輔仁大學電機工程學系專題實驗成果總報告

題目

解密智慧電表-用 CNN 深度學習洞悉人口資訊的秘密

Solving the Mystery of Population Data with CNN Deep Learning and Smart Meters

作者

409242023_電機四乙_楊秉晟

409242188_電機四乙_鍾博安

409242322_電機四乙_洪健維

409242528_電機四乙_張晉瑋

指導教授: 余金郎 特聘教授

中華民國 113 年 1 月 3 日

目錄

第一章 緒論

利用智慧電表所提供的大量數據，可以更好的瞭解消費者的習慣，來幫助公用事業和零售商能更有效地提供個性化的服務，本專題是研究如何從智慧電表數據中推斷出這些特徵。由卷積神經網路(CNN)自動從大量檔案中提取特徵，然後由支援向量機(SVM)去辨識消費者特徵，並證明基於 CNN 的研究方法在此愛爾蘭數據集是有效的。

深度學習

深度學習是機器學習領域的一個重要分支，它通過類比人腦神經網路的結構和功能，利用大規模資料訓練模型，實現對複雜資料的學習和推理能力。深度學習的核心是人工神經網路，它由多個層次組成，每一層都包含多個神經元，層與層之間通過權重進行連接。

智慧電表

智慧電表是一種先進的電能計量設備，具備智慧化和數位化功能，用於測量、記錄和管理電能使用情況，且具備許多的好處，包括準確計量和結算電能使用量、提高電力系統的管理效率、降低能源浪費和環境影響、促進能源可持續發展等。它是實現智慧電網、能源互聯網和智慧家居的重要組成部分，為能源管理和能源效率提供了重要支援。

社會人口資訊辨識

社會人口資訊辨識是指通過使用電腦視覺和影像處理技術來分析和識別圖像或視頻資料中的人口資訊。它可以幫助我們瞭解人口的數量、年齡、性別、人群密度和行為等特徵，從而提供重要的社會分析和決策支援。

社會人口資訊辨識通常涉及圖像採集、目標檢測與跟蹤、人臉識別、行為分析等技術。近年來，深度學習和人工智慧的發展為社會人口資訊辨識帶來了新的突破，使得準確性和效率得到顯著提升。

然而，社會人口資訊辨識也引發了一些隱私和倫理問題。在應用社會人口資訊辨識技術時，需要遵守相關的法律法規和隱私保護原則，確保個人資訊的安全和隱私不受侵犯。同時，還需要進行公正、透明和可解釋的使用，避免濫用和歧視等不良後果。

第二章、神經網路介紹

2.1 CNN & MLP

2.1.1 MLP 介紹

MLP 多層感知機（Multilayer Perceptron）是一種神經網路的監督學習模型，也是最基本的神經網路結構之一。MLP 在機器學習和深度學習中廣泛應用，特別是在分類和回歸問題中。

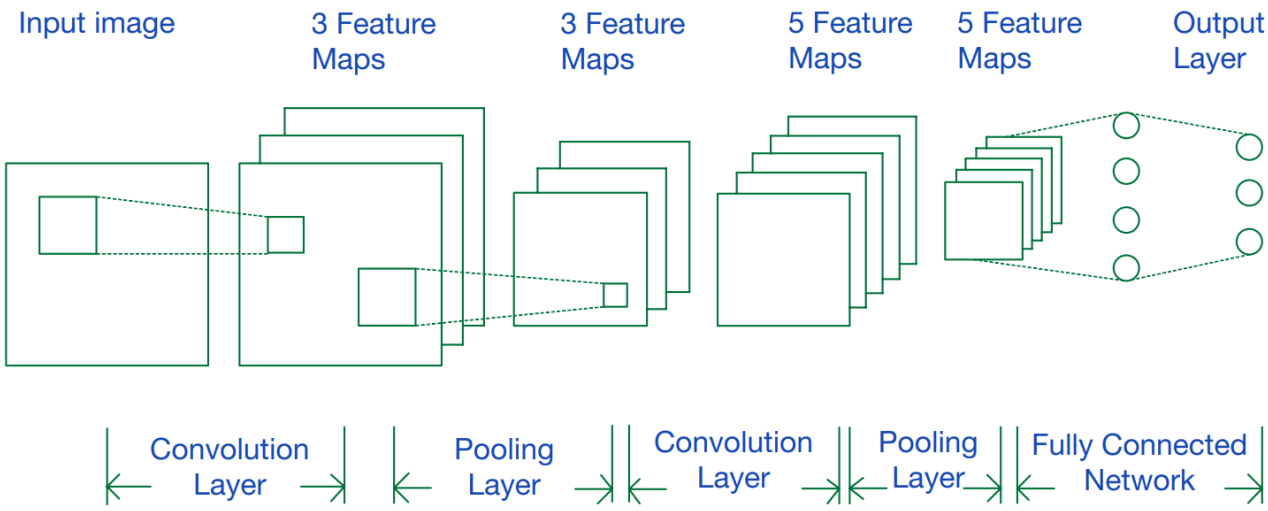
MLP 的結構由多個全連接層（Fully Connected Layer）組成，每個全連接層由多個神經元組成，相鄰層的神經元之間彼此連接。每個神經元接收來自前一層神經元的輸入，並通過權重和偏差進行線性組合，然後通過激活函數進行非線性轉換。

2.1.2 CNN 介紹

CNN 卷積神經網路（Convolutional Neural Network）是一種深度學習模型，主要用於處理具有網格結構的數據，例如圖像、視頻、聲音等。CNN 在圖像處理領域取得了重大的突破，並廣泛應用於圖像識別、物體檢測、人臉識別等任務。

CNN 的基本結構包括三個主要層：卷積層（Convolutional Layer）、池化層（Pooling Layer）和全連接層（Fully Connected Layer）

2.1.3 CNN 的模型與架構圖



卷積層:在深度 CNN 架構中主要是用來進行特徵提取，每個卷積層都有一定數量的特徵過濾層

池化層:此層用於下採樣與保持判別資訊

經過多次卷積、池化後，得到抽象化的特徵，壓扁為一維向量，作為完全連接層的輸入，最終得到輸出

2.1.4 CNN 使用情境

(1)圖像識別： CNN 在圖像識別任務中表現優異，例如將圖像中的物體分類為不同的類別。

(2) 圖像分割： CNN 可用於將圖像分割成不同的區域，每個區域屬於不同的對象或結構

(3) 物體檢測： CNN 可以定位圖像中多個物體的位置，並標識它們的類別，這在自動駕駛、安全監控等場景中很有用。

(4)風格轉換： 通過訓練一個 CNN 模型，可以將一張圖像的風格轉換為另一張圖像，這在藝術創作中具有應用價值。

(5)醫學影像分析： 在醫學領域，CNN 可用於分析醫學影像，幫助醫生診斷和預測疾病。

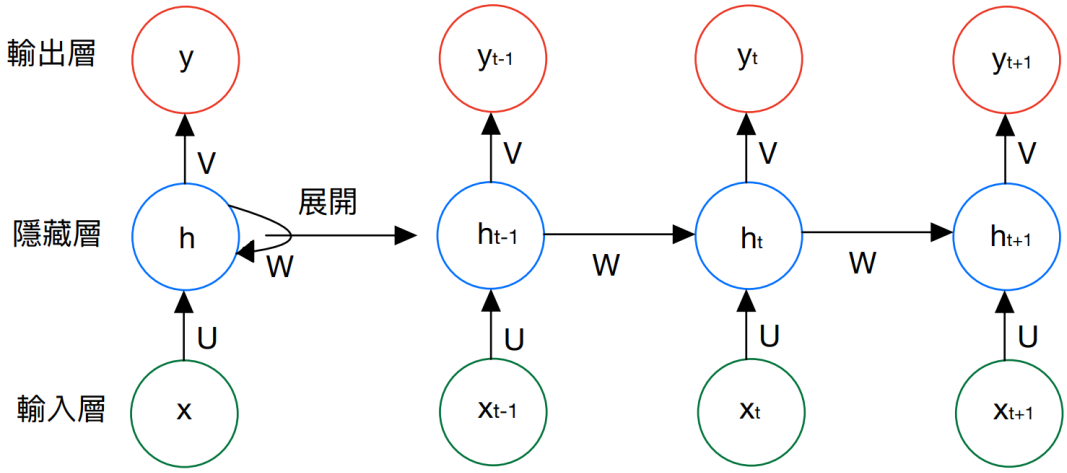
2.2 RNN

2.2.1 RNN 介紹

RNN 遞迴神經網路（Recurrent Neural Network）是一種深度學習模型，它主要用於處理序列型資料，序列型資料指的是按照時間順序排列的數據，例如語音、文本和影片等。

RNN 最主要的特點是能夠在處理序列型資料時，儲存之前的狀態，這使得模型能夠考慮到之前的輸入，對當前的輸入進行預測或分類，廣泛應用於自然語言處理、機器翻譯、語音識別等領域。

2.2.2 RNN 模型與架構圖



時間往右慢慢遞增，每個時間會有一個輸出 y，並把 h 傳到下個時間點，再加上新的輸入一直反覆

2.2.3 RNN 使用情境

自然語言處理(NLP):使電腦能夠理解、生成、處理人類使用的自然語言。機器翻譯(MT):電腦自動將一種自然語言的文本轉換為另一種自然語言的文本的過程。語音辨識:自動識別和轉錄人類語音的過程，將口語語音轉化為文本或指令。

主要優勢在於它能夠適應各種不同長度的輸入序列，並且能夠從過去的輸入中學習到關鍵的上下文信息，進而對當前的輸入進行預測或分類。

2.3 CNN、RNN、MLP 比較

特點	MLP	CNN	RNN
模型結構	由多層全連接層構成	包含卷積層和池化層，可進行特徵提取和降維	循環神經網路層具有記憶功能可處理序列資料
輸入資料	向量或矩陣	二維圖像資料	一維或二維序列資料
資料特徵	無時序姓 適合用於非序列型資料	具有局部相關性和時空特徵 適合用於影像、視頻等資料	具有時序性和長期相關性 適合用於語音、文本等序列資料
應用領域	分類、回歸等	影像識別、物體偵測等	自然語言處理、語音辨識、時間序列預測等
神經元連結方式	全連接層	局部連結 權重共享	循環連接
擴展性	對於高維資料有較強的擴展性	對於高維資料具有較強的擴展性	對序列型資料具有較強的擴展性
優點	訓練速度快 易於實現	能自動提取特徵 對圖像等資料效果較好	能夠處理時序性資料，對於序列型資料效果好
缺點	無法處理時序性資料	對複雜特徵提取和預測能力較弱	易產生梯度消失或梯度爆炸 訓練難度較大

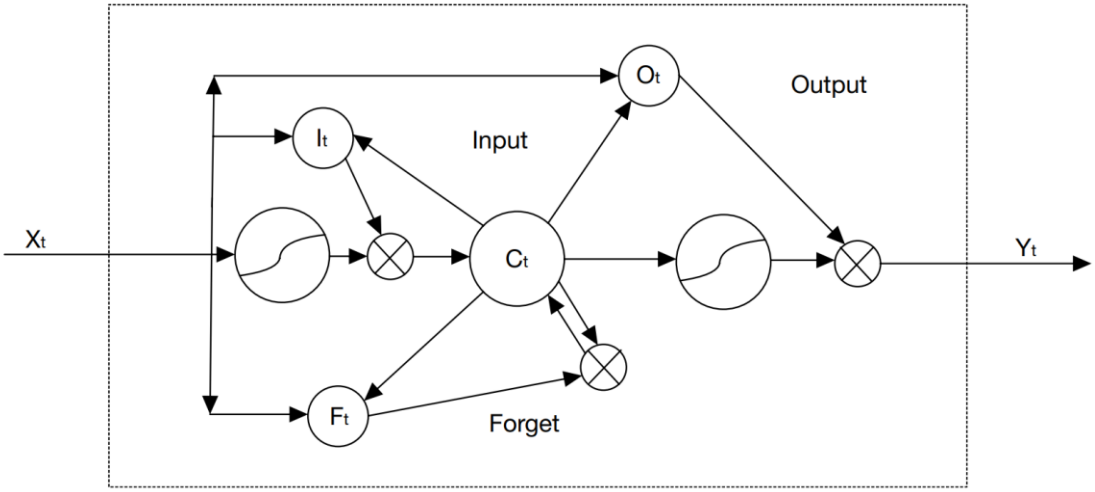
2.4 LSTM

2.4.1 LSTM 介紹

LSTM（Long Short-Term Memory）是一種特殊的遞迴神經網路，能夠有效地解決 RNN 中的梯度消失問題，並且能夠長時間地保持和記憶先前的輸入資訊，廣泛應用在語音辨識、自然語言處理和時間序列分析等領域。

梯度消失:在神經網路訓練時，模型通過計算梯度來進行參數更新。梯度表示了模型參數的變化方向和步長。然而當神經網路較深時，梯度在傳遞過程中可能會變得非常小或者消失，導致模型難以進一步學習，梯度消失問題會影響神經網路的準確性和性能。

2.4.2 LSTM 模型與架構圖



Xt:輸入向量
Yt:輸出向量
Ct:記憶細胞的狀態(cell state)

LSTM 通過一種名為閘門(Gate)的機制控制 cell 記憶細胞的狀態，刪減或增加其中訊息
It:輸入閘門(Input Gate)
Ft:忘記閘門(Forget Gate)
Ot:輸出閘門(Output Gate)
藉由閘門(Gate)機制，LSTM 就可以記住長期記憶

2.5 數字辨識範例(MLP 實作)

數字辨識範例過程(程式附在 7.4)

(1)問題描述：簡要描述數字辨識任務的目標，例如從手寫數字圖像中識別出對應的數字。

(2)數據集介紹：介紹用於訓練和測試的數據集，通常包含手寫數字圖像和對應的標籤（數字標識)。

(3)數據預處理：描述在輸入神經網路之前對數據進行的預處理步驟，如圖像歸一化、像素縮放等。

(4)神經網路架構：介紹用於數字辨識的神經網路架構，例如卷積神經網路（CNN)。

(5)激活函數：解釋用於神經元的激活函數，如 ReLU、sigmoid 等。

(6)損失函數：介紹用於衡量預測結果和真實標籤之間的差距的損失函數，例如交叉熵損失函數。

(7)反向傳播：解釋神經網路是如何進行反向傳播，計算損失函數對於模型參數的梯度，以便更新權重和偏差。

(8)優化算法：描述用於更新模型參數的優化算法，如隨機梯度下降（SGD)、Adam 等。

(9)訓練過程：介紹整個模型訓練的過程，包括多個 epoch 的訓練、批次大小等。

(10)測試和評估：解釋模型訓練後的測試和評估過程，計算模型的準確率和其他指標。

(11)改進方法： 探討提高模型性能的可能改進方法，如調整超參數、引入更大的數據集等。

第三章、專題目標與介紹

3.1 目標

現有的識別消費者人口統計資訊的方法包含三個主要階段

1. 特徵提取以形成特徵集
2. 特徵選取
3. 特徵分類或回歸

在本文中 CNN 是用來提取不同時間和不同日期的電力消耗與消費者的社會人口統計狀況之間的高度非線性關係。並且為了提高辨識能力，使用支援向量機(SVM)代替 softmax 分類器，自動提取消費者的社會人口統計資訊之特徵。

(1)提出一個 CNN 來自動提取智慧電表數據中的特徵，此 CNN 與 SVM 結合來識別消費者的社會人口統計資訊。
(2)由於深度神經網路涉及到大量的參數必須最佳化，有很高的機率會發生 Overfitting，所以提出由提取參數的 CNN 為原則來提升性能
(3)比較的機器學習方法包含 PCA、跟 spare coding 和 spare auto-encoder

3.2 問題定義（定義識別消費者特徵的問題並提出）

在本文中，社會人口統計訊息是從消費者調查中或的，包括性別、年齡、就業、社會階層和居住地。該調查由多個選擇題組成，而這些問題易於消費者回答，並且可以方便的使用一系列離散數字進行編碼。

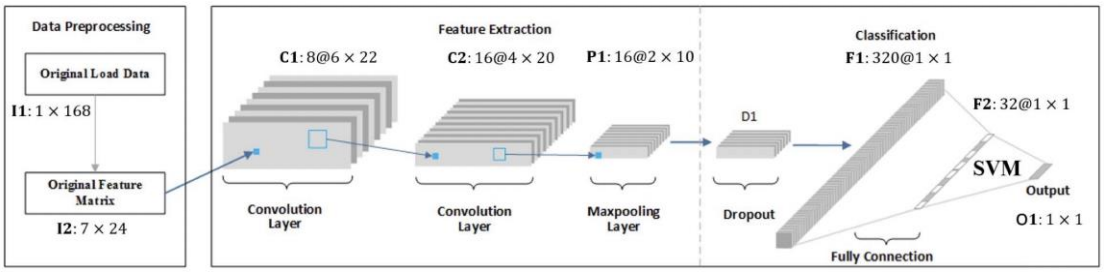
對於社會人口識別的問題來說，有三個應解決的問題:
1. 確定提取的特徵模型來得到輸入數據
2. 決定分類模型來估算標籤
3. 決定訓練方法來達到最優良的分類效果

3.3 方法

提出一種能從大量智慧電表數據中提取特徵的 CNN 架構，並利用減少過度擬合的方法和深度網路的訓練方法

Q.為何我們要使用 CNN 而非其他機器學習技術?
（1）時間的不確定性(ex:白天用電較高，但午夜卻趨近於零)
（2）非線性關係(ex:容易受到天氣條件或日期影響)
（3）數據可視化(CNN 的卷積層可以讓學習過的權重可視化)

3.3.1 網路架構與分層



上圖為深度 CNN 架構共有八層:

i. 卷積層:三層
ii. 池化層:三層
iii. 全連接層:一層
iv. SVM 層:一層

<分層介紹>

(1) 激活函數:資訊沿著每個神經元轉化方法稱為激活函數，神經元的啟動為輸入到輸出之間的函數(EX:RELU 函數就時常被使用於深度 CNN 架構)
(2) 卷積層:在深度 CNN 架構中主要是用來進行特徵提取，每個卷積層都有一定數量的特徵過濾層
(3) dense 層:經過多次卷積、池化後，得到抽象化的特徵，壓扁為一維向量，作為完全連接層(Dense)的輸入
(4) 池化層:此層用於下採樣與保持判別資訊
(5) dropout 層:隨機選擇一部分的資料將它設置為 0，為了避免 overfitting

Q.如何減少 overfitting?

具有大量參數的深度神經網路對於提取特徵和分類的能力是非常強大的,但也應此很容易出現 overfitting。在這種情況下，我們對輸入、模型和訓練方法中進行了改善，以減少深度 CNN 的 overfitting

(1)數據擴充

增加樣本量是減少 overfitting 的有效方法，包含噪聲注入、水平反射、隨機取樣，可以應用在 CNN 的圖形分類上，來增加輸入的大小。

為了進行社會人口信息的辨識問題，我們使用一周的智慧電表數據來代表消費者的每一個社會人口信息,也可以通過個人消費者的用電行為來影響他們的社會人口狀況、天氣狀況、甚至他們的情緒。

每個每周週負荷曲線或多或少都能揭示消費者的社會人口信息，因此數據擴充只是簡單的使用其他週的智慧電表數據作為訓練數據,如果數據集包含 K 週的智慧電表數據，即可以擴大訓練數據集 K 次

(2)Dropout

Dropout 是一種正則化技術，用於減少神經網路的過擬合。在訓練過程中，隨機地丟棄一些神經元，從而減少神經元之間的相互依賴，提高模型的泛化能力。

具體來說，Dropout 通過在每個訓練樣本中隨機丟棄一些神經元的輸出，使它們的輸出值為零。這樣做的效果是，每次訓練時，模型都在不同的子集上進行了訓練，這些子集是原始神經網路的子集。這相當於訓練了許多不同的神經網路，因此可以看作是模型集成的一種方式。

使用 Dropout 的主要原因是為了減少神經網路中神經元之間的共適應（co-adaptation）現象，即神經元彼此相互依賴並在一起適應訓練數據。這種共適應會使得模型對訓練數據過度擬合，導致在新的未見數據上表現不佳。Dropout 的引入可以迫使神經元獨立地學習有用的特徵，而不依賴其他特定的神經元。

(3)減少權重

權重衰減（Weight Decay）是一種常用的正則化技術，用於神經網路模型中。它的目的是為了防止過擬合（overfitting）現象，同時促使模型學習到較為簡單和平滑的權重分佈。

第四章、用電量

4.1 引言

按照論文提供的方法，對用電量的數據要求是**完整一週的用電量**。

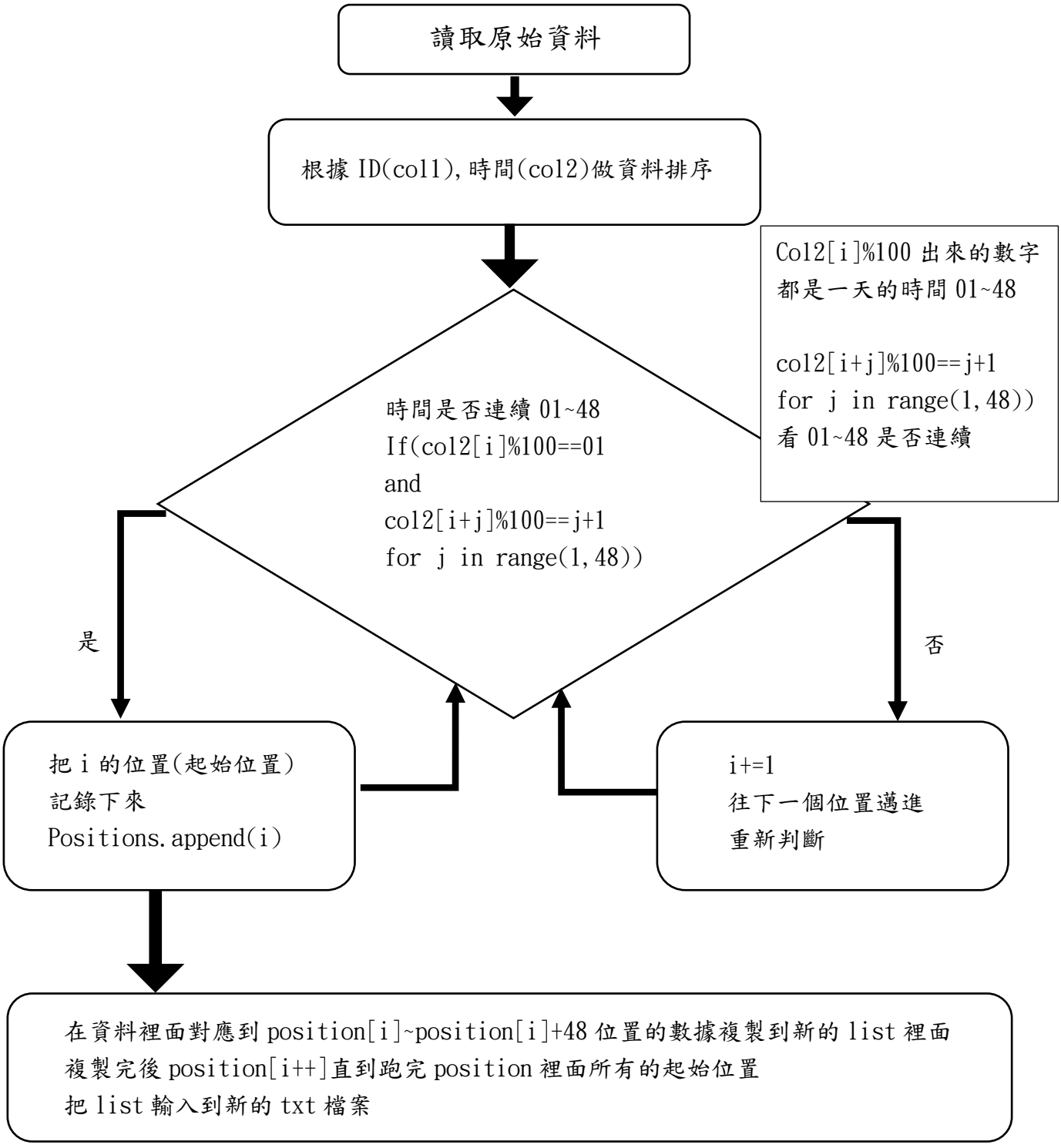
所以從申請到的資料CER Electricity Revised March 2012得到用電量的資料，總共有6個txt檔，檔案內容有三行（col1:ID、col2:記錄的天數和時間 col3:用電量）。

可以發現到，在col2中有紀錄的天數和時間，每天有48筆用電量數據(30分鐘紀錄一次)，在原始資料發現數據的紀錄不完全EX:一天只有14筆數據不是完整的48筆數據，又或是一天50筆數據，那我們從原始資料做第一次的程式處理，觀察每天是不是完整48筆數據，不是的話「刪除」；是的話「保留」，這樣出來的資料會都是完整一天的數據而不會是1天只有14筆。

4.2 用電量架構圖與預處理

第一次處理是保留01-48的完整資料，不完整的就去除。圖4.1為處理01-48的流程圖，大致分為5個步驟。

程式碼請見附錄<程式一>

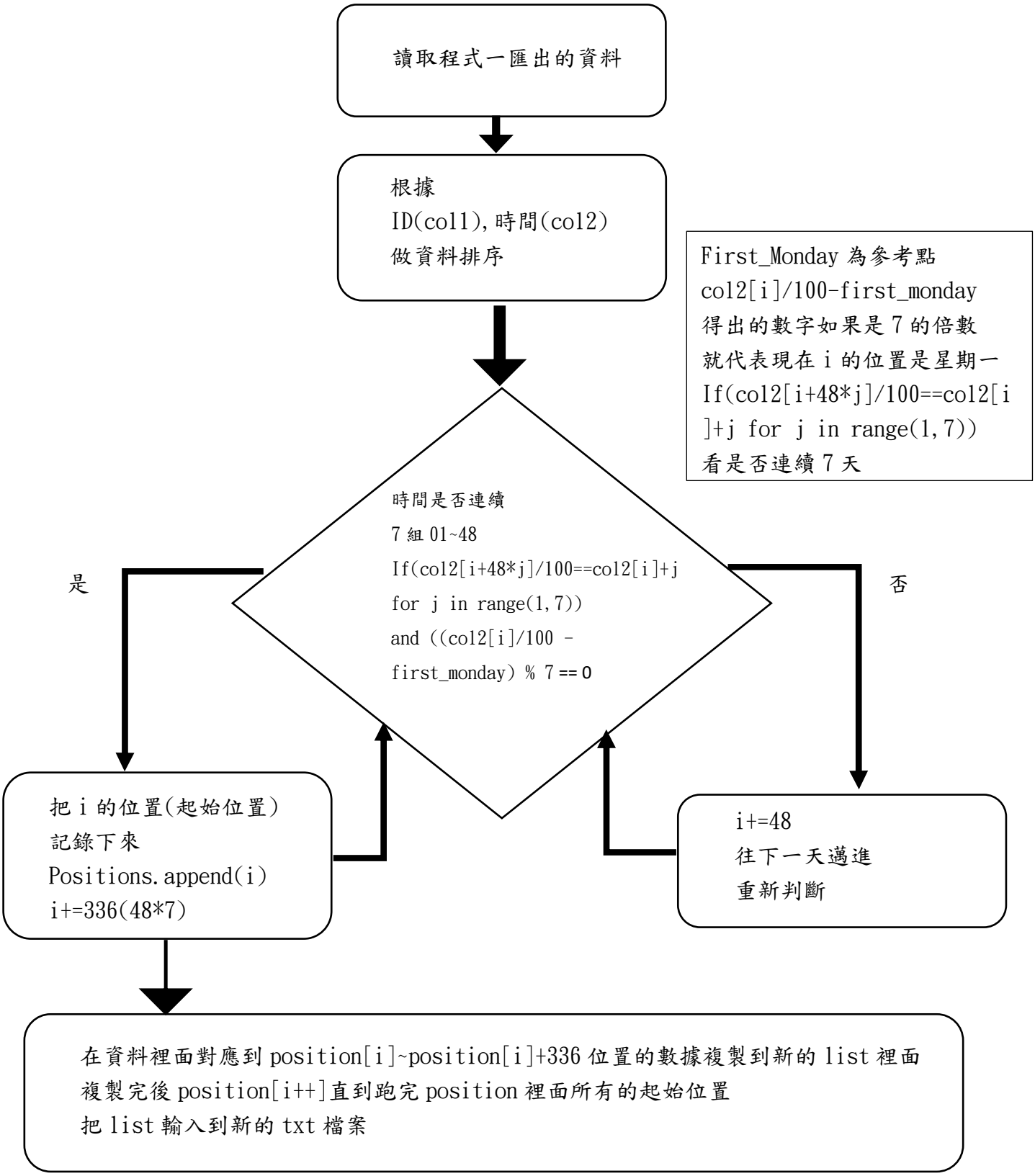

讀取原始資料
根據 ID(col1),時間(col2)做資料排序
Col2[i]%100 出來的數字
都是一天的時間 01~48
col2[i+j]%100==j+1
for j in range(1,48))
看 01~48 是否連續
時間是否連續 01~48
If(col2[i]%100==01
and
col2[i+j]%100==j+1
for j in range(1,48))
是
否
把 i 的位置(起始位置)
記錄下來
Positions.append(i)
i+=1
往下一個位置邁進
重新判斷
在資料裡面對應到 position[i]~position[i]+48 位置的數據複製到新的 list 裡面
複製完後 position[i++]直到跑完 position 裡面所有的起始位置
把 list 輸入到新的 txt 檔案

▲圖 4.1

再來做 1 週完整的程式處理，首先找出 col2(天數)裡面的最小數字，把它定義為星期一(參考點)，接著根據參考點，參考點加上 7 的倍數都是代表星期一，觀察參考點到參考點加上 6 之後是不是完整的數據(48*6)，如果不是的話就刪除，是的話就留下，這樣輸出的資料就是一週完整的數據了。

第二次處理是處理一週 7 天，先進行排序再判斷是否連續。圖 4.2 為處理一週 7 天的流程圖，大致分為 5 個步驟。

程式碼請見附錄<程式二>


讀取程式一匯出的資料
根據
ID(col1),時間(col2)
做資料排序
First_Monday 為參考點
col2[i]/100-first_monday
得出的數字如果是 7 的倍數
就代表現在 i 的位置是星期一
If(col2[i+48*j]/100==col2[i]+j for j in range(1,7))
看是否連續 7 天
時間是否連續
7 組 01~48
If(col2[i+48*j]/100==col2[i]+j
for j in range(1,7))
and ((col2[i]/100 -
first_monday) % 7 == 0
是
否
把 i 的位置(起始位置)
記錄下來
Positions.append(i)
i+=336(48*7)
i+=48
往下一天邁進
重新判斷
在資料裡面對應到 position[i]~position[i]+336 位置的數據複製到新的 list 裡面
複製完後 position[i++]直到跑完 position 裡面所有的起始位置
把 list 輸入到新的 txt 檔案

▲圖 4.2

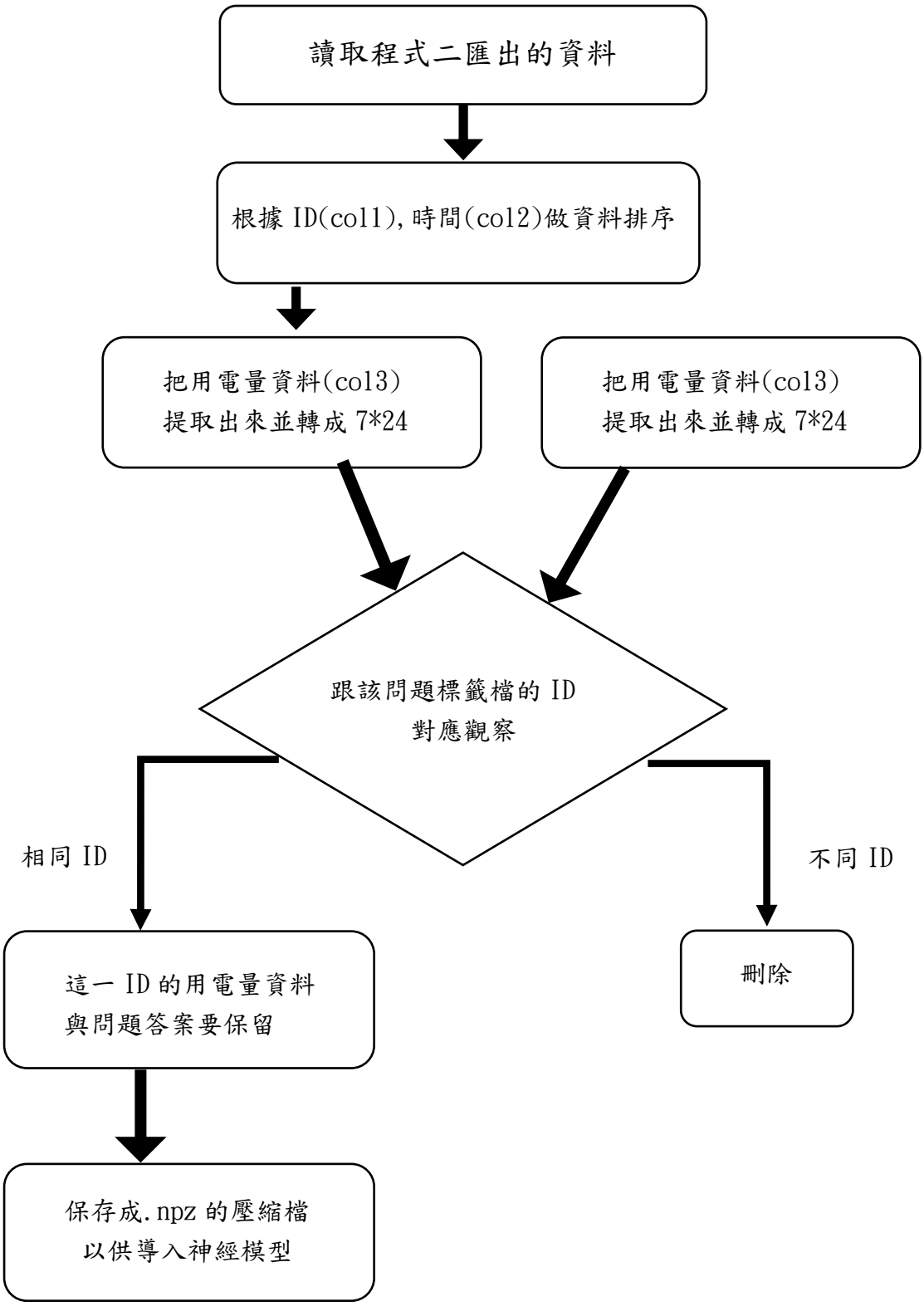
接著把一天 48 筆數據變成一天 24 筆數據，48 筆數據是因為每 30 分鐘紀錄一次，變成 24 筆數據就是 1 小時紀錄一次，所以我們每兩筆用電量數據相加起來存在 result 的 list 中，再利用 df=df[df[col2]%100<25]把 25~48 的數據給刪除(剩下的資料就是 01~24)，在把 result 儲存在 df 的 col3 中，這樣就可以把一天 48 筆數據變成一天 24 筆數據。

然後把這些數據轉成 7*24 的照片矩陣(也就是把資料裡的 col3 提取出來然後再轉成 7*24)，然後與標籤檔做對應，按照問題區分，如果 ID 在這個問題沒有答案，就把這 ID 刪除。最後將這些檔案保存成資料集，為了之後較好導入學習模型，我們把它存成.npz 檔。

.npz 檔是 NumPy 的壓縮檔案格式，用於存儲 NumPy 數組和相關數據，通常當數據需要在不同的 Python 程序之間共享或保存時，.npz 檔是一個常用的選擇。它可以存儲多個 NumPy 數組，每個數組都有一個名稱，可以通過名稱來存取和提取數據。

第三次處理是將資料 01~48 變成 01~24，並存成資料集。圖 4.3 為此處理的流程圖，大致分為 5 個步驟。

程式碼請見附錄<程式三>


讀取程式二匯出的資料
根據 ID(col1),時間(col2)做資料排序
把用電量資料(col3)
提取出來並轉成 7*24
把用電量資料(col3)
提取出來並轉成 7*24
跟該問題標籤檔的 ID
對應觀察
相同 ID
不同 ID
這一 ID 的用電量資料
與問題答案要保留
刪除
保存成.npz 的壓縮檔
以供導入神經模型

▲圖 4.3

第五章、標籤檔

5.1.引言

隨著人工智慧的迅速發展，深度神經網路在各個領域中扮演著越來越重要的角色，其中關鍵的一環就是標籤檔的使用。標籤檔能夠為神經網路提供關鍵的指導信息，並在訓練過程中扮演著至關重要的角色。

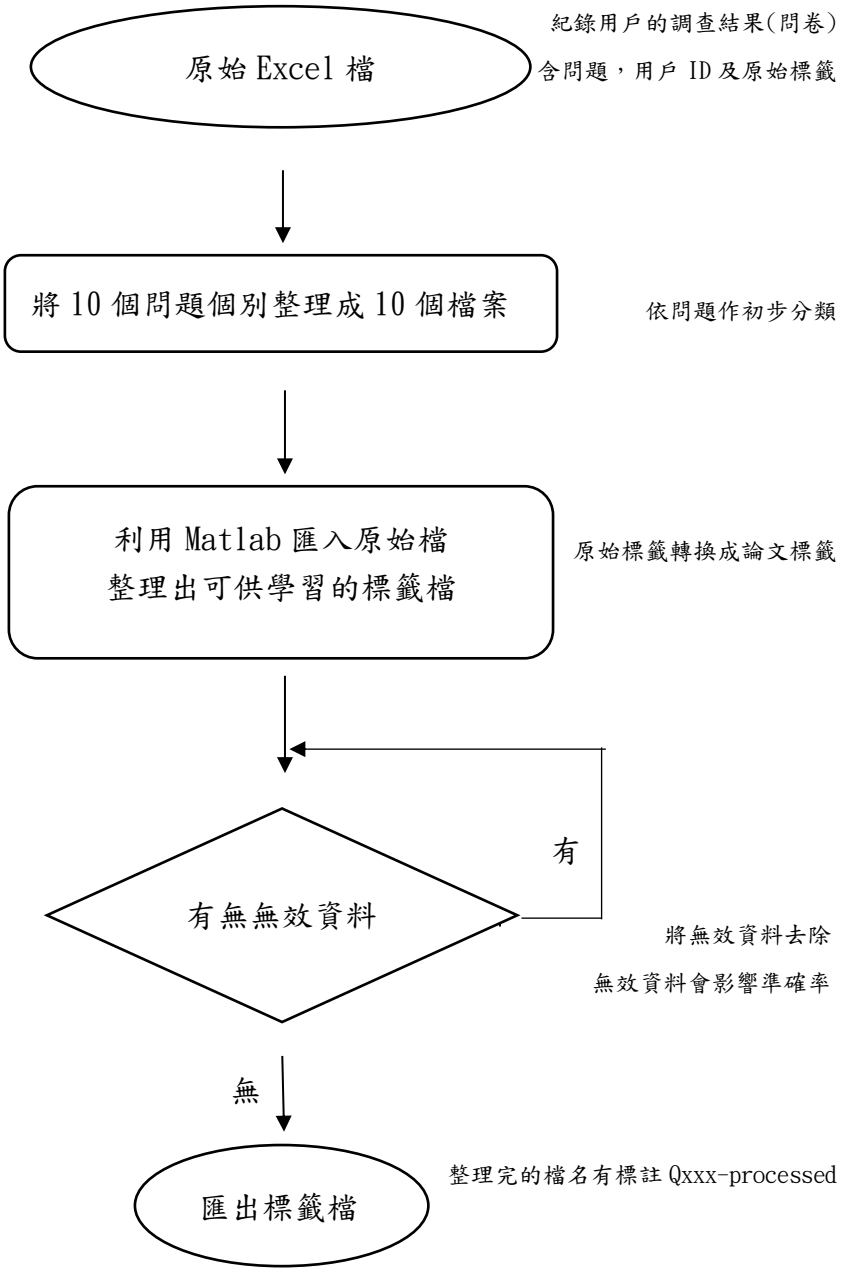
深度神經網路通常具有複雜的模型和大量的參數，這使得它們能有效進行非線性建模和自我學習。也正是這種複雜性，神經網路需要大量的標籤檔來指導其學習過程，以便準確地理解和分類數據。

此外，標籤檔對於深度神經網路的訓練和評估也至關重要。透過適當的標籤檔設計，確保神經網路在訓練過程中可以有效地收斂，避免過度擬合的問題。同時，在測試和評估階段，正確的標籤檔能夠提供可靠的標準，以評估神經網路的性能和準確度。

總的來說，標籤檔指導神經網路學習並提供關鍵信息，用來準確預測和分類數據，確保模型的有效訓練和準確評估。

5.2 標籤檔架構圖與預處理

架構圖



問題總表

SOCIO-DEMOGRAPHIC INFORMATION TO BE IDENTIFIED

No.	Question No.	Socio-demographic Information Question	Answers	Number
1	300	Age of chief income earner	Young(<35)	436
			Medium(35∼65)	2819
			Old(>65)	953
2	310	Chief income earner has retired or not	Yes	1285
			No	2947
3	401	Social class of chief income earner	A or B	642
			C1 or C2	1840
			D or E	1593
4	410	Have children or not	Yes	1229
			No	3003
5	450	House type	Detached or bungalow	2189
			Semi-detached or terraced	1964
6	453	Age of the house	Old(>30)	2151
			New(<30)	2077
7	460	Number of bedrooms	Very low(<3)	404
			Low(=3)	1884
			High(=4)	1470
			Very High(>4)	474
8	4704	Cooking facility type	Electrical	1272
			Not Electrical	2960
9	4905	Energy-efficient light bulb proportion	Up to half	2041
			Three quarters or more	2191
10	6103	Floor area	Small(<100)	232
			Medium(>100&<200)	1198
			Big(>200)	351

問題一（Q300）

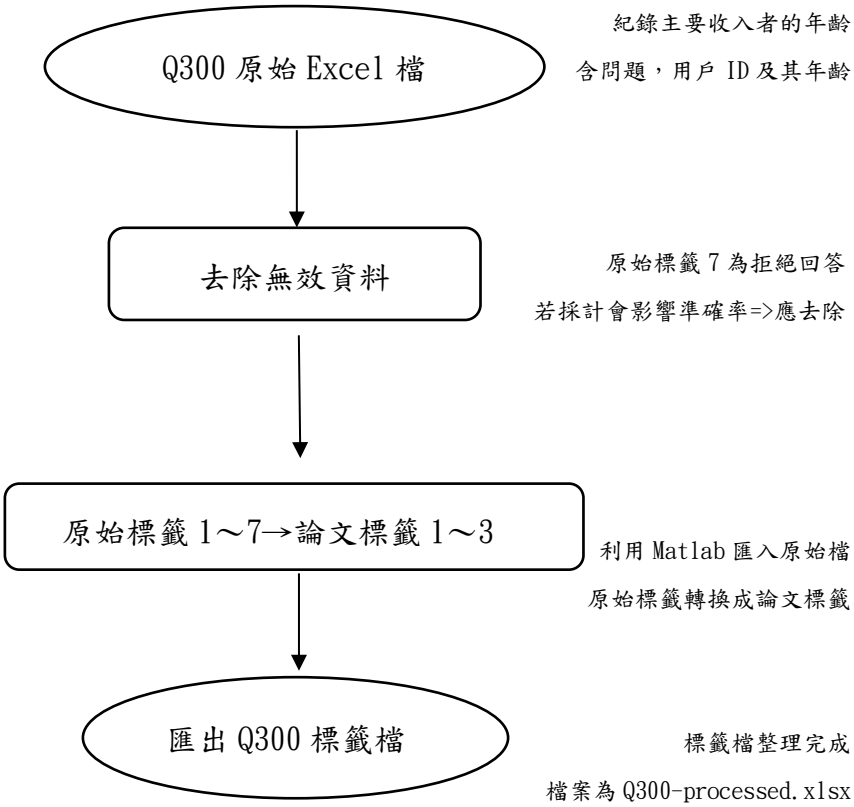
本問題探討用戶家中主要收入者的年齡。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果可以得到依照年齡層分類的 7 個原始標籤，本次處理就是要依照論文的分類條件轉換，由原始的 7 個標籤先將拒絕回答者去除後再轉換成論文的 3 個標籤。

表 5.1 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題一(Q300)	論文標籤條件	人數
Age of chief income earner（主要收入者的年齡）	Young(<35)	436
	Medium(35～65)	2819
	Old(>65)	953

▲表 5.1

圖 5.2 為 Q300 的標籤檔處理流程圖，大致分為四步驟處理。



▲圖 5.2

表 5.3 為 Q300 原始資料統整表，第一欄為原始資料的原始標籤(共 7 個標籤)，第二欄為原始分類的年齡層，第三欄為統計的人數。

原始標籤	分類條件(歲)	人數
1	18-25	16
2	26-35	420
3	36-45	905
4	46-55	1031
5	56-65	883
6	>65	953
7	拒絕回答	24
	總計	4232

▲表 5.3

由表 5.3 能發現主要收入者年齡介在 18-25 歲者，原始標籤為 1，共 16 人；年齡介在 26-35 歲者，原始標籤為 2，共 420 人；年齡介在 36-45 歲者，原始標籤為 3，共 905 人；年齡介在 46-55 歲者，原始標籤為 4，共 1031 人；年齡介在 56-65 歲者，原始標籤為 5，共 883 人；年齡超過 65 歲者，原始標籤為 6，共 953 人。其中拒絕回答者共 24 人。

經過 Matlab 處理將拒絕回答去除，並將原始標籤轉換成論文標籤(詳細程式碼請見附錄)

表 5.4 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 3 類)，第二欄為依論文條件轉換後而成的新分類(共 3 個標籤)，第三欄為統計的人數。

論文分類	論文標籤	有效人數
Young(<35)	1	436
Medium(36-65)	2	2819
Old(>65)	3	953
	總計	4208

▲表 5.4

由表 5.4 可知分類 Young（<35 歲），就是原始標籤 1&2，即論文標籤 1，共 436 人；分類 Medium（介於 35 至 65 歲），就是原始標籤 3&4&5，即論文標籤 2，共 2820 人；分類 Old（>65 歲），就是原始標籤 6，即論文標籤 3，共 953 人。

經過 Excel 統計總數 4232-24=4208，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，所以問題一(Q300)標籤檔整理完成。

問題二（Q310）

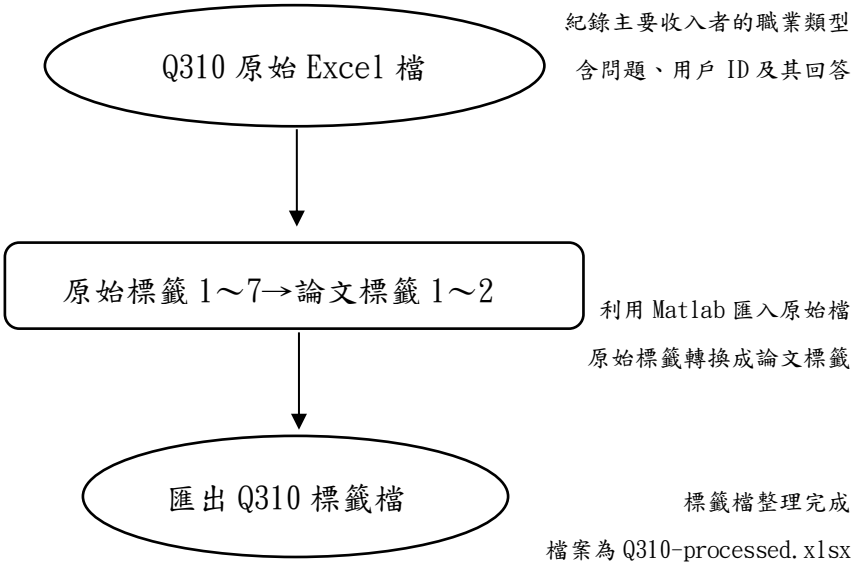
本問題探討用戶家中主要收入者是否退休。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果可以得到依照就業情形分類出的 7 個原始標籤，本次處理就是要依照論文的分類條件轉換，由原始的 7 個標籤轉換成論文的 2 個標籤。

表 5.5 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題二(Q310)	論文標籤條件	人數
Chief income earner has retired or not （主要收入者退休與否）	YES	1285
	NO	2947

▲表 5.5

圖 5.6 為 Q310 的標籤檔處理流程圖，大致分為三步驟處理。



▲圖 5.6

表 5.7 為 Q300 原始資料統整表，第一欄為原始資料的原始標籤(共 7 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件	人數
1	An employee	2001
2	Self-employed (with employees)	232
3	Self-employed (with no employees)	303
4	Unemployed (actively seeking work)	200
5	Unemployed (not actively seeking work)	170
6	Retired	1285
7	Carer: Looking after relative family	41
	總計	4232

▲表 5.7

由表 5.7 能發現根據主要收入者職業類型分類後，在原始標籤 1，歸類為未退休，共 2001 人；原始標籤 2，歸類為未退休，共 232 人；原始標籤 3，歸類為未退休，共 303 人；原始標籤 4，歸類為未退休，共 200 人；原始標籤 5，歸類為未退休，共 170 人；原始標籤 6，歸類為已退休，共 1285 人；原始標籤 7，歸類為未退休，共 41 人。

經過 Matlab 處理將原始標籤轉換成論文標籤(詳細程式碼請見附錄)

表 5.8 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 2 類)，第二欄為依論文條件轉換後而成的新分類(共 2 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Yes	1	1285
No	2	2947
	總計	4232

▲表 5.8

由表 5.8 可知分類為 Yes(意即 retired)，就是原始標籤除 6 之外的全部，即論文標籤 1，共 1285 人；No(意即 no retired)，就是原始標籤 6，即論文標籤 2，共 2947 人。

經過 Excel 統計確認總數為 4232，為正確結果。因此能確定以上人數皆與論文所紀錄的人數相符，所以問題二(Q310)標籤檔整理完成。

問題三（Q401）

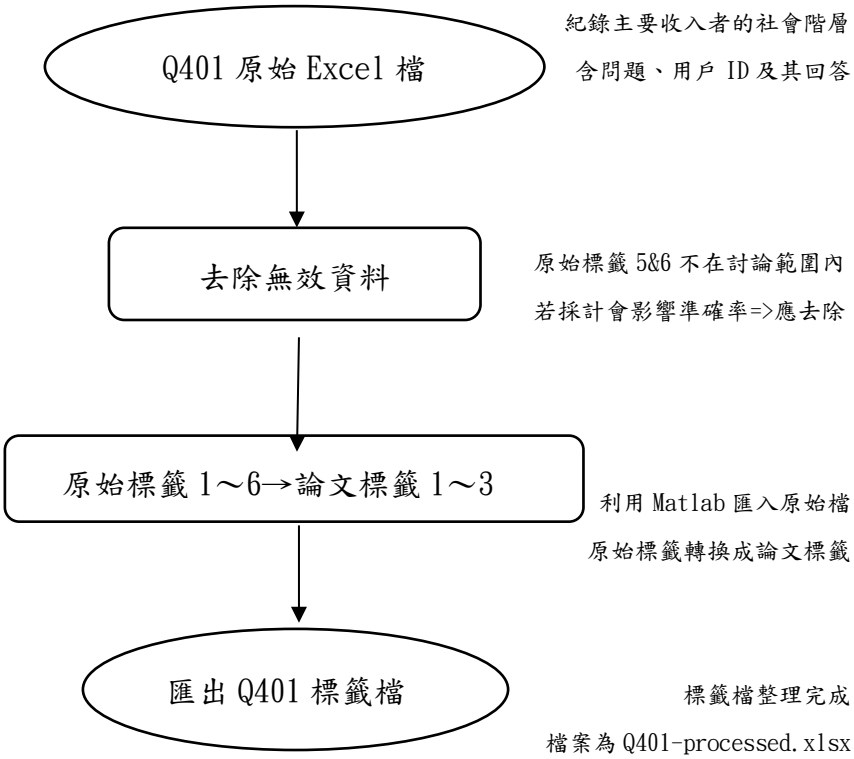
本問題探討用戶家中主要收入者的社會階層。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，其中在調查前先將社會階層以 A 到 F 做分類，並把 C 分類拆成 C1 及 C2，另外將 A 和 B 及 D 和 E 歸為一類，因此可以分類出了 6 個原始標籤，本次處理就是先將不再討論範圍的資料去除後，再依照論文的分類條件轉換，由原始的 7 個標籤轉換成論文的 3 個標籤。

表 5.9 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題三(Q401)	論文標籤條件	人數
Social class of chief income earner（主要收入者的社會階層）	A or B	642
	C1 or C2	1840
	D or E	1593

▲表 5.9

圖 5.10 為 Q401 的標籤檔處理流程圖，大致分為四步驟處理。



▲圖 5.10

表 5.11 為 Q401 原始資料統整表，第一欄為原始資料的原始標籤(共 7 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件	人數
1	A+B	642
2	C1	1134
3	C2	706
4	D+E	1593
5	F	113
6	拒絕回答	44
	總計	4232

▲表 5.11

再來能從表 5.11 根據主要收入者的社會階層分類，首先將 A 和 B 歸為一類，將 C 分類拆成 C1 類及 C2 類，再將 D 和 E 歸為一類。之後再統計 AB 類，即原始標籤 1，共 642 人；C1 類，即原始標籤 2，共 1134 人；C2 類，即原始標籤 3，共 706 人；DE 類，即原始標籤 4，共 1593 人；F 類，即原始標籤 5，共 113 人；拒絕回答者，即原始標籤 6，共 44 人。

經過 Matlab 處理將 F(原始標籤 5)及拒絕回答去除，再將原始標籤轉換成論文標籤(詳細程式碼請見附錄)

表 5.12 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 3 類)，第二欄為依論文條件轉換後而成的新分類(共 3 個標籤)，第三欄為統計的人數。

	論文標籤	人數
A+B	1	642
C1+C2	2	1840
D+E	3	1593
	總計	4075

▲表 5.12

由表 5.12 可知 AB 類，就是原始標籤 1，即論文標籤 1，共 642 人；C1+C2 類，就是原始標籤 2&3，即論文標籤 2，共 1840 人；DE 類，就是原始標籤 4，即論文標籤 3，共 1593 人。

經過 Excel 統計總數為 4232-113-44=4075，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，所以問題三(Q401)標籤檔整理完成。

問題四（Q410）

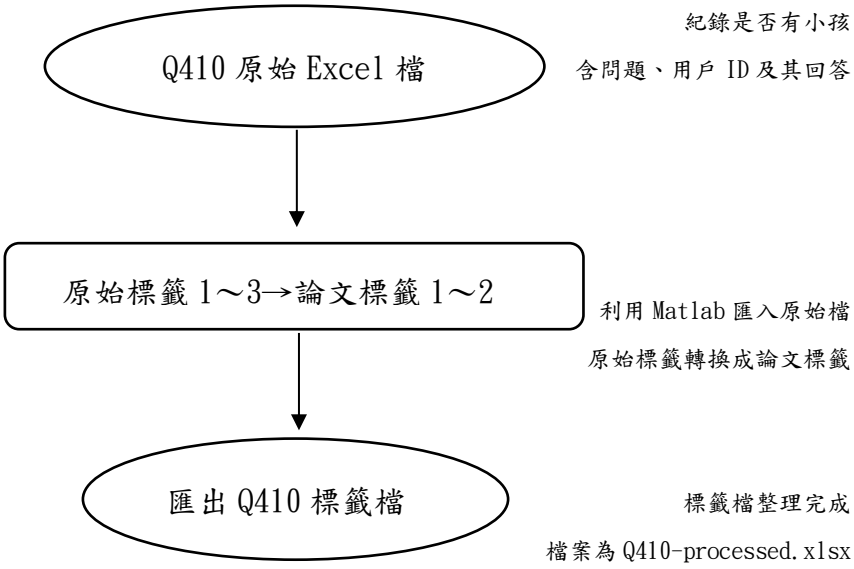
本問題探討用戶家中是否有小孩。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，依家庭成員的描述分類出了 3 個原始標籤，本次處理就是要依照論文的分類條件轉換，由原始的 3 個標籤轉換成論文的 2 個標籤。

表 5.13 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題四(Q410)	論文標籤條件	人數
Have children or not （有無小孩）	Yes	1229
	No	3003

▲表 5.13

圖 5.14 為 Q401 的標籤檔處理流程圖，大致分為四步驟處理。



▲圖 5.14

表 5.15 為 Q410 原始資料統整表，第一欄為原始資料的原始標籤(共 3 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件	人數
1	I live alone	642
2	All people in my home are over 15 years of age	1134
3	Both adults and children under 15 years of age live in my home	706
	總計	4232

▲表 5.15

由表 5.15 發現家庭成員的現況，可分為獨居，即原始標籤 1，共 808 人；所有家庭成員皆大於 15 歲，即原始標籤 2，共 2195 人；家中有小於 15 歲的成員，即原始標籤 3，共 1229 人。

經過 Matlab 處理將原始回答轉換成標籤後(詳細程式碼請見附錄)

表 5.16 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 3 類)，第二欄為依論文條件轉換後而成的新分類(共 3 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Yes	1	1229
No	2	3003
	總計	4232

▲表 5.16

由表 5.16 可知分類在 Yes(意即有小孩)，即原始標籤 3，即論文標籤 1，共 3003 人；No(意即沒有小孩)，即原始標籤 1&2，即論文標籤 2，共 1229 人。

經過 Excel 統計總數為 4232，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，所以問題四(Q410)標籤檔整理完成。

問題五（Q450）

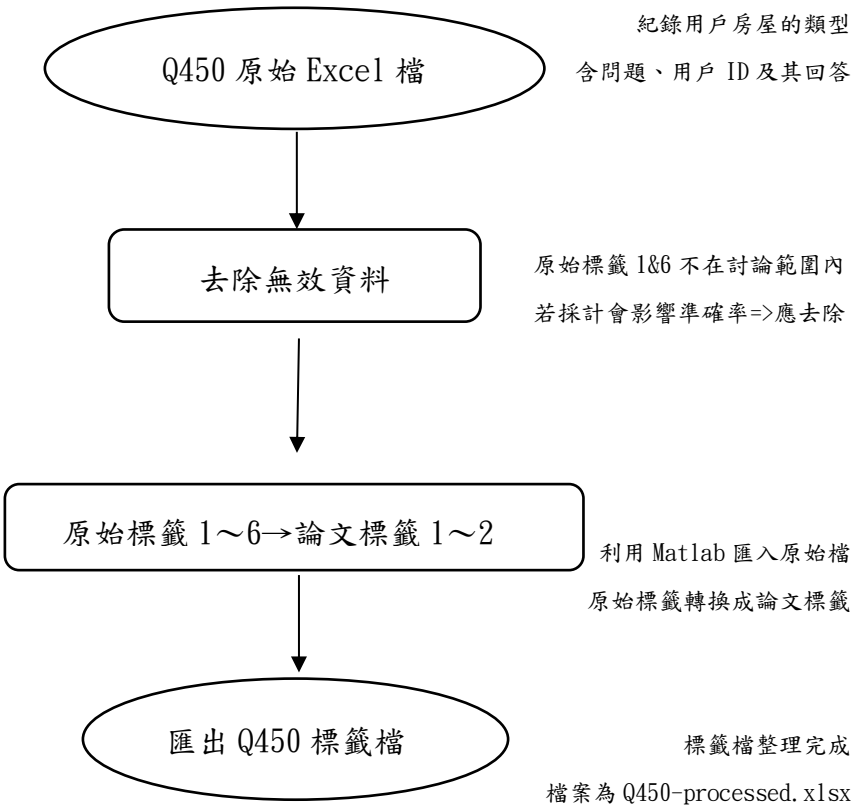
本問題探討用戶房屋的類型。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，依房屋類型分類出了 6 個原始標籤，本次處理就是先將不再討論範圍的資料及拒絕回答去除後，再依照論文的分類條件轉換，由原始的 6 個標籤轉換成論文的 2 個標籤。

表 5.17 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題五(Q450)	論文標籤條件	人數
House type （房屋類型）	Detached or bungalow	2189
	Semi-detached or terraced	1964

▲表 5.17

圖 5.18 為 Q450 的標籤檔處理流程圖，大致分為四步驟處理。



▲圖 5.18

表 5.19 為 Q450 原始資料統整表，第一欄為原始資料的原始標籤(共 6 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件	人數
1	公寓	72
2	半獨立式	1351
3	獨立式	1121
4	連排	613
5	平房	1068
6	拒絕回答	7
	總計	4232

▲表 5.19

由 5.19 能看出將用戶的房屋類型做了分類，其中分為公寓，即原始標籤 1，共 72 人；半獨立式，即原始標籤 2，共 1351 人；獨立式，即原始標籤 3，共 1351 人；連排，即原始標籤 4，共 613 人；平房，即原始標籤 5，共 1068 人；拒絕回答者，即原始標籤 6，共 7 人。

經過 Matlab 處理先將不再討論範圍內的資料去除(原始標籤 1 及原始標籤 6 不計)，再將原始回答轉換成標籤（詳細程式碼請見附錄）

表 5.20 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 2 類)，第二欄為依論文條件轉換後而成的新分類(共 2 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Detached or bungalow	1	2189
Semi-detached or terraced	2	1964
	總計	4153

▲表 5.20

由表 5.20 可知將房屋類型大致分成獨立式或簡易別墅(Detached or bungalow)，即論文標籤 1，共 2189 人；半獨立式或梯田式(Semi-detached or terraced)，即論文標籤 2，共 1964 人。

經過 Excel 統計總數為 4232-72-7=4153，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，所以問題五(Q450)標籤檔整理完成。

問題六（Q453）

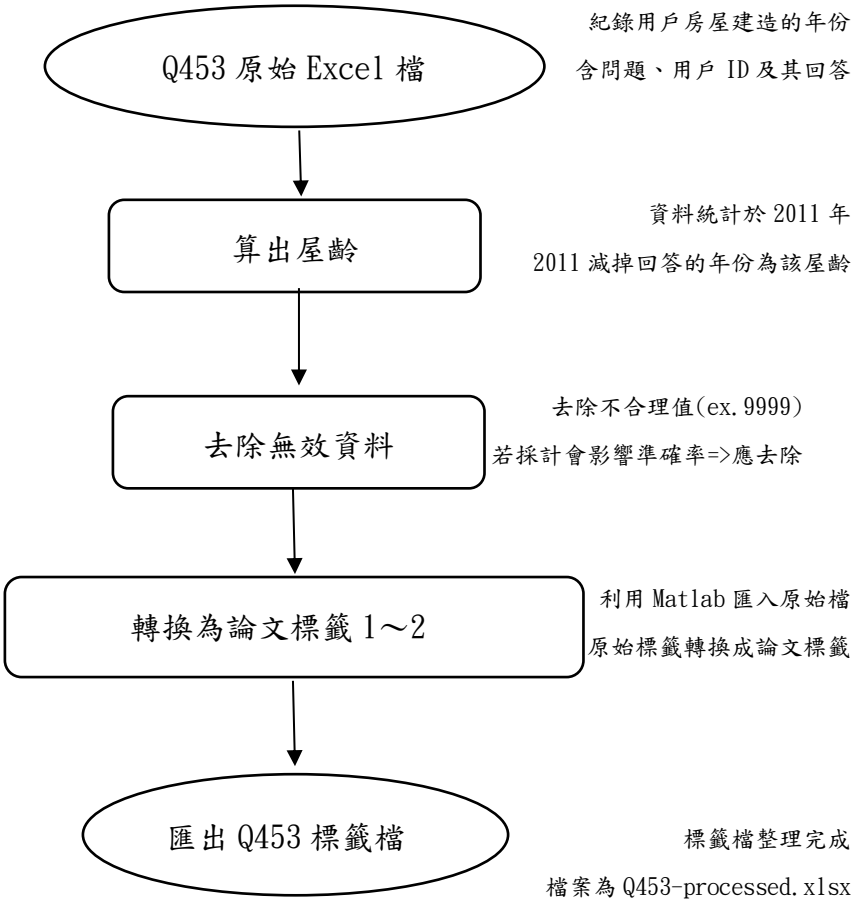
本問題探討用戶屋齡。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，調查中是直接填寫房子在哪年(西元年份)所建造，由於資料統計於 2011 年，因此將由 2011 減掉回答的年份為該屋齡。因為無調查統計結果，直接依論文條件進行轉換處理。

表 5.21 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題六(Q453)	論文標籤條件	人數
Age of the house（屋齡）	Old（>30）	2151
	New（<30）	2077

▲表 5.21

圖 5.22 為 Q453 的標籤檔處理流程圖，大致分為五步驟處理。



▲圖 5.22

因為輸入資料為數值(房屋建造的年份)，沒有原始標籤分類的統計表，所以統計完數量則直接進行轉換處理。經過統計後可分為屋齡大於 30 年，記為 Old，共 2151 人；屋齡小於 30 年，記為 New，共 2077 人。

先經過 Matlab 將原始檔內的無效資料去除，像是有小於 1000 的值(ex.75)，共 4 人；大於 2011(ex.9999)的回答，共 422 人，這些都屬於不合理的值。再將 2011 減去原始回答(年份)，就可得知屋齡，之後進行轉換成標籤後(詳細程式碼請見附錄)

表 5.23 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 2 類)，第二欄為依論文條件轉換後而成的新分類(共 2 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Old (>30)	1	1784
New (<30)	2	2022
	總計	3806

▲表 5.23

由表 5.23 可知轉換後分類 Old (>30)，即論文標籤 1，共 1784 人；New (<30)，即論文標籤 2，共 2022 人。

經過 Excel 統計總數為 4232-422-4=3806，相比論文之總數 4228(4232-4)，我們猜測論文是只將小於 1000 的回答去除，但 9999 為年份更不合理，因此總數 3806 更為正確。所以問題六(Q453)標籤檔整理完成。

問題七（Q460）

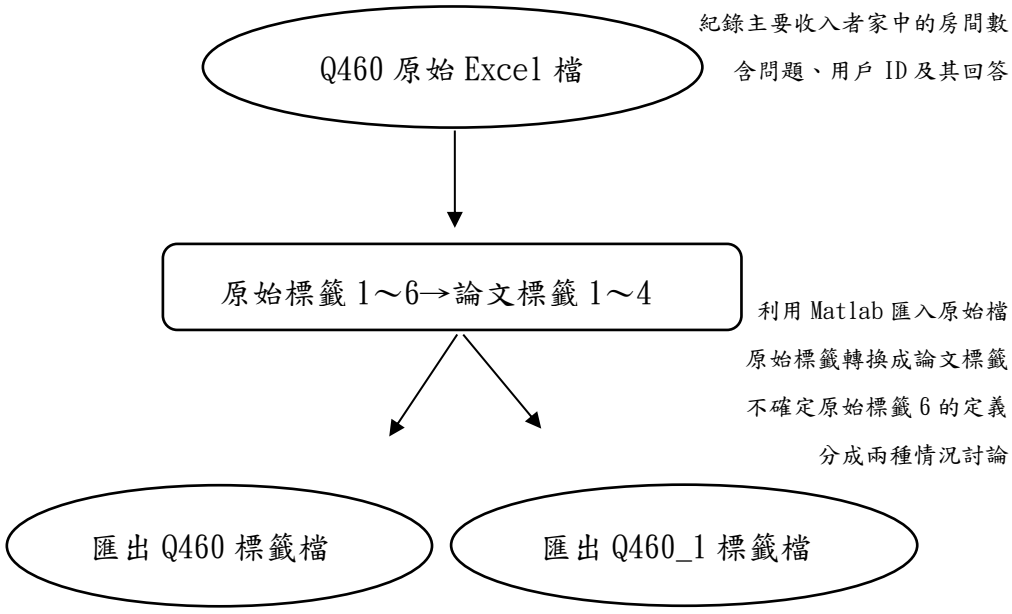
本問題探討用戶房屋的類型。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，依房間數分類出了 6 個原始標籤，本次處理就是依照論文的分類條件轉換，由原始的 6 個標籤轉換成論文的 4 個標籤。

表 5.24 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數。

問題七(Q460)	論文標籤條件	人數
Number of bedrooms（房間數）	Very low (<3)	450
	Low (=3)	1884
	High (=4)	1470
	Very high (>4)	474

▲表 5.24

圖 5.25 為 Q460 的標籤檔處理流程圖，大致分為三步驟處理。



▲圖 5.25

表 5.26 為 Q460 原始資料統整表，第一欄為原始資料的原始標籤(共 6 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件(間)	人數
1	1	46
2	2	404
3	3	1884
4	4	1470
5	5	465
6	6 or 拒絕回答	9
	總計	4232

▲表 5.26

表 5.26 能發現家中房屋數為 1 間，即原始標籤 1，共 46 人；房屋數為 2 間，即原始標籤 2，共 358 人；房屋數為 3 間，即原始標籤 3，共 1884 人；房屋數為 4 間，即原始標籤 4，共 1470 人；房屋數為 5 間，即原始標籤 5，共 465 人；原始標籤 6 共 9 人，但不確定是代表房屋數 6 間還是拒絕回答，因此分成兩種情況討論。

經過 Matlab 處理將原始回答轉換成標籤後(詳細程式碼請見附錄，本題有兩個)
表 5.27 為標籤檔整理後統整表(6 為 6 間房間的情況)
表 5.28 為標籤檔整理後統整表(6 為拒絕回答的情況)
第一欄為論文上的分類條件(分 4 類)，第二欄為依論文條件轉換後而成的新分類(共 4 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Very low (<3)	1	450
Low (=3)	2	1884
High (=4)	3	1470
Very high (>4)	4	474
	總計	4232

▲表 5.27

	論文標籤	人數
Very low (<3)	1	450
Low (=3)	2	1884
High (=4)	3	1470
Very high (>4)	4	465
	總計	4223

▲表 5.28

由表 5.27 可知家中房間數 Very Low(小於 3 間)，即論文標籤 1，共 404 人；家中房間數 Low(剛好為 3 間)，即論文標籤 2，共 1884 人；家中房間數 High(剛好為 4 間)，即論文標籤 3，共 1470 人；在原始標籤 6 為房間數 6 間的情況，家中房間數 Very High(大於 4 間)，即論文標籤 4，共 465 人；在原始標籤 6 為拒絕回答的情況下，家中房間數 Very High(大於 4 間)則會減少變成共 465 人。

經過 Excel 統計總數，若 6 表示 6 間房間時，總數為 4232，若 6 表示拒絕回答時，總數為 4232-9=4223，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，所以問題七(Q460)標籤檔整理完成。

問題八（Q4704）

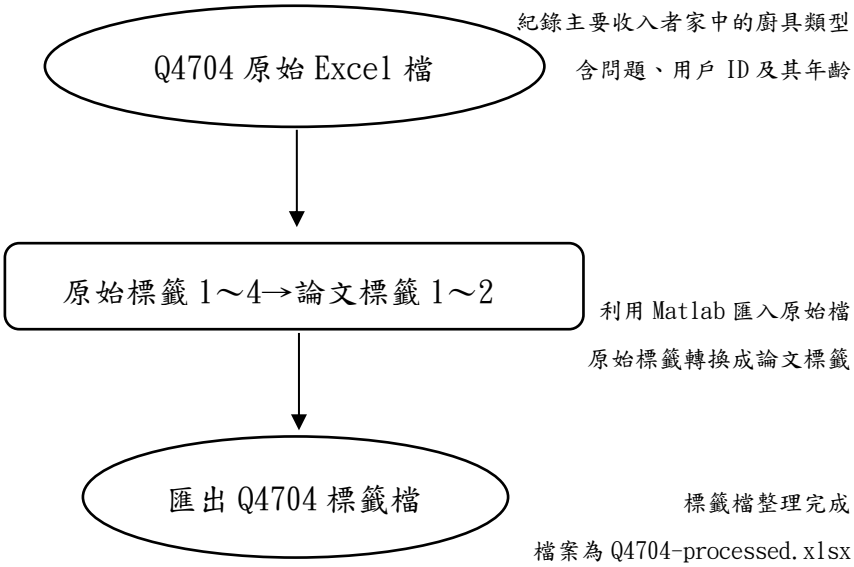
本問題探討用戶家中的廚具類型。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，依廚具的類型分類出了 4 個原始標籤，本次處理就是依照論文的分類條件轉換，由原始的 4 個標籤轉換成論文的 2 個標籤。

表 5.29 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數，但記載的人數兩個相反(下面有推論)。

問題八(Q4704)	論文標籤條件	人數
Cooking facility type（廚具類型）	Electrical	1272
	Not electrical	2960

▲表 5.29

圖 5.30 為 Q4704 的標籤檔處理流程圖，大致分為三步驟處理。



▲圖 5.30

表 5.31 為 Q4704 原始資料統整表，第一欄為原始資料的原始標籤(共 4 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件	人數
1	Electric cooker	2960
2	Gas cooker	1086
3	Oil fired cooker	97
4	Solid fuel cooker	89
	總計	4232

▲表 5.31

由表 5.31 得知將廚具分為電器廚具，即原始標籤 1，共 2960 人；燃氣廚具，即原始標籤 2，共 1086 人；燃油廚具，即原始標籤 3，共 97 人；爐子，即原始標籤 4，共 89 人。

經過 Matlab 處理將原始回答轉換成標籤後(詳細程式碼請見附錄)
表 5.32 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 2 類)，第二欄為依論文條件轉換後而成的新分類(共 2 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Electrical	1	2960
Not electrical	2	1272
	總計	4232

▲表 5.32

由表 5.32 可知以廚具是否為電器做區分，其中電器廚具，即論文標籤 1，共 2960 人；非電器廚具，即論文標籤 2，共 1272 人。

經過 Excel 統計總數 4232，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，但順序顛倒，推測是論文有誤，所以問題八(Q4704)標籤檔整理完成。

問題九（Q4905）

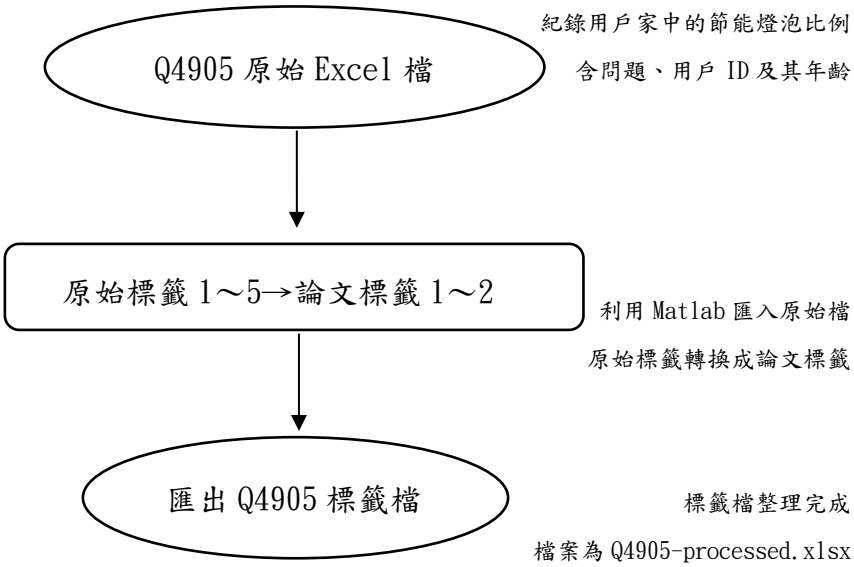
本問題探討節能燈泡在用戶家中的比例。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，依節能燈泡在家中的比例分類出了 5 個原始標籤，本次處理就是依照論文的分類條件轉換，由原始的 5 個標籤轉換成論文的 2 個標籤。

表 5.33 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數，但比例有點出入(下面會有推論)。

問題九(Q4905)	論文標籤條件	人數
Energy-efficient light bulb proportion （節能燈泡的比例）	Up to half	1272
	Three quarters or more	2960

▲表 5.33

圖 5.34 為 Q4905 的標籤檔處理流程圖，大致分為三步驟處理。



▲圖 5.34

表 5.35 為 Q4905 原始資料統整表，第一欄為原始資料的原始標籤(共 5 個標籤)，第二欄為原始分類的就業類別，第三欄為統計的人數。

原始標籤	分類條件	人數
1	None	937
2	About a quarter	1104
3	About half	705
4	About three quarters	697
5	All	789
	總計	4232

▲表 5.35

由表 5.35 能發現用戶家中傑能燈泡的比例為 0%，即原始標籤 1，共 937 人；25%，即原始標籤 2，共 1104 人；50%，即原始標籤 3，共 705 人；75%，即原始標籤 4，共 697 人；100%，即原始標籤 5，共 789 人。

若根據論文中 three quarters or more，則人數不相符，而將條件改為 half or more 則人數相符(推測是論文條件有誤)。

經過 Matlab 處理將原始回答轉換成標籤後(詳細程式碼請見附錄)
表 5.36 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 2 類)，第二欄為依論文條件轉換後而成的新分類(共 2 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Up to half	1	2041
Half or more	2	2191
	總計	4232

▲表 5.36

由表 5.36 可知節能燈泡比例為最多一半(up to half)，即論文標籤 1，共 2041 人；50%(含)以上(half or more)，即論文標籤 2，共 2191 人。

經過 Excel 統計總數為 4232，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，但論文條件敘述有誤，所以問題九(Q4905)標籤檔整理完成。

問題十（Q6103）

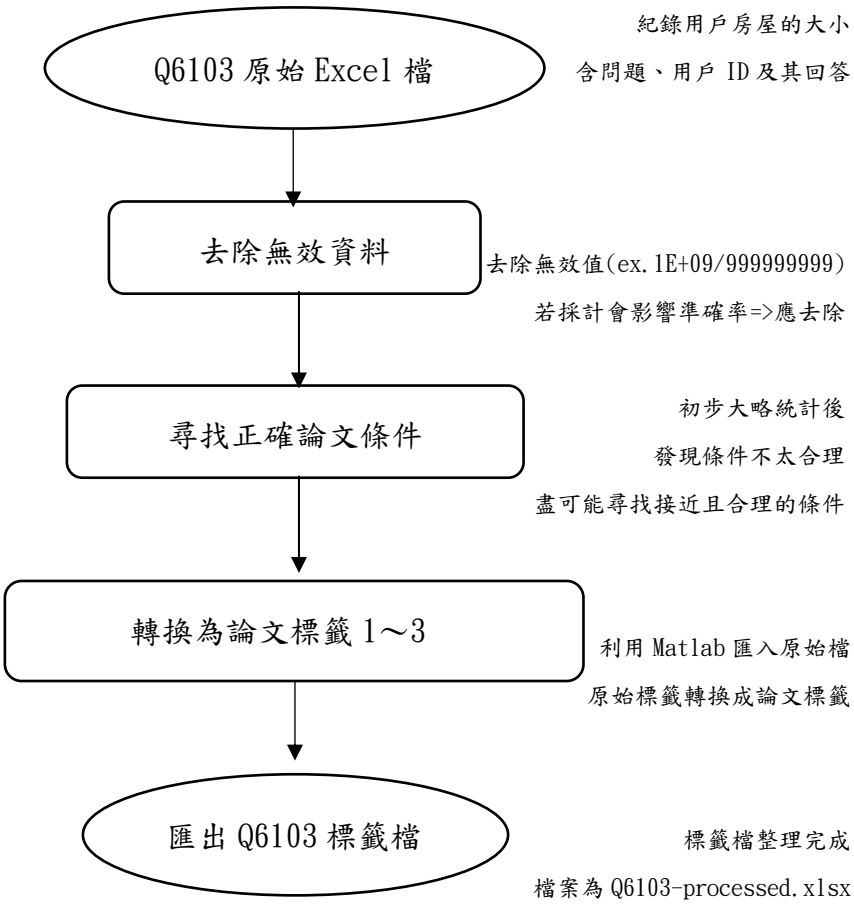
本問題探討用戶房子的大小。根據 ISSDA 愛爾蘭數據庫申請到的用戶調查結果，調查中是直接填寫房子的大小，因為無調查統計結果，直接依論文條件進行轉換處理。

表 5.37 為本問題的統計表格，第一欄為調查的問題，第二欄為論文上的條件，第三欄則是對應到的人數，但論文條件有誤。

問題十(Q4704)	論文標籤條件	人數
Floor area	Small (<100)	232
(房子的大小)	Medium (>100&<200)	1198
	Big (>200)	351

▲表 5.37

圖 5.38 為 Q6103 的標籤檔處理流程圖，大致分為三步驟處理。



▲圖 5.38

此調查為直接回答大小數值，因此沒有歸類成原始標籤。然而在處理前發現原始檔中有許多錯誤值(1E+09/999999999)，共 2451 筆，所以先用 Matlab 將其去除，總數就變成 4232-2451=1781。
表 5.39 為統計錯誤值統整表，第一欄為統計條件，第二欄為其人數。

	人數
統計(正確/有效)	1781
錯誤值(1E+09/999999999)	2451
統計(正確+錯誤)	4232

▲表 5.39

再來發現若照著論文上的條件分類會有極大落差，因此合理懷疑是分類條件有誤，所以需經過反覆嘗試，找出較合理且總數能符合的條件。

表 5.40 為 Small 推測條件統計表，第一欄為的分類條件，第二欄為人數。
表 5.41 為 Medium 推測條件統計表，第一欄為的分類條件，第二欄為人數。

統計 Small	
條件	人數
<1000	191
>1000	280

▲表 5.40

統計 Medium	
條件	人數
>1000&<=2000	1128
>1000&<=2500	1336
>1000&<=2200	1202

▲表 5.41

表 5.42 為找尋較合理的分類條件

	條件(不合理)	人數(不合理)	條件(合理)	人數(合理)
Small	<=100	28	<=1000	280
Medium	>100&<=200	47	>1000&<=2200	1202
Big	>200	1713	>2200	299
	總人數	1788	總人數	1781

▲表 5.42

由表 5.42 可推得 Big 的條件為>2200 較合理。這樣就可以對照論文的資料總數 232+1198+351=1781 和推斷的條件下所得的總數 280+1202+299=1781，發現總數相符。

表 5.43 為標籤檔整理後統整表，第一欄為論文上的分類條件(分 3 類)，第二欄為依論文條件轉換後而成的新分類(共 3 個標籤)，第三欄為統計的人數。

	論文標籤	人數
Small	1	280
Medium	2	1202
Big	3	299
	總計	1781

▲表 5.43

由表 5.43 可知 Small(<=1000)，即論文標籤 1，共 280 人；
Medium(<1000&>=2200)，即論文標籤 2，共 1202 人；Big(>2200)，即論文標籤 3，共 299 人。

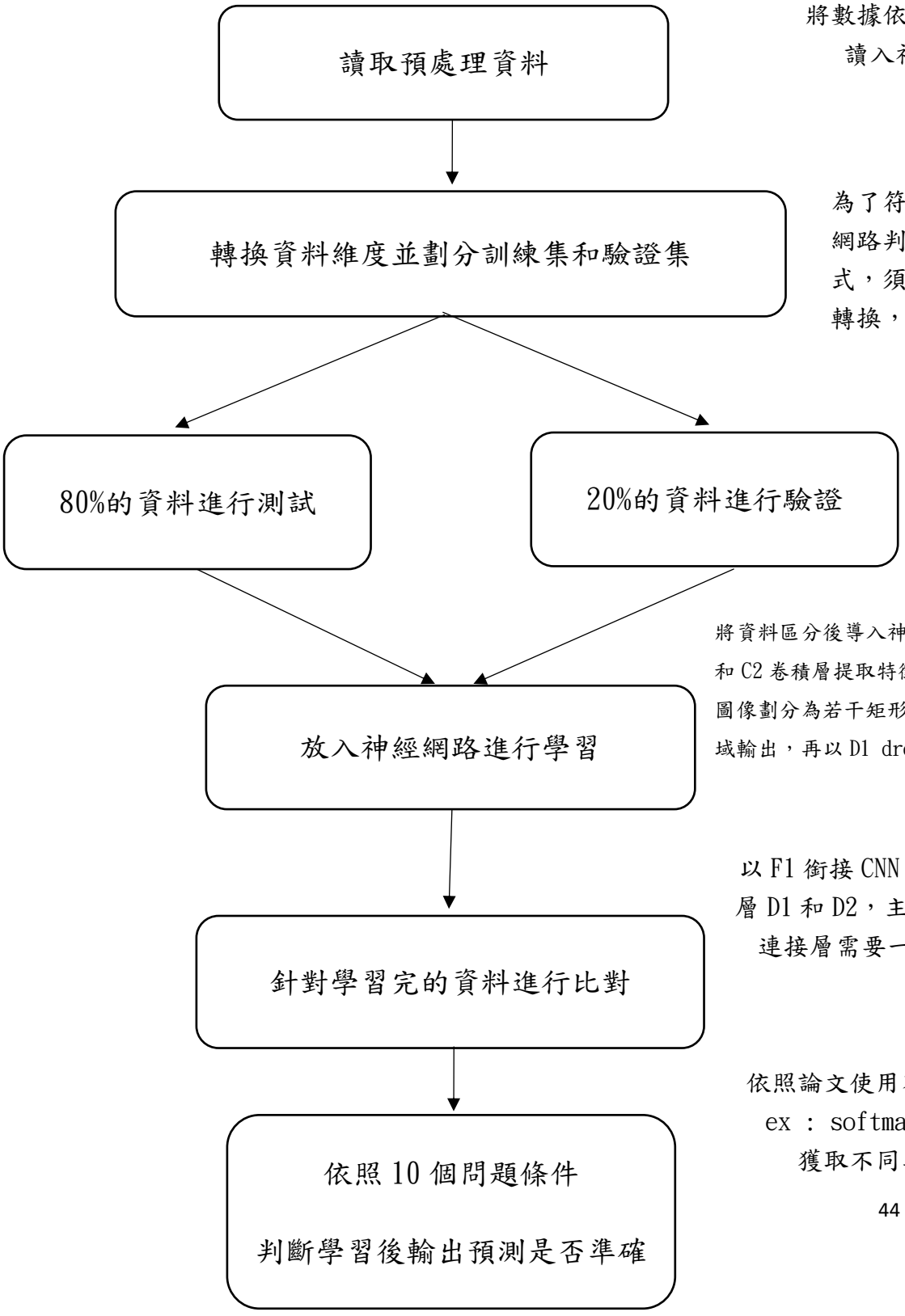
經過 Excel 統計總數 1781，為正確結果，因此能確定以上人數皆與論文所紀錄的人數相符，但論文條件敘述有誤，所以問題十(Q6103)標籤檔整理完成。

第六章、分析與討論

6.1 神經網路流程圖

整理完有效的用電量和標籤檔後，將數據導入架設好的神經網路，並依照論文中提供的神經網路架設格式進行。

本節提出了其他方法進行了比較測試



6.2 效能評估與比較

<參考論文中問題及人數總表>

SOCIO-DEMOGRAPHIC INFORMATION TO BE IDENTIFIED

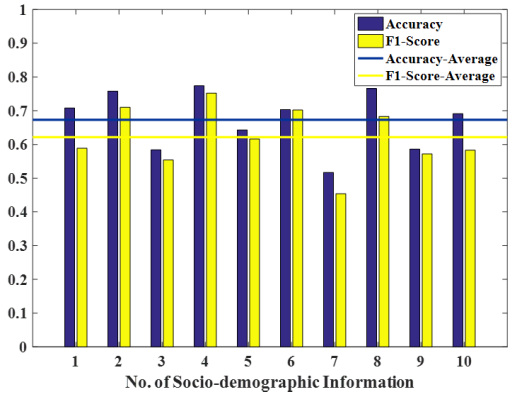
No.	Question No.	Socio-demographic Information Question	Answers	Number
1	300	Age of chief income earner	Young(<35)	436
			Medium(35∼65)	2819
			Old(>65)	953
2	310	Chief income earner has retired or not	Yes	1285
			No	2947
3	401	Social class of chief income earner	A or B	642
			C1 or C2	1840
			D or E	1593
4	410	Have children or not	Yes	1229
			No	3003
5	450	House type	Detached or bungalow	2189
			Semi-detached or terraced	1964
6	453	Age of the house	Old(>30)	2151
			New(<30)	2077
7	460	Number of bedrooms	Very low(<3)	404
			Low(=3)	1884
			High(=4)	1470
			Very High(>4)	474
8	4704	Cooking facility type	Electrical	1272
			Not Electrical	2960
9	4905	Energy-efficient light bulb proportion	Up to half	2041
			Three quarters or more	2191
10	6103	Floor area	Small(<100)	232
			Medium(>100&<200)	1198
			Big(>200)	351

<參考論文中不同分類器下的準確率總表>

TABLE IV
ACCURACIES OF DIFFERENT METHODS

	BG	MF	SVM	LS	PS	SS	CS	Proposed	Improvement 1	Improvement 2	Improvement 3
1	0.511	0.59	0.648	0.664	0.667	0.666	0.688	**0.708**	3.15%	**2.95%**	29.47%
2	0.577	0.73	0.697	0.702	0.696	0.693	0.748	**0.758**	2.47%	**1.37%**	24.39%
3	0.382	0.53	0.506	0.498	0.513	0.5	0.55	**0.584**	3.77%	**6.09%**	37.66%
4	0.588	0.73	0.709	0.715	0.73	0.714	0.748	**0.774**	2.47%	**3.45%**	24.39%
5	0.501	0.59	0.572	0.567	0.567	0.531	0.621	**0.643**	5.25%	**3.53%**	16.65%
6	0.5	0.64	0.564	0.577	0.582	0.566	0.665	**0.703**	3.91%	**5.69%**	22.77%
7	0.416	0.39	0.472	0.476	0.49	0.467	0.501	**0.517**	2.24%	**3.17%**	13.77%
8	0.579	0.71	0.687	0.675	0.694	0.698	0.739	**0.766**	4.08%	**3.70%**	22.60%
9	0.501	0.55	0.511	0.492	0.528	0.525	0.565	**0.586**	2.73%	**3.70%**	7.13%
10	0.508	0.5	0.643	0.649	0.644	0.639	0.658	**0.691**	1.39%	**4.96%**	24.41%

<參考論文中不同分類器下的準確率及 F1 值統計圖表>



6.3 案例分析

<實際 CS 及 SVM 分類器準確率統計總表>

TABLE I
Accuracies Of Different Methods

		Paper Value (CS)	CS	Paper Value (SVM)	SVM	Proposed	Improvement 1	Improvement 2
Q300	1	0.688	0.683	0.648	0.665	0.708	2.44%	1.10%
Q310	2	0.693	0.739	0.697	0.689	0.758	3.03%	1.52%
Q401	3	0.500	0.534	0.506	0.499	0.584	2.95%	1.61%
Q410	4	0.714	0.740	0.709	0.709	0.774	2.30%	1.08%
Q450	5	0.531	0.593	0.572	0.525	0.643	3.05%	3.40%
Q453	6	0.566	0.609	0.564	0.562	0.703	3.45%	0.80%
Q460	7_1	0.467	0.490	0.472	0.469	0.517	2.34%	1.51%
	7_2		0.490		0.471		2.22%	1.22%
Q4704	8	0.698	0.729	0.687	0.690	0.766	2.18%	1.08%
Q4905	9	0.525	0.544	0.511	0.516	0.586	2.61%	1.30%
Q6103	10	0.639	0.676	0.643	0.666	0.691	1.49%	2.23%

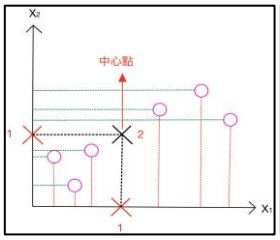
Paper Value ： 參考論文的模型準確率
CS ： CNN + Softmax 分類器的實作值
SVM ： CNN + SVM 分類器的實作值
Improvement 1 ： CS 的提升幅度
Improvement 2 ： SVM 的提升幅度

◆ 優化模型準確率

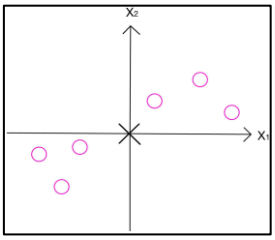
PCA（Principal Component Analysis）是一種將 2 維降維成 1 維的分類方式。首先將輸入的 2 維資料投影在兩軸上(如圖 6.3.1)，依照對應座標平均找出各軸中心，即可匯合成一中心點。再將中心點移至原點(如圖 6.3.2)並生成一條過原點的直線開始旋轉，找出最佳分類線，最後降維(投影)成 1 維，找出最佳 M 軸(如圖 6.3.3)。

透過 M 軸的旋轉，且根據畢氏定理，斜邊(a)固定不變
如圖 6.3.4，以單一資料點 A 為例，並依照以下兩規則:

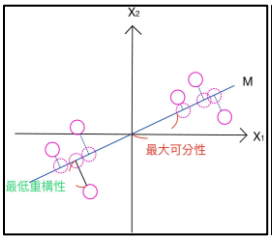
1. 最大可分性:將資料點投影到 M 軸，使其長邊(c)為最大值。
2. 最低重構性:將資料點投影到 M 軸，使其短邊(b)為最小值。



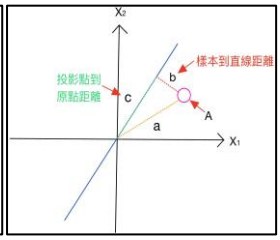
▲圖 6.3.1



▲圖 6.3.2



▲圖 6.3.3



▲圖 6.3.4

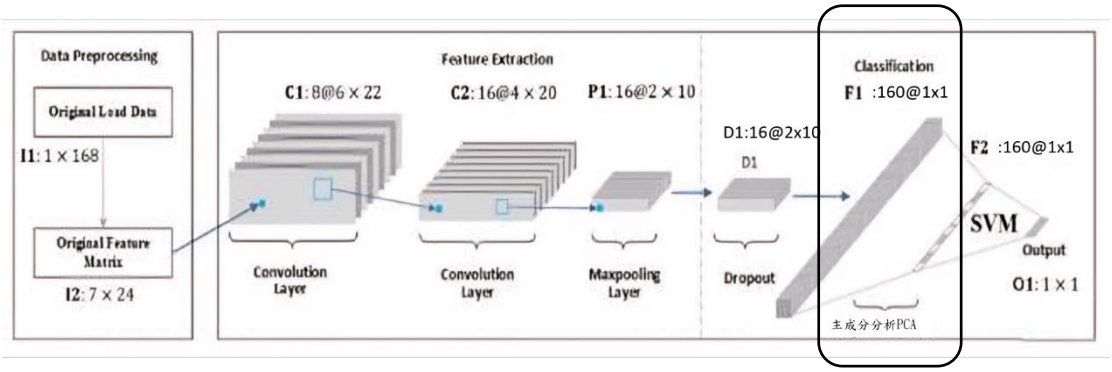
我們選擇加入 PCA 方法來提升模型準確率主要是以下原因:

（1）降維處理: 將原始的高維資料降維能加速訓練的過程，增加模型的泛化能力，同時能保留大部分信息，提高模型性能。

（2）特徵提取: 藉由上述原理介紹可知，PCA 有助於捕捉數據的主要特徵，而在高度相關的特徵中，它能保留更少但更具代表性的特徵，同時提升計算效率及模型效能。

同時也考量到 SVM+PCA 的組合能夠有效地提高模型的泛化能力，降低過擬合風險，並且在保留數據特徵的同時大幅度減少模型需要處理的維度，使模型更適合處理高維數據集，進而提升預測效果。

- 以下為加入 PCA 後的架構圖及實際運算過程↓



D1 輸出(16@2x10) => $x_{2x10} = \begin{bmatrix} x1 & \cdots & x10 \\ y1 & \cdots & y10 \end{bmatrix}$

<1> 去中心化(中心點移到原點) => $X_{2x10} = \begin{bmatrix} x1-\mu x & \cdots & x10-\mu x \\ y1-\mu y & \cdots & y10-\mu y \end{bmatrix}$

<2> 求共變異數得 2x2 方陣 => $\Sigma = \frac{1}{n} X_{2x10} X_{2x10}^{T} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$（詳見註解 1）

<3> 求特徵值 => $\begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0$，得到$\lambda_1, \lambda_2$，其中$\lambda_1 > \lambda_2$（詳見註解 2）

<4> 將λ_1代回 Σ 並標準化，求出單位特徵向量 e= $\begin{bmatrix} I \\ J \end{bmatrix}$（詳見註解 3）

<5> 將求出的單位向量與X_{2x10}做降維處理，可得 $Y_{1x10} = \begin{bmatrix} I \\ J \end{bmatrix}_{2x1}^{T} X_{2x10}$

由於 D1 輸出有 16 層，故 PCA 降維後輸出總共有 160@1x1

<註解 1>

Variance = $\frac{1}{n}\sum_{i-1}^{n}(e^T(x_i - x) - 0)^2$（去中心化故平均為 0）

$= e^T\left(\frac{1}{n}\sum_{i-1}^{n}(x_i - x)(x_i - x)^T\right)e = e^T \sum e$

$\Rightarrow \sum = \frac{1}{n}\sum_{i-1}^{n}(x_i - x)(x_i - x)^T$ ： Convariance Matrix

<註解 2>

(1) Lagrange Function ： f(e, λ) = $e^T \sum e - \lambda e^T e$

⇨ $e^T \Sigma$e ： 目標函數 ； $\lambda(1 - e^T e)$ ： 限制條件

(2) $\frac{\partial f}{\partial e} = 0$ => Σe = λe => 解出 λ

(3) $\frac{\partial f}{\partial \lambda} = 1 - e^T e = 0$ => $e^T e$ = 1（限制條件）

<註解 3> 詳細推導過程請見下頁

PCA 目的 ： 高維→低維，資料可視化

PCA 重點 ： 主軸互相垂直，投影到低維度時，資料越分散越能有效去做分類

=> 因此投影後須找到最大變異數，並由註解 1 & 2 可驗證 λ 須為最大

因為 λ 值有兩個，所以求出 e1、e2，故組成二維平面 S = span{e1,e2}

目標: e = $\frac{e1 + \alpha e2}{\sqrt{1 + \alpha^2}}$ 且 variance 最大

可推得 variance = $\lambda_1 \frac{1 + \alpha^2 \frac{\lambda_2}{\lambda_1}}{1 + \alpha^2}$

又因為 λ_1(最大) > λ_2 ，所以 $\frac{\lambda_2}{\lambda_1} < 1$

⇨ α 為 0 時，會有最大的 λ

另外，我們也針對用電量數據也有進行程式碼的微調修改，主要是因為先前整理連續 168 週的部分程式碼有檢查到參數設定錯誤，可能導致用電量數據多出許多無效資料，修正之後也有達到提升效果。

$X_{2x10} = \begin{bmatrix} x1 & \cdots & x10 \\ y1 & \cdots & y10 \end{bmatrix} \Rightarrow X_{2x10} = \begin{bmatrix} x1-\mu x & \cdots & x10-\mu x \\ y1-\mu y & \cdots & y10-\mu y \end{bmatrix}$

$\Rightarrow \quad \Sigma = \frac{1}{n} X_{2x10} X_{2x10}^T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

求 $\lambda = \begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0$ ，$\lambda = \lambda_1, \lambda_2$

（I） $\lambda = \lambda_1 \Rightarrow \text{span}\left\{ e_1 = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \right\}$

（II） $\lambda = \lambda_2 \Rightarrow \text{span}\left\{ e_2 = \begin{bmatrix} t_3 \\ t_4 \end{bmatrix} \right\}$

因為 C 有兩相異特徵值且兩特徵向量獨立

$\Rightarrow \quad P = \begin{bmatrix} t_1 & t_3 \\ t_2 & t_4 \end{bmatrix} \Rightarrow P^{-1}CP = D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$

$e = e_1 + \alpha e_2 \Rightarrow \Sigma e = \Sigma e_1 + \alpha \Sigma e_2$

$\Sigma e = \lambda e$

$\Sigma e_1 = \lambda_1 e_1 + 0 e_2$

$\Sigma e_2 = 0 e_1 + \lambda_2 e_2$

$Q_A(e) = e^T \Sigma e = Q_A(\frac{e_1 + \alpha e_2}{\sqrt{1+\alpha^2}}) = (\frac{e_1 + \alpha e_2}{\sqrt{1+\alpha^2}})^T \Sigma (\frac{e_1 + \alpha e_2}{\sqrt{1+\alpha^2}})$

$= \frac{1}{1+\alpha^2} (e_1 + \alpha e_2)^T \Sigma (e_1 + \alpha e_2)$

$= \frac{1}{1+\alpha^2} ({e_1}^T + \alpha {e_2}^T)(\Sigma e_1 + \alpha \Sigma e_2)$

$= \frac{1}{1+\alpha^2} \left({e_1}^T \Sigma e_1 + {e_1}^T \alpha \Sigma e_2 + \alpha {e_2}^T \Sigma e_1 + \alpha {e_2}^T \alpha \Sigma e_2 \right)$

$({e_1}^T \Sigma e_1 = {e_1}^T \lambda_1 e_1 = \lambda_1 {e_1}^T e_1 = \lambda_1)$

$({e_2}^T \Sigma e_2 = {e_2}^T \lambda_2 e_2 = \lambda_2 {e_2}^T e_2 = \lambda_2)$

$(\Sigma e_1 = \lambda_1 e_1)$

$(\Sigma e_2 = \lambda_2 e_2)$

$= \frac{1}{1+\alpha^2} (\lambda_1 + \alpha {e_1}^T \lambda_2 e_2 + {e_2}^T \lambda_1 e_1 + \alpha^2 \lambda_2)$

$= \frac{1}{1+\alpha^2} (\lambda_1 + \alpha^2 \lambda_2) = \lambda_1 \frac{1 + \alpha^2 \frac{\lambda_2}{\lambda_1}}{1+\alpha^2}$

6.4 結論

我們的深度學習模型是利用 CNN，特別厲害的地方是它能考慮不同時間點和日期之間的相互關係。我們的方法可以自動地挖掘大量不同的智慧電錶數據中的隱藏用電規律，這樣辨識社會人口資訊的準確度就會更高，並在一個愛爾蘭的數據集上做了實驗，結果發現 CNN 在辨識上比其他方法表現得更好。

不過，值得注意的是，辨識社會人口資訊牽涉到消費者的隱私，這是一個重要議題。所以，未來我們打算深入研究智慧電錶數據的時間細節和數據範圍，看看這些因素對隱私的影響。並計畫在未來的研究中將 CNN 與其他特徵提取方法結合，以進一步提高辨識的精確度！

以下是我們經過本次專題研究所整理的重點：

1. 提出了一種基於 CNN 的深度學習方法，用於識別社會人口信息。
2. CNN 可考慮不同時段和日期之間的相關性，提高準確性。
3. 自動提取大規模智能電表數據中的隱藏用電模式。
4. 在愛爾蘭數據集上的案例研究顯示 CNN 優於其他方法。
5. 未來研究將關注數據粒度和持續時間對隱私的影響。
6. 未來工作將結合 CNN 與其他特徵提取方法，以提高識別準確性。

第七章、附錄

7.1 數據預處理程式碼（Python）

<程式一> 處理 01-48

```
import numpy as np
import os
import errno
import sys
from tqdm import tqdm
import time
import pandas as pd
# 刷新 stdout 緩衝區，以便及時輸出資訊
sys.stdout.flush()
# 指定輸入和輸出檔路徑
input_path = "/Users/zhongboan/Desktop/python/CER Electricity Revised
March 2012/processed_data01~48/processed_File6.txt"
output_path = "/Users/zhongboan/Desktop/python/CER Electricity
Revised March 2012/processed_week_data/processed_Week_File6.txt"
# 檢查輸出檔所在的目錄是否存在，如果不存在，就創建它
if not os.path.exists(os.path.dirname(output_path)):
    try:
        os.makedirs(os.path.dirname(output_path))
    except OSError as exc:
        if exc.errno != errno.EEXIST:
            raise
start_time = time.time()
# 讀取輸入檔的每一行，並將資料存儲在一個清單中
with open(input_path, "r") as f:
    lines = f.readlines()
end_time1 = time.time()
print(f"Time elapsed for current line: {end_time1 - start_time:.2f}
seconds\n")
data = []
for line in tqdm(lines):
    line = line.strip().split()
    data.append([float(line[0]), float(line[1]), float(line[2])])
```

```
end_time2 = time.time()
data = np.array(data)
# 將資料轉換為 Pandas DataFrame 格式
df = pd.DataFrame(data, columns=['col1', 'col2', 'col3'])
# 對 col1 進行分組並按 col2 進行排序
df = df.groupby('col1').apply(lambda x:
x.sort_values('col2')).reset_index(drop=True)
print("\n1\n")
col2 = df['col2'] / 100
col2mod = df['col2'] % 100
col2 = col2.astype('int')
first_monday = col2.min()
print("2\n")
end_time2 = time.time()
print(f"Time elapsed for current line: {end_time2 - end_time1:.2f}
seconds\n")
col1 = df['col1']
day = 48
week_length = day * 7
positions = []
i = 0
with tqdm(total=len(col2) - week_length + 1, desc="Finding
positions") as pbar:
    while i < len(col2) - week_length + 1:
        if col1[i] == col1[i + week_length - 1] and col2mod[i] == 1
and all(col2[i + day * j] == col2[i] + j for j in range(1, 7)) and
((col2[i] - first_monday) % 7 == 0):
            positions.append(i)
            i += week_length
            pbar.update(week_length)
        else:
            i += day
            pbar.update(day)
pos = np.array(positions)
pos = pd.DataFrame(pos)
print("3\n")
end_time3 = time.time()
```

```
print(f"Time elapsed for current line: {end_time3 - end_time2:.2f}
seconds\n")
if not positions:
    df = pd.DataFrame()
    data = df
else:
    new_data = []
    for start_pos in tqdm(positions, desc="Extracting continuous
data"):
        new_data.append(df.iloc[start_pos:start_pos+336])
    df = pd.concat(new_data).reset_index(drop=True)
    data = df.to_numpy()
#dfcol22_25 = df[df['col2']%100 >= 25]
print("4\n")
end_time4 = time.time()
print(f"Time elapsed for current line: {end_time4 - end_time3:.2f}
seconds")
with open(output_path, "w") as f:
    for row in data:
        f.write("\t".join(map(str, row)) + "\n")
end_time5 = time.time()
print(f"Time elapsed for current line: {end_time5 - end_time4:.2f}
seconds\n")
print(f"Time elapsed for current line: {end_time5 - start_time:.2f}
seconds\n")
print("File processed successfully!")
```

<程式二> 處理一週 7 天

```
import numpy as np
import os
import errno
import sys
from tqdm import tqdm
import time
import pandas as pd
# 刷新 stdout 緩衝區，以便及時輸出資訊
sys.stdout.flush()
# 指定輸入和輸出檔路徑
input_path = "/Users/zhongboan/Desktop/python/CER Electricity Revised
March 2012/processed_week_data/processed_Week_File6.txt"
output_path = "/Users/zhongboan/Desktop/python/CER Electricity
Revised March
2012/processed_week_data01~24/processed_Week01~24_File6.txt"
# 檢查輸出檔所在的目錄是否存在，如果不存在，就創建它
if not os.path.exists(os.path.dirname(output_path)):
    try:
        os.makedirs(os.path.dirname(output_path))
    except OSError as exc:
        if exc.errno != errno.EEXIST:
            raise
start_time = time.time()
# 讀取輸入檔的每一行，並將資料存儲在一個清單中
with open(input_path, "r") as f:
    lines = f.readlines()
end_time1 = time.time()
print(f"Time elapsed for current line: {end_time1 - start_time:.2f}
seconds\n")
data = []
for line in tqdm(lines):
    line = line.strip().split()
    data.append([float(line[0]), float(line[1]), float(line[2])])
end_time2 = time.time()
print(f"Time elapsed for data reading: {end_time2 - end_time1:.2f}
seconds\n")
```

```
data = np.array(data)
# 將資料轉換為 Pandas DataFrame 格式
df = pd.DataFrame(data, columns=['col1', 'col2', 'col3'])
# 對資料進行處理
df = df.groupby('col1').apply(lambda x:
x.sort_values('col2')).reset_index(drop=True)
df = df.sort_values(by=['col1', 'col2'])
# 對 col3 每兩個值相加
df = df.reset_index(drop=True)
col3_values = df['col3'].values
result = []
for i in range(0, len(col3_values), 2):
    summed_value = col3_values[i] + col3_values[i+1]
    result.append(summed_value)
df = df[df['col2']%100 < 25]
df['col3'] = result
data = df.to_numpy()
end_time3 = time.time()
print(f"Time elapsed for data processing: {end_time3 - end_time2:.2f}
seconds\n")
# 將資料寫入輸出檔
with open(output_path, "w") as f:
    for row in data:
        f.write("\t".join(map(str, row)) + "\n")
end_time4 = time.time()
print(f"Time elapsed for data writing: {end_time4 - end_time3:.2f}
seconds\n")
print(f"Time elapsed for current line: {end_time4 - start_time:.2f}
seconds\n")
print("File processed successfully!")
```

<程式三>(01~48 變成 01~24)

```
import numpy as np
import os
import errno
import sys
from tqdm import tqdm
import time
import pandas as pd
# 刷新 stdout 緩衝區，以便及時輸出資訊
sys.stdout.flush()
# 定義輸入檔路徑清單
input_paths = [
    "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_week_data01~24/processed_Week01~24_File1.txt",
    "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_week_data01~24/processed_Week01~24_File2.txt",
    "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_week_data01~24/processed_Week01~24_File3.txt",
    "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_week_data01~24/processed_Week01~24_File4.txt",
    "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_week_data01~24/processed_Week01~24_File5.txt",
    "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_week_data01~24/processed_Week01~24_File6.txt"
]
# 定義輸出檔路徑
output_path = "/Users/zhongboan/Desktop/python/CER Electricity
Revised March 2012/processed_data_7x24/processed_data_7x24.npz"
# 創建空清單存儲資料
train_images = []
train_labels = []
# 定義函數來處理每個輸入檔
def process_input_file(input_path):
    start_time = time.time()
    # 讀取輸入檔的每一行，並將資料存儲在一個清單中
    with open(input_path, "r") as f:
        lines = f.readlines()
```

```
    end_time1 = time.time()
    print(f"Time elapsed for current line: {end_time1 -
start_time:.2f} seconds\n")
    data = []
    for line in tqdm(lines):
        line = line.strip().split()
        data.append([float(line[0]), float(line[1]), float(line[2])])
    end_time2 = time.time()
    print(f"Time elapsed for data reading: {end_time2 -
end_time1:.2f} seconds\n")
    data = np.array(data)
    # 將資料轉換為 Pandas DataFrame 格式
    df = pd.DataFrame(data, columns=['col1', 'col2', 'col3'])
    # 對資料進行處理
    df = df.groupby('col1').apply(lambda x:
x.sort_values('col2')).reset_index(drop=True)
    df = df.sort_values(by=['col1', 'col2'])
    df = df.reset_index(drop=True)
    col1 = df['col1']
    # 讀取 Excel 數據
    excel_data = pd.read_excel("/Users/zhongboan/Desktop/python/CER
Electricity Revised March 2012/標籤檔/Q300_processed.xlsx") # 根據實
際情況修改檔案名和路徑
    excel_col1 = excel_data.iloc[:, 0]
    excel_col2 = excel_data.iloc[:, 1]
    df = df[df['col1'].isin(excel_col1)] # 相同 ID 保留 不同刪除
    df = df.groupby('col1').apply(lambda x:
x.sort_values('col2')).reset_index(drop=True)
    df = df.sort_values(by=['col1', 'col2'])
    df = df.reset_index(drop=True)
    col1 = df['col1']
    # 提取 col3 並重塑資料
    col3_values = df['col3'].values
    label = []
    resized_data = []
    for i in tqdm(range(0, len(col3_values), 168), desc="Reshaping
data"):
        for k in range(len(excel_col1)):
```

```
            if col1[i] == excel_col1[k]:
                chunk = col3_values[i:i+168].reshape((7, 24, 1))
                resized_data.append(chunk)
                label.append(excel_col2[k])
                break  # 找到匹配的值後，跳出內層迴圈
    end_time3 = time.time()
    print(f"Time elapsed for reshaping and saving data: {end_time3 -
end_time2:.2f} seconds\n")
    # 將數據添加到訓練集清單中
    train_images.extend(resized_data)
    train_labels.extend(label)
# 處理每個輸入檔
for input_path in input_paths:
    process_input_file(input_path)
# 轉換為 NumPy 陣列
train_images = np.array(train_images)
train_labels = np.array(train_labels)
# 保存資料集為 NPZ 檔
np.savez(output_path, images=train_images, labels=train_labels)
```

7.2 標籤檔預處理程式碼 (Matlab)

<Q301>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q300.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 Young (<35)
%2 表示 Medium (35~65)
%3 表示 Old (>65)
C(A(:, 2) == 1 | A(:, 2) == 2, 2) = 1;
C(A(:, 2) == 3 | A(:, 2) == 4 | A(:, 2) == 5, 2) = 2;
C(A(:, 2) == 6, 2) = 3;
% 從 C 中刪除值為 7 的行
C(A(:, 2) == 7, :) = [];

%寫入新的 excel 檔
outputfilename = 'Q300_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q310>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q310.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);
```

```
%轉換成 label 表示
%1 表示 Yes
%2 表示 No
C(A(:, 2) == 1 | A(:, 2) == 2 | A(:, 2) == 3, 2) = 1;
C(A(:, 2) == 4 | A(:, 2) == 5 | A(:, 2) == 7, 2) = 1;
C(A(:, 2) == 6, 2) = 2;

%寫入新的 excel 檔
outputfilename = 'Q310_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q401>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q410.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 AB(與原本分類相同所以不用改)
%2 表示 C1+C2
%3 表示 DE
%其餘皆刪除(原本的 5&6)
C(A(:, 2) == 1, 2) = 1;
C(A(:, 2) == 2 | A(:, 2) == 3, 2) = 2;
C(A(:, 2) == 4, 2) = 3;
C(A(:, 2) == 5 | A(:, 2) == 6 ,:) = [];

%寫入新的 excel 檔
outputfilename = 'Q410_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q410>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q410.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 Yes
%2 表示 No
C(A(:, 2) == 3 , 2) = 1;
C(A(:, 2) == 1 | A(:, 2) == 2, 2) = 2;

%寫入新的 excel 檔
outputfilename = 'Q410_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q450>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q450.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 Detached or bungalow
%2 表示 Semi-detached or terraced
% 從 C 中刪除值 1&6
C(A(:, 2) == 2 | A(:, 2) == 4, 2) = 1;
C(A(:, 2) == 3 | A(:, 2) == 5, 2) = 2;
```

```
C(A(:, 2) == 1 | A(:, 2) == 6, :) = [];

%寫入新的 excel 檔
outputfilename = 'Q450_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q453>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q453.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);

%建立一個表格 B 先刪除無效值
B = A;
B(B(:, 2) < 1000 , :) = [];
B(B(:, 2) > 2011 , :) = [];

%轉換成 label 表示
%1 表示 old(>30)
%2 表示 new(<30)
C=B;
C((2011 - B(:, 2)) < 30 ,2) = 1;
C((2011 - B(:, 2)) >= 30 ,2) = 2;

%寫入新的 excel 檔
outputfilename = 'Q453_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q460>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q460.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');
```

```
%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 very low (<3)
%2 表示 low (=3)
%3 表示 high (=4)
%4 表示 very high (>4)
C(A(:, 2) == 1 | A(:, 2) == 2,2) = 1;
C(A(:, 2) == 3 ,2) = 2;
C(A(:, 2) == 4 ,2) = 3;
C(A(:, 2) == 5 ,2) = 4;
C(A(:, 2) == 6 ,:) = [];

%寫入新的 excel 檔
outputfilename = 'Q460_processed(6 是拒絕回答).xlsx';
xlswrite(outputfilename, C);
```

<Q460_1>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q460.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 very low (<3)
%2 表示 low (=3)
%3 表示 high (=4)
%4 表示 very high (>4)
C(A(:, 2) == 1 | A(:, 2) == 2,2) = 1;
C(A(:, 2) == 3 ,2) = 2;
```

```
C(A(:, 2) == 4 ,2) = 3;
C(A(:, 2) == 5 | A(:, 2) == 6,2) = 4;

%寫入新的 excel 檔
outputfilename = 'Q460_processed(6 是 6 間房間).xlsx';
xlswrite(outputfilename, C);
```

<Q4704>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q4905.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 Up to half
%2 表示 Three quarters or more
C(A(:, 2) == 1 | A(:, 2) == 2, 2) = 1;
C(A(:, 2) == 3 | A(:, 2) == 4 | A(:, 2) == 5, 2) = 2;

%寫入新的 excel 檔
outputfilename = 'Q4905_processed.xlsx';
xlswrite(outputfilename, C);
```

Q4905

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q4905.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);
```

```
%轉換成 label 表示
%1 表示 Up to half
%2 表示 Three quarters or more
C(A(:, 2) == 1 | A(:, 2) == 2, 2) = 1;
C(A(:, 2) == 3 | A(:, 2) == 4 | A(:, 2) == 5, 2) = 2;

%寫入新的 excel 檔
outputfilename = 'Q4905_processed.xlsx';
xlswrite(outputfilename, C);
```

<Q6103>

```
%讀入原始 excel 檔並存到 table
inputfilename = 'Q6103.xlsx'
table=readtable(inputfilename,'TextType','string','VariableNamingRule
', 'preserve');

%表格 A 是原始資料/表格 C 是處理後的
A = table2array(table);
C = A(:,1);

%轉換成 label 表示
%1 表示 small(<100)
%2 表示 medium(100-200)
%3 表示 big(>200)
C(A(:, 2) <= 1000 ,2) = 1;
C(A(:, 2) > 1000 & A(:, 2) <= 2200 ,2) = 2;
C(A(:, 2) > 2200 ,2) = 3;
C(A(:, 2) == 999999999 ,:) = [];

%寫入新的 excel 檔
outputfilename = 'Q6103_processed.xlsx';
xlswrite(outputfilename, C);
```

7.3 神經網路程式碼

```
import tensorflow as tf
from tensorflow import keras
import os
import cv2
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from sklearn import svm
from sklearn.model_selection import train_test_split
from tqdm import tqdm
tf.keras.backend.clear_session()
#%%
# 定義文件路徑
npz_file = "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_data_7x24/processed_data_7x24_453.npz"

# 加載數據集
data = np.load(npz_file)

# 提取訓練數據和標籤
train_images = data['images']
train_labels = data['labels']

# 將數據轉換為 4 維格式
input_data = train_images.reshape(train_images.shape[0], 7, 24, 1)

# 劃分訓練集和驗證集
train_ratio = 0.8
X_train, X_val, y_train, y_val = train_test_split(input_data, train_labels, test_size=1-
train_ratio, random_state=42)
#%%
# 建立 CNN 模型
model = Sequential()
model.add(Conv2D(8, (2, 3), padding='valid', activation='relu', input_shape=(7, 24,
1), name='C1'))
```

```
model.add(Conv2D(16, (3, 3), padding='valid', activation='relu', input_shape=(6, 22,
8), name='C2'))
model.add(MaxPooling2D(pool_size=(2, 2), name='P1'))
model.add(Dropout(0.5, name='DR1'))
model.add(Flatten(name='F1'))
model.add(Dense(units=32, activation='relu', name='D1'))
model.add(Dense(units=10, activation='softmax', name='D2'))
print(model.summary())
#%%
# 編譯模型
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
train_history = model.fit(X_train, y_train, validation_data=(X_val, y_val),
batch_size=32, epochs=10)
#%%
import matplotlib.pyplot as plt
def show_train_history(train_history,train,validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train history')
    plt.ylabel('train')
    plt.xlabel('epoch')
    # 設置圖例在左上角
    plt.legend(['train','validation'],loc='upper left')
    plt.show()
#%%
show_train_history(train_history,'accuracy','val_accuracy')
show_train_history(train_history,'loss','val_loss')
```

7.4 SVM 提升準確率程式碼

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from tqdm import tqdm

# 加載數據集
npz_file = "/Users/zhongboan/Desktop/python/CER Electricity Revised March
2012/processed_data_7x24/processed_data_7x24_300.npz"
data = np.load(npz_file)
train_images = data['images']
train_labels = data['labels']

# 將數據轉換為 4 維格式
input_data = train_images.reshape(train_images.shape[0], 7, 24, 1)
# 劃分訓練集和驗證集
train_ratio = 0.8
X_train, X_val, y_train, y_val = train_test_split(input_data, train_labels, test_size=1-
train_ratio, random_state=42)
# 設定抽樣比例
subsample_ratio = 0.1    # 10% 的抽樣比例
# 計算抽樣的數量
num_samples_train = int(X_train.shape[0] * subsample_ratio)
num_samples_val = int(X_val.shape[0] * subsample_ratio)
# 隨機選取抽樣的索引
random_indices_train = np.random.choice(X_train.shape[0], num_samples_train,
replace=False)
random_indices_val = np.random.choice(X_val.shape[0], num_samples_val,
replace=False)
# 使用抽樣的索引來獲取子樣本
X_train_subsampled = X_train[random_indices_train]
X_val_subsampled = X_val[random_indices_val]
y_train_subsampled = y_train[random_indices_train]
y_val_subsampled = y_val[random_indices_val]
```

```
# 印出子樣本的形狀
print("X_train_subsampled shape:", X_train_subsampled.shape)
print("X_val_subsampled shape:", X_val_subsampled.shape)
print("y_train_subsampled shape:", y_train_subsampled.shape)
print("y_val_subsampled shape:", y_val_subsampled.shape)
# 使用 PCA 對整個訓練集和驗證集進行降維
pca = PCA(n_components=50)  # 指定 PCA 降維後的維度，這裡設置為 50
X_train_flatten = X_train_subsampled.reshape(num_samples_train, -1)  # 將 4 維數據轉換為 2 維以滿足 PCA 的要求
X_val_flatten = X_val_subsampled.reshape(num_samples_val, -1)
# 對整個訓練集和驗證集進行 PCA 轉換
X_train_pca = pca.fit_transform(X_train_flatten)
X_val_pca = pca.transform(X_val_flatten)

# 使用 SVM 模型
svm_model = SVC(kernel='linear', decision_function_shape='ovr')
n_iterations = 10  # 增加迭代次數
batch_size = num_samples_train // n_iterations  # 確保每個迭代的批次大小相同

# 訓練 SVM 並顯示進度條
with tqdm(total=n_iterations, desc="Training svm") as pbar:
    for i in range(n_iterations):
        start_idx = i * batch_size
        end_idx = start_idx + batch_size
        # 在每次迭代中，對批次的原始數據進行 PCA 轉換
        X_batch =
pca.transform(X_train_subsampled[start_idx:end_idx].reshape(batch_size, -1))
        y_batch = y_train_subsampled[start_idx:end_idx]
        svm_model.fit(X_batch, y_batch)
        pbar.update(1)

# 預測並驗證 SVM 並顯示進度條
svm_predictions = pca.transform(X_val_subsampled.reshape(num_samples_val, -1))
# 將驗證集轉換為 PCA 表示
svm_predictions = svm_model.predict(svm_predictions)  # 使用 SVM 進行預測
accuracy = accuracy_score(y_val_subsampled, svm_predictions)
print("\n")
print("SVM 準確率 :", accuracy)
```

7.5 數字辨識範例程式碼

<訓練並儲存模型>

```
import tensorflow as tf
import keras
import os
from tqdm import tqdm
import cv2
import numpy as np
import pandas as pd
from keras.datasets import mnist
from keras.utils import np_utils
import numpy as np
np.random.seed(10)
#%%
(x_Train,y_Train),(x_Test,y_Test)=mnist.load_data()
print(x_Train.shape)
# 將 feature 轉換成 4 維度 ，也就是 6000*28*28*1
x_Train4D =x_Train.reshape(x_Train.shape[0],28,28,1).astype('float32')
x_Test4D =x_Test.reshape(x_Test.shape[0],28,28,1).astype('float32')
print(x_Train.shape)
#%%
# 數值標準化
x_Train4D_normalize=x_Train4D/255
x_Test4D_normalize=x_Test4D/255
#%%
# label 用 one hot enconding 轉換
y_TrainOneHot=np_utils.to_categorical(y_Train)
y_TestOneHot=np_utils.to_categorical(y_Test)
#%%
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D
model = Sequential()
# 建立卷積層 1=>28*28 大小，但是產生 16 個影像
model.add(Conv2D(filters=16,kernel_size=(5,5),padding='same',input_shape=(28,28,
1),activation='relu'))
# 建立池化層 1
```

```
# 執行第一次降採樣，原本的 28*28 變成 14*14=>16 個 14*14
model.add(MaxPooling2D(pool_size=(2,2)))
# 建立卷積層 2=>將原本的 16 個影像轉換成 36，大小還是 14*14
model.add(Conv2D(filters=36,kernel_size=(5,5),padding='same',activation='relu'))
# 建立池化層 2=>原本 36 個 14*14 變成，36 個 7*7
model.add(MaxPooling2D(pool_size=(2,2)))
# 加入 dropout 避免 overfitting，這裡的 0.25，是在每一次捨棄 25%的神經元
model.add(Dropout(0.25))
# ----------------------------
# 建立神經網路
# 建立平坦層=>把他們壓扁，所以是 36 個*14*14=1764 對應 1764 個神經元
model.add(Flatten())
# 建立隱藏層
model.add(Dense(128,activation='relu'))
# 捨棄 50%神經元
model.add(Dropout(0.5))
# 建立輸出層
model.add(Dense(10,activation='softmax'))
print(model.summary())
#%%
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'
])
train_history=model.fit(x=x_Train4D_normalize,y=y_TrainOneHot,validation_split=0.
2,epochs=10,batch_size=300,verbose=2)
#%%
import matplotlib.pyplot as plt
def show_train_history(train_history,train,validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train history')
    plt.ylabel('train')
    plt.xlabel('epoch')
    # 設置圖例在左上角
    plt.legend(['train','validation'],loc='upper left')
    plt.show()
#%%
show_train_history(train_history,'accuracy','val_accuracy')
show_train_history(train_history,'loss','val_loss')
```

```
#%%
scores=model.evaluate(x_Test4D_normalize,y_TestOneHot)
print(scores[1])
#%%
predict_x=model.predict(x_Test4D_normalize)
prediction=np.argmax(predict_x,axis=1)
prediction[:10]
#%%
def plot_images_labels_prediction(images,labels,prediction,idx,num=10):
    fig= plt.gcf()
    fig.set_size_inches(12,14)
    if num>25:num=25
    for i in range(0,num):
        ax=plt.subplot(5,5,i+1)
        ax.imshow(images[idx],cmap='binary')
        title="label=" +str(labels[idx])
        if len(prediction)>0:
            title+=",predict="+str(prediction[idx])
        ax.set_title(title,fontsize=10)
        ax.set_xticks([]);ax.set_yticks([])
        idx+=1
    plt.show()
#%%
plot_images_labels_prediction(x_Test,y_Test,prediction,idx=0)
#%%
import pandas as pd
pd.crosstab(y_Test,prediction,rownames=['label'],colnames=['predict'])
# 儲存模型
model.save('my_model.h5')

# 讀取模型
from keras.models import load_model
loaded_model = load_model('my_model.h5')
```

<測試自己的手寫圖片>

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```
"""
Created on Mon Mar 13 12:01:38 2023

@author: zhongboan
"""

#%%
import tensorflow as tf
import keras
import os
from tqdm import tqdm
import cv2
import numpy as np
import pandas as pd
from keras.datasets import mnist
from keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
np.random.seed(10)
from tensorflow.keras.models import load_model
#%%

#%%
#  讀取模型
model_path = os.path.join(os.getcwd(), 'my_model.h5')
loaded_model = load_model(model_path)
# Path to the folder containing the JPEG images
folder_path = '/Users/zhongboan/programming/test_file'

# Load the trained model
#model = tf.keras.models.load_model('/path/to/saved/model')

# Empty list to hold the processed arrays
processed_arrays = []
pred_array=[]
pred_number=[]
# Loop over all files in the folder
```

```
for file_name in os.listdir(folder_path):
    file_path = os.path.join(folder_path, file_name)
    if file_name.endswith('.jpg') or file_name.endswith('.jpeg'):
        img = Image.open(file_path)
        box = (800, 1000, 2500, 3000)
        img = img.crop(box)
        processed_img = img.convert('L').resize((28, 28))
        output_path = os.path.join('/Users/zhongboan/programming', 'processed_'
+ file_name)
        processed_img.save(output_path)
        plt.imshow(processed_img, cmap='gray')
        plt.show()
        processed_array = np.asarray(processed_img).copy()
        processed_array[processed_array > 130] = 255    # 二值化
        processed_array = 255-processed_array
        processed_img = Image.fromarray(processed_array)# 反轉
        plt.imshow(processed_img, cmap='gray')
        plt.show()

        #processed_array[processed_array <= 130] = 0
        #processed_array = np.logical_not(processed_array)
        #threshold_value = 130
        #processed_array = processed_img.point(lambda x: x if x <=
threshold_value else 255, '1')
        #processed_array = np.asarray(processed_img).copy()
        #processed_array = threshold(processed_array, 120, 255,
cv2.THRESH_BINARY_INV)
        # 將二值化後的數組賦回到 processed_img 中
        #processed_img = Image.fromarray(processed_array)

        # 顯示圖像
        #plt.imshow(processed_img, cmap='gray')
        #plt.show()

        # 將處理後的圖像保存
        output_path_black = os.path.join('/Users/zhongboan/programming',
'processed_black' + file_name)
        processed_img.save(output_path_black)
```

```
            # Preprocess the image for prediction
            img = cv2.imread(output_path_black, cv2.IMREAD_GRAYSCALE)#讀取圖片
            img = cv2.resize(img, (28, 28), interpolation=cv2.INTER_AREA)
            img = img.reshape(1, 28, 28, 1)
            img = img.astype('float32') / 255
            img = np.expand_dims(img, axis=-1)
            # Make predictions
            pred = loaded_model.predict(img)
            # Print the predicted class
            print(f"Predicted class: {np.argmax(pred)}")
            #  获取预测结果的矩阵
            pred_array.append(pred)
            pred_number.append(np.argmax(pred))
#%%
#  打印预测结果矩阵
print(pred_array)
print(pred_number)
#%%
#%%

#%%
import numpy as np
import matplotlib.pyplot as plt

#  載入  MNIST  資料集
from keras.datasets import mnist
(x_Train,y_Train),(x_Test,y_Test)=mnist.load_data()
print(x_Train.shape)
#  將 feature  轉換成 4 維度  ，也就是 6000*28*28*1
x_Train4D =x_Train.reshape(x_Train.shape[0],28,28,1).astype('float32')
x_Test4D =x_Test.reshape(x_Test.shape[0],28,28,1).astype('float32')
print(x_Train.shape)
#%%
#  數值標準化
x_Train4D_normalize=x_Train4D/255
x_Test4D_normalize=x_Test4D/255
#%%
# label  用 one hot enconding  轉換
```

```
# 找出 y_Test 中所有為 1 的 index
idx_one = np.where(y_Test == 5)[0]

# 篩選出 x_Test 中對應的圖片
x_Test_one = x_Test[idx_one]

# 預測 x_Test_one 的標籤
predict_one = loaded_model.predict(x_Test_one)
prediction_one = np.argmax(predict_one, axis=1)

# 定義秀圖函數
def plot_images_labels_prediction(images, labels, prediction, idx, num=10):
    fig = plt.gcf()
    fig.set_size_inches(12, 14)
    if num > 1:
        num =1
    for i in range(0, num):
        ax = plt.subplot(5, 5, i+1)
        ax.imshow(images[idx], cmap='binary')
        title = "label=" + str(labels[idx])
        if len(prediction) > 0:
            title += ",predict=" + str(prediction[idx])
        ax.set_title(title, fontsize=10)
        ax.set_xticks([])
        ax.set_yticks([])
        idx += 1
    plt.show()
#%%
# 顯示所有數字 1 的圖片以及預測值
plot_images_labels_prediction(x_Test_one, y_Test[idx_one], prediction_one, idx=25,
num=len(x_Test_one))
print(predict_one)
#%%
# 找出 y_Test 中所有為 1 的 index
idx_one = np.where(y_Test == 1)[0]

# 預測所有標籤為 1 的圖片
predict_one = loaded_model.predict(x_Test4D_normalize[idx_one])
```

```
prediction_one = np.argmax(predict_one, axis=1)

# 找出預測不是 1 的圖片
idx_wrong = np.where(prediction_one != 1)[0]

# 篩選出 x_Test 中對應的圖片
x_Test_wrong = x_Test[idx_one[idx_wrong]]

# 顯示所有標籤是 1 但預測不是 1 的圖片
plot_images_labels_prediction(x_Test_wrong, y_Test[idx_one[idx_wrong]],
prediction_one[idx_wrong], idx=0, num=len(x_Test_wrong))
#%%
'''
os
提供了與操作系統相關的函數和方法
例如文件/目錄操作、環境變量、進程管理等
在導入了 os 模塊後，我們可以使用 os 中的函數和方法來操作系統。
————————————————————
pil 圖像處理庫
image 處理圖像 保存圖像 讀取圖像
————————————————————
cv2
cv2 是 OpenCV 的 Python 接口庫
提供了許多圖像和視頻處理的功能
例如加載、保存和修改圖像，以及人臉檢測、物體跟踪等
通過 import cv2，可以在 Python 中使用 OpenCV 的函數和類來進行圖像和視頻
處理。
'''
#%%
'''
os.listdir(folder_path)
是用來列出指定目錄中的所有文件和文件夾的名稱
返回一個包含所有文件名的列表
for file_name in os.listdir(folder_path)
就是對目錄中的每個文件名進行迭代，對於每個文件名，程式將會執行相應的
操作
在此例中，如果文件名以".jpg"或".jpeg"結尾，則會對該文件執行圖像處理操
作。
```

'''

#%%

'''

OpenCV 的 cv2.imread 在讀取圖片時，可以在第二個參數指定圖片的格式，可用的選項有三種：

cv2.IMREAD_COLOR

此為預設值，這種格式會讀取 RGB 三個 channels 的彩色圖片，而忽略透明度的 channel。

cv2.IMREAD_GRAYSCALE

以灰階的格式來讀取圖片。

cv2.IMREAD_UNCHANGED

讀取圖片中所有的 channels，包含透明度的 channel。

————————————————————

void resize

(InputArray src, OutputArray dst, Size dsize, double fx=0, double fy=0, int interpolation=INTER_LINEAR);

src：輸入，原圖像，即待改變大小的圖像；

dst：輸出，改變大小之後的圖像，這個圖像和原圖像具有相同的內容，只是大小和原圖像不一樣而已；

dsize：輸出圖像的大小

如果這個參數不為 0

那麼就代表將原圖像縮放到這個 Size(width，height)指定的大小

如果這個參數為 0，那麼原圖像縮放之後的大小就要通過下面的公式來計算：

size = Size(round(fx*src.cols), round(fy*src.rows))

其中，fx 和 fy 就是下面要說的兩個參數，是圖像 width 方向和 height 方向的縮放比例。

fx：width 方向的縮放比例，如果它是 0，那麼它就會按照

(double)dsize.width/src.cols 來計算；

fy：height 方向的縮放比例，如果它是 0，那麼它就會按照

(double)dsize.height/src.rows 來計算；

interpolation：這個是指定插值的方式，圖像縮放之後，肯定像素要進行重新計算的

就靠這個參數來指定重新計算像素的方式，有以下幾種：

INTER_NEAREST - 最鄰近插值

INTER_LINEAR - 雙線性插值，如果最後一個參數你不指定，默認使用這種方法

INTER_AREA -區域插值

resampling using pixel area relation

It may be a preferred method for image decimation
as it gives moire’-free results. But when the image is zoomed
it is similar to the INTER_NEAREST method.
INTER_CUBIC - 4x4 像素鄰域內的雙立方插值
INTER_LANCZOS4 - 8x8 像素鄰域內的 Lanczos 插值
————————————————————

reshape 函式是為了將 img 這張圖片轉換成模型接受的輸入格式
模型接受的輸入格式為 (samples, height, width, channels)
samples 表示樣本數，這裡是 1，因為我們只輸入一張圖片
height 和 width 表示圖片的高度和寬度，這裡都是 28，因為我們將圖片轉換成了 28x28 的大小
channels 表示圖片的通道數，這裡是 1，因為我們將圖片轉換成了灰階格式，只有一個通道。
因此，img.reshape(1, 28, 28, 1) 的作用是將 img 這張圖片轉換成模型接受的輸入格式 (1, 28, 28, 1)。

第八章、參考文獻

1. 林大貴，TensorFlow+Keras 深度學習人工智慧實務應用，博碩圖書出版公司，民國 106 年。
2. 黃志勝，機器學習的統計基礎:深度學習背後的核心技術，旗標圖書出版公司，民國 110 年。
3. Yi Wang, Student Member, IEEE, Qixin Chen, Senior Member, IEEE, Dahua Gan, Jingwei Yang, Student Member, IEEE, Daniel S. Kirschen, Fellow, IEEE and Chongqing Kang, Fellow, IEEE, “Deep Learning-Based Socio-Demographic Information Identification From Smart Meter Data”,IEEE Transactions on smart grid,Vol.10, No. 3, May 2019 2593.
4. 李文峰,邓晓平,彭伟,孟宋萍.基于自学习边权重图卷积网络的用户用能分类.计算机系统应用,2022,31(9):294－299. http://www.c-s-a.org.cn/1003-3254/8694.html