IN THIS NOTEBOOK:

- ADVANCE HOUSE PRICE PREDICTION - DATA ANALYSIS

- ADVANCE HOUSE PRICE PREDICTION - FEATURE SELECTION

- DATASET:

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

DATA ANALYSIS - ADVANCE HOUSE PRICE PREDICTION

- Missing Values
  - Categorical missing values
  - Numerical missing valeues
- Relationships between independent and dependent feature(SalePrice)
- All The Numerical Variables
  - Temporal variables (Eg.)
  - Discrete features
  - Continuous features
  - Distribution of the Numerical Variables
- All Categorical Variables
  - Cardinality
  - Rare categorical variables
- Outliers
- Feature Scaling

Import Libraries

In [412…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

pd.pandas.set_option('display.max_columns', None)
```

Read data

In [413…
```
dataset = pd.read_csv('train.csv')
```

In [414…
```
dataset.shape
```

Out[414]:
```
(1460, 81)
```

In [415…
```
dataset.head()
```

Out[415]:

	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

In [416…
```
dataset.dtypes
```

Out[416]:
```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

ALL MISSING VALUES

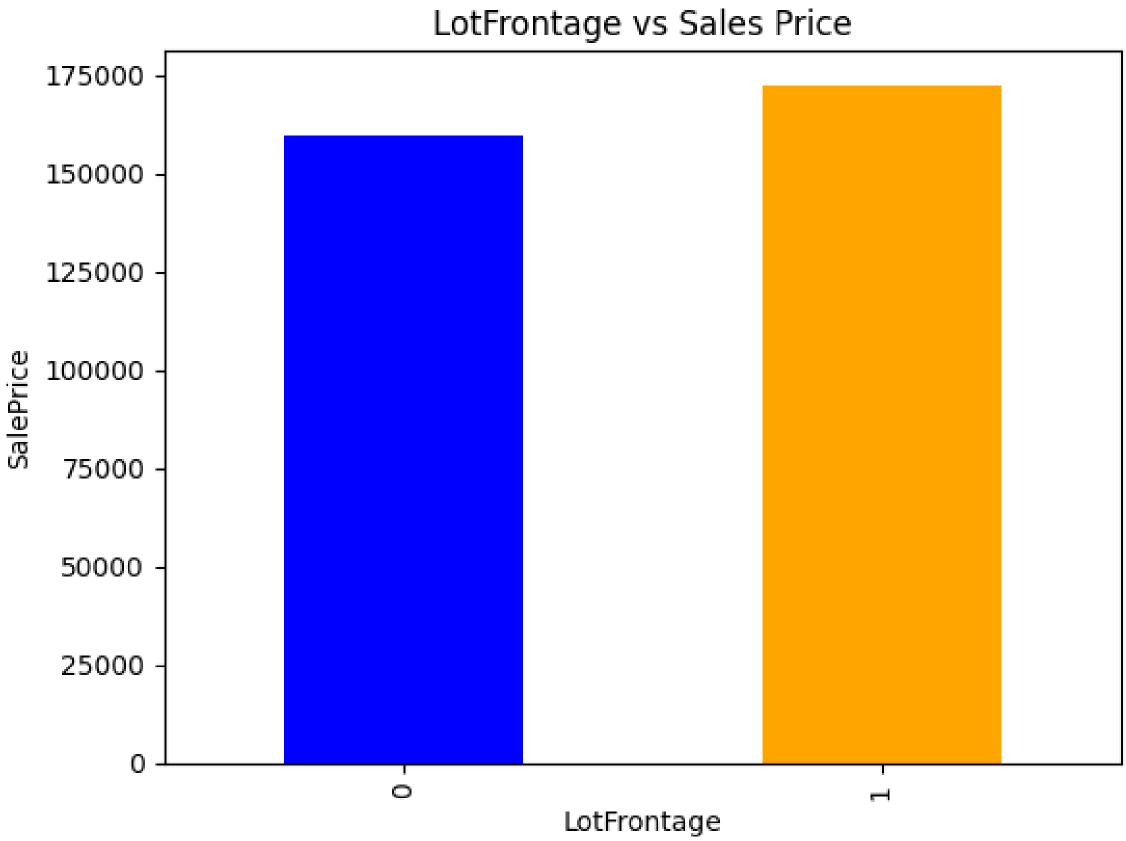
In [417…
```
# check percentage of NaN values in each feature

# features_with_na = List of all features containing null values
features_with_na = []
for feature in dataset.columns:
    if dataset[feature].isnull().sum() > 1:
        features_with_na.append(feature)

# print the feature name along with percentage of missing value
for feature in features_with_na:
  print(feature, "-" ,np.round(dataset[feature].isnull().mean(), 4), "% missing
```

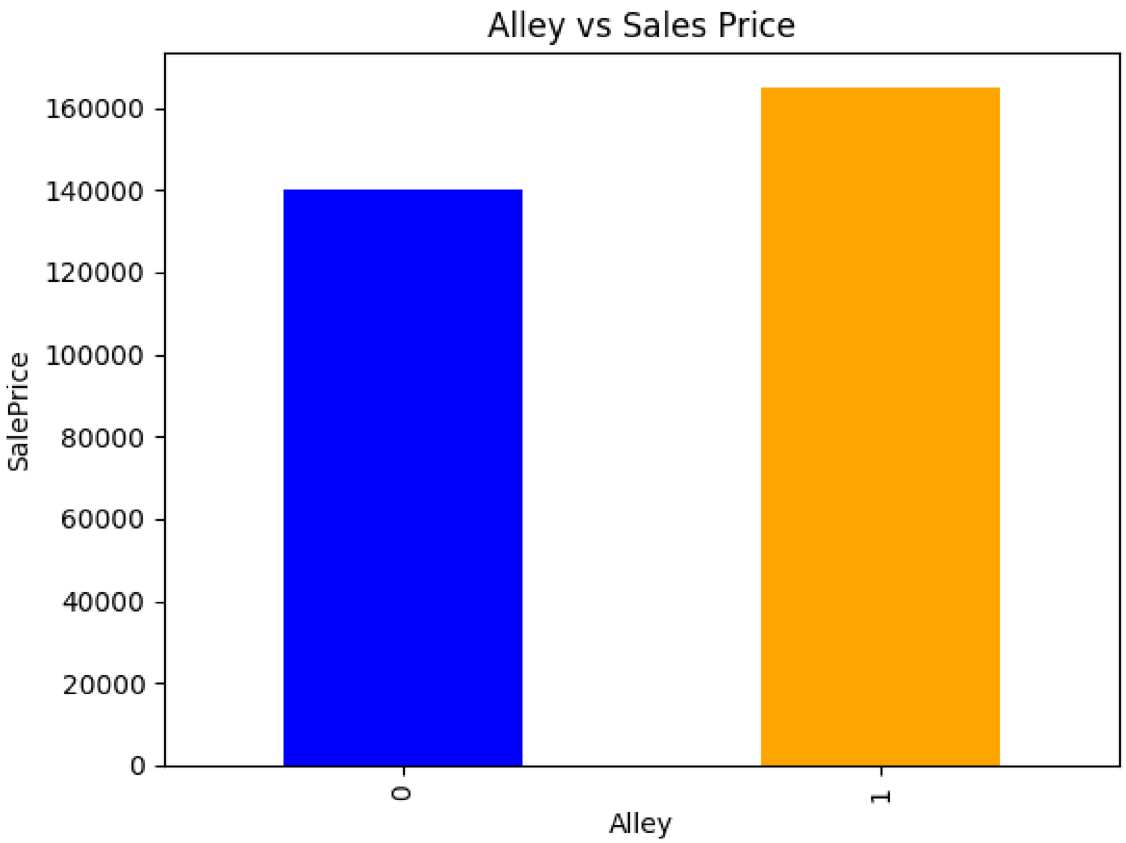
```
LotFrontage - 0.1774 % missing values
Alley - 0.9377 % missing values
MasVnrType - 0.0055 % missing values
MasVnrArea - 0.0055 % missing values
BsmtQual - 0.0253 % missing values
BsmtCond - 0.0253 % missing values
BsmtExposure - 0.026 % missing values
BsmtFinType1 - 0.0253 % missing values
BsmtFinType2 - 0.026 % missing values
FireplaceQu - 0.4726 % missing values
GarageType - 0.0555 % missing values
GarageYrBlt - 0.0555 % missing values
GarageFinish - 0.0555 % missing values
GarageQual - 0.0555 % missing values
GarageCond - 0.0555 % missing values
PoolQC - 0.9952 % missing values
Fence - 0.8075 % missing values
MiscFeature - 0.963 % missing values
```

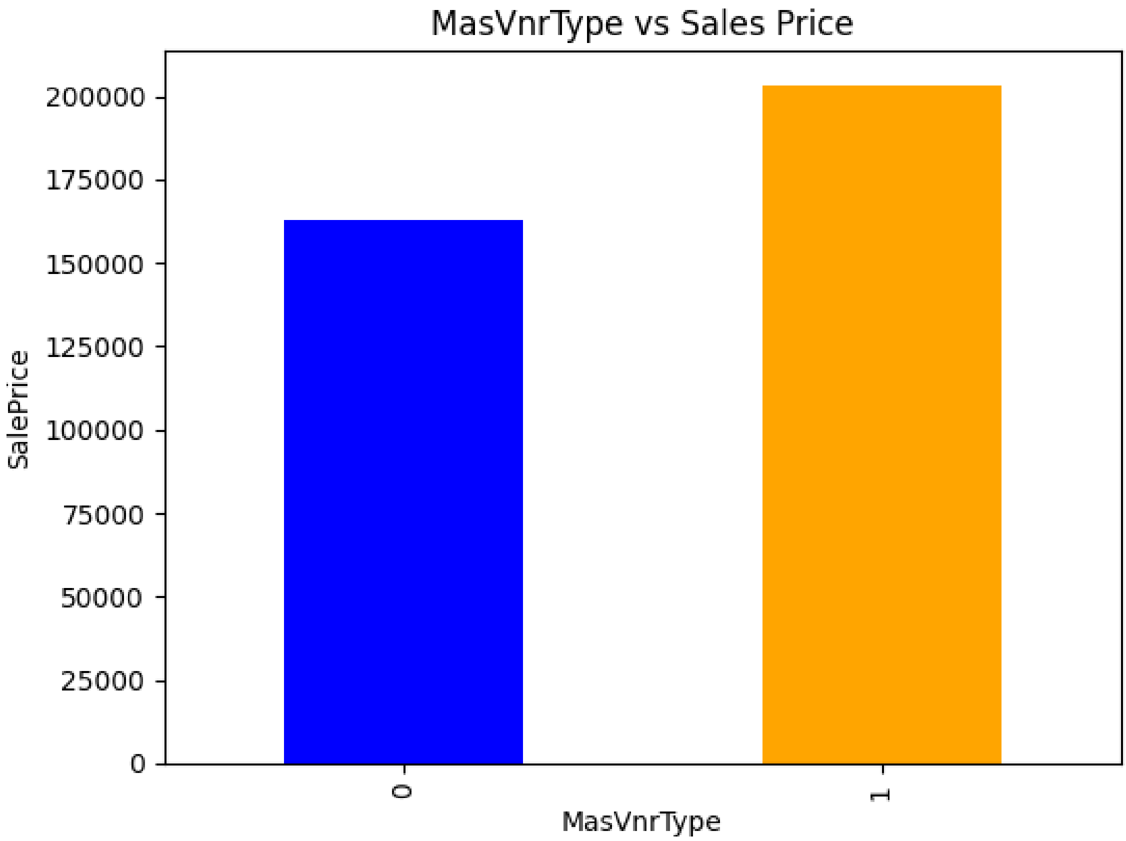
Since there are many missing values we need to find the relationship between the missing values and SalePrice

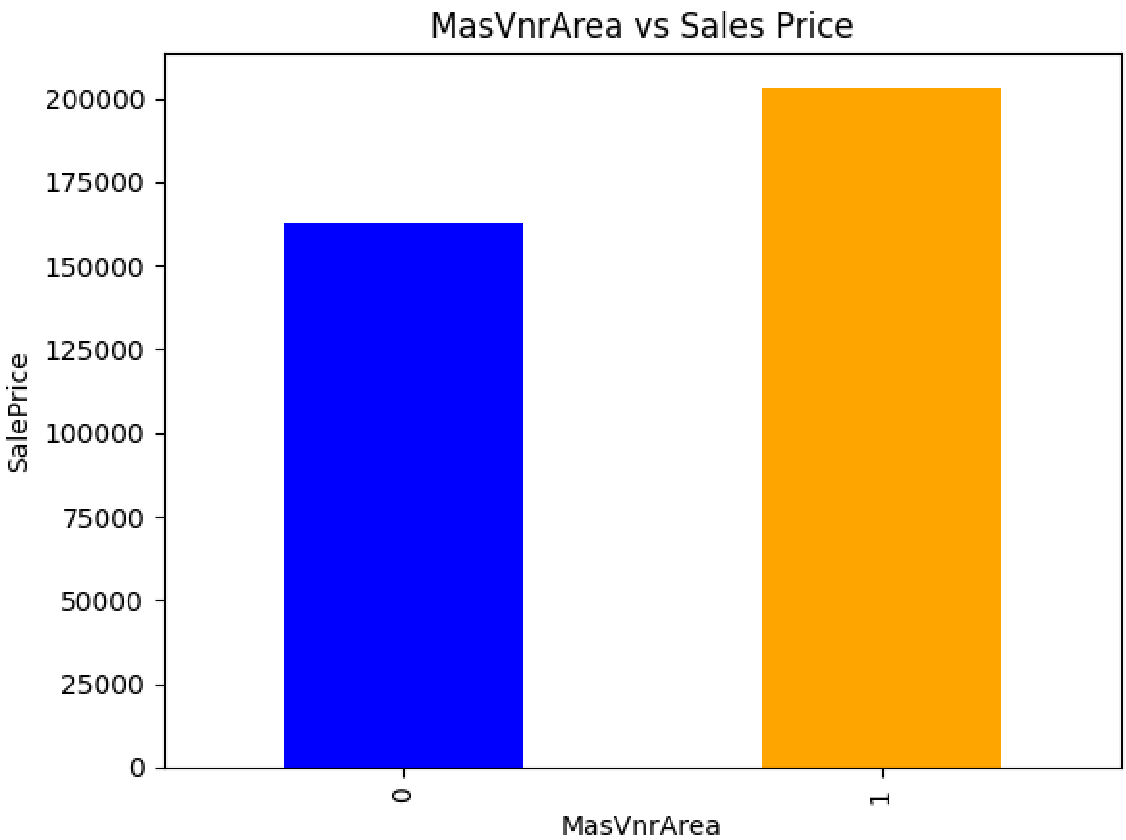
In [418…

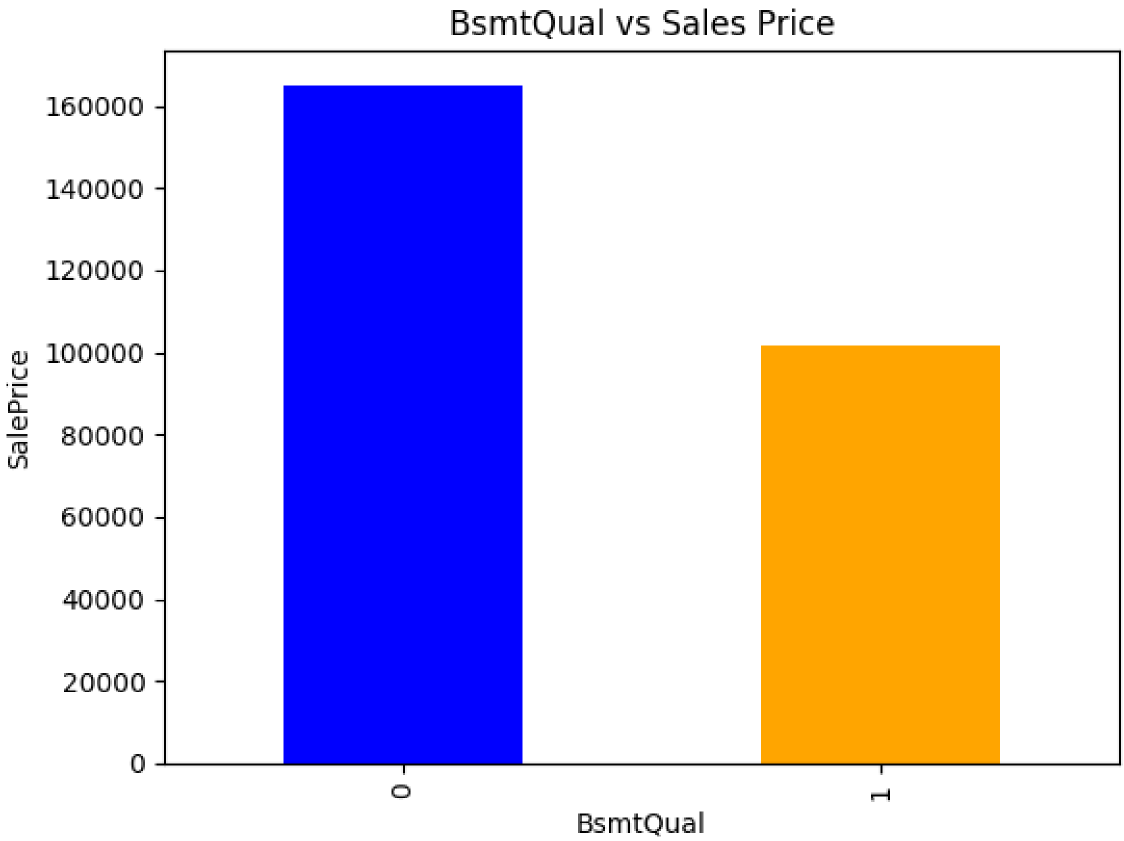
```
for feature in features_with_na:
  data = dataset.copy()

  # 1: missing value present; 0: otherwise; does this to the whole copy of datas
  data[feature] = np.where(data[feature].isnull(), 1, 0)

  median_sale_price_by_feature = data.groupby(feature)['SalePrice'].median()
  median_sale_price_by_feature.plot.bar(color=['blue', 'orange']) # blue = missi
  plt.xlabel(feature)
  plt.ylabel("SalePrice")
  plt.title(feature + " vs Sales Price")
  plt.show()
```

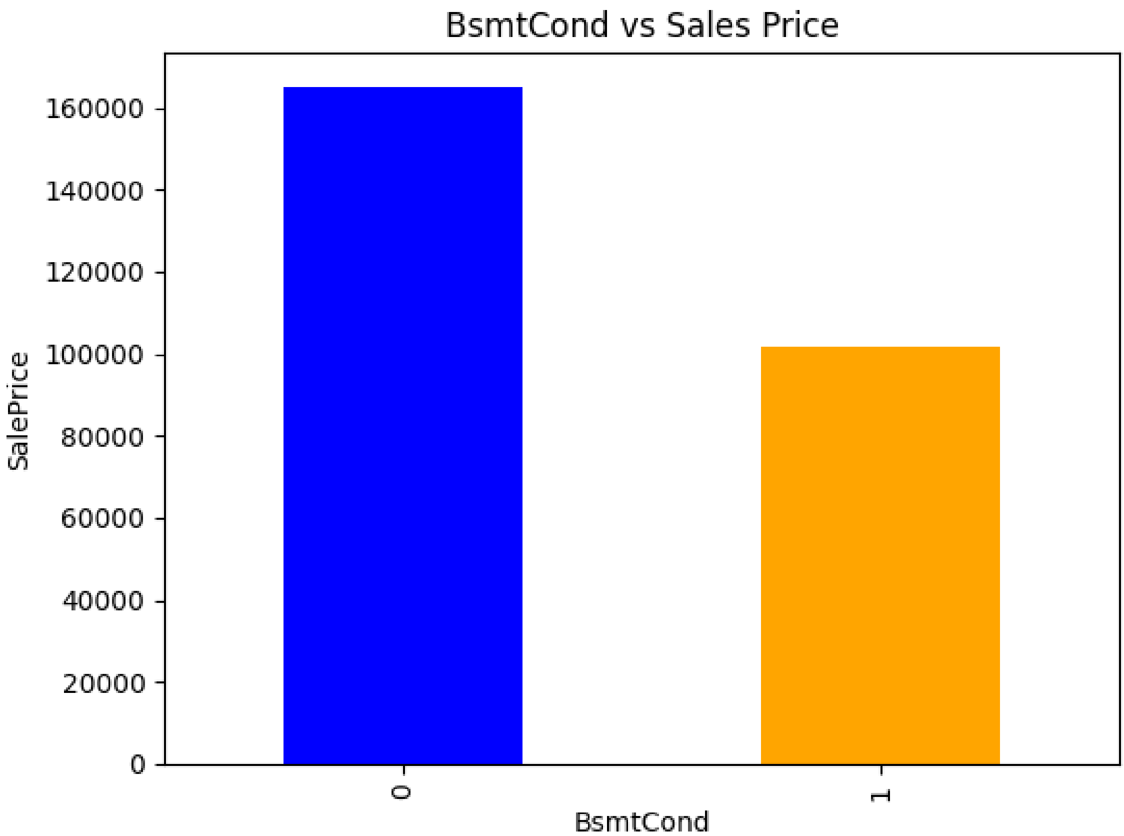

LotFrontage vs Sales Price
175000
150000
125000
100000
75000
50000
25000
0
SalePrice
0
1
LotFrontage

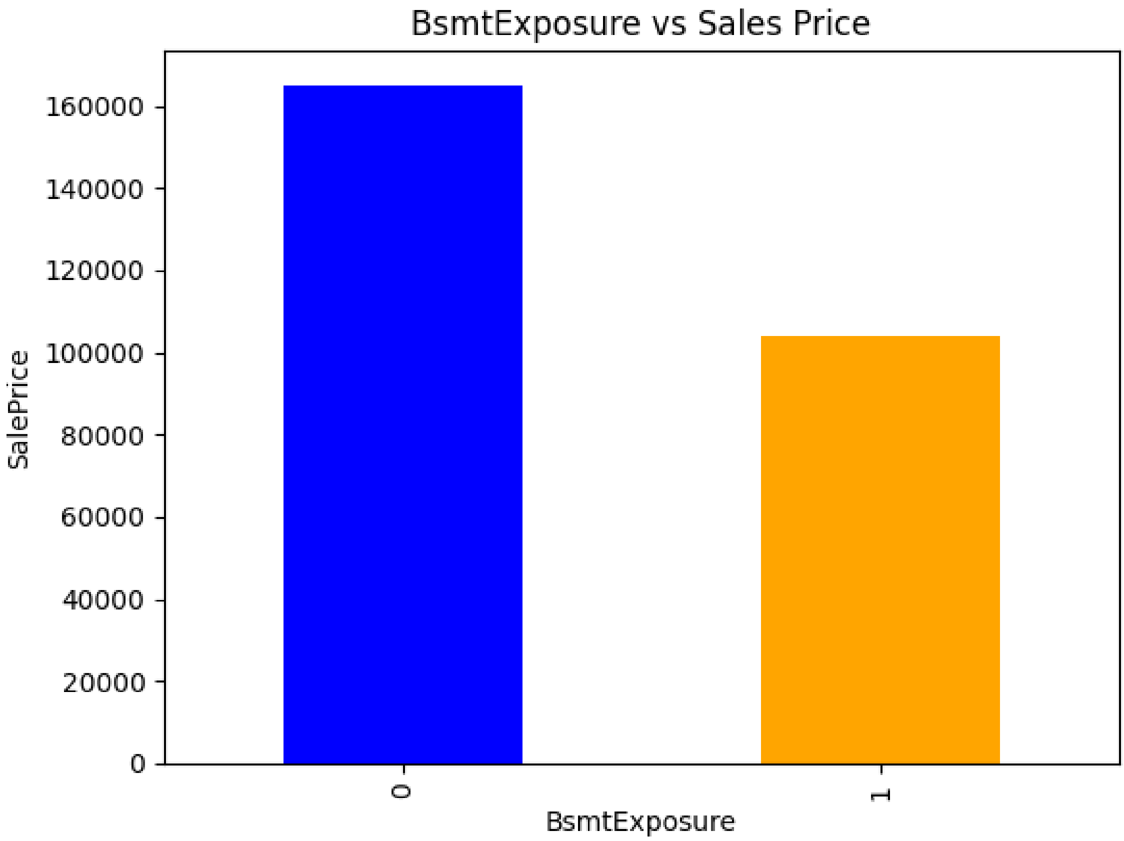

Alley vs Sales Price
160000
140000
120000
100000
80000
60000
40000
20000
0
SalePrice
0
1
Alley

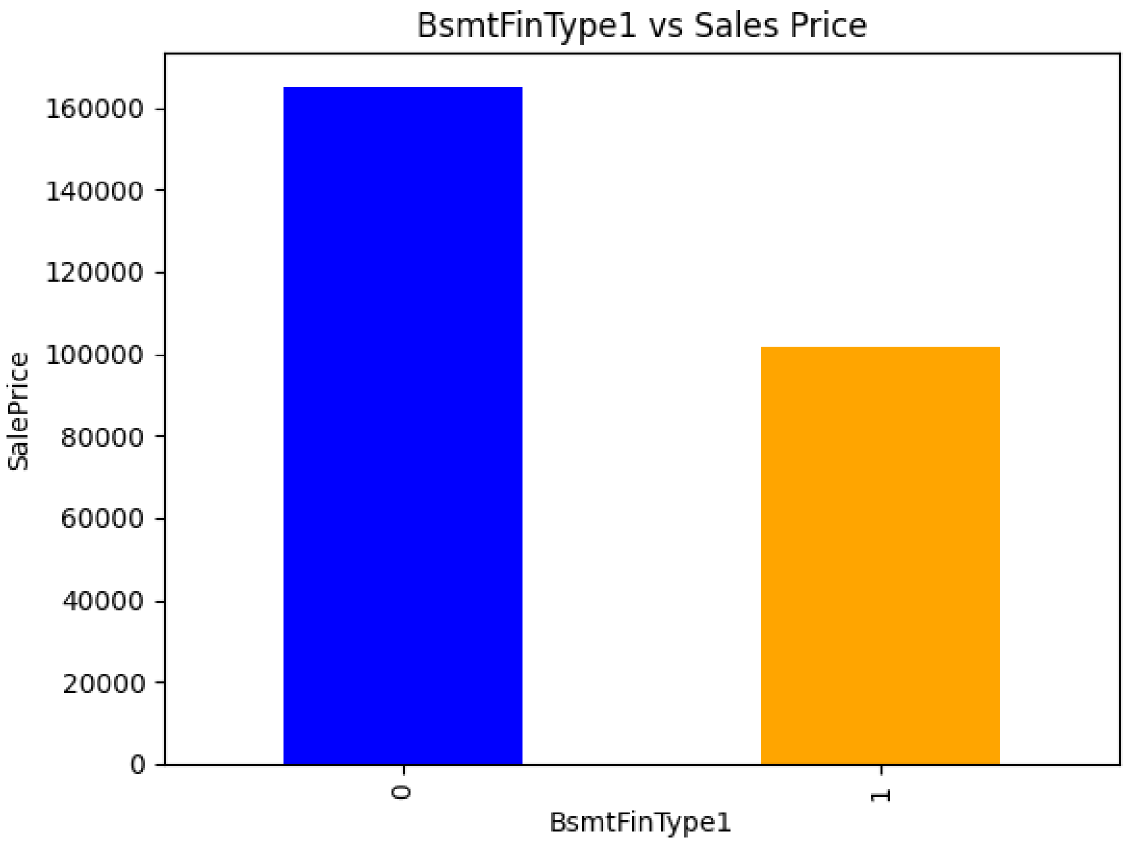

MasVnrType vs Sales Price
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrType

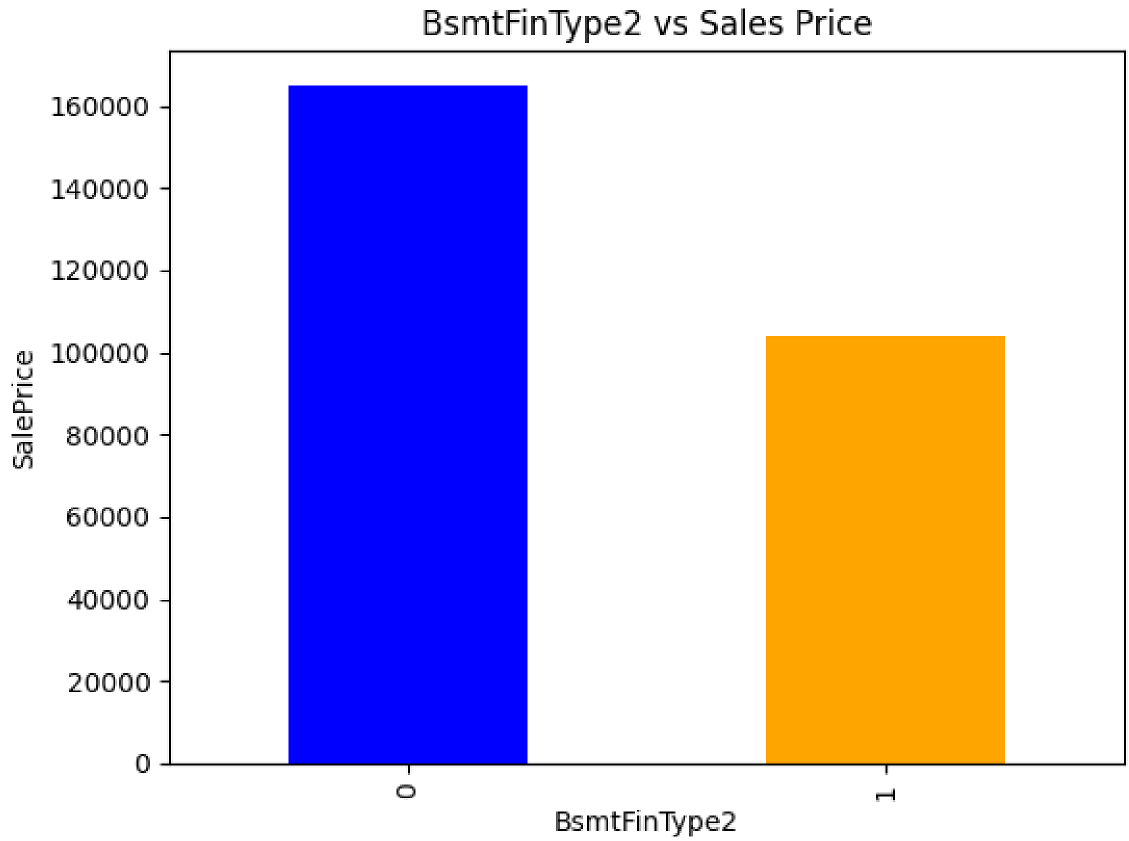

MasVnrArea vs Sales Price
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrArea

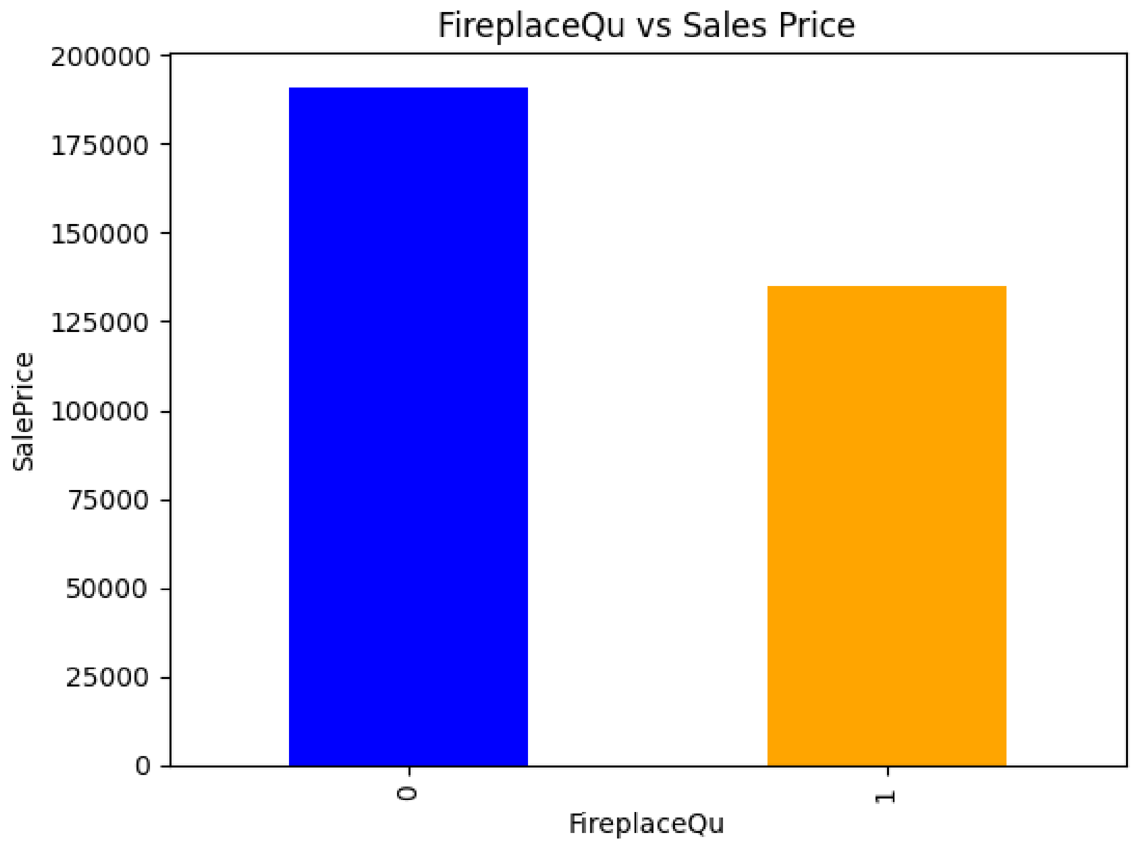

BsmtQual vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtQual

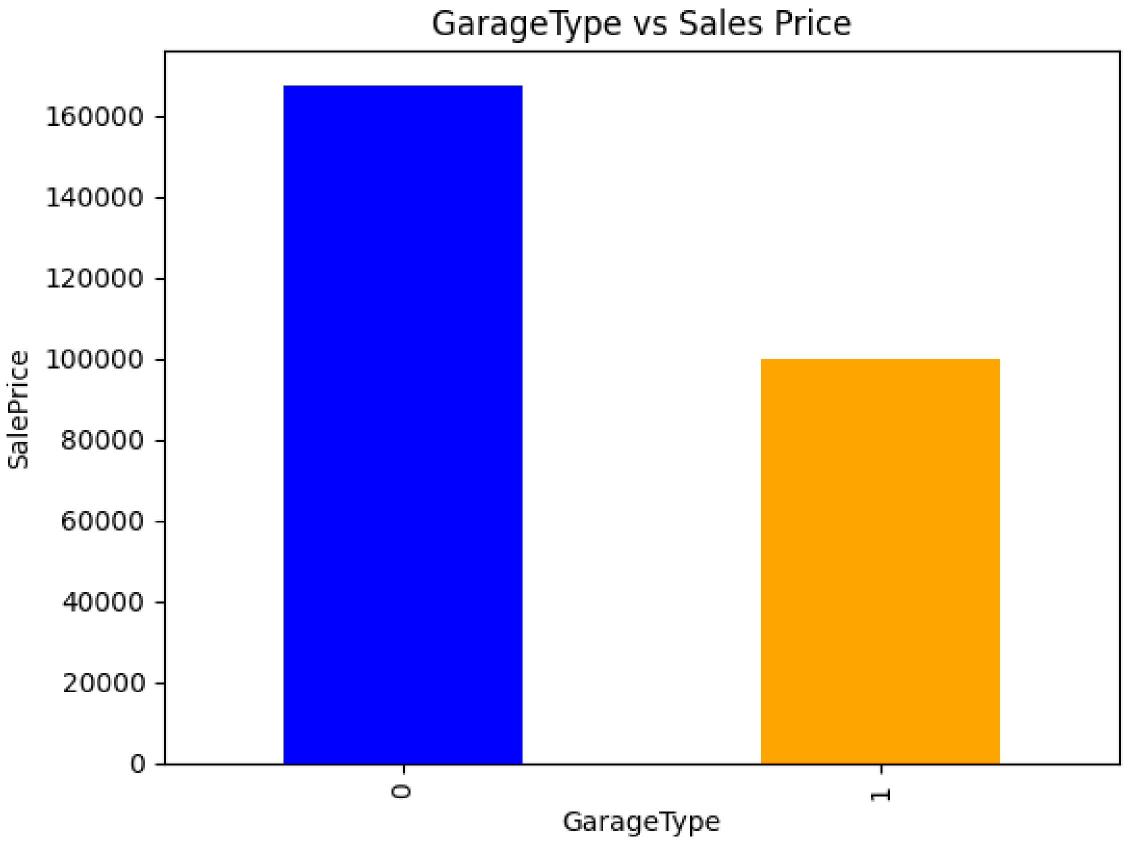

BsmtCond vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtCond

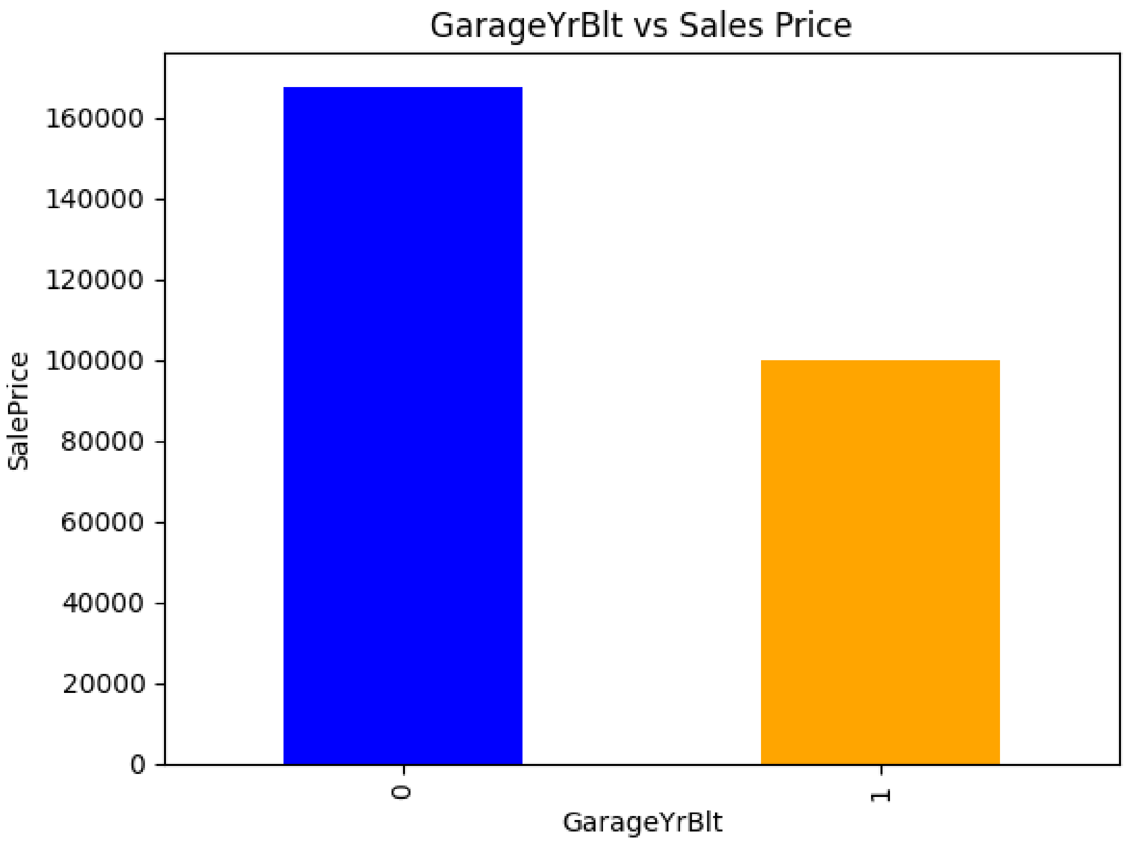

BsmtExposure vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtExposure

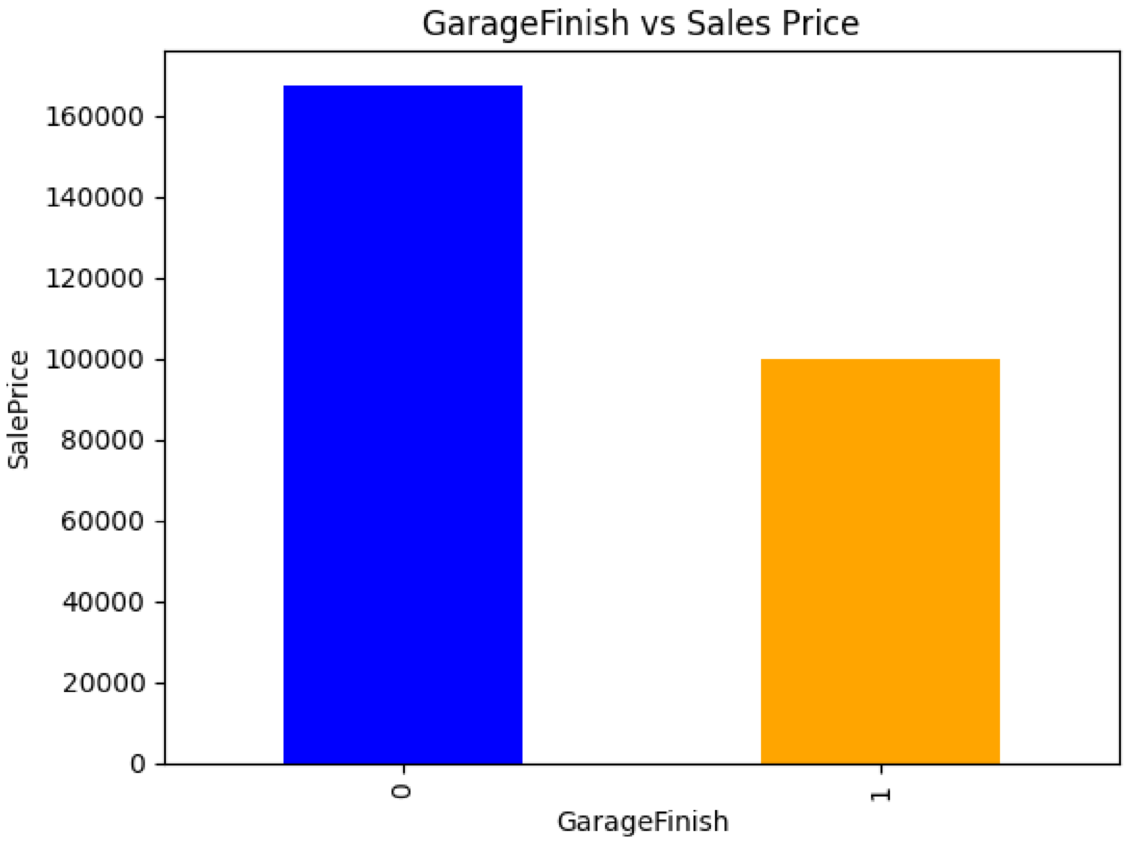

BsmtFinType1 vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtFinType1


BsmtFinType2 vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtFinType2

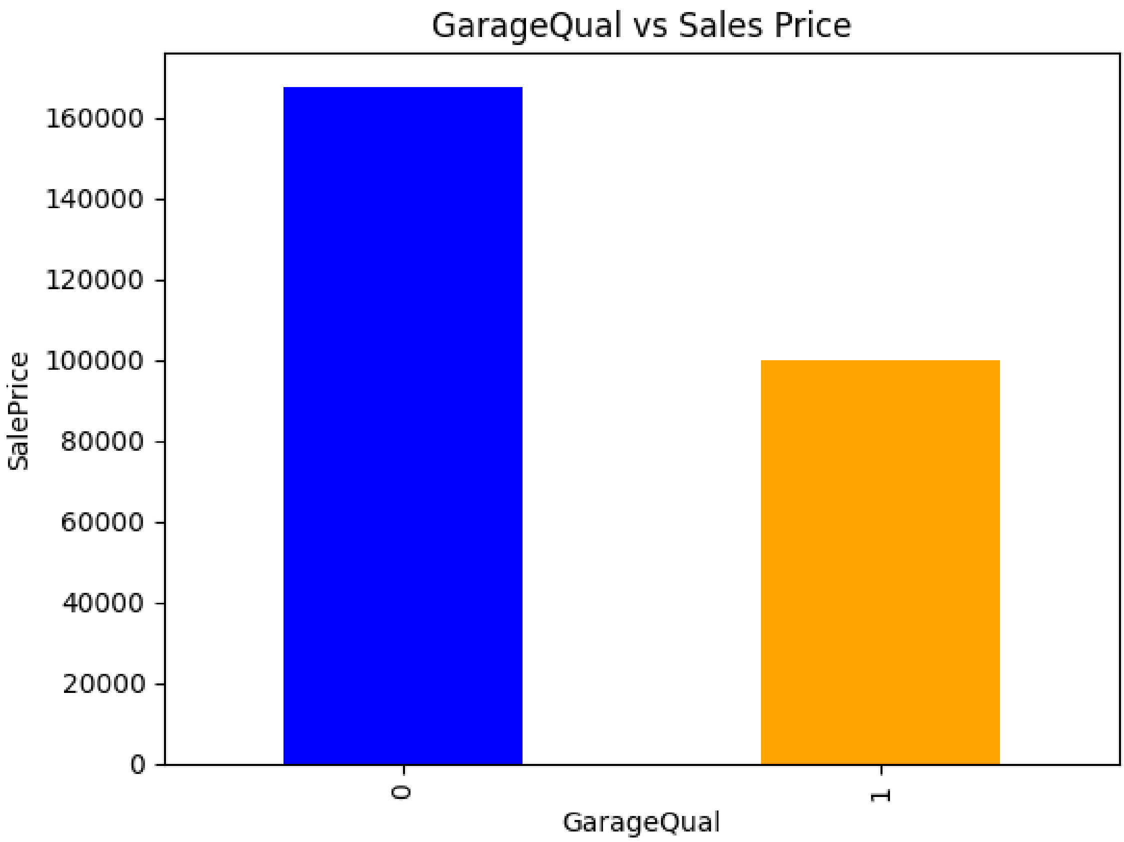

FireplaceQu vs Sales Price
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
FireplaceQu


GarageType vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageType

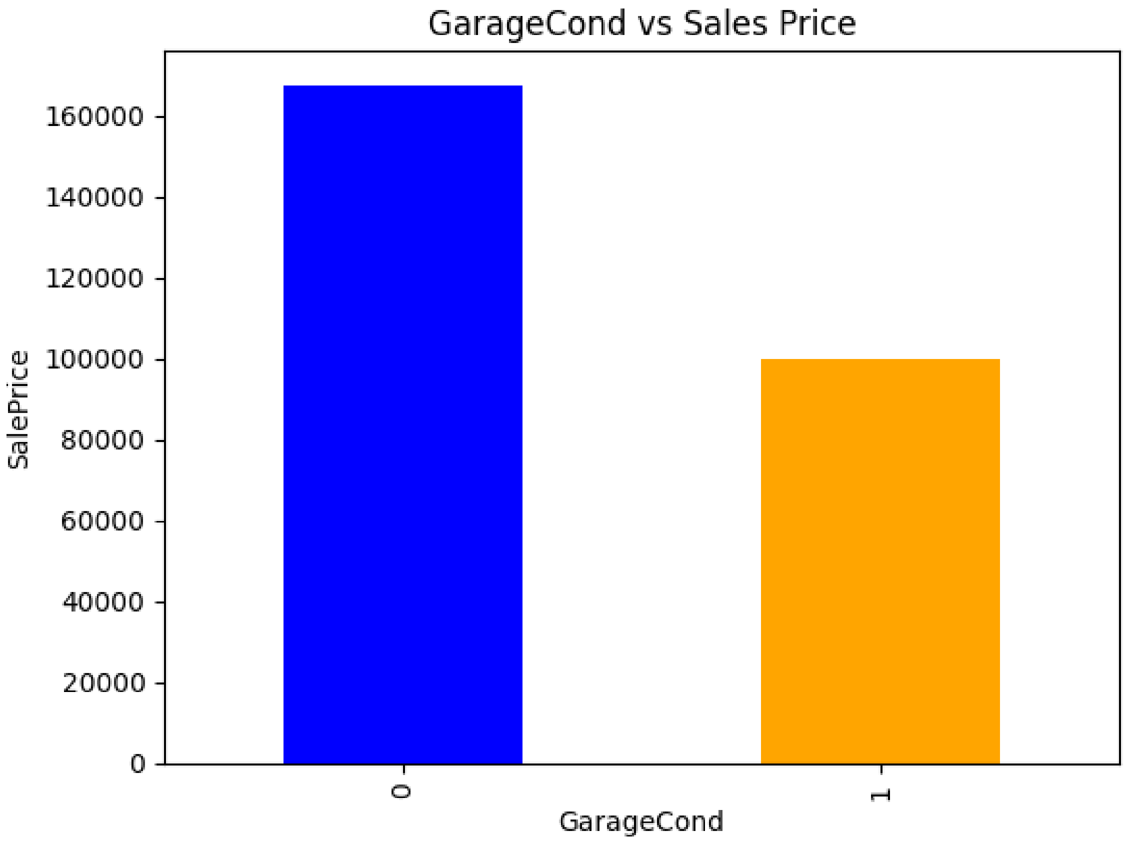

GarageYrBlt vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageYrBlt

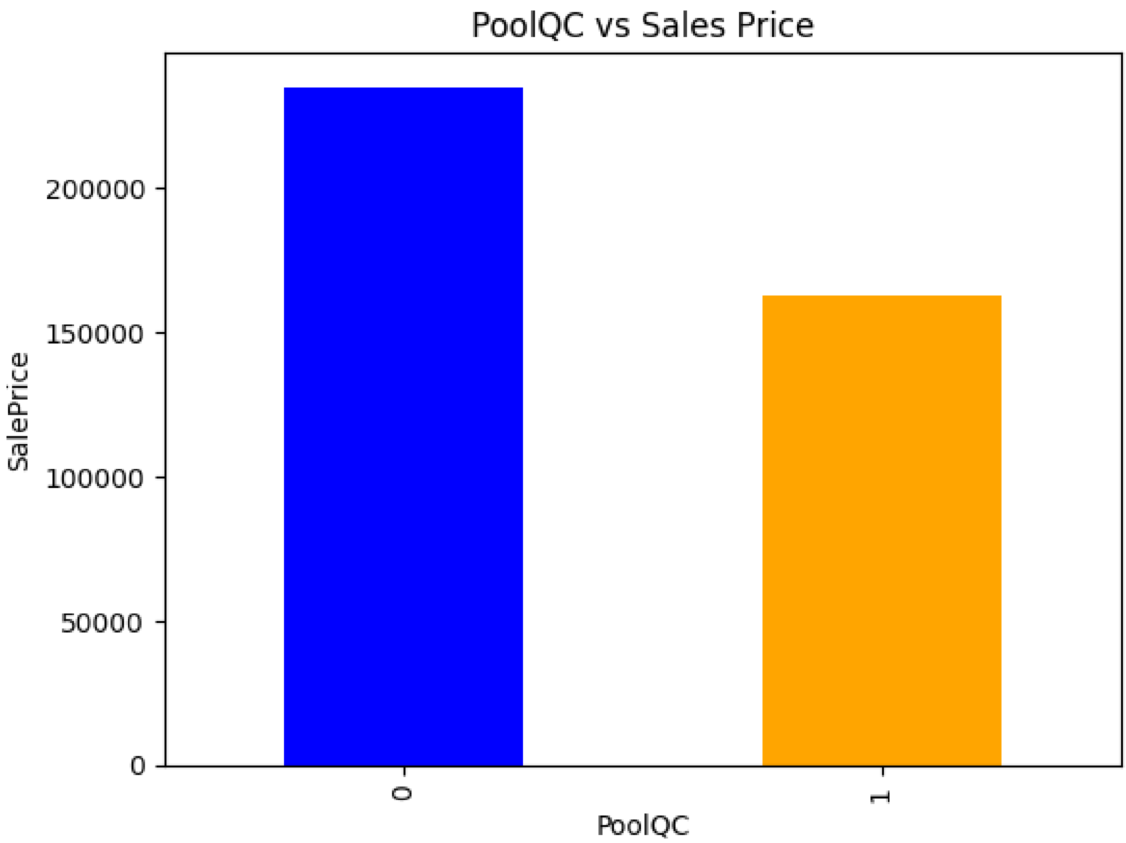


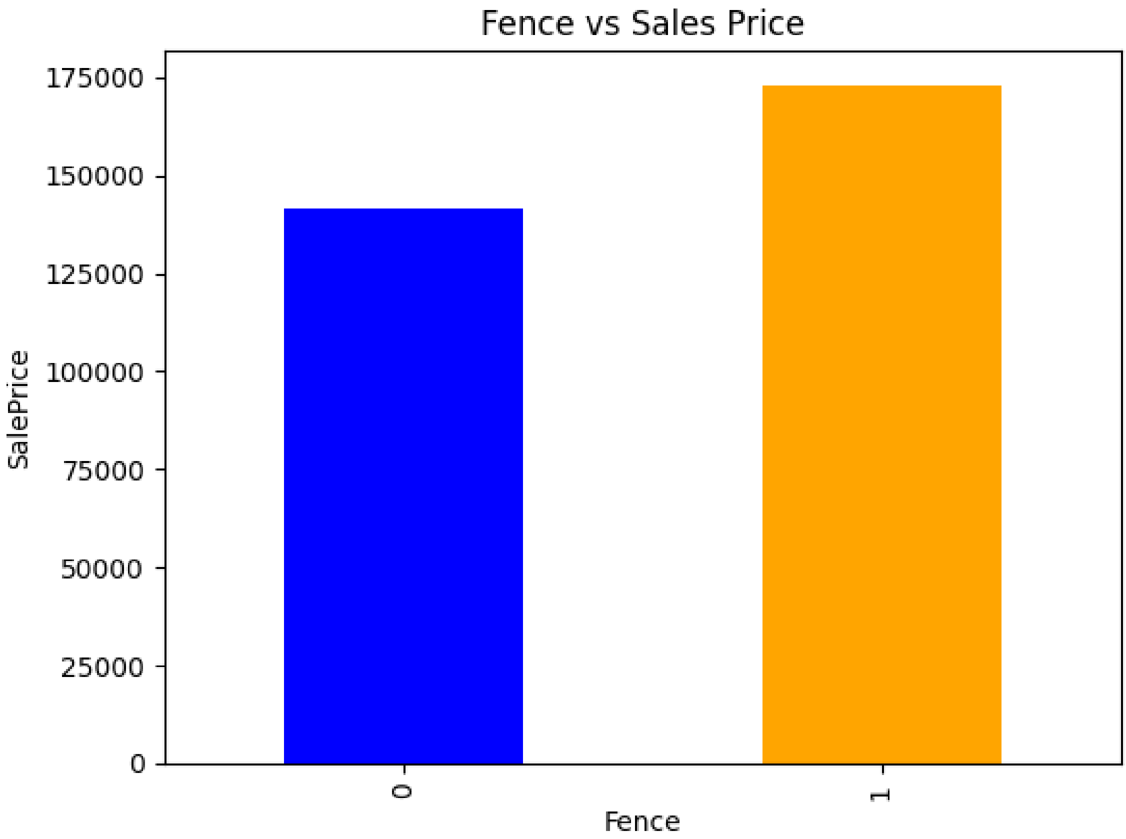
GarageFinish vs Sales Price
SalePrice
GarageFinish
0
1

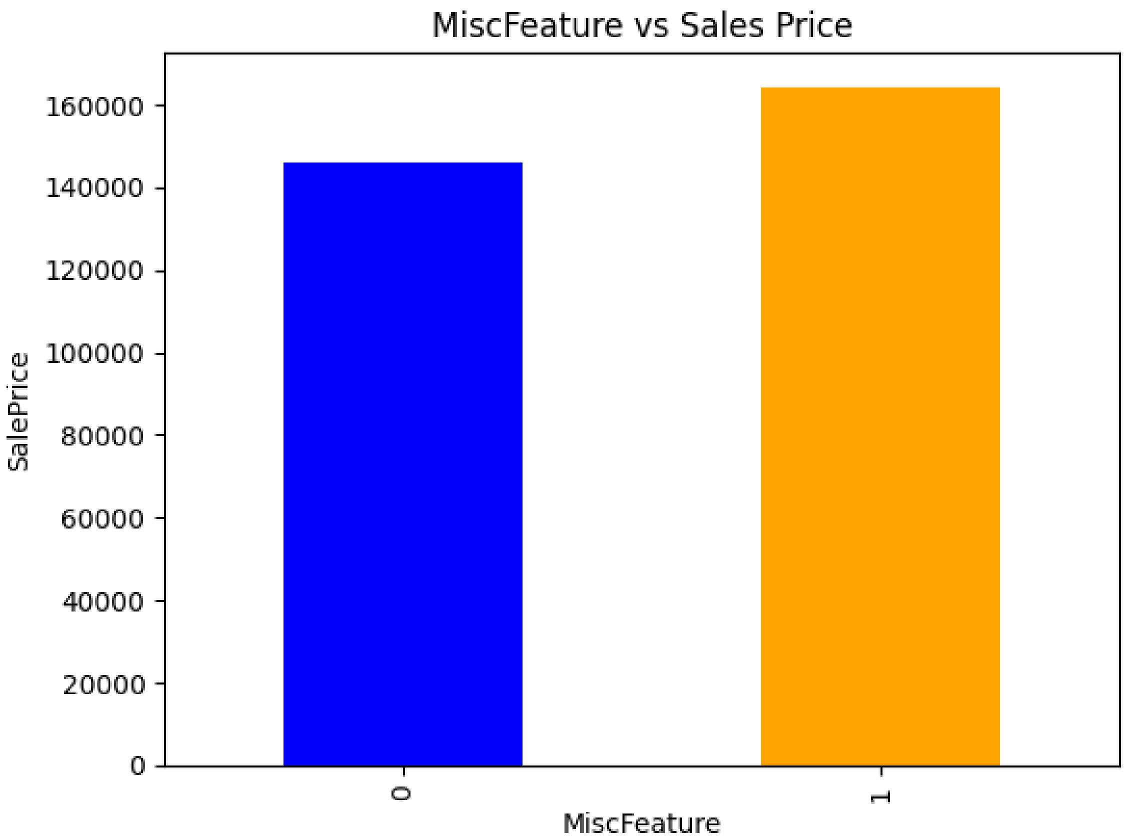


GarageQual vs Sales Price
SalePrice
GarageQual
0
1


GarageCond vs Sales Price
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageCond


PoolQC vs Sales Price
SalePrice
200000
150000
100000
50000
0
0
1
PoolQC





Here the relation between the missing values and the sale price is clearly visible. So We need to replace these NaN values with something meaningful.

CATEGORICAL MISSING VALUES

In [419…

```
# find nan/missing values in categorical features
cat_features_nan = []
for feature in dataset.columns:
  if dataset[feature].isnull().sum() > 1 and dataset[feature].dtypes == 'O':
    cat_features_nan.append(feature)

# print the feature name along with percentage of missing value
for feature in cat_features_nan:
  print(feature, "-" ,np.round(dataset[feature].isnull().mean(), 4), "% missing

## Replace missing value with a new label
def replace_cat_feature(dataset, cat_features_nan):
    data = dataset.copy()
    data[cat_features_nan] = data[cat_features_nan].fillna('Missing')
    return data

dataset = replace_cat_feature(dataset, cat_features_nan)
dataset[cat_features_nan].isnull().sum()
```

```
Alley - 0.9377 % missing values
MasVnrType - 0.0055 % missing values
BsmtQual - 0.0253 % missing values
BsmtCond - 0.0253 % missing values
BsmtExposure - 0.026 % missing values
BsmtFinType1 - 0.0253 % missing values
BsmtFinType2 - 0.026 % missing values
FireplaceQu - 0.4726 % missing values
GarageType - 0.0555 % missing values
GarageFinish - 0.0555 % missing values
GarageQual - 0.0555 % missing values
GarageCond - 0.0555 % missing values
PoolQC - 0.9952 % missing values
Fence - 0.8075 % missing values
MiscFeature - 0.963 % missing values
```

Out[419]:

```
Alley           0
MasVnrType      0
BsmtQual        0
BsmtCond        0
BsmtExposure    0
BsmtFinType1    0
BsmtFinType2    0
FireplaceQu     0
GarageType      0
GarageFinish    0
GarageQual      0
GarageCond      0
PoolQC          0
Fence           0
MiscFeature     0
dtype: int64
```

In [420…

```
dataset.head()
```

Out[420]:

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

NUMERICAL MISSING VALUES

In [421…

```
# find list of all numerical features
numerical_features = []
for feature in dataset.columns:
  if dataset[feature].dtypes != 'O':
    numerical_features.append(feature)
    print(feature)

dataset[numerical_features].head()
```

```
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
SalePrice
```

Out[421]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
0	1	60	65.0	8450	7	5	2003	2003
1	2	20	80.0	9600	6	8	1976	1976
2	3	60	68.0	11250	7	5	2001	2002
3	4	70	60.0	9550	7	5	1915	1970
4	5	60	84.0	14260	8	5	2000	2000

In [422…

```
# find nan/missing values in numerical features
num_features_nan = []
for feature in dataset.columns:
  if feature in numerical_features and dataset[feature].isnull().sum() > 1:
    num_features_nan.append(feature)

# print the feature name along with percentage of missing value
```

```
for feature in num_features_nan:
  print(feature, "-" ,np.round(dataset[feature].isnull().mean(), 4), "% missing
```

```
LotFrontage - 0.1774 % missing values
MasVnrArea - 0.0055 % missing values
GarageYrBlt - 0.0555 % missing values
```

In [423…

```
# Replacing the numerical Missing Values

for feature in num_features_nan:
    # replace by using median since there could be outliers
    median_value = dataset[feature].median()

    # create a new feature to capture NaN values
    dataset[feature + '_NaN'] = np.where(dataset[feature].isnull(), 1, 0)
    dataset[feature].fillna(median_value,inplace = True)

dataset[num_features_nan].isnull().sum()
```

Out[423]:

```
LotFrontage    0
MasVnrArea     0
GarageYrBlt    0
dtype: int64
```

TEMPORAL VARIABLES (Eg: Datetime Variables)

From the Dataset we have 4 year variables. We have to extract information from the datetime variables like number of years or number of days. One example in this specific scenario can be the difference between the year the house was built and the year the house was sold.

In [424…

```
# find list of all features containing year data
year_feature = []
for feature in numerical_features:
  if 'Yr' in feature or 'Year' in feature:
    year_feature.append(feature)

year_feature
```

Out[424]:

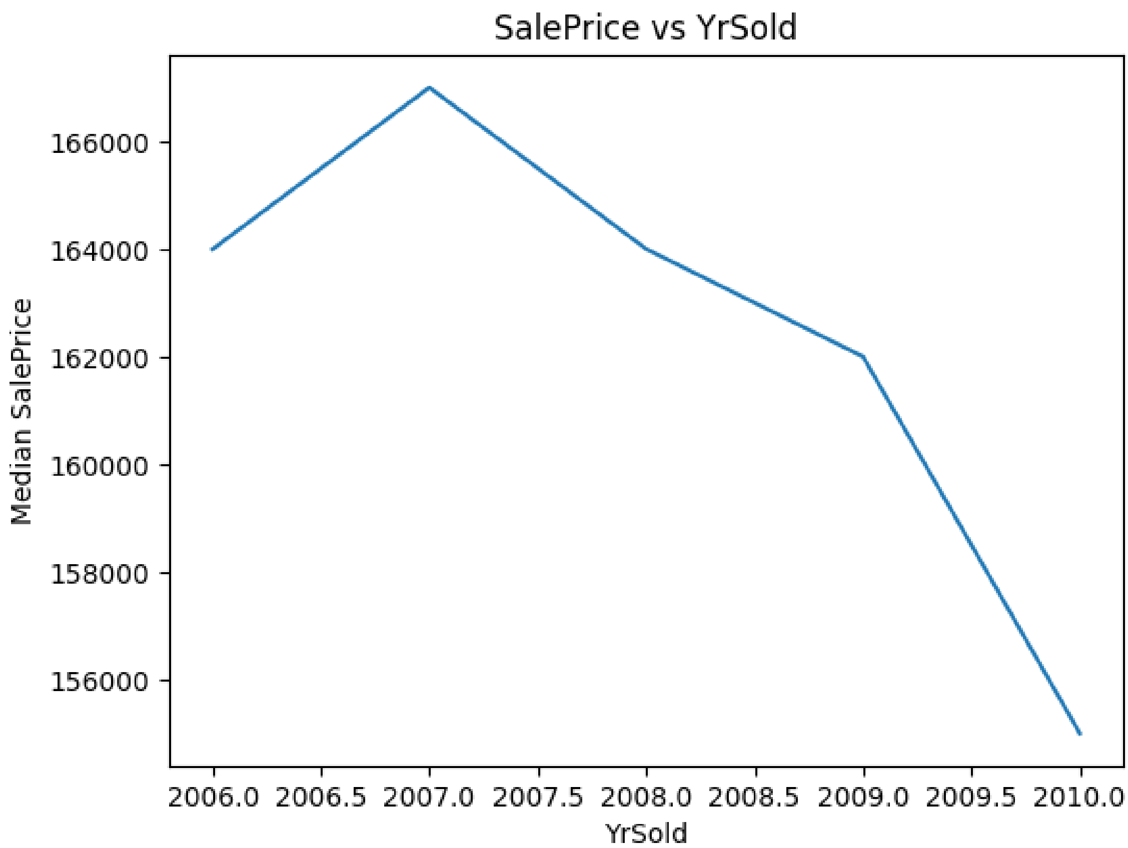
```
['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
```

In [425…

```
# check whether there is a relation between year the house is sold and the sales
print(dataset.groupby('YrSold')['SalePrice'].median().to_frame())

dataset.groupby('YrSold')['SalePrice'].median().plot()
plt.xlabel('YrSold')
plt.ylabel('Median SalePrice')
plt.title("SalePrice vs YrSold")
```

```
        SalePrice
YrSold
2006     163995.0
2007     167000.0
2008     164000.0
2009     162000.0
2010     155000.0
```

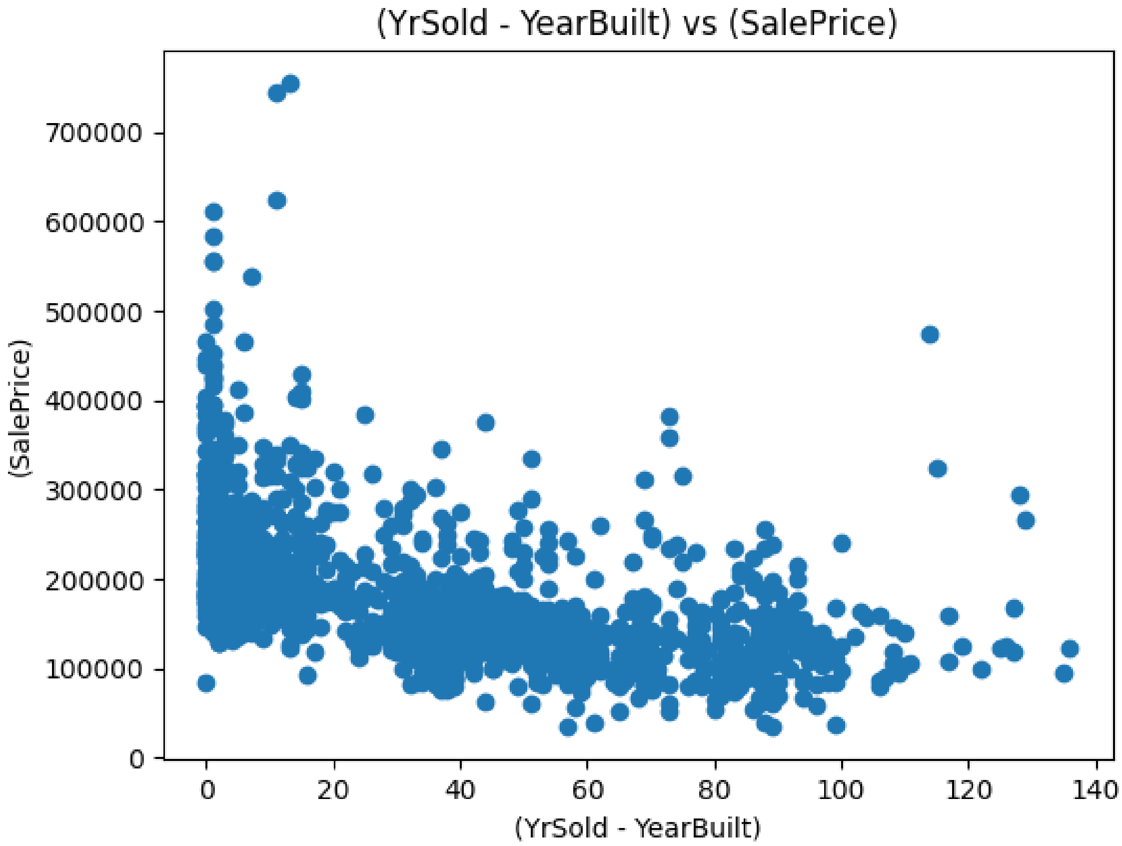
Out[425]: Text(0.5, 1.0, 'SalePrice vs YrSold')

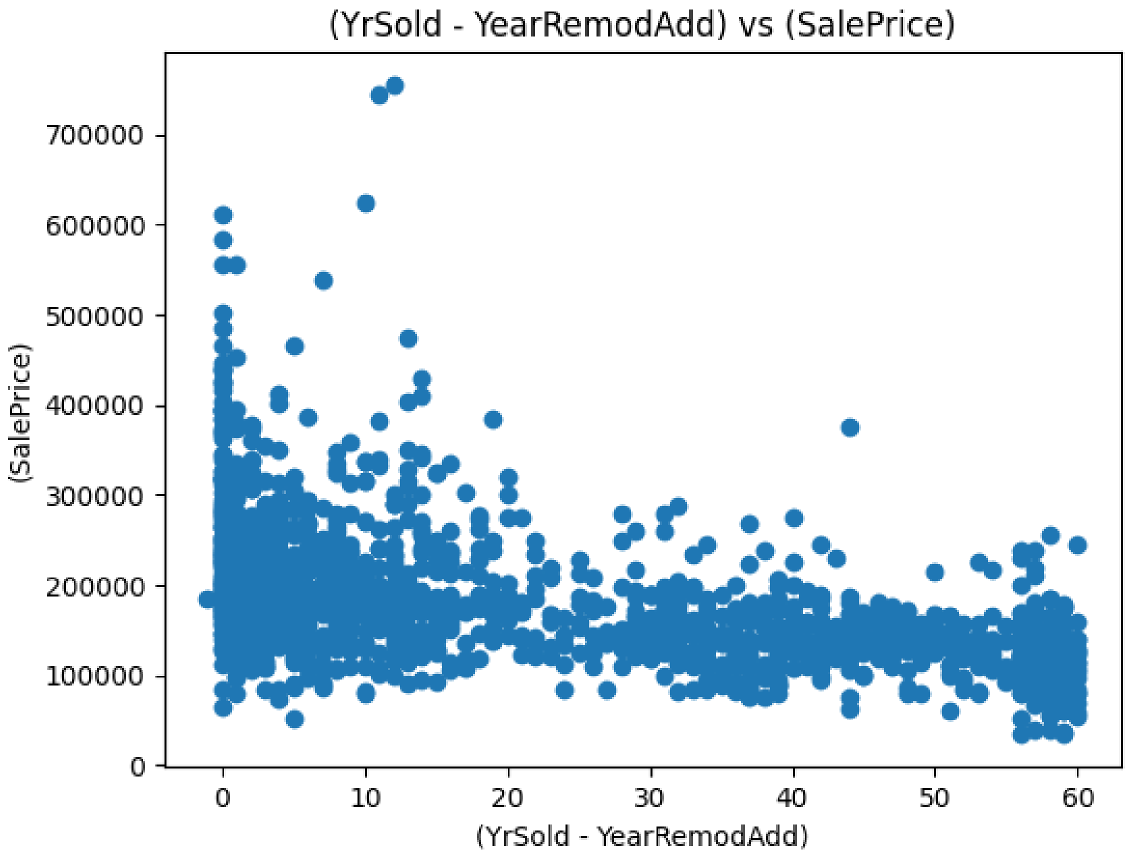


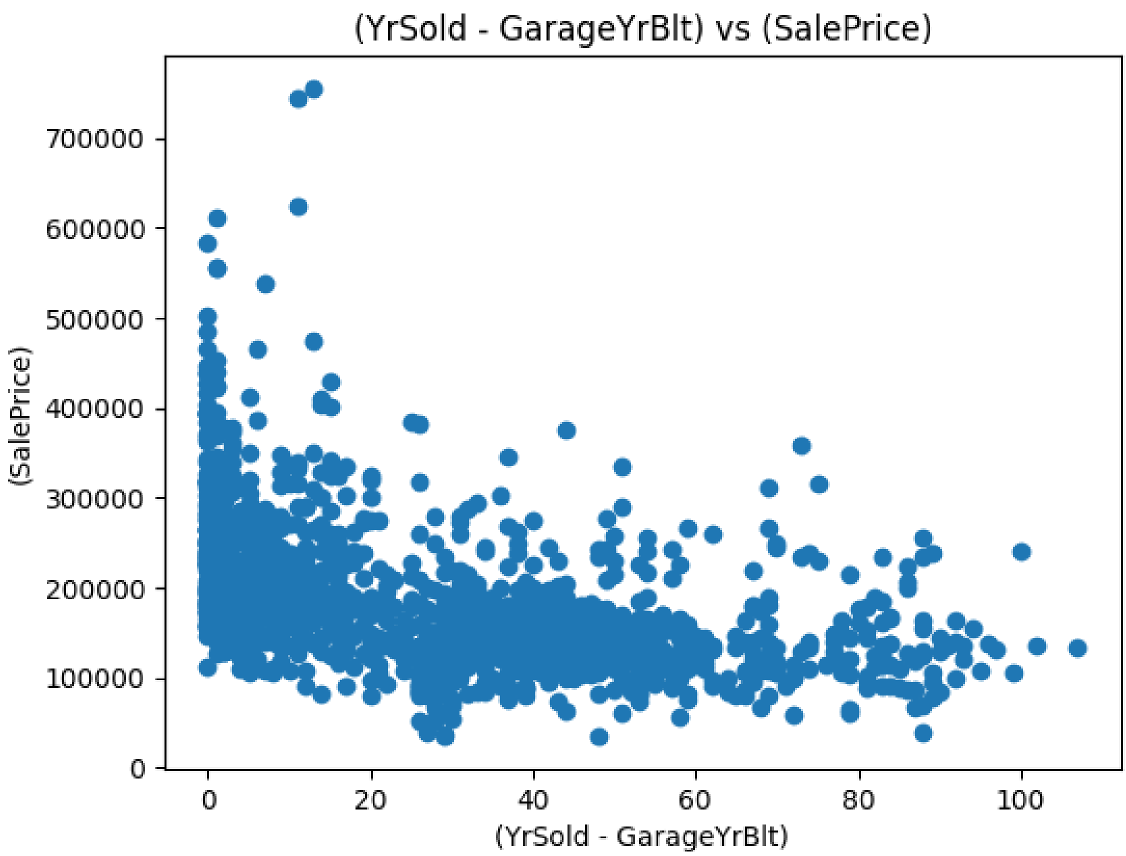
As YrSold increases, the price drops.

In [426…

```
# compare the (['YrSold'] - rest of the features containing year data) with (Sal
for feature in year_feature:
  if feature != 'YrSold':
    data = dataset.copy()
    data[feature] = data['YrSold'] - data[feature]

    plt.scatter(data[feature], data['SalePrice'])
    plt.xlabel('(YrSold - ' + feature + ')')
    plt.ylabel('(SalePrice)')
    plt.title('(YrSold - ' + feature + ') vs (SalePrice)')
    plt.show()
```


(YrSold - YearBuilt) vs (SalePrice)
(SalePrice)
(YrSold - YearBuilt)
0
100000
200000
300000
400000
500000
600000
700000
0
20
40
60
80
100
120
140


(YrSold - YearRemodAdd) vs (SalePrice)
(SalePrice)
(YrSold - YearRemodAdd)
0
100000
200000
300000
400000
500000
600000
700000
0
10
20
30
40
50
60



In [427…

```
for feature in ['YearBuilt','YearRemodAdd','GarageYrBlt']:
    dataset[feature] = dataset['YrSold'] - dataset[feature] # gives number of yr
```

In [428…

```
dataset[['YearBuilt','YearRemodAdd','GarageYrBlt']].head()
```

Out[428]:

	YearBuilt	YearRemodAdd	GarageYrBlt
0	5	5	5.0
1	31	31	31.0
2	7	6	7.0
3	91	36	8.0
4	8	8	8.0

DISCRETE VARIABLES

In [429…

```
# find all discrete features
discrete_feature = []
for feature in numerical_features:
  if len(dataset[feature].unique()) < 25 and feature not in year_feature + ['Id'
    discrete_feature.append(feature)

len(discrete_feature)
```

Out[429]: 17

In [430…

```
discrete_feature
```

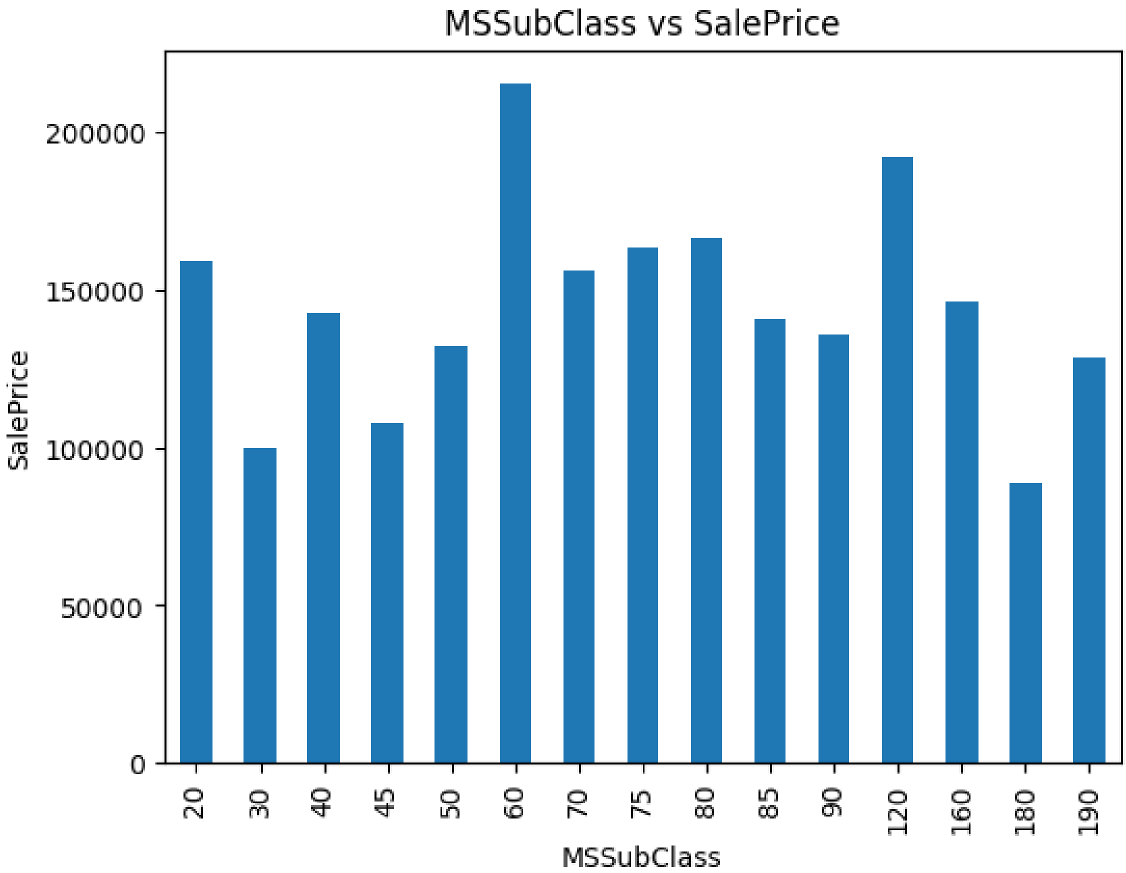
Out[430]:
```
['MSSubClass',
 'OverallQual',
 'OverallCond',
 'LowQualFinSF',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'BedroomAbvGr',
 'KitchenAbvGr',
 'TotRmsAbvGrd',
 'Fireplaces',
 'GarageCars',
 '3SsnPorch',
 'PoolArea',
 'MiscVal',
 'MoSold']
```

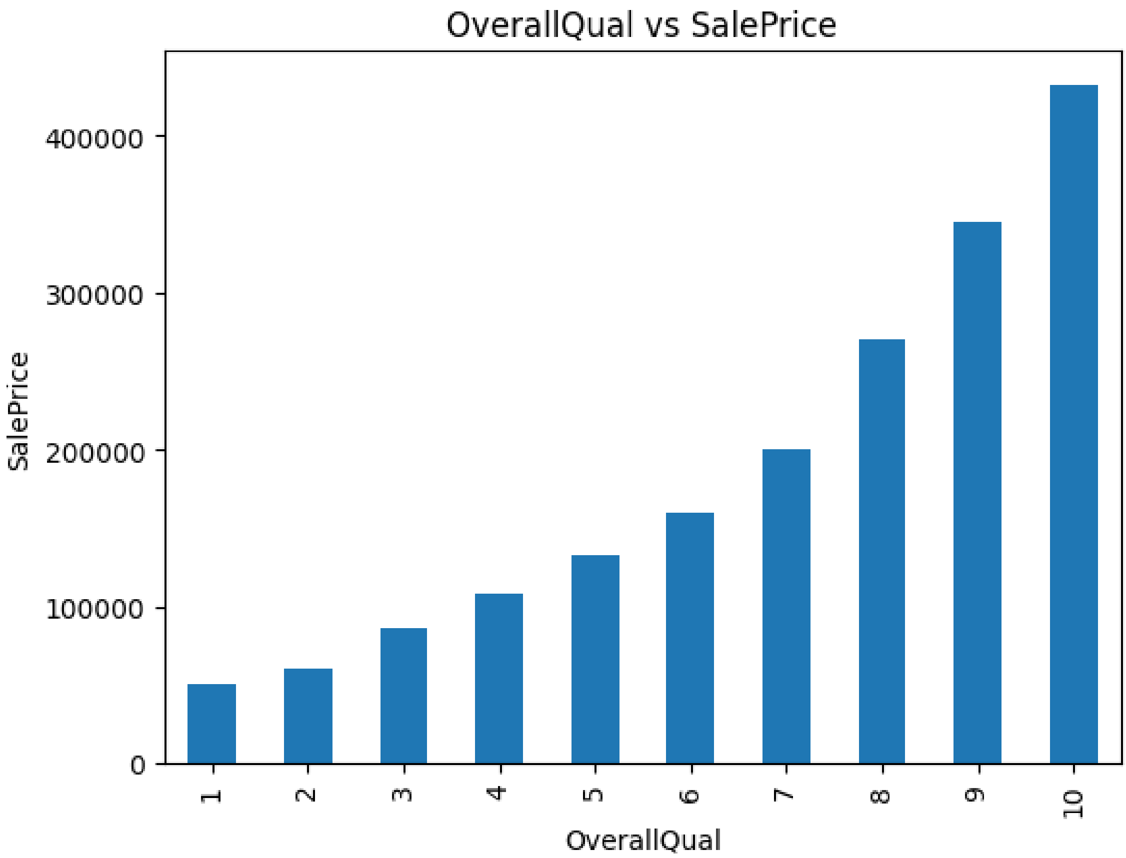
In [431…
```
dataset[discrete_feature].head()
```

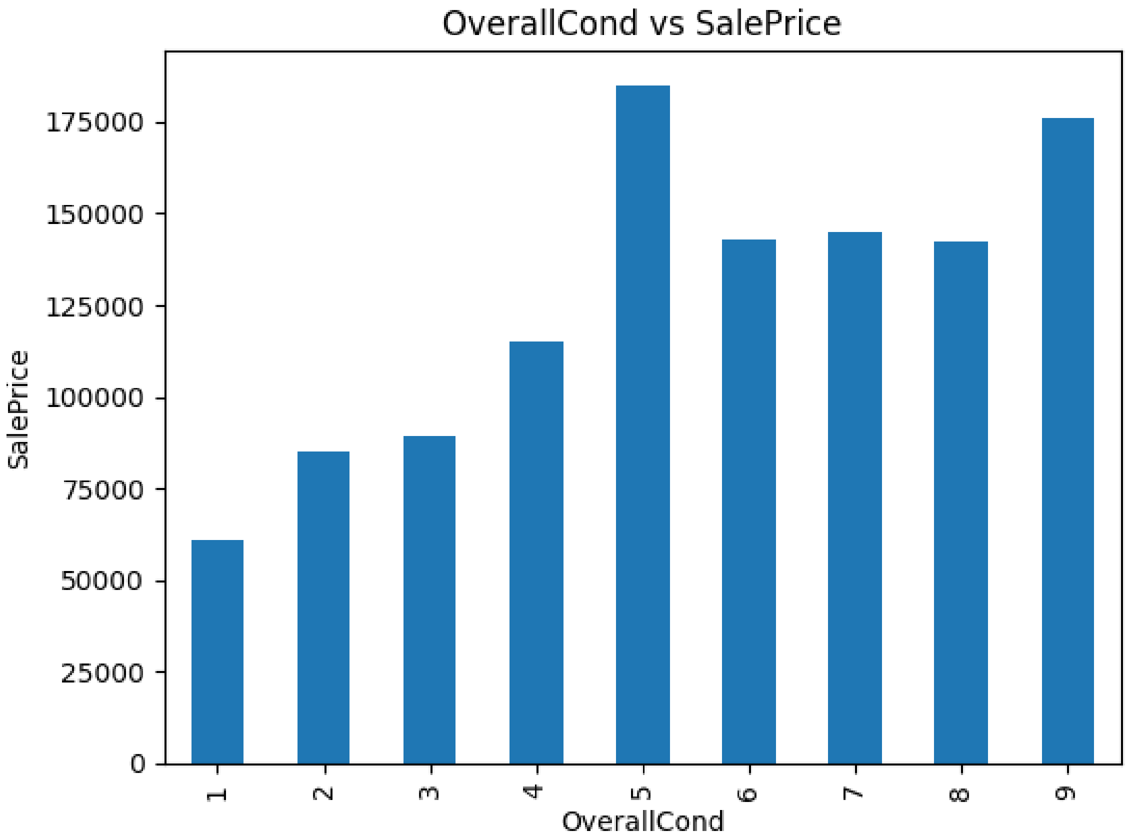
Out[431]:

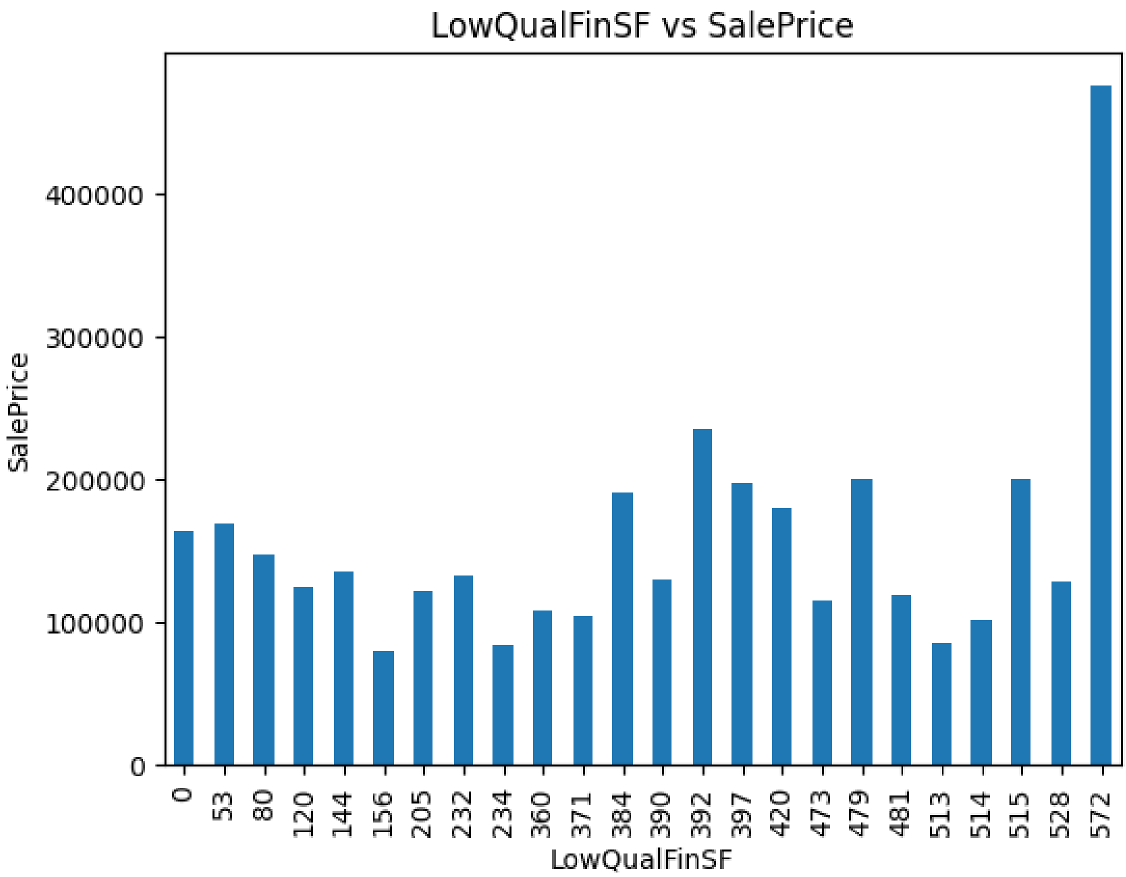
	MSSubClass	OverallQual	OverallCond	LowQualFinSF	BsmtFullBath	BsmtHalfBath	FullBatl
0	60	7	5	0	1	0	[illegible]
1	20	6	8	0	0	1	[illegible]
2	60	7	5	0	1	0	[illegible]
3	70	7	5	0	1	0	[illegible]
4	60	8	5	0	1	0	[illegible]

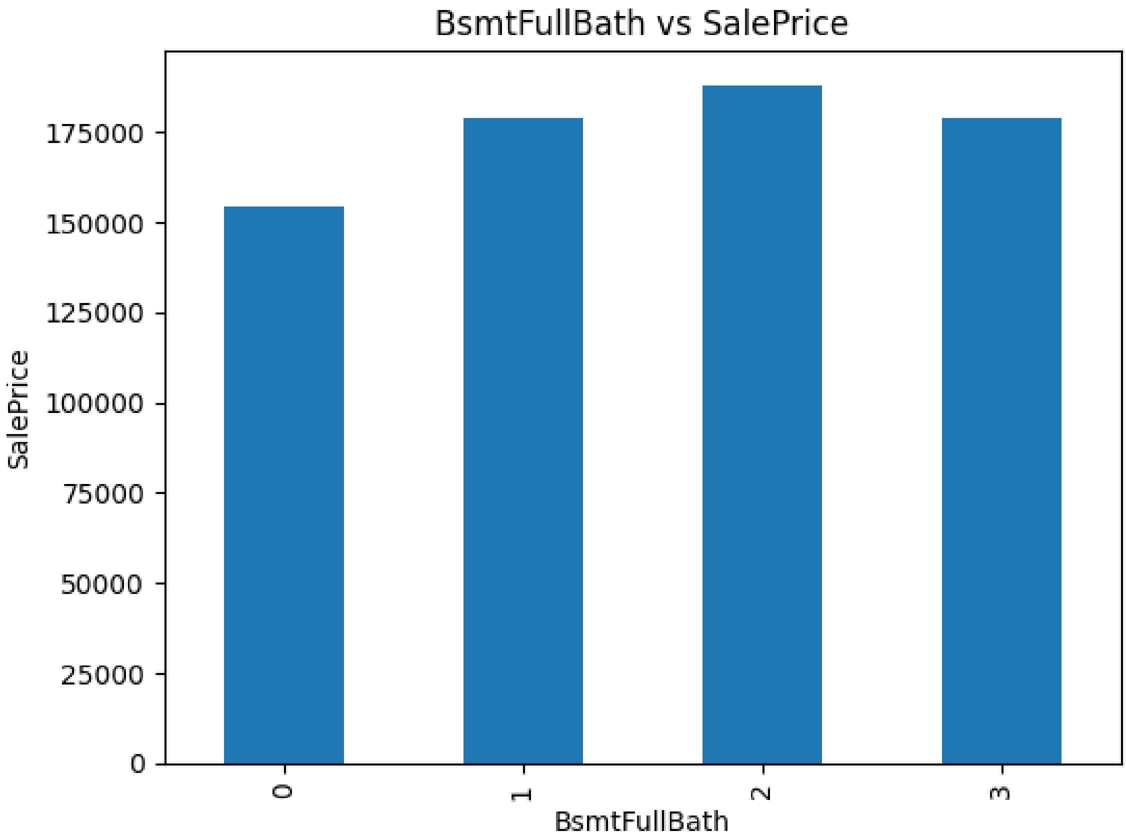
In [432…
```
for feature in discrete_feature:
  data = dataset.copy()
  data.groupby(feature)['SalePrice'].median().plot.bar()

  plt.xlabel(feature)
  plt.ylabel('SalePrice')
  plt.title(feature + ' vs SalePrice')
  plt.show()
```

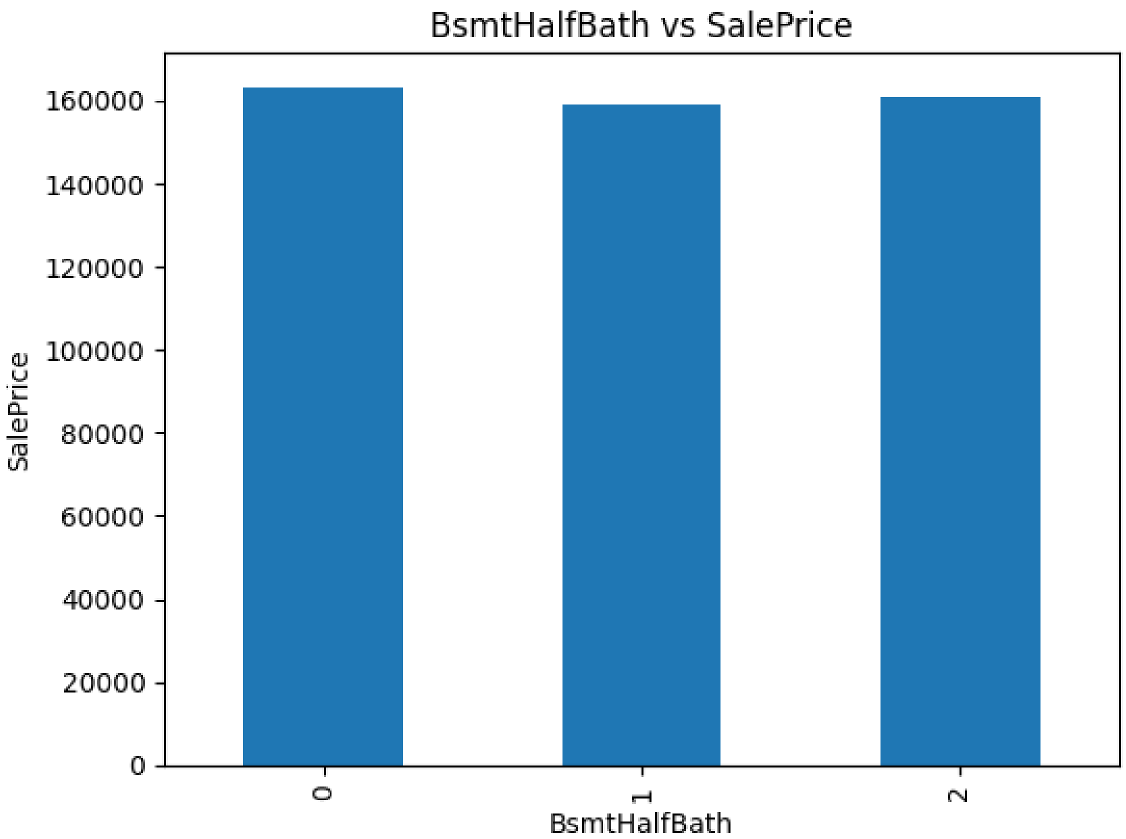

MSSubClass vs SalePrice
SalePrice
MSSubClass
0
50000
100000
150000
200000
20
30
40
45
50
60
70
75
80
85
90
120
160
180
190

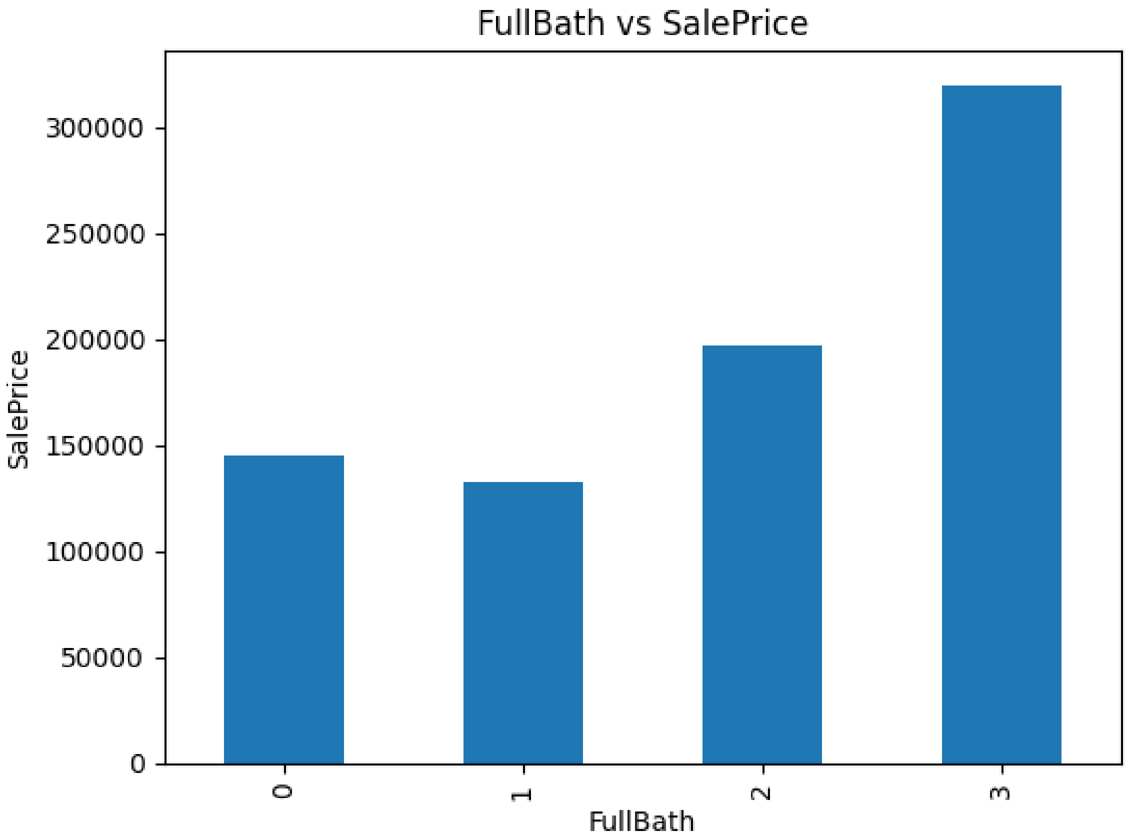

OverallQual vs SalePrice
SalePrice
OverallQual
0
100000
200000
300000
400000
1
2
3
4
5
6
7
8
9
10

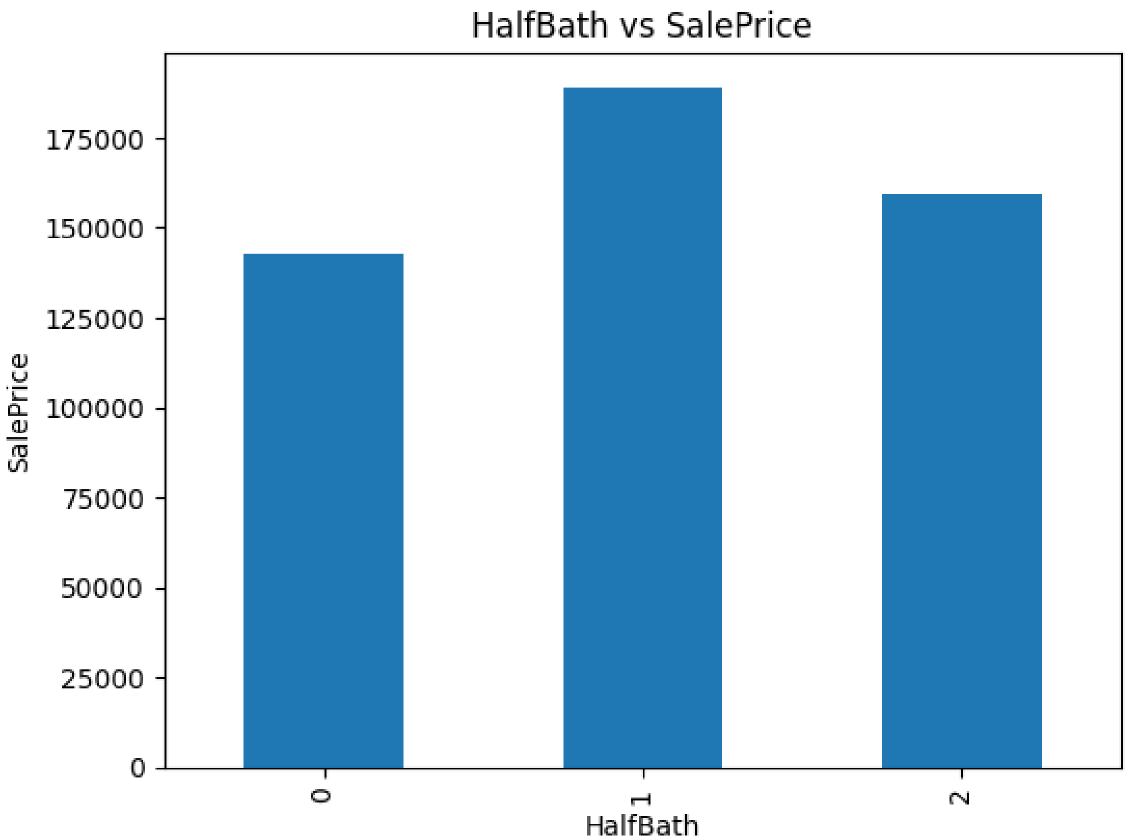

OverallCond vs SalePrice
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
1
2
3
4
5
6
7
8
9
OverallCond

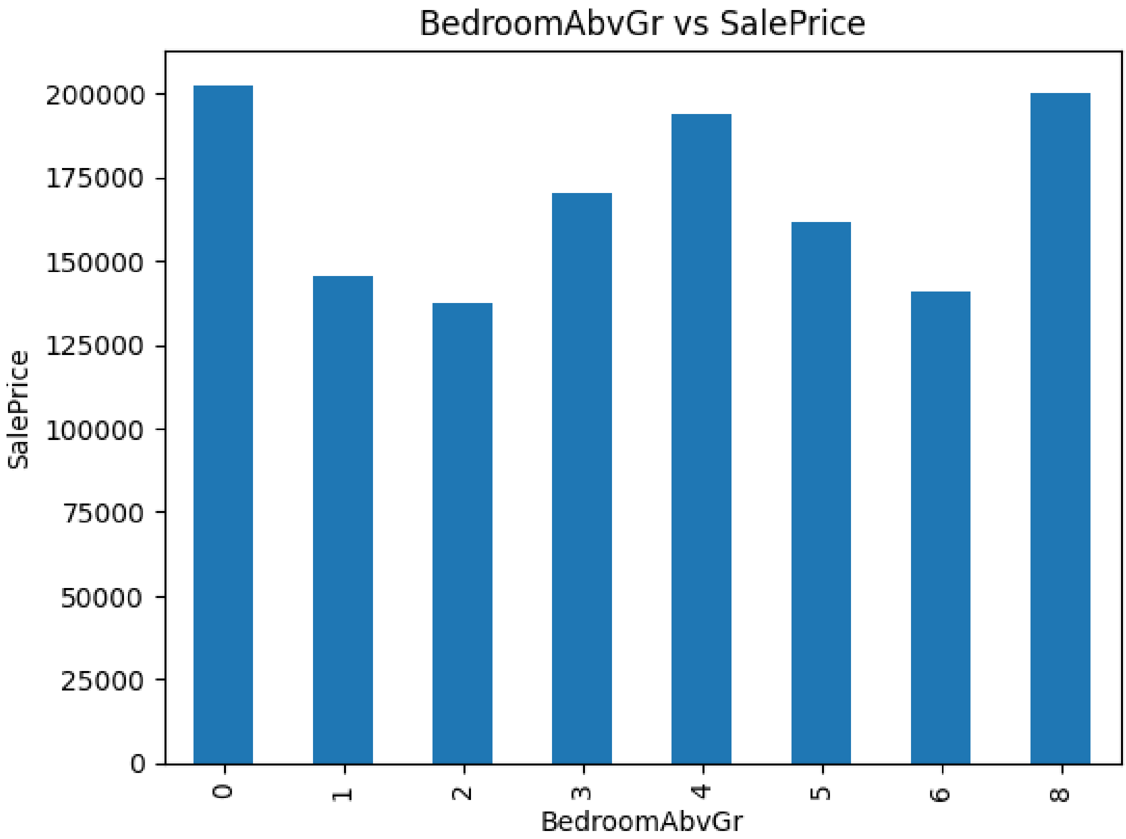

LowQualFinSF vs SalePrice
SalePrice
400000
300000
200000
100000
0
0
53
80
120
144
156
205
232
234
360
371
384
390
392
397
420
473
479
481
513
514
515
528
572
LowQualFinSF


BsmtFullBath vs SalePrice
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
BsmtFullBath

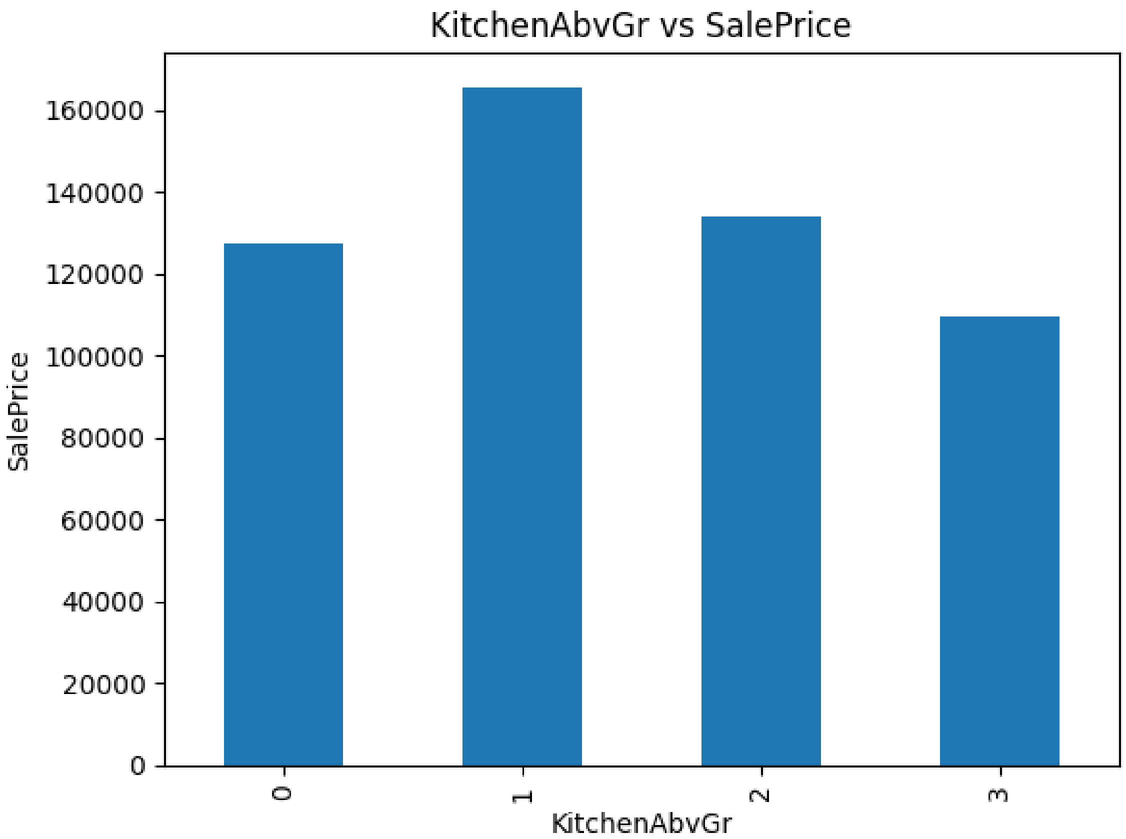

BsmtHalfBath vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
BsmtHalfBath


FullBath vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
0
1
2
3
FullBath

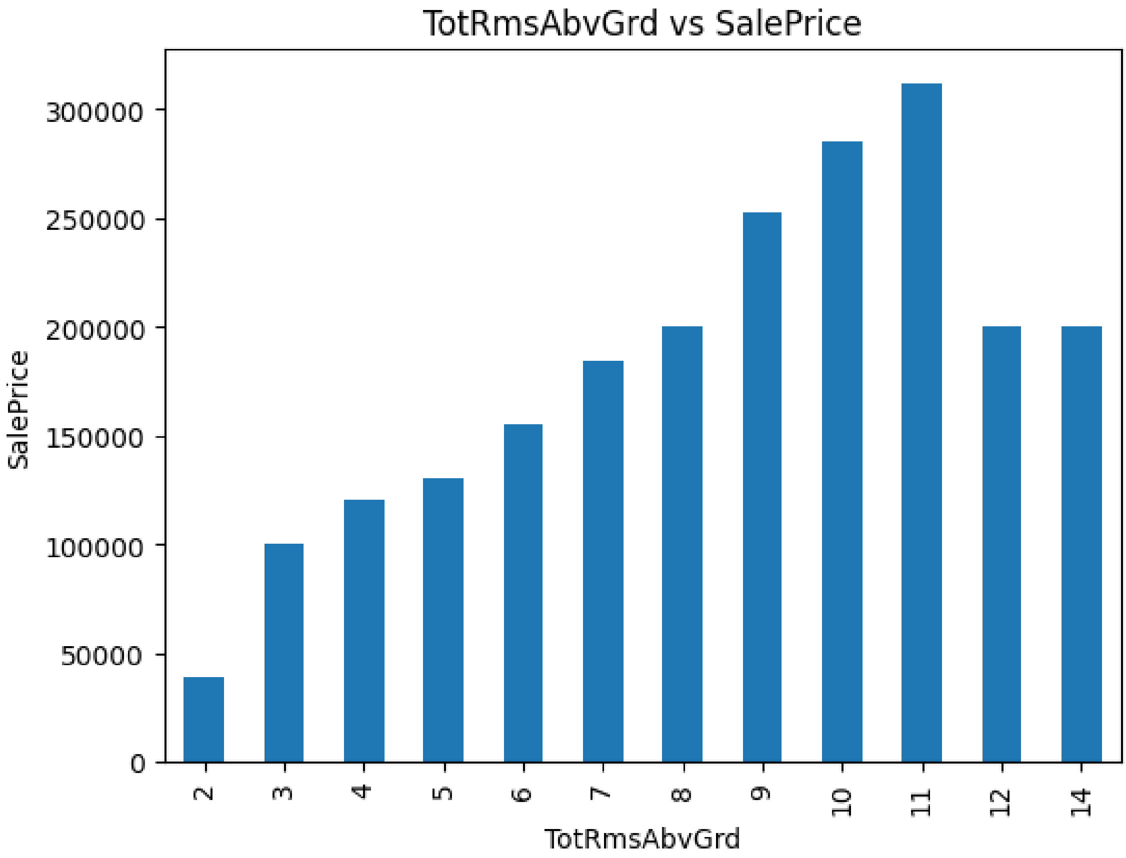

HalfBath vs SalePrice
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
HalfBath

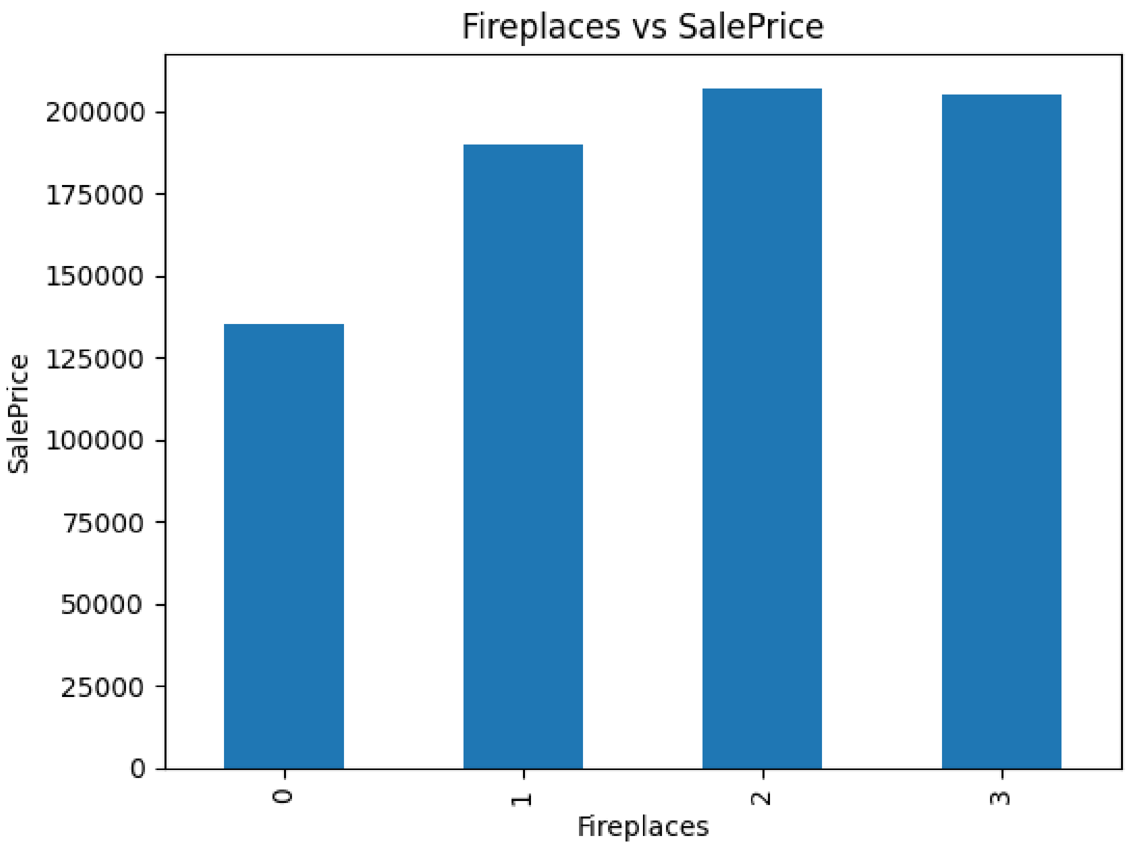


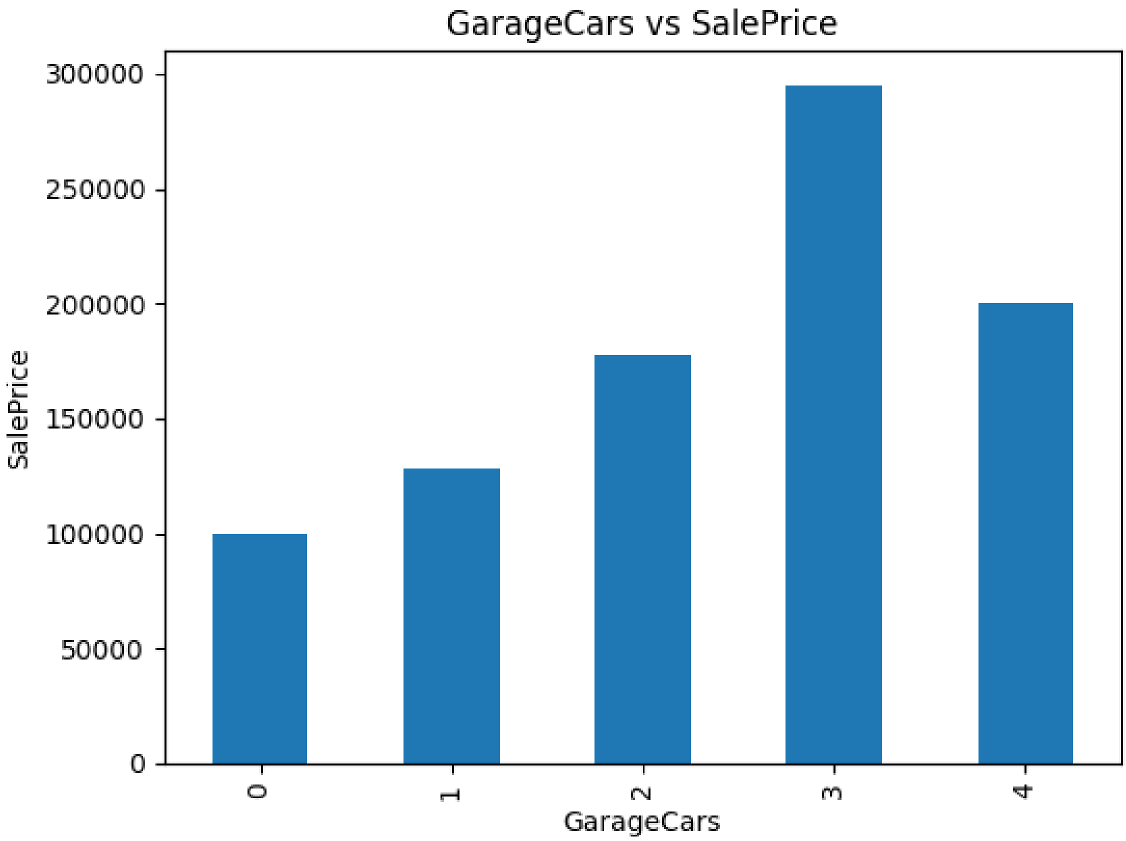
BedroomAbvGr vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
4
5
6
8
BedroomAbvGr

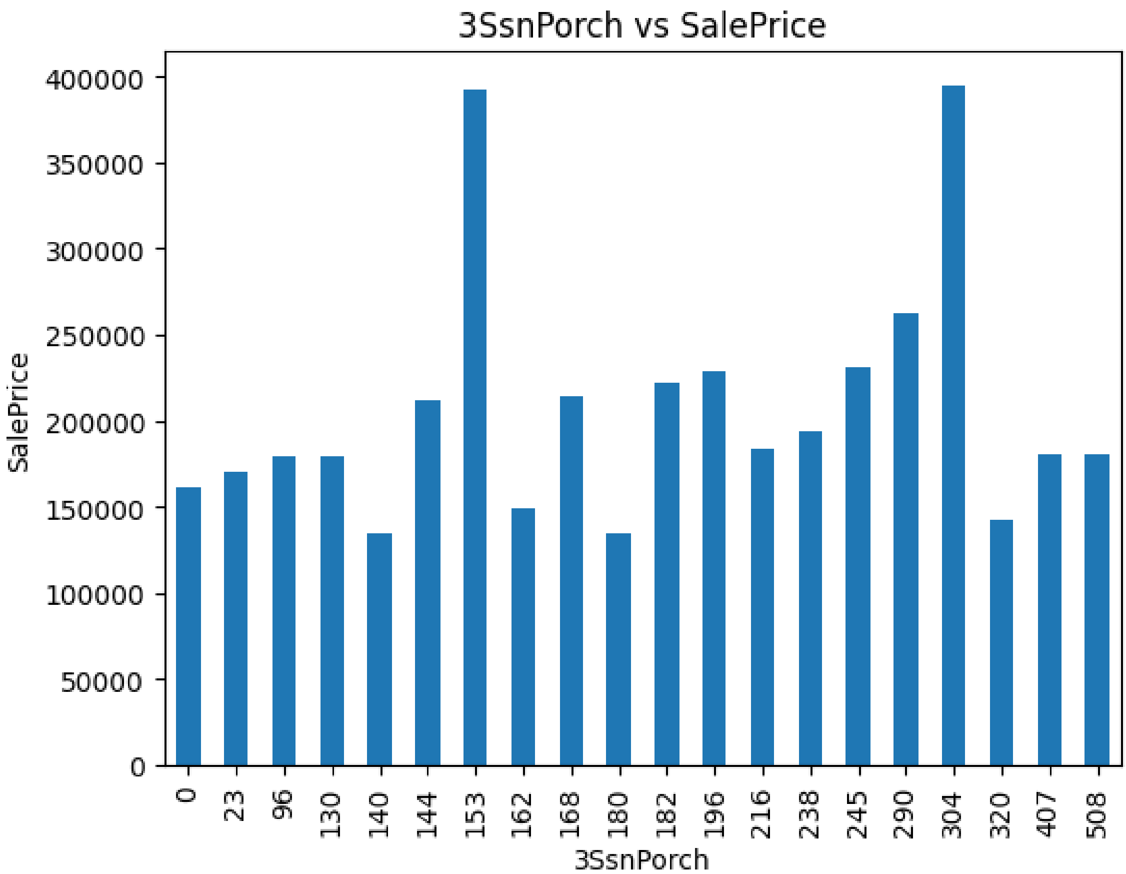


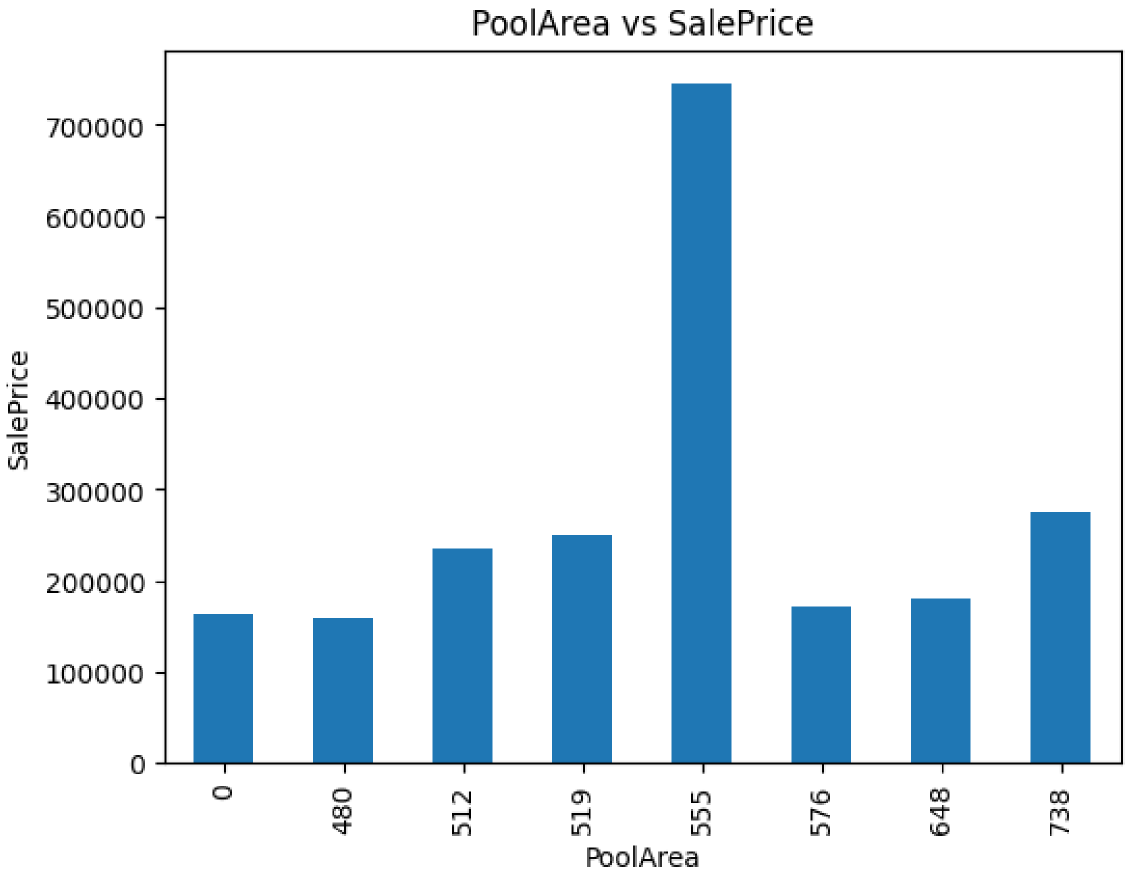
KitchenAbvGr vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
3
KitchenAbvGr

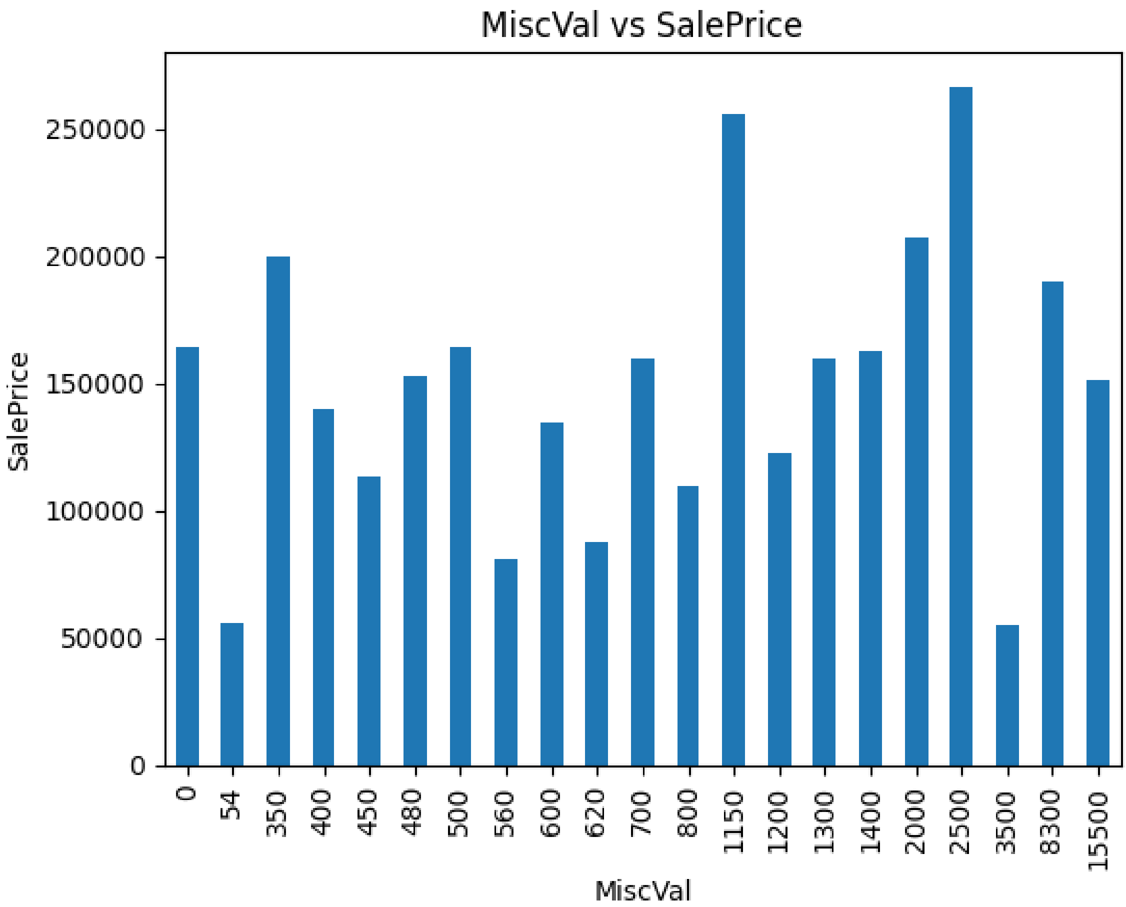

TotRmsAbvGrd vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd

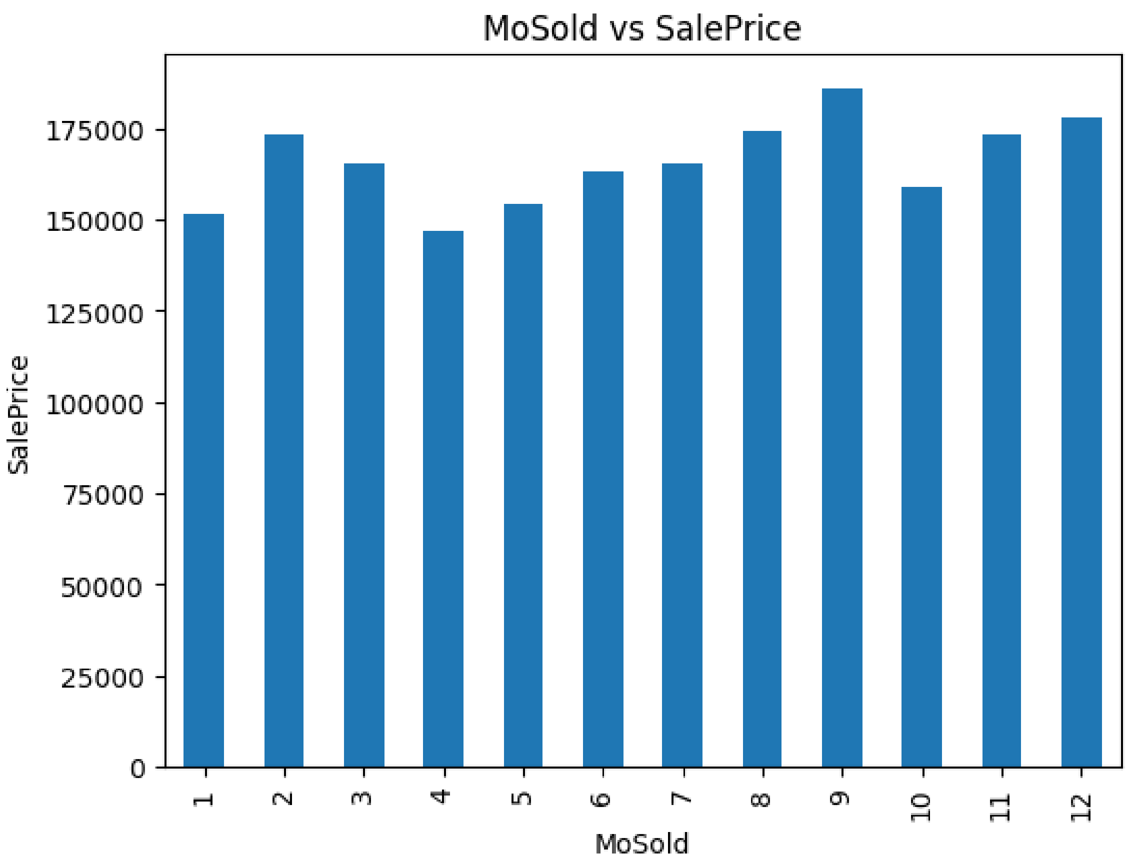

Fireplaces vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
Fireplaces


GarageCars vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
0
1
2
3
4
GarageCars


3SsnPorch vs SalePrice
SalePrice
400000
350000
300000
250000
200000
150000
100000
50000
0
0
23
96
130
140
144
153
162
168
180
182
196
216
238
245
290
304
320
407
508
3SsnPorch


PoolArea vs SalePrice
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
480
512
519
555
576
648
738
PoolArea


MiscVal vs SalePrice
SalePrice
250000
200000
150000
100000
50000
0
0
54
350
400
450
480
500
560
600
620
700
800
1150
1200
1300
1400
2000
2500
3500
8300
15500
MiscVal



CONTINUOUS VARIABLES

In [433…

```
# find all continuous features
continuous_feature = []
for feature in numerical_features:
  if feature not in discrete_feature + year_feature + ['Id']:
    continuous_feature.append(feature)

len(continuous_feature)
```

Out[433]: 16

In [434…

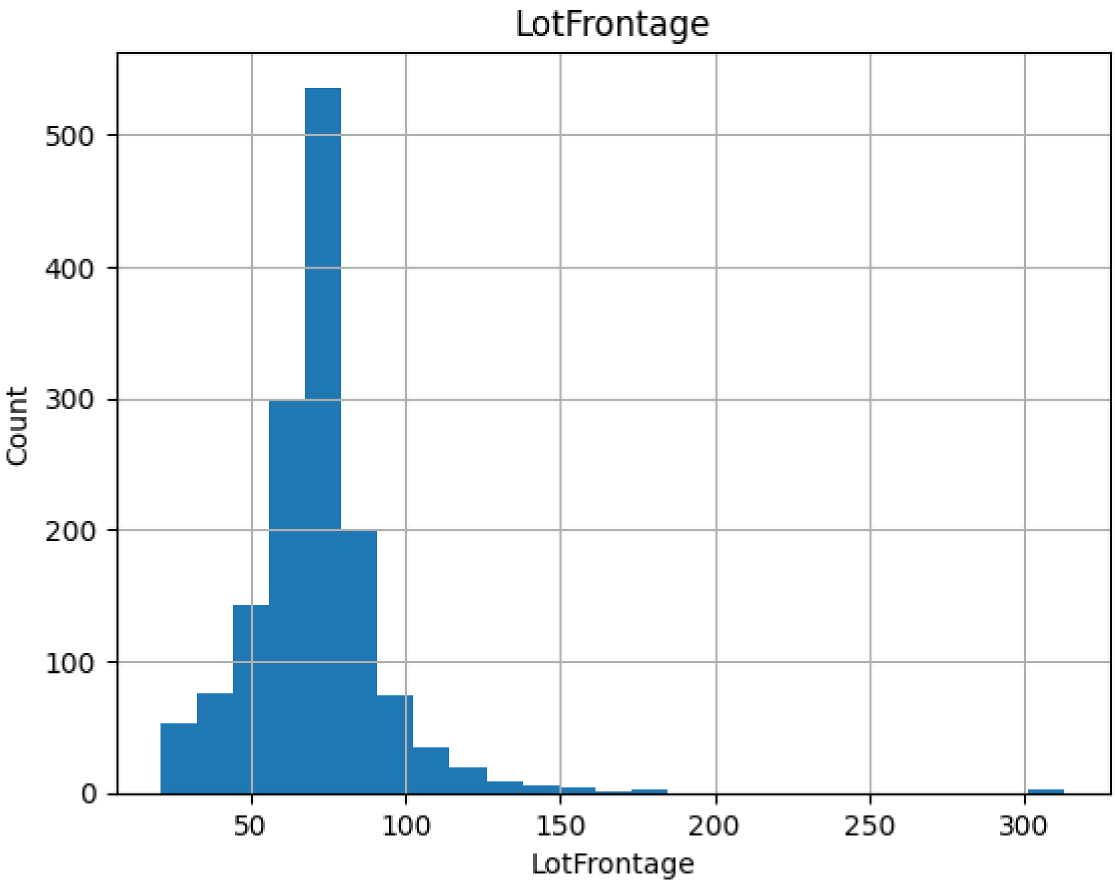
```
continuous_feature
```

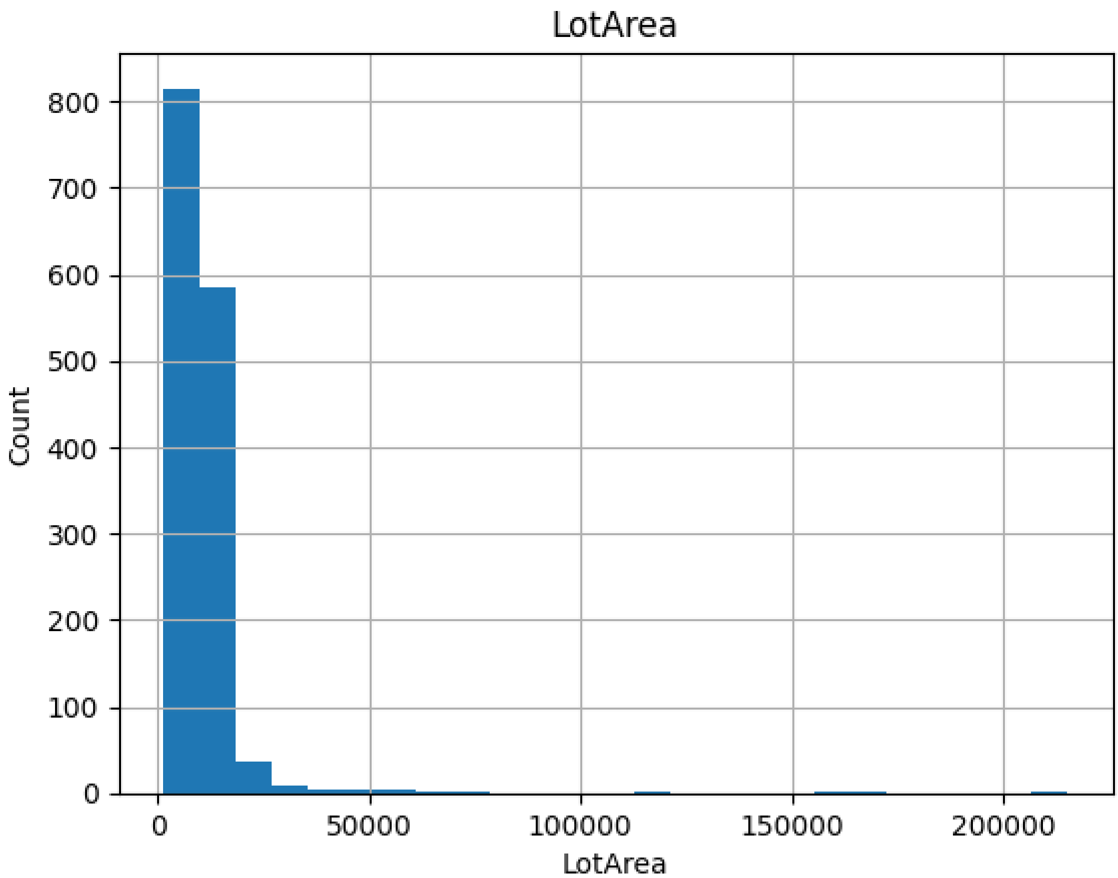
Out[434]:

```
['LotFrontage',
 'LotArea',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'GrLivArea',
 'GarageArea',
 'WoodDeckSF',
 'OpenPorchSF',
 'EnclosedPorch',
 'ScreenPorch',
 'SalePrice']
```

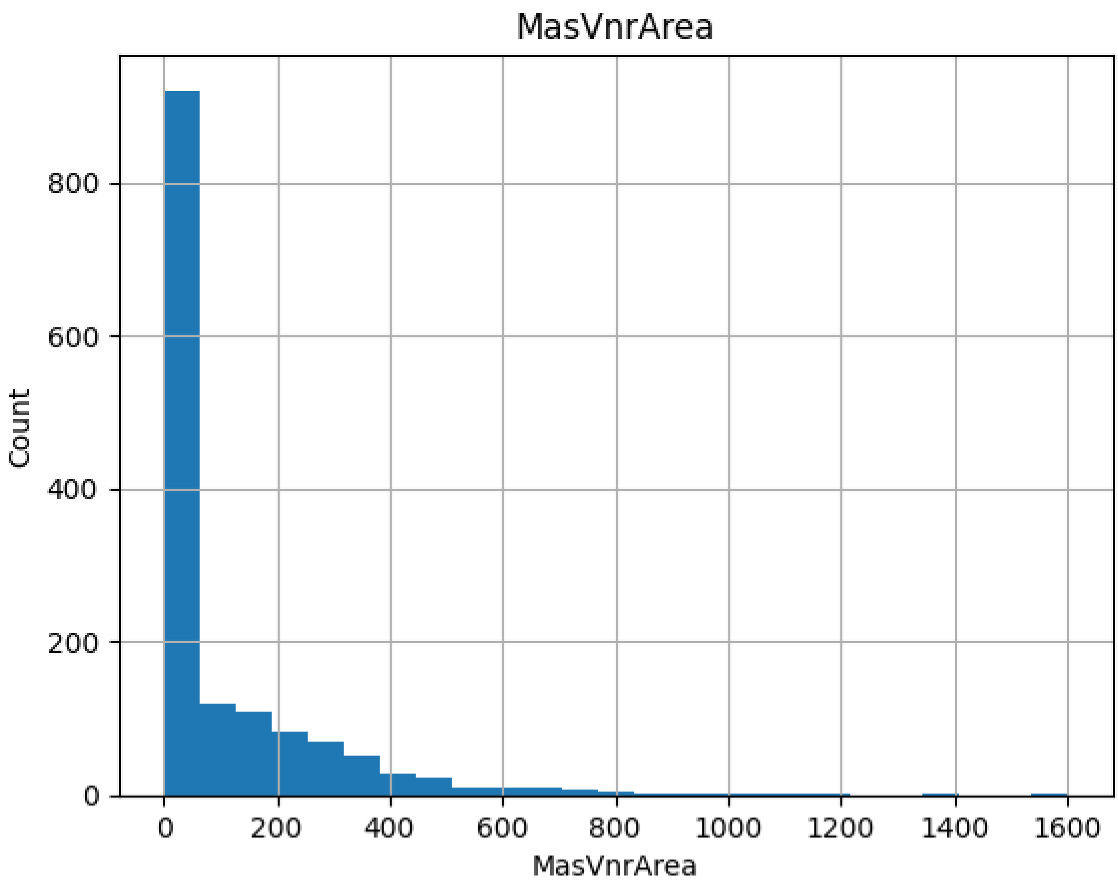
In [435…

```
# analyse the continuous values by creating histograms to understand the distrib

for feature in continuous_feature:
    data=dataset.copy()
    data[feature].hist(bins = 25)

    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title(feature)
    plt.show()
```

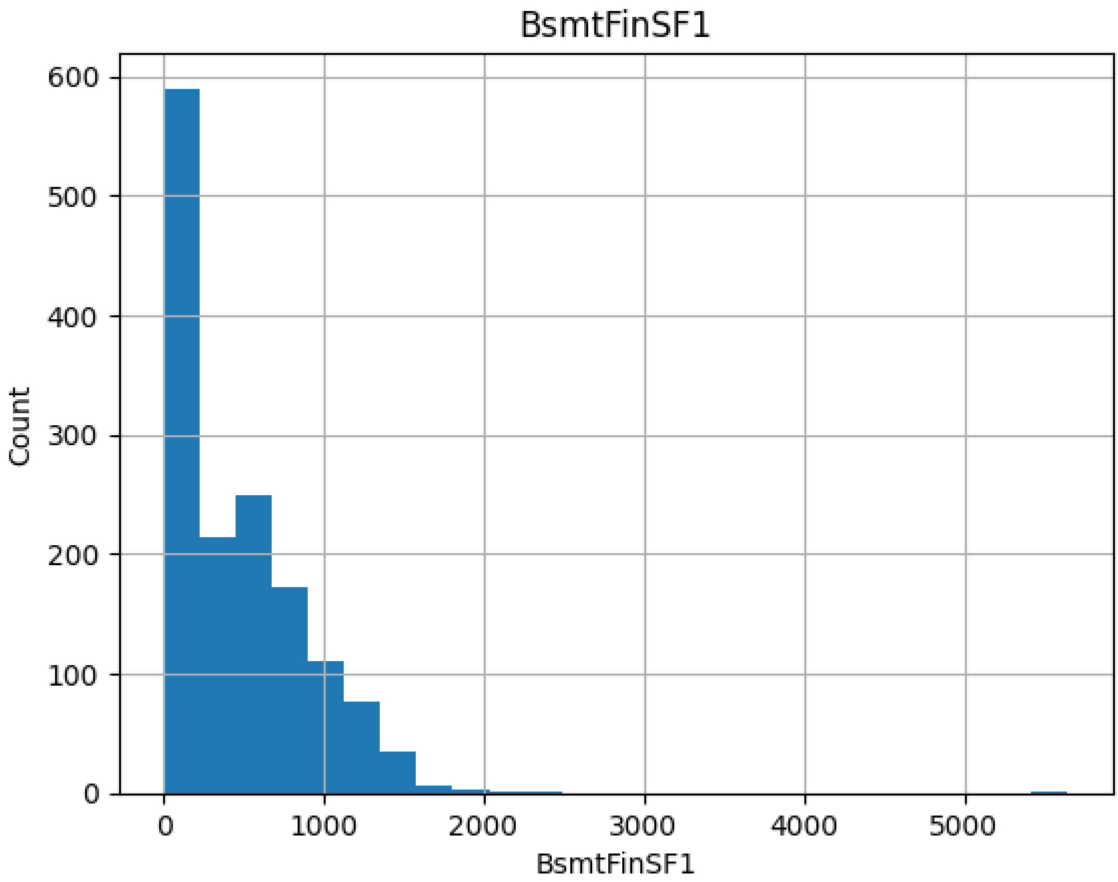


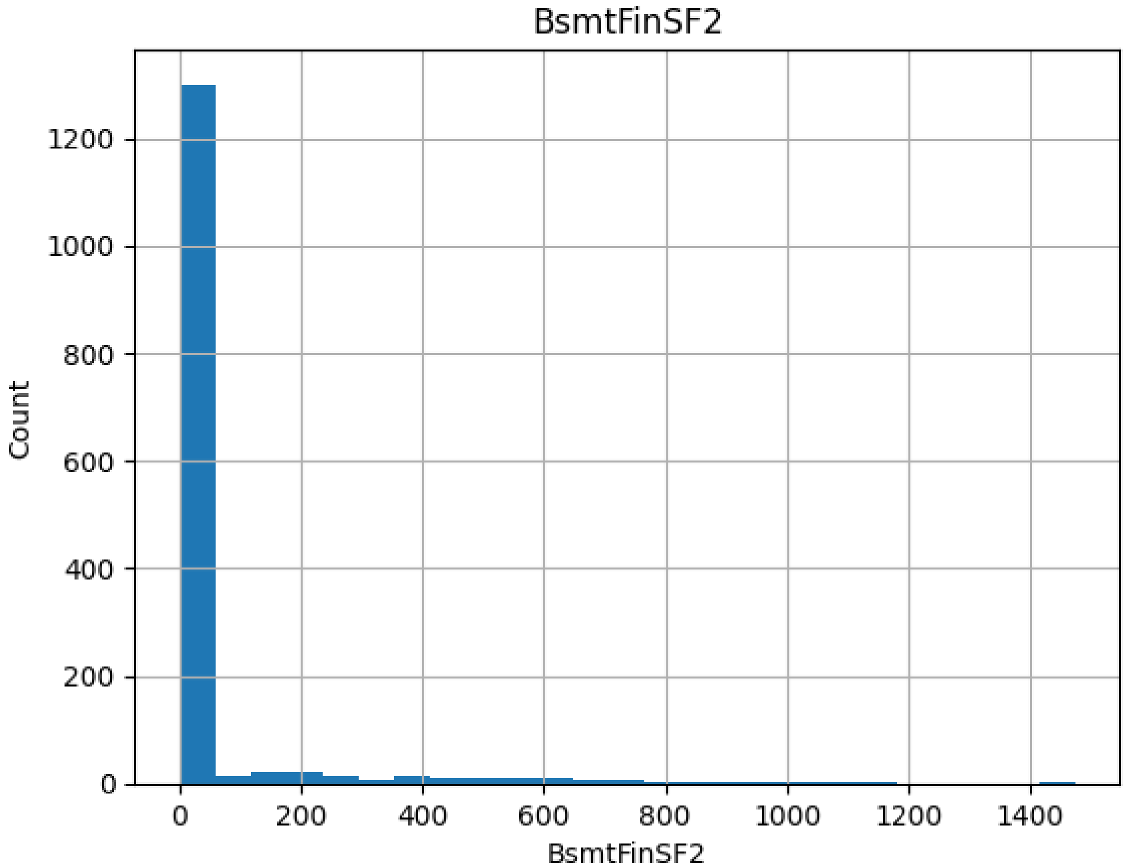


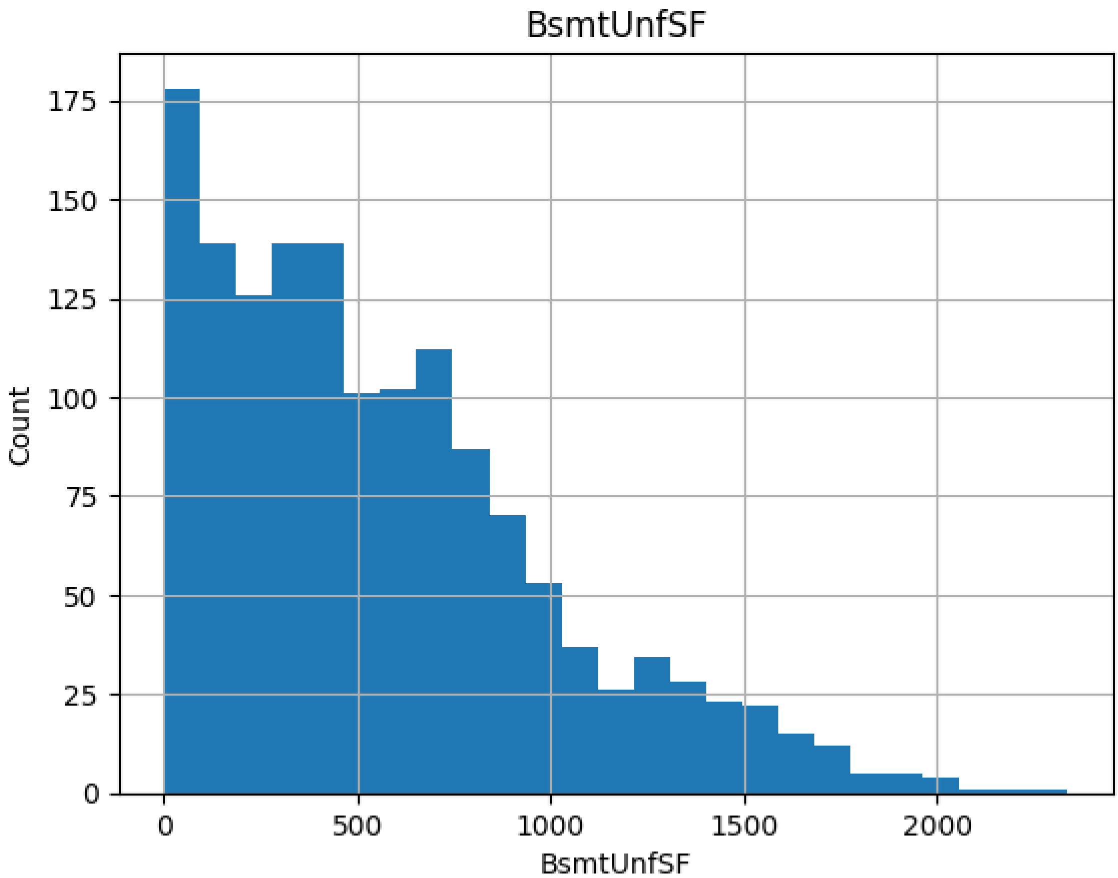
LotArea
Count
800
700
600
500
400
300
200
100
0
0
50000
100000
150000
200000
LotArea

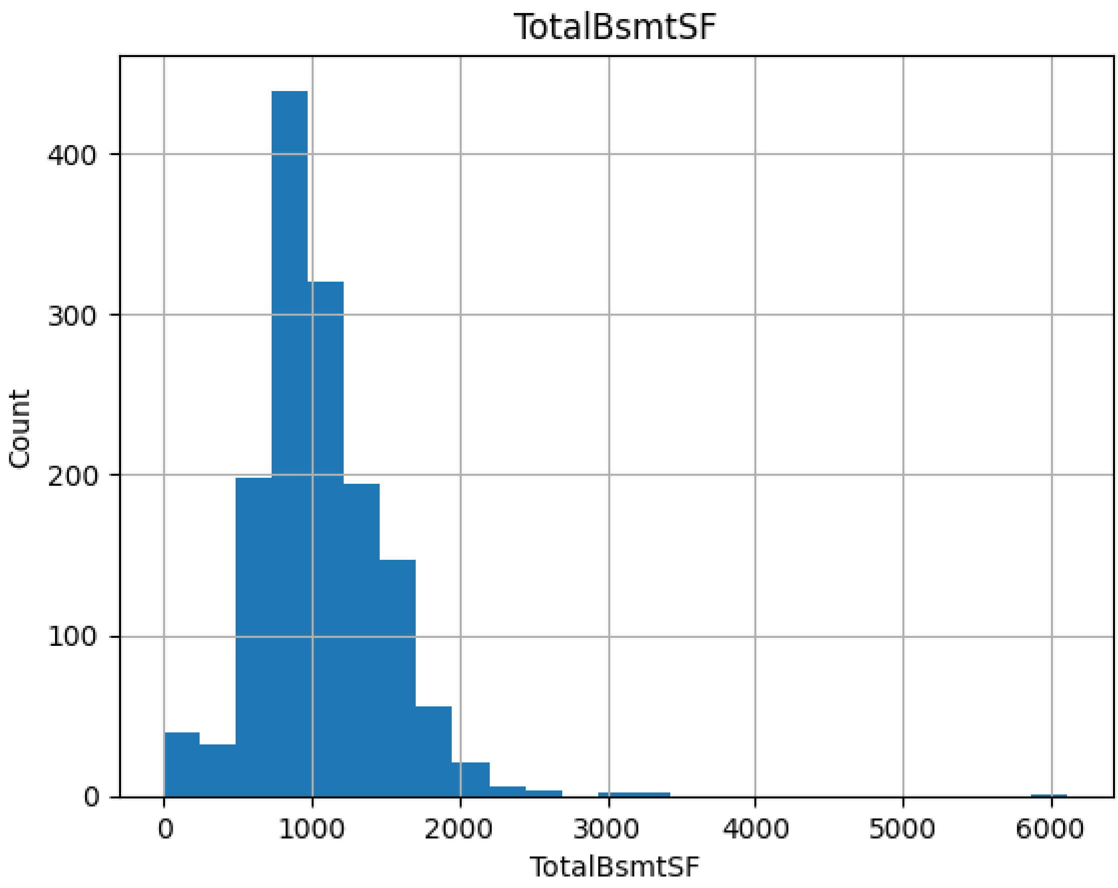


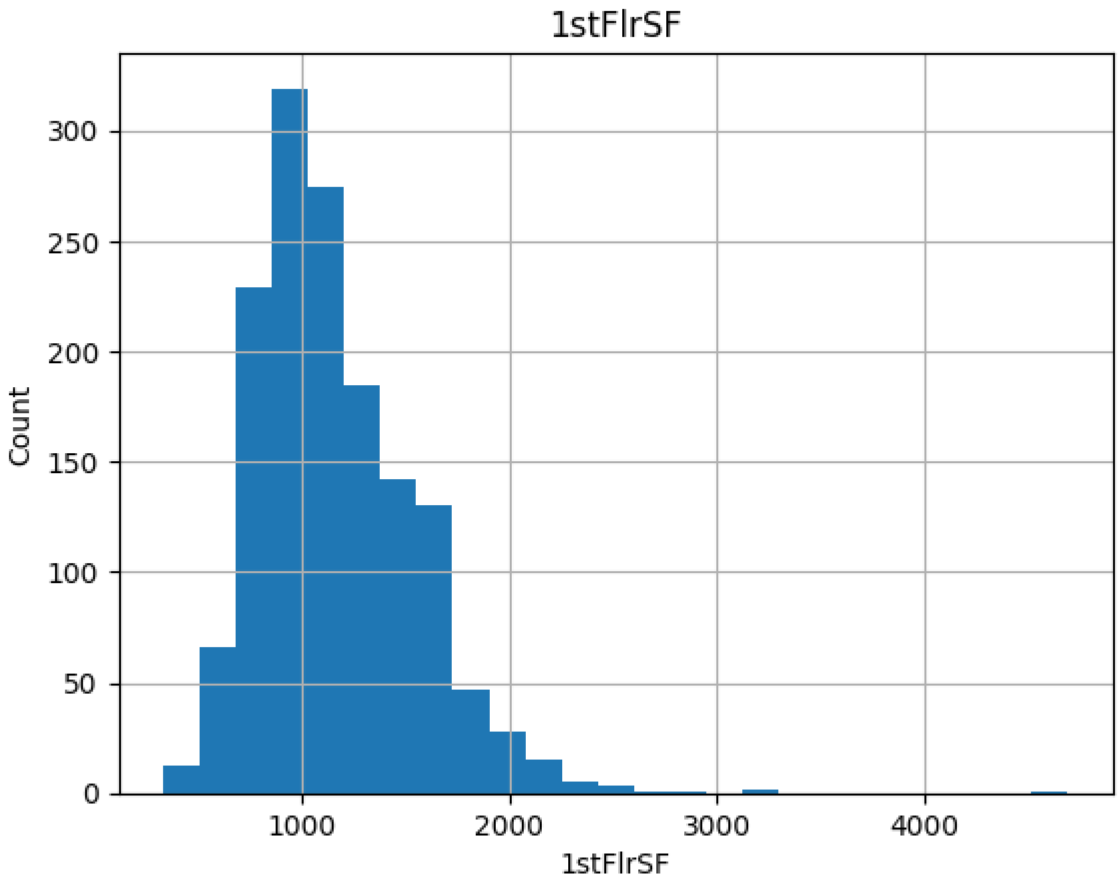
MasVnrArea
Count
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

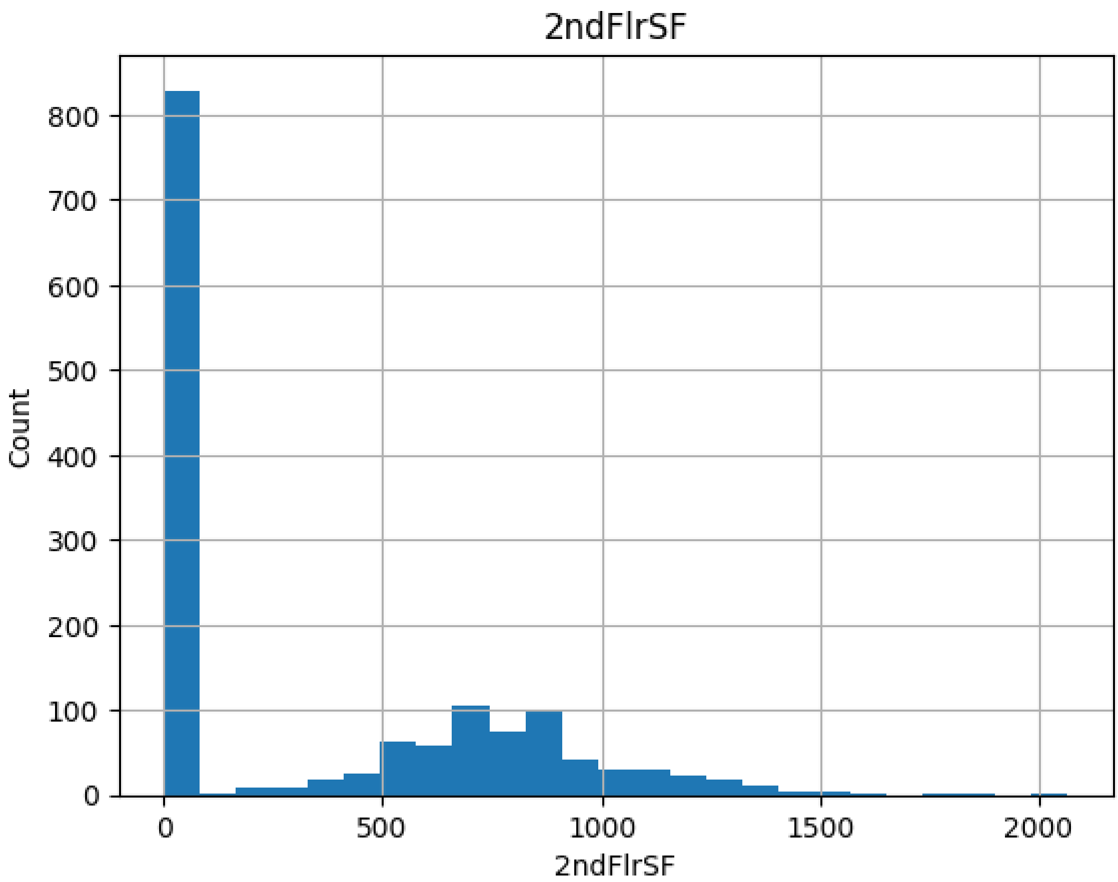

BsmtFinSF1
600
500
400
300
200
100
0
Count
0
1000
2000
3000
4000
5000
BsmtFinSF1

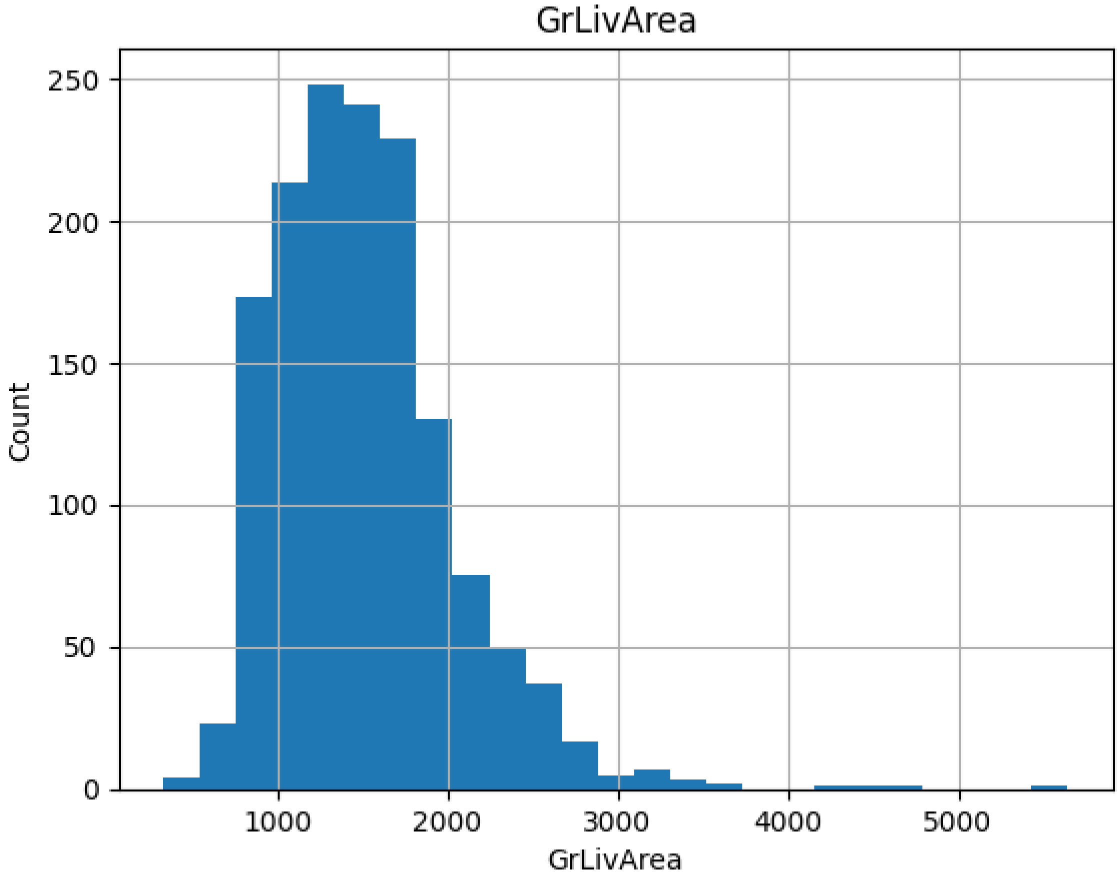

BsmtFinSF2
1200
1000
800
600
400
200
0
Count
0
200
400
600
800
1000
1200
1400
BsmtFinSF2


BsmtUnfSF
Count
175
150
125
100
75
50
25
0
0
500
1000
1500
2000
BsmtUnfSF

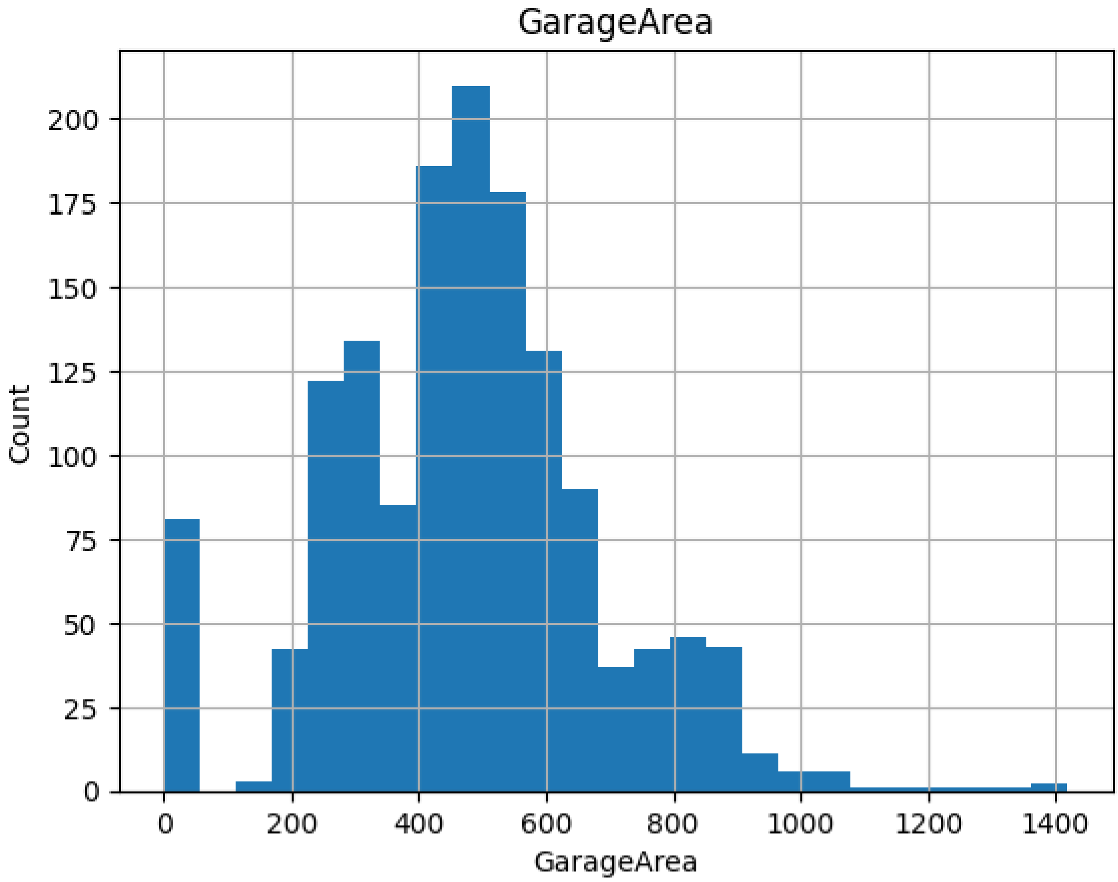

TotalBsmtSF
Count
400
300
200
100
0
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF


1stFlrSF
Count
0
50
100
150
200
250
300
1000
2000
3000
4000
1stFlrSF

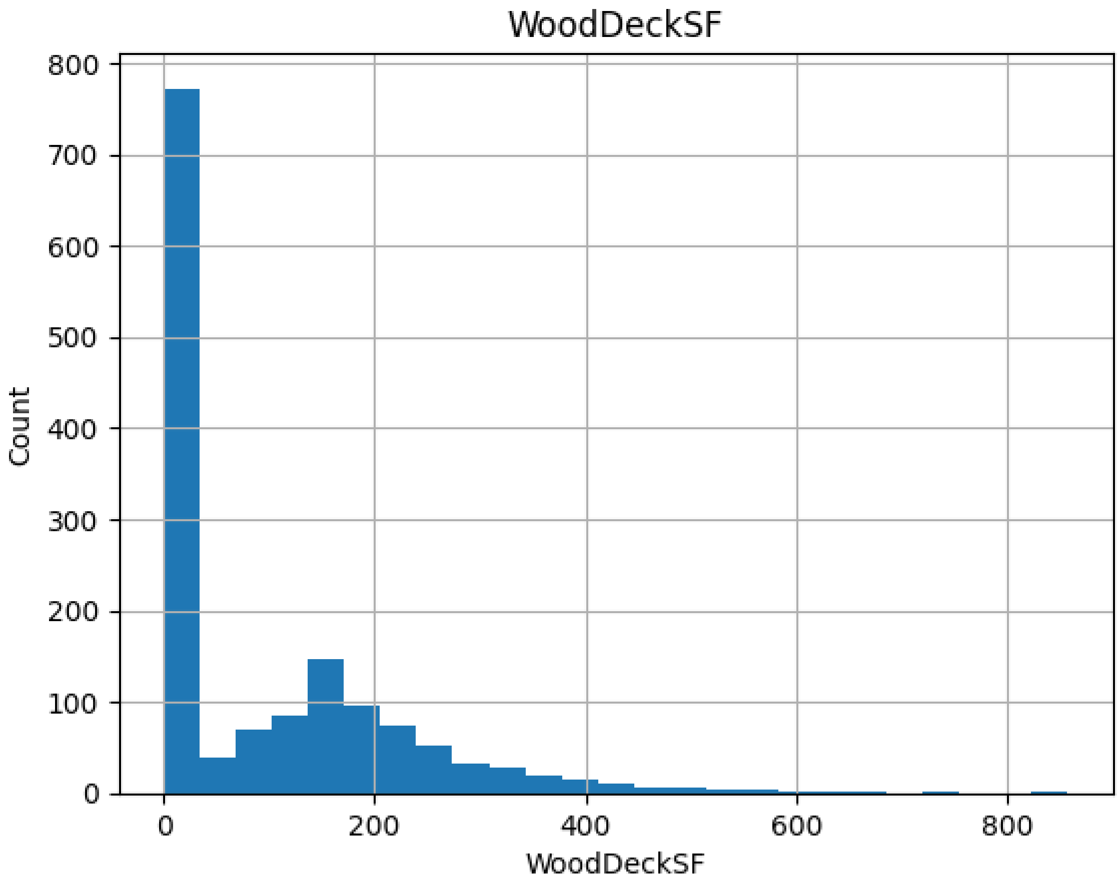

2ndFlrSF
Count
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000
2ndFlrSF

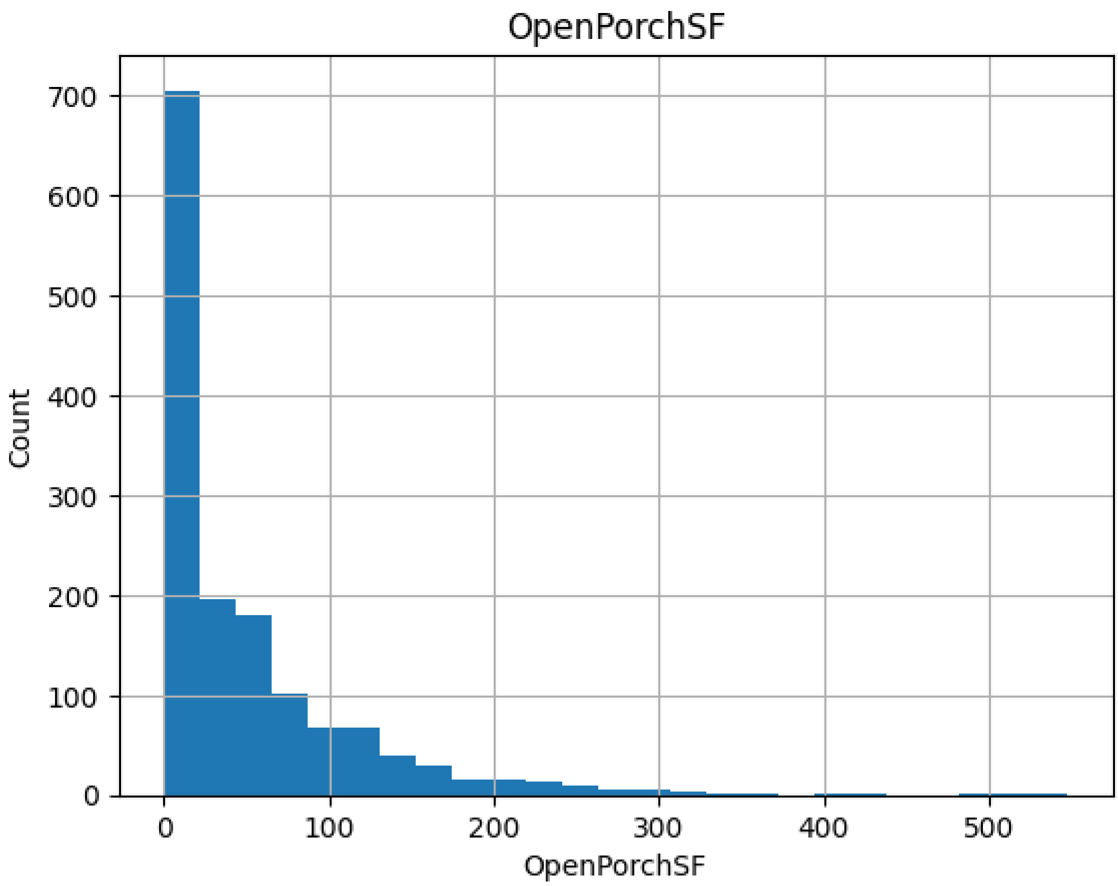


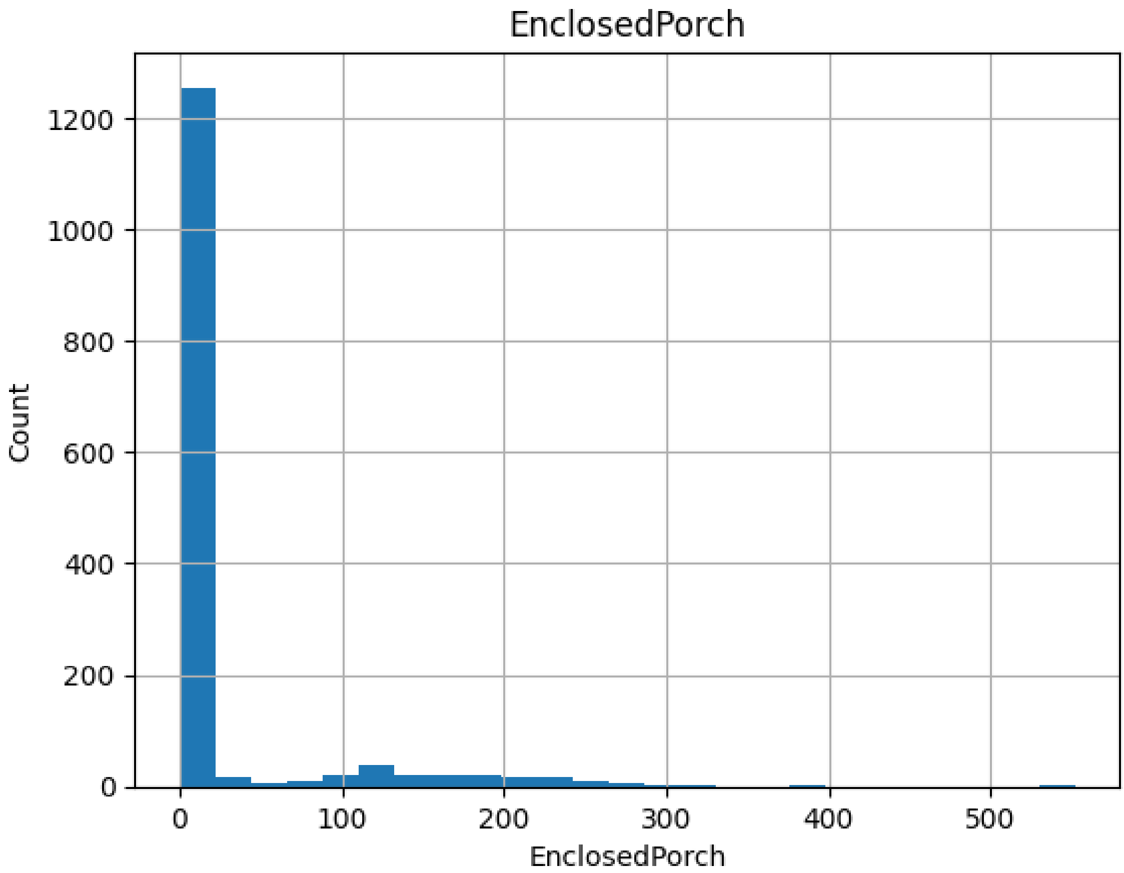
GrLivArea
Count
0
50
100
150
200
250
1000
2000
3000
4000
5000
GrLivArea

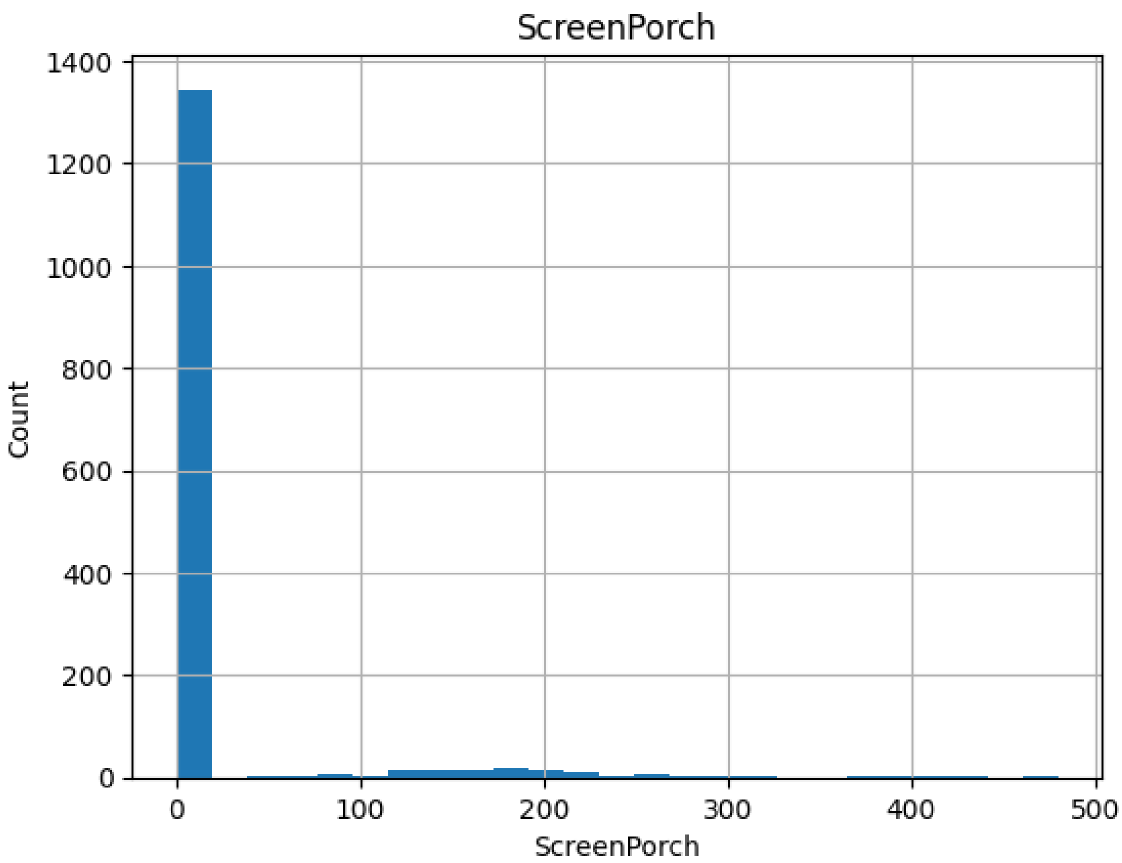


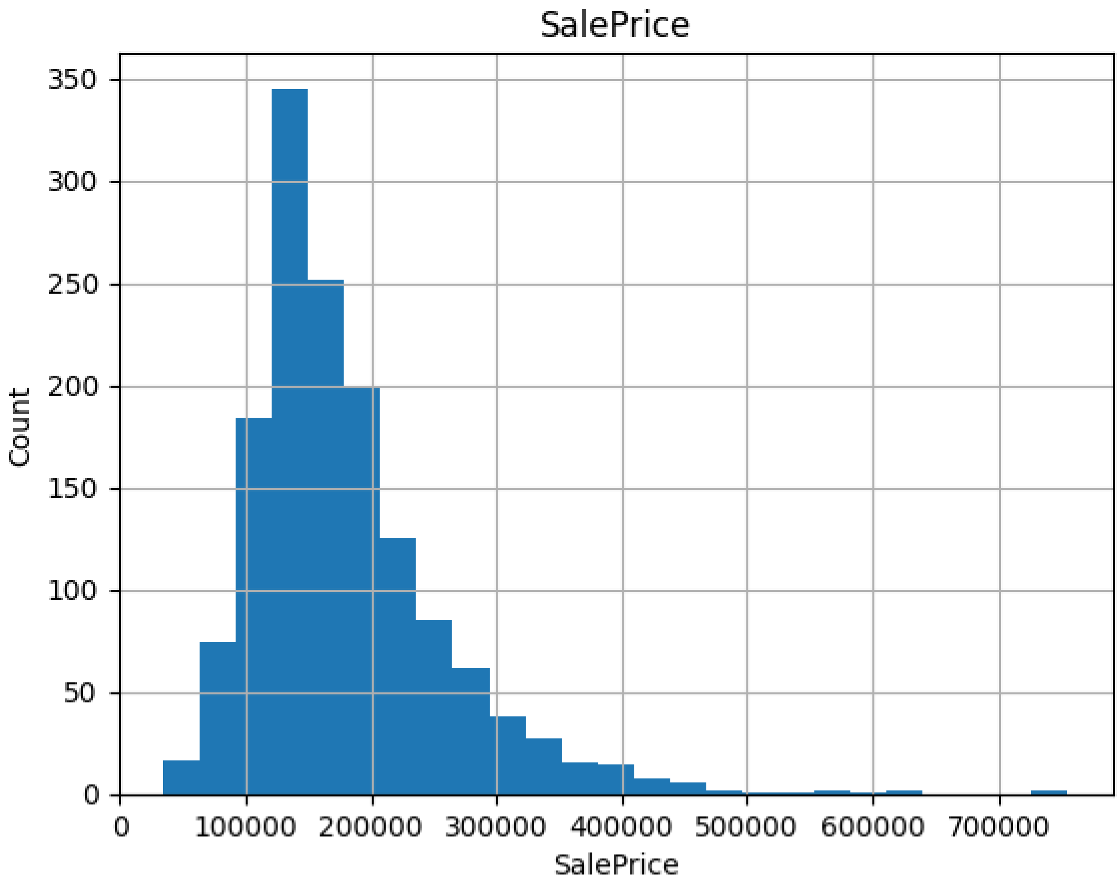
GarageArea
Count
0
25
50
75
100
125
150
175
200
0
200
400
600
800
1000
1200
1400
GarageArea


WoodDeckSF
Count
800
700
600
500
400
300
200
100
0
0
200
400
600
800
WoodDeckSF


OpenPorchSF
Count
700
600
500
400
300
200
100
0
0
100
200
300
400
500
OpenPorchSF

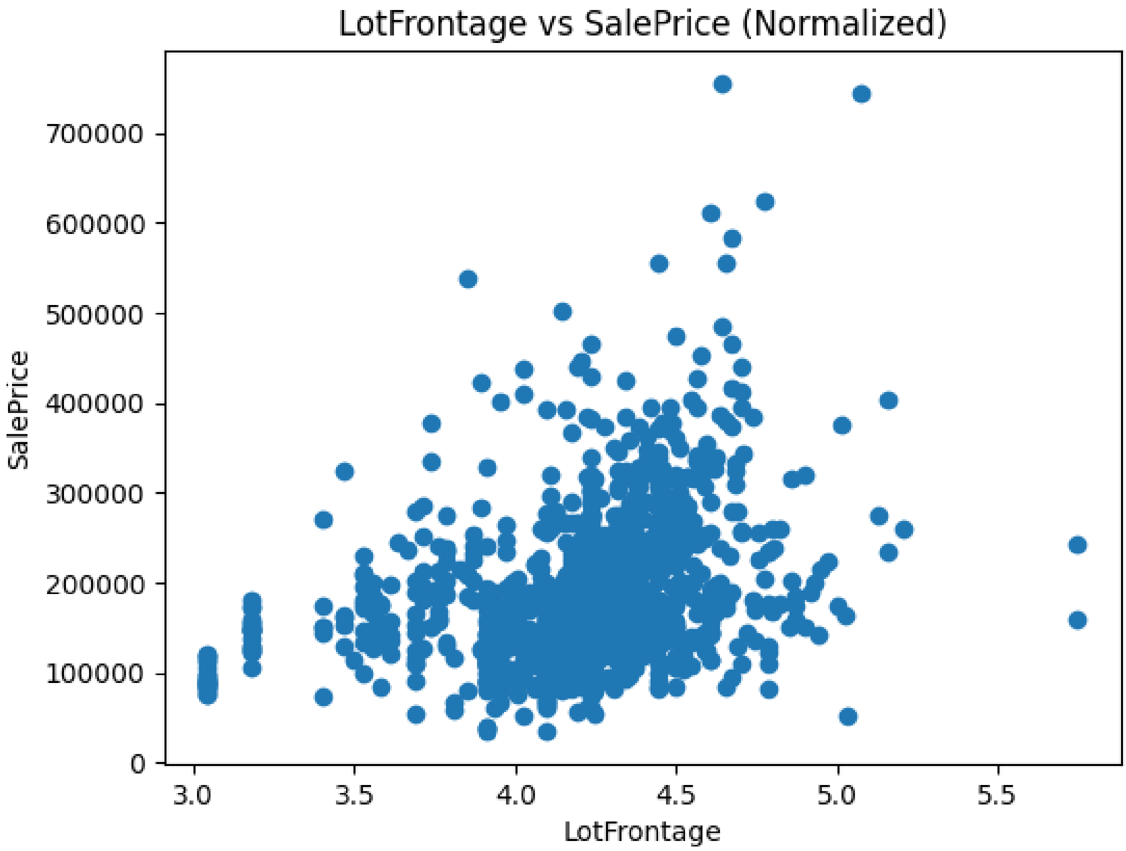

EnclosedPorch
Count
1200
1000
800
600
400
200
0
0
100
200
300
400
500
EnclosedPorch

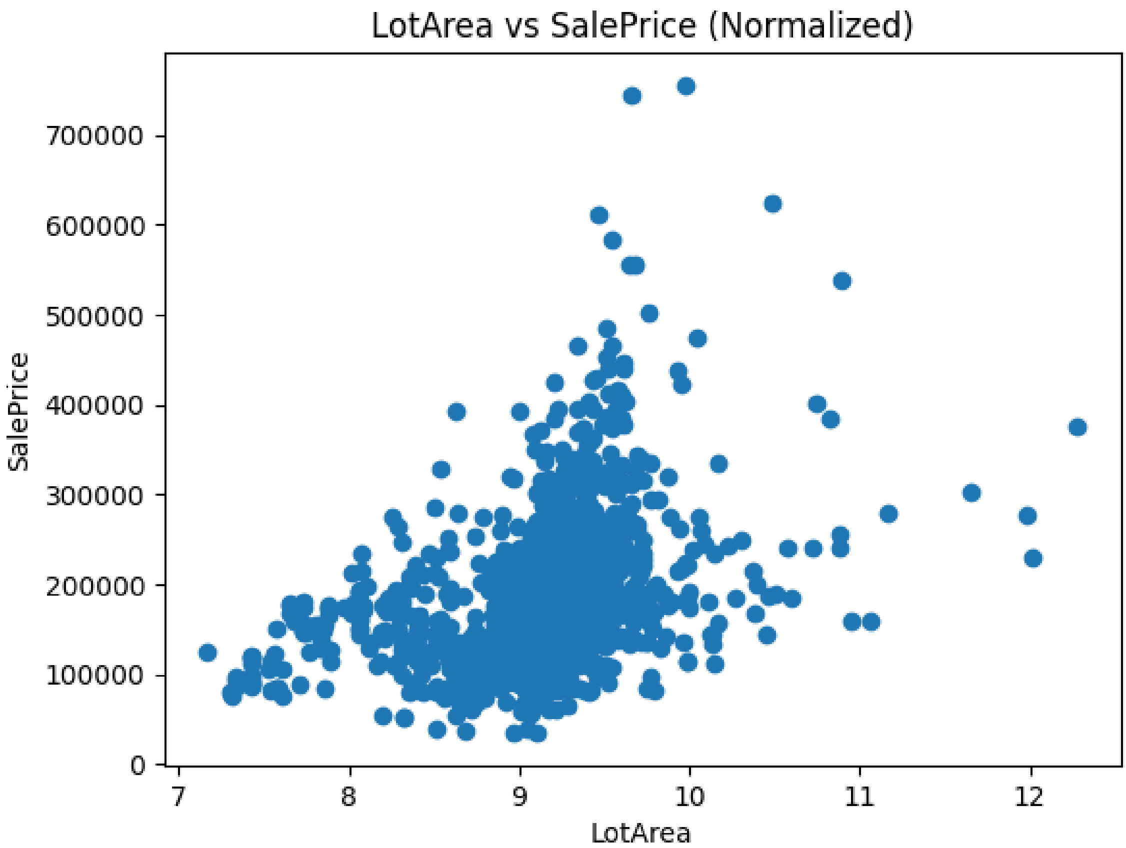

ScreenPorch
Count
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
ScreenPorch

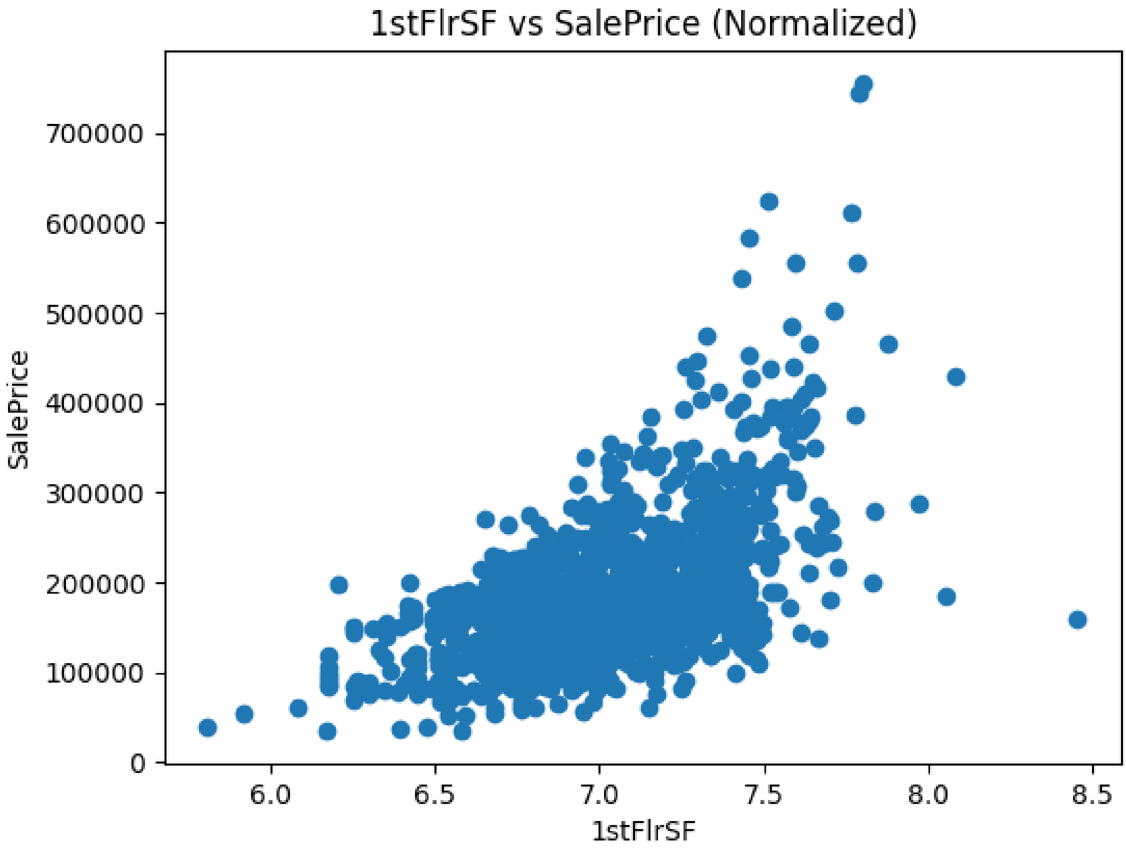


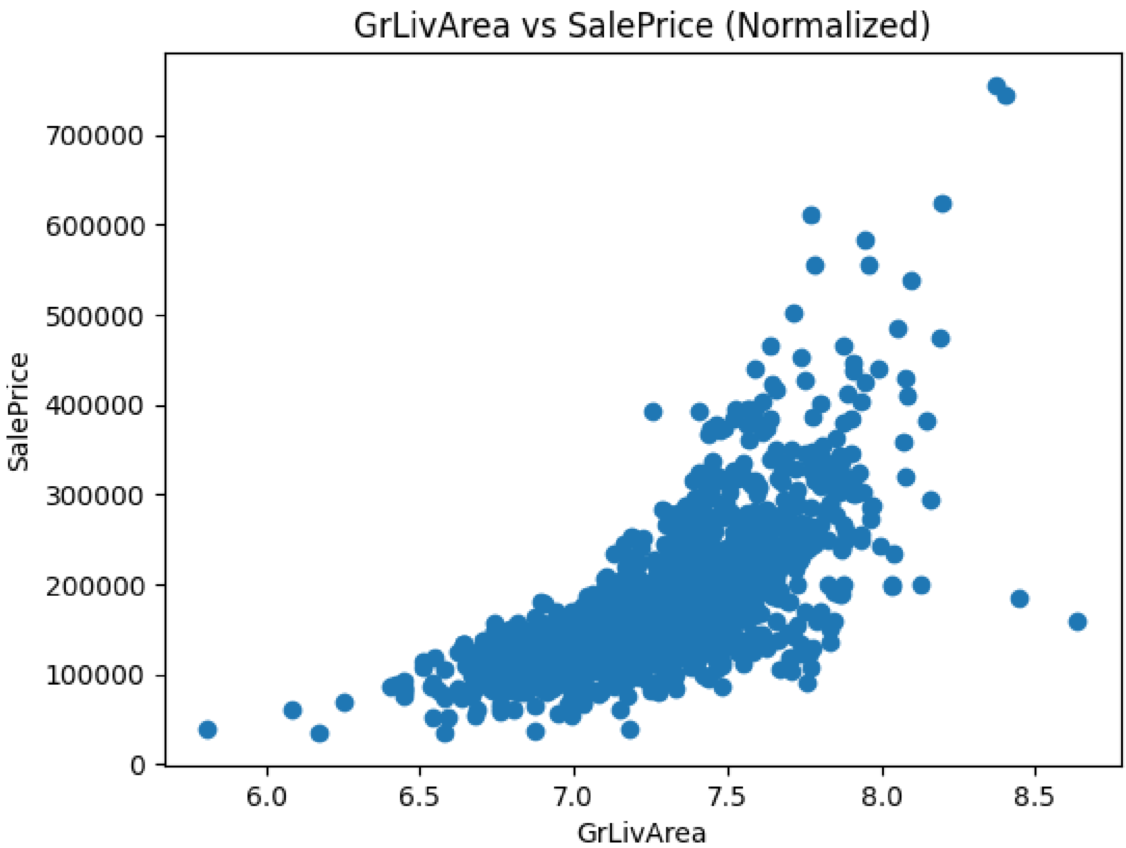
In [436…

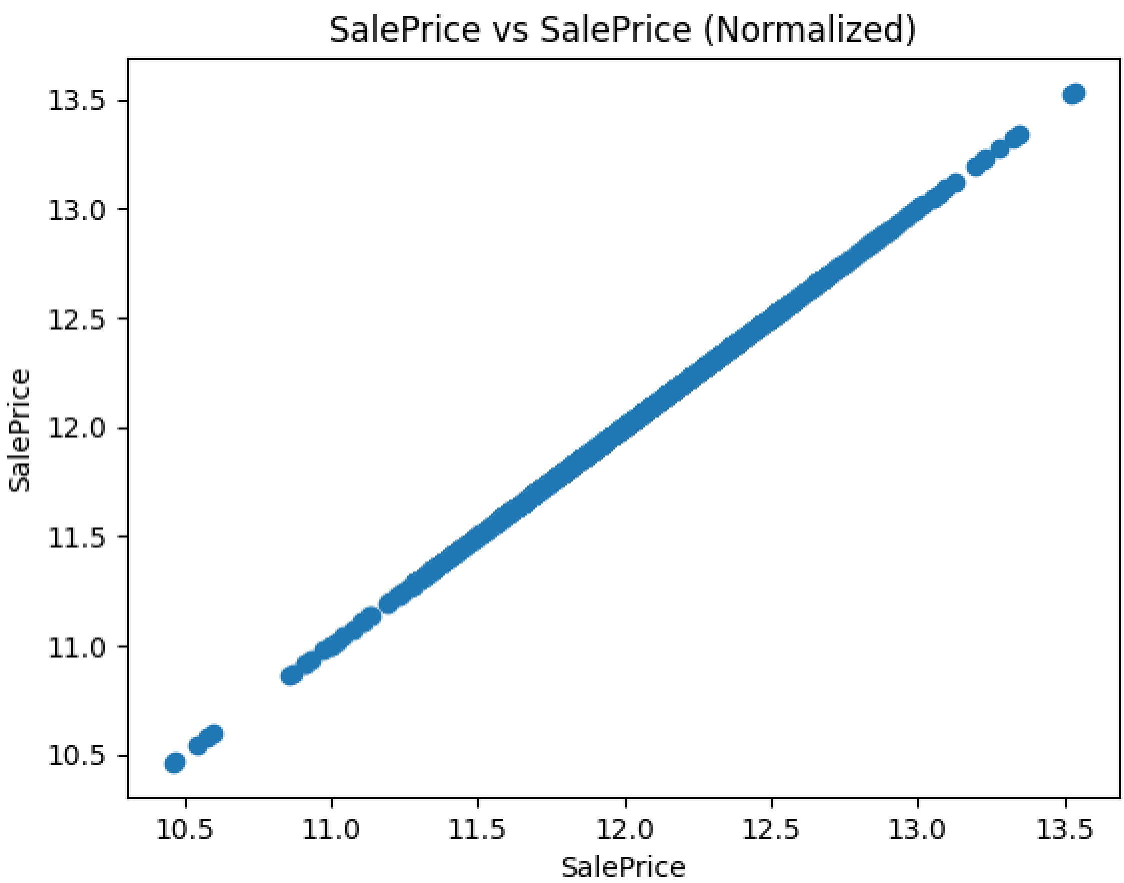
```
# use logarithmic transformation to normalize the above skewed data
skewed_num_features=['LotFrontage', 'LotArea', '1stFlrSF', 'GrLivArea', 'SalePri

for feature in skewed_num_features:
    if 0 in data[feature].unique(): # because log(0) is not defined
        pass
    else:
        dataset[feature]=np.log(dataset[feature])

        plt.scatter(dataset[feature], dataset['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalePrice')
        plt.title(feature + ' vs SalePrice (Normalized)')
        plt.show()
```


LotFrontage vs SalePrice (Normalized)
SalePrice
LotFrontage
0
100000
200000
300000
400000
500000
600000
700000
3.0
3.5
4.0
4.5
5.0
5.5


LotArea vs SalePrice (Normalized)
SalePrice
LotArea
0
100000
200000
300000
400000
500000
600000
700000
7
8
9
10
11
12


1stFlrSF vs SalePrice (Normalized)
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
6.0
6.5
7.0
7.5
8.0
8.5
1stFlrSF


GrLivArea vs SalePrice (Normalized)
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
6.0
6.5
7.0
7.5
8.0
8.5
GrLivArea



In [437…

```
dataset.head()
```

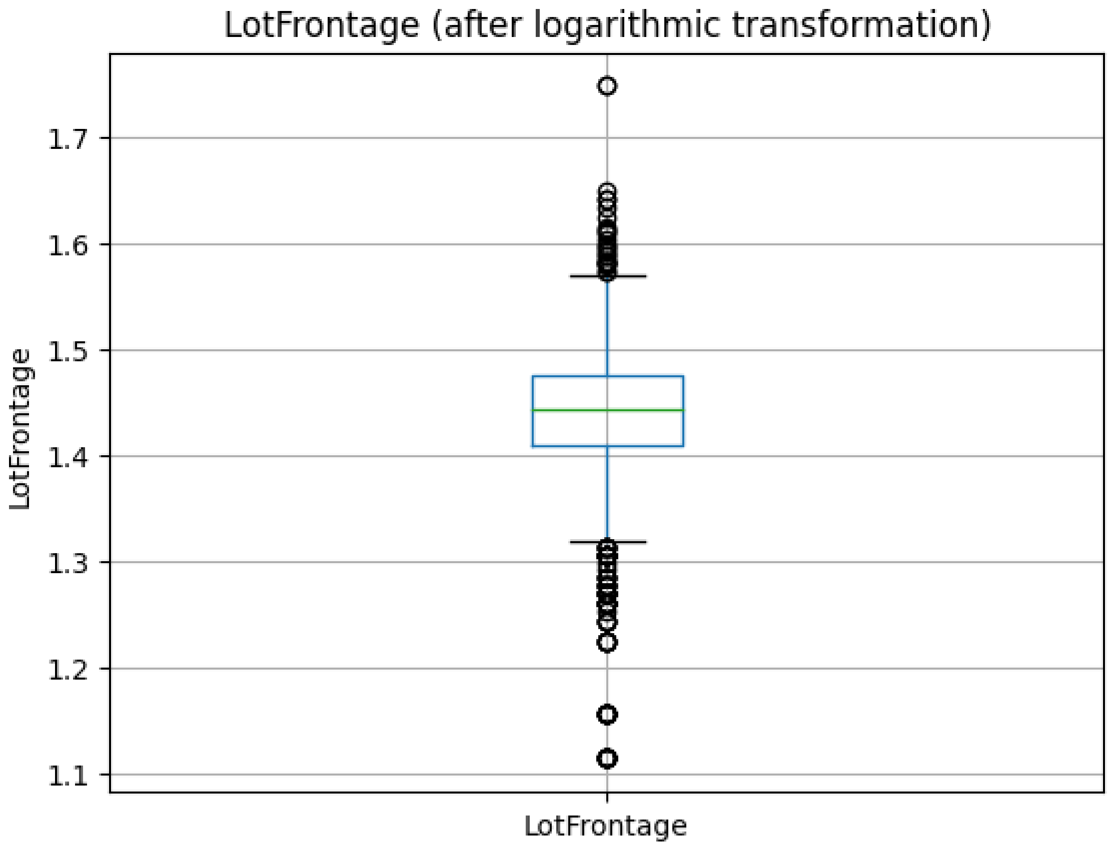
Out[437]:

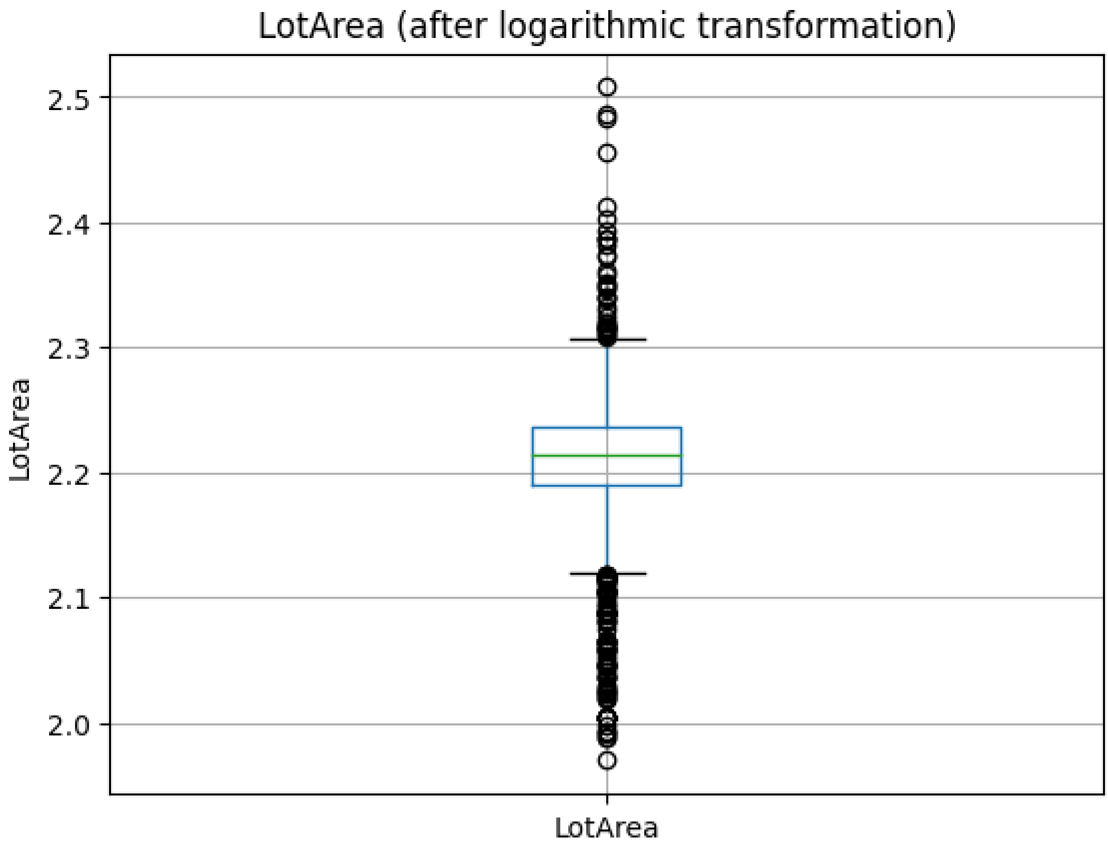
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	4.174387	9.041922	Pave	Missing	Reg	Lvl
1	2	20	RL	4.382027	9.169518	Pave	Missing	Reg	Lvl
2	3	60	RL	4.219508	9.328123	Pave	Missing	IR1	Lvl
3	4	70	RL	4.094345	9.164296	Pave	Missing	IR1	Lvl
4	5	60	RL	4.430817	9.565214	Pave	Missing	IR1	Lvl

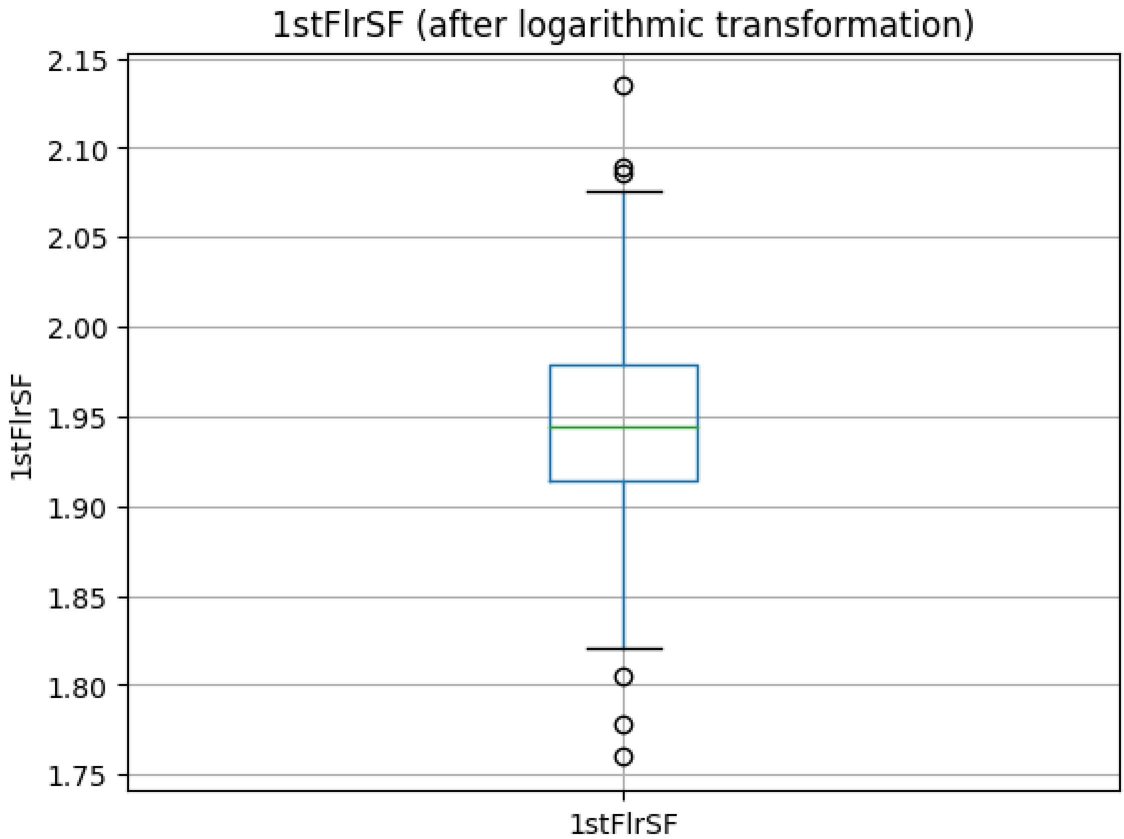
OUTLIERS

In [438…

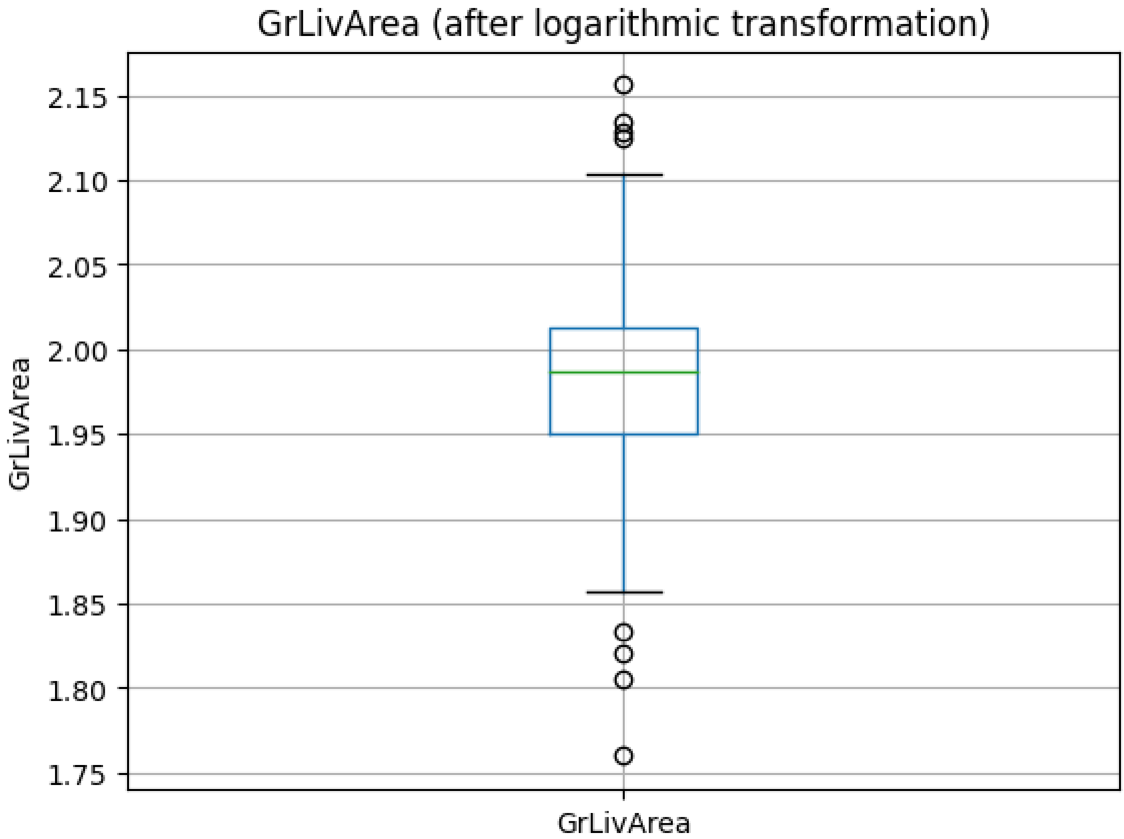
```
# use boxplots
for feature in continuous_feature:
    data=dataset.copy()
    if 0 in data[feature].unique():
        pass
    else:
        data[feature]=np.log(data[feature])

        data.boxplot(column = feature)
        plt.ylabel(feature)
        plt.title(feature + " (after logarithmic transformation)")
        plt.show()
```


LotFrontage (after logarithmic transformation)
1.7
1.6
1.5
1.4
1.3
1.2
1.1
LotFrontage
LotFrontage

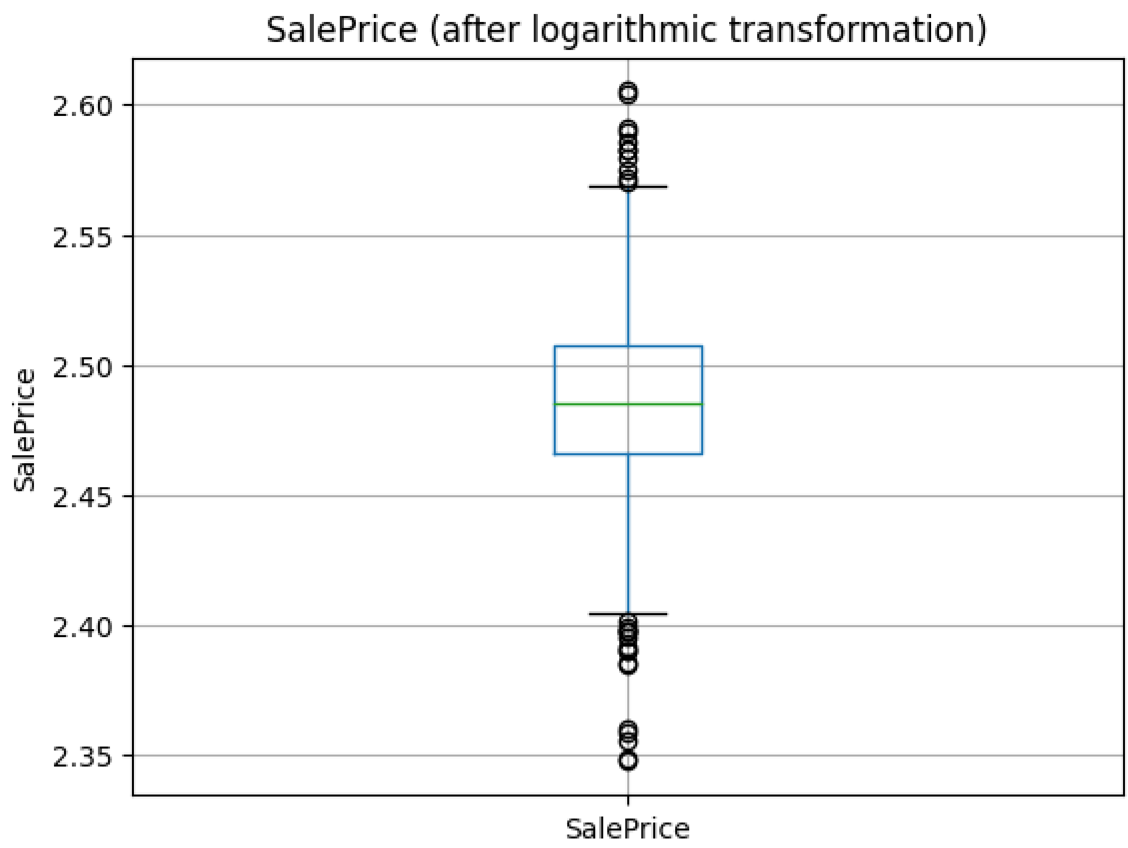

LotArea (after logarithmic transformation)
2.5
2.4
2.3
2.2
2.1
2.0
LotArea
LotArea



1stFlrSF (after logarithmic transformation)
2.15
2.10
2.05
2.00
1.95
1.90
1.85
1.80
1.75
1stFlrSF
1stFlrSF



GrLivArea (after logarithmic transformation)
2.15
2.10
2.05
2.00
1.95
1.90
1.85
1.80
1.75
GrLivArea
GrLivArea



CATEGORICAL VARIABLES

In [439…

```
# find all categorical features
categorical_features = []
for feature in dataset.columns:
  if data[feature].dtypes == 'O':
    categorical_features.append(feature)

categorical_features
```

```
Out[439]: ['MSZoning',
 'Street',
 'Alley',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 'KitchenQual',
 'Functional',
 'FireplaceQu',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PavedDrive',
 'PoolQC',
 'Fence',
 'MiscFeature',
 'SaleType',
 'SaleCondition']
```

```
In [440…  dataset[categorical_features].head()
```

Out[440]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighb
0	RL	Pave	Missing	Reg	Lvl	AllPub	Inside	Gtl	
1	RL	Pave	Missing	Reg	Lvl	AllPub	FR2	Gtl	
2	RL	Pave	Missing	IR1	Lvl	AllPub	Inside	Gtl	
3	RL	Pave	Missing	IR1	Lvl	AllPub	Corner	Gtl	
4	RL	Pave	Missing	IR1	Lvl	AllPub	FR2	Gtl	

CARDINALITY OF CATEGORICAL VARIABLES

In [441…

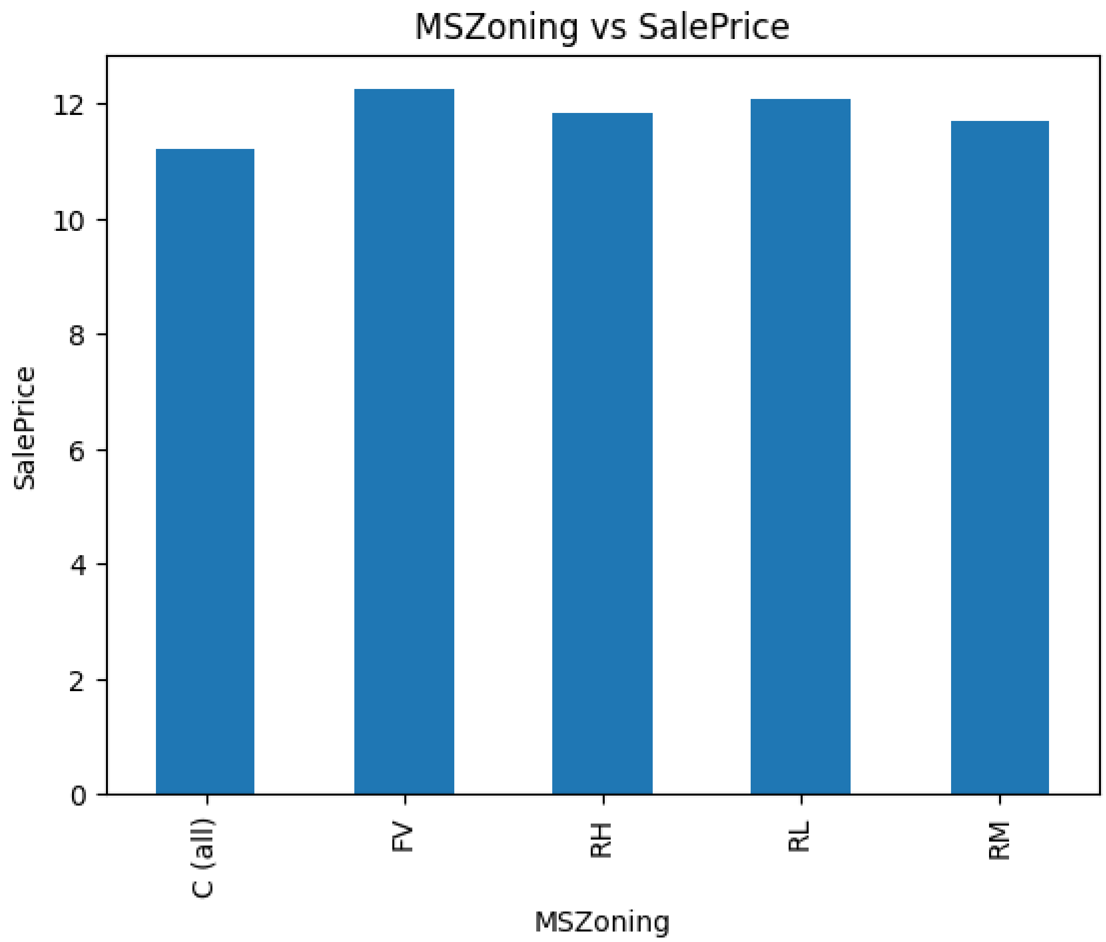
```
for feature in categorical_features:
    print('Feature: {}, Number of categories: {}'.format(feature, len(dataset[fe
```

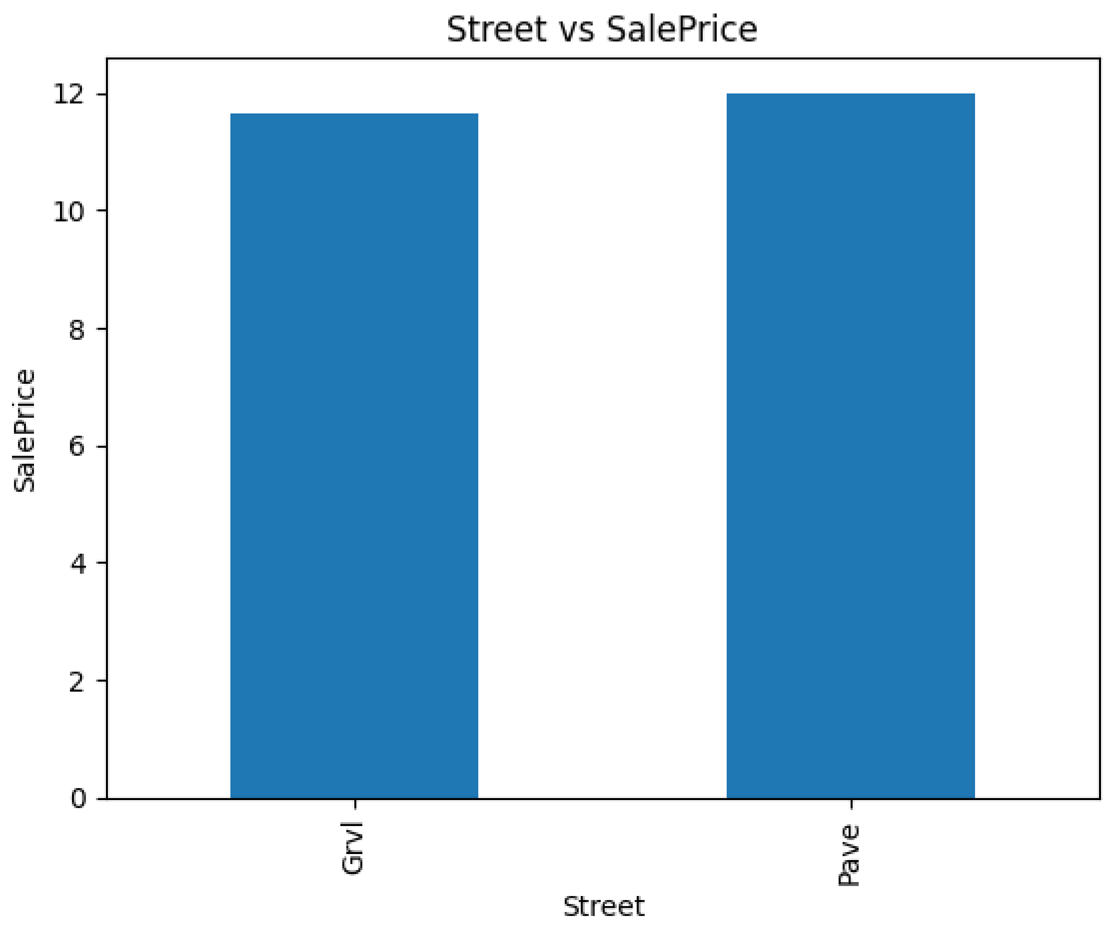
```
Feature: MSZoning, Number of categories: 5
Feature: Street, Number of categories: 2
Feature: Alley, Number of categories: 3
Feature: LotShape, Number of categories: 4
Feature: LandContour, Number of categories: 4
Feature: Utilities, Number of categories: 2
Feature: LotConfig, Number of categories: 5
Feature: LandSlope, Number of categories: 3
Feature: Neighborhood, Number of categories: 25
Feature: Condition1, Number of categories: 9
Feature: Condition2, Number of categories: 8
Feature: BldgType, Number of categories: 5
Feature: HouseStyle, Number of categories: 8
Feature: RoofStyle, Number of categories: 6
Feature: RoofMatl, Number of categories: 8
Feature: Exterior1st, Number of categories: 15
Feature: Exterior2nd, Number of categories: 16
Feature: MasVnrType, Number of categories: 5
Feature: ExterQual, Number of categories: 4
Feature: ExterCond, Number of categories: 5
Feature: Foundation, Number of categories: 6
Feature: BsmtQual, Number of categories: 5
Feature: BsmtCond, Number of categories: 5
Feature: BsmtExposure, Number of categories: 5
Feature: BsmtFinType1, Number of categories: 7
Feature: BsmtFinType2, Number of categories: 7
Feature: Heating, Number of categories: 6
Feature: HeatingQC, Number of categories: 5
Feature: CentralAir, Number of categories: 2
Feature: Electrical, Number of categories: 5
Feature: KitchenQual, Number of categories: 4
Feature: Functional, Number of categories: 7
Feature: FireplaceQu, Number of categories: 6
Feature: GarageType, Number of categories: 7
Feature: GarageFinish, Number of categories: 4
Feature: GarageQual, Number of categories: 6
Feature: GarageCond, Number of categories: 6
Feature: PavedDrive, Number of categories: 3
Feature: PoolQC, Number of categories: 4
Feature: Fence, Number of categories: 5
Feature: MiscFeature, Number of categories: 5
Feature: SaleType, Number of categories: 9
Feature: SaleCondition, Number of categories: 6
```

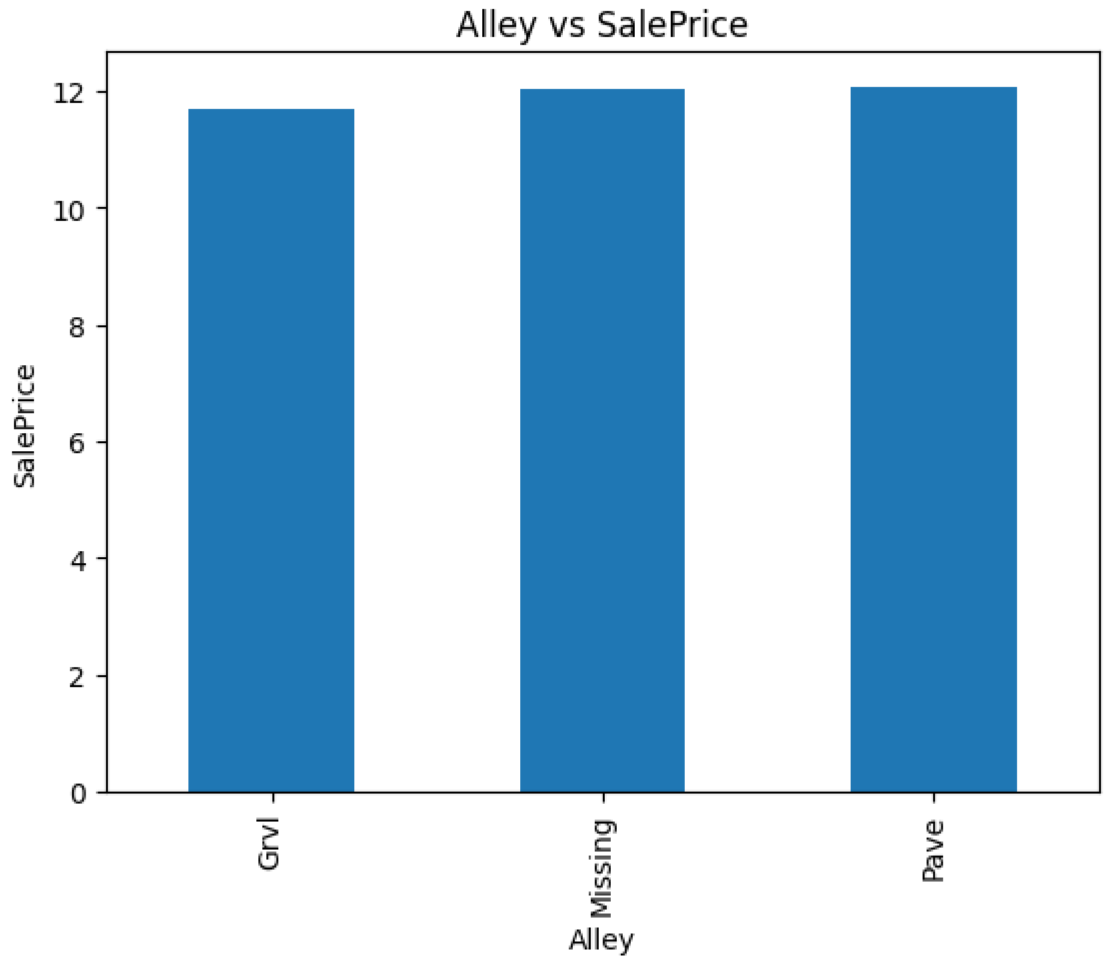
In [442…

```
# find the relationship between categorical variables and SalePrice
for feature in categorical_features:
    data = dataset.copy()
    data.groupby(feature)['SalePrice'].median().plot.bar()

    plt.xlabel(feature)
    plt.ylabel('SalePrice')
```

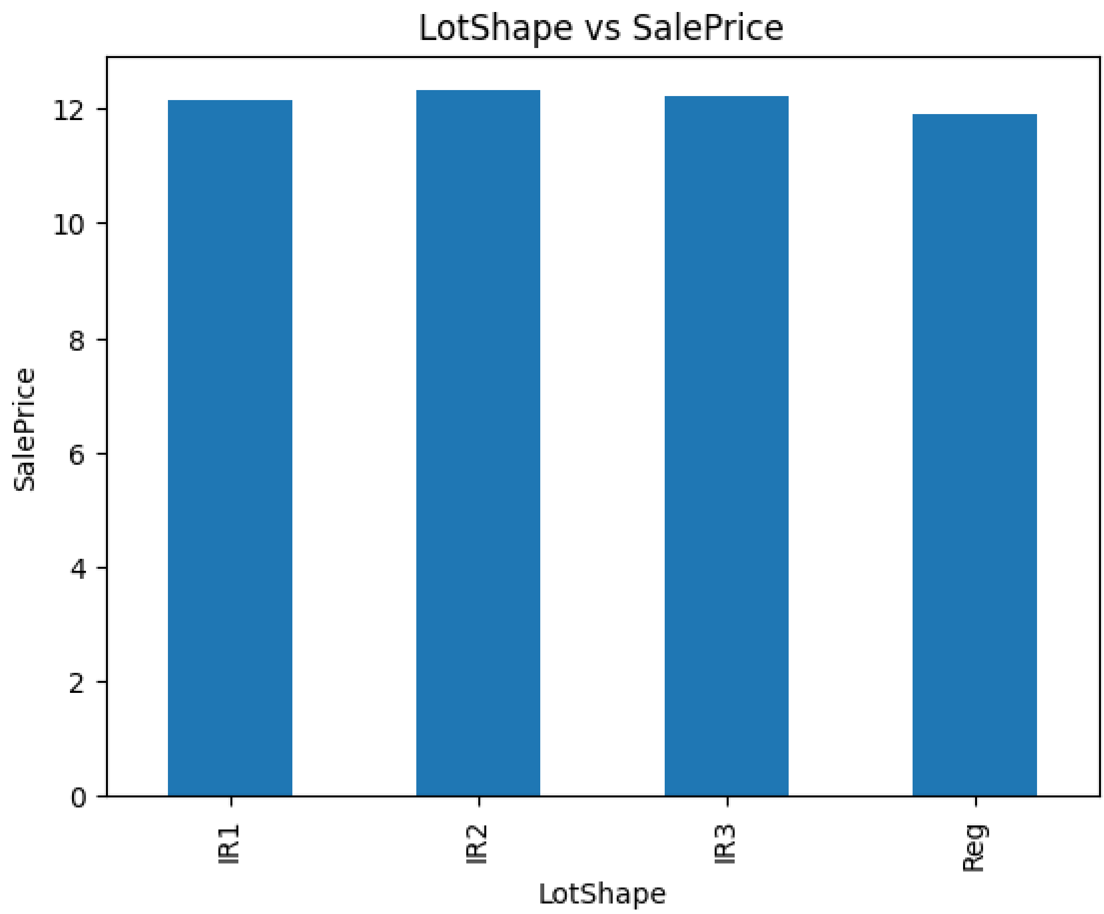
```
    plt.title(feature + ' vs SalePrice')
    plt.show()
```



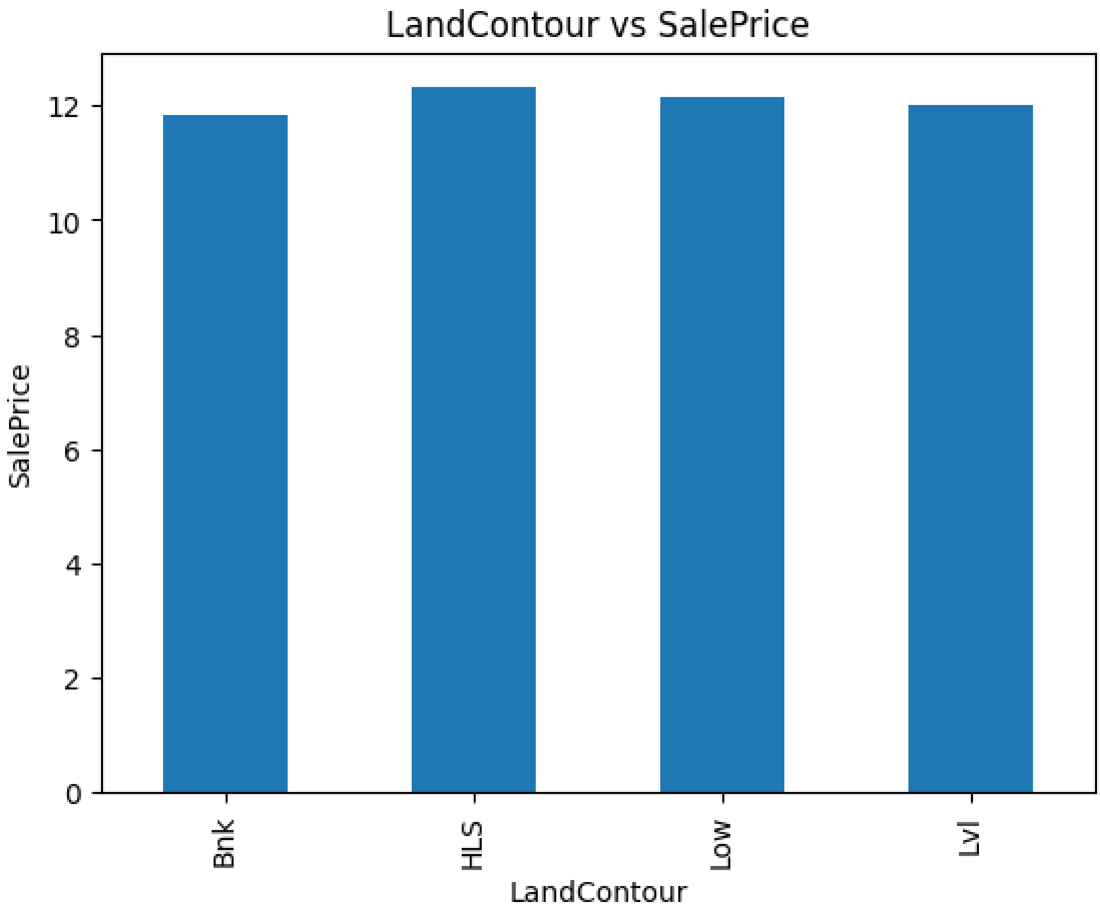


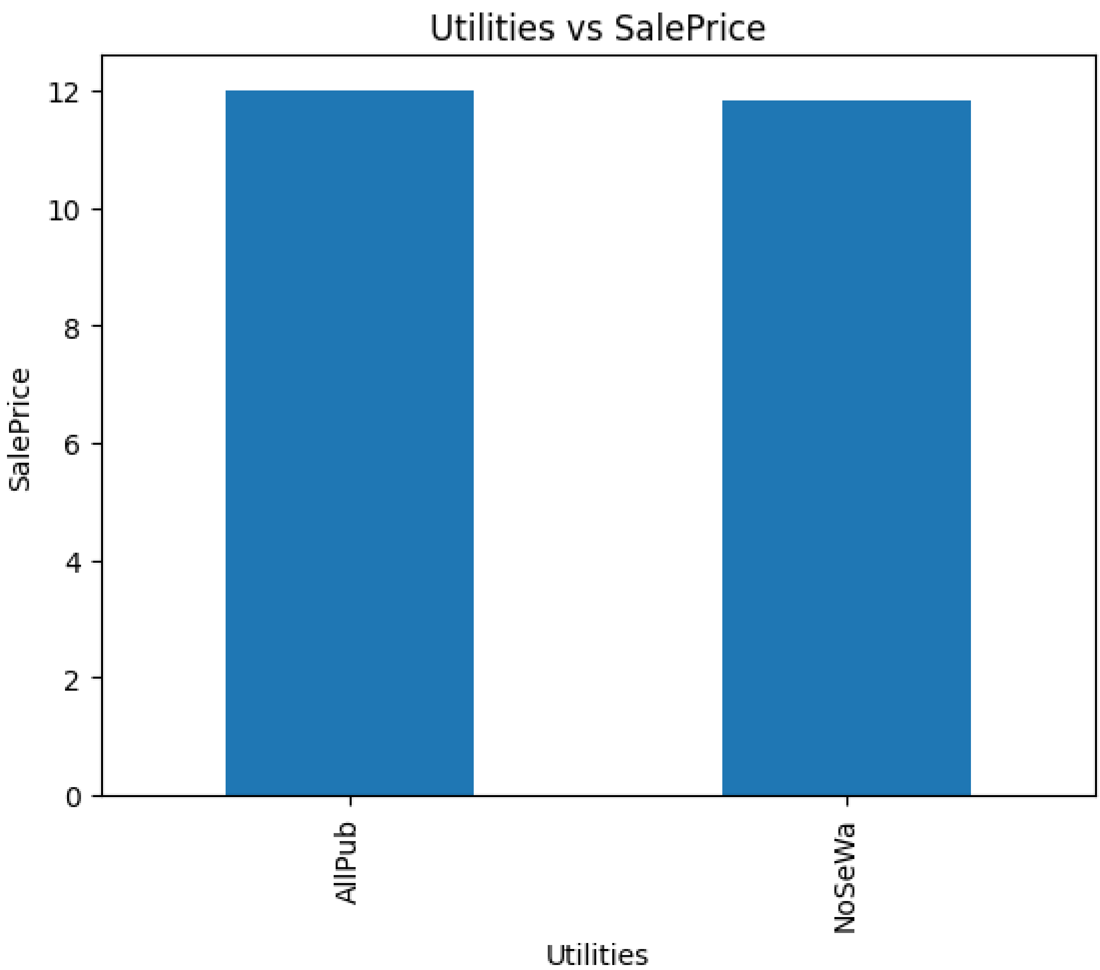


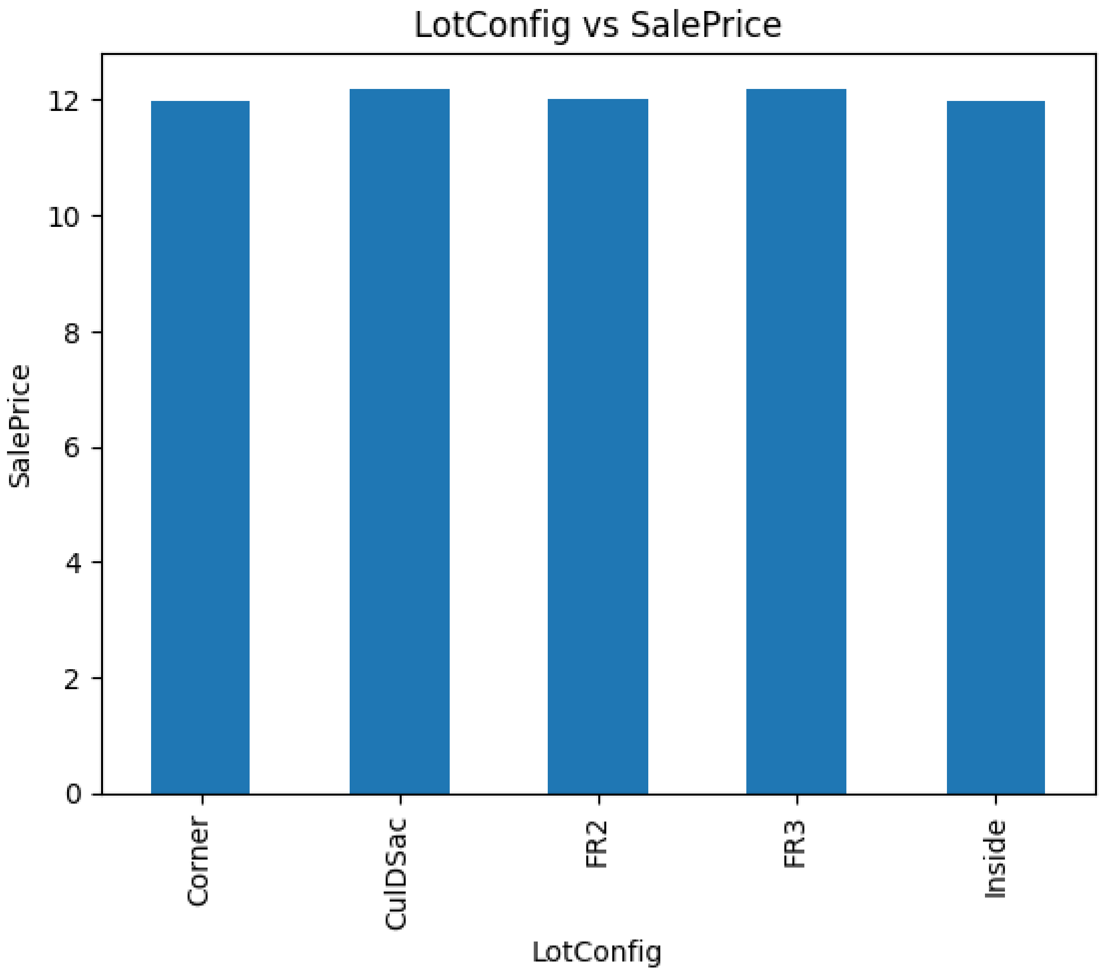
Alley vs SalePrice
SalePrice
0
2
4
6
8
10
12
Grvl
Missing
Pave
Alley

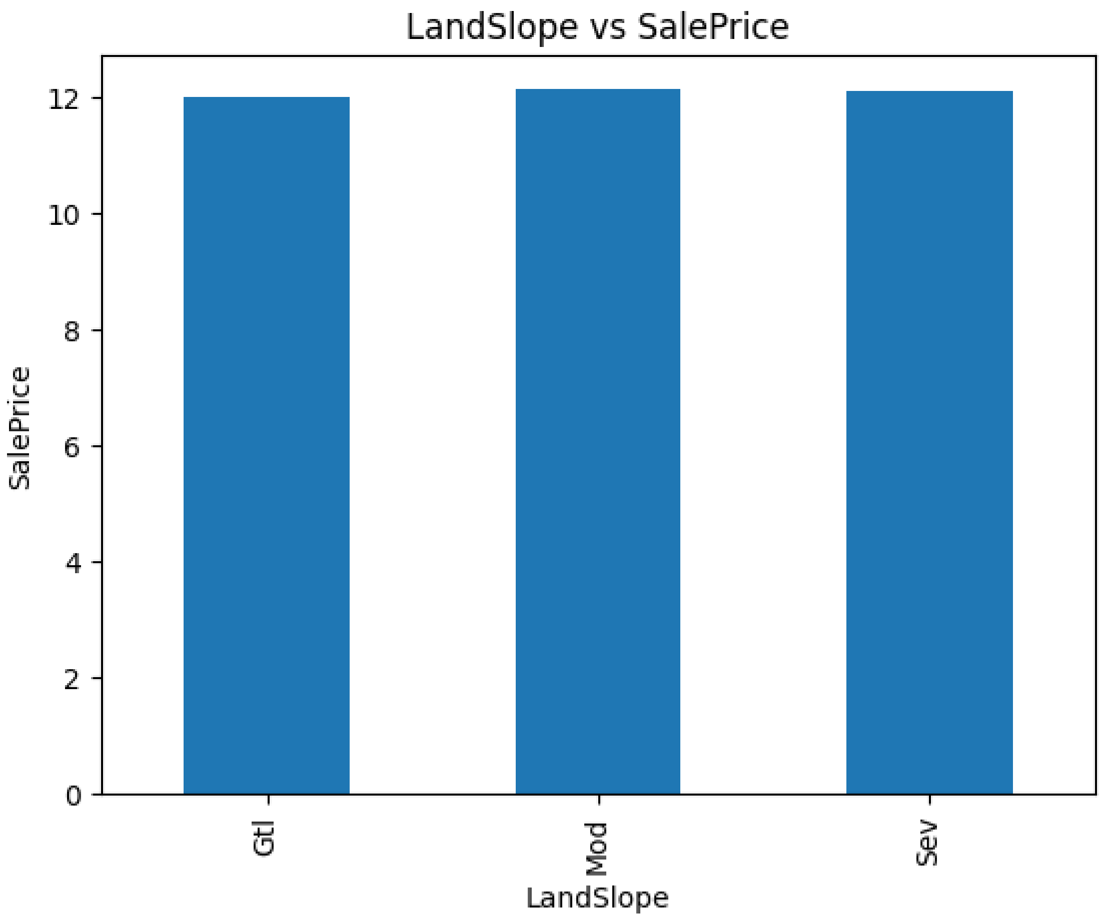


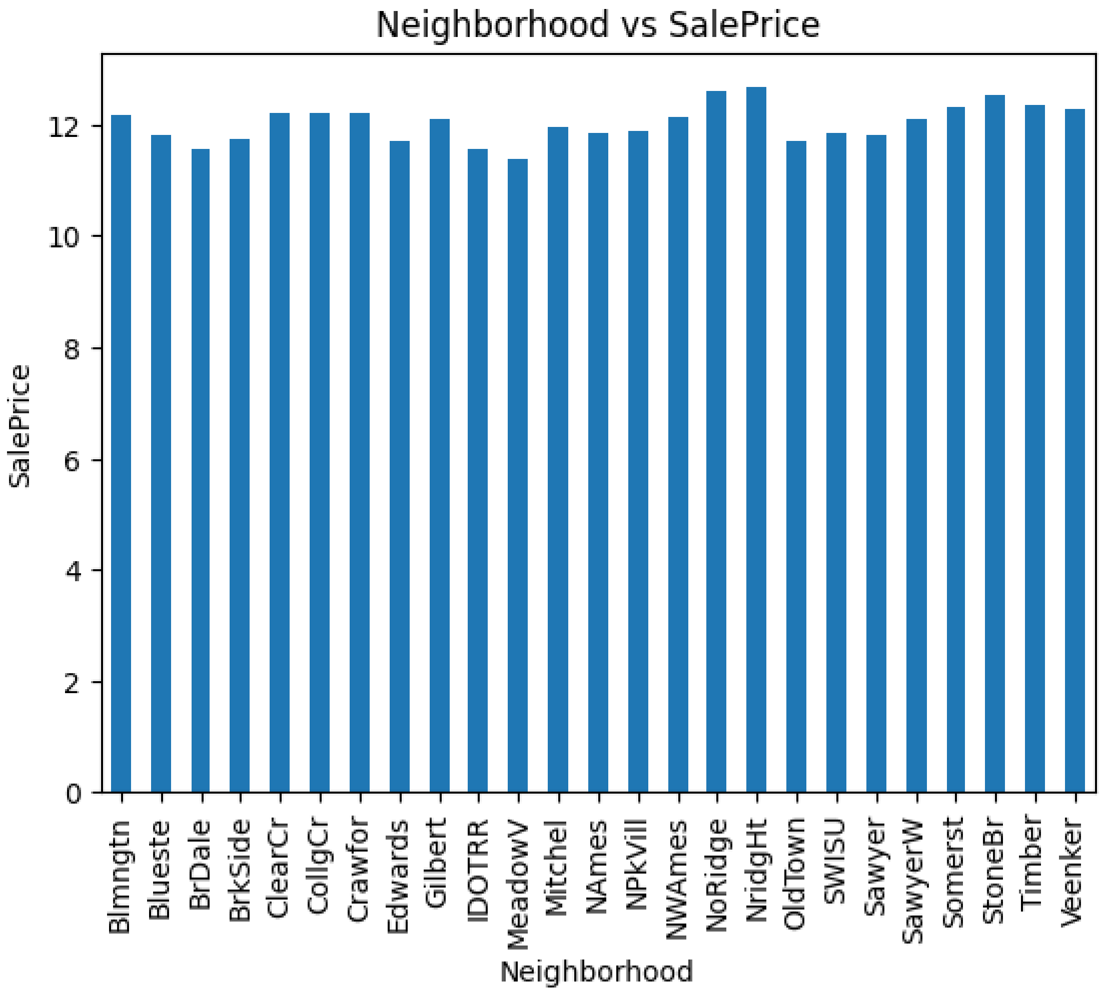
LotShape vs SalePrice
SalePrice
0
2
4
6
8
10
12
IR1
IR2
IR3
Reg
LotShape

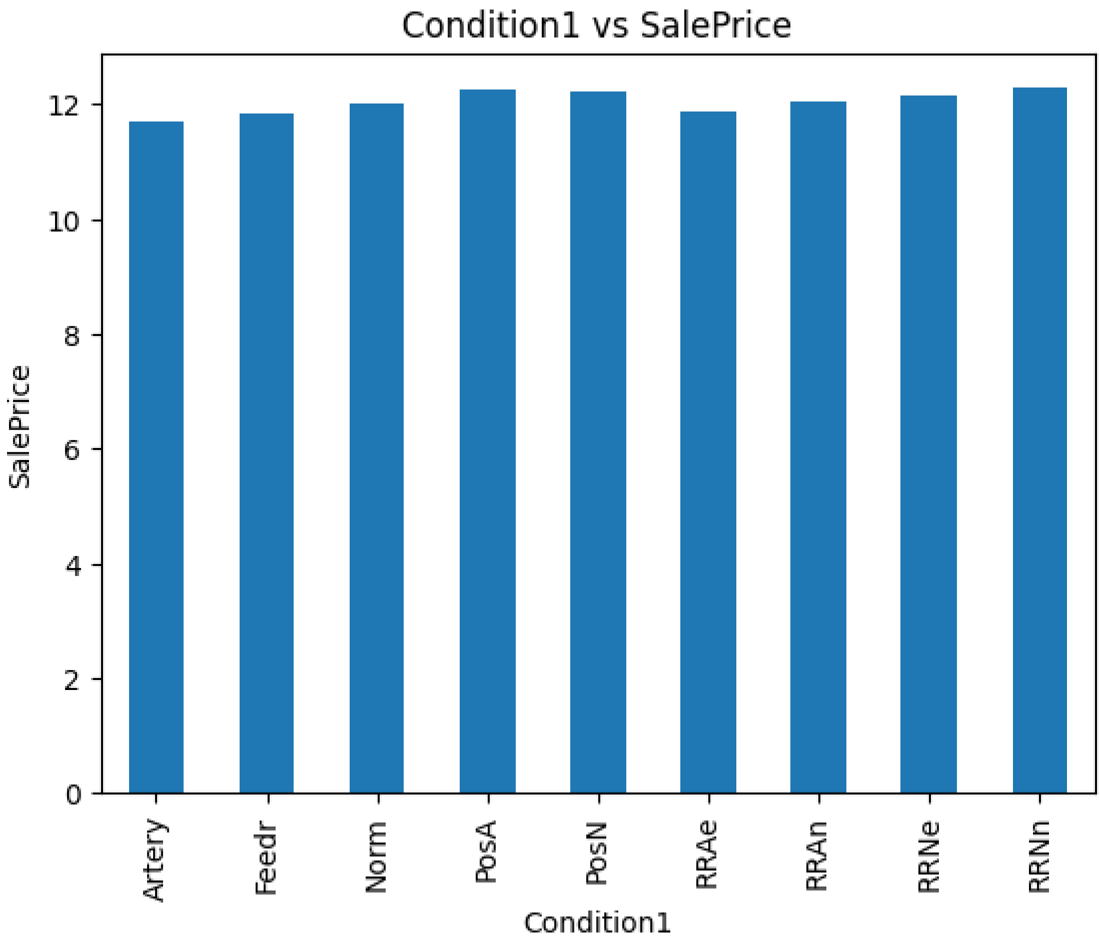

LandContour vs SalePrice
SalePrice
0
2
4
6
8
10
12
Bnk
HLS
Low
Lvl
LandContour

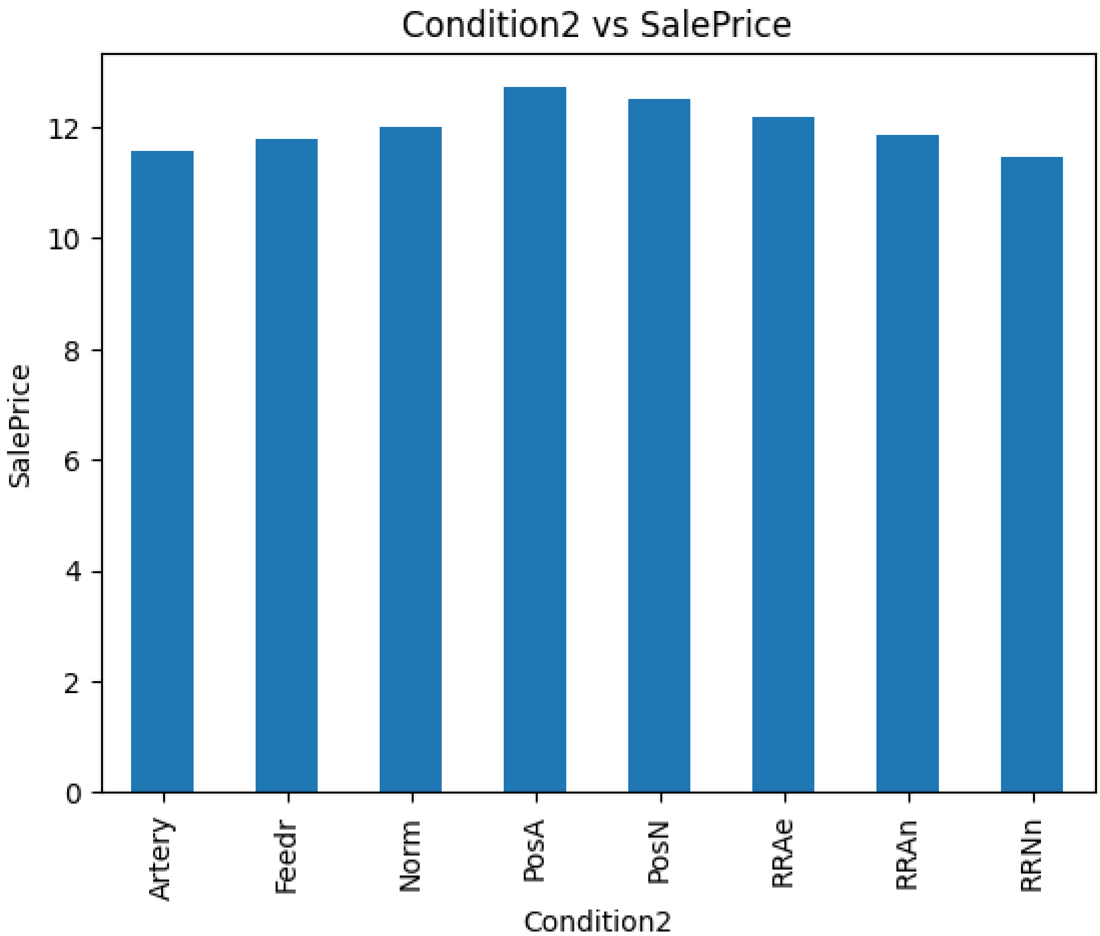

Utilities vs SalePrice
SalePrice
0
2
4
6
8
10
12
AllPub
NoSeWa
Utilities

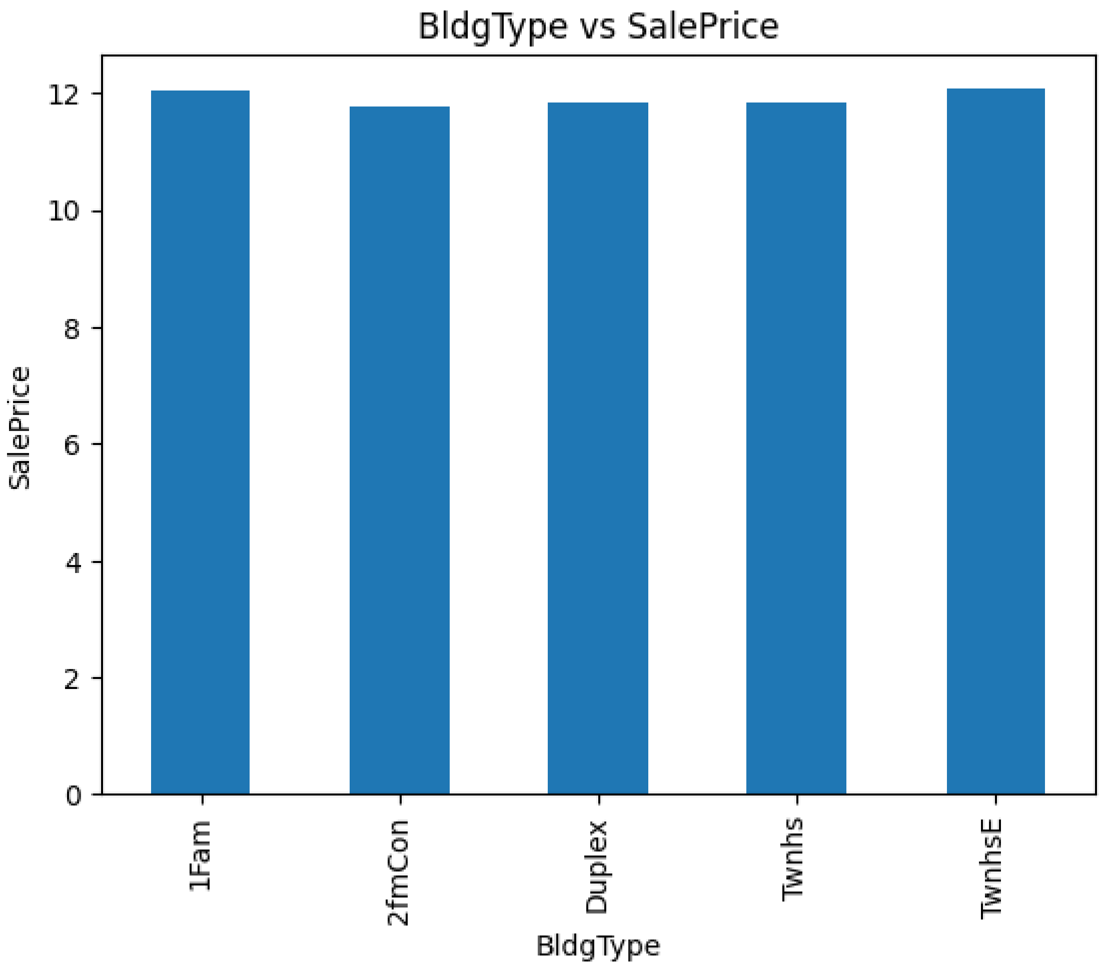

LotConfig vs SalePrice
SalePrice
0
2
4
6
8
10
12
Corner
CulDSac
FR2
FR3
Inside
LotConfig

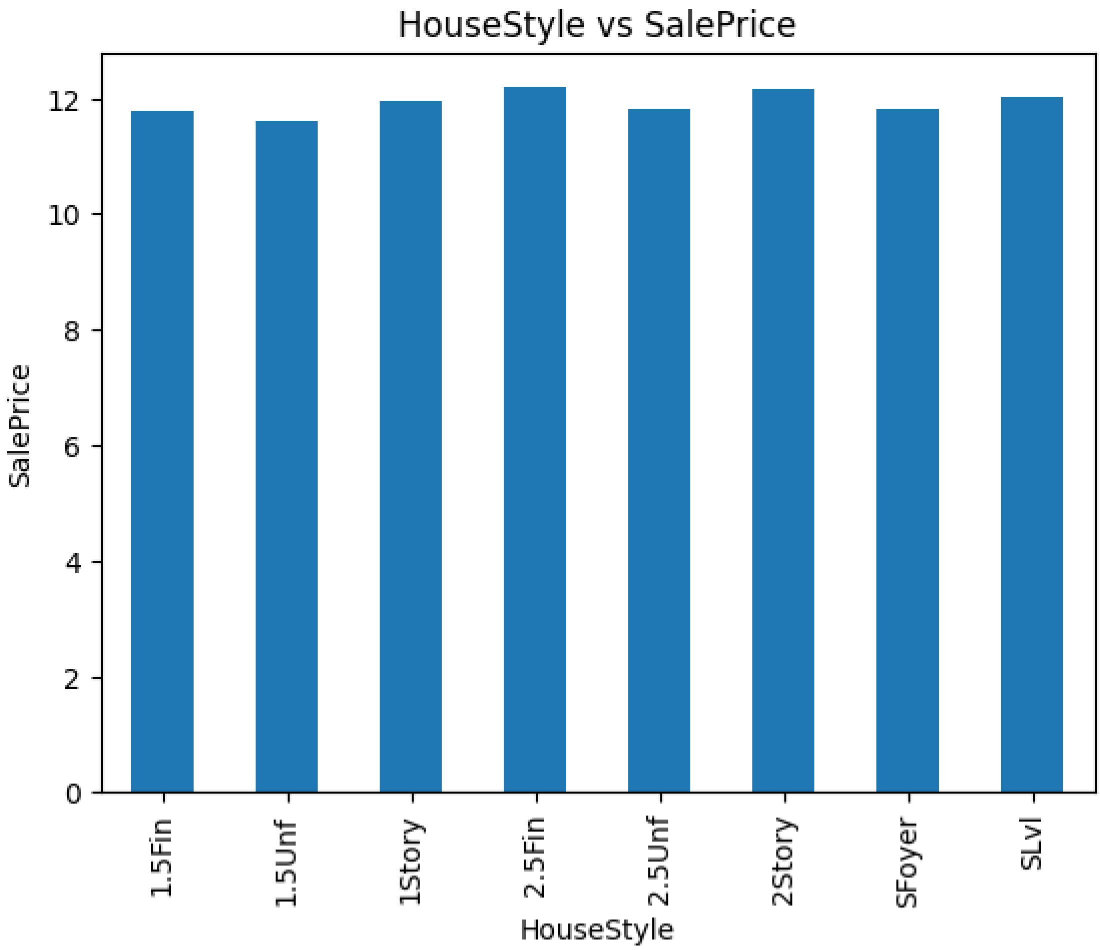

LandSlope vs SalePrice
SalePrice
0
2
4
6
8
10
12
Gtl
Mod
Sev
LandSlope

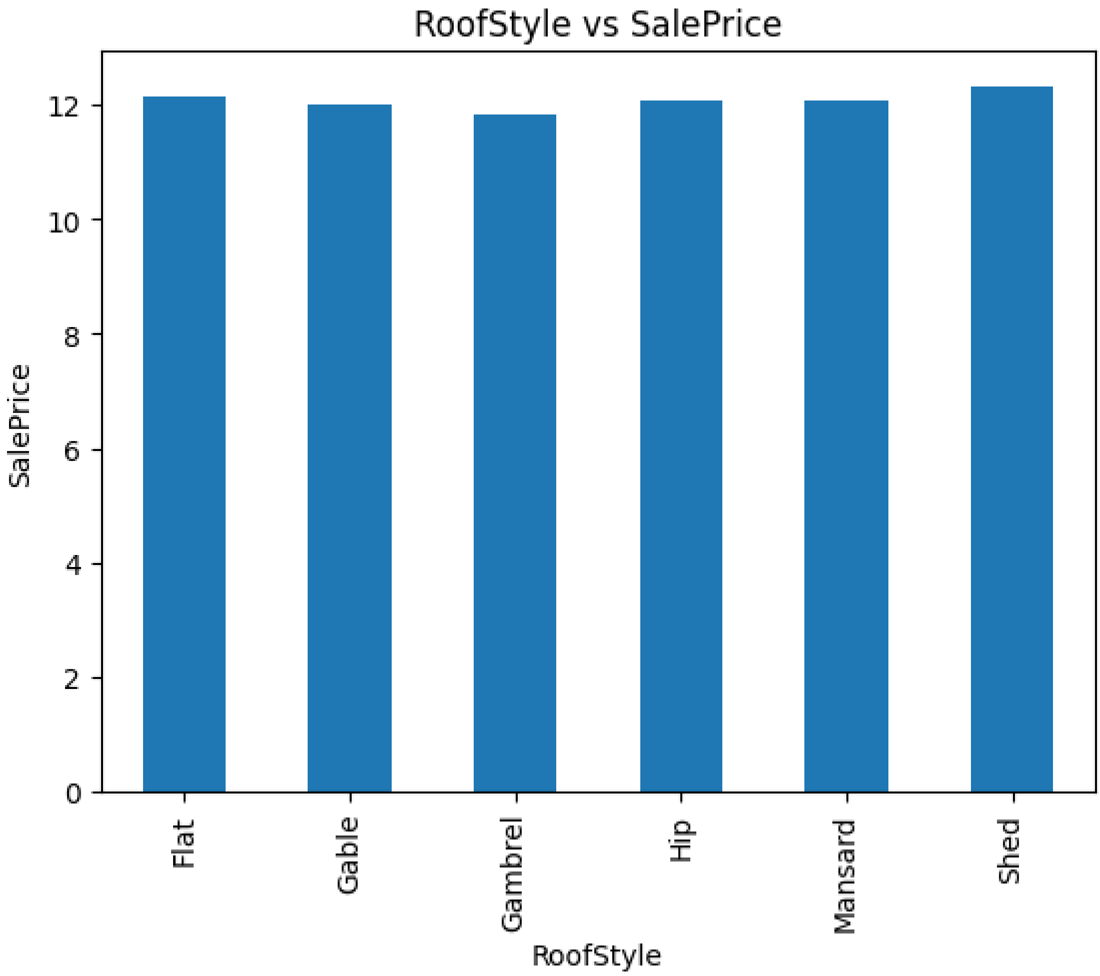

Neighborhood vs SalePrice
SalePrice
0
2
4
6
8
10
12
Blmngtn
Blueste
BrDale
BrkSide
ClearCr
CollgCr
Crawfor
Edwards
Gilbert
IDOTRR
MeadowV
Mitchel
NAmes
NPkVill
NWAmes
NoRidge
NridgHt
OldTown
SWISU
Sawyer
SawyerW
Somerst
StoneBr
Timber
Veenker
Neighborhood

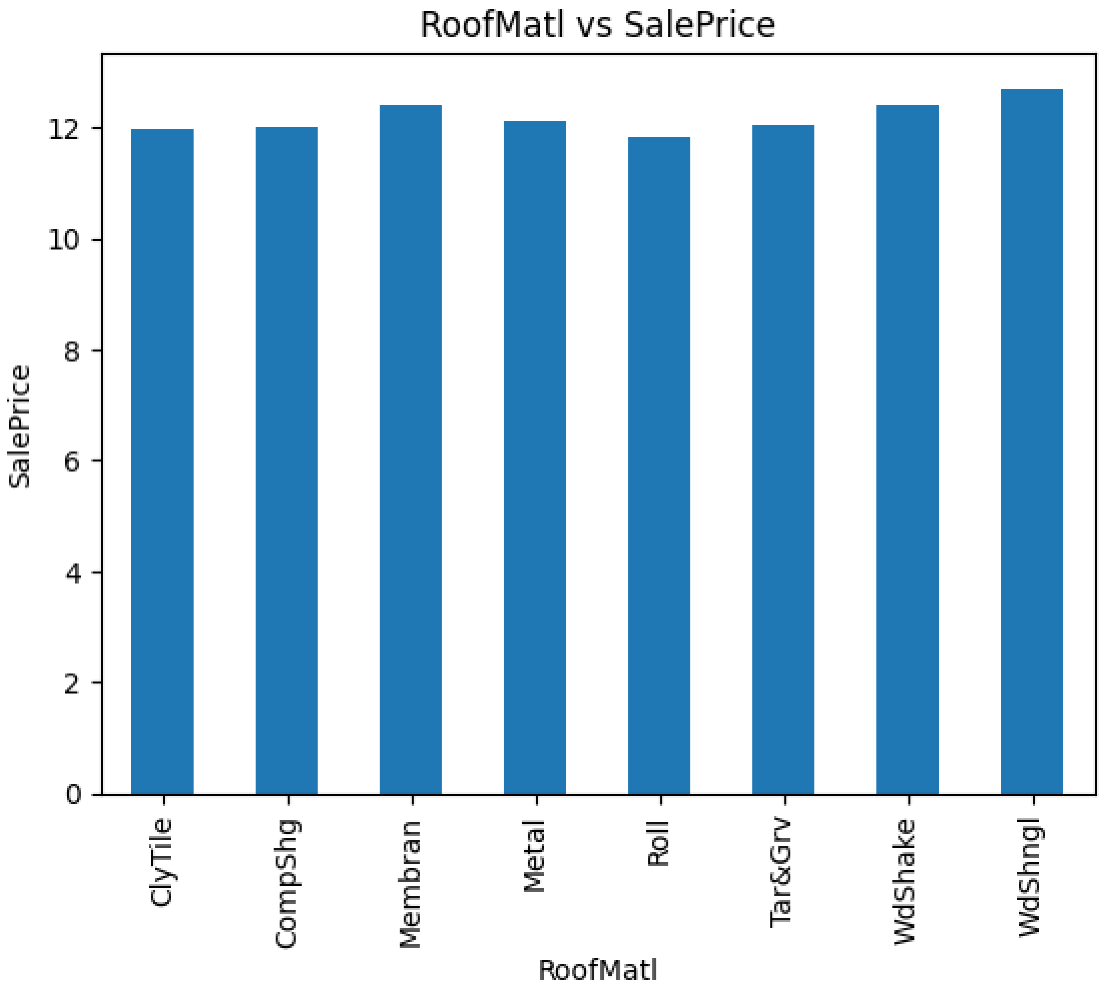

Condition1 vs SalePrice
SalePrice
0
2
4
6
8
10
12
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNe
RRNn
Condition1

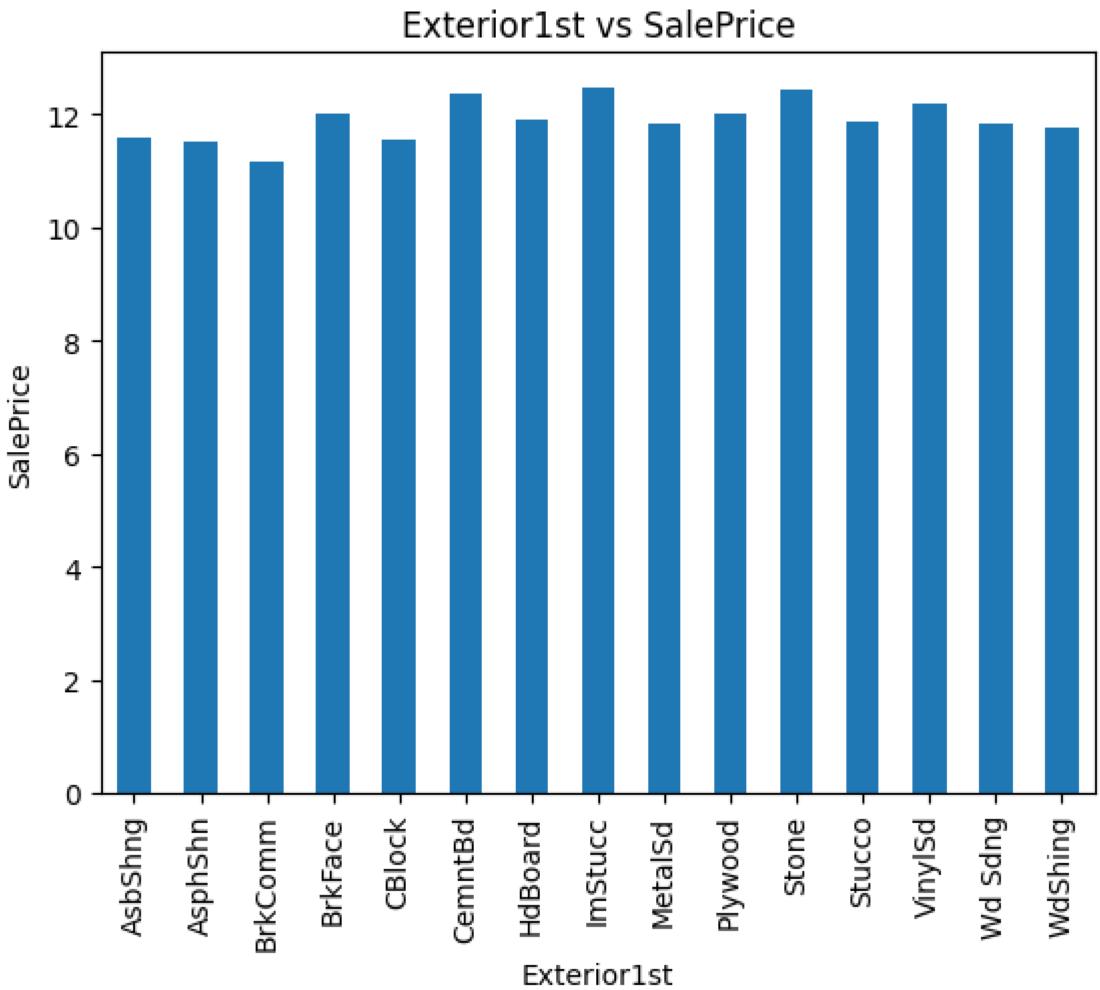

Condition2 vs SalePrice
SalePrice
0
2
4
6
8
10
12
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNn
Condition2

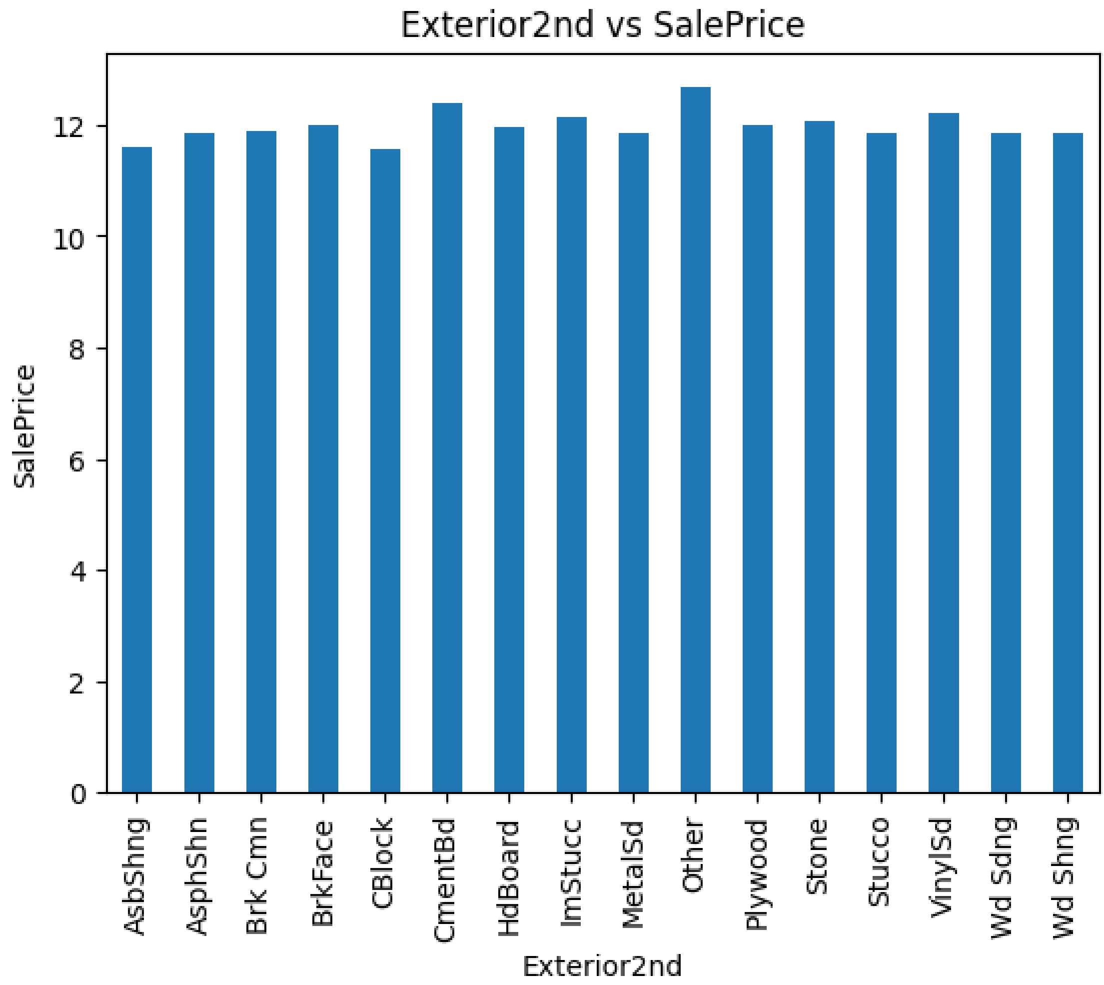

BldgType vs SalePrice
SalePrice
0
2
4
6
8
10
12
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
BldgType

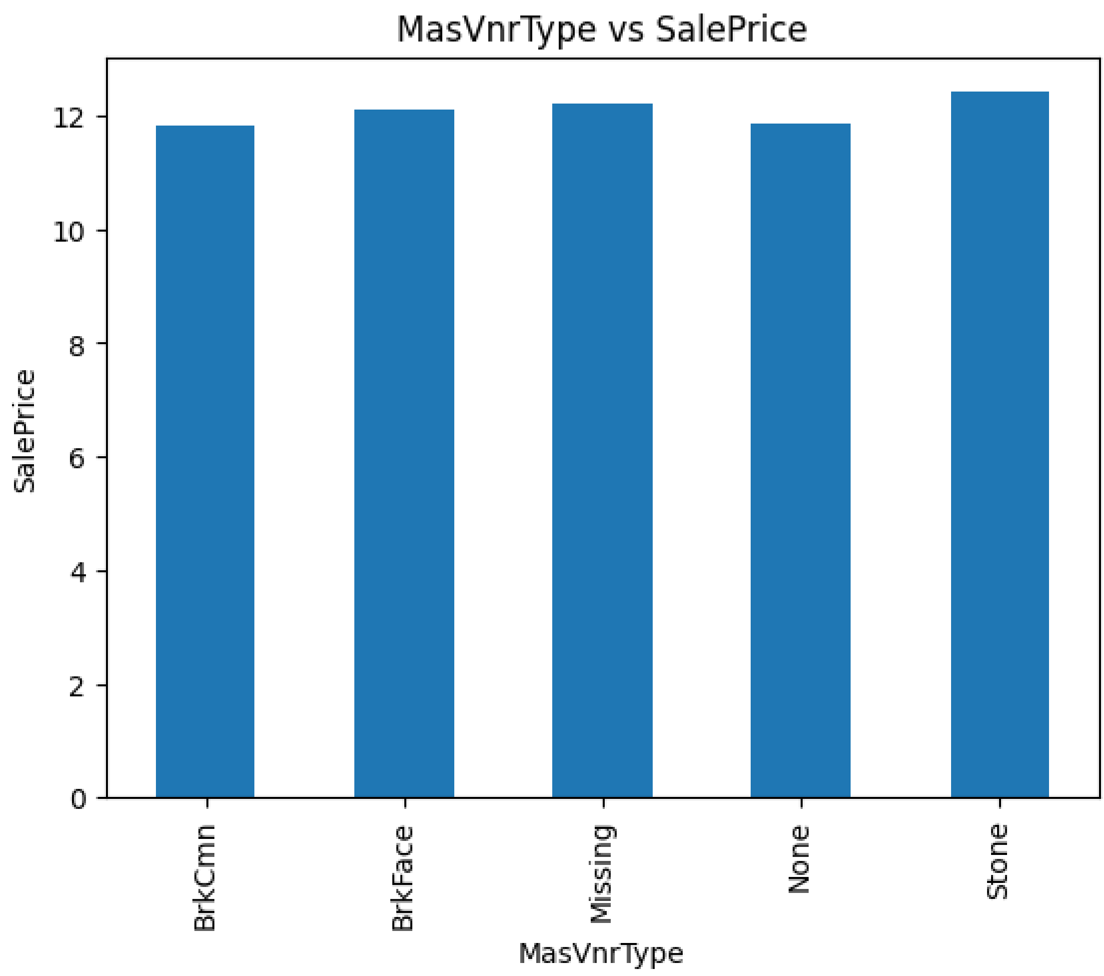

HouseStyle vs SalePrice
SalePrice
0
2
4
6
8
10
12
1.5Fin
1.5Unf
1Story
2.5Fin
2.5Unf
2Story
SFoyer
SLvl
HouseStyle

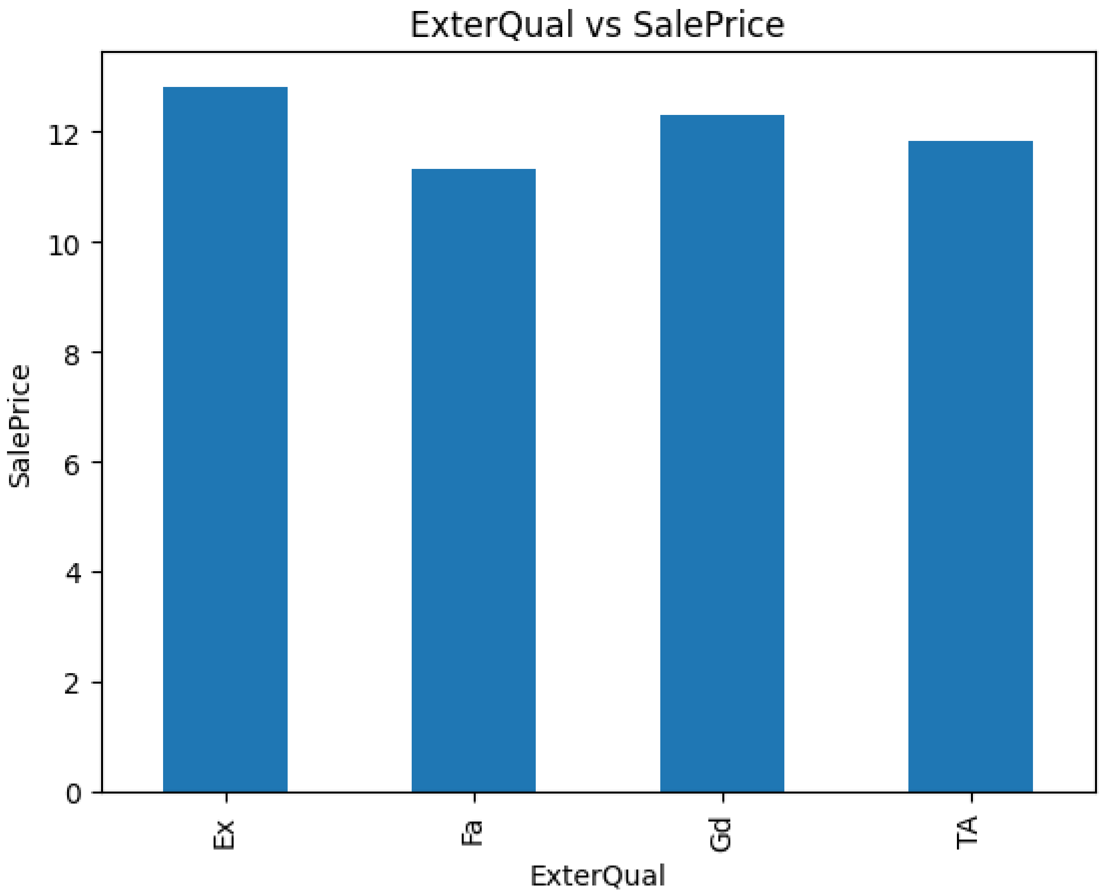

RoofStyle vs SalePrice
SalePrice
0
2
4
6
8
10
12
Flat
Gable
Gambrel
Hip
Mansard
Shed
RoofStyle

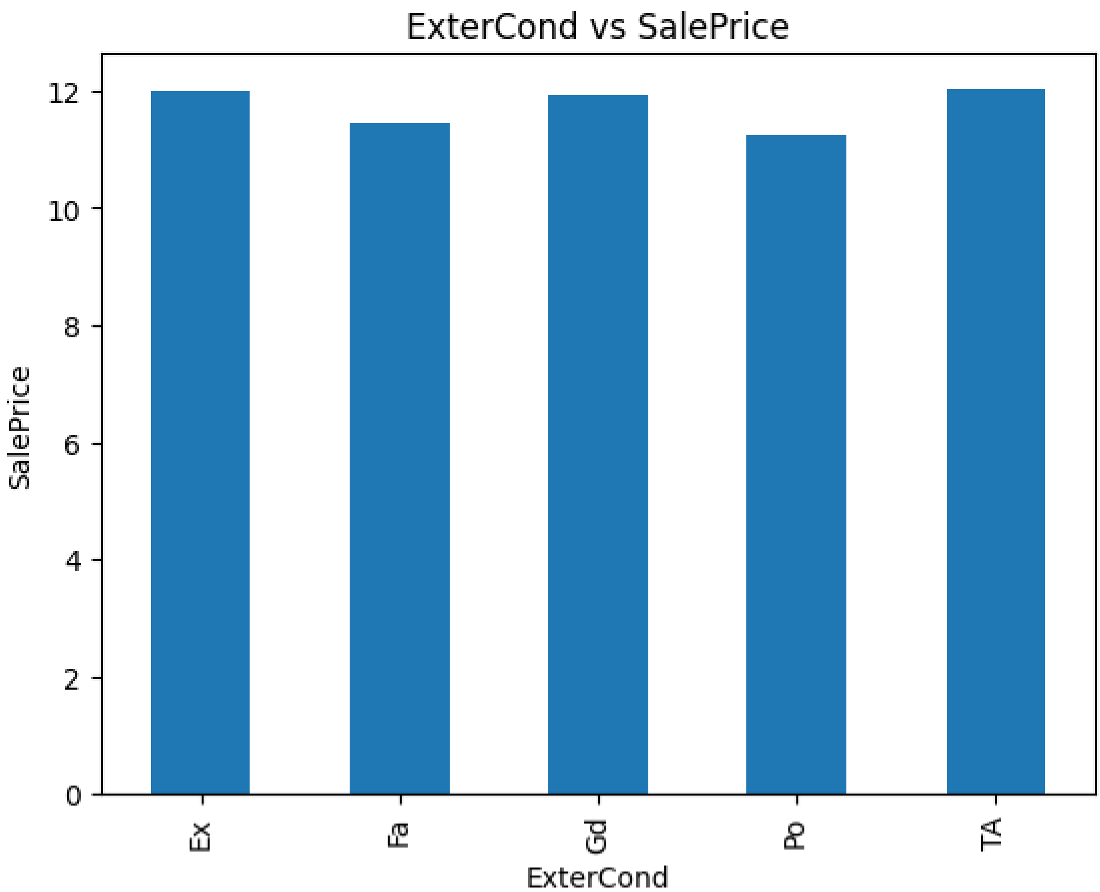

RoofMatl vs SalePrice
SalePrice
0
2
4
6
8
10
12
ClyTile
CompShg
Membran
Metal
Roll
Tar&Grv
WdShake
WdShngl
RoofMatl

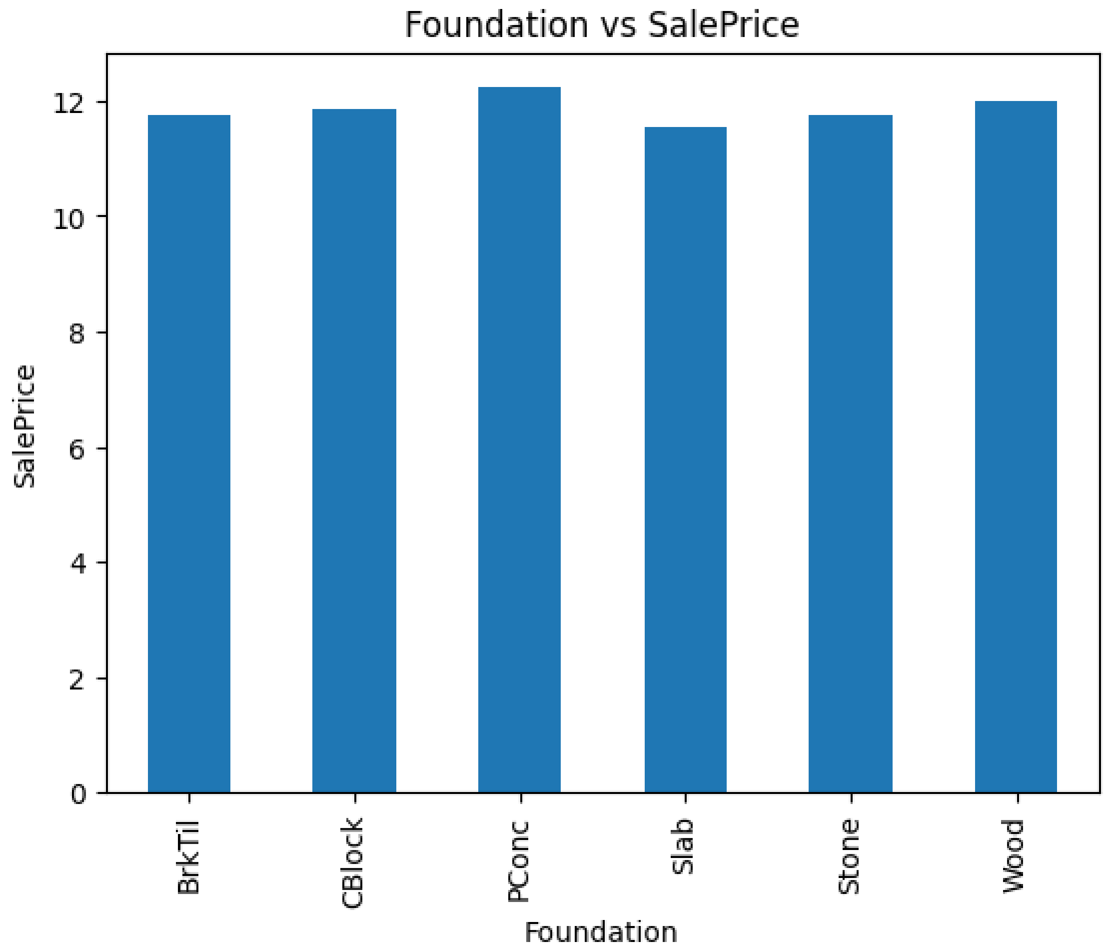

Exterior1st vs SalePrice
SalePrice
0
2
4
6
8
10
12
AsbShng
AsphShn
BrkComm
BrkFace
CBlock
CemntBd
HdBoard
ImStucc
MetalSd
Plywood
Stone
Stucco
VinylSd
Wd Sdng
WdShing
Exterior1st

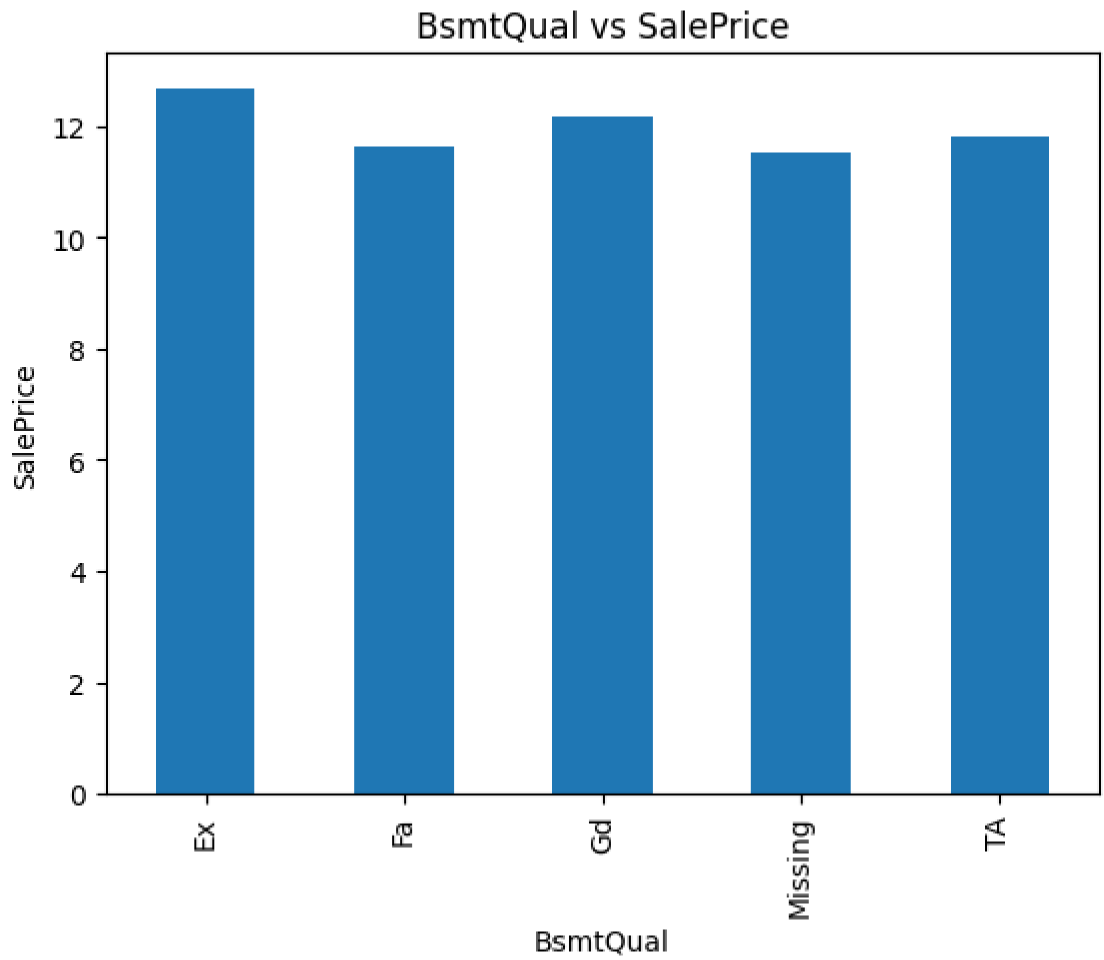

Exterior2nd vs SalePrice
SalePrice
0
2
4
6
8
10
12
AsbShng
AsphShn
Brk Cmn
BrkFace
CBlock
CmentBd
HdBoard
ImStucc
MetalSd
Other
Plywood
Stone
Stucco
VinylSd
Wd Sdng
Wd Shng
Exterior2nd

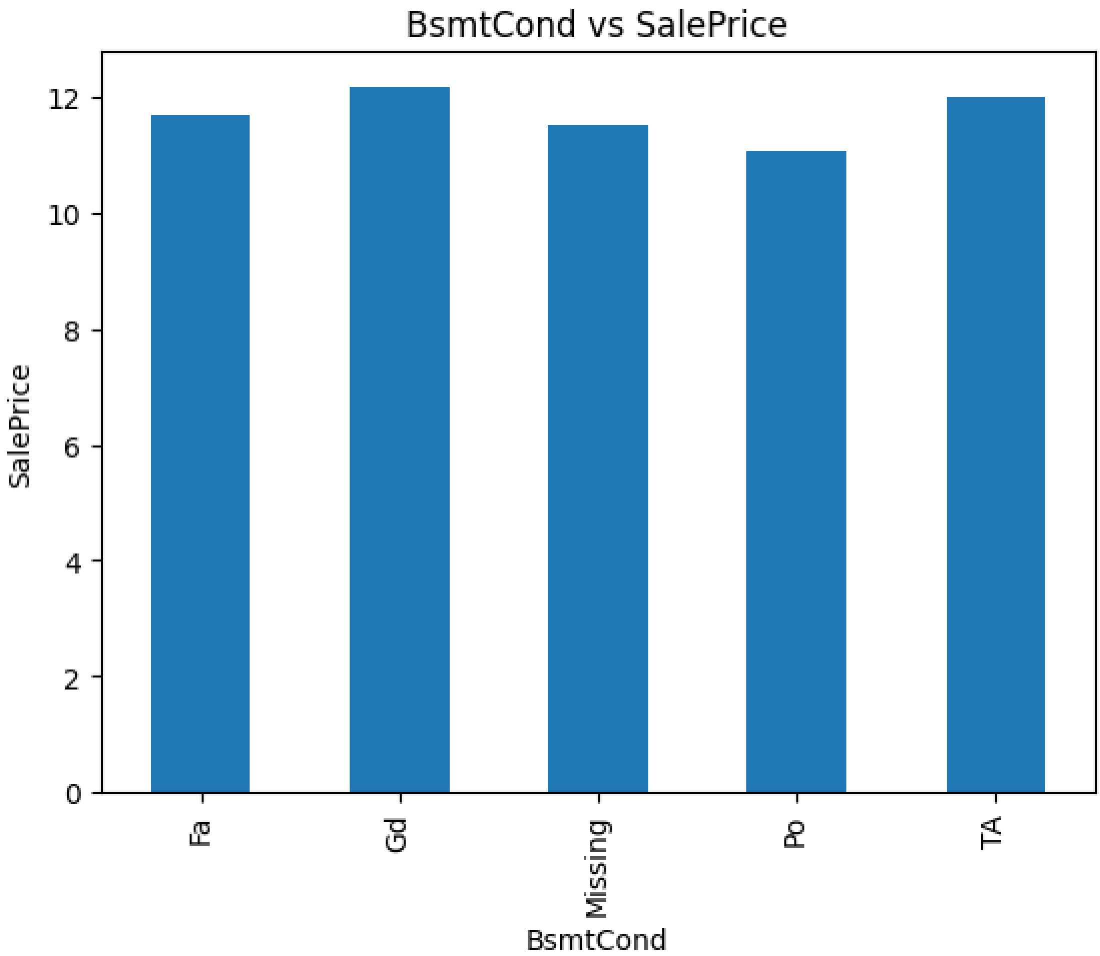

MasVnrType vs SalePrice
SalePrice
0
2
4
6
8
10
12
BrkCmn
BrkFace
Missing
None
Stone
MasVnrType

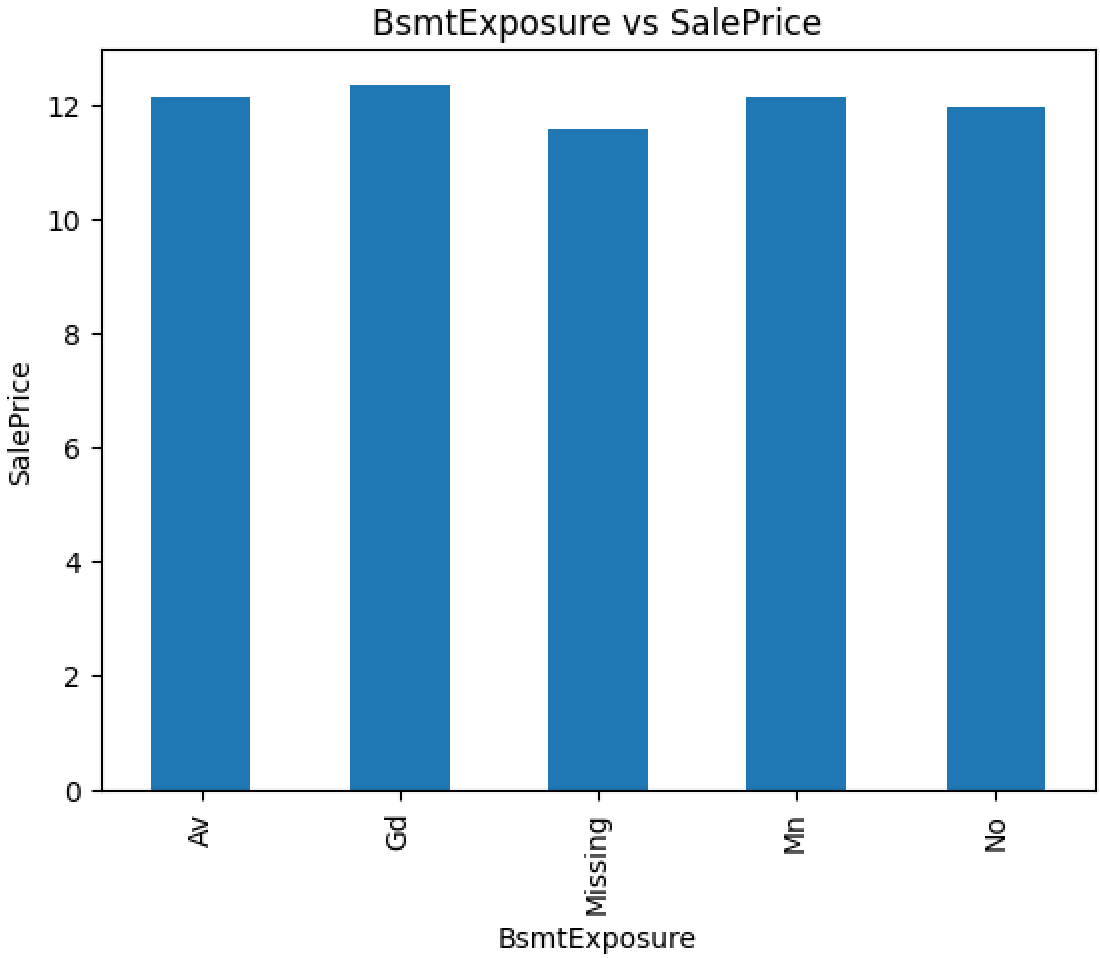

ExterQual vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
TA
ExterQual

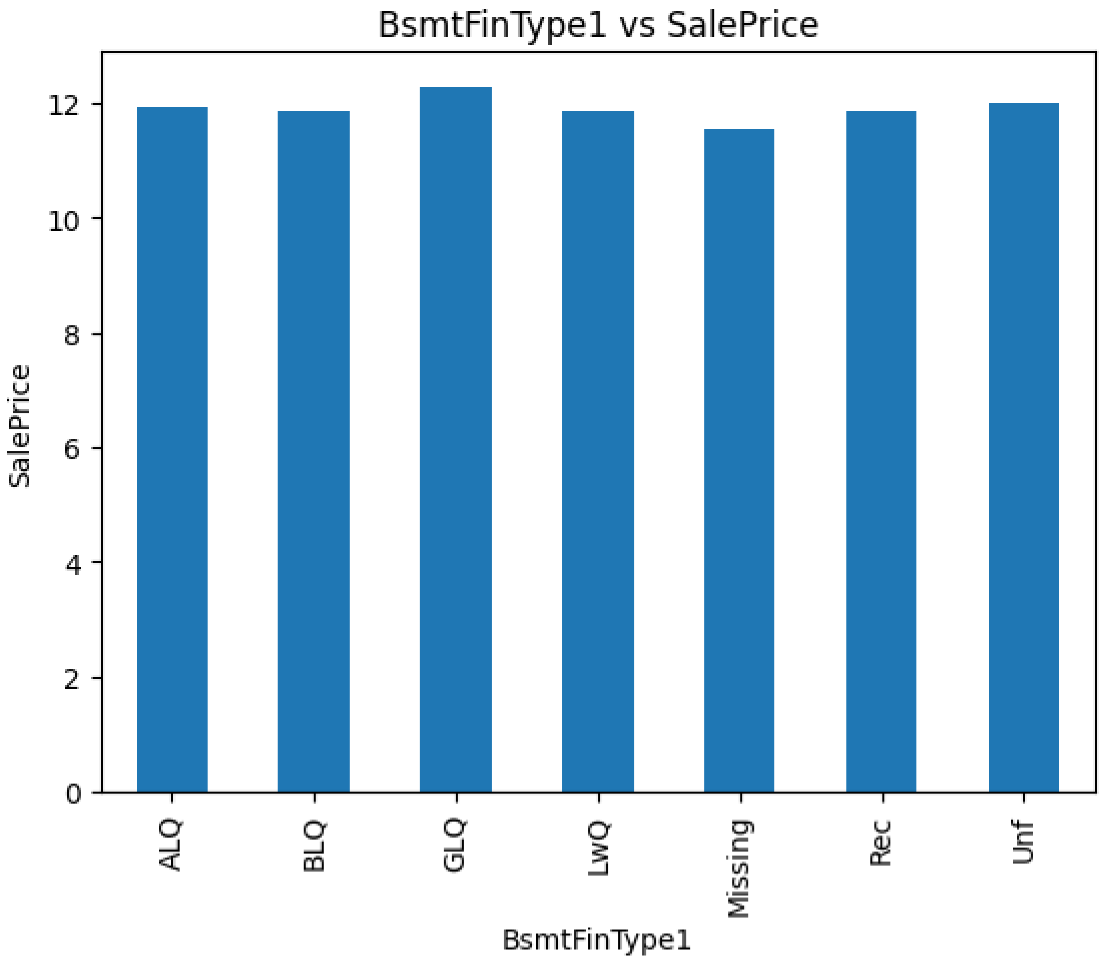

ExterCond vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Po
TA
ExterCond

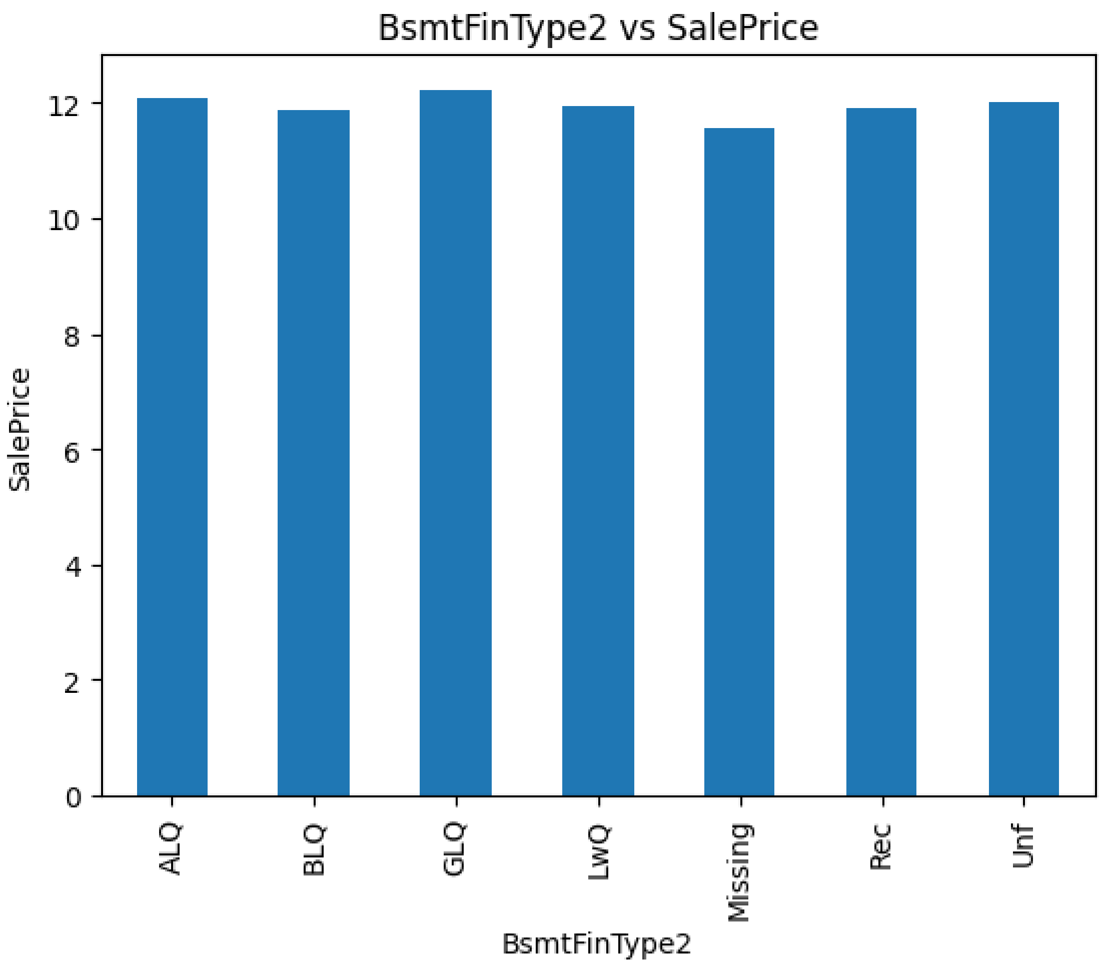

Foundation vs SalePrice
SalePrice
0
2
4
6
8
10
12
BrkTil
CBlock
PConc
Slab
Stone
Wood
Foundation

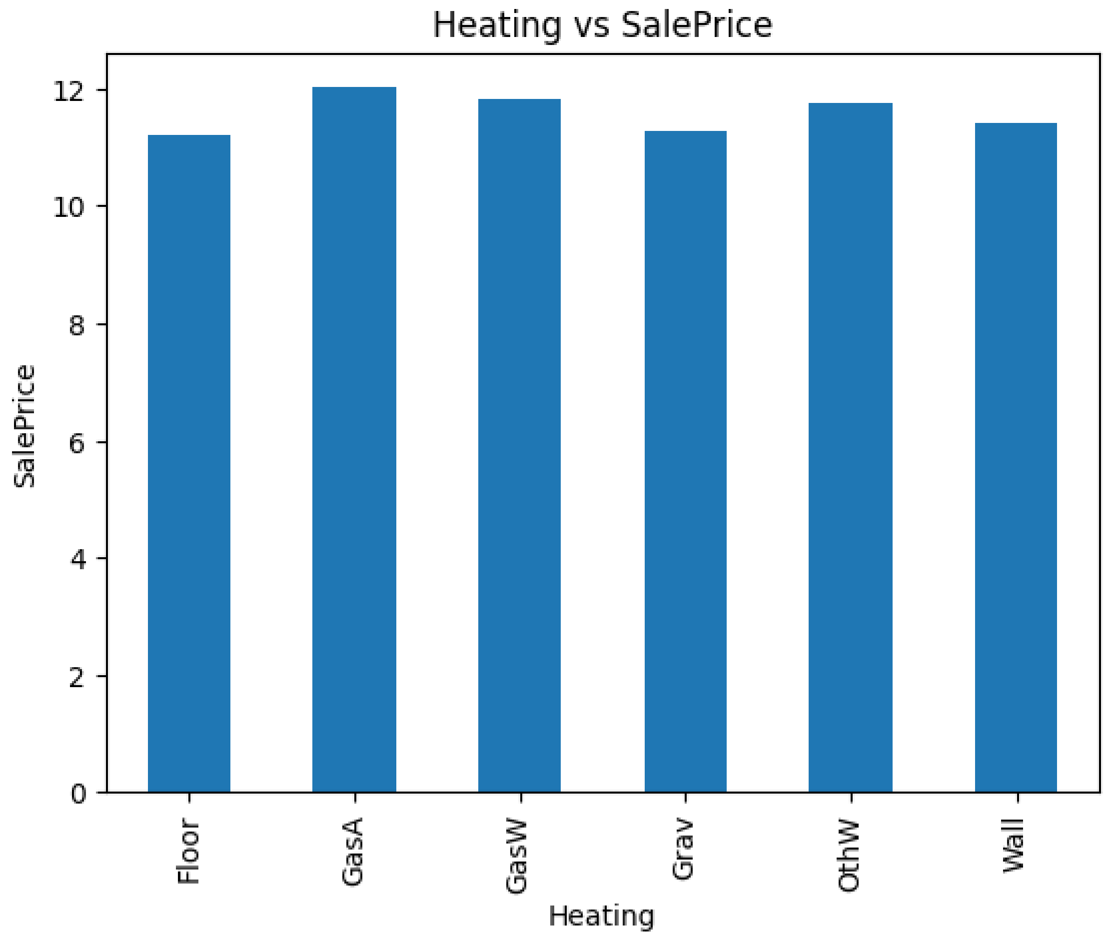

BsmtQual vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Missing
TA
BsmtQual

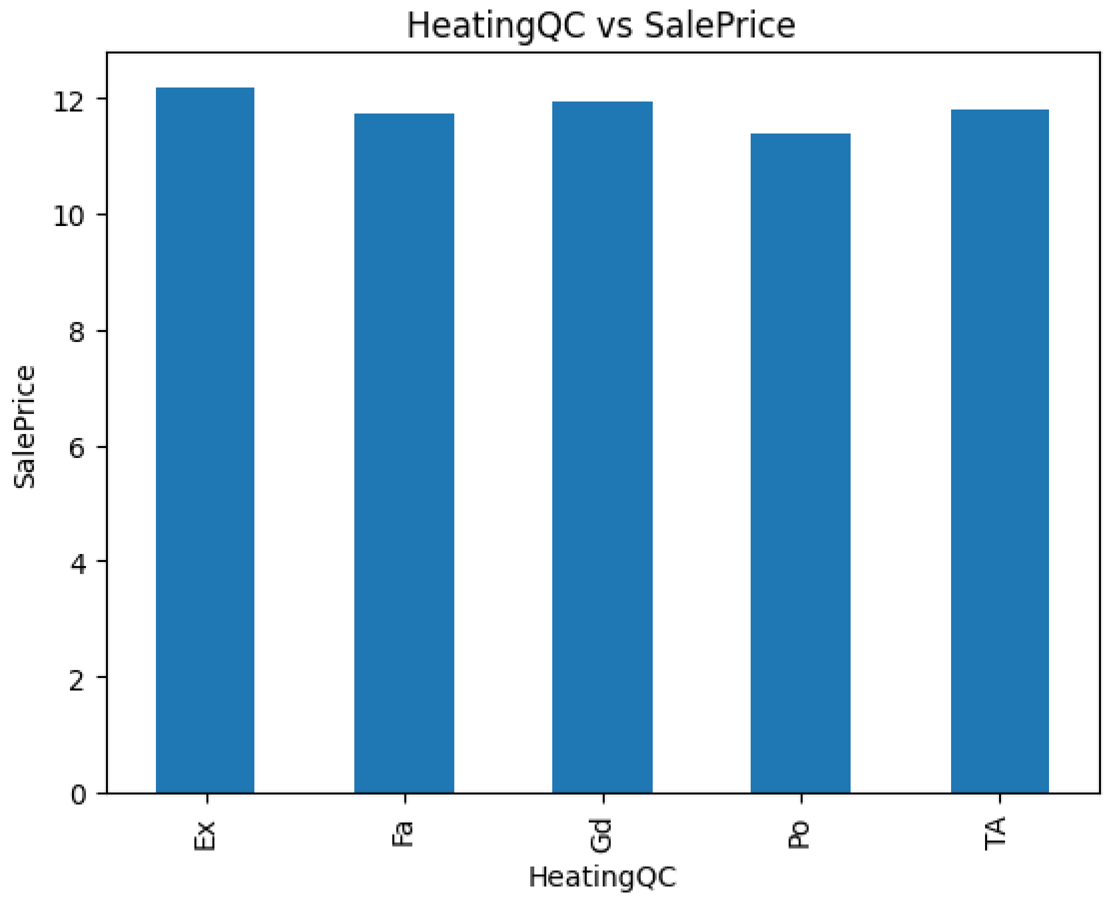

BsmtCond vs SalePrice
SalePrice
0
2
4
6
8
10
12
Fa
Gd
Missing
Po
TA
BsmtCond

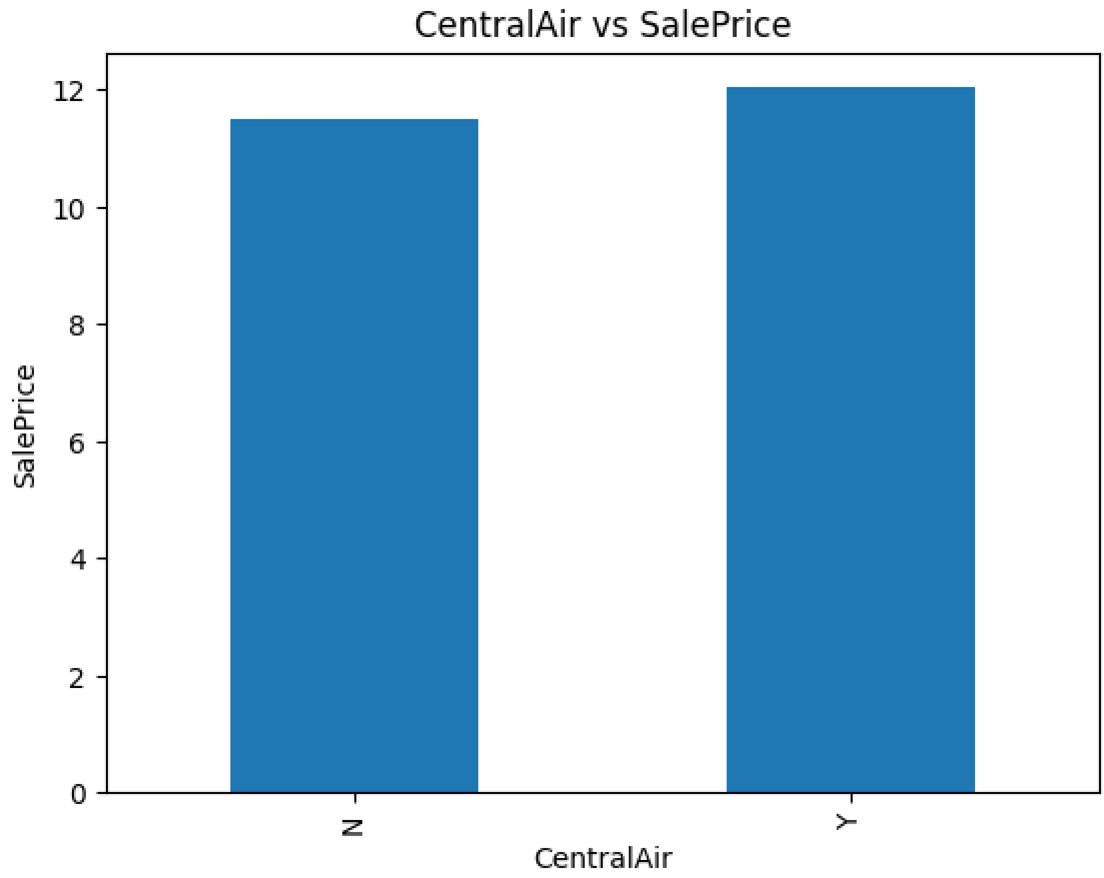

BsmtExposure vs SalePrice
SalePrice
0
2
4
6
8
10
12
Av
Gd
Missing
Mn
No
BsmtExposure

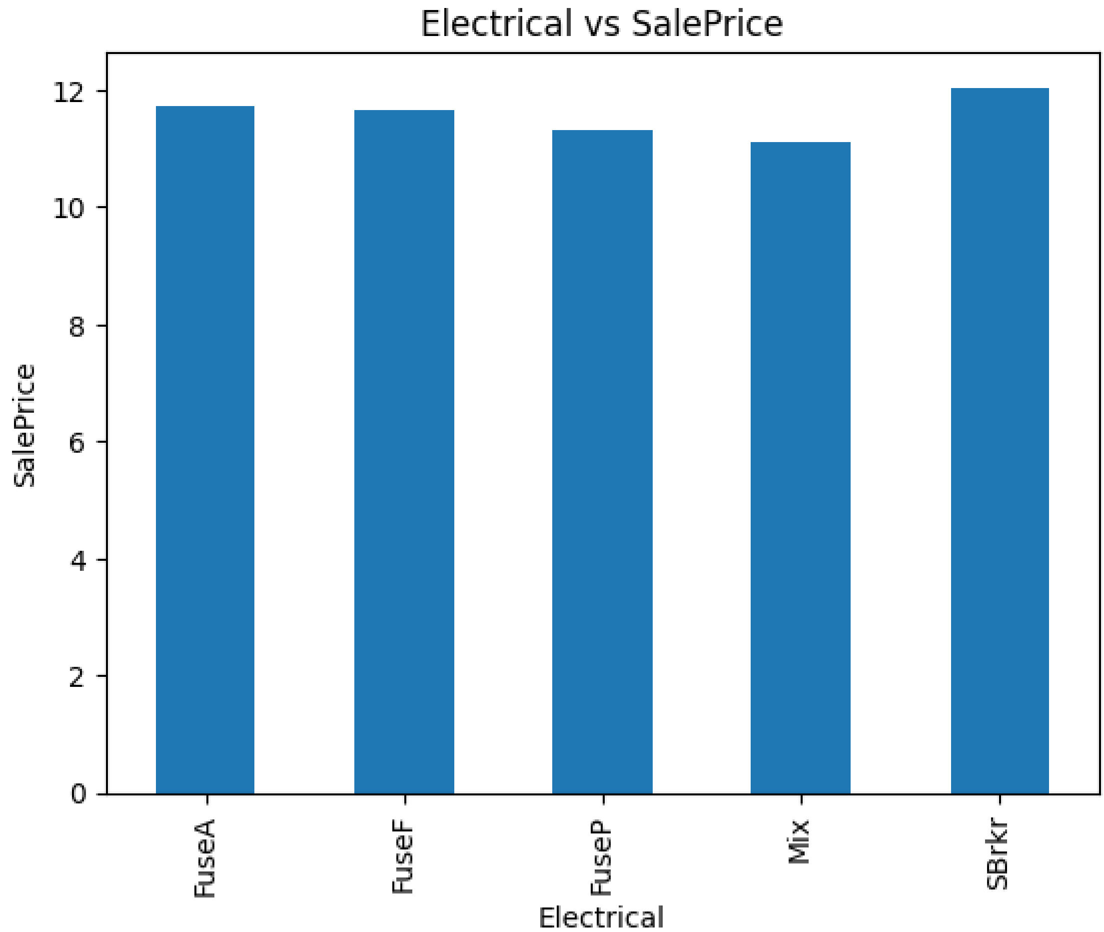

BsmtFinType1 vs SalePrice
SalePrice
0
2
4
6
8
10
12
ALQ
BLQ
GLQ
LwQ
Missing
Rec
Unf
BsmtFinType1

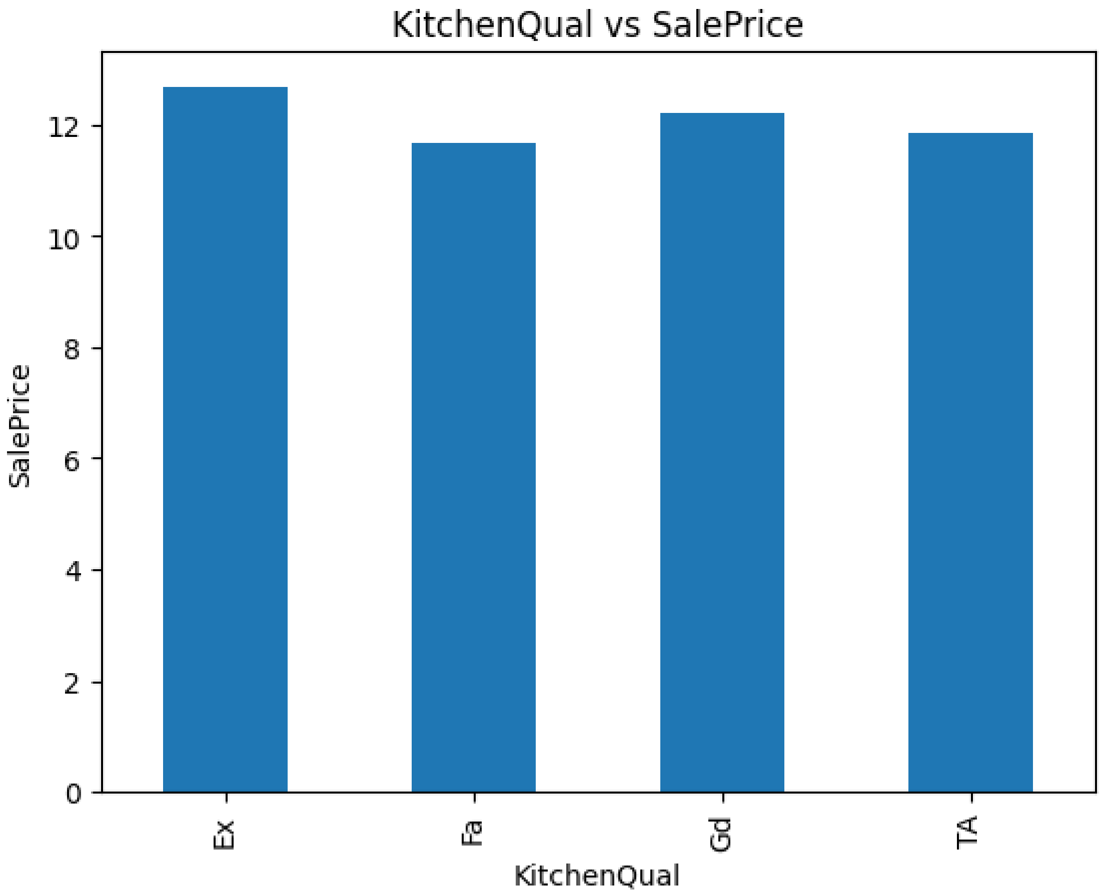

BsmtFinType2 vs SalePrice
SalePrice
0
2
4
6
8
10
12
ALQ
BLQ
GLQ
LwQ
Missing
Rec
Unf
BsmtFinType2

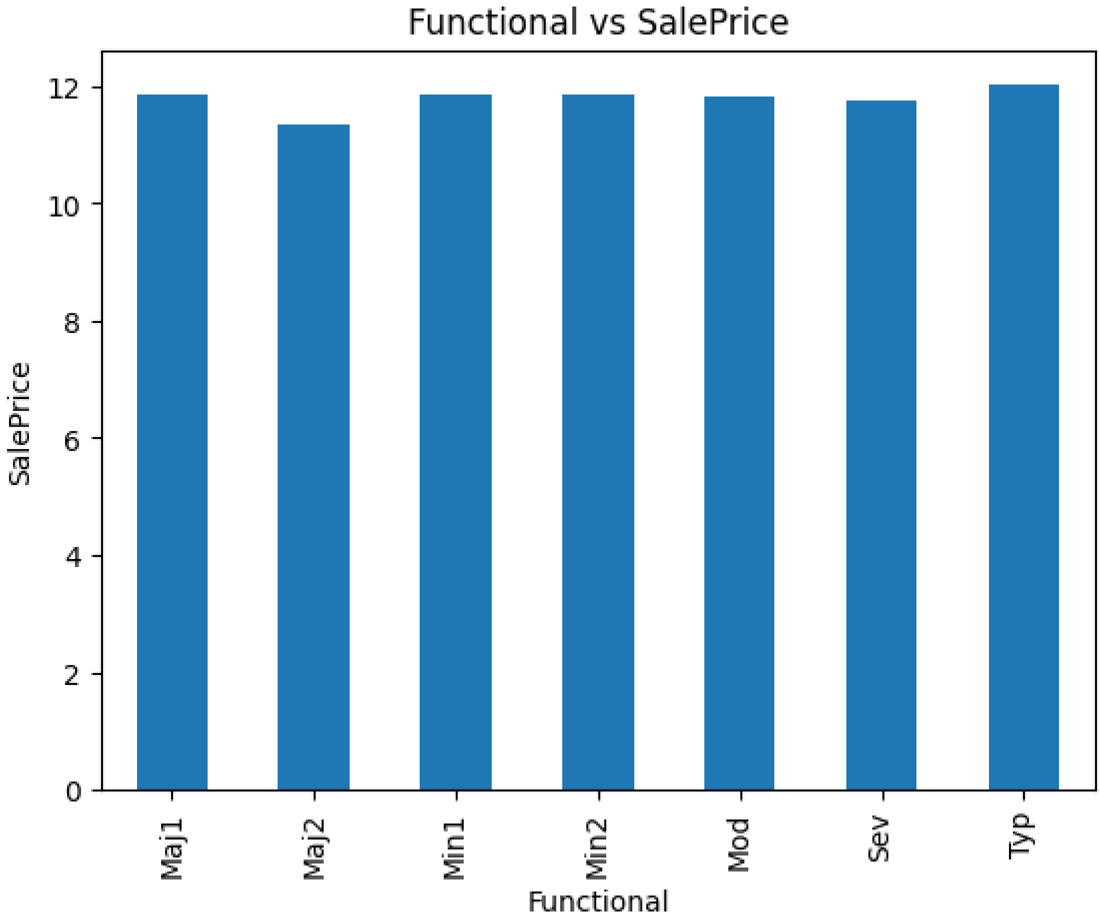

Heating vs SalePrice
SalePrice
0
2
4
6
8
10
12
Floor
GasA
GasW
Grav
OthW
Wall
Heating

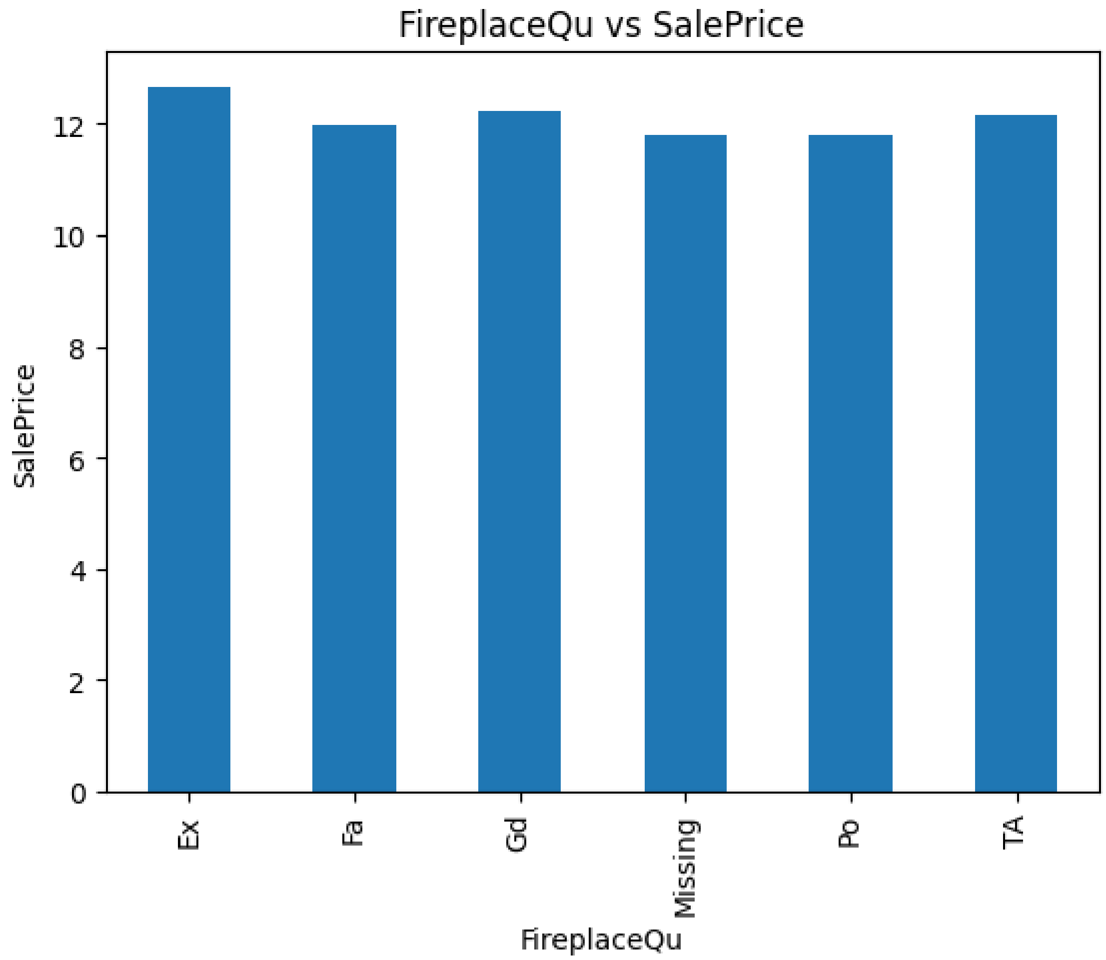

HeatingQC vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Po
TA
HeatingQC

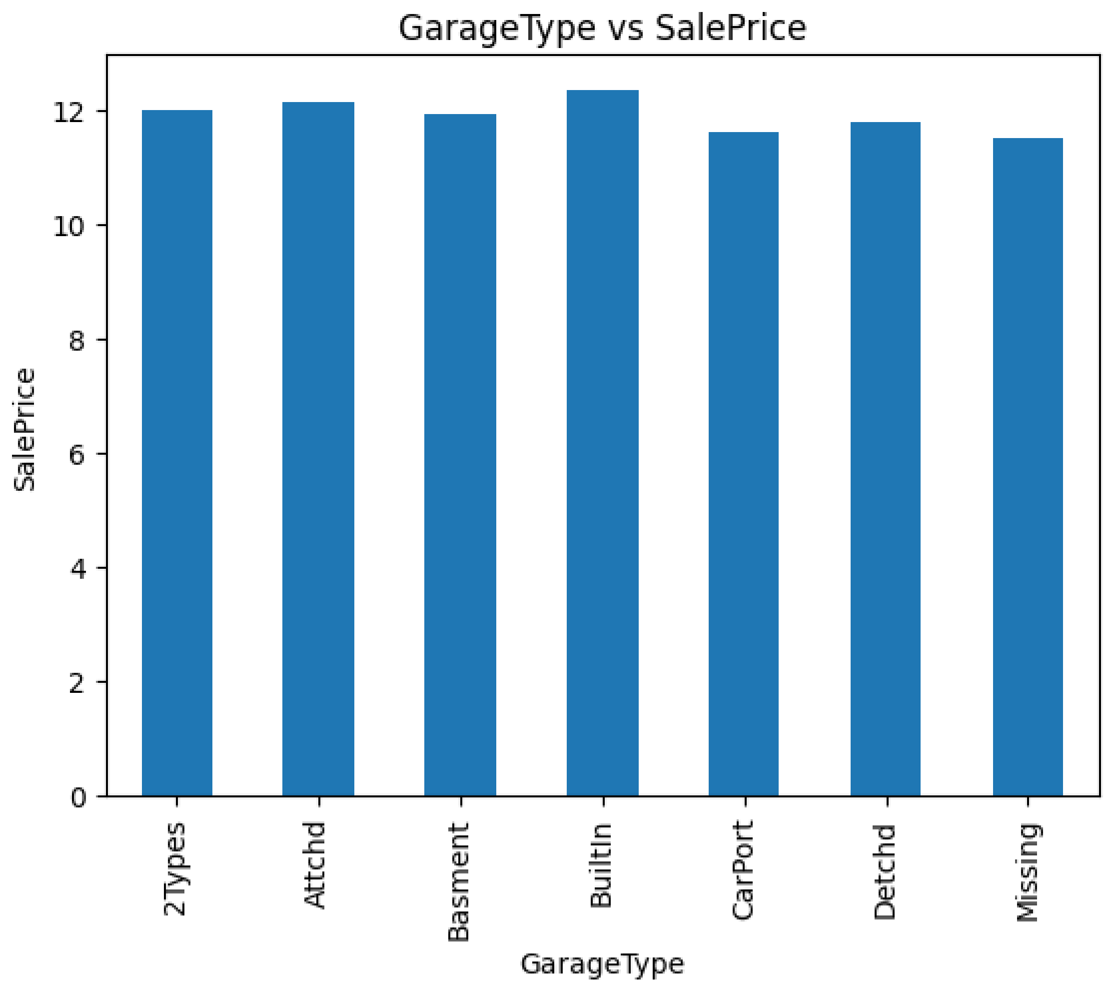

CentralAir vs SalePrice
SalePrice
0
2
4
6
8
10
12
N
Y
CentralAir

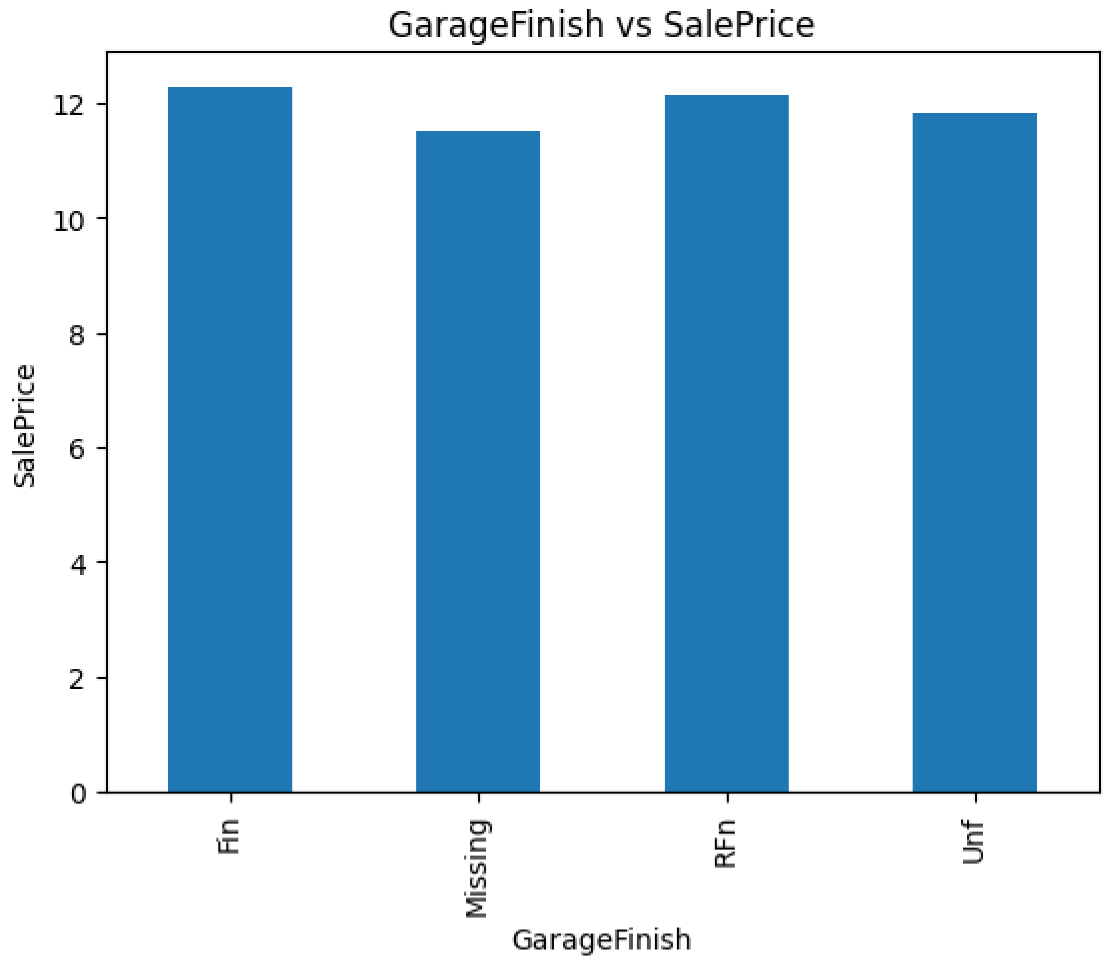

Electrical vs SalePrice
SalePrice
0
2
4
6
8
10
12
FuseA
FuseF
FuseP
Mix
SBrkr
Electrical

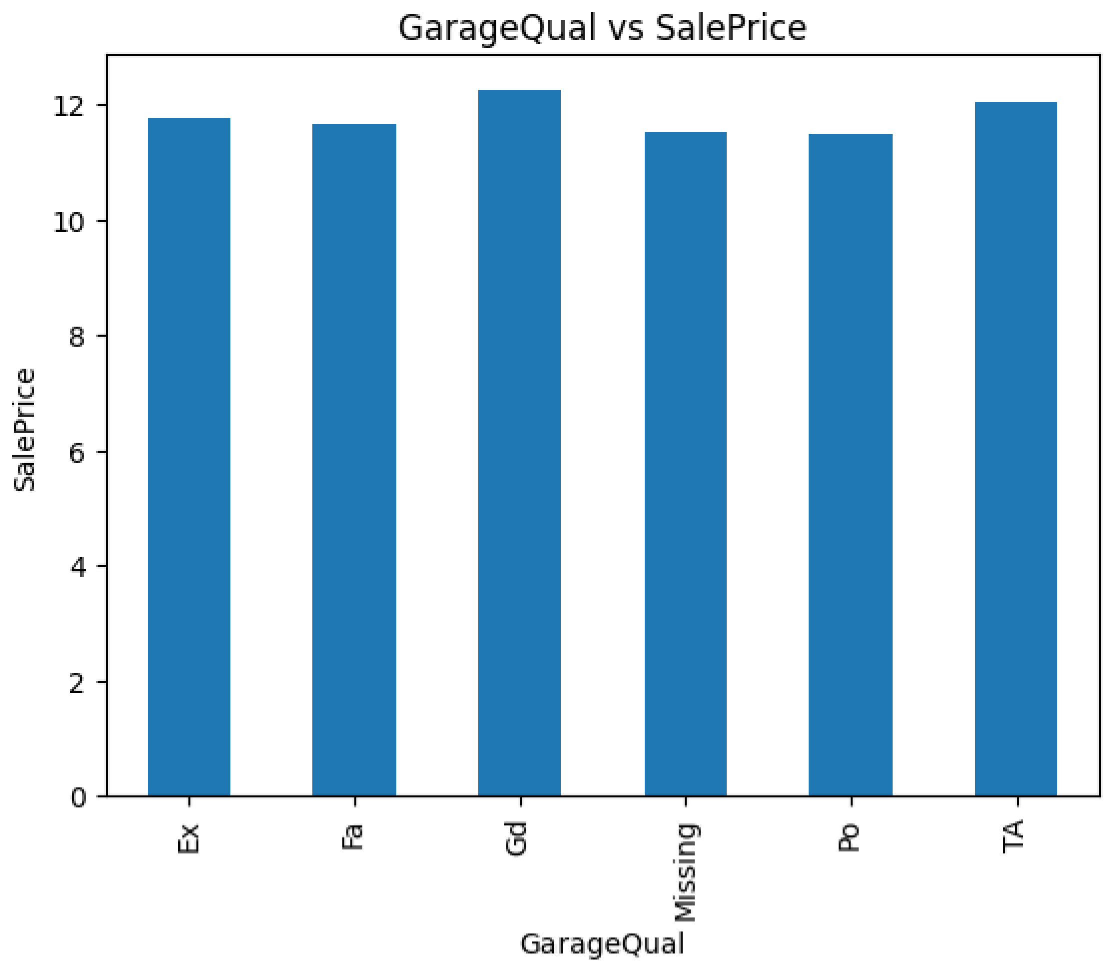

KitchenQual vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
TA
KitchenQual

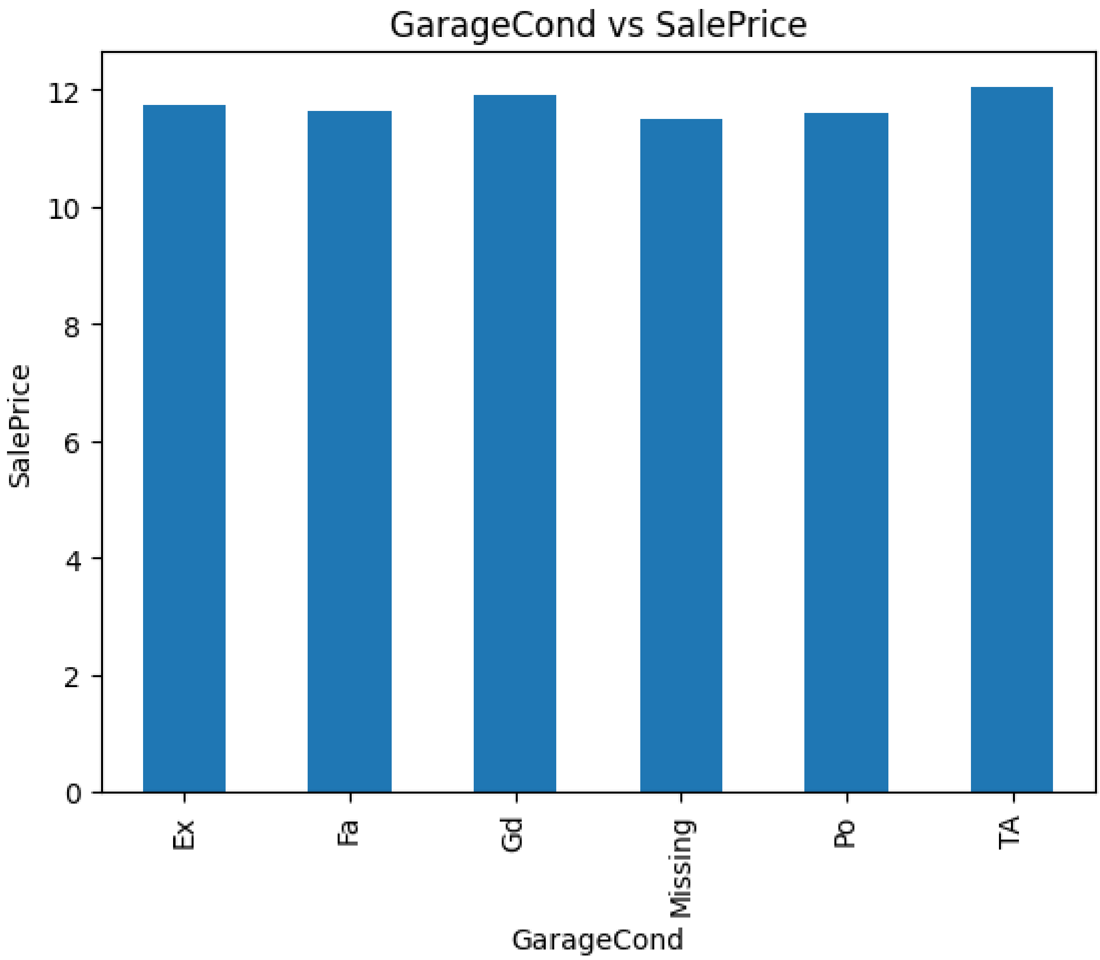

Functional vs SalePrice
SalePrice
0
2
4
6
8
10
12
Maj1
Maj2
Min1
Min2
Mod
Sev
Typ
Functional

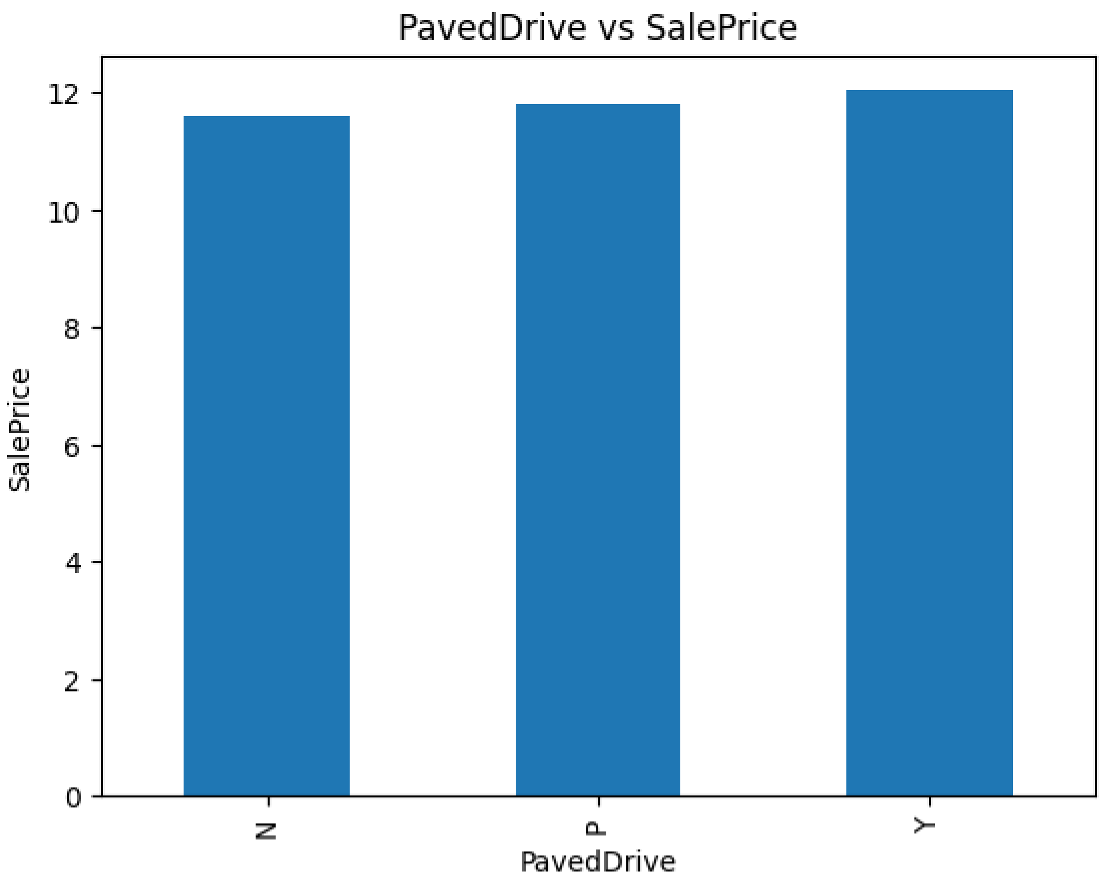

FireplaceQu vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Missing
Po
TA
FireplaceQu

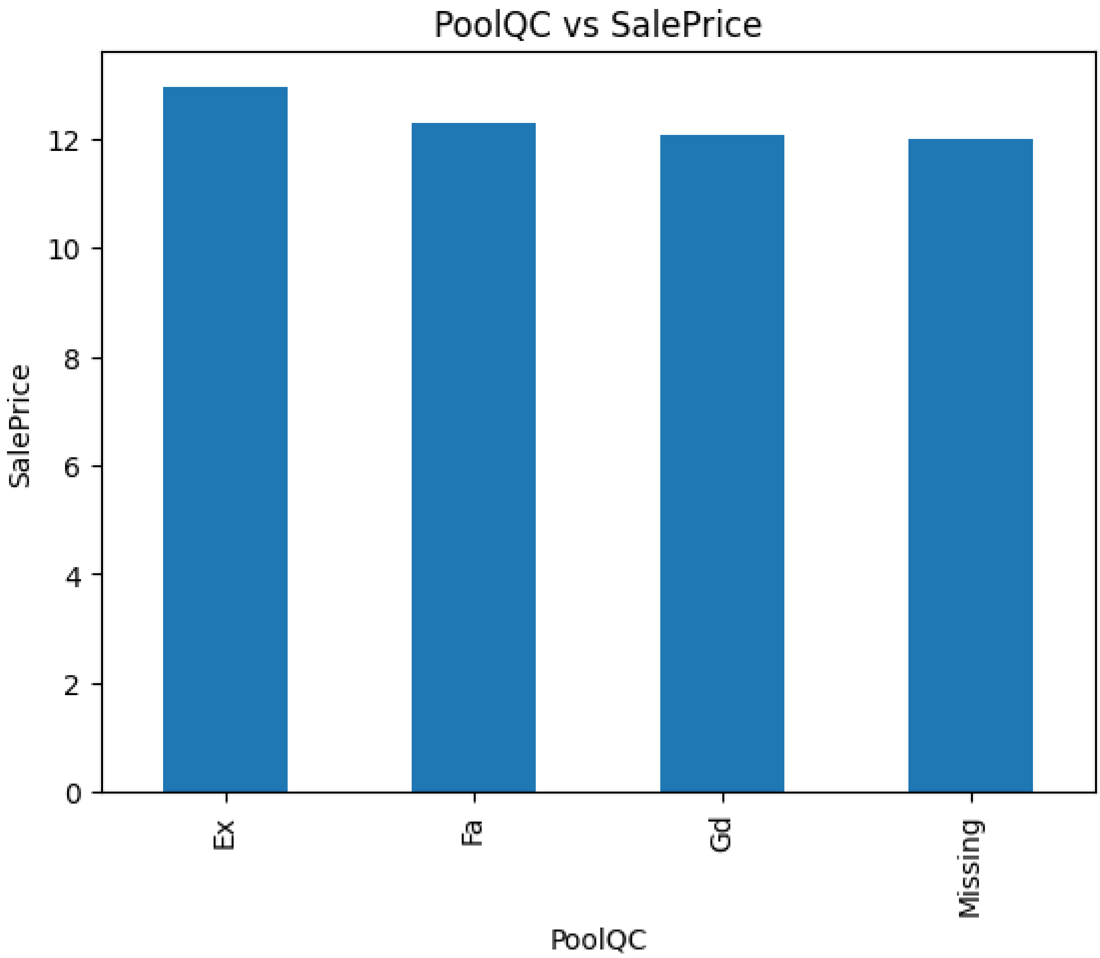

GarageType vs SalePrice
SalePrice
0
2
4
6
8
10
12
2Types
Attchd
Basment
BuiltIn
CarPort
Detchd
Missing
GarageType

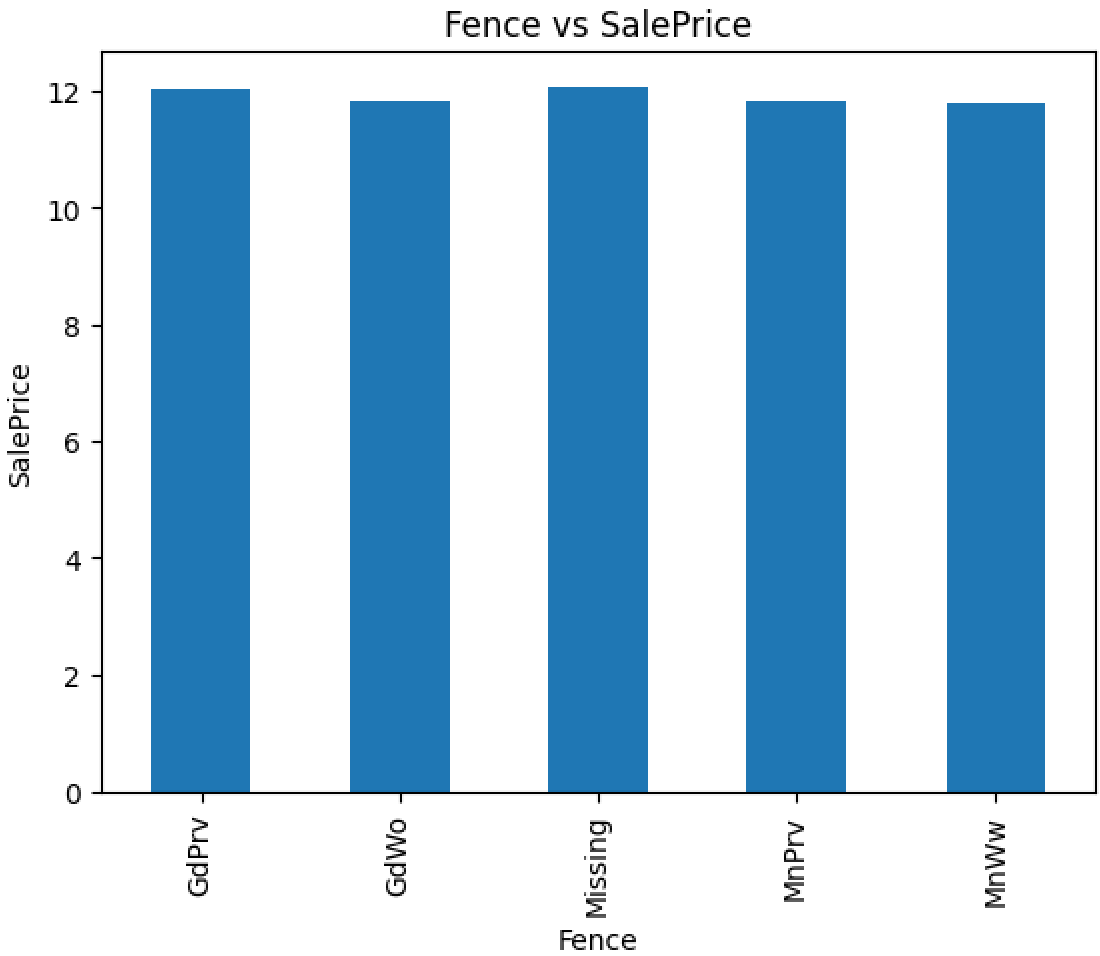

GarageFinish vs SalePrice
SalePrice
0
2
4
6
8
10
12
Fin
Missing
RFn
Unf
GarageFinish

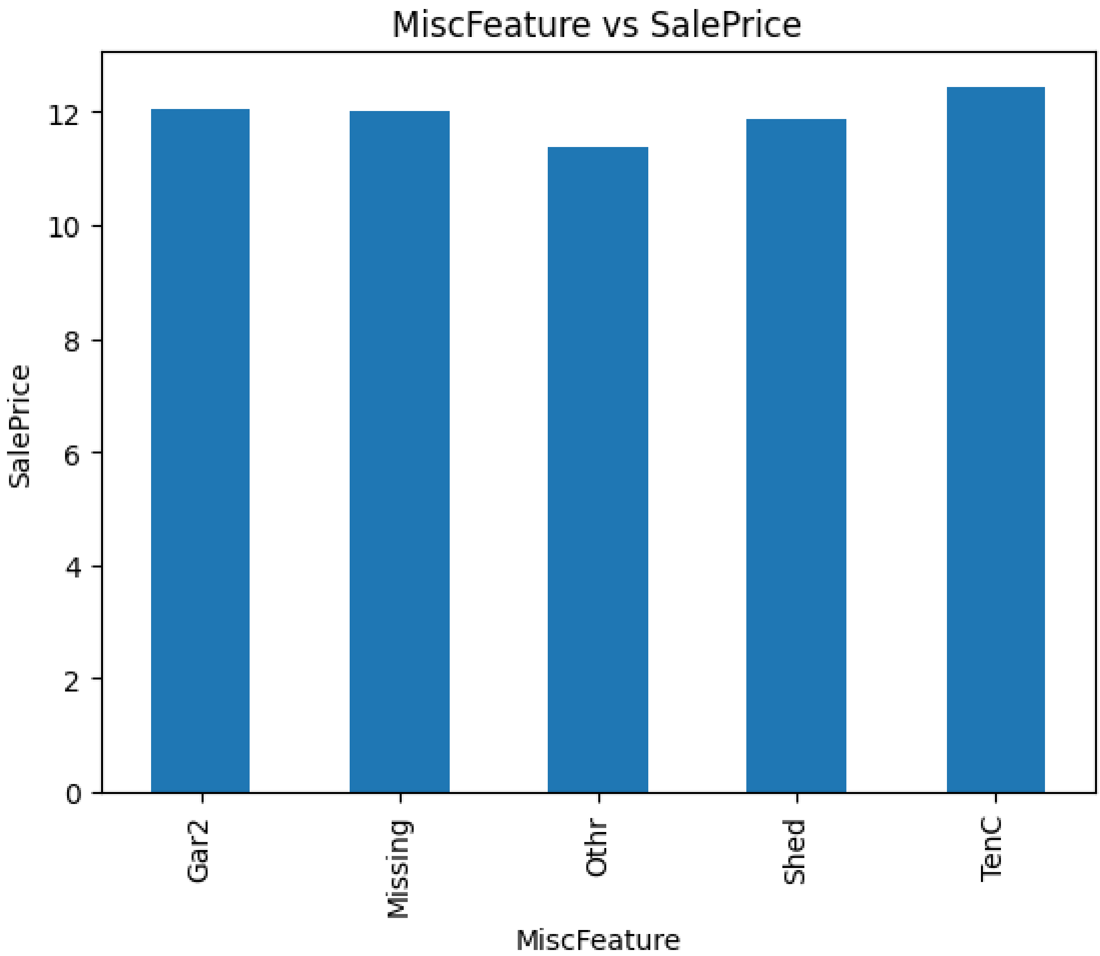

GarageQual vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Missing
Po
TA
GarageQual

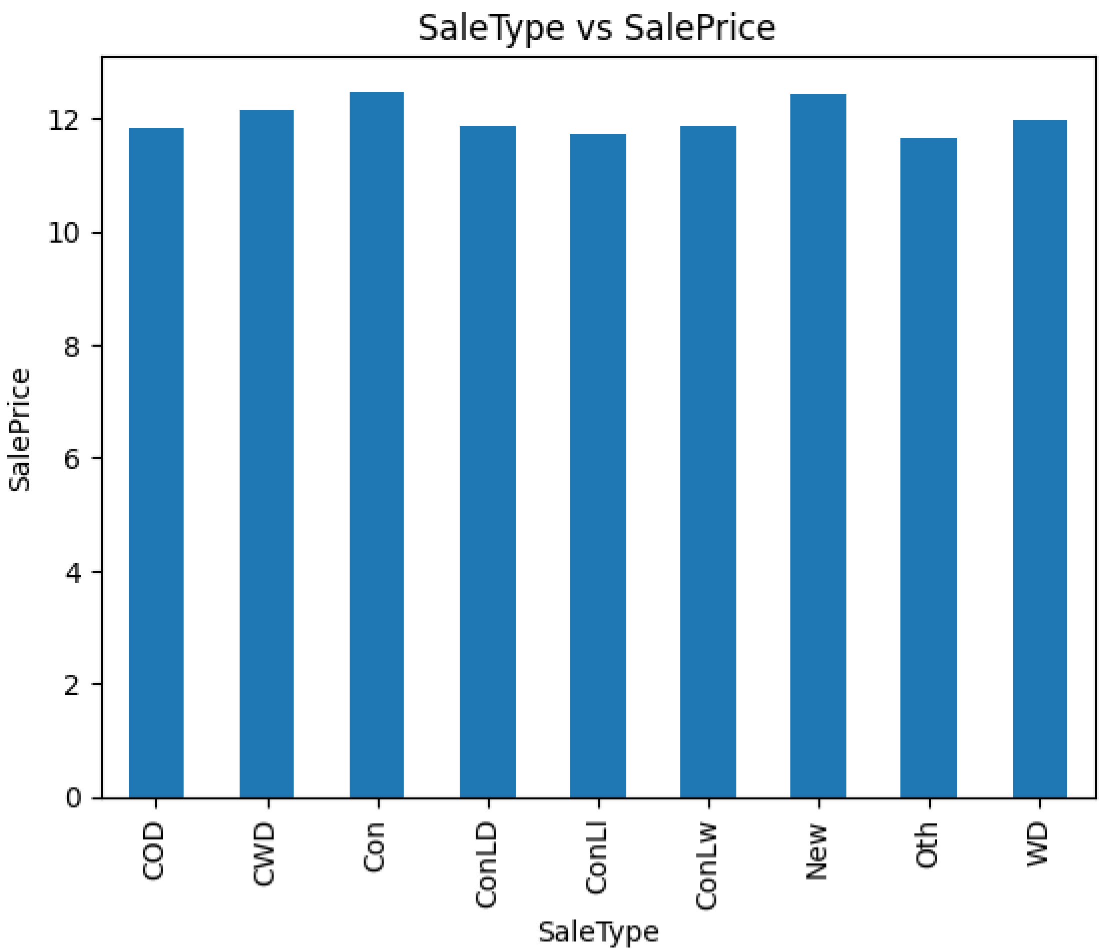

GarageCond vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Missing
Po
TA
GarageCond

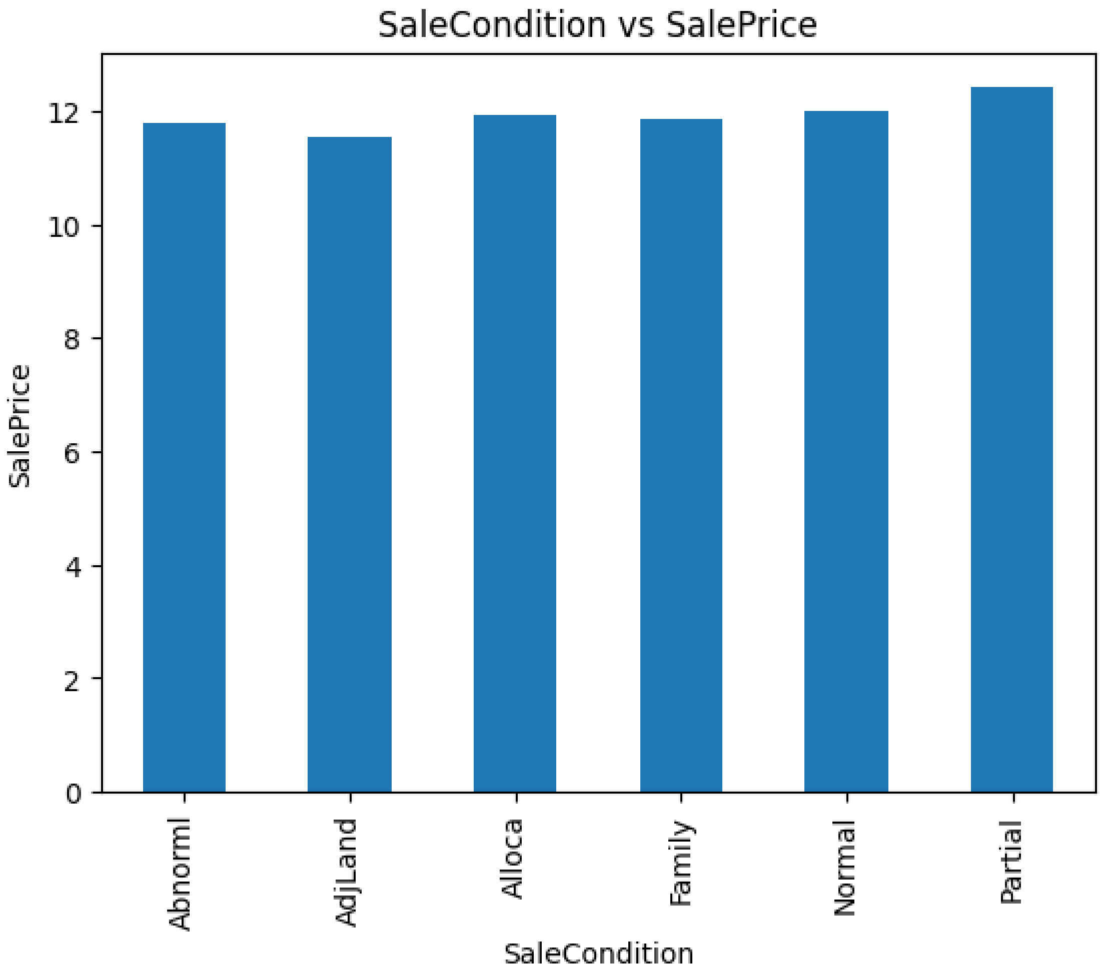

PavedDrive vs SalePrice
SalePrice
0
2
4
6
8
10
12
N
P
Y
PavedDrive


PoolQC vs SalePrice
SalePrice
0
2
4
6
8
10
12
Ex
Fa
Gd
Missing
PoolQC


Fence vs SalePrice
SalePrice
0
2
4
6
8
10
12
GdPrv
GdWo
Missing
MnPrv
MnWw
Fence


MiscFeature vs SalePrice
SalePrice
0
2
4
6
8
10
12
Gar2
Missing
Othr
Shed
TenC
MiscFeature


SaleType vs SalePrice
SalePrice
0
2
4
6
8
10
12
COD
CWD
Con
ConLD
ConLI
ConLw
New
Oth
WD
SaleType



RARE CATEGORICAL VARIABLES

We will replace categorical variables that are present less than 1% of the observations by 'Rare_var'.

In [443…

```
for feature in categorical_features:

  # group the dataset by each categorical feature and calculate the count of eac
  # divide the count by the length of the dataset to get the relative frequency
  temp = dataset.groupby(feature)['SalePrice'].count() / len(dataset)

  # create a new DataFrame (temp_df) by filtering the categories that have a rel
  # The .index attribute retrieves the index values of the filtered categories.
  temp_df = temp[temp > 0.01].index

  dataset[feature] = np.where(dataset[feature].isin(temp_df), dataset[feature],'
```

In [444…

```
dataset.head(100)
```

Out[444]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContc
0	1	60	RL	4.174387	9.041922	Pave	Missing	Reg	
1	2	20	RL	4.382027	9.169518	Pave	Missing	Reg	
2	3	60	RL	4.219508	9.328123	Pave	Missing	IR1	
3	4	70	RL	4.094345	9.164296	Pave	Missing	IR1	
4	5	60	RL	4.430817	9.565214	Pave	Missing	IR1	
...	...	...	...	...	...	...	...	...	
95	96	60	RL	4.234107	9.186560	Pave	Missing	IR2	
96	97	20	RL	4.356709	9.236398	Pave	Missing	IR1	
97	98	20	RL	4.290459	9.298443	Pave	Missing	Reg	H
98	99	30	RL	4.442651	9.270965	Pave	Missing	Reg	
99	100	20	RL	4.343805	9.139918	Pave	Missing	IR1	

100 rows × 84 columns

In [445…

```
for feature in categorical_features:
    labels_ordered = dataset.groupby([feature])['SalePrice'].mean().sort_values(
    labels_ordered = {k:i for i, k in enumerate(labels_ordered, 0)}
    dataset[feature] = dataset[feature].map(labels_ordered)
```

In [446…

```
dataset.head(10)
```

Out[446]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	3	4.174387	9.041922	1	2	0	1
1	2	20	3	4.382027	9.169518	1	2	0	1
2	3	60	3	4.219508	9.328123	1	2	1	1
3	4	70	3	4.094345	9.164296	1	2	1	1
4	5	60	3	4.430817	9.565214	1	2	1	1
5	6	50	3	4.442651	9.554993	1	2	1	1
6	7	20	3	4.317488	9.218705	1	2	0	1
7	8	60	3	4.234107	9.247829	1	2	1	1
8	9	50	1	3.931826	8.719317	1	2	0	1
9	10	190	3	3.912023	8.911934	1	2	0	1

FEATURE SCALING

(will not be performed on 'Id' column because it can be dropped later and 'SalePrice' column because it is the

dependennt variable.)

In [447...

```
feature_scale = []
for feature in dataset.columns:
  if feature not in ['Id','SalePerice']:
    feature_scale.append(feature)

print(len(feature_scale))
dataset[feature_scale].head()
```

```
83
```

Out[447]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utili
0	60	3	4.174387	9.041922	1	2	0	1	
1	20	3	4.382027	9.169518	1	2	0	1	
2	60	3	4.219508	9.328123	1	2	1	1	
3	70	3	4.094345	9.164296	1	2	1	1	
4	60	3	4.430817	9.565214	1	2	1	1	

In [448...

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(dataset[feature_scale])
scaler.transform(dataset[feature_scale])
```

Out[448]:

```
array([[0.23529412, 0.75      , 0.41820812, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.75      , 0.49506375, ..., 0.        , 0.        ,
        0.        ],
       [0.23529412, 0.75      , 0.434909  , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.29411765, 0.75      , 0.42385922, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.75      , 0.434909  , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.75      , 0.47117546, ..., 0.        , 0.        ,
        0.        ]])
```

In [449...

```
# transform the train and test set, and add on the Id and SalePrice variables
data = pd.concat([dataset[['Id', 'SalePrice']].reset_index(drop=True),
                    pd.DataFrame(scaler.transform(dataset[feature_scale]), colum
                    axis=1)
```

In [450...

```
data.head()
```

Out[450]:

	Id	SalePrice	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
0	1	12.247694	0.235294	0.75	0.418208	0.366344	1.0	1.0	0.000000	
1	2	12.109011	0.000000	0.75	0.495064	0.391317	1.0	1.0	0.000000	
2	3	12.317167	0.235294	0.75	0.434909	0.422359	1.0	1.0	0.333333	
3	4	11.849398	0.294118	0.75	0.388581	0.390295	1.0	1.0	0.333333	
4	5	12.429216	0.235294	0.75	0.513123	0.468761	1.0	1.0	0.333333	

In [451…

```
data.to_csv('X_train.csv', index = False)
```

FEATURE SELECTION - ADVANCE HOUSE PRICE PREDICTION

The main aim of this project is to predict the house price based on various features.

IMPORT LIBRARIES

In [452…

```
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
from sklearn.preprocessing import OneHotEncoder
```

READ DATA

In [453…

```
dataset_fs = pd.read_csv('X_train.csv')
```

In [454…

```
dataset_fs.head()
```

Out[454]:

	Id	SalePrice	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
0	1	12.247694	0.235294	0.75	0.418208	0.366344	1.0	1.0	0.000000	
1	2	12.109011	0.000000	0.75	0.495064	0.391317	1.0	1.0	0.000000	
2	3	12.317167	0.235294	0.75	0.434909	0.422359	1.0	1.0	0.333333	
3	4	11.849398	0.294118	0.75	0.388581	0.390295	1.0	1.0	0.333333	
4	5	12.429216	0.235294	0.75	0.513123	0.468761	1.0	1.0	0.333333	

In [455…

```
# Capture the dependent feature
y_train = dataset_fs[['SalePrice']]
```

In [456…

```
# drop dependent feature from dataset
X_train = dataset_fs.drop(['Id', 'SalePrice'], axis = 1)
```

In [457…

```
## Apply Feature Selection
# first, specify the Lasso Regression model, and
# select a suitable alpha (equivalent of penalty).
# The bigger the alpha the less features that will be selected.

# Then use the selectFromModel object from sklearn, which
# will select the features whose coefficients are non-zero

feature_sel_model = SelectFromModel(Lasso(alpha=0.005, random_state=0)) # rememb
feature_sel_model.fit(X_train, y_train)
```

Out[457]:

```
▸ SelectFromModel
▸ estimator: Lasso
    ▸ Lasso
```

In [458…

```
feature_sel_model.get_support()
```

Out[458]:

```
array([False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False])
```

In [459…

```
# print the number of total and selected features

# make a list of the selected features
selected_feat = X_train.columns[(feature_sel_model.get_support())]

# print stats
print('total features: {}'.format((X_train.shape[1])))
print('selected features: {}'.format(len(selected_feat)))
print('features with coefficients shrank to zero: {}'.format(
    np.sum(feature_sel_model.estimator_.coef_ == 0)))
```

```
total features: 83
selected features: 5
features with coefficients shrank to zero: 78
```

In [460…

```
selected_feat
```

Out[460]:

```
Index(['Neighborhood', 'YearRemodAdd', 'FireplaceQu', 'GarageFinish',
       'SalePrice.1'],
      dtype='object')
```

In [461…

```
X_train = X_train[selected_feat]
```

In [462…

```
X_train.head()
```

Out[462]:

	Neighborhood	YearRemodAdd	FireplaceQu	GarageFinish	SalePrice.1
0	0.636364	0.098361	0.2	0.666667	0.581431
1	0.500000	0.524590	0.6	0.666667	0.536319
2	0.636364	0.114754	0.6	0.666667	0.604029
3	0.727273	0.606557	0.8	0.333333	0.451871
4	1.000000	0.147541	0.6	0.666667	0.640477

In [462…