Assignment 1.2

January 27, 2025

[138]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[139]:
```
df=pd.read_csv("Housing.csv")
```

[140]:
```
df.info()
```

```
<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              539 non-null    float64
 2   bedrooms          543 non-null    float64
 3   bathrooms         545 non-null    int64
 4   stories           543 non-null    float64
 5   mainroad          544 non-null    object
 6   guestroom         545 non-null    object
 7   basement          543 non-null    object
 8   hotwaterheating   543 non-null    object
 9   airconditioning   542 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          543 non-null    object
 12  furnishingstatus  539 non-null    object
dtypes: float64(3), int64(3), object(7)
memory usage: 55.5+ KB
```

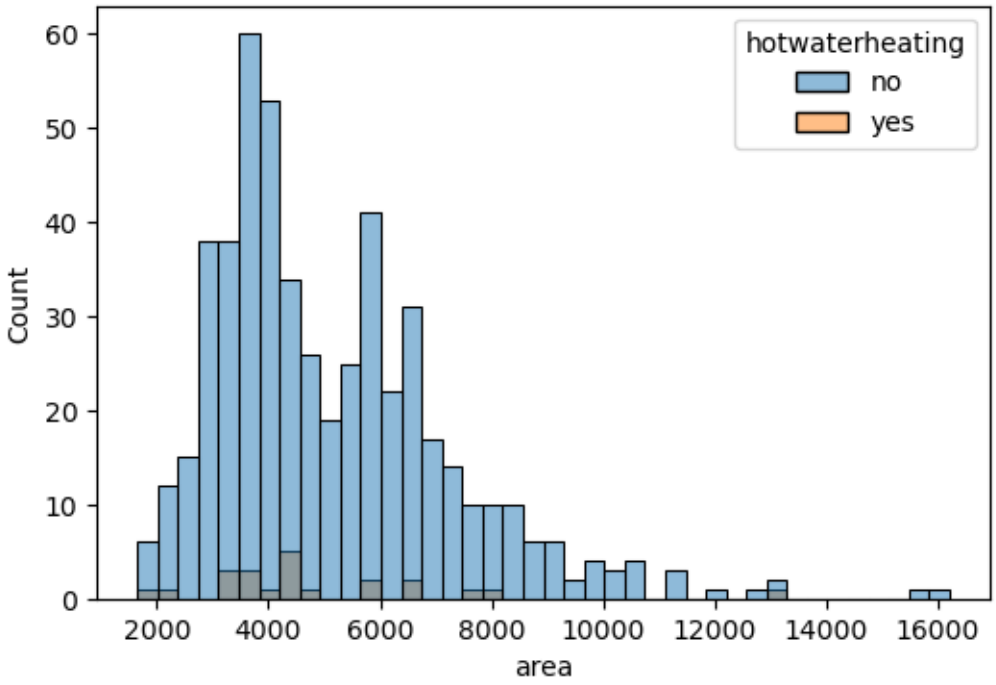
[141]:
```
df.describe(include='all')
```

[141]:

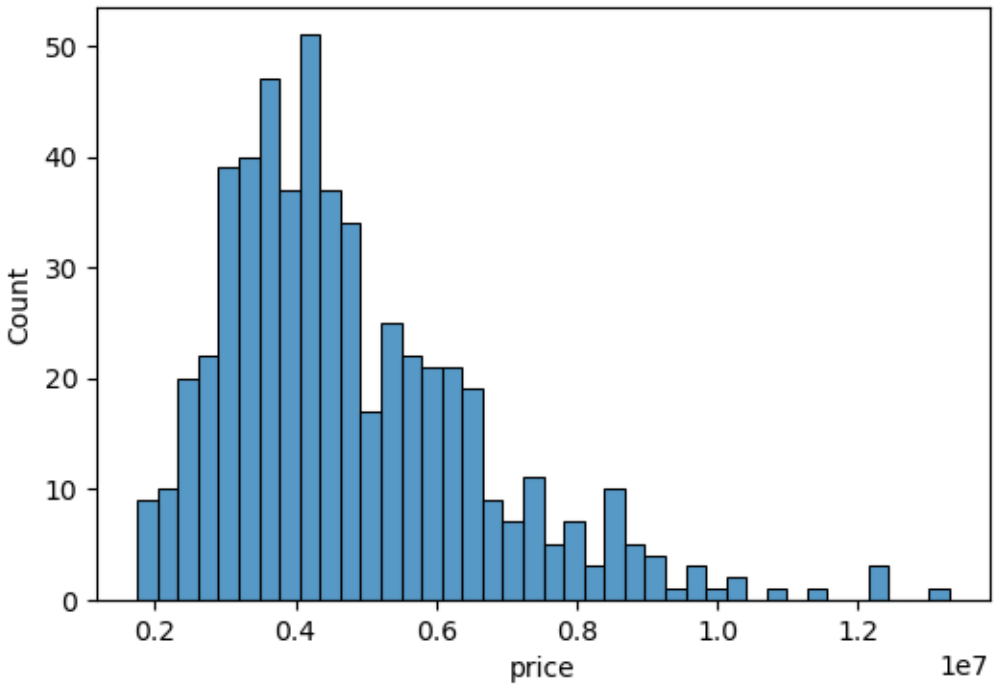
	price	area	bedrooms	bathrooms	stories	\
count	5.450000e+02	539.000000	543.000000	545.000000	543.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	4.766729e+06	5126.870130	2.963168	1.286239	1.804788	

```
std     1.870440e+06   2159.433198    0.738083    0.502470    0.869011
min     1.750000e+06   1650.000000    1.000000    1.000000    1.000000
25%     3.430000e+06   3577.000000    2.000000    1.000000    1.000000
50%     4.340000e+06   4510.000000    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

       mainroad guestroom basement hotwaterheating airconditioning  \
count       544       545      543              543             542
unique        2         2        2                2               2
top         yes        no       no               no              no
freq        467       448      354              519             372
mean        NaN       NaN      NaN              NaN             NaN
std         NaN       NaN      NaN              NaN             NaN
min         NaN       NaN      NaN              NaN             NaN
25%         NaN       NaN      NaN              NaN             NaN
50%         NaN       NaN      NaN              NaN             NaN
75%         NaN       NaN      NaN              NaN             NaN
max         NaN       NaN      NaN              NaN             NaN

           parking prefarea furnishingstatus
count   545.000000      543              539
unique         NaN        2                3
top            NaN       no   semi-furnished
freq           NaN      416              223
mean      0.693578      NaN              NaN
std       0.861586      NaN              NaN
min       0.000000      NaN              NaN
25%       0.000000      NaN              NaN
50%       0.000000      NaN              NaN
75%       1.000000      NaN              NaN
max       3.000000      NaN              NaN
```

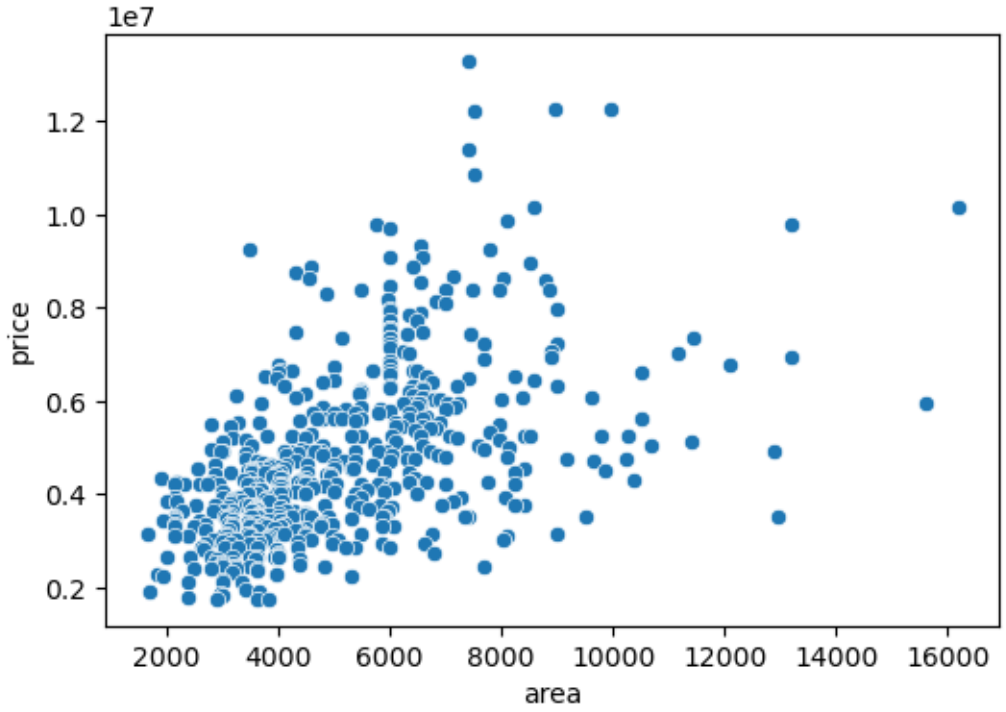
[142]:
```
plt.figure(figsize=(6,4))
sns.histplot(x=df['area'],bins=40,hue=df['hotwaterheating']).plot()
plt.show()
```



[143]:
```
plt.figure(figsize=(6,4))
sns.histplot(x=df['price'],bins=40).plot()
plt.show()
```



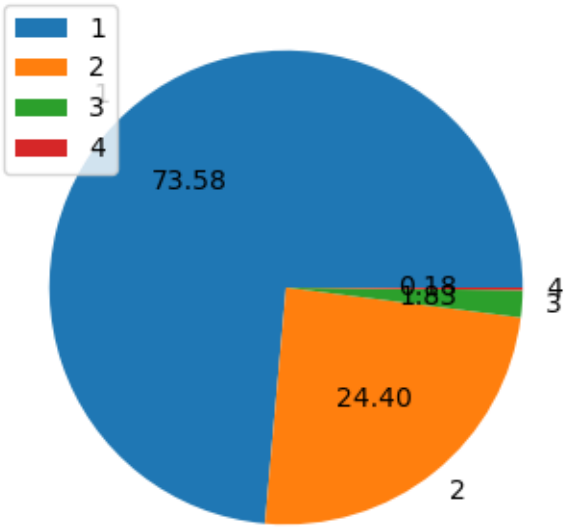
[144]:
```
plt.figure(figsize=(6,4))
sns.scatterplot(x=df['area'],y=df['price'])
plt.show()
```



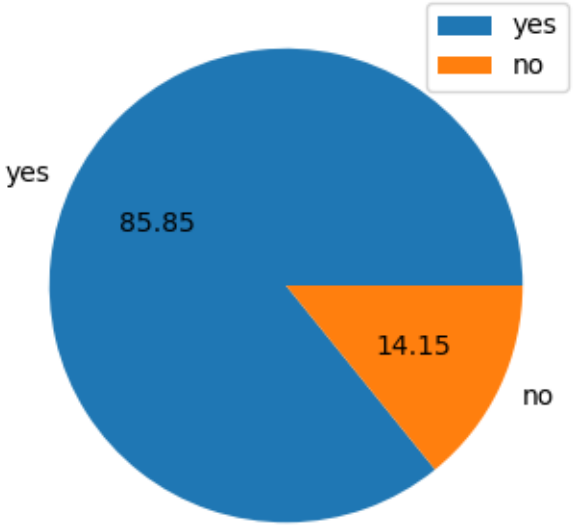
A bit of linear relation between area and price of the house

[145]:
```
def pieGraph(x):
    plt.figure(figsize=(6,4))
    plt.pie(x.value_counts().values,labels=x.value_counts().index,autopct='%1.
↪2f')
    plt.legend()
```

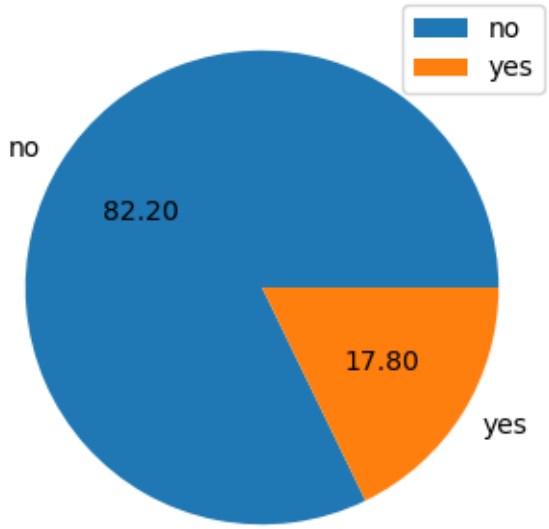
[146]:
```
pieGraph(df['bathrooms'])
```



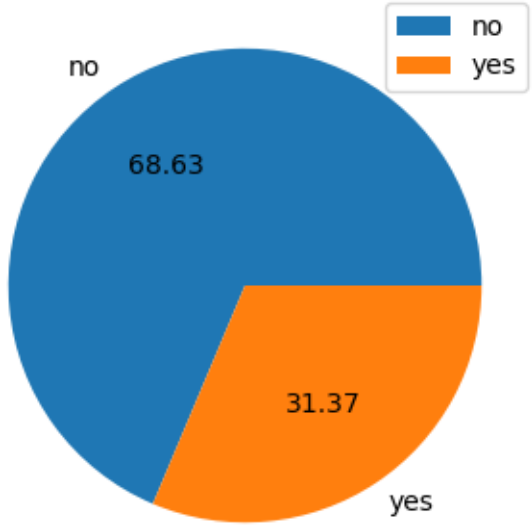
[147]:
```
pieGraph(df['mainroad'])
```



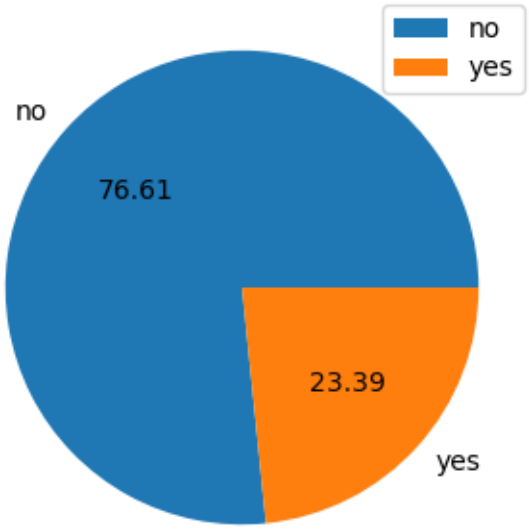
[148]:
```
pieGraph(df['guestroom'])
```



[149]:
```
pieGraph(df['airconditioning'])
```



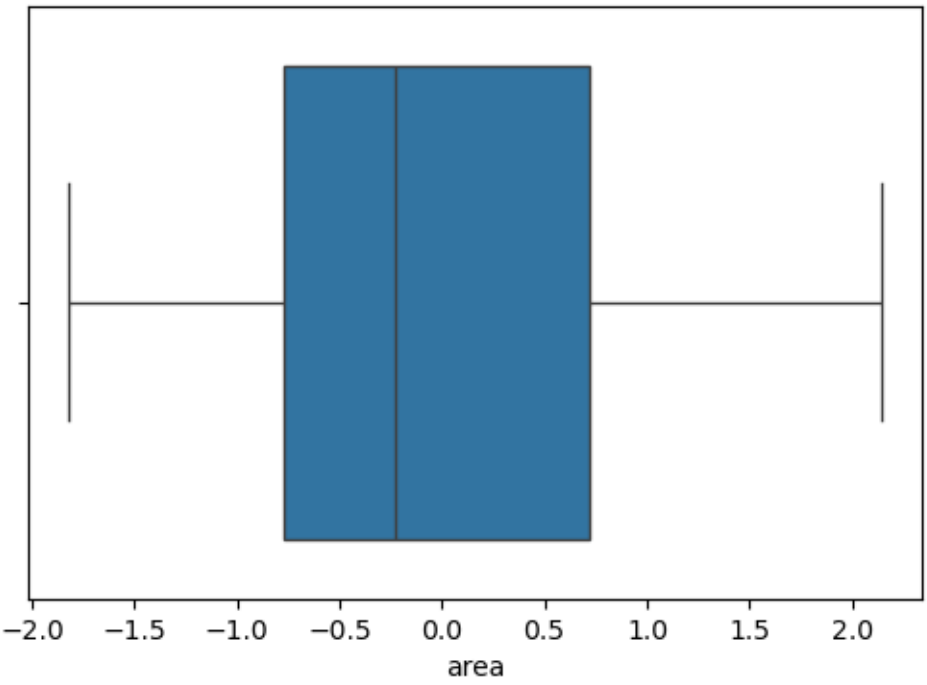
[150]:
```
pieGraph(df['prefarea'])
```



Most of the rooms have 1 bathroom, no air conditioning, no guestrooms, no hotwaterheating, on mainroad, not prefarea and no basement

0.0.1 Outliers

[217]:
```
plt.figure(figsize=(6,4))
sns.boxplot(x='area',data=df)
plt.show()
```



[152]:
```
print(df['area'].quantile(0.95))
```

```
9000.0
```

[153]:
```
df['area']=np.where(df['area']>9000,9000,df['area'])
```

0.0.2 Missing Values

[157]:
```
df.isna().sum()
```

[157]:
```
price              0
area               6
bedrooms           2
bathrooms          0
stories            2
mainroad           1
guestroom          0
basement           2
hotwaterheating    2
airconditioning    3
parking            0
prefarea           2
furnishingstatus   6
dtype: int64
```

[159]:
```
for column in df.columns:
    if(column=='area'):
        df[column]=df[column].fillna(df[column].mean())
    else:
        df[column]=df[column].fillna(df[column].mode()[0])
```

0.0.3 Handling Categorical Data

[166]:
```
from sklearn import preprocessing
label_encoder=preprocessing.LabelEncoder()
ordinal_encoder=preprocessing.OrdinalEncoder()
def label_encoding(column):
    df[column]=label_encoder.fit_transform(df[column])

def ordinal_encoding(column):
    df[column]=ordinal_encoder.fit_transform(df[[column]])
```

[167]:
```
label_encoding('mainroad')
label_encoding('guestroom')
label_encoding('basement')
label_encoding('hotwaterheating')
label_encoding('airconditioning')
ordinal_encoding('prefarea')
ordinal_encoding('furnishingstatus')
```

0.0.4 Standardization

[172]:
```
def standardization(column):
    df[column]=((df[column]-df[column].mean())/df[column].std())
```

[173]:
```
for column in df.columns:
    if df[column].max()>2:
        standardization(column)
```

0.0.5 Training the model

[175]:
```
y=df['price']
X=df.drop('price',axis=1)
```

[204]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪25,random_state=42)
```

[205]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression().fit(X_train,y_train)
```

[206]:
```
y_pred=model.predict(X_test)
```

[207]:
```
from sklearn.metrics import mean_squared_error
print("RMSE:",np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
RMSE: 0.6740633266965012
```

[216]:
```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],␣
 ↪color="red", linestyle="--")  # 45-degree line
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted")
plt.show()
```