Project Summary:

Predict Housing Prices Project 1 Using Simple Linear Regression

Objective:

The task is to develop a simple linear regression model that predicts housing prices using the Boston Housing dataset. The entire process—data loading, preprocessing, model training, evaluation, and making predictions within a single script file.

Process:

1. Imported necessary libraries and connected to Kaggle API via Anaconda Prompt to download the dataset.
2. Extracted the data, defined file paths, and loaded the datasets into Jupyter Notebook.
3. Explored the data to understand its structure and contents.
4. Performed preprocessing:

```
      Imputed missing numerical values with the mean.
      Imputed missing categorical values with the most frequent values using Simple Imputer.
```

5. Applied label encoding to categorical columns.
6. Detected outliers using the IQR method and visualized them.
7. Handled outliers by "Capping" (setting them to upper and lower bounds).
8. Conducted correlation analysis to assess relationships with the target variable.
9. Dropped constant columns and those with low correlation (threshold: 0.5).
10. Scaled the data using StandardScaler.
11. Split the data into training and test sets using train_test_split.
12. Trained a linear regression model on the training set and made predictions on the test set.
13. Evaluated model performance using MAE, RMSE, R^2, and MSE.
14. Printed some sample predictions made by the model for comparison.

Importing Libraries and frameworks

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing  import StandardScaler
import seaborn as sns
import matplotlib.pyplot as pls
```

Downloading the Data from Kaggle

Downloaded the data using the Anaconda Prompt command

```
# (base) C:\Users\ADDIS>del house-prices-advanced-regression-techniques.zip

#(base) C:\Users\ADDIS>
#(base) C:\Users\ADDIS>kaggle competitions download -c house-prices-advanced-regression-techniques
#Downloading house-prices-advanced-regression-techniques.zip to C:\Users\ADDIS
#100%|██████████████████████████████████████████████████████████████████████████████████████| 199k/199k [00:00<00:00, 248kB/s]
#100%|██████████████████████████████████████████████████████████████████████████████████████| 199k/199k [00:00<00:00, 246kB/s]
```

Unzipping the Data file

```
import zipfile
import os

# Define the path to the zip file and the directory where you want to extract it
zip_file_path = r'C:\Users\ADDIS\house-prices-advanced-regression-techniques.zip'
extract_dir = r'C:\Users\ADDIS\house-prices-data'

# Create the extraction directory
```

```
os.makedirs(extract_dir, exist_ok=True)

# Unzip the file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print(f"Files extracted to {extract_dir}")
```

```
Files extracted to C:\Users\ADDIS\house-prices-data
```

Defining the path for the data

```
train_file_path = 'C:\\Users\\ADDIS\\house-prices-advanced-regression-techniques\\train.csv'
test_file_path = 'C:\\Users\\ADDIS\\house-prices-advanced-regression-techniques\\test.csv'
```

Loding the datasets to a dataframe

```
# Loading the datasets
train_data = pd.read_csv(train_file_path)
test_data = pd.read_csv(test_file_path)

# Display Data Types of Each Column
for column in train_data.columns:
    print(f"{column}: {train_data[column].dtype}")
```

```
Id: int64
MSSubClass: int64
MSZoning: object
LotFrontage: float64
LotArea: int64
Street: object
Alley: object
LotShape: object
LandContour: object
Utilities: object
LotConfig: object
LandSlope: object
Neighborhood: object
Condition1: object
Condition2: object
BldgType: object
HouseStyle: object
OverallQual: int64
OverallCond: int64
YearBuilt: int64
YearRemodAdd: int64
RoofStyle: object
```

```
RoofMatl: object
Exterior1st: object
Exterior2nd: object
MasVnrType: object
MasVnrArea: float64
ExterQual: object
ExterCond: object
Foundation: object
BsmtQual: object
BsmtCond: object
BsmtExposure: object
BsmtFinType1: object
BsmtFinSF1: int64
BsmtFinType2: object
BsmtFinSF2: int64
BsmtUnfSF: int64
TotalBsmtSF: int64
Heating: object
HeatingQC: object
CentralAir: object
Electrical: object
1stFlrSF: int64
2ndFlrSF: int64
LowQualFinSF: int64
GrLivArea: int64
BsmtFullBath: int64
BsmtHalfBath: int64
FullBath: int64
HalfBath: int64
BedroomAbvGr: int64
KitchenAbvGr: int64
KitchenQual: object
TotRmsAbvGrd: int64
Functional: object
Fireplaces: int64
FireplaceQu: object
GarageType: object
GarageYrBlt: float64
GarageFinish: object
GarageCars: int64
GarageArea: int64
GarageQual: object
GarageCond: object
PavedDrive: object
WoodDeckSF: int64
OpenPorchSF: int64
EnclosedPorch: int64
3SsnPorch: int64
ScreenPorch: int64
```

```
PoolArea: int64
PoolQC: object
Fence: object
MiscFeature: object
MiscVal: int64
MoSold: int64
YrSold: int64
SaleType: object
SaleCondition: object
SalePrice: int64
```

Summary Statistics for Numerical Columns in dataset

Check for Missing Values for train data

Missing Values in Each Column for test_data

```
print(train_data.shape)
print(test_data.shape)
```

```
(1460, 81)
(1459, 80)
```

```
train_data.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 LotConfig LandSlope Neighborhood Condition1  \
0         Lvl    AllPub    Inside       Gtl      CollgCr       Norm
1         Lvl    AllPub       FR2       Gtl      Veenker      Feedr
2         Lvl    AllPub    Inside       Gtl      CollgCr       Norm
3         Lvl    AllPub    Corner       Gtl      Crawfor       Norm
4         Lvl    AllPub       FR2       Gtl      NoRidge       Norm

  Condition2 BldgType HouseStyle  OverallQual  OverallCond  YearBuilt
\
```

```
0       Norm     1Fam     2Story            7            5       2003

1       Norm     1Fam     1Story            6            8       1976

2       Norm     1Fam     2Story            7            5       2001

3       Norm     1Fam     2Story            7            5       1915

4       Norm     1Fam     2Story            8            5       2000


   YearRemodAdd RoofStyle RoofMatl Exterior1st Exterior2nd MasVnrType
\
0          2003     Gable  CompShg     VinylSd     VinylSd    BrkFace

1          1976     Gable  CompShg     MetalSd     MetalSd        NaN

2          2002     Gable  CompShg     VinylSd     VinylSd    BrkFace

3          1970     Gable  CompShg     Wd Sdng     Wd Shng        NaN

4          2000     Gable  CompShg     VinylSd     VinylSd    BrkFace


   MasVnrArea ExterQual ExterCond Foundation BsmtQual BsmtCond
BsmtExposure  \
0       196.0        Gd        TA      PConc       Gd       TA
No
1         0.0        TA        TA     CBlock       Gd       TA
Gd
2       162.0        Gd        TA      PConc       Gd       TA
Mn
3         0.0        TA        TA     BrkTil       TA       Gd
No
4       350.0        Gd        TA      PConc       Gd       TA
Av

  BsmtFinType1  BsmtFinSF1 BsmtFinType2  BsmtFinSF2  BsmtUnfSF
TotalBsmtSF  \
0          GLQ         706          Unf           0        150
856
1          ALQ         978          Unf           0        284
1262
2          GLQ         486          Unf           0        434
920
3          ALQ         216          Unf           0        540
756
4          GLQ         655          Unf           0        490
1145

  Heating HeatingQC CentralAir Electrical  1stFlrSF  2ndFlrSF
```

```
LowQualFinSF  \
0    GasA        Ex           Y      SBrkr       856       854
0
1    GasA        Ex           Y      SBrkr      1262         0
0
2    GasA        Ex           Y      SBrkr       920       866
0
3    GasA        Gd           Y      SBrkr       961       756
0
4    GasA        Ex           Y      SBrkr      1145      1053
0

   GrLivArea  BsmtFullBath  BsmtHalfBath  FullBath  HalfBath
BedroomAbvGr  \
0       1710             1             0         2         1
3
1       1262             0             1         2         0
3
2       1786             1             0         2         1
3
3       1717             1             0         1         0
3
4       2198             1             0         2         1
4

   KitchenAbvGr KitchenQual  TotRmsAbvGrd Functional  Fireplaces
FireplaceQu  \
0             1          Gd             8        Typ           0
NaN
1             1          TA             6        Typ           1
TA
2             1          Gd             6        Typ           1
TA
3             1          Gd             7        Typ           1
Gd
4             1          Gd             9        Typ           1
TA

  GarageType  GarageYrBlt GarageFinish  GarageCars  GarageArea
GarageQual  \
0     Attchd       2003.0          RFn           2         548
TA
1     Attchd       1976.0          RFn           2         460
TA
2     Attchd       2001.0          RFn           2         608
TA
3     Detchd       1998.0          Unf           3         642
TA
4     Attchd       2000.0          RFn           3         836
TA
```

```
  GarageCond PavedDrive  WoodDeckSF  OpenPorchSF  EnclosedPorch 
3SsnPorch  \
0         TA          Y           0           61              0 
0   
1         TA          Y         298            0              0 
0   
2         TA          Y           0           42              0 
0   
3         TA          Y           0           35            272 
0   
4         TA          Y         192           84              0 
0   

   ScreenPorch  PoolArea PoolQC Fence MiscFeature  MiscVal  MoSold 
YrSold  \
0            0         0    NaN   NaN         NaN        0       2 
2008   
1            0         0    NaN   NaN         NaN        0       5 
2007   
2            0         0    NaN   NaN         NaN        0       9 
2008   
3            0         0    NaN   NaN         NaN        0       2 
2006   
4            0         0    NaN   NaN         NaN        0      12 
2008   

  SaleType SaleCondition  SalePrice  
0       WD        Normal     208500  
1       WD        Normal     181500  
2       WD        Normal     223500  
3       WD       Abnorml     140000  
4       WD        Normal     250000  
```

Checks to see the columns and the type of data in each column (TEST DATA)

It showed that the data contains categorical and numerical columns which isnt good for prediction

```
test_data.head()
```

```
     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

  LandContour Utilities LotConfig LandSlope Neighborhood Condition1  \
0         Lvl    AllPub    Inside       Gtl        NAmes      Feedr
1         Lvl    AllPub    Corner       Gtl        NAmes       Norm
2         Lvl    AllPub    Inside       Gtl      Gilbert       Norm
3         Lvl    AllPub    Inside       Gtl      Gilbert       Norm
4         HLS    AllPub    Inside       Gtl      StoneBr       Norm

  Condition2 BldgType HouseStyle  OverallQual  OverallCond  YearBuilt
\
0       Norm     1Fam     1Story            5            6       1961

1       Norm     1Fam     1Story            6            6       1958

2       Norm     1Fam     2Story            5            5       1997

3       Norm     1Fam     2Story            6            6       1998

4       Norm   TwnhsE     1Story            8            5       1992


   YearRemodAdd RoofStyle RoofMatl Exterior1st Exterior2nd MasVnrType
\
0          1961     Gable  CompShg     VinylSd     VinylSd        NaN

1          1958       Hip  CompShg     Wd Sdng     Wd Sdng    BrkFace

2          1998     Gable  CompShg     VinylSd     VinylSd        NaN

3          1998     Gable  CompShg     VinylSd     VinylSd    BrkFace

4          1992     Gable  CompShg     HdBoard     HdBoard        NaN


   MasVnrArea ExterQual ExterCond Foundation BsmtQual BsmtCond
BsmtExposure  \
0         0.0        TA        TA     CBlock       TA       TA
No
1       108.0        TA        TA     CBlock       TA       TA
No
2         0.0        TA        TA      PConc       Gd       TA
No
3        20.0        TA        TA      PConc       TA       TA
No
4         0.0        Gd        TA      PConc       Gd       TA
No

  BsmtFinType1  BsmtFinSF1 BsmtFinType2  BsmtFinSF2  BsmtUnfSF
```

```
TotalBsmtSF  \
0          Rec       468.0          LwQ       144.0      270.0
882.0
1          ALQ       923.0          Unf         0.0      406.0
1329.0
2          GLQ       791.0          Unf         0.0      137.0
928.0
3          GLQ       602.0          Unf         0.0      324.0
926.0
4          ALQ       263.0          Unf         0.0     1017.0
1280.0

  Heating HeatingQC CentralAir Electrical  1stFlrSF  2ndFlrSF
LowQualFinSF  \
0    GasA        TA          Y      SBrkr       896         0
0
1    GasA        TA          Y      SBrkr      1329         0
0
2    GasA        Gd          Y      SBrkr       928       701
0
3    GasA        Ex          Y      SBrkr       926       678
0
4    GasA        Ex          Y      SBrkr      1280         0
0

   GrLivArea  BsmtFullBath  BsmtHalfBath  FullBath  HalfBath
BedroomAbvGr  \
0        896           0.0           0.0         1         0
2
1       1329           0.0           0.0         1         1
3
2       1629           0.0           0.0         2         1
3
3       1604           0.0           0.0         2         1
3
4       1280           0.0           0.0         2         0
2

   KitchenAbvGr KitchenQual  TotRmsAbvGrd Functional  Fireplaces
FireplaceQu  \
0             1          TA             5        Typ           0
NaN
1             1          Gd             6        Typ           0
NaN
2             1          TA             6        Typ           1
TA
3             1          Gd             7        Typ           1
Gd
4             1          Gd             5        Typ           0
NaN
```

```
  GarageType  GarageYrBlt GarageFinish  GarageCars  GarageArea
GarageQual  \
0     Attchd       1961.0          Unf         1.0       730.0
TA
1     Attchd       1958.0          Unf         1.0       312.0
TA
2     Attchd       1997.0          Fin         2.0       482.0
TA
3     Attchd       1998.0          Fin         2.0       470.0
TA
4     Attchd       1992.0          RFn         2.0       506.0
TA

  GarageCond PavedDrive  WoodDeckSF  OpenPorchSF  EnclosedPorch
3SsnPorch  \
0         TA          Y         140            0              0
0
1         TA          Y         393           36              0
0
2         TA          Y         212           34              0
0
3         TA          Y         360           36              0
0
4         TA          Y           0           82              0
0

   ScreenPorch  PoolArea PoolQC  Fence MiscFeature  MiscVal  MoSold
YrSold  \
0          120         0    NaN  MnPrv         NaN        0       6
2010
1            0         0    NaN    NaN        Gar2    12500       6
2010
2            0         0    NaN  MnPrv         NaN        0       3
2010
3            0         0    NaN    NaN         NaN        0       6
2010
4          144         0    NaN    NaN         NaN        0       1
2010

  SaleType SaleCondition
0       WD        Normal
1       WD        Normal
2       WD        Normal
3       WD        Normal
4       WD        Normal
```

Checks for missing values (TRAIN DATA)

Results showed that the test_data contains columns with missing values

```
train_data.isnull().sum()
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

```
# Set display options to show all columns
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# Display the count of missing values in each column
print(train_data.isnull().sum())
```

```
Id                  0
MSSubClass          0
MSZoning            0
LotFrontage       259
LotArea             0
Street              0
Alley            1369
LotShape            0
LandContour         0
Utilities           0
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
OverallQual         0
OverallCond         0
YearBuilt           0
YearRemodAdd        0
RoofStyle           0
RoofMatl            0
Exterior1st         0
Exterior2nd         0
```

```
MasVnrType        872
MasVnrArea          8
ExterQual           0
ExterCond           0
Foundation          0
BsmtQual           37
BsmtCond           37
BsmtExposure       38
BsmtFinType1       37
BsmtFinSF1          0
BsmtFinType2       38
BsmtFinSF2          0
BsmtUnfSF           0
TotalBsmtSF         0
Heating             0
HeatingQC           0
CentralAir          0
Electrical          1
1stFlrSF            0
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        0
BsmtHalfBath        0
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         0
TotRmsAbvGrd        0
Functional          0
Fireplaces          0
FireplaceQu       690
GarageType         81
GarageYrBlt        81
GarageFinish       81
GarageCars          0
GarageArea          0
GarageQual         81
GarageCond         81
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           1453
Fence            1179
```

```
MiscFeature      1406
MiscVal             0
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
dtype: int64
```

Checks for missing values (TEST DATA)

Results showed that the test_data contains columns with missing values

```
test_data.isnull().sum()
```

```
Id                  0
MSSubClass          0
MSZoning            4
LotFrontage       227
LotArea             0
Street              0
Alley            1352
LotShape            0
LandContour         0
Utilities           2
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
OverallQual         0
OverallCond         0
YearBuilt           0
YearRemodAdd        0
RoofStyle           0
RoofMatl            0
Exterior1st         1
Exterior2nd         1
MasVnrType        894
MasVnrArea         15
ExterQual           0
ExterCond           0
Foundation          0
BsmtQual           44
BsmtCond           45
BsmtExposure       44
```

```
BsmtFinType1       42
BsmtFinSF1          1
BsmtFinType2       42
BsmtFinSF2          1
BsmtUnfSF           1
TotalBsmtSF         1
Heating             0
HeatingQC           0
CentralAir          0
Electrical          0
1stFlrSF            0
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        2
BsmtHalfBath        2
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         1
TotRmsAbvGrd        0
Functional          2
Fireplaces          0
FireplaceQu       730
GarageType         76
GarageYrBlt        78
GarageFinish       78
GarageCars          1
GarageArea          1
GarageQual         78
GarageCond         78
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           1456
Fence            1169
MiscFeature      1408
MiscVal             0
MoSold              0
YrSold              0
SaleType            1
SaleCondition       0
dtype: int64
```

```python
# Set display options to show all columns
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# Display the count of missing values in each column
print(test_data.isnull().sum())
```

```
Id                 0
MSSubClass         0
MSZoning           4
LotFrontage      227
LotArea            0
Street             0
Alley           1352
LotShape           0
LandContour        0
Utilities          2
LotConfig          0
LandSlope          0
Neighborhood       0
Condition1         0
Condition2         0
BldgType           0
HouseStyle         0
OverallQual        0
OverallCond        0
YearBuilt          0
YearRemodAdd       0
RoofStyle          0
RoofMatl           0
Exterior1st        1
Exterior2nd        1
MasVnrType       894
MasVnrArea        15
ExterQual          0
ExterCond          0
Foundation         0
BsmtQual          44
BsmtCond          45
BsmtExposure      44
BsmtFinType1      42
BsmtFinSF1         1
BsmtFinType2      42
BsmtFinSF2         1
BsmtUnfSF          1
TotalBsmtSF        1
Heating            0
HeatingQC          0
CentralAir         0
Electrical         0
```

```
1stFlrSF            0
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        2
BsmtHalfBath        2
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         1
TotRmsAbvGrd        0
Functional          2
Fireplaces          0
FireplaceQu       730
GarageType         76
GarageYrBlt        78
GarageFinish       78
GarageCars          1
GarageArea          1
GarageQual         78
GarageCond         78
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           1456
Fence            1169
MiscFeature      1408
MiscVal             0
MoSold              0
YrSold              0
SaleType            1
SaleCondition       0
dtype: int64
```

Data Preprocessing

Data preprocessing is a crucial step in preparing a dataset for machine learning models. It ensures that the data is clean, consistent, and ready for analysis.

```
##### HANDLING MISSING VALUES IN  NUMERICAL COLUMNS WITH THE MEAN OF THE COLUMN
```

```
from sklearn.impute import SimpleImputer

# Create the imputer object with mean strategy
imputer = SimpleImputer(strategy='mean')

# Select numerical columns excluding the target variable in train_data
numerical_cols =
train_data.drop(columns=['SalePrice']).select_dtypes(include=['float64
', 'int64']).columns

# Fit the imputer on the training data excluding the target variable
imputer.fit(train_data[numerical_cols])

# Transform the training data excluding the target variable
train_data[numerical_cols] =
imputer.transform(train_data[numerical_cols])

# Transform the test data
test_data[numerical_cols] =
imputer.transform(test_data[numerical_cols])



#### HANDLING MISSING CATEGORICAL VALES IN THE DATASETS

# Fill missing categorical values with the nearest value using
backward fill
train_data = train_data.bfill()
test_data = test_data.bfill()

# Verify that there are no missing values in the categorical columns
print(train_data.isnull().sum())
print(test_data.isnull().sum())
```

```
Id                  0
MSSubClass          0
MSZoning            0
LotFrontage         0
LotArea             0
Street              0
Alley               5
LotShape            0
LandContour         0
Utilities           0
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
```

```
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       3
MasVnrArea       0
ExterQual        0
ExterCond        0
Foundation       0
BsmtQual         0
BsmtCond         0
BsmtExposure     0
BsmtFinType1     0
BsmtFinSF1       0
BsmtFinType2     0
BsmtFinSF2       0
BsmtUnfSF        0
TotalBsmtSF      0
Heating          0
HeatingQC        0
CentralAir       0
Electrical       0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
KitchenQual      0
TotRmsAbvGrd     0
Functional       0
Fireplaces       0
FireplaceQu      2
GarageType       0
GarageYrBlt      0
GarageFinish     0
GarageCars       0
GarageArea       0
GarageQual       0
GarageCond       0
PavedDrive       0
```

```
WoodDeckSF        0
OpenPorchSF       0
EnclosedPorch     0
3SsnPorch         0
ScreenPorch       0
PoolArea          0
PoolQC           36
Fence             2
MiscFeature       2
MiscVal           0
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
dtype: int64
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage        0
LotArea            0
Street             0
Alley             47
LotShape           0
LandContour        0
Utilities          0
LotConfig          0
LandSlope          0
Neighborhood       0
Condition1         0
Condition2         0
BldgType           0
HouseStyle         0
OverallQual        0
OverallCond        0
YearBuilt          0
YearRemodAdd       0
RoofStyle          0
RoofMatl           0
Exterior1st        0
Exterior2nd        0
MasVnrType         0
MasVnrArea         0
ExterQual          0
ExterCond          0
Foundation         0
BsmtQual           0
BsmtCond           0
BsmtExposure       0
```

```
BsmtFinType1       0
BsmtFinSF1         0
BsmtFinType2       0
BsmtFinSF2         0
BsmtUnfSF          0
TotalBsmtSF        0
Heating            0
HeatingQC          0
CentralAir         0
Electrical         0
1stFlrSF           0
2ndFlrSF           0
LowQualFinSF       0
GrLivArea          0
BsmtFullBath       0
BsmtHalfBath       0
FullBath           0
HalfBath           0
BedroomAbvGr       0
KitchenAbvGr       0
KitchenQual        0
TotRmsAbvGrd       0
Functional         0
Fireplaces         0
FireplaceQu        0
GarageType         0
GarageYrBlt        0
GarageFinish       0
GarageCars         0
GarageArea         0
GarageQual         0
GarageCond         0
PavedDrive         0
WoodDeckSF         0
OpenPorchSF        0
EnclosedPorch      0
3SsnPorch          0
ScreenPorch        0
PoolArea           0
PoolQC           208
Fence              1
MiscFeature        1
MiscVal            0
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
dtype: int64
```

```python
### COMFIRMATION THAT ALL MISSING NUMERICAL VALUES ARE HANDLED


### The code handled the missing numerical values appropiately

import pandas as pd

# Select numerical columns in train_data excluding the target variable
numerical_cols_train =
train_data.drop(columns=['SalePrice']).select_dtypes(include=['float64
', 'int64']).columns

# Check for null values in numerical columns of train_data
null_columns_train = train_data[numerical_cols_train].isnull().sum()
null_columns_train = null_columns_train[null_columns_train > 0]
print("Numerical columns with null values in train_data:")
print(null_columns_train)

# Select numerical columns in test_data
numerical_cols_test = test_data.select_dtypes(include=['float64',
'int64']).columns

# Check for null values in numerical columns of test_data
null_columns_test = test_data[numerical_cols_test].isnull().sum()
null_columns_test = null_columns_test[null_columns_test > 0]
print("Numerical columns with null values in test_data:")
print(null_columns_test)
```

```
Numerical columns with null values in train_data:
Series([], dtype: int64)
Numerical columns with null values in test_data:
Series([], dtype: int64)
```

```python
### COMFIRMATION OF THE BACKFILLING OF CATEGORICAL COLUMNS


#It happend that the backfill of categorical columns did not fill the
null values because the missing values are scattered randomly rather
than in a sequence, backfilling might not fill all gaps correctly.

import pandas as pd

# Select categorical columns in train_data excluding the target
variable
categorical_cols_train =
train_data.drop(columns=['SalePrice']).select_dtypes(include=['object'
]).columns

# Check for null values in categorical columns of train_data
null_columns_train = train_data[categorical_cols_train].isnull().sum()
null_columns_train = null_columns_train[null_columns_train > 0]
```

```python
print("Categorical columns with null values in train_data:")
print(null_columns_train)

# Select categorical columns in test_data
categorical_cols_test =
test_data.select_dtypes(include=['object']).columns

# Check for null values in categorical columns of test_data
null_columns_test = test_data[categorical_cols_test].isnull().sum()
null_columns_test = null_columns_test[null_columns_test > 0]
print("Categorical columns with null values in test_data:")
print(null_columns_test)
```

```
Categorical columns with null values in train_data:
Alley            5
MasVnrType       3
FireplaceQu      2
PoolQC          36
Fence            2
MiscFeature      2
dtype: int64
Categorical columns with null values in test_data:
Alley            47
PoolQC          208
Fence             1
MiscFeature       1
dtype: int64
```

```python
## CODE to fill the categorical columns

from sklearn.impute import SimpleImputer

# Create the imputer object with the most frequent strategy
imputer = SimpleImputer(strategy='most_frequent')

# Select categorical columns excluding the target variable in
train_data
categorical_cols_train =
train_data.drop(columns=['SalePrice']).select_dtypes(include=['object'
]).columns

# Fit the imputer on the training data excluding the target variable
imputer.fit(train_data[categorical_cols_train])

# Transform the training data excluding the target variable
train_data[categorical_cols_train] =
imputer.transform(train_data[categorical_cols_train])

# Select categorical columns in test_data
```

```
categorical_cols_test =
test_data.select_dtypes(include=['object']).columns

# Transform the test data
test_data[categorical_cols_test] =
imputer.transform(test_data[categorical_cols_test])



## comfirmation of the filling of missing categorical columns

import pandas as pd

# Select categorical columns in train_data excluding the target
variable
categorical_cols_train =
train_data.drop(columns=['SalePrice']).select_dtypes(include=['object'
]).columns

# Check for null values in categorical columns of train_data
null_columns_train = train_data[categorical_cols_train].isnull().sum()
null_columns_train = null_columns_train[null_columns_train > 0]
print("Categorical columns with null values in train_data:")
print(null_columns_train)

# Select categorical columns in test_data
categorical_cols_test =
test_data.select_dtypes(include=['object']).columns

# Check for null values in categorical columns of test_data
null_columns_test = test_data[categorical_cols_test].isnull().sum()
null_columns_test = null_columns_test[null_columns_test > 0]
print("Categorical columns with null values in test_data:")
print(null_columns_test)
```

```
Categorical columns with null values in train_data:
Series([], dtype: int64)
Categorical columns with null values in test_data:
Series([], dtype: int64)
```

Label encoding categorical columns to Numerical data so it can fit into the model

```
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Combine train and test data for fitting the encoder
combined_data = pd.concat([train_data.drop(columns=['SalePrice']),
```

```python
test_data])

# Select categorical columns in the combined data
categorical_cols =
combined_data.select_dtypes(include=['object']).columns

# Apply label encoding to each categorical column in the combined data
for col in categorical_cols:
    label_encoder.fit(combined_data[col].astype(str))
    train_data[col] =
label_encoder.transform(train_data[col].astype(str))
    test_data[col] =
label_encoder.transform(test_data[col].astype(str))

# Verify the changes
print("First few rows of the transformed train_data:")
print(train_data.head())

print("First few rows of the transformed test_data:")
print(test_data.head())
```

```
First few rows of the transformed train_data:
    Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  Alley
LotShape  \
0  1.0        60.0         3         65.0   8450.0       1      0
3
1  2.0        20.0         3         80.0   9600.0       1      0
3
2  3.0        60.0         3         68.0  11250.0       1      0
0
3  4.0        70.0         3         60.0   9550.0       1      0
0
4  5.0        60.0         3         84.0  14260.0       1      0
0

   LandContour  Utilities  LotConfig  LandSlope  Neighborhood
Condition1  \
0            3          0          4          0            20
2
1            3          0          2          0            17
1
2            3          0          4          0            20
2
3            3          0          0          0            21
2
4            3          0          2          0             7
2

   Condition2  BldgType  HouseStyle  OverallQual  OverallCond
YearBuilt  \
```

```
0           2         0          10           7.0           5.0
2003.0
1           2         0           5           6.0           8.0
1976.0
2           2         0          10           7.0           5.0
2001.0
3           2         0          10           7.0           5.0
1915.0
4           2         0          10           8.0           5.0
2000.0

   YearRemodAdd  RoofStyle  RoofMatl  Exterior1st  Exterior2nd
MasVnrType  \
0        2003.0          1         1            4            5
1
1        1976.0          1         1           13           14
1
2        2002.0          1         1            4            5
1
3        1970.0          1         1            5            7
1
4        2000.0          1         1            4            5
1

   MasVnrArea  ExterQual  ExterCond  Foundation  BsmtQual  BsmtCond  \
0       196.0          2          4           2         2         3
1         0.0          3          4           1         2         3
2       162.0          2          4           2         2         3
3         0.0          3          4           0         3         1
4       350.0          2          4           2         2         3

   BsmtExposure  BsmtFinType1  BsmtFinSF1  BsmtFinType2  BsmtFinSF2  \
0             3             2       706.0             5         0.0
1             1             0       978.0             5         0.0
2             2             2       486.0             5         0.0
3             3             0       216.0             5         0.0
4             0             2       655.0             5         0.0

   BsmtUnfSF  TotalBsmtSF  Heating  HeatingQC  CentralAir  Electrical
\
0      150.0        856.0        1          0           1           4

1      284.0       1262.0        1          0           1           4

2      434.0        920.0        1          0           1           4

3      540.0        756.0        1          2           1           4

4      490.0       1145.0        1          0           1           4
```

```
   1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  BsmtFullBath
BsmtHalfBath  \
0     856.0     854.0           0.0     1710.0           1.0
0.0
1    1262.0       0.0           0.0     1262.0           0.0
1.0
2     920.0     866.0           0.0     1786.0           1.0
0.0
3     961.0     756.0           0.0     1717.0           1.0
0.0
4    1145.0    1053.0           0.0     2198.0           1.0
0.0

   FullBath  HalfBath  BedroomAbvGr  KitchenAbvGr  KitchenQual
TotRmsAbvGrd  \
0       2.0       1.0           3.0           1.0            2
8.0
1       2.0       0.0           3.0           1.0            3
6.0
2       2.0       1.0           3.0           1.0            2
6.0
3       1.0       0.0           3.0           1.0            2
7.0
4       2.0       1.0           4.0           1.0            2
9.0

   Functional  Fireplaces  FireplaceQu  GarageType  GarageYrBlt
GarageFinish  \
0           6         0.0            4           1       2003.0
1
1           6         1.0            4           1       1976.0
1
2           6         1.0            4           1       2001.0
1
3           6         1.0            2           6       1998.0
2
4           6         1.0            4           1       2000.0
1

   GarageCars  GarageArea  GarageQual  GarageCond  PavedDrive
WoodDeckSF  \
0         2.0       548.0           4           4           2
0.0
1         2.0       460.0           4           4           2
298.0
2         2.0       608.0           4           4           2
0.0
3         3.0       642.0           4           4           2
0.0
```

```
4         3.0       836.0            4           4           2
192.0

   OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea
PoolQC  \
0         61.0            0.0        0.0          0.0       0.0
0
1          0.0            0.0        0.0          0.0       0.0
0
2         42.0            0.0        0.0          0.0       0.0
0
3         35.0          272.0        0.0          0.0       0.0
0
4         84.0            0.0        0.0          0.0       0.0
0

   Fence  MiscFeature  MiscVal  MoSold  YrSold  SaleType
SaleCondition  \
0      2            2      0.0     2.0  2008.0         8
4
1      2            2      0.0     5.0  2007.0         8
4
2      2            2      0.0     9.0  2008.0         8
4
3      2            2      0.0     2.0  2006.0         8
0
4      2            2      0.0    12.0  2008.0         8
4

   SalePrice
0     208500
1     181500
2     223500
3     140000
4     250000
First few rows of the transformed test_data:
       Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street
Alley  \
0  1461.0        20.0         7         80.0  11622.0       3      3

1  1462.0        20.0         8         81.0  14267.0       3      3

2  1463.0        60.0         8         74.0  13830.0       3      3

3  1464.0        60.0         8         78.0   9978.0       3      3

4  1465.0       120.0         8         43.0   5005.0       3      3


   LotShape  LandContour  Utilities  LotConfig  LandSlope
```

```
Neighborhood  \
0         7           7          2          9          3
37
1         4           7          2          5          3
37
2         4           7          2          9          3
33
3         4           7          2          9          3
33
4         4           5          2          9          3
47

   Condition1  Condition2  BldgType  HouseStyle  OverallQual
OverallCond  \
0          10          10         2           4          5.0
6.0
1          11          10         2           4          6.0
6.0
2          11          10         2           7          5.0
5.0
3          11          10         2           7          6.0
6.0
4          11          10         9           4          8.0
5.0

   YearBuilt  YearRemodAdd  RoofStyle  RoofMatl  Exterior1st
Exterior2nd  \
0     1961.0        1961.0          7         8           25
28
1     1958.0        1958.0          9         8           26
29
2     1997.0        1998.0          7         8           25
28
3     1998.0        1998.0          7         8           25
28
4     1992.0        1992.0          7         8           21
22

   MasVnrType  MasVnrArea  ExterQual  ExterCond  Foundation  BsmtQual
\
0           4         0.0          7          9           7         7

1           4       108.0          7          9           7         7

2           4         0.0          7          9           8         6

3           4        20.0          7          9           8         7

4           4         0.0          6          9           8         6
```

```
   BsmtCond  BsmtExposure  BsmtFinType1  BsmtFinSF1  BsmtFinType2  
BsmtFinSF2  \
0         7             7            10       468.0             9   
144.0   
1         7             7             6       923.0            11   
0.0   
2         7             7             8       791.0            11   
0.0   
3         7             7             8       602.0            11   
0.0   
4         7             7             6       263.0            11   
0.0   

   BsmtUnfSF  TotalBsmtSF  Heating  HeatingQC  CentralAir  Electrical  
\
0      270.0        882.0        6          9           3           8   

1      406.0       1329.0        6          9           3           8   

2      137.0        928.0        6          7           3           8   

3      324.0        926.0        6          5           3           8   

4     1017.0       1280.0        6          5           3           8   


   1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  BsmtFullBath  
BsmtHalfBath  \
0     896.0       0.0           0.0      896.0           0.0   
0.0   
1    1329.0       0.0           0.0     1329.0           0.0   
0.0   
2     928.0     701.0           0.0     1629.0           0.0   
0.0   
3     926.0     678.0           0.0     1604.0           0.0   
0.0   
4    1280.0       0.0           0.0     1280.0           0.0   
0.0   

   FullBath  HalfBath  BedroomAbvGr  KitchenAbvGr  KitchenQual  
TotRmsAbvGrd  \
0       1.0       0.0           2.0           1.0            7   
5.0   
1       1.0       1.0           3.0           1.0            6   
6.0   
2       2.0       1.0           3.0           1.0            7   
6.0   
3       2.0       1.0           3.0           1.0            6   
7.0   
```

```
4       2.0       0.0           2.0           1.0            6
5.0

   Functional  Fireplaces  FireplaceQu  GarageType  GarageYrBlt
GarageFinish  \
0          13         0.0            9           7       1961.0
5
1          13         0.0            9           7       1958.0
5
2          13         1.0            9           7       1997.0
3
3          13         1.0            7           7       1998.0
3
4          13         0.0            9           7       1992.0
4

   GarageCars  GarageArea  GarageQual  GarageCond  PavedDrive
WoodDeckSF  \
0         1.0       730.0           8           9           5
140.0
1         1.0       312.0           8           9           5
393.0
2         2.0       482.0           8           9           5
212.0
3         2.0       470.0           8           9           5
360.0
4         2.0       506.0           8           9           5
0.0

   OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea
PoolQC  \
0          0.0            0.0        0.0        120.0       0.0
3
1         36.0            0.0        0.0          0.0       0.0
3
2         34.0            0.0        0.0          0.0       0.0
3
3         36.0            0.0        0.0          0.0       0.0
3
4         82.0            0.0        0.0        144.0       0.0
3

   Fence  MiscFeature  MiscVal  MoSold  YrSold  SaleType
SaleCondition
0      6            4      0.0     6.0  2010.0        17
10
1      6            4  12500.0     6.0  2010.0        17
10
2      6            6      0.0     3.0  2010.0        17
10
```

```
3       4             6       0.0      6.0  2010.0        17
10
4       4             6       0.0      1.0  2010.0        17
10
```

OUTLIER DETECTION

it was detected that the both datasets contains outleirs which if not carefully handled, can result to skewed predictions

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = train_data.quantile(0.25)
Q3 = train_data.quantile(0.75)
IQR = Q3 - Q1

# Define outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = ((train_data < lower_bound) | (train_data >
upper_bound)).any(axis=1)
outlier_indices = train_data[outliers].index

train_data.head()

    Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  Alley
LotShape  \
0  1.0        60.0         3         65.0   8450.0       1      0
3
1  2.0        20.0         3         80.0   9600.0       1      0
3
2  3.0        60.0         3         68.0  11250.0       1      0
0
3  4.0        70.0         3         60.0   9550.0       1      0
0
4  5.0        60.0         3         84.0  14260.0       1      0
0

   LandContour  Utilities  LotConfig  LandSlope  Neighborhood
Condition1  \
0            3          0          4          0            20
2
```

```
1             3          0          2          0           17
1
2             3          0          4          0           20
2
3             3          0          0          0           21
2
4             3          0          2          0            7
2

   Condition2  BldgType  HouseStyle  OverallQual  OverallCond
YearBuilt  \
0           2         0          10          7.0          5.0
2003.0
1           2         0           5          6.0          8.0
1976.0
2           2         0          10          7.0          5.0
2001.0
3           2         0          10          7.0          5.0
1915.0
4           2         0          10          8.0          5.0
2000.0

   YearRemodAdd  RoofStyle  RoofMatl  Exterior1st  Exterior2nd
MasVnrType  \
0        2003.0          1         1            4            5
1
1        1976.0          1         1           13           14
1
2        2002.0          1         1            4            5
1
3        1970.0          1         1            5            7
1
4        2000.0          1         1            4            5
1

   MasVnrArea  ExterQual  ExterCond  Foundation  BsmtQual  BsmtCond  \
0       196.0          2          4           2         2         3
1         0.0          3          4           1         2         3
2       162.0          2          4           2         2         3
3         0.0          3          4           0         3         1
4       350.0          2          4           2         2         3

   BsmtExposure  BsmtFinType1  BsmtFinSF1  BsmtFinType2  BsmtFinSF2  \
0             3             2       706.0             5         0.0
1             1             0       978.0             5         0.0
2             2             2       486.0             5         0.0
3             3             0       216.0             5         0.0
4             0             2       655.0             5         0.0

   BsmtUnfSF  TotalBsmtSF  Heating  HeatingQC  CentralAir  Electrical
```

```
\
0     150.0        856.0        1          0          1          4

1     284.0       1262.0        1          0          1          4

2     434.0        920.0        1          0          1          4

3     540.0        756.0        1          2          1          4

4     490.0       1145.0        1          0          1          4


   1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  BsmtFullBath
BsmtHalfBath  \
0     856.0     854.0           0.0     1710.0           1.0
0.0
1    1262.0       0.0           0.0     1262.0           0.0
1.0
2     920.0     866.0           0.0     1786.0           1.0
0.0
3     961.0     756.0           0.0     1717.0           1.0
0.0
4    1145.0    1053.0           0.0     2198.0           1.0
0.0

   FullBath  HalfBath  BedroomAbvGr  KitchenAbvGr  KitchenQual
TotRmsAbvGrd  \
0       2.0       1.0           3.0           1.0            2
8.0
1       2.0       0.0           3.0           1.0            3
6.0
2       2.0       1.0           3.0           1.0            2
6.0
3       1.0       0.0           3.0           1.0            2
7.0
4       2.0       1.0           4.0           1.0            2
9.0

   Functional  Fireplaces  FireplaceQu  GarageType  GarageYrBlt
GarageFinish  \
0           6         0.0            4           1       2003.0
1
1           6         1.0            4           1       1976.0
1
2           6         1.0            4           1       2001.0
1
3           6         1.0            2           6       1998.0
2
4           6         1.0            4           1       2000.0
1
```

```
   GarageCars  GarageArea  GarageQual  GarageCond  PavedDrive
WoodDeckSF  \
0         2.0       548.0           4           4           2
0.0
1         2.0       460.0           4           4           2
298.0
2         2.0       608.0           4           4           2
0.0
3         3.0       642.0           4           4           2
0.0
4         3.0       836.0           4           4           2
192.0

   OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea
PoolQC  \
0         61.0            0.0        0.0          0.0       0.0
0
1          0.0            0.0        0.0          0.0       0.0
0
2         42.0            0.0        0.0          0.0       0.0
0
3         35.0          272.0        0.0          0.0       0.0
0
4         84.0            0.0        0.0          0.0       0.0
0

   Fence  MiscFeature  MiscVal  MoSold  YrSold  SaleType
SaleCondition  \
0      2            2      0.0     2.0  2008.0         8
4
1      2            2      0.0     5.0  2007.0         8
4
2      2            2      0.0     9.0  2008.0         8
4
3      2            2      0.0     2.0  2006.0         8
0
4      2            2      0.0    12.0  2008.0         8
4

   SalePrice
0     208500
1     181500
2     223500
3     140000
4     250000
```

outleir detection for train_dataset

to see columns with outleirs in the train_data set

```
import pandas as pd

# Function to detect outliers using the IQR method
def detect_outliers(df):
    outliers = {}
    for column in df.select_dtypes(include=['int64',
'float64']).columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outlier_indices = df[(df[column] < lower_bound) | (df[column]
> upper_bound)].index
        if not outlier_indices.empty:
            outliers[column] = len(outlier_indices)
    return outliers

# Detect outliers in the training data
outliers = detect_outliers(train_data)

# Print outliers in the desired format
print(f"{'column name':<25} {'outlier found':<15}")
for column, count in outliers.items():
    print(f"{column:<25} {count:<15}")
```

```
column name               outlier found
MSSubClass                103
LotFrontage               106
LotArea                   69
OverallQual               2
OverallCond               125
YearBuilt                 7
MasVnrArea                98
BsmtFinSF1                7
BsmtFinSF2                167
BsmtUnfSF                 29
TotalBsmtSF               61
1stFlrSF                  20
2ndFlrSF                  2
LowQualFinSF              26
GrLivArea                 31
BsmtFullBath              1
BsmtHalfBath              82
BedroomAbvGr              35
KitchenAbvGr              68
TotRmsAbvGrd              30
```

```
Fireplaces                          5
GarageYrBlt                         1
GarageCars                          5
GarageArea                          21
WoodDeckSF                          32
OpenPorchSF                         77
EnclosedPorch                       208
3SsnPorch                           24
ScreenPorch                         116
PoolArea                            7
MiscVal                             52
SalePrice                           61
```

```python
#### outleir detection for test_data set

###### to see columns in the test_data with outliers

import pandas as pd

# Function to detect outliers using the IQR method
def detect_outliers(df):
    outliers = {}
    for column in df.select_dtypes(include=['int64',
'float64']).columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outlier_indices = df[(df[column] < lower_bound) | (df[column]
> upper_bound)].index
        if not outlier_indices.empty:
            outliers[column] = len(outlier_indices)
    return outliers

# Detect outliers in the training data
outliers = detect_outliers(test_data)

# Print outliers in the desired format
print(f"{'column name':<25} {'outlier found':<15}")
for column, count in outliers.items():
    print(f"{column:<25} {count:<15}")
```

```
column name                         outlier found
MSSubClass                          104
LotFrontage                         141
LotArea                             60
OverallQual                         2
OverallCond                         127
```

```
YearBuilt                  2
MasVnrArea                 104
BsmtFinSF1                 8
BsmtFinSF2                 181
BsmtUnfSF                  26
TotalBsmtSF                61
1stFlrSF                   23
2ndFlrSF                   5
LowQualFinSF               14
GrLivArea                  44
BsmtFullBath               1
BsmtHalfBath               95
FullBath                   4
BedroomAbvGr               43
KitchenAbvGr               66
TotRmsAbvGrd               21
Fireplaces                 7
GarageYrBlt                3
GarageCars                 12
GarageArea                 21
WoodDeckSF                 35
OpenPorchSF                79
EnclosedPorch              251
3SsnPorch                  13
ScreenPorch                140
PoolArea                   6
MiscVal                    51
```

HANDLING OUTLIERS (CAPPING)

“Capping” outliers means replacing extreme values with a specified limit to reduce their impact on the dataset. Instead of removing outliers, which can result in data loss, capping modifies the outliers to fall within a defined range. This approach retains all data points but adjusts the extreme values to be less influential.

Based on the lists provided, the FullBath column has outliers in the test_data but not in the train_data. The capping of the outliers will be handled separately with the columns in the lists .

```
def cap_outliers(df, columns):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df[column] = df[column].apply(lambda x: lower_bound if x <
```

```
lower_bound else upper_bound if x > upper_bound else x)
    return df

# List of columns to cap in train_data
columns_to_cap = [
    'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
'OverallCond', 'YearBuilt',
    'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF',
    '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
'BsmtHalfBath', 'BedroomAbvGr',
    'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt',
'GarageCars', 'GarageArea',
    'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
'ScreenPorch', 'PoolArea',
    'MiscVal'
]

# Cap outliers in the specified columns of train_data
train_data_capped =
cap_outliers(train_data.drop(columns=['SalePrice']), columns_to_cap)

# Add the SalePrice column back to the capped training data
train_data_capped['SalePrice'] = train_data['SalePrice']

# Verify the changes
print(f"Original train_data shape: {train_data.shape}")
print(f"Capped train_data shape: {train_data_capped.shape}")
```

```
Original train_data shape: (1460, 81)
Capped train_data shape: (1460, 81)
```

```
def cap_outliers(df, columns):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df[column] = df[column].apply(lambda x: lower_bound if x <
lower_bound else upper_bound if x > upper_bound else x)
    return df

# List of columns to cap in test_data
columns_to_cap_test = [
    'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
'OverallCond', 'YearBuilt',
    'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF',
    '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
```

```
'BsmtHalfBath', 'FullBath',
    'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
'GarageYrBlt', 'GarageCars',
    'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch',
'3SsnPorch', 'ScreenPorch',
    'PoolArea', 'MiscVal'
]

# Cap outliers in the specified columns of test_data
test_data_capped = cap_outliers(test_data, columns_to_cap_test)

# Verify the changes
print(f"Original test_data shape: {test_data.shape}")
print(f"Capped test_data shape: {test_data_capped.shape}")
```

```
Original test_data shape: (1459, 80)
Capped test_data shape: (1459, 80)
```

```
# Compare original and capped values for a few columns
columns_to_check = ['LotFrontage', 'GrLivArea', 'FullBath']  # Add
more columns as needed

for column in columns_to_check:
    original_values = test_data[column].head(10)  # Display first 10
values for comparison
    capped_values = test_data_capped[column].head(10)
    print(f"Original {column} values:\n{original_values}")
    print(f"Capped {column} values:\n{capped_values}\n")
```

```
Original LotFrontage values:
0    80.000000
1    81.000000
2    74.000000
3    78.000000
4    43.000000
5    75.000000
6    70.049958
7    63.000000
8    85.000000
9    70.000000
Name: LotFrontage, dtype: float64
Capped LotFrontage values:
0    80.000000
1    81.000000
2    74.000000
3    78.000000
4    43.000000
5    75.000000
6    70.049958
7    63.000000
```

```
8    85.000000
9    70.000000
Name: LotFrontage, dtype: float64

Original GrLivArea values:
0     896.0
1    1329.0
2    1629.0
3    1604.0
4    1280.0
5    1655.0
6    1187.0
7    1465.0
8    1341.0
9     882.0
Name: GrLivArea, dtype: float64
Capped GrLivArea values:
0     896.0
1    1329.0
2    1629.0
3    1604.0
4    1280.0
5    1655.0
6    1187.0
7    1465.0
8    1341.0
9     882.0
Name: GrLivArea, dtype: float64

Original FullBath values:
0    1.0
1    1.0
2    2.0
3    2.0
4    2.0
5    2.0
6    2.0
7    2.0
8    1.0
9    1.0
Name: FullBath, dtype: float64
Capped FullBath values:
0    1.0
1    1.0
2    2.0
3    2.0
4    2.0
5    2.0
6    2.0
```

```
7    2.0
8    1.0
9    1.0
Name: FullBath, dtype: float64


# Summary statistics before capping
print("Summary statistics before capping:")
print(test_data[columns_to_check].describe())

# Summary statistics after capping
print("Summary statistics after capping:")
print(test_data_capped[columns_to_check].describe())

Summary statistics before capping:
       LotFrontage    GrLivArea     FullBath
count  1459.000000  1459.000000  1459.000000
mean     68.423126  1478.000685     1.569568
std      17.164354   457.873870     0.549778
min      33.000000   407.000000     0.000000
25%      60.000000  1117.500000     1.000000
50%      70.049958  1432.000000     2.000000
75%      78.000000  1721.000000     2.000000
max     105.000000  2626.250000     3.500000
Summary statistics after capping:
       LotFrontage    GrLivArea     FullBath
count  1459.000000  1459.000000  1459.000000
mean     68.423126  1478.000685     1.569568
std      17.164354   457.873870     0.549778
min      33.000000   407.000000     0.000000
25%      60.000000  1117.500000     1.000000
50%      70.049958  1432.000000     2.000000
75%      78.000000  1721.000000     2.000000
max     105.000000  2626.250000     3.500000

def cap_outliers_verbose(df, columns):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        print(f"{column}: Lower Bound = {lower_bound}, Upper Bound =
{upper_bound}")
        df[column] = df[column].apply(lambda x: lower_bound if x <
lower_bound else upper_bound if x > upper_bound else x)
    return df

# Cap outliers in the specified columns of test_data with verbose
output
```

```
test_data_capped_verbose = cap_outliers_verbose(test_data.copy(),
columns_to_cap_test)

# Verify the changes
print(f"Original test_data shape: {test_data.shape}")
print(f"Capped test_data shape: {test_data_capped_verbose.shape}")
```

```
MSSubClass: Lower Bound = -55.0, Upper Bound = 145.0
LotFrontage: Lower Bound = 33.0, Upper Bound = 105.0
LotArea: Lower Bound = 1201.25, Upper Bound = 17707.25
OverallQual: Lower Bound = 2.0, Upper Bound = 10.0
OverallCond: Lower Bound = 3.5, Upper Bound = 7.5
YearBuilt: Lower Bound = 1881.0, Upper Bound = 2073.0
MasVnrArea: Lower Bound = -243.0, Upper Bound = 405.0
BsmtFinSF1: Lower Bound = -1128.0, Upper Bound = 1880.0
BsmtFinSF2: Lower Bound = 0.0, Upper Bound = 0.0
BsmtUnfSF: Lower Bound = -647.5, Upper Bound = 1664.5
TotalBsmtSF: Lower Bound = 4.0, Upper Bound = 2084.0
1stFlrSF: Lower Bound = 110.0, Upper Bound = 2146.0
2ndFlrSF: Lower Bound = -1014.0, Upper Bound = 1690.0
LowQualFinSF: Lower Bound = 0.0, Upper Bound = 0.0
GrLivArea: Lower Bound = 212.25, Upper Bound = 2626.25
BsmtFullBath: Lower Bound = -1.5, Upper Bound = 2.5
BsmtHalfBath: Lower Bound = 0.0, Upper Bound = 0.0
FullBath: Lower Bound = -0.5, Upper Bound = 3.5
BedroomAbvGr: Lower Bound = 0.5, Upper Bound = 4.5
KitchenAbvGr: Lower Bound = 1.0, Upper Bound = 1.0
TotRmsAbvGrd: Lower Bound = 2.0, Upper Bound = 10.0
Fireplaces: Lower Bound = -1.5, Upper Bound = 2.5
GarageYrBlt: Lower Bound = 1899.75, Upper Bound = 2061.75
GarageCars: Lower Bound = -0.5, Upper Bound = 3.5
GarageArea: Lower Bound = -69.0, Upper Bound = 963.0
WoodDeckSF: Lower Bound = -252.0, Upper Bound = 420.0
OpenPorchSF: Lower Bound = -108.0, Upper Bound = 180.0
EnclosedPorch: Lower Bound = 0.0, Upper Bound = 0.0
3SsnPorch: Lower Bound = 0.0, Upper Bound = 0.0
ScreenPorch: Lower Bound = 0.0, Upper Bound = 0.0
PoolArea: Lower Bound = 0.0, Upper Bound = 0.0
MiscVal: Lower Bound = 0.0, Upper Bound = 0.0
Original test_data shape: (1459, 80)
Capped test_data shape: (1459, 80)
```

```
def check_outliers(df, columns):
    outliers = {}
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
```

```
        outlier_indices = df[(df[column] < lower_bound) | (df[column]
> upper_bound)].index
        outliers[column] = len(outlier_indices)
    return outliers

# Check for outliers in the specified columns of test_data
outliers_before_capping = check_outliers(test_data,
columns_to_cap_test)

# Print outliers in the desired format
print(f"{'Column Name':<25} {'Outliers Found':<15}")
for column, count in outliers_before_capping.items():
    print(f"{column:<25} {count:<15}")
```

```
Column Name               Outliers Found
MSSubClass                0
LotFrontage               0
LotArea                   0
OverallQual               0
OverallCond               0
YearBuilt                 0
MasVnrArea                0
BsmtFinSF1                0
BsmtFinSF2                0
BsmtUnfSF                 0
TotalBsmtSF               0
1stFlrSF                  0
2ndFlrSF                  0
LowQualFinSF              0
GrLivArea                 0
BsmtFullBath              0
BsmtHalfBath              0
FullBath                  0
BedroomAbvGr              0
KitchenAbvGr              0
TotRmsAbvGrd              0
Fireplaces                0
GarageYrBlt               0
GarageCars                0
GarageArea                0
WoodDeckSF                0
OpenPorchSF               0
EnclosedPorch             0
3SsnPorch                 0
ScreenPorch               0
PoolArea                  0
MiscVal                   0
```

```
def cap_outliers_verbose(df, columns):
    for column in columns:
```

```
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        print(f"{column}: Lower Bound = {lower_bound}, Upper Bound =
{upper_bound}")
        df[column] = df[column].apply(lambda x: lower_bound if x <
lower_bound else upper_bound if x > upper_bound else x)
    return df

# Cap outliers in the specified columns of test_data with verbose
output
train_data_capped_verbose = cap_outliers_verbose(train_data.copy(),
columns_to_cap_test)

# Verify the changes
print(f"Original test_data shape: {train_data.shape}")
print(f"Capped test_data shape: {train_data_capped_verbose.shape}")
```

```
MSSubClass: Lower Bound = -55.0, Upper Bound = 145.0
LotFrontage: Lower Bound = 31.5, Upper Bound = 107.5
LotArea: Lower Bound = 1481.5, Upper Bound = 17673.5
OverallQual: Lower Bound = 2.0, Upper Bound = 10.0
OverallCond: Lower Bound = 3.5, Upper Bound = 7.5
YearBuilt: Lower Bound = 1885.0, Upper Bound = 2069.0
MasVnrArea: Lower Bound = -246.375, Upper Bound = 410.625
BsmtFinSF1: Lower Bound = -1068.375, Upper Bound = 1780.625
BsmtFinSF2: Lower Bound = 0.0, Upper Bound = 0.0
BsmtUnfSF: Lower Bound = -654.5, Upper Bound = 1685.5
TotalBsmtSF: Lower Bound = 42.0, Upper Bound = 2052.0
1stFlrSF: Lower Bound = 118.125, Upper Bound = 2155.125
2ndFlrSF: Lower Bound = -1092.0, Upper Bound = 1820.0
LowQualFinSF: Lower Bound = 0.0, Upper Bound = 0.0
GrLivArea: Lower Bound = 158.625, Upper Bound = 2747.625
BsmtFullBath: Lower Bound = -1.5, Upper Bound = 2.5
BsmtHalfBath: Lower Bound = 0.0, Upper Bound = 0.0
FullBath: Lower Bound = -0.5, Upper Bound = 3.5
BedroomAbvGr: Lower Bound = 0.5, Upper Bound = 4.5
KitchenAbvGr: Lower Bound = 1.0, Upper Bound = 1.0
TotRmsAbvGrd: Lower Bound = 2.0, Upper Bound = 10.0
Fireplaces: Lower Bound = -1.5, Upper Bound = 2.5
GarageYrBlt: Lower Bound = 1903.5, Upper Bound = 2059.5
GarageCars: Lower Bound = -0.5, Upper Bound = 3.5
GarageArea: Lower Bound = -27.75, Upper Bound = 938.25
WoodDeckSF: Lower Bound = -252.0, Upper Bound = 420.0
OpenPorchSF: Lower Bound = -102.0, Upper Bound = 170.0
EnclosedPorch: Lower Bound = 0.0, Upper Bound = 0.0
3SsnPorch: Lower Bound = 0.0, Upper Bound = 0.0
ScreenPorch: Lower Bound = 0.0, Upper Bound = 0.0
```

```
PoolArea: Lower Bound = 0.0, Upper Bound = 0.0
MiscVal: Lower Bound = 0.0, Upper Bound = 0.0
Original test_data shape: (1460, 81)
Capped test_data shape: (1460, 81)
```

```
def check_outliers(df, columns):
    outliers = {}
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outlier_indices = df[(df[column] < lower_bound) | (df[column]
> upper_bound)].index
        outliers[column] = len(outlier_indices)
    return outliers

# Check for outliers in the specified columns of test_data
outliers_before_capping = check_outliers(train_data,
columns_to_cap_test)

# Print outliers in the desired format
print(f"{'Column Name':<25} {'Outliers Found':<15}")
for column, count in outliers_before_capping.items():
    print(f"{column:<25} {count:<15}")
```

```
Column Name               Outliers Found
MSSubClass                103
LotFrontage               106
LotArea                   69
OverallQual               2
OverallCond               125
YearBuilt                 7
MasVnrArea                98
BsmtFinSF1                7
BsmtFinSF2                167
BsmtUnfSF                 29
TotalBsmtSF               61
1stFlrSF                  20
2ndFlrSF                  2
LowQualFinSF              26
GrLivArea                 31
BsmtFullBath              1
BsmtHalfBath              82
FullBath                  0
BedroomAbvGr              35
KitchenAbvGr              68
TotRmsAbvGrd              30
Fireplaces                5
```

```
GarageYrBlt                 1
GarageCars                  5
GarageArea                  21
WoodDeckSF                  32
OpenPorchSF                 77
EnclosedPorch               208
3SsnPorch                   24
ScreenPorch                 116
PoolArea                    7
MiscVal                     52
```

Correlation Analysis

to accurately understand the relationships between features, leading to better feature selection and model performance.

```python
import pandas as pd



# Add SalePrice to the capped DataFrame for correlation analysis
train_data_capped_verbose['SalePrice'] = train_data['SalePrice']

# Calculate the correlation matrix
correlation_matrix = train_data_capped_verbose.corr()

# Extract the correlation with SalePrice
correlation_with_saleprice = correlation_matrix['SalePrice']

# Display the correlation values
print(correlation_with_saleprice)
```

```
Id              -0.021917
MSSubClass      -0.063602
MSZoning        -0.166872
LotFrontage      0.371558
LotArea          0.432216
Street           0.041036
Alley            0.037646
LotShape        -0.255580
LandContour      0.015453
Utilities       -0.014314
LotConfig       -0.067396
LandSlope        0.051152
Neighborhood     0.009118
Condition1       0.091155
Condition2       0.007513
BldgType        -0.084931
HouseStyle       0.206210
OverallQual      0.791965
```

```
OverallCond     -0.106261
YearBuilt        0.524172
YearRemodAdd     0.507101
RoofStyle        0.222405
RoofMatl         0.132383
Exterior1st     -0.163240
Exterior2nd     -0.103815
MasVnrType       0.113287
MasVnrArea       0.452127
ExterQual       -0.636884
ExterCond        0.117303
Foundation       0.382479
BsmtQual        -0.581349
BsmtCond         0.065844
BsmtExposure    -0.276932
BsmtFinType1    -0.072068
BsmtFinSF1       0.400330
BsmtFinType2     0.037640
BsmtFinSF2            NaN
BsmtUnfSF        0.203278
TotalBsmtSF      0.636999
Heating         -0.098812
HeatingQC       -0.400178
CentralAir       0.251328
Electrical       0.234760
1stFlrSF         0.620743
2ndFlrSF         0.316547
LowQualFinSF          NaN
GrLivArea        0.708153
BsmtFullBath     0.227813
BsmtHalfBath          NaN
FullBath         0.560664
HalfBath         0.284108
BedroomAbvGr     0.185740
KitchenAbvGr          NaN
KitchenQual     -0.589189
TotRmsAbvGrd     0.536067
Functional       0.115328
Fireplaces       0.468700
FireplaceQu     -0.079038
GarageType      -0.321603
GarageYrBlt      0.470269
GarageFinish    -0.485097
GarageCars       0.644002
GarageArea       0.630138
GarageQual       0.107677
GarageCond       0.135823
PavedDrive       0.231357
WoodDeckSF       0.330378
```

```
OpenPorchSF      0.369024
EnclosedPorch         NaN
3SsnPorch             NaN
ScreenPorch           NaN
PoolArea              NaN
PoolQC          -0.054580
Fence           -0.019741
MiscFeature      0.012167
MiscVal               NaN
MoSold           0.046432
YrSold          -0.028923
SaleType        -0.054911
SaleCondition    0.213092
SalePrice        1.000000
Name: SalePrice, dtype: float64
```

```
train_data_capped_verbose.isnull().sum()
```

```
Id               0
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
Street           0
Alley            0
LotShape         0
LandContour      0
Utilities        0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
MasVnrArea       0
ExterQual        0
ExterCond        0
Foundation       0
```

```
BsmtQual         0
BsmtCond         0
BsmtExposure     0
BsmtFinType1     0
BsmtFinSF1       0
BsmtFinType2     0
BsmtFinSF2       0
BsmtUnfSF        0
TotalBsmtSF      0
Heating          0
HeatingQC        0
CentralAir       0
Electrical       0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
KitchenQual      0
TotRmsAbvGrd     0
Functional       0
Fireplaces       0
FireplaceQu      0
GarageType       0
GarageYrBlt      0
GarageFinish     0
GarageCars       0
GarageArea       0
GarageQual       0
GarageCond       0
PavedDrive       0
WoodDeckSF       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
PoolQC           0
Fence            0
MiscFeature      0
MiscVal          0
MoSold           0
YrSold           0
SaleType         0
```

```
SaleCondition    0
SalePrice        0
dtype: int64
```

Dropping Columns with constant Values (A column with constant values has the same value for every row in the dataset. Example: If a column X has the value '20' for all rows, it is a constant column.

from the correlation results, found out the data contains columns with constant values which should be removed because

No Variability: Since the values do not change, the column does not provide any useful information for analysis.

Correlation: The correlation of a constant column with any other variable is undefined or zero because there is no variability to compare.

Model Performance: Including constant columns in a model does not add any predictive power and can be safely removed.

```
# Identify columns with constant values in train_data_capped_verbose
constant_columns_train = [col for col in
train_data_capped_verbose.columns if
train_data_capped_verbose[col].nunique() == 1]

# Display the constant columns
print("Columns with constant values in train_data_capped_verbose:",
constant_columns_train)

Columns with constant values in train_data_capped_verbose:
['BsmtFinSF2', 'LowQualFinSF', 'BsmtHalfBath', 'KitchenAbvGr',
'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal']

# Drop columns with constant values from train_data_capped_verbose
train_data_capped_verbose.drop(columns=constant_columns_train,
inplace=True)

# Verify the changes
print("Remaining columns in train_data_capped_verbose:",
train_data_capped_verbose.columns)

Remaining columns in train_data_capped_verbose: Index(['Id',
'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType',
```

```
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation',
'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
'HeatingQC',
       'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
'GrLivArea',
       'BsmtFullBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu',
'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea',
'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
'PoolQC',
       'Fence', 'MiscFeature', 'MoSold', 'YrSold', 'SaleType',
'SaleCondition',
       'SalePrice'],
      dtype='object')
```

```
train_data_capped_verbose.shape
```

```
(1460, 72)
```

```
# Identify columns with constant values
constant_columns = [col for col in test_data_capped_verbose.columns if
test_data_capped_verbose[col].nunique() == 1]

# Display the constant columns
print("Columns with constant values:", constant_columns)
```

```
Columns with constant values: ['Utilities', 'BsmtFinSF2',
'LowQualFinSF', 'BsmtHalfBath', 'KitchenAbvGr', 'EnclosedPorch',
'3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal']
```

```
# Drop columns with constant values
test_data_capped_verbose.drop(columns=constant_columns, inplace=True)

# Verify the changes
print("Remaining columns:", test_data_capped_verbose.columns)
```

```
Remaining columns: Index(['Id', 'MSSubClass', 'MSZoning',
'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
'HouseStyle',
       'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
```

```
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
       'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
'CentralAir',
       'Electrical', '1stFlrSF', '2ndFlrSF', 'GrLivArea',
'BsmtFullBath',
       'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenQual',
'TotRmsAbvGrd',
       'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'PoolQC', 'Fence',
       'MiscFeature', 'MoSold', 'YrSold', 'SaleType',
'SaleCondition'],
      dtype='object')
```

```
test_data_capped_verbose.shape
```

```
(1459, 70)
```

Dropping columns with low correlation to the Target Variable (SalePrice)

Correlation Strength:: Strong Correlation: Typically, correlations above 0.5 (positive or negative)

```
import pandas as pd

# Calculate the correlation matrix for the train dataset
corr_matrix = train_data_capped_verbose.corr()

# Set the correlation threshold
threshold = 0.5

# Get features with a correlation above or below the threshold for the
train dataset
correlated_features = corr_matrix.index[(corr_matrix["SalePrice"] >=
threshold) | (corr_matrix["SalePrice"] <= -threshold)]

# Filter the train dataset
train_data_filtered = train_data_capped_verbose[correlated_features]

# Filter the test dataset using the same correlated features
(excluding 'SalePrice' if it exists)
correlated_features_test = correlated_features.drop('SalePrice',
errors='ignore')
test_data_filtered =
test_data_capped_verbose[correlated_features_test]
```

```python
# Display the first few rows of the filtered train and test datasets
print(train_data_filtered.head())
print(test_data_filtered.head())
```

```
   OverallQual  YearBuilt  YearRemodAdd  ExterQual  BsmtQual
TotalBsmtSF  \
0          7.0     2003.0        2003.0          2         2
856.0
1          6.0     1976.0        1976.0          3         2
1262.0
2          7.0     2001.0        2002.0          2         2
920.0
3          7.0     1915.0        1970.0          3         3
756.0
4          8.0     2000.0        2000.0          2         2
1145.0

   1stFlrSF  GrLivArea  FullBath  KitchenQual  TotRmsAbvGrd
GarageCars  \
0     856.0     1710.0       2.0            2           8.0
2.0
1    1262.0     1262.0       2.0            3           6.0
2.0
2     920.0     1786.0       2.0            2           6.0
2.0
3     961.0     1717.0       1.0            2           7.0
3.0
4    1145.0     2198.0       2.0            2           9.0
3.0

   GarageArea  SalePrice
0       548.0     208500
1       460.0     181500
2       608.0     223500
3       642.0     140000
4       836.0     250000
   OverallQual  YearBuilt  YearRemodAdd  ExterQual  BsmtQual
TotalBsmtSF  \
0          5.0     1961.0        1961.0          7         7
882.0
1          6.0     1958.0        1958.0          7         7
1329.0
2          5.0     1997.0        1998.0          7         6
928.0
3          6.0     1998.0        1998.0          7         7
926.0
4          8.0     1992.0        1992.0          6         6
1280.0

   1stFlrSF  GrLivArea  FullBath  KitchenQual  TotRmsAbvGrd
```

```
GarageCars  \
0     896.0      896.0       1.0             7           5.0
1.0
1    1329.0     1329.0       1.0             6           6.0
1.0
2     928.0     1629.0       2.0             7           6.0
2.0
3     926.0     1604.0       2.0             6           7.0
2.0
4    1280.0     1280.0       2.0             6           5.0
2.0

   GarageArea
0       730.0
1       312.0
2       482.0
3       470.0
4       506.0
```

```
train_data_filtered.shape
```

```
(1460, 14)
```

```
test_data_filtered.shape
```

```
(1459, 13)
```

```
train_data_filtered.head()
```

```
   OverallQual  YearBuilt  YearRemodAdd  ExterQual  BsmtQual
TotalBsmtSF  \
0          7.0     2003.0        2003.0          2         2
856.0
1          6.0     1976.0        1976.0          3         2
1262.0
2          7.0     2001.0        2002.0          2         2
920.0
3          7.0     1915.0        1970.0          3         3
756.0
4          8.0     2000.0        2000.0          2         2
1145.0

   1stFlrSF  GrLivArea  FullBath  KitchenQual  TotRmsAbvGrd
GarageCars  \
0     856.0     1710.0       2.0             2           8.0
2.0
1    1262.0     1262.0       2.0             3           6.0
2.0
2     920.0     1786.0       2.0             2           6.0
2.0
3     961.0     1717.0       1.0             2           7.0
```

```
3.0
4    1145.0     2198.0       2.0             2            9.0
3.0

   GarageArea  SalePrice
0       548.0     208500
1       460.0     181500
2       608.0     223500
3       642.0     140000
4       836.0     250000
```

Scalling the data and Splitiing the data

1. Equal Contribution of Features Scaling ensures that all features contribute equally to the model. Without scaling, features with larger ranges can dominate the model's performance, leading to biased results1.

2. Improved Model Performance Many machine learning algorithms, including Linear Regression, perform better when the data is scaled. This is because these algorithms often rely on distance calculations, and unscaled data can distort these distances1.

3. Faster Convergence Scaling can speed up the convergence of gradient descent, which is used in many optimization algorithms. This results in faster training times and more efficient model building1.

4. Consistency in Data Scaling helps maintain consistency in the data, making it easier to compare and interpret the results. This is particularly important when dealing with features that have different units or scales1

```python
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Initialize the scaler
scaler = StandardScaler()

# Separate the features (X) and target (y)
X_train = train_data_filtered.drop(columns=['SalePrice'])
y_train = train_data_filtered['SalePrice']

# Standardize the numeric features in the training set
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train),
columns=X_train.columns)

# Standardize the numeric features in the test set
X_test_scaled = pd.DataFrame(scaler.transform(test_data_filtered),
columns=X_train.columns)  # Use X_train's columns for consistency

# Check the first few rows of the standardized training data
print(X_train_scaled.head())
```

```
   OverallQual  YearBuilt  YearRemodAdd  ExterQual  BsmtQual  
TotalBsmtSF  \
0     0.652644   1.053246      0.878668  -0.777976 -0.290552    -
0.488321   
1    -0.073068   0.156179     -0.429577   0.663451 -0.290552     
0.532289   
2     0.652644   0.986797      0.830215  -0.777976 -0.290552    -
0.327437   
3     0.652644  -1.870528     -0.720298   0.663451  0.852861    -
0.739702   
4     1.378355   0.953572      0.733308  -0.777976 -0.290552     
0.238172   

   1stFlrSF  GrLivArea  FullBath  KitchenQual  TotRmsAbvGrd  
GarageCars  \
0 -0.830489   0.428636  0.789741    -0.409369      0.981148    
0.315946   
1  0.289638  -0.502349  0.789741     0.795629     -0.316385    
0.315946   
2 -0.653917   0.586571  0.789741    -0.409369     -0.316385    
0.315946   
3 -0.540801   0.443182 -1.026041    -0.409369      0.332382    
1.662750   
4 -0.033157   1.442744  0.789741    -0.409369      1.629914    
1.662750   

   GarageArea  
0    0.373509  
1   -0.051541  
2    0.663315  
3    0.827539  
4    1.764579  
```

Importing the simple Linear Regression model for the FIT, training and testing .

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Initialize the Linear Regression model
model = LinearRegression()

# Train the model on standardized data
model.fit(X_train_scaled, y_train)
```

```
# Predict the test data
y_pred = model.predict(X_test_scaled)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Assuming your features and target variable are correctly assigned
X_train = X_train_scaled   ##.drop(columns=['SalePrice'])
y_train = train_data['SalePrice']

# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

Preparing the Test data set for Prediction

```
X_test =X_test_scaled
```

```
y_test_pred = model.predict(X_test)
```

```
print("Predicted house prices for the test data:")
print(y_test_pred)
```

```
Predicted house prices for the test data:
[  3581.49894512  57894.581041    57157.77183062 ...  41187.92002634
   4818.91918207 114638.14467042]
```

Model performance Evaluation Using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared and Mean Squared Error (MSE) metrics

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Predictions on the training data
y_train_pred = model.predict(X_train)
```

```python
# Calculate MAE and RMSE
mae = mean_absolute_error(y_train, y_train_pred)
rmse = mean_squared_error(y_train, y_train_pred, squared=False)

# Print the results
print(f'Training MAE: {mae}')
print(f'Training RMSE: {rmse}')

# Additional metrics
r_squared = model.score(X_train, y_train)
mse = mean_squared_error(y_train, y_train_pred)

print(f'R-squared on training data: {r_squared}')
print(f'Mean Squared Error on training data: {mse}')
```

```
Training MAE: 22423.206122427106
Training RMSE: 34668.73596670487
R-squared on training data: 0.8094242042104971
Mean Squared Error on training data: 1201921253.5290961
```

Mean and Median of the training dataset

```python
mean_price = train_data['SalePrice'].mean()
median_price = train_data['SalePrice'].median()
print(f"Mean SalePrice: {mean_price}")
print(f"Median SalePrice: {median_price}")
```

```
Mean SalePrice: 180921.19589041095
Median SalePrice: 163000.0
```

```python
# Make predictions on the training data
train_predictions = model.predict(X_train)

# Print first 5 predictions alongside actual prices
for i in range(30):
    print(f'Actual SalePrice: {y_train.iloc[i]}, Predicted SalePrice: 
{train_predictions[i]}')
```

```
Actual SalePrice: 208500, Predicted SalePrice: 212060.88445107004
Actual SalePrice: 181500, Predicted SalePrice: 163409.79247865872
Actual SalePrice: 223500, Predicted SalePrice: 220740.0983045473
Actual SalePrice: 140000, Predicted SalePrice: 189051.0719786906
Actual SalePrice: 250000, Predicted SalePrice: 276036.7465114494
Actual SalePrice: 143000, Predicted SalePrice: 149592.65107443393
Actual SalePrice: 307000, Predicted SalePrice: 282254.5452889392
Actual SalePrice: 200000, Predicted SalePrice: 216356.6533658756
Actual SalePrice: 129900, Predicted SalePrice: 172366.86374207088
Actual SalePrice: 118000, Predicted SalePrice: 106490.10264236729
Actual SalePrice: 129500, Predicted SalePrice: 116711.88445916242
```

```
Actual SalePrice: 345000, Predicted SalePrice: 342315.6425546423
Actual SalePrice: 144000, Predicted SalePrice: 102546.95500961428
Actual SalePrice: 279500, Predicted SalePrice: 240452.01629296137
Actual SalePrice: 157000, Predicted SalePrice: 148967.28469024325
Actual SalePrice: 132000, Predicted SalePrice: 133607.35428353102
Actual SalePrice: 149000, Predicted SalePrice: 135015.75113638488
Actual SalePrice: 90000, Predicted SalePrice: 88617.97272179213
Actual SalePrice: 159000, Predicted SalePrice: 154779.91973603086
Actual SalePrice: 139000, Predicted SalePrice: 132587.38890276017
Actual SalePrice: 325300, Predicted SalePrice: 300014.33701249905
Actual SalePrice: 139400, Predicted SalePrice: 134226.27870761263
Actual SalePrice: 230000, Predicted SalePrice: 269553.77691814094
Actual SalePrice: 129900, Predicted SalePrice: 139821.63973336632
Actual SalePrice: 154000, Predicted SalePrice: 130924.46034890861
Actual SalePrice: 256300, Predicted SalePrice: 264891.46087731625
Actual SalePrice: 134800, Predicted SalePrice: 124779.86412297426
Actual SalePrice: 306000, Predicted SalePrice: 292243.4579140794
Actual SalePrice: 207500, Predicted SalePrice: 169933.99781867318
Actual SalePrice: 68500, Predicted SalePrice: 59136.46397952481
```

Cross- Validation

used to assess the performance of a machine learning model by dividing the dataset into multiple subsets, or "folds," and then training and testing the model multiple times on different combinations of these folds. It helps evaluate how well the model generalizes to unseen data and prevents overfitting.

```python
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, mean_absolute_error,
mean_squared_error
import numpy as np

# Define scorers
mae_scorer = make_scorer(mean_absolute_error)
rmse_scorer = make_scorer(mean_squared_error, squared=False)

# Perform cross-validation
mae_scores = cross_val_score(model, X_train, y_train,
scoring=mae_scorer, cv=5)
rmse_scores = cross_val_score(model, X_train, y_train,
scoring=rmse_scorer, cv=5)

print("Cross-validated MAE: ", np.mean(mae_scores))
print("Cross-validated RMSE: ", np.mean(rmse_scores))

Cross-validated MAE:  22807.962869899988
Cross-validated RMSE:  34994.64321135447
```

Model Performance Summary:

The Linear Regression model was evaluated on both the training set and through cross-validation, yielding the following results:

1. Training MAE (Mean Absolute Error): 22,423.21
2. Training RMSE (Root Mean Squared Error): 34,668.74
3. R-squared on training data: 0.8094, indicating that the model explains approximately 81% of the variance in the target variable (SalePrice).
4. Cross-validated MAE: 22,807.96
5. Cross-validated RMSE: 34,994.64

These performance metrics suggest that the model generalizes well across different subsets of the data, as indicated by the similarity between the training and cross-validated results. The MAE and RMSE values reveal that the model's predictions are, on average, off by about 22,000 in SalePrice, and the magnitude of prediction errors is around 34,000.

Insights:

1. The model explains a significant portion of the variance in SalePrice, with an R-squared score of approximately 81%. However, the error metrics (MAE and RMSE) indicate that the model's predictions are still far from perfect.
2. With a mean SalePrice of 180,921 and a median SalePrice of 163,000, the MAE and RMSE values of 22,000+ and 34,000+ represent a substantial error relative to these typical SalePrice values.
3. The closeness of the training and cross-validated error metrics suggests that the model is not overfitting and generalizes well, but there is still room for improvement to reduce the prediction errors.

Recommendations for Improvement:

1. Feature Engineering: Introduce interaction terms or non-linear features to better capture the relationships in the data.
2. Advanced Models: Consider using more sophisticated models like Random Forests, Gradient Boosting Machines (GBM), or XGBoost, which may capture complex patterns better than linear models.
3. Hyperparameter Tuning:To explore more advanced models, performing hyperparameter tuning could help optimize the model's performance and reduce prediction errors.

4. Overall, the model shows promise with consistent results across training and cross-validation, but additional improvements could further enhance prediction