Nikhil Prabhu, Aaron Chen, Elvis Matos, Reese Mayer

Assignment 1: House Regression

Introduction:

The dataset that was leveraged for this analysis was the House Prices Advanced Regression obtained from Kaggle. This dataset contains, “79 explanatory variables describing every aspect of residential homes in Ames, Iowa with the dependent variable being Sales Price to help predict the final price of each home. The goal is to establish and research the variables of house square footage and lot size that will impact the final price of the home.

Descriptive Statistics:

The dependent variable was identified as the Sales Price. By using this variable it was determined the average sale price of a home was 180,921.20. The minimum price was observed as 34,900 and the maximum was observed as 755,000. The standard deviation of this variable was shown to be 79,442.50. A histogram was constructed to display the distribution of the dependent variable. This histogram showed to be skewed heavily towards the right. While observing the histogram it displays that the bulk of the distribution sits within the lower price range(100,000-200,000).

Missing Data and Outliers:

The dataset includes housing features with numerous missing values, particularly in 'Alley', 'PoolQC', 'MiscFeature', and 'Fence', likely because not all properties have these attributes. Whole 'LotFrontage' and 'FireplaceQu' also show missing data, this may stem from recording issues. Such gaps in data can impact model accuracy and necessitate strategies like imputation or record removal. Also, a box plot analysis revealed outliers, with some properties' selling prices significantly deviating from the average, notably two examples exceeding $700,000.

Total Lot Size and Total Square Foot:

The following predictors that were chosen to investigate were TotalBsmtSF, '1stFlrSF', '2ndFlrSF', 'LotFrontage', 'LotArea', 'WoodDeckSF', 'OpenPorchSF', and 'SalePrice'. The correlation was calculated between these predictors and the sales price. It was determined that all calculated values show a positive correlation with the sales price variable. For example, a scatter plot was created between the sales price and the first floor sq ft which displays a positive correlation.

Merging predictor:

Two new variables were created to include the total square footage and total lot size. To create the total square footage variables such as TotalBsmtSF, '1stFlrSF', and '2ndFlrSF' were added to create TotalSF. These variables, 'LotFrontage' and 'LotArea' were added up to make the total lot size variable. These combined variables potentially have an impact on the sales price. Multiple scatter plots were created to show the relationship between Sales Price, Total square footage, and Total Lot Size which displayed a largely positive correlation.

Min-max Scaling and Standard Scaling:

For our analysis, the 'SalePrice' variable underwent two normalization processes. The Min-Max scaling mapped the original prices into a new range where the smallest value corresponds to 0 and the largest to 1, transforming them into values like 0.2410. The standard scaling adjusted the prices so that their distribution has a mean of zero and a variance of one, resulting in normalized values such as -0.4885. These transformations are essential for models that are sensitive to the scale of data, ensuring that the variable's scale does not unduly influence the model's performance.

Conclusion:

Throughout this project, we gained valuable insights about the housing market, distribution of sales prices, missing data, outliers, and potential predictors. Additionally, using min-max scaling and standard scaling techniques can help identify if the variables can be used to create an advanced regression model. By using these findings it will provide a good source of knowledge to help predict the house prices as we seek to build a compelling model.

In [63]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

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

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
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
...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl

1460 rows × 81 columns

Qestion 1

In [5]:
```
house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
```

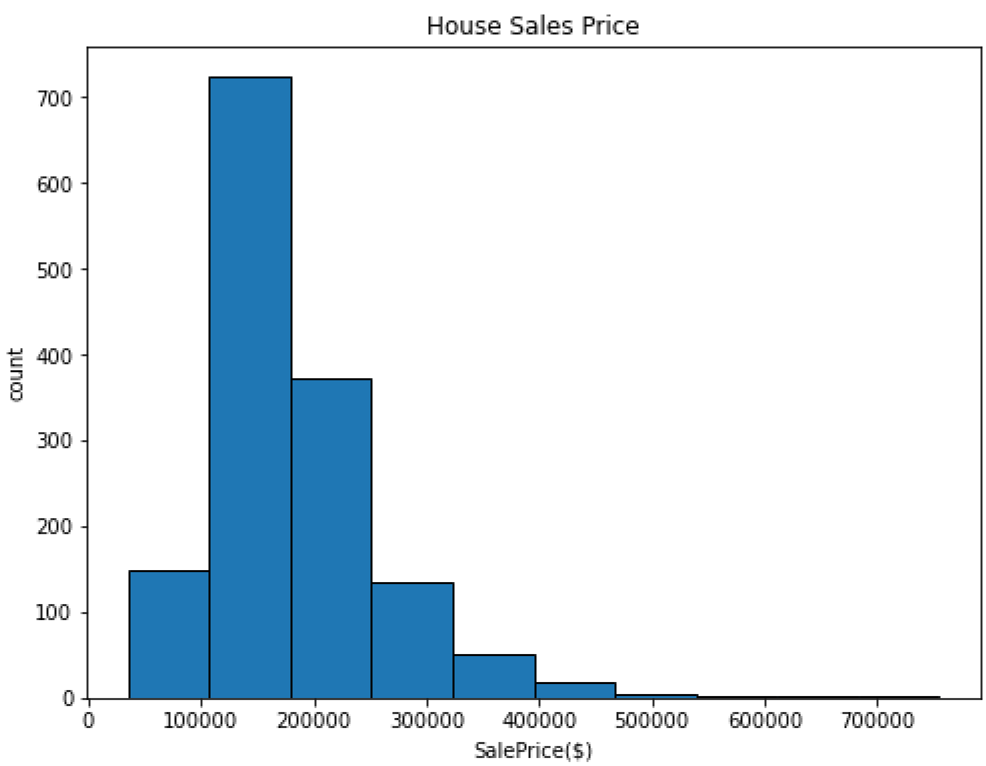
```
16  HouseStyle     1460 non-null   object
17  OverallQual    1460 non-null   int64
18  OverallCond    1460 non-null   int64
19  YearBuilt      1460 non-null   int64
20  YearRemodAdd   1460 non-null   int64
21  RoofStyle      1460 non-null   object
22  RoofMatl       1460 non-null   object
23  Exterior1st    1460 non-null   object
24  Exterior2nd    1460 non-null   object
25  MasVnrType     1452 non-null   object
26  MasVnrArea     1452 non-null   float64
27  ExterQual      1460 non-null   object
28  ExterCond      1460 non-null   object
29  Foundation     1460 non-null   object
30  BsmtQual       1423 non-null   object
31  BsmtCond       1423 non-null   object
32  BsmtExposure   1422 non-null   object
33  BsmtFinType1   1423 non-null   object
34  BsmtFinSF1     1460 non-null   int64
35  BsmtFinType2   1422 non-null   object
36  BsmtFinSF2     1460 non-null   int64
37  BsmtUnfSF      1460 non-null   int64
38  TotalBsmtSF    1460 non-null   int64
39  Heating        1460 non-null   object
40  HeatingQC      1460 non-null   object
41  CentralAir     1460 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1460 non-null   int64
44  2ndFlrSF       1460 non-null   int64
45  LowQualFinSF   1460 non-null   int64
46  GrLivArea      1460 non-null   int64
47  BsmtFullBath   1460 non-null   int64
48  BsmtHalfBath   1460 non-null   int64
49  FullBath       1460 non-null   int64
50  HalfBath       1460 non-null   int64
51  BedroomAbvGr   1460 non-null   int64
52  KitchenAbvGr   1460 non-null   int64
53  KitchenQual    1460 non-null   object
54  TotRmsAbvGrd   1460 non-null   int64
55  Functional     1460 non-null   object
56  Fireplaces     1460 non-null   int64
57  FireplaceQu    770 non-null    object
58  GarageType     1379 non-null   object
59  GarageYrBlt    1379 non-null   float64
60  GarageFinish   1379 non-null   object
61  GarageCars     1460 non-null   int64
62  GarageArea     1460 non-null   int64
63  GarageQual     1379 non-null   object
64  GarageCond     1379 non-null   object
65  PavedDrive     1460 non-null   object
66  WoodDeckSF     1460 non-null   int64
67  OpenPorchSF    1460 non-null   int64
68  EnclosedPorch  1460 non-null   int64
69  3SsnPorch      1460 non-null   int64
70  ScreenPorch    1460 non-null   int64
71  PoolArea       1460 non-null   int64
72  PoolQC         7 non-null      object
73  Fence          281 non-null    object
74  MiscFeature    54 non-null     object
75  MiscVal        1460 non-null   int64
76  MoSold         1460 non-null   int64
77  YrSold         1460 non-null   int64
78  SaleType       1460 non-null   object
79  SaleCondition  1460 non-null   object
80  SalePrice      1460 non-null   int64
```

```
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [35]:
```
descriptive_stats = house['SalePrice'].describe()


skewness = house['SalePrice'].skew()
kurtosis = house['SalePrice'].kurtosis()


print(descriptive_stats)
print("Skewness:\n", skewness)
print("Kurtosis:\n", kurtosis)
```

```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
Skewness:
 1.8828757597682129
Kurtosis:
 6.536281860064529
```

In [65]:
```
plt.figure(figsize=(8,6))
plt.hist(df['SalePrice'], edgecolor='black')
plt.title('House Sales Price')
plt.xlabel('SalePrice($)')
plt.ylabel('count')
plt.show()
```



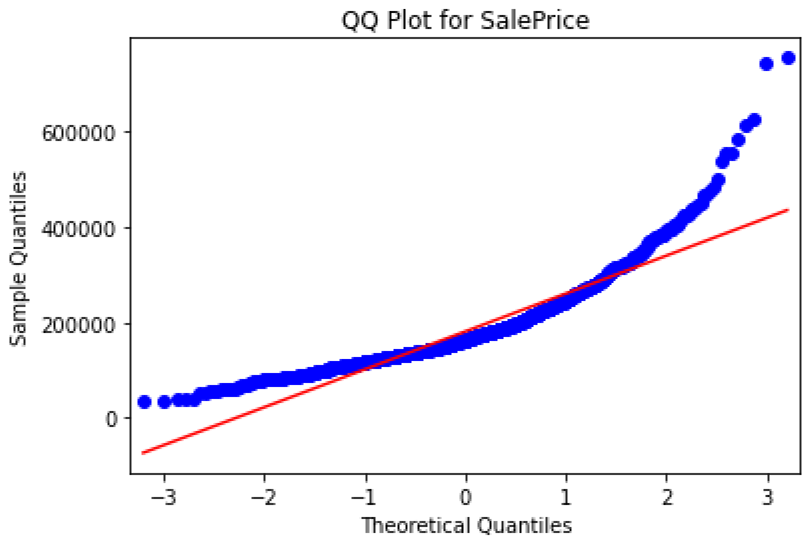
Question 2

In [64]:
```
house.isnull().sum()
missing_data = house.isnull().sum()
print("\nMissing data:")
print (missing_data[missing_data>0])
```

```
Missing data:
LotFrontage      259
Alley           1369
MasVnrType         8
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1453
Fence           1179
MiscFeature     1406
TotalLotSize     259
dtype: int64
```

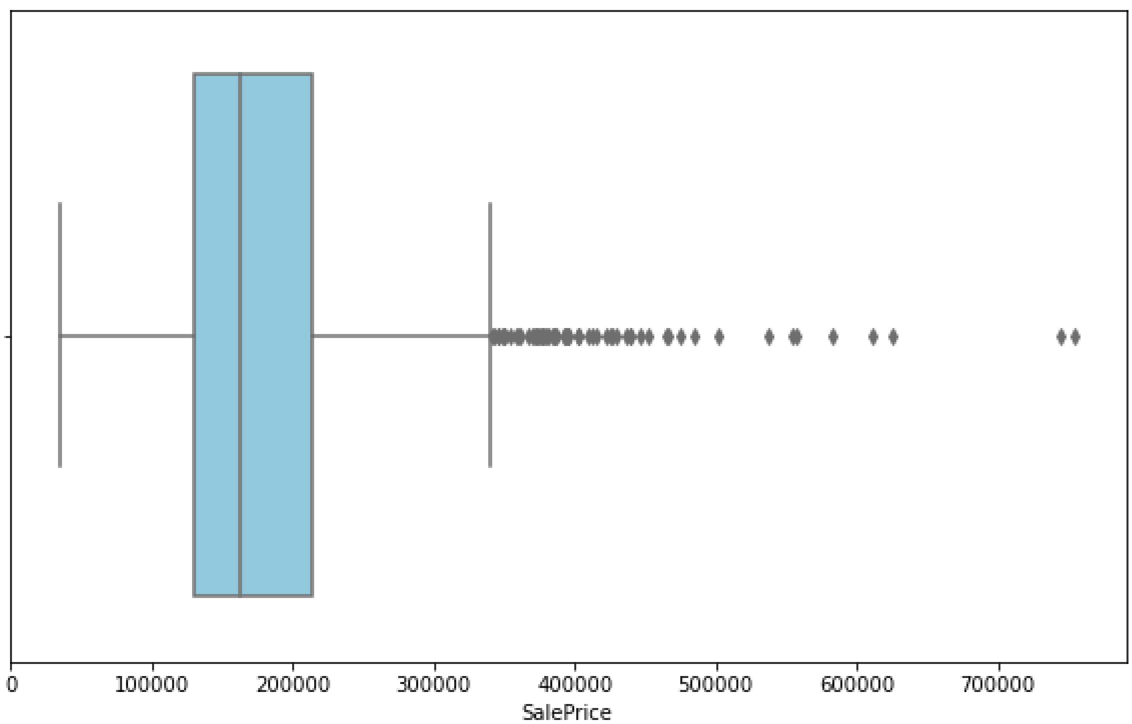
In [24]:
```
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.gofplots import qqplot

# Create the QQ plot for 'SalePrice'
plt.figure(figsize=(8, 6))
qqplot(house['SalePrice'], line='s') # 's' parameter adds a standardized line to
plt.title('QQ Plot for SalePrice')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.show()
```

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



In [28]:

```
plt.figure(figsize=(10 , 6))
sns.boxplot(x='SalePrice', data=df,
color='skyblue')
plt.show()
```



Question 3

In [85]:

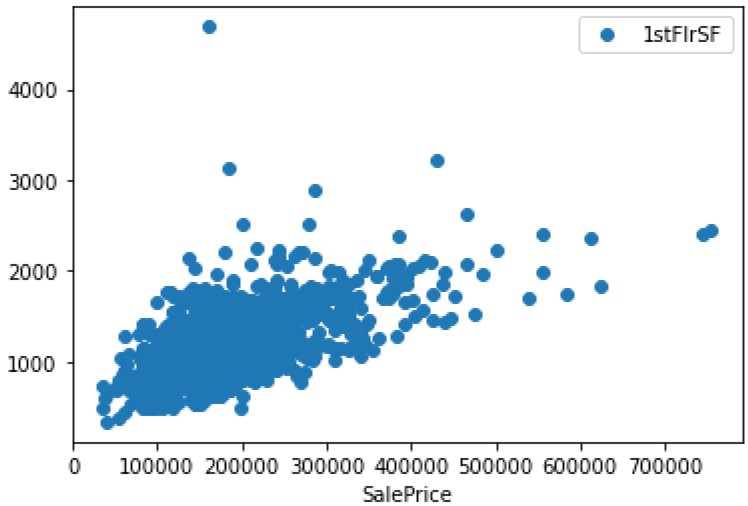
```
predict = house[['TotalBsmtSF','1stFlrSF','2ndFlrSF','LotFrontage', 'LotArea','S
print(predict.corr())
```

```
             TotalBsmtSF  1stFlrSF  2ndFlrSF  LotFrontage   LotArea  SalePrice
TotalBsmtSF     1.000000  0.819530 -0.174512     0.392075  0.260833   0.613581
1stFlrSF        0.819530  1.000000 -0.202646     0.457181  0.299475   0.605852
2ndFlrSF       -0.174512 -0.202646  1.000000     0.080177  0.050986   0.319334
LotFrontage     0.392075  0.457181  0.080177     1.000000  0.426095   0.351799
```

```
LotArea          0.260833  0.299475  0.050986     0.426095  1.000000   0.263843
SalePrice        0.613581  0.605852  0.319334     0.351799  0.263843   1.000000
```

In [87]:
```
predict.plot(x='SalePrice', y='1stFlrSF', style='o')
```

Out[87]: <AxesSubplot:xlabel='SalePrice'>



Question 4

In [77]:
```
house['TotalSF'] = house['TotalBsmtSF'] + house['1stFlrSF'] + house['2ndFlrSF']

house.head()
```

Out[77]:

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

5 rows × 83 columns

In [78]:
```
house['TotalLotSize'] = house['LotFrontage'] + house['LotArea']

house.head()
```

Out[78]:

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

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilit
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllF

5 rows × 83 columns

In [79]:
```
df = house[['SalePrice','TotalSF','TotalLotSize']]
df
```

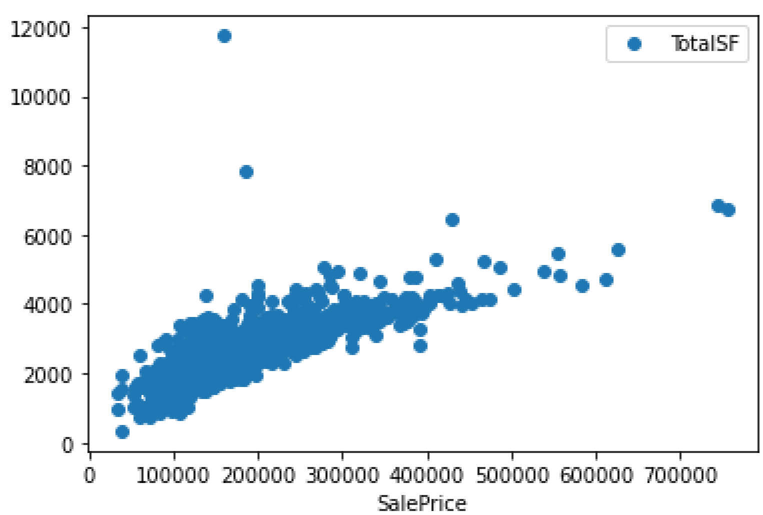
Out[79]:

	SalePrice	TotalSF	TotalLotSize
0	208500	2566	8515.0
1	181500	2524	9680.0
2	223500	2706	11318.0
3	140000	2473	9610.0
4	250000	3343	14344.0
...	...	...	...
1455	175000	2600	7979.0
1456	210000	3615	13260.0
1457	266500	3492	9108.0
1458	142125	2156	9785.0
1459	147500	2512	10012.0

1460 rows × 3 columns

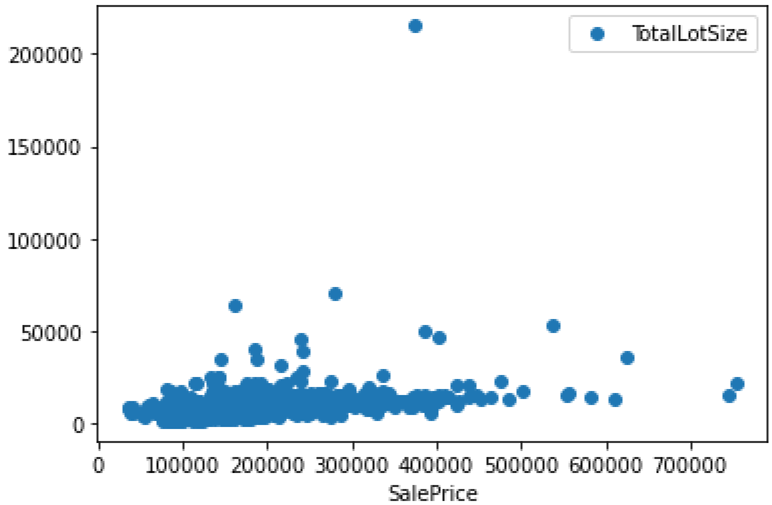
In [81]:
```
df.plot(x='SalePrice', y='TotalSF', style='o')
```

Out[81]: <AxesSubplot:xlabel='SalePrice'>



In [84]:
```
df.plot(x='SalePrice', y='TotalLotSize', style='o')
```

Out[84]: <AxesSubplot:xlabel='SalePrice'>



In [52]:

```
print(df.corr())
```

```
             SalePrice   TotalSF  TotalLotSize
SalePrice     1.000000  0.782260      0.319049
TotalSF       0.782260  1.000000      0.370958
TotalLotSize  0.319049  0.370958      1.000000
```

Question 5

In [62]:

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler


sale_price = df['SalePrice'].values.reshape(-1, 1)

min_max_scaler = MinMaxScaler()
sale_price_minmax = min_max_scaler.fit_transform(sale_price)

standard_scaler = StandardScaler()
sale_price_standard = standard_scaler.fit_transform(sale_price)

print("Min-Max scaled SalePrice:\n", sale_price_minmax)
print("Standard scaled SalePrice:\n", sale_price_standard)
```

```
Min-Max scaled SalePrice:
 [[0.24107763]
 [0.20358284]
 [0.26190807]
 ...
 [0.321622  ]
 [0.14890293]
 [0.15636717]]
Standard scaled SalePrice:
 [[ 0.34727322]
 [ 0.00728832]
 [ 0.53615372]
 ...
 [ 1.07761115]
 [-0.48852299]
 [-0.42084081]]
```