House Price Prediction Analysis

Research Question: What are the key features that drive house prices in Ames, Iowa, and how can we use these features to build a reliable predictive model for house prices?

1. Exploratory Data Analysis (EDA)

(1) Data Cleaning

A. Missing Values Handling

- For features such as `PoolQC`, `MiscFeature`, and `Alley`, missing values were interpreted as the absence of these features. They were replaced with the string `"None"` to explicitly indicate the lack of these characteristics.
- Missing values in features like `GarageArea`, `BsmtFinSF1`, and `TotalBsmtSF` represented the absence of these amenities. These were filled with `0` to signify that the property lacked these attributes.
- For `LotFrontage` (street frontage), missing values were filled using the median value for the respective `Neighborhood`. This ensured the imputed values aligned with local trends.
- Missing values in the `Electrical` column were replaced by the mode, under the assumption that the most common electrical system was installed in properties with missing data.

B. Outlier Handling

- Outliers were identified using a scatter plot of `GrLivArea` versus `SalePrice`. Two large houses with unusually low prices were highlighted during this analysis.
- Upon further investigation, both properties were located in the Edwards neighborhood and classified as `Partial Built`.
- To confirm whether these properties were true outliers, the `SalePrice` statistics for the Edwards neighborhood were compared to the overall dataset. While Edwards had a generally lower price range, these two properties were priced above the 75th percentile for the neighborhood.
- Conclusion: These houses were not outliers and were retained in the dataset.

2. Feature Engineering

(1) Feature Creation:

- New features, `TotalBath`, were created to represent distinct property attributes.

(2) Polynomial Features:

A. Why `GrLivArea` and `TotalBsmtSF`?

- Both features showed strong correlations with `SalePrice` and are key indicators of property size and value.
- A linear relationship might not fully capture the pricing behavior for very large properties.

B. What features were created?

- `GrLivArea^2`: Captures the non-linear increase in value associated with larger living areas.
- `TotalBsmtSF^2`: Reflects how a larger basement area impacts property prices in non-linear.
- `GrLivArea*TotalBsmtSF`: Captures the interaction between above-ground living space and basement size.

(3) Handling Multicollinearity:

- Variance Inflation Factor (VIF) was used to identify multicollinear features.
- The following strategies were employed to manage multicollinearity:

Removing features that were highly correlated with others (e.g., 1stFlrSF strongly correlated with GrLivArea).

Post-removal, the VIF values for all retained features were below the threshold of 10, indicating acceptable multicollinearity levels.

3. Feature Selection

(1) Numerical Features:

- Correlation analysis identified features with a correlation coefficient greater than 0.3 with `SalePrice`.
- An F-test further evaluated the statistical significance of these features, retaining those with p-values below 0.05.

(2) Categorical Features:

- Target encoding was applied to compute the correlation between each categorical feature and `SalePrice`, and selected features with a correlation coefficient greater than 0.3.
- ANOVA tests assessed the statistical significance of each categorical variable, retaining those with p-values below 0.05. Features such as `Neighborhood`, `ExterQual`, and `KitchenQual` were retained, as they exhibited strong relationships with the target variable.

4. Regression Models

(1) Linear Regression:

- Baseline model with standardized features.
- MSE: `6.8568e+27`, R^2: `-8.939`, indicating severe overfitting or numerical instability.

(2) OLS Regression:

- K-Fold Cross-Validation:
  - A 20-fold cross-validation was performed to assess model performance; Mean RMSE: 0.1352.

In [2]:
```
# Import libraries
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,KFold
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.preprocessing import MinMaxScaler, StandardScaler, PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from scipy.stats import f_oneway
from sklearn.feature_selection import f_regression
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# Load the dataset
train = pd.read_csv('/Users/zionmicwu/Desktop/train.csv')
test = pd.read_csv('/Users/zionmicwu/Desktop/test.csv')
test_id = test.copy()
```

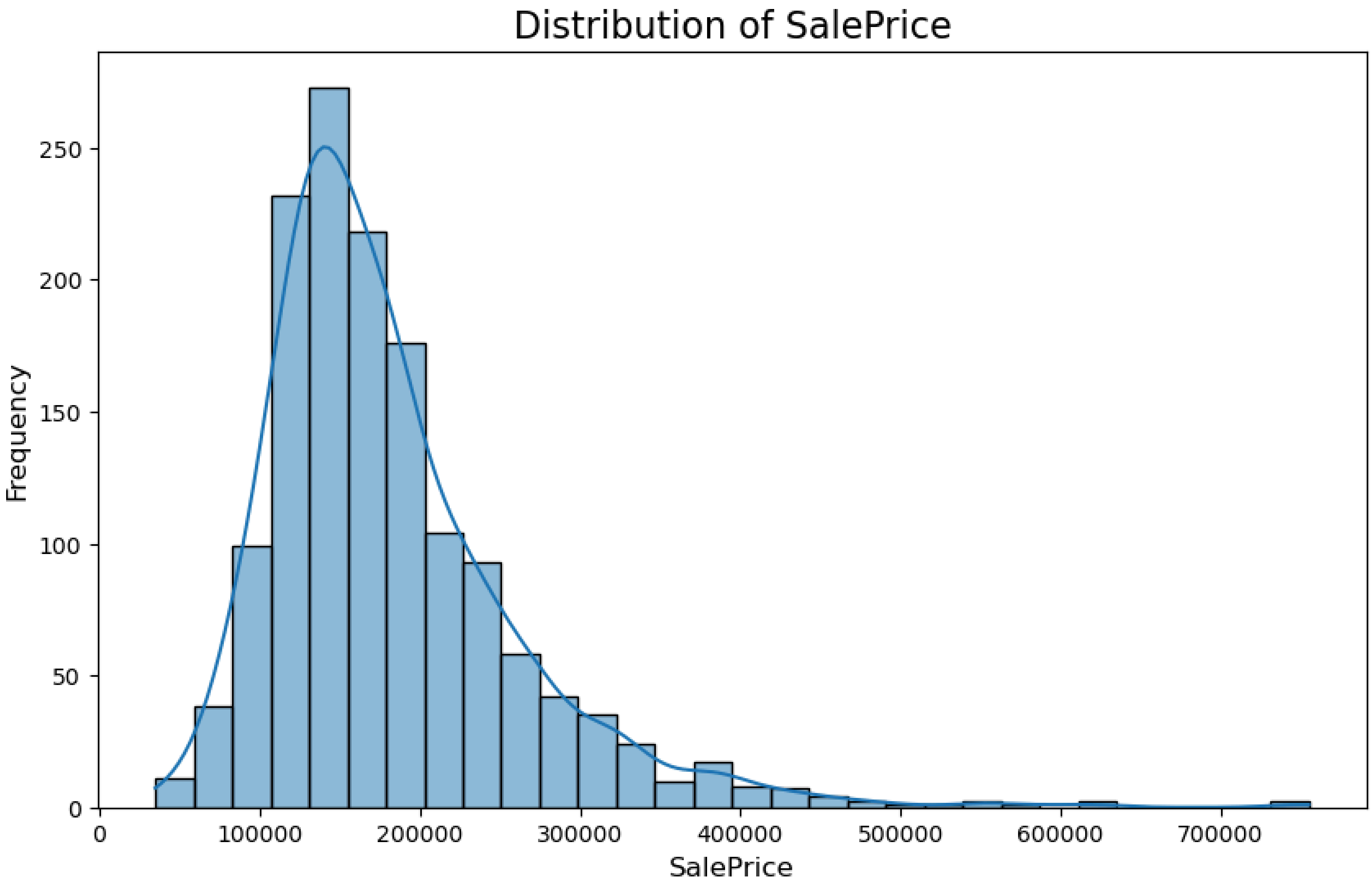
In [4]:
```
# Analyze missing values
mis_val = train.isnull().sum()
mis_val_percent = 100 * mis_val / len(train)
mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
mis_val_table.columns = ["Missing Values", "% of Total Values"]
print(mis_val_table[mis_val_table["Missing Values"] > 0].sort_values("% of Total Values", ascending=False))
```

```
              Missing Values  % of Total Values
PoolQC                  1453          99.520548
MiscFeature             1406          96.301370
Alley                   1369          93.767123
Fence                   1179          80.753425
MasVnrType               872          59.726027
FireplaceQu              690          47.260274
LotFrontage              259          17.739726
GarageType                81           5.547945
GarageYrBlt               81           5.547945
GarageFinish              81           5.547945
GarageQual                81           5.547945
GarageCond                81           5.547945
BsmtFinType2              38           2.602740
BsmtExposure              38           2.602740
BsmtFinType1              37           2.534247
BsmtCond                  37           2.534247
BsmtQual                  37           2.534247
MasVnrArea                 8           0.547945
Electrical                 1           0.068493
```

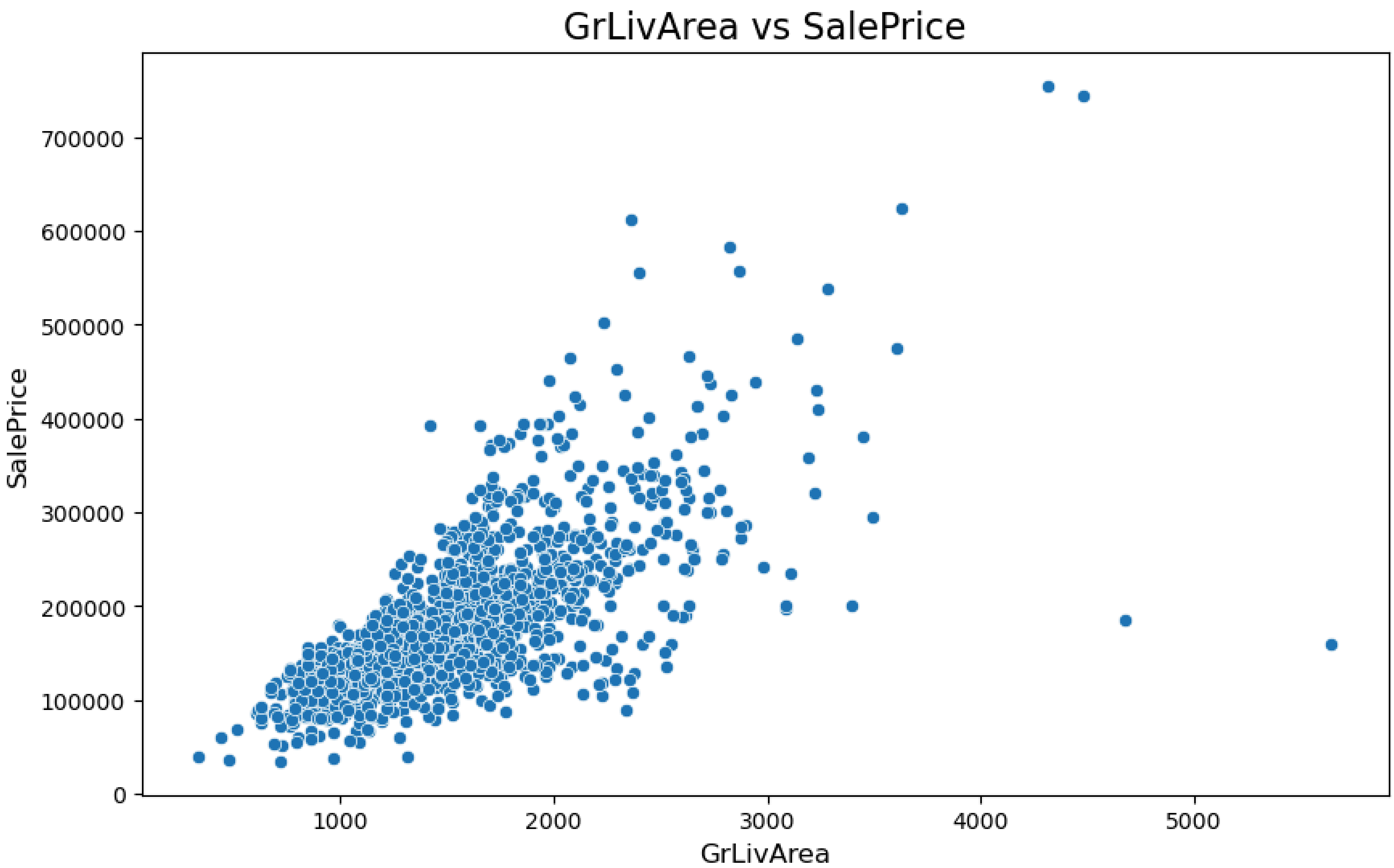
In [5]:
```
# Handle missing values
# Fill categorical columns with "None"
for col in ["PoolQC", "MiscFeature", "Alley", "Fence", "FireplaceQu", "GarageType", "GarageFinish", "GarageQual", "GarageCo
    train[col].fillna("None", inplace=True)
    test[col].fillna("None", inplace=True)

# Fill numerical columns logically
for col in ["GarageYrBlt", "GarageCars", "GarageArea", "BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF", "TotalBsmtSF", "MasVnrArea"
    train[col].fillna(0, inplace=True)
    test[col].fillna(0, inplace=True)

# Fill LotFrontage with median based on Neighborhood
train["LotFrontage"] = train.groupby("Neighborhood")["LotFrontage"].transform(lambda x: x.fillna(x.median()))
test["LotFrontage"] = test.groupby("Neighborhood")["LotFrontage"].transform(lambda x: x.fillna(x.median()))

# Fill Electrical with mode
train["Electrical"].fillna(train["Electrical"].mode()[0], inplace=True)

# Verify remaining missing values
print("\nRemaining missing values in training data:")
print(train.isnull().sum().sum())
print("\nRemaining missing values in testing data:")
print(test.isnull().sum().sum())
```

```
Remaining missing values in training data:
0

Remaining missing values in testing data:
16
```

In [6]:
```
# Visualization: SalePrice distribution
plt.figure(figsize=(10, 6))
sns.histplot(train['SalePrice'], kde=True, bins=30)
plt.title("Distribution of SalePrice", fontsize=16)
plt.xlabel("SalePrice", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.show()
```



In [7]:
```python
# Investigate outliers in GrLivArea vs. SalePrice
plt.figure(figsize=(10, 6))
sns.scatterplot(x=train['GrLivArea'], y=train['SalePrice'])
plt.title("GrLivArea vs SalePrice", fontsize=16)
plt.xlabel("GrLivArea", fontsize=12)
plt.ylabel("SalePrice", fontsize=12)
plt.show()
```



In [8]:
```python
# Identify and explore large houses with low prices
large_low_price = train[(train['GrLivArea'] > 4000) & (train['SalePrice'] < 300000)]
print("\nLarge houses with low prices:\n", large_low_price)
```

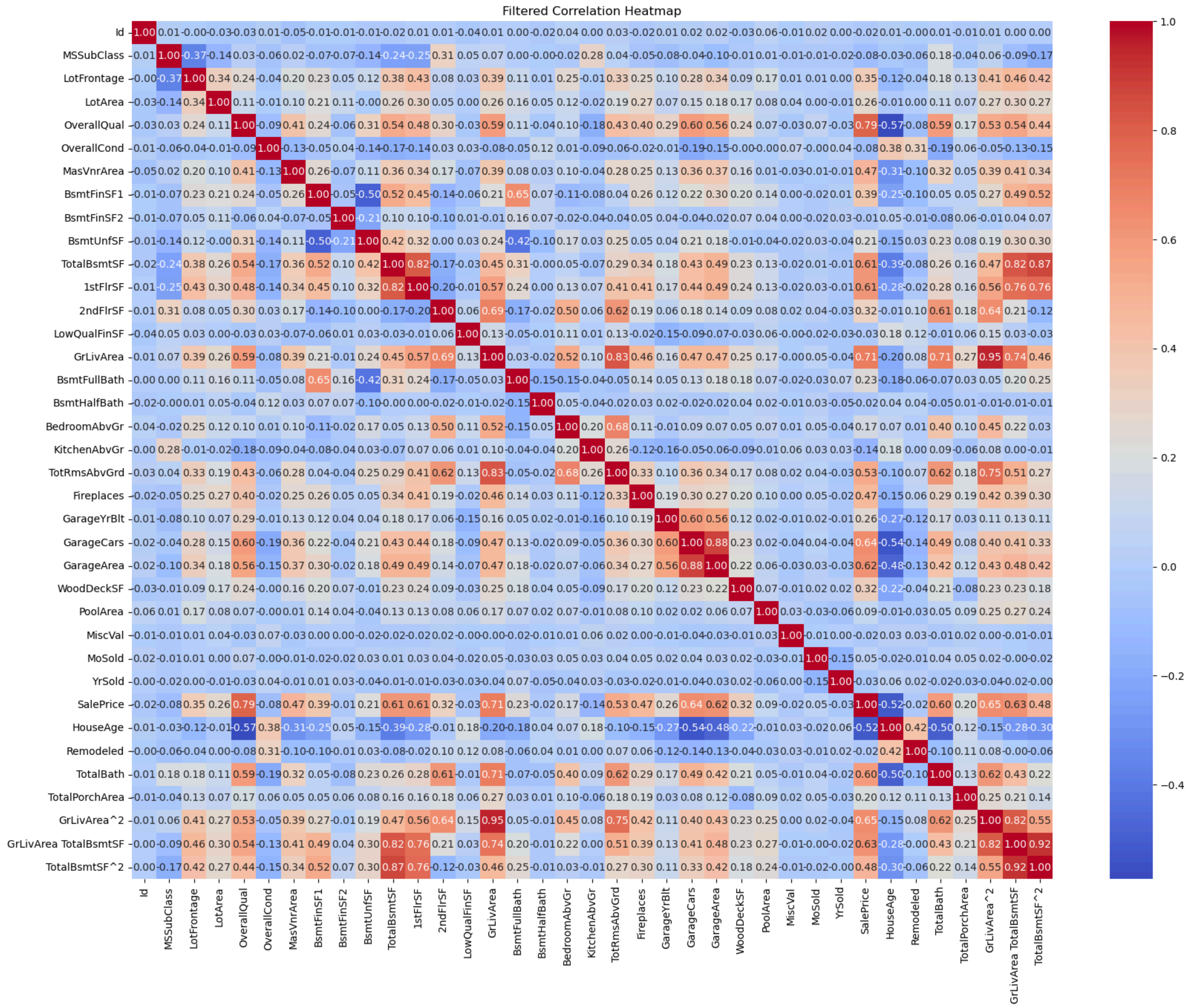
```
Large houses with low prices:
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
523    524          60       RL        130.0    40094   Pave  None      IR1
1298  1299          60       RL        313.0    63887   Pave  None      IR3

     LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal  \
523          Bnk    AllPub  ...        0   None  None        None       0
1298         Bnk    AllPub  ...      480     Gd  None        None       0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
523      10   2007       New        Partial     184750
1298      1   2008       New        Partial     160000

[2 rows x 81 columns]
```

In [9]:
```
# Compare Edwards neighborhood vs overall prices
edwards_prices = train[train['Neighborhood'] == 'Edwards']['SalePrice']
print("\nEdwards Neighborhood Price Statistics:")
print(edwards_prices.describe())
print("\nOverall Price Statistics:")
print(train['SalePrice'].describe())
```

```
Edwards Neighborhood Price Statistics:
count       100.000000
mean     128219.700000
std       43208.616459
min       58500.000000
25%      101500.000000
50%      121750.000000
75%      145225.000000
max      320000.000000
Name: SalePrice, dtype: float64

Overall Price Statistics:
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

In [10]:
```
# Add a new feature for house age
train['HouseAge'] = train['YrSold'] - train['YearBuilt']
test['HouseAge'] = test['YrSold'] - test['YearBuilt']

# Add HasFireplace feature
train['HasFireplace'] = (train['Fireplaces'] > 0).astype(bool)
test['HasFireplace'] = (test['Fireplaces'] > 0).astype(bool)

# Add Remodeled feature
train['Remodeled'] = (train['YearBuilt'] != train['YearRemodAdd']).astype(int)
test['Remodeled'] = (test['YearBuilt'] != test['YearRemodAdd']).astype(int)

# Add TotalBath feature
train['TotalBath'] = train['FullBath'] + 0.5 * train['HalfBath']
test['TotalBath'] = test['FullBath'] + 0.5 * test['HalfBath']

# Add TotalPorchArea feature
train['TotalPorchArea'] = train['OpenPorchSF'] + train['EnclosedPorch'] + train['3SsnPorch'] + train['ScreenPorch']
test['TotalPorchArea'] = test['OpenPorchSF'] + test['EnclosedPorch'] + test['3SsnPorch'] + test['ScreenPorch']

# Polynomial features
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features_train = poly.fit_transform(train[['GrLivArea', 'TotalBsmtSF']])
poly_features_test = poly.transform(test[['GrLivArea', 'TotalBsmtSF']])
poly_train_columns = poly.get_feature_names_out(['GrLivArea', 'TotalBsmtSF'])
poly_train_df = pd.DataFrame(poly_features_train, columns=poly_train_columns, index=train.index)
poly_train_df.drop(columns = ['GrLivArea', 'TotalBsmtSF'], inplace = True)
poly_test_df = pd.DataFrame(poly_features_test, columns=poly_train_columns, index=test.index)
poly_test_df.drop(columns = ['GrLivArea', 'TotalBsmtSF'], inplace = True)
train = pd.concat([train, poly_train_df], axis=1)
test = pd.concat([test, poly_test_df], axis=1)

# Drop used features
used_features = ["YearBuilt", "YearRemodAdd", "FullBath", "HalfBath", "OpenPorchSF", "EnclosedPorch", "3SsnPorch", "ScreenP
train.drop(columns=used_features, inplace=True)
test.drop(columns=used_features, inplace=True)
```

In [11]:
```
# Filter numeric columns for correlation matrix
numeric_columns = train.select_dtypes(include=['float64', 'int64']).columns
correlation_matrix = train[numeric_columns].corr()

# Plot filtered heatmap
plt.figure(figsize=(20, 15))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Filtered Correlation Heatmap")
plt.show()
```



In [12]:

```python
# Calculate Variance Inflation Factor (VIF)
train_vif_copy = train[numeric_columns].copy()
train_vif_copy.drop(columns=['SalePrice'], inplace = True)
vif_data = pd.DataFrame()
vif_data['Feature'] = train_vif_copy.columns
vif_data['VIF'] = [variance_inflation_factor(train_vif_copy.values, i) for i in range(train_vif_copy.shape[1])]

print("VIF for each feature:")
print(vif_data)

# Shoe features with VIF > 10
high_vif_features = (vif_data[vif_data['VIF'] > 10]['Feature']).tolist()
print(high_vif_features)
```

```
VIF for each feature:
                Feature         VIF
0                    Id    4.098068
1            MSSubClass    4.820569
2           LotFrontage   18.449923
3               LotArea    2.678239
4           OverallQual   66.714354
5           OverallCond   35.746939
6            MasVnrArea    1.844395
7            BsmtFinSF1         inf
8            BsmtFinSF2         inf
9             BsmtUnfSF         inf
10          TotalBsmtSF         inf
11             1stFlrSF         inf
12             2ndFlrSF         inf
13         LowQualFinSF         inf
14            GrLivArea         inf
15         BsmtFullBath    3.831731
16         BsmtHalfBath    1.228016
17         BedroomAbvGr   31.397616
18         KitchenAbvGr   38.392844
19         TotRmsAbvGrd   84.093383
20           Fireplaces    2.944703
21          GarageYrBlt   32.528709
22           GarageCars   40.461190
23           GarageArea   31.564667
24           WoodDeckSF    1.892920
25             PoolArea    1.194393
26              MiscVal    1.031621
27               MoSold    6.633242
28               YrSold  164.574369
29             HouseAge    8.360376
30            Remodeled    2.684592
31            TotalBath   33.135091
32       TotalPorchArea    2.092765
33          GrLivArea^2  140.507263
34  GrLivArea TotalBsmtSF  266.665171
35        TotalBsmtSF^2   72.793493
['LotFrontage', 'OverallQual', 'OverallCond', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrS
F', 'LowQualFinSF', 'GrLivArea', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'GarageYrBlt', 'GarageCars', 'GarageArea',
'YrSold', 'TotalBath', 'GrLivArea^2', 'GrLivArea TotalBsmtSF', 'TotalBsmtSF^2']
```

In [13]:

```python
# Drop features with high collinearity
selected_high_vif_features = ['LotFrontage','BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF'
train = train.drop(columns = selected_high_vif_features, errors = 'ignore')
test = test.drop(columns = selected_high_vif_features, errors = 'ignore')
```

In [14]:

```python
# reFilter numeric columns for correlation matrix
numeric_columns = train.select_dtypes(include=['float64', 'int64']).columns
correlation_matrix = train[numeric_columns].corr()

# Filter only features with high correlation with SalePrice
correlation_threshold = 0.3
high_correlation_vars = correlation_matrix['SalePrice'][abs(correlation_matrix['SalePrice']) > correlation_threshold].index

# F-test for numerical feature selection
f_scores, p_values = f_regression(train[high_correlation_vars.drop('SalePrice')], train['SalePrice'])
f_test_results = pd.DataFrame({'Feature': high_correlation_vars.drop('SalePrice'), 'F-Score': f_scores, 'P-Value': p_values
selected_num_features = f_test_results[f_test_results['P-Value'] < 0.05]['Feature'].tolist()
selected_num_features.append('SalePrice')

# Display selected features
print("\nSelected numerical features for regression:", selected_num_features)
```

```
Selected numerical features for regression: ['OverallQual', 'MasVnrArea', 'TotalBsmtSF', 'GrLivArea', 'TotRmsAbvGrd', 'Firep
laces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'HouseAge', 'TotalBath', 'GrLivArea^2', 'GrLivArea TotalBsmtSF', 'TotalBsm
tSF^2', 'SalePrice']
```

In [15]:

```python
# Handle categorical variables
categorical_columns = train.select_dtypes(include=['object', 'category']).columns
print("Categorical Features:", len(categorical_columns))

train_copy = train.copy()

# Target encoding for correlation analysis
for col in categorical_columns:
    train_copy[f"{col}_encoded"] = train_copy.groupby(col)['SalePrice'].transform('mean')

# Correlation analysis
correlation = train_copy[[f"{col}_encoded" for col in categorical_columns] + ['SalePrice']].corr()
print("Correlation with SalePrice:")
print(correlation['SalePrice'].sort_values(ascending=False))

# ANOVA for significance testing
for col in categorical_columns:
    groups = [group['SalePrice'].values for name, group in train.groupby(col)]
    anova_result = f_oneway(*groups)
    print(f"{col}: p-value = {anova_result.pvalue}")

# Filter based on correlation and p-value thresholds
selected_cat_features = [
```

```python
    col for col in categorical_columns
    if correlation[f"{col}_encoded"]["SalePrice"] > 0.3
    and f_oneway(*[group['SalePrice'].values for name, group in train.groupby(col)]).pvalue < 0.05
]
print("Important categorical features:", selected_cat_features)
```

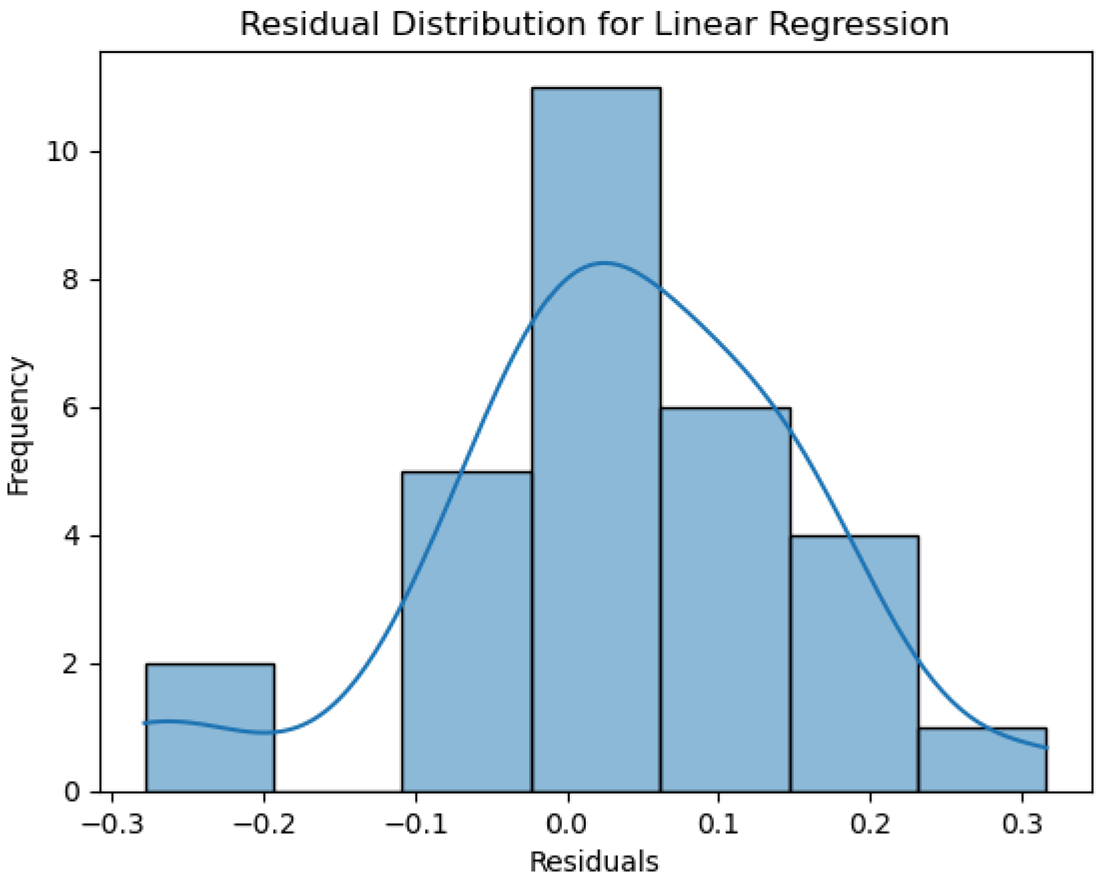
```
Categorical Features: 43
Correlation with SalePrice:
SalePrice                1.000000
Neighborhood_encoded     0.738630
ExterQual_encoded        0.690933
BsmtQual_encoded         0.681905
KitchenQual_encoded      0.675721
GarageFinish_encoded     0.553059
FireplaceQu_encoded      0.542181
Foundation_encoded       0.506328
GarageType_encoded       0.499204
BsmtFinType1_encoded     0.459141
HeatingQC_encoded        0.442154
MasVnrType_encoded       0.428108
Exterior2nd_encoded      0.392211
Exterior1st_encoded      0.390862
BsmtExposure_encoded     0.386653
SaleType_encoded         0.370523
SaleCondition_encoded    0.368100
MSZoning_encoded         0.327963
HouseStyle_encoded       0.293790
GarageQual_encoded       0.285344
GarageCond_encoded       0.285213
LotShape_encoded         0.276362
CentralAir_encoded       0.251328
Electrical_encoded       0.244235
RoofStyle_encoded        0.240201
PavedDrive_encoded       0.233537
BsmtCond_encoded         0.226706
Fence_encoded            0.188719
BldgType_encoded         0.185833
Condition1_encoded       0.180640
RoofMatl_encoded         0.177237
BsmtFinType2_encoded     0.174052
LandContour_encoded      0.160605
ExterCond_encoded        0.153680
PoolQC_encoded           0.145588
LotConfig_encoded        0.144981
Alley_encoded            0.142855
Functional_encoded       0.128376
Heating_encoded          0.120155
Condition2_encoded       0.099495
MiscFeature_encoded      0.084141
LandSlope_encoded        0.051784
Street_encoded           0.041036
Utilities_encoded        0.014314
Name: SalePrice, dtype: float64
MSZoning: p-value = 8.817633866272648e-35
Street: p-value = 0.11704860406782483
Alley: p-value = 2.9963796805460783e-07
LotShape: p-value = 6.447523852011766e-25
LandContour: p-value = 2.7422167521379096e-08
Utilities: p-value = 0.5847167739689381
LotConfig: p-value = 3.163167473604189e-06
LandSlope: p-value = 0.1413963584114019
Neighborhood: p-value = 1.5586002827707996e-225
Condition1: p-value = 8.904549416138853e-08
Condition2: p-value = 0.043425658360948464
BldgType: p-value = 2.0567364604967015e-10
HouseStyle: p-value = 3.376776535121222e-25
RoofStyle: p-value = 3.653523047099125e-17
RoofMatl: p-value = 7.231444779987188e-08
Exterior1st: p-value = 2.5860887286376316e-43
Exterior2nd: p-value = 4.8421856706985465e-43
MasVnrType: p-value = 1.2797035312662622e-63
ExterQual: p-value = 1.4395510967787893e-204
ExterCond: p-value = 5.106680608671862e-07
Foundation: p-value = 5.791895002232233e-91
BsmtQual: p-value = 8.15854808471181e-196
BsmtCond: p-value = 8.195793756122466e-16
BsmtExposure: p-value = 7.557758359196251e-50
BsmtFinType1: p-value = 2.3863579356150602e-71
BsmtFinType2: p-value = 5.22564949005886e-08
Heating: p-value = 0.000753472106445497
HeatingQC: p-value = 2.667062092104357e-67
CentralAir: p-value = 1.8095061559267854e-22
Electrical: p-value = 1.6412076757769925e-18
KitchenQual: p-value = 3.0322127528402335e-192
Functional: p-value = 0.0004841696801078294
FireplaceQu: p-value = 2.9712169727633336e-107
GarageType: p-value = 6.117025805439062e-87
GarageFinish: p-value = 6.228747181514921e-115
GarageQual: p-value = 5.388762379335977e-25
GarageCond: p-value = 5.711745645774751e-25
PavedDrive: p-value = 1.803568890651533e-18
PoolQC: p-value = 7.7009894157147e-07
Fence: p-value = 9.379976594788224e-11
MiscFeature: p-value = 0.0350036718754261
SaleType: p-value = 5.039766889462451e-42
SaleCondition: p-value = 7.988268404991176e-44
Important categorical features: ['MSZoning', 'Neighborhood', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'Found
```

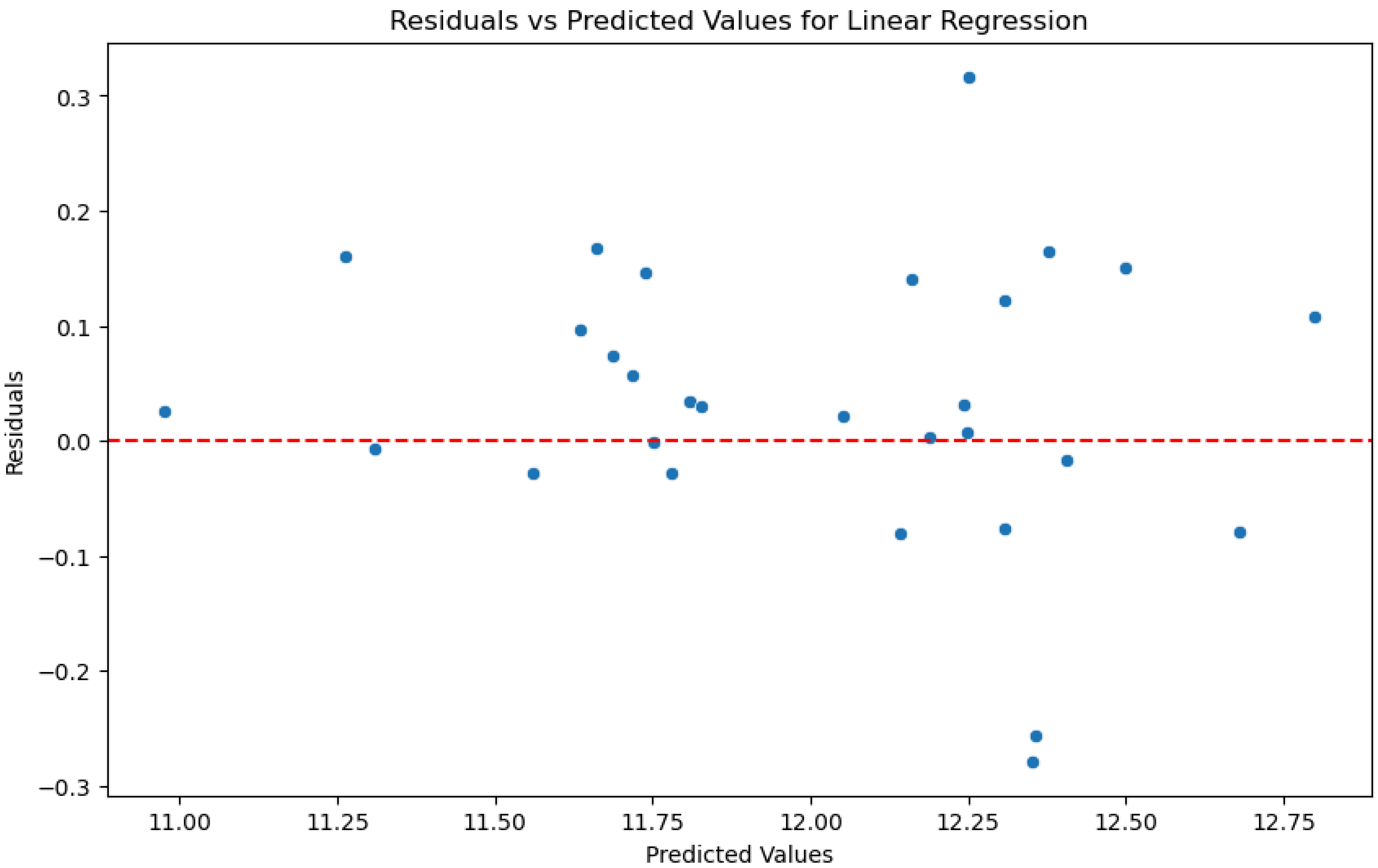
ation', 'BsmtQual', 'BsmtExposure', 'BsmtFinType1', 'HeatingQC', 'KitchenQual', 'FireplaceQu', 'GarageType', 'GarageFinish', 'SaleType', 'SaleCondition']

In [16]:
```
# Update the datasets after selection
train = train[selected_num_features + selected_cat_features]
selected_num_features.remove('SalePrice')
test = test[selected_num_features + selected_cat_features]

# Create dummy variables for important categorical features
train = pd.get_dummies(train, columns=selected_cat_features, drop_first=True)
test = pd.get_dummies(test, columns=selected_cat_features, drop_first=True)

# Convert a Boolean value to a numeric value
train = train.astype(int)
test = test.astype(int)

# Align train and test datasets
train, test = train.align(test, join='left', axis=1)
test.fillna(0, inplace=True)

# Prepare data for modeling
X = train.drop(columns=['SalePrice'])
y = np.log1p(train['SalePrice'])  # log transformation

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply scaling

X_train_scaled = X_train.copy()
X_val_scaled = X_val.copy()
test_scaled = test.copy()
scaler = StandardScaler()
X_train_scaled[selected_num_features] = scaler.fit_transform(X_train[selected_num_features])
X_val_scaled[selected_num_features] = scaler.transform(X_val[selected_num_features])
test_scaled[selected_num_features] = scaler.transform(test[selected_num_features])
test_scaled.drop(columns = ['SalePrice'], inplace = True)
```

In [17]:
```
# Model 1: Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train_scaled, y_train)
linear_preds = linear_model.predict(X_val_scaled)

# Evaluate Linear Regression
linear_mse = mean_squared_error(y_val, linear_preds)
linear_r2 = r2_score(y_val, linear_preds)
print("Linear Regression MSE:", linear_mse)
print("Linear Regression R2:", linear_r2)

# Residual analysis for Linear Regression
sample_index = np.random.choice(len(y_val), size=int(0.1 * len(y_val)), replace=False)
sample_linear_preds = linear_preds[sample_index]
residuals_linear = (y_val.iloc[sample_index] - sample_linear_preds)

# Residual normality
sns.histplot(residuals_linear, kde=True)
plt.title("Residual Distribution for Linear Regression")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()

# Residual Homoscedasticity
plt.figure(figsize=(10, 6))
sns.scatterplot(x=sample_linear_preds, y=residuals_linear)
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals vs Predicted Values for Linear Regression")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.show()

# Residual independence
dw_stat = durbin_watson(residuals_linear)
print(f"Durbin-Watson Statistic: {dw_stat}")
```

```
Linear Regression MSE: 9.438032194376308e+19
Linear Regression R2: -5.0575972324767295e+20
```



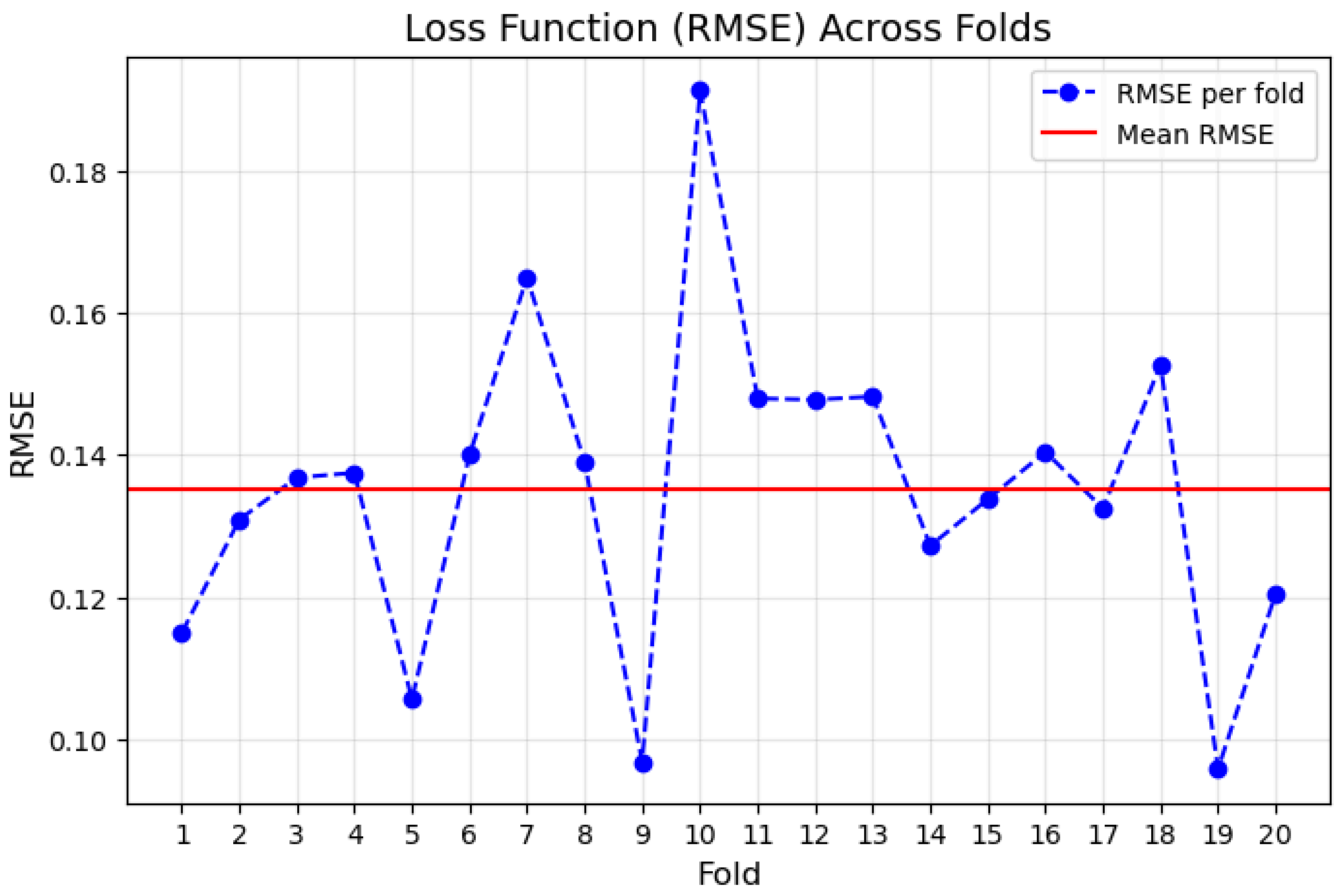


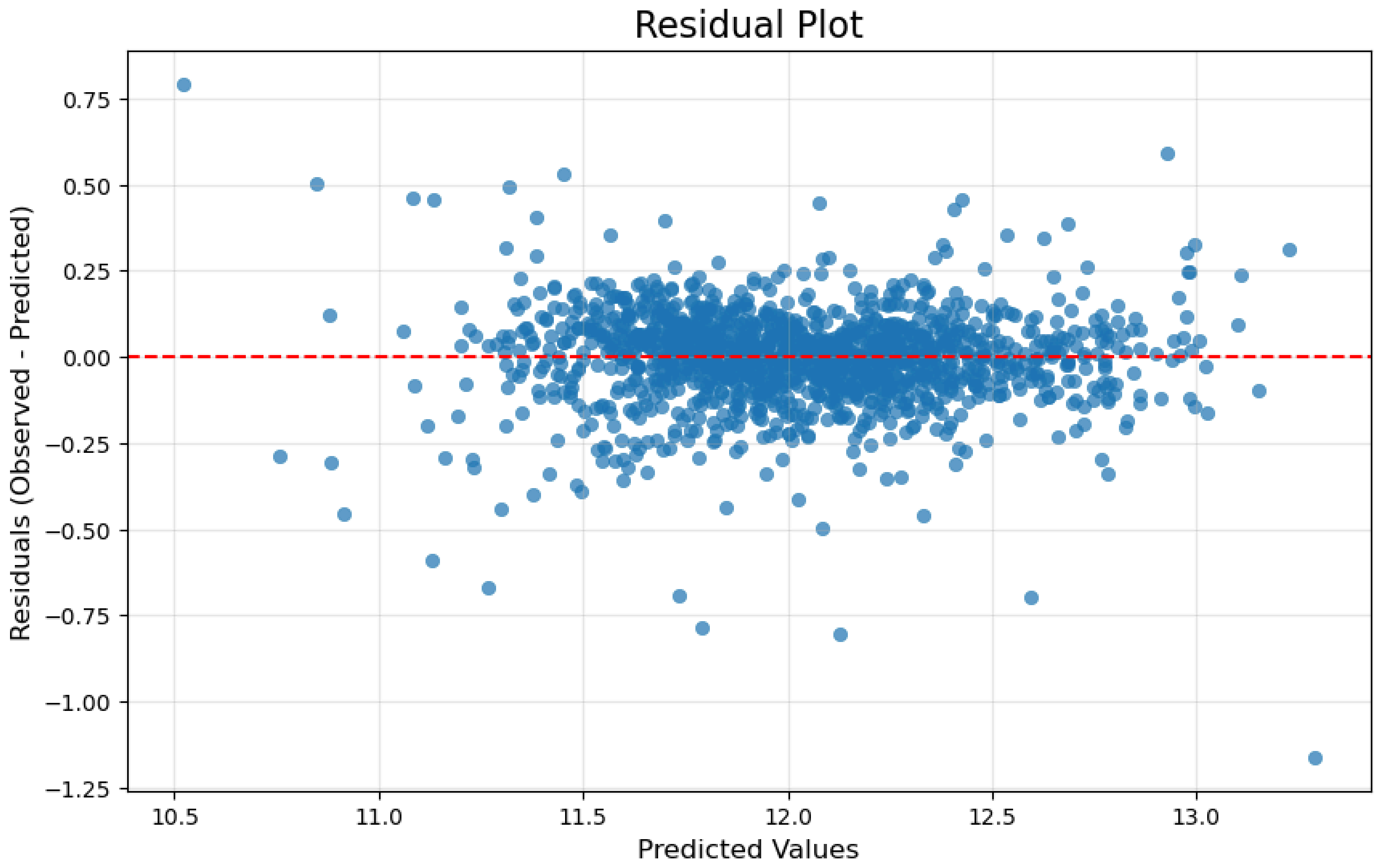

```
Durbin-Watson Statistic: 1.7536220148273862
```

In [18]:
```python
# Model 2: OLS

def ols_with_cross_validation_and_loss_plot(X, y, n_splits):
    """
    Perform OLS regression with k-fold cross-validation, feature importance visualization,
    and graph the loss function across folds.

    Parameters:
    X (pd.DataFrame): Feature matrix.
    y (pd.Series): Target vector.
    n_splits (int): Number of folds for cross-validation.

    Returns:
    dict: Cross-validation metrics and feature importance plot.
    """
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    fold_rmse = []
    fold_r2 = []

    residuals = []  # To store residuals for all folds
    predictions = []  # To store predictions for all folds
    actuals = []  # To store actual values for all folds

    # Apply scaling
    scaler = StandardScaler()
    X[selected_num_features] = scaler.fit_transform(X[selected_num_features])

    for fold_idx, (train_index, test_index) in enumerate(kf.split(X), start=1):
        # Split the data into training and testing sets
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
```

```
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        # Add a constant for the intercept
        X_train_const = sm.add_constant(X_train, has_constant="add")

        X_test_const = sm.add_constant(X_test, has_constant="add")

        # Fit the OLS model
        ols_model = sm.OLS(y_train, X_train_const).fit()

        # Predict on the validation set
        y_pred = ols_model.predict(X_test_const)

        # Calculate RMSE
        rmse = np.sqrt(mean_squared_error(y_test, y_pred))
        fold_rmse.append(rmse)

        # Calculate R squared
        r2 = r2_score(y_test, y_pred)
        fold_r2.append(r2)

        # Store residuals and predictions
        residuals.extend(y_test - y_pred)
        predictions.extend(y_pred)
        actuals.extend(y_test)

        # Print the fold RMSE
        print(f"Fold {fold_idx}: RMSE = {rmse:.4f}")
        print(f"Fold {fold_idx}: R2 = {r2:.4f}")

    # Plot the loss function across folds
    plt.figure(figsize=(8, 5))
    plt.plot(range(1, n_splits + 1), fold_rmse, marker='o', linestyle='--', color='b', label='RMSE per fold')
    plt.axhline(np.mean(fold_rmse), color='r', linestyle='-', label='Mean RMSE')
    plt.title('Loss Function (RMSE) Across Folds', fontsize=14)
    plt.xlabel('Fold', fontsize=12)
    plt.ylabel('RMSE', fontsize=12)
    plt.xticks(range(1, n_splits + 1))
    plt.grid(alpha=0.3)
    plt.legend()
    plt.show()

     # Plot Residuals
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=predictions, y=residuals, alpha=0.7, edgecolor=None)
    plt.axhline(0, color='r', linestyle='--')
    plt.title('Residual Plot', fontsize=16)
    plt.xlabel('Predicted Values', fontsize=12)
    plt.ylabel('Residuals (Observed - Predicted)', fontsize=12)
    plt.grid(alpha=0.3)
    plt.show()

    # Train the final OLS model on the full dataset
    X_const = sm.add_constant(X)
    final_model = sm.OLS(y, X_const).fit()

    # Print the OLS summary
    print(final_model.summary())

    # Create coefficients DataFrame
    coefficients = pd.DataFrame({
        "Feature": X.columns,
        "Coefficient Estimate": final_model.params[1:]  # Exclude the intercept
    }).sort_values(by="Coefficient Estimate", ascending=False)

    # Plot feature importance
    plt.figure(figsize=(12, len(coefficients) * 0.3))
    sns.barplot(x="Coefficient Estimate", y="Feature", data=coefficients, palette="viridis")
    plt.title("Feature Importance (OLS Coefficients)", fontsize=16)
    plt.xlabel("Coefficient Value")
    plt.ylabel("Feature")
    plt.grid(alpha=0.3)
    plt.show()

    return {
        "mean_rmse": np.mean(fold_rmse),
        "fold_rmse": fold_rmse,
        "final_model": final_model
    }

# Run OLS with cross-validation and loss function plot on the diabetes dataset
results = ols_with_cross_validation_and_loss_plot(X, y, n_splits=20)
print("Mean RMSE:", results["mean_rmse"])
print("Fold RMSEs:", results["fold_rmse"])
```

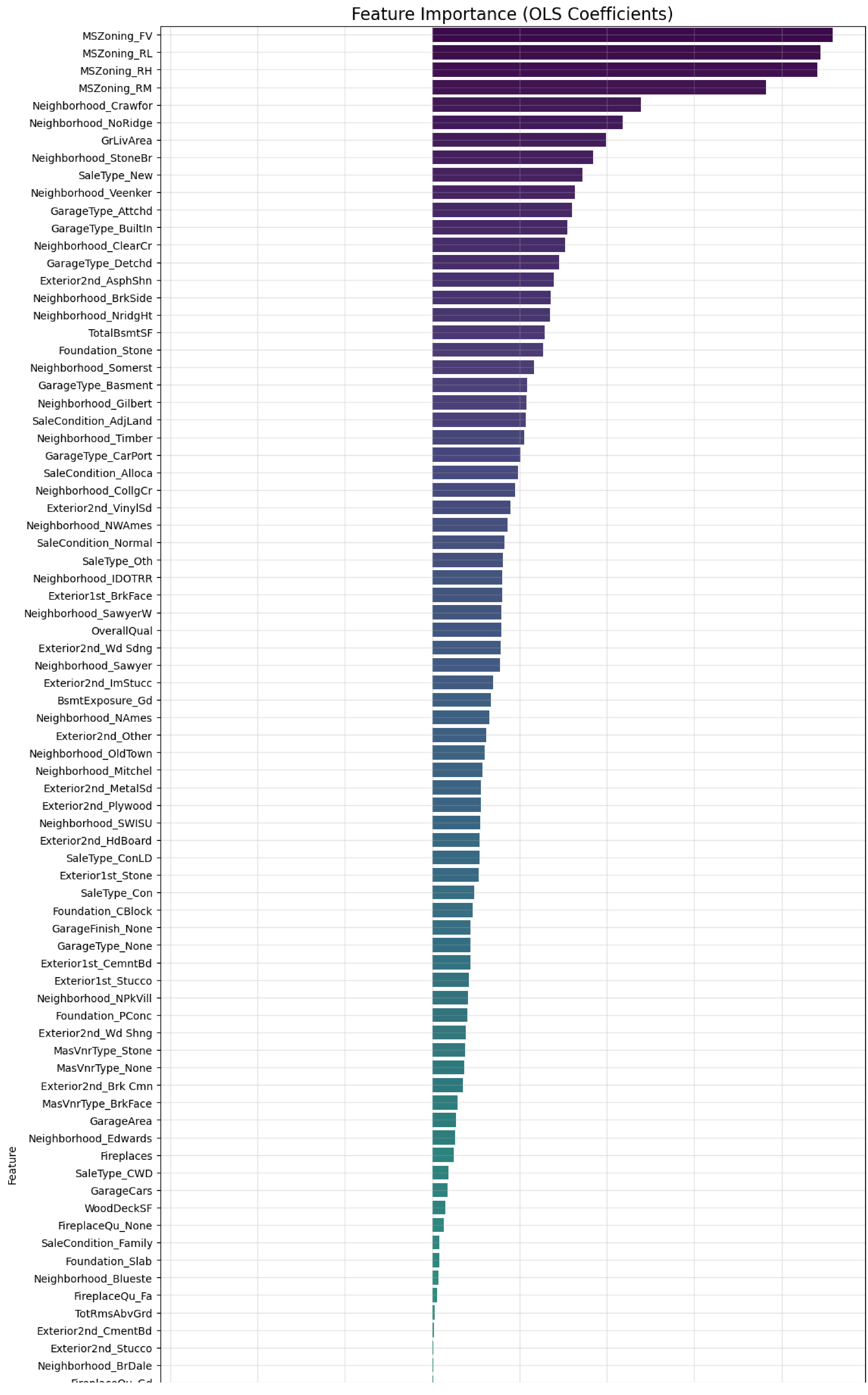
```
Fold 1: RMSE = 0.1149
Fold 1: R2 = 0.9151
Fold 2: RMSE = 0.1309
Fold 2: R2 = 0.9274
Fold 3: RMSE = 0.1368
Fold 3: R2 = 0.8956
Fold 4: RMSE = 0.1375
Fold 4: R2 = 0.8915
Fold 5: RMSE = 0.1056
Fold 5: R2 = 0.9061
Fold 6: RMSE = 0.1400
Fold 6: R2 = 0.8688
Fold 7: RMSE = 0.1650
Fold 7: R2 = 0.8569
Fold 8: RMSE = 0.1390
Fold 8: R2 = 0.8802
Fold 9: RMSE = 0.0968
Fold 9: R2 = 0.9260
Fold 10: RMSE = 0.1914
Fold 10: R2 = 0.7769
Fold 11: RMSE = 0.1479
Fold 11: R2 = 0.8093
Fold 12: RMSE = 0.1478
Fold 12: R2 = 0.8575
Fold 13: RMSE = 0.1482
Fold 13: R2 = 0.9181
Fold 14: RMSE = 0.1272
Fold 14: R2 = 0.8922
Fold 15: RMSE = 0.1338
Fold 15: R2 = 0.8703
Fold 16: RMSE = 0.1404
Fold 16: R2 = 0.8611
Fold 17: RMSE = 0.1325
Fold 17: R2 = 0.8817
Fold 18: RMSE = 0.1527
Fold 18: R2 = 0.8223
Fold 19: RMSE = 0.0958
Fold 19: R2 = 0.9162
Fold 20: RMSE = 0.1204
Fold 20: R2 = 0.9041
```

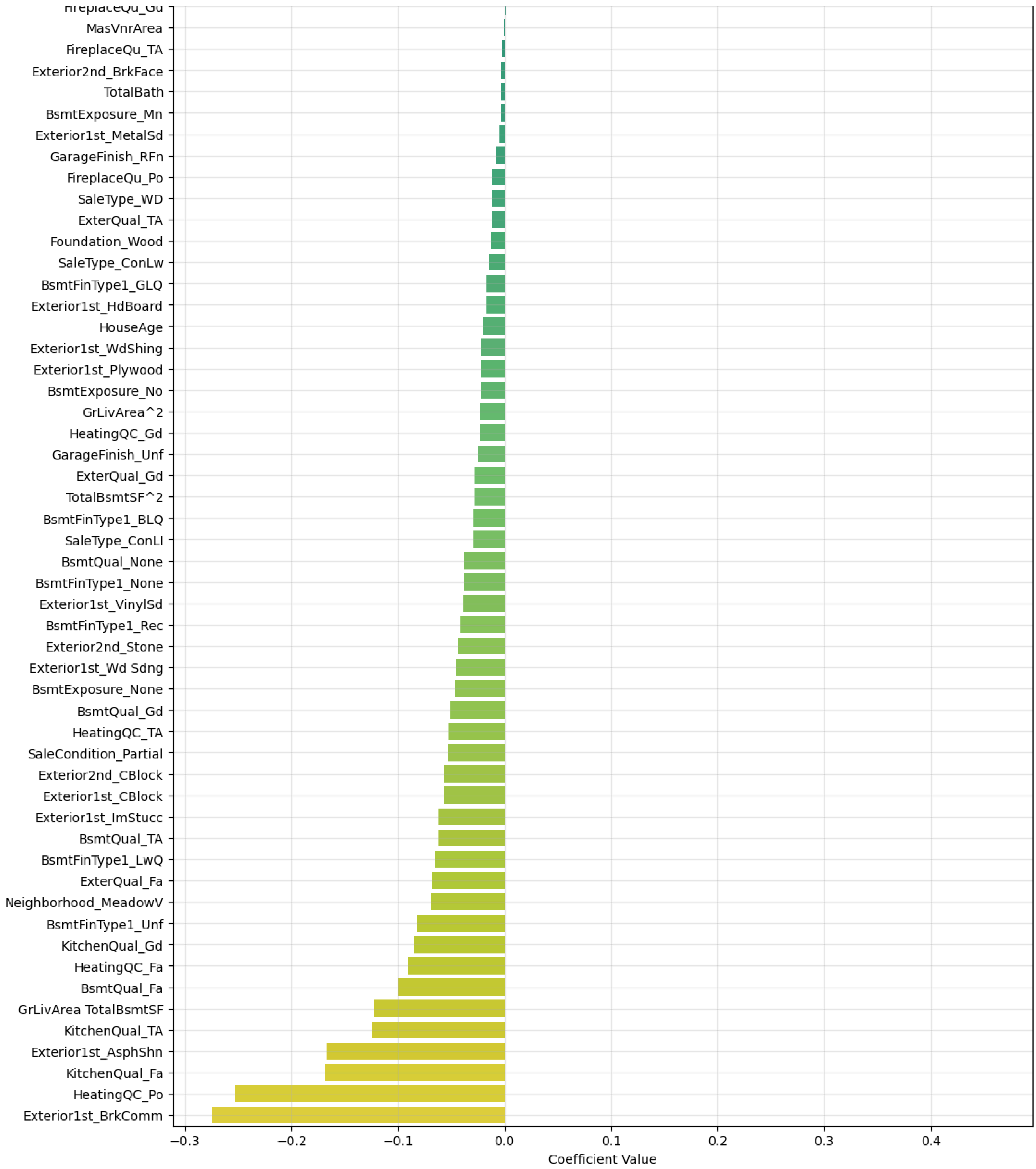



Residual Plot
Residuals (Observed - Predicted)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
10.5
11.0
11.5
12.0
12.5
13.0
Predicted Values

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              SalePrice   R-squared:                       0.912
Model:                            OLS   Adj. R-squared:                  0.904
Method:                 Least Squares   F-statistic:                     108.8
Date:                Fri, 24 Jan 2025   Prob (F-statistic):               0.00
Time:                        11:31:02   Log-Likelihood:                 1043.2
No. Observations:                1460   AIC:                            -1830.
Df Residuals:                    1332   BIC:                            -1154.
Df Model:                         127
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                    11.4135      0.104    109.418      0.000      11.209      11.618
OverallQual               0.0796      0.007     11.551      0.000       0.066       0.093
MasVnrArea               -0.0005      0.005     -0.100      0.920      -0.011       0.010
TotalBsmtSF               0.1287      0.012     10.947      0.000       0.106       0.152
GrLivArea                 0.1989      0.019     10.465      0.000       0.162       0.236
TotRmsAbvGrd              0.0026      0.007      0.375      0.707      -0.011       0.016
Fireplaces                0.0242      0.008      2.877      0.004       0.008       0.041
GarageCars                0.0176      0.009      2.018      0.044       0.000       0.035
GarageArea                0.0274      0.008      3.338      0.001       0.011       0.044
WoodDeckSF                0.0153      0.004      4.142      0.000       0.008       0.023
HouseAge                 -0.0203      0.010     -2.006      0.045      -0.040      -0.000
TotalBath                -0.0030      0.006     -0.538      0.590      -0.014       0.008
GrLivArea^2              -0.0228      0.025     -0.922      0.357      -0.071       0.026
GrLivArea TotalBsmtSF    -0.1231      0.036     -3.445      0.001      -0.193      -0.053
TotalBsmtSF^2            -0.0287      0.023     -1.272      0.204      -0.073       0.016
MSZoning_FV               0.4586      0.062      7.449      0.000       0.338       0.579
MSZoning_RH               0.4414      0.061      7.258      0.000       0.322       0.561
MSZoning_RL               0.4446      0.051      8.701      0.000       0.344       0.545
MSZoning_RM               0.3825      0.048      7.940      0.000       0.288       0.477
Neighborhood_Blueste      0.0073      0.099      0.074      0.941      -0.187       0.201
Neighborhood_BrDale       0.0013      0.054      0.025      0.980      -0.104       0.107
Neighborhood_BrkSide      0.1355      0.043      3.139      0.002       0.051       0.220
Neighborhood_ClearCr      0.1522      0.043      3.535      0.000       0.068       0.237
Neighborhood_CollgCr      0.0945      0.034      2.761      0.006       0.027       0.162
Neighborhood_Crawfor      0.2388      0.041      5.841      0.000       0.159       0.319
Neighborhood_Edwards      0.0267      0.038      0.696      0.487      -0.048       0.102
Neighborhood_Gilbert      0.1083      0.036      3.036      0.002       0.038       0.178
Neighborhood_IDOTRR       0.0800      0.050      1.602      0.109      -0.018       0.178
Neighborhood_MeadowV     -0.0694      0.056     -1.231      0.219      -0.180       0.041
Neighborhood_Mitchel      0.0578      0.039      1.475      0.140      -0.019       0.135
Neighborhood_NAmes        0.0651      0.037      1.756      0.079      -0.008       0.138
Neighborhood_NPkVill      0.0408      0.075      0.544      0.587      -0.106       0.188
Neighborhood_NWAmes       0.0862      0.038      2.266      0.024       0.012       0.161
Neighborhood_NoRidge      0.2178      0.040      5.394      0.000       0.139       0.297
Neighborhood_NridgHt      0.1348      0.038      3.565      0.000       0.061       0.209
Neighborhood_OldTown      0.0603      0.044      1.360      0.174      -0.027       0.147
Neighborhood_SWISU        0.0552      0.047      1.183      0.237      -0.036       0.147
Neighborhood_Sawyer       0.0777      0.039      2.009      0.045       0.002       0.154
Neighborhood_SawyerW      0.0796      0.038      2.119      0.034       0.006       0.153
Neighborhood_Somerst      0.1167      0.044      2.634      0.009       0.030       0.204
Neighborhood_StoneBr      0.1844      0.043      4.311      0.000       0.100       0.268
Neighborhood_Timber       0.1049      0.038      2.731      0.006       0.030       0.180
Neighborhood_Veenker      0.1630      0.053      3.101      0.002       0.060       0.266
Exterior1st_AsphShn      -0.1674      0.170     -0.983      0.326      -0.502       0.167
Exterior1st_BrkComm      -0.2748      0.139     -1.984      0.047      -0.547      -0.003
Exterior1st_BrkFace       0.0799      0.063      1.278      0.201      -0.043       0.203
Exterior1st_CBlock       -0.0566      0.069     -0.818      0.413      -0.192       0.079
Exterior1st_CemntBd       0.0438      0.096      0.455      0.649      -0.145       0.232
Exterior1st_HdBoard      -0.0169      0.063     -0.267      0.789      -0.141       0.107
Exterior1st_ImStucc      -0.0620      0.148     -0.420      0.675      -0.352       0.228
Exterior1st_MetalSd      -0.0046      0.072     -0.064      0.949      -0.147       0.137
Exterior1st_Plywood      -0.0221      0.062     -0.354      0.723      -0.145       0.100
Exterior1st_Stone         0.0528      0.115      0.461      0.645      -0.172       0.278
Exterior1st_Stucco        0.0421      0.069      0.608      0.543      -0.094       0.178
Exterior1st_VinylSd      -0.0386      0.066     -0.581      0.561      -0.169       0.092
Exterior1st_Wd Sdng      -0.0453      0.061     -0.745      0.456      -0.165       0.074
Exterior1st_WdShing      -0.0220      0.065     -0.338      0.735      -0.150       0.106
Exterior2nd_AsphShn       0.1394      0.110      1.267      0.205      -0.076       0.355
Exterior2nd_Brk Cmn       0.0350      0.104      0.336      0.737      -0.169       0.239
Exterior2nd_BrkFace      -0.0027      0.067     -0.041      0.968      -0.134       0.129
Exterior2nd_CBlock       -0.0566      0.069     -0.818      0.413      -0.192       0.079
Exterior2nd_CmentBd       0.0019      0.096      0.020      0.984      -0.187       0.191
Exterior2nd_HdBoard       0.0542      0.062      0.872      0.383      -0.068       0.176
Exterior2nd_ImStucc       0.0701      0.073      0.966      0.334      -0.072       0.212
Exterior2nd_MetalSd       0.0561      0.072      0.776      0.438      -0.086       0.198
Exterior2nd_Other         0.0619      0.145      0.427      0.669      -0.222       0.346
Exterior2nd_Plywood       0.0555      0.060      0.921      0.357      -0.063       0.174
Exterior2nd_Stone        -0.0438      0.086     -0.510      0.610      -0.212       0.125
Exterior2nd_Stucco        0.0015      0.068      0.022      0.983      -0.132       0.135
Exterior2nd_VinylSd       0.0895      0.066      1.363      0.173      -0.039       0.218
Exterior2nd_Wd Sdng       0.0785      0.060      1.303      0.193      -0.040       0.197
Exterior2nd_Wd Shng       0.0385      0.063      0.615      0.539      -0.084       0.161
MasVnrType_BrkFace        0.0291      0.035      0.834      0.405      -0.039       0.098
MasVnrType_None           0.0369      0.035      1.046      0.296      -0.032       0.106
MasVnrType_Stone          0.0381      0.037      1.022      0.307      -0.035       0.111
ExterQual_Fa             -0.0679      0.050     -1.345      0.179      -0.167       0.031
ExterQual_Gd             -0.0284      0.025     -1.145      0.253      -0.077       0.020
```

```
ExterQual_TA           -0.0120      0.027     -0.438      0.661      -0.066       0.042
Foundation_CBlock       0.0464      0.016      2.904      0.004       0.015       0.078
Foundation_PConc        0.0402      0.018      2.296      0.022       0.006       0.075
Foundation_Slab         0.0084      0.049      0.171      0.864      -0.088       0.105
Foundation_Stone        0.1273      0.054      2.352      0.019       0.021       0.233
Foundation_Wood        -0.0129      0.077     -0.169      0.866      -0.163       0.137
BsmtQual_Fa            -0.1000      0.032     -3.157      0.002      -0.162      -0.038
BsmtQual_Gd            -0.0506      0.017     -2.922      0.004      -0.085      -0.017
BsmtQual_None          -0.0376      0.067     -0.560      0.576      -0.169       0.094
BsmtQual_TA            -0.0622      0.021     -2.938      0.003      -0.104      -0.021
BsmtExposure_Gd         0.0670      0.015      4.437      0.000       0.037       0.097
BsmtExposure_Mn        -0.0030      0.015     -0.196      0.844      -0.033       0.027
BsmtExposure_No        -0.0222      0.011     -2.084      0.037      -0.043      -0.001
BsmtExposure_None      -0.0466      0.125     -0.371      0.711      -0.293       0.199
BsmtFinType1_BLQ       -0.0294      0.014     -2.073      0.038      -0.057      -0.002
BsmtFinType1_GLQ       -0.0168      0.013     -1.329      0.184      -0.042       0.008
BsmtFinType1_LwQ       -0.0652      0.018     -3.654      0.000      -0.100      -0.030
BsmtFinType1_None      -0.0376      0.067     -0.560      0.576      -0.169       0.094
BsmtFinType1_Rec       -0.0409      0.015     -2.714      0.007      -0.070      -0.011
BsmtFinType1_Unf       -0.0823      0.012     -6.708      0.000      -0.106      -0.058
HeatingQC_Fa           -0.0906      0.021     -4.359      0.000      -0.131      -0.050
HeatingQC_Gd           -0.0231      0.011     -2.153      0.031      -0.044      -0.002
HeatingQC_Po           -0.2532      0.128     -1.975      0.049      -0.505      -0.002
HeatingQC_TA           -0.0526      0.010     -5.074      0.000      -0.073      -0.032
KitchenQual_Fa         -0.1685      0.031     -5.493      0.000      -0.229      -0.108
KitchenQual_Gd         -0.0844      0.018     -4.606      0.000      -0.120      -0.048
KitchenQual_TA         -0.1242      0.020     -6.139      0.000      -0.164      -0.085
FireplaceQu_Fa          0.0054      0.036      0.149      0.881      -0.066       0.077
FireplaceQu_Gd          0.0010      0.028      0.035      0.972      -0.054       0.056
FireplaceQu_None        0.0134      0.033      0.409      0.683      -0.051       0.078
FireplaceQu_Po         -0.0115      0.041     -0.282      0.778      -0.092       0.069
FireplaceQu_TA         -0.0025      0.029     -0.087      0.931      -0.060       0.055
GarageType_Attchd       0.1599      0.056      2.842      0.005       0.050       0.270
GarageType_Basment      0.1086      0.064      1.687      0.092      -0.018       0.235
GarageType_BuiltIn      0.1550      0.059      2.647      0.008       0.040       0.270
GarageType_CarPort      0.1008      0.072      1.392      0.164      -0.041       0.243
GarageType_Detchd       0.1455      0.056      2.583      0.010       0.035       0.256
GarageType_None         0.0440      0.031      1.416      0.157      -0.017       0.105
GarageFinish_None       0.0440      0.031      1.416      0.157      -0.017       0.105
GarageFinish_RFn       -0.0082      0.010     -0.792      0.428      -0.028       0.012
GarageFinish_Unf       -0.0251      0.012     -2.010      0.045      -0.050      -0.001
SaleType_CWD            0.0187      0.068      0.276      0.782      -0.114       0.152
SaleType_Con            0.0484      0.095      0.511      0.609      -0.138       0.234
SaleType_ConLD          0.0537      0.049      1.091      0.276      -0.043       0.150
SaleType_ConLI         -0.0294      0.061     -0.482      0.630      -0.149       0.090
SaleType_ConLw         -0.0140      0.062     -0.227      0.820      -0.135       0.107
SaleType_New            0.1718      0.079      2.169      0.030       0.016       0.327
SaleType_Oth            0.0809      0.077      1.047      0.295      -0.071       0.232
SaleType_WD            -0.0119      0.022     -0.547      0.585      -0.055       0.031
SaleCondition_AdjLand   0.1068      0.069      1.539      0.124      -0.029       0.243
SaleCondition_Alloca    0.0986      0.042      2.346      0.019       0.016       0.181
SaleCondition_Family    0.0085      0.032      0.265      0.791      -0.055       0.072
SaleCondition_Normal    0.0824      0.015      5.564      0.000       0.053       0.111
SaleCondition_Partial  -0.0532      0.076     -0.697      0.486      -0.203       0.097
==============================================================================
Omnibus:                      301.095   Durbin-Watson:                   1.967
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1779.806
Skew:                          -0.823   Prob(JB):                         0.00
Kurtosis:                       8.152   Cond. No.                     1.11e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.41e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```


Feature Importance (OLS Coefficients)
Feature
MSZoning_FV
MSZoning_RL
MSZoning_RH
MSZoning_RM
Neighborhood_Crawfor
Neighborhood_NoRidge
GrLivArea
Neighborhood_StoneBr
SaleType_New
Neighborhood_Veenker
GarageType_Attchd
GarageType_BuiltIn
Neighborhood_ClearCr
GarageType_Detchd
Exterior2nd_AsphShn
Neighborhood_BrkSide
Neighborhood_NridgHt
TotalBsmtSF
Foundation_Stone
Neighborhood_Somerst
GarageType_Basment
Neighborhood_Gilbert
SaleCondition_AdjLand
Neighborhood_Timber
GarageType_CarPort
SaleCondition_Alloca
Neighborhood_CollgCr
Exterior2nd_VinylSd
Neighborhood_NWAmes
SaleCondition_Normal
SaleType_Oth
Neighborhood_IDOTRR
Exterior1st_BrkFace
Neighborhood_SawyerW
OverallQual
Exterior2nd_Wd Sdng
Neighborhood_Sawyer
Exterior2nd_ImStucc
BsmtExposure_Gd
Neighborhood_NAmes
Exterior2nd_Other
Neighborhood_OldTown
Neighborhood_Mitchel
Exterior2nd_MetalSd
Exterior2nd_Plywood
Neighborhood_SWISU
Exterior2nd_HdBoard
SaleType_ConLD
Exterior1st_Stone
SaleType_Con
Foundation_CBlock
GarageFinish_None
GarageType_None
Exterior1st_CemntBd
Exterior1st_Stucco
Neighborhood_NPkVill
Foundation_PConc
Exterior2nd_Wd Shng
MasVnrType_Stone
MasVnrType_None
Exterior2nd_Brk Cmn
MasVnrType_BrkFace
GarageArea
Neighborhood_Edwards
Fireplaces
SaleType_CWD
GarageCars
WoodDeckSF
FireplaceQu_None
SaleCondition_Family
Foundation_Slab
Neighborhood_Blueste
FireplaceQu_Fa
TotRmsAbvGrd
Exterior2nd_CmentBd
Exterior2nd_Stucco
Neighborhood_BrDale




```
Mean RMSE: 0.1352205598606172
Fold RMSEs: [0.11491628999747724, 0.13089324894671112, 0.13680350534315486, 0.13748157001002845, 0.10557397323743635, 0.1399
5219393332659, 0.1649903215336288, 0.13898748364882807, 0.09678305401035807, 0.1913661124976918, 0.14794618660399353, 0.1477
7697106089666, 0.14820973941577, 0.12720841356478443, 0.13376718734790097, 0.14042225418591261, 0.1324604548653019, 0.152691
42959054324, 0.09577750982366161, 0.12040329759493856]
```

In [19]:

```
# Predictions for test set
test_scaled_const = sm.add_constant(test_scaled, has_constant="add")
test_preds = results['final_model'].predict(test_scaled_const)
test_preds_exp = np.expm1(test_preds)  # exp(test_preds_log) - 1

# Save predictions to CSV
submission = pd.DataFrame({
    'Id': test_id['Id'],
    'SalePrice': test_preds_exp
})
submission.to_csv('submission.csv', index=False)
```