A

Data science

Project

on

“HOUSING PRICE PREDICTION”

Submitted by:

Santosh Arvind Dharam

ACKNOWLEDGMENT

I feel great pleasure to present the Project entitled “housing price prediction”. But it would be unfair on our part if I do not acknowledge efforts of some of the people without the support of whom, this Project would not have been a success. First and for most I am very much thankful to my respected SME ‘shrishti Maan’ for his leading guidance in this Project. Also he has been persistent source of inspiration to me. I would like to express my sincere thanks and appreciation to ‘flip robo’ for their valuable support. Most importantly I would like to express our sincere gratitude towards my Friend & Family for always being there when I needed them most.

Mr. Santosh Arvind Dharam

INTRODUCTION

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.

PROBLEM STATEMENT

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below. The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

Analytical Problem Framing

EDA steps:

1) import necessary libraries:

first we will import all the necessary libraries which will be usefull for analysis of data

```
In [1]: #import all libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
from sklearn.linear_model import LogisticRegression,Lasso,LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from scipy.stats import zscore
from sklearn.model_selection import cross_val_score
```

in this case we have to import all the necessary library that are usefull for data analysis in jupyter notebook

2)extract the dataset in jupyter notebook:

now lets we extract the data for analysis from excel file by pandas library

```
In [2]: data=pd.read_excel("C:\\Users\\SAI BABA\\Desktop\\train.xlsx")
data.head()
```

Out[2]:

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

5 rows × 81 columns

Data is extracted for further analysis in jupyter notebook

3)checking null values:

In this case we have to find out the null values present in our data set if yes it is required to remove it in our data set it has some null values it is also shown by heat

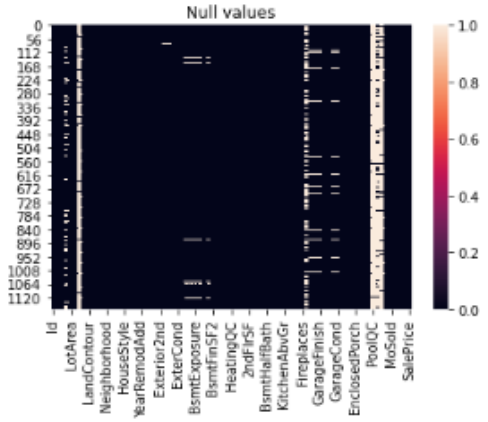
map

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     214
LotArea           0
                 ...
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64
```

it gives the null values present in the dataset ,dataset contains null values in it so we will remove it in the further steps

In [6]:
```
sns.heatmap(data.isnull())
plt.title("Null values")
plt.show()
```



now lets we will replace the null values present in the dataset with the mode

In [11]:
```
data['LotFrontage']=data['LotFrontage'].fillna(data['LotFrontage'].mode()[0])
data['MasVnrType']=data['MasVnrType'].fillna(data['MasVnrType'].mode()[0])
data['MasVnrArea']=data['MasVnrArea'].fillna(data['MasVnrArea'].mode()[0])
data['BsmtQual']=data['BsmtQual'].fillna(data['BsmtQual'].mode()[0])
data['BsmtCond']=data['BsmtCond'].fillna(data['BsmtCond'].mode()[0])
data['BsmtExposure']=data['BsmtExposure'].fillna(data['BsmtExposure'].mode()[0])
data['BsmtFinType1']=data['BsmtFinType1'].fillna(data['BsmtFinType1'].mode()[0])
data['BsmtFinType2']=data['BsmtFinType2'].fillna(data['BsmtFinType2'].mode()[0])
data['FireplaceQu']=data['FireplaceQu'].fillna(data['FireplaceQu'].mode()[0])
data['GarageType']=data['GarageType'].fillna(data['GarageType'].mode()[0])
data['GarageYrBlt']=data['GarageYrBlt'].fillna(data['GarageYrBlt'].mode()[0])
data['GarageFinish']=data['GarageFinish'].fillna(data['GarageFinish'].mode()[0])
data['GarageQual']=data['GarageQual'].fillna(data['GarageQual'].mode()[0])
data['GarageCond']=data['GarageCond'].fillna(data['GarageCond'].mode()[0])
```

In [12]:
```
data.isnull().sum().sum()
```

Out[12]:
```
0
```

4)Encoding the dataset:

In this case as our data contains some categorical column having object type of data it is necessary to convert it into numerical form by OrdinalEncoder

Encoding of dataframe

```
[14]: from sklearn.preprocessing import OrdinalEncoder

[15]: enc=OrdinalEncoder()

[16]: for i in data.columns:
          if data[i].dtype=="object":
              data[i]=enc.fit_transform(data[i].values.reshape(-1,1))
      data
```

[16]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	EnclosedPorch	3SsnPorch	ScreenPorch	Po
0	127	120	3.0	60.0	4928	1.0	0.0	3.0	0.0	4.0	...	0	0	0	
1	889	20	3.0	95.0	15865	1.0	0.0	3.0	0.0	4.0	...	0	0	224	
2	793	60	3.0	92.0	9920	1.0	0.0	3.0	0.0	1.0	...	0	0	0	
3	110	20	3.0	105.0	11751	1.0	0.0	3.0	0.0	4.0	...	0	0	0	
4	422	20	3.0	60.0	16635	1.0	0.0	3.0	0.0	2.0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	3.0	60.0	9819	1.0	0.0	3.0	0.0	4.0	...	0	0	0	
1164	554	20	3.0	67.0	8777	1.0	3.0	3.0	0.0	4.0	...	0	0	0	
1165	196	160	3.0	24.0	2280	1.0	3.0	3.0	0.0	2.0	...	0	0	0	
1166	31	70	0.0	50.0	8500	1.0	3.0	3.0	0.0	4.0	...	172	0	0	
1167	617	60	3.0	60.0	7861	1.0	0.0	3.0	0.0	4.0	...	0	0	0	

1168 rows × 77 columns

Data contains 77 columns and 1168 rows

5) Data Description:

In this case data is described in detail which helping us for detail analysis

```
In [18]: data.describe()
```

Out[18]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	EnclosedPorch	
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.0	1168.000000	...	1168.000000	1
mean	724.136130	56.767979	3.013699	68.975171	10484.749144	0.996575	1.938356	2.773973	0.0	3.004281	...	23.015411	
std	416.159877	41.940850	0.633120	22.836520	8957.442311	0.058445	1.412262	0.710027	0.0	1.642667	...	63.191089	
min	1.000000	20.000000	0.000000	21.000000	1300.000000	0.000000	0.000000	0.000000	0.0	0.000000	...	0.000000	
25%	360.500000	20.000000	3.000000	60.000000	7621.500000	1.000000	0.000000	3.000000	0.0	2.000000	...	0.000000	
50%	714.500000	50.000000	3.000000	64.000000	9522.500000	1.000000	3.000000	3.000000	0.0	4.000000	...	0.000000	
75%	1079.500000	70.000000	3.000000	79.250000	11515.500000	1.000000	3.000000	3.000000	0.0	4.000000	...	0.000000	
max	1460.000000	190.000000	4.000000	313.000000	164660.000000	1.000000	3.000000	3.000000	0.0	4.000000	...	552.000000	

8 rows × 77 columns

it gives the description about the dataset with its total count ,mean,std,and mini to max values distribution of the respective column

It is found from data description in some column min row consist of zero values in it so practically it is not

possible so it is necessary to remove it ,so we have removed it

now we can see from the dataset that some column contains zero values in it from mini to maximum so practically it is not possible such condition so lets w will replace that zeros with the mean of column

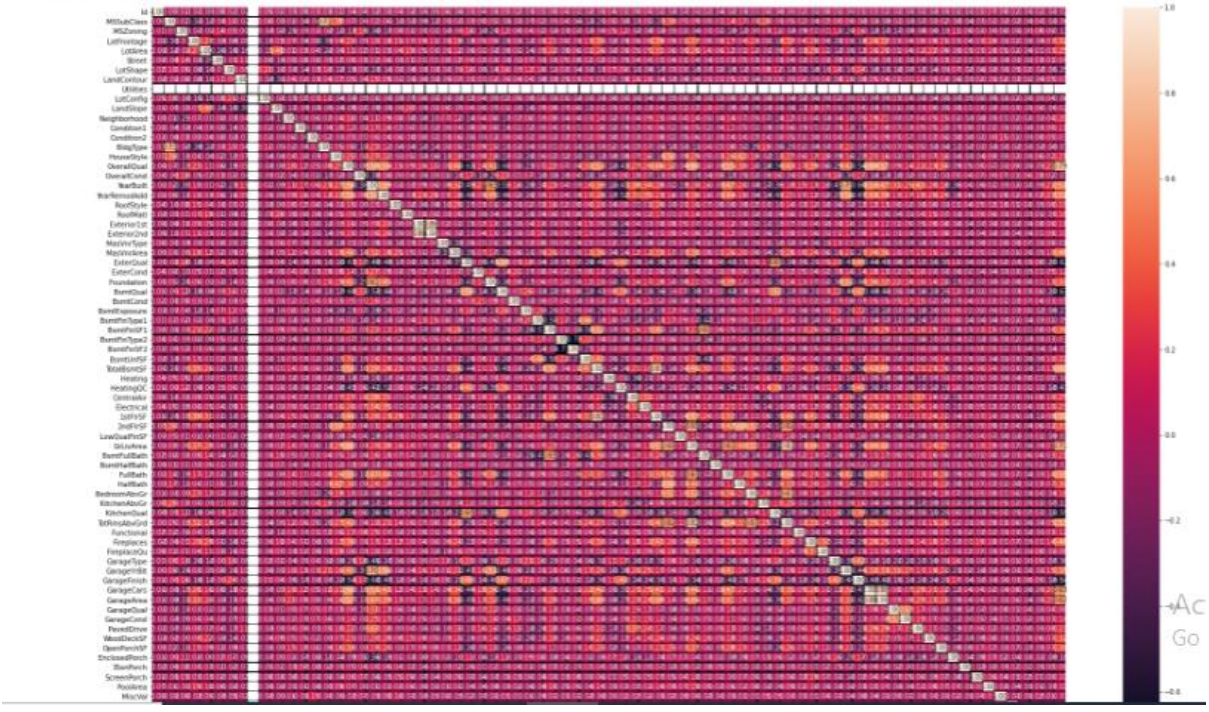
```
In [20]: #replacing zero values with mean of column
data['EnclosedPorch']=data['EnclosedPorch'].replace(0,data['EnclosedPorch'].mean())
data['3SsnPorch']=data['3SsnPorch'].replace(0,data['3SsnPorch'].mean())
data['ScreenPorch']=data['ScreenPorch'].replace(0,data['ScreenPorch'].mean())
data['PoolArea']=data['PoolArea'].replace(0,data['PoolArea'].mean())
data['MiscVal']=data['MiscVal'].replace(0,data['MiscVal'].mean())
```

6) Data Correlation:

Heat Map-

```
In [23]: #heat map
plt.figure(figsize=(30,20))
sns.heatmap(data.corr(),annot=True,linewidths=0.2,linecolor='black',fmt='0.2f')
```

```
Out[23]: <AxesSubplot:>
```

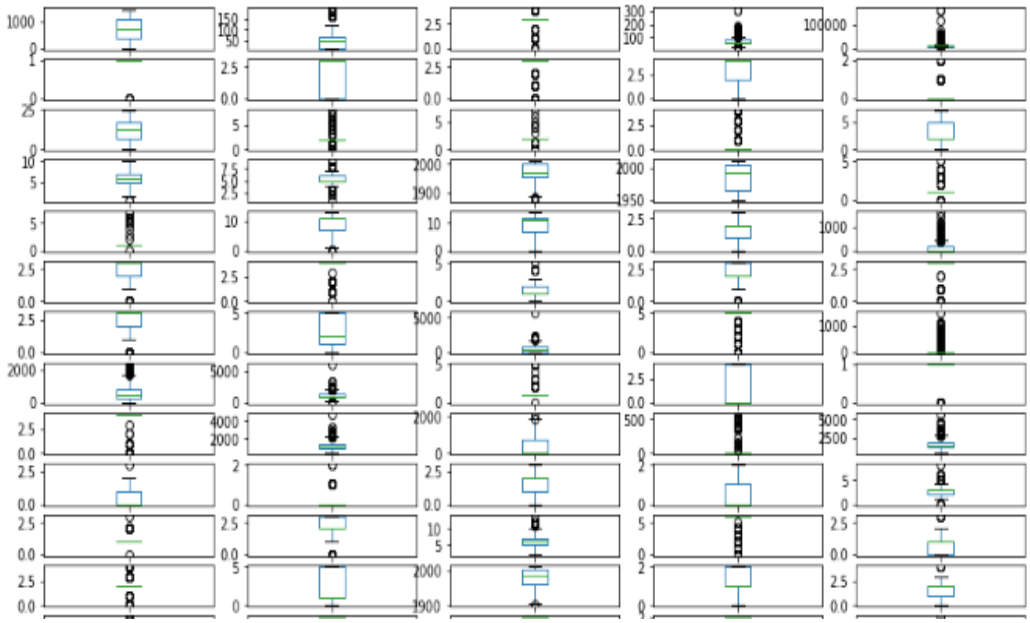


Data is correlated with other column data and also with its own it also gives the positive negative correlation of data with respective one another. it shows that some

column has maximum correlation with the sale price,also the column ‘Utilities’has no data present in it so lets we will drop that column. now lets we will find the outliers present in the dataset with the box plot

7)Checking of Outliers:

```
In [26]: #plotting the boxplot of each column to check the outliers
         data.plot(kind='box',subplots = True,layout=(16,5),figsize = (15,10))

Out[26]: Id                AxesSubplot(0.125,0.840263;0.133621x0.0397368)
         MSSubClass        AxesSubplot(0.285345,0.840263;0.133621x0.0397368)
         MSZoning          AxesSubplot(0.44569,0.840263;0.133621x0.0397368)
         LotFrontage       AxesSubplot(0.606034,0.840263;0.133621x0.0397368)
         LotArea           AxesSubplot(0.766379,0.840263;0.133621x0.0397368)
                                                  ...
         MoSold            AxesSubplot(0.285345,0.172684;0.133621x0.0397368)
         YrSold             AxesSubplot(0.44569,0.172684;0.133621x0.0397368)
         SaleType          AxesSubplot(0.606034,0.172684;0.133621x0.0397368)
         SaleCondition     AxesSubplot(0.766379,0.172684;0.133621x0.0397368)
         SalePrice               AxesSubplot(0.125,0.125;0.133621x0.0397368)
         Length: 76, dtype: object
```



data contains some outliers lets we will remove it.

8)Removing outliers:

data contains some amount of outliers present in it so lets we will remove it by taking the threshold as 3

In [27]:
```
#calculate the zscore
z = np.abs(zscore(data))
print(z)
```

```
[[1.43548658 1.50830058 0.02164599 ... 0.33003329 0.20793187 0.67631017]
 [0.39632483 0.87704243 0.02164599 ... 0.33003329 0.20793187 1.09423443]
 [0.16554544 0.07709478 0.02164599 ... 0.33003329 0.20793187 1.11687211]
 ...
 [1.26961389 2.46243779 0.02164599 ... 0.33003329 0.20793187 0.41705186]
 [1.66626597 0.31562908 4.76211672 ... 0.33003329 0.20793187 1.78922393]
 [0.25755011 0.07709478 0.02164599 ... 0.33003329 0.20793187 0.02179027]]
```

In [28]:
```
threshold=3
print(np.where(z<3))
print(data.shape)
```

```
(array([   0,    0,    0, ..., 1167, 1167, 1167], dtype=int64), array([ 0,  1,  2, ..., 73, 74, 75], dtype=int64))
(1168, 76)
```

In [29]:
```
#Assign the value to df_new which are less the threshold value and removing the outliers
data_new=data[(z<3).all(axis = 1)]
```

In [30]:
```
print(data.shape)
print(data_new.shape)
data = data_new
print('Shape after removing outlires',data.shape)
```

```
(1168, 76)
(483, 76)
Shape after removing outlires (483, 76)
```

So after removing the outliers we have 483 rows and 76 column remaining

9) Finding the skewness:

now lets check whether our data set contains skewness in it .if yes then lets we will remove it

lets we will check the skewness present in the dataset

In [32]:
```
data.skew()
```

Out[32]:
```
Id               -0.081971
MSSubClass        1.252783
MSZoning          2.328845
LotFrontage       0.353654
LotArea           0.197913
                    ...
MoSold            0.270695
YrSold            0.069565
SaleType         -3.196449
SaleCondition     1.591452
SalePrice         0.801238
Length: 76, dtype: float64
```

In [33]:
```
#remove skewness
data['MSSubClass']=np.sqrt(data['MSSubClass'])
data['MSZoning']=np.sqrt(data['MSZoning'])
data['SaleCondition']=np.sqrt(data['SaleCondition'])
```

skewness is removed with the help of sqrt

In [34]:
```
data.skew()
```

Out[34]:
```
Id               -0.081971
MSSubClass        0.695875
MSZoning          2.160294
LotFrontage       0.353654
LotArea           0.197913
                    ...
MoSold            0.270695
YrSold            0.069565
SaleType         -3.196449
SaleCondition     1.321703
SalePrice         0.801238
Length: 76, dtype: float64
```

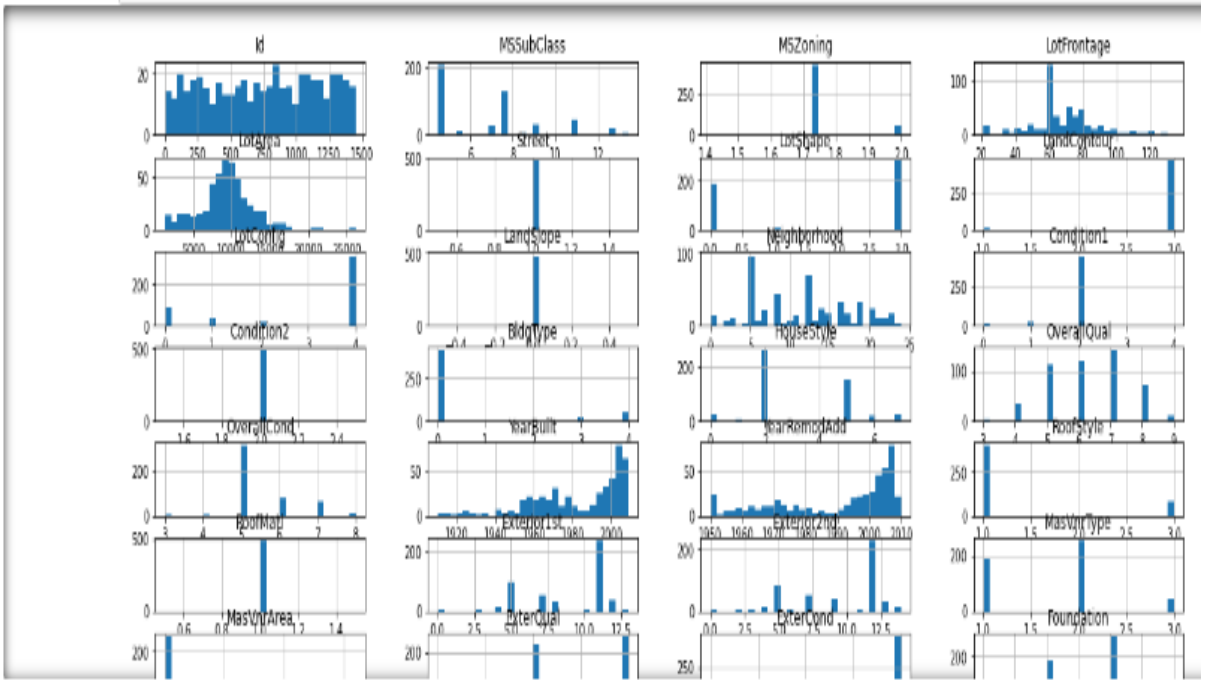
So column MSSubClass, MSZoning, SaleCondition has some skewness so we have removed it

10) Graphical Representation of Data:

Now let will check whether data is uniformely distributed or not

we can also plot the histogram to check the distribution of data

```
In [45]: #plotting histogram for univariate analysis and checking the Normal Distribution
         data.hist(figsize=(20,20), grid = True, layout = (19,4), bins = 30)
```



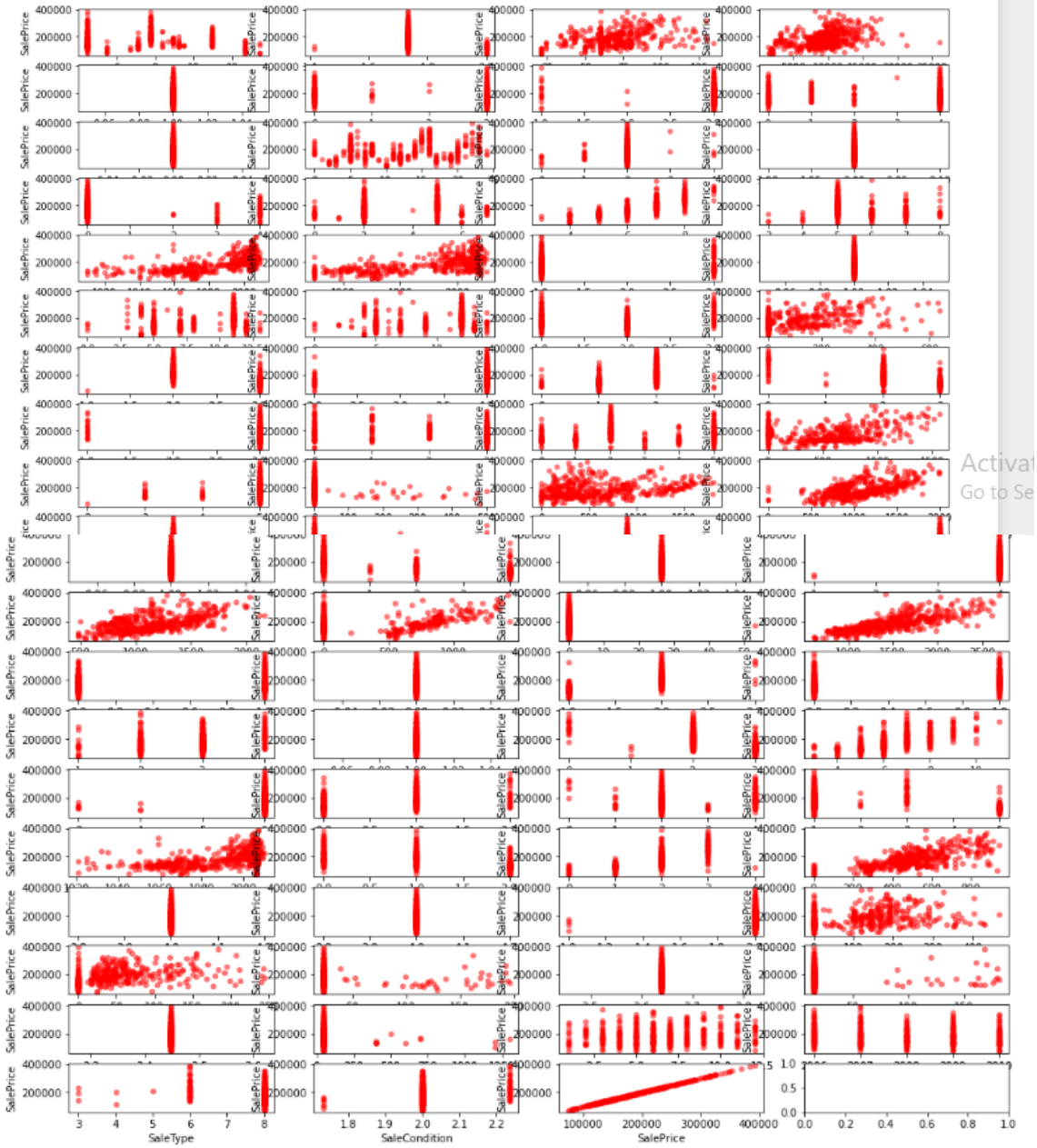
we can also plot scatter plot to show the relation of target column with the respective column

It shows that data is uniformely distributed

11) Scatter Plot:

scatter plot shows that how the data is distributed with respective the target variable

```
In [46]: # setup figure
fig, axes = plt.subplots(nrows=19, ncols=4, figsize=(16, 20))

# iterate and plot subplots
for xcol, ax in zip(data.columns[1:], [x for v in axes for x in v]):
    data.plot.scatter(x=xcol, y='SalePrice', ax=ax, alpha=0.5, color='r')
```



it shows the linear corelation with the target column

It shows the linear relationship between target variable and respective columns

12)Divide dataset:

In [48]:
```
#import the necessary libraries
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
from sklearn.linear_model import LogisticRegression,Lasso,LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from scipy.stats import zscore
from sklearn.model_selection import cross_val_score
```

In [49]:
```
#devide data set into feature and label
y=data['SalePrice']
x=data.drop(['SalePrice'],axis=1)
```

In [50]:
```
from sklearn.preprocessing import LabelEncoder,StandardScaler
```

In [51]:
```
#Standardize the value of x so that mean will 0 and SD will become 1 , and make the data as normal distributed
sc = StandardScaler()
sc.fit_transform(x)
x = pd.DataFrame(x,columns=x.columns)
```

Dataset is divided into x and y variable for further analysis

13)multiple Algorithms:

In [52]:
```
#Now by using multiple Algorithms we are calculating the best Algo which suit best for our data set

model = [DecisionTreeRegressor(),KNeighborsRegressor(),AdaBoostRegressor(),LinearRegression(),GradientBoostingRegressor()]
max_r2_score = 0
for r_state in range(40,90):
    train_x,test_x,train_y,test_y = train_test_split(x,y,random_state = r_state,test_size = 0.33)
    for i in model:
        i.fit(train_x,train_y)
        pre = i.predict(test_x)
        r2_sc = r2_score(test_y,pre)
        print("R2 score correspond to random state " ,r_state ,"is", r2_sc)
        if r2_sc> max_r2_score:
            max_r2_score=r2_sc
            final_state = r_state
            final_model = i

print()
print()
print()
print()
print("max R2 score correspond to random state " ,final_state , "is" , max_r2_score ,"and model is",final_model)
```

```
R2 score correspond to random state  40 is 0.7496481826113385
R2 score correspond to random state  40 is 0.6414011320892585
R2 score correspond to random state  40 is 0.8524242382218268
R2 score correspond to random state  40 is 0.9063119527142071
R2 score correspond to random state  40 is 0.8795479821501994
R2 score correspond to random state  41 is 0.7234707600632584
R2 score correspond to random state  41 is 0.5527187788147323
R2 score correspond to random state  41 is 0.7827957140957278
R2 score correspond to random state  41 is 0.8493767009615828
R2 score correspond to random state  41 is 0.8162555184351312
R2 score correspond to random state  42 is 0.5857637189209309
R2 score correspond to random state  42 is 0.6271222728615629
R2 score correspond to random state  42 is 0.8406354233787425
R2 score correspond to random state  42 is 0.8753967350555067
R2 score correspond to random state  42 is 0.8696373738219063
R2 score correspond to random state  43 is 0.6625534754567927
R2 score correspond to random state  43 is 0.5794201622062524
R2 score correspond to random state  43 is 0.8077898807173371
R2 score correspond to random state  43 is 0.8913825584580901
```

```
max R2 score correspond to random state  54 is 0.9316357221166225 and model is LinearRegression()
```

we got the max accuracy of 93.16% for random state of 54 for linearRegression() model

So by using the various alogorithm we found that we have maximum accuracy for LinearRegression()with accuracy of 93.16% for the random state of 54

14)Cross Validation:

Now lets we will do the cross validation of our model to confirm the accuracy of model

cross validation

lets we cross validate the model

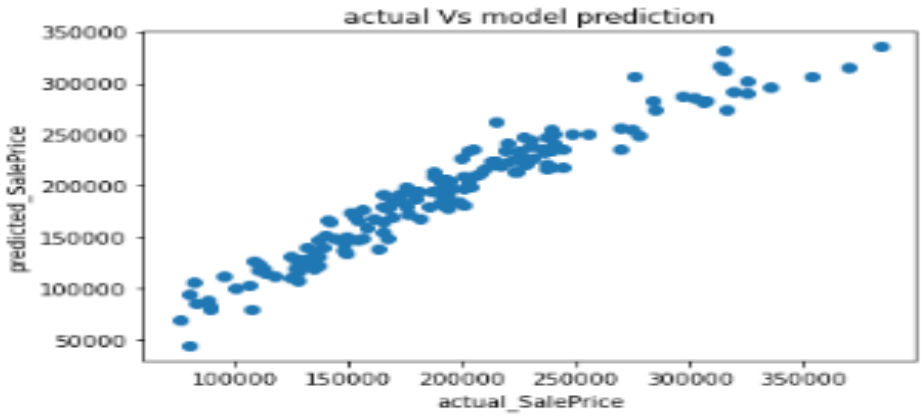
```
In [58]: from sklearn.model_selection import cross_val_score

In [59]: for i in range(2,10):
             cv=cross_val_score(LR,x,y,cv=i)
             print(LR,cv.mean())
```

```
LinearRegression() 0.8628243566669234
LinearRegression() 0.8692496763516032
LinearRegression() 0.8828764329914502
LinearRegression() 0.882516350569962
LinearRegression() 0.8827051185023134
LinearRegression() 0.8843870533554382
LinearRegression() 0.8888786586863904
LinearRegression() 0.8875926771510733
```

We have done cross validation of model in that case also we got it 88.88%

15)Visualization:

```
In [61]: plt.scatter(y_test,y_pred)
         plt.xlabel('actual_SalePrice')
         plt.ylabel('predicted_SalePrice')
         plt.title('actual Vs model prediction')
         plt.show()
```



it show that actual and predicted values are close to each other

It shows the actual VS model prediction result of model devoloped

16)Regularization:

Regularization

In [62]:
```
#import the necessary libraries
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
import warnings
from sklearn.linear_model import Lasso
warnings.filterwarnings('ignore')
```

In [63]:
```
parameters={'alpha':[.0001,0.001,.01,.1,1,10],'random_state':list(range(0,30))}
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'alpha': 10, 'random_state': 0}
```

In [64]:
```
from sklearn.metrics import r2_score
```

In [65]:
```
ls=Lasso(alpha=10,random_state=0)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
pred_ls=ls.predict(x_test)
lss=r2_score(y_test,pred_ls)
lss
```

Out[65]:
```
0.9315050179836585
```

In [66]:
```
cv_score=cross_val_score(ls,x,y,cv=5)
cv_mean=cv_score.mean()
cv_mean
```

Out[66]:
```
0.8831873103999553
```

We have done a regularization from that we got cv score as 88.31% and R2 score as 93.15%

17)Ensemble technique:

Ensemble technique

```
In [67]: from sklearn.model_selection import GridSearchCV

In [68]: from sklearn.ensemble import RandomForestRegressor

In [69]: parameters={'criterion':['mse','mae'],'max_features':['auto','sqrt','log2']}
         rf=RandomForestRegressor()
         clf=GridSearchCV(rf,parameters)
         clf.fit(x_train,y_train)
         print(clf.best_params_)

         {'criterion': 'mse', 'max_features': 'auto'}

In [70]: rf=RandomForestRegressor(criterion='mse',max_features='auto')
         rf.fit(x_train,y_train)
         rf.score(x_train,y_train)
         pred_decision=rf.predict(x_test)
         rfs=r2_score(y_test,pred_decision)
         print('R2score:',rfs*100)
         rfscore=cross_val_score(rf,x,y,cv=5)
         rfc=rfscore.mean()
         print('cross val score:',rfc*100)

         R2score: 87.35709533580102
         cross val score: 87.00056349488807
```

lets we will save best fit model

from this we got a R2score as 87.35% and cross validation score as 87%

18)Saving Model:

As we have tested out our selected model through various process lets we will save the given model

Saving Model

```
In [71]: #saving model
         import joblib
         joblib.dump(LR,'Housing_Project')

Out[71]: ['Housing_Project']
```

CONCLUSION

So we found the following conclusion from the saved model

conclusion

```
In [72]: loaded_model=joblib.load('Housing_Project')
         result=loaded_model.score(x_test,y_test)
         print(result)

         0.931635696708535
```

so in this way we have devoloped the model,saved it and also drawn a result from the devoloped model now lets we will draw the result from the given test data which is provided to us

We observed that data was filled with some outliers it is removed, also there some encoding is done now data is uniformly distributed in column. it is also observed from subplot, we have saved model and also predicted the result

with help of saved model .model is ready for the future data prediction. now lets we will draw the result from the given test data which is provided to us.

19)extract the test dataset in jupyter notebook:

```
In [73]: testdata=pd.read_excel("C:\\Users\\SAI BABA\\Desktop\\test.xlsx")
         testdata.head()
```

Out[73]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeatu
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	N
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	N
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	N
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	N
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	N

5 rows × 80 columns

So we have extracted testdata set in jupyter notebook it contains 80 columns and 292 rows

20)Load saves model and predict the result for test data:

```
In [84]: #Load the saved model
         SalePrice=joblib.load('Housing_Project')
         prices=SalePrice.predict(testdata)
         prices
```

```
Out[84]: array([ 2.40684140e+17,  1.54870399e+17,  2.55984028e+17,  1.04926247e+17,
                 1.30804757e+17, -4.20157284e+16,  9.72904146e+16,  2.05595197e+17,
                 1.82586559e+17,  1.48635200e+17,  7.63266945e+16,  5.81861266e+16,
                 1.43521601e+17,  1.92024808e+17,  2.82417992e+17,  2.29160551e+16,
                 1.02363243e+17,  6.46495873e+16,  8.92076214e+16,  8.42170754e+16,
                 8.83140721e+16,  1.36410009e+17,  1.55198937e+17,  2.55922172e+16,
                 1.01627684e+17,  1.10602767e+17,  1.39003866e+17,  1.67627328e+17,
                 9.15913984e+16,  1.50681175e+17,  8.64225897e+16,  1.47132958e+17,
                 2.66890948e+17,  1.70529148e+17,  1.13258465e+17,  8.20598791e+16,
                 1.11614285e+17,  8.54951591e+16,  8.88233867e+16,  1.15462672e+17,
                 6.76281503e+16,  1.84426461e+17,  1.23891958e+17,  1.64744765e+17,
                 1.20327434e+17,  1.20902919e+17,  1.39876340e+17,  1.04436504e+17,
                 1.63356737e+17,  2.08837221e+17,  2.18459296e+16,  1.80065379e+17,
                 1.09411726e+17,  1.13065133e+17,  1.85013735e+17,  9.51035370e+16,
                 1.60025925e+17,  1.16046818e+17, -7.07332753e+16,  2.35933597e+17,
                 5.82902950e+16,  1.22570108e+17,  1.93556160e+16,  1.03507538e+17,
                 1.75211966e+17,  9.42257894e+14,  1.42248677e+17,  1.95381084e+17,
                 1.08851822e+17,  1.23872003e+17,  2.52067721e+17,  5.74080470e+16,
                 1.35967124e+17,  1.48073373e+17,  1.37585056e+17,  1.32196596e+17,
                 1.89262520e+17,  1.82190460e+17,  1.98369261e+17,  1.39315170e+17,
                 1.03547291e+17,  7.41670754e+16,  2.43205345e+17,  1.51650246e+17,
                 1.14255138e+17,  2.58195014e+17,  1.15666439e+17,  2.72002417e+17,
                 1.04094694e+17,  1.39019017e+17,  2.44501459e+17,  8.68156500e+16,
                 1.02241654e+17,  1.33339052e+17,  1.36782987e+17,  1.49436815e+17,
                 2.01486836e+17,  8.45207284e+16,  1.79117504e+17,  1.14770321e+17,
                 1.86897516e+17,  9.69612985e+16,  1.35357565e+17,  8.07570023e+16,
                 1.47178957e+17,  1.21605584e+17,  1.73243914e+17,  2.41733331e+17,
                 1.38349227e+17,  1.40958800e+17,  1.73512739e+17,  1.15898009e+17,
                 1.60656453e+17,  1.67871397e+17,  1.17254837e+17,  1.01816625e+17,
                 1.31484450e+17,  1.16342022e+17,  1.22273715e+17,  6.85057713e+16,
                 7.63896180e+16,  1.98219835e+17,  1.46248940e+17,  1.30965667e+17,
```

So in this way we have developed LinearRegression() model, we saved it and also predicted the results.