```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler,
OneHotEncoder,LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression as
LR,LogisticRegression
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_squared_error,mean_absolute_error
import joblib as jb

#Load House Price Dataset &train
hp_train=pd.read_csv("/Users/ishimwecharleshagenimana/Desktop/Project_
DS_FINAL/train.csv")
hp_train
```

```
        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
...    ...         ...      ...          ...      ...    ...   ...
...
1455  1456          60       RL         62.0     7917   Pave   NaN
Reg
1456  1457          20       RL         85.0    13175   Pave   NaN
Reg
1457  1458          70       RL         66.0     9042   Pave   NaN
Reg
1458  1459          20       RL         68.0     9717   Pave   NaN
Reg
1459  1460          20       RL         75.0     9937   Pave   NaN
Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature
```

```
MiscVal  \
0            Lvl    AllPub  ...        0    NaN    NaN          NaN
0
1            Lvl    AllPub  ...        0    NaN    NaN          NaN
0
2            Lvl    AllPub  ...        0    NaN    NaN          NaN
0
3            Lvl    AllPub  ...        0    NaN    NaN          NaN
0
4            Lvl    AllPub  ...        0    NaN    NaN          NaN
0
...          ...       ...  ...      ...    ...    ...          ...
...
1455         Lvl    AllPub  ...        0    NaN    NaN          NaN
0
1456         Lvl    AllPub  ...        0    NaN  MnPrv          NaN
0
1457         Lvl    AllPub  ...        0    NaN  GdPrv         Shed
2500
1458         Lvl    AllPub  ...        0    NaN    NaN          NaN
0
1459         Lvl    AllPub  ...        0    NaN    NaN          NaN
0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
0         2   2008        WD         Normal     208500
1         5   2007        WD         Normal     181500
2         9   2008        WD         Normal     223500
3         2   2006        WD        Abnorml     140000
4        12   2008        WD         Normal     250000
...     ...    ...       ...            ...        ...
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[1460 rows x 81 columns]
```

```
# Load hp dataset test
hp_test=pd.read_csv("/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv")
hp_test
```

```
        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
...    ...         ...      ...          ...      ...    ...   ...
...
1454  2915         160       RM         21.0     1936   Pave   NaN
Reg
1455  2916         160       RM         21.0     1894   Pave   NaN
Reg
1456  2917          20       RL        160.0    20000   Pave   NaN
Reg
1457  2918          85       RL         62.0    10441   Pave   NaN
Reg
1458  2919          60       RL         74.0     9627   Pave   NaN
Reg

     LandContour Utilities  ... ScreenPorch PoolArea PoolQC  Fence  \
0            Lvl    AllPub  ...         120        0    NaN  MnPrv
1            Lvl    AllPub  ...           0        0    NaN    NaN
2            Lvl    AllPub  ...           0        0    NaN  MnPrv
3            Lvl    AllPub  ...           0        0    NaN    NaN
4            HLS    AllPub  ...         144        0    NaN    NaN
...          ...       ...  ...         ...      ...    ...    ...
1454         Lvl    AllPub  ...           0        0    NaN    NaN
1455         Lvl    AllPub  ...           0        0    NaN    NaN
1456         Lvl    AllPub  ...           0        0    NaN    NaN
1457         Lvl    AllPub  ...           0        0    NaN  MnPrv
1458         Lvl    AllPub  ...           0        0    NaN    NaN

     MiscFeature MiscVal MoSold  YrSold  SaleType  SaleCondition
0            NaN       0      6    2010        WD         Normal
1           Gar2   12500      6    2010        WD         Normal
2            NaN       0      3    2010        WD         Normal
3            NaN       0      6    2010        WD         Normal
4            NaN       0      1    2010        WD         Normal
...          ...     ...    ...     ...       ...            ...
1454         NaN       0      6    2006        WD         Normal
1455         NaN       0      4    2006        WD        Abnorml
1456         NaN       0      9    2006        WD        Abnorml
1457        Shed     700      7    2006        WD         Normal
1458         NaN       0     11    2006        WD         Normal

[1459 rows x 80 columns]
```

```
# Add a column to differentiate between train and test data
hp_train["/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/
train.csv"] = "train"
```

```
hp_test["/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/
test.csv"] = "test"
# Concatenate both datasets
hp_combined_train_test = pd.concat([hp_train, hp_test], axis=0,
ignore_index=True)
# Check the new shape for the dataset
print(hp_combined_train_test.shape)
```

```
(2919, 83)
```

```
hp_combined_train_test.head()
```

```
   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  ... Fence MiscFeature MiscVal MoSold YrSold
SaleType  \
0         Lvl    AllPub  ...   NaN         NaN       0      2   2008
WD
1         Lvl    AllPub  ...   NaN         NaN       0      5   2007
WD
2         Lvl    AllPub  ...   NaN         NaN       0      9   2008
WD
3         Lvl    AllPub  ...   NaN         NaN       0      2   2006
WD
4         Lvl    AllPub  ...   NaN         NaN       0     12   2008
WD

  SaleCondition  SalePrice  \
0        Normal   208500.0
1        Normal   181500.0
2        Normal   223500.0
3       Abnorml   140000.0
4        Normal   250000.0

   /Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/train.csv
\
0                                                train

1                                                train
```

```
2                                                   train

3                                                   train

4                                                   train


   /Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv
0                                                   NaN
1                                                   NaN
2                                                   NaN
3                                                   NaN
4                                                   NaN

[5 rows x 83 columns]

hp_combined_train_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 83 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Id
2919 non-null   int64
 1   MSSubClass
2919 non-null   int64
 2   MSZoning
2915 non-null   object
 3   LotFrontage
2433 non-null   float64
 4   LotArea
2919 non-null   int64
 5   Street
2919 non-null   object
 6   Alley
198 non-null    object
 7   LotShape
2919 non-null   object
 8   LandContour
2919 non-null   object
 9   Utilities
2917 non-null   object
 10  LotConfig
2919 non-null   object
 11  LandSlope
2919 non-null   object
 12  Neighborhood
```

```
2919 non-null   object
 13  Condition1
2919 non-null   object
 14  Condition2
2919 non-null   object
 15  BldgType
2919 non-null   object
 16  HouseStyle
2919 non-null   object
 17  OverallQual
2919 non-null   int64
 18  OverallCond
2919 non-null   int64
 19  YearBuilt
2919 non-null   int64
 20  YearRemodAdd
2919 non-null   int64
 21  RoofStyle
2919 non-null   object
 22  RoofMatl
2919 non-null   object
 23  Exterior1st
2918 non-null   object
 24  Exterior2nd
2918 non-null   object
 25  MasVnrType
1153 non-null   object
 26  MasVnrArea
2896 non-null   float64
 27  ExterQual
2919 non-null   object
 28  ExterCond
2919 non-null   object
 29  Foundation
2919 non-null   object
 30  BsmtQual
2838 non-null   object
 31  BsmtCond
2837 non-null   object
 32  BsmtExposure
2837 non-null   object
 33  BsmtFinType1
2840 non-null   object
 34  BsmtFinSF1
2918 non-null   float64
 35  BsmtFinType2
2839 non-null   object
 36  BsmtFinSF2
2918 non-null   float64
```

```
 37  BsmtUnfSF
2918 non-null   float64
 38  TotalBsmtSF
2918 non-null   float64
 39  Heating
2919 non-null   object
 40  HeatingQC
2919 non-null   object
 41  CentralAir
2919 non-null   object
 42  Electrical
2918 non-null   object
 43  1stFlrSF
2919 non-null   int64
 44  2ndFlrSF
2919 non-null   int64
 45  LowQualFinSF
2919 non-null   int64
 46  GrLivArea
2919 non-null   int64
 47  BsmtFullBath
2917 non-null   float64
 48  BsmtHalfBath
2917 non-null   float64
 49  FullBath
2919 non-null   int64
 50  HalfBath
2919 non-null   int64
 51  BedroomAbvGr
2919 non-null   int64
 52  KitchenAbvGr
2919 non-null   int64
 53  KitchenQual
2918 non-null   object
 54  TotRmsAbvGrd
2919 non-null   int64
 55  Functional
2917 non-null   object
 56  Fireplaces
2919 non-null   int64
 57  FireplaceQu
1499 non-null   object
 58  GarageType
2762 non-null   object
 59  GarageYrBlt
2760 non-null   float64
 60  GarageFinish
2760 non-null   object
 61  GarageCars
```

```
2918 non-null   float64
 62  GarageArea
2918 non-null   float64
 63  GarageQual
2760 non-null   object
 64  GarageCond
2760 non-null   object
 65  PavedDrive
2919 non-null   object
 66  WoodDeckSF
2919 non-null   int64
 67  OpenPorchSF
2919 non-null   int64
 68  EnclosedPorch
2919 non-null   int64
 69  3SsnPorch
2919 non-null   int64
 70  ScreenPorch
2919 non-null   int64
 71  PoolArea
2919 non-null   int64
 72  PoolQC
10 non-null     object
 73  Fence
571 non-null    object
 74  MiscFeature
105 non-null    object
 75  MiscVal
2919 non-null   int64
 76  MoSold
2919 non-null   int64
 77  YrSold
2919 non-null   int64
 78  SaleType
2918 non-null   object
 79  SaleCondition
2919 non-null   object
 80  SalePrice
1460 non-null   float64
 81
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/train.csv
1460 non-null   object
 82  /Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv
1459 non-null   object
dtypes: float64(12), int64(26), object(45)
memory usage: 1.8+ MB

#Checking data types in hp dataset
hp_combined_train_test.dtypes
```

```
Id
int64
MSSubClass
int64
MSZoning
object
LotFrontage
float64
LotArea
int64

...
SaleType
object
SaleCondition
object
SalePrice
float64
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/train.csv
object
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv
object
Length: 83, dtype: object
```

```
# All Columns i have in my test&train
hp_combined_train_test.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea',
'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation',
'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF',
'2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu',
'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea',
'GarageQual',
```

```
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold',
'SaleType',
       'SaleCondition', 'SalePrice',

'/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/train.csv',

'/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv'],
      dtype='object')
```

```
#numerical features hp dataset
numerical_features
=hp_combined_train_test.select_dtypes(include=[np.number]).columns.tol
ist()
print("Numerical Columns hp:", numerical_features)
```

```
Numerical Columns hp: ['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']
```

```
#Verifing statistical Summary in my dataset
hp_combined_train_test.describe().T
```

```
                count           mean           std      min       25%
\
Id             2919.0    1460.000000    842.787043      1.0     730.5

MSSubClass     2919.0      57.137718     42.517628     20.0      20.0

LotFrontage    2433.0      69.305795     23.344905     21.0      59.0

LotArea        2919.0   10168.114080   7886.996359   1300.0    7478.0

OverallQual    2919.0       6.089072      1.409947      1.0       5.0

OverallCond    2919.0       5.564577      1.113131      1.0       5.0

YearBuilt      2919.0    1971.312778     30.291442   1872.0    1953.5

YearRemodAdd   2919.0    1984.264474     20.894344   1950.0    1965.0

MasVnrArea     2896.0     102.201312    179.334253      0.0       0.0
```

BsmtFinSF1	2918.0	441.423235	455.610826	0.0	0.0
BsmtFinSF2	2918.0	49.582248	169.205611	0.0	0.0
BsmtUnfSF	2918.0	560.772104	439.543659	0.0	220.0
TotalBsmtSF	2918.0	1051.777587	440.766258	0.0	793.0
1stFlrSF	2919.0	1159.581706	392.362079	334.0	876.0
2ndFlrSF	2919.0	336.483727	428.701456	0.0	0.0
LowQualFinSF	2919.0	4.694416	46.396825	0.0	0.0
GrLivArea	2919.0	1500.759849	506.051045	334.0	1126.0
BsmtFullBath	2917.0	0.429894	0.524736	0.0	0.0
BsmtHalfBath	2917.0	0.061364	0.245687	0.0	0.0
FullBath	2919.0	1.568003	0.552969	0.0	1.0
HalfBath	2919.0	0.380267	0.502872	0.0	0.0
BedroomAbvGr	2919.0	2.860226	0.822693	0.0	2.0
KitchenAbvGr	2919.0	1.044536	0.214462	0.0	1.0
TotRmsAbvGrd	2919.0	6.451524	1.569379	2.0	5.0
Fireplaces	2919.0	0.597122	0.646129	0.0	0.0
GarageYrBlt	2760.0	1978.113406	25.574285	1895.0	1960.0
GarageCars	2918.0	1.766621	0.761624	0.0	1.0
GarageArea	2918.0	472.874572	215.394815	0.0	320.0
WoodDeckSF	2919.0	93.709832	126.526589	0.0	0.0
OpenPorchSF	2919.0	47.486811	67.575493	0.0	0.0
EnclosedPorch	2919.0	23.098321	64.244246	0.0	0.0
3SsnPorch	2919.0	2.602261	25.188169	0.0	0.0
ScreenPorch	2919.0	16.062350	56.184365	0.0	0.0
PoolArea	2919.0	2.251799	35.663946	0.0	0.0
MiscVal	2919.0	50.825968	567.402211	0.0	0.0

```
MoSold          2919.0       6.213087      2.714762      1.0        4.0

YrSold          2919.0    2007.792737      1.314964   2006.0     2007.0

SalePrice       1460.0  180921.195890  79442.502883  34900.0  129975.0


                    50%       75%       max
Id                1460.0    2189.5    2919.0
MSSubClass          50.0      70.0     190.0
LotFrontage         68.0      80.0     313.0
LotArea           9453.0   11570.0  215245.0
OverallQual          6.0       7.0      10.0
OverallCond          5.0       6.0       9.0
YearBuilt         1973.0    2001.0    2010.0
YearRemodAdd      1993.0    2004.0    2010.0
MasVnrArea           0.0     164.0    1600.0
BsmtFinSF1         368.5     733.0    5644.0
BsmtFinSF2           0.0       0.0    1526.0
BsmtUnfSF          467.0     805.5    2336.0
TotalBsmtSF        989.5    1302.0    6110.0
1stFlrSF          1082.0    1387.5    5095.0
2ndFlrSF             0.0     704.0    2065.0
LowQualFinSF         0.0       0.0    1064.0
GrLivArea         1444.0    1743.5    5642.0
BsmtFullBath         0.0       1.0       3.0
BsmtHalfBath         0.0       0.0       2.0
FullBath             2.0       2.0       4.0
HalfBath             0.0       1.0       2.0
BedroomAbvGr         3.0       3.0       8.0
KitchenAbvGr         1.0       1.0       3.0
TotRmsAbvGrd         6.0       7.0      15.0
Fireplaces           1.0       1.0       4.0
GarageYrBlt       1979.0    2002.0    2207.0
GarageCars           2.0       2.0       5.0
GarageArea         480.0     576.0    1488.0
WoodDeckSF           0.0     168.0    1424.0
OpenPorchSF         26.0      70.0     742.0
EnclosedPorch        0.0       0.0    1012.0
3SsnPorch            0.0       0.0     508.0
ScreenPorch          0.0       0.0     576.0
PoolArea             0.0       0.0     800.0
MiscVal              0.0       0.0   17000.0
MoSold               6.0       8.0      12.0
YrSold            2008.0    2009.0    2010.0
SalePrice       163000.0  214000.0  755000.0

# Check for missing values hp data
# Returns the number of missing values in each column
```

```
missing_data_hp= hp_combined_train_test.isnull().sum()
print(missing_data_hp)
```

```
Id
0
MSSubClass
0
MSZoning
0
LotFrontage
0
LotArea
0
                                                                                .
.
SaleType
0
SaleCondition
0
SalePrice
0
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/train.csv
0
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv
0
Length: 83, dtype: int64
```

```
# Impute hp missing  values with mean,mode and median
hp_combined_train_test['MSZoning'] =
hp_combined_train_test['MSZoning'].fillna(hp_combined_train_test['MSZo
ning'].mode())
hp_combined_train_test['MSZoning'].mode
```

```
<bound method Series.mode of 0       RL
1       RL
2       RL
3       RL
4       RL
       ..
2914    RM
2915    RM
2916    RL
2917    RL
2918    RL
Name: MSZoning, Length: 2919, dtype: object>
```

```
hp_combined_train_test['LotFrontage'] =
hp_combined_train_test['LotFrontage'].fillna(hp_combined_train_test['L
otFrontage'].median())
hp_combined_train_test['LotFrontage'].median
```

```
<bound method Series.median of 0        65.0
1        80.0
2        68.0
3        60.0
4        84.0
        ...
2914     21.0
2915     21.0
2916    160.0
2917     62.0
2918     74.0
Name: LotFrontage, Length: 2919, dtype: float64>
```

```
hp_combined_train_test['SalePrice'] =
hp_combined_train_test['SalePrice'].fillna(hp_combined_train_test['Sal
ePrice'].median())
hp_combined_train_test['SalePrice'].mean
```

```
<bound method Series.mean of 0       208500.0
1       181500.0
2       223500.0
3       140000.0
4       250000.0
          ...
2914    163000.0
2915    163000.0
2916    163000.0
2917    163000.0
2918    163000.0
Name: SalePrice, Length: 2919, dtype: float64>
```

```
hp_combined_train_test['SaleType'] =
hp_combined_train_test['SaleType'].fillna(hp_combined_train_test['Sale
Type'].mode())
hp_combined_train_test['SaleType'].mode
```

```
<bound method Series.mode of 0       WD
1       WD
2       WD
3       WD
4       WD
        ..
2914    WD
2915    WD
2916    WD
2917    WD
2918    WD
Name: SaleType, Length: 2919, dtype: object>
```

```
hp_combined_train_test['LotFrontage'] =
hp_combined_train_test['LotFrontage'].fillna(hp_combined_train_test['L
otFrontage'].mode())
hp_combined_train_test['LotFrontage'].mode
```

```
<bound method Series.mode of 0        65.0
1        80.0
2        68.0
3        60.0
4        84.0
        ...
2914     21.0
2915     21.0
2916    160.0
2917     62.0
2918     74.0
Name: LotFrontage, Length: 2919, dtype: float64>
```

```
#Check at what percentage values are missing hp
perc_missing=(hp_combined_train_test.isnull().sum()/len(hp_combined_tr
ain_test))*100
print(perc_missing)
```

```
Id
0.0
MSSubClass
0.0
MSZoning
0.0
LotFrontage
0.0
LotArea
0.0
                                                                      .
..
SaleType
0.0
SaleCondition
0.0
SalePrice
0.0
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/train.csv
0.0
/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv
0.0
Length: 83, dtype: float64
```

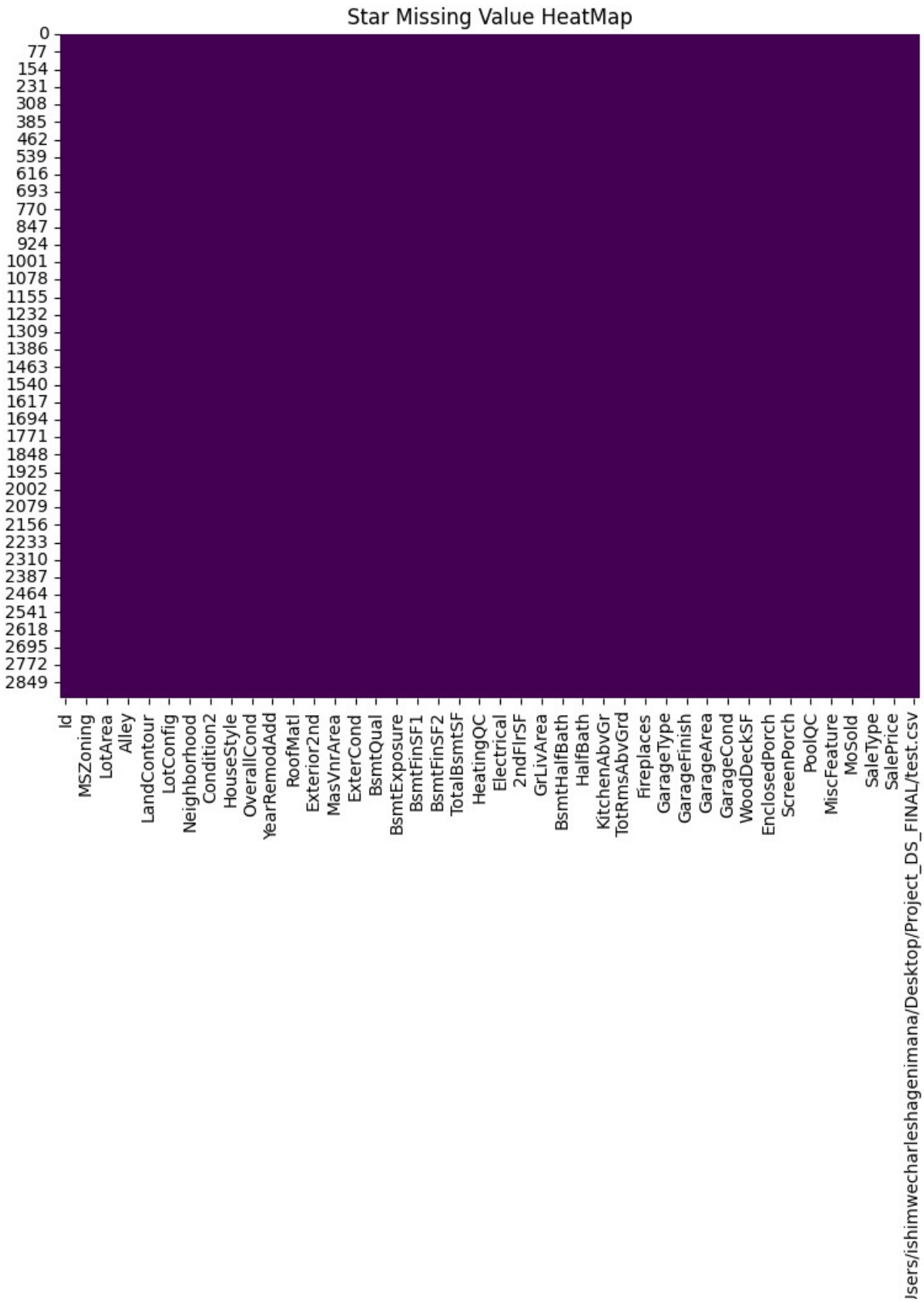
```
# Check for missing values before handling
print("Missing values before handling:\n",
hp_combined_train_test.isnull().sum().sum())
```

```
# Fill missing values
hp_combined_train_test.fillna(hp_combined_train_test.mode(),
inplace=True)  # Replace NaNs in numeric columns with mean
hp_combined_train_test.fillna("Unknown", inplace=True)  # Replace NaNs
in categorical columns with "Unknown"

# Check again to confirm no missing values
print("Missing values after handling:\n",
hp_combined_train_test.isnull().sum().sum())
```

```
Missing values before handling:
 0
Missing values after handling:
 0
```

```
#Visualise missing hp data by heatmap
plt.figure(figsize=(9,7))
sns.heatmap(hp_combined_train_test.isnull(),cbar=False ,
cmap="viridis")
plt.title("Star Missing Value HeatMap")
plt.show
```

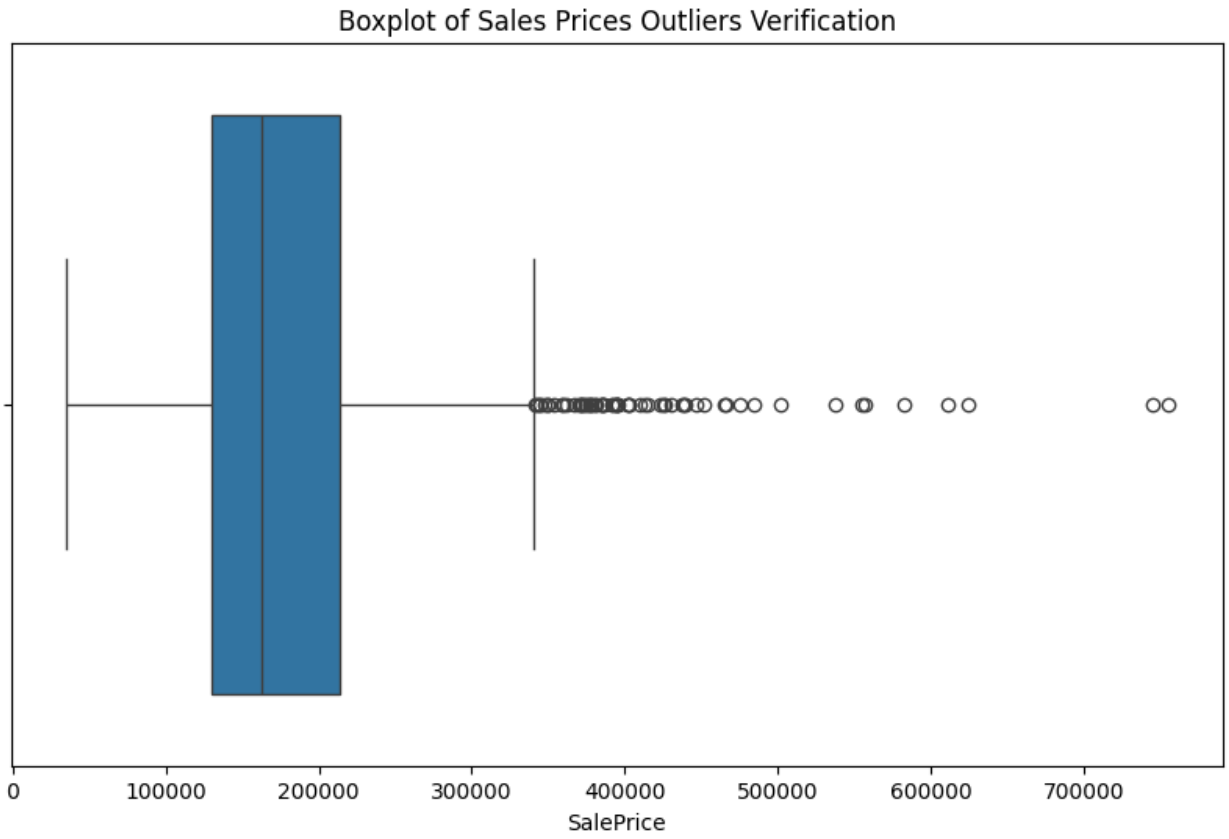
```
<function matplotlib.pyplot.show(close=None, block=None)>
```


Star Missing Value HeatMap
0
77
154
231
308
385
462
539
616
693
770
847
924
1001
1078
1155
1232
1309
1386
1463
1540
1617
1694
1771
1848
1925
2002
2079
2156
2233
2310
2387
2464
2541
2618
2695
2772
2849
Id
MSZoning
LotArea
Alley
LandContour
LotConfig
Neighborhood
Condition2
HouseStyle
OverallCond
YearRemodAdd
RoofMatl
Exterior2nd
MasVnrArea
ExterCond
BsmtQual
BsmtExposure
BsmtFinSF1
BsmtFinSF2
TotalBsmtSF
HeatingQC
Electrical
2ndFlrSF
GrLivArea
BsmtHalfBath
HalfBath
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageType
GarageFinish
GarageArea
GarageCond
WoodDeckSF
EnclosedPorch
ScreenPorch
PoolQC
MiscFeature
MoSold
SaleType
SalePrice
Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/test.csv

```
#Identify Outliers
Q1 = hp_train['SalePrice'].quantile(0.25)
Q3 = hp_train['SalePrice'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print("Q1 (25th percentile):", Q1)
print("Q3 (75th percentile):", Q3)
print("IQR (Interquartile Range):", IQR)
print("Lower Bound (Q1 - 1.5 * IQR):", lower_bound)
print("Upper Bound (Q3 + 1.5 * IQR):", upper_bound)

outliers_iqr_hp = hp_train[(hp_train['SalePrice'] < lower_bound) |
(hp_train['SalePrice'] > upper_bound)]
```

```
Q1 (25th percentile): 129975.0
Q3 (75th percentile): 214000.0
IQR (Interquartile Range): 84025.0
Lower Bound (Q1 - 1.5 * IQR): 3937.5
Upper Bound (Q3 + 1.5 * IQR): 340037.5
```

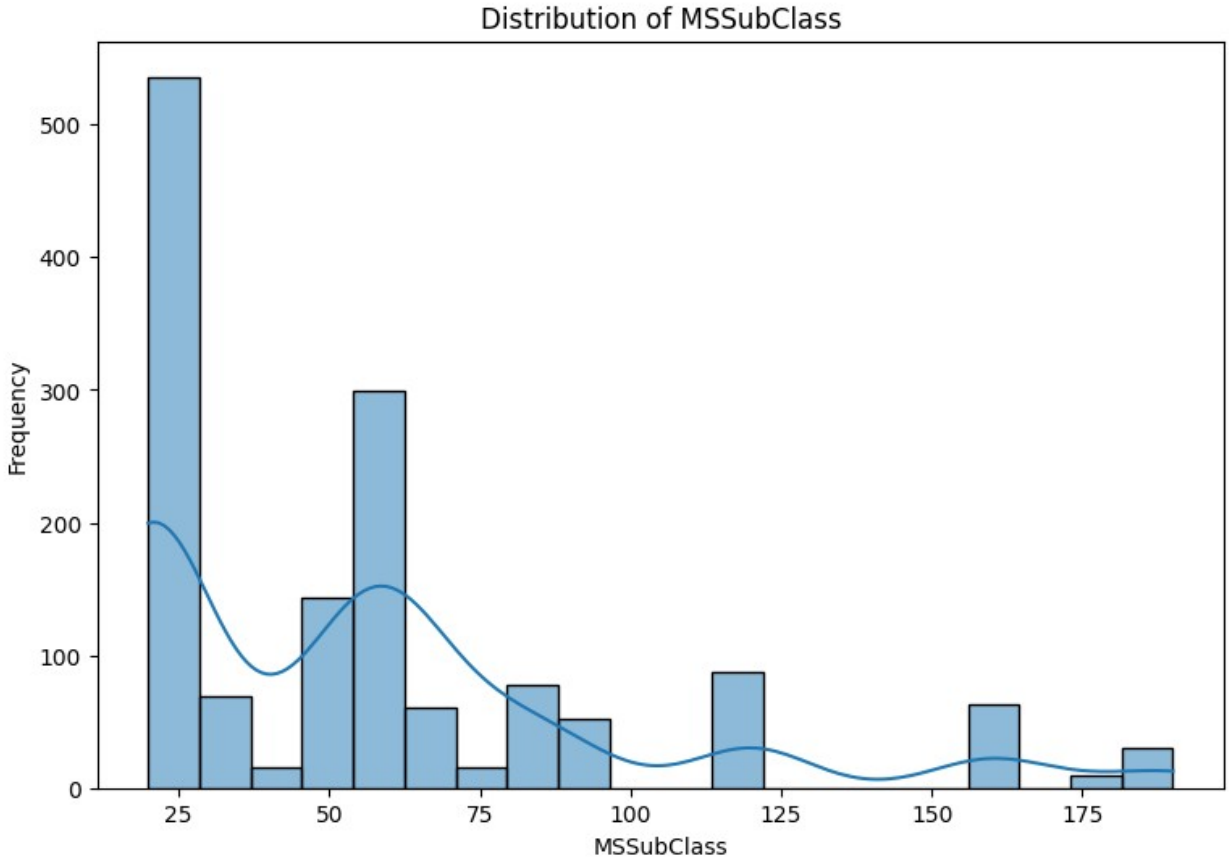
```
#Verify outliers for hp dataset by using boxplot
plt.figure(figsize=(10,6))
sns.boxplot(x=hp_train["SalePrice"])
plt.title("Boxplot of Sales Prices Outliers Verification")
plt.show()
```

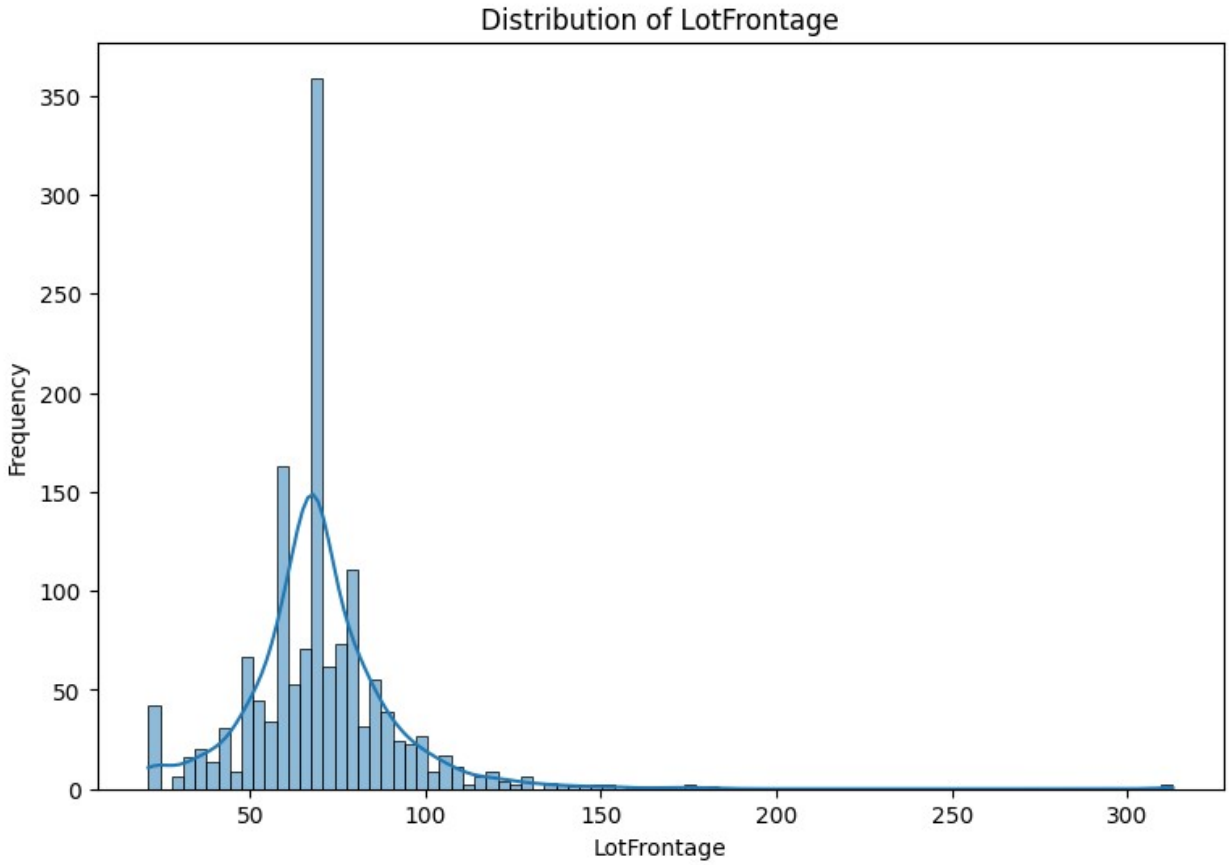


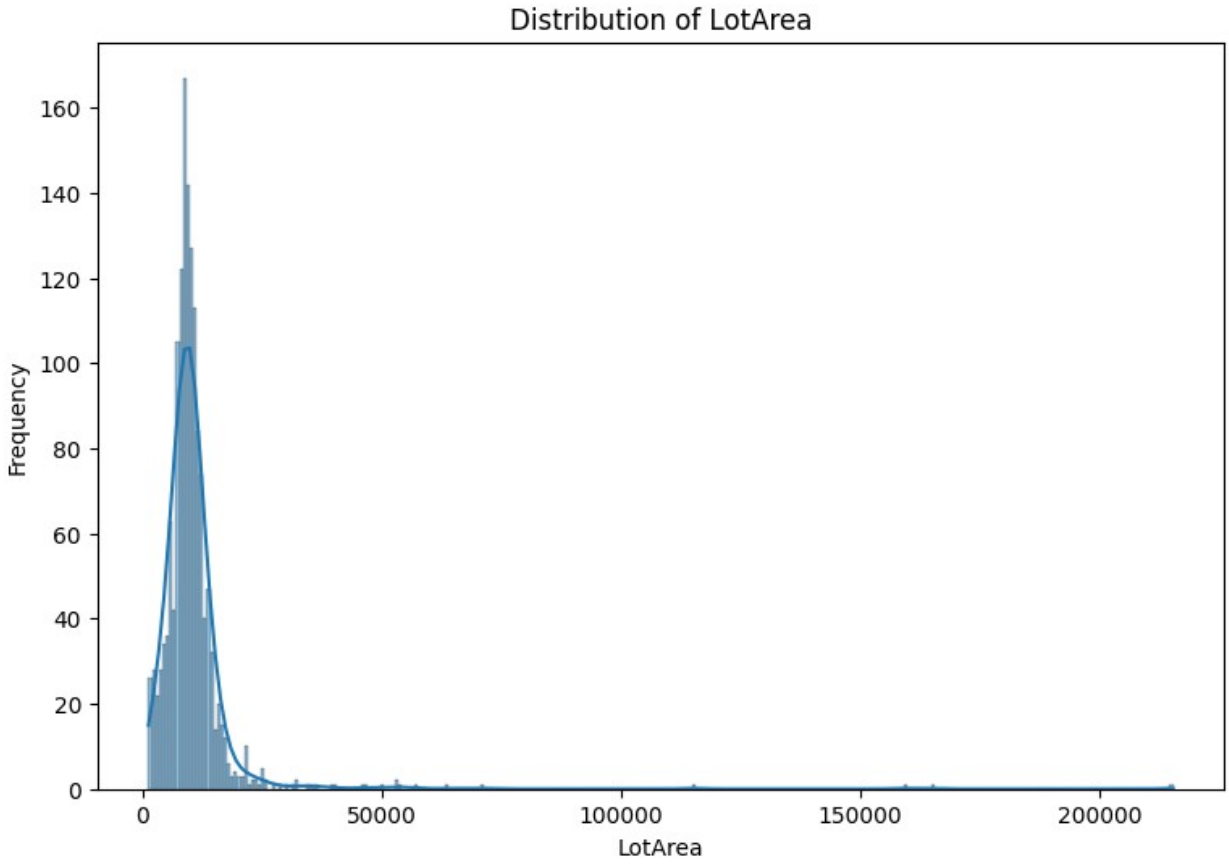

```
# Convert ALL columns with numbers stored as strings to floats
for col in hp_combined_train_test.columns:
    try:
        hp_train[col] = pd.to_numeric(hp_combined_train_test[col],
errors='coerce')  # Convert strings to numbers
    except:
        pass  # Ignore errors

# 2 Encode Categorical Columns
categorical_cols =
hp_combined_train_test.select_dtypes(include=['object']).columns  #
Find categorical columns
for col in categorical_cols:
    le = LabelEncoder()
    hp_combined_train_test[col] =
le.fit_transform(hp_combined_train_test[col].astype(str))  # Convert
categories to numeric

X_train = hp_combined_train_test.fillna(0)
X_test = hp_combined_train_test.fillna(0)

#Checking if i still have duplicates for hp data
diplicate_hp_data=hp_combined_train_test.duplicated()
print(diplicate_hp_data.sum())
```

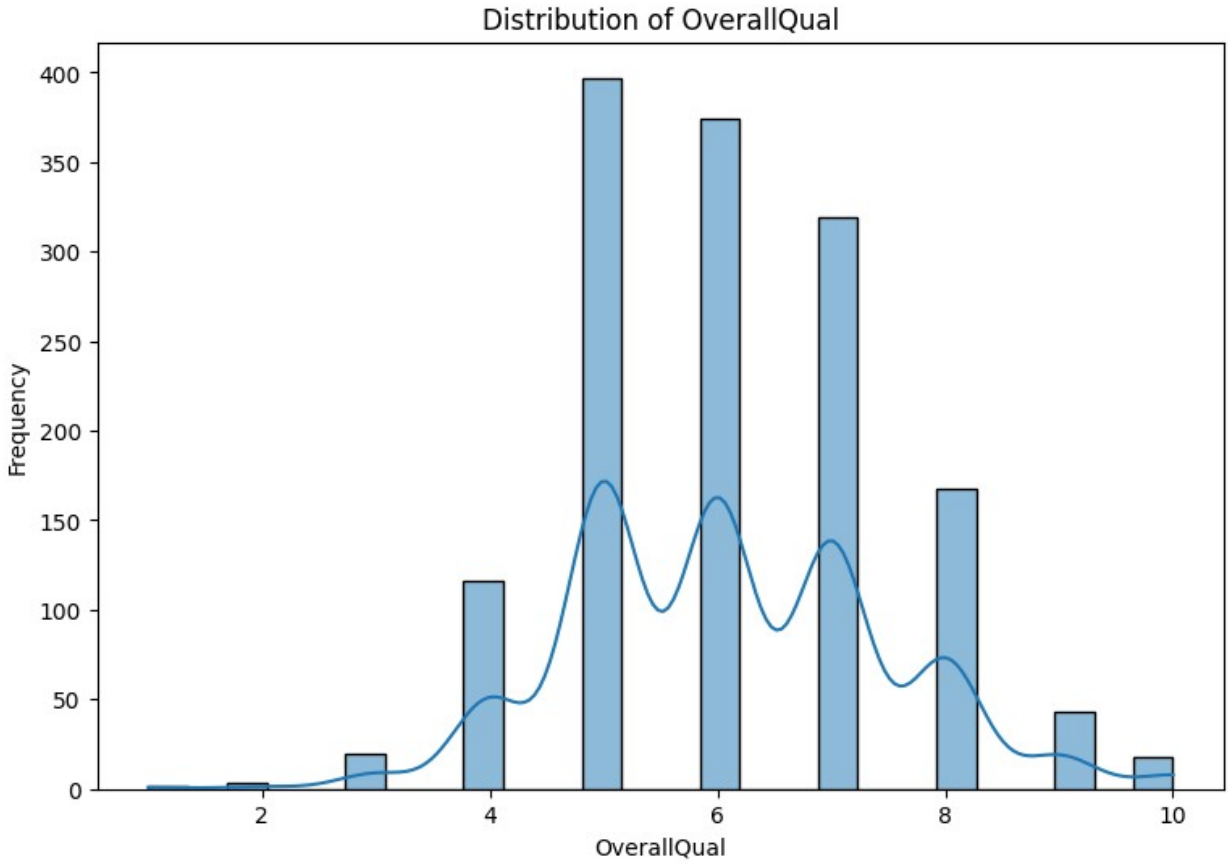
```
0
```

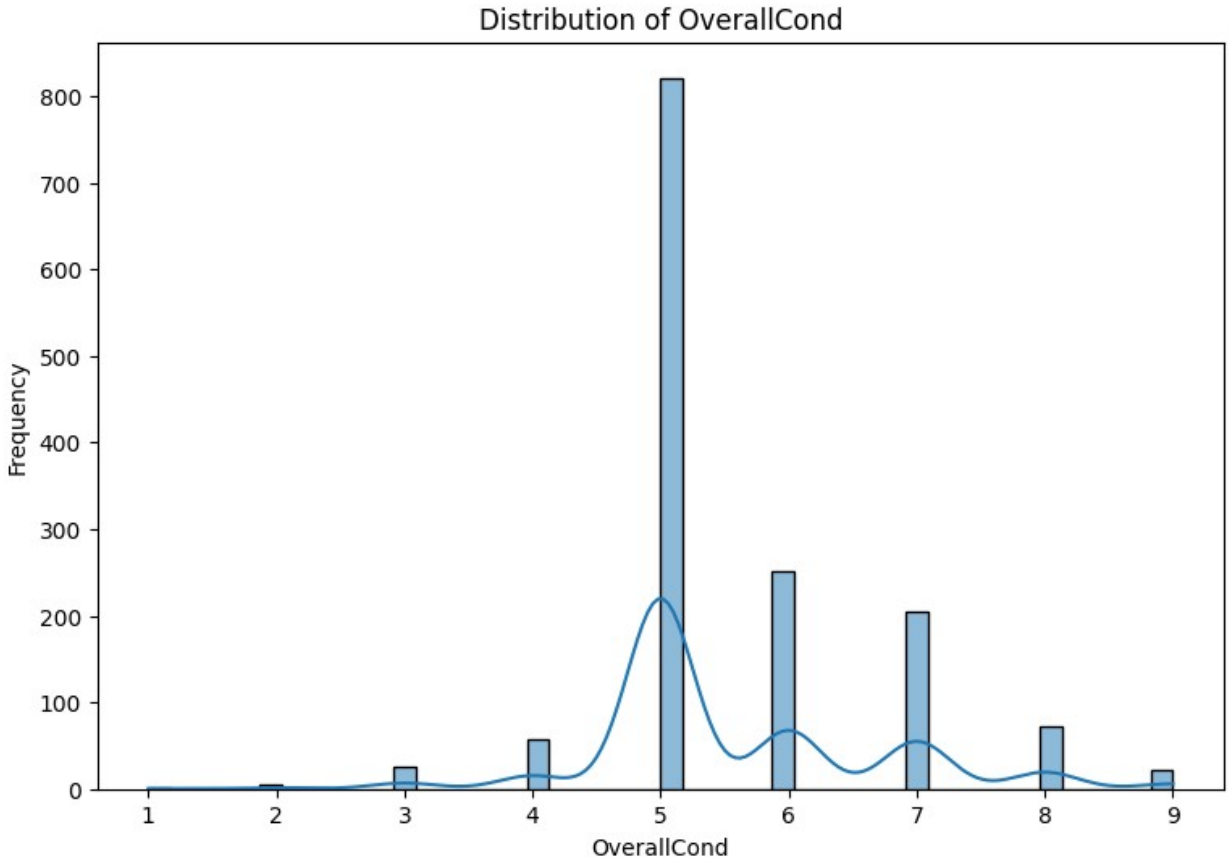
```
# Distribution of numerical features hp
numerical_columns_hp =['MSSubClass', 'LotFrontage', 'LotArea',
'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
'1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt',
'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
'MoSold', 'YrSold', 'SalePrice']
for col in numerical_columns_hp:
    plt.figure(figsize=(9,6))
    sns.histplot(hp_train[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```

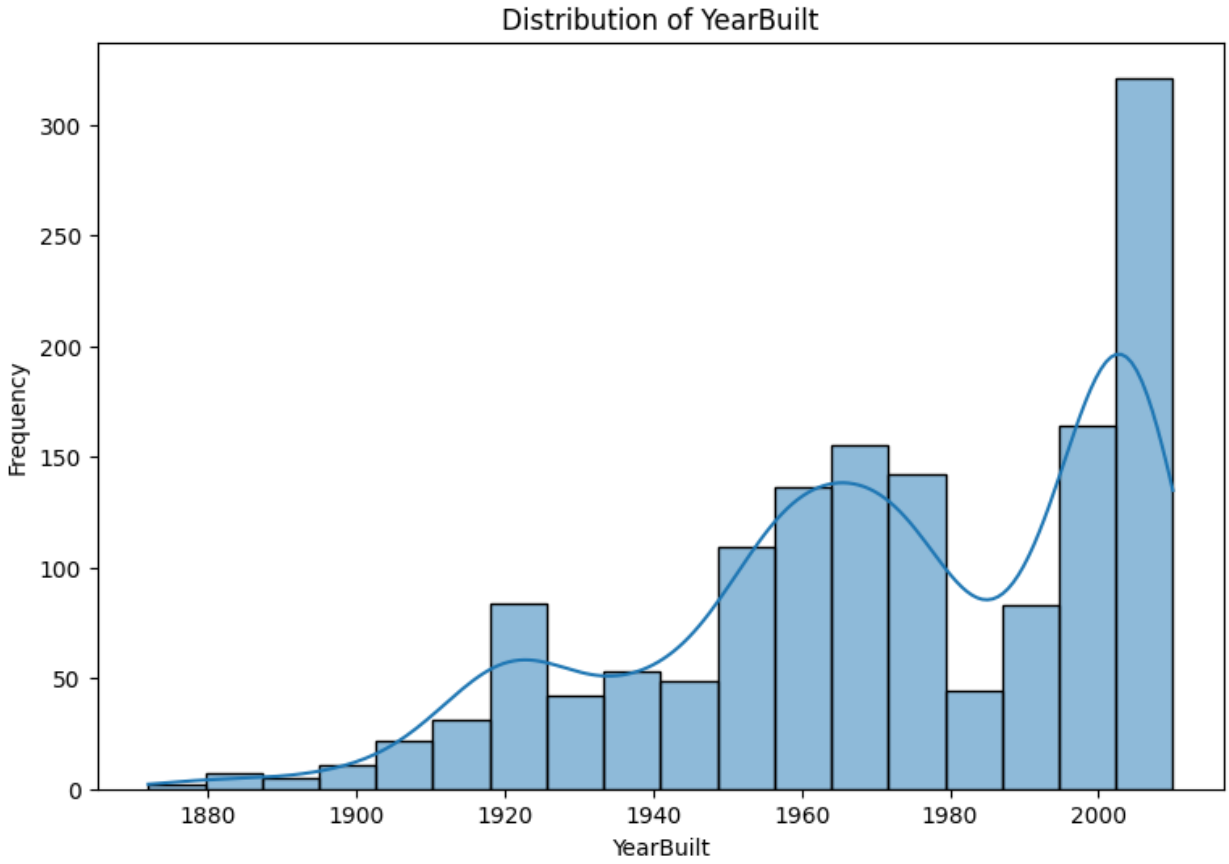


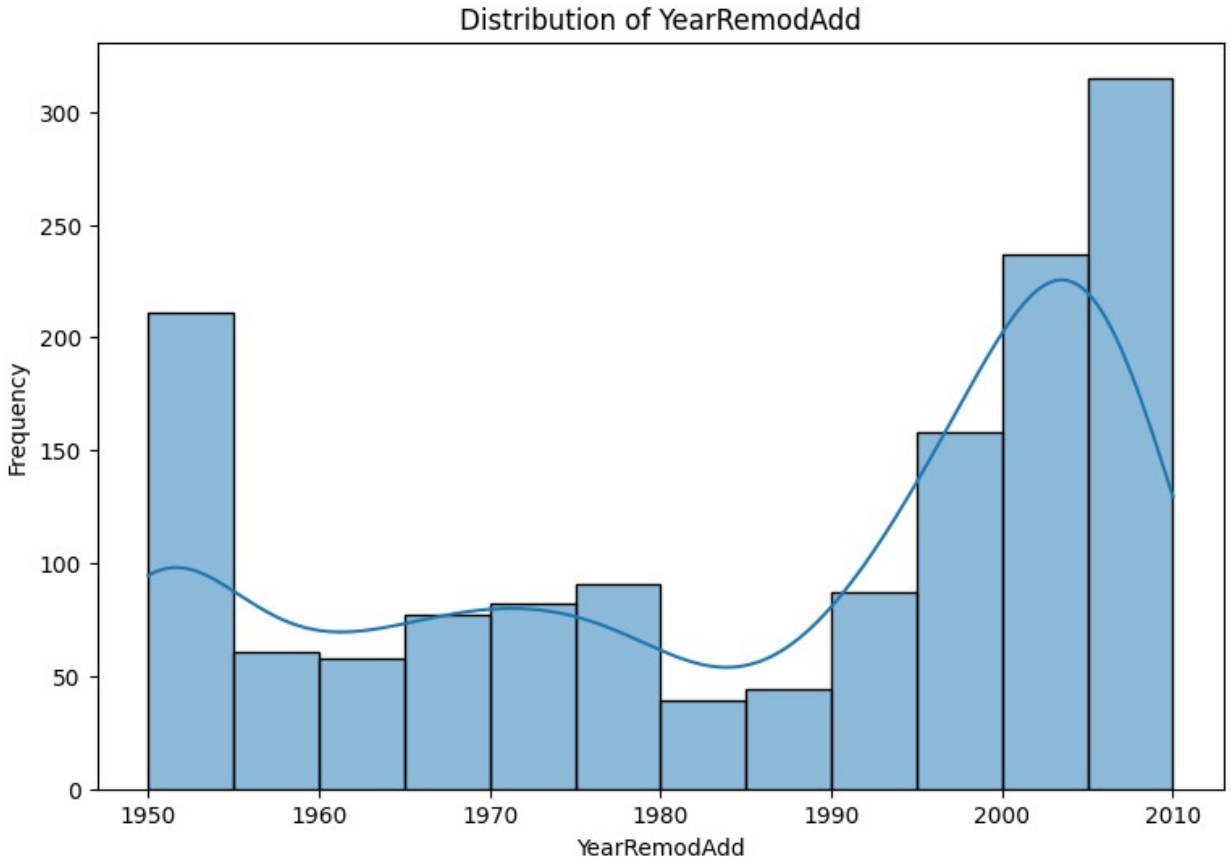

Distribution of LotFrontage
Frequency
LotFrontage
0
50
100
150
200
250
300
350

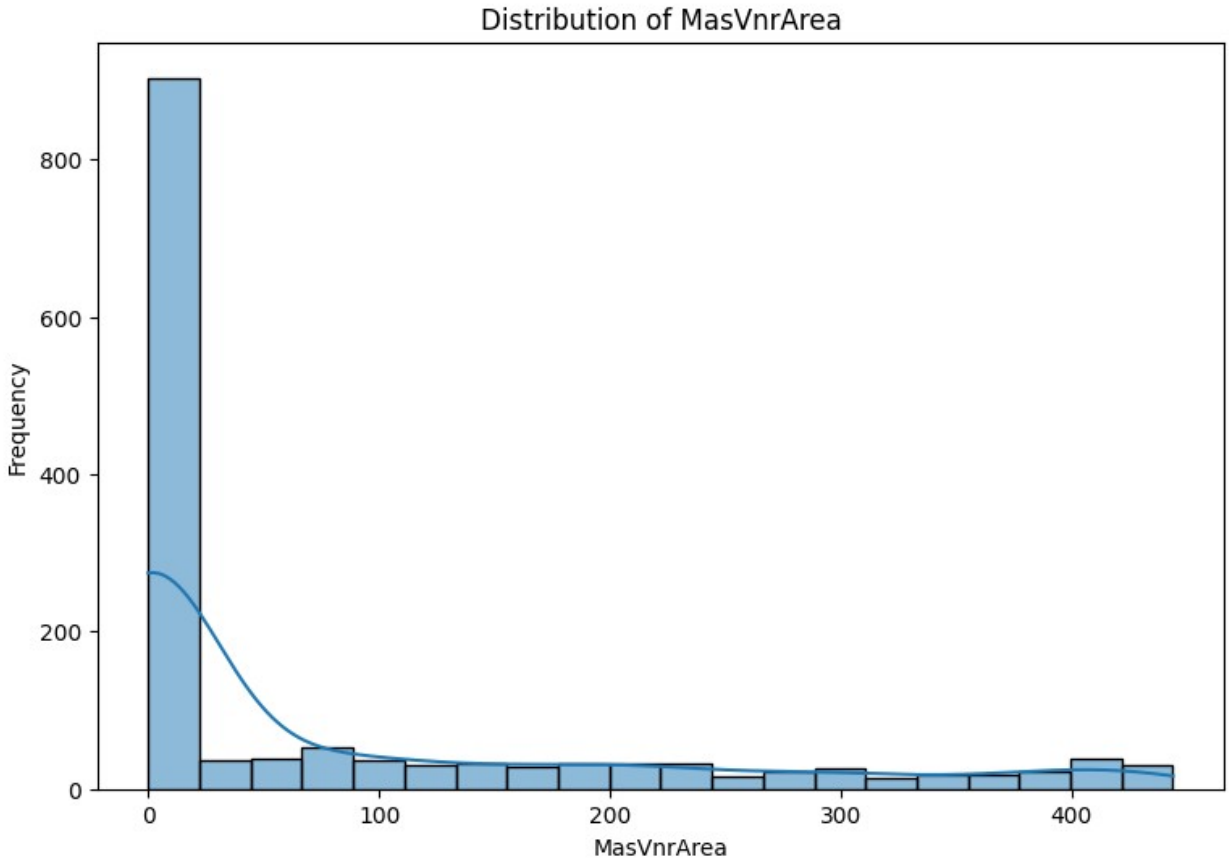

Distribution of LotArea
Frequency
LotArea
0
20
40
60
80
100
120
140
160
0
50000
100000
150000
200000

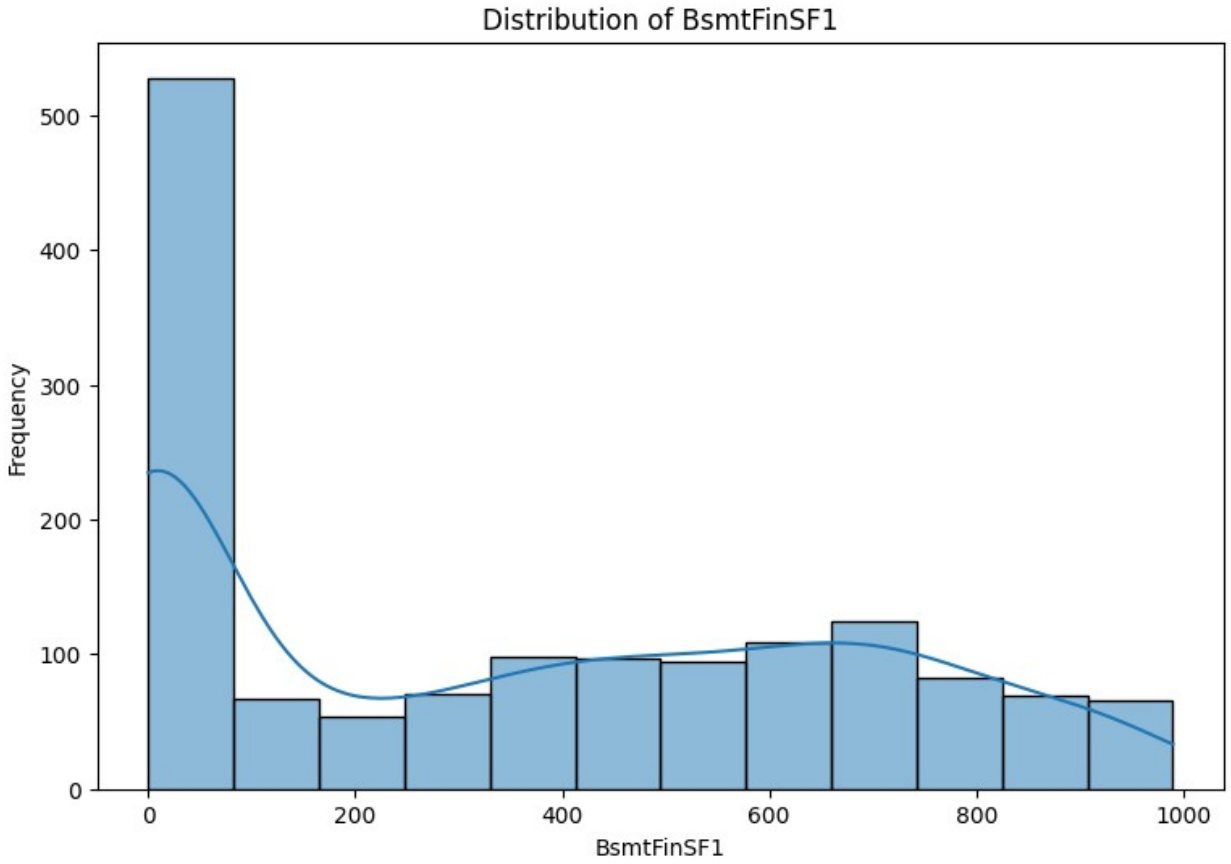

Distribution of OverallQual
Frequency
OverallQual
0
50
100
150
200
250
300
350
400
2
4
6
8
10

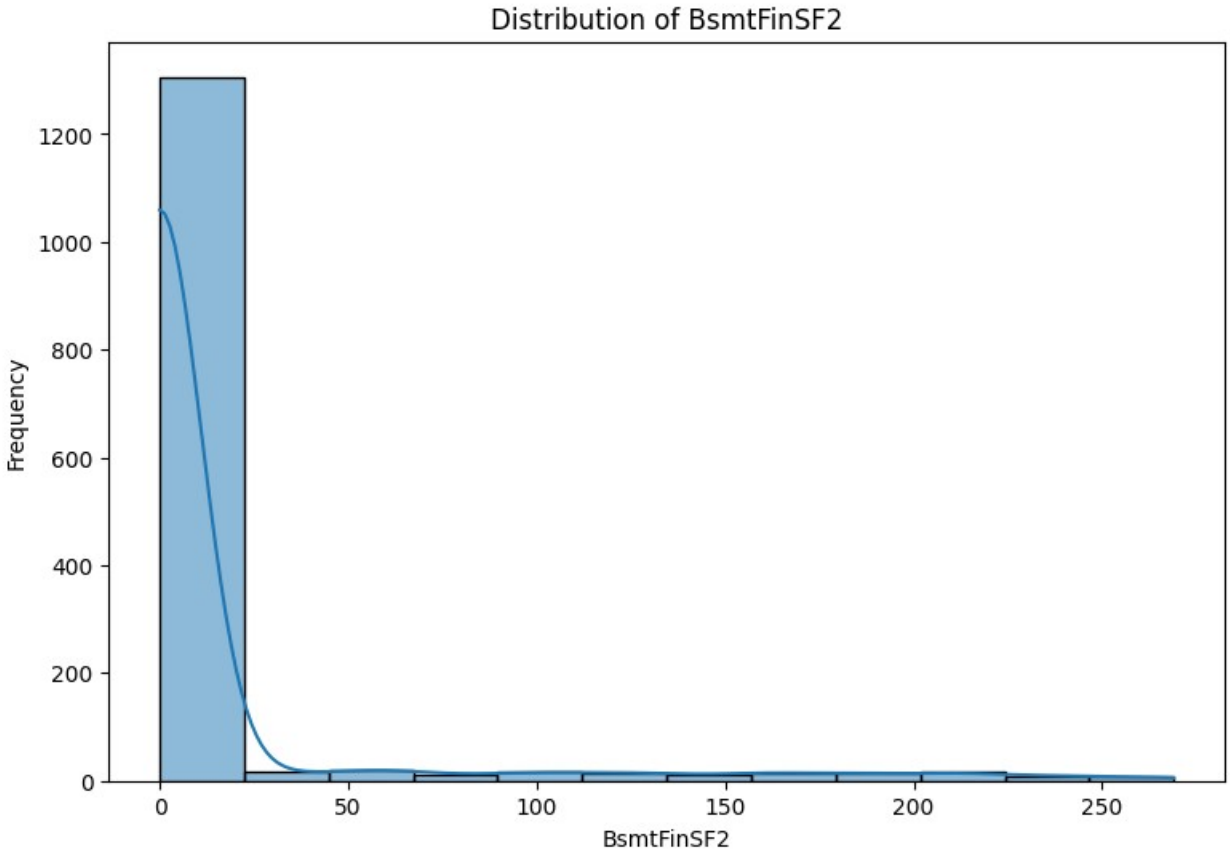

Distribution of OverallCond
Frequency
OverallCond
0
100
200
300
400
500
600
700
800
1
2
3
4
5
6
7
8
9

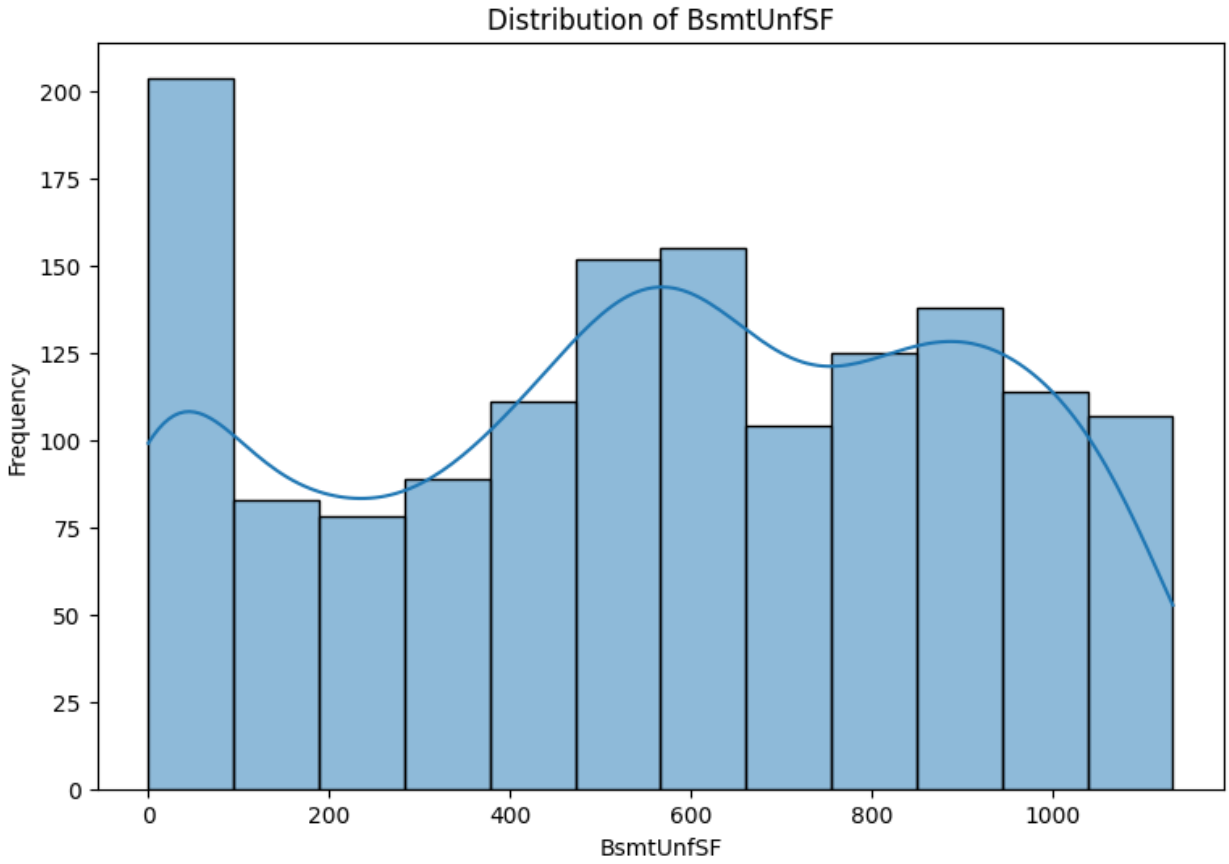

Distribution of YearBuilt
Frequency
YearBuilt
0
50
100
150
200
250
300
1880
1900
1920
1940
1960
1980
2000

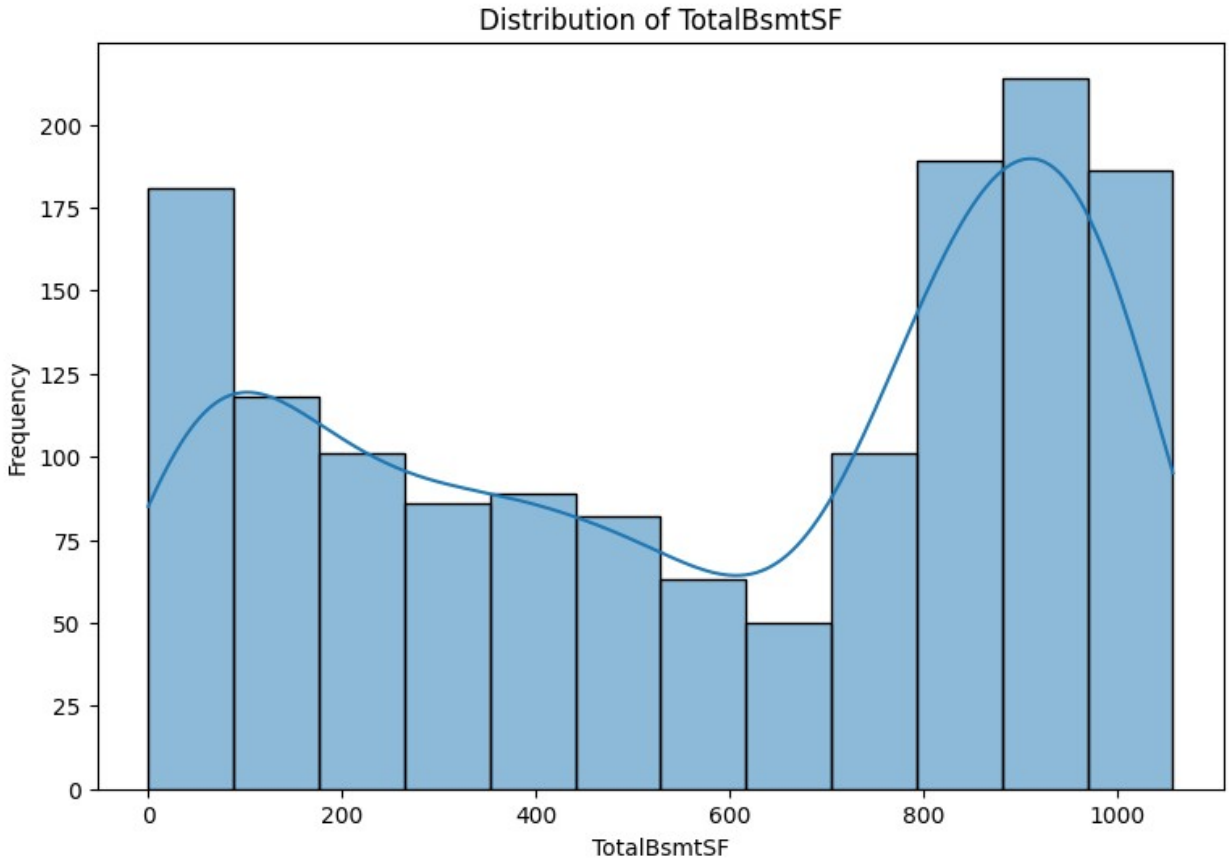

Distribution of YearRemodAdd
Frequency
YearRemodAdd
0
50
100
150
200
250
300
1950
1960
1970
1980
1990
2000
2010

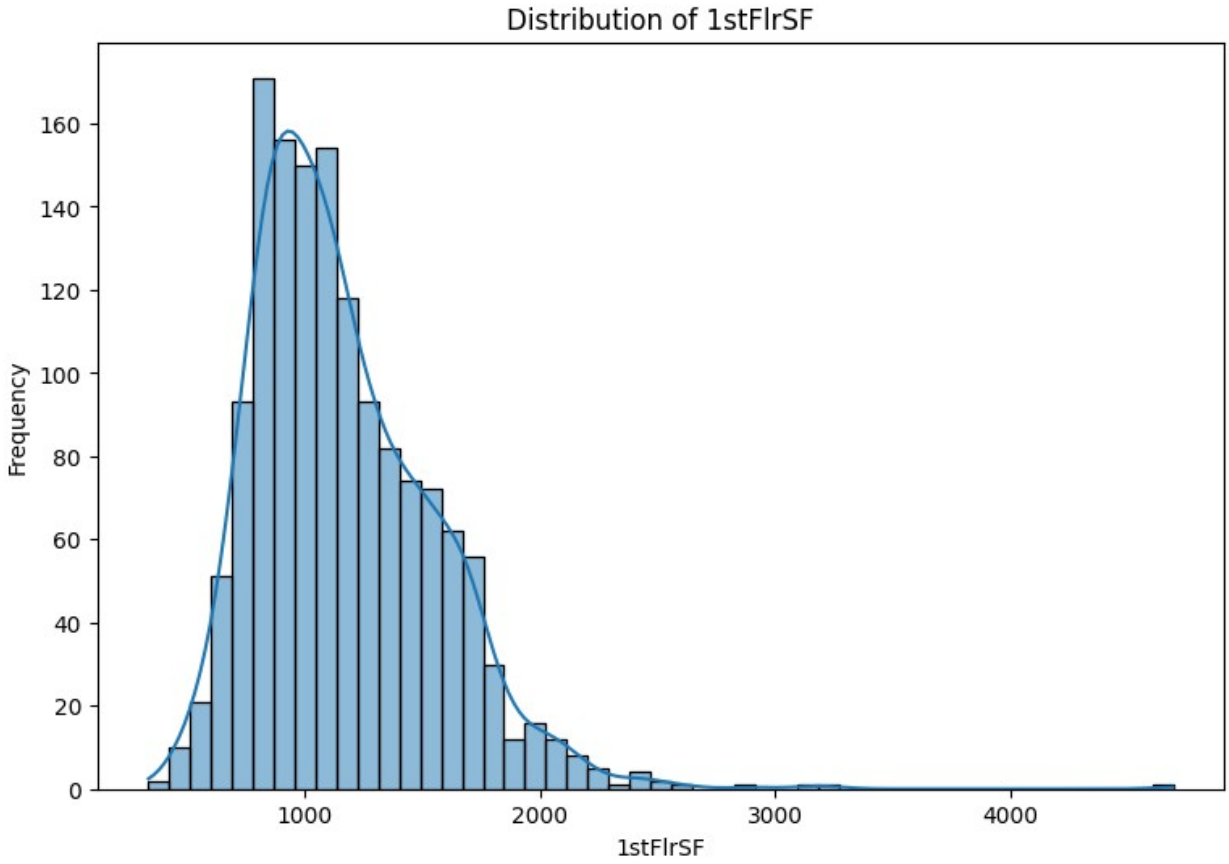

Distribution of MasVnrArea
Frequency
MasVnrArea
0
200
400
600
800
0
100
200
300
400

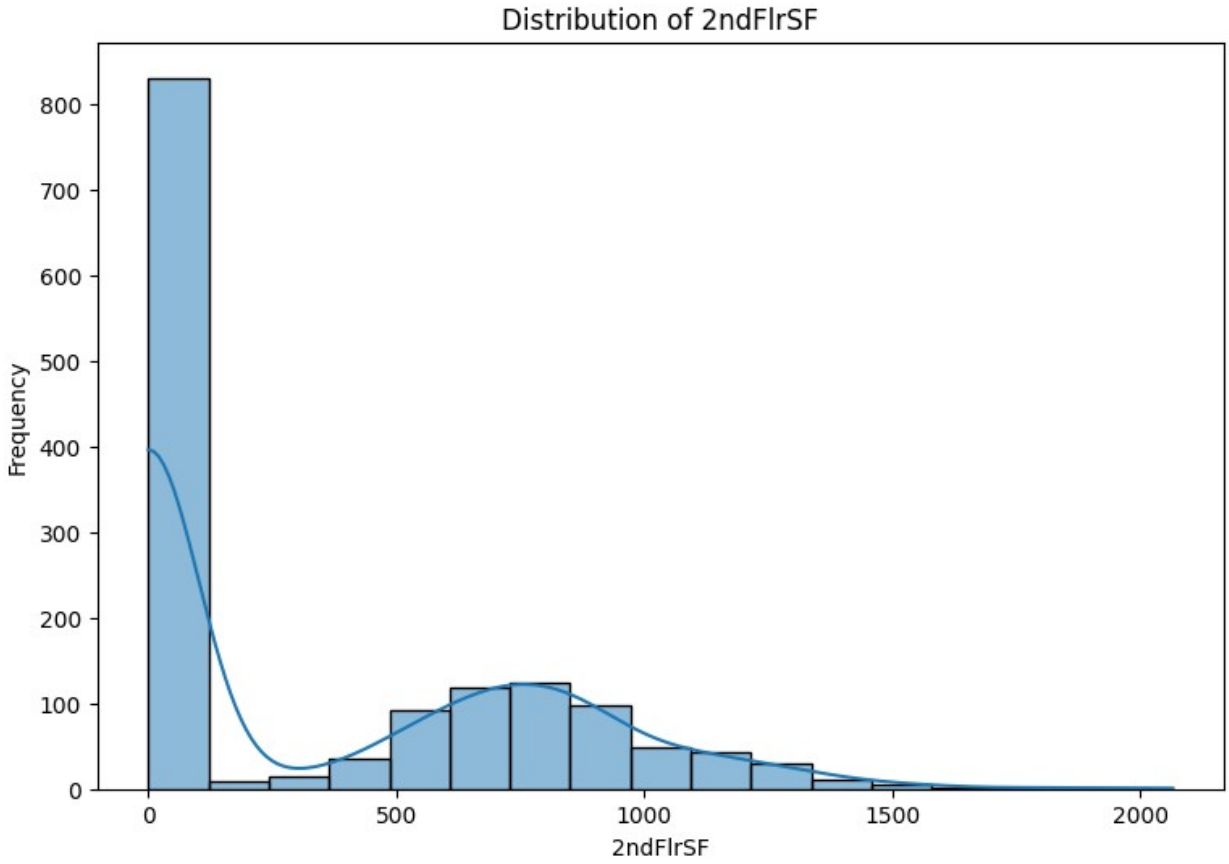

Distribution of BsmtFinSF1
Frequency
BsmtFinSF1
0
100
200
300
400
500
0
200
400
600
800
1000

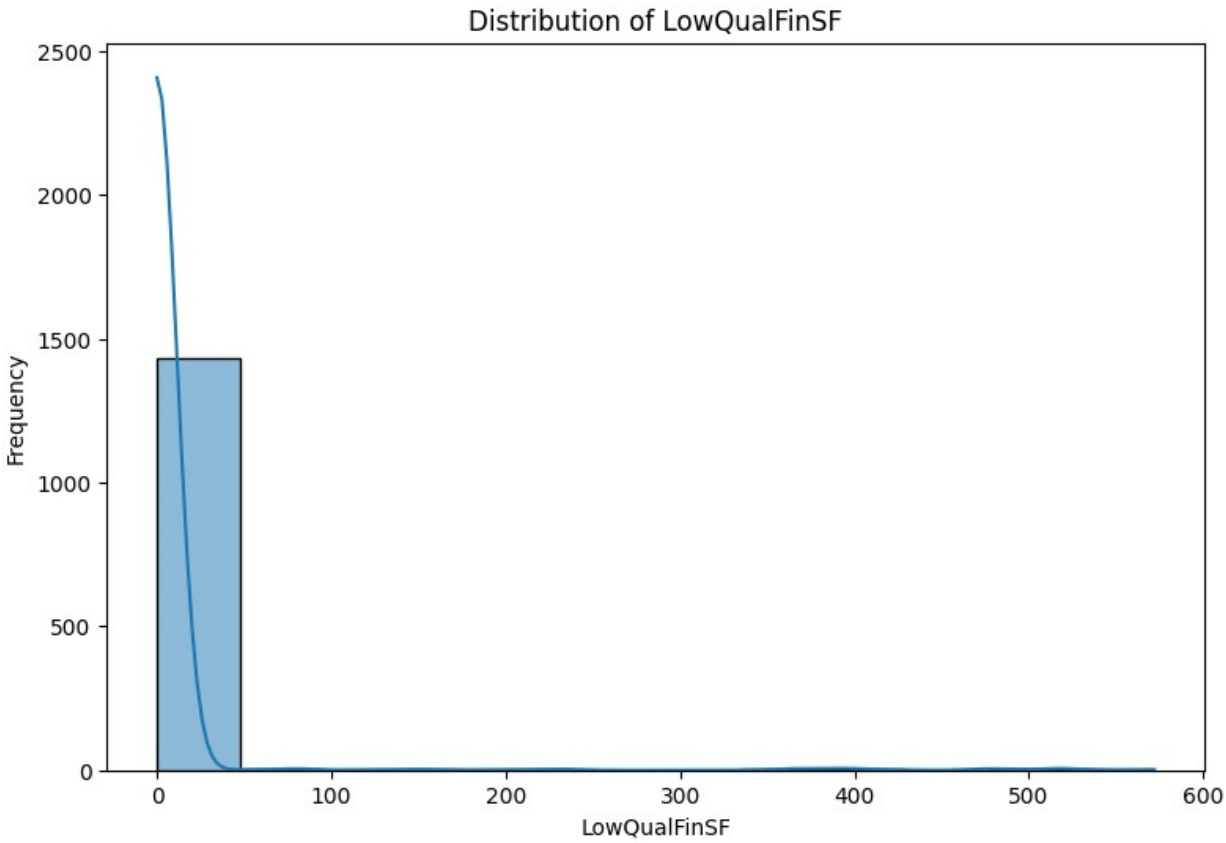

Distribution of BsmtFinSF2
Frequency
BsmtFinSF2
0
200
400
600
800
1000
1200
0
50
100
150
200
250

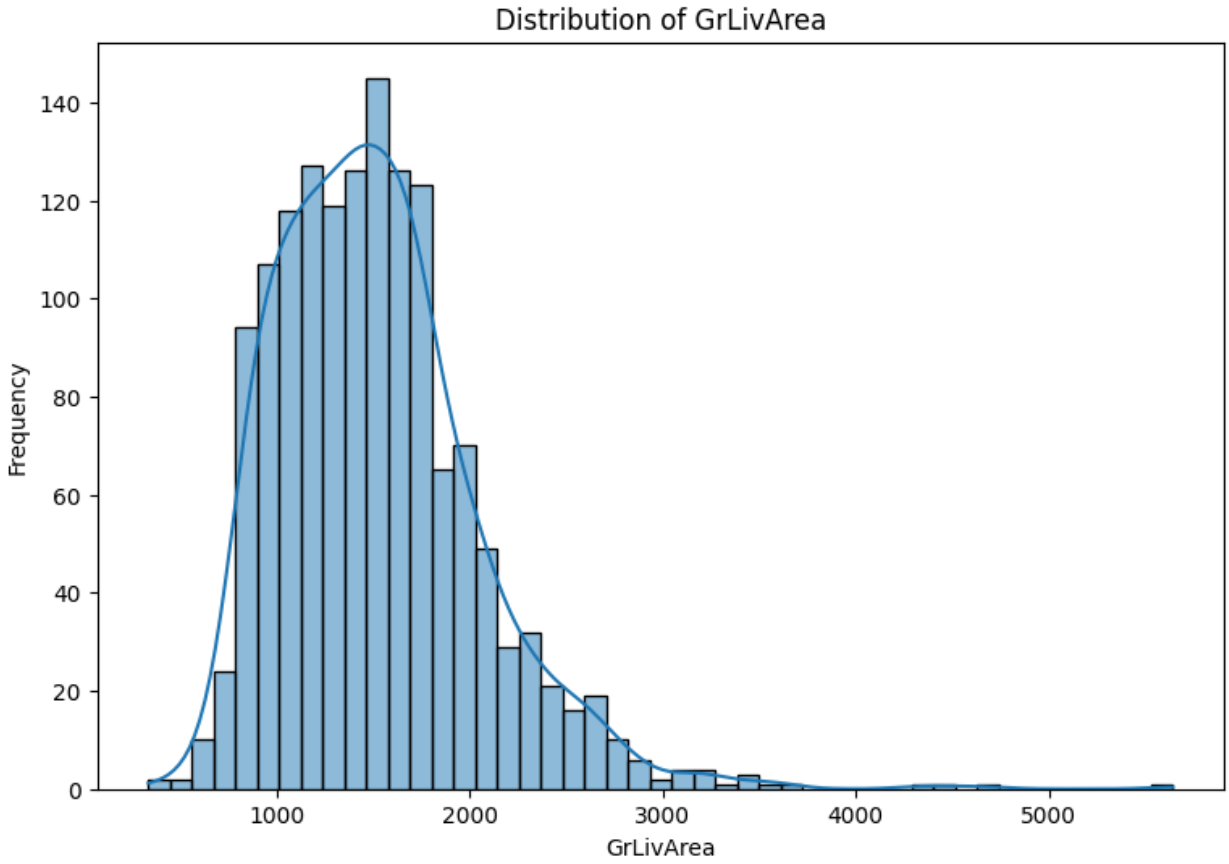

Distribution of BsmtUnfSF
Frequency
BsmtUnfSF
0
25
50
75
100
125
150
175
200
0
200
400
600
800
1000

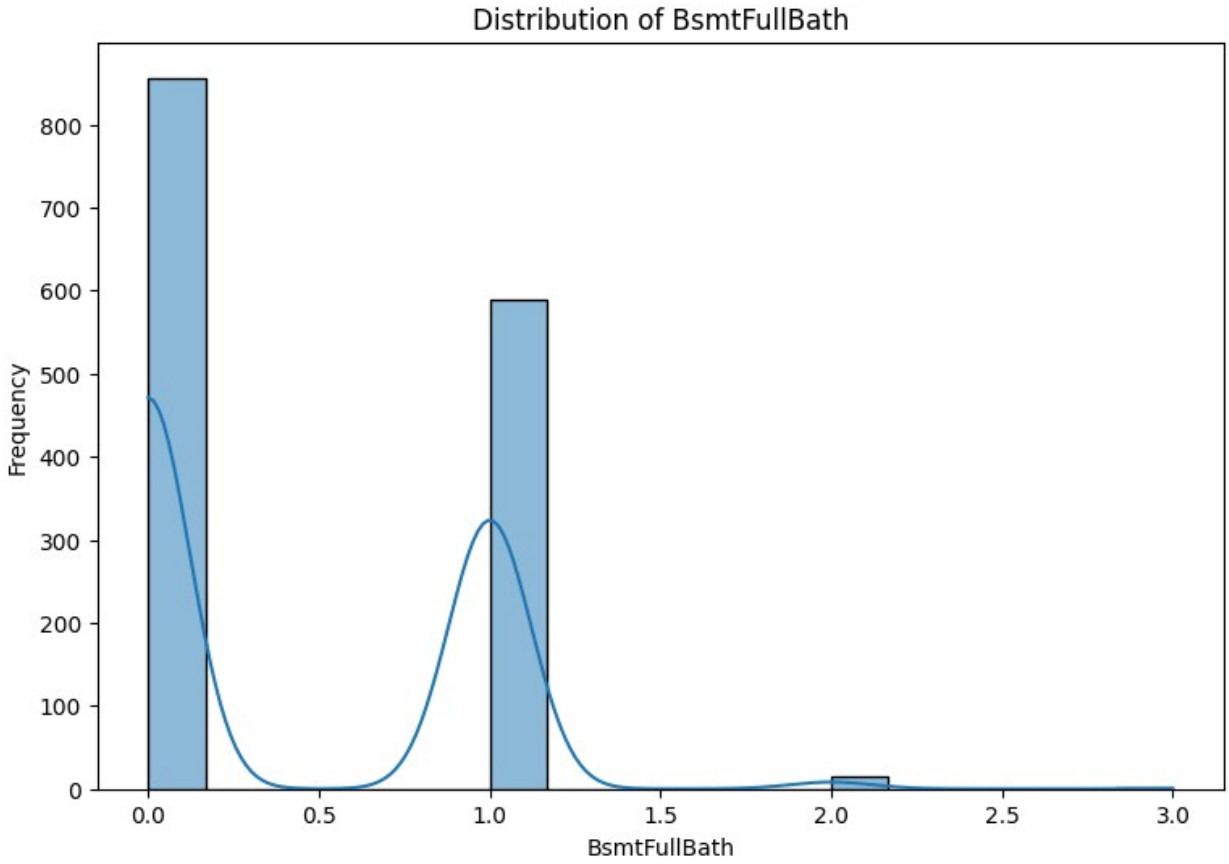

Distribution of TotalBsmtSF
Frequency
TotalBsmtSF
0
25
50
75
100
125
150
175
200
0
200
400
600
800
1000

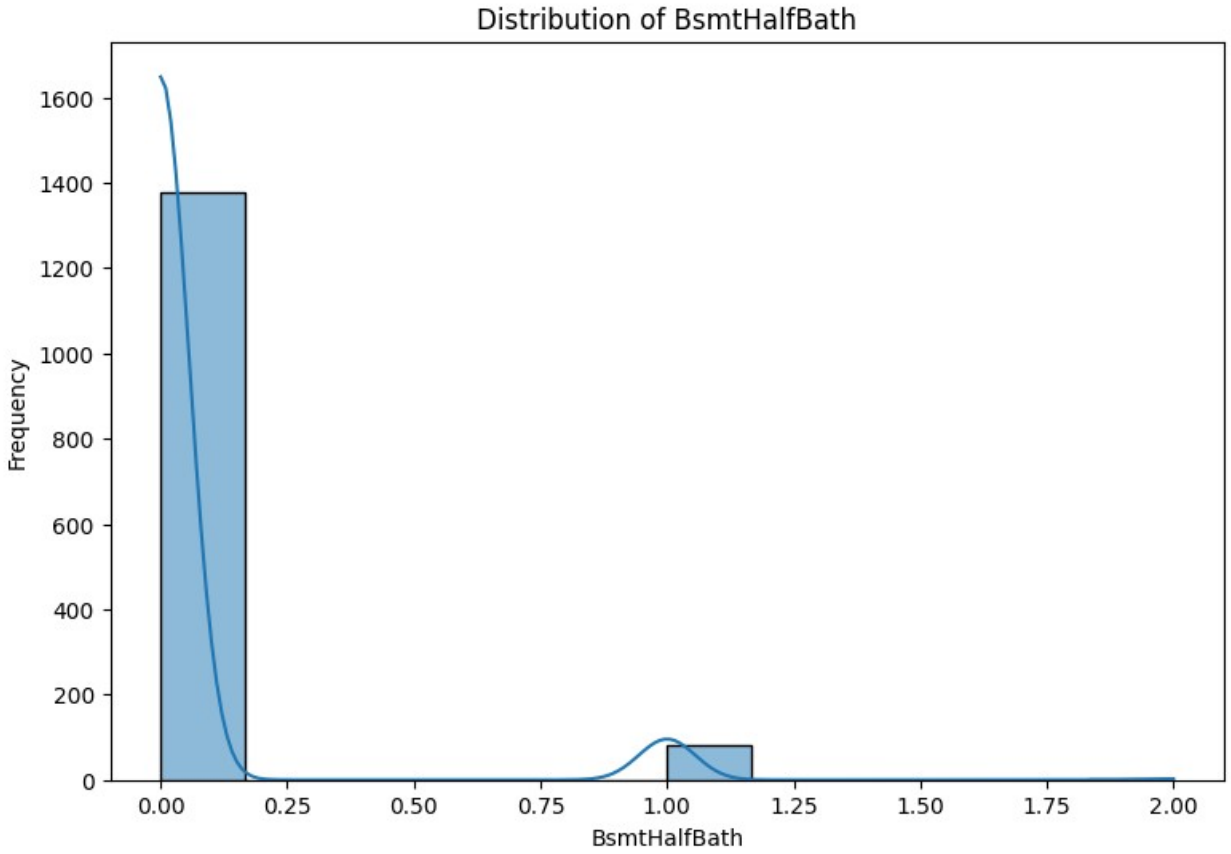

Distribution of 1stFlrSF
Frequency
1stFlrSF
0
20
40
60
80
100
120
140
160
1000
2000
3000
4000

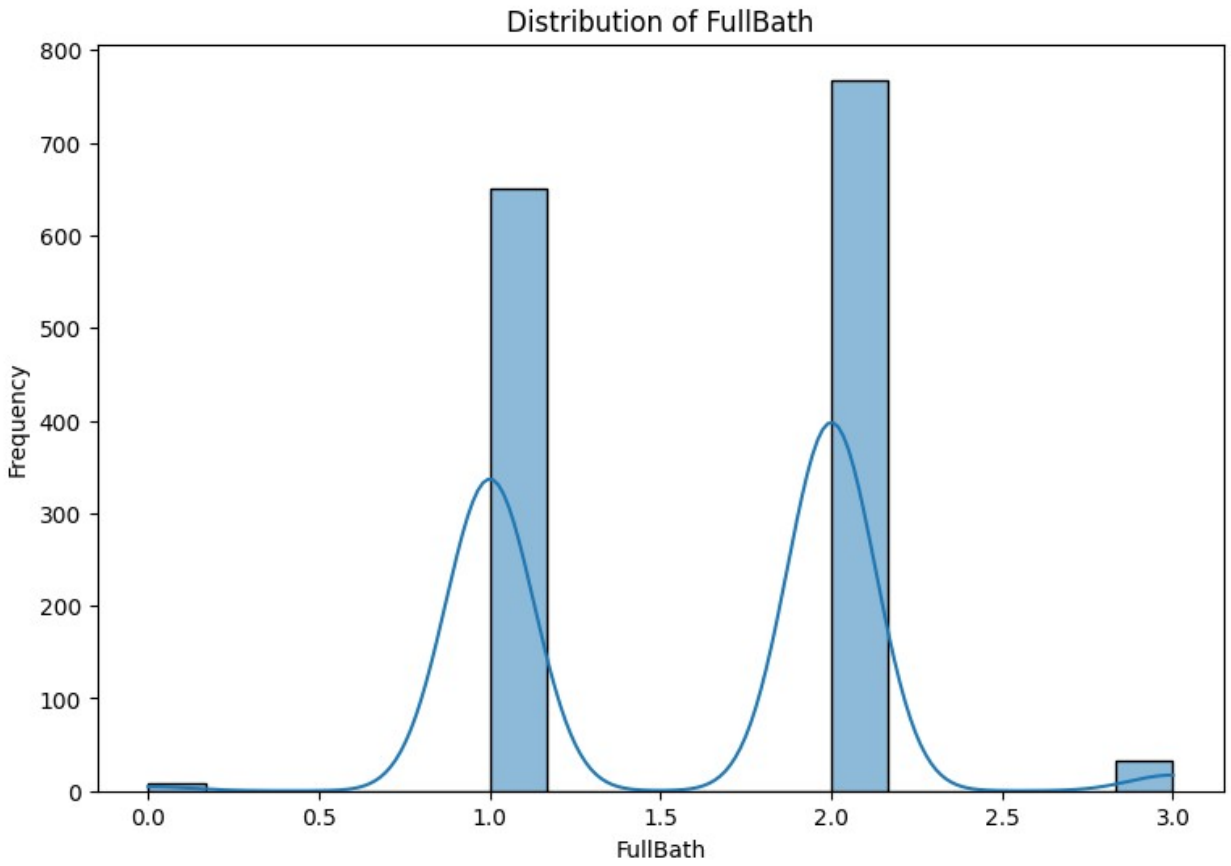

Distribution of 2ndFlrSF
Frequency
2ndFlrSF
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000

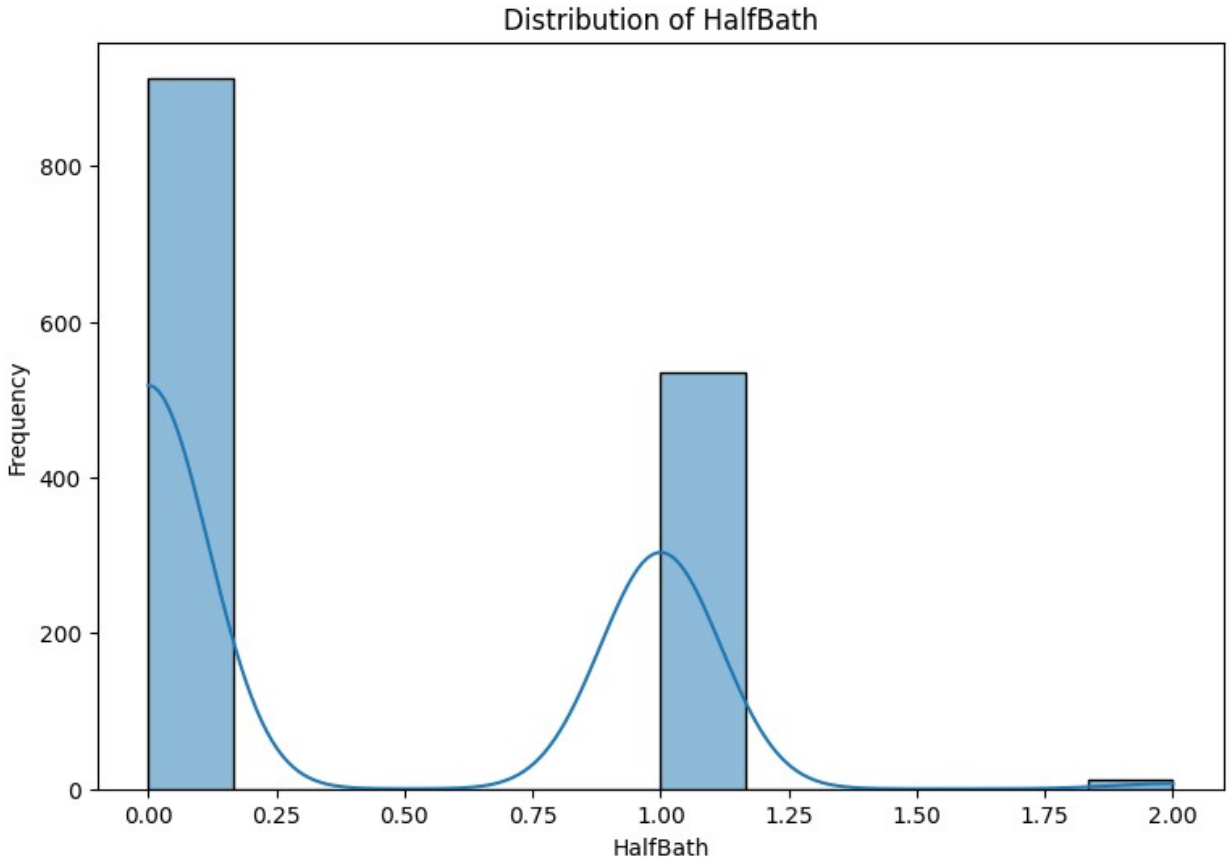

Distribution of LowQualFinSF
Frequency
LowQualFinSF
2500
2000
1500
1000
500
0
0
100
200
300
400
500
600

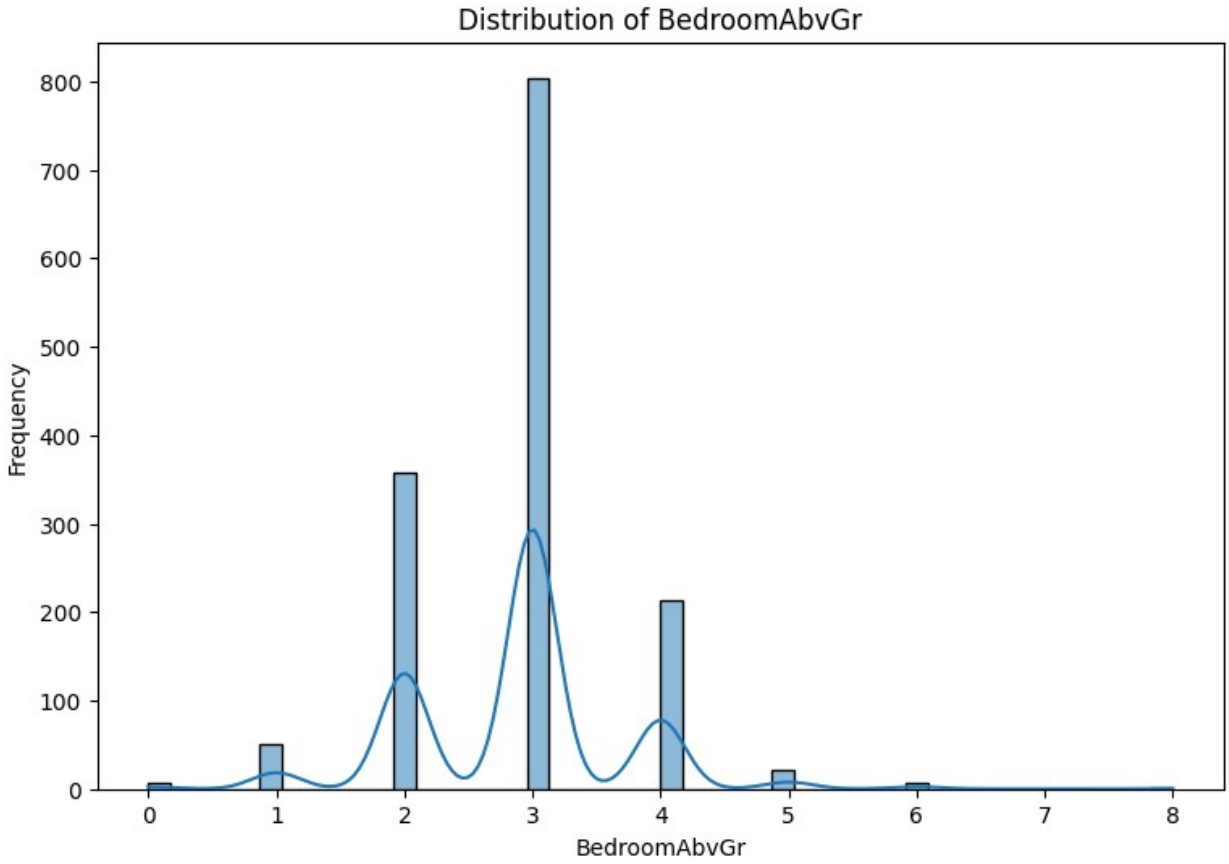

Distribution of GrLivArea
Frequency
GrLivArea
0
20
40
60
80
100
120
140
1000
2000
3000
4000
5000

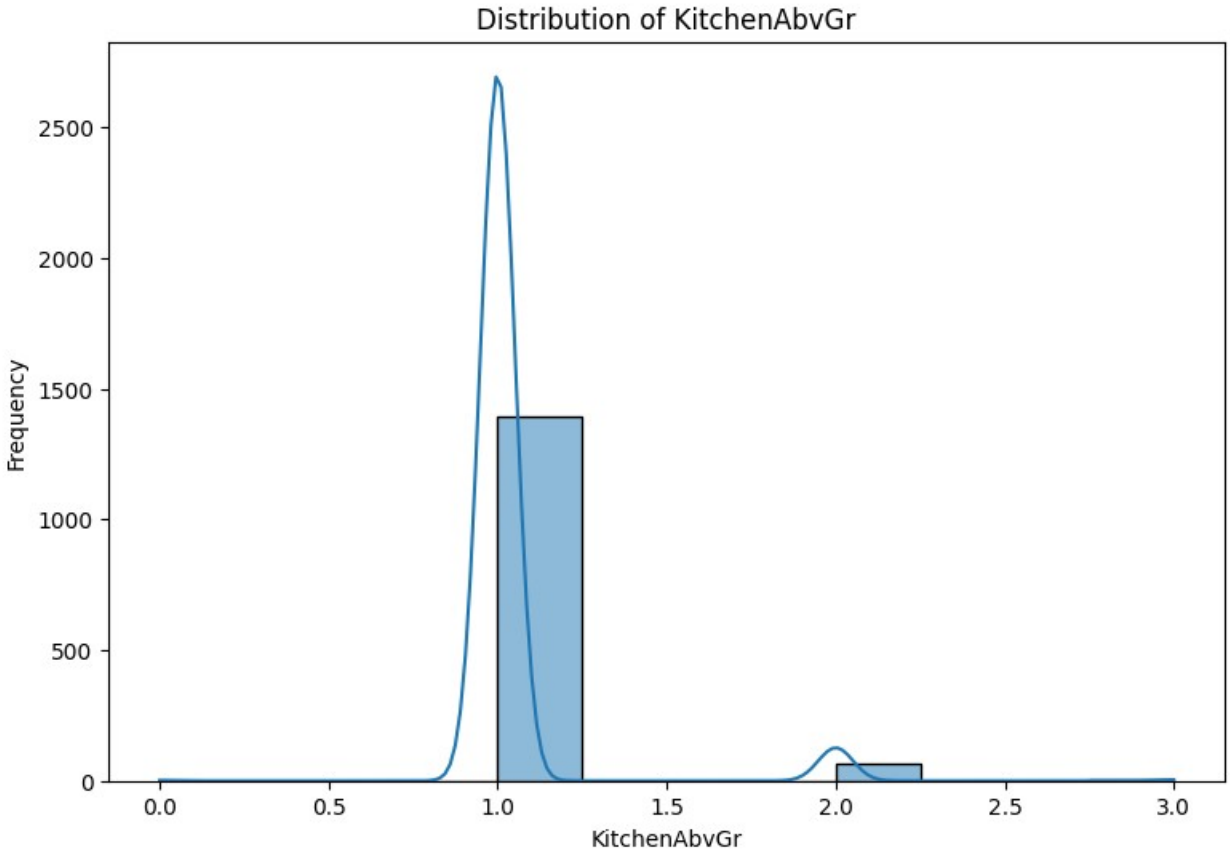

Distribution of BsmtFullBath
Frequency
BsmtFullBath
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0

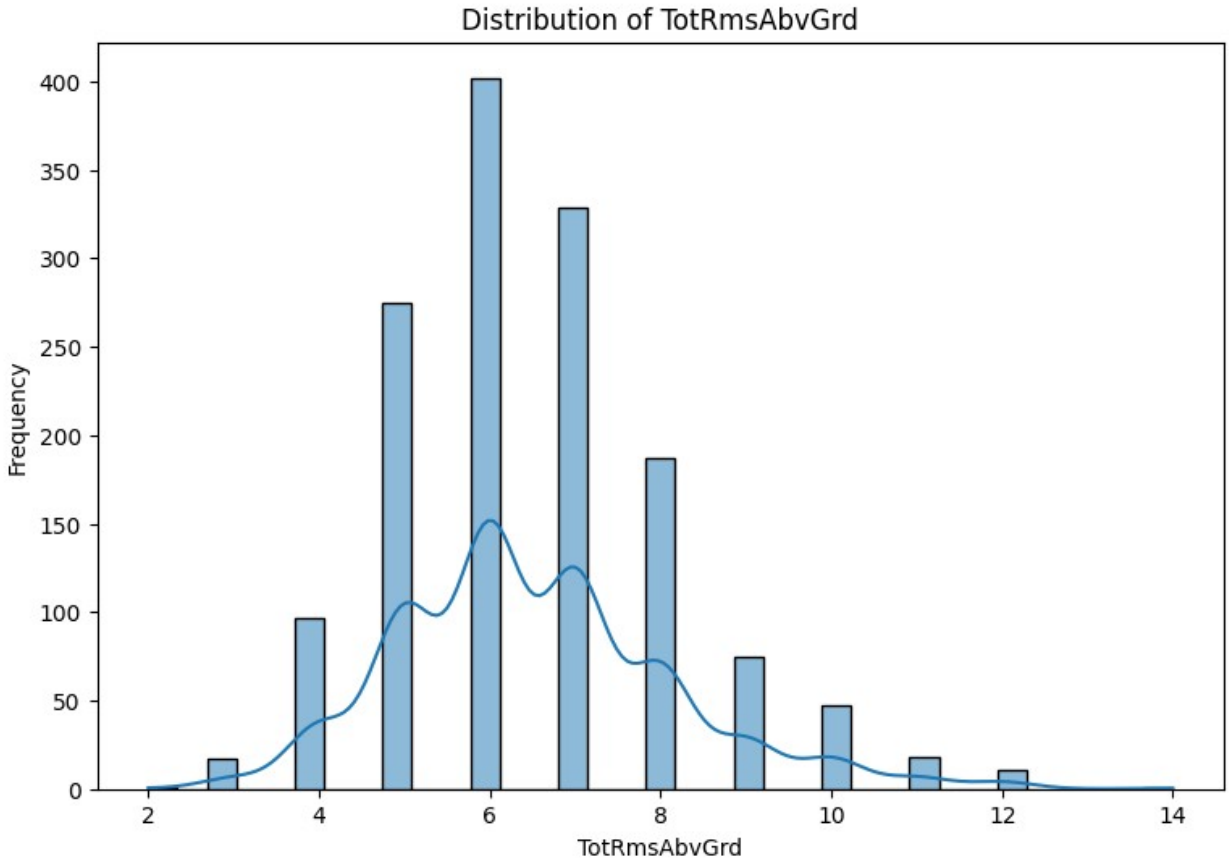

Distribution of BsmtHalfBath
Frequency
BsmtHalfBath
0
200
400
600
800
1000
1200
1400
1600
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

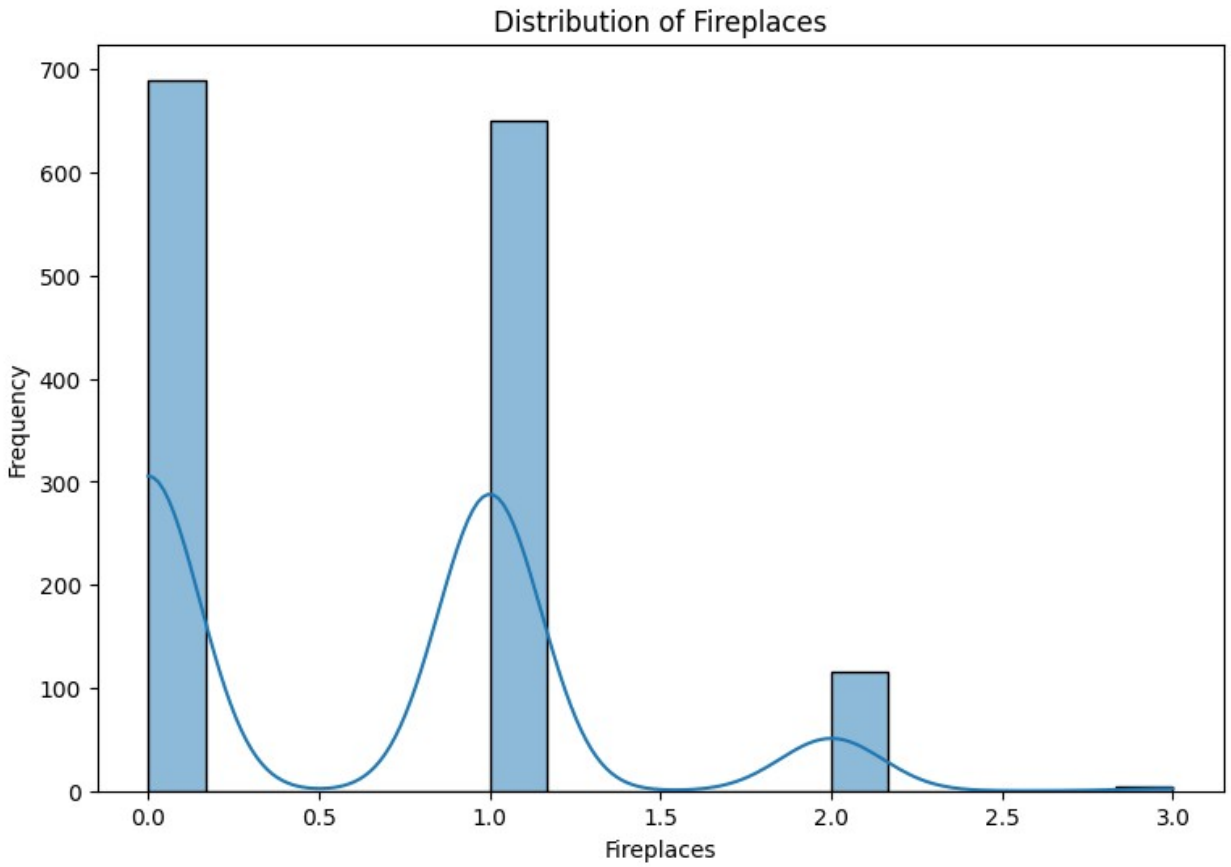

Distribution of FullBath
Frequency
FullBath
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

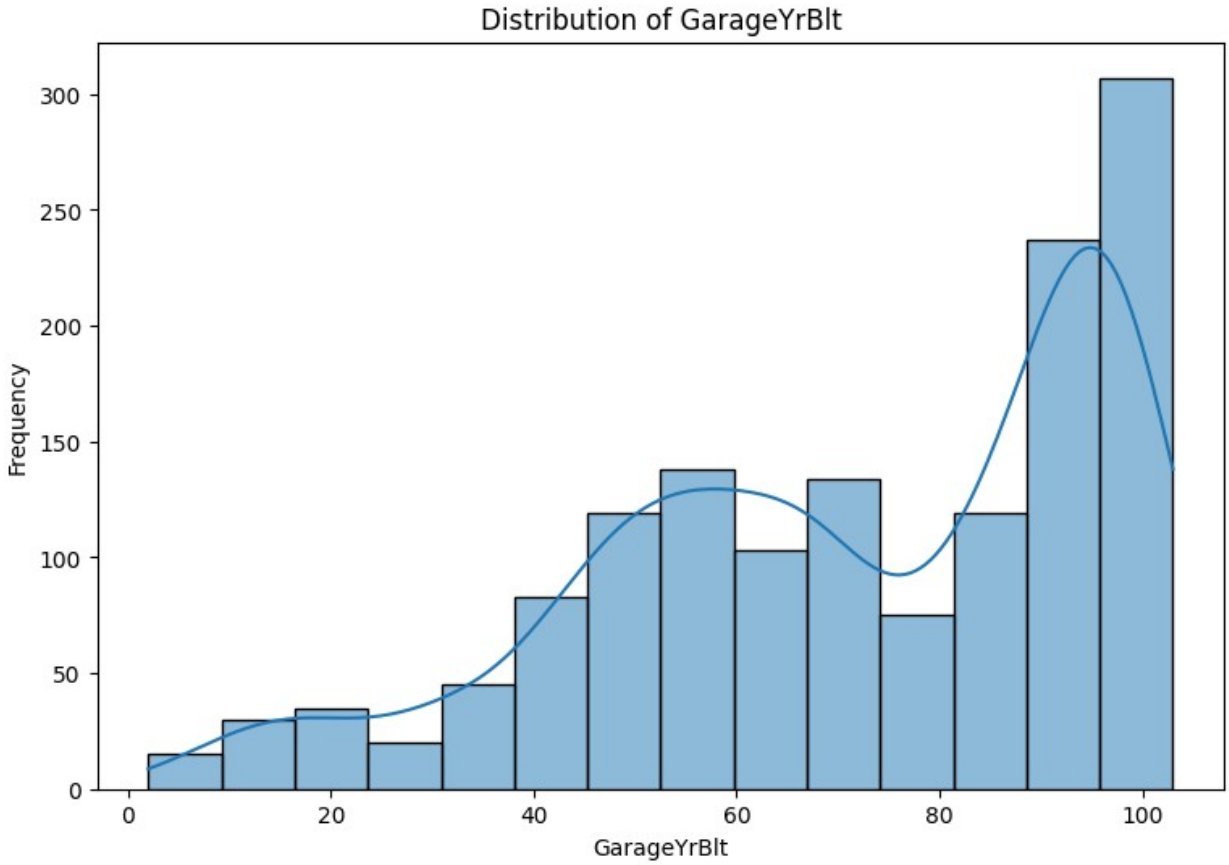

Distribution of HalfBath
Frequency
800
600
400
200
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
HalfBath

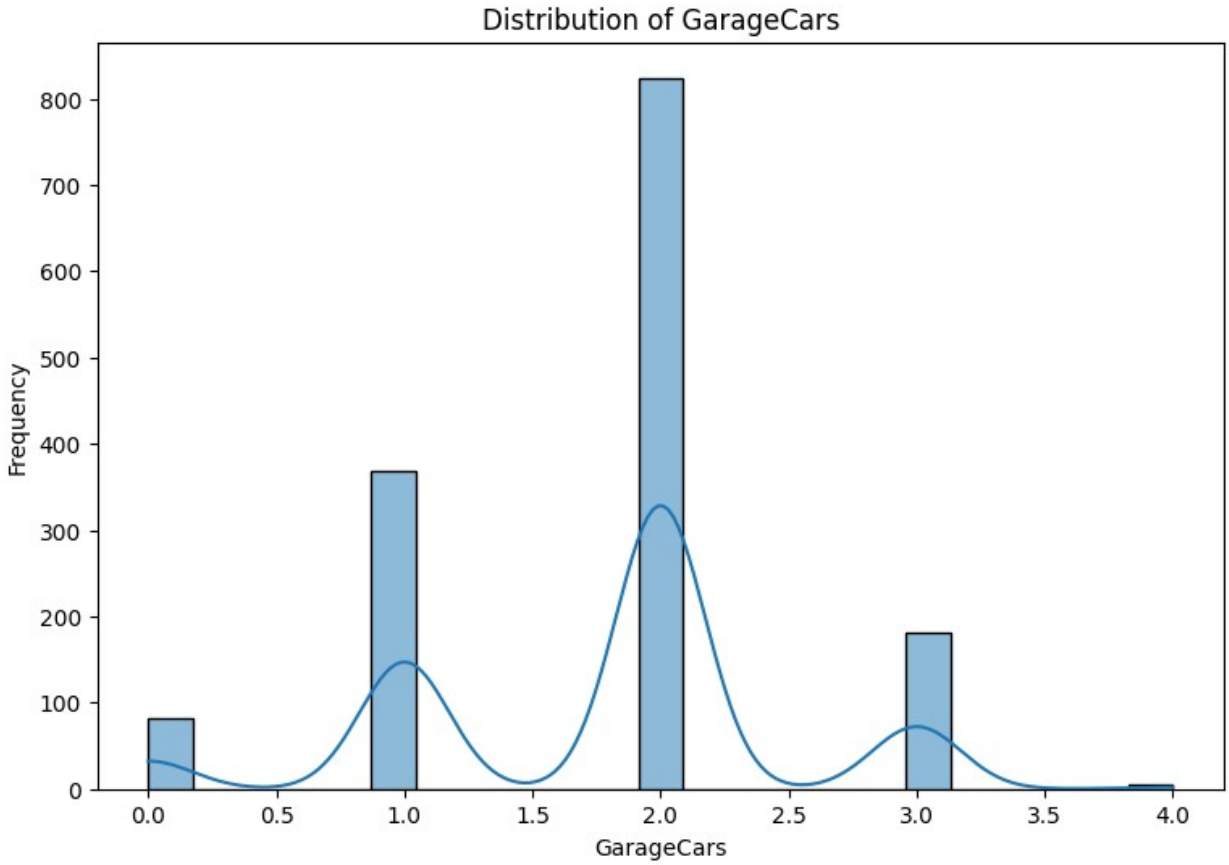

Distribution of BedroomAbvGr
Frequency
BedroomAbvGr
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6
7
8

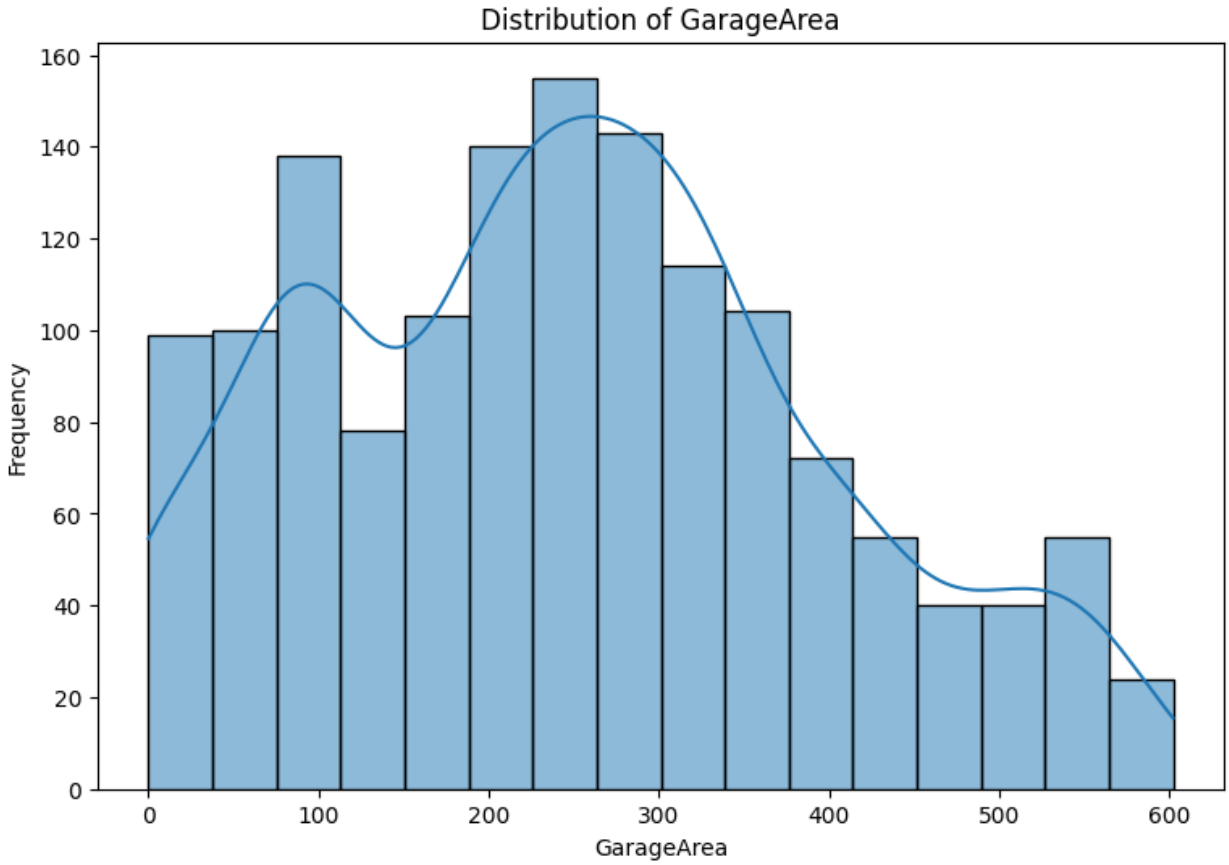

Distribution of KitchenAbvGr
Frequency
2500
2000
1500
1000
500
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
KitchenAbvGr

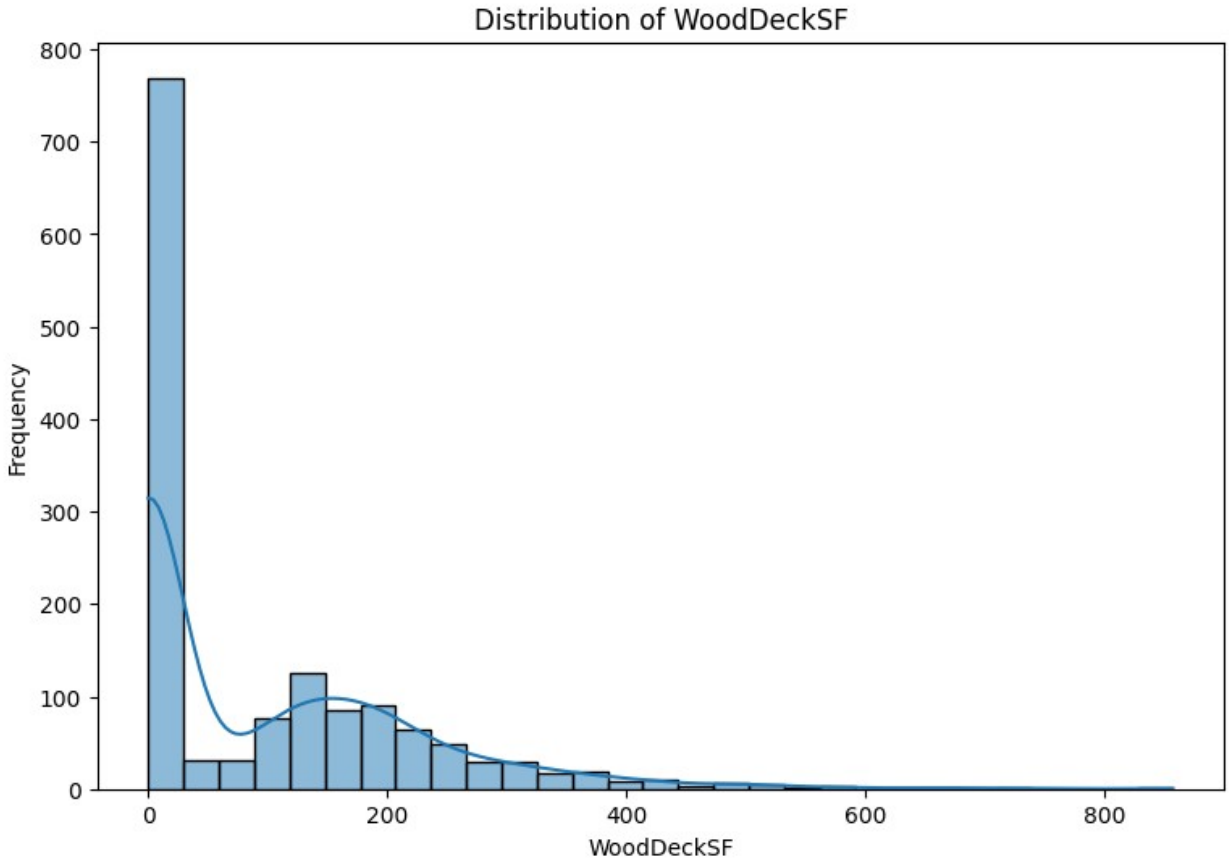

Distribution of TotRmsAbvGrd
Frequency
TotRmsAbvGrd
0
50
100
150
200
250
300
350
400
2
4
6
8
10
12
14

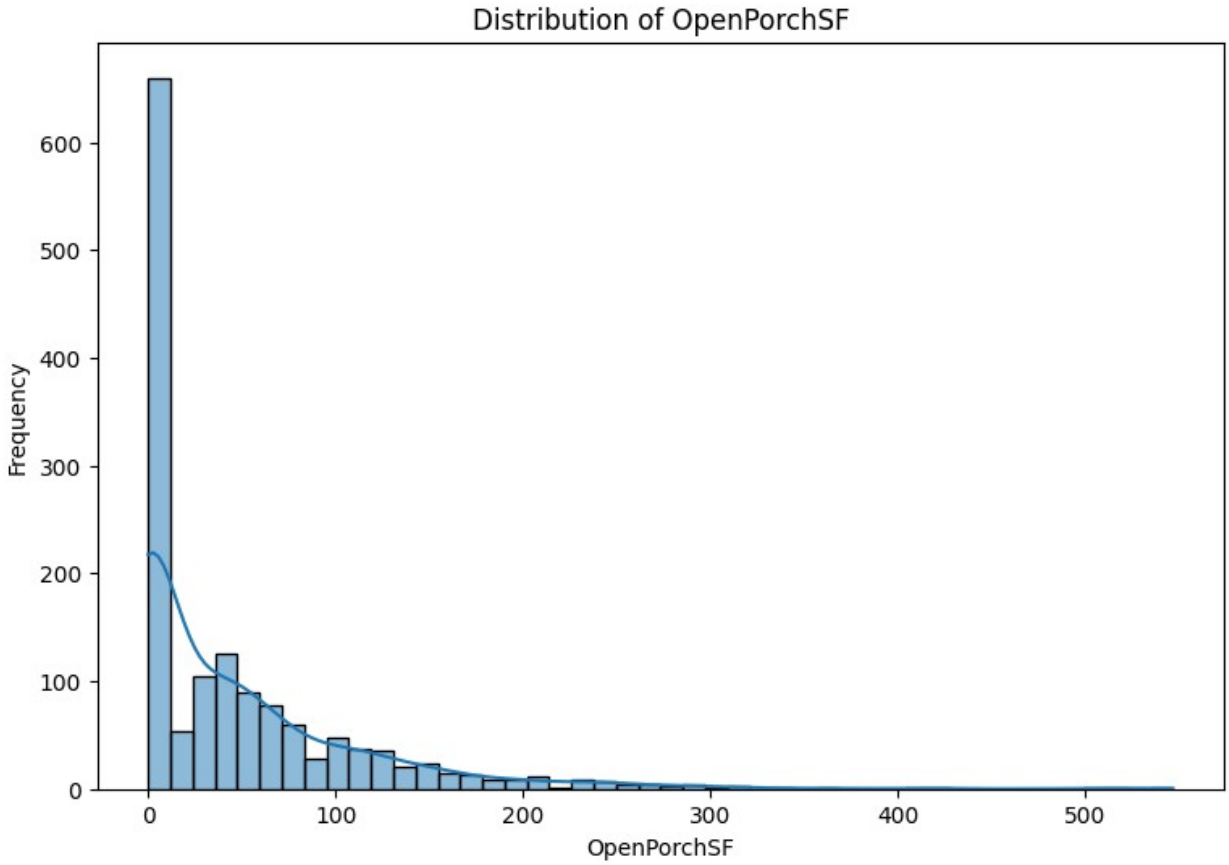

Distribution of Fireplaces
Frequency
Fireplaces
0
100
200
300
400
500
600
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0

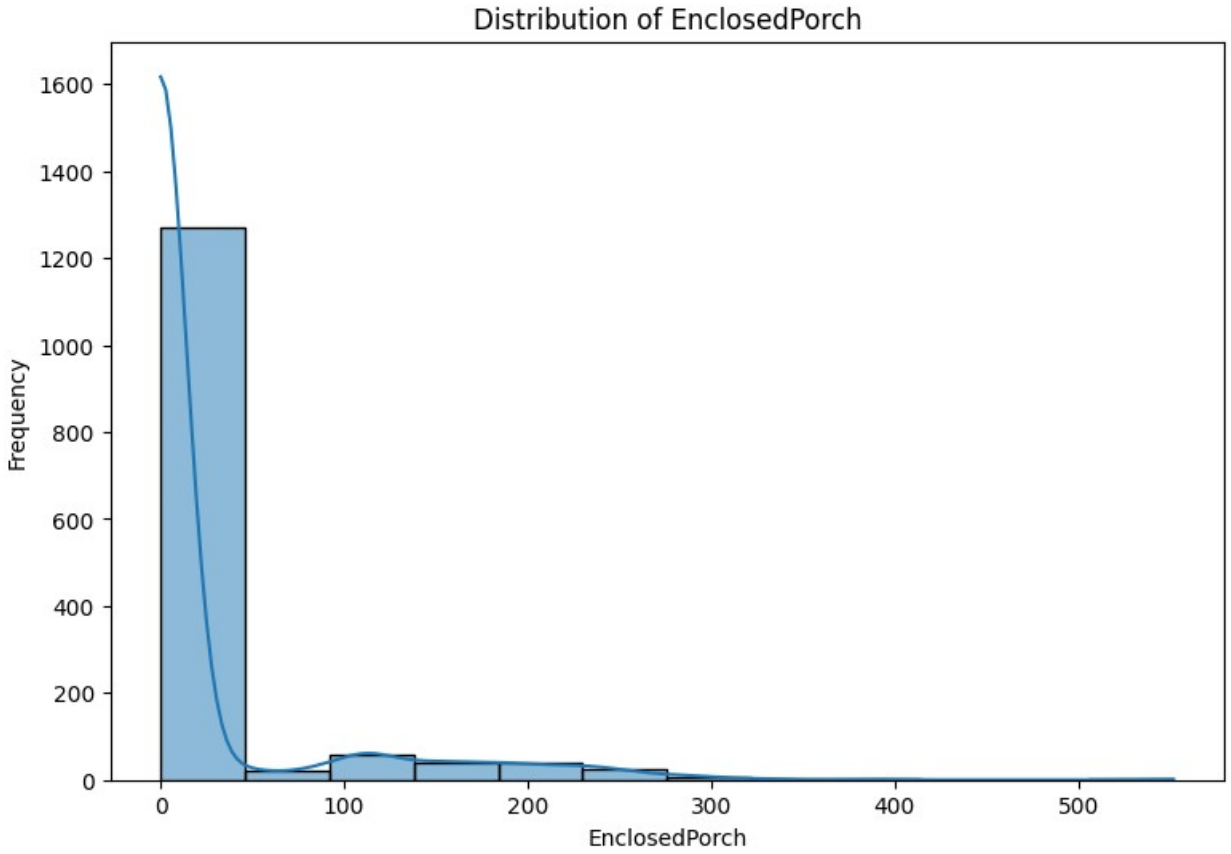

Distribution of GarageYrBlt
Frequency
GarageYrBlt
0
50
100
150
200
250
300
0
20
40
60
80
100

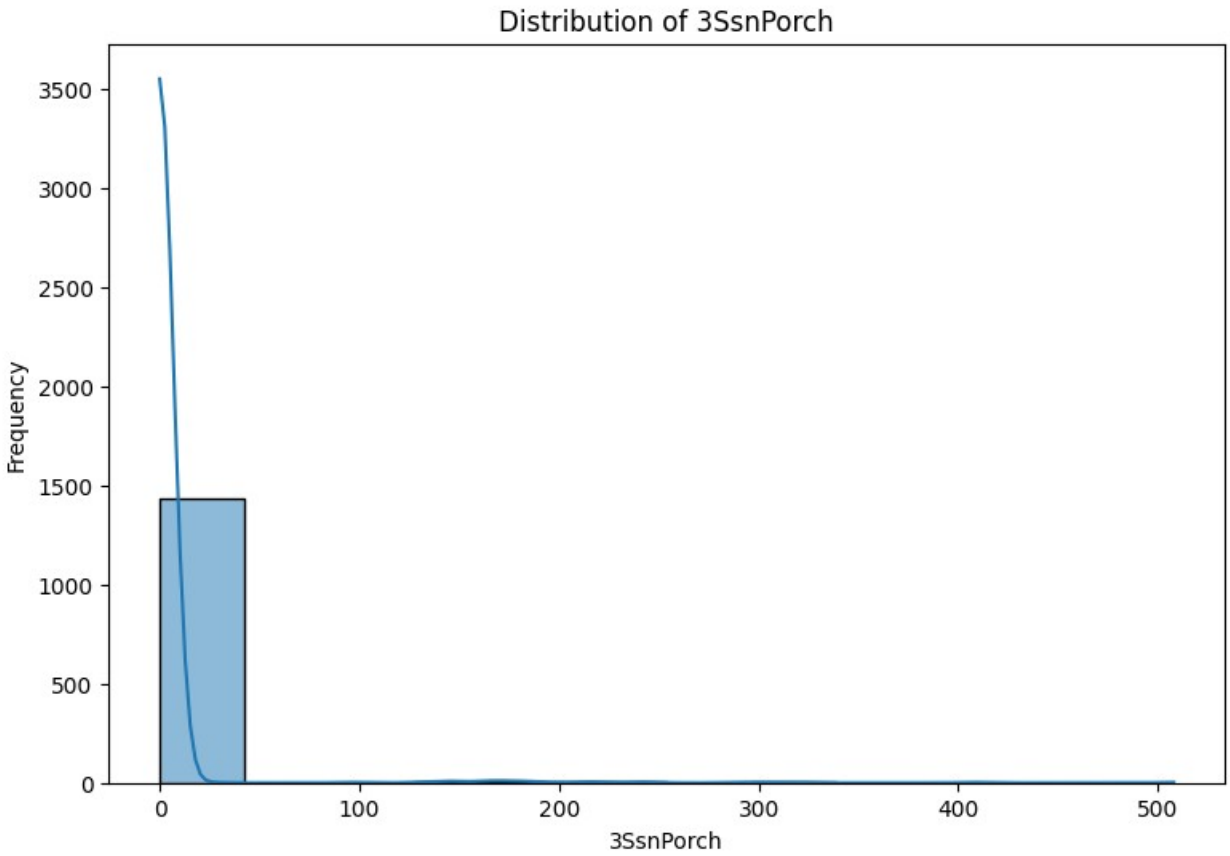

Distribution of GarageCars
Frequency
GarageCars
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

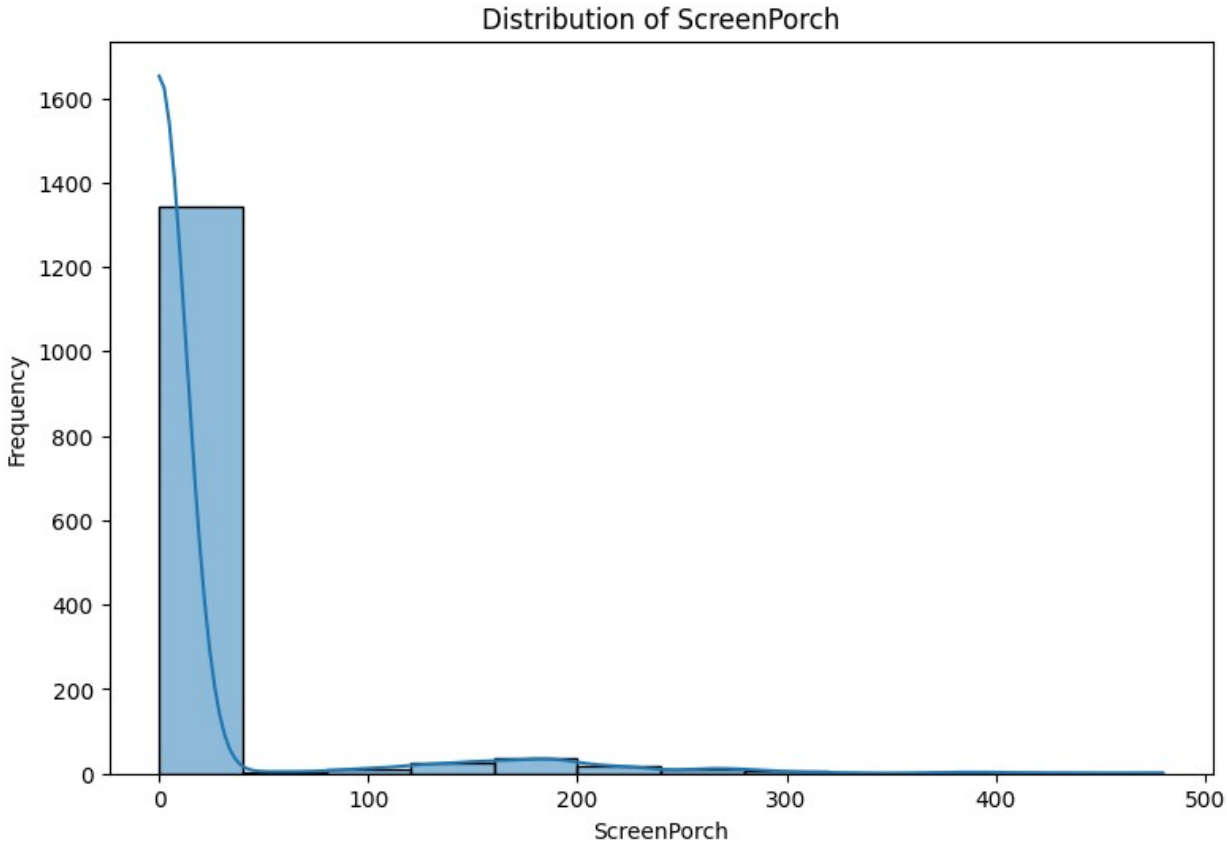

Distribution of GarageArea
Frequency
GarageArea
0
20
40
60
80
100
120
140
160
0
100
200
300
400
500
600

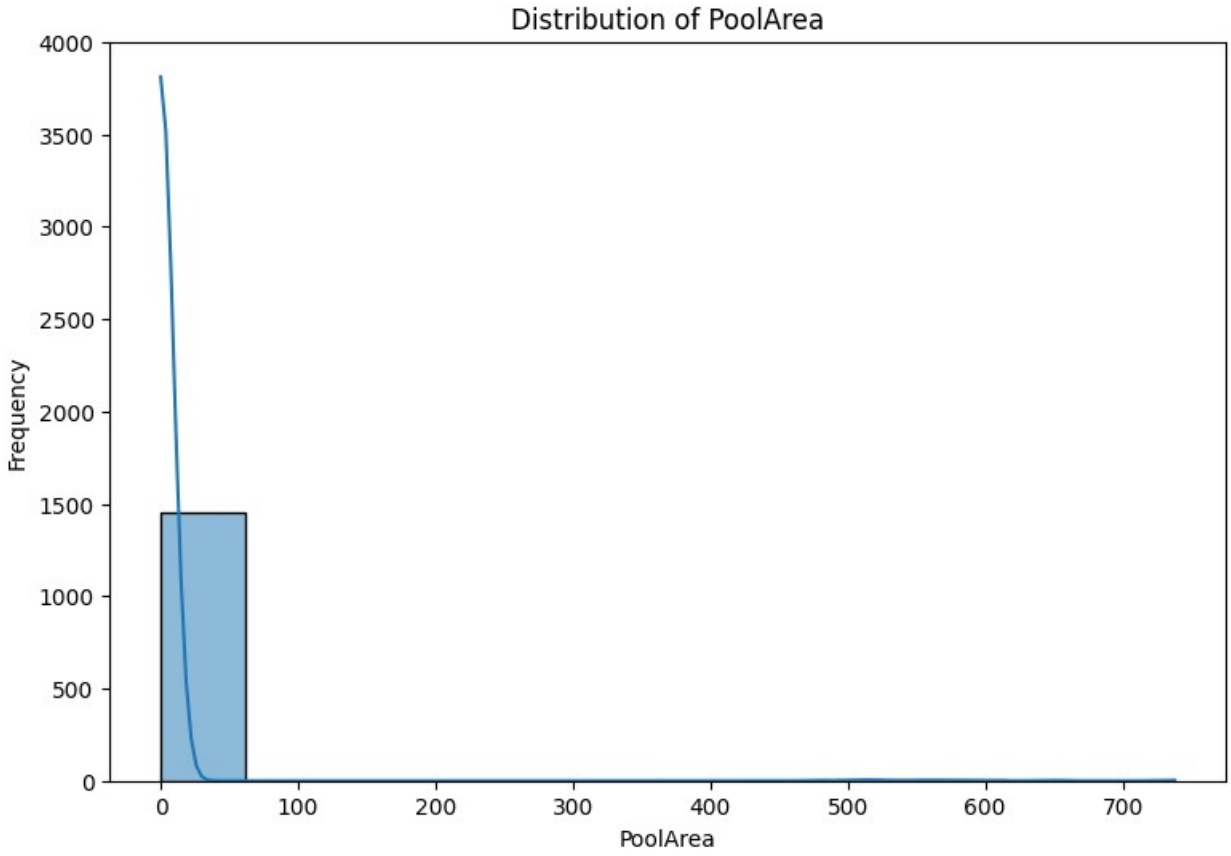

Distribution of WoodDeckSF
Frequency
WoodDeckSF
0
100
200
300
400
500
600
700
800
0
200
400
600
800

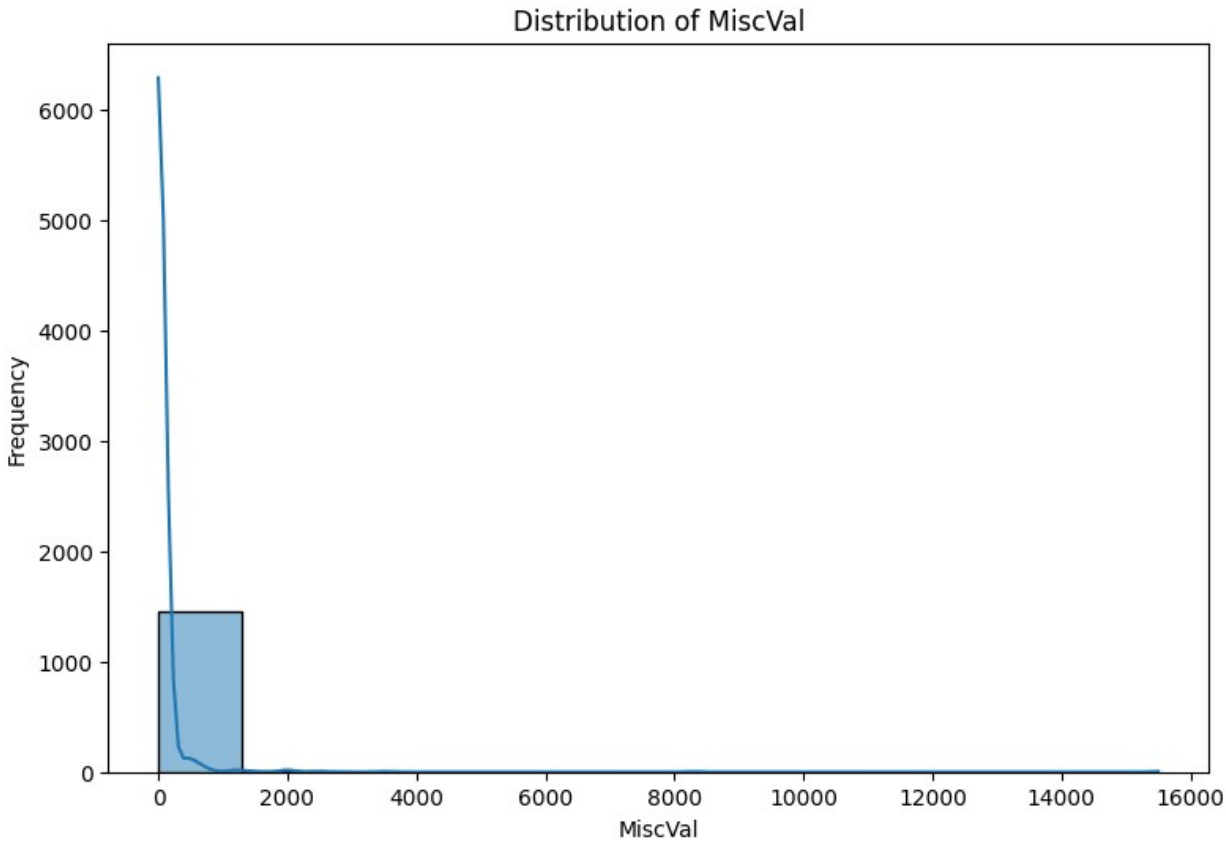

Distribution of OpenPorchSF
Frequency
OpenPorchSF
0
100
200
300
400
500
600

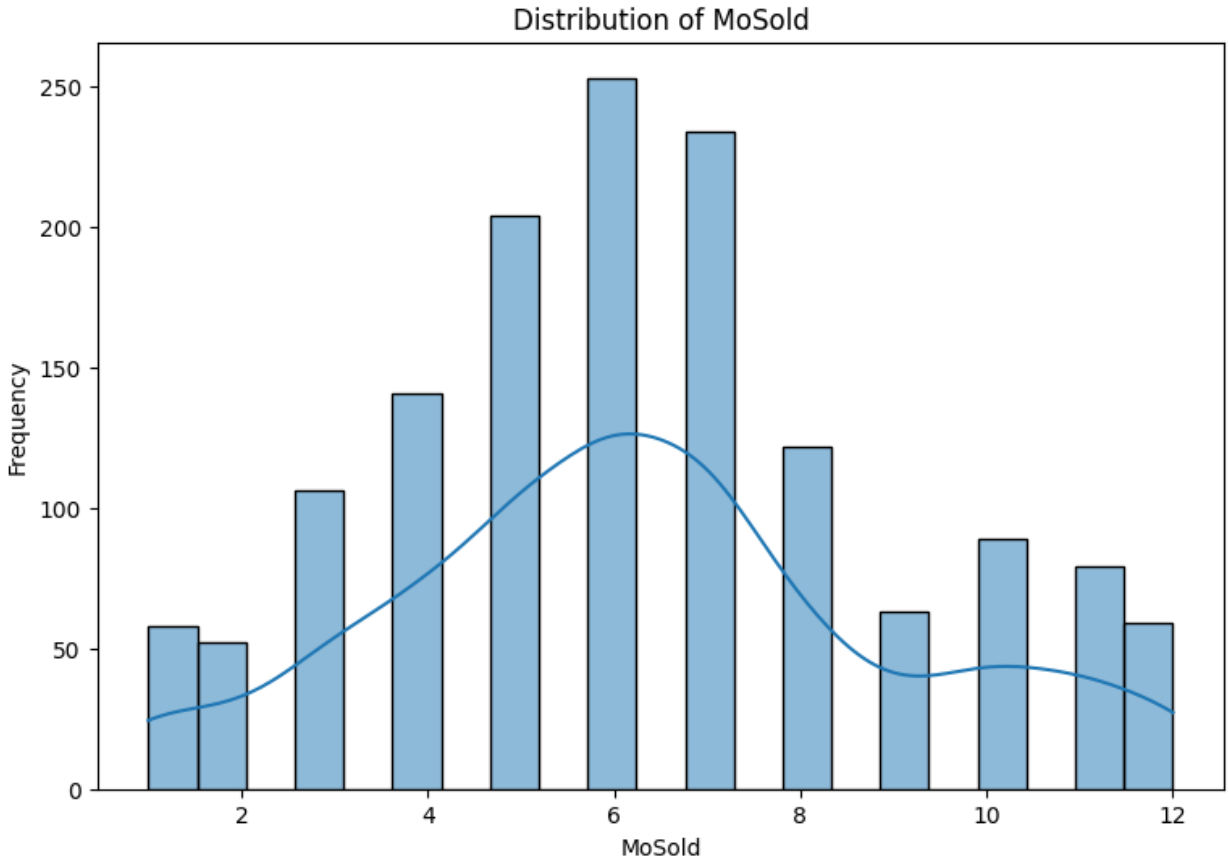

Distribution of EnclosedPorch
Frequency
EnclosedPorch
0
200
400
600
800
1000
1200
1400
1600
0
100
200
300
400
500

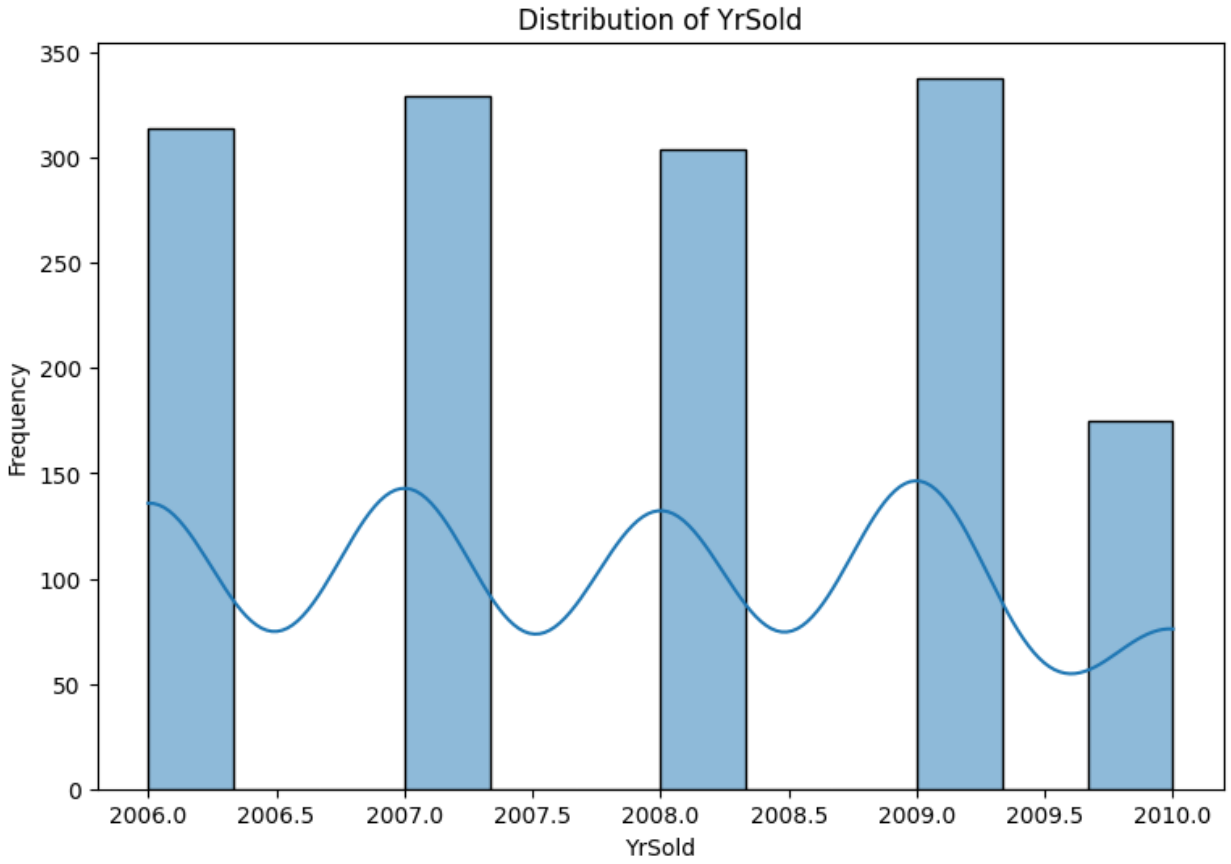

Distribution of 3SsnPorch
Frequency
3SsnPorch
0
500
1000
1500
2000
2500
3000
3500
0
100
200
300
400
500

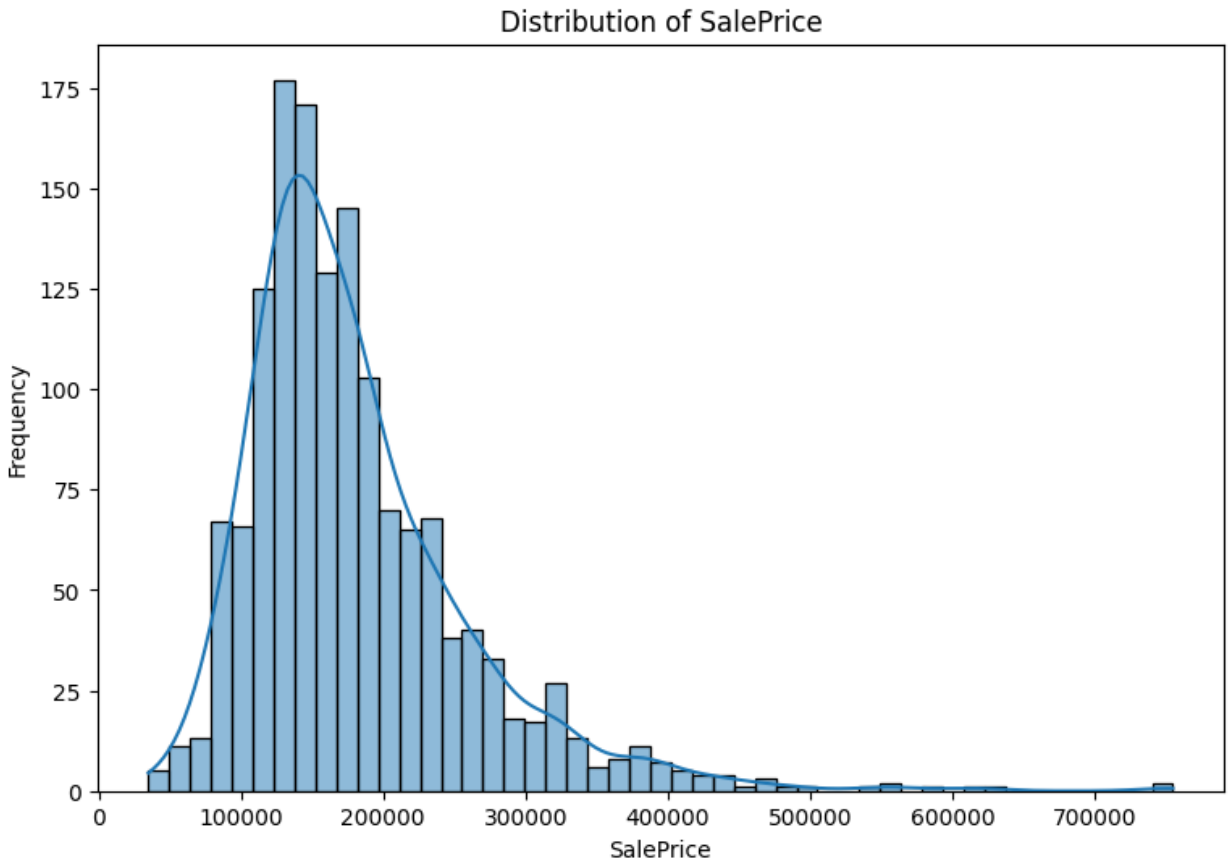

Distribution of ScreenPorch
Frequency
ScreenPorch
0
200
400
600
800
1000
1200
1400
1600
0
100
200
300
400
500


Distribution of PoolArea
Frequency
PoolArea
4000
3500
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500
600
700

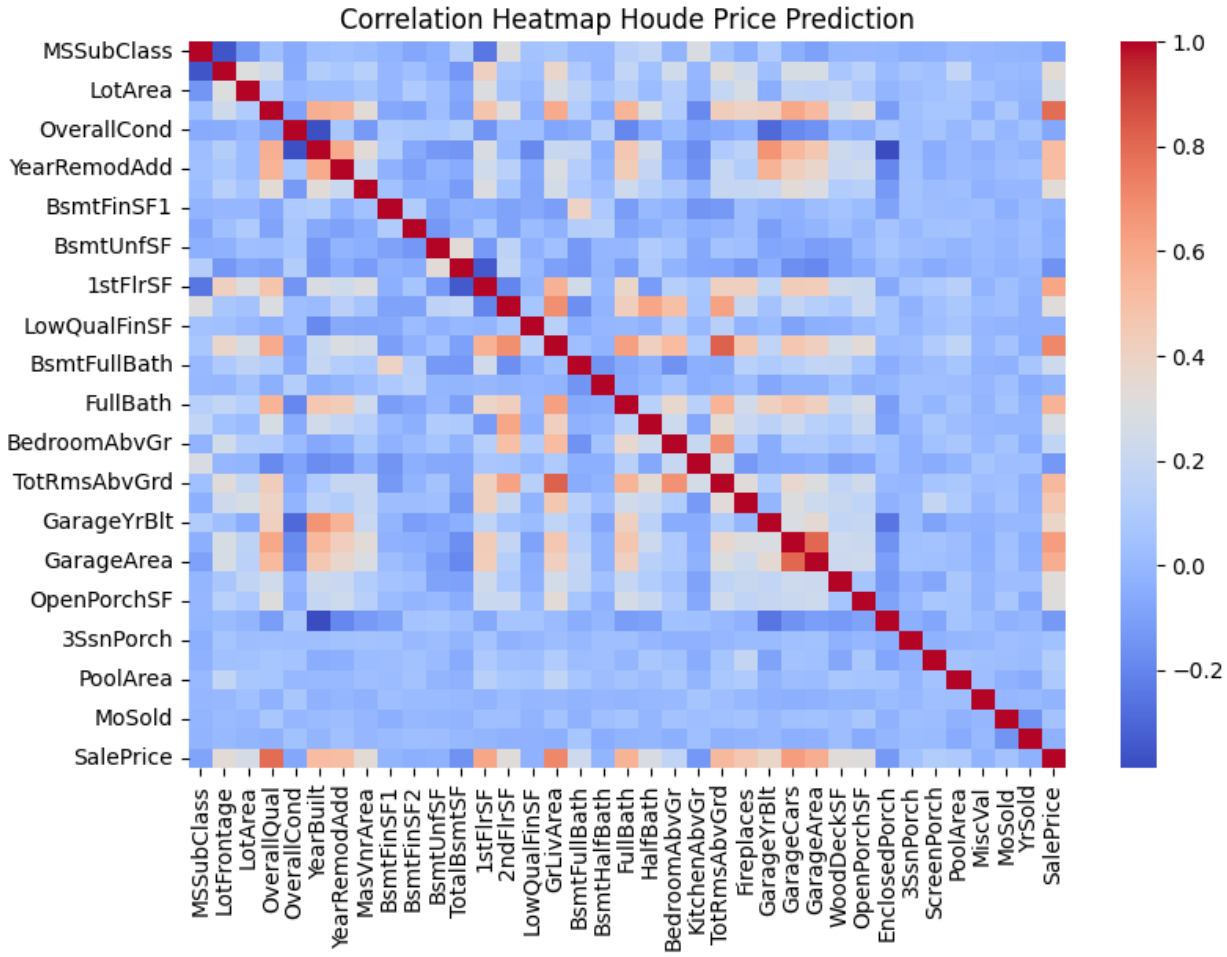

Distribution of MiscVal
Frequency
MiscVal
0
1000
2000
3000
4000
5000
6000
0
2000
4000
6000
8000
10000
12000
14000
16000


Distribution of MoSold
Frequency
MoSold
0
50
100
150
200
250
2
4
6
8
10
12

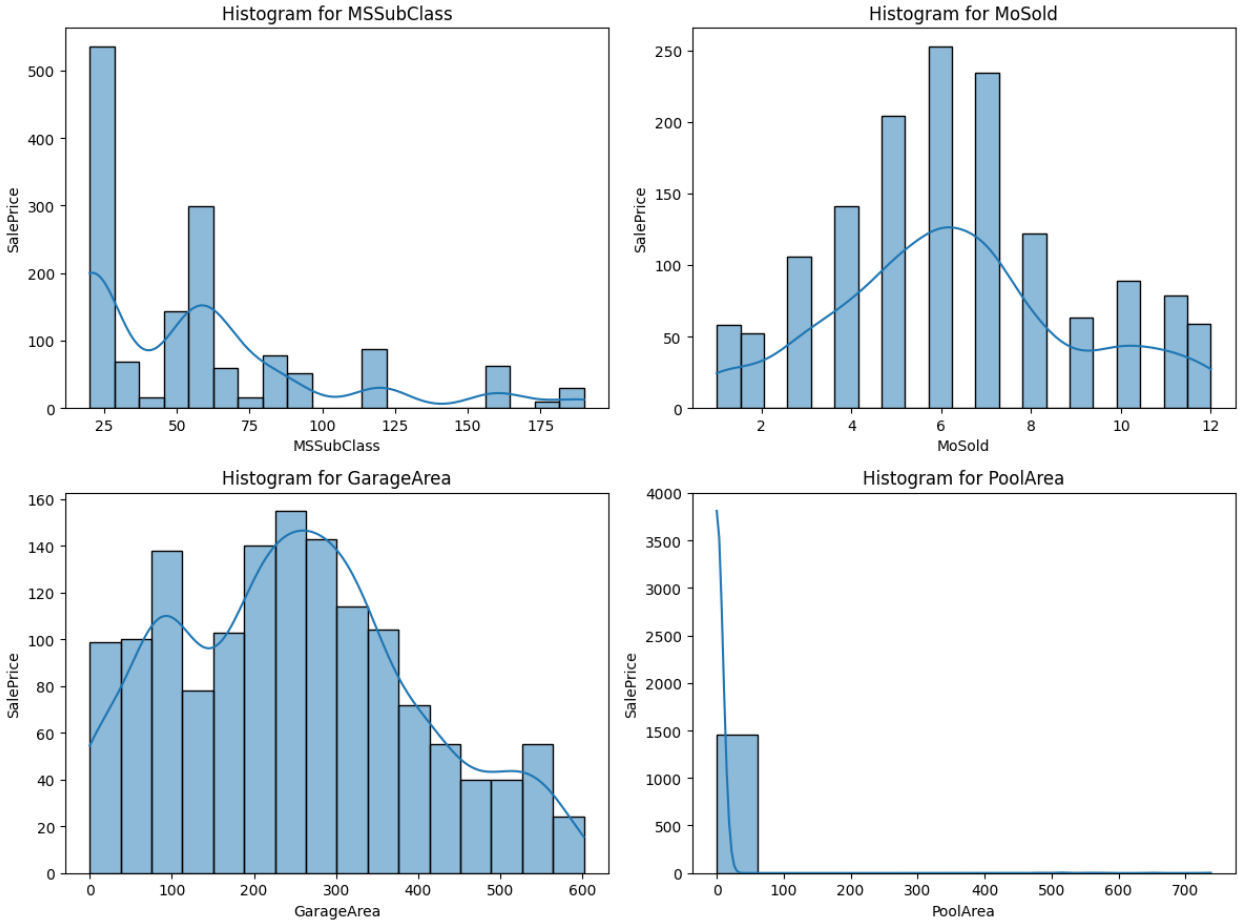

Distribution of YrSold
Frequency
YrSold
0
50
100
150
200
250
300
350
2006.0
2006.5
2007.0
2007.5
2008.0
2008.5
2009.0
2009.5
2010.0



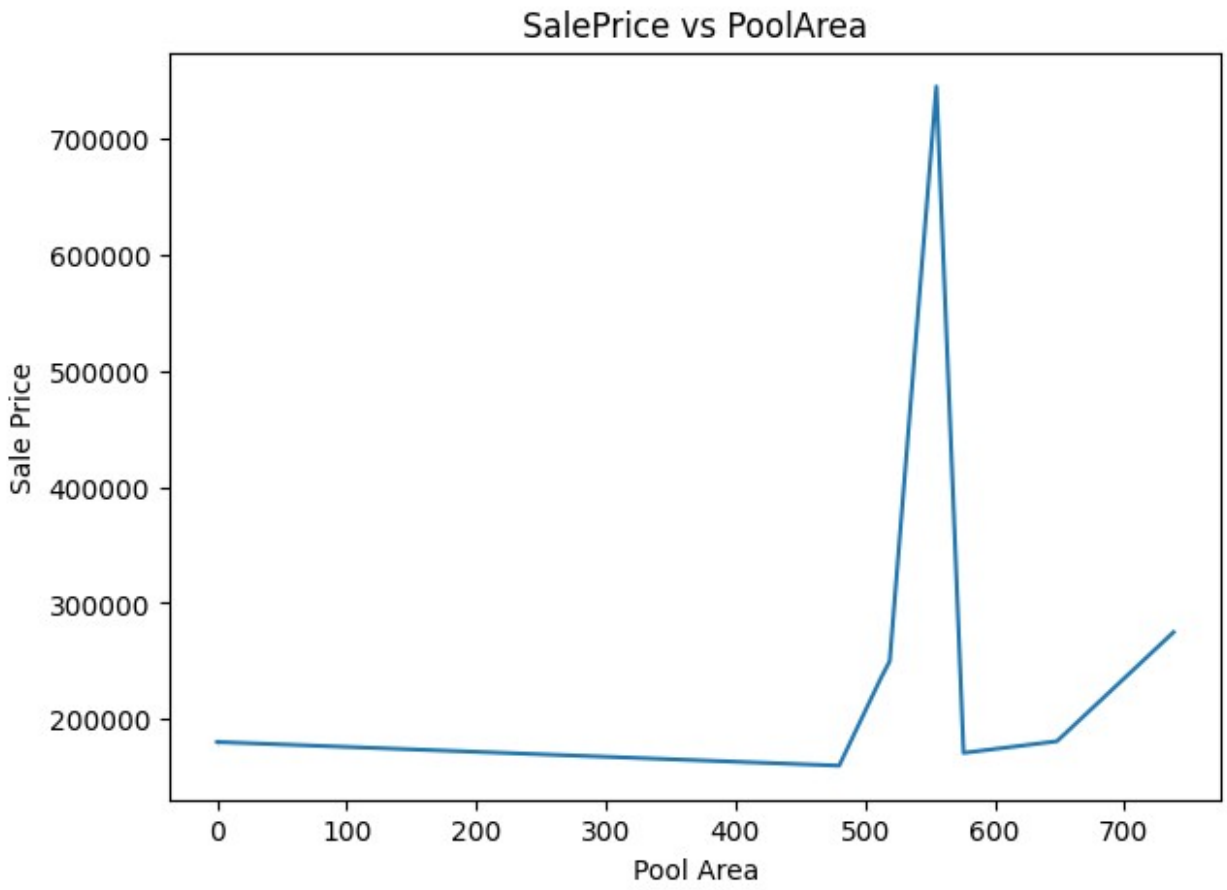

```
# Correlation heatmap  for hp dataset
plt.figure(figsize=(9,6))
sns.heatmap(hp_train[numerical_columns_hp]

.corr(), annot=False, cmap='coolwarm')
plt.title("Correlation Heatmap Houde Price Prediction")
plt.show()
```



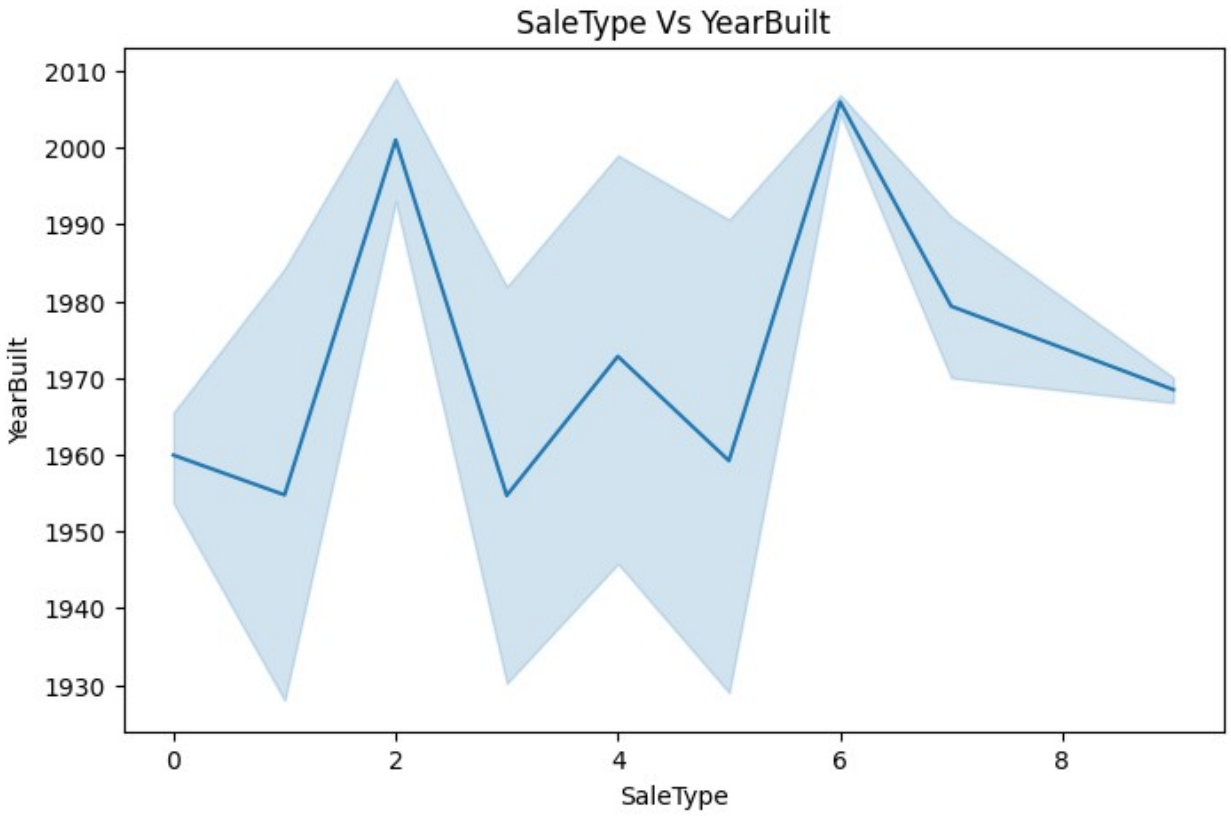

```
plt.figure(figsize=(12,9))

# List of continuous variables
variables = [ 'MSSubClass', 'MoSold','GarageArea',
       'PoolArea' ]

# Creating a histogram with KDE for each continuous variable
for i, variable in enumerate(variables, 1):
    plt.subplot(2,2,i)
    sns.histplot(hp_train[variable], kde=True)
    plt.title(f'Histogram for {variable}')
    plt.xlabel(variable)
    plt.ylabel('SalePrice')

# Adjusting the layout for better readability
plt.tight_layout()
plt.show()
```



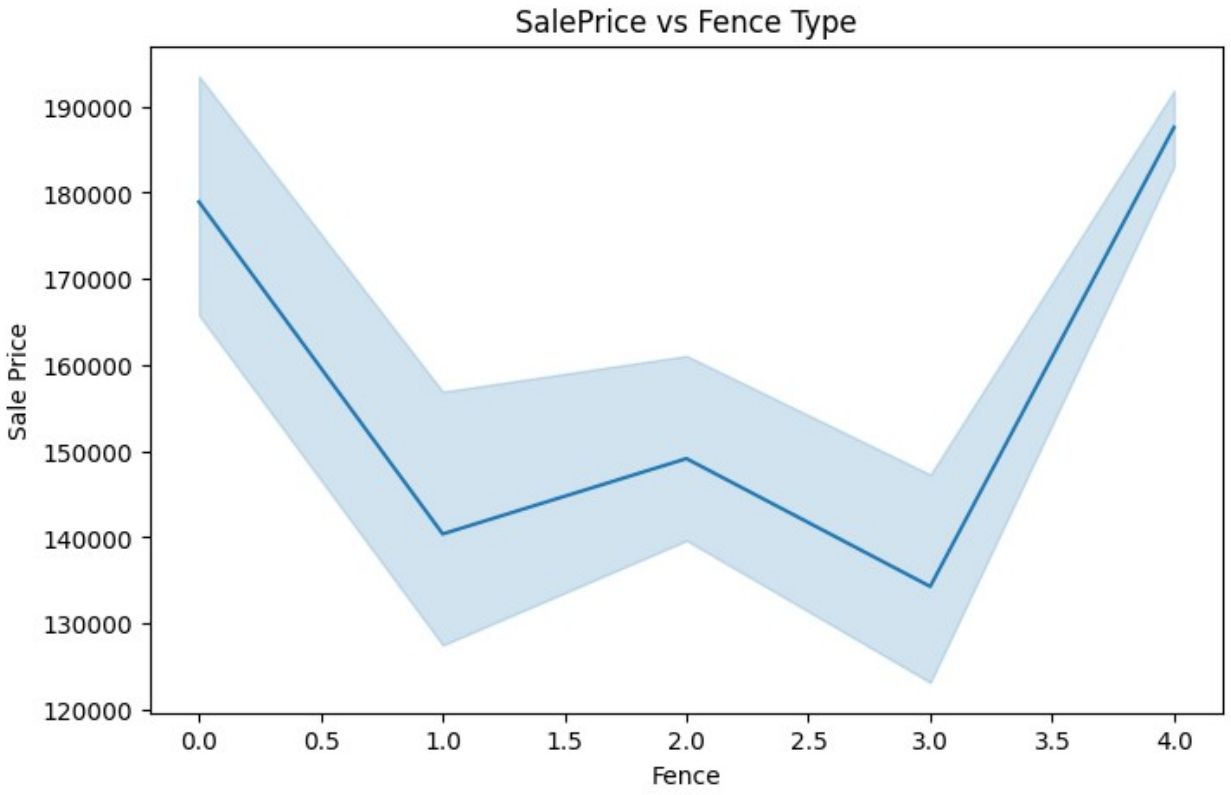

```
# Line Plot for Sale Price Vs PoolArea
plt.figure(figsize=(7,5))
sns.lineplot(x='PoolArea', y='SalePrice',data=hp_train)
plt.title("SalePrice vs PoolArea")
plt.xlabel("Pool Area")
plt.ylabel("Sale Price")
plt.show()
```



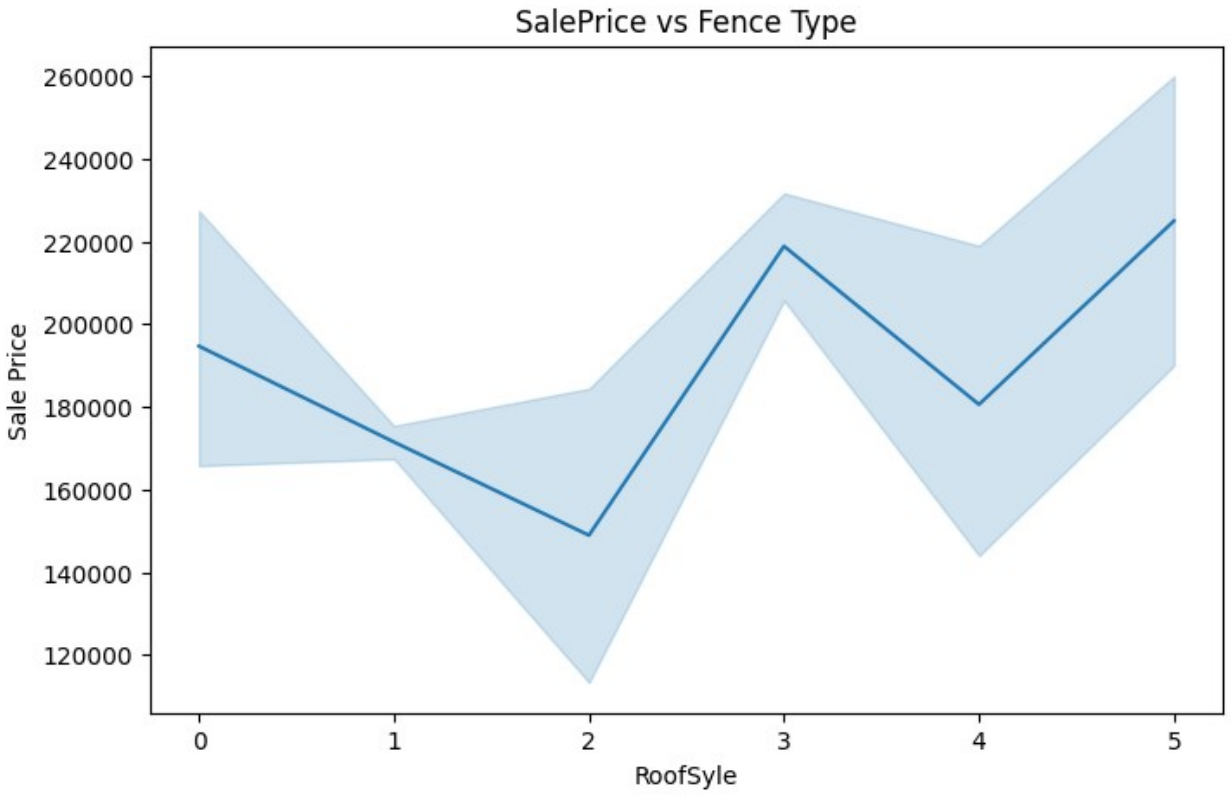

```
#Line plot of Year Built per SaleType
plt.figure(figsize=(8,5))
sns.lineplot(x='SaleType',y='YearBuilt',data=hp_train)
plt.title("SaleType Vs YearBuilt")
plt.xlabel("SaleType")
plt.ylabel("YearBuilt")
plt.show()
```



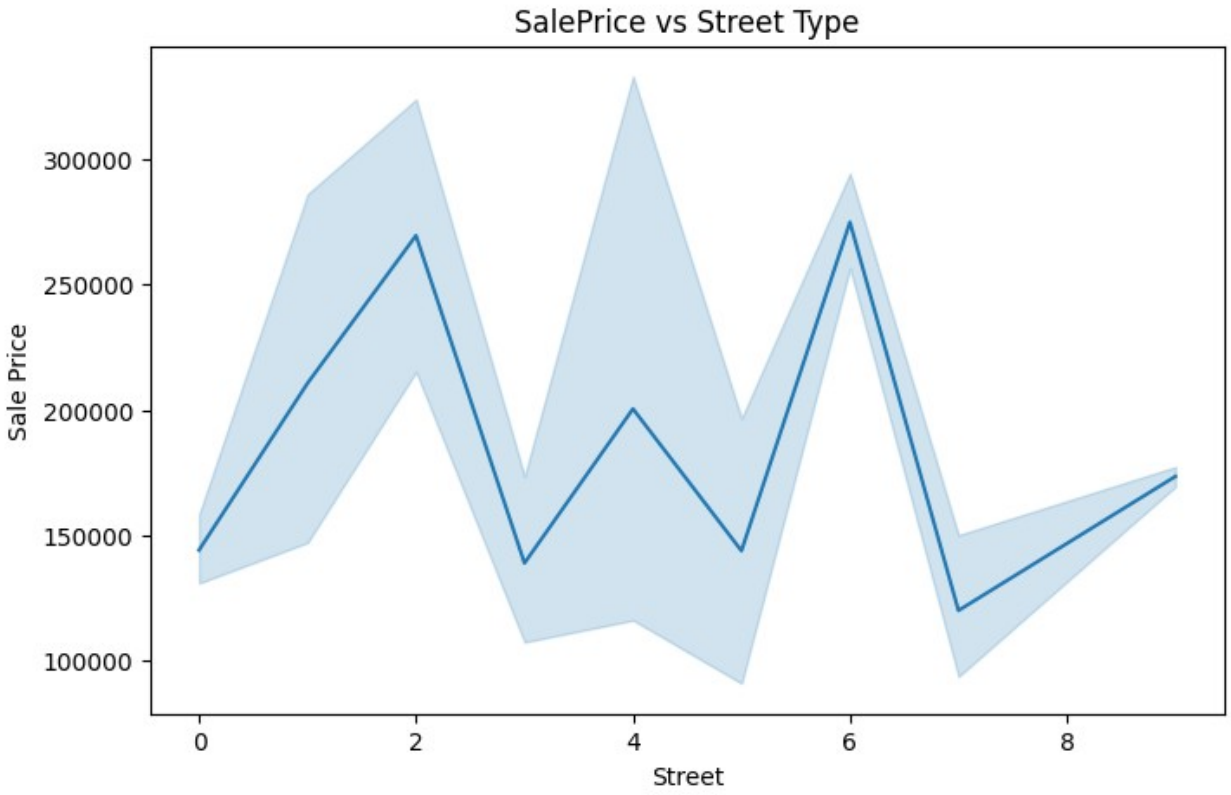

```
#Line plot for Sale and Fence Type
plt.figure(figsize=(8,5))
sns.lineplot(x='Fence', y='SalePrice',data=hp_train)
plt.title("SalePrice vs Fence Type")
plt.xlabel("Fence")
plt.ylabel("Sale Price")
plt.show()
```



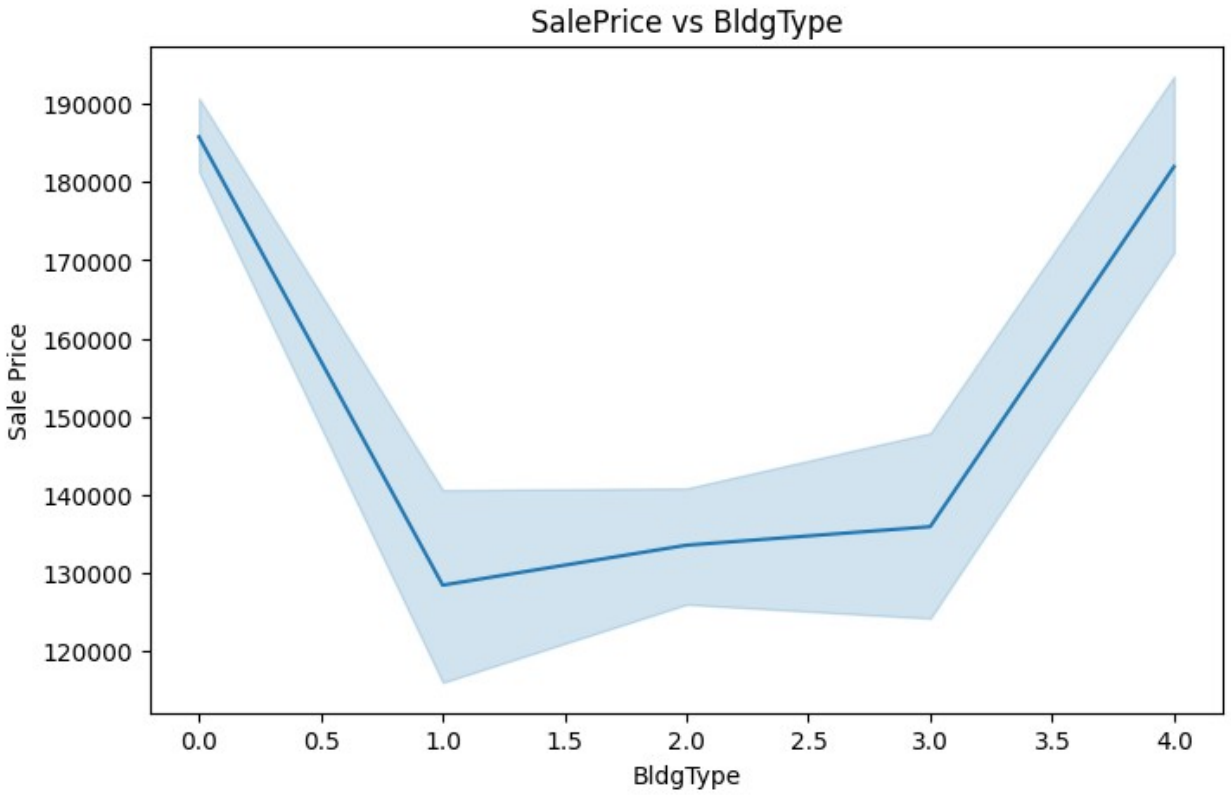

```
#Line plot for Sale and Roofstlye Type
plt.figure(figsize=(8,5))
sns.lineplot(x='RoofStyle', y='SalePrice',data=hp_train)
plt.title("SalePrice vs Fence Type")
plt.xlabel("RoofSyle")
plt.ylabel("Sale Price")
plt.show()
```




```
#Pair plot for Sales Price by Street(Pvd or Grvl)
plt.figure(figsize=(8,5))
sns.lineplot(x='SaleType', y='SalePrice',data=hp_train)
plt.title("SalePrice vs Street Type")
plt.xlabel("Street")
plt.ylabel("Sale Price")
plt.show()
```




```
# line Plot Showing relation btn Saleprice and Building Type
plt.figure(figsize=(8,5))
sns.lineplot(x='BldgType', y='SalePrice',data=hp_train)
plt.title("SalePrice vs BldgType")
plt.xlabel("BldgType")
plt.ylabel("Sale Price")
plt.show()
```




```
hp_train.fillna(X_train.mode(), inplace=True)
hp_train.fillna(X_test.mode(), inplace=True)

# Target Variable for hp data
y = hp_train['SalePrice']
X = hp_train.drop (columns=['SalePrice'])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Define models/Algorthim for hp
models = {
    "Linear Regression": LR(),
    "Random Forest": RandomForestRegressor(n_estimators=100,
random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, random_state=42)
}

# Train and evaluate models for hp
X = hp_train.drop(columns=['SalePrice'])
y = hp_train['SalePrice']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Train the model (Linear Regression)
lr_model_hp = LR()
lr_model_hp.fit(X_train, y_train)

y_pred = lr_model_hp.predict(X)
rmse_HP = np.sqrt(mean_squared_error(y, y_pred))
print("RMSE (Linear Regression):", rmse_HP)
```

```
RMSE (Linear Regression): 31015.231006256086
```

```
# Train the model(Random Forest)
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Evaluate the model
y_pred_rf = rf_model.predict(X)
rmse_rf = np.sqrt(mean_squared_error(y, y_pred_rf))
print("RMSE (Random Forest):", rmse_rf)
```

```
RMSE (Random Forest): 16462.468112659837
```

```
# Train the model(Gradient Boosting)
gb_model = GradientBoostingRegressor(n_estimators=100,
random_state=42)
gb_model.fit(X_train, y_train)

# Evaluate the model
y_pred_gb = gb_model.predict(X)
rmse_gb = np.sqrt(mean_squared_error(y, y_pred_gb))
print("RMSE (Gradient Boosting):", rmse_gb)
```

```
RMSE (Gradient Boosting): 17549.956776289975
```

```
# Cross-validation for Random Forest
cv_scores_hp_rf = cross_val_score(rf_model, X, y, cv=5,
scoring='neg_mean_squared_error')
print("Cross-Validation Scores (Random Forest):", -cv_scores_hp_rf)
```

```
Cross-Validation Scores (Random Forest): [7.20387115e+08
1.04758438e+09 9.90863007e+08 6.66064941e+08
 1.30574621e+09]
```

```
# Hyperparameter tuning for Random Forest
param_dist = {
    'n_estimators': [50, 100, 150],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10]
}
rf_random = RandomizedSearchCV(RandomForestRegressor(random_state=42),
param_dist, n_iter=10, cv=3, random_state=42)
rf_random.fit(X_train, y_train)
```

```
print("Best Parameters for Random Forest:", rf_random.best_params_)
```

```
Best Parameters for Random Forest: {'n_estimators': 100,
'min_samples_split': 5, 'max_depth': None}
```

```
# Save the trained model  for hp predictions
with open('house_price_model_Charles.pkl', 'wb') as file:
    jb.dump(RandomForestRegressor, file)

print(" Model saved successfully!")
```

```
 Model saved successfully!
```

```
# Load the test dataset for hp
hp_test =
pd.read_csv('/Users/ishimwecharleshagenimana/Desktop/Project_DS_FINAL/
test.csv')

# Ensure the test set has the same preprocessing as the training set
# Handling missing values (Fill missing numeric values with median)
hp_test['LotFrontage'].fillna (hp_test['LotFrontage'].median(),
inplace=True)
hp_test['GarageType'].fillna('None', inplace=True)

# Feature Engineering (Creating new features)
hp_test['TotalSF'] = hp_test['1stFlrSF'] + hp_test['2ndFlrSF'] +
hp_test['GrLivArea']
hp_test['Age'] = hp_test['YrSold'] - hp_test['YearBuilt']
hp_test['RemodelAge'] = hp_test['YrSold'] - hp_test['YearRemodAdd']

# Encoding Categorical Features (One-Hot Encoding)
hp_test= pd.get_dummies(hp_test, columns=['Neighborhood'],
drop_first=True)

# Ensure test dataset has the same columns as training dataset
# Drop any extra columns that might not be in training
missing_cols = [col for col in X_train.columns if col not in
hp_test.columns]
for col in missing_cols:
    hp_test[col] = 0

# Reorder columns to match training data
test_df = hp_test[X_train.columns]

# Scaling Numeric Features
scaler = StandardScaler()
hp_test[['LotArea', 'GrLivArea', 'TotalBsmtSF']] =
scaler.fit_transform(hp_test[['LotArea', 'GrLivArea', 'TotalBsmtSF']])

# Load the trained model (Assuming Random Forest is the best model)
```

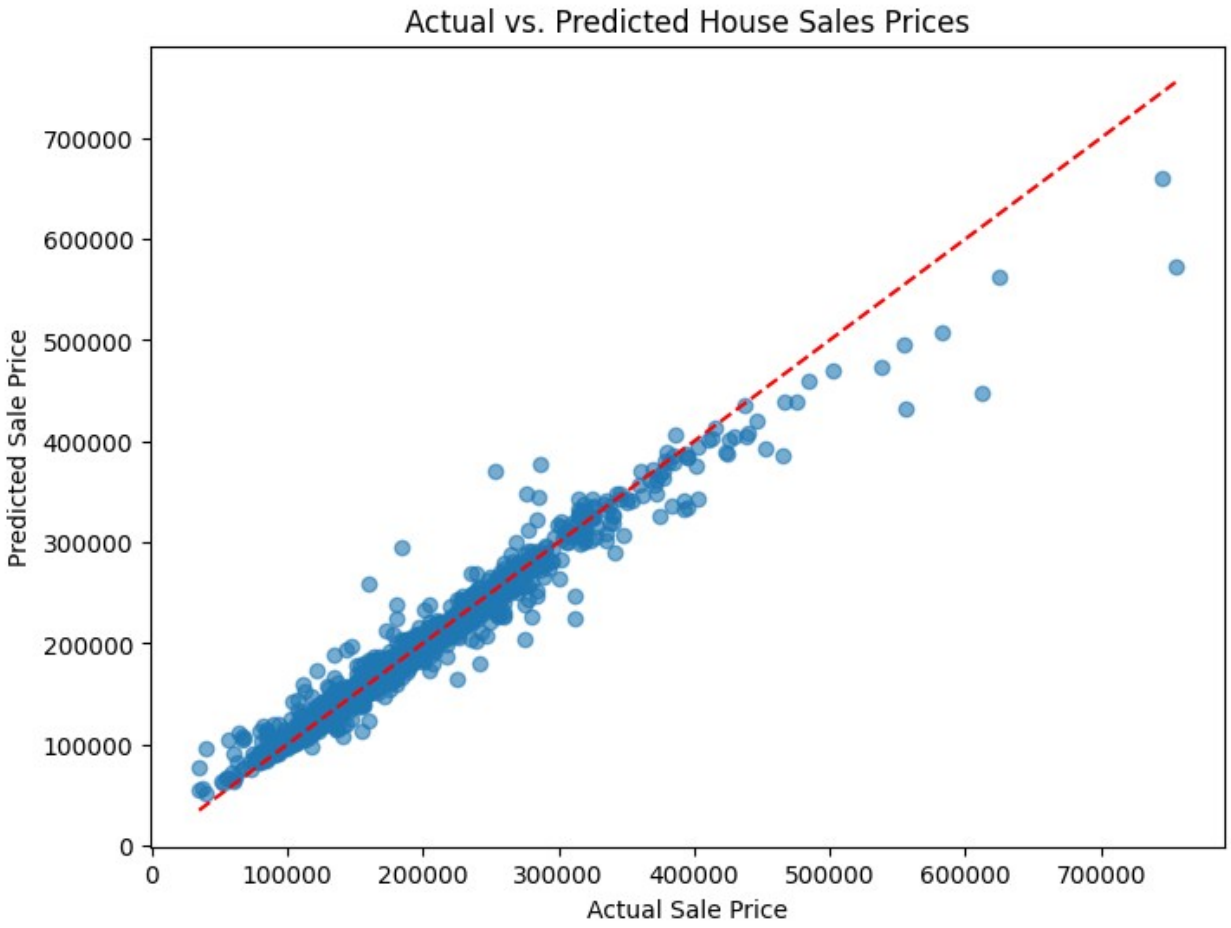
```
rf_model =
jb.load('/Users/ishimwecharleshagenimana/Desktop/house_price_mdl.pkl')

# Predict on the test set
rf_model=RandomForestRegressor()
rf_model.fit(X_train,y_train)
test_predictions_hp = rf_model.predict(X_test)

# Ensure no negatives & round values to whole numbers
test_predictions_hp = np.maximum(test_predictions_hp, 0)  # Remove
negative values
test_predictions_hp = np.round(test_predictions_hp)  # Round to
nearest integer

# Create a DataFrame for predictions
submission = pd.DataFrame({'Id': X_test.index, 'Predicted_SalePrice':
test_predictions_hp})

# Save to CSV
submission.to_csv('house_price_predictions_Charles.csv', index=False)

print(" Test predictions saved successfully to
'house_price_predictions_Charles.csv'!")
```

```
/var/folders/gm/5fc6tr_x7lvgmrffjwkdqz140000gn/T/
ipykernel_47273/3113334078.py:6: 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.


  hp_test['LotFrontage'].fillna (hp_test['LotFrontage'].median(),
inplace=True)
/var/folders/gm/5fc6tr_x7lvgmrffjwkdqz140000gn/T/ipykernel_47273/31133
34078.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.


  hp_test['GarageType'].fillna('None', inplace=True)

---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
Cell In[92], line 32
     28 hp_test[['LotArea', 'GrLivArea', 'TotalBsmtSF']] =
scaler.fit_transform(hp_test[['LotArea', 'GrLivArea', 'TotalBsmtSF']])
     30 # Load the trained model (Assuming Random Forest is the best
model)
---> 32 rf_model =
jb.load('/Users/ishimwecharleshagenimana/Desktop/house_price_mdl.pkl')
     34 # Predict on the test set
     35 rf_model=RandomForestRegressor()

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/joblib/numpy_pickle.py:650, in load(filename, mmap_mode)
    648         obj = _unpickle(fobj)
    649 else:
--> 650     with open(filename, 'rb') as f:
    651         with _read_fileobject(f, filename, mmap_mode) as fobj:
    652             if isinstance(fobj, str):
    653                 # if the returned file object is a string,
this means we
    654                 # try to load a pickle file generated with an
version of
    655                 # Joblib so we load it with joblib
compatibility function.

FileNotFoundError: [Errno 2] No such file or directory:
'/Users/ishimwecharleshagenimana/Desktop/house_price_mdl.pkl'


#Scatter Plot for the Actual and Predicted Values for House Price
Prediction

# Make predictions on the validation set
y_pred = rf_model.predict(X)

# Create a DataFrame to compare actual and predicted values
comparison_hp = pd.DataFrame({'Actual': y, 'Predicted': y_pred})

# Display the first few rows
```

```
print(comparison_hp.head())

# Calculate error metrics
mae = mean_absolute_error(y, y)
rmse = np.sqrt(mean_squared_error(y, y_pred))

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

# Plot Actual vs. Predicted Values
plt.figure(figsize=(8, 6))
plt.scatter(y, y_pred, alpha=0.6)
plt.plot([min(y), max(y)], [min(y), max(y)], color='red',
linestyle='dashed')
plt.xlabel("Actual Sale Price")
plt.ylabel("Predicted Sale Price")
plt.title("Actual vs. Predicted House Sales Prices")
plt.show()
```

```
     Actual  Predicted
0  208500.0  207439.75
1  181500.0  172012.00
2  223500.0  220988.78
3  140000.0  154779.00
4  250000.0  266378.17
Mean Absolute Error (MAE): 0.00
Root Mean Squared Error (RMSE): 16462.47
```


Actual vs. Predicted House Sales Prices
Predicted Sale Price
Actual Sale Price
0
100000
200000
300000
400000
500000
600000
700000