

HW8_109403021

步驟 2：建立模型、 MLP：

選此建立 MLP 模型

切割 80% 為訓練集

預測 Class

Classifier: MultilayerPerceptron

Test options: ☒ Use training set, ☐ Supplied test set, ☐ Cross-validation, ☒ Percentage split

Percentage split: % 80

Predict: (Nom) Class

Result list (right-click for options):

- 14:20:28 - functions.MultilayerPerceptron
- 14:25:17 - functions.MultilayerPerceptron
- 14:27:24 - functions.MultilayerPerceptron

Classifier output:

```
Node 21 -0.7719938040955967
Class 0
Input
Node 0
Class 1
Input
Node 1
```

Time taken to build model: 146.98 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.03 seconds

=== Summary ===

Correctly Classified Instances	448	98.2456 %
Incorrectly Classified Instances	8	1.7544 %
Kappa statistic	0.9512	
Mean absolute error	0.0307	
Root mean squared error	0.1299	
Relative absolute error	8.2946 %	
Root relative squared error	30.4378 %	
Total Number of Instances	456	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.945	0.006	0.981	0.945	0.963	0.951	0.994	0.983	0
	0.994	0.055	0.983	0.994	0.989	0.951	0.994	0.998	1
Weighted Avg.	0.982	0.043	0.982	0.982	0.982	0.951	0.994	0.995	

=== Confusion Matrix ===

```
a b <-- classified as
103 6 | a = 0
2 345 | b = 1
```

Status: OK

使用我用 python 前處理完的 output.csv 來建立模型並訓練，以上是得到的結果

weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

More

Capabilities

GUI True

autoBuild True

batchSize 128

debug False

decay False

doNotCheckCapabilities False

hiddenLayers 16, 4, 8

learningRate 0.03

momentum 0.3

nominalToBinaryFilter True

normalizeAttributes True

normalizeNumericClass True

numDecimalPlaces 2

reset False

resume False

seed 10

trainingTime 500

validationSetSize 0

validationThreshold 20

Open... Save... OK Cancel

這裡主要調整了：

超參數：學習率(learningRate)和動量(momentum)，把學習率調得比默認值更小、動量調得比默認值更大

設置了三層隱藏層：16, 4, 8

RandomForest:

選此建立 Random
Forest 模型

切割 80%為訓練集

預測 Class

The screenshot shows the Weka Explorer interface. The 'Classifier' tab is selected, and 'RandomForest' is chosen from the 'Choose' dropdown. The 'Test options' section has 'Percentage split' selected with a value of 80. The 'Result list' on the left shows a list of classifiers, with '14:45:10 - trees.RandomForest' selected. The 'Classifier output' pane on the right displays the following information:

```
=== Classifier model (full training set) ===
RandomForest
Bagging with 100 iterations and base learner
weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 20 -do-not-check-capabilities -batch-size 128
Time taken to build model: 2.96 seconds

=== Evaluation on test split ===
Time taken to test model on test split: 0.02 seconds

=== Summary ===
Correctly Classified Instances      449      98.4649 %
Incorrectly Classified Instances     7        1.5351 %
Kappa statistic                     0.9569
Mean absolute error                  0.0618
Root mean squared error              0.1359
Relative absolute error              16.6993 %
Root relative squared error          31.8486 %
Total Number of Instances           456

=== Detailed Accuracy By Class ===

```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0.985	0.049	0.985	0.985	0.984	0.958	0.999	0.999	

```

=== Confusion Matrix ===
      a  b  <-- classified as
102   7 |  a = 0
 0 347 |  b = 1

```

Annotations in red:

- '選此建立 Random Forest 模型' points to the 'RandomForest' selection in the 'Classifier' dropdown.
- '切割 80%為訓練集' points to the 'Percentage split' option and the '80' value.
- '預測 Class' points to the '(Nom) Class' dropdown.
- '準確率' points to the '98.4649 %' value in the 'Summary' section.

使用我用 python 前處理完的 output.csv 來建立模型並訓練，以上是得到的結果

weka.gui.GenericObjectEditor

weka.classifiers.trees.RandomForest

About

Class for constructing a forest of random trees.

More

Capabilities

bagSizePercent 100

batchSize 128

breakTiesRandomly False

calcOutOfBag False

computeAttributeImportance False

debug False

doNotCheckCapabilities False

maxDepth 0

numDecimalPlaces 2

numExecutionSlots 1

numFeatures 0

numIterations 100

outputOutOfBagComplexityStatistics False

printClassifiers False

seed 20

storeOutOfBagPredictions False

Open... Save... OK Cancel

這裡的超參數與模型架構大部分皆使用默認值，僅稍微調整了 batchSize

CNN:

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize DL4j Inference

Classifier

Choose **DL4jMlpClassifier** -S 1 -cache-mode MEMORY -early-stopping "weka.dl4j.earlystopping.EarlyStopping -maxEpochsNoImprovement 0 -valPercentage 0.0" -normalization

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☒ Percentage split % 80

More options...

(Nom) Class

Start Stop

Result list (right-click for options)

- 16:40:59 - functions.DL4jMlpClassifier
- 16:44:06 - functions.DL4jMlpClassifier
- 16:55:37 - functions.DL4jMlpClassifier
- 16:55:58 - functions.DL4jMlpClassifier
- 17:08:15 - functions.DL4jMlpClassifier
- 17:11:28 - functions.DL4jMlpClassifier
- 17:20:24 - functions.DL4jMlpClassifier

Classifier output

Output layer (OutputLayer) 64,2 130 W:{64,2}, b:{1,2} [Dense layer]

Total Parameters: 57,794

Trainable Parameters: 57,794

Frozen Parameters: 0

Time taken to build model: 99.77 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.2 seconds

=== Summary ===

Correctly Classified Instances	449	98.4649 %
Incorrectly Classified Instances	7	1.5351 %
Kappa statistic	0.9582	
Mean absolute error	0.019	
Root mean squared error	0.1261	
Relative absolute error	5.145 %	
Root relative squared error	29.5529 %	
Total Number of Instances	456	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.982	0.014	0.955	0.982	0.968	0.958	0.995	0.988	0
	0.986	0.018	0.994	0.986	0.990	0.958	0.995	0.999	1
Weighted Avg.	0.985	0.017	0.985	0.985	0.985	0.958	0.995	0.996	

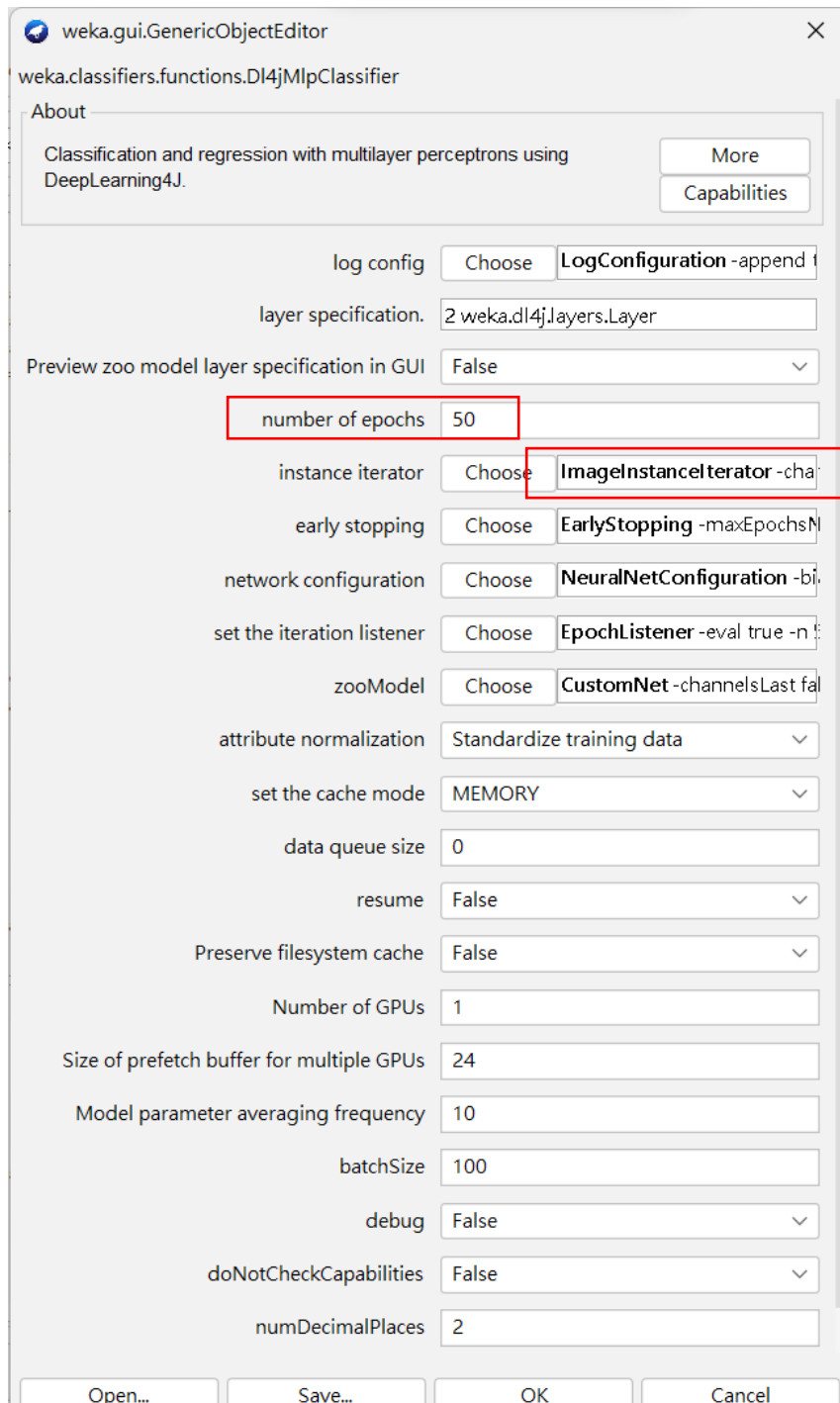
=== Confusion Matrix ===

a	b	<-- classified as
107	2	a = 0
5	342	b = 1

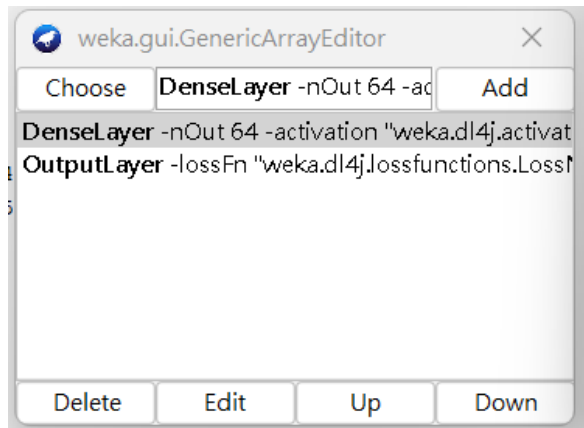
Status

OK

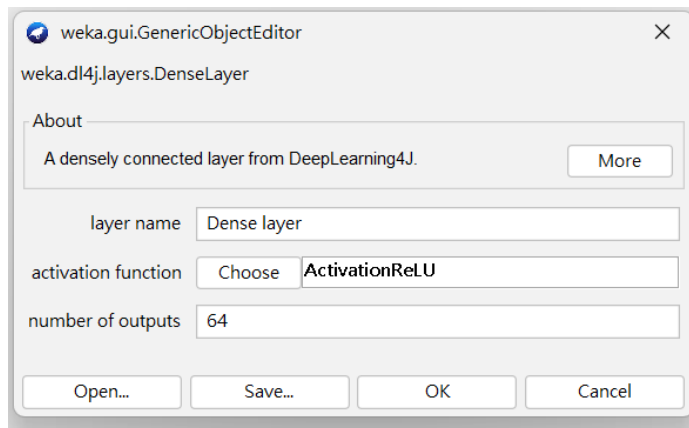
使用我用 python 前處理完的 output.csv 和輸出的圖檔來建立模型並訓練，以上是得到的結果



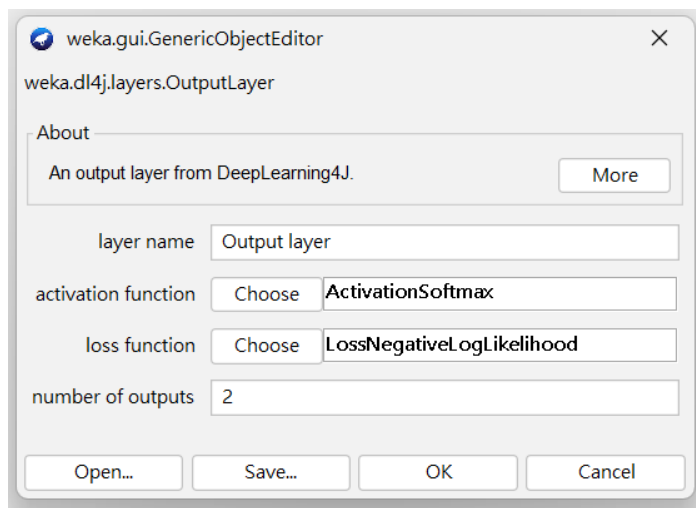
將 epochs 從默認值 10 提升到 50，instance iterator 選取 ImageInstanceIterator 並選到圖檔資料夾



模型這邊疊了一層 Dense Layer 和一層 Output Layer



Dense Layer 的參數設置



Output Layer 的參數設置

步驟 3：模型大比較、

1. 各模型不同超參數表現差異

以下各模型改變某一超參數值其他不變

模型與比較之超參數	超參數值	準確率
MLP-比較 learningRate	0.3	76.0965 %
	0.03	98.2456 %
RandomForest-比較 MaxDepth	0	98.0263 %
	1	76.0965 %
CNN-比較 batchSize	100	98.2456 %
	500	98.2456 %

2. 模型優缺點

	優點	缺點
MLP	靈活、易使用	易陷入局部最小值
RandomForest	不太會過擬合	內存消耗大
CNN	善於處理視覺數據	計算成本大

3. 改進之處

MLP:

可以用正則化技術減少過擬合風險、可以選擇更好的優化算法(Adam 等等)

RandomForest:

調整森林大小以平衡準確性和預測速度

CNN:

增加卷積層和全連接層來捕捉更多圖像特徵

4. 最後選擇

此次任務我會選擇用 RandomForest 模型，因為它的準確度表現最好而且速度最快