HousePrice_Predictor

October 6, 2024

[1]:
```
!kaggle competitions download -c house-prices-advanced-regression-techniques
```

```
house-prices-advanced-regression-techniques.zip: Skipping, found more recently
modified local copy (use --force to force download)
```

[2]:
```
import pandas as pd
import numpy as np
```

[3]:
```
pd.set_option('display.max_columns', None)

train_df = pd.read_csv('DATA/train.csv')
test_df = pd.read_csv('DATA/test.csv')
submission_sample = pd.read_csv('DATA/sample_submission.csv')
```

[4]:
```
train_df.head()
```

[4]:

	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
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	2Story	7	5	2003	2003	Gable	CompShg	VinylSd	VinylSd	BrkFace
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	1Fam	1Story	6	8	1976	1976	Gable	CompShg	MetalSd	MetalSd	None
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	2Story	7	5	2001	2002	Gable	CompShg	VinylSd	VinylSd	BrkFace
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	1Fam	2Story	7	5	1915	1970	Gable	CompShg	Wd Sdng	Wd Shng	None
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	1Fam	2Story	8	5	2000	2000	Gable	CompShg	VinylSd	VinylSd	BrkFace

```
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
```

[5]:
```
submission_sample.head()
```

[5]:
```
     Id      SalePrice
0  1461  169277.052498
1  1462  187758.393989
2  1463  183583.683570
3  1464  179317.477511
4  1465  150730.079977
```

[6]:
```
train_df.describe()
```

[6]:
```
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726
std       1.112799    30.202904     20.645407   181.066207   456.098091
```

```
min       1.000000  1872.000000   1950.000000     0.000000     0.000000
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000

        BsmtFinSF2    BsmtUnfSF  TotalBsmtSF     1stFlrSF     2ndFlrSF  \
count  1460.000000  1460.000000  1460.000000  1460.000000  1460.000000
mean     46.549315   567.240411  1057.429452  1162.626712   346.992466
std     161.319273   441.866955   438.705324   386.587738   436.528436
min       0.000000     0.000000     0.000000   334.000000     0.000000
25%       0.000000   223.000000   795.750000   882.000000     0.000000
50%       0.000000   477.500000   991.500000  1087.000000     0.000000
75%       0.000000   808.000000  1298.250000  1391.250000   728.000000
max    1474.000000  2336.000000  6110.000000  4692.000000  2065.000000

       LowQualFinSF    GrLivArea  BsmtFullBath  BsmtHalfBath     FullBath  \
count   1460.000000  1460.000000   1460.000000   1460.000000  1460.000000
mean       5.844521  1515.463699      0.425342      0.057534     1.565068
std       48.623081   525.480383      0.518911      0.238753     0.550916
min        0.000000   334.000000      0.000000      0.000000     0.000000
25%        0.000000  1129.500000      0.000000      0.000000     1.000000
50%        0.000000  1464.000000      0.000000      0.000000     2.000000
75%        0.000000  1776.750000      1.000000      0.000000     2.000000
max      572.000000  5642.000000      3.000000      2.000000     3.000000

          HalfBath  BedroomAbvGr  KitchenAbvGr  TotRmsAbvGrd   Fireplaces  \
count  1460.000000   1460.000000   1460.000000   1460.000000  1460.000000
mean      0.382877      2.866438      1.046575      6.517808     0.613014
std       0.502885      0.815778      0.220338      1.625393     0.644666
min       0.000000      0.000000      0.000000      2.000000     0.000000
25%       0.000000      2.000000      1.000000      5.000000     0.000000
50%       0.000000      3.000000      1.000000      6.000000     1.000000
75%       1.000000      3.000000      1.000000      7.000000     1.000000
max       2.000000      8.000000      3.000000     14.000000     3.000000

       GarageYrBlt   GarageCars   GarageArea   WoodDeckSF  OpenPorchSF  \
count  1379.000000  1460.000000  1460.000000  1460.000000  1460.000000
mean   1978.506164     1.767123   472.980137    94.244521    46.660274
std      24.689725     0.747315   213.804841   125.338794    66.256028
min    1900.000000     0.000000     0.000000     0.000000     0.000000
25%    1961.000000     1.000000   334.500000     0.000000     0.000000
50%    1980.000000     2.000000   480.000000     0.000000    25.000000
75%    2002.000000     2.000000   576.000000   168.000000    68.000000
max    2010.000000     4.000000  1418.000000   857.000000   547.000000

       EnclosedPorch    3SsnPorch  ScreenPorch     PoolArea       MiscVal  \
```

```
count    1460.000000  1460.000000  1460.000000  1460.000000   1460.000000
mean       21.954110     3.409589    15.060959     2.758904     43.489041
std        61.119149    29.317331    55.757415    40.177307    496.123024
min         0.000000     0.000000     0.000000     0.000000      0.000000
25%         0.000000     0.000000     0.000000     0.000000      0.000000
50%         0.000000     0.000000     0.000000     0.000000      0.000000
75%         0.000000     0.000000     0.000000     0.000000      0.000000
max       552.000000   508.000000   480.000000   738.000000  15500.000000
```

```
             MoSold       YrSold      SalePrice
count   1460.000000  1460.000000    1460.000000
mean       6.321918  2007.815753  180921.195890
std        2.703626     1.328095   79442.502883
min        1.000000  2006.000000   34900.000000
25%        5.000000  2007.000000  129975.000000
50%        6.000000  2008.000000  163000.000000
75%        8.000000  2009.000000  214000.000000
max       12.000000  2010.000000  755000.000000
```

Data Cleanup on the test Data Lets look at the no of nulls in the training dataset

[7]:
```
pd.set_option('display.max_rows', None)
train_df.isnull().sum()
```

[7]:
```
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
```

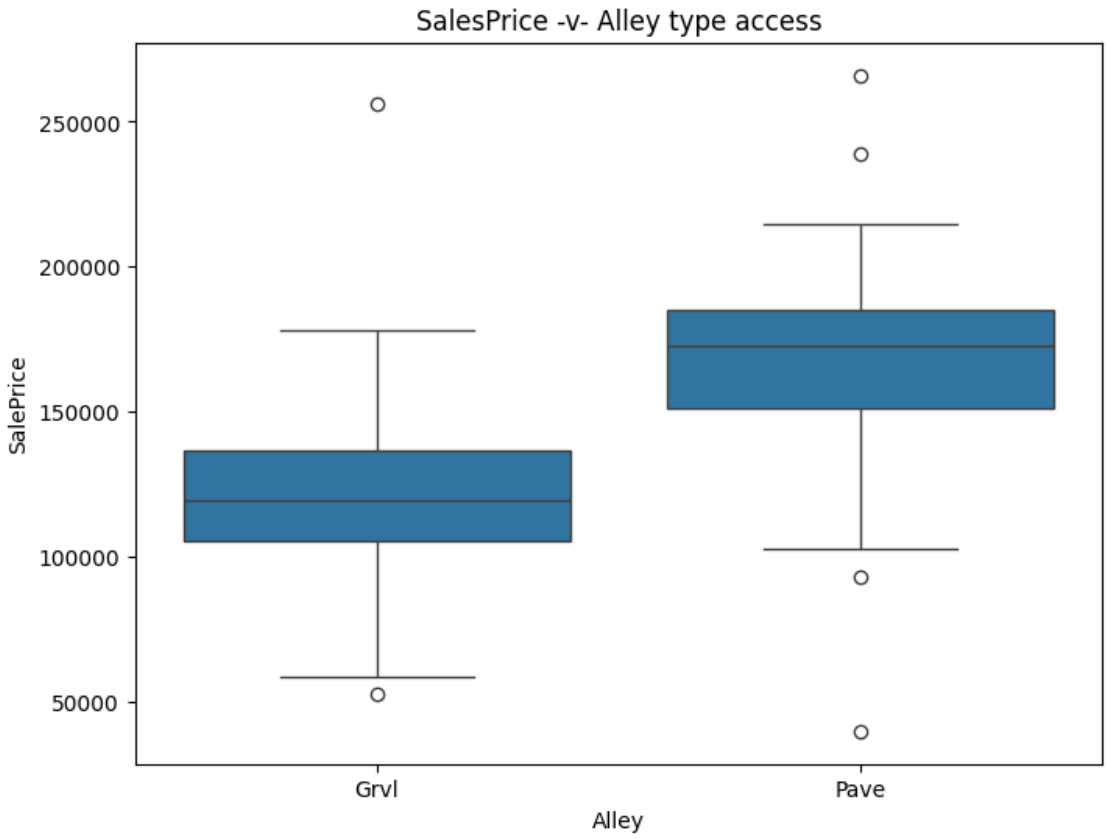
LotFrontage column Nulls dealt with as the average figure for that column

[8]:
```
train_df['LotFrontage'].fillna(train_df['LotFrontage'].median(), inplace=True)
```

```
/tmp/ipykernel_16577/861508967.py:1: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  train_df['LotFrontage'].fillna(train_df['LotFrontage'].median(), inplace=True)
```

Alley column - Lets see if there is any corolation between the sales price and Alley access with the house

[9]:
```
import seaborn as sns
import matplotlib.pyplot as plt

alley_price_comparison = train_df.groupby('Alley')['SalePrice'].mean()
#Boxplot chart to see SalesPrice distribution amoungst alley types
plt.figure(figsize=(8,6))
sns.boxplot(data=train_df, x='Alley', y='SalePrice')
plt.title('SalesPrice -v- Alley type access')
plt.show()
```



Paved Alley access does seem to increase the average value of the property, so we will keep the column but replace the nulls with 'None'

[10]:
```
train_df['Alley'] = train_df['Alley'].astype('object')
train_df['Alley'].fillna('None', inplace=True)
```

```
/tmp/ipykernel_16577/4038184766.py:2: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  train_df['Alley'].fillna('None', inplace=True)
```

[11]:
```
train_df['MasVnrType'].fillna('None', inplace=True)
```

```
/tmp/ipykernel_16577/948743105.py:1: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  train_df['MasVnrType'].fillna('None', inplace=True)
```

Now lets anaylze the BsmtQual column to see if Basement quality has an effect on the SalesPrice

[12]:
```
bsmtqual = train_df.groupby('BsmtQual')['SalePrice'].mean()
print(bsmtqual)
```

```
BsmtQual
Ex    327041.041322
Fa    115692.028571
Gd    202688.478964
TA    140759.818182
Name: SalePrice, dtype: float64
```

[13]:
```
train_df.fillna({'BsmtQual' : 'None'}, inplace=True)
```

[14]:
```
train_df.fillna({'BsmtCond' : 'None'}, inplace=True)
train_df.fillna({'BsmtExposure': 'None'}, inplace=True)
train_df.fillna({'BsmtFinType1': 'None'}, inplace=True)
train_df.fillna({'BsmtFinType2': 'None'}, inplace=True)
```

Analyzing the Electrics column

[15]:
```
electrics = train_df.groupby('Electrical')['SalePrice'].mean()
print(electrics)
```

```
Electrical
FuseA    122196.893617
FuseF    107675.444444
FuseP     97333.333333
Mix       67000.000000
SBrkr    186825.113193
Name: SalePrice, dtype: float64
```

[16]:
```
train_df.dropna(subset=['Electrical'], inplace=True)
```

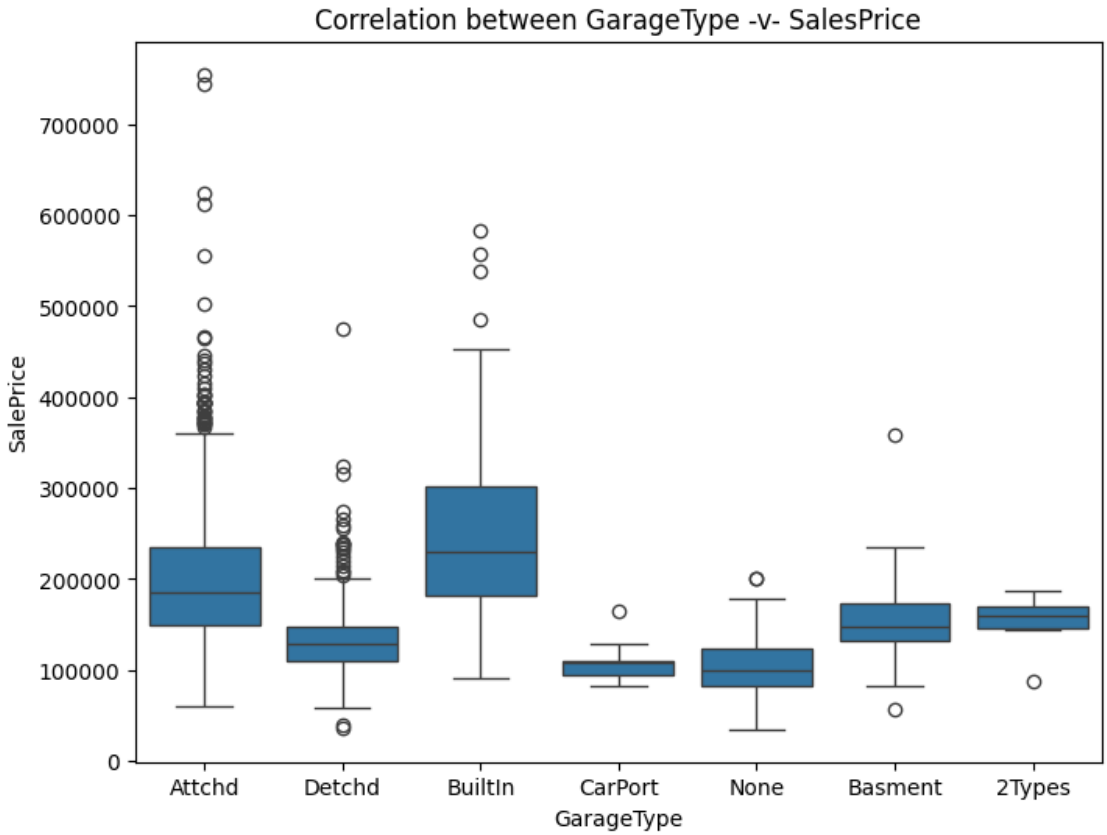
[17]:
```
fireplaces = train_df.groupby('FireplaceQu')['SalePrice'].mean()
print(fireplaces)
```

```
FireplaceQu
Ex    337712.500000
Fa    167298.484848
Gd    226351.415789
Po    129764.150000
TA    205723.488818
Name: SalePrice, dtype: float64
```

[18]:
```
train_df.fillna({'FireplaceQu':'None'}, inplace=True)
```

Visualize the correlation between the SalesPrices of homes against the property having a garage or not

[19]:
```
train_df.fillna({'GarageType':'None'}, inplace=True)

plt.figure(figsize=(8,6))
sns.boxplot(data=train_df, x='GarageType', y='SalePrice')
plt.title('Correlation between GarageType -v- SalesPrice')

plt.show()
```

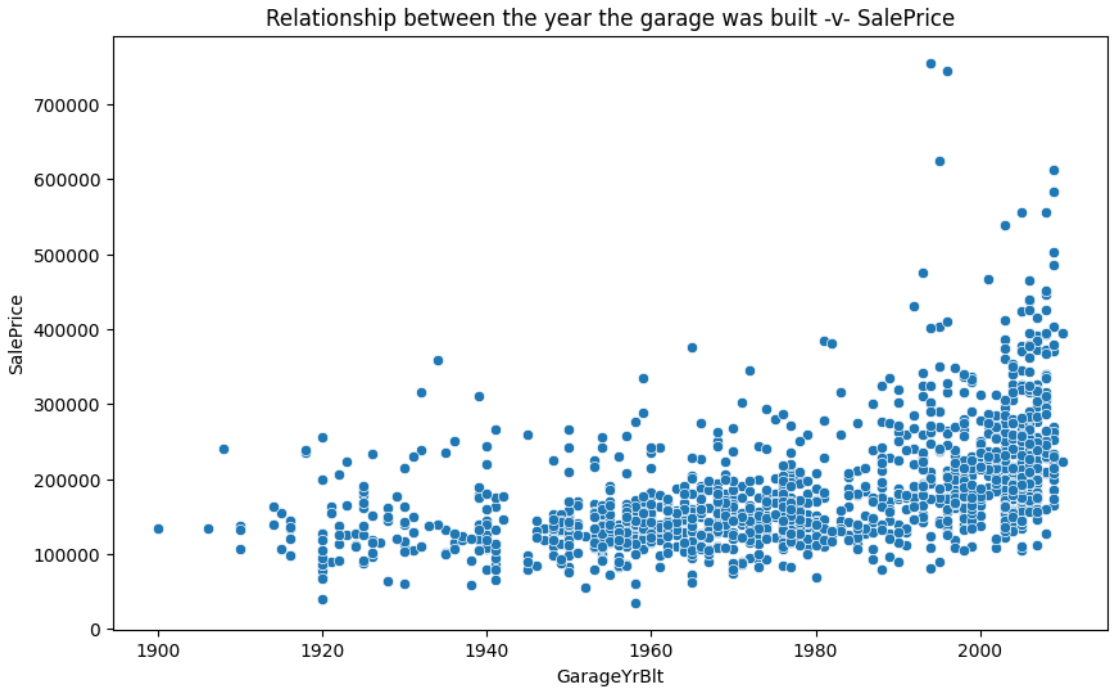


[20]:
```
train_df['GarageYrBlt'].unique()
```

[20]:
```
array([2003., 1976., 2001., 1998., 2000., 1993., 2004., 1973., 1931.,
       1939., 1965., 2005., 1962., 2006., 1960., 1991., 1970., 1967.,
       1958., 1930., 2002., 1968., 2007., 2008., 1957., 1920., 1966.,
       1959., 1995., 1954., 1953.,   nan, 1983., 1977., 1997., 1985.,
       1963., 1981., 1964., 1999., 1935., 1990., 1945., 1987., 1989.,
       1915., 1956., 1948., 1974., 2009., 1950., 1961., 1921., 1900.,
       1979., 1951., 1969., 1936., 1975., 1971., 1923., 1984., 1926.,
       1955., 1986., 1988., 1916., 1932., 1972., 1918., 1980., 1924.,
       1996., 1940., 1949., 1994., 1910., 1978., 1982., 1992., 1925.,
       1941., 2010., 1927., 1947., 1937., 1942., 1938., 1952., 1928.,
       1922., 1934., 1906., 1914., 1946., 1908., 1929., 1933.])
```

Analyze what effect the year the garage was bult against price

[21]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(data=train_df, x='GarageYrBlt', y='SalePrice')
plt.title('Relationship between the year the garage was built -v- SalePrice')
plt.show()
```



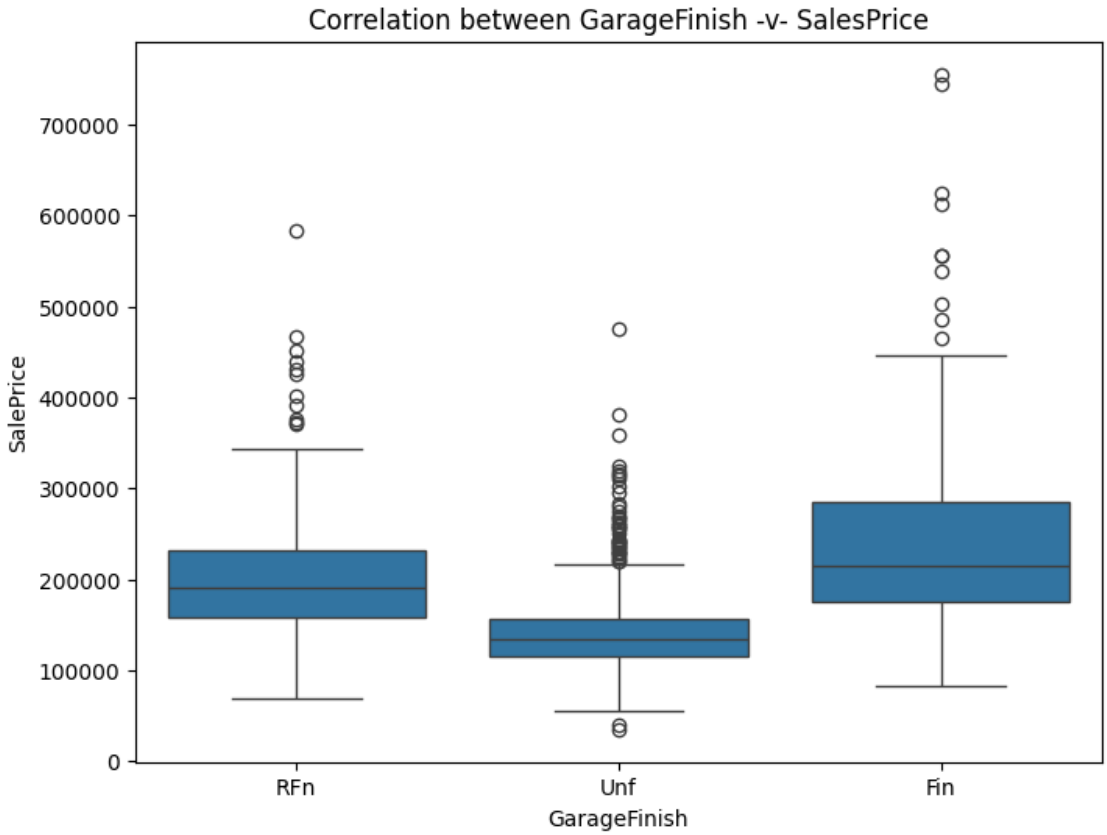
In my view correlation is week so will drop the GarageYrBlt column

[22]:
```
train_df.drop(columns=['GarageYrBlt'], inplace=True)
```

[23]:
```
train_df['GarageFinish'].unique()
```

[23]:
```
array(['RFn', 'Unf', 'Fin', nan], dtype=object)
```

[24]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=train_df, x='GarageFinish', y='SalePrice')
plt.title('Correlation between GarageFinish -v- SalesPrice')

plt.show()
```



Set all the other nulls in the other Garage finish, quality and condition to 'None'

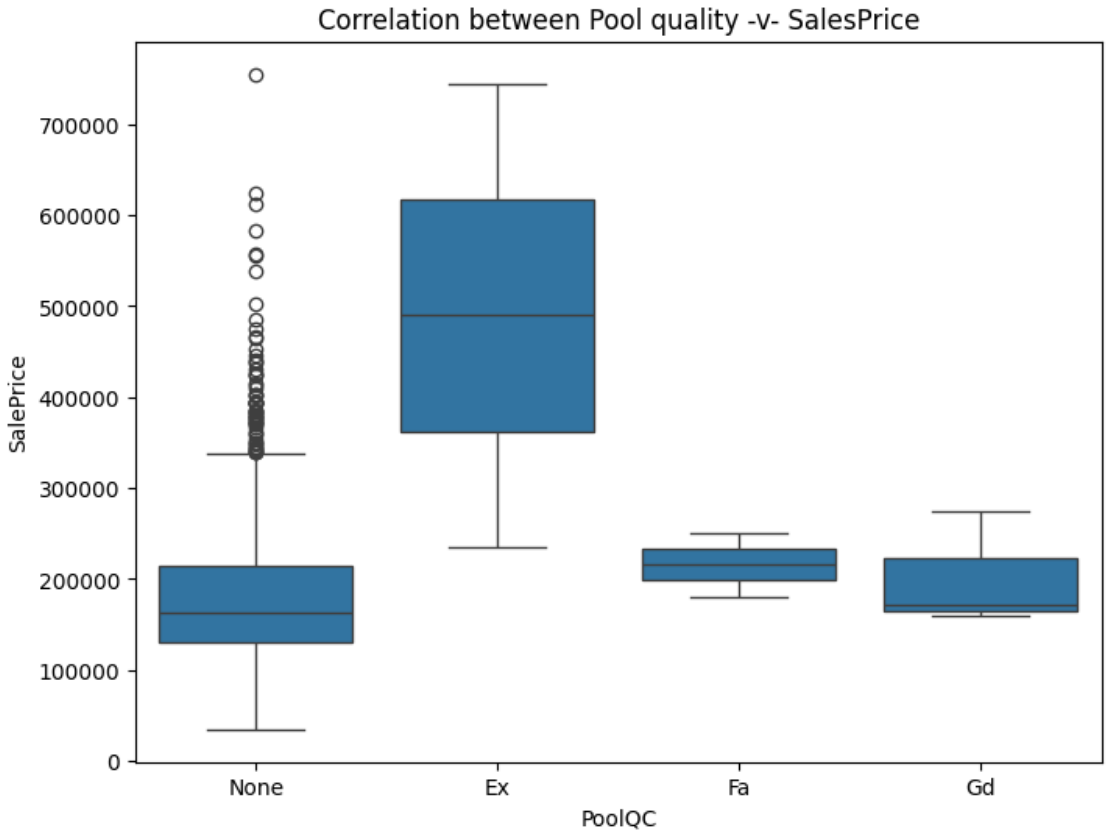
[25]:
```
train_df.fillna({'GarageFinish':'None'},inplace=True)
```

[26]:
```
train_df.fillna({'GarageQual':'None'},inplace=True)
train_df.fillna({'GarageCond':'None'},inplace=True)
```

[27]:
```
train_df['PoolQC'].unique()
```

[27]:
```
array([nan, 'Ex', 'Fa', 'Gd'], dtype=object)
```

[28]:
```
train_df.fillna({'PoolQC':'None'},inplace=True)

plt.figure(figsize=(8,6))
sns.boxplot(data=train_df, x='PoolQC', y='SalePrice')
plt.title('Correlation between Pool quality -v- SalesPrice')

plt.show()
```

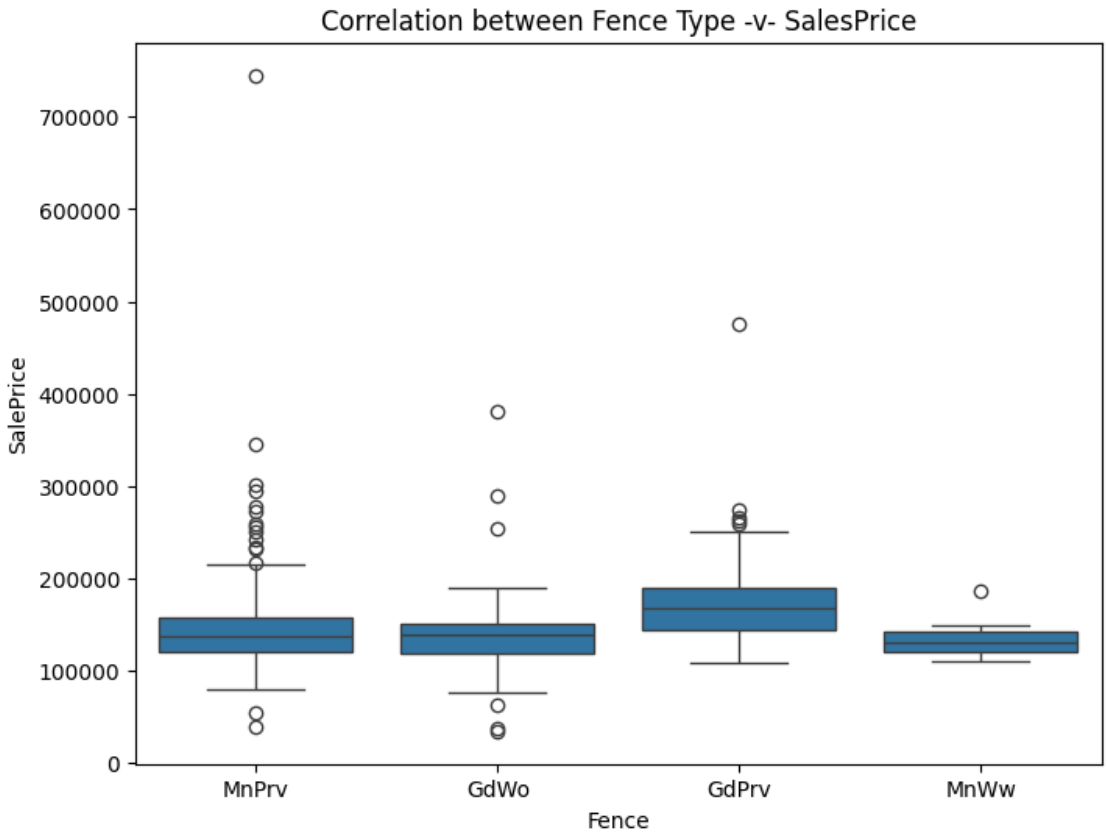


Strong correlation between the pool quality and those houses that don't have pools with SalePrice

[29]:
```
train_df['Fence'].unique()
```

[29]:
```
array([nan, 'MnPrv', 'GdWo', 'GdPrv', 'MnWw'], dtype=object)
```

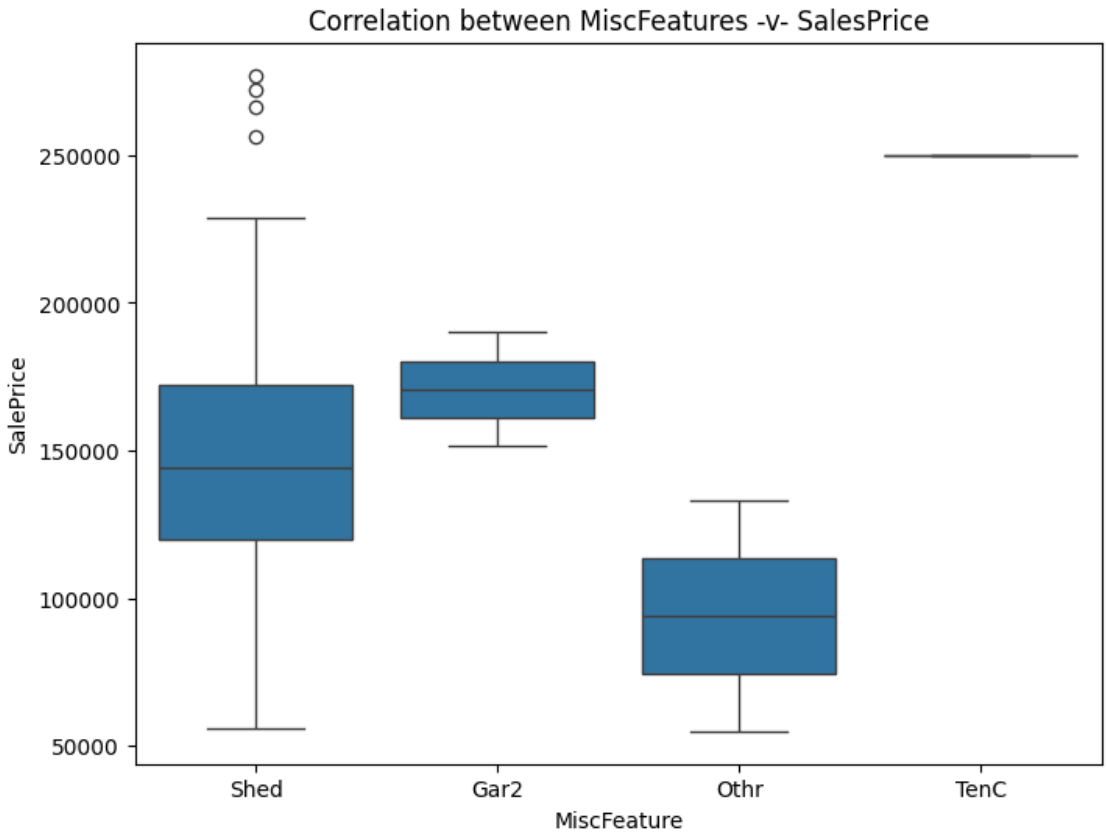
[30]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=train_df, x='Fence', y='SalePrice')
plt.title('Correlation between Fence Type -v- SalesPrice')

plt.show()
```



Regarding how the type of fencing effects the SalePrice there does not seem to be much change in the price compared to the type of fence, so will drop the Fence column

[31]:
```
train_df.drop(columns='Fence', inplace=True)
```

[32]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=train_df, x='MiscFeature', y='SalePrice')
plt.title('Correlation between MiscFeatures -v- SalesPrice')

plt.show()
```



There appears to be an affect on House Sales Prices for Tennis Courts, but for the other features it is mixed so will drop this column.

[33]:
```
train_df.fillna({'MasVnrArea':train_df['MasVnrArea'].mean()}, inplace=True)
train_df.
  ↪drop(columns=['Id','MiscFeature','Utilities','Street','Condition2','LandSlope','3SsnPorch',
  ↪inplace=True)
train_df.isnull().sum()
```

[33]:
```
MSSubClass      0
MSZoning        0
LotFrontage     0
LotArea         0
Alley           0
LotShape        0
LandContour     0
LotConfig       0
Neighborhood    0
Condition1      0
BldgType        0
HouseStyle      0
```

```
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
RoofStyle        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
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
GarageFinish     0
GarageCars       0
GarageArea       0
GarageQual       0
GarageCond       0
PavedDrive       0
WoodDeckSF       0
```

```
OpenPorchSF      0
EnclosedPorch    0
ScreenPorch      0
PoolArea         0
PoolQC           0
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
dtype: int64
```

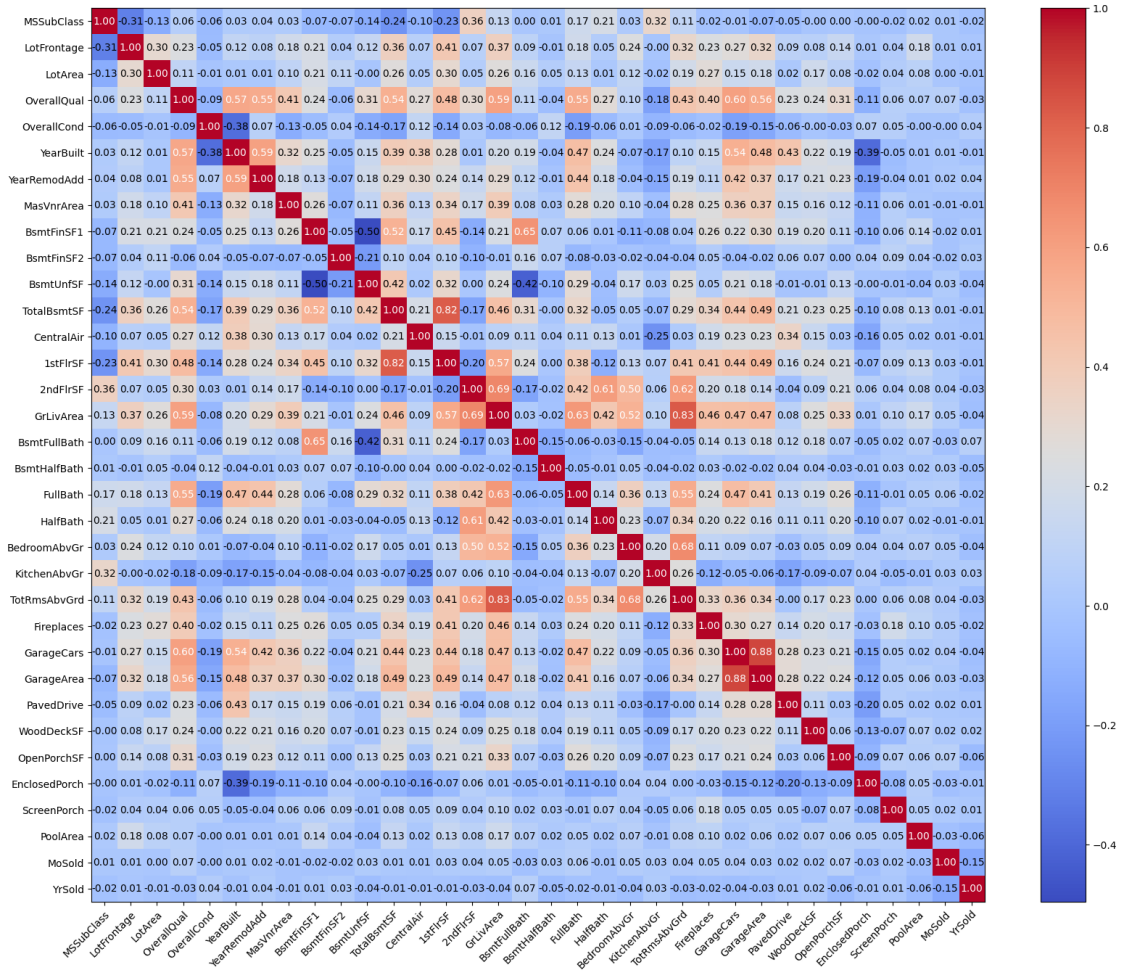
[34]:
```
y = train_df['SalePrice']
train_df.drop(columns=['SalePrice'], inplace=True)
X = train_df
```

[ ]:
```
```

[35]:
```
# Now get the rest of the data ready for training and testing

from sklearn.model_selection import train_test_split, learning_curve,␣
 ↪GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder


le=LabelEncoder()
X['PavedDrive'] = le.fit_transform(X['PavedDrive'])
X['MSSubClass'] = le.fit_transform(X['MSSubClass'])

## We will use mapping for the CentralAir column as these are binary columns
X['CentralAir'] = X['CentralAir'].map({'Y':1, 'N':0})


#With the categorical columns we will apply the get dummies method on them
#Label encode the categorical column MSSubClass
columns_to_dummy = [
    'MSZoning', 'Alley', 'LotShape', 'LandContour', 'LotConfig', 'Neighborhood',
    'Condition1', 'BldgType', 'HouseStyle', 'RoofStyle', 'Exterior1st',␣
 ↪'Exterior2nd',
    'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',␣
 ↪'BsmtCond',
    'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC',
     'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu',
    'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond',
    'PoolQC', 'SaleCondition','SaleType'
]
```

```
X = pd.get_dummies(X,columns=columns_to_dummy, drop_first=True)

#Produce a correlation matrix to view how each column relates to each other

from mlxtend.plotting import heatmap
numeric_cols = X.select_dtypes(include=[np.number])
cm = np.corrcoef(numeric_cols.values.T)
plt.figure(figsize=(20,16))
hm = heatmap(cm, row_names=numeric_cols.columns, column_names=numeric_cols.
  ↪columns,figsize=(20,16), cmap="coolwarm")

plt.show()
```

```
<Figure size 2000x1600 with 0 Axes>
```



[36]:
```
# Now split the data into our train and testing sets
```

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=43)

#Convert the numerical columns into a standard scale of a median of 0 and a std␣
 ↪of 1
sc=StandardScaler()
numerical_features =␣
 ↪['LotFrontage','LotArea','MasVnrArea','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','

X_train[numerical_features] = sc.fit_transform(X_train[numerical_features])
X_val[numerical_features] = sc.transform(X_val[numerical_features])
```

[37]:
```
#X_train.head(20)
```

[38]:
```
# Training our model using LinearRegression method
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
import xgboost as xgb
import cupy as cp


#model = LinearRegression()

#dtrain = xgb.DMatrix(X_train, label=y_train)
#dtest = xgb.DMatrix(X_val, label=y_val)

X_train = X_train.astype({col: 'int64' for col in X_train.
 ↪select_dtypes(include=['bool']).columns})

X_train_gpu = cp.array(X_train)
y_train_gpu = cp.array(y_train)

model = XGBRegressor(tree_method='hist', early_stopping_rounds=10, device="cuda:
 ↪0", n_estimators=100, learning_rate=0.1)
#model.fit(X_train, y_train)
model.fit(X_train, y_train, eval_set=[(X_val, y_val)], verbose=True )
predictions = model.predict(X_val)
```

```
[0]     validation_0-rmse:68684.31221
[1]     validation_0-rmse:62995.89538
[2]     validation_0-rmse:58206.09089
[3]     validation_0-rmse:54271.73612
[4]     validation_0-rmse:50723.84845
[5]     validation_0-rmse:47808.51621
[6]     validation_0-rmse:45056.83878
[7]     validation_0-rmse:42577.47407
```

```
[8]     validation_0-rmse:40652.59334
[9]     validation_0-rmse:39126.49834
[10]    validation_0-rmse:37891.42521
[11]    validation_0-rmse:36896.50776
[12]    validation_0-rmse:36076.62386
[13]    validation_0-rmse:35395.18481
[14]    validation_0-rmse:34848.78556
[15]    validation_0-rmse:34152.21452
[16]    validation_0-rmse:33837.64290
[17]    validation_0-rmse:33572.76211
[18]    validation_0-rmse:33033.46378
[19]    validation_0-rmse:32612.00312
[20]    validation_0-rmse:32271.02453
[21]    validation_0-rmse:32054.84639
[22]    validation_0-rmse:31914.31598
[23]    validation_0-rmse:31828.21063
[24]    validation_0-rmse:31840.74158
[25]    validation_0-rmse:31839.54969
[26]    validation_0-rmse:31716.40642
[27]    validation_0-rmse:31650.06251
[28]    validation_0-rmse:31530.09822
[29]    validation_0-rmse:31427.41999
[30]    validation_0-rmse:31284.35632
[31]    validation_0-rmse:31179.29968
[32]    validation_0-rmse:31184.46847
[33]    validation_0-rmse:31239.38360
[34]    validation_0-rmse:31190.97815
[35]    validation_0-rmse:31237.89111
[36]    validation_0-rmse:31163.92462
[37]    validation_0-rmse:31083.87767
[38]    validation_0-rmse:31037.83531
[39]    validation_0-rmse:30979.96446
[40]    validation_0-rmse:30927.57264
[41]    validation_0-rmse:30957.78686
[42]    validation_0-rmse:30922.30906
[43]    validation_0-rmse:30903.60785
[44]    validation_0-rmse:30878.06547
[45]    validation_0-rmse:30852.94717
[46]    validation_0-rmse:30838.79172
[47]    validation_0-rmse:30838.98154
[48]    validation_0-rmse:30798.30704
[49]    validation_0-rmse:30817.55772
[50]    validation_0-rmse:30813.06819
[51]    validation_0-rmse:30783.22842
[52]    validation_0-rmse:30793.92719
[53]    validation_0-rmse:30771.29342
[54]    validation_0-rmse:30739.38864
[55]    validation_0-rmse:30736.36873
```

```
[56]    validation_0-rmse:30749.30402
[57]    validation_0-rmse:30739.15437
[58]    validation_0-rmse:30703.06945
[59]    validation_0-rmse:30686.29402
[60]    validation_0-rmse:30671.44024
[61]    validation_0-rmse:30666.02041
[62]    validation_0-rmse:30656.51729
[63]    validation_0-rmse:30636.45760
[64]    validation_0-rmse:30635.28754
[65]    validation_0-rmse:30645.69585
[66]    validation_0-rmse:30637.88009
[67]    validation_0-rmse:30633.04939
[68]    validation_0-rmse:30627.73530
[69]    validation_0-rmse:30619.63524
[70]    validation_0-rmse:30613.36051
[71]    validation_0-rmse:30620.90506
[72]    validation_0-rmse:30626.07842
[73]    validation_0-rmse:30621.37425
[74]    validation_0-rmse:30608.15705
[75]    validation_0-rmse:30631.17517
[76]    validation_0-rmse:30624.77105
[77]    validation_0-rmse:30627.69837
[78]    validation_0-rmse:30606.53869
[79]    validation_0-rmse:30607.53421
[80]    validation_0-rmse:30604.40765
[81]    validation_0-rmse:30601.08708
[82]    validation_0-rmse:30605.85965
[83]    validation_0-rmse:30613.89867
[84]    validation_0-rmse:30615.56423
[85]    validation_0-rmse:30605.71825
[86]    validation_0-rmse:30603.88630
[87]    validation_0-rmse:30607.22539
[88]    validation_0-rmse:30598.80346
[89]    validation_0-rmse:30601.50802
[90]    validation_0-rmse:30602.63994
[91]    validation_0-rmse:30606.44706
[92]    validation_0-rmse:30607.08710
[93]    validation_0-rmse:30604.86033
[94]    validation_0-rmse:30595.75642
[95]    validation_0-rmse:30592.68232
[96]    validation_0-rmse:30594.54893
[97]    validation_0-rmse:30598.76466
[98]    validation_0-rmse:30597.01759
[99]    validation_0-rmse:30596.35062

/home/hirstar/DataScience/DSEnv/lib/python3.12/site-
packages/xgboost/core.py:158: UserWarning: [14:45:54] WARNING:
/workspace/src/common/error_msg.cc:58: Falling back to prediction using DMatrix
```

```
due to mismatched devices. This might lead to higher memory usage and slower
performance. XGBoost is running on: cuda:0, while the input data is on: cpu.
Potential solutions:
- Use a data structure that matches the device ordinal in the booster.
- Set the device for booster before call to inplace_predict.

This warning will only be shown once.

  warnings.warn(smsg, UserWarning)
```

[39]:
```
#Test the accuracy score and what are the best parameters to use in XGBRegressor


param_grid = {
                'learning_rate':[0.01],
                'n_estimators':[100],
                'max_depth':[10],
                'subsample':[0.6],
                'colsample_bytree':[0.6],
                'reg_alpha':[0.1],
                'reg_lambda':[0.1]
                 }


grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train_gpu, y_train_gpu)

print('Best parameters:', grid_search.best_params_)
print('Best Score:', grid_search.best_score_)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
File cupy_backends/cuda/_softlink.pyx:25, in cupy_backends.cuda._softlink.
 ↪SoftLink.__init__()

File /usr/lib/python3.12/ctypes/__init__.py:379, in CDLL.__init__(self, name,␣
 ↪mode, handle, use_errno, use_last_error, winmode)
    378 if handle is None:
--> 379     self._handle = _dlopen(self._name, mode)
    380 else:

OSError: libnvrtc.so.11.2: cannot open shared object file: No such file or␣
 ↪directory

The above exception was the direct cause of the following exception:
```

```
RuntimeError                              Traceback (most recent call last)
Cell In[39], line 16
      4 param_grid = {
      5                 'learning_rate':[0.01],
      6                 'n_estimators':[100],
   (…)
     11                 'reg_lambda':[0.1]
     12                  }
     15 grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5
 ↪scoring='neg_mean_squared_error')
---> 16 grid_search.fit(X_train_gpu, y_train_gpu)
     18 print('Best parameters:', grid_search.best_params_)
     19 print('Best Score:', grid_search.best_score_)

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/base.py:1473, in
 ↪_fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1466     estimator._validate_params()
   1468 with config_context(
   1469     skip_parameter_validation=(
   1470         prefer_skip_nested_validation or global_skip_validation
   1471     )
   1472 ):
-> 1473     return fit_method(estimator, *args, **kwargs)

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/model_selection/
 ↪_search.py:1018, in BaseSearchCV.fit(self, X, y, **params)
   1012     results = self._format_results(
   1013         all_candidate_params, n_splits, all_out, all_more_results
   1014     )
   1016     return results
-> 1018 self._run_search(evaluate_candidates)
   1020 # multimetric is determined here because in the case of a callable
   1021 # self.scoring the return type is only known after calling
   1022 first_test_score = all_out[0]["test_scores"]

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/model_selection/
 ↪_search.py:1572, in GridSearchCV._run_search(self, evaluate_candidates)
   1570 def _run_search(self, evaluate_candidates):
   1571     """Search all candidates in param_grid"""
-> 1572     evaluate_candidates(ParameterGrid(self.param_grid))

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/model_selection/
 ↪_search.py:964, in BaseSearchCV.fit.<locals>.
 ↪evaluate_candidates(candidate_params, cv, more_results)
    956 if self.verbose > 0:
    957     print(
    958         "Fitting {0} folds for each of {1} candidates,"
    959         " totalling {2} fits".format(
```

```
    960             n_splits, n_candidates, n_candidates * n_splits
    961         )
    962     )
--> 964 out = parallel(
    965     delayed(_fit_and_score)(
    966         clone(base_estimator),
    967         X,
    968         y,
    969         train=train,
    970         test=test,
    971         parameters=parameters,
    972         split_progress=(split_idx, n_splits),
    973         candidate_progress=(cand_idx, n_candidates),
    974         **fit_and_score_kwargs,
    975     )
    976     for (cand_idx, parameters), (split_idx, (train, test)) in product(
    977         enumerate(candidate_params),
    978         enumerate(cv.split(X, y, **routed_params.splitter.split)),
    979     )
    980 )
    982 if len(out) < 1:
    983     raise ValueError(
    984         "No fits were performed. "
    985         "Was the CV iterator empty? "
    986         "Were there no candidates?"
    987     )

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/utils/parallel.py
 ↪74, in Parallel.__call__(self, iterable)
     69 config = get_config()
     70 iterable_with_config = (
     71     (_with_config(delayed_func, config), args, kwargs)
     72     for delayed_func, args, kwargs in iterable
     73 )
---> 74 return super().__call__(iterable_with_config)

File ~/DataScience/DSEnv/lib/python3.12/site-packages/joblib/parallel.py:1918,␣
 ↪in Parallel.__call__(self, iterable)
   1916     output = self._get_sequential_output(iterable)
   1917     next(output)
-> 1918     return output if self.return_generator else list(output)
   1920 # Let's create an ID that uniquely identifies the current call. If the
   1921 # call is interrupted early and that the same instance is immediately
   1922 # re-used, this id will be used to prevent workers that were
   1923 # concurrently finalizing a task from the previous call to run the
   1924 # callback.
   1925 with self._lock:
```

```
File ~/DataScience/DSEnv/lib/python3.12/site-packages/joblib/parallel.py:1847,␣
 ↪in Parallel._get_sequential_output(self, iterable)
   1845 self.n_dispatched_batches += 1
   1846 self.n_dispatched_tasks += 1
-> 1847 res = func(*args, **kwargs)
   1848 self.n_completed_tasks += 1
   1849 self.print_progress()

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/utils/parallel.py
 ↪136, in _FuncWrapper.__call__(self, *args, **kwargs)
    134     config = {}
    135 with config_context(**config):
--> 136     return self.function(*args, **kwargs)

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/model_selection/
 ↪_validation.py:880, in _fit_and_score(estimator, X, y, scorer, train, test,␣
 ↪verbose, parameters, fit_params, score_params, return_train_score,␣
 ↪return_parameters, return_n_test_samples, return_times, return_estimator,␣
 ↪split_progress, candidate_progress, error_score)
    876     estimator = estimator.set_params(**clone(parameters, safe=False))
    878 start_time = time.time()
--> 880 X_train, y_train = _safe_split(estimator, X, y, train)
    881 X_test, y_test = _safe_split(estimator, X, y, test, train)
    883 result = {}

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/utils/
 ↪metaestimators.py:156, in _safe_split(estimator, X, y, indices, train_indices
    154         X_subset = X[np.ix_(indices, train_indices)]
    155 else:
--> 156     X_subset = _safe_indexing(X, indices)
    158 if y is not None:
    159     y_subset = _safe_indexing(y, indices)

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/utils/_indexing.p :
 ↪267, in _safe_indexing(X, indices, axis)
    265     return _polars_indexing(X, indices, indices_dtype, axis=axis)
    266 elif hasattr(X, "shape"):
--> 267     return _array_indexing(X, indices, indices_dtype, axis=axis)
    268 else:
    269     return _list_indexing(X, indices, indices_dtype)

File ~/DataScience/DSEnv/lib/python3.12/site-packages/sklearn/utils/_indexing.p :
 ↪33, in _array_indexing(array, key, key_dtype, axis)
     31 if isinstance(key, tuple):
     32     key = list(key)
---> 33 return array[key, ...] if axis == 0 else array[:, key]

File cupy/_core/core.pyx:1527, in cupy._core.core._ndarray_base.__getitem__()
```

```
File cupy/_core/_routines_indexing.pyx:43, in cupy._core._routines_indexing.
 ↪_ndarray_getitem()

File cupy/_core/core.pyx:835, in cupy._core.core._ndarray_base.take()

File cupy/_core/_routines_indexing.pyx:132, in cupy._core._routines_indexing.
 ↪_ndarray_take()

File cupy/_core/_routines_indexing.pyx:826, in cupy._core._routines_indexing.
 ↪_take()

File cupy/_core/_kernel.pyx:920, in cupy._core._kernel.ElementwiseKernel.
 ↪__call__()

File cupy/_core/_kernel.pyx:945, in cupy._core._kernel.ElementwiseKernel.
 ↪_get_elementwise_kernel()

File cupy/_util.pyx:64, in cupy._util.memoize.decorator.ret()

File cupy/_core/_kernel.pyx:728, in cupy._core._kernel._get_elementwise_kernel(

File cupy/_core/_kernel.pyx:82, in cupy._core._kernel.
 ↪_get_simple_elementwise_kernel_from_code()

File cupy/_core/core.pyx:2281, in cupy._core.core.compile_with_cache()

File cupy/_core/core.pyx:2219, in cupy._core.core.
 ↪assemble_cupy_compiler_options()

File cupy_backends/cuda/libs/nvrtc.pyx:57, in cupy_backends.cuda.libs.nvrtc.
 ↪getVersion()

File cupy_backends/cuda/libs/_cnvrtc.pxi:72, in cupy_backends.cuda.libs.nvrtc.
 ↪initialize()

File cupy_backends/cuda/libs/_cnvrtc.pxi:76, in cupy_backends.cuda.libs.nvrtc.
 ↪_initialize()

File cupy_backends/cuda/libs/_cnvrtc.pxi:143, in cupy_backends.cuda.libs.nvrtc.
 ↪_get_softlink()

File cupy_backends/cuda/_softlink.pyx:32, in cupy_backends.cuda._softlink.
 ↪SoftLink.__init__()

RuntimeError: CuPy failed to load libnvrtc.so.11.2: OSError: libnvrtc.so.11.2:␣
 ↪cannot open shared object file: No such file or directory
```

[ ]:
```
#Plot the learning curve rate
train_sizes, train_scores, test_scores = learning_curve(
    XGBRegressor(**grid_search.best_params_), X_train, y_train, cv=5,␣
 ↪scoring='neg_mean_squared_error',
    train_sizes = [0.01, 0.1, 1], n_jobs=-1)


train_scores_mean = np.mean(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)

plt.plot(train_sizes, -train_scores_mean, label="Training Error")
plt.plot(train_sizes, -test_scores_mean, label="Validation Error")
plt.title("Learning rate Curve")
plt.xlabel('Training Set Size')
plt.ylabel('Error')
plt.legend()
plt.grid(True)
plt.show()
```

[ ]:
```
import xgboost as xgb
print(xgb.__version__)
print(xgb.XGBRegressor().get_params())
```

[ ]:
```
!nvcc --version
```

[ ]:
```
max_lines = (lambda X_train: X_train.shape[0])(X_train)
```

[ ]:
```
print (max_lines)
```