Artificial Intelligence
Homework 3

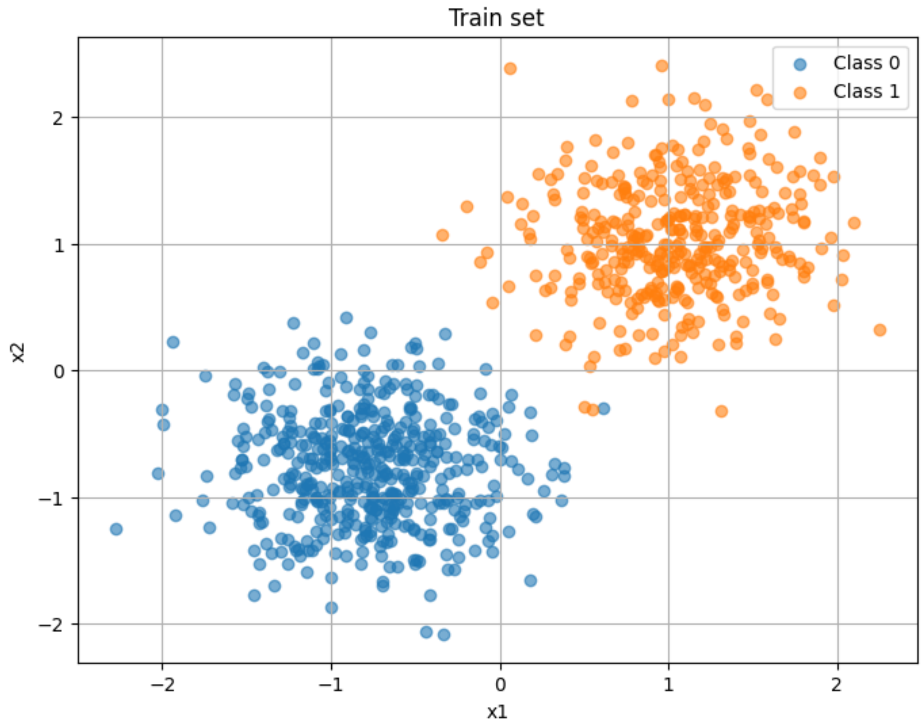
姓名: 劉育辰

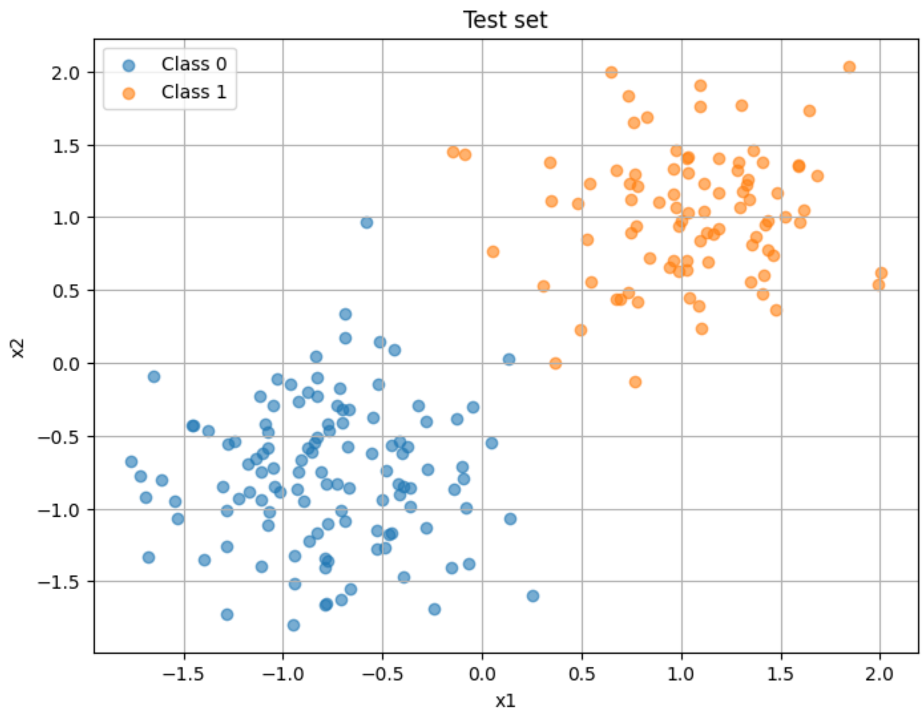
學號: 110303585

系級: 機械 4C

1. Decision Tree & Random Forest

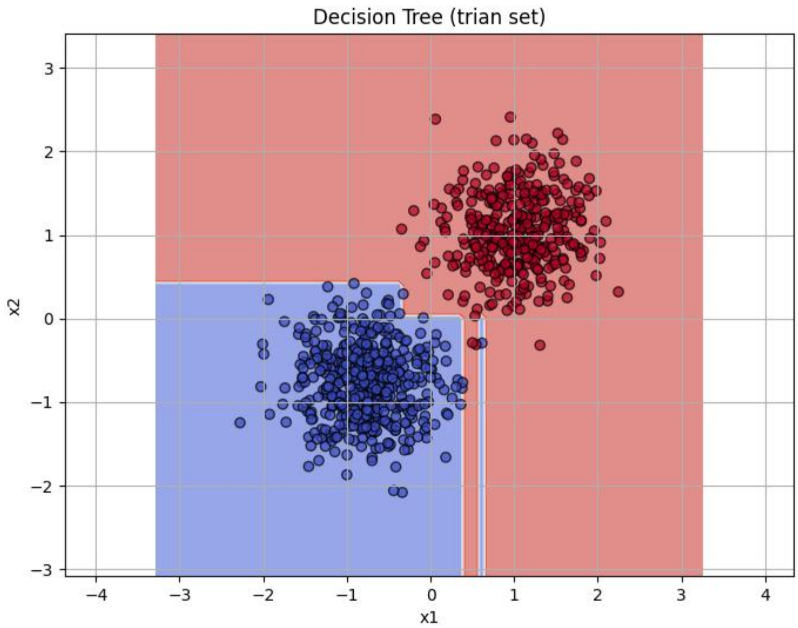
(1) Split S1 and S2 into two sets: 80% for training and 20% for test.



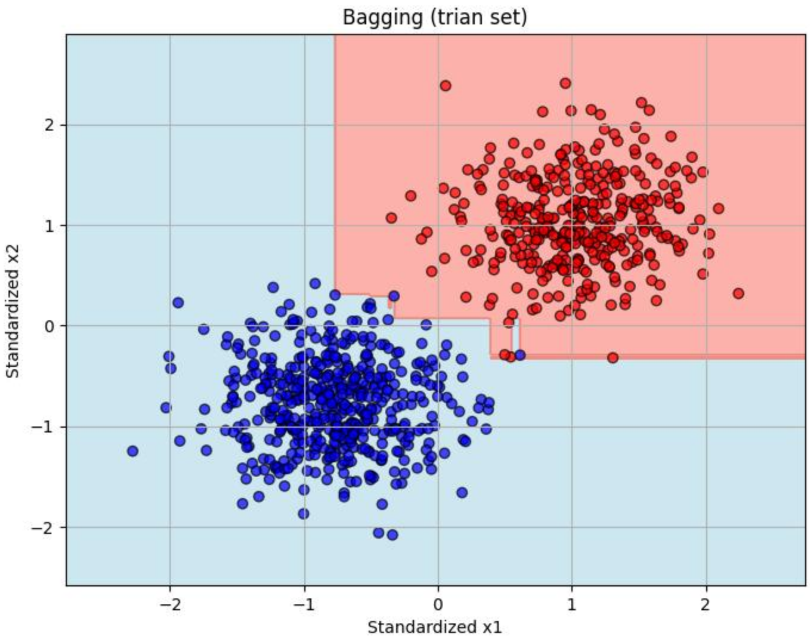


(2) With your own code, find the decision boundary by applying the following methods:

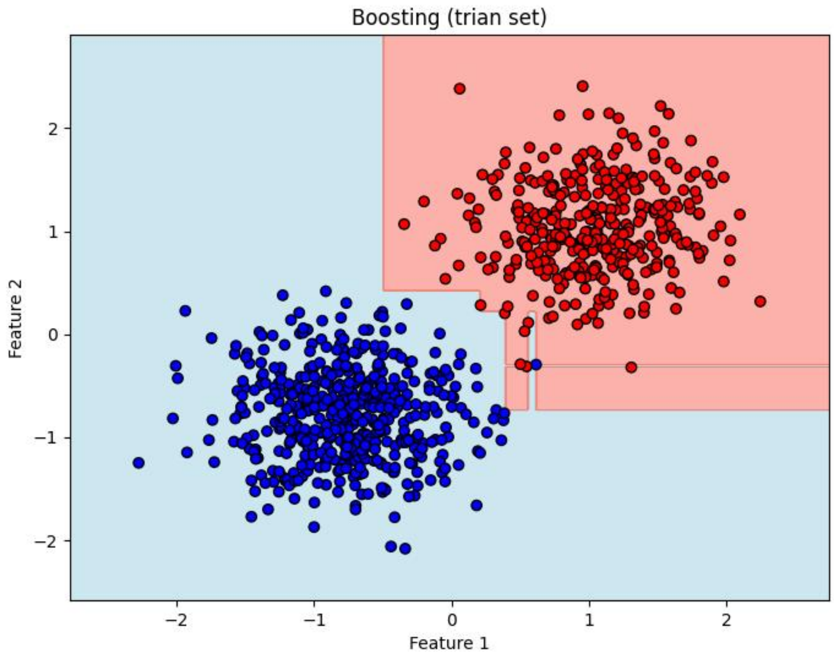
a. Decision tree



b. Random forest with bagging method



c. Random forest with boosting method

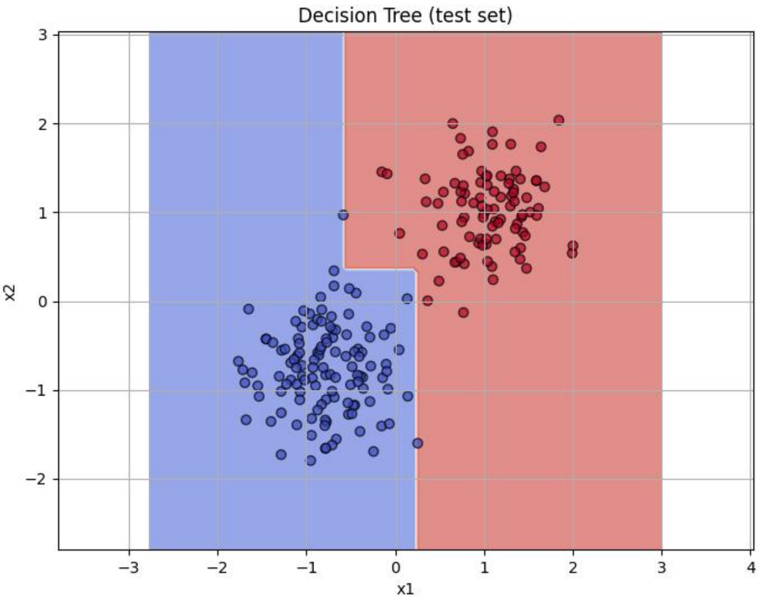


由以上圖片可以看出三個做法都有分類到兩類資料點，甚至連圖片上右方，紅藍交錯的部分都有分類到。但決策樹的被認定為紅色的範圍太大，未來加入新的資料點可能會有分類錯誤的問題。

(3) Verify your classification performance by the test data set.

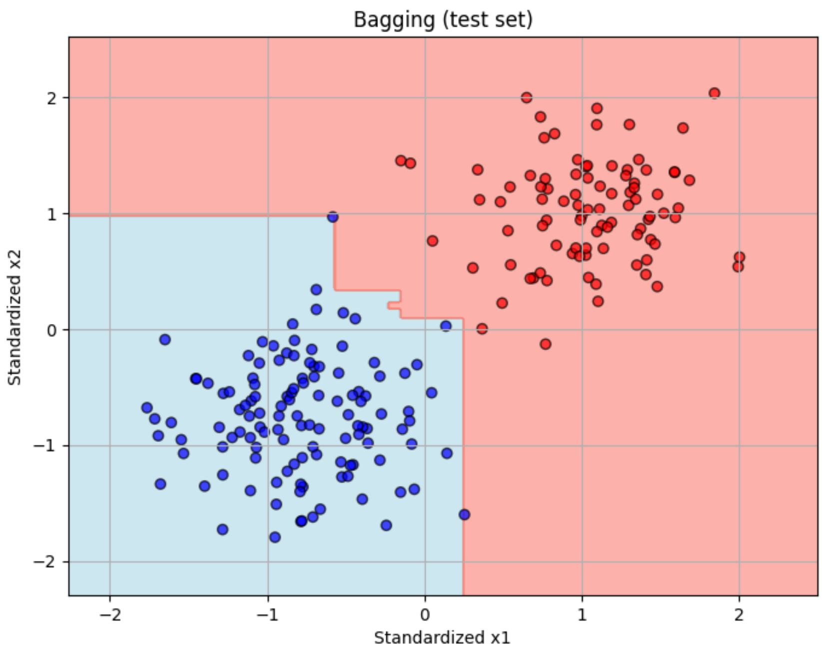
a. Decision tree

```
def build_tree(X, y, max_depth=None, depth=0, n_features=None):
    m, n = X.shape
    unique_classes = np.unique(y)

    if len(unique_classes) == 1:
        return Node(value=unique_classes[0])

    if max_depth is not None and depth >= max_depth:
        return Node(value=np.bincount(y).argmax())

    feature, threshold = best_split(X, y, n_features=n_features)
    if feature is None:
        return Node(value=np.bincount(y).argmax())

    left_mask = X[:, feature] <= threshold
    right_mask = ~left_mask
    left_node = build_tree(X[left_mask], y[left_mask], max_depth, depth + 1, n_features)
    right_node = build_tree(X[right_mask], y[right_mask], max_depth, depth + 1, n_features)

    return Node(feature=feature, threshold=threshold, left=left_node, right=right_node)
```



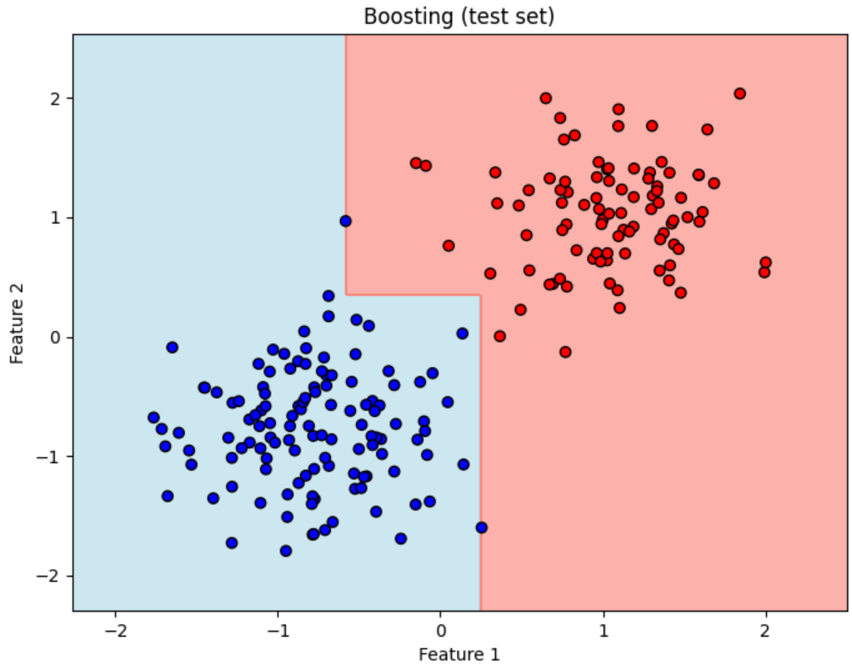
b. Random forest with bagging method

```
def train_bagging(X, y, n_trees=10, max_depth=4, n_features=None):
    forest = []
    for _ in range(n_trees):
        X_sample, y_sample = bootstrap_sample(X, y)
        tree = build_tree(X_sample, y_sample, max_depth=max_depth, n_features=n_features)
        forest.append(tree)
    return forest
```



c. Random forest with boosting method

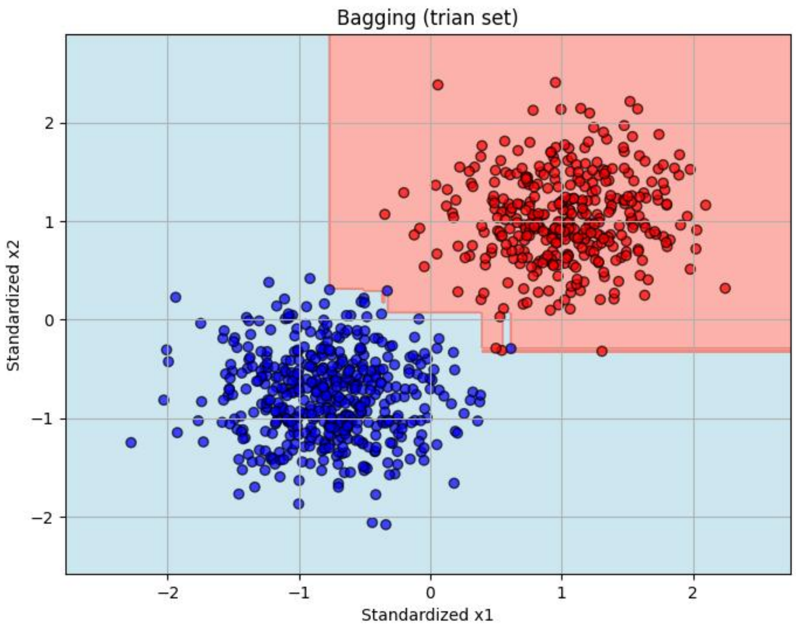
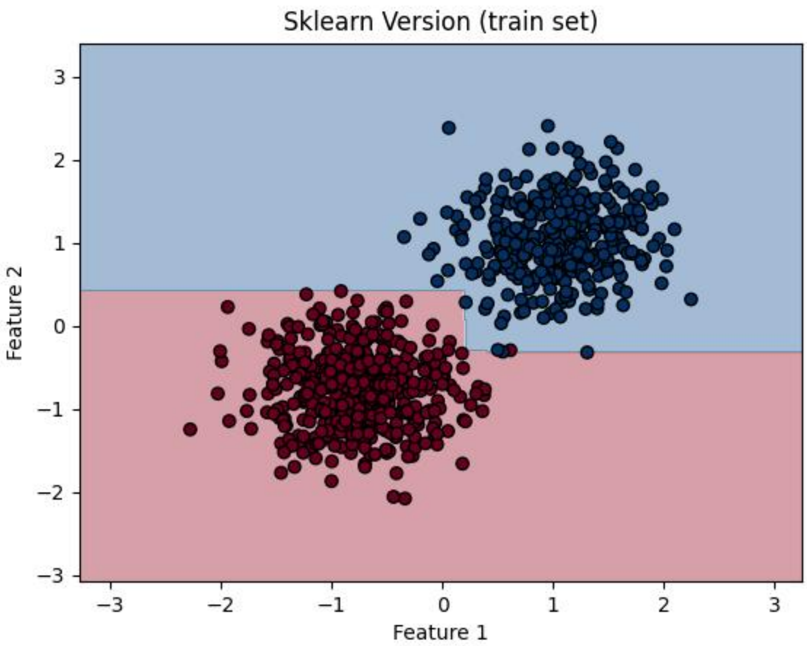
```
def train_boosting(X, y, tree_nums, max_depth=1):
    m = X.shape[0]
    weights = np.ones(m) / m
    classifiers = []
    alphas = []

    for _ in range(tree_nums):
        indices = np.random.choice(m, m, replace=True, p=weights)
        X_sample, y_sample = X[indices], y[indices]

        stump = build_tree(X_sample, y_sample, max_depth=max_depth)
        pred = np.array([predict_tree(stump, x) for x in X])

        err = np.sum(weights * (pred != y)) / np.sum(weights)
        if err > 0.5:
            continue

        alpha = 0.5 * np.log((1 - err) / (err + 1e-10))
        alphas.append(alpha)
        classifiers.append(stump)

        weights *= np.exp(-alpha * y * (2 * (pred == y) - 1))
        weights /= np.sum(weights)

    return classifiers, alphas
```

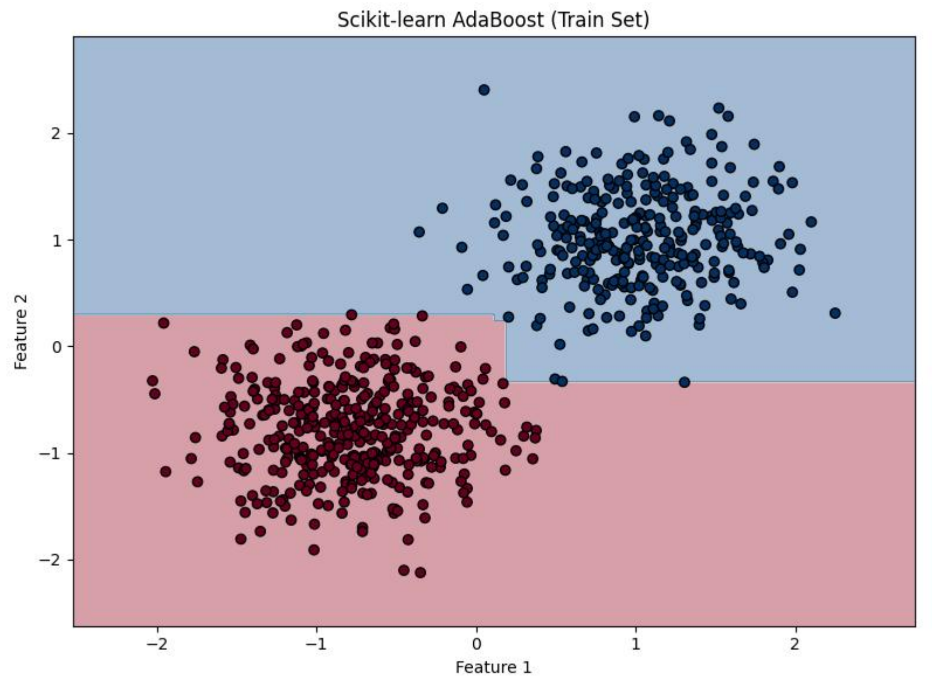


由測試集可以看出需練出來的回歸森林和決策樹分類的效果不錯，都有正確分類出兩類資料點。

(4) Compare your result with the Scikit-Learn's package and generative AI.

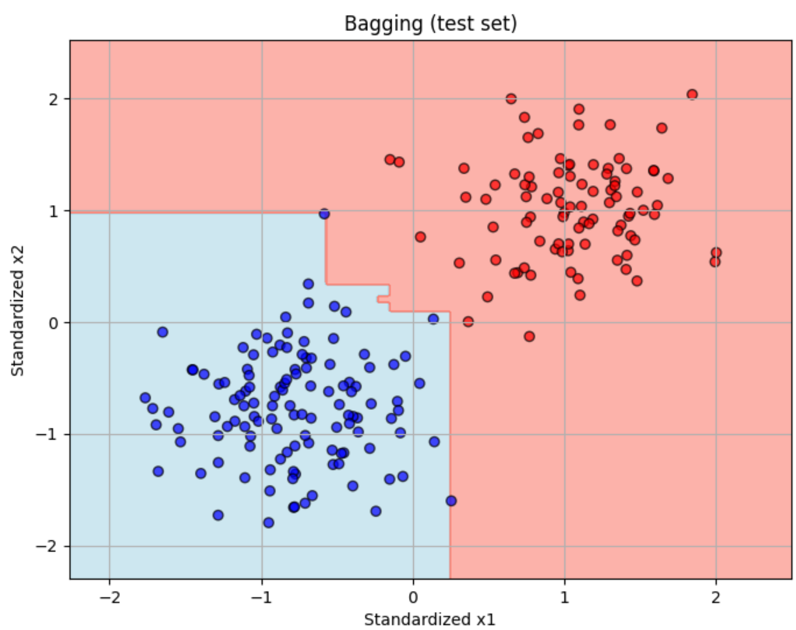
a. 訓練集比較:

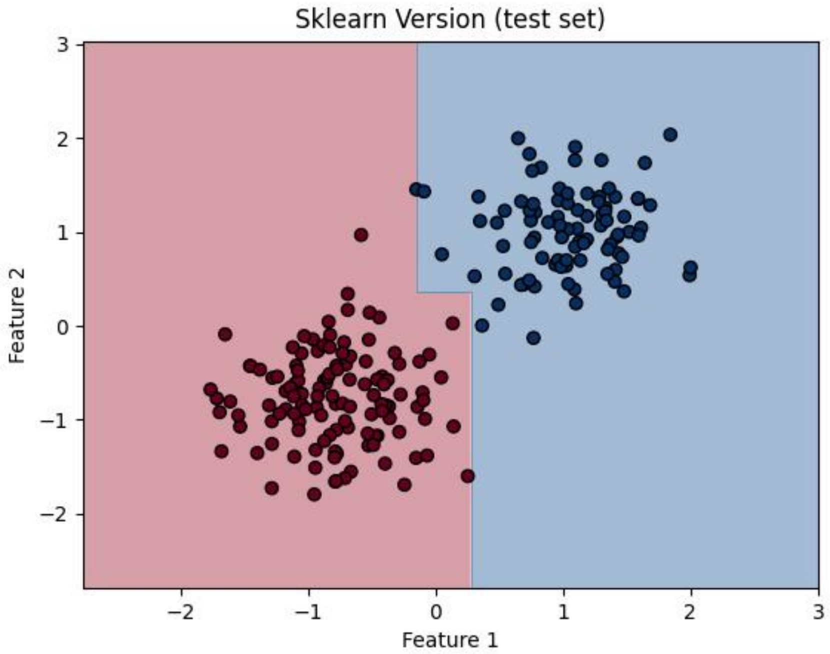
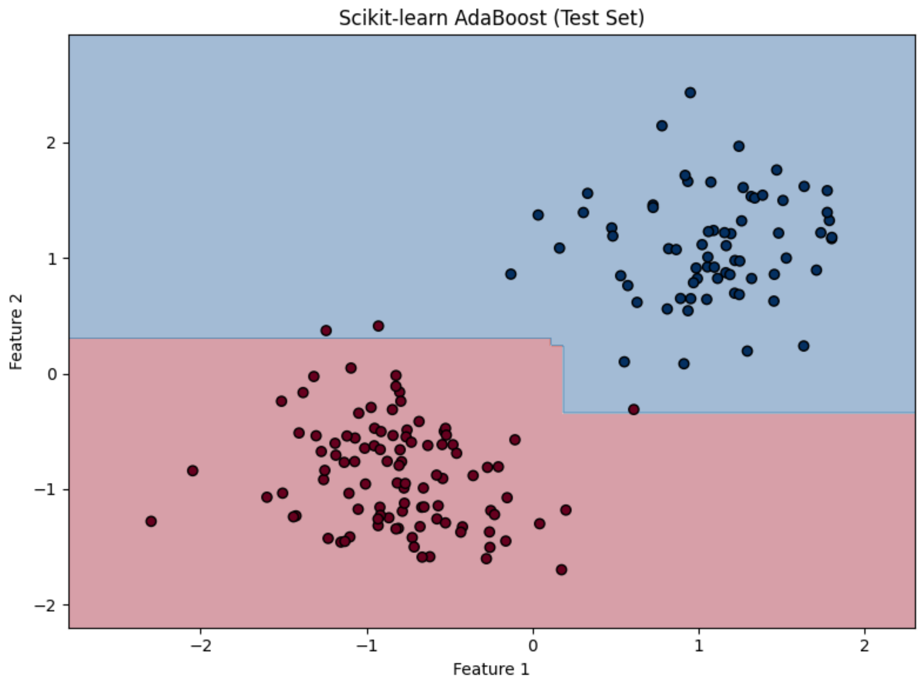
MY	
SK	

AI	

從圖上來看，分類效果不錯，我的結果和 AI 跟 SK 套件相比，沒有將整個空間分成一半，只有 4 分之 1 的部分是紅色那類。

b. 測試集比較:

M Y	

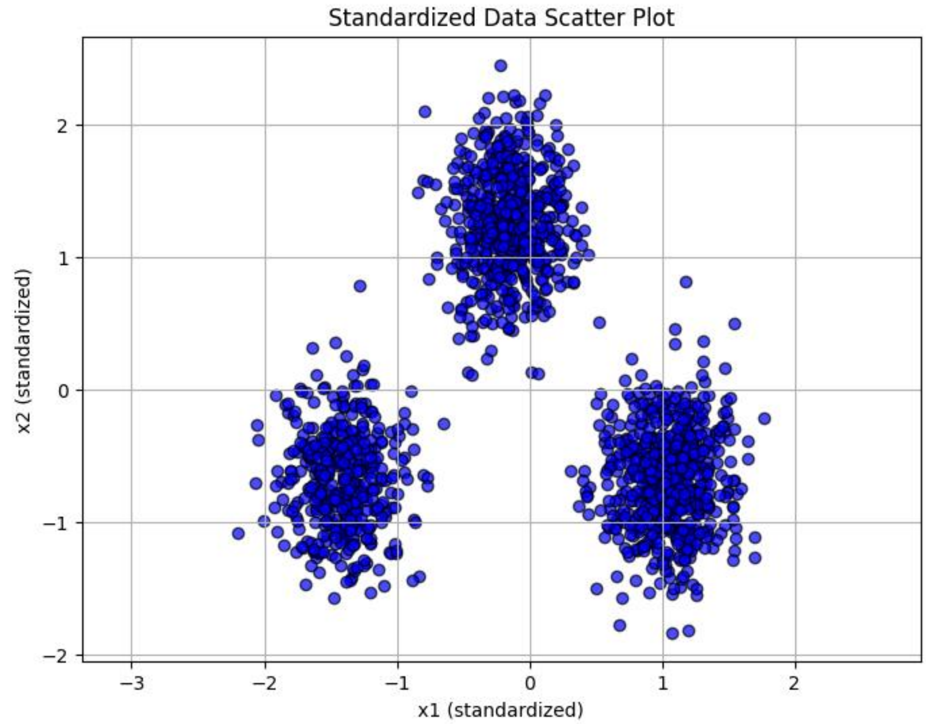
SK	
AI	

結果和訓練集差不多，但 AI 的突有點問題，有兩個點是在藍色類別的範圍內，卻被分類為紅色。

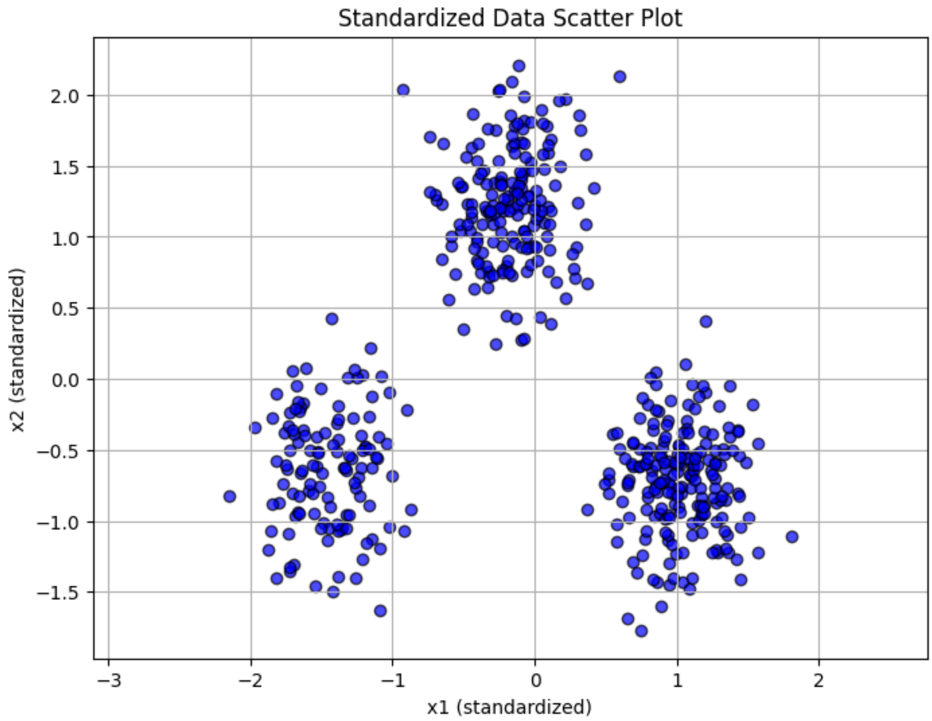
2. K-means

(1) Split S1, S2 and S3 into two sets: 80% for training and 20% for test

Train set



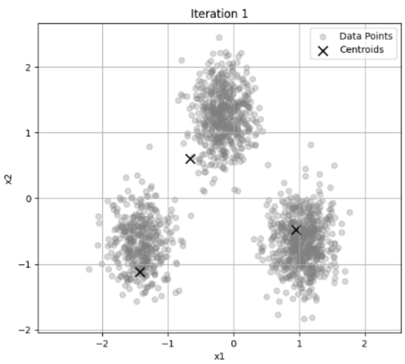
Test set

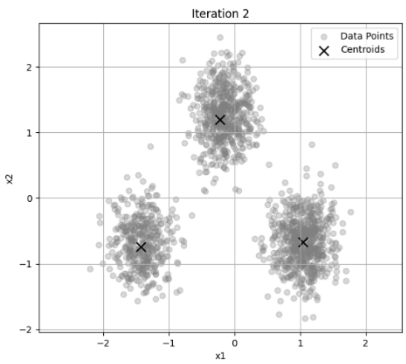


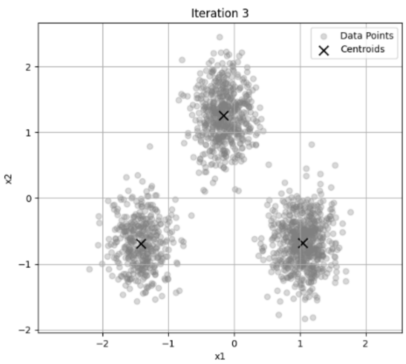
```
# Train
while not mutux and iteration < 10:
    labels = assign_clusters(X, centrol)
    new_centrol = update_centrol(X, labels, k)
```

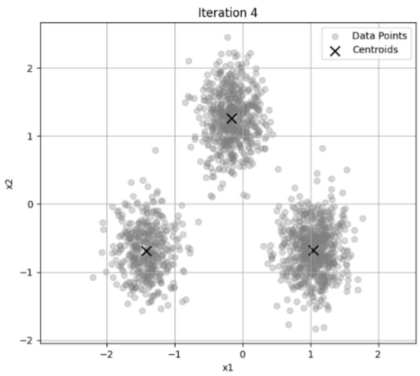
```
def assign_clusters(X, centrol):
    distances = np.linalg.norm(X[:, np.newaxis] - centrol, axis=2)
    return np.argmin(distances, axis=1)

def update_centrol(X, labels, k):
    new_centrol = []
    for i in range(k):
        points = X[labels == i]
        if len(points) > 0:
            new_centrol.append(points.mean(axis=0))
        else:
            new_centrol.append(X[np.random.randint(0, X.shape[0])])
    return np.array(new_centrol)
```

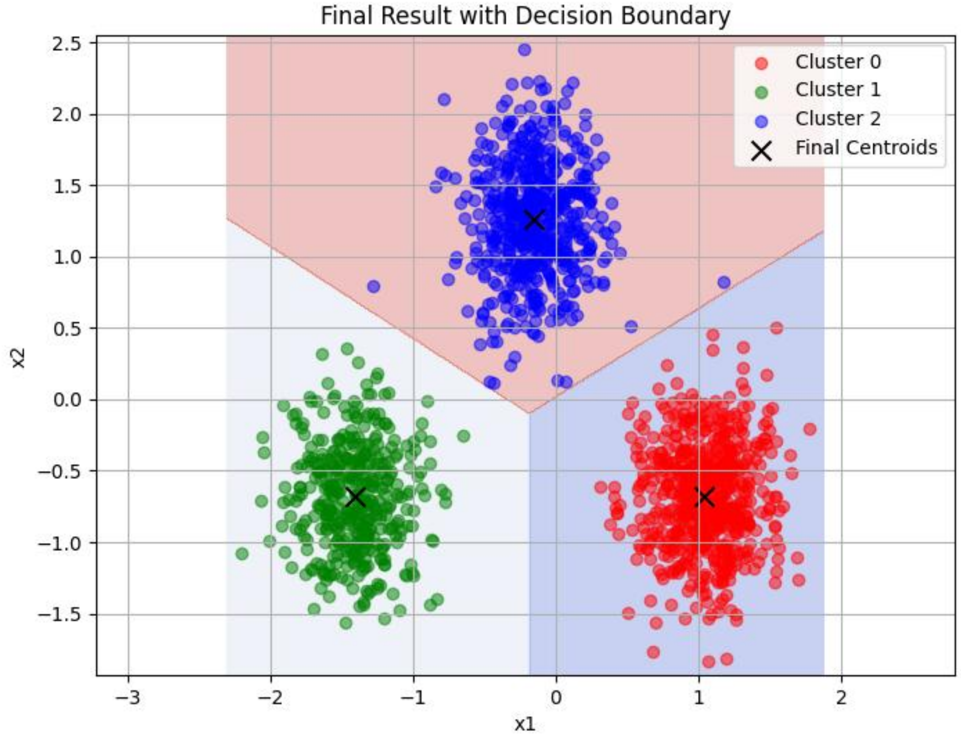
Training process



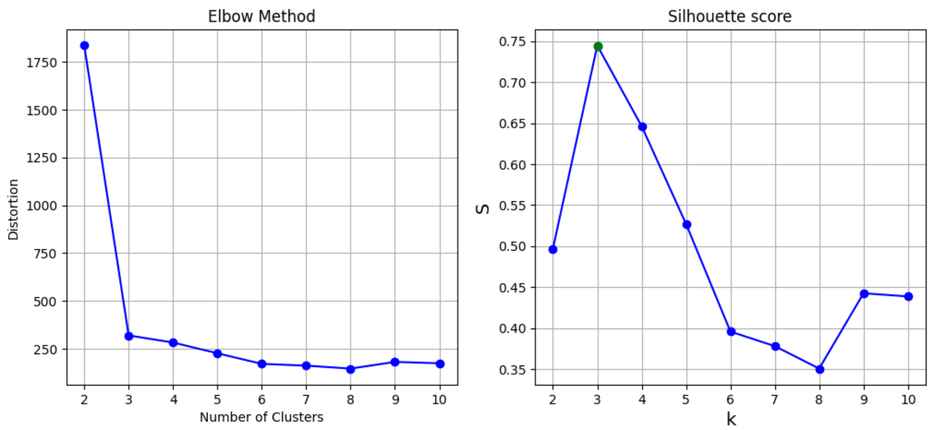






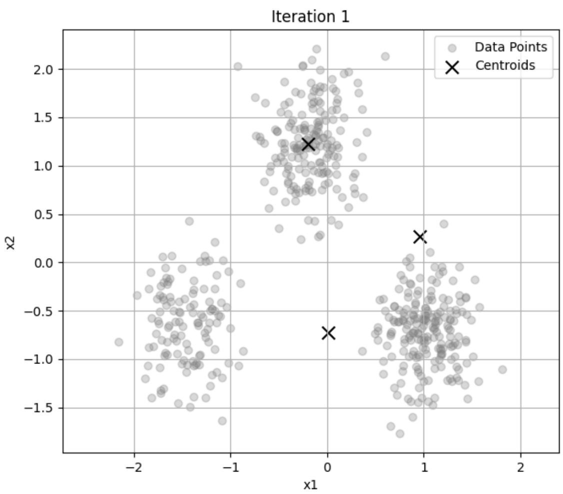


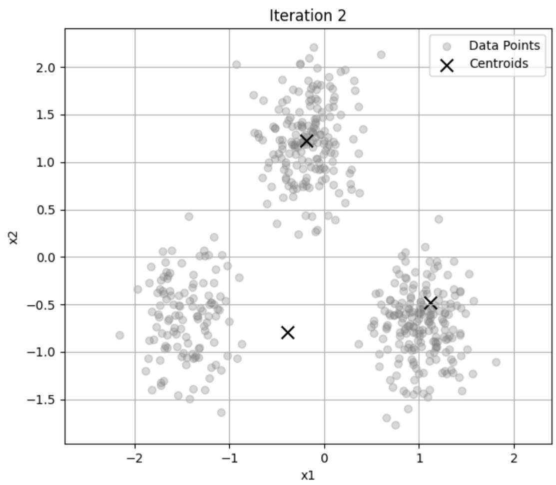
由上圖可以知道迭代到第 5 次時，得到了最終結果，三個資料中心沒有過大的變化，且確實在三群資料點的中心，且透過決策邊界圖也看出分類效果很好。下圖顯示 Elbow Method 和 Silhouette score 的結果，Number of Clusters 在到 3 時快速下降，表示分類為 3 群是最好的。

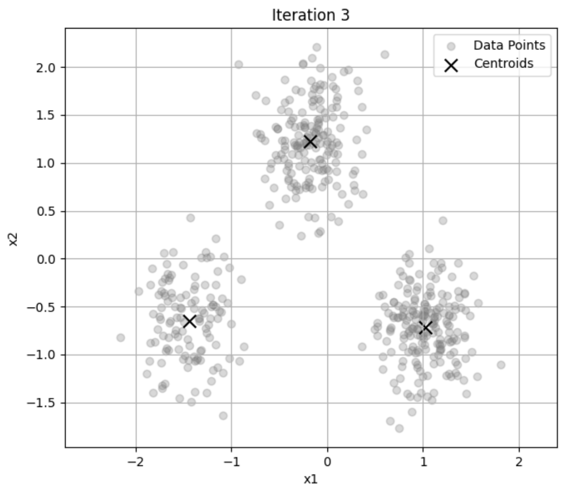


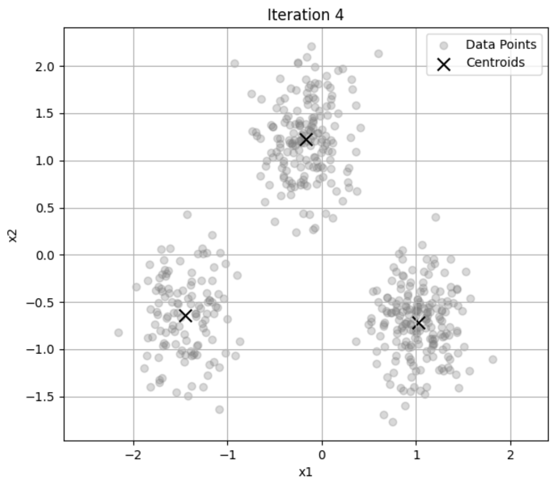
(2) Verify your classification performance by the test data set.

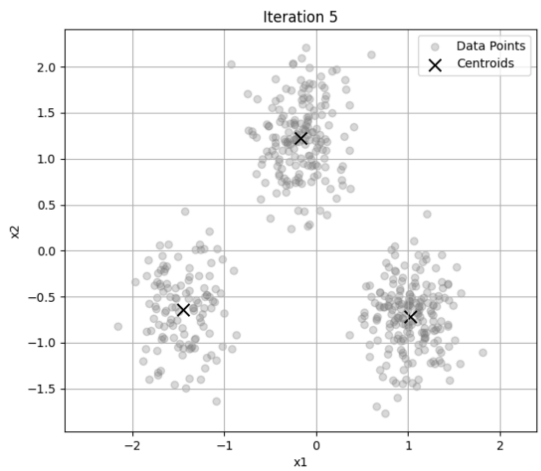
Train process

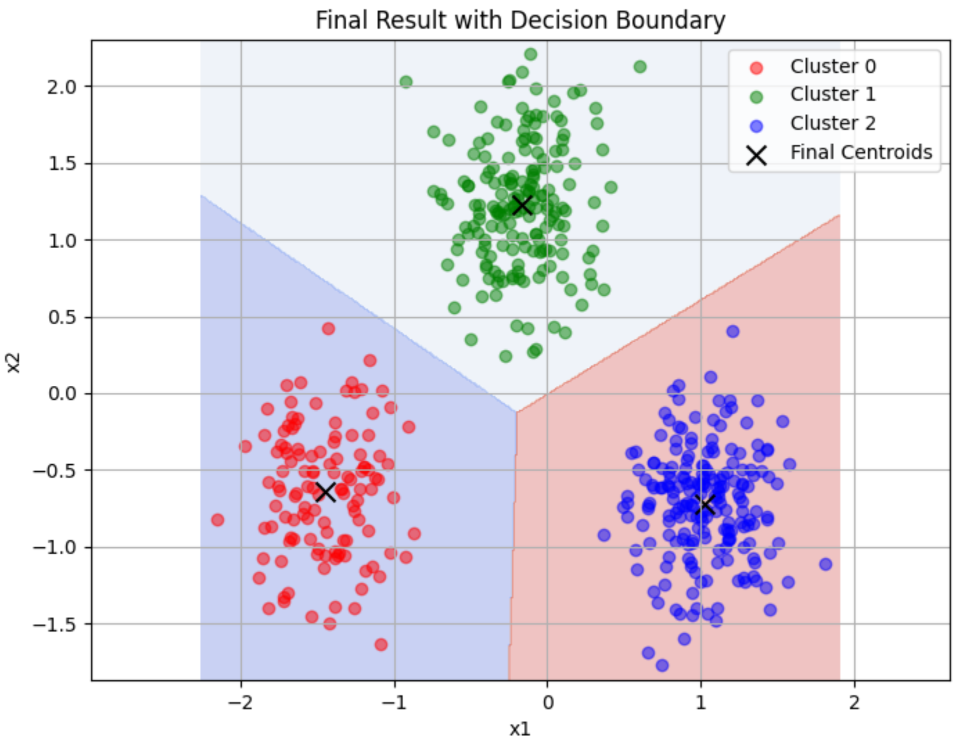




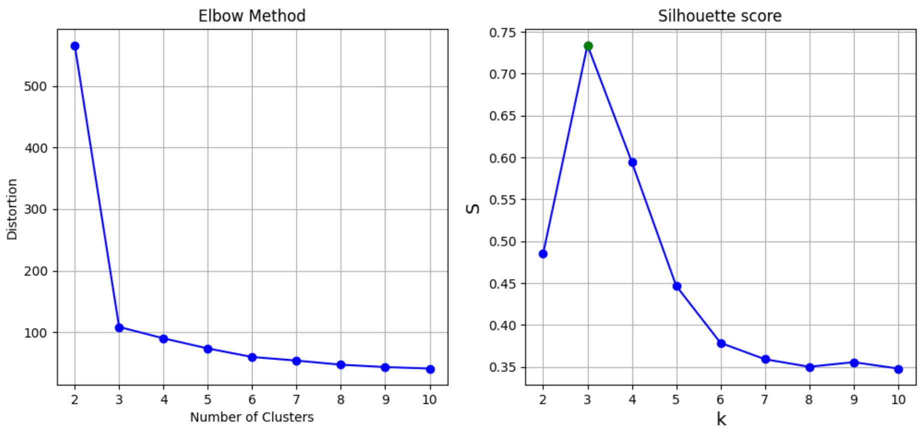




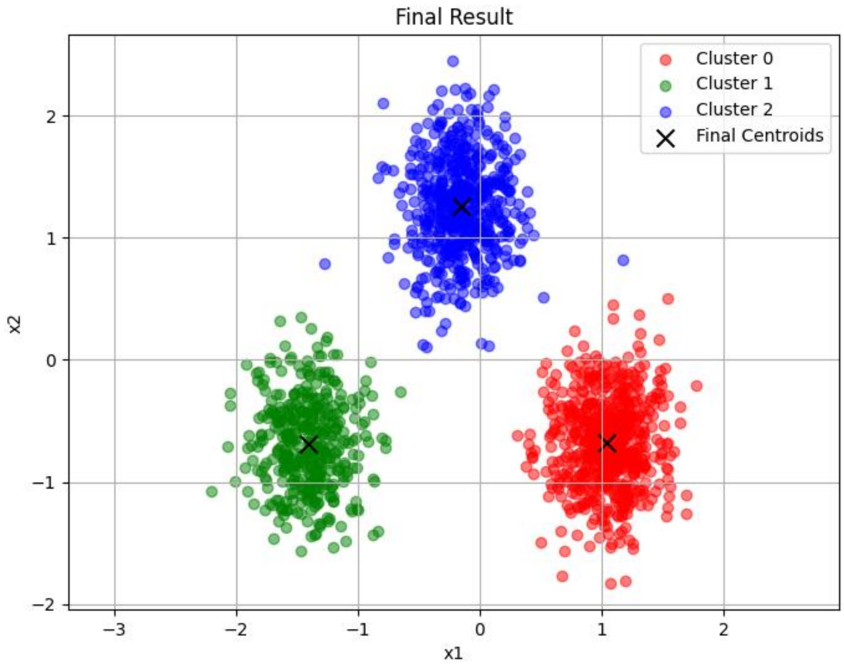
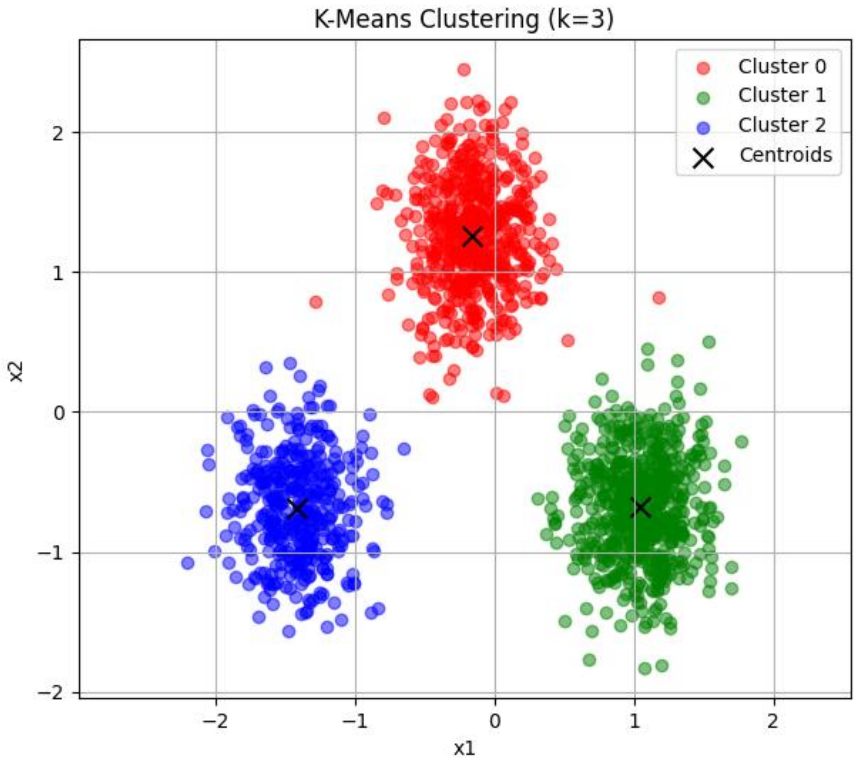


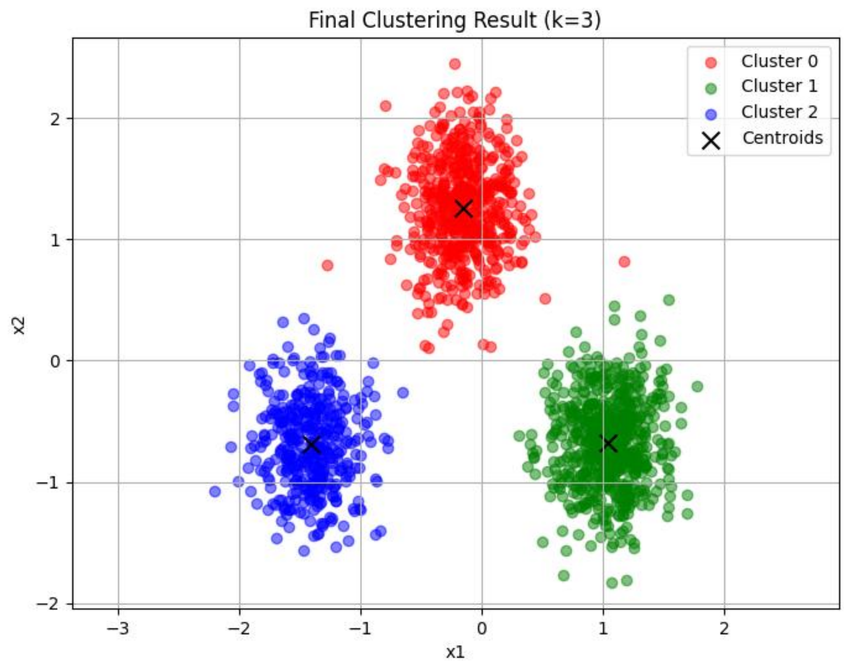


從測試集可以看出效果不錯，跟訓練集一樣有成功分出三類，Elbow Method 和 Silhouette score 的結果和訓練集的結果一樣。

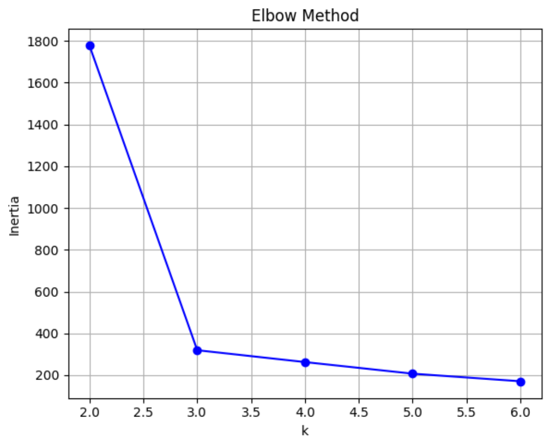


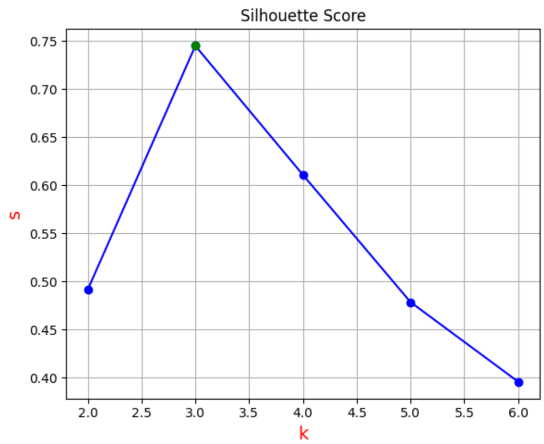
(3) Compare your result with the Scikit-Learn's package and generative AI

MY	
SK	

AI	

三個版本的比對結果，基本上沒有差別，分類效果也很好。





上面是 SKlearn 版本，下面是 AI 版本。