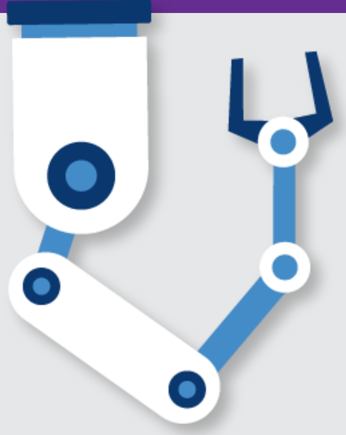
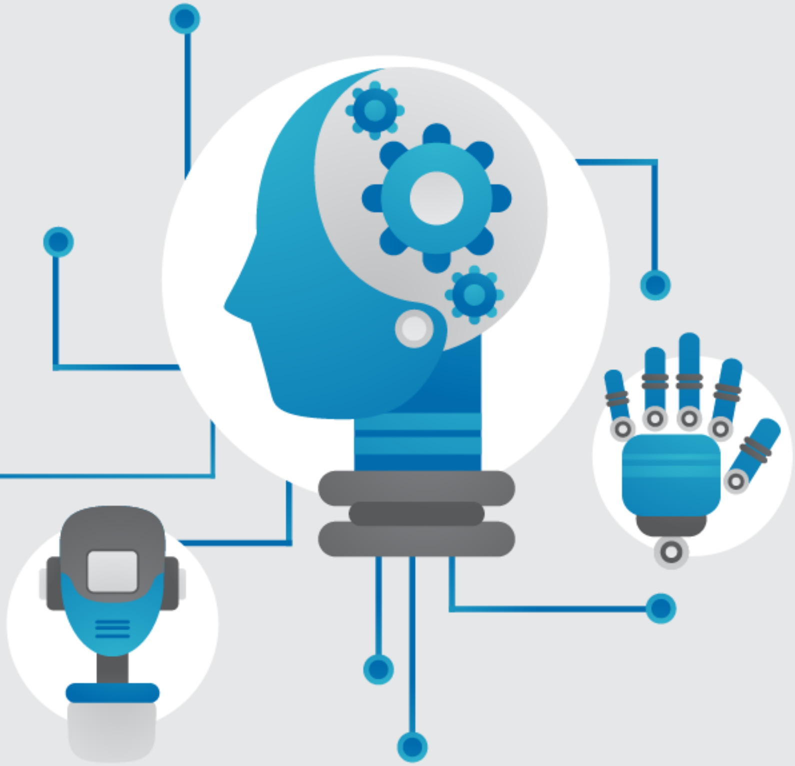
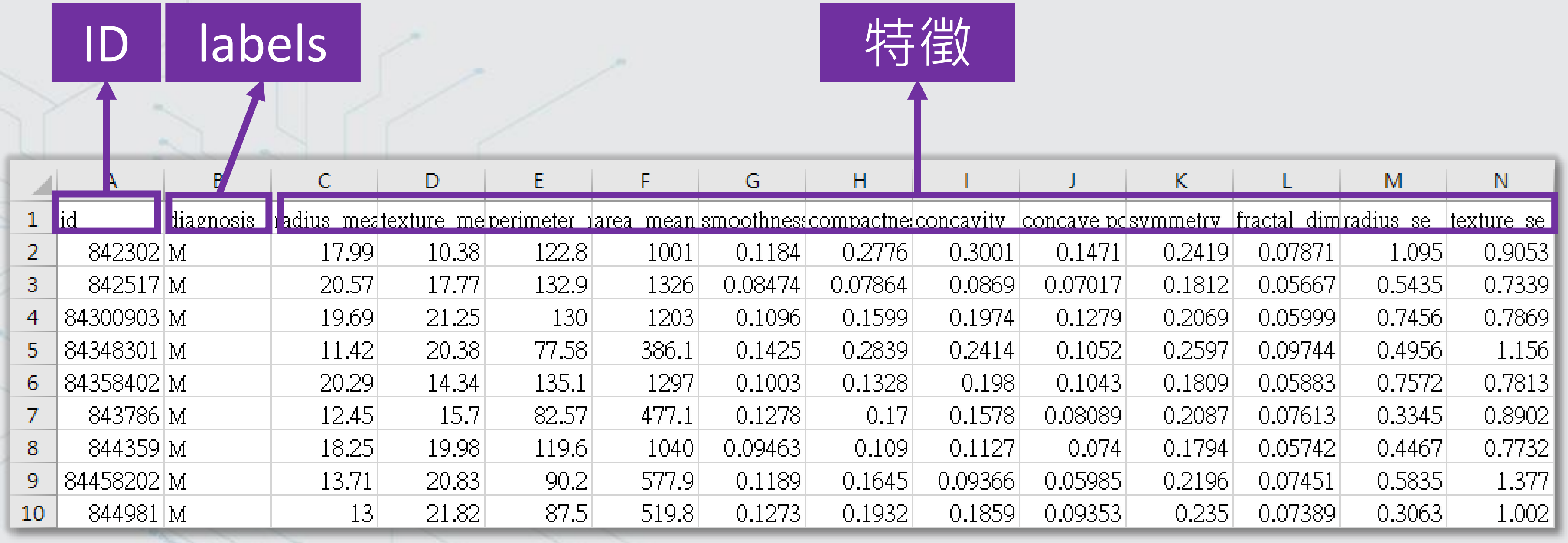
Random Forest
模型建置流程





Kaggle數據集 –
Breast Cancer Wisconsin (Diagnostic) Data Set

- 資料檔案：data.csv
- 含有569筆資料，每筆資料有32格欄位，第一格為ID，第二格為labels，之後的30格為該筆資料的特徵。



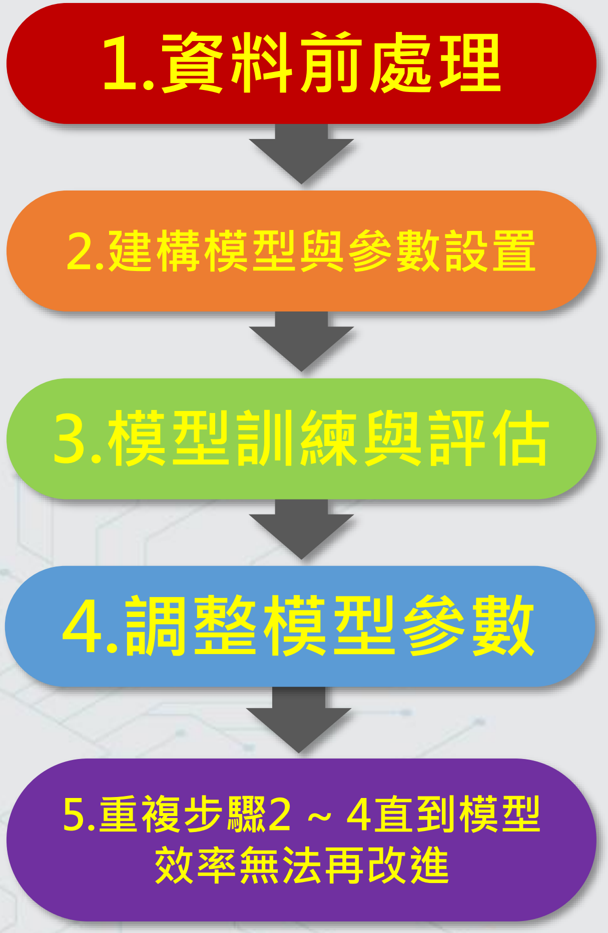
ID

labels

特徵

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	id	diagnosis	radius_mea	texture_me	perimeter_	area_mean	smoothness	compactne	concavity	concave po	symmetry	fractal_dim	radius_se	texture_se
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156
6	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813
7	843786	M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902
8	844359	M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467	0.7732
9	84458202	M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835	1.377
10	844981	M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063	1.002

Random Forest 模型建置流程



1.資料前處理

2.建構模型與參數設置

3.模型訓練與評估

4.調整模型參數

5.重複步驟2 ~ 4直到模型
效率無法再改進

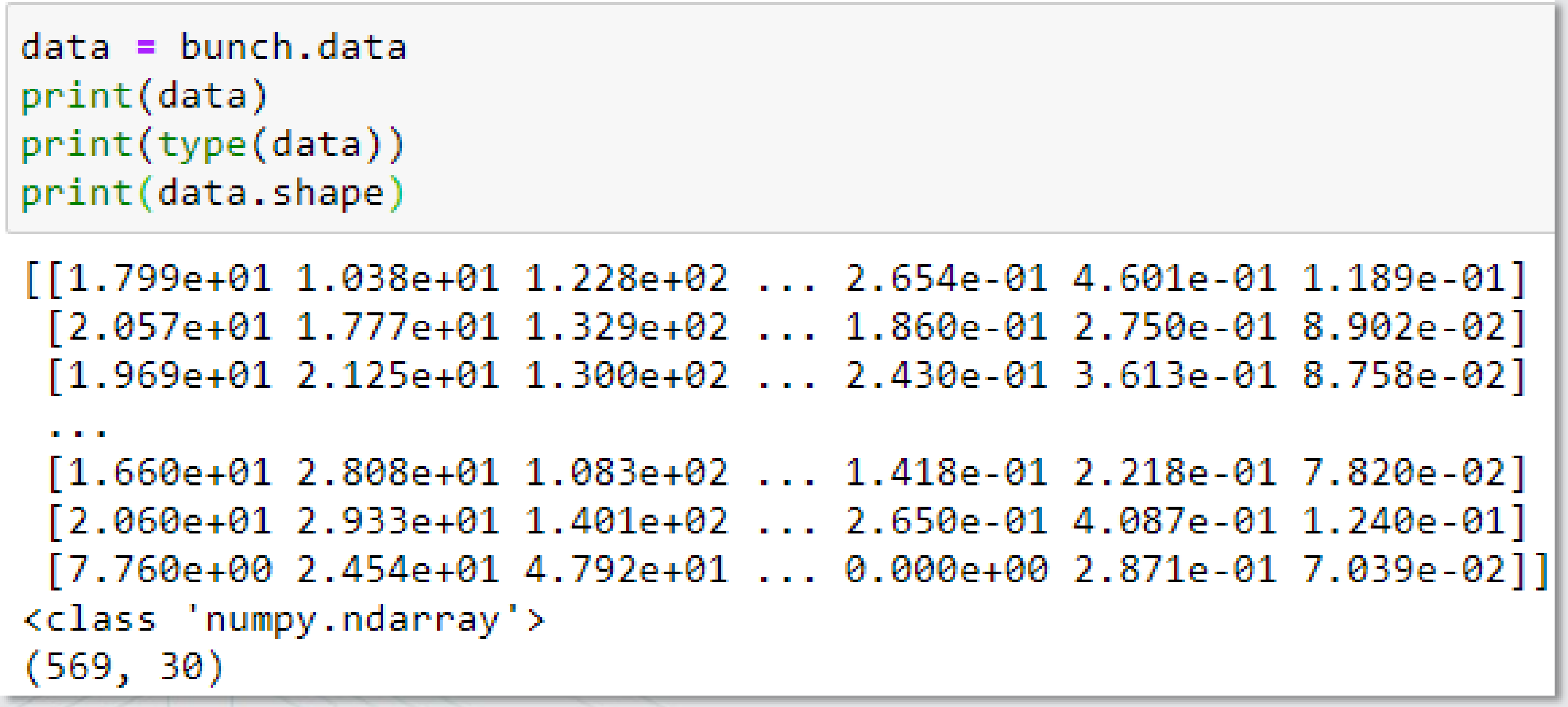
資料前處理

- 從sklearn.datasets載入數據資料

```
# 載入資料
from sklearn import datasets
bunch = datasets.load_breast_cancer()
```

資料前處理

- 訓練資料



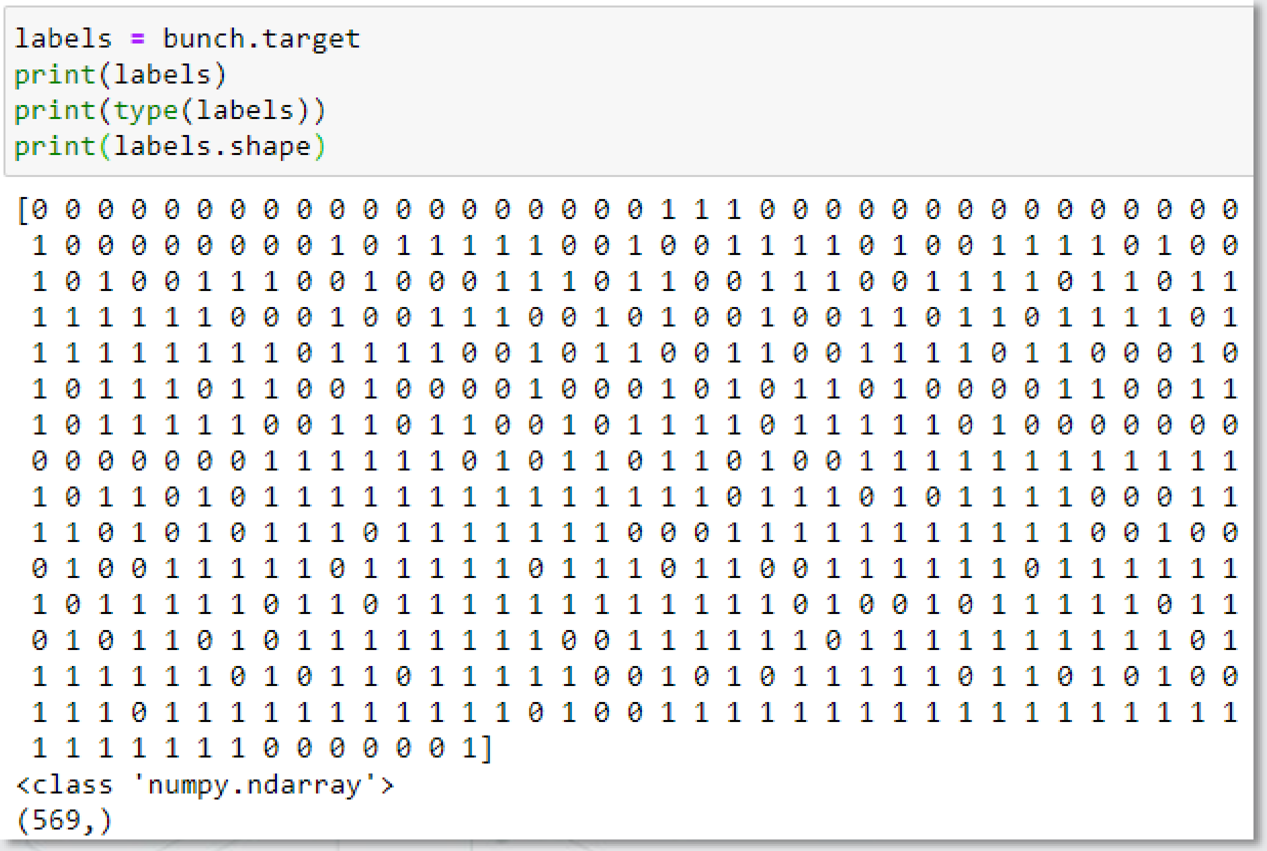
```
data = bunch.data
print(data)
print(type(data))
print(data.shape)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
<class 'numpy.ndarray'>
(569, 30)
```

威斯康辛乳癌數據集處理

機器學習實務

› 分類的標籤



```python
labels = bunch.target
print(labels)
print(type(labels))
print(labels.shape)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
<class 'numpy.ndarray'>
(569,)
```

資料前處理

- 將data以及labels分割成train和test資料

```
#  切割資料
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(bunch.data,
bunch.target, test_size=0.3,shuffle=True,stratify=bunch.target)
```

建立模型與參數設定

- 設定n_estimators, class_weight, n_jobs, verbose

```
# 建立模型
from sklearn.ensemble import RandomForestClassifier
randomForest = RandomForestClassifier(n_estimators=100,
class_weight='balanced',n_jobs=-1, verbose=1)
```

模型訓練與評估

模型訓練與評估

```
# 進行訓練
randomForest.fit(X_train, Y_train)

# 進行預測
acc = randomForest.score(X_test, Y_test)

print('Accuracy:',acc)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.0s finished
Accuracy: 0.9766081871345029
```

特徵選擇

- **過濾法（Filter）**
  根據feature發散的程度（變異數）和features與target的相關性，對各個features進行評分,
  可以設定要選擇的features個數或者設定一個固定的閥值（threshold）並留下評分在閥值以內的features。

- **過濾方式**
  - 移除變異數低的特徵
  - 單變量特徵選擇

特徵選擇

移除變異數低的特徵

使用scikit-learn的VarianceThreshold套件來透過變異數剔除不重要的特徵。

範例程式

```
from sklearn.feature_selection import VarianceThreshold
# 閥值(threshold)為0.01,表示其變異數值低於0.01會被剔除
selector = VarianceThreshold(threshold=0.01)
selector = selector.fit(X_train, Y_train)
X_train = selector.transform(X_train)
X_test = selector.transform(X_test)
```

特徵選擇

- **單變量特徵選擇**

  個別計算每個feature的統計指標，根據該指標判斷特徵的重要性，然後剔除不重要的特徵。

- **評分指標**

  分類問題：

  f_classif(ANOVA F-value), mutual_info_classif和chi2(卡方檢定)

  迴歸問題：

  f_regression和mutual_info_regression

特徵選擇

- scikit-learn方法

  SelectKBest：

  用評分指標算出每個features的得分，並依據這個feature的得分，只留下得分前k名的特徵（取top k）。

  SelectPercentile：

  用評分指標算出每個features的得分，並依據這個feature的得分，只留下得分在指定百分比之前的特徵（取top k%）。

- 單變量特徵選擇範例程式

```
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2,k=20)
selector = selector.fit(X_train, Y_train)
X_train = selector.transform(X_train)
X_test = selector.transform(X_test)
```

調整模型參數

› 調整模型參數

```
# 特徵選擇
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2,k=20)
selector = selector.fit(X_train, Y_train)
X_train = selector.transform(X_train)
X_test = selector.transform(X_test)

# 建立模型
from sklearn.ensemble import RandomForestClassifier
randomForest = RandomForestClassifier(n_estimators=120,
criterion=‘entropy', class_weight='balanced',
n_jobs=-1,verbose=1)
```

模型訓練與評估

› 模型訓練與評估

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:    0.0s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 120 out of 120 | elapsed:    0.0s finished
Accuracy: 0.9824561403508771
```