```
from google.colab import files
uploaded = files.upload()
```

Browse... No files selected. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving HousePricePrediction.xlsx to HousePricePrediction (1).xlsx
```

```
import pandas as pd

df = pd.read_excel('HousePricePrediction.xlsx')

df.head()
```

	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearB
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	

```
Step 1: Importing Libraries and Dataset

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

dataset = pd.read_excel("HousePricePrediction.xlsx")

print(dataset.head(5))

Output:
```

```
   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
```

Step 2: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
```

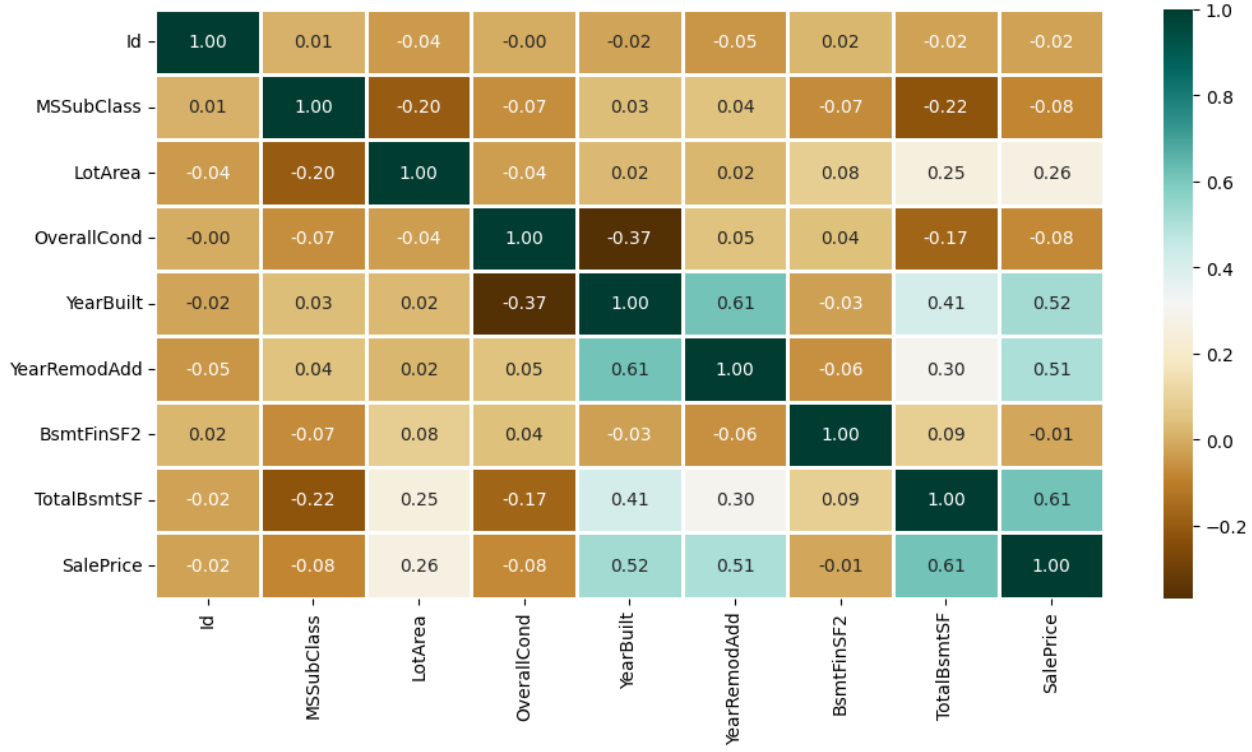
Step 3:Exploratory Data Analysis (EDA)

```
numerical_dataset = dataset.select_dtypes(include=['number'])

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

Output:

```
<Axes: >
```



Step 4: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()
```

Output:

	0
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

dtype: int64

Step 5:OneHotEncoder – For Label categorical

```
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))

Output:

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



OH_encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
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_out()
df_final = new_dataset.drop(object_cols, axis=1)
df_final = pd.concat([df_final, OH_cols], axis=1)
```

Step 6: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']

X_train, X_valid, Y_train, Y_valid = train_test_split(
    X, Y, train_size=0.8, test_size=0.2, random_state=0)
```

Step 7: Model Training and Accuracy

1. SVM – Support vector Machine

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import mean_absolute_percentage_error

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

```
print(mean_absolute_percentage_error(Y_valid, Y_pred))

Output:

    0.1870512931870423
```

2. Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

model_RFR = RandomForestRegressor(n_estimators=10)
model_RFR.fit(X_train, Y_train)
Y_pred = model_RFR.predict(X_valid)

mean_absolute_percentage_error(Y_valid, Y_pred)

Output:

    0.19360674296562172
```

3. Linear Regression	**3. Linear Regression**

```
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))

Output:

    0.1874168384159986
```