# 2025年臺灣國際科學展覽會
# 優勝作品專輯

作品編號　190005

參展科別　電腦科學與資訊工程

作品名稱　基於心電圖的智慧睡眠分析

得獎獎項　四等獎


就讀學校　臺北市私立復興實驗高級中學

指導教師　馬瑪宣

呂毓蘋

作者姓名　黃楚涵

柯宇庭



關鍵詞　心電圖（ECG）、睡眠品質（Sleep Quality）、深度學習與機器學習（Deep Learning and Machine Learning）

# 作者簡介

我們是就讀於復興實中的柯宇庭和黃楚涵，因對電腦科學感興趣，且發現睡眠問題在生活周遭愈發普及，故投入「基於心電圖的智慧睡眠分析」之研究，期望建立精準評估方法。我們非常感謝馬瑪宣老師的指導與鼓勵及臺灣科技大學林淵翔教授與學長姐的不吝指教，使我們在相關方面獲益良多，未來我們將在更多相關領域繼續鑽研、學習！

# 2025 年台灣國際科學展覽會

# 研究報告

區別：北區

科別：電腦科學與資訊工程科

作品名稱：基於心電圖的智慧睡眠分析

關鍵詞：心電圖（ECG）、睡眠品質（Sleep Quality）、深度學習與機器學習（Deep Learning and Machine Learning）

編號：

# 摘要

睡眠相關問題常見於現代緊張的社會，傳統睡眠分析方法需要腦電圖（EEG）、肌電圖（EMG）、眼電圖（EOG）等信號，量測複雜度高。本研究透過 Python 程式語言以深度學習和階層式投票機器學習方法，開發一套自動分析程式，僅透過心電圖（ECG）信號分析睡眠階段。並結合睡眠評估標準，製訂一可量化的睡眠品質評估表，提供臨床醫師判讀睡眠品質的指標。本研究的優點是僅透過一種信號便能準確、客觀、快速分析，且操作介面簡易。研究結果顯示，本研究清醒和睡眠狀態之辨識準確率高達約 90%，與其他類似睡眠品質評估研究的論文比較，準確率高出 10～17%，整體睡眠階段分析準確度高達 87%。本研究方法未來可應用於臨床醫療，協助醫師做精準的患者睡眠品質診斷。



# Abstract

Sleep-related issues are common in today's fast-paced society. Traditional sleep analysis methods require complex measurements using electroencephalograms (EEG), electromyograms (EMG), and electrooculograms (EOG). This study develops an automated analysis system using Python programming language combined with deep learning and ensemble voting of machine learning, which analyzes sleep stages solely through electrocardiogram (ECG) signals. By integrating sleep assessment standards, a quantifiable sleep quality evaluation form is established to provide indicators for clinicians to interpret sleep quality. The strength of this research lies in its ability to accurately, objectively, and quickly analyze sleep quality using just one signal, and it features an easy-to-use interface. The results show that the identification accuracy for wake and sleep states is approximately 90%, which is 10-17% higher than similar sleep quality assessment studies. Overall, the accuracy of sleep stage analysis reaches 87%. The methods developed in this study could be applied in clinical medicine to assist physicians in making precise sleep-quality diagnoses for patients.

# 壹、前言

一、研究動機

睡眠是一種重要的生理現象和必要的生理過程，人類可以透過睡眠消除疲勞、恢復精神和體力、保持良好的狀態。然而，現代社會隨著科技的發展，生活步調愈發緊湊，使得許多人的壓力也越來越大，導致睡眠相關的問題愈來愈常見。甚至，根據統計 [1]，臺灣人睡眠品質不佳的比例偏高，使近年睡眠品質的議題越發廣受大家討論和重視。而現今之睡眠品質量測方法多採用睡眠多項生理檢查（Polysomnography, PSG）[2]，然而，其中資料量過於繁雜，加劇偵測的困難度與複雜度，因此，我們希望可以只藉由心電圖來分析睡眠品質。而在參考論文後，我們決定採用深度學習方法中的卷積神經網路模型（Convolutional Neural Network Model, CNN Model）和機器學習方法中之隨機森林（Random Forest, RF）、近鄰演算法（K Nearest Neighbors, KNN）、決策樹（Decision Tree, DT）、集成投票分類（Ensemble Voting Classifier, VC）之 Hard Voting 來進行分析，使人們在解決睡眠問題時可以有更為快速準確的判斷方式，以提供妥善治療，造福大眾。

二、研究目的

（一）以深度學習配合 ECG 波形分類**清醒**與**睡眠**狀態。

（二）以**監督式機器學習**配合 ECG 波形分類不同**睡眠階段。**

（三）藉由不同的睡眠階段的輪替以分析**睡眠品質**，並產生較為直觀的**睡眠評分**與建議。

（四）計算預測值與實際值之間誤差，並修改模型，使其準確度達到 80%以上。

（五）將判斷結果、心電圖、真實結果、準確率與睡眠評分顯示於**使用者介面**上，並提供睡眠相關資訊，整合成一**睡眠分析系統。**

三、文獻回顧

（一）心電圖（ECG）：

人在心搏週期中由節律點發出的脈衝沿傳導系統傳至整個心臟時所產生的電流可經由體液傳導，並在體表檢測後紀錄之電位變化圖形。

- **R 波：**

  心電圖上**最明顯之突起**波形，其中最高峰稱之為 **R 點（R peak）。**

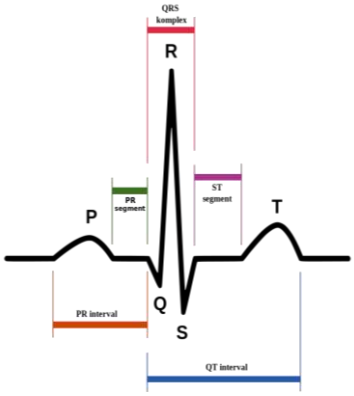



圖 1：P、Q、R、S、T 波圖 [3]

（二）MIT-BIH 多通道睡眠資料庫 [4]：

由波士頓貝斯醫院以色列醫院睡眠實驗室對受試者進行監測，提供受試者睡眠時的多種生理訊號（心電圖、腦電圖、肌電圖、每分鐘呼吸頻率 RESP 等）。在該資料庫中，所有 16 名受試者均為男性，年齡介於 32～56 歲之間（平均年齡 43 歲），體重範圍在 89～152 公斤之間（平均體重 119 公斤）。

表 1：MIT-BIH 多通道睡眠資料庫各睡眠階段資料總量

| | | | |
|---|---|---|---|
| **W（清醒）** | 3108 | **N2（淺睡期）** | 3883 |
| **R（快速動眼期）** | 700 | **N3（熟睡期）** | 483 |
| **N1（入睡期）** | 1814 | **N4（深睡期）** | 181 |
| **M（移動）** | 16 | | |

單位：段（一段為 7500 個資料點）

註：M（Move）指偵測時因受測者進行移動而產生雜訊之片段


圖 2：MIT-BIH 多通道睡眠資料庫 ECG 資料範例圖（圖片由作者擷取自資料庫）

（三）**Rechtschaffen & Kales（R&K）睡眠分類標準（分為 6 項）：**

1. 清醒（Wake, **W**）
2. 快速動眼期（Rapid eye movement, REM, **R**）
3. 非快速動眼期（non-rapid eye movement, NREM）：

入睡期（**N1**）淺睡期（**N2**）熟睡期（**N3**）深睡期（**N4**）

- 一個完整的睡眠週期大約為 90 ~ 100 分鐘，平均一晚進行 4 ~ 5 個。
- 正常睡眠週期，由 N1 循序進入 N2、由淺入深至 N3、最後進入 N4，接著逆向回到淺睡眠，之後才進入 R。快速動眼期結束後，接著再進入 N1，如此週而復始。
- 在較新的 American Academy of Sleep Medicine（**AASM）睡眠分類標準**中，N3 與 N4 **合稱為 N3**。
- 隨著週期的增加，快速動眼期所佔的時間變得越來越長，非快速動眼期則越來越短。

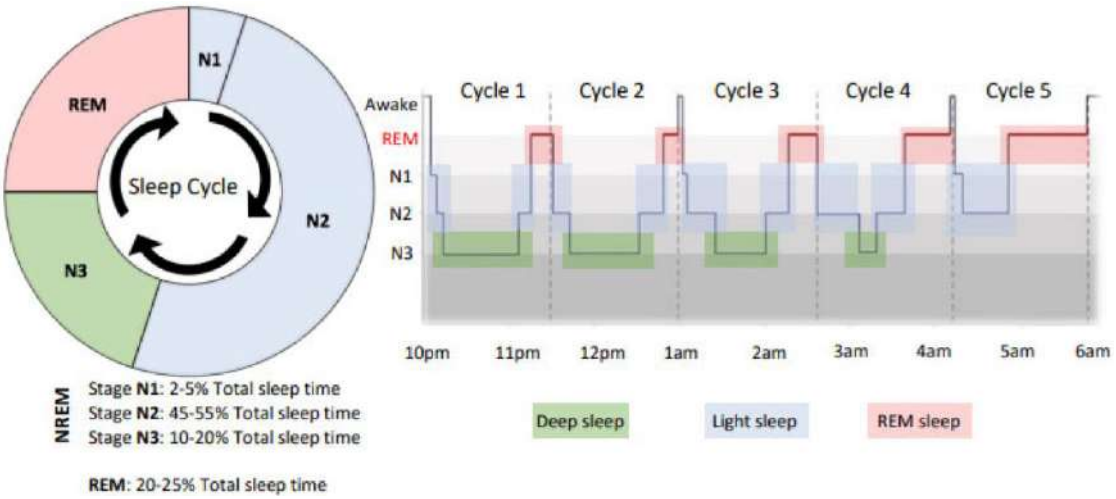



圖 3： 睡眠階段與睡眠週期 [5]

表 2：良好睡眠中各睡眠階段佔總睡眠時長之百分比

| | |
|---|---|
| **N1** | 5% |
| **N2** | 50% |
| **N3 + N4** | 20% |
| **REM** | 25% |

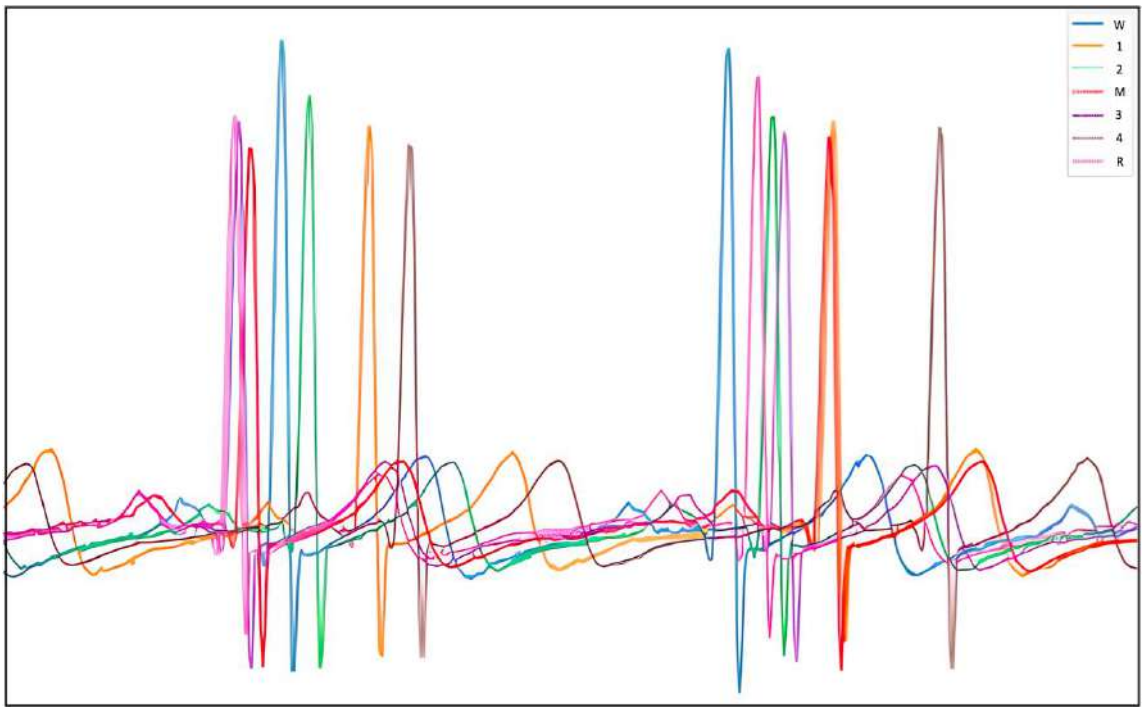



圖 4：各睡眠階段波形疊圖（圖片由作者自行繪製）

（四）**機器學習**（Machine Learning, **ML**）：

1. 人工智慧（Artificial Intelligence, AI）的一個分支。經由人工進行特徵提取，訓練機器從中學習判斷資料類型，進而預測結果，為一種可透過經驗不斷優化、改進之技術。
2. 本研究使用之機器學習方法：
   (1) 隨機森林（Random Forest，RF）
   (2) 決策樹（Decision Tree，DT）
   (3) K-近鄰演算法（K Nearest Neighbor, KNN）
   (4) 集成投票分類（Ensemble Voting Classifier）

（五）**深度學習**（Deep Learning, **DL**）：

1. 簡介：

   為機器學習的一個分支，透過多個處理層間的轉換，自行抽取足以代表資料特性的特徵以進行學習，是一種特徵學習，可以節省機器學習的特徵提取工程所花費的時間。

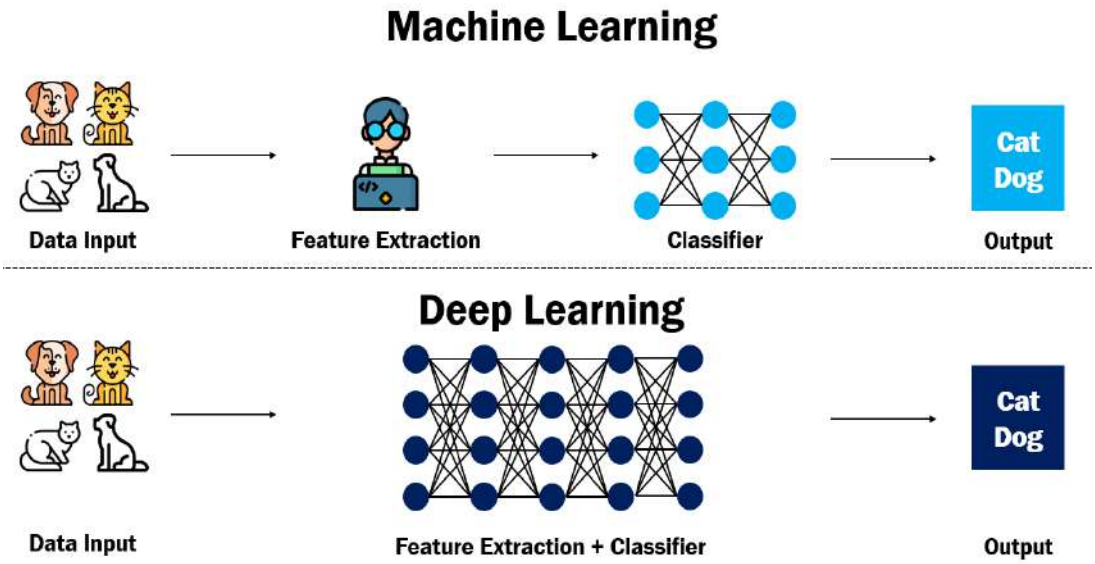



圖 5：機器學習與深度學習比較圖 [6]

2. 本研究使用之深度學習模型：

   **卷積神經網路模型（Convolutional Neural Network, CNN）：**

   參考人的大腦視覺組織，以多重神經元來建立的深度學習模型，在影像辨識中可以超越人類辨識的精準度。

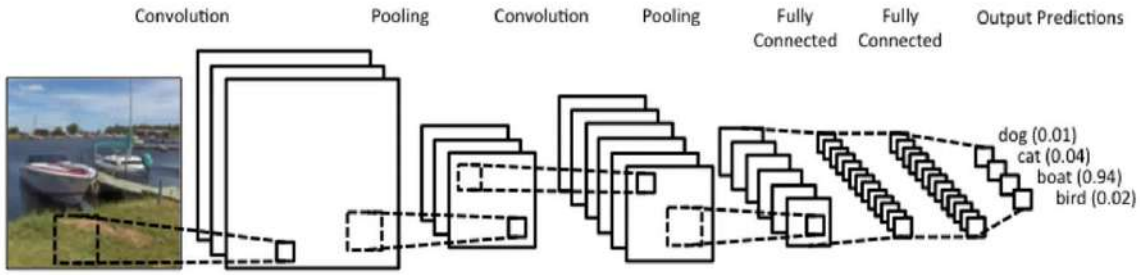



圖 6：CNN 概念圖 [7]

（六）心率分析參考數值：

1. 心率變異度（Heart Rate Variability, HRV）
2. 平均脈間期（Mean NN Intervals, NNI_Mean）
3. 脈間期標準差（Standard Deviation of the NN Intervals, SDNN）
4. 均方根連續差（Root Mean Square of the Successive Differences, RMSSD）
5. 交感神經指標（sympatho - vagal balance Ratio of Fast Fourier Transform, FFT Ratio）
6. 功率頻譜密度（Power Spectral Density, PSD）
7. 心源性呼吸訊號（ECG - Derived Respiration, EDR）

（七）混淆矩陣（Confusion Matrix）：

混淆矩陣是以預測和實際之間的比較進行的表格整理，將結果透過實際與預測值的異同分成四個象限，從中可以衍伸出準確率（**Accuracy**）、精準率（**Precision**）、召回率（**Recall**）等基本常用的評估指中標，如下圖所示：

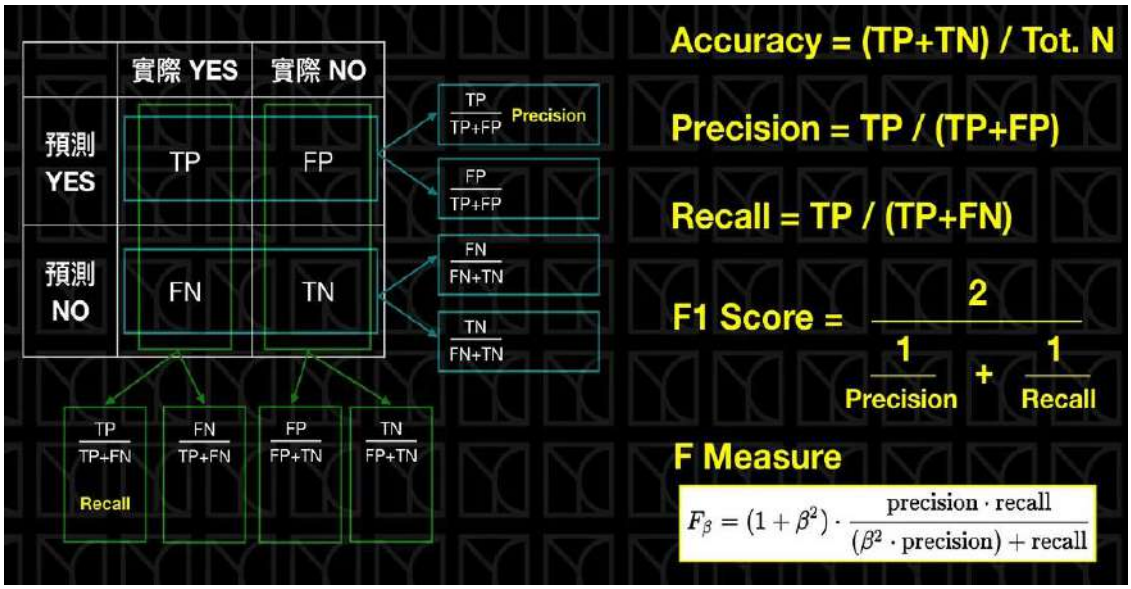



圖 7：模型評估指標說明 [8]

# 貳、研究設備、系統與器材

- 設備：
  - MacBook Air 2012：
    - 處理器：1.8 GHz 雙核心 Intel Core i5
    - 記憶體：4 GB
  - Acer Swift SF514-52T：
    - 處理器：Intel (R) Core (TM) i5-8250U CPU @ 1.60GHz  1.80 GHz
    - 記憶體：8 GB
- 系統：
  - MIT-BIH 多通道睡眠資料庫
  - Python 3.9.17
  - Kivy 2.3.0

# 參、研究過程與方法

一、研究方法及步驟

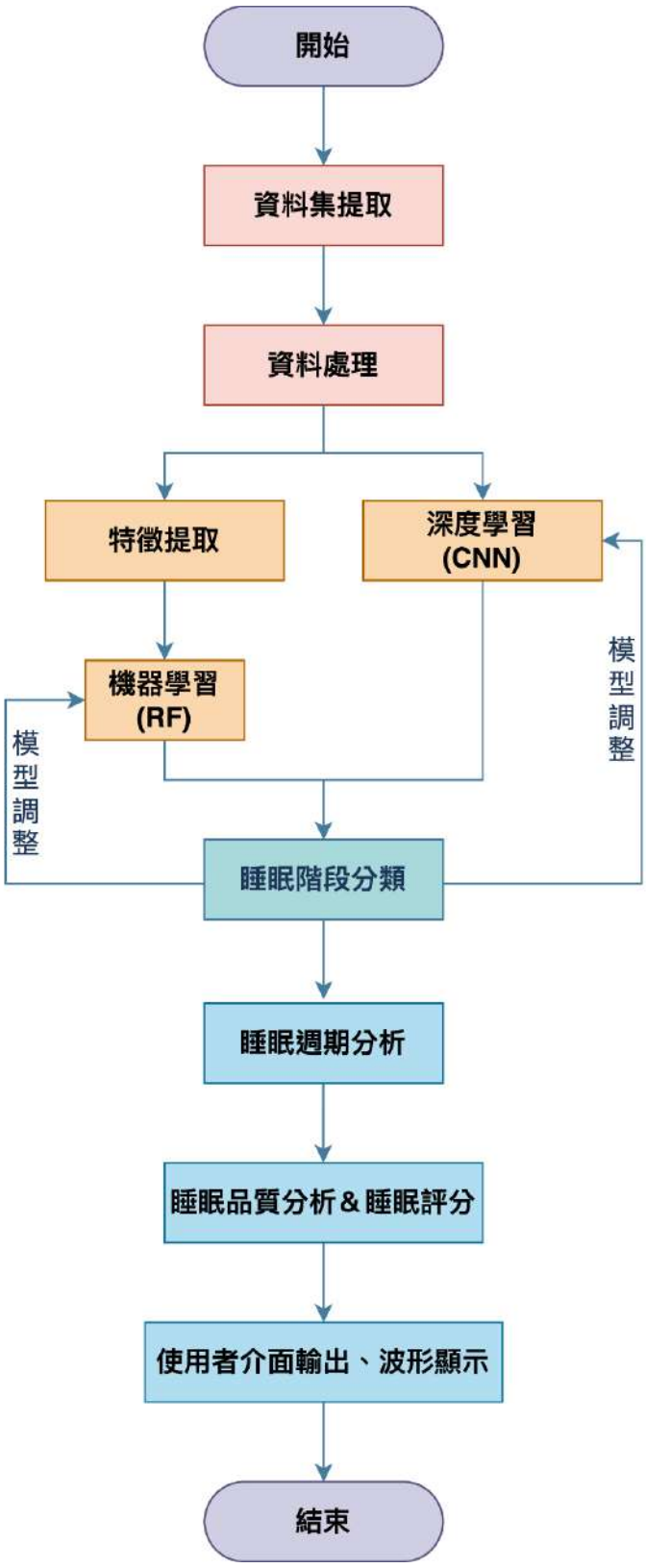



圖 8：整體實驗流程示意圖（圖片由作者自行繪製）

本實驗先提取資料集中之 ECG 及睡眠標註資料，接著分為深度學習與機器學習部分：深度學習部分以切割和濾波後之原始訊號進行學習，以判斷清醒或睡著；機器學習部分以切割和濾波後所儲存之資料進行學習，以判斷睡眠階段為 N1、N2 或 N3 與 N4，並不斷調整模型與參數，最後再將資料整合至一睡眠系統。

（一）資料前處理：

1. 資料集提取：

   我們先將資料集檔案匯入，接著透過其本來就已提供之**睡眠標註（W, R, 1, 2, 3, 4）**，以每 30 秒一個標註（Annotations）切分睡眠階段資料，再取 ECG 訊號，並將其轉為**一維**。

2. 資料切分：

   先設定一段 ECG 訊號為 7500 個資料點（因一段資料為 30 秒，1 秒為 250 個資料點），以進行訊號切割。接著透過 if - else 運算式區分**清醒（W）**與**睡著**資料，並剔除**動作訊號（Move, M）**，避免雜訊影響實驗結果，再將處理好的資料存成檔案，方便模型直接讀取。

```
for j in range((result.size // s)-1):
    try:
        signal_cut = result[j*s:(j+1)*s]
        gt = annotations.aux_note[j][0]

        if gt == 'W':
            xData.append(signal_cut.tolist()
            yData.append(0)
        elif gt == "M":
            pass
        else:
            xData.append(signal_cut.tolist()
            yData.append(1)
    except:
        pass
```

圖 9：資料切分程式碼

（擷取自作者自行運算程式時的畫面）

表 3：訓練與測試資料之清醒與睡著比例

| 資料類別 | 訓練資料比例（段） | 測試資料比例（段） |
|---|---|---|
| **清醒** | 2486 | 622 |
| **睡著** | 2486 | 4575 |

單位：段

3. 特徵提取：

   接著透過函式庫提取訊號中 5 個特徵 NNI_Mean、SDNN、RMSSD、FFT Ratio 及 EDR 之**特徵重要度**，並將其存成 CSV 檔，共 9992 筆資料。

   **註：特徵重要度為衡量每個特徵對目標的影響程度之指標。**

```python
for j in range((result.size // s)-1):
    try:
        signal_cut = result[j*s:(j+1)*s]
        gt = annotations.aux_note[j][0]

        if gt != "w" and gt != "M":
            # 切割訊號（這一段訊號長度~下一段）
            signal_hrv = hrv(signal = signal_cut, sampling_rate=250)
            # [:3]：取維度為3
            t, filtered_signal, rpeaks = biosppy.signals.ecg.ecg(signal = signal_cut, sampling_rate=250)[:3]
            nni = tools.nn_intervals(t[rpeaks])
            signal_psd = fd.welch_psd(nni, show = False) #, show_param = False, show = False
            signal_edr = len(biosppy.signals.resp.resp(signal_cut, sampling_rate = 250) ["resp_rate"])/30
```

圖 10：取特徵重要度程式碼（擷取自作者自行運算程式時的畫面）

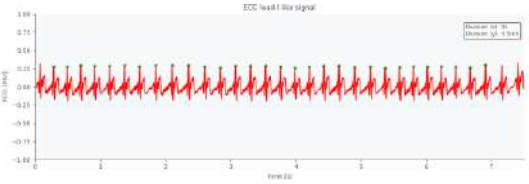


圖 11：通過 HRV 在心電圖中偵測所得的 R peak（圖片由作者自行繪製）

圖 12：CSV 檔部分示意圖（擷取自作者自行運算程式時的畫面）

（二）深度學習模型的建立：

1. 建立 CNN 模型：

設定各層參數（Filters、Activation、Batch Size、Validation Split、Epochs）。

表 4：初始設定之參數

| Epochs | Activation | Batch Size | Validation Split |
|---|---|---|---|
| 30 | relu (all) | 32 | 0.3 |

```python
def build_CNN_model(n_timesteps,n_features):
    model=Sequential()
    model.add(Conv1D(filters=64,kernel_size=5,padding="valid",activation="relu"))
    model.add(Conv1D(filters=32,kernel_size=5,activation="relu"))
    model.add(MaxPooling1D(pool_size=3))
    model.add(Conv1D(filters=32,kernel_size=5,activation="relu"))
    model.add(Conv1D(filters=32, kernel_size = 5, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Conv1D(filters=16, kernel_size = 5, activation='relu'))
    model.add(Conv1D(filters=16, kernel_size = 5, activation='relu'))
    model.add(Conv1D(filters=8,kernel_size=5,activation="relu"))
    model.add(GlobalAveragePooling1D())
    model.add(Dense(40, activation="relu"))
    model.add(Dense(1,activation="linear"))
    model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["mae","accuracy"])
    return model

cnn_model=build_CNN_model(x_train.shape[1],x_train.shape[2])
history=cnn_model.fit(x_train, y_train, epochs=20,validation_split=0.3,batch_size=32,verbose=1)
```

圖 13：CNN 模型建立程式碼（擷取自作者自行運算程式時的畫面）

2. 繪製混淆矩陣：

繪製每次模型之混淆矩陣，以比較每次模型之資料判讀狀況。

（三）深度學習模型的修改與優化：

表 5：本實驗使用之深度學習參數意義

| 參數名稱 | 功能、意義 |
|---|---|
| filters | 決定每階段進入深度學習的資料分層數 |
| kernel_size | 決定每階段進入深度學習的資料的範圍、大小 |
| activation | 決定不同階段之間連結的函數關係 |
| batch_size | 決定一次放入深度學習的資料筆數 |
| validation_split | 決定深度學習訓練與測試資料的比例 |
| epochs | 決定深度學習訓練的回合數 |
| optimizer | 透過衡量預測值和目標值的差異來調整模型的優化器 |
| pooling | 調整池化，即減少資料量並保留、結合重要資訊 |
| dropout | 透過刪除部分（卷積神經網路之）神經元來減緩過度擬合 |

註：紅底者為本研究作為操縱變因調整之項目。

1. 操縱變因：活化函數（Activation）：

(1) 將最後全連接層之活化函數由線性（Linear）改為 S 型（Sigmoid），因相較於線性函數，此函數能將任何實數值映射到 0 到 1 之間的範圍，更加適合**二元分類問題**。

(2) 將其他活化函數由原先的 Relu 全部替換成 Tanh、Selu、Elu、Relu6、Gelu 與 Softsign，並找出最好的組合。（見表 12 實驗第 1~7 次）

```
def build_CNN_model(n_timesteps,n_features):
  model=Sequential()
  model.add(Conv1D(filters=64,kernel_size=5,padding="valid",activation="elu"))
  model.add(Conv1D(filters=32,kernel_size=5,activation="elu"))
  model.add(MaxPooling1D(pool_size=3))
  model.add(Conv1D(filters=32,kernel_size=5,activation="elu"))
  model.add(Conv1D(filters=32, kernel_size = 5, activation='elu'))
  model.add(Dropout(0.2))
  model.add(Conv1D(filters=16, kernel_size = 5, activation='elu'))
  model.add(Conv1D(filters=16, kernel_size = 5, activation='elu'))
  model.add(Conv1D(filters=8,kernel_size=5,activation="elu"))
  model.add(GlobalAveragePooling1D())
  model.add(Dense(40, activation="elu"))
  model.add(Dense(1,activation="sigmoid"))
  model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["mae","accuracy"])
  return model

cnn_model=build_CNN_model(x_train.shape[1],x_train.shape[2])
history=cnn_model.fit(x_train, y_train, epochs=20,validation_split=0.3,batch_size=32,verbose=1)
```

圖 14：改變活化函數之程式碼（擷取自作者自行運算程式時的畫面）

2. 操縱變因：一次放入深度學習的資料筆數（Batch Size）、訓練與測試資料的比例（Validation Split）（見表 12 實驗第 8～13 次）

```
def build_CNN_model(n_timesteps,n_features):
  model=Sequential()
  model.add(Conv1D(filters=64,kernel_size=5,padding="valid",activation="relu"))
  model.add(Conv1D(filters=32,kernel_size=5,activation="relu"))
  model.add(MaxPooling1D(pool_size=3))
  model.add(Conv1D(filters=32,kernel_size=5,activation="relu"))
  model.add(Conv1D(filters=32, kernel_size = 5, activation='relu'))
  model.add(Dropout(0.2))
  model.add(Conv1D(filters=16, kernel_size = 5, activation='relu'))
  model.add(Conv1D(filters=16, kernel_size = 5, activation='relu'))
  model.add(Conv1D(filters=8,kernel_size=5,activation="relu"))
  model.add(GlobalAveragePooling1D())
  model.add(Dense(40, activation="relu"))
  model.add(Dense(1,activation="sigmoid"))
  model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["mae","accuracy"])
  return model

cnn_model=build_CNN_model(x_train.shape[1],x_train.shape[2])
history=cnn_model.fit(x_train, y_train, epochs=20,validation_split=0.1,batch_size=64,verbose=1)
```

圖 15：改變 Batch Size 與 Validation Split 之程式碼（擷取自作者自行運算程式時的畫面）

3. 操縱變因：訓練週期（Epochs）：

(1) 新增**準確率（Accuracy）**做為模型評估指標

(2) Epochs：逐漸增加其值，觀察其與準確率之關係（見表 12 實驗第 14～20 次）

```
def build_CNN_model(n_timesteps,n_features):
  model = Sequential()
  model.add(Conv1D(filters=64,kernel_size=5,padding="valid",activation="relu"))
  model.add(Conv1D(filters=32,kernel_size=5,activation="gelu"))
  model.add(MaxPooling1D(pool_size=3))
  model.add(Conv1D(filters=32,kernel_size=5,activation="elu"))
  model.add(Conv1D(filters=32, kernel_size=5, activation="relu"))
  model.add(Dropout(0.2))
  model.add(Conv1D(filters=16, kernel_size=5, activation="gelu"))
  model.add(Conv1D(filters=16,kernel_size=5,activation="elu"))
  model.add(GlobalAveragePooling1D())
  model.add(Dense(40, activation="relu"))
  model.add(Dense(1,activation="sigmoid"))
  model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["mae","accuracy"])
  return model

cnn_model = build_CNN_model(x_train.shape[1],x_train.shape[2])
history = cnn_model.fit(x_train, y_train, epochs=150,validation_split=0.2,batch_size=64,verbose=1)
```

圖 16：改變 Epochs 並加上 Accuracy 之程式碼（擷取自作者自行運算程式時的畫面）

（四）機器學習模型的建立：

1. 放入隨機森林模型進行機器學習並存取模型成 joblib 檔案：

```
rfModel = RandomForestClassifier(n_estimators=50, criterion='gini')
history = rfModel.fit(x_train, y_train)
joblib.dump(rfModel, 'mitbih__RF_1.pk1')
predicted = rfModel.predict(x_test)
predicted = np.argmax(predicted, axis=1)

print('特徵重要度', rfModel.feature_importances_)
```

圖 17：機器學習程式碼（擷取自作者自行運算程式時的畫面）

2. 繪製混淆矩陣：

```
cm = confusion_matrix(y_test, predicted, labels = [0, 1, 2, 3, 4, 5])  # 建構混淆矩陣
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=gt_label)  # 繪製混淆矩陣
disp.plot()
plt.show()
```

圖 18：混淆矩陣程式碼（擷取自作者自行運算程式時的畫面）

（五）機器學習模型的修改與優化：

1. 因本資料庫資料量分配並不平均（如表 1），導致先前模型訓練準確度不佳，因此我們決定將模型根據資料數量分為三類（如圖 19）：

   (1) 判斷是否為階段 N2（資料量最多）

   (2) 判斷是否為階段 N1（資料量次之）

   (3) 判斷是否為階段 R（資料量第三）

   (4) 判斷為 N3 或 N4（加總資料量與階段 R 相近）

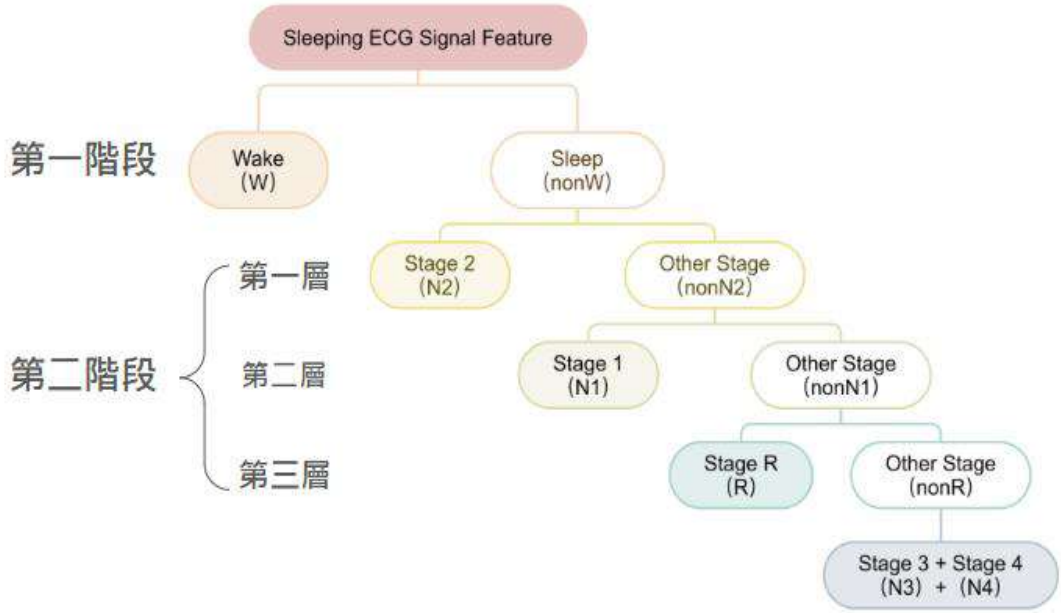



圖 19：睡眠週期階層式判斷（圖片由作者自行繪製）

```
gt_label=["2","1","3","4","R"] #set up index
i=0
while i<len(y):
    try:
        if gt_label.index(y[i])>0:
            f_x.append(x[i])
            f_y.append(1)
            if gt_label.index(y[i])>1:
                s_x.append(x[i])
                s_y.append(1)
                if gt_label.index(y[i])==4:
                    t_x.append(x[i])
                    t_y.append(0)
                else:
                    t_x.append(x[i])
                    t_y.append(1)
                    four_x.append(x[i])
                    four_y.append(gt_label.index(y[i])-2)
            else:
                s_x.append(x[i])
                s_y.append(0)
        else:
            f_x.append(x[i])
            f_y.append(0)
        i+=1
    except:
        y.pop(i)
        x.pop(i)
```

圖 20：睡眠階段分層程式碼（擷取自作者自行運算程式時的畫面）

2. 利用 Oversampler 與 Undersampler 進行機器學習，以平衡資料類別差異。

```
oversampler=RandomOverSampler(random_state=0)
undersampler=RandomUnderSampler(random_state=0)
```

圖 21：Oversampler 與 Undersampler 匯入程式碼（擷取自作者自行運算程式時的畫面）

3. 利用 Python 函式庫**網格分析**進行機器學習，先參考文獻以及先前實驗來找出較適合的數值，藉以設定範圍以運算出最佳參數組合之模型。

```
param_grid={
    "n_estimators":[100,110,130,180], #100~200
    "max_depth":[10,15,20],   #10~30
    "min_samples_split":[i for i in range(2,11)], #1~10
    "min_samples_leaf":[i for i in range(1,6)] #1~5
}
```

圖 22：網格分析程式碼（擷取自作者自行運算程式時的畫面）

4. 使用 Hard Voting 整合不同機器學習方法的判斷結果，以提升判斷精確度。

(1) 使用 RF、DT、KNN 三種適合二維圖形的機器學習方法作為子模型。

```
f_x_train, f_x_test, f_y_train, f_y_test=train_test_split(f_x,f_y,test_size=0.2, random_state=0)
#f_x_train,f_y_train=oversampler.fit_resample(f_x_train,f_y_train)
f_x_train=np.array(f_x_train)
f_x_test=np.array(f_x_test)
f_y_train=np.array(f_y_train)
f_y_test=np.array(f_y_test)

f_rfModel=GridSearchCV(estimator=RandomForestClassifier(bootstrap=True),param_grid=param_grid,cv=5,n_jobs=-1,verbose=2)
f_history=f_rfModel.fit(f_x_train,f_y_train)
#joblib.dump(f_rfModel.best_estimator_,"mit_bih_RF_F_5.joblib")
f_predicted=f_rfModel.best_estimator_.predict(f_x_test)
f_predicted=(f_predicted>0.5).astype(int)

print("Best parameters found:",f_rfModel.best_params_) #
print("特徵重要度:",f_rfModel.best_estimator_.feature_importances_)
print("Score:",f_rfModel.best_estimator_.score(f_x_test,f_y_test)) #

f_gt_label=["2","others"] #set up index
cm=confusion_matrix(f_y_test, f_predicted, labels=[0,1])
disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=f_gt_label)
disp.plot()
plt.show()
```

圖 23： RF 分層式機器學習（以第一層為例）（擷取自作者自行運算程式時的畫面）

a. 將 20%的睡眠段落當成最終測試資料，其餘的 80%為訓練資料。

b. 參數調整：

表 6：本研究使用之 RF 參數功能與意義

| 參數名稱 | 功能、意義 |
|---|---|
| N_Estimators | 決定隨機森林中子樹的數量 |
| Max_Depth | 決定子樹的最大深度 |
| Min_Samples_Split | 決定要分枝需要的最小資料量 |
| Min_Samples_Leaf | 決定葉子（子節點）的大小（最小資料量） |

註：紅底者為本實驗作為操縱變因調整之項目

c. 使用網格分析將隨機森林機器學習後的最佳模型儲存、計算分數（Score）、繪製混淆矩陣。

```
f_x_train, f_x_test, f_y_train, f_y_test=train_test_split(f_x,f_y,test_size=0.2, random_state=0)
#f_x_train,f_y_train=oversampler.fit_resample(f_x_train,f_y_train)
f_x_train=np.array(f_x_train)
f_x_test=np.array(f_x_test)
f_y_train=np.array(f_y_train)
f_y_test=np.array(f_y_test)

f_model=GridSearchCV(estimator=DecisionTreeClassifier(),param_grid=param_grid,cv=5,n_jobs=-1,verbose=2)
f_model.fit(f_x_train,f_y_train)

joblib.dump(f_model.best_estimator_,"../dt_f_1.joblib")

f_predicted=f_model.best_estimator_.predict(f_x_test)
f_predicted=(f_predicted>0.5).astype(int)

print("Best parameters found:",f_model.best_params_)
print("特徵重要度:",f_model.best_estimator_.feature_importances_)
print("Score:",f_model.best_estimator_.score(f_x_test,f_y_test))

f_gt_label=["2","others"]
cm=confusion_matrix(f_y_test, f_predicted, labels=[0,1])
disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=f_gt_label)
disp.plot()
plt.show()
```

圖 24：DT 分層式機器學習（以第一層為例）（擷取自作者自行運算程式時的畫面）

a. 將 20%的睡眠段落當成最終測試資料，其餘的 80%為訓練資料。

b. 參數調整：

表 7：本研究使用之 DT 參數功能與意義

| 參數名稱 | 功能、意義 |
|---|---|
| splitter | 控制決策樹的隨機性與特徵劃分標準 |
| max_depth | 決定決策樹的最大深度 |
| criterion | 不純度（不純度愈高，擬合愈低）相關指標最優化之調整用參數 |

註：紅底者為本實驗作為操縱變因調整之項目。

c. 使用網格分析將決策數機器學習後的最佳模型儲存、計算分數（Score）、繪製混淆矩陣。

```
f_x_train, f_x_test, f_y_train, f_y_test=train_test_split(f_x,f_y,test_size=0.2, random_state=0)
#f_x_train,f_y_train=oversampler.fit_resample(f_x_train,f_y_train)
f_x_train=np.array(f_x_train)
f_x_test=np.array(f_x_test)
f_y_train=np.array(f_y_train)
f_y_test=np.array(f_y_test)

#first
for i in range(1,31):
  knn = KNeighborsClassifier(n_neighbors=i)
  knn.fit(f_x_train,f_y_train)
  pred_i = knn.predict(f_x_test)
  f_error_rate.append(np.mean(pred_i != f_y_test))
  f_error_all.append(pred_i)
  if f_error_rate[i-1]<f_error_rate[i-2]:
    joblib.dump(knn,"knn_f_2.joblib")

plt.figure(figsize=(10,6))
plt.plot(range(1,31),f_error_rate,color='blue',linestyle='dashed',marker='o',markerfacecolor='red',markersize=10)
plt.title('Error Rate vs. K Value - 1')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()

f_best=f_error_rate.index(min(f_error_rate))
print(f_best+1)
print(confusion_matrix(f_y_test,f_error_all[f_best]))
print(classification_report(f_y_test,f_error_all[f_best]))

f_gt_label=["2","others"]
cm=confusion_matrix(f_y_test.astype(int),pd.Series(f_error_all[f_best]).astype(int), labels=[0,1])
disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=f_gt_label)
disp.plot()
plt.show()
```

圖 25：KNN 分層式機器學習（以第一層為例）（擷取自作者自行運算程式時的畫面）

a. 將 20%的睡眠段落當成最終測試資料，其餘的 80%為訓練資料。

b. 以 n_neighbor 參數 1～30 分別使 KNN 進行機器學習，將其中最好的結果。儲存成模型、計算分數（Score）、繪製混淆矩陣，並繪製 30 次參數調整中失敗率曲線圖（為 U 型曲線）。

表 8：本研究使用之 KNN 機器學習參數功能與意義

| 參數名稱 | 功能、意義 |
|---|---|
| n_neighbors | 決定目標周圍選取的資料點數（鄰居個數） |

註：紅底者為本研究作為操縱變因調整之項目。

(2) 使用 Hard Voting 整合三個機器學習模型

```
f_RF=joblib.load("./models/F_RF.joblib")
f_knn=joblib.load("./models/F_KNN.joblib")
f_dt=joblib.load("./models/F_DT.joblib")
```

圖 26：載入前述三次訓練存下之模型（以第一層為例）（擷取自作者自行運算程式時的畫面）

```
f_x_train, f_x_test, f_y_train, f_y_test=train_test_split(f_x,f_y,test_size=0.2, random_state=0)
f_x_train=np.array(f_x_train)
f_x_test=np.array(f_x_test)
f_y_train=np.array(f_y_train)
f_y_test=np.array(f_y_test)

f_model_list=[]
f_model_list.append(('RF',f_RF))
f_model_list.append(('knn',f_knn))
f_model_list.append(('dt',f_dt))
vc=VotingClassifier(f_model_list)

f_model=vc.fit(f_x_train,f_y_train)

joblib.dump(f_model,"../voting_f_1.joblib")

f_predicted=vc.predict(f_x_test)
f_score=vc.score(f_x_test,f_y_test)

print("Score:",f_score)

f_gt_label=["2","others"] #set up index
cm=confusion_matrix(f_y_test, f_predicted, labels=[0,1])
disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=f_gt_label)
disp.plot()
plt.show()
```

圖 27：Voting 分層式機器學習（以第一層為例）（擷取自作者自行運算程式時的畫面）

a. 將 20%的睡眠段落當成最終測試資料，其餘的 80%為訓練資料

b. 以 Hard Voting 進行機器學習，並儲存模型、計算分數(Score)

（六）使用者介面的建立：

1. 睡眠分析：

我們首先以 Kivy 語言編排出簡單的初始介面，接著建立受試者選單，並設定按下開始後之分析指令以及進度條更新程式，以建立分析系統，最後再繪製 ECG 波形與睡眠階段切分圖與繪製睡眠階段比例圓餅圖。

```
def start_button(self):
    print('Press start.....')

    threading.Thread(target = self.startAnalytics).start()

def update_progress(self, dt):
    self.ids.analytics_progress.value = self.progressValue

    if self.progressValue != 1:
        self.ids.file_path.text = "分析中 … " + str(int(self.progressValue * 100)) + "%"

        if self.progressValue == 0:
            self.ids.ecg_diagram.opacity = 0
            self.ids.pie_chart.opacity = 0
            self.ids.score.text = "睡眠評分："
            self.ids.accuracy.text = "準確率：     "
            self.ids.time.text = "階段時間：\nW：\nR：\n1：\n2：\n3：\n4："
    else:
        self.ids.file_path.text = self.ids.drop_down_sub.text
```

圖 28：分析指令設定程式碼（擷取自作者自行運算程式時的畫面）

2. 睡眠評分標準：

**條件一的睡眠評分：**

$$r(N1) = \left|\frac{t(N1)}{t(N1)+t(N2)+t(N3)+t(R)} \times 100 - 5\right|$$

$$r(N2) = \left|\frac{t(N2)}{t(N1)+t(N2)+t(N3)+t(R)} \times 100 - 50\right|$$

$$r(N3) = \left|\frac{t(N3)}{t(N1)+t(N2)+t(N3)+t(R)} \times 100 - 20\right|$$

$$r(R) = \left|\frac{t(R)}{t(N1)+t(N2)+t(N3)+t(R)} \times 100 - 25\right|$$

$$r(all) = 0.25 \cdot \big(100 - r(N1)\big) + 0.25 \cdot \big(100 - r(N2)\big)0.25 \cdot \big(100 - r(N3)\big)0.25 \cdot \big(100 - r(R)\big)$$

**簡化後-**

$$r(all) = 0.25 \cdot (400 - r(N1) - r(N2) - r(N3) - r(N4)$$

**條件二的睡眠評分：**

$$d(all) = 100 - \left|\frac{t(N3)}{t(N1)+t(N2)+t(N3)+t(R)+t(W)} \times 100 - 20.5\right|$$

**條件三的睡眠評分：**

$$w(all) = \frac{t(N1)+t(N2)+t(N3)+t(R)}{t(N1)+t(N2)+t(N3)+t(R)+t(W)} \times 100$$

**加權後的總體睡眠評分：**

$$s(all) = r(all) \cdot 0.4 + d(all) \cdot 0.35 + w(all) \cdot 0.25$$

圖 29：睡眠評分計算公式（圖片由作者自行繪製）

(1) 本睡眠評分參考之睡眠品質條件為以下三個：

a. 條件一：各睡眠階段與理想比例（見表 2）之重合成度

b. 條件二：深眠(N3)需佔總睡眠時長 16%~25%（平均值 20.5%）

c. 條件三：是否達到高睡眠效率（睡眠效率 = 睡眠時間 / 躺床時間）

(2) t(x)為時長函數、N3 為 N3 與 N4 的加總

(3) 加權參考各睡眠階段之判別準確度與各睡眠條件影響範圍調整至 0.4, 0.35, 0.25

(4) 睡眠評分與睡眠品質的對應：

表 9：睡眠評分與睡眠品質之對應表

| **分數（0~100）** | **等第** | **睡眠狀況** |
|---|---|---|
| 90~100 | A | 理想，符合標準比例 |
| 70~89 | B | 中等，雖與睡眠比例有部分不符，但大致上仍有達到一定標準 |
| 60~69 | C | 不理想，與理想值不符 |
| 0~59 | D | 非常不理想，睡眠階段之輪替、睡眠效率與深眠比例都有可能需要改進 |

3. 助眠方法：

以 Kivy 語言中的表格元件建立一助眠方法頁面。

4. 睡眠諮詢：

以 Kivy 語言中的表格元件建立一助眠方法頁面。

# 肆、研究結果與討論

一、實驗結果分析

（一）深度學習（CNN）：

1. 初始模型：

一個良好訓練之模型，紅色框線部分(True Positive, TP / True Negative, TN)資料量應大於藍色框線部分(False Positive, FP / False Negative, FP)，然而在睡眠狀態的部分資料誤判成清醒狀態之比例過高，甚至超越判斷正確的部分，因此可得知此模型尚有改善空間。

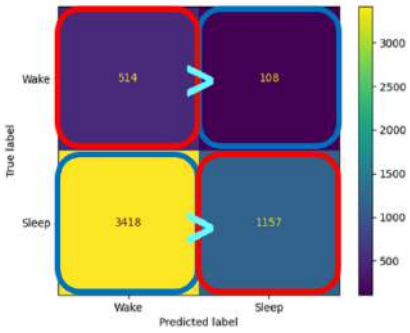
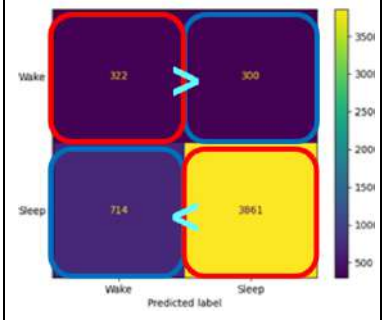
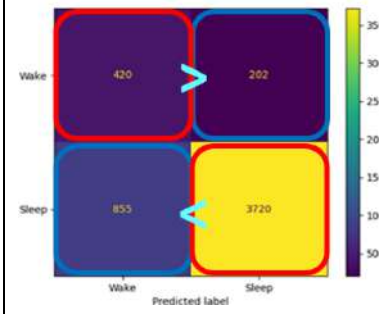
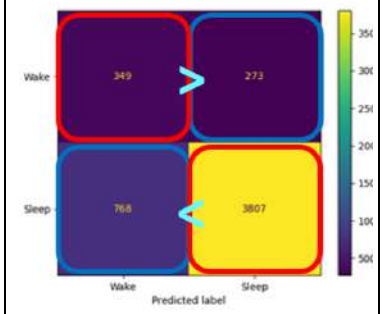



圖 30：初始模型之混淆矩陣（圖片由作者自行繪製）

2. 以活化函數（Activation）作為操縱變因之模型：

**Activation = Elu、Gelu 與 Relu** 時，TP 與 TN 的比例為最高，並且解決了初始模型誤判比例大於正確比例的問題，混淆矩陣判讀結果為各 Activation 中最佳，其中，Gelu 在判斷清醒方面為最理想，Elu 則在判斷睡眠方面為最理想，而整體資料判讀正確量大幅提升，模型獲得優化。

表 10：Elu、Gelu 與 Relu 之混淆矩陣（圖片由作者自行繪製）

| Elu | Gelu | Relu |
|---|---|---|
|  |  |  |



3. 以學習訓練與測試資料的比例（Batch Size）與深度學習訓練的回合數（Validation Split）作為操縱變因之模型：

**Batch Size = 64, Validation Split = 0.2** 時，TP 與 TN 的比例為 Batch Size 介於 $2^5$ 至 $2^7$、Validation Split 介於 0.1 至 0.3 時最高者，即判讀正確率最佳者，可看出下方資料判讀正確量亦較初始模型提升，模型獲得優化。

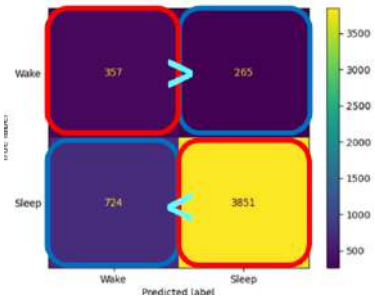



圖 31：Batch Size = 64, Validation Split = 0.2 之混淆矩陣（圖片由作者自行繪製）

4. 以訓練回合數（Epochs）作為操縱變因之模型：

   **於調整 Epochs 時開始加入準確率(Accuracy)函數作為另一判別標準，具較為客觀的優點。而 Epochs = 175** 時（搭配其他參數最佳結果進行調整），**準確率達最高，為 0.90，且混淆矩陣判讀結果 TP 與 TN 比例亦最高，可見其最為準確之特性，為本實驗最佳模型**。

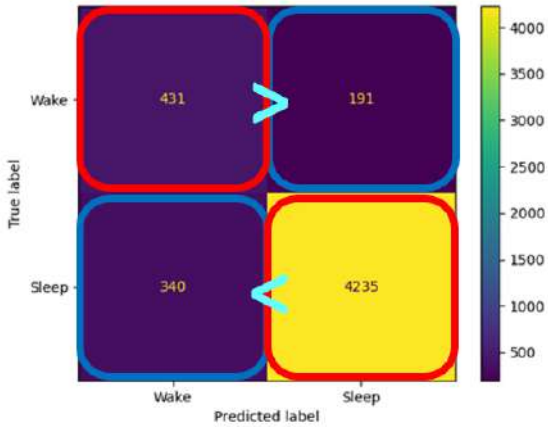



圖 32：Epochs = 175 之混淆矩陣（圖片由作者自行繪製）

5. 透過參數調整來優化模型之過程

   部分參數於本實驗未成為操縱變因，僅作為控制變因之原因：

   (1)受輸入資料量/資料型態影響大於其餘因素（本實驗輸入資料皆為固定）

   (2)有專門適合二維分析的參數值

   (3)數值可變動範圍較小

   (4)影響的層面與初始模型遇到之困境較無關聯

   表 11：控制變因（參數意義見前方表 5）

| Filter | Kernel Size | Dropout | Pooling | Criterion | Loss | Optimizer |
|---|---|---|---|---|---|---|
| 64,32,32, 32,16,16 | 5（全） | 0.2 | 3 | Gini | Binary_crossentropy | Adam |

表 12：操縱變因與實驗結果

| 實驗次數 | Epochs | Activation | Batch Size | Validation Split | Accuracy |
|---|---|---|---|---|---|
| 第一次 | 30 | Relu (all) | 32 | 0.3 | x |
| 第二次 | 30 | Tanh (all) | 32 | 0.3 | x |
| 第三次 | 30 | Selu (all) | 32 | 0.3 | x |
| 第四次 | 30 | Elu (all) | 32 | 0.3 | x |
| 第五次 | 30 | Relu6 (all) | 32 | 0.3 | x |
| 第六次 | 30 | Gelu (all) | 32 | 0.3 | x |
| 第七次 | 30 | Softsign (all) | 32 | 0.3 | x |
| 第八次 | 30 | Relu (all) | 32 | 0.1 | x |
| 第九次 | 30 | Relu (all) | 64 | 0.1 | x |
| 第十次 | 30 | Relu (all) | 128 | 0.1 | x |
| 第十一次 | 30 | Relu (all) | 32 | 0.2 | x |
| 第十二次 | 30 | Relu (all) | 64 | 0.2 | x |
| 第十三次 | 30 | Relu (all) | 64 | 0.3 | x |
| 第十四次 | 120 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.85 |
| 第十五次 | 130 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.85 |
| 第十六次 | 140 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.86 |
| 第十七次 | 150 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.88 |
| 第十八次 | 175 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.9 |
| 第十九次 | 200 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.82 |
| 第二十次 | 225 | Relu, Gelu, Elu, Relu, Gelu, Elu, Relu | 64 | 0.2 | 0.78 |

註：Accuracy 四捨五入到小數點後第二位

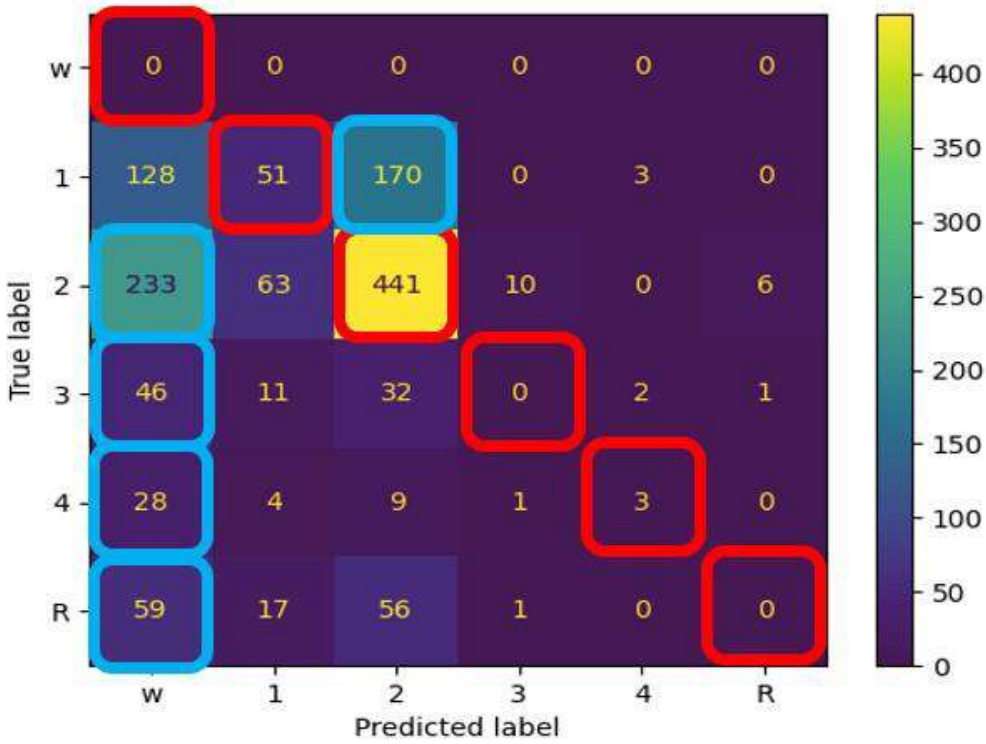
（二）機器學習（ML）：

1. 無階層式分類

表 13：無階層式分類機器學習之混淆矩陣（未輸入清醒階段資料）

（圖片由作者自行繪製）

**RF**




Score：0.36

一個能正確判別睡眠階段的模型，紅色框線部分判別結果資料量應大於其餘部分，意即真實資訊(True Label)與預測結果(Predicted Label)相對應，然而多數資料並未如此，而資料量比其他階段多的 N2 亦被不規則分流，預測為各個睡眠階段，藍色框線部分則是各睡眠階段被錯誤判別的最高項，由此混淆矩陣可得知判別不正確比例過高，模型尚有改善空間。

因此，鑑於資料量的懸殊以及此混淆矩陣的不規律分佈結果，本研究改為階層式分層機器學習，並將數據充足的清醒、睡眠階段以深度學習進行辨識，其餘以機器學習進行，以避免一次性投入多類型資料致使其產生混淆。

2. 階層式分類

表 14：機器學習階層式分類第一層混淆矩陣（圖片由作者自行繪製）

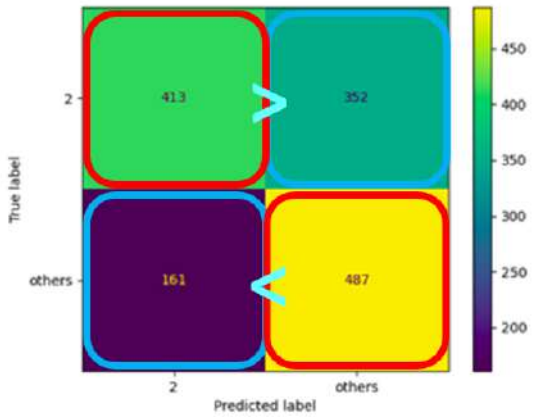
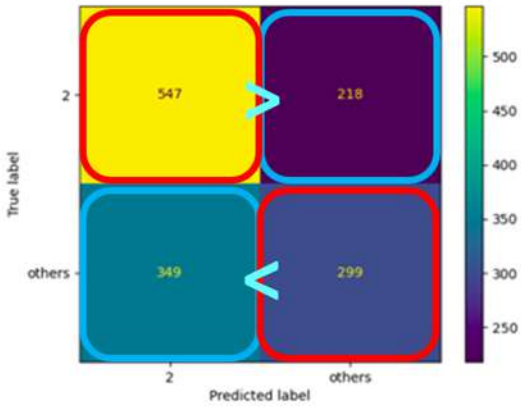
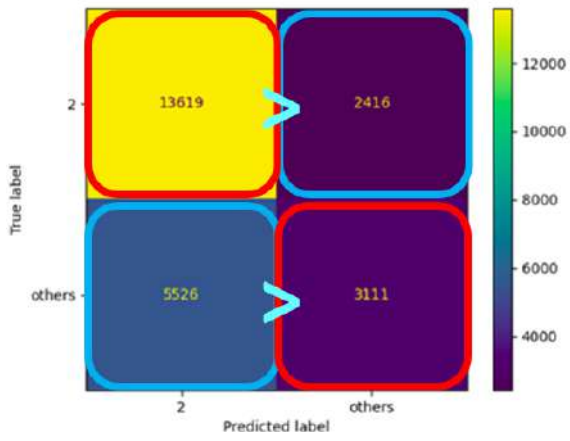
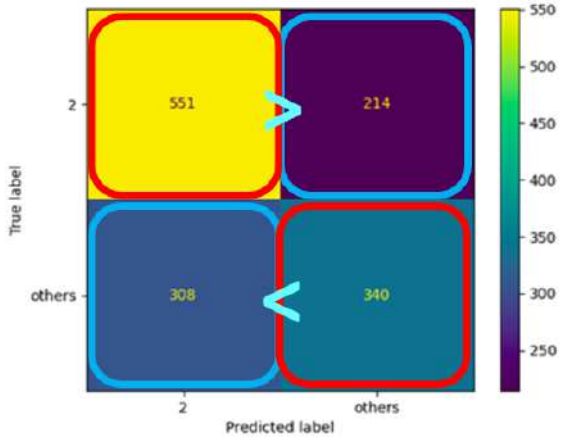
<table>
<tr><td colspan="2">第一層（N2/N1、N2、N3、N4、R）</td></tr>
<tr><td>RF</td><td>DT</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Best parameters found：<br>{'max_depth': 10, 'min_samples_leaf': 3, 'min_samples_split': 4, 'n_estimators': 100}</td><td>Best parameters found：<br>{'criterion': 'entropy', 'max_depth': 10, 'splitter': 'best'}</td></tr>
<tr><td>Score：0.64</td><td>Score：0.60</td></tr>
<tr><td>KNN</td><td>Hard Voting</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Score：0.68</td><td>Score：0.63</td></tr>
</table>

表 15：機器學習階層式分類第二層混淆矩陣（圖片由作者自行繪製）

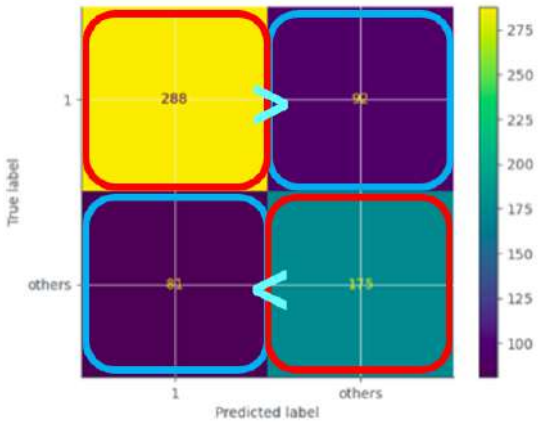
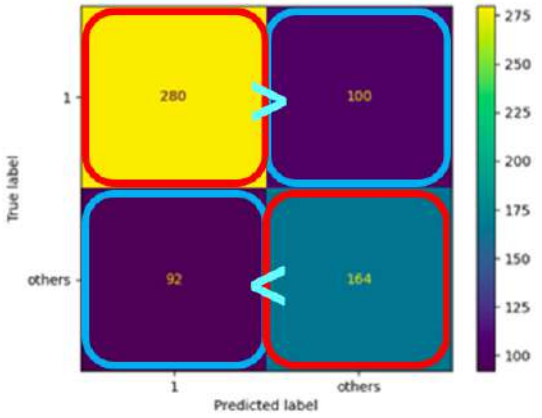
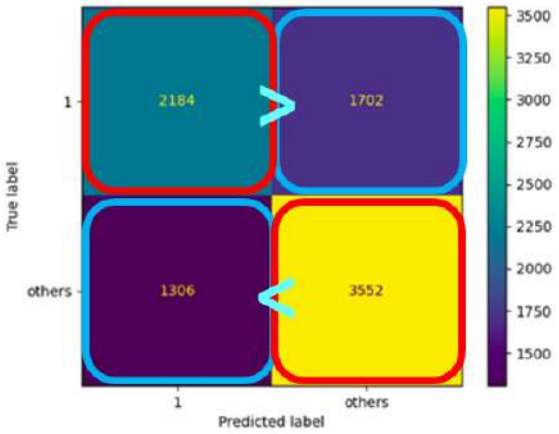
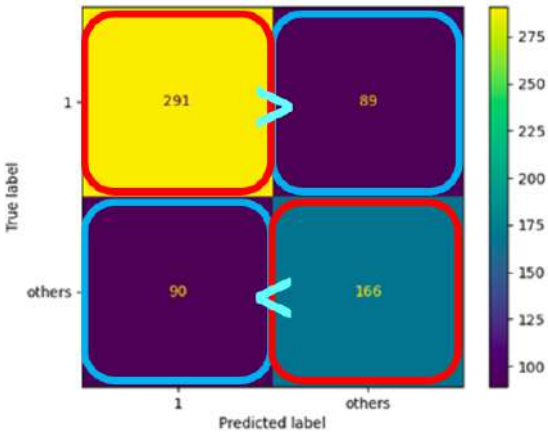
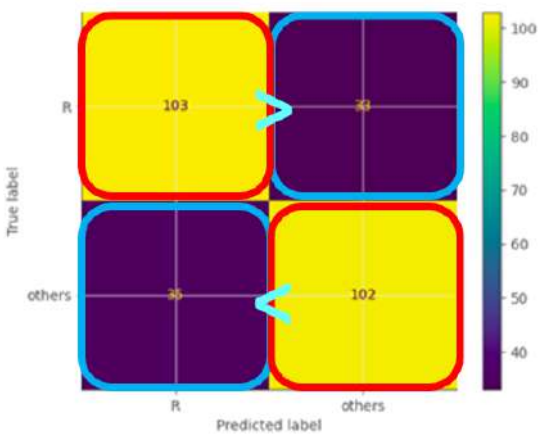
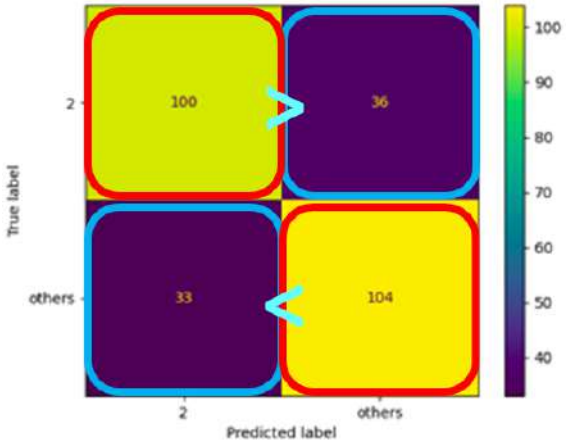
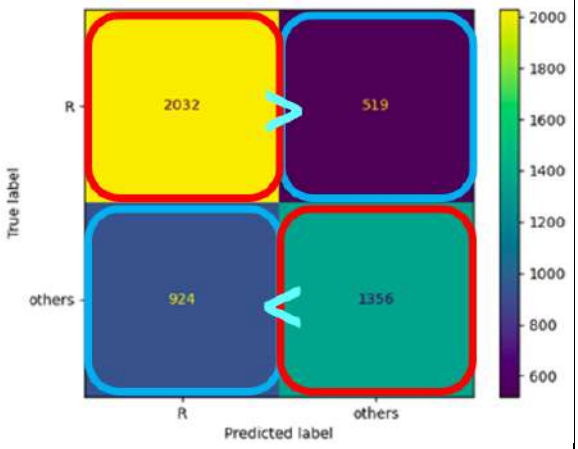
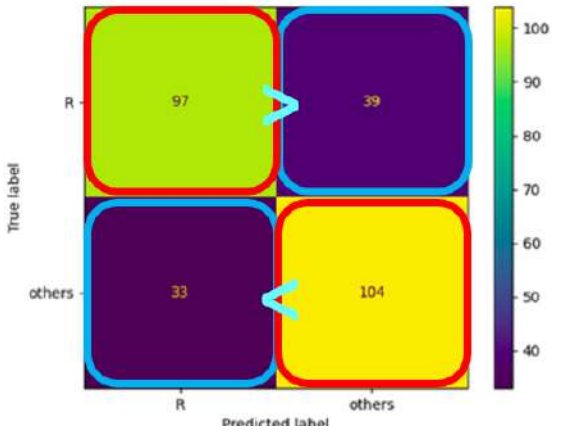
<table>
<tr><th colspan="2">第二層（N2/N3、N4、R）</th></tr>
<tr><th>RF</th><th>DT</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Best parameters found：<br>{'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 100}</td><td>Best parameters found：<br>{'criterion': 'entropy', 'max_depth': 10, 'splitter': 'best'}</td></tr>
<tr><td>Score：0.73</td><td>Score：0.70</td></tr>
<tr><th>KNN</th><th>Hard Voting</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Score：0.66</td><td>Score：0.72</td></tr>
</table>

表 16：機器學習階層式分類第三層混淆矩陣（圖片由作者自行繪製）

<table>
<tr><td colspan="2">第三層（R/N3、N4）</td></tr>
<tr><td>RF</td><td>DT</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Best parameters found：<br>{'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 130}</td><td>Best parameters found：<br>{'criterion': 'entropy', 'max_depth': 20, 'splitter': 'best'}</td></tr>
<tr><td>Score：0.75</td><td>Score：0.75</td></tr>
<tr><td>KNN</td><td>Hard Voting</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Score：0.70</td><td>Score：0.74</td></tr>
</table>

（三）使用者介面（UI）：

1. 睡眠分析：

(1) 初始介面：

透過左上按鈕即可切換**睡眠分析**、**睡眠評分**、**助眠方法**與**睡眠諮詢**頁面。

(2) 受試者選單：

按下按鈕後即可選擇受試者。

(3) 開始分析：

按下開始鍵後，系統便會開始分析該資料之睡眠品質，並透過上方進度條使使用者了解進度。

(4) 分析完成：繪製 ECG 波形與睡眠階段切分圖與睡眠階段比例圓餅圖：分析完成後，系統將生成**睡眠評分**、**準確率**、**各睡眠階段時間比例**、**睡眠階段切分心電圖**及**各睡眠階段時間比例圓餅圖**。

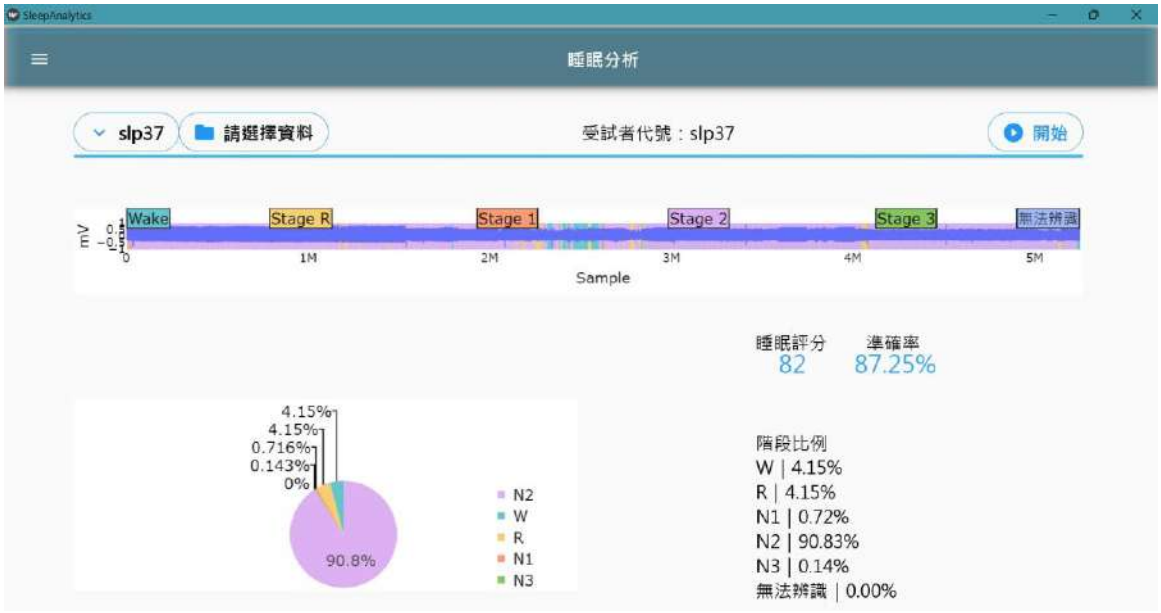



圖 33：分析結果（擷取自作者自行運算程式時的畫面）

註：準確率係指此系統（融合各層 Voting 模型）將該受試者之各睡眠階段判斷成功之機率（與資料庫標籤比對），與睡眠品質無關，睡眠評分則為依照圖 29 的公式與程式判定出來的睡眠階段來對睡眠品質進行的直觀評分。

2. 睡眠評分：

提供使用者較為直觀的睡眠評分（表 9）。

3. 助眠方法與睡眠資訊：

提供使用者各式助眠方法與各種睡眠相關的常見問題之解答。


圖 34-1：助眠方法介面圖

（擷取自作者自行運算程式時的畫面）


34-2：問題諮詢介面

（擷取自作者自行運算程式時的畫面）


圖 34-3：睡眠等第說明介面（擷取自作者自行運算程式時的畫面）

二、討論

1. 各受試者睡眠階段判別準測率差異原因

**(1)過度擬合（Over - fitting）：**

模型**只能為訓練資料提供準確的預測，而無法針對新資料提供準確的預測**。

**(2)擬合不足（Under - fitting）：**

**資料量的模型不易擬合判別函數，**訓練誤差與測試誤差**並沒有**隨著迭代增加訓練而**收斂。**

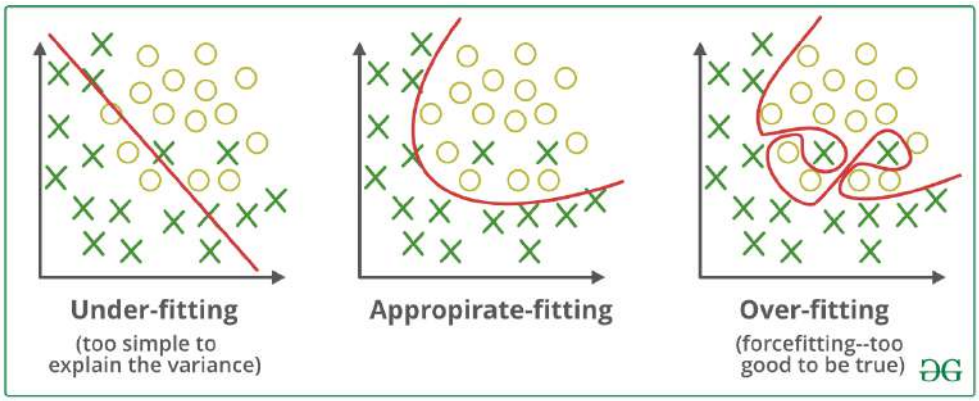



圖 35：擬合不足、正常情況、過度擬合示意圖 [9]

(3)本研究最終模型之不平衡現象：

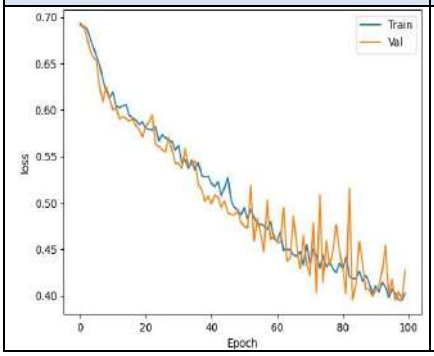
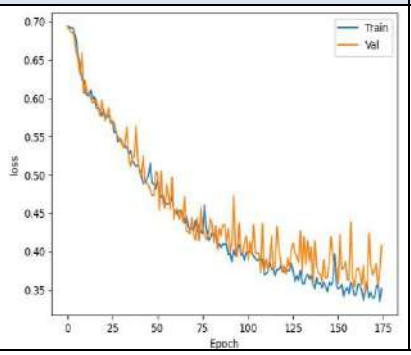
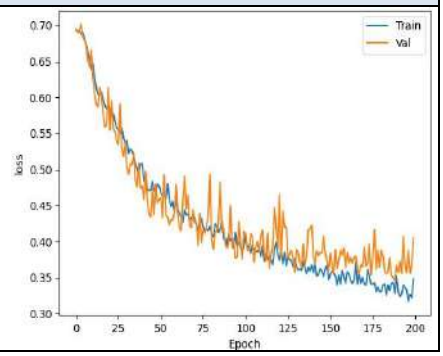
本實驗最後階段逾使用者介面上以最終的階層式模型進行偵測時，不同受試者之判別準確率介於 72%~87%間不等，故本研究之模型雖然能控制準確率維持在 70%以上，但仍有過度擬合的現象，導致在進行判別時，會較有利於特定受試者，而在某些受試者上出現辨識能力較薄弱的問題，雖然嘗試過**調整訓練與測試資料之比例**，使其近乎相同，以訓練時的偏差現象，但仍有提升空間。

2. 深度學習與訓練回合數（Epochs）與準確率（Accuracy）之探討：


圖 36：CNN的準確率（Accuracy）與訓練回合數（Epochs）關係圖（圖片由作者自行繪製）

表 17：CNN 損失率（Loss）圖（以訓練 120、175、200 回合為例）

（圖片由作者自行繪製）

| 訓練 120 回合 | 訓練 175 回合（最佳） | 訓練 200 回合 |
|---|---|---|
|  |  |  |

由圖 36 可看出本研究深度學習 CNN 模型之準確率與訓練回合數關係呈現鐘形曲線，且在 Epochs 達到 175 時擁有最高準確率，在此之前準確率隨回合數增加遞增，之後則漸減，而搭配損失值（Loss）圖來比對，推測出可能的原因是因為在回合數達到 175 之前，模型尚未收斂，仍有進步空間，而在回合數達到 175 之後，則已達到收斂之極限，有局部震盪的趨勢，故準確度沒有再提升。

三、與相關研究之比較

（一）與其他以心電圖（ECG）分辨清醒與睡著狀態之論文的比較

表 18：本研究與其他論文於清醒與睡著部分之比較

| 本研究與其他論文於清醒與睡著部分之比較 | 方法 | 準確率 |
|---|---|---|
| 本研究（第一層模型：清醒與睡著部分） | DL | 0.90 |
| Şule Yücelbaş., *et al.* (2018) | ML | 0.77 |
| Pedro Fonseca., *et al.* (2015) | ML | 0.80 |
| Mourad Adnane., *et al.* (2012) | ML | 0.79 |
| Bülent Yılmaz., *et al.* (2010) | ML | 0.74 |
| Shashank Rao., *et al.* (2019) | ML | 0.73 |
| Ran Wei., *et al.* (2017) | ML | 0.77 |

**註：其他以 ECG 判別睡眠階段論文皆僅判斷至清醒與睡著部分，未再細分其他階段**

（二）與其他偵測裝置的比較

表 19：本研究與其他常見睡眠分析裝置之比較

| 偵測系統 | 使用偵測依據 | 特色 |
|---|---|---|
| **本研究** | ECG | 1. 偵測簡便程度介於專業儀器與智慧手錶之間<br>2. 精準度介於專業儀器與智慧手錶之間<br>3. 判斷標準單一化（僅有 ECG） |
| **智慧手錶** | PPG 為主 | 1. 器材簡便、佩戴方便<br>2. 僅透過血管體積變化來推算相關生理資訊， 精準度較低 |
| **高精度醫療裝置** | EEG、ECG 為主 | 1. 設備精密，精準度高<br>2. 偵測儀器較繁雜、數據處理量龐大<br>3. 成本高昂 |

- 由此表可見，本研究保有其他偵測方法之優點，在簡便程度、精確度等方面都能維持一定水準，可以作為折衷的裝置，進行初步醫療診斷。

# 伍、結論與應用

一、結論

（一）本研究創新使用**階層式分類法**，將心電圖波形與睡眠週期結合，以深度學習及機器學習判別**睡眠階段**。

（二）本研究不同於其他研究使用數個腦電波圖（EEG），提出僅以**單一**心電圖波來細分睡眠階段之創新方法。

（三）第一階段使用免於提取特徵、更為自動化及簡便的**深度學習**來區分清醒與睡著兩狀態，突破其他論文僅使用機器學習模型分析的方法，提升系統在數據分析時的**自主性**。

（四）透過不斷調整參數以提升準確率，第一階段最終準確率達**90%**，相較近年相關論文高出 **10～17%**。

（五）第二階段將模型由原先單層式改為階層式以**平衡資料量差異**，進而優化模型，使第一層準確率達 63%，第二層達 72%，第三層達 75%。

（六）第三階段的資料呈現與分析，本研究建立一睡眠分析系統，設計直觀便捷的使用者操作介面，並整合第一階段與第二階段的判別模型，使系統為使用者分析**睡眠品質**，同時參考睡眠品質衡量標準建立公式以產生**睡眠評分**、提供睡眠建議。

（七）整體系統最終準確率最高達 **87.25%**，成功只藉由**單一偵測方法**分辨睡眠階段，也以更為簡便的深度學習模型分析清醒與睡著和分析睡眠品質，達成研究目的，並於睡眠分析領域有所突破。

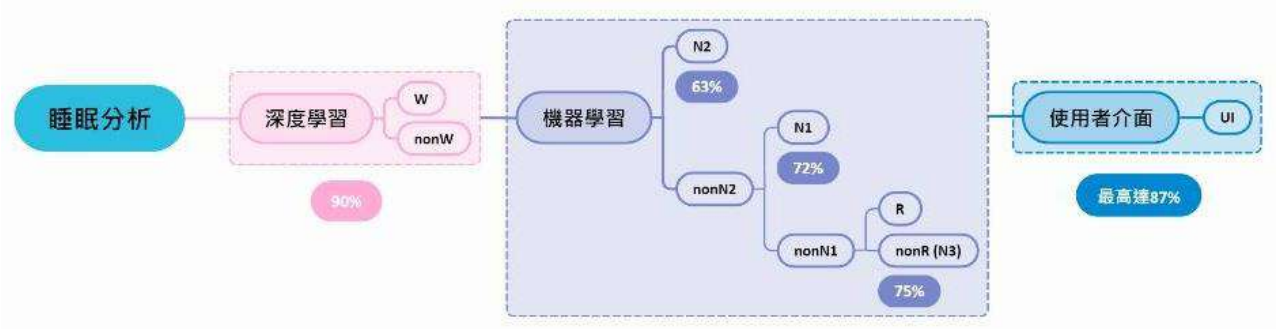



圖 37：本研究各模型最終準確率（機器學習準確率為集成投票之準確率）

（圖片由作者自行繪製）

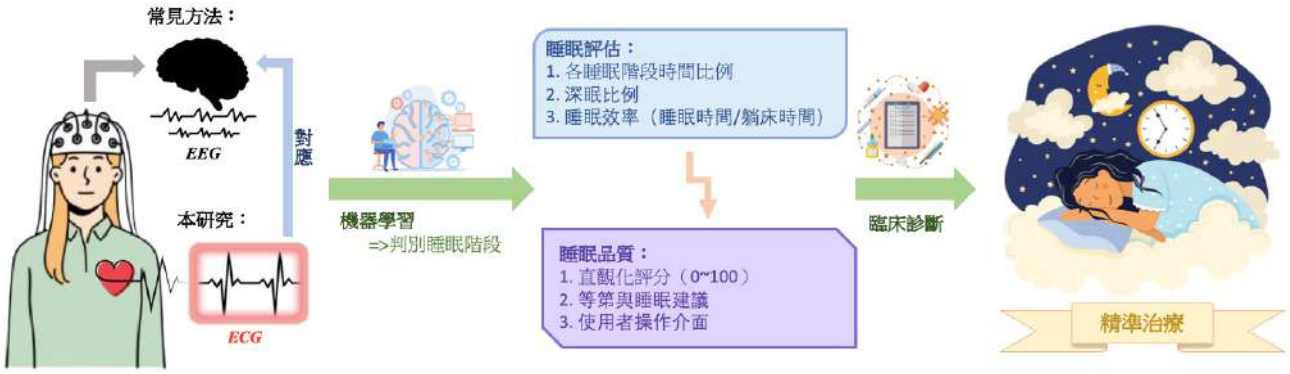



圖 38：本研究總架構圖（圖片由作者自行繪製）

二、未來展望

（一）透過參數調整、訊號處理或其他方法，使機器學習各模型準確度達到 85%以上

（二）整體系統準確率達到 95%以上

（三）透過實際以儀器偵測來協助判斷受試者的睡眠品質

（五）透過使用多執行序列或縮減程式碼來加快運行的速度，使其達成即時判斷的目標

（五）將此系統廣泛應用於臨床醫療診斷

# 陸、參考文獻

[1] Tai, S., Wang, W., Yang, Y. (2015). Current status of sleep quality in Taiwan: a nationwide walk-in survey. Ann Gen Psychiatry, 36(14)

[2] Rundo, J., Downey, R. (2019). Polysomnography. Handbook of Clinical Neurology, 160(25), 381-392

[3] Madona, P., Basti R., Zain M. (2021). PQRST wave detection on ECG signals. Gaceta Sanitaria, 35(2), 364-369

[4] Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R. & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

[5] Driller, M., Dunican, I.,Omond, S., Boukhris, O., Stevenson, S., Lambing, K., Bender, A. (2023). Sleep Achitecture [JPG]. National Library of Medicine.

[6] Liu, Y., (2020). Machine Learning vs. Deep Learning [WEBP]. Medium.

[7] Yeh, J., (2017). CNN Concept [WEBP]. Medium.

[8] Chiou, J., (2024). Confusion Matrix Concept [WEBP]. Medium.

[9] Dewa, I., (2024). Bias and Variance [WEBP]. GeeksforGeeks.

# 【評語】190005

1. 利用心電圖辨識睡眠深淺程度已有文獻支持. 主要挑戰是辨別睡眠階段與睡眠品質. 本作品用ＲＦ, DT, ＫＮＮ, CNN 深度學習來訓練做階層式判斷睡眠階段, 辨識正確率約 90%, 似乎不錯, 但無法確認為何會這麼高。

2. 睡眠品質的評估, 基本上是以各個睡眠階段的比例時間討公式算出來, 階層式訓練方法有點玄機, 但似乎未說明。

3. 機器學習分類器需事先對心電圖做特徵偵測與擷取, 文中若能詳細說明如何做到心電圖的訊號處理則更佳。

4. 除心電圖外亦可加其他如呼吸資料, 來增加睡眠品質分析的基準度。

5. 對於訓練資料與測試資料來源可以進一步分析如老人, 中年人, 青少年與小孩, 或男性女性等。

6. 文中宣稱正確率高於文獻 10%-18%，似乎有點 overclaim。