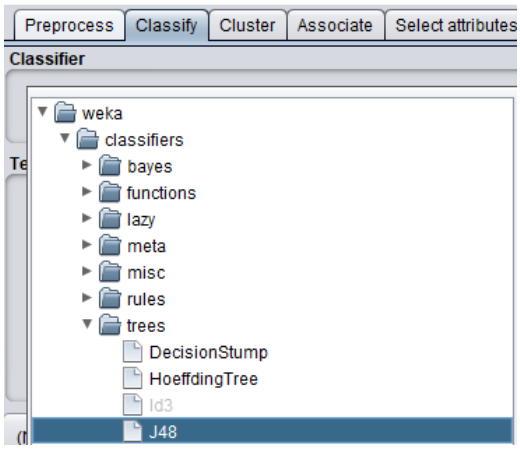
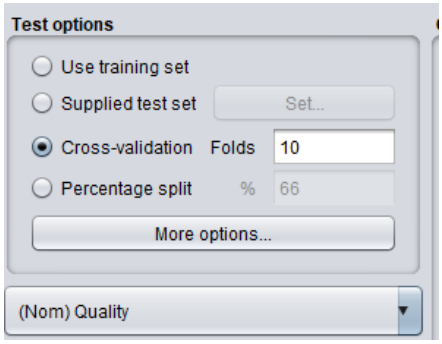
ECT HW8

Weka Part:

(a)



➔ 至 Classify 中，找到題目要求的 J48

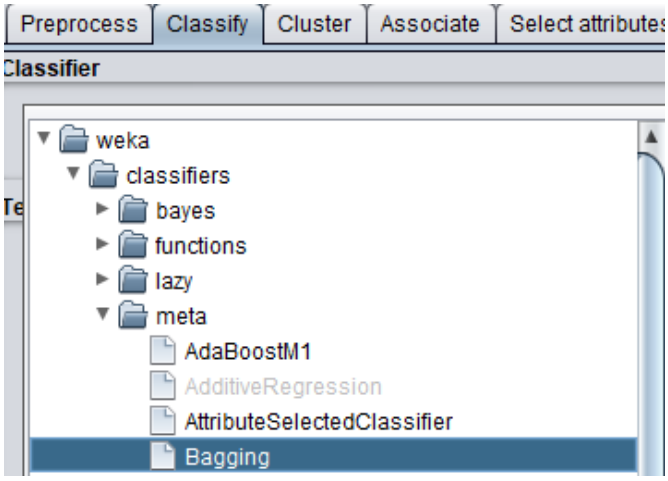


➔ 依照題目要求 Folds = 10，其餘皆使用默認參數

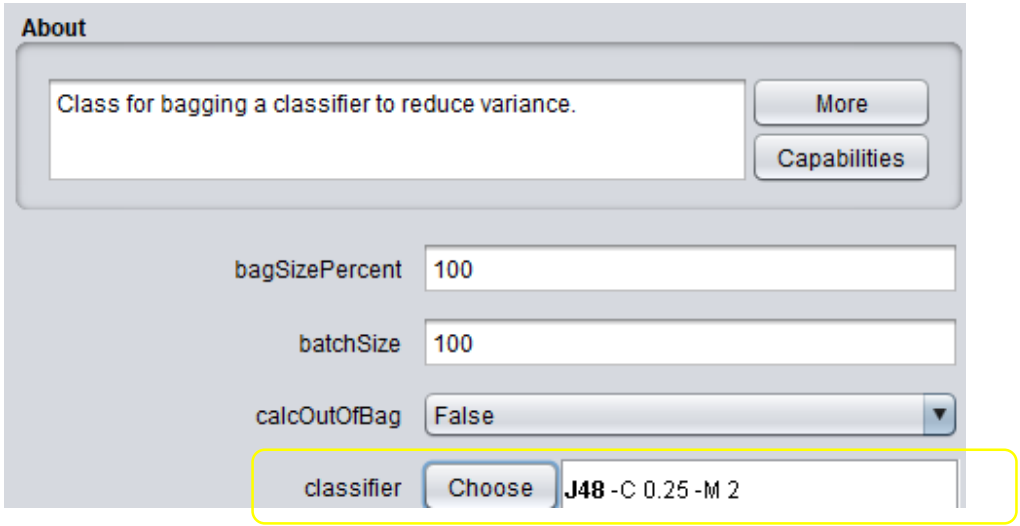
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         167               93.8202 %
Incorrectly Classified Instances        11                6.1798 %
Kappa statistic                          0.9058
Mean absolute error                      0.0486
Root mean squared error                  0.2019
Relative absolute error                 11.0723 %
Root relative squared error             43.0865 %
Total Number of Instances              178
```

➔ 分析結果如圖所示，有 93.8202%正確率、RMSE = 0.2019

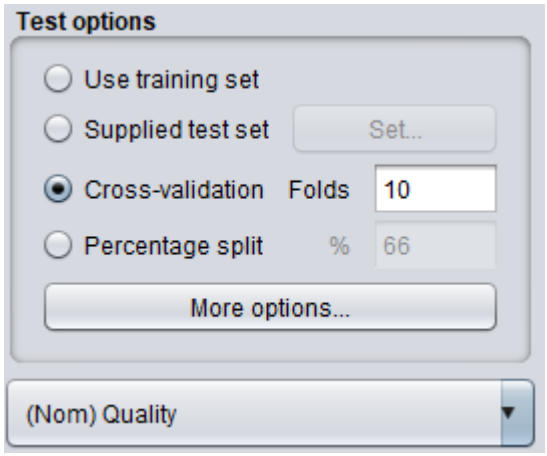
(b)



➔ 在 meta 底下找到題目要求的 Bagging



➔ 將其 classifier 設定為題目要求的 J48，其餘使用默認選項

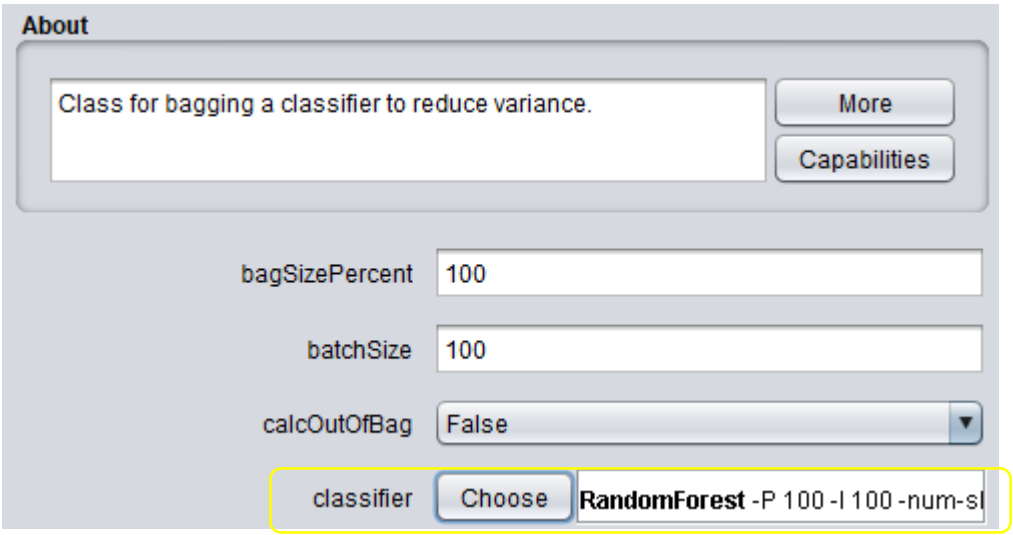


➔ Folds 仍然設為 10

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

Correctly Classified Instances         167               93.8202 %
Incorrectly Classified Instances        11                6.1798 %
Kappa statistic                          0.9061
Mean absolute error                      0.074
Root mean squared error                  0.1777
Relative absolute error                 16.8619 %
Root relative squared error             37.9221 %
Total Number of Instances              178
```

➔ 雖然準確率跟(a)小題一樣，仍然是 93.8202%，但 Root Mean Square Error 有下降至 0.1777，代表他其實是有進步的，雖然不明顯。

(c)

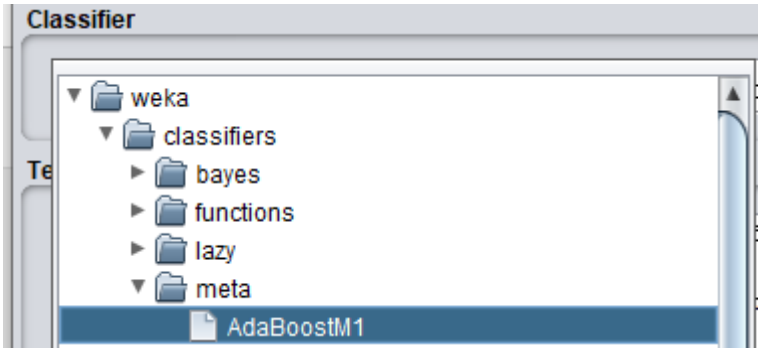


➔ 依照題目要求，這次一樣用 Bagging，但 classifier 改成 RandomForest

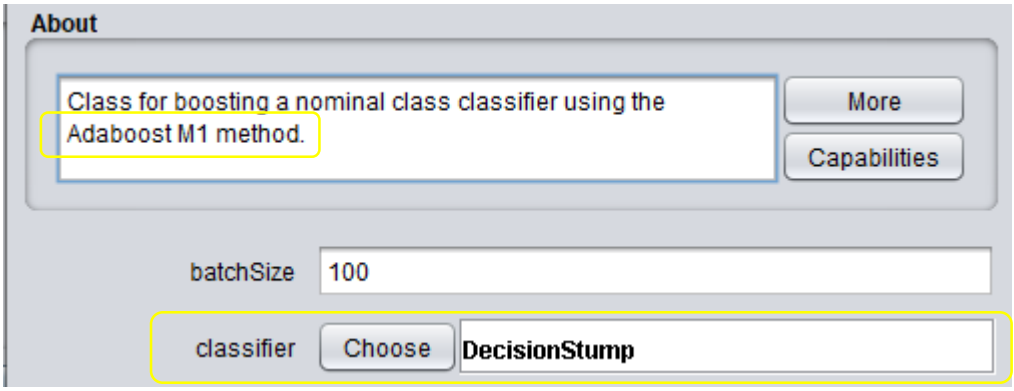
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         175               98.3146 %
Incorrectly Classified Instances         3                1.6854 %
Kappa statistic                          0.9745
Mean absolute error                      0.0781
Root mean squared error                  0.1404
Relative absolute error                 17.7862 %
Root relative squared error             29.9769 %
Total Number of Instances              178
```

➔ 可以看出跟前面 2 個相比，準確率大大的上升了，RMSE 也穩定下降中

(d)

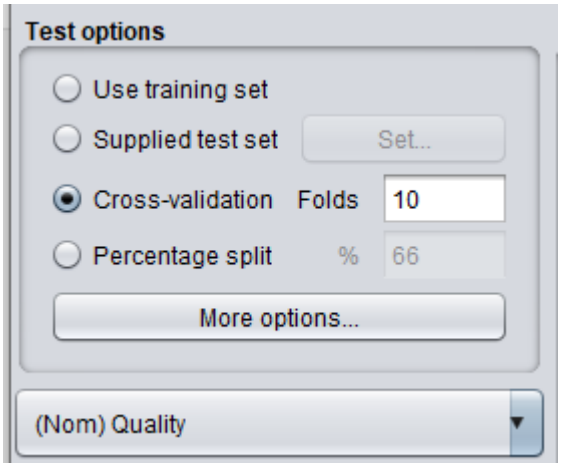


➔ 一樣繼續在 meta 底下，找到題目要求的 AdaBoost，Weka 中叫做 AdaBoostM1。



➔ Classifier 選擇題目要求的 DecisionStump

➔ 可以從 About 中看出，他的確是 Adaboost 的方法，只是用 M1 Method



➔ Folds 仍然是 10

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

Correctly Classified Instances         163               91.573  %
Incorrectly Classified Instances        15                8.427  %
Kappa statistic                          0.8728
Mean absolute error                      0.1303
Root mean squared error                  0.2188
Relative absolute error                 29.6833 %
Root relative squared error             46.7084 %
Total Number of Instances              178
```

➔ 準確率比起之前任何方法都低，來到了 91.573%，RMSE 也提高至 0.2188，可見得這次的處理不是個好的選擇

Python Part:

前置處理:

```
import pandas as pd
df = pd.read_csv('Wine.csv')
df
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline	Quality
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740	3
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750	3
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835	3
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840	3
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560	3

➔ 載入資料集

```
# input
x = df.loc[:,"Alcohol":"Proline"].values
print(x)
# output
y = df['Quality'].values
print(y)
```

➔ 將資料集切分為 input、output，因題目要求 target 為 Quality，所以 output 為資料集裡面的「Quality」，其他為 input

(a)

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
clf_DT = DecisionTreeClassifier(random_state=0)
```

➔ Import 相關套件，並建立 DecisionTreeClassifier。

- 在此份作業中 random_state 都設為 0，因為題目中並未要求 seed 設為多少，為求方便並告知了解 seed 的概念，因此不是使用默認，而是統一設為 0

```
total_cv_score = cross_val_score(clf_DT, x, y, cv=10)
avg_cv_score = total_cv_score.mean()
print("total_cv_score:", total_cv_score)
print("avg_cv_score:", avg_cv_score)
```

```
total_cv_score: [0.88888889 0.88888889 0.72222222 0.88888889 0.83333333 0.83333333
 1.         0.94444444 0.94117647 0.76470588]
avg_cv_score: 0.8705882352941178
```

➔ 使用 cross_val_score()並設置 cv = 10 代表 10 Folds cross-validation。

➔ 第 1 個參數為 Classifier，其次為 input、output

➔ 因為用這個方法算出來的是每個 Folds 的準確率，依照上課所學，會把這些數值平均起來當作準確率，因此調用 mean()函數

➔ 如上圖所示，平均準確率 = 0.8705882352941178

(b)

```
from sklearn.ensemble import BaggingClassifier
clf_BG = BaggingClassifier(n_estimators=10, random_state=0) #使用默認的 base_estimator = decision tree
```

➔ Import 套件，並建立 BaggingClassifier

➔ 依照題目要求 n_estimator = 10, random_state 如(a)小題所提及，設為 0

➔ 因為題目並未要求指定 base_estimator，因此使用默認的 decision tree

base_estimator : *object, default=None*

- The base estimator to fit on random subsets of the dataset. If None, then the base estimator is a decision tree.

```
total_cv_score = cross_val_score(clf_BG, x, y, cv=10)
avg_cv_score = total_cv_score.mean()
print("total_cv_score:", total_cv_score)
print("avg_cv_score:", avg_cv_score)
```

```
total_cv_score: [0.94444444 0.83333333 0.77777778 0.94444444 0.94444444 1.
 1.         0.94444444 1.         1.        ]
avg_cv_score: 0.9388888888888889
```

➔ 一樣使用 10 Folds cross-validation，並印出所每個 Fold 的準確率、平均準確率

- 平均準確率 = 0.9388888888888889，比(a)進步了許多，推測是因為使用 Bagging 參考多次平均的原因

(c)

```
from sklearn.ensemble import RandomForestClassifier
clf_RF = RandomForestClassifier(random_state=0)
```

➔ Import 套件，並建立 RandomForestClassifier，random_state = 0

```
total_cv_score = cross_val_score(clf_RF, x, y, cv=10)
avg_cv_score = total_cv_score.mean()
print("total_cv_score:", total_cv_score)
print("avg_cv_score:", avg_cv_score)
```

```
total_cv_score: [0.94444444 1.         0.94444444 0.94444444 1.         1.
 1.         1.         1.         1.        ]
avg_cv_score: 0.9833333333333332
```

➔ 一樣進行 10 Folds cross-validation 分析

- 平均準確率來了 0.9833333333333332 !比前面兩提高了許多，已經接近 100%了，可見在此例中使用 RandomForestClassifier 是一個好的方案，可以多方嘗試看有沒有機會至 100%

(d)

```
from sklearn.ensemble import AdaBoostClassifier
clf_AdaB = AdaBoostClassifier(n_estimators=10, random_state=0)
```

➔ Import 套件，並建立 AdaBoostClassifier

➔ 依照題目要求 n_estimators=10, random_state 仍設為 0

```
total_cv_score = cross_val_score(clf_AdaB, x, y, cv=10)
avg_cv_score = total_cv_score.mean()
print("total_cv_score:", total_cv_score)
print("avg_cv_score:", avg_cv_score)
```

```
total_cv_score: [0.55555556 1.         1.         0.88888889 0.88888889 0.94444444
 0.72222222 0.94444444 1.         1.        ]
avg_cv_score: 0.8944444444444445
```

➔ 進行 10 Folds cross-validation 分析

- 平均準確度只有 0.8944444444444445，帶比起(a)小題純粹使用 DecisionTreeClassifier 得到 0.8705882352941178，還是有些許的上升