Ames Residential Sales Price Prediction

Data Triad Team

The objective of this project is to develop a robust deep learning model on this data, that can help determine from residential information of the Ames, IA neighborhoods the sale price for a house or property.

Data Set

The first step will be to analyze and understan the data we are working with to determine what type of changes should be made in order to make the model more efficient.

In [1]:
```
pip install shap
```

```
Requirement already satisfied: shap in /opt/anaconda3/lib/python3.11/site-pa
ckages (0.45.1)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.11/site-p
ackages (from shap) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.11/site-p
ackages (from shap) (1.11.4)
Requirement already satisfied: scikit-learn in /opt/anaconda3/lib/python3.1
1/site-packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.11/site-
packages (from shap) (2.1.4)
Requirement already satisfied: tqdm>=4.27.0 in /opt/anaconda3/lib/python3.1
1/site-packages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in /opt/anaconda3/lib/python3.
11/site-packages (from shap) (23.1)
Requirement already satisfied: slicer==0.0.8 in /opt/anaconda3/lib/python3.1
1/site-packages (from shap) (0.0.8)
Requirement already satisfied: numba in /opt/anaconda3/lib/python3.11/site-p
ackages (from shap) (0.59.0)
Requirement already satisfied: cloudpickle in /opt/anaconda3/lib/python3.11/
site-packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.43,>=0.42.0dev0 in /opt/anaconda3/
lib/python3.11/site-packages (from numba->shap) (0.42.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/anaconda3/lib/
python3.11/site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/lib/python3.1
1/site-packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /opt/anaconda3/lib/python3.
11/site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in /opt/anaconda3/lib/python3.1
1/site-packages (from scikit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/anaconda3/lib/py
thon3.11/site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.11/sit
e-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import math
sns.set_theme(color_codes=True)


import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv("90_percent_sample.csv", delimiter=",")
```

In [4]:
```
pd.set_option("display.max_columns", None)
df.head()
```

Out[4]:

	MS_SubClass	MS_Zoning	Lot_Frontage	Lot_
0	Two_Story_1945_and_Older	Residential_Medium_Density	0	
1	Two_Story_PUD_1946_and_Newer	Residential_Medium_Density	21	
2	Two_Story_1946_and_Newer	Residential_Low_Density	62	
3	One_Story_1946_and_Newer_All_Styles	Residential_Low_Density	60	
4	One_Story_1945_and_Older	Residential_Medium_Density	50	

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2637 entries, 0 to 2636
Data columns (total 81 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   MS_SubClass         2637 non-null   object
 1   MS_Zoning           2637 non-null   object
 2   Lot_Frontage        2637 non-null   int64 
 3   Lot_Area            2637 non-null   int64 
 4   Street              2637 non-null   object
 5   Alley               2637 non-null   object
 6   Lot_Shape           2637 non-null   object
 7   Land_Contour        2637 non-null   object
 8   Utilities           2637 non-null   object
 9   Lot_Config          2637 non-null   object
 10  Land_Slope          2637 non-null   object
 11  Neighborhood        2637 non-null   object
 12  Condition_1         2637 non-null   object
 13  Condition_2         2637 non-null   object
 14  Bldg_Type           2637 non-null   object
 15  House_Style         2637 non-null   object
 16  Overall_Qual        2637 non-null   object
 17  Overall_Cond        2637 non-null   object
 18  Year_Built          2637 non-null   int64 
 19  Year_Remod_Add      2637 non-null   int64 
 20  Roof_Style          2637 non-null   object
 21  Roof_Matl           2637 non-null   object
 22  Exterior_1st        2637 non-null   object
 23  Exterior_2nd        2637 non-null   object
 24  Mas_Vnr_Type        1045 non-null   object
 25  Mas_Vnr_Area        2637 non-null   int64 
 26  Exter_Qual          2637 non-null   object
 27  Exter_Cond          2637 non-null   object
 28  Foundation          2637 non-null   object
 29  Bsmt_Qual           2637 non-null   object
 30  Bsmt_Cond           2637 non-null   object
 31  Bsmt_Exposure       2637 non-null   object
 32  BsmtFin_Type_1      2637 non-null   object
 33  BsmtFin_SF_1        2637 non-null   int64 
 34  BsmtFin_Type_2      2637 non-null   object
 35  BsmtFin_SF_2        2637 non-null   int64 
 36  Bsmt_Unf_SF         2637 non-null   int64 
 37  Total_Bsmt_SF       2637 non-null   int64 
 38  Heating             2637 non-null   object
 39  Heating_QC          2637 non-null   object
 40  Central_Air         2637 non-null   object
 41  Electrical          2637 non-null   object
 42  First_Flr_SF        2637 non-null   int64 
 43  Second_Flr_SF       2637 non-null   int64 
 44  Low_Qual_Fin_SF     2637 non-null   int64 
 45  Gr_Liv_Area         2637 non-null   int64 
 46  Bsmt_Full_Bath      2637 non-null   int64 
 47  Bsmt_Half_Bath      2637 non-null   int64 
 48  Full_Bath           2637 non-null   int64 
 49  Half_Bath           2637 non-null   int64 
 50  Bedroom_AbvGr       2637 non-null   int64 
```

```
 51  Kitchen_AbvGr       2637 non-null   int64
 52  Kitchen_Qual        2637 non-null   object
 53  TotRms_AbvGrd       2637 non-null   int64
 54  Functional          2637 non-null   object
 55  Fireplaces          2637 non-null   int64
 56  Fireplace_Qu        2637 non-null   object
 57  Garage_Type         2637 non-null   object
 58  Garage_Finish       2637 non-null   object
 59  Garage_Cars         2637 non-null   int64
 60  Garage_Area         2637 non-null   int64
 61  Garage_Qual         2637 non-null   object
 62  Garage_Cond         2637 non-null   object
 63  Paved_Drive         2637 non-null   object
 64  Wood_Deck_SF        2637 non-null   int64
 65  Open_Porch_SF       2637 non-null   int64
 66  Enclosed_Porch      2637 non-null   int64
 67  Three_season_porch  2637 non-null   int64
 68  Screen_Porch        2637 non-null   int64
 69  Pool_Area           2637 non-null   int64
 70  Pool_QC             2637 non-null   object
 71  Fence               2637 non-null   object
 72  Misc_Feature        97 non-null     object
 73  Misc_Val            2637 non-null   int64
 74  Mo_Sold             2637 non-null   int64
 75  Year_Sold           2637 non-null   int64
 76  Sale_Type           2637 non-null   object
 77  Sale_Condition      2637 non-null   object
 78  Sale_Price          2637 non-null   int64
 79  Longitude           2637 non-null   float64
 80  Latitude            2637 non-null   float64
dtypes: float64(2), int64(33), object(46)
memory usage: 1.6+ MB
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
MS_SubClass       0
MS_Zoning         0
Lot_Frontage      0
Lot_Area          0
Street            0
                 ..
Sale_Type         0
Sale_Condition    0
Sale_Price        0
Longitude         0
Latitude          0
Length: 81, dtype: int64
```

Exploratory Data Analysis

In [7]:

```
#Change data on Overall Quality and Overall Condition column to avoid incons

df['Overall_Qual'] = df['Overall_Qual'].replace({
    'Very_Poor': 1,
    'Poor': 2,
```

```
    'Fair': 3,
    'Below_Average': 4,
    'Average': 5,
    'Above_Average': 6,
    'Good': 7,
    'Very_Good': 8,
    'Excellent': 9,
    'Very_Excellent': 10
})
df['Overall_Cond'] = df['Overall_Cond'].replace({
    'Very_Poor': 1,
    'Poor': 2,
    'Fair': 3,
    'Below_Average': 4,
    'Average': 5,
    'Above_Average': 6,
    'Good': 7,
    'Very_Good': 8,
    'Excellent': 9,
    'Very_Excellent': 10
})
```

In [8]:

```
#Categorical Data Selection
df_categorical = df[['MS_SubClass', 'MS_Zoning', 'Street', 'Alley', 'Lot_Sha
                    'Land_Contour', 'Utilities', 'Lot_Config', 'Land_Slope',
                    'Neighborhood', 'Condition_1', 'Condition_2', 'Bldg_Type
                    'House_Style', 'Roof_Style', 'Roof_Matl', 'Exterior_1st'
                    'Exterior_2nd', 'Mas_Vnr_Type', 'Exter_Qual', 'Exter_Cor
                    'Foundation', 'Bsmt_Qual', 'Bsmt_Cond', 'Bsmt_Exposure',
                    'BsmtFin_Type_1', 'BsmtFin_Type_2', 'Heating', 'Heating_
                    'Central_Air', 'Electrical', 'Kitchen_Qual', 'Functional
                    'Fireplace_Qu', 'Garage_Type', 'Garage_Finish', 'Garage_
                    'Garage_Cond', 'Paved_Drive', 'Pool_QC', 'Fence', 'Misc_
                    'Sale_Type', 'Sale_Condition', ]]
df_categorical.head()
```

Out[8]:

	MS_SubClass	MS_Zoning	Street	
0	Two_Story_1945_and_Older	Residential_Medium_Density	Pave	G
1	Two_Story_PUD_1946_and_Newer	Residential_Medium_Density	Pave	No_Alley_Ac
2	Two_Story_1946_and_Newer	Residential_Low_Density	Pave	No_Alley_Ac
3	One_Story_1946_and_Newer_All_Styles	Residential_Low_Density	Pave	No_Alley_Ac
4	One_Story_1945_and_Older	Residential_Medium_Density	Pave	No_Alley_Ac

In [9]:

```
#Numerical Data Selection
df_numerical = df[[
    "Lot_Frontage", "Lot_Area", 'Overall_Qual' , 'Overall_Cond', 'Year_Built
    'Year_Remod_Add', 'Mas_Vnr_Area', 'BsmtFin_SF_1', 'BsmtFin_SF_2', 'Bsmt_
    'Total_Bsmt_SF', 'First_Flr_SF', 'Second_Flr_SF', 'Low_Qual_Fin_SF',
    'Gr_Liv_Area', 'Bsmt_Full_Bath', 'Bsmt_Half_Bath', 'Full_Bath',
    'Half_Bath', 'Bedroom_AbvGr', 'Kitchen_AbvGr', 'TotRms_AbvGrd',
    'Fireplaces', 'Garage_Cars', 'Garage_Area', 'Wood_Deck_SF',
```

```
    'Open_Porch_SF', 'Enclosed_Porch', 'Three_season_porch', 'Screen_Porch',
    'Pool_Area', 'Misc_Val', 'Sale_Price', 'Longitude', 'Latitude', 'Mo_Solc
]]

df_numerical.head()
```

Out[9]:

	Lot_Frontage	Lot_Area	Overall_Qual	Overall_Cond	Year_Built	Year_Remod_Add
0	0	5100	8	7	1925	1996
1	21	1890	6	7	1972	1972
2	62	7162	7	5	2003	2004
3	60	8070	4	5	1994	1995
4	50	7000	6	8	1926	1998

In [10]:

```
from sklearn import preprocessing

# List of categorical columns
categorical_columns = [
    'MS_SubClass', 'MS_Zoning', 'Street', 'Alley', 'Lot_Shape', 'Land_Contou
    'Utilities', 'Lot_Config', 'Land_Slope', 'Neighborhood', 'Condition_1',
    'Condition_2', 'Bldg_Type', 'House_Style', 'Roof_Style', 'Roof_Matl',
    'Exterior_1st', 'Exterior_2nd', 'Mas_Vnr_Type', 'Exter_Qual', 'Exter_Cor
    'Foundation', 'Bsmt_Qual', 'Bsmt_Cond', 'Bsmt_Exposure', 'BsmtFin_Type_1
    'BsmtFin_Type_2', 'Heating', 'Heating_QC', 'Central_Air', 'Electrical',
    'Kitchen_Qual', 'Functional', 'Fireplace_Qu', 'Garage_Type', 'Garage_Fir
    'Garage_Qual', 'Garage_Cond', 'Paved_Drive', 'Pool_QC', 'Fence',
    'Misc_Feature', 'Sale_Type', 'Sale_Condition'
]

# Apply LabelEncoder to each column and show before/after values
for col in categorical_columns:
    # Display original values
    print(f"Original values in '{col}':", df[col].unique())

    # Fit and transform the column
    label_encoder = preprocessing.LabelEncoder()
    df[col] = label_encoder.fit_transform(df[col])

    # Display encoded values
    print(f"Encoded values in '{col}':", df[col].unique())
    print('-' * 40)
```

```
Original values in 'MS_SubClass': ['Two_Story_1945_and_Older' 'Two_Story_PUD
_1946_and_Newer'
 'Two_Story_1946_and_Newer' 'One_Story_1946_and_Newer_All_Styles'
 'One_Story_1945_and_Older' 'One_Story_PUD_1946_and_Newer'
 'Duplex_All_Styles_and_Ages' 'Two_Family_conversion_All_Styles_and_Ages'
 'Split_or_Multilevel' 'Split_Foyer'
 'One_and_Half_Story_Finished_All_Ages'
 'One_and_Half_Story_Unfinished_All_Ages' 'Two_and_Half_Story_All_Ages'
 'PUD_Multilevel_Split_Level_Foyer'
 'One_Story_with_Finished_Attic_All_Ages'
 'One_and_Half_Story_PUD_All_Ages']
Encoded values in 'MS_SubClass': [12 14 13  2  1  3  0 11 10  9  5  7 15  8
4  6]
----------------------------------------
Original values in 'MS_Zoning': ['Residential_Medium_Density' 'Residential_L
ow_Density'
 'Residential_High_Density' 'Floating_Village_Residential' 'C_all' 'I_all'
 'A_agr']
Encoded values in 'MS_Zoning': [6 5 4 2 1 3 0]
----------------------------------------
Original values in 'Street': ['Pave' 'Grvl']
Encoded values in 'Street': [1 0]
----------------------------------------
Original values in 'Alley': ['Gravel' 'No_Alley_Access' 'Paved']
Encoded values in 'Alley': [0 1 2]
----------------------------------------
Original values in 'Lot_Shape': ['Regular' 'Slightly_Irregular' 'Irregular'
'Moderately_Irregular']
Encoded values in 'Lot_Shape': [2 3 0 1]
----------------------------------------
Original values in 'Land_Contour': ['Lvl' 'Bnk' 'HLS' 'Low']
Encoded values in 'Land_Contour': [3 0 1 2]
----------------------------------------
Original values in 'Utilities': ['AllPub' 'NoSewr']
Encoded values in 'Utilities': [0 1]
----------------------------------------
Original values in 'Lot_Config': ['Inside' 'FR2' 'CulDSac' 'Corner' 'FR3']
Encoded values in 'Lot_Config': [4 2 1 0 3]
----------------------------------------
Original values in 'Land_Slope': ['Gtl' 'Mod' 'Sev']
Encoded values in 'Land_Slope': [0 1 2]
----------------------------------------
Original values in 'Neighborhood': ['Old_Town' 'Briardale' 'Edwards' 'Colleg
e_Creek' 'Iowa_DOT_and_Rail_Road'
 'Gilbert' 'Blueste' 'Northpark_Villa' 'Sawyer' 'Northridge_Heights'
 'Northwest_Ames' 'Meadow_Village' 'North_Ames' 'Sawyer_West' 'Timberland'
 'Bloomington_Heights' 'Somerset' 'Brookside' 'Stone_Brook' 'Clear_Creek'
 'Crawford' 'Mitchell' 'Northridge'
 'South_and_West_of_Iowa_State_University' 'Veenker' 'Greens' 'Landmark'
 'Green_Hills']
Encoded values in 'Neighborhood': [20  2  7  5 11  8  1 16 21 18 19 13 15 22
26  0 23  3 25  4  6 14 17 24
 27 10 12  9]
----------------------------------------
Original values in 'Condition_1': ['Norm' 'PosA' 'Artery' 'Feedr' 'RRAn' 'Po
sN' 'RRAe' 'RRNn' 'RRNe']
```

```
Encoded values in 'Condition_1': [2 3 0 1 6 4 5 8 7]
----------------------------------------
Original values in 'Condition_2': ['Norm' 'Artery' 'PosA' 'Feedr' 'RRNn' 'Po
sN' 'RRAe' 'RRAn']
Encoded values in 'Condition_2': [2 0 3 1 7 4 5 6]
----------------------------------------
Original values in 'Bldg_Type': ['OneFam' 'Twnhs' 'TwnhsE' 'Duplex' 'TwoFmCo
n']
Encoded values in 'Bldg_Type': [1 2 3 0 4]
----------------------------------------
Original values in 'House_Style': ['Two_Story' 'One_Story' 'SLvl' 'SFoyer'
'One_and_Half_Fin'
 'One_and_Half_Unf' 'Two_and_Half_Unf' 'Two_and_Half_Fin']
Encoded values in 'House_Style': [5 0 4 3 1 2 7 6]
----------------------------------------
Original values in 'Roof_Style': ['Hip' 'Gable' 'Gambrel' 'Mansard' 'Flat'
'Shed']
Encoded values in 'Roof_Style': [3 1 2 4 0 5]
----------------------------------------
Original values in 'Roof_Matl': ['CompShg' 'Tar&Grv' 'Roll' 'WdShngl' 'WdSha
ke' 'ClyTile' 'Membran'
 'Metal']
Encoded values in 'Roof_Matl': [1 5 4 7 6 0 2 3]
----------------------------------------
Original values in 'Exterior_1st': ['Stucco' 'HdBoard' 'VinylSd' 'Plywood'
'MetalSd' 'CemntBd' 'Wd Sdng'
 'BrkFace' 'AsbShng' 'WdShing' 'BrkComm' 'Stone' 'PreCast' 'AsphShn'
 'ImStucc' 'CBlock']
Encoded values in 'Exterior_1st': [12  6 13  9  8  5 14  3  0 15  2 11 10  1
7  4]
----------------------------------------
Original values in 'Exterior_2nd': ['Wd Shng' 'HdBoard' 'Stucco' 'VinylSd'
'Brk Cmn' 'MetalSd' 'Plywood'
 'CmentBd' 'Wd Sdng' 'BrkFace' 'AsbShng' 'Other' 'ImStucc' 'Stone'
 'AsphShn' 'PreCast' 'CBlock']
Encoded values in 'Exterior_2nd': [16  6 13 14  2  8 10  5 15  3  0  9  7 12
1 11  4]
----------------------------------------
Original values in 'Mas_Vnr_Type': [nan 'BrkFace' 'Stone' 'BrkCmn' 'CBlock']
Encoded values in 'Mas_Vnr_Type': [4 1 3 0 2]
----------------------------------------
Original values in 'Exter_Qual': ['Typical' 'Good' 'Excellent' 'Fair']
Encoded values in 'Exter_Qual': [3 2 0 1]
----------------------------------------
Original values in 'Exter_Cond': ['Good' 'Typical' 'Fair' 'Excellent' 'Poo
r']
Encoded values in 'Exter_Cond': [2 4 1 0 3]
----------------------------------------
Original values in 'Foundation': ['PConc' 'CBlock' 'BrkTil' 'Slab' 'Stone'
'Wood']
Encoded values in 'Foundation': [2 1 0 3 4 5]
----------------------------------------
Original values in 'Bsmt_Qual': ['Typical' 'Good' 'Fair' 'Excellent' 'No_Bas
ement' 'Poor']
Encoded values in 'Bsmt_Qual': [5 2 1 0 3 4]
----------------------------------------
```

```
Original values in 'Bsmt_Cond': ['Typical' 'Good' 'Fair' 'No_Basement' 'Poo
r' 'Excellent']
Encoded values in 'Bsmt_Cond': [5 2 1 3 4 0]
----------------------------------------
Original values in 'Bsmt_Exposure': ['No' 'Av' 'Gd' 'Mn' 'No_Basement']
Encoded values in 'Bsmt_Exposure': [3 0 1 2 4]
----------------------------------------
Original values in 'BsmtFin_Type_1': ['Unf' 'ALQ' 'GLQ' 'Rec' 'BLQ' 'LwQ' 'N
o_Basement']
Encoded values in 'BsmtFin_Type_1': [6 0 2 5 1 3 4]
----------------------------------------
Original values in 'BsmtFin_Type_2': ['Unf' 'GLQ' 'Rec' 'LwQ' 'No_Basement'
'BLQ' 'ALQ']
Encoded values in 'BsmtFin_Type_2': [6 2 5 3 4 1 0]
----------------------------------------
Original values in 'Heating': ['GasA' 'Grav' 'GasW' 'Wall' 'OthW' 'Floor']
Encoded values in 'Heating': [1 3 2 5 4 0]
----------------------------------------
Original values in 'Heating_QC': ['Fair' 'Excellent' 'Typical' 'Good' 'Poo
r']
Encoded values in 'Heating_QC': [1 0 4 2 3]
----------------------------------------
Original values in 'Central_Air': ['Y' 'N']
Encoded values in 'Central_Air': [1 0]
----------------------------------------
Original values in 'Electrical': ['SBrkr' 'FuseA' 'FuseF' 'Unknown' 'FuseP'
'Mix']
Encoded values in 'Electrical': [4 0 1 5 2 3]
----------------------------------------
Original values in 'Kitchen_Qual': ['Good' 'Typical' 'Fair' 'Excellent' 'Poo
r']
Encoded values in 'Kitchen_Qual': [2 4 1 0 3]
----------------------------------------
Original values in 'Functional': ['Typ' 'Min2' 'Maj1' 'Min1' 'Mod' 'Maj2' 'S
al' 'Sev']
Encoded values in 'Functional': [7 3 0 2 4 1 5 6]
----------------------------------------
Original values in 'Fireplace_Qu': ['Good' 'No_Fireplace' 'Typical' 'Fair'
'Poor' 'Excellent']
Encoded values in 'Fireplace_Qu': [2 3 5 1 4 0]
----------------------------------------
Original values in 'Garage_Type': ['Detchd' 'BuiltIn' 'No_Garage' 'Attchd'
'CarPort' 'Basment'
 'More_Than_Two_Types']
Encoded values in 'Garage_Type': [4 2 6 0 3 1 5]
----------------------------------------
Original values in 'Garage_Finish': ['Unf' 'Fin' 'No_Garage' 'RFn']
Encoded values in 'Garage_Finish': [3 0 1 2]
----------------------------------------
Original values in 'Garage_Qual': ['Typical' 'No_Garage' 'Fair' 'Good' 'Poo
r' 'Excellent']
Encoded values in 'Garage_Qual': [5 3 1 2 4 0]
----------------------------------------
Original values in 'Garage_Cond': ['Typical' 'No_Garage' 'Fair' 'Poor' 'Goo
d' 'Excellent']
Encoded values in 'Garage_Cond': [5 3 1 4 2 0]
```

```
----------------------------------------
Original values in 'Paved_Drive': ['Paved' 'Partial_Pavement' 'Dirt_Gravel']
Encoded values in 'Paved_Drive': [2 1 0]
----------------------------------------
Original values in 'Pool_QC': ['No_Pool' 'Fair' 'Excellent' 'Good' 'Typica
l']
Encoded values in 'Pool_QC': [3 1 0 2 4]
----------------------------------------
Original values in 'Fence': ['Minimum_Privacy' 'No_Fence' 'Good_Privacy' 'Go
od_Wood'
 'Minimum_Wood_Wire']
Encoded values in 'Fence': [2 4 0 1 3]
----------------------------------------
Original values in 'Misc_Feature': [nan 'Shed' 'TenC' 'Gar2' 'Elev' 'Othr']
Encoded values in 'Misc_Feature': [5 3 4 1 0 2]
----------------------------------------
Original values in 'Sale_Type': ['WD ' 'New' 'COD' 'ConLw' 'ConLD' 'Oth' 'Co
nLI' 'VWD' 'CWD' 'Con']
Encoded values in 'Sale_Type': [9 6 0 5 3 7 4 8 1 2]
----------------------------------------
Original values in 'Sale_Condition': ['Normal' 'Partial' 'Family' 'Abnorml'
'AdjLand' 'Alloca']
Encoded values in 'Sale_Condition': [4 5 3 0 1 2]
----------------------------------------
```

In [11]:
```
X = df.drop('Sale_Price', axis=1)
y = df['Sale_Price']
```

Linear Regression

In [12]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,rando
```

In [13]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_scor
# Step 3: Initialize and train the Linear Regression model
lr = LinearRegression()
lr.fit(X_train, y_train)

# Step 4: Make predictions on the test set
y_pred = lr.predict(X_test)

# Step 5: Calculate performance metrics on the test set
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error on Test Set: {mae:.2f}')
print(f'Mean Squared Error on Test Set: {mse:.2f}')
print(f'R-squared on Test Set: {r2:.2f}')
```

```
Mean Absolute Error on Test Set: 19787.26
Mean Squared Error on Test Set: 1370465347.08
R-squared on Test Set: 0.80
```

SVM

In [14]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_scor

# Step 3: Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 4: Initialize the SVR model with the RBF kernel
svr = SVR(kernel='rbf')

# Step 5: Define the hyperparameter grid for tuning
param_grid = {
    'C': [0.1, 1, 10, 100],
    'epsilon': [0.01, 0.1, 0.2, 0.5],
    'gamma': ['scale', 'auto']
}

# Step 6: Use GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(svr, param_grid, cv=5, scoring='r2', n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)

# Step 7: Get the best model from GridSearchCV
best_svr = grid_search.best_estimator_

# Step 8: Make predictions on the test set
y_pred = best_svr.predict(X_test_scaled)

# Step 9: Calculate performance metrics on the test set
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Best Hyperparameters: {grid_search.best_params_}')
print(f'Mean Absolute Error on Test Set: {mae:.2f}')
print(f'Mean Squared Error on Test Set: {mse:.2f}')
print(f'R-squared on Test Set: {r2:.2f}')
```

```
Best Hyperparameters: {'C': 100, 'epsilon': 0.01, 'gamma': 'auto'}
Mean Absolute Error on Test Set: 52113.54
Mean Squared Error on Test Set: 6361710220.17
R-squared on Test Set: 0.08
```

Decision Tree

In [15]:
```
from sklearn.tree import DecisionTreeRegressor

# Use 'squared_error' for the criterion
```

```
regressor = DecisionTreeRegressor(criterion='squared_error', max_depth=3, ra

# Fit the model
regressor.fit(X_train, y_train)

# Make predictions
y_pred = regressor.predict(X_test)
```

In [16]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_scor

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae:.2f}")
print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Absolute Error: 31175.73
Mean Squared Error: 1907512301.00
Root Mean Squared Error: 43675.08
R-squared: 0.72
```

Random Forest

In [17]:
```
from sklearn.ensemble import RandomForestRegressor

# Initialize a RandomForestRegressor with n_estimators = 300
regressor = RandomForestRegressor(n_estimators=300, random_state=0)

# Fit the model to the training set
regressor.fit(X_train, y_train)

# Make predictions
y_pred = regressor.predict(X_test)
```

In [18]:
```
# Most important scores of the model's features
feature_scores = pd.Series(regressor.feature_importances_, index=X_train.col

pd.set_option('display.max_rows', None)

# Display feature scores
print(feature_scores)
```

Overall_Qual	0.599095
Gr_Liv_Area	0.096839
First_Flr_SF	0.039677
Total_Bsmt_SF	0.038026
Second_Flr_SF	0.019448
Garage_Area	0.018280
Lot_Area	0.015105
Kitchen_Qual	0.014560
Bsmt_Qual	0.014444
Garage_Cars	0.014218
Year_Built	0.010135
Longitude	0.008205
Year_Remod_Add	0.008196
Latitude	0.008039
Full_Bath	0.007555
Bsmt_Unf_SF	0.006569
Mas_Vnr_Area	0.006251
Fireplaces	0.005312
Open_Porch_SF	0.004067
Lot_Frontage	0.004036
Overall_Cond	0.003921
Wood_Deck_SF	0.003545
Garage_Type	0.003298
Mo_Sold	0.003183
TotRms_AbvGrd	0.003101
Neighborhood	0.002599
Screen_Porch	0.002559
Bsmt_Full_Bath	0.002471
MS_Zoning	0.002187
Exter_Qual	0.002024
Bsmt_Exposure	0.002019
Sale_Condition	0.001924
BsmtFin_Type_1	0.001596
BsmtFin_SF_1	0.001565
MS_SubClass	0.001548
Year_Sold	0.001488
Bedroom_AbvGr	0.001359
Exterior_1st	0.001212
Exterior_2nd	0.001192
Central_Air	0.001136
Fireplace_Qu	0.001103
Land_Contour	0.000995
Garage_Finish	0.000969
Lot_Shape	0.000923
Roof_Style	0.000919
Mas_Vnr_Type	0.000896
Heating_QC	0.000807
Pool_Area	0.000766
Enclosed_Porch	0.000730
Half_Bath	0.000705
House_Style	0.000603
Foundation	0.000599
Lot_Config	0.000556
Bsmt_Half_Bath	0.000483
Paved_Drive	0.000461
Exter_Cond	0.000461

```
Sale_Type              0.000460
Fence                  0.000460
Land_Slope             0.000446
Garage_Qual            0.000445
Bsmt_Cond              0.000432
Condition_1            0.000398
Pool_QC                0.000380
BsmtFin_Type_2         0.000374
Bldg_Type              0.000366
Functional             0.000351
BsmtFin_SF_2           0.000298
Alley                  0.000294
Kitchen_AbvGr          0.000211
Garage_Cond            0.000197
Electrical             0.000182
Misc_Feature           0.000156
Condition_2            0.000156
Misc_Val               0.000119
Three_season_porch     0.000108
Roof_Matl              0.000104
Heating                0.000068
Low_Qual_Fin_SF        0.000019
Street                 0.000018
Utilities              0.000000
dtype: float64
```

In [19]:
```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae:.2f}")
print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Absolute Error: 15751.14
Mean Squared Error: 658577807.90
Root Mean Squared Error: 25662.77
R-squared: 0.90
```

XG Boost

In [20]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.11/site
-packages (2.0.3)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.11/site-p
ackages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.11/site-p
ackages (from xgboost) (1.11.4)
Note: you may need to restart the kernel to use updated packages.
```

In [21]:
```
from xgboost import XGBRegressor
xgb = XGBRegressor(n_estimators=1000, learning_rate=0.001, random_state=0)
```

```python
xgb.fit(X_train, y_train)
y_pred = xgb.predict(X_test)
```

In [22]:
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_scor

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae:.2f}")
print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Absolute Error: 29013.66
Mean Squared Error: 1840591138.13
Root Mean Squared Error: 42902.11
R-squared: 0.73
```

In [23]:
```python
feature_scores = pd.Series(xgb.feature_importances_, index=X_train.columns).
print(feature_scores)
```

```
Overall_Qual          0.486642
Garage_Cars           0.053597
First_Flr_SF          0.036847
Gr_Liv_Area           0.034610
Kitchen_Qual          0.031577
Land_Contour          0.030302
Central_Air           0.018713
Total_Bsmt_SF         0.018086
Bsmt_Qual             0.016190
Second_Flr_SF         0.014994
Garage_Type           0.014971
Latitude              0.013602
Fireplaces            0.013317
Exter_Qual            0.012543
Full_Bath             0.011217
Neighborhood          0.010765
Year_Remod_Add        0.009917
Lot_Area              0.008650
MS_Zoning             0.008279
BsmtFin_Type_1        0.007970
Year_Built            0.007941
BsmtFin_SF_2          0.007489
Longitude             0.007317
Mas_Vnr_Type          0.007143
Garage_Area           0.007018
Lot_Shape             0.006627
Paved_Drive           0.006447
Overall_Cond          0.005543
Foundation            0.005333
Bsmt_Full_Bath        0.005329
Bsmt_Unf_SF           0.004511
Half_Bath             0.004488
House_Style           0.004425
BsmtFin_Type_2        0.004234
Open_Porch_SF         0.003912
Sale_Condition        0.003845
Sale_Type             0.003676
Lot_Frontage          0.003520
Garage_Finish         0.003438
Mas_Vnr_Area          0.003123
Fence                 0.003080
Fireplace_Qu          0.002901
Exter_Cond            0.002829
Roof_Style            0.002794
Screen_Porch          0.002674
Wood_Deck_SF          0.002607
Bldg_Type             0.002571
Garage_Qual           0.002276
Bsmt_Exposure         0.002081
Bedroom_AbvGr         0.002068
Exterior_2nd          0.002005
Mo_Sold               0.001902
Year_Sold             0.001700
MS_SubClass           0.001508
Functional            0.001467
Exterior_1st          0.001393
```

```
Kitchen_AbvGr        0.001131
Condition_1          0.001054
TotRms_AbvGrd        0.001012
Bsmt_Half_Bath       0.000843
Electrical           0.000774
Heating_QC           0.000625
Enclosed_Porch       0.000345
Lot_Config           0.000205
Condition_2          0.000008
Misc_Feature         0.000000
Roof_Matl            0.000000
Street               0.000000
Land_Slope           0.000000
Bsmt_Cond            0.000000
BsmtFin_SF_1         0.000000
Misc_Val             0.000000
Pool_Area            0.000000
Pool_QC              0.000000
Heating              0.000000
Utilities            0.000000
Garage_Cond          0.000000
Low_Qual_Fin_SF      0.000000
Alley                0.000000
Three_season_porch   0.000000
dtype: float32
```

Best algorithm

In [24]:

```
from sklearn.metrics import accuracy_score,log_loss
```

In [25]:

```
lr = LinearRegression()
dt = DecisionTreeRegressor(max_depth=3, random_state=0)
rfr = RandomForestRegressor(n_estimators=100, random_state=0)
svr = SVR(kernel='poly', C=1, degree=3)
xgbr = XGBRegressor(n_estimators=1000, learning_rate=0.01, random_state=0)
```

In [26]:

```
algos = [lr, rfr, svr, dt, xgbr]
ml_algo = ['Linear Regression', 'Random Forest', 'SVR', 'Decision Tree', 'XG
```

In [27]:

```
# Step 3: Loop through all the algorithms and evaluate their performance
for i, j in zip(algos, ml_algo):
    i.fit(X_train, y_train)  # Fit the model on training data
    pred = i.predict(X_test)  # Predict on test data

    # Calculate regression metrics
    mae = mean_absolute_error(y_test, pred)
    mse = mean_squared_error(y_test, pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, pred)

    # Print the results
    print(j, ':\n')
    print("Test metrics")
    print(f'Mean Absolute Error: {mae:.2f}')
```

```
    print(f'Mean Squared Error: {mse:.2f}')
    print(f"Root Mean Squared Error: {rmse:.2f}")
    print(f'R-squared: {r2:.2f}')
    print('=' * 40)

    pred = i.predict(X_train)  # Predict on test data

    # Calculate regression metrics
    mae = mean_absolute_error(y_train, pred)
    mse = mean_squared_error(y_train, pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_train, pred)

    # Print the results
    print(j, ':\n')
    print("Train metrics")
    print(f'Mean Absolute Error: {mae:.2f}')
    print(f'Mean Squared Error: {mse:.2f}')
    print(f"Root Mean Squared Error: {rmse:.2f}")
    print(f'R-squared: {r2:.2f}')
    print('=' * 40)
```

```
Linear Regression :

Test metrics
Mean Absolute Error: 19787.26
Mean Squared Error: 1370465347.08
Root Mean Squared Error: 37019.80
R-squared: 0.80
========================================
Linear Regression :

Train metrics
Mean Absolute Error: 17756.28
Mean Squared Error: 740154429.88
Root Mean Squared Error: 27205.78
R-squared: 0.88
========================================
Random Forest :

Test metrics
Mean Absolute Error: 15729.70
Mean Squared Error: 647382431.26
Root Mean Squared Error: 25443.71
R-squared: 0.91
========================================
Random Forest :

Train metrics
Mean Absolute Error: 6149.49
Mean Squared Error: 107941849.95
Root Mean Squared Error: 10389.51
R-squared: 0.98
========================================
SVR :

Test metrics
Mean Absolute Error: 57264.13
Mean Squared Error: 7096139404.22
Root Mean Squared Error: 84238.59
R-squared: -0.03
========================================
SVR :

Train metrics
Mean Absolute Error: 55764.28
Mean Squared Error: 6605002656.48
Root Mean Squared Error: 81271.17
R-squared: -0.06
========================================
Decision Tree :

Test metrics
Mean Absolute Error: 31175.73
Mean Squared Error: 1907512301.00
Root Mean Squared Error: 43675.08
R-squared: 0.72
========================================
```

```
Decision Tree :

Train metrics
Mean Absolute Error: 29782.53
Mean Squared Error: 1640383948.15
Root Mean Squared Error: 40501.65
R-squared: 0.74
========================================
XGBoost :

Test metrics
Mean Absolute Error: 14623.69
Mean Squared Error: 536638209.82
Root Mean Squared Error: 23165.45
R-squared: 0.92
========================================
XGBoost :

Train metrics
Mean Absolute Error: 4559.32
Mean Squared Error: 36471085.86
Root Mean Squared Error: 6039.13
R-squared: 0.99
========================================
```

In [29]:
```
import pandas as pd

# Assume df is your DataFrame and 'variable_name' is the column of interest
min_value = df['Sale_Price'].min()
max_value = df['Sale_Price'].max()

print(f"Minimum value of Price Sales: {min_value}")
print(f"Maximum value of Price Sales: {max_value}")
```

```
Minimum value of Price Sales: 12789
Maximum value of Price Sales: 755000
```

In [34]:
```
for model, name in zip([best_rf, best_xgb], ['Random Forest', 'XGBoost']):
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f'{name} - Best Hyperparameters: {model.get_params()}')
    print(f'Mean Absolute Error: {mae:.2f}')
    print(f'Mean Squared Error: {mse:.2f}')
    print(f'Root Mean Squared Error: {rmse:.2f}')
    print(f'R-squared: {r2:.2f}')
    print('=' * 40)
```

```
Random Forest - Best Hyperparameters: {'bootstrap': True, 'ccp_alpha': 0.0,
'criterion': 'squared_error', 'max_depth': 15, 'max_features': 1.0, 'max_lea
f_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samp
les_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_es
timators': 200, 'n_jobs': None, 'oob_score': False, 'random_state': 0, 'verb
ose': 0, 'warm_start': False}
Mean Absolute Error: 12092.23
Mean Squared Error: 466191149.56
Root Mean Squared Error: 21591.46
R-squared: 0.92
========================================
XGBoost - Best Hyperparameters: {'objective': 'reg:squarederror', 'base_scor
e': None, 'booster': None, 'callbacks': None, 'colsample_bylevel': None, 'co
lsample_bynode': None, 'colsample_bytree': 0.8, 'device': None, 'early_stopp
ing_rounds': None, 'enable_categorical': False, 'eval_metric': None, 'featur
e_types': None, 'gamma': None, 'grow_policy': None, 'importance_type': None,
'interaction_constraints': None, 'learning_rate': 0.1, 'max_bin': None, 'max
_cat_threshold': None, 'max_cat_to_onehot': None, 'max_delta_step': None, 'm
ax_depth': 3, 'max_leaves': None, 'min_child_weight': None, 'missing': nan,
'monotone_constraints': None, 'multi_strategy': None, 'n_estimators': 500,
'n_jobs': None, 'num_parallel_tree': None, 'random_state': 0, 'reg_alpha': N
one, 'reg_lambda': None, 'sampling_method': None, 'scale_pos_weight': None,
'subsample': 0.9, 'tree_method': None, 'validate_parameters': None, 'verbosi
ty': None}
Mean Absolute Error: 12651.57
Mean Squared Error: 402790465.93
Root Mean Squared Error: 20069.64
R-squared: 0.93
========================================
```

In [ ]: