HW1_0813371_李嘉泰

作業說明

本次作業要求為預測角色是否死亡，資料來源是character-deaths.csv，裡面共有917筆資料。並且使用決策樹作為主要的模型。

```
1  # Import dataset
2
3  character_deaths = pd.read_csv("./Dataset/character-deaths.csv")
4  print(f"character_deaths length: {len(character_deaths)}")
```

✓ 0.0s

```
character_deaths length: 917
```

作業流程

1. 讀取資料

透過pandas的read_csv()方法來將資料集讀取進來並計算資料筆數。

```
1  # Import dataset
2
3  character_deaths = pd.read_csv("./Dataset/character-deaths.csv")
4  print(f"character_deaths length: {len(character_deaths)}")
```

✓ 0.0s

```
character_deaths length: 917
```

2. 資料前處理

2.1. 將Book of death欄位有數值的以1代替

我挑選了Book of death這個欄位作為label的參考依據，值得注意的是我使用的方法是先創造一個新的欄位叫做Label，然後將Label欄位全部填上0，之後再從資料集中找到所有

Book of death有值的row，再把這些row對應的Label填上1。

```
## Make label
character_deaths["Label"] = 0
character_deaths["Label"][character_deaths["Book of Death"].notnull()] = 1
dataset = character_deaths.drop(["Death Year", "Book of Death", "Death Chapter"], axis = 1)
```

2.2. 把空值以0代替

一樣使用pandas的fillna方法把空值全部填補0

```
## Fillna 0
dataset = dataset.fillna(0)
```

2.3. 將Allegiances轉成dummy特徵

使用pandas的get_dummies方法將Allengiances這個欄位轉成dummy欄位，並且指定轉成數值（默認是會轉成布林值）。

```
## Transfer to dummies
dataset = pd.get_dummies(dataset, columns=["Allegiances"], dtype = int)
```

2.4. 將原始資料集拆成訓練集(75%)和測試集(25%)

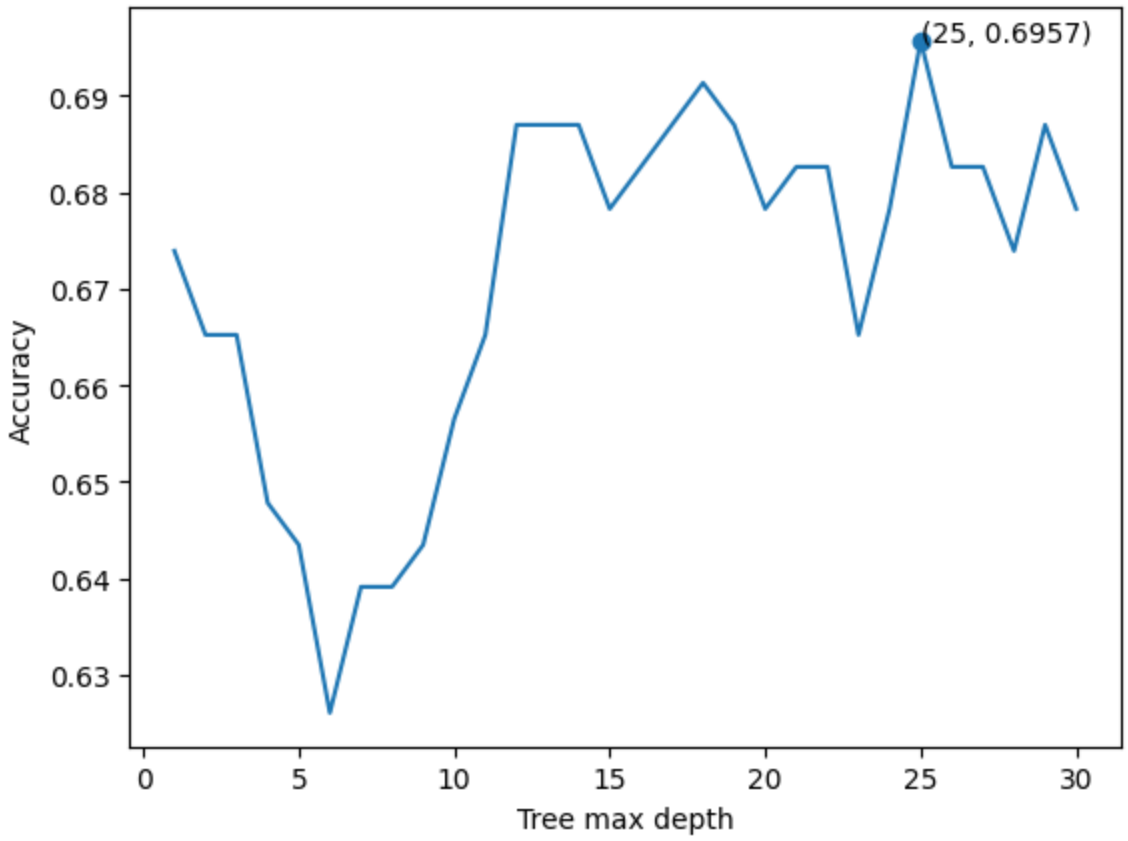
隨後先將原始資料集分割成feature與label，然後再丟入sklearn提供的train_test_split方法之中，並且將測試集的大小設為25%。

```
# Split dataset
X = dataset.drop(["Label"], axis = 1)
y = dataset["Label"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

3. 使用DecisionTreeClassifier進行預測

接下來使用DecisionTreeClassifier進行model的訓練與預測，我在這裡對max_depth這個參數進行30次的實驗，最後發現當樹的深度在25的時候訓練會有最好的表現。因此我將樹的深度參數設在25。

```python
# Build model
classifier = tree.DecisionTreeClassifier(max_depth=25)
classifier = classifier.fit(X_train, y_train)
# Prediction
prediction = classifier.predict(X_test)
```



4. 做出Confusion matrix和Precision、Recall、Accuracy

最後將預測的結果與真值對答案，透過sklearn內建的confusion_matrix與classification_report這兩個方法將結果印出來。可以從表格中發現，Accuracy有70%代表模型預測對的機率有70%，然後Precision有79％代表模型預測是0的時候有79％的機率會是對的，Recall為78%則是當正確答案是0的時候模型有78%會是預測成功。

```python
# Confusion metrix
print(f"confusion_matrix: \n{confusion_matrix(y_test, prediction)}")
# Classify report
print(classification_report(y_test, prediction))
```

✓ 0.0s

```
confusion_matrix:
[[123  35]
 [ 33  39]]
              precision    recall  f1-score   support

           0       0.79      0.78      0.78       158
           1       0.53      0.54      0.53        72

    accuracy                           0.70       230
   macro avg       0.66      0.66      0.66       230
weighted avg       0.71      0.70      0.71       230
```

5. 畫決策樹的圖

這是最終決策樹的結果圖，可能是因為深度太深所以導致node都看不太清楚，不知道有沒有什麼辦法可以讓他看得清楚一點

Kaggle

使用了助教給的資料集進行測試後上傳到kaggle 得到了93%的成績，並且這並沒有對資料做太多的前處理。

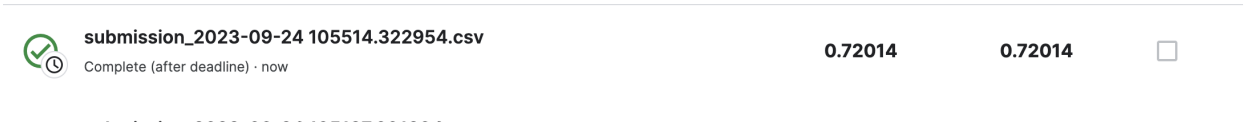
submission.csv Complete (after deadline) · now	0.93077	0.93077	☐

討論

我參考了網路上其他人的做法將Book Intro Chapter這個欄位變為二元值，把空值的部分填0，其他則是填1，因為有被介紹的角色可能會影響到他的死亡率，所以這個欄位或許可以當作參考。

```
10  ## Transfer Book intro Chapter to binrary
11  dataset.loc[dataset.loc[:, "Book Intro Chapter"] > 1, "Book Intro Chapter"] = 1
```

不過在經過實驗之後發現這樣處理過的資料訓練的模型在kaggle的分數反而降低，所以我認爲或許像之前一樣讓這個欄位的值有大有小會比較好，因為被介紹的先後順序也是一種資訊，或許越後面被介紹的角色死亡率會跟低也說不一定。有鑑於此，我想嘗試看看做標準化會不會好一點。



Normalize

我使用z score的方法將Book Intro Chapter這個欄位進行標準化，可以明顯地看到與之前的方法比起來高了8%的準確度，所以可以看出Book Intro Chapter這個欄位每個數值的大小確實會對資料訓練有幫助。但依然比不上之前沒經過標準化之前的結果(93%)，所以或許讓他維持著跟之前一樣的數值範圍會更好。

```
10  ## Normalize Book Intro Chapter
11  mean = dataset["Book Intro Chapter"].mean()
12  std = dataset["Book Intro Chapter"].std()
13  dataset["Book Intro Chapter"] = (dataset["Book Intro Chapter"] - mean) / std
```