Installing the Dataset from Kaggle

```
! pip install -q kaggle

from google.colab import files

files.upload()
```

```
! mkdir ~/.kaggle

!cp kaggle.json ~/.kaggle/

! chmod 600 ~/.kaggle/kaggle.json

! kaggle datasets list
```

```
! kaggle competitions download -c house-prices-advanced-regression-techniques
```

```
! mkdir input
! unzip house-prices-advanced-regression-techniques.zip -d input
```

Import Dependencies

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

from xgboost import XGBRegressor
```

```
df = pd.read_csv('./input/train.csv')
```

Initial Inspection

```
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCo
0	1	60	RL	65.00	8450	Pave	NaN	Reg	
1	2	20	RL	80.00	9600	Pave	NaN	Reg	
2	3	60	RL	68.00	11250	Pave	NaN	IR1	
3	4	70	RL	60.00	9550	Pave	NaN	IR1	
4	5	60	RL	84.00	14260	Pave	NaN	IR1	

5 rows × 81 columns

```
df.info()
```

```
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64

 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
```

```
 67  OpenPorchSF     1460 non-null   int64
 68  EnclosedPorch   1460 non-null   int64
 69  3SsnPorch       1460 non-null   int64
 70  ScreenPorch     1460 non-null   int64
 71  PoolArea        1460 non-null   int64
 72  PoolQC          7 non-null      object
 73  Fence           281 non-null    object
 74  MiscFeature     54 non-null     object
 75  MiscVal         1460 non-null   int64
 76  MoSold          1460 non-null   int64
 77  YrSold          1460 non-null   int64
 78  SaleType        1460 non-null   object
 79  SaleCondition   1460 non-null   object
 80  SalePrice       1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
df.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Year
count	1,460.00	1,460.00	1,201.00	1,460.00	1,460.00	1,460.00	1,
mean	730.50	56.90	70.05	10,516.83	6.10	5.58	1,
std	421.61	42.30	24.28	9,981.26	1.38	1.11	
min	1.00	20.00	21.00	1,300.00	1.00	1.00	1,
25%	365.75	20.00	59.00	7,553.50	5.00	5.00	1,
50%	730.50	50.00	69.00	9,478.50	6.00	5.00	1,
75%	1,095.25	70.00	80.00	11,601.50	7.00	6.00	2,
max	1,460.00	190.00	313.00	215,245.00	10.00	9.00	2,

8 rows × 38 columns

Dealing with NaNs

```
df.isna().sum()
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                 ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
```

```
SalePrice          0
Length: 81, dtype: int64
```

```
df.columns[df.isnull().any()]
```

```
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Electrical', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence',
       'MiscFeature'],
      dtype='object')
```

```
percent_missing = df.isnull().sum() * 100 / len(df)
missing_value_df = pd.DataFrame({'column_name': df.columns,
                                 'percent_missing': percent_missing})

missing_value_df.sort_values('percent_missing', inplace = True, ascending = False)
missing_value_df
```

	column_name	percent_missing
PoolQC	PoolQC	99.52
MiscFeature	MiscFeature	96.30
Alley	Alley	93.77
Fence	Fence	80.75
FireplaceQu	FireplaceQu	47.26
...	...	...
ExterQual	ExterQual	0.00
Exterior2nd	Exterior2nd	0.00
Exterior1st	Exterior1st	0.00
RoofMatl	RoofMatl	0.00
SalePrice	SalePrice	0.00

81 rows × 2 columns

PoolQC, MiscFeature, Alley, and Fence columns all have missing values that go above 80%, hence, they will be dropped as a column.

```
drop_cols = ['PoolQC', 'MiscFeature', 'Alley', 'Fence']
df.drop(drop_cols, axis = 1, inplace = True)
```

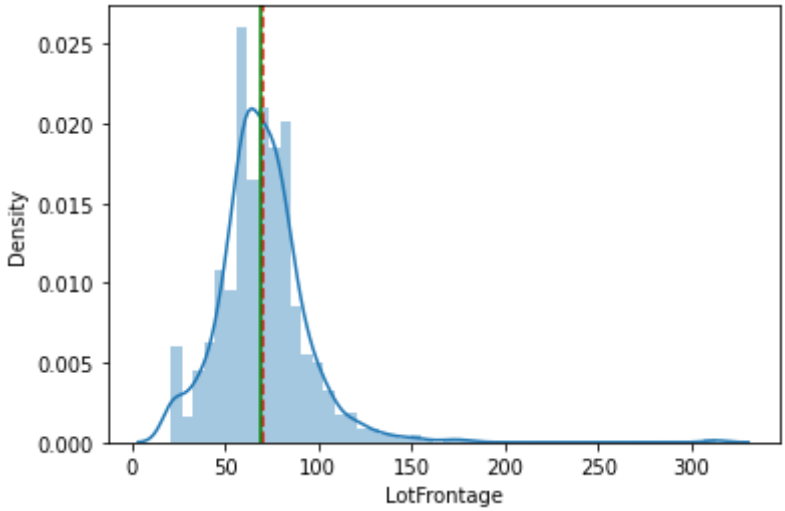
```
df.columns[df.isnull().any()]
```

```
Index(['LotFrontage', 'MasVnrType', 'MasVnrArea', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Electrical',
       'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish',
```

```
          'GarageQual', 'GarageCond'],
         dtype='object')
```

```
ax = sns.distplot(df['LotFrontage'])

ax.axvline(df['LotFrontage'].mean(), color = 'red', ls = '--')
ax.axvline(df['LotFrontage'].median(), color = 'green')

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
```



Data is skewed but not enough that the mean and median is far apart, hence, mean will be used as measure of central tendency which will be used to impute null values.

```
df['LotFrontage'].fillna(df['LotFrontage'].mean(), inplace = True)

df['LotFrontage'].isna().sum()
```

```
0
```

```
df['MasVnrType'].value_counts()
```

```
None        864
BrkFace     445
Stone       128
BrkCmn       15
Name: MasVnrType, dtype: int64
```

Since None is an available category, it can be deduced that the null values are also None, given that it's the most frequently occuring (mode). The mode will be used to impute null values.

```
df['MasVnrType'].fillna('None', inplace = True)
df['MasVnrType'].isna().sum()
```

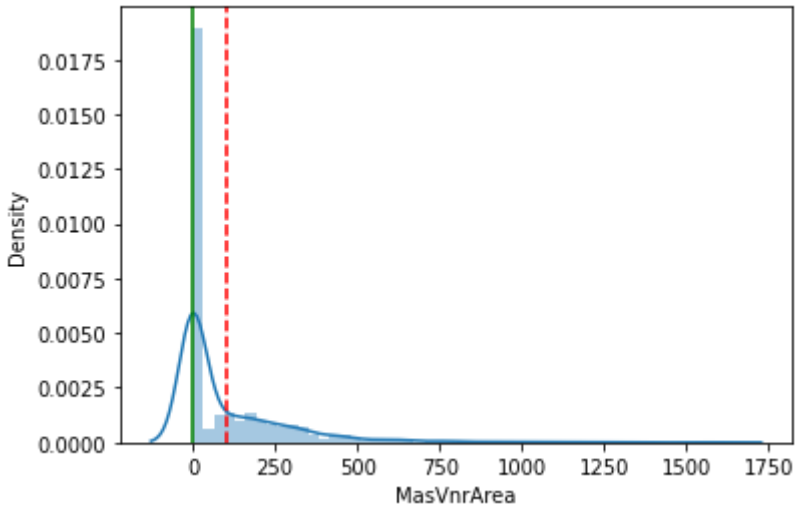
```
0
```

```
df['MasVnrArea'].value_counts().sort_values(ascending = False)
```

```
0.00      861
72.00       8
108.00      8
180.00      8
120.00      7
          ...
760.00      1
391.00      1
27.00       1
361.00      1
119.00      1
Name: MasVnrArea, Length: 327, dtype: int64
```

```
ax = sns.distplot(df['MasVnrArea'])

ax.axvline(df['MasVnrArea'].mean(), color = 'red', ls = '--')
ax.axvline(df['MasVnrArea'].median(), color = 'green')

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
```




```
print(df['MasVnrArea'].mean(), df['MasVnrArea'].median())
```

```
103.68526170798899 0.0
```

Data is skewed, hence, it would be better off to use the median as measure of central tendency to impute null values.

```
df['MasVnrArea'].fillna(0.0, inplace = True)
df['MasVnrArea'].isna().sum()
```

```
0
```

```
basement_cols = ['BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2']
df[basement_cols].isna().sum()
```

```
BsmtQual        37
BsmtCond        37
BsmtExposure    38
BsmtFinType1    37
BsmtFinType2    38
dtype: int64
```

```
df['BsmtQual'].unique()
```

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

It's understandable to impute null values in all the basement related columns with "NA" as not all houses are equipped with a basement.

```
for i in basement_cols:
  df[i].fillna('NA', inplace = True)

df[basement_cols].isna().sum()
```

```
BsmtQual        0
BsmtCond        0
BsmtExposure    0
BsmtFinType1    0
BsmtFinType2    0
dtype: int64
```

```
df['Electrical'].value_counts()
```

```
SBrkr    1334
FuseA      94
FuseF      27
FuseP       3
Mix         1
Name: Electrical, dtype: int64
```

There is only 1 null value in the column.

```
df['Electrical'].fillna('SBrkr', inplace = True)
df['Electrical'].isna().sum()
```

```
0
```

```
df['FireplaceQu'].value_counts()
```

```
Gd    380
TA    313
Fa     33
```

```
Ex     24
Po     20
Name: FireplaceQu, dtype: int64
```

```
df['FireplaceQu'].unique()
```

```
array([nan, 'TA', 'Gd', 'Fa', 'Ex', 'Po'], dtype=object)
```

As with the basement column, not all houses are equipped with a fireplace, hence, it's understandable to assume that the null values mean "NA".

```
df['FireplaceQu'].fillna('NA', inplace = True)
df['FireplaceQu'].isna().sum()
```

```
0
```

```
garage_cols = ['GarageType','GarageFinish', 'GarageQual', 'GarageCond']
df[garage_cols].isna().sum()
```

```
GarageType      81
GarageFinish    81
GarageQual      81
GarageCond      81
dtype: int64
```

Not all houses are also equipped with a garage, hence, they can be filled with "NA".

```
for i in garage_cols:
  df[i].fillna('NA', inplace = True)

df[garage_cols].isna().sum()
```

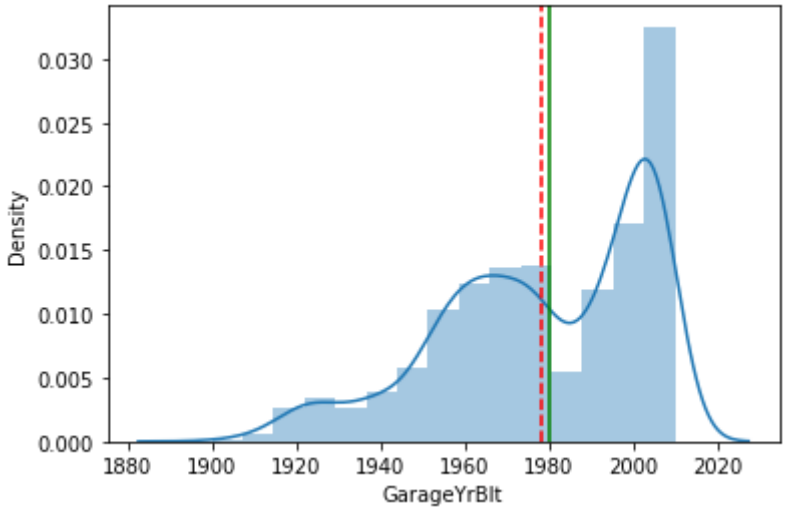
```
GarageType      0
GarageFinish    0
GarageQual      0
GarageCond      0
dtype: int64
```

```
df['GarageYrBlt'].dtype
```

```
dtype('float64')
```

```
ax = sns.distplot(df['GarageYrBlt'])

ax.axvline(df['GarageYrBlt'].mean(), color = 'red', ls = '--')
ax.axvline(df['GarageYrBlt'].median(), color = 'green')

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
```



Data is skewed, which is why median will be used for imputation.

```
df['GarageYrBlt'].fillna(df['GarageYrBlt'].median(), inplace = True)
df['GarageYrBlt'].isna().sum()
```

```
0
```

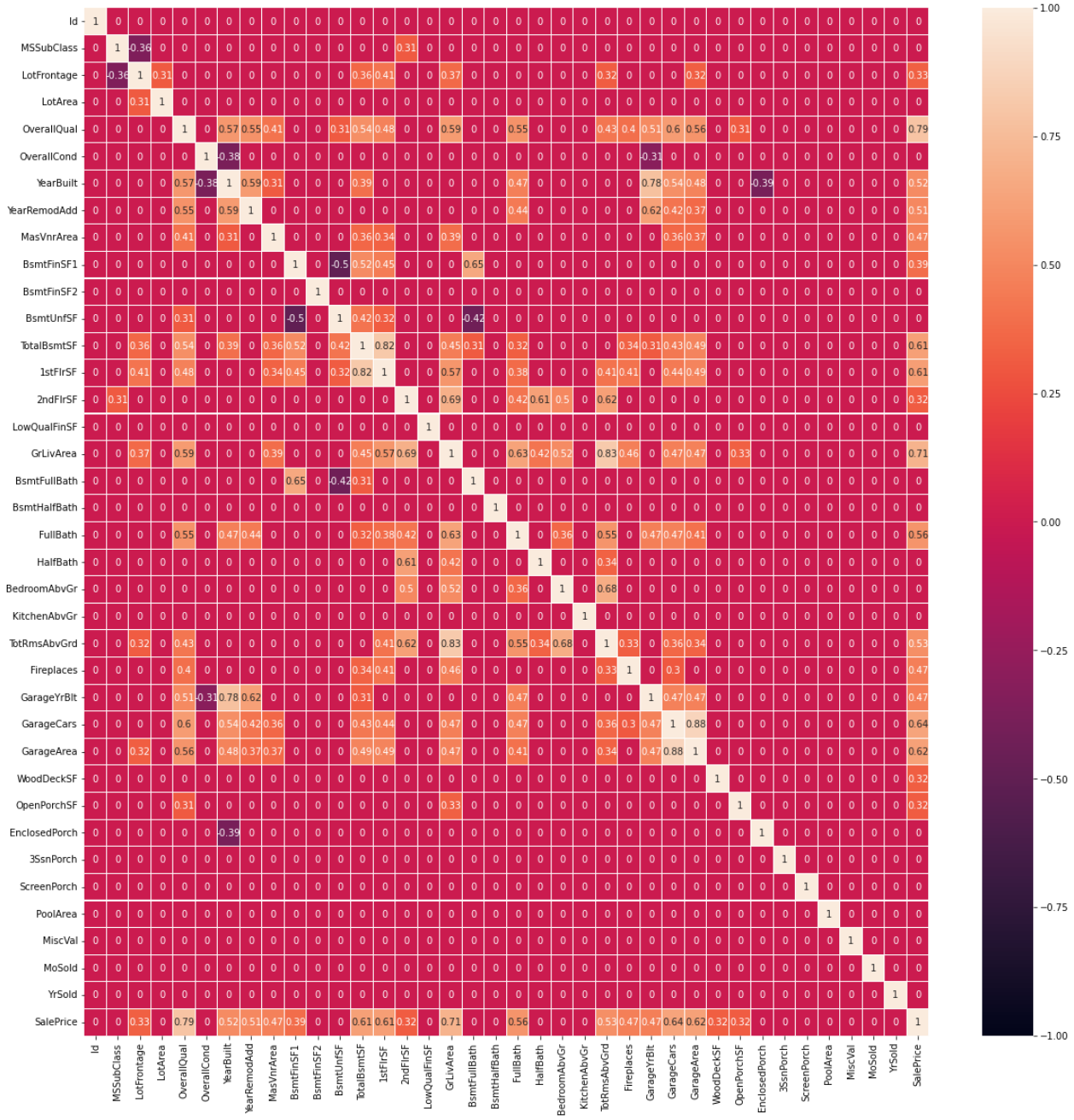
```
df.columns[df.isnull().any()]
```

```
Index([], dtype='object')
```

```
categorical_df = list(df.select_dtypes(include = [object]).columns)
numerical_df = list(df.drop(categorical_df, axis = 1).columns)

print(categorical_df, '\n')
print(numerical_df)
```

```
['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlop

['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBui
```

```
num_corr = df[numerical_df].corr()

num_corr[(num_corr < 0.3) & (num_corr > -0.3)] = 0
plt.figure(figsize = (20, 20))

sns.heatmap(num_corr, vmax = 1.0, vmin = -1.0, linewidths = 0.1, annot = True)

plt.show()
```


Id
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

```
df.corr()['SalePrice'].sort_values(ascending = False)[1:]
```

```
OverallQual      0.79
GrLivArea        0.71
GarageCars       0.64
GarageArea       0.62
TotalBsmtSF      0.61
1stFlrSF         0.61
FullBath         0.56
TotRmsAbvGrd     0.53
YearBuilt        0.52
YearRemodAdd     0.51
MasVnrArea       0.47
Fireplaces       0.47
GarageYrBlt      0.47
BsmtFinSF1       0.39
LotFrontage      0.33
WoodDeckSF       0.32
2ndFlrSF         0.32
OpenPorchSF      0.32
HalfBath         0.28
LotArea          0.26
BsmtFullBath     0.23
BsmtUnfSF        0.21
BedroomAbvGr     0.17
ScreenPorch      0.11
PoolArea         0.09
MoSold           0.05
3SsnPorch        0.04
BsmtFinSF2      -0.01
BsmtHalfBath    -0.02
MiscVal         -0.02
Id              -0.02
LowQualFinSF    -0.03
YrSold          -0.03
OverallCond     -0.08
MSSubClass      -0.08
EnclosedPorch   -0.13
KitchenAbvGr    -0.14
Name: SalePrice, dtype: float64
```

Features that have a correlation coefficient that's greater than 0.50 are considered.

Categorical Features Encoding

```
le = LabelEncoder()
```

```
for category in categorical_df:
  df[category] = le.fit_transform(df[category].astype(str))

df.info()
```

```
 21  RoofMatl       1460 non-null   int64
 22  Exterior1st    1460 non-null   int64
 23  Exterior2nd    1460 non-null   int64
 24  MasVnrType     1460 non-null   int64
 25  MasVnrArea     1460 non-null   float64
 26  ExterQual      1460 non-null   int64
 27  ExterCond      1460 non-null   int64
 28  Foundation     1460 non-null   int64
 29  BsmtQual       1460 non-null   int64
 30  BsmtCond       1460 non-null   int64
 31  BsmtExposure   1460 non-null   int64
 32  BsmtFinType1   1460 non-null   int64
 33  BsmtFinSF1     1460 non-null   int64
 34  BsmtFinType2   1460 non-null   int64
 35  BsmtFinSF2     1460 non-null   int64
 36  BsmtUnfSF      1460 non-null   int64
 37  TotalBsmtSF    1460 non-null   int64
 38  Heating        1460 non-null   int64
 39  HeatingQC      1460 non-null   int64
 40  CentralAir     1460 non-null   int64
 41  Electrical     1460 non-null   int64
 42  1stFlrSF       1460 non-null   int64
 43  2ndFlrSF       1460 non-null   int64
 44  LowQualFinSF   1460 non-null   int64
 45  GrLivArea      1460 non-null   int64
 46  BsmtFullBath   1460 non-null   int64
 47  BsmtHalfBath   1460 non-null   int64
 48  FullBath       1460 non-null   int64

 49  HalfBath       1460 non-null   int64
 50  BedroomAbvGr   1460 non-null   int64
 51  KitchenAbvGr   1460 non-null   int64
 52  KitchenQual    1460 non-null   int64
 53  TotRmsAbvGrd   1460 non-null   int64
 54  Functional     1460 non-null   int64
 55  Fireplaces     1460 non-null   int64
 56  FireplaceQu    1460 non-null   int64
 57  GarageType     1460 non-null   int64
 58  GarageYrBlt    1460 non-null   float64
 59  GarageFinish   1460 non-null   int64
 60  GarageCars     1460 non-null   int64
 61  GarageArea     1460 non-null   int64
 62  GarageQual     1460 non-null   int64
 63  GarageCond     1460 non-null   int64
 64  PavedDrive     1460 non-null   int64
 65  WoodDeckSF     1460 non-null   int64
 66  OpenPorchSF    1460 non-null   int64
 67  EnclosedPorch  1460 non-null   int64
 68  3SsnPorch      1460 non-null   int64
 69  ScreenPorch    1460 non-null   int64
 70  PoolArea       1460 non-null   int64
 71  MiscVal        1460 non-null   int64
 72  MoSold         1460 non-null   int64
 73  YrSold         1460 non-null   int64
 74  SaleType       1460 non-null   int64
 75  SaleCondition  1460 non-null   int64
```

```
 75  SaleCondition  1460 non-null   int64
 76  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(74)
memory usage: 878.4 KB
```

Categorical features have been converted to numerical, hence, dtypes that are "objects" have been eliminated completely.

Scaling Numerical Features

```
mms = MinMaxScaler()

X = df.drop(['Id', 'SalePrice'], axis = 1)
scaled_X = mms.fit_transform(X)

y = df['SalePrice']
```

```
X_train, X_test, y_train, y_test = train_test_split(scaled_X, y, test_size = 0.20, random_
```

Model Training

```
XGB = XGBRegressor()

XGB.fit(X_train, y_train)

y_pred = XGB.predict(X_test)
```

```
[14:06:09] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
```

Model Evaluation

```
print('XGBOOST TRAIN SCORE:', XGB.score(X_train, y_train))
print('XGBOOST TEST SCORE:', XGB.score(X_test, y_test))
```

```
XGBOOST TRAIN SCORE: 0.9691768626138606
XGBOOST TEST SCORE: 0.8429322288200407
```

Colab paid products - Cancel contracts here

✓ 0s completed at 10:06 PM