Devanshu Surana
PC-23, 1032210755
Panel C, Batch C1

DEC Lab Assignment 8

Problem Statement: Consider a suitable dataset and apply different clustering techniques.

Objectives: • To build the cluster using different clus--ter techniques
• To implement the k-means and Hierarchical clustering
• To check the performance of clustering algorithm

Conclusion:
In conclusion the clustering is a powerful technique in machine learning that helps uncover hidden patterns and structures within datasets. Demonstr--ated their implementation using Python programming language and the sci-kit learn library.

FAQ's

1) Differentiate betn unsupervised and supervised learning.

→ In supervised learning the algo is trained on a labeled dataset, where the input data is paired with corresponding output labels. The goal is to learn a mapping from inputs to outputs and the model makes predictions based on this learned mapping.

Page No.
Date

In unsupervised learning the algorithm is given unlabelled data and must find patterns or relationships within the data without explicit guidance. The goal is often to discover the under lying structure or distribution of the data.

Q2) What is the purpose of using cluster analysis in data science?

→ Cluster analysis in data science is used to group similar data points together on their inherent characteristics. The purpose is to uncover patterns, identify natural groupings and gain insights into the structure of the data, facilitating tasks like segmentation, anomaly detection and pattern recognition.

Q3) What are the different types of clustering algorithms available?

→ There are several types of clustering algorithms, including

1) K-means clustering
2) Hierarchical clustering
3) DBSCAN (Density based Spatial Clustering of Applications with noise).
4) Gaussian Mixture Models
5) Agglomerative Clustering
6) Affinity propagation
7) Mean shift clustering
8) Self-organizing Maps (SOM)

8/8

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
housing = pd.read_csv("HousingData.csv")
```

In [3]:
```
housing.columns
```

Out[3]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

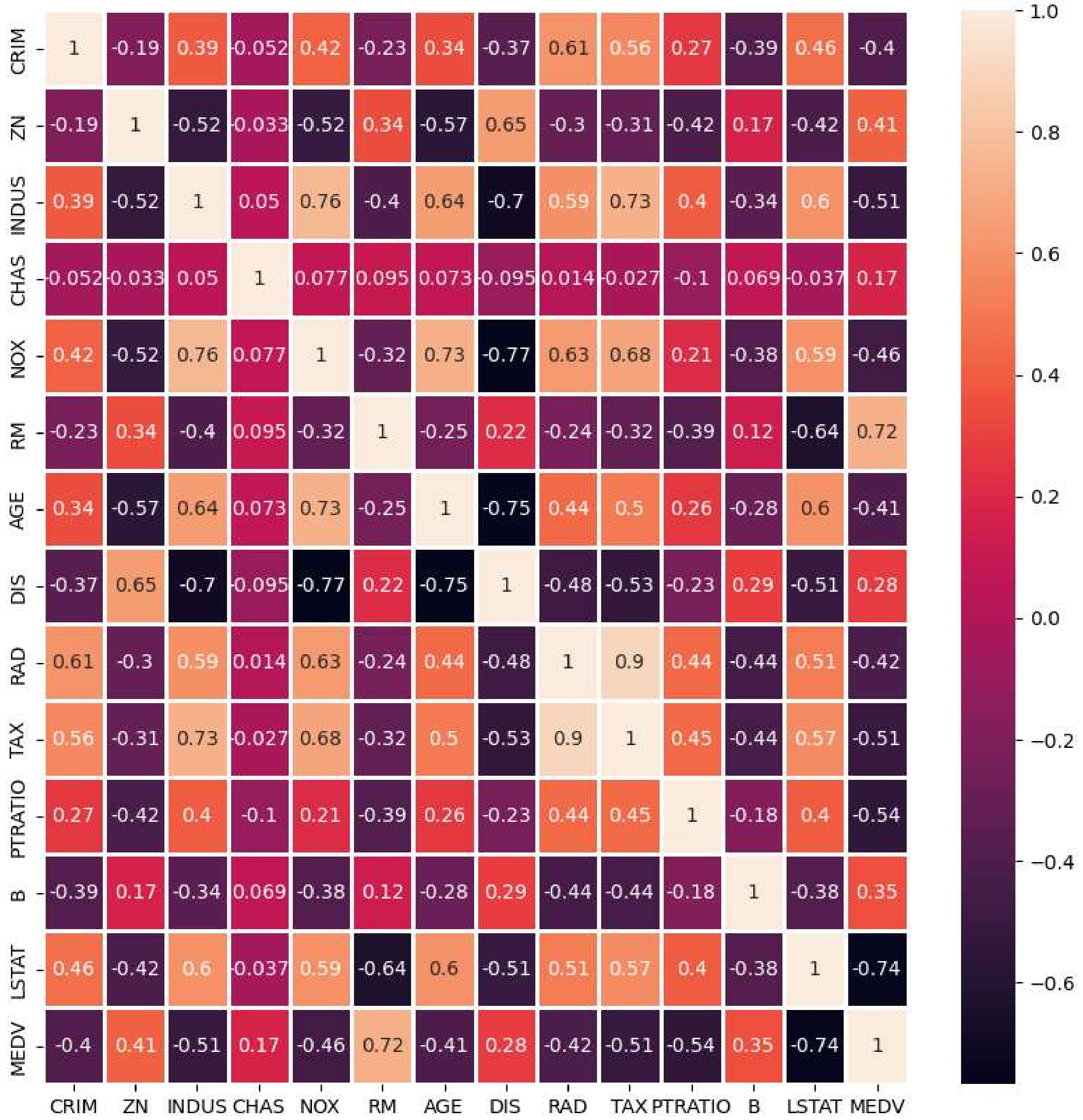
In [4]:
```
housing.dropna(inplace=True)
```

In [5]:
```
housing.isnull().sum()
```

Out[5]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:
```
plt.figure(figsize=(10, 10))
sns.heatmap(housing.corr(), annot=True , linewidths=1);
```



In [8]:
```python
from sklearn.cluster import KMeans
k = 3
```

In [9]:
```python
data_sample= housing.loc[:,['CRIM','MEDV']]
```

In [10]:
```python
model = KMeans(n_clusters=3)

model.fit(data_sample)

labels = model.predict(data_sample)
```

```
C:\Users\91902\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWa
rning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91902\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=2.
  warnings.warn(
```

In [11]:
```python
data_sample['Label_data']=labels
```

In [12]:
```
data_sample
```

Out[12]:

	CRIM	MEDV	Label_data
0	0.00632	24.0	1
1	0.02731	21.6	1
2	0.02729	34.7	0
3	0.03237	33.4	0
5	0.02985	28.7	0
...	...	...	...
499	0.17783	17.5	1
500	0.22438	16.8	1
502	0.04527	20.6	1
503	0.06076	23.9	1
504	0.10959	22.0	1

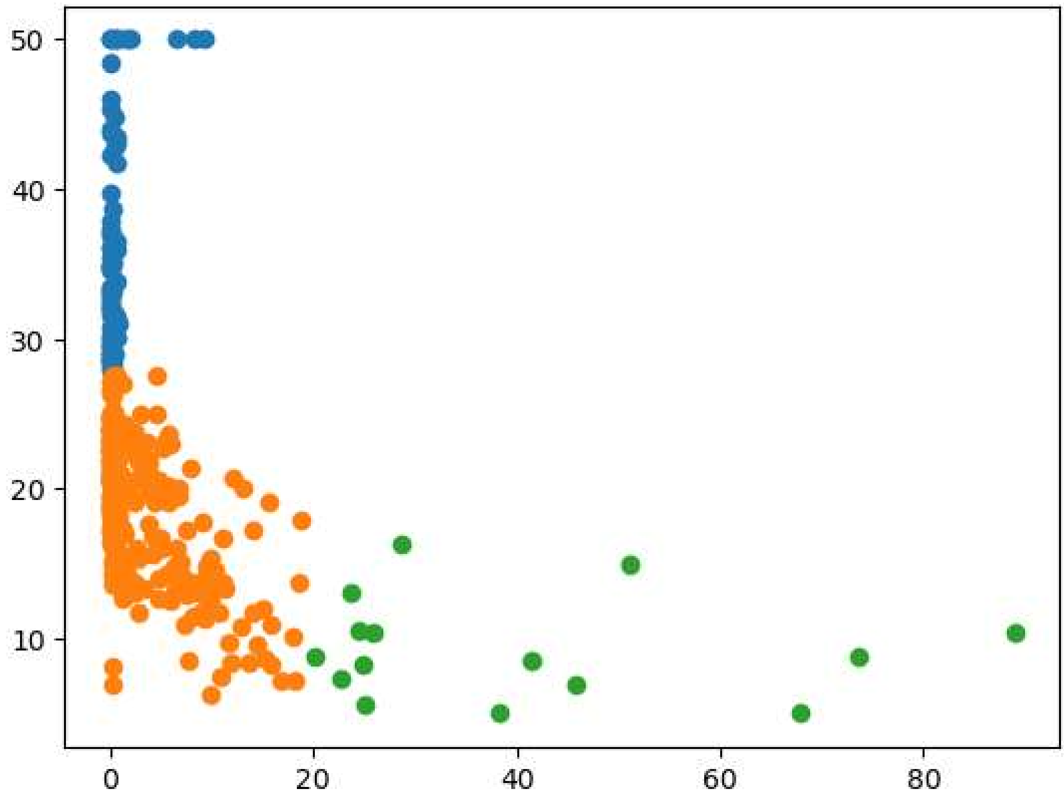
394 rows × 3 columns

In [13]:
```
clusters= {}
for i in range(k):
  clusters[i] = []

for i in range(k):
  clusters[i].append(data_sample[data_sample['Label_data'] == i])
```

In [14]:
```
print(clusters[1][0]['MEDV'])
```

```
0      24.0
1      21.6
7      27.1
8      16.5
10     15.0
       ...
499    17.5
500    16.8
502    20.6
503    23.9
504    22.0
Name: MEDV, Length: 298, dtype: float64
```

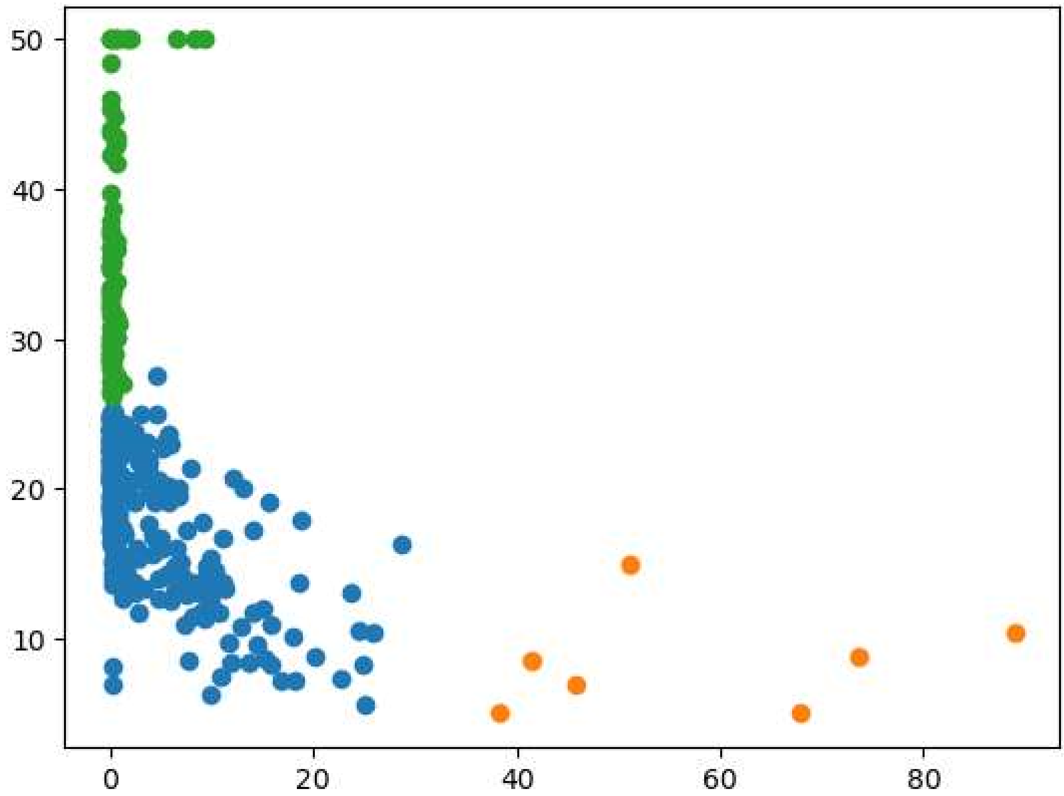
In [15]:
```
for i in range(k):
  plt.scatter(clusters[i][0]['CRIM'],clusters[i][0]['MEDV'])

plt.show()
```



In [16]:
```
from sklearn.cluster import AgglomerativeClustering
data_sample2 = data_sample
# Create Hierarchical clustering object
hierarchical = AgglomerativeClustering(n_clusters=3)

# Fit the model
hierarchical.fit(data_sample2)

# Get cluster labels
labels = hierarchical.labels_
```

In [17]:
```
data_sample2['Label_data']=labels

clusters= {}
for i in range(k):
  clusters[i] = []

for i in range(k):
  clusters[i].append(data_sample2[data_sample2['Label_data'] == i])
```

In [18]:
```
for i in range(k):
  plt.scatter(clusters[i][0]['CRIM'],clusters[i][0]['MEDV'])

plt.show()
```



In [ ]: