Data Mining Research and Practice HW1

Department: IAM Student ID: 309652008 Name: 廖家緯

October 14, 2021

1 讀取資料

```
import pandas as pd
df = pd.read_csv('character-deaths.csv')
df.head()
df.info()
```

	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
0	Addam Marbrand	Lannister	NaN	NaN	NaN	56.0	1	1	1	1	1	1	0
1	Aegon Frey (Jinglebell)	None	299.0	3.0	51.0	49.0	1	1	0	0	1	0	0
2	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5.0	1	1	0	0	0	0	1
3	Adrack Humble	House Greyjoy	300.0	5.0	20.0	20.0	1	1	0	0	0	0	1
4	Aemon Costayne	Lannister	NaN	NaN	NaN	NaN	1	1	0	0	1	0	0

Figure 1. Data Frame

```
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
```

Figure 2. Data Frame Information

- 共有 13 個欄位
- Gender, Nobility, GoT, CoK, SoS, FfC, DwD 等欄位值域為 $\{0, 1\}$，表示有或無
- Allegiances 含有 21 個類別，分別為 Arryn, Baratheon, Greyjoy, House Arryn, House Baratheon, House Greyjoy, House Lannister, House Martell, House Stark, House Targaryen, House Tully, House Tyrell, Lannister, Martell, Night's Watch, None, Stark, Targaryen, Tully, Tyrell, Wildling

2 資料前處理

2.1 處理缺失資料

由 Figure 2. 可知，Death Year、Book of Death、Death Chapter、Book Intro Chapter 欄位有缺失資料，因此先把缺失資料補 0

```
df['Death Year'] = df['Death Year'].fillna(0)
df['Book of Death'] = df['Book of Death'].fillna(0)
df['Death Chapter'] = df['Death Chapter'].fillna(0)
df['Book Intro Chapter'] = df['Book Intro Chapter'].fillna(0)
```

2.2 取出 label

取出 Death Year 欄位，將大於 0 的數值轉成 1，當作 label

```
Y = (df['Death Year']>0).astype(int)
```

2.3 特徵轉換

將 Allegiances 欄位以 one-hot encoding 方式表示，並取出訓練與測試資料

```
df = df.join(pd.get_dummies(df["Allegiances"]))
X = df.drop(columns=['Name', 'Allegiances', 'Death Year', 'Book of Death', 'Death
    Chapter'])
```

2.4 資料集切分

將 data set 隨機拆成 75% 的訓練集與 25% 的測試集

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
    random_state=42)
```

3 模型訓練

3.1 Decision Tree

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_depth=4, random_state=0)
clf = clf.fit(X_train, Y_train)
```

可調整的超參數:

- max_depth: 樹的最大深度
- random_state: 隨機因子

3.2 Grid Search

將樹的深度限制在 2 到 5 層之間，隨機因子取前 100 個，形成參數空間

$$\Theta = \{(d, r) \in \mathbb{N} \times \mathbb{N} \mid 2 \leq d \leq 7, 1 \leq r \leq 100\}$$

```
max_depth_space = range(2, 6)
random_state_space = range(100)
parameter_space = [(d, r) for d in max_depth_space for r in random_state_space]
best_accuracy = 0

for d, r in parameter_space:
  clf = tree.DecisionTreeClassifier(max_depth=d, random_state=r)
  clf = clf.fit(X_train, Y_train)
  test_score = clf.score(X_test, Y_test)

  if test_score>best_accuracy:
    best_accuracy = test_score
    best_parameter = (d, r)

clf = tree.DecisionTreeClassifier(max_depth=best_parameter[0],
    random_state=best_parameter[1])
clf = clf.fit(X_train, Y_train)
```

因為模型訓練速度很快，所以使用 grid search 找最佳參數並不會花太多時間，最後試出最佳參數 $\theta^* = (3, 0)$

4 模型評估

```
from sklearn.metrics import confusion_matrix

X_train_predict = clf.predict(X_train)
X_test_predict = clf.predict(X_test)

tr_TN, tr_FP, tr_FN, tr_TP = confusion_matrix(X_train_predict, Y_train).ravel()
ts_TN, ts_FP, ts_FN, ts_TP = confusion_matrix(X_test_predict, Y_test).ravel()

tr_precision = tr_TP / (tr_TP + tr_FP)
tr_recall = tr_TP / (tr_TP + tr_FN)
tr_accuracy = (tr_TP + tr_TN) / (tr_TN + tr_FP + tr_FN + tr_TP)

ts_precision = ts_TP / (ts_TP + ts_FP)
ts_recall = ts_TP / (ts_TP + ts_FN)
ts_accuracy = (ts_TP + ts_TN) / (ts_TN + ts_FP + ts_FN + ts_TP)
```

- Confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	TP	FP
Actual Negative	FN	TN

Training set	Predicted Positive	Predicted Negative
Actual Positive	152	93
Actual Negative	115	327

Test set	Predicted Positive	Predicted Negative
Actual Positive	40	22
Actual Negative	39	129

- Evaluation

1. Precision $= \dfrac{TP}{TP+FP}$
2. Recall $= \dfrac{TP}{TP+FN}$
3. Precision $= \dfrac{TP+TN}{TP+FP+FN+TN}$

Evaluation	Precision	Recall	Accuracy
Training set	0.62041	0.56929	0.69723
Test set	0.64516	0.50633	0.73478

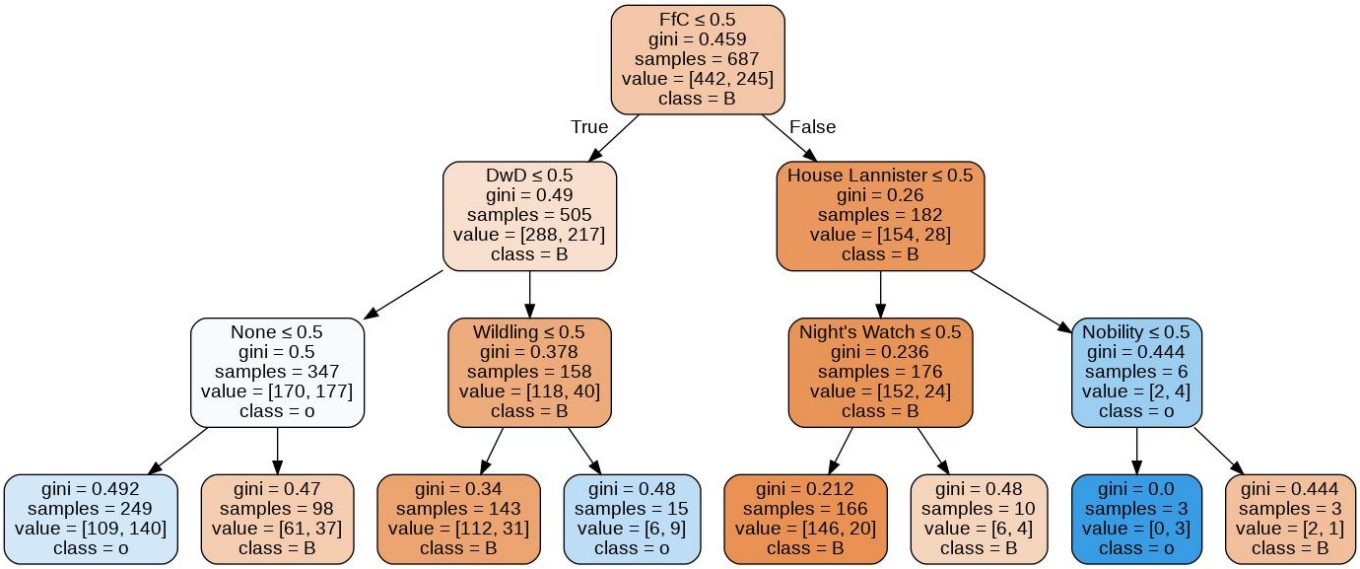
比較 training accuracy 與 test accuracy，模型沒有 overfitting

5 視覺化決策樹

```
import graphviz

dot_data = tree.export_graphviz(
    decision_tree=clf,
    out_file=None,
    feature_names=X_train.columns.values,
    class_names=Y_train.name,
    filled=True,
    rounded=True,
    special_characters=True)

graph=graphviz.Source(dot_data)
graph.render('decision_tree', view=True, format='jpg')
```



6 整理 Code

將以上步驟寫成 main.py、utilts.py 兩個檔案，並將前處理、grid search、模型評估用物件形式打包，增加可讀性。