使用各種分類演算法來預測病人是否有糖尿病

向子聰
國立中山大學
B113040058

洪理川
國立中山大學
B113040056

黃聖傑
國立中山大學
B113040059

陳育霖
國立中山大學
B113040052

曾俊諺
國立中山大學
B093022036

摘要

在現今越來越精緻的飲食，以及加工食品氾濫的影響下，我們身體所遇到的疾病越來越多元。糖尿病就是一個非常典型而且越來越多人會遇到的疾病。但是我們的醫護人力其實非常短缺，如果能夠透過演算法來診斷醫生只負責確認演算法所得出的結果必定能夠大幅增加效率而且能舒緩人力短缺的情況。

我們在本研究中，想要透過病人的各種參數然後用分類演算法來預測此病人是否罹患糖尿病。我們將用 K-鄰近演算法(KNN)、支援向量機(SVM)、決策樹(Decision Tree)、隨機森林(Random forest)、以及 Logistic regression 來進行預測並與實際資料比對來確認各個演算法的正確率。

1.簡介

機器學習會因為模型的目標不同而分出兩種問題，一個是回歸問題，一個是分類問題。這兩種問題下的模型，目的都是為了能夠根據標籤資料去做準確的預測。回歸問題的模型主要是為了去預測連續的數值，而分類問題主要是為了能夠預測或是將離散的質做分類。

因為今天我們主要是要預測病人是否罹患糖尿病，模型會將輸入資料分成 yes、no 兩類，為離散數值，也就代表我們將面對的是分類問題而非回歸問題。所以在本研究中我們利用病人的各項數據當作輸入，導入分類式演算法進行預測，並藉由實際值比對預測值得到正確率。

2.相關研究

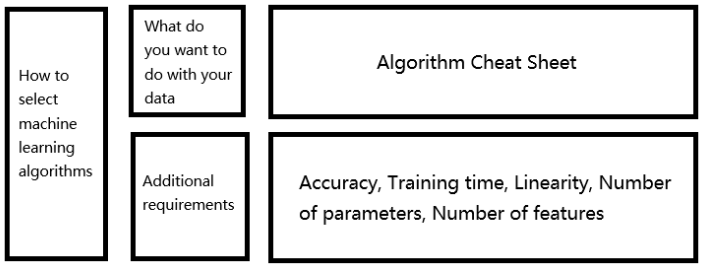
2.1 機器學習

隨著科技的發展，科學家們開始發現利用計算機加上機器學習演算法，就能從資料中整理與建立出不同的行為模式，希望以此來更理解人類的行為模式。機器學習並不是指計算機可以像人類一樣自動去學習，而是我們給予大量的資料，並讓它自動從中獲得某種規律的模式，並以此模式來達到預測的效果。機器學習可以分成「監督」與「非監督」式學習，這是指所收集的資料是否有被 labeled，也就是有否被定義。

監督式學習	預測	1. Linear Regression 2. Decision Tree 3. Random Forest 4. Neural Network 5. Gradient Booting Tree
	分類	1. Decision Tree 2. Naïve Bayes 3. Logistic Regression 4. Random Forest 5. SVM 6. Neural Network 7. Gradient Booting Tree
非監督式學習	分群	K-means
	關聯	Apriori
	降維	PCA

[1]以上是一些常用的監督式學習及非監督式學習的演算法。

2.2 如何決定使用哪個演算法



[2]要決定我們要用哪一個分類演算法，我們要先清楚知道我們的需求，像是我們需要的正確性，訓練時間長短，資料是否線性，特徵數量等等，並根據這些需求去選取合適的演算法。

演算法	準確度	定型時間	線性	參數
二元羅吉斯迴歸	良好	快速	Yes	4
二元決策樹系	非常好	中等	No	5
二元促進式決策樹	非常好	中等	No	6
二元神經網路	良好	中等	No	8
二元平均感知器	良好	中等	Yes	4
二元支援向量機器	良好	快速	Yes	5
多元羅吉斯迴歸	良好	快速	Yes	4
多元決策樹系	非常好	中等	No	5
多元促進式決策樹	非常好	中等	No	6
多元神經網路	良好	中等	No	8
「一對多」多元分類	-	-	-	-
線性迴歸	良好	快速	Yes	4
決策樹系迴歸	非常好	中等	No	5
促進式決策樹迴歸	非常好	中等	No	6
神經網路迴歸	良好	中等	No	8
K-Means 叢集	非常好	中等	Yes	8

[2]以上是一些演算法的特徵，我們可以根據需求去選擇最佳的演算法。

3.分類演算法

3.1 K-近鄰演算法(KNN)

KNN 是入門分類演算法的其中一個，是最簡單之一的演算法。就如他的名字一樣，他的概念其實就是把 K(一個自己設定的常數)個資料視為鄰居，並找出每個點之間的距離，再進行投票以決定它的類別。至於距離計算上，普遍會用到最常用的算法有三個[3]1.歐基里德距離 (Euclidean distance):

$$D=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2}$$

$$=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

2. 曼哈頓距離 (Manhattan distance):

$$D=|x_1-y_1|+|x_2-y_2|+\ldots+|x_n-y_n|$$

$$=\sum_{i=1}^{n}|x_i-y_i|$$

3. 明氏距離 (Minkowski distance):

$$D=(\sum_{i=1}^{n}|x_i-y_i|^p)^{\frac{1}{p}}$$

當計算完距離後，我們就可以選擇最接近的 K 個鄰居來進行投票，這個 K 的選擇通常會是奇數，以避免可能出現平手的情況。假設 K 設定為偶數，我們也可以利用權重投票法來決定最終的結果。而 K 值過大，就有可能會包含一些非相關的樣本，而太小又會受噪音影響，因此選擇合適的 K 值非常重要。KNN 作為機器學習的入門演算法，是一個簡單易懂的演算法，但同時也有缺點，像是計算量比起其他演算法較多，因為當中的每個點都需要計算。以下我會介紹我們是如何使用 C++程式語言來實現 KNN 演算法的。首先我們將是次實作中的 K 值設定為 11，奇數可以避免平手的情況。

```
float Pregnancies = 1, Glucose = 5, BloodPressure = 1, SkinThickness = 1, Insulin = 1;
float BMI = 2, DiabetesPedigreeFunction = 10, Age = 1;
```

另外我們也有將每一項資料分配權重，因為每一項資料特征對於病人是否會患有糖尿病的影響是不一樣的，像是有沒有懷孕可能沒有太大影響，但糖尿病函數跟 BMI 等就有比較直接的影響了。因此我們利用權重的分配可以使預測的準確度有所提升。我們的程式中會首先將 training 的 data 存入 vector 中，再將 test 的 data 一行一行的讀進來，並利用 test 的資料與 training 的資料進行距離的計算。我們使用的計算方式為歐基里德距離:我們這邊還會乘上設定的權重以獲得更準確的結果。

```
        continue;
    else {
        d[i].value += pow(input[j] - data[i][j], 2) * Proportion[j];
        Proportion[j];
        total += Proportion[j];
    }
}
d[i].value = sqrt(d[i].value/total);
```

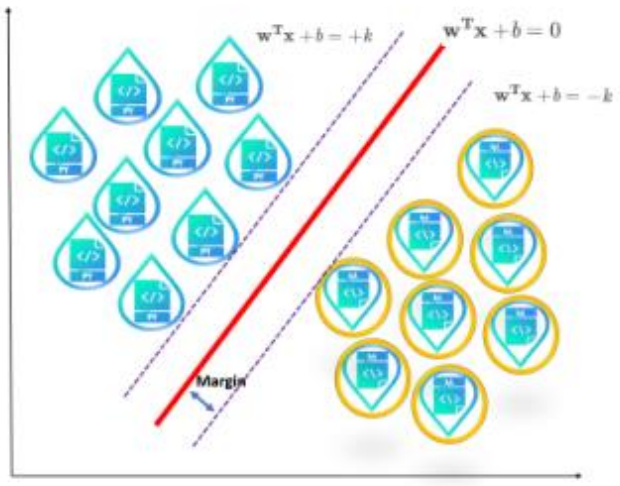
```
sort(d, d+n, cmp);
for(int i = 0; i < N; i++) {
    Outcome += d[i].Outcome;
}
if(Outcome > N/2) {
    Correct += (input[8] == 1);
    return 1;
}
else {
    Correct += (input[8] == 0);
    return 0;
}
```

之後我們會利用 sort() function 來排序以方便我們取得距離最近的 11 個點。之後我們會將這 11 個點的 Outcome 分為大於一半或小

於等於一半兩個情況來決定類別，大於一半我們預測為有糖尿病(1)，反之為沒有糖尿病(0)，並以此來判斷我們的預測是對還是錯的，假如是對的那就將 Correct 變數+1，反之不會+1。最後我們就可以將 Correct*100/樣本總數來取得正確率。

3.2 支援向量機(SVM)

SVM 的概念早在 1960-1990 年代就已經由數學家 Vapnic 及 Chervonenkis 等人提出，此機器學習模型演算法是針對非線性樣本的模型。一般來說比較簡單的樣本都是線性可分的樣本，很多簡單的模型都可以取得很不錯的分類結果。可是當我們遇到一些比較複雜的樣本，就有可能會遇到非線性的樣本，那普通的分類演算法就未必能運作了。SVM 透過將低維度線性不可分的樣本映射到高維度空間，並透過計算找出一個超平面將這些樣本進行切割，以此達到分類的效果。超平面的這條線必須是兩邊的樣本都離這條線最遠的，也就是 margin 是最大的，這樣我們添加新的樣本時成功的機率就會更高。[4]我們假



設分隔線為 wTx+b=0，那兩條虛線就可以設為 wTx+b=+k and wTx+b=-k。那兩條虛線之間的距離就會是

$$\frac{\mathbf{w}}{||\mathbf{w}||}\cdot(\mathbf{x}_+-\mathbf{x}_-)=\frac{\mathbf{w}^T(\mathbf{x}_+-\mathbf{x}_-)}{||\mathbf{w}||}=\frac{\mathbf{k}}{||\mathbf{w}||}$$

只要我們把上面的公式最大化就可以將 margin 最大化了。以下是幾個常用的核函數(Kernel Function):[4]

多項式核函數:

$$K(\mathbf{x_i},\mathbf{x_j})=(\mathbf{x_i}\cdot\mathbf{x_j})^{\mathbf{d}}$$

$$K(\mathbf{x_i},\mathbf{x_j})=(1+\mathbf{x_i}\cdot\mathbf{x_j})^{\mathbf{d}}$$

高斯核函數:

$$K(\mathbf{x_i},\mathbf{x_j})=\exp(-\frac{||\mathbf{x_i}-\mathbf{x_j}||^2}{2\sigma^2})$$

Sigmoid核函數:

$$K(\mathbf{x_i},\mathbf{x_j})=\tanh(\kappa(\mathbf{x_i}\cdot\mathbf{x_j})+\theta)$$

核函數就是我們在算高維度的樣本解的時候會用到的函數。以下我會介紹我們是如何利用 python 以及 sklearn 的套件來實現 SVM。首先我們會先初始化 learning rate，iterate count，lambda，這些 Parameters 都是用作實行 gradient descent algorithm 的。接下來我們有 fit() function，這是用作 training 的，我們會把 weight 跟 bias 初始化，然後按照設定的 iterate 次數進行 iteration，每次都會更新 weight 和 bias 以訓練出最合適的數值，令最後的準確率更高。

```
# Gradient Descent iteration
for _ in range(self.iterate_count):
  outcome = np.array([1 if y > 0 else -1 for y in self.Y]) # make it for binary classification

  for i, row_data in enumerate(self.X):
    dweight = 0
    dbias = 0
    # check if data is outside the margin
    if (outcome[i] * (np.dot(row_data, weight) - bias) >= 1):
      dweight = 2 * self.lambdaa * weight
    else:
      # if it's inside the margin, increase the margin
      dweight = 2 * self.lambdaa * weight - np.dot(row_data, outcome[i])
      dbias = outcome[i]

    weight -= self.lrate * dweight
    bias -= self.lrate * dbias

# Lastly change the weight and the bias
self.Weight = weight
self.Bias = bias
```

然後我們有 predict() function，我們計算結果的公式為 y = wx - b。

```
output = np.dot(X, self.weight) - self.bias
predicted_labels = np.sign(output)
result = np.where(predicted_labels <= -1, 0, 1)
```

而當我們使用已經 Optimized 的 weight 跟 bias 進行計算後，結果有可能是-3,-5,6,8 等等不一定的正負數，而我們在進行 gradient descent 時因為方便原因我們將 0 換成-1 了。因此我們要將這些負數轉換成-1，正數轉換成 1，並再將-1 轉換成 0，以 0 表示沒有糖尿病，1 表示有糖尿病。最後我們就可以開始進行訓練跟測試了。我們讀取資料並將最後一列分開成結果，並使用 sklearn 的 standardScaler 進行 standardization，然後建立 SVM 並進行訓練與預測。最後使用 sklearn 套件中的 accuracy_score 去檢查 training data 與 test data 的準確率。

3.3 決策樹(Decision Tree)

決策樹主要是根據訓練資料產生一棵樹，依照訓練規則對新樣本做出預測。決策樹本身是透過 divide and conquer 的策略，透過 greedy search 來找最佳分割點。分割的過程是從上到下並且以遞迴的方式進行，一直做到所有紀錄被分到特定的類別標籤下。我們通常透過 Information gain 以及 Gini impurity 這兩個指標來評估分支的好壞。Information Gain 主要是透過訓練來找出規則，讓 Information Gain 能夠最大化。通常是透過計算 entropy。entropy 的計算主要是透過以下公式

$$Entropy = -\sum_j p_j \log_2 p_j$$

然後再透過是的 entropy(-p*logp)跟否的 entropy(-q*logq)去結合成 Information Gain

$$\text{Information Gain} = -p * \log_2 p - q * \log_2 q$$

另外一種評估方式是透過 Gini impurity。當 Gini impurity 數字越大時，代表資料越混亂。公式如下圖所示，p 代表是的機率，q 代表否的機率。

$$\text{Gini Impurity} = 1 - (p^2+q^2)$$

舉例來說如果資料一致的時候混亂的程度為 1 - (1^2+0^2) = 0，當資料各有一半不同時混亂程度為 1 - (0.5^2+0.5^2) = 0.5。接著會講解如何透過 sklearn 這個套件來實現 Decision Tree。首先透過 DecisionTreeClassifier()這個函式來形成 model，接著將 training data 透過 fit()這個函式去訓練。接著透過 predict()這個 function 把要測試的資料丟進 model 裡並進行預測，將預測的結果跟實際結果進行比對去算出並印出此 model 的正確率。實際 code 如下圖，其中 total 為正確率，times 為計算次數，clf 為 model，y_pred 為預測結果，Outcome_test 為測資實際結果

```python
total = 0 # 總正確率
times = 10 # 計算次數

for i in range(times):
    # 決策樹
    clf = DecisionTreeClassifier()
    clf.fit(X_train, Outcome_train)

    # 預測
    y_pred = clf.predict(X_test)

    total += np.sum(Outcome_test == y_pred) / len(y_pred)

print(f"Average Accuracy in {times} times: {total * 100 / times:.2f}%")
```

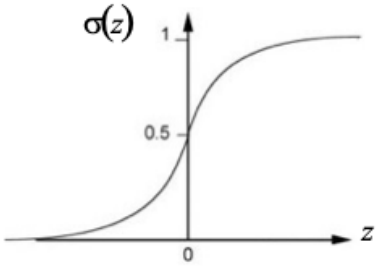
3.4 隨機森林(Random Forest)

Random forest 主要是透過建立多棵相互獨立的 CART(classification and regression tree)，然後將所有樹的分類結果以平均或多數決為依據，作為模型的預測結果。為了增進計算結果還會搭配隨機的訓練資料。接著會講解如何透過 sklearn 這個套件來實現 Random forest。首先會先用出隨機的訓練資料，並透過 DecisionTreeClassifier()這個函式來形成 model，接著將 training data 透過 fit()這個函式去訓練。接著透過 predict()這個 function 把要測試的資料丟進 model 裡並進行預測，最後把預測結果都收集起來。接著把蒐集完的結果取平均當作 random forest 的預測跟實際結果進行比對去算出並印出此 model 的正確率。

3.5 Logistic Regression

Logistic regression 是一個平滑曲線，hhinge 主要透過找機率來分類。當機率大於 0.5 時輸出是，反之則輸出否。當資料是高斯機率分布時可以說機率是σ(z)。其中 z 是透過所有特徵*權重+偏差，銓重與偏差為訓練所得的一個參數。得到 z 之後再帶入 sigmoid function 就可以得到機率。以下附上 sigmoid function 以及圖

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



接著我會介紹我們是如何實現 logistic regression。首先我們會先建立 model(使用 Logistic_Regression function)並附帶兩個參數 lrate 以及 iterate_count。接著我們訓練 model(用 fit function)時會先將原始的特徵、權重、及偏差帶入 sigmoid function 計算出機率，接著會用 cost function 算出 error，再用 gradient descent 的方式去重新計算出新的權重跟偏差以優化 model。iterate_count 主要用在我們要訓練這個 model 幾次，lrate 則是用在更新權重以及偏差。最後再透過 accuracy 這個 function 去比對 predict_value 以及 actual_value 去算出正確率。

4 系統實作與模擬

4.1 系統環境

本次實作的程式，主要是以 python 以及 C++ 所撰寫。實驗環境的電腦作業系統為 Windows 11，CPU 是 Intel(R) Core(TM) i7-10700K @ 3.80GHz，RAM 為 DDR4 32GB。實驗使用的範例資料是老師提供的測資 A 跟 B。

	KNN	Logistic Regression (No Data Standardization)	Logistic Regression (With Data Standardization)	SVM (No Data Standardization)	SVM (With Data Standardization)	Decision Tree	Random Forest
Average Accuracy Data A(10x)	75.622	70.647	80.597	70.647	78.109	73.114	76.122
Average Accuracy Data B(10x)	79.000	77.000	79.000	38.000	78.000	71.000	77.000

4.2 實驗結果

以上是我們這一次 5 種演算法實作所得出的結果。我們發現有沒有進行 data standardization 對準確率有很大的影響.在各種演算法中，我們發現有標準化的 logistic regression 是平均來說最好的。最終的排名為:Logistic regression(有標準化) > SVM(有標準化) > KNN > Random Forest > Logistic regression(無標準化) > Decision Tree > SVM（無標準化）

5 結論

我們這次的實驗主要目的是預測病人是否罹患糖尿病。透過病人之各項測量值當作特徵丟入分類的演算法去判預測。實驗中也利用五種演算法去預測。根據實驗結果可以知道我們已經可以透過分類演算法去預測病人是否罹患糖尿病正確率接近 80%。如果能夠繼續提升演算法的正確率再加入醫生的主觀判斷不只能夠提升醫護人員診斷的效率也能確保穩定性以及正確性。

參考文獻

[1]. 機器學習演算法－監督與非監督式學習 https://medium.com/marketingdatascience/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E6%BC%94%E7%AE%97%E6%B3%95-%E7%9B%A3%E7%9D%A3%E8%88%87%E9%9D%9E%E7%9B%A3%E7%9D%A3%E5%BC%8F%E5%AD%B8%E7%BF%92-e9dbeee94a30

[2]. Microsoft learn https://learn.microsoft.com/zh-tw/azure/machine-learning/how-to-select-algorithms?view=azureml-api-1

[3]. Pyinvest [機器學習首部曲]K-近鄰演算法 KNN https://pyecontech.com/2020/04/19/knn/

[4]. Pyinvest [機器學習首部曲] 支援向量機 SVM https://pyecontech.com/2020/03/24/svm/