House Pricing

House Price deals with various factors:-

- Its is not easy to buy a House
- House price depend on Area
- Neighbors
- No of Rooms
- No of Stories and many more factor comes while buying house

I have loaded the Basic Libraries and make two variable for test and train to store dataset.
In Train training data stored and in test test data stored.

To show all rows and columns I have used pd.set_option('diplay.max_columns',None) -> show all columns and pd.set_option('diplay.max_rows',None) -> show all rows

IMPORT BASIC LIBRARIES TO PERFROM

In [2655]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.preprocessing import StandardScaler
```

LOAD DATA SET AND SEE THE INFORMATION OF DATASET

In [2656]:
```
train_df = pd.read_csv('housepricetrain.csv')
test_df =pd.read_csv('housepricetest.csv')
```

To DISPLAY ALL THE COLUMNS AND ROWS

In [2657]:
```
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

Training and Test data

```
In [2658]: train_df
```

BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical	1stFlrSF	2
Gd	TA	No	ALQ	120	Unf	0	958	1078	GasA	TA	Y	SBrkr	958	
TA	Gd	Gd	ALQ	351	Rec	823	1043	2217	GasA	Ex	Y	SBrkr	2217	
Gd	TA	Av	GLQ	862	Unf	0	255	1117	GasA	Ex	Y	SBrkr	1127	
Gd	TA	No	BLQ	705	Unf	0	1139	1844	GasA	Ex	Y	SBrkr	1844	
Gd	TA	No	ALQ	1246	Unf	0	356	1602	GasA	Gd	Y	SBrkr	1602	
Gd	TA	Av	Unf	0	Unf	0	879	879	GasA	Ex	Y	SBrkr	879	
Gd	TA	No	ALQ	1302	Unf	0	90	1392	GasA	TA	Y	SBrkr	1392	
TA	TA	No	Rec	168	BLQ	682	284	1134	GasA	Ex	Y	SBrkr	1803	
TA	TA	No	ALQ	698	GLQ	96	420	1214	GasA	Ex	Y	SBrkr	1214	
TA	TA	No	Rec	442	Unf	0	390	832	GasA	TA	Y	SBrkr	832	
TA	TA	No	Unf	0	Unf	0	780	780	GasA	TA	Y	SBrkr	780	

```
In [2659]: test_df
```

BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical	1stFlrSF	2
Ex	TA	Gd	GLQ	1249	Unf	0	673	1922	GasA	Ex	Y	SBrkr	1922	
Gd	TA	Av	GLQ	1036	Unf	0	184	1220	GasA	Gd	Y	SBrkr	1360	
Gd	TA	Av	Unf	0	Unf	0	1753	1753	GasA	Ex	Y	SBrkr	1788	
TA	TA	No	Rec	275	Unf	0	429	704	GasA	Ex	Y	SBrkr	860	
Gd	TA	Mn	Unf	0	Unf	0	894	894	GasA	Ex	Y	SBrkr	894	
Gd	TA	Av	BLQ	131	GLQ	499	0	630	GasA	Gd	Y	SBrkr	630	
Gd	TA	Gd	GLQ	547	Unf	0	0	547	GasA	Gd	Y	SBrkr	1072	
Ex	TA	Gd	GLQ	1400	Unf	0	310	1710	GasA	Ex	Y	SBrkr	1710	
Gd	TA	Gd	GLQ	1518	Unf	0	0	1518	GasA	Gd	Y	SBrkr	1644	
Ex	TA	No	GLQ	866	Unf	0	338	1204	GasA	Ex	Y	SBrkr	1204	
TA	TA	No	Unf	0	Unf	0	520	520	GasA	Fa	N	SBrkr	520	

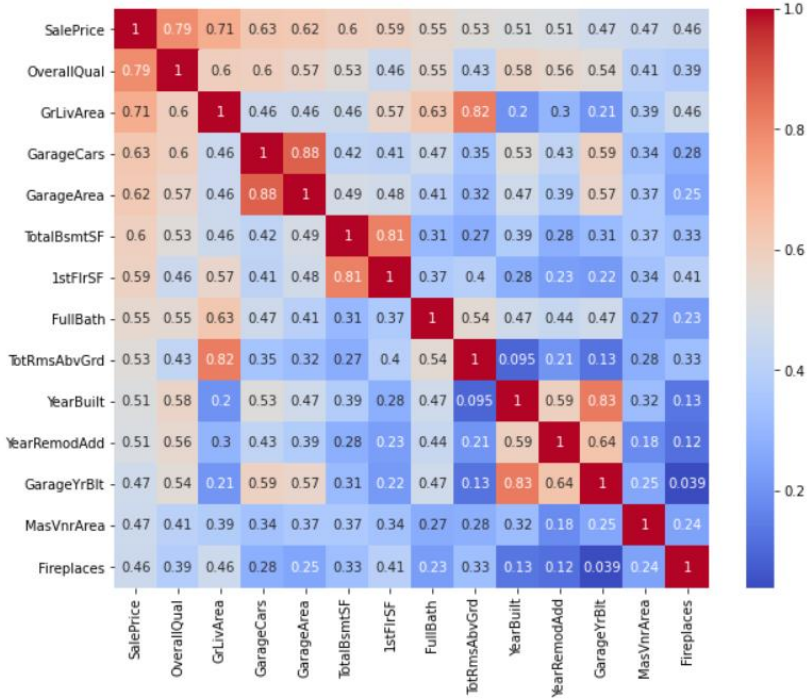
EDA – In EDA I have describe the data set .show its shape ,Data types, Missing value , Statistical information of dataset, Heatmap to show correlation and other visualization techniques like Boxplot to detect outliers ,Bar plot to show the Relation between price with independent columns.

NOW I WILL CHECK THE SHAPE AND SIZE OF DATASET

```
[2660]: train_df.shape
[2660]: (1168, 81)
```

```
[2661]: test_df.shape
[2661]: (292, 80)
```

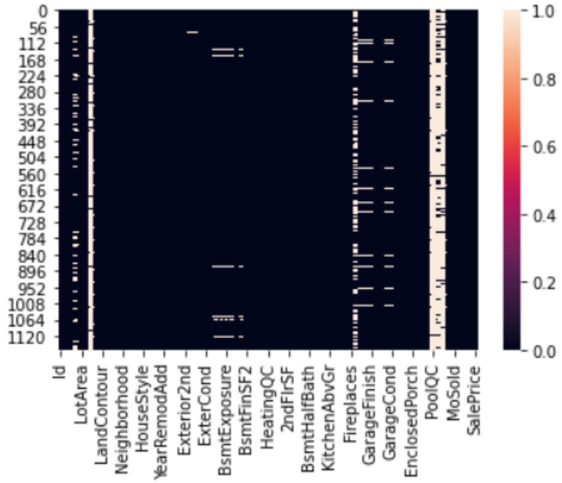
```
plt.figure(figsize=(10,8))
sns.heatmap(train_df[cols].corr(),annot=True,cmap='coolwarm')
plt.show()
```




```
]: train_df.dtypes
]: Id               int64
   MSSubClass       int64
   MSZoning        object
   LotFrontage    float64
   LotArea          int64
   Street          object
   Alley           object
   LotShape        object
   LandContour     object
   Utilities       object
```

SHOW THE NULL VALUE IN HEATMAP

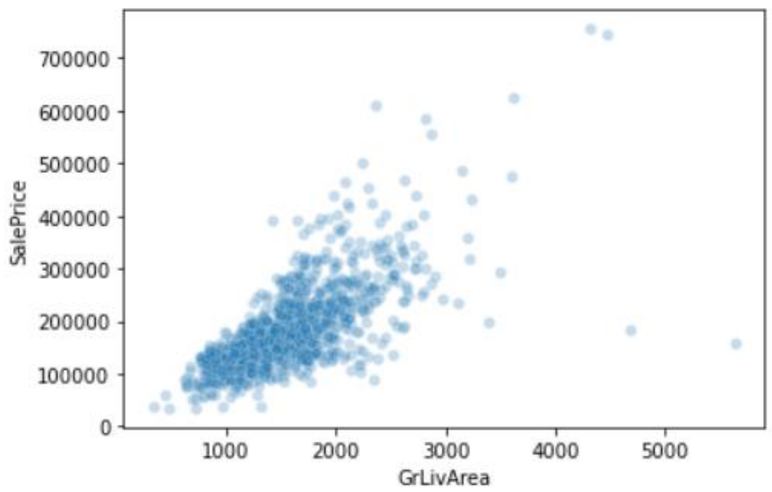
```
: sns.heatmap(train_df.isnull())
: <AxesSubplot:>
```



Scatter Plot to show the outliers and bonding between both columns

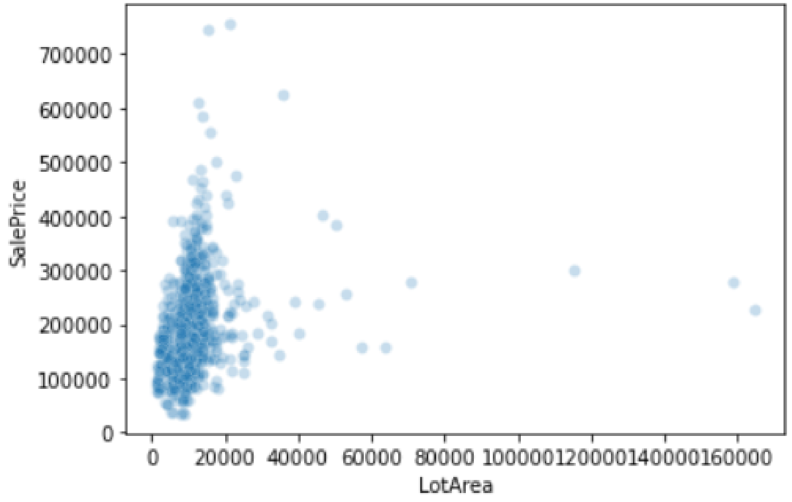
```
## ALpha used to show the transparency

sns.scatterplot(x=train_df['GrLivArea'],y=train_df['SalePrice'],alpha=0.25
```

```
<AxesSubplot:xlabel='GrLivArea', ylabel='SalePrice'>
```




```
sns.scatterplot(x=train_df['LotArea'],y=train_df['SalePrice'],alpha=0.25)
```

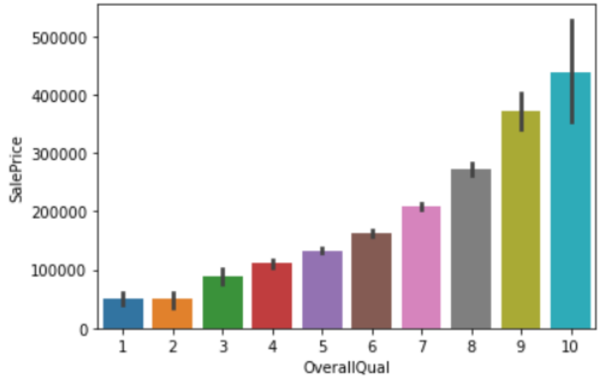
```
<AxesSubplot:xlabel='LotArea', ylabel='SalePrice'>
```



Bar Plot to show the variation in price

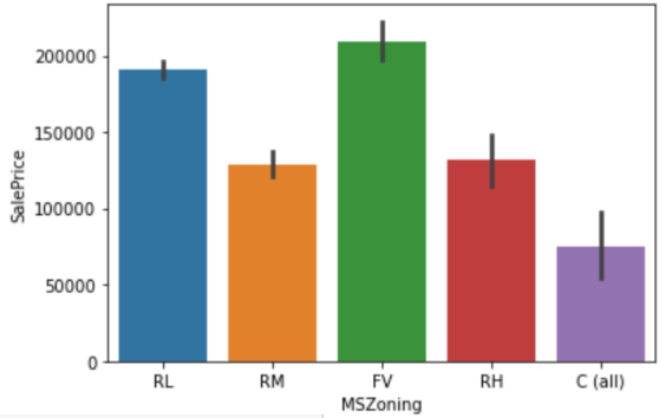
75]:

```
#### Bar Plot
sns.barplot(x=train_df['OverallQual'],y=train_df['SalePrice'])
plt.show()
```



]:

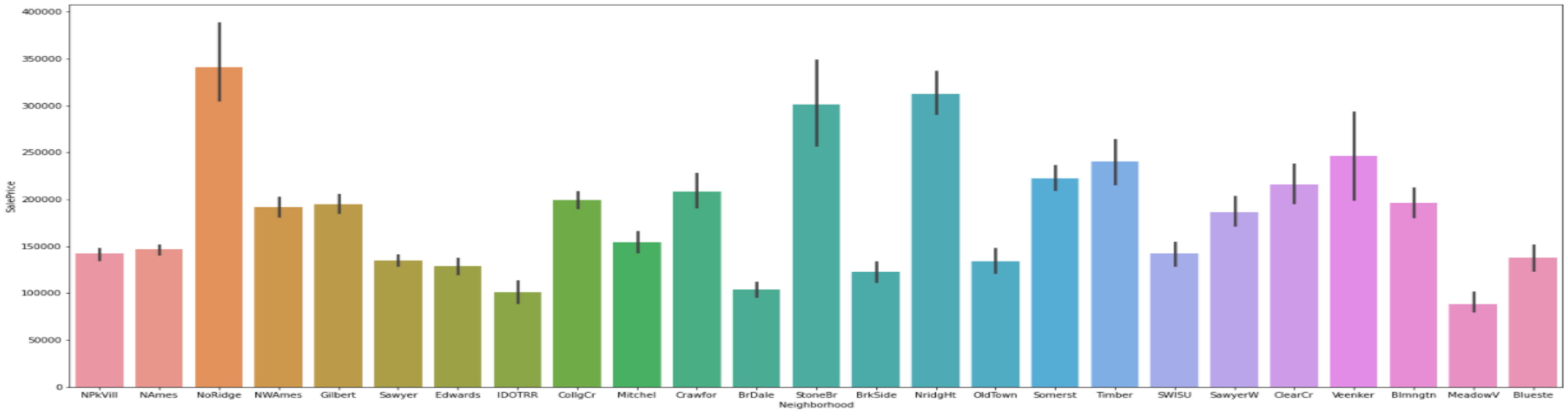
```
sns.barplot(x=train_df['MSZoning'],y=train_df['SalePrice'])
plt.show()
```



7]:

```
plt.figure(figsize=(25,10))
sns.barplot(x=train_df['Neighborhood'],y=train_df['SalePrice'])
plt.show
```

7]: <function matplotlib.pyplot.show(close=None, block=None)>



Pre-Processing Techniques used to handle the missing vale,outliers,skewness and many more techniques used. Pre-Processing is most important part of dataset,drop the one pair if highly corelated.

Pre-Processing

In [2693]:
```
# As per heatmap which shows th corelation among all field , drop one pair which are highly corerelated
df = train_df.drop(['GarageCars','YearRemodAdd','GarageYrBlt'],axis=1)
```

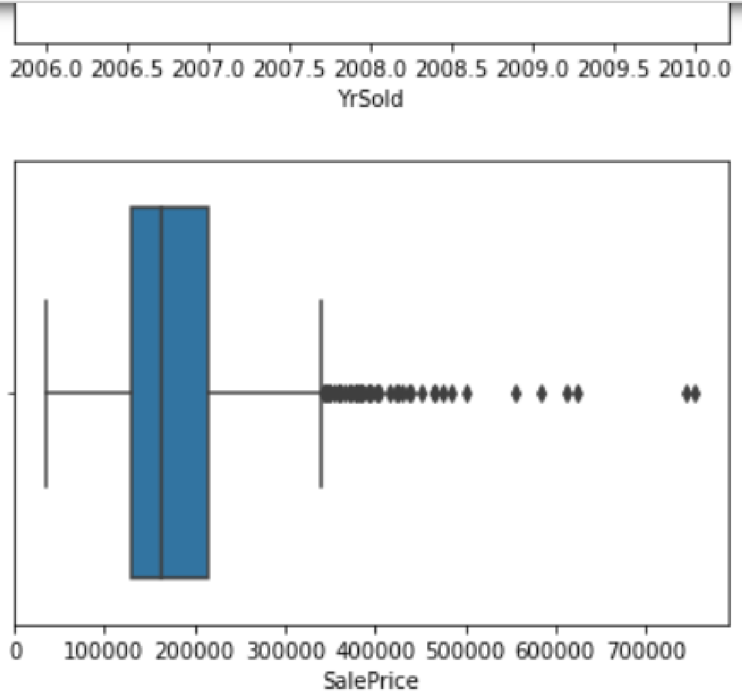
In [2694]:
```
df
```

WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	205	0	0	0	0	NaN	NaN	NaN	0	2	2007	WD	Normal	128000
81	207	0	0	224	0	NaN	NaN	NaN	0	10	2007	WD	Normal	268000
180	130	0	0	0	0	NaN	NaN	NaN	0	6	2007	WD	Normal	269790
0	122	0	0	0	0	NaN	MnPrv	NaN	0	1	2010	COD	Normal	190000
240	0	0	0	0	0	NaN	NaN	NaN	0	6	2009	WD	Normal	215000
100	17	0	0	0	0	NaN	NaN	NaN	0	11	2006	New	Partial	219210
0	0	0	0	95	0	NaN	NaN	NaN	0	5	2010	WD	Normal	121500
0	0	0	0	0	0	NaN	GdPrv	NaN	0	1	2006	WD	Normal	155000
0	0	184	0	0	0	NaN	GdPrv	Shed	400	4	2007	WD	Normal	140000
158	0	102	0	0	0	NaN	NaN	NaN	0	10	2008	COD	Abnorml	118500

Boxplot to detect the outliers.Use for loop to get all at once and apply filter as per view

[2710]:

```
#### check the presence of outliers
for i in count_col:
    sns.boxplot(df[i])
    plt.show()
```



Here I check the Skewness

```
2712]: df.skew()

2712]: Id               0.026526
       MSSubClass       1.422019
       LotFrontage      2.710383
       LotArea         10.659285
       OverallQual      0.175082
       OverallCond      0.580714
       YearBuilt       -0.579204
       MasVnrArea       2.834658
       BsmtFinSF1       1.871606
       BsmtFinSF2       4.365829
       BsmtUnfSF        0.909057
       TotalBsmtSF      1.744591
       1stFlrSF         1.513707
       2ndFlrSF         0.823479
       LowQualFinSF     8.666142
       GrLivArea        1.449952
       BsmtFullBath     0.627106
       BsmtHalfBath     4.264403
       FullBath         0.057809
       HalfBath         0.656492
       BedroomAbvGr     0.243855
       KitchenAbvGr     4.365259
       TotRmsAbvGrd     0.644657
       Fireplaces       0.671966
       GarageArea       0.189665
       WoodDeckSF       1.504929
```

Handling the outliers. Create one function to handle the outliers

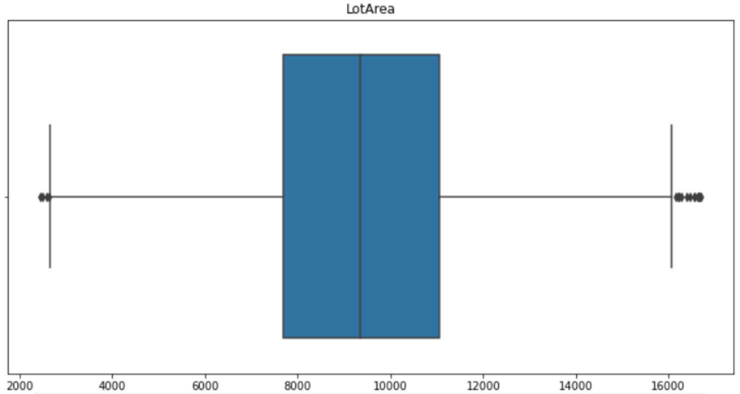
```
2713]: #### lets handle the outlier first
def outliers(s):
    iqr = (np.quantile(s, 0.75))-(np.quantile(s, 0.25))
    upper_bound = np.quantile(s, 0.75)+(1.5*iqr)
    lower_bound = np.quantile(s, 0.25)-(1.5*iqr)
    f = []
    for i in s:
        if i > upper_bound:
            f.append(i)
        elif i < lower_bound:
            f.append(i)
    sums = len(f)
    pros = len(f)/len(s)*100
    d = {'IQR':iqr,
         'Upper Bound':upper_bound,
        'Lower Bound':lower_bound,
        'Sum outliers': sums,'percentage outliers':pros}
    d = pd.DataFrame(d.items(),columns = ['sub','values'])
    return(d)

outliers(df.LotFrontage)
```

2713]:

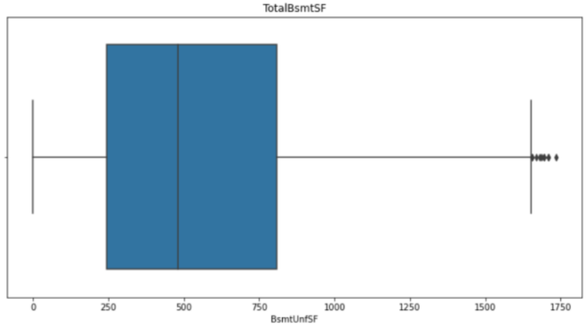
	sub	values
0	IQR	19.250000
1	Upper Bound	108.125000
2	Lower Bound	31.125000
3	Sum outliers	82.000000
4	percentage outliers	7.020548

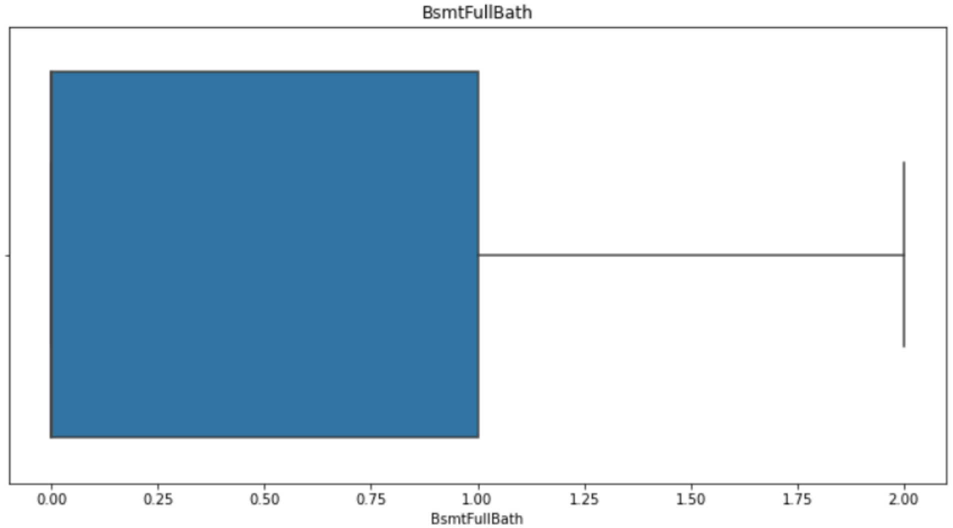
Showing graph after handle the outliers

```
#drop outliers
s = df['LotArea']
iqr = (np.quantile(s, 0.75))-(np.quantile(s, 0.25))
upper_bound = np.quantile(s, 0.75)+(1.5*iqr)
lower_bound = np.quantile(s, 0.25)-(1.5*iqr)
df = df[(df['LotArea'] <= upper_bound)]
df = df[(df['LotArea'] >= lower_bound)]
fig,ax = plt.subplots(figsize=(12,6))
fig = sns.boxplot(df.LotArea).set_title('LotArea')
```




```
#drop outliers
s = df['TotalBsmtSF']
iqr = (np.quantile(s, 0.75))-(np.quantile(s, 0.25))
upper_bound = np.quantile(s, 0.75)+(1.5*iqr)
lower_bound = np.quantile(s, 0.25)-(1.5*iqr)
df = df[(df['TotalBsmtSF'] <= upper_bound)]
df = df[(df['TotalBsmtSF'] >= lower_bound)]
fig,ax = plt.subplots(figsize=(12,6))
fig = sns.boxplot(df.BsmtUnfSF).set_title('TotalBsmtSF')
```





Apply Label Encoder for Categorical Columns and standard scaler for continuous columns

```
Le = LabelEncoder()
```

```
In [2741]: for i in columns:
               df[i] = Le.fit_transform(df[i])
```

```
In [2742]: ### for continous columns i will aplly standard scaler to normalize
           from sklearn.preprocessing import StandardScaler
           sc = StandardScaler()
```

```
In [2743]: df[count_col].columns
```

```
Out[2743]: Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
                  'OverallCond', 'YearBuilt', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
                  'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
                  'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
                  'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
                  'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
                  'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
                 dtype='object')
```

```
In [2744]: target_col = df[count_col]['SalePrice'];
           target_col
```

```
In [2748]: ### Now i dropped the unscaled data from df
           df = df.drop(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
                  'OverallCond', 'YearBuilt', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
                  'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
                  'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
                  'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
                  'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
                  'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],axis=1)
```

```
In [2749]: df_independent = df_independent.drop(['SalePrice'],axis=1)
```

```
In [2753]: df_cat = df
```

```
In [2754]: df_cat
```

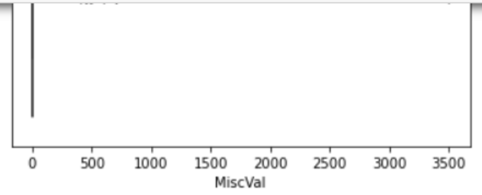
2	Heating	HeatingQC	CentralAir	Electrical	KitchenQual	Functional	FireplaceQu	GarageType	GarageFinish	GarageQual	GarageCond	PavedDrive	SaleType	SaleCondition
5	0	4	1	2	3	5	4	1	1	3	4	2	8	4
5	0	0	1	2	3	5	4	1	2	3	4	2	8	4
5	0	0	1	2	3	5	4	1	1	3	4	2	0	4
5	0	2	1	2	2	5	4	1	0	3	4	2	8	4
5	0	0	1	2	2	5	2	3	0	3	4	2	6	5
5	0	4	1	2	3	4	2	5	2	3	4	2	8	4
1	0	0	1	2	3	0	4	1	1	3	4	2	8	4
2	0	0	1	2	3	5	2	5	2	1	1	2	8	4
5	0	4	1	2	3	5	2	5	2	3	4	2	0	0

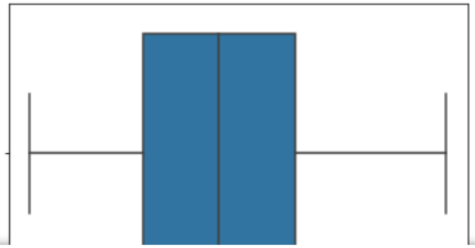
Same Techniques Applied with testing of Data

```
'65]: test_df.isnull().sum()
```

```
'65]: Id              0
      MSSubClass      0
      MSZoning        0
      LotFrontage    45
      LotArea         0
      Street          0
      Alley         278
      LotShape        0
      LandContour     0
```

```
5]: #### check for the ouliers
    for i in tcount_col:
        sns.boxplot(test_df[i])
        plt.show()
```





```
SaleCondition      0
dtype: int64
```

```
2770]: #### Apply Mode for catagical columns to fill missing
       for i in test_columns:
           test_df[i] = test_df[i].fillna(test_df[i].mode()[0])
```

```
2771]: test_df[columns].isnull().sum()
```

```
2771]: MSZoning        0
       Street          0
       LotShape        0
       LandContour     0
       Utilities       0
       LotConfig       0
       LandSlope       0
       Neighborhood    0
       Condition1      0
```

Finally Prepare model with training dataset

Model Building It will be done with train data set

```
0]: from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
rf=RandomForestRegressor()
dtc = DecisionTreeRegressor()
lr=LinearRegression()
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

Find the best accuracy and random state.
Use Train_test_split for split the data into training and testing

]:
```
maxAcc=0
maxRs=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=i)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
   # print(f"At Random State {i},the tarining accuracy is :- ",{r2_score(y_train,pred_train)})
   # print(f"At Random State {i},the Test accuracy is :- ",{r2_score(y_test,pred_test)})
    accu = r2_score(y_test,pred_test)
    if accu>maxAcc:
        maxAcc=accu
        maxRs=i
print("Best accuracy -",maxAcc,'Best Random state = ',maxRs)
```

```
Best accuracy - 0.733865117916287 Best Random state =  127
```

]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=127)
```

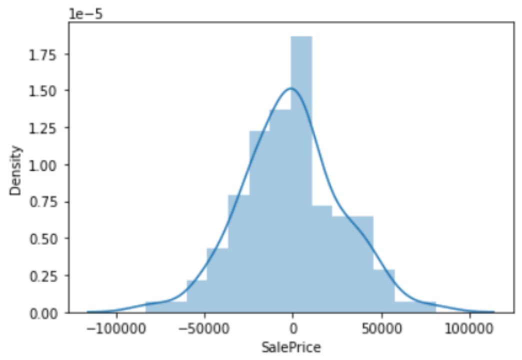
Define Function to Prepare model which shows how model is fit the data and predict the data

```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
def predict(ml_model):
    print('Model is : {}'.format(ml_model))
    model = ml_model.fit(x_train,y_train)
    print("Training Score : {}".format(model.score(x_train,y_train)))
    predictions = model.predict(x_test)
    print("Predictions are : {}" ,format(predictions))
    print('\n')
    print('Prediction')
    r2score = r2_score(y_test,predictions)
    print("r2 Score is : {}",format(r2score))
    print('Cross Validation Score: {}'.format(cross_val_score(ml_model,x_train,y_train,cv=5,scoring='r2')))
    print('MAE: {}'.format(mean_absolute_error(y_test,predictions)))
    print('MSE: {}'.format(mean_squared_error(y_test,predictions)))
    print('RMSE: {}'.format(np.sqrt(mean_squared_error(y_test,predictions))))
    print('\n')

    sns.distplot(y_test-predictions)
```

Linear Regression

```
[2828]: predict(LinearRegression())

Model is : LinearRegression()
Training Score : 0.6772130643170668
Predictions are : {} [187315.10534729 131360.04057653 140546.32677218 147113.35814702
 181950.21194224 211616.22876986 111123.8698592  145609.82531945
 194313.79143785 163714.52402499 166507.20309974 311262.45299612
 284336.90283542 154784.89187193 217188.21728718 139134.82963924
 125355.55699753 209317.93636953  93866.38143591 159285.60262996
 238732.39874496 109609.03242809 153481.32639851  94841.85307386
 113541.62562783 195762.69460042 167099.77469893 132633.29105407
 259557.74704459 203688.67962483 201420.92893756 187559.32077718
 135063.52745758 200374.30687608 211925.75598388 159338.36247464
 168292.91797257 205477.0865882  129883.42138626 168930.21655682
 179720.80639264 203275.05798033 247991.09757666 143656.9039847
 177794.93751635  75447.38593333 145373.71293003 143906.58459139
 188404.55612137 192804.66198754 163053.84755183 220500.73218794
 177420.72824878 258908.97604776 144121.18751135 142336.32380227
 226672.92488563 148634.10218277 159321.68314725 149155.01676384
 256567.90379413 154130.24122409 146188.27294281 173971.63760231
```

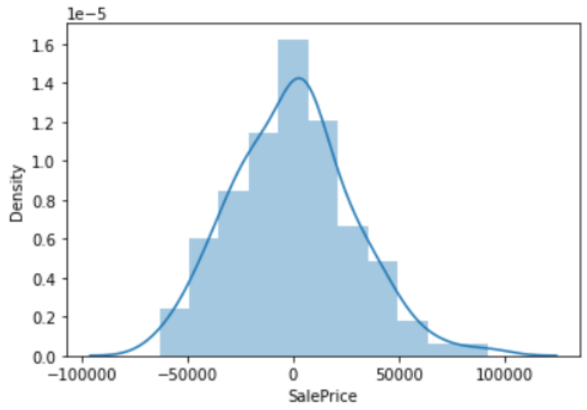
```
Prediction
r2 Score is : {} 0.733865117916287
Cross Validation Score: [ 5.98611041e-01  4.76057132e-01  5.80157608e-01 -7.08808887e+22
  5.11166869e-01]
MAE: 21300.984119480523
MSE: 776201591.9922227
RMSE: 27860.394684789062
```



Gradient Boosting

```
]:
predict(GradientBoostingRegressor())

Model is : GradientBoostingRegressor()
Training Score : 0.9148994937625885
Predictions are : {} [178118.73886236 129990.3030386  126607.95985099 156399.36300754
 188817.07787524 199985.94322473 128143.04939122 155690.76242109
 218737.41269257 152148.76216384 136471.07551198 328508.01274428
 274593.79494782 130785.09488949 213041.75592028 115586.67374067
 123704.36667408 172377.14376857 108521.47242745 142022.96021371
 230450.56436954 130473.62652332 170107.17675535 107833.39350721
 105481.6445501  185751.04955831 134475.6969403  141778.00033627
 262383.51972905 190558.54822857 197264.19157906 152827.24758086
 119071.06227471 220964.344729   220152.79589566 139717.4020219
 143794.69765723 216828.72755176 124435.66721043 150881.87718666
 160133.52711743 199685.95844305 244755.99001419 103748.76236677
 180393.91677673 115636.30205044 151999.33692925 194265.12195869
 182520.03780129 195495.70220729 172879.05789627 202565.17253892
 142501.77702214 266025.95536066 151141.69596608 130299.87839528
 226214.88231403 134916.54828776 165523.73288524 145743.97602537
 252852.123369   143609.51474458 109737.49615184 148550.36824742
 140392.47191963 148741.35737724 105523.32630949 254459.46459535
```

```
Prediction
r2 Score is : {} 0.7286341985814417
Cross Validation Score: [0.64425705 0.66195954 0.57036998 0.33891489 0.67840654]
MAE: 21945.45822033898
MSE: 791457945.7760633
RMSE: 28132.8623814937
```



Hyper Parameter Tune

50000 100000 150000 200000 250000 300000
Actual Charges

11]:

```
from sklearn.model_selection import GridSearchCV
```

12]:

```
parameters = {
    "n_estimators":[5,50,250,500],
    "max_depth":[1,3,5,7,9],
    "learning_rate":[0.01,0.1,1,10,100]
}
```

13]:

```

GCV=GridSearchCV(GradientBoostingRegressor(),parameters,cv=5)
GCV.fit(x_train,y_train)

GCV.best_params_
```

13]:

```
{'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 250}
```

14]:

```

Final_model=GradientBoostingRegressor(learning_rate=0.1,max_depth=5,n_estimators=250)
Final_model.fit(x_train,y_train)
pred=Final_model.predict(x_test)
accuracy = r2_score(y_test,pred)
print(accuracy*100)
```

```
71.97978855238661
```

Conclusion

- Finally we get the r score for best model and then we save it.
- I have used different techniques to get the r score,cross validation score,RMSE,MSE,MAE.
- To get better Accuracy use hyper parameter tunning.
- Also used test data set to predict with tarining dataset.
- There is also a chance of improvement to find more better accuracy .

Thanks You