Data Mining－Project2

學生：吳汶峻

學號：Q36071229

系級：電通甲

程式與環境：**Python3(Jupyter notebook)/Windows10**

一、 使用的資料集

這次作業我所選擇的資料集是”Wine”，這資料集是對位於義大利的同一地區生長、但來自三種不同品種的葡萄酒進行化學分析的結果。該資料集分析確定了在這三種葡萄酒中，每種葡萄酒中都含有 13 種成分的濃度。其種類及意義為：

Alcohol：酒精
Malic acid：蘋果酸
Ash：粉煤灰
Alcalinity of ash：Alcalinity 灰
Magnesium：鎂
Total phenols：總酚
Flavanoids：黃酮
Nonflavanoid phenols：Nonflavanoid 酚類
Proanthocyanins：前花青素
Color intensity：顏色強度
Hue：色相
OD280/OD315 of diluted wines：OD280/OD315 被稀釋的酒
Proline：脯氨酸

增加了 Dataset Rules

而實驗目的是期望能依據葡萄酒的這 13 種成分來預測葡萄酒的類型。

接著，我去比較 Class label 中每個葡萄酒品種所有的 13 種成份，他們各自的平均值為：

Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids
1	13.74474576	2.01067797	2.45559	17.03728814	106.338983	2.840169492	2.982372881
2	12.27873239	1.93267606	2.24479	20.23802817	94.5492958	2.258873239	2.08084507
3	13.15375	3.33375	2.43708	21.41666667	99.3125	1.67875	0.781458333

增加了
Dataset Rules

Class label	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
1	0.29	1.899322034	5.528305085	1.062034	3.15779661	1115.7119
2	0.363661972	1.63028169	3.086619718	1.056282	2.785352113	519.50704
3	0.4475	1.153541667	7.396249979	0.682708	1.683541667	629.89583

從上面這兩張圖表中可以發現，從 Malic acid 來看，品種一與品種二的含量較品種三的低；從 Magnesium 來看，品種一含量普遍高於 100、品種二則是低於 95、品種三則是普遍在 100 左右；從 Total phenols 來看，品種一與品種二的含量平均高過 2、品種三則是低於 2，之後可能可以拿 threshold = 2 來區分出品種三；從 Flavanoids 來看，品種一大概平均在 3 左右、品種二在平均 2、品種三則很明顯地低於 1；從 Nonflavanoid phenols 來看，品種一大概在 0.3 左右、品種二在 0.35、品種三則在 0.45 左右；最後的 Proline，品種一幾乎都高於 1100、品種二在 500 左右、品種三則在 600 左右。

因此，從我初步的推估，若要將葡萄酒做分類，大略可以從 Total phenols 和 Flavanoids 分辨出哪瓶葡萄酒是品種三，用 Proline 分辨出三種不同品種的葡萄酒；而若是用 Decision Tree 去做區分，也可以朝這個方向去實作，但我認為每個 feature 之間應該也是會存在某些關係，這些都是我們要在 Decision Tree 裡去做探討與了解的。

二、 Decision Tree

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04

首先，我先抓資料集中前面 5 筆資料來檢查是否有誤，顯示結果如下：

接著，將 Wine 資料集隨機拆分為訓練(train)和測試(test)集，比例為 7：3，程式碼及結果如下：

```
# split X into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)
```

```
Number of Training data: 124
Number of Testing data: 54
```

現在，可以利用 Sklearn (scikit-learn)套件來構建 Decision Tree，並檢測訓練集與測試集的誤差值，程式碼及結果如下：

In [2]:
```
from sklearn.tree import DecisionTreeClassifier

# criterion : impurity function
# max_depth : maximum depth of tree
# random_state : seed of random number generator
tree = DecisionTreeClassifier(criterion='entropy',
                              max_depth=3,
                              random_state=0)
tree.fit(X_train, y_train)
```

In [3]:
```
y_pred = tree.predict(X_test)
print('Misclassified samples: %d' % (y_test != y_pred).sum())
print('Accuracy (tree): %.2f' % ((y_test == y_pred).sum() / y_test.shape[0]))

# a more convenient way to evaluate a trained model is to use the sklearn.metrics
from sklearn.metrics import accuracy_score
print('Accuracy (tree, sklearn): %.2f' % accuracy_score(y_test, y_pred))
```

```
Misclassified samples: 2
Accuracy (tree): 0.96
Accuracy (tree, sklearn): 0.96
```

由上圖結果可知，在訓練集與測試集比例為 7：3 的情況下，此棵 Decision Tree 的準確度為 96%，相當地高。

＊補充：在經過測試後，發現到訓練集比例越低，準確度會越低。下圖為比例 1：9 的情況下：

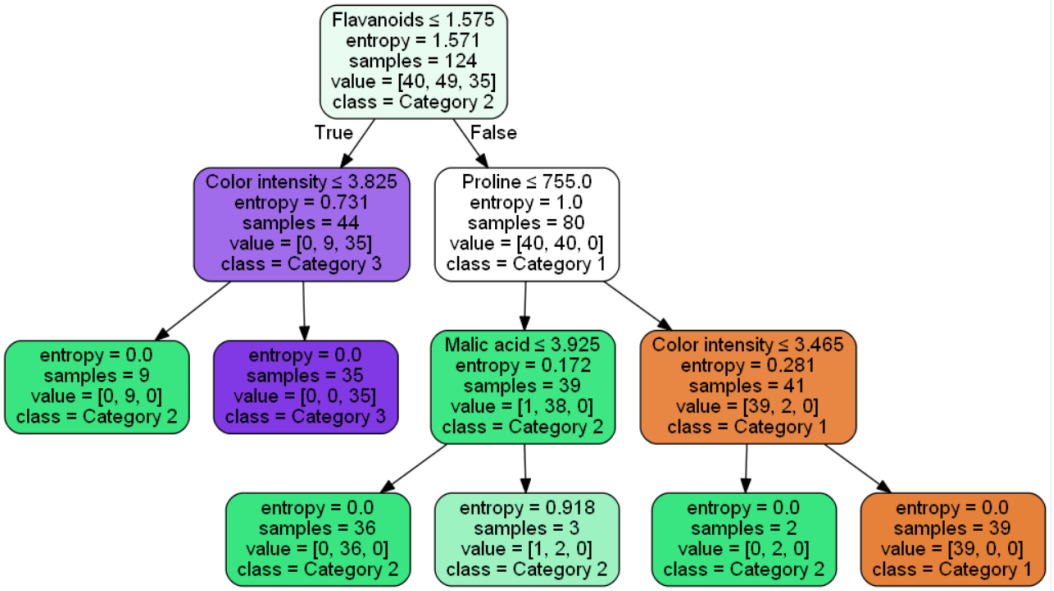
```
# split X into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.9, random_state=0)
```

```
Misclassified samples: 34
Accuracy (tree): 0.79
Accuracy (tree, sklearn): 0.79
```

為了要讓 Decision Tree 更容易觀察，我們可以利用 graphviz 套件，讓 Tree 更容易視覺化。以下為程式碼：

In [4]:
```
import graphviz
from sklearn.tree import export_graphviz

f=export_graphviz(tree, out_file=None,
                  feature_names=X.columns.values,
                  class_names = ['Category 1', 'Category 2','Category 3'],
                  filled=True,
                  rounded=True,
                  special_characters=True)
graph = graphviz.Source(f)
graph
```

依照上圖程式碼，可畫出一棵 Decision Tree。如下圖：



從此圖可以知道分類的依據及標準，而屬於哪一種類的葡萄酒是以框內背景顏色來做區分！

三、 Random Forest

＊概念：

Random Forest 被廣泛運用在資料探勘中，它的效能及準確度都相當地高，而且可以觀察每一個屬性的重要度，簡單的說，Random Forest 可以把它當作是多個 Decision Tree 組合而成的。

我們可以使用 Sklearn 的 RandomForestClassifier 套件去創建一個 Random Forest，程式碼及結果顯示如下：

In [5]:
```
from sklearn.ensemble import RandomForestClassifier

# criterion : impurity function
# n_estimators :  number of decision trees
# random_state : seed used by the random number generator
# n_jobs : number of cores for parallelism
forest = RandomForestClassifier(criterion='entropy',
                                n_estimators=200,
                                random_state=1,
                                n_jobs=2)
forest.fit(X_train, y_train)

y_pred = forest.predict(X_test)
print('Accuracy (forest): %.2f' % accuracy_score(y_test, y_pred))
```

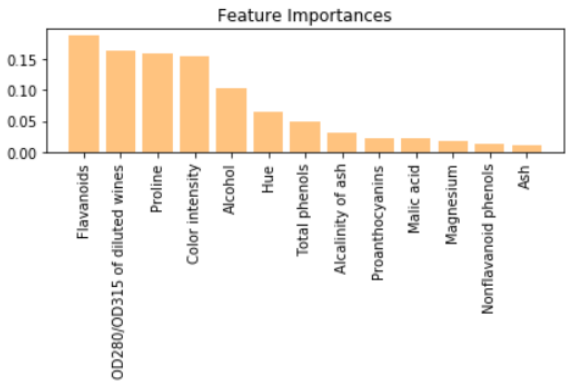
```
Accuracy (forest): 0.98
```

由此圖可知 Random Forest 的準確度為 98%，比 Decision Tree 準確要更高、預測得更準！

此外，可以利用 Random Forest 來計算屬性重要度，這樣可以將比較不重要的屬性刪減，可以讓預測與準確度更高。程式碼及結果如下顯示：

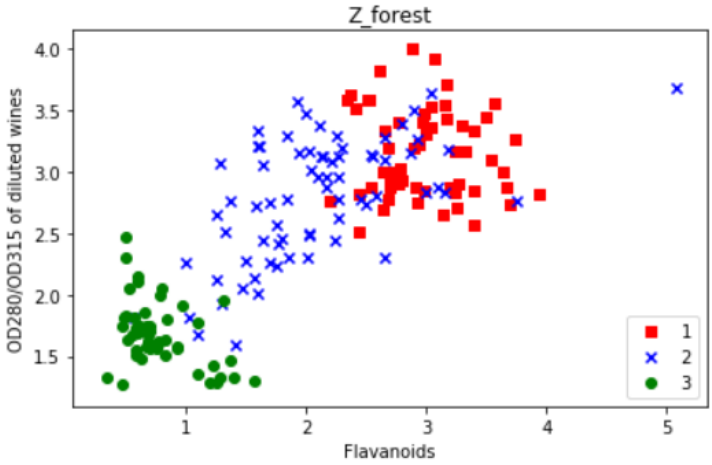
```
for f in range(X_train.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30,
                            X.columns.values[indices[f]],
                            importances[indices[f]]))

plt.figure()
plt.title('Feature Importances')
plt.bar(range(X_train.shape[1]),
        importances[indices],
        align='center',
        alpha=0.5,
        color='#FF8800')

plt.xticks(range(X_train.shape[1]),
           X.columns.values[indices], rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.tight_layout()
plt.savefig('./output/fig-forest-feature-importances.png', dpi=300)
plt.show()
```

```
 1) Flavanoids                  0.188736
 2) OD280/OD315 of diluted wines 0.162445
 3) Proline                     0.158390
 4) Color intensity             0.154620
 5) Alcohol                     0.102004
 6) Hue                         0.065470
 7) Total phenols               0.049602
 8) Alcalinity of ash           0.030379
 9) Proanthocyanins             0.023283
10) Malic acid                  0.022439
11) Magnesium                   0.018800
12) Nonflavanoid phenols        0.012507
13) Ash                         0.011325
```



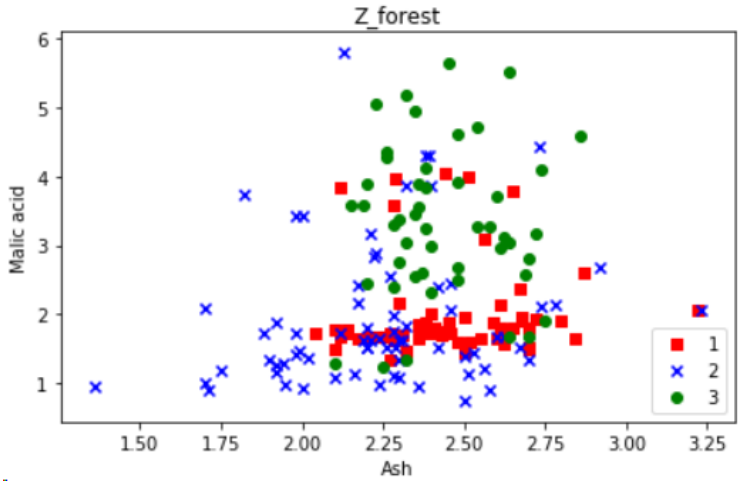
在刪減一些不重要的屬性後，得到了 Flavanoids 與 OD280/OD315 of diluted wines 是這個資料集重要度最高的，將此兩項屬性萃取出來並作圖，結果可以蠻容易區分葡萄酒種類：

```
import matplotlib.pyplot as plt

Z_forest = X[['Flavanoids', 'OD280/OD315 of diluted wines']].values

colors = ['r', 'b', 'g']
markers = ['s', 'x', 'o']
for l, c, m in zip(np.unique(y.values), colors, markers):
    plt.scatter(Z_forest[y.values==l, 0],
                Z_forest[y.values==l, 1],
                c=c, label=l, marker=m)

plt.title('Z_forest')
plt.xlabel('Flavanoids')
plt.ylabel('OD280/OD315 of diluted wines')
plt.legend(loc='lower right')
plt.tight_layout()
plt.savefig('./output/fig-forest-z.png', dpi=300)
plt.show()
```



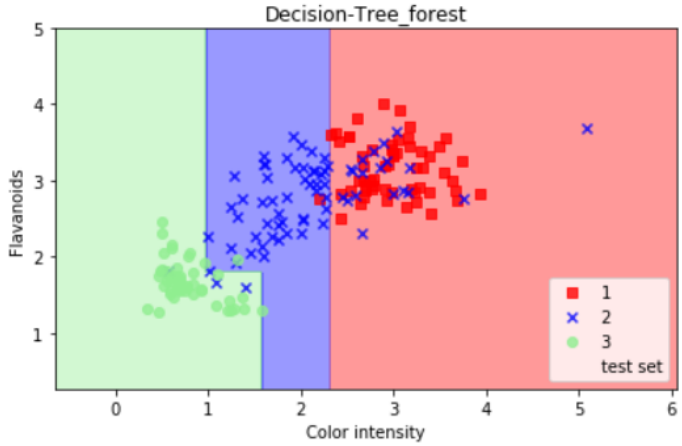
我們可以做測試，若將某兩個對資料集較不重要的兩個屬性，如：Ash 與 Malic acid 拿來作圖，會產生葡萄酒種類無法區分的結果：

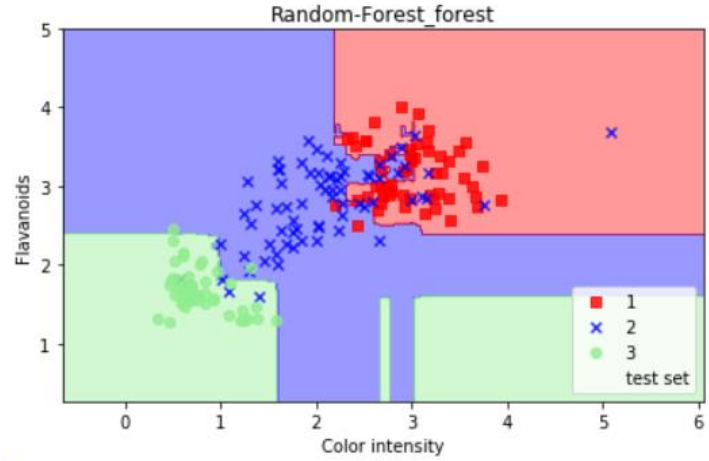
```
import matplotlib.pyplot as plt

Z_forest = X[['Ash', 'Malic acid']].values

colors = ['r', 'b', 'g']
markers = ['s', 'x', 'o']
for l, c, m in zip(np.unique(y.values), colors, markers):
    plt.scatter(Z_forest[y.values==l, 0],
                Z_forest[y.values==l, 1],
                c=c, label=l, marker=m)

plt.title('Z_forest')
plt.xlabel('Ash')
plt.ylabel('Malic acid')
plt.legend(loc='lower right')
plt.tight_layout()
plt.savefig('./output/fig-forest-z.png', dpi=300)
plt.show()
```



總結：重要度低的屬性根本區分不了葡萄酒的種類，因此取重要度高的屬性也是相當重要的一環。也可知道，並不需要全部屬性去偵測，只需要將重要度高的屬性抓出來比對，就能大略分類出葡萄酒的種類。

另外，為了比較 Decision tree 及 Random Forest 精準度，可以畫出兩種的決策邊界，下圖為 Decision-tree 與 Random-Forest 的決策邊界：





由這兩張圖做比較後，可以發現 Random Forest 比 Decision Tree 更為精準些！

四、 實作心得

這份作業需要利用 Decision Tree 來估測資料集的分類。剛開始還不太懂 Decision Tree 背後原理以及如何將資料做分類，但在上網看很多有關 Decision Tree 程式的邏輯及演算法方向後，從怎麼樣找到好的資料集、做資料處理、到用程式畫出 Decision Tree 最後再做分析及比對預測與實際結果，可知道用 Decision Tree 做資料探勘其實是件不簡單的事，但只要肯學，其實是可以融會貫通的。而且也藉由此次機會，不只學到 Decision Tree 分類法，也學到 Random Forest 的應用，希望未來在需要探勘資料時，能夠將今日所學學以致用！