Nan Value Analysis

In [366…

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import statsmodels .api as sm
from sklearn.preprocessing import StandardScaler
import ISLP
```

In [367…

```python
test_data = pd.read_csv('test.csv')
train_data = pd.read_csv('train.csv')
test_df = pd.DataFrame(test_data)
train_df = pd.DataFrame(train_data)
df = train_df.copy()
```

Electrical (1 missing): one category (“SBrkr”) dominates. Use mode to fill in missing value.

MasVnrType (872 missing) & **MasVnrArea** (8 missing): Eight homes lacked both type and area.
• Two tiny “1 ft²” entries were set to zero and treated as no veneer.
• Three large-area cases were relabeled to “Stone” based on proximity to veneer-type medians.
• Remaining blanks were marked “None” with area zero, aligning all 872 “no veneer” homes.

LotFrontage:
• Nan value indicates no street connected to property so fill with 0 \

Basement Features:
Treat all missing basement features as “no basement” by:

• Imputing missing values → "None".
• Creating a HasBsmt flag (0 if any quality is missing, 1 otherwise).
• Ordinals each feature so that higher codes mean better quality/exposure/finish.
• Removes the raw text columns, and replaced with numeric indicators. • BsmtQual,BsmtCond, BsmtFinType1 - 37 missing
• BsmtExposure,BsmtFinType2 - 38 missing
• Desprpensy arrises from row index 948. Solution is to treat No Exposure and No Basement as the same. \

Garage Features:
• 7 garage related features: GarageType, GarageFinish, GarageQual, GarageCond, GarageCars, GarageArea, GarageYrBlt.
• Missing Count: 81 rows – identical across all garage features.

• All NAs correspond to homes without a garage. Filled with “None” for type/finish/qual, 0 for area/cars \

Time Features:
• Replace GarageYrBlt with GarageAge = YrSold – GarageYrBlt
• Replace YearBuilt with HouseAge = YrSold – YearBuilt
• Replace YearRemodAdd with RemodAge = YrSold – YearRemodAdd
• Cyclical encoding for MoSold
• Dropped the original YearBuilt, GarageYrBlt, YearRemodAdd, and MoSold columns.

Miscellaneous features:
• Fill MiscFeature NAs → "None"
• Create a binary flag HasMisc to indicate whether any miscellaneous feature exists.
• Create a count variable MiscCount to record how many non-None misc features each home has.
• Drop the original MiscFeature column.

Fireplace:
• 690 homes with no fireplace were filled with “None”
• Encoded the fireplace quality with a 0 for “None” and 1-5 for the quality levels.
• Dropped the original FireplaceQu columns.

Pool Features:
PoolQC and PoolArea are two features related to swimming pools in the dataset. The PoolArea feature indicates the area of the pool, while PoolQC provides a quality rating for the pool. However, both features are incredibly sparse: only 7 homes out of 1 460 have a pool.

Steps taken to handle these features:

Created binary flag feature HasPool to indicate whether a house has a pool or not.
Created an ordinal variable PoolQC_ord to represent the quality of the pool on a scale from 0 to 4 (NaN → 0).
Created an interaction feature PoolQualityArea to capture the relationship between pool quality and area.
Created a binary feature PoolHighPriceOutlier to identify the single pool home in the top 0.1 % of sale prices. This house is a potential outlier and may need special handling in the analysis.
Ensured that the cross-validation folds do not end up with all 7 entries of HasPool in the same fold, which could lead to biased results.
Used LOOCV (Leave-One-Out Cross-Validation)—with stratification on HasPool—to evaluate the model performance with and without the pool-related features.
Conclusion:

The only pool-derived feature that consistently improves upon the intercept-only model is the PoolHighPriceOutlier × PoolArea interaction; the outlier flag alone also yields a small gain.

All other pool features—PoolQC, PoolArea, PoolQualityArea, HasPool, and the remaining interaction terms—degrade or fail to improve performance under LOOCV. Drop PoolQC, PoolArea, PoolQualityArea, HasPool, and the unused interactions from the modeling set. Retain only the PoolHighPriceOutlier and PoolArea feature for predictive modeling.

Alley:
• 93.76% of the homes are missing this feature.
• In this case one-hot encoding is identical to indicator feature + ordinal map encoding feature.
• Drop the Alley feature and create a Alley_Grvl and Alley_Pave binary features. \

Fence:

- 1179 missing values indicates no fence
- the combination of a binary presence flag plus one-hot dummies for the specific fence types gives the best out-of-sample performance
- Drop the original Fence feature and create a HasFence binary flag and 4 one-hot dummies for the specific fence types.

In [368…

```
# 2. Electrical
df['Electrical'] = df['Electrical'].fillna(df['Electrical'].mode()[0])

# 3. Masonry Veneer
#   - Map tiny/large anomalies then fill rest
#   (Assume that logic has been applied in place already)
df['MasVnrType'] = df['MasVnrType'].fillna('None')
df['MasVnrArea'] = df['MasVnrArea'].fillna(0)

# 4. Basement features
bsmt_map = {
    'BsmtQual':     {'None':0,'Fa':1,'TA':2,'Gd':3,'Ex':4},
    'BsmtCond':     {'None':0,'Po':1,'Fa':2,'TA':3,'Gd':4,'Ex':5},
    'BsmtExposure': {'None':0,'No':1,'Mn':2,'Av':3,'Gd':4},
    'BsmtFinType1': {'None':0,'Unf':1,'LwQ':2,'Rec':3,'BLQ':4,'ALQ':5,'GL
    'BsmtFinType2': {'None':0,'Unf':1,'LwQ':2,'Rec':3,'BLQ':4,'ALQ':5,'GL
}
for col, mp in bsmt_map.items():
    df[col] = df[col].fillna('None')
    df[col + '_ord'] = df[col].map(mp)
df['HasBsmt'] = (df['BsmtQual'] != 'None').astype(int)
df.drop(columns=list(bsmt_map.keys()), inplace=True)

# 5. Garage features
garage_cat_map = {'None':0,'CarPort':1,'Detchd':2,'2Types':3,'Attchd':4,'
garage_qc_map  = {'None':0,'Po':1,'Fa':2,'TA':3,'Gd':4,'Ex':5}
for col in ['GarageType','GarageFinish','GarageQual','GarageCond']:
    df[col] = df[col].fillna('None')
    mp = garage_cat_map if col=='GarageType' else garage_qc_map
    df[col + '_ord'] = df[col].map(mp)
for col in ['GarageCars','GarageArea']:
    df[col] = df[col].fillna(0)
# Age
df['GarageAge'] = np.where(df['GarageYrBlt'].isna(), 0, df['YrSold'] - df
df.drop(columns=['GarageType','GarageFinish','GarageQual','GarageCond','G
df['HasGarage'] = (df['GarageAge']>0).astype(int)

# 6. Time features
df['HouseAge']  = df['YrSold'] - df['YearBuilt']
df['RemodAge']  = df['YrSold'] - df['YearRemodAdd']
df['MoSold_sin']= np.sin(2*np.pi*df['MoSold']/12)
df['MoSold_cos']= np.cos(2*np.pi*df['MoSold']/12)
df.drop(columns=['YearBuilt','YearRemodAdd','MoSold'], inplace=True)

# 7. MiscFeature
df['HasMisc'] = df['MiscFeature'].notna().astype(int)
df.drop(columns=['MiscFeature'], inplace=True)

# 8. Fireplace
df['FireplaceQu'] = df['FireplaceQu'].fillna('None')
fire_map = {'None':0,'Po':1,'Fa':2,'TA':3,'Gd':4,'Ex':5}
df['FireplaceQu_ord'] = df['FireplaceQu'].map(fire_map)
df.drop(columns=['FireplaceQu'], inplace=True)

# 9. Pool (keep only PoolArea & PoolHighOutlier)
df['PoolArea'] = df['PoolArea'].fillna(0)
thr = df['PoolArea'].quantile(0.95)
df['PoolHighOutlier'] = (df['PoolArea'] > thr).astype(int)
# Drop any other Pool* columns if present
df.drop(columns=[c for c in df.columns if c.startswith('Pool') and c not

# 10. Alley (one-hot)
df['Alley'] = df['Alley'].fillna('None')
d = pd.get_dummies(df['Alley'], prefix='Alley').drop(columns=['Alley_None
df = pd.concat([df, d], axis=1)
df.drop(columns=['Alley'], inplace=True)

# 11. Fence (Has + one-hot)
df['Fence'] = df['Fence'].fillna('None')
df['HasFence'] = (df['Fence']!='None').astype(int)
fd = pd.get_dummies(df['Fence'], prefix='Fence').drop(columns=['Fence_Non
df = pd.concat([df, fd], axis=1)
df.drop(columns=['Fence'], inplace=True)

# 12. LotFrontage
df['LotFrontage'] = df['LotFrontage'].fillna(0)
```

Now that we have handled all the missing values, we can move on to the next step of the data cleaning process: converting categorical features into numerical representations.

I ran leave-one-out cross-validated log-RMSE tests comparing ordinal versus one-hot encoding for each key categorical variable. The results were as follows:

- The folling features are Ordinal-Encoded Columns: LotShape, LandSlope, ExterQual, ExterCond, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, HeatingQC, KitchenQual, Functional, PavedDrive, FireplaceQu, GarageType, GarageFinish, GarageQual, GarageCond.
- The following features are one-hot encoded:MSSubClass,MSZoning,Street,LandContour,Utilities,LotConfig,Neighborho

MasVnrType, Alley_Grvl, Alley_Pave, Fence_GdPrv, Fence_MnPrv, Fence_GdWo, Fence_MnWw

In [373…

```
output_path = '/Users/adityajoshi/Documents/SCHOOL/STA 141C/FinalProject/
df.to_csv(output_path, index=False)
print(f"Cleaned data written to {output_path}")
```

```
Cleaned data written to /Users/adityajoshi/Documents/SCHOOL/STA 141C/Final
Project/data/cleaned_train.csv
```

In [ ]: