1. Introduction

這次作業主要使用 autoencoder 搭配建立 NN 模型下去做 link prediction，而 autoencoder 和 NN 模型這兩個部分會分別嘗試使用不同層(例如:3 層或 4 層 hidden layer)以及不同 epoch 下去做預測，另外 NN 模型內除了輸入、隱藏、輸出層之外，還加入 dropout rate 下去訓練分類模型，最後也有得到更好的結果。

2. Methodology

這次作業的方法會分別從資料處理、autoencoder、NN model 三個部分下去做說明。

(1) 資料處理

首先，先處理 content 資料集，將 content 每個 bit 分割，把每個 bit 變一個特徵，因為 content 的 Node 編號是隨機，所以進行重新排列，並且把第一個 bit 欄位名稱設為 id，資料呈現如下圖(以 dataset1 作為範例):

	id	1	2	3	4	5	6	7	8	9	...	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433
0	351	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
1	1357	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
2	272	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	583	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	391	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2703	1081	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2704	743	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2705	1006	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2706	1826	0	0	0	0	1	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
2707	2272	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

2708 rows × 1434 columns

下圖為依照 id 大小的排序結果，可以看出 dataset1 具有 2708 個 nodes、dataset2 具有 3312 個 nodes、dataset3 具有 877 個 nodes。

	0	1	2	3	4	5	6	7	8	9	...	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
4	4	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2703	2703	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2704	2704	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2705	2705	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2706	2706	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2707	2707	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

2708 rows × 1434 columns

	0	1	2	3	4	5	6	7	8	9	...	3694	3695	3696	3697	3698	3699	3700	3701	3702	3703
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3307	3307	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3308	3308	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3309	3309	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3310	3310	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3311	3311	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

3312 rows × 3704 columns

	0	1	2	3	4	5	6	7	8	9	...	1694	1695	1696	1697	1698	1699	1700	1701	1702	1703
0	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
2	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
872	872	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
873	873	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
874	874	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
875	875	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
876	876	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0

877 rows × 1704 columns

過程中為了測試分割的過程是否有錯誤，所以有用矩陣下去看

```
contentArray_sort[3][1747]
```

```
1
```

依照對應的 node 和對應特徵 bit，若呈現為 1，就可以證明分割上大概沒什麼問題了。

然後創建相鄰矩陣，根據 node 數量建立 matrix 大小，對應 train 資料有連接上則給予 1，無則訂為 0。我在建立相鄰矩陣這邊卡很久，一開始建出來的資料如下圖（是一個 2708*5416 的矩陣）這邊在最後實驗使用這個矩陣跑出來的效果分數很低，後來發現這個矩陣的建立方式是有方向的，我把 from 和 to 分別建立相鄰矩陣再把它合併，但這次作業是無方向性的資料，互相追隨的情況下，不需要分兩個矩陣

	0	1	2	3	4	5	6	7	8	9	...	2698	2699	2700	2701	2702	2703	2704	2705	2706	2707
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2703	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2704	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2705	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2706	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2707	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2708 rows × 5416 columns

所以做完更動後，正確的相鄰矩陣應該如下圖（是一個 2708*1434 的矩陣），而因為 dataset2 和 dataset3 在做資料處理的方式是一樣的，所以這邊全部都只用 dataset1 代表來做說明。

	0	1	2	3	4	5	6	7	8	9	...	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
4	4	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2703	2703	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2704	2704	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2705	2705	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2706	2706	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2707	2707	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

2708 rows × 1434 columns

根據上圖的表格可以得知 node 之間的連接狀況，1 代表相連、0 代表不相連。

Content 和 nodes 間的 matrix 都建立好特徵矩陣之後，把他們倆個合併為最終的訓練特徵資料，如下圖:

	0	1	2	3	4	5	6	7	8	9	...	4132	4133	4134	4135	4136	4137	4138	4139	4140	4141
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2703	2703.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2704	2704.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2705	2705.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2706	2706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2707	2707.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2708 rows × 4142 columns

然後再把對應的特徵矩陣匯入到 train data 裡面，最後得到的 train data 如下圖:

	0	1	2	3	4	5	6	7	8	9	...	8272	8273	8274	8275	8276	8277	8278	8279	8280	8281
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8681	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8682	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8683	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8684	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8685	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

8686 rows × 8282 columns

然後把這個 train data 當成 autoencoder 的 input data，進行下一步。

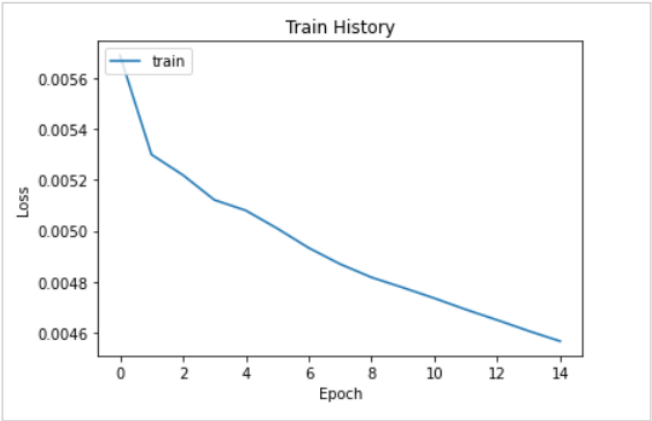
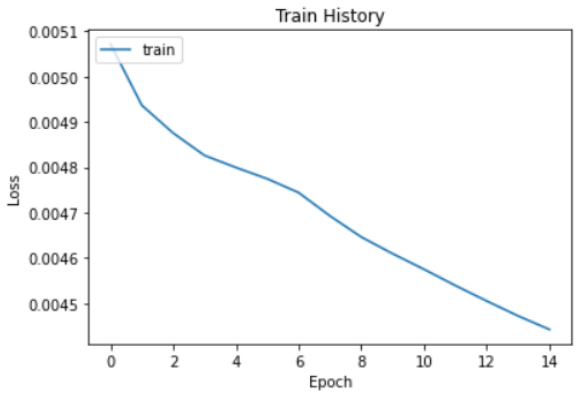
(2) Autoencoder

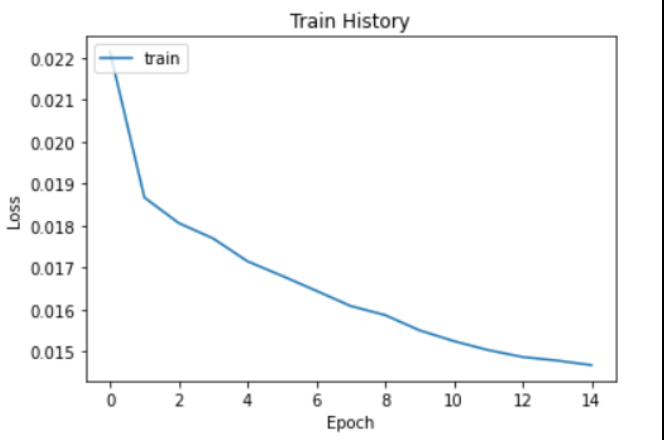
因為產出的特徵非常多，所以試著先使用 autoencoder 降維再升維，萃取其中最有用的特徵。這邊使用一個 input 層，再來使用三層 encoder，再加一層 encoder_output，然後三層 decoder，最後 output 層會得到新的特徵組合，autoencoder model 實際結構如下圖:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 8282)]            0

 dense (Dense)               (None, 512)               4240896

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 20)                2580

 dense_3 (Dense)             (None, 10)                210

 dense_4 (Dense)             (None, 20)                220

 dense_5 (Dense)             (None, 128)               2688

 dense_6 (Dense)             (None, 512)               66048

 dense_7 (Dense)             (None, 8282)              4248666

=================================================================
Total params: 8,626,972
Trainable params: 8,626,972
Non-trainable params: 0
_________________________________________________________________
```

過程中有嘗試增加或減少 encoder 和 decoder 的 layer 來看是否會改善模型的預測表現，但是跑出來的結果並沒有多大的差異，所以這邊還是維持原本的設定。

而 autoencoder 的 epoch 設為 15，loss 的訓練表現如下圖:

	Dataset1
	Dataset2



Dataset3

因為模型的 layer 數設定一樣，所以 loss 的呈現在三個資料集上面就只是資料量(nodes)上的差異而已。

然後我有嘗試增加 epoch 看最後預測效果是否有差，但結果跑出來表現是差不多的，而且 autoencoder 在訓練上越多 epoch 時間越久，所以最後還是維持在 epoch=15。

經過 autoencoder 之後，接下來做 NN 分類模型的訓練。

(3) NN model

NN model 和 Autoencoder 不一樣的地方是，因為資料量的差異，在 NN 上 layer 的設定有因為不同資料集做一點微調，所以下面會依照資料集來分別說明，而這邊所呈現的模型架構都是經過多次測試之後，依照預測表現最好的模型把它呈現出來。

i. Dataset1

Dataset1 的 NN 模型架構如下圖: (最終取右邊紅色為 dataset1 的模型架構)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 512)               4240896

 dropout (Dropout)           (None, 512)               0

 dense_9 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_10 (Dense)            (None, 64)                16448

 dropout_2 (Dropout)         (None, 64)                0

 dense_11 (Dense)            (None, 20)                1300

 dropout_3 (Dropout)         (None, 20)                0

 dense_12 (Dense)            (None, 1)                 21

=================================================================
Total params: 4,389,993
Trainable params: 4,389,993
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_17 (Dense)            (None, 256)               2120448

 dropout_7 (Dropout)         (None, 256)               0

 dense_18 (Dense)            (None, 64)                16448

 dropout_8 (Dropout)         (None, 64)                0

 dense_19 (Dense)            (None, 20)                1300

 dropout_9 (Dropout)         (None, 20)                0

 dense_20 (Dense)            (None, 1)                 21

=================================================================
Total params: 2,138,217
Trainable params: 2,138,217
Non-trainable params: 0
_________________________________________________________________
```

ii. Dataset2

Dataset2 的 NN 模型架構如下圖: (最終取左邊紅色為 dataset2 的模型架構)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 256)               3591936

 dropout (Dropout)           (None, 256)               0

 dense_9 (Dense)             (None, 64)                16448

 dropout_1 (Dropout)         (None, 64)                0

 dense_10 (Dense)            (None, 20)                1300

 dropout_2 (Dropout)         (None, 20)                0

 dense_11 (Dense)            (None, 1)                 21

=================================================================
Total params: 3,609,705
Trainable params: 3,609,705
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_19 (Dense)            (None, 64)                897984

 dropout_8 (Dropout)         (None, 64)                0

 dense_20 (Dense)            (None, 20)                1300

 dropout_9 (Dropout)         (None, 20)                0

 dense_21 (Dense)            (None, 1)                 21

=================================================================
Total params: 899,305
Trainable params: 899,305
Non-trainable params: 0
_________________________________________________________________
```

iii. Dataset3

Dataset3 的 NN 模型架構如下圖: (最終取右邊紅色為 dataset3 的模型架構

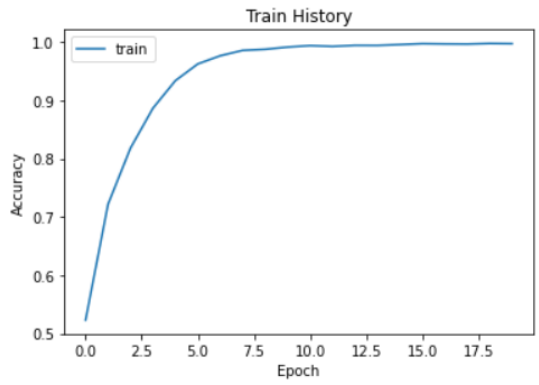
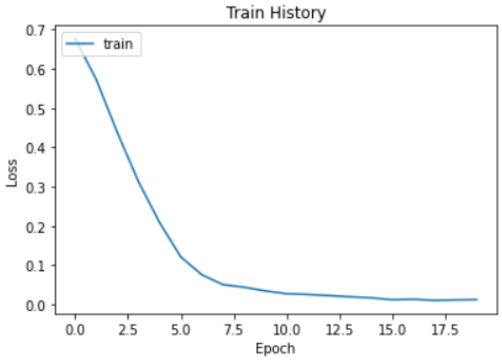
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 256)               2258688

 dropout_4 (Dropout)         (None, 256)               0

 dense_15 (Dense)            (None, 64)                16448

 dropout_5 (Dropout)         (None, 64)                0

 dense_16 (Dense)            (None, 20)                1300

 dropout_6 (Dropout)         (None, 20)                0

 dense_17 (Dense)            (None, 1)                 21

=================================================================
Total params: 2,276,457
Trainable params: 2,276,457
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 64)                564672

 dropout_2 (Dropout)         (None, 64)                0

 dense_12 (Dense)            (None, 20)                1300

 dropout_3 (Dropout)         (None, 20)                0

 dense_13 (Dense)            (None, 1)                 21

=================================================================
Total params: 565,993
Trainable params: 565,993
Non-trainable params: 0
_________________________________________________________________
```

3. Experimental analysis

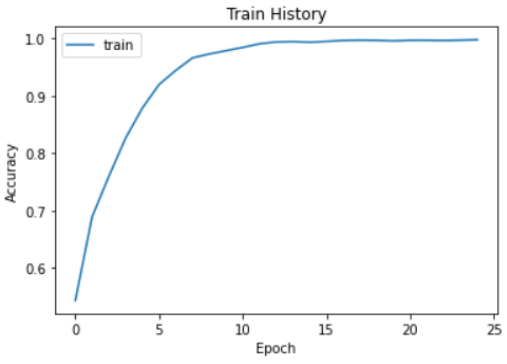
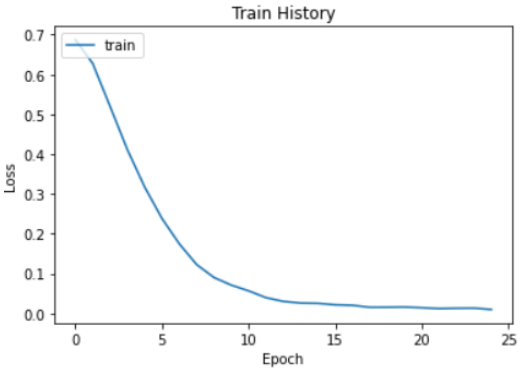
資料集 1、2、3 有分別做不同 epoch 的測試，下面會以圖的方式呈現訓練的 Accuracy 和 Loss 過程。

因為當初在做測試的時候，每次一改數值就直接上傳到平台，沒有特別紀錄哪一個檔案對應的 epoch、layer 數是哪一個，只有把最好的成果更新下來，所以這邊只呈現最後測出來，紀錄最好的模型表現截圖。

Dataset1	
Accuracy	Loss
	
Dataset1 有分別做三層和四層 hidden layer 的 NN，結果為三層的表現好於四層。而 epoch 的部分，一開始設定 epoch=15, accuracy: 0.9953，發現 accuracy 可以更好，進而增加至 epoch=20, accuracy: 0.9971。	

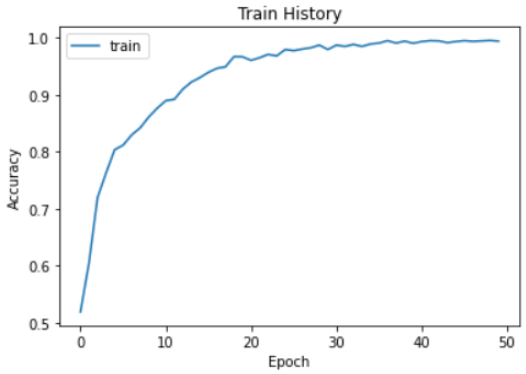
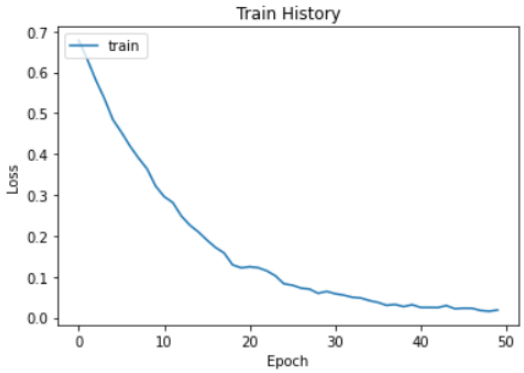
最終預測結果:

	AUC	AP
Dataset1	0.7543	0.8

Dataset2	
Accuracy	Loss
	
Dataset2 有分別做三層和二層 hidden layer 的 NN，結果為三層的表現好於二層。而 epoch 的部分，一開始設定 epoch=15, accuracy: 0.9971，發現 accuracy 可以更好，進而增加至 epoch=25, accuracy: 0.9984。	

最終預測結果:

	AUC	AP
Dataset2	0.8383	0.8641

<table>
<tr><td colspan="2">Dataset3</td></tr>
<tr><td>Accuracy</td><td>Loss</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Dataset3 有分別做三層和二層 hidden layer 的 NN，結果為二層的表現好於三層。而 epoch 的部分，一開始設定 epoch=15, accuracy: 0.9288，發現 accuracy 可以更好，進而增加至 epoch=50, accuracy: 0.9946。</td></tr>
</table>

最終預測結果:

	AUC	AP
Dataset3	0.8849	0.9058

實驗過程中最有心得的是資料集三，是我模型所有表現中最好的資料集，AP 有達到 0.9，其中差異比較大的地方是，一開始我的模型並沒有加入 dropout rate，後來加入之後，尤其資料集三的 nodes 數比較小，效果有顯著的提升，資料集一和二頂多提升到 0.8 左右，但資料集三可以達到 0.9，我覺得資料數大小是影響加入 dropout rate 很大的因素。

另外，我在建模型一開始，建立特徵向量的時候，把特徵當成是有方向性的，做出來預測結果很低，大概 0.2、0.3 左右而已

7	2023-04-10 01:23:24	result1_set3.csv	0.1743	0.3596
6	2023-04-10 01:04:47	result1_set2.csv	0.3114	0.4244
5	2023-04-10 01:04:10	result2_set1.csv	0.2393	0.3871

後來發現是這個問題，重新把程式碼做改善，可以把成果提高到 0.8 以上我覺得是這個作業裡學到最多並且是非常大的進步。

4. Conclusions

在這次作業裡我使用 autoencoder 加上 NN 來做 link prediction，會有使用 autoencoder 這個想法是因為上課時看到情緒分析章節，老師有提到這

也可以是一種嘗試，雖然我使用的方法是非監督式學習，但是預測結果出來並沒有到很差，我覺得還是可行的，只是後來有嘗試要改一些數值來提升效果，發現改善的幅度並不會到很大。

所以如果想要把三個資料集的預測結果都提升到 0.9 以上，我覺得還是需要使用監督式學習的方法，再建立一個模型看看，未來如果有時間可能會嘗試使用 SVM 來做 link prediction。

5. Citations

報告中參考以下連結網頁中的方法下去做改良。
https://medium.com/@fredericklee_73485/%E8%B3%87%E6%96%99%E4%BB%8B%E7%B4%B9-f6609b4f7924

最後程式碼依照資料集分為:
HW2_R76114026_dataset1.ipynb
HW2_R76114026_dataset2.ipynb
HW2_R76114026_dataset3.ipynb