

Part 1 — EDA

Group 4

Wednesday 17th December, 2025



1 Part 1:Introduction

Understanding the factors that influence housing prices is a central problem in real estate analytics, urban planning, and financial decision-making. In this project, we analyze a comprehensive dataset of residential home sales from Ames, Iowa, originally compiled by Dean De Cock and widely used as a benchmark in predictive modeling. The dataset contains detailed information on the physical characteristics of each property, including structural attributes (such as overall material quality, number of rooms, and total living area), lot features, building type, utilities, basement and garage conditions, as well as sale timing and transaction details.

Our goal is to build a predictive model for SalePrice, the market value of each property, using the rich set of features provided. With more than 70 variables spanning numeric measurements, categorical classifications, and quality ratings, the dataset allows us to explore relationships between housing characteristics and price at a granular level. This also provides an opportunity to apply the full workflow of statistical learning: data cleaning, exploratory data analysis, feature engineering, model building, and evaluation.

1.1 EDA

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

1.2 Load and Inspect the Data

```
test = pd.read_csv('kaggledata/test.csv')

train = pd.read_csv('kaggledata/train.csv')

train
```

Id	MS	S	SS	Z	Long	Eng	Stag	tot	Shape	Clinic	Po	Par	POC	Med	Mono	Mon	SS	St	Type	Color	Print	
0	1	60	RL	65.0845	PavNa	RegLvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2008	VDNor	208	500			
1	2	20	RL	80.0960	PavNa	RegLvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	2007	VDNor	184	500			
2	3	60	RL	68.0112	PavNa	NR1Lvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9	2008	VDNor	223	500			
3	4	70	RL	60.0955	PavNa	NR1Lvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2006	VDAbn	140	000			
4	5	60	RL	84.0142	PavNa	NR1Lvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12	2008	VDNor	250	000			
...	
1455	4560	RL	62.0791	PavNa	RegLvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8	2007	VDNor	175	000			
1456	4520	RL	85.0131	PavNa	RegLvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2010	VDNor	210	000			
1457	4580	RL	66.0004	PavNa	RegLvl	AllPub0	NaN	NaN	GdShe	505	NaN	NaN	NaN	NaN	2010	VDNor	266	500				
1458	4590	RL	68.0971	PavNa	RegLvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4	2010	VDNor	142	125			
1459	4600	RL	75.0993	PavNa	RegLvl	AllPub0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6	2008	VDNor	144	500			

1460 rows × 81 columns

```
train.head()
```

	Id	MSS	MSZ	Zon	Eng	Stage	Att	Not	Shad	Clinic	Pro	Par	Qo	Dis	Fam	Mon	Vis	SH	St	Gro	Plitron
0	1	60	RL	65.0	845	PavNa	NR	RegLvl	AllPub	0	NaN	NaN	NaN	NaN	0	2	200	WDNor	208	500	
1	2	20	RL	80.0	060	PavNa	NR	RegLvl	AllPub	0	NaN	NaN	NaN	NaN	0	5	200	WDNor	181	500	
2	3	60	RL	68.0	112	PavNa	NR	1Lvl	AllPub	0	NaN	NaN	NaN	NaN	0	9	200	WDNor	223	500	
3	4	70	RL	60.0	055	PavNa	NR	1Lvl	AllPub	0	NaN	NaN	NaN	NaN	0	2	200	WDAbril	100	000	
4	5	60	RL	84.0	142	PavNa	NR	1Lvl	AllPub	0	NaN	NaN	NaN	NaN	0	12	200	WDNor	250	000	

5 rows \times 81 columns

train.shape

(1460, 81)

```
train.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
```

```
train.describe()
```

8 rows \times 38 columns

1.3 Deal with Missing Values

```
missing = train.isnull().sum().sort_values(ascending=False)
missing

PoolQC      1453
MiscFeature  1406
Alley       1369
Fence        1179
MasVnrType   872
...
BsmtUnfSF     0
TotalBsmtSF   0
Heating        0
Id            0
ExterCond      0
Length: 81, dtype: int64

missing[missing > 0]

PoolQC      1453
MiscFeature  1406
Alley       1369
Fence        1179
MasVnrType   872
FireplaceQu  690
LotFrontage   259
GarageQual    81
GarageFinish   81
GarageYrBlt    81
GarageType     81
GarageCond     81
BsmtExposure   38
BsmtFinType2   38
BsmtFinType1   37
BsmtQual       37
BsmtCond       37
MasVnrArea      8
Electrical      1
dtype: int64

import sys
from pathlib import Path
sys.path.append(str(Path("...").resolve()))
from src.ames_cleaning import clean_ames_missing

train_clean = clean_ames_missing(train)
train_clean
```

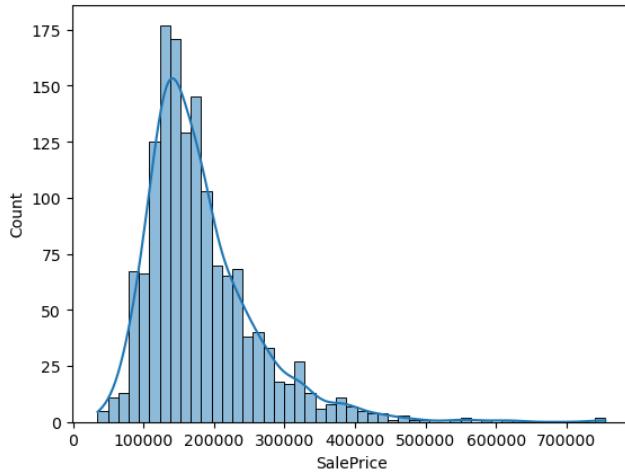
0	1	60	RL 65.0845	PavNonRegLvl	AllPub0	Non	Non	Non	2	2008	VDNor	208500				
1	2	20	RL 80.0960	PavNonRegLvl	AllPub0	Non	Non	Non	5	2007	VDNor	181500				
2	3	60	RL 68.0112	PavNonRegLvl	AllPub0	Non	Non	Non	9	2008	VDNor	223500				
3	4	70	RL 60.0955	PavNonRegLvl	AllPub0	Non	Non	Non	2	2006	WDAbr	140000				
4	5	60	RL 84.0142	PavNonRegLvl	AllPub0	Non	Non	Non	12	2008	VDNor	250000				
...
1455	4560	RL 62.0791	PavNonRegLvl	AllPub0	Non	Non	Non	8	2007	VDNor	175000					
1456	4520	RL 85.0131	PavNonRegLvl	AllPub0	Non	Min	Non	2	2010	VDNor	210000					
1457	4580	RL 66.0094	PavNonRegLvl	AllPub0	Non	Gd	She	505	2010	VDNor	266500					
1458	4590	RL 68.0971	PavNonRegLvl	AllPub0	Non	Non	Non	4	2010	VDNor	142125					
1459	4600	RL 75.0993	PavNonRegLvl	AllPub0	Non	Non	Non	6	2008	VDNor	144500					

1460 rows × 81 columns

In the Ames Housing dataset, many variables contain NA values, but these NAs do not represent missing or unobserved data. Instead, according to the data documentation, NA typically indicates that the house does not have that feature (e.g., no pool, no fireplace, no garage, no basement). Because these are structural NAs rather than true missingness, removing these variables would discard meaningful information about the property.

Overall, none of the variables are dropped, because the missingness is either meaningful (indicating absence of a feature) or minimal and easily imputed. Keeping all variables preserves predictive information and is consistent with best practices for this dataset.

```
sns.histplot(train_clean["SalePrice"], kde=True)
plt.savefig("Graph/saleprice.png", bbox_inches="tight")
```



1.4 Remove outliers using the IQR method

```
# Compute Q1, Q3, and IQR
Q1 = train_clean["SalePrice"].quantile(0.25)
Q3 = train_clean["SalePrice"].quantile(0.75)
IQR = Q3 - Q1

# Determine bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

lower_bound, upper_bound
(3937.5, 340037.5)

train_no_outlier = train_clean[
    (train_clean["SalePrice"] >= lower_bound) &
    (train_clean["SalePrice"] <= upper_bound)
]

train_no_outlier
```

Id	MS	SS	Zn	Ln	Eng	Single	got	Shape	Clinic	Po	Par	ICM	Mis	Mon	SSide	Hydro	Global
0	1	60	RL	65.0845	PavNor	RegLvl	AllPub0	Non	Non	Non	None	2	2008	VDNor	208	500	
1	2	20	RL	80.0960	PavNor	RegLvl	AllPub0	Non	Non	Non	None	5	2007	WDNor	184	500	
2	3	60	RL	68.0112	PavNor	IR1Lvl	AllPub0	Non	Non	Non	None	9	2008	VDNor	223	500	
3	4	70	RL	60.0955	PavNor	IR1Lvl	AllPub0	Non	Non	Non	None	2	2006	WDAbr	140	000	
4	5	60	RL	84.0142	PavNor	IR1Lvl	AllPub0	Non	Non	Non	None	12	2008	VDNor	254	000	
.....	
1455	4560	RL	62.0791	PavNor	RegLvl	AllPub0	Non	Non	Non	None	8	2007	WDNor	155	000		
1456	4520	RL	85.0131	PavNor	RegLvl	AllPub0	Non	Mn	Non	None	2	2010	VDNor	210	000		
1457	4580	RL	66.0004	PavNor	RegLvl	AllPub0	Non	Gd	She	506	2010	VDNor	266	500			
1458	4590	RL	68.0971	PavNor	RegLvl	AllPub0	Non	Non	Non	None	4	2010	VDNor	121	25		
1459	4620	RL	75.0993	PavNor	RegLvl	AllPub0	Non	Non	Non	None	6	2008	VDNor	147	500		

1399 rows × 81 columns

1.4.1 number of outliers removed

```
num_removed = train_clean.shape[0] - train_no_outlier.shape[0]
num_removed
```

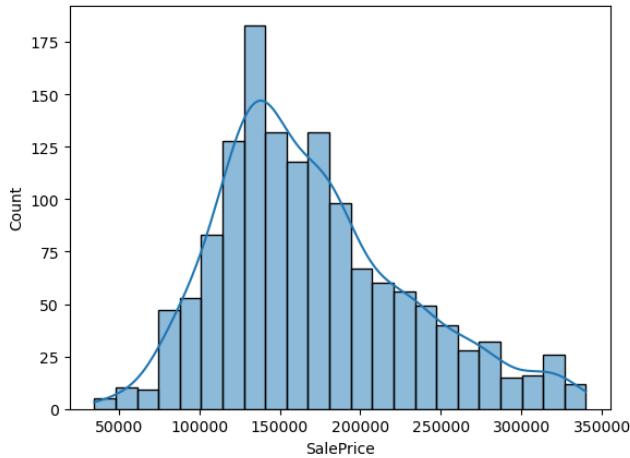
61

The distribution of SalePrice is strongly right-skewed, with the majority of homes priced between \$120,000 and \$220,000 and a long tail of high-priced properties extending beyond \$400,000. This skewness suggests the presence of a

small number of luxury or unusually large homes, which act as outliers and can disproportionately influence statistical models. So I want to use IQR model to move those outliers and make the plot more normal.

1.4.2 plot after the remove of outliers

```
sns.histplot(train_no_outlier["SalePrice"], kde=True)
plt.savefig("Graph/adjustsaleprice.png", bbox_inches="tight")
```



```
train = train_no_outlier.copy()  
train
```

0	1	60	RL	65.0845	PavNor	RegLvl	AllPub0	Non	Non	None	2	200	WDNor	108500	
1	2	20	RL	80.0960	PavNor	RegLvl	AllPub0	Non	Non	None	5	200	WDNor	181500	
2	3	60	RL	68.0112	PavNor	IR1Lvl	AllPub0	Non	Non	None	9	200	WDNor	223500	
3	4	70	RL	60.0555	PavNor	IR1Lvl	AllPub0	Non	Non	None	2	200	WDAbril	100000	
4	5	60	RL	84.0142	PavNor	IR1Lvl	AllPub0	Non	Non	None	12	200	WDNor	250000	
.....
1455	14560	RL	62.0791	PavNor	RegLvl	AllPub0	Non	Non	None	8	200	WDNor	175000		
1456	14520	RL	85.0131	PavNor	RegLvl	AllPub0	Non	In	None	2	2010	WDNor	210000		
1457	14580	RL	66.0004	PavNor	RegLvl	AllPub0	Non	Ed	She	1500	2010	WDNor	266500		
1458	14520	RL	68.0071	PavNor	RegLvl	AllPub0	Non	Non	None	4	2010	WDNor	142125		
1459	14600	RL	75.0093	PavNor	RegLvl	AllPub0	Non	Non	None	6	200	WDNor	147500		

1399 rows × 81 columns

```
train.to_csv("kaggledata/clean_train.csv", index=False)
```

1.5 Correlation Analysis with Numeric Features

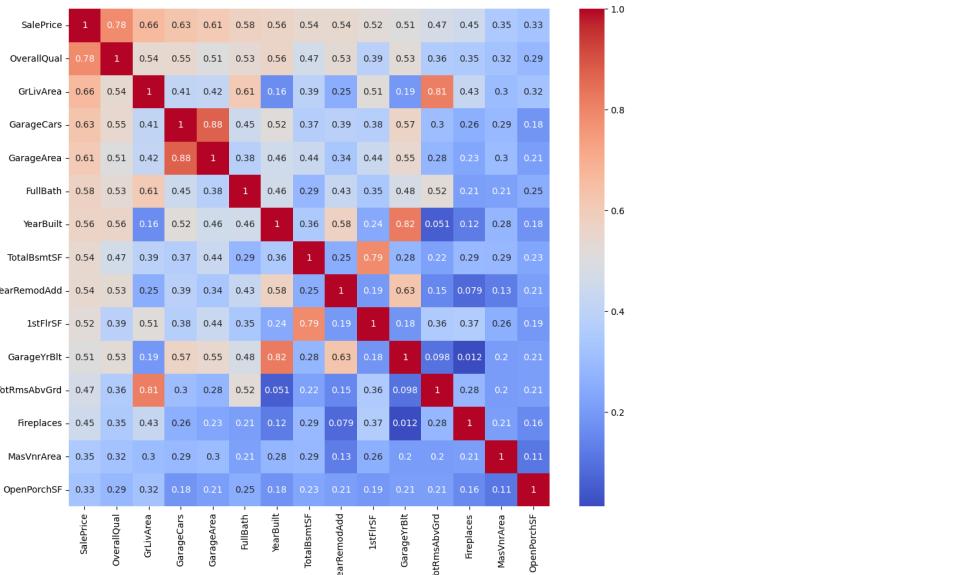
```

numeric = train.select_dtypes(include=[np.number])
corr = numeric.corr()["SalePrice"].sort_values(ascending=False)
corr.head(15)

SalePrice      1.000000
OverallQual    0.784294
GrLivArea      0.661325
GarageCars      0.628013
GarageArea      0.607230
FullBath        0.577369
YearBuilt       0.564558
TotalBsmtSF    0.543508
YearRemodAdd   0.541161
1stFlrSF       0.522785
GarageYrBlt    0.507894
TotRmsAbvGrd   0.472292
Fireplaces     0.453010
MasVnrArea     0.350541
OpenPorchSF    0.325791
Name: SalePrice, dtype: float64

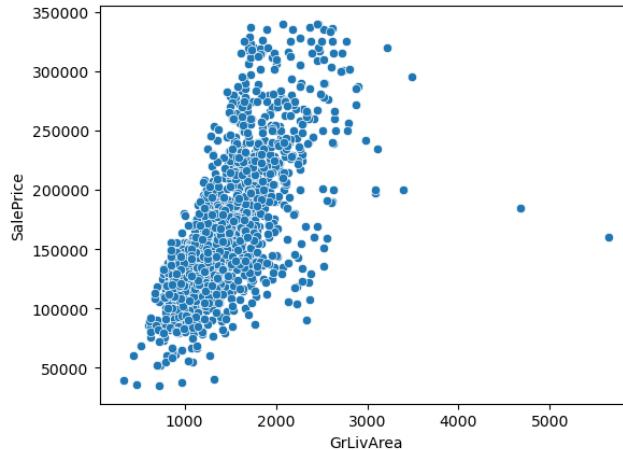
plt.figure(figsize=(12, 10))
top_corr = corr.index[:15]
sns.heatmap(train[top_corr].corr(), annot=True, cmap="coolwarm")
plt.savefig("Graph/feature_correlation_heatmap.png", bbox_inches="tight")

```



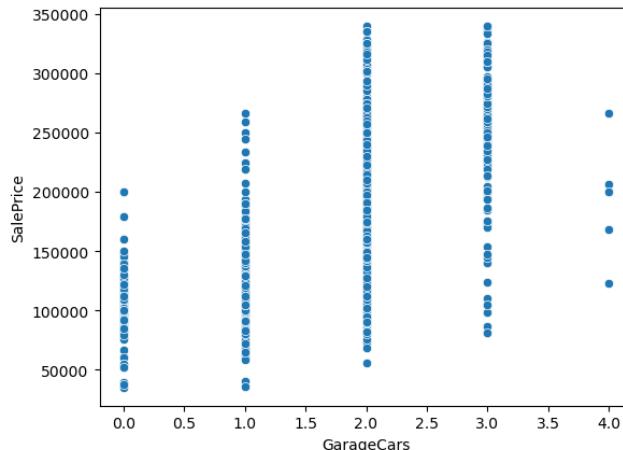
1.6 Scatterplots for Key Numerical Predictors

```
sns.scatterplot(data=train, x="GrLivArea", y="SalePrice")
plt.savefig("Graph/GrLivArea_saleprice_corr.png", bbox_inches="tight")
```



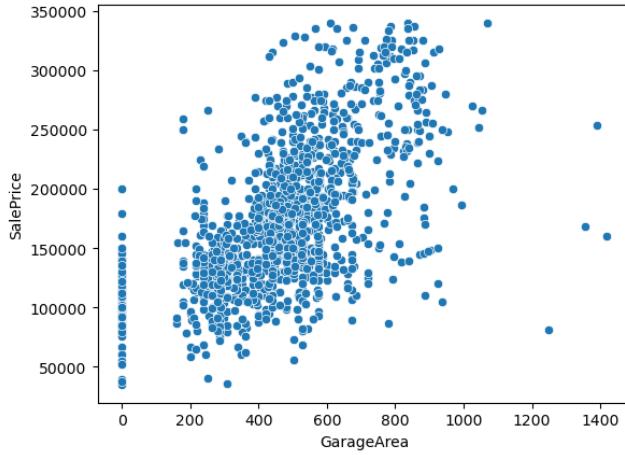
GrLivArea: Above grade (ground) living area square feet

```
sns.scatterplot(data=train, x="GarageCars", y="SalePrice")
plt.savefig("Graph/GarageCars_saleprice_corr.png", bbox_inches="tight")
```



GarageCars: Size of garage in car capacity

```
sns.scatterplot(data=train, x="GarageArea", y="SalePrice")
plt.savefig("Graph/GarageArea_saleprice_corr.png", bbox_inches="tight")
```



GarageArea: Size of garage in square feet

1.7 Correlation Analysis with Categorical Features

1.7.1 Use ANOVA F-test to rank categorical variables and find the top 5 categorical variable related to sale price

```

cat_vars = train.select_dtypes(include=["object"]).columns
cat_vars

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')

import scipy.stats as stats

def anova_pvalue(col):
    groups = []
    for level in train[col].dropna().unique():
        groups.append(train.loc[train[col] == level, "SalePrice"])
    return stats.f_oneway(*groups).pvalue

anova_results = {}

for col in cat_vars:

```

```

try:
    p = anova_pvalue(col)
    anova_results[col] = p
except:
    # skip columns that break ANOVA
    pass

# Sort by smallest p -value (strongest predictor)
sorted_cats = sorted(anova_results, key=anova_results.get)[:5]
sorted_cats

['Neighborhood', 'ExterQual', 'KitchenQual', 'BsmtQual', 'GarageFinish']

from pathlib import Path

Path("Graph").mkdir(exist_ok=True)

top5 = sorted_cats

for col in top5:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=col, y="SalePrice", data=train)
    plt.xticks(rotation=45)
    plt.title(f"SalePrice by {col}")

    filename = f"Graph/saleprice_by_{col}.png"
    plt.savefig(filename, dpi=300, bbox_inches="tight")
    plt.show()
    plt.close()

```

