```
import pandas as pd

df=pd.read_csv('/content/Housing.csv')
print(df.head())
print(df.isnull().sum())
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```

```
import pandas as pd
df=pd.read_csv('/content/Housing.csv')
missing = df.isnull().sum()
print("Missing values per column:\n", missing[missing > 0])
df = df.dropna()

duplicates = df.duplicated().sum()
print(f'Duplicate rows: {duplicates}')
df = df.drop_duplicates()
df = df[df['price'] < df['price'].quantile(0.99)]
df = df[df['area'] < df['area'].quantile(0.99)]
X = df.drop('price', axis=1)
y = df['price']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
Missing values per column:
 Series([], dtype: int64)
Duplicate rows: 0
```

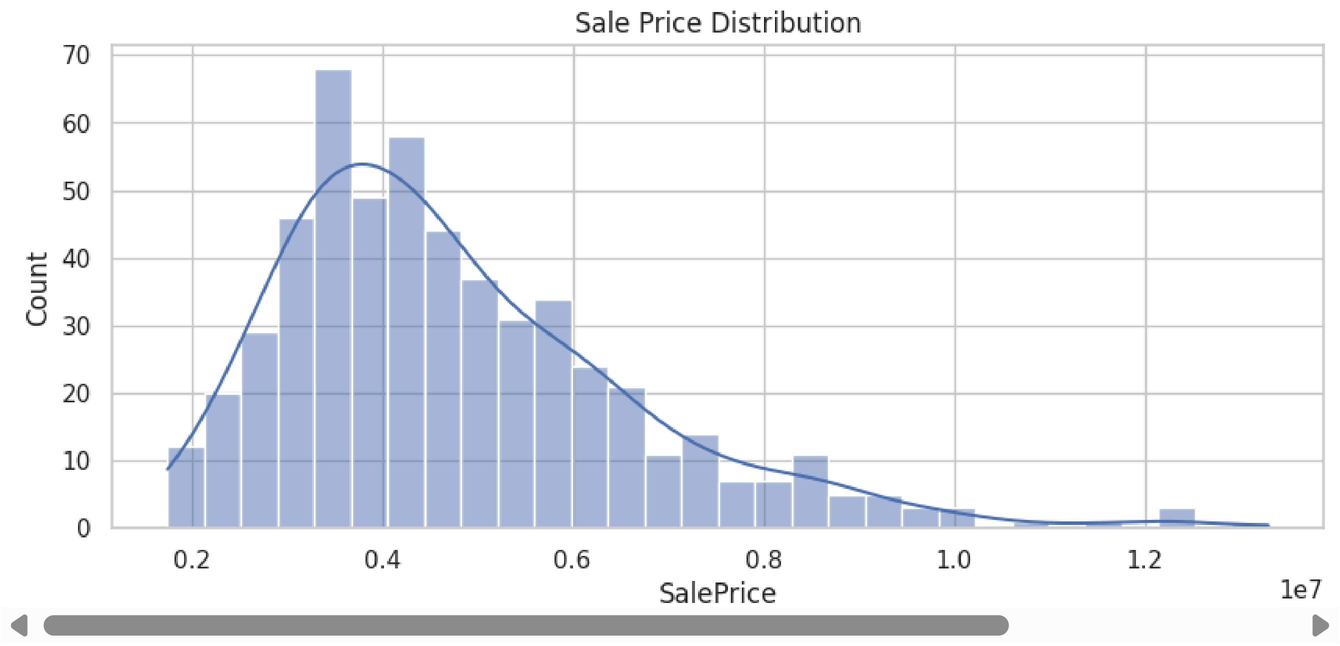
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.rcParams['figure.figsize'] = (10, 5)
df = pd.read_csv('/content/Housing.csv')
print("Dataset Shape:", df.shape)
print("\nData Types and Nulls:\n", df.info())
```

```
Dataset Shape: (545, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB

Data Types and Nulls:
 None
```

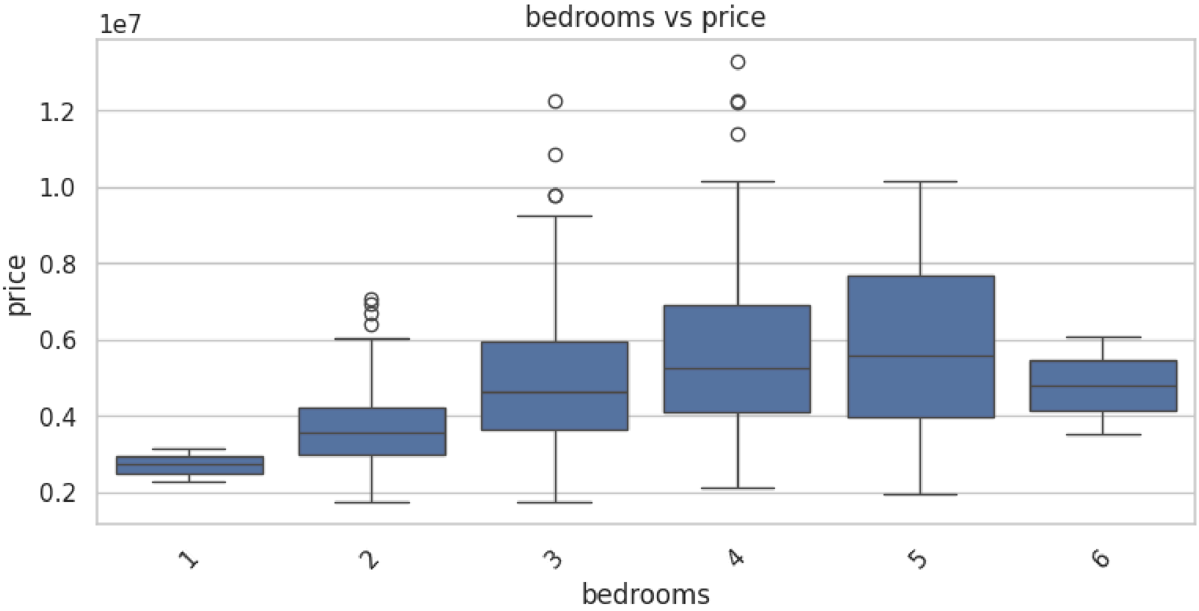
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
print("Dataset Shape:", df.shape)
print(df.info())
print(df.describe())
```

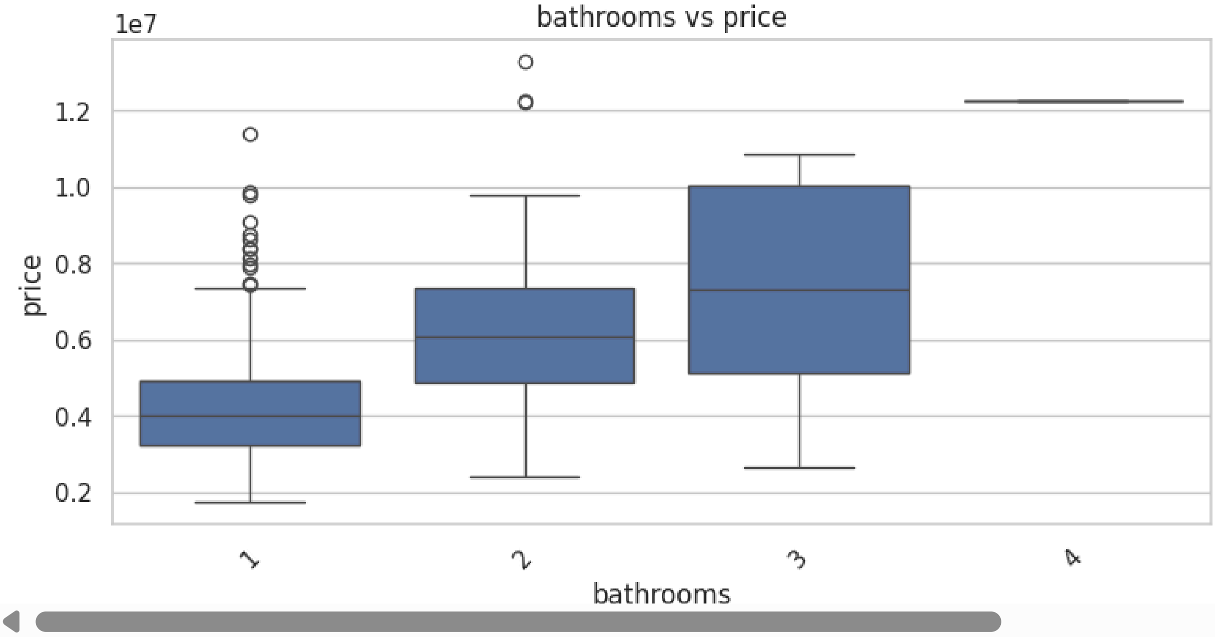
```
Dataset Shape: (545, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
None
              price          area    bedrooms   bathrooms     stories  \
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000   
mean   4.766729e+06   5150.541284    2.965138    1.286239    1.805505   
std    1.870440e+06   2170.141023    0.738064    0.502470    0.867492   
min    1.750000e+06   1650.000000    1.000000    1.000000    1.000000   
25%    3.430000e+06   3600.000000    2.000000    1.000000    1.000000   
50%    4.340000e+06   4600.000000    3.000000    1.000000    2.000000   
75%    5.740000e+06   6360.000000    3.000000    2.000000    2.000000   
max    1.330000e+07  16200.000000    6.000000    4.000000    4.000000   

          parking  
count  545.000000  
mean     0.693578  
std      0.861586  
min      0.000000  
25%      0.000000  
50%      0.000000  
75%      1.000000  
max      3.000000  
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('/content/Housing.csv')
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 4)
sns.histplot(df['price'], kde=True, bins=30)
plt.title('Sale Price Distribution')
plt.xlabel('SalePrice')
plt.show()
```

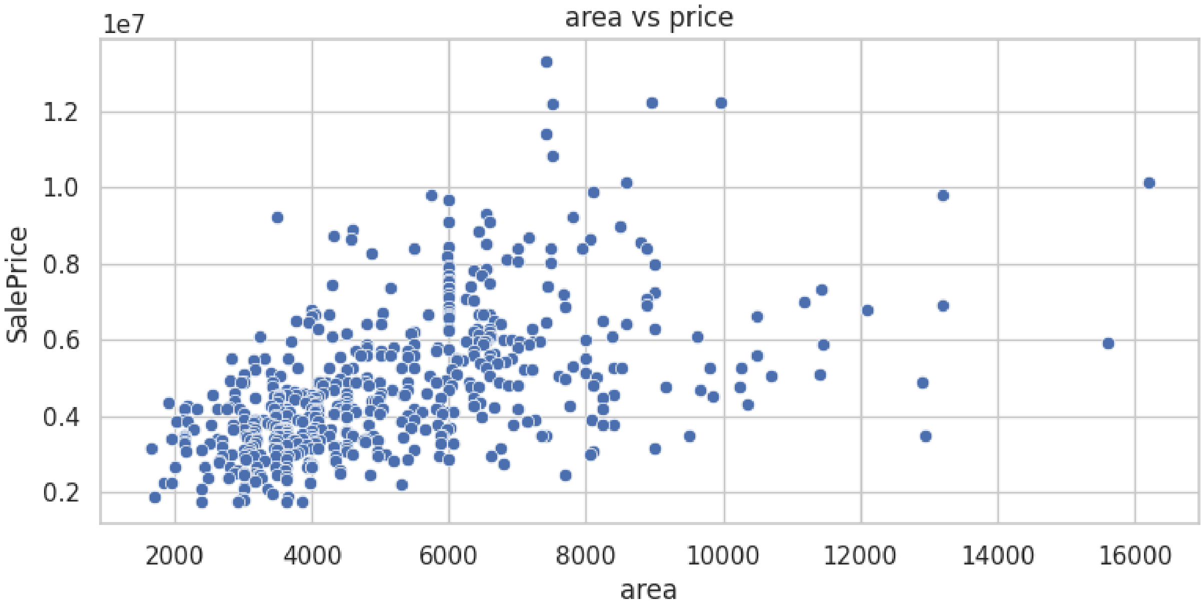


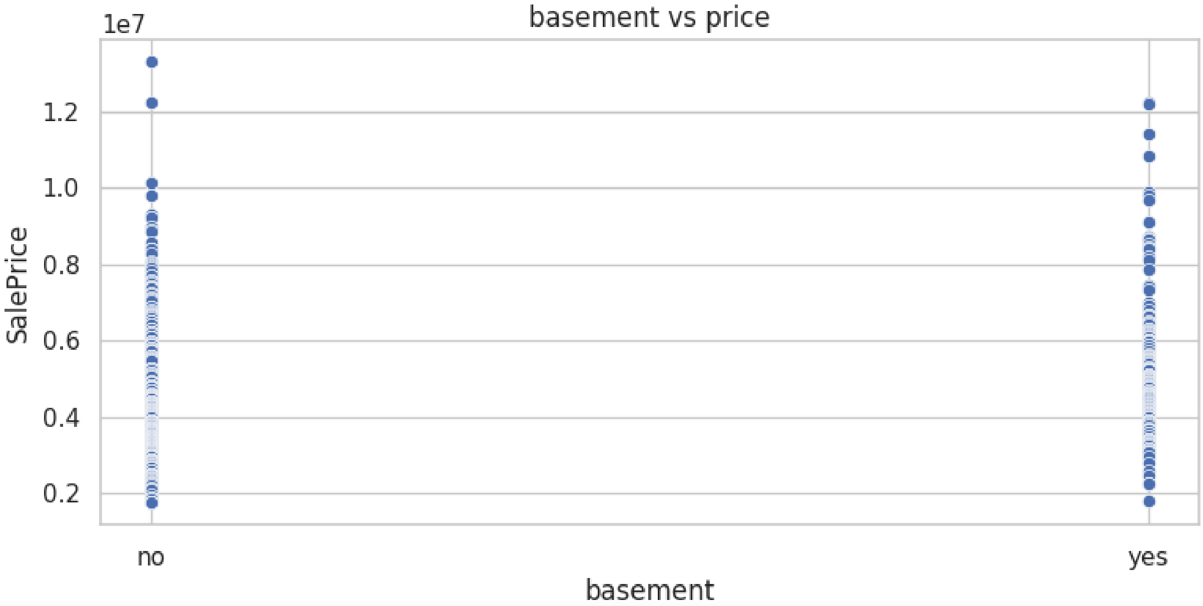
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('/content/Housing.csv')
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (9, 4)
categorical_features = ['bedrooms', 'bathrooms']
for feature in categorical_features:
    sns.boxplot(x=df[feature], y=df['price'])
    plt.title(f'{feature} vs price')
    plt.xticks(rotation=45)
    plt.show()
```





```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('/content/Housing.csv')
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (9, 4)
numerical_features = ['area', 'basement']
for feature in numerical_features:
    sns.scatterplot(x=df[feature], y=df['price'])
    plt.title(f'{feature} vs price')
    plt.xlabel(feature)
    plt.ylabel('SalePrice')
    plt.show()
```





```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline


df = pd.read_csv("/content/Housing.csv")


print(df.head())
print(df.info())


df.columns = df.columns.str.lower()


df = df.dropna()


categorical_features = df.select_dtypes(include='object').columns.tolist()
numerical_features = df.select_dtypes(include=['int64', 'float64']).drop('price', axis=1).columns.tolist()
```

```python
if 'bedrooms' in df.columns and 'bathrooms' in df.columns and 'area' in df.columns:
    df['total_rooms'] = df['bedrooms'] + df['bathrooms']
    df['area_per_room'] = df['area'] / df['total_rooms'].replace(0, np.nan)


X = df.drop('price', axis=1)
y = df['price']
numerical_pipeline = Pipeline([
    ('scaler', StandardScaler())
])

categorical_pipeline = Pipeline([
    ('encoder', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_pipeline, numerical_features),
    ('cat', categorical_pipeline, categorical_features)
])


X_preprocessed = preprocessor.fit_transform(X)

encoded_cols = preprocessor.named_transformers_['cat']['encoder'].get_feature_names_out(categorical_features)
all_feature_names = numerical_features + list(encoded_cols)


X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, y, test_size=0.2, random_state=42)

print(f"Preprocessed training shape: {X_train.shape}")
print(f"Features used: {all_feature_names}")
preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_pipeline, numerical_features),
    ('cat', categorical_pipeline, categorical_features)
])


X_preprocessed = preprocessor.fit_transform(X)


encoded_cols = preprocessor.named_transformers_['cat']['encoder'].get_feature_names_out(categorical_features)
all_feature_names = numerical_features + list(encoded_cols)


X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, y, test_size=0.2, random_state=42)

print(f"Preprocessed training shape: {X_train.shape}")
print(f"Features used: {all_feature_names}")
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no   
1  12250000  8960         4          4        4      yes        no       no   
2  12250000  9960         3          2        2      yes        no      yes   
3  12215000  7500         4          2        2      yes        no      yes   
4  11410000  7420         4          1        2      yes       yes      yes   

  hotwaterheating airconditioning  parking prefarea furnishingstatus  
0              no             yes        2      yes        furnished  
1              no             yes        3       no        furnished  
2              no              no        2      yes   semi-furnished  
3              no             yes        3      yes        furnished  
4              no             yes        2       no        furnished  
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
None
Preprocessed training shape: (436, 20)
Features used: ['area', 'bedrooms', 'bathrooms', 'stories', 'parking', 'mainroad_no', 'mainroad_yes', 'guestro
Preprocessed training shape: (436, 20)
Features used: ['area', 'bedrooms', 'bathrooms', 'stories', 'parking', 'mainroad_no', 'mainroad_yes', 'guestro
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline


df = pd.read_csv('/content/Housing.csv')


print(df.head())


X = df.drop('price', axis=1)
y = df['price']

categorical_cols = X.select_dtypes(include='object').columns.tolist()
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns.tolist()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ],
    remainder='passthrough'
)


model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model.fit(X_train, y_train)

predictions = model.predict(X_test)

print("Sample predictions:", predictions[:5])
print("Actual values     :", y_test.values[:5])
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
```

```
1              no              yes        3       no        furnished
2              no               no        2      yes   semi-furnished
3              no              yes        3      yes        furnished
4              no              yes        2       no        furnished
Sample predictions: [5164653.90033958 7224722.29802165 3109863.24240343 4612075.32722563
 3294646.25725961]
Actual values     : [4060000 6650000 3710000 6440000 2800000]
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv('/content/Housing.csv')

X = data.drop('price', axis=1)
y = data['price']
X= pd.get_dummies(X, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs faile
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
y_pred = model.predict(X_test)
print(y_pred,"y_prediction")
```

```
[3500000 3500000 4200000 4200000 4200000 3500000 3500000 4200000 4200000
 4200000 3500000 3500000 4200000 4200000 3500000 4200000 4200000 4200000
 3500000 3500000 4200000 3500000 3500000 4200000 3500000 3500000 4200000
 4200000 3500000 4200000 4200000 4200000 4200000 3500000 3500000 4200000
 4200000 4200000 4200000 4200000 3500000 4200000 3500000 4200000 3500000
 3500000 4200000 4200000 3500000 4200000 3500000 4200000 3500000 4200000
 4200000 4200000 4200000 4200000 3500000 4200000 4200000 3430000 3500000
 3500000 3500000 4200000 3500000 4200000 3500000 4200000 4200000 3500000
 3500000 3500000 4200000 4200000 4200000 4200000 4200000 3500000 3500000
 4200000 4200000 3500000 4200000 4200000 4200000 4200000 4200000 4200000
 5495000 3500000 3500000 4200000 3500000 4200000 3500000 3500000 4200000
 3500000 4200000 4200000 4200000 4200000 4200000 3500000 4200000 4200000
 3500000] y_prediction
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier  i  ?

RandomForestClassifier()
```

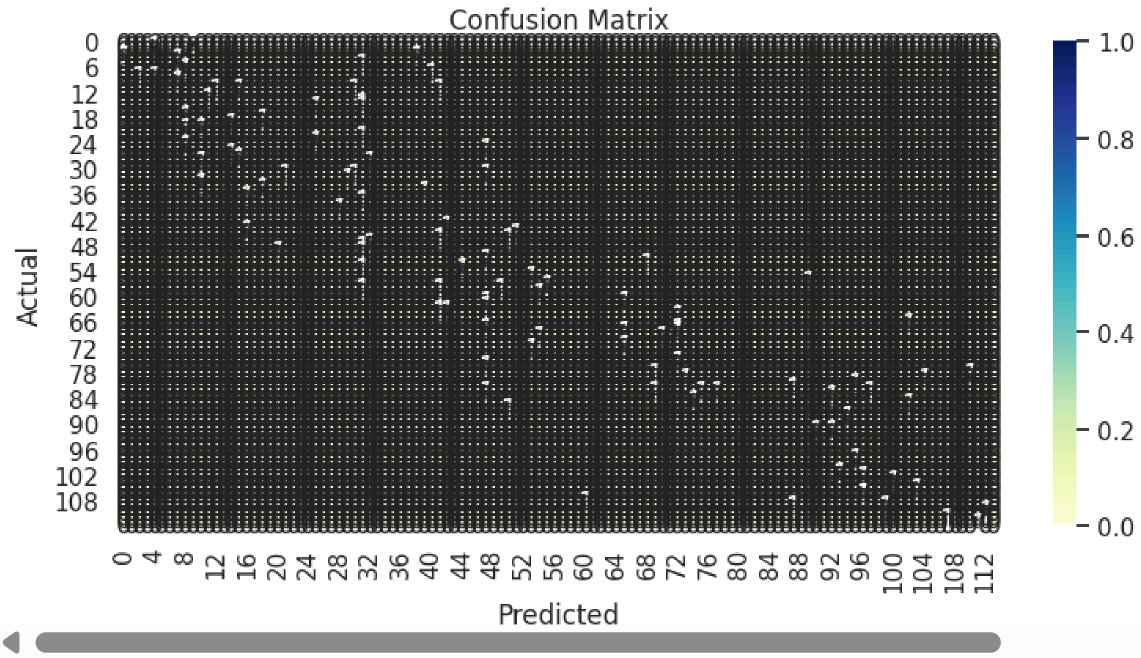
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
df=pd.read_csv('/content/Housing.csv')
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.009174311926605505
              precision    recall  f1-score   support

     1750000       0.00      0.00      0.00         1
     1820000       0.00      0.00      0.00         1
     1855000       0.00      0.00      0.00         0
     1890000       0.00      0.00      0.00         2
     2100000       0.00      0.00      0.00         1
     2233000       0.00      0.00      0.00         1
     2275000       0.00      0.00      0.00         1
     2380000       0.00      0.00      0.00         1
     2450000       0.00      0.00      0.00         2
     2520000       0.00      0.00      0.00         1
     2653000       0.00      0.00      0.00         0
     2660000       0.00      0.00      0.00         4
     2695000       0.00      0.00      0.00         0
     2800000       0.00      0.00      0.00         1
     2870000       0.00      0.00      0.00         1
     2940000       0.00      0.00      0.00         2
     2975000       0.00      0.00      0.00         0
     3003000       0.00      0.00      0.00         1
     3010000       0.50      1.00      0.67         1
     3045000       0.00      0.00      0.00         1
     3080000       0.00      0.00      0.00         2
     3115000       0.00      0.00      0.00         0
     3150000       0.00      0.00      0.00         1
     3220000       0.00      0.00      0.00         1
     3234000       0.00      0.00      0.00         1
     3290000       0.00      0.00      0.00         1
     3325000       0.00      0.00      0.00         1
     3353000       0.00      0.00      0.00         1
     3360000       0.00      0.00      0.00         2
     3430000       0.00      0.00      0.00         0
     3465000       0.00      0.00      0.00         0
     3500000       0.00      0.00      0.00         3
     3640000       0.00      0.00      0.00         1
     3675000       0.00      0.00      0.00         1
     3703000       0.00      0.00      0.00         1
     3710000       0.00      0.00      0.00         1
     3773000       0.00      0.00      0.00         1
     3780000       0.00      0.00      0.00         1
     3836000       0.00      0.00      0.00         0
     3850000       0.00      0.00      0.00         1
     3885000       0.00      0.00      0.00         0
     3920000       0.00      0.00      0.00         0
     3990000       0.00      0.00      0.00         0
     4007500       0.00      0.00      0.00         1
     4060000       0.00      0.00      0.00         1
     4165000       0.00      0.00      0.00         1
     4193000       0.00      0.00      0.00         2
     4200000       0.00      0.00      0.00         1
     4270000       0.00      0.00      0.00         1
     4340000       0.00      0.00      0.00         2
     4473000       0.00      0.00      0.00         0
     4480000       0.00      0.00      0.00         1
     4543000       0.00      0.00      0.00         1
     4550000       0.00      0.00      0.00         2
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
print("MAE:", mean_absolute_error(y_test, predictions))
print("MSE:", mean_squared_error(y_test, predictions))
print("R² Score:", r2_score(y_test, predictions))
```

```
MAE: 970043.4039201646
MSE: 1754318687330.7036
R² Score: 0.6529242642153106
```

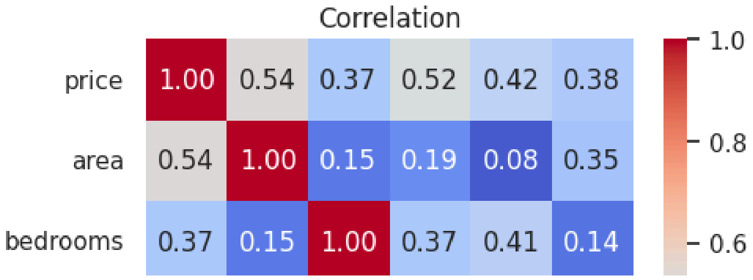
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(9, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu', linewidths=0.5)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



import numpy as np import seaborn as sns

sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d', cmap='blue') plt.title('Confusion Matrix') plt.xlabel('Predicted') plt.ylabel('Actual') plt.show()

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

plt.figure(figsize=(5, 4))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation')
plt.show()
```


Correlation
price
area
bedrooms