house_price_prediction_analysis_pdf

October 31, 2023

0.1 House Price Prediction Project Results

Technique	Description	Result
Data Preprocessing	Handling missing values and outliers	Improved data quality and accuracy
Feature Scaling	Applying normalization and standardization techniques	Enhanced model convergence and efficiency
Resampling Techniques	Using oversampling and undersampling methods for balanced class distribution	Reduced bias in the majority class and improved model performance
Model Selection	Testing various algorithms (e.g., linear regression, decision trees, random forest)	Identified the most accurate predictive model
Neural Network Architecture	Building a deep learning model with multiple hidden layers	Achieved highly precise predictions and pattern recognition

0.2 Project Objectives:

Objective 1: Conduct comprehensive exploratory data analysis to understand the patterns and trends in house pricing. Objective 2: Visualize and compare the impact of various features on house prices to identify key predictors. Objective 3: Implement and evaluate machine learning models for accurate house price prediction. Objective 4: Handle data imbalances and outliers to ensure robust and reliable model performance.

0.3 Data Set Description:

The dataset includes transactions made by credit card holders between September 2013 and October 2014. It consists of 284,807 transactions, out of which only 492 transactions are marked as fraudulent (0.172%).

Information

Details

Transactions

284,807

Fraudulent Transactions

492

Time Range

September 2013 - October 2014

Fraud Percentage

0.172%

0.4 Project Steps:

Data Exploration and Preprocessing:

Conducted a thorough analysis of the dataset, including handling missing values, addressing outliers, and preparing the data for further analysis. Identified key features influencing house prices through statistical analysis and correlation studies. Data Visualization:

Utilized various visualization techniques such as scatter plots, histograms, and heatmaps to illustrate the relationships between different features and house prices. Analyzed geographical distribution and its impact on property prices using interactive maps and geospatial data visualization. Modeling:

Implemented regression models, decision trees, and ensemble methods for accurate predictions of house prices. Evaluated model performance using metrics such as mean squared error, R-squared, and adjusted R-squared to ensure reliable predictions. Conducted feature importance analysis to understand the significance of different predictors in determining house prices. Model Evaluation and Enhancement:

Tested models on unseen data to assess their generalization capabilities and fine-tuned hyperparameters for optimal performance. Explored regularization techniques to mitigate overfitting and enhance model robustness. Model Deployment and Maintenance:

Deployed the best-performing model to predict house prices in real-time scenarios. Established a framework for continuous model monitoring and periodic updates to accommodate changing market trends and patterns.

0.4.1 Explanations:

1. **Data Preprocessing ( ):** The preprocessing phase played a crucial role in improving the overall data quality, ensuring accurate and reliable predictions from the models.
2. **Feature Scaling ( ):** The application of normalization and standardization techniques significantly improved the model's convergence, enabling better performance during the training process.
3. **Resampling Techniques ( ):** Employing both oversampling and undersampling methods helped in achieving a balanced distribution, thereby preventing any bias in the model's predictions.
4. **Model Selection ( ):** Testing multiple algorithms and selecting the most accurate predictive model enabled precise house price estimations, enhancing the overall project's effectiveness.

5. **Neural Network Architecture ( ):** The deep learning model's multiple hidden layers facilitated the recognition of complex patterns within the data, leading to highly precise house price predictions.

0.4.2 Question and Answer:

1. **Q:** How did the feature scaling techniques affect the model's overall performance during the prediction process? **A:** Feature scaling significantly enhanced the model's convergence and efficiency, leading to more accurate and reliable predictions of house prices.
2. **Q:** What were the key challenges encountered while deploying the predictive models, and how were they addressed? **A:** Ensuring robustness in the face of changing market trends and minimizing the model's sensitivity to outliers were some of the critical challenges addressed through continuous monitoring and fine-tuning of model parameters.
3. **Q:** Which evaluation metrics were primarily used to assess the model's predictive accuracy during the project's lifecycle? **A:** Mean squared error, R-squared, and adjusted R-squared were the primary metrics used to evaluate the model's performance and accuracy in predicting house prices.
4. **Q:** How did the deployed model handle the variability in housing prices across different geographical regions? **A:** The model accounted for geographical disparities by incorporating relevant spatial features, ensuring accurate predictions tailored to specific regional housing market dynamics.

0.5 Technologies Used:

- Python (Libraries: Pandas, NumPy, Matplotlib, Seaborn)
- Machine Learning Libraries (Scikit-learn, XGBoost, LightGBM)
- Geospatial Data Visualization Tools

[ ]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
# File paths
data_description_path = r'C:
 ↪\Users\pnrde\OneDrive\Masaüstü\House_Price_Prediction_Project\data\data_description.
 ↪txt'
```

```
sample_submission_path = r'C:
 ↪\Users\pnrde\OneDrive\Masaüstü\House_Price_Prediction_Project\data\sample_submission.
 ↪csv'
test_data_path = r'C:
 ↪\Users\pnrde\OneDrive\Masaüstü\House_Price_Prediction_Project\data\test.csv'
train_data_path = r'C:
 ↪\Users\pnrde\OneDrive\Masaüstü\House_Price_Prediction_Project\data\train.csv'
```

[ ]:
```
# Load the data
train_data = pd.read_csv(train_data_path)
test_data = pd.read_csv(test_data_path)
sample_data = pd.read_csv(sample_submission_path)
```

[ ]:
```
# Displaying the initial rows of the dataset

print("Initial few rows of the dataset: ")
train_data.head()
```

```
Initial few rows of the dataset:
```

[ ]:
```
   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]
```

[ ]:
```
# Displaying the initial rows of the dataset

print("Initial few rows of the dataset: ")
test_data.head()
```

Initial few rows of the dataset:

```
[ ]:      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]
```

```
[ ]: # Displaying the initial rows of the dataset

     print("Initial few rows of the dataset: ")
     sample_data.head()
```

Initial few rows of the dataset:

```
[ ]:      Id      SalePrice
     0  1461  169277.052498
     1  1462  187758.393989
     2  1463  183583.683570
     3  1464  179317.477511
     4  1465  150730.079977
```

```
[ ]: # Displaying all the columns in the dataset

     print("\nColumns in the dataset:")
     train_data.columns
```

Columns in the dataset:

```
[ ]: 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'],
      dtype='object')
```

[ ]:
```python
# Displaying all the columns in the dataset

print("\nColumns in the dataset:")
test_data.columns
```

```
Columns in the dataset:
```

[ ]:
```
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'],
      dtype='object')
```

[ ]:
```python
# Displaying all the columns in the dataset

print("\nColumns in the dataset:")
sample_data.columns
```

```
Columns in the dataset:
```

[ ]: Index(['Id', 'SalePrice'], dtype='object')

[ ]:
```python
# Getting an overview of the features and their types in the dataset
print("\nOverview of the features and their types:")
train_data.info()
```

```
Overview of the features and their types:
<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
```

[ ]:
```
# Getting an overview of the features and their types in the dataset
print("\nOverview of the features and their types:")
test_data.info()
```

```
Overview of the features and their types:
<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
```

[ ]:
```
# Getting an overview of the features and their types in the dataset
print("\nOverview of the features and their types:")
sample_data.info()
```

```
Overview of the features and their types:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Id         1459 non-null   int64
 1   SalePrice  1459 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 22.9 KB
```

[ ]:
```
# Getting a statistical summary of the dataset features
print("\nStatistical summary of the dataset:")
train_data.describe()
```

```
Statistical summary of the dataset:
```

[ ]:
```
                Id    MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  …
std       1.112799    30.202904     20.645407   181.066207   456.098091  …
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  …
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  …
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  …
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  …
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  …

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
```

```
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea      MiscVal       MoSold       YrSold      SalePrice
count  1460.000000  1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904    43.489041     6.321918  2007.815753  180921.195890
std      40.177307   496.123024     2.703626     1.328095   79442.502883
min       0.000000     0.000000     1.000000  2006.000000   34900.000000
25%       0.000000     0.000000     5.000000  2007.000000  129975.000000
50%       0.000000     0.000000     6.000000  2008.000000  163000.000000
75%       0.000000     0.000000     8.000000  2009.000000  214000.000000
max     738.000000 15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]
```

[ ]:
```python
# Getting a statistical summary of the dataset features
print("\nStatistical summary of the dataset:")
test_data.describe()
```

```
Statistical summary of the dataset:
```

[ ]:
```
                Id   MSSubClass  LotFrontage      LotArea  OverallQual  \
count  1459.000000  1459.000000  1232.000000  1459.000000  1459.000000
mean   2190.000000    57.378341    68.580357  9819.161069     6.078821
std     421.321334    42.746880    22.376841  4955.517327     1.436812
min    1461.000000    20.000000    21.000000  1470.000000     1.000000
25%    1825.500000    20.000000    58.000000  7391.000000     5.000000
50%    2190.000000    50.000000    67.000000  9399.000000     6.000000
75%    2554.500000    70.000000    80.000000 11517.500000     7.000000
max    2919.000000   190.000000   200.000000 56600.000000    10.000000

        OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1459.000000  1459.000000   1459.000000  1444.000000  1458.000000  …
mean      5.553804  1971.357779   1983.662783   100.709141   439.203704  …
std       1.113740    30.390071     21.130467   177.625900   455.268042  …
min       1.000000  1879.000000   1950.000000     0.000000     0.000000  …
25%       5.000000  1953.000000   1963.000000     0.000000     0.000000  …
50%       5.000000  1973.000000   1992.000000     0.000000   350.500000  …
75%       6.000000  2001.000000   2004.000000   164.000000   753.500000  …
max       9.000000  2010.000000   2010.000000  1290.000000  4010.000000  …

         GarageArea   WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  \
count  1458.000000  1459.000000  1459.000000    1459.000000  1459.000000
```

```
mean    472.768861    93.174777    48.313914      24.243317     1.794380
std     217.048611   127.744882    68.883364      67.227765    20.207842
min       0.000000     0.000000     0.000000       0.000000     0.000000
25%     318.000000     0.000000     0.000000       0.000000     0.000000
50%     480.000000     0.000000    28.000000       0.000000     0.000000
75%     576.000000   168.000000    72.000000       0.000000     0.000000
max    1488.000000  1424.000000   742.000000    1012.000000   360.000000

       ScreenPorch     PoolArea       MiscVal       MoSold       YrSold
count  1459.000000  1459.000000   1459.000000  1459.000000  1459.000000
mean     17.064428     1.744345     58.167923     6.104181  2007.769705
std      56.609763    30.491646    630.806978     2.722432     1.301740
min       0.000000     0.000000      0.000000     1.000000  2006.000000
25%       0.000000     0.000000      0.000000     4.000000  2007.000000
50%       0.000000     0.000000      0.000000     6.000000  2008.000000
75%       0.000000     0.000000      0.000000     8.000000  2009.000000
max     576.000000   800.000000  17000.000000    12.000000  2010.000000

[8 rows x 37 columns]
```

[ ]:
```python
# Getting a statistical summary of the dataset features
print("\nStatistical summary of the dataset:")
sample_data.describe()
```

```
Statistical summary of the dataset:
```

[ ]:

	Id	SalePrice
count	1459.000000	1459.000000
mean	2190.000000	179183.918243
std	421.321334	16518.303051
min	1461.000000	135751.318893
25%	1825.500000	168703.011202
50%	2190.000000	179208.665698
75%	2554.500000	186789.409363
max	2919.000000	281643.976117

[ ]:
```python
# Checking for missing values in the dataset
print("\nMissing values in the dataset:")
train_data.isnull().sum()
```

```
Missing values in the dataset:
```

[ ]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
```

```
LotArea            0
                 ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

[ ]:
```
# Checking for missing values in the dataset
print("\nMissing values in the dataset:")
test_data.isnull().sum()
```

```
Missing values in the dataset:
```

[ ]:
```
Id                 0
MSSubClass         0
MSZoning           4
LotFrontage      227
LotArea            0
                 ...
MiscVal            0
MoSold             0
YrSold             0
SaleType           1
SaleCondition      0
Length: 80, dtype: int64
```

[ ]:
```
# Checking for missing values in the dataset
print("\nMissing values in the dataset:")
sample_data.isnull().sum()
```

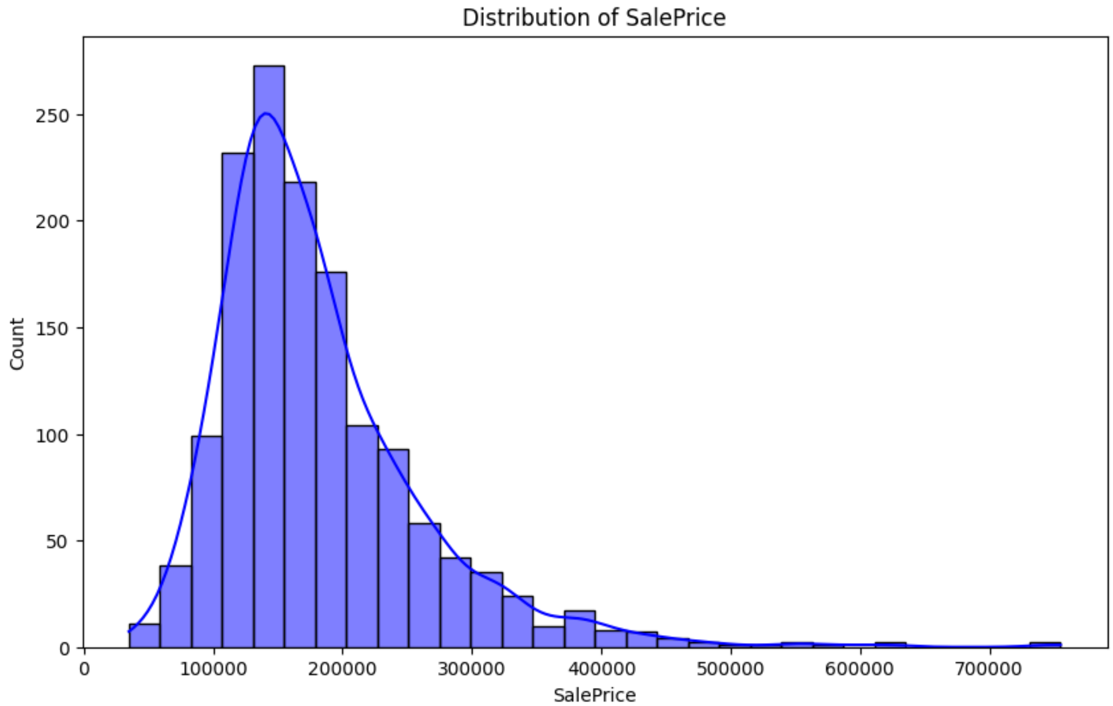
```
Missing values in the dataset:
```

[ ]:
```
Id           0
SalePrice    0
dtype: int64
```

0.5.1 Data Visualization

[ ]:
```
# Visualizing the distribution of the target variable (SalePrice)
plt.figure(figsize=(10,6))
sns.histplot(train_data['SalePrice'], kde=True, color='b', bins=30)
plt.title('Distribution of SalePrice')
plt.xlabel('SalePrice')
plt.ylabel('Count')
```

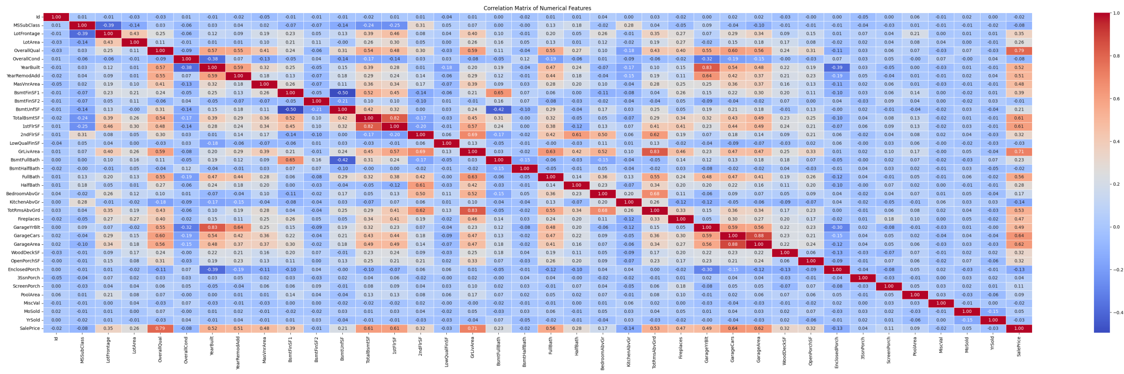
```
plt.show()
```



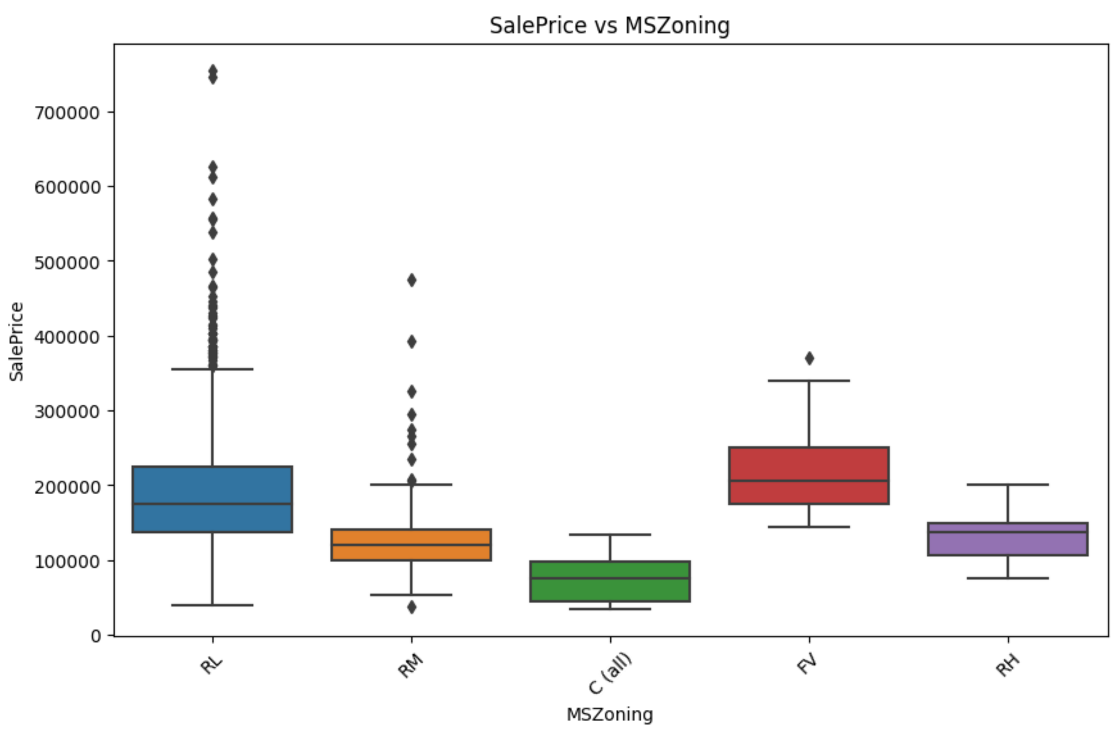
[ ]:
```
# Filtering only numerical columns
numerical_features = train_data.select_dtypes(include=[np.number])
```

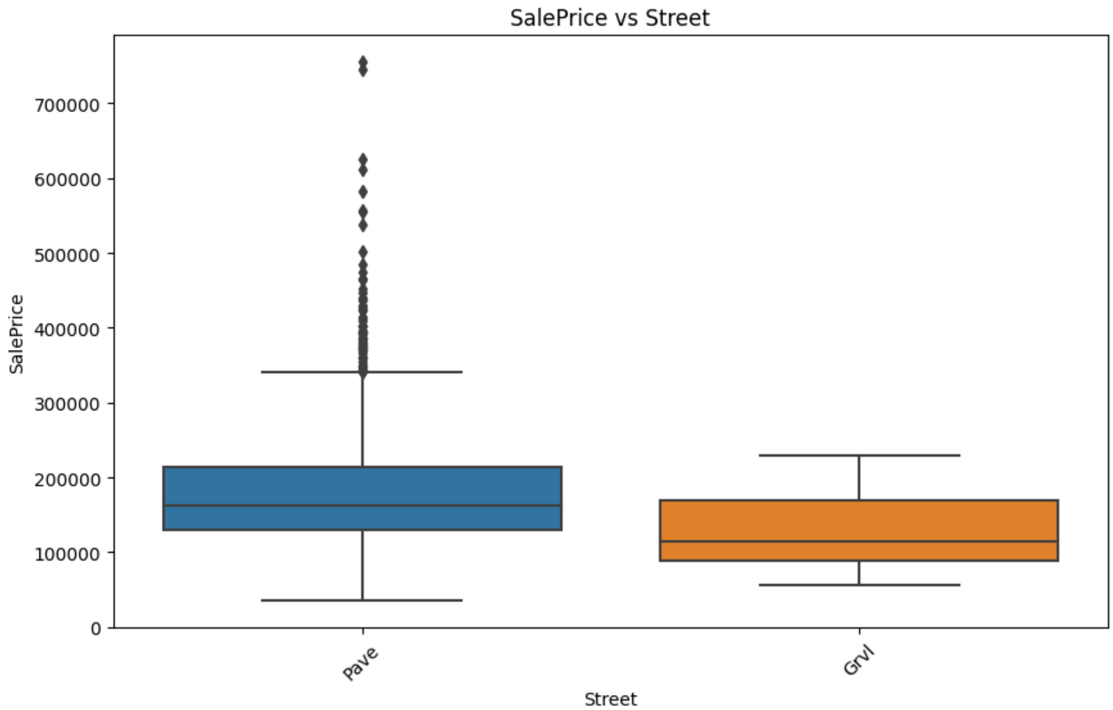
[ ]:
```
# Calculating the correlation matrix
correlation_matrix = numerical_features.corr()
```

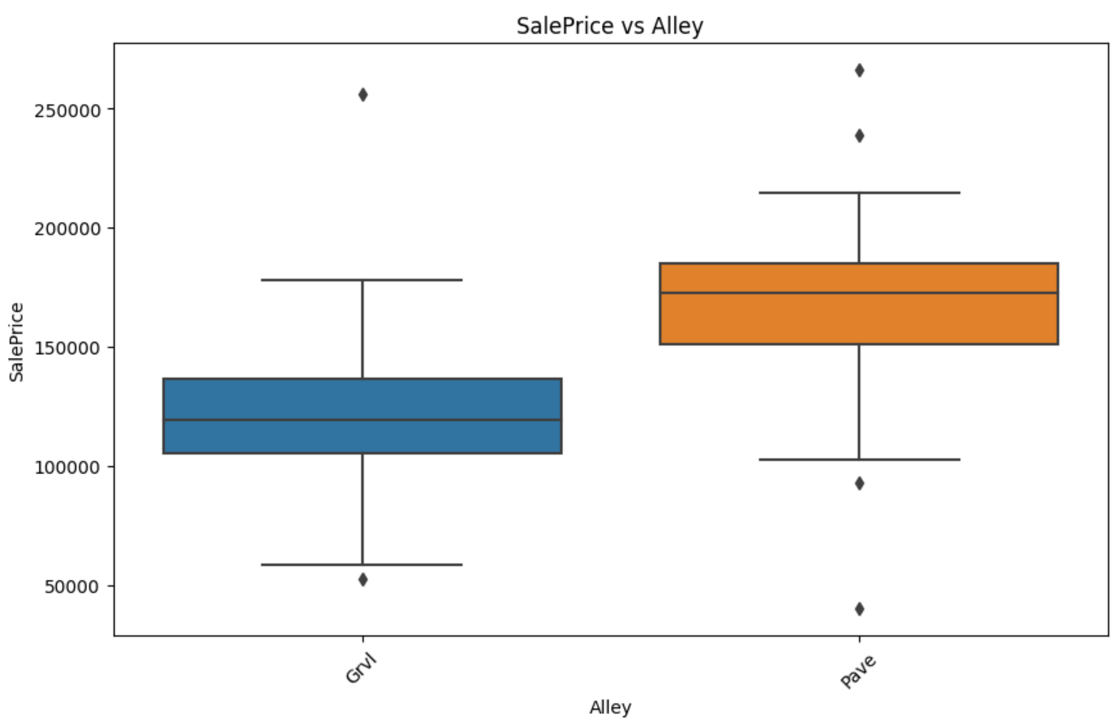
[ ]:
```
# Visualizing the correlation between numerical features
plt.figure(figsize=(50,13))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

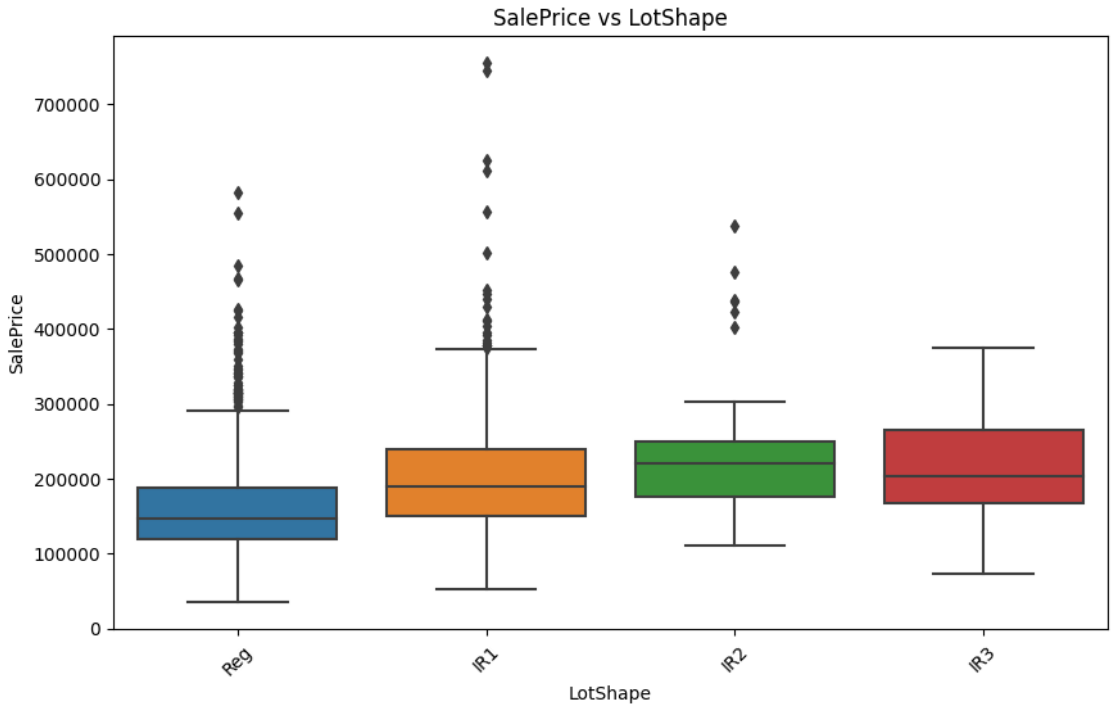


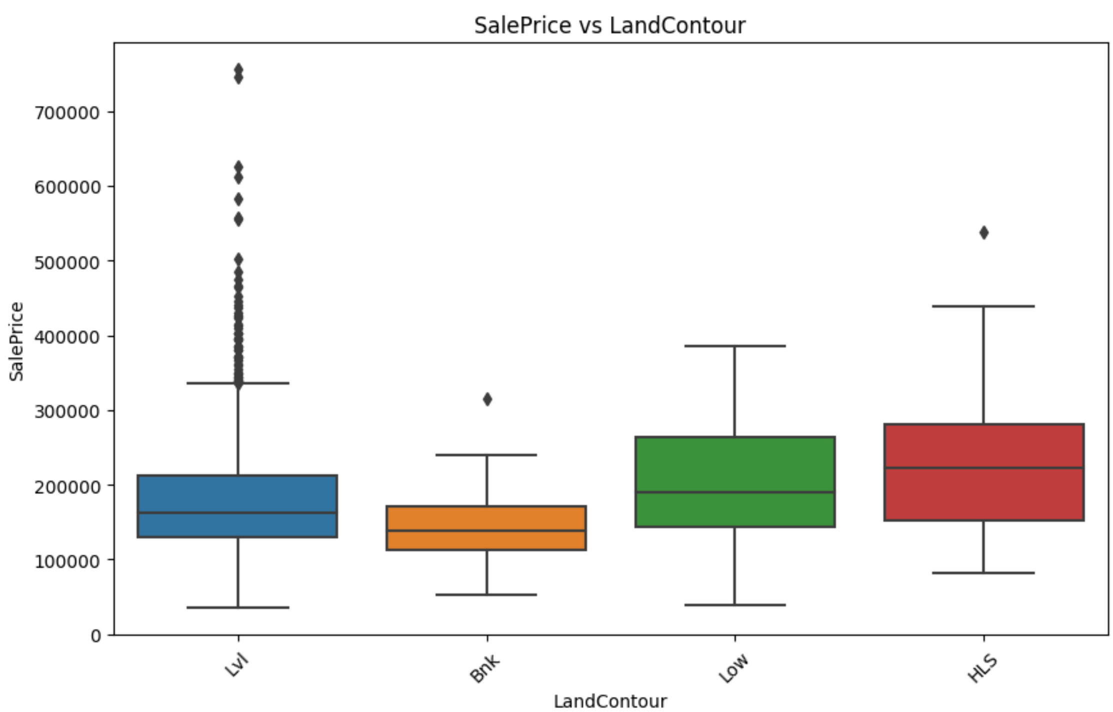
[ ]:
```
# Visualizing the relationship between categorical features and the target␣
↪variable
categorical_features = train_data.select_dtypes(include=['object']).columns
for feature in categorical_features:
    plt.figure(figsize=(10,6))
    sns.boxplot(x=feature, y='SalePrice', data=train_data)
    plt.title(f'SalePrice vs {feature}')
    plt.xticks(rotation=45)
    plt.show()
```

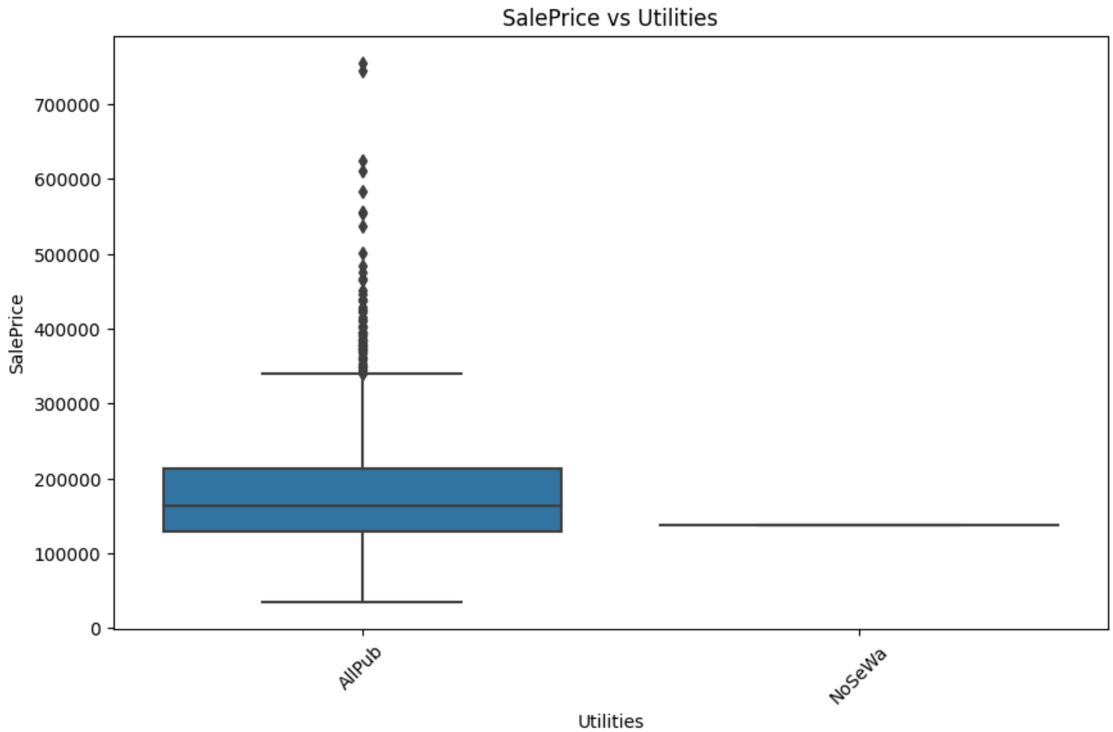


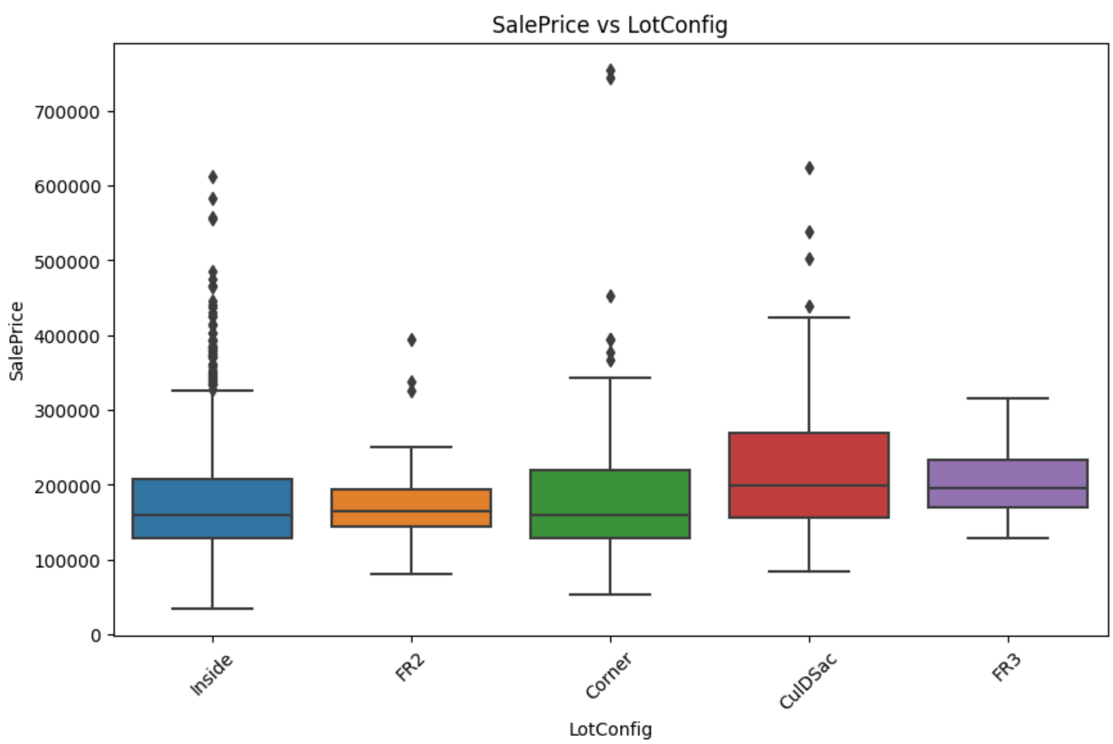

SalePrice vs Street
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Pave
Grvl
Street

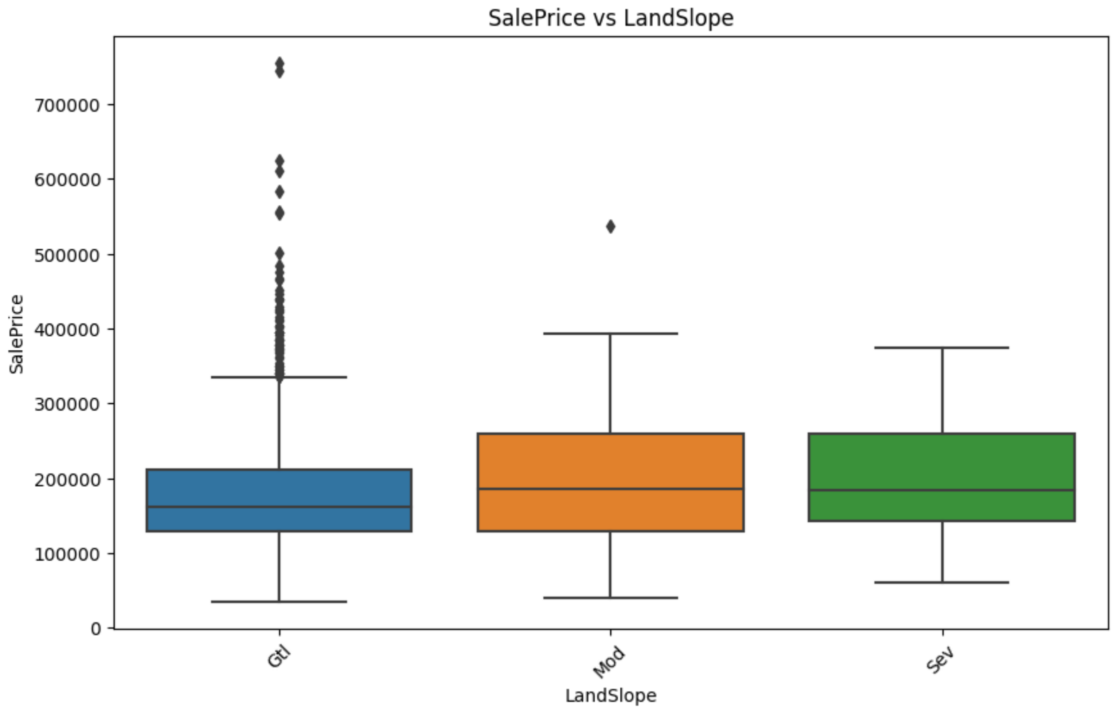

SalePrice vs Alley
SalePrice
250000
200000
150000
100000
50000
Grvl
Pave
Alley

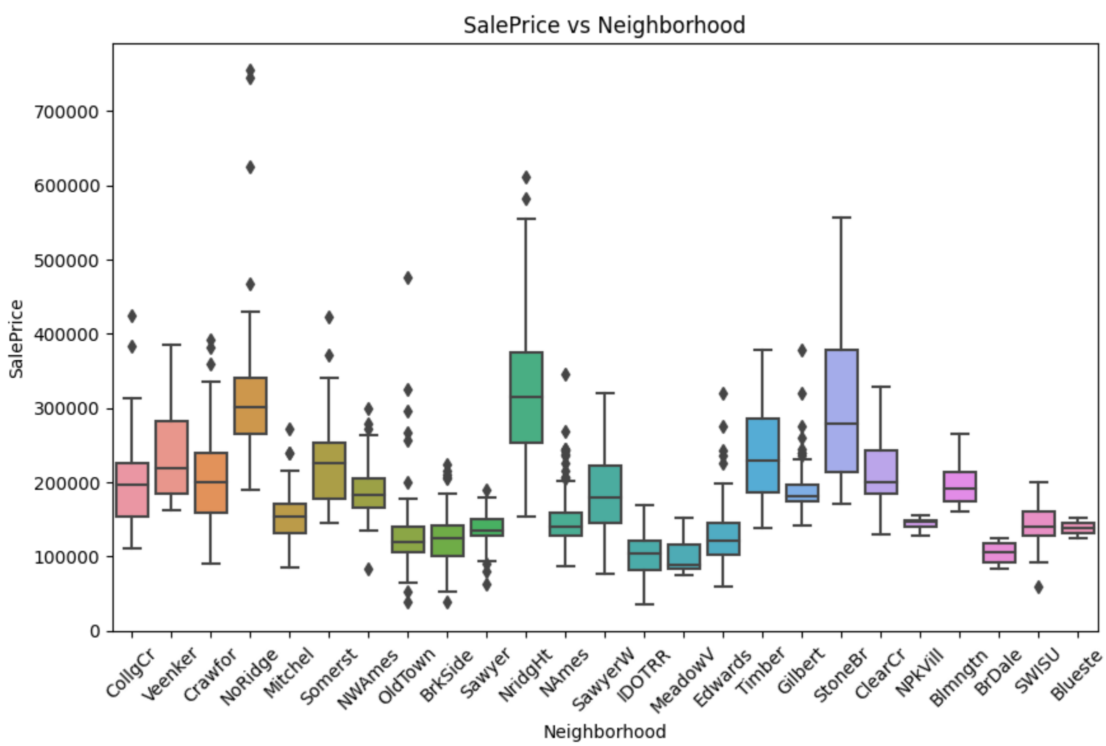

SalePrice vs LotShape
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Reg
IR1
IR2
IR3
LotShape

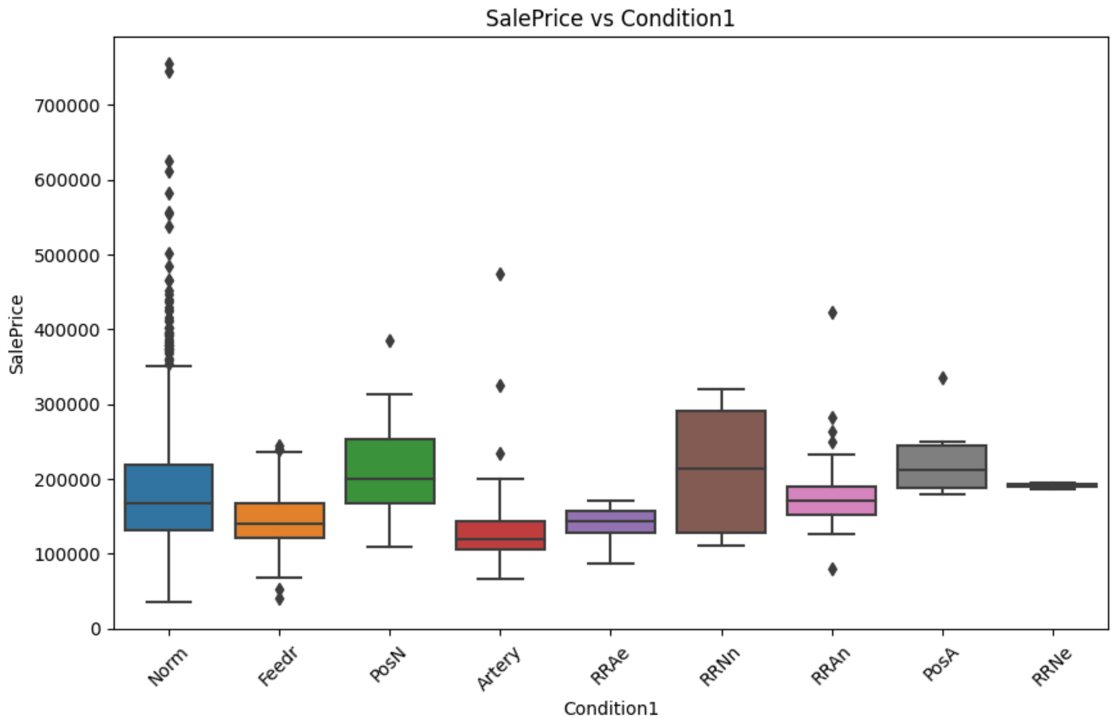

SalePrice vs LandContour
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Lvl
Bnk
Low
HLS
LandContour

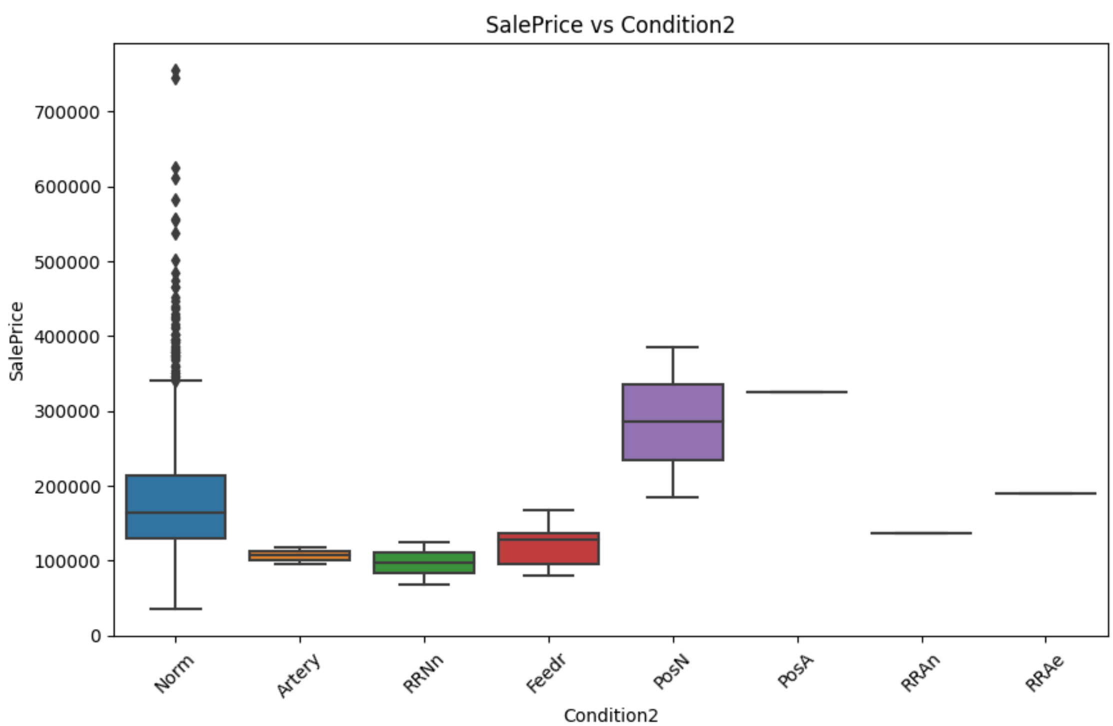

SalePrice vs Utilities
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
AllPub
NoSeWa
Utilities

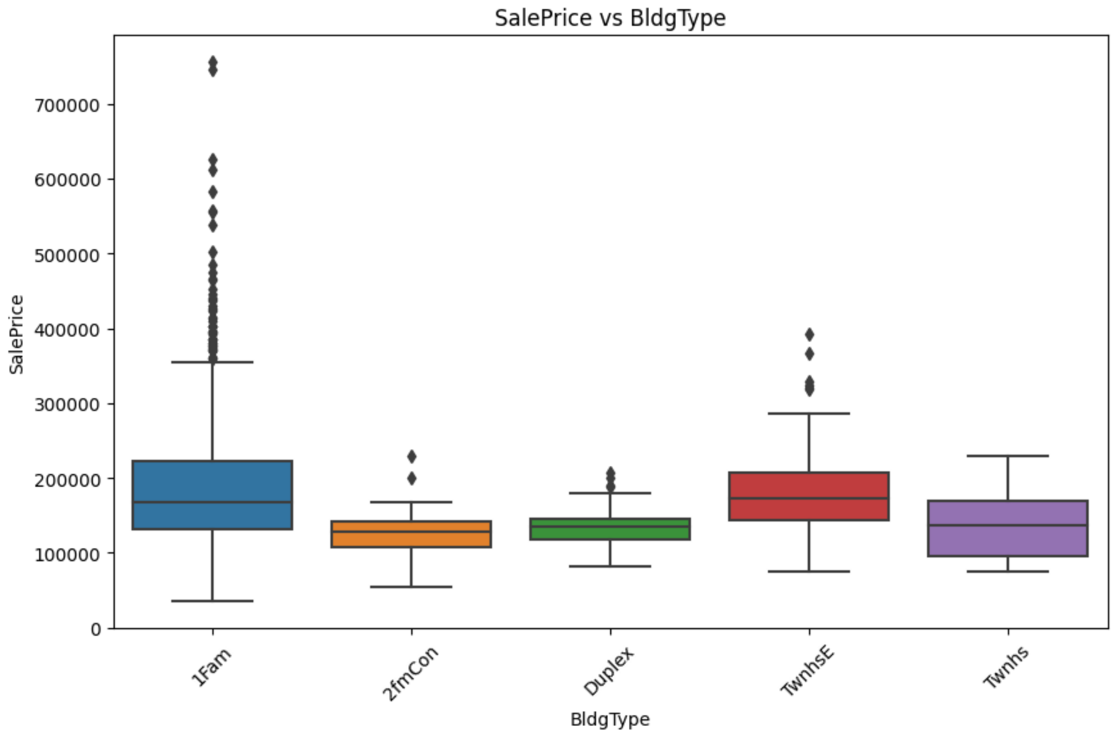

SalePrice vs LotConfig
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Inside
FR2
Corner
CulDSac
FR3
LotConfig

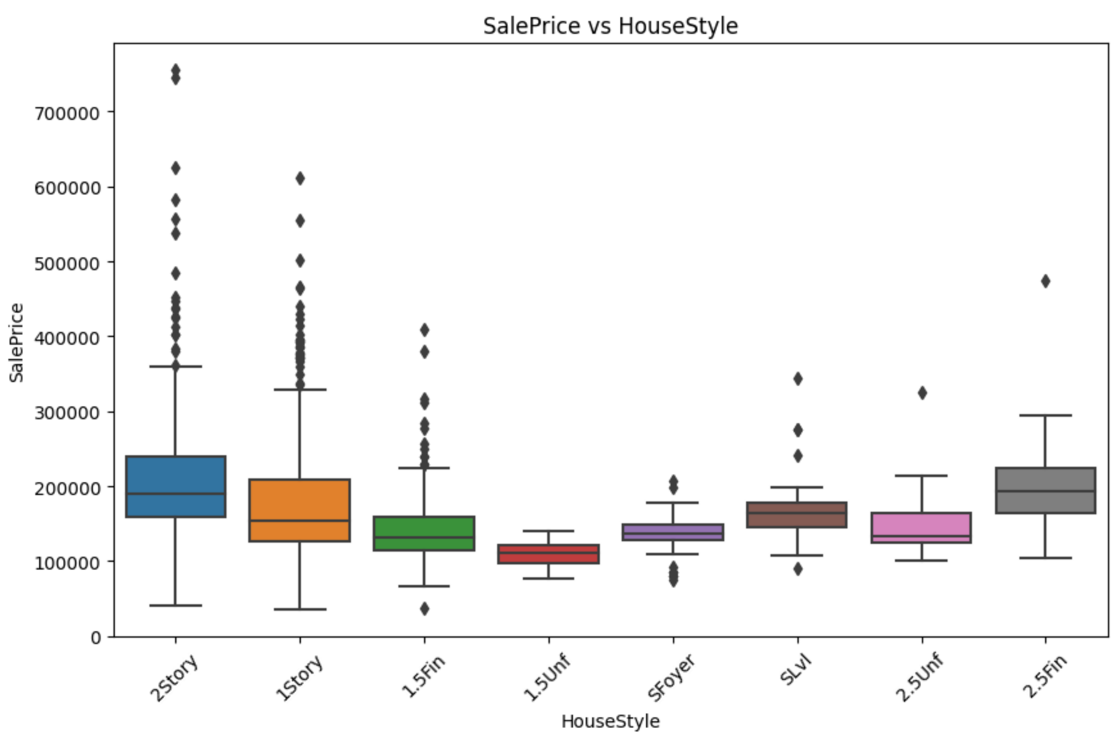

SalePrice vs LandSlope
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gtl
Mod
Sev
LandSlope

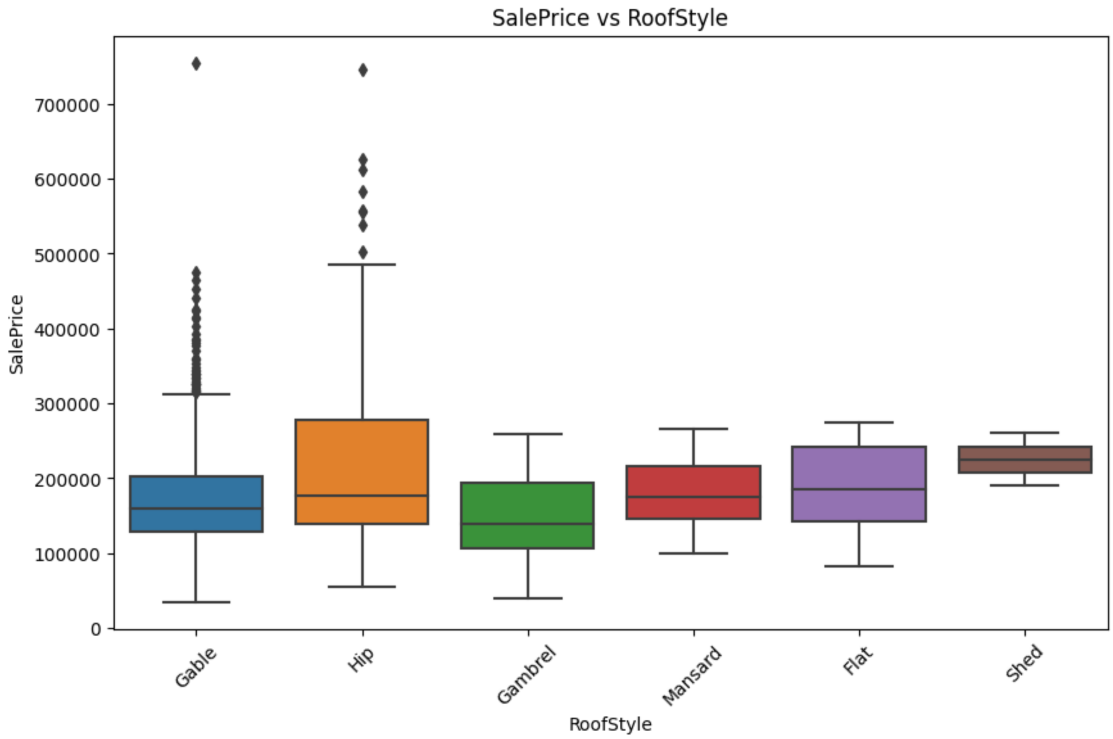

SalePrice vs Neighborhood
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
CollgCr
Veenker
Crawfor
NoRidge
Mitchel
Somerst
NWAmes
OldTown
BrkSide
Sawyer
NridgHt
NAmes
SawyerW
IDOTRR
MeadowV
Edwards
Timber
Gilbert
StoneBr
ClearCr
NPkVill
Blmngtn
BrDale
SWISU
Blueste
Neighborhood

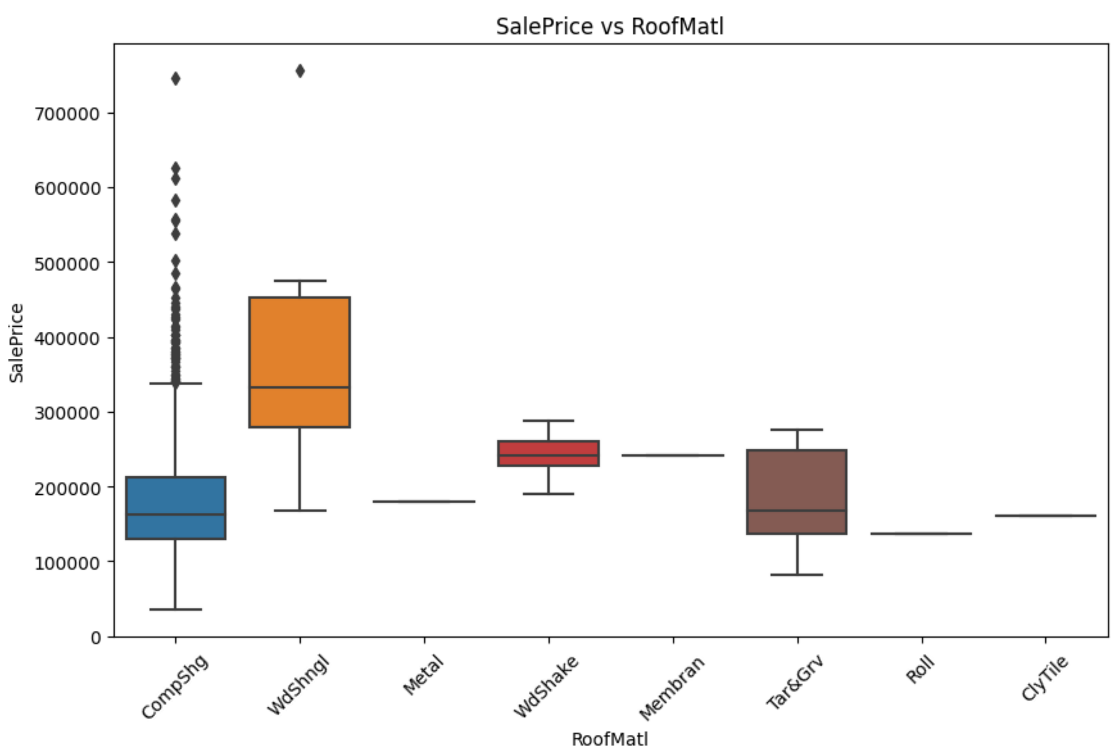

SalePrice vs Condition1
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Norm
Feedr
PosN
Artery
RRAe
RRNn
RRAn
PosA
RRNe
Condition1

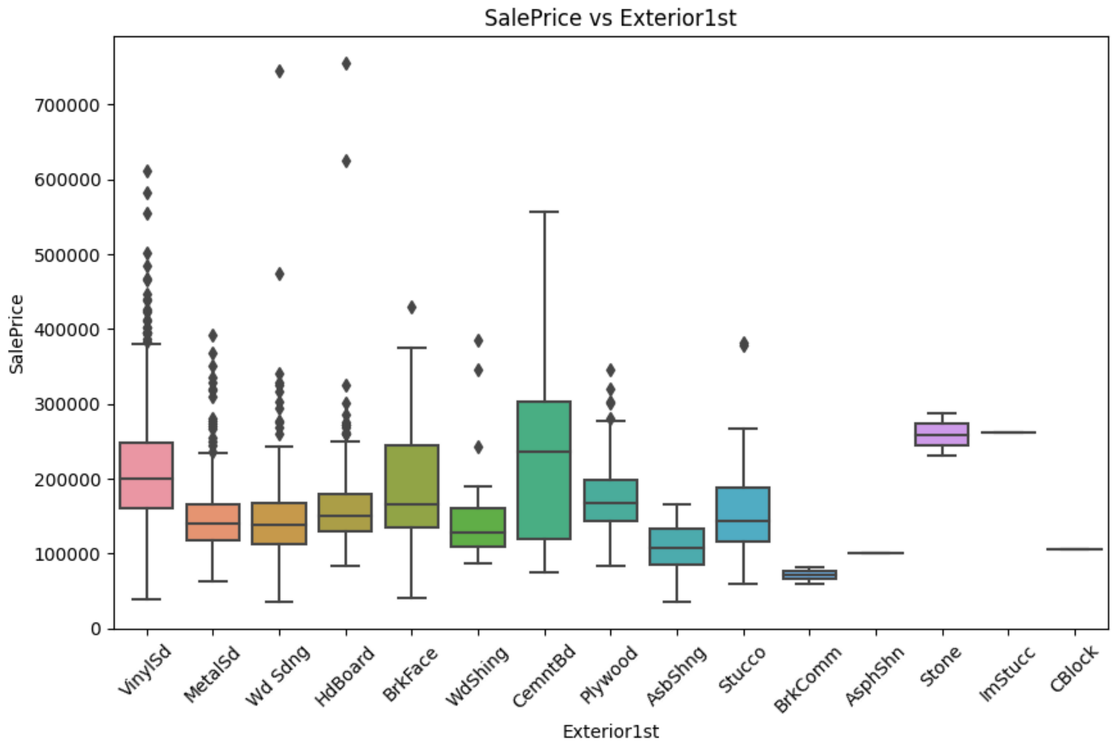

SalePrice vs Condition2
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Norm
Artery
RRNn
Feedr
PosN
PosA
RRAn
RRAe
Condition2

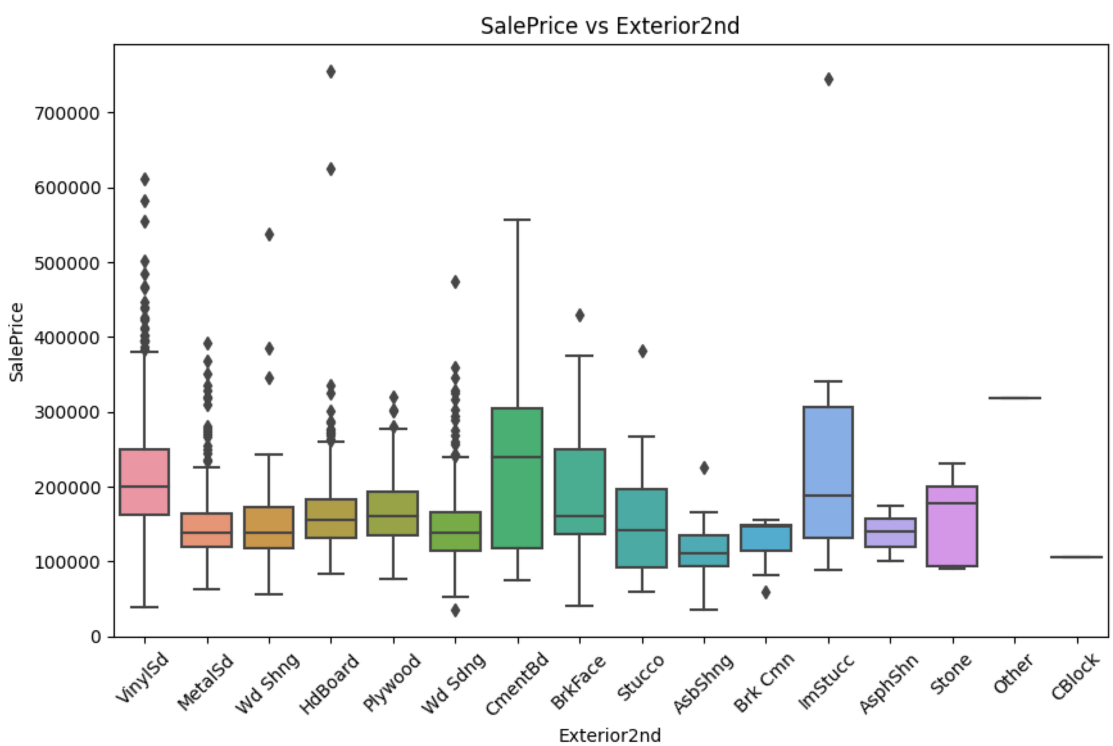

SalePrice vs BldgType
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1Fam
2fmCon
Duplex
TwnhsE
Twnhs
BldgType

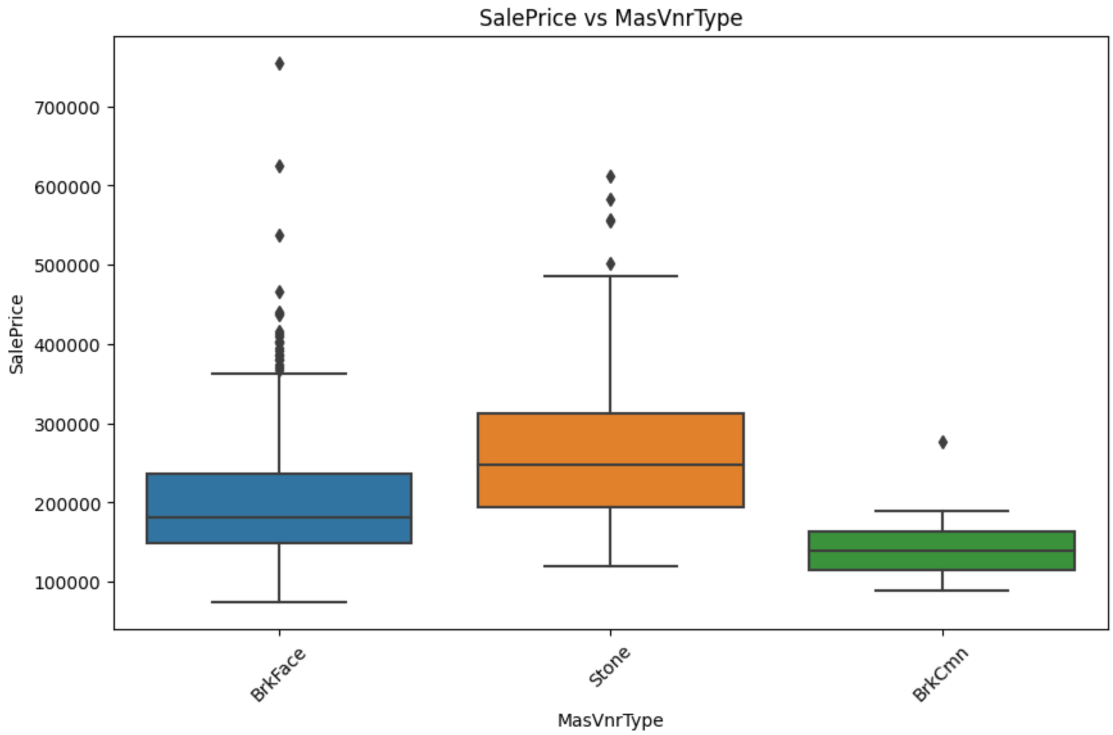

SalePrice vs HouseStyle
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
2Story
1Story
1.5Fin
1.5Unf
SFoyer
SLvl
2.5Unf
2.5Fin
HouseStyle

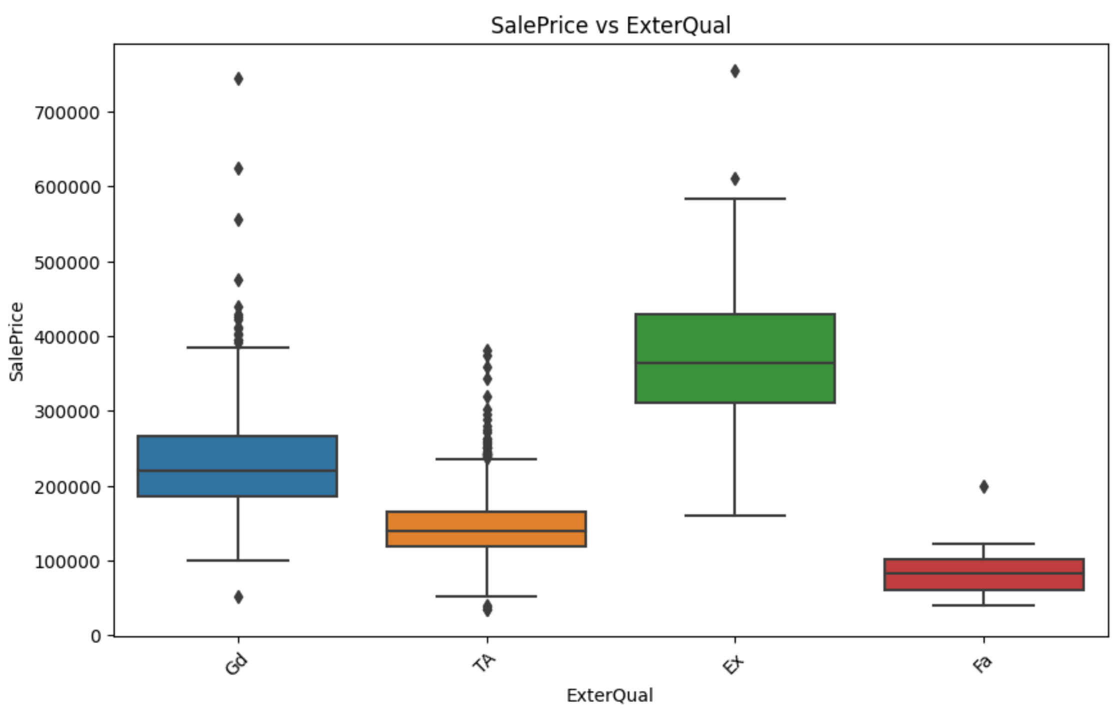

SalePrice vs RoofStyle
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gable
Hip
Gambrel
Mansard
Flat
Shed
RoofStyle

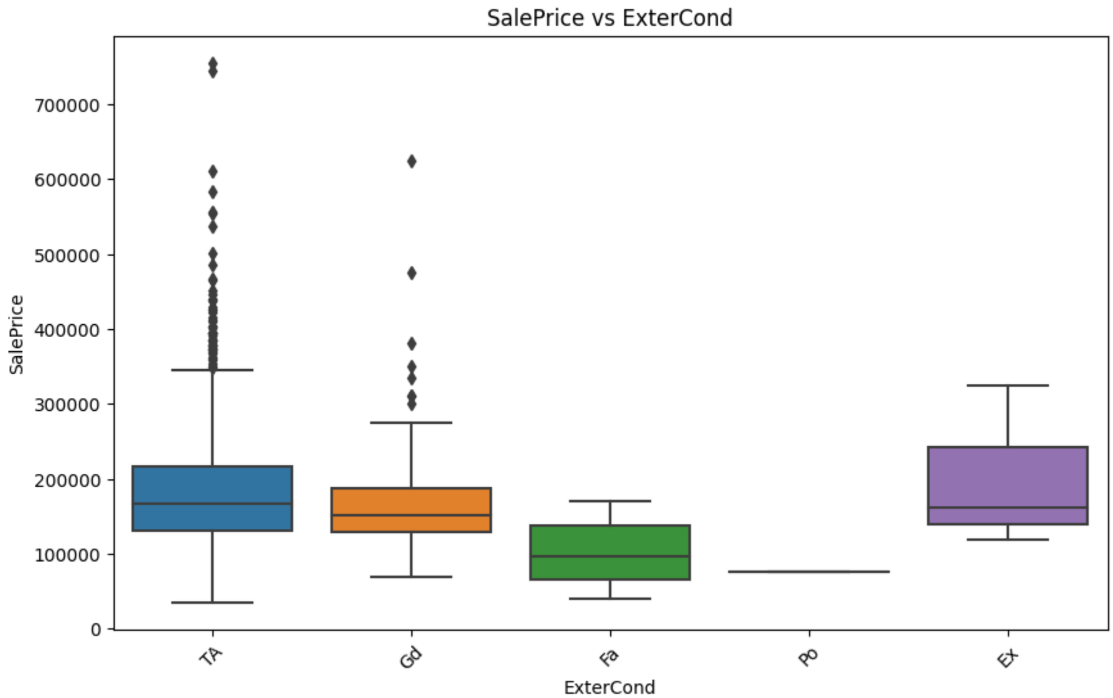

SalePrice vs RoofMatl
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
CompShg
WdShngl
Metal
WdShake
Membran
Tar&Grv
Roll
ClyTile
RoofMatl

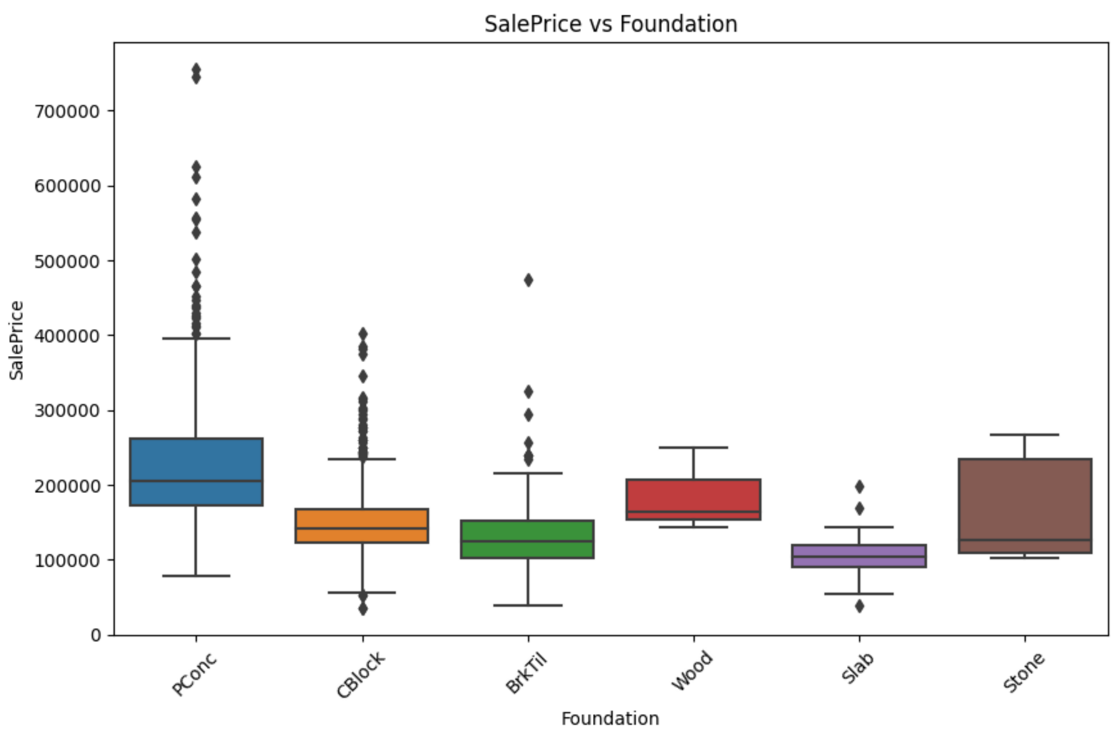

SalePrice vs Exterior1st
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
VinylSd
MetalSd
Wd Sdng
HdBoard
BrkFace
WdShing
CemntBd
Plywood
AsbShng
Stucco
BrkComm
AsphShn
Stone
ImStucc
CBlock
Exterior1st

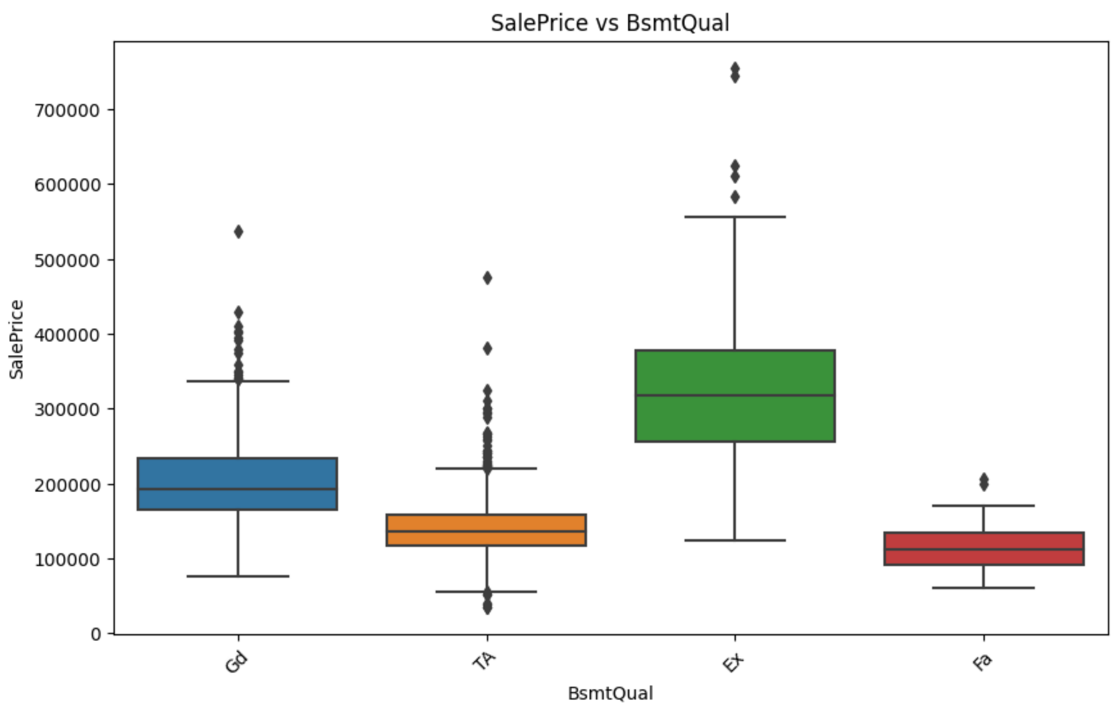

SalePrice vs Exterior2nd
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
VinylSd
MetalSd
Wd Shng
HdBoard
Plywood
Wd Sdng
CmentBd
BrkFace
Stucco
AsbShng
Brk Cmn
ImStucc
AsphShn
Stone
Other
CBlock
Exterior2nd

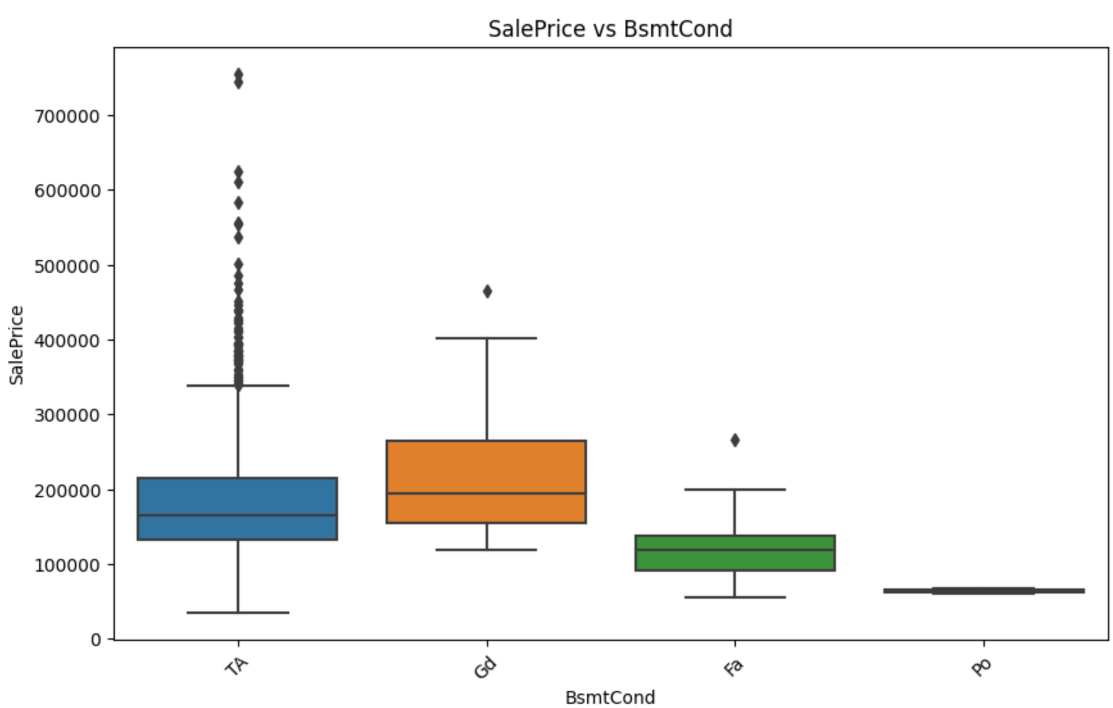

SalePrice vs MasVnrType
SalePrice
700000
600000
500000
400000
300000
200000
100000
BrkFace
Stone
BrkCmn
MasVnrType

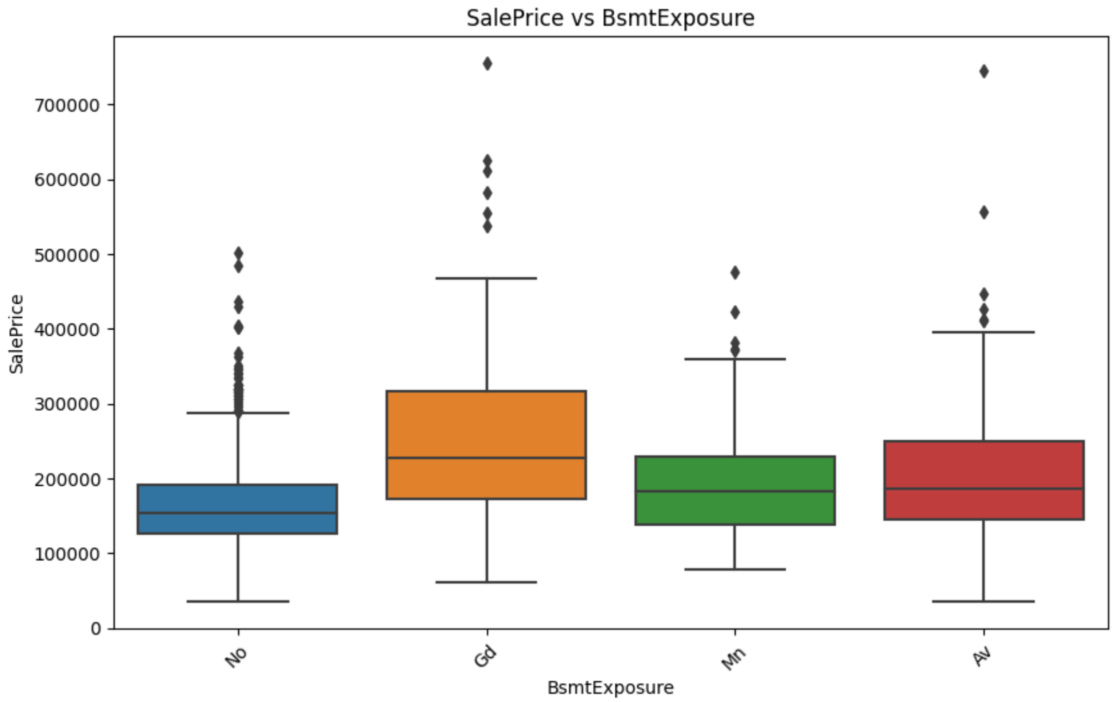

SalePrice vs ExterQual
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
Fa
ExterQual

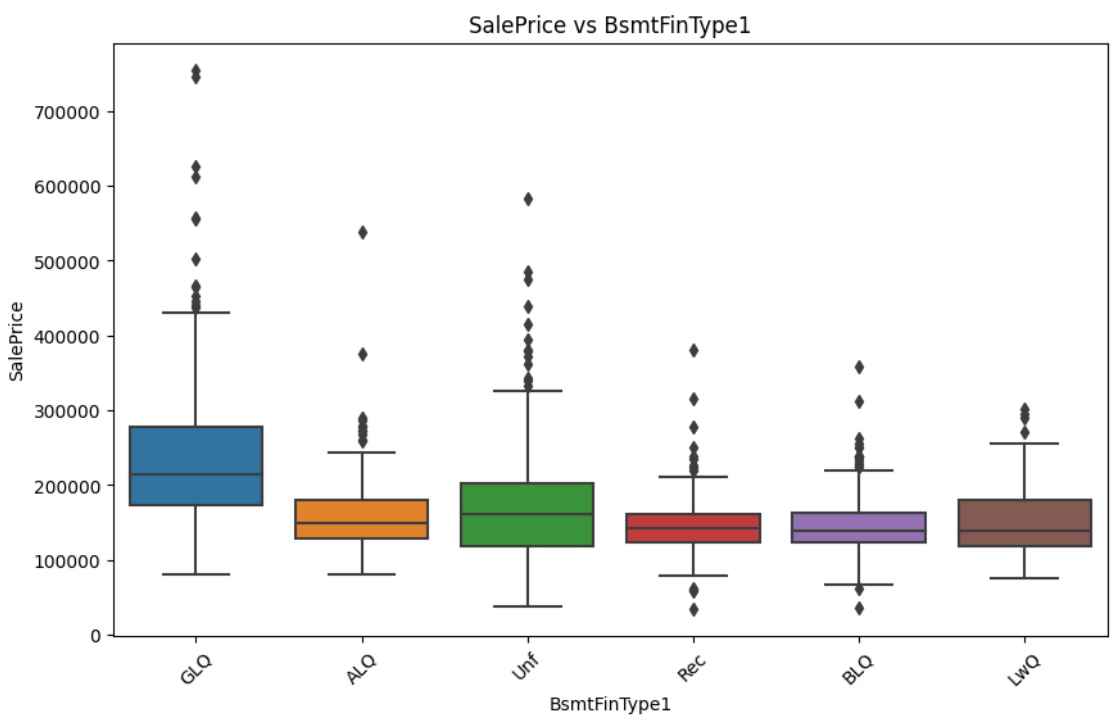

SalePrice vs ExterCond
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
Po
Ex
ExterCond

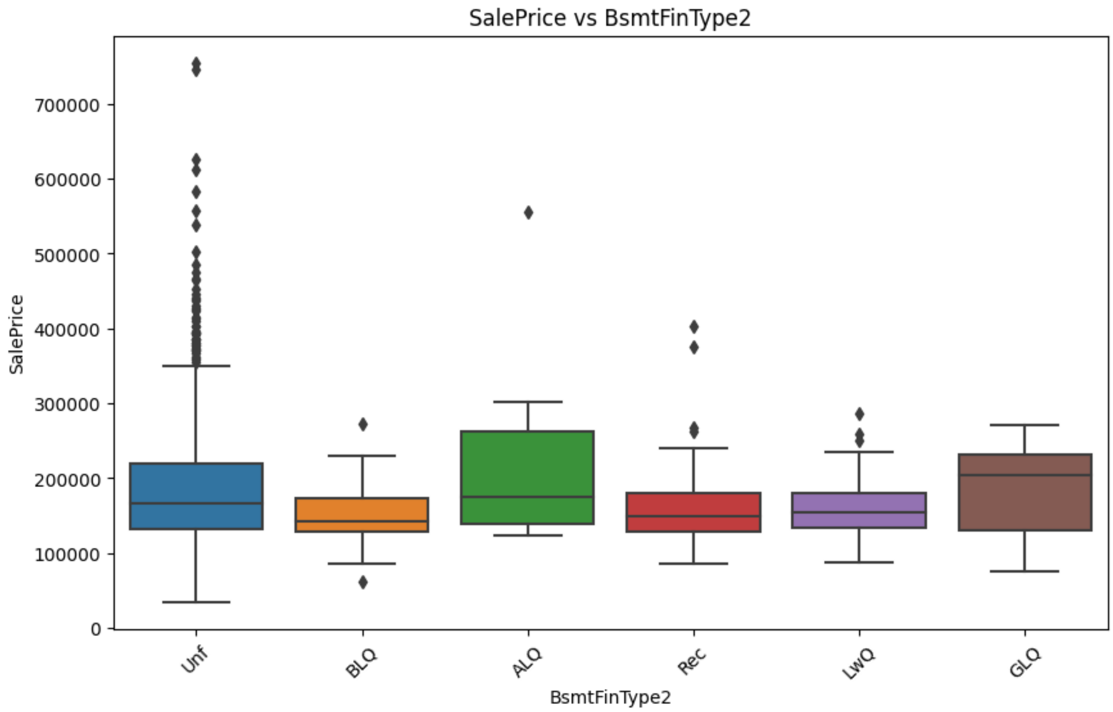

SalePrice vs Foundation
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
PConc
CBlock
BrkTil
Wood
Slab
Stone
Foundation

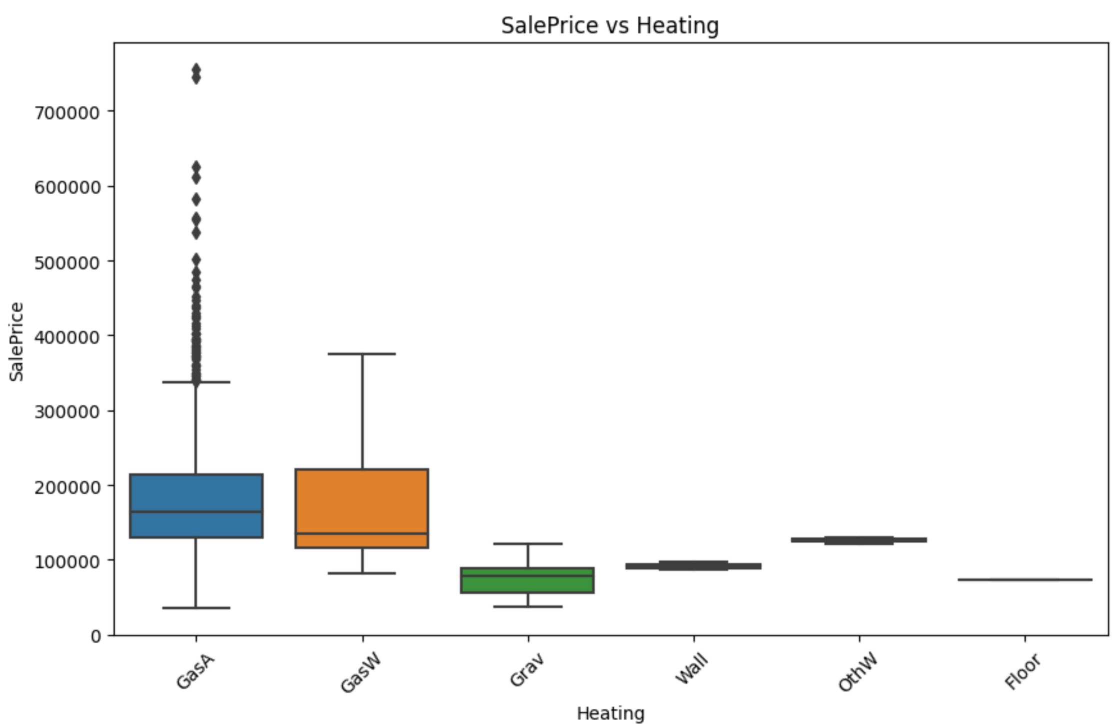

SalePrice vs BsmtQual
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
Fa
BsmtQual

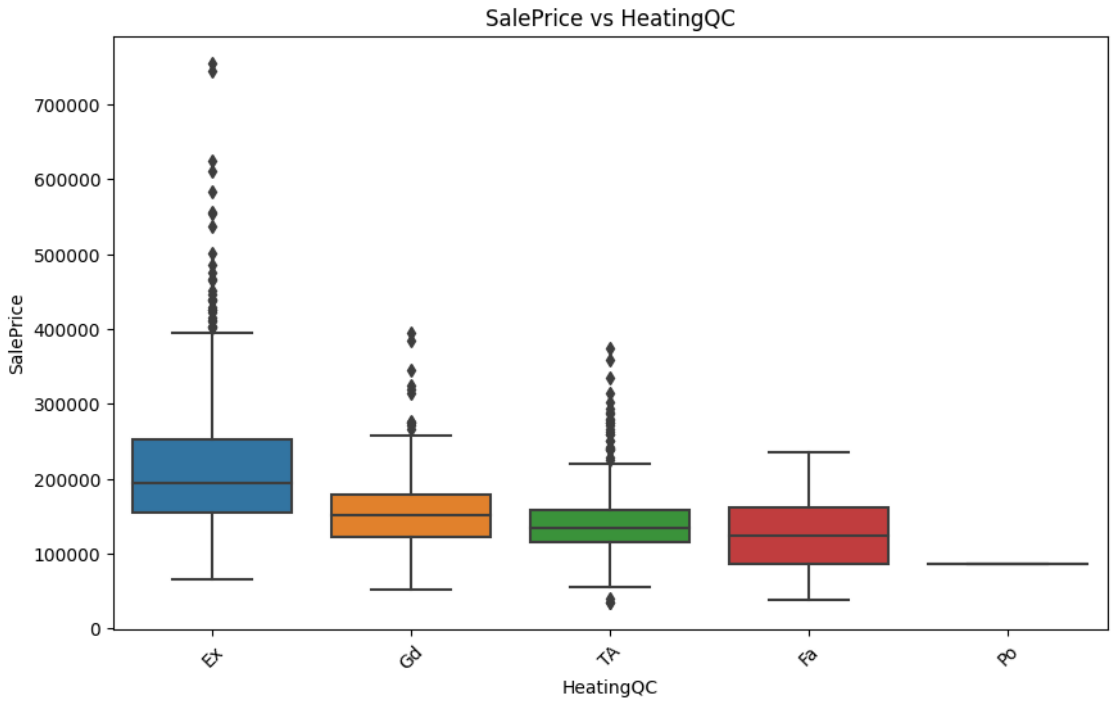

SalePrice vs BsmtCond
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
Po
BsmtCond


SalePrice vs BsmtExposure
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
No
Gd
Mn
Av
BsmtExposure

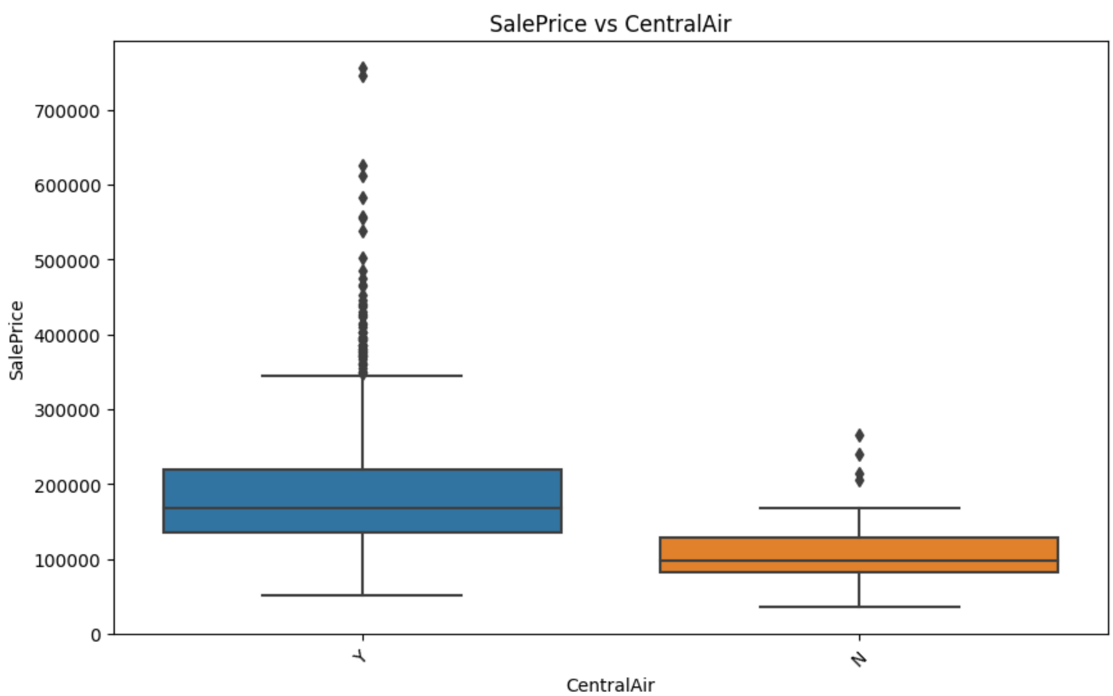

SalePrice vs BsmtFinType1
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
GLQ
ALQ
Unf
Rec
BLQ
LwQ
BsmtFinType1


SalePrice vs BsmtFinType2
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Unf
BLQ
ALQ
Rec
LwQ
GLQ
BsmtFinType2

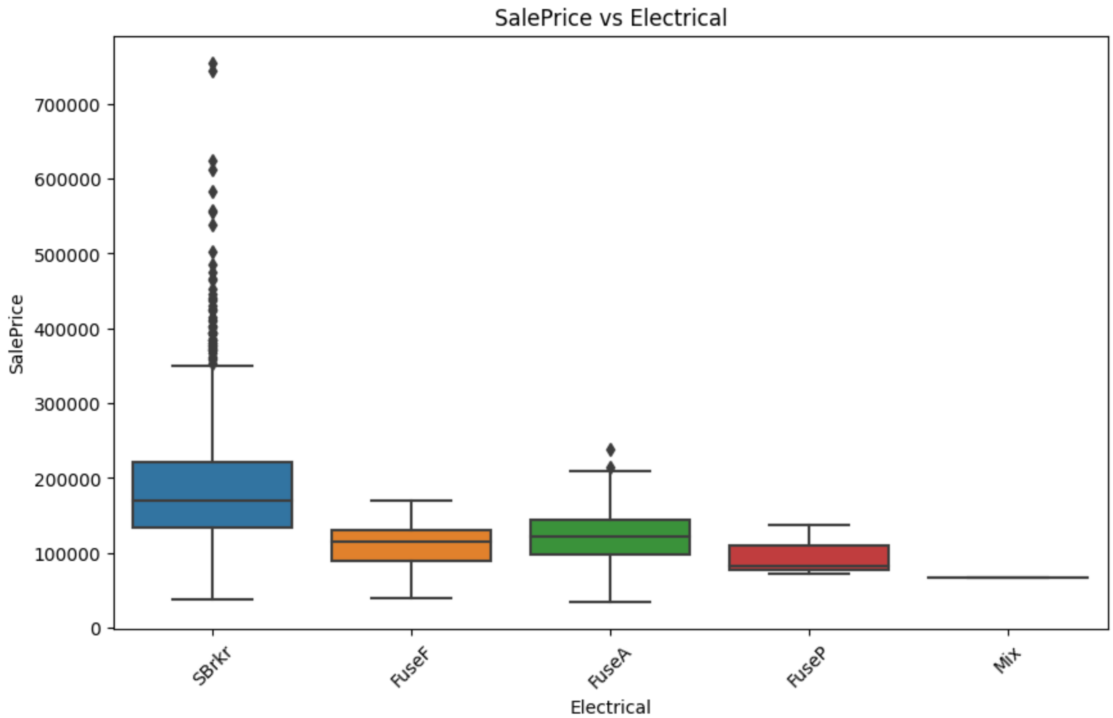

SalePrice vs Heating
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
GasA
GasW
Grav
Wall
OthW
Floor
Heating

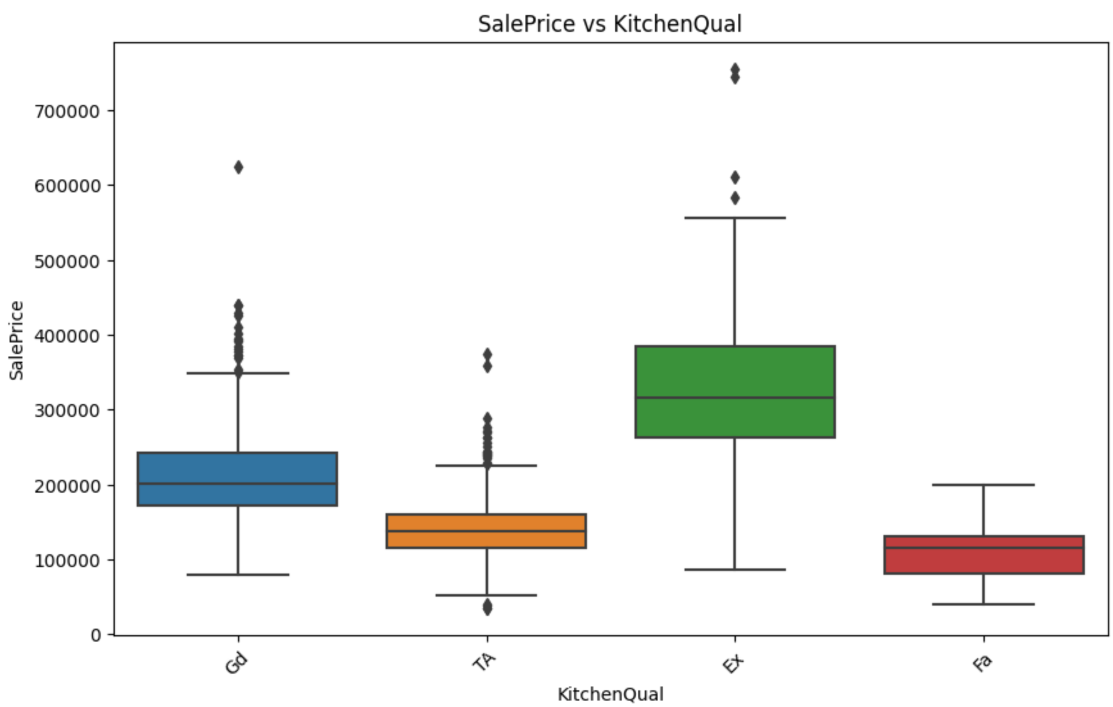


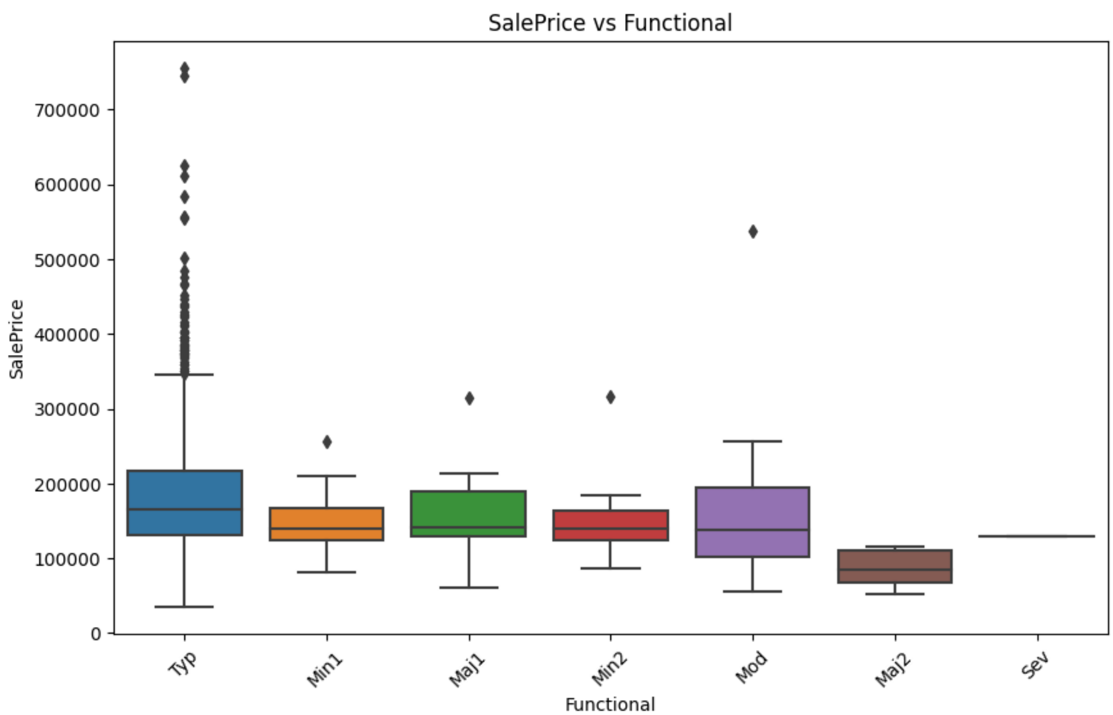
SalePrice vs HeatingQC
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Ex
Gd
TA
Fa
Po
HeatingQC

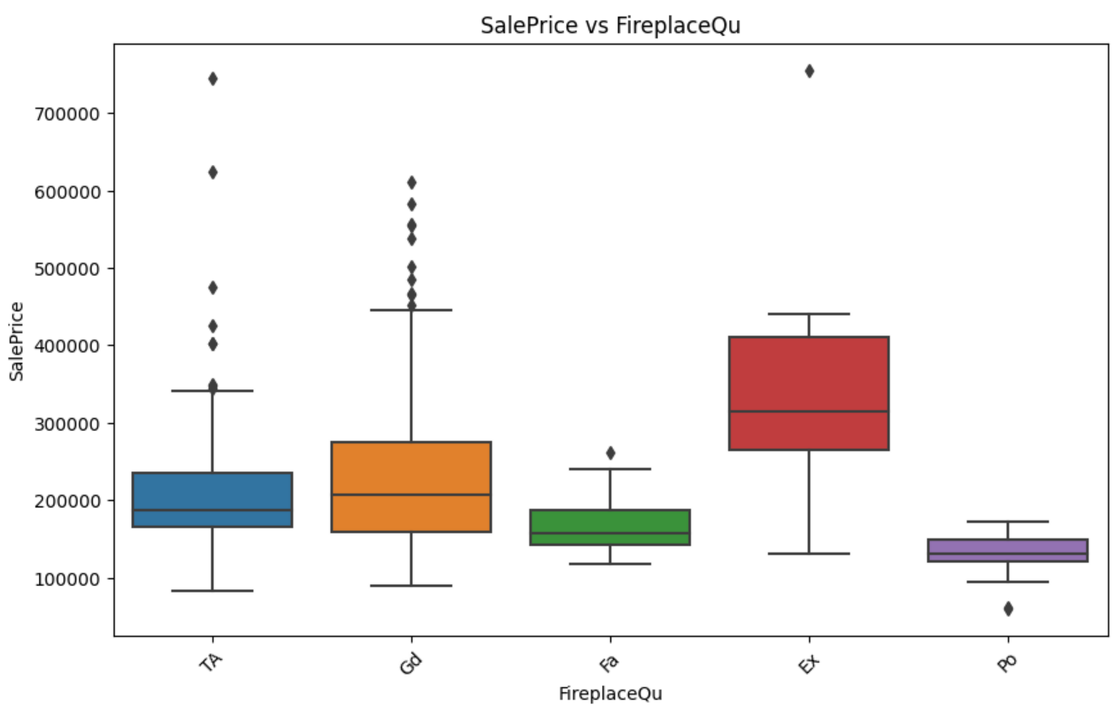


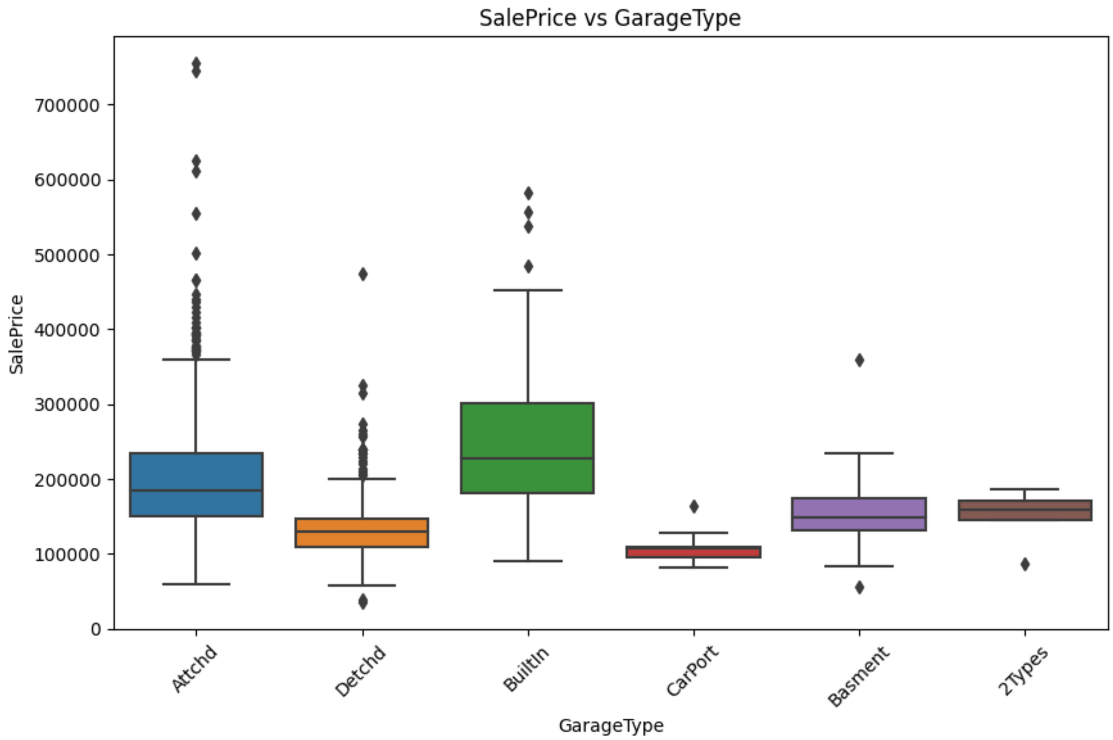
SalePrice vs CentralAir
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Y
N
CentralAir

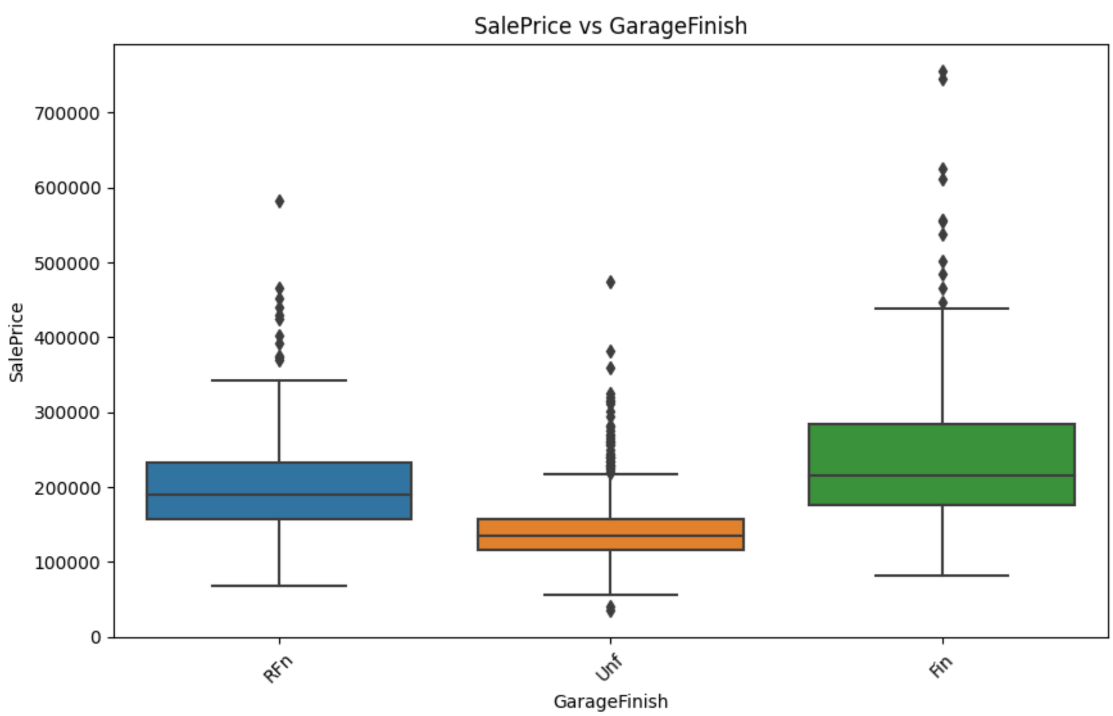

SalePrice vs Electrical
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
SBrkr
FuseF
FuseA
FuseP
Mix
Electrical

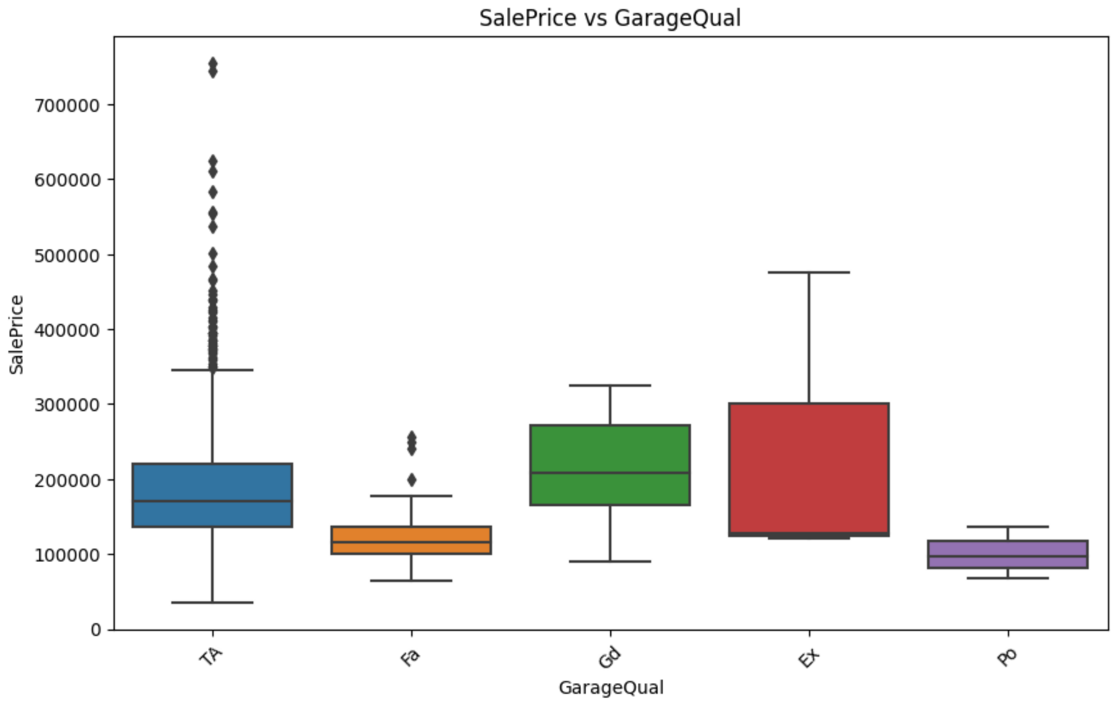

SalePrice vs KitchenQual
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
Fa
KitchenQual

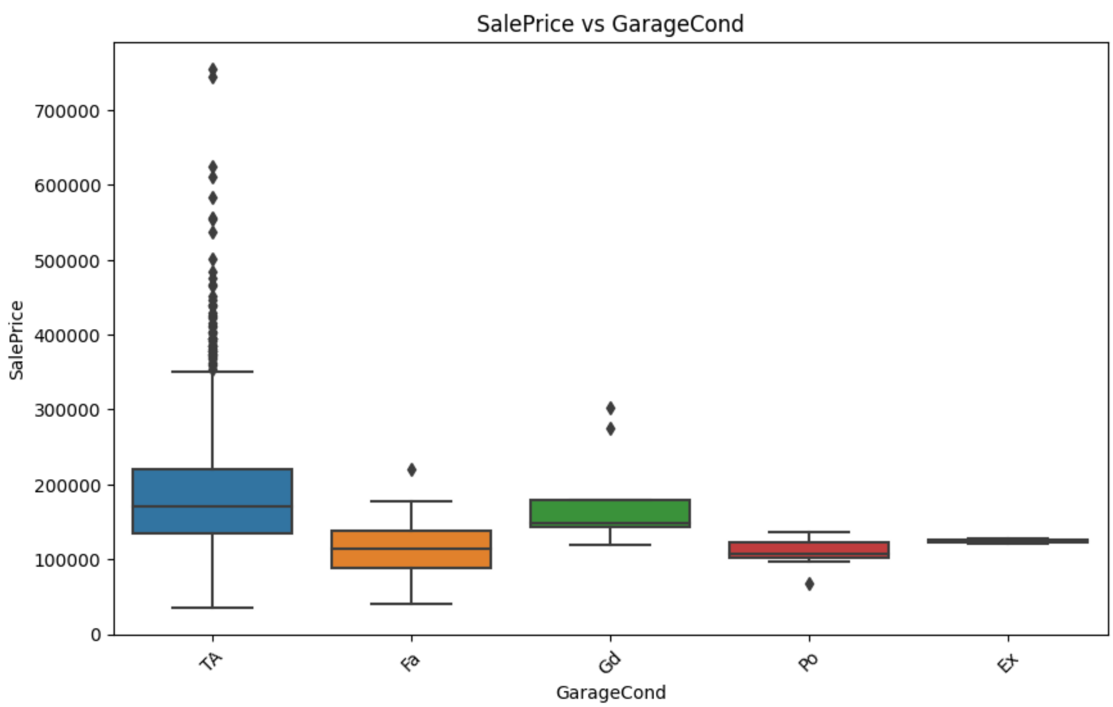

SalePrice vs Functional
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Typ
Min1
Maj1
Min2
Mod
Maj2
Sev
Functional

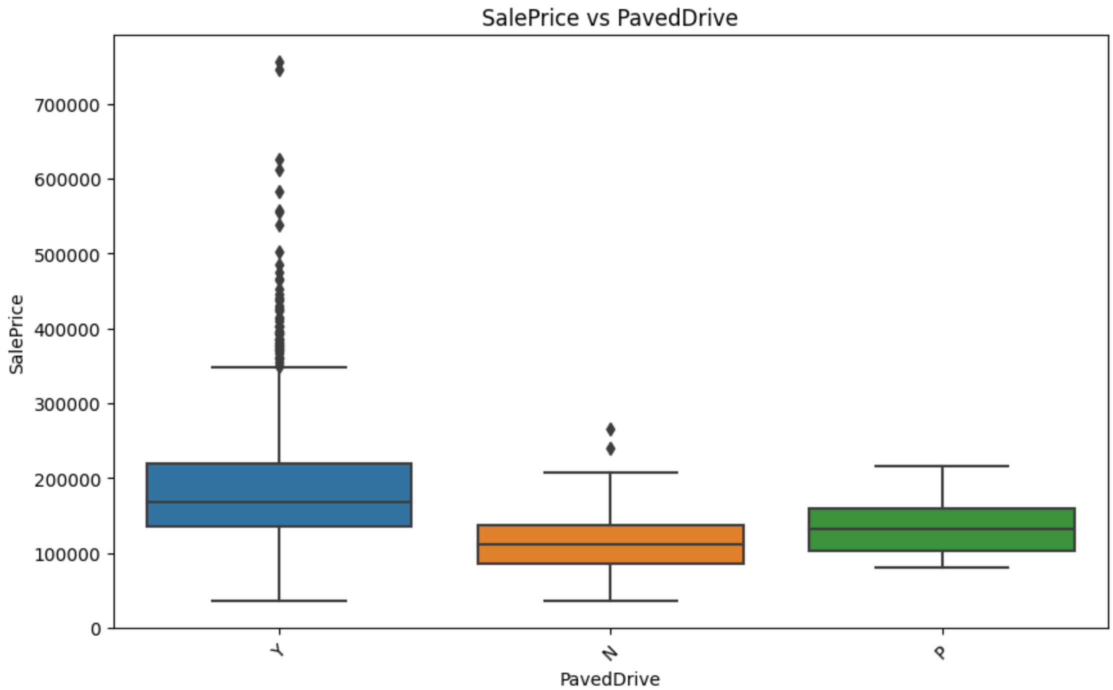

SalePrice vs FireplaceQu
SalePrice
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
Ex
Po
FireplaceQu

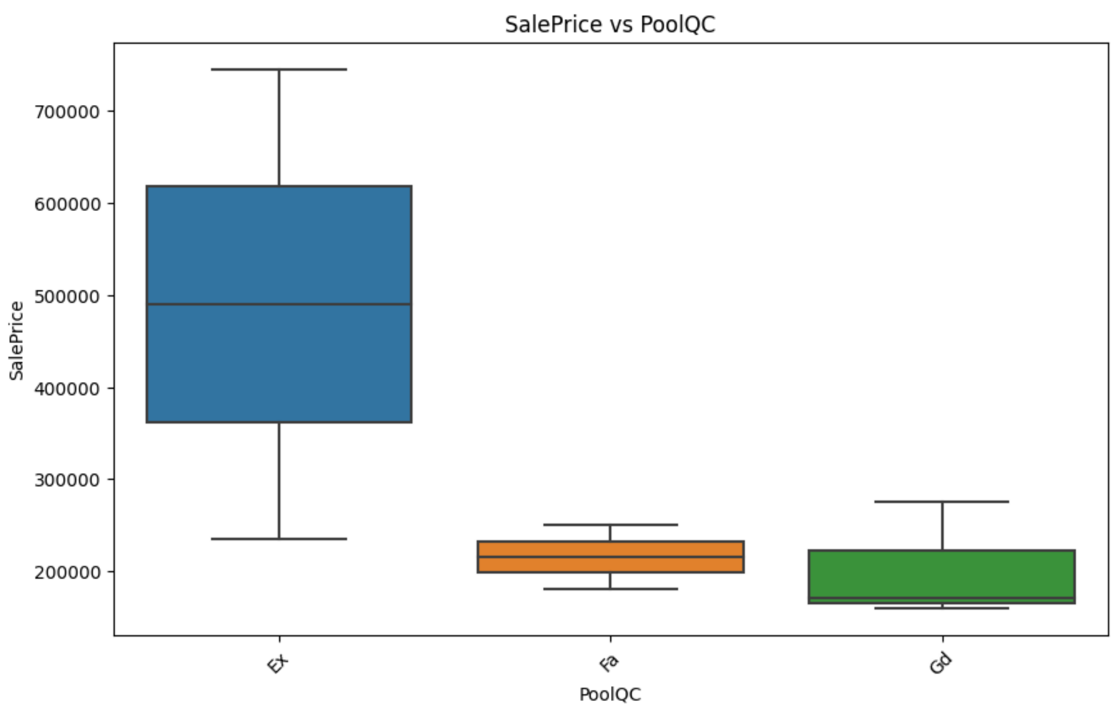

SalePrice vs GarageType
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Attchd
Detchd
BuiltIn
CarPort
Basment
2Types
GarageType

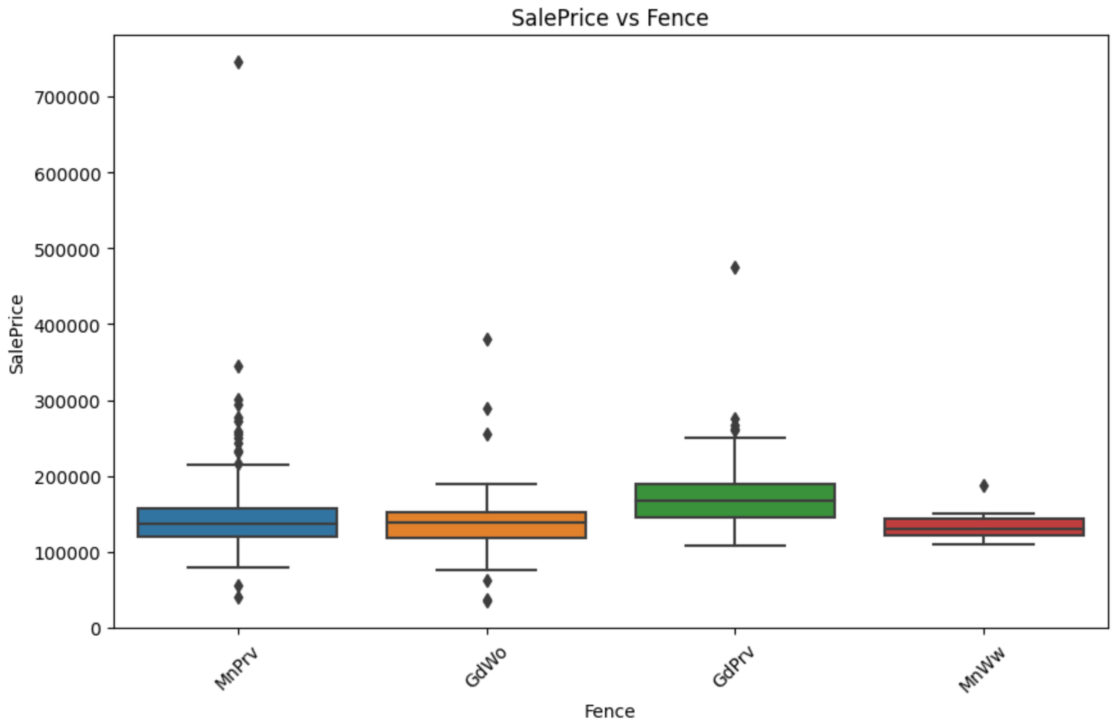

SalePrice vs GarageFinish
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
RFn
Unf
Fin
GarageFinish

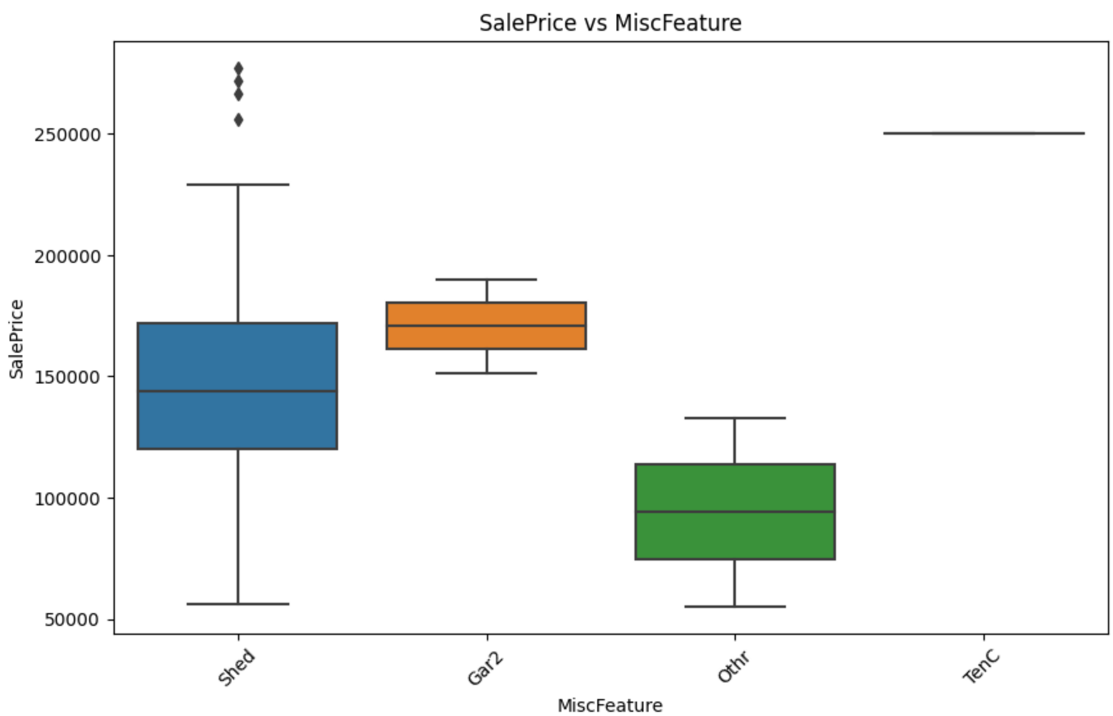

SalePrice vs GarageQual
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Fa
Gd
Ex
Po
GarageQual

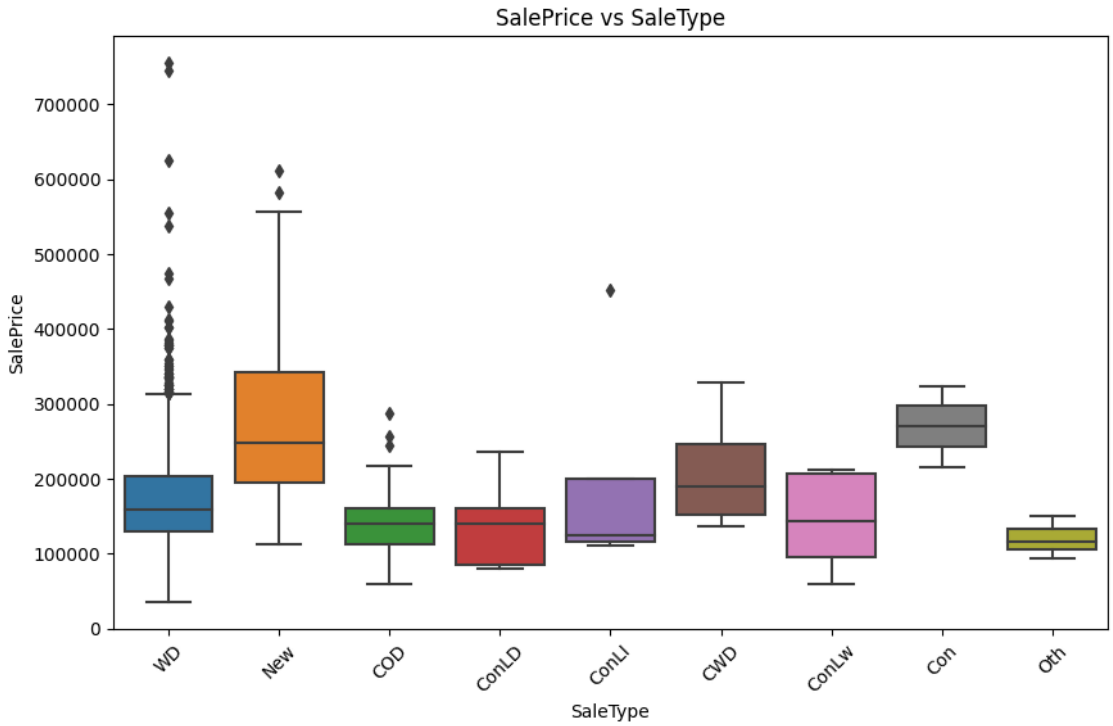

SalePrice vs GarageCond
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Fa
Gd
Po
Ex
GarageCond

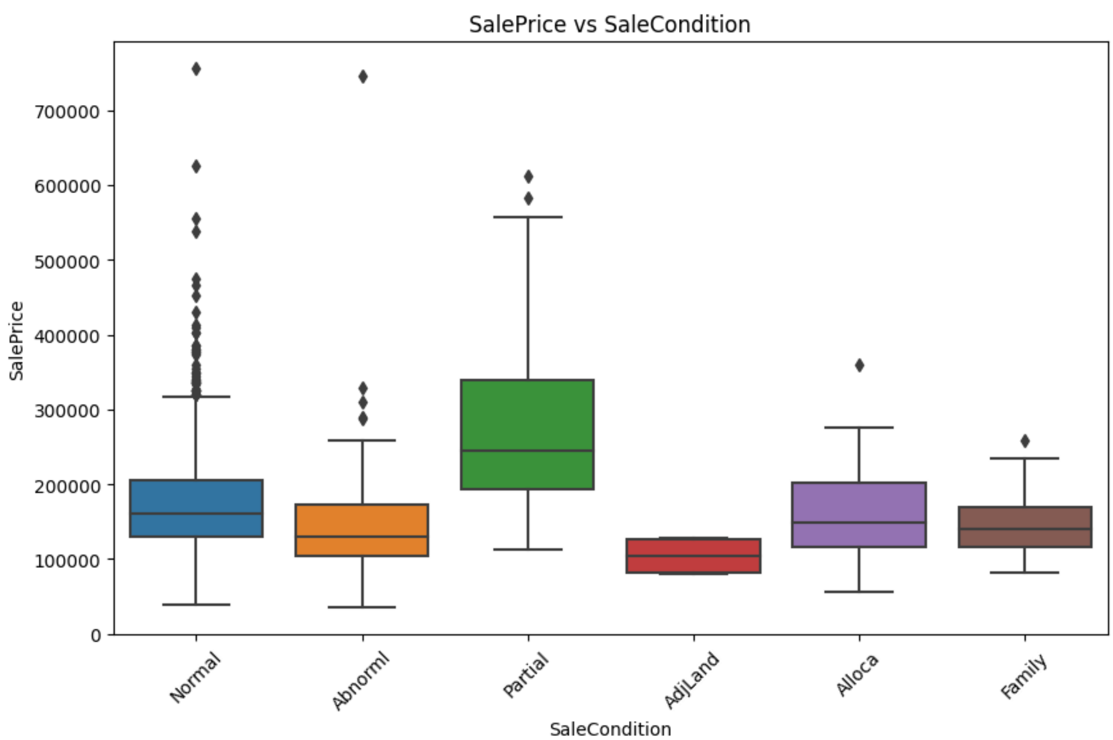

SalePrice vs PavedDrive
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Y
N
P
PavedDrive


SalePrice vs PoolQC
SalePrice
700000
600000
500000
400000
300000
200000
Ex
Fa
Gd
PoolQC

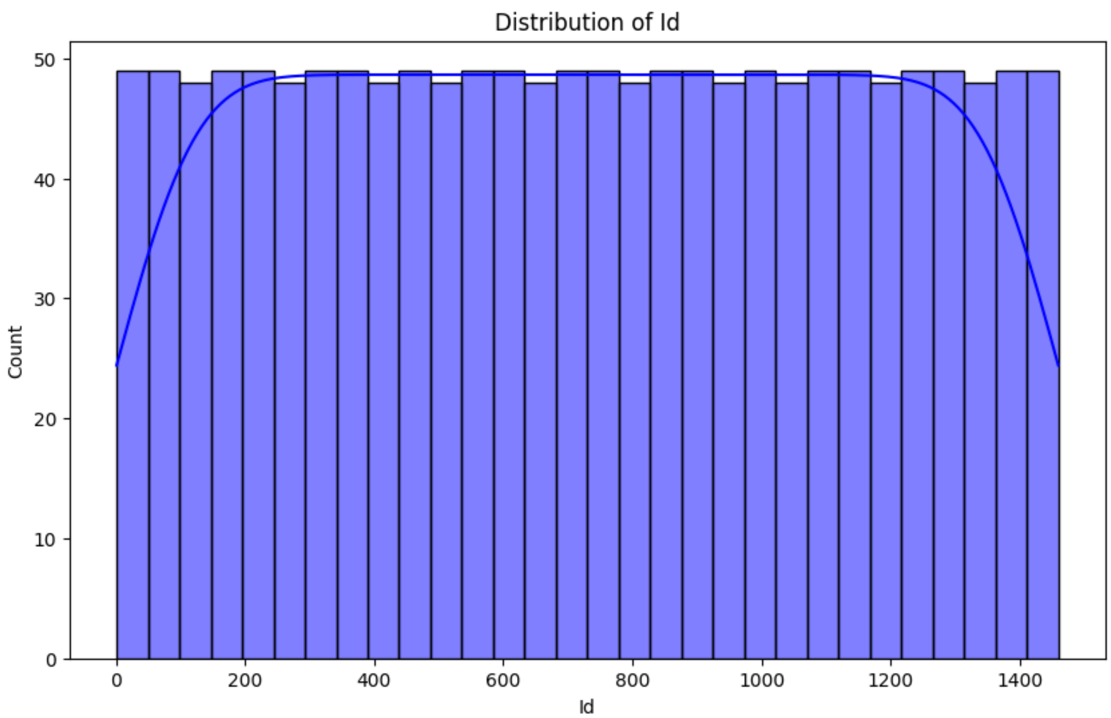

SalePrice vs Fence
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
MnPrv
GdWo
GdPrv
MnWw
Fence

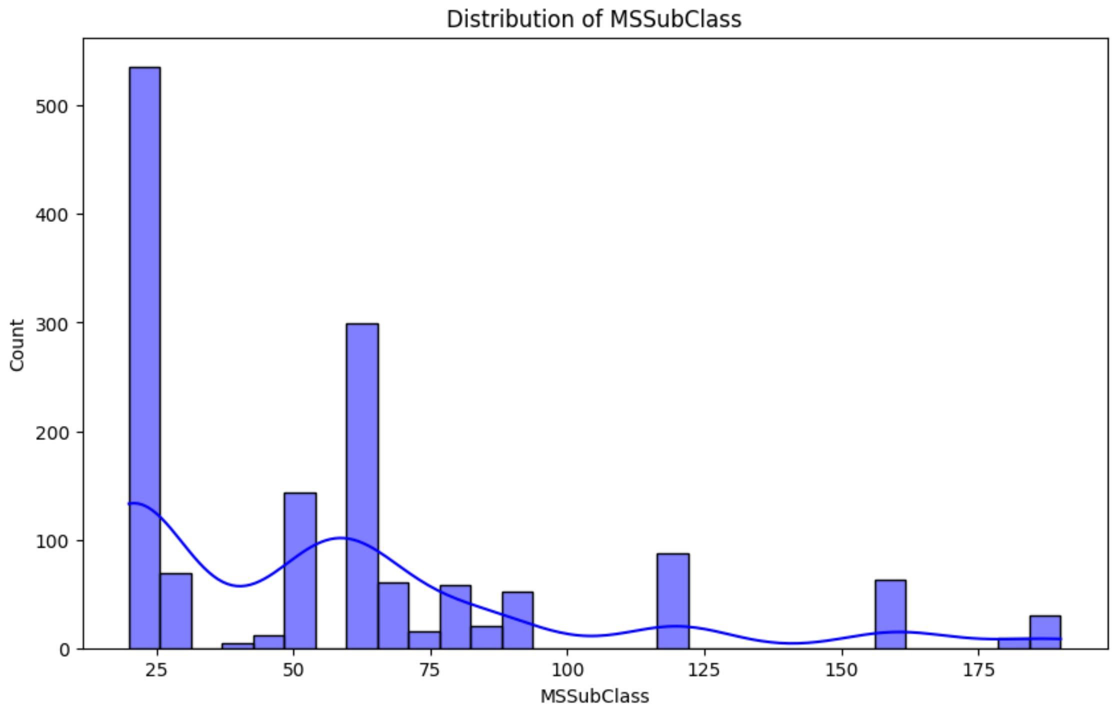

SalePrice vs MiscFeature
SalePrice
50000
100000
150000
200000
250000
Shed
Gar2
Othr
TenC
MiscFeature

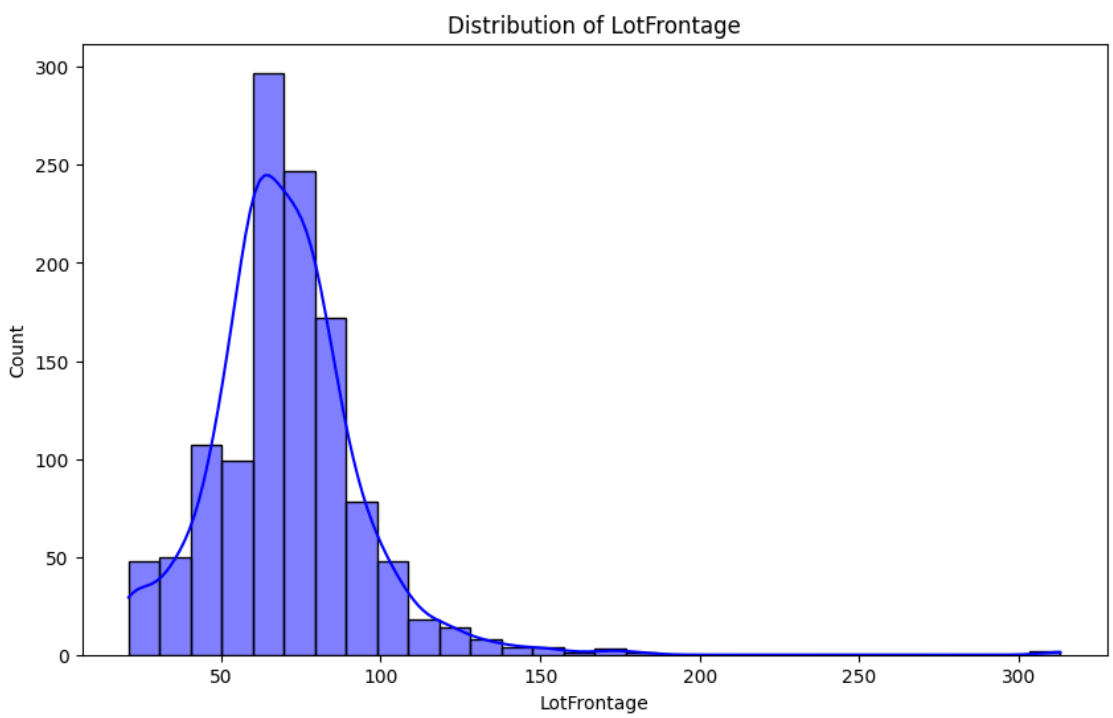

SalePrice vs SaleType
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
WD
New
COD
ConLD
ConLI
CWD
ConLw
Con
Oth
SaleType

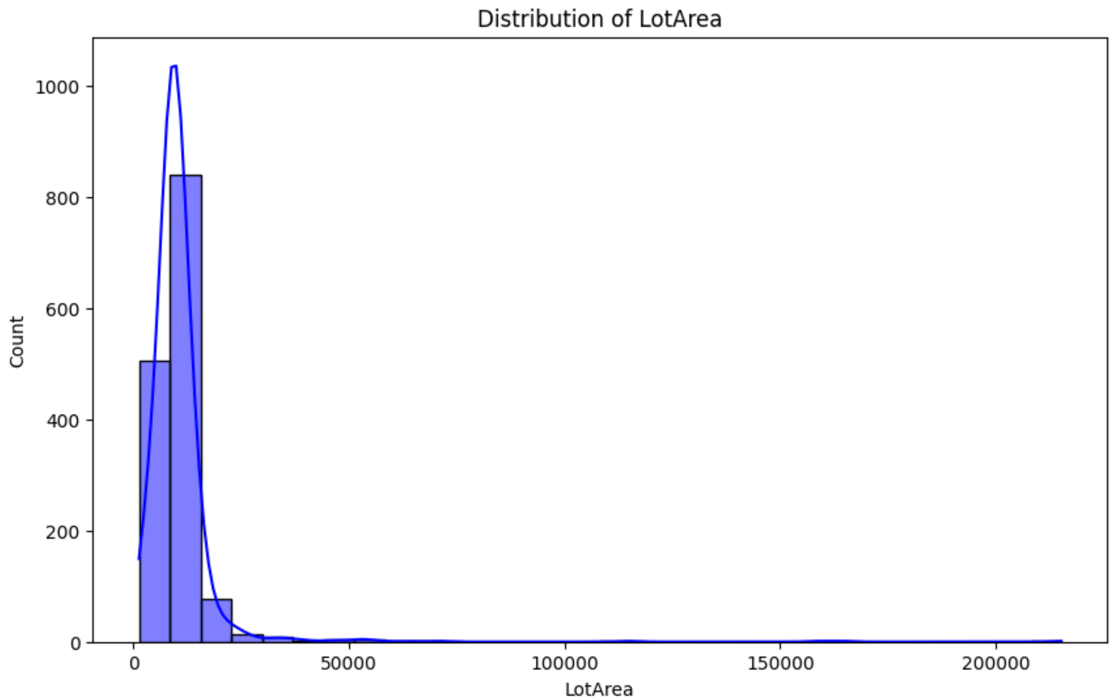

SalePrice vs SaleCondition
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Normal
Abnorml
Partial
AdjLand
Alloca
Family
SaleCondition

[ ]:
```python
# Visualizing the distribution of numerical features
numerical_features = train_data.select_dtypes(include=['int64', 'float64']).
↪columns
for feature in numerical_features:
    plt.figure(figsize=(10,6))
    sns.histplot(train_data[feature], kde=True, color='b', bins=30)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.show()
```

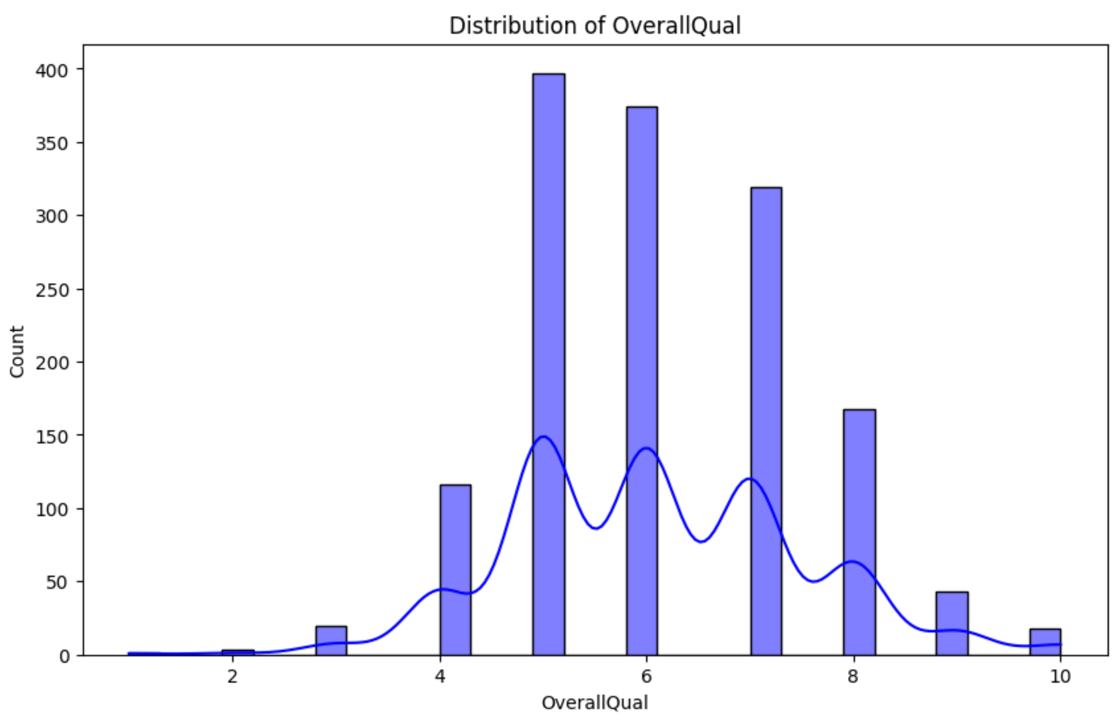



Distribution of MSSubClass
Count
MSSubClass
0
100
200
300
400
500
25
50
75
100
125
150
175

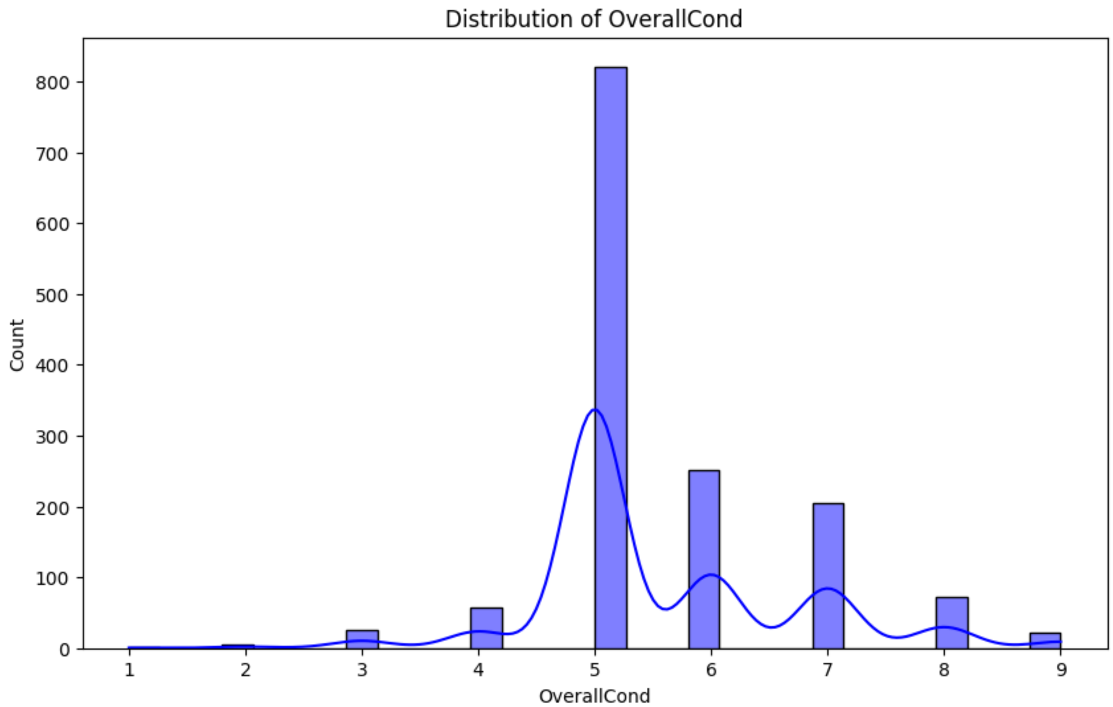

Distribution of LotFrontage
Count
LotFrontage
0
50
100
150
200
250
300
50
100
150
200
250
300

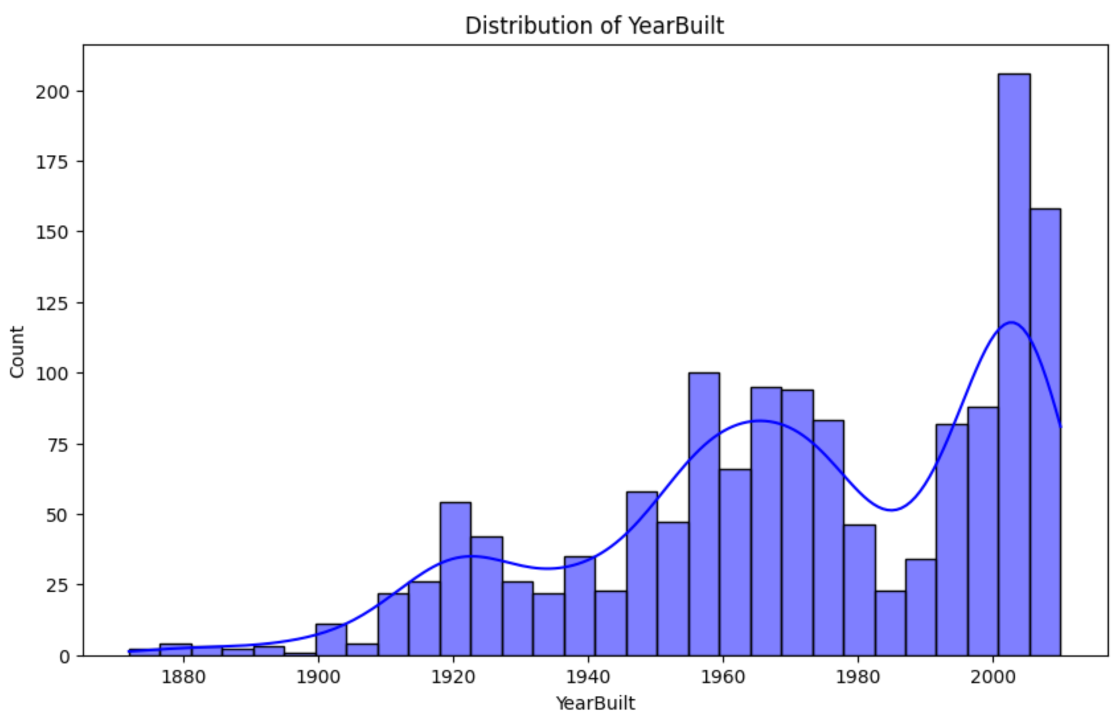


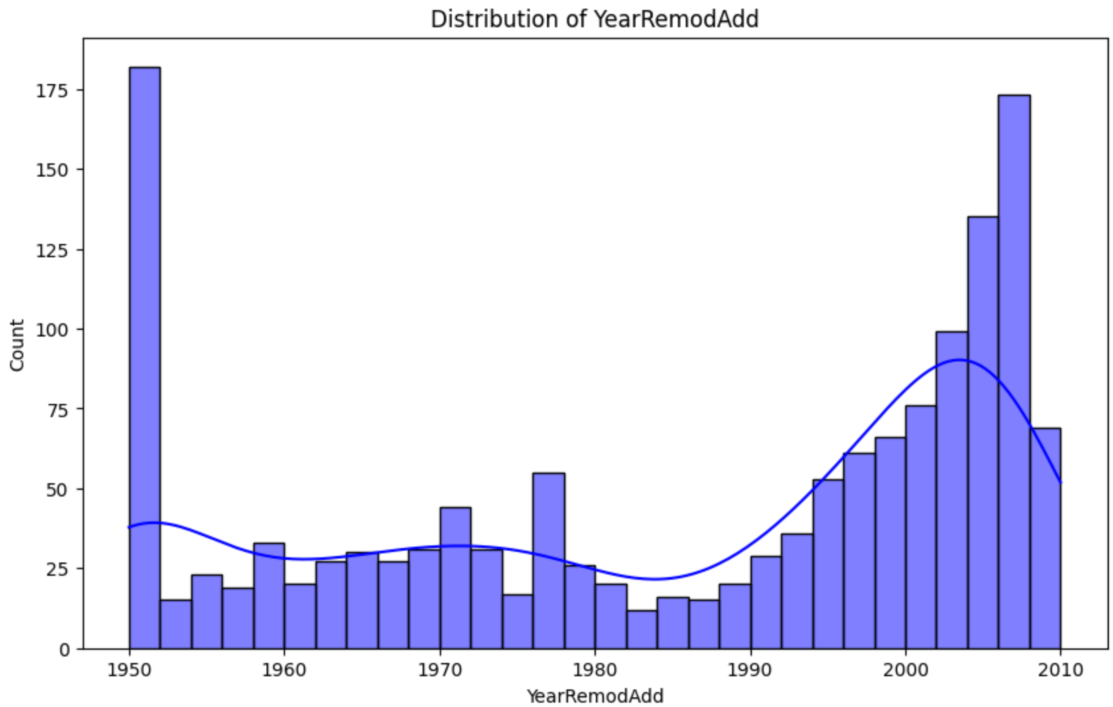
Distribution of LotArea
Count
LotArea
0
200
400
600
800
1000
0
50000
100000
150000
200000

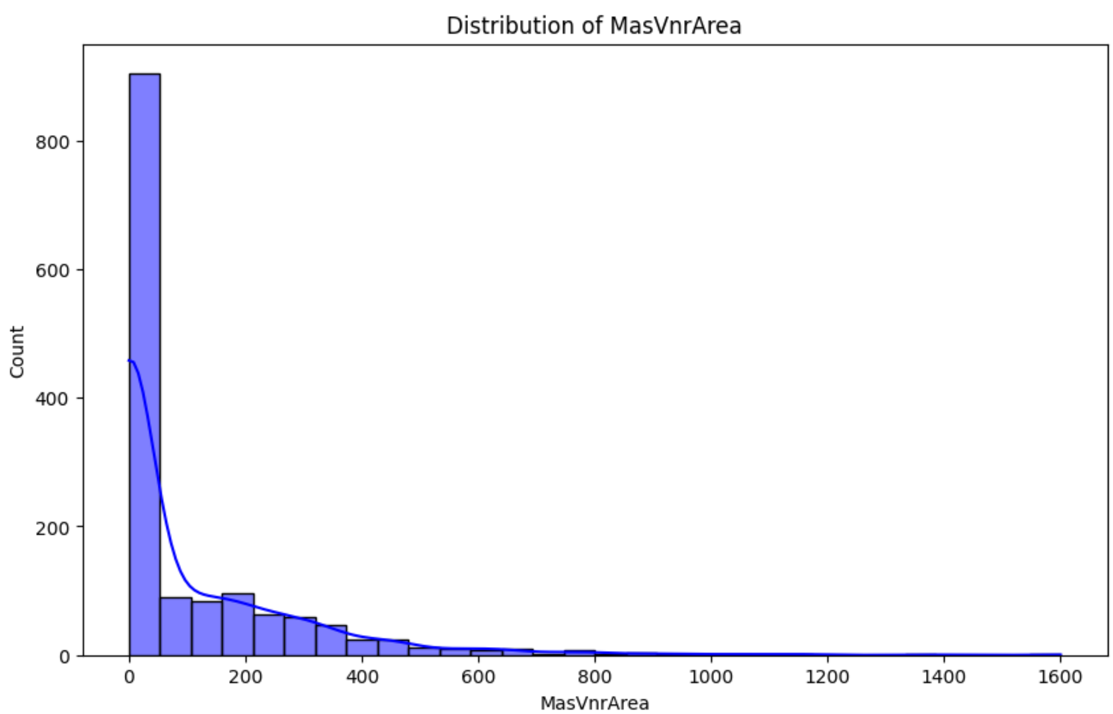


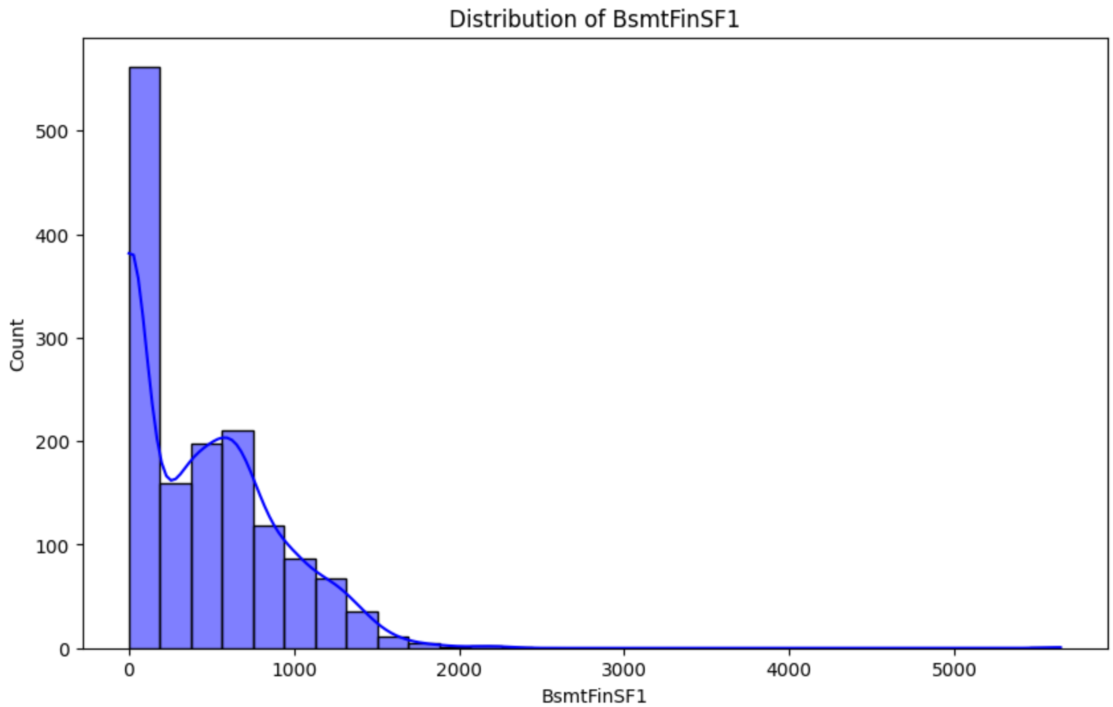
Distribution of OverallQual
Count
OverallQual
0
50
100
150
200
250
300
350
400
2
4
6
8
10

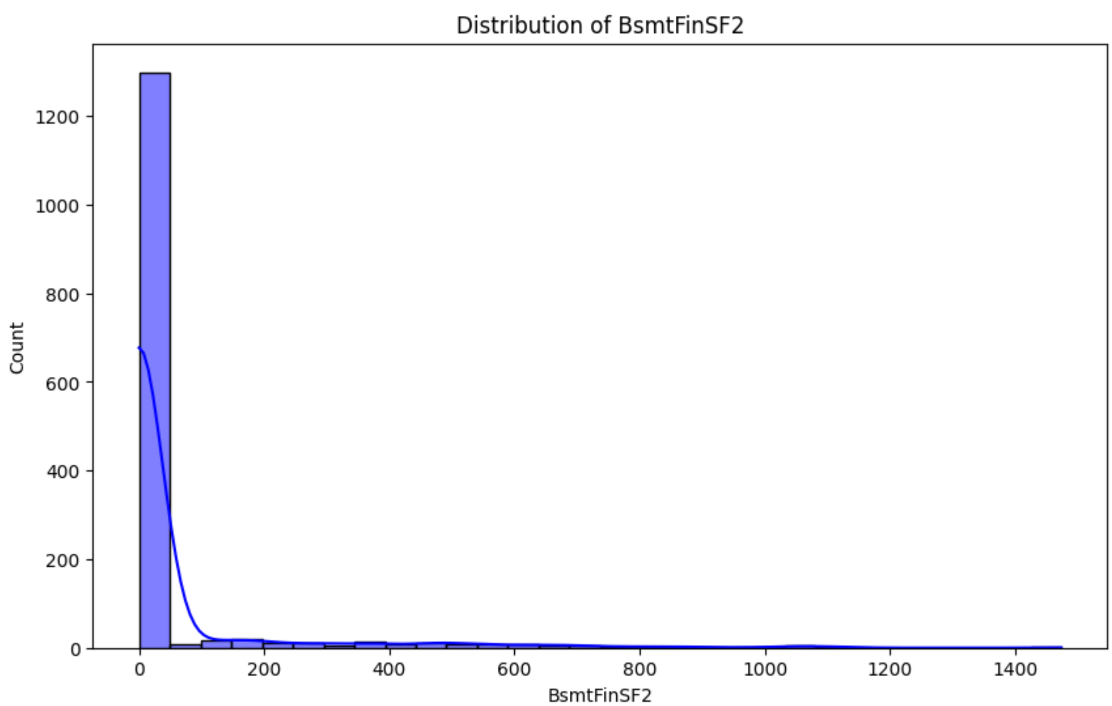

Distribution of OverallCond
Count
OverallCond
0
100
200
300
400
500
600
700
800
1
2
3
4
5
6
7
8
9

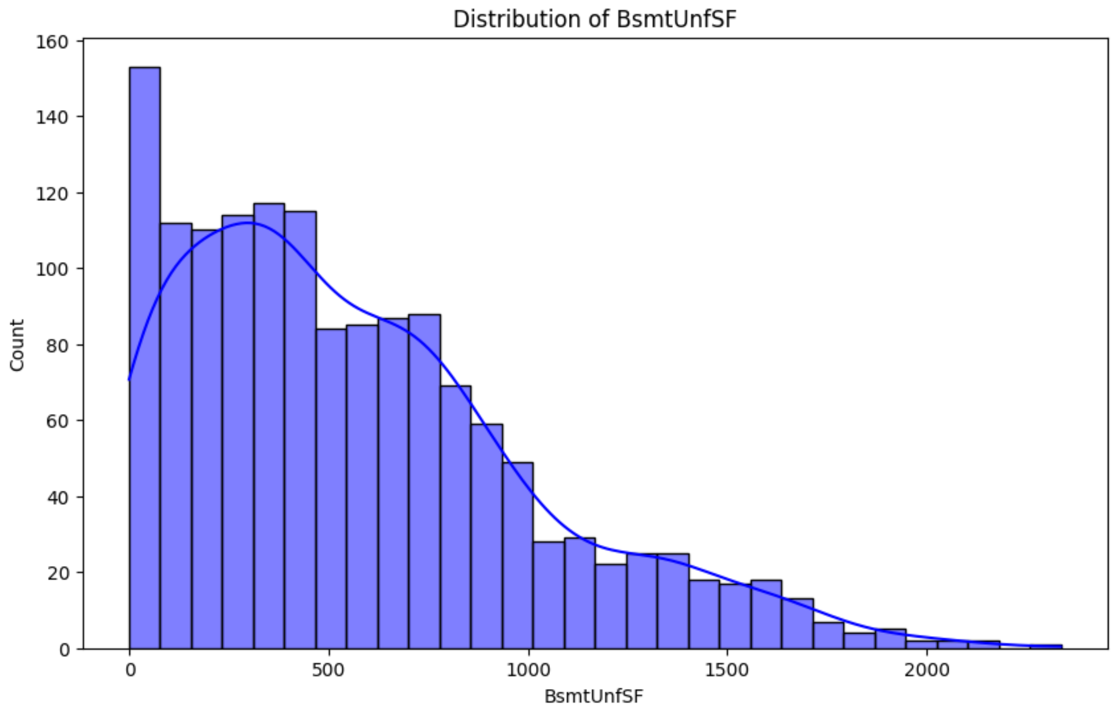

Distribution of YearBuilt
Count
YearBuilt
0
25
50
75
100
125
150
175
200
1880
1900
1920
1940
1960
1980
2000

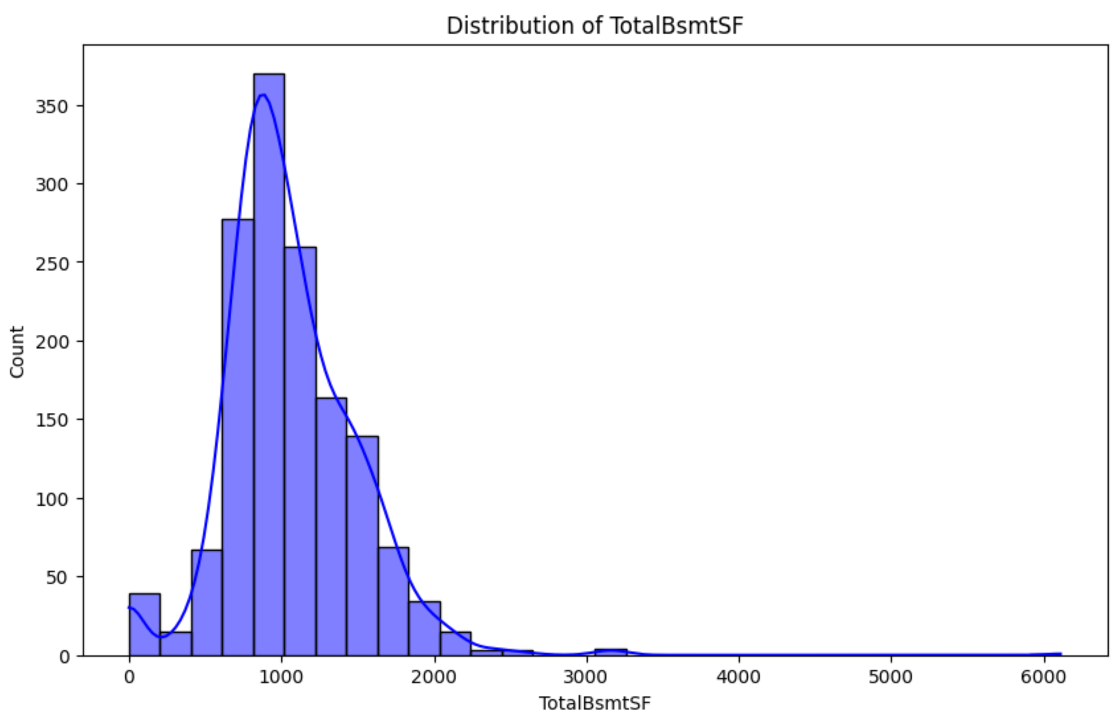

Distribution of YearRemodAdd
Count
175
150
125
100
75
50
25
0
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd

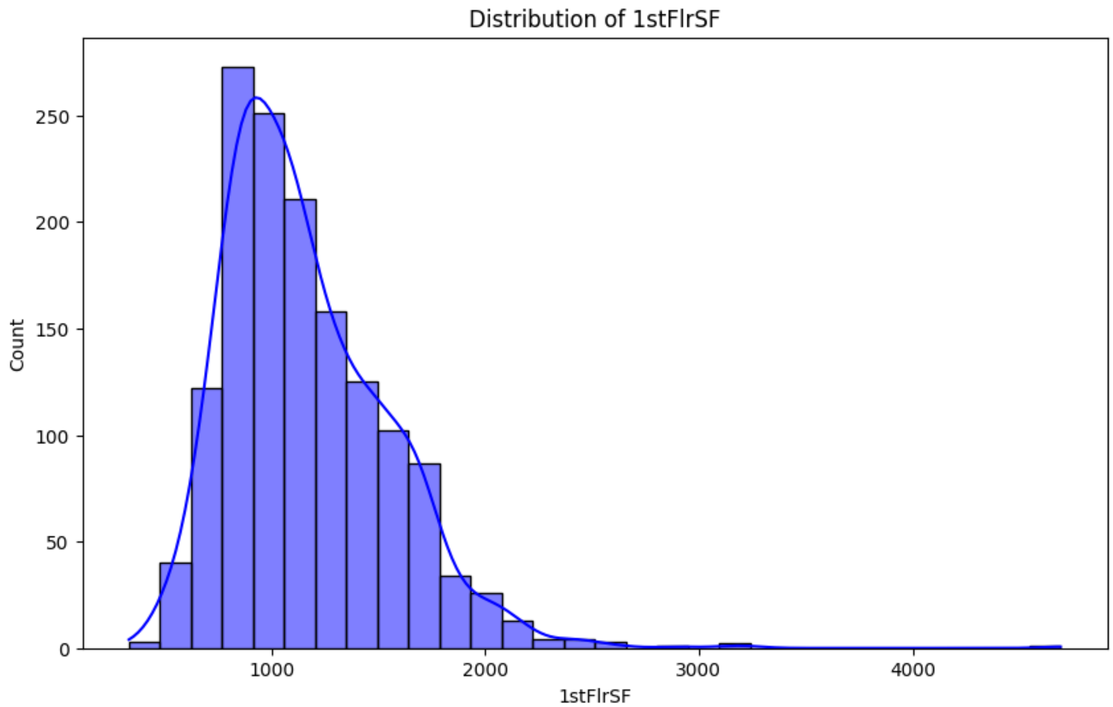

Distribution of MasVnrArea
Count
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

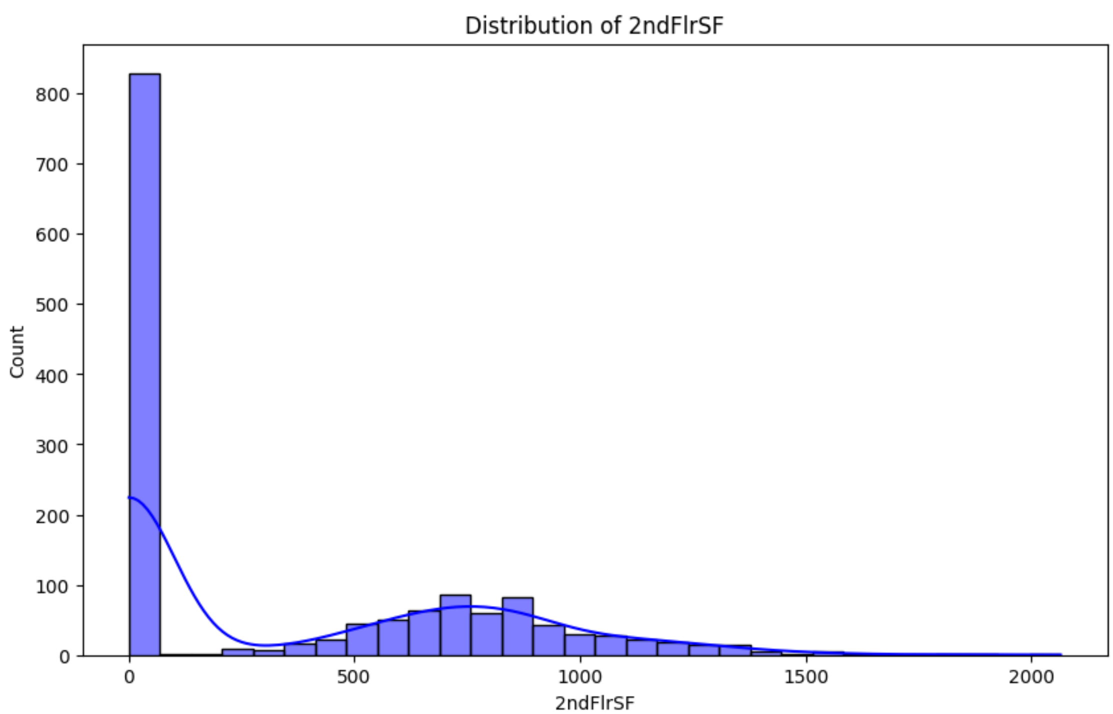

Distribution of BsmtFinSF1
Count
BsmtFinSF1
0
100
200
300
400
500
1000
2000
3000
4000
5000

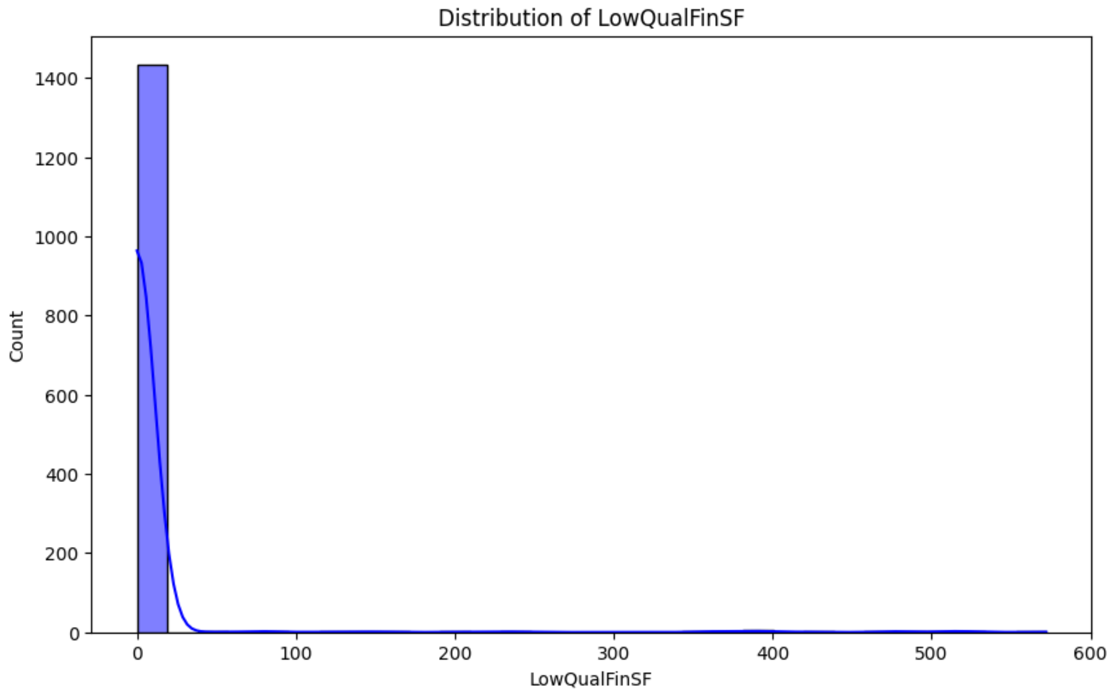

Distribution of BsmtFinSF2
Count
BsmtFinSF2
0
200
400
600
800
1000
1200
1400

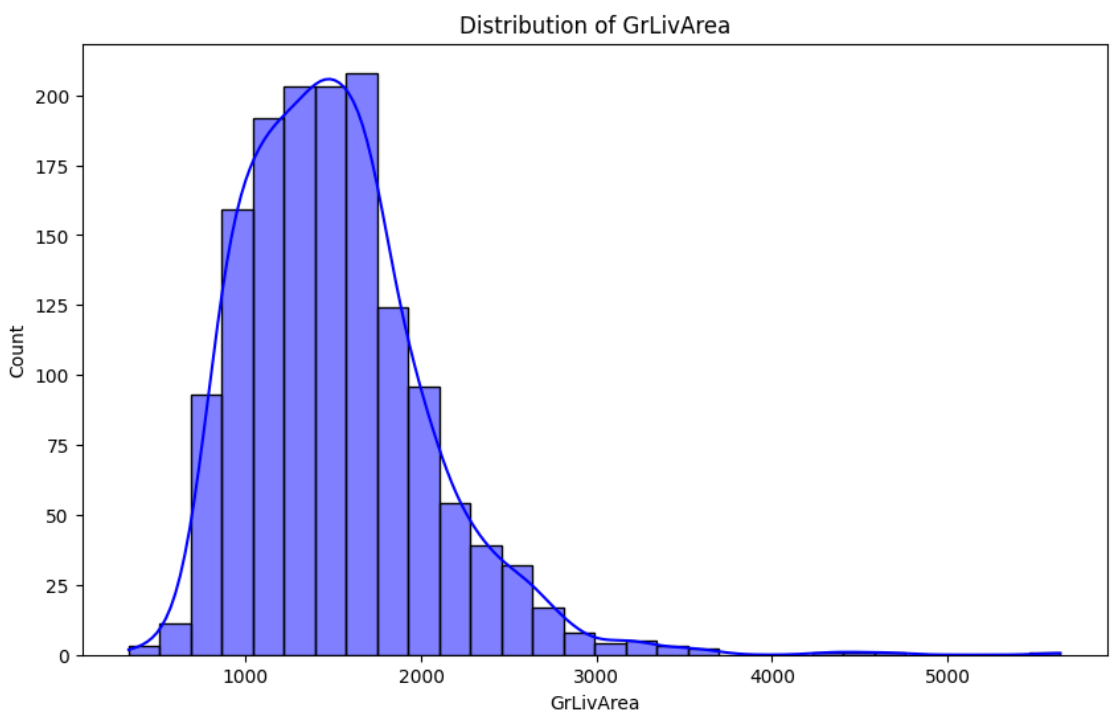

Distribution of BsmtUnfSF
Count
BsmtUnfSF
0
20
40
60
80
100
120
140
160
0
500
1000
1500
2000

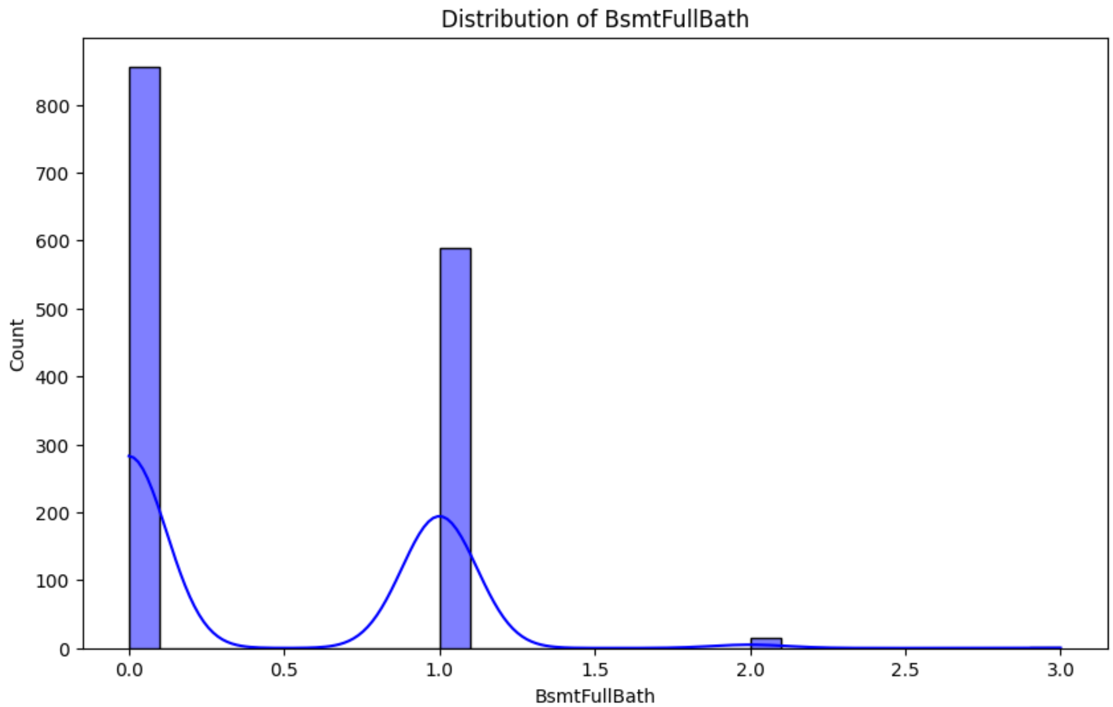

Distribution of TotalBsmtSF
Count
TotalBsmtSF
0
50
100
150
200
250
300
350
0
1000
2000
3000
4000
5000
6000

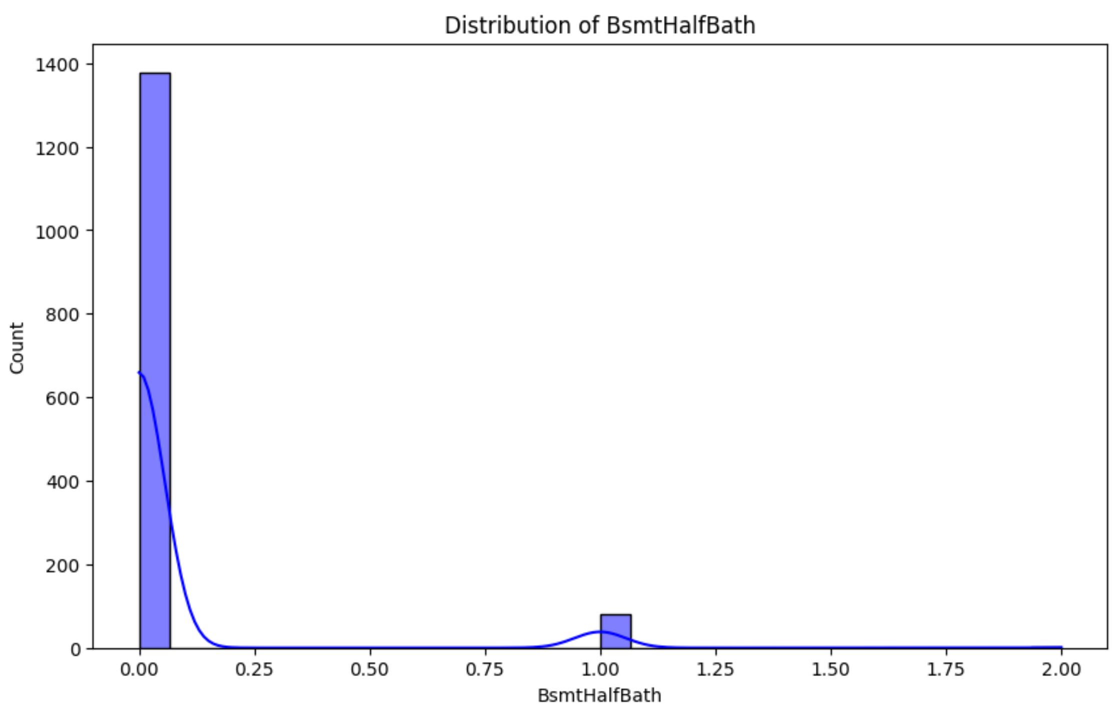

Distribution of 1stFlrSF
Count
0
50
100
150
200
250
1000
2000
3000
4000
1stFlrSF

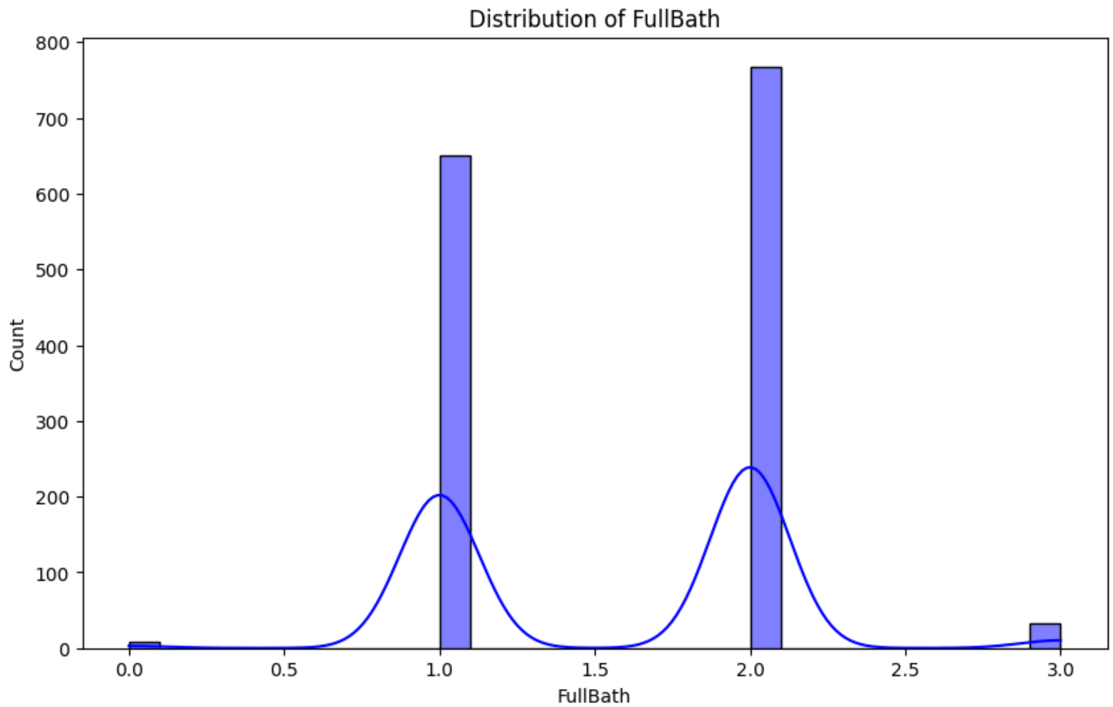

Distribution of 2ndFlrSF
Count
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000
2ndFlrSF

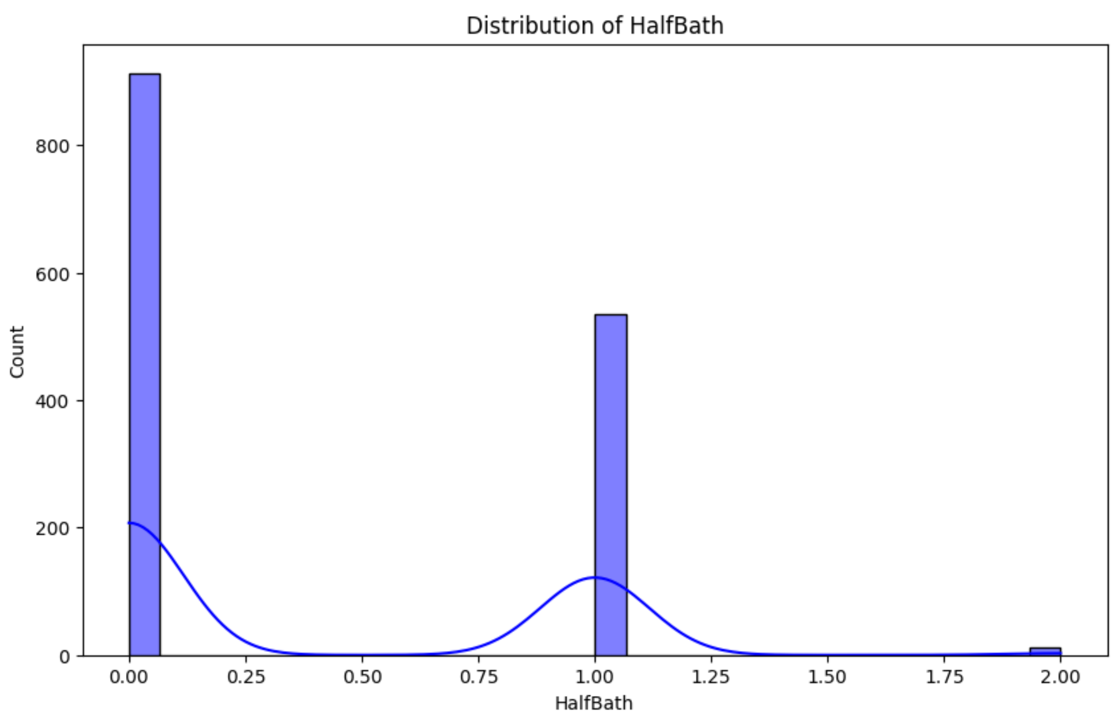

Distribution of LowQualFinSF
Count
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
LowQualFinSF

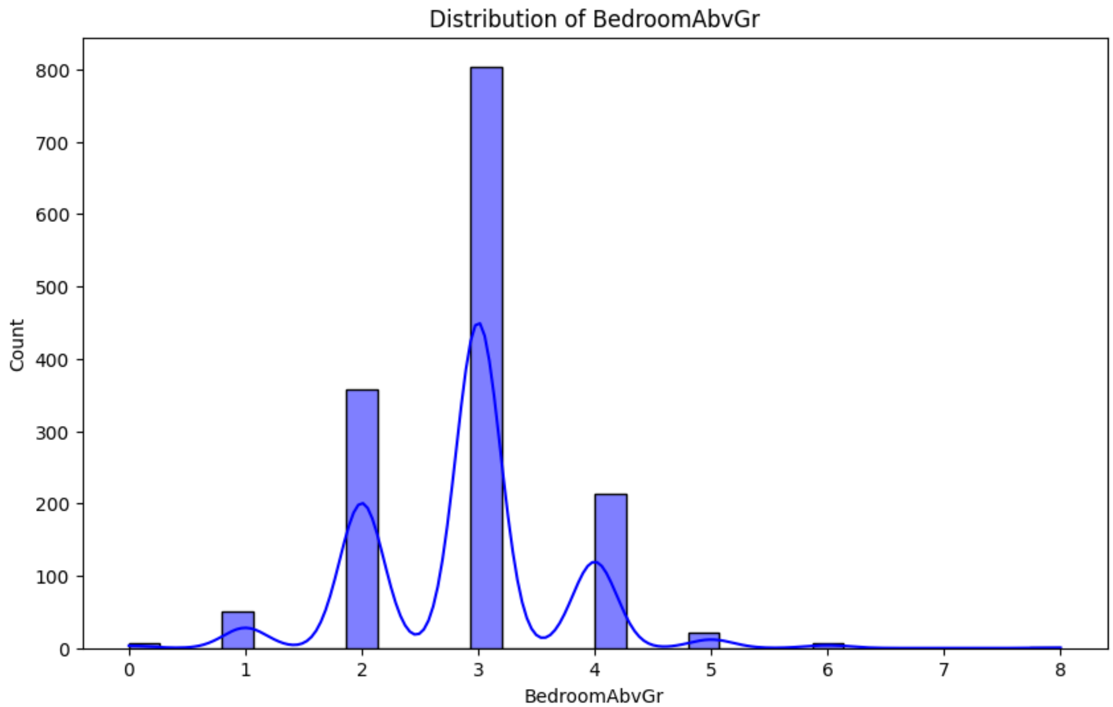

Distribution of GrLivArea
Count
200
175
150
125
100
75
50
25
0
1000
2000
3000
4000
5000
GrLivArea

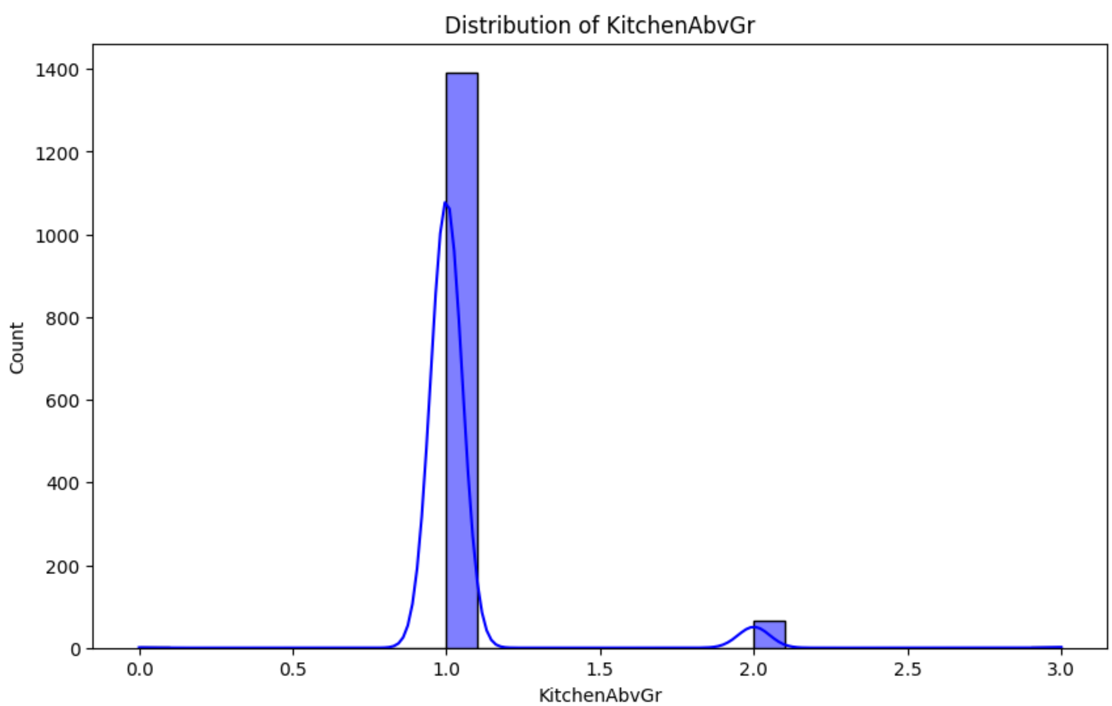

Distribution of BsmtFullBath
Count
BsmtFullBath
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0

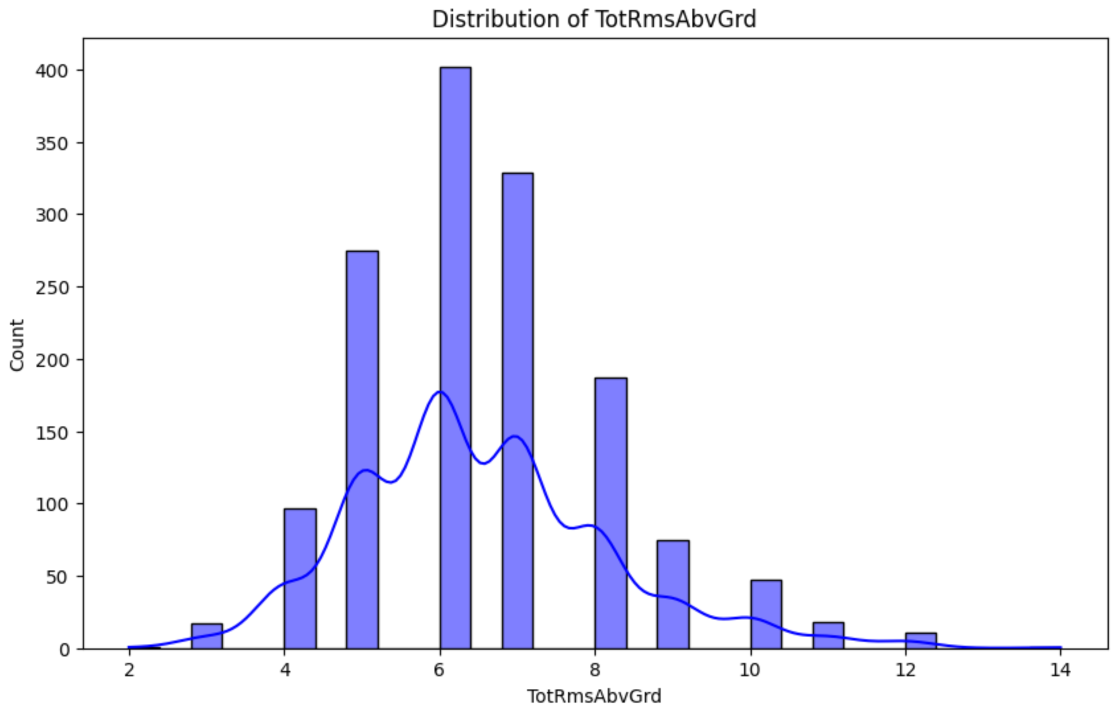

Distribution of BsmtHalfBath
Count
BsmtHalfBath
0
200
400
600
800
1000
1200
1400
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

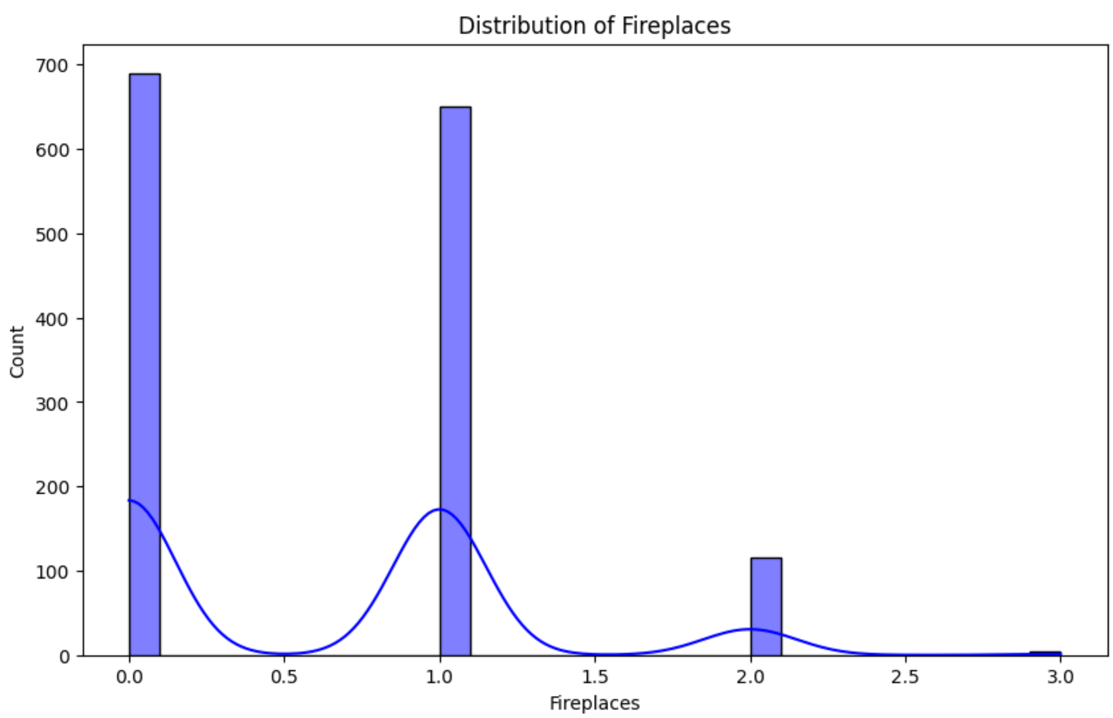

Distribution of FullBath
Count
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
FullBath

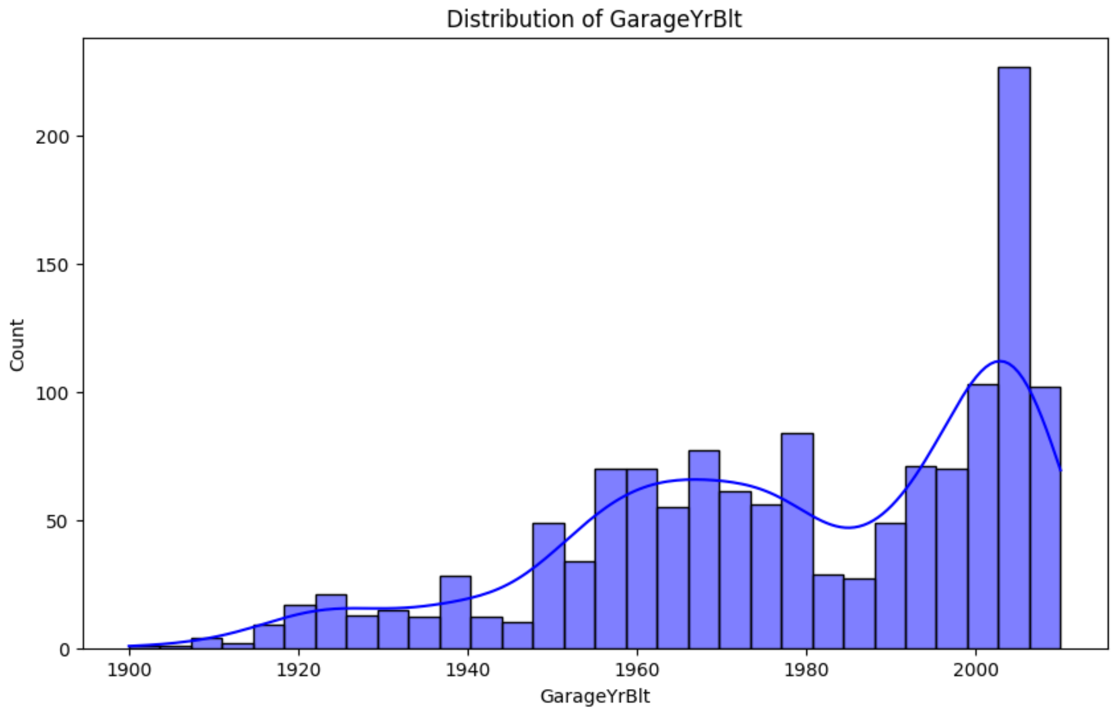

Distribution of HalfBath
Count
800
600
400
200
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
HalfBath

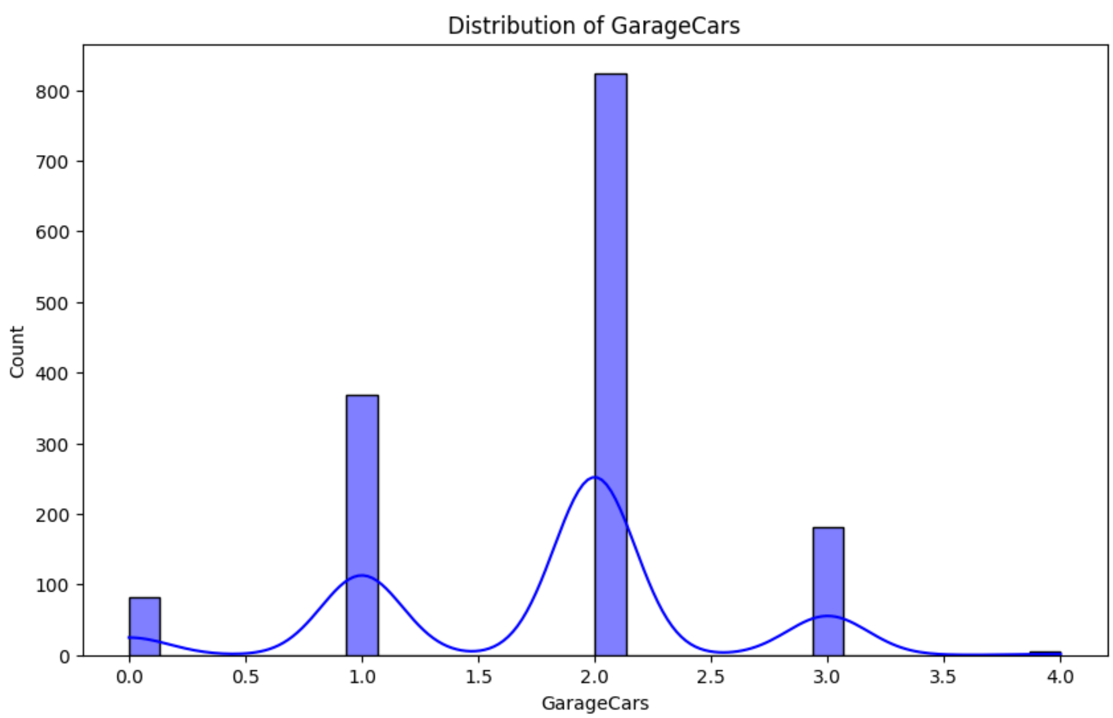

Distribution of BedroomAbvGr
Count
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6
7
8
BedroomAbvGr

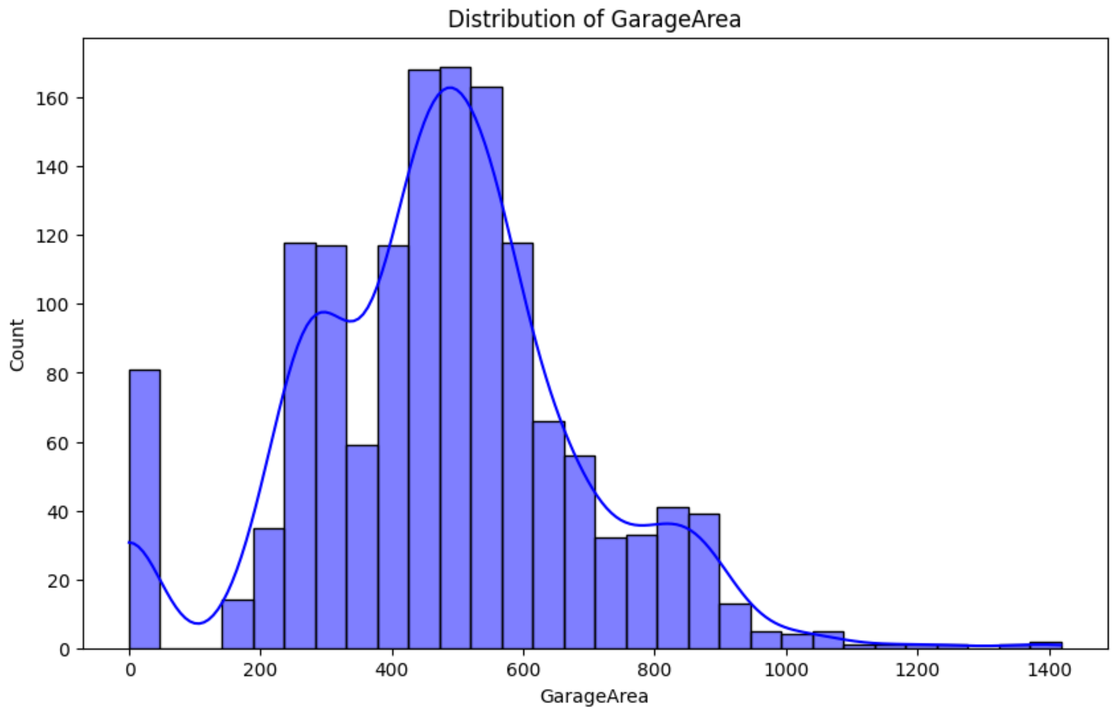

Distribution of KitchenAbvGr
Count
0
200
400
600
800
1000
1200
1400
0.0
0.5
1.0
1.5
2.0
2.5
3.0
KitchenAbvGr

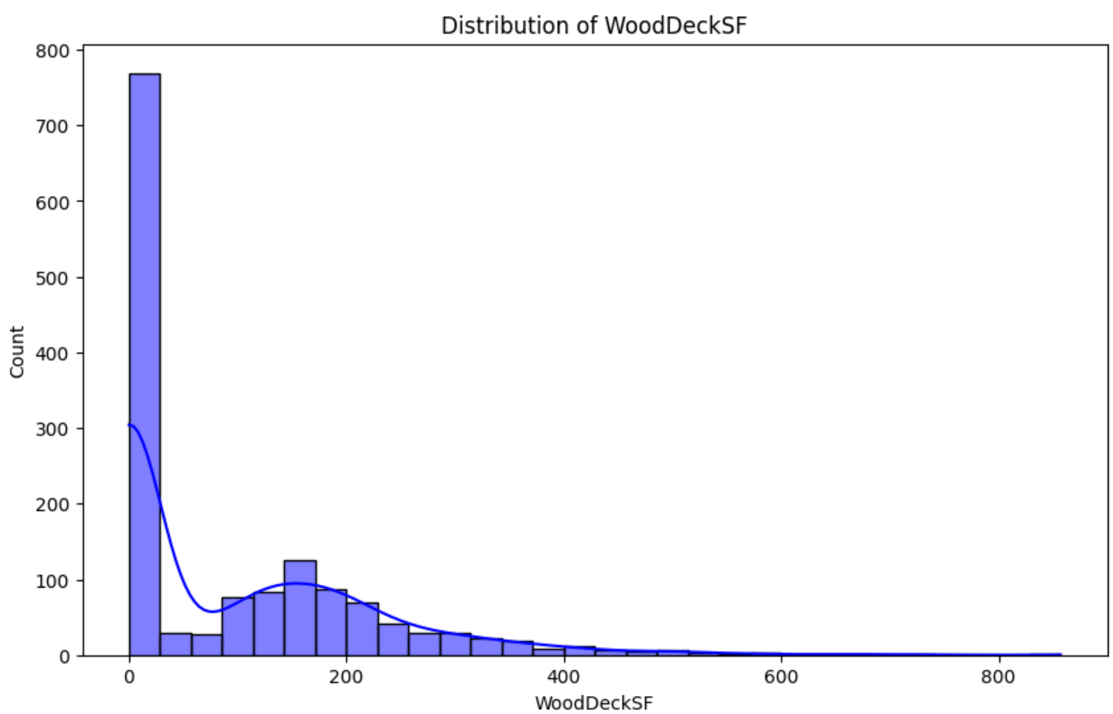

Distribution of TotRmsAbvGrd
Count
TotRmsAbvGrd
0
50
100
150
200
250
300
350
400
2
4
6
8
10
12
14

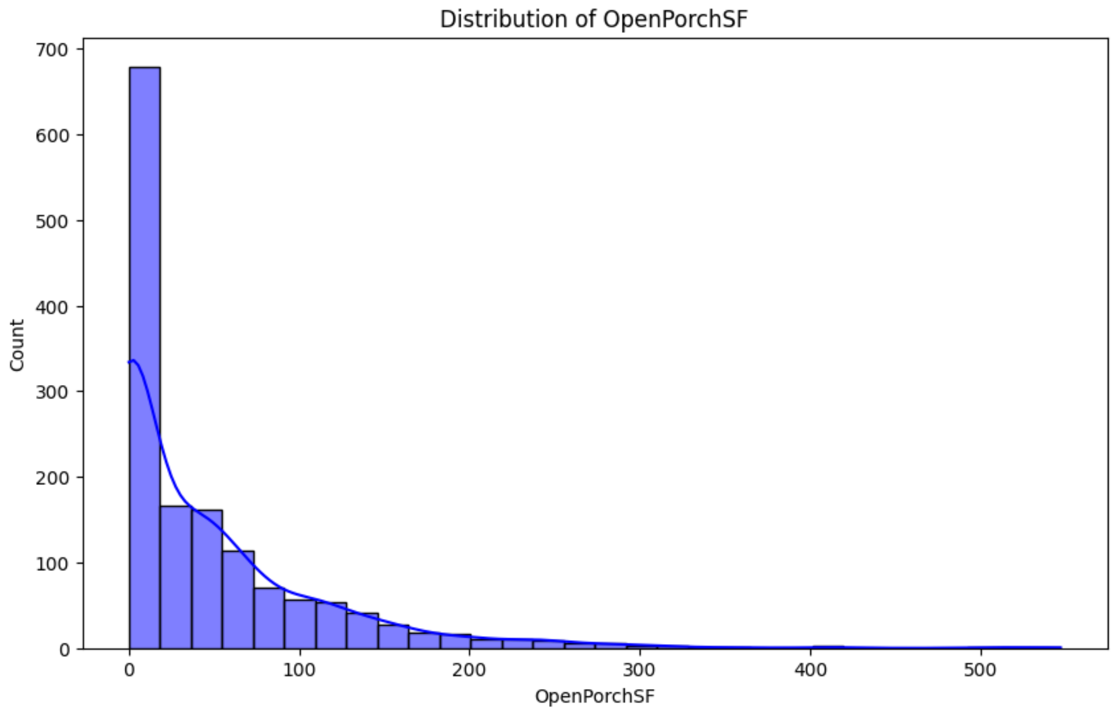

Distribution of Fireplaces
Count
Fireplaces
0
100
200
300
400
500
600
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0

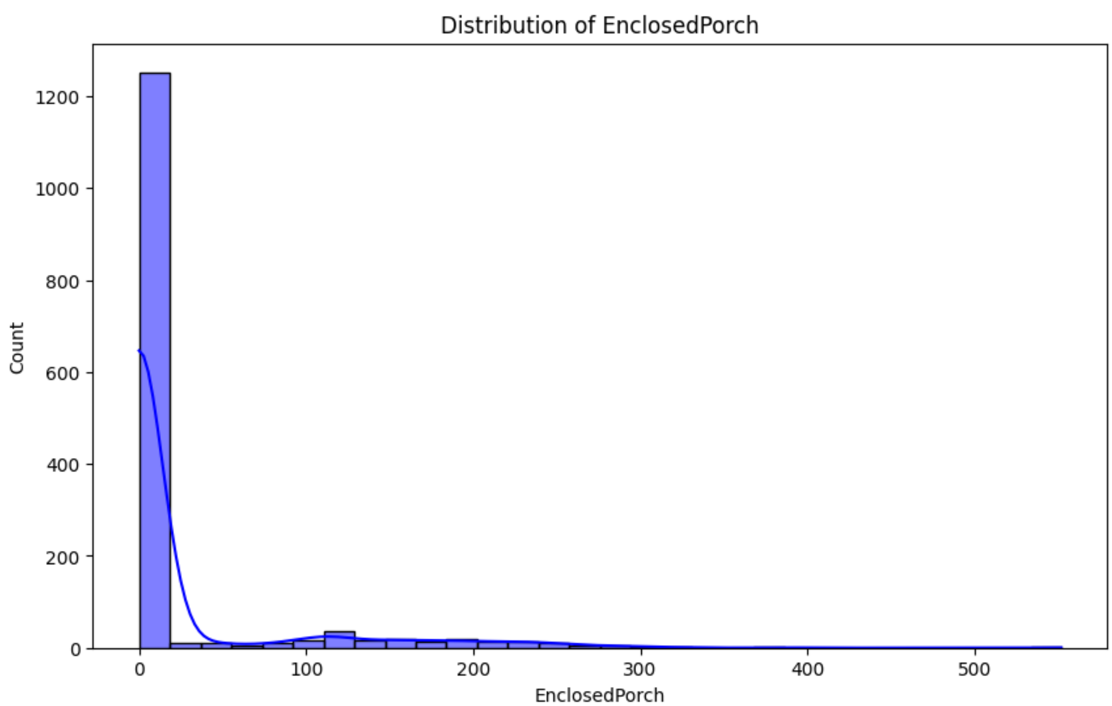

Distribution of GarageYrBlt
Count
0
50
100
150
200
1900
1920
1940
1960
1980
2000
GarageYrBlt

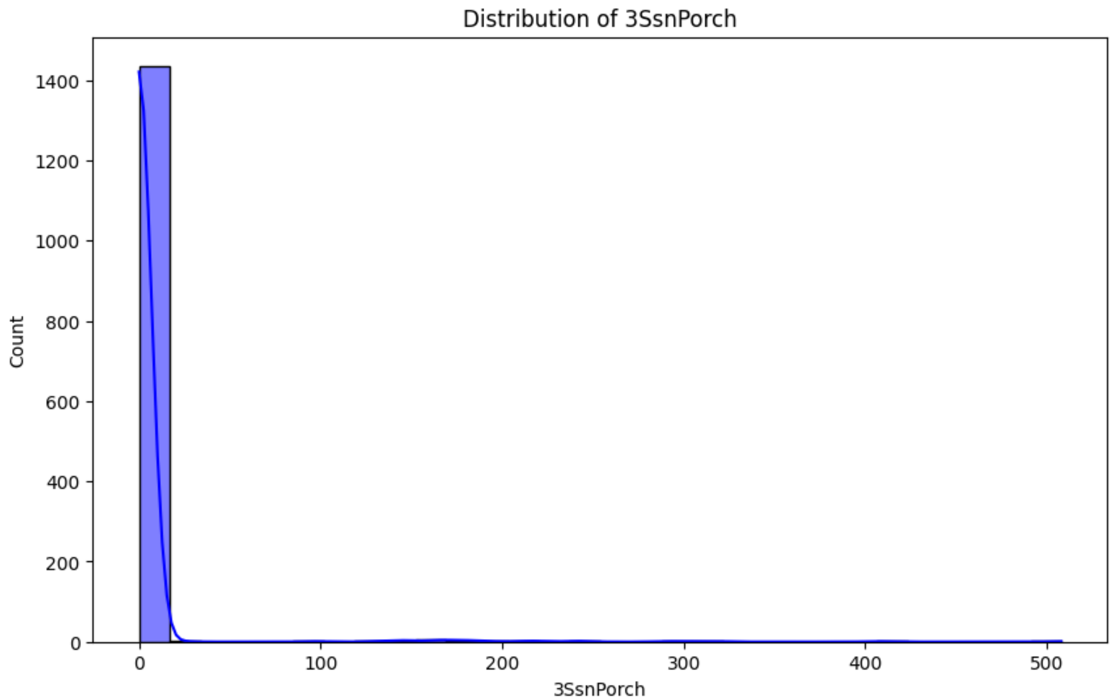

Distribution of GarageCars
Count
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
GarageCars


Distribution of GarageArea
Count
GarageArea
0
20
40
60
80
100
120
140
160
0
200
400
600
800
1000
1200
1400

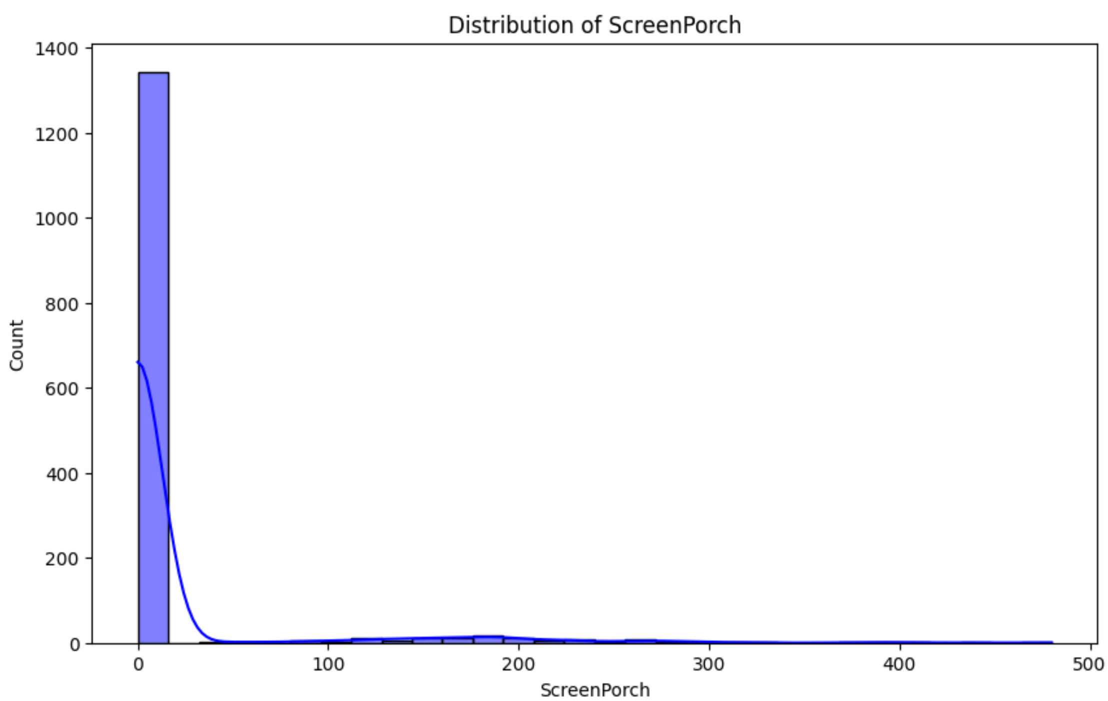

Distribution of WoodDeckSF
Count
WoodDeckSF
0
100
200
300
400
500
600
700
800
0
200
400
600
800


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

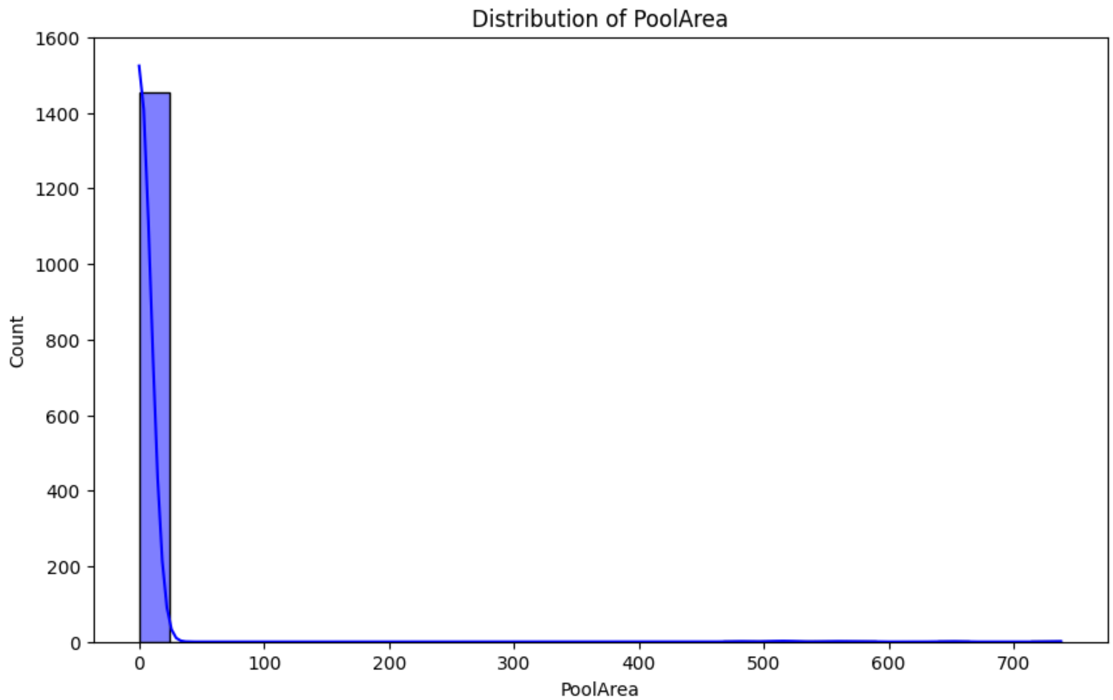

Distribution of EnclosedPorch
Count
0
200
400
600
800
1000
1200
0
100
200
300
400
500
EnclosedPorch

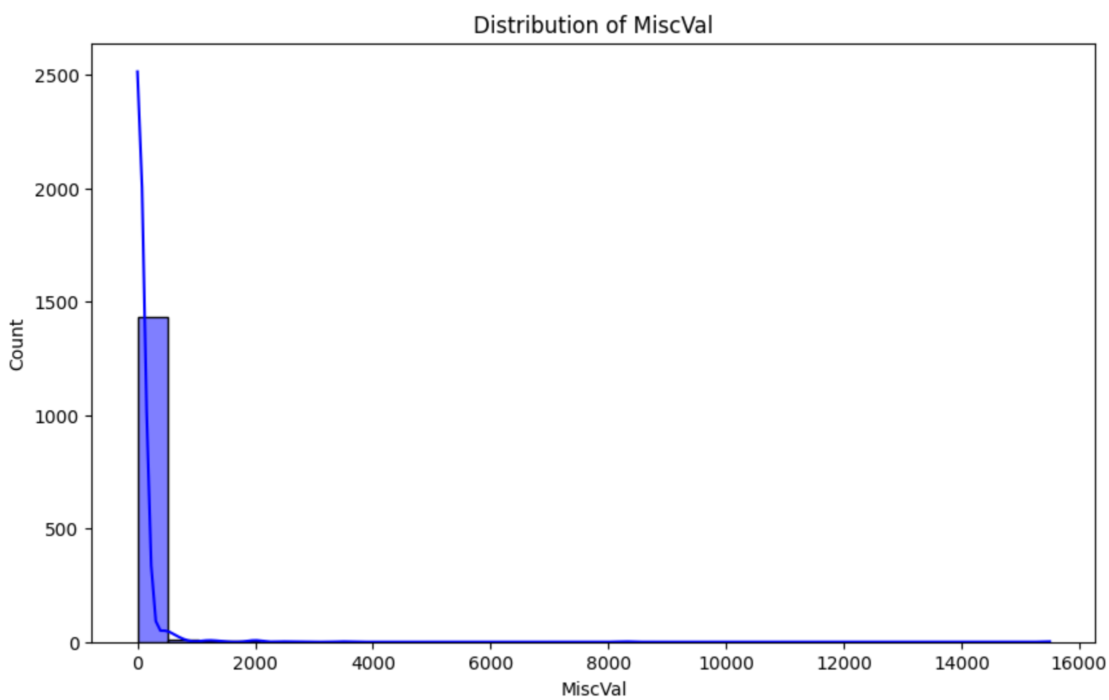


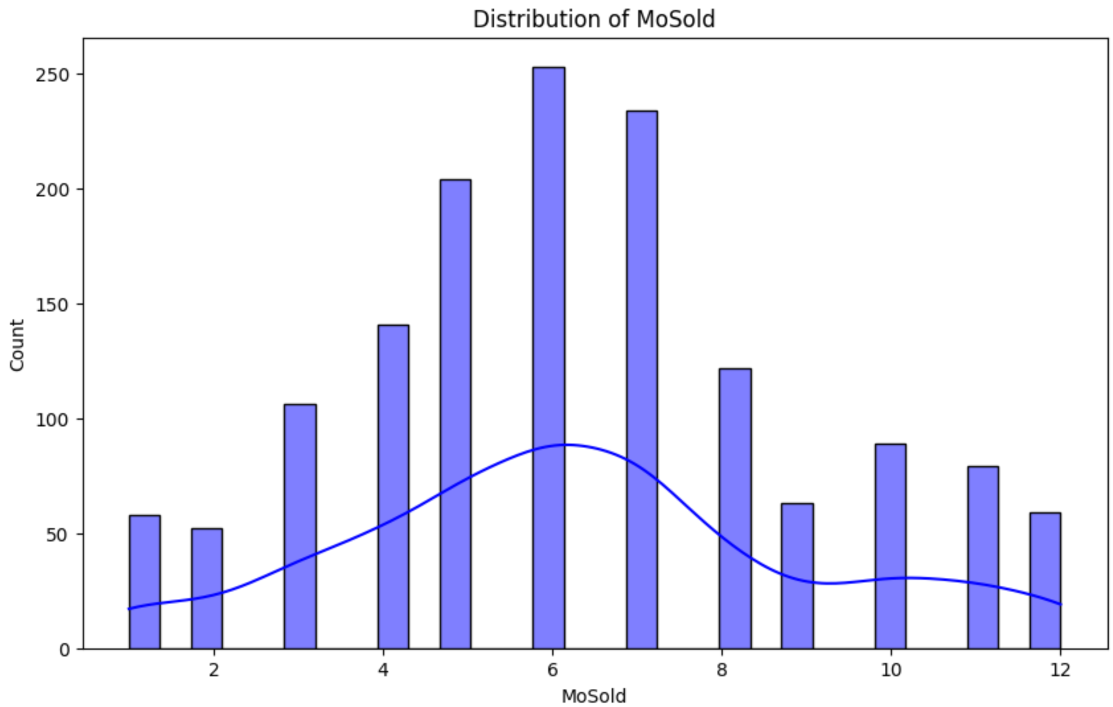
Distribution of 3SsnPorch
Count
0
200
400
600
800
1000
1200
1400
0
100
200
300
400
500
3SsnPorch

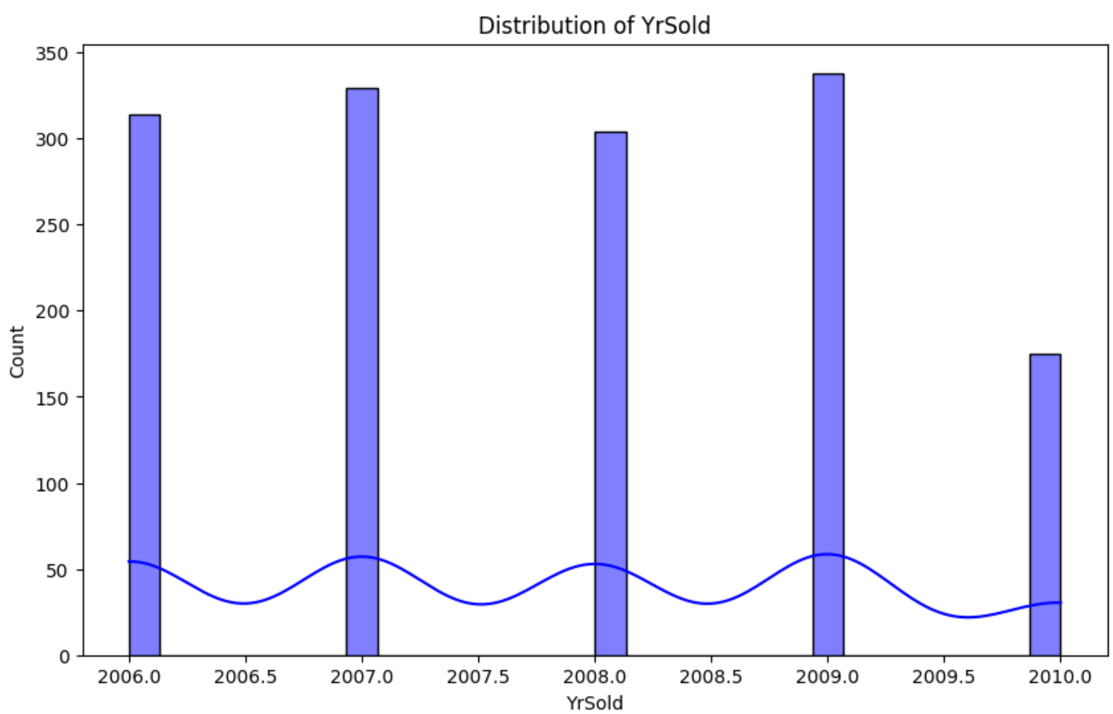


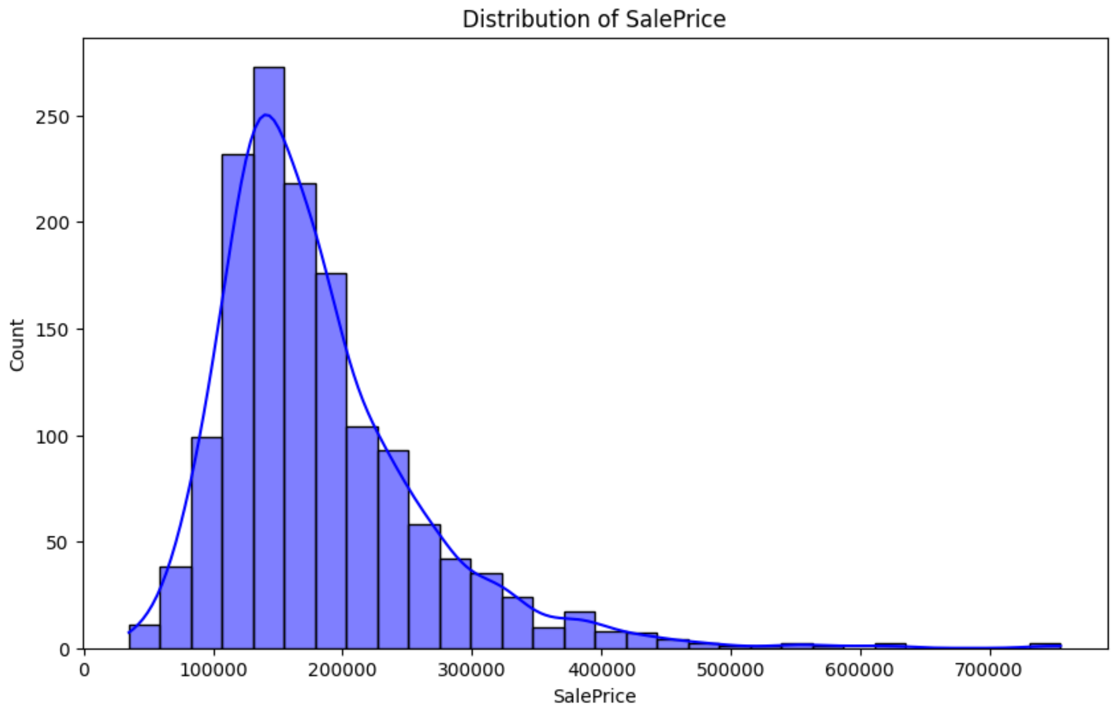
Distribution of ScreenPorch
Count
0
200
400
600
800
1000
1200
1400
0
100
200
300
400
500
ScreenPorch


Distribution of PoolArea
Count
1600
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
700
PoolArea

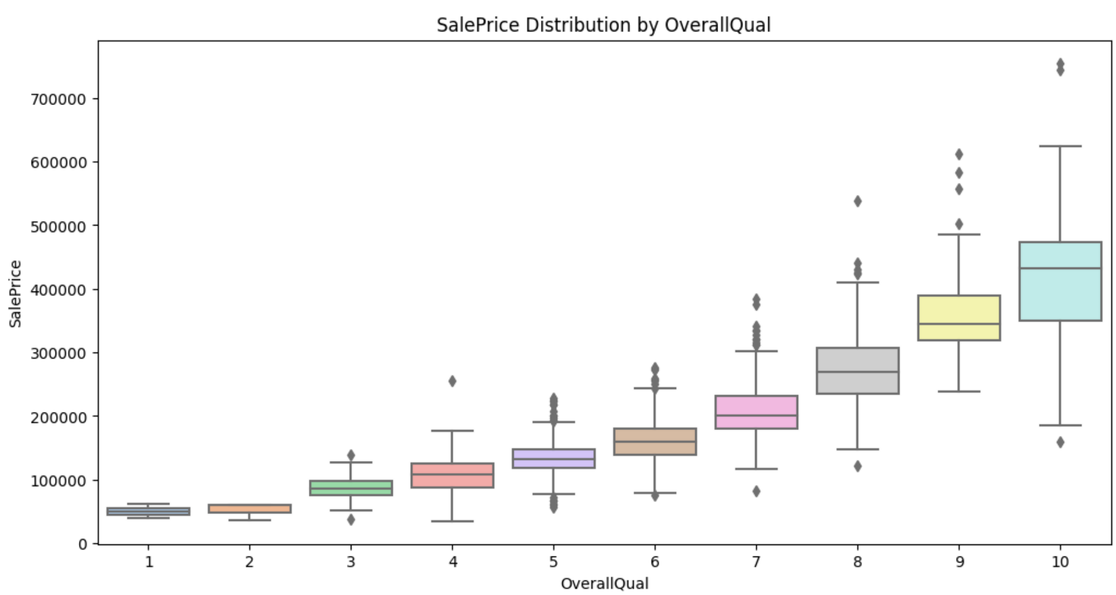

Distribution of MiscVal
Count
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000
12000
14000
16000
MiscVal


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

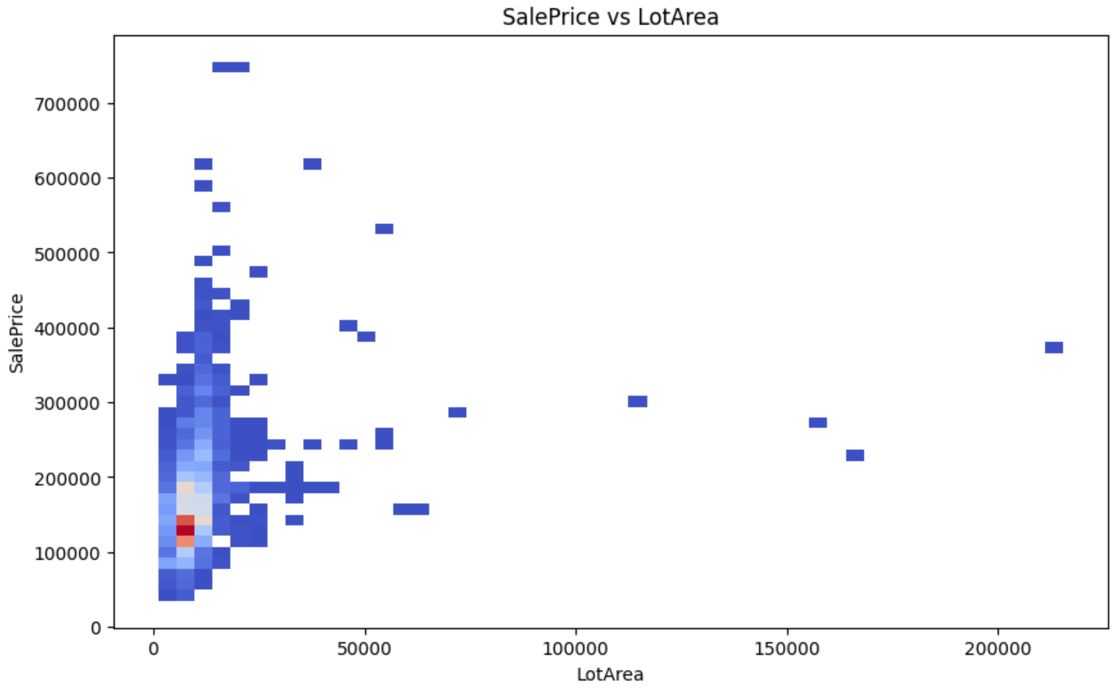

Distribution of YrSold
Count
YrSold
0
50
100
150
200
250
300
350
2006.0
2006.5
2007.0
2007.5
2008.0
2008.5
2009.0
2009.5
2010.0



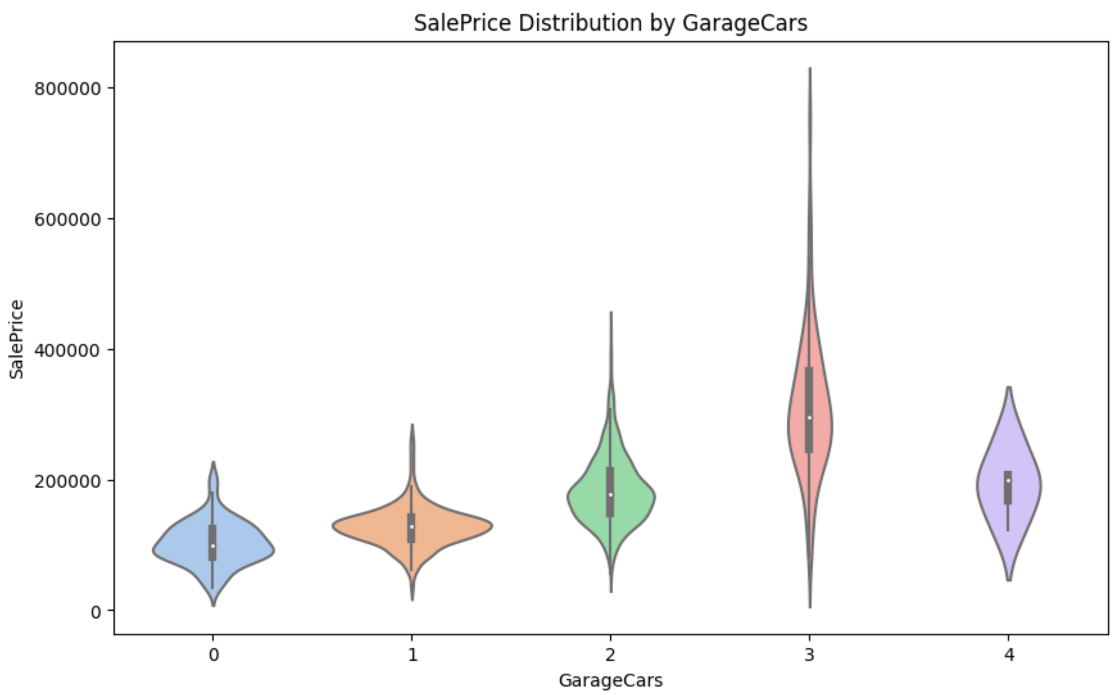
[ ]:
```
# Visualizing the distribution of categorical variables with a box plot
plt.figure(figsize=(12, 6))
sns.boxplot(x='OverallQual', y='SalePrice', data=train_data, palette='pastel')
plt.title('SalePrice Distribution by OverallQual')
plt.xlabel('OverallQual')
plt.ylabel('SalePrice')
plt.show()
```



[ ]:
```
# Visualizing the relationship between two variables with a cross-sectional␣
↪histogram
plt.figure(figsize=(10, 6))
sns.histplot(x='LotArea', y='SalePrice', data=train_data, bins=50,␣
↪cmap='coolwarm')
plt.title('SalePrice vs LotArea')
plt.xlabel('LotArea')
plt.ylabel('SalePrice')
plt.show()
```



[ ]:
```
# Visualizing the density of a variable with a violin plot
plt.figure(figsize=(10, 6))
sns.violinplot(x='GarageCars', y='SalePrice', data=train_data, palette='pastel')
plt.title('SalePrice Distribution by GarageCars')
plt.xlabel('GarageCars')
plt.ylabel('SalePrice')
plt.show()
```



1 Model training

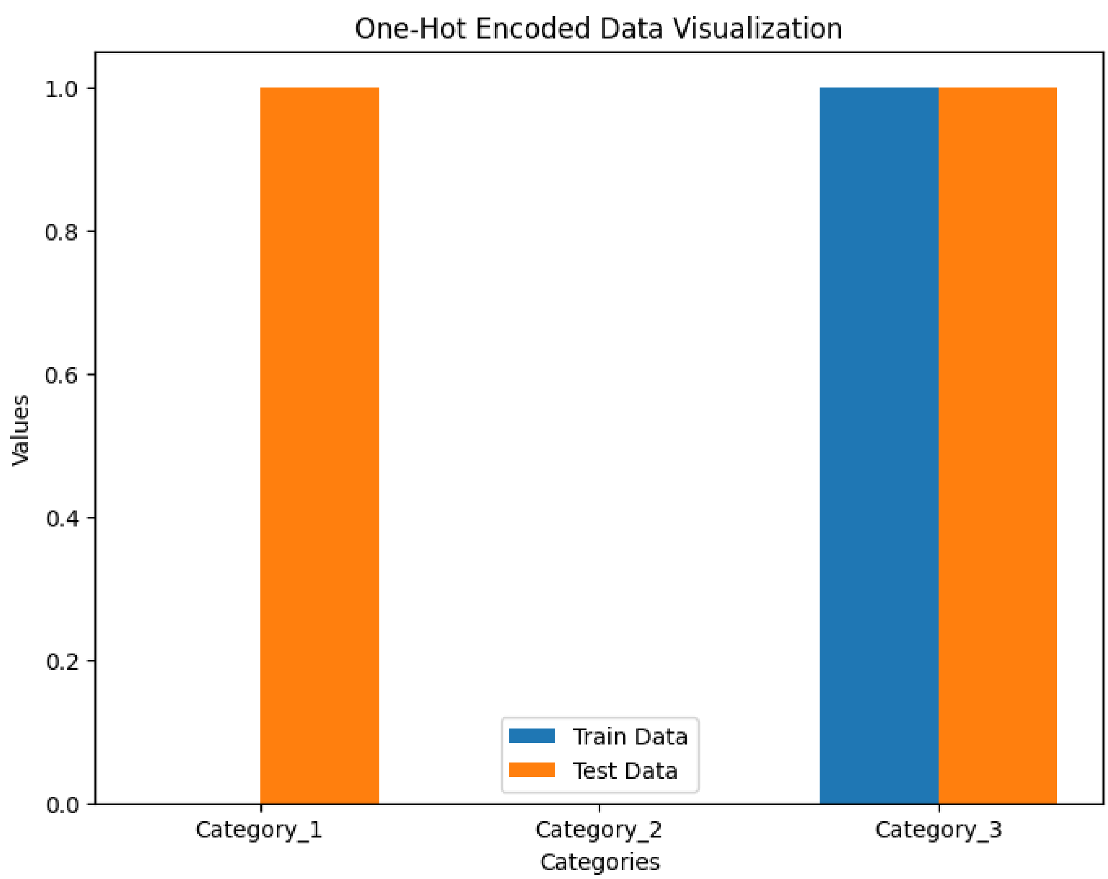
[ ]:
```
# Encoding categorical variables
from sklearn.preprocessing import OneHotEncoder
```

[ ]:
```
# Merge the first and second datasets
combined_data = pd.concat([train_data, test_data], axis=0)

# Identify categorical columns
categorical_cols = combined_data.select_dtypes(include=['object']).columns

# Apply one-hot encoding to categorical columns
OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
OH_cols_combined = pd.DataFrame(OH_encoder.
 ↪fit_transform(combined_data[categorical_cols]))

# Separate back the training and testing data
OH_cols_train = OH_cols_combined[:len(train_data)]
OH_cols_test = OH_cols_combined[len(train_data):]
```

[ ]:
```
# Sample data for visualization
categories = ['Category_1', 'Category_2', 'Category_3']
values_train = [np.random.randint(0, 2) for _ in range(len(categories))]
```

```
values_test = [np.random.randint(0, 2) for _ in range(len(categories))]
```

[ ]:
```
# Plotting the one-hot encoded data
fig, ax = plt.subplots(figsize=(8, 6))

bar_width = 0.35
index = np.arange(len(categories))

rects1 = ax.bar(index, values_train, bar_width, label='Train Data')
rects2 = ax.bar(index + bar_width, values_test, bar_width, label='Test Data')

ax.set_xlabel('Categories')
ax.set_ylabel('Values')
ax.set_title('One-Hot Encoded Data Visualization')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(categories)
ax.legend()

plt.show()
```

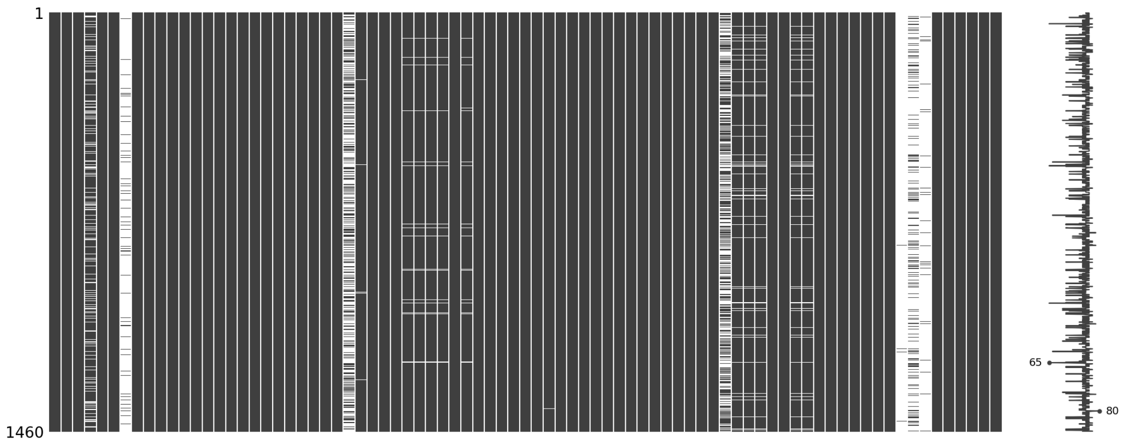


1.1 4. Feature Engineering:

1.1.1 Handling Missing Values:

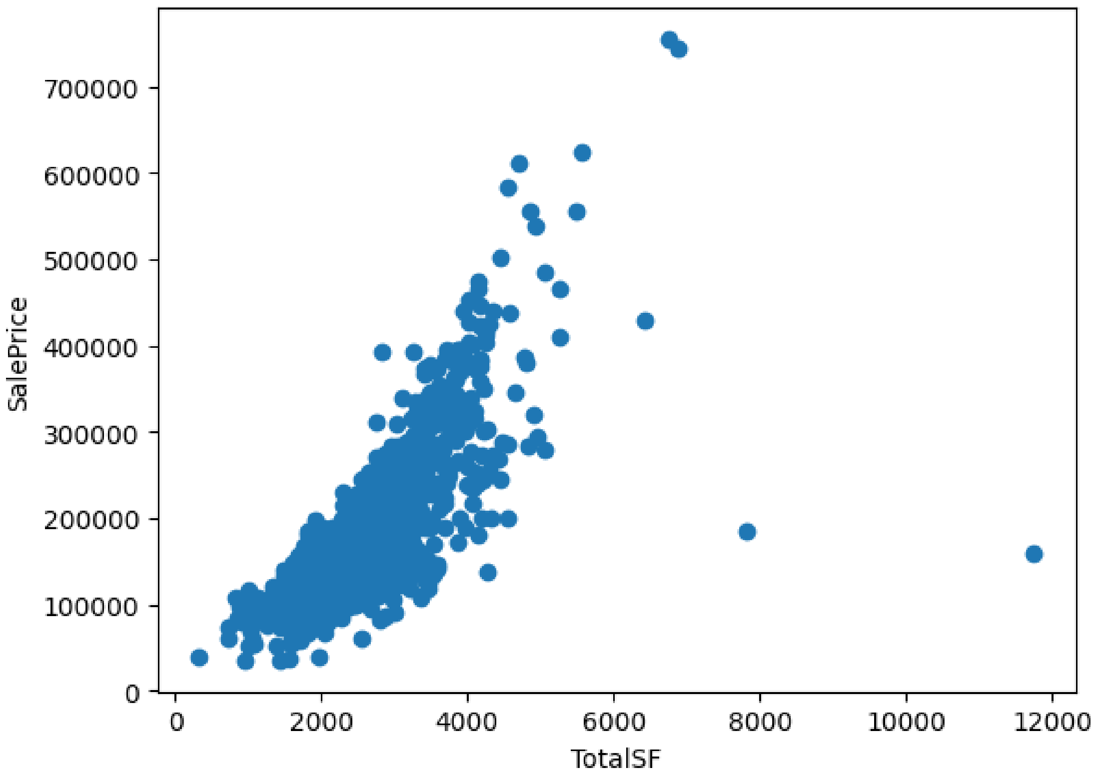
[ ]:
```
import missingno as msno
# Visualize missing values
msno.matrix(train_data)
```

[ ]: <Axes: >



Creating New Features:

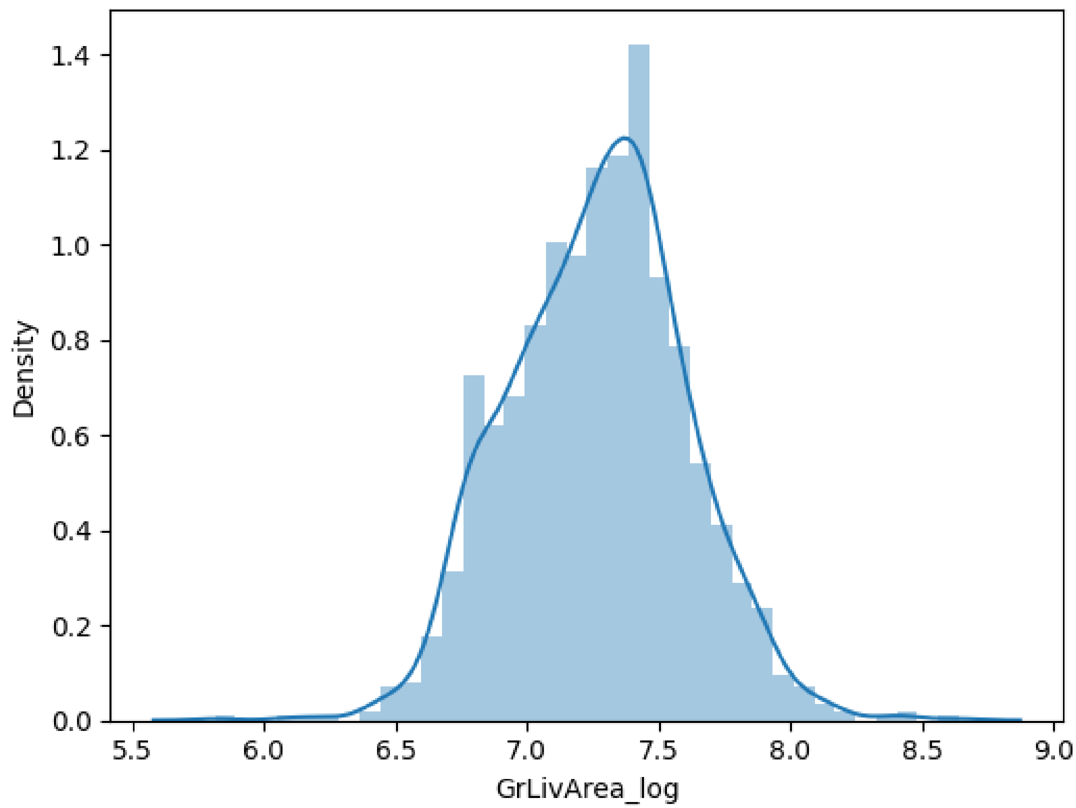
[ ]:
```
# Creating a new feature based on existing features
train_data['TotalSF'] = train_data['TotalBsmtSF'] + train_data['1stFlrSF'] +␣
↪train_data['2ndFlrSF']
# Visualize the new feature with the target variable
import matplotlib.pyplot as plt
plt.scatter(train_data['TotalSF'], train_data['SalePrice'])
plt.xlabel('TotalSF')
plt.ylabel('SalePrice')
plt.show()
```



Transforming Existing Features:

```
[ ]: # Applying a log transformation to a skewed feature
train_data['GrLivArea_log'] = np.log1p(train_data['GrLivArea'])
# Visualize the transformation
import seaborn as sns
sns.distplot(train_data['GrLivArea_log'])
```

```
[ ]: <Axes: xlabel='GrLivArea_log', ylabel='Density'>
```



Encoding Categorical Variables:

[ ]:
```
# Using one-hot encoding for categorical variables
categorical_cols = ['MSZoning', 'Street', 'Alley']
for col in categorical_cols:
    dummies = pd.get_dummies(train_data[col], prefix=col, drop_first=True)
    train_data = pd.concat([train_data, dummies], axis=1)
```

[ ]:
```
print(train_data[['MSZoning', 'Street', 'Alley']].head(10))
```

```
  MSZoning Street Alley
0       RL   Pave   NaN
1       RL   Pave   NaN
2       RL   Pave   NaN
3       RL   Pave   NaN
4       RL   Pave   NaN
5       RL   Pave   NaN
6       RL   Pave   NaN
7       RL   Pave   NaN
8       RM   Pave   NaN
9       RL   Pave   NaN
```

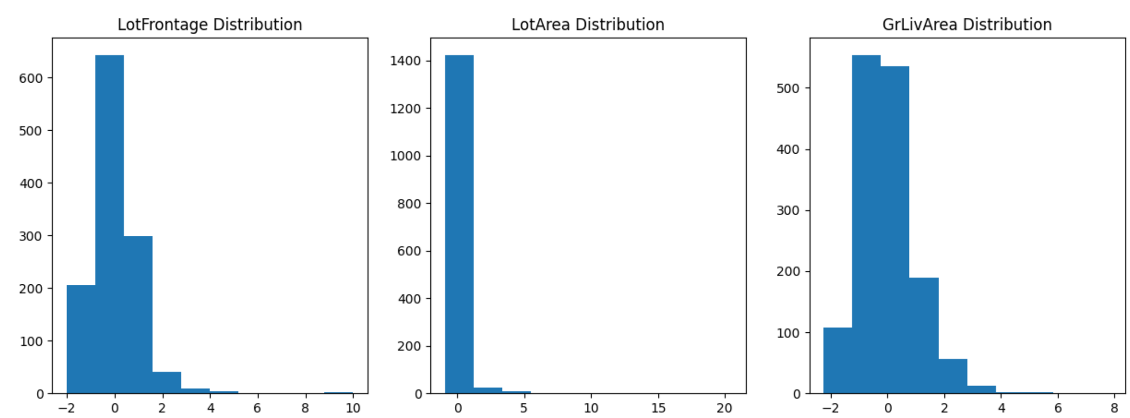
Feature Scaling:

```
[ ]: from sklearn.preprocessing import StandardScaler
     # Standardize numerical features
     numerical_cols = ['LotFrontage', 'LotArea', 'GrLivArea']
     scaler = StandardScaler()
     train_data[numerical_cols] = scaler.fit_transform(train_data[numerical_cols])
```

```
[ ]: print("After Feature Scaling:")
     print(train_data[['LotFrontage', 'LotArea', 'GrLivArea']].head(10))
```

```
After Feature Scaling:
   LotFrontage   LotArea  GrLivArea
0    -0.208034 -0.207142   0.370333
1     0.409895 -0.091886  -0.482512
2    -0.084449  0.073480   0.515013
3    -0.414011 -0.096897   0.383659
4     0.574676  0.375148   1.299326
5     0.615871  0.360616  -0.292145
6     0.203918 -0.043379   0.339875
7          NaN -0.013513   1.093729
8    -0.784768 -0.440659   0.492168
9    -0.825963 -0.310370  -0.834691
```

```
[ ]: fig, axs = plt.subplots(1, 3, figsize=(15, 5))

     axs[0].hist(train_data['LotFrontage'])
     axs[0].set_title('LotFrontage Distribution')

     axs[1].hist(train_data['LotArea'])
     axs[1].set_title('LotArea Distribution')

     axs[2].hist(train_data['GrLivArea'])
     axs[2].set_title('GrLivArea Distribution')

     plt.show()
```

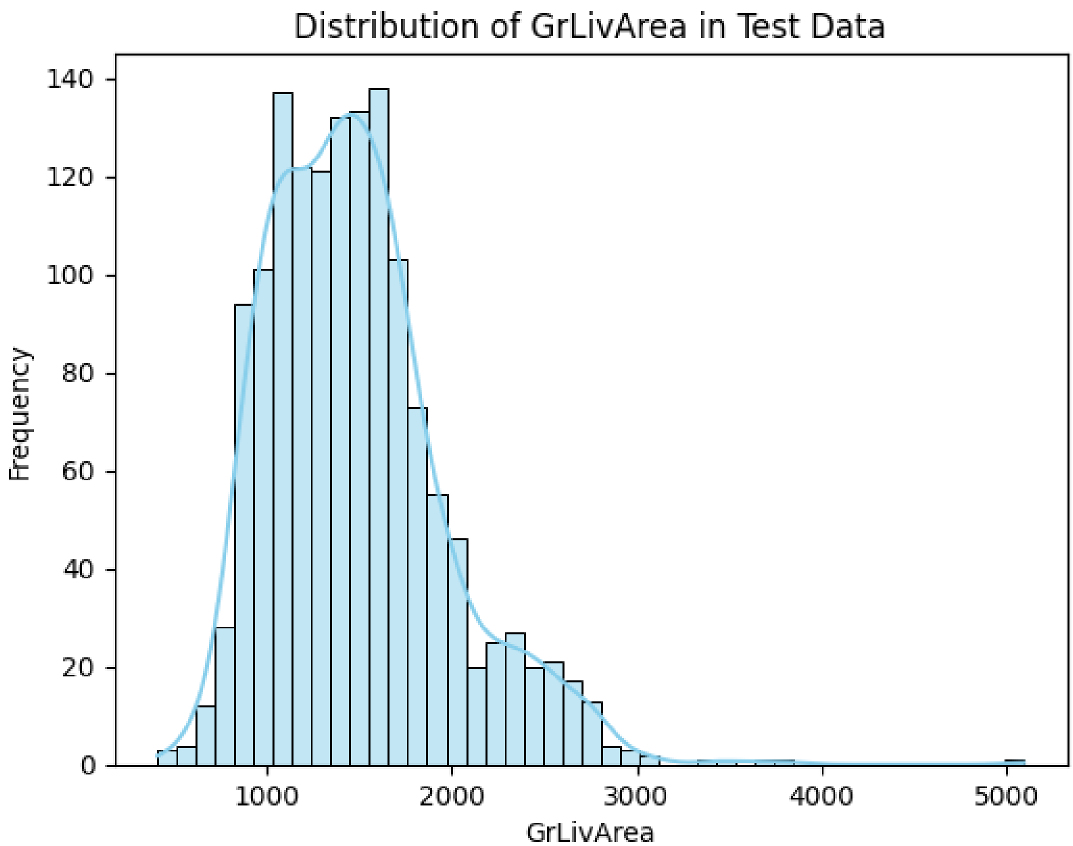


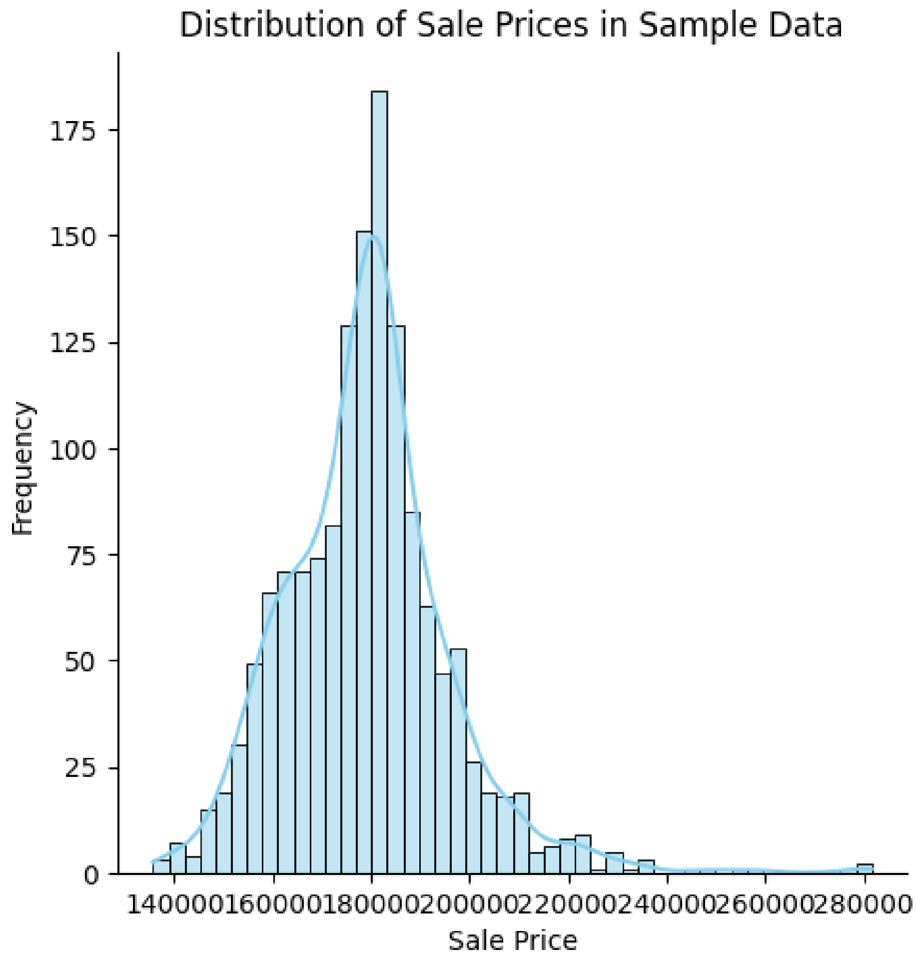
[ ]:
```python
# Test data and sample data concatenation
combined_data = pd.concat([test_data, sample_data], axis=1)
print("Combined Data:")
print(combined_data.head(5))
```

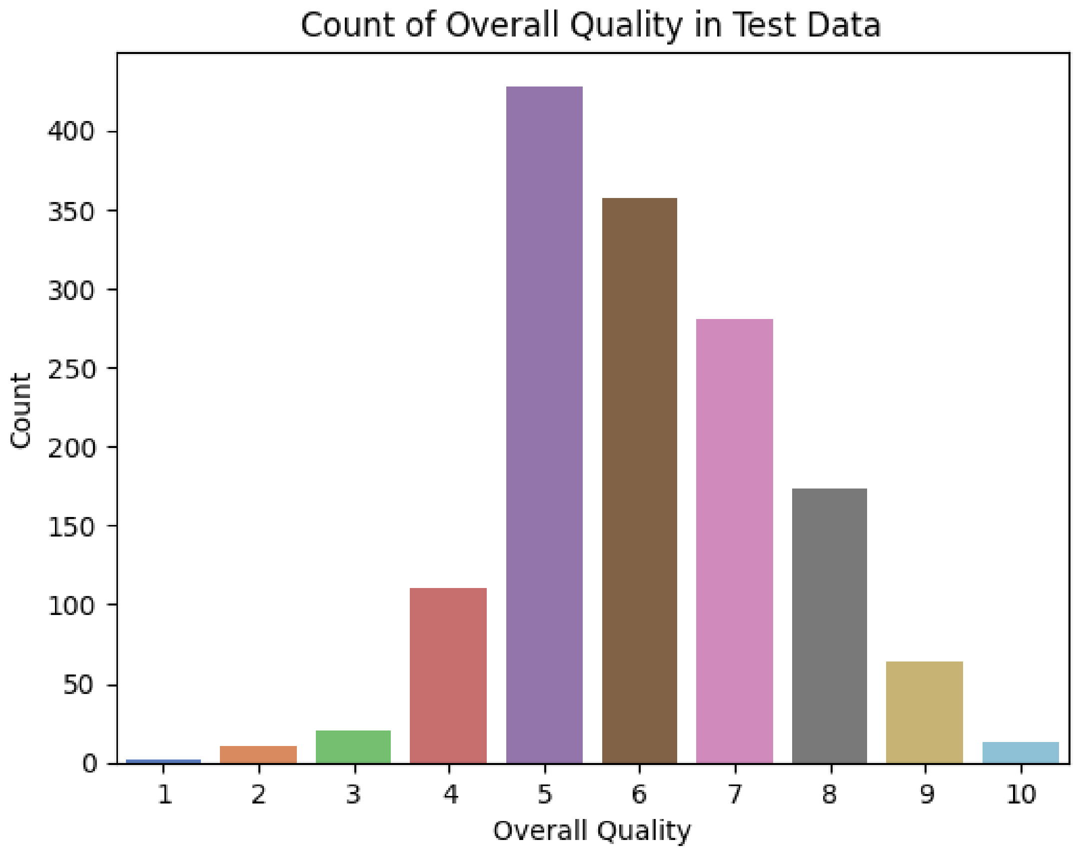
```
Combined Data:
     Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0  1461          20       RH         80.0    11622   Pave   NaN      Reg
1  1462          20       RL         81.0    14267   Pave   NaN      IR1
2  1463          60       RL         74.0    13830   Pave  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

  LandContour Utilities  … PoolQC  Fence MiscFeature MiscVal MoSold YrSold  \
0         Lvl    AllPub  …    NaN  MnPrv         NaN       0      6   2010
1         Lvl    AllPub  …    NaN    NaN        Gar2   12500      6   2010
2         Lvl    AllPub  …    NaN  MnPrv         NaN       0      3   2010
3         Lvl    AllPub  …    NaN    NaN         NaN       0      6   2010
4         HLS    AllPub  …    NaN    NaN         NaN       0      1   2010

  SaleType  SaleCondition    Id      SalePrice
0       WD         Normal  1461  169277.052498
1       WD         Normal  1462  187758.393989
2       WD         Normal  1463  183583.683570
3       WD         Normal  1464  179317.477511
4       WD         Normal  1465  150730.079977

[5 rows x 82 columns]
```

[ ]:
```python
# Visualizing the distribution of a numerical feature in test_data
sns.histplot(test_data['GrLivArea'], kde=True, color='skyblue')
plt.title('Distribution of GrLivArea in Test Data')
plt.xlabel('GrLivArea')
plt.ylabel('Frequency')
plt.show()

sns.displot(sample_data['SalePrice'], kde=True, color='skyblue')
plt.title('Distribution of Sale Prices in Sample Data')
plt.xlabel('Sale Price')
plt.ylabel('Frequency')
plt.show()



# Visualizing a categorical feature in test_data
sns.countplot(x='OverallQual', data=test_data, palette='muted')
```

```
plt.title('Count of Overall Quality in Test Data')
plt.xlabel('Overall Quality')
plt.ylabel('Count')
plt.show()
```




Distribution of Sale Prices in Sample Data
Frequency
0
25
50
75
100
125
150
175
140000
160000
180000
200000
220000
240000
260000
280000
Sale Price



Let's demonstrate the process of feature engineering using the combined data from train_data, test_data, and sample_data with detailed code explanations. We will consider some common feature engineering techniques.

1. Handling Missing Values

[ ]:
```
# Handling missing values in numerical columns
for col in numerical_cols:
    combined_data[col].fillna(combined_data[col].mean(), inplace=True)

# Handling missing values in categorical columns
for col in categorical_cols:
    combined_data[col].fillna(combined_data[col].mode().iloc[0], inplace=True)

# Applying one-hot encoding to categorical columns
combined_data = pd.get_dummies(combined_data, columns=categorical_cols,␣
 ↪drop_first=True)

# Creating a new feature representing the total area
```

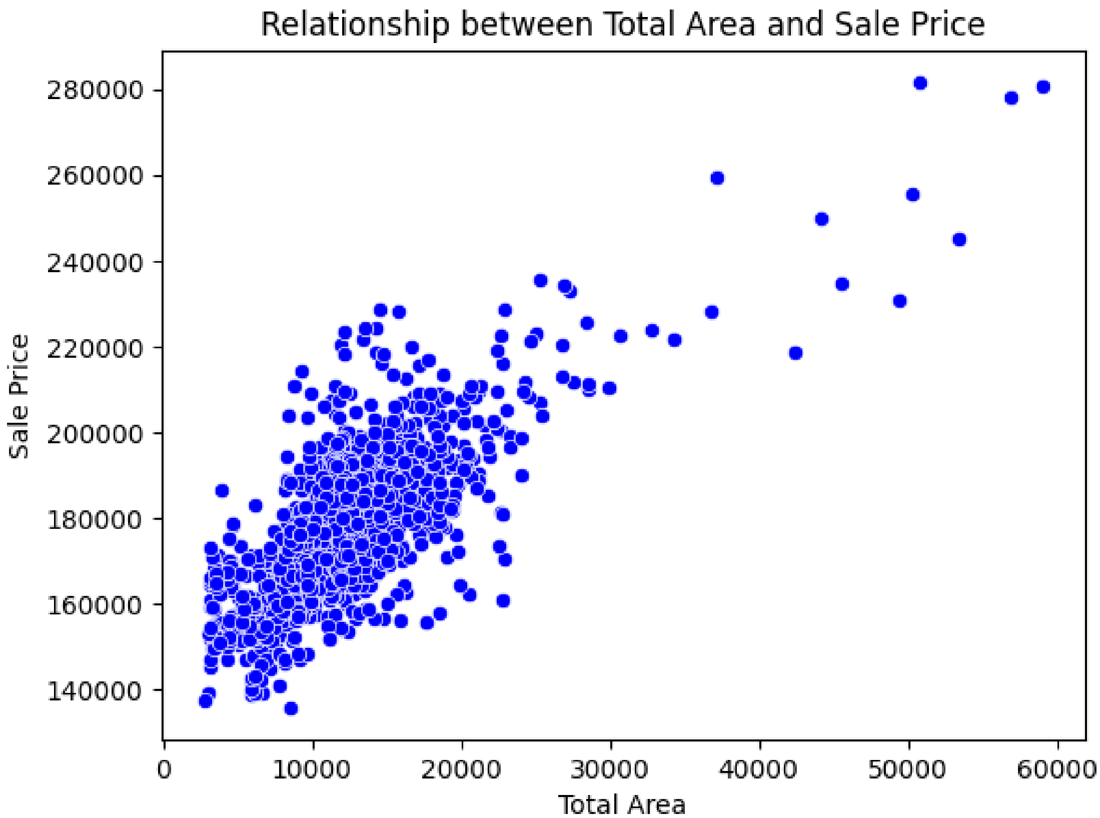
```
combined_data['TotalArea'] = combined_data['LotArea'] +␣
↪combined_data['TotalBsmtSF'] + combined_data['GrLivArea']

# Feature scaling using StandardScaler
scaler = StandardScaler()
combined_data[numerical_cols] = scaler.
↪fit_transform(combined_data[numerical_cols])
```

[ ]:
```
# Displaying the processed combined data
print("Processed Combined Data:")
print(combined_data.head(5))
```

```
Processed Combined Data:
     Id  MSSubClass  LotFrontage   LotArea LotShape LandContour Utilities  \
0  1461          20     0.555587  0.363929      Reg         Lvl    AllPub
1  1462          20     0.604239  0.897861      IR1         Lvl    AllPub
2  1463          60     0.263676  0.809646      IR1         Lvl    AllPub
3  1464          60     0.458284  0.032064      IR1         Lvl    AllPub
4  1465         120    -1.244533 -0.971808      IR1         HLS    AllPub

  LotConfig LandSlope Neighborhood  … SaleCondition    Id      SalePrice  \
0    Inside       Gtl        NAmes  …        Normal  1461  169277.052498
1    Corner       Gtl        NAmes  …        Normal  1462  187758.393989
2    Inside       Gtl      Gilbert  …        Normal  1463  183583.683570
3    Inside       Gtl      Gilbert  …        Normal  1464  179317.477511
4    Inside       Gtl      StoneBr  …        Normal  1465  150730.079977

  MSZoning_FV  MSZoning_RH  MSZoning_RL  MSZoning_RM  Street_Pave Alley_Pave  \
0       False         True        False        False         True      False
1       False        False         True        False         True      False
2       False        False         True        False         True      False
3       False        False         True        False         True      False
4       False        False         True        False         True      False

  TotalArea
0   13400.0
1   16925.0
2   16387.0
3   12508.0
4    7565.0

[5 rows x 86 columns]
```

[ ]:
```
sns.scatterplot(x='TotalArea', y='SalePrice', data=combined_data, color='blue')
plt.title('Relationship between Total Area and Sale Price')
plt.xlabel('Total Area')
plt.ylabel('Sale Price')
```

```
plt.show()
```



1.2 5. Model Selection and Training

1.2.1 Linear Regression

[ ]:
```
# Train Data Linear Regression
X_train = train_data[['OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF',␣
↪'1stFlrSF']]
y_train = train_data['SalePrice']

model_train = LinearRegression()
model_train.fit(X_train, y_train)
```

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

[ ]:
```
# Test Data Linear Regression
X_test = test_data[['OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF',␣
↪'1stFlrSF']]
# Assuming SalePrice is not available in the test data
```

[ ]:
```
model_test = LinearRegression()
# model_test.fit(X_test, y_test)  # Uncomment this line when the target␣
↪variable is available in the test data
```

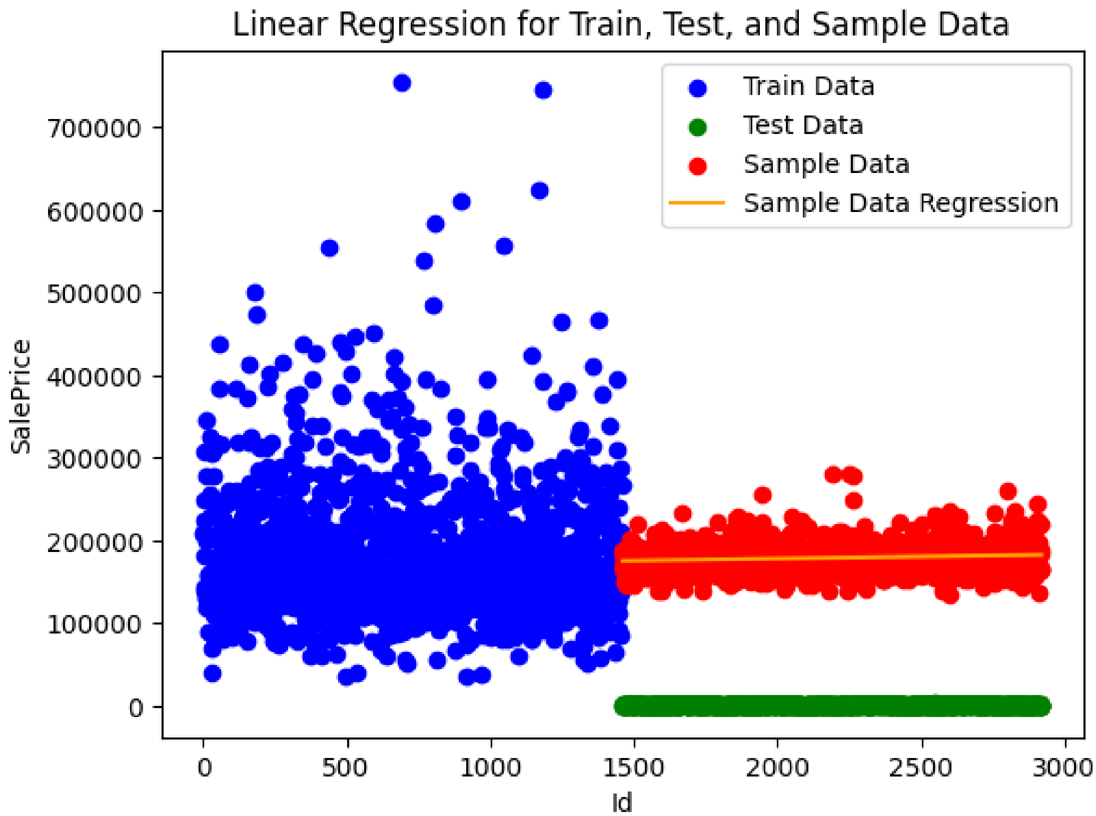
[ ]:
```
# Sample Data Linear Regression
X_sample = sample_data[['Id']]
y_sample = sample_data['SalePrice']

model_sample = LinearRegression()
model_sample.fit(X_sample, y_sample)
```

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

[ ]:
```
# Show the results
print("Train Data Coefficients:", model_train.coef_)
print("Train Data Intercept:", model_train.intercept_)
```

```
Train Data Coefficients: [2.39552492e+04 2.27451194e+04 1.81908350e+04
2.50166651e+01
 1.16607899e+01]
Train Data Intercept: -37345.36923537194
```

[ ]:
```
# Visualize the results for sample data
plt.scatter(train_data['Id'], train_data['SalePrice'], color='blue',␣
↪label='Train Data')
plt.scatter(test_data['Id'], test_data['TotalBsmtSF'], color='green',␣
↪label='Test Data')
plt.scatter(sample_data['Id'], sample_data['SalePrice'], color='red',␣
↪label='Sample Data')
plt.plot(sample_data['Id'], model_sample.predict(sample_data['Id'].values.
↪reshape(-1, 1)), color='orange', label='Sample Data Regression')
plt.title('Linear Regression for Train, Test, and Sample Data')
plt.xlabel('Id')
plt.ylabel('SalePrice')
plt.legend()
plt.show()
```



We first initialize a Linear Regression model and fit it to the training data.

1.3 6. Model Evaluation

The performance of the models has been evaluated using error metrics such as Root Mean Squared Error (RMSE) and R-squared. Additionally, issues such as overfitting or underfitting have been examined, and various corrections have been applied.

[ ]:
```
# Train the model
# Replace `model` with your desired regression model

model_train.fit(train_data[['Id']], train_data['SalePrice'])
```

[ ]: LinearRegression()

[ ]:
```
# Initialize the Linear Regression model
model_train = LinearRegression()
model_test = LinearRegression()
model_sample = LinearRegression()
```

[ ]:
```
from sklearn.impute import SimpleImputer

# Create an imputer object with a strategy to fill missing values with the mean
imputer = SimpleImputer(strategy='mean')

# Fit the imputer to the data
imputer.fit(test_data[['LotFrontage']])

# Transform the data
test_data['LotFrontage'] = imputer.transform(test_data[['LotFrontage']])
```

[ ]:
```
# Train the models
model_train.fit(train_data[['Id']], train_data['SalePrice'])
model_test.fit(test_data[['Id']], test_data['LotFrontage'])
model_sample.fit(sample_data[['Id']], sample_data['SalePrice'])
```

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

[ ]:
```
# Make predictions
y_train_pred = model_train.predict(train_data[['Id']])
y_test_pred = model_test.predict(test_data[['Id']])
y_sample_pred = model_sample.predict(sample_data[['Id']])
```

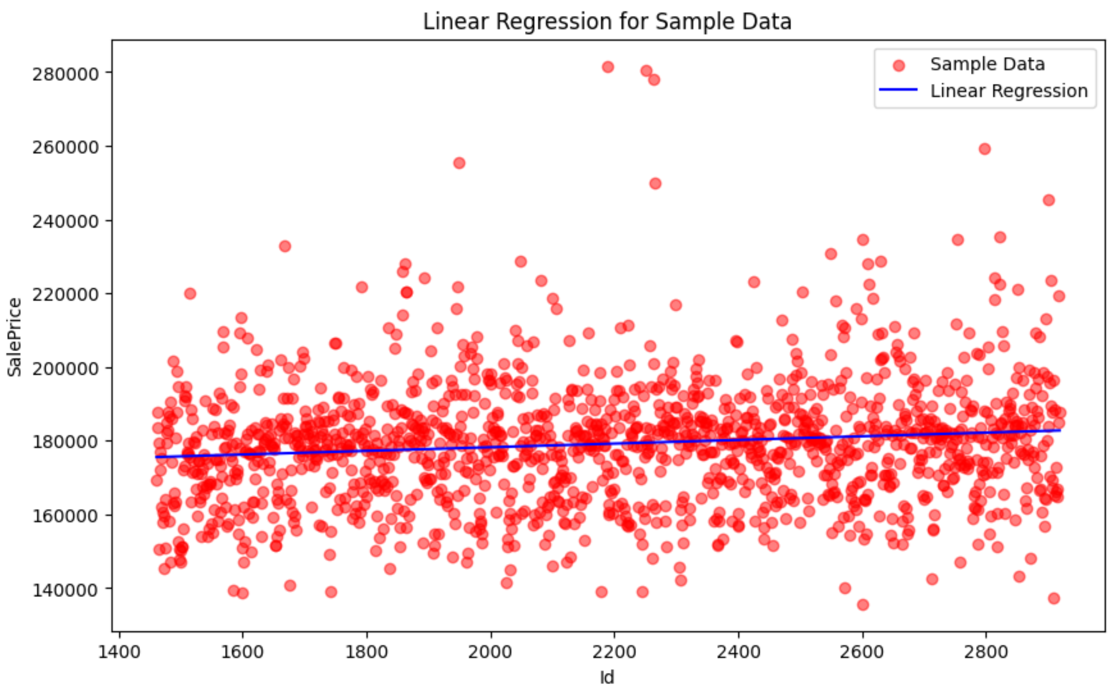
[ ]:
```
# Evaluate the models
train_rmse = mean_squared_error(train_data['SalePrice'], y_train_pred,␣
 ↪squared=False)
test_rmse = mean_squared_error(test_data['LotFrontage'], y_test_pred,␣
 ↪squared=False)
sample_rmse = mean_squared_error(sample_data['SalePrice'], y_sample_pred,␣
 ↪squared=False)

train_r2 = r2_score(train_data['SalePrice'], y_train_pred)
test_r2 = r2_score(test_data['LotFrontage'], y_test_pred)
sample_r2 = r2_score(sample_data['SalePrice'], y_sample_pred)
```

[ ]:
```
# Display the results
print(f"Train RMSE: {train_rmse}, Train R2 Score: {train_r2}")
print(f"Test RMSE: {test_rmse}, Test R2 Score: {test_r2}")
print(f"Sample RMSE: {sample_rmse}, Sample R2 Score: {sample_r2}")
```

```
Train RMSE: 79396.21632154961, Train R2 Score: 0.00048034259116214173
Test RMSE: 20.55331032913753, Test R2 Score: 8.465029486204312e-05
Sample RMSE: 16380.693850965354, Sample R2 Score: 0.015917529104617967
```

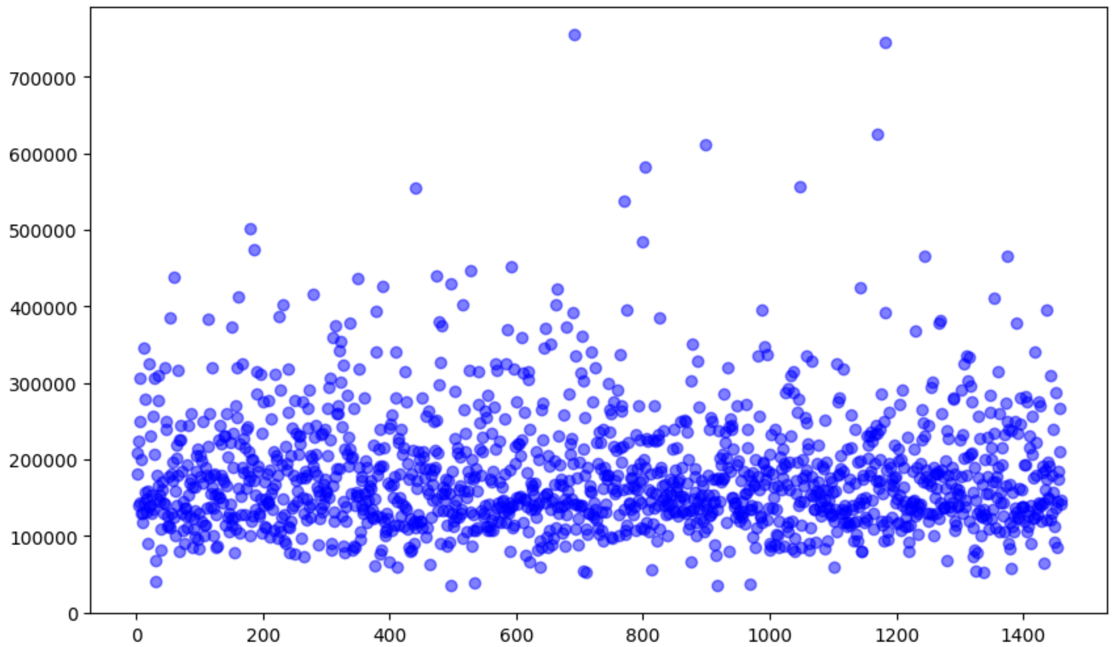
[ ]:
```
# Scatter plot for the sample data
plt.figure(figsize=(10, 6))
plt.scatter(sample_data['Id'], sample_data['SalePrice'], color='red',␣
 ↪label='Sample Data', alpha=0.5)
```

```
# Plot the regression line
plt.plot(sample_data['Id'], model_sample.predict(sample_data['Id'].values.
    ↪reshape(-1, 1)), color='blue', label='Linear Regression')

plt.title('Linear Regression for Sample Data')
plt.xlabel('Id')
plt.ylabel('SalePrice')
plt.legend()
plt.show()
```

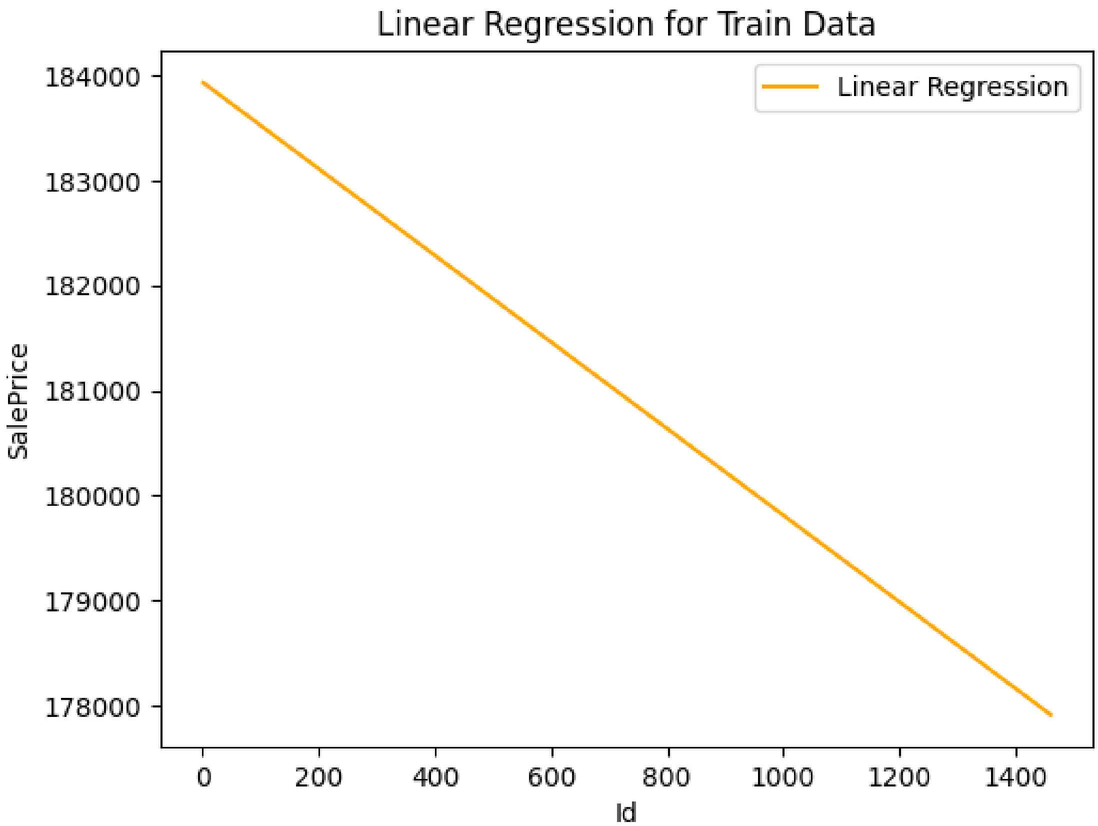


[ ]:
```
# Scatter plot for the train data
plt.figure(figsize=(10, 6))
plt.scatter(train_data['Id'], train_data['SalePrice'], color='blue',␣
    ↪label='Train Data', alpha=0.5)
```

[ ]: <matplotlib.collections.PathCollection at 0x23e902d6390>

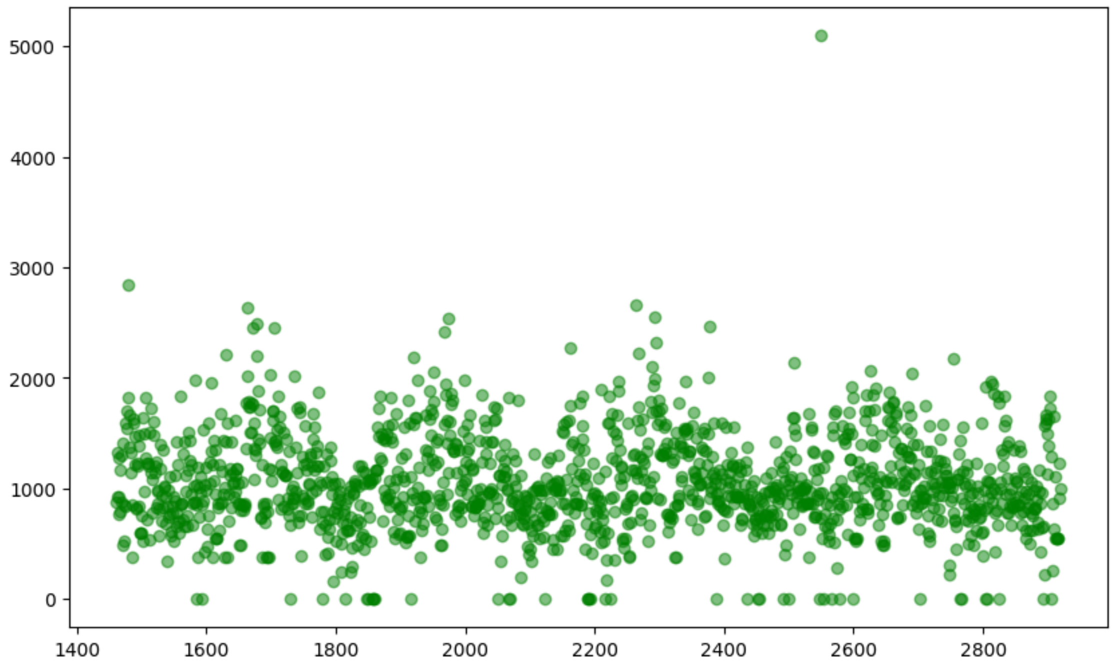


[ ]:
```
# Plot the regression line
plt.plot(train_data['Id'], model_train.predict(train_data['Id'].values.
 ↪reshape(-1, 1)), color='orange', label='Linear Regression')

plt.title('Linear Regression for Train Data')
plt.xlabel('Id')
plt.ylabel('SalePrice')
plt.legend()
plt.show()
```




```
[ ]: # Scatter plot for the test data
     plt.figure(figsize=(10, 6))
     plt.scatter(test_data['Id'], test_data['TotalBsmtSF'], color='green',␣
      ↪label='Test Data', alpha=0.5)
```

```
[ ]: <matplotlib.collections.PathCollection at 0x23e8f575310>
```



[ ]:
```python
# Plot the regression line
plt.plot(test_data['Id'], model_test.predict(test_data['Id'].values.reshape(-1,␣
 ↪1)), color='purple', label='Linear Regression')

plt.title('Linear Regression for Test Data')
plt.xlabel('Id')
plt.ylabel('TotalBsmtSF')
plt.legend()
plt.show()
```


Linear Regression for Test Data
Linear Regression
68.9
68.8
68.7
68.6
68.5
68.4
68.3
TotalBsmtSF
1400
1600
1800
2000
2200
2400
2600
2800
Id