Homework #2

403410034 資工四 黃鈺程

Abstract

題目：使用 SVM 來跑 MNIST 與其他 dataset。

我使用 scikit-learn 來實作，因為他內建了搜 hyperparameter 的 `GridSearchCV` 並可以搭配 python 的科學計算環境來方便繪圖，比較等。我最終跑了三個 dataset： `MNIST`， `Wine`， `Cifar10` ，並比較了 `SVC` (SVM Classifier) 與 `RandomForest` 。在 `MNIST` 與 `Wine` 上表現良好的 `SVC` （在 `Wine` 上甚至得到 100%），在 `Cifar10` 上卻不甚理想， `RandomForest` 反而有更佳的結果。

`Wine`：https://archive.ics.uci.edu/ml/datasets/wine (https://archive.ics.uci.edu/ml/datasets/wine)
`Cifar10`：https://www.cs.toronto.edu/~kriz/cifar.html (https://www.cs.toronto.edu/~kriz/cifar.html) (python version)

Environment/Requirement

使用

1. scikit-learn
2. matplotlib, seaborn
3. numpy, pandas
4. jupyter

程式碼可以在 這裡 (https://github.com/amoshyc/ccu-multimedia/tree/master/hw2) 找到。

Result

`SVC` hyperparameter 搜尋的範圍為：

```
params = {
    'kernel': ['rbf', 'linear', 'poly'],
    'C': [1e-2, 1e-1, 1, 10, 100],
    'gamma': [1e-3, 1e-2, 1e-1, 1, 10],
}
```

共有 75 種組合，每種組合再 cross validation，如果手寫程式碼的話，程式碼量並不小，所幸 scikit-learn 有內建 `GridSearchCV` ，我只要寫：

```
clf = GridSearchCV(SVC(), params, cv=5, scoring='accuracy', n_jobs=3)
clf.fit(xs, ys)
```

就可以自動化這個過程，並指定程式同時使用 3 個 cpu core。

另外，如果使用所有 features (784 個) ， svc 會非常非常慢，慢到無法實際使用的情況，所以我先跑了 `PCA` 將資料降到 20 個 features:

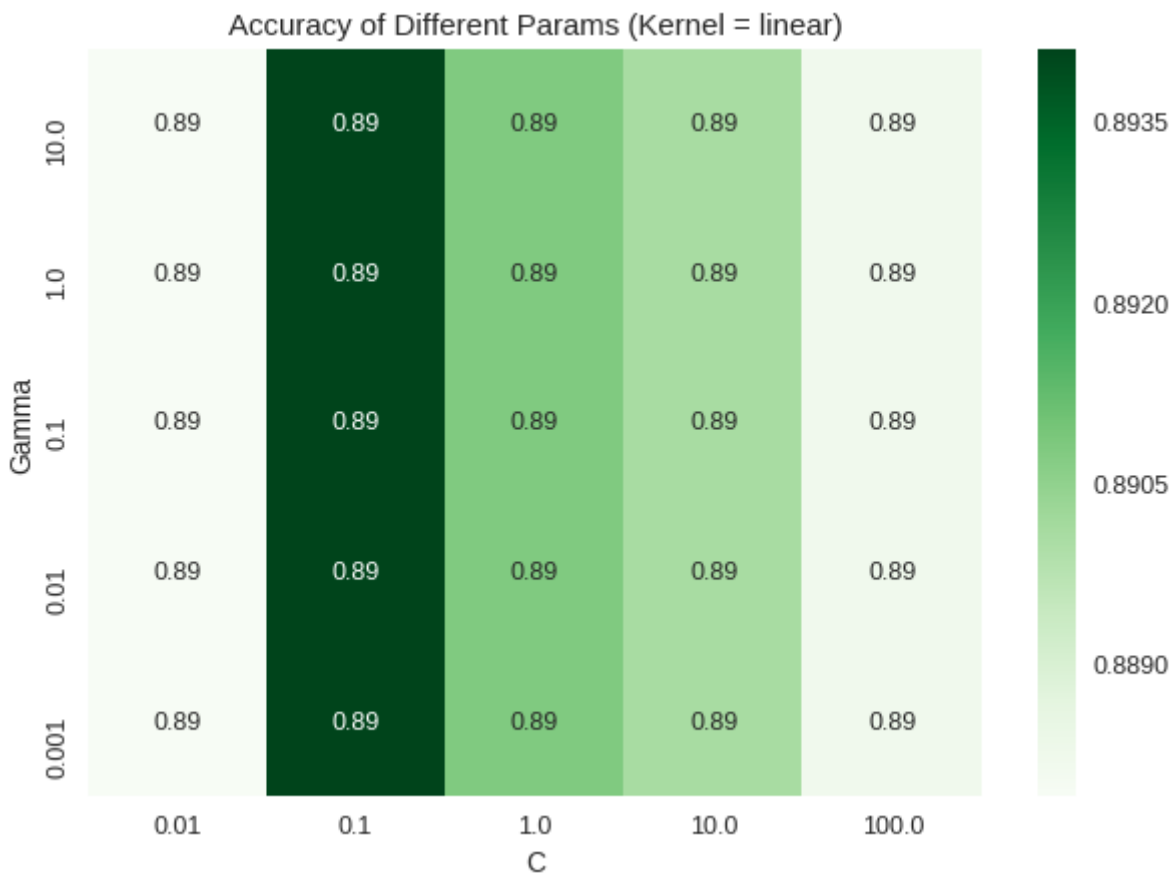
```
pca = PCA(n_components=20)
pca.fit(xt)
xt = pca.transform(xt)
xv = pca.transform(xv)
```

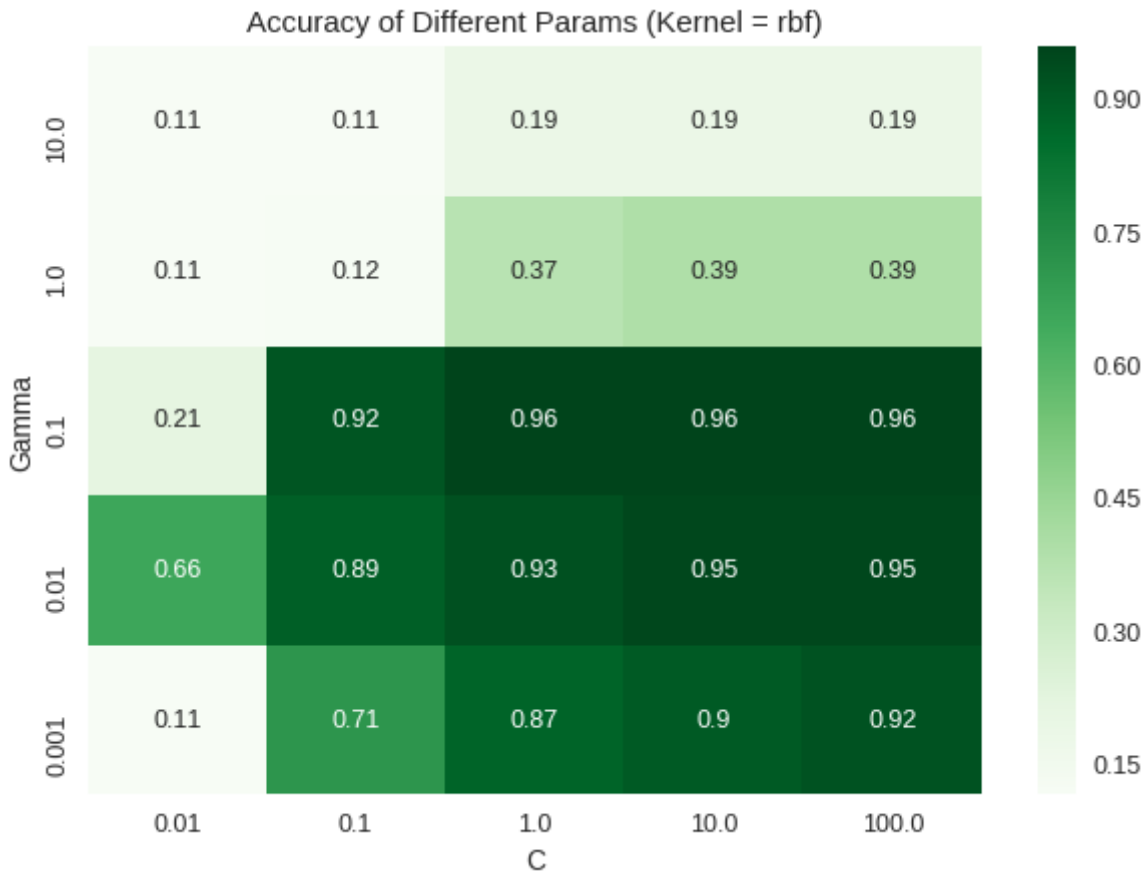
各 Dataset 的結果如下，我使用 `RandomForest(n_estimators=20)` 作為 Baseline 來比較。

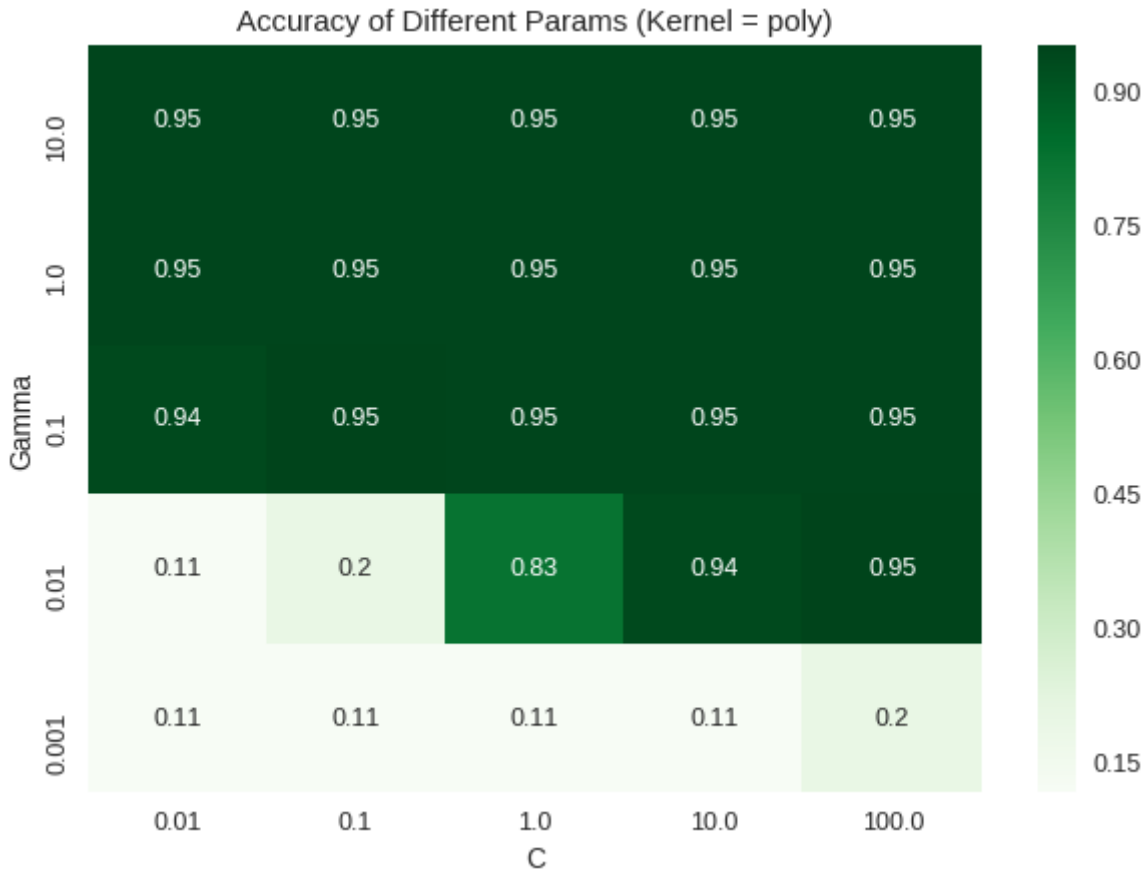
MNIST

Metrics 是 Accuracy。

GridSearchCV







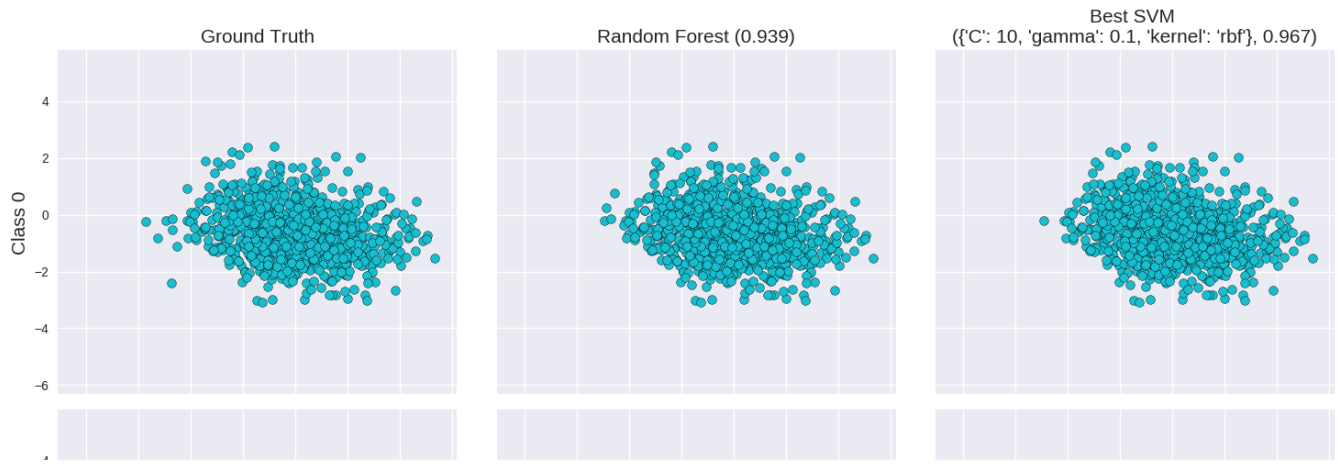
Metrics on Each Class

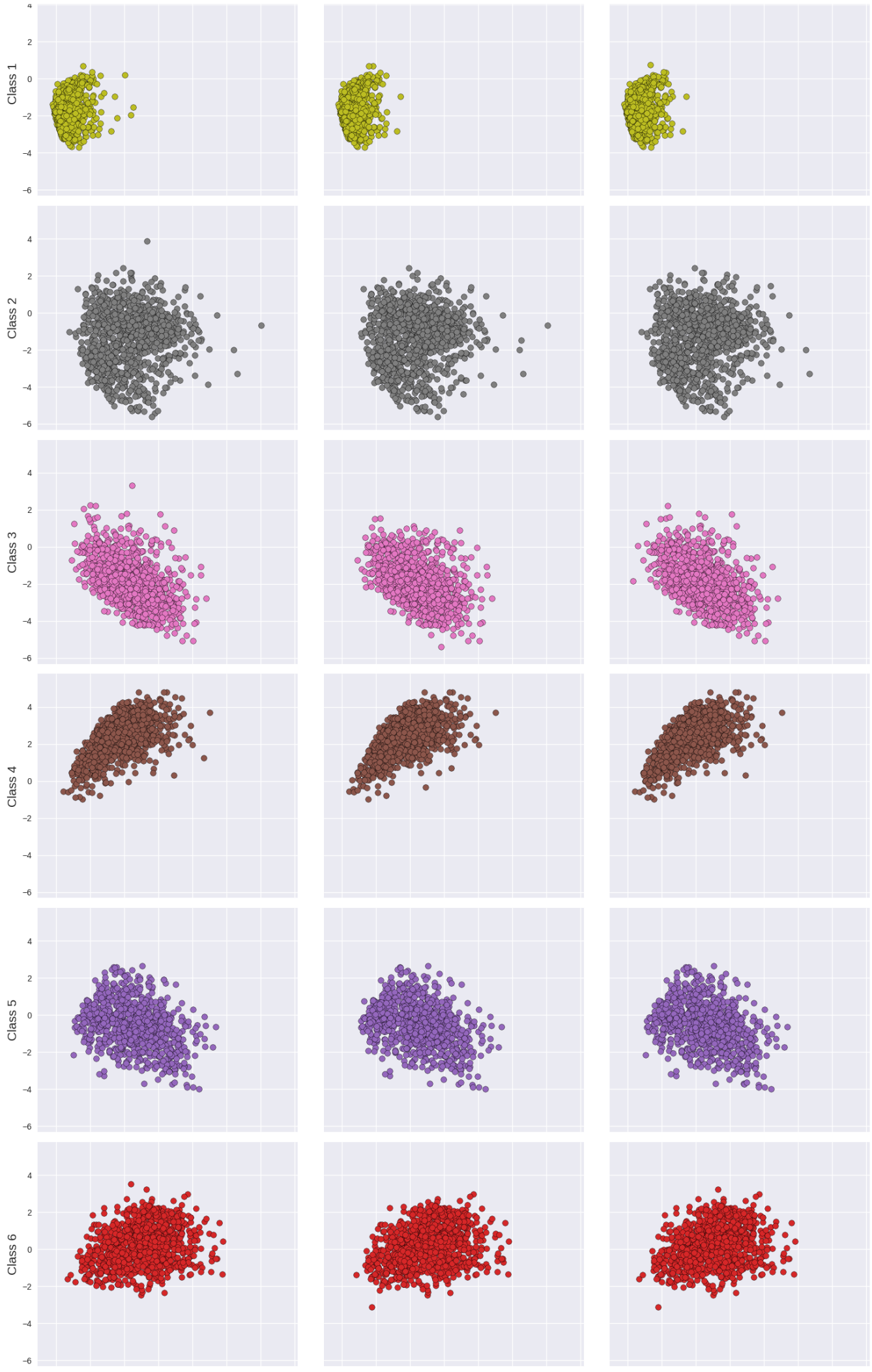
```
Best SVC:
             precision    recall  f1-score   support

          0       0.97      0.99      0.98       980
          1       0.99      0.99      0.99      1135
          2       0.95      0.97      0.96      1032
          3       0.96      0.96      0.96      1010
          4       0.97      0.97      0.97       982
          5       0.96      0.96      0.96       892
          6       0.99      0.97      0.98       958
          7       0.96      0.95      0.96      1028
          8       0.96      0.95      0.96       974
          9       0.96      0.95      0.95      1009

avg / total       0.97      0.97      0.97     10000


Random Forest:
             precision    recall  f1-score   support

          0       0.96      0.98      0.97       980
          1       0.98      0.99      0.99      1135
          2       0.93      0.93      0.93      1032
          3       0.91      0.94      0.93      1010
          4       0.91      0.93      0.92       982
          5       0.93      0.91      0.92       892
          6       0.95      0.97      0.96       958
          7       0.96      0.93      0.95      1028
          8       0.92      0.90      0.91       974
          9       0.92      0.91      0.91      1009

avg / total       0.94      0.94      0.94     10000
```

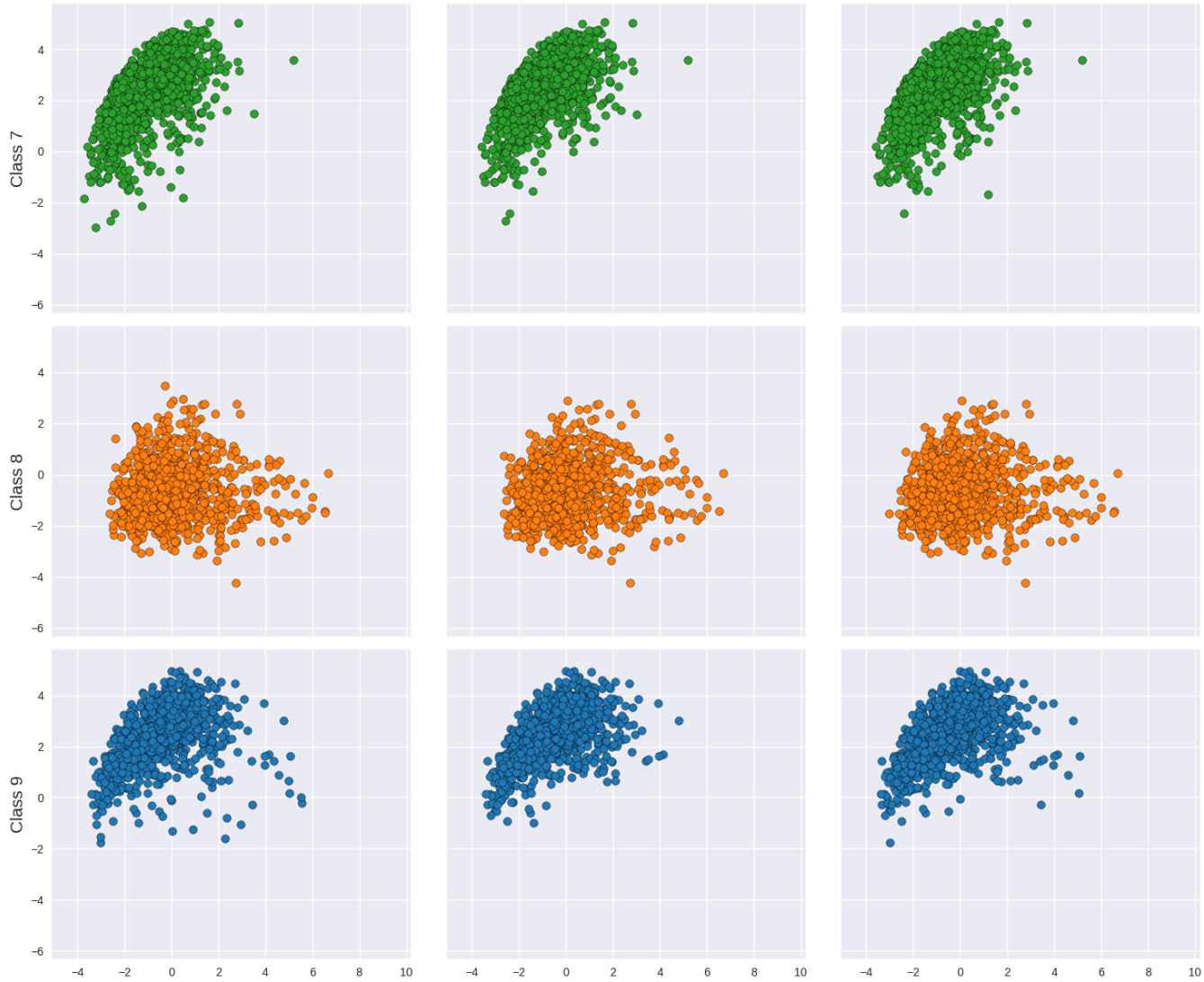
可以看到 `SVC` 各 class 的值都差不多，沒有說有哪一個 class 特別好或特別不好預測。但 `RandomForest` 就有偏斜的情況，數字 `1` 特別好而數字 `9` 特別差。

Visualization

我使用 `PCA` 後的資料的前二維來可視化，結果如下:



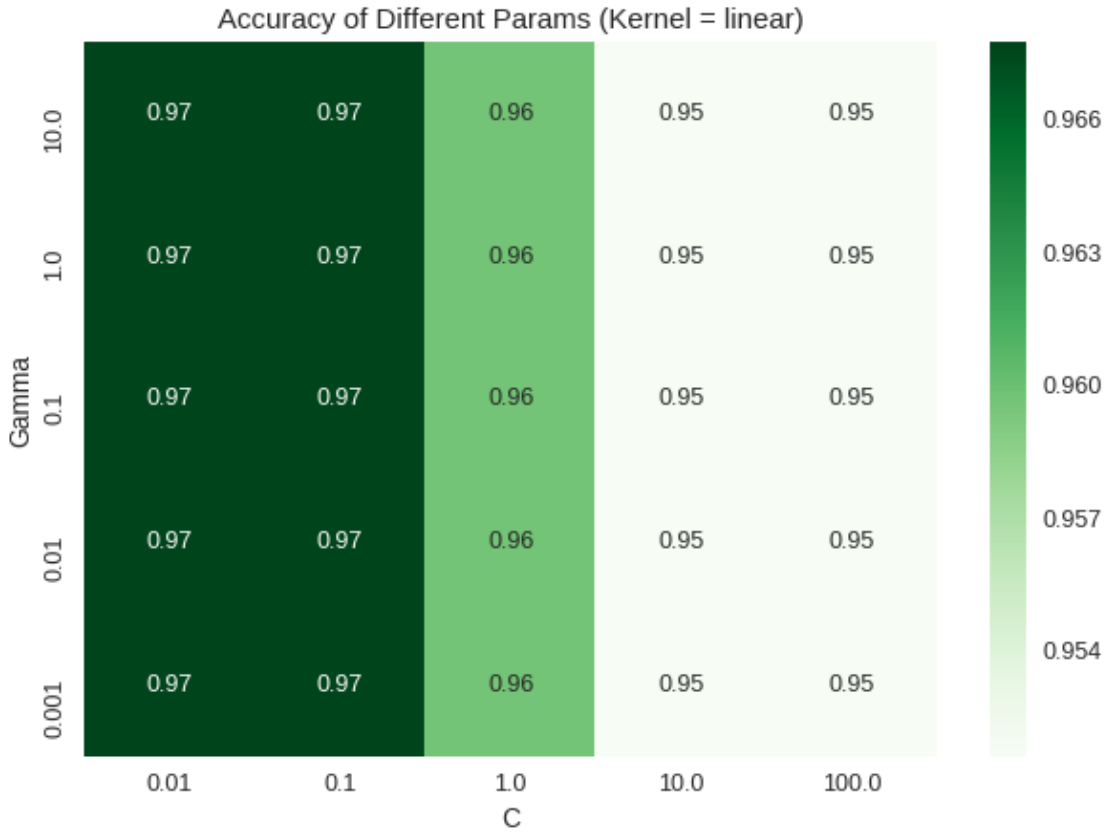

Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
4
2
0
−2
−4
−6

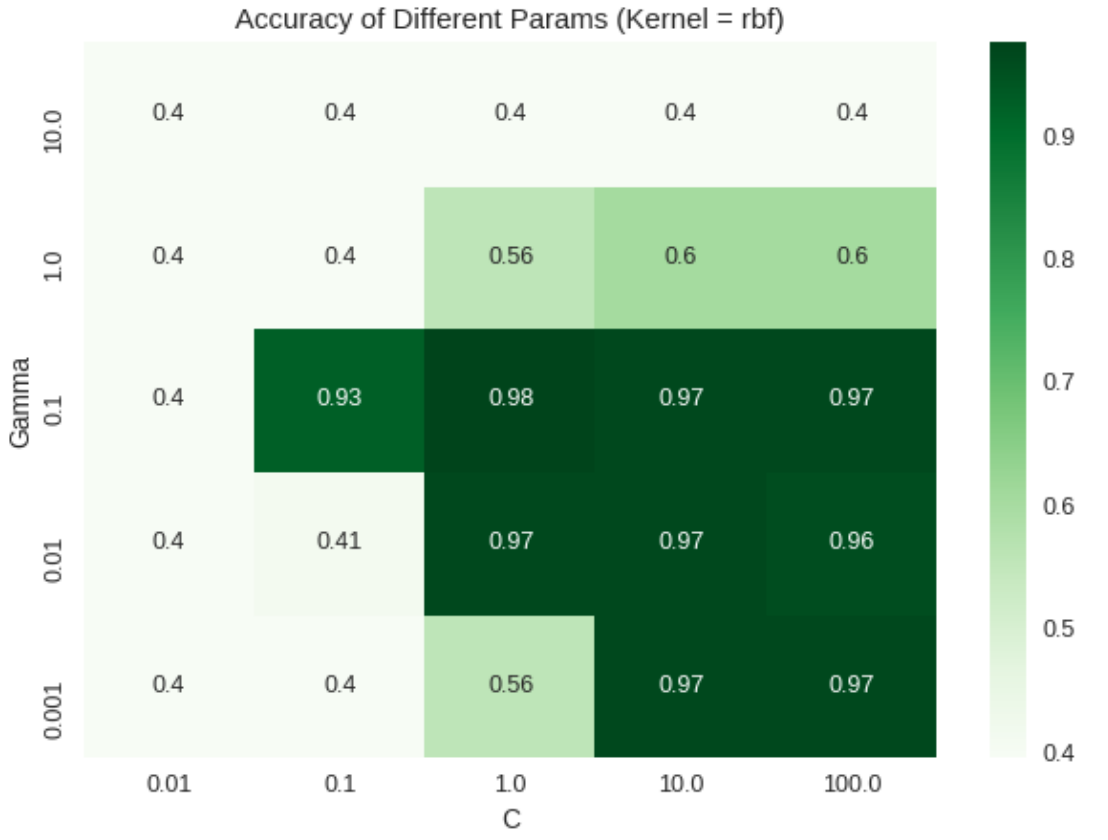


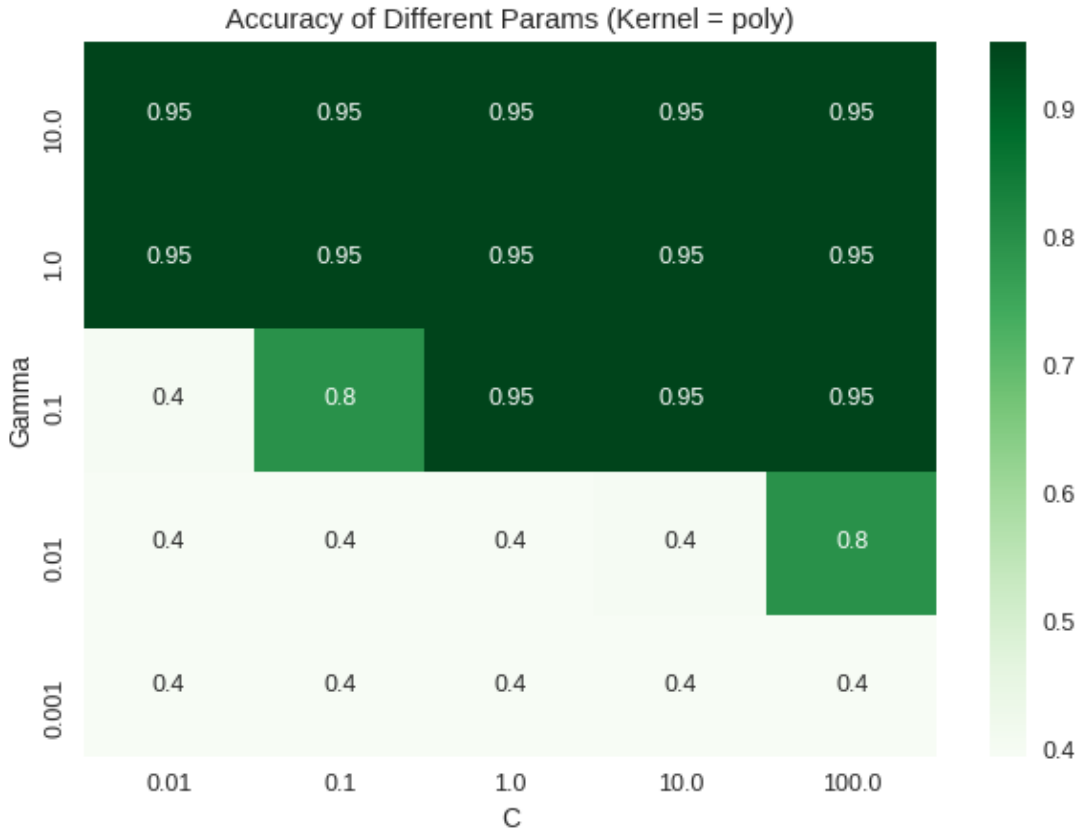
Wine

Wine 是一個小型的 Dataset，資料量只有 178 筆，每筆 13 個 features，共 3 種 class。我使用 124 筆作為 training data ，54 筆做為 validation data。最後竟然做出 1.0 的 accuracy，這讓我非常高興~可視化的結果也顯示出，資料 PCA 後似乎就是線性可分了。

GridSearchCV


Accuracy of Different Params (Kernel = linear)
Gamma
10.0
1.0
0.1
0.01
0.001
0.97 0.97 0.96 0.95 0.95
0.97 0.97 0.96 0.95 0.95
0.97 0.97 0.96 0.95 0.95
0.97 0.97 0.96 0.95 0.95
0.97 0.97 0.96 0.95 0.95
0.01 0.1 1.0 10.0 100.0
C
0.966
0.963
0.960
0.957
0.954


Accuracy of Different Params (Kernel = rbf)
Gamma
10.0
1.0
0.1
0.01
0.001
0.4 0.4 0.4 0.4 0.4
0.4 0.4 0.56 0.6 0.6
0.4 0.93 0.98 0.97 0.97
0.4 0.41 0.97 0.97 0.96
0.4 0.4 0.56 0.97 0.97
0.01 0.1 1.0 10.0 100.0
C
0.9
0.8
0.7
0.6
0.5
0.4

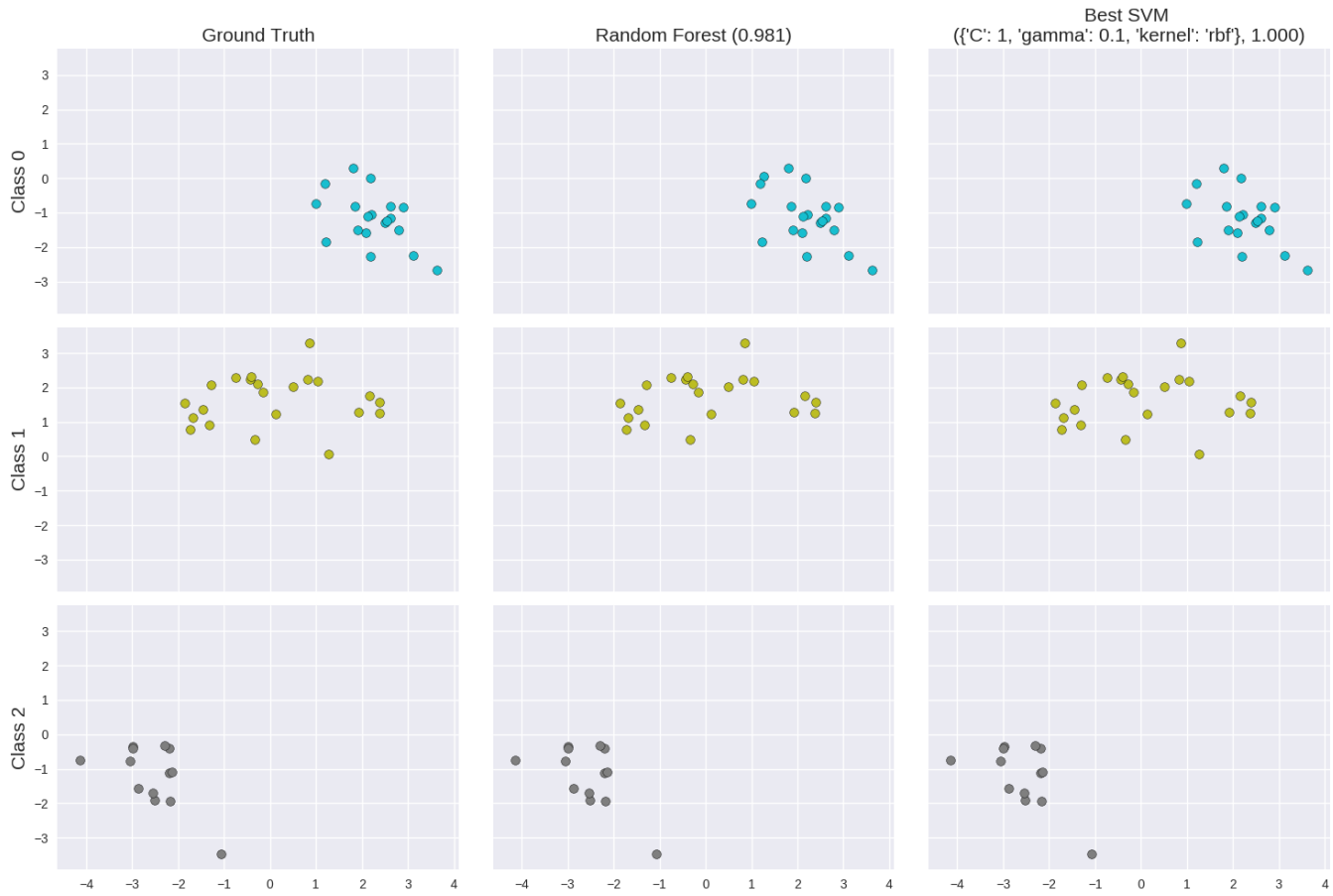


Metrics on Each Class

```
Best SVC:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        19
          1       1.00      1.00      1.00        22
          2       1.00      1.00      1.00        13

avg / total       1.00      1.00      1.00        54


Random Forest:
             precision    recall  f1-score   support

          0       0.95      1.00      0.97        19
          1       1.00      0.95      0.98        22
          2       1.00      1.00      1.00        13

avg / total       0.98      0.98      0.98        54
```

看到 Best SVC 中那一排的 1.0，就讓人感到高興~

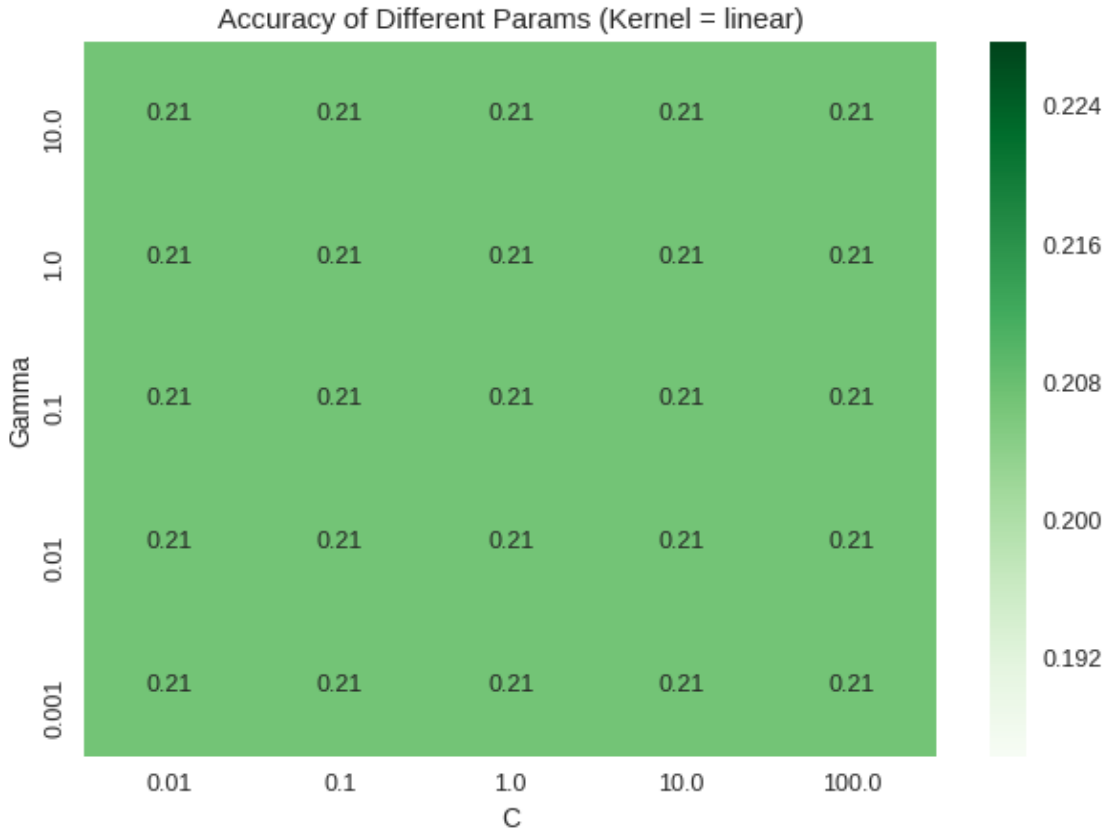
Visualization

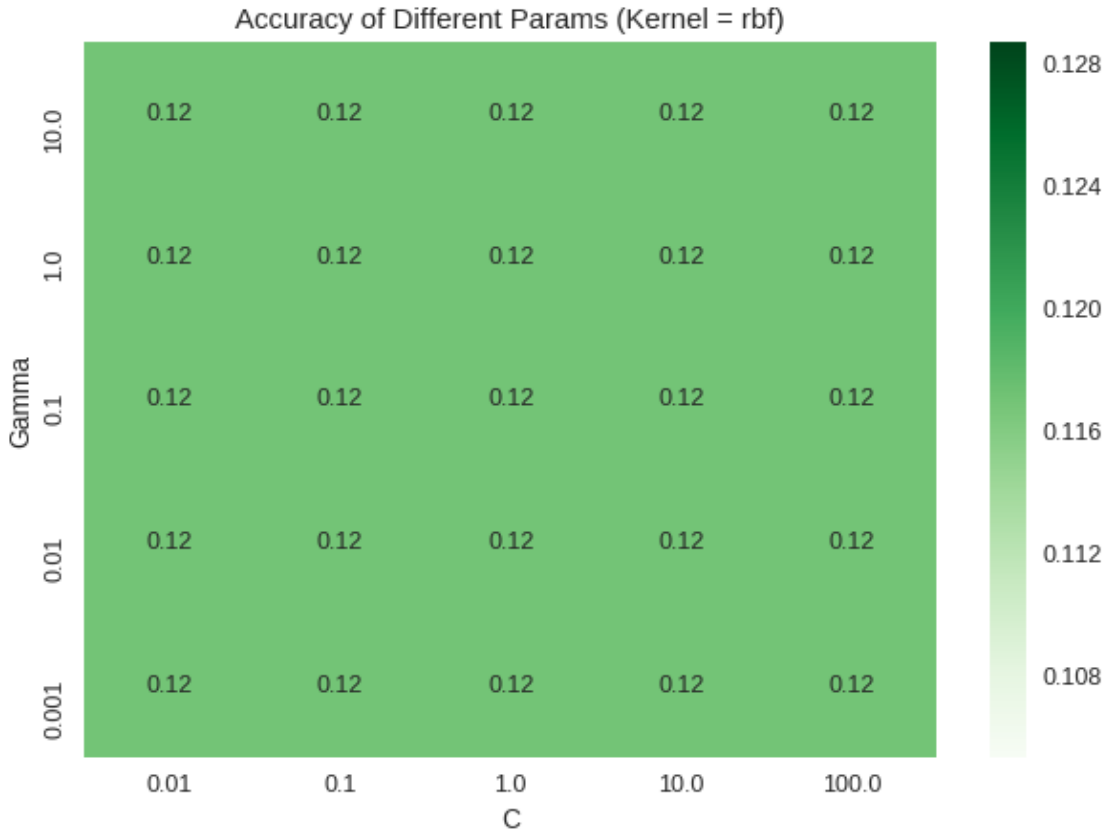


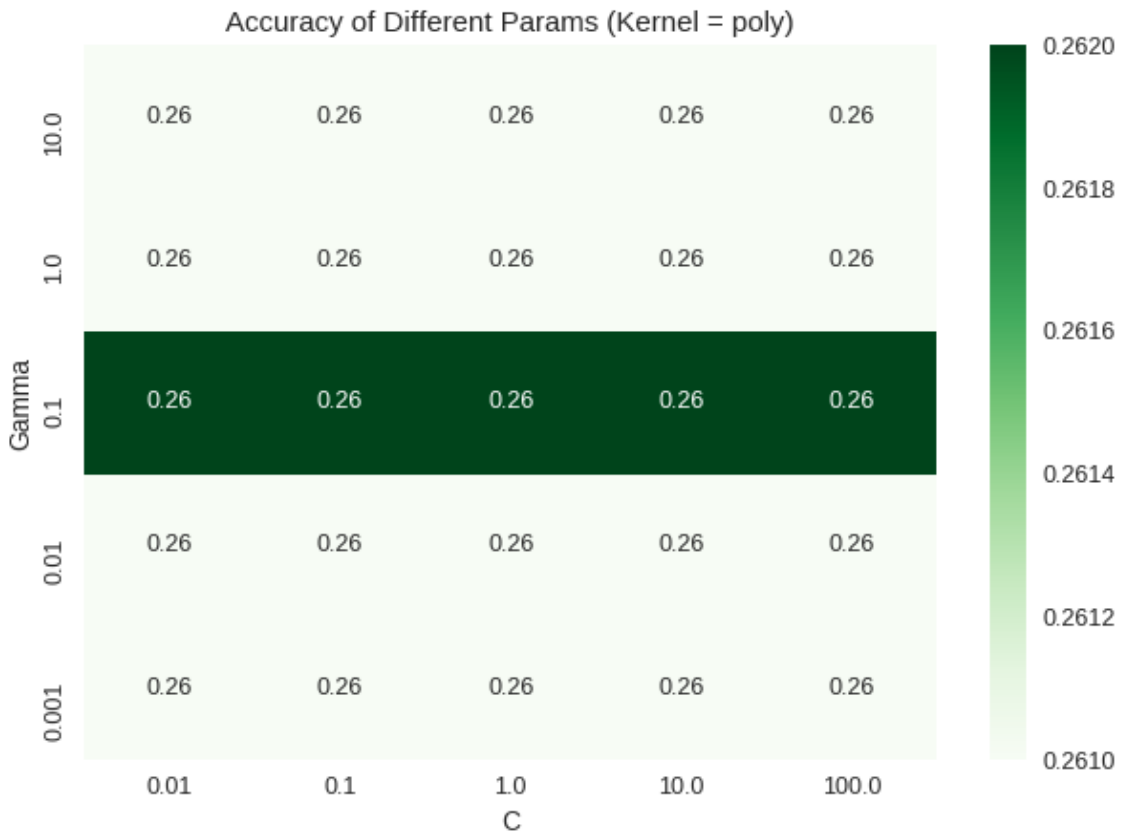
Cifar10

Cifar10 是 Cifar100 的子集，共有 50000 筆資料，每筆資料 3072 個 features，分成 10 個 class。我使用 40000 個作為 training data，剩下 10000 個作為 validation。

GridSearchCV


Accuracy of Different Params (Kernel = linear)
Gamma
10.0
1.0
0.1
0.01
0.001
0.21
0.01
0.1
1.0
10.0
100.0
C
0.224
0.216
0.208
0.200
0.192


Accuracy of Different Params (Kernel = rbf)
Gamma
10.0
1.0
0.1
0.01
0.001
0.12
0.01
0.1
1.0
10.0
100.0
C
0.128
0.124
0.120
0.116
0.112
0.108



非常慘，想不到 SVC 在 cifar10 上表現這麼差，而且不管是什麼參數都沒用。

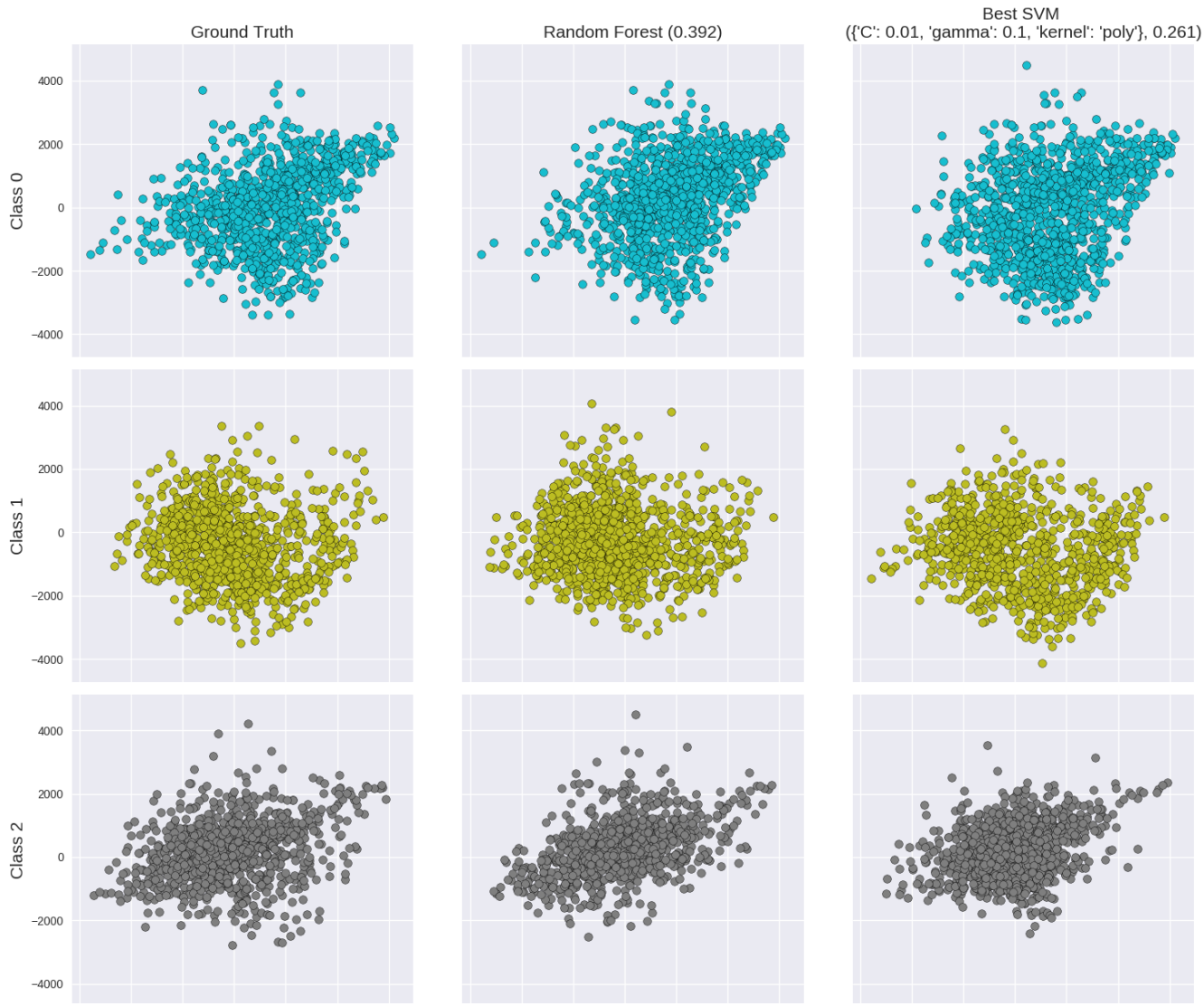
Metrics on Each Class

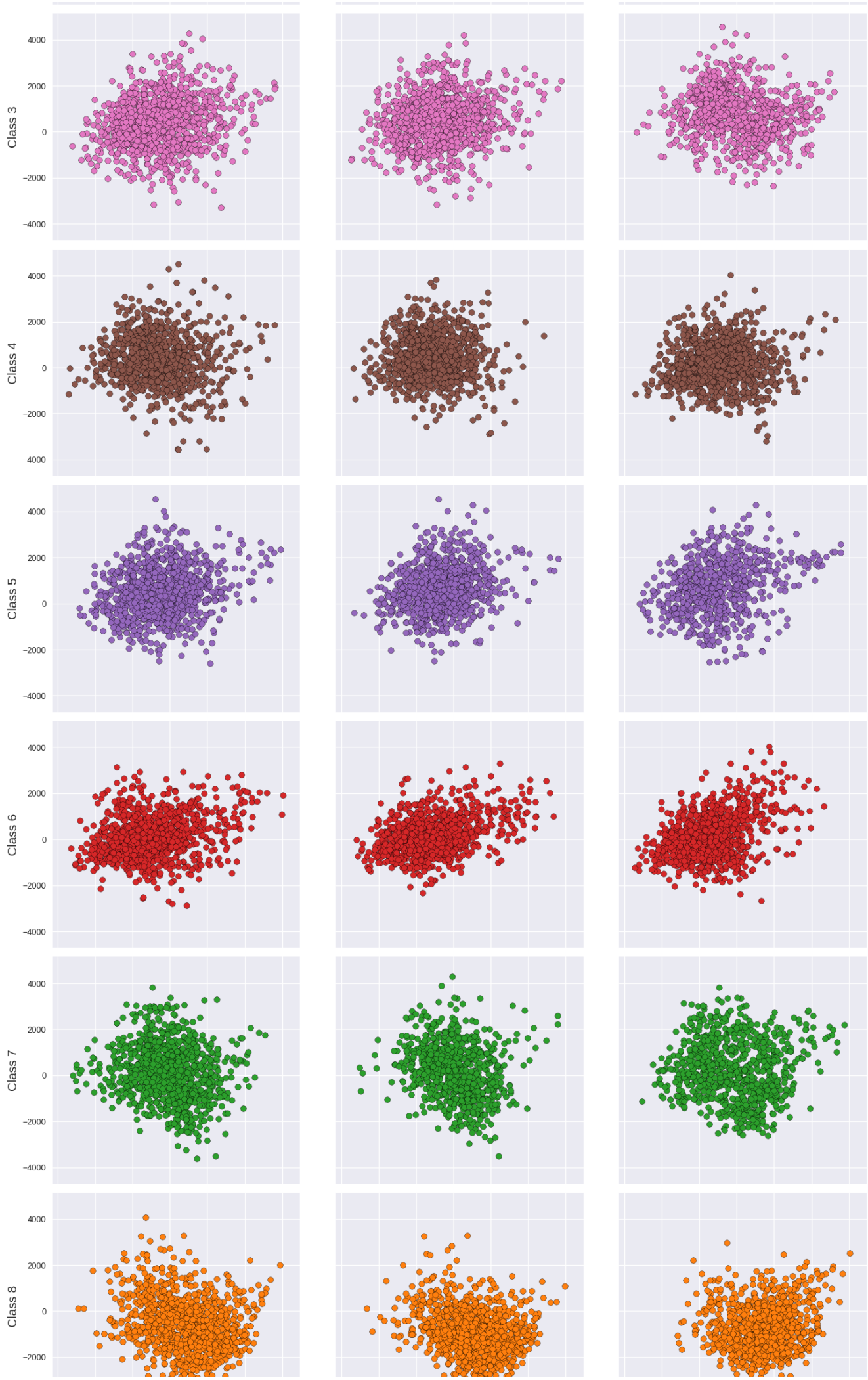
```
Best SVC:
             precision    recall  f1-score   support

          0       0.31      0.37      0.34       973
          1       0.29      0.25      0.27       979
          2       0.19      0.35      0.25      1030
          3       0.18      0.15      0.17      1023
          4       0.19      0.23      0.21       933
          5       0.25      0.18      0.21      1015
          6       0.24      0.25      0.25       996
          7       0.27      0.22      0.24       994
          8       0.41      0.36      0.38      1017
          9       0.35      0.24      0.29      1040

avg / total       0.27      0.26      0.26     10000
```

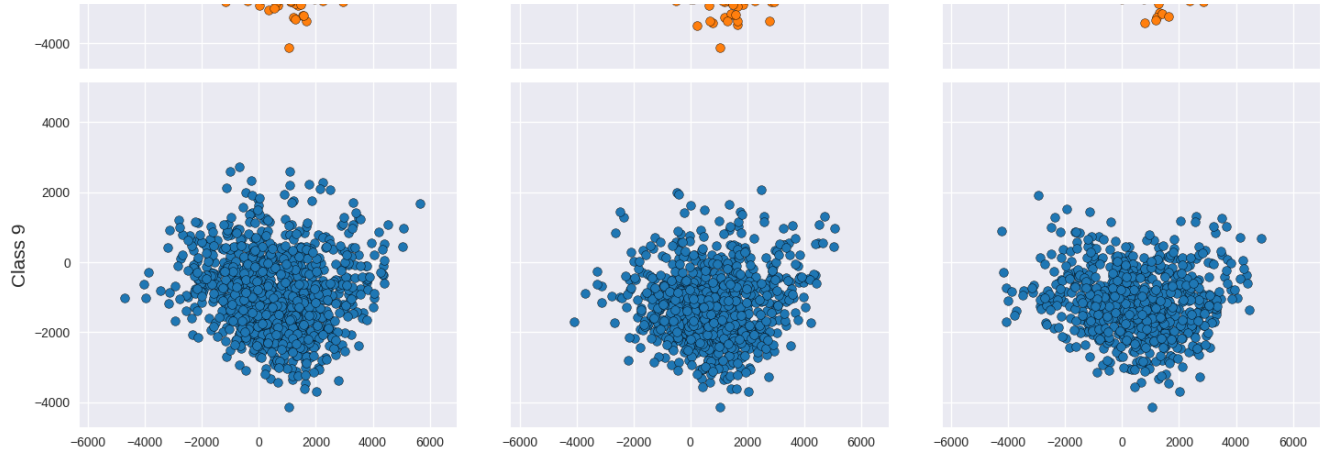
```
Random Forest:
             precision    recall  f1-score   support

          0       0.42      0.49      0.45       973
          1       0.41      0.50      0.45       979
          2       0.31      0.30      0.31      1030
          3       0.28      0.26      0.27      1023
          4       0.31      0.33      0.32       933
          5       0.34      0.32      0.33      1015
          6       0.42      0.42      0.42       996
          7       0.45      0.36      0.40       994
          8       0.50      0.51      0.50      1017
          9       0.48      0.43      0.45      1040

avg / total       0.39      0.39      0.39     10000
```

`RandomForest` 的表現反而比較比，這想這可能是因為圖片本身就是比較難分的，在這種情況下，不基於各維度距離來分類的 `RandomForest` 比較有抗性，得到較好的結果。雖然正確率偏低，不過這也符合網路上的說法，根據網路所查到的資料，Cifar10 要做到 50% 以上，基本得使用 Deep Learning 的方法。

Visualization




Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
4000
2000
0
−2000
−4000



從上圖也可發現 `SVM` 確實表現不甚理想。但這個圖我覺得怪怪的，明明 SVM 的 Accuracy 只有 0.26 左右，但看圖似乎大部份點都預測正確啊。我有檢查過程式碼，沒問題，所以我覺得是 Accuracy 不夠有代表性造成的。

Difficulties

Gamma takes no effect on Linear SVM

根據這三個 Dataset 跑出來的結果，我發現 `Gamma` 對 `Linear SVM` 的結果沒有任何影響，不過我在網路上並沒有找到類似的結果，所以我認為這應該只是巧合。

T-SNE

我本來是要使用 t-sne 這個高級技術來可視化資料，但發現跑起來 太慢了，尤其當資料量非常多的時候，例如 Cifar 50000 張圖片，每張 3072 維。所以我最後使用 `PCA` 來可視化，而一開始的做法是所有資料畫在同一張圖片，但發現這樣許多不同 class 的點會重疊，造成許多點看不到，因此我分 class 來顯示，並把所有資料同到同一個線性空間中。

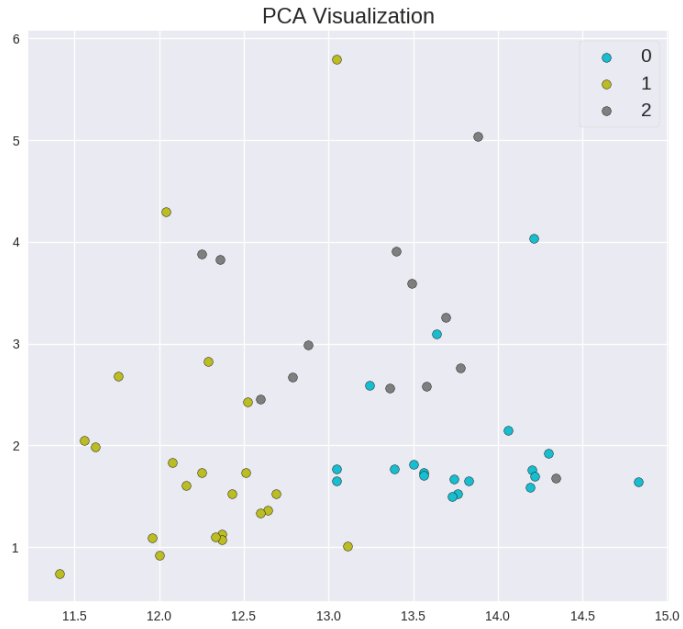
Pickle

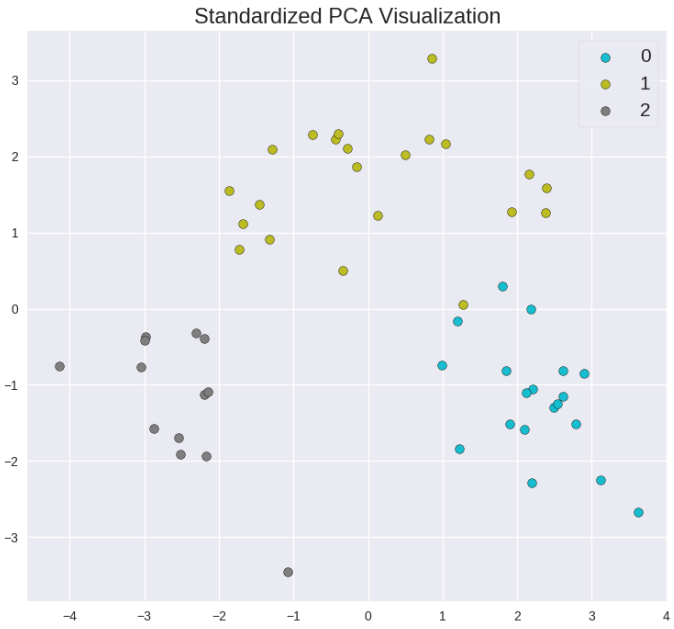
在使用 `GridSearchCV` 時，程式會執行非常久，通常不只一天，我得到的結果是一個變數，程式一關係就沒有了，所以這時就需要將之存在一個檔案，方便之後存取，也防止一不小心 un-reference 該變數，該變數被回收，然後得重跑程式。因此，我使用了 `pickle` 這個 python library，他可以將大部份的變數存在檔案持久化，用法為：

```
# save
with open('gridcv.pkl', 'wb') as f:
    pickle.dump(clf, f)

# load
with open('gridcv.pkl', 'rb') as f:
    clf = pickle.load(f)
```

Standardization PCA is good

在處理 Wine 這個 dataset 時，我發現有沒有做 `Standardization` 與 `PCA` 差距很大。做了之後，資料直接變成線性可分，如圖：





SVM(rbf) diverse

一開始在跑 MNIST 時，發現 `SVM(rbf kernel)` 一直跑不出來，但 `SVM(poly kernel)` 就 train 很快，後來在同學的幫助下才發現，原因是我沒有做 `Normalization` 也沒有做 `Standardization` ，造成 **Curse of Dimensionality** 發生了。我將每個維度都除上 `255` ，問題就解決了。