Importing the required libraries

In [1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Dataset

Reading data from the given csv file and creating a DataFrame

In [108…
```
data = pd.read_csv("4033471.csv",names = None)
df = data
data.head()
```

Out[108]:

	TOWN/VILLAGE	number of rooms	age of property	distance to nearest town centre	property tax per year	number of pupils per teacher	property price
0	Framingham	7	37	7.31	996.87	13	30300
1	Framingham	7	38	7.31	1138.34	13	34600
2	Framingham	7	59	3.92	544.12	19	24400
3	Framingham	8	52	4.37	784.96	19	35200
4	Framingham	8	25	5.89	990.00	14	50000

Getting the data type of each column

In [109…
```
print("Data type of each column:")
data.dtypes
```

Out[109]:
```
Data type of each column:
TOWN/VILLAGE                       object
number of rooms                     int64
age of property                     int64
distance to nearest town centre   float64
property tax per year             float64
number of pupils per teacher        int64
property price                      int64
dtype: object
```

The number of rooms, number of pupils per teacher, property price and age of proeprty columns in the Dataframe are integer type, while distance to nearest town centre and property tax per year columns are float type.

Describing the index of the table

In [64]:
```
data.index
```

Out[64]:
```
RangeIndex(start=0, stop=21, step=1)
```

Getting info on the dataframe

In [110…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 7 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   TOWN/VILLAGE                    21 non-null     object 
 1   number of rooms                 21 non-null     int64  
 2   age of property                 21 non-null     int64  
 3   distance to nearest town centre  21 non-null     float64
 4   property tax per year           21 non-null     float64
 5   number of pupils per teacher    21 non-null     int64  
 6   property price                  21 non-null     int64  
dtypes: float64(2), int64(4), object(1)
memory usage: 1.3+ KB
```

Data Manipulation

Getting duplicate rows

In [111…
```
duplicateRows = data[data.duplicated()]
print(duplicateRows)
```

```
Empty DataFrame
Columns: [TOWN/VILLAGE, number of rooms, age of property, distance to nearest town ce
ntre, property tax per year, number of pupils per teacher, property price]
Index: []
```

There are no rows with the same values.

Getting the number of rows and columns in the DataFrame, names of columns

In [112…
```
table_size = data.shape

print('The size of the table is '+ str(table_size))   #converted the table_size into

print("The columns are: ")
columns = data.columns   #Gets the names of columns
print(columns)
```

```
The size of the table is (21, 7)
The columns are:
Index(['TOWN/VILLAGE', 'number of rooms', 'age of property',
       'distance to nearest town centre', 'property tax per year',
       'number of pupils per teacher', 'property price'],
      dtype='object')
```

Sorting data along any specific axis

In [38]:
```
data.sort_values(by='property price')
```

Out[38]:

	TOWN/VILLAGE	number of rooms	age of property	distance to nearest town centre	property tax per year	number of pupils per teacher	property price
9	Milton	6	97	3.76	414.45	21	13500
14	Quincy	6	100	1.41	999.00	20	15000
16	Quincy	5	38	2.52	489.44	18	16100
15	Quincy	7	100	1.53	1145.52	20	17200
13	Milton	6	22	10.59	607.88	22	18200
18	Quincy	6	90	2.83	589.76	18	19400
12	Milton	6	62	6.09	452.02	18	19400
19	Quincy	6	83	3.26	656.64	18	21600
17	Quincy	6	53	2.64	671.84	18	22100
20	Quincy	7	87	3.60	723.52	18	23800
2	Framingham	7	59	3.92	544.12	19	24400
6	Framingham	6	9	7.40	818.40	19	24800
11	Milton	7	58	3.37	630.48	18	28400
7	Milton	7	7	8.91	976.80	19	29600
5	Framingham	7	36	12.13	562.87	17	30100
0	Framingham	7	37	7.31	996.87	13	30300
1	Framingham	7	38	7.31	1138.34	13	34600
3	Framingham	8	52	4.37	784.96	19	35200
10	Milton	7	41	4.02	801.42	18	36100
8	Milton	8	8	8.91	1412.40	19	42800
4	Framingham	8	25	5.89	990.00	14	50000

The above chart is sorted in ascending order in accordance with the property price values. Same can be done with any of the columns.

The number of unique values in each row

In [49]:
```
data.nunique()
```

Out[49]:
```
TOWN/VILLAGE                    3
number of rooms                 4
age of property                19
distance to nearest town centre    19
property tax per year          21
number of pupils per teacher    8
property price                 20
dtype: int64
```

Calculating statistical data

Calculating the minimum, maximum and mean of all the columns in accordance with their Town/Village

In [51]:
```
town_group =data.groupby('TOWN/VILLAGE')
town_group.agg(['min','max','mean'])
```

Out[51]:

	number of rooms			age of property			distance to nearest town centre			property ta	
	min	max	mean	min	max	mean	min	max	mean	min	max
TOWN/VILLAGE											
Framingham	6	8	7.142857	9	59	36.571429	3.92	12.13	6.904286	544.12	1138.34
Milton	6	8	6.714286	7	97	42.142857	3.37	10.59	6.521429	414.45	1412.40
Quincy	5	7	6.142857	38	100	78.714286	1.41	3.60	2.541429	489.44	1145.52

Calculating Mean values in each column

In [55]:
```
numeric_data = df[['number of rooms', 'age of property', 'distance to nearest town ce

print("Mean:")
numeric_data.mean()
```

Out[55]:
```
Mean:
number of rooms                     6.666667
age of property                    52.476190
distance to nearest town centre     5.322381
property tax per year             781.272857
number of pupils per teacher       18.047619
property price                  26314.285714
dtype: float64
```

Calculating Median in each column

In [56]:
```
print("Median:")
numeric_data.median()
```

Out[56]:
```
Median:
number of rooms                     7.00
age of property                    52.00
distance to nearest town centre     4.02
property tax per year             723.52
number of pupils per teacher       18.00
property price                  24400.00
dtype: float64
```

Calculating Standard Deviation in each column

In [69]:
```
print("Standard Deviation:")
numeric_data.std()
```

```
Standard Deviation:
```

Out[69]:
```
number of rooms                       0.795822
age of property                      30.722336
distance to nearest town centre       3.016246
property tax per year               263.474537
number of pupils per teacher          2.290768
property price                     9525.927027
dtype: float64
```

Calculating Range of Values in each column

In [70]:
```
print("Range of Values")
print(numeric_data.max() - numeric_data.min())
```

```
Range of Values
number of rooms                       3.00
age of property                      93.00
distance to nearest town centre      10.72
property tax per year               997.95
number of pupils per teacher          9.00
property price                    36500.00
dtype: float64
```

Summarising all statistical data in one table

In [71]:
```
data.describe()
```

Out[71]:

	number of rooms	age of property	distance to nearest town centre	property tax per year	number of pupils per teacher	property price
count	21.000000	21.000000	21.000000	21.000000	21.000000	21.000000
mean	6.666667	52.476190	5.322381	781.272857	18.047619	26314.285714
std	0.795822	30.722336	3.016246	263.474537	2.290768	9525.927027
min	5.000000	7.000000	1.410000	414.450000	13.000000	13500.000000
25%	6.000000	36.000000	3.260000	589.760000	18.000000	19400.000000
50%	7.000000	52.000000	4.020000	723.520000	18.000000	24400.000000
75%	7.000000	83.000000	7.310000	990.000000	19.000000	30300.000000
max	8.000000	100.000000	12.130000	1412.400000	22.000000	50000.000000

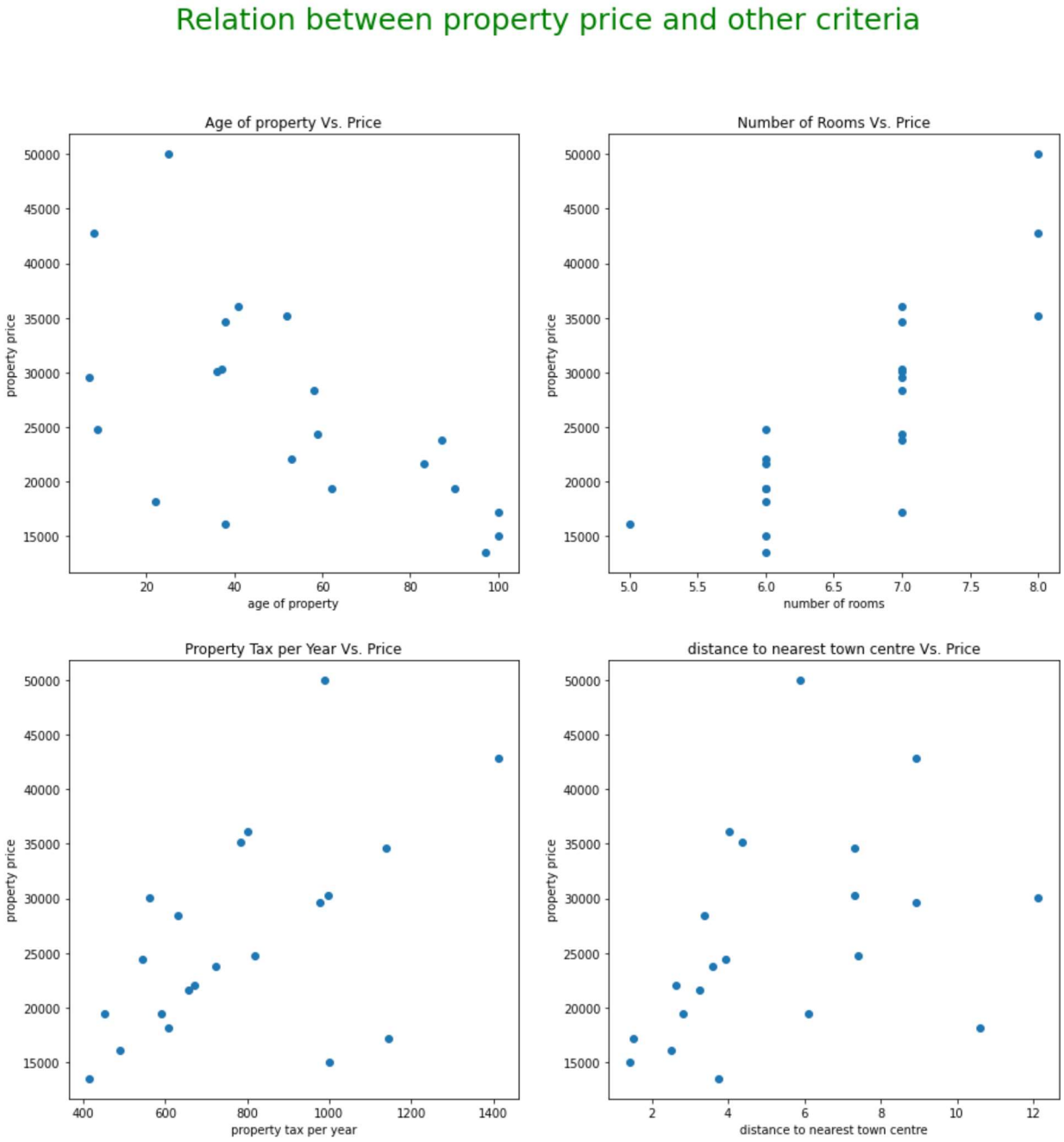
Visual Representation

Scatter Plot

The relationship between two numeric variables is shown in a scatter plot. A circle is used to represent each data point. [1]

In [77]:
```
# import pandas as pd
# import matplotlib.pyplot as plt
```

```
fig = plt.figure(figsize = (15,15))
fig.suptitle('Relation between property price and other criteria', fontsize=25, color

plt.subplot(2,2,1)
plt.scatter(df['age of property'], df['property price'])
# Adding Title to the Plot
plt.title("Age of property Vs. Price")
# Setting the X and Y labels
plt.xlabel('age of property')
plt.ylabel('property price')

plt.subplot(2,2,2)
plt.scatter(df['number of rooms'], df['property price'])
plt.title("Number of Rooms Vs. Price")
plt.xlabel('number of rooms')
plt.ylabel('property price')

plt.subplot(2,2,3)
plt.scatter(df['property tax per year'], df['property price'])
plt.title("Property Tax per Year Vs. Price")
plt.xlabel('property tax per year')
plt.ylabel('property price')

plt.subplot(2,2,4)
plt.scatter(df['distance to nearest town centre'], df['property price'])
plt.title("distance to nearest town centre Vs. Price")
plt.xlabel('distance to nearest town centre')
plt.ylabel('property price')


plt.show()
```



Box Plot

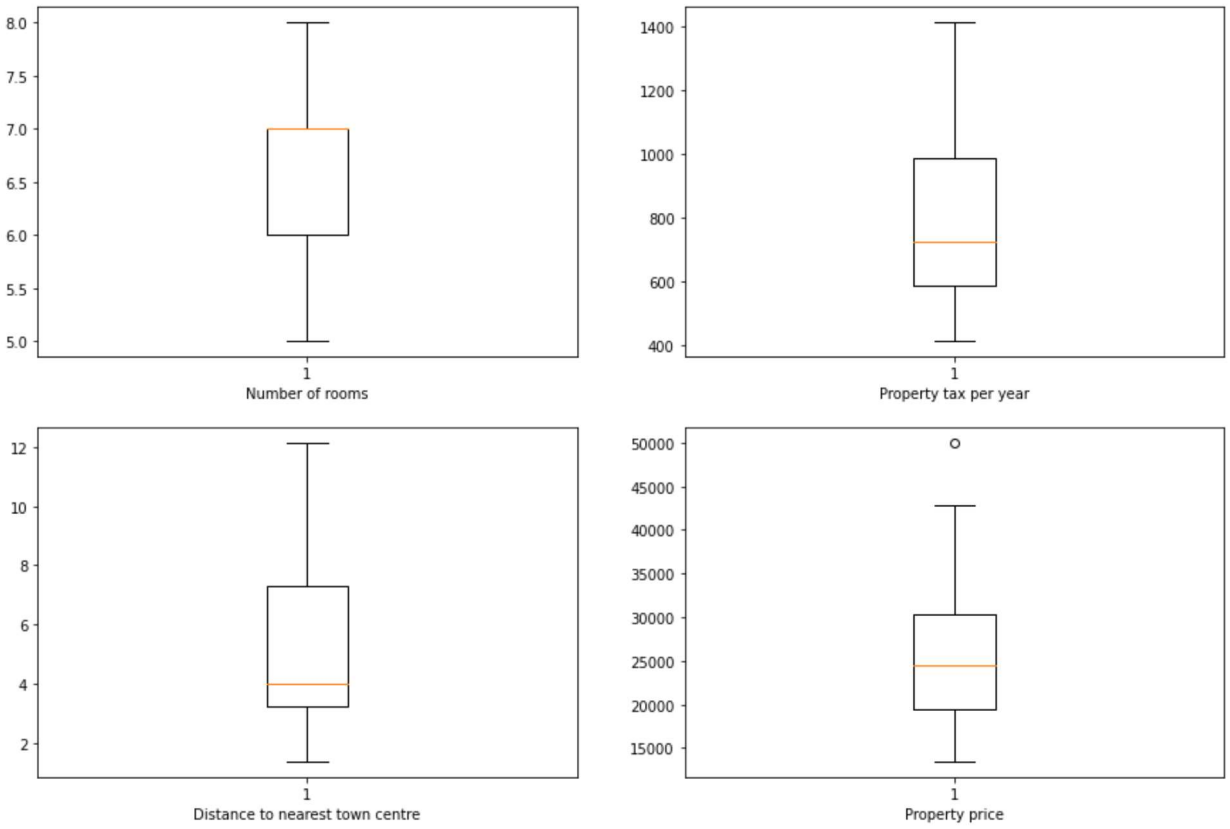
The distribution of a numeric variable for one or more groups is summarized using a boxplot. It allows acquiring the median, quartiles, and outliers quickly, but it also hides the individual data points in the dataset. [1]

In [83]:

```
fig, ((axis1,axis2),(axis3,axis4)) = plt.subplots(2,2,sharex=False,sharey=False)
fig.set_size_inches(15,10)
fig.suptitle("Boxplot",fontsize=18, color="g")

axis1.boxplot(df['number of rooms'])
axis1.set_xlabel("Number of rooms")

axis2.boxplot(df['property tax per year'])
axis2.set_xlabel("Property tax per year")
```

```
axis3.boxplot(df['distance to nearest town centre'])
axis3.set_xlabel("Distance to nearest town centre")

axis4.boxplot(df['property price'])
axis4.set_xlabel("Property price")

plt.show()
```

Boxplot



In [114…

```
p_list = df['property price'].values.tolist()
price_freq = [0,0,0,0]
for item in range(0,len(p_list)):
    if (p_list[item]>10000 and p_list[item]<=20000):
        price_freq[0]+=1
    if (p_list[item]>20000 and p_list[item]<=30000):
        price_freq[1]+=1
    if (p_list[item]>30000 and p_list[item]<=40000):
        price_freq[2]+=1
    if (p_list[item]>40000 and p_list[item]<=50000):
        price_freq[3]+=1

price_label = ["10001-20000","20001-30000","30001-40000","40001-50000"]

age_list = df['age of property'].values.tolist()
age_freq = [0,0,0,0]
for item in range(0,len(age_list)):
    if (age_list[item]>1 and age_list[item]<=25):
        age_freq[0]+=1
    if (age_list[item]>26 and age_list[item]<=50):
        age_freq[1]+=1
```

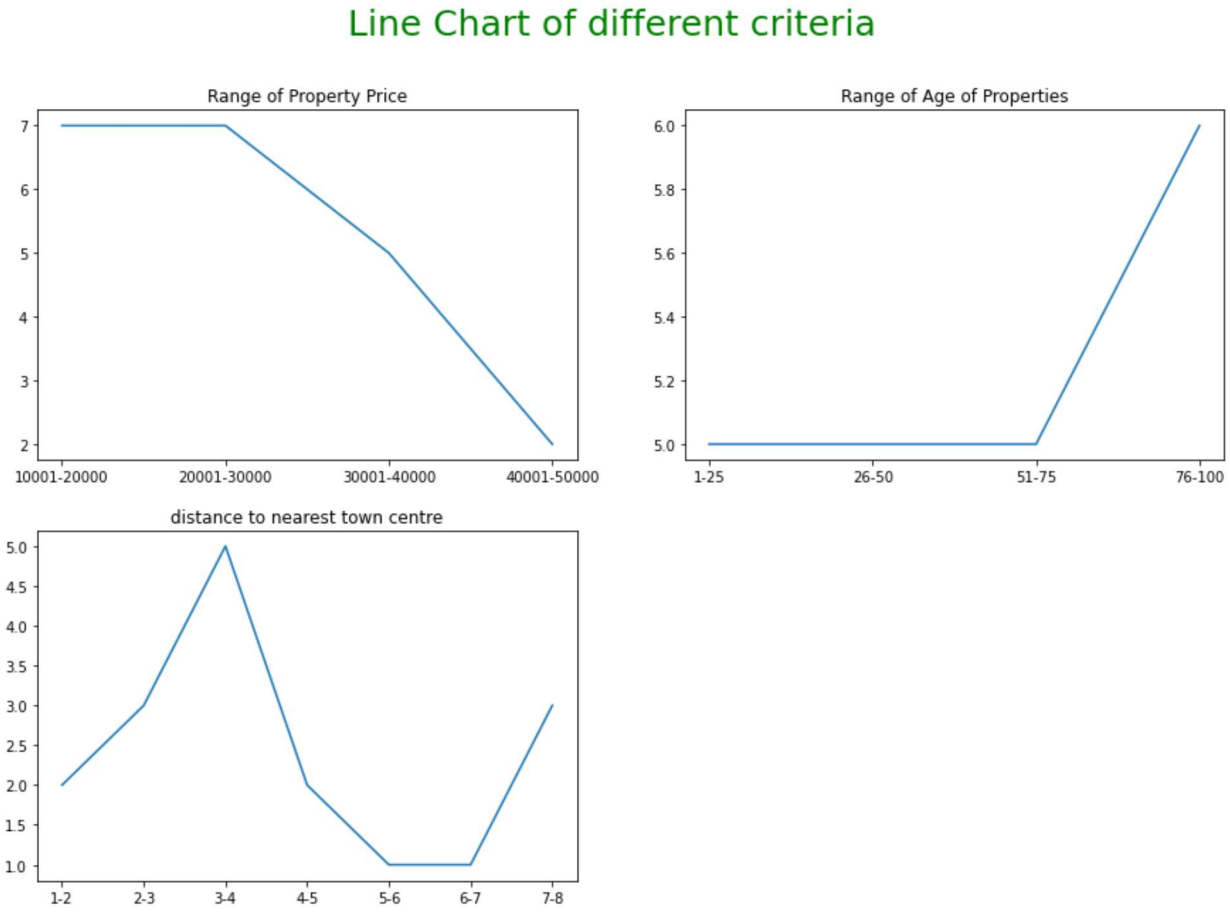
```
    if (age_list[item]>51 and age_list[item]<=75):
        age_freq[2]+=1
    if (age_list[item]>76 and age_list[item]<=100):
        age_freq[3]+=1

age_label = ["1-25","26-50","51-75","76-100"]


dist_list = df['distance to nearest town centre'].values.tolist()
dist_freq = [0,0,0,0,0,0,0]
for item in range(0,len(dist_list)):
    if (dist_list[item]>1.0 and dist_list[item]<=2.0):
        dist_freq[0]+=1
    if (dist_list[item]>2.0 and dist_list[item]<=3.0):
        dist_freq[1]+=1
    if (dist_list[item]>3.0 and dist_list[item]<=4.0):
        dist_freq[2]+=1
    if (dist_list[item]>4.0 and dist_list[item]<=5.0):
        dist_freq[3]+=1
    if (dist_list[item]>5.0 and dist_list[item]<=6.0):
        dist_freq[4]+=1
    if (dist_list[item]>6.0 and dist_list[item]<=7.0):
        dist_freq[5]+=1
    if (dist_list[item]>7.0 and dist_list[item]<=8.0):
        dist_freq[6]+=1

dist_label = ["1-2","2-3","3-4","4-5","5-6","6-7","7-8"]
```

Line Plot

A line graph depicts the progression of one or more numeric variables.

In [115…

```
fig = plt.figure(figsize = (15,10))
fig.suptitle('Line Chart of different criteria', fontsize=25, color = 'g')

plt.subplot(2,2,1)
plt.title("Range of Property Price")
plt.plot(price_label, price_freq)

plt.subplot(2,2,2)
plt.title("Range of Age of Properties")
plt.plot(age_label, age_freq)

plt.subplot(2,2,3)
plt.title("distance to nearest town centre")
plt.plot(dist_label, dist_freq)

plt.show()
```



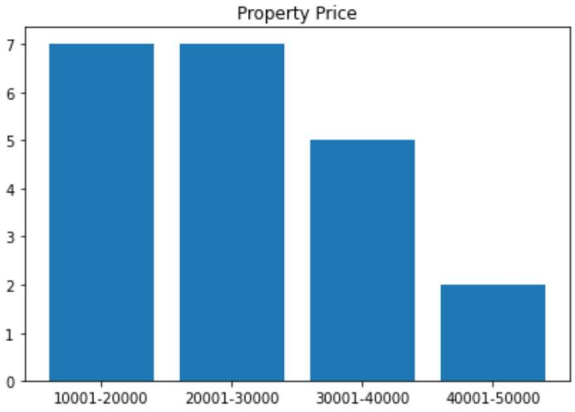
Bar Chart

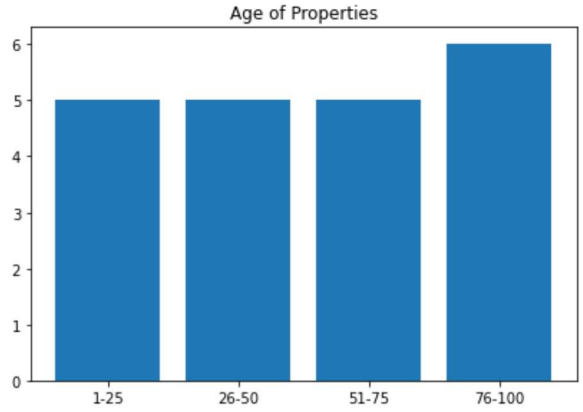
The link between a numeric and a categoric variable is portrayed by a barplot. Each categorical variable entity is represented by a bar. The numeric value is represented by the size of the bar. [1]

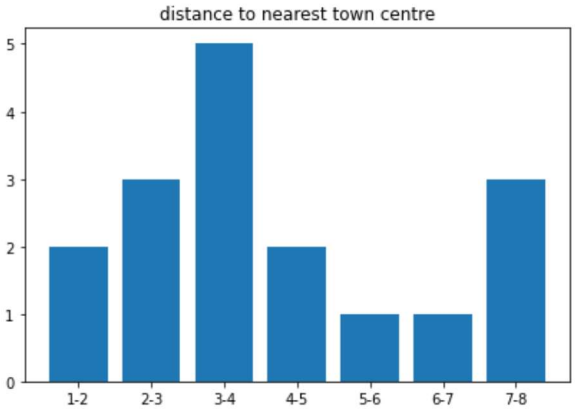
In [117…

```
fig = plt.figure(figsize = (15,10))
fig.suptitle('Bar Chart of different features', fontsize=25, color = 'g')

plt.subplot(2,2,1)
plt.title("Property Price")
plt.bar(price_label, price_freq)

plt.subplot(2,2,2)
plt.title("Age of Properties")
plt.bar(age_label, age_freq)

plt.subplot(2,2,3)
plt.title("distance to nearest town centre")
plt.bar(dist_label, dist_freq)

plt.show()
```

Bar Chart of different features







In [ ]: