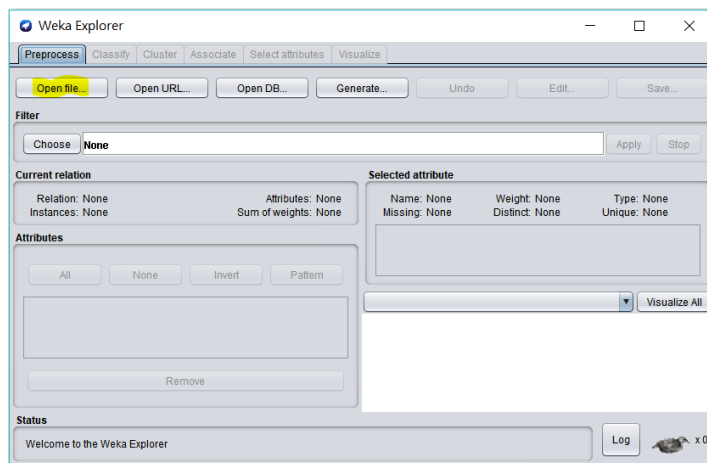


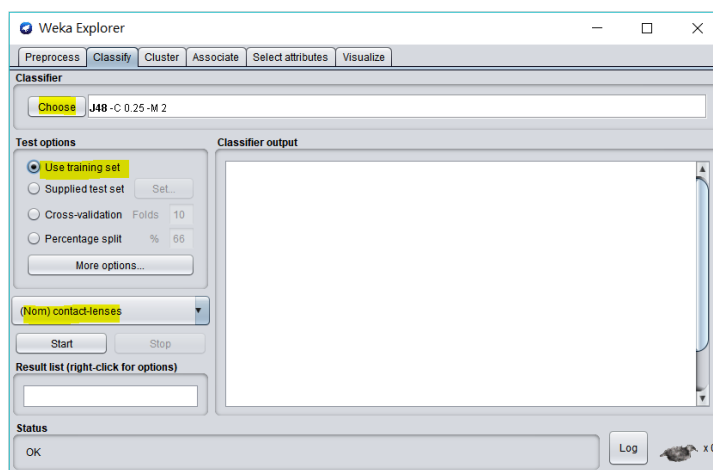
1. 用 Weka 軟體對 contact-lenses.arff 建立 J48 決策樹，選擇 “Use training set” ，設定 Attribute: contact-lenses 為 Output ，在過程中對重要步驟截圖並加以說明，並回答以下問題：

步驟

- i. 點選「Open file」，選擇檔案「contact-lenses.arff」



- ii. 在 Classifier 點選「Choose」，選擇 weka/classifiers/trees/J48 ；在 Test options 選擇「Use training set」，；在 Start 上方的欄位選擇「(Nom)contact-lenses」。



(a) 在前處理部分，右下角選擇不同屬性作為 Class，請解釋長條圖

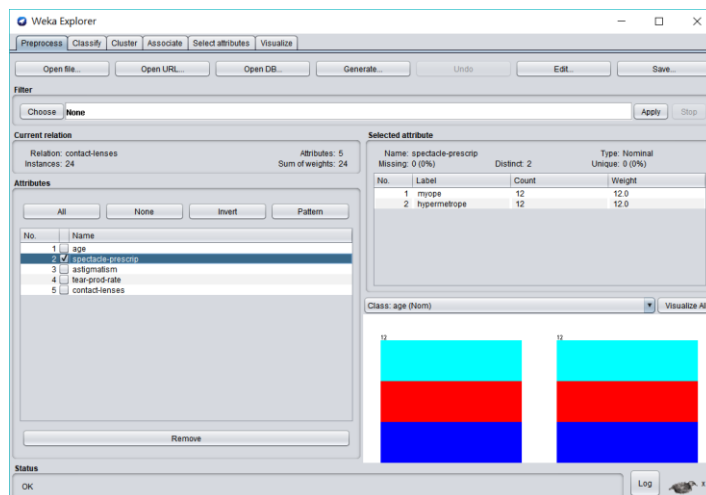
的數量、上方的數字以及不同顏色意義為何?(15%)

Attribute 屬性選擇了 spectacle-prescrip (包含了 myope, hypermetrope)

Class 選擇了 age (屬性值包含 young, pre-presbyopic, presbyopic) 得到的直條圖則代表 myope 中 young, pre-presbyopic, presbyopic 各出現了多少次、hypermetrope 中 young, pre-presbyopic, presbyopic 各出現了多少次。

因此，長條圖的數量由 Attribute 決定、上方的數字代表

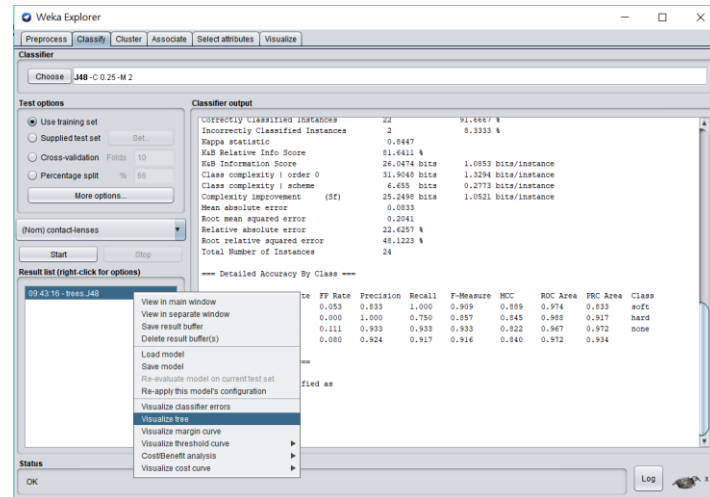
Attribute 中的各個屬性總共出現的次數、顏色種類由 Class 中的屬性值總數決定，每一種顏色代表 Class 中的一種屬性值。



(b) 使用 Visualize Tree 或 Classifier Output 列出三個

Classification Rule 並解釋。(20%)

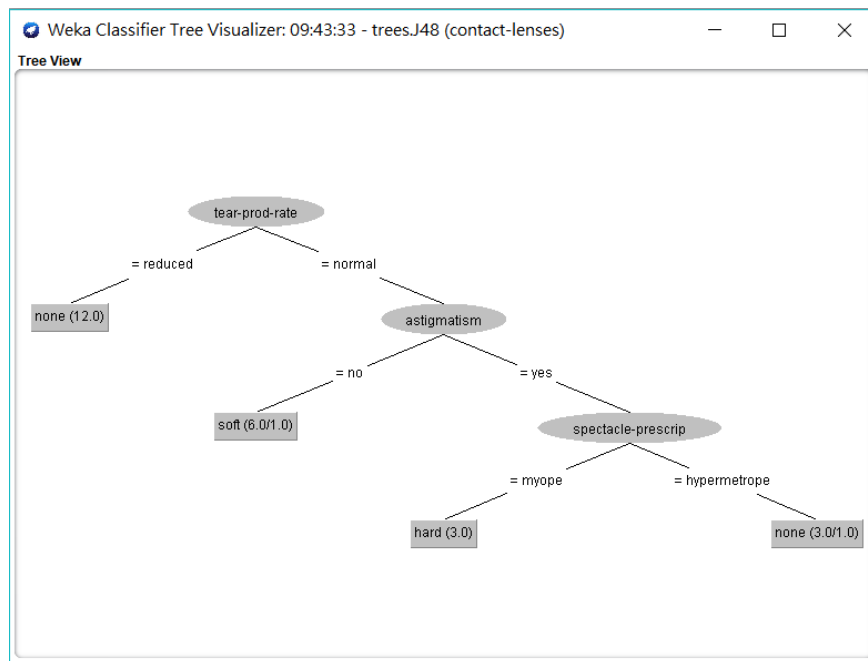
在 Result list 選取 Result，右鍵點選「Visualize tree」，即可看到決策樹。



Tree 中的每一個節點 node (橢圓形)，代表著不同的屬性測試，而最下方的葉 leaf (矩形)，則代表分類的結果。由圖可以知道第一個 Classification Rule 是由 tear-prod-rate 屬於 reduced 還是 normal 分類，第二個 Classification Rule 是由 astigmatism 屬於 no 還是 yes 分類，第三個 Classification Rule 是由 spectacle-prescrip 屬於 myope 還是 hypermetrope 分類。

J48 pruned tree

```
-----  
  
tear-prod-rate = reduced: none (12.0)  
tear-prod-rate = normal  
| astigmatism = no: soft (6.0/1.0)  
| astigmatism = yes  
| | spectacle-prescrip = myope: hard (3.0)  
| | spectacle-prescrip = hypermetrope: none (3.0/1.0)
```



2. 請利用 weka 和 python 對 glass.csv 進行 Supervised learning 中的 DecisionTree 分析 ,並回答以下問題:

- (a) 請運用 python 的 train_test_split 對 glass.csv 資料集 , 預測目標屬性為 Type 進行訓練集(66%)、測試集(34%)切分 , 請將重要程式碼截圖並說明(10%)

Python 部分 :

- i. 新增指令使用套件「graphviz」

```
!pip install --upgrade pip --user
!pip install graphviz
import pandas as pd
import graphviz
from sklearn import preprocessing
from sklearn import tree
from sklearn import metrics
```

- ii. 讀取 glass.csv 檔案、設定 x:input 與 y:output、將 type 轉為數字 label、將 x 屬性合併成 list

```
#讀取CSV檔案
data = pd.read_csv('glass.csv')
```

切分input 和output

```
#x:input
x=data.loc[:,['R1','Na','Hg','Al','Si','K','Ca','Ba','Fe']]
#y:output
y=data.loc[:,['Type']]
```

轉換屬性及標籤型別

```
#轉換屬性型別
#將屬性轉為數字Label
le = preprocessing.LabelEncoder()

#將type 轉為數字Label
#type: build wind float: 0 ,build wind non-float: 1 ,vehic wind float: 2
# containers: 3 ,headlamps: 4 ,tablewares: 5
Y_type_label=le.fit_transform(y['Type'])

#將屬性合併
#應列list
feature=list(zip(x.R1,x.Na,x.Hg,x.Al,x.Si,x.K,x.Ca,x.Ba,x.Fe))

#轉為array
import numpy as np
features=np.asarray(feature)
```

iii. 設定測試集為 0.34(34%)

切分訓練集與資料集

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, Y_type_label, test_size=0.34)
```

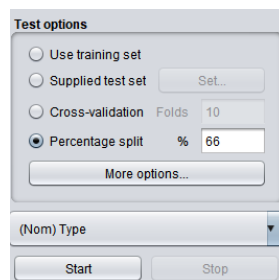
Weka 部分：

i. 點選「Open file」，選擇檔案「glass.arff」並在

Classifier 點選「Choose」，選擇

weka/classifiers/trees/J48。

ii. 選取「Percentage split」，輸入「66」。



(b) 請利用參數(criterion = 'entropy' , max_depth=3,

max_leaf_nodes = 4)對切分出的訓練集進行訓練，並用

metrics.accuracy_score()分別計算出模型對於訓練集和測試集

的精準度，並與 WEKA 設定演算法 J48 Percentage spilt 66%

跑出的結果截圖說明並一起呈現比較(20%)

Python 部分：

設定 max_depth=3, max_leaf_nodes=4。

下圖為使用 python 輸出的測試集精準度及訓練集精準度。

切分訓練集與資料集

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, Y_type_label, test_size=0.34)
```

sklearn: DecisionTreeClassifier

```
clf = tree.DecisionTreeClassifier(criterion = 'entropy',max_depth=3,max_leaf_nodes = 4)
glass_clf = clf.fit(X_train,y_train)
```

測試模型

```
test_predicted = glass_clf.predict(X_test)
train_predicted = glass_clf.predict(X_train)

# 驗證準確度
test_accuracy = metrics.accuracy_score(test_predicted, y_test)
train_accuracy = metrics.accuracy_score(train_predicted, y_train)
print("測試集精準度: ",test_accuracy)
print("訓練集精準度: ",train_accuracy)
```

```
測試集精準度: 0.589041095890411
訓練集精準度: 0.6950354609929078
```

Weka 部分：

下圖為 Weka 輸出的精準度。

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) Type

Start Stop

Result list (right-click for options)

12.24.31 - trees.J48

Classifier output

Number of Leaves : 30

Size of the tree : 59

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances	42	57.5342 %
Incorrectly Classified Instances	31	42.4658 %
Kappa statistic	0.4259	
Mean absolute error	0.1246	
Root mean squared error	0.3287	
Relative absolute error	59.7442 %	
Root relative squared error	101.9335 %	
Total Number of Instances	73	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.600	0.208	0.522	0.600	0.558	0.377	0.727	0.482	bus
	0.500	0.146	0.727	0.500	0.593	0.362	0.669	0.564	bus
	0.250	0.029	0.333	0.250	0.286	0.253	0.817	0.393	veh

Status

OK Log x0

比較兩圖可以發現 Python 與 Weka 輸出的精準度有差異，原因

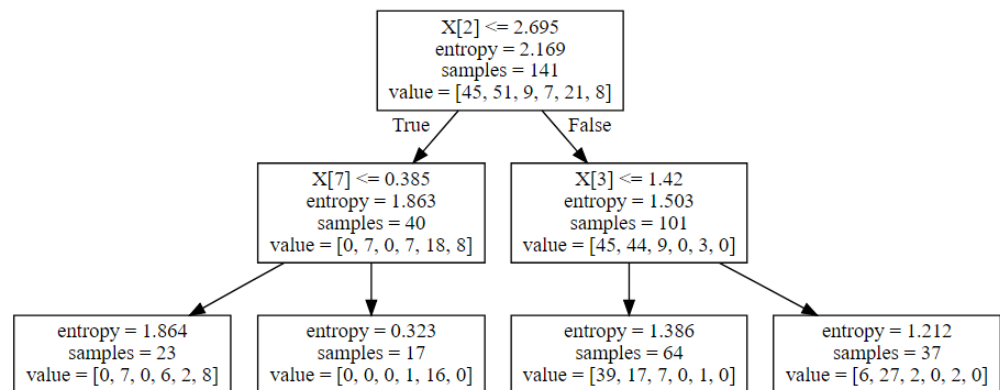
在於 Python 所切出的測試集與訓練集與 Weka 並不相同，所以結果也會不一樣。

(c) 請利用 graphviz 套件跑出決策樹圖形截圖並加以說明當中

X[?]、samples、及 value 各代表的訊息，另外運用 WEKA

visualize tree 觀察決策樹圖形並說明 WEKA 中葉節點的標籤

及括號內的數字代表的意義。(20%)



X[?] 表示使用哪些條件作為分類的區別。由 0-8 依序為

Rl, Na, Mg, Al, Si, K, Ca, Ba, Fe。例如：第一層 $X[2] \leq 2.169$ 表

示 Mg 的量小於等於 2.169，若符合則歸類到 True，不符合則

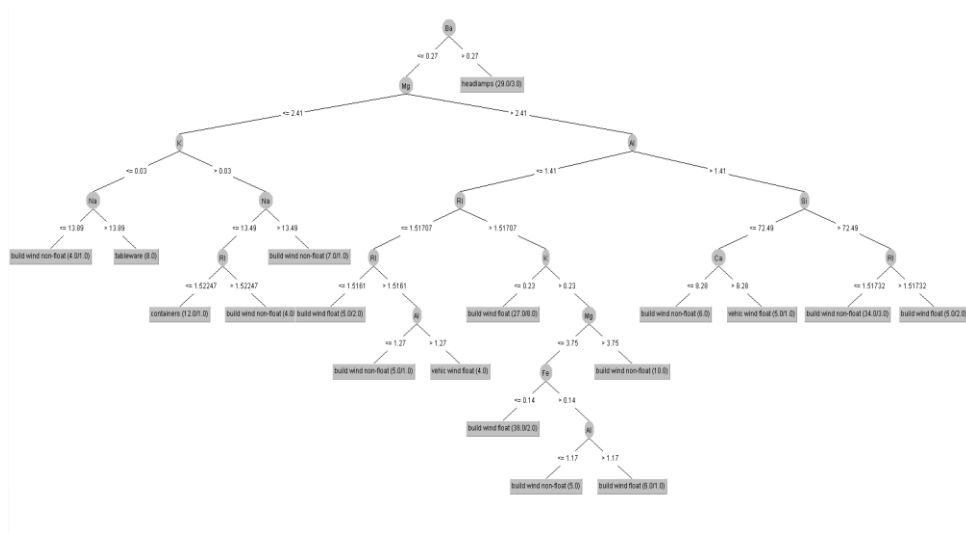
歸類到 FALSE，以此類推。

Sample 表示樣本數，因為我們將測試集設為 66% 因此這邊的

樣本數量為 $214 \times 0.66 = 141$ 。

Value 則是代表我們設定的 Attribute 各自有多少個樣本，從左

到右依序為 build wind float、build wind non-float、vehic
wind float、containers、headlamps、tableware。



Weka 跑出來的決策樹可以很直覺的知道分類的規則及依據，

如：節點的標籤（橢圓形）表示已什麼做為資料的劃分的依據，

而線上的判斷式則提供了分類條件。而方框的文字則代表

Output Attribute，方框內的括弧內的數字左邊表示符合的數

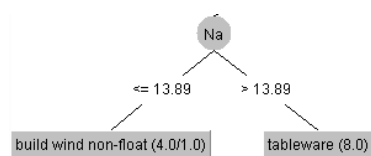
量，右邊則表示不符合的數量。

例如：Na 當作分類的依據，若 Na 小於或等於 13.89，則歸類到

build wind non-float，而有 4 筆是歸類正確，1 筆是歸類錯誤；

若 Na 大於 13.89，則歸類到 tableware，而全部 8 筆都是歸類正

確。



(d) 請試著調整 DecisionTreeClassifier 的參數，提升模型準確率，

請截圖並附上每次測試的結果(3~5 次)，觀察並說明準確率上升

的原因，另外說明模型在訓練集的準確度通常較測試集的準確度

高的原因為(15%)

首先比較參數值 minNumObj 的差異，測試集皆定為 66%

點選「Choose」右方的欄位進行參數設定。

將 minNumObj 修改成 4，得精準度為 64.3836%。

The screenshot shows two windows. The left window, 'weka.gui.GenericObjectEditor', has 'minNumObj' set to 4. The right window, 'Weka Explorer', shows the 'Classifier' tab with 'J48 - C 0.25 - M 4' selected. The 'Test options' section shows 'Percentage split' at 66%. The 'Classifier output' section displays the following summary:

```
Time taken to build model: 0 seconds
Time taken to test model on test split: 0 seconds
=== Summary ===
Correctly Classified Instances      47      64.3836 %
Incorrectly Classified Instances    26      35.6164 %
Kappa statistic                    0.5159
Mean absolute error                 0.1153
Root mean squared error             0.3009
Relative absolute error             54.3301 %
Root relative squared error         59.2218 %
Total Number of Instances          73
```

The 'Detailed Accuracy By Class' table is also visible:

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
bui	0.700	0.170	0.609	0.700	0.651	0.509	0.803	0.593	bui
bui	0.563	0.122	0.783	0.563	0.655	0.471	0.721	0.665	bui
veh	0.250	0.014	0.500	0.250	0.333	0.328	0.813	0.264	veh
veh	?	0.000	?	?	?	?	?	?	veh
cod	0.750	0.029	0.600	0.750	0.667	0.650	0.951	0.495	cod
rat	0.500	0.042	0.250	0.500	0.333	0.328	0.729	0.139	rat
hee	0.909	0.097	0.625	0.909	0.741	0.702	0.902	0.582	hee
Weighted Avg.	0.644	0.118	0.671	0.644	0.641	0.514	0.789	0.587	

將 minNumObj 修改成 2，得精準度為 57.5342%。

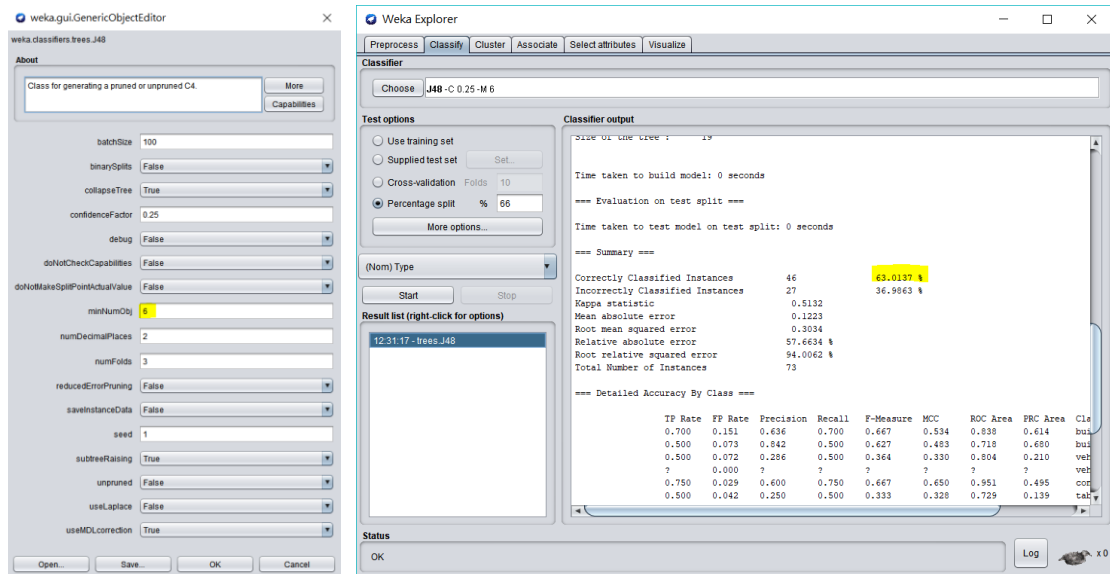
The screenshot shows two windows. The left window, 'weka.gui.GenericObjectEditor', has 'minNumObj' set to 2. The right window, 'Weka Explorer', shows the 'Classifier' tab with 'J48 - C 0.25 - M 2' selected. The 'Test options' section shows 'Percentage split' at 66%. The 'Classifier output' section displays the following summary:

```
Number of Leaves : 30
Size of the tree : 59
Time taken to build model: 0 seconds
Time taken to test model on test split: 0 seconds
=== Summary ===
Correctly Classified Instances      42      57.5342 %
Incorrectly Classified Instances    31      42.4658 %
Kappa statistic                    0.4259
Mean absolute error                 0.1246
Root mean squared error             0.3287
Relative absolute error             58.7442 %
Root relative squared error         101.8336 %
Total Number of Instances          73
```

The 'Detailed Accuracy By Class' table is also visible:

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
bui	0.600	0.208	0.522	0.600	0.558	0.377	0.727	0.492	bui
bui	0.500	0.146	0.727	0.500	0.593	0.382	0.669	0.564	bui
veh	0.250	0.029	0.333	0.250	0.286	0.253	0.817	0.393	veh

將 minNumObj 修改成 6，得精準度為 63.0137%。



由上兩圖及多次測試後可得 minNumObj 在等於 4 時，會有最大精準值，原因在於若 minNumObj 太小，會使模型複雜度越高則容易 overfitting；但若 minNumObj 太大，因為資料數不足，反而會使精準度降低。

接著修改 Percentage split 參數，且固定 minNumObj = 4

當 Percentage split = 50%，準確度為 64.486%

Percentage split	%	50	Correctly Classified Instances	69	64.486 %
			Incorrectly Classified Instances	38	35.514 %

當 Percentage split = 55%，準確度為 55.2083%

Percentage split	%	55	Correctly Classified Instances	53	55.2083 %
			Incorrectly Classified Instances	43	44.7917 %

當 Percentage split = 60%，準確度為 65.1163%

Percentage split	%	60	Correctly Classified Instances	56	65.1163 %
			Incorrectly Classified Instances	30	34.8837 %

當 Percentage split = 65%，準確度為 61.3333%

Percentage split	%	65	Correctly Classified Instances	46	61.3333 %
			Incorrectly Classified Instances	29	38.6667 %

當 Percentage split = 70% , 準確度為 60.9375%

Percentage split	%	70	Correctly Classified Instances	39	60.9375 %
			Incorrectly Classified Instances	25	39.0625 %

當 Percentage split = 75% , 準確度為 62.2642%

Percentage split	%	75	Correctly Classified Instances	33	62.2642 %
			Incorrectly Classified Instances	20	37.7358 %

當 Percentage split = 80% , 準確度為 62.7907%

Percentage split	%	80	Correctly Classified Instances	27	62.7907 %
			Incorrectly Classified Instances	16	37.2093 %

觀察趨勢後發現接近 Percentage split = 60% 與 Percentage split = 50% 時可能會有最大準確度。因此，代入 Percentage split = 58%, 59%, 61%, 62%, 48%, 49%, 51%, 52% 進行測試。

得當 Percentage split = 59% 時，會有最大準確度 65.9091%

Percentage split	%	59	Correctly Classified Instances	58	65.9091 %
			Incorrectly Classified Instances	30	34.0909 %

準確度會忽大忽小的原因在於，當 Percentage split 很大時，會有更大的可能挑到有差異的部分；當 Percentage split 很小時，則會有相對較小的可能挑到準確的部分。

訓練集的準確度通常是訓練數據上應用模型時獲得的準確度，而訓練集的準確度通常較測試集的準確度高是因為 overfitting，此時的分類模型是不可靠的。