score value for KNeighbors regressor
print("KNeighbors regressor ")

knn=KNeighborsRegressor(n_neighbors= 8 , algorithm = 'auto')
i=maxr2_score(knn,x,y)
print(" ")
print("Mean r2 score for KNeighbor  Regression:",cross_val_score(knn,x,y,cv=135,scoring="r2").mean())
print("standard deviation in r2 score for KNeighbor   Regression",cross_val_score(knn,x,y,cv=5,scoring="r2").std())

print("\n______________________________________________________________________________________________\n")


# Finding best r2 score value for DecisionTreeRegressor
print("DecisionTreeRegressor")

dtc=DecisionTreeRegressor(criterion = 'mae'  , max_features = 'auto')
i=maxr2_score(dtc,x,y)
print(" ")
from sklearn.model_selection import cross_val_score
print("Mean r2 score for DecisionTreeRegressor :",cross_val_score(dtc,x,y,cv=139,scoring="r2").mean())
print("standard deviation in r2 score for DecisionTreeRegressor",cross_val_score(dtc,x,y,cv=5,scoring="r2").std())

print("\n______________________________________________________________________________________________\n")

# Finding best r2 score value for  GradientBoostingRegressor
print("GradientBoostingRegressor")

gbr=GradientBoostingRegressor(n_estimators=100 , learning_rate = 0.1)
i=maxr2_score(gbr,x,y)
print(" ")
print("Mean r2 score for GradientBoostingRegressor :",cross_val_score(gbr,x,y,cv=10,scoring="r2").mean())
print("standard deviation in r2 score for GradientBoostingRegressor ",cross_val_score(gbr,x,y,cv=5,scoring="r2").std())

print("\n______________________________________________________________________________________________\n")


# Finding best r2 score value for Lasso Regressor
print("Lasso Regressor")

lsreg=Lasso(alpha=1 , selection = 'random')
i=maxr2_score(lsreg,x,y)
print(" ")
print("Mean r2 score for Lasso Regression:",cross_val_score(lsreg,x,y,cv=63,scoring="r2").mean())
print("standard deviation in r2 score for Lasso Regression",cross_val_score(lsreg,x,y,cv=5,scoring="r2").std())

print("\n______________________________________________________________________________________________\n")
```

```
# Finding best r2 score value for Random Forest Regressor
print("Random Forest Regressor Regressor")

rfr=RandomForestRegressor(max_features='sqrt' , n_estimators = 150)
i=maxr2_score(rfr,x,y)
print(" ")
print("Mean r2 score for Lasso Regression:",cross_val_score(lsreg,x,y,cv=63,scoring="r2").mean())
print("standard deviation in r2 score for Lasso Regression",cross_val_score(lsreg,x,y,cv=5,scoring="r2").std())

print("\n_______________________________________________________________________________\n")


# Finding best r2 score value for  AdaBoostRegressor
print("AdaBoostRegressor")

ada=AdaBoostRegressor(n_estimators=100 , learning_rate = 1)
i=maxr2_score(ada,x,y)
print(" ")
print("Mean r2 score for AdaBoostRegressor :",cross_val_score(ada,x,y,cv=10,scoring="r2").mean())
print("standard deviation in r2 score for AdaBoostRegressor ",cross_val_score(ada,x,y,cv=5,scoring="r2").std())

print("\n_______________________________________________________________________________\n")
```

```
KNeighbors regressor
 We are getting maximum  r2 score corresponding to 98 is 0.7977072611046131

Mean r2 score for KNeighbor  Regression: 0.4678005051763844
standard deviation in r2 score for KNeighbor   Regression 0.04745955998655803

_______________________________________________________________________________

DecisionTreeRegressor
 We are getting maximum  r2 score corresponding to 42 is 1.0

Mean r2 score for DecisionTreeRegressor : 0.5075463100945778
standard deviation in r2 score for DecisionTreeRegressor 0.06360428761298062

_______________________________________________________________________________

GradientBoostingRegressor
 We are getting maximum  r2 score corresponding to 92 is 0.9773907712750841

Mean r2 score for GradientBoostingRegressor : 0.8542658132398382
standard deviation in r2 score for GradientBoostingRegressor  0.05076818422769042

_______________________________________________________________________________

Lasso Regressor
 We are getting maximum  r2 score corresponding to 80 is 0.9016141005662217

Mean r2 score for Lasso Regression: 0.673807560782654
standard deviation in r2 score for Lasso Regression 0.13654099493783195

_______________________________________________________________________________

Random Forest Regressor Regressor
 We are getting maximum  r2 score corresponding to 98 is 0.9862371541789222

Mean r2 score for Lasso Regression: 0.6738075580766497
standard deviation in r2 score for Lasso Regression 0.13654097611944369

_______________________________________________________________________________

AdaBoostRegressor
 We are getting maximum  r2 score corresponding to 75 is 0.9036812323445376

Mean r2 score for AdaBoostRegressor : 0.7810159317254959
standard deviation in r2 score for AdaBoostRegressor  0.032062317187178044

_______________________________________________________________________________
```

- # Interpretation of the Results

From the above running different algorithms we found that Gradient Boosting Regressor is giving better value of r2 score , cross value score and standard deviation so we will choose it as final model.

Since GradientBoostRegressor is giving better result so we will use it as final model

```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=92,test_size=0.20)
gbr=GradientBoostingRegressor(n_estimators=120 , learning_rate = 0.1)
gbr.fit(x,y)
y_pred=gbr.predict(x_test)
```

```
print("RMSE is: ",np.sqrt(mean_squared_error(y_test,y_pred)))
print("r2_score is:",r2_score(y_test,y_pred))
```

```
RMSE is:  12151.69771576506
r2_score is: 0.980104735564237
```

Now we will predict the sale price for our Test data.

Using Model to predict House Price

```
House_Pricemodel.predict(Test)
House_Price
```

```
array([341646.18376134,   206498.04707676,   236896.8771191,7    172286.2298407 ,
       187304.98104127,    77057.9087113 5,   109493.37214665,    304701.82874211,
       225126.90195449,    141663.36017583,     69877.31793737,    125180.23598902,
       114338.14504302,    217566.39998374,    246207.24123438,    116443.4567067 1,
        96322.8113 3851,   104725.49190 475,   15857 6.06039254,   187593.46588868,
       143504.36936174,    137161.49792511 ,   133406.22470905,     62327.48528345,
        95634.99664255,     10 3860.079 77 346 ,   157448.94004833,  133187.88788251,
       167718.36610689,     90404.41558415,    122317.81734268,    158283.25321229,
       182281.89242771,    158137.28602735,     99159,79845355,    152349.48591721,
       164583.09616254,    112129.66443646,    148084.5447217 ,    141062.68380383,
        93429.51293165,    283996.232193 1 2,   182165.59684621 ,    16533 4.73392687 ,
       110188,20722877,    119622,67268147,    114643,26382662,     83543,61579328,
       175072.73216094 ,   322663.32512 64 2,   1308 50.6988 5726,   1939 72.58924 507 ,
        88128.7207099 1,    89876.59059596 ,   243586.87588166,    107148.96962193 ,
       139382.22726415,    175554,82003253,     91189,76663255,    220736.12525742,
        83038.76637962,    163506.052 58591 ,   11 3989.1066308 1 ,   1164 59.8540111 ,
       178845.15644976,     80646.312461 2 ,   130803.90683712,    184099.4270406 ,
       128030.20907327,    155560.61697377,    298 103.10300947,    156975.10668822,
       173536.51242443,    132991.93443243,    123686.1 2882566 ,   212 558.27688397 ,
       26111 2.62298887,   189900.0631927 ,    276491.24674818,    13 0510.96293 598,
       199826.87830886,    110362.25250198 ,   129629.7397419 ,    149849.74810881,
       165324.66368367,    249238.03041726,    105974.0 925 236 ,   376089.2631073 3,
       142095.89235587,    159975.72540027,     2240 31.57781227,   126856.03352284,
       110598.46166536,    114995.2475371 ,    180609.17152401 ,   135154.72922476,
       211981.27650266,    162403.05197429,    334634.6579042,9    112970.86210759,
       229947.68953069,     86405.43015889,    110171.81233784,    117962.40703756,
       183817.36568579,    114220.43429819,     2487 46.3984781 ,    125611.84187587,
       1 61848.62 4 7648 ,  1 84604.30 203958 ,  179064.8390931 ,   163121.299 91452,
       206937.29605309,    214410.00744522 ,    105228.47583083,      8007 4.84814358,
       115866.3933592 ,    167735.62546013,     125011.64520354,     89114,72645099,
        8690 7.9 56 7258 5,  171301.53956165,   233 798.61702 575 ,   11557 0.7346602 2,
       137305.42830289,    176788.740809 7 ,    115650.70278605 ,    16 4244.05181693,
        67712.7840190 1,    98159,20265875,    126772.25825074,    202098.3674583 ,
       130405.06404005     14 286 1.25394628 ,  16764 5.4967 4 39 5   28 1 60 .80174391J
       177937.95175063,    104832.87128653 ,    243630.95617511,    111577.45328079,
       119866.58636765,    398830.02370962,     74930.04172835,    362629,00466036,
       181263.2531687 J    1 91008 .9 1 291866 ,  151 217.7923070 1   106  503.42530134J
       101237.50620437,    1742 52.659095 18,    1 27935.75363619,   129866.76861472,
       178559.49737974,    104464.75666432,     94047,25215225,    175722,6805295 ,
```

```
       141663.36017583,    168998.26097465,    109357.27630559,    163099.60336736 ,
       1 28673.41849069 ,   946 4 3.6660208 ,    232898.9918220 3,   285572.213 31044 ,
       110689.49939292,    113848.61622341 ,    207509.53230837,    1:33833.1 760 30 84,
       127369.80600361,    13853861386682,      93841.46920425 ,   1513 71.014 18738 ,
       14234 0.38965283,   155227.07927 475 ,    86305.42893739 ,   224137.462386 56,
       119104.67026596 ,   114822.70596726,      97516.0680478 1 ,   160185.83576948,
       18972 4.12177649,   1838 27.7708 1988 ,   247 584.26121899,    110153.45840247 ,
       1812 15.02659886,   311460.71883722 ,    19744 3.5555 4791 ,    72545.164053 1 ,
       348669.1791 197 ,   155810.02589702,      91300.76001429,    140 947.11 9619 45,
        91 11 1.602038     1254 56.13770324 ,    10825 4.23294537 ,   1965 85.33164 175,
       157573.68727602 ,   148288.09979781,     237567.66295714,    151992.210526 1 2,
       20 269 4 - 1 94 3 4786 ,  118054 .364 24 55 ,  371027.4612122 ,   88064.049977 55,
       128479.703 14061,   2213 88.634 14971 ,   18230 4.30269108 ,   247509.28496707 ,
       12381,0 1 6176 262,  1 53374.345 1783 ,   168929.56681026,    190914.9319918 3,
       158159.372 3717 5,    86424.8700 4427 ,    957 56.2538 168 ,   17917 9.64927577
       11 27 8 1.11 89909 5,  370916.17002981 ,  168999.36072074,     98215.19952126 ,
       149748.68184 10 5,  172858.27091353,      86956.66292226 ,   136933.37839276,
       188593.8 525646 ,   147089.37218983 ,     9S839.607050 36 ,  157117.11876306 ,
       218322.42609748 ,   187964.69482819,     131945.58882144,    125842.7971477 ,
       315696.80313074,    259958.06945022 ,    269 129.2945 178 ,  181 838.96312638 J
       16 3 84 1 .00 232375 ,  117771.55746 334 ,  179490.01496699,   391101.715722 32,
       111915.291 9028 ,   303858.86880989,     133338.94742081 ,   269501.57885201,
       134466.338227S7,     10S539.2827 S825 ,   149896.08772733 ,    21792 4.684811 25,
       11 388 4.91018      143377.93214941 ,    147468.78278582,      92623.21850569])
```

Saving Prediction in csv file

```
In [62]:  df3=pd.DataFrame({'House_Price':House_Price})
          df3.head(10)
```

Out [62]:

	House_Price
0	341646.183761
1	206498.047077
2	236896.877119
3	172286.229841
4	187304.981041
5	77057.908711
6	109493.372147
7	304701.828742
8	225126.901954
9	[illegible]

CONCLUSION

- ## Key Findings and Conclusions of the Study

  In this we found that Variables like OverallQual (overall material and finish of the house) , Year Built , TotRmsAbvGrd (Total rooms above grade (does not include bathrooms) , GarageCars (Size of garage in car capacity) ,GarageArea (Size of garage in square feet) , GrLivArea (Above grade (ground) living area square feet) , FullBath (Full bathrooms abovegrade) have positive relationship with the sales Price and they effect the sales price hence these factors should be considered

- ## Learning Outcomes of the Study in respect of Data Science

  The goal is to achieve the system which will reduce the human effort to find a house having reasonable price. The proposed system. House Price Prediction model approximately try to achieve the same one. Proposed system focused on predictthe house price according to the area for that image processing and machine learning methods are used. The experimental results showed that this technique that are used while developing system will give accurate prediction of house price

- ## Limitations of this work and Scope for Future Work

  There are some more algorithms which can be used and checked if they are giving better results , Using Deep Learning for predicting house price may get some good results.