

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset = pd.read_csv('/content/HousePricePrediction.csv')
```

```
# Printing first 5 records of the dataset
print(dataset.head(5))
```

```

➡ Id  MSSubClass  MSZoning  LotArea  LotConfig  BldgType  OverallCond  \
0    0           60        RL    8450      Inside    1Fam           5
1    1           20        RL    9600      FR2       1Fam           8
2    2           60        RL   11250     Inside    1Fam           5
3    3           70        RL    9550     Corner    1Fam           5
4    4           60        RL   14260     FR2       1Fam           5

   YearBuilt  YearRemodAdd  Exterior1st  BsmtFinSF2  TotalBsmtSF  SalePrice
0      2003         2003     VinylSd           0.0         856.0    208500.0
1      1976         1976     MetalSd           0.0        1262.0    181500.0
2      2001         2002     VinylSd           0.0         920.0    223500.0
3      1915         1970      Wd Sdng           0.0         756.0   140000.0
4      2000         2000     VinylSd           0.0        1145.0   250000.0
```

```
dataset.shape
```

```
(2919, 13)
```

data preprocessing

```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:", len(object_cols))
```

```
int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:", len(num_cols))
```

```
fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:", len(fl_cols))
```

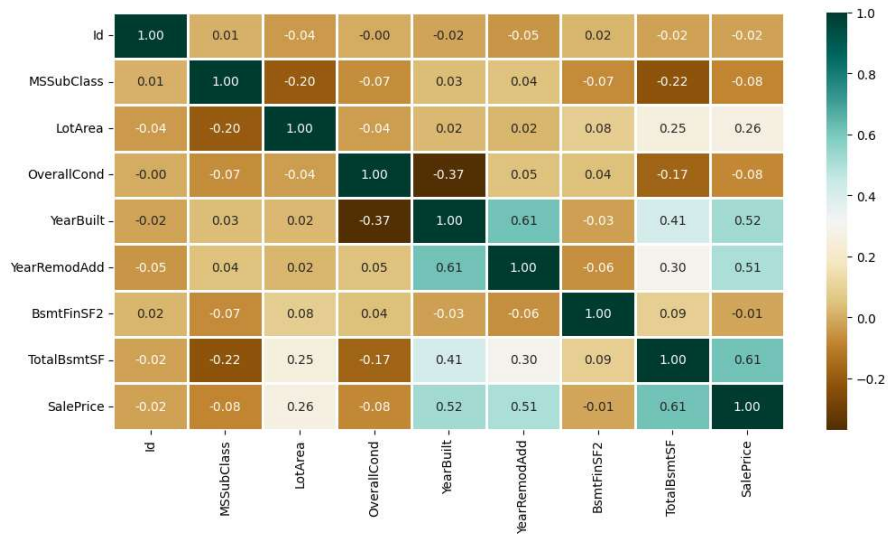
```

Categorical variables: 4
Integer variables: 6
Float variables: 3
```

exploratory data analysis

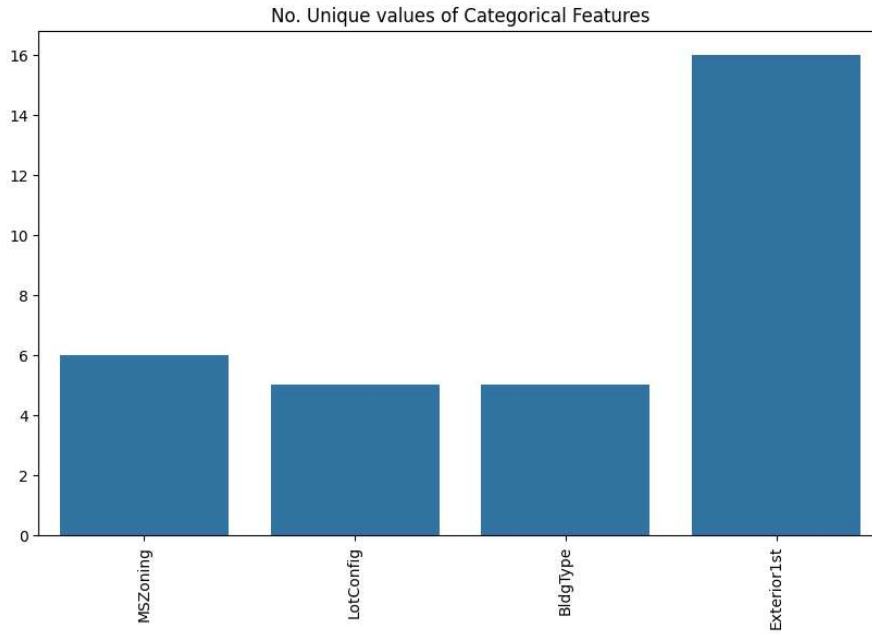
```
plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
             cmap = 'BrBG',
             fmt = '.2f',
             linewidths = 2,
             annot = True)
```

```
<ipython-input-4-8feaf9d49085>:2: FutureWarning: The default value of numeric_only in D
sns.heatmap(dataset.corr()),
<Axes: >
```



```
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)
```

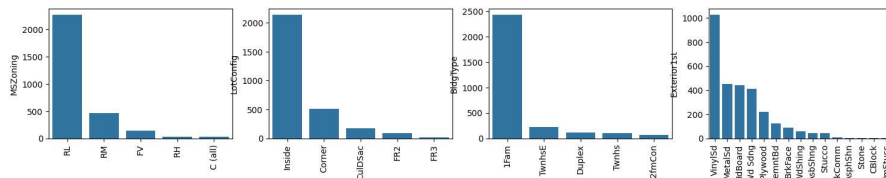
```
<Axes: title={'center': 'No. Unique values of Categorical Features'}>
```



```
plt.figure(figsize=(18, 36))
plt.title('Categorical 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
```

```
<ipython-input-6-648a2c778247>:8: MatplotlibDeprecationWarning: Auto-removal of overlap
plt.subplot(11, 4, index)
```



data cleaning

```
dataset.drop(['Id'],
             axis=1,
             inplace=True)
```

```
dataset['SalePrice'] = dataset['SalePrice'].fillna(
dataset['SalePrice'].mean())
```

```
new_dataset = dataset.dropna()
```

```
new_dataset.isnull().sum()
```

```

MSSubClass      0
MSZoning        0
LotArea         0
LotConfig       0
BldgType        0
OverallCond     0
YearBuilt       0
YearRemodAdd    0
Exterior1st     0
BsmtFinSF2      0
TotalBsmtSF     0
SalePrice       0
dtype: int64

```

```
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))

```

```

Categorical variables:
['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']
No. of. categorical features:  4

```

```

import pandas as pd
from sklearn.preprocessing import OneHotEncoder

```

```

# Assuming new_dataset is your original DataFrame and object_cols is a list of categorical column names
# Make sure to replace 'object_cols' with your actual list of categorical column names

```

```

# Step 1: Create a OneHotEncoder object
OH_encoder = OneHotEncoder(sparse=False)

```

```

# Step 2: Apply one-hot encoding to the categorical columns in new_dataset
OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))

```

```

# Step 3: Set the index of OH_cols to match the index of new_dataset
OH_cols.index = new_dataset.index

```

```

# Step 4: Assign column names to OH_cols using feature names obtained from the OneHotEncoder
OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))

```

```

# Step 5: Drop the original categorical columns from new_dataset
df_final = new_dataset.drop(object_cols, axis=1)

```

```

# Step 6: Concatenate df_final and OH_cols along the columns axis
df_final = pd.concat([df_final, OH_cols], axis=1)

```

```

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output`
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output`
  warnings.warn(

```

```
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
```