SalePricePrediction

June 12, 2024

1 Machine Learning Internship at Prodigy InfoTech - House Prediction Sale Price Project

In this project, our objective is to conduct an in-depth exploration and analysis of the House Prediction dataset. We use linear regression model to predict the prices of houses based on some relevent features that affect on House sale price .

2 Importing Important Libraries

[189]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import seaborn as sns
sns.set_style('darkgrid')
```

2.1 Load Train and Test Data

[190]:
```
train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")
```

2.1.1 Get information about Our data

[191]:
```
train.info()
print(50*"-")
test.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	588 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
--------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1459 non-null   int64
 1   MSSubClass     1459 non-null   int64
 2   MSZoning       1455 non-null   object
 3   LotFrontage    1232 non-null   float64
 4   LotArea        1459 non-null   int64
 5   Street         1459 non-null   object
 6   Alley          107 non-null    object
 7   LotShape       1459 non-null   object
 8   LandContour    1459 non-null   object
```

```
9   Utilities      1457 non-null   object
10  LotConfig      1459 non-null   object
11  LandSlope      1459 non-null   object
12  Neighborhood   1459 non-null   object
13  Condition1     1459 non-null   object
14  Condition2     1459 non-null   object
15  BldgType       1459 non-null   object
16  HouseStyle     1459 non-null   object
17  OverallQual    1459 non-null   int64
18  OverallCond    1459 non-null   int64
19  YearBuilt      1459 non-null   int64
20  YearRemodAdd   1459 non-null   int64
21  RoofStyle      1459 non-null   object
22  RoofMatl       1459 non-null   object
23  Exterior1st    1458 non-null   object
24  Exterior2nd    1458 non-null   object
25  MasVnrType     565 non-null    object
26  MasVnrArea     1444 non-null   float64
27  ExterQual      1459 non-null   object
28  ExterCond      1459 non-null   object
29  Foundation     1459 non-null   object
30  BsmtQual       1415 non-null   object
31  BsmtCond       1414 non-null   object
32  BsmtExposure   1415 non-null   object
33  BsmtFinType1   1417 non-null   object
34  BsmtFinSF1     1458 non-null   float64
35  BsmtFinType2   1417 non-null   object
36  BsmtFinSF2     1458 non-null   float64
37  BsmtUnfSF      1458 non-null   float64
38  TotalBsmtSF    1458 non-null   float64
39  Heating        1459 non-null   object
40  HeatingQC      1459 non-null   object
41  CentralAir     1459 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1459 non-null   int64
44  2ndFlrSF       1459 non-null   int64
45  LowQualFinSF   1459 non-null   int64
46  GrLivArea      1459 non-null   int64
47  BsmtFullBath   1457 non-null   float64
48  BsmtHalfBath   1457 non-null   float64
49  FullBath       1459 non-null   int64
50  HalfBath       1459 non-null   int64
51  BedroomAbvGr   1459 non-null   int64
52  KitchenAbvGr   1459 non-null   int64
53  KitchenQual    1458 non-null   object
54  TotRmsAbvGrd   1459 non-null   int64
55  Functional     1457 non-null   object
56  Fireplaces     1459 non-null   int64
```

```
 57  FireplaceQu    729 non-null    object
 58  GarageType     1383 non-null   object
 59  GarageYrBlt    1381 non-null   float64
 60  GarageFinish   1381 non-null   object
 61  GarageCars     1458 non-null   float64
 62  GarageArea     1458 non-null   float64
 63  GarageQual     1381 non-null   object
 64  GarageCond     1381 non-null   object
 65  PavedDrive     1459 non-null   object
 66  WoodDeckSF     1459 non-null   int64
 67  OpenPorchSF    1459 non-null   int64
 68  EnclosedPorch  1459 non-null   int64
 69  3SsnPorch      1459 non-null   int64
 70  ScreenPorch    1459 non-null   int64
 71  PoolArea       1459 non-null   int64
 72  PoolQC         3 non-null      object
 73  Fence          290 non-null    object
 74  MiscFeature    51 non-null     object
 75  MiscVal        1459 non-null   int64
 76  MoSold         1459 non-null   int64
 77  YrSold         1459 non-null   int64
 78  SaleType       1458 non-null   object
 79  SaleCondition  1459 non-null   object
dtypes: float64(11), int64(26), object(43)
memory usage: 912.0+ KB
```

[192]:
```
train.head(5)
```

[192]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
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

  LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal MoSold  \
0         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
1         Lvl    AllPub  …        0    NaN   NaN         NaN       0      5
2         Lvl    AllPub  …        0    NaN   NaN         NaN       0      9
3         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
4         Lvl    AllPub  …        0    NaN   NaN         NaN       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
```

```
4   2008         WD          Normal      250000

[5 rows x 81 columns]
```

[193]:
```python
test.head(5)
```

[193]:
```
     Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
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

  LandContour Utilities  … ScreenPorch PoolArea PoolQC  Fence MiscFeature  \
0         Lvl    AllPub  …         120        0    NaN  MnPrv         NaN
1         Lvl    AllPub  …           0        0    NaN    NaN        Gar2
2         Lvl    AllPub  …           0        0    NaN  MnPrv         NaN
3         Lvl    AllPub  …           0        0    NaN    NaN         NaN
4         HLS    AllPub  …         144        0    NaN    NaN         NaN

  MiscVal MoSold  YrSold  SaleType  SaleCondition
0       0      6    2010        WD         Normal
1   12500      6    2010        WD         Normal
2       0      3    2010        WD         Normal
3       0      6    2010        WD         Normal
4       0      1    2010        WD         Normal

[5 rows x 80 columns]
```

2.2 Get Statistical information about our Data

[194]:
```python
train.describe(include='all')
```

[194]:
```
                 Id   MSSubClass MSZoning  LotFrontage        LotArea Street  \
count   1460.000000  1460.000000     1460  1201.000000    1460.000000   1460
unique          NaN          NaN        5          NaN            NaN      2
top             NaN          NaN       RL          NaN            NaN   Pave
freq            NaN          NaN     1151          NaN            NaN   1454
mean     730.500000    56.897260      NaN    70.049958   10516.828082    NaN
std      421.610009    42.300571      NaN    24.284752    9981.264932    NaN
min        1.000000    20.000000      NaN    21.000000    1300.000000    NaN
25%      365.750000    20.000000      NaN    59.000000    7553.500000    NaN
50%      730.500000    50.000000      NaN    69.000000    9478.500000    NaN
75%     1095.250000    70.000000      NaN    80.000000   11601.500000    NaN
max     1460.000000   190.000000      NaN   313.000000  215245.000000    NaN

       Alley LotShape LandContour Utilities  …     PoolArea PoolQC  Fence  \
```

```
count       91      1460          1460      1460  ...  1460.000000      7    281
unique       2         4             4         2  ...          NaN      3      4
top       Grvl       Reg           Lvl    AllPub  ...          NaN     Gd  MnPrv
freq        50       925          1311      1459  ...          NaN      3    157
mean       NaN       NaN           NaN       NaN  ...     2.758904    NaN    NaN
std        NaN       NaN           NaN       NaN  ...    40.177307    NaN    NaN
min        NaN       NaN           NaN       NaN  ...     0.000000    NaN    NaN
25%        NaN       NaN           NaN       NaN  ...     0.000000    NaN    NaN
50%        NaN       NaN           NaN       NaN  ...     0.000000    NaN    NaN
75%        NaN       NaN           NaN       NaN  ...     0.000000    NaN    NaN
max        NaN       NaN           NaN       NaN  ...   738.000000    NaN    NaN

       MiscFeature       MiscVal       MoSold       YrSold  SaleType  \
count           54   1460.000000  1460.000000  1460.000000      1460
unique           4           NaN          NaN          NaN         9
top           Shed           NaN          NaN          NaN        WD
freq            49           NaN          NaN          NaN      1267
mean           NaN     43.489041     6.321918  2007.815753       NaN
std            NaN    496.123024     2.703626     1.328095       NaN
min            NaN      0.000000     1.000000  2006.000000       NaN
25%            NaN      0.000000     5.000000  2007.000000       NaN
50%            NaN      0.000000     6.000000  2008.000000       NaN
75%            NaN      0.000000     8.000000  2009.000000       NaN
max            NaN  15500.000000    12.000000  2010.000000       NaN

        SaleCondition      SalePrice
count            1460    1460.000000
unique              6            NaN
top            Normal            NaN
freq             1198            NaN
mean              NaN  180921.195890
std               NaN   79442.502883
min               NaN   34900.000000
25%               NaN  129975.000000
50%               NaN  163000.000000
75%               NaN  214000.000000
max               NaN  755000.000000

[11 rows x 81 columns]
```

[195]:
```
test.describe(include='all')
```

[195]:
```
                 Id   MSSubClass MSZoning  LotFrontage       LotArea Street  \
count   1459.000000  1459.000000     1455  1232.000000   1459.000000   1459
unique          NaN          NaN        5          NaN           NaN      2
top             NaN          NaN       RL          NaN           NaN   Pave
freq            NaN          NaN     1114          NaN           NaN   1453
```

```
mean    2190.000000   57.378341      NaN   68.580357   9819.161069    NaN
std      421.321334   42.746880      NaN   22.376841   4955.517327    NaN
min     1461.000000   20.000000      NaN   21.000000   1470.000000    NaN
25%     1825.500000   20.000000      NaN   58.000000   7391.000000    NaN
50%     2190.000000   50.000000      NaN   67.000000   9399.000000    NaN
75%     2554.500000   70.000000      NaN   80.000000  11517.500000    NaN
max     2919.000000  190.000000      NaN  200.000000  56600.000000    NaN

       Alley LotShape LandContour Utilities  …  ScreenPorch     PoolArea  \
count    107     1459        1459      1457  …  1459.000000  1459.000000
unique     2        4           4         1  …          NaN          NaN
top     Grvl      Reg         Lvl    AllPub  …          NaN          NaN
freq      70      934        1311      1457  …          NaN          NaN
mean     NaN      NaN         NaN       NaN  …    17.064428     1.744345
std      NaN      NaN         NaN       NaN  …    56.609763    30.491646
min      NaN      NaN         NaN       NaN  …     0.000000     0.000000
25%      NaN      NaN         NaN       NaN  …     0.000000     0.000000
50%      NaN      NaN         NaN       NaN  …     0.000000     0.000000
75%      NaN      NaN         NaN       NaN  …     0.000000     0.000000
max      NaN      NaN         NaN       NaN  …   576.000000   800.000000

       PoolQC  Fence MiscFeature       MiscVal       MoSold       YrSold  \
count       3    290          51   1459.000000  1459.000000  1459.000000
unique      2      4           3           NaN          NaN          NaN
top        Ex  MnPrv        Shed           NaN          NaN          NaN
freq        2    172          46           NaN          NaN          NaN
mean      NaN    NaN         NaN     58.167923     6.104181  2007.769705
std       NaN    NaN         NaN    630.806978     2.722432     1.301740
min       NaN    NaN         NaN      0.000000     1.000000  2006.000000
25%       NaN    NaN         NaN      0.000000     4.000000  2007.000000
50%       NaN    NaN         NaN      0.000000     6.000000  2008.000000
75%       NaN    NaN         NaN      0.000000     8.000000  2009.000000
max       NaN    NaN         NaN  17000.000000    12.000000  2010.000000

        SaleType  SaleCondition
count       1458           1459
unique         9              6
top           WD         Normal
freq        1258           1204
mean         NaN            NaN
std          NaN            NaN
min          NaN            NaN
25%          NaN            NaN
50%          NaN            NaN
75%          NaN            NaN
max          NaN            NaN
```

```
[11 rows x 80 columns]
```

3 Preprocessing:-

3.1 Checking duplicates Values

[196]:
```
train.duplicated().sum()
```

[196]:
```
0
```

[197]:
```
test.duplicated().sum()
```

[197]:
```
0
```

Nice, There is no duplicate values

[198]:
```
# Calculate Percentage of null values in training set
missing_percentage_train = (train.isnull().sum() / len(train)) * 100
print("Percentage of missing values for training data:")
print(missing_percentage_train.to_string())
```

```
Percentage of missing values for training data:
Id                0.000000
MSSubClass        0.000000
MSZoning          0.000000
LotFrontage      17.739726
LotArea           0.000000
Street            0.000000
Alley            93.767123
LotShape          0.000000
LandContour       0.000000
Utilities         0.000000
LotConfig         0.000000
LandSlope         0.000000
Neighborhood      0.000000
Condition1        0.000000
Condition2        0.000000
BldgType          0.000000
HouseStyle        0.000000
OverallQual       0.000000
OverallCond       0.000000
YearBuilt         0.000000
YearRemodAdd      0.000000
RoofStyle         0.000000
RoofMatl          0.000000
Exterior1st       0.000000
Exterior2nd       0.000000
```

```
MasVnrType       59.726027
MasVnrArea        0.547945
ExterQual         0.000000
ExterCond         0.000000
Foundation        0.000000
BsmtQual          2.534247
BsmtCond          2.534247
BsmtExposure      2.602740
BsmtFinType1      2.534247
BsmtFinSF1        0.000000
BsmtFinType2      2.602740
BsmtFinSF2        0.000000
BsmtUnfSF         0.000000
TotalBsmtSF       0.000000
Heating           0.000000
HeatingQC         0.000000
CentralAir        0.000000
Electrical        0.068493
1stFlrSF          0.000000
2ndFlrSF          0.000000
LowQualFinSF      0.000000
GrLivArea         0.000000
BsmtFullBath      0.000000
BsmtHalfBath      0.000000
FullBath          0.000000
HalfBath          0.000000
BedroomAbvGr      0.000000
KitchenAbvGr      0.000000
KitchenQual       0.000000
TotRmsAbvGrd      0.000000
Functional        0.000000
Fireplaces        0.000000
FireplaceQu      47.260274
GarageType        5.547945
GarageYrBlt       5.547945
GarageFinish      5.547945
GarageCars        0.000000
GarageArea        0.000000
GarageQual        5.547945
GarageCond        5.547945
PavedDrive        0.000000
WoodDeckSF        0.000000
OpenPorchSF       0.000000
EnclosedPorch     0.000000
3SsnPorch         0.000000
ScreenPorch       0.000000
PoolArea          0.000000
PoolQC           99.520548
```

```
Fence            80.753425
MiscFeature      96.301370
MiscVal           0.000000
MoSold            0.000000
YrSold            0.000000
SaleType          0.000000
SaleCondition     0.000000
SalePrice         0.000000
```

[199]:

```python
# Percentage of null values in testing set
missing_percentage_test = (test.isnull().sum() / len(test)) * 100
print("Percentage of missing values for testing data:")
print(missing_percentage_test.to_string())
```

```
Percentage of missing values for testing data:
Id                0.000000
MSSubClass        0.000000
MSZoning          0.274160
LotFrontage      15.558602
LotArea           0.000000
Street            0.000000
Alley            92.666210
LotShape          0.000000
LandContour       0.000000
Utilities         0.137080
LotConfig         0.000000
LandSlope         0.000000
Neighborhood      0.000000
Condition1        0.000000
Condition2        0.000000
BldgType          0.000000
HouseStyle        0.000000
OverallQual       0.000000
OverallCond       0.000000
YearBuilt         0.000000
YearRemodAdd      0.000000
RoofStyle         0.000000
RoofMatl          0.000000
Exterior1st       0.068540
Exterior2nd       0.068540
MasVnrType       61.274846
MasVnrArea        1.028101
ExterQual         0.000000
ExterCond         0.000000
Foundation        0.000000
BsmtQual          3.015764
BsmtCond          3.084304
BsmtExposure      3.015764
```

```
BsmtFinType1       2.878684
BsmtFinSF1         0.068540
BsmtFinType2       2.878684
BsmtFinSF2         0.068540
BsmtUnfSF          0.068540
TotalBsmtSF        0.068540
Heating            0.000000
HeatingQC          0.000000
CentralAir         0.000000
Electrical         0.000000
1stFlrSF           0.000000
2ndFlrSF           0.000000
LowQualFinSF       0.000000
GrLivArea          0.000000
BsmtFullBath       0.137080
BsmtHalfBath       0.137080
FullBath           0.000000
HalfBath           0.000000
BedroomAbvGr       0.000000
KitchenAbvGr       0.000000
KitchenQual        0.068540
TotRmsAbvGrd       0.000000
Functional         0.137080
Fireplaces         0.000000
FireplaceQu       50.034270
GarageType         5.209047
GarageYrBlt        5.346127
GarageFinish       5.346127
GarageCars         0.068540
GarageArea         0.068540
GarageQual         5.346127
GarageCond         5.346127
PavedDrive         0.000000
WoodDeckSF         0.000000
OpenPorchSF        0.000000
EnclosedPorch      0.000000
3SsnPorch          0.000000
ScreenPorch        0.000000
PoolArea           0.000000
PoolQC            99.794380
Fence             80.123372
MiscFeature       96.504455
MiscVal            0.000000
MoSold             0.000000
YrSold             0.000000
SaleType           0.068540
SaleCondition      0.000000
```

3.2 Drop Columns with high missing values percentage

we drop Columns that exceeds 50% of Missing Values and the other will be imputed with Mean and Mode

[200]:
```
train.drop(['Alley', 'MasVnrType', 'FireplaceQu', 'PoolQC', 'Fence',␣
 ↪'MiscFeature'], axis=1, inplace=True)
test.drop(['Alley', 'MasVnrType', 'FireplaceQu', 'PoolQC', 'Fence',␣
 ↪'MiscFeature'], axis=1, inplace=True)
```

3.3 Impute Missing Values

[201]:
```
# Identify numerical and categorical columns for training Columns
train_numerical_cols = train.select_dtypes(include=[np.number]).columns
train_categorical_cols = train.select_dtypes(include=[object]).columns

# Impute missing values for numerical columns with mean
for col in train_numerical_cols:
    train[col].fillna(train[col].mean(), inplace=True)


# Impute missing values for categorical columns with mode
for col in train_categorical_cols:
    train[col].fillna(train[col].mode()[0], inplace=True)


# Identify numerical and categorical columns for Testing  Columns
test_numerical_cols = test.select_dtypes(include=[np.number]).columns
test_categorical_cols = test.select_dtypes(include=[object]).columns

# Impute missing values for numerical columns with mean
for col in test_numerical_cols:
   test[col].fillna(test[col].mean(), inplace=True)


# Impute missing values for categorical columns with mode
for col in test_categorical_cols:
    test[col].fillna(test[col].mode()[0], inplace=True)
```

```
C:\Users\engyo\AppData\Local\Temp\ipykernel_1252\612869699.py:7: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
```

```
instead, to perform the operation inplace on the original object.


  train[col].fillna(train[col].mean(), inplace=True)
C:\Users\engyo\AppData\Local\Temp\ipykernel_1252\612869699.py:12: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  train[col].fillna(train[col].mode()[0], inplace=True)
C:\Users\engyo\AppData\Local\Temp\ipykernel_1252\612869699.py:21: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  test[col].fillna(test[col].mean(), inplace=True)
C:\Users\engyo\AppData\Local\Temp\ipykernel_1252\612869699.py:26: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  test[col].fillna(test[col].mode()[0], inplace=True)
```

3.4 Check Missing Values

[202]:
```python
# Display the percentage of missing values after imputation
missing_percentage_train_after = (train.isnull().sum() / len(train)) * 100
print("Percentage of missing values for training data after imputation:")
print(missing_percentage_train_after.sum())

missing_percentage_test_after = (test.isnull().sum() / len(test)) * 100
print("Percentage of missing values for testing data after imputation:")
print(missing_percentage_test_after.sum())
```

```
Percentage of missing values for training data after imputation:
0.0
Percentage of missing values for testing data after imputation:
0.0
```

3.5 Handling outliers from train data

[203]:
```python
def remove_outliers_IQR(original_data, train_numerical_cols, threshold=1.5):
    for col in train_numerical_cols:
        q1 = original_data[col].quantile(0.25)
        q3 = original_data[col].quantile(0.75)
        IQR = q3 - q1
        lower_bound = q1 - threshold * IQR
        upper_bound = q3 + threshold * IQR
        outliers_mask = (original_data[col] < lower_bound) |␣
 ↪(original_data[col] > upper_bound)
        original_data = original_data[~outliers_mask].reset_index(drop=True)
    return original_data

# Applying the function to remove outliers
new_train = remove_outliers_IQR(train, train_numerical_cols)

# Displaying the number of outliers removed from each numerical column
for col in train_numerical_cols:
    outliers_removed = len(train[col]) - len(new_train[col])
    print(f"Number of outliers removed in {col}: {outliers_removed}")
```

```
Number of outliers removed in Id: 912
Number of outliers removed in MSSubClass: 912
Number of outliers removed in LotFrontage: 912
Number of outliers removed in LotArea: 912
Number of outliers removed in OverallQual: 912
Number of outliers removed in OverallCond: 912
Number of outliers removed in YearBuilt: 912
Number of outliers removed in YearRemodAdd: 912
Number of outliers removed in MasVnrArea: 912
Number of outliers removed in BsmtFinSF1: 912
```

```
Number of outliers removed in BsmtFinSF2: 912
Number of outliers removed in BsmtUnfSF: 912
Number of outliers removed in TotalBsmtSF: 912
Number of outliers removed in 1stFlrSF: 912
Number of outliers removed in 2ndFlrSF: 912
Number of outliers removed in LowQualFinSF: 912
Number of outliers removed in GrLivArea: 912
Number of outliers removed in BsmtFullBath: 912
Number of outliers removed in BsmtHalfBath: 912
Number of outliers removed in FullBath: 912
Number of outliers removed in HalfBath: 912
Number of outliers removed in BedroomAbvGr: 912
Number of outliers removed in KitchenAbvGr: 912
Number of outliers removed in TotRmsAbvGrd: 912
Number of outliers removed in Fireplaces: 912
Number of outliers removed in GarageYrBlt: 912
Number of outliers removed in GarageCars: 912
Number of outliers removed in GarageArea: 912
Number of outliers removed in WoodDeckSF: 912
Number of outliers removed in OpenPorchSF: 912
Number of outliers removed in EnclosedPorch: 912
Number of outliers removed in 3SsnPorch: 912
Number of outliers removed in ScreenPorch: 912
Number of outliers removed in PoolArea: 912
Number of outliers removed in MiscVal: 912
Number of outliers removed in MoSold: 912
Number of outliers removed in YrSold: 912
Number of outliers removed in SalePrice: 912
```

Here, we are using a function designed to detect and eliminate outliers. This function accepts three parameters: the dataset, the selected numerical columns, and an optional threshold value (default set to 1.5).

For each numerical column, the function computes the first quartile (q1), third quartile (q3), and interquartile range (IQR). Then, it establishes lower and upper bounds based on the IQR and the provided threshold. Using these bounds, a mask (outliers_mask) is created to identify rows containing outlier values.

The dataset is then updated by removing these outlier rows, and the index is reset using "reset_index" to ensure that the outlier indices are dropped. Finally, the updated dataset is returned.

At the end, the number of removed rows is printed.

3.6 Get Statistical information after Removing Outliers

[204]:
```
new_train.describe(include='number')
```

[204]:
```
                 Id  MSSubClass  LotFrontage       LotArea  OverallQual  \
count    548.000000  548.000000   548.000000    548.000000   548.000000
```

```
mean    738.819343   49.105839   68.576908   9100.343066     6.231752
std     418.421232   31.660674   13.391548   2593.927851     1.235939
min       1.000000   20.000000   30.000000   2887.000000     2.000000
25%     382.750000   20.000000   60.750000   7683.000000     5.000000
50%     760.000000   50.000000   70.049958   9000.000000     6.000000
75%    1101.500000   60.000000   75.000000  10676.750000     7.000000
max    1456.000000  120.000000  109.000000  16770.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd  MasVnrArea   BsmtFinSF1  …  \
count   548.000000   548.000000    548.000000  548.000000   548.000000  …
mean      5.368613  1983.169708   1989.204380   75.369391   429.483577  …
std       0.707159    25.252264     19.607907  106.053404   401.742272  …
min       4.000000  1910.000000   1950.000000    0.000000     0.000000  …
25%       5.000000  1965.000000   1972.000000    0.000000     0.000000  …
50%       5.000000  1995.500000   1999.000000    0.000000   422.500000  …
75%       6.000000  2004.000000   2005.000000  143.000000   725.000000  …
max       7.000000  2009.000000   2010.000000  420.000000  1619.000000  …

       WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  \
count  548.000000   548.000000          548.0      548.0        548.0
mean    90.144161    40.246350            0.0        0.0          0.0
std     97.322300    43.943285            0.0        0.0          0.0
min      0.000000     0.000000            0.0        0.0          0.0
25%      0.000000     0.000000            0.0        0.0          0.0
50%     99.000000    33.000000            0.0        0.0          0.0
75%    165.000000    63.000000            0.0        0.0          0.0
max    379.000000   162.000000            0.0        0.0          0.0

       PoolArea  MiscVal      MoSold       YrSold      SalePrice
count     548.0    548.0  548.000000   548.000000     548.000000
mean        0.0      0.0    6.324818  2007.784672  177890.208029
std         0.0      0.0    2.658566     1.322420   54472.843763
min         0.0      0.0    1.000000  2006.000000   37900.000000
25%         0.0      0.0    5.000000  2007.000000  136975.000000
50%         0.0      0.0    6.000000  2008.000000  175700.000000
75%         0.0      0.0    8.000000  2009.000000  212225.000000
max         0.0      0.0   12.000000  2010.000000  328900.000000

[8 rows x 38 columns]
```

[205]:
```
new_train.describe(include='object')
```

[205]:
```
       MSZoning Street LotShape LandContour Utilities LotConfig LandSlope  \
count       548    548      548         548       548       548       548
unique        4      2        4           4         1         5         2
top          RL   Pave      Reg         Lvl    AllPub    Inside       Gtl
freq        474    547      326         509       548       412       534
```

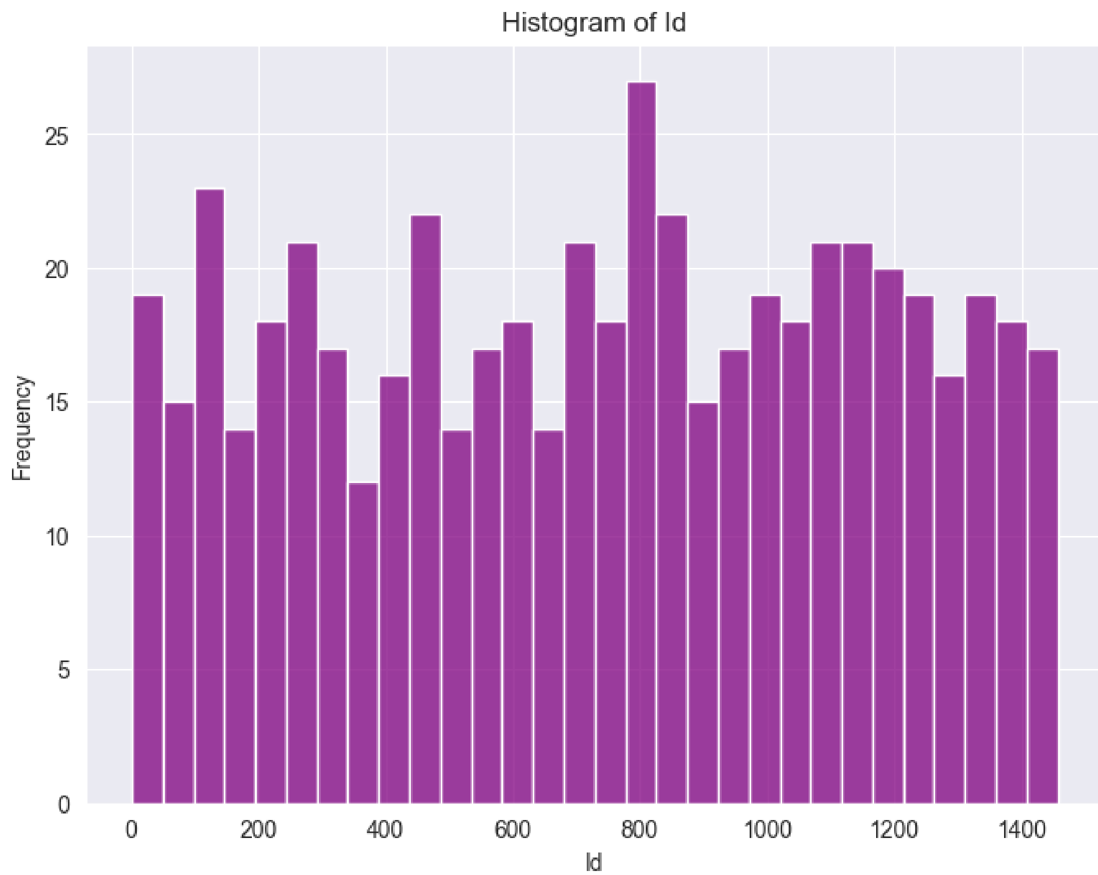
```
       Neighborhood Condition1 Condition2  … Electrical KitchenQual  \
count           548        548        548  …        548         548
unique           23          8          1  …          4           4
top         CollgCr       Norm       Norm  …      SBrkr          Gd
freq            112        491        548  …        519         284

       Functional GarageType GarageFinish GarageQual GarageCond PavedDrive  \
count         548        548          548        548        548        548
unique          5          4            3          3          4          3
top           Typ     Attchd          RFn         TA         TA          Y
freq          534        406          200        533        539        529

       SaleType SaleCondition
count       548           548
unique        9             5
top          WD        Normal
freq        460           443

[4 rows x 37 columns]
```

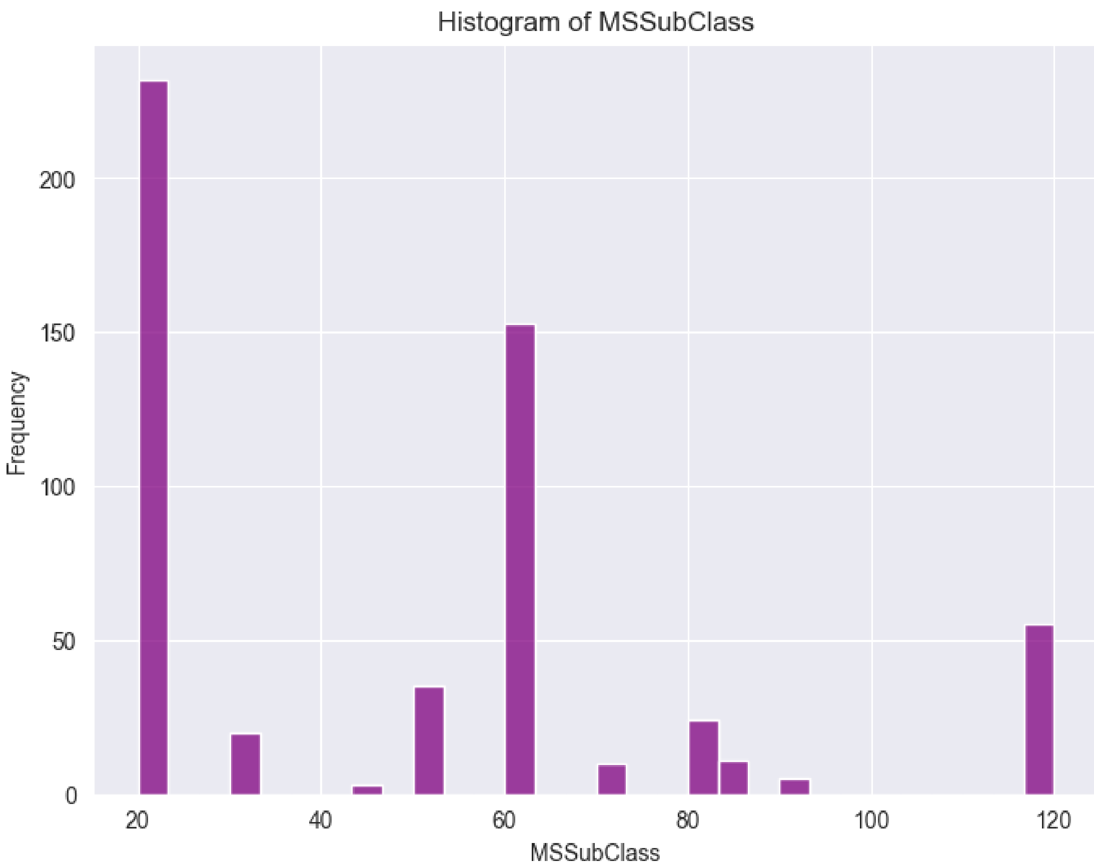
4 Visualization and Analysis:-

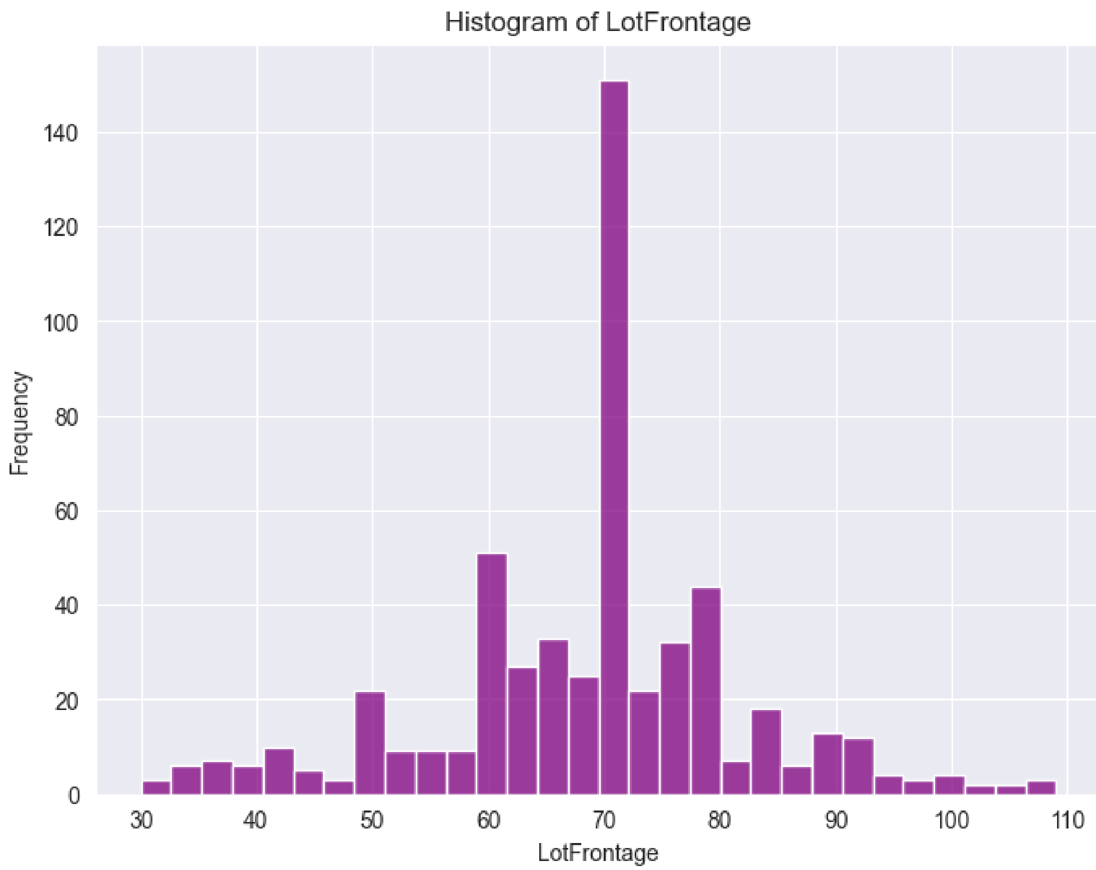
5 Distributions of Numerical columns

In this step, we plot histograms for various numerical columns within our dataset to gain a deeper understanding of the data distribution.

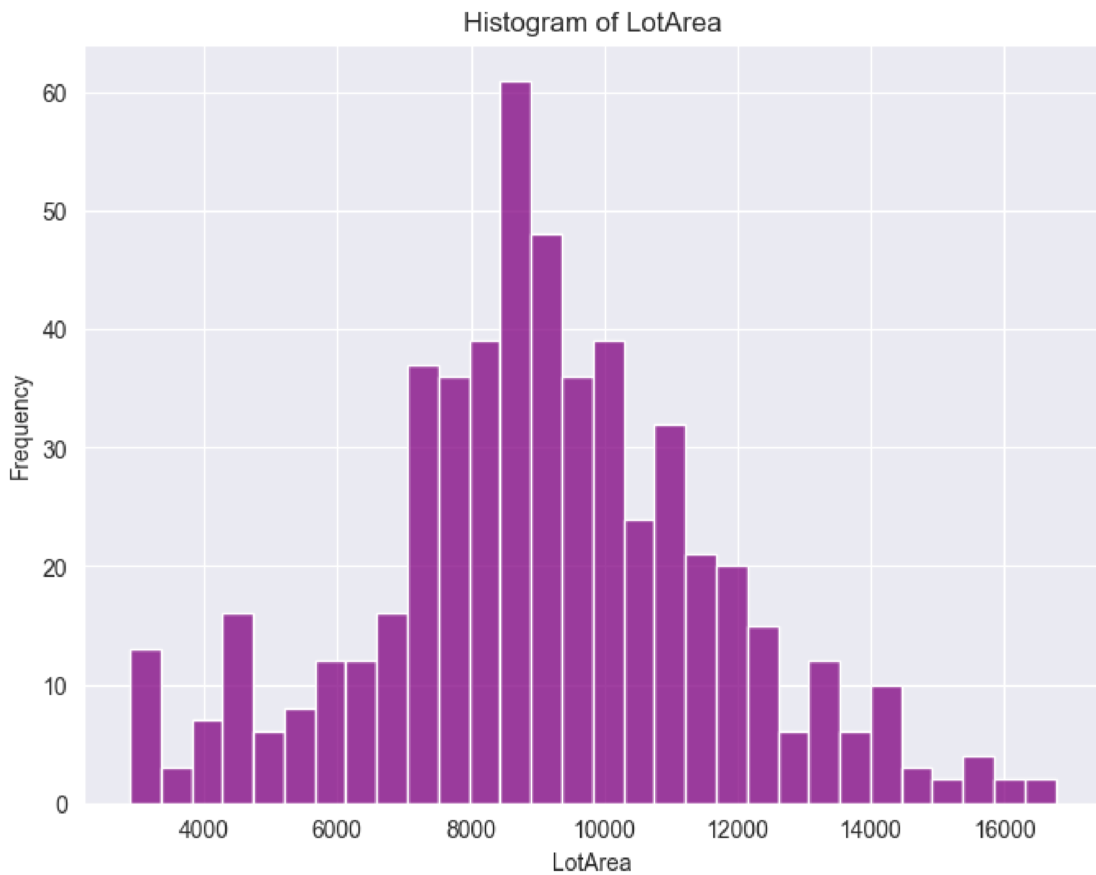
[206]:
```python
for col in train_numerical_cols:
    plt.figure(figsize=(8, 6))
    sns.histplot(new_train[col], bins=30,color='purple')
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```

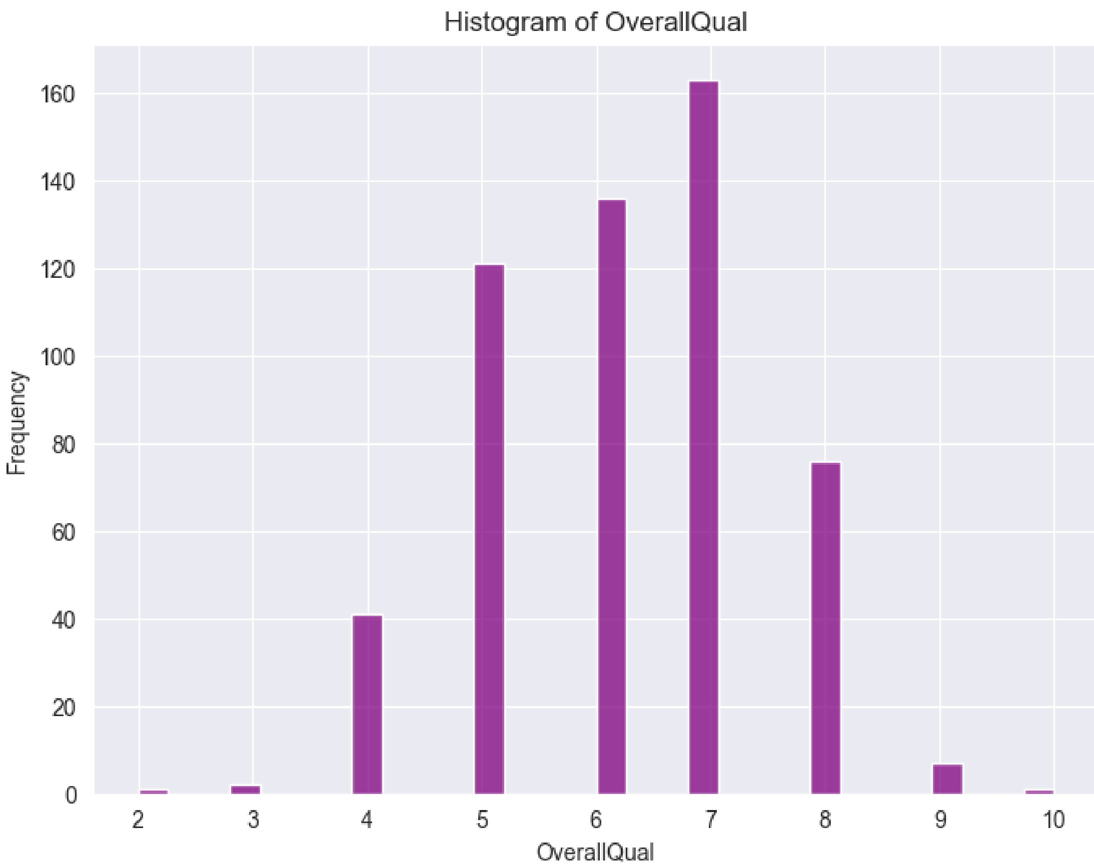

Histogram of Id
Frequency
Id
0
5
10
15
20
25
0
200
400
600
800
1000
1200
1400

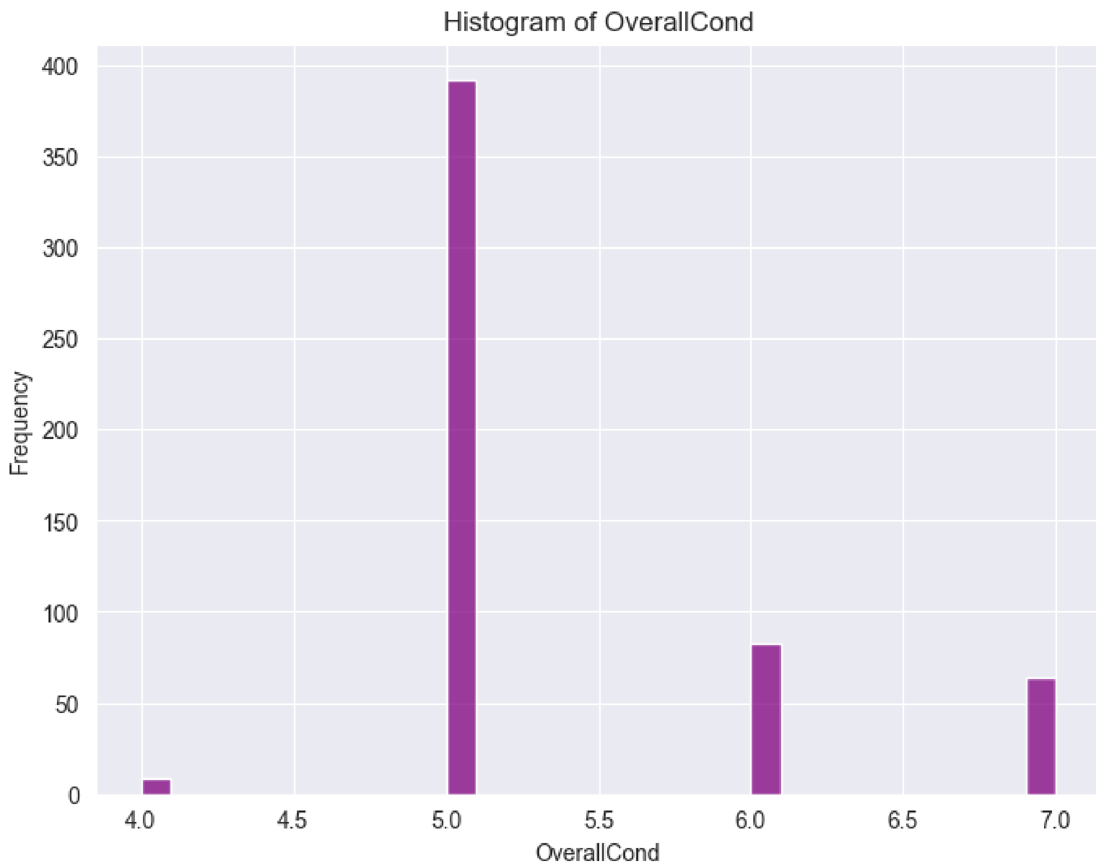

Histogram of MSSubClass
Frequency
MSSubClass
0
50
100
150
200
20
40
60
80
100
120

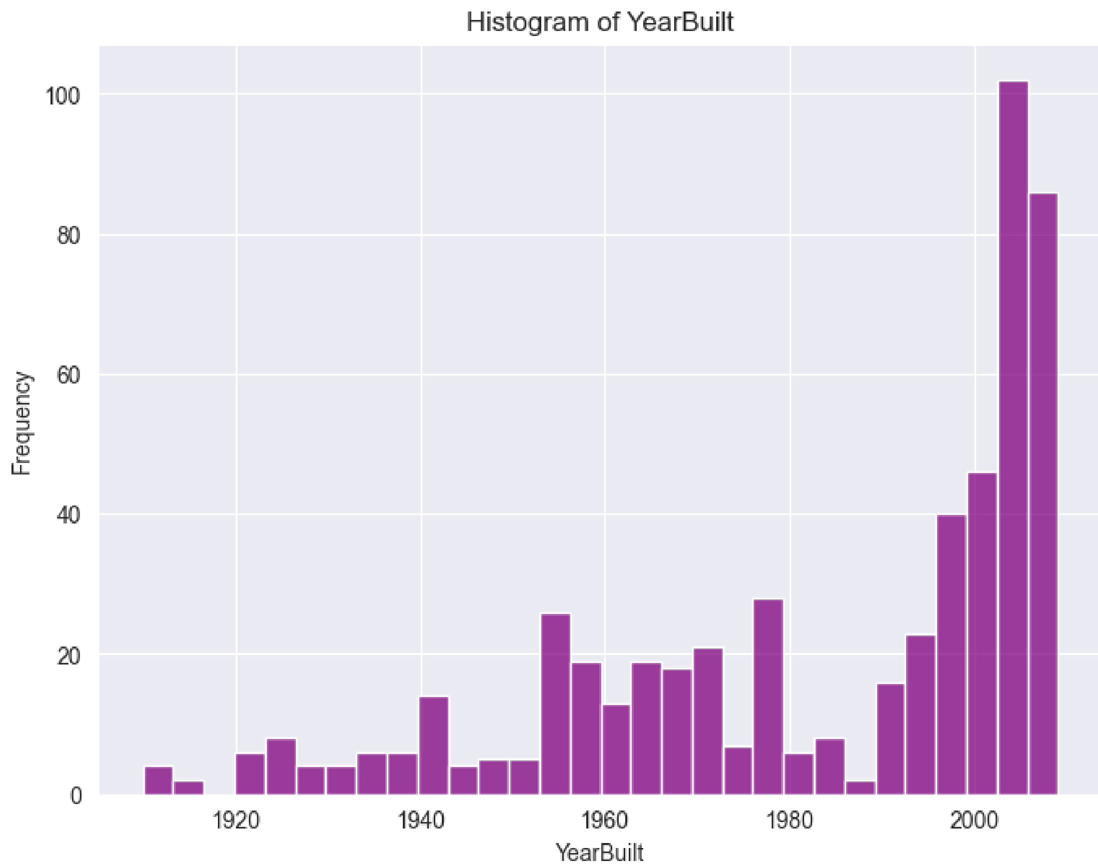


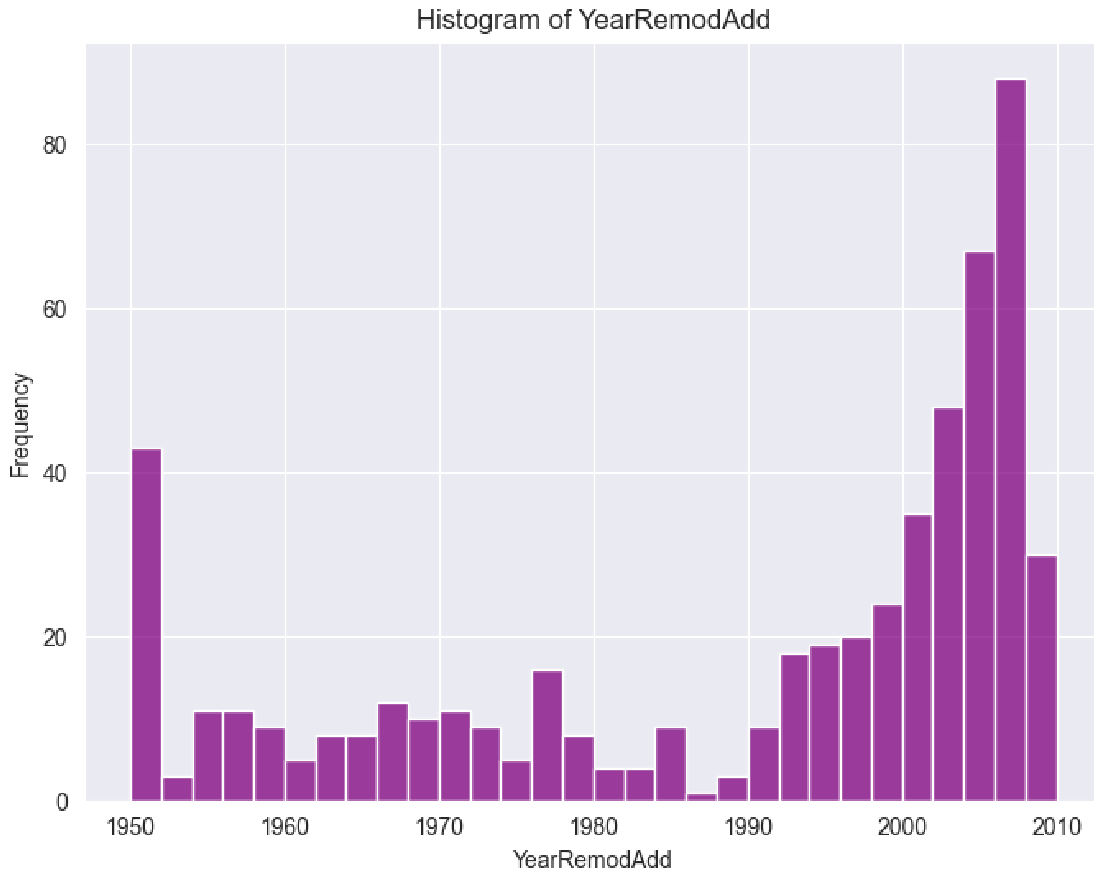
Histogram of LotFrontage
Frequency
0
20
40
60
80
100
120
140
30
40
50
60
70
80
90
100
110
LotFrontage

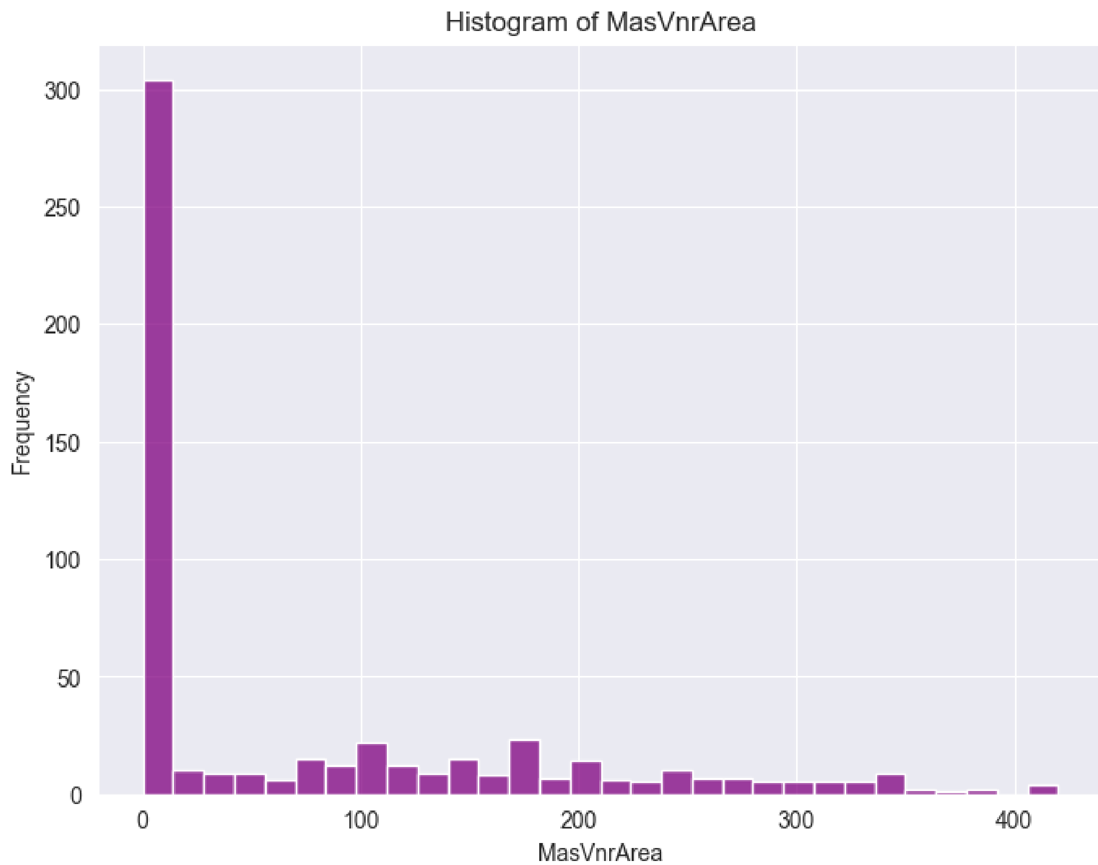

Histogram of LotArea
Frequency
0
10
20
30
40
50
60
4000
6000
8000
10000
12000
14000
16000
LotArea

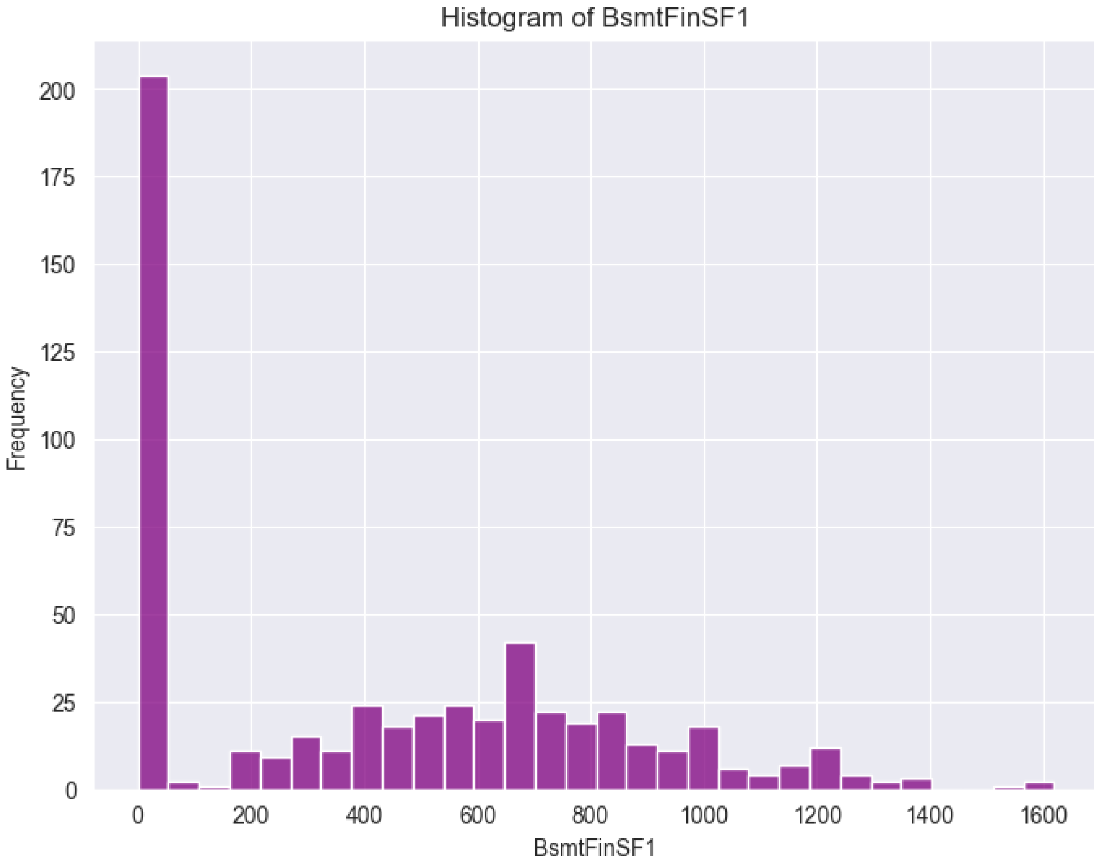

Histogram of OverallQual
Frequency
OverallQual
0
20
40
60
80
100
120
140
160
2
3
4
5
6
7
8
9
10

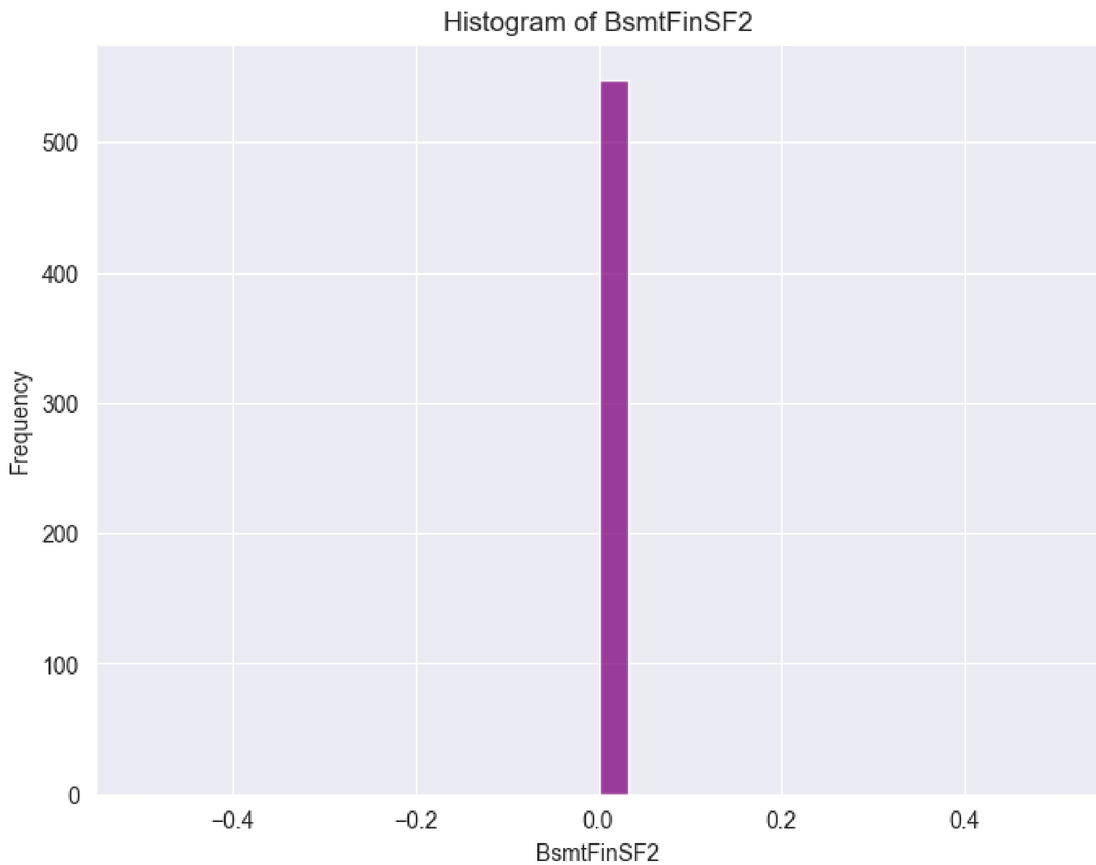

Histogram of OverallCond
Frequency
0
50
100
150
200
250
300
350
400
4.0
4.5
5.0
5.5
6.0
6.5
7.0
OverallCond

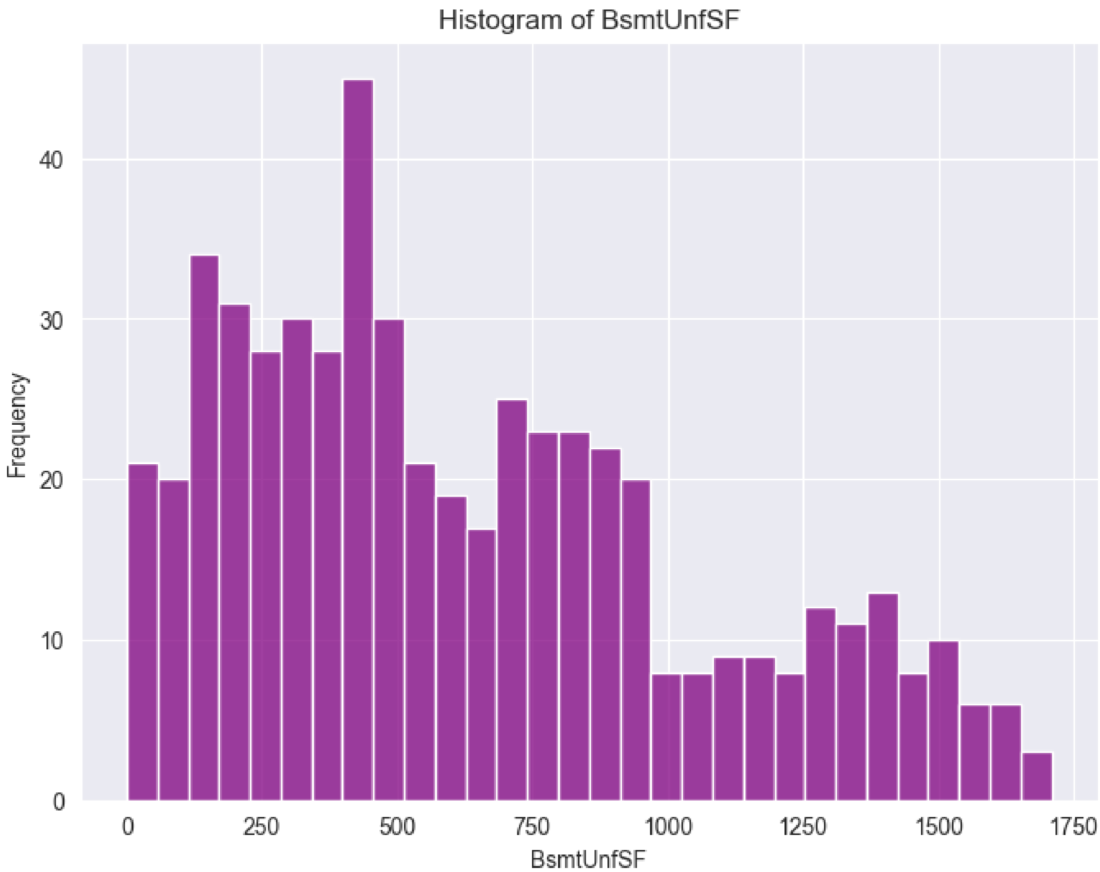

Histogram of YearBuilt
100
80
60
40
20
0
Frequency
1920
1940
1960
1980
2000
YearBuilt

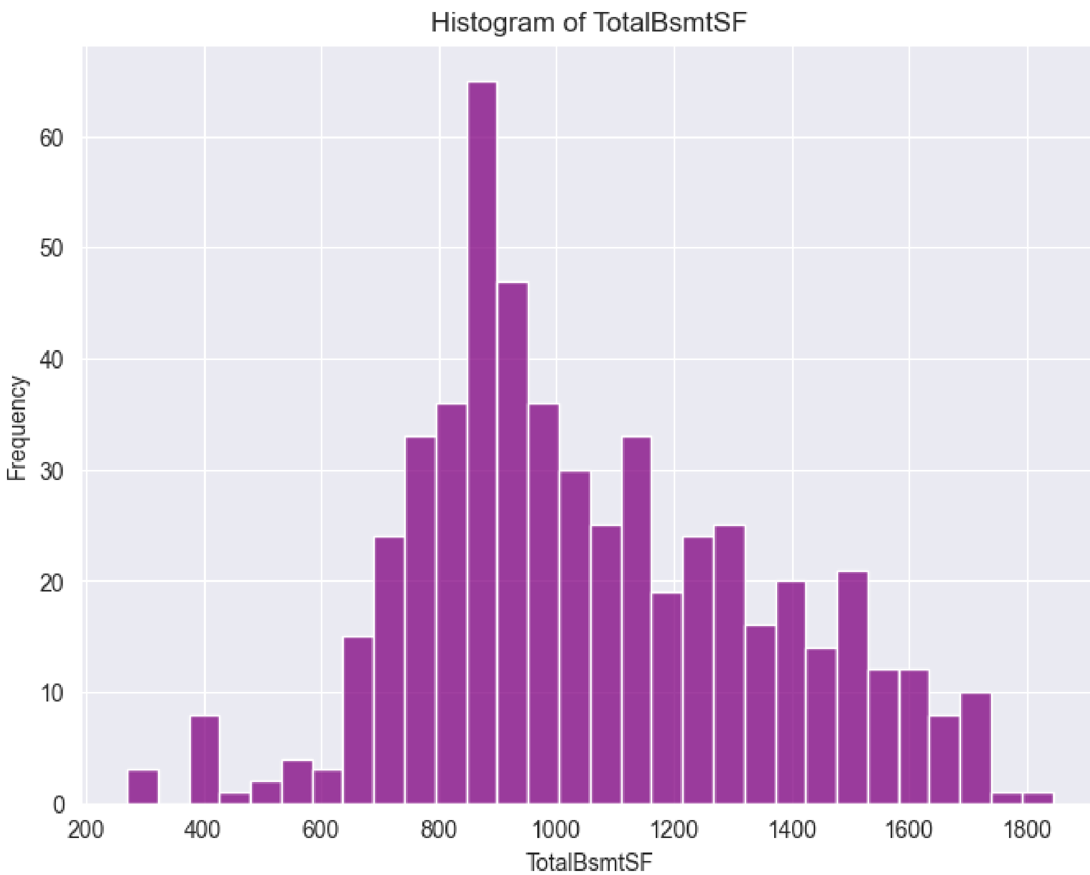

Histogram of YearRemodAdd
Frequency
YearRemodAdd
0
20
40
60
80
1950
1960
1970
1980
1990
2000
2010

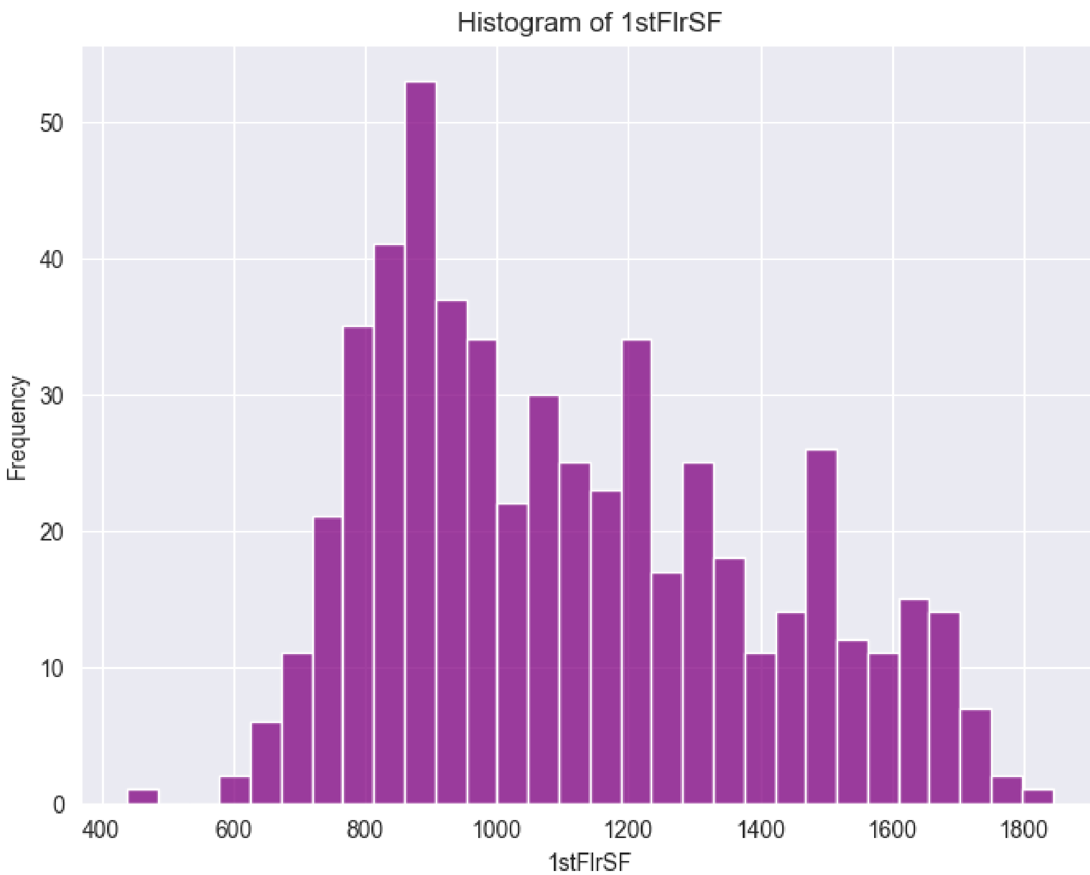

Histogram of MasVnrArea
Frequency
MasVnrArea
0
50
100
150
200
250
300
400

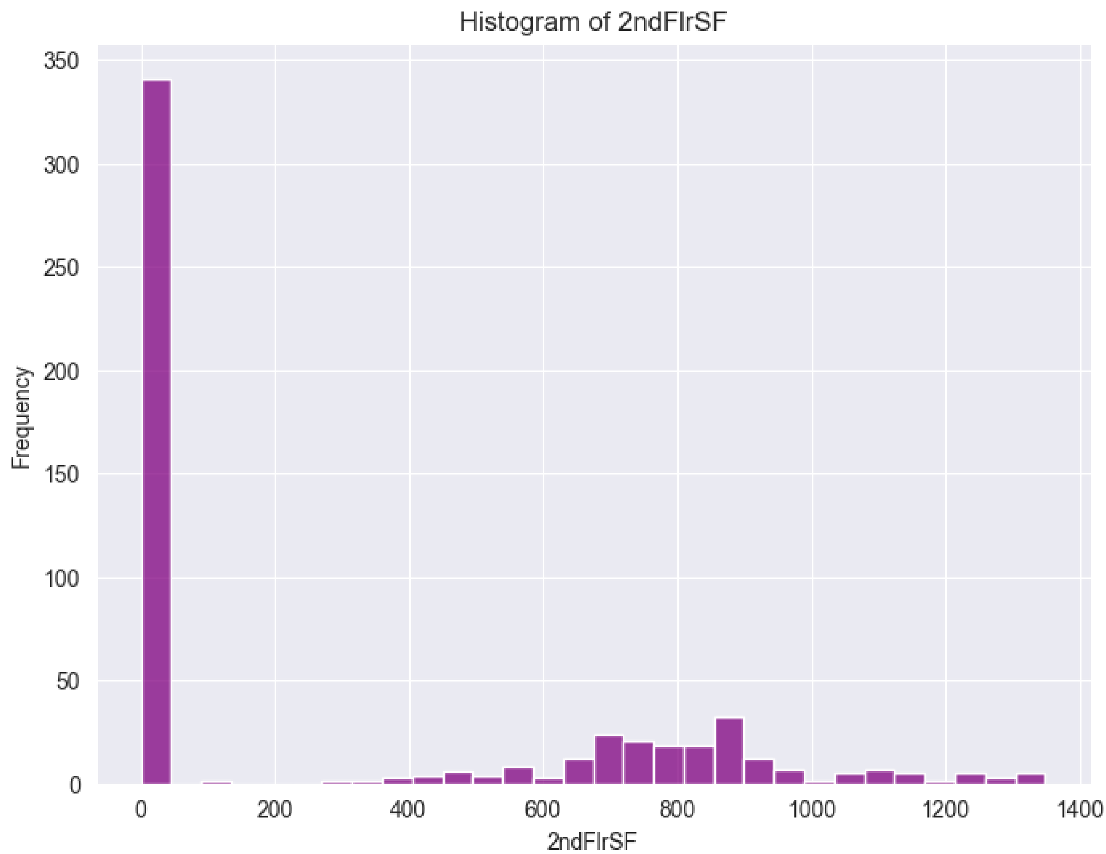

Histogram of BsmtFinSF1
Frequency
BsmtFinSF1
0
25
50
75
100
125
150
175
200
0
200
400
600
800
1000
1200
1400
1600

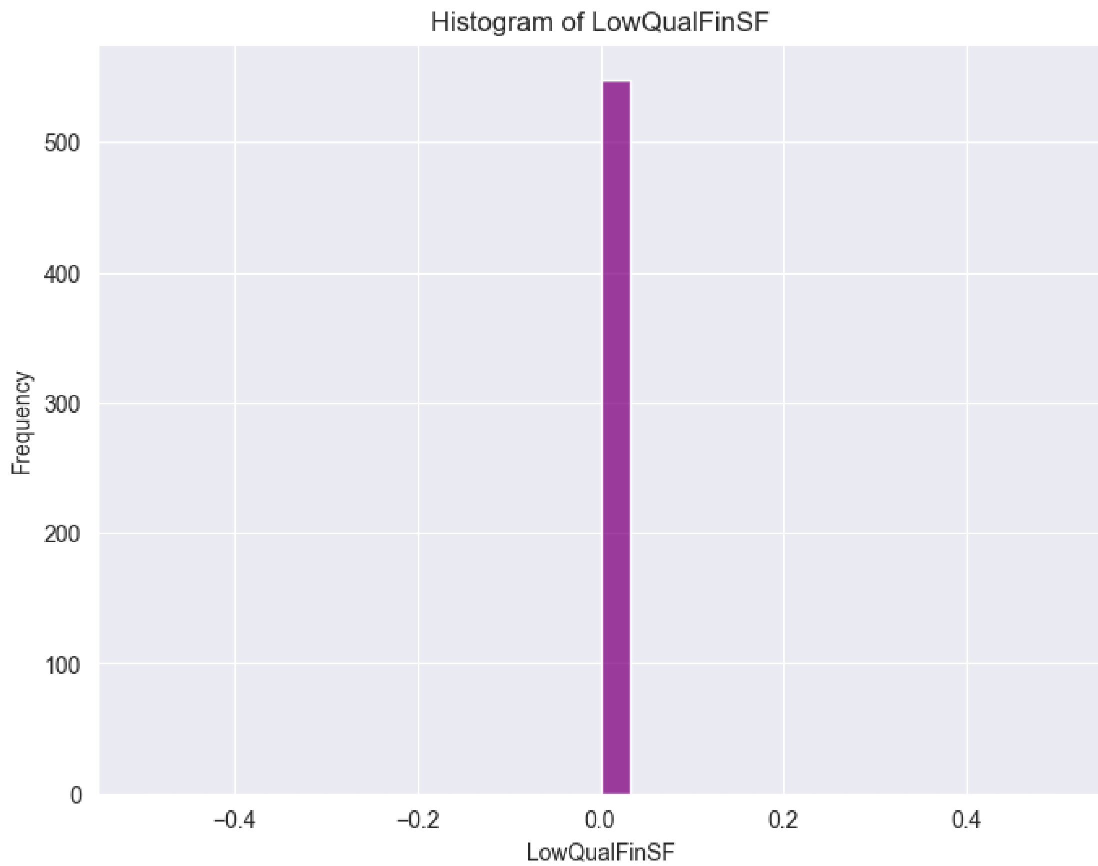

Histogram of BsmtFinSF2
Frequency
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
0.4
BsmtFinSF2

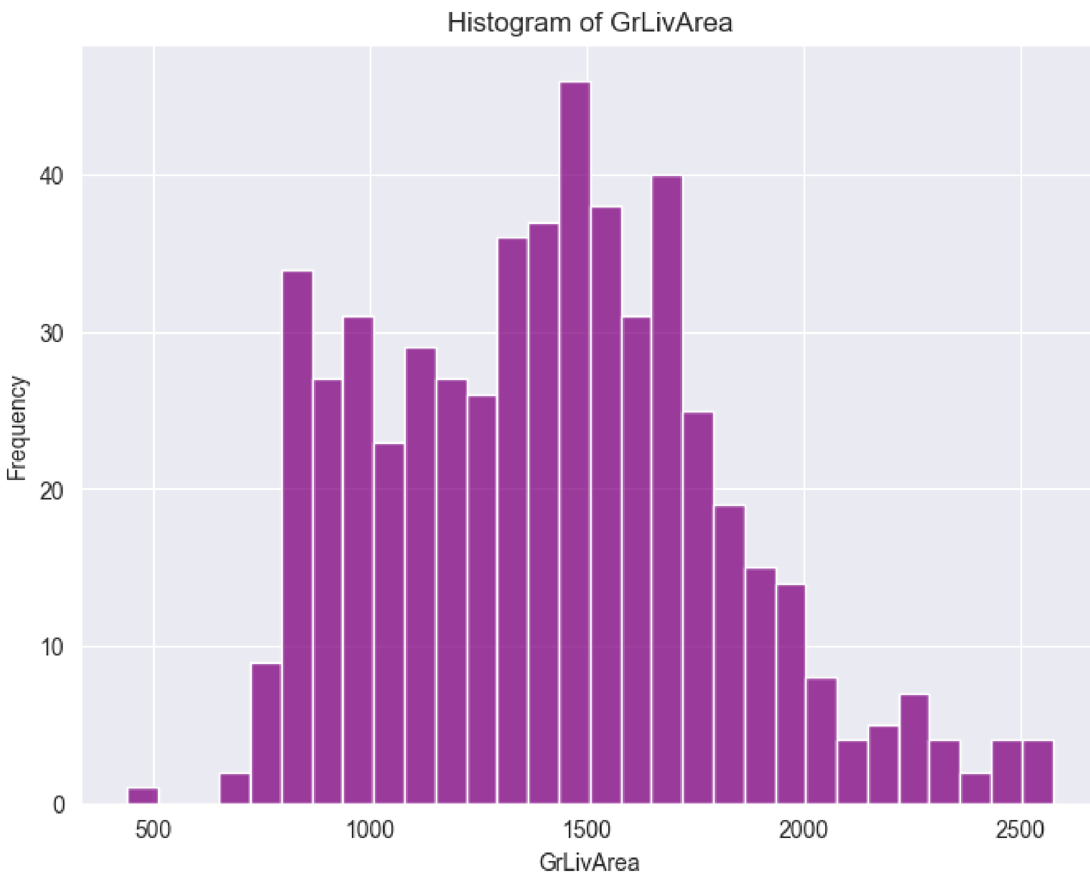

Histogram of BsmtUnfSF
Frequency
BsmtUnfSF
0
10
20
30
40
0
250
500
750
1000
1250
1500
1750

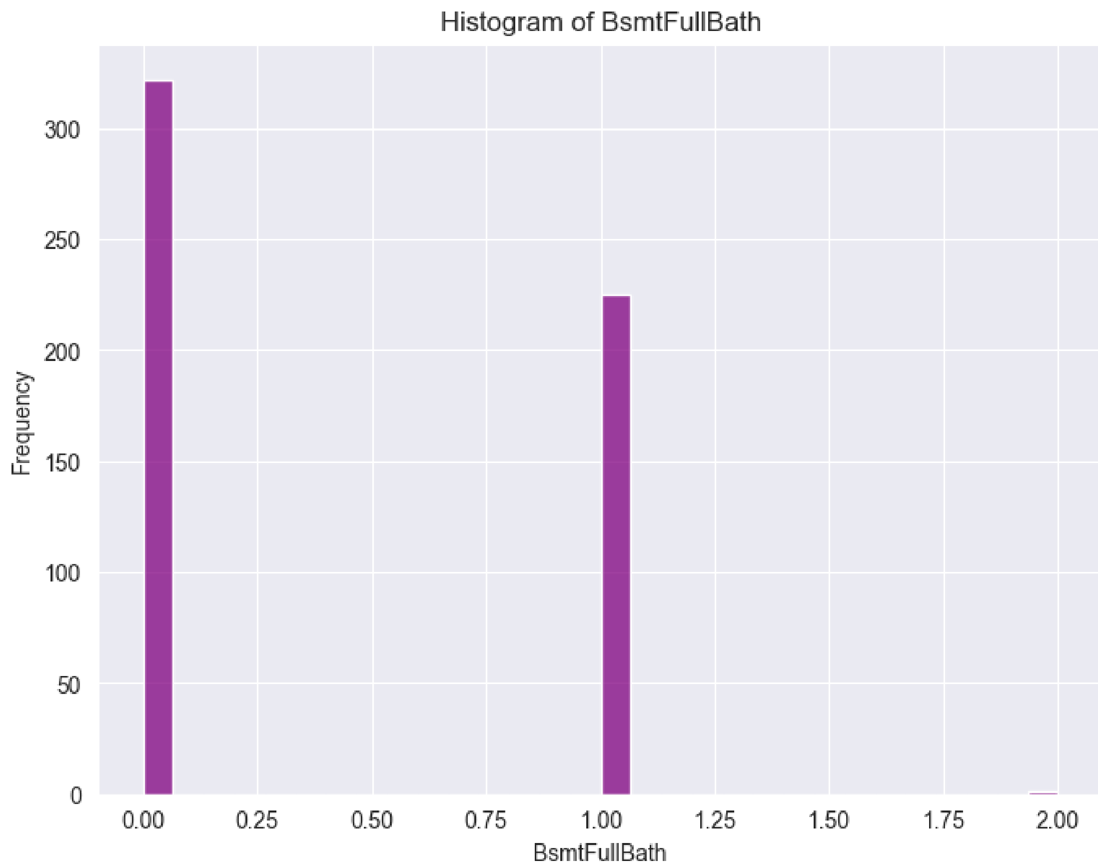

Histogram of TotalBsmtSF
Frequency
TotalBsmtSF
0
10
20
30
40
50
60
200
400
600
800
1000
1200
1400
1600
1800

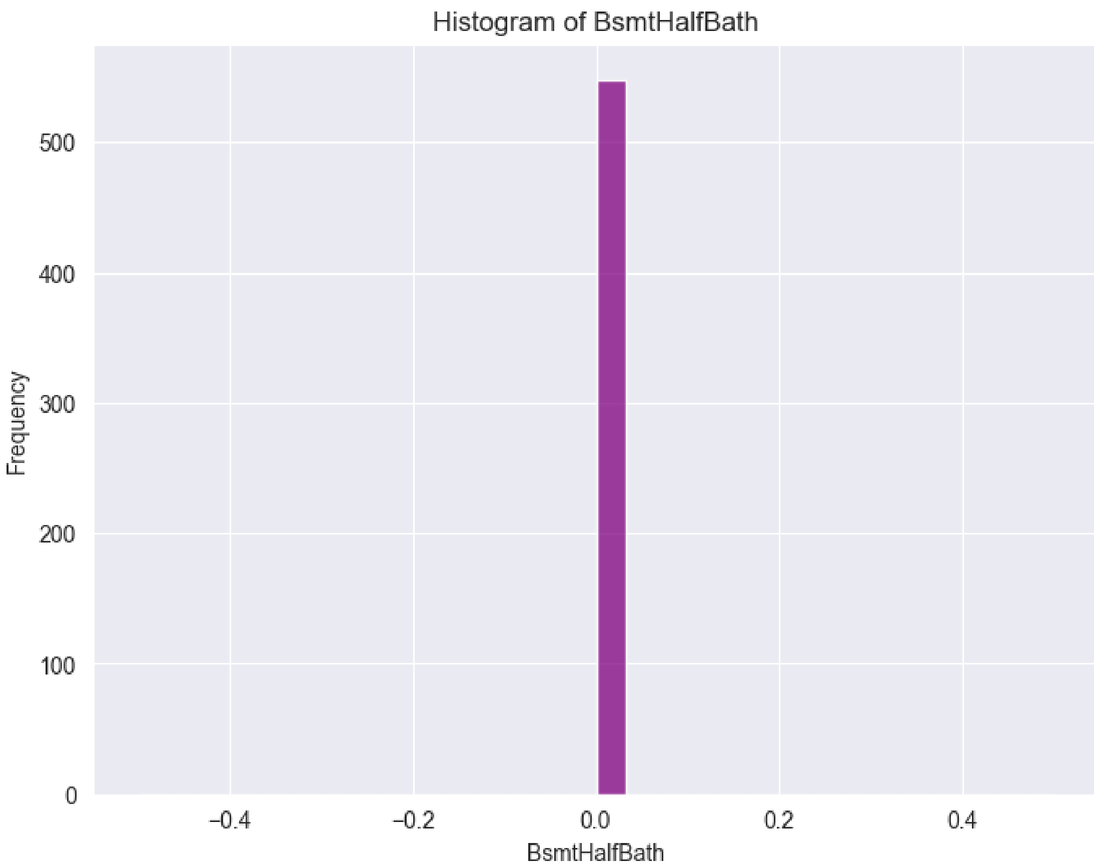

Histogram of 1stFlrSF
Frequency
1stFlrSF
0
10
20
30
40
50
400
600
800
1000
1200
1400
1600
1800

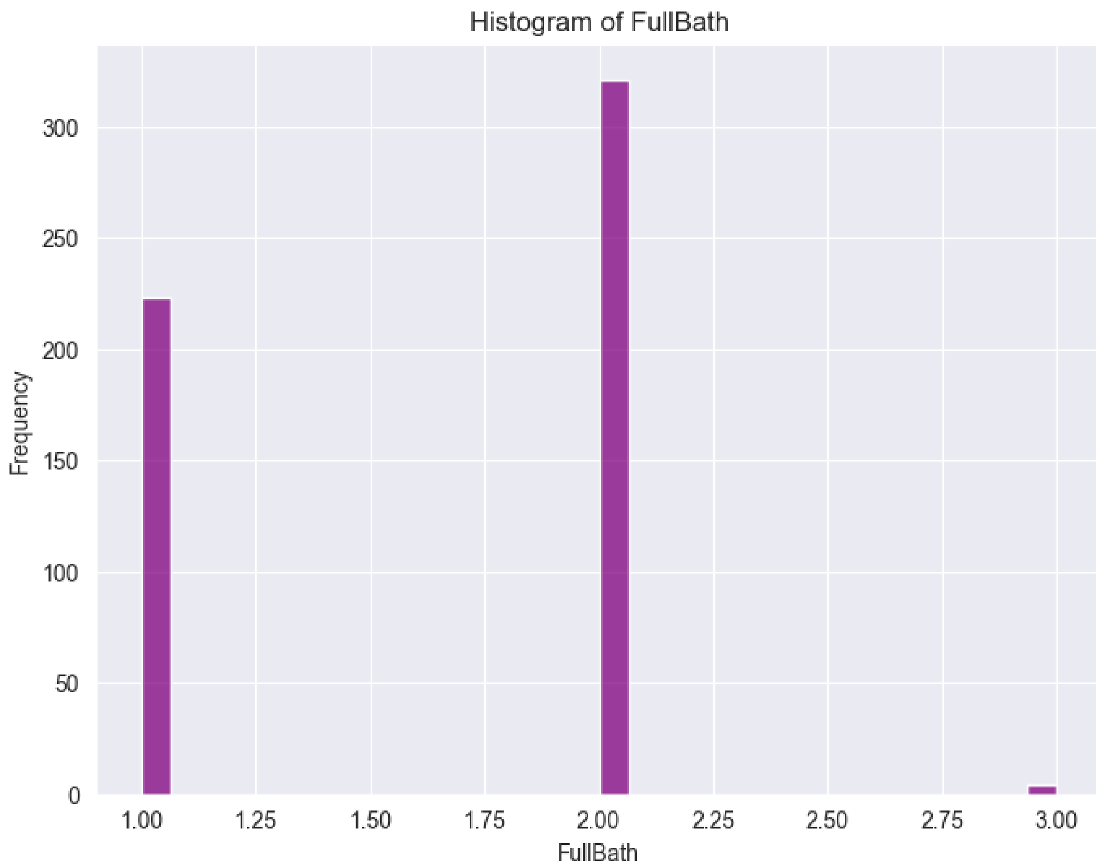

Histogram of 2ndFlrSF
Frequency
350
300
250
200
150
100
50
0
0
200
400
600
800
1000
1200
1400
2ndFlrSF

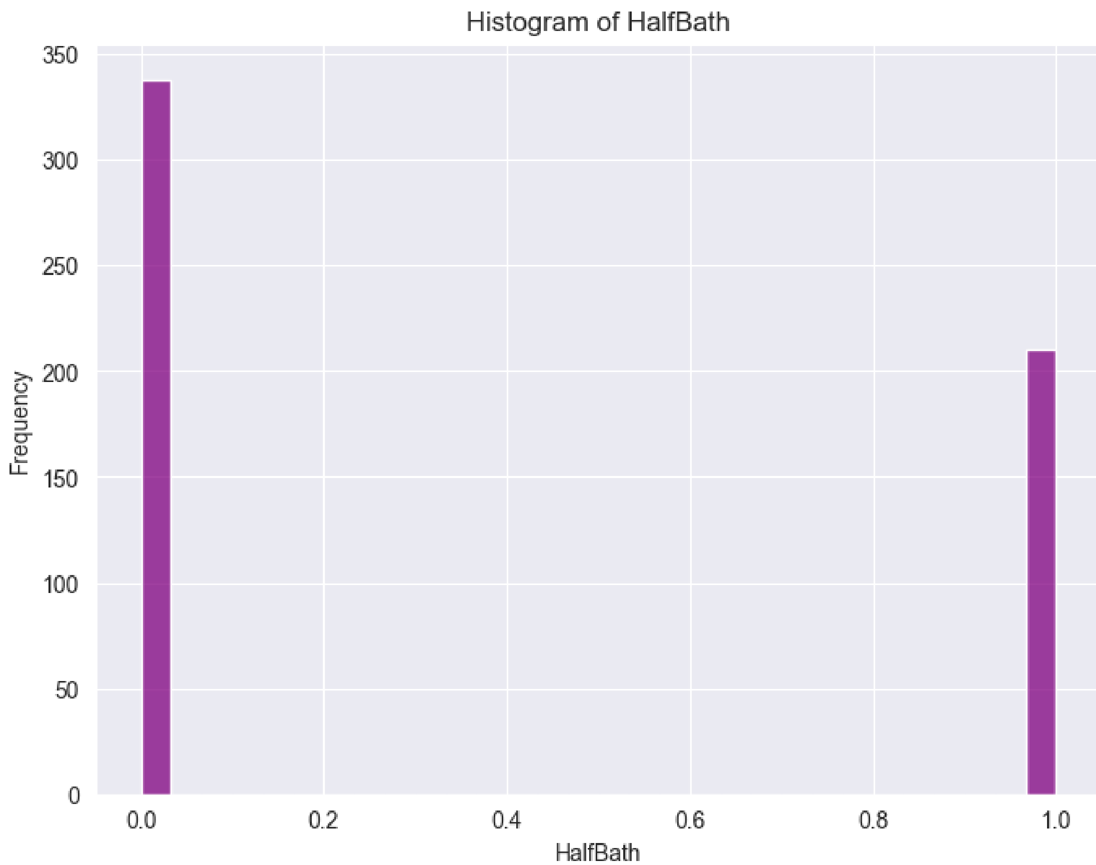

Histogram of LowQualFinSF
Frequency
500
400
300
200
100
0
−0.4
−0.2
0.0
0.2
0.4
LowQualFinSF


Histogram of GrLivArea
Frequency
0
10
20
30
40
500
1000
1500
2000
2500
GrLivArea

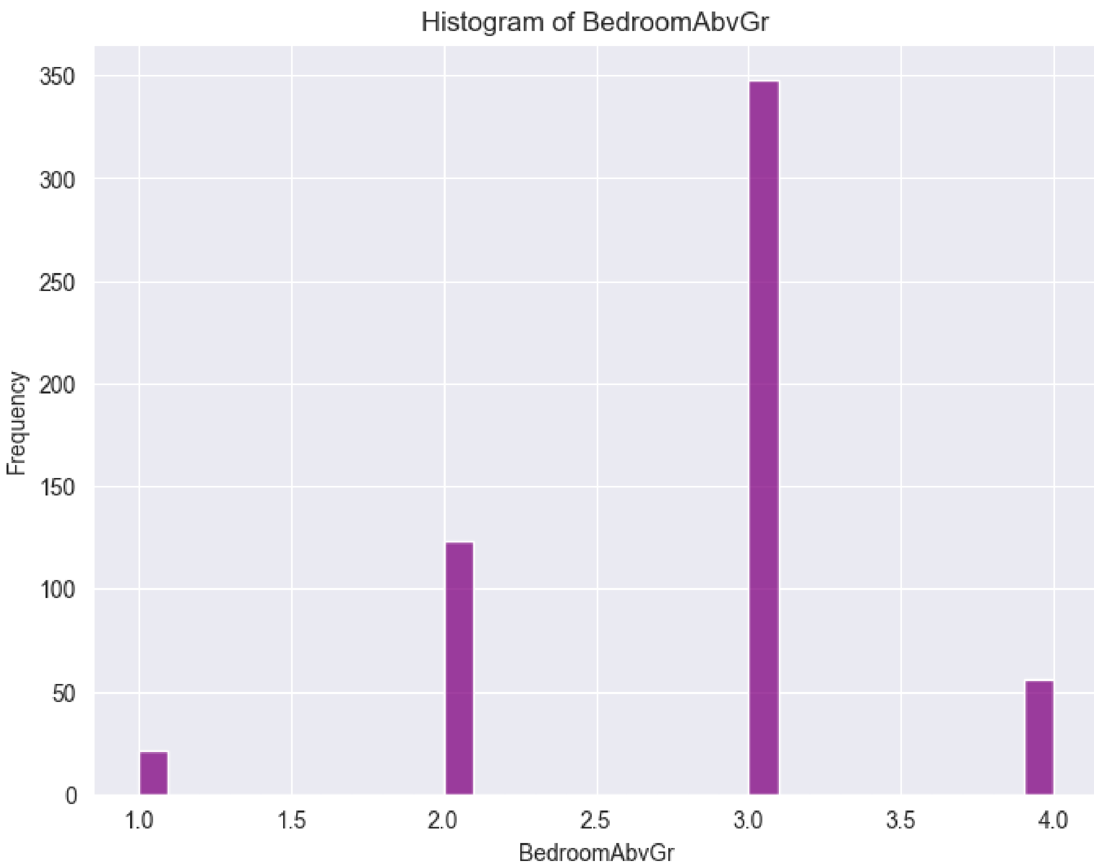

Histogram of BsmtFullBath
300
250
200
150
100
50
0
Frequency
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
BsmtFullBath

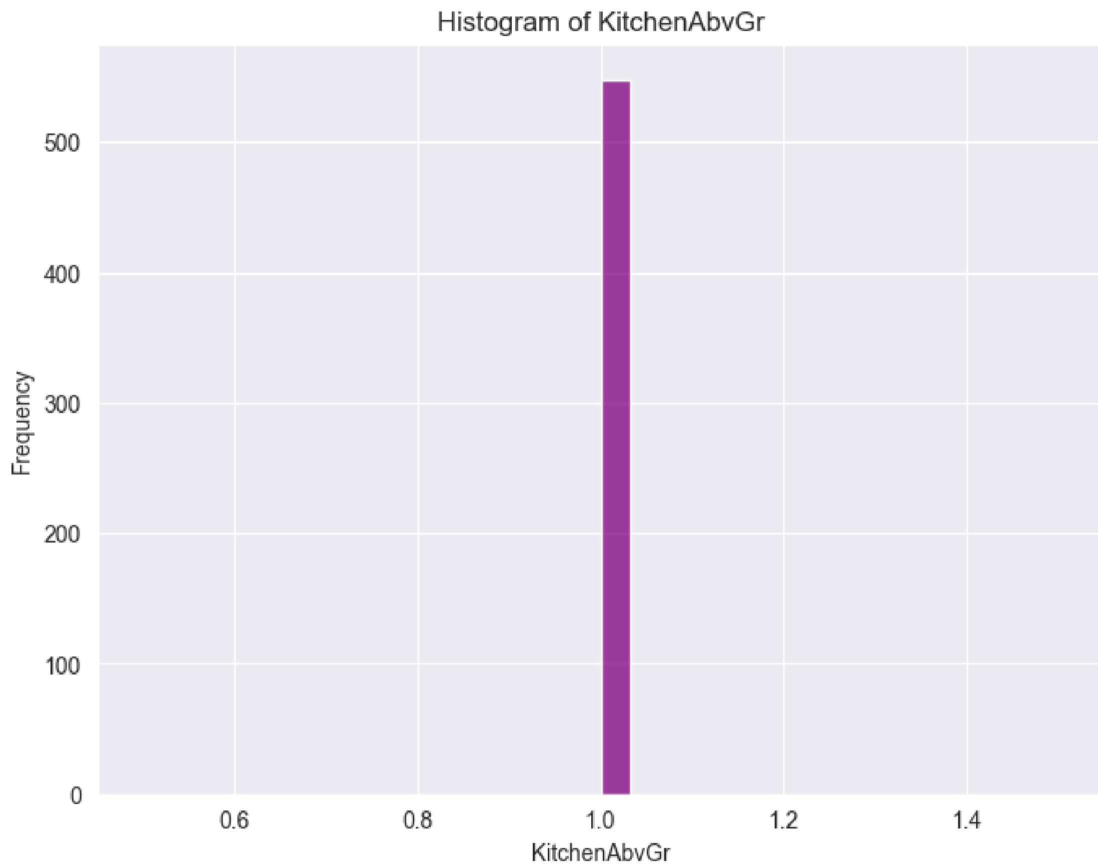

Histogram of BsmtHalfBath
Frequency
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
0.4
BsmtHalfBath


Histogram of FullBath
Frequency
0
50
100
150
200
250
300
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
FullBath

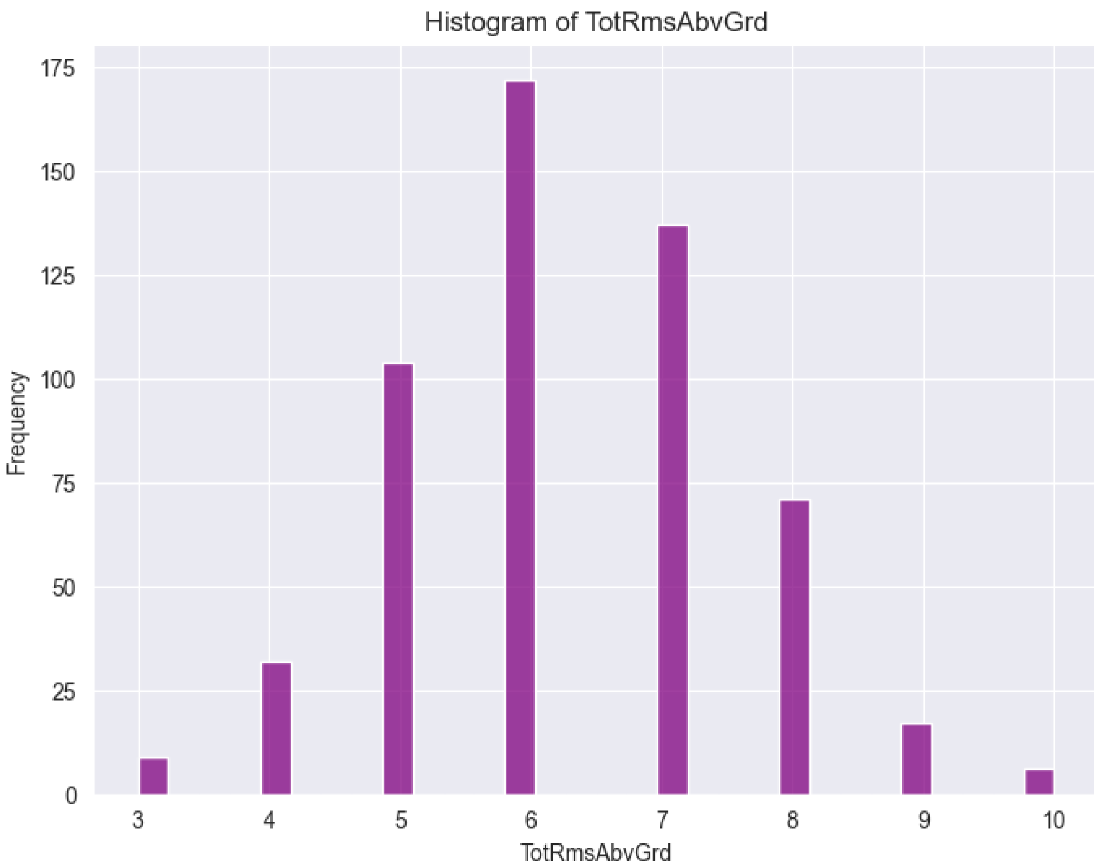


Histogram of HalfBath
Frequency
HalfBath
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0

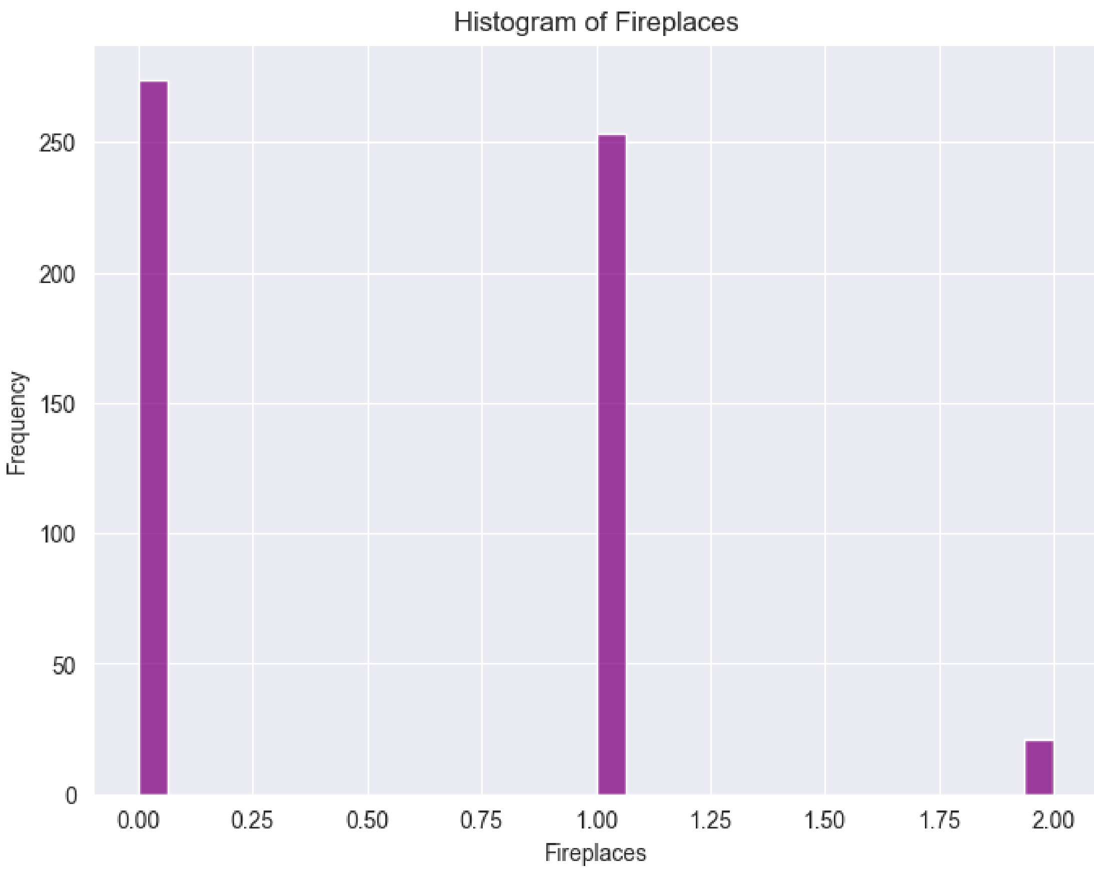

Histogram of BedroomAbvGr
Frequency
BedroomAbvGr
0
50
100
150
200
250
300
350
1.0
1.5
2.0
2.5
3.0
3.5
4.0

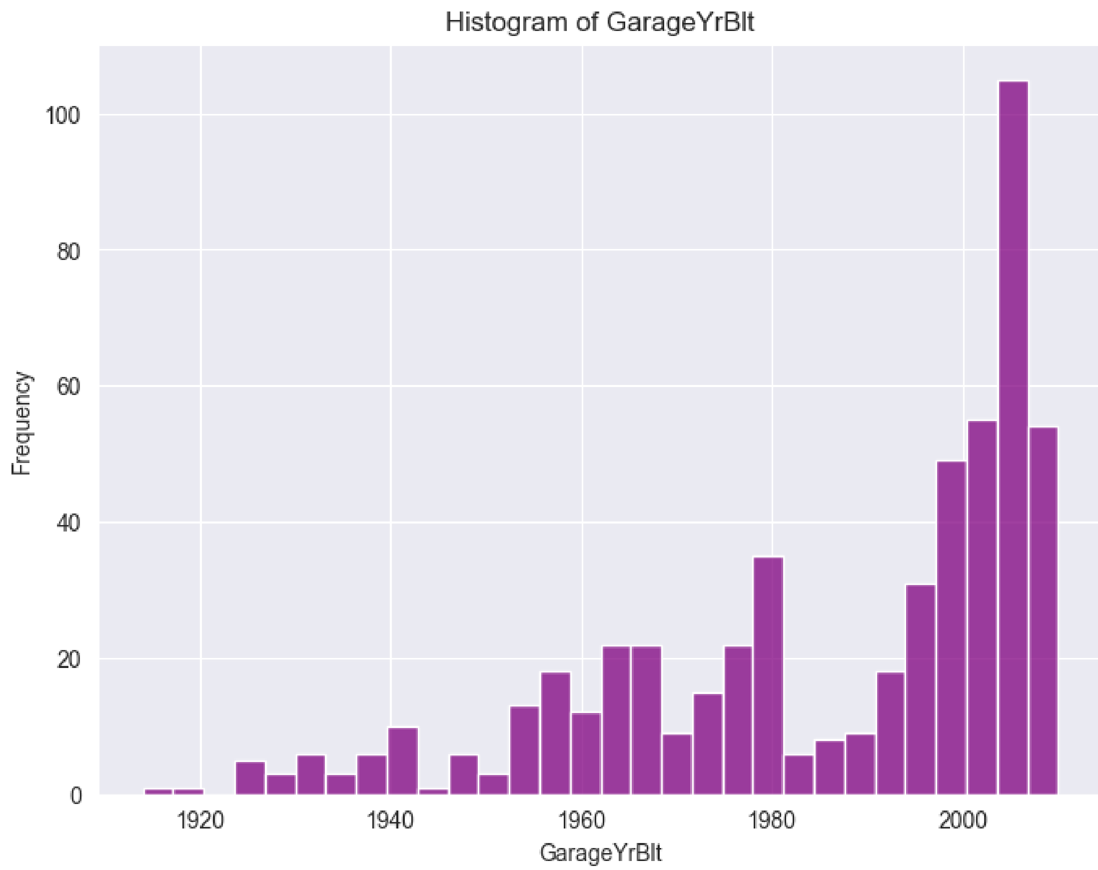


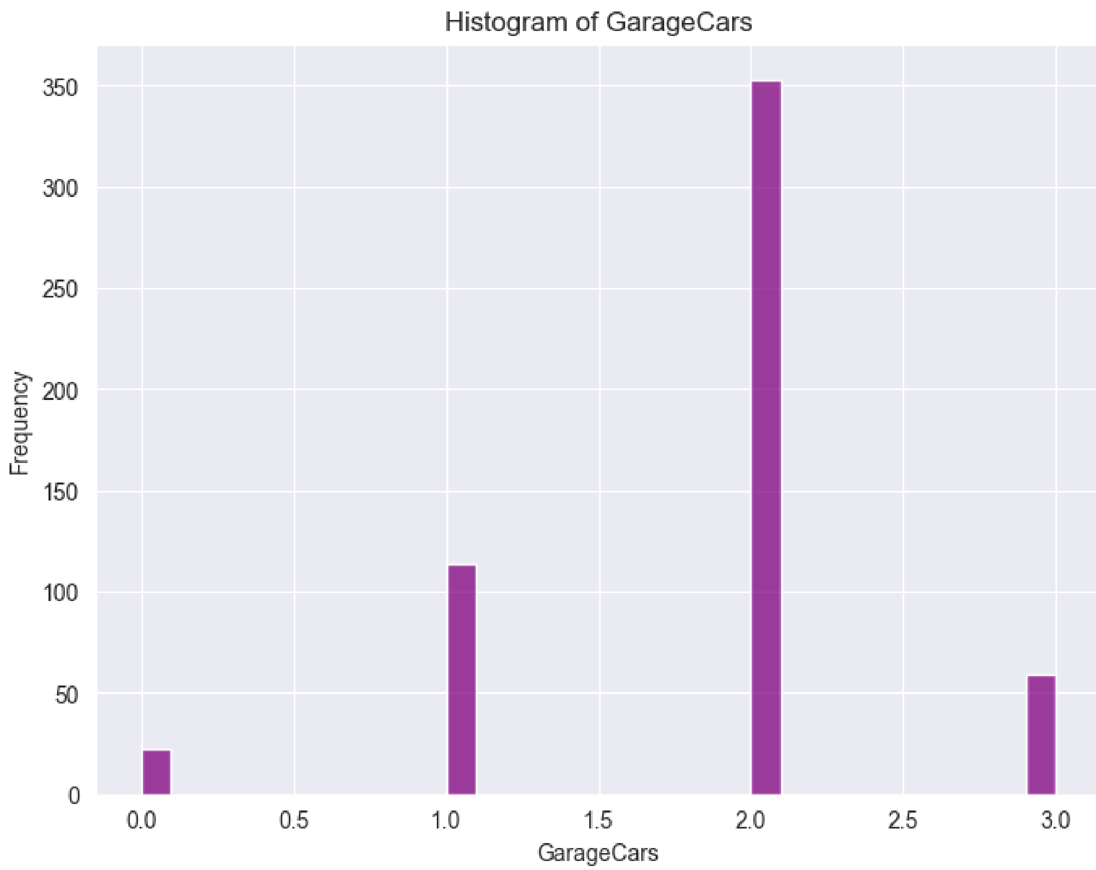
Histogram of KitchenAbvGr
Frequency
KitchenAbvGr
0
100
200
300
400
500
0.6
0.8
1.0
1.2
1.4

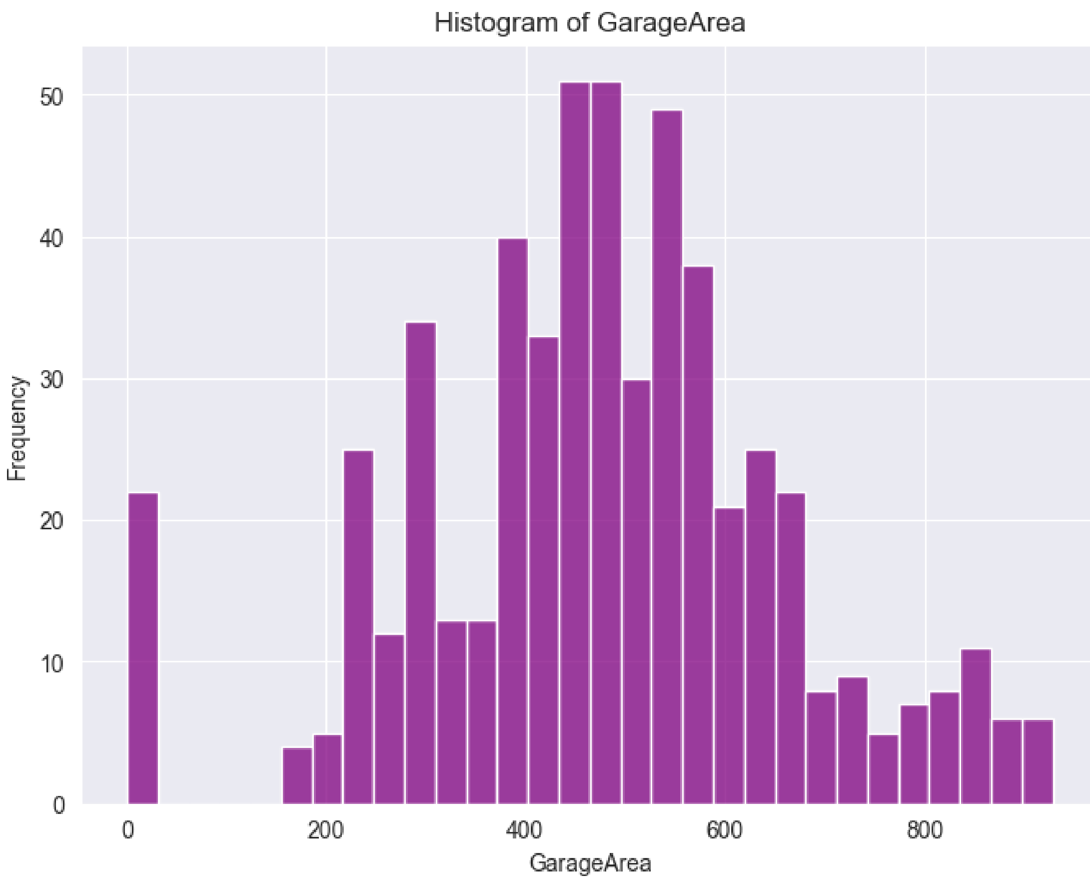


Histogram of TotRmsAbvGrd
Frequency
0
25
50
75
100
125
150
175
3
4
5
6
7
8
9
10
TotRmsAbvGrd

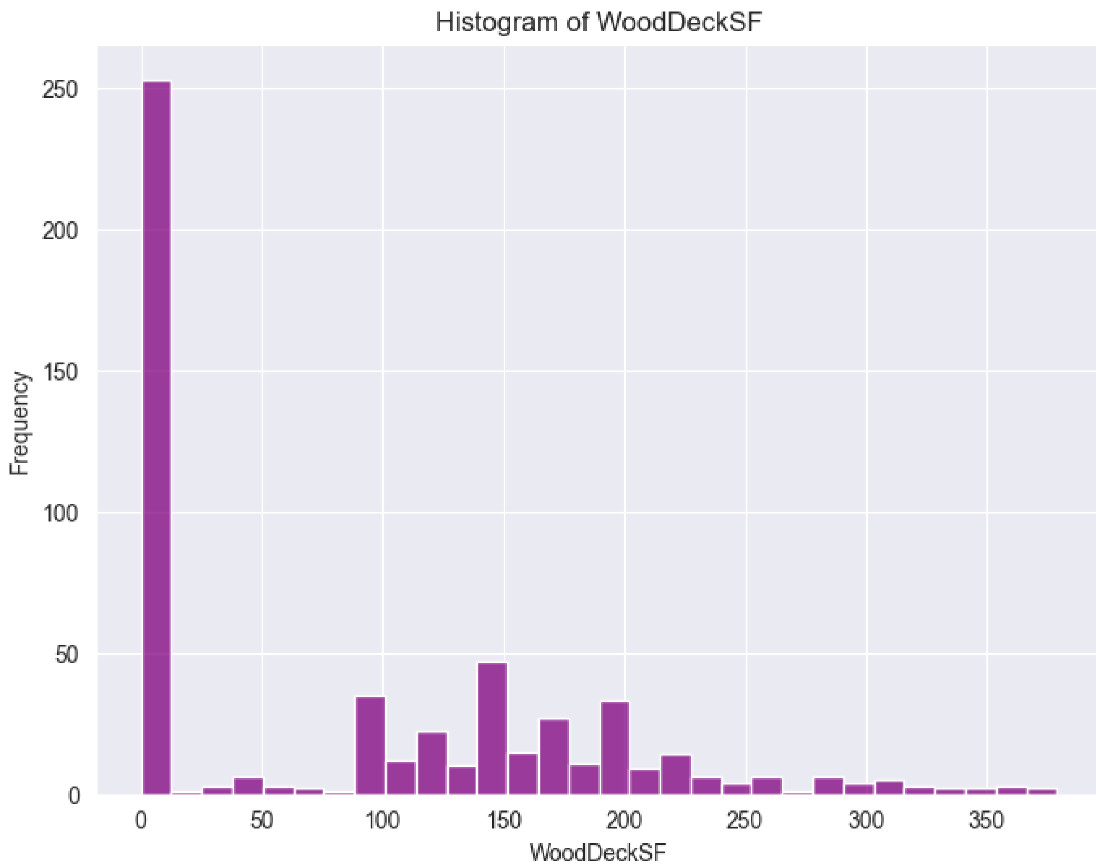

Histogram of Fireplaces
Frequency
Fireplaces
0
50
100
150
200
250
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

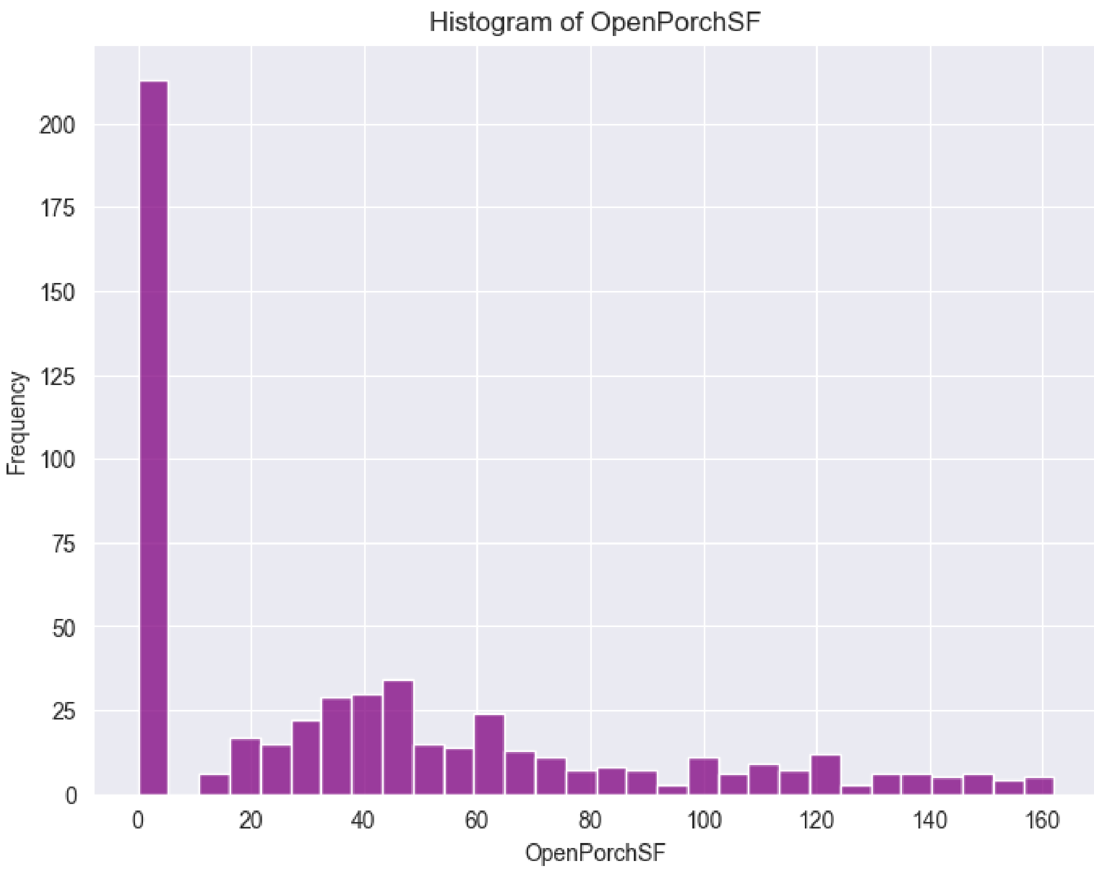

Histogram of GarageYrBlt
Frequency
GarageYrBlt
0
20
40
60
80
100
1920
1940
1960
1980
2000

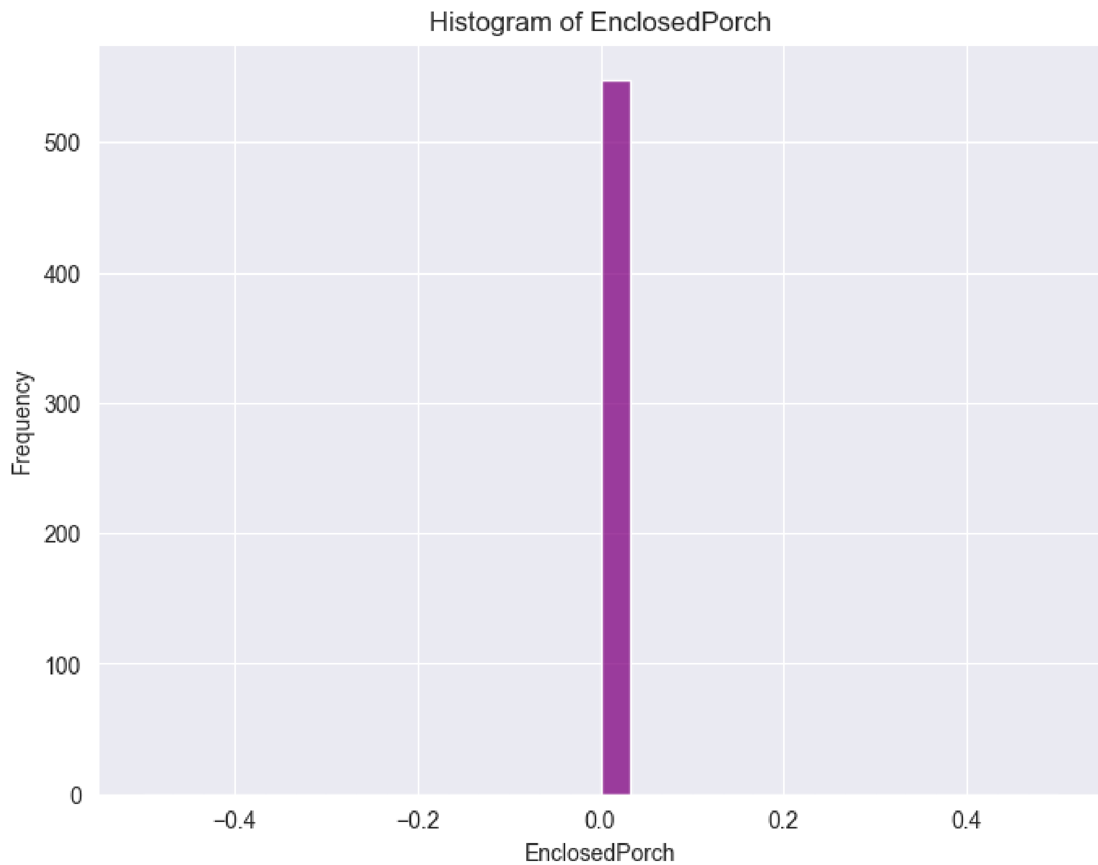

Histogram of GarageCars
Frequency
0
50
100
150
200
250
300
350
0.0
0.5
1.0
1.5
2.0
2.5
3.0
GarageCars

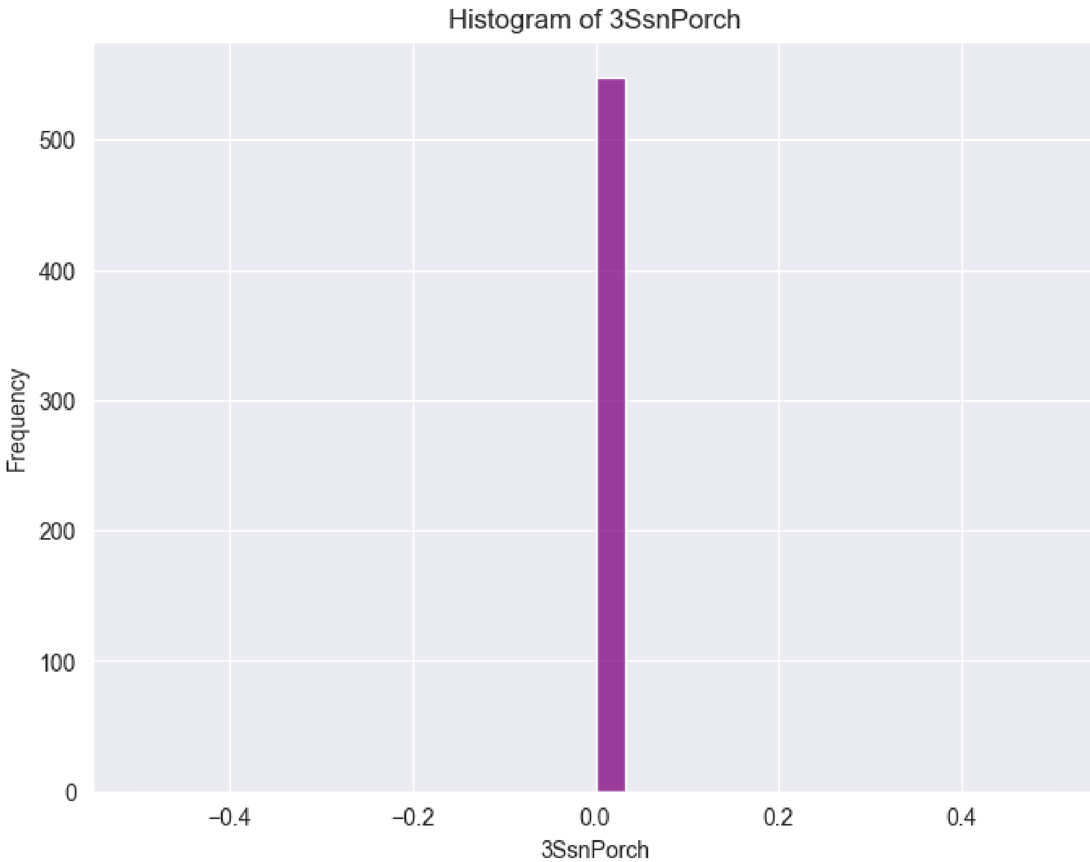


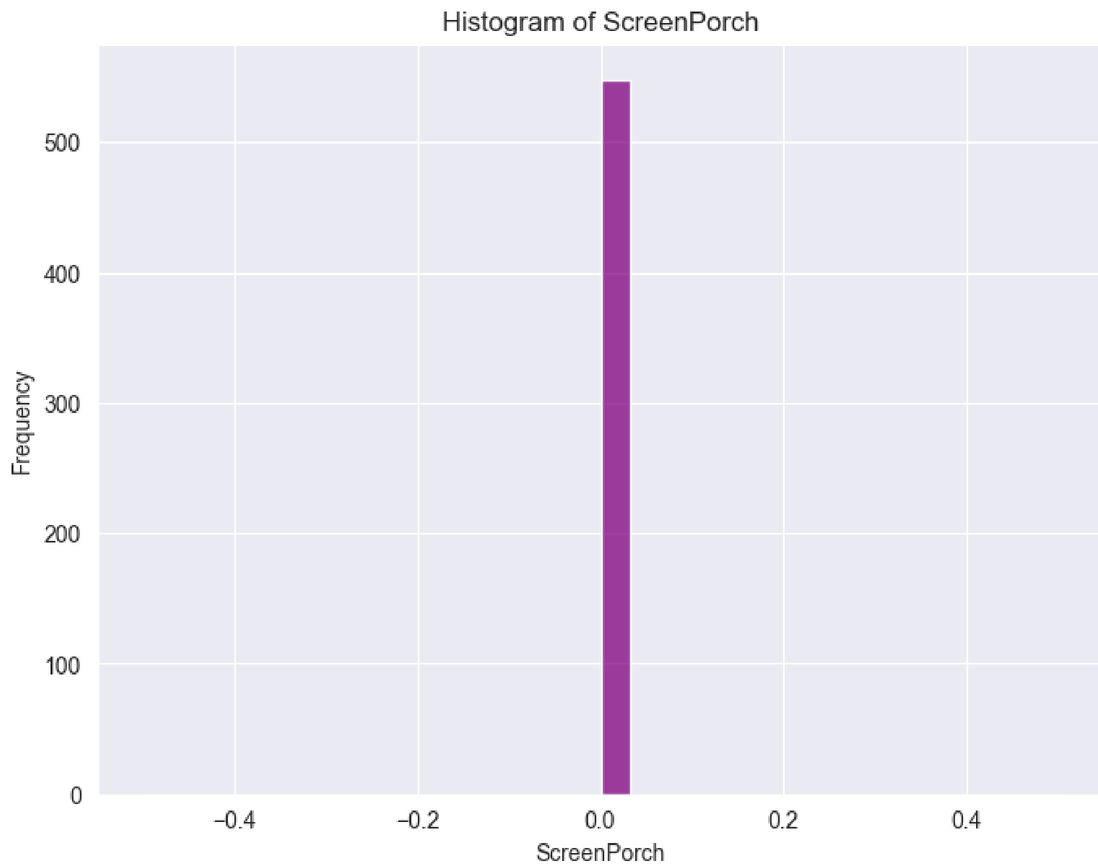
Histogram of GarageArea
Frequency
GarageArea
0
10
20
30
40
50
0
200
400
600
800

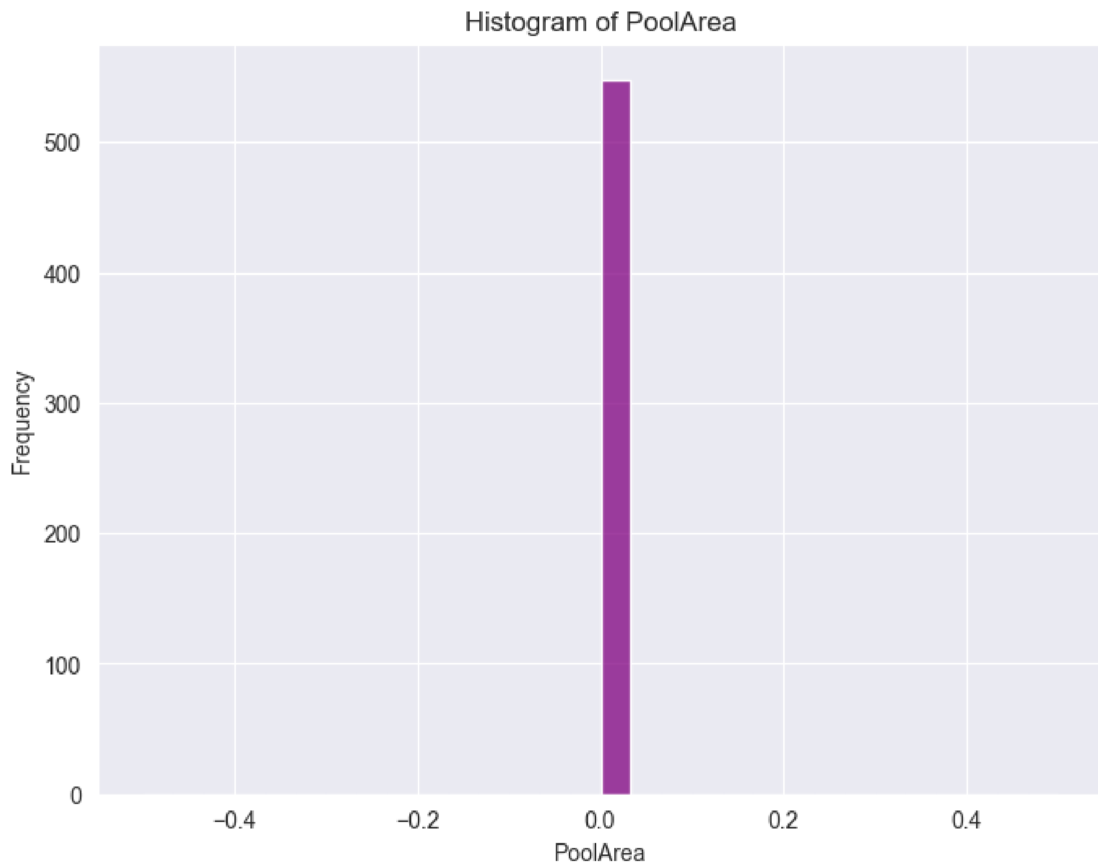

Histogram of WoodDeckSF
Frequency
0
50
100
150
200
250
0
50
100
150
200
250
300
350
WoodDeckSF

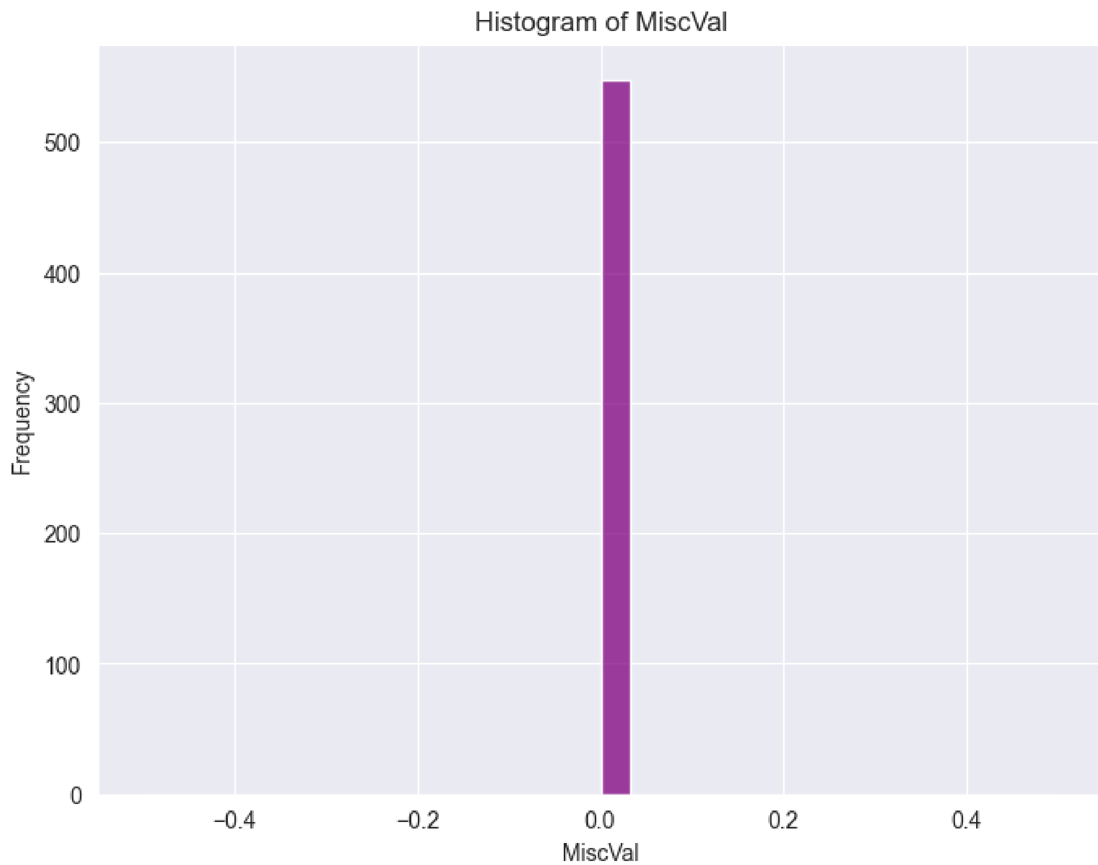

Histogram of OpenPorchSF
Frequency
OpenPorchSF
0
25
50
75
100
125
150
175
200
0
20
40
60
80
100
120
140
160

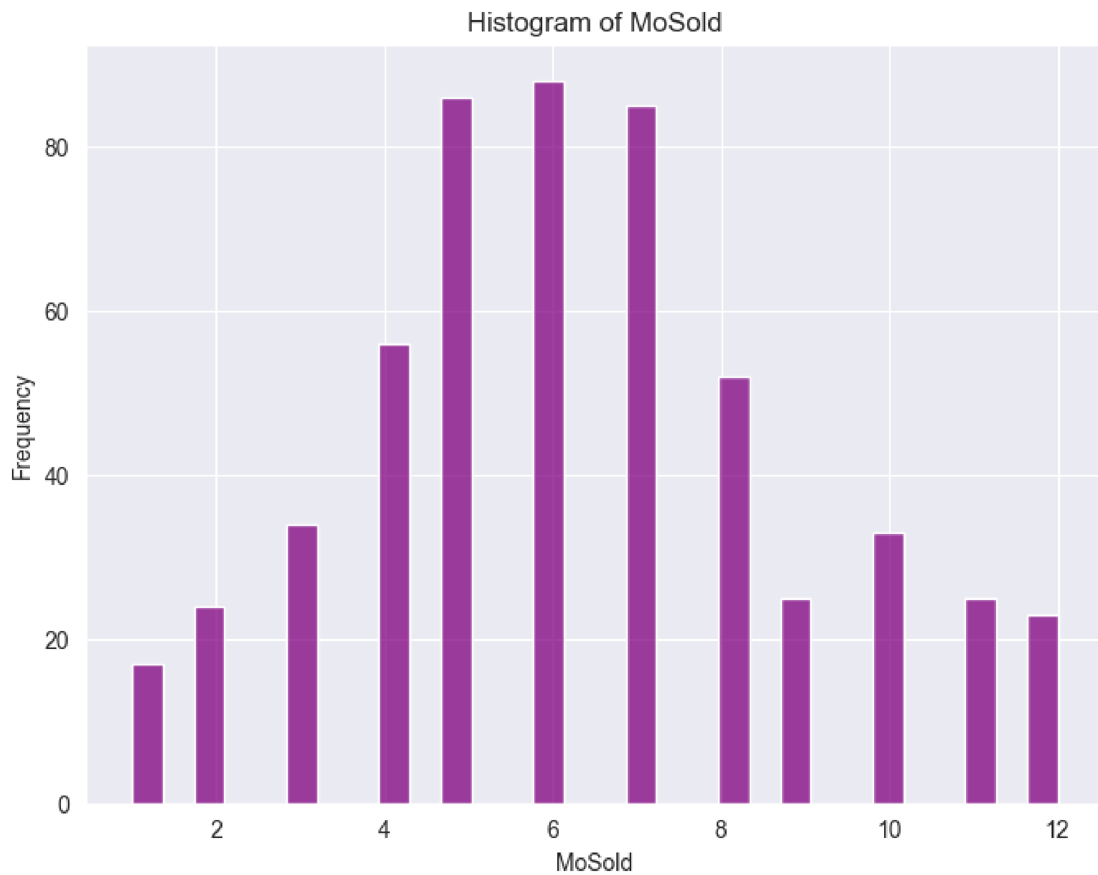

Histogram of EnclosedPorch
Frequency
500
400
300
200
100
0
−0.4
−0.2
0.0
0.2
0.4
EnclosedPorch


Histogram of 3SsnPorch
Frequency
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
0.4
3SsnPorch

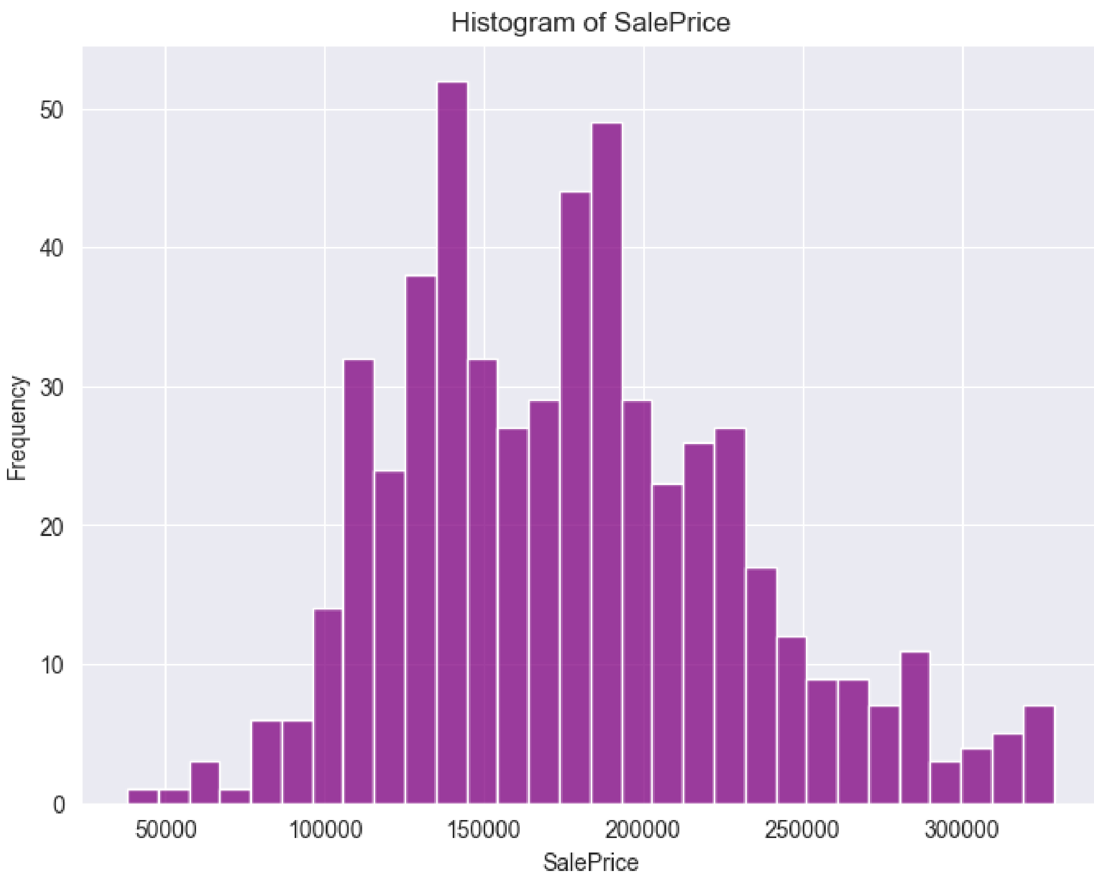

Histogram of ScreenPorch
Frequency
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
0.4
ScreenPorch


Histogram of PoolArea
Frequency
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
0.4
PoolArea


Histogram of MiscVal
Frequency
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
0.4
MiscVal


Histogram of MoSold
80
60
40
20
0
Frequency
2
4
6
8
10
12
MoSold

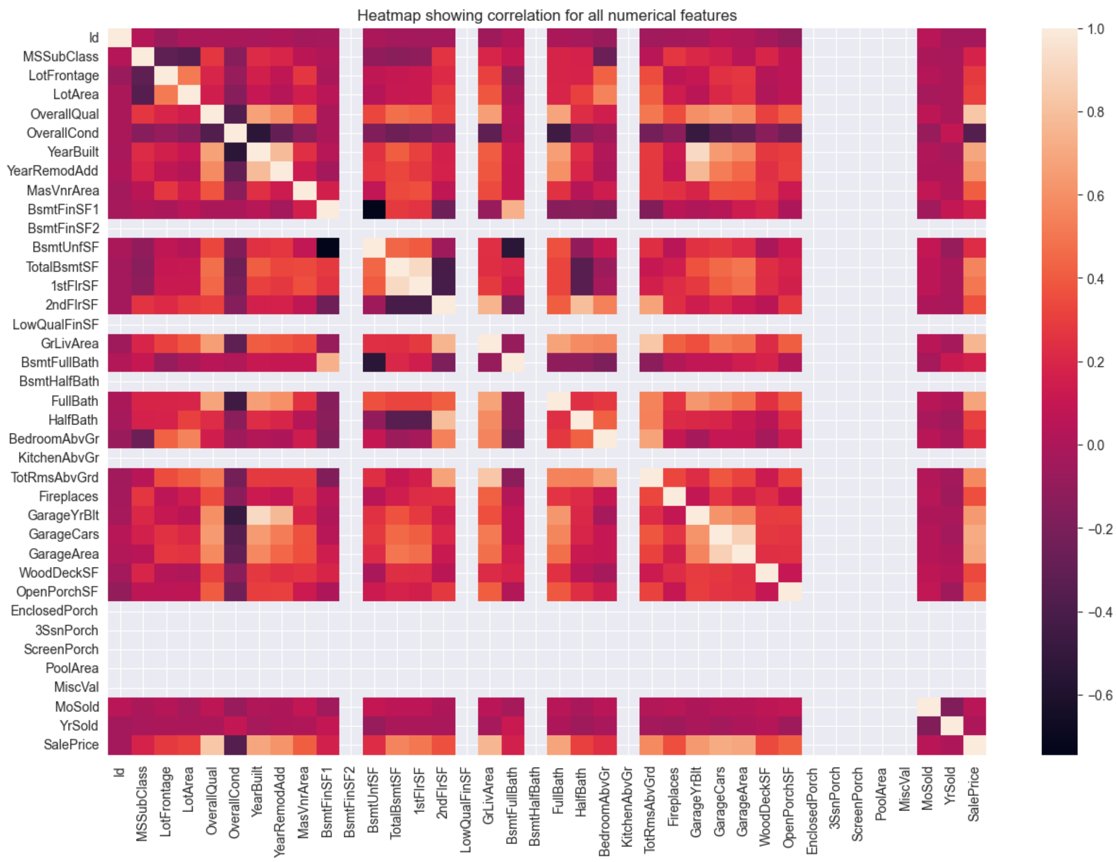

Histogram of YrSold
Frequency
0
20
40
60
80
100
120
2006.0
2006.5
2007.0
2007.5
2008.0
2008.5
2009.0
2009.5
2010.0
YrSold



5.1 Correlation between numerical features

[207]:
```
corr_matrix = new_train[train_numerical_cols].corr()
plt.figure(figsize = (15,10))
plt.title("Heatmap showing correlation for all numerical features")
sns.heatmap(corr_matrix)
```

[207]:
```
<Axes: title={'center': 'Heatmap showing correlation for all numerical
features'}>
```



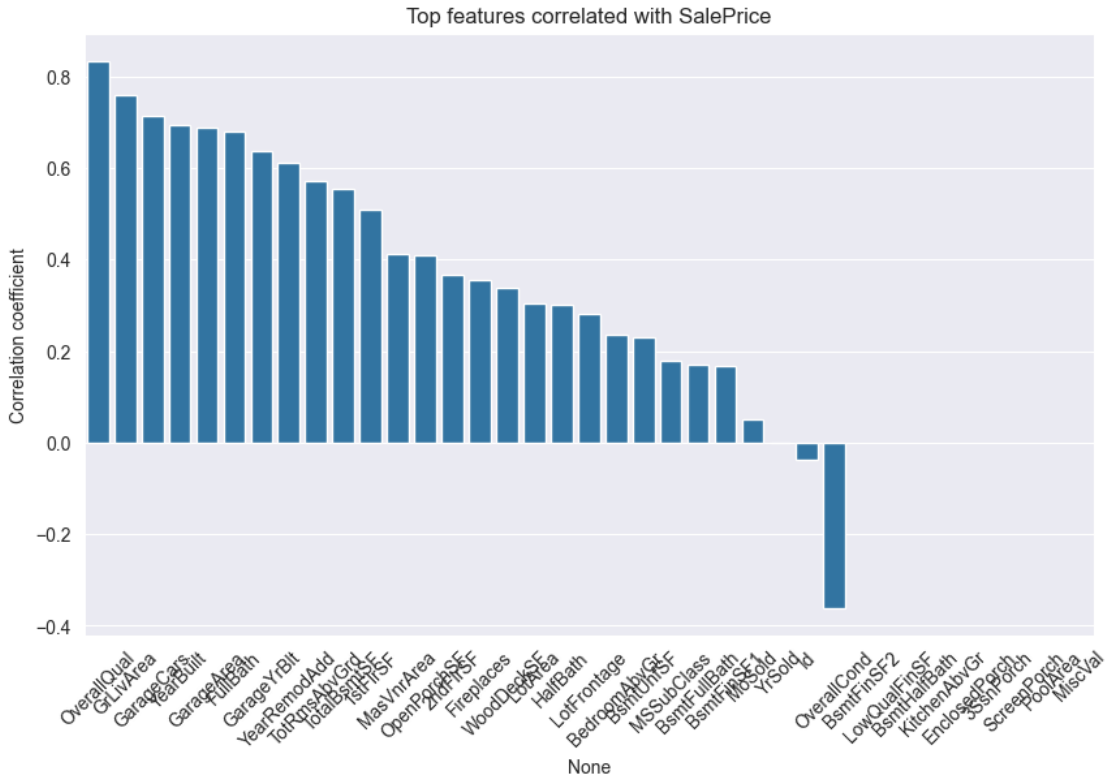
6 Selecting relevant columns to the target

Based on the insights gained from our preprocessing steps, we proceed to select the most relevant and useful features for our analysis.

[208]:
```
# Calculate the correlation matrix
corr_matrix = new_train[train_numerical_cols].corr()

# Extract the correlations with 'SalePrice'
saleprice_corr = corr_matrix["SalePrice"].drop("SalePrice")  # Exclude␣
↪self-correlation

# Sort the correlations and get the highest ones
top_correlations = saleprice_corr.sort_values(ascending=False)

# Display the top correlations with 'SalePrice'
print("Top features correlated with 'SalePrice':")
print(top_correlations)
```

```python
# Optionally, plot the top correlations
plt.figure(figsize=(10, 6))
sns.barplot(x=top_correlations.index, y=top_correlations.values)
plt.title("Top features correlated with SalePrice")
plt.ylabel("Correlation coefficient")
plt.xticks(rotation=45)
plt.show()
```

```
Top features correlated with 'SalePrice':
OverallQual      0.834155
GrLivArea        0.758951
GarageCars       0.712544
YearBuilt        0.692705
GarageArea       0.688153
FullBath         0.680493
GarageYrBlt      0.636252
YearRemodAdd     0.612175
TotRmsAbvGrd     0.572121
TotalBsmtSF      0.554271
1stFlrSF         0.507753
MasVnrArea       0.411090
OpenPorchSF      0.409521
2ndFlrSF         0.365972
Fireplaces       0.356323
WoodDeckSF       0.338591
LotArea          0.302663
HalfBath         0.300508
LotFrontage      0.281854
BedroomAbvGr     0.234447
BsmtUnfSF        0.229988
MSSubClass       0.178116
BsmtFullBath     0.168721
BsmtFinSF1       0.166024
MoSold           0.051561
YrSold           0.000136
Id              -0.036886
OverallCond     -0.361252
BsmtFinSF2            NaN
LowQualFinSF          NaN
BsmtHalfBath          NaN
KitchenAbvGr          NaN
EnclosedPorch         NaN
3SsnPorch             NaN
ScreenPorch           NaN
PoolArea              NaN
MiscVal               NaN
Name: SalePrice, dtype: float64
```



[209]:
```
cdf=['OverallQual','GrLivArea','GarageCars','YearBuilt','GarageArea','FullBath','SalePrice']
cdf_train = new_train[cdf]
```

[210]:
```
cdf_train
```

[210]:
```
     OverallQual  GrLivArea  GarageCars  YearBuilt  GarageArea  FullBath  \
0              7       1710           2       2003         548         2
1              7       1786           2       2001         608         2
2              8       2198           3       2000         836         2
3              8       1694           2       2004         636         2
4              5       1040           1       1965         384         1
..           …          …         …         …          …        …
543            7       1422           2       2004         626         2
544            4       1346           1       1910         384         1
545            8       1578           3       2008         840         2
546            7       1221           2       2004         400         2
547            6       1647           2       1999         460         2

     SalePrice
0       208500
1       223500
2       250000
```

```
3       307000
4       129500
..         ...
543     179600
544     112000
545     287090
546     185000
547     175000

[548 rows x 7 columns]
```

6.1 Perform Scaling

[211]:
```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
cdf_train= pd.DataFrame(scaler.fit_transform(cdf_train), columns=cdf_train.
 ↪columns)
```

[212]:
```python
cdf_train
```

[212]:
```
     OverallQual  GrLivArea  GarageCars  YearBuilt  GarageArea  FullBath  \
0          0.625   0.595506    0.666667   0.939394    0.590517       0.5
1          0.625   0.631086    0.666667   0.919192    0.655172       0.5
2          0.750   0.823970    1.000000   0.909091    0.900862       0.5
3          0.750   0.588015    0.666667   0.949495    0.685345       0.5
4          0.375   0.281835    0.333333   0.555556    0.413793       0.0
..           ...        ...         ...        ...    ...         ...
543        0.625   0.460674    0.666667   0.949495    0.674569       0.5
544        0.250   0.425094    0.333333   0.000000    0.413793       0.0
545        0.750   0.533708    1.000000   0.989899    0.905172       0.5
546        0.625   0.366573    0.666667   0.949495    0.431034       0.5
547        0.500   0.566011    0.666667   0.898990    0.495690       0.5

     SalePrice
0     0.586254
1     0.637801
2     0.728866
3     0.924742
4     0.314777
..         ...
543   0.486942
544   0.254639
545   0.856323
546   0.505498
547   0.471134

[548 rows x 7 columns]
```

6.2 Splitting the cdf data into train and test

[213]:
```
X = cdf_train.drop(columns = 'SalePrice')
y = cdf_train['SalePrice']
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.
 ↪3,random_state=42)
```

6.3 Applying Linear Regression Algorithm :-

6.3.1 Train our Model

[214]:
```
model = LinearRegression()
model.fit(train_X, train_y)
```

[214]:
```
LinearRegression()
```

6.3.2 Test our Model

[215]:
```
# Make predictions on the test set
y_pred = model.predict(test_X)
```

6.3.3 Evaluate our Model using MSE (Mean Square Error)

[216]:
```
mse = mean_squared_error(test_y, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 0.0059097865420728295
```

6.3.4 Model Coefficients :-

[217]:
```
print("Model Coefficients:")
for feature, coef in zip(X.columns, model.coef_):
    print(f"{feature}: {coef}")
```

```
Model Coefficients:
OverallQual: 0.4504240293662034
GrLivArea: 0.37680760996465845
GarageCars: -0.013291787683437445
YearBuilt: 0.1820058822350239
GarageArea: 0.2194846551530315
FullBath: -0.06857731890433652
```

7 Function to take user inputs and make predictions

[221]:
```python
def predict_sale_price(model, feature_names):
    user_input = {}
    for feature in feature_names:
        while True:
            value = input(f"Enter value for {feature}: ")
            try:
                # Try converting input to float
                value = float(value)
                user_input[feature] = value
                break  # Break the loop if conversion succeeds
            except ValueError:
                print("Please enter a numeric value.")

    # Convert user input to DataFrame
    input_df = pd.DataFrame([user_input])

    # Predict the sale price
    predicted_price = model.predict(input_df)

    return user_input, predicted_price[0]  # Return both user input and␣
 ↪predicted price

# Get user inputs and predict the sale price
user_input, predicted_price = predict_sale_price(model, X.columns)

# Print user input and predicted sale price
print("\nUser Input:")
for feature, value in user_input.items():
    print(f"{feature}: {value}")
print(f"\nPredicted Sale Price: {predicted_price}")
```

```
User Input:
OverallQual: 90.0
GrLivArea: 550.0
GarageCars: 2.0
YearBuilt: 1996.0
GarageArea: 225.0
FullBath: 3.0

Predicted Sale Price: 660.0674678254483
```