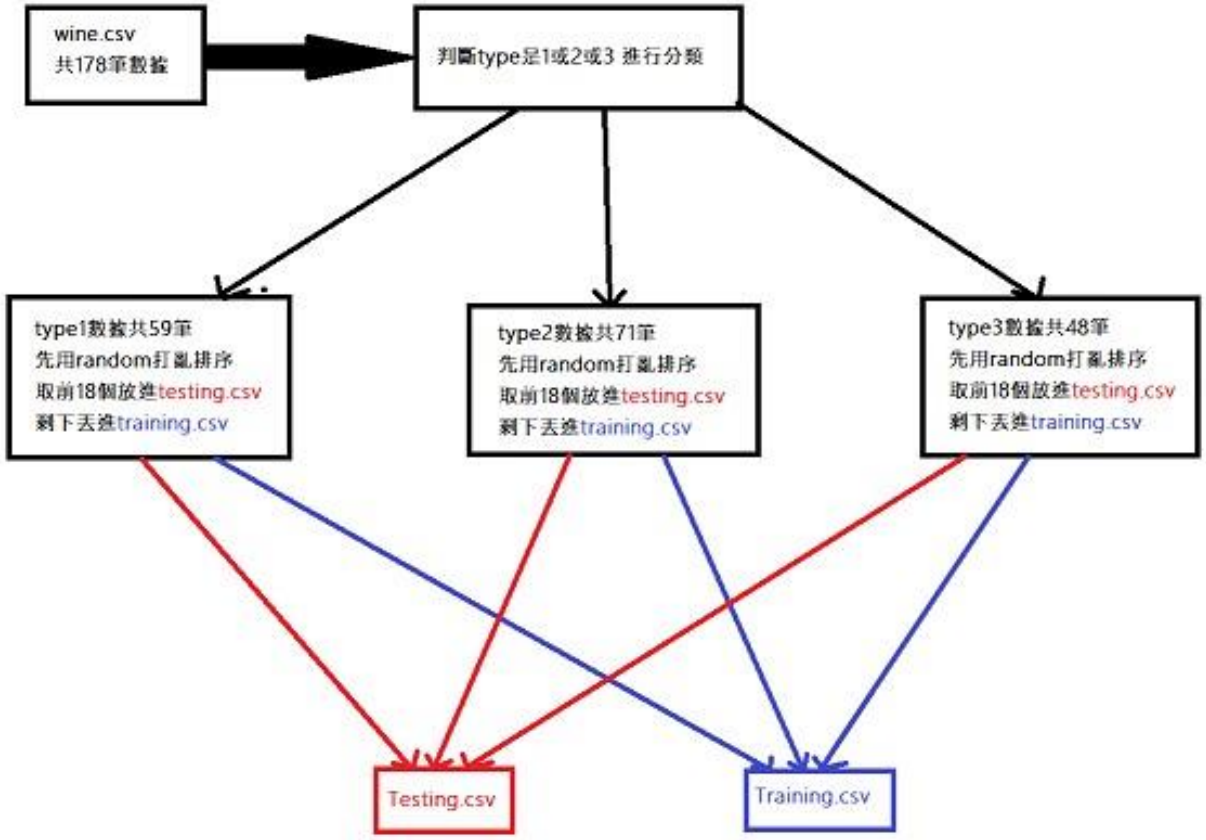
109061613 黃柏凱

ML hw1

一、 程式架構

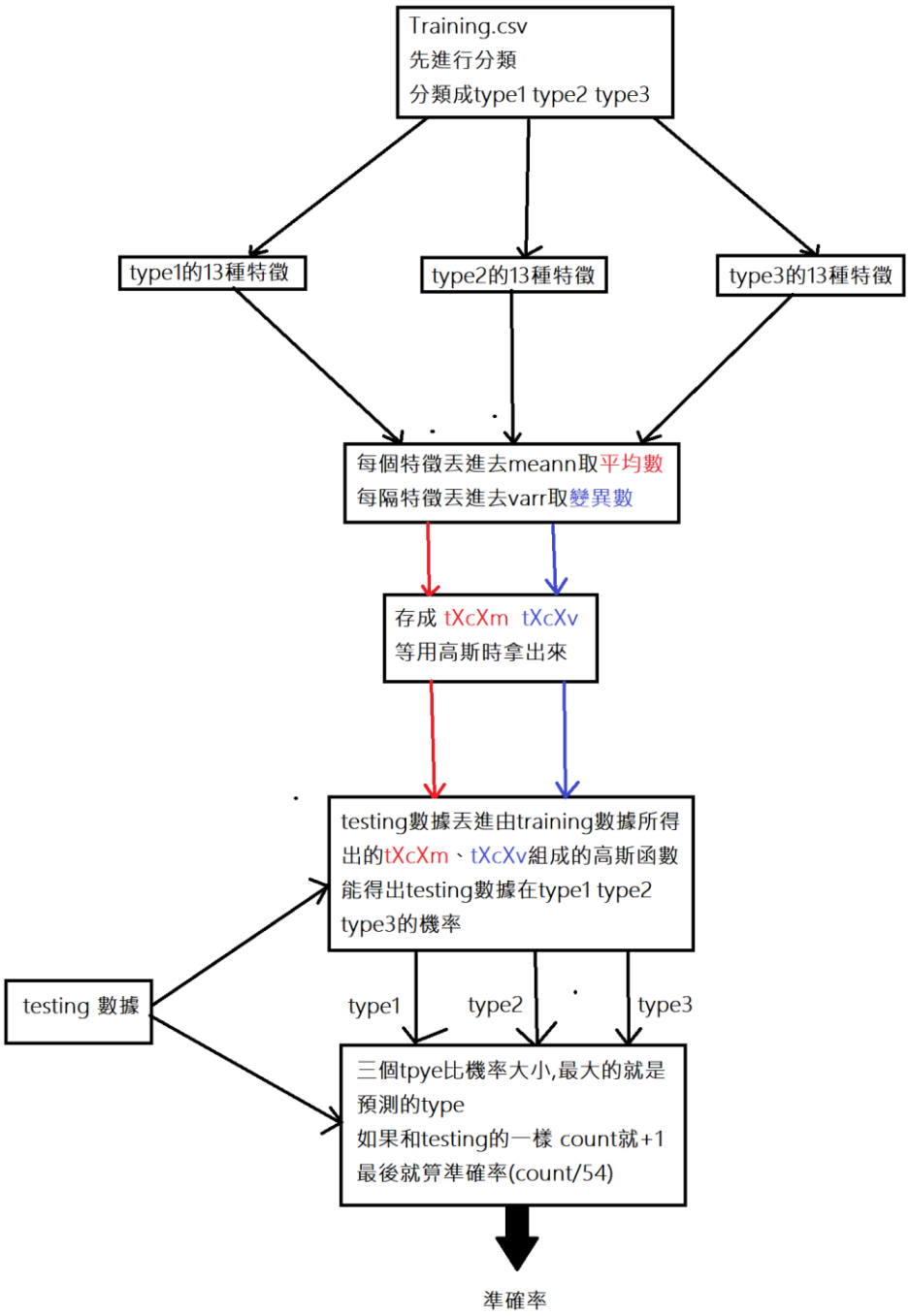
1. Testing



主要就是先用 if 來判斷 wine.csv(data)裡面數據開頭第一行的值也就是 data[n][0]是 type 幾，然後整批資料再一次用 random.sample()，用全部的資料隨機抽取全部資料的方式來進行亂排、打亂順序，最後再取前 18 個當作 testing data，剩下的就當作 traing data

最後完成後會輸出
" ----------make testing and training , done----------"

2. Training


Training.csv
先進行分類
分類成type1 type2 type3
type1的13種特徵
type2的13種特徵
type3的13種特徵
每個特徵丟進去meann取平均數
每隔特徵丟進去varr取變異數
存成 tXcXm tXcXv
等用高斯時拿出來
testing數據丟進由training數據所得出的tXcXm、tXcXv組成的高斯函數能得出testing數據在type1 type2 type3的機率
testing 數據
type1
type2
type3
三個tpye比機率大小,最大的就是預測的type
如果和testing的一樣 count就+1
最後就算準確率(count/54)
準確率

這部分最主要就是由幾個 function 所組成的
第一個就是 meann 取 mean 的，要輸入三個參數，a 就是 training.csv，b 就是選擇要取哪個 type 的特徵，c 就是從 13 個特徵中選要計算哪一個，輸出就是平均數

```
def meann(a, b, c):    # a:training.csv , b: type , c:特徵
    t1 = 0
    t2 = 0
    t3 = 0
    t11 = np.zeros((1, 41))
    t22 = np.zeros((1, 53))
    t33 = np.zeros((1, 30))
    for n in range(row_number):
        hh = a[n][0]

        if(hh == 1 and b == 1):
            # print(a[n])
            t11[0][n] = a[n][c]
            t1 += 1

        if(hh == 2 and b == 2):
            # print(a[n])
            t22[0][n-41] = a[n][c]
            t2 += 1

        if(hh == 3 and b == 3):
            # print(a[n])
            t33[0][n-94] = a[n][c]
            # print(t33)
            t3 += 1
    #print(t1, t2, t3)
    #t3_mean = np.mean(t33)
    # print(t3_mean)
    if(b == 1):
        m = np.mean(t11)
        return m
    if(b == 2):
        mm = np.mean(t22)
```

```
        return mm
    if(b == 3):
        mmm = np.mean(t33)
        return mmm
```

第二個是 varr 算變異數的，多虧有 numpy.var 所以和上面那個幾乎一樣，就下面最後取完全部數字要做平均的地方改成取變異數就好

```
    if(b == 1):
        m = np.var(t11)
        return m
    if(b == 2):
        mm = np.var(t22)
        return mm
    if(b == 3):
        mmm = np.var(t33)
        return mmm
```

算完 mean 和 var 後就會把數字存在 tXcXm(存 mean 的)還有 tXcXv(存 var 的)留著給之後算高斯用

第三部分就是 n 用來取高斯函數的，x 就是 testing data 要丟進去的特徵，m 就是上面所算完的平均數，v 就是上面求出的變異數

```
def n(x, m, v):    # x:input m:mean v:var
    norm = (1/((2*math.pi*v)**(1/2)))*math.exp((-1/(2*v))*((x-m)**2))

    return norm
```

最後就是用 for loop 丟進 54 個 testing data，得出該次數據出現的 tpye123 的機率
然後再比較 3 種 type 的機率比大小，取最大當預測結果，最後再和原本

testing.csv 裡的 type 做比較即可
這邊我是把 type 123 的機率分別丟進 s 陣列裡再用 argmax 來抓最大的機率是哪個 v 不過因為他取出來是 0 開始算，所以我在結果後還有+1(w 的部分)讓他符合 123 這三個數字，就能直接做比較了

二、 結果

圖片是我大概做兩次的結果
先執行 ML_HW1_part1.py 分出 traing data 和 testing data
在執行 ML_HW1_part2.py 做訓練和比較結果輸出
試了蠻多次,準確率都在 95%以上,甚至會到 100%

```
[Running] python -u "c:\Users\Kai\Desktop\ML_HW\ML_HW1_part1.py"
----------make testing and training , done----------

[Done] exited with code=0 in 0.231 seconds

[Running] python -u "c:\Users\Kai\Desktop\ML_HW\ML_HW1_part2.py"
col_number =  14  ,  row_number =  124
col_number2 =  14  ,  row_number2 =  54
---------
!!!!!!!!!!!!!!!!!!!!!
98.14814814814815 %
!!!!!!!!!!!!!!!!!!!!!

[Done] exited with code=0 in 0.261 seconds

[Running] python -u "c:\Users\Kai\Desktop\ML_HW\ML_HW1_part1.py"
----------make testing and training , done----------

[Done] exited with code=0 in 0.228 seconds

[Running] python -u "c:\Users\Kai\Desktop\ML_HW\ML_HW1_part2.py"
col_number =  14  ,  row_number =  124
col_number2 =  14  ,  row_number2 =  54
---------
!!!!!!!!!!!!!!!!!!!!!
96.29629629629629 %
!!!!!!!!!!!!!!!!!!!!!

[Done] exited with code=0 in 0.251 seconds
```

三、 討論

```
type1p = n(a[m][1], t1c1m, t1c1v)*n(a[m][2], t1c2m, t1c2v)*n(a[m][3], t1c3m,
t1c3v)*n(a[m][4], t1c4m, t1c4v)*n(a[m][5], t1c5m, t1c5v)*n(a[m][6], t1c6m, t1c6v) * \
    n(a[m][7], t1c7m, t1c7v)*n(a[m][8], t1c8m, t1c8v)*n(a[m][9], t1c9m, t1c9v) * \
    n(a[m][10], t1c10m, t1c10v)*n(a[m][11], t1c11m, t1c11v) * \
    n(a[m][12], t1c12m, t1c12v)*n(a[m][13], t1c13m, t1c13v)*(41/124)
```

以 type1 為例，紅色字部分就是運算連乘所得出的 likelihood，藍色字就是 prior probability，黑色的就是 posterior probability
之後就和上面講得一樣了，三個 type 分別算出 posterior probability 然後再取最大(argmax)，就是 MAP 了