

Machine Learning HW6

B04705003 資工三 林子雋

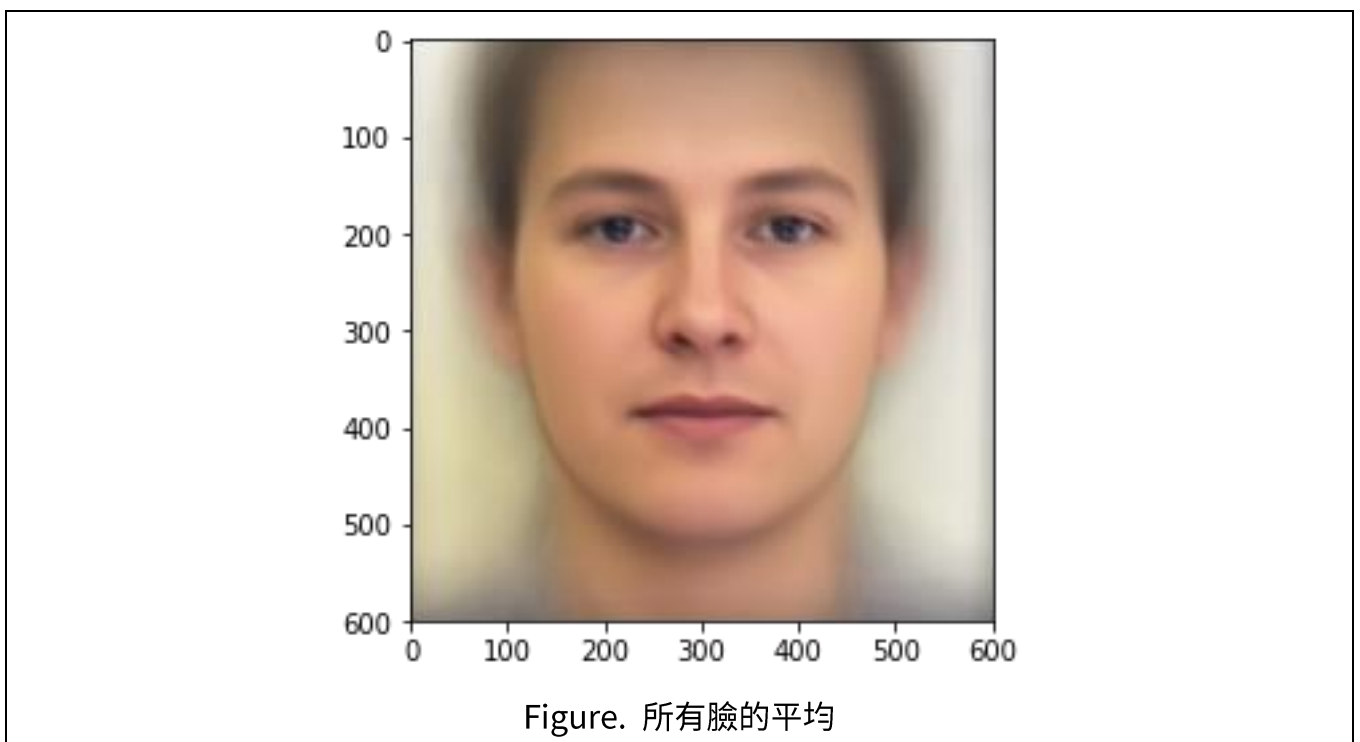
Collaborator :

資工三 陳弘梵 提供 autoencoder 要加 activation 的提醒

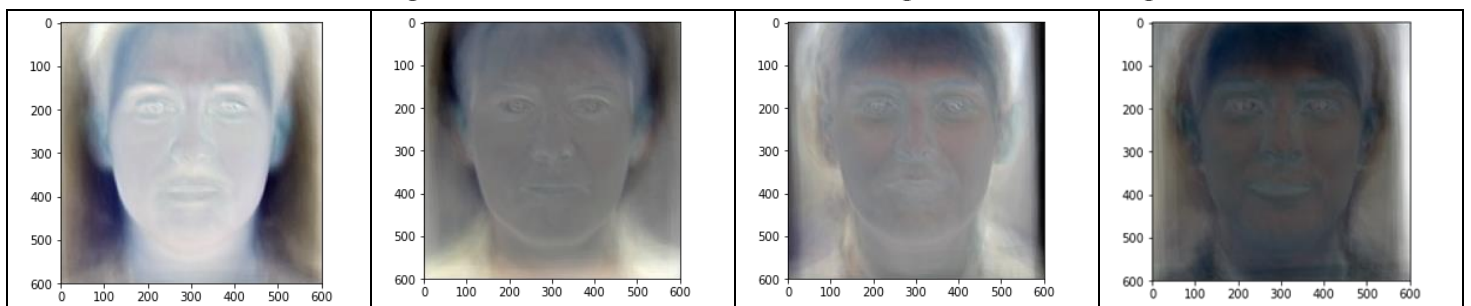
楊耀程 提供簡單 DNN 就可以有不錯 performance 的建議

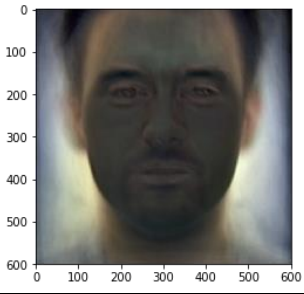
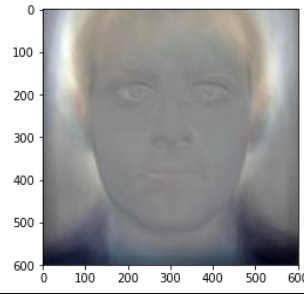
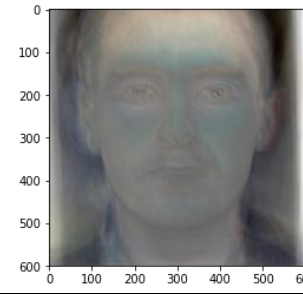
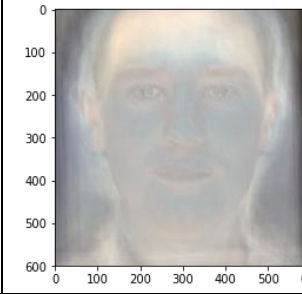
A. PCA of colored faces

1. (.5%) 請畫出所有臉的平均。

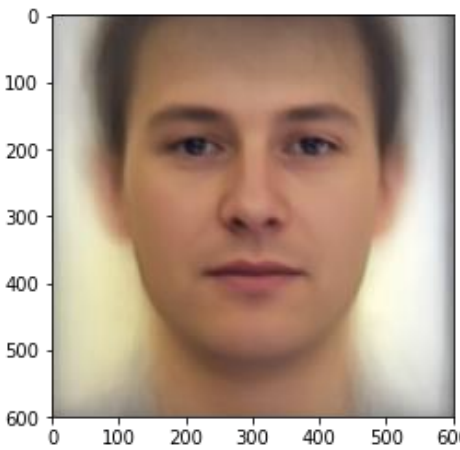
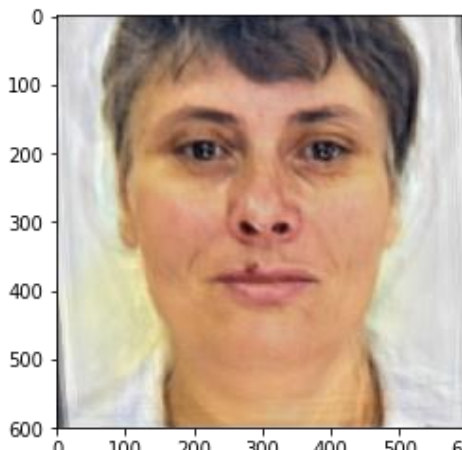
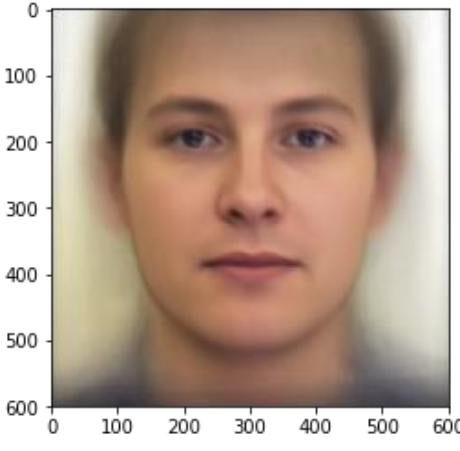
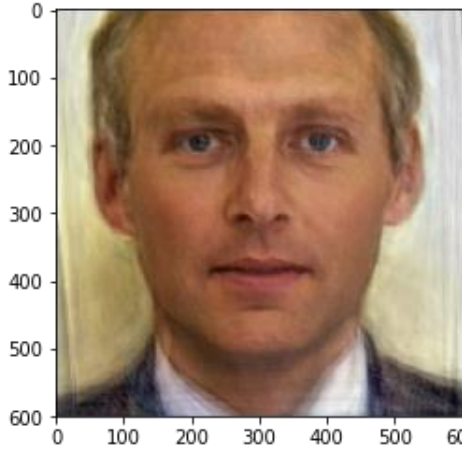


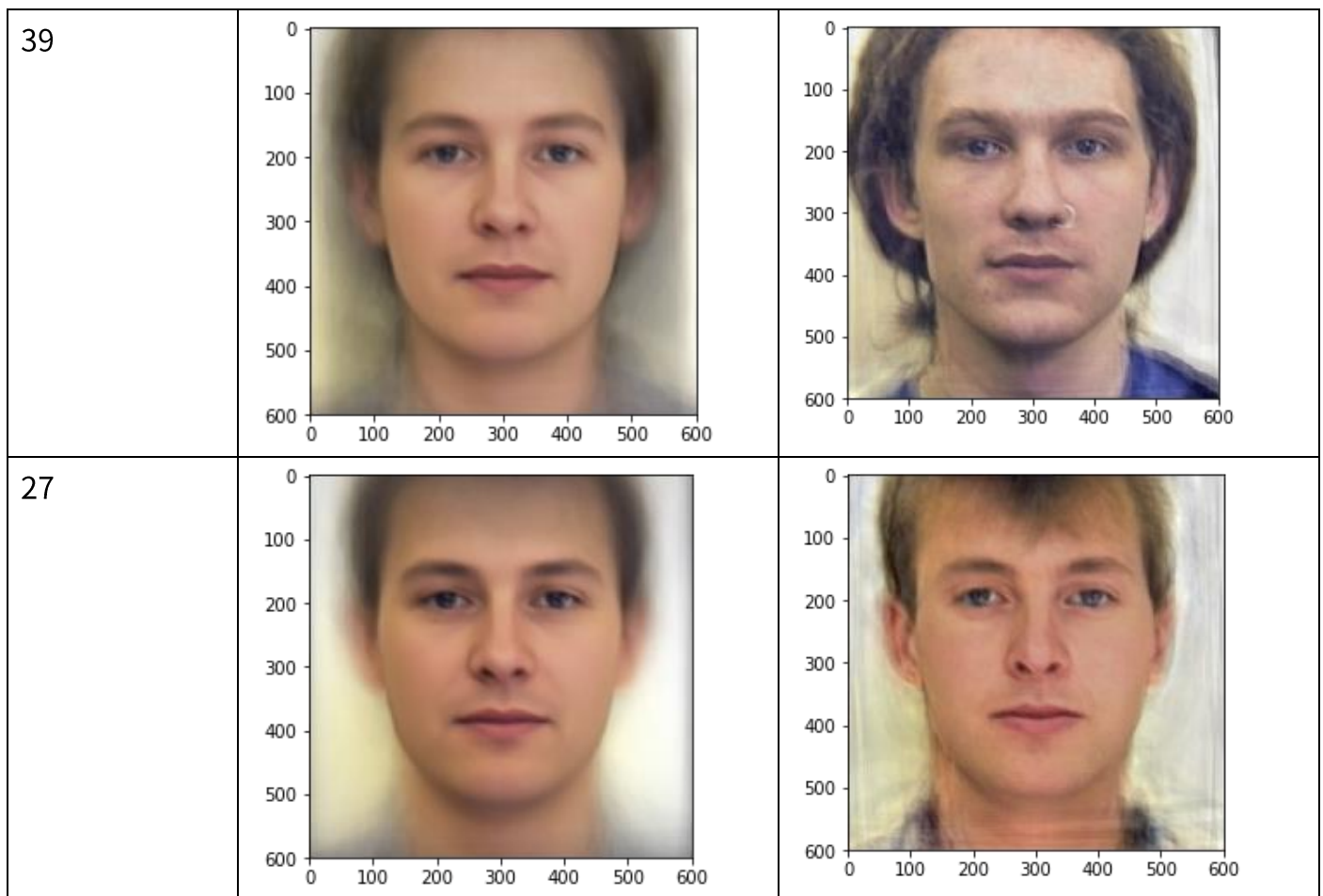
2. (.5%) 請畫出前四個 Eigenfaces，也就是對應到前四大 Eigenvalues 的 Eigenvectors。



			
Figure. 第一 eigenface	Figure. 第二 eigenface	Figure. 第三 eigenface	Figure. 第四 eigenface
備註：因為差一個負號的關係，我將 vector 原本的樣子和乘以負一的樣子都畫了出來			

3. (.5%) 請從數據集中挑出任意四個圖片，並用前四大 Eigenfaces 進行 reconstruction，並畫出結果。

Index	用前四 Eigenvector 重建	用 100 個 Eigenvector 重建
50		
77		



4. (.5%) 請寫出前四大 Eigenfaces 各自所佔的比重 (explained variance ratio)，請四捨五入到小數點後一位。

	比重
λ_1	4.1%
λ_2	2.9%
λ_3	2.4%
λ_4	2.2%

B. Visualization of Chinese word embedding

1. (.5%) 請說明你用哪一個 word2vec 套件，並針對你有調整的參數說明那個參數的意義。

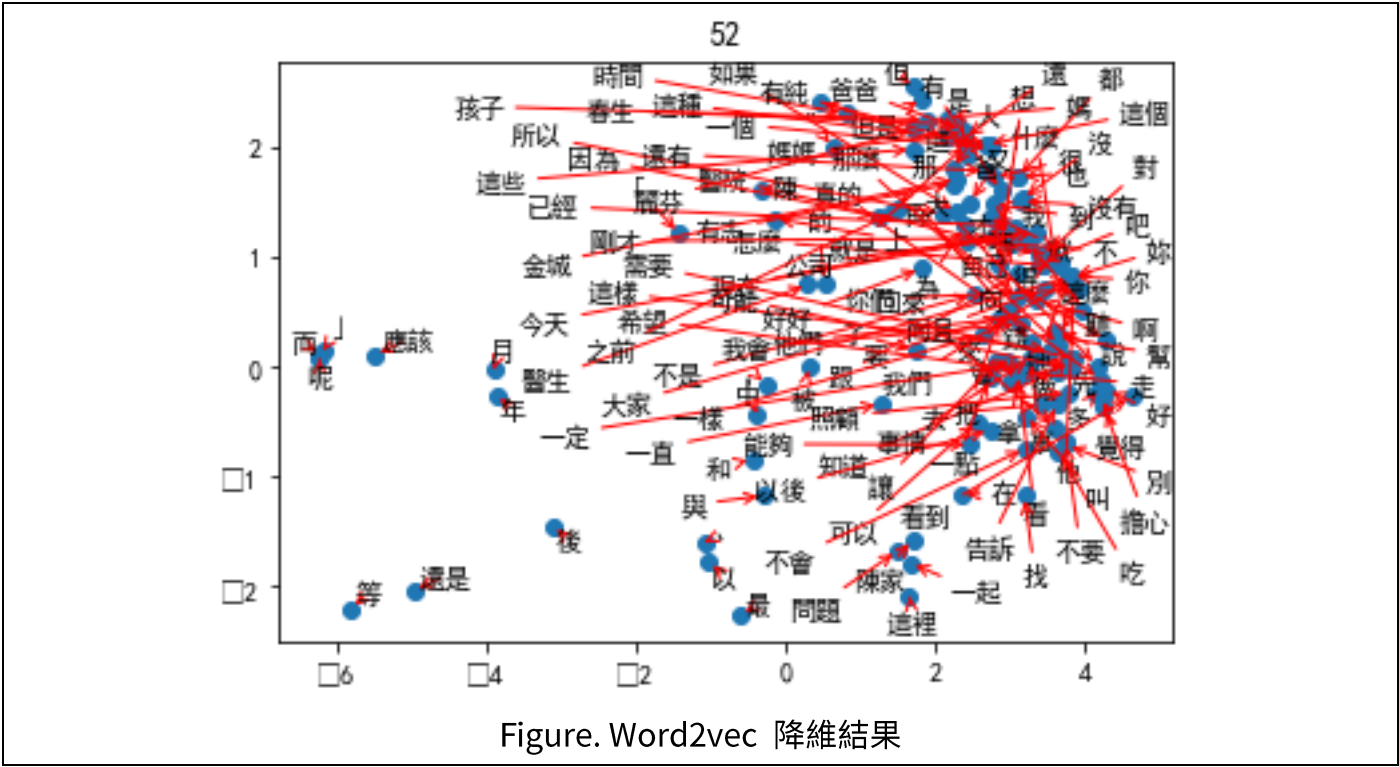
我使用 gensim 的 word2vec 來訓練，調整的參數如下：

```
In [*]: model = gensim.models.Word2Vec(sentences, size=300, window=5, min_count=5, workers=20, iter=50)
```

Size 代表是每一個詞的維度，window 代表是每次在訓練的時候要在多少長度的框格內做 training pair 的 sample，min_count 代表這個詞至少要出現幾遍才把他放入字典當中，workers 是指要使

用幾顆 CPU，iter 代表要訓練多少次。

2. (.5%) 請在 Report 上放上你 visualization 的結果。



3. (.5%) 請討論你從 visualization 的結果觀察到什麼。

觀察：
發現到在左半邊中，有一些相近的字詞的確有靠在一起，譬如說時間性的詞：「月、年、今天」。除此之外，許多比較明顯的動詞也有靠在一起的趨勢，例如：「去、把、拿、做」。也發現到「爸爸—媽媽」的向量和「爸—媽」的向量平行，顯示這兩組的詞相減的話有類似的意思。

C. Image clustering

1. (.5%) 請比較至少兩種不同的 feature extraction 及其結果。(不同的降維方法或不同的 cluster 方法都可以算是不同的方法)

方法	方法細節	Leaderboard Score	分析
PCA dim 2 + Kmeans	先把 784 維用 PCA reduce 到 2 維，再用 Kmeans 去分群	0.02992	事後做 inverse transform 發現圖片幾乎已經不能重現，所以猜測是用 PCA 降維資訊損失太多

PCA dim 300 + Kmeans	先把 784 維用 PCA 降到 300 維，再用 Kmeans 分群	0.03022	較 2 維好一些
CNN dim 360 + Kmeans	先把 784 維用 CNN Autoencoder 降維到 360，再用 Kmeans	0.03034	雖然有發現 autoencoder reconstruct 的圖片做的蠻好的，但結果還是很差，推測可能是因為 maxpooling 造成的資訊流失
PCA dim 300 + tsne dim 2 + Kmeans	PCA 降維到 300 再用 tsne 降維到 2 再用 Kmeans	0.03222	比上面都在好一些，推測是因為 tsne 的降維方法很穩定
DNN dim 32 + Kmeans	DNN 降維到 32 再用 Kmeans 分類。 沒加 activation	0.02015	因為 pytorch nn.Linear 預測 bias 是 uniform distribution，因此造成 autoencoder 不好訓練
DNN dim 32 + tsne 2 + Kmeans	DNN 降維到 32 再用 tsne 降維到 2 再用 Kmeans 分類	0.33804	實際肉眼看 visual 的圖片，發現真的資料有分成兩群，推測是因為 tsne 降維到 2 會讓 Kmeans 的分類比較穩定。
DNN dim 32 + Kmeans	DNN 降到 32 維再用 Kmeans 分類(有 RELU activation)	0.99527	經過 normalize 之後，效果顯著

2. (.5%) 預測 visualization.npy 中的 label，在二維平面上視覺化 label 的分佈。

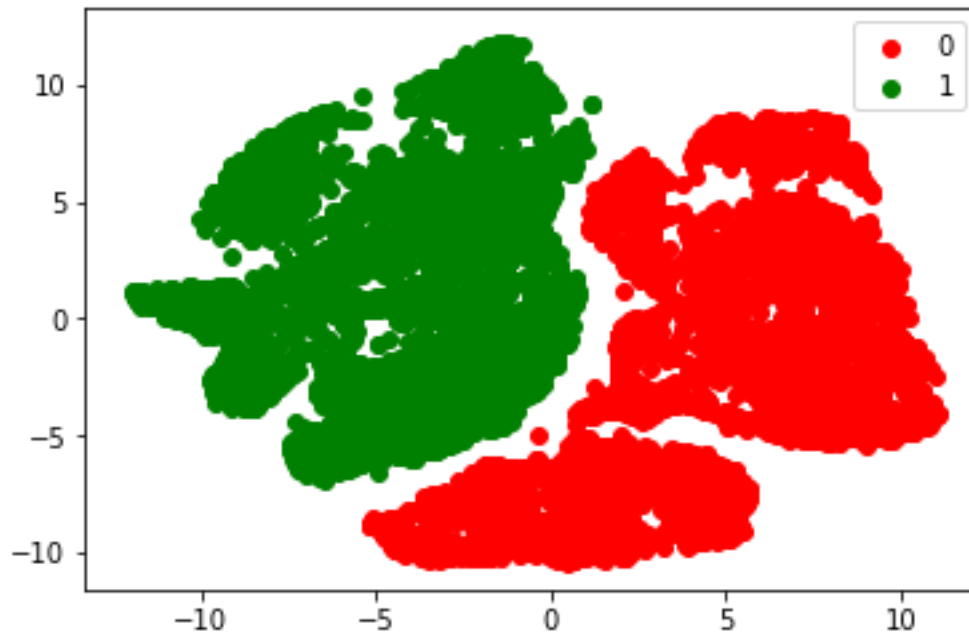


Figure. 預測出的 label

3. (.5%) visualization.npy 中前 5000 個 images 跟後 5000 個 images 來自不同 dataset。
請根據這個資訊，在二維平面上視覺化 label 的分佈，接著比較和自己預測的 label 之間有何不同。

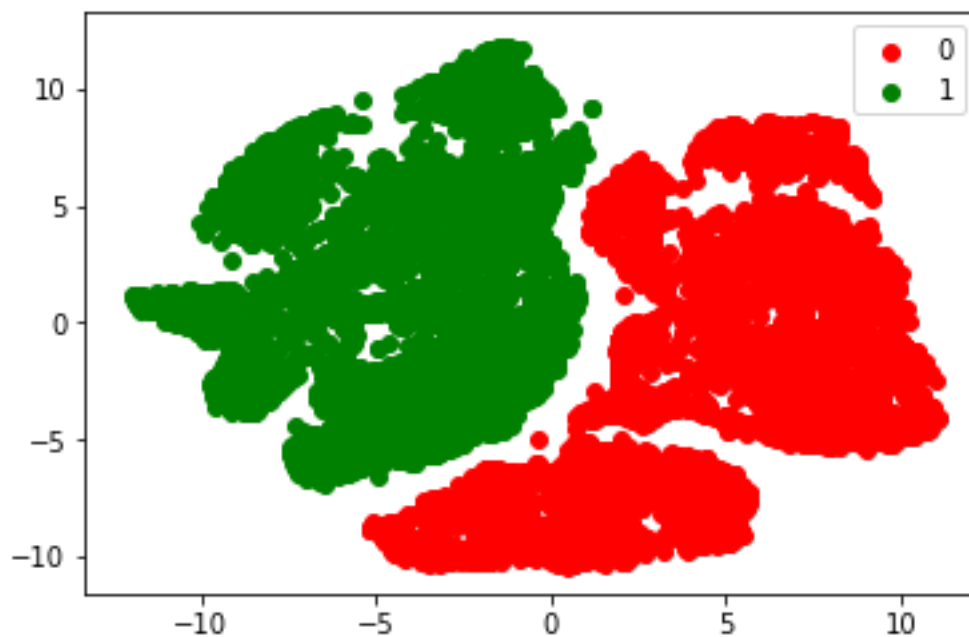


Figure. Ground truth

Precision = 100%

Recall = 99.9%

F1 = 99.9%

結論：發現到預測的結果與實際沒甚麼不同（實際上，只有一筆預測錯）