Assignment 11

In [1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
```

In [2]:
```
os.chdir(r"E:\GoogleDriveNew\PSU\DAAN862\Course contents\Lesson 11")
seeds = pd.read_csv("seeds_dataset.csv", header= None, names = ['area', 'premiter
                                                                'kernel_width', '
```

1. Explore the Brest Cancer Data (10 points)

In [3]:
```
seeds.head()
```

Out[3]:

	area	premiter	compactness	kernel_length	kernel_width	asymmetry_coef	groove_legth	type
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

In [4]:
```
seeds.describe()
```

Out[4]:

	area	premiter	compactness	kernel_length	kernel_width	asymmetry_coef	groove_
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.0(
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.4(
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	0.4(
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.5
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	5.0
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	5.2
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	5.8
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	6.5

In [5]:
```
seeds.type.value_counts()
```

Out[5]:
```
3    70
2    70
1    70
Name: type, dtype: int64
```

2. Use K-means clustering to group the seed data. (30 points)

In [6]:
```
X = seeds.iloc[:, 0:7]
y = seeds.type
```

In [7]:
```
y_pred= KMeans(n_clusters = 3).fit_predict(X)
metrics.homogeneity_score(y, y_pred)
```

Out[7]:
```
0.6934607041029826
```

In [8]:
```
metrics.completeness_score(y, y_pred)
```

Out[8]:
```
0.696395547296022
```

In [9]:
```
metrics.adjusted_rand_score(y, y_pred)
```

Out[9]:
```
0.7166198557361053
```

3. Use different linkage type for Hierarchical clustering to the seed data, which linkage type give the best result? (30 points)

In [10]:
```
Hierarchical_results = []
for link in ['complete', 'average', 'ward']:
    y_pred_temp = AgglomerativeClustering(
        n_clusters = 3,
        affinity = 'euclidean',
        linkage = link).fit_predict(X)
    score1 = metrics.homogeneity_score(y, y_pred_temp)
    score2 = metrics.completeness_score(y, y_pred_temp)
    score3 = metrics.adjusted_rand_score(y, y_pred_temp)
    Hierarchical_results.append([link, score1, score2, score3])
```

In [11]:
```
pd.DataFrame(Hierarchical_results, columns = ['link', 'homogenity', 'completeness
```

Out[11]:

	link	homogenity	completeness	adjusted_rand
0	complete	0.606366	0.624196	0.546135
1	average	0.713129	0.717076	0.744175
2	ward	0.726692	0.735202	0.713154

4. Use DBscan clustering group the seed data and find the best epses and min_samples value. (30 points)

In [12]:
```
DBSCAN_optimize = []
for m in np.linspace(0.5, 1.5, 11):
    for n in np.linspace(5, 15, 11):
        # print(m, n)
        y_pred_temp = DBSCAN(eps = m, min_samples= n).fit_predict(X)
        score1 = metrics.homogeneity_score(y, y_pred_temp)
        score2 = metrics.completeness_score(y, y_pred_temp)
        score3 = metrics.adjusted_rand_score(y, y_pred_temp)
        DBSCAN_optimize.append([m, n, score1, score2, score3])
```

In [13]:
```
DBSCAN_optimize = pd.DataFrame(DBSCAN_optimize, columns = ['eps', 'min_sampel', '
DBSCAN_optimize.sort_values(['homogenity', 'completeness', 'adjusted_rand'], asce
```

Out[13]:

	eps	min_sampel	homogenity	completeness	adjusted_rand
50	0.9	11.0	0.617332	0.496582	0.488853
51	0.9	12.0	0.614990	0.462807	0.429062
22	0.7	5.0	0.603244	0.444388	0.431949
36	0.8	8.0	0.593044	0.443410	0.397951
11	0.6	5.0	0.577543	0.372400	0.298657