In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler,QuantileTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

%matplotlib inline
```

In [2]:
```
traindf = pd.read_csv('train.csv')
traindf
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCor
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	

1460 rows × 81 columns

In [3]:
```
traindf.columns
```

Out[3]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgTyp
e',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemod
Add',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrTyp
e',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heatin
g',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullB
ath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageT
ype',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQ
ual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [4]:
```
numeric_df = traindf.select_dtypes(include='number')
correlation_matrix = numeric_df.corr()
correlation_matrix['SalePrice'].sort_values(ascending = False)
```

Out[4]:
```
SalePrice        1.000000
OverallQual      0.790982
GrLivArea        0.708624
GarageCars       0.640409
GarageArea       0.623431
TotalBsmtSF      0.613581
1stFlrSF         0.605852
FullBath         0.560664
TotRmsAbvGrd     0.533723
YearBuilt        0.522897
YearRemodAdd     0.507101
GarageYrBlt      0.486362
MasVnrArea       0.477493
Fireplaces       0.466929
BsmtFinSF1       0.386420
LotFrontage      0.351799
WoodDeckSF       0.324413
2ndFlrSF         0.319334
OpenPorchSF      0.315856
HalfBath         0.284108
LotArea          0.263843
BsmtFullBath     0.227122
BsmtUnfSF        0.214479
BedroomAbvGr     0.168213
ScreenPorch      0.111447
PoolArea         0.092404
MoSold           0.046432
3SsnPorch        0.044584
BsmtFinSF2      -0.011378
BsmtHalfBath    -0.016844
MiscVal         -0.021190
Id              -0.021917
LowQualFinSF    -0.025606
YrSold          -0.028923
OverallCond     -0.077856
MSSubClass      -0.084284
EnclosedPorch   -0.128578
KitchenAbvGr    -0.135907
Name: SalePrice, dtype: float64
```

In [5]:
```
req_tr = ["GarageArea","OverallQual","TotalBsmtSF","1stFlrSF","2ndFlrSF","L
req_tr
```

Out[5]:
```
['GarageArea',
 'OverallQual',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'TotRmsAbvGrd',
 'SalePrice']
```

In [6]:
```
selected_tr = traindf[req_tr]
selected_tr
```

Out[6]:

	GarageArea	OverallQual	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea
0	548	7	856	856	854	0	1710
1	460	6	1262	1262	0	0	1262
2	608	7	920	920	866	0	1786
3	642	7	756	961	756	0	1717
4	836	8	1145	1145	1053	0	2198
...	...	...	...	...	...	...	...
1455	460	6	953	953	694	0	1647
1456	500	6	1542	2073	0	0	2073
1457	252	7	1152	1188	1152	0	2340
1458	240	5	1078	1078	0	0	1078
1459	276	5	1256	1256	0	0	1256

1460 rows × 13 columns

In [7]:
```
selected_tr.loc[:, 'TotalBath'] = (selected_tr['BsmtFullBath'].fillna(0) +
                                   selected_tr['BsmtHalfBath'].fillna(0) +
                                   selected_tr['FullBath'].fillna(0) +
                                   selected_tr['HalfBath'].fillna(0))

selected_tr.loc[:, 'TotalSF'] = (selected_tr['TotalBsmtSF'].fillna(0) +
                                 selected_tr['1stFlrSF'].fillna(0) +
                                 selected_tr['2ndFlrSF'].fillna(0) +
                                 selected_tr['LowQualFinSF'].fillna(0) +
                                 selected_tr['GrLivArea'].fillna(0))
```

```
C:\Users\Aditya Kudva\AppData\Local\Temp\ipykernel_22768\341052813.py:1: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  selected_tr.loc[:, 'TotalBath'] = (selected_tr['BsmtFullBath'].fillna(0)
+
C:\Users\Aditya Kudva\AppData\Local\Temp\ipykernel_22768\341052813.py:6: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  selected_tr.loc[:, 'TotalSF'] = (selected_tr['TotalBsmtSF'].fillna(0) +
```

In [8]:
```
selected_tr
```

Out[8]:

	GarageArea	OverallQual	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea
0	548	7	856	856	854	0	1710
1	460	6	1262	1262	0	0	1262
2	608	7	920	920	866	0	1786
3	642	7	756	961	756	0	1717
4	836	8	1145	1145	1053	0	2198
...	...	...	...	...	...	...	...
1455	460	6	953	953	694	0	1647
1456	500	6	1542	2073	0	0	2073
1457	252	7	1152	1188	1152	0	2340
1458	240	5	1078	1078	0	0	1078
1459	276	5	1256	1256	0	0	1256

1460 rows × 15 columns

In [9]:
```
train_df = selected_tr[['TotRmsAbvGrd','TotalBath','GarageArea','TotalSF','(
train_df
```

Out[9]:

	TotRmsAbvGrd	TotalBath	GarageArea	TotalSF	OverallQual	SalePrice
0	8	4	548	4276	7	208500
1	6	3	460	3786	6	181500
2	6	4	608	4492	7	223500
3	7	2	642	4190	7	140000
4	9	4	836	5541	8	250000
...	...	...	...	...	...	...
1455	7	3	460	4247	6	175000
1456	7	3	500	5688	6	210000
1457	9	2	252	5832	7	266500
1458	5	2	240	3234	5	142125
1459	6	3	276	3768	5	147500

1460 rows × 6 columns

In [10]:
```
from sklearn.model_selection import train_test_split
train_set,test_set =train_test_split(train_df,test_size = 0.2,random_state =
print(f"Rows in train set: {len(train_set)}\nRows in test set:{len(test_set
```

```
Rows in train set: 1168
Rows in test set:292
```

In [11]:
```
housing = train_set.drop("SalePrice",axis=1)
housing_labels = train_set["SalePrice"].copy()
```

In [12]:
```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy="median")),
    ('std_scaler',StandardScaler())
])
```

In [13]:
```
X_train = my_pipeline.fit_transform(housing)
X_train
```

Out[13]:
```
array([[-0.96456591, -0.48377079, -0.86383727, -0.13352109, -0.82044456],
       [ 0.27075534,  0.61127627, -0.45626397, -0.13428593, -0.08893368],
       [-1.58222654, -1.57881784, -2.25716927, -1.32207838, -0.82044456],
       ...,
       [-0.96456591, -0.48377079,  0.45366713, -1.16605156, -0.82044456],
       [ 0.27075534, -0.48377079, -1.23349678, -0.26966215,  0.64257719],
       [ 0.27075534, -0.48377079,  0.87071888,  0.28025593,  0.64257719]])
```

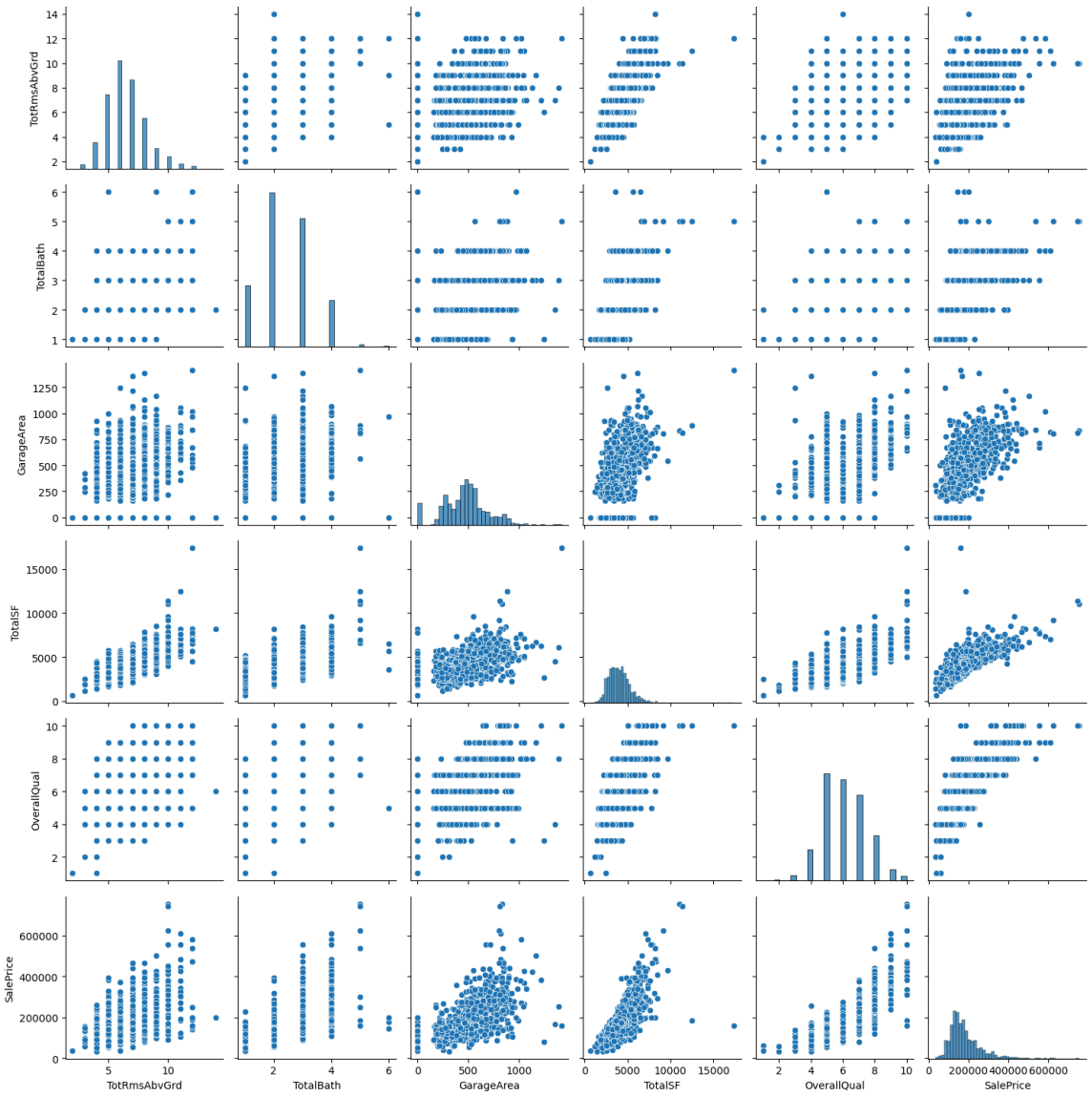
In [14]:
```
Y_train = housing_labels
Y_train
```

Out[14]:
```
254     145000
1066    178000
638      85000
799     175000
380     127000
         ...
1095    176432
1130    135000
1294    115000
860     189950
1126    174000
Name: SalePrice, Length: 1168, dtype: int64
```

In [15]:
```
Y_train.shape
```

Out[15]:
```
(1168,)
```

In [16]:
```
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
%matplotlib inline
sns.pairplot(train_df)
plt.tight_layout()
plt.show()
```

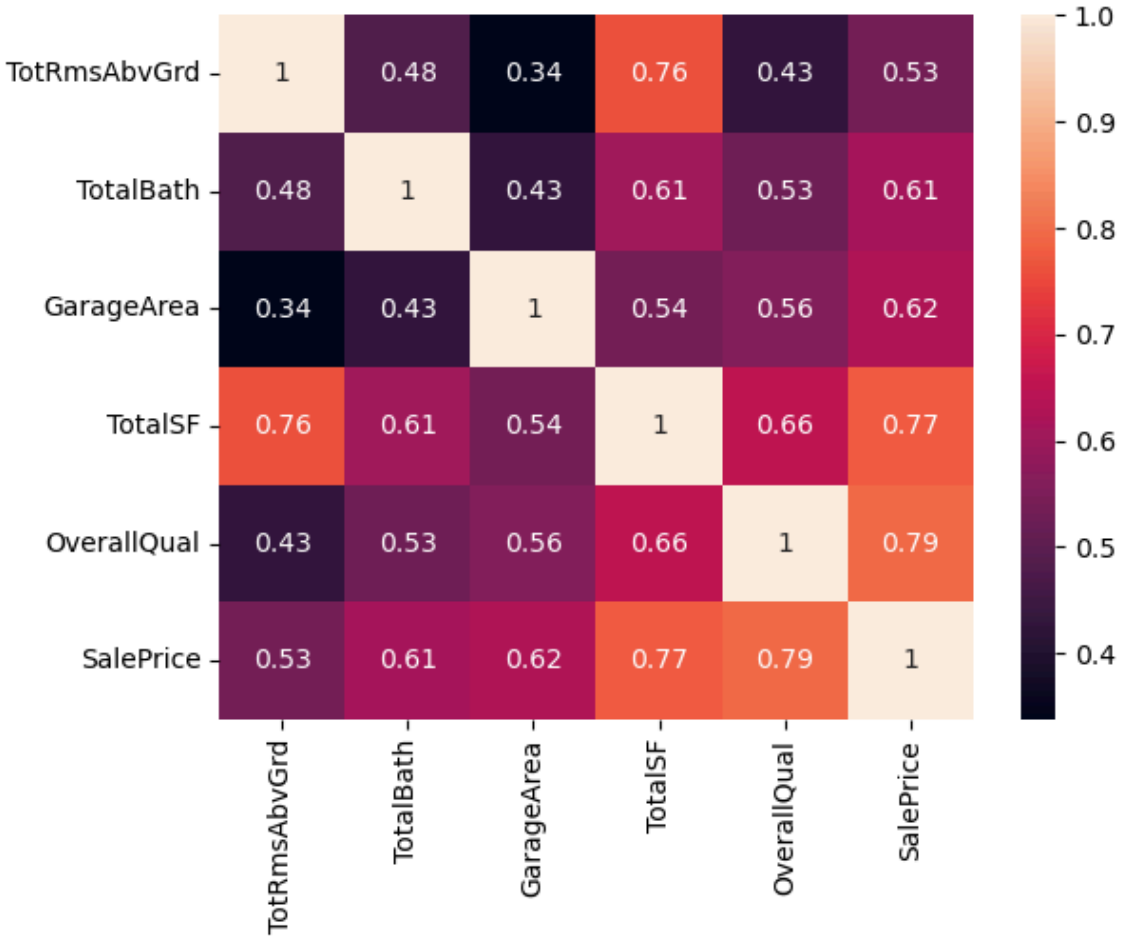


In [17]:
```
corr_matrix = train_df.corr()
corr_matrix['SalePrice'].sort_values(ascending = False)
```

Out[17]:
```
SalePrice       1.000000
OverallQual     0.790982
TotalSF         0.773909
GarageArea      0.623431
TotalBath       0.613005
TotRmsAbvGrd    0.533723
Name: SalePrice, dtype: float64
```

In [18]:
```
sns.heatmap(train_df.corr(),annot = True)
```

Out[18]: <Axes: >



In [19]:
```
testdf = pd.read_csv("test.csv")
testdf
```

Out[19]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCor
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	
1454	2915	160	RM	21.0	1936	Pave	NaN	Reg	
1455	2916	160	RM	21.0	1894	Pave	NaN	Reg	
1456	2917	20	RL	160.0	20000	Pave	NaN	Reg	
1457	2918	85	RL	62.0	10441	Pave	NaN	Reg	
1458	2919	60	RL	74.0	9627	Pave	NaN	Reg	

1459 rows × 80 columns

In [20]:
```
testdf.head()
```

Out[20]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContou
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lv
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lv
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lv
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lv
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS

5 rows × 80 columns

In [21]:
```
req_tst = ["GarageArea","OverallQual","TotalBsmtSF","1stFlrSF","2ndFlrSF","
req_tst
```

Out[21]:
```
['GarageArea',
 'OverallQual',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'TotRmsAbvGrd']
```

In [22]:
```
selected_tst = testdf[req_tst]
selected_tst
```

Out[22]:

	GarageArea	OverallQual	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea
0	730.0	5	882.0	896	0	0	896
1	312.0	6	1329.0	1329	0	0	1329
2	482.0	5	928.0	928	701	0	1629
3	470.0	6	926.0	926	678	0	1604
4	506.0	8	1280.0	1280	0	0	1280
...	...	...	...	...	...	...	...
1454	0.0	4	546.0	546	546	0	1092
1455	286.0	4	546.0	546	546	0	1092
1456	576.0	5	1224.0	1224	0	0	1224
1457	0.0	5	912.0	970	0	0	970
1458	650.0	7	996.0	996	1004	0	2000

1459 rows × 12 columns

In [23]:
```
selected_tst.loc[:, 'TotalBath'] = (selected_tst['BsmtFullBath'].fillna(0) +
                                    selected_tst['BsmtHalfBath'].fillna(0) +
                                    selected_tst['FullBath'].fillna(0) +
                                    selected_tst['HalfBath'].fillna(0))

selected_tst.loc[:, 'TotalSF'] = (selected_tst['TotalBsmtSF'].fillna(0) +
                                  selected_tst['1stFlrSF'].fillna(0) +
                                  selected_tst['2ndFlrSF'].fillna(0) +
                                  selected_tst['LowQualFinSF'].fillna(0) +
                                  selected_tst['GrLivArea'].fillna(0))
```

```
C:\Users\Aditya Kudva\AppData\Local\Temp\ipykernel_22768\771691818.py:1: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  selected_tst.loc[:, 'TotalBath'] = (selected_tst['BsmtFullBath'].fillna
(0) +
C:\Users\Aditya Kudva\AppData\Local\Temp\ipykernel_22768\771691818.py:6: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  selected_tst.loc[:, 'TotalSF'] = (selected_tst['TotalBsmtSF'].fillna(0)
+
```

In [24]:
```
selected_tst
```

Out[24]:

	GarageArea	OverallQual	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea
0	730.0	5	882.0	896	0	0	896
1	312.0	6	1329.0	1329	0	0	1329
2	482.0	5	928.0	928	701	0	1629
3	470.0	6	926.0	926	678	0	1604
4	506.0	8	1280.0	1280	0	0	1280
...	...	...	...	...	...	...	...
1454	0.0	4	546.0	546	546	0	1092
1455	286.0	4	546.0	546	546	0	1092
1456	576.0	5	1224.0	1224	0	0	1224
1457	0.0	5	912.0	970	0	0	970
1458	650.0	7	996.0	996	1004	0	2000

1459 rows × 14 columns

In [25]:
```
test_df_unproc = selected_tst[['TotRmsAbvGrd','TotalBath','GarageArea','Tota
test_df_unproc
```

Out[25]:

	TotRmsAbvGrd	TotalBath	GarageArea	TotalSF	OverallQual
0	5	1.0	730.0	2674.0	5
1	6	2.0	312.0	3987.0	6
2	6	3.0	482.0	4186.0	5
3	7	3.0	470.0	4134.0	6
4	5	2.0	506.0	3840.0	8
...	...	...	...	...	...
1454	5	2.0	0.0	2730.0	4
1455	6	2.0	286.0	2730.0	4
1456	7	2.0	576.0	3672.0	5
1457	6	2.0	0.0	2852.0	5
1458	9	3.0	650.0	4996.0	7

1459 rows × 5 columns

In [26]:
```
test_df = test_df_unproc.fillna(test_df_unproc.mean())
test_df
```

Out[26]:

	TotRmsAbvGrd	TotalBath	GarageArea	TotalSF	OverallQual
0	5	1.0	730.0	2674.0	5
1	6	2.0	312.0	3987.0	6
2	6	3.0	482.0	4186.0	5
3	7	3.0	470.0	4134.0	6
4	5	2.0	506.0	3840.0	8
...	...	...	...	...	...
1454	5	2.0	0.0	2730.0	4
1455	6	2.0	286.0	2730.0	4
1456	7	2.0	576.0	3672.0	5
1457	6	2.0	0.0	2852.0	5
1458	9	3.0	650.0	4996.0	7

1459 rows × 5 columns

In [27]:
```
x_test = my_pipeline.transform(test_df[['TotRmsAbvGrd','TotalBath','GarageAr
x_test
```

Out[27]:
```
array([[-0.96456591, -1.57881784,  1.2024646 , -1.10333489, -0.82044456],
       [-0.34690528, -0.48377079, -0.77853123, -0.09910341, -0.08893368],
       [-0.34690528,  0.61127627,  0.02713693,  0.05309923, -0.82044456],
       ...,
       [ 0.27075534, -0.48377079,  0.47262403, -0.34002719, -0.82044456],
       [-0.34690528, -0.48377079, -2.25716927, -0.96719384, -0.82044456],
       [ 1.50607659,  0.61127627,  0.82332664,  0.67261751,  0.64257719]])
```

In [28]:
```
#model = LinearRegression()
#model = DecisionTreeRegressor()
model = RandomForestRegressor()
model.fit(X_train,Y_train)
```

Out[28]:
```
RandomForestRegressor()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [29]:
```
y_train_pred = model.predict(X_train)
y_train_pred
```

Out[29]:
```
array([146635.  , 170766.5 ,  91754.  , ..., 120424.83, 177563.95,
       190052.8 ])
```

In [30]:
```
y_train_pred[:5]
```

Out[30]:
```
array([146635.  , 170766.5 ,  91754.  , 167666.74, 141082.  ])
```

In [31]:
```
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
```

In [32]:
```
proc_data = my_pipeline.transform(some_data)
proc_data
```

Out[32]:
```
array([[-0.96456591, -0.48377079, -0.86383727, -0.13352109, -0.82044456],
       [ 0.27075534,  0.61127627, -0.45626397, -0.13428593, -0.08893368],
       [-1.58222654, -1.57881784, -2.25716927, -1.32207838, -0.82044456],
       [ 0.27075534,  0.61127627, -1.11975539,  0.11505106, -0.82044456],
       [-0.34690528, -0.48377079, -0.79748813,  0.22289313, -0.82044456]])
```

In [33]:
```
model.predict(proc_data)
```

Out[33]:
```
array([146635.  , 170766.5 ,  91754.  , 167666.74, 141082.  ])
```

In [34]:
```
list(some_labels)
```

Out[34]:
```
[145000, 178000, 85000, 175000, 127000]
```

In [35]:
```
train_mse = mean_squared_error(Y_train,y_train_pred)
train_mse
```

Out[35]:
```
170426901.4466204
```

In [36]:
```
train_rmse = np.sqrt(train_mse)
train_rmse
```

Out[36]:
```
13054.76546884778
```

In [43]:
```
from sklearn.model_selection import cross_val_score
```

In [44]:
```
scores = cross_val_score(model,X_train,Y_train,scoring="neg_mean_squared_er
scores
```

Out[44]:
```
array([-4.17042658e+08, -1.97665322e+08, -7.07799177e+08, -1.54138912e+08,
       -2.17355681e+09, -1.48332245e+08, -4.03148812e+08, -1.62294032e+08,
       -1.35290552e+08, -2.87857463e+09, -1.30030232e+09, -8.89090485e+08,
       -1.94930984e+08, -9.62243333e+07, -3.79258415e+08, -5.73970712e+08,
       -2.67995647e+08, -9.72036694e+08, -1.52021582e+09, -6.13196949e+08,
       -6.75235696e+08, -3.52287921e+08, -3.47514407e+08, -7.13996222e+08,
       -3.55001519e+08, -2.93063477e+08, -1.69396062e+09, -1.41733837e+09,
       -2.61393581e+10, -2.15896850e+09, -5.42914316e+08, -1.00951119e+09,
       -3.94847490e+08, -9.89373288e+08, -2.62993796e+09, -1.47563935e+08,
       -6.00289399e+08, -8.97184920e+08, -3.31002093e+08, -9.72364897e+08,
       -4.56395684e+08, -9.96997209e+08, -7.58248661e+08, -1.15871275e+09,
       -1.11244083e+09, -7.86895040e+08, -7.34146100e+08, -1.66383682e+09,
       -4.72450927e+08, -5.42167097e+08, -4.66637623e+08, -2.67771778e+09,
       -1.44179796e+09, -1.32552220e+09, -4.88336505e+08, -7.31613327e+08,
       -8.04540529e+07, -6.97651799e+08, -5.58330951e+08, -9.50873755e+08,
       -3.69590911e+10, -1.88370858e+08, -6.12143410e+08, -1.39566644e+09,
       -5.47132553e+08, -6.13316705e+08, -1.57074723e+09, -7.73886884e+08,
       -1.13427252e+08, -2.11394092e+08, -2.91253659e+09, -6.83555937e+08,
       -3.04420033e+09, -3.61427296e+08, -2.13027579e+09, -1.81952558e+09,
       -1.56080140e+09, -8.03591387e+08, -1.54789097e+09, -1.02593416e+09,
       -7.49226235e+08, -1.05436689e+09, -1.03017181e+10, -1.03564790e+08,
       -2.32032983e+09, -1.07987819e+09, -3.42153879e+08, -5.45467410e+08,
       -1.22928016e+09, -5.02091807e+09, -3.25535533e+08, -4.97154551e+08,
       -7.17822606e+08, -6.06301458e+08, -4.11457681e+08, -4.41478876e+08,
       -6.60814028e+08, -7.22824213e+08, -4.98583306e+09, -1.75231112e+08,
       -1.04245234e+09, -8.64181646e+08, -2.28929282e+09, -2.75352470e+09,
       -3.27278427e+08, -3.45195687e+08, -1.55201185e+09, -3.33863501e+08,
       -4.28959700e+08, -2.41651364e+08, -3.49649992e+08, -1.66242710e+08,
       -1.14046524e+08, -4.43206673e+08, -9.62437711e+08, -1.13291442e+09,
       -5.79250048e+09, -6.41371551e+08, -3.82673308e+08, -7.85446269e+08,
       -9.67281485e+08, -6.50441665e+08, -2.52742763e+08, -3.98549170e+08,
       -9.26495086e+08, -4.41937843e+08, -8.91583016e+08, -2.02544284e+09,
       -1.56266441e+09, -2.92654614e+08, -1.25293356e+09, -7.00808689e+07,
       -6.45091336e+08, -7.67086393e+08, -8.24577275e+08, -1.38868761e+08,
       -2.25742579e+09, -1.05085910e+09, -8.74216913e+08, -1.58613923e+08,
       -6.75530416e+08, -5.29062477e+08, -7.66270683e+08, -1.28677622e+08,
       -7.21557728e+08, -9.05456641e+08, -1.19518839e+09, -3.84569554e+08,
       -1.36240938e+08, -2.72454269e+08, -5.99228881e+08, -6.66863934e+08,
       -2.04041510e+08, -1.76445461e+09, -9.83262243e+08, -2.68258374e+08,
       -8.55489802e+08, -6.23662105e+08, -4.06920723e+08, -6.02168359e+08,
       -2.86292831e+09, -6.83119511e+08, -7.21062601e+08, -2.77192968e+08,
       -2.60348040e+09, -4.70059140e+08, -1.60870787e+08, -3.78769584e+09,
       -4.23275843e+08, -9.18686174e+08, -9.35618293e+08, -4.87145956e+08,
       -2.12200418e+08, -1.76526663e+08, -3.44751886e+08, -2.14325026e+08,
       -4.14992393e+08, -3.10322436e+08, -1.33699434e+09, -4.37835830e+08,
       -1.48083441e+09, -6.11915115e+07, -3.10104433e+08, -6.56366992e+07,
       -3.92807317e+08, -8.33921871e+08, -5.84170413e+08, -2.04730128e+08,
       -6.87567952e+08, -1.76358756e+09, -7.23786002e+08, -3.51189097e+08,
       -2.80502811e+08, -1.72812675e+08, -2.43837010e+08, -2.37738369e+08,
       -3.58871863e+09, -4.32843438e+08, -3.91157488e+08, -8.96570252e+0
8])
```

In [45]:
```
rmse_scores = np.sqrt(-scores)
rmse_scores
```

Out[45]:
```
array([ 20421.62231985,  14059.34999417,  26604.49542242,  12415.26932029,
        46621.42006738,  12179.17257686,  20078.56599512,  12739.46748623,
        11631.44670413,  53652.34970255,  36059.70497865,  29817.62038411,
        13961.76865429,   9809.40024955,  19474.55814746,  23957.68585533,
        16370.57259318,  31177.50300974,  38989.94514901,  24762.81382912,
        25985.29768245,  18769.33458064,  18641.7382987 ,  26720.7077355 ,
        18841.48399098,  17119.09683954,  41157.75283661,  37647.55460983,
       161676.70867475,  46464.70169437,  23300.52180193,  31772.80588389,
        19870.76973745,  31454.30476139,  51282.92070038,  12147.58966996,
        24500.80404397,  29953.04525758,  18193.46291696,  31182.76603045,
        21363.41928876,  31575.26261569,  27536.31531857,  34039.86994288,
        33353.27319014,  28051.64951266,  27095.13055329,  40790.15588786,
        21735.93630363,  23284.48188617,  21601.79675183,  51746.66926597,
        37971.01471061,  36407.72177461,  22098.33715075,  27048.35165511,
         8969.61832508,  26413.09901011,  23629.02772131,  30836.24093428,
       192247.47353907,  13724.82635406,  24741.5320865 ,  37358.61934773,
        23390.86473113,  24765.23178021,  39632.65356182,  27818.82247285,
        10650.22308706,  14539.39791608,  53967.92185592,  26144.90268695,
        55174.27233228,  19011.24129571,  46154.91077424,  42655.89737298,
        39506.97916261,  28347.687503  ,  39343.24549033,  32030.20706993,
        27371.9972785 ,  32471.01611226, 101497.37971179,  10176.67870795,
        48169.80204482,  32861.50012766,  18497.40195294,  23355.24373602,
        35061.09187602,  70858.43681838,  18042.60327901,  22296.96281607,
        26792.21165885,  24623.18943223,  20284.41965296,  21011.39871919,
        25706.30327173,  26885.39032826,  70610.43164678,  13237.48887748,
        32287.03051348,  29396.96661108,  47846.55492446,  52474.03835472,
        18090.83819884,  18579.44259864,  39395.58165501,  18271.93204463,
        20711.34231195,  15545.13956305,  18698.93024989,  12893.51424764,
        10679.25671276,  21052.47426398,  31023.18022077,  33658.79406504,
        76108.47837615,  25325.31443278,  19562.0374269 ,  28025.81432911,
        31101.1492487 ,  25503.75786046,  15897.8854945 ,  19963.69629401,
        30438.38179024,  21022.31773812,  29859.38740157,  45004.92015812,
        39530.55036631,  17107.15095656,  35396.80158255,   8371.43171136,
        25398.64830751,  27696.3245432 ,  28715.45358427,  11784.25904436,
        47512.37513438,  32416.95691447,  29567.15937169,  12594.20194397,
        25990.96797165,  23001.35814713,  27681.59466798,  11343.61591835,
        26861.82659618,  30090.80658364,  34571.49680702,  19610.44501771,
        11672.22936148,  16506.1888215 ,  24479.1519668 ,  25823.70876419,
        14284.309915  ,  42005.41169575,  31357.01266229,  16378.59499388,
        29248.75726622,  24973.22776769,  20172.27609696,  24539.11895334,
        53506.33896075,  26136.55506736,  26852.60883703,  16649.11312762,
        51024.31188735,  21680.84730782,  12683.48481456,  61544.25917616,
        20573.66868056,  30309.83625887,  30587.87819954,  22071.38319424,
        14567.10054533,  13286.33367565,  18567.49541334,  14639.84379043,
        20371.36207363,  17615.97103503,  36564.93316762,  20924.52700087,
        38481.61127154,   7822.50033411,  17609.78232105,   8101.64792837,
        19819.36721238,  28877.70543877,  24169.6175614 ,  14308.39361952,
        26221.51697011,  41995.08975082,  26903.27120863,  18740.03994577,
        16748.21815101,  13145.82348494,  15615.28128527,  15418.7667904 ,
        59905.91482839,  20804.8897536 ,  19777.70178568,  29942.782964
])
```

In [46]:
```
def print_scores(scores):
    print("Scores:",scores)
    print("Mean:",scores.mean())
    print("Standard Deviation",scores.std())
```

In [47]:
```
print_scores(rmse_scores)
```

```
Scores: [ 20421.62231985  14059.34999417  26604.49542242  12415.26932029
  46621.42006738  12179.17257686  20078.56599512  12739.46748623
  11631.44670413  53652.34970255  36059.70497865  29817.62038411
  13961.76865429   9809.40024955  19474.55814746  23957.68585533
  16370.57259318  31177.50300974  38989.94514901  24762.81382912
  25985.29768245  18769.33458064  18641.7382987   26720.7077355
  18841.48399098  17119.09683954  41157.75283661  37647.55460983
 161676.70867475  46464.70169437  23300.52180193  31772.80588389
  19870.76973745  31454.30476139  51282.92070038  12147.58966996
  24500.80404397  29953.04525758  18193.46291696  31182.76603045
  21363.41928876  31575.26261569  27536.31531857  34039.86994288
  33353.27319014  28051.64951266  27095.13055329  40790.15588786
  21735.93630363  23284.48188617  21601.79675183  51746.66926597
  37971.01471061  36407.72177461  22098.33715075  27048.35165511
   8969.61832508  26413.09901011  23629.02772131  30836.24093428
 192247.47353907  13724.82635406  24741.5320865   37358.61934773
  23390.86473113  24765.23178021  39632.65356182  27818.82247285
  10650.22308706  14539.39791608  53967.92185592  26144.90268695
  55174.27233228  19011.24129571  46154.91077424  42655.89737298
  39506.97916261  28347.687503    39343.24549033  32030.20706993
  27371.9972785   32471.01611226 101497.37971179  10176.67870795
  48169.80204482  32861.50012766  18497.40195294  23355.24373602
  35061.09187602  70858.43681838  18042.60327901  22296.96281607
  26792.21165885  24623.18943223  20284.41965296  21011.39871919
  25706.30327173  26885.39032826  70610.43164678  13237.48887748
  32287.03051348  29396.96661108  47846.55492446  52474.03835472
  18090.83819884  18579.44259864  39395.58165501  18271.93204463
  20711.34231195  15545.13956305  18698.93024989  12893.51424764
  10679.25671276  21052.47426398  31023.18022077  33658.79406504
  76108.47837615  25325.31443278  19562.0374269   28025.81432911
  31101.1492487   25503.75786046  15897.8854945   19963.69629401
  30438.38179024  21022.31773812  29859.38740157  45004.92015812
  39530.55036631  17107.15095656  35396.80158255   8371.43171136
  25398.64830751  27696.3245432   28715.45358427  11784.25904436
  47512.37513438  32416.95691447  29567.15937169  12594.20194397
  25990.96797165  23001.35814713  27681.59466798  11343.61591835
  26861.82659618  30090.80658364  34571.49680702  19610.44501771
  11672.22936148  16506.1888215   24479.1519668   25823.70876419
  14284.309915    42005.41169575  31357.01266229  16378.59499388
  29248.75726622  24973.22776769  20172.27609696  24539.11895334
  53506.33896075  26136.55506736  26852.60883703  16649.11312762
  51024.31188735  21680.84730782  12683.48481456  61544.25917616
  20573.66868056  30309.83625887  30587.87819954  22071.38319424
  14567.10054533  13286.33367565  18567.49541334  14639.84379043
  20371.36207363  17615.97103503  36564.93316762  20924.52700087
  38481.61127154   7822.50033411  17609.78232105   8101.64792837
  19819.36721238  28877.70543877  24169.6175614   14308.39361952
  26221.51697011  41995.08975082  26903.27120863  18740.03994577
  16748.21815101  13145.82348494  15615.28128527  15418.7667904
  59905.91482839  20804.8897536   19777.70178568  29942.782964  ]
Mean: 28888.513378984808
Standard Deviation 20096.72101411328
```

In [48]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[48]:
```
array([130018.49, 151465.95, 140568.  , ..., 140460.5 , 106952.5 ,
       236885.56])
```

In [49]:
```
pred=pd.DataFrame(y_pred)
sub_df=pd.read_csv('sample_submission.csv')
datasets=pd.concat([sub_df['Id'],pred],axis=1)
datasets.columns=['Id','SalePrice']
datasets.to_csv('sample_submission.csv',index=False)
```

In [ ]:
```

```