Housing Price Prediction

Submitted by:

Abhishek Kumar

ACKNOWLEDGMENT

I would like to thanks to Flip Robo Technologies to give me a wonderful opportunity. This project is given by my SME Ms Sapna Verma. I have referred below resources that helped and guided me in completion of this project as below:-

- https://www.kaggle.com/erick5/predicting-house-prices-with-machine-learning
- https://studygyaan.com/data-science-ml/linear-regression-machine-learning-project-for-house-price-prediction
- https://loddonhouse.co.uk/?gclid=CjwKCAjw-ZCKBhBkEiwAM4qfF_ZWhedS9VWDcP3TZ5_SVB7xuurHYsU5s4MaQzoRhiVB5fnbA1l1DxoC3G0QAvD_BwE

INTRODUCTION

Business Problem Framing

- Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

- A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

- The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

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

Conceptual Background of the Domain Problem

Predicting sale prices for houses, even stranger ones. Use a test-driven approach to build a Linear Regression model using Python from scratch. We will use our trained model to predict house sale prices and extend it to a multivariate Linear Regression.

Review of Literature

We are required to model the price of houses with the available independent variables.

Technical Requirements:

• Data contains 1460 entries each having 81 variables.

• Data contains Null values. We need to treat them using the domain knowledge and your own understanding.

• Extensive EDA has to be performed to gain relationships of important variable and price.

• Data contains numerical as well as categorical variable. We need to handle them accordingly.

• We have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters.

• We need to find important features which affect the price positively or negatively.

• Two datasets are being provided to us (test.csv, train.csv). We will train on train.csv dataset and predict on test.csv file.

The "Data file.csv" and "Data description.txt" are enclosed with this file.

Motivation for the Problem Undertaken

- House is one of the important elements in basic human needs. People need a house to stay away from danger, hot weather, rainy day and as well as a place to stay calm. As long as people can fill the comfort of living under a roof then it is called a house. However, the things that matter is that the affordability of a person to purchase a house. Some people can afford a house that is really comfortable to stay in and some not. People who are called the rich and famous can afford a house that is almost called a heaven and some can only lives in an ordinary but comfortable house. But it doesnââ‚¬â„¢t matter how our house may look like because the price of house is what that matter. We can see that the housing price is increasing as the time goes by. This may be an important area to look upon because more or less it could affect the economic level of a country. Therefore, a housing price can be defined as the rate of payment that one has to pay in order to purchase a house and for sure there are several factors that lead to housing price determination.
- In my own point of view, I believe that the increment of a housing price is due to the price increment in the raw material. Many may have similar idea but after looking into 10 journals as references for my propose topic, I have found out several more important variables that leads to the factors of housing price determination. There are few number of knowledgeable individual turned up and able to find the contributing factors in determination of housing price. One who has studied using an empirical analysis has shown that income (demography trends) and nominal interest rates are the key explanatory factors in housing price. On the other hand, the equity returns may also have been an influential factor in the determination of housing price.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

- This problem is a Linear Regression problem. The dataset is in CSV format and It contains *1460* training data points and 81 features that might help us predict the selling price of a house.
- Build a model of housing prices to predict median house values in California using the provided dataset.
- Train the model to learn from the data to predict the median housing price in any district, given all the other metrics.
- Predict housing prices based on median_income and plot the regression chart for it.

Data Sources and their formats

This Dataset is provided by Flip Robo Technologies CSV format. In this dataset, there are 1460 rows and 81 columns.

Load Data

```
In [92]: #uploading test dataset
test=pd.read_csv("Housing_test.csv")
train=pd.read_csv("Housing_train.csv")
```

```
In [93]: test
```

Out[93]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm	
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm	
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Feedr	
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	BrDale	Norm	

292 rows × 80 columns

```
In [94]: train
```

Out[94]:

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

```
In [102]: #letscheck columns name of both dataset
print(train.columns)
print("**********")
print(test.columns)
```

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

#check information of train and test dataset to find null values and type of columns
print(test.info())
print('********************')
print(train.info())

```
In [103]: #check information of train and test dataset to find null values and type of columns
          print(test.info())
          print('********************')
          print(train.info())

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 292 entries, 0 to 291
          Data columns (total 79 columns):
           #   Column         Non-Null Count  Dtype
          ---  ------         --------------  -----
           0   MSSubClass     292 non-null    int64
           1   MSZoning       292 non-null    object
           2   LotFrontage    247 non-null    float64
           3   LotArea        292 non-null    int64
           4   Street         292 non-null    object
           5   Alley          14 non-null     object
           6   LotShape       292 non-null    object
           7   LandContour    292 non-null    object
           8   Utilities      292 non-null    object
           9   LotConfig      292 non-null    object
           10  LandSlope      292 non-null    object
           11  Neighborhood   292 non-null    object
           12  Condition1     292 non-null    object
           13  Condition2     292 non-null    object
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292 entries, 0 to 291
Data columns (total 79 columns):

```
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     292 non-null    int64
 1   MSZoning       292 non-null    object
 2   LotFrontage    247 non-null    float64
 3   LotArea        292 non-null    int64
 4   Street         292 non-null    object
 5   Alley          14 non-null     object
 6   LotShape       292 non-null    object
 7   LandContour    292 non-null    object
 8   Utilities      292 non-null    object
 9   LotConfig      292 non-null    object
 10  LandSlope      292 non-null    object
 11  Neighborhood   292 non-null    object
 12  Condition1     292 non-null    object
 13  Condition2     292 non-null    object
 14  BldgType       292 non-null    object
 15  HouseStyle     292 non-null    object
 16  OverallQual    292 non-null    int64
 17  OverallCond    292 non-null    int64
 18  YearBuilt      292 non-null    int64
 19  YearRemodAdd   292 non-null    int64
 20  RoofStyle      292 non-null    object
 21  RoofMatl       292 non-null    object
 22  Exterior1st    292 non-null    object
```

23	Exterior2nd	292 non-null	object
24	MasVnrType	291 non-null	object
25	MasVnrArea	291 non-null	float64
26	ExterQual	292 non-null	object
27	ExterCond	292 non-null	object
28	Foundation	292 non-null	object
29	BsmtQual	285 non-null	object
30	BsmtCond	285 non-null	object
31	BsmtExposure	285 non-null	object
32	BsmtFinType1	285 non-null	object
33	BsmtFinSF1	292 non-null	int64
34	BsmtFinType2	285 non-null	object
35	BsmtFinSF2	292 non-null	int64
36	BsmtUnfSF	292 non-null	int64
37	TotalBsmtSF	292 non-null	int64
38	Heating	292 non-null	object
39	HeatingQC	292 non-null	object
40	CentralAir	292 non-null	object
41	Electrical	291 non-null	object
42	1stFlrSF	292 non-null	int64
43	2ndFlrSF	292 non-null	int64
44	LowQualFinSF	292 non-null	int64
45	GrLivArea	292 non-null	int64
46	BsmtFullBath	292 non-null	int64
47	BsmtHalfBath	292 non-null	int64
48	FullBath	292 non-null	int64
49	HalfBath	292 non-null	int64
50	BedroomAbvGr	292 non-null	int64
51	KitchenAbvGr	292 non-null	int64
52	KitchenQual	292 non-null	object
53	TotRmsAbvGrd	292 non-null	int64
54	Functional	292 non-null	object
55	Fireplaces	292 non-null	int64
56	FireplaceQu	153 non-null	object
57	GarageType	275 non-null	object
58	GarageYrBlt	275 non-null	float64
59	GarageFinish	275 non-null	object
60	GarageCars	292 non-null	int64
61	GarageArea	292 non-null	int64
62	GarageQual	275 non-null	object
63	GarageCond	275 non-null	object
64	PavedDrive	292 non-null	object
65	WoodDeckSF	292 non-null	int64
66	OpenPorchSF	292 non-null	int64
67	EnclosedPorch	292 non-null	int64
68	3SsnPorch	292 non-null	int64
69	ScreenPorch	292 non-null	int64

```
 70  PoolArea       292 non-null    int64
 71  PoolQC         0 non-null      float64
 72  Fence          44 non-null     object
 73  MiscFeature    10 non-null     object
 74  MiscVal        292 non-null    int64
 75  MoSold         292 non-null    int64
 76  YrSold         292 non-null    int64
 77  SaleType       292 non-null    object
 78  SaleCondition  292 non-null    object
dtypes: float64(4), int64(33), object(42)
memory usage: 180.3+ KB
None
**********************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1168 non-null   int64
 1   MSZoning       1168 non-null   object
 2   LotFrontage    954 non-null    float64
 3   LotArea        1168 non-null   int64
 4   Street         1168 non-null   object
 5   Alley          77 non-null     object
 6   LotShape       1168 non-null   object
 7   LandContour    1168 non-null   object
 8   Utilities      1168 non-null   object
 9   LotConfig      1168 non-null   object
 10  LandSlope      1168 non-null   object
 11  Neighborhood   1168 non-null   object
 12  Condition1     1168 non-null   object
 13  Condition2     1168 non-null   object
 14  BldgType       1168 non-null   object
 15  HouseStyle     1168 non-null   object
 16  OverallQual    1168 non-null   int64
 17  OverallCond    1168 non-null   int64
 18  YearBuilt      1168 non-null   int64
 19  YearRemodAdd   1168 non-null   int64
 20  RoofStyle      1168 non-null   object
 21  RoofMatl       1168 non-null   object
 22  Exterior1st    1168 non-null   object
 23  Exterior2nd    1168 non-null   object
 24  MasVnrType     1161 non-null   object
 25  MasVnrArea     1161 non-null   float64
 26  ExterQual      1168 non-null   object
 27  ExterCond      1168 non-null   object
 28  Foundation     1168 non-null   object
```

```
29  BsmtQual       1138 non-null   object
30  BsmtCond       1138 non-null   object
31  BsmtExposure   1137 non-null   object
32  BsmtFinType1   1138 non-null   object
33  BsmtFinSF1     1168 non-null   int64
34  BsmtFinType2   1137 non-null   object
35  BsmtFinSF2     1168 non-null   int64
36  BsmtUnfSF      1168 non-null   int64
37  TotalBsmtSF    1168 non-null   int64
38  Heating        1168 non-null   object
39  HeatingQC      1168 non-null   object
40  CentralAir     1168 non-null   object
41  Electrical     1168 non-null   object
42  1stFlrSF       1168 non-null   int64
43  2ndFlrSF       1168 non-null   int64
44  LowQualFinSF   1168 non-null   int64
45  GrLivArea      1168 non-null   int64
46  BsmtFullBath   1168 non-null   int64
47  BsmtHalfBath   1168 non-null   int64
48  FullBath       1168 non-null   int64
49  HalfBath       1168 non-null   int64
50  BedroomAbvGr   1168 non-null   int64
51  KitchenAbvGr   1168 non-null   int64
52  KitchenQual    1168 non-null   object
53  TotRmsAbvGrd   1168 non-null   int64
54  Functional     1168 non-null   object
55  Fireplaces     1168 non-null   int64
56  FireplaceQu    617 non-null    object
57  GarageType     1104 non-null   object
58  GarageYrBlt    1104 non-null   float64
59  GarageFinish   1104 non-null   object
60  GarageCars     1168 non-null   int64
61  GarageArea     1168 non-null   int64
62  GarageQual     1104 non-null   object
63  GarageCond     1104 non-null   object
64  PavedDrive     1168 non-null   object
65  WoodDeckSF     1168 non-null   int64
66  OpenPorchSF    1168 non-null   int64
67  EnclosedPorch  1168 non-null   int64
68  3SsnPorch      1168 non-null   int64
69  ScreenPorch    1168 non-null   int64
70  PoolArea       1168 non-null   int64
71  PoolQC         7 non-null      object
72  Fence          237 non-null    object
73  MiscFeature    44 non-null     object
74  MiscVal        1168 non-null   int64
75  MoSold         1168 non-null   int64
```

```
 76  YrSold         1168 non-null   int64
 77  SaleType       1168 non-null   object
 78  SaleCondition  1168 non-null   object
 79  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(34), object(43)
memory usage: 730.1+ KB
None
```

Check the Data type

```
print(test.dtypes)
print('**************')
print(train.dtypes)
```

```
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
                  ...
MiscVal            int64
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
Length: 79, dtype: object
**************
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 80, dtype: object
```


```
In [104]: print(test.dtypes)
          print('**************')
          print(train.dtypes)

MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
                  ...
MiscVal            int64
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
Length: 79, dtype: object
**************
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 80, dtype: object
```


Observation: There are two types of data present in the dataset categorical and numerical.

Data Preprocessing Done

- I checked the information, data types, null values, correlation of the independent and dependent features and f**rom the correlation table.**
- Some columns can't have any negative value, so those columns were treated accordingly.
- Treated Null vales accordingly columns type.
- Skewness,Outliers are treated manually for the features giving some important information, and then the threshold values were set to make the data free from outliers.
- Applied StandardScaler.
- Applied various machine learning model and compared it.

Handling Missing Values

```
# Let's check the missing values of top 30 columns

print(train.isnull().values.any())

print("********************")

print(train.isnull().sum().sort_values(ascending = False).head(30))

print("******************************************")

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

```
True
********************
PoolQC          1161
MiscFeature     1124
Alley           1091
Fence            931
FireplaceQu      551
LotFrontage      214
GarageType        64
GarageFinish      64
GarageQual        64
GarageCond        64
GarageYrBlt       64
BsmtExposure      31
BsmtFinType2      31
BsmtCond          30
BsmtFinType1      30
BsmtQual          30
MasVnrArea         7
MasVnrType         7
RoofStyle          0
RoofMatl           0
ExterQual          0
Exterior1st        0
Exterior2nd        0
YearBuilt          0
ExterCond          0
Foundation         0
YearRemodAdd       0
```

```
          SalePrice         0
          OverallCond       0
          OverallQual       0
          dtype: int64
****************************************
PoolQC          292
MiscFeature     282
Alley           278
Fence           248
FireplaceQu     139
LotFrontage      45
GarageCond       17
GarageType       17
GarageYrBlt      17
GarageFinish     17
GarageQual       17
BsmtFinType1      7
BsmtExposure      7
BsmtCond          7
BsmtQual          7
BsmtFinType2      7
Electrical        1
MasVnrArea        1
MasVnrType        1
LandSlope         0
RoofMatl          0
MSZoning          0
LotArea           0
Street            0
LotShape          0
Foundation        0
ExterCond         0
ExterQual         0
Exterior2nd       0
Exterior1st       0
dtype: int64
```

Observation:

In train dataset There are 18 columns that have missing values. Major missing values columns are PoolQC, 1124-in MiscFeature, 11091-in Alley, 931-in Fence, 551-in FireplaceQu

There are 1161-missing values in the column PoolQC, 1124-in MiscFeature, 11091-in Alley, 931-in Fence, 551-in FireplaceQu,214- in LotFrontage, 64-each in GarageType, GarageCond, GarageYrBlt, GarageFinish, GarageQual, 31-in BsmtExposure and BsmtFinType2, 30-in BsmtCond and BsmtQual, 7-in MasVnrArea and MasVnrType present in our dataset.

IN test Dataset There are 19 columns that have missing values. Major missing values columns are PoolQC 292 MiscFeature 282 Alley 278 Fence 248 FireplaceQu 139

Total Missing Value Percantage for Train dataset

```
Your selected dataframe has 80 columns.
There are 18 columns that have missing values.
```

Out[107]:

	Missing Values	% of Total Values
PoolQC	1161	99.4
MiscFeature	1124	96.2
Alley	1091	93.4
Fence	931	79.7
FireplaceQu	551	47.2
LotFrontage	214	18.3
GarageType	64	5.5
GarageYrBlt	64	5.5
GarageFinish	64	5.5
GarageQual	64	5.5
GarageCond	64	5.5
BsmtExposure	31	2.7
BsmtFinType2	31	2.7
BsmtCond	30	2.6
BsmtFinType1	30	2.6
BsmtQual	30	2.6
MasVnrArea	7	0.6
MasVnrType	7	0.6

Total Missing Value Percantage for Test Dataset

t[108]:

	Missing Values	% of Total Values
PoolQC	292	100.0
MiscFeature	282	96.6
Alley	278	95.2
Fence	248	84.9
FireplaceQu	139	47.6
LotFrontage	45	15.4
GarageType	17	5.8
GarageYrBlt	17	5.8
GarageFinish	17	5.8
GarageQual	17	5.8
GarageCond	17	5.8
BsmtExposure	7	2.4
BsmtFinType1	7	2.4
BsmtFinType2	7	2.4
BsmtCond	7	2.4
BsmtQual	7	2.4
MasVnrArea	1	0.3
MasVnrType	1	0.3
Electrical	1	0.3

In [105]:

```
# Let's explore the categorical columns

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

MSZoning : ['RL' 'RM' 'FV' 'RH' 'C (all)']

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


Street : ['Pave' 'Grvl']
Pave    1164
Grvl       4
Name: Street, dtype: int64


Alley : [nan 'Grvl' 'Pave']
Grvl    41
Pave    36
Name: Alley, dtype: int64


LotShape : ['IR1' 'Reg' 'IR2' 'IR3']
Reg    740
IR1    390
IR2     32
IR3      6
Name: LotShape, dtype: int64


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


Utilities : ['AllPub']
AllPub    1168
Name: Utilities, dtype: int64


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


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

```
Neighborhood : ['NPkVill' 'NAmes' 'NoRidge' 'NWAmes' 'Gilbert' 'Sawyer' 'Edwards'
 'IDOTRR' 'CollgCr' 'Mitchel' 'Crawfor' 'BrDale' 'StoneBr' 'BrkSide'
 'NridgHt' 'OldTown' 'Somerst' 'Timber' 'SWISU' 'SawyerW' 'ClearCr'
 'Veenker' 'Blmngtn' 'MeadowV' 'Blueste']
NAmes      182
CollgCr    118
OldTown     86
Edwards     83
Somerst     68
Gilbert     64
NridgHt     61
Sawyer      60
NWAmes      59
SawyerW     51
BrkSide     50
Crawfor     45
NoRidge     35
Mitchel     34
IDOTRR      30
Timber      24
ClearCr     24
SWISU       21
StoneBr     19
Blmngtn     15
BrDale      11
MeadowV      9
Veenker      9
NPkVill      8
Blueste      2
Name: Neighborhood, dtype: int64


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


Condition2 : ['Norm' 'RRAe' 'Feedr' 'PosN' 'Artery' 'RRNn' 'PosA' 'RRAn']
Norm      1154
Feedr        6
Artery       2
PosN         2
RRAe         1
RRNn         1
RRAn         1
PosA         1
Name: Condition2, dtype: int64
```

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

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

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

RoofMatl : ['CompShg' 'Tar&Grv' 'WdShngl' 'WdShake' 'Roll' 'ClyTile' 'Metal'
'Membran']
CompShg 1144
Tar&Grv 10
WdShngl 6
WdShake 4
Membran 1
Metal 1
ClyTile 1
Roll 1
Name: RoofMatl, dtype: int64

Exterior1st : ['Plywood' 'Wd Sdng' 'MetalSd' 'CemntBd' 'VinylSd' 'HdBoard' 'Stucco'
'WdShing' 'BrkFace' 'Stone' 'AsbShng' 'AsphShn' 'ImStucc' 'BrkComm']
VinylSd 396
HdBoard 179
MetalSd 178
Wd Sdng 174
Plywood 93
CemntBd 42
BrkFace 41
Stucco 22
WdShing 19
AsbShng 19
Stone 2

```
AsphShn      1
BrkComm      1
ImStucc      1
Name: Exterior1st, dtype: int64


Exterior2nd : ['Plywood' 'Wd Sdng' 'MetalSd' 'CmentBd' 'VinylSd' 'HdBoard' 'Wd Shng'
 'Stucco' 'ImStucc' 'Stone' 'BrkFace' 'AsbShng' 'Brk Cmn' 'AsphShn'
 'Other']
VinylSd    387
MetalSd    173
HdBoard    170
Wd Sdng    165
Plywood    118
CmentBd     42
Wd Shng     31
Stucco      23
BrkFace     20
AsbShng     18
ImStucc      8
Brk Cmn      5
Stone        4
AsphShn      3
Other        1
Name: Exterior2nd, dtype: int64


MasVnrType : ['None' 'BrkFace' 'Stone' 'BrkCmn' nan]
None       696
BrkFace    354
Stone       98
BrkCmn      13
Name: MasVnrType, dtype: int64


ExterQual : ['TA' 'Gd' 'Ex' 'Fa']
TA    717
Gd    397
Ex     43
Fa     11
Name: ExterQual, dtype: int64


ExterCond : ['TA' 'Gd' 'Fa' 'Po' 'Ex']
TA    1022
Gd     117
Fa      26
Ex       2
Po       1
Name: ExterCond, dtype: int64


Foundation : ['CBlock' 'PConc' 'BrkTil' 'Slab' 'Stone' 'Wood']
CBlock    516
PConc     513
BrkTil    112
Slab       21
```

```
Stone       5
Wood        1
Name: Foundation, dtype: int64


BsmtQual : ['Gd' 'TA' 'Ex' nan 'Fa']
TA    517
Gd    498
Ex     94
Fa     29
Name: BsmtQual, dtype: int64


BsmtCond : ['TA' 'Gd' 'Fa' nan 'Po']
TA    1041
Gd      56
Fa      39
Po       2
Name: BsmtCond, dtype: int64


BsmtExposure : ['No' 'Gd' 'Av' 'Mn' nan]
No    756
Av    180
Gd    108
Mn     93
Name: BsmtExposure, dtype: int64


BsmtFinType1 : ['ALQ' 'GLQ' 'BLQ' 'Unf' 'Rec' 'LwQ' nan]
Unf    345
GLQ    330
ALQ    174
BLQ    121
Rec    109
LwQ     59
Name: BsmtFinType1, dtype: int64


BsmtFinType2 : ['Unf' 'Rec' 'BLQ' 'GLQ' nan 'ALQ' 'LwQ']
Unf    1002
Rec      43
LwQ      40
BLQ      24
ALQ      16
GLQ      12
Name: BsmtFinType2, dtype: int64


Heating : ['GasA' 'GasW' 'Floor' 'OthW' 'Wall' 'Grav']
GasA     1143
GasW       14
Grav        5
Wall        4
Floor       1
OthW        1
Name: Heating, dtype: int64
```

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


CentralAir : ['Y' 'N']
Y    1090
N      78
Name: CentralAir, dtype: int64


Electrical : ['SBrkr' 'FuseA' 'FuseF' 'FuseP' 'Mix']
SBrkr    1070
FuseA      74
FuseF      21
FuseP       2
Mix         1
Name: Electrical, dtype: int64


KitchenQual : ['TA' 'Gd' 'Ex' 'Fa']
TA    578
Gd    478
Ex     82
Fa     30
Name: KitchenQual, dtype: int64


Functional : ['Typ' 'Mod' 'Maj1' 'Min1' 'Min2' 'Sev' 'Maj2']
Typ     1085
Min2      30
Min1      25
Mod       12
Maj1      11
Maj2       4
Sev        1
Name: Functional, dtype: int64


FireplaceQu : ['TA' 'Gd' nan 'Fa' 'Ex' 'Po']
Gd    301
TA    252
Fa     25
Ex     21
Po     18
Name: FireplaceQu, dtype: int64


GarageType : ['Attchd' 'BuiltIn' 'Detchd' 'Basment' nan '2Types' 'CarPort']
Attchd     691
Detchd     314
```

```
BuiltIn     70
Basment     16
CarPort      8
2Types       5
Name: GarageType, dtype: int64


GarageFinish : ['RFn' 'Unf' 'Fin' nan]
Unf    487
RFn    339
Fin    278
Name: GarageFinish, dtype: int64


GarageQual : ['TA' 'Fa' nan 'Gd' 'Ex' 'Po']
TA    1050
Fa      39
Gd      11
Po       2
Ex       2
Name: GarageQual, dtype: int64


GarageCond : ['TA' 'Fa' 'Gd' nan 'Po' 'Ex']
TA    1061
Fa      28
Gd       8
Po       6
Ex       1
Name: GarageCond, dtype: int64


PavedDrive : ['Y' 'N' 'P']
Y    1071
N      74
P      23
Name: PavedDrive, dtype: int64


PoolQC : [nan 'Ex' 'Gd' 'Fa']
Gd    3
Fa    2
Ex    2
Name: PoolQC, dtype: int64


Fence : [nan 'MnPrv' 'GdPrv' 'GdWo' 'MnWw']
MnPrv    129
GdPrv     51
GdWo      47
MnWw      10
Name: Fence, dtype: int64


MiscFeature : [nan 'Shed' 'Gar2' 'TenC' 'Othr']
Shed    40
Gar2     2
```

TenC 1
Othr 1
Name: MiscFeature, dtype: int64

SaleType : ['WD' 'COD' 'New' 'ConLI' 'ConLw' 'Con' 'ConLD' 'Oth' 'CWD']
WD 999
New 106
COD 38
ConLD 8
ConLI 5
ConLw 4
Oth 3
CWD 3
Con 2
Name: SaleType, dtype: int64

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

Observation:
There is only one unique value present in utilities column so we will be dropping this column.

2. In categorical columns there are missing values present in coumns Alley, MasVnrType, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, FireplaceQu, GarageType, GarageFinish, GarageQual, GarageCond, PoolQC, Fence, MiscFeature

Fill Missing Values

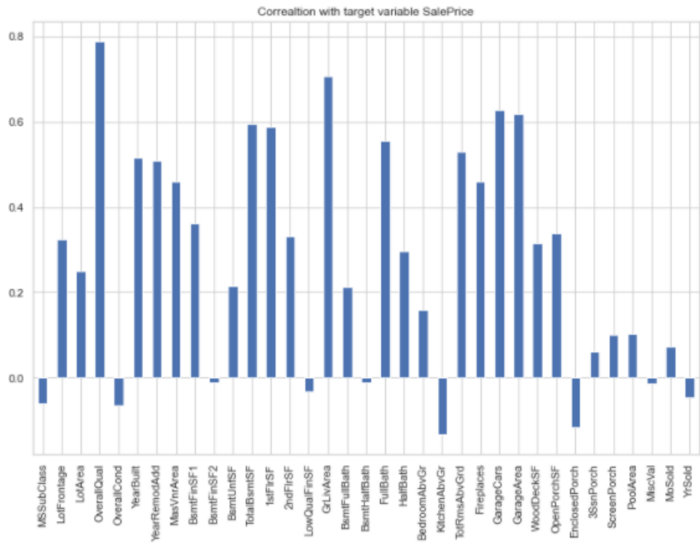
```
In [111]: # Let's fill the missing values in categorical columns as NA in train dataset

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

# Let's fill the missing values in MasVnrType with None
train['MasVnrType'] = train['MasVnrType'].fillna('None')

# Let's fill the missing values in GarageYrBlt with 0
train['GarageYrBlt'] = train['GarageYrBlt'].fillna('0')

# Let's Imputing the missing values and replace it with the median
train['LotFrontage'].fillna(train['LotFrontage'].median(),inplace=True)
train['MasVnrArea'].fillna(train['MasVnrArea'].median(),inplace=True)
```

```
In [112]: # Let's fill the missing values in categorical columns as NA in test dataset

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

# Let's fill the missing values in MasVnrType with None
test['MasVnrType'] = test['MasVnrType'].fillna('None')

# Let's fill the missing values in GarageYrBlt with 0
test['GarageYrBlt'] = test['GarageYrBlt'].fillna('0')

# Let's Imputing the missing values and replace it with the median
test['LotFrontage'].fillna(test['LotFrontage'].median(),inplace=True)
test['MasVnrArea'].fillna(test['MasVnrArea'].median(),inplace=True)
```

Data Inputs- Logic- Output Relationships

Lets check the correlation with target variable "Salesprice".

```
In [118]: # Let's check the correlation with target variable 'SalePrice'

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



Observation:

Observation:

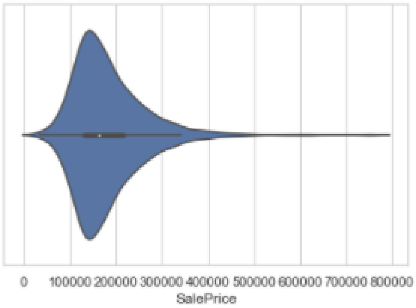
1. The column OverallQual is most positively correlated with SalePrice.

2.The column KitchenAbvGrd and EnclosedPorch is most negatively correlated with SalePrice.

Maximum number of SalePrice lies between 140000 and 230000.

Univatriate Analysis

In [119]:
```
# Let's Check the target variable

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

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



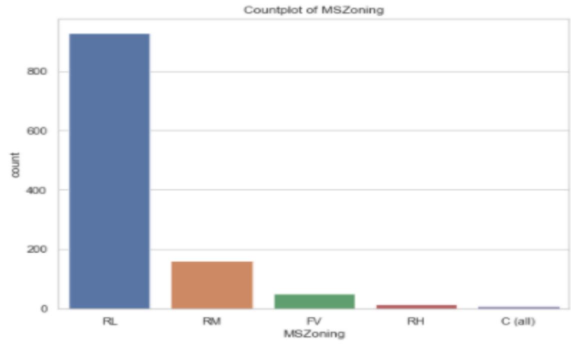
Out[119]:
```
140000    18
135000    16
155000    12
139000    11
160000    11
          ..
126175     1
204000     1
186000     1
369900     1
105500     1
Name: SalePrice, Length: 581, dtype: int64
```

Maximum number of SalePrice lies between 140000 and 230000.

Maximum, 928 number of MSZoning are RL

Maximum number of SalePrice lies between 140000 and 230000.

In [120]:
```
# Let's check the column MSZoning

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

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



Out[120]:
```
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
```

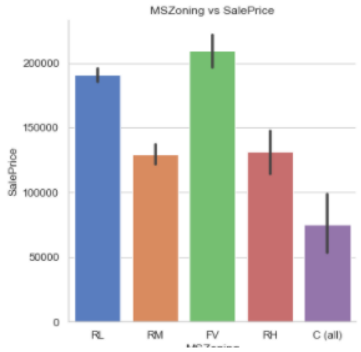
Bivariate Analysis

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

for col in train.describe().columns:

    data=train.copy()

    plt.scatter(data[col],data['SalePrice'])

    plt.xlabel(col)

    plt.ylabel('SalePrice')

    plt.show()
```

1. SalePrice is maximum with FV MSZOning.

```
In [129]: # Let's plot the Factor plot of MSZoning vs SalePrice

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

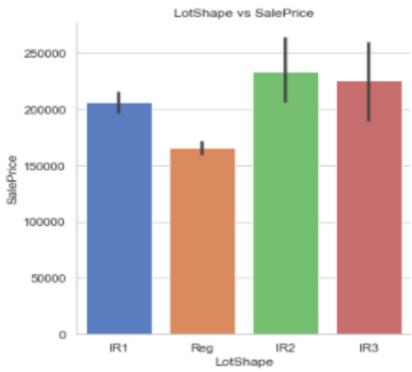
```
<Figure size 576x432 with 0 Axes>
```



2. SalePrice is maximum with IR2 LotShape.

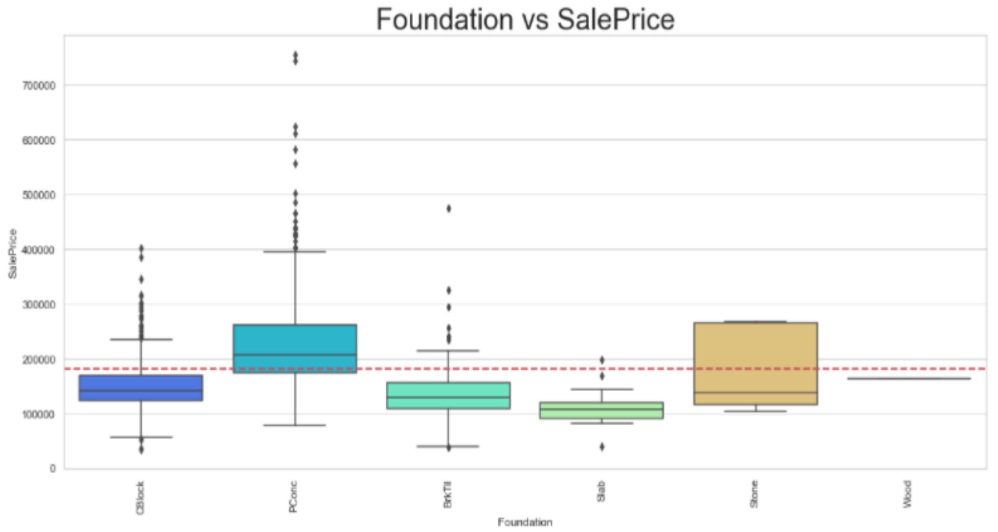
```
In [130]: # Let's plot the Factor plot of LotShape vs SalePrice

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

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



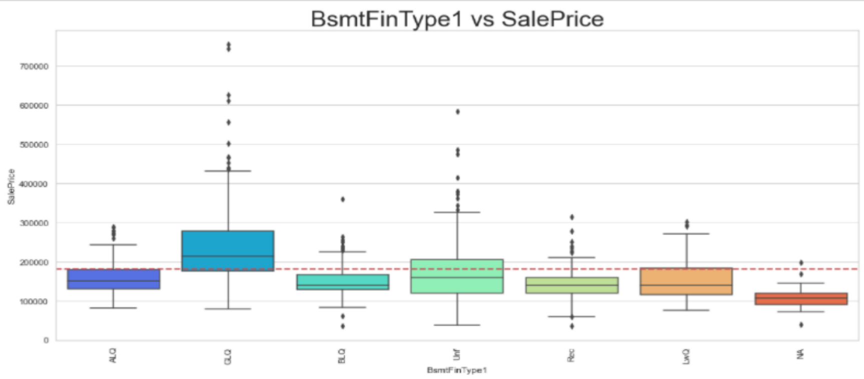
3. SalePrice is maximum with PConc

In [133]:
```
# Let's plot the Foundation vs SalePrice plot

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



4. SalePrice is maximum with GLQ BsmtFinType1.

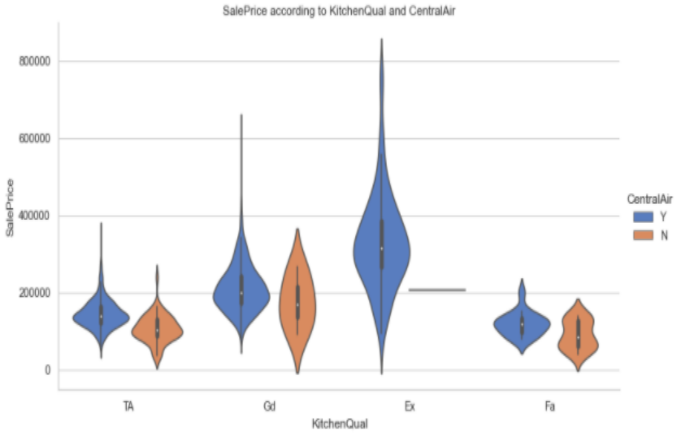
In [134]:
```
# Let's plot the BsmtFinType1 vs SalePrice plot

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



Multivariate Analysis

In [135]:

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

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

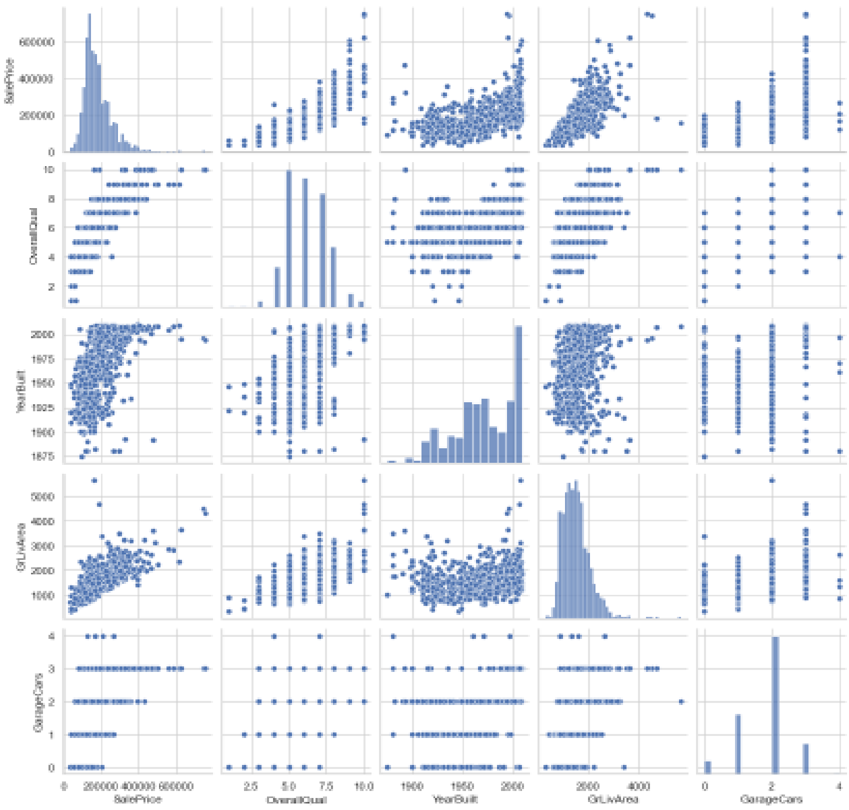


 # Let's plot the pairplot

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

In [136]:

```
# Let's plot the pairplot
sns.pairplot(train, vars=['SalePrice','OverallQual','YearBuilt','GrLivArea','GarageCars']);
```



- State the set of assumptions (if any) related to the problem under consideration
  I have not consider any pre-assumption , project performance from beginning to end is based on data facts only.

- **Hardware and Software Requirements and Tools Used**
  Windows Edition-Windows 8.1 Pro
  Processor-Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00GHz
  Installed memory RAM- 4 GB
  System Type-64 bit OS, x64 based processor

  Software Requirement- Anaconda 4.9.2 , Python 3.8.5, Jupiter Notebook.

  Libraries used:-

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

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

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

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

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

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

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

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
- **Analytical Approach –**Based on type of data by performing EDA I have decided which model to be used for this data.
- **Statistical Approach –** Data should be in scaled manner, it should not be distorted, for that all values using mean method due to continuous data numbers.

Statistical Approach for train Dataset

In [113]:

```
# Let's check the statistical summary of our dataset
train.describe()
```

Out[113]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	56.767979	70.807363	10484.749144	6.104452	5.595890	1970.930651	1984.758562	101.696918	444.726027	46.647260	569.721747
std	41.940650	22.440317	8957.442311	1.390153	1.124343	30.145255	20.785185	182.218483	462.664785	163.520016	449.375525
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000	216.000000
50%	50.000000	70.000000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000	474.000000
75%	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000	816.000000
max	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	2336.000000

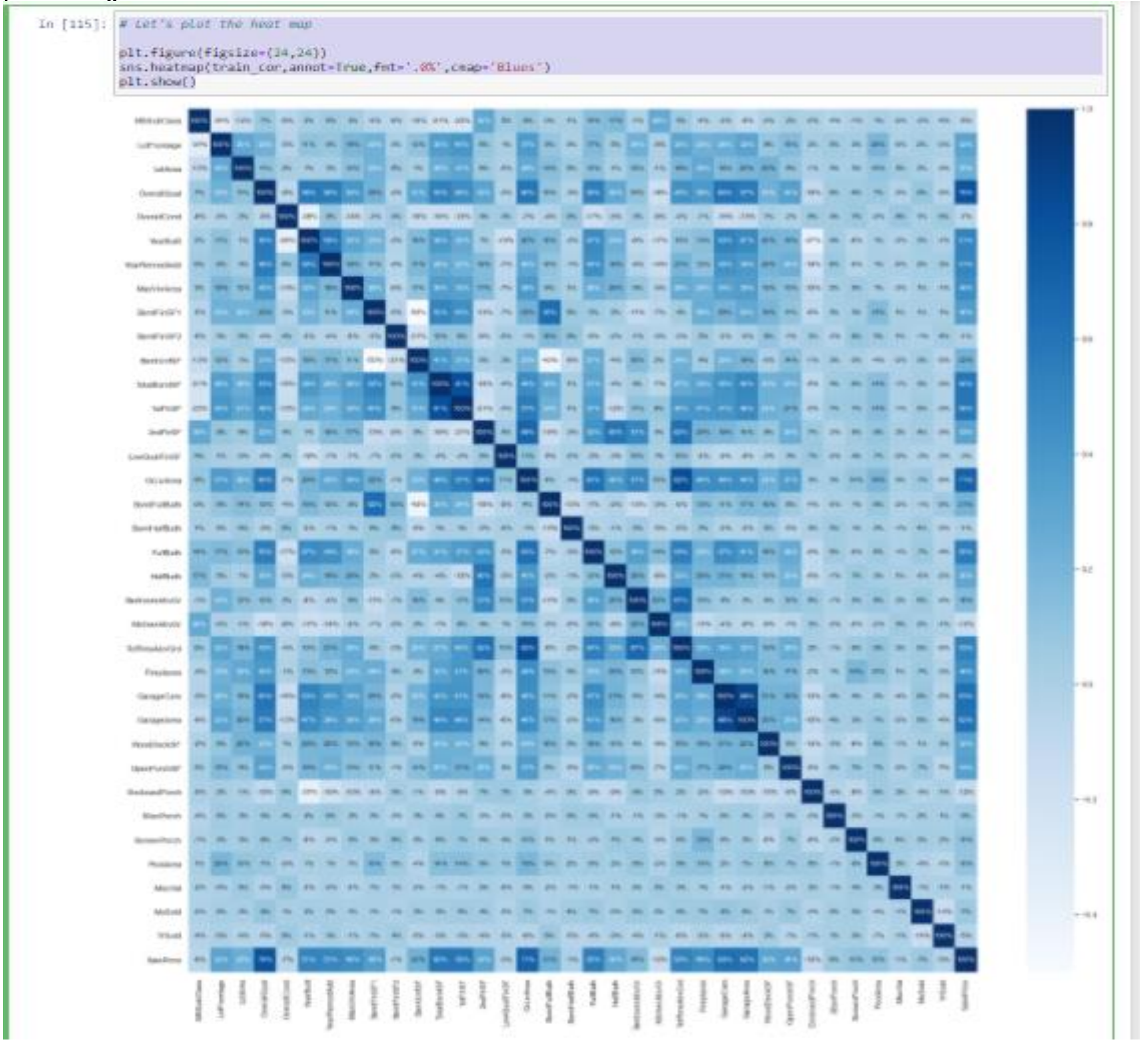
1. Maximum standard deviation of 8957.44 is observed in LotArea column.

2.Maximum SalePrice of a house observed is 755000 and minimum is 34900.

3. In the columns Id, MSSubclass, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, Miscval, salePrice mean is considerably greater than median so the columns are positively skewed.
4. In the columns FullBath, BedroomAbvGr, Fireplaces, Garagecars, GarageArea, YrSold Median is greater than mean so the columns are negatively skewed.
5. In the columns Id, MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, SalePrice there is considerable difference between the 75 percentile and maximum so outliers are present.

Checking Correlation with Heatmap

```
# Let's plot the heat map

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



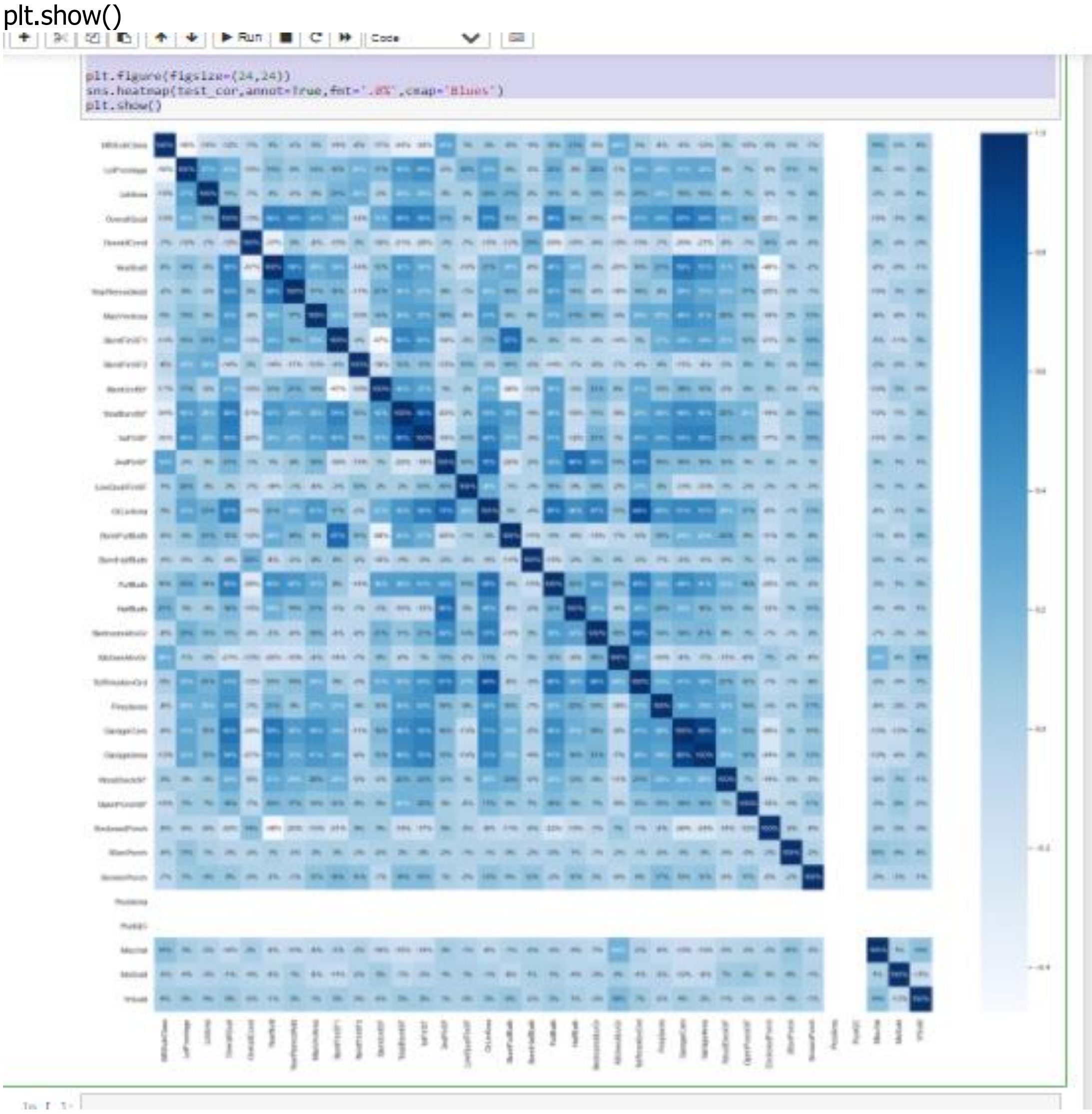
1.SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea.

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

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

Let's plot the heat map for test dataset

plt.figure(figsize=(24,24))
sns.heatmap(test_cor,annot=True,fmt='.0%',cmap='Blues')
plt.show()



Handling Outliers and skewness

Handling outliers and skewness

```
In [142]: # Let's make a copy of our dataset

          train_cap = train.copy()
```

```
In [143]: def percentile_capping(train, cols, from_low_end, from_high_end):
              for col in cols:
                  stats.mstats.winsorize(a=train[col], limits=(from_low_end, from_high_end), inplace=True)
```

```
In [144]: features=['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea','BsmtFi
```

```
In [145]: percentile_capping(train, features, 0.01, 0.10)
          # Let's check the shape

          train_cap.shape

Out[145]: (1168, 244)
```

```
In [146]: for col in features:
              plt.figure(figsize=(16,4))

              plt.subplot(141)
              sns.distplot(train[col], label="skew: " + str(np.round(train[col].skew(),2)))
              plt.title('Before')
              plt.legend()

              plt.subplot(142)
              sns.distplot(train_cap[col], label="skew: " + str(np.round(train_cap[col].skew(),2)))
              plt.title('After')
              plt.legend()

              plt.subplot(143)
              sns.boxplot(train[col])
              plt.title('Before')

              plt.subplot(144)
              sns.boxplot(train_cap[col])
              plt.title('After')
              plt.tight_layout()
              plt.show();
```

- **Testing of Identified Approaches (Algorithms)**

All Algorithms list

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

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

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

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

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

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

Below are Linear Regression algorithms used for the training and testing this dataset.

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

```
In [167]: model=[LinearRegression(),
                 DecisionTreeRegressor(),
                 KNeighborsRegressor(),
                 SVR(),
                 Lasso(),
                 Ridge(),
                 ElasticNet(),
                 RandomForestRegressor(),
                 AdaBoostRegressor(),
```

a. Run and Evaluate selected models

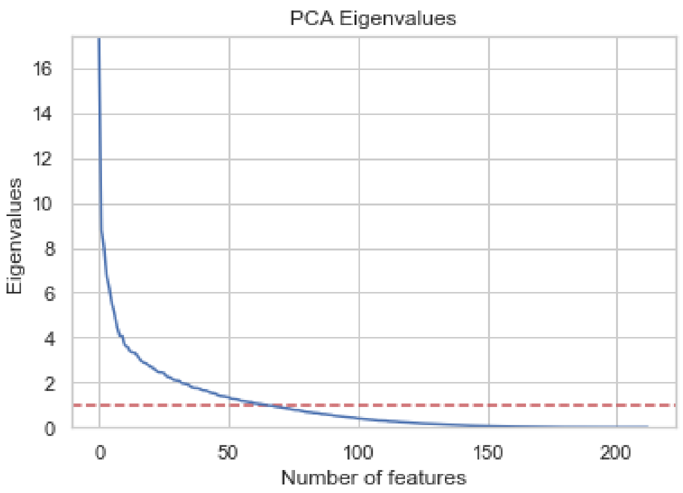
PCA

```
In [154]: # Let's explore the PCA train dataset

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

Out[154]: PCA(n_components=243)
```

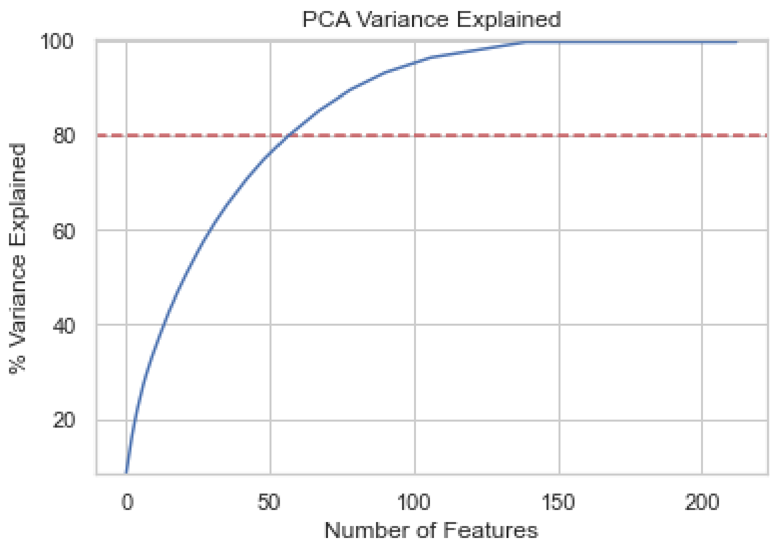
```
In [155]: # Let's check the PCA test dataset

          covar_matrix = PCA(n_components = len(x1.columns))
          covar_matrix.fit(x1)

Out[155]: PCA(n_components=213)
```

```
In [156]: # Let's plot the PCA componenets

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

plt.show()



In [157]:
```
variance = covar_matrix.explained_variance_ratio_
var=np.cumsum(np.round(covar_matrix.explained_variance_ratio_, decimals=3)*100)

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



Lets find the best Random forest score

In [162]:
```
# Let's find the best random state

max_r_score=0
for r_state in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=0.20)
    regr=linear_model.LinearRegression()
    regr.fit(x_train,y_train)
    y_pred=regr.predict(x_test)
    r2_scr=r2_score(y_test,y_pred)
    if r2_scr>max_r_score:
        max_r_score=r2_scr
        final_r_state=r_state
print("max r2 score corresponding to",final_r_state,"is",max_r_score)
```

```
max r2 score corresponding to 48 is 0.8496659416265843
```

max r2 score corresponding to 48 is 0.8496659416265843

In [163]:
```
# Let's split the dataset into test and train

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=54)
```

In [167]:
```
model=[LinearRegression(),
       DecisionTreeRegressor(),
       KNeighborsRegressor(),
       SVR(),
       Lasso(),
       Ridge(),
       ElasticNet(),
       RandomForestRegressor(),
       AdaBoostRegressor(),

      ]
for m in model:
    m.fit(x_train,y_train)
    print('score of',m,'is:',m.score(x_train,y_train))
    predm=m.predict(x_test)
    print('Error:')
    print('Mean absolute error:',mean_absolute_error(y_test,predm))
    print('Mean squared error:',mean_squared_error(y_test,predm))
    print('Root Mean Squared Error:',np.sqrt(mean_squared_error(y_test,predm)))
    print("r2_score:",r2_score(y_test,predm))
    print('*************************************************************')
    print('\n')
```

score of LinearRegression() is: 0.82477807880984
Error:
Mean absolute error: 21214.668674560457
Mean squared error: 988711234.5151851
Root Mean Squared Error: 31443.778947753482
r2_score: 0.8493293024469674
****

score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 32526.239316239316
Mean squared error: 2655740464.3076925
Root Mean Squared Error: 51533.87686083488
r2_score: 0.5952890446589649
****

score of KNeighborsRegressor() is: 0.8005580881741488
Error:
Mean absolute error: 26287.979487179484
Mean squared error: 1638191388.2099144
Root Mean Squared Error: 40474.57706029693
r2_score: 0.7503543698398546
****

score of SVR() is: -0.04568255380776742
Error:
Mean absolute error: 58256.27581850125
Mean squared error: 6883564961.965359
Root Mean Squared Error: 82967.2523467263
r2_score: -0.04899337467274023
****

score of Lasso() is: 0.8247780692387235
Error:
Mean absolute error: 21212.494199237746
Mean squared error: 988627101.9067537

Root Mean Squared Error: 31442.441093317702
r2_score: 0.8493421234996235
****

score of Ridge() is: 0.8247780154162465
Error:
Mean absolute error: 21207.71025561312
Mean squared error: 988429671.2769494
Root Mean Squared Error: 31439.301380230278
r2_score: 0.8493722101514916
****

score of ElasticNet() is: 0.8174701175087531
Error:
Mean absolute error: 19910.440060332676
Mean squared error: 1002174977.3116897
Root Mean Squared Error: 31657.147333764766
r2_score: 0.8472775491665279
****

score of RandomForestRegressor() is: 0.96680098429731
Error:
Mean absolute error: 21953.085384615388
Mean squared error: 1164063400.6093924
Root Mean Squared Error: 34118.37335819796
r2_score: 0.8226072098272706
****

score of AdaBoostRegressor() is: 0.8380210963982864
Error:
Mean absolute error: 30931.247989936797
Mean squared error: 1749721542.0135958
Root Mean Squared Error: 41829.67298477954
r2_score: 0.7333581777413218
****

Ridge is giving us minimum Rmse score so we choose it as our final model.

- **Key Metrics for success in solving problem under consideration**
- Key Metrics used were the Lasso, ridge, Elasticnet to find r2 Score and GridsearcCV score as this was Linear Regression problem and we focus more on R2score metrics to observe Mean absolute error, Mean squared error and Root Mean Squared Error.

- **Visualizations**

Hyperperameter tunning

```
In [79]: # Let's Use the GridSearchCV to find the best paarameters in Ridge Regressor

         parameters={'alpha': [25,10,4,2,1.0,0.8,0.5,0.3,0.2,0.1,0.05,0.02,0.01]}
         rg=Ridge()

         reg=GridSearchCV(rg,parameters,n_jobs=-1)
         reg.fit(x,y)
         print(reg.best_params_)

         {'alpha': 25}
```

```
In [80]: # Let's use the Ridge Regressor with its best parameters

         RG=Ridge(alpha=25)
         RG.fit(x_train,y_train)
         print('Score:',RG.score(x_train,y_train))
         y_pred=RG.predict(x_test)
         print('\n')
         print('Mean absolute error:',mean_absolute_error(y_test,y_pred))
         print('Mean squared error:',mean_squared_error(y_test,y_pred))
         print('Root Mean Squared error:',np.sqrt(mean_squared_error(y_test,y_pred)))
         print('\n')
         print("r2_score:",r2_score(y_test,y_pred))
         print('\n')

         Score: 0.8325270027518725
```

Score: 0.8325270027518725

Mean absolute error: 20277.903007353238
Mean squared error: 921483021.3925847
Root Mean Squared error: 30355.938815865746

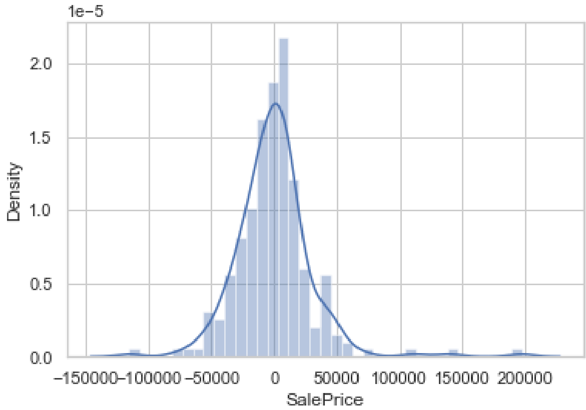
r2_score: 0.8595742773322715

```
In [81]: # Let's Cross validate the Ridge model

         score=cross_val_score(RG,x,y,cv=10,scoring='r2')
         print("Score:",score)
         print('Mean Score:',score.mean())
         print("Standard deviation:",score.std())

         Score: [0.87093015 0.79113836 0.83668642 0.68392408 0.82132711 0.40663273
          0.78924226 0.76344277 0.66557158 0.87778786]
         Mean Score: 0.7506683331134026
         Standard deviation: 0.13280051158633877
```

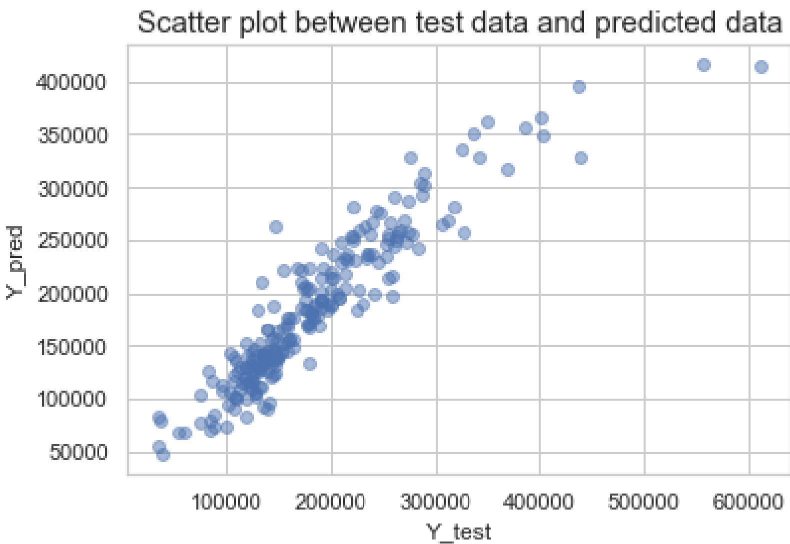
Let's plot the distribution plot and the Gaussian plot

sns.distplot(y_test-y_pred)
plt.show()

```
In [82]: # Let's plot the distribution plot and the Gaussian plot

         sns.distplot(y_test-y_pred)
         plt.show()
```




```
In [83]: # Let' plot the Scatter plot between test data and predicted data

         plt.scatter(y_test,y_pred, alpha=0.5)
         plt.xlabel("Y_test")
         plt.ylabel("Y_pred")
         plt.title("Scatter plot between test data and predicted data",fontsize=15)
         plt.show()
```



- **Interpretation of the Results**
  Data Pre-processing done by performing EDA (Exploratory Data Analysis), checking for best r2 score.

We will save our Model by Ridge Regression as it is giving us minimum Rmse score as it's having 30369.236527153855 and r2_score: 0.8594512207052254.

Model Saving

```
In [84]: #Ridge Regressor is giving us minimum Rmse score so we choose it as our final model.
         # Let's save our best model

         import joblib
         joblib.dump(RG,'Housing_Price_Project.pkl')

Out[84]: ['Housing_Price_Project.pkl']

In [85]: # Let;s load our save model

         model=joblib.load('Housing_Price_Project.pkl')

In [86]: # Let's Test our save model

         import sys
         nums= model.predict(x1)
         np.set_printoptions(threshold=sys.maxsize)
         print(nums)
```

CONCLUSION

- **Key Findings and Conclusions of the Study**

  Linear regression models assume that the relationship between a dependent continuous variable Y and one or more explanatory (independent) variables X is linear (that is, a straight line). It's used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog).

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

- This dataset is Linear Regression in nature, we can verify data by using read method & get stats related information for each column using describe method.
- Visualizations, Pre-processing and Data Cleaning part was very crucial as without all these all method we were not able to judge the data effectively and won't be able to remove the outliers, handling null values and adding into the errors.
- Data contains numerical as well as categorical variable. So we handled them accordingly
- Check the r2 score using Mean absolute error, Mean squared error & get root mean squared error score.
- Train data using Linear Regression models to get the best score & finalise best score giver model for this dataset.
- Get the test score for same model.
- Save file using joblib library.

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

  Visualizations helped a lot in finding out those outliers values and helped in finding out the features having direct relation between the feature and the label.

Its always good to to have complete data while performing model but 7-8 % of data can be excluded based on performance impact.

.