Name:- Parshwa Shah

Experiment No.:- 5

Roll No.- 34

UID:- 2019230071

Batch:- B

Kaggle:- https://www.kaggle.com/parshwa52

Aim:- To measure the performance of the model

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import warnings
warnings.filterwarnings('ignore')

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets pre
# You can also write temporary files to /kaggle/temp/, but they won't be saved outsi
```

Import the training dataset

In [2]:
```
dataset = pd.read_csv('./train.csv')
```

Import the testing dataset

In [3]:
```
testdata = pd.read_csv('./test.csv')
```

In [4]:
```
dataset
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl
...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl

1460 rows × 81 columns

Check the dataset columns

In [5]:
```
dataset.columns
```

Out[5]:
```
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')
```

Check the columns with null values

In [6]:
```
for col in dataset.columns:
    if(dataset[col].isna().sum()>0):
        #print("Column=",col)
        print(f"Null values in {col} are {dataset[col].isna().sum()}")
```

```
Null values in LotFrontage are 259
Null values in Alley are 1369
Null values in MasVnrType are 8
Null values in MasVnrArea are 8
Null values in BsmtQual are 37
Null values in BsmtCond are 37
Null values in BsmtExposure are 38
Null values in BsmtFinType1 are 37
Null values in BsmtFinType2 are 38
Null values in Electrical are 1
Null values in FireplaceQu are 690
Null values in GarageType are 81
Null values in GarageYrBlt are 81
```

```
Null values in GarageFinish are 81
Null values in GarageQual are 81
Null values in GarageCond are 81
Null values in PoolQC are 1453
Null values in Fence are 1179
Null values in MiscFeature are 1406
```

Check the total no. of rows

In [7]:
```
totalrows=len(dataset)
```

In [8]:
```
totalrows
```

Out[8]:
```
1460
```

Check data type value counts

In [9]:
```
dataset.dtypes.value_counts()
```

Out[9]:
```
object     43
int64      35
float64     3
dtype: int64
```

Check for categorical and numerical columns in training data

In [10]:
```
cat_columns=[]
num_columns=[]
for col in dataset.columns.values:
    if dataset[col].dtype=='object':
        cat_columns.append(col)
    else:
        num_columns.append(col)
print(len(cat_columns)," Categorical Columns are \n",cat_columns,'\n')
print(len(num_columns),"Numeric columns are \n",num_columns)

cat_data=dataset[cat_columns]
num_data=dataset[num_columns]
```

```
43  Categorical Columns are
 ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfi
g', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyl
e', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQua
l', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType
1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQua
l', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'Garage
Cond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']

38 Numeric columns are
 ['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearB
uilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'Total
BsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtH
alfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'F
ireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSol
d', 'SalePrice']
```

Remove numerical columns whose Nan values > 40% and replace columns rest with median values in train data

In [11]:
```
print("Data Size Before Numerical NAN Column(>40%) Removal :",num_data.shape)
for col in num_data.columns.values:
    if (pd.isna(num_data[col]).sum())>0:
```

```
        if pd.isna(num_data[col]).sum() > (40/100*len(num_data)):
            print(col,"removed")
            num_data=num_data.drop([col], axis=1)
        else:
            num_data[col]=num_data[col].fillna(num_data[col].median())
print("Data Size After Numerical NAN Column(>40%) Removal :",num_data.shape)
```

```
Data Size Before Numerical NAN Column(>40%) Removal : (1460, 38)
Data Size After Numerical NAN Column(>40%) Removal : (1460, 38)
```

Check for categorical and numerical columns in test data

In [12]:

```
#remove columns which have null values > 40%
test_cat_columns=[]
test_num_columns=[]
for col in testdata.columns.values:
    if testdata[col].dtype=='object':
        test_cat_columns.append(col)
    else:
        test_num_columns.append(col)
print(len(test_cat_columns)," Categorical Columns are \n",test_cat_columns,'\n')
print(len(test_num_columns),"Numeric columns are \n",test_num_columns)

test_cat_data=testdata[test_cat_columns]
test_num_data=testdata[test_num_columns]
```

```
43  Categorical Columns are
 ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfi
g', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyl
e', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQua
l', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType
1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQua
l', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'Garage
Cond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']

37 Numeric columns are
 ['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearB
uilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'Total
BsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtH
alfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'F
ireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSol
d']
```

Check for null values in test data

In [13]:

```
test_num_data.isna().sum()
```

Out[13]:

```
Id                0
MSSubClass        0
LotFrontage     227
LotArea           0
OverallQual       0
OverallCond       0
YearBuilt         0
YearRemodAdd      0
MasVnrArea       15
BsmtFinSF1        1
BsmtFinSF2        1
BsmtUnfSF         1
TotalBsmtSF       1
1stFlrSF          0
2ndFlrSF          0
LowQualFinSF      0
GrLivArea         0
BsmtFullBath      2
```

```
BsmtHalfBath       2
FullBath           0
HalfBath           0
BedroomAbvGr       0
KitchenAbvGr       0
TotRmsAbvGrd       0
Fireplaces         0
GarageYrBlt       78
GarageCars         1
GarageArea         1
WoodDeckSF         0
OpenPorchSF        0
EnclosedPorch      0
3SsnPorch          0
ScreenPorch        0
PoolArea           0
MiscVal            0
MoSold             0
YrSold             0
dtype: int64
```

Remove numerical columns whose Nan values > 40% and replace columns rest with median values in test data

In [14]:
```
print("Data Size Before Numerical NAN Column(>40%) Removal :",test_num_data.shape)
for col in test_num_data.columns.values:
    if (pd.isna(test_num_data[col]).sum())>0:
        if pd.isna(test_num_data[col]).sum() > (40/100*len(test_num_data)):
            print(col,"removed")
            test_num_data=test_num_data.drop([col], axis=1)
        else:
            test_num_data[col]=test_num_data[col].fillna(test_num_data[col].median()
print("Data Size After Numerical NAN Column(>40%) Removal :",test_num_data.shape)
```

```
Data Size Before Numerical NAN Column(>40%) Removal : (1459, 37)
Data Size After Numerical NAN Column(>40%) Removal : (1459, 37)
```

In [15]:
```
test_num_data.isna().sum()
```

Out[15]:
```
Id               0
MSSubClass       0
LotFrontage      0
LotArea          0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
MasVnrArea       0
BsmtFinSF1       0
BsmtFinSF2       0
BsmtUnfSF        0
TotalBsmtSF      0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
TotRmsAbvGrd     0
Fireplaces       0
GarageYrBlt      0
GarageCars       0
GarageArea       0
```

```
WoodDeckSF       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
MiscVal          0
MoSold           0
YrSold           0
dtype: int64
```

Remove categorical columns whose Nan values > 40% and replace columns rest with mode values in test data

In [16]:
```
print("Data Size Before Categorical NAN Column(>40%) Removal :",test_cat_data.shape)
for col in test_cat_data.columns.values:
    if (pd.isna(test_cat_data[col]).sum())>0:
        if pd.isna(test_cat_data[col]).sum() > (40/100*len(test_cat_data)):
            print(col,"removed")
            test_cat_data=test_cat_data.drop([col], axis=1)
        else:
            test_cat_data[col]=test_cat_data[col].fillna(test_cat_data[col].mode()[0
print("Data Size After Categorical NAN Column(>40%) Removal :",test_cat_data.shape)
```

```
Data Size Before Categorical NAN Column(>40%) Removal : (1459, 43)
Alley removed
FireplaceQu removed
PoolQC removed
Fence removed
MiscFeature removed
Data Size After Categorical NAN Column(>40%) Removal : (1459, 38)
```

In [17]:
```
test_cat_data
```

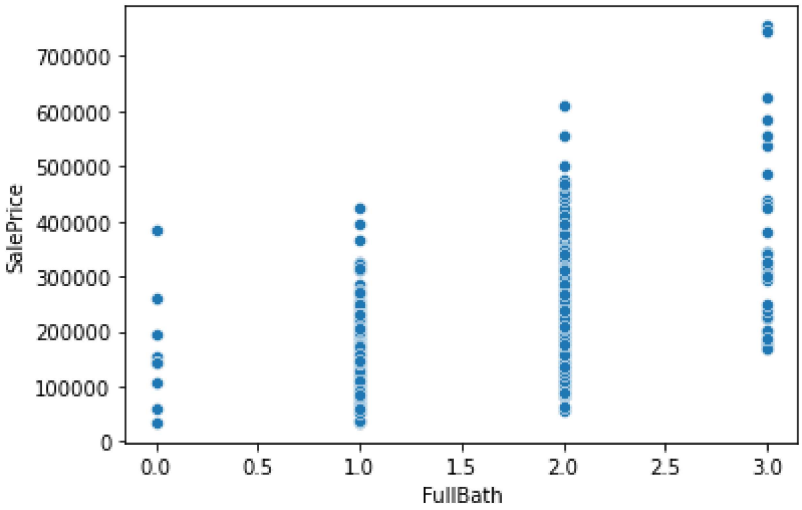
Out[17]:

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	C
0	RH	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	
1	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	NAmes	
2	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	
3	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	
4	RL	Pave	IR1	HLS	AllPub	Inside	Gtl	StoneBr	
...	...	...	...	...	...	...	...	...	
1454	RM	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	
1455	RM	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	
1456	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	
1457	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	
1458	RL	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	

1459 rows × 38 columns

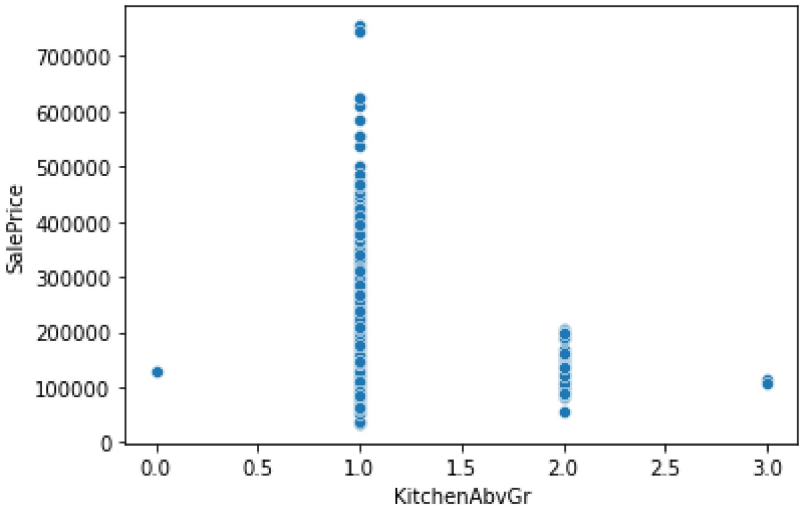
In [58]:
```
import seaborn as sns
sns.scatterplot(x="FullBath",
                y="SalePrice",
                data=num_data)
```

Out[58]: <AxesSubplot:xlabel='FullBath', ylabel='SalePrice'>



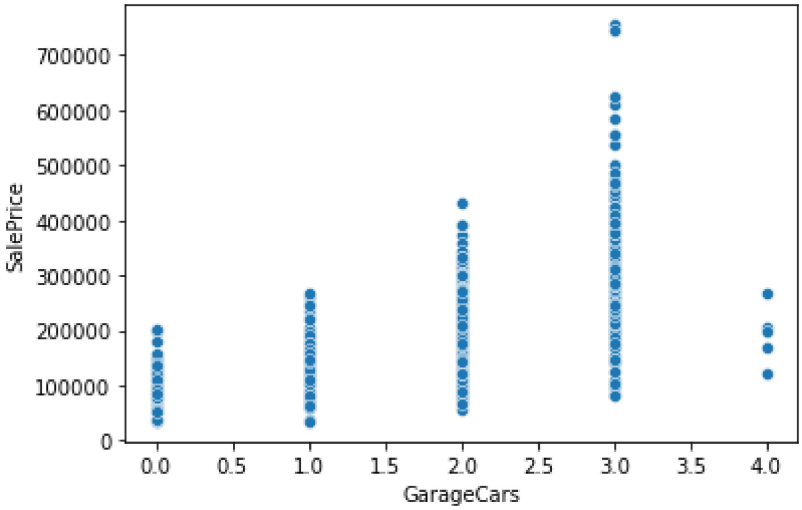
In [65]:
```
import seaborn as sns
sns.scatterplot(x="KitchenAbvGr",
                y="SalePrice",
                data=num_data)
```

Out[65]: <AxesSubplot:xlabel='KitchenAbvGr', ylabel='SalePrice'>



In [66]:
```
import seaborn as sns
sns.scatterplot(x="GarageCars",
                y="SalePrice",
                data=num_data)
```

Out[66]: <AxesSubplot:xlabel='GarageCars', ylabel='SalePrice'>



Inference:
1) More bathroom size, more is SalePrice
2) KitchenAbvGr is 1, then more is SalePrice
3) GarageCars more than 2, more is SalePrice

Create a baseline model using numerical columns of train data

In [18]:
```
y = num_data['SalePrice']
X = num_data.drop(['SalePrice'],axis=1)
```

In [19]:
```
X
```

Out[19]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdc
0	1	60	65.0	8450	7	5	2003	2003
1	2	20	80.0	9600	6	8	1976	1976
2	3	60	68.0	11250	7	5	2001	2002
3	4	70	60.0	9550	7	5	1915	1970
4	5	60	84.0	14260	8	5	2000	2000
...	...	...	...	...	...	...	...	...
1455	1456	60	62.0	7917	6	5	1999	2000
1456	1457	20	85.0	13175	6	6	1978	1988
1457	1458	70	66.0	9042	7	9	1941	2006
1458	1459	20	68.0	9717	5	6	1950	1996
1459	1460	20	75.0	9937	5	6	1965	1965

1460 rows × 37 columns

In [20]:
```
y
```

Out[20]:
```
0       208500
1       181500
2       223500
3       140000
4       250000
```

```
        ...
1455    175000
1456    210000
1457    266500
1458    142125
1459    147500
Name: SalePrice, Length: 1460, dtype: int64
```

Fit Linear Regression model as baseline model and note R2 score 0.81

In [21]:
```
from sklearn.linear_model import LinearRegression
basereg = LinearRegression().fit(X, y)
baseypred = basereg.predict(test_num_data)
basereg.score(X, y)
```

Out[21]:
```
0.8131858955487713
```

Referring Regression ROC AUC Score from: https://towardsdatascience.com/how-to-calculate-roc-auc-score-for-regression-models-c0be4fdf76bb

In [22]:
```
def regression_roc_auc_score(y_true, y_pred, num_rounds = 10):
  """
  Computes Regression-ROC-AUC-score.

  Parameters:
  ----------
  y_true: array-like of shape (n_samples,). Binary or continuous target variable.
  y_pred: array-like of shape (n_samples,). Target scores.
  num_rounds: int or string. If integer, number of random pairs of observations.
              If string, 'exact', all possible pairs of observations will be evaluat

  Returns:
  -------
  rroc: float. Regression-ROC-AUC-score.
  """

  import numpy as np

  y_true = np.array(y_true)
  y_pred = np.array(y_pred)

  num_pairs = 0
  num_same_sign = 0

  for i, j in _yield_pairs(y_true, num_rounds):
    diff_true = y_true[i] - y_true[j]
    diff_score = y_pred[i] - y_pred[j]
    if diff_true * diff_score > 0:
      num_same_sign += 1
    elif diff_score == 0:
      num_same_sign += .5
    num_pairs += 1

  return num_same_sign / num_pairs


def _yield_pairs(y_true, num_rounds):
  """
  Returns pairs of valid indices. Indices must belong to observations having differe

  Parameters:
  ----------
```

```
    y_true: array-like of shape (n_samples,). Binary or continuous target variable.
    num_rounds: int or string. If integer, number of random pairs of observations to r
                If string, 'exact', all possible pairs of observations will be returne

    Yields:
    -------
    i, j: tuple of int of shape (2,). Indices referred to a pair of samples.

    """
    import numpy as np

    if num_rounds == 'exact':
      for i in range(len(y_true)):
        for j in np.where((y_true != y_true[i]) & (np.arange(len(y_true)) > i))[0]:
          yield i, j
    else:
      for r in range(num_rounds):
        i = np.random.choice(range(len(y_true)))
        j = np.random.choice(np.where(y_true != y_true[i])[0])
        yield i, j
```

In [23]:
```
from sklearn.metrics import roc_auc_score
print("ROC AUC Score=",regression_roc_auc_score(y.to_numpy(),baseypred,10))
```

```
ROC AUC Score= 0.3
```

Fit Random Forest Regression model as baseline model and note R2 score 0.70

In [24]:
```
#baseline model on numerical data
from sklearn.ensemble import RandomForestRegressor
baserf = RandomForestRegressor(max_depth=2, random_state=0)
baserf.fit(X, y)
baserf.score(X,y)
```

Out[24]:
```
0.7014494620244635
```

Remove categorical columns whose Nan values > 40% and replace columns rest with mode values in train data

In [25]:
```
print("Data Size Before Categorical NAN Column(>40%) Removal :",cat_data.shape)
for col in cat_data.columns.values:
    if (pd.isna(cat_data[col]).sum())>0:
        if pd.isna(cat_data[col]).sum() > (40/100*len(cat_data)):
            print(col,"removed")
            cat_data=cat_data.drop([col], axis=1)
        else:
            cat_data[col]=cat_data[col].fillna(cat_data[col].mode()[0])
print("Data Size After Categorical NAN Column(>40%) Removal :",cat_data.shape)
```

```
Data Size Before Categorical NAN Column(>40%) Removal : (1460, 43)
Alley removed
FireplaceQu removed
PoolQC removed
Fence removed
MiscFeature removed
Data Size After Categorical NAN Column(>40%) Removal : (1460, 38)
```

Check null values in numeric and categorical data

In [26]:
```
num_data
```

Out[26]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdc
0	1	60	65.0	8450	7	5	2003	2003
1	2	20	80.0	9600	6	8	1976	1976
2	3	60	68.0	11250	7	5	2001	2002
3	4	70	60.0	9550	7	5	1915	1970
4	5	60	84.0	14260	8	5	2000	2000
...	...	...	...	...	...	...	...	...
1455	1456	60	62.0	7917	6	5	1999	2000
1456	1457	20	85.0	13175	6	6	1978	1988
1457	1458	70	66.0	9042	7	9	1941	2006
1458	1459	20	68.0	9717	5	6	1950	1996
1459	1460	20	75.0	9937	5	6	1965	1965

1460 rows × 38 columns

In [27]:
```
num_data.isna().sum()
```

Out[27]:
```
Id              0
MSSubClass      0
LotFrontage     0
LotArea         0
OverallQual     0
OverallCond     0
YearBuilt       0
YearRemodAdd    0
MasVnrArea      0
BsmtFinSF1      0
BsmtFinSF2      0
BsmtUnfSF       0
TotalBsmtSF     0
1stFlrSF        0
2ndFlrSF        0
LowQualFinSF    0
GrLivArea       0
BsmtFullBath    0
BsmtHalfBath    0
FullBath        0
HalfBath        0
BedroomAbvGr    0
KitchenAbvGr    0
TotRmsAbvGrd    0
Fireplaces      0
GarageYrBlt     0
GarageCars      0
GarageArea      0
WoodDeckSF      0
OpenPorchSF     0
EnclosedPorch   0
3SsnPorch       0
ScreenPorch     0
PoolArea        0
MiscVal         0
MoSold          0
YrSold          0
SalePrice       0
dtype: int64
```

In [28]:
```
cat_data
```

Out[28]:

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	C
0	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	
1	RL	Pave	Reg	Lvl	AllPub	FR2	Gtl	Veenker	
2	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	
3	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	
4	RL	Pave	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	
...	...	...	...	...	...	...	...	...	
1455	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Gilbert	
1456	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	
1457	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Crawfor	
1458	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	
1459	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Edwards	

1460 rows × 38 columns

In [29]:
```
cat_data.isna().sum()
```

Out[29]:
```
MSZoning         0
Street           0
LotShape         0
LandContour      0
Utilities        0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
ExterQual        0
ExterCond        0
Foundation       0
BsmtQual         0
BsmtCond         0
BsmtExposure     0
BsmtFinType1     0
BsmtFinType2     0
Heating          0
HeatingQC        0
CentralAir       0
Electrical       0
KitchenQual      0
Functional       0
GarageType       0
GarageFinish     0
GarageQual       0
GarageCond       0
PavedDrive       0
```

```
SaleType         0
SaleCondition    0
dtype: int64
```

Remove unnecessary columns from numeric data in train set

In [30]:
```
num_data=num_data.drop(['Id'], axis = 1)
```

In [31]:
```
num_data
```

Out[31]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	Mas
0	60	65.0	8450	7	5	2003	2003	
1	20	80.0	9600	6	8	1976	1976	
2	60	68.0	11250	7	5	2001	2002	
3	70	60.0	9550	7	5	1915	1970	
4	60	84.0	14260	8	5	2000	2000	
...	...	...	...	...	...	...	...	
1455	60	62.0	7917	6	5	1999	2000	
1456	20	85.0	13175	6	6	1978	1988	
1457	70	66.0	9042	7	9	1941	2006	
1458	20	68.0	9717	5	6	1950	1996	
1459	20	75.0	9937	5	6	1965	1965	

1460 rows × 37 columns

Derive newness column from yrsold and yearremodadd columns

In [32]:
```
num_data['newness'] = num_data['YrSold'] - num_data['YearRemodAdd']
```

In [33]:
```
num_data
```

Out[33]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	Mas
0	60	65.0	8450	7	5	2003	2003	
1	20	80.0	9600	6	8	1976	1976	
2	60	68.0	11250	7	5	2001	2002	
3	70	60.0	9550	7	5	1915	1970	
4	60	84.0	14260	8	5	2000	2000	
...	...	...	...	...	...	...	...	
1455	60	62.0	7917	6	5	1999	2000	
1456	20	85.0	13175	6	6	1978	1988	
1457	70	66.0	9042	7	9	1941	2006	
1458	20	68.0	9717	5	6	1950	1996	
1459	20	75.0	9937	5	6	1965	1965	

1460 rows × 38 columns

Remove year based columns from train data

In [34]:
```
num_data=num_data.drop(['YearBuilt','YearRemodAdd','YrSold'], axis = 1)
```

In [35]:
```
num_data
```

Out[35]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	Bsm
0	60	65.0	8450	7	5	196.0	706	
1	20	80.0	9600	6	8	0.0	978	
2	60	68.0	11250	7	5	162.0	486	
3	70	60.0	9550	7	5	0.0	216	
4	60	84.0	14260	8	5	350.0	655	
...	...	...	...	...	...	...	...	
1455	60	62.0	7917	6	5	0.0	0	
1456	20	85.0	13175	6	6	119.0	790	
1457	70	66.0	9042	7	9	0.0	275	
1458	20	68.0	9717	5	6	0.0	49	
1459	20	75.0	9937	5	6	0.0	830	

1460 rows × 35 columns

Label encode categorical data in train set

In [36]:
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

for col in cat_data.columns:
    label_encoder.fit(cat_data[col])
    cat_data[col]= label_encoder.transform(cat_data[col])
    test_cat_data[col] = label_encoder.transform(test_cat_data[col])
```

Create final train data containing numeric and categorical data

In [37]:
```
finaltraindata=pd.concat([num_data,cat_data],axis=1)
finaltraindata
```

Out[37]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	Bsm
0	60	65.0	8450	7	5	196.0	706	
1	20	80.0	9600	6	8	0.0	978	
2	60	68.0	11250	7	5	162.0	486	
3	70	60.0	9550	7	5	0.0	216	

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	Bsmt
4	60	84.0	14260	8	5	350.0	655	
...	...	...	...	...	...	...	...	
1455	60	62.0	7917	6	5	0.0	0	
1456	20	85.0	13175	6	6	119.0	790	
1457	70	66.0	9042	7	9	0.0	275	
1458	20	68.0	9717	5	6	0.0	49	
1459	20	75.0	9937	5	6	0.0	830	

1460 rows × 73 columns

Keep train data ready for model

In [38]:
```
yfinal = finaltraindata['SalePrice']
Xfinal = finaltraindata.drop(['SalePrice'],axis=1)
```

In [39]:
```
Xfinal
```

Out[39]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	Bsmt
0	60	65.0	8450	7	5	196.0	706	
1	20	80.0	9600	6	8	0.0	978	
2	60	68.0	11250	7	5	162.0	486	
3	70	60.0	9550	7	5	0.0	216	
4	60	84.0	14260	8	5	350.0	655	
...	...	...	...	...	...	...	...	
1455	60	62.0	7917	6	5	0.0	0	
1456	20	85.0	13175	6	6	119.0	790	
1457	70	66.0	9042	7	9	0.0	275	
1458	20	68.0	9717	5	6	0.0	49	
1459	20	75.0	9937	5	6	0.0	830	

1460 rows × 72 columns

In [40]:
```
yfinal
```

Out[40]:
```
0       208500
1       181500
2       223500
3       140000
4       250000
         ...
1455    175000
1456    210000
1457    266500
1458    142125
```

```
1459    147500
Name: SalePrice, Length: 1460, dtype: int64
```

Check test numeric and categorical data

In [41]:
```
test_num_data
```

Out[41]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdc
0	1461	20	80.0	11622	5	6	1961	1961
1	1462	20	81.0	14267	6	6	1958	1958
2	1463	60	74.0	13830	5	5	1997	1998
3	1464	60	78.0	9978	6	6	1998	1998
4	1465	120	43.0	5005	8	5	1992	1992
...	...	...	...	...	...	...	...	..
1454	2915	160	21.0	1936	4	7	1970	1970
1455	2916	160	21.0	1894	4	5	1970	1970
1456	2917	20	160.0	20000	5	7	1960	1996
1457	2918	85	62.0	10441	5	5	1992	1992
1458	2919	60	74.0	9627	7	5	1993	1994

1459 rows × 37 columns

In [42]:
```
test_cat_data
```

Out[42]:

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	C
0	2	1	3	3	0	4	0	12	
1	3	1	0	3	0	0	0	12	
2	3	1	0	3	0	4	0	8	
3	3	1	0	3	0	4	0	8	
4	3	1	0	1	0	4	0	22	
...	...	...	...	...	...	...	...	...	
1454	4	1	3	3	0	4	0	10	
1455	4	1	3	3	0	4	0	10	
1456	3	1	3	3	0	4	0	11	
1457	3	1	3	3	0	4	0	11	
1458	3	1	3	3	0	4	1	11	

1459 rows × 38 columns

Apply same preprocessing on test data as train data

In [43]:
```
testid = test_num_data['Id']
```

```
test_num_data=test_num_data.drop(['Id'], axis = 1)
```

In [44]:
```
testid
```

Out[44]:
```
0       1461
1       1462
2       1463
3       1464
4       1465
        ...
1454    2915
1455    2916
1456    2917
1457    2918
1458    2919
Name: Id, Length: 1459, dtype: int64
```

In [45]:
```
test_num_data['newness'] = test_num_data['YrSold'] - test_num_data['YearRemodAdd']
```

In [46]:
```
test_num_data=test_num_data.drop(['YearBuilt','YearRemodAdd','YrSold'], axis = 1)
```

In [47]:
```
finaltestdata=pd.concat([test_num_data,test_cat_data],axis=1)
finaltestdata
```

Out[47]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	Bsm
0	20	80.0	11622	5	6	0.0	468.0	
1	20	81.0	14267	6	6	108.0	923.0	
2	60	74.0	13830	5	5	0.0	791.0	
3	60	78.0	9978	6	6	20.0	602.0	
4	120	43.0	5005	8	5	0.0	263.0	
...	...	...	...	...	...	...	...	
1454	160	21.0	1936	4	7	0.0	0.0	
1455	160	21.0	1894	4	5	0.0	252.0	
1456	20	160.0	20000	5	7	0.0	1224.0	
1457	85	62.0	10441	5	5	0.0	337.0	
1458	60	74.0	9627	7	5	94.0	758.0	

1459 rows × 72 columns

Create Linear Regression as Model 1

In [48]:
```
from sklearn.linear_model import LinearRegression
model1reg = LinearRegression().fit(Xfinal, yfinal)
model1pred = model1reg.predict(finaltestdata)
model1reg.score(Xfinal, yfinal)
```

Out[48]:
```
0.848459088870739
```

Get results on test set for Linear Regression model and store results in csv file

In [49]:
```
model1pred
```

Out[49]:
```
array([106833.48805192, 154893.91847564, 169443.50344021, ...,
       143054.0772239 , 113413.10175559, 243788.01036936])
```

In [50]:
```
iddf = pd.DataFrame(testid, columns = ['Id'])
```

In [51]:
```
def convert_to_csv(modelpred,iddf,idn):
    model1df = pd.DataFrame(modelpred, columns = ['SalePrice'])
    frames = [iddf, model1df]
    result = pd.concat(frames,axis=1)
    result.to_csv(f'ytestres{idn}.csv',index=False)
```

In [52]:
```
convert_to_csv(model1pred,iddf,0)
```

Create Decision Tree Regression as Model 2

In [53]:
```
from sklearn.tree import DecisionTreeRegressor
model2rf = DecisionTreeRegressor()
model2rf.fit(Xfinal, yfinal)
model2pred = model2rf.predict(finaltestdata)
```

In [54]:
```
convert_to_csv(model2pred,iddf,1)
```

Conclusion:- Hence, in this experiment, I participated in Kaggle competition. I preprocessed the data and first created baseline model. Then, I encoded data and added some columns and finally after cleaning, I used two models, Linear Regression and Decision Tree Regression and submitted the results.