迴歸分析與羅吉斯迴歸的分類問題

郭翊萱 711133115

2022.11

在機器學習中，迴歸分析方法是最常見的方法之一，但大多應用於預測定量資料的情況，而在本文中，將利用一般資料集探討在反應變量是定性資料的情況下如何利用 Simple Linear Regression 以及 Augmented Regression 來準確地進行二元分類，並利用各種模擬的資料來更好的比較兩種不同迴歸分析方法的效能。另外，本文也將探討如何利用羅吉斯迴歸 (Logistic Regression) 分類多類別的資料。

1 Introduction to Machine Learning

機器學習 (Machine Learning) 是一套用以理解數據的龐大工具，其工具主要分為兩大類，即監督式學習 (Supervised Learning) 與非監督式學習 (Unsupervised Learning)。一般來說，監督式學習與非監督式學習有不同的用途: 監督式學習一般用於建立預測統計模型，或者利用一個或多的輸入變量 (input variable) 來估計某個輸出變量 (output variable)。而在非監督式學習中，則大多指定輸入變量卻不指定輸出變量。假設觀察到一個定量的輸出變量 Y 以及 p 個不同的輸入變量 $X_1, X_2, ..., X_p$，若 Y 與 $X = (X_1, X_2, ..., X_p)$ 有關係，則可以將兩者寫作一個表達式:

$$Y = f(X) + \varepsilon \tag{1}$$

其中 f 是 $X_1, X_2, ...X_p$ 的函數，ε 是均值為 0 且與 X 獨立的隨機誤差項。而機器學習就是用來估計 f 的一系列方法。

1.1 Supervised Learning and Unsupervised Learning

接著我們更詳細的介紹監督式學習與非監督式學習的差異。監督式學習 (Supervised Learning) 一般通過建立預測變量 (Predictor variable) 與反應變量 (Response) 之間的關係，精準預測反應變量或更好的理解兩者之間的關係。許多傳統統計學習方法，比如線性迴歸 (Linear Regression)、羅吉斯迴歸 (Logistic Regression)，以及廣義可加模型 (

GAM)、提升法 (Boosting) 與支持向量機 (SVM) 都屬於監督式學習方法的一種。

相反，非監督式學習 (Unsupervised Learning) 更有挑戰性。在非監督式學習中，只有預測變量 $x_i, i = 1, 2...n$ 已知，且這些變量沒有相應的反應變量 y_i 對應。由於缺乏響應變量，因此在這類問題中建立線性模型是不可能的，因此往往利用非監督式學習來理解變量之間或者觀察值之間的關係，例如聚類分析 (Cluster Analysis)。在本文中將聚焦在監督式學習中的分類方法 (Classification)，因此不對非監督式學習進行過多闡述。

1.2 Classification and Regression Problem

變量常分為定量 (Quantitive) 與定性 (Qualitative) 兩種，定量變量如年齡 (Age)、銷售量 (Sales) 或者收入 (Income) ，定性變量則是類別變量，例如性別、產品的品牌、是否違約等等。一般在建立模型時會將響應變量為定性的問題歸類到分類問題 (Classification)，而反應變量為定量的問題則歸類到迴歸問題。然而，迴歸問題與分類問題的區分並不是如此絕對，例如，羅吉斯迴歸 (Logistic Regression) 一般被認為是一種分類方法，但由於它估計了每個類別發生的機率，因此同樣被認為是一種迴歸問題; 支持向量機 (SVM) 一般被用來進行二元分類或多元分類問題，但同樣可以利用支持向量機迴歸來預測定量資料得到很好的結果。大部分機器學習方法都可以應用在迴歸問題與分類問題中，在下節中，即將利用簡單線性迴歸來嘗試處理二元分類問題。

以下簡單的介紹分類問題中模型精度的計算方式。假設建模的目標是在 $(x_1, y_1), ...(x_n, y_n)$ 尋找對 f 的估計，其中 $y_1, ...y_n$ 是定性變量。最常用的衡量估計 $\hat{f}$ 準確度的方法是計算錯誤率 (error rate)，也就是對訓練資料 (Training Data) 使用估計模型 $\hat{f}$ 所造成的誤差比例，如下所示:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{2}$$

其中 $\hat{y}_i$ 是使用 $\hat{f}$ 預測數據的第 i 個值，而如果 $I(y_i \neq \hat{y}_i) = 0$，那麼第 i 個觀測值用分類模型實現了正確分類，否則就是被誤分。

2 Simple Linear Regression Model

2.1 The Theory of Simple Linear Regression

線性迴歸 (Linear Regression) 是一種常見的監督式學習方法，是一種常見的預測定量響應變量 (Response) 的工具，它假定了 X 與 Y 之間存在線性關係，在數學上將這種關

係記為:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon_i \tag{3}$$

在上式 (**??**) 中假設存在兩個預測變量 X_1 與 X_2 並利用最小平方法期望使觀察值與估計值的誤差越小越好。其計算方法如下:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1(1) & X_2(1) \\ 1 & X_1(2) & X_2(2) \\ \vdots & & \\ 1 & X_1(n) & X_2(n) \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

透過上述矩陣即可求得最佳解:

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{4}$$

2.2 Classification of Simple Linear Regression

雖然迴歸分析一般應用於響應變量 y 是定量資料的情況，在定性資料中的表現普通，甚至只能用於二元資料分類，在本節中仍嘗試利用簡單線性迴歸方法來對二元資料進行分類模型的建立，以更了解迴歸分析方法的應用，二元資料包含響應變量為男生或女生、是否成功賣出產品等等。

利用迴歸分析方法來建立模型後，當得到新的預測變量 (x_1, x_2) 時，將首先得到 y 的估計值 $\hat{y} = x^T$，其中 $x^T = [1\ x_1\ x_2]$。在得到估計值後，若 $\hat{y} \leq 0.5$，則將該筆資料分類至第一類，若 $\hat{y} > 0.5$，則將該筆資料分類至第二類。

利用迴歸分析為二元資料建模並進行預測時，我們會遵循以下步驟:

1. 繪製散布圖: 透過繪圖了解兩個類別資料的關係。
    - 利用 $scatter$ 函數進行畫圖。
    - 先將資料分成二群再分別進行畫圖。
2. 參數估計:
    - 寫程式做最小平方法估計參數並建立迴歸模型。
    - 利用 sklearn 套件直接建立迴歸模型。

3. 畫分界線: 利用配適好的迴歸模型繪製分界線。

4. 算準確率: 計算模型繪製出的分界線的分類準確率。

2.3 The Example of Classification

繪製散布圖

在本節中，將利用資料集 $la_2.txt$ 來嘗試建立迴歸模型進行類別分類。該資料為 $200 * 4$ 的資料集，包含兩組不同類別的觀察值以及他們的類別 0 (Group A) 與 1 (Group B)。其散布圖如圖 **??**:

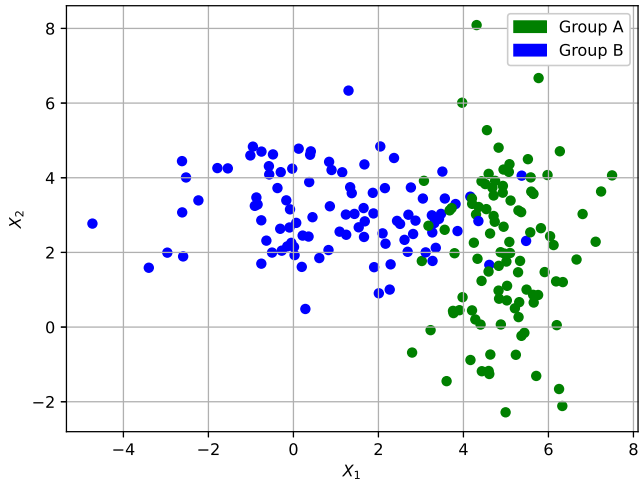

圖 1: Scatter plot of $la_2.txt$

由圖 **??** 可知，類別 0 (Group A) 的觀察值的變異相較於類別 1 (Group B) 較小，觀察點較為密集，而類別 1 (Group B) 的資料點變異較大，資料點較為分散。另外，兩組資料之間重疊的點似乎並不算多。

```
data_dir = "D:/vscodepython/Statistical Calculation/Homework4_
   Regression/"
D = np.loadtxt(data_dir + "la_2.txt", comments="%")
```

我們利用上面的程式碼將資料讀入 Python。如前面所述，一般有兩種常用的方法可以用來繪製散布圖，其一是利用 *scatter* 畫圖，其二則是先將資料分開再繪製散布圖。以下我們呈現兩種方法的部分程式碼:

```
##利用scatter函數畫圖##
colors = ["green" if i == 0 else "blue" for i in D[:,2]]
#colors = [[1,0,0] if i == 0 else [0,0,1] for i in D[:,2]]
```

```
#[R,G,B]

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

##個別分群後畫圖##
Idx = (D[:,2]==0)
plt.plot(D[Idx, 0], D[Idx, 1], "ro", alpha = 0.5, label = "
   Group A")
Idx = (D[:,2]==1)
plt.plot(D[Idx,0], D[Idx,1],"bo", alpha = 0.5, label = "Group
   B")
```

參數估計

在利用散布圖觀察完資料簡單的特性後，接著將進行參數估計並建立迴歸模型，一般同樣有兩種常用的方法，第一種是利用程式計算最小平方法所估計出的參數 $\hat{beta} = (X^T X)^{-1} X^T y$，並利用此估計參數建立迴歸模型。其部分程式碼呈現如下:

```
n = len(D[:, 0])
X = np.c_[np.ones(n), D[:, 0:2]] #不會再橫豎不分
y = D[:, 2]
b = LA.inv(X.T @ X) @ X.T @ y
#@:矩陣相乘 #inv:inverse
```

另外也可以直接利用 Python 的 sklearn 套件來建立迴歸模型，其程式碼如下:

```
Mdl = LinearRegression()
X =  D[:, 0:2]
y = D[:, 2]
n = len(y)

##配適模型
Mdl.fit(X, y)
##R-squared
R2 = Mdl.score(X, y)

##截距項
intrcp = Mdl.intercept_
##係數
coeffs = Mdl.coef_
```

畫分界線並算出準確率

建立好迴歸分析模型後，接著畫令 $y = 0.5$ 時的點 (x_1, x_2) 所形成的分界線，即令：

$$0.5 = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \tag{5}$$

接著將該式 (??) 轉化成式 (??)，並以 $X_1 = (-3, 5)$ 代入，即可求出分界線，並透過圖 ?? 觀察其分類準確度，若是迴歸模型預測的值大於 0.5，則分類至群組 1 (Group B)，若是小於 0.5，則分類至群組 0 (Group A)。透過計算可知，此迴歸模型的準確度為 91%。

$$X_2 = \frac{-(\beta_0 - 0.5 + \beta_1 * X_1)}{\beta_2} \tag{6}$$

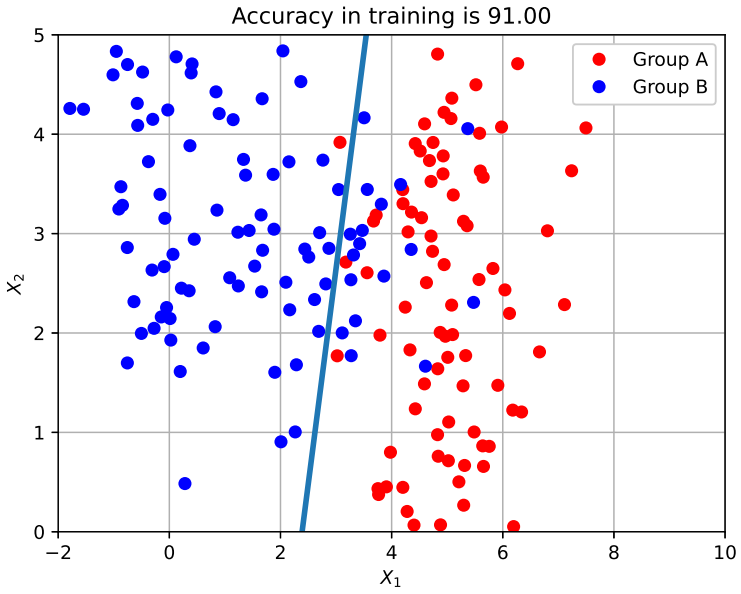

圖 2: Simple Linear Regression of $la_2.txt$

3 Augmented Regression Model

由上圖 ?? 可知，Group A 與 Group B 的點似乎重合度較高，用線性迴歸來配適模型似乎並不是一個好方法，因此可嘗試重新配適一個非線性的迴歸模型以得到較好的分類準確度。我們重新假設其迴歸模型如式 (??):

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 + \beta_5 X_2^2 \tag{7}$$

接著與配適簡單線性迴歸 (Simple Linear Regression) 相同，先利用 sklearn 套件進行參數估計建立模型，再畫出分界線算出模型分類準確率，其結果與簡單線性迴歸模型相比如下圖 ??:

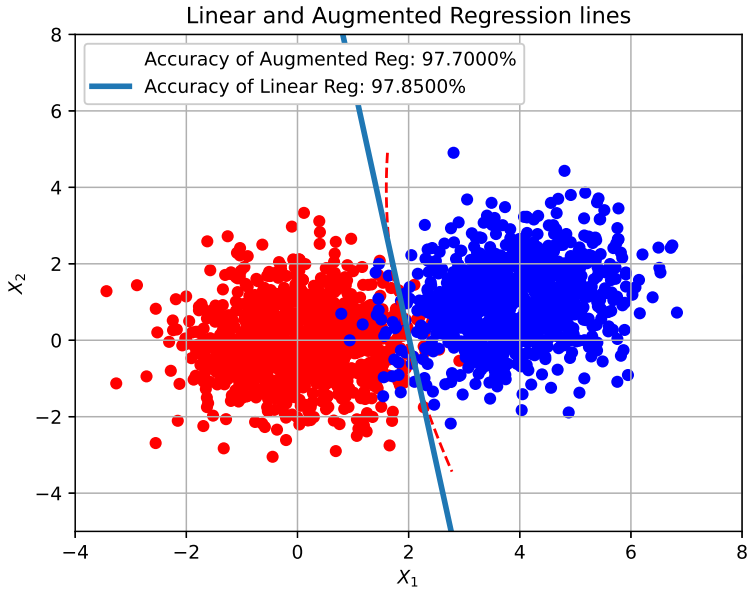

圖 3: Simple Linear and Augmented Regression of $la_2.txt$

由圖 **??** 可知，利用 Augmented Regression 來進行分類得到的準確度為 93.5 %，比利用簡單線性迴歸來分類得到的準確度 91% 來的高。部分繪製圖形程式碼呈現如下:

```
x1 = D[:,0:1]
x2 = D[:,1:2]
X = np.hstack((x1, x2, x1 * x2, x1 ** 2, x2 ** 2))
y = D[:, 2]
n = len(y)
Mdl = LinearRegression()
Mdl.fit(X, y) # 進行估計 ( 配適 )
intrcp = Mdl.intercept_
coeffs = Mdl.coef_
y_hat = Mdl.predict(X)

y_pre = [1 if i > 0.5 else 0 for i in y_hat]
x = np.array([-3, 5])
f = (
lambda x: intrcp
+ coeffs[0] * x[0]
+ coeffs[1] * x[1]
+ coeffs[2] * x[0] * x[1]
+ coeffs[3] * x[0] ** 2
+ coeffs[4] * x[1] ** 2)
xx = np.linspace(x1.min(), x1.max(), 100)
yy = np.linspace(x1.min(), x2.max(), 100)
X, Y = np.meshgrid(xx, yy) #網格:100*100的網格
Z = f([X, Y])
contours = plt.contour(
X, Y, Z, levels = [0.5], colors="red", linestyles="--", lw=3)
```

```
labels = ["Accuracy of Augmented Reg: {:.4f}%".format(100 *np.
    mean(y_pre == y))]
for i in range(len(labels)):
    contours.collections[i].set_label(labels[i])
```

4 Comparison to Regression Model

為了更好的了解簡單線性迴歸 (Simple Linear Regression) 以及加廣迴歸模型 (Augmented Regression) 的分類準確度，在此節中將模擬六組不同的資料，將其切割成 80% 的訓練資料集 (Training Data) 以及 20% 的測試資料集 (Testing Data) 並觀察其準確度。為了方便寫程式，將資料與資料的類別各自進行命名，分割資料集的程式碼呈現如下:

```
train_data , test_data , train_label , test_label = train_test
    _split(data, label, test_size=0.2)
```

接著將模擬六組不同的相依雙變量常態母體資料來觀察兩種迴歸模型方法的分類效能。

- 兩組模擬資料樣本數大小改變

  在此比較中，將分別設樣本數為 $n_1 = 200, n_2 = 200$ 以及 $n_1 = n_2 = 1000$，並維持兩雙變量常態的變數各自為:

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix},$$

  其結果如圖 **??**:

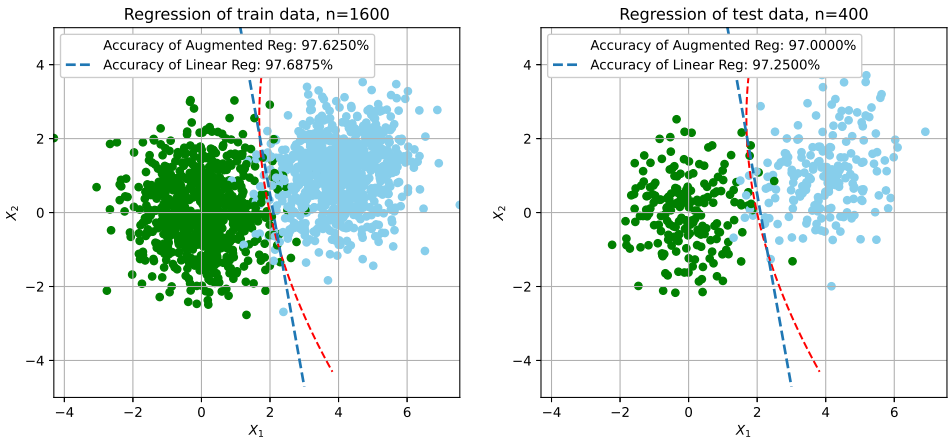

圖 4: Regression of Simulation, $n_1 = n_2 = 1000$

由圖 **??** 可知，訓練資料集有 1600 筆資料，測試資料集有 400 筆資料，且簡單線性迴歸在訓練資料與測試資料集的準確度都高於加廣迴歸模型。接著將兩組資料的樣本數皆調整至 $n_1 = n_2 = 200$。由圖 **??** 可知，簡單線性迴歸的準確度仍高於加廣迴歸模型。

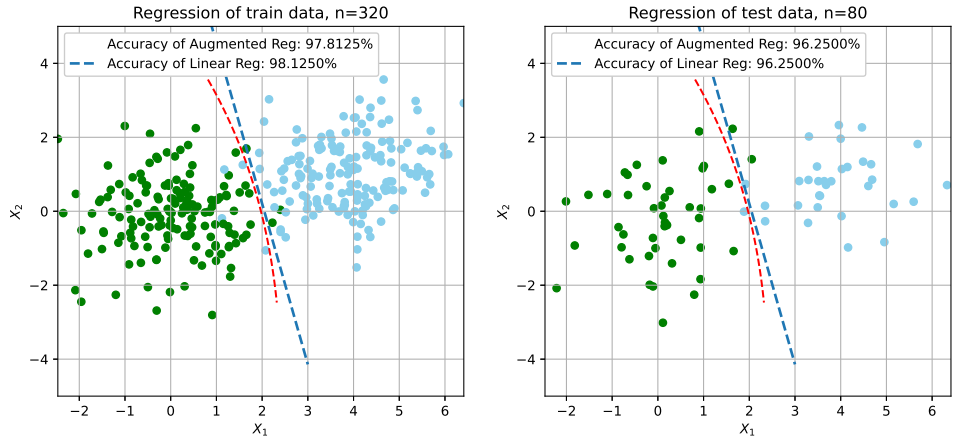

圖 5: Regression of Simulation, $n_1 = n_2 = 200$

在此我們呈現部分程式碼:

```
from scipy.stats import multivariate_normal
n1, n2 = 1000, 1000
m1, m2 = np.array([3, 3]), np.array([2, 4])
a, b = 0.4, 0.5
Cov1 = np.array([[1, a], [a, 1]])
Cov2 = np.array([[1, b], [b, 1]])

##兩組多變量常態資料
mvn1 = multivariate_normal(mean = m1, cov = Cov1)
mvn2 = multivariate_normal(mean = m2, cov = Cov2)
##多變量常態隨機變數
A, B = mvn1.rvs(n1), mvn2.rvs(n2)
##資料矩陣
D = np.vstack((A, B))
##群組值0 or 1
y = np.hstack((np.zeros(n1), np.ones(n2)))
np.savetxt("demo_data.txt", np.c_[D, y], fmt = "%.4f %.4f
    %d")
#存下資料

data_dir = "D:/vscodepython/Statistical Calculation/"
D = np.loadtxt(data_dir + "demo_data.txt", comments="%")

from sklearn import datasets
from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
data = D[:,0:2]
label = D[:, 2]

#train:1600 test:400
train_data , test_data , train_label , test_label = train_
   test_split(data, label, test_size=0.2)

#####Augmented Regression
###train_data
x1 = train_data[:,0:1]
x2 = train_data[:,1:2]
X = np.hstack((x1, x2, x1 * x2, x1 ** 2, x2 ** 2))
y = train_label
n = len(y)
```

- 兩組模擬資料平均數改變

接著嘗試改變兩雙變量常態母體的平均數來觀察分類準確度，我們設樣本數皆為 $n_1 = n_2 = 1000$，且其各自的參數為:

$$\mu_1 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \mu_2 = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix},$$

其結果如圖 **??**:

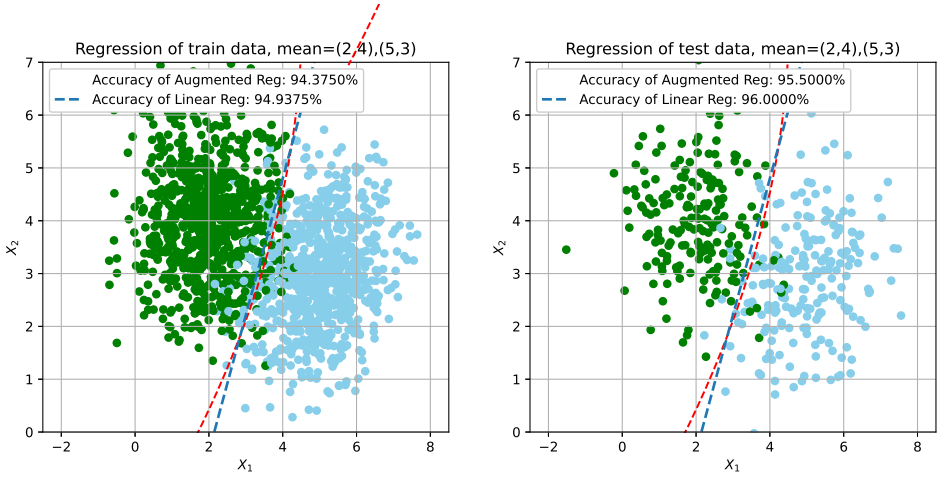

圖 6: Regression of Simulation

由圖 **??** 可知，簡單線性迴歸在訓練資料集與測試資料集表現仍稍微優於加廣迴歸模型。接著我們將母體參數修正為:

$$\mu_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \mu_2 = \begin{bmatrix} 6 \\ 5 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix},$$

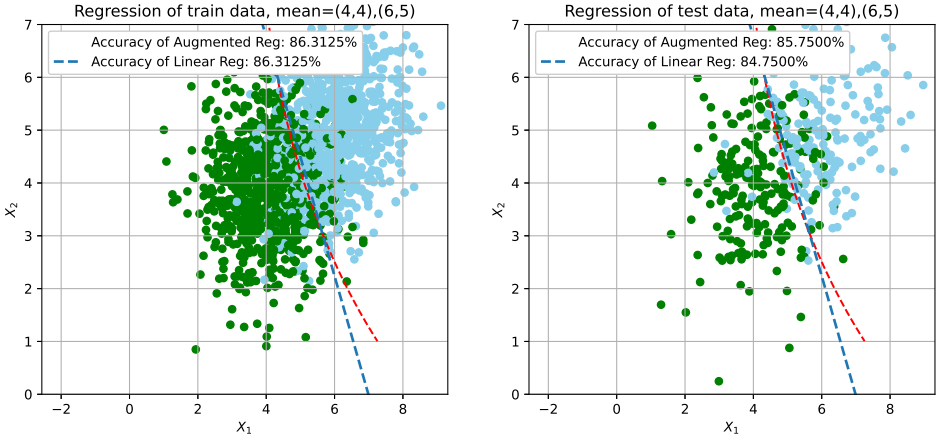

圖 7: Regression of Simulation

由圖 **??** 可知，在此時資料比起圖 **??** 更加貼近，加廣迴歸模型在訓練資料集的準確度與簡單線性迴歸相同，但其在測試資料集的準確度略高於肩擔線性迴歸模型。

- 兩組模擬資料變異數改變

  維持樣本數皆為 $n_1 = n_2 = 1000$，並更改共變異數矩陣中的參數，其參數如下:

$$\mu_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \mu_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix},$$

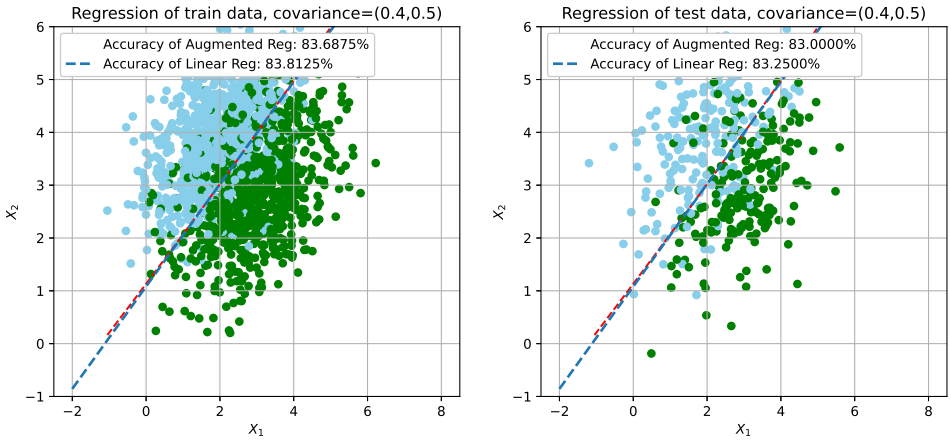

圖 8: Regression of Simulation

由圖 **??** 可知，兩筆資料的重合度偏高時，兩種方法的準確度都下降到 83% 左右，且基本擬合出的分界線一模一樣，只是簡單線性迴歸的準確度略高一點點，基本可以忽略不計。

接著再變更參數進行觀察一次，我們將母體參數更改如下:

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix},$$

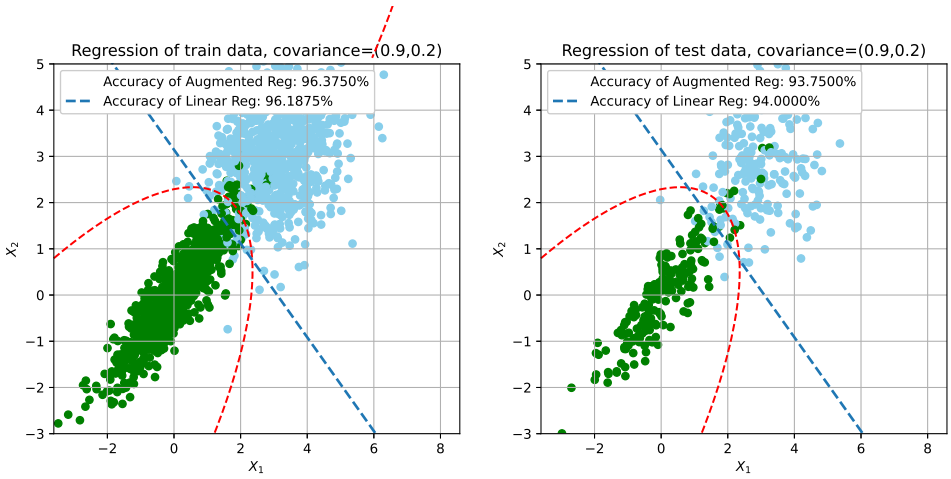

圖 9: Regression of Simulation

根據圖 **??** 可知，在兩筆資料重合部分，且其中一筆資料散布較廣、點也較密集時 (綠色點)，加廣迴歸模型在訓練資料集以及測試資料集的準確度都略高於簡單線性迴歸模型，且準確度有 95% 左右。

5 Logistic Regression Model

5.1 The Theory of Logistic Regression

在上文中探討了利用簡單線性迴歸與加廣迴歸模型來進行二元分類的各種例子，但其實在機器學習的分類問題中，較多的使用羅吉斯迴歸 (Logistic Regression) 來進行分類。假設存在兩個類別來解釋羅吉斯迴歸的簡單原理。在線性迴歸中，我們利用 $p(x) = \beta_0 + \beta_1 X$ 來表示機率，而在羅吉斯迴歸中，則使用羅吉斯函數:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{8}$$

並利用最大概似估計 (MLE) 方法來擬合模型。我們會將式 (??) 改寫成式 (??)，並將其稱之為勝算比 (odd)。

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X} \tag{9}$$

若將式 (??) 取 log，則將其稱為對數勝算比 (log-odd)，寫作式 (??)

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X \tag{10}$$

5.2 Example of Logistic Regression

接著直接模擬各種不同參數的三筆相依常態母體，並觀察羅吉斯迴歸的分類準確度。首先假定樣本數為 $n_1 = n_2 = n_3 = 1000$，且其參數為:

$$\mu_1 = \begin{bmatrix} 1 \\ 5 \end{bmatrix}, \mu_2 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \mu_3 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

根據圖 ?? 可知，此三筆資料羅吉斯迴歸 (Logistic Regression) 的分類準確度為 89.367%。以下我們亦呈現部分程式碼。

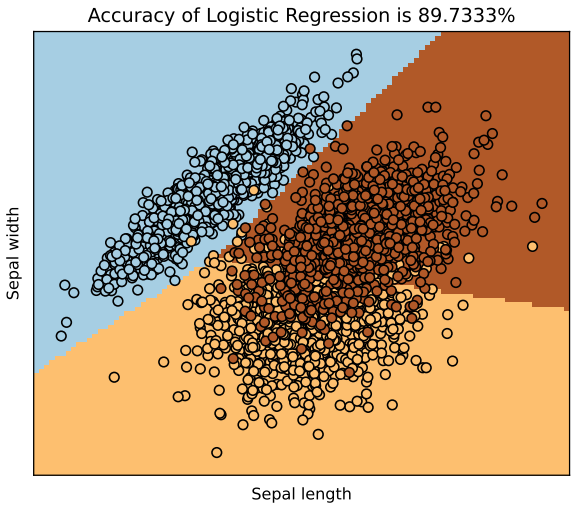

圖 10: Logistic Regression Simulation (1)

```
logreg = LogisticRegression(multi_class='multinomial',solver='
   lbfgs',class_weight='balanced',max_iter=1000)
logreg.fit(X, Y)
y_hat = logreg.predict(X)

score = logreg.score(X, Y)

#預測機率
y_pro = logreg.predict_proba(X)
#預測類別
y_predict = logreg.predict(X)

print("Accuracy in logistic regression: {:.4f}%".format(100 *
   np.mean(y_predict == Y)))

_, ax = plt.subplots(figsize=(6, 5))
DecisionBoundaryDisplay.from_estimator(
    logreg,
    X,
    cmap=plt.cm.Paired,
    ax=ax,
    response_method="predict",
    plot_method="pcolormesh",
    shading="auto",
    xlabel="Sepal length",
    ylabel="Sepal width",
    eps=0.5,
)
```

接著模擬各種不同參數的三筆相依常態母體，並觀察羅吉斯迴歸的分類準確度。假定樣本數為 $n_1 = n_2 = n_3 = 1000$，且其參數為:

$$\mu_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \mu_2 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mu_3 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

根據圖 **??** 可知，此三筆資料利用羅吉斯迴歸 (Logistic Regression) 分類的分類準確度為 94.63%。

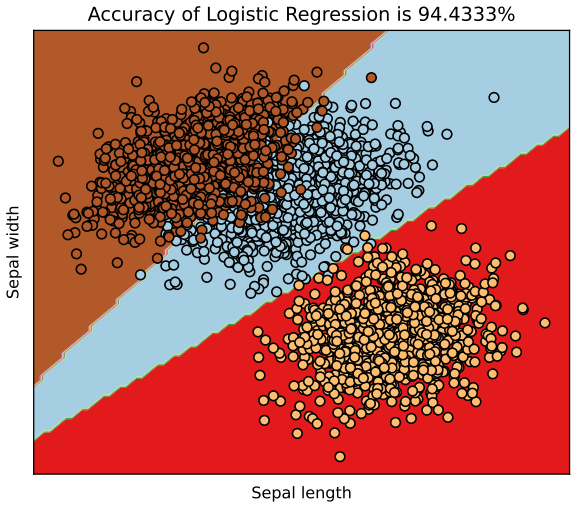

圖 11: Logistic Regression Simulation (2)

接著繼續模擬各種不同參數的三筆相依常態母體，並觀察羅吉斯迴歸的分類準確度。假定樣本數為 $n_1 = n_2 = n_3 = 1000$，且其參數為:

$$\mu_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \mu_2 = \begin{bmatrix} 7 \\ 7 \end{bmatrix}, \mu_3 = \begin{bmatrix} 9 \\ 1 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

根據圖 **??** 可知，此三筆資料利用羅吉斯迴歸分類的準確度為 92.4%。

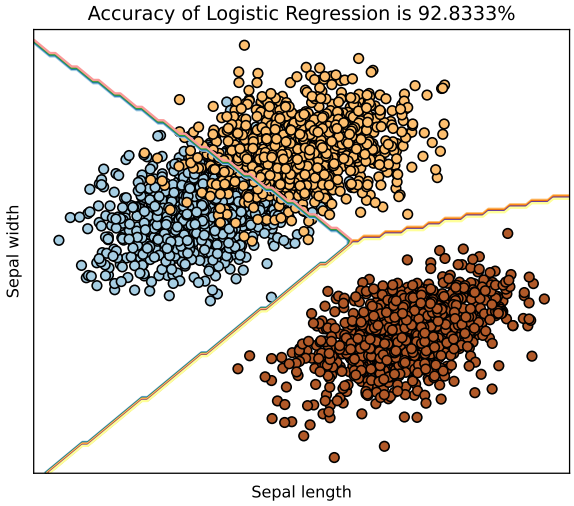

圖 12: Logistic Regression Simulation (3)

接著再模擬各種不同參數的三筆相依常態母體，並觀察羅吉斯迴歸的分類準確度。假定樣本數為 $n_1 = n_2 = n_3 = 1000$，且其參數為:

$$\mu_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \mu_2 = \begin{bmatrix} 9 \\ 9 \end{bmatrix}, \mu_3 = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

根據圖 **??** 可知，此三筆資料利用羅吉斯迴歸分類的準確度為 88.43%。

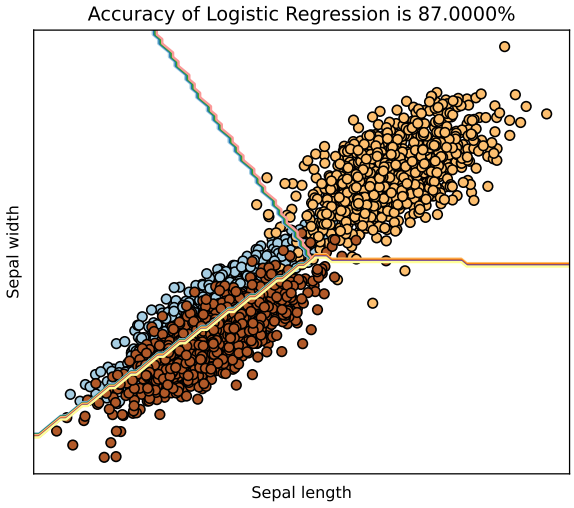

圖 13: Logistic Regression Simulation (4)

6 Discussion

在本文中，我們介紹了利用簡單線性迴歸以及加廣迴歸模型來執行二元分類可以發現，迴歸分析在處理分類問題時也可以得到很不錯的準確度，但它並不能應用在多元分類以上的問題。另外，與原本預想加廣迴歸模型的分類準確度都會比簡單線性迴歸高不同，在一般的問題上簡單線性迴歸的表現甚至比加廣迴歸模型略好一點。除此之外，羅吉斯迴歸在各種模擬資料的表現都很不錯，依據羅吉斯迴歸的理論，原本預想羅吉斯迴歸只能處理二元分類問題，但透過本文可以了解其如何利用 sklearn 套件進行多元分類以及繪圖。