HW1: Game of Thrones

309554001 劉雨恩

1. Data Evaluation

首先使用 info() 來檢視 dataset，發現人物資料總共有 917 筆，其中在 `Death Year` 、 `Book of Death` 、 `Death Chapter` 和 `Book Intro Chapter` 這幾個欄位中有空值的存在

```
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Name                917 non-null    object 
 1   Allegiances         917 non-null    object 
 2   Death Year          305 non-null    float64
 3   Book of Death       307 non-null    float64
 4   Death Chapter       299 non-null    float64
 5   Book Intro Chapter  905 non-null    float64
 6   Gender              917 non-null    int64  
 7   Nobility            917 non-null    int64  
 8   GoT                 917 non-null    int64  
 9   CoK                 917 non-null    int64  
 10  SoS                 917 non-null    int64  
 11  FfC                 917 non-null    int64  
 12  DwD                 917 non-null    int64  
```

```python
import pandas as pd

# read file
death_data = pd.read_csv(data_path + 'character-deaths.csv')
death_data.info()
```

2. Data Preprocessing

2.1 Fill NAN

將欄位的空值填補為 0

```python
# 1. Fill null value
death_data.fillna(value=0, inplace=True)
```

2.2 建立 `Death` 欄位

建立 `Death` 欄位，若 `Death Year` 、 `Book of Death` 或 `Death Chapter` 三者任一有值，則設成 1 (代表死亡)，其餘皆設成 0 (代表存活)

```python
# 2. Create 'Death' feature
death_data['Death'] = death_data['Death Year'] + death_data['Book of Death'] +
death_data.loc[death_data['Death'] != 0, 'Death'] = 1
```

2.3 將 `Allegiances` 轉成 dummy 特徵

底下有幾種分類就會變成幾個特徵，值是 0 或 1，本來的資料集就會再增加約 20 種特徵；並將不需要的特徵 (`Death Year` , `Book of Death` , `Death Chapter` , `Allegiances` , `Name`) 移除

```
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Book Intro Chapter  917 non-null    float64
 1   Gender              917 non-null    int64  
 2   Nobility            917 non-null    int64  
 3   GoT                 917 non-null    int64  
 4   CoK                 917 non-null    int64  
 5   SoS                 917 non-null    int64  
 6   FfC                 917 non-null    int64  
 7   DwD                 917 non-null    int64  
 8   Death               917 non-null    float64
```

```
 9   Arryn               917 non-null    uint8  
 10  Baratheon           917 non-null    uint8  
 11  Greyjoy             917 non-null    uint8  
 12  House Arryn         917 non-null    uint8  
 13  House Baratheon     917 non-null    uint8  
 14  House Greyjoy       917 non-null    uint8  
 15  House Lannister     917 non-null    uint8  
 16  House Martell       917 non-null    uint8  
 17  House Stark         917 non-null    uint8  
 18  House Targaryen     917 non-null    uint8  
 19  House Tully         917 non-null    uint8  
 20  House Tyrell        917 non-null    uint8  
 21  Lannister           917 non-null    uint8  
 22  Martell             917 non-null    uint8  
 23  Night's Watch       917 non-null    uint8  
 24  None                917 non-null    uint8  
 25  Stark               917 non-null    uint8  
 26  Targaryen           917 non-null    uint8  
 27  Tully               917 non-null    uint8  
 28  Tyrell              917 non-null    uint8  
 29  Wildling            917 non-null    uint8  
```

```python
# 3. Change 'Allegiances' to dummy features
all_dum = pd.get_dummies(pd.Series(death_data['Allegiances']))
new_death_data = pd.concat([death_data, all_dum], axis=1).drop( \
    columns=['Death Year', 'Book of Death', 'Death Chapter', \
    'Allegiances', 'Name'])
new_death_data.info()
```

2.4 亂數拆成訓練集(75%)與測試集(25%)

```python
from sklearn.model_selection import train_test_split

# 4. Randomly split to training (75%) and testing (25%) dataset
x_train, x_test, y_train, y_test = train_test_split(new_death_data.drop(column
```

3. Model training

使用 Decision Tree Classifier 做為 model

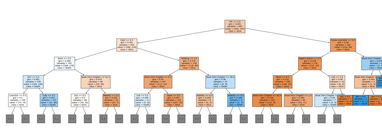
```python
from sklearn import tree

# 5. Training model & Predict
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x_train, y_train)
```

4. Results

4.1 產出決策樹的圖

用 train 好的 model 來 predict，並產生決策樹的圖

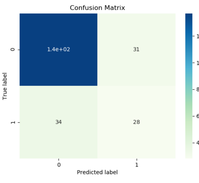


```python
y_test_pred = clf.predict(x_test)

plt.figure(figsize=(35, 10))
tree.plot_tree(clf, \
    feature_names=new_death_data.drop(columns=['Death']).columns, \
    class_names=['Alive', 'Death'], \
    filled=True, max_depth=4, fontsize=10)
plt.savefig(target_path + 'decistion_tree.png')
```

4.2 做出Confusion Matrix，並計算Precision, Recall, 和 Accuracy

- Accuracy: 0.717391304347826
- Precision: 0.4745762711864407
- Recall: 0.45161290322580644




```python
# 6. Calculate & plot the Confusion Matrix
acc = accuracy_score(y_test, y_test_pred)
prec = precision_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
print('Accuracy: {}, Precision: {}, Recall: {}'.format(acc, prec, recall))

plt.figure()
cm = confusion_matrix(y_test, y_test_pred)
df_cm = pd.DataFrame(cm, range(2), range(2))
sns.heatmap(df_cm, annot=True, cmap='GnBu')

plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix')
plt.savefig(target_path + 'conf_mat.png')
```