APPLIED DATA SCIENCE WITH PYTHON

FEATURE ENGINEERING REAL ESTATE ANALYTICS PROJECT

Task-1

Import the necessary libraries

Pandas is a Python library for data manipulation and analysis.

NumPy is a package that contains a multidimensional array object and several derivative ones.

Matplotlib is a Python visualization package for 2D array plots.Seaborn is built on top of Matplotlib. It's used for exploratory data analysis and data visualization.

CODE:

```
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn

import warnings
warnings.filterwarnings("ignore")
```

Task-2

2.0 Read the dataset
2.1 Understand the dataset
2.2 Print the name of the columns
2.3 Print the shape of the dataframe
2.4 Check for null values
2.5 Print the unique values
2.6 Select the numerical and categorical variables

CODE

```
#Fetching CSV file with the help of pandas library
df_houses = pd.read_csv("PEP1.csv",index_col=0)
df_houses.head()


#Checking number of rows and number columns with shape attribute
df_houses.shape
```

SCREENSHOTS

In [3]:
```
#Checking top 5 rows of dataframe
df_houses.head()
```

Out[3]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	PoolQC	Fence	MiscFeature	MiscV
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	NaN	NaN	NaN	
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	NaN	NaN	NaN	
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	NaN	NaN	NaN	

5 rows × 80 columns

In [4]:
```
#checking bottom 5 rows of a Dataframe
df_houses.tail()
```

Out[4]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	PoolQC	Fence	MiscFeature	Mi
1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	
1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	MnPrv	NaN	
1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	GdPrv	Shed	
1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	
1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	

5 rows × 80 columns

In [5]:
```
#Checking number of rows and number columns with shape attribute
df_houses.shape
```

Out[5]:
```
(1460, 80)
```

```
#Finding count of null values in each column
df_houses.isnull().sum()
```

In [7]:
```
#Finding count of null values in each column
df_houses.isnull().sum()
```

Out[7]:
```
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
Street             0
                 ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 80, dtype: int64
```

#Checking/Finding/Spotting for null values present in specific column and overall null columns count in entire dataframe

```
cnt=0
col_null_count = []
for x in df_houses.columns:
    if df_houses[x].isnull().any():
# Using .any() function to check if any element of a column is True
        cnt=cnt+1
        col_null_count.append(df_houses[x].isnull().sum())
 print(f"The {x} column contain:{df_houses[x].isnull().sum()} null values")
    else:
        continue

print("\n")
print('Total number of null valued columns present in dataframe are:',cnt)
```

```
print("\n")
print('Total nummber of null valued columns present in dataframe are:',cnt)
```

```
The LotFrontage column contain:    259 null values
The Alley column contain:    1369 null values
The MasVnrType column contain:    8 null values
The MasVnrArea column contain:    8 null values
The BsmtQual column contain:    37 null values
The BsmtCond column contain:    37 null values
The BsmtExposure column contain:    38 null values
The BsmtFinType1 column contain:    37 null values
The BsmtFinType2 column contain:    38 null values
The Electrical column contain:    1 null values
The FireplaceQu column contain:    690 null values
The GarageType column contain:    81 null values
The GarageYrBlt column contain:    81 null values
The GarageFinish column contain:    81 null values
The GarageQual column contain:    81 null values
The GarageCond column contain:    81 null values
The PoolQC column contain:    1453 null values
The Fence column contain:    1179 null values
The MiscFeature column contain:    1406 null values


Total nummber of null valued columns present in dataframe are: 19
```

```
In [9]: #Total number of columns present
```

#Filtering numeric columns from dataframe

```
numeric_columns = [col for col in df_houses.columns if
df_houses[col].dtype in ["float64", "int64"]]
print(numeric_columns)
print("\n")
print(type(numeric_columns))
```

In [13]:
```
#Filtering numeric columns from dataframe

numeric_columns = [col for col in df_houses.columns if df_houses[col].dtype in ["float64", "int64"]]
print(numeric_columns)
print("\n")
print(type(numeric_columns))
```

```
['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'Bs
mtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBa
th', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'Bedroom', 'Kitchen', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'Gara
geCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal
', 'MoSold', 'YrSold', 'SalePrice']


<class 'list'>
```

#Filtering categorical columns from dataframe using same logic as above

```
categorical_columns = [col for col in df_houses.columns if col not in
numeric_columns]
print(categorical_columns)
print("\n")
print(type(categorical_columns))
```

In [14]:
```
#Filtering categorical columns from dataframe using same logic as above

categorical_columns = [col for col in df_houses.columns if col not in numeric_columns]
print(categorical_columns)
print("\n")
print(type(categorical_columns))
```

```
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Co
ndition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType
', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'H
eating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFi
nish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']


<class 'list'>
```

```python
#Creating a dataframe with numerical and categorical variables
#Storing in new dataframe object

df_houses_new = pd.DataFrame([numeric_columns,categorical_columns])
Df_houses_new
```


In [17]:

```python
#Creating a dataframe with numerical and categorical variables
#Storing in new dataframe object

df_houses_new = pd.DataFrame([numeric_columns,categorical_columns])
df_houses_new
```


Out[17]:

	0	1	2	3	4	5	6	7	8	9	...	33	34	
0	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	MiscVal	MoSold	
1	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageType	GarageFinish	Ga

2 rows × 43 columns

```python
#Appliying unique funtion on variables

print("LotFrontage")
print(df_houses["LotFrontage"].unique())
print("\n")
print("MSSubClass")
print(df_houses["MSSubClass"].unique())
```


In [16]:

```python
#Appliying unique funtion on variables

print("LotFrontage")
print(df_houses["LotFrontage"].unique())
print("\n")
print("MSSubClass")
print(df_houses["MSSubClass"].unique())
```


```
LotFrontage
[ 65.  80.  68.  60.  84.  85.  75.  nan  51.  50.  70.  91.  72.  66.
 101.  57.  44. 110.  98.  47. 108. 112.  74. 115.  61.  48.  33.  52.
 100.  24.  89.  63.  76.  81.  95.  69.  21.  32.  78. 121. 122.  40.
 105.  73.  77.  64.  94.  34.  90.  55.  88.  82.  71. 120. 107.  92.
 134.  62.  86. 141.  97.  54.  41.  79. 174.  99.  67.  83.  43. 103.
  93.  30. 129. 140.  35.  37. 118.  87. 116. 150. 111.  49.  96.  59.
  36.  56. 102.  58.  38. 109. 130.  53. 137.  45. 106. 104.  42.  39.
 144. 114. 128. 149. 313. 168. 182. 138. 160. 152. 124. 153.  46.]


MSSubClass
[ 60  20  70  50 190  45  90 120  30  85  80 160  75 180  40]
```

TASK 3

EDA of numerical variables:
Missing value treatment
Identify the skewness and distribution
Identify significant variables using a correlation matrix
Pair plot for distribution and density

CODE:

Here we are dropping rows that have less than or equal to 30% missing values

```
df_houses1 =
df_houses.dropna(subset=["MasVnrType","MasVnrArea","BsmtQual","BsmtCond","
BsmtExposure","BsmtFinType2","Electrical",

"GarageType","GarageYrBlt","GarageFinish","GarageQual","GarageCond","LotFr
ontage"])
```

```
#Checking shape of dataframe after dropna() function
Df_houses1.shape
```

Here we are dropping rows that have less than or equal to 30% missing values

```
In [11]: df_houses1 = df_houses.dropna(subset=["MasVnrType","MasVnrArea","BsmtQual","BsmtCond","BsmtExposure","BsmtFinType2","E
lectrical",
                              "GarageType","GarageYrBlt","GarageFinish","GarageQual","GarageCond","LotFrontage"])
```

```
In [12]: #Checking shape of dataframe after dropna() function
df_houses1.shape

Out[12]: (1094, 80)
```

If Less than 30% ,drop the rows using dropna()funtion and if more than 30%, drop the variable/column itself by using drop() function

This step included EDA for both types - Numerical and categorical

```
#Doing EDA and Dropping variables that has 90% missing data

df_houses_after_EDA = df_houses1.drop(["Alley", "FireplaceQu", "PoolQC", "Fence", "MiscFeature"], axis=1)

df_houses_after_EDA.head()

df_houses_after_EDA.shape
```

In [18]:
```
#Doing EDA and Dropping variables that has 90% missing data
df_houses_after_EDA = df_houses1.drop(["Alley", "FireplaceQu", "PoolQC", "Fence", "MiscFeature"], axis=1)

df_houses_after_EDA.head()
```

Out[18]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	...	EnclosedPorch	3SsnPorch	ScreenPorch
1	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0	0	0
2	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub	FR2	Gtl	...	0	0	0
3	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub	Inside	Gtl	...	0	0	0
4	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub	Corner	Gtl	...	272	0	0
5	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub	FR2	Gtl	...	0	0	0

5 rows × 75 columns

In [19]:
```
df_houses_after_EDA.shape
```

Out[19]: (1094, 75)

```
#Displaying statistical summary of the dataframe

df_houses_after_EDA.describe()
```

In [21]:
```
#Displaying statistical summary of the dataframe

df_houses_after_EDA.describe()
```

Out[21]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeck
count	1094.000000	1094.000000	1094.000000	1094.000000	1094.000000	1094.000000	1094.000000	1094.000000	1094.000000	1094.000000	...	1094.0000
mean	56.128885	70.759598	10132.346435	6.247715	5.575868	1972.412249	1985.915905	109.855576	448.191956	45.252285	...	94.3418
std	41.976345	24.508859	8212.249621	1.366797	1.066500	31.189752	20.930772	190.667459	468.728095	159.075003	...	122.6246
min	20.000000	21.000000	1300.000000	2.000000	2.000000	1880.000000	1950.000000	0.000000	0.000000	0.000000	...	0.0000
25%	20.000000	60.000000	7606.750000	5.000000	5.000000	1953.000000	1967.000000	0.000000	0.000000	0.000000	...	0.0000
50%	50.000000	70.000000	9444.500000	6.000000	5.000000	1975.000000	1995.000000	0.000000	384.500000	0.000000	...	0.0000
75%	70.000000	80.000000	11387.250000	7.000000	6.000000	2003.000000	2005.000000	171.750000	712.750000	0.000000	...	169.7500
max	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	...	857.0000

A positively skewed (or right-skewed) distribution is a type of distribution in which most values are clustered around the left tail of the distribution while the right tail of the distribution is longer.

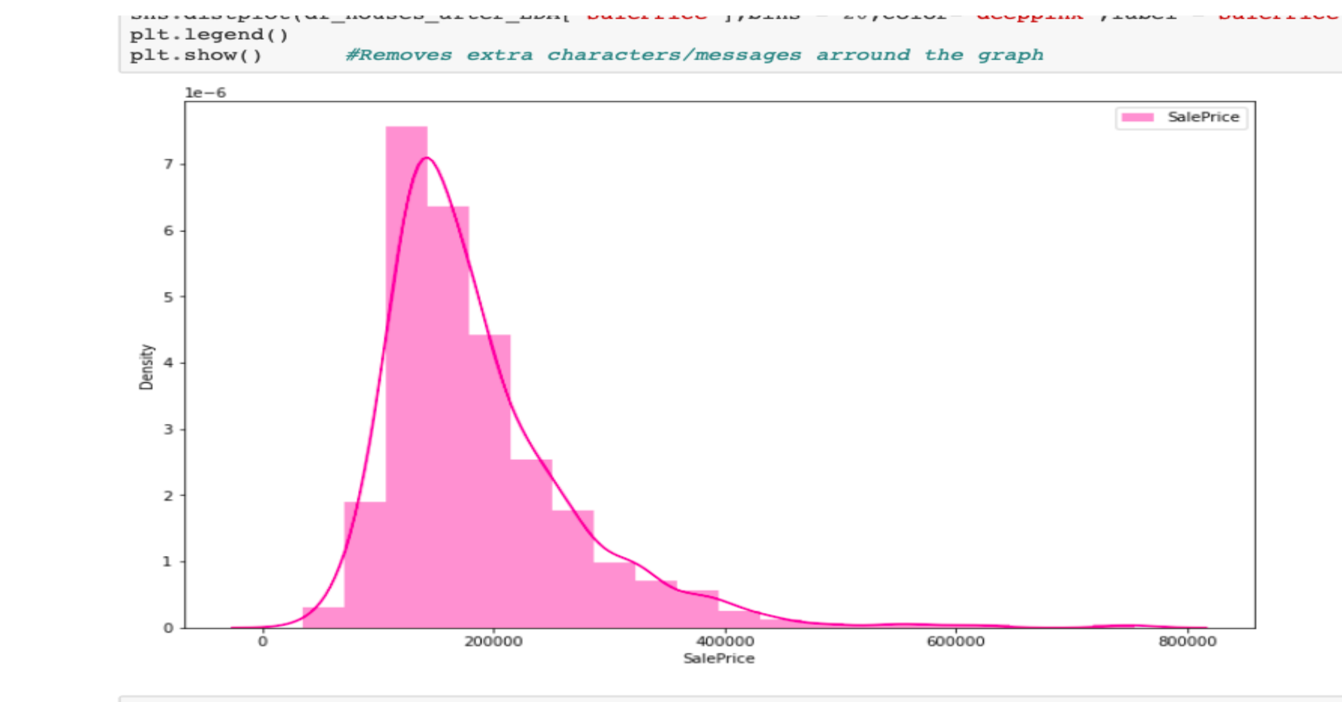
```
df_houses_after_EDA.head(1)

to_find_skewness = df_houses_after_EDA["SalePrice"]

to_find_skewness.skew()

# We see that it's a positive skewness, meaning right tailed
```

Output

```
1.9319910146053174
```

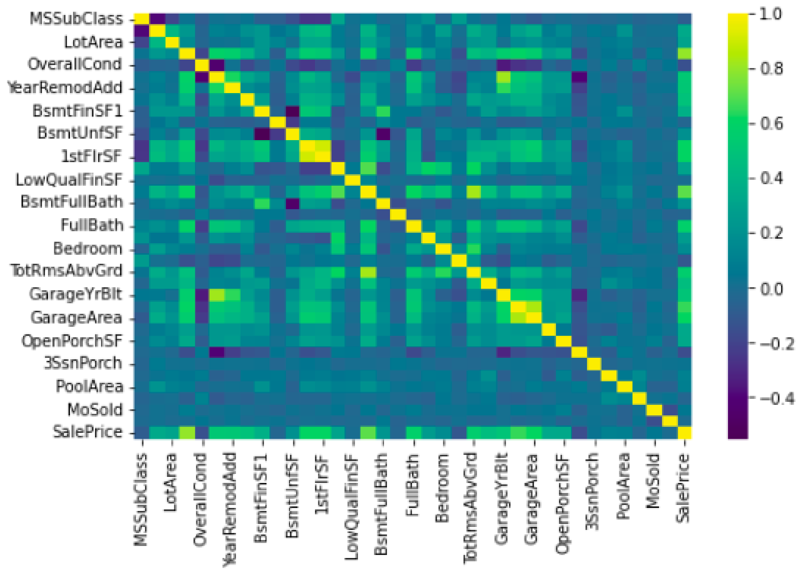
 # plotting a histogram to check the distribution of our DV "salesprice"

```
plt.figure(figsize =(12,8))
sns.distplot(df_houses_after_EDA['SalePrice'],bins = 20,color="deeppink",label ="SalePrice");
plt.legend()
plt.show()#Removes extra characters/messages around the graph
```



```
#We visualize the correlation matrix with the help of a heatmap:
##Brighter colors represent high activity/high correlation
areas,dull shades represent low correlation

plt.figure(figsize=(8,5))
corr_matrix = df_houses_after_EDA.corr()
#print(corr_matrix)
sns.heatmap(corr_matrix,cmap='viridis')
```

Out[24]: <AxesSubplot:>



A Lot of variables like lotfrontage,overallqual,yearbuilt have a high correlation with the salesprice which is good. (relation between the DV and IDV)

GrLivArea and Fireplaces are highly correlated (bright in color)(Relation between the IDVS)

```
#Pairplot

rock = df_houses_after_EDA._get_numeric_data()
rock = rock.iloc[:,np.r_[0:5,7,8:12,36]] #taking variables selectively as
per index
#rock.head()
import seaborn as sns
sns.pairplot(rock)
```

```
sns.pairplot(rock)
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x7febb65abc90>

TASK-4,5,6

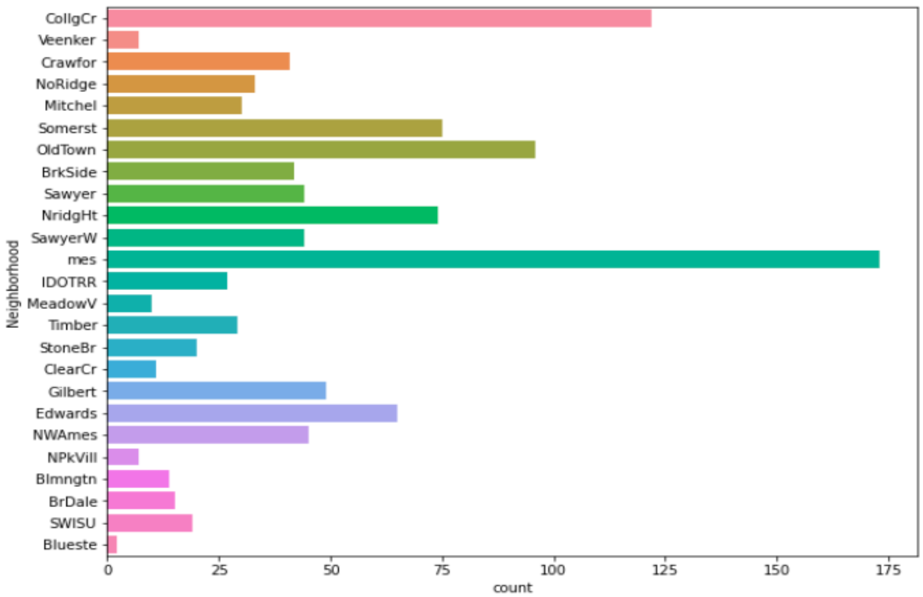
EDA of categorical variables
Missing value treatment
Count plot for bivariate analysis
Identify significant variables using p-values and Chi-Square values
Combine all the significant categorical and numerical variables
Plot box plot for the new dataset to find the variables with outliers

CODE:

```
#With seaborn library we call a histogram as a distplot and a bar graph as a countplot
# chosing next categorical variable for horizontal bars

plt.figure(figsize=(10,8))
sns.countplot(y='Neighborhood',data = df_houses_after_EDA)
```

```
plt.figure(figsize=(10,8))

sns.countplot(y='Neighborhood',data = df_houses_after_EDA)
```

26]: <AxesSubplot:xlabel='count', ylabel='Neighborhood'>

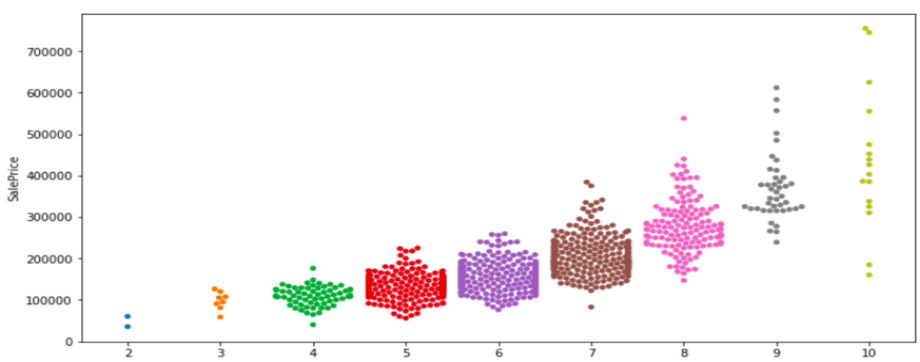


#swarmplot-type of scatter plot that is used for representing categorical values.

```
plt.figure(figsize=(12,6))
sns.swarmplot('OverallQual','SalePrice',data=df_houses_after_EDA)
```

```
plt.figure(figsize=(12,6))

sns.swarmplot('OverallQual','SalePrice',data=df_houses_after_EDA)
```

27]: <AxesSubplot:xlabel='OverallQual', ylabel='SalePrice'>



#There is a marked increase in the saleprice with an increase in the overall quality. So it is an important predictor for saleprice

CHI-SQUARED TEST

Testing different different independent variables with chi-squared test

```
from scipy.stats import chi2_contingency

table = df_houses_after_EDA.loc[:,["MSSubClass","LotArea"]]

stat, p, dof, expected = chi2_contingency(table)

P
```

<u>OUTPUT</u>

```
0.0

# p is less than alpha (0.05), hence we reject the Null which says there is no relation, and
#we conclude there is a statistical relationship
```

Numerical vs categorical

```
plt.figure(figsize=(15,8))
plt.xticks(rotation=90)
sns.boxplot("Neighborhood","SalePrice",data =df_houses_after_EDA)
```

```
sns.boxplot("Neighborhood","SalePrice",data =df_houses_after_EDA)
```

```
32]: <AxesSubplot:xlabel='Neighborhood', ylabel='SalePrice'>
```