Comprehensive Predictive Modeling of Ames House Prices

June 17, 2024

1 Comprehensive Predictive Modeling of Ames House Prices

1. Introduction (200 words) • Overview of Real Estate Price Prediction (100 words): Discuss the significance of predicting real estate prices, its impact on industries like marketing, business intelligence, and urban planning. • Scope and Objectives (50 words): Outline the aims of the research, including developing predictive models and evaluating their effectiveness. • Structure of the Report (50 words): Briefly describe the structure of the report, summarizing the main sections.

2 2. Exploratory Data Analysis

2.1 Dataset Description

• Description of the Dataset (100 words): Explain the key attributes (e.g., zoning, lot size, building characteristics, neighborhood factors). • Source (25 words): Cite the dataset source (e.g., Kaggle, UCI Machine Learning Repository).

Import libraries

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import skew

%matplotlib inline
sns.set_style('darkgrid')
```

Load data

[ ]:
```
dataset = pd.read_csv("house-price-data-apr2024.csv")
dataset.describe()
```

[ ]:
```
       LotFrontage       LotArea  OverallQual  OverallCond   MasVnrArea  \
count  1201.000000   1460.000000  1460.000000  1460.000000  1452.000000
mean     70.049958  10516.828082     6.099315     5.575342   103.685262
std      24.284752   9981.264932     1.382997     1.112799   181.066207
min      21.000000   1300.000000     1.000000     1.000000     0.000000
```

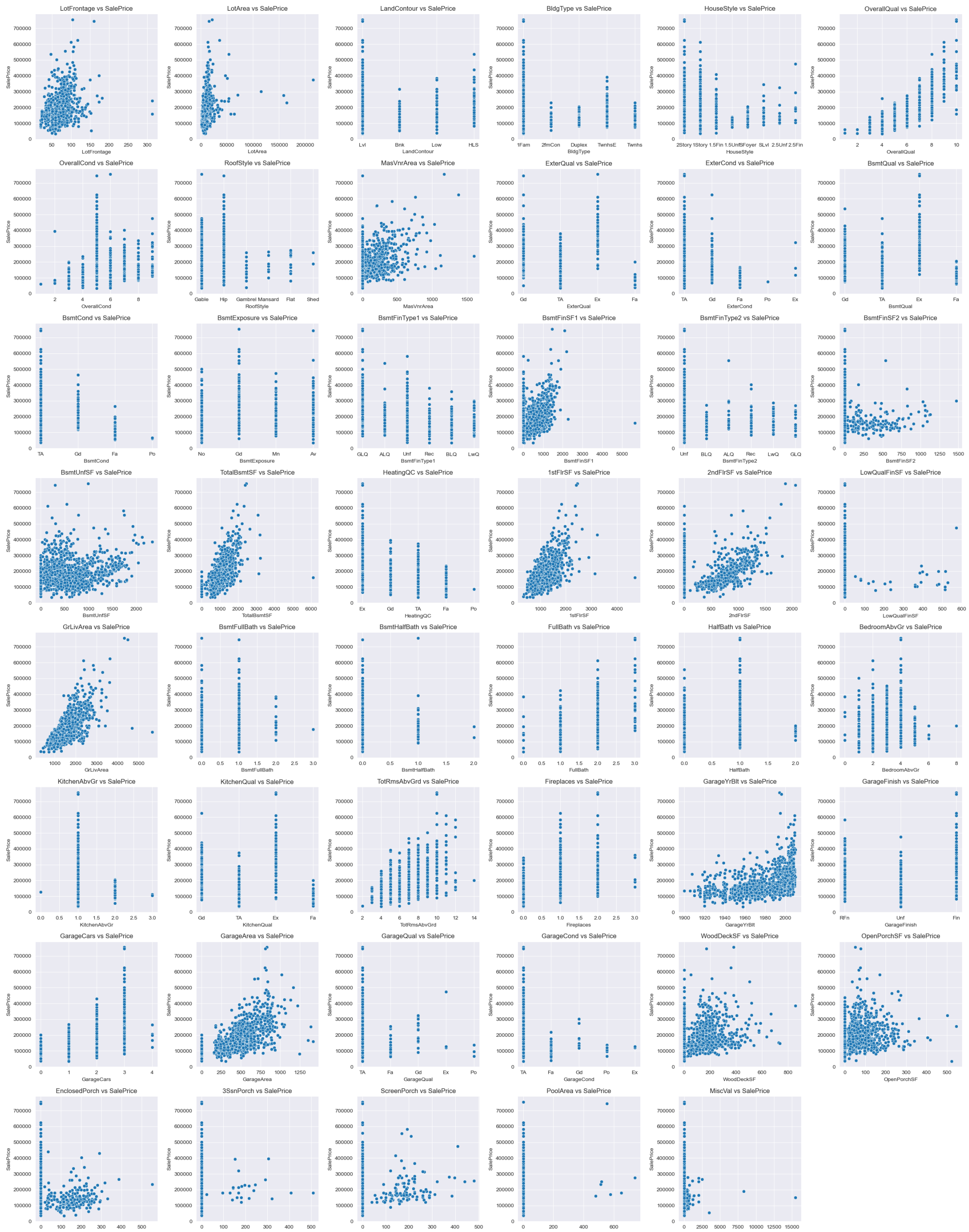
```
25%      59.000000    7553.500000     5.000000     5.000000     0.000000
50%      69.000000    9478.500000     6.000000     5.000000     0.000000
75%      80.000000   11601.500000     7.000000     6.000000   166.000000
max     313.000000  215245.000000    10.000000     9.000000  1600.000000

        BsmtFinSF1   BsmtFinSF2    BsmtUnfSF  TotalBsmtSF     1stFlrSF  …  \
count  1460.000000  1460.000000  1460.000000  1460.000000  1460.000000  …
mean    443.639726    46.549315   567.240411  1057.429452  1162.626712  …
std     456.098091   161.319273   441.866955   438.705324   386.587738  …
min       0.000000     0.000000     0.000000     0.000000   334.000000  …
25%       0.000000     0.000000   223.000000   795.750000   882.000000  …
50%     383.500000     0.000000   477.500000   991.500000  1087.000000  …
75%     712.250000     0.000000   808.000000  1298.250000  1391.250000  …
max    5644.000000  1474.000000  2336.000000  6110.000000  4692.000000  …

        GarageCars   GarageArea   WoodDeckSF  OpenPorchSF  EnclosedPorch  \
count  1460.000000  1460.000000  1460.000000  1460.000000    1460.000000
mean      1.767123   472.980137    94.244521    46.660274      21.954110
std       0.747315   213.804841   125.338794    66.256028      61.119149
min       0.000000     0.000000     0.000000     0.000000       0.000000
25%       1.000000   334.500000     0.000000     0.000000       0.000000
50%       2.000000   480.000000     0.000000    25.000000       0.000000
75%       2.000000   576.000000   168.000000    68.000000       0.000000
max       4.000000  1418.000000   857.000000   547.000000     552.000000

         3SsnPorch  ScreenPorch     PoolArea       MiscVal      SalePrice
count  1460.000000  1460.000000  1460.000000   1460.000000    1460.000000
mean      3.409589    15.060959     2.758904     43.489041  180921.195890
std      29.317331    55.757415    40.177307    496.123024   79442.502883
min       0.000000     0.000000     0.000000      0.000000   34900.000000
25%       0.000000     0.000000     0.000000      0.000000  129975.000000
50%       0.000000     0.000000     0.000000      0.000000  163000.000000
75%       0.000000     0.000000     0.000000      0.000000  214000.000000
max     508.000000   480.000000   738.000000  15500.000000  755000.000000

[8 rows x 32 columns]
```

2.1 Feature Distributions Relative to target

A comprehensive initial assessment of how each of the 46 features is distributed relative to the target variable `SalePrice` is visualized.

[ ]:
```python
# Assuming 'dataset' is your DataFrame
df = dataset.copy()  # Make a copy to avoid modifying the original dataset

# List of features to visualize
features = df.columns.tolist()
features.remove('SalePrice')
```

```python
target = 'SalePrice'

# Number of rows and columns for subplots
num_rows = 8
num_cols = 6

# Create subplots
fig, axes = plt.subplots(num_rows, num_cols, figsize=(25, 4*num_rows))
axes = axes.flatten()

# Plot each feature and calculate skewness
for i, feature in enumerate(features):
    # Plotting scatterplot
    sns.scatterplot(x=df[feature], y=df[target], ax=axes[i])
    axes[i].set_title(f'{feature} vs {target}')
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel(target)

# Remove any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



2.2 Sale Price

[ ]:
```
saleprice = dataset["SalePrice"]
sns.displot(saleprice,kde = True)
```

```
plt.title("House Price Distribution")
plt.show()
```

From the plot the distribution of `SalePrice` data is apparently right skewed. Its Skewness and Kurtosis statistics are checked.This was expected as few people can afford very expensive houses.

[ ]:
```
print(f"""Skewness: {saleprice.skew()}
Kurtosis: {saleprice.kurt()}""")
```

```
Skewness: 1.8828757597682129
Kurtosis: 6.536281860064529
```

2.3 Numerical Features

There are total 31 numercial features.

[ ]:
```
numerical_data = df.select_dtypes(include="number")
numerical_list = numerical_data.columns.tolist()
numerical_list.remove("SalePrice")
```

```python
print("Number of numerical variables " , len(numerical_list))
```

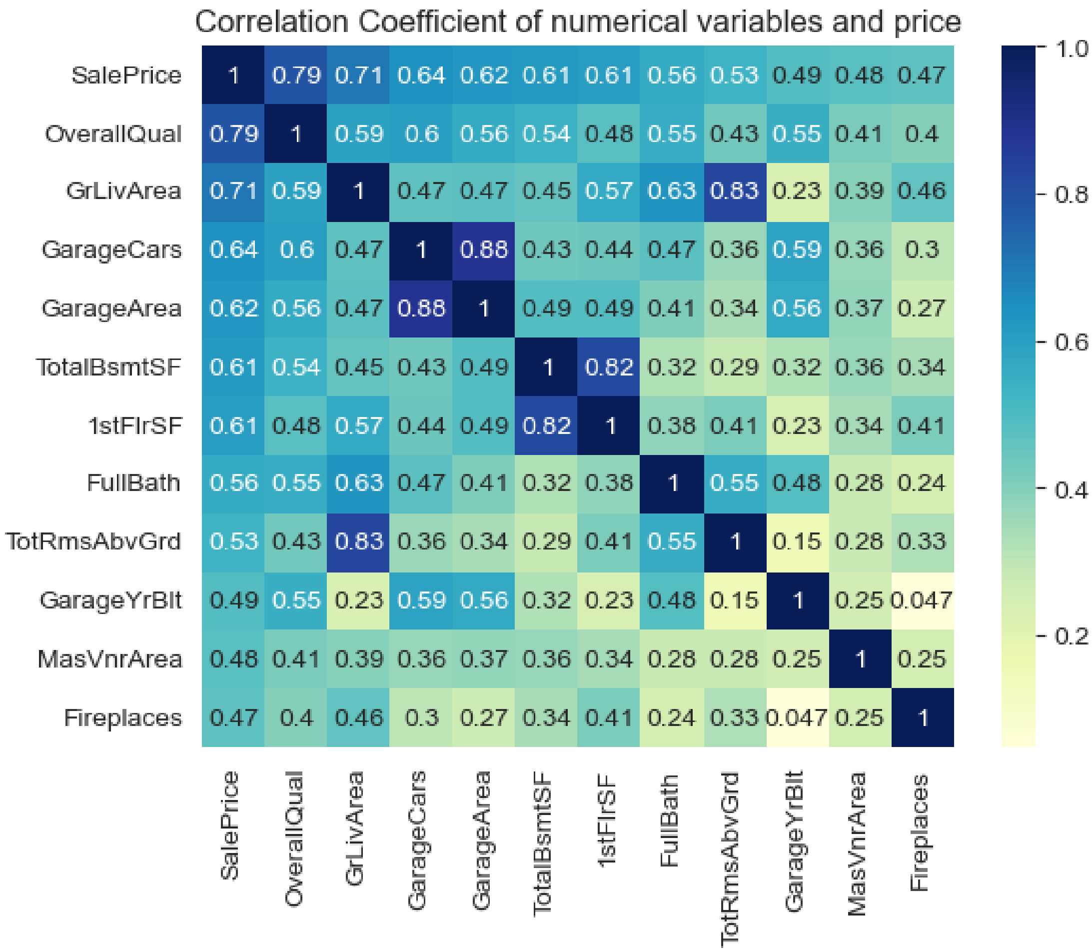
```
Number of numerical variables  31
```

Correlation coefficient with target variable

To have a first glance of which numeric variables have a high correlation with the `SalePrice`. The correlation coefficient is computed. All coefficient>0.4 is initially defined as a relative strong linear relationship and visualized. However, it is clear that the multicollinearity is an issue. For example: the correlation between `GarageCars` and `GarageArea` is very high (0.89), while both have relatively high correlations with `SalePrice`.

Pearson Correlation Range	Strength of Correlation	Interpretation
0.7 to 1.0 (positive or negative)	Strong	Indicates a strong linear relationship where one variable tends to increase (or decrease) as the other variable increases.
0.5 to 0.7 (positive or negative)	Moderate to Strong	Suggests a meaningful linear relationship between the variables, though not as strong as the highest range.
0.3 to 0.5 (positive or negative)	Weak to Moderate	The relationship exists but is less pronounced and may not be as influential in linear models.
0 to 0.3 (positive or negative)	Weak or No Correlation	Indicates a weak linear relationship or no linear relationship between the variables.

[ ]:
```python
corr = numerical_data.corr()

corr_sorted = corr["SalePrice"].sort_values(ascending=False)
corr_high = corr_sorted[abs(corr_sorted)>0.4].index.tolist()
corr_numVar = corr.loc[corr_high,corr_high]

sns.heatmap(corr_numVar, cmap="YlGnBu", annot=True)
plt.title("Correlation Coefficient of numerical variables and price")
plt.show()
```



Skewness

Skewness Range	Description
Near Zero (0)	Data is symmetric, with skewness between -0.5 and 0.5.
Moderate Skewness	Data is moderately asymmetric, with skewness between -1 to -0.5 or 0.5 to 1.
High Skewness	Data is highly skewed, with skewness greater than 1 or less than -1. Transformations needed.

[ ]:
```
# Make a copy of numerical_data to avoid modifying the original dataset
numerical = numerical_data.drop(columns=["SalePrice"]).copy()
features = numerical.columns.tolist()
skewed_features = []

num_rows = 8
num_cols = 6
```

```python
# Create subplots for visualization
fig, axes = plt.subplots(num_rows, num_cols, figsize=(25, 4*num_rows))
axes = axes.flatten()

# Iterate over each feature
for i, feature in enumerate(features):
    # Plot histogram with KDE (Kernel Density Estimate)
    sns.histplot(numerical[feature], kde=True, ax=axes[i])

    # Calculate skewness for the current feature
    skewness = skew(numerical[feature].dropna())

    # Identify and store features with skewness greater than 1
    if skewness > 1:
        skewed_features.append(feature)

    axes[i].set_title(f'{feature} - Skewness: {skewness:.2f}')
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('')

print("Skewed Features:", skewed_features)
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```

```
Skewed Features: ['LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1',
'BsmtFinSF2', 'TotalBsmtSF', '1stFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtHalfBath', 'KitchenAbvGr', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch',
'3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal']
```

Characteristics summary of numercial variable derived from EDA

Numercial Variable	Characteristics summary derived from EDA
LotFrontage	Skewed data, initially had outliers, strong correlation with SalePrice.
LotArea	Skewed data, initially had outliers, moderate correlation with SalePrice.
OverallQual	Strong positive correlation with SalePrice.
OverallCond	Weak negative correlation with SalePrice.
MasVnrArea	Skewed data, moderate correlation with SalePrice.
BsmtFinSF1	Skewed data, moderate correlation with SalePrice.
BsmtFinSF2	Skewed data, low correlation with SalePrice.
BsmtUnfSF	Low correlation with SalePrice.
TotalBsmtSF	Skewed data, initially had outliers, strong correlation with SalePrice.
1stFlrSF	Skewed data, moderate correlation with SalePrice.
2ndFlrSF	Low correlation with SalePrice.

Numercial Variable	Characteristics summary derived from EDA
LowQualFinSF	Low correlation with SalePrice.
GrLivArea	Skewed data, strong correlation with SalePrice.
BsmtFullBath	Low correlation with SalePrice.
BsmtHalfBath	Skewed data, low correlation with SalePrice.
FullBath	Moderate correlation with SalePrice.
HalfBath	Low correlation with SalePrice.
BedroomAbvGr	Low correlation with SalePrice.
KitchenAbvGr	Skewed data, weak negative correlation with SalePrice.
TotRmsAbvGrd	Moderate correlation with SalePrice.
Fireplaces	Moderate correlation with SalePrice.
GarageYrBlt	Low correlation with SalePrice.
GarageCars	Moderate correlation with SalePrice.
GarageArea	Moderate correlation with SalePrice.
WoodDeckSF	Skewed data, moderate correlation with SalePrice.
OpenPorchSF	Skewed data, moderate correlation with SalePrice.
EnclosedPorch	Low correlation with SalePrice.
3SsnPorch	Low correlation with SalePrice.
ScreenPorch	Low correlation with SalePrice.
PoolArea	Low correlation with SalePrice.
MiscVal	Skewed data, initially had outliers, low correlation with SalePrice.

2.4 Categorical Features

There are total 16 categorical features.

[ ]:
```
categorical_data = df.select_dtypes(exclude="number")
categorical_list = categorical_data.columns.tolist()
print("Number of categorical variables " , len(categorical_list))
```

```
Number of categorical variables  16
```

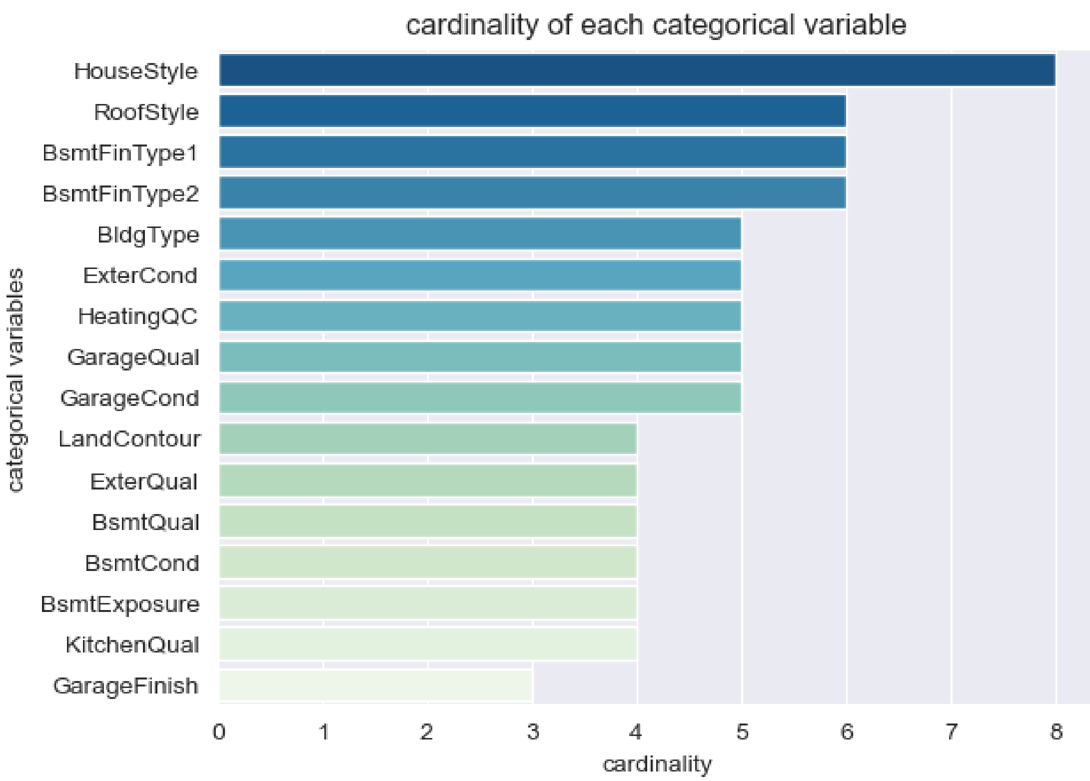
[ ]:
```
print(categorical_list)
```

```
['LandContour', 'BldgType', 'HouseStyle', 'RoofStyle', 'ExterQual', 'ExterCond',
'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
'HeatingQC', 'KitchenQual', 'GarageFinish', 'GarageQual', 'GarageCond']
```

To have an initial feel of categorical variables 's characteristics. The cardinality of categorical variables is visualized. Categorical variables with high cardinality (i.e. `HouseStyle`, `RoofSytle`, `BsmtFinType1`, and `BsmtFinType2`) may have a significant impact on the analysis or modeling process.

[ ]:
```
cardinality = {var: df[var].nunique() for var in categorical_list}
cardinality_df = pd.DataFrame(cardinality.items(), columns=["Variable",␣
↪"Cardinality"])
cardinality_df=cardinality_df.sort_values(by="Cardinality",ascending=False)
```

```
sns.
 ↪barplot(y=cardinality_df["Variable"],x=cardinality_df["Cardinality"],data=cardinality_df,␣
 ↪palette="GnBu_r")
plt.ylabel("categorical variables")
plt.xlabel("cardinality")
plt.title("cardinality of each categorical variable")
plt.show()
```



3 3. Data Preprocessing

3.1 Missing Value

For numerical variables, given that the missing percentage for `LotFrontage` and `GarageYrBlt` is relatively high (17.7% and 5.5% respectively), creating missing value indicators is more suitable compared to other approaches such as mean imputation. Each of these variables are replaced by a new variable that works as a missing value indicator. If the value in the original variable is missing then the corresponding value in the new variable should be 1, otherwise it should be 0.

[ ]:
```
numerical_missing = numerical_data.isna().sum().sort_values(ascending=False)
numerical_missing_percent = numerical_missing/len(numerical_data) * 100
numerical_missing_data = pd.DataFrame({
                                     "Percent": numerical_missing_percent,
```

```
                                             "Count": numerical_missing})
numerical_missing_data.head(10)
```

[ ]:

	Percent	Count
LotFrontage	17.739726	259
GarageYrBlt	5.547945	81
MasVnrArea	0.547945	8
BedroomAbvGr	0.000000	0
MiscVal	0.000000	0
PoolArea	0.000000	0
ScreenPorch	0.000000	0
3SsnPorch	0.000000	0
EnclosedPorch	0.000000	0
OpenPorchSF	0.000000	0

[ ]:

```
vars_with_missing = []
for var in df.select_dtypes(include="number").columns:
   if df[var].isnull().any():
      vars_with_missing.append(var)
for var in vars_with_missing:
   df[var+"_missing"] = df[var].isnull().astype(int)
   df.drop(columns=[var],inplace=True)

sum(df.select_dtypes(include="number").isna().sum())
```

[ ]: 0

For categorical variables, the proportion of missing values is relatively low (`GarageFinish`, `GarageQual`, and '`BsmtExposure` range between 2.5% and 5.5%, this is considered low) and unlikely to skew the data. Mode imputation for categorical variables replaces missing values with the most frequent category.

[ ]:

```
categorical_missing = categorical_data.isna().sum().sort_values(ascending=False)
categorical_missing_percent = categorical_missing/len(categorical_data) * 100
categorical_missing_data = pd.DataFrame({
                                             "Percent": categorical_missing_percent,
                                             "Count": categorical_missing})
categorical_missing_data.head(10)
```

[ ]:

	Percent	Count
GarageFinish	5.547945	81
GarageQual	5.547945	81
GarageCond	5.547945	81
BsmtExposure	2.602740	38
BsmtFinType2	2.602740	38
BsmtQual	2.534247	37
BsmtCond	2.534247	37
BsmtFinType1	2.534247	37

```
LandContour    0.000000      0
BldgType       0.000000      0
```

[ ]:
```
for column in categorical_data.columns:
   mode_value = df[column].mode()[0]
   df[column].fillna(mode_value,inplace=True)
sum(df.select_dtypes(exclude="number").isna().sum())
```

[ ]:
```
0
```

3.2 Label Encoding

Many machine learning algorithms require numerical input. Label encoding is necessary to transform these categories into numerical values. All ordinal categorical variables are encoded into numbers based on the mean value of the target variable (`SalePrice`). The smaller value corresponds to the category that has the smaller mean house sale price. That is. the category that has the smallest mean house sale price can be replaced with 0, the next category with 1, and so on. Compared to one-hot encoding, which creates binary columns for each category, this approach reduces the dimensionality of the feature space.

[ ]:
```
cat_vars = df.select_dtypes(exclude=['number']).columns

for var in cat_vars:
    # Calculate mean sale price for each category in the training set
    mean_sale_price = df.groupby(var)['SalePrice'].mean().sort_values()

    # Create a mapping dictionary for encoding
    encoding_map = {category: i for i, category in enumerate(mean_sale_price.
 ↪index)}

    # Apply encoding to both training and testing datasets
    # Define the default value to assign to missing values in the test dataset
    # This value represents the encoding for missing categories that were not␣
 ↪present in the training dataset
    df["encoded_" + var] = df[var].map(encoding_map)
    df["encoded_" + var] = df[var].map(encoding_map)

    # Drop original categorical variables
    df.drop(columns=[var], inplace=True)
```

4 4. Feature Engineering

4.1 Split Data into Training and Testing Sets

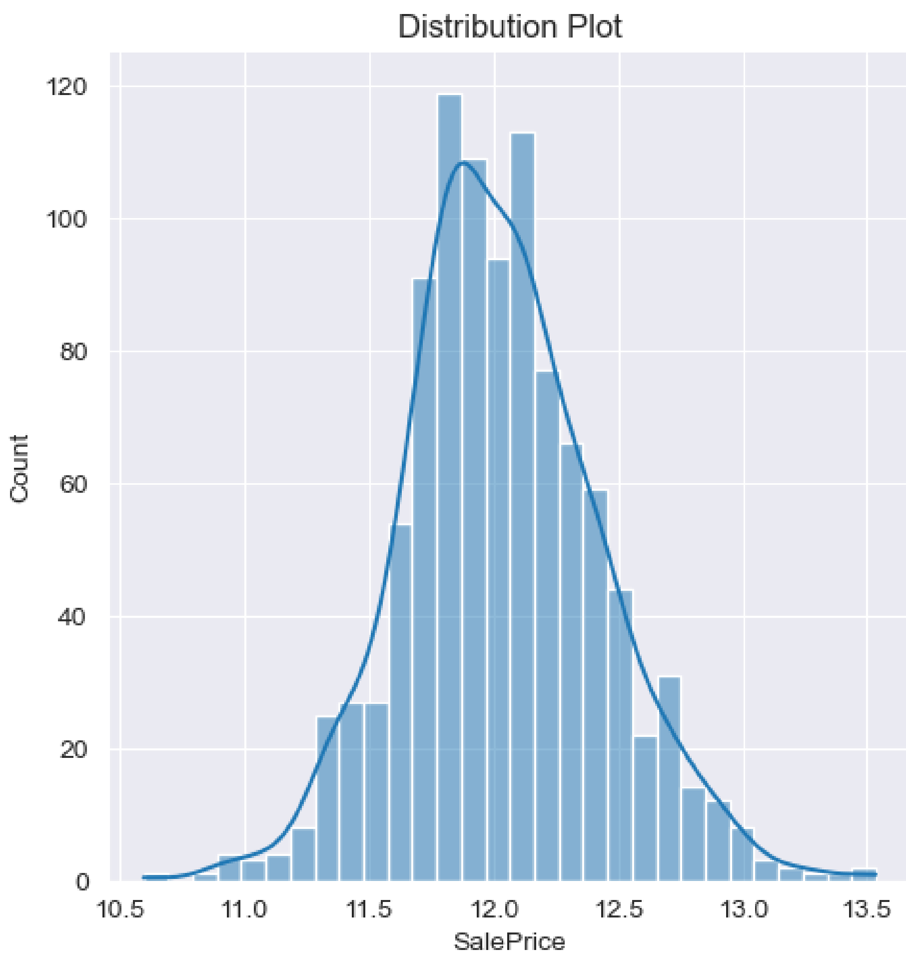
Split the data into training and test sets, with 70% for training and 30% for testing. This step is done after data preprocessing but before transformation and standardization to prevent data leakage.

[ ]:
```python
from sklearn.model_selection import train_test_split

train_df, test_df = train_test_split(df, test_size=0.3, random_state=10)
```

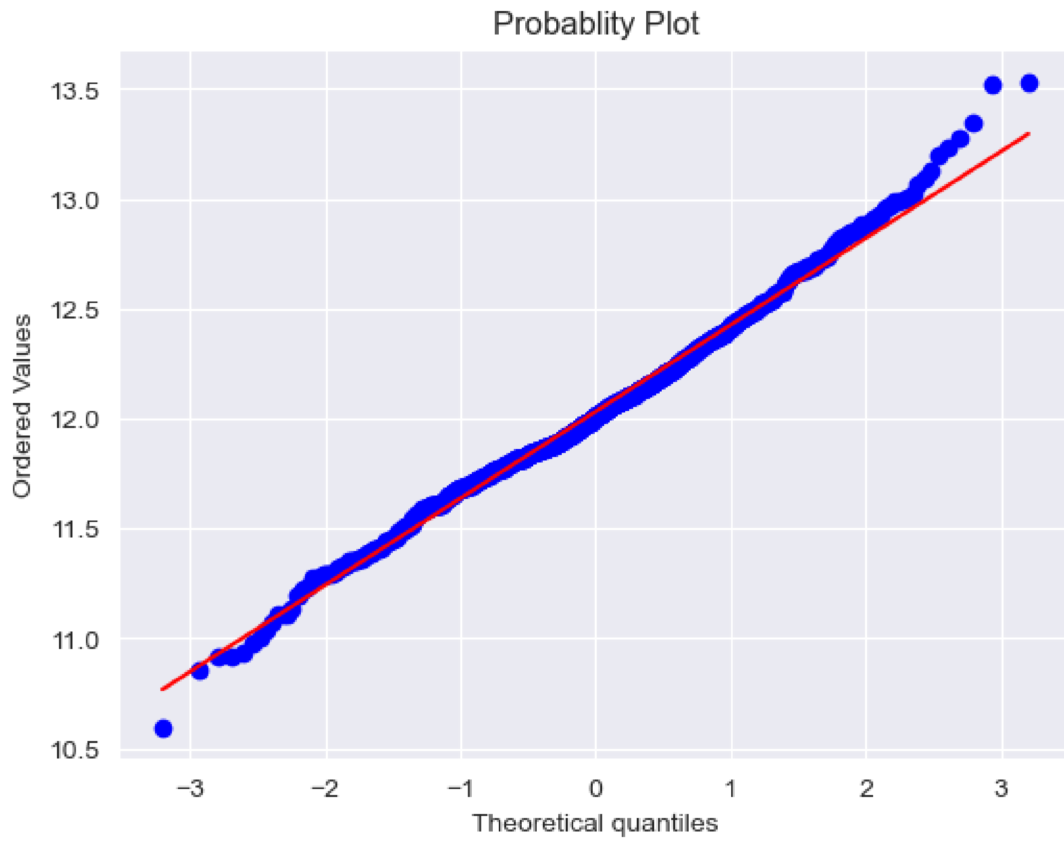
4.2 Fixing skewness with Log Transformation

The target variable(`SalePrice`) is right skewed as shown before. Since normal distribution is crucial for linear regression, a log transformation is performed on skewed data to approximate a normal distribution. From distribution plot and QQ plot, the transformed data is approximately normally distributed.

[ ]:
```python
train_df['SalePrice'] = np.log(train_df['SalePrice'])
test_df['SalePrice'] = np.log(test_df['SalePrice'])

sns.displot(train_df["SalePrice"], kde=True)
plt.title("Distribution Plot")
plt.show()
```



[ ]:
```
stats.probplot(train_df["SalePrice"], dist="norm", plot=plt)
plt.title("Probablity Plot")
plt.show()
```



4.3 Standardization and transformation

$$x_i' = \frac{x_i - \mu}{\sigma}$$

[ ]:
```
from sklearn.preprocessing import PowerTransformer

y_train= train_df['SalePrice']
y_test= test_df['SalePrice']

X_train = train_df.drop('SalePrice', axis=1)
X_test = test_df.drop('SalePrice', axis=1)

# Standardize the features to have mean=0 and variance=1
# StandardScaler can help mitigate overflow issues by scaling the data
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[ ]:
```
from sklearn.preprocessing import PowerTransformer

# Apply the Yeo-Johnson transformation to the standardized features
# PowerTransformer with 'yeo-johnson' method handles non-positive data, but␣
↪overflow may still occur
pt = PowerTransformer(method='yeo-johnson')
X_train_scaled = pd.DataFrame(pt.fit_transform(X_train_scaled), columns=X_train.
↪columns)
X_test_scaled = pd.DataFrame(pt.transform(X_test_scaled), columns=X_test.
↪columns)
```

[ ]:
```
mean_before = X_train.mean()
std_before = X_train.std()

mean_after = X_train_scaled.mean()
std_after = X_train_scaled.std()

summary_df = pd.DataFrame(
    {'Feature': X_train.columns,
    'Mean Before': mean_before.values,
    'Std Before': std_before.values,
    'Mean After standardalization and transformation': mean_after,
    'Std After standardalization and transformation': std_after}
)

summary_df
```

[ ]:
```
                   Feature   Mean Before    Std Before  \
LotArea            LotArea  10681.630137  11305.457364
OverallQual    OverallQual      6.141879      1.372897
OverallCond    OverallCond      5.562622      1.087500
BsmtFinSF1      BsmtFinSF1    447.314090    468.198594
BsmtFinSF2      BsmtFinSF2     40.860078    150.289590
BsmtUnfSF        BsmtUnfSF    575.344423    442.929842
TotalBsmtSF    TotalBsmtSF   1063.518591    443.412565
1stFlrSF          1stFlrSF   1167.607632    393.824408
2ndFlrSF          2ndFlrSF    347.983366    436.440810
LowQualFinSF  LowQualFinSF      7.010763     53.596059
GrLivArea        GrLivArea   1522.601761    533.663862
BsmtFullBath  BsmtFullBath      0.422701      0.517465
BsmtHalfBath  BsmtHalfBath      0.053816      0.230062
FullBath          FullBath      1.573386      0.549237
HalfBath          HalfBath      0.388454      0.505391
BedroomAbvGr  BedroomAbvGr      2.873777      0.816527
```

KitchenAbvGr	KitchenAbvGr	1.044031	0.205265
TotRmsAbvGrd	TotRmsAbvGrd	6.526419	1.598477
Fireplaces	Fireplaces	0.615460	0.644327
GarageCars	GarageCars	1.775930	0.748284
GarageArea	GarageArea	478.349315	215.881679
WoodDeckSF	WoodDeckSF	95.712329	125.395412
OpenPorchSF	OpenPorchSF	47.555773	66.206621
EnclosedPorch	EnclosedPorch	21.959883	61.819216
3SsnPorch	3SsnPorch	3.849315	32.068988
ScreenPorch	ScreenPorch	13.813112	51.399277
PoolArea	PoolArea	2.799413	40.430832
MiscVal	MiscVal	54.387476	585.607282
LotFrontage_missing	LotFrontage_missing	0.173190	0.378597
MasVnrArea_missing	MasVnrArea_missing	0.004892	0.069808
GarageYrBlt_missing	GarageYrBlt_missing	0.053816	0.225765
encoded_LandContour	encoded_LandContour	1.045988	0.439079
encoded_BldgType	encoded_BldgType	3.679061	0.851761
encoded_HouseStyle	encoded_HouseStyle	4.800391	1.385980
encoded_RoofStyle	encoded_RoofStyle	1.587084	1.198166
encoded_ExterQual	encoded_ExterQual	1.409980	0.574702
encoded_ExterCond	encoded_ExterCond	2.868885	0.393919
encoded_BsmtQual	encoded_BsmtQual	1.568493	0.686162
encoded_BsmtCond	encoded_BsmtCond	2.007828	0.269103
encoded_BsmtExposure	encoded_BsmtExposure	0.651663	1.026577
encoded_BsmtFinType1	encoded_BsmtFinType1	3.386497	1.624474
encoded_BsmtFinType2	encoded_BsmtFinType2	3.760274	0.850343
encoded_HeatingQC	encoded_HeatingQC	3.164384	0.951510
encoded_KitchenQual	encoded_KitchenQual	1.524462	0.670410
encoded_GarageFinish	encoded_GarageFinish	0.789628	0.813554
encoded_GarageQual	encoded_GarageQual	1.980431	0.241625
encoded_GarageCond	encoded_GarageCond	3.899217	0.542620

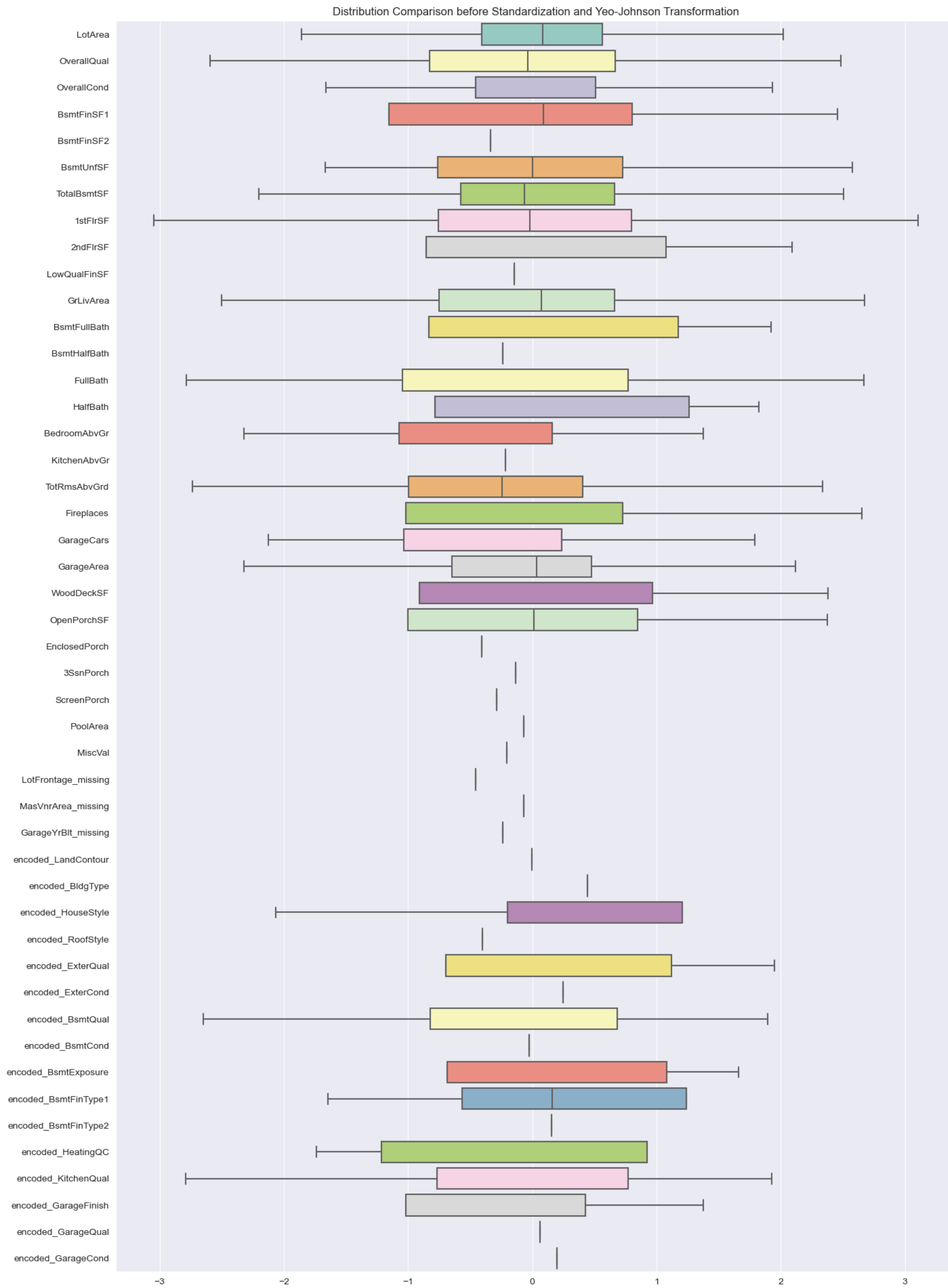
	Mean After standardalization and transformation \
LotArea	7.332686e-18
OverallQual	4.323569e-17
OverallCond	-2.169932e-16
BsmtFinSF1	3.454510e-17
BsmtFinSF2	-3.653307e-16
BsmtUnfSF	1.151503e-17
TotalBsmtSF	2.707662e-17
1stFlrSF	-7.020368e-18
2ndFlrSF	2.752745e-16
LowQualFinSF	8.127875e-16
GrLivArea	1.596896e-17
BsmtFullBath	-2.628904e-17
BsmtHalfBath	1.565854e-15
FullBath	-7.093695e-17

HalfBath	-3.354568e-16
BedroomAbvGr	-1.295441e-16
KitchenAbvGr	1.645781e-17
TotRmsAbvGrd	-1.412221e-17
Fireplaces	8.147429e-17
GarageCars	-7.843259e-17
GarageArea	4.562560e-18
WoodDeckSF	4.169311e-16
OpenPorchSF	6.452764e-17
EnclosedPorch	8.505916e-17
3SsnPorch	2.850731e-15
ScreenPorch	1.735946e-16
PoolArea	9.742153e-16
MiscVal	1.040101e-15
LotFrontage_missing	-2.400776e-17
MasVnrArea_missing	-2.096361e-15
GarageYrBlt_missing	1.124454e-15
encoded_LandContour	-1.198534e-16
encoded_BldgType	3.574006e-17
encoded_HouseStyle	-8.820950e-17
encoded_RoofStyle	-1.435577e-16
encoded_ExterQual	-1.218855e-16
encoded_ExterCond	-5.040543e-17
encoded_BsmtQual	-8.397284e-17
encoded_BsmtCond	1.119117e-16
encoded_BsmtExposure	3.091678e-16
encoded_BsmtFinType1	-1.306304e-17
encoded_BsmtFinType2	-7.089893e-16
encoded_HeatingQC	-9.364112e-17
encoded_KitchenQual	-9.342386e-18
encoded_GarageFinish	-2.096605e-16
encoded_GarageQual	-3.087740e-16
encoded_GarageCond	2.363298e-16

	Std After standardalization and transformation
LotArea	1.00049
OverallQual	1.00049
OverallCond	1.00049
BsmtFinSF1	1.00049
BsmtFinSF2	1.00049
BsmtUnfSF	1.00049
TotalBsmtSF	1.00049
1stFlrSF	1.00049
2ndFlrSF	1.00049
LowQualFinSF	1.00049
GrLivArea	1.00049
BsmtFullBath	1.00049

```
BsmtHalfBath                                                  1.00049
FullBath                                                      1.00049
HalfBath                                                      1.00049
BedroomAbvGr                                                  1.00049
KitchenAbvGr                                                  1.00049
TotRmsAbvGrd                                                  1.00049
Fireplaces                                                    1.00049
GarageCars                                                    1.00049
GarageArea                                                    1.00049
WoodDeckSF                                                    1.00049
OpenPorchSF                                                   1.00049
EnclosedPorch                                                 1.00049
3SsnPorch                                                     1.00049
ScreenPorch                                                   1.00049
PoolArea                                                      1.00049
MiscVal                                                       1.00049
LotFrontage_missing                                           1.00049
MasVnrArea_missing                                            1.00049
GarageYrBlt_missing                                           1.00049
encoded_LandContour                                           1.00049
encoded_BldgType                                              1.00049
encoded_HouseStyle                                            1.00049
encoded_RoofStyle                                             1.00049
encoded_ExterQual                                             1.00049
encoded_ExterCond                                             1.00049
encoded_BsmtQual                                              1.00049
encoded_BsmtCond                                              1.00049
encoded_BsmtExposure                                          1.00049
encoded_BsmtFinType1                                          1.00049
encoded_BsmtFinType2                                          1.00049
encoded_HeatingQC                                             1.00049
encoded_KitchenQual                                           1.00049
encoded_GarageFinish                                          1.00049
encoded_GarageQual                                            1.00049
encoded_GarageCond                                            1.00049
```

[ ]:
```
# Box plot comparing original, standardized, and transformed distributions
plt.figure(figsize=(16, 24))
sns.boxplot(data=pd.DataFrame(X_train_scaled, columns=X_train.columns), orient=
    'h', palette='Set3', showfliers=False)
plt.title('Distribution Comparison before Standardization and Yeo-Johnson
    Transformation')
plt.xticks()
plt.show()
```


Distribution Comparison before Standardization and Yeo-Johnson Transformation
LotArea
OverallQual
OverallCond
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
LotFrontage_missing
MasVnrArea_missing
GarageYrBlt_missing
encoded_LandContour
encoded_BldgType
encoded_HouseStyle
encoded_RoofStyle
encoded_ExterQual
encoded_ExterCond
encoded_BsmtQual
encoded_BsmtCond
encoded_BsmtExposure
encoded_BsmtFinType1
encoded_BsmtFinType2
encoded_HeatingQC
encoded_KitchenQual
encoded_GarageFinish
encoded_GarageQual
encoded_GarageCond
−3
−2
−1
0
1
2
3