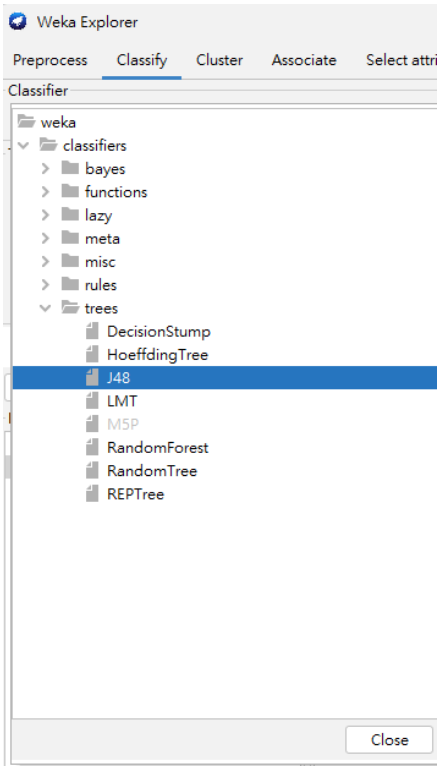
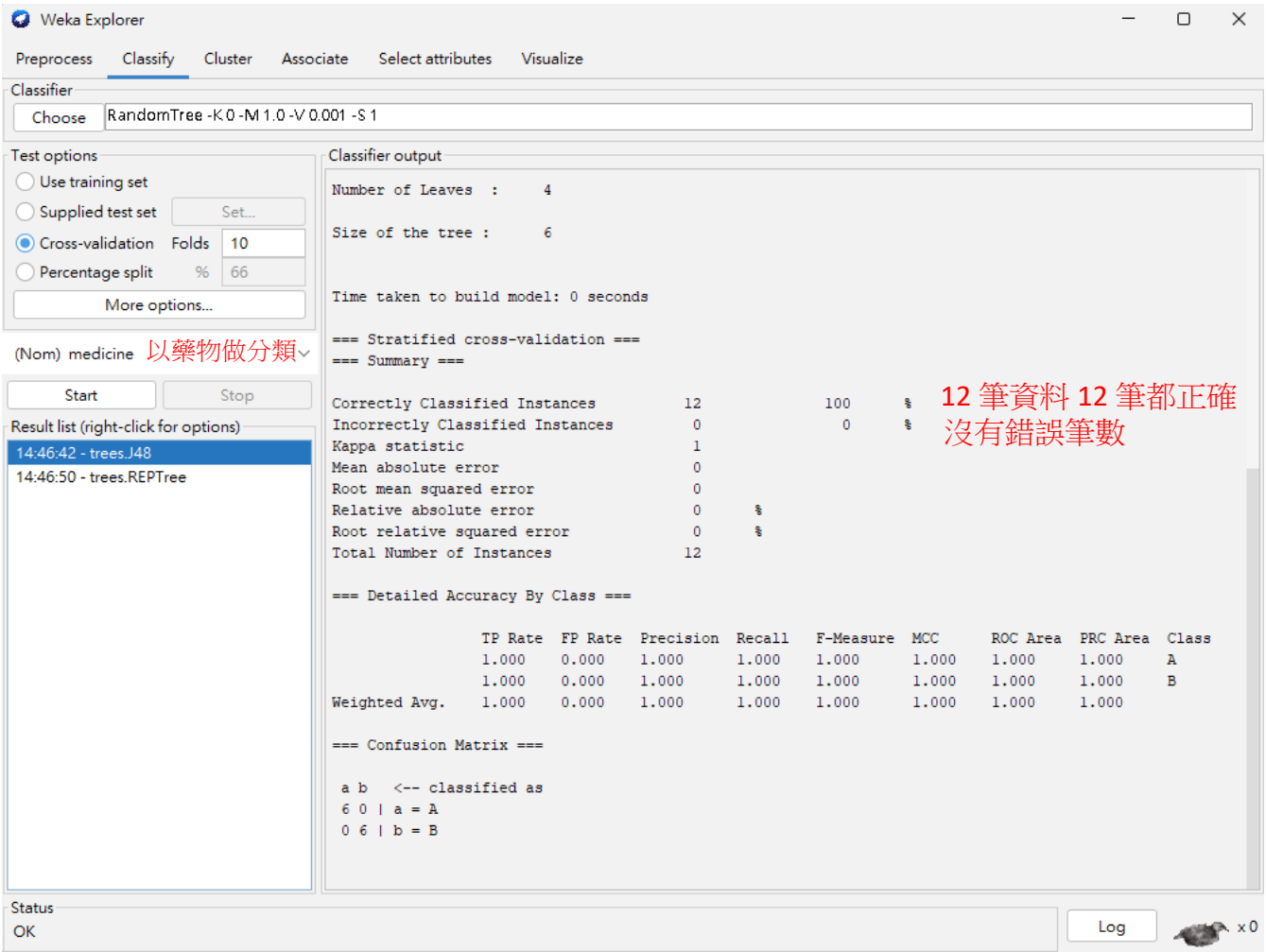
決策樹在 Weka 的範例

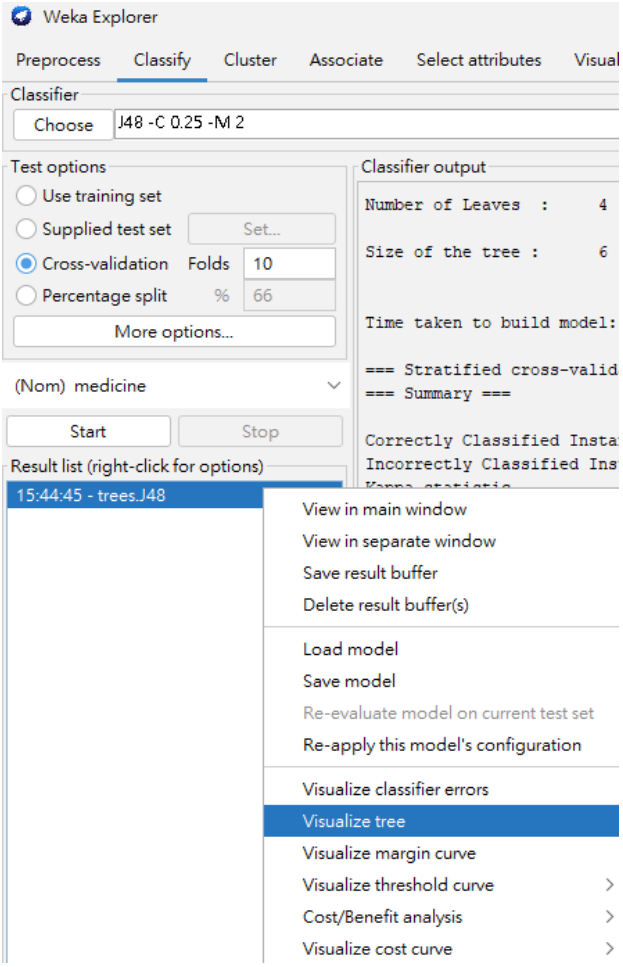
1. 藥物決策樹

先放入資料選擇 Classify > tree > J48 > start

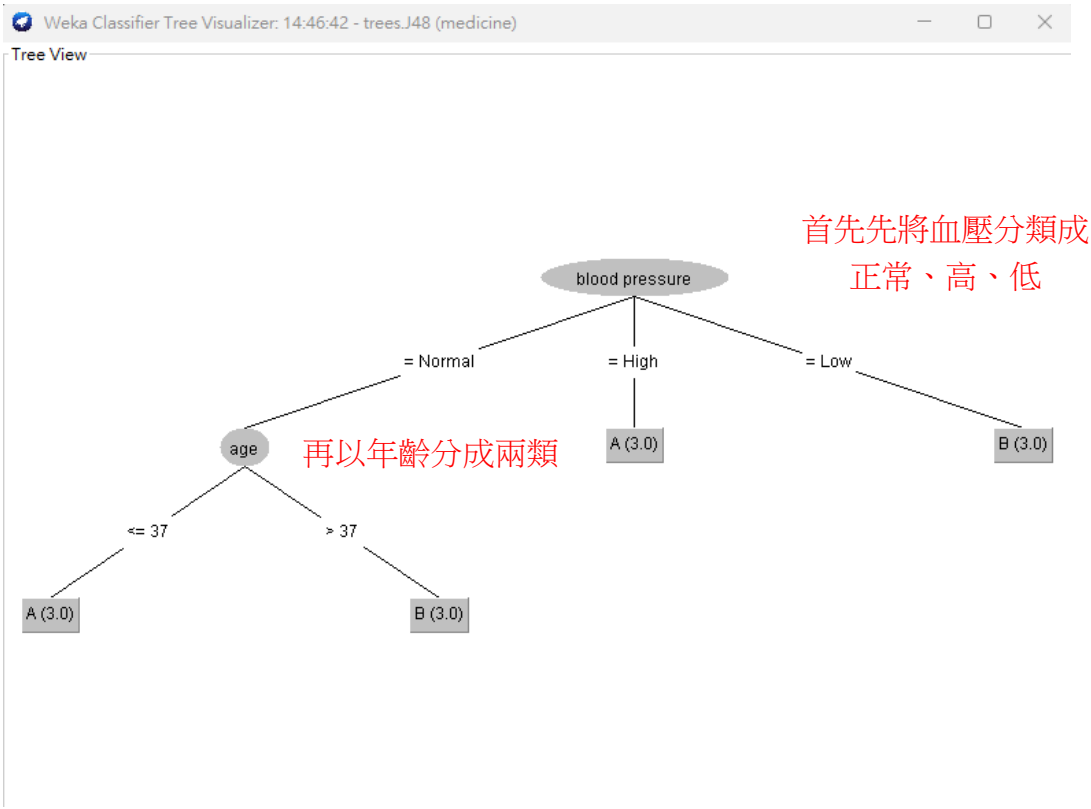




執行結果按右鍵，選擇 Visualize tree，瀏覽分類樹結果



分類數



2. 鳶尾花決策樹

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier

Choose: J48 -C 0.25 -M 2

Test options

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds: 10
- Percentage split — %: 66

More options...

(Nom) class

Start | Stop

Result list (right-click for options)

- 15:08:25 - rules.ZeroR
- 15:08:39 - trees.J48

Classifier output

```
Size of the tree :      9


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.980    0.000    1.000      0.980    0.990      0.985    0.990     0.987     Iris-seto
                 0.940    0.030    0.940      0.940    0.940      0.910    0.952     0.880     Iris-vers
                 0.960    0.030    0.941      0.960    0.950      0.925    0.961     0.905     Iris-virg
Weighted Avg.    0.960    0.020    0.960      0.960    0.960      0.940    0.968     0.924

=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

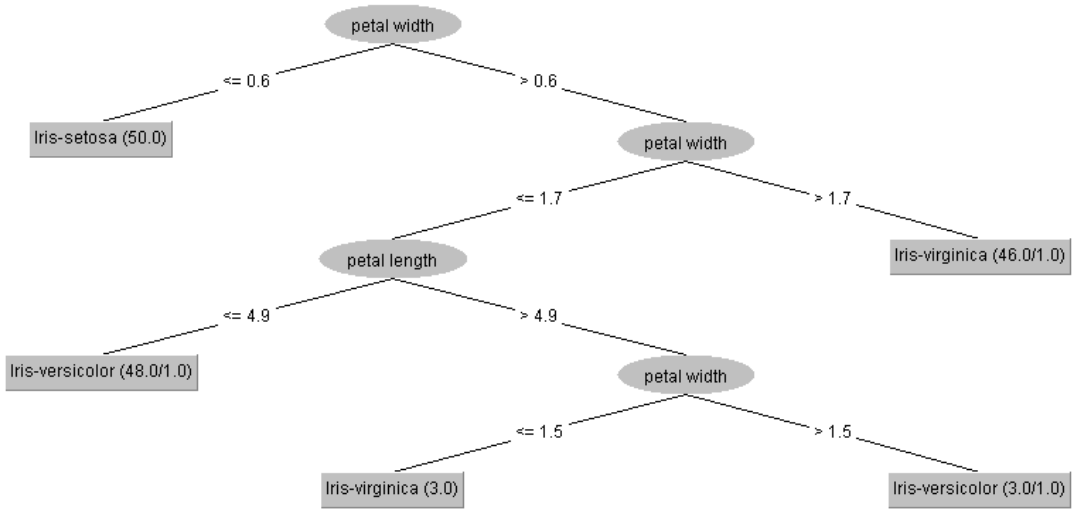
150 筆資料 144 筆正確

150 筆資料 6 筆錯誤

Status: OK | Log | x 0

Weka Classifier Tree Visualizer: 15:08:39 - trees.J48 (iris)

Tree View



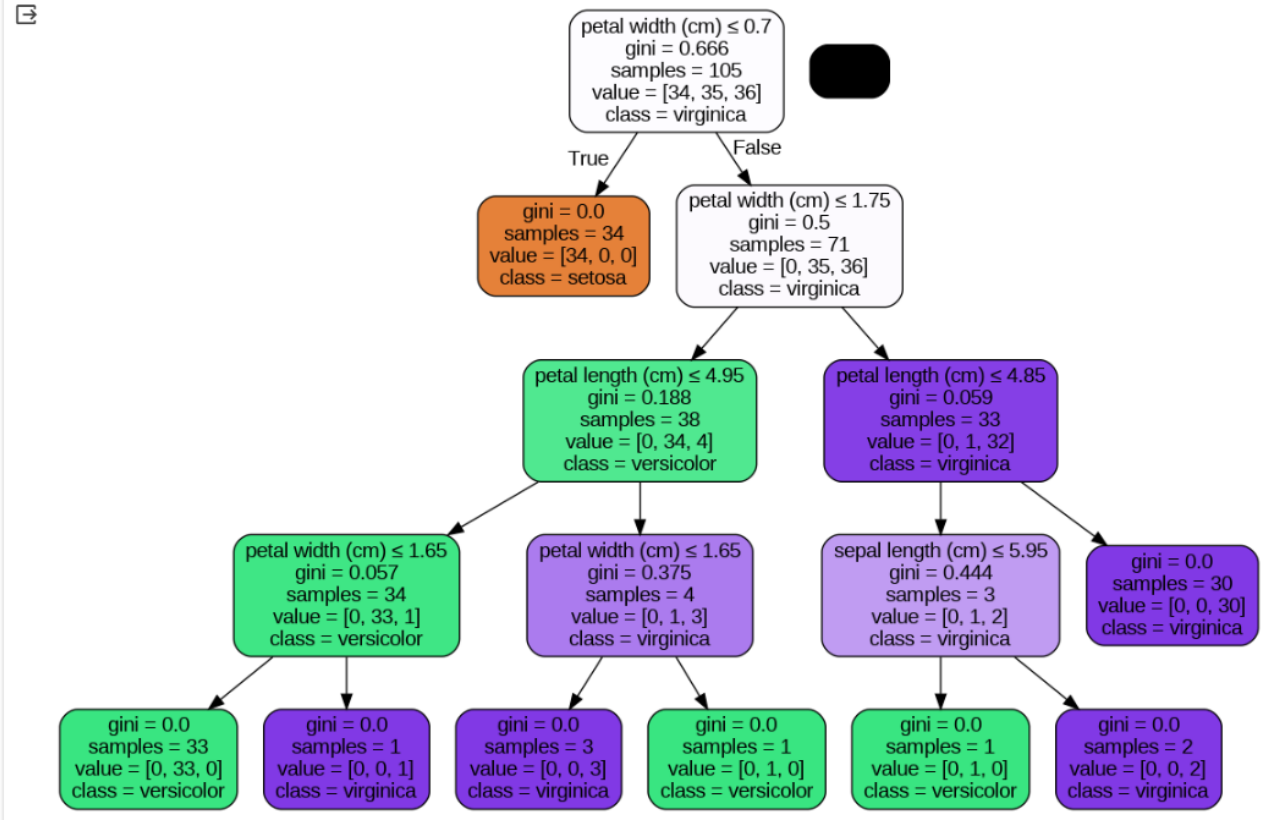
```
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# 讀入鳶尾花資料
iris = load_iris()
iris_data = iris.data
iris_label = iris.target

# 切分訓練與測試資料
train_data, test_data, train_label, test_label = train_test_split(iris_data, iris_label, test_size = 0.3)

# 建立分類器
clf = tree.DecisionTreeClassifier()
iris_clf = clf.fit(train_data, train_label)

print("訓練資料正確率:",clf.score(train_data,train_label)) # 訓練資料正確率
print("測試資料正確率:",clf.score(test_data, test_label)) # 測試資料正確率

predict=clf.predict(test_data) # 預測
print("標準答案:",test_label) # 標準答案
print("預測答案:",predict) # 預測答案
```

```
訓練資料正確率: 1.0
測試資料正確率: 0.977777777777777
標準答案: [0 0 2 1 2 2 2 1 0 2 2 0 1 1 2 0 1 0 2 1 2 0 0 1 1 0 1 2 1 0 2 1 2 0 1 1 1
 1 2 2 0 0 0 0 0]
預測答案: [0 0 2 1 2 2 2 1 0 2 2 0 1 1 2 0 1 0 2 1 2 0 0 1 1 0 1 2 1 0 2 1 2 0 1 1 2
 1 2 2 0 0 0 0 0]
```

[2]
```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris") #將結果存成pdf檔案
```

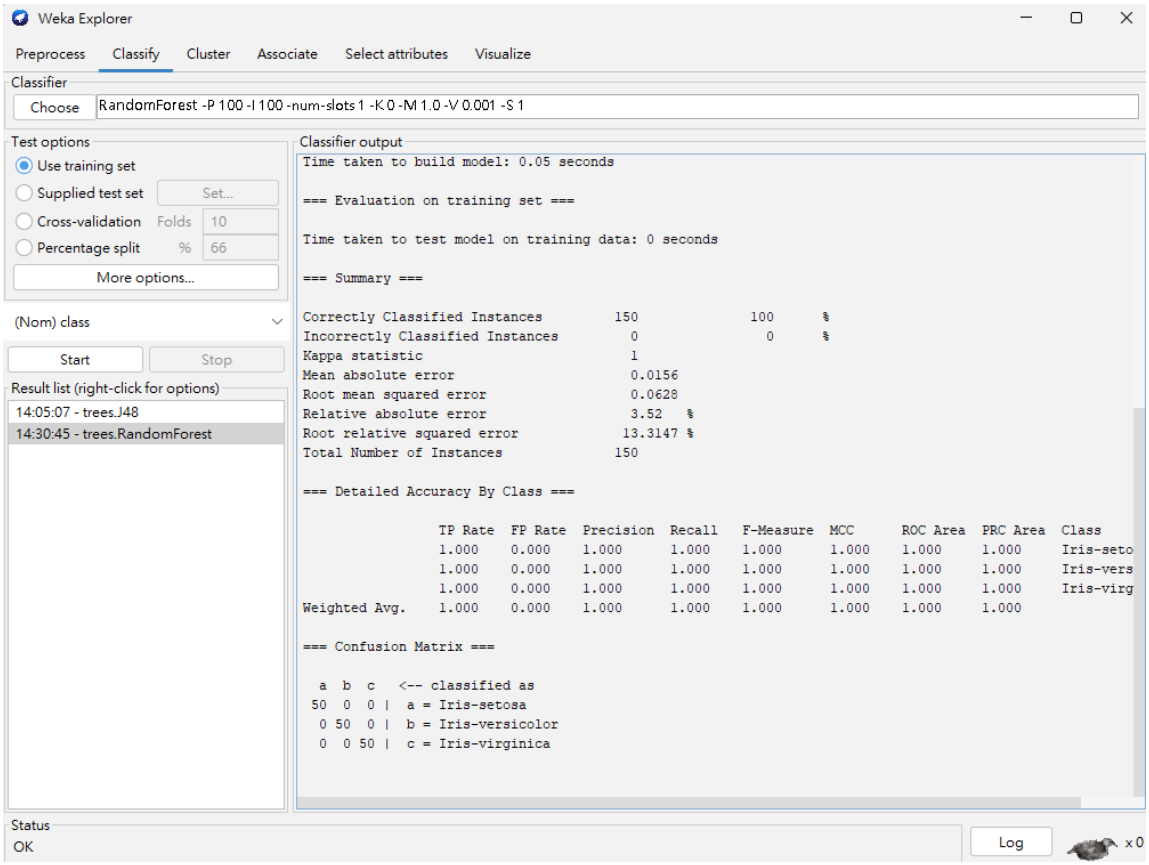
```
'iris.pdf'
```

```
import pydotplus
from IPython.display import Image
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())    #將結果用圖形化呈現
```

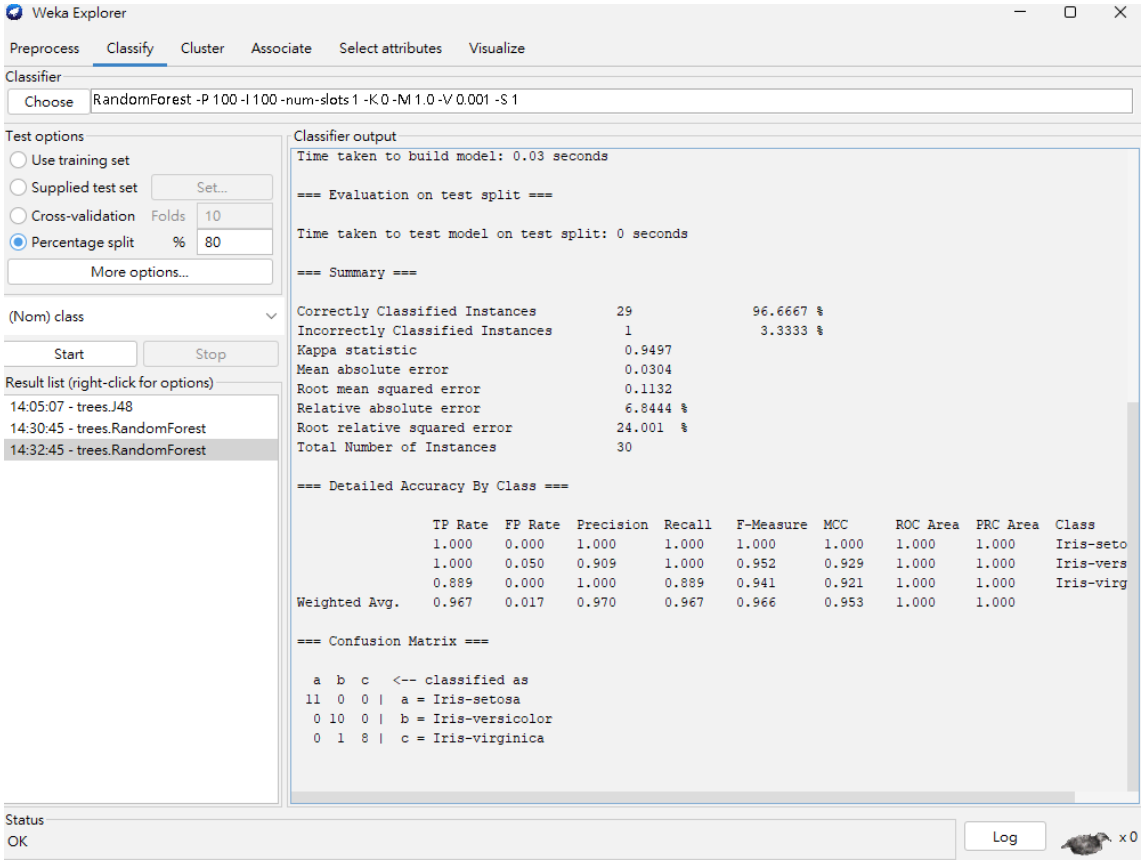


使用 Randow Forest 測試資料

完全使用測試資料 資料會 100%正確



使用 80%的測試資料 會有 1 比錯誤資料



產生 create a dataframe with 2 columns and 10 rows

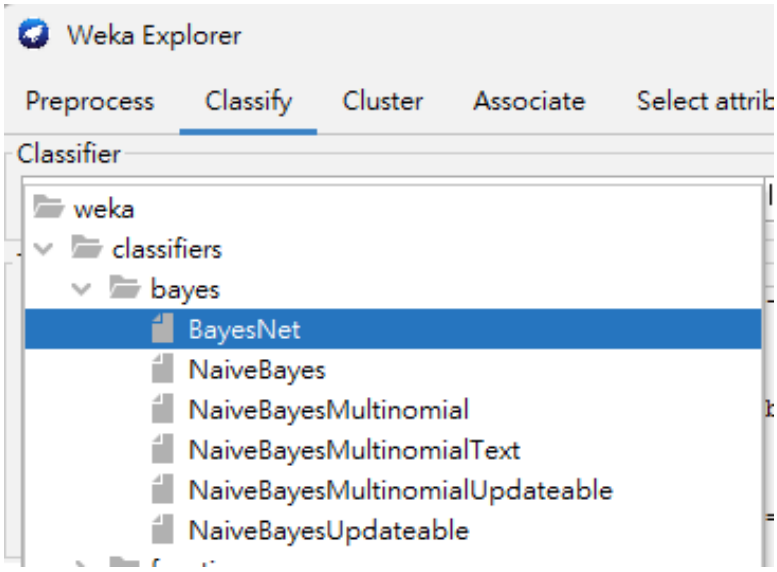
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# 讀入鳶尾花資料
iris = load_iris()
iris_data = iris.data
iris_label = iris.target

# 切分訓練與測試資料
train_data, test_data, train_label, test_label = train_test_split(iris_data, iris_label, test_size = 0.3)

# 建立分類器
clf = RandomForestClassifier(n_estimators=30, max_depth=4)
iris_clf = clf.fit(train_data, train_label)
print("訓練資料正確率:",clf.score(train_data,train_label)) # 訓練資料正確率
print("測試資料正確率:",clf.score(test_data, test_label)) # 測試資料正確率

predict=clf.predict(test_data) # 預測
print("標準答案:",test_label) # 標準答案
print("預測答案:",predict) # 預測答案
```

```
訓練資料正確率: 1.0
測試資料正確率: 0.9333333333333333
標準答案: [2 1 2 2 1 2 1 0 0 0 0 1 0 2 0 0 2 0 0 2 2 2 0 2 1 0 1 2 1 1 2 1 2 1 0 2 1
 1 0 0 1 2 0 1 1]
預測答案: [2 1 2 2 1 1 2 0 0 0 0 2 0 2 0 0 2 0 0 2 2 2 0 2 1 0 1 2 1 1 2 1 2 1 0 2 1
 1 0 0 1 2 0 1 1]
```

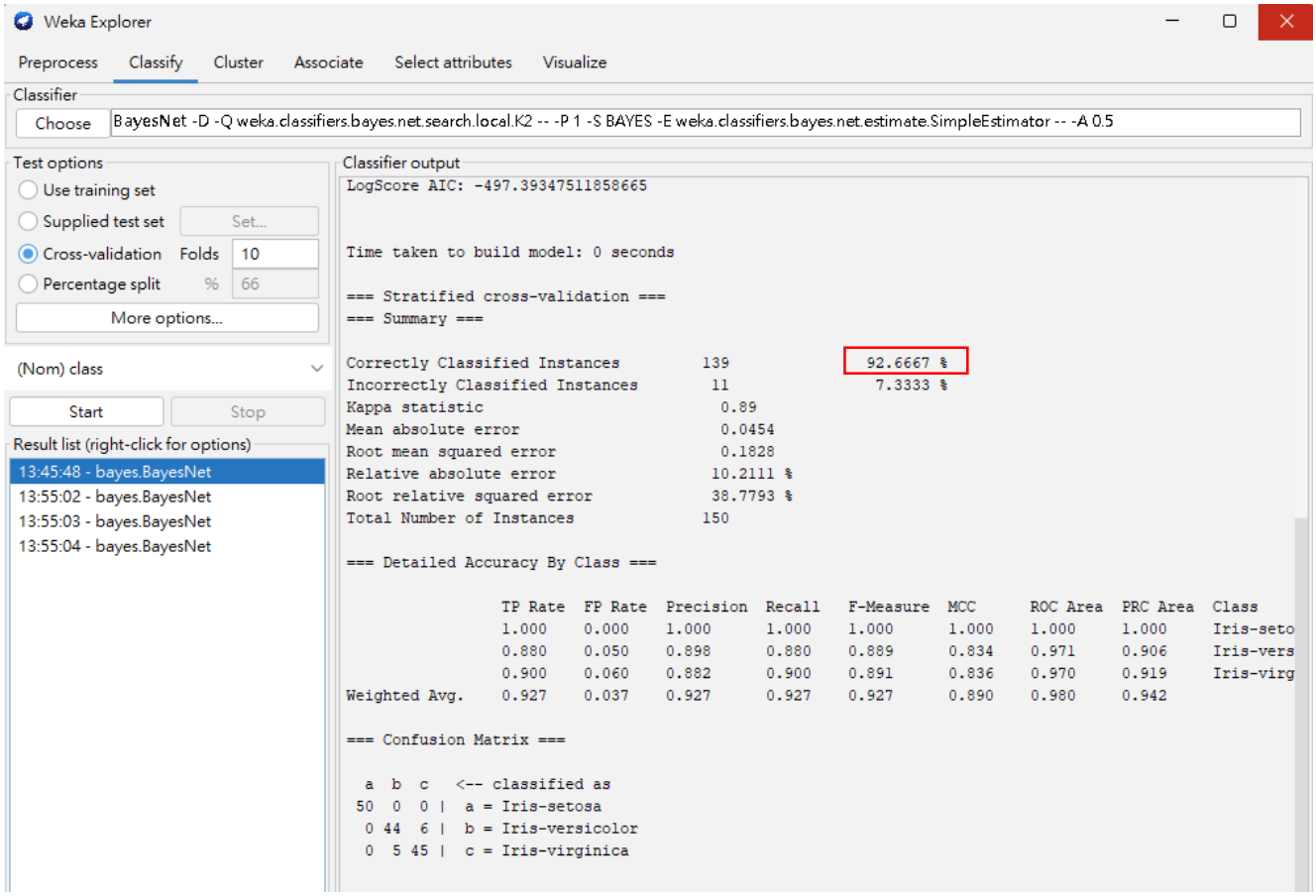
Naive Bayesian Classifier 貝氏分類器

1. 開啟開啟 Iris.csv

2. 選擇 Classify -> bayes BayesNet



3. 測試出結果正確率為 92.6667%



4. 查看圖形結果　執行結果右鍵 -> Visualize graph