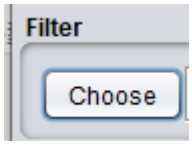
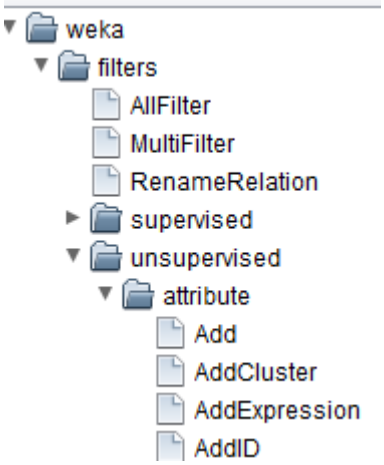
ECT HW7

Weka Part:

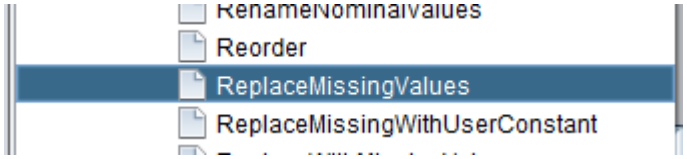
(a)



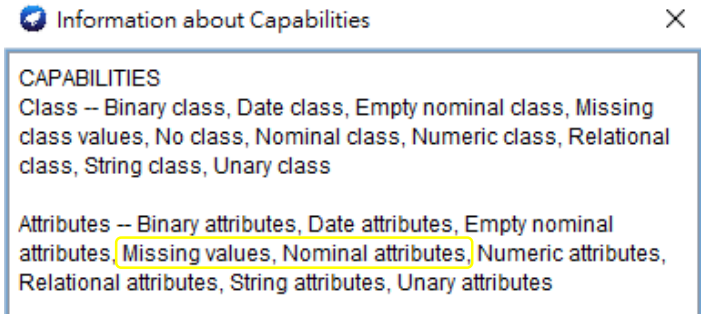
➔ 找到 Filter 的地方，點選 Choose



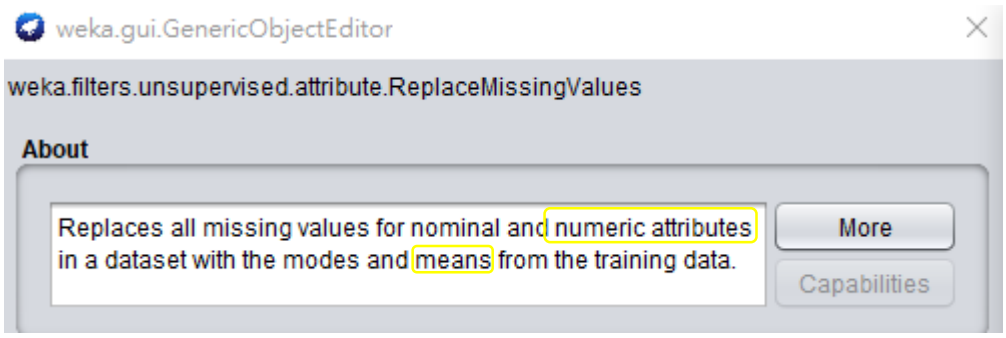
➔ 找 Weka -> filters -> unsupervised -> attribute



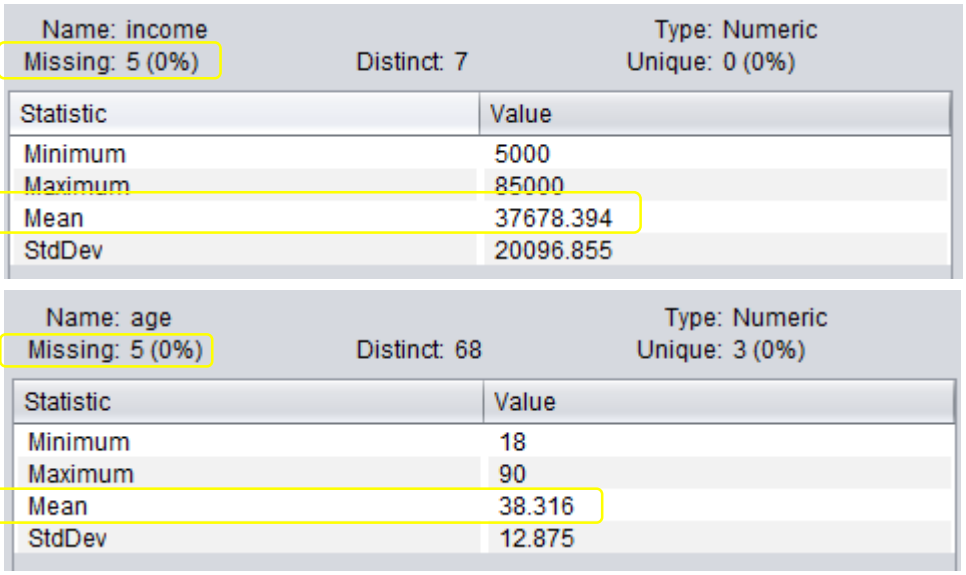
➔ 其中有一個 ReplaceMissingValues



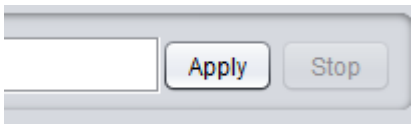
➔ 查看使用條件，可以處理 Missing values 和 Numeric attribute，因此可用



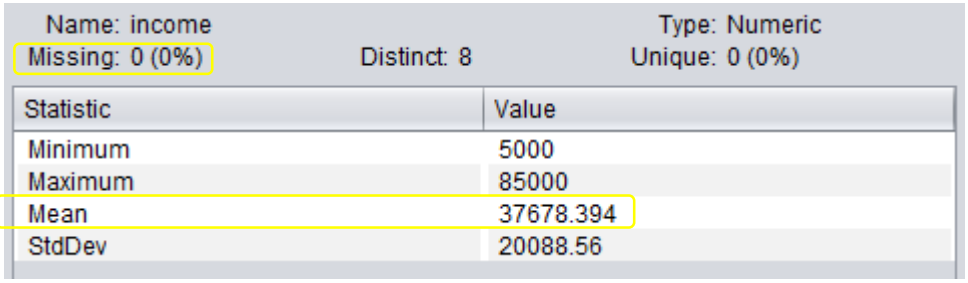
➔ 根據使用說明，他會把 numeric 的 missing value 用 mean 填補上去

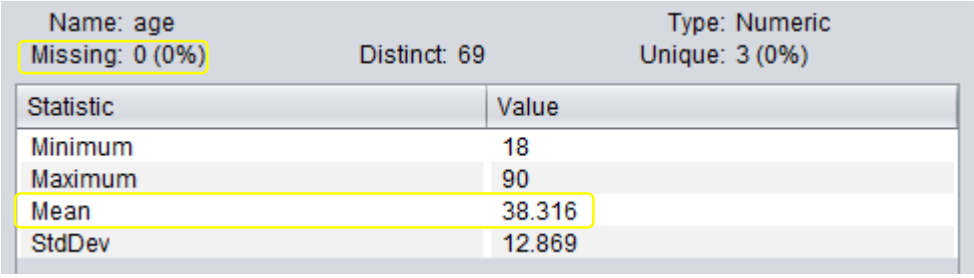


➔ 如上圖所示，有 Missing Value 的屬性分別為 income、age。根據使用說明預期，用 ReplaceMissingValues 處理過後，mean 並不會改變，因為只是加入一些為 value = mean 的數據。



➔ 點擊 Apply 使用 ReplaceMissingValues 處理數據





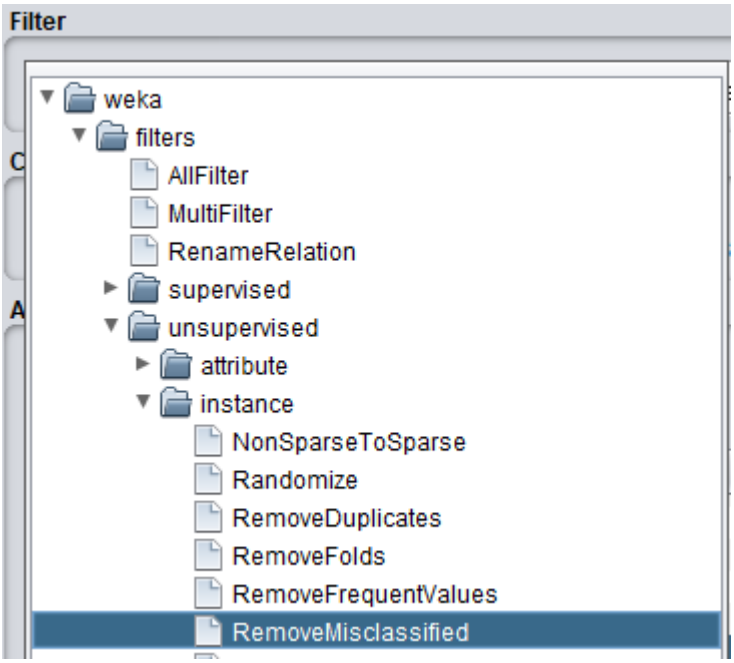
➔ 如圖所示，Missing 的部分已變為 0%，且 Mean 均沒有改變

因此題目所要求的：使用 ReplaceMissingValue，「列出補上的值」為何？

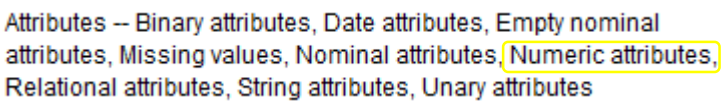
➔ 可以合理推測

- 屬性 Income 填補值 = 37678.394
- 屬性 Age 填補值 = 38.316

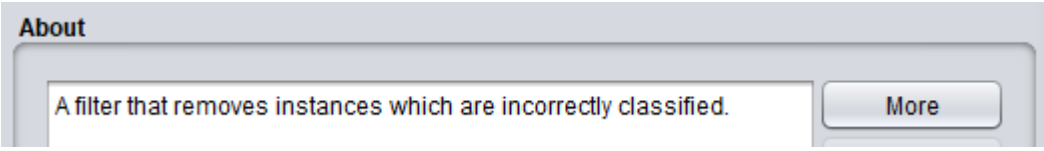
(b)



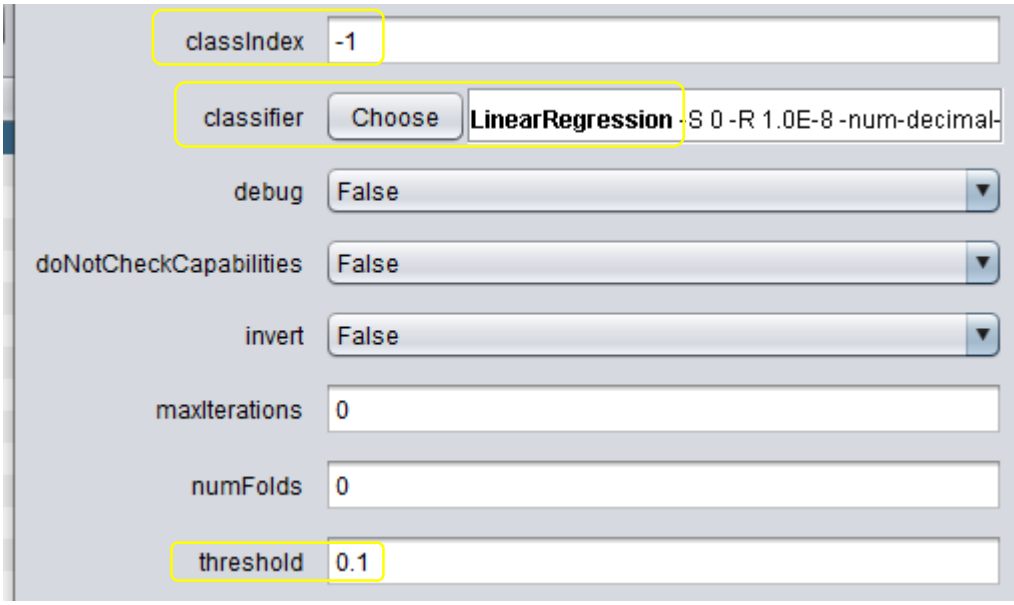
➔ 從 Filter 中依照此路徑找到 RemoveMisclassified



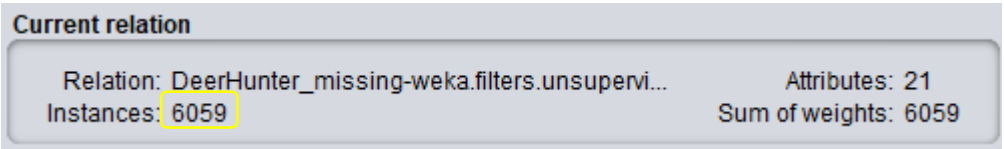
➔ 查看使用條件，可以處理 Numeric 的資料。



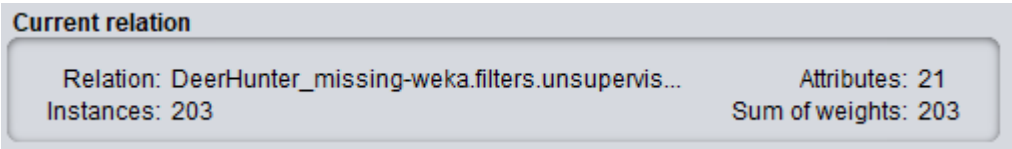
➔ 查看使用說明，發現他是使用分類器後把分類錯誤的部分當作 outlier 移除



➔ 依照題目要求，使用 LinearRegression 當作分類器，並且用默認設置的 classIndex = -1，代表使用最後一個屬性當作 class label，在此為「yes」屬性。Threshold 代表對 numeric 數據進行處理時，所可以容忍的誤差值，在此也是使用默認的 0.1

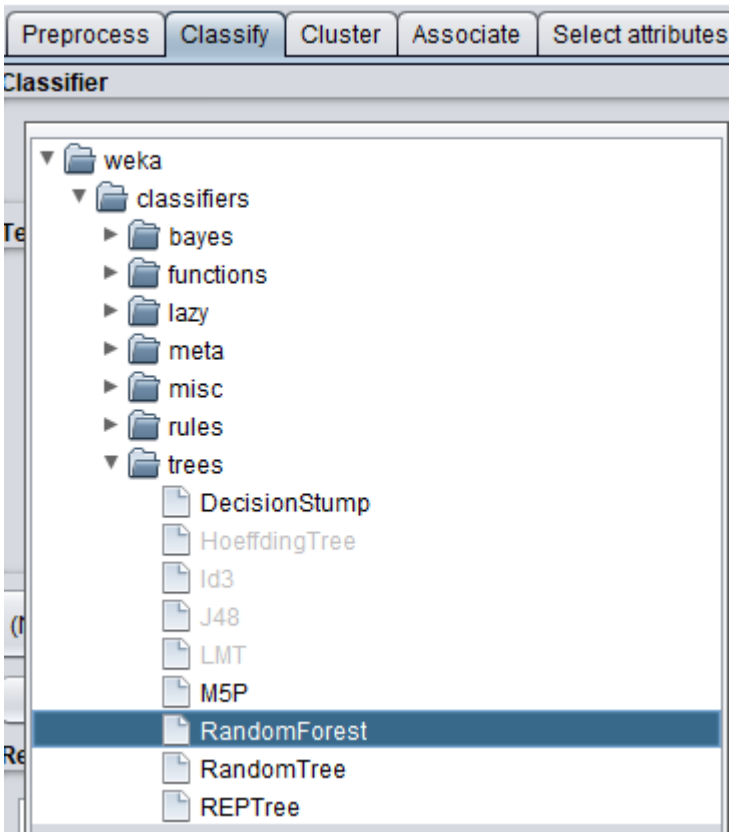


➔ 可以看到分類前有 6059 個 instance

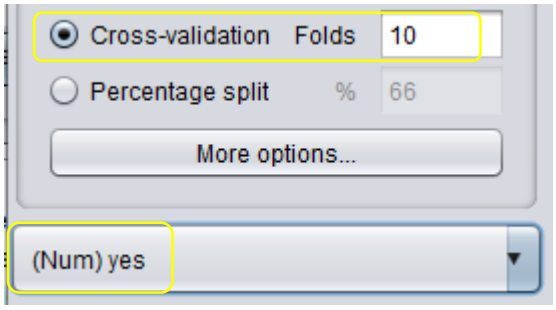


➔ 分類後只剩下 203 個 instance 在 0.1 的誤差之內。

(c)



➔ 切換至「Classify」並找到題目要求的「RandomForest」

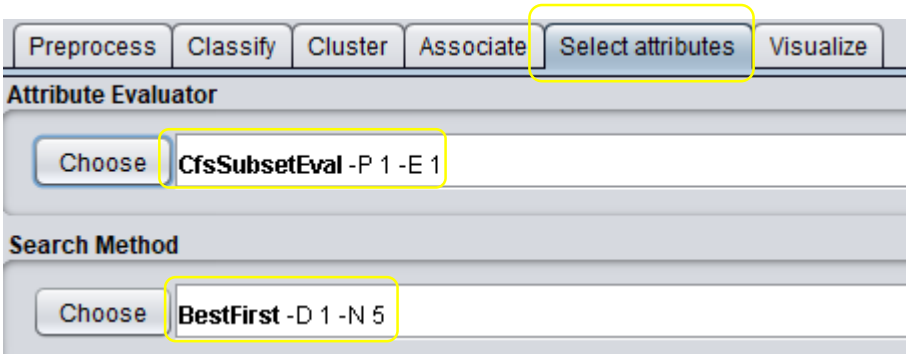


➔ 根據題目要求，使用 10-Fold Cross-validation，class label = yse

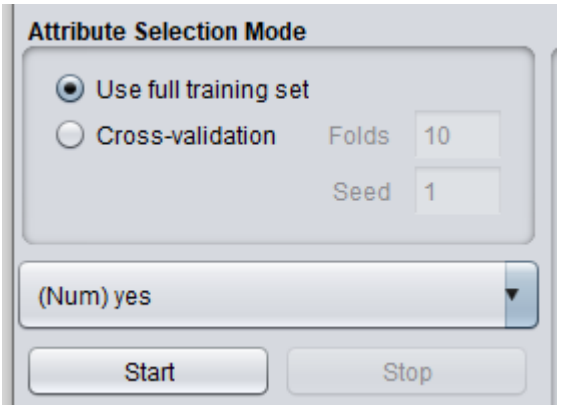
```
=== Summary ===

Correlation coefficient                  0.9863
Mean absolute error                      0.0292
Root mean squared error                  0.0672
Relative absolute error                 10.927  %
Root relative squared error             18.3093 %
Total Number of Instances              203
```

➔ 可以看到分類結果如上圖所示

(d)



➔ 切換至 Select Attributes，依照題目要求 Attributes Evaluator 選擇「CfsSubsetEval」、Search Method 選擇「BestFirst」

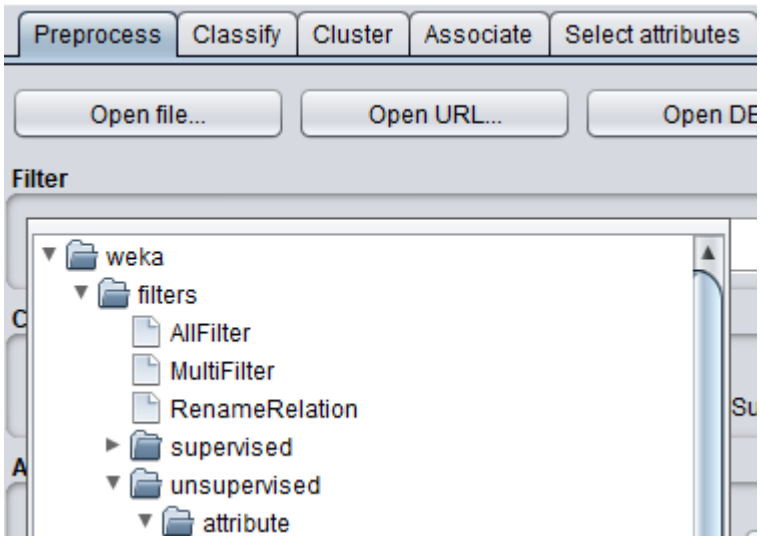


➔ 雖然 (c) 小題有要求使用 Cross-Validation，但此題並未提及，因此直接使用默認的設定「Use full training set」+「class label = yes」。

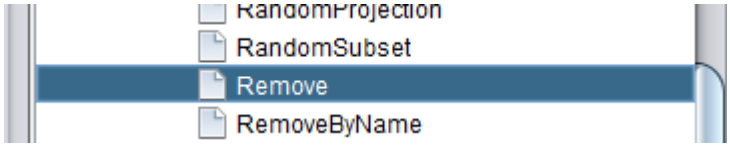
```
Selected attributes: 2,5,6,7,9,10,11,13,14,15,16,17,19,20 : 14
                     state
                     retire
                     employ
                     educ
                     income
                     gender
                     age
                     agehunt
                     trips
                     bagdeer
                     numbag
                     bagbuck
                     totcost
                     a
```

➔ 最後篩選出了 14 個 attributes，少了 1,3,4,8,12,18 的 attributes

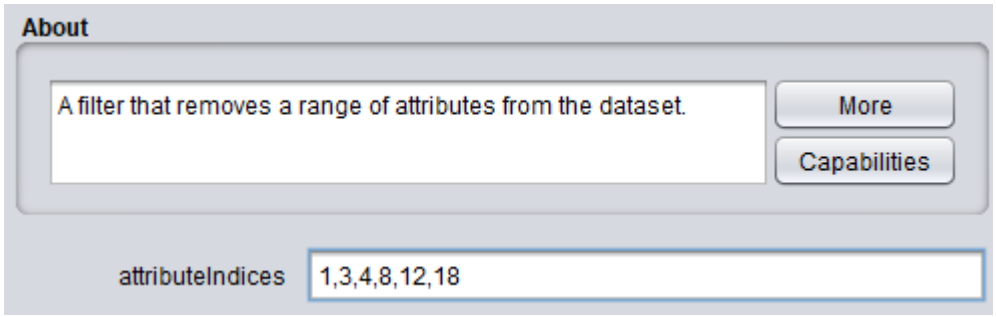
(e)



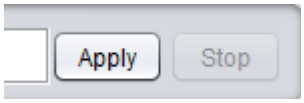
➔ 回到 Preprocess 的部分，找一個方法來處理被篩選掉的 attributes



➔ 選擇其中的 Remove



➔ 根據使用說明，他能幫我們從 dataset 中刪除不要的 attribute，依照 (d) 小題的結果，我們要刪除 1,3,4,8,12,18 的共 6 個 attributes，因此第一個參數部分就是這些 Indices



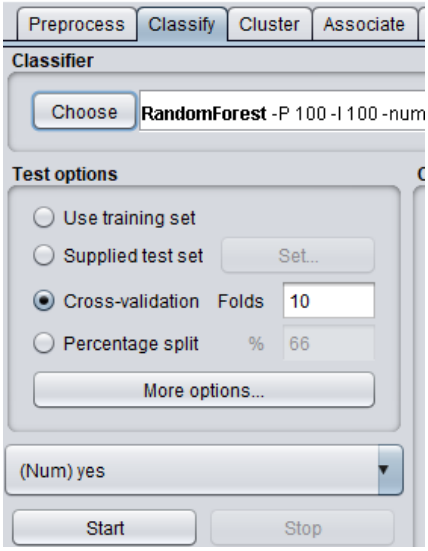
➔ 點擊 Apply 開始篩選

No.	Name
1	state
2	retire
3	employ
4	educ
5	income
6	gender
7	age
8	agehunt
9	trips
10	bagdeer
11	numbag
12	bagbuck
13	totcost
14	a
15	yes

➔ 篩選結果如上圖所示

```
Selected attributes: 2,5,6,7,9,10,11,13,14,15,16,17,19,20 : 14
                     state
                     retire
                     employ
                     educ
                     income
                     gender
                     age
                     agehunt
                     trips
                     bagdeer
                     numbag
                     bagbuck
                     totcost
                     a
```

➔ 跟 (d) 小題的結果對照一致



➔ 回到 Classify 並且不更改任何設定，直接點擊 Start

```
Attributes:   15
              state
              retire
              employ
              educ
              income
              gender
              age
              agehunt
              trips
              bagdeer
              numbag
              bagbuck
              totcost
              a
              yes
```

➔ 可看到現在只剩下 15 個屬性「14」個 input attributes +「1」個 output attribute

Attribute Selection 之後:

```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9877
Mean absolute error                      0.0253
Root mean squared error                  0.0621
Relative absolute error                  9.4713 %
Root relative squared error             16.9153 %
Total Number of Instances              203
```

➔

Attribute Selection 之前:

```
=== Summary ===

Correlation coefficient                  0.9863
Mean absolute error                      0.0292
Root mean squared error                  0.0672
Relative absolute error                 10.927  %
Root relative squared error             18.3093 %
Total Number of Instances              203
```

➔

- 可以看出進行 Attribute Selection 之後，Root mean squared error 下降了，相關係數上升了，因此是一個有效的結果。

Python Part:

(a)

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
data = datasets.load_iris()
print(data)
```

➔ 先載入 IRIS 資料集，並印出來看大致上的格式

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],

'target': array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

➔ 可以看出他是一個 dictionary 的形式，其中「data」代表 Input

attribute、「target」代表 output attribute

```
x = data["data"] # input attributes
y = data["target"] # output target
print(x.shape)
print(y.shape)
```

```
(150, 4)
(150,)
```

➔ 因此把資料用這兩個 key 切分成 Attribute、Target

```
from mpl_toolkits.mplot3d import Axes3D
ax3D = Axes3D(plt.figure())
```

➔ 用 Axes3D 初始化 3D 模型

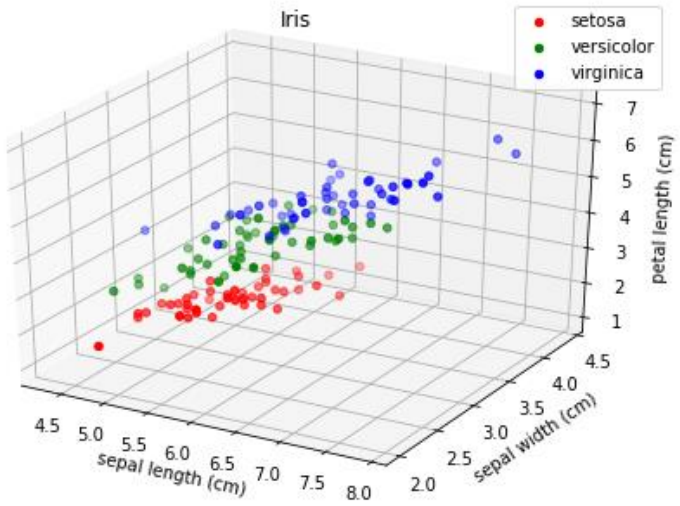
```
# [0,1,2]代表3種target class
for c, i, target_name in zip('rgb', [0,1,2], data.target_names):
    # 前3個參數設定3軸的值。
    # x[y==i, 0]中 : y==i代表屬於哪個類別； 0代表在屬性是col = 0的屬性
    # 整段就是在 y==i 的類別中，用前3個屬性去表示它
    ax3D.scatter(x[y==i, 0],x[y==i, 1] ,x[y==i, 2], c=c, label=target_name)
```

➔ 用 zip 把各個需要的參數包起來，用 scatter 把散佈圖畫出來。

- Scatter 前 3 個參數分別為在 3D 空間中 3 軸的值
- c 代表顏色
- label 代表此點之後在圖例上的名字
- 詳細內容可參考圖中的註解

```
# 設置各軸屬性名稱
ax3D.set_xlabel(data.feature_names[0])
ax3D.set_ylabel(data.feature_names[1])
ax3D.set_zlabel(data.feature_names[2])

ax3D.set_title('Iris') # 設置圖片標題
plt.legend() # 把之前設置的label畫成圖例顯示出來
plt.show()
```

➔ 做一些基本設定，把圖片該有的標示標清楚



➔ 畫出來的 3D 分布圖如上圖所示

➔ 挑選的 3 個屬性 :「sepal length、sepal width、petal length」

(b)

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2) # 指定降維程度
x_2d = pca.fit_transform(x) # 降維
```

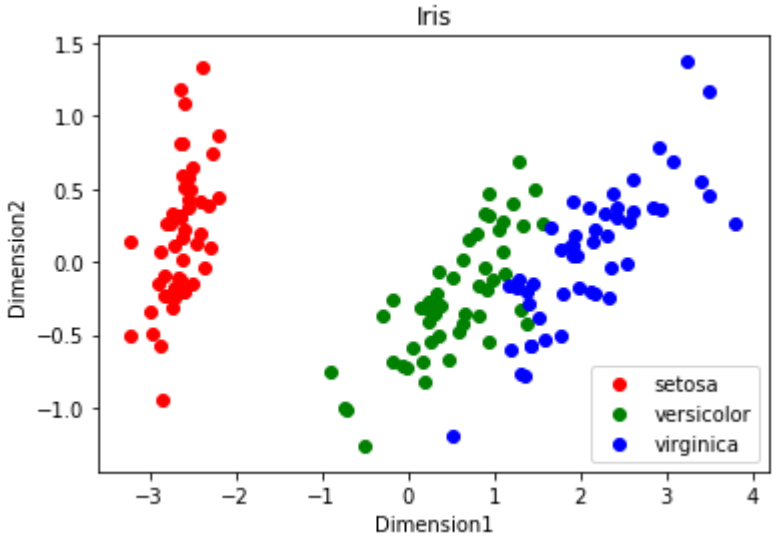
➔ 使用 sklearn.decomposition 中的 PCA

- n_components：代表要降至多少維度
- fit_transform：降維並返回結果

(c)

```
ax2D = plt.figure() # 產生畫布
for c, i, target_name in zip('rgb', [0,1,2], data.target_names):
    plt.scatter(x_2d[y==i, 0], x_2d[y==i, 1] , c=c, label=target_name)

plt.xlabel("Dimension1")
plt.ylabel("Dimension2")
plt.title('Iris') # 設置圖片標題
plt.legend() # 把之前設置的label畫成圖例顯示出來
plt.show()
```

➔ 這次純粹使用「plt」，因為是畫 2D 圖，方法跟前述相同，只是這次 scatter 的參數只剩下 2 軸的值



➔ 結果如圖所示

(d)

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

➔ 先導入資料集

```
# x:input
x_BC = df.loc[:,"radius_mean":].values # 用values轉換成array來處理
print(x_BC)
# y:output
y_BC = df.loc[:,["diagnosis"]].values.ravel() # 要先ravel不然之後維度會不正確
print(y_BC)

target_names = ["diagnosis = 0", "diagnosis = 1"] # 自製label名稱
```

➔ 把資料集切分成 input、output

- target_names 是之後畫圖時用來代表圖例中的 label name
- 要使用 ravel()展開 y 的部分，不然之後會報錯因為維度不正確

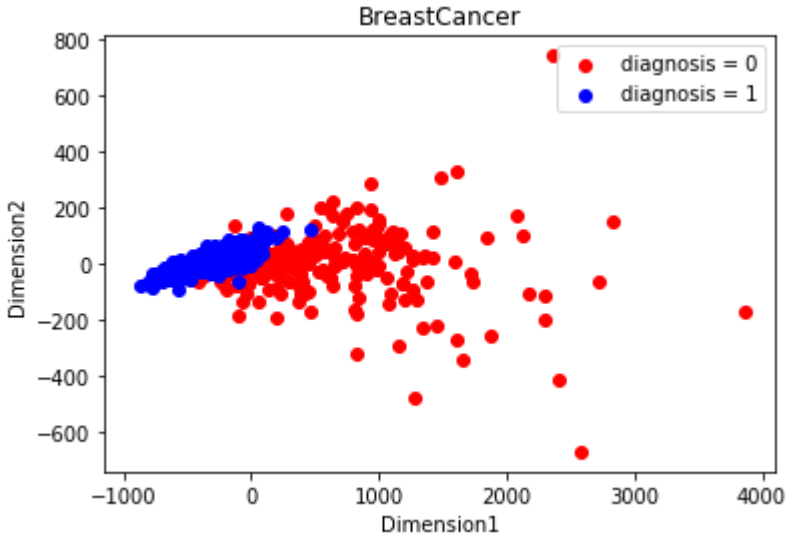
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2) # 指定降維程度
x_BC_2d = pca.fit_transform(x_BC) # 降維
```

➔ 把 input 部分降維至 2 維，存放至 x_BC_2d

(e)

```
ax2D_BC = plt.figure() # 產生畫布
for c, i, target_name in zip('rb', [0,1], target_names):
    plt.scatter(x_BC_2d[y_BC==i, 0], x_BC_2d[y_BC==i, 1], c=c, label=target_name)

plt.xlabel("Dimension1")
plt.ylabel("Dimension2")
plt.title('BreastCancer') # 設置圖片標題
plt.legend() # 把之前設置的label畫成圖例顯示出來
plt.show()
```

➔ 方法跟前述相同，不加贅述


BreastCancer
diagnosis = 0
diagnosis = 1
Dimension2
Dimension1
800
600
400
200
0
−200
−400
−600
−1000
0
1000
2000
3000
4000

➔ 結果如圖所示