***To predict house price***

***Here I used python as my programming language***

***To import dataset , I implemented these codes***

***Code:***

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

dataset=pd.read\_excel("House PricePrediction.xlsx")

# Printing first 5 records of the dataset

print(dataset.head(5))

After getting the data , second major phase is prediction of dimensions

So we use :

Code

dataset.shape

Now , I have categorized the entire data based on datatypes

obj = (dataset.dtypes == 'object')

object\_cols = list(obj[obj].index)

print("Category variables:",len(object\_cols))

 int\_ = (dataset.dtypes == 'int')

num\_cols = list(int\_[int\_].index)

print("Int variables:",len(num\_cols))

fl = (dataset.dtypes == 'float')

fl\_cols = list(fl[fl].index)

print("Float variables:",len(fl\_cols))

To make it more efficient I have undergone deep analysis to examine all types of variables .

Library used : seaborn

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

sns.heatmap(dataset.corr(),

            cmap = 'BrBG',

            fmt = '.2f',

            linewidths = 2,

            annot = True)

To classify into different categories. Let's draw bar plot

unique\_values = [ ]

for col in object\_cols:

  unique\_values.append(dataset[col].unique().size)

plt.figure(figsize=(10,6))

plt.title('Number Unique values of Categorical Features')

plt.xticks(rotation=90)

sns.barplot(x=object\_cols,y=unique\_values)

To find features within given Category we further categorize them :

plt.figure(figsize=(18, 36) )

plt.title('Category Features:Distribution')

plt.xticks(rotation=90)

index = 1

for col in object\_cols:

    y = dataset[col].value\_counts()

    plt.subplot(11, 4, index)

    plt.xticks(rotation=90)

    sns.barplot(x=list(y.index), y=y)

    index += 1

Let's proceed towards data cleaning, where corrupt and irrelevant data is removed

dataset.drop( [' Id '],

             axis=1,

             inplace=True)

dataset[' SalePrice '] = dataset[' SalePrice '].fillna(

  dataset[' SalePrice '].mean( ) )

new\_dataset = dataset.dropna( )

new\_dataset.isnull( ).sum( )

By using OneHotEncoder:

//collecting data having object datatype

from sklearn.preprocessing import OneHotEncoder

s = (new\_dataset.dtypes == 'object')

object\_cols = list(s[s].index)

print("Categorical variables:")

print(object\_cols)

print('No. of. categorical features: ',

      len(object\_cols) )

OHencoder=OneHotEncoder(sparse=False)

OH\_cols=pd.DataFrame(OHencoder.fit\_transform(new\_dataset[object\_cols ] ) )

OH\_cols.index = new\_dataset.index

OH\_cols.columns = OHencoder.get\_feature\_names( )

df\_final = new\_dataset.drop(object\_cols, axis=1)

df\_final = pd.concat([df\_final, OH\_cols], axis=1)

Now splitting dataset into training and testing

from sklearn.metrics import mean\_absolute\_error

from sklearn.model\_selection import train\_test\_split

X = df\_final.drop([' SalePrice '], axis=1)

Y = df\_final['SalePrice']

 # Splitting the training set into training and validation set

X\_train, X\_valid, Y\_train,

Y\_valid = traintest\_split(

    X,Y,train\_size=0.8, test\_size=0.2, random\_state=0)

***Finally using linear regression to predict outcome***

***Here we build a model and evaluate it***

from sklearn.linear\_model import LinearRegression

model\_LR = LinearRegression( )

model\_LR.fit(X\_train, Y\_train)

Y\_pred = model\_LR.predict(X\_valid)

print(mean\_absolute\_percentage\_error(Y\_valid, Y\_pred) )