BADKUL TECHNOLOGIES

Task 2 - Regression Problem

```
Apply regression techniques to a real-world dataset.
```

I. INTRODUCTION

The real estate market is one of the most dynamic sectors, influenced by various factors such as location, construction quality, size, and neighborhood characteristics.
In this project, we aim to build a **House Price Prediction model** that can accurately estimate the sale price of a property based on its key features.

We used a dataset containing details about residential houses, including both **numerical attributes** (like area, number of rooms, and overall quality) and **categorical attributes** (like neighborhood and house style).

The main objective is to:

- Explore and understand the data through **EDA (Exploratory Data Analysis)**
- Preprocess and engineer useful features
- Train multiple **regression models** (Linear Regression, Random Forest, XGBoost)
- Use **cross-validation** to ensure the model's reliability and avoid overfitting
- Finally, evaluate and identify the best-performing model for predicting house prices.

By the end of this analysis, we aim to find a balance between model interpretability and predictive power, ultimately recommending the model that performs best on unseen data.

II. IMPORTING LIBRARIES

```
!pip install xgboost

Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: xgboost in c:\users\jayes\appdata\
roaming\python\python312\site-packages (3.0.5)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\
site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\
site-packages (from xgboost) (1.13.1)
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score,
KFold
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

III. LOADING DATASET

```python
df = pd.read_csv('house_price_datasets/train.csv')
test_df = pd.read_csv('house_price_datasets/test.csv')

df
```

```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley
LotShape  \
0        1          60       RL         65.0     8450   Pave   NaN
Reg
1        2          20       RL         80.0     9600   Pave   NaN
Reg
2        3          60       RL         68.0    11250   Pave   NaN
IR1
3        4          70       RL         60.0     9550   Pave   NaN
IR1
4        5          60       RL         84.0    14260   Pave   NaN
IR1
...    ...         ...      ...          ...      ...    ...   ...
...
1455  1456          60       RL         62.0     7917   Pave   NaN
Reg
1456  1457          20       RL         85.0    13175   Pave   NaN
Reg
1457  1458          70       RL         66.0     9042   Pave   NaN
Reg
1458  1459          20       RL         68.0     9717   Pave   NaN
Reg
1459  1460          20       RL         75.0     9937   Pave   NaN
Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature
MiscVal  \
0            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1            Lvl    AllPub  ...        0    NaN    NaN         NaN
```

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

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

[1460 rows x 81 columns]
```

test_df

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

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

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

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

[1459 rows x 80 columns]
```

IV. EXPLORATORY DATA ANALYSIS (EDA)

1. Basic Overview of the Data

```
df.shape
```

```
(1460, 81)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     588 non-null    object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 non-null   object 
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
 39  Heating        1460 non-null   object 
 40  HeatingQC      1460 non-null   object 
 41  CentralAir     1460 non-null   object 
 42  Electrical     1459 non-null   object 
 43  1stFlrSF       1460 non-null   int64  
 44  2ndFlrSF       1460 non-null   int64  
```

```
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
df.head()
```

```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
\
0   1          60       RL         65.0     8450   Pave   NaN      Reg

1   2          20       RL         80.0     9600   Pave   NaN      Reg

2   3          60       RL         68.0    11250   Pave   NaN      IR1

3   4          70       RL         60.0     9550   Pave   NaN      IR1
```

```
4   5          60       RL         84.0    14260   Pave   NaN      IR1


  LandContour Utilities  ... 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]

df.describe()

                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
# Summary statistics for numerical columns
df.describe().T
```

```
                count         mean          std     min      25%  \
Id             1460.0   730.500000   421.610009     1.0   365.75

MSSubClass     1460.0    56.897260    42.300571    20.0    20.00

LotFrontage    1201.0    70.049958    24.284752    21.0    59.00

LotArea        1460.0 10516.828082  9981.264932  1300.0  7553.50

OverallQual    1460.0     6.099315     1.382997     1.0     5.00

OverallCond    1460.0     5.575342     1.112799     1.0     5.00

YearBuilt      1460.0  1971.267808    30.202904  1872.0  1954.00

YearRemodAdd   1460.0  1984.865753    20.645407  1950.0  1967.00

MasVnrArea     1452.0   103.685262   181.066207     0.0     0.00

BsmtFinSF1     1460.0   443.639726   456.098091     0.0     0.00

BsmtFinSF2     1460.0    46.549315   161.319273     0.0     0.00

BsmtUnfSF      1460.0   567.240411   441.866955     0.0   223.00

TotalBsmtSF    1460.0  1057.429452   438.705324     0.0   795.75

1stFlrSF       1460.0  1162.626712   386.587738   334.0   882.00

2ndFlrSF       1460.0   346.992466   436.528436     0.0     0.00

LowQualFinSF   1460.0     5.844521    48.623081     0.0     0.00

GrLivArea      1460.0  1515.463699   525.480383   334.0  1129.50

BsmtFullBath   1460.0     0.425342     0.518911     0.0     0.00

BsmtHalfBath   1460.0     0.057534     0.238753     0.0     0.00

FullBath       1460.0     1.565068     0.550916     0.0     1.00
```

HalfBath	1460.0	0.382877	0.502885	0.0	0.00
BedroomAbvGr	1460.0	2.866438	0.815778	0.0	2.00
KitchenAbvGr	1460.0	1.046575	0.220338	0.0	1.00
TotRmsAbvGrd	1460.0	6.517808	1.625393	2.0	5.00
Fireplaces	1460.0	0.613014	0.644666	0.0	0.00
GarageYrBlt	1379.0	1978.506164	24.689725	1900.0	1961.00
GarageCars	1460.0	1.767123	0.747315	0.0	1.00
GarageArea	1460.0	472.980137	213.804841	0.0	334.50
WoodDeckSF	1460.0	94.244521	125.338794	0.0	0.00
OpenPorchSF	1460.0	46.660274	66.256028	0.0	0.00
EnclosedPorch	1460.0	21.954110	61.119149	0.0	0.00
3SsnPorch	1460.0	3.409589	29.317331	0.0	0.00
ScreenPorch	1460.0	15.060959	55.757415	0.0	0.00
PoolArea	1460.0	2.758904	40.177307	0.0	0.00
MiscVal	1460.0	43.489041	496.123024	0.0	0.00
MoSold	1460.0	6.321918	2.703626	1.0	5.00
YrSold	1460.0	2007.815753	1.328095	2006.0	2007.00
SalePrice	1460.0	180921.195890	79442.502883	34900.0	129975.00

	50%	75%	max
Id	730.5	1095.25	1460.0
MSSubClass	50.0	70.00	190.0
LotFrontage	69.0	80.00	313.0
LotArea	9478.5	11601.50	215245.0
OverallQual	6.0	7.00	10.0
OverallCond	5.0	6.00	9.0
YearBuilt	1973.0	2000.00	2010.0
YearRemodAdd	1994.0	2004.00	2010.0
MasVnrArea	0.0	166.00	1600.0
BsmtFinSF1	383.5	712.25	5644.0
BsmtFinSF2	0.0	0.00	1474.0
BsmtUnfSF	477.5	808.00	2336.0
TotalBsmtSF	991.5	1298.25	6110.0

```
1stFlrSF          1087.0     1391.25     4692.0
2ndFlrSF             0.0      728.00     2065.0
LowQualFinSF         0.0        0.00      572.0
GrLivArea         1464.0     1776.75     5642.0
BsmtFullBath         0.0        1.00        3.0
BsmtHalfBath         0.0        0.00        2.0
FullBath             2.0        2.00        3.0
HalfBath             0.0        1.00        2.0
BedroomAbvGr         3.0        3.00        8.0
KitchenAbvGr         1.0        1.00        3.0
TotRmsAbvGrd         6.0        7.00       14.0
Fireplaces           1.0        1.00        3.0
GarageYrBlt       1980.0     2002.00     2010.0
GarageCars           2.0        2.00        4.0
GarageArea         480.0      576.00     1418.0
WoodDeckSF           0.0      168.00      857.0
OpenPorchSF         25.0       68.00      547.0
EnclosedPorch        0.0        0.00      552.0
3SsnPorch            0.0        0.00      508.0
ScreenPorch          0.0        0.00      480.0
PoolArea             0.0        0.00      738.0
MiscVal              0.0        0.00    15500.0
MoSold               6.0        8.00       12.0
YrSold            2008.0     2009.00     2010.0
SalePrice       163000.0   214000.00   755000.0
```

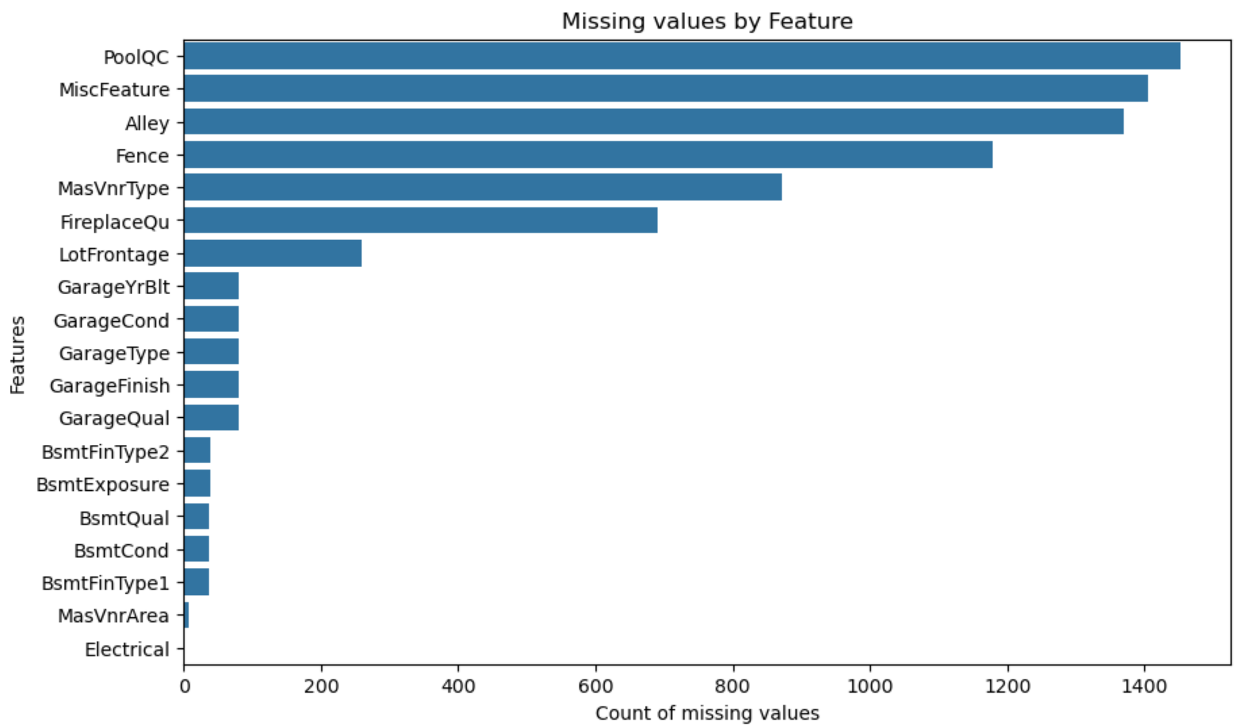
2. Check for Missing Values

```
missing = df.isnull().sum().sort_values(ascending =False)
missing = missing[missing>0]
print("Columns with missing values:\n", missing)
print("Number of columns of missing values:",missing.shape[0])
```

```
Columns with missing values:
 PoolQC          1453
MiscFeature     1406
Alley           1369
Fence           1179
MasVnrType       872
FireplaceQu      690
LotFrontage      259
GarageYrBlt       81
GarageCond        81
GarageType        81
GarageFinish      81
GarageQual        81
BsmtFinType2      38
BsmtExposure      38
BsmtQual          37
BsmtCond          37
```

```
BsmtFinType1      37
MasVnrArea         8
Electrical         1
dtype: int64
Number of columns of missing values: 19
```

```
plt.figure(figsize = (10,6))
sns.barplot(x = missing.values, y = missing.index)
plt.title("Missing values by Feature")
plt.xlabel("Count of missing values")
plt.ylabel("Features")

plt.show()
```



3. Understanding numerical and categorical columns

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
```

```
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     588 non-null    object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
```

```
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

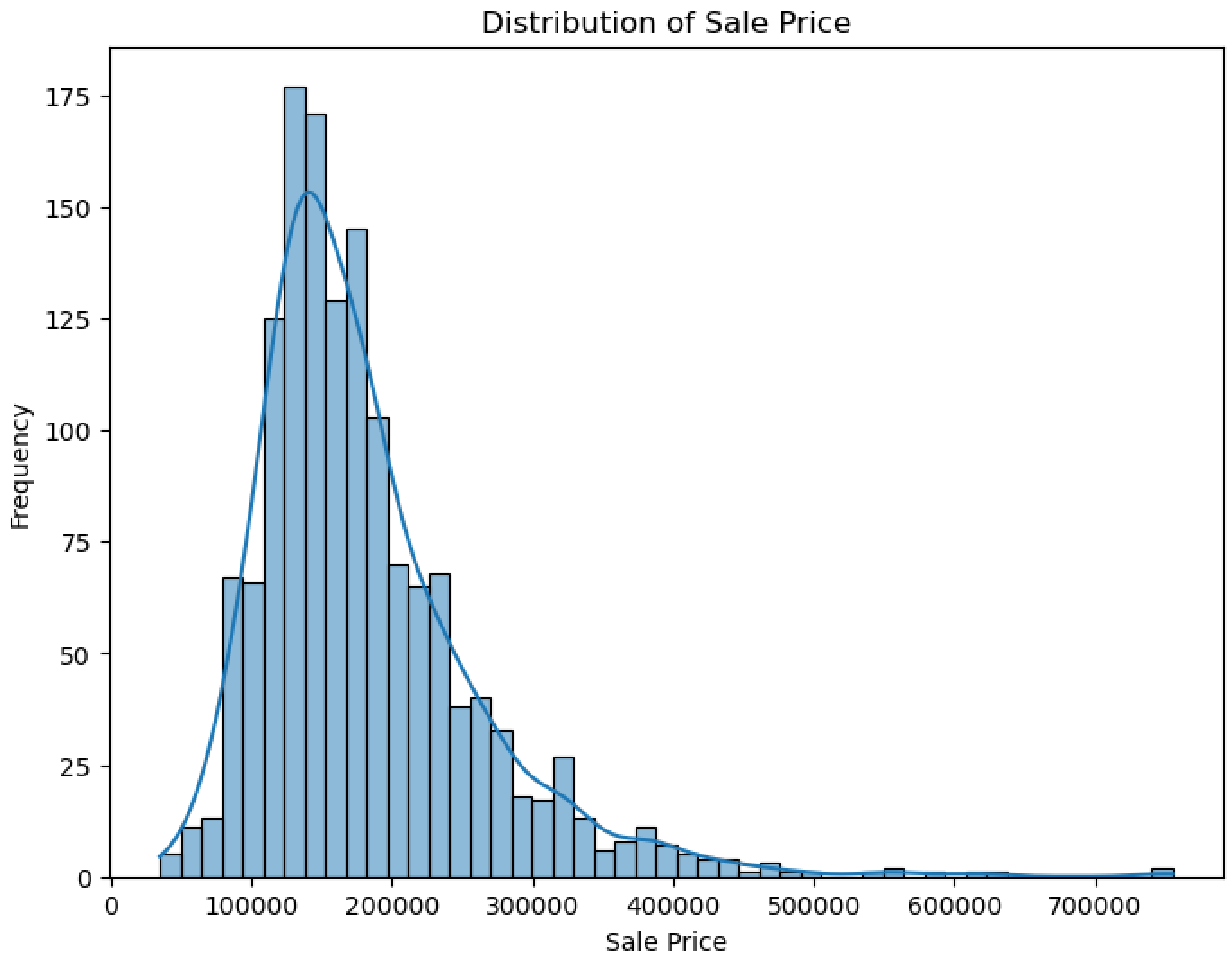
```
num_cols = df.select_dtypes(include=['int64','float64']).columns
cat_cols = df.select_dtypes(include=['object']).columns

print("Numerical Columns:", len(num_cols))
print("Categorical Columns:", len(cat_cols))
```

```
Numerical Columns: 38
Categorical Columns: 43
```

4. Target Variable Distribution

```
plt.figure(figsize=(8,6))
sns.histplot(df['SalePrice'], kde=True)
plt.title("Distribution of Sale Price")
plt.xlabel("Sale Price")
plt.ylabel("Frequency")
plt.show()
```

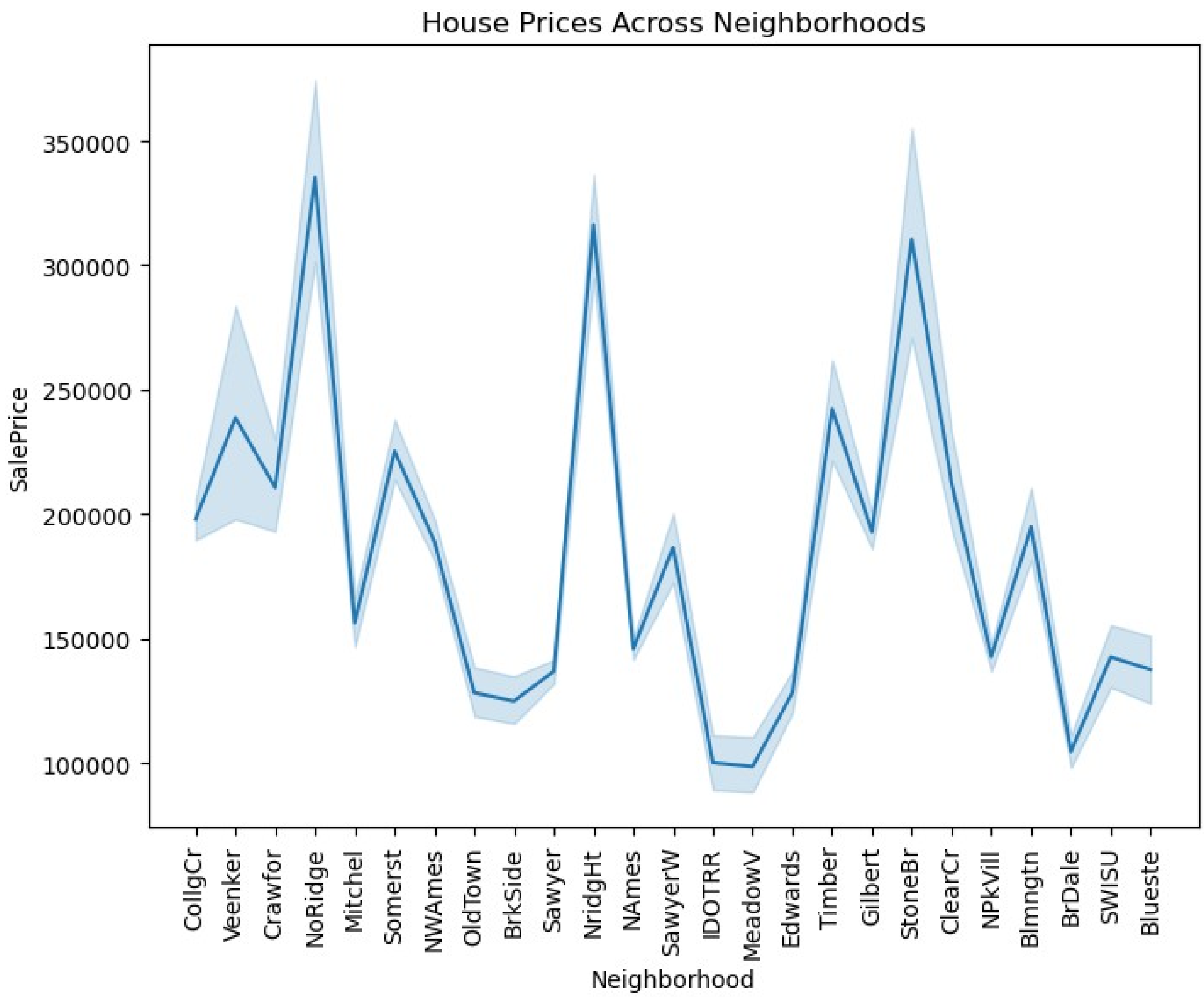
```python
# Skewness check
print("Skewness of SalePrice:", df['SalePrice'].skew())
```




```
Skewness of SalePrice: 1.8828757597682129
```

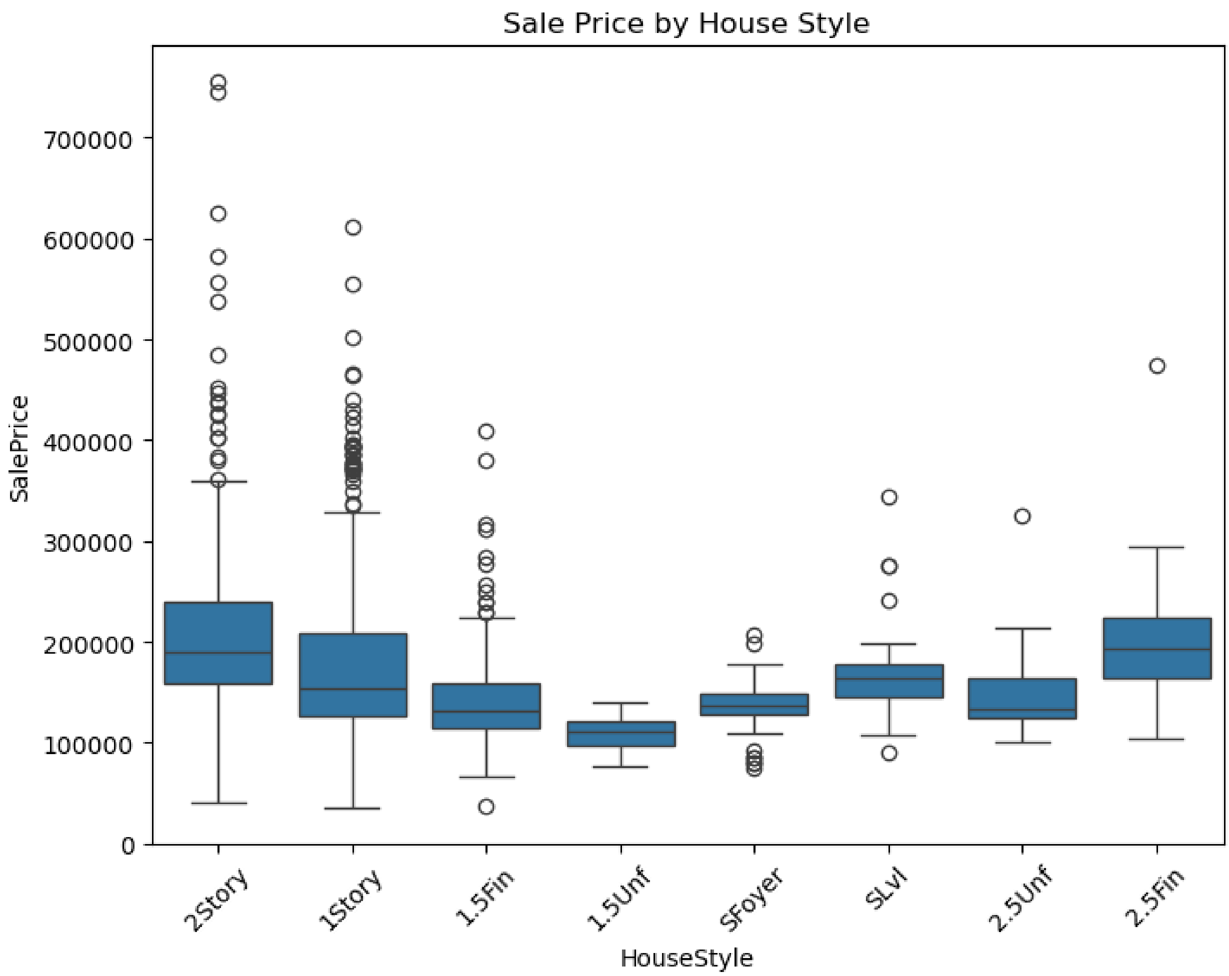
5. Categorical feature Analysis

```python
5.1 Neighborhood vs SalePrice

plt.figure(figsize=(8,6))
sns.lineplot(x='Neighborhood', y='SalePrice', data=df)
plt.xticks(rotation=90)
plt.title("House Prices Across Neighborhoods")
plt.show()
```



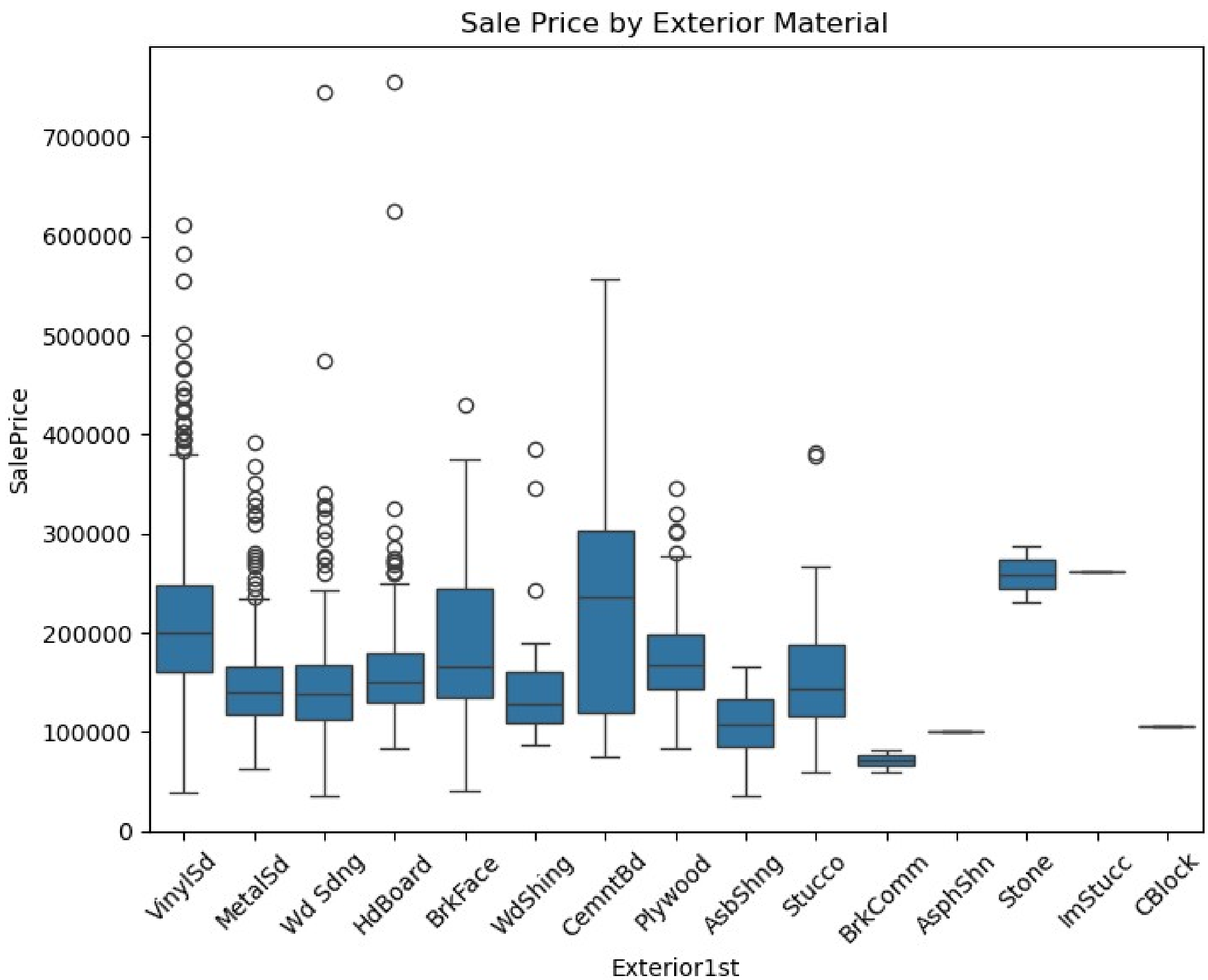
Some neighborhoods consistently have higher median sale prices — often due to better amenities or location.

```
5.2 House Style vs SalePrice

plt.figure(figsize=(8,6))
sns.boxplot(x='HouseStyle', y='SalePrice', data=df)
plt.xticks(rotation=45)
plt.title("Sale Price by House Style")
plt.show()
```



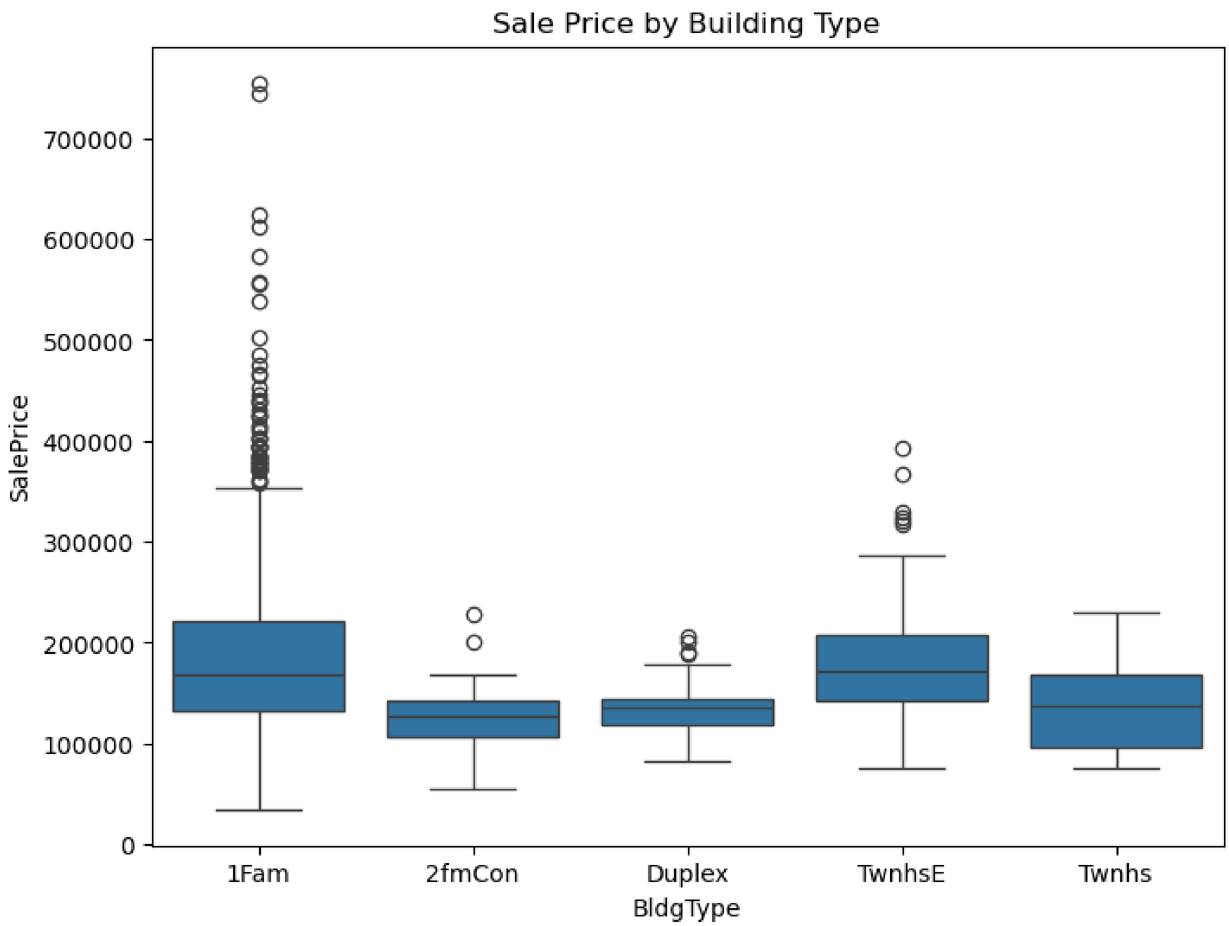
Multi-story and newer house styles tend to have higher average prices.

```
5.3 Exterior Material vs Sale Price

plt.figure(figsize=(8,6))
sns.boxplot(x='Exterior1st', y='SalePrice', data=df)
plt.xticks(rotation=45)
plt.title("Sale Price by Exterior Material")
plt.show()
```



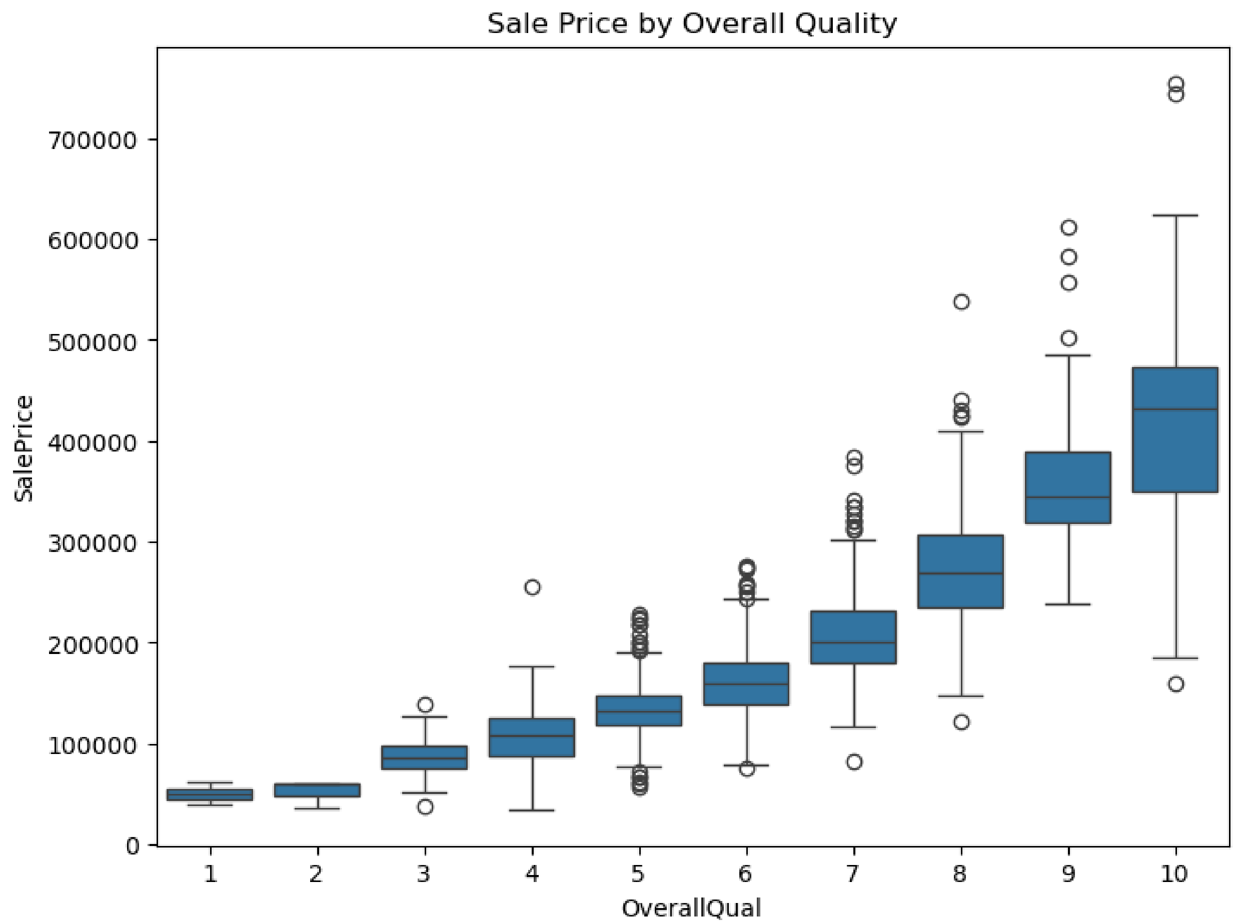
Exterior quality impacts overall value. Premium exteriors like BrickFace or Stone often show higher prices.

```
5.4 Building Type vs Sale Price

plt.figure(figsize=(8,6))
sns.boxplot(x='BldgType', y='SalePrice', data=df)
plt.title("Sale Price by Building Type")
plt.show()
```



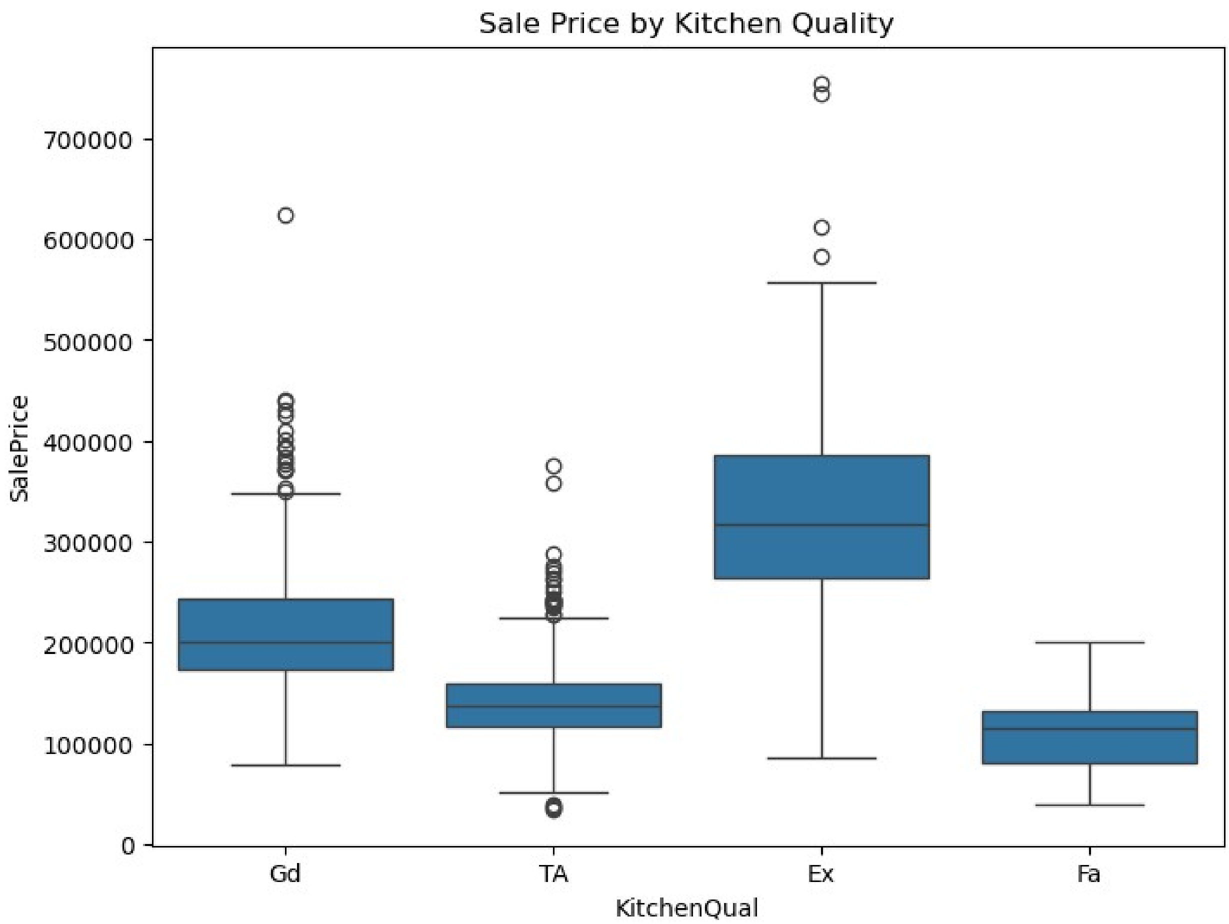
Detached (1Fam) buildings usually cost more than townhouses or duplexes.

```
5.5 Overall Quality vs Sale Price

plt.figure(figsize=(8,6))
sns.boxplot(x='OverallQual', y='SalePrice', data=df)
plt.title("Sale Price by Overall Quality")
plt.show()
```



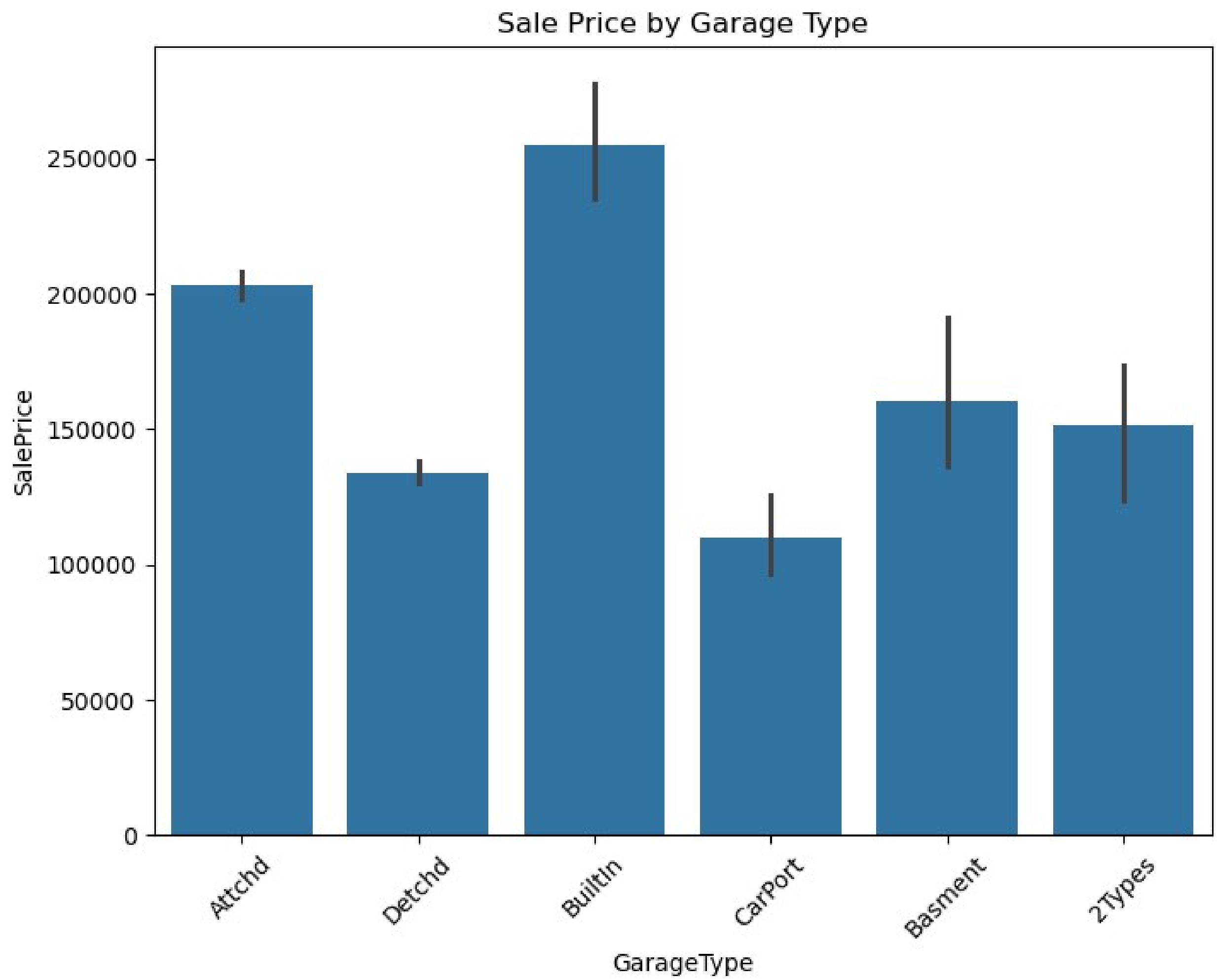
OverallQual is one of the most influential features in predicting price — better overall quality = higher price.

```
5.6 Kitchen Quality and Sale Price

plt.figure(figsize=(8,6))
sns.boxplot(x = "KitchenQual", y="SalePrice", data = df)
plt.title("Sale Price by Kitchen Quality")
plt.show()
```



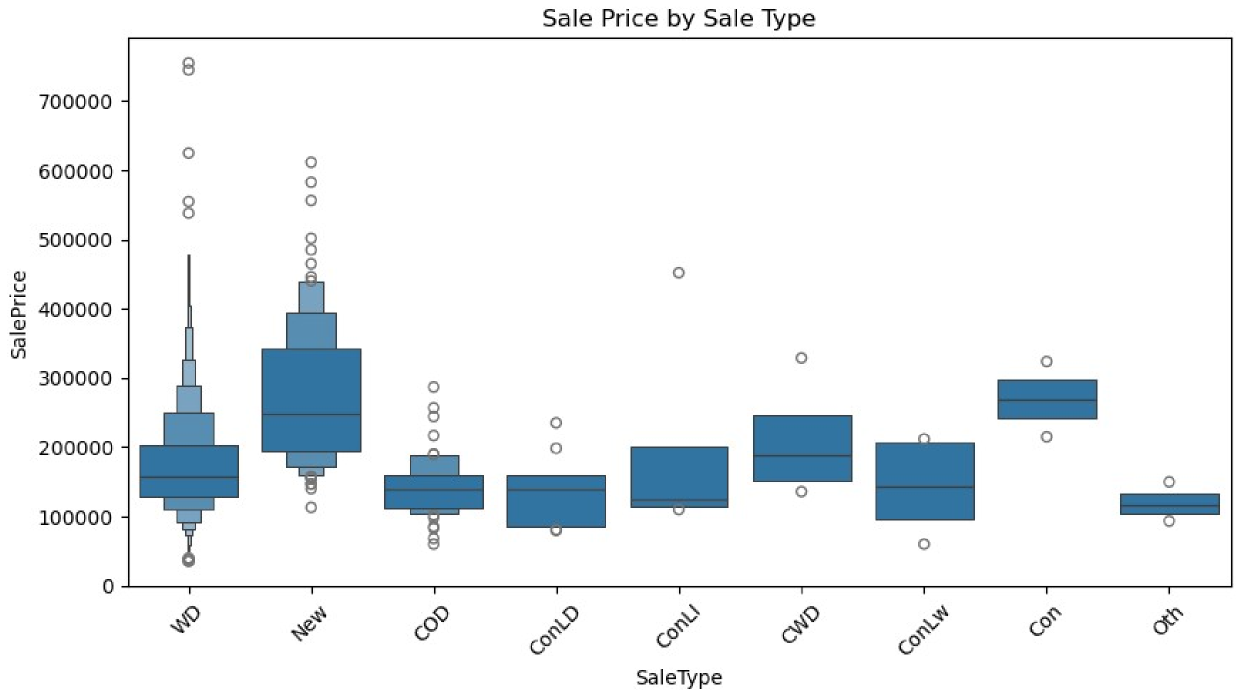
Kitchens rated Excellent (Ex) or Good (Gd) have a noticeably higher sale price.

```
5.7 Garage Type vs Sale Price

plt.figure(figsize=(8,6))
sns.barplot(x='GarageType', y='SalePrice', data=df)
plt.xticks(rotation=45)
plt.title("Sale Price by Garage Type")
plt.show()
```



Attached garages tend to add more value than detached ones.

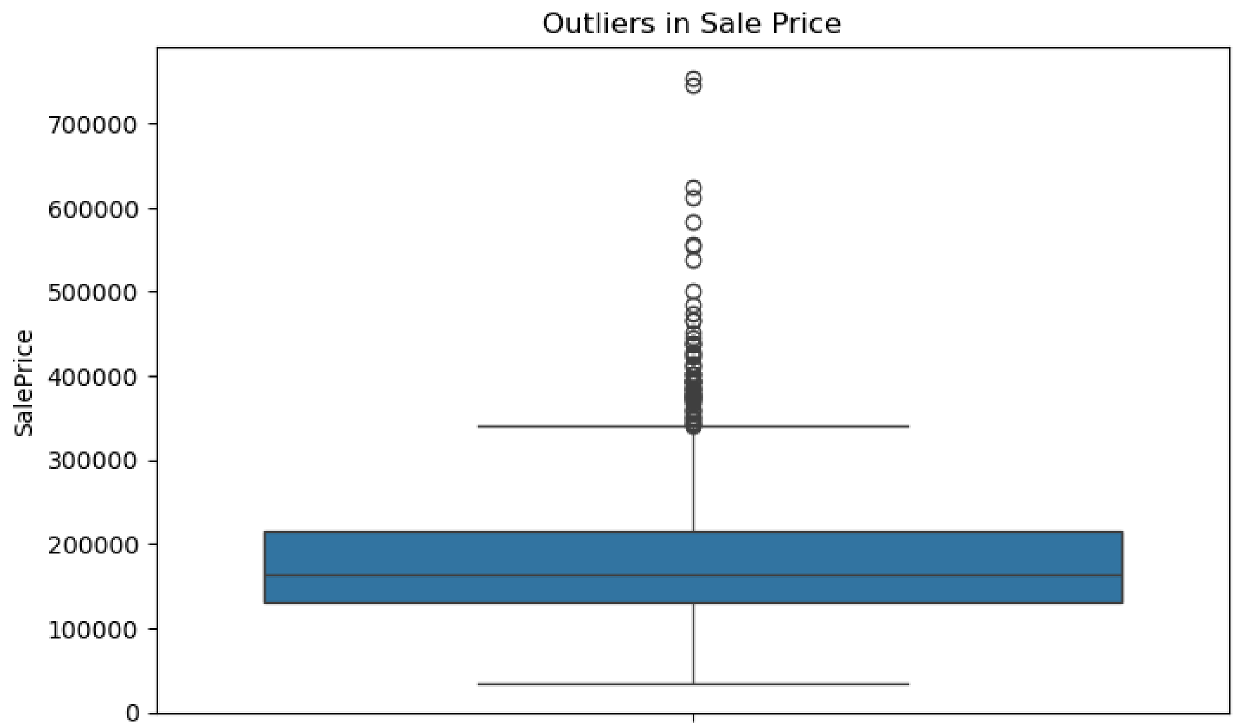
```
5.8 Sale Type vs Sale Price

plt.figure(figsize=(10,5))
sns.boxenplot(x='SaleType', y='SalePrice', data=df)
plt.xticks(rotation=45)
plt.title("Sale Price by Sale Type")
plt.show()
```



"New" or "Warranty Deed" types usually correspond to newer or well-maintained properties with higher sale prices.

6. Outliner Detection

```
plt.figure(figsize=(8,5))
sns.boxplot(df['SalePrice'])
plt.title("Outliers in Sale Price")
plt.show()

# GrLivArea vs SalePrice
sns.scatterplot(x='GrLivArea', y='SalePrice', data=df)
plt.title("Living Area vs Sale Price")
plt.show()
```


Outliers in Sale Price
SalePrice
0
100000
200000
300000
400000
500000
600000
700000


Living Area vs Sale Price
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
5000
GrLivArea

V. DATA PREPROCESSING

1. Missing Values

```
# for training data
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = df[col].fillna(df[col].mode()[0])

    else:
        df[col] = df[col].fillna(df[col].median())

# for testing data
for col in test_df.columns:
    if test_df[col].dtype == 'object':
        test_df[col] = test_df[col].fillna(test_df[col].mode()[0])

    else:
        test_df[col] = test_df[col].fillna(test_df[col].median())
```

2. Encoding Categorical Variables

```
cat_cols = df.select_dtypes(include = ['object']).columns
print(cat_cols)
```

```
Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour',
'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1',
'Condition2',
```

```
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl',
'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond',
'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
'BsmtFinType2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical',
'KitchenQual',
       'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish',
'GarageQual',
       'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType', 'SaleCondition'],
      dtype='object')

le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col].astype(str))
    test_df[col] = le.fit_transform(test_df[col].astype(str))
```

3. Outlier Handling

```
df = df[df['GrLivArea'] < 4500]
df = df[df['SalePrice'] < 700000]
```

VI. FEATURE ENGINEERING

1. Combine Area Features

```
df['TotalArea'] = df['TotalBsmtSF'] + df['1stFlrSF'] + df['2ndFlrSF']
test_df.loc[:, 'TotalArea'] = test_df['TotalBsmtSF'] +
test_df['1stFlrSF'] + test_df['2ndFlrSF']
```

2. Total Bathrooms

```
df['TotalBathrooms'] = df['FullBath'] + (0.5 * df['HalfBath']) +
df['BsmtFullBath'] + (0.5 * df['BsmtHalfBath'])
test_df.loc[:,'TotalBathrooms'] = test_df['FullBath'] + (0.5
*test_df['HalfBath']) + test_df['BsmtFullBath'] + (0.5 *
test_df['BsmtHalfBath'])
```

3. Total Porch Area

```
df['TotalPorchArea'] = df['OpenPorchSF'] + df['EnclosedPorch'] +
df['3SsnPorch'] + df['ScreenPorch']
test_df.loc[:,'TotalPorchArea'] = test_df['OpenPorchSF'] +
test_df['EnclosedPorch'] + test_df['3SsnPorch'] +
test_df['ScreenPorch']
```

4. House Age

```
df['HouseAge'] = df['YrSold'] - df['YearBuilt']
test_df.loc[:,'HouseAge'] = test_df['YrSold'] - test_df['YearBuilt']
```

5. rooms per Area

```
df['RoomsPerArea'] = df['TotRmsAbvGrd'] / df['GrLivArea']
test_df.loc[:,'RoomsPerArea'] = test_df['TotRmsAbvGrd'] /
test_df['GrLivArea']
```

6. Neighbourhood Median Price Encoding

```
neighborhood_map = df.groupby('Neighborhood')
['SalePrice'].mean().to_dict()
df['NeighborhoodPrice'] = df['Neighborhood'].map(neighborhood_map)
test_df.loc[:,'NeighborhoodPrice'] =
test_df['Neighborhood'].map(neighborhood_map)
```

7. Basement Quality Interaction

```
df['BsmtQual_Area'] = df['TotalBsmtSF'] *
df['BsmtQual'].astype('category').cat.codes
test_df.loc[:,'BsmtQual_Area'] = test_df['TotalBsmtSF'] *
test_df['BsmtQual'].astype('category').cat.codes
```

8. Luxury Score

```
df['LuxuryScore'] = (
    df['OverallQual'] * 0.4 +
    df['KitchenQual'].astype('category').cat.codes * 0.2 +
    df['GarageCars'] * 0.2 +
    df['Fireplaces'] * 0.2
)

test_df.loc[:,'LuxuryScore'] = (
    test_df['OverallQual'] * 0.4 +
    test_df['KitchenQual'].astype('category').cat.codes * 0.2 +
    test_df['GarageCars'] * 0.2 +
    test_df['Fireplaces'] * 0.2
)
```

9. Drop Unnecessary Columns

```
df.drop(['Id', 'MiscFeature', 'Alley', 'Fence', 'PoolQC'], axis=1,
inplace=True)
test_df.drop(['Id', 'MiscFeature', 'Alley', 'Fence', 'PoolQC'],
axis=1, inplace=True)
```

10. Splitting dataset into train, test

```
X = df.drop('SalePrice', axis = 1)
y = df['SalePrice']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

11. Standardization of Dataset

```
scaler = StandardScaler()

num_cols = X_train.select_dtypes(include=['int64', 'float64']).columns

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])

X_test
```

```
      MSSubClass  MSZoning  LotFrontage   LotArea  Street  LotShape  \
497    -0.164151         3    -0.452686 -0.142125       1         3
1264    1.500444         2    -1.687151 -0.618248       1         3
411     3.165039         3     1.446491  2.260134       1         3
1048   -0.877549         3     1.446491  1.046302       1         3
1035   -0.877549         3    -0.025371  0.081823       1         0
...          ...       ...          ...       ...     ...       ...
477     0.073648         3     1.683888  0.288174       1         3
675     2.451641         3    -2.161945 -0.784891       1         3
1411   -0.164151         3     0.496902 -0.096959       1         3
650     0.073648         1    -0.215289 -0.235750       1         3
722    -0.877549         3     0.022108 -0.236220       1         3

      LandContour  Utilities  LotConfig  LandSlope  ...  SaleType  \
497             3          0          4          0  ...         8
1264            3          0          4          0  ...         0
411             0          0          4          0  ...         8
1048            3          0          4          0  ...         8
1035            3          0          1          0  ...         8
...           ...        ...        ...        ...  ...       ...
477             3          0          4          0  ...         8
675             3          0          4          0  ...         8
1411            3          0          4          0  ...         8
650             3          0          4          0  ...         8
722             3          0          4          0  ...         8

      SaleCondition  TotalArea  TotalBathrooms  TotalPorchArea
HouseAge  \
497               4   0.377519        1.015628        0.120354
1.547545
1264              0   0.240161        1.015628       -0.180575 -
0.860711
```

```
411               4  -0.591909       -0.281345       -0.820049
0.491871
1048              4  -1.042283       -1.578318       -0.820049
0.425892
1035              4  -2.265294       -1.578318       -0.820049
0.524861
...             ...        ...             ...             ...       .
..
477               4   2.952974        0.367141       -0.030111 -
1.157619
675               4  -0.348891        0.367141       -0.820049 -
0.167925
1411              4  -0.047761       -0.929831       -0.820049
0.755790
650               4  -0.108516        0.367141       -0.820049 -
1.157619
722               4  -1.099075       -1.578318       -0.820049
0.095994

      RoomsPerArea  NeighborhoodPrice  BsmtQual_Area  LuxuryScore
497      -0.128731          -0.989418       0.716134     0.333376
1264     -0.920419          -0.624798       0.486983    -0.300174
411       0.280752           0.190882       0.793088    -0.616950
1048      0.699629          -0.443935      -1.915686    -0.300174
1035      1.707856          -0.946307      -1.915686    -1.567275
...            ...                ...            ...          ...
477      -1.324170           2.332415       1.766126     1.600477
675       0.429085          -0.679502       0.277497     0.333376
1411     -1.092536          -0.624798       0.280062    -0.300174
650      -0.359760           0.755266      -0.525388     0.333376
722       1.550792          -0.624798       0.300583    -1.250500

[292 rows x 83 columns]

X_train

      MSSubClass  MSZoning  LotFrontage   LotArea  Street  LotShape  \
254    -0.877549         3     0.022108 -0.209874       1         3
1065    0.073648         3     0.496902  0.317061       1         3
637     3.165039         4    -0.927480 -0.435703       1         3
1292    0.311448         4    -0.452686 -0.379246       1         3
514    -0.283051         3    -0.690083 -0.003428       1         3
...          ...       ...          ...       ...     ...       ...
1097    1.500444         3    -0.025371 -0.652499       1         3
1132    0.311448         4     0.971696 -0.068730       1         3
1297    2.927240         4    -1.639672 -0.654475       1         3
862    -0.877549         3     0.544382 -0.090184       1         3
1128    0.073648         3    -0.500165  0.109675       1         0

      LandContour  Utilities  LotConfig  LandSlope  ...  SaleType  \
```

```
254              3          0          4          0  ...        8
1065             3          0          4          1  ...        8
637              3          0          4          0  ...        8
1292             3          0          0          0  ...        8
514              3          0          4          0  ...        8
...            ...        ...        ...        ...  ...      ...
1097             3          0          4          0  ...        8
1132             3          0          4          0  ...        8
1297             3          0          4          0  ...        6
862              3          0          0          0  ...        8
1128             3          0          4          0  ...        8

      SaleCondition  TotalArea  TotalBathrooms  TotalPorchArea
HouseAge  \
254               4   0.089596       -0.281345       -0.820049
0.557851
1065              4   1.328456        1.664114       -0.406272 -
0.728752
637               4  -0.478325       -0.281345       -0.820049
0.623830
1292              4   1.064307       -0.281345        1.878908
2.669198
514               4  -1.324923       -1.578318        0.233202
1.481565
...             ...        ...             ...             ...        .
..
1097              4  -0.525871       -0.929831        1.164201 -
0.497823
1132              4   0.868836       -0.281345       -0.368656
2.999096
1297              5  -1.243037        1.015628       -0.406272 -
1.157619
862               4  -0.558890       -0.281345       -0.820049 -
0.332874
1128              4   0.324689        0.367141       -0.368656 -
1.091639

      RoomsPerArea  NeighborhoodPrice  BsmtQual_Area  LuxuryScore
254      -0.841407          -0.624798       1.454891    -0.933725
1065     -1.695791           0.532909      -1.915686     0.333376
637       0.657369          -0.930573       0.164632    -1.250500
1292      0.163171          -0.930573       0.634050    -0.933725
514       2.214787           0.499233       0.054331    -0.933725
...            ...                ...            ...          ...
1097      0.112658           2.232265      -0.079055     0.966926
1132     -1.065031          -0.930573       0.669962    -0.933725
1297      0.195451          -0.946307      -0.980270     0.016601
862       1.167465           0.081586      -0.137198     0.016601
1128     -0.505195           0.190882      -0.467245     0.650151
```

```
[1164 rows x 83 columns]
```

VII. MODEL TRAINING

1. Initialize Models

```
models = {
    "Linear Regression":LinearRegression(),
    "Random Forest":RandomForestRegressor(random_state=42,
n_estimators=200),
    "XGBoost":XGBRegressor(random_state=42, n_estimators=300,
learning_rate=0.1)
}
```

2. Apply Cross Validation

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

cv_results = {}

for name, model in models.items():
    scores = cross_val_score(model, X_train, y_train, cv=kf,
scoring='r2')
    cv_results[name] = {
        'Mean R2': np.mean(scores),
        'Std Dev': np.std(scores)
    }

cv_df = pd.DataFrame(cv_results).T
cv_df
```

	Mean R2	Std Dev
Linear Regression	-1.699289e+19	3.398579e+19
Random Forest	8.923375e-01	1.275951e-02
XGBoost	8.929311e-01	1.678230e-02

```
final_results = {}
from sklearn.metrics import accuracy_score
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    final_results[name] = {
        'R2 Score': r2_score(y_test, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred))
    }

final_df = pd.DataFrame(final_results).T
final_df
```

```
                     R2 Score          RMSE
Linear Regression  0.909693  21770.831375
Random Forest      0.891883  23821.080404
XGBoost            0.892176  23788.797243
```

```
from sklearn.metrics import r2_score, mean_absolute_error
import numpy as np

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(np.mean((y_test - y_pred)**2))

    results[name] = {"R2": r2, "MAE": mae, "RMSE": rmse}

for name, metrics in results.items():
    print(f"{name}: R2={metrics['R2']:.3f}, MAE={metrics['MAE']:.3f},
RMSE={metrics['RMSE']:.3f}")
```

```
Linear Regression: R2=0.910, MAE=16078.265, RMSE=21770.831
Random Forest: R2=0.892, MAE=16117.693, RMSE=23821.080
XGBoost: R2=0.892, MAE=15789.842, RMSE=23788.797
```

VIII. MODEL COMPARISON

```
final_results = {}
from sklearn.metrics import accuracy_score
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    final_results[name] = {
        'R2 Score': r2_score(y_test, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred))
    }

final_df = pd.DataFrame(final_results).T
final_df
```

```
                     R2 Score          RMSE
Linear Regression  0.909693  21770.831375
Random Forest      0.891883  23821.080404
XGBoost            0.892176  23788.797243
```

IX. CONCLUSION

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

test_predictions = best_model.predict(test_df)

submission_temp = pd.read_csv('house_price_datasets/test.csv')
submission_temp.dropna
submission = pd.DataFrame({
    "Id": pd.read_csv('house_price_datasets/test.csv')['Id'],
    "SalePrice": test_predictions
})

submission.to_csv("task2_submission.csv", index=False)
print("□ Submission file saved as 'submission.csv'")

□ Submission file saved as 'submission.csv'
```

Throughout this project, we explored, cleaned, and analyzed the dataset to build a reliable model for predicting house prices.
We experimented with three regression algorithms — **Linear Regression, Random Forest, and XGBoost** — and compared their performance using **cross-validation** and evaluation metrics such as R^2, MAE, and RMSE.

Here's what we observed:

- Linear Regression provided a solid baseline but struggled to capture complex relationships.
- Random Forest improved the accuracy significantly by handling non-linear patterns.
- XGBoost delivered the best overall performance, balancing bias and variance effectively.

Cross-validation ensured that our models were not just memorizing data but genuinely learning underlying patterns.
In real-world applications, this kind of model could assist property sellers, buyers, and real estate firms in making **data-driven pricing decisions**.

Overall, the project demonstrates how combining **data analysis**, **feature engineering**, and **machine learning** can turn raw housing data into valuable market insights.