BZAN 6355 Final Project / Team 11

Introduction

The final sale price of homes in the US depends on many factors, like the area of the house and location, being able to predict the sale price makes your job in the invesment field easier, this task can be done using some of the models we learned through the course using the provided data of other homes with known sale price we can train a model to predict the price of another list of homes.

Objective

Our goal in the project is use the provided datato train a model in order to predict the sale price of 100 properties with unknown sale price.

Process Summary

1. Get a feel of the data provided.
2. Check the data for missing values.
3. Merge the 4 datasets into one dataset to feed into the model.
4. Do EDA on the merged dataset to know more about the data provided and the relations between the variables.
5. Train 2 types of models on the merged data.
6. Evaluate different variations of each model type to choose the one with the best score.
7. Compare the chosen modelof each model type and see which is better.
8. Do the prediction using the chosen modell.

Code

```
#Importing Libraries

import pandas as pd
import numpy as np
from numpy import NaN as NA
import matplotlib.pyplot as plt
import seaborn as sns

# non-interactive plot
%matplotlib inline
```

```
import statsmodels.api as sm
import sklearn.linear_model as lm
import sklearn.neighbors as knn
```

Importing the Data Files.

```
House_Feature = pd.read_csv("House_Feature.csv")
Property_Survey_1 = pd.read_csv("Property_Survey_1.csv")
Property_Survey_2 = pd.read_csv("Property_Survey_2.csv")
Quality_Assessment = pd.read_csv("Quality_Assessment.csv")
```

1. Data Managment

A) Checking the imported data sets.

```
House_Feature.head(5)
```

```
         PID  YearBuilt  YearRemodel  VeneerExterior  BsmtFinTp
BsmtFinSqft  \
0  526301100       1960         1960             112          1
639
1  526350040       1961         1961               0          1
468
2  526351010       1958         1958             108          1
923
3  526353030       1968         1968               0          1
1065
4  527105010       1997         1998               0          1
791

   BsmtUnfinSqft  HeatingQC  FstFlrSqft  SecFlrSqft  ...
FullBathHouse  \
0            441          0        1656           0  ...
1
1            270          1         896           0  ...
1
2            406          1        1329           0  ...
1
3           1045          3        2110           0  ...
2
4            137          2         928         701  ...
2

   BdrmAbvGrnd  RmAbvGrnd  Fireplaces  GarageTp  GarageCars
GarageArea  \
0            3          7           2         3           2
528
```

```
1             2           5             0          3             1
730
2             3           6             0          3             1
312
3             3           8             2          3             2
522
4             3           6             1          3             2
482
```

```
   WdDckSqft  OpenPrchSqft  SalePrice
0        210            62     215000
1        140             0     105000
2        393            36     172000
3          0             0     244000
4        212            34     189900

[5 rows x 23 columns]
```

```
Property_Survey_1.head(5)
```

```
         PID  LotArea  LotShape  BldgTp
0  526301100    31770         0       1
1  526350040    11622         1       1
2  526351010    14267         0       1
3  526353030    11160         1       1
4  527105010    13830         0       1
```

```
Property_Survey_2.head(5)
```

```
         PID  LotArea  LotShape  BldgTp
0  903430060     5520         1       1
1  903451090     6876         1       1
2  903458170     6240         1       1
3  903475100     5775         0       1
4  903476090     5852         0       1
```

```
Quality_Assessment.head(5)
```

```
         PID  OverallQuality  OverallCondition
0  526301100               6                 5
1  526350040               5                 6
2  526351010               6                 6
3  526353030               7                 5
4  527105010               5                 5
```

```
House_Feature.shape

(2370, 23)

Property_Survey_2.shape

(1770, 4)
```

```
Property_Survey_2.shape
```

```
(1770, 4)
```

```
Quality_Assessment.shape
```

```
(2370, 3)
```

B) Chechking for missing values

```
House_Feature.isnull().sum().sum()
```

```
0
```

```
Property_Survey_1.isnull().sum().sum()
```

```
0
```

```
Property_Survey_2.isnull().sum().sum()
```

```
0
```

```
Quality_Assessment.isnull().sum().sum()
```

```
0
```

C) Checking the types of the columns

```
House_Feature.dtypes
```

```
PID               int64
YearBuilt         int64
YearRemodel       int64
VeneerExterior    int64
BsmtFinTp         int64
BsmtFinSqft       int64
BsmtUnfinSqft     int64
HeatingQC         int64
FstFlrSqft        int64
SecFlrSqft        int64
AbvGrndLiving     int64
FullBathBsmt      int64
HalfBathHouse     int64
FullBathHouse     int64
BdrmAbvGrnd       int64
RmAbvGrnd         int64
Fireplaces        int64
GarageTp          int64
GarageCars        int64
GarageArea        int64
WdDckSqft         int64
```

```
OpenPrchSqft      int64
SalePrice         int64
dtype: object

Property_Survey_1.dtypes

PID         int64
LotArea     int64
LotShape    int64
BldgTp      int64
dtype: object

Property_Survey_2.dtypes

PID         int64
LotArea     int64
LotShape    int64
BldgTp      int64
dtype: object

Quality_Assessment.dtypes

PID                 int64
OverallQuality      int64
OverallCondition    int64
dtype: object
```

Merging the Datasets

First we need to combain the rows of Property_Survey_1 and Property_Survey_2 as they have they are two datasets with the same features.

```
Full_Property_Survey = Property_Survey_1.append(Property_Survey_2)
Full_Property_Survey

C:\Users\hania\AppData\Local\Temp\ipykernel_13588\1999146108.py:1:
FutureWarning: The frame.append method is deprecated and will be
removed from pandas in a future version. Use pandas.concat instead.
  Full_Property_Survey = Property_Survey_1.append(Property_Survey_2)

            PID  LotArea  LotShape  BldgTp
0     526301100    31770         0       1
1     526350040    11622         1       1
2     526351010    14267         0       1
3     526353030    11160         1       1
4     527105010    13830         0       1
...         ...      ...       ...     ...
1765  923251180    20000         1       1
1766  923275080     7937         0       1
1767  923276100     8885         0       1
1768  923400125    10441         1       1
```

```
1769  924100070    10010          1       1

[2370 rows x 4 columns]
```

Sanity Checks:

```
len(Full_Property_Survey) == len(Property_Survey_1) +
len(Property_Survey_2)

True

Full_Property_Survey.shape

(2370, 4)
```

We then need to merge the Full_Property_Survey with the House_Feature.

```
df3 = pd.merge(House_Feature, Full_Property_Survey, on = "PID", how =
"inner", validate="1:1")
df3

            PID  YearBuilt  YearRemodel  VeneerExterior  BsmtFinTp  \
0     526301100       1960         1960             112          1
1     526350040       1961         1961               0          1
2     526351010       1958         1958             108          1
3     526353030       1968         1968               0          1
4     527105010       1997         1998               0          1
...         ...        ...          ...             ...        ...
2365  923251180       1960         1996               0          1
2366  923275080       1984         1984               0          1
2367  923276100       1983         1983               0          1
2368  923400125       1992         1992               0          1
2369  924100070       1974         1975               0          1

      BsmtFinSqft  BsmtUnfinSqft  HeatingQC  FstFlrSqft
SecFlrSqft  ...  \
0             639            441          0        1656
0  ...
1             468            270          1         896
0  ...
2             923            406          1        1329
0  ...
3            1065           1045          3        2110
0  ...
4             791            137          2         928
701  ...
...           ...            ...        ...         ...         ...  .
..
2365         1224              0          3        1224
0  ...
```

```
2366          819             184           1        1003
0  ...
2367          301             239           1         902
0  ...
2368          337             575           1         970
0  ...
2369         1071             195           2        1389
0  ...

      Fireplaces  GarageTp  GarageCars  GarageArea  WdDckSqft
OpenPrchSqft  \
0               2         3           2         528        210
62
1               0         3           1         730        140
0
2               0         3           1         312        393
36
3               2         3           2         522          0
0
4               1         3           2         482        212
34
...           ...       ...         ...         ...        ...
...
2365            1         1           2         576        474
0
2366            0         1           2         588        120
0
2367            0         3           2         484        164
0
2368            0         0           0           0         80
32
2369            1         3           2         418        240
38

      SalePrice  LotArea  LotShape  BldgTp
0        215000    31770         0       1
1        105000    11622         1       1
2        172000    14267         0       1
3        244000    11160         1       1
4        189900    13830         0       1
...         ...      ...       ...     ...
2365     131000    20000         1       1
2366     142500     7937         0       1
2367     131000     8885         0       1
2368     132000    10441         1       1
2369     170000    10010         1       1

[2370 rows x 26 columns]
```

Sanity Checks:

```
df3.shape[1] == House_Feature.shape[1] + Full_Property_Survey.shape[1]
- 1 # 1 minus for the PID column that wasn't repeated.
```

```
True
```

We then need to merge the the previous dataframe df3 with the Quality_Assessment.

```
FullData = pd.merge(df3, Quality_Assessment, on = "PID", how = "inner"
,validate="1:1")
FullData
```

```
            PID  YearBuilt  YearRemodel  VeneerExterior  BsmtFinTp  \
0     526301100       1960         1960             112          1
1     526350040       1961         1961               0          1
2     526351010       1958         1958             108          1
3     526353030       1968         1968               0          1
4     527105010       1997         1998               0          1
...         ...        ...          ...             ...        ...
2365  923251180       1960         1996               0          1
2366  923275080       1984         1984               0          1
2367  923276100       1983         1983               0          1
2368  923400125       1992         1992               0          1
2369  924100070       1974         1975               0          1

      BsmtFinSqft  BsmtUnfinSqft  HeatingQC  FstFlrSqft
SecFlrSqft  ...  \
0             639            441          0        1656
0  ...
1             468            270          1         896
0  ...
2             923            406          1        1329
0  ...
3            1065           1045          3        2110
0  ...
4             791            137          2         928
701  ...
...           ...            ...        ...         ...          ...  .
..
2365         1224              0          3        1224
0  ...
2366          819            184          1        1003
0  ...
2367          301            239          1         902
0  ...
2368          337            575          1         970
0  ...
2369         1071            195          2        1389
0  ...

      GarageCars  GarageArea  WdDckSqft  OpenPrchSqft  SalePrice
```

```
LotArea  \
0              2          528         210              62     215000
31770
1              1          730         140               0     105000
11622
2              1          312         393              36     172000
14267
3              2          522           0               0     244000
11160
4              2          482         212              34     189900
13830
...          ...          ...         ...             ...        ...
...
2365           2          576         474               0     131000
20000
2366           2          588         120               0     142500
7937
2367           2          484         164               0     131000
8885
2368           0            0          80              32     132000
10441
2369           2          418         240              38     170000
10010

      LotShape  BldgTp  OverallQuality  OverallCondition
0            0       1               6                 5
1            1       1               5                 6
2            0       1               6                 6
3            1       1               7                 5
4            0       1               5                 5
...        ...     ...             ...               ...
2365         1       1               5                 7
2366         0       1               6                 6
2367         0       1               5                 5
2368         1       1               5                 5
2369         1       1               5                 5

[2370 rows x 28 columns]
```

Sanity Checks:

```
FullData.shape[1] == df3.shape[1] + Quality_Assessment.shape[1] - 1 #
1 minus for the PID column that wasn't repeated.

True
```

Checking For Missing Values that maybe introduced while combaining the datasets.

```
FullData.isnull().sum().sum()
```

```
0
```

2. Exploratory Data Analysis

```
# Check the structure and size of the dataset
print("Dataset shape:", FullData.shape)
print("Dataset information:")
print(FullData.info())
```

```
Dataset shape: (2370, 28)
Dataset information:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2370 entries, 0 to 2369
Data columns (total 28 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   PID               2370 non-null   int64
 1   YearBuilt         2370 non-null   int64
 2   YearRemodel       2370 non-null   int64
 3   VeneerExterior    2370 non-null   int64
 4   BsmtFinTp         2370 non-null   int64
 5   BsmtFinSqft       2370 non-null   int64
 6   BsmtUnfinSqft     2370 non-null   int64
 7   HeatingQC         2370 non-null   int64
 8   FstFlrSqft        2370 non-null   int64
 9   SecFlrSqft        2370 non-null   int64
 10  AbvGrndLiving     2370 non-null   int64
 11  FullBathBsmt      2370 non-null   int64
 12  HalfBathHouse     2370 non-null   int64
 13  FullBathHouse     2370 non-null   int64
 14  BdrmAbvGrnd       2370 non-null   int64
 15  RmAbvGrnd         2370 non-null   int64
 16  Fireplaces        2370 non-null   int64
 17  GarageTp          2370 non-null   int64
 18  GarageCars        2370 non-null   int64
 19  GarageArea        2370 non-null   int64
 20  WdDckSqft         2370 non-null   int64
 21  OpenPrchSqft      2370 non-null   int64
 22  SalePrice         2370 non-null   int64
 23  LotArea           2370 non-null   int64
 24  LotShape          2370 non-null   int64
 25  BldgTp            2370 non-null   int64
 26  OverallQuality    2370 non-null   int64
 27  OverallCondition  2370 non-null   int64
dtypes: int64(28)
memory usage: 537.0 KB
None
```

```python
# Check for missing values
print("Number of missing values in each column:")
print(FullData.isna().sum())
```

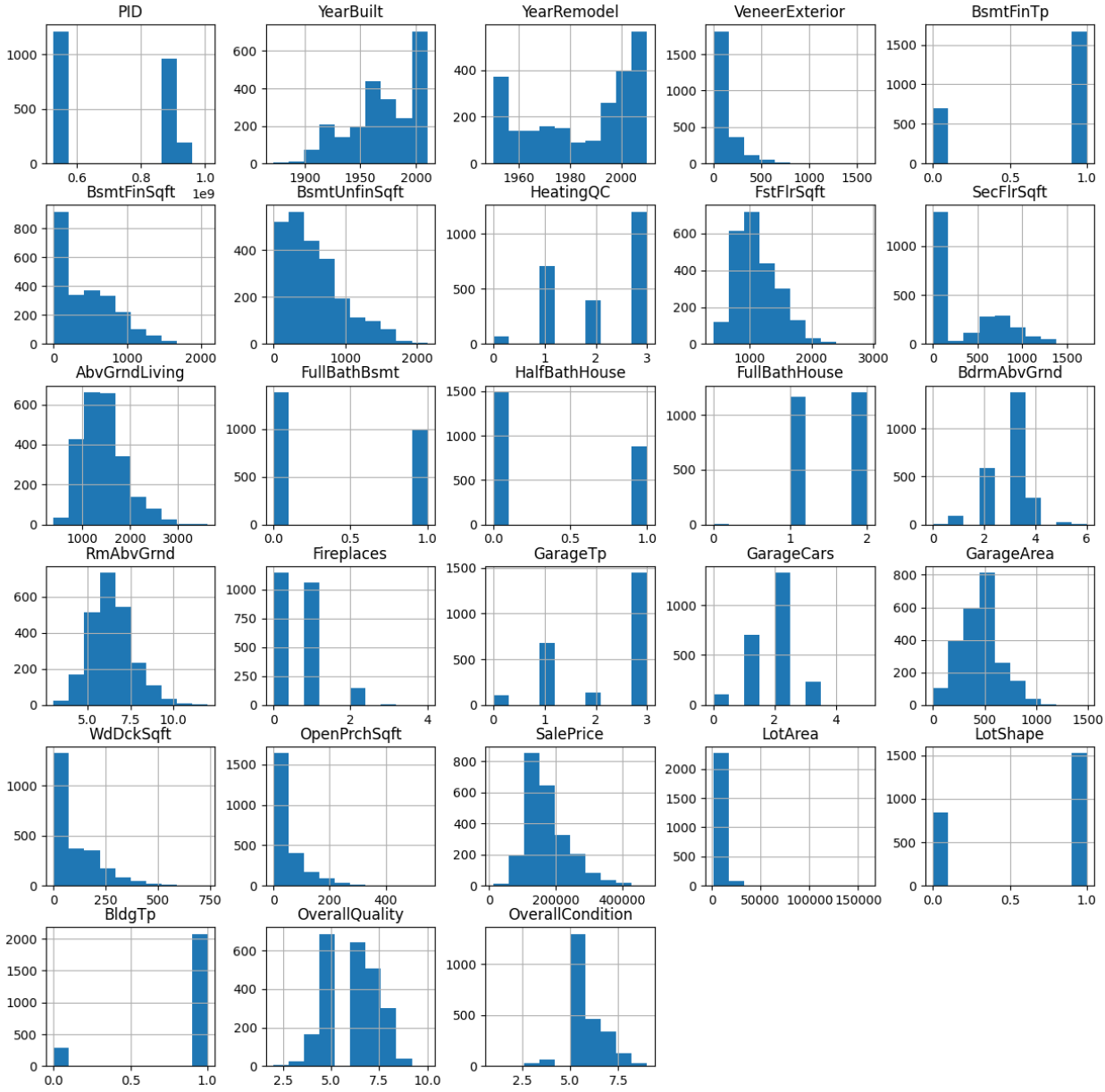
```
Number of missing values in each column:
PID                0
YearBuilt          0
YearRemodel        0
VeneerExterior     0
BsmtFinTp          0
BsmtFinSqft        0
BsmtUnfinSqft      0
HeatingQC          0
FstFlrSqft         0
SecFlrSqft         0
AbvGrndLiving      0
FullBathBsmt       0
HalfBathHouse      0
FullBathHouse      0
BdrmAbvGrnd        0
RmAbvGrnd          0
Fireplaces         0
GarageTp           0
GarageCars         0
GarageArea         0
WdDckSqft          0
OpenPrchSqft       0
SalePrice          0
LotArea            0
LotShape           0
BldgTp             0
OverallQuality     0
OverallCondition   0
dtype: int64
```

```python
# Check the distribution and summary statistics of the variables
print("Summary statistics of numerical variables:")
print(FullData.describe())
FullData.hist(figsize=(15,15))
```

```
Summary statistics of numerical variables:
                PID    YearBuilt  YearRemodel  VeneerExterior
BsmtFinTp  \
count  2.370000e+03  2370.000000  2370.000000     2370.000000
2370.000000
mean   7.153307e+08  1970.570886  1984.086920       88.143882
0.704641
std    1.886401e+08    30.109415    20.694221      158.718586
0.456300
min    5.263011e+08  1872.000000  1950.000000        0.000000
```

```
0.000000
25%    5.313690e+08  1953.000000  1965.000000        0.000000
0.000000
50%    5.354551e+08  1972.000000  1993.000000        0.000000
1.000000
75%    9.071351e+08  1999.000000  2003.000000      144.000000
1.000000
max    1.007100e+09  2010.000000  2010.000000     1600.000000
1.000000

       BsmtFinSqft  BsmtUnfinSqft    HeatingQC   FstFlrSqft
SecFlrSqft  ...  \
count  2370.000000    2370.000000  2370.000000  2370.000000
2370.000000  ...
mean    429.805907     557.101688     2.153586  1116.442616
325.198734  ...
std     408.779757     410.685375     0.944222   344.282409
406.198426  ...
min       0.000000       0.000000     0.000000   407.000000
0.000000  ...
25%       0.000000     239.250000     1.000000   864.000000
0.000000  ...
50%     378.000000     470.000000     3.000000  1056.000000
0.000000  ...
75%     715.500000     792.000000     3.000000  1331.500000
688.750000  ...
max    2085.000000    2140.000000     3.000000  2898.000000
1721.000000  ...

       GarageCars   GarageArea    WdDckSqft  OpenPrchSqft
SalePrice  \
count  2370.000000  2370.000000  2370.000000   2370.000000
2370.000000
mean      1.718987   457.978903    91.650633     46.002532
173730.772574
std       0.708998   197.608559   120.616635     64.245617
64080.843305
min       0.000000     0.000000     0.000000      0.000000
12789.000000
25%       1.000000   312.000000     0.000000      0.000000
129900.000000
50%       2.000000   463.000000     0.000000     25.000000
159000.000000
75%       2.000000   572.000000   168.000000     68.000000
207000.000000
max       5.000000  1488.000000   736.000000    547.000000
475000.000000

             LotArea     LotShape       BldgTp  OverallQuality  \
count    2370.000000  2370.000000  2370.000000     2370.000000
```

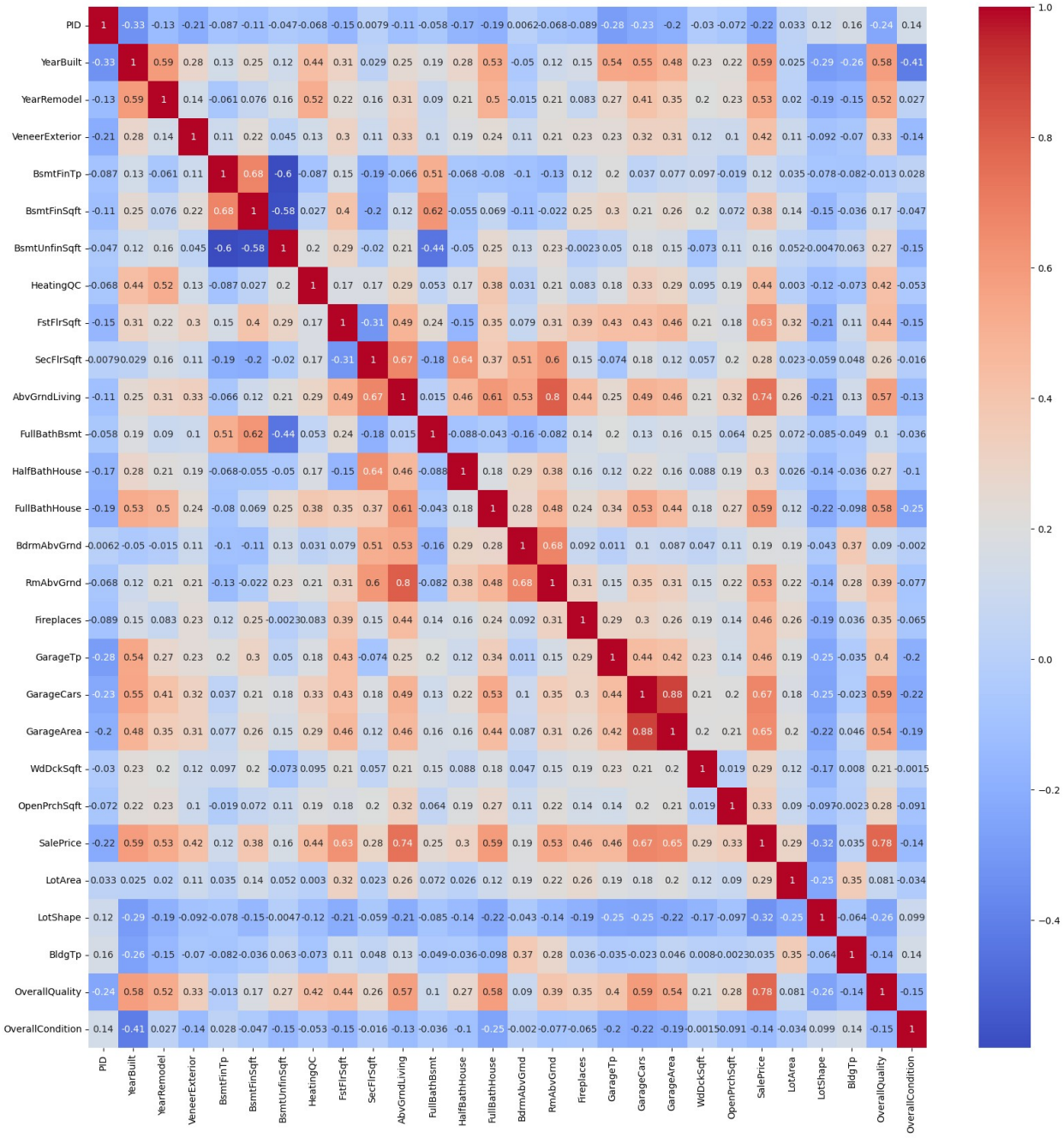
```
mean     9700.865401     0.643460     0.877215        6.050633
std      6153.729681     0.479078     0.328259        1.252423
min      1300.000000     0.000000     0.000000        2.000000
25%      7200.000000     0.000000     1.000000        5.000000
50%      9316.000000     1.000000     1.000000        6.000000
75%     11235.000000     1.000000     1.000000        7.000000
max    159000.000000     1.000000     1.000000       10.000000

       OverallCondition
count       2370.000000
mean           5.630380
std            1.095717
min            1.000000
25%            5.000000
50%            5.000000
75%            6.000000
max            9.000000

[8 rows x 28 columns]

array([[<Axes: title={'center': 'PID'}>,
        <Axes: title={'center': 'YearBuilt'}>,
        <Axes: title={'center': 'YearRemodel'}>,
        <Axes: title={'center': 'VeneerExterior'}>,
        <Axes: title={'center': 'BsmtFinTp'}>],
       [<Axes: title={'center': 'BsmtFinSqft'}>,
        <Axes: title={'center': 'BsmtUnfinSqft'}>,
        <Axes: title={'center': 'HeatingQC'}>,
        <Axes: title={'center': 'FstFlrSqft'}>,
        <Axes: title={'center': 'SecFlrSqft'}>],
       [<Axes: title={'center': 'AbvGrndLiving'}>,
        <Axes: title={'center': 'FullBathBsmt'}>,
        <Axes: title={'center': 'HalfBathHouse'}>,
        <Axes: title={'center': 'FullBathHouse'}>,
        <Axes: title={'center': 'BdrmAbvGrnd'}>],
       [<Axes: title={'center': 'RmAbvGrnd'}>,
        <Axes: title={'center': 'Fireplaces'}>,
        <Axes: title={'center': 'GarageTp'}>,
        <Axes: title={'center': 'GarageCars'}>,
        <Axes: title={'center': 'GarageArea'}>],
       [<Axes: title={'center': 'WdDckSqft'}>,
        <Axes: title={'center': 'OpenPrchSqft'}>,
        <Axes: title={'center': 'SalePrice'}>,
        <Axes: title={'center': 'LotArea'}>,
        <Axes: title={'center': 'LotShape'}>],
       [<Axes: title={'center': 'BldgTp'}>,
        <Axes: title={'center': 'OverallQuality'}>,
        <Axes: title={'center': 'OverallCondition'}>, <Axes: >, <Axes:
>]],
      dtype=object)
```



We found highly skewed distributions of some varibles: VeneerExterior, BsmtFinSqft, BsmtUnfinSqft, FstFlrSqft, SecFlrSqft, WdDckSqft and OpenPrchSqft

```
# Making column list of the highly skewed variables:
Skewed_Var = ['VeneerExterior', 'BsmtFinSqft', 'BsmtUnfinSqft',
'FstFlrSqft', 'SecFlrSqft', 'WdDckSqft','OpenPrchSqft']

# Create a heatmap to visualize correlations
plt.figure(figsize=(20, 20))
sns.heatmap(FullData.corr(), annot=True, cmap='coolwarm')
```

```
<Axes: >
```



From the heatmap above we don't find any linearly correlated variables, there are some stong corelations but the variables with these strong relations should not be excluded from the model

3. Modeling

Data Standariztaion

```
# Defining the traget variable DataFrame:
Y = pd.DataFrame(FullData["SalePrice"])
Y
```

```
      SalePrice
0        215000
1        105000
2        172000
3        244000
4        189900
...         ...
2365     131000
2366     142500
2367     131000
2368     132000
2369     170000

[2370 rows x 1 columns]
```

```
# Defining the Input variables DataFrame:
X = FullData.drop("SalePrice", axis = 1)
X
```

```
            PID  YearBuilt  YearRemodel  VeneerExterior  BsmtFinTp  \
0     526301100       1960         1960             112          1
1     526350040       1961         1961               0          1
2     526351010       1958         1958             108          1
3     526353030       1968         1968               0          1
4     527105010       1997         1998               0          1
...         ...        ...          ...             ...        ...
2365  923251180       1960         1996               0          1
2366  923275080       1984         1984               0          1
2367  923276100       1983         1983               0          1
2368  923400125       1992         1992               0          1
2369  924100070       1974         1975               0          1

      BsmtFinSqft  BsmtUnfinSqft  HeatingQC  FstFlrSqft
SecFlrSqft  ...  \
0             639            441          0        1656
0  ...
1             468            270          1         896
0  ...
2             923            406          1        1329
0  ...
3            1065           1045          3        2110
0  ...
```

```
4              791             137           2          928
701  ...
...            ...             ...         ...          ...          ...  .
..
2365          1224               0           3         1224
0  ...
2366           819             184           1         1003
0  ...
2367           301             239           1          902
0  ...
2368           337             575           1          970
0  ...
2369          1071             195           2         1389
0  ...

      GarageTp  GarageCars  GarageArea  WdDckSqft  OpenPrchSqft
LotArea  \
0            3           2         528        210            62
31770
1            3           1         730        140             0
11622
2            3           1         312        393            36
14267
3            3           2         522          0             0
11160
4            3           2         482        212            34
13830
...        ...         ...         ...        ...           ...       .
..
2365         1           2         576        474             0
20000
2366         1           2         588        120             0
7937
2367         3           2         484        164             0
8885
2368         0           0           0         80            32
10441
2369         3           2         418        240            38
10010

      LotShape  BldgTp  OverallQuality  OverallCondition
0            0       1               6                 5
1            1       1               5                 6
2            0       1               6                 6
3            1       1               7                 5
4            0       1               5                 5
...        ...     ...             ...               ...
2365         1       1               5                 7
2366         0       1               6                 6
```

```
2367         0       1                5                 5
2368         1       1                5                 5
2369         1       1                5                 5

[2370 rows x 27 columns]
```

```
from sklearn import preprocessing

#Standarizing Y:

yScaler = preprocessing.StandardScaler()
Y_final = pd.DataFrame(yScaler.fit_transform(Y),
columns=['SalePrice'])
Y_final.head(3)
```

```
   SalePrice
0   0.644154
1  -1.072790
2  -0.027015
```

```
Y_final.describe().round(4)
```

```
       SalePrice
count  2370.0000
mean     -0.0000
std       1.0002
min      -2.5121
25%      -0.6841
50%      -0.2299
75%       0.5193
max       4.7024
```

We can verfiy the standrization as the mean is so close to zero and the SD is around 1

```
X.isnull().sum().sum()

0
```

Since X has no missing data we can proceed with the standarization process:

```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline, make_pipeline

X.columns

Index(['PID', 'YearBuilt', 'YearRemodel', 'VeneerExterior',
'BsmtFinTp',
       'BsmtFinSqft', 'BsmtUnfinSqft', 'HeatingQC', 'FstFlrSqft',
'SecFlrSqft',
       'AbvGrndLiving', 'FullBathBsmt', 'HalfBathHouse',
'FullBathHouse',
```

```
       'BdrmAbvGrnd', 'RmAbvGrnd', 'Fireplaces', 'GarageTp',
'GarageCars',
       'GarageArea', 'WdDckSqft', 'OpenPrchSqft', 'LotArea',
'LotShape',
       'BldgTp', 'OverallQuality', 'OverallCondition'],
      dtype='object')

Skewed_Var

['VeneerExterior',
 'BsmtFinSqft',
 'BsmtUnfinSqft',
 'FstFlrSqft',
 'SecFlrSqft',
 'WdDckSqft',
 'OpenPrchSqft']

nominalCols = ['YearBuilt', 'YearRemodel',  'BsmtFinTp', 'HeatingQC',
       'AbvGrndLiving', 'FullBathBsmt', 'HalfBathHouse',
'FullBathHouse',
       'BdrmAbvGrnd', 'RmAbvGrnd', 'Fireplaces', 'GarageTp',
'GarageCars',
       'GarageArea', 'LotArea', 'LotShape',
       'BldgTp', 'OverallQuality', 'OverallCondition']

X_nominal = X[nominalCols].copy()
X_skewed  = X[Skewed_Var].copy()



# Defining a pipeline 1 for Imputation and Standarization:
Pipe1_transformer = make_pipeline(SimpleImputer(strategy =
'most_frequent'), preprocessing.StandardScaler())

# Defining a pipeline 2 for Imputation, Log transformation and
Standarization:
Pipe2_transformer = make_pipeline(SimpleImputer(strategy = 'median'),

preprocessing.FunctionTransformer(np.log1p, validate=False),
                                        preprocessing.StandardScaler())


# Adding suffix "_z" to the columns names of X:
Xlabels = X_nominal.add_suffix('_z').columns.tolist()
XSkewlabels = X_skewed.add_suffix('_z').columns.tolist()

from sklearn.compose import ColumnTransformer, make_column_transformer
# combineing the 2  Pipelines into a ColumnTransformer
preprocessor = make_column_transformer((Pipe1_transformer,
X_nominal.columns),
                                        (Pipe2_transformer,
```

```
X_skewed.columns))

# prepare all labels
labelsPreprocessor = [label for labels in [ Xlabels, XSkewlabels] for
label in labels]
print(labelsPreprocessor)
```

```
['YearBuilt_z', 'YearRemodel_z', 'BsmtFinTp_z', 'HeatingQC_z',
'AbvGrndLiving_z', 'FullBathBsmt_z', 'HalfBathHouse_z',
'FullBathHouse_z', 'BdrmAbvGrnd_z', 'RmAbvGrnd_z', 'Fireplaces_z',
'GarageTp_z', 'GarageCars_z', 'GarageArea_z', 'LotArea_z',
'LotShape_z', 'BldgTp_z', 'OverallQuality_z', 'OverallCondition_z',
'VeneerExterior_z', 'BsmtFinSqft_z', 'BsmtUnfinSqft_z',
'FstFlrSqft_z', 'SecFlrSqft_z', 'WdDckSqft_z', 'OpenPrchSqft_z']
```

```
outputPreprocessor = preprocessor.fit_transform(X)
dfPreprocessed = pd.DataFrame(outputPreprocessor,
columns=labelsPreprocessor, index=X.index)
display(dfPreprocessed.shape, dfPreprocessed.head(),
dfPreprocessed.tail())
```

```
(2370, 26)
```

```
   YearBuilt_z  YearRemodel_z  BsmtFinTp_z  HeatingQC_z
AbvGrndLiving_z  \
0    -0.351157      -1.164190     0.647427    -2.281286
0.470986
1    -0.317937      -1.115857     0.647427    -1.221990        -
1.234949
2    -0.417595      -1.260856     0.647427    -1.221990        -
0.263015
3    -0.085403      -0.777527     0.647427     0.896603
1.490057
4     0.877954       0.672459     0.647427    -0.162694
0.410380

   FullBathBsmt_z  HalfBathHouse_z  FullBathHouse_z  BdrmAbvGrnd_z  \
0        1.184758        -0.769897        -1.012751       0.247145
1       -0.844054        -0.769897        -1.012751      -1.099367
2       -0.844054         1.298874        -1.012751       0.247145
3        1.184758         1.298874         0.980787       0.247145
4       -0.844054         1.298874         0.980787       0.247145

   RmAbvGrnd_z  ...  BldgTp_z  OverallQuality_z  OverallCondition_z  \
0     0.532453  ...  0.374127         -0.040437           -0.575434
1    -0.903987  ...  0.374127         -0.839058            0.337403
2    -0.185767  ...  0.374127         -0.040437            0.337403
3     1.250673  ...  0.374127          0.758185           -0.575434
4    -0.185767  ...  0.374127         -0.839058           -0.575434

   VeneerExterior_z  BsmtFinSqft_z  BsmtUnfinSqft_z  FstFlrSqft_z  \
```

```
0          1.086051       0.731625         0.196125      1.446934
1         -0.762629       0.624542        -0.112734     -0.571243
2          1.071958       0.858128         0.144039      0.724024
3         -0.762629       0.907372         0.739997      2.243253
4         -0.762629       0.805028        -0.538822     -0.455972

   SecFlrSqft_z  WdDckSqft_z  OpenPrchSqft_z
0     -0.869290     1.126685        0.857098
1     -0.869290     0.970747       -1.073082
2     -0.869290     1.368268        0.609151
3     -0.869290    -0.943665       -1.073082
4      1.139904     1.130334        0.583263

[5 rows x 26 columns]

      YearBuilt_z  YearRemodel_z  BsmtFinTp_z  HeatingQC_z
AbvGrndLiving_z  \
2365    -0.351157       0.575793     0.647427     0.896603        -
0.498704
2366     0.446105      -0.004201     0.647427    -1.221990        -
0.994772
2367     0.412885      -0.052534     0.647427    -1.221990        -
1.221481
2368     0.711858       0.382462     0.647427    -1.221990        -
1.068845
2369     0.113912      -0.439197     0.647427    -0.162694        -
0.128336

      FullBathBsmt_z  HalfBathHouse_z  FullBathHouse_z  BdrmAbvGrnd_z
\
2365        1.184758        -0.769897        -1.012751       1.593656

2366        1.184758        -0.769897        -1.012751       0.247145

2367        1.184758        -0.769897        -1.012751      -1.099367

2368       -0.844054        -0.769897        -1.012751       0.247145

2369        1.184758        -0.769897        -1.012751      -1.099367


      RmAbvGrnd_z  ...  BldgTp_z  OverallQuality_z  OverallCondition_z
\
2365     0.532453  ...  0.374127         -0.839058            1.250240

2366    -0.185767  ...  0.374127         -0.040437            0.337403

2367    -0.903987  ...  0.374127         -0.839058           -0.575434

2368    -0.185767  ...  0.374127         -0.839058           -0.575434
```

```
2369    -0.185767  ...  0.374127         -0.839058           -0.575434


      VeneerExterior_z  BsmtFinSqft_z  BsmtUnfinSqft_z
FstFlrSqft_z  \
2365         -0.762629       0.955262        -3.649728      0.453585

2366         -0.762629       0.816996        -0.353767     -0.200658

2367         -0.762629       0.472916        -0.189432     -0.549320

2368         -0.762629       0.511709         0.363310     -0.310562

2369         -0.762629       0.909305        -0.317300      0.869129


      SecFlrSqft_z  WdDckSqft_z  OpenPrchSqft_z
2365      -0.86929     1.440594       -1.073082
2366      -0.86929     0.911572       -1.073082
2367      -0.86929     1.031554       -1.073082
2368      -0.86929     0.756314        0.555851
2369      -0.86929     1.178112        0.633677

[5 rows x 26 columns]
```

```
dfPreprocessed.describe().round(4)
```

```
       YearBuilt_z  YearRemodel_z  BsmtFinTp_z  HeatingQC_z
AbvGrndLiving_z  \
count    2370.0000      2370.0000    2370.0000    2370.0000
2370.0000
mean       -0.0000        -0.0000      -0.0000      -0.0000
0.0000
std         1.0002         1.0002       1.0002       1.0002
1.0002
min        -3.2744        -1.6475      -1.5446      -2.2813
-2.3326
25%        -0.5837        -0.9225      -1.5446      -1.2220
-0.7647
50%         0.0475         0.4308       0.6474       0.8966
-0.1104
75%         0.9444         0.9141       0.6474       0.8966
0.5445
max         1.3098         1.2525       0.6474       0.8966
4.8525

       FullBathBsmt_z  HalfBathHouse_z  FullBathHouse_z  BdrmAbvGrnd_z
\
count       2370.0000        2370.0000        2370.0000      2370.0000

mean          -0.0000           0.0000           0.0000         0.0000
```

```
std             1.0002           1.0002           1.0002          1.0002

min            -0.8441          -0.7699          -3.0063         -3.7924

25%            -0.8441          -0.7699          -1.0128         -1.0994

50%            -0.8441          -0.7699           0.9808          0.2471

75%             1.1848           1.2989           0.9808          0.2471

max             1.1848           1.2989           0.9808          4.2867


       RmAbvGrnd_z  ...   BldgTp_z  OverallQuality_z
OverallCondition_z  \
count    2370.0000  ...  2370.0000         2370.0000
2370.0000
mean       -0.0000  ...    -0.0000           -0.0000
0.0000
std         1.0002  ...     1.0002            1.0002
1.0002
min        -2.3404  ...    -2.6729           -3.2349              -
4.2268
25%        -0.9040  ...     0.3741           -0.8391              -
0.5754
50%        -0.1858  ...     0.3741           -0.0404              -
0.5754
75%         0.5325  ...     0.3741            0.7582
0.3374
max         4.1236  ...     0.3741            3.1540
3.0759

       VeneerExterior_z  BsmtFinSqft_z  BsmtUnfinSqft_z  FstFlrSqft_z
\
count         2370.0000      2370.0000        2370.0000     2370.0000

mean            -0.0000         0.0000          -0.0000        0.0000

std              1.0002         1.0002           1.0002        1.0002

min             -0.7626        -1.4941          -3.6497       -3.1619

25%             -0.7626        -1.4941          -0.1888       -0.6907

50%             -0.7626         0.5511           0.2362       -0.0315

75%              1.1836         0.7705           0.5652        0.7302

max              2.1227         1.1386           1.1922        3.2864
```

```
       SecFlrSqft_z  WdDckSqft_z  OpenPrchSqft_z
count     2370.0000    2370.0000       2370.0000
mean        -0.0000      -0.0000         -0.0000
std          1.0002       1.0002          1.0002
min         -0.8693      -0.9437         -1.0731
25%         -0.8693      -0.9437         -1.0731
50%         -0.8693      -0.9437          0.4448
75%          1.1345       1.0408          0.8995
max          1.4150       1.6105          1.8648

[8 rows x 26 columns]
```

We can verfiy the standrization as the means of all the variables are so close to zero and the SDs are around 1

```
X_z = dfPreprocessed
X_z
```

```
      YearBuilt_z  YearRemodel_z  BsmtFinTp_z  HeatingQC_z
AbvGrndLiving_z  \
0       -0.351157      -1.164190     0.647427    -2.281286
0.470986
1       -0.317937      -1.115857     0.647427    -1.221990        -
1.234949
2       -0.417595      -1.260856     0.647427    -1.221990        -
0.263015
3       -0.085403      -0.777527     0.647427     0.896603
1.490057
4        0.877954       0.672459     0.647427    -0.162694
0.410380
...           ...            ...          ...          ...
...
2365    -0.351157       0.575793     0.647427     0.896603        -
0.498704
2366     0.446105      -0.004201     0.647427    -1.221990        -
0.994772
2367     0.412885      -0.052534     0.647427    -1.221990        -
1.221481
2368     0.711858       0.382462     0.647427    -1.221990        -
1.068845
2369     0.113912      -0.439197     0.647427    -0.162694        -
0.128336

      FullBathBsmt_z  HalfBathHouse_z  FullBathHouse_z  BdrmAbvGrnd_z
\
0           1.184758        -0.769897        -1.012751       0.247145

1          -0.844054        -0.769897        -1.012751      -1.099367
```

```
2          -0.844054          1.298874        -1.012751        0.247145
3           1.184758          1.298874         0.980787        0.247145
4          -0.844054          1.298874         0.980787        0.247145
...              ...               ...              ...             ...
2365        1.184758         -0.769897        -1.012751        1.593656
2366        1.184758         -0.769897        -1.012751        0.247145
2367        1.184758         -0.769897        -1.012751       -1.099367
2368       -0.844054         -0.769897        -1.012751        0.247145
2369        1.184758         -0.769897        -1.012751       -1.099367

      RmAbvGrnd_z  ...  BldgTp_z  OverallQuality_z  OverallCondition_z
\
0        0.532453  ...  0.374127         -0.040437           -0.575434
1       -0.903987  ...  0.374127         -0.839058            0.337403
2       -0.185767  ...  0.374127         -0.040437            0.337403
3        1.250673  ...  0.374127          0.758185           -0.575434
4       -0.185767  ...  0.374127         -0.839058           -0.575434
...           ...  ...       ...               ...                 ...
2365     0.532453  ...  0.374127         -0.839058            1.250240
2366    -0.185767  ...  0.374127         -0.040437            0.337403
2367    -0.903987  ...  0.374127         -0.839058           -0.575434
2368    -0.185767  ...  0.374127         -0.839058           -0.575434
2369    -0.185767  ...  0.374127         -0.839058           -0.575434

      VeneerExterior_z  BsmtFinSqft_z  BsmtUnfinSqft_z
FstFlrSqft_z  \
0             1.086051       0.731625         0.196125      1.446934
1            -0.762629       0.624542        -0.112734     -0.571243
2             1.071958       0.858128         0.144039      0.724024
```

```
3            -0.762629       0.907372         0.739997      2.243253

4            -0.762629       0.805028        -0.538822     -0.455972

...                ...            ...              ...           ...

2365         -0.762629       0.955262        -3.649728      0.453585

2366         -0.762629       0.816996        -0.353767     -0.200658

2367         -0.762629       0.472916        -0.189432     -0.549320

2368         -0.762629       0.511709         0.363310     -0.310562

2369         -0.762629       0.909305        -0.317300      0.869129


      SecFlrSqft_z  WdDckSqft_z  OpenPrchSqft_z
0        -0.869290     1.126685        0.857098
1        -0.869290     0.970747       -1.073082
2        -0.869290     1.368268        0.609151
3        -0.869290    -0.943665       -1.073082
4         1.139904     1.130334        0.583263
...            ...          ...             ...
2365     -0.869290     1.440594       -1.073082
2366     -0.869290     0.911572       -1.073082
2367     -0.869290     1.031554       -1.073082
2368     -0.869290     0.756314        0.555851
2369     -0.869290     1.178112        0.633677

[2370 rows x 26 columns]
```

Backward Selection Regression

```
# Importing the recursive feature elimination "REF"
from sklearn.feature_selection import RFE

# first we define a linear regression instance
regRFE = lm.LinearRegression()

df_retained_bool = pd.DataFrame()
for i in range(1, X_z.shape[1]+2, 2):
    rfe_temp = RFE(regRFE, n_features_to_select = i)
    rfe_temp.fit(X = X_z, y = Y_final['SalePrice'])
    df_retained_bool['retained_' + str(i)] = rfe_temp.get_support()

display(df_retained_bool.shape, df_retained_bool.sum())

(26, 14)
```

```
retained_1      1
retained_3      3
retained_5      5
retained_7      7
retained_9      9
retained_11    11
retained_13    13
retained_15    15
retained_17    17
retained_19    19
retained_21    21
retained_23    23
retained_25    25
retained_27    26
dtype: int64
```

Model 1 (Linear Regression)

```
# Importing cross_val_score
from sklearn.model_selection import cross_val_score

# Calculating the x-valid scores for each iteration number:
regression = lm.LinearRegression()
df_xvalid_reg = pd.DataFrame()
for label, content in df_retained_bool.iteritems():
    selected_labels_list = X_z.columns[content.to_numpy()]
    df_xvalid_reg[label] = cross_val_score(regression, X =
X_z[selected_labels_list], y = Y_final['SalePrice'], cv = 10, scoring
= "neg_mean_absolute_error" )

df_xvalid_reg.shape
```

```
C:\Users\hania\AppData\Local\Temp\ipykernel_13588\689719760.py:7:
FutureWarning: iteritems is deprecated and will be removed in a future
version. Use .items instead.
  for label, content in df_retained_bool.iteritems():
```

```
(10, 14)
```

```
df_xvalid_reg
```

```
   retained_1  retained_3  retained_5  retained_7  retained_9
retained_11  \
0   -0.468445   -0.352066   -0.317260   -0.282802   -0.265271    -
0.267776
1   -0.536890   -0.411896   -0.354504   -0.313104   -0.298938    -
0.291876
2   -0.403338   -0.346880   -0.328597   -0.276148   -0.258841    -
0.251069
3   -0.503120   -0.363249   -0.325128   -0.296316   -0.284192    -
```

```
0.274967
4   -0.454336   -0.356028   -0.313313   -0.256347   -0.232601    -
0.224164
5   -0.525578   -0.379635   -0.354059   -0.305878   -0.299038    -
0.293134
6   -0.383958   -0.312849   -0.304160   -0.278600   -0.259495    -
0.253855
7   -0.465435   -0.344933   -0.313474   -0.303463   -0.295233    -
0.281586
8   -0.450539   -0.314466   -0.292733   -0.258590   -0.249641    -
0.251898
9   -0.476820   -0.382973   -0.331958   -0.306620   -0.289389    -
0.284709
```

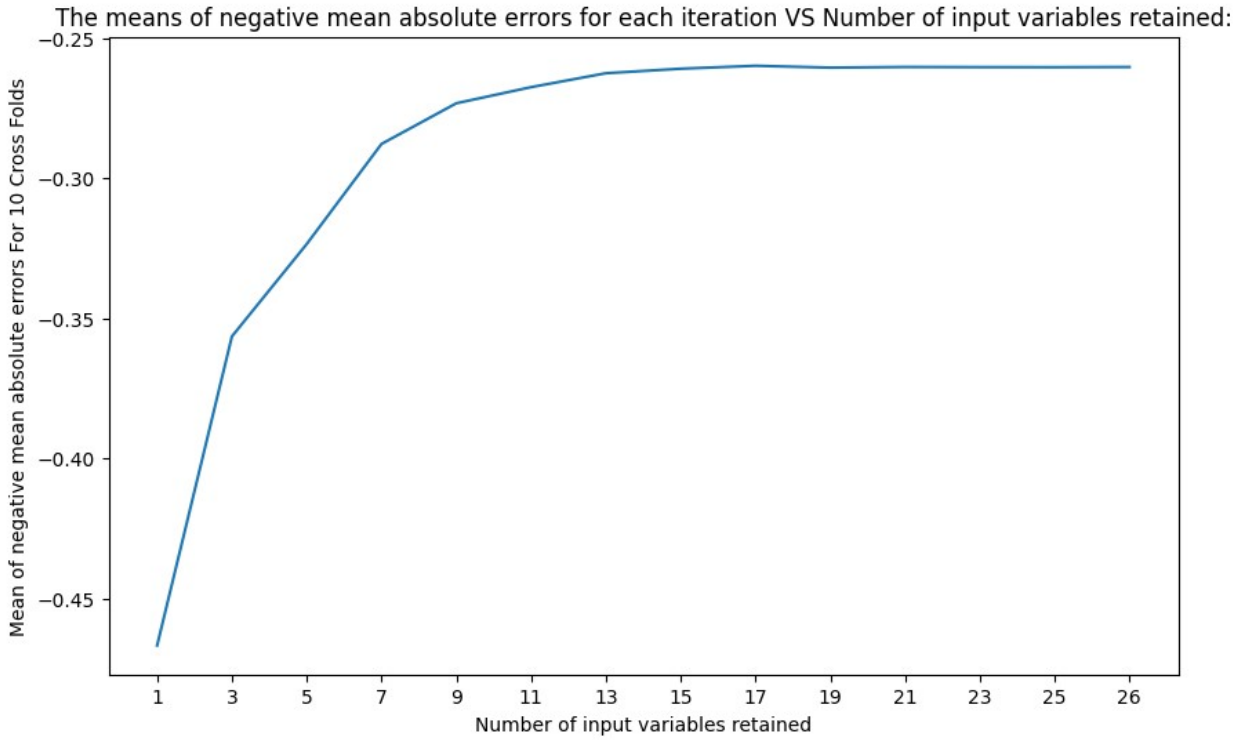
	retained_13	retained_15	retained_17	retained_19	retained_21	\
0	-0.260875	-0.263672	-0.265223	-0.264664	-0.263931	
1	-0.281117	-0.279898	-0.280553	-0.281026	-0.279597	
2	-0.251689	-0.246160	-0.241688	-0.247948	-0.248915	
3	-0.263498	-0.264684	-0.265359	-0.265022	-0.265809	
4	-0.222058	-0.218340	-0.219696	-0.222613	-0.223109	
5	-0.282614	-0.284536	-0.282342	-0.279471	-0.281010	
6	-0.246497	-0.245198	-0.244371	-0.246378	-0.245325	
7	-0.281190	-0.271912	-0.271001	-0.270909	-0.270422	
8	-0.247468	-0.243693	-0.240309	-0.239800	-0.238790	
9	-0.288307	-0.291117	-0.288164	-0.287486	-0.286079	

	retained_23	retained_25	retained_27
0	-0.265986	-0.266418	-0.266297
1	-0.279338	-0.279243	-0.279132
2	-0.248891	-0.247863	-0.247803
3	-0.265517	-0.265307	-0.265233
4	-0.223784	-0.222226	-0.222554
5	-0.280829	-0.281025	-0.280642
6	-0.244888	-0.246278	-0.246078
7	-0.269894	-0.272383	-0.272080
8	-0.238478	-0.237794	-0.237349
9	-0.285950	-0.285323	-0.285951

Plotting The means of negative mean absolute errors for each iteration with number of input variables retained:

```
# Changine the x-axes lables
xlables
=['1','3','5','7','9','11','13','15','17','19','21','23','25','26']
Mapper = {}
for key in df_xvalid_reg.columns.values.tolist():
    for value in xlables:
        Mapper[key] = value
        xlables.remove(value)
```

```
        break
df_xvalid_reg = df_xvalid_reg.rename(columns = Mapper)
xvalid_mean_reg = df_xvalid_reg.mean()

# The plotting code
REFfig = plt.figure(figsize=(10, 6))
REFax = REFfig.add_subplot(1, 1, 1)
REFax.plot(xvalid_mean_reg)
REFax.set_xlabel('Number of input variables retained')
REFax.set_ylabel('Mean of negative mean absolute errors For 10 Cross
Folds')
REFax.set_title("The means of negative mean absolute errors for each
iteration VS Number of input variables retained:")

Text(0.5, 1.0, 'The means of negative mean absolute errors for each
iteration VS Number of input variables retained:')
```



From the above plot we can see no change of direction of the means other than increaseing with increased number of input variables retained, thus we will use all the 26 input variable

```
xvalid_best_reg = cross_val_score(regression, X = X_z, y =
Y_final['SalePrice'], cv = 10, scoring = "neg_mean_absolute_error" )
```

Model 2 (KNN)

```
df_xvalid_knnlwa = pd.DataFrame()
for i in range(2, 26,2):
    # define a kNN instance
    knnlwa_temp = knn.KNeighborsRegressor(n_neighbors=i,
weights='distance')
    df_xvalid_knnlwa['k_' + str(i)] = cross_val_score(knnlwa_temp, X =
X_z, y = Y_final['SalePrice'], cv = 10,
                                                      scoring =
'neg_mean_absolute_error')

df_xvalid_knnlwa.shape
```

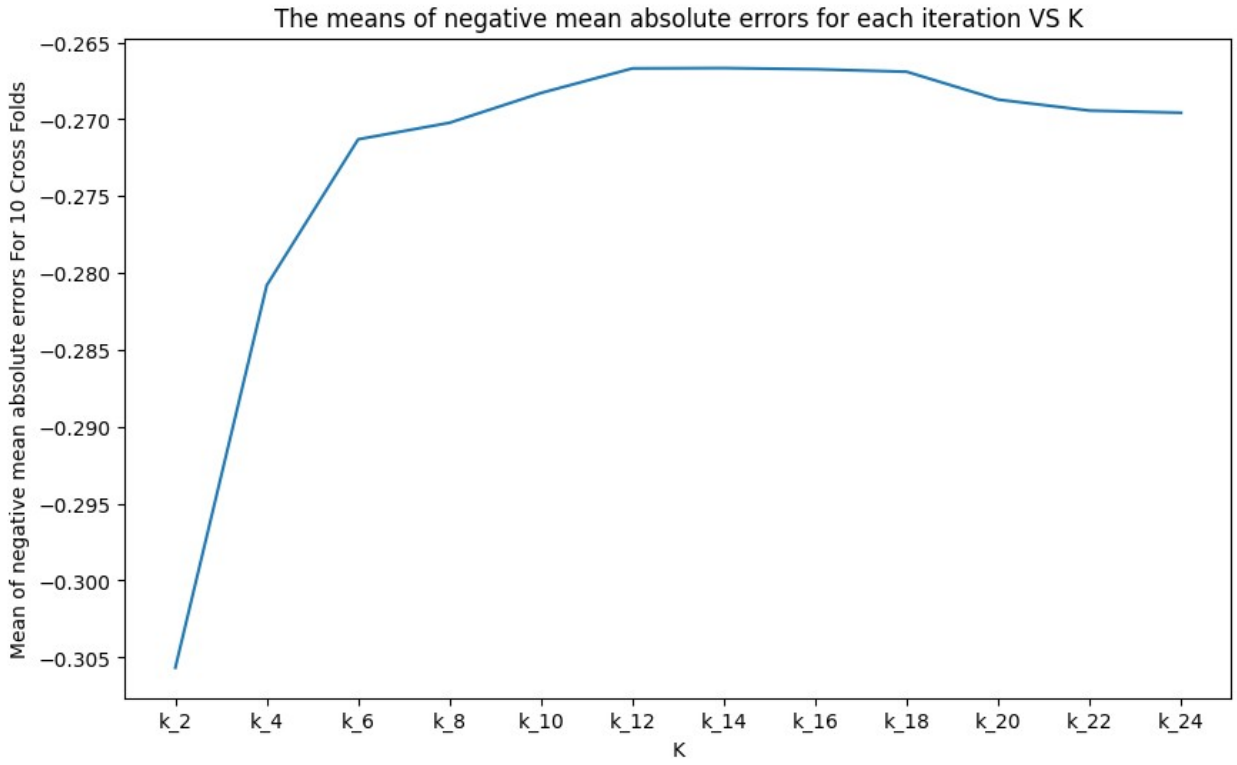
```
(10, 12)
```

```
xvalid_mean_knnlwa = df_xvalid_knnlwa.mean()
xvalid_mean_knnlwa
```

```
k_2    -0.305674
k_4    -0.280804
k_6    -0.271307
k_8    -0.270232
k_10   -0.268293
k_12   -0.266695
k_14   -0.266675
k_16   -0.266750
k_18   -0.266914
k_20   -0.268725
k_22   -0.269439
k_24   -0.269578
dtype: float64
```

```
# The plotting code
KNNfig = plt.figure(figsize=(10, 6))
KNNax = KNNfig.add_subplot(1, 1, 1)
KNNax.plot(xvalid_mean_knnlwa)
KNNax.set_xlabel('K')
KNNax.set_ylabel('Mean of negative mean absolute errors For 10 Cross
Folds')
KNNax.set_title("The means of negative mean absolute errors for each
iteration VS K")
```

```
Text(0.5, 1.0, 'The means of negative mean absolute errors for each
iteration VS K')
```



From the above chart we find that kNN model with k = 12 has the highest x-valid score (largest mean "neg_mean_absolute_error").

```
#Defining a KNN(LWA) Model with K=12:
knnlwa_12 = knn.KNeighborsRegressor(n_neighbors=12,
weights='distance')
xvalid_best_knnlwa = cross_val_score(knnlwa_12, X=X_z,
y=Y_final['SalePrice'], cv=10, scoring='neg_mean_absolute_error')
xvalid_best_knnlwa.mean()
```

```
-0.26669475808339976
```

Scoring

```
Score_Data = pd.read_csv("Score_Data-No_Sale_Price.csv")

FullData.head(3)
```

```
         PID  YearBuilt  YearRemodel  VeneerExterior  BsmtFinTp
BsmtFinSqft  \
0  526301100       1960         1960             112          1
639
1  526350040       1961         1961               0          1
468
2  526351010       1958         1958             108          1
923
```

```
   BsmtUnfinSqft  HeatingQC  FstFlrSqft  SecFlrSqft  ...
GarageCars  \
0            441          0        1656           0  ...           2

1            270          1         896           0  ...           1

2            406          1        1329           0  ...           1


   GarageArea  WdDckSqft  OpenPrchSqft  SalePrice  LotArea  LotShape
BldgTp  \
0         528        210            62     215000    31770         0
1
1         730        140             0     105000    11622         1
1
2         312        393            36     172000    14267         0
1

   OverallQuality  OverallCondition
0               6                 5
1               5                 6
2               6                 6

[3 rows x 28 columns]
```

```
Score_Data.head(5)
```

```
         PID  LotArea  LotShape  BldgTp  OverallQuality
OverallCondition  \
0  528445060     8987         1       1               8
5
1  528456160     9215         1       1               7
5
2  528458070     8640         1       1               7
5
3  906380190     6762         1       1               7
5
4  906385010    10402         0       1               7
5

   YearBuilt  YearRemodel  VeneerExterior  BsmtFinTp  ...
HalfBathHouse  \
0       2005         2006           226.0          0  ...
0
1       2009         2010             0.0          0  ...
0
2       2009         2009             0.0          1  ...
1
3       2006         2006            24.0          1  ...
```

```
0
4       2009          2009              0.0           0  ...
0

   FullBathHouse  BdrmAbvGrnd  RmAbvGrnd  Fireplaces  GarageTp
GarageCars  \
0              2          2.0        6.0           1         3
3
1              2          2.0        4.0           0         3
2
2              2          3.0        7.0           0         3
2
3              2          2.0        6.0           0         3
2
4              2          3.0        6.0           0         3
3

   GarageArea  WdDckSqft  OpenPrchSqft
0         880        144             0
1         676          0           136
2         614        169            45
3         632        105            61
4         740          0            36

[5 rows x 27 columns]
```

```
Score_Data.shape
```

```
(100, 27)
```

```
Score_Data.isnull().sum().sum()
```

```
20
```

```
Score_Data.dtypes
```

```
PID                    int64
LotArea                int64
LotShape               int64
BldgTp                 int64
OverallQuality         int64
OverallCondition       int64
YearBuilt              int64
YearRemodel            int64
VeneerExterior       float64
BsmtFinTp              int64
BsmtFinSqft            int64
BsmtUnfinSqft          int64
HeatingQC              int64
FstFlrSqft           float64
SecFlrSqft           float64
```

```
AbvGrndLiving          int64
FullBathBsmt           int64
HalfBathHouse          int64
FullBathHouse          int64
BdrmAbvGrnd          float64
RmAbvGrnd            float64
Fireplaces             int64
GarageTp               int64
GarageCars             int64
GarageArea             int64
WdDckSqft              int64
OpenPrchSqft           int64
dtype: object
```

Modifying the index values of score DataFrame so that they have no overlap with train DataFrame.

```
Score_Data.index = np.arange(FullData.shape[0] + 100,
FullData.shape[0] + 100 + Score_Data.shape[0], 1)

# sanity check
print('Ending index of train data: {}\nBeginning index of score data:
{}'.format(
    FullData.index.values[-1], Score_Data.index.values[0]))
```

```
Ending index of train data: 2369
Beginning index of score data: 2470
```

Comparing the columns labels of the score and the train data

```
print('Is there any difference in the input column labels between the
two dataframes?',
      (Score_Data.columns != FullData.drop('SalePrice',
axis=1).columns).any())
```

```
Is there any difference in the input column labels between the two
dataframes? True
```

Changing the columns order of the score dataset

```
Score_Data = Score_Data[FullData.drop('SalePrice', axis=1).columns]
```

Sanity Check

```
print('Is there any difference in the input column labels between the
two dataframes?',
      (Score_Data.columns != FullData.drop('SalePrice',
axis=1).columns).any())
```

```
Is there any difference in the input column labels between the two
dataframes? False
```

Understanding and cleaning the high-level structure of score data

```
Score_Data.describe()

                PID    YearBuilt  YearRemodel  VeneerExterior
BsmtFinTp  \
count  1.000000e+02   100.000000    100.00000       97.000000
100.000000
mean   6.576229e+08  1990.730000   1993.37000      137.659794
0.670000
std    1.808138e+08    18.079429     15.26444      194.065490
0.472582
min    5.271631e+08  1921.000000   1950.00000        0.000000
0.000000
25%    5.274515e+08  1974.750000   1977.00000        0.000000
0.000000
50%    5.284571e+08  1999.000000   2000.00000       72.000000
1.000000
75%    9.071821e+08  2003.250000   2004.00000      202.000000
1.000000
max    9.144764e+08  2010.000000   2010.00000     1110.000000
1.000000

       BsmtFinSqft  BsmtUnfinSqft   HeatingQC   FstFlrSqft
SecFlrSqft  ...  \
count   100.000000     100.000000  100.000000    96.000000
96.000000  ...
mean    402.850000     647.390000    2.520000  1115.000000
521.395833  ...
std     378.931309     418.081957    0.822413   321.526294
606.678107  ...
min       0.000000       0.000000    0.000000   483.000000
0.000000  ...
25%       0.000000     300.000000    2.000000   850.000000
0.000000  ...
50%     417.500000     574.500000    3.000000  1053.500000
561.500000  ...
75%     716.250000     867.250000    3.000000  1384.000000
851.000000  ...
max    1350.000000    1704.000000    3.000000  1976.000000
3041.000000  ...

        GarageTp  GarageCars  GarageArea  WdDckSqft  OpenPrchSqft  \
count  100.00000  100.000000  100.000000  100.00000    100.000000
mean     2.61000    2.010000  532.280000  108.53000     48.370000
std      0.75069    0.437971  140.276788  123.17471     45.848599
min      1.00000    1.000000  186.000000    0.00000      0.000000
```

```
25%      3.00000    2.000000  440.000000    0.00000      0.000000
50%      3.00000    2.000000  528.500000  100.00000     43.500000
75%      3.00000    2.000000  614.250000  192.00000     73.250000
max      3.00000    3.000000  908.000000  576.00000    224.000000
```

```
             LotArea    LotShape      BldgTp  OverallQuality
OverallCondition
count    100.000000  100.000000  100.000000      100.000000
100.000000
mean    9322.570000    0.460000    0.910000        6.520000
5.350000
std     2775.829625    0.500908    0.287623        1.029661
0.715979
min     1680.000000    0.000000    0.000000        4.000000
4.000000
25%     8126.500000    0.000000    1.000000        6.000000
5.000000
50%     9008.500000    0.000000    1.000000        7.000000
5.000000
75%    10461.000000    1.000000    1.000000        7.000000
5.250000
max    16900.000000    1.000000    1.000000        9.000000
8.000000

[8 rows x 27 columns]
```

```
Score_Data.isnull().sum()
```

```
PID                0
YearBuilt          0
YearRemodel        0
VeneerExterior     3
BsmtFinTp          0
BsmtFinSqft        0
BsmtUnfinSqft      0
HeatingQC          0
FstFlrSqft         4
SecFlrSqft         4
AbvGrndLiving      0
FullBathBsmt       0
HalfBathHouse      0
FullBathHouse      0
BdrmAbvGrnd        5
RmAbvGrnd          4
Fireplaces         0
GarageTp           0
GarageCars         0
GarageArea         0
WdDckSqft          0
OpenPrchSqft       0
```

```
LotArea              0
LotShape             0
BldgTp               0
OverallQuality       0
OverallCondition     0
dtype: int64
```

```
Score_Data.columns[Score_Data.isnull().sum().astype(bool)].tolist()
```

```
['VeneerExterior', 'FstFlrSqft', 'SecFlrSqft', 'BdrmAbvGrnd',
'RmAbvGrnd']
```

Preprocessing Score Dataset

```
# Adding suffix "_z" to the columns names of Score_Data:
Scorelabels =
Score_Data.drop("PID",axis=1).add_suffix('_z').columns.tolist()

#Fitting the Standard Scaler to X with ID column "PID" dropped:
Score_z =
pd.DataFrame(preprocessor.transform(Score_Data.drop("PID",axis=1)),
columns = Scorelabels)
Score_z
```

```
    YearBuilt_z  YearRemodel_z  VeneerExterior_z  BsmtFinTp_z
BsmtFinSqft_z  \
0      1.143708       1.059122         -1.544576     0.896603
0.334062
1      1.276585       1.252453         -1.544576     0.896603      -
0.512172
2      1.276585       1.204121          0.647427     0.896603
0.226319
3      1.176927       1.059122          0.647427     0.896603      -
0.534618
4      1.276585       1.204121         -1.544576     0.896603      -
0.494214
..          ...            ...               ...          ...
...
95     1.143708       1.059122         -1.544576     0.896603
2.944591
96     1.077270       0.962456          0.647427     0.896603
2.742572
97     1.077270       0.962456          0.647427     0.896603
1.346400
98     1.077270       0.962456          0.647427     0.896603
0.289169
99     1.010831       0.865791          0.647427     0.896603
2.699924
```

```
    BsmtUnfinSqft_z  HeatingQC_z  FstFlrSqft_z  SecFlrSqft_z
AbvGrndLiving_z  \
0         -0.844054    -0.769897      0.980787     -1.099367
0.185767
1         -0.844054    -0.769897      0.980787     -1.099367
1.622208
2         -0.844054     1.298874      0.980787      0.247145
0.532453
3          1.184758    -0.769897      0.980787     -1.099367
0.185767
4         -0.844054    -0.769897      0.980787      0.247145
0.185767
..              ...          ...           ...           ...
...
95        -0.844054     1.298874      0.980787      1.593656
1.968893
96         1.184758     1.298874      0.980787      0.247145
2.687113
97         1.184758     1.298874      0.980787      0.247145
1.250673
98         1.184758    -0.769897      0.980787     -1.099367
0.185767
99         1.184758     1.298874      0.980787      1.593656
2.687113

    ...  GarageTp_z  GarageCars_z  GarageArea_z  WdDckSqft_z
OpenPrchSqft_z  \
0   ...    0.374127      1.556806     -0.575434     1.358838
1.494135
1   ...    0.374127      0.758185     -0.575434    -0.762629
1.494135
2   ...    0.374127      0.758185     -0.575434    -0.762629
0.385340
3   ...    0.374127      0.758185     -0.575434     0.496136
0.756036
4   ...    0.374127      0.758185     -0.575434    -0.762629
1.494135
..  ...         ...           ...           ...           ...
...
95  ...    0.374127      2.355427     -0.575434     1.545640
1.494135
96  ...    0.374127      2.355427     -0.575434     1.779161
0.891162
97  ...    0.374127      0.758185     -0.575434     1.458645
0.836195
98  ...    0.374127      1.556806     -0.575434     1.671531
0.933499
99  ...    0.374127      1.556806     -0.575434     1.979863
```

```
0.910588

    LotArea_z  LotShape_z  BldgTp_z  OverallQuality_z
OverallCondition_z
0    1.006766    1.323588 -0.869290          0.981569           -
1.073082
1    0.836632    0.437438 -0.869290         -0.943665
1.219010
2    0.515493   -1.094706  1.173771          1.043103
0.710583
3    0.276492    0.410350 -0.869290          0.860372
0.849643
4    0.840762    0.458949 -0.869290         -0.943665
0.609151
..        ...         ...       ...               ...
...
95   1.048477    1.575376  1.259510          1.240259
0.802225
96   0.342132    1.275853  1.276754          1.375080           -
1.073082
97  -0.562122   -0.190846  1.260691          1.288999
0.857098
98   0.214427    1.282118 -0.869290          1.137532
0.495654
99   0.140929    1.154476  1.287193         -0.943665
0.842068

[100 rows x 26 columns]

Score_z.describe().round(4)

       YearBuilt_z  YearRemodel_z  VeneerExterior_z  BsmtFinTp_z  \
count     100.0000       100.0000          100.0000     100.0000
mean        0.6697         0.4487           -0.0759       0.3881
std         0.6006         0.7378            1.0359       0.8712
min        -1.6467        -1.6475           -1.5446      -2.2813
25%         0.1388        -0.3425           -1.5446      -0.1627
50%         0.9444         0.7691            0.6474       0.8966
75%         1.0856         0.9625            0.6474       0.8966
max         1.3098         1.2525            0.6474       0.8966

       BsmtFinSqft_z  BsmtUnfinSqft_z  HeatingQC_z  FstFlrSqft_z  \
count       100.0000         100.0000     100.0000      100.0000
mean          0.1834           0.1095       0.3266        2.0174
std           0.8749           1.0177       1.0377        7.1933
min          -1.3068          -0.8441      -0.7699       -1.0128
25%          -0.4140          -0.8441      -0.7699        0.9808
50%           0.1152          -0.8441       1.2989        0.9808
75%           0.6079           1.1848       1.2989        0.9808
max           2.9446           1.1848       1.2989       36.8645
```

```
       SecFlrSqft_z  AbvGrndLiving_z  ...  GarageTp_z  GarageCars_z  \
count      100.0000         100.0000  ...    100.0000      100.0000
mean        -0.0222           0.1590  ...      0.0999        0.3748
std          0.6900           0.8631  ...      0.8764        0.8223
min         -1.0994          -1.6222  ...     -2.6729       -1.6377
25%         -0.0895          -0.1858  ...      0.3741       -0.0404
50%          0.2471          -0.1858  ...      0.3741        0.7582
75%          0.2471           0.5325  ...      0.3741        0.7582
max          1.5937           2.6871  ...      0.3741        2.3554

       GarageArea_z  WdDckSqft_z  OpenPrchSqft_z  LotArea_z
LotShape_z  \
count      100.0000     100.0000        100.0000   100.0000
100.0000
mean        -0.2559       0.3014         -0.0770     0.2284
0.0034
std          0.6536       1.0665          1.0338     0.7180
0.9711
min         -1.4883      -0.7626         -1.4941    -3.6497     -
2.6001
25%         -0.5754      -0.7626         -1.4941    -0.0464     -
0.7261
50%         -0.5754       0.6355          0.5852     0.3628     -
0.0315
75%         -0.3472       1.3063          0.7709     0.6224
0.8287
max          2.1631       1.9799          0.9890     1.0485
2.0276

       BldgTp_z  OverallQuality_z  OverallCondition_z
count  100.0000          100.0000            100.0000
mean     0.2301            0.1691              0.2806
std      1.0439            1.0040              0.8942
min     -0.8693           -0.9437             -1.0731
25%     -0.8693           -0.9437             -1.0731
50%      1.0389            0.8417              0.6949
75%      1.1984            1.0922              0.9336
max      1.5894            1.5158              1.4501

[8 rows x 26 columns]
```

We can see means and SDs that are different from the 0 and 1 but that's becuase the trasformation should not be fitted to the test data

Checking for missing data after preprocessing

```
Score_z.isnull().sum().sum()

0
```

Comparing the 2 Models

```
print('Is the best x-valid regression model better than the best x-valid kNN (LWA) model?',
      xvalid_best_reg.mean() > xvalid_best_knnlwa.mean())
```

```
Is the best x-valid regression model better than the best x-valid kNN (LWA) model? True
```

We find that the regression model is better than the KNN model so we will use it for predictions.

Scoring with Model 1 (Linear Regression)

Sanity Check

```
print('Is there any difference in the input column labels between the two dataframes?',
      (Score_z.columns != X_z.columns).any())
```

```
Is there any difference in the input column labels between the two dataframes? True
```

```
Score_z = Score_z[X_z.columns]
```

```
print('Is there any difference in the input column labels between the two dataframes?',
      (Score_z.columns != X_z.columns).any())
```

```
Is there any difference in the input column labels between the two dataframes? False
```

```
#Fitting The Model
Model_1= regression.fit(X=X_z, y=Y_final['SalePrice'])
```

```
#Predicting with the model
SalePrice_Reg = Model_1.predict(X =Score_z)
```

```
print('Mean and Std of predicted Sale Price are: {:.3f} ({:.3f})'.format(np.mean(SalePrice_Reg), np.std(SalePrice_Reg)))
```

```
Mean and Std of predicted Sale Price are: 0.200 (0.771)
```

```
SalePrice_Reg.shape
```

```
(100,)
```

```
#Inversing the standarization
SalePrice_Reg_inversed = pd.DataFrame(yScaler.inverse_transform(SalePrice_Reg.reshape(-1, 1)), columns=['PSP'],
                                      index=Score_Data.index)
```

```
#Joining The PID column
SalePrice_Reg_inversed =
SalePrice_Reg_inversed.join(Score_Data["PID"])

#Checking for Missing Values
SalePrice_Reg_inversed.isnull().sum()
```

```
PSP    0
PID    0
dtype: int64
```

```
#Final Data
SalePrice_Reg_inversed = SalePrice_Reg_inversed[["PID","PSP"]]
SalePrice_Reg_inversed
```

	PID	PSP
2470	528445060	182048.188705
2471	528456160	99245.365459
2472	528458070	224638.151682
2473	906380190	180267.590243
2474	906385010	122160.562507
...	...	...
2565	528118040	301701.484160
2566	528166120	338084.738825
2567	528170070	264623.214996
2568	528172080	190840.217873
2569	528178070	291621.354393

```
[100 rows x 2 columns]
```

```
SalePrice_Reg_inversed.to_csv("Final Predictions (Sale Price).csv",
index=False)
```

Bouns Question

```
# The top 20 priced predicted houses
SalePrice_Reg_inversed.sort_values("PSP",
ascending=False).iloc[0:20,:]
```

	PID	PSP
2566	528166120	338084.738825
2565	528118040	301701.484160
2569	528178070	291621.354393
2507	528250030	275456.704315
2501	527163100	271890.677390
2499	907285020	268962.309519
2522	907254090	266510.579284
2567	528170070	264623.214996
2477	528235090	259537.507766

```
2491  534128020  253978.090767
2510  528280180  249831.512767
2490  534128010  243161.514164
2517  907187030  243081.875208
2520  907253110  239493.233908
2528  527325110  238683.979761
2529  527327050  237204.549705
2495  907185050  236039.886857
2537  528221060  235002.961963
2521  907254020  234546.133545
2526  907412070  232097.872730
```

Conclusion

We started by taking a look at the provided datasets and knowing more about problems with them, then we merged the 4 datasets to get one dataset to input into the models, EDA was done on the final merged dataset to know about the realations between the variables, then data was prepared to go into the model, Linear Regression models and KNN models were compared to see which model type and variation has the lease mean absolute error while training, afterthat the Score dataset was prepared and feed into the chosen linear regression model that has the least error, Finally predictions were made on the score set to get the Sale Price of 100 houses.