Importing the necessary libraries

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the Dataset

Open/Load the csv file into a Data Frame.

```
df = pd.read_csv("Datasets/dataset.csv",names = None)
df.head()
```

	TOWN_VILLAGE	number_of_rooms	age_of_property	distance_to_nearest_town_centre	pro
0	Weston	8	50	5.21	
1	Weston	7	41	8.32	
2	Weston	8	33	5.12	
3	Weston	8	32	5.12	
4	Weston	7	28	5.12	

Get the size of the data table.

```
# dataframe.size
sizeofdf = df.size

# dataframe.shape
shapeofdf = df.shape

# printing size and shape
print("Size = {}\nShape ={}".format(sizeofdf, shapeofdf))
```

```
Size = 147
Shape =(21, 7)
```

Get the column name

```
print("The column names are:")
for col in df.columns:
  print(col)
```

```
The column names are:
TOWN_VILLAGE
number_of_rooms
age_of_property
distance_to_nearest_town_centre
property_tax_per_year
number_of_pupils_per_teacher
property_price
```

- Get the data type of the columns

```
print("Data type of each column:")
df.dtypes
```

```
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
```

Data pre-processing and Manipulation

- Checking null/missing values.

```
df.isna().sum()
```

```
TOWN_VILLAGE                       0
number_of_rooms                    0
age_of_property                    0
distance_to_nearest_town_centre    0
property_tax_per_year              0
number_of_pupils_per_teacher       0
property_price                     0
dtype: int64
```

- Checking duplicate values.

```
duplicateRows = df[df.duplicated()]
print(duplicateRows)
```

```
Empty DataFrame
Columns: [TOWN_VILLAGE, number_of_rooms, age_of_property, distance_to_nearest_town_centr
Index: []
```

There is no duplicate row in the dataframe.

- Count different column values.

```
df.nunique()
```

```
TOWN_VILLAGE                      3
number_of_rooms                   3
age_of_property                  18
distance_to_nearest_town_centre  10
property_tax_per_year            21
number_of_pupils_per_teacher      4
property_price                   21
dtype: int64
```

Describe the contents of the dataset.

The given dataset is about Property price prediction.
Each type of measurement on the dataset is a feature. The features of this dataset are:

1. TOWN_VILLAGE - Categorical (Nominal), string.
2. number_of_rooms - Quantitative (ratio), integer
3. age_of_property - Quantitative (ratio), integer
4. distance_to_nearest_town_centre - Quantitative (ratio), float
5. property_tax_per_year - Quantitative (ratio), float
6. number_of_pupils_per_teacher - Quantitative (ratio), integer
7. property_price - Quantitative (ratio), float

There are 21 rows of data and 7 columns.
There exists no missing values.

Statistical Analysis

Now we are going to do some statistical analysis.

```
1. Mean
2. Median
3. Variance
4. standard deviation
5. Range
```

Our target is to measure the central tendecy and the spread of values. This methods will be applicable for the quantitative continuous variables. They are:

```
1. number_of_rooms
2. age_of_property
3. distance_to_nearest_town_centre
4. property_tax_per_year
5. number_of_pupils_per_teacher
6. property_price
```

```
df_continuous = df[['number_of_rooms','age_of_property','distance_to_nearest_town_centre','pr
df_continuous.head()
```

	number_of_rooms	age_of_property	distance_to_nearest_town_centre	property_ta
0	8	50	5.21	
1	7	41	8.32	
2	8	33	5.12	
3	8	32	5.12	
4	7	28	5.12	

```
df_village =df.groupby('TOWN_VILLAGE')
df_village.agg(['min','max','mean'])
```

	number of rooms	age of property	distance to nearest town ce
TOWN_VILLAGE			

Mean

Here, **Mean** describes the average value of quantitative features.

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

```
Mean:
number_of_rooms                      6.619048
age_of_property                     42.333333
distance_to_nearest_town_centre      5.395714
property_tax_per_year              686.584762
number_of_pupils_per_teacher        18.476190
property_price                   27566.666667
dtype: float64
```

Median

In a sorted, ascending or descending, list of numbers, the median is the middle number.
It can be more descriptive of that data set than the mean.

```
print("Median:")
df_continuous.median()
```

```
Median:
number_of_rooms                      6.00
age_of_property                     38.00
distance_to_nearest_town_centre      5.12
property_tax_per_year              683.55
number_of_pupils_per_teacher        19.00
property_price                   22000.00
dtype: float64
```

Variance

Variance is a measure of how data points differ from the mean.

```
print("Variance:")
df_continuous.var()
```

```
Variance:
number_of_rooms                  6.476190e-01
age_of_property                  2.238333e+02
```

```
distance_to_nearest_town_centre    1.356336e+00
property_tax_per_year              5.725405e+04
number_of_pupils_per_teacher       3.461905e+00
property_price                     1.100143e+08
dtype: float64
```

Standard Deviation

The standard deviation of a data set is a measure of the magnitude of deviations between values of the observations contained.

```
print("Standard Deviation:")
df_continuous.std()
```

```
Standard Deviation:
number_of_rooms                       0.804748
age_of_property                      14.961061
distance_to_nearest_town_centre       1.164618
property_tax_per_year               239.278188
number_of_pupils_per_teacher          1.860619
property_price                    10488.771774
dtype: float64
```

```
print("Range of Values")
print(df_continuous.max() - df_continuous.min())
```

```
Range of Values
number_of_rooms                       2.00
age_of_property                      55.00
distance_to_nearest_town_centre       4.82
property_tax_per_year               705.60
number_of_pupils_per_teacher          5.00
property_price                    32600.00
dtype: float64
```

Visualizing Data

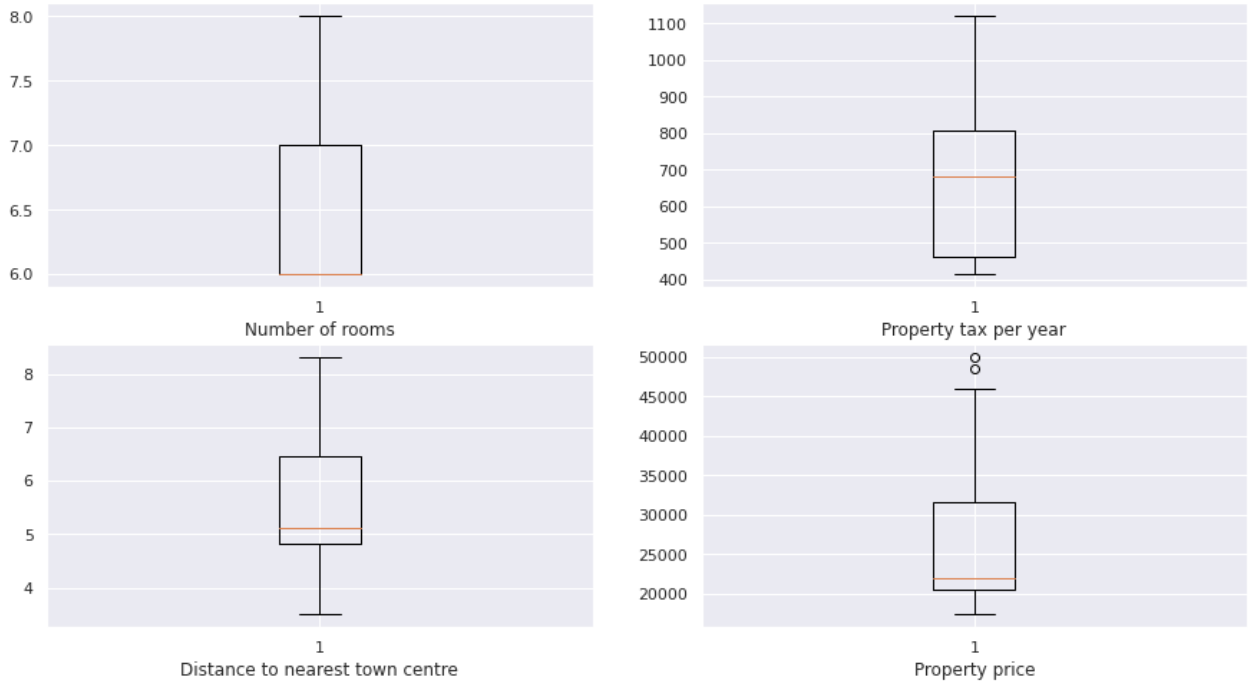
Boxplot

A boxplot is a method of summarizing a set of data which are measured on an interval scale. [7]

```
fig, ((axis1,axis2),(axis3,axis4)) = plt.subplots(2,2,sharex=False,sharey=False)
fig.set_size_inches(15,8)
fig.suptitle("Boxplot of different features",fontsize=18, color="g")
```

```
axis1.boxplot(df['number_of_rooms'],whis=1.5)
axis1.set_xlabel("Number of rooms")
axis2.boxplot(df['property_tax_per_year'],whis=1.5)
axis2.set_xlabel("Property tax per year")
axis3.boxplot(df['distance_to_nearest_town_centre'],whis=1.5)
axis3.set_xlabel("Distance to nearest town centre")
axis4.boxplot(df['property_price'],whis=1.5)
axis4.set_xlabel("Property price")
plt.show()
```

Boxplot of different features



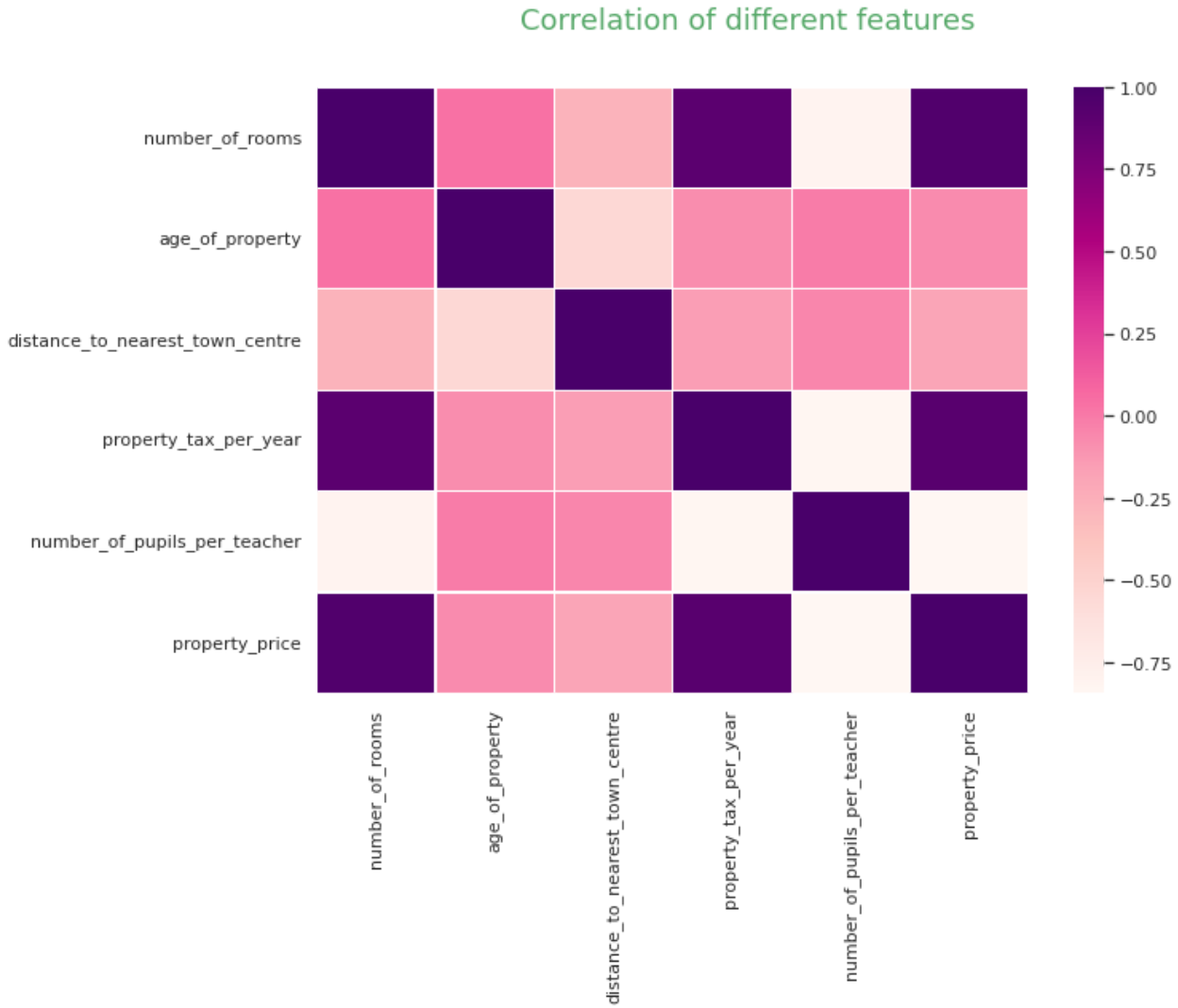
Here box plot shows visualization of 4 features: number_of_doors, property_tax_per_year, distance-to_nearest_town_centre and property_price. **What we can know from the box plots?**

```
* In the first plot, the median value for the number of doors is 7.
  It isn't skewed and there is no outliners.
* The median value for the property tax is between 800-1000. It got
  some outliers and it's slightly right skewed.
* In the boxplot for distance to nearest town centre, the median value
  lies between 2 to 3 and its greatly right skewed.
```

```
* For the last boxplot of property price, the median is between 30000-35000
  There are some outliersand this property price is greatly left skewed.
```

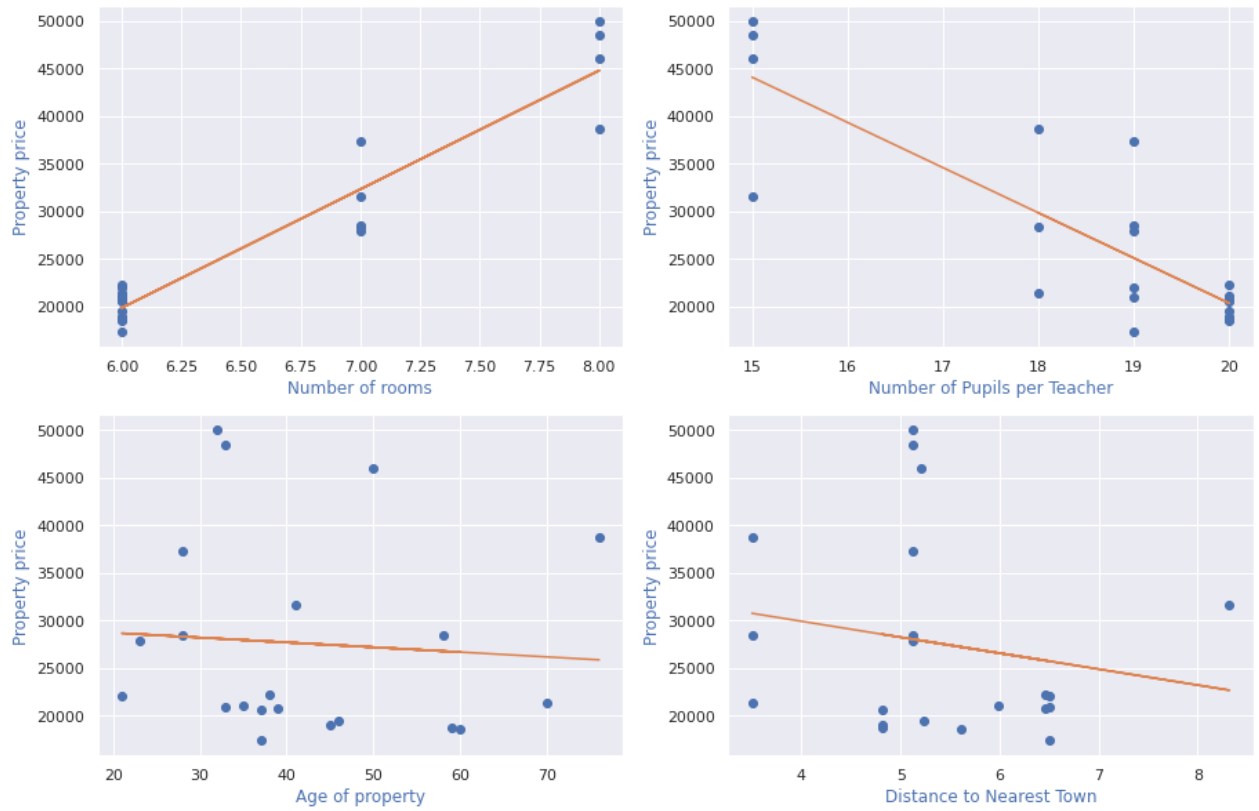
Correlation Between Features

```
correlation = df.corr()
plt.figure(figsize=(10,7))
plt.suptitle("Correlation of different features",fontsize=18, color="g")
sns.heatmap(correlation, cmap='RdPu',linewidths=0.2)
plt.show()
```



Scatter Plot

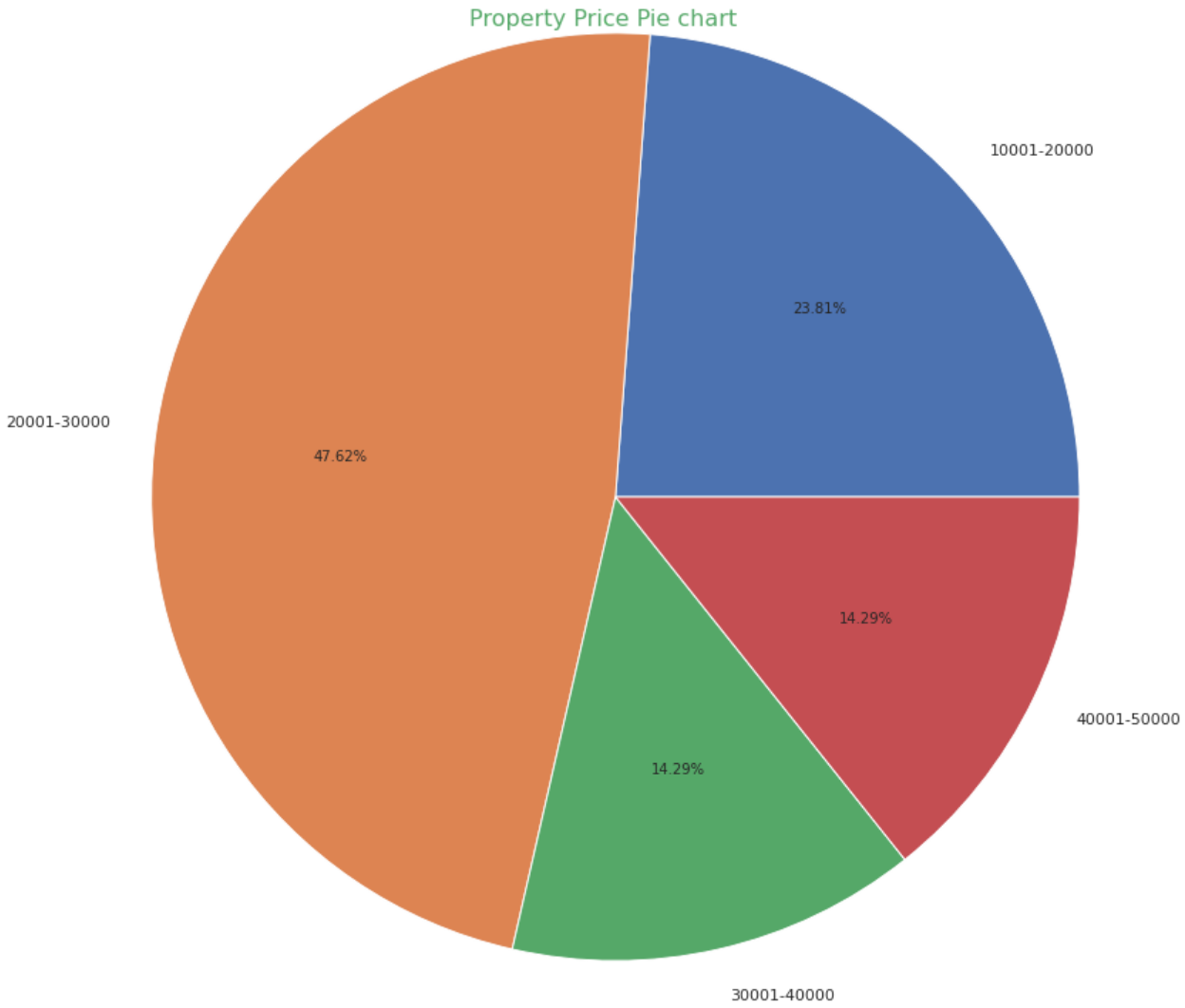
```
fig = plt.figure(figsize = (15,10))
fig.suptitle('Property Price Affected by other Features', fontsize=25, color = 'g')
y = df['property_price']

plt.subplot(2,2,1)
x = df['number_of_rooms']
plt.plot(x,y,'o')
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b)
plt.xlabel("Number of rooms", color='b')
plt.ylabel("Property price", color='b')

plt.subplot(2,2,2)
x = df['number_of_pupils_per_teacher']
plt.plot(x,y,'o')
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b)
plt.xlabel("Number of Pupils per Teacher", color='b')
plt.ylabel("Property price", color='b')

plt.subplot(2,2,3)
x = df['age_of_property']
plt.plot(x,y,'o')
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b)
plt.xlabel("Age of property", color='b')
plt.ylabel("Property price", color='b')

plt.subplot(2,2,4)
x = df['distance_to_nearest_town_centre']
plt.plot(x,y,'o')
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b)
plt.xlabel("Distance to Nearest Town", color='b')
plt.ylabel("Property price", color='b')



plt.show()
```



Property price Pie Chart

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


fig = plt.figure(figsize=(10,10))
fig.suptitle("Property Price Pie chart",fontsize=16, color = 'g')
```

```
label = ["10001-20000","20001-30000","30001-40000","40001-50000"]
plt.axis('equal')
plt.pie(p_per,labels=label,radius=1.5,autopct='%0.2f%%')

import warnings
warnings.filterwarnings('ignore')
```



Property price Bar Diagram

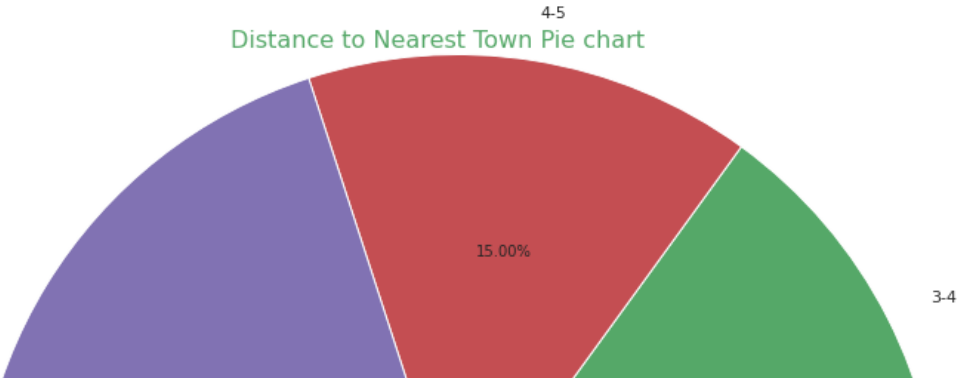
```
fig = plt.figure(figsize=(15,10))
```

```
fig.suptitle("Property Price Bar diagram",fontsize=16, color = 'g')
plt.bar(label,p_per)
plt.xlabel("Property Price")
plt.ylabel("Frequency")
plt.show()
```

Distance to nearest town Pie chart

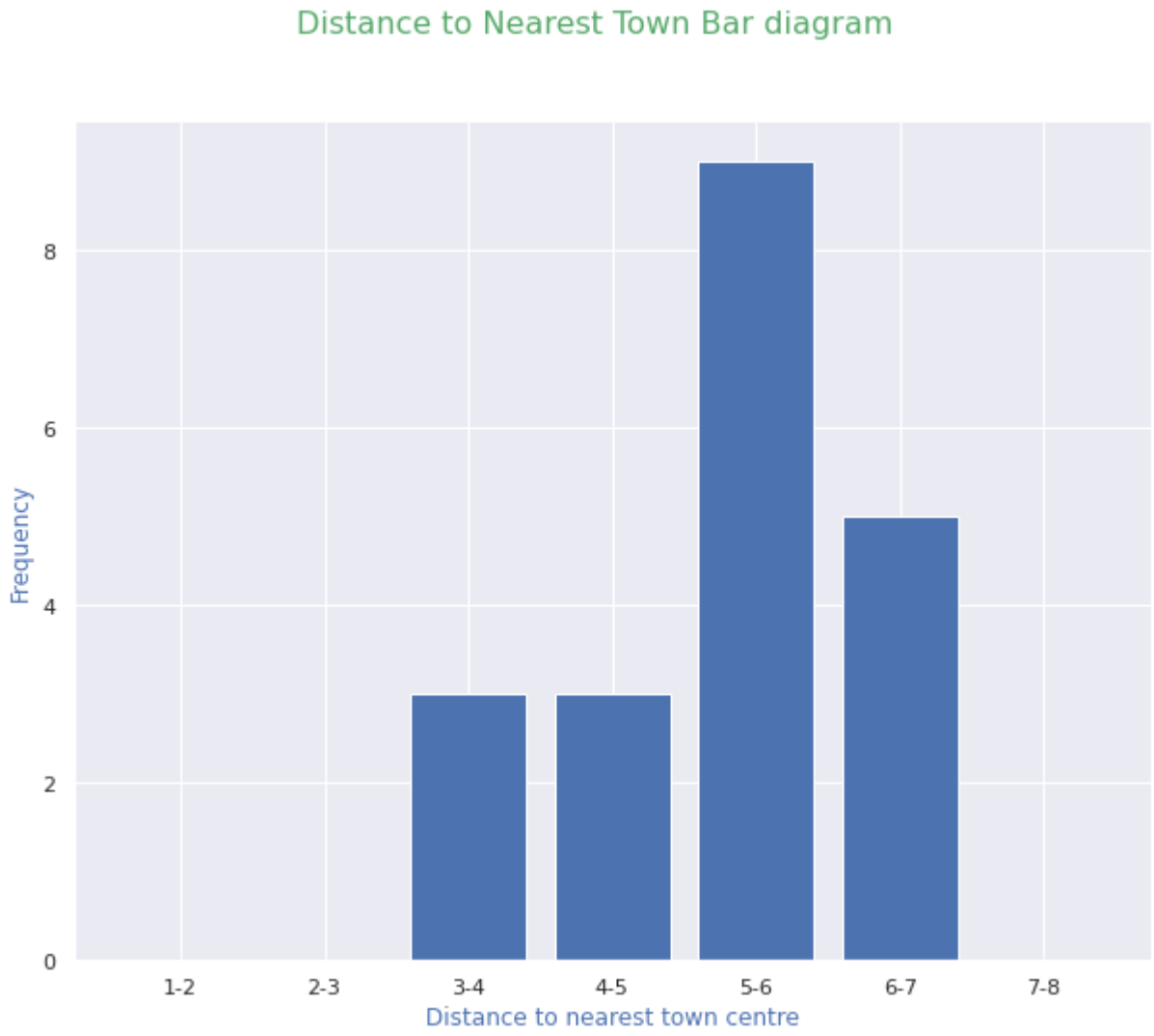
```
dist_list = data['distance_to_nearest_town_centre'].values.tolist()
dist_per = [0,0,0,0,0,0,0]
for item in range(0,len(dist_list)):
    if (dist_list[item]>7.0 and dist_list[item]<=8.0):
```

```
      dist_per[6]+=1
    if (dist_list[item]>6.0 and dist_list[item]<=7.0):
        dist_per[5]+=1
    if (dist_list[item]>5.0 and dist_list[item]<=6.0):
        dist_per[4]+=1
    if (dist_list[item]>4.0 and dist_list[item]<=5.0):
        dist_per[3]+=1
    if (dist_list[item]>3.0 and dist_list[item]<=4.0):
        dist_per[2]+=1
    if (dist_list[item]>2.0 and dist_list[item]<=3.0):
        dist_per[1]+=1
    if (dist_list[item]>1.0 and dist_list[item]<=2.0):
        dist_per[0]+=1


fig = plt.figure(figsize=(15,10))
fig.suptitle("Distance to Nearest Town Pie chart",fontsize=16, color = 'g')
label = ["1-2","2-3","3-4","4-5","5-6","6-7","7-8"]
plt.axis('equal')
plt.pie(dist_per,labels=label,radius=1.5,autopct='%0.2f%%')


warnings.filterwarnings('ignore')
```



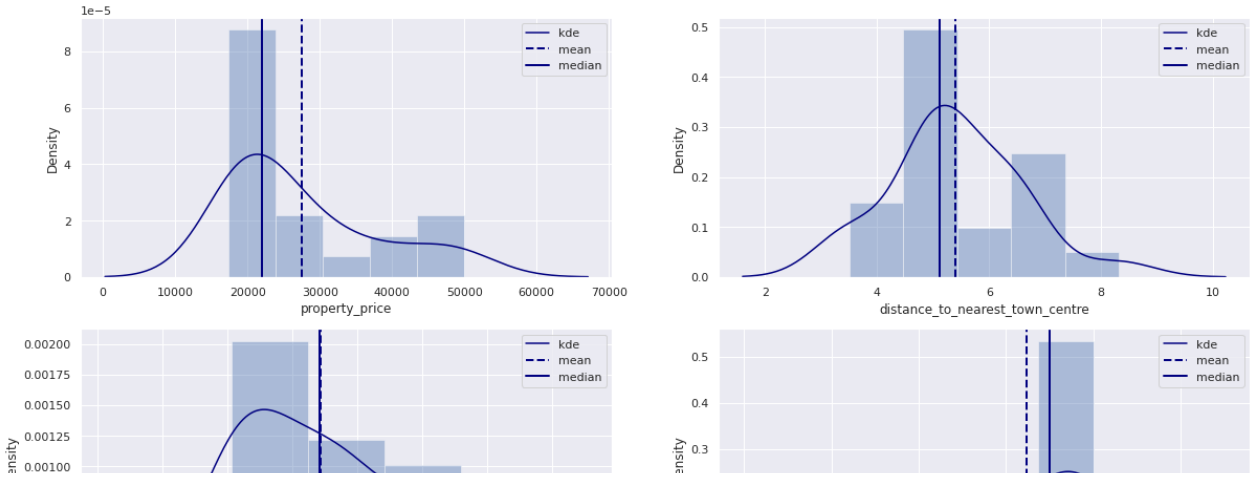
Distance to nearest town Bar Diagram

```
fig = plt.figure(figsize=(10,8))
fig.suptitle("Distance to Nearest Town Bar diagram",fontsize=16, color = 'g')
plt.bar(label,dist_per)
plt.xlabel("Distance to nearest town centre", color = 'b')
plt.ylabel("Frequency", color = 'b')
plt.show()
```



Histogram

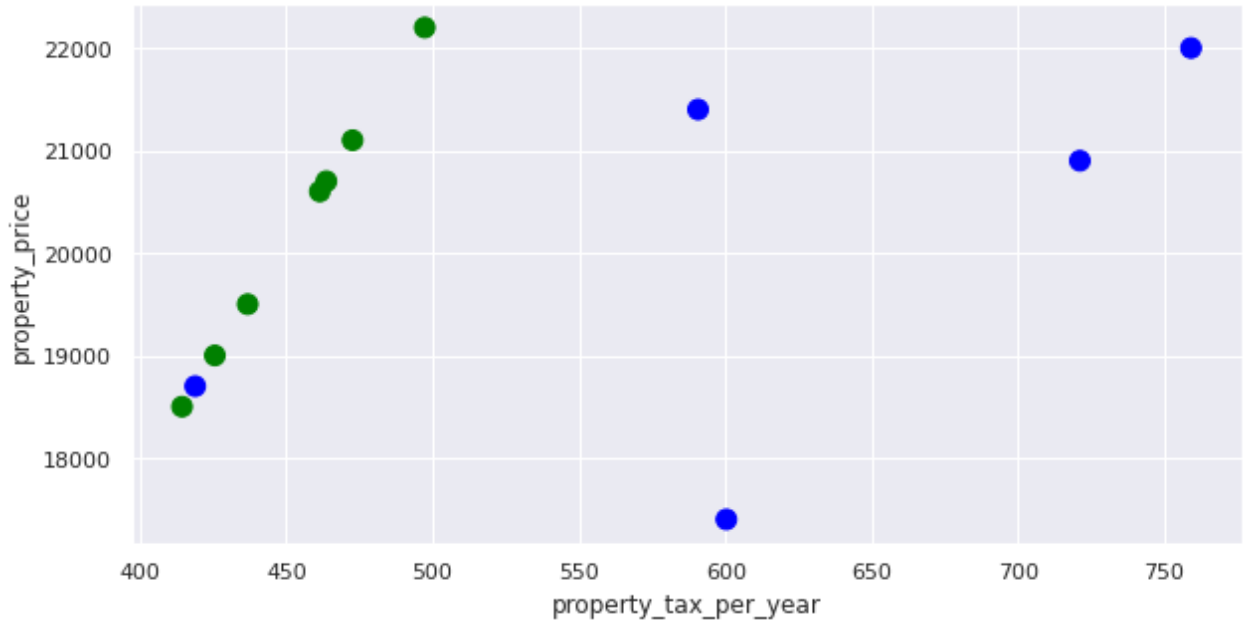
```
import warnings
warnings.filterwarnings('ignore')

fig, ((axis1,axis2),(axis3,axis4)) = plt.subplots(2,2,figsize=(20,10))

sns.distplot(df['property_price'],kde_kws={'color':'Navy'},ax=axis1);
axis1.axvline(df['property_price'].mean(),color='Navy', linestyle='dashed', linewidth=2)
axis1.axvline(df['property_price'].median(), color='Navy', linestyle='-', linewidth=2)
axis1.legend(['kde','mean','median'])

sns.distplot(df['distance_to_nearest_town_centre'],kde_kws={'color':'Navy'},ax=axis2);
axis2.axvline(df['distance_to_nearest_town_centre'].mean(),color='Navy', linestyle='dashed',
axis2.axvline(df['distance_to_nearest_town_centre'].median(), color='Navy', linestyle='-', li
axis2.legend(['kde','mean','median'])

sns.distplot(df['property_tax_per_year'],kde_kws={'color':'Navy'},ax=axis3);
axis3.axvline(df['property_tax_per_year'].mean(),color='Navy', linestyle='dashed', linewidth=
axis3.axvline(df['property_tax_per_year'].median(), color='Navy', linestyle='-', linewidth=2)
axis3.legend(['kde','mean','median'])

sns.distplot(df['number_of_pupils_per_teacher'],kde_kws={'color':'Navy'},ax=axis4);
axis4.axvline(df['number_of_pupils_per_teacher'].mean(),color='Navy', linestyle='dashed', lin
axis4.axvline(df['number_of_pupils_per_teacher'].median(), color='Navy', linestyle='-', linew
axis4.legend(['kde','mean','median'])

plt.show()
```



Correlation between Property Price and Room Number


```
def plt_function(room_number):
    temp = df[df['number_of_rooms'] == room_number]
    sns.set()
    colors = {'Weston':'red','Weymouth':'green','Wilmington':'blue'}
    temp.plot.scatter('property_tax_per_year','property_price',s = 100,figsize=(10,5),c = tem

import ipywidgets as w
w.interact(plt_function,room_number = w.IntSlider(min= 6,max = 8,step = 1))
```

room_num... 6




```
<function __main__.plt_function>
```

Report on Analysis & Summarization

The dataset has total 21 different tuples, which were the information of different town or village's property. The main focused feature was the price of the various property correlated to other features. We saw the analyzation and descriptive narrative of data with the help of different visualizations.

I can answer some questions such as:

```
- Wilmington property is the nearest to town
- Weston property price is higher
- The most property was in between the budget range (20001-30000)
- Property price and room numbers are strongly correlated
- Age of property and property price are not so much related
- pupils per teacher affects price slightly. Where there is greater number of pupils per teacher,
  education system is better there. So, price is kinda higher.
```

These are the some examples that anyone can answer by understanding my statistical ananlysis. The dataset was clean, I didn't have to put that much of effort for cleaning it. There are also many scopes of learning and predicting something from the dataset.

References

[1] Pandas Documentation pandas

[2] Numpy Documentation numpy

[3] Matplotlib Documentation matplotlib

[4] Seaborn Documentation seaborn

[5] Jupyter Widgets Documentation ipywidgets

[6] Understanding Boxplot boxplot

[7] Boxplot Definition Boxplot

[8] Distplot stackoverflow

[9] Handling missing values geeks