Practice Lab: Pairwise Seperation

R Abhijit Srivathsan

2448044

In [1]:
```
import numpy as np
from itertools import combinations

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
```

In [2]:
```
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names
```

In [3]:
```
print("Feature matrix shape:", X.shape)
print("Labels shape:", y.shape)
print("Unique class labels:", np.unique(y), target_names)
```

```
Feature matrix shape: (150, 4)
Labels shape: (150,)
Unique class labels: [0 1 2] ['setosa' 'versicolor' 'virginica']
```

In [4]:
```
class_pairs = list(combinations(np.unique(y), 2))
print("\nPairwise class combinations:")
for pair in class_pairs:
    print(f"  Classes {pair[0]} vs {pair[1]} ({target_names[pair[0]]} vs {target_names[pair[1]]})")
```

```
Pairwise class combinations:
  Classes 0 vs 1 (setosa vs versicolor)
  Classes 0 vs 2 (setosa vs virginica)
  Classes 1 vs 2 (versicolor vs virginica)
```

In [5]:
```
for pair in class_pairs:
    # Extract samples belonging to just these two classes
```

```
    idx = np.where((y == pair[0]) | (y == pair[1]))
    X_pair = X[idx]
    y_pair = y[idx]

    # Define and train the model (SVC)
    model = SVC(kernel='linear', random_state=42)

    # Let's do a train/test split for demonstration
    X_train, X_test, y_train, y_test = train_test_split(
        X_pair, y_pair, test_size=0.2, random_state=42
    )

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print("\n==================================")
    print(f"Results for classes {pair} ({target_names[pair[0]]} vs {target_names[pair[1]]})")
    print("==================================")
    print("Classification Report:")
    print(classification_report(y_test, y_pred, target_names=[target_names[pair[0]], target_names[pair[1]]]))
    print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
==================================
Results for classes (np.int64(0), np.int64(1)) (setosa vs versicolor)
==================================
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       1.00      1.00      1.00         8

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Accuracy: 1.0

==================================
Results for classes (np.int64(0), np.int64(2)) (setosa vs virginica)
==================================
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
   virginica       1.00      1.00      1.00         8

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Accuracy: 1.0

==================================
Results for classes (np.int64(1), np.int64(2)) (versicolor vs virginica)
==================================
Classification Report:
              precision    recall  f1-score   support

  versicolor       1.00      0.83      0.91        12
   virginica       0.80      1.00      0.89         8

    accuracy                           0.90        20
   macro avg       0.90      0.92      0.90        20
```

```
weighted avg       0.92      0.90      0.90        20

Accuracy: 0.9
```

In [6]:
```python
cv_scores = cross_val_score(model, X_pair, y_pair, cv=5)
print("Cross-validation scores (5-fold):", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

```
Cross-validation scores (5-fold): [0.95 1.   0.95 0.95 1.  ]
Mean CV Accuracy: 0.97
```

Observation

From the 5-fold cross-validation, we see individual fold accuracies of `[0.95, 1.0, 0.95, 0.95, 1.0]` and a mean accuracy of `0.97`. This indicates that in most folds, the classifier performs very well, often achieving a perfect score (100%). The slight drop to 0.95 in some folds suggests a small number of misclassifications. Overall, the model consistently separates these two classes with high accuracy.

Conclusion

The consistently high cross-validation accuracy (mean = `0.97`) demonstrates that the classifier (in this pairwise setting) is highly effective at distinguishing between these two particular classes. The results suggest strong separability in the chosen feature space and indicate that the model is robust for this binary (pairwise) classification task.