HousingPrice Prediction Using Lasso & Ridge Regression

November 30, 2022

1 Australia Housing Price Prediction

1.1 Business Goal:

- You are required 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. They 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.

1.1.1 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.

The Steps we will follow in this assignment as follow: 1. Reading, understanding and visualising the data 2. Preparing the data for modelling(train-test split etc) 3. Model building and evaluation - Lasso and Ridge Regression - Ridge Regression - Lasso Regression 5. Observation

1.1.2 Importing Packages

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import PolynomialFeatures,MinMaxScaler
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.feature_selection import RFE
from sklearn import metrics
from sklearn.model_selection import GridSearchCV

import statsmodels
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```python
import warnings
# ignore annoying warning if any
warnings.filterwarnings('ignore')
```

[2]:
```python
# Load the data from dataset
housing = pd.read_csv('train.csv')
housing.head()
```

[2]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave   NaN      Reg
1   2          20       RL         80.0     9600   Pave   NaN      Reg
2   3          60       RL         68.0    11250   Pave   NaN      IR1
3   4          70       RL         60.0     9550   Pave   NaN      IR1
4   5          60       RL         84.0    14260   Pave   NaN      IR1

  LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal MoSold  \
0         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
1         Lvl    AllPub  …        0    NaN   NaN         NaN       0      5
2         Lvl    AllPub  …        0    NaN   NaN         NaN       0      9
3         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
4         Lvl    AllPub  …        0    NaN   NaN         NaN       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]
```

[3]:
```python
housing.shape
```

[3]:
```
(1460, 81)
```

[4]:
```python
# Check for column details
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
7   LotShape       1460 non-null   object
8   LandContour    1460 non-null   object
9   Utilities      1460 non-null   object
10  LotConfig      1460 non-null   object
11  LandSlope      1460 non-null   object
12  Neighborhood   1460 non-null   object
13  Condition1     1460 non-null   object
14  Condition2     1460 non-null   object
15  BldgType       1460 non-null   object
16  HouseStyle     1460 non-null   object
17  OverallQual    1460 non-null   int64
18  OverallCond    1460 non-null   int64
19  YearBuilt      1460 non-null   int64
20  YearRemodAdd   1460 non-null   int64
21  RoofStyle      1460 non-null   object
22  RoofMatl       1460 non-null   object
23  Exterior1st    1460 non-null   object
24  Exterior2nd    1460 non-null   object
25  MasVnrType     1452 non-null   object
26  MasVnrArea     1452 non-null   float64
27  ExterQual      1460 non-null   object
28  ExterCond      1460 non-null   object
29  Foundation     1460 non-null   object
30  BsmtQual       1423 non-null   object
31  BsmtCond       1423 non-null   object
32  BsmtExposure   1422 non-null   object
33  BsmtFinType1   1423 non-null   object
34  BsmtFinSF1     1460 non-null   int64
35  BsmtFinType2   1422 non-null   object
36  BsmtFinSF2     1460 non-null   int64
37  BsmtUnfSF      1460 non-null   int64
38  TotalBsmtSF    1460 non-null   int64
39  Heating        1460 non-null   object
40  HeatingQC      1460 non-null   object
41  CentralAir     1460 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1460 non-null   int64
44  2ndFlrSF       1460 non-null   int64
45  LowQualFinSF   1460 non-null   int64
46  GrLivArea      1460 non-null   int64
47  BsmtFullBath   1460 non-null   int64
48  BsmtHalfBath   1460 non-null   int64
49  FullBath       1460 non-null   int64
50  HalfBath       1460 non-null   int64
51  BedroomAbvGr   1460 non-null   int64
52  KitchenAbvGr   1460 non-null   int64
```

```
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

[5]:
```python
# To get the description of the dataset
housing.describe()
```

[5]:
```
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  …
```

```
std        1.112799    30.202904    20.645407   181.066207   456.098091  …
min        1.000000  1872.000000  1950.000000     0.000000     0.000000  …
25%        5.000000  1954.000000  1967.000000     0.000000     0.000000  …
50%        5.000000  1973.000000  1994.000000     0.000000   383.500000  …
75%        6.000000  2000.000000  2004.000000   166.000000   712.250000  …
max        9.000000  2010.000000  2010.000000  1600.000000  5644.000000  …

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]
```

1.1.3 Data Cleaning

1.1.4 Finding the Missing values

[6]:
```python
housing.isnull().sum()
```

[6]:
```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

[7]:
```
# Checking for percentage nulls
round(100*(housing.isnull().sum()/len(housing.index)), 2)
```

[7]:
```
Id                0.00
MSSubClass        0.00
MSZoning          0.00
LotFrontage      17.74
LotArea           0.00
                 …
MoSold            0.00
YrSold            0.00
SaleType          0.00
SaleCondition     0.00
SalePrice         0.00
Length: 81, dtype: float64
```

1.1.5 Outlier Check

[8]:
```
#Checking for outlier in the numerical columns
housing.describe(percentiles=[.25,.5,.75,.90,.95,.99])
```

[8]:
```
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
90%    1314.100000   120.000000    96.000000   14381.700000     8.000000
95%    1387.050000   160.000000   107.000000   17401.150000     8.000000
99%    1445.410000   190.000000   141.000000   37567.640000    10.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  …
std       1.112799    30.202904     20.645407   181.066207   456.098091  …
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  …
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  …
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  …
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  …
90%       7.000000  2006.000000   2006.000000   335.000000  1065.500000  …
95%       8.000000  2007.000000   2007.000000   456.000000  1274.000000  …
99%       9.000000  2009.000000   2009.000000   791.920000  1572.410000  …
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  …
```

```
       WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
90%     262.000000   130.000000     112.000000     0.000000     0.000000
95%     335.000000   175.050000     180.150000     0.000000   160.000000
99%     505.460000   285.820000     261.050000   168.000000   268.050000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
90%       0.000000      0.000000    10.000000  2010.000000  278000.000000
95%       0.000000      0.000000    11.000000  2010.000000  326100.000000
99%       0.000000    700.000000    12.000000  2010.000000  442567.010000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[11 rows x 38 columns]
```

1.1.6 Remove outliers

[9]:
```python
def remove_outliers(x,y):
    q1 = x[y].quantile(0.25)
    q3 = x[y].quantile(0.75)
    value = q3-q1
    lower_value  = q1-1.5*value
    higer_value = q3+1.5*value
    out= x[(x[y]<higer_value) & (x[y]>lower_value)]
    return out
```

[10]:
```python
#Checking the shape of the dataframe
housing.shape
```

[10]:
```
(1460, 81)
```

[11]:
```python
# since, it is clear that there are multiple columns with high nulls, lets␣
↪group them together
housing.columns[housing.isnull().any()]
```

```
null = housing.isnull().sum()/len(housing)*100
null = null[null>0]
null.sort_values(inplace=True, ascending=False)
null
```

[11]:
```
PoolQC          99.520548
MiscFeature     96.301370
Alley           93.767123
Fence           80.753425
FireplaceQu     47.260274
LotFrontage     17.739726
GarageType       5.547945
GarageYrBlt      5.547945
GarageFinish     5.547945
GarageQual       5.547945
GarageCond       5.547945
BsmtExposure     2.602740
BsmtFinType2     2.602740
BsmtFinType1     2.534247
BsmtCond         2.534247
BsmtQual         2.534247
MasVnrArea       0.547945
MasVnrType       0.547945
Electrical       0.068493
dtype: float64
```

According to the data dictionary provided, the nulls in these columns indicates the absence of facility which may affect the price

- Hence, we will first impute the categorical variables with 'None'

[12]:
```
housing_df = housing.copy()
null_with_meaning = ["Alley", "MasVnrType", "BsmtQual", "BsmtCond",␣
 ↪"BsmtExposure", "BsmtFinType1", "BsmtFinType2", "FireplaceQu", "GarageType",␣
 ↪"GarageFinish", "GarageQual", "GarageCond", "PoolQC", "Fence", "MiscFeature"]
for i in null_with_meaning:
    housing_df[i].fillna("none", inplace=True)
```

[13]:
```
# Check nulls once again

housing_df.columns[housing_df.isnull().any()]

null_2 = housing_df.isnull().sum()/len(housing_df)*100
null_2 = null_2[null_2>0]
null_2.sort_values(inplace=True, ascending=False)
null_2
```

[13]:
```
LotFrontage    17.739726
GarageYrBlt     5.547945
MasVnrArea      0.547945
Electrical      0.068493
dtype: float64
```

Check the `LotFrontage`,`GarageYrBlt`,`MasVnrArea`,`Electrical` features data

[14]:
```
# Will check these columns one by one
housing_df['LotFrontage'].describe()
```

[14]:
```
count    1201.000000
mean       70.049958
std        24.284752
min        21.000000
25%        59.000000
50%        69.000000
75%        80.000000
max       313.000000
Name: LotFrontage, dtype: float64
```

[15]:
```
housing_df['GarageYrBlt'].describe()
```

[15]:
```
count    1379.000000
mean     1978.506164
std        24.689725
min      1900.000000
25%      1961.000000
50%      1980.000000
75%      2002.000000
max      2010.000000
Name: GarageYrBlt, dtype: float64
```

[16]:
```
housing_df['MasVnrArea'].describe()
```

[16]:
```
count    1452.000000
mean      103.685262
std       181.066207
min         0.000000
25%         0.000000
50%         0.000000
75%       166.000000
max      1600.000000
Name: MasVnrArea, dtype: float64
```

[17]:
```
housing_df['Electrical'].describe()
```

[17]:
```
count      1459
unique        5
top       SBrkr
freq       1334
Name: Electrical, dtype: object
```

1.1.7 Impute the `Neighborhood` data

- As per the data dictionary "LotFrontage" is Linear feet of street connected to property.
- Since it is a numeric with a fair distribution, it can be imputed with similar 'Neighborhood' values

[18]:
```
# As per the data dictionary "LotFrontage" is Linear feet of street connected␣
↪to property.
# Since it is a numeric with a fair distribution, it can be imputed with␣
↪similar 'Neighborhood' values
housing_df['LotFrontage'] = housing_df.groupby("Neighborhood")["LotFrontage"].
↪transform(lambda x: x.fillna(x.median()))
housing_df["GarageYrBlt"].fillna(housing_df["GarageYrBlt"].median(),␣
↪inplace=True)
housing_df["MasVnrArea"].fillna(housing_df["MasVnrArea"].median(), inplace=True)
housing_df["Electrical"].dropna(inplace=True)
```

1.1.8 Cross check the `LotFrontage`,`GarageYrBlt`,`MasVnrArea` and `Electrical`features after imputed `Neighborhood` data

[19]:
```
# Crosscheck the updated 'LotFrontage' column
housing_df['LotFrontage'].describe()
```

[19]:
```
count    1460.000000
mean       70.199658
std        22.431902
min        21.000000
25%        60.000000
50%        70.000000
75%        80.000000
max       313.000000
Name: LotFrontage, dtype: float64
```

[20]:
```
housing_df['GarageYrBlt'].describe()
```

[20]:
```
count    1460.000000
mean     1978.589041
std        23.997022
min      1900.000000
25%      1962.000000
```

```
50%      1980.000000
75%      2001.000000
max      2010.000000
Name: GarageYrBlt, dtype: float64
```

[21]:
```
housing_df['MasVnrArea'].describe()
```

[21]:
```
count    1460.000000
mean      103.117123
std       180.731373
min         0.000000
25%         0.000000
50%         0.000000
75%       164.250000
max      1600.000000
Name: MasVnrArea, dtype: float64
```

[22]:
```
housing_df['Electrical'].describe()
```

[22]:
```
count      1459
unique        5
top       SBrkr
freq       1334
Name: Electrical, dtype: object
```

[23]:
```
# Check the no. of rows retained
len(housing_df.index)
len(housing_df.index)/1460
```

[23]:
```
1.0
```

2.

1.2 Exploratory Data Analysis (EDA)

All numeric (float and int) variables in given the dataset

[24]:
```
data_numeric = housing_df.select_dtypes(include=['float64', 'int64'])
data_numeric.head()
```

[24]:
```
   Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  \
0   1          60         65.0     8450            7            5       2003
1   2          20         80.0     9600            6            8       1976
2   3          60         68.0    11250            7            5       2001
3   4          70         60.0     9550            7            5       1915
4   5          60         84.0    14260            8            5       2000

   YearRemodAdd  MasVnrArea  BsmtFinSF1  …  WoodDeckSF  OpenPorchSF  \
0          2003       196.0         706  …           0           61
```

```
1          1976       0.0         978  …          298            0
2          2002     162.0         486  …            0           42
3          1970       0.0         216  …            0           35
4          2000     350.0         655  …          192           84

   EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  MoSold  YrSold  \
0              0          0            0         0        0       2    2008
1              0          0            0         0        0       5    2007
2              0          0            0         0        0       9    2008
3            272          0            0         0        0       2    2006
4              0          0            0         0        0      12    2008

   SalePrice
0     208500
1     181500
2     223500
3     140000
4     250000

[5 rows x 38 columns]
```

[25]:
```python
# Dropping ID Column
data_numeric = data_numeric.drop(['Id'], axis=1)
data_numeric.head()
```

[25]:
```
   MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  \
0          60         65.0     8450            7            5       2003
1          20         80.0     9600            6            8       1976
2          60         68.0    11250            7            5       2001
3          70         60.0     9550            7            5       1915
4          60         84.0    14260            8            5       2000

   YearRemodAdd  MasVnrArea  BsmtFinSF1  BsmtFinSF2  …  WoodDeckSF  \
0          2003       196.0         706           0  …           0
1          1976         0.0         978           0  …         298
2          2002       162.0         486           0  …           0
3          1970         0.0         216           0  …           0
4          2000       350.0         655           0  …         192

   OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  \
0           61              0          0            0         0        0
1            0              0          0            0         0        0
2           42              0          0            0         0        0
3           35            272          0            0         0        0
4           84              0          0            0         0        0

   MoSold  YrSold  SalePrice
```

```
0         2     2008       208500
1         5     2007       181500
2         9     2008       223500
3         2     2006       140000
4        12     2008       250000

[5 rows x 37 columns]
```

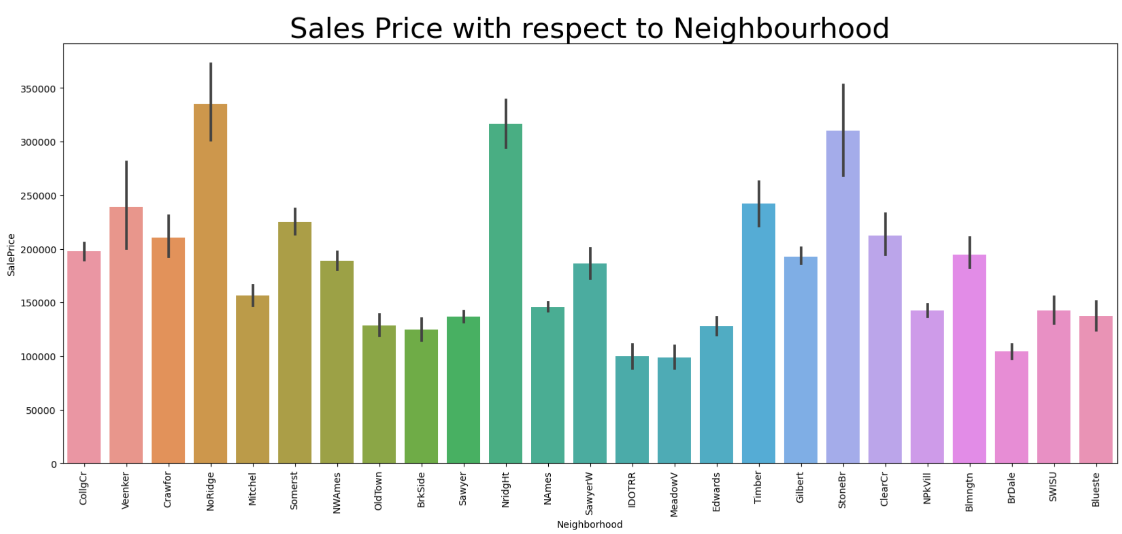
1.2.1 Visualising the Data for better understand the data and variables

The most important step - understanding the data.

- If there is some obvious multicollinearity going on, this is the first place to catch it
- Here's where you'll also identify if some predictors directly have a strong association with the outcome variable We'll visualise our data using matplotlib and seaborn.

Target variable 'sale Price' vs a few select columns

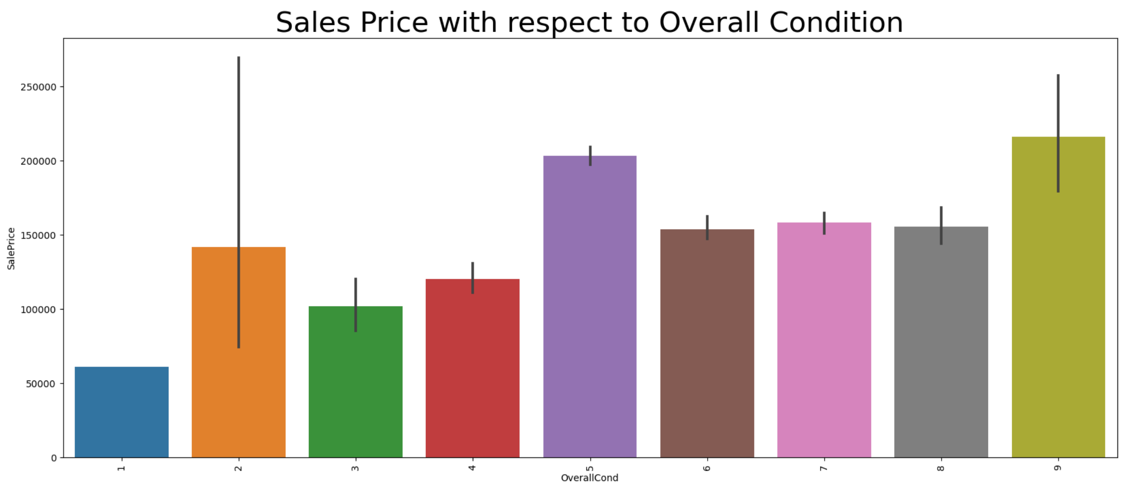
[26]:
```python
# plot 'Sale Price' with respect to 'Neighborhood'
plt.figure(figsize=(20, 8))
sns.barplot(x="Neighborhood", y="SalePrice", data= housing_df)
plt.title("Sales Price with respect to Neighbourhood",fontsize=30)
plt.xticks(rotation=90)
plt.show()
```



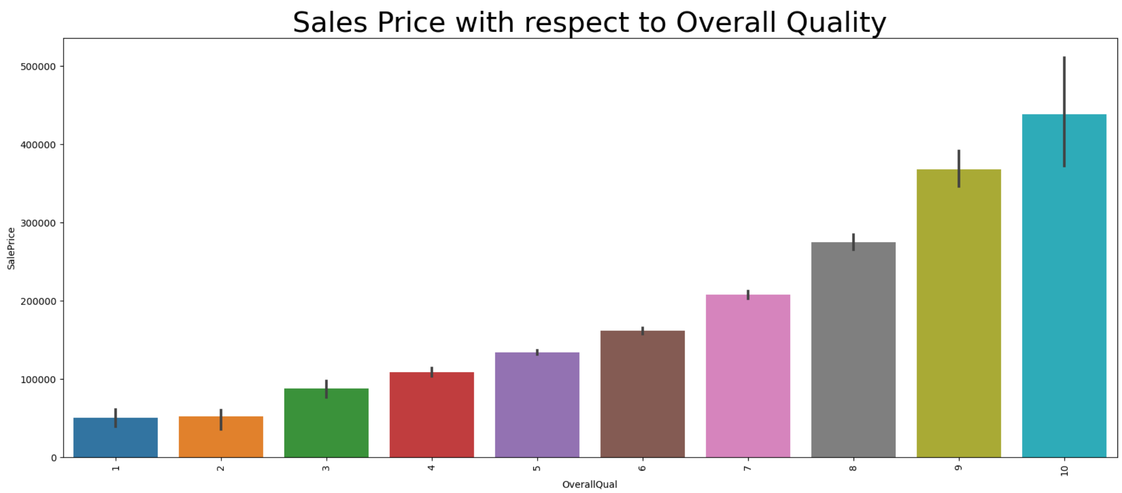
Properties in some of the Neighborhoods are high priced.

[27]:
```python
# plot 'overall condition' with respect to 'Saleprice'

plt.figure(figsize=(20, 8))
sns.barplot(x="OverallCond", y="SalePrice", data= housing_df)
plt.title("Sales Price with respect to Overall Condition",fontsize=30)
```

```
plt.xticks(rotation=90)
plt.show()
```

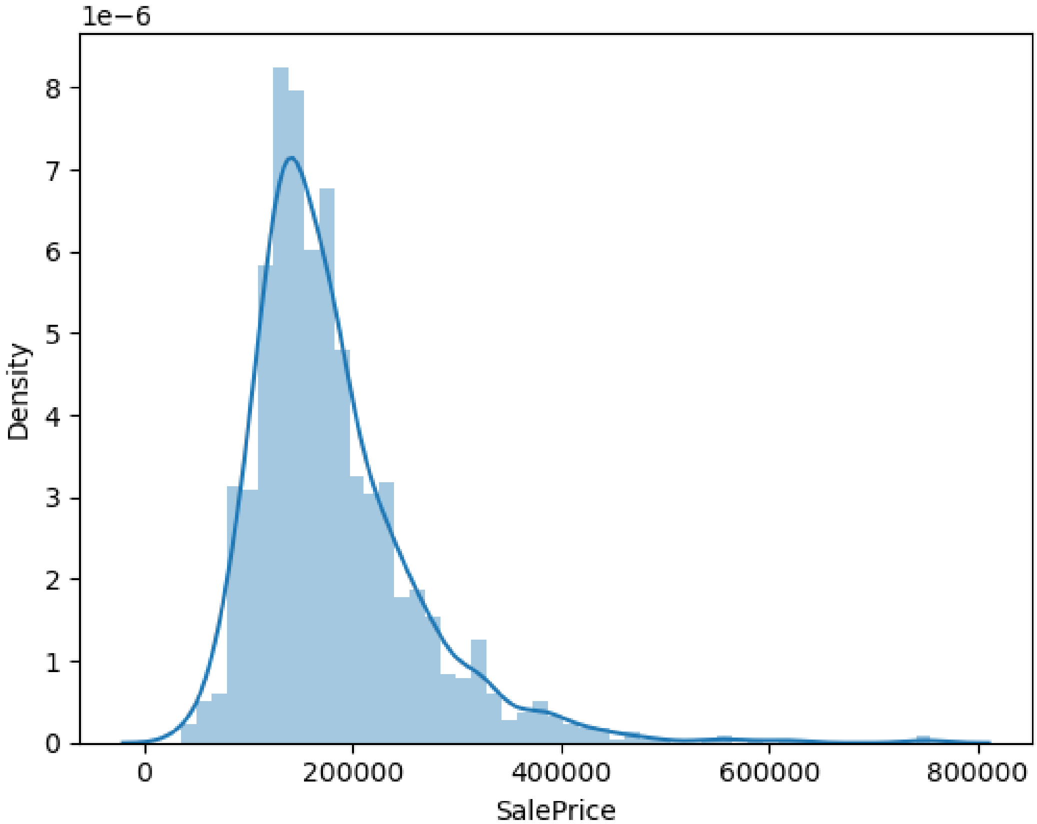


[28]:
```
# plot 'overall quality' with respect to 'Saleprice'

plt.figure(figsize=(20, 8))
sns.barplot(x="OverallQual", y="SalePrice", data= housing_df)
plt.title("Sales Price with respect to Overall Quality",fontsize=30)
plt.xticks(rotation=90)
plt.show()
```



Increase in the overall quality has a direct positive effect on the `sale price`

[29]:
```
sns.distplot(housing_df['SalePrice'])
plt.show()
```



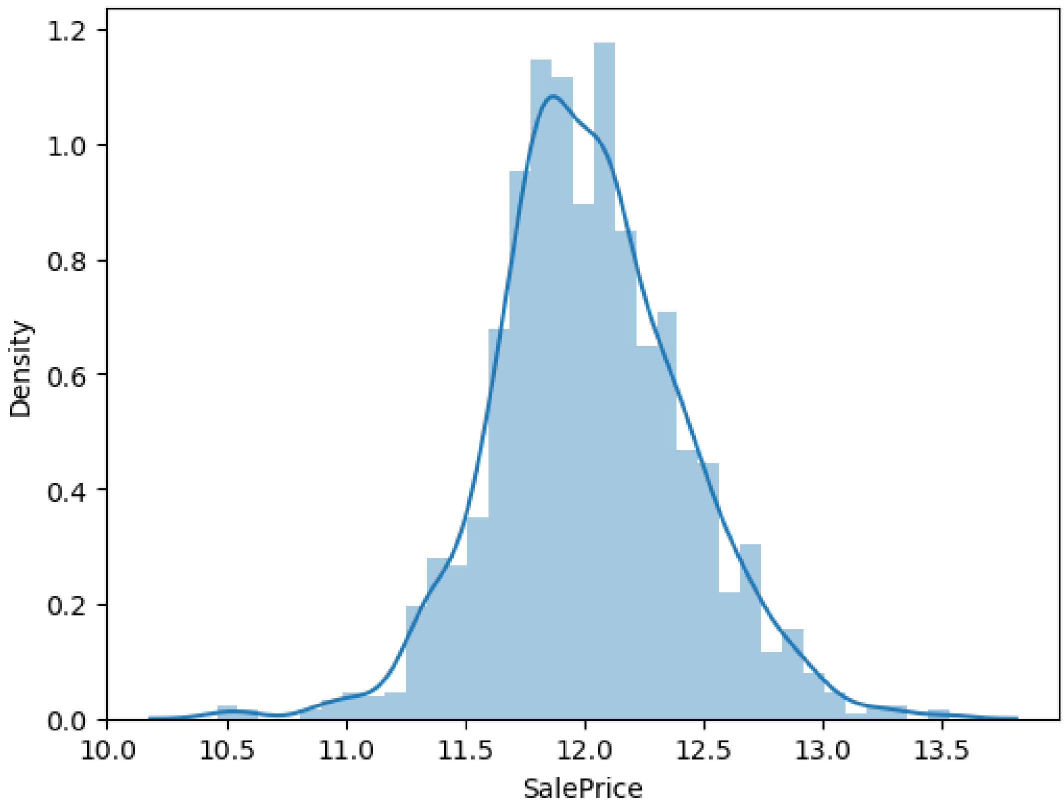
[30]:
```
data_raw = housing_df.copy
```

Since the `Saleprice` figures are skewed towards left, we will apply the log transformation to obtain a centralized data

[31]:
```
#Log Transformation
housing_df['SalePrice']=np.log1p(housing_df['SalePrice'])
```

[32]:
```
sns.distplot(housing_df['SalePrice'])
```

[32]: <AxesSubplot:xlabel='SalePrice', ylabel='Density'>



[33]:
```python
# correlation matrix
cor = data_numeric.corr()
cor
```

[33]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	\
MSSubClass	1.000000	-0.370367	-0.139781	0.032628	-0.059316	
LotFrontage	-0.370367	1.000000	0.335957	0.239546	-0.043595	
LotArea	-0.139781	0.335957	1.000000	0.105806	-0.005636	
OverallQual	0.032628	0.239546	0.105806	1.000000	-0.091932	
OverallCond	-0.059316	-0.043595	-0.005636	-0.091932	1.000000	
YearBuilt	0.027850	0.120999	0.014228	0.572323	-0.375983	
YearRemodAdd	0.040581	0.084550	0.013788	0.550684	0.073741	
MasVnrArea	0.023573	0.195830	0.103321	0.407252	-0.125694	
BsmtFinSF1	-0.069836	0.232576	0.214103	0.239666	-0.046231	
BsmtFinSF2	-0.065649	0.052237	0.111170	-0.059119	0.040229	
BsmtUnfSF	-0.140759	0.119174	-0.002618	0.308159	-0.136841	
TotalBsmtSF	-0.238518	0.381038	0.260833	0.537808	-0.171098	
1stFlrSF	-0.251758	0.434109	0.299475	0.476224	-0.144203	
2ndFlrSF	0.307886	0.075686	0.050986	0.295493	0.028942	
LowQualFinSF	0.046474	0.031873	0.004779	-0.030429	0.025494	
GrLivArea	0.074853	0.385190	0.263116	0.593007	-0.079686	

BsmtFullBath	0.003491	0.107226	0.158155	0.111098	-0.054942
BsmtHalfBath	-0.002333	0.006620	0.048046	-0.040150	0.117821
FullBath	0.131608	0.186561	0.126031	0.550600	-0.194149
HalfBath	0.177354	0.054190	0.014259	0.273458	-0.060769
BedroomAbvGr	-0.023438	0.245232	0.119690	0.101676	0.012980
KitchenAbvGr	0.281721	-0.005627	-0.017784	-0.183882	-0.087001
TotRmsAbvGrd	0.040380	0.332619	0.190015	0.427452	-0.057583
Fireplaces	-0.045569	0.249295	0.271364	0.396765	-0.023820
GarageYrBlt	0.081396	0.062380	-0.025865	0.514231	-0.306276
GarageCars	-0.040110	0.281393	0.154871	0.600671	-0.185758
GarageArea	-0.098672	0.339085	0.180403	0.562022	-0.151521
WoodDeckSF	-0.012579	0.088736	0.171698	0.238923	-0.003334
OpenPorchSF	-0.006100	0.141734	0.084774	0.308819	-0.032589
EnclosedPorch	-0.012037	0.008057	-0.018340	-0.113937	0.070356
3SsnPorch	-0.043825	0.064654	0.020423	0.030371	0.025504
ScreenPorch	-0.026030	0.041063	0.043160	0.064886	0.054811
PoolArea	0.008283	0.174567	0.077672	0.065166	-0.001985
MiscVal	-0.007683	0.005332	0.038068	-0.031406	0.068777
MoSold	-0.013585	0.007370	0.001205	0.070815	-0.003511
YrSold	-0.021407	0.004756	-0.014261	-0.027347	0.043950
SalePrice	-0.084284	0.349876	0.263843	0.790982	-0.077856

	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	\
MSSubClass	0.027850	0.040581	0.023573	-0.069836	-0.065649	
LotFrontage	0.120999	0.084550	0.195830	0.232576	0.052237	
LotArea	0.014228	0.013788	0.103321	0.214103	0.111170	
OverallQual	0.572323	0.550684	0.407252	0.239666	-0.059119	
OverallCond	-0.375983	0.073741	-0.125694	-0.046231	0.040229	
YearBuilt	1.000000	0.592855	0.311600	0.249503	-0.049107	
YearRemodAdd	0.592855	1.000000	0.176529	0.128451	-0.067759	
MasVnrArea	0.311600	0.176529	1.000000	0.261256	-0.071330	
BsmtFinSF1	0.249503	0.128451	0.261256	1.000000	-0.050117	
BsmtFinSF2	-0.049107	-0.067759	-0.071330	-0.050117	1.000000	
BsmtUnfSF	0.149040	0.181133	0.113862	-0.495251	-0.209294	
TotalBsmtSF	0.391452	0.291066	0.360067	0.522396	0.104810	
1stFlrSF	0.281986	0.240379	0.339850	0.445863	0.097117	
2ndFlrSF	0.010308	0.140024	0.173800	-0.137079	-0.099260	
LowQualFinSF	-0.183784	-0.062419	-0.068628	-0.064503	0.014807	
GrLivArea	0.199010	0.287389	0.388052	0.208171	-0.009640	
BsmtFullBath	0.187599	0.119470	0.083010	0.649212	0.158678	
BsmtHalfBath	-0.038162	-0.012337	0.027403	0.067418	0.070948	
FullBath	0.468271	0.439046	0.272999	0.058543	-0.076444	
HalfBath	0.242656	0.183331	0.199108	0.004262	-0.032148	
BedroomAbvGr	-0.070651	-0.040581	0.102775	-0.107355	-0.015728	
KitchenAbvGr	-0.174800	-0.149598	-0.038450	-0.081007	-0.040751	
TotRmsAbvGrd	0.095589	0.191740	0.279568	0.044316	-0.035227	
Fireplaces	0.147716	0.112581	0.247015	0.260011	0.046921	

GarageYrBlt	0.777182	0.616444	0.244444	0.148782	-0.087684
GarageCars	0.537850	0.420622	0.361945	0.224054	-0.038264
GarageArea	0.478954	0.371600	0.370884	0.296970	-0.018227
WoodDeckSF	0.224880	0.205726	0.159991	0.204306	0.067898
OpenPorchSF	0.188686	0.226298	0.122528	0.111761	0.003093
EnclosedPorch	-0.387268	-0.193919	-0.109907	-0.102303	0.036543
3SsnPorch	0.031355	0.045286	0.019144	0.026451	-0.029993
ScreenPorch	-0.050364	-0.038740	0.062248	0.062021	0.088871
PoolArea	0.004950	0.005829	0.011928	0.140491	0.041709
MiscVal	-0.034383	-0.010286	-0.029512	0.003571	0.004940
MoSold	0.012398	0.021490	-0.006723	-0.015727	-0.015211
YrSold	-0.013618	0.035743	-0.008317	0.014359	0.031706
SalePrice	0.522897	0.507101	0.472614	0.386420	-0.011378

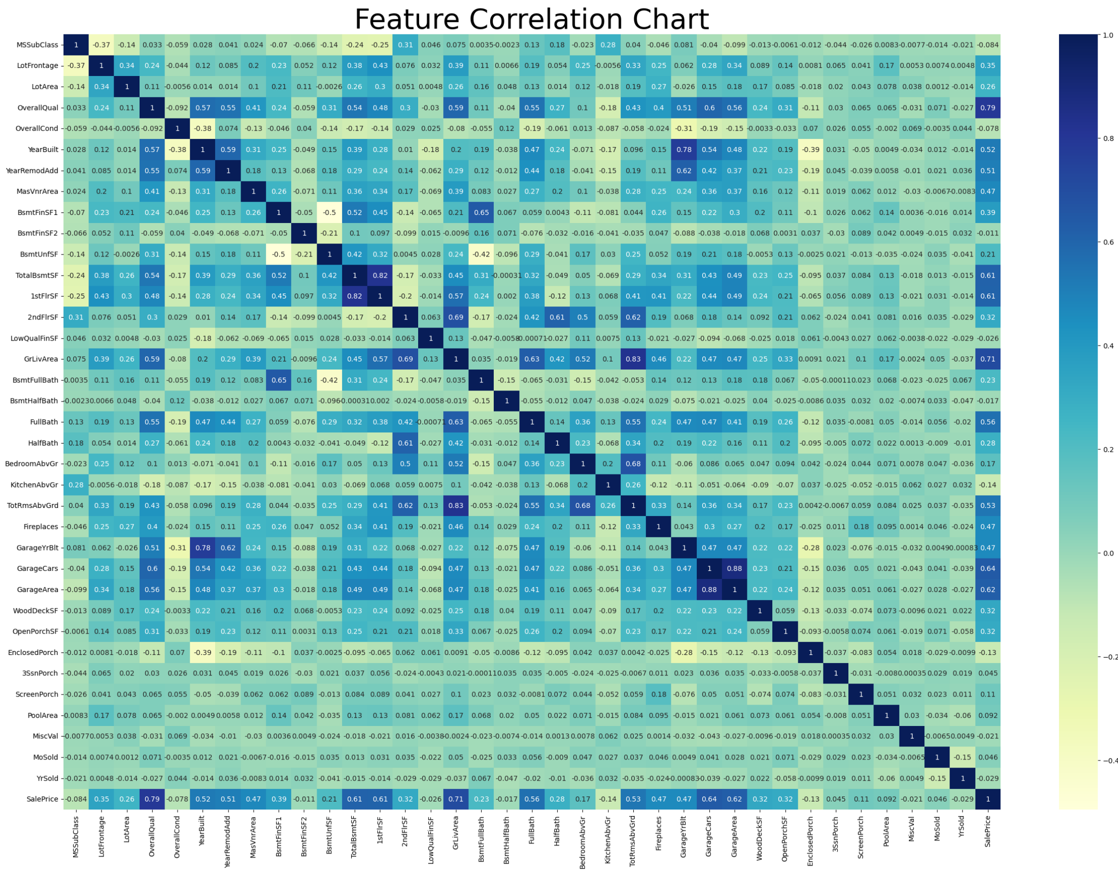
	…	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	\
MSSubClass	…	-0.012579	-0.006100	-0.012037	-0.043825	
LotFrontage	…	0.088736	0.141734	0.008057	0.064654	
LotArea	…	0.171698	0.084774	-0.018340	0.020423	
OverallQual	…	0.238923	0.308819	-0.113937	0.030371	
OverallCond	…	-0.003334	-0.032589	0.070356	0.025504	
YearBuilt	…	0.224880	0.188686	-0.387268	0.031355	
YearRemodAdd	…	0.205726	0.226298	-0.193919	0.045286	
MasVnrArea	…	0.159991	0.122528	-0.109907	0.019144	
BsmtFinSF1	…	0.204306	0.111761	-0.102303	0.026451	
BsmtFinSF2	…	0.067898	0.003093	0.036543	-0.029993	
BsmtUnfSF	…	-0.005316	0.129005	-0.002538	0.020764	
TotalBsmtSF	…	0.232019	0.247264	-0.095478	0.037384	
1stFlrSF	…	0.235459	0.211671	-0.065292	0.056104	
2ndFlrSF	…	0.092165	0.208026	0.061989	-0.024358	
LowQualFinSF	…	-0.025444	0.018251	0.061081	-0.004296	
GrLivArea	…	0.247433	0.330224	0.009113	0.020643	
BsmtFullBath	…	0.175315	0.067341	-0.049911	-0.000106	
BsmtHalfBath	…	0.040161	-0.025324	-0.008555	0.035114	
FullBath	…	0.187703	0.259977	-0.115093	0.035353	
HalfBath	…	0.108080	0.199740	-0.095317	-0.004972	
BedroomAbvGr	…	0.046854	0.093810	0.041570	-0.024478	
KitchenAbvGr	…	-0.090130	-0.070091	0.037312	-0.024600	
TotRmsAbvGrd	…	0.165984	0.234192	0.004151	-0.006683	
Fireplaces	…	0.200019	0.169405	-0.024822	0.011257	
GarageYrBlt	…	0.219093	0.217921	-0.284972	0.023130	
GarageCars	…	0.226342	0.213569	-0.151434	0.035765	
GarageArea	…	0.224666	0.241435	-0.121777	0.035087	
WoodDeckSF	…	1.000000	0.058661	-0.125989	-0.032771	
OpenPorchSF	…	0.058661	1.000000	-0.093079	-0.005842	
EnclosedPorch	…	-0.125989	-0.093079	1.000000	-0.037305	
3SsnPorch	…	-0.032771	-0.005842	-0.037305	1.000000	
ScreenPorch	…	-0.074181	0.074304	-0.082864	-0.031436	

PoolArea	…	0.073378	0.060762	0.054203	-0.007992
MiscVal	…	-0.009551	-0.018584	0.018361	0.000354
MoSold	…	0.021011	0.071255	-0.028887	0.029474
YrSold	…	0.022270	-0.057619	-0.009916	0.018645
SalePrice	…	0.324413	0.315856	-0.128578	0.044584

	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SalePrice
MSSubClass	-0.026030	0.008283	-0.007683	-0.013585	-0.021407	-0.084284
LotFrontage	0.041063	0.174567	0.005332	0.007370	0.004756	0.349876
LotArea	0.043160	0.077672	0.038068	0.001205	-0.014261	0.263843
OverallQual	0.064886	0.065166	-0.031406	0.070815	-0.027347	0.790982
OverallCond	0.054811	-0.001985	0.068777	-0.003511	0.043950	-0.077856
YearBuilt	-0.050364	0.004950	-0.034383	0.012398	-0.013618	0.522897
YearRemodAdd	-0.038740	0.005829	-0.010286	0.021490	0.035743	0.507101
MasVnrArea	0.062248	0.011928	-0.029512	-0.006723	-0.008317	0.472614
BsmtFinSF1	0.062021	0.140491	0.003571	-0.015727	0.014359	0.386420
BsmtFinSF2	0.088871	0.041709	0.004940	-0.015211	0.031706	-0.011378
BsmtUnfSF	-0.012579	-0.035092	-0.023837	0.034888	-0.041258	0.214479
TotalBsmtSF	0.084489	0.126053	-0.018479	0.013196	-0.014969	0.613581
1stFlrSF	0.088758	0.131525	-0.021096	0.031372	-0.013604	0.605852
2ndFlrSF	0.040606	0.081487	0.016197	0.035164	-0.028700	0.319334
LowQualFinSF	0.026799	0.062157	-0.003793	-0.022174	-0.028921	-0.025606
GrLivArea	0.101510	0.170205	-0.002416	0.050240	-0.036526	0.708624
BsmtFullBath	0.023148	0.067616	-0.023047	-0.025361	0.067049	0.227122
BsmtHalfBath	0.032121	0.020025	-0.007367	0.032873	-0.046524	-0.016844
FullBath	-0.008106	0.049604	-0.014290	0.055872	-0.019669	0.560664
HalfBath	0.072426	0.022381	0.001290	-0.009050	-0.010269	0.284108
BedroomAbvGr	0.044300	0.070703	0.007767	0.046544	-0.036014	0.168213
KitchenAbvGr	-0.051613	-0.014525	0.062341	0.026589	0.031687	-0.135907
TotRmsAbvGrd	0.059383	0.083757	0.024763	0.036907	-0.034516	0.533723
Fireplaces	0.184530	0.095074	0.001409	0.046357	-0.024096	0.466929
GarageYrBlt	-0.076181	-0.014735	-0.031779	0.004903	-0.000829	0.466754
GarageCars	0.050494	0.020934	-0.043080	0.040522	-0.039117	0.640409
GarageArea	0.051412	0.061047	-0.027400	0.027974	-0.027378	0.623431
WoodDeckSF	-0.074181	0.073378	-0.009551	0.021011	0.022270	0.324413
OpenPorchSF	0.074304	0.060762	-0.018584	0.071255	-0.057619	0.315856
EnclosedPorch	-0.082864	0.054203	0.018361	-0.028887	-0.009916	-0.128578
3SsnPorch	-0.031436	-0.007992	0.000354	0.029474	0.018645	0.044584
ScreenPorch	1.000000	0.051307	0.031946	0.023217	0.010694	0.111447
PoolArea	0.051307	1.000000	0.029669	-0.033737	-0.059689	0.092404
MiscVal	0.031946	0.029669	1.000000	-0.006495	0.004906	-0.021190
MoSold	0.023217	-0.033737	-0.006495	1.000000	-0.145721	0.046432
YrSold	0.010694	-0.059689	0.004906	-0.145721	1.000000	-0.028923
SalePrice	0.111447	0.092404	-0.021190	0.046432	-0.028923	1.000000

[37 rows x 37 columns]

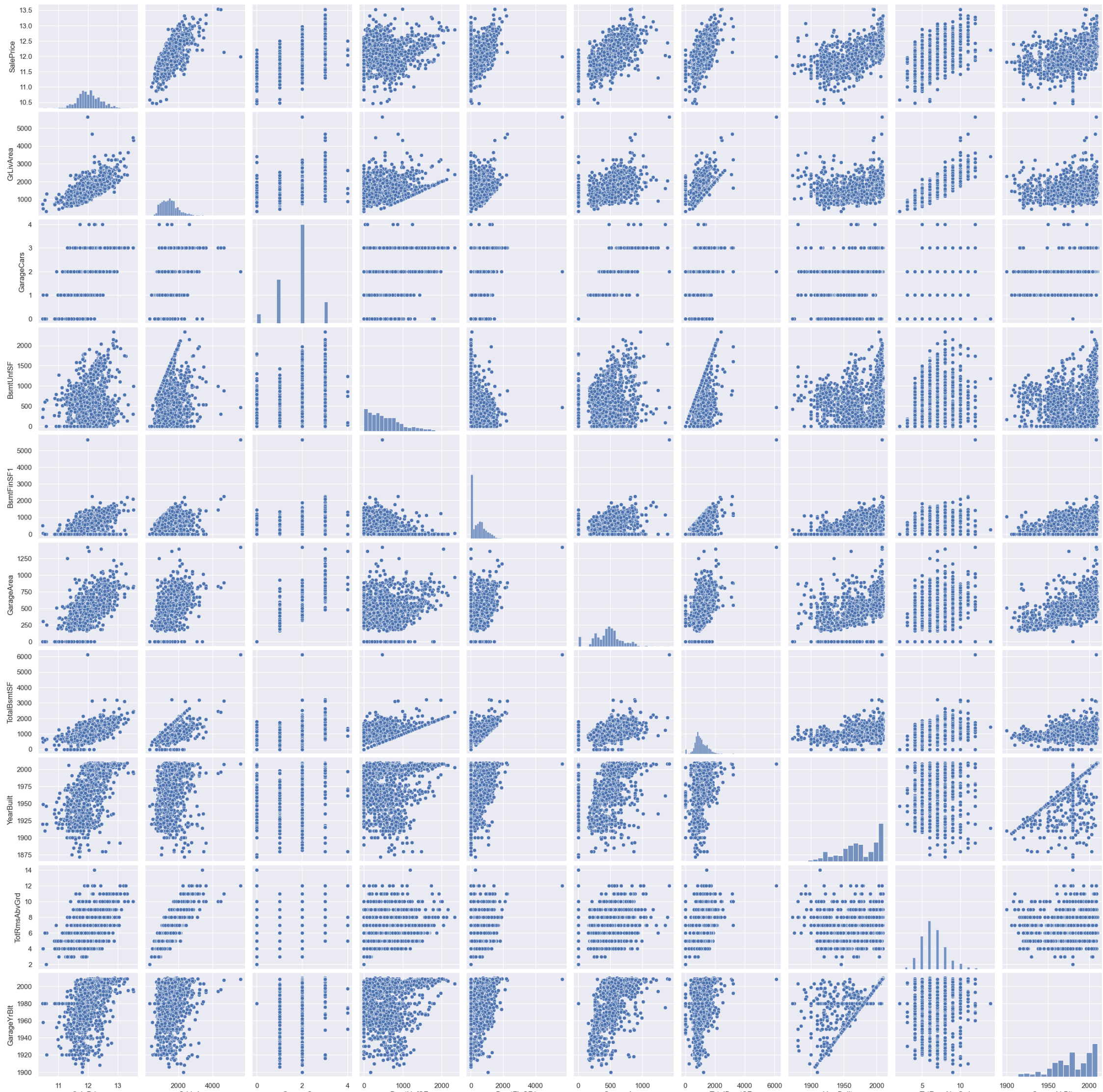
```
[34]: # plotting the correlations on a heatmap
plt.figure(figsize=(30,20))
sns.heatmap(cor, cmap="YlGnBu", annot=True)
plt.title('Feature Correlation Chart',fontsize = 40)
plt.show()
```



Some of the variables are correlated

- Before dropping these columns, we will first check their predictive power

```
[35]: # Checking the same with a pairplot
sns.set()
cols = ['SalePrice', 'GrLivArea', 'GarageCars', 'BsmtUnfSF', 'BsmtFinSF1',␣
↪'GarageArea', 'TotalBsmtSF', 'YearBuilt', 'TotRmsAbvGrd', 'GarageYrBlt']
sns.pairplot(housing_df[cols], size = 2.5)
plt.show()
```



1.2.2 Drop columns that are correlated and not contributing to 'SalePrice'

[36]:
```
housing_df = housing_df.drop(['GarageCars'], axis = 1)
housing_df = housing_df.drop(['BsmtUnfSF'], axis = 1)
housing_df = housing_df.drop(['TotRmsAbvGrd'], axis = 1)
housing_df = housing_df.drop(['GarageYrBlt'], axis = 1)

housing_df.head()
```

[36]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave  none      Reg
1   2          20       RL         80.0     9600   Pave  none      Reg
2   3          60       RL         68.0    11250   Pave  none      IR1
```

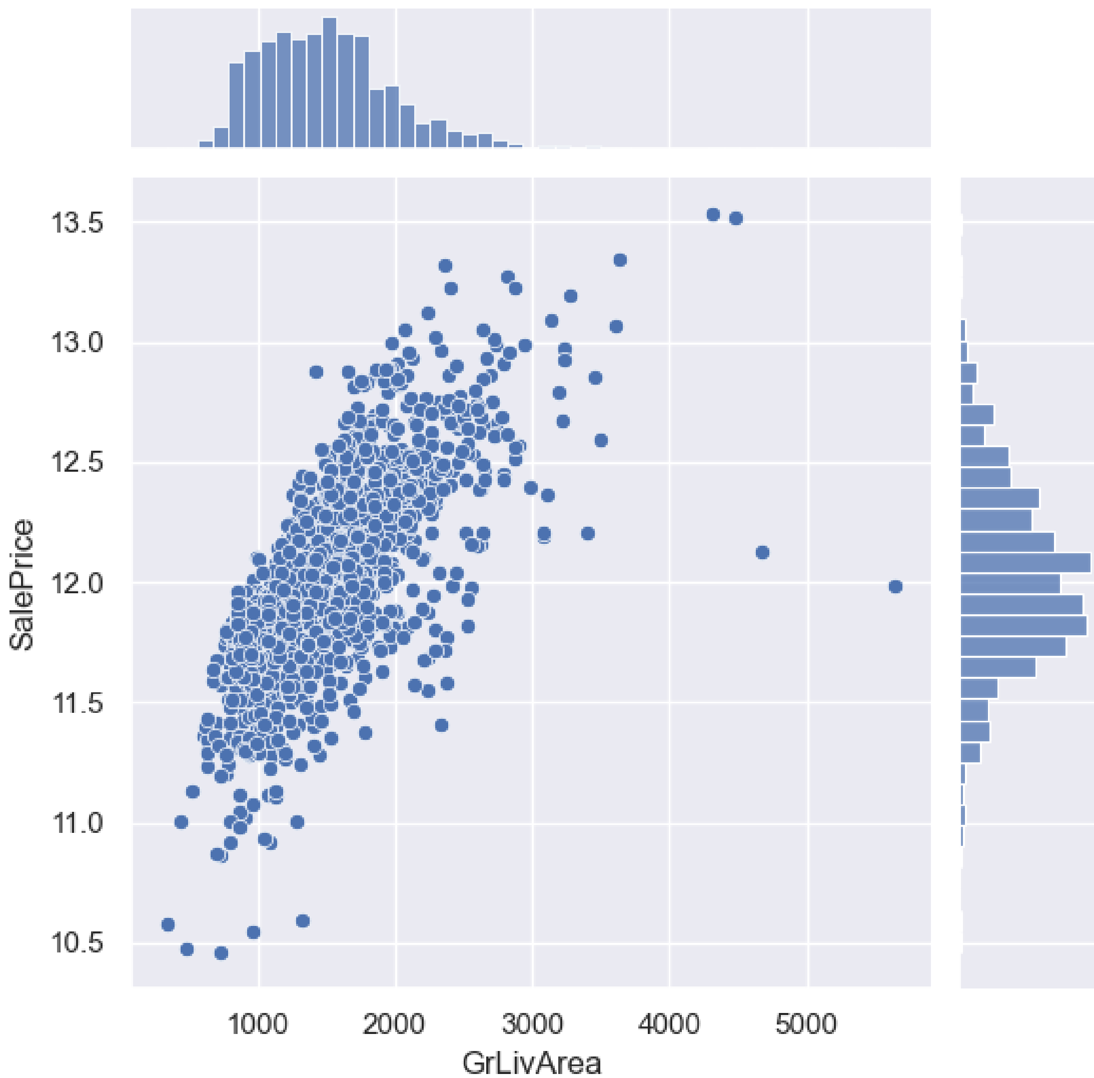
```
3   4          70        RL         60.0     9550    Pave   none       IR1
4   5          60        RL         84.0    14260    Pave   none       IR1

  LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal MoSold  \
0         Lvl    AllPub  …        0   none  none        none       0      2
1         Lvl    AllPub  …        0   none  none        none       0      5
2         Lvl    AllPub  …        0   none  none        none       0      9
3         Lvl    AllPub  …        0   none  none        none       0      2
4         Lvl    AllPub  …        0   none  none        none       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal  12.247699
1   2007        WD         Normal  12.109016
2   2008        WD         Normal  12.317171
3   2006        WD        Abnorml  11.849405
4   2008        WD         Normal  12.429220

[5 rows x 77 columns]
```

[37]:
```python
#Numeric columns
housing_df.select_dtypes(exclude=['object']).head()
```

[37]:
```
   Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  \
0   1          60         65.0     8450            7            5       2003
1   2          20         80.0     9600            6            8       1976
2   3          60         68.0    11250            7            5       2001
3   4          70         60.0     9550            7            5       1915
4   5          60         84.0    14260            8            5       2000

   YearRemodAdd  MasVnrArea  BsmtFinSF1  …  WoodDeckSF  OpenPorchSF  \
0          2003       196.0         706  …           0           61
1          1976         0.0         978  …         298            0
2          2002       162.0         486  …           0           42
3          1970         0.0         216  …           0           35
4          2000       350.0         655  …         192           84

   EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  MoSold  YrSold  \
0              0          0            0         0        0       2    2008
1              0          0            0         0        0       5    2007
2              0          0            0         0        0       9    2008
3            272          0            0         0        0       2    2006
4              0          0            0         0        0      12    2008

   SalePrice
0  12.247699
1  12.109016
2  12.317171
```

```
3   11.849405
4   12.429220

[5 rows x 34 columns]
```

[38]:
```python
# Analyse some important numeric columns
sns.jointplot(x='GrLivArea', y='SalePrice', data=housing_df)
plt.show()
```



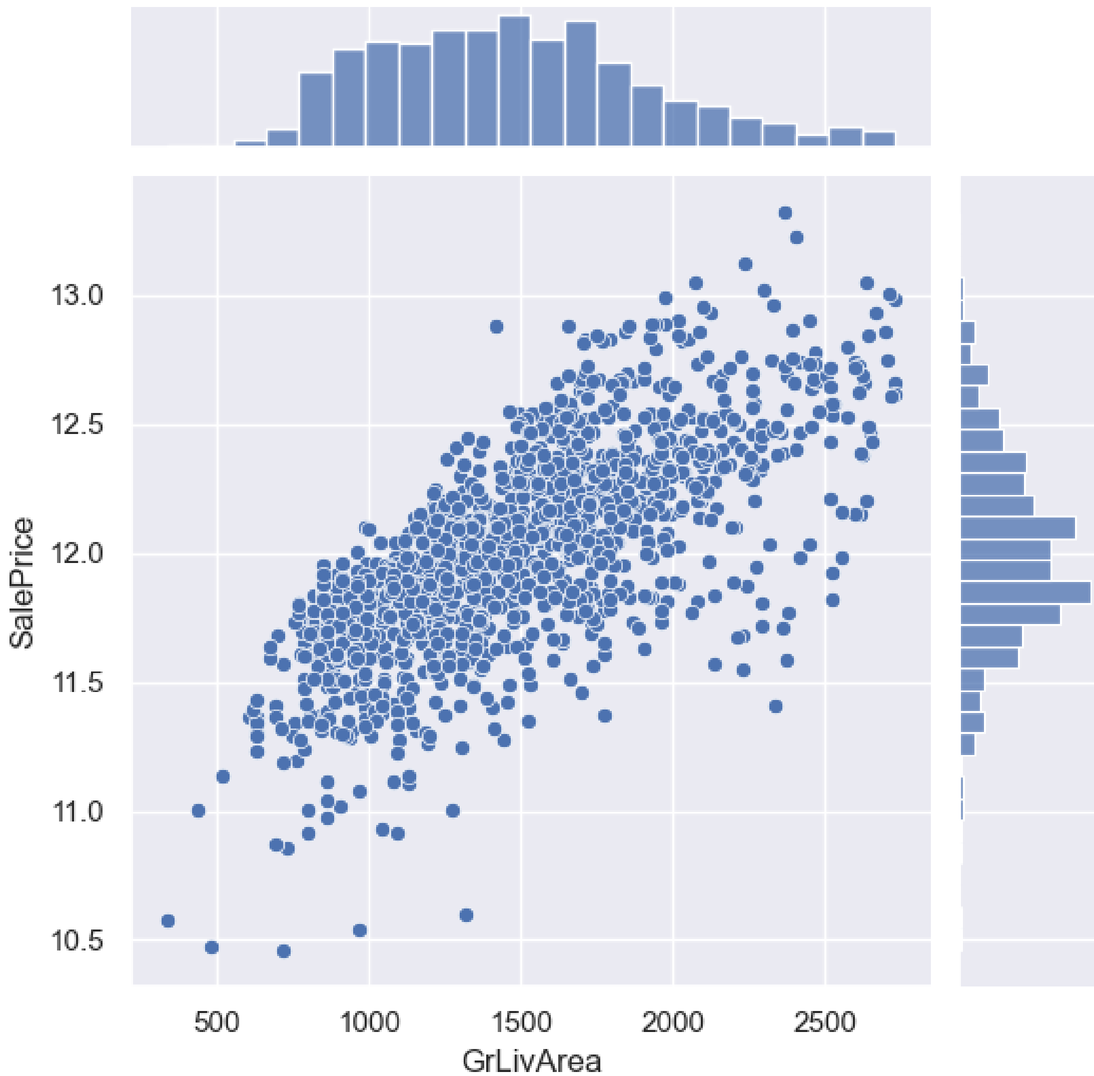
[39]:
```python
# Removing some outliers on lower right side of 'GrLivArea'
housing_df = remove_outliers(housing_df,'GrLivArea')
```

Since the dataset is small it is not advisable to do remove outliers.

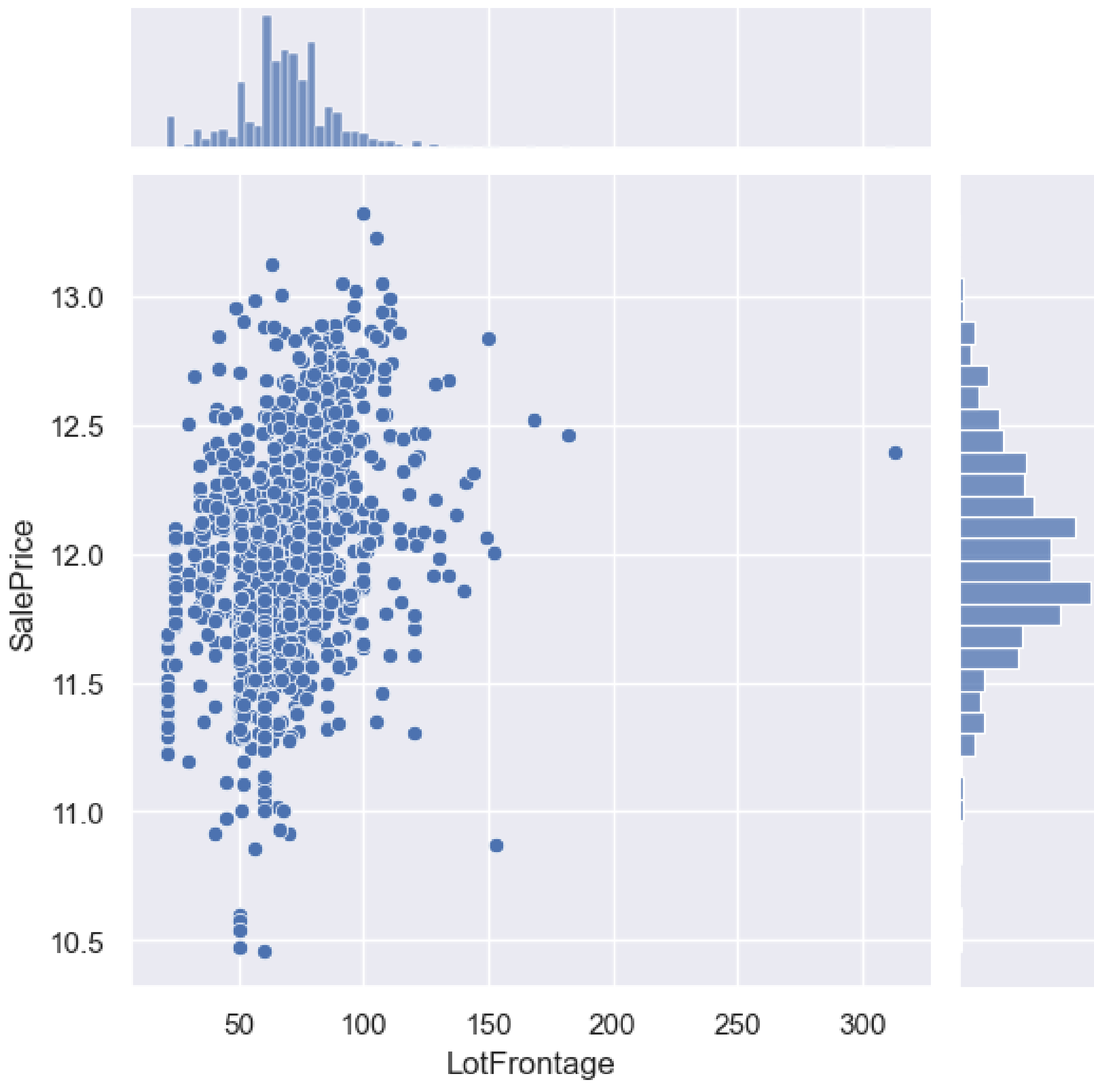
[40]:
```
housing_df.shape
```

[40]: (1429, 77)

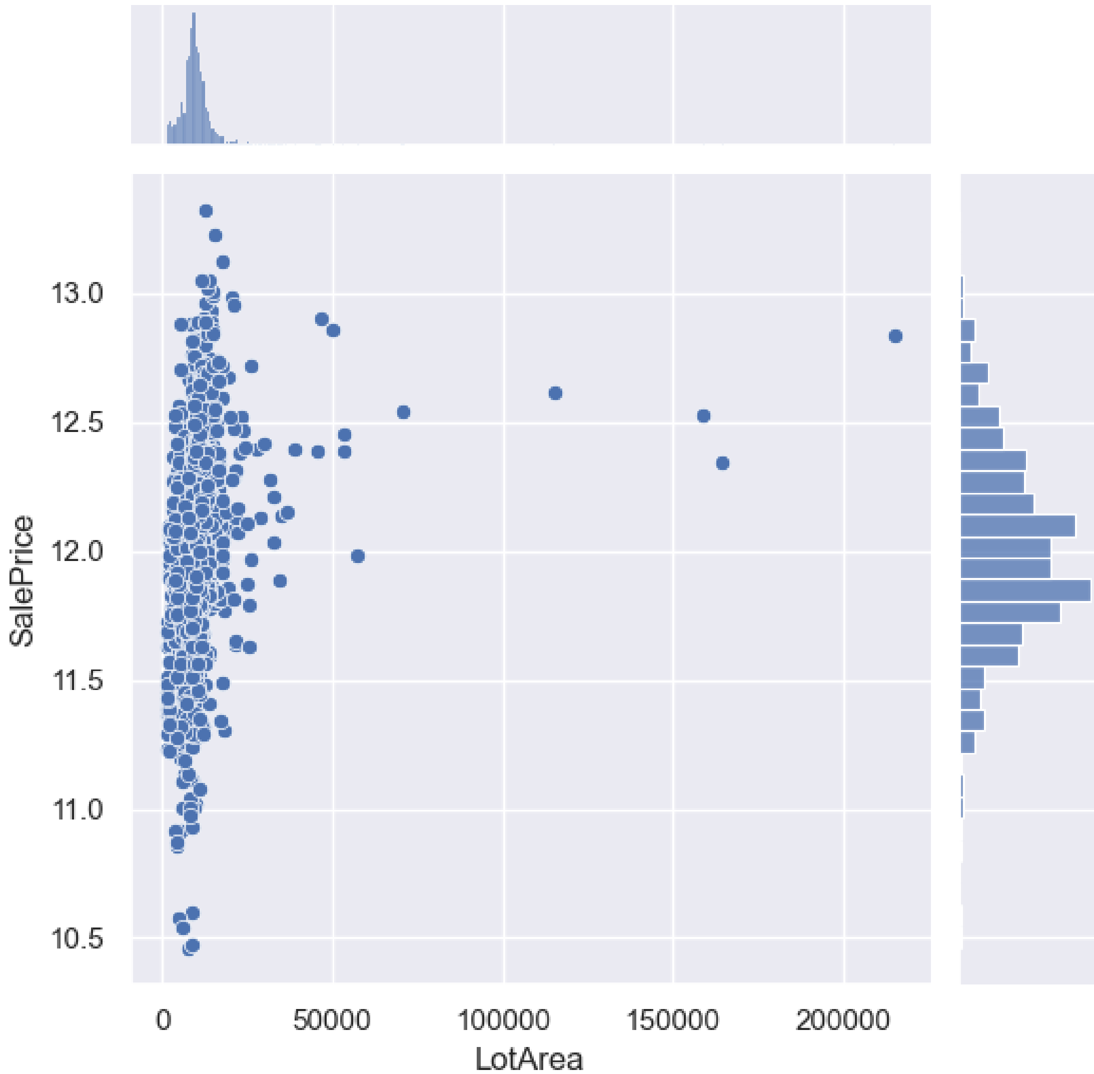
[41]:
```
# Again plotting GeLivArea vs SalePrice
sns.jointplot(x = housing_df['GrLivArea'], y = housing_df['SalePrice'])
plt.show()
```



[42]:
```
# Lot frontage vs SalePrice
sns.jointplot(x = housing_df['LotFrontage'], y = housing_df['SalePrice'])
plt.show()
```

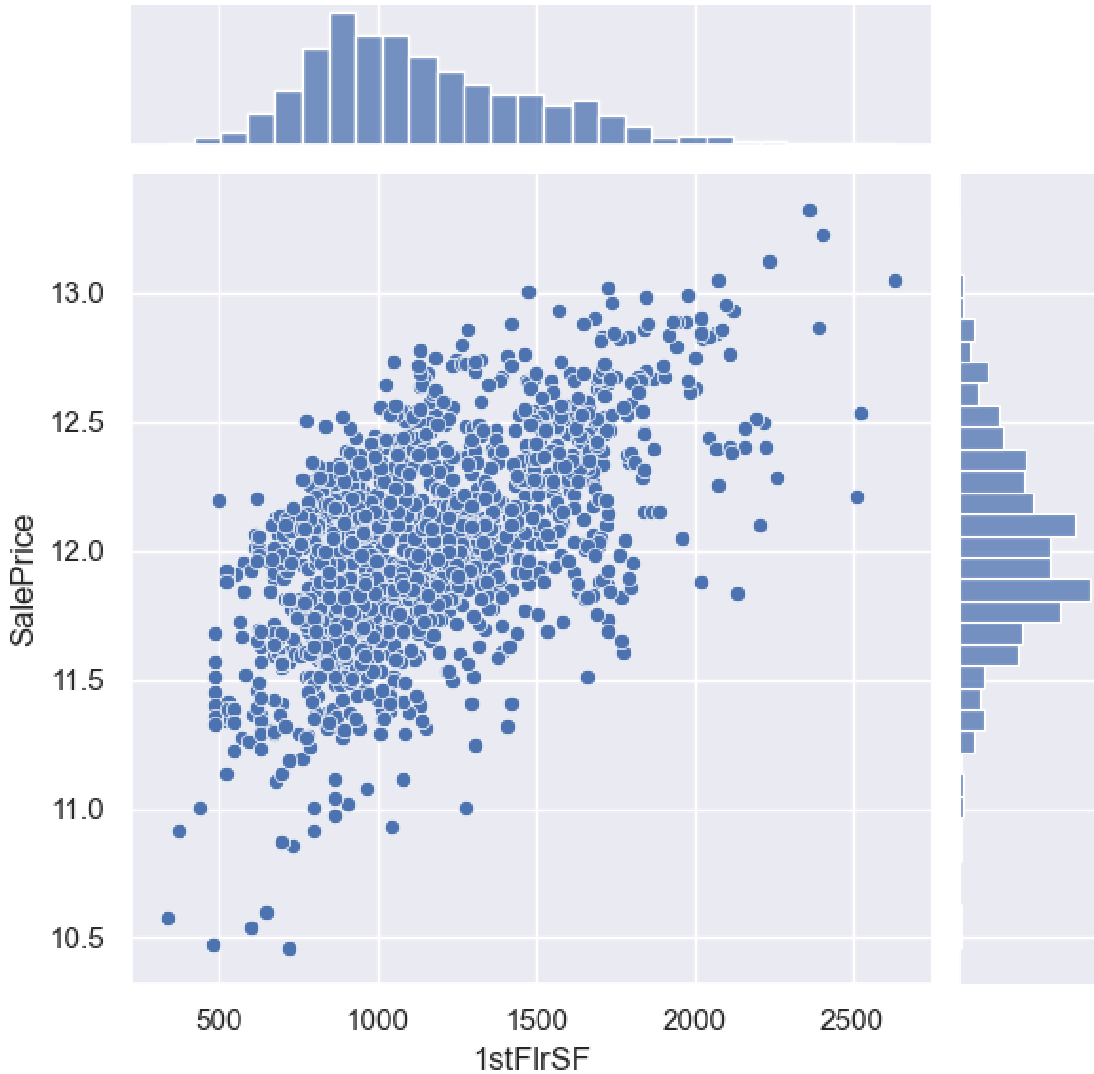


[43]:
```
# LotArea vs SalePrice
sns.jointplot(x = housing_df['LotArea'], y = housing_df['SalePrice'])
plt.show()
```

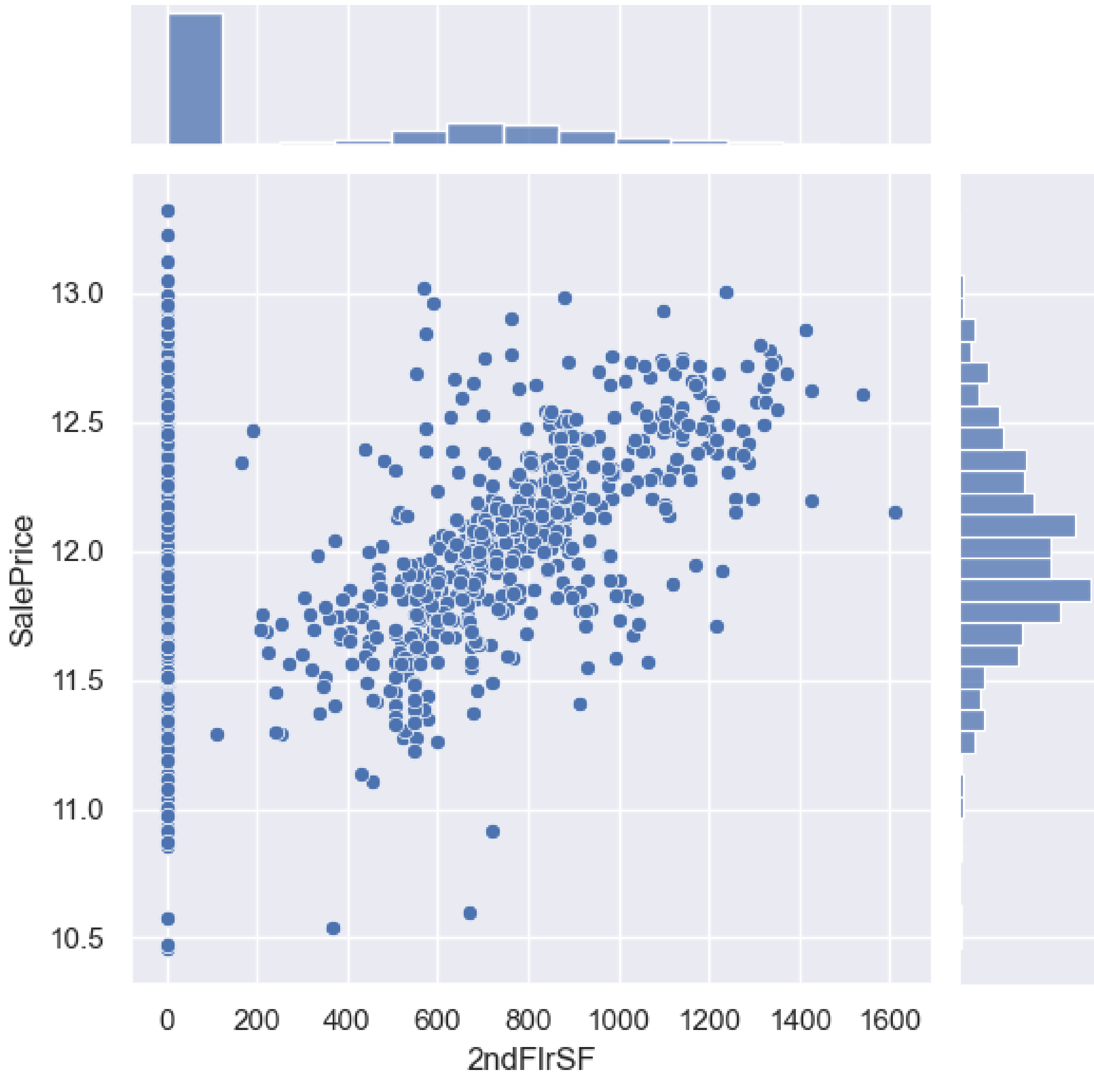


[44]:

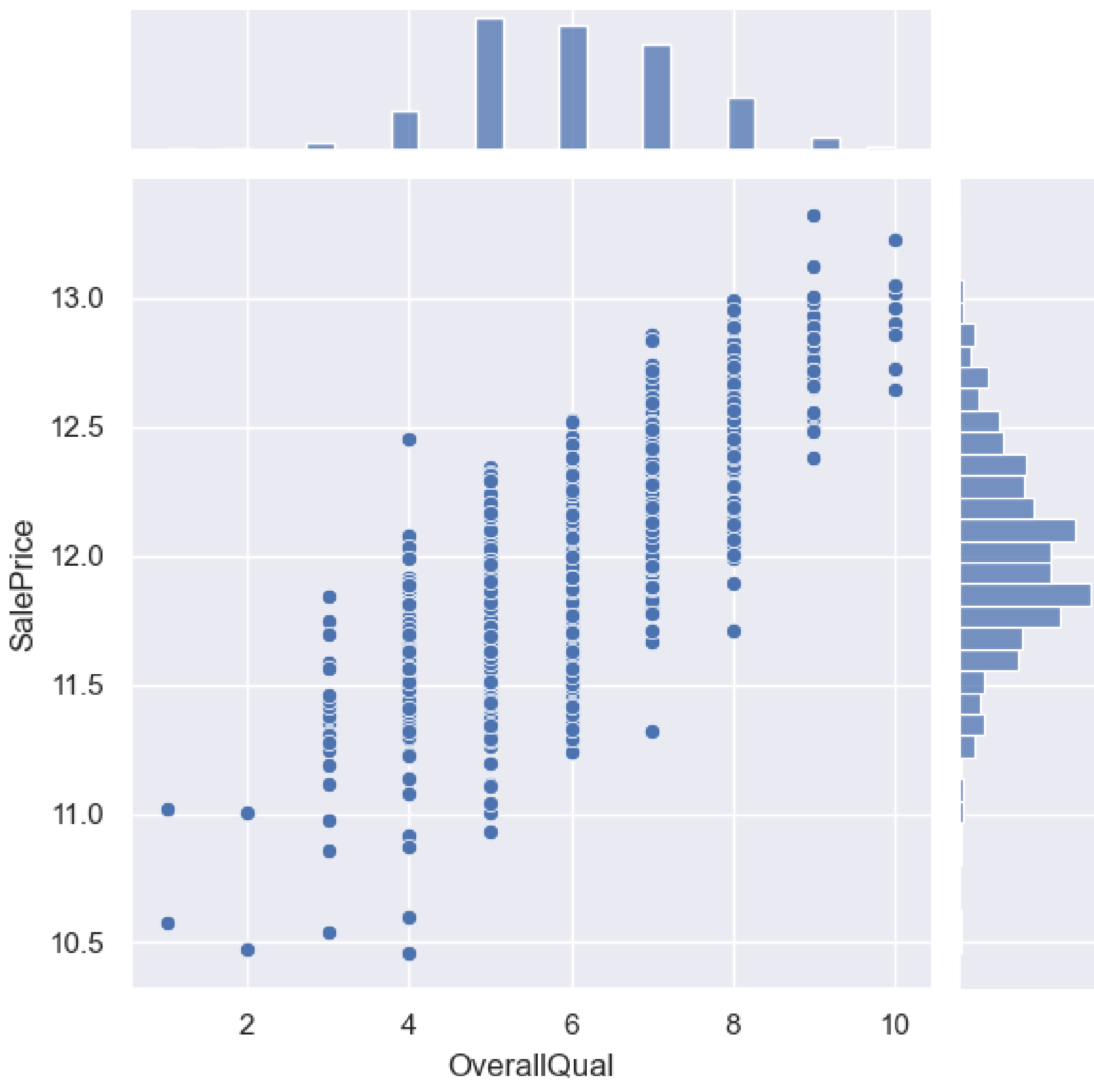
```
# 1stFlrSF vs SalePrice
sns.jointplot(x = housing_df['1stFlrSF'], y = housing_df['SalePrice'])
plt.show()
```



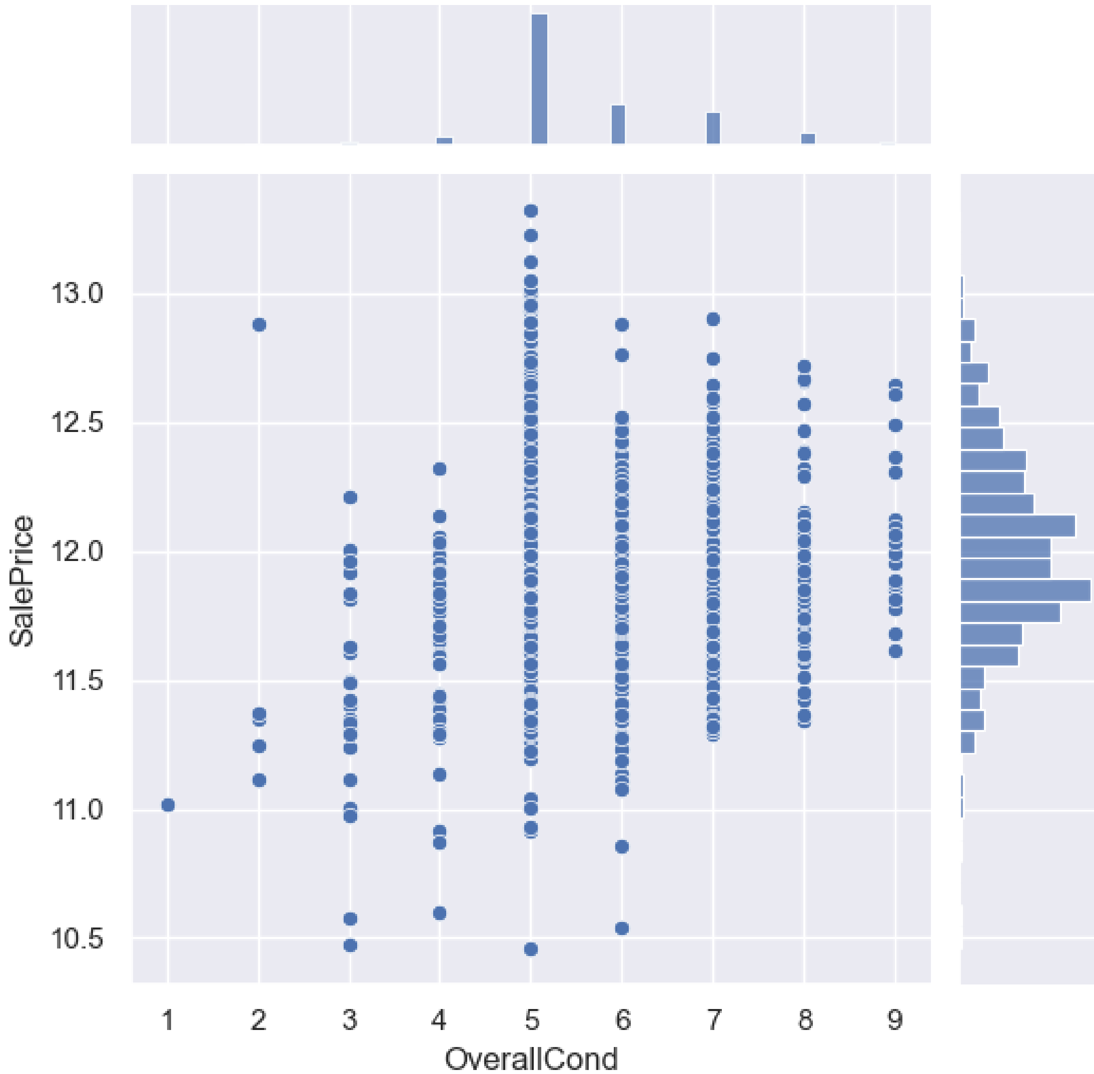
[45]:
```
# 2ndFlrSF vs SalePrice
sns.jointplot(x = housing_df['2ndFlrSF'], y = housing_df['SalePrice'])
plt.show()
```



[46]:
```
# OverallQual vs SalePrice
sns.jointplot(x = housing_df['OverallQual'], y = housing_df['SalePrice'])
plt.show()
```



[47]:
```
# OverallCond vs SalePrice
sns.jointplot(x=housing_df['OverallCond'], y = housing_df['SalePrice'])
plt.show()
```



Ground or First level houses i.e. '0' second floor Sq.Ft has also a steady increase

1.2.3 We can derive a column for 'Age of the property' when it was sold: Name it as 'PropAge'

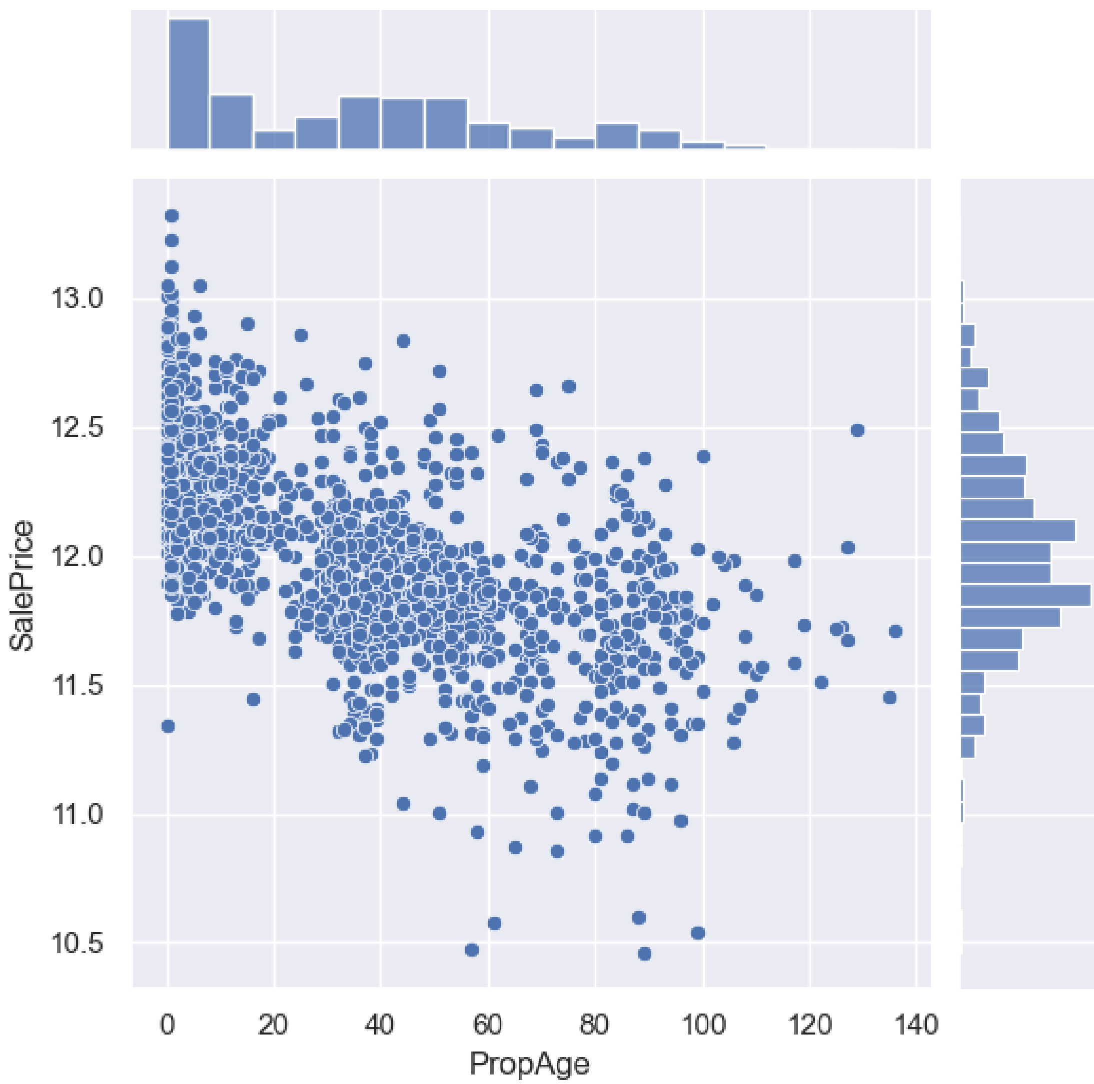
[48]:
```
# PropAge -  Property Age from yearsold - yearbuilt
housing_df['PropAge'] = (housing_df['YrSold'] - housing_df['YearBuilt'])
housing_df.head()
```

[48]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave  none      Reg
1   2          20       RL         80.0     9600   Pave  none      Reg
2   3          60       RL         68.0    11250   Pave  none      IR1
```

```
3   4           70        RL          60.0      9550    Pave   none       IR1
4   5           60        RL          84.0     14260    Pave   none       IR1

  LandContour Utilities  … PoolQC Fence MiscFeature MiscVal MoSold YrSold  \
0         Lvl    AllPub  …   none  none        none       0      2   2008
1         Lvl    AllPub  …   none  none        none       0      5   2007
2         Lvl    AllPub  …   none  none        none       0      9   2008
3         Lvl    AllPub  …   none  none        none       0      2   2006
4         Lvl    AllPub  …   none  none        none       0     12   2008

  SaleType  SaleCondition  SalePrice  PropAge
0       WD         Normal  12.247699        5
1       WD         Normal  12.109016       31
2       WD         Normal  12.317171        7
3       WD        Abnorml  11.849405       91
4       WD         Normal  12.429220        8

[5 rows x 78 columns]
```

[49]:

```python
# PropAge vs SalePrice
sns.jointplot(x = housing_df['PropAge'], y = housing_df['SalePrice'])
plt.show()
```



Increase in Property Age shows a decreasing `SalePrice` trend i.e newer the property, high is the value

1.2.4 Now we can drop the column Month sold and Year Sold, Year built and Year remodelled since it will not be required further

[50]:
```
housing_df = housing_df.drop(['MoSold'], axis = 1)
housing_df = housing_df.drop(['YrSold'], axis = 1)
housing_df = housing_df.drop(['YearBuilt'], axis = 1)
housing_df = housing_df.drop(['YearRemodAdd'], axis = 1)
housing_df.head()
```

[50]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave  none      Reg
1   2          20       RL         80.0     9600   Pave  none      Reg
2   3          60       RL         68.0    11250   Pave  none      IR1
3   4          70       RL         60.0     9550   Pave  none      IR1
4   5          60       RL         84.0    14260   Pave  none      IR1

  LandContour Utilities  … ScreenPorch PoolArea PoolQC Fence MiscFeature  \
0         Lvl    AllPub  …           0        0   none  none        none
1         Lvl    AllPub  …           0        0   none  none        none
2         Lvl    AllPub  …           0        0   none  none        none
3         Lvl    AllPub  …           0        0   none  none        none
4         Lvl    AllPub  …           0        0   none  none        none

  MiscVal SaleType  SaleCondition  SalePrice PropAge
0       0       WD         Normal  12.247699       5
1       0       WD         Normal  12.109016      31
2       0       WD         Normal  12.317171       7
3       0       WD        Abnorml  11.849405      91
4       0       WD         Normal  12.429220       8

[5 rows x 74 columns]
```

[51]:
```
housing_df.Street.value_counts()
```

[51]:
```
Pave    1423
Grvl       6
Name: Street, dtype: int64
```

[52]:
```
housing_df.Utilities.value_counts()
```

[52]:
```
AllPub    1428
NoSeWa       1
Name: Utilities, dtype: int64
```

[53]:
```
# We can also drop columns that show very low variance and thus not required␣
 ↪for predictions
housing_df = housing_df.drop(['Street'], axis = 1)
housing_df = housing_df.drop(['Utilities'], axis = 1)
```

1.2.5 Just to check the variance of these columns

[54]:
```
# l1 = ['Condition2', 'Heating', 'PoolQC', 'RoofMatl', 'BsmtCond',␣
 ↪'GarageQual', 'GarageCond', 'MiscVal', '3SsnPorch', 'FireplaceQu',␣
 ↪'BsmtHalfBath', 'BsmtFinSF2', 'Alley', 'MiscFeature', 'Fence', 'Functional']
l2= housing_df.select_dtypes(include=['float64', 'int64'])
l2
```

[54]:
```
        Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  \
0        1          60         65.0     8450            7            5
1        2          20         80.0     9600            6            8
2        3          60         68.0    11250            7            5
3        4          70         60.0     9550            7            5
4        5          60         84.0    14260            8            5
…      …           …            …        …            …            …
1455  1456          60         62.0     7917            6            5
1456  1457          20         85.0    13175            6            6
1457  1458          70         66.0     9042            7            9
1458  1459          20         68.0     9717            5            6
1459  1460          20         75.0     9937            5            6

      MasVnrArea  BsmtFinSF1  BsmtFinSF2  TotalBsmtSF  …  GarageArea  \
0          196.0         706           0          856  …         548
1            0.0         978           0         1262  …         460
2          162.0         486           0          920  …         608
3            0.0         216           0          756  …         642
4          350.0         655           0         1145  …         836
…            …           …           …           …  …          …
1455         0.0           0           0          953  …         460
1456       119.0         790         163         1542  …         500
1457         0.0         275           0         1152  …         252
1458         0.0          49        1029         1078  …         240
1459         0.0         830         290         1256  …         276

      WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  \
0              0           61              0          0            0
1            298            0              0          0            0
2              0           42              0          0            0
3              0           35            272          0            0
4            192           84              0          0            0
…            …            …              …          …          …
1455           0           40              0          0            0
1456         349            0              0          0            0
1457           0           60              0          0            0
1458         366            0            112          0            0
1459         736           68              0          0            0

      PoolArea  MiscVal  SalePrice  PropAge
0            0        0  12.247699        5
1            0        0  12.109016       31
2            0        0  12.317171        7
3            0        0  11.849405       91
4            0        0  12.429220        8
…          …        …        …       …
1455         0        0  12.072547        8
```

```
1456          0         0  12.254868       32
1457          0      2500  12.493133       69
1458          0         0  11.864469       60
1459          0         0  11.901590       43

[1429 rows x 31 columns]
```

[55]:

```python
for i in l2:
    print(housing_df[i].value_counts())
```

```
1       1
956     1
977     1
976     1
975     1
       ..
482     1
481     1
480     1
479     1
1460    1
Name: Id, Length: 1429, dtype: int64
20     534
60     283
50     139
120     87
30      69
160     63
70      59
80      57
90      52
190     29
85      20
45      12
75      11
180     10
40       4
Name: MSSubClass, dtype: int64
60.0     150
80.0     110
70.0      94
65.0      74
73.0      70
        …
33.0       1
150.0      1
38.0       1
111.0      1
```

```
46.0       1
Name: LotFrontage, Length: 112, dtype: int64
7200     25
9600     24
6000     17
8400     14
9000     14
         ..
10637     1
16033     1
11846     1
2500      1
9717      1
Name: LotArea, Length: 1047, dtype: int64
5     396
6     371
7     314
8     157
4     116
9      40
3      20
10     10
2       3
1       2
Name: OverallQual, dtype: int64
5    804
6    248
7    201
8     72
4     56
3     24
9     18
2      5
1      1
Name: OverallCond, dtype: int64
0.0      856
72.0       8
108.0      8
180.0      8
120.0      7
        …
435.0      1
378.0      1
562.0      1
333.0      1
119.0      1
Name: MasVnrArea, Length: 315, dtype: int64
0      460
```

```
24       12
16        9
662       5
20        5
       …
897       1
299       1
1261      1
994       1
830       1
Name: BsmtFinSF1, Length: 622, dtype: int64
0       1266
180        5
374        3
290        2
64         2
        …
532        1
165        1
1120       1
311        1
1029       1
Name: BsmtFinSF2, Length: 141, dtype: int64
0       37
864     35
672     17
912     15
1040    14
        ..
1581     1
707      1
611      1
2035     1
1542     1
Name: TotalBsmtSF, Length: 704, dtype: int64
864     25
1040    16
912     14
848     12
894     12
        ..
751      1
1509     1
2515     1
605      1
1256     1
Name: 1stFlrSF, Length: 734, dtype: int64
0       827
```

```
728      10
504       9
672       8
546       8
       …
1000      1
687       1
910       1
811       1
1152      1
Name: 2ndFlrSF, Length: 393, dtype: int64
0      1408
80        3
360       2
156       1
205       1
514       1
120       1
481       1
232       1
53        1
473       1
420       1
390       1
371       1
144       1
528       1
234       1
513       1
384       1
Name: LowQualFinSF, dtype: int64
864     22
1040    14
894     11
1456    10
848     10
        ..
2296     1
1199     1
1586     1
1473     1
1256     1
Name: GrLivArea, Length: 831, dtype: int64
0    840
1    574
2     14
3      1
Name: BsmtFullBath, dtype: int64
```

```
0    1349
1      78
2       2
Name: BsmtHalfBath, dtype: int64
2    754
1    650
3     16
0      9
Name: FullBath, dtype: int64
0    904
1    513
2     12
Name: HalfBath, dtype: int64
3    798
2    357
4    195
1     50
5     16
6      7
0      6
Name: BedroomAbvGr, dtype: int64
1    1362
2      64
3       2
0       1
Name: KitchenAbvGr, dtype: int64
0    689
1    631
2    105
3      4
Name: Fireplaces, dtype: int64
0      79
440    49
576    47
240    38
484    34
       ..
435     1
320     1
831     1
325     1
192     1
Name: GarageArea, Length: 428, dtype: int64
0      752
192     37
100     36
144     32
120     31
```

```
       …
42       1
35       1
326      1
382      1
736      1
Name: WoodDeckSF, Length: 266, dtype: int64
0      649
36      28
48      22
20      21
40      19
       …
85       1
187      1
123      1
134      1
236      1
Name: OpenPorchSF, Length: 195, dtype: int64
0      1226
112      15
96        6
120       5
216       5
        …
148       1
272       1
210       1
248       1
99        1
Name: EnclosedPorch, Length: 116, dtype: int64
0      1405
168       3
144       2
180       2
216       2
290       1
153       1
96        1
23        1
162       1
182       1
196       1
320       1
245       1
238       1
508       1
140       1
```

```
130       1
407       1
304       1
Name: 3SsnPorch, dtype: int64
0      1318
192       5
224       5
120       5
189       4
       …
122       1
95        1
260       1
385       1
40        1
Name: ScreenPorch, Length: 72, dtype: int64
0      1426
648       1
576       1
738       1
Name: PoolArea, dtype: int64
0        1379
400        11
500         8
700         5
450         4
600         4
2000        3
1200        2
480         2
15500       1
800         1
350         1
3500        1
1300        1
54          1
620         1
560         1
1400        1
8300        1
2500        1
Name: MiscVal, dtype: int64
11.849405    20
11.813037    17
11.951187    14
11.884496    14
11.608245    13
             ..
```

```
12.219315     1
12.013101     1
12.246739     1
12.574185     1
11.901590     1
Name: SalePrice, Length: 646, dtype: int64
1      96
0      61
4      41
2      39
3      36
       ..
111     1
129     1
102     1
126     1
125     1
Name: PropAge, Length: 119, dtype: int64
```

[56]:
```
housing_df = housing_df.drop(['PoolQC','MiscVal', 'Alley', 'RoofMatl',␣
↪'Condition2', 'Heating', 'GarageCond', 'Fence', 'Functional' ], axis = 1)
```

These Columns are having high `null` values, some of which were imputed. After imputing, it was found that there was very little variance in the data. So we have decided to drop these columns.

[57]:
```
housing_df.shape
```

[57]:
```
(1429, 63)
```

1.3 Data Preparation

- Let's now prepare the data and build the model.

[58]:
```
# Drop 'Id' from Dataframe

housing_df = housing_df.drop(['Id'], axis=1)
housing_df.head()
```

[58]:
```
   MSSubClass MSZoning  LotFrontage  LotArea LotShape LandContour LotConfig  \
0          60       RL         65.0     8450      Reg         Lvl    Inside
1          20       RL         80.0     9600      Reg         Lvl       FR2
2          60       RL         68.0    11250      IR1         Lvl    Inside
3          70       RL         60.0     9550      IR1         Lvl    Corner
4          60       RL         84.0    14260      IR1         Lvl       FR2

  LandSlope Neighborhood Condition1  … OpenPorchSF EnclosedPorch  3SsnPorch  \
0       Gtl      CollgCr       Norm  …          61             0          0
1       Gtl      Veenker      Feedr  …           0             0          0
```

```
2       Gtl      CollgCr        Norm  …          42            0          0
3       Gtl      Crawfor        Norm  …          35          272          0
4       Gtl      NoRidge        Norm  …          84            0          0

   ScreenPorch PoolArea MiscFeature SaleType SaleCondition  SalePrice PropAge
0            0        0        none       WD        Normal  12.247699       5
1            0        0        none       WD        Normal  12.109016      31
2            0        0        none       WD        Normal  12.317171       7
3            0        0        none       WD       Abnorml  11.849405      91
4            0        0        none       WD        Normal  12.429220       8

[5 rows x 62 columns]
```

[59]:
```
#type of each feature in data: int, float, object
types = housing_df.dtypes
#numerical values are either type int or float
numeric_type = types[(types == 'int64') | (types == float)]
#categorical values are type object
categorical_type = types[types == object]
```

[60]:
```
pd.DataFrame(types).reset_index().set_index(0).reset_index()[0].value_counts()
```

[60]:
```
object     33
int64      26
float64     3
Name: 0, dtype: int64
```

[61]:
```
#we should convert numeric_type to a list to make it easier to work
numerical_columns = list(numeric_type.index)
print(numerical_columns)
```

```
['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'Fireplaces', 'GarageArea',
'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch',
'PoolArea', 'SalePrice', 'PropAge']
```

[62]:
```
#Categorical columns
categorical_columns = list(categorical_type.index)
print(categorical_columns)
```

```
['MSZoning', 'LotShape', 'LandContour', 'LotConfig', 'LandSlope',
'Neighborhood', 'Condition1', 'BldgType', 'HouseStyle', 'RoofStyle',
'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond',
'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
'BsmtFinType2', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'PavedDrive',
```

```
'MiscFeature', 'SaleType', 'SaleCondition']
```

1.3.1 Creating Dummy columns to convert categorical into numerical

[63]:
```python
housing_df = pd.get_dummies(housing_df, drop_first=True )
housing_df.head()
```

[63]:
```
   MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  MasVnrArea  \
0          60         65.0     8450            7            5       196.0
1          20         80.0     9600            6            8         0.0
2          60         68.0    11250            7            5       162.0
3          70         60.0     9550            7            5         0.0
4          60         84.0    14260            8            5       350.0

   BsmtFinSF1  BsmtFinSF2  TotalBsmtSF  1stFlrSF  …  SaleType_ConLI  \
0         706           0          856       856  …               0
1         978           0         1262      1262  …               0
2         486           0          920       920  …               0
3         216           0          756       961  …               0
4         655           0         1145      1145  …               0

   SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
0               0             0             0            1
1               0             0             0            1
2               0             0             0            1
3               0             0             0            1
4               0             0             0            1

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                      0                     0                     0
1                      0                     0                     0
2                      0                     0                     0
3                      0                     0                     0
4                      0                     0                     0

   SaleCondition_Normal  SaleCondition_Partial
0                     1                      0
1                     1                      0
2                     1                      0
3                     0                      0
4                     1                      0

[5 rows x 211 columns]
```

[64]:
```python
X = housing_df.drop(['SalePrice'], axis=1)
X.head()
```

[64]:
```
   MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  MasVnrArea  \
0          60         65.0     8450            7            5       196.0
1          20         80.0     9600            6            8         0.0
2          60         68.0    11250            7            5       162.0
3          70         60.0     9550            7            5         0.0
4          60         84.0    14260            8            5       350.0

   BsmtFinSF1  BsmtFinSF2  TotalBsmtSF  1stFlrSF  …  SaleType_ConLI  \
0         706           0          856       856  …               0
1         978           0         1262      1262  …               0
2         486           0          920       920  …               0
3         216           0          756       961  …               0
4         655           0         1145      1145  …               0

   SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
0               0             0             0            1
1               0             0             0            1
2               0             0             0            1
3               0             0             0            1
4               0             0             0            1

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                      0                     0                     0
1                      0                     0                     0
2                      0                     0                     0
3                      0                     0                     0
4                      0                     0                     0

   SaleCondition_Normal  SaleCondition_Partial
0                     1                      0
1                     1                      0
2                     1                      0
3                     0                      0
4                     1                      0

[5 rows x 210 columns]
```

[65]:
```python
# Putting response variable to y
y = housing_df['SalePrice']
y.head()
```

[65]:
```
0    12.247699
1    12.109016
2    12.317171
3    11.849405
4    12.429220
Name: SalePrice, dtype: float64
```

1.4 Train Test Split

[66]:
```
# Splitting the data into train and test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,␣
↪test_size=0.3, random_state=50)
```

1.4.1 Standardized the dataset

[67]:
```
scaler = StandardScaler()

X_train[['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',␣
↪'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF', '1stFlrSF',␣
↪'2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',␣
↪'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'Fireplaces',␣
↪'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',␣
↪'ScreenPorch', 'PoolArea', 'PropAge']] = scaler.
↪fit_transform(X_train[['MSSubClass', 'LotFrontage', 'LotArea',␣
↪'OverallQual', 'OverallCond', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',␣
↪'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',␣
↪'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',␣
↪'KitchenAbvGr', 'Fireplaces', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',␣
↪'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PropAge']])

X_test[['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',␣
↪'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF', '1stFlrSF',␣
↪'2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',␣
↪'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'Fireplaces',␣
↪'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',␣
↪'ScreenPorch', 'PoolArea', 'PropAge']] = scaler.
↪fit_transform(X_test[['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',␣
↪'OverallCond', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF',␣
↪'1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',␣
↪'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',␣
↪'Fireplaces', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch',␣
↪'3SsnPorch', 'ScreenPorch', 'PoolArea', 'PropAge']])
```

[68]:
```
X_train.head()
```

[68]:
```
      MSSubClass  LotFrontage   LotArea  OverallQual  OverallCond  MasVnrArea  \
11      0.085645     0.746261  0.154684     2.241710    -0.513939    1.145212
1070   -0.869945     0.130905 -0.020017    -0.764271    -0.513939    0.152993
513    -0.869945     0.083570 -0.115156    -0.012775    -0.513939   -0.564274
467     0.324542     0.462250 -0.086269    -0.764271     1.258264    0.774624
993     0.085645    -0.058435 -0.148775    -0.012775    -0.513939   -0.564274

      BsmtFinSF1  BsmtFinSF2  TotalBsmtSF  1stFlrSF  …  SaleType_ConLI  \
```

```
11     1.323938  -0.300737    0.345478  0.113320  ...            0
1070   0.360916  -0.300737    0.030191 -0.267693  ...            0
513   -0.223442  -0.300737    0.119563 -0.176705  ...            0
467   -0.106571  -0.300737   -0.764234 -0.722635  ...            0
993   -1.008820  -0.300737   -0.709617 -1.115022  ...            0

      SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
11                 0             1             0            0
1070               0             0             0            1
513                0             0             0            1
467                0             0             0            1
993                0             1             0            0

      SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
11                        0                     0                     0
1070                      0                     0                     0
513                       0                     0                     0
467                       0                     0                     0
993                       0                     0                     0

      SaleCondition_Normal  SaleCondition_Partial
11                       0                      1
1070                     1                      0
513                      1                      0
467                      1                      0
993                      0                      1

[5 rows x 210 columns]
```

[69]:
```
X_test.head()
```

[69]:
```
      MSSubClass  LotFrontage    LotArea  OverallQual  OverallCond  \
461     0.282163    -0.495702  -0.339155     0.620334     3.325664
335     3.002345     0.729276  17.009026    -0.825989     0.435637
200    -0.851246     0.484280  -0.190859    -1.549150    -0.527705
214     0.055482    -0.005711   0.068493    -0.102827     1.398980
1003    0.735527     0.484280   0.134598    -0.825989     0.435637

      MasVnrArea  BsmtFinSF1  BsmtFinSF2  TotalBsmtSF  1stFlrSF  ...  \
461    -0.606456   -0.211663    0.997135    -1.205047 -1.556925  ...
335    -0.606456    1.881258    0.620602     1.075768  1.258064  ...
200    -0.606456   -1.026482   -0.257974     0.157612 -0.084718  ...
214     0.272709   -0.146477   -0.257974    -0.891708 -1.249541  ...
1003    0.335917   -1.026482   -0.257974     1.515413  1.422541  ...

      SaleType_ConLI  SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
461                0               0             0             0            1
```

```
335                  0                0              0              0             1
200                  0                0              0              0             1
214                  0                0              0              0             1
1003                 0                0              0              0             1

      SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
461                       0                     0                     0
335                       0                     0                     0
200                       0                     0                     0
214                       0                     0                     0
1003                      0                     0                     0

      SaleCondition_Normal  SaleCondition_Partial
461                      1                      0
335                      1                      0
200                      1                      0
214                      1                      0
1003                     1                      0

[5 rows x 210 columns]
```

1.5 Model Building and Evaluation

1.5.1 Lets first check the model using Linear Regression and RFE (OPTIONAL)

[70]:
```
# Running RFE
# Since there are more than 250 variables for analysis, we will run RFE to
 ↪select some that have high predictive power
lm = LinearRegression()
lm.fit(X_train, y_train)
 # running RFE for top 100 variables
rfe = RFE(lm, step= 100)
rfe = rfe.fit(X_train, y_train)
```

[71]:
```
# Check the ranks
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

[71]:
```
[('MSSubClass', False, 3),
 ('LotFrontage', False, 3),
 ('LotArea', False, 3),
 ('OverallQual', True, 1),
 ('OverallCond', True, 1),
 ('MasVnrArea', False, 3),
 ('BsmtFinSF1', False, 3),
 ('BsmtFinSF2', False, 3),
 ('TotalBsmtSF', True, 1),
 ('1stFlrSF', True, 1),
```

```
('2ndFlrSF', True, 1),
('LowQualFinSF', True, 1),
('GrLivArea', True, 1),
('BsmtFullBath', False, 3),
('BsmtHalfBath', False, 3),
('FullBath', False, 3),
('HalfBath', False, 3),
('BedroomAbvGr', False, 3),
('KitchenAbvGr', False, 3),
('Fireplaces', False, 3),
('GarageArea', False, 3),
('WoodDeckSF', False, 3),
('OpenPorchSF', False, 3),
('EnclosedPorch', False, 3),
('3SsnPorch', False, 3),
('ScreenPorch', False, 3),
('PoolArea', False, 3),
('PropAge', True, 1),
('MSZoning_FV', True, 1),
('MSZoning_RH', True, 1),
('MSZoning_RL', True, 1),
('MSZoning_RM', True, 1),
('LotShape_IR2', False, 3),
('LotShape_IR3', False, 3),
('LotShape_Reg', False, 3),
('LandContour_HLS', False, 3),
('LandContour_Low', False, 3),
('LandContour_Lvl', False, 3),
('LotConfig_CulDSac', False, 3),
('LotConfig_FR2', False, 3),
('LotConfig_FR3', True, 1),
('LotConfig_Inside', False, 3),
('LandSlope_Mod', False, 3),
('LandSlope_Sev', True, 1),
('Neighborhood_Blueste', True, 1),
('Neighborhood_BrDale', True, 1),
('Neighborhood_BrkSide', False, 3),
('Neighborhood_ClearCr', False, 3),
('Neighborhood_CollgCr', False, 3),
('Neighborhood_Crawfor', True, 1),
('Neighborhood_Edwards', True, 1),
('Neighborhood_Gilbert', False, 3),
('Neighborhood_IDOTRR', True, 1),
('Neighborhood_MeadowV', True, 1),
('Neighborhood_Mitchel', True, 1),
('Neighborhood_NAmes', True, 1),
('Neighborhood_NPkVill', False, 3),
```

```
('Neighborhood_NWAmes', True, 1),
('Neighborhood_NoRidge', False, 3),
('Neighborhood_NridgHt', False, 3),
('Neighborhood_OldTown', True, 1),
('Neighborhood_SWISU', True, 1),
('Neighborhood_Sawyer', True, 1),
('Neighborhood_SawyerW', False, 3),
('Neighborhood_Somerst', True, 1),
('Neighborhood_StoneBr', True, 1),
('Neighborhood_Timber', False, 3),
('Neighborhood_Veenker', False, 3),
('Condition1_Feedr', False, 3),
('Condition1_Norm', True, 1),
('Condition1_PosA', True, 1),
('Condition1_PosN', True, 1),
('Condition1_RRAe', True, 1),
('Condition1_RRAn', True, 1),
('Condition1_RRNe', False, 2),
('Condition1_RRNn', True, 1),
('BldgType_2fmCon', False, 3),
('BldgType_Duplex', False, 3),
('BldgType_Twnhs', True, 1),
('BldgType_TwnhsE', False, 3),
('HouseStyle_1.5Unf', False, 3),
('HouseStyle_1Story', False, 3),
('HouseStyle_2.5Fin', True, 1),
('HouseStyle_2.5Unf', False, 3),
('HouseStyle_2Story', False, 3),
('HouseStyle_SFoyer', False, 3),
('HouseStyle_SLvl', False, 3),
('RoofStyle_Gable', True, 1),
('RoofStyle_Gambrel', True, 1),
('RoofStyle_Hip', True, 1),
('RoofStyle_Mansard', False, 3),
('RoofStyle_Shed', False, 3),
('Exterior1st_AsphShn', True, 1),
('Exterior1st_BrkComm', True, 1),
('Exterior1st_BrkFace', True, 1),
('Exterior1st_CBlock', True, 1),
('Exterior1st_CemntBd', True, 1),
('Exterior1st_HdBoard', True, 1),
('Exterior1st_ImStucc', False, 3),
('Exterior1st_MetalSd', False, 3),
('Exterior1st_Plywood', True, 1),
('Exterior1st_Stone', False, 3),
('Exterior1st_Stucco', False, 3),
('Exterior1st_VinylSd', True, 1),
```

```
('Exterior1st_Wd Sdng', True, 1),
('Exterior1st_WdShing', True, 1),
('Exterior2nd_AsphShn', False, 2),
('Exterior2nd_Brk Cmn', False, 3),
('Exterior2nd_BrkFace', False, 3),
('Exterior2nd_CBlock', True, 1),
('Exterior2nd_CmentBd', True, 1),
('Exterior2nd_HdBoard', True, 1),
('Exterior2nd_ImStucc', False, 3),
('Exterior2nd_MetalSd', False, 3),
('Exterior2nd_Other', True, 1),
('Exterior2nd_Plywood', True, 1),
('Exterior2nd_Stone', True, 1),
('Exterior2nd_Stucco', False, 3),
('Exterior2nd_VinylSd', True, 1),
('Exterior2nd_Wd Sdng', True, 1),
('Exterior2nd_Wd Shng', False, 3),
('MasVnrType_BrkFace', True, 1),
('MasVnrType_None', True, 1),
('MasVnrType_Stone', True, 1),
('MasVnrType_none', False, 3),
('ExterQual_Fa', True, 1),
('ExterQual_Gd', False, 3),
('ExterQual_TA', True, 1),
('ExterCond_Fa', True, 1),
('ExterCond_Gd', True, 1),
('ExterCond_Po', True, 1),
('ExterCond_TA', True, 1),
('Foundation_CBlock', False, 3),
('Foundation_PConc', True, 1),
('Foundation_Slab', False, 3),
('Foundation_Stone', True, 1),
('Foundation_Wood', True, 1),
('BsmtQual_Fa', False, 3),
('BsmtQual_Gd', False, 3),
('BsmtQual_TA', False, 3),
('BsmtQual_none', True, 1),
('BsmtCond_Gd', False, 3),
('BsmtCond_Po', True, 1),
('BsmtCond_TA', False, 3),
('BsmtCond_none', True, 1),
('BsmtExposure_Gd', True, 1),
('BsmtExposure_Mn', False, 3),
('BsmtExposure_No', False, 3),
('BsmtExposure_none', True, 1),
('BsmtFinType1_BLQ', False, 3),
('BsmtFinType1_GLQ', False, 3),
```

```
('BsmtFinType1_LwQ', False, 3),
('BsmtFinType1_Rec', False, 3),
('BsmtFinType1_Unf', False, 3),
('BsmtFinType1_none', True, 1),
('BsmtFinType2_BLQ', False, 3),
('BsmtFinType2_GLQ', False, 3),
('BsmtFinType2_LwQ', False, 3),
('BsmtFinType2_Rec', False, 2),
('BsmtFinType2_Unf', False, 3),
('BsmtFinType2_none', True, 1),
('HeatingQC_Fa', False, 3),
('HeatingQC_Gd', False, 3),
('HeatingQC_Po', True, 1),
('HeatingQC_TA', True, 1),
('CentralAir_Y', True, 1),
('Electrical_FuseF', False, 3),
('Electrical_FuseP', False, 2),
('Electrical_Mix', True, 1),
('Electrical_SBrkr', False, 3),
('KitchenQual_Fa', True, 1),
('KitchenQual_Gd', True, 1),
('KitchenQual_TA', True, 1),
('FireplaceQu_Fa', False, 3),
('FireplaceQu_Gd', False, 3),
('FireplaceQu_Po', True, 1),
('FireplaceQu_TA', False, 3),
('FireplaceQu_none', False, 3),
('GarageType_Attchd', True, 1),
('GarageType_Basment', True, 1),
('GarageType_BuiltIn', True, 1),
('GarageType_CarPort', True, 1),
('GarageType_Detchd', True, 1),
('GarageType_none', True, 1),
('GarageFinish_RFn', False, 3),
('GarageFinish_Unf', False, 3),
('GarageFinish_none', True, 1),
('GarageQual_Fa', True, 1),
('GarageQual_Gd', True, 1),
('GarageQual_Po', True, 1),
('GarageQual_TA', True, 1),
('GarageQual_none', True, 1),
('PavedDrive_P', False, 3),
('PavedDrive_Y', False, 3),
('MiscFeature_Othr', True, 1),
('MiscFeature_Shed', True, 1),
('MiscFeature_none', True, 1),
('SaleType_CWD', True, 1),
```

```
 ('SaleType_Con', True, 1),
 ('SaleType_ConLD', True, 1),
 ('SaleType_ConLI', False, 3),
 ('SaleType_ConLw', False, 3),
 ('SaleType_New', True, 1),
 ('SaleType_Oth', False, 3),
 ('SaleType_WD', False, 3),
 ('SaleCondition_AdjLand', False, 2),
 ('SaleCondition_Alloca', False, 3),
 ('SaleCondition_Family', False, 3),
 ('SaleCondition_Normal', True, 1),
 ('SaleCondition_Partial', True, 1)]
```

[72]:
```
# Select the top 100 variables
col = X_train.columns[rfe.support_]
col
```

[72]:
```
Index(['OverallQual', 'OverallCond', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'PropAge', 'MSZoning_FV', 'MSZoning_RH',
       …
       'GarageQual_none', 'MiscFeature_Othr', 'MiscFeature_Shed',
       'MiscFeature_none', 'SaleType_CWD', 'SaleType_Con', 'SaleType_ConLD',
       'SaleType_New', 'SaleCondition_Normal', 'SaleCondition_Partial'],
      dtype='object', length=105)
```

[73]:
```
X_train.columns[~rfe.support_]
```

[73]:
```
Index(['MSSubClass', 'LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       …
       'GarageFinish_Unf', 'PavedDrive_P', 'PavedDrive_Y', 'SaleType_ConLI',
       'SaleType_ConLw', 'SaleType_Oth', 'SaleType_WD',
       'SaleCondition_AdjLand', 'SaleCondition_Alloca',
       'SaleCondition_Family'],
      dtype='object', length=105)
```

[74]:
```
# Creating X_test dataframe with RFE selected variables
X_train_rfe = X_train[col]
```

[75]:
```
X_train_rfe = pd.DataFrame(X_train[col])
```

[76]:
```
X_train_rfe.head()
```

[76]:
```
      OverallQual  OverallCond  TotalBsmtSF  1stFlrSF  2ndFlrSF  LowQualFinSF  \
11       2.241710    -0.513939     0.345478  0.113320  1.995226     -0.111211
1070    -0.764271    -0.513939     0.030191 -0.267693 -0.769750     -0.111211
513     -0.012775    -0.513939     0.119563 -0.176705 -0.769750     -0.111211
467     -0.764271     1.258264    -0.764234 -0.722635  1.060654     -0.111211
```

```
993     -0.012775    -0.513939    -0.709617 -1.115022  1.046127     -0.111211

      GrLivArea   PropAge  MSZoning_FV  MSZoning_RH  …  GarageQual_none  \
11     1.923409 -1.209026            0            0  …                0
1070  -0.932170  0.445587            0            0  …                0
513   -0.860557 -0.447904            0            0  …                0
467    0.401627  0.908879            0            0  …                0
993    0.079368 -1.209026            0            0  …                0

      MiscFeature_Othr  MiscFeature_Shed  MiscFeature_none  SaleType_CWD  \
11                   0                 0                 1             0
1070                 0                 0                 1             0
513                  0                 0                 1             0
467                  0                 0                 1             0
993                  0                 0                 1             0

      SaleType_Con  SaleType_ConLD  SaleType_New  SaleCondition_Normal  \
11               0               0             1                     0
1070             0               0             0                     1
513              0               0             0                     1
467              0               0             0                     1
993              0               0             1                     0

      SaleCondition_Partial
11                        1
1070                      0
513                       0
467                       0
993                       1

[5 rows x 105 columns]
```

[77]:
```python
X_train_rfe.shape
```

[77]: (1000, 105)

[78]:
```python
# predict
y_train_pred = lm.predict(X_train)
metrics.r2_score(y_true=y_train, y_pred=y_train_pred)
```

[78]: 0.939757901286709

[79]:
```python
y_test_pred = lm.predict(X_test)
metrics.r2_score(y_true=y_test, y_pred=y_test_pred)
```

[79]: -1.6000176574253728e+20

1.5.2 Test R2 is too low, we will check for alternate methods of Regression

[80]:
```
# Check the ranks
list(zip(X_test.columns,rfe.support_,rfe.ranking_))
```

[80]:
```
[('MSSubClass', False, 3),
 ('LotFrontage', False, 3),
 ('LotArea', False, 3),
 ('OverallQual', True, 1),
 ('OverallCond', True, 1),
 ('MasVnrArea', False, 3),
 ('BsmtFinSF1', False, 3),
 ('BsmtFinSF2', False, 3),
 ('TotalBsmtSF', True, 1),
 ('1stFlrSF', True, 1),
 ('2ndFlrSF', True, 1),
 ('LowQualFinSF', True, 1),
 ('GrLivArea', True, 1),
 ('BsmtFullBath', False, 3),
 ('BsmtHalfBath', False, 3),
 ('FullBath', False, 3),
 ('HalfBath', False, 3),
 ('BedroomAbvGr', False, 3),
 ('KitchenAbvGr', False, 3),
 ('Fireplaces', False, 3),
 ('GarageArea', False, 3),
 ('WoodDeckSF', False, 3),
 ('OpenPorchSF', False, 3),
 ('EnclosedPorch', False, 3),
 ('3SsnPorch', False, 3),
 ('ScreenPorch', False, 3),
 ('PoolArea', False, 3),
 ('PropAge', True, 1),
 ('MSZoning_FV', True, 1),
 ('MSZoning_RH', True, 1),
 ('MSZoning_RL', True, 1),
 ('MSZoning_RM', True, 1),
 ('LotShape_IR2', False, 3),
 ('LotShape_IR3', False, 3),
 ('LotShape_Reg', False, 3),
 ('LandContour_HLS', False, 3),
 ('LandContour_Low', False, 3),
 ('LandContour_Lvl', False, 3),
 ('LotConfig_CulDSac', False, 3),
 ('LotConfig_FR2', False, 3),
 ('LotConfig_FR3', True, 1),
 ('LotConfig_Inside', False, 3),
```

```
('LandSlope_Mod', False, 3),
('LandSlope_Sev', True, 1),
('Neighborhood_Blueste', True, 1),
('Neighborhood_BrDale', True, 1),
('Neighborhood_BrkSide', False, 3),
('Neighborhood_ClearCr', False, 3),
('Neighborhood_CollgCr', False, 3),
('Neighborhood_Crawfor', True, 1),
('Neighborhood_Edwards', True, 1),
('Neighborhood_Gilbert', False, 3),
('Neighborhood_IDOTRR', True, 1),
('Neighborhood_MeadowV', True, 1),
('Neighborhood_Mitchel', True, 1),
('Neighborhood_NAmes', True, 1),
('Neighborhood_NPkVill', False, 3),
('Neighborhood_NWAmes', True, 1),
('Neighborhood_NoRidge', False, 3),
('Neighborhood_NridgHt', False, 3),
('Neighborhood_OldTown', True, 1),
('Neighborhood_SWISU', True, 1),
('Neighborhood_Sawyer', True, 1),
('Neighborhood_SawyerW', False, 3),
('Neighborhood_Somerst', True, 1),
('Neighborhood_StoneBr', True, 1),
('Neighborhood_Timber', False, 3),
('Neighborhood_Veenker', False, 3),
('Condition1_Feedr', False, 3),
('Condition1_Norm', True, 1),
('Condition1_PosA', True, 1),
('Condition1_PosN', True, 1),
('Condition1_RRAe', True, 1),
('Condition1_RRAn', True, 1),
('Condition1_RRNe', False, 2),
('Condition1_RRNn', True, 1),
('BldgType_2fmCon', False, 3),
('BldgType_Duplex', False, 3),
('BldgType_Twnhs', True, 1),
('BldgType_TwnhsE', False, 3),
('HouseStyle_1.5Unf', False, 3),
('HouseStyle_1Story', False, 3),
('HouseStyle_2.5Fin', True, 1),
('HouseStyle_2.5Unf', False, 3),
('HouseStyle_2Story', False, 3),
('HouseStyle_SFoyer', False, 3),
('HouseStyle_SLvl', False, 3),
('RoofStyle_Gable', True, 1),
('RoofStyle_Gambrel', True, 1),
```

```
('RoofStyle_Hip', True, 1),
('RoofStyle_Mansard', False, 3),
('RoofStyle_Shed', False, 3),
('Exterior1st_AsphShn', True, 1),
('Exterior1st_BrkComm', True, 1),
('Exterior1st_BrkFace', True, 1),
('Exterior1st_CBlock', True, 1),
('Exterior1st_CemntBd', True, 1),
('Exterior1st_HdBoard', True, 1),
('Exterior1st_ImStucc', False, 3),
('Exterior1st_MetalSd', False, 3),
('Exterior1st_Plywood', True, 1),
('Exterior1st_Stone', False, 3),
('Exterior1st_Stucco', False, 3),
('Exterior1st_VinylSd', True, 1),
('Exterior1st_Wd Sdng', True, 1),
('Exterior1st_WdShing', True, 1),
('Exterior2nd_AsphShn', False, 2),
('Exterior2nd_Brk Cmn', False, 3),
('Exterior2nd_BrkFace', False, 3),
('Exterior2nd_CBlock', True, 1),
('Exterior2nd_CmentBd', True, 1),
('Exterior2nd_HdBoard', True, 1),
('Exterior2nd_ImStucc', False, 3),
('Exterior2nd_MetalSd', False, 3),
('Exterior2nd_Other', True, 1),
('Exterior2nd_Plywood', True, 1),
('Exterior2nd_Stone', True, 1),
('Exterior2nd_Stucco', False, 3),
('Exterior2nd_VinylSd', True, 1),
('Exterior2nd_Wd Sdng', True, 1),
('Exterior2nd_Wd Shng', False, 3),
('MasVnrType_BrkFace', True, 1),
('MasVnrType_None', True, 1),
('MasVnrType_Stone', True, 1),
('MasVnrType_none', False, 3),
('ExterQual_Fa', True, 1),
('ExterQual_Gd', False, 3),
('ExterQual_TA', True, 1),
('ExterCond_Fa', True, 1),
('ExterCond_Gd', True, 1),
('ExterCond_Po', True, 1),
('ExterCond_TA', True, 1),
('Foundation_CBlock', False, 3),
('Foundation_PConc', True, 1),
('Foundation_Slab', False, 3),
('Foundation_Stone', True, 1),
```

```
('Foundation_Wood', True, 1),
('BsmtQual_Fa', False, 3),
('BsmtQual_Gd', False, 3),
('BsmtQual_TA', False, 3),
('BsmtQual_none', True, 1),
('BsmtCond_Gd', False, 3),
('BsmtCond_Po', True, 1),
('BsmtCond_TA', False, 3),
('BsmtCond_none', True, 1),
('BsmtExposure_Gd', True, 1),
('BsmtExposure_Mn', False, 3),
('BsmtExposure_No', False, 3),
('BsmtExposure_none', True, 1),
('BsmtFinType1_BLQ', False, 3),
('BsmtFinType1_GLQ', False, 3),
('BsmtFinType1_LwQ', False, 3),
('BsmtFinType1_Rec', False, 3),
('BsmtFinType1_Unf', False, 3),
('BsmtFinType1_none', True, 1),
('BsmtFinType2_BLQ', False, 3),
('BsmtFinType2_GLQ', False, 3),
('BsmtFinType2_LwQ', False, 3),
('BsmtFinType2_Rec', False, 2),
('BsmtFinType2_Unf', False, 3),
('BsmtFinType2_none', True, 1),
('HeatingQC_Fa', False, 3),
('HeatingQC_Gd', False, 3),
('HeatingQC_Po', True, 1),
('HeatingQC_TA', True, 1),
('CentralAir_Y', True, 1),
('Electrical_FuseF', False, 3),
('Electrical_FuseP', False, 2),
('Electrical_Mix', True, 1),
('Electrical_SBrkr', False, 3),
('KitchenQual_Fa', True, 1),
('KitchenQual_Gd', True, 1),
('KitchenQual_TA', True, 1),
('FireplaceQu_Fa', False, 3),
('FireplaceQu_Gd', False, 3),
('FireplaceQu_Po', True, 1),
('FireplaceQu_TA', False, 3),
('FireplaceQu_none', False, 3),
('GarageType_Attchd', True, 1),
('GarageType_Basment', True, 1),
('GarageType_BuiltIn', True, 1),
('GarageType_CarPort', True, 1),
('GarageType_Detchd', True, 1),
```

```
('GarageType_none', True, 1),
('GarageFinish_RFn', False, 3),
('GarageFinish_Unf', False, 3),
('GarageFinish_none', True, 1),
('GarageQual_Fa', True, 1),
('GarageQual_Gd', True, 1),
('GarageQual_Po', True, 1),
('GarageQual_TA', True, 1),
('GarageQual_none', True, 1),
('PavedDrive_P', False, 3),
('PavedDrive_Y', False, 3),
('MiscFeature_Othr', True, 1),
('MiscFeature_Shed', True, 1),
('MiscFeature_none', True, 1),
('SaleType_CWD', True, 1),
('SaleType_Con', True, 1),
('SaleType_ConLD', True, 1),
('SaleType_ConLI', False, 3),
('SaleType_ConLw', False, 3),
('SaleType_New', True, 1),
('SaleType_Oth', False, 3),
('SaleType_WD', False, 3),
('SaleCondition_AdjLand', False, 2),
('SaleCondition_Alloca', False, 3),
('SaleCondition_Family', False, 3),
('SaleCondition_Normal', True, 1),
('SaleCondition_Partial', True, 1)]
```

[81]:
```
# Select the top 100 variables

col1 = X_test.columns[rfe.support_]
col1
```

[81]:
```
Index(['OverallQual', 'OverallCond', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'PropAge', 'MSZoning_FV', 'MSZoning_RH',
       …
       'GarageQual_none', 'MiscFeature_Othr', 'MiscFeature_Shed',
       'MiscFeature_none', 'SaleType_CWD', 'SaleType_Con', 'SaleType_ConLD',
       'SaleType_New', 'SaleCondition_Normal', 'SaleCondition_Partial'],
      dtype='object', length=105)
```

[82]:
```
X_test_rfe = X_test[col1]
```

[83]:
```
X_test_rfe.head()
```

[83]:
```
     OverallQual  OverallCond  TotalBsmtSF  1stFlrSF  2ndFlrSF  LowQualFinSF  \
461     0.620334     3.325664    -1.205047 -1.556925  0.506037      -0.09698
```

```
335     -0.825989     0.435637     1.075768  1.258064 -0.435654      -0.09698
200     -1.549150    -0.527705     0.157612 -0.084718 -0.835813      -0.09698
214     -0.102827     1.398980    -0.891708 -1.249541  0.848688      -0.09698
1003    -0.825989     0.435637     1.515413  1.422541 -0.835813      -0.09698

      GrLivArea   PropAge  MSZoning_FV  MSZoning_RH  …  GarageQual_none  \
461   -0.813247  1.326790            0            0  …                0
335    0.616745  0.303675            0            0  …                0
200   -0.843999 -0.924063            0            0  …                0
214   -0.248718 -0.037363            0            0  …                0
1003   0.383905 -0.105571            0            0  …                0

      MiscFeature_Othr  MiscFeature_Shed  MiscFeature_none  SaleType_CWD  \
461                  0                 0                 1             0
335                  0                 1                 0             0
200                  0                 0                 1             0
214                  0                 1                 0             0
1003                 0                 0                 1             0

      SaleType_Con  SaleType_ConLD  SaleType_New  SaleCondition_Normal  \
461              0               0             0                     1
335              0               0             0                     1
200              0               0             0                     1
214              0               0             0                     1
1003             0               0             0                     1

      SaleCondition_Partial
461                       0
335                       0
200                       0
214                       0
1003                      0

[5 rows x 105 columns]
```

1.5.3 Ridge & Lasso Regression

Let's now try predicting house prices and perform `Ridge` and `Lasso` regression.

1.5.4 Ridge Regression

[84]:
```python
# list of alphas to tune
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, 10.0, 20, 50, 100, 500, 1000 ]}

ridge = Ridge()
```

```python
# cross validation
folds = 5
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
model_cv.fit(X_train, y_train)
```

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

[84]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             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, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

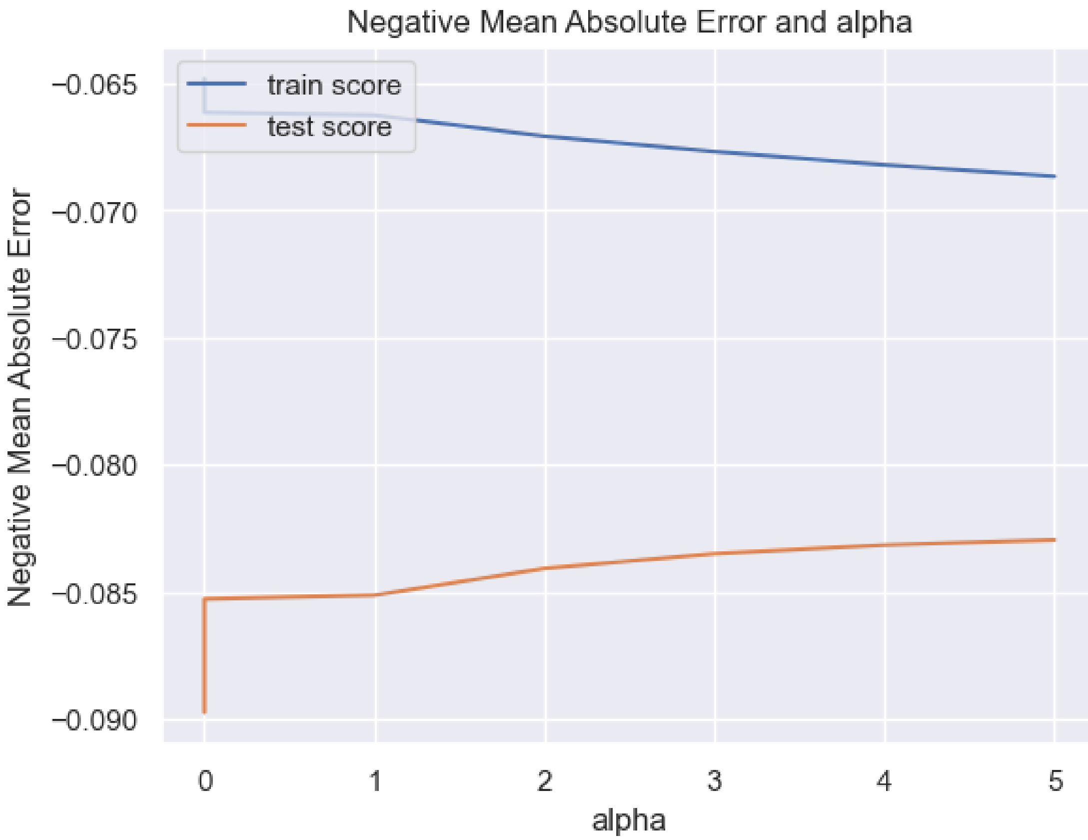
[85]:
```python
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results = cv_results[cv_results['param_alpha']<=5]
cv_results.head()
```

[85]:
```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time param_alpha  \
0       0.005714      0.000827         0.002016        0.000642      0.0001
1       0.004400      0.000800         0.001463        0.000458       0.001
2       0.005510      0.000448         0.002009        0.000019        0.01
3       0.005753      0.000416         0.001599        0.000489        0.05
4       0.005719      0.001286         0.002037        0.000639         0.1

              params  split0_test_score  split1_test_score  split2_test_score  \
0  {'alpha': 0.0001}          -0.098827          -0.080586          -0.093923
1   {'alpha': 0.001}          -0.098810          -0.080580          -0.093891
2    {'alpha': 0.01}          -0.098639          -0.080523          -0.093591
3    {'alpha': 0.05}          -0.097945          -0.080315          -0.092583
4     {'alpha': 0.1}          -0.097200          -0.080133          -0.091735

   split3_test_score  …  mean_test_score  std_test_score  rank_test_score  \
0          -0.093385  …        -0.089726        0.007190               26
1          -0.093357  …        -0.089703        0.007188               25
2          -0.093089  …        -0.089491        0.007154               24
3          -0.092083  …        -0.088811        0.006922               23
4          -0.091160  …        -0.088212        0.006658               22

   split0_train_score  split1_train_score  split2_train_score  \
0           -0.063890           -0.066811           -0.062901
```

```
1           -0.063891           -0.066811           -0.062903
2           -0.063904           -0.066811           -0.062931
3           -0.063979           -0.066830           -0.063098
4           -0.064088           -0.066874           -0.063300

   split3_train_score  split4_train_score  mean_train_score  std_train_score
0           -0.064702           -0.065821         -0.064825         0.001380
1           -0.064704           -0.065823         -0.064826         0.001380
2           -0.064715           -0.065846         -0.064842         0.001374
3           -0.064761           -0.065935         -0.064921         0.001335
4           -0.064818           -0.066043         -0.065025         0.001292

[5 rows x 21 columns]
```

[86]:

```
# plotting mean test and train scoes with alpha
cv_results['param_alpha'] = cv_results['param_alpha'].astype('int32')

# plotting
plt.plot(cv_results['param_alpha'], cv_results['mean_train_score'])
plt.plot(cv_results['param_alpha'], cv_results['mean_test_score'])
plt.xlabel('alpha')
plt.ylabel('Negative Mean Absolute Error')
plt.title("Negative Mean Absolute Error and alpha")
plt.legend(['train score', 'test score'], loc='upper left')
plt.show()
```



Since the `Negative Mean Absolute Error` stabilises at `alpha = 2`;hence, we will choose this for further analysis

[87]:
```
alpha = 2
ridge = Ridge(alpha=alpha)

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

[87]:
```
array([-0.01661814,  0.00879344,  0.02534461,  0.06457657,  0.05124942,
       -0.00169877,  0.0304906 ,  0.00747482,  0.03650081,  0.03991915,
        0.04767425,  0.00523671,  0.07595503,  0.01170416, -0.00092158,
        0.00892371,  0.01249545,  0.0003776 , -0.00534099,  0.00350875,
        0.02078896,  0.01348507, -0.0010637 ,  0.01230575,  0.00491464,
        0.0141215 ,  0.00971184, -0.0702245 ,  0.14857807,  0.10523788,
        0.12502615,  0.09704995,  0.00219378, -0.0208621 ,  0.00027408,
        0.01235864, -0.02072982,  0.00302972,  0.01744635, -0.02148039,
       -0.05117096, -0.02397809,  0.01020923, -0.03899013, -0.0150698 ,
       -0.02340803,  0.00931668,  0.03728476, -0.00759928,  0.11394039,
       -0.05593223, -0.00971649, -0.08597375, -0.11266515, -0.04195278,
       -0.01867322,  0.04765164, -0.02093394,  0.03560318,  0.04414944,
```

```
       -0.0527455 , -0.04663915, -0.00821161,  0.00358752,  0.02349391,
        0.09259274, -0.00371216,  0.03744706,  0.01443113,  0.06657113,
       -0.03426037,  0.0480804 , -0.0564439 ,  0.04897665,  0.0186921 ,
        0.02829856,  0.03294526,  0.00356988, -0.04904027, -0.00211279,
        0.01248938, -0.00336536, -0.09974669, -0.02243104, -0.00934262,
        0.02158726,  0.02205955, -0.02477328, -0.04896389, -0.03375157,
        0.03933884, -0.00116791,  0.        , -0.09570161,  0.06855018,
       -0.01564316, -0.02090358, -0.02176426,  0.00262202,  0.02660443,
       -0.01864088, -0.00152479,  0.05491534, -0.02252611, -0.04617031,
       -0.01376371,  0.01452098, -0.05031844, -0.03514282, -0.01564316,
        0.0326572 ,  0.0041462 , -0.02834453, -0.00727858,  0.        ,
       -0.00138449, -0.04145296, -0.03032255,  0.02193795,  0.03912536,
       -0.0109263 ,  0.03016852,  0.02426654,  0.0524687 , -0.01805023,
        0.00639812,  0.00181613,  0.01333847, -0.04795866, -0.00713641,
        0.        ,  0.01099352,  0.02031477,  0.05093578,  0.00145567,
        0.06650144, -0.05886281,  0.01621104, -0.01186543, -0.01599737,
        0.0114272 ,  0.01496961,  0.00631816,  0.01340311,  0.0114272 ,
        0.04763113, -0.0026089 , -0.01006554, -0.00752408,  0.01189597,
        0.00822396, -0.02919434, -0.02362872, -0.01454754,  0.0114272 ,
       -0.01735872,  0.01146087, -0.01556235, -0.04442294,  0.00528201,
       -0.01971572, -0.02108974, -0.01651765, -0.01721806, -0.03316122,
        0.05067932,  0.00998974, -0.01908463, -0.03386936, -0.00034725,
       -0.06841249, -0.0654142 , -0.07996969, -0.02470646,  0.00964123,
        0.05003853, -0.00453408, -0.01864042,  0.03979777, -0.01172016,
        0.0284846 ,  0.01518725,  0.04076478, -0.01840686, -0.00114335,
       -0.0018659 , -0.01840686, -0.04303768,  0.02905014, -0.04131782,
       -0.01767408, -0.01840686, -0.00629071,  0.00702421,  0.01445663,
       -0.00518292,  0.02600727,  0.05396848,  0.03299049,  0.0584817 ,
       -0.01713427, -0.00039039,  0.01515489,  0.01168028, -0.01466376,
        0.06126474, -0.01023007,  0.01176977,  0.0736808 ,  0.09692439])
```

[88]:
```python
# Ridge model parameters
model_parameters_rg = list(ridge.coef_)
model_parameters_rg.insert(0, ridge.intercept_)
model_parameters_rg = [round(x, 3) for x in model_parameters_rg]
cols = X.columns
cols = cols.insert(0, "constant")
list(zip(cols, model_parameters_rg))
```

[88]:
```
[('constant', 11.739),
 ('MSSubClass', -0.017),
 ('LotFrontage', 0.009),
 ('LotArea', 0.025),
 ('OverallQual', 0.065),
 ('OverallCond', 0.051),
 ('MasVnrArea', -0.002),
 ('BsmtFinSF1', 0.03),
```

```
('BsmtFinSF2', 0.007),
('TotalBsmtSF', 0.037),
('1stFlrSF', 0.04),
('2ndFlrSF', 0.048),
('LowQualFinSF', 0.005),
('GrLivArea', 0.076),
('BsmtFullBath', 0.012),
('BsmtHalfBath', -0.001),
('FullBath', 0.009),
('HalfBath', 0.012),
('BedroomAbvGr', 0.0),
('KitchenAbvGr', -0.005),
('Fireplaces', 0.004),
('GarageArea', 0.021),
('WoodDeckSF', 0.013),
('OpenPorchSF', -0.001),
('EnclosedPorch', 0.012),
('3SsnPorch', 0.005),
('ScreenPorch', 0.014),
('PoolArea', 0.01),
('PropAge', -0.07),
('MSZoning_FV', 0.149),
('MSZoning_RH', 0.105),
('MSZoning_RL', 0.125),
('MSZoning_RM', 0.097),
('LotShape_IR2', 0.002),
('LotShape_IR3', -0.021),
('LotShape_Reg', 0.0),
('LandContour_HLS', 0.012),
('LandContour_Low', -0.021),
('LandContour_Lvl', 0.003),
('LotConfig_CulDSac', 0.017),
('LotConfig_FR2', -0.021),
('LotConfig_FR3', -0.051),
('LotConfig_Inside', -0.024),
('LandSlope_Mod', 0.01),
('LandSlope_Sev', -0.039),
('Neighborhood_Blueste', -0.015),
('Neighborhood_BrDale', -0.023),
('Neighborhood_BrkSide', 0.009),
('Neighborhood_ClearCr', 0.037),
('Neighborhood_CollgCr', -0.008),
('Neighborhood_Crawfor', 0.114),
('Neighborhood_Edwards', -0.056),
('Neighborhood_Gilbert', -0.01),
('Neighborhood_IDOTRR', -0.086),
('Neighborhood_MeadowV', -0.113),
```

```
('Neighborhood_Mitchel', -0.042),
('Neighborhood_NAmes', -0.019),
('Neighborhood_NPkVill', 0.048),
('Neighborhood_NWAmes', -0.021),
('Neighborhood_NoRidge', 0.036),
('Neighborhood_NridgHt', 0.044),
('Neighborhood_OldTown', -0.053),
('Neighborhood_SWISU', -0.047),
('Neighborhood_Sawyer', -0.008),
('Neighborhood_SawyerW', 0.004),
('Neighborhood_Somerst', 0.023),
('Neighborhood_StoneBr', 0.093),
('Neighborhood_Timber', -0.004),
('Neighborhood_Veenker', 0.037),
('Condition1_Feedr', 0.014),
('Condition1_Norm', 0.067),
('Condition1_PosA', -0.034),
('Condition1_PosN', 0.048),
('Condition1_RRAe', -0.056),
('Condition1_RRAn', 0.049),
('Condition1_RRNe', 0.019),
('Condition1_RRNn', 0.028),
('BldgType_2fmCon', 0.033),
('BldgType_Duplex', 0.004),
('BldgType_Twnhs', -0.049),
('BldgType_TwnhsE', -0.002),
('HouseStyle_1.5Unf', 0.012),
('HouseStyle_1Story', -0.003),
('HouseStyle_2.5Fin', -0.1),
('HouseStyle_2.5Unf', -0.022),
('HouseStyle_2Story', -0.009),
('HouseStyle_SFoyer', 0.022),
('HouseStyle_SLvl', 0.022),
('RoofStyle_Gable', -0.025),
('RoofStyle_Gambrel', -0.049),
('RoofStyle_Hip', -0.034),
('RoofStyle_Mansard', 0.039),
('RoofStyle_Shed', -0.001),
('Exterior1st_AsphShn', 0.0),
('Exterior1st_BrkComm', -0.096),
('Exterior1st_BrkFace', 0.069),
('Exterior1st_CBlock', -0.016),
('Exterior1st_CemntBd', -0.021),
('Exterior1st_HdBoard', -0.022),
('Exterior1st_ImStucc', 0.003),
('Exterior1st_MetalSd', 0.027),
('Exterior1st_Plywood', -0.019),
```

```
('Exterior1st_Stone', -0.002),
('Exterior1st_Stucco', 0.055),
('Exterior1st_VinylSd', -0.023),
('Exterior1st_Wd Sdng', -0.046),
('Exterior1st_WdShing', -0.014),
('Exterior2nd_AsphShn', 0.015),
('Exterior2nd_Brk Cmn', -0.05),
('Exterior2nd_BrkFace', -0.035),
('Exterior2nd_CBlock', -0.016),
('Exterior2nd_CmentBd', 0.033),
('Exterior2nd_HdBoard', 0.004),
('Exterior2nd_ImStucc', -0.028),
('Exterior2nd_MetalSd', -0.007),
('Exterior2nd_Other', 0.0),
('Exterior2nd_Plywood', -0.001),
('Exterior2nd_Stone', -0.041),
('Exterior2nd_Stucco', -0.03),
('Exterior2nd_VinylSd', 0.022),
('Exterior2nd_Wd Sdng', 0.039),
('Exterior2nd_Wd Shng', -0.011),
('MasVnrType_BrkFace', 0.03),
('MasVnrType_None', 0.024),
('MasVnrType_Stone', 0.052),
('MasVnrType_none', -0.018),
('ExterQual_Fa', 0.006),
('ExterQual_Gd', 0.002),
('ExterQual_TA', 0.013),
('ExterCond_Fa', -0.048),
('ExterCond_Gd', -0.007),
('ExterCond_Po', 0.0),
('ExterCond_TA', 0.011),
('Foundation_CBlock', 0.02),
('Foundation_PConc', 0.051),
('Foundation_Slab', 0.001),
('Foundation_Stone', 0.067),
('Foundation_Wood', -0.059),
('BsmtQual_Fa', 0.016),
('BsmtQual_Gd', -0.012),
('BsmtQual_TA', -0.016),
('BsmtQual_none', 0.011),
('BsmtCond_Gd', 0.015),
('BsmtCond_Po', 0.006),
('BsmtCond_TA', 0.013),
('BsmtCond_none', 0.011),
('BsmtExposure_Gd', 0.048),
('BsmtExposure_Mn', -0.003),
('BsmtExposure_No', -0.01),
```

```
('BsmtExposure_none', -0.008),
('BsmtFinType1_BLQ', 0.012),
('BsmtFinType1_GLQ', 0.008),
('BsmtFinType1_LwQ', -0.029),
('BsmtFinType1_Rec', -0.024),
('BsmtFinType1_Unf', -0.015),
('BsmtFinType1_none', 0.011),
('BsmtFinType2_BLQ', -0.017),
('BsmtFinType2_GLQ', 0.011),
('BsmtFinType2_LwQ', -0.016),
('BsmtFinType2_Rec', -0.044),
('BsmtFinType2_Unf', 0.005),
('BsmtFinType2_none', -0.02),
('HeatingQC_Fa', -0.021),
('HeatingQC_Gd', -0.017),
('HeatingQC_Po', -0.017),
('HeatingQC_TA', -0.033),
('CentralAir_Y', 0.051),
('Electrical_FuseF', 0.01),
('Electrical_FuseP', -0.019),
('Electrical_Mix', -0.034),
('Electrical_SBrkr', -0.0),
('KitchenQual_Fa', -0.068),
('KitchenQual_Gd', -0.065),
('KitchenQual_TA', -0.08),
('FireplaceQu_Fa', -0.025),
('FireplaceQu_Gd', 0.01),
('FireplaceQu_Po', 0.05),
('FireplaceQu_TA', -0.005),
('FireplaceQu_none', -0.019),
('GarageType_Attchd', 0.04),
('GarageType_Basment', -0.012),
('GarageType_BuiltIn', 0.028),
('GarageType_CarPort', 0.015),
('GarageType_Detchd', 0.041),
('GarageType_none', -0.018),
('GarageFinish_RFn', -0.001),
('GarageFinish_Unf', -0.002),
('GarageFinish_none', -0.018),
('GarageQual_Fa', -0.043),
('GarageQual_Gd', 0.029),
('GarageQual_Po', -0.041),
('GarageQual_TA', -0.018),
('GarageQual_none', -0.018),
('PavedDrive_P', -0.006),
('PavedDrive_Y', 0.007),
('MiscFeature_Othr', 0.014),
```

```
 ('MiscFeature_Shed', -0.005),
 ('MiscFeature_none', 0.026),
 ('SaleType_CWD', 0.054),
 ('SaleType_Con', 0.033),
 ('SaleType_ConLD', 0.058),
 ('SaleType_ConLI', -0.017),
 ('SaleType_ConLw', -0.0),
 ('SaleType_New', 0.015),
 ('SaleType_Oth', 0.012),
 ('SaleType_WD', -0.015),
 ('SaleCondition_AdjLand', 0.061),
 ('SaleCondition_Alloca', -0.01),
 ('SaleCondition_Family', 0.012),
 ('SaleCondition_Normal', 0.074),
 ('SaleCondition_Partial', 0.097)]
```

[89]:
```
# Ridge regression
lm = Ridge(alpha=alpha)
lm.fit(X_train, y_train)

# predict
y_train_pred = lm.predict(X_train)
print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred))
y_test_pred = lm.predict(X_test)
print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred))
```

```
0.9364594823911135
0.9077597079466583
```

[90]:
```
print('RMSE :', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
RMSE : 0.11485785595060986
```

1.5.5 Lasso Regression

[91]:
```
# Checking the dimension of X_train & y_train
print("X_train", X_train.shape)
print("y_train", y_train.shape)
```

```
X_train (1000, 210)
y_train (1000,)
```

[92]:
```
# Applying Lasso

# list of alphas to tune
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, 10.0, 20, 50, 100, 500, 1000 ]}
lasso = Lasso()
```

```python
# cross validation
folds = 5
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

model_cv.fit(X_train, y_train)
```

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

[92]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             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, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

[93]:
```python
# cv_results
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results = cv_results[cv_results['param_alpha']<=1]
cv_results.head()
```

[93]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	\
0	0.048401	0.002322	0.003136	0.000861	0.0001	
1	0.012932	0.000920	0.002619	0.000510	0.001	
2	0.007377	0.001538	0.002140	0.000701	0.01	
3	0.006162	0.001608	0.001824	0.000478	0.05	
4	0.005460	0.001219	0.003050	0.001221	0.1	

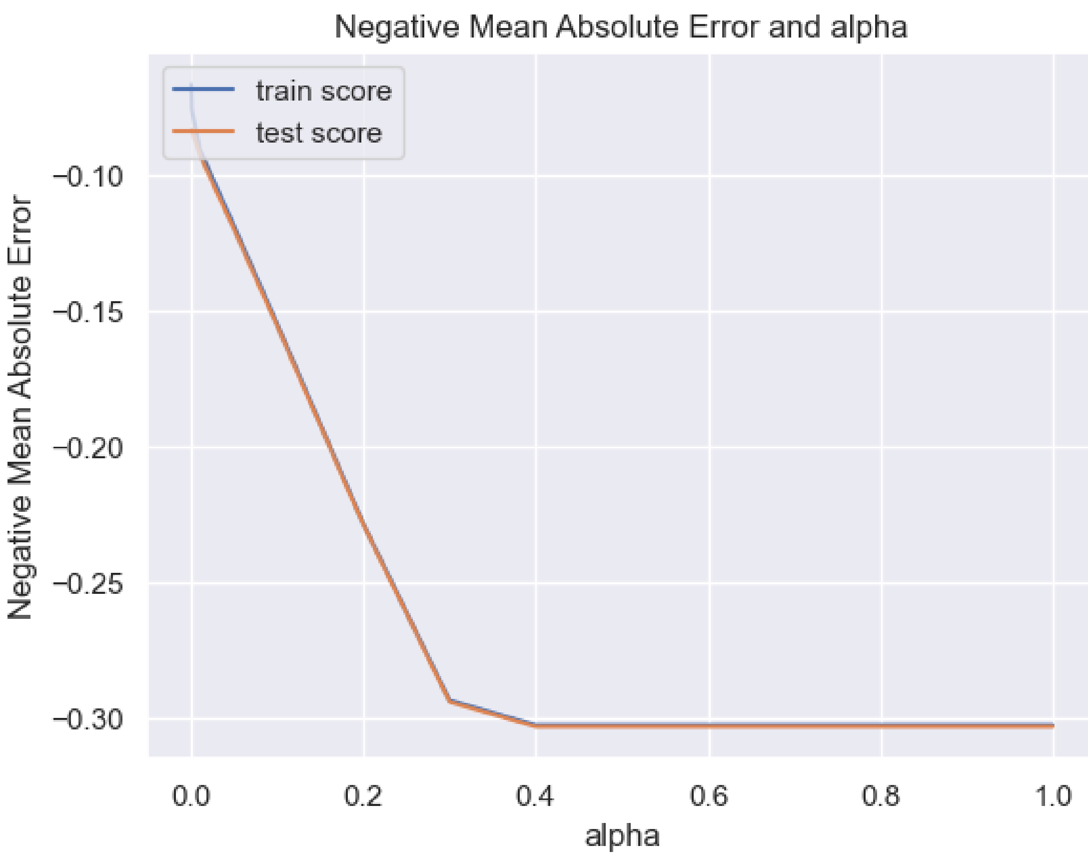
	params	split0_test_score	split1_test_score	split2_test_score	\
0	{'alpha': 0.0001}	-0.089702	-0.078848	-0.088372	
1	{'alpha': 0.001}	-0.082755	-0.083177	-0.087933	
2	{'alpha': 0.01}	-0.090790	-0.093399	-0.097715	
3	{'alpha': 0.05}	-0.129246	-0.113828	-0.132264	
4	{'alpha': 0.1}	-0.172055	-0.148653	-0.168033	

	split3_test_score	…	mean_test_score	std_test_score	rank_test_score	\
0	-0.085734	…	-0.083955	0.005070	2	
1	-0.082553	…	-0.083213	0.002670	1	
2	-0.092036	…	-0.092514	0.003038	3	
3	-0.112722	…	-0.120237	0.008648	4	
4	-0.143888	…	-0.155762	0.011827	5	

```
   split0_train_score  split1_train_score  split2_train_score  \
0           -0.065801           -0.068092           -0.065346
1           -0.075478           -0.077009           -0.074278
2           -0.091236           -0.090420           -0.089019
3           -0.119298           -0.118718           -0.117623
4           -0.155429           -0.155643           -0.153254

   split3_train_score  split4_train_score  mean_train_score  std_train_score
0           -0.066311           -0.067607         -0.066631         0.001052
1           -0.076496           -0.076136         -0.075880         0.000943
2           -0.090524           -0.091156         -0.090471         0.000796
3           -0.118882           -0.119586         -0.118822         0.000673
4           -0.153682           -0.156270         -0.154856         0.001174

[5 rows x 21 columns]
```

[94]:

```
# plotting mean test and train scoes with alpha
cv_results['param_alpha'] = cv_results['param_alpha'].astype('float32')

# plotting
plt.plot(cv_results['param_alpha'], cv_results['mean_train_score'])
plt.plot(cv_results['param_alpha'], cv_results['mean_test_score'])
plt.xlabel('alpha')
plt.ylabel('Negative Mean Absolute Error')
plt.title("Negative Mean Absolute Error and alpha")
plt.legend(['train score', 'test score'], loc='upper left')
plt.show()
```



As per above graph, we can see that the `Negative Mean Absolute Error` is quite low at `alpha = 0.4` and stabilises thereafter;but we will choose a low value of alpha to balance the trade-off between Bias-Variance and to get the coefficients of smallest of features.

[95]:
```
# At alpha = 0.01, even the smallest of negative coefficients that have some␣
↪predictive power towards 'SalePrice' have been generated
alpha = 0.01
lasso = Lasso(alpha=alpha)

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

[95]:
```
array([-0.00698318,  0.01385863,  0.01530092,  0.11219857,  0.04956005,
        0.        ,  0.03451586,  0.        ,  0.04155321,  0.        ,
        0.        , -0.        ,  0.12480705,  0.00976136,  0.        ,
        0.        ,  0.        , -0.        , -0.00805575,  0.02372561,
        0.033997  ,  0.00962824,  0.        ,  0.        ,  0.        ,
        0.00541322,  0.        , -0.0953824 ,  0.        , -0.        ,
        0.        , -0.        ,  0.        , -0.        , -0.        ,
        0.        ,  0.        , -0.        ,  0.        ,  0.        ,
```

```
       -0.        , -0.        ,  0.        ,  0.        , -0.        ,
       -0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.        ,  0.        , -0.        , -0.        , -0.        ,
        0.        , -0.        , -0.        ,  0.        ,  0.        ,
       -0.        , -0.        , -0.        , -0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.        ,  0.        ,
       -0.        ,  0.        , -0.        ,  0.        ,  0.        ,
       -0.        , -0.        , -0.        , -0.        ,  0.        ,
       -0.        ,  0.        , -0.        , -0.        ,  0.        ,
       -0.        ,  0.        , -0.        , -0.        ,  0.        ,
        0.        , -0.        ,  0.        , -0.        ,  0.        ,
       -0.        ,  0.        , -0.        , -0.        ,  0.        ,
       -0.        ,  0.        ,  0.        ,  0.        , -0.        ,
       -0.        ,  0.        , -0.        ,  0.        , -0.        ,
        0.        , -0.        , -0.        ,  0.        ,  0.        ,
       -0.        ,  0.        ,  0.        ,  0.        , -0.        ,
       -0.        , -0.        , -0.        ,  0.        , -0.        ,
       -0.        ,  0.        , -0.        , -0.        ,  0.        ,
        0.        ,  0.        , -0.        ,  0.        , -0.        ,
        0.        , -0.        ,  0.        ,  0.        , -0.        ,
        0.        ,  0.        , -0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.        ,  0.        ,  0.        ,
        0.        , -0.        , -0.        , -0.        ,  0.        ,
       -0.        ,  0.        ,  0.        , -0.        , -0.        ,
        0.        , -0.        , -0.        , -0.        , -0.        ,
        0.        , -0.        , -0.        , -0.        ,  0.        ,
       -0.        ,  0.        , -0.00713365,  0.        ,  0.        ,
        0.        , -0.        , -0.        ,  0.        , -0.        ,
        0.        , -0.        , -0.        , -0.        ,  0.        ,
       -0.        , -0.        , -0.        ,  0.        , -0.        ,
        0.        , -0.        , -0.        ,  0.        ,  0.        ,
       -0.        ,  0.        ,  0.        ,  0.        , -0.        ,
       -0.        , -0.        ,  0.        , -0.        , -0.        ,
        0.        , -0.        , -0.        ,  0.        ,  0.        ])
```

The advantage of this technique is clearly visible here as Lasso brings the coefficients of insignificant features to zero

[96]:
```
# Lasso model parameters
model_parameters_ls = list(lasso.coef_ )
model_parameters_ls.insert(0, lasso.intercept_)
model_parameters_ls = [round(x, 3) for x in model_parameters_ls]
cols = X.columns
cols = cols.insert(0, "constant")
list(zip(cols, model_parameters_ls))
```

[96]:
```
[('constant', 12.003),
 ('MSSubClass', -0.007),
 ('LotFrontage', 0.014),
 ('LotArea', 0.015),
 ('OverallQual', 0.112),
 ('OverallCond', 0.05),
 ('MasVnrArea', 0.0),
 ('BsmtFinSF1', 0.035),
 ('BsmtFinSF2', 0.0),
 ('TotalBsmtSF', 0.042),
 ('1stFlrSF', 0.0),
 ('2ndFlrSF', 0.0),
 ('LowQualFinSF', -0.0),
 ('GrLivArea', 0.125),
 ('BsmtFullBath', 0.01),
 ('BsmtHalfBath', 0.0),
 ('FullBath', 0.0),
 ('HalfBath', 0.0),
 ('BedroomAbvGr', -0.0),
 ('KitchenAbvGr', -0.008),
 ('Fireplaces', 0.024),
 ('GarageArea', 0.034),
 ('WoodDeckSF', 0.01),
 ('OpenPorchSF', 0.0),
 ('EnclosedPorch', 0.0),
 ('3SsnPorch', 0.0),
 ('ScreenPorch', 0.005),
 ('PoolArea', 0.0),
 ('PropAge', -0.095),
 ('MSZoning_FV', 0.0),
 ('MSZoning_RH', -0.0),
 ('MSZoning_RL', 0.0),
 ('MSZoning_RM', -0.0),
 ('LotShape_IR2', 0.0),
 ('LotShape_IR3', -0.0),
 ('LotShape_Reg', -0.0),
 ('LandContour_HLS', 0.0),
 ('LandContour_Low', 0.0),
 ('LandContour_Lvl', -0.0),
 ('LotConfig_CulDSac', 0.0),
 ('LotConfig_FR2', 0.0),
 ('LotConfig_FR3', -0.0),
 ('LotConfig_Inside', -0.0),
 ('LandSlope_Mod', 0.0),
 ('LandSlope_Sev', 0.0),
 ('Neighborhood_Blueste', -0.0),
 ('Neighborhood_BrDale', -0.0),
```

```
('Neighborhood_BrkSide', 0.0),
('Neighborhood_ClearCr', 0.0),
('Neighborhood_CollgCr', 0.0),
('Neighborhood_Crawfor', 0.0),
('Neighborhood_Edwards', -0.0),
('Neighborhood_Gilbert', 0.0),
('Neighborhood_IDOTRR', -0.0),
('Neighborhood_MeadowV', -0.0),
('Neighborhood_Mitchel', -0.0),
('Neighborhood_NAmes', 0.0),
('Neighborhood_NPkVill', -0.0),
('Neighborhood_NWAmes', -0.0),
('Neighborhood_NoRidge', 0.0),
('Neighborhood_NridgHt', 0.0),
('Neighborhood_OldTown', -0.0),
('Neighborhood_SWISU', -0.0),
('Neighborhood_Sawyer', -0.0),
('Neighborhood_SawyerW', -0.0),
('Neighborhood_Somerst', 0.0),
('Neighborhood_StoneBr', 0.0),
('Neighborhood_Timber', 0.0),
('Neighborhood_Veenker', 0.0),
('Condition1_Feedr', -0.0),
('Condition1_Norm', 0.0),
('Condition1_PosA', -0.0),
('Condition1_PosN', 0.0),
('Condition1_RRAe', -0.0),
('Condition1_RRAn', 0.0),
('Condition1_RRNe', 0.0),
('Condition1_RRNn', -0.0),
('BldgType_2fmCon', -0.0),
('BldgType_Duplex', -0.0),
('BldgType_Twnhs', -0.0),
('BldgType_TwnhsE', 0.0),
('HouseStyle_1.5Unf', -0.0),
('HouseStyle_1Story', 0.0),
('HouseStyle_2.5Fin', -0.0),
('HouseStyle_2.5Unf', -0.0),
('HouseStyle_2Story', 0.0),
('HouseStyle_SFoyer', -0.0),
('HouseStyle_SLvl', 0.0),
('RoofStyle_Gable', -0.0),
('RoofStyle_Gambrel', -0.0),
('RoofStyle_Hip', 0.0),
('RoofStyle_Mansard', 0.0),
('RoofStyle_Shed', -0.0),
('Exterior1st_AsphShn', 0.0),
```

```
('Exterior1st_BrkComm', -0.0),
('Exterior1st_BrkFace', 0.0),
('Exterior1st_CBlock', -0.0),
('Exterior1st_CemntBd', 0.0),
('Exterior1st_HdBoard', -0.0),
('Exterior1st_ImStucc', -0.0),
('Exterior1st_MetalSd', 0.0),
('Exterior1st_Plywood', -0.0),
('Exterior1st_Stone', 0.0),
('Exterior1st_Stucco', 0.0),
('Exterior1st_VinylSd', 0.0),
('Exterior1st_Wd Sdng', -0.0),
('Exterior1st_WdShing', -0.0),
('Exterior2nd_AsphShn', 0.0),
('Exterior2nd_Brk Cmn', -0.0),
('Exterior2nd_BrkFace', 0.0),
('Exterior2nd_CBlock', -0.0),
('Exterior2nd_CmentBd', 0.0),
('Exterior2nd_HdBoard', -0.0),
('Exterior2nd_ImStucc', -0.0),
('Exterior2nd_MetalSd', 0.0),
('Exterior2nd_Other', 0.0),
('Exterior2nd_Plywood', -0.0),
('Exterior2nd_Stone', 0.0),
('Exterior2nd_Stucco', 0.0),
('Exterior2nd_VinylSd', 0.0),
('Exterior2nd_Wd Sdng', -0.0),
('Exterior2nd_Wd Shng', -0.0),
('MasVnrType_BrkFace', -0.0),
('MasVnrType_None', -0.0),
('MasVnrType_Stone', 0.0),
('MasVnrType_none', -0.0),
('ExterQual_Fa', -0.0),
('ExterQual_Gd', 0.0),
('ExterQual_TA', -0.0),
('ExterCond_Fa', -0.0),
('ExterCond_Gd', 0.0),
('ExterCond_Po', 0.0),
('ExterCond_TA', 0.0),
('Foundation_CBlock', -0.0),
('Foundation_PConc', 0.0),
('Foundation_Slab', -0.0),
('Foundation_Stone', 0.0),
('Foundation_Wood', -0.0),
('BsmtQual_Fa', 0.0),
('BsmtQual_Gd', 0.0),
('BsmtQual_TA', -0.0),
```

```
('BsmtQual_none', 0.0),
('BsmtCond_Gd', 0.0),
('BsmtCond_Po', -0.0),
('BsmtCond_TA', 0.0),
('BsmtCond_none', 0.0),
('BsmtExposure_Gd', 0.0),
('BsmtExposure_Mn', 0.0),
('BsmtExposure_No', -0.0),
('BsmtExposure_none', 0.0),
('BsmtFinType1_BLQ', 0.0),
('BsmtFinType1_GLQ', 0.0),
('BsmtFinType1_LwQ', -0.0),
('BsmtFinType1_Rec', -0.0),
('BsmtFinType1_Unf', -0.0),
('BsmtFinType1_none', 0.0),
('BsmtFinType2_BLQ', -0.0),
('BsmtFinType2_GLQ', 0.0),
('BsmtFinType2_LwQ', 0.0),
('BsmtFinType2_Rec', -0.0),
('BsmtFinType2_Unf', -0.0),
('BsmtFinType2_none', 0.0),
('HeatingQC_Fa', -0.0),
('HeatingQC_Gd', -0.0),
('HeatingQC_Po', -0.0),
('HeatingQC_TA', -0.0),
('CentralAir_Y', 0.0),
('Electrical_FuseF', -0.0),
('Electrical_FuseP', -0.0),
('Electrical_Mix', -0.0),
('Electrical_SBrkr', 0.0),
('KitchenQual_Fa', -0.0),
('KitchenQual_Gd', 0.0),
('KitchenQual_TA', -0.007),
('FireplaceQu_Fa', 0.0),
('FireplaceQu_Gd', 0.0),
('FireplaceQu_Po', 0.0),
('FireplaceQu_TA', -0.0),
('FireplaceQu_none', -0.0),
('GarageType_Attchd', 0.0),
('GarageType_Basment', -0.0),
('GarageType_BuiltIn', 0.0),
('GarageType_CarPort', -0.0),
('GarageType_Detchd', -0.0),
('GarageType_none', -0.0),
('GarageFinish_RFn', 0.0),
('GarageFinish_Unf', -0.0),
('GarageFinish_none', -0.0),
```

```
 ('GarageQual_Fa', -0.0),
 ('GarageQual_Gd', 0.0),
 ('GarageQual_Po', -0.0),
 ('GarageQual_TA', 0.0),
 ('GarageQual_none', -0.0),
 ('PavedDrive_P', -0.0),
 ('PavedDrive_Y', 0.0),
 ('MiscFeature_Othr', 0.0),
 ('MiscFeature_Shed', -0.0),
 ('MiscFeature_none', 0.0),
 ('SaleType_CWD', 0.0),
 ('SaleType_Con', 0.0),
 ('SaleType_ConLD', -0.0),
 ('SaleType_ConLI', -0.0),
 ('SaleType_ConLw', -0.0),
 ('SaleType_New', 0.0),
 ('SaleType_Oth', -0.0),
 ('SaleType_WD', -0.0),
 ('SaleCondition_AdjLand', 0.0),
 ('SaleCondition_Alloca', -0.0),
 ('SaleCondition_Family', -0.0),
 ('SaleCondition_Normal', 0.0),
 ('SaleCondition_Partial', 0.0)]
```

[97]:
```
# Lasso regression
lm = Lasso(alpha=alpha)
lm.fit(X_train, y_train)

# prediction on the test set(Using R2)
y_train_pred = lm.predict(X_train)
print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred))
y_test_pred = lm.predict(X_test)
print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred))
```

```
0.8854624158407248
0.8894603158029368
```

[98]:
```
print('RMSE :', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
RMSE : 0.12573595366706636
```

[99]:
```
#### The R2 values for Train and Test matches well, indicating an optimum model
```

[100]:
```
# Creating a dataframe for the coefficients obtained from Lasso
mod = list(zip(cols, model_parameters_ls))
```

[101]:
```
para = pd.DataFrame(mod)
para.columns = ['Variable', 'Coeff']
```

```python
para.head()
```

[101]:
```
      Variable   Coeff
0     constant  12.003
1   MSSubClass  -0.007
2  LotFrontage   0.014
3      LotArea   0.015
4  OverallQual   0.112
```

[102]:
```python
# sort the coefficients in ascending order
para = para.sort_values((['Coeff']), axis = 0, ascending = False)
para
```

[102]:
```
                  Variable   Coeff
0                 constant  12.003
13               GrLivArea   0.125
4              OverallQual   0.112
5              OverallCond   0.050
9              TotalBsmtSF   0.042
..                     …      …
210  SaleCondition_Partial   0.000
173         KitchenQual_TA  -0.007
1               MSSubClass  -0.007
19            KitchenAbvGr  -0.008
28                 PropAge  -0.095

[211 rows x 2 columns]
```

[103]:
```python
# Chose variables whose coefficients are non-zero
pred = pd.DataFrame(para[(para['Coeff'] != 0)])
pred
```

[103]:
```
           Variable   Coeff
0          constant  12.003
13        GrLivArea   0.125
4       OverallQual   0.112
5       OverallCond   0.050
9       TotalBsmtSF   0.042
7        BsmtFinSF1   0.035
21       GarageArea   0.034
20       Fireplaces   0.024
3           LotArea   0.015
2       LotFrontage   0.014
14     BsmtFullBath   0.010
22       WoodDeckSF   0.010
26      ScreenPorch   0.005
173  KitchenQual_TA  -0.007
```

```
1        MSSubClass  -0.007
19      KitchenAbvGr  -0.008
28          PropAge  -0.095
```

[104]:
```
# These 16 variables obtained from Lasso Regression can be concluded to have␣
↪the strong effect on the SalePrice
pred.shape
```

[104]:
```
(17, 2)
```

- These 16 variables obtained from Lasso Regression can be concluded to have the strong effect on the Housing `SalePrice`

[105]:
```
Lassso_var = list(pred['Variable'])
print(Lassso_var)
```

```
['constant', 'GrLivArea', 'OverallQual', 'OverallCond', 'TotalBsmtSF',
'BsmtFinSF1', 'GarageArea', 'Fireplaces', 'LotArea', 'LotFrontage',
'BsmtFullBath', 'WoodDeckSF', 'ScreenPorch', 'KitchenQual_TA', 'MSSubClass',
'KitchenAbvGr', 'PropAge']
```

[106]:
```
X_train_lasso = X_train[['GrLivArea', 'OverallQual', 'OverallCond',␣
↪'TotalBsmtSF', 'BsmtFinSF1', 'GarageArea', 'Fireplaces', 'LotArea',␣
↪'LotFrontage', 'BsmtFullBath', 'WoodDeckSF', 'ScreenPorch',␣
↪'KitchenQual_TA', 'MSSubClass', 'KitchenAbvGr']]

X_train_lasso.head()
```

[106]:

	GrLivArea	OverallQual	OverallCond	TotalBsmtSF	BsmtFinSF1
11	1.923409	2.241710	-0.513939	0.345478	1.323938
1070	-0.932170	-0.764271	-0.513939	0.030191	0.360916
513	-0.860557	-0.012775	-0.513939	0.119563	-0.223442
467	0.401627	-0.764271	1.258264	-0.764234	-0.106571
993	0.079368	-0.012775	-0.513939	-0.709617	-1.008820

	GarageArea	Fireplaces	LotArea	LotFrontage	BsmtFullBath	WoodDeckSF
11	1.267298	2.231812	0.154684	0.746261	1.131973	0.485675
1070	-0.857140	-0.918240	-0.020017	0.130905	1.131973	-0.758474
513	0.077613	-0.918240	-0.115156	0.083570	-0.816345	0.257158
467	-0.734395	2.231812	-0.086269	0.462250	-0.816345	0.663411
993	0.455291	-0.918240	-0.148775	-0.058435	-0.816345	-0.758474

	ScreenPorch	KitchenQual_TA	MSSubClass	KitchenAbvGr
11	-0.268919	0	0.085645	-0.222797
1070	3.351363	1	-0.869945	-0.222797
513	-0.268919	1	-0.869945	-0.222797
467	-0.268919	0	0.324542	-0.222797
993	-0.268919	0	0.085645	-0.222797

[107]:
```
X_train_lasso.shape
```

[107]: (1000, 15)

[108]:
```
X_test_lasso = X_train[['GrLivArea', 'OverallQual', 'OverallCond',␣
 ↪'TotalBsmtSF', 'BsmtFinSF1', 'GarageArea', 'Fireplaces', 'LotArea',␣
 ↪'LotFrontage', 'BsmtFullBath', 'WoodDeckSF', 'ScreenPorch',␣
 ↪'KitchenQual_TA', 'MSSubClass', 'KitchenAbvGr']]
X_test_lasso.head()
```

[108]:
```
      GrLivArea  OverallQual  OverallCond  TotalBsmtSF  BsmtFinSF1  \
11     1.923409     2.241710    -0.513939     0.345478    1.323938
1070  -0.932170    -0.764271    -0.513939     0.030191    0.360916
513   -0.860557    -0.012775    -0.513939     0.119563   -0.223442
467    0.401627    -0.764271     1.258264    -0.764234   -0.106571
993    0.079368    -0.012775    -0.513939    -0.709617   -1.008820

      GarageArea  Fireplaces   LotArea  LotFrontage  BsmtFullBath  WoodDeckSF  \
11      1.267298    2.231812  0.154684     0.746261      1.131973    0.485675
1070   -0.857140   -0.918240 -0.020017     0.130905      1.131973   -0.758474
513     0.077613   -0.918240 -0.115156     0.083570     -0.816345    0.257158
467    -0.734395    2.231812 -0.086269     0.462250     -0.816345    0.663411
993     0.455291   -0.918240 -0.148775    -0.058435     -0.816345   -0.758474

      ScreenPorch  KitchenQual_TA  MSSubClass  KitchenAbvGr
11      -0.268919               0    0.085645     -0.222797
1070     3.351363               1   -0.869945     -0.222797
513     -0.268919               1   -0.869945     -0.222797
467     -0.268919               0    0.324542     -0.222797
993     -0.268919               0    0.085645     -0.222797
```

Create a dataframe for Ridge Coefficients

[109]:
```
# Create a dataframe for Ridge Coefficients
mod_ridge = list(zip(cols, model_parameters_rg))
```

[110]:
```
paraRFE = pd.DataFrame(mod_ridge)
paraRFE.columns = ['Variable', 'Coeff']
res=paraRFE.sort_values(by=['Coeff'], ascending = False)
res.head(20)
```

[110]:
```
                  Variable   Coeff
0                 constant  11.739
29             MSZoning_FV   0.149
31             MSZoning_RL   0.125
50    Neighborhood_Crawfor   0.114
30             MSZoning_RH   0.105
32             MSZoning_RM   0.097
```

```
210  SaleCondition_Partial   0.097
66    Neighborhood_StoneBr   0.093
13               GrLivArea   0.076
209   SaleCondition_Normal   0.074
95     Exterior1st_BrkFace   0.069
70         Condition1_Norm   0.067
136       Foundation_Stone   0.067
4              OverallQual   0.065
206  SaleCondition_AdjLand   0.061
200         SaleType_ConLD   0.058
103     Exterior1st_Stucco   0.055
198           SaleType_CWD   0.054
124        MasVnrType_Stone   0.052
5              OverallCond   0.051
```

[111]:

```python
# Sorting the coefficients in ascending order
paraRFE = paraRFE.sort_values((['Coeff']), axis = 0, ascending = False)
paraRFE
```

[111]:

```
                 Variable   Coeff
0                constant  11.739
29            MSZoning_FV   0.149
31            MSZoning_RL   0.125
50   Neighborhood_Crawfor   0.114
30            MSZoning_RH   0.105
..                    …       …
173         KitchenQual_TA  -0.080
53    Neighborhood_IDOTRR  -0.086
94    Exterior1st_BrkComm  -0.096
83      HouseStyle_2.5Fin  -0.100
54   Neighborhood_MeadowV  -0.113

[211 rows x 2 columns]
```

[112]:

```python
## since there were few coefficients at 0, we removed them from features
predRFE = pd.DataFrame(paraRFE[(paraRFE['Coeff'] != 0)])
predRFE
```

[112]:

```
                 Variable   Coeff
0                constant  11.739
29            MSZoning_FV   0.149
31            MSZoning_RL   0.125
50   Neighborhood_Crawfor   0.114
30            MSZoning_RH   0.105
..                    …       …
173         KitchenQual_TA  -0.080
53    Neighborhood_IDOTRR  -0.086
```

```
94    Exterior1st_BrkComm  -0.096
83      HouseStyle_2.5Fin  -0.100
54   Neighborhood_MeadowV  -0.113

[204 rows x 2 columns]
```

[113]:
```python
predRFE.shape
```

[113]:
```
(204, 2)
```

- As per the R2 values of Train and Test, Ridge regression the model performance is better than Lasso Regression. The train and the test scores are matching well

1.5.6 Observation:

- Though the model performance by Ridge Regression was better in terms of R2 values of Train and Test, it is better to use Lasso, since it brings and assigns a zero value to insignificant features, enabling us to choose the predictive variables.
- It is always advisable to use simple yet robust model.
- Equation can be formulated using the features and coefficients obtained by Lasso
- When the market value of the property is lower than the Predicted Sale Price, its the time to buy.

[114]:
```python
### Assign the Features as x1, x2.....
pred.set_index(pd.Index(['C','x1', 'x2', 'x3', 'x4', 'x5' , 'x6', 'x7', 'x8',␣
 ↪'x9', 'x10', 'x11', 'x12', 'x13', 'x14', 'x15', 'x16']), inplace = True)
pred
```

[114]:

	Variable	Coeff
C	constant	12.003
x1	GrLivArea	0.125
x2	OverallQual	0.112
x3	OverallCond	0.050
x4	TotalBsmtSF	0.042
x5	BsmtFinSF1	0.035
x6	GarageArea	0.034
x7	Fireplaces	0.024
x8	LotArea	0.015
x9	LotFrontage	0.014
x10	BsmtFullBath	0.010
x11	WoodDeckSF	0.010
x12	ScreenPorch	0.005
x13	KitchenQual_TA	-0.007
x14	MSSubClass	-0.007
x15	KitchenAbvGr	-0.008
x16	PropAge	-0.095

These are the final features that should be selected for predicting the price of house

- Equation is : Log(Y) = C + 0.125(x1) + 0.112(x2) + 0.050(x3) + 0.042(x4) + 0.035(x5) + 0.034(x6) + 0.024(x7) + 0.015(x8) + 0.014(x9) + 0.010(x10) + 0.010(x11) + 0.005(x12) - 0.007(x13) - 0.007(x14) - 0.008(x15) - 0.095(x16) + Error term(RSS + alpha * (sum of absolute value of coefficients)

1.5.7 Reference:

- Upgrad course materials
- Machine Learning with Regression in Python: With Ordinary Least Squares, Ridge, Decision Trees and Neural Networks
- Introduction to Linear Regression Analysis
- Practical Statistics for Data Scientists

[ ]:

```

```