作業五 KMmeans

> NCU MIS 106403551 呂晟維 這次作業都是寫過程，所以直接把程式碼的.ipynb複製貼上到pdf唷

1. 請用 python 依照步驟對 BreastCancer.csv 進行 KMeans 分析，過程中對所有重要程式步驟進行截圖並加以說明。 (60%)
   - (a) 將 radius_mean 及 area_mean 切為 feature diagnosis 切為 target
   - (b) 用 cluster.Means 設 n_clusters=2
   - (c) 用 fit_predict 對 feature 進行分類
   - (d) 運用 matplotlib 中的 scatter 圖， x 軸設為 radius_mean y 軸設為 area_mean，c 設為分群結果，印出分類圖形
   - (e) 移除 area_mean 中大於 2000 的資料
   - (f) 重複上述動作印出分類圖形
2. 請用 weka 對 BreastCancer.csv，進行 simplekMeans 分析 Cluster mode 設為 Use training set，numClusters 設為 2。 (40%)
   - (a) 印出 KMeans 分類結果兩類的數目
   - (b) 運用 Visualize cluster 得到與上述 python 印出之圖形(未進行刪減前)類似的結果

Python部分

a. Read & Divid dataset into Features and Target

- Feature: radius_mean, area_mean
- Target: diagnosis

In [1]:

```
import pandas as pd
df = pd.read_csv("BreastCancer.csv")
df.head()
```

Out[1]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0.
1	842517	M	20.57	17.77	132.90	1326.0	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.1

5 rows × 32 columns

In [2]:

```
len(df)
```

Out[2]:

```
569
```

Divid dataset into Features and Target. Target's data is Nominal `M` `B` , here we convert to `1` and `0` .

In [3]:

```
feature = df[["radius_mean","area_mean"]]
target  = df["diagnosis"]
```

In [4]:

```
target = target.replace('M', 1).replace('B', 0)
target.head()
```

Out[4]:

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

b. 用 cluster.Means 設 n_clusters=2

- 官方文件參數介紹: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)
- 官方文件文字介紹: https://scikit-learn.org/stable/modules/clustering.html#k-means (https://scikit-learn.org/stable/modules/clustering.html#k-means)
- 手刻+套件機器學習- K-means clustering in Python (https://medium.com/@a4793706/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-k-means-clustering-in-python-%E9%99%84%E7%A8%8B%E5%BC%8F%E7%A2%BC%E4%BB%8B%E7%B4%B9-55c19bcf2280)

KMeans初始的中心點是相當重要的，整體算法會更著中心點更近，若是因為中心點設得不好，出來的結果也會不好，且在數距複雜且龐大下，算法優化的次數會更多，所耗的時間成本也就越高。 那要解決中心點問題可以依靠k-maens++，也就是大家往上看當我們fit的時候init=k-means++ ，k-means++的方法就是讓初始中心之間的距離盡可能地遠使得加速 迭代過程的收斂。

In [5]:

```
from sklearn.cluster import KMeans
import numpy as np

kmeans = KMeans(n_clusters=2, random_state=0).fit(feature)
kmeans
```

Out[5]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

Coordinates of cluster centers.

In [6]:

```
kmeans.cluster_centers_
```

Out[6]:

```
array([[  12.59721124,  499.66696629],
       [  19.61830645, 1211.93629032]])
```

Labels of each point

In [7]:

```
kmeans.labels_
```

Out[7]:

```
array([1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0])
```

c. 用 **fit_predict** 對 **feature** 進行分類

`fit_predict` : Compute cluster centers and predict cluster index for each sample. Result is **same as** `kmeans.labels_` .

In [8]:

```
predict = kmeans.fit_predict(feature)
predict
```

Out[8]:

```
array([1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0])
```

d. 運用 matplotlib 中的 scatter 圖， x 軸設為 radius_mean y 軸設為 area_mean，c 設為分群結果，印出分類圖形

官網matplotlib.pyplot.scatter: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html)

- A scatter plot of y vs x with varying marker size and/or color.
- 輸入X和Y作為每個點的location
- c是點的顏色，這裡用1和0表示

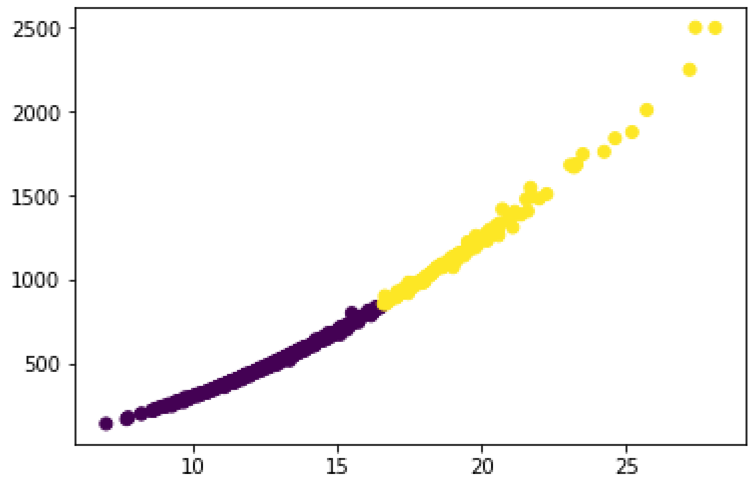
In [10]:

```
import matplotlib.pyplot as plt

plt.scatter(feature['radius_mean'], feature['area_mean'], c = predict)
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x1eb57b33908>
```



e. Remove area_mean 中大於 2000 的資料

In [11]:

```
feature = feature[feature.area_mean <= 2000]
```

f. 重複上述動作印出分類圖形

We re-fit the KMeans model with new data. And plot the clusters with scatter plot.

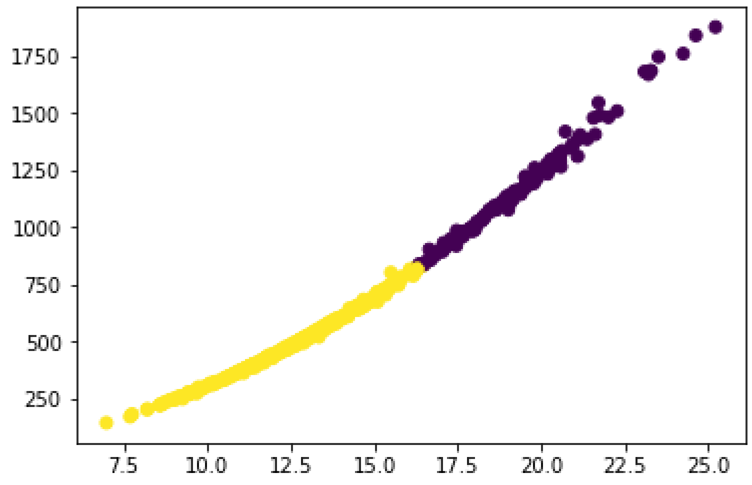
In [12]:

```
kmeans = KMeans(n_clusters=2, random_state=0).fit(feature)
predict = kmeans.fit_predict(feature)

plt.scatter(feature['radius_mean'], feature['area_mean'], c = predict)
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x1eb57c93748>
```



Now we have another question. **Why the color of 2 clusters changed?**

By checking the prediction array, we find out the **new** array starts with `0`, meaning color `yellow`. However the **original** clustering array starts with `1`, meaning color `purple`.

By the way, the second **cluster center** changes **a lot**, which points that KMeans is affected by **extreme values** easily.

In [13]:

```
kmeans.cluster_centers_
```

Out[13]:

```
array([[  19.32      , 1169.66065574],
       [  12.57993002,  498.13386005]])
```

In [14]:

```
kmeans.labels_ # equils to the result array 'predict'
```

Out[14]:

```
array([0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1])
```

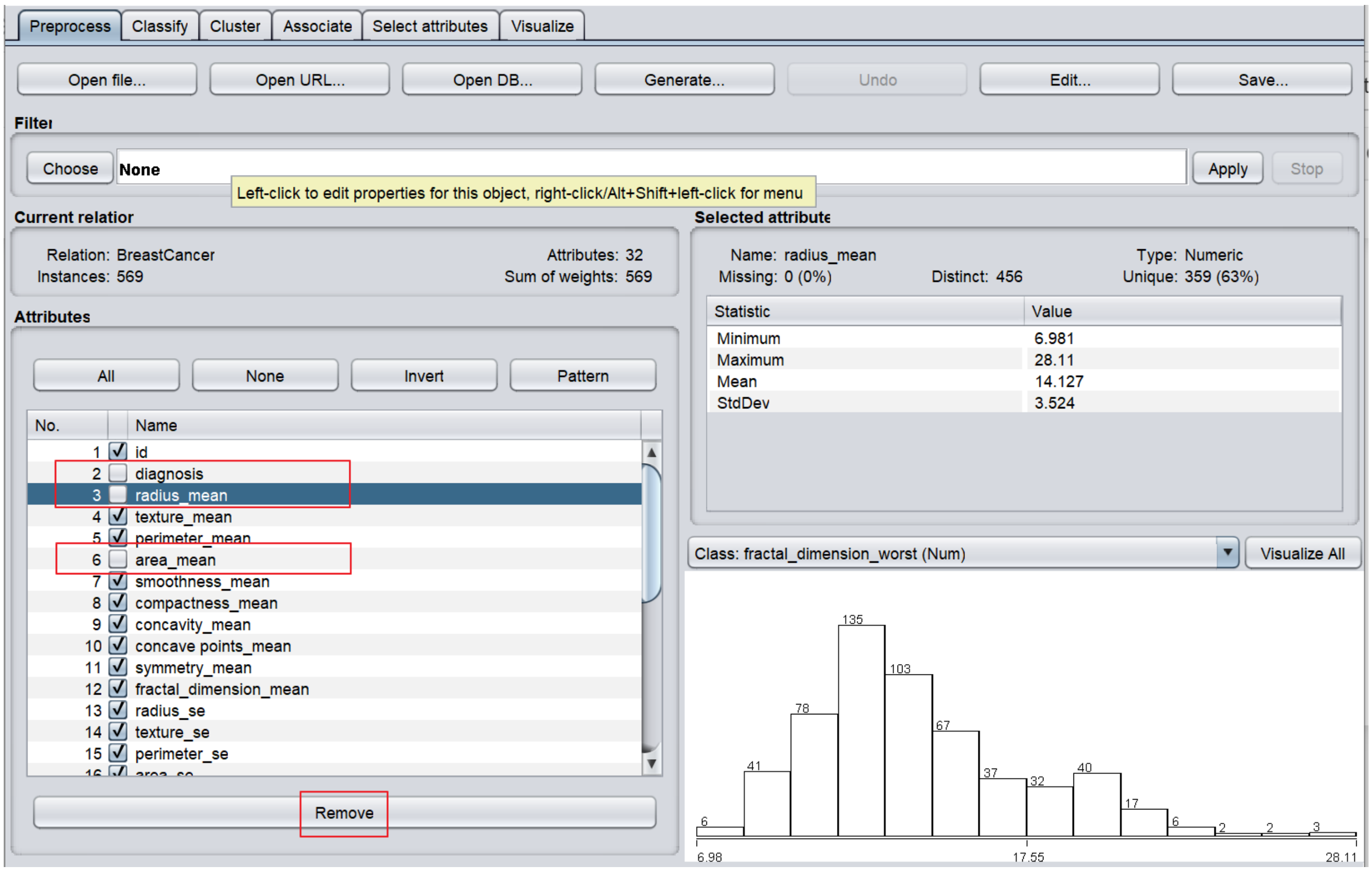
Weka部分

請用 weka 對 BreastCancer.csv，進行 simplekMeans 分析 Cluster mode 設為 Use training set，numClusters 設為 2。 (40%)

(a) 印出 KMeans 分類結果兩類的數目

首先我們進行資料的前處理，匯入csv檔，進入Processing頁面，勾選除了要用到的3個以外的全部屬性，然後刪除他們。

- Feature: radius_mean, area_mean
- Target: diagnosis



Current relation
Relation: BreastCancer
Instances: 569
Attributes: 32
Sum of weights: 569

Selected attribute
Name: radius_mean
Missing: 0 (0%)
Distinct: 456
Type: Numeric
Unique: 359 (63%)

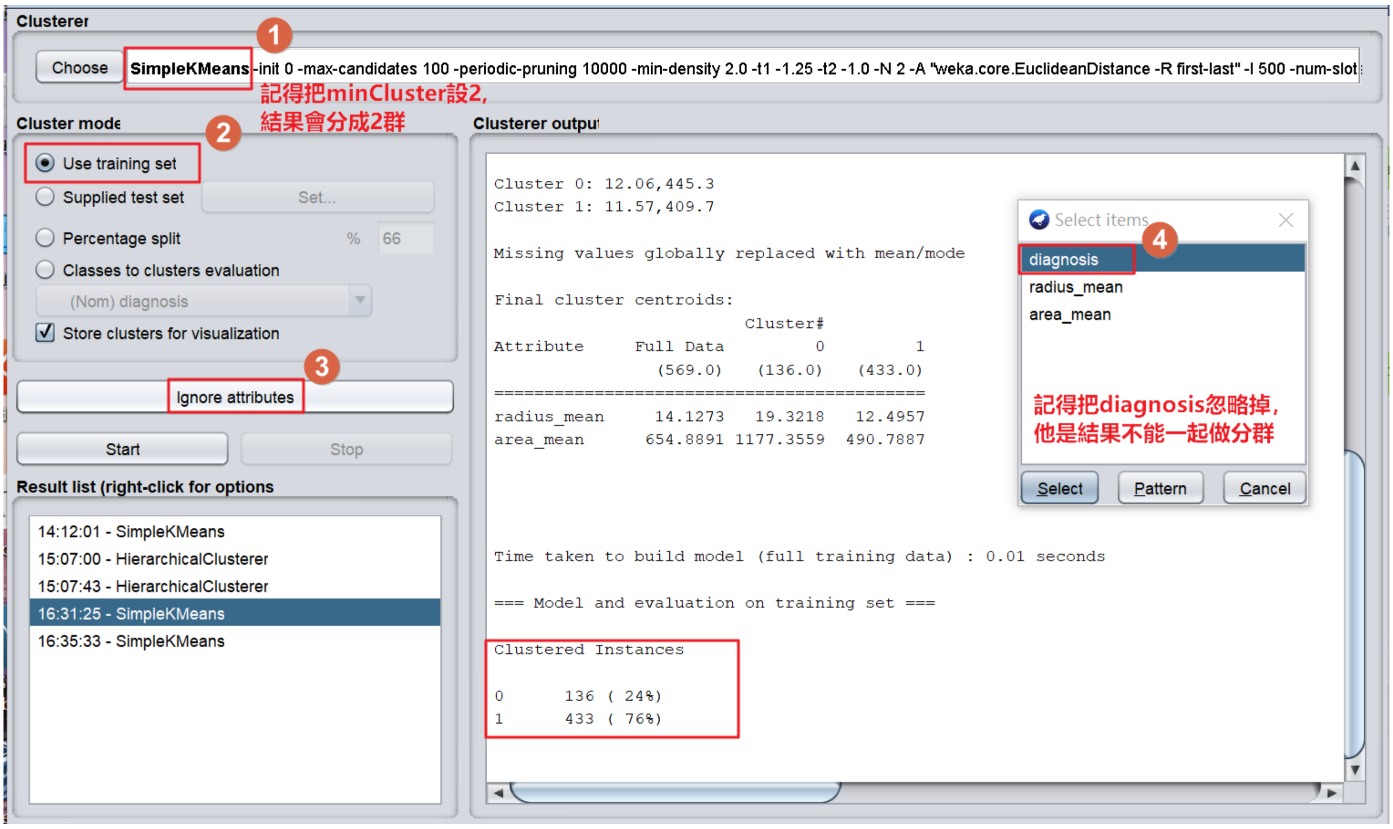
Statistic	Value
Minimum	6.981
Maximum	28.11
Mean	14.127
StdDev	3.524

Class: fractal_dimension_worst (Num)

No.		Name
1	✓	id
2		diagnosis
3		radius_mean
4	✓	texture_mean
5	✓	perimeter_mean
6		area_mean
7	✓	smoothness_mean
8	✓	compactness_mean
9	✓	concavity_mean
10	✓	concave points_mean
11	✓	symmetry_mean
12	✓	fractal_dimension_mean
13	✓	radius_se
14	✓	texture_se
15	✓	perimeter_se

接著進入cluster頁面，依照下圖步驟選擇 `SimpleKmeans` 方法並把numClusters設為 2，因為我們要分兩群；Mode則使用training set；然後ignore掉diagnosis屬性即可執行。

- 決定分成幾群比較好可以參考這則影片: [StatQuest: K-means clustering (https://youtu.be/4b5d3muPQmA)](https://youtu.be/4b5d3muPQmA)



SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slot

Use training set [✓]
Store clusters for visualization [✓]

Select items: [✓ diagnosis], radius_mean, area_mean

Result list (right-click for options)
14:12:01 - SimpleKMeans
15:07:00 - HierarchicalClusterer
15:07:43 - HierarchicalClusterer
16:31:25 - SimpleKMeans
16:35:33 - SimpleKMeans

Clusterer output

```
Cluster 0: 12.06,445.3
Cluster 1: 11.57,409.7

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute     Full Data         0          1
                (569.0)   (136.0)    (433.0)
============================================
radius_mean     14.1273   19.3218    12.4957
area_mean      654.8891 1177.3559   490.7887




Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       136 ( 24%)
1       433 ( 76%)
```

輸出結果如下，第一類有136筆資料，是右上角的那群；第二類有433筆資料，是圖形化左下角的那一群。

- **離中心點的平方和 (模型好不好的依據)**

```
Number of iterations: 12
Within cluster sum of squared errors: 8.877720953704475
```

- **Cluster**的起始中心位置

```
Initial starting points (random):

Cluster 0: 12.06,445.3
Cluster 1: 11.57,409.7
```

- 兩群**Cluster**的內容

```
Final cluster centroids:
                                  Cluster#
Attribute      Full Data          0          1
                 (569.0)    (136.0)    (433.0)
===========================================
radius_mean      14.1273    19.3218    12.4957
area_mean       654.8891  1177.3559   490.7887
```

- 兩群**Cluster**的個數

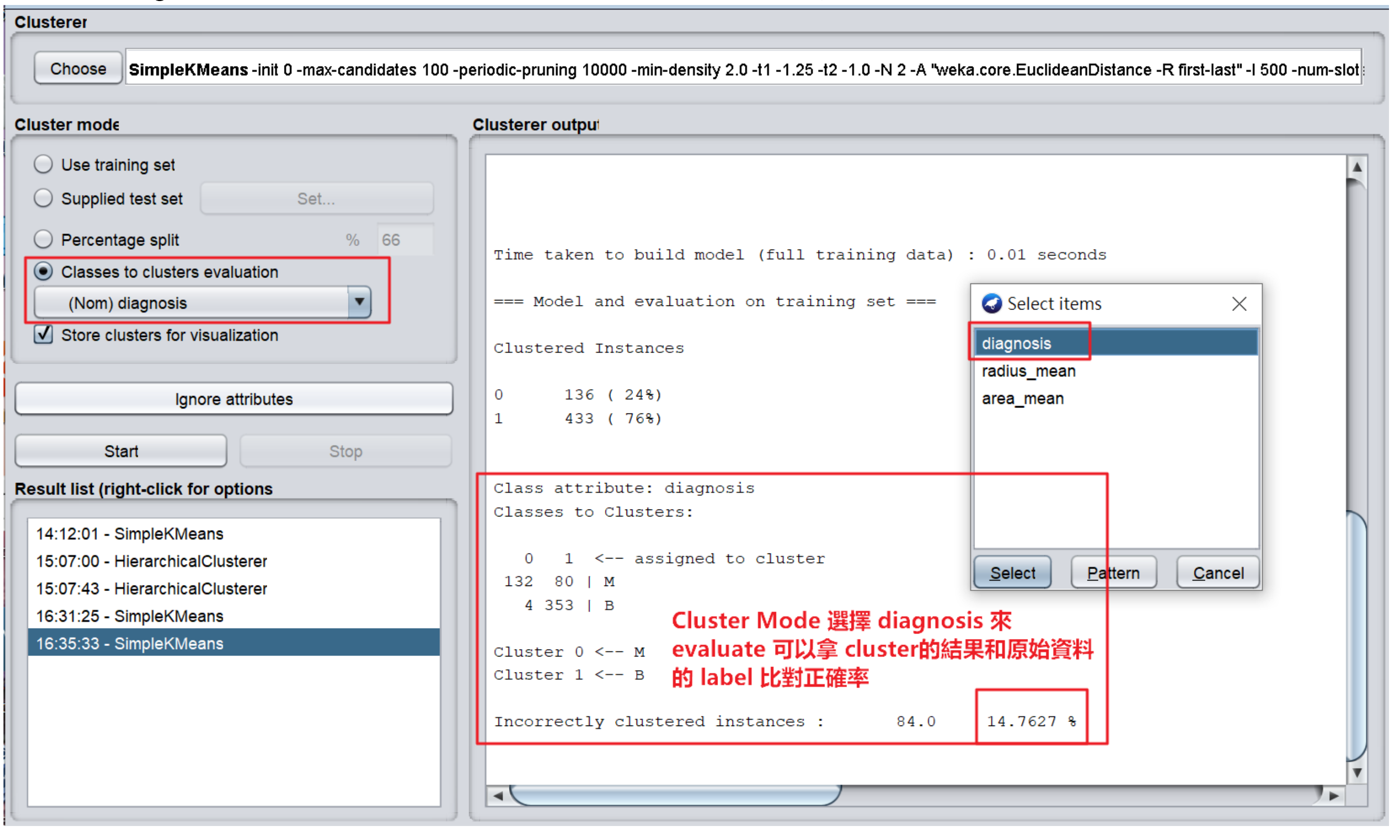
```
=== Model and evaluation on training set ===

Clustered Instances

0       136 ( 24%)
1       433 ( 76%)
```

#補充

如果選擇diagnosis當作evaluation的對象，可以查看cluster的正確率唷!



Clusterer

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slot

Cluster mode

Use training set
Supplied test set
Percentage split % 66
[✓ Classes to clusters evaluation]
(Nom) diagnosis
[✓ Store clusters for visualization]

Result list (right-click for options)

14:12:01 - SimpleKMeans
15:07:00 - HierarchicalClusterer
15:07:43 - HierarchicalClusterer
16:31:25 - SimpleKMeans
16:35:33 - SimpleKMeans

Clusterer output

```
Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      136 ( 24%)
1      433 ( 76%)


Class attribute: diagnosis
Classes to Clusters:

  0   1  <-- assigned to cluster
 132  80 | M
   4 353 | B

Cluster 0 <-- M
Cluster 1 <-- B

Incorrectly clustered instances :      84.0      14.7627 %
```

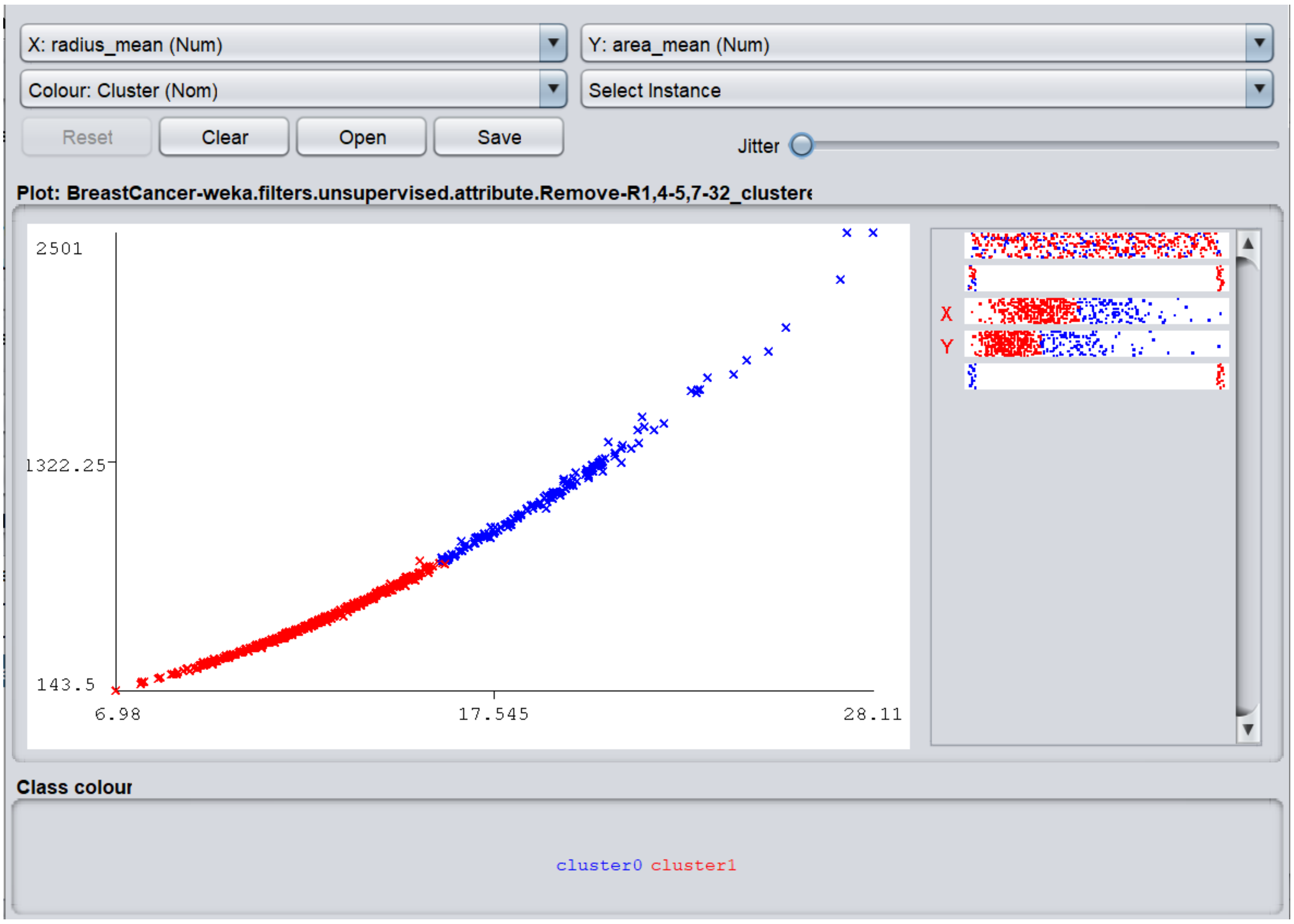
Select items

[✓ diagnosis]
radius_mean
area_mean

Cluster Mode 選擇 diagnosis 來 evaluate 可以拿 cluster的結果和原始資料的 label 比對正確率

(b) 運用 Visualize cluster 得到與上述 python 印出之圖形(未進行刪減前)類似的結果

右鍵點選visualize cluster，把x軸設成radius_mean，y軸設成area_mean，即可得出2個的群集散佈圖。



Plot: BreastCancer-weka.filters.unsupervised.attribute.Remove-R1,4-5,7-32_cluster

Class colour

cluster0 cluster1