In [161…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy import stats
from scipy.stats import norm, skew
from collections import Counter
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error
import warnings
warnings.filterwarnings("ignore")
```

In [162…

```
df_train=pd.read_excel(r'D:\Project blog - datatraine\train.xlsx')
df_train
```

Out[162…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscF
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	

1168 rows × 81 columns

In [163…

```
df_test=pd.read_excel(r'D:\Project blog - datatraine\test.xlsx')
df_test
```

Out[163…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	...	0	0	NaN
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN

292 rows × 80 columns

In [164…

```
df_train.columns
```

Out[164…

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
```

```
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [197…

```
df_train.describe()
```

Out[197…

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF
count	1168.000000	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.00000
mean	724.136130	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.72602
std	416.159877	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.66478
min	1.000000	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.00000
25%	360.500000	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.00000
50%	714.500000	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.50000
75%	1079.500000	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.50000
max	1460.000000	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.00000

8 rows × 38 columns

In [199…

```
df_test.describe()
```

Out[199…

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
count	292.000000	292.000000	247.000000	292.000000	292.000000	292.000000	292.000000	292.000000	291.000000	292.00000
mean	755.955479	57.414384	66.425101	10645.143836	6.078767	5.493151	1972.616438	1985.294521	109.171821	439.29452
std	442.565228	43.780649	21.726343	13330.669795	1.356147	1.063267	30.447016	20.105792	175.030021	429.55967
min	6.000000	20.000000	21.000000	1526.000000	3.000000	3.000000	1872.000000	1950.000000	0.000000	0.00000
25%	377.750000	20.000000	53.500000	7200.000000	5.000000	5.000000	1954.000000	1968.000000	0.000000	0.00000
50%	778.000000	50.000000	65.000000	9200.000000	6.000000	5.000000	1976.000000	1994.000000	0.000000	369.50000
75%	1152.250000	70.000000	79.000000	11658.750000	7.000000	6.000000	2001.000000	2003.250000	180.000000	700.50000
max	1456.000000	190.000000	150.000000	215245.000000	10.000000	9.000000	2009.000000	2010.000000	1031.000000	1767.00000

8 rows × 38 columns

In [165…

```
train_x=df_train
test_x=df_test

train_Id=train_x['Id']
test_Id=test_x['Id']

train_x=train_x.drop(columns=['Id'])
test_x=test_x.drop(columns=['Id'])
```

In [166…

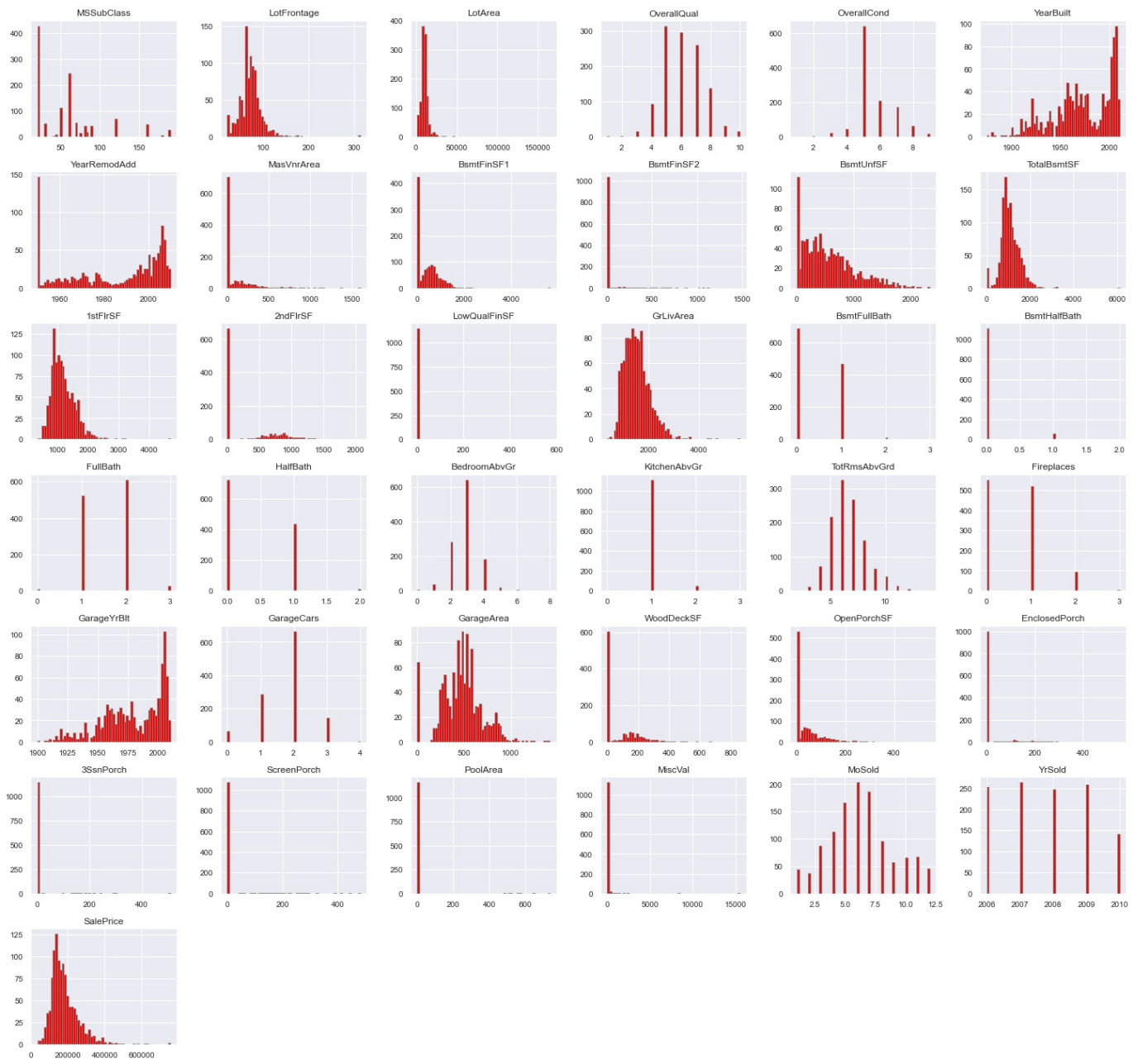
```
def check_na(data):
    nan_keys=[]
    for key in data.keys():
        for i in range(len(data[key].isna())):
            if data[key].isna()[i]:
                nan_keys.append(key)
                break
    return nan_keys
```

In [167…

```
def check_categorical(data):
    categorical_keys=[]
    for key in data.keys():
        if data[key].dtype==np.dtype('O'):
            categorical_keys.append(key)
    return categorical_keys
```

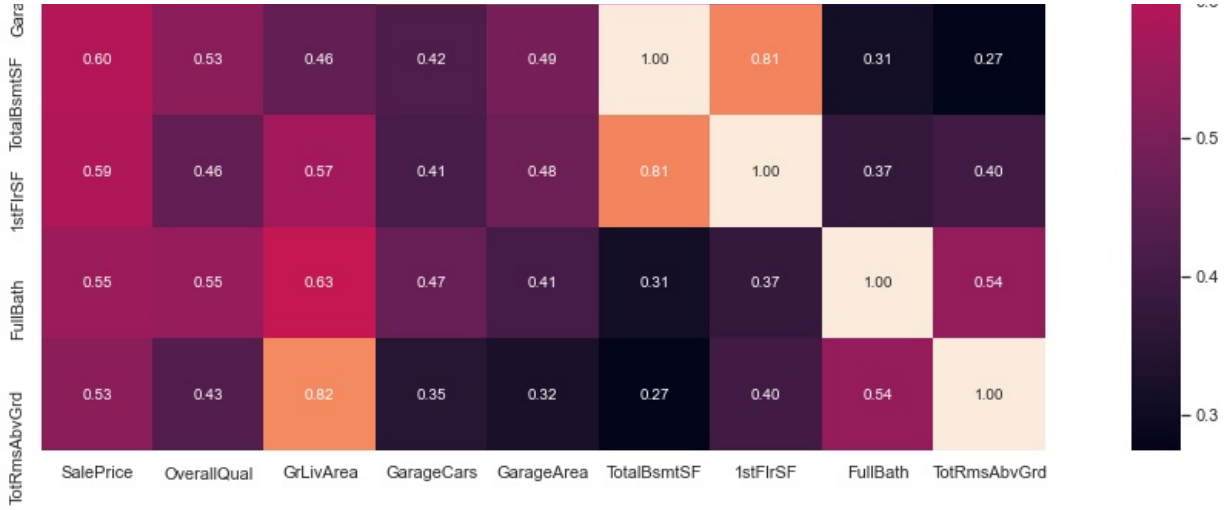
In [168…

```
_fig=train_x.hist(figsize=(25,24),bins=60,color='red',edgecolor='grey',xlabelsize=10, ylabelsize=10)
```



In [169…

```
corr_mat=train_x.corr()
k=9
cols=corr_mat.nlargest(k,'SalePrice')['SalePrice'].index
cm=np.corrcoef(train_x[cols].values.T)
sns.set(font_scale=1)
plt.subplots(figsize=(20,12))
heat_map=sns.heatmap(cm,cbar=True,annot=True, square=True,fmt='.2f',annot_kws={'size':10},yticklabels=cols.values
plt.show()
```



In [170…
```
train_x_numerical=train_x[cols]
```

In [171…
```
test_x_numerical=test_x[cols[1:]]
```

In [172…
```
categorical_train_keys=check_categorical(train_x)
nan_train_keys=check_na(train_x)
print(f"Nan Keys : {len(nan_train_keys)}\nCategorical keys: {len(categorical_train_keys)}")
print(categorical_train_keys)
```

```
Nan Keys : 18
Categorical keys: 43
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'Mas
VnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFi
nType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageT
ype', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'Sa
leCondition']
```

In [173…
```
categorical_test_keys=check_categorical(test_x)
nan_test_keys=check_na(test_x)
print(f"Nan Keys: {len(nan_test_keys)}\nCategorical keys: {len(categorical_test_keys)}")
```

```
Nan Keys: 19
Categorical keys: 42
```

In [174…
```
train_categoric=train_x[categorical_train_keys]
print(categorical_train_keys)
```

```
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'Mas
VnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFi
nType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageT
ype', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'Sa
leCondition']
```

In [177…
```
sns.countplot(x='SaleCondition', alpha=0.7,data=train_categoric)
```

Out[177… <AxesSubplot:xlabel='SaleCondition', ylabel='count'>

In [178…

```
categorical_features=['MSZoning','LotShape', 'LandContour', 'BldgType', 'HouseStyle', 'MasVnrType', 'ExterQual',
print(f"Len of categorical features:{len(categorical_features)}")
```

```
Len of categorical features:15
```

In [179…

```
train_categoric=train_categoric[categorical_features]
test_categoric=test_x[categorical_features]
print(train_categoric.shape, test_categoric.shape)
```

```
(1168, 15) (292, 15)
```

In [180…

```
train_x=pd.concat([train_x_numerical,train_categoric],axis=1)
test_x=pd.concat([test_x_numerical,test_categoric],axis=1)
print(train_x.shape,test_x.shape)
train_x.head()
```

```
(1168, 24) (292, 23)
```

Out[180…

	SalePrice	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	MSZoning	...	MasVnrType	E
0	128000	6	958	2	440	1078	958	2	5	RL	...	None	
1	268000	8	2217	2	621	2217	2217	2	8	RL	...	None	
2	269790	7	2013	2	455	1117	1127	2	8	RL	...	None	
3	190000	6	1844	2	546	1844	1844	2	7	RL	...	BrkFace	
4	215000	6	1602	2	529	1602	1602	2	8	RL	...	Stone	

5 rows × 24 columns

In [181…

```
for key in train_x.keys():
    if train_x[key].dtype==np.dtype('O'):
        train_x=train_x.fillna(value=train_x[key].value_counts().index[0])
    else:
        train_x=train_x.fillna(value=train_x[key].mean())
        print("Worked")
train_nan_keys=check_na(train_x)
print(train_nan_keys)
```

```
Worked
Worked
Worked
Worked
Worked
Worked
Worked
Worked
Worked
[]
```

In [182…

```
for key in test_x:
    if test_x[key].dtype==np.dtype('O'):
        test_x=test_x.fillna(value=test_x[key].value_counts().index[0])
    else:
        test_x=test_x.fillna(value=test_x[key].mean())
        print("Worked")
test_nan_keys=check_na(test_x)
print(test_nan_keys)
```

```
Worked
Worked
Worked
Worked
```

```
Worked
Worked
Worked
Worked
Worked
[]
```

In [183…

```
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
for key in categorical_features:
    transformed_data=label_encoder.fit_transform(train_x[key].values.astype("str").ravel())
    train_x=train_x.drop(columns=[key])
    train_x=pd.concat([train_x,pd.DataFrame(transformed_data,columns=[key])],axis=1)

print(check_categorical(train_x))
train_x.head()
```

```
[]
```

Out[183…

	SalePrice	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	MSZoning	...	MasVnrType	E
0	128000	6	958	2	440	1078	958	2	5	3	...	3	
1	268000	8	2217	2	621	2217	2217	2	8	3	...	3	
2	269790	7	2013	2	455	1117	1127	2	8	3	...	3	
3	190000	6	1844	2	546	1844	1844	2	7	3	...	2	
4	215000	6	1602	2	529	1602	1602	2	8	3	...	4	

5 rows × 24 columns

In [184…

```
for key in categorical_features:
    transformed_data=label_encoder.fit_transform(train_x[key].values.astype("str").ravel())
    train_x=train_x.drop(columns=[key])
    train_x=pd.concat([train_x,pd.DataFrame(transformed_data,columns=[key])],axis=1)

print(check_categorical(train_x))
train_x.head()
```

```
[]
```

Out[184…

	SalePrice	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	MSZoning	...	MasVnrType	E
0	128000	6	958	2	440	1078	958	2	5	3	...	3	
1	268000	8	2217	2	621	2217	2217	2	8	3	...	3	
2	269790	7	2013	2	455	1117	1127	2	8	3	...	3	
3	190000	6	1844	2	546	1844	1844	2	7	3	...	2	
4	215000	6	1602	2	529	1602	1602	2	8	3	...	4	

5 rows × 24 columns

In [185…

```
SalePrice=train_x['SalePrice']
train_x=train_x.drop(columns=['SalePrice'])
train_y=pd.DataFrame(SalePrice,columns=['SalePrice'])
train_y
```

Out[185…

	SalePrice
0	128000
1	268000
2	269790
3	190000
4	215000
...	...
1163	122000
1164	108000
1165	148500

1166	40000
1167	183200

1168 rows × 1 columns

In [188…
```
train_x['MSZoning'] = pd.to_numeric(train_x['MSZoning'], errors='coerce')
train_x.head()
```

Out[188…

	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	MSZoning	LotShape	...	MasVnrType	E
0	6	958	2	440	1078	958	2	5	3	0	...	3	
1	8	2217	2	621	2217	2217	2	8	3	0	...	3	
2	7	2013	2	455	1117	1127	2	8	3	0	...	3	
3	6	1844	2	546	1844	1844	2	7	3	0	...	2	
4	6	1602	2	529	1602	1602	2	8	3	0	...	4	

5 rows × 23 columns

In [190…
```
test_x['MSZoning'] = pd.to_numeric(test_x['MSZoning'], errors='coerce')
test_x.head()
```

Out[190…

	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	MSZoning	LotShape	...	MasVnrType	E
0	9	1922	3	676	1922	1922	2	8	NaN	IR1	...	Stone	
1	8	1360	2	565	1220	1360	1	4	NaN	IR1	...	None	
2	8	1788	2	522	1753	1788	2	7	NaN	Reg	...	None	
3	7	1564	1	234	704	860	1	7	NaN	Reg	...	None	
4	6	1933	3	668	894	894	2	9	NaN	IR1	...	Stone	

5 rows × 23 columns

In [196…
```
test_x = pd.to_numeric(test_x, errors='coerce')
test_x.head()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-196-0d6549831883> in <module>
----> 1 test_x = pd.to_numeric(test_x, errors='coerce')
      2 test_x.head()

~\anaconda3\lib\site-packages\pandas\core\tools\numeric.py in to_numeric(arg, errors, downcast)
    139         values = np.array([arg], dtype="O")
    140     elif getattr(arg, "ndim", 1) > 1:
--> 141         raise TypeError("arg must be a list, tuple, 1-d array, or Series")
    142     else:
    143         values = arg

TypeError: arg must be a list, tuple, 1-d array, or Series
```

In [193…
```
from sklearn.preprocessing import StandardScaler
SS=StandardScaler()
scaled_train_x=SS.fit_transform(train_x)
scaler_test_x=SS.fit_transform(test_x)
scaled_train_x=pd.DataFrame(scaled_train_x,columns=train_x.columns)
scaled_test_x=pd.DataFrame(scaled_test_x,columns=test_x.columns)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-193-0ccb194ad455> in <module>
      2 SS=StandardScaler()
      3 scaled_train_x=SS.fit_transform(train_x)
----> 4 scaler_test_x=SS.fit_transform(test_x)
      5 scaled_train_x=pd.DataFrame(scaled_train_x,columns=train_x.columns)
      6 scaled_test_x=pd.DataFrame(scaled_test_x,columns=test_x.columns)

~\anaconda3\lib\site-packages\sklearn\base.py in fit_transform(self, X, y, **fit_params)
```

```
    697         if y is None:
    698             # fit method of arity 1 (unsupervised transformation)
--> 699             return self.fit(X, **fit_params).transform(X)
    700         else:
    701             # fit method of arity 2 (supervised transformation)

~\anaconda3\lib\site-packages\sklearn\preprocessing\_data.py in fit(self, X, y, sample_weight)
    728         # Reset internal state before fitting
    729         self._reset()
--> 730         return self.partial_fit(X, y, sample_weight)
    731
    732     def partial_fit(self, X, y=None, sample_weight=None):

~\anaconda3\lib\site-packages\sklearn\preprocessing\_data.py in partial_fit(self, X, y, sample_weight)
    764         """
    765         first_call = not hasattr(self, "n_samples_seen_")
--> 766         X = self._validate_data(X, accept_sparse=('csr', 'csc'),
    767                                 estimator=self, dtype=FLOAT_DTYPES,
    768                                 force_all_finite='allow-nan', reset=first_call)

~\anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately, **check_p
arams)
    419             out = X
    420         elif isinstance(y, str) and y == 'no_validation':
--> 421             X = check_array(X, **check_params)
    422             out = X
    423         else:

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_spars
e, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    614                     array = array.astype(dtype, casting="unsafe", copy=False)
    615                 else:
--> 616                     array = np.asarray(array, order=order, dtype=dtype)
    617             except ComplexWarning as complex_warning:
    618                 raise ValueError("Complex data not supported\n"

~\anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray(a, dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order, like=like)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

~\anaconda3\lib\site-packages\pandas\core\generic.py in __array__(self, dtype)
   1897
   1898     def __array__(self, dtype=None) -> np.ndarray:
-> 1899         return np.asarray(self._values, dtype=dtype)
   1900
   1901     def __array_wrap__(

~\anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray(a, dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order, like=like)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

ValueError: could not convert string to float: 'HLS'
```

In [194…

```
from sklearn.manifold import Isomap
iso=Isomap(n_neighbors=5,n_components=3)
iso_scaled_train_x=iso.fit_transform(scaled_train_x)
iso_scaled_test_x=iso.fit_transform(scaled_test_x)

iso_scaled_train_x=pd.DataFrame(iso_scaled_train_x,columns=['ISO1','ISO2','ISO3'])
iso_scaled_test_x=pd.DataFrame(iso_scaled_test_x,columns=['ISO1','ISO2','ISO3'])

print(iso_scaled_train_x)
print(iso_scaled_test_x)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-194-0115904cee43> in <module>
      2 iso=Isomap(n_neighbors=5,n_components=3)
```

```
      3 iso_scaled_train_x=iso.fit_transform(scaled_train_x)
----> 4 iso_scaled_test_x=iso.fit_transform(scaled_test_x)
      5
      6 iso_scaled_train_x=pd.DataFrame(iso_scaled_train_x,columns=['ISO1','ISO2','ISO3'])

NameError: name 'scaled_test_x' is not defined
```

In [195…

```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(scaled_train_x,train_y)

from sklearn.ensemble import GradientBoostingRegressor
GBR=GradientBoostingRegressor(random_state=42, loss='squared_error',n_estimators=100,learning_rate=0.1)
GBR.fit(scaled_train_x,train_y.values.ravel())
XGB=xgb.XGBRegressor(booster='gbtree',eta=0.05,max_depth=7,n_estimators=200,gamma=0.2,reg_lambda=1)
XGB.fit(scaled_train_x,train_y.values.ravel())
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-195-85d8fb0ec92a> in <module>
      5 from sklearn.ensemble import GradientBoostingRegressor
      6 GBR=GradientBoostingRegressor(random_state=42, loss='squared_error',n_estimators=100,learning_rate=0.1)
----> 7 GBR.fit(scaled_train_x,train_y.values.ravel())
      8 XGB=xgb.XGBRegressor(booster='gbtree',eta=0.05,max_depth=7,n_estimators=200,gamma=0.2,reg_lambda=1)
      9 XGB.fit(scaled_train_x,train_y.values.ravel())

~\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py in fit(self, X, y, sample_weight, monitor)
    446             X_val = y_val = sample_weight_val = None
    447
--> 448         self._check_params()
    449
    450         if not self._is_initialized():

~\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py in _check_params(self)
    237         if (self.loss not in self._SUPPORTED_LOSS
    238                 or self.loss not in _gb_losses.LOSS_FUNCTIONS):
--> 239             raise ValueError("Loss '{0:s}' not supported. ".format(self.loss))
    240
    241         if self.loss == 'deviance':

ValueError: Loss 'squared_error' not supported.
```

In [ ]:

In [ ]: