Cluster similar Houses

1.Data cleaning & getting rid of irrelevant information before clustering

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

```
df = pd.read_csv("/content/house_rental_data.txt")
df.drop(["Unnamed: 0"],axis=1,inplace=True)
df.head()
```

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	1177.698	2	7	2	2	2	62000
1	2134.800	5	7	4	2	2	78000
2	1138.560	5	7	2	2	1	58000
3	1458.780	2	7	3	2	2	45000
4	967.776	11	14	3	2	2	45000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 645 entries, 0 to 644
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Sqft         645 non-null    float64
 1   Floor        645 non-null    int64  
 2   TotalFloor   645 non-null    int64  
 3   Bedroom      645 non-null    int64  
 4   Living.Room  645 non-null    int64  
 5   Bathroom     645 non-null    int64  
 6   Price        645 non-null    int64  
dtypes: float64(1), int64(6)
memory usage: 35.4 KB
```

```
df.describe()
```

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	
count	645.000000	645.000000	645.000000	645.000000	645.000000	645.000000	
mean	1527.656260	5.939535	10.855814	2.837209	1.813953	1.810853	6
std	767.386531	3.884721	4.996208	1.010740	0.462364	0.683574	3
min	359.358000	1.000000	1.000000	1.000000	0.000000	0.000000	
25%	925.080000	3.000000	7.000000	2.000000	2.000000	1.000000	3
50%	1423.200000	5.000000	12.000000	3.000000	2.000000	2.000000	5

```
#df.drop(["Unnamed: 0"],axis=1,inplace=True)
```

```
df.isnull().sum()
```

```
Sqft          0
Floor         0
TotalFloor    0
Bedroom       0
Living.Room   0
Bathroom      0
Price         0
dtype: int64
```

```
len(df["Sqft"].unique())
```

```
288
```

```
df.columns
```

```
Index(['Sqft', 'Floor', 'TotalFloor', 'Bedroom', 'Living.Room', 'Bathroom',
       'Price'],
      dtype='object')
```

```
df.drop_duplicates()
```

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	1177.698	2	7	2	2	2	62000
1	2134.800	5	7	4	2	2	78000
2	1138.560	5	7	2	2	1	58000

```
df["Floor"].unique()
```

```
array([ 2,  5, 11, 10,  4,  9,  6,  3, 19,  7, 21,  8,  1, 12, 13, 18, 14,
       16, 15, 22, 17])
```

...	...	...	...	...	...	...	...

```
df["TotalFloor"].unique()
```

```
array([ 7, 14, 12,  5, 11, 19, 13,  9, 21, 15, 24,  4,  6, 25,  3, 10,  8,
       17, 20, 16, 18, 38, 27, 26,  1])
```

2.Finding the optimal value of k

3.Storing cluster to which the house belongs along with the data

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
```

```
X = df
y = df["Sqft"]
```

```
le = LabelEncoder()
X["Sqft"] = le.fit_transform(X["Sqft"])
le = LabelEncoder()
X["Price"] = le.fit_transform(X["Price"])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:115: DataConve
  y = column_or_1d(y, warn=True)
```

```
X.head()
```

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	91	2	7	2	2	2	91
1	219	5	7	4	2	2	109
3	139	2	7	3	2	2	61

```
#y = le.transform(y)
#y
```

```
col = X.columns
col
```

```
MultiIndex([(       'Sqft',),
            (      'Floor',),
            ( 'TotalFloor',),
            (    'Bedroom',),
            ('Living.Room',),
            (   'Bathroom',),
            (      'Price',)],
           )
```

```
ms = MinMaxScaler()
X = ms.fit_transform(X)
X = pd.DataFrame(X, columns = [col])
X.head()
```

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	0.317073	0.047619	0.162162	0.166667	0.5	0.4	0.598684
1	0.763066	0.190476	0.162162	0.500000	0.5	0.4	0.717105
2	0.296167	0.190476	0.162162	0.166667	0.5	0.2	0.546053
3	0.484321	0.047619	0.162162	0.333333	0.5	0.4	0.401316
4	0.240418	0.476190	0.351351	0.333333	0.5	0.4	0.401316

```
Kmeans = KMeans(n_clusters = 3)
Kmeans.fit(X)
```

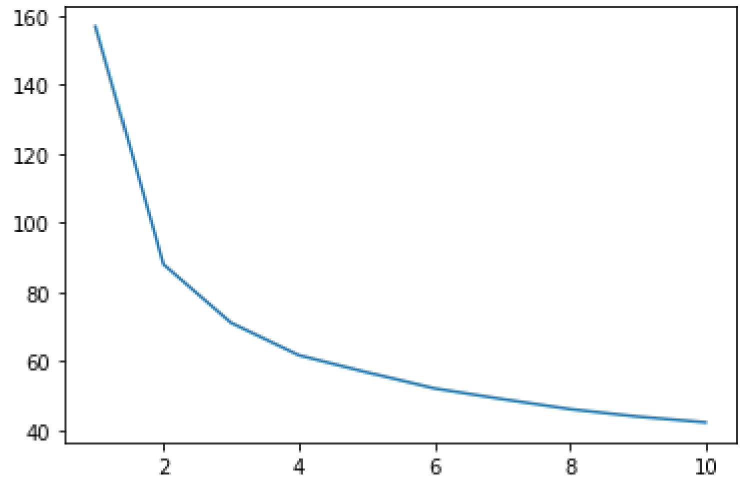
```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarnir
  FutureWarning,
KMeans(n_clusters=3)
```

```
Kmeans.cluster_centers_
```

```
array([[0.14720318, 0.23169483, 0.24149956, 0.12903226, 0.34946237,
        0.21827957, 0.26428976],
       [0.75940687, 0.30506632, 0.33901024, 0.43150685, 0.50342466,
        0.46392694, 0.75883201],
       [0.44940476, 0.17420635, 0.21936937, 0.32916667, 0.48854167,
        0.38083333, 0.44580592]])
```

```
cs=[]
for i in range(1,11):
  kmeans = KMeans(n_clusters = i,init = "k-means++" , max_iter = 300)
  kmeans.fit(X)
  cs.append(kmeans.inertia_)
plt.plot(range(1,11),cs)
plt.show
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWar
  FutureWarning,
<function matplotlib.pyplot.show>
```




```
kmeans = KMeans(n_clusters = 2)
kmeans.fit(X)
labels  = kmeans.labels_
correct_lable = sum (y==labels)
print(correct_lable , y.size)
print("Accuracy",correct_lable/float(y.size))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarnin
  FutureWarning,
0 645
Accuracy 0.0
```

```
kmeans.labels_
```

```
array([0, 1, 2, 0, 2, 0, 0, 2, 2, 0, 2, 2, 0, 2, 1, 1, 2, 2, 1, 2, 1, 1,
       2, 0, 1, 1, 1, 1, 2, 2, 0, 0, 1, 0, 2, 2, 2, 2, 1, 2, 1, 2, 2, 0,
       1, 0, 0, 1, 1, 2, 1, 0, 2, 2, 1, 1, 2, 2, 2, 0, 2, 2, 0, 0, 2, 2,
       2, 2, 2, 0, 2, 0, 1, 2, 2, 1, 1, 0, 0, 2, 0, 2, 2, 2, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
       1, 2, 1, 1, 0, 0, 1, 2, 2, 2, 2, 0, 1, 1, 1, 0, 0, 1, 2, 2, 0, 0,
       1, 2, 0, 1, 2, 1, 1, 0, 1, 0, 0, 0, 0, 2, 1, 1, 0, 0, 1, 1, 0, 2,
       2, 2, 2, 0, 0, 0, 0, 2, 2, 1, 2, 1, 1, 1, 0, 2, 1, 0, 1, 1, 2, 1,
       1, 0, 0, 0, 0, 2, 1, 2, 0, 0, 0, 2, 2, 1, 2, 2, 1, 1, 0, 1, 0, 2,
       2, 0, 2, 0, 0, 2, 0, 1, 2, 2, 0, 0, 0, 2, 1, 2, 0, 0, 0, 1, 2, 1,
       0, 1, 1, 2, 1, 2, 2, 2, 1, 0, 0, 0, 1, 1, 0, 2, 0, 0, 2, 1, 2, 0,
       1, 1, 1, 2, 1, 2, 1, 1, 0, 1, 1, 0, 1, 2, 2, 0, 1, 0, 2, 2, 0, 1,
       2, 2, 2, 2, 2, 0, 0, 1, 2, 2, 2, 0, 0, 1, 0, 2, 2, 1, 0, 0, 1, 0,
       1, 1, 2, 1, 0, 2, 1, 2, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 2, 1, 1, 2, 0, 0, 1, 2, 0,
       0, 1, 2, 1, 1, 2, 0, 2, 2, 0, 2, 2, 1, 0, 0, 2, 2, 0, 2, 2, 0, 2,
       1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 0, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0,
       0, 2, 0, 0, 2, 1, 1, 1, 1, 0, 0, 1, 1, 2, 0, 0, 2, 1, 0, 0, 2, 0,
       1, 2, 0, 0, 0, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 1, 1,
       2, 2, 2, 0, 0, 1, 1, 1, 1, 1, 2, 2, 0, 1, 2, 2, 1, 1, 2, 2, 0, 0,
       2, 1, 1, 0, 1, 1, 1, 0, 0, 2, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 0, 2,
       0, 0, 2, 2, 0, 2, 0, 0, 1, 0, 0, 2, 1, 2, 1, 2, 2, 1, 2, 0, 2, 2,
       1, 1, 2, 2, 1, 1, 2, 2, 1, 0, 2, 0, 2, 2, 2, 2, 1, 2, 1, 0, 0, 0,
       2, 1, 2, 2, 0, 1, 2, 1, 1, 0, 2, 2, 1, 2, 2, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 2, 1, 0, 2, 2, 2, 0, 0, 2, 2, 1, 0, 1, 1, 2, 2, 0, 2,
       1, 2, 2, 2, 2, 0, 1, 2, 1, 2, 0, 2, 0, 1, 2, 2, 2, 1, 2, 0, 2, 2,
       2, 0, 0, 1, 2, 0, 2, 0, 1, 2, 0, 2, 2, 2, 2, 2, 1, 0, 2, 2, 2, 0,
       2, 2, 1, 2, 2, 1, 1, 1, 1, 0, 2, 0, 0, 0, 2, 2, 2, 1, 2, 1, 0, 0,
       0, 1, 0, 2, 2, 0, 0], dtype=int32)
```

```
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
labels  = kmeans.labels_
correct_lable = sum (y==labels)
print("out of ", y.size,"the labelled samples are " ,correct_lable)
print("Accuracy",correct_lable/float(y.size))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarnir
  FutureWarning,
out of  645 the labelled samples are  0
Accuracy 0.0
```

✓ 0s completed at 10:31 PM