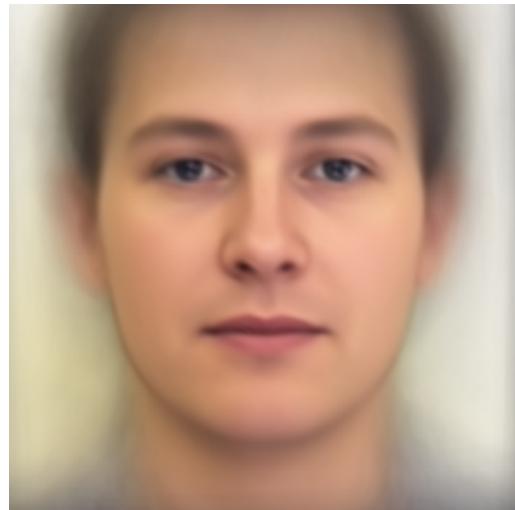


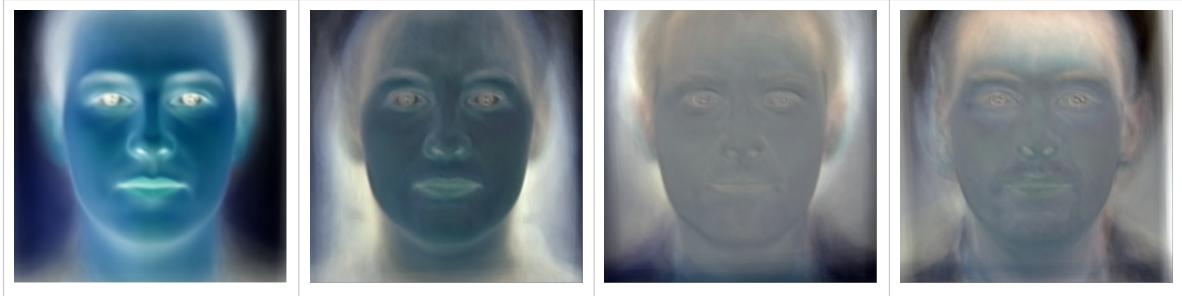
1.(2%) PCA of colored faces

(collaborator: None)

請畫出所有臉的平均。



請畫出前四個 Eigenfaces，也就是對應到前四大 Eigenvalues 的 Eigenvectors。



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。

29	92	102	397

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。

以下是我計算的前四大 Eigenfaces

7.5%	3.1%	2.8%	2.2%
------	------	------	------

## 2.(1.5%) Visualization of Chinese word embedding

(collaborator: None)

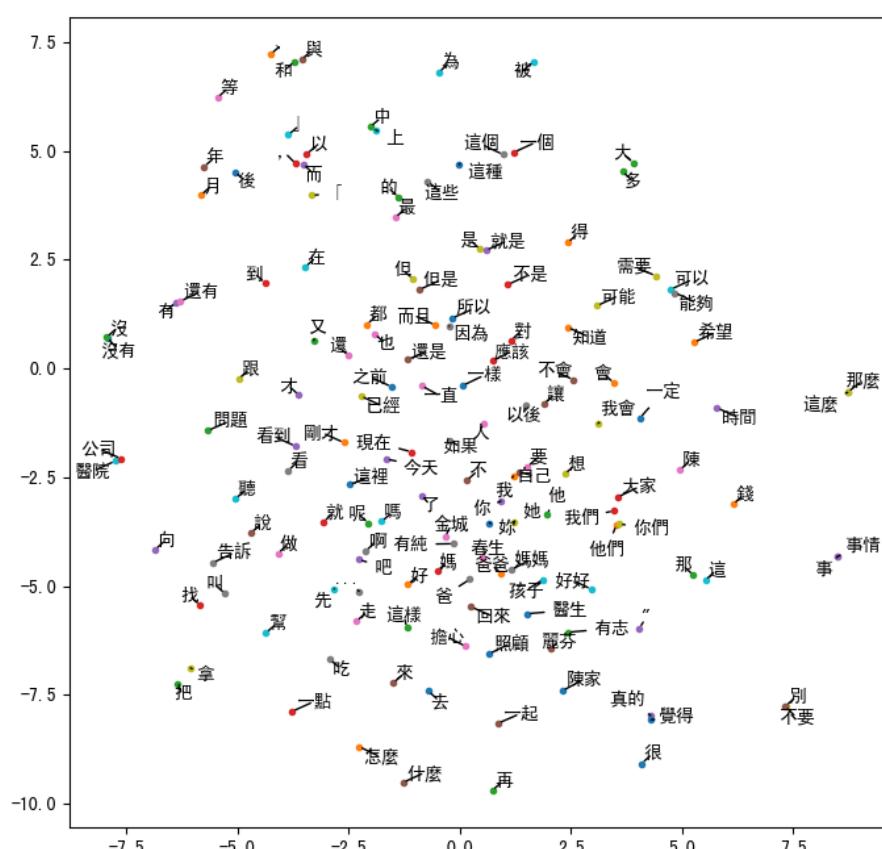
請說明你用哪一個 word2vec 套件，並針對你有調整的參數說明那個參數的意義。

這題我是先使用 jieba 進行斷詞，最後用 gensim 進行 training。以下是我 training 調整的參數。

```
model = word2vec.Word2Vec(sentences, size=200, workers = 6, min_count=10)
```

每個字最後會成為一個 200 維的 vector，另外訓練的字典裡，低於 10 個字的不會一起 training，另外調整workers 數目，讓 training 速度更快。

請在 Report 上放上你 visualization 的結果。



請討論你從 **visualization** 的結果觀察到什麼。

上圖我只畫了 word count 大於 3000 的字，我覺得周圍的文字，訓練還算準確，比如說“別”、“不要”至少是在一起的，而圖表中間就比較混雜，可能可以試試看把 **stopword** 去掉然後訓練看看。

### 3.(1.5%) Image clustering

(collaborator: None)

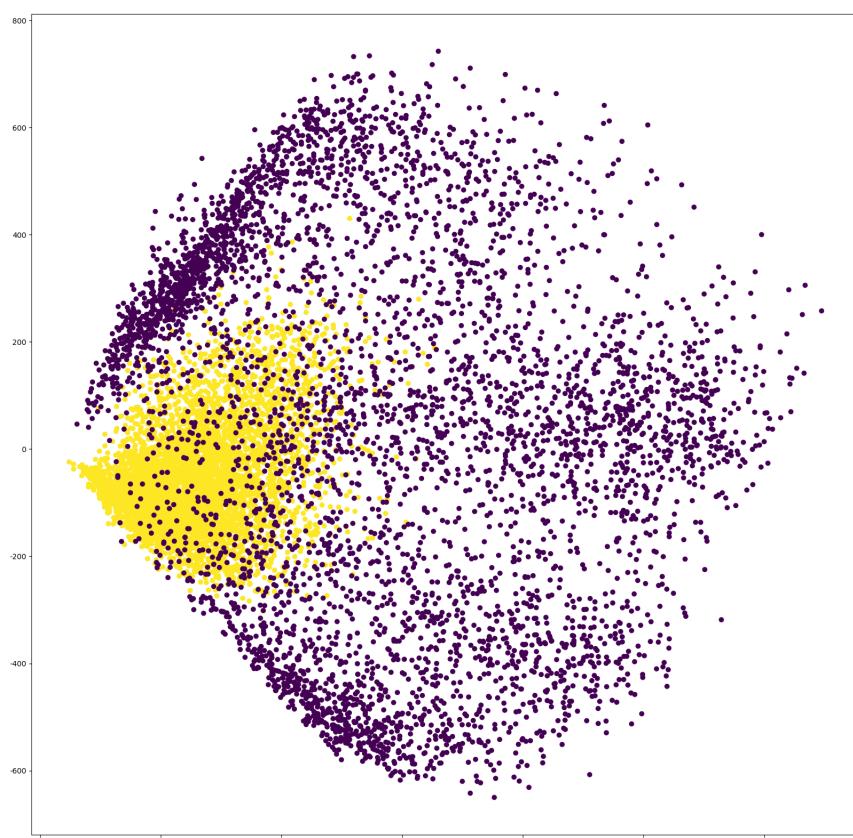
請比較至少兩種不同的 **feature extraction** 及其結果。

我嘗試了兩種方法，第一種是使用助教手把手教學的方法，第二種是直接使用 **sklearn** 的 **PCA** 將 **feature** 降維分群，最後在用 **sklearn** 的 **KMEANS** 做二類分群。

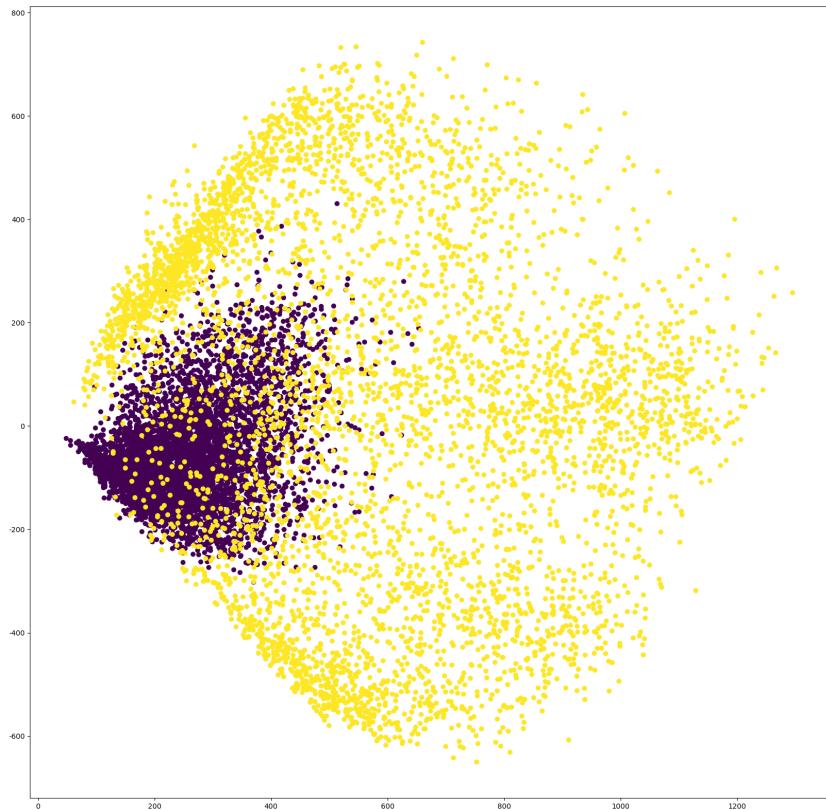
```
pca = PCA(n_components=300, whiten=True, svd_solver='randomized')
pca_features = pca.fit_transform(train_data)
# kmeans
kmeans = KMeans(init='k-means++', n_clusters=2, random_state=0).fit(pca_features)
```

用助教提供的方法，在 **kaggle** 上會有大約 0.7 的分數，而使用 **sklearn** 提供的方法，分數可以來到 0.99，非常精確

預測 **visualization.npy** 中的 **label**，在二維平面上視覺化 **label** 的分佈。



`visualization.npy` 中前 5000 個 `images` 來自 dataset A，後 5000 個 `images` 來 dataset B。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。



使用 `sklearn` 做二類分群這個做法預測的非常準確。可以看到分佈幾乎相等。