ECT HW5

Python part:

前置確認 ：

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

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...
...	...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...

569 rows × 32 columns

➔ 先導入資料集

```
# check missing values
df.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
```

➔ 確認資料完整性，若有 missing values，可能要進行處理

(a)

```
# x:input
x = df.loc[:,["radius_mean","area_mean"]]
print(x)
# KMeans is unsupervised so it don't need the output
# # y:output
# y = df.loc[:,["diagnosis"]]
# print(y)
```

```
     radius_mean  area_mean
0          17.99     1001.0
1          20.57     1326.0
2          19.69     1203.0
3          11.42      386.1
4          20.29     1297.0
..           ...        ...
564        21.56     1479.0
565        20.13     1261.0
566        16.60      858.1
567        20.60     1265.0
568         7.76      181.0

[569 rows x 2 columns]
```

➔ 因為 KMeans 是 Unsupervised，所以只有 input，根據題意把 input 切為 radius_mean, area_mean 兩部分。

(b)

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

➔ 先把之後會用到的套件 import 進來

```
# KMeans
model = KMeans(n_clusters=2) # 創建model
```

➔ 初始化 Kmeans model 時，設定參數 n_cluster = 2, 代表要分成兩群

- 其他像是迭代次數、初始亂數都有參數可以調整，在此沒有特別要求就使用默認參數。

(c)

```
model.fit_predict(x) # fir進去，計算各clustering的中心，並把各個instance分給clustering
classificationResult = model.labels_ # clustering's result
```

➔ 使用 fit_predict(x)對 x 進行分群，會先計算 clustering 的中心再把每個 instance 分給各個 clustering

➔ 使用 model.labels_ 把分群結果保存下來

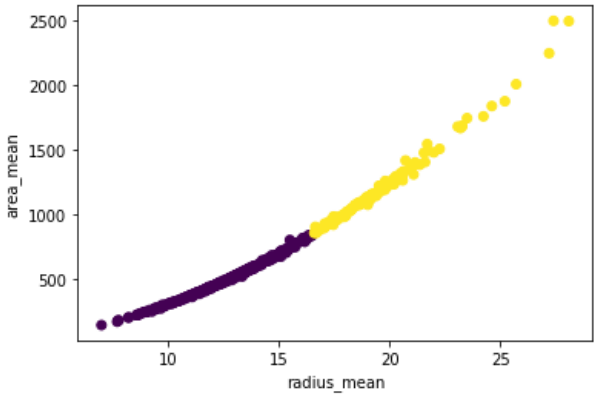
```
[1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 0
 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 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 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 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 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 0 1 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 0 1 1 0 0 0 1 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 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 1 1 1 1 1 0]
```

■

■ 可以看到分群完後分成 0、1 兩群

(d)

```
# plt繪圖
plt.scatter(x['radius_mean'], x['area_mean'], c=classificationResult) # 散佈圖
plt.xlabel('radius_mean') # X軸取名字
plt.ylabel('area_mean') # Y軸取名字
plt.show() # 畫出來
```



➔ 使用 matplotlib.pyplot 中的 scatter 來繪製散佈圖，前兩個參數為 X 軸 Y 軸、最後一個參數「c」代表顏色，我們把剛剛分群結果的「0、1」列表丟進去，讓他繪製不同的顏色。

➔ Xlabel、ylabel 就只是單純地幫兩軸取名字

➔ Show()把圖畫出來，call 過一次後之後不寫也可以畫出來

(e)

```
# re-step for area_mean <= 2000
print( (x["area_mean"] <= 2000) ) # 查看設定條件後的格式
```

```
0       True
1       True
2       True
3       True
4       True
       ...
564     True
565     True
566     True
567     True
568     True
Name: area mean, Length: 569, dtype: bool
```

➔ 先試試看直接用條件切割，會發現 values 變成 True、False，因此要做修正

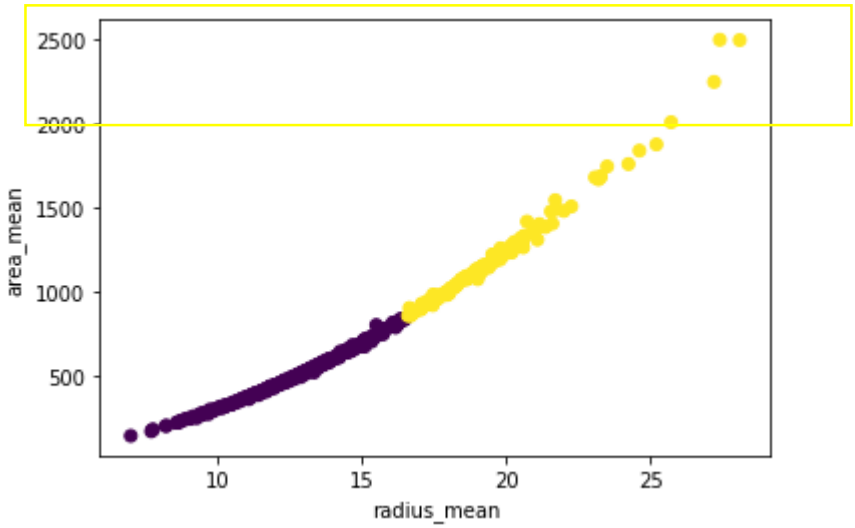
```
x = x.loc[x["area_mean"] <= 2000] # 題目要求area_mean <= 2000
print(x)
```

```
     radius_mean  area_mean
0          17.99     1001.0
1          20.57     1326.0
2          19.69     1203.0
3          11.42      386.1
4          20.29     1297.0
..           ...        ...
564        21.56     1479.0
565        20.13     1261.0
566        16.60      858.1
567        20.60     1265.0
568         7.76      181.0

[565 rows x 2 columns]
```

➔ 用 loc()函數把結果為 True 的保留下來，False 的去除

➔ 可發現處理前有 569 筆資料，處理後有 565 筆資料，刪除了 4 筆

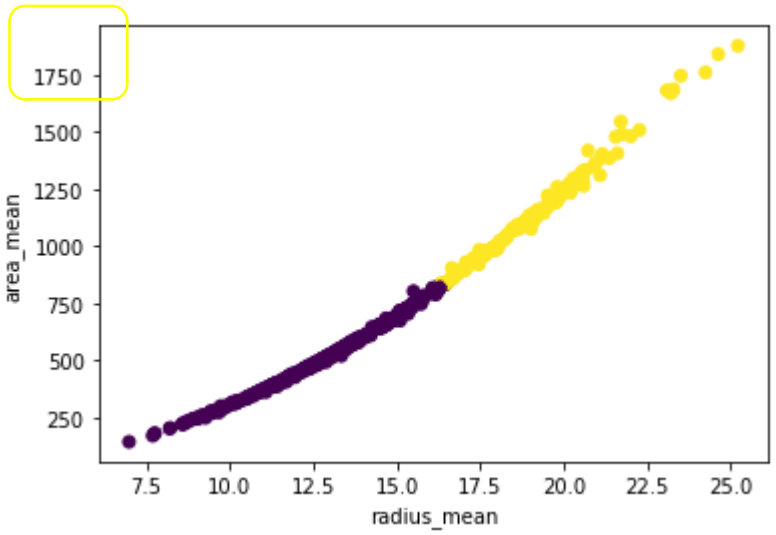


➔ 而原始圖形中的確有 4 筆資料在大於 2000，的確相符。

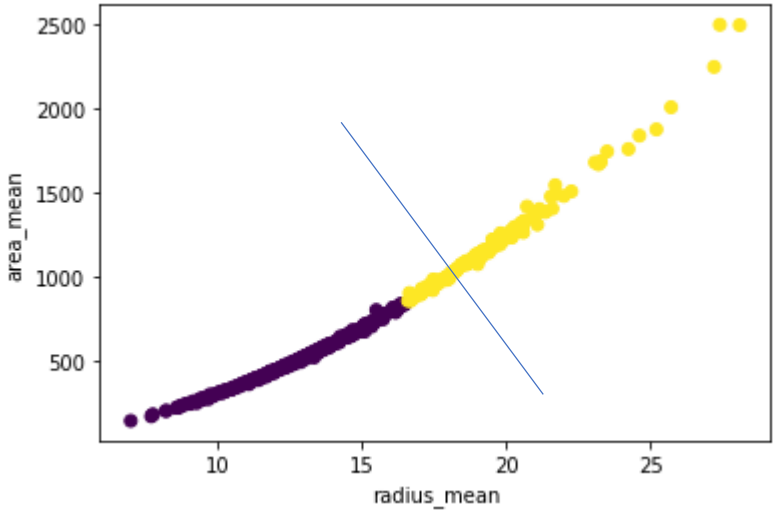
(f)

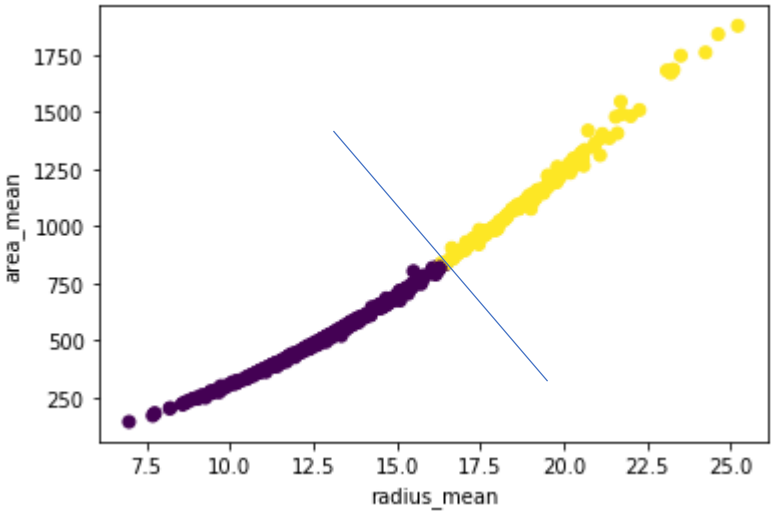
```
# KMeans
model = KMeans(n_clusters=2)
model.fit_predict(x)
classificationResult = model.labels_

# plt繪圖
plt.scatter(x['radius_mean'], x['area_mean'], c=classificationResult)
plt.xlabel('radius_mean')
plt.ylabel('area_mean')
plt.show()
```



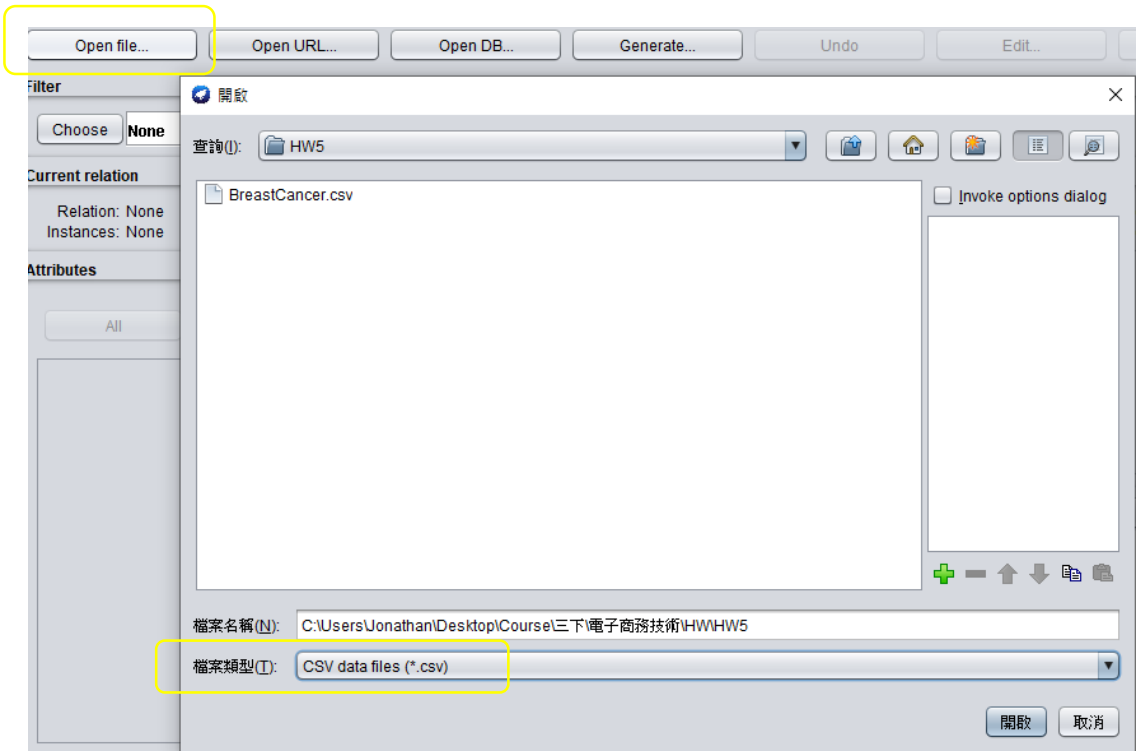
➔ 重複之前的動作，先用 KMeans 處理過後，再把結果畫出來。可以注意圖中 Y 軸已經沒有 2000 以上的數字了



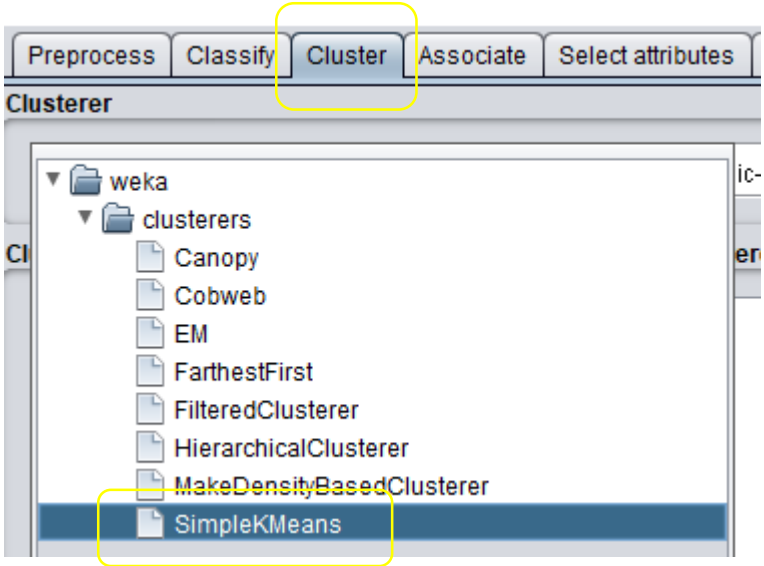


➔ 比較處理前、處理後的結果，可以看出把 2000 以上的偏差值處理掉後，clustering 的結果更為平均，基本上就是一半一半了。因此可以認為這一步驟是有一定程度的意義。

Weka part:

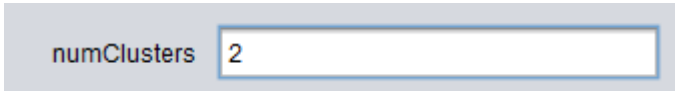


➔ 先用 open file 開啟檔案，注意因為這次是給 csv，所以檔案類型部分要改成 CSV data files



➔ 因為 KMeans 是 clusteing 的方式，所以去 Cluster 部分尋找方法使用，依照題目要求選擇 SimpleKMeans

➔ 依照題意設定 Use training set



➔ 依照題意設定 numClusters = 2

(a)

```
Final cluster centroids:
                                        Cluster#
Attribute                    Full Data          0          1
                               (569.0)    (358.0)    (211.0)
=============================================================
id                       30371831.4323 26472172.0112 36988315.1896
diagnosis                            B          B          M
radius_mean                    14.1273    12.1454    17.4899
texture_mean                   19.2896    17.9154    21.6213
perimeter_mean                  91.969    78.0668   115.5567
area_mean                      654.8891   462.7017   980.9701
smoothness_mean                 0.0964     0.0925     0.1029
compactness_mean                0.1043       0.08     0.1456
concavity_mean                  0.0888      0.046     0.1614
concave points_mean             0.0489     0.0257     0.0882
symmetry_mean                   0.1812     0.1742     0.1931
fractal_dimension_mean          0.0628     0.0629     0.0627
radius_se                       0.4052     0.2851     0.6089
texture_se                      1.2169     1.2229     1.2067
perimeter_se                    2.8661     2.0063     4.3248
area_se                        40.3371    21.2133    72.7841
smoothness_se                   0.007     0.0072     0.0068
compactness_se                 0.0255     0.0214     0.0324
concavity_se                   0.0319      0.026      0.042
```

➔ 比較詳細的結果，可以看出 cluster 0 有 358 個 instance，cluster 1 有 211 個 instance

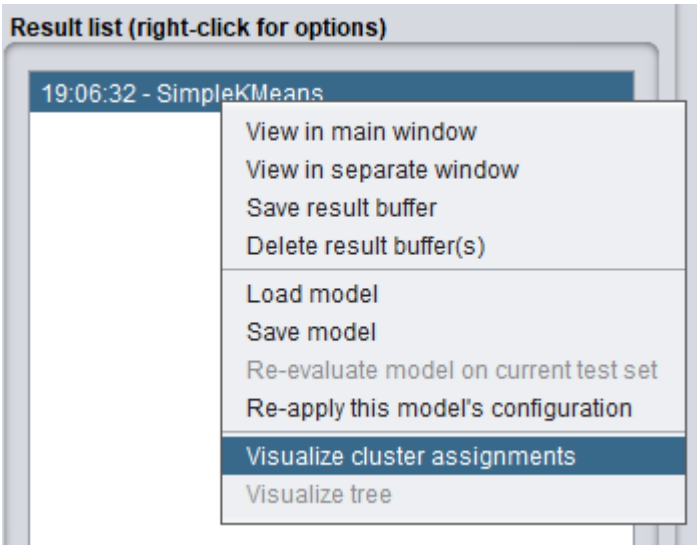
```
Clustered Instances

0       358 ( 63%)
1       211 ( 37%)
```

➔ 比較簡潔的結果，可以看出各自的比例。3

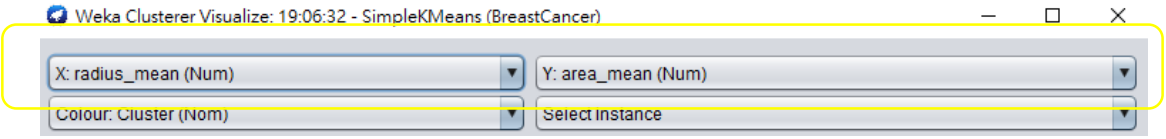
➔ Hint：此結果並未把大於 2000 的 instance 刪除，因為我觀察到(b)小題要求的也是刪減前的結果。

(b)

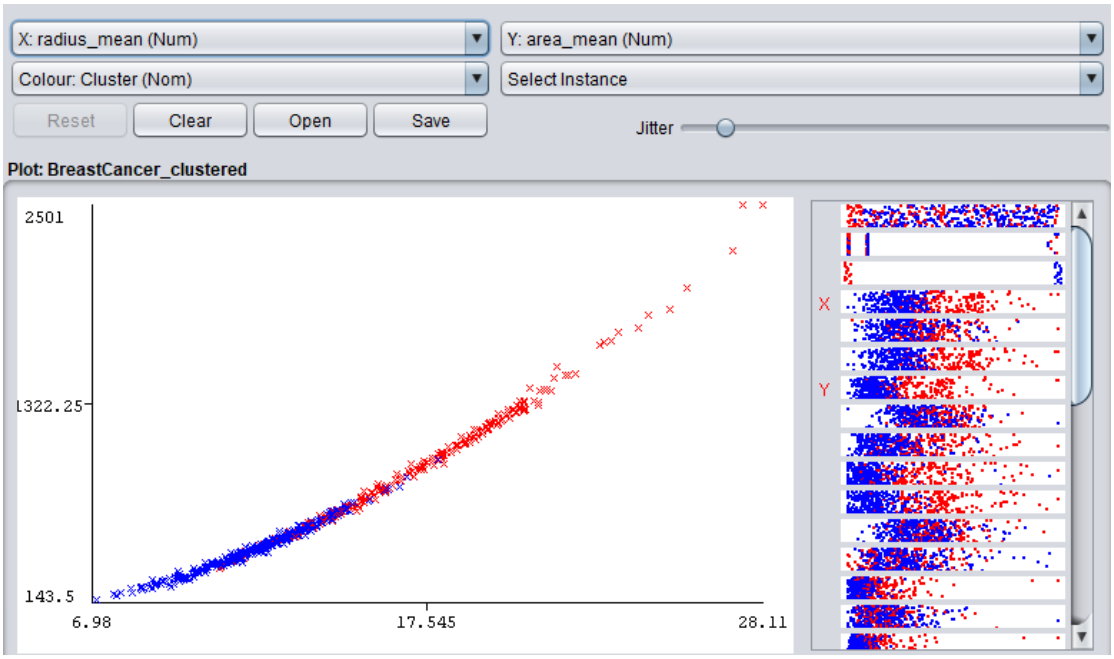


➔ 在 Weka 左下較的 Result list 中，選取此次的分析，用右鍵點取後選擇

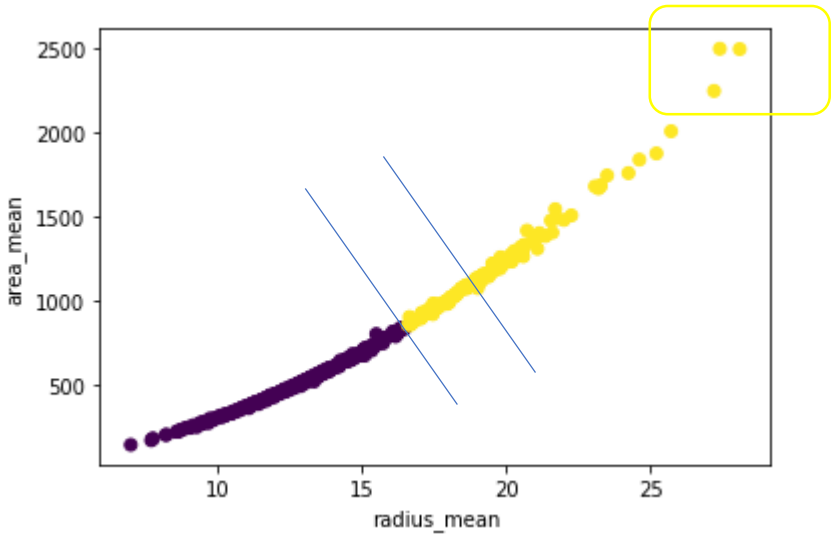
Visualize cluster assignments

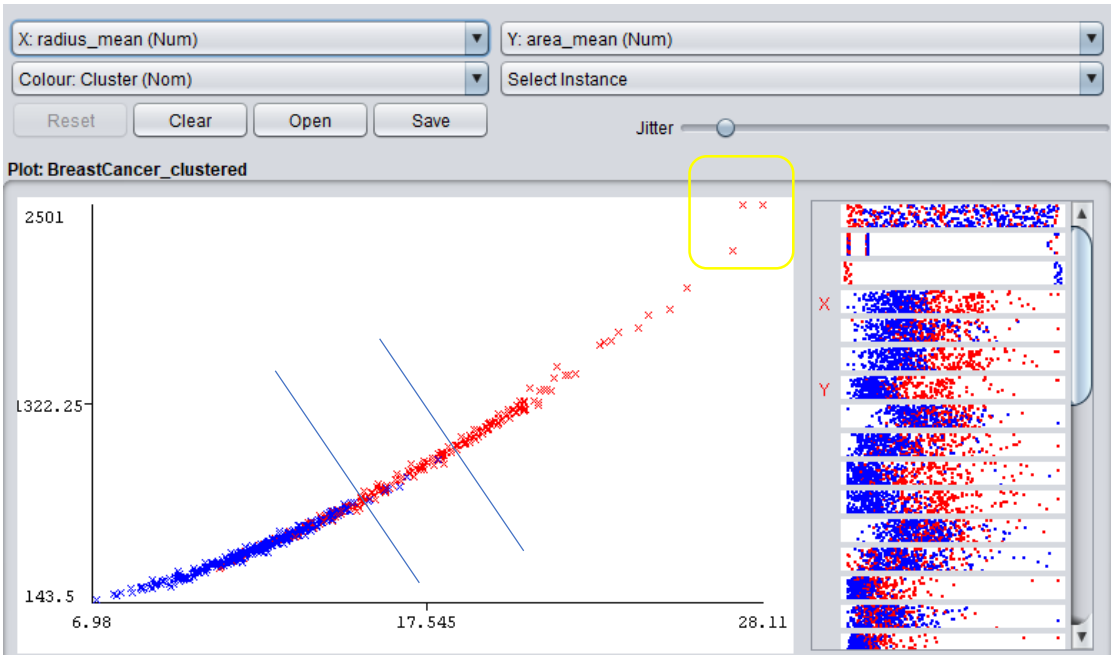


➔ 最上面兩個參數分別代表 X、Y 軸，依序調成 radius_mean, area_mean



➔ 完成之後即可看到與之前在 python 分析時相似的結果


2500
2000
1500
1000
500
area_mean
10
15
20
25
radius_mean


X: radius_mean (Num)
Y: area_mean (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: BreastCancer_clustered
2501
1322.25
143.5
6.98
17.545
28.11
X
Y

➔ 比較兩圖，的確非常相似。