

判別分析與 K-近鄰演算法的分類問題

郭翊萱 711133115

2022.11

在機器學習中，存在有很多種方法能對資料進行分類並建立模型，包含僅能利用於二元資料的迴歸分析方法 (Regression Analysis)、常用的羅吉斯迴歸 (Logistic Regression) 與本節即將介紹的三種方法-線性判別分析 (Linear Discriminant Analysis)、二次判別分析 (Quadratic Discriminant Analysis) 以及 K-近鄰演算法 (K-Nearest Neighbors)。在本節中將依次闡述線性判別分析、二次判別分析與 K-近鄰演算法的理論並展示各自如何針對二群與三群的類別資料進行分類，以及各自的訓練誤差與測試誤差，其中，本節將採用 Bootstrapping 方法來計算各自的平均訓練誤差與測試誤差。最後，本節將對三種方法的分類效能進行比較，期望能對三種分類方法有更深入的了解。

1 Introduction to LDA, QDA and KNN

1.1 Linear Discriminant Analysis (LDA)

在線性判別分析 (Linear Discriminant Analysis) 中，試圖找出預測變數 X 在不同的反應變數 G (不同類別) 的分配，並利用貝氏定理去估計 $P(G = k|X = x)$ 。假設先驗分配 (prior probability) ($P(G = k)$) 代表隨機從類別 k 抽出觀察值的機率，而後驗分配 $f_k(x) = P(X = x|G = k)$ 代表觀察值來自類別 k 時 X 的機率密度函數，則透過貝氏定理可知：

$$P(G = k|X = x) = \frac{P(G = k)P(X = x|G = k)}{\sum_{l=1}^k P(G = l)P(x = X|g = l)} \quad (1)$$

其中，若使後驗分配 $P(X = x|G = k)$ 最大，則此分類的分類誤差 (error rate) 最少。

接著利用存在一個預測變數 X 時，線性判別分析 (LDA) 如何有效的進行二元分類。假設 $f_k(x) = P(X = x|G = k)$ 來自常態母體 (Normal) 或多變量常態母體 (Multivariate Normal)，其機率密度函數為：

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right) \quad (2)$$

其中 μ_k 與 σ_k^2 為第 k 類的平均數與變異數。在線性判別分析 (LDA) 中會假設 $\sigma_1^2 = \dots = \sigma_k^2$ ，故由此可得：

$$P(G = k|X = x) = \frac{P(G = k) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^k P(G = l) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \quad (3)$$

接著，將式 (3) 取 \log ，即可得到式 (4)：

$$\delta_k(x) = x \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(P(G = k)) \quad (4)$$

由式 (4) 可知，若 $2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$ ，則將該筆觀察值分類至第一群 (其平均數為 m_1)，且此分類的分界線如式 (5)：

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2} \quad (5)$$

在線性判別分析中假設了預測變數 X 的每個分類皆是來自於常態母體或者是多變量常態母體，但在實務中往往並不知道其母體服從何種分配。而在此練習中，假設預測變數 X 的每個類別都來自常態母體去估計母體參數 $\mu_1, \dots, \mu_k, \pi_1, \dots, \pi_k$ 以及 σ^2 ，得到式 (6)、式 (7) 以及式 (8) 之估計量。

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \quad (6)$$

$$\hat{\sigma}^2 = \frac{1}{n - K} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \quad (7)$$

$$\hat{\pi}_k = n_k/n \quad (8)$$

其中 n 為所有訓練資料集的觀察值的數量、 n_k 為第 k 個類別中訓練資料集的觀察值數量，而 $\hat{\sigma}^2$ 可被視為 K 個類別的樣本變異數的加權平均。另外，若缺少了 π_1, \dots, π_k 任何一個值，則線性判別分析 (LDA) 會利用式 (8) 的 $\hat{\pi}_k$ 來估計 π_k 。

為什麼羅吉斯迴歸分類表現極好，仍需探討其他的機器學習方法來進行分類問題？

- 當分類資料區別很大時（可透過散布圖觀察），羅吉斯迴歸（Logistic Regression）方法的結果並不穩定（unstable），而線性判別分析（Linear Discriminant Analysis）並不會有此類問題。
- 如果樣本數很小且預測變數 X 是趨近常態的，則線性判別分析同樣相較於羅吉斯迴歸會更加穩定。

在介紹完 $p = 1$ ，即預測變數 X 只有一個時線性判別分析（LDA）的理論後，接著將線性判別分析擴增至 $p > 1$ 的情況下進行討論，即預測變數服從多變量常態分配母體（Multivariate Normal Distribution）的情況，另外，在線性判別分析中皆假設所有類別的共變異數相等。為了進一步了解多變量常態母體資料，首先先模擬資料並繪製多變量常態母體的散布圖來了解其基本性質，假設平均數為 0，而共變異數矩陣各為：

$$\Sigma_1 = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix},$$

則多變量常態分配的圖形如圖 1，其部分程式碼亦呈現如下。

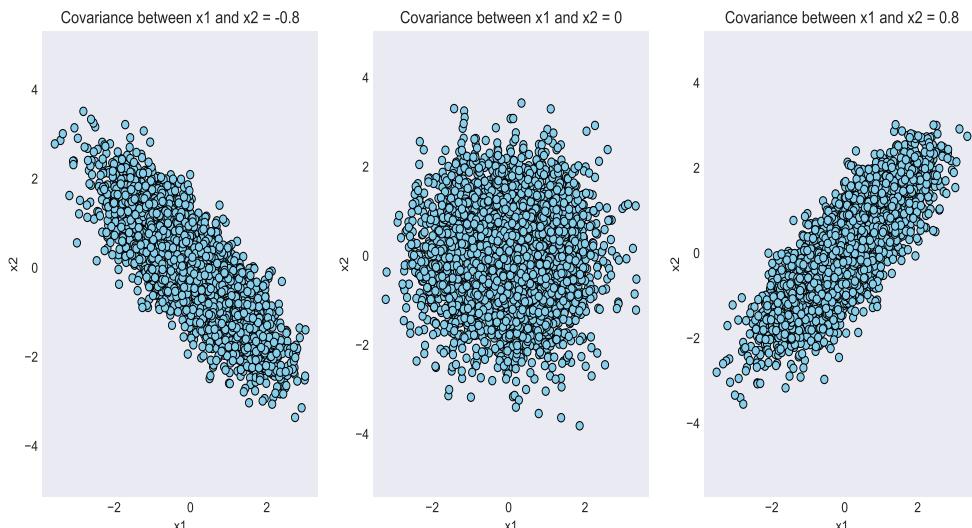


圖 1: Plot of Multivariate Normal Distribution-2D

```
plt.style.use('seaborn-dark')
plt.rcParams['figure.figsize']=14, 6
fig = plt.figure()
```

```

random_seed=1000
cov_val = [-0.8, 0, 0.8]
mean = np.array([0,0])

for idx, val in enumerate(cov_val):
    plt.subplot(1, 3, idx+1)
    cov = np.array([[1, val], [val, 1]])

    distr = multivariate_normal(cov = cov, mean = mean,
                                 seed = random_seed)
    data = distr.rvs(size = 5000)

    plt.plot(data[:,0],data[:,1], 'o', c='red',
              markeredgewidth = 0.5,
              markeredgecolor = 'black')

```

透過圖 1 難以觀察到兩多變量常態分配的關係究竟為何，因此嘗試將圖形繪製成 3D 圖形來重新觀察兩變量的關係，其結果如圖 2，可以明顯看出兩變數 x_1 與 x_2 的關係。其部分程式碼亦呈現如下。

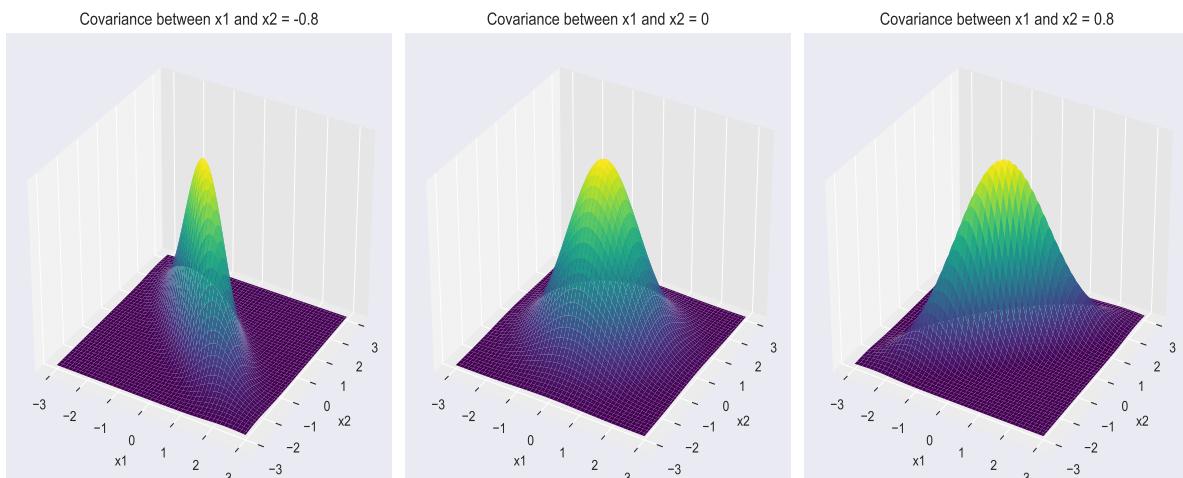


圖 2: Plot of Multivariate Normal Distribution-3D

```

pdf_list = []
for idx, val in enumerate(cov_val):
    cov = np.array([[1, val], [val, 1]])
    distr = multivariate_normal(cov = cov, mean = mean,
                                 seed = random_seed)

    mean_1, mean_2 = mean[0], mean[1]
    sigma_1, sigma_2 = cov[0,0], cov[1,1]

```

```

x = np.linspace(-3*sigma_1, 3*sigma_1, num=100)
y = np.linspace(-3*sigma_2, 3*sigma_2, num=100)
X, Y = np.meshgrid(x,y)

pdf = np.zeros(X.shape)
for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        pdf[i,j] = distr.pdf([X[i,j], Y[i,j]])

key = 131+idx
ax = fig.add_subplot(key, projection = '3d')
ax.plot_surface(X, Y, pdf, cmap = 'viridis')
pdf_list.append(pdf)
ax.axes.zaxis.set_ticks([])

```

接著詳細討論線性判別分析 (LDA) 在母體為多變量常態分配 (Multivariate Normal Distribution) 時的理論方法。假設 $X \sim N(\mu, \Sigma)$ ，且其機率密度函數如式(9)：

$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right) \quad (9)$$

而線性判別分析 (LDA) 會將觀察值 $X = x$ 分類至使式(10)的值最大的第 k 類。

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log P(G = k) \quad (10)$$

透過以上式(10)即可得到線性判別分析的分群結果，而在繪製圖形時，若欲繪製線性判別分析的群組分界線，則將繪製屬於群組 k 和屬於群組 l 的機率相同的地方，即繪製式(11)成立的函數。

$$P(G = k | X = x) = P(G = l | X = x) \quad (11)$$

亦可將上式(11)取 \log 轉換成如式(12)

$$\begin{aligned}
\ln \frac{P(G = k | X = x)}{P(G = l | X = x)} &= \ln \frac{f_k(x)}{f_l(x)} + \ln \frac{P(G = k)}{P(G = l)} \\
&= \ln \frac{P(G = k)}{P(G = l)} - \frac{1}{2} (\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) \\
&\quad + x^T \Sigma^{-1} (\mu_k - \mu_l) \\
&= 0
\end{aligned} \quad (12)$$

1.2 Classification by LDA

為了理解線性判別分析 (LDA) 的分類效果，在此利用 Python 來更深入了解其分群的效能與方法，其部分程式碼呈現如下。

```
C1, C2 = X[y==0,:], X[y==1,:]

n = D[:,0].size
n1, n2 = C1[:,0].size, C2[:,0].size
pi1, pi2 = n1/n, n2/n
mu1, mu2 = np.mean(C1, axis = 0), np.mean(C2, axis = 0)
Sigma = (np.cov(C1.T) + np.cov(C2.T))/2

K = np.log(pi1/pi2) - 0.5 * (mu1 + mu2) \
@ LA.inv(Sigma) @ (mu1 - mu2).T
L = LA.inv(Sigma) @ (mu1 - mu2).T
f = lambda x : -L[0]/L[1] * x - K/L[1]
x = np.linspace(1, 5, 10)
plt.plot(x, f(x))
```

$$K = \ln \frac{P(G=1)}{P(G=2)} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) \quad (13)$$

設置 $C1, C2$ 分別是資料集中屬於類別 0 以及屬於類別 1 的資料，並根據式 (6)、式 (7) 以及式 (8) 對兩變量的參數進行估計得到各自的 $\hat{\mu}_k, \hat{\sigma}^2$ 以及 $\hat{\pi}_k$ 。程式碼中的 K 則來自於式 (12)，由於此範例是式 (12) 雙變量的情況，故將式 (12) 改寫成如式 (13)，並設其中之 $\Sigma^{-1}(\mu_k - \mu_l)$ 為式 (14)：

$$L = \Sigma^{-1}(\mu_k - \mu_l) \quad (14)$$

$$K + L(1)x_1 + L(2)x_2 = 0 \quad (15)$$

透過以上之方式，可以將函式改寫成如式 (15)，接著即可以利用程式碼 `plt.plot(x, f(x))` 求出如圖 3 之分界線。

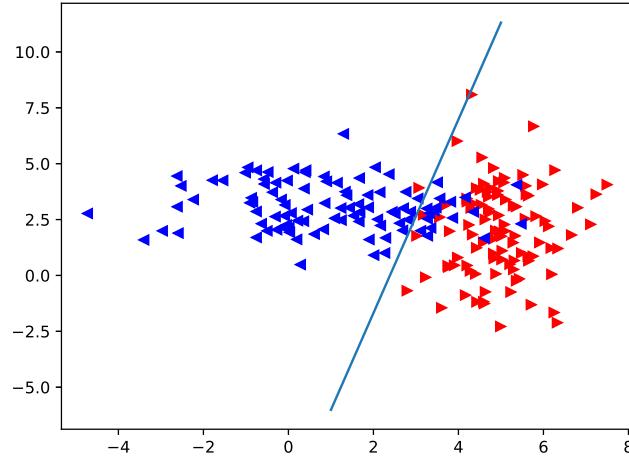


圖 3: Example of LDA in binary classification

1.3 Quadratic Discriminant Analysis (QDA)

在線性判別分析中假設母體服從常態分配且每個類別的共變異數相同（共變異矩陣一樣），但在實務上往往很難滿足此假設，而二次判別分析中完善了此一缺點。二次判別分析假設了每個類別都有各自的共變異數，因此式 (10) 可被改寫成如式 (16)：

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \ln |\Sigma_k| + \ln P(G = k) \\ &= -\frac{1}{2}x^T \Sigma_k^{-1} x + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \ln |\Sigma_k| + \ln P(G = k)\end{aligned}\quad (16)$$

其中，二次判別分析 (QDA) 需估計 Σ_k , μ_k 以及 $P(G = k)$ 並將觀察值 $X = x$ 分類至使式 (16) 的值最大的類別 k 。

另外，二次判別分析的分界線為變數的二次方，被寫作如式 (17)

$$\{x | \delta_k(x) = \delta_l(x)\} \quad (17)$$

1.4 Comparison to LDA and QDA

在了解線性判別分析與二次判別分析的理論後，嘗試利用 Python 實際進行操作並繪圖以更了解如何利用此兩種機器學習方法進行分群，以下列出基本的分群步驟：

- 繪製散布圖

首先利用以下程式碼繪製資料集 *la2.txt* 中兩群資料的散布圖，其中利用函數 *np.random.randint* 來產生小於 50、大小為該資料集大小的整數。

```
X = D[:, 0:2]
y = D[:, 2]
fig, ax = plt.subplots(figsize=(8, 6))
area = 2 * np.random.randint(50, size = D[:, 0].size)

grp_color = [[1,0,0] if i == 0 else [0,0,1] for i in y]

plt.scatter(D[:, 0], D[:, 1], c = grp_color, s = area,
            alpha = 0.5, marker = "o" )
```

- 訓練線性判別分析 (LDA) 與二次判別分析 (QDA) 模型

接著利用以下程式碼來分別訓練 LDA 模型以及 QDA 模型，並且利用函數 *Lda.score* 與 *Qda.score* 來計算兩種方法各自的訓練誤差。

```
Lda = LinearDiscriminantAnalysis(tol = 1e-6)
Lda.fit(X, y)
trainErrLDA = 1 - Lda.score(X, y)

Qda = QuadraticDiscriminantAnalysis(tol = 1e-6, store_
    covariance = True)
Qda.fit(X, y)
trainErrQDA = 1 - Qda.score(X, y)
```

- 將畫布切成網格以備進行繪圖

以下即是相關程式碼，利用 *x* 以及 *y* 來將畫布切割成網格，並利用 *np.meshgrid* 來產生 *xx* 與 *yy* 的矩陣，即每一個網格交錯點的 *x* 座標以及 *y* 座標。

```
nx, ny = 100, 100
x_min, x_max = plt.xlim()
y_min, y_max = plt.ylim()
x_ = np.linspace(x_min, x_max, nx)
y_ = np.linspace(y_min, y_max, ny)
xx, yy = np.meshgrid(x_, y_)
```

- 計算後驗機率

接著利用以下程式碼計算每一個網格交錯點座標的機率，即 $P(G = K | X = x)$ ，為了進行計算，會先利用函數 *ravel()* 將 *xx* 與 *yy* 從矩陣拉成一整條以進行

broadcasting (拉成一條線才能運算，矩陣則不能運行)。在計算好機率後，再利用函數 `reshape` 將 Z 中第一欄屬於群組 0 的機率取出並重新變回矩陣。

```
Z = Lda.predict_proba(np.c_[xx.ravel(), yy.ravel()])
Z = Qda.predict_proba(np.c_[xx.ravel(), yy.ravel()])
Z = Z[:, 1].reshape(xx.shape)
```

- 定義繪圖色系並繪製地板分界線

在此介紹兩種繪圖的方法，第一種是如果不想利用系統預設之顏色，也可以利用套件 `colors` 中的函數 `LinearSegmentColormap` 來定義自己喜歡的顏色，並利用函數 `pcolormesh` 來繪製地板分界線，結果如圖 4，其部分程式碼呈現如下。

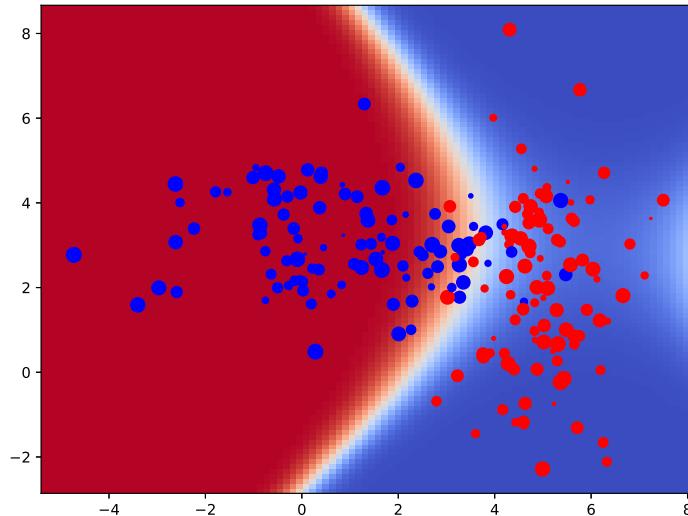


圖 4: Method of plot (1)

```
cdit = {"red": [(0, 1, 1), (1, 0.7, 0.7)],
        "green": [(0, 0.7, 0.7), (1, 0.7, 0.7)],
        "blue": [(0, 0.7, 0.7), (1, 1, 1)]}
cmap = colors.LinearSegmentedColormap("coolwarm", cdit)
plt.pcolormesh(xx, yy, Z, cmap = "coolwarm",
               norm = colors.Normalize(0., 1.),
               shading = "auto", zorder = 0)
```

第二種方式是將畫布切割成網格並對每個交叉點的座標進行預測，以下面的程式碼為例，將畫布切割出 $200 * 100$ 筆資料點，預測每一個點並將結果放在變數 Z 中，最後將這些預測結果分開並繪製散佈圖，即形成地板分界線，其結果如圖 5。

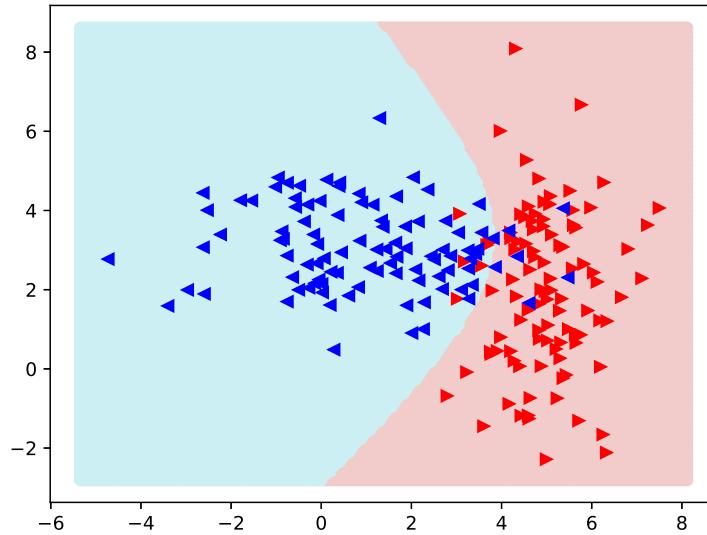


圖 5: Method of plot (2)

```

nx, ny = 200, 100
x_min, x_max = plt.xlim()
y_min, y_max = plt.ylim()
x_ = np.linspace(x_min, x_max, nx)
y_ = np.linspace(y_min, y_max, ny)
xx, yy = np.meshgrid(x_, y_)
x1, x2 = xx.ravel(), yy.ravel()
zz = Qda.predict(np.c_[x1, x2])

#colors = [ 'r' , 'b' ]
colors = ["#F2CFCB", "#CBEFF2"]
for i in range(2) :
    plt.scatter(x1[zz==i], x2[zz==i], marker="o", color=
        colors[i])

```

- 繪製分界線並比較訓練誤差

在繪製地板分界線後，接著利用以下程式碼進行繪製分類在群組 1 的機率為 0.5 的分界線，如圖 6，另外由此亦可知，線性判別分析 (LDA) 的訓練誤差為 0.09，而二次判別分析 (QDA) 的訓練誤差則為 0.065，故此例中二次判別分析 (QDA) 的分類效果比線性判別分析 (LDA) 還要好。

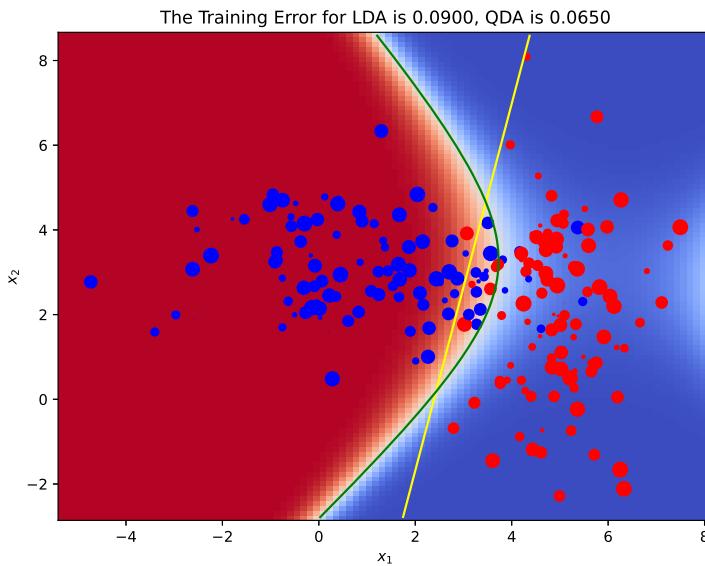


圖 6: Classification of LDA and QDA

```

contoursLDA = plt.contour(xx, yy, z, [0.5], colors =
    "yellow")
contoursQDA = plt.contour(xx, yy, z, [0.5], colors =
    "green")

```

1.5 The K-Nearest Neighbors Algorithm (KNN)

雖然在機器學習中有幾種常見的方法是利用貝氏定理來求得條件機率 $P(Y|X = x)$ ，其中 x 與 y 分別是反應變數與預測變數，但在真實世界中，很難知道此條件機率的值，因此利用貝氏定理來分類有時是不可行的，而有一些方法便嘗試去估計 $P(Y|X = x)$ ，其中包括 K-近鄰演算法 (KNN)。K-近鄰演算法透過給定 K 值以及任一測試點 x_0 ，觀察離該點 x_0 最近的 K 個點 (即 $N_k(x)$)，並將這些鄰近的 K 筆資料所對應的 y 值取平均，得到式 (18)：

$$\hat{y} = Ave(y_i | x_i \in N_k(x)) = \frac{1}{K} \sum_{x_i \in N_k(x)} y_i \quad (18)$$

且其給定 x 時 y 的條件機率如式 (19)：

$$P(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j) \quad (19)$$

1.6 Classification by KNN

大致了解 K-近鄰演算法 (KNN) 的理論後，接著一樣透過繪圖來了解其分群的效果到底如何，與線性判別分析 (LDA) 相同，KNN 一樣可以有兩種方式來配適模型，第一種是從理論來著手，第二種則是直接使用 sklearn 套件中 *neighbors* 的函數 *KNeighborsClassifier* 來配適模型，而這兩種方法都將在之後分別進行展示。

- 繪製散布圖

在繪製圖形時，首先利用下面的程式碼就資料集 *la3.txt* 的兩個變數繪製散布圖，此處嘗試利用套件 seaborn 來進行繪製，以利後續之繪圖。

```
cmap_bold = ["darkblue", "darkorange"]
Group_name = np.array(["Group A", "Group B"])
plt.figure(figsize=(8, 6))

sns.scatterplot(x = X[:, 0], y = X[:, 1], hue = Group_name
[y], palette = cmap_bold, alpha = 0.9, edgecolor = "
black")
```

- 配適 KNN 模型

一般有兩種方法用以配適 KNN 模型，此處首先先展示如何利用理論來進行配適。假設 $K = 15$ ，與線性判別分析 (LDA) 相同，利用函數 *meshgrid* 將畫布切成間距為 0.2、大小比照兩變數極大值與極小值的網格，並建立一個 for 迴圈，設立變數 *tmp* 儲存將網格的值拉成向量後再沿 y 軸複製 n 倍的資料集，每一次迴圈都會得到一個 $200 * 2$ 的矩陣。接著根據理論，計算觀察點 x_0 到每一個點的距離得到變數 *d*，並利用 *np.mean(y[idx[:K]])* 計算距離最近的 $K = 15$ 個點的均值，若此均值小於 0.5 則歸類在類別 1，若大於 0.5 則歸類在類別 2，由此即可得到該點所屬的類別。

```
K = 15
intrvl = 0.2
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, intrvl), np.
arange(y_min, y_max, 0.1))

z = np.zeros(xx.size)
for i in range(xx.size) :
    tmp = np.tile([xx.ravel()[i], yy.ravel()[i]], (n, 1))
    #d = ((tmp-X)**2).sum(axis=1) #兩種都可以
    d = np.linalg.norm(tmp - X, axis = 1)
```

```

idx = np.argsort(d)
z[i] = np.mean(y[idx[:K]])

z = [0 if i < 0.5 else 1 for i in z]

```

在配適完成 KNN 模型後，即可利用函數 `sns.scatterplot` 繪製每個網格點的類別散布圖當作地板分界線，並利用函數 `plt.contour` 繪製分界線，得到圖 7 之結果。

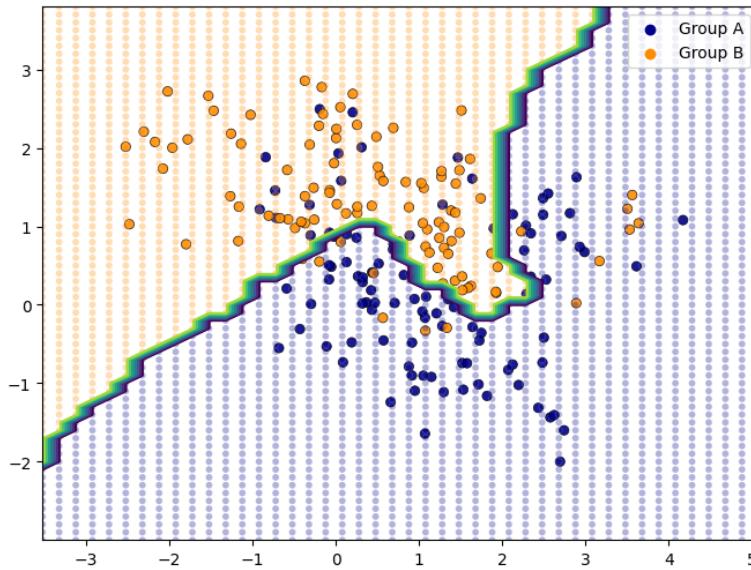


圖 7: Classification of KNN (1)

第二種方法是利用 `sklearn` 套件來配適模型，如下方之程式碼。KNN 與線性判別分析 (LDA) 與二次判別分析 (QDA) 相同，可以利用函數 `Knn.score` 來得到 KNN 模型的訓練誤差，並建立網格來預測每個網格點座標所屬之類別，最後利用函數 `plt.contourf` 得到如圖 8 之分類結果。此模型的訓練誤差為 0.155。

```

K = 15
weights = "uniform"
Knn = neighbors.KNeighborsClassifier(K, weights = weights)
Knn.fit(X, y)
trainingErr = 1 - Knn.score(X, y)
x_min, x_max = X[:,0].min() - 1, X[:,0].max() + 1
y_min, y_max = X[:,1].min() - 1, X[:,1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), \
np.arange(y_min, y_max, 0.1))
z = Knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = z.reshape(xx.shape)

```

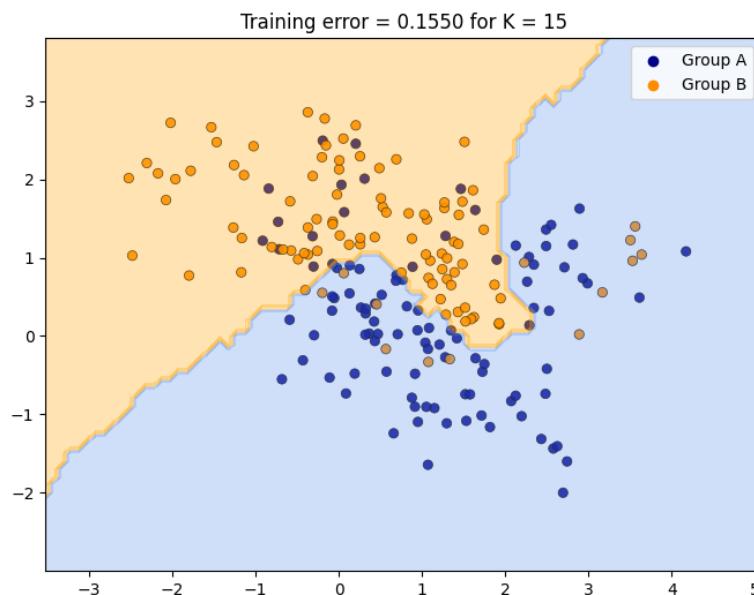


圖 8: Classification of KNN (2)

```
cmap_light = ListedColormap(["cornflowerblue", "orange"])
plt.contourf(xx, yy, Z, cmap = cmap_light, alpha = 0.3)
```

為了更好的理解如何設定 K 值以得到最佳的 KNN 分類模型，接著將分別模擬兩群、三群的雙變量常態資料，並觀察在 $k = 1$ 到 $k = 30$ 之間，利用 Bootstrapping 方法重複重樣 100 次所得到的平均訓練誤差以及平均測試誤差的變化。首先介紹分為兩群的雙變量常態資料，在此範例中，將原始資料集切割成 80% 的訓練資料集以及 20% 的測試資料集，且兩個變量的模擬樣本數設為 $n_1 = n_2 = 1000$ ，並設置各變量的平均數、樣本數與共變異數分別為：

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

為了協助更好的理解此雙變量常態資料，將分別對此雙變量常態資料繪製 2D 散布圖以及 3D 的圖形，其結果如圖 9 以及圖 10。

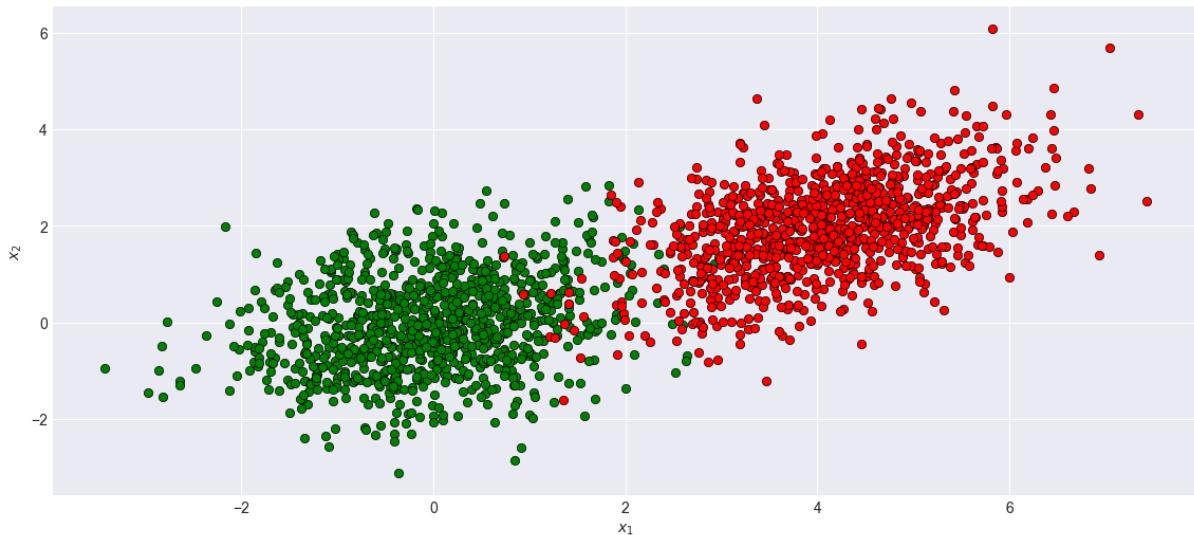


圖 9: 2D plot of simulation of two classes

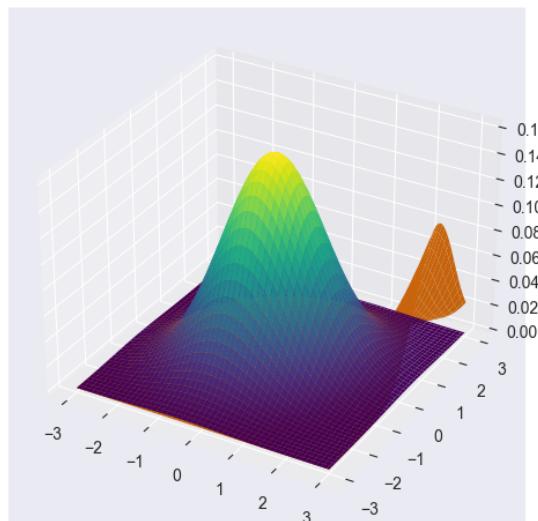


圖 10: 3D plot of simulation of two classes

透過圖 9 以及圖 10 可以明顯看出兩生成之模擬資料分配的不同之處。圖 9 中綠色的散布點是來自於平均數為 μ_1 且共變異數為 Σ_1 的雙變量常態資料，紅色的散布點則來自於平均數為 μ_2 且共變異數為 Σ_2 的雙變量常態資料。圖 10 中橘色的部分來自於平均數為 μ_2 且共變異數為 Σ_2 的雙變量常態資料，另一個則是來自於平均數為 μ_1 且共變異數為 Σ_1 的雙變量常態資料。

接著介紹生成的三群雙變量常態資料，其平均數與變異數分別為：

$$\mu'_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu'_2 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \mu'_3 = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$$

$$\Sigma'_1 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma'_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \Sigma'_3 = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

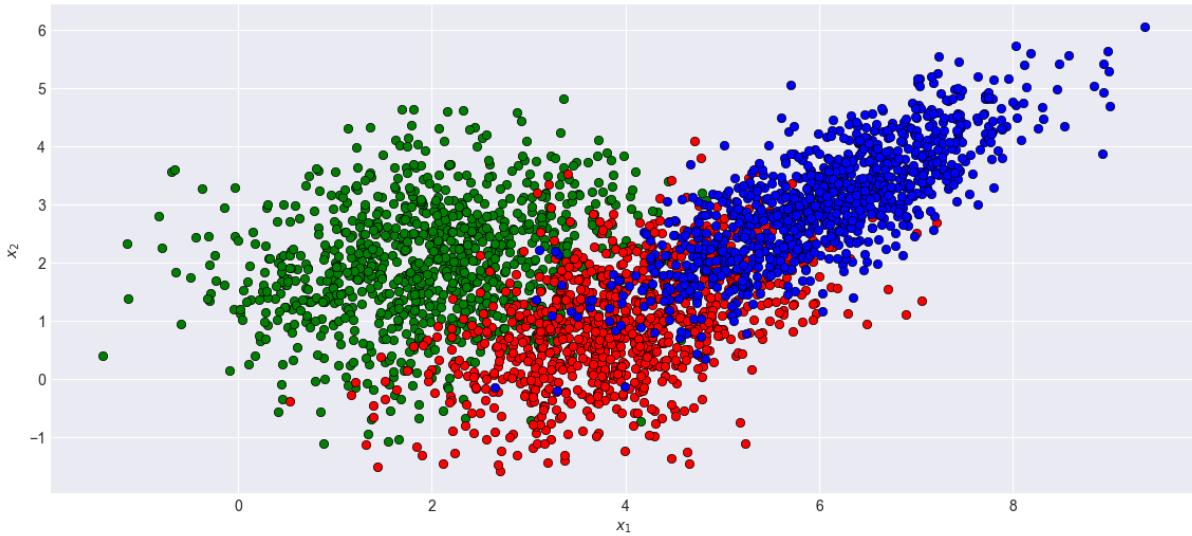


圖 11: 2D plot of simulation of three classes

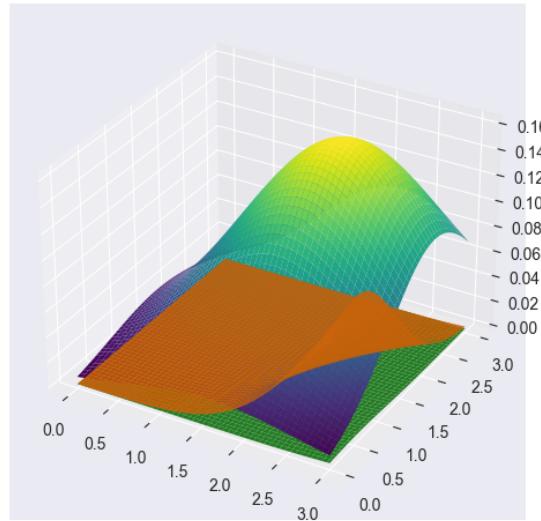


圖 12: 3D plot of simulation of three classes

圖 11 是三群雙變量常態資料的散布圖，綠色點是平均數為 μ'_1 、共變異數為 Σ'_1 的雙變量常態資料，紅色點是平均數為 μ'_2 、共變異數為 Σ'_2 的雙變量常態資料，藍色點則為參數為 μ'_3 與 Σ'_3 的雙變量常態資料。由圖 11 與圖 12 可以看出，在 2D 圖中很多點會混在一起，似乎難以找到適合的分界線來分開三群不同類別的資料點，但如果畫成 3D 圖就能較為清楚的看出三組資料的差異。以下呈現部分繪製 3D 圖的程式碼。

```

plt.style.use('seaborn-dark')
plt.rcParams['figure.figsize']=14, 6
fig = plt.figure()
sigma_1, sigma_2 = Cov1[0,0], Cov1[1,1]
x = np.linspace(-3*sigma_1, 3*sigma_1, num=100)
y = np.linspace(-3*sigma_2, 3*sigma_2, num=100)
X, Y = np.meshgrid(x,y)

pdf = np.zeros(X.shape)
for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        pdf[i,j] = mvn1.pdf([X[i,j], Y[i,j]])
key = 131
ax = fig.add_subplot(key, projection = '3d')
ax.plot_surface(X, Y, pdf, cmap = 'viridis')
plt.tight_layout()

```

接著利用 μ_1 、 μ_2 、 Σ_1 與 Σ_2 來對兩群雙變量常態資料配適 KNN 模型，並利用 μ'_1 、 μ'_2 、 μ'_3 、 Σ'_1 、 Σ'_2 與 Σ'_3 對三群雙變量常態資料配適 KNN 模型。以下面的程式碼為例，建立兩個全為 0 的集合 $Errortrain$ 與 $Errortest$ 來裝 $K = 1$ 到 $K = 30$ 時 KNN 模型 100 次模擬的平均訓練誤差與平均測試誤差，另外，利用全為 0 的集合 $KNNtrainingError$ 以及 $KNNtestingError$ 來存取 100 次 Bootstrapping 抽樣所得到的 100 個訓練誤差與測試誤差，最後分別得到如下圖 13 之結果。

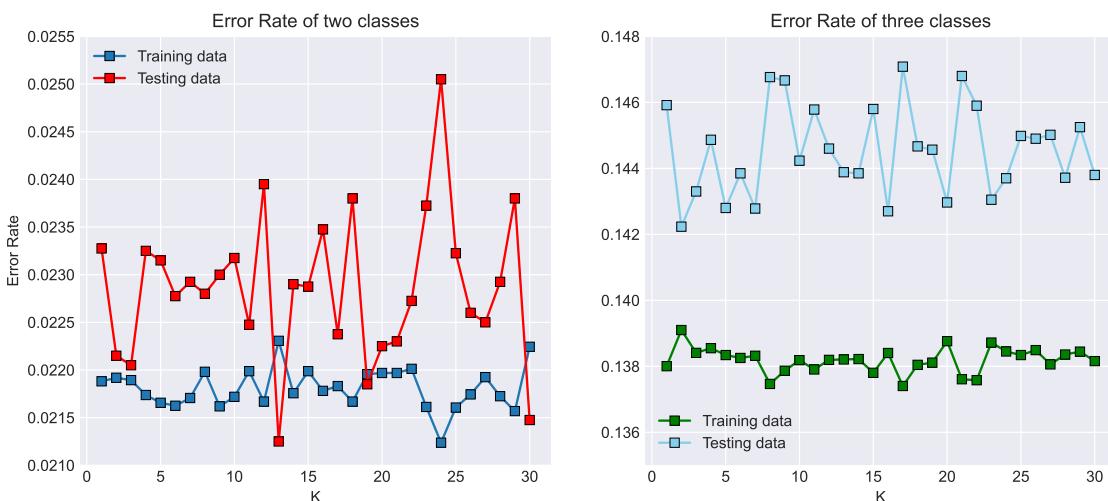


圖 13: Classification of two classes and three classes by KNN

根據圖 13 可知，在對兩群雙變量常態資料配適分類模型時， $K = 13$ 、 $K = 19$ 與 $K = 30$ 的測試誤差是 30 個 KNN 模型中最小的三個，而在三群雙變量常態資料中， $K = 2$ 、 $K = 7$ 與 $K = 16$ 則能得到較好的結果，其程式碼呈現如下。此處有個可改進

之處是此迴圈在兩群雙變量資料花了 3 分鐘的時間才跑完，在此三群雙變量資料的分群中更是花了 10 分鐘，故有待研究更快速更有效的方法來求得此結果。

```
N = 100
K = 30
Errortrain = np.zeros(K)
Errortest = np.zeros(K)
weights = "uniform"
for j in range(K):
    Knn = neighbors.KNeighborsClassifier(K, weights = weights)
    KNNtrainingError = np.zeros(N)
    KNNtestingError = np.zeros(N)
    for i in range(N) :
        X_train, X_test, y_train, y_test = train_test_split(X,
            y, test_size = 0.2)
        Knn.fit(X_train, y_train)
        KNNtrainingError[i] = 1 - Knn.score(X_train, y_train)
        Knn.predict(X_test)
        KNNtestingError[i] = 1 - Knn.score(X_test, y_test)
    Errortrain[j] = KNNtrainingError.mean()
    Errortest[j] = KNNtestingError.mean()
```

最後嘗試繪製利用 KNN 模型 ($K = 13$) 與 KNN 模型 ($K = 30$) 分類兩群雙變量資料的分界線圖，以及利用 KNN 模型 ($K = 2$) 與 KNN 模型 ($K = 16$) 分類此三群雙變量資料的分界線圖，其結果如下圖 14 與 15。

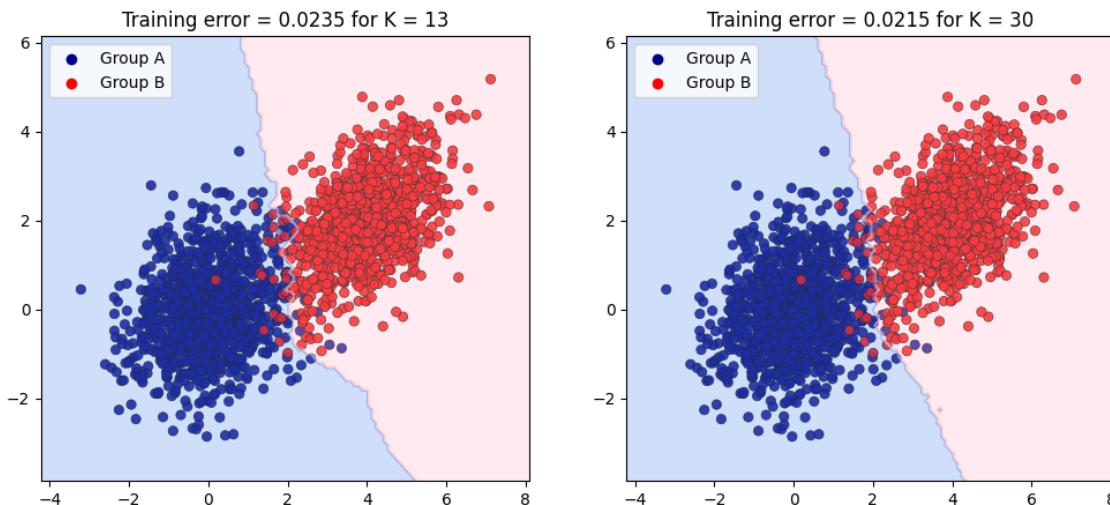


圖 14: Classification plot of simulation of two classes

根據圖 14 可知 KNN 模型 ($K = 13$) 在此模擬之兩群雙變量常態資料的分類訓練誤差為 0.0235，在 $K = 30$ 時的訓練誤差為 0.0215。接著看在三群雙變量常態資料時 KNN 模型的分類效果。

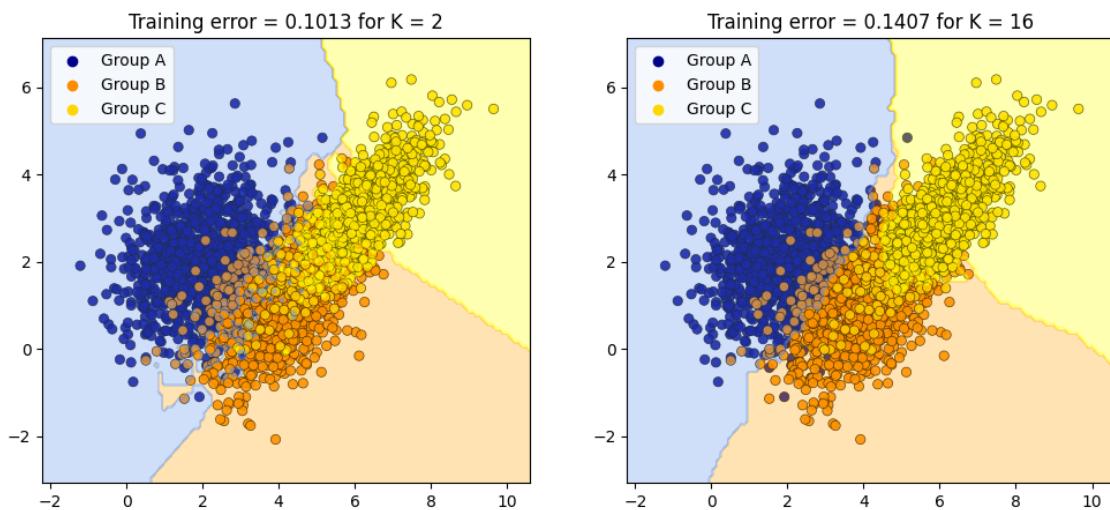


圖 15: Classification plot of simulation of three classes

根據圖 15 可知，KNN 模型在 $K = 2$ 時對三群雙變量常態資料的訓練誤差為 0.1013，在 $K = 16$ 時的訓練誤差則為 0.1407。

2 The Simulation of Classification

為了更了解線性判別分析 (LDA)、二次判別分析 (QDA) 與 K-近鄰演算法在分類二群與三群資料時的表現，在此節中將分別模擬兩群與三群的雙變量常態資料，並透過改變各自的樣本數大小、平均數與共變異數來觀察三種方法的訓練誤差與測試誤差，為了更方便進行比較，先建立一個函數來儲存所有欲更改的參數，該函數部分程式碼呈現如下，接著便先就分類兩群雙變量常態資料的情況下進行討論，再討論分類三群雙變量常態資料的情況。

```
def param(n1, n2, mean1, mean2, mean3, mean4, val1, val2):
    m1, m2 = np.array([mean1, mean2]), np.array([mean3, mean4])
    Cov1 = np.array([[1, val1], [val1, 1]])
    Cov2 = np.array([[1, val2], [val2, 1]])
    mvn1 = multivariate_normal(mean = m1, cov = Cov1)
    mvn2 = multivariate_normal(mean = m2, cov = Cov2)
    A = mvn1.rvs(n1)
    B = mvn2.rvs(n2)
    Xvar = np.vstack((A, B))#2000*1矩陣\
    y = np.hstack((np.zeros(n1), np.ones(n2)))#2000
    param = np.c_[Xvar, y]
    return param
```

2.1 Two groups of simulation of LDA, QDA, and KNN

- 改變樣本數大小 (sample size)

首先先嘗試改變樣本數大小來對三種方法的訓練誤差與測試誤差進行觀察。假設二群雙變量常態資料的參數分別如下，且其散佈圖如圖 16。

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}$$

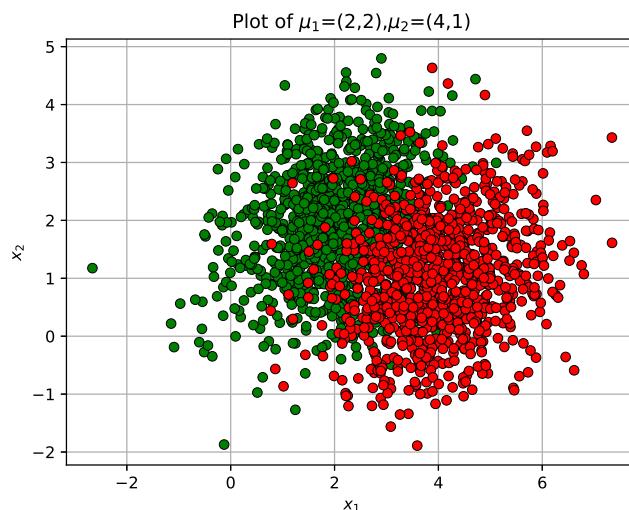


圖 16: Simulation of two classes for changing sample size

根據圖 16 可以大致觀察到此兩群雙變量常態資料的關係，接著嘗試建立 LDA、QDA、以及 $K = 5$ 與 $K = 10$ 時的 KNN 在樣本數分別為 $n_1 = n_2 = 200$ 、 $n_1 = n_2 = 500$ 、 $n_1 = n_2 = 1000$ 以及 $n_1 = 1000$ 且 $n_2 = 500$ 的模型，進而得到以下表 1 的結果，其部分程式碼亦於下面呈現。

```

n = 100
K1 = 5
weights = "uniform"
for i in range(n) :
    X_train, X_test, y_train, y_test = train_test_split(X,
                y, test_size = 0.2)
    Lda.fit(X_train, y_train)
    Lda.predict(X_test)
    LDAtestingError[i] = 1 - Lda.score(X_train, y_train)

```

```

LDAtestingError[i] = 1 - Lda.score(X_test, y_test)
Qda.fit(X_train, y_train)
Qda.predict(X_test)
QDATrainingError = 1 - Qda.score(X_train, y_train)
QDAtestingError = 1 - Qda.score(X_test, y_test)
Knn = neighbors.KNeighborsClassifier(K1, weights =
    weights)
Knn.fit(X_train, y_train)
Knn.predict(X_test)
KNNtrainingError1 = 1 - Knn.score(X_train, y_train)
KNNtestingError1 = 1 - Knn.score(X_test, y_test)

```

表 1: Error Rate of three Methods of changing sample size for two classes

	LDA		QDA		KNN($K = 5$)		KNN($K = 10$)	
	Train	Test	Train	Test	Train	Test	Train	Test
$n_1 = n_2 = 200$	0.1261	0.1275	0.1188	0.15	0.0969	0.1875	0.1125	0.16
$n_1 = n_2 = 500$	0.1049	0.1056	0.1075	0.105	0.1012	0.095	0.1112	0.1
$n_1 = n_2 = 1000$	0.1066	0.1088	0.1081	0.115	0.1	0.14	0.1	0.11
$n_1 = 1000, n_2 = 500$	0.0999	0.0992	0.1025	0.0967	0.0808	0.13	0.0958	0.12

為了計算三種機器學習方法在不同樣本數下的訓練誤差 (Training Error) 與測試誤差 (Testing Error)，建立一個 for 迴圈同時對模擬的兩群雙變量常態資料配適三種模型，並各自利用 Bootstrapping 方法來將資料切割成訓練資料與測試資料 100 次，並計算這 100 次的平均訓練誤差與測試誤差，最後得到如表 1 之結果。在表 1 中將測試誤差最小的標為紅字，而誤差最大的模型則標為粗體，根據其結果可知，在兩群模擬之雙變量常態資料的樣本數都只有 200 時，KNN 模型 ($K = 5$) 的訓練誤差最小，為 0.0969，但 LDA 模型的測試誤差才是最小的，只有 0.1275，其在樣本數為 1000 時的測試誤差也最小，只有 0.1088。

- 改變平均數向量 (mean vector)

接著嘗試設置兩群雙變量常態資料的樣本數為 $n_1 = n_2 = 1000$ ，變異數皆為 0.2，並透過更改平均數向量來了解三種模型的分類表現。除了圖 16 的模擬資料，亦設置兩組新的模擬資料來進行比較，其參數如下且其散布圖如圖 17。

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mu_1' = \begin{bmatrix} 2 \\ 6 \end{bmatrix}, \mu_2' = \begin{bmatrix} 5 \\ 9 \end{bmatrix}$$

$$\Sigma_1 = \Sigma_1' = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_2 = \Sigma_2' = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}$$

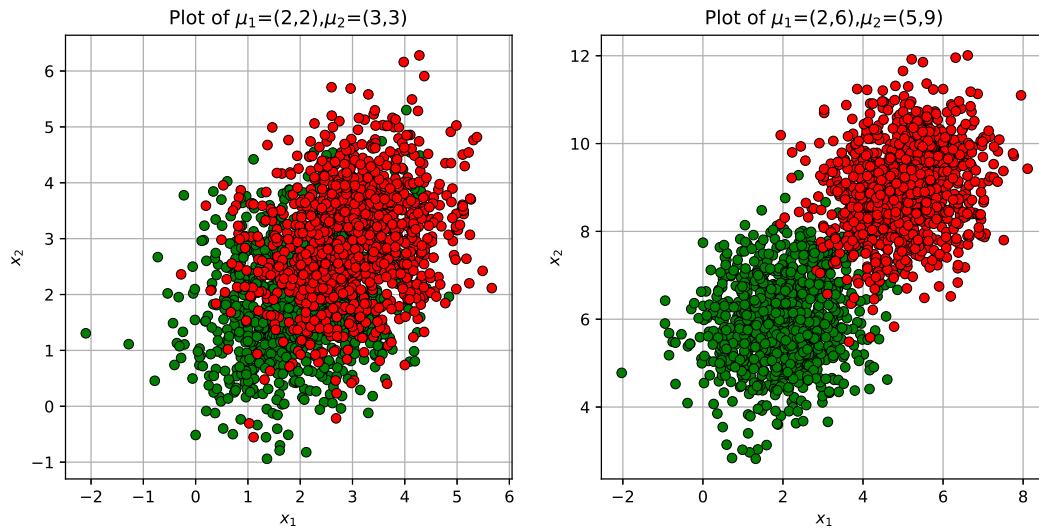


圖 17: Simulation of two classes for changing mean vector

在圖 17 中，左圖為平均數為 μ_1 與 μ_2 、共變異矩陣為 Σ_1 與 Σ_2 的雙變量常態散布圖，右圖則為平均數為 μ'_1 與 μ'_2 、共變異矩陣為 Σ'_1 與 Σ'_2 的雙變量常態散布圖。接著觀察利用三種方法配適此兩種雙變量常態資料以及圖 16 之模擬資料的平均訓練誤差與測試誤差，其結果如表 2。根據表 2 可知四種模型各有優劣的表現，其所能得到的分類效果相差不大。

表 2: Error Rate of three Methods of changing mean vector for two classes

	LDA		QDA		KNN($K = 5$)		KNN($K = 10$)	
	Train	Test	Train	Test	Train	Test	Train	Test
$\mu_1 = (2, 2), \mu_2 = (4, 1)$	0.1066	0.1088	0.1081	0.115	0.1	0.14	0.1	0.11
$\mu_1 = (2, 2), \mu_2 = (3, 3)$	0.2687	0.27	0.2681	0.2625	0.2125	0.3175	0.2525	0.2825
$\mu_1 = (2, 6), \mu_2 = (5, 9)$	0.018	0.0175	0.02	0.015	0.0156	0.015	0.0162	0.015

- 改變共變異矩陣 (covariance matrix)

在 LDA 模型中有個前提假設是資料服從常態且每群的共變異數相同，但在實務資料中往往很難服從此前提假設，因此嘗試觀察改變共變異數時四種模型的訓練誤差與測試誤差發生的改變。我們設置幾種參數如下並觀察各自的模擬資料散布圖 18。

$$\mu_1 = \mu'_1 = \mu''_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \mu'_2 = \mu''_2 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$\Sigma'_1 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma'_2 = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

$$\Sigma''_1 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}, \Sigma''_2 = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}$$

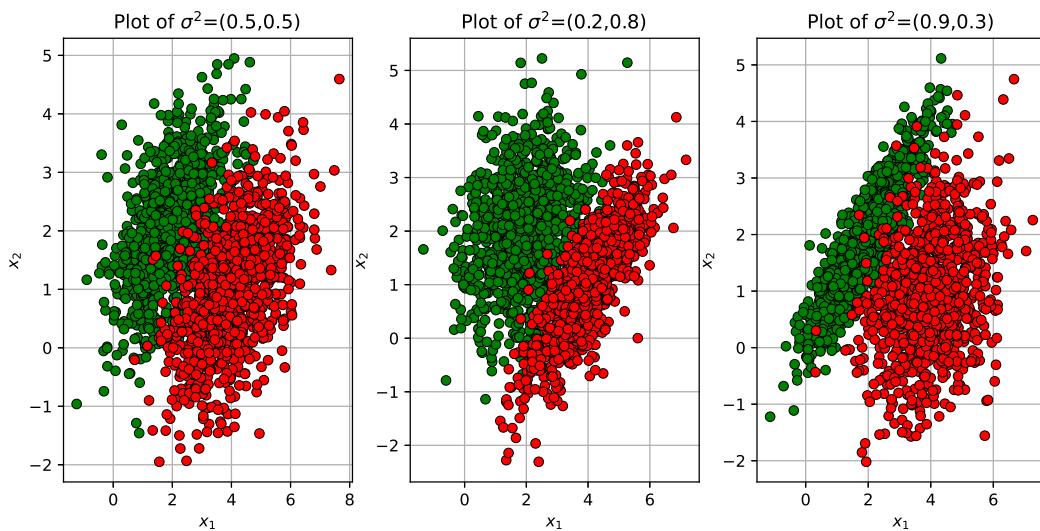


圖 18: Simulation of two classes for changing variance

根據圖 18 可以觀察平均數分別為 $(2, 2)$ 、 $(4, 1)$ 而共變異矩陣為 Σ_1 與 Σ_2 、 Σ'_1 與 Σ'_2 以及 Σ''_1 與 Σ''_2 的常態資料的散布圖。另外亦將圖 16 中平均數為 $(2, 2)$ 、 $(4, 1)$ 而變異數為 $(0.2, 0.2)$ 的雙變量常態資料列入比較並計算其訓練誤差與測試誤差，其結果如表 3。

表 3: Error Rate of three Methods of changing variance for two classes

	LDA		QDA		KNN($K = 5$)		KNN($K = 10$)	
	Train	Test	Train	Test	Train	Test	Train	Test
$\sigma^2 = (0.2, 0.2)$	0.1066	0.1088	0.1081	0.115	0.1	0.14	0.1	0.11
$\sigma^2 = (0.5, 0.5)$	0.0728	0.076	0.0725	0.085	0.0613	0.0725	0.0669	0.07
$\sigma^2 = (0.2, 0.8)$	0.0715	0.0726	0.0537	0.06	0.05	0.065	0.0506	0.06
$\sigma^2 = (0.9, 0.3)$	0.0547	0.0545	0.0262	0.025	0.0212	0.03	0.0288	0.03

接著透過表 3 可以觀察四種模型的訓練誤差與測試誤差，可以發現在變異數相同，即 $\sigma^2 = (0.2, 0.2)$ 與 $\sigma^2 = (0.5, 0.5)$ 時，QDA 模型與 KNN ($K = 5$) 的測試誤差最大，而 LDA 模型的表現算是不錯，但是在變異數不同時，LDA 模型的測試誤差明顯比其他三種模型要高得多，尤其在 $\sigma^2 = (0.9, 0.3)$ 時，其他三種模型的測試誤差最高只到 0.03，LDA 模型的測試誤差則達到 0.0545。

2.2 Three groups of simulation of LDA, QDA, and KNN

在觀察過改變樣本數大小、平均數向量以及共變異矩陣的二群雙變量常態資料後，接著嘗試改變三群雙變量常態資料的樣本數、平均數與共變異並同樣觀察 LDA 模型、QDA 模型、KNN 模型 ($K = 5$) 與 KNN 模型 ($K = 10$) 四種模型的訓練誤差與測試誤差進行比較，首先先觀察改變樣本數大小的情況。

- 改變樣本數大小 (sample size)

此處將設置三群雙變量常態資料的參數如下，並比較 $n_1 = n_2 = n_3 = 200$ 、 $n_1 = n_2 = n_3 = 500$ 、 $n_1 = n_2 = n_3 = 1000$ 以及 $n_1 = 1000$ 、 $n_2 = 500$ 且 $n_3 = 200$ 四種樣本數的雙變量常態資料， $n_1 = n_2 = n_3 = 1000$ 的散布圖亦呈現於圖 19。

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \mu_3 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}$$

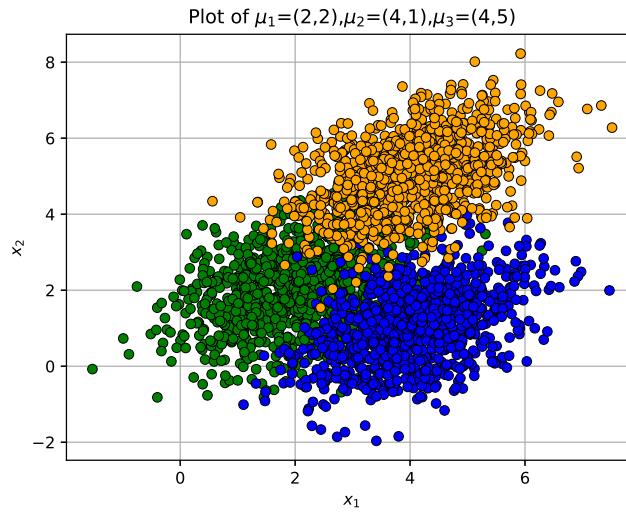


圖 19: Simulation of three classes for changing sample size

圖 19 即為此三群雙變量常態資料的散布圖，其中綠色點是平均數為 μ_1 共變異數為 Σ_1 的資料點、藍色點是平均數為 μ_2 共變異數為 Σ_2 的資料點， μ_3 與 Σ_3 則是黃色雙變量常態資料的參數，其配適四種模型的誤差如表 4。

表 4: Error Rate of three Methods of changing sample size for three classes

	LDA		QDA		KNN($K = 5$)		KNN($K = 10$)	
	Train	Test	Train	Test	Train	Test	Train	Test
$n_1 = n_2 = n_3 = 200$	0.3876	0.4095	0.3896	0.3833	0.2667	0.3333	0.3521	0.4
$n_1 = n_2 = n_3 = 500$	0.3889	0.4032	0.3717	0.4	0.2892	0.3833	0.315	0.42
$n_1 = n_2 = n_3 = 1000$	0.399	0.4105	0.3942	0.415	0.2825	0.395	0.3129	0.4133
$n_1 = 1000, n_2 = 500, n_3 = 200$	0.2084	0.2081	0.2022	0.2294	0.1801	0.2382	0.1958	0.25

根據表 4 可以發現，KNN 模型 ($K = 5$) 在樣本數為 200、500 與 1000 時的測試誤差都最小，而在三群資料的樣本數不同時，則是 LDA 模型表現最好。

- 改變平均數向量 (mean vector)

在觀察過改變樣本數時四種模型的誤差後，接著觀察在三群資料的樣本數都是 1000 時，平均數與共變異數如下之三群雙變量常態資料配是四種模型的誤差，此結果會與圖 19 的結果一起進行比較。

$$\mu_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \mu_3 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$\mu'_1 = \begin{bmatrix} 1 \\ 6 \end{bmatrix}, \mu'_2 = \begin{bmatrix} 2 \\ 5 \end{bmatrix}, \mu'_3 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$\Sigma_1 = \Sigma'_1 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_2 = \Sigma'_2 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_3 = \Sigma'_3 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$$

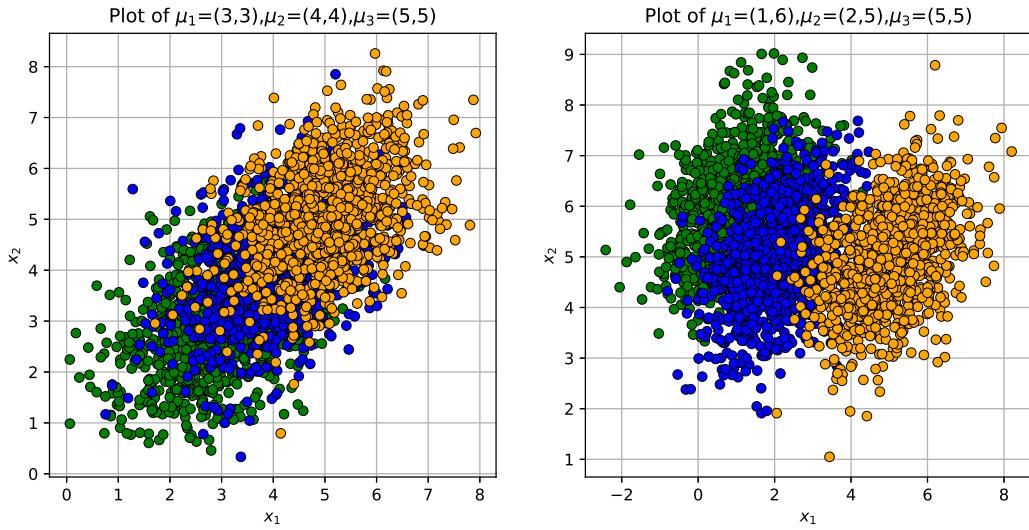


圖 20: Simulation of three classes for changing mean vector

圖 20 展示了上方參數的資料散布圖。其左圖展示了參數為 μ_1 、 μ_2 、 μ_3 與 Σ_1 、 Σ_2 以及 Σ_3 的三群雙變量常態資料，右圖則展示了參數為 μ'_1 、 μ'_2 、 μ'_3 與 Σ'_1 、 Σ'_2 以及 Σ'_3 的三群雙變量常態資料，其訓練誤差與測試誤差如表 5。

表 5: Error Rate of three Methods of changing mean vector for three classes

	LDA		QDA		KNN($K = 5$)		KNN($K = 10$)	
	Train	Test	Train	Test	Train	Test	Train	Test
$\mu_1 = (2, 2), \mu_2 = (4, 1), \mu_3 = (4, 5)$	0.399	0.4105	0.3942	0.415	0.2825	0.395	0.3129	0.4133
$\mu_1 = (3, 3), \mu_2 = (4, 4), \mu_3 = (5, 5)$	0.4963	0.5081	0.5046	0.515	0.3571	0.5317	0.4087	0.525
$\mu_1 = (1, 6), \mu_2 = (2, 4), \mu_3 = (5, 5)$	0.497	0.5099	0.4829	0.5117	0.3579	0.5383	0.3992	0.5333

根據表 5 可知，LDA 模型在 $\mu_1 = (3, 3), \mu_2 = (4, 4), \mu_3 = (5, 5)$ 以及 $\mu_1 = (1, 6), \mu_2 = (2, 4), \mu_3 = (5, 5)$ 的測試誤差最低，但在 $\mu_1 = (2, 2), \mu_2 = (4, 1), \mu_3 = (4, 5)$

時則是 KNN 模型 ($K = 5$) 的測試誤差最低，因此並沒有在所有改變平均數的模擬資料表現最佳之模型。

- 改變共變異矩陣 (covariance matrix)

接著再嘗試改變共變異數矩陣來比較四種模型的誤差，LDA 模型設定了資料共變異數相同的假設，因此期望能看到在改變共變異數時，QDA 模型的誤差一定會比 LDA 模型還低之結果。故設置以下參數之三群雙變量常態資料來進行此觀察，其資料散布圖如圖 21，並與兩種 KNN 模型進行比較。

$$\mu_1 = \mu'_1 = \mu''_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \mu'_2 = \mu''_2 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \mu_3 = \mu'_3 = \mu''_3 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$$

$$\Sigma'_1 = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}, \Sigma'_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}, \Sigma'_3 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$$

$$\Sigma''_1 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}, \Sigma''_2 = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}, \Sigma''_3 = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}$$

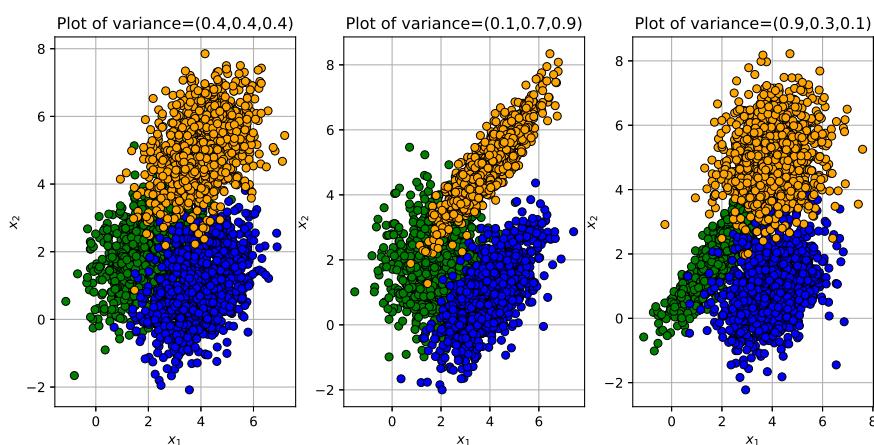


圖 21: Simulation of three classes for changing variance

圖 21 分別表示了三群樣本數皆為 1000、平均數為 (2, 2)、(4, 1) 與 (4, 5) 之三群雙變量常態資料，其變異數分別為 (0.4, 0.4, 0.4)、(0.1, 0.7, 0.9) 與 (0.9, 0.3, 0.1)，而其配適四種模型的訓練誤差與測試誤差呈現於表 6。

表 6: Error Rate of three Methods of changing variance for three classes

	LDA		QDA		KNN($K = 5$)		KNN($K = 10$)	
	Train	Test	Train	Test	Train	Test	Train	Test
$\sigma^2 = (0.2, 0.2, 0.2)$	0.399	0.4105	0.3942	0.415	0.2825	0.395	0.3129	0.4133
$\sigma^2 = (0.4, 0.4, 0.4)$	0.3804	0.3952	0.3646	0.4183	0.2608	0.4217	0.3087	0.425
$\sigma^2 = (0.1, 0.7, 0.9)$	0.3674	0.372	0.3579	0.37	0.2458	0.3783	0.2879	0.3717
$\sigma^2 = (0.9, 0.3, 0.1)$	0.3609	0.3788	0.3488	0.3533	0.2317	0.3467	0.2775	0.345

根據表 6 可知，在改變共變異數前，LDA 模型的測試誤差都比 QDA 模型還低，而在改變共變異數後，LDA 模型的測試誤差都變得比 QDA 模型還高，尤其在 $\sigma^2 = (0.9, 0.3, 0.1)$ 時，LDA 模型的測試誤差甚至比兩種 KNN 模型都還要高。另外，KNN 模型 ($K = 5$) 在改變共變異數前的測試誤差都比 KNN 模型 ($K = 10$) 還低，而在改變共變異數後則是 KNN 模型 ($K = 10$) 的表現都比 KNN 模型 ($K = 5$) 還要好。

3 Discussion

在此文中介紹了三種機器學習方法，線性判別分析 (LDA)、二次判別分析 (QDA) 與 K-近鄰演算法的理論並介紹如何利用 Python 配適模型並繪圖進行觀察。為了了解三種模型的優劣，本文亦利用 Bootstrapping 方法模擬了兩群與三群的雙變量常態資料對模型的訓練誤差與測試誤差進行比較，期望能協助更好的了解三種模型的分類效果。