SURPRISE HOUSING CASE STUDY

T. SANJIV

ABSTRACT

This case study revolves around Surprise Housing, a US-based real estate company, venturing into the Australian market. Leveraging data analytics, the company aims to strategically purchase houses below their market values and subsequently sell them at a profit. The core objective is to develop a robust predictive model for house prices based on relevant independent variables extracted from a dataset of Australian property sales. This model will empower Surprise Housing's management to comprehend the nuanced relationships between various factors and housing prices. By discerning these patterns, the company can refine its strategy, concentrating efforts on areas poised to yield optimal returns.

OBJECTIVE

The company wants to know which variables are significant in predicting the price of a house and how well those variables describe the price of a house. Hence it is necessary to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. The company can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for management to understand the pricing dynamics of a new market.

INTRODUCTION

Surprise Housing, a renowned US-based housing company known for its adept use of data analytics, has set its sights on the Australian real estate landscape. The company's proven strategy involves acquiring properties below market value and selling them at a premium. To execute this strategy successfully in a new market, Surprise Housing has gathered a comprehensive dataset from the Australian property market. This dataset comprises information on various independent variables related to house sales. The aim is to develop a predictive model that illuminates the intricate connections between these variables and housing prices. Such a model will serve as a valuable tool for Surprise Housing's management, guiding them in making data-driven decisions and tailoring their approach to maximize returns in the Australian market.

METHODOLOGY

- Creation of Virtual Environment
- Import Dataset/Read CSV
- Data Cleaning
- Exploratory Data Analysis
- Data Preparation
- Building ML Model
- Evaluate the Model

CODE

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

```
housing = pd.read_csv("train.csv")
housing.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Conditi
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	N
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	N
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	N
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	N
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	N

```
housing.shape
```

(1460, 81)

```
housing.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726	46.549315
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091	161.319273
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	0.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	0.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	0.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000	0.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
```

```
housing.isnull().sum()/housing.shape[0]
```

```
Id             0.000000
MSSubClass     0.000000
MSZoning       0.000000
LotFrontage    0.177397
LotArea        0.000000
Street         0.000000
Alley          0.937671
```

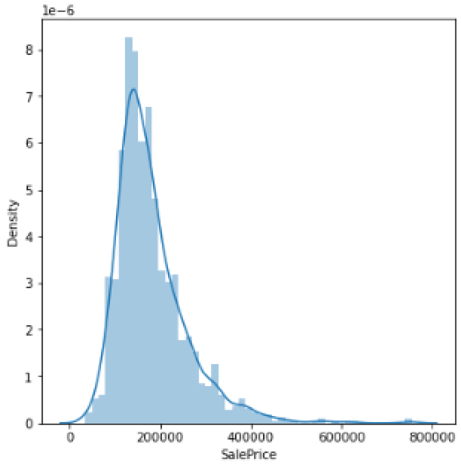
```
cols=['Alley','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','FireplaceQu','GarageType','GarageFinish','Garag
for i in cols:
    housing[i].fillna("None",inplace=True)
```

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          1460 non-null   object
```

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.figure(figsize=[6,6])
sns.distplot(housing['SalePrice'])
plt.show()
```

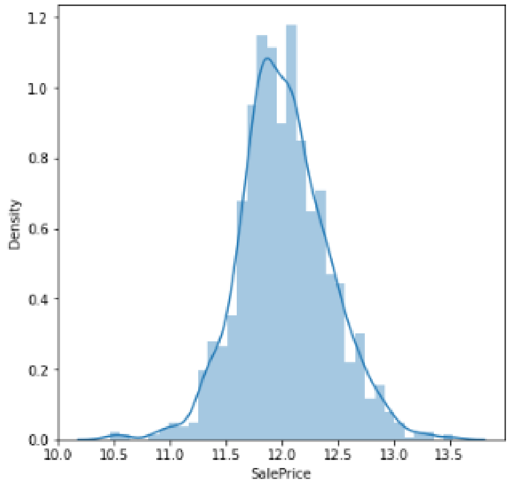



```
print("Skewness: ", housing['SalePrice'].skew())
print("Kurtosis: ", housing['SalePrice'].kurt())
```

```
Skewness:  1.8828757597682129
Kurtosis:  6.536281860064529
```

```
housing['SalePrice']=np.log(housing["SalePrice"])
```

```
plt.figure(figsize=[6,6])
sns.distplot(housing['SalePrice'])
plt.show()
```



```python
print("Skewness: ", housing['SalePrice'].skew())
print("Kurtosis: ", housing['SalePrice'].kurt())
```

```
Skewness:  0.12133506220520406
Kurtosis:  0.8095319958036296
```

```python
housing.drop("Id",axis=1,inplace=True)
```

```python
housing[['MSSubClass','OverallQual','OverallCond']]=housing[['MSSubClass','OverallQual','OverallCond']].astype('object')
```

```python
housing['LotFrontage'] = pd.to_numeric(housing['LotFrontage'],errors='coerce')
housing['MasVnrArea'] = pd.to_numeric(housing['MasVnrArea'],errors='coerce')
```

```python
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1460 non-null   object
 1   MSZoning       1460 non-null   object
 2   LotFrontage    1201 non-null   float64
 3   LotArea        1460 non-null   int64
 4   Street         1460 non-null   object
 5   Alley          1460 non-null   object
 6   LotShape       1460 non-null   object
```

```python
null_cols=housing.columns[housing.isnull().any()]
null_cols
```

```
Index(['LotFrontage', 'MasVnrType', 'MasVnrArea', 'Electrical', 'GarageYrBlt'], dtype='object')
```

```python
for i in null_cols:
    if housing[i].dtype==np.float64 or housing[i].dtype==np.int64:
        housing[i].fillna(housing[i].mean(),inplace=True)
    else:
        housing[i].fillna(housing[i].mode()[0],inplace=True)
```

```python
housing.isna().sum()
```

```
MSSubClass      0
MSZoning        0
LotFrontage     0
LotArea         0
Street          0
Alley           0
```

```python
cat_cols=housing.select_dtypes(include='object').columns
cat_cols
```

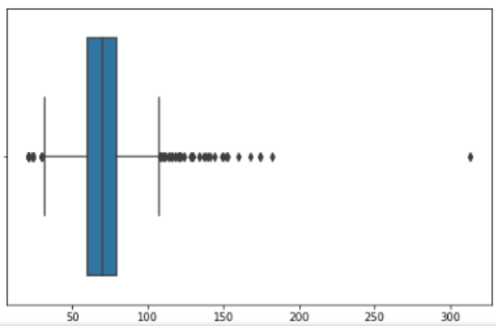
```
Index(['MSSubClass', 'MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour',
       'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1',
       'Condition2', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC',
       'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu',
       'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive',
       'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition'],
      dtype='object')
```

```python
num_cols=housing.select_dtypes(include=['int64','float64']).columns
num_cols
```

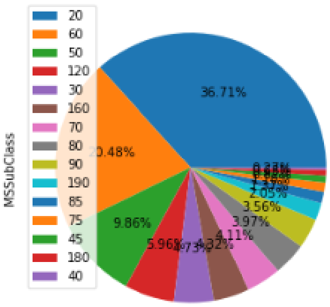
```
Index(['LotFrontage', 'LotArea', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea',
       'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF',
       '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
       'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

```python
for i in num_cols:
    plt.figure(figsize=[8,5])
    print (i)
    sns.boxplot(housing[i])
    plt.show()
```

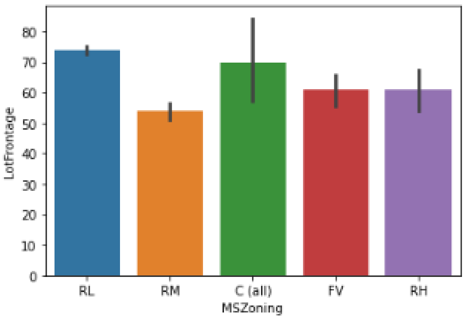
```
LotFrontage
```



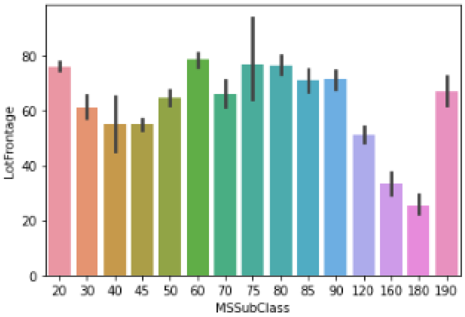
```
for i in cat_cols:
    print(housing[i].value_counts(normalize=True))
    plt.figure(figsize=[5,5])
    housing[i].value_counts(normalize=True).plot.pie(labeldistance=None,autopct='%1.2f%%')
    plt.legend()
    plt.show()
    print("----------------------------------------------------")
```




```
sns.barplot(x='MSZoning',y='LotFrontage',data=housing)
plt.show()
```

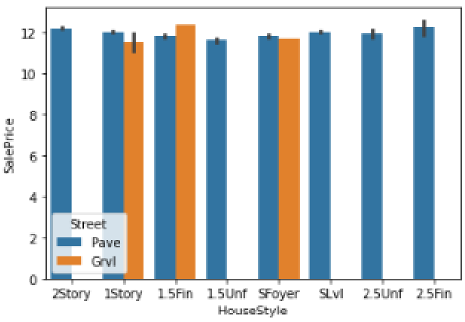



```
sns.barplot(x='MSSubClass',y='LotFrontage',data=housing)
plt.show()
```

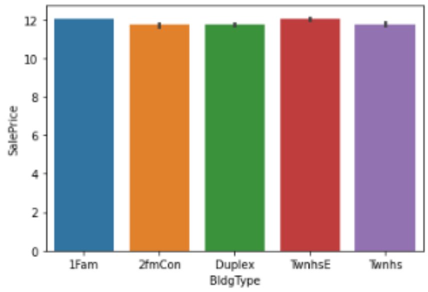



```
sns.barplot(x='HouseStyle',y='SalePrice',hue='Street',data=housing)
```

<AxesSubplot:xlabel='HouseStyle', ylabel='SalePrice'>



```
sns.barplot(x='BldgType',y='SalePrice',data=housing)
plt.show()
```




```
housing["Age"]=housing["YrSold"]-housing["YearBuilt"]
housing["Age"].head()
```

```
0     5
1    31
2     7
3    91
4     8
Name: Age, dtype: int64
```

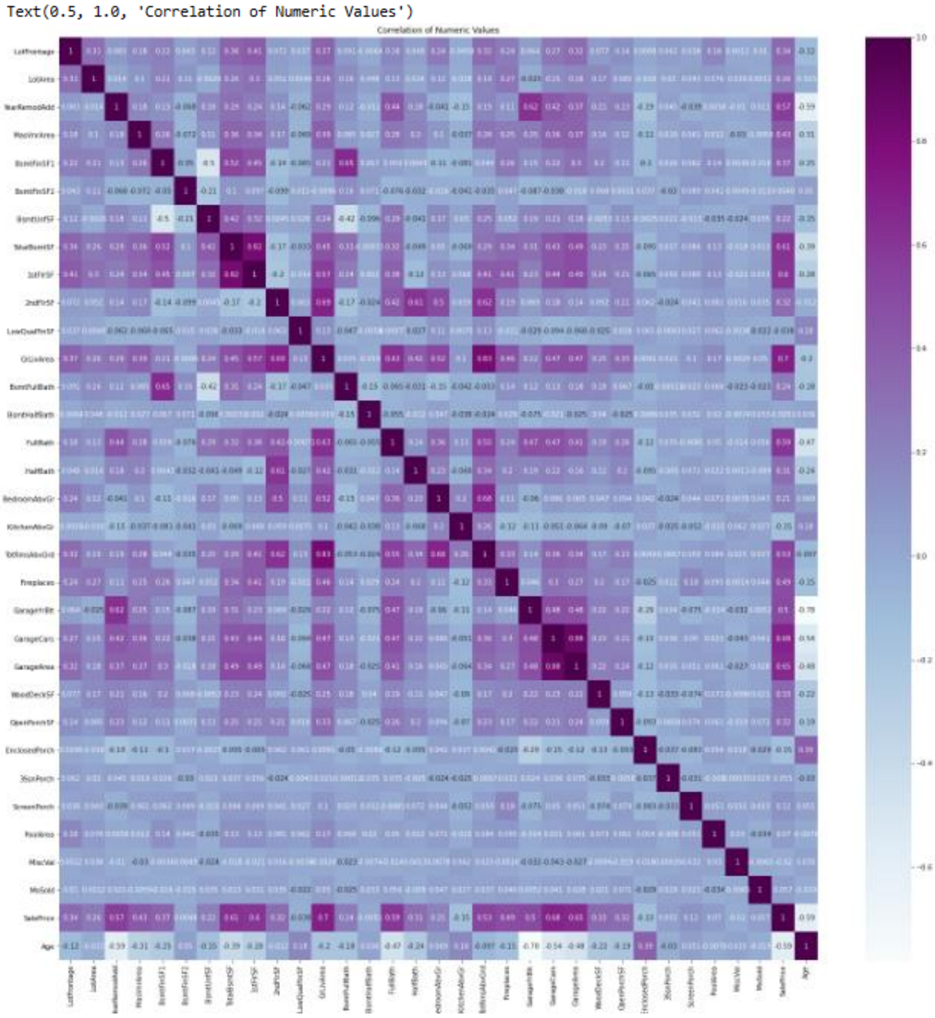
```
housing.drop(columns=["YearBuilt","YrSold"],axis=1,inplace=True)
```

```
housing.head()
```

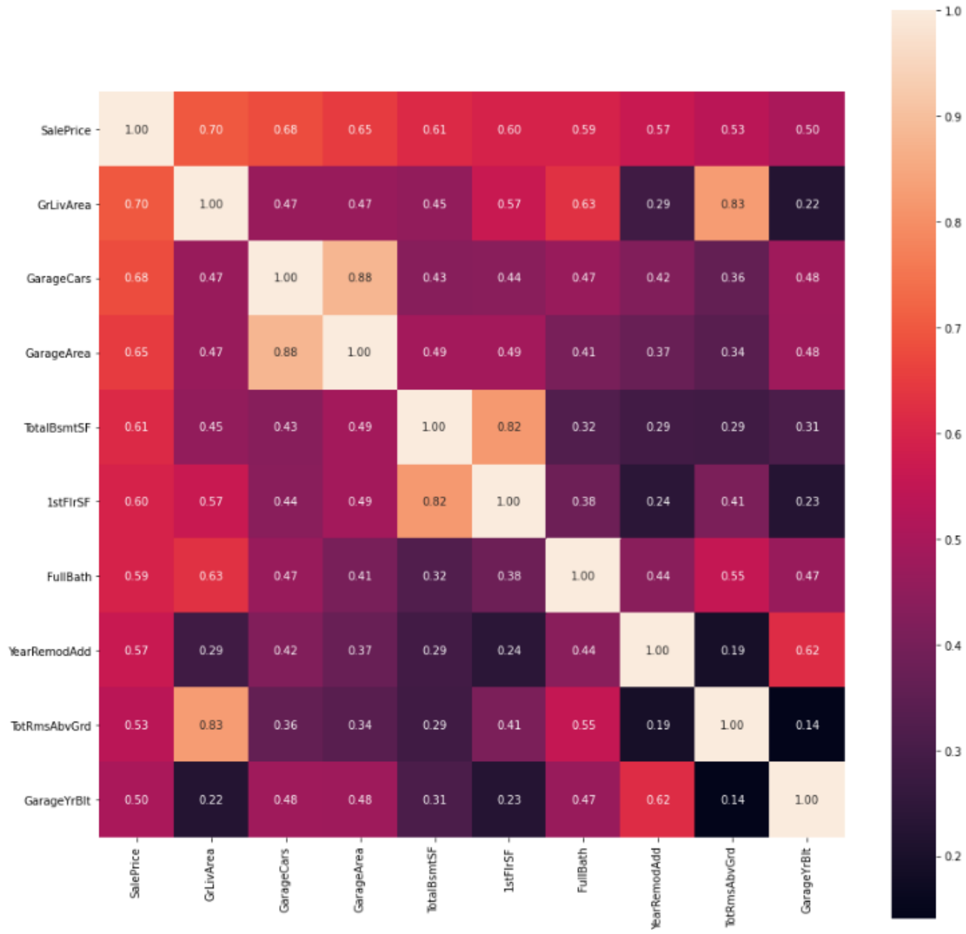
	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
0	60	RL	65.0	8450	Pave	None	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm
1	20	RL	80.0	9600	Pave	None	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm
2	60	RL	68.0	11250	Pave	None	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm
3	70	RL	60.0	9550	Pave	None	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm
4	60	RL	84.0	14260	Pave	None	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm

```
plt.figure(figsize=[25,25])
sns.heatmap(housing.corr(),annot=True,cmap='BuPu')
plt.title("Correlation of Numeric Values")
```

```
Text(0.5, 1.0, 'Correlation of Numeric Values')
```

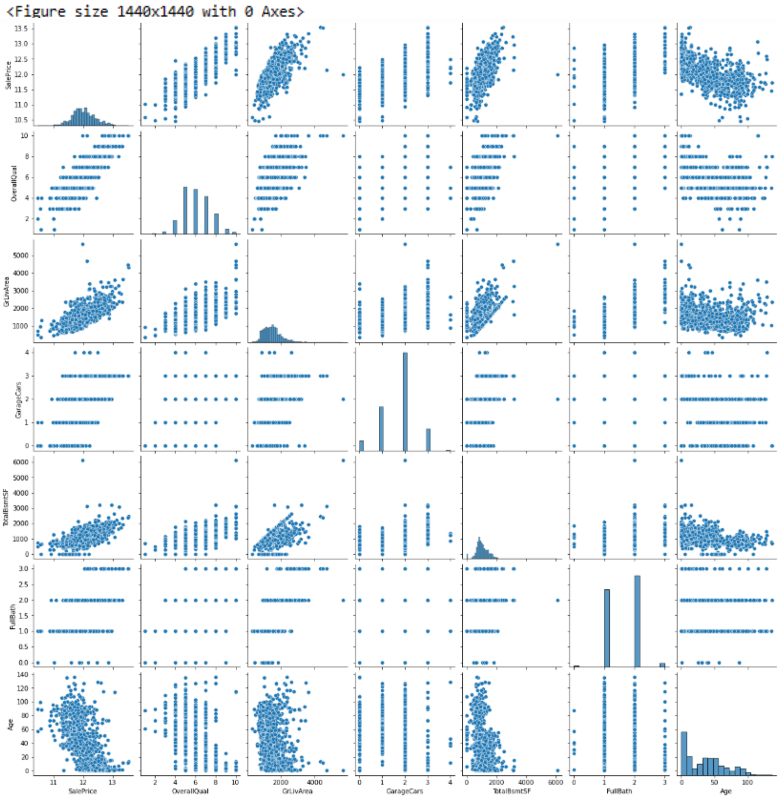


```
k=10
plt.figure(figsize=[15,15])
cols=housing.corr().nlargest(k,"SalePrice").index
cm=np.corrcoef(housing[cols].values.T)
sns.heatmap(cm,annot=True,square=True,fmt='.2f',cbar=True,annot_kws={'size':10},yticklabels=cols.values,xticklabels=cols.values)
plt.show()
```



```
cols=["SalePrice","OverallQual","GrLivArea","GarageCars","TotalBsmtSF","FullBath","Age"]
plt.figure(figsize=[20,20])
sns.pairplot(housing[cols])
plt.show()
```

```
<Figure size 1440x1440 with 0 Axes>
```



```
housing_num=housing.select_dtypes(include=['int64','float64'])
housing_cat=housing.select_dtypes(include='object')
```

```
housing_cat
```

	MSSubClass	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	Ho
0	60	RL	Pave	None	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	
1	20	RL	Pave	None	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	1Fam	
2	60	RL	Pave	None	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	
3	70	RL	Pave	None	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	1Fam	
4	60	RL	Pave	None	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	1Fam	
5	50	RL	Pave	None	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	Norm	Norm	1Fam	
6	20	RL	Pave	None	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm	Norm	1Fam	
7	60	RL	Pave	None	IR1	Lvl	AllPub	Corner	Gtl	NWAmes	PosN	Norm	1Fam	
8	50	RM	Pave	None	Reg	Lvl	AllPub	Inside	Gtl	OldTown	Artery	Norm	1Fam	
9	190	RL	Pave	None	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Artery	Artery	2fmCon	
10	20	RL	Pave	None	Reg	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	Norm	1Fam	

```
housing_cat_dm=pd.get_dummies(housing_cat, drop_first=True)
```

```
housing_cat_dm
```

	MSSubClass_30	MSSubClass_40	MSSubClass_45	MSSubClass_50	MSSubClass_60	MSSubClass_70	MSSubClass_75	MSSubClass_80	MSSubClass_8
0	0	0	0	0	1	0	0	0	
1	0	0	0	0	0	0	0	0	
2	0	0	0	0	1	0	0	0	
3	0	0	0	0	0	1	0	0	
4	0	0	0	0	1	0	0	0	
5	0	0	0	1	0	0	0	0	
6	0	0	0	0	0	0	0	0	
7	0	0	0	0	1	0	0	0	
8	0	0	0	1	0	0	0	0	
9	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	

```
house=pd.concat([housing_num,housing_cat_dm],axis=1)
```

```
house.head()
```

	LotFrontage	LotArea	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	Bsm
0	65.0	8450	2003	196.0	706	0	150	856	856	854	0	1710	
1	80.0	9600	1976	0.0	978	0	284	1262	1262	0	0	1262	
2	68.0	11250	2002	162.0	486	0	434	920	920	866	0	1786	
3	60.0	9550	1970	0.0	216	0	540	756	961	756	0	1717	
4	84.0	14260	2000	350.0	655	0	490	1145	1145	1053	0	2198	

```
house.shape
```

```
(1460, 287)
```

```
X=house.drop(["SalePrice"],axis=1).copy()
y=house["SalePrice"].copy()
```

```
X.head()
```

	LotFrontage	LotArea	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	Bsm
0	65.0	8450	2003	196.0	706	0	150	856	856	854	0	1710	
1	80.0	9600	1976	0.0	978	0	284	1262	1262	0	0	1262	
2	68.0	11250	2002	162.0	486	0	434	920	920	866	0	1786	
3	60.0	9550	1970	0.0	216	0	540	756	961	756	0	1717	
4	84.0	14260	2000	350.0	655	0	490	1145	1145	1053	0	2198	

```
y.head()
```

```
0    12.247694
1    12.109011
2    12.317167
3    11.849398
4    12.429216
Name: SalePrice, dtype: float64
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

```
X_train.shape
```

```
(1022, 286)
```

```
y_train.shape
```

```
(1022,)
```

```
num_cols=list(X_train.select_dtypes(include=['int64','float64']).columns)
```

```
scaler=StandardScaler()
X_train[num_cols]=scaler.fit_transform(X_train[num_cols])
X_test[num_cols]=scaler.fit_transform(X_test[num_cols])
```

```
def eval_metrics(y_train,y_train_pred,y_test,y_pred):
    print("r2 score (train) = ", '%.2f' % r2_score(y_train,y_train_pred))
    print("r2 score (test) = ", '%.2f' % r2_score(y_test,y_pred))
    mse_train=mean_squared_error(y_train,y_train_pred)
    mse_test=mean_squared_error(y_test,y_pred)
    rmse_train=mse_train**0.5
    rmse_test=mse_test**0.5
    print("RMSE(Train)=","%.2f" % rmse_train)
    print("RMSE(Test)=","%.2f" % rmse_test)
```

```
import sklearn
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import Ridge,Lasso
from sklearn.model_selection import GridSearchCV
```

```
params={'alpha':[0.0001,0.001,0.01,0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10,20,50,100,500,
ridge=Ridge()
ridgeCV=GridSearchCV(estimator=ridge,param_grid=params,scoring='neg_mean_absolute_error',cv=5,return_train_score=True,verbose=1,r
ridgeCV.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits

GridSearchCV(cv=5, estimator=Ridge(), n_jobs=-1,
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

```
ridgeCV.best_params_
```

```
{'alpha': 9.0}
```

```
ridgeCV.cv_results_
```

```
{'mean_fit_time': array([0.13720846, 0.0993504 , 0.03081727, 0.04070816, 0.04740314,
        0.06029181, 0.05292244, 0.03477216, 0.03919353, 0.0383626 ,
        0.05204473, 0.03514929, 0.04671869, 0.03985758, 0.03586249,
        0.04731069, 0.04268937, 0.0400022 , 0.04253049, 0.04914675,
        0.0487206 , 0.05676227, 0.06193485, 0.05470505, 0.04030724,
        0.04616952, 0.04762011, 0.04682202]),
 'std_fit_time': array([0.00774621, 0.05907468, 0.00444849, 0.00880401, 0.00684109,
        0.02986007, 0.02280127, 0.00516874, 0.00768354, 0.00484652,
        0.01263197, 0.00220556, 0.01303325, 0.00612041, 0.00369411,
        0.01105469, 0.00936135, 0.01039435, 0.00136688, 0.00757708,
```

```
ridge = Ridge(alpha=9)
```

```
ridge.fit(X_train,y_train)
```

```
Ridge(alpha=9)
```

```
ridge.coef_
```

```
array([-0.00905264,  0.01689835,  0.02655542,  0.00066475, -0.00509661,
        0.00889712,  0.00092087, -0.00131473,  0.02953909,  0.04557713,
        0.00741312,  0.0602059 ,  0.02296584,  0.0015807 ,  0.02398273,
        0.01812267,  0.00911103, -0.01985625,  0.02555875,  0.01728518,
       -0.00765676,  0.04437004,  0.0062415 ,  0.01369866, -0.00167914,
        0.01061907,  0.00830089,  0.01703403, -0.00490612, -0.00322371,
        0.00527018, -0.05580708, -0.07625653,  0.01124708,  0.00386694,
        0.0060703 , -0.02682263,  0.0454229 ,  0.02073923, -0.00465894,
        0.00671961, -0.00193414, -0.02212858, -0.0798776 , -0.02253906,
        0.002192  .  0.04569751.  0.01111378.  0.03548836. -0.01160636.
```

```
y_train_pred=ridge.predict(X_train)
y_pred=ridge.predict(X_test)
```

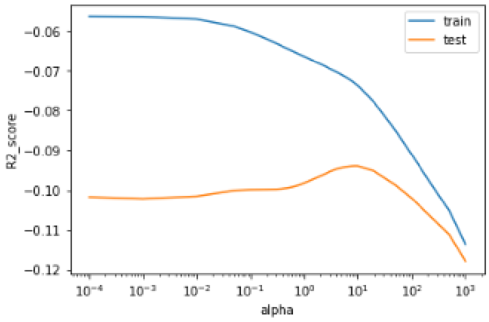
```
eval_metrics(y_train,y_train_pred,y_test,y_pred)
```

```
r2 score (train) =  0.92
r2 score (test) =  0.89
RMSE(Train)= 0.11
RMSE(Test)= 0.14
```

```
ridgeCV_res=pd.DataFrame(ridgeCV.cv_results_)
ridgeCV_res.head()
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score
0	0.137208	0.007746	0.009927	0.000490	0.0001	{'alpha': 0.0001}	-0.086974	-0.117866	-0.107960	-0.092888
1	0.099350	0.059075	0.009738	0.000669	0.001	{'alpha': 0.001}	-0.087063	-0.120552	-0.107401	-0.093355
2	0.030817	0.004448	0.013162	0.004504	0.01	{'alpha': 0.01}	-0.087974	-0.122446	-0.105153	-0.093149
3	0.040708	0.008804	0.010937	0.001777	0.05	{'alpha': 0.05}	-0.089697	-0.121157	-0.102321	-0.093026
4	0.047403	0.006841	0.014405	0.003546	0.1	{'alpha': 0.1}	-0.090679	-0.119532	-0.103001	-0.093605

```
plt.plot(ridgeCV_res['param_alpha'],ridgeCV_res['mean_train_score'],label='train')
plt.plot(ridgeCV_res['param_alpha'],ridgeCV_res['mean_test_score'],label='test')
plt.xlabel('alpha')
plt.ylabel('R2_score')
plt.xscale('log')
plt.legend()
plt.show()
```




```
lasso=Lasso()
lassoCV=GridSearchCV(estimator=lasso,param_grid=params,scoring='neg_mean_absolute_error',cv=5,return_train_score=True,verbose=1,n
lassoCV.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits

GridSearchCV(cv=5, estimator=Lasso(), n_jobs=-1,
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

```
lassoCV.best_params_
```

```
{'alpha': 0.0001}
```

```
lasso=Lasso(alpha=0.0001)
```

```
lasso.fit(X_train,y_train)
```

```
Lasso(alpha=0.0001)
```

```
lasso.coef_
```

```
array([ 3.25135868e-03,  1.61837652e-02,  2.36009828e-02,  2.99399618e-03,
        1.19110368e-02,  8.08934246e-03, -0.00000000e+00,  2.33193914e-02,
        3.00010118e-02,  4.34119465e-02,  9.03026183e-03,  5.87504892e-02,
        1.56629404e-02, -2.92098978e-04,  1.63617290e-02,  1.53623135e-02,
        6.59710651e-03, -1.94386846e-02,  1.66620750e-02,  1.50116077e-02,
       -3.71988381e-03,  2.03739586e-02,  2.10083133e-02,  1.20855898e-02,
        1.60792882e-04,  9.74643775e-03,  7.08932456e-03,  1.34753150e-02,
```

```
y_train_pred1=lasso.predict(X_train)
y_pred1=lasso.predict(X_test)
```

```
eval_metrics(y_train,y_train_pred1,y_test,y_pred1)
```

```
r2 score (train) =  0.94
r2 score (test) =  0.87
RMSE(Train)= 0.09
RMSE(Test)= 0.15
```

```
y_train_pred1=lasso.predict(X_train)
y_pred1=lasso.predict(X_test)
```

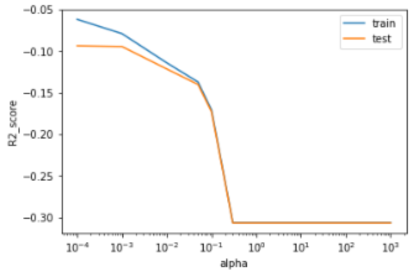
```
eval_metrics(y_train,y_train_pred1,y_test,y_pred1)
```

```
r2 score (train) =  0.94
r2 score (test) =  0.87
RMSE(Train)= 0.09
RMSE(Test)= 0.15
```

```
lassoCV_res=pd.DataFrame(lassoCV.cv_results_)
lassoCV_res.head()
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score
0	0.368023	0.126920	0.012352	0.006304	0.0001	{'alpha': 0.0001}	-0.082327	-0.111931	-0.095138	-0.084788
1	0.124638	0.036319	0.010886	0.001488	0.001	{'alpha': 0.001}	-0.089921	-0.113011	-0.094944	-0.086753
2	0.034319	0.005018	0.011862	0.003355	0.01	{'alpha': 0.01}	-0.112691	-0.139874	-0.122627	-0.115889
3	0.032445	0.005948	0.011265	0.002677	0.05	{'alpha': 0.05}	-0.133794	-0.152806	-0.157166	-0.136104
4	0.038272	0.013364	0.013267	0.007456	0.1	{'alpha': 0.1}	-0.166562	-0.180952	-0.200826	-0.165984

```
plt.plot(lassoCV_res['param_alpha'],lassoCV_res['mean_train_score'],label='train')
plt.plot(lassoCV_res['param_alpha'],lassoCV_res['mean_test_score'],label='test')
plt.xlabel('alpha')
plt.ylabel('R2_score')
plt.xscale('log')
plt.legend()
plt.show()
```




```
betas=pd.DataFrame(index=X.columns)
betas.rows=X.columns
betas['Ridge']=ridge.coef_
betas['Lasso']=lasso.coef_
betas
```

	Ridge	Lasso
LotFrontage	-0.009053	3.251359e-03
LotArea	0.016898	1.618377e-02
YearRemodAdd	0.026555	2.360098e-02
MasVnrArea	0.000665	2.993996e-03
BsmtFinSF1	-0.005097	1.191104e-02
BsmtFinSF2	0.008897	8.089342e-03
BsmtUnfSF	0.000921	-0.000000e+00
TotalBsmtSF	-0.001315	2.331939e-02
1stFlrSF	0.029539	3.000101e-02
2ndFlrSF	0.045577	4.341195e-02
LowQualFinSF	0.007413	9.030262e-03
GrLivArea	0.060206	5.875049e-02

```
lasso_cols_removed=list(betas[betas['Lasso']==0].index)
print(lasso_cols_removed)
```

```
['BsmtUnfSF', 'MSSubClass_40', 'MSSubClass_45', 'MSSubClass_50', 'MSSubClass_75', 'MSSubClass_85', 'MSSubClass_120', 'MSSubClas
s_180', 'MSSubClass_190', 'LotShape_IR3', 'Neighborhood_Blueste', 'Neighborhood_BrDale', 'Neighborhood_SWISU', 'Neighborhood_Sa
wyerW', 'Condition1_PosA', 'Condition1_RRNe', 'Condition2_Norm', 'Condition2_PosA', 'Condition2_RRAe', 'Condition2_RRAn', 'Cond
ition2_RRNn', 'BldgType_2fmCon', 'HouseStyle_2.5Fin', 'OverallQual_4', 'OverallCond_2', 'OverallCond_5', 'RoofStyle_Shed', 'Roo
fMatl_CompShg', 'RoofMatl_Membran', 'RoofMatl_Metal', 'RoofMatl_Roll', 'RoofMatl_Tar&Grv', 'Exterior1st_AsphShn', 'Exterior1st_
CBlock', 'Exterior1st_CemntBd', 'Exterior1st_ImStucc', 'Exterior1st_Stone', 'Exterior2nd_Brk Cmn', 'Exterior2nd_CBlock', 'Exter
ior2nd_ImStucc', 'Exterior2nd_Other', 'Exterior2nd_Stone', 'Exterior2nd_Wd Shng', 'ExterCond_Gd', 'ExterCond_Po', 'Foundation_S
tone', 'BsmtCond_Po', 'BsmtFinType1_Rec', 'BsmtFinType2_LwQ', 'BsmtFinType2_None', 'BsmtFinType2_Rec', 'HeatingQC_Po', 'Electri
cal_FuseF', 'Electrical_Mix', 'Functional_Min2', 'GarageType_BuiltIn', 'GarageType_CarPort', 'GarageFinish_Unf', 'GarageQual_P
o', 'GarageCond_Gd', 'GarageCond_TA', 'PoolQC_Fa', 'Fence_None', 'MiscFeature_Othr', 'MiscFeature_Shed', 'MiscFeature_TenC', 'S
aleType_ConLI', 'SaleType_ConLw']
```

```
lasso_cols_selected=list(betas[betas['Lasso']!=0].index)
print(lasso_cols_selected)
```

```
['LotFrontage', 'LotArea', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'Lo
wQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGr
d', 'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'Screen
Porch', 'PoolArea', 'MiscVal', 'MoSold', 'Age', 'MSSubClass_30', 'MSSubClass_60', 'MSSubClass_70', 'MSSubClass_80', 'MSSubClass
_90', 'MSSubClass_160', 'MSZoning_FV', 'MSZoning_RH', 'MSZoning_RL', 'MSZoning_RM', 'Street_Pave', 'Alley_None', 'Alley_Pave',
'LotShape_IR2', 'LotShape_Reg', 'LandContour_HLS', 'LandContour_Low', 'LandContour_Lvl', 'Utilities_NoSeWa', 'LotConfig_CulDSa
c', 'LotConfig_FR2', 'LotConfig_FR3', 'LotConfig_Inside', 'LandSlope_Mod', 'LandSlope_Sev', 'Neighborhood_BrkSide', 'Neighborho
od_ClearCr', 'Neighborhood_CollgCr', 'Neighborhood_Crawfor', 'Neighborhood_Edwards', 'Neighborhood_Gilbert', 'Neighborhood_IDOT
RR', 'Neighborhood_MeadowV', 'Neighborhood_Mitchel', 'Neighborhood_NAmes', 'Neighborhood_NPkVill', 'Neighborhood_NWAmes', 'Neig
hborhood_NoRidge', 'Neighborhood_NridgHt', 'Neighborhood_OldTown', 'Neighborhood_Sawyer', 'Neighborhood_Somerst', 'Neighborhood
_StoneBr', 'Neighborhood_Timber', 'Neighborhood_Veenker', 'Condition1_Feedr', 'Condition1_Norm', 'Condition1_PosN', 'Condition1
_RRAe', 'Condition1_RRAn', 'Condition1_RRNn', 'Condition2_Feedr', 'Condition2_PosN', 'BldgType_Duplex', 'BldgType_Twnhs', 'Bldg
```

```
print(len(lasso_cols_removed))
print(len(lasso_cols_selected))
```

```
68
218
```

```
betas['Ridge'].sort_values(ascending=False)[:10]
```

```
OverallQual_9           0.124933
Neighborhood_StoneBr    0.093923
OverallQual_8           0.084831
Neighborhood_Crawfor    0.083821
Exterior1st_BrkFace     0.083030
Neighborhood_NridgHt    0.081228
LandContour_HLS         0.071733
CentralAir_Y            0.070116
OverallCond_9           0.069989
BsmtCond_TA             0.066543
Name: Ridge, dtype: float64
```

```
ridge_coeffs=np.exp(betas['Ridge'])
ridge_coeffs.sort_values(ascending=False)[:10]
```

```
OverallQual_9           1.133073
Neighborhood_StoneBr    1.098476
OverallQual_8           1.088533
Neighborhood_Crawfor    1.087435
Exterior1st_BrkFace     1.086575
Neighborhood_NridgHt    1.084618
LandContour_HLS         1.074368
CentralAir_Y            1.072632
OverallCond_9           1.072496
BsmtCond_TA             1.068807
Name: Ridge, dtype: float64
```

```
betas['Lasso'].sort_values(ascending=False)[:10]
```

```
PoolQC_None             3.406185
PoolArea                0.255007
OverallQual_9           0.213745
OverallQual_10          0.195942
SaleCondition_Alloca    0.171168
SaleType_Oth            0.164500
GarageCond_Po           0.163084
MSZoning_FV             0.156478
OverallQual_8           0.146838
OverallCond_9           0.135652
Name: Lasso, dtype: float64
```

```
lasso_coeffs=np.exp(betas['Lasso'])
lasso_coeffs.sort_values(ascending=False)[:10]
```

```
PoolQC_None             30.149999
PoolArea                 1.290471
OverallQual_9            1.238307
OverallQual_10           1.216456
SaleCondition_Alloca     1.186690
SaleType_Oth             1.178804
GarageCond_Po            1.177136
MSZoning_FV              1.169385
OverallQual_8            1.158166
OverallCond_9            1.145283
Name: Lasso, dtype: float64
```

CONCLUSION

Based on the above analysis, we can observe the top 10 features with corresponding coefficients according to Ridge and Lasso models. We can also infer that the price of the house will increase by 1.11 with increase in GrLivArea, 1.08 times if the finish of the house is very good or if the house has centralized AC, 1.06 times if the basement condition is typical. The price might also increase if the neighborhood has Crawford, Stone Brook and Northridge Heights as physical locations within Ames city limits. The optimal value of lambda for Ridge Regression is 9 and for Lasso, it is 0.001