In [2]:

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
from sklearn.svm import SVR
from lightgbm import LGBMRegressor
from mlxtend.regressor import StackingRegressor
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GridSearchCV
from sklearn.metrics import mean_absolute_error
pd.pandas.set_option("display.max_columns",None)
print("all necessary libraries are imported")
```

```
all necessary libraries are imported
```

In [3]:

```
train=pd.read_csv('C:\\Users\\Deeksha Rai\\Desktop\\projects\\train.csv')
test=pd.read_csv('C:\\Users\\Deeksha Rai\\Desktop\\projects\\test.csv')
```

In [4]:

```
train.shape,test.shape
```

Out[4]:

```
((1460, 81), (1459, 80))
```

In [5]:

```
test.tail()
```

Out[5]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
1454	2915	160	RM	21.0	1936	Pave	NaN	Reg	L
1455	2916	160	RM	21.0	1894	Pave	NaN	Reg	L
1456	2917	20	RL	160.0	20000	Pave	NaN	Reg	L
1457	2918	85	RL	62.0	10441	Pave	NaN	Reg	L
1458	2919	60	RL	74.0	9627	Pave	NaN	Reg	L

In [6]:

```
y=train['SalePrice'].values
train.drop('SalePrice',axis=1,inplace=True)
train.head()
```

Out[6]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Ut
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	/
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	/
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	/
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	/
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	/

In [7]:

```
train.shape
```

Out[7]:

```
(1460, 80)
```

In [8]:

```
# checking null in training data
```

In [9]:

```
col_train=list(train.columns)
```

In [10]:

```
for feature in col_train:
    if(train[feature].isnull().any()):
        print(f'{feature} : {train[feature].isnull().sum()}')
    else:
         print(f'{feature} : {0}')
```

```
Id : 0
MSSubClass : 0
MSZoning : 0
LotFrontage : 259
LotArea : 0
Street : 0
Alley : 1369
LotShape : 0
LandContour : 0
Utilities : 0
LotConfig : 0
LandSlope : 0
Neighborhood : 0
Condition1 : 0
Condition2 : 0
BldgType : 0
HouseStyle : 0
OverallQual : 0
OverallCond : 0
YearBuilt : 0
YearRemodAdd : 0
RoofStyle : 0
RoofMatl : 0
Exterior1st : 0
Exterior2nd : 0
MasVnrType : 8
MasVnrArea : 8
ExterQual : 0
ExterCond : 0
Foundation : 0
BsmtQual : 37
BsmtCond : 37
BsmtExposure : 38
BsmtFinType1 : 37
BsmtFinSF1 : 0
BsmtFinType2 : 38
BsmtFinSF2 : 0
BsmtUnfSF : 0
TotalBsmtSF : 0
Heating : 0
HeatingQC : 0
CentralAir : 0
Electrical : 1
1stFlrSF : 0
2ndFlrSF : 0
LowQualFinSF : 0
GrLivArea : 0
BsmtFullBath : 0
BsmtHalfBath : 0
FullBath : 0
HalfBath : 0
```

```
BedroomAbvGr : 0
KitchenAbvGr : 0
KitchenQual : 0
TotRmsAbvGrd : 0
Functional : 0
Fireplaces : 0
FireplaceQu : 690
GarageType : 81
GarageYrBlt : 81
GarageFinish : 81
GarageCars : 0
GarageArea : 0
GarageQual : 81
GarageCond : 81
PavedDrive : 0
WoodDeckSF : 0
OpenPorchSF : 0
EnclosedPorch : 0
3SsnPorch : 0
ScreenPorch : 0
PoolArea : 0
PoolQC : 1453
Fence : 1179
MiscFeature : 1406
MiscVal : 0
MoSold : 0
YrSold : 0
SaleType : 0
SaleCondition : 0
```

In [11]:

```
# checking null in testing data
col_test=list(test.columns)
for feature in col_test:
    if(test[feature].isnull().any()):
        print(f'{feature} : {test[feature].isnull().sum()}')
    else:
         print(f'{feature} : {0}')
```

```
Id : 0
MSSubClass : 0
MSZoning : 4
LotFrontage : 227
LotArea : 0
Street : 0
Alley : 1352
LotShape : 0
LandContour : 0
Utilities : 2
LotConfig : 0
LandSlope : 0
Neighborhood : 0
Condition1 : 0
Condition2 : 0
BldgType : 0
HouseStyle : 0
OverallQual : 0
OverallCond : 0
YearBuilt : 0
YearRemodAdd : 0
RoofStyle : 0
RoofMatl : 0
Exterior1st : 1
Exterior2nd : 1
MasVnrType : 16
MasVnrArea : 15
ExterQual : 0
ExterCond : 0
Foundation : 0
BsmtQual : 44
BsmtCond : 45
BsmtExposure : 44
BsmtFinType1 : 42
BsmtFinSF1 : 1
BsmtFinType2 : 42
BsmtFinSF2 : 1
BsmtUnfSF : 1
TotalBsmtSF : 1
Heating : 0
HeatingQC : 0
CentralAir : 0
Electrical : 0
1stFlrSF : 0
2ndFlrSF : 0
LowQualFinSF : 0
GrLivArea : 0
BsmtFullBath : 2
BsmtHalfBath : 2
FullBath : 0
```

```
HalfBath : 0
BedroomAbvGr : 0
KitchenAbvGr : 0
KitchenQual : 1
TotRmsAbvGrd : 0
Functional : 2
Fireplaces : 0
FireplaceQu : 730
GarageType : 76
GarageYrBlt : 78
GarageFinish : 78
GarageCars : 1
GarageArea : 1
GarageQual : 78
GarageCond : 78
PavedDrive : 0
WoodDeckSF : 0
OpenPorchSF : 0
EnclosedPorch : 0
3SsnPorch : 0
ScreenPorch : 0
PoolArea : 0
PoolQC : 1456
Fence : 1169
MiscFeature : 1408
MiscVal : 0
MoSold : 0
YrSold : 0
SaleType : 1
SaleCondition : 0
```

In [12]:

```
# concatenating the train and test data to remove null values
data=pd.concat([train,test],axis=0,ignore_index=True)
data.tail()
```

Out[12]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
2914	2915	160	RM	21.0	1936	Pave	NaN	Reg	l
2915	2916	160	RM	21.0	1894	Pave	NaN	Reg	l
2916	2917	20	RL	160.0	20000	Pave	NaN	Reg	l
2917	2918	85	RL	62.0	10441	Pave	NaN	Reg	l
2918	2919	60	RL	74.0	9627	Pave	NaN	Reg	l

In [13]:

```
# checking null values in test+train data
col_train_test=list(data.columns)
for feature in col_train_test:
    if(data[feature].isnull().any()):
        print(f'{feature} : {data[feature].isnull().sum()}')
```

```
MSZoning : 4
LotFrontage : 486
Alley : 2721
Utilities : 2
Exterior1st : 1
Exterior2nd : 1
MasVnrType : 24
MasVnrArea : 23
BsmtQual : 81
BsmtCond : 82
BsmtExposure : 82
BsmtFinType1 : 79
BsmtFinSF1 : 1
BsmtFinType2 : 80
BsmtFinSF2 : 1
BsmtUnfSF : 1
TotalBsmtSF : 1
Electrical : 1
BsmtFullBath : 2
BsmtHalfBath : 2
KitchenQual : 1
Functional : 2
FireplaceQu : 1420
GarageType : 157
GarageYrBlt : 159
GarageFinish : 159
GarageCars : 1
GarageArea : 1
GarageQual : 159
GarageCond : 159
PoolQC : 2909
Fence : 2348
MiscFeature : 2814
SaleType : 1
```

In [14]:

```
col_num_feature=data.select_dtypes(exclude='object')
col_num_feature=col_num_feature.columns
col_num_feature=list(col_num_feature)
col_num_feature
```

Out[14]:

```
['Id',
 'MSSubClass',
 'LotFrontage',
 'LotArea',
 'OverallQual',
 'OverallCond',
 'YearBuilt',
 'YearRemodAdd',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'BedroomAbvGr',
 'KitchenAbvGr',
 'TotRmsAbvGrd',
 'Fireplaces',
 'GarageYrBlt',
 'GarageCars',
 'GarageArea',
 'WoodDeckSF',
 'OpenPorchSF',
 'EnclosedPorch',
 '3SsnPorch',
 'ScreenPorch',
 'PoolArea',
 'MiscVal',
 'MoSold',
 'YrSold']
```

In [15]:

```
col_obj_feature=data.select_dtypes(include='object')
col_obj_feature=col_obj_feature.columns
col_obj_feature
```

Out[15]:

```
Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilitie
s',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType
2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
       'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQua
l',
       'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType', 'SaleCondition'],
      dtype='object')
```

In [16]:

```
num_nan_feature=[]
for feature in col_num_feature:
    if(data[feature].isnull().any()):
        num_nan_feature.append(feature)
```

In [17]:

```
num_nan_feature
```

Out[17]:

```
['LotFrontage',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 'BsmtFullBath',
 'BsmtHalfBath',
 'GarageYrBlt',
 'GarageCars',
 'GarageArea']
```

In [18]:

```
# replace all the null values with 0
for feature in num_nan_feature:
    data[feature].fillna(value=0,inplace=True)
```

In [19]:

```
obj_nan_feature=[]
for feature in col_obj_feature:
    if(data[feature].isnull().any()):
        obj_nan_feature.append(feature)
```

In [20]:

```
obj_nan_feature
```

Out[20]:

```
['MSZoning',
 'Alley',
 'Utilities',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Electrical',
 'KitchenQual',
 'Functional',
 'FireplaceQu',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PoolQC',
 'Fence',
 'MiscFeature',
 'SaleType']
```

In [21]:

```
# replace all the null values with 'missing'
for feature in obj_nan_feature:
    data[feature].fillna(value='missing',inplace=True)
```

In [22]:

```
dummy=pd.get_dummies(data[col_obj_feature],prefix=col_obj_feature)
dummy
```

Out[22]:

	MSZoning_C (all)	MSZoning_FV	MSZoning_RH	MSZoning_RL	MSZoning_RM	MSZoning_miss
0	0	0	0	1	0	
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	0	0	1	0	
4	0	0	0	1	0	
...	...	...	...	...	...	
2914	0	0	0	0	1	
2915	0	0	0	0	1	
2916	0	0	0	1	0	
2917	0	0	0	1	0	
2918	0	0	0	1	0	

2919 rows × 275 columns

In [23]:

```
data.drop(col_obj_feature,axis=1,inplace=True)
```

In [24]:

```
data_final=pd.concat([data,dummy],axis=1)
```

In [25]:

```
data_final.drop(['Id'],axis=1,inplace=True)
print(data_final.head())
```

```
   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  BsmtUnfSF  TotalBsmtS
F  \
0          2003       196.0       706.0         0.0      150.0        856.
0
1          1976         0.0       978.0         0.0      284.0       1262.
0
2          2002       162.0       486.0         0.0      434.0        920.
0
3          1970         0.0       216.0         0.0      540.0        756.
0
4          2000       350.0       655.0         0.0      490.0       1145.
```

In [26]:

```
# taking logarithm to remove skewness of data
```

In [27]:

```
# for feature in data_final.columns:
#     data_final[feature].hist(bins=30)
#     plt.show()
```

In [28]:

```
numeric_features=['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'Ye
          'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmt
          '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBat
          'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplac
          'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
          'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'Yr
```

In [29]:

```
# col_num_feature.remove('Id')
for feature in numeric_features :
    data_final[feature]=np.log(data_final[feature]+1)
print(data_final.head())
print(y.shape)
y=np.log(y+1)
print(y)
```

```
   MSSubClass  LotFrontage   LotArea  OverallQual  OverallCond  YearBuilt
\
0    4.110874     4.189655  9.042040     2.079442     1.791759   7.602900
1    3.044522     4.394449  9.169623     1.945910     2.197225   7.589336
2    4.110874     4.234107  9.328212     2.079442     1.791759   7.601902
3    4.262680     4.110874  9.164401     2.079442     1.791759   7.557995
4    4.110874     4.442651  9.565284     2.197225     1.791759   7.601402

   YearRemodAdd  MasVnrArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  TotalBsmtS
F  \
0      7.602900    5.283204    6.561031         0.0   5.017280     6.75343
8
1      7.589336    0.000000    6.886532         0.0   5.652489     7.14124
5
2      7.602401    5.093750    6.188264         0.0   6.075346     6.82546
0
3      7.586296    0.000000    5.379897         0.0   6.293419     6.62936
3
4      7.601402    5.860786    6.486161         0.0   6.196444     7.04403
```

In [30]:

```
data_final.shape
```

Out[30]:

```
(2919, 311)
```

In [31]:

```
for i in range(len(data_final.iloc[0,:])):
    p=[i+1,data_final.iloc[:,i].isnull().sum()]
    print(p)
```

```
[1, 0]
[2, 0]
[3, 0]
[4, 0]
[5, 0]
[6, 0]
[7, 0]
[8, 0]
[9, 0]
[10, 0]
[11, 0]
[12, 0]
[13, 0]
[14, 0]
[15, 0]
[16, 0]
[17, 0]
[18, 0]
[19, 0]
```

In [ ]:

In [32]:

```
# removing correlated features by one method
cor_matrix = data_final.corr().abs()
# print(cor_matrix)
upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),k=1).astype(np.bool))
# print(upper_tri)
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.95)]
print(); print(to_drop)
df1 = data_final.drop(to_drop, axis=1)
df1.shape
```

```
['GarageArea', 'Street_Pave', 'Exterior2nd_CmentBd', 'Exterior2nd_MetalSd',
'Exterior2nd_VinylSd', 'Exterior2nd_missing', 'MasVnrType_None', 'BsmtCond_m
issing', 'BsmtExposure_missing', 'BsmtFinType1_missing', 'BsmtFinType2_missi
ng', 'CentralAir_Y', 'FireplaceQu_missing', 'GarageType_missing', 'GarageFin
ish_missing', 'GarageQual_missing', 'GarageCond_missing', 'MiscFeature_missi
ng', 'SaleCondition_Partial']
```

Out[32]:

```
(2919, 292)
```

In [33]:

```
# removing correlated features by another method
def get_corr(da_ta,threshold):
    corr_col=set()
    corr_mat=da_ta.corr()
    for i in range(len(corr_mat.columns)):
        for j in range(i):
            if abs(corr_mat.iloc[i,j])>threshold:
                col_name=corr_mat.columns[i]
                corr_col.add(col_name)
    return corr_col
corr_features=get_corr(data_final,0.95)
dataset_final=data_final.drop(labels=corr_features,axis=1)
print(dataset_final.shape)
tr=dataset_final.iloc[0:1460,:]
print(tr.shape)
te=dataset_final.iloc[1460:2919,:]
```

```
(2919, 292)
(1460, 292)
```

In [34]:

```
scaling=StandardScaler()
data_s_tr = scaling.fit_transform(tr)
data_s_te=scaling.transform((te))
Data_s_tr=pd.DataFrame(data_s_tr)
Data_s_te=pd.DataFrame(data_s_te)
```

In [35]:

```
Parameters1=[{'reg_lambda': [0.4,0.5], 'reg_alpha': [0.9,1], 'n_estimators': [650,700], 'mi
             'max_depth': [4,3], 'learning_rate': [0.03,0.02], 'gamma': [0.00001,0.0001], '
               ]
scores1 = ['neg_mean_squared_error']

reg1 = GridSearchCV(xgb.XGBRegressor(), Parameters1, scoring='neg_root_mean_squared_error',
reg1.fit(Data_s_tr.iloc[:,:].values,y)


# print(reg1.best_params_)

y_pred11 = reg1.predict(Data_s_tr.iloc[:,:].values)
y_pred1 = np.exp(reg1.predict(Data_s_te.iloc[:,:].values)).round(2)
# print(mean_absolute_error(y_pred11,Y))

Parameters2=[{'num_leaves':[31,32], 'max_depth':[3,4], 'learning_rate':[0.1,0.2],
            'n_estimators':[500,400]}]
scores = ['neg_mean_squared_error']

reg2 = GridSearchCV(LGBMRegressor(), Parameters2, scoring='neg_root_mean_squared_error', ve
                          )
reg2.fit(Data_s_tr.iloc[:,:].values,y)


# print(reg2.best_params_)

y_pred12 = reg2.predict(Data_s_tr.iloc[:,:].values)
y_pred2 = np.exp(reg2.predict(Data_s_te.iloc[:,:].values)).round(2)
# print(mean_absolute_error(y_pred12,Y))



Parameters3=[{'cache_size': [185,180],
              'tol': [0.0011,0.0013], 'kernel': ['rbf'],'gamma': [0.00009,0.0001],'epsilon'
              }]
scores = ['neg_mean_squared_error']

reg3 = GridSearchCV(SVR(), Parameters3, scoring='neg_root_mean_squared_error', verbose=2, c
                          )
reg3.fit(Data_s_tr.iloc[:,:].values,y)


# print(reg3.best_params_)

y_pred13 = reg3.predict(Data_s_tr.iloc[:,:].values)
y_pred3 = np.exp(reg3.predict(Data_s_te.iloc[:,:].values)).round(2)
# print(mean_absolute_error(y_pred13,Y))
```

```
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
1; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
1; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
1; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
1; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
3; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
```

```
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
3; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
3; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
3; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=9e-05, kernel=rbf, tol=0.001
3; total time=   0.1s
[CV] END cache_size=180, epsilon=0.013, gamma=0.0001, kernel=rbf, tol=0.00
11; total time=   0.1s
```

In [37]:

```
params_stack={
               'lgbmregressor__learning_rate': [0.01,0.02], 'lgbmregressor__max_depth': [3,
               'lgbmregressor__n_estimators': [500,600], 'lgbmregressor__num_leaves': [4,3]

              'xgbregressor__max_depth': [4,6],
              'xgbregressor__reg_lambda': [0.4,0.3],
              'xgbregressor__reg_alpha': [0.9],
              'xgbregressor__n_estimators': [500],
              'xgbregressor__min_child_weight': [2.5],
              'xgbregressor__learning_rate': [0.01,0.03],
              'xgbregressor__gamma': [0.00001],
              'xgbregressor__booster': ['dart'], 'svr__C': [75],
              'svr__cache_size': [185],
              'svr__tol': [0.0011], 'svr__kernel': ['rbf'],'svr__gamma': [0.00009],'svr__ep
              'svr__degree': [4],
              'meta_regressor__C': [75],
              'meta_regressor__cache_size': [185],
              'meta_regressor__tol': [0.0011],
              'meta_regressor__kernel': ['rbf'],
              'meta_regressor__gamma': [0.00009],
              'meta_regressor__epsilon': [0.011],
              'meta_regressor__degree': [4]
        }
xg_boost=xgb.XGBRegressor()
s_vr=SVR()
lgbm=LGBMRegressor()

regs=[xg_boost,s_vr,lgbm]

stack_reg=StackingRegressor(regressors=regs, meta_regressor=s_vr)

stack_gen=GridSearchCV(stack_reg,params_stack,cv=5,refit=True,verbose=2)
stack_gen.fit(Data_s_tr.iloc[:,:].values,y)
y_pred14=stack_gen.predict(Data_s_tr.iloc[:,:].values)
y_pred4=np.exp(stack_gen.predict(Data_s_te.iloc[:,:].values)).round(2)
# print(stack_gen.best_params_)
# print(mean_absolute_error(y_pred14,Y))
```

```
Fitting 5 folds for each of 128 candidates, totalling 640 fits
[CV] END lgbmregressor__learning_rate=0.01, lgbmregressor__max_depth=3, lg
bmregressor__n_estimators=500, lgbmregressor__num_leaves=4, meta_regressor
__C=75, meta_regressor__cache_size=185, meta_regressor__degree=4, meta_reg
ressor__epsilon=0.011, meta_regressor__gamma=9e-05, meta_regressor__kernel
=rbf, meta_regressor__tol=0.0011, svr__C=75, svr__cache_size=185, svr__deg
ree=4, svr__epsilon=0.011, svr__gamma=9e-05, svr__kernel=rbf, svr__tol=0.0
011, xgbregressor__booster=dart, xgbregressor__gamma=1e-05, xgbregressor__
learning_rate=0.01, xgbregressor__max_depth=4, xgbregressor__min_child_wei
ght=2.5, xgbregressor__n_estimators=500, xgbregressor__reg_alpha=0.9, xgbr
egressor__reg_lambda=0.4; total time=   2.5s
[CV] END lgbmregressor__learning_rate=0.01, lgbmregressor__max_depth=3, lg
bmregressor__n_estimators=500, lgbmregressor__num_leaves=4, meta_regressor
__C=75, meta_regressor__cache_size=185, meta_regressor__degree=4, meta_reg
ressor__epsilon=0.011, meta_regressor__gamma=9e-05, meta_regressor__kernel
=rbf, meta_regressor__tol=0.0011, svr__C=75, svr__cache_size=185, svr__deg
ree=4, svr__epsilon=0.011, svr__gamma=9e-05, svr__kernel=rbf, svr__tol=0.0
011, xgbregressor__booster=dart, xgbregressor__gamma=1e-05, xgbregressor__
learning_rate=0.01, xgbregressor__max_depth=4, xgbregressor__min_child_wei
```

In [42]:

```
y_pred4
```

Out[42]:

```
array([118615.62, 161514.56, 191317.54, ..., 167925.1 , 120131.23,
       218755.48])
```

In [43]:

```
df_submission=pd.DataFrame({'Id':test['Id'],'Saleprice':y_pred4})
```

In [44]:

```
df_submission
```

Out[44]:

	Id	Saleprice
0	1461	118615.62
1	1462	161514.56
2	1463	191317.54
3	1464	198507.07
4	1465	186223.99
...	...	...
1454	2915	84437.80
1455	2916	81827.28
1456	2917	167925.10
1457	2918	120131.23
1458	2919	218755.48

1459 rows × 2 columns

In [45]:

```
df_submission.to_csv('submission_new.csv',index=False)
```