In [1]:
```
import sklearn
print(sklearn.__version__)
```

```
1.5.1
```

In [15]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages
(from seaborn) (2.0.1)
Requirement already satisfied: pandas>=1.2 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from se
aborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packa
ges (from seaborn) (3.9.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fr
om matplotlib!=3.6.1,>=3.4->seaborn) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from m
atplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (f
rom matplotlib!=3.6.1,>=3.4->seaborn) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (f
rom matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fro
m matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from matp
lotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fr
om matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from p
andas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from
pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from pytho
n-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [16]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (3.9.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fr
om matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from m
atplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (f
rom matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (f
rom matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.23 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from ma
tplotlib) (2.0.1)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fro
m matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from matp
lotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fr
om matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages
(from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from pytho
n-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [19]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression
```

In [17]:

```python
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (1.5.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (2.0.1)
Requirement already satisfied: scipy>=1.6.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (3.5.0)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```
dtst = pd.read_csv("HousePricePrediction.csv")
dtst.head()
```

Out[20]:

	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	BsmtFinSF2	TotalBsmt
0	0	60	RL	8450	Inside	1Fam	5	2003	2003	VinylSd	0.0	85
1	1	20	RL	9600	FR2	1Fam	8	1976	1976	MetalSd	0.0	126
2	2	60	RL	11250	Inside	1Fam	5	2001	2002	VinylSd	0.0	92
3	3	70	RL	9550	Corner	1Fam	5	1915	1970	Wd Sdng	0.0	75
4	4	60	RL	14260	FR2	1Fam	5	2000	2000	VinylSd	0.0	114

In [21]:
```
dtst.shape
```

Out[21]: (2919, 13)

In [ ]:

In [ ]:
```
# 1. Data Preprocessing :
# Now, we categorize the features depending on their datatype (int, float, object)
# and then calculate the number of them.
```

In [17]:
```
obj = (dtst.dtypes=='object')
object_cols = list(obj[obj].index)
```

```
# print("categorial Variables : ", len(object_cols))
object_cols
```

Out[17]: ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']

In [22]:
```
obj = dtst.dtypes=='object'
object_cols = list(obj[obj].index)
print("categorical Variables : ", len(object_cols))
object_cols
```

```
categorical Variables :  4
```

Out[22]: ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']

In [23]:
```
int1 = dtst.dtypes=='int'
int_cols = list(int1[int1].index)
print("Integer Variables : ", len(int_cols))
int_cols
```

```
Integer Variables :  6
```

Out[23]: ['Id', 'MSSubClass', 'LotArea', 'OverallCond', 'YearBuilt', 'YearRemodAdd']

In [24]:
```
flt = dtst.dtypes=='float'
float_cols = list(flt[flt].index)
print("Float Variables : " , len(float_cols))
float_cols
```

```
Float Variables :  3
```

Out[24]: ['BsmtFinSF2', 'TotalBsmtSF', 'SalePrice']

In [1]:
```
# Exploratory Data Analysis :
# EDA refers to the deep analysis of data so as to discover different patterns
# and spot anomalies/outliers. Before making inferences from data it is essential to examine all your variables.

# So here let’s make a heatmap using seaborn library.
```

In [27]:
```
dtst = pd.read_csv("HousePricePrediction.csv")
dtst.head()

dtst1 = dtst.drop("LotConfig" , axis=1)
```

```
dtst2 = dtst1.drop("MSZoning", axis=1)
dtst3 = dtst2.drop("BldgType", axis=1)
dtst4 = dtst3.drop("Exterior1st", axis=1)
dtst4
dtst4.head()
```

Out[27]:

	Id	MSSubClass	LotArea	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF2	TotalBsmtSF	SalePrice
0	0	60	8450	5	2003	2003	0.0	856.0	208500.0
1	1	20	9600	8	1976	1976	0.0	1262.0	181500.0
2	2	60	11250	5	2001	2002	0.0	920.0	223500.0
3	3	70	9550	5	1915	1970	0.0	756.0	140000.0
4	4	60	14260	5	2000	2000	0.0	1145.0	250000.0

In [28]:
```
dtst4.dtypes
```

Out[28]:
```
Id               int64
MSSubClass       int64
LotArea          int64
OverallCond      int64
YearBuilt        int64
YearRemodAdd     int64
BsmtFinSF2     float64
TotalBsmtSF    float64
SalePrice      float64
dtype: object
```

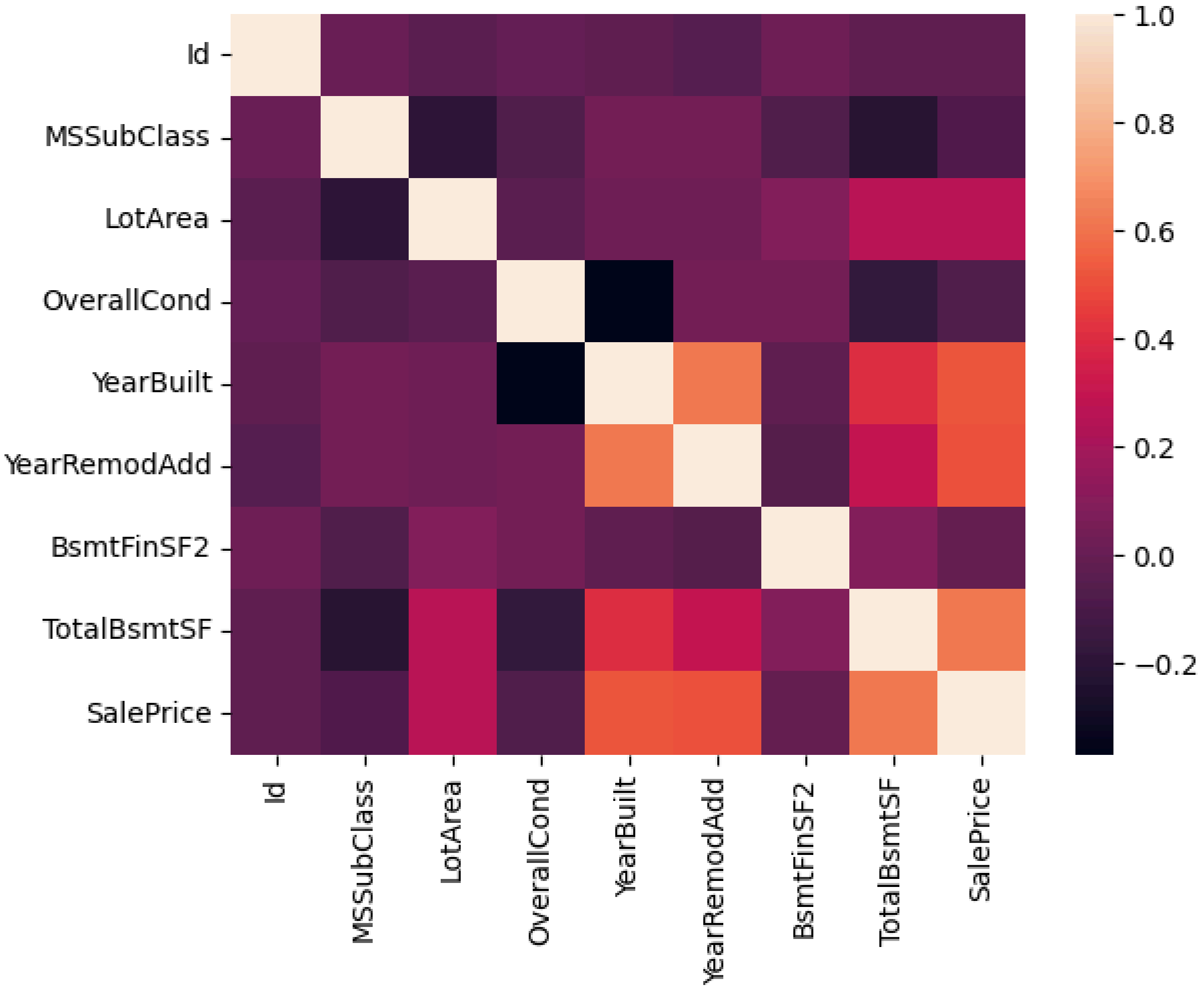
In [29]:
```
print(dtst4.corr())
```

```
                   Id  MSSubClass   LotArea  OverallCond  YearBuilt  \
Id           1.000000    0.008931 -0.040746    -0.002839  -0.016581
MSSubClass   0.008931    1.000000 -0.201730    -0.065625   0.034409
LotArea     -0.040746   -0.201730  1.000000    -0.035617   0.024128
OverallCond -0.002839   -0.065625 -0.035617     1.000000  -0.368477
YearBuilt   -0.016581    0.034409  0.024128    -0.368477   1.000000
YearRemodAdd -0.050438    0.043315  0.021612     0.047654   0.612235
BsmtFinSF2   0.018251   -0.072530  0.084059     0.041501  -0.027595
TotalBsmtSF -0.024924   -0.219965  0.254138    -0.174002   0.408515
SalePrice   -0.021917   -0.084284  0.263843    -0.077856   0.522897

             YearRemodAdd  BsmtFinSF2  TotalBsmtSF  SalePrice
Id              -0.050438    0.018251    -0.024924  -0.021917
MSSubClass       0.043315   -0.072530    -0.219965  -0.084284
LotArea          0.021612    0.084059     0.254138   0.263843
OverallCond      0.047654    0.041501    -0.174002  -0.077856
YearBuilt        0.612235   -0.027595     0.408515   0.522897
YearRemodAdd     1.000000   -0.062153     0.298107   0.507101
BsmtFinSF2      -0.062153    1.000000     0.089410  -0.011378
TotalBsmtSF      0.298107    0.089410     1.000000   0.613581
SalePrice        0.507101   -0.011378     0.613581   1.000000
```

In [30]:
```python
print(dtst4.corr())

sns.heatmap(dtst4.corr())
```

```
                   Id  MSSubClass   LotArea  OverallCond  YearBuilt  \
Id           1.000000    0.008931 -0.040746    -0.002839  -0.016581   
MSSubClass   0.008931    1.000000 -0.201730    -0.065625   0.034409   
LotArea     -0.040746   -0.201730  1.000000    -0.035617   0.024128   
OverallCond -0.002839   -0.065625 -0.035617     1.000000  -0.368477   
YearBuilt   -0.016581    0.034409  0.024128    -0.368477   1.000000   
YearRemodAdd -0.050438    0.043315  0.021612     0.047654   0.612235   
BsmtFinSF2   0.018251   -0.072530  0.084059     0.041501  -0.027595   
TotalBsmtSF -0.024924   -0.219965  0.254138    -0.174002   0.408515   
SalePrice   -0.021917   -0.084284  0.263843    -0.077856   0.522897   

             YearRemodAdd  BsmtFinSF2  TotalBsmtSF  SalePrice  
Id              -0.050438    0.018251    -0.024924  -0.021917  
MSSubClass       0.043315   -0.072530    -0.219965  -0.084284  
LotArea          0.021612    0.084059     0.254138   0.263843  
OverallCond      0.047654    0.041501    -0.174002  -0.077856  
YearBuilt        0.612235   -0.027595     0.408515   0.522897  
YearRemodAdd     1.000000   -0.062153     0.298107   0.507101  
BsmtFinSF2      -0.062153    1.000000     0.089410  -0.011378  
TotalBsmtSF      0.298107    0.089410     1.000000   0.613581  
SalePrice        0.507101   -0.011378     0.613581   1.000000  
```

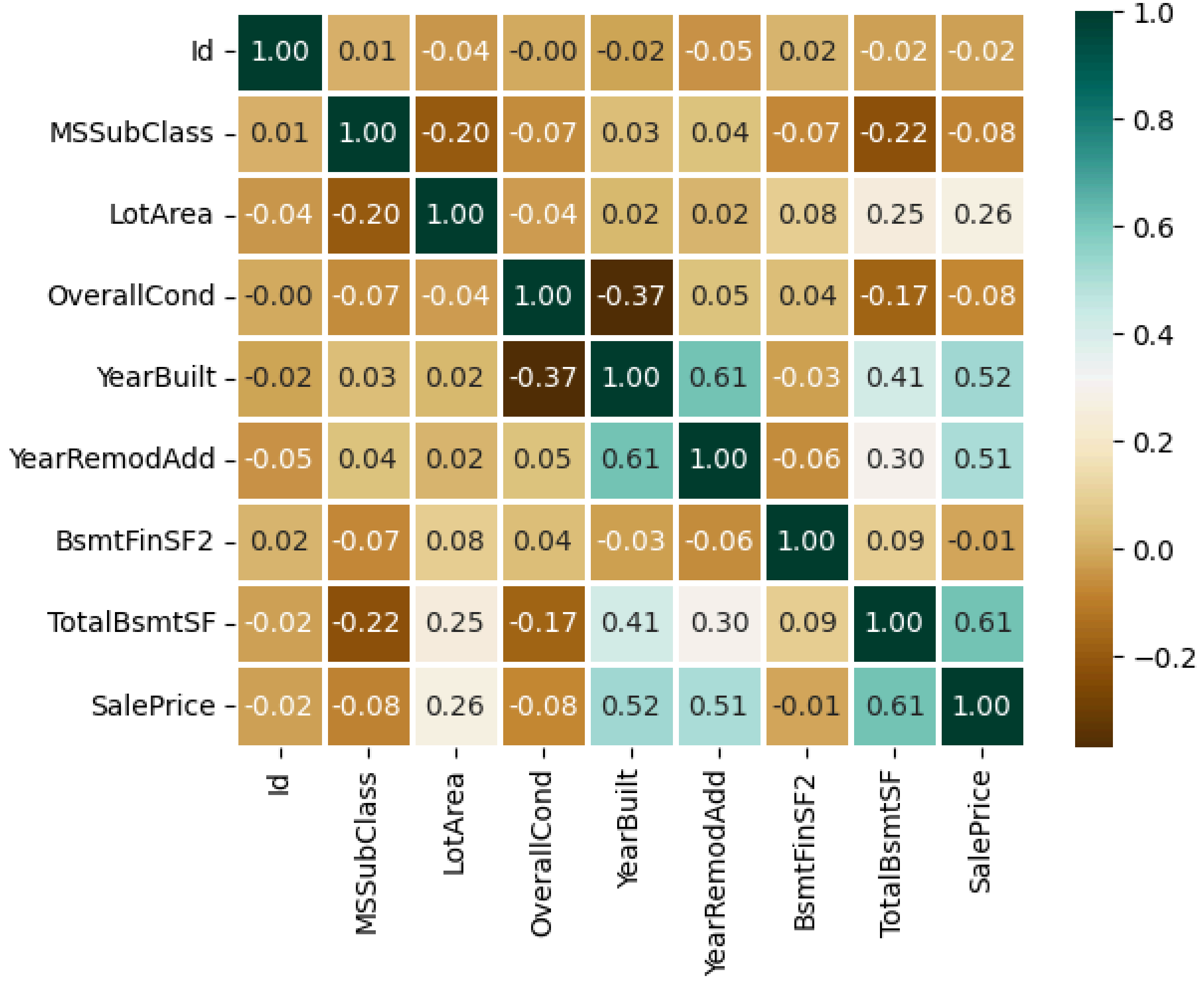
Out[30]: <Axes: >



In [31]:
```
print(dtst4.corr())
sns.heatmap(dtst4.corr(), cmap = 'BrBG', fmt = '.2f', linewidths = 2, annot = True)
```

```
                   Id  MSSubClass   LotArea  OverallCond  YearBuilt  \
Id           1.000000    0.008931 -0.040746    -0.002839  -0.016581
MSSubClass   0.008931    1.000000 -0.201730    -0.065625   0.034409
LotArea     -0.040746   -0.201730  1.000000    -0.035617   0.024128
OverallCond -0.002839   -0.065625 -0.035617     1.000000  -0.368477
YearBuilt   -0.016581    0.034409  0.024128    -0.368477   1.000000
YearRemodAdd -0.050438    0.043315  0.021612     0.047654   0.612235
BsmtFinSF2   0.018251   -0.072530  0.084059     0.041501  -0.027595
TotalBsmtSF -0.024924   -0.219965  0.254138    -0.174002   0.408515
SalePrice   -0.021917   -0.084284  0.263843    -0.077856   0.522897

             YearRemodAdd  BsmtFinSF2  TotalBsmtSF  SalePrice
Id              -0.050438    0.018251    -0.024924  -0.021917
MSSubClass       0.043315   -0.072530    -0.219965  -0.084284
LotArea          0.021612    0.084059     0.254138   0.263843
OverallCond      0.047654    0.041501    -0.174002  -0.077856
YearBuilt        0.612235   -0.027595     0.408515   0.522897
YearRemodAdd     1.000000   -0.062153     0.298107   0.507101
BsmtFinSF2      -0.062153    1.000000     0.089410  -0.011378
TotalBsmtSF      0.298107    0.089410     1.000000   0.613581
SalePrice        0.507101   -0.011378     0.613581   1.000000
```

Out[31]: <Axes: >



In [32]:
```
dtst.head()
```

Out[32]:

	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	BsmtFinSF2	TotalBsmt
0	0	60	RL	8450	Inside	1Fam	5	2003	2003	VinylSd	0.0	85
1	1	20	RL	9600	FR2	1Fam	8	1976	1976	MetalSd	0.0	126
2	2	60	RL	11250	Inside	1Fam	5	2001	2002	VinylSd	0.0	92
3	3	70	RL	9550	Corner	1Fam	5	1915	1970	Wd Sdng	0.0	75
4	4	60	RL	14260	FR2	1Fam	5	2000	2000	VinylSd	0.0	114

In [33]:

```
# Identifying categorical columns :
# The variable object_cols is expected to contain a list of column names that are categorical.
# (typically columns with string or object data types).

obj_cols = [col for col in dtst.columns
           if dtst[col].dtype == 'object']
print("Categorial coulmns :", obj_cols)
```

```
Categorial coulmns : ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']
```

In [34]:

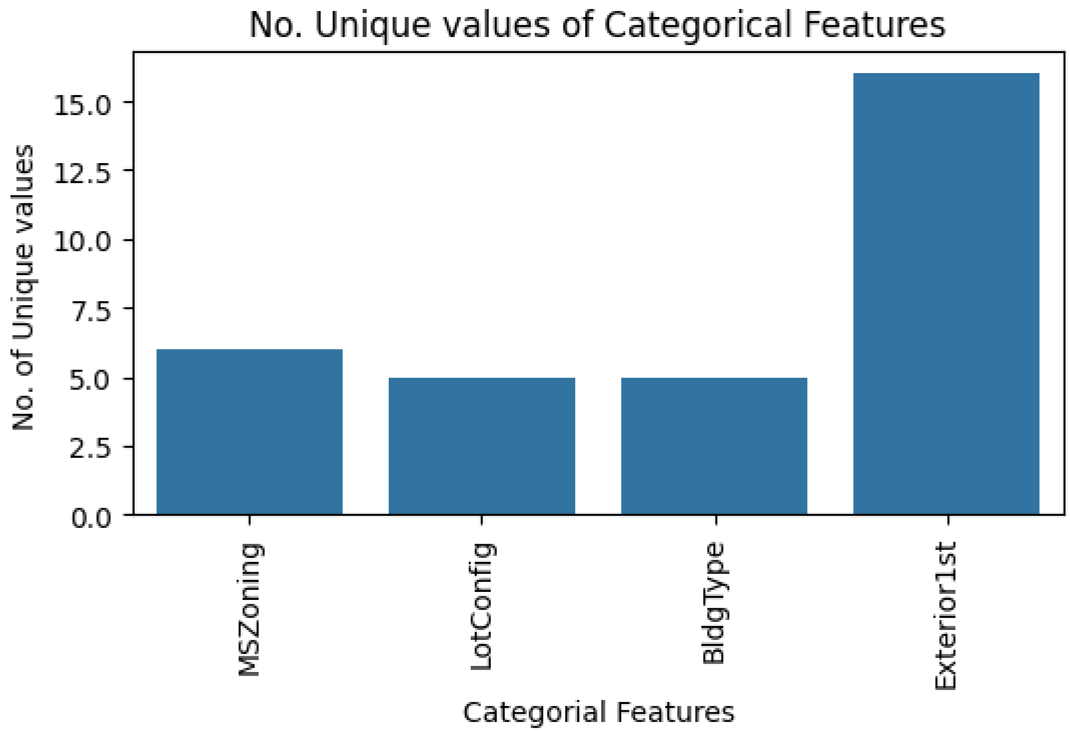
```
# Another way to get Categorical Columns in list :

obj = dtst.dtypes=='object'
object_cols = list(obj[obj].index)
print("Categorical Columns : ", object_cols)
```

```
Categorical Columns :  ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']
```

In [35]:

```
# Collect unique value counts for each categorical feature :
# 1. A list called unique_values is created to store the count of unique values
# for each categorical column.
# 2. A for loop iterates over each column name in object_cols.
# 3. For each column, dataset[col].unique().size calculates the number of unique
# values, which is then appended to the unique_values list.

unique_values = []
for col in obj_cols:
    unique_values.append(dtst[col].unique().size)
```

```
# Plotting the bar plot :
# 1. plt.figure(figsize=(6,3)): This line sets the size of the figure to be 10 inches wide and 6 inches tall.
# 2. plt.title('No. Unique values of Categorical Features'): This line sets the title of the plot.
# 3. plt.xticks(rotation=90): This line rotates the x-axis labels by 90 degrees to make them readable,
# especially when there are many categorical features.
# 4. sns.barplot(x=object_cols, y=unique_values): This line creates a bar plot using Seaborn, where the x-axis
# represents the categorical columns, and the y-axis represents the number of unique values in each column.

plt.figure(figsize=(6,3))
plt.title("No. Unique values of Categorical Features")
plt.xticks(rotation=90)
sns.barplot(x=object_cols, y=unique_values)
plt.xlabel('Categorial Features')
plt.ylabel('No. of Unique values')
plt.show()

# Another way to write code

# unique_values = []
# for col in object_cols:
#   unique_values.append(dataset[col].unique().size)
# plt.figure(figsize=(10,6))
# plt.title('No. Unique values of Categorical Features')
# plt.xticks(rotation=90)
# sns.barplot(x=object_cols,y=unique_values)


# 1. The plot shows that Exterior1st has around 16 unique categories and other features have
# around  6 unique categories.
# 2. To findout the actual count of each category we can plot the bargraph of each four features
# separately.
```


No. Unique values of Categorical Features
No. of Unique values
0.0
2.5
5.0
7.5
10.0
12.5
15.0
MSZoning
LotConfig
BldgType
Exterior1st
Categorial Features

In [36]:
```
dtst
```

Out[36]:

	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	BsmtFinSF2	Tota
0	0	60	RL	8450	Inside	1Fam	5	2003	2003	VinylSd	0.0	
1	1	20	RL	9600	FR2	1Fam	8	1976	1976	MetalSd	0.0	
2	2	60	RL	11250	Inside	1Fam	5	2001	2002	VinylSd	0.0	
3	3	70	RL	9550	Corner	1Fam	5	1915	1970	Wd Sdng	0.0	
4	4	60	RL	14260	FR2	1Fam	5	2000	2000	VinylSd	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	
2914	2914	160	RM	1936	Inside	Twnhs	7	1970	1970	CemntBd	0.0	
2915	2915	160	RM	1894	Inside	TwnhsE	5	1970	1970	CemntBd	0.0	
2916	2916	20	RL	20000	Inside	1Fam	7	1960	1996	VinylSd	0.0	
2917	2917	85	RL	10441	Inside	1Fam	5	1992	1992	HdBoard	0.0	
2918	2918	60	RL	9627	Inside	1Fam	5	1993	1994	HdBoard	0.0	

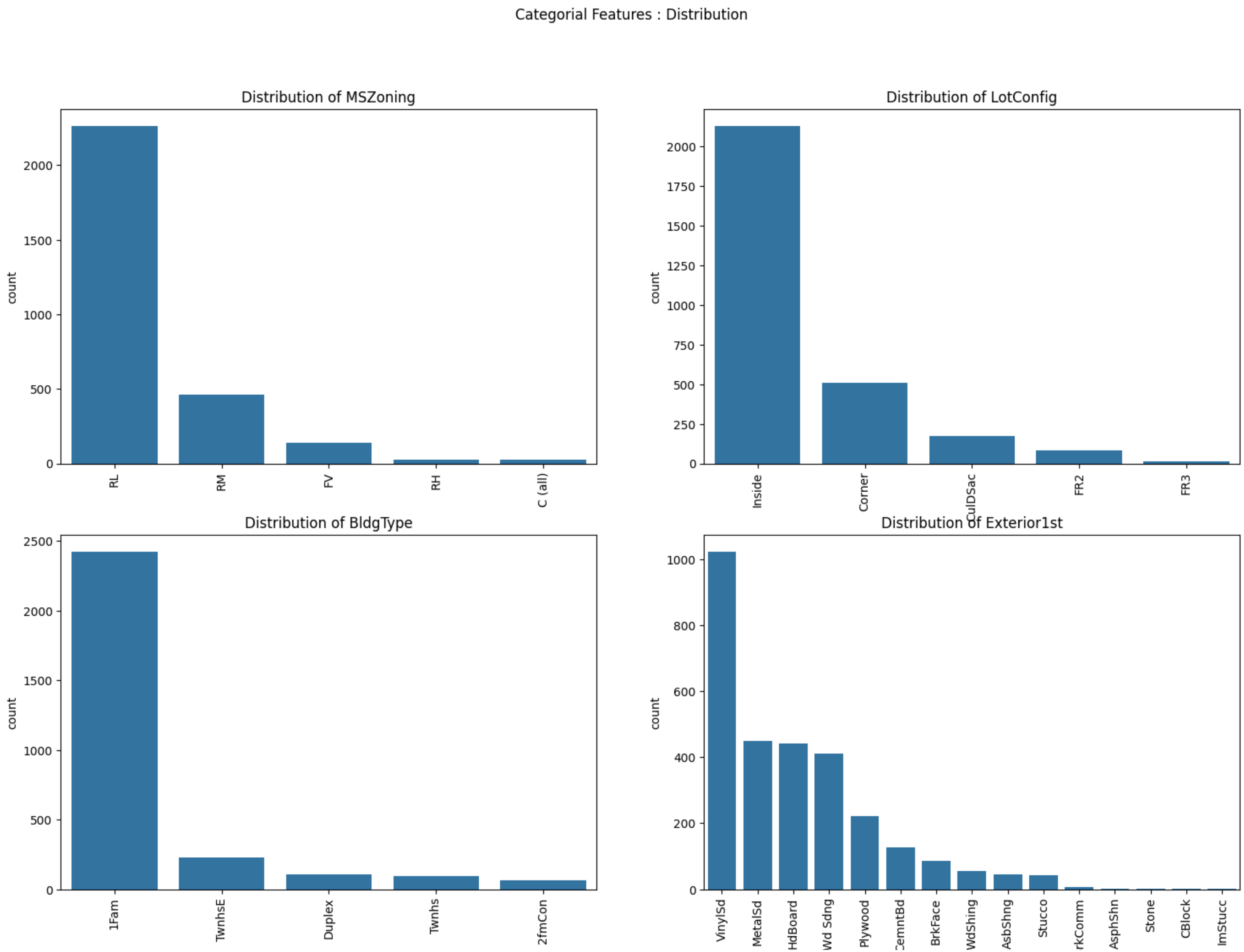
2919 rows × 13 columns

In [37]:

```
# Figure Setup :
# 1. plt.figure(figsize=(18, 36)):
   # This creates a new figure with a specified size of 18 inches wide and 36 inches tall.
# The larger height is likely to accommodate multiple subplots.
# 2. plt.title('Categorical Features: Distribution'): Sets the title of the entire figure.
# 3. plt.xticks(rotation=90): Rotates the x-axis labels by 90 degrees for readability,
# which helps when the category names are long.

# plt.figure(figsize=(18,36))
# plt.ltite('Categorial Features : Distribution')
# plt.xticks(rotation=90)

# Subplots for Each Categorical Column :
# 1. index = 1: This initializes a variable to track subplot positions.
# It's incremented with each iteration to create a new subplot.
```

```
# 2. for col in object_cols: Loops over each categorical column (from the list object_cols)
# in the DataFrame dataset.
# 3. y = dataset[col].value_counts():
# Calculates the count of unique values in the current categorical column col.
# 4. plt.subplot(11, 4, index): Creates a subplot grid with 11 rows and 4 columns.
   # The index parameter positions each subplot in the grid.
# You can adjust 11, 4 to fit your number of categorical features or the preferred layout.
# 5. plt.xticks(rotation=90):
# Again rotates the x-axis labels for each subplot to make them readable.
# 6. sns.barplot(x=list(y.index), y=y): Creates a bar plot using Seaborn, where the x-axis represents
# the unique categories and the y-axis represents their frequency count.
# 7. index += 1: Increments the index to move to the next subplot.

# Identifying categorical columns
obj_cols = [col for col in dtst.columns if dtst[col].dtype == 'object']
print("Categorial coulmns :", obj_cols)

# Plotting the distribution of categorical features
plt.figure(figsize=(18,12))   # Adjustable size
plt.suptitle('Categorial Features : Distribution')

index=1
for col in obj_cols:
    y = dtst[col].value_counts()
    plt.subplot(2,2,index)  #Adjusted for fewer columns(2x2 grid)
    plt.xticks(rotation=90)
    sns.barplot(x=list(y.index), y=y)
    plt.title(f'Distribution of {col}')
    index += 1


# Explanation :
# 1. object_cols: A list containing all the categorical column names. The loop iterates over these columns.
# 2. Subplots: Each categorical feature gets its own subplot, making it easier to see the distribution
# of values across different features.
# 3. plt.subplot(2, 2, index): This example assumes there are fewer categorical columns and uses a 2x2 grid for
# demonstration purposes Adjust the subplot grid dimensions(11,4) as needed based on the actual number of categorical columns.
```

```
Categorial coulmns : ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']
```


Categorial Features : Distribution
Distribution of MSZoning
count
RL
RM
FV
RH
C (all)
Distribution of LotConfig
Inside
Corner
CulDSac
FR2
FR3
Distribution of BldgType
1Fam
TwnhsE
Duplex
Twnhs
2fmCon
Distribution of Exterior1st
VinylSd
MetalSd
HdBoard
Wd Sdng
Plywood
CemntBd
BrkFace
WdShing
AsbShng
Stucco
BrkComm
AsphShn
Stone
CBlock
ImStucc

In [ ]:

```

```

In [ ]:

```
# Considerations :

# 1. Adjusting Grid Size: The grid size (11, 4) should be adjusted based on the number of categorical columns.
# Make sure the grid has enough cells to accommodate all plots.
# 2. Figure Size: The size of the figure may need to be adjusted depending on the number of subplots and the dataset's charact
# Larger datasets with more categories or longer names may require larger figures or different rotation angles for labels.
# 3. Label Overlap: Rotating x-axis labels helps prevent overlap, which is important for readability,
# especially when dealing with categorical variables with long or numerous categories.
# 4. Layout Adjustments: Using plt.tight_layout() or manually specifying margins can help
# prevent overlap between subplots and ensure the title and axis labels are visible.
```

In [ ]:

```
 # Data Cleaning :

# Data Cleaning is the way to improvise the data or remove incorrect, corrupted or irrelevant data.
# As in our dataset, there are some columns that are not important and irrelevant for the model training.
# So, we can drop that column before training. There are 2 approaches to dealing with empty/null values
# 1. We can easily delete the column/row (if the feature or record is not much important).
# 2. Filling the empty slots with mean/mode/0/NA/etc. (depending on the dataset requirement).
```

In [38]:

```
dtst = pd.read_csv("HousePricePrediction.csv")
dtst
```

Out[38]:

	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	BsmtFinSF2	Tota
0	0	60	RL	8450	Inside	1Fam	5	2003	2003	VinylSd	0.0	
1	1	20	RL	9600	FR2	1Fam	8	1976	1976	MetalSd	0.0	
2	2	60	RL	11250	Inside	1Fam	5	2001	2002	VinylSd	0.0	
3	3	70	RL	9550	Corner	1Fam	5	1915	1970	Wd Sdng	0.0	
4	4	60	RL	14260	FR2	1Fam	5	2000	2000	VinylSd	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	
2914	2914	160	RM	1936	Inside	Twnhs	7	1970	1970	CemntBd	0.0	
2915	2915	160	RM	1894	Inside	TwnhsE	5	1970	1970	CemntBd	0.0	
2916	2916	20	RL	20000	Inside	1Fam	7	1960	1996	VinylSd	0.0	
2917	2917	85	RL	10441	Inside	1Fam	5	1992	1992	HdBoard	0.0	
2918	2918	60	RL	9627	Inside	1Fam	5	1993	1994	HdBoard	0.0	

2919 rows × 13 columns

In [39]:

```
# As Id Column will not be participating in any prediction. So we can Drop it.

dtst.drop('Id', axis=1)
```

Out[39]:

	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	BsmtFinSF2	TotalBsmt
0	60	RL	8450	Inside	1Fam	5	2003	2003	VinylSd	0.0	856
1	20	RL	9600	FR2	1Fam	8	1976	1976	MetalSd	0.0	1262
2	60	RL	11250	Inside	1Fam	5	2001	2002	VinylSd	0.0	920
3	70	RL	9550	Corner	1Fam	5	1915	1970	Wd Sdng	0.0	756
4	60	RL	14260	FR2	1Fam	5	2000	2000	VinylSd	0.0	1145
...	...	...	...	...	...	...	...	...	...	...	
2914	160	RM	1936	Inside	Twnhs	7	1970	1970	CemntBd	0.0	546
2915	160	RM	1894	Inside	TwnhsE	5	1970	1970	CemntBd	0.0	546
2916	20	RL	20000	Inside	1Fam	7	1960	1996	VinylSd	0.0	1224
2917	85	RL	10441	Inside	1Fam	5	1992	1992	HdBoard	0.0	912
2918	60	RL	9627	Inside	1Fam	5	1993	1994	HdBoard	0.0	996

2919 rows × 12 columns

In [40]:

```
# Replacing SalePrice empty values with their mean values to make the data distribution symmetric.

dtst['SalePrice'] = dtst['SalePrice'].fillna(dtst['SalePrice']).mean()

# Drop records with null values (as the empty records are very less).

new_dtst = dtst.dropna()

# Checking features which have null values in the new dataframe (if there are still any).

# new_dtst.isnull().sum()
```

In [ ]:

```
# OneHotEncoder - For Label categorical features :
```

```
# One hot Encoding is the best way to convert categorical data into binary vectors.
# This maps the values to integer values. By using OneHotEncoder, we can easily convert object data into int.
# So for that, firstly we have to collect all the features which have the object datatype.
```

In [41]:
```
obj1 = new_dtst.dtypes=='object'
object_cols1 = list(obj1[obj1].index)
print("Categorical Columns : ", len(object_cols1))
object_cols1
```

```
Categorical Columns :  4
```

Out[41]: ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']

In [1]:
```
# Then once we have a list of all the features. We can apply OneHotEncoding to the whole list.
```

In [ ]:
```
# OH_encoder = OneHotEncoder(sparse=False)
# OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))
# OH_cols.index = new_dataset.index
# OH_cols.columns = OH_encoder.get_feature_names()
# df_final = new_dataset.drop(object_cols, axis=1)
# df_final = pd.concat([df_final, OH_cols], axis=1)
```

In [42]:
```
OH_encoder = OneHotEncoder(sparse_output=False)  # Initialize OneHotEncoder
# OneHotEncoder(sparse=False): This initializes an instance of the OneHotEncoder.
# The parameter sparse=False indicates that the output should be a dense array rather than a sparse matrix.
# This makes the output easier to work with directly in pandas.

OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dtst[object_cols1])) # One-hot encode the categorical columns
# 1. new_dataset[object_cols]: Here, new_dataset is the original DataFrame, and
# object_cols is a list of column names that contain categorical (or object type) data.
# 2. OH_encoder.fit_transform(...): This method fits the encoder to the categorical columns and
# transforms the data into one-hot encoded format.
# 3. pd.DataFrame(...): The result is converted into a pandas DataFrame for easier handling and
# integration with the rest of the DataFrame.

OH_cols.index = new_dtst.index    # Set the index to match the original DataFrame
OH_cols.columns = OH_encoder.get_feature_names_out(object_cols1) # Set column names to be more descriptive
# 1. OH_cols.index = new_dataset.index: This sets the index of the newly created OH_cols DataFrame to match the index of the o
# This ensures that the rows align correctly when concatenating.
# 2. OH_cols.columns = OH_encoder.get_feature_names_out(object_cols) : This sets the column names of the
```

```
# one-hot encoded DataFrame to be descriptive, showing which category each new column corresponds to.
# The get_feature_names_out method automatically generates names like columnName_value.

df_final = new_dtst.drop(object_cols1,axis=1)  # Drop original categorical columns
# Dropping Original Categorical Columns :
# This line creates a new DataFrame (df_final) by dropping the original categorical columns from new_dataset.
# The original categorical columns are being replaced by their one-hot encoded versions.

df_final = pd.concat([df_final, OH_cols], axis=1)  # Concatenate the one-hot encoded columns with the rest of the DataFrame
# Concatenating DataFrames :
# pd.concat([...], axis=1): This line concatenates the modified DataFrame
# (df_final, which no longer has the original categorical columns)
# with the new one-hot encoded columns (OH_cols).
# The axis=1 argument specifies that the concatenation should be column-wise.


print(df_final)
```

```
        Id  MSSubClass  LotArea  OverallCond  YearBuilt  YearRemodAdd  \
0        0          60     8450            5       2003          2003
1        1          20     9600            8       1976          1976
2        2          60    11250            5       2001          2002
3        3          70     9550            5       1915          1970
4        4          60    14260            5       2000          2000
...    ...         ...      ...          ...        ...           ...
2914  2914         160     1936            7       1970          1970
2915  2915         160     1894            5       1970          1970
2916  2916          20    20000            7       1960          1996
2917  2917          85    10441            5       1992          1992
2918  2918          60     9627            5       1993          1994

      BsmtFinSF2  TotalBsmtSF     SalePrice  MSZoning_C (all)  ...  \
0            0.0        856.0  180921.19589               0.0  ...
1            0.0       1262.0  180921.19589               0.0  ...
2            0.0        920.0  180921.19589               0.0  ...
3            0.0        756.0  180921.19589               0.0  ...
4            0.0       1145.0  180921.19589               0.0  ...
...          ...          ...           ...               ...  ...
2914         0.0        546.0  180921.19589               0.0  ...
2915         0.0        546.0  180921.19589               0.0  ...
2916         0.0       1224.0  180921.19589               0.0  ...
2917         0.0        912.0  180921.19589               0.0  ...
2918         0.0        996.0  180921.19589               0.0  ...

      Exterior1st_CemntBd  Exterior1st_HdBoard  Exterior1st_ImStucc  \
0                     0.0                  0.0                  0.0
1                     0.0                  0.0                  0.0
2                     0.0                  0.0                  0.0
3                     0.0                  0.0                  0.0
4                     0.0                  0.0                  0.0
...                   ...                  ...                  ...
2914                  1.0                  0.0                  0.0
2915                  1.0                  0.0                  0.0
2916                  0.0                  0.0                  0.0
2917                  0.0                  1.0                  0.0
2918                  0.0                  1.0                  0.0

      Exterior1st_MetalSd  Exterior1st_Plywood  Exterior1st_Stone  \
0                     0.0                  0.0                0.0
```

```
1                    1.0                 0.0                0.0
2                    0.0                 0.0                0.0
3                    0.0                 0.0                0.0
4                    0.0                 0.0                0.0
...                  ...                 ...                ...
2914                 0.0                 0.0                0.0
2915                 0.0                 0.0                0.0
2916                 0.0                 0.0                0.0
2917                 0.0                 0.0                0.0
2918                 0.0                 0.0                0.0

      Exterior1st_Stucco  Exterior1st_VinylSd  Exterior1st_Wd Sdng  \
0                    0.0                  1.0                  0.0
1                    0.0                  0.0                  0.0
2                    0.0                  1.0                  0.0
3                    0.0                  0.0                  1.0
4                    0.0                  1.0                  0.0
...                  ...                  ...                  ...
2914                 0.0                  0.0                  0.0
2915                 0.0                  0.0                  0.0
2916                 0.0                  1.0                  0.0
2917                 0.0                  0.0                  0.0
2918                 0.0                  0.0                  0.0

      Exterior1st_WdShing
0                    0.0
1                    0.0
2                    0.0
3                    0.0
4                    0.0
...                  ...
2914                 0.0
2915                 0.0
2916                 0.0
2917                 0.0
2918                 0.0

[2913 rows x 39 columns]
```

In [ ]:

```
# Splitting Dataset into Training and Testing
```

In [43]:
```
X = df_final.drop(['SalePrice'], axis=1)  # SalePrice column
Y = df_final['SalePrice']                 # Rest of the other columns

# Split the training set into training and validation set

X_train, X_valid, Y_train, Y_valid = train_test_split(X,Y, train_size=0.2,test_size=0.2, random_state=0)

print("Training set")
print(X_train)
print(Y_train)

print("\nvalidation set")
print(X_valid)
print(Y_valid)
```

```
Training set
        Id  MSSubClass  LotArea  OverallCond  YearBuilt  YearRemodAdd  \
1453  1453          20    17217            5       2006          2006   
2021  2021          20    19508            5       1974          1974   
188    188          90     7018            5       1979          1979   
1273  1273          80    11512            7       1959          2006   
1960  1960          20     7560            5       1971          1971   
...    ...         ...      ...          ...        ...           ...   
1790  1790          90     9671            5       1969          1969   
1045  1045          20    13680            5       1955          1955   
31      31          20     8544            6       1966          2006   
2727  2727          20    10368            6       1964          1964   
2031  2031         120     4385            5       2001          2001   

      BsmtFinSF2  TotalBsmtSF  MSZoning_C (all)  MSZoning_FV  ...  \
1453         0.0       1140.0               0.0          0.0  ...   
2021         0.0       1430.0               0.0          0.0  ...   
188          0.0       1086.0               0.0          0.0  ...   
1273         0.0       1019.0               0.0          0.0  ...   
1960       613.0        864.0               0.0          0.0  ...   
...          ...          ...               ...          ...  ...   
1790         0.0       1248.0               0.0          0.0  ...   
1045         0.0          0.0               0.0          0.0  ...   
31           0.0       1228.0               0.0          0.0  ...   
2727       748.0       1008.0               0.0          0.0  ...   
2031         0.0       1419.0               0.0          1.0  ...   

      Exterior1st_CemntBd  Exterior1st_HdBoard  Exterior1st_ImStucc  \
1453                  0.0                  0.0                  0.0   
2021                  0.0                  1.0                  0.0   
188                   0.0                  0.0                  0.0   
1273                  0.0                  0.0                  0.0   
1960                  0.0                  0.0                  0.0   
...                   ...                  ...                  ...   
1790                  0.0                  0.0                  0.0   
1045                  0.0                  0.0                  0.0   
31                    0.0                  1.0                  0.0   
2727                  0.0                  1.0                  0.0   
2031                  0.0                  0.0                  0.0   

      Exterior1st_MetalSd  Exterior1st_Plywood  Exterior1st_Stone  \
```

```
1453                  0.0                 0.0                0.0
2021                  0.0                 0.0                0.0
188                   0.0                 1.0                0.0
1273                  0.0                 1.0                0.0
1960                  1.0                 0.0                0.0
...                   ...                 ...                ...
1790                  1.0                 0.0                0.0
1045                  0.0                 0.0                0.0
31                    0.0                 0.0                0.0
2727                  0.0                 0.0                0.0
2031                  0.0                 0.0                0.0

      Exterior1st_Stucco  Exterior1st_VinylSd  Exterior1st_Wd Sdng  \
1453                 0.0                  1.0                  0.0
2021                 0.0                  0.0                  0.0
188                  0.0                  0.0                  0.0
1273                 0.0                  0.0                  0.0
1960                 0.0                  0.0                  0.0
...                  ...                  ...                  ...
1790                 0.0                  0.0                  0.0
1045                 0.0                  0.0                  0.0
31                   0.0                  0.0                  0.0
2727                 0.0                  0.0                  0.0
2031                 0.0                  1.0                  0.0

      Exterior1st_WdShing
1453                  0.0
2021                  0.0
188                   0.0
1273                  0.0
1960                  0.0
...                   ...
1790                  0.0
1045                  0.0
31                    0.0
2727                  0.0
2031                  0.0

[582 rows x 38 columns]
1453    180921.19589
2021    180921.19589
```

```
188     180921.19589
1273    180921.19589
1960    180921.19589
            ...
1790    180921.19589
1045    180921.19589
31      180921.19589
2727    180921.19589
2031    180921.19589
Name: SalePrice, Length: 582, dtype: float64

validation set
        Id  MSSubClass  LotArea  OverallCond  YearBuilt  YearRemodAdd  \
1728  1728          60    10274            7       1986          1986
2689  2689          60    13162            5       2006          2006
1151  1151          20    17755            4       1959          1959
2762  2762          20    10800            6       1961          1992
1180  1180          60    11170            5       1990          1991
...    ...         ...      ...          ...        ...           ...
1840  1840          90    10547            5       1978          1978
776    776          20    11210            5       2005          2006
1519  1519          20     8050            5       1959          1959
1359  1359          20    16737            5       2004          2005
563    563          50    21780            7       1918          1950

      BsmtFinSF2  TotalBsmtSF  MSZoning_C (all)  MSZoning_FV  ...  \
1728         0.0        676.0               0.0          0.0  ...
2689         0.0       2036.0               0.0          1.0  ...
1151         0.0       1466.0               0.0          0.0  ...
2762         0.0       1313.0               0.0          0.0  ...
1180         0.0       1216.0               0.0          0.0  ...
...          ...          ...               ...          ...  ...
1840         0.0       1152.0               0.0          0.0  ...
776          0.0       1614.0               0.0          0.0  ...
1519       162.0       1143.0               0.0          0.0  ...
1359         0.0       1980.0               0.0          0.0  ...
563          0.0       1163.0               0.0          0.0  ...

      Exterior1st_CemntBd  Exterior1st_HdBoard  Exterior1st_ImStucc  \
1728                  0.0                  0.0                  0.0
2689                  0.0                  0.0                  0.0
```

```
1151                    0.0                 1.0                 0.0
2762                    0.0                 1.0                 0.0
1180                    0.0                 0.0                 0.0
...                     ...                 ...                 ...
1840                    0.0                 0.0                 0.0
776                     0.0                 0.0                 0.0
1519                    0.0                 0.0                 0.0
1359                    0.0                 0.0                 0.0
563                     0.0                 0.0                 0.0

      Exterior1st_MetalSd  Exterior1st_Plywood  Exterior1st_Stone  \
1728                  0.0                  0.0                0.0
2689                  0.0                  0.0                0.0
1151                  0.0                  0.0                0.0
2762                  0.0                  0.0                0.0
1180                  1.0                  0.0                0.0
...                   ...                  ...                ...
1840                  0.0                  1.0                0.0
776                   0.0                  0.0                0.0
1519                  1.0                  0.0                0.0
1359                  0.0                  0.0                0.0
563                   0.0                  0.0                0.0

      Exterior1st_Stucco  Exterior1st_VinylSd  Exterior1st_Wd Sdng  \
1728                 0.0                  1.0                  0.0
2689                 0.0                  1.0                  0.0
1151                 0.0                  0.0                  0.0
2762                 0.0                  0.0                  0.0
1180                 0.0                  0.0                  0.0
...                  ...                  ...                  ...
1840                 0.0                  0.0                  0.0
776                  0.0                  1.0                  0.0
1519                 0.0                  0.0                  0.0
1359                 0.0                  1.0                  0.0
563                  0.0                  0.0                  1.0

      Exterior1st_WdShing
1728                  0.0
2689                  0.0
1151                  0.0
2762                  0.0
```

```
1180                  0.0
...                   ...
1840                  0.0
776                   0.0
1519                  0.0
1359                  0.0
563                   0.0

[583 rows x 38 columns]
1728    180921.19589
2689    180921.19589
1151    180921.19589
2762    180921.19589
1180    180921.19589
            ...
1840    180921.19589
776     180921.19589
1519    180921.19589
1359    180921.19589
563     180921.19589
Name: SalePrice, Length: 583, dtype: float64
```

In [ ]:
```
# Model and Accuracy
# As we have to train the model to determine the continuous values,
# so we will be using these regression models.

# 1. SVM-Support Vector Machine
# 2. Random Forest Regressor
# 3. Linear Regressor

# And To calculate loss we will be using the mean_absolute_percentage_error module.
# It can easily be imported by using sklearn library.
```

In [ ]:
```
# 2. Random Forest Regressor :
# Random Forest is an ensemble technique that uses multiple of decision trees and
# can be used for both regression and classification tasks.
```

In [44]:
```
model_RFR = RandomForestRegressor(n_estimators=10)
# Initialize the RandomForestRegressor model
model_RFR.fit(X_train, Y_train)
```

```python
# Train the model
Y_pred = model_RFR.predict(X_valid)
# Predict on the validation set

mape = mean_absolute_percentage_error(Y_valid, Y_pred)
print("Mean Absolute Percentage Error :", mape)

# Calculate and print the Mean Absolute Percentage Error (MAPE)
```

```
Mean Absolute Percentage Error : 3.2172936193018222e-15
```

In [ ]:

```python
# 1. SVM - Support vector Machine :
# SVM can be used for both regression and classification model.
# It finds the hyperplane in the n-dimensional plane.
```

In [45]:

```python
model_SVR = svm.SVR()
model_SVR.fit(X_train, Y_train)
Y_pred = model_SVR.predict(X_valid)

mape = mean_absolute_percentage_error(Y_valid, Y_pred)
print("Mean Absolute Percentage Error :", mape)
```

```
Mean Absolute Percentage Error : 0.0
```

In [ ]:

```python
# 3. Linear Regression
# Linear Regression predicts the final output-dependent value based on the given independent features.
# Like, here we have to predict SalePrice depending on features
# like MSSubClass, YearBuilt, BldgType, Exterior1st etc.
```

In [46]:

```python
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))
```

```
1.6086468096509106e-16
```

In [ ]: