HW1 - Decision Tree

一、 程式碼截圖與講解：

1. **讀檔：**使用 pandas 套件將 csv 檔案讀入，並命為 data

```
# 1 Read file
data = pd.read_csv('E:/DB/character-deaths.csv')
```

2. **資料前處理：**

```
# 2-1 data complition : set 0
data['Book of Death'] = data['Book of Death'].fillna(0)
data['Book Intro Chapter'] = data['Book Intro Chapter'].fillna(0)
data = data.drop(columns = ['Death Year','Death Chapter'])
# check nan number of data
#data['Death Year'].isna().sum()
#data['Book of Death'].isna().sum()
#data['Death Chapter'].isna().sum()

# check out 'Death Year' == na but Book of 'Death' != nan
#data[np.logical_and(pd.isna(data['Death Year']) , pd.notna(data['Book of Death'])) == True]
```

a、 **2-1** 將資料空值部分，補 0，表示尚活著

i. 經過紅色框框裡的三行確認 Book of Death、Book Intro Chapter、Death Year 三者當中**缺值最少者作為是否死亡的主要依據**

```
# 2-2 data complition : set 1
data['Book of Death'][data['Book of Death'] > 0] = 1
```

b、 **2-2** 將非空值部分，轉為 1，代表死亡

```
# 2-3 adding columns by 'Allegiances'
data = pd.concat([data,pd.get_dummies(data['Allegiances'])],axis =1 ,join='outer')
data = data.drop(columns = ['Allegiances'])
```

c、 **2-3** 將資料依照 Allegiances 這個欄位，轉換為 dummy 狀態以便後續建決策樹

```
# 2-4 divid into training set  and testing set
train, test = train_test_split(data, test_size=0.25,random_state = 0)
x_train = train.drop( ['Book of Death' ,'Name'],axis=1 )
y_train = train['Book of Death']
x_test = test.drop( ['Book of Death' ,'Name'],axis=1 )
y_test = test['Book of Death']
```

d、 **2-4** 把資料依照比例及切分出 training 和 testing

3. **使用 Scikit-learn 建構決策樹：**設定最大深度為 10，且使用 random_state 固定樹的長相

```
# 3 DecisionTreeClassifier of scikit-learn
clf = tree.DecisionTreeClassifier(max_depth = 10 , random_state = 0)
clf = clf.fit(x_train, y_train)
```

4. **列出 Confusion matrix ,precision,Recall 跟 Accuracy**

```
# 4 Make the confusion matrix and calculate Precision, Recall, Accuracy
y_pred = clf.predict(x_test)
y_true = y_test
print('Confusion matrix：' + '\n' + str(confusion_matrix(y_true,y_pred)))
print('Precision：' + str(precision_score(y_true,y_pred)))
print('Recall：' + str(recall_score(y_true, y_pred)))
print('Accuracy：'+ str(accuracy_score(y_true, y_pred)))
```

5. **列印出決策樹**

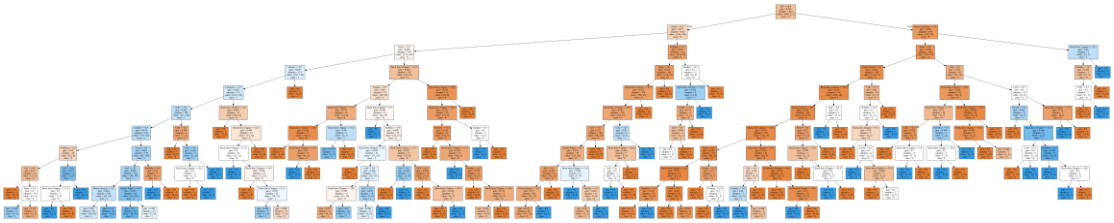
```
# 5 Generate the grap of the decisiontree
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=x_train.columns,
                                class_names=['0','1'],
                                filled=True)

graph = graphviz.Source(dot_data)
graph
```

二、 程式實作結果：

1. Confusion matrix、Precision、Recall、Accuracy

```
Confusion matrix:
[[106  38]
 [ 28  58]]
Precision: 0.6041666666666666
Recall: 0.6744186046511628
Accuracy: 0.7130434782608696
```

2. 決策樹(詳細圖片在附檔)



三、 作業心得：

在這次作業上花了不少時間，感覺還不夠熟悉 python 的各操作及套件，不過能夠有個小小實驗結果感覺很好，希望日後有更多時間的話可以嘗試提升 Accuracy。