資料探勘研究與實務
HW01 Decision Tree

Department : IMM , 洪翊誠, ID : 310653005

October 13, 2021

1 HW1 - Decision Tree

資料集在以下網址: https://www.kaggle.com/mylesoneill/game-of-thrones

制式化流程

1. 讀取資料透過 pandas 的 dataframe 建立表格
2. 透過 info, describe 等最初淺的方式先認識資料的型態數值

[1]:
```
import os
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

path = r'E:\Users\User\DataSet_for_Class\DataSet'
file = 'character-deaths.csv'

df = pd.read_csv(os.path.join(path,file))
df.head()
```

[1]:
```
                   Name       Allegiances  Death Year  Book of Death  \
0         Addam Marbrand         Lannister         NaN            NaN
1  Aegon Frey (Jinglebell)            None       299.0            3.0
2        Aegon Targaryen  House Targaryen         NaN            NaN
3          Adrack Humble    House Greyjoy       300.0            5.0
4        Aemon Costayne         Lannister         NaN            NaN

   Death Chapter  Book Intro Chapter  Gender  Nobility  GoT  CoK  SoS  FfC
0            NaN                56.0       1         1    1    1    1    1
1           51.0                49.0       1         1    0    0    1    0
2            NaN                 5.0       1         1    0    0    0    0
3           20.0                20.0       1         1    0    0    0    0
4            NaN                 NaN       1         1    0    0    1    0

DwD
0    0
1    0
2    1
3    1
4    0
```

[2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 917 entries, 0 to 916
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  
dtypes: float64(4), int64(7), object(2)
memory usage: 93.3+ KB
```

[3]:
```
df.describe()
```

[3]:
```
       Death Year  Book of Death  Death Chapter  Book Intro Chapter  \
count  305.000000     307.000000     299.000000          905.000000   
mean   299.157377       2.928339      40.070234           28.861878   
std      0.703483       1.326482      20.470270           20.165788   
min    297.000000       1.000000       0.000000            0.000000   
25%    299.000000       2.000000      25.500000           11.000000   
50%    299.000000       3.000000      39.000000           27.000000   
75%    300.000000       4.000000      57.000000           43.000000   
max    300.000000       5.000000      80.000000           80.000000   

           Gender    Nobility         GoT         CoK         SoS         FfC  \
count  917.000000  917.000000  917.000000  917.000000  917.000000  917.000000   
mean     0.828790    0.468920    0.272628    0.353326    0.424209    0.272628   
std      0.376898    0.499305    0.445554    0.478264    0.494492    0.445554   
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000   
25%      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000   
50%      1.000000    0.000000    0.000000    0.000000    0.000000    0.000000   
75%      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000   
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000   

              DwD  
count  917.000000  
mean     0.284624  
std      0.451481  
min      0.000000  
25%      0.000000  
50%      0.000000  
75%      1.000000  
max      1.000000  
```

指定 Book of Death 做為目標項，有數值的轉成 1

[4]:
```python
# Set the target
Y = df["Book of Death"]
Y[pd.notna(Y)] = 1
Y = Y.fillna(0)
pd.unique(Y), Y.shape
```

[4]:
```
(array([0., 1.]), (917,))
```

2-1 把空值以 0 替代

[5]:
```python
df_filled = df.fillna(1)
# df_filled
```

2-2 Choose：Book of Death 將有數值的轉成 1

[6]:
```python
df_filled["Book of Death"] = 1
# df_filled
```

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)

[7]:
```python
pd.unique(df["Allegiances"]),len(pd.unique(df["Allegiances"]))
```

[7]:
```
(array(['Lannister', 'None', 'House Targaryen', 'House Greyjoy',
        'Baratheon', "Night's Watch", 'Arryn', 'House Stark',
        'House Tyrell', 'Tyrell', 'Stark', 'Greyjoy', 'House Lannister',
        'Martell', 'House Martell', 'Wildling', 'Targaryen', 'House Arryn',
        'House Tully', 'Tully', 'House Baratheon'], dtype=object),
 21)
```

刪除不該出現在訓練時需要用的資料

```
[8]: X = df_filled.join(pd.get_dummies(df_filled["Allegiances"]))
     del X["Allegiances"], X["Name"], X["Death Chapter"], X["Death Year"]
```

Total 30 columns

```
[9]: X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 917 entries, 0 to 916
Data columns (total 30 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Book of Death       917 non-null    int64
 1   Book Intro Chapter  917 non-null    float64
 2   Gender              917 non-null    int64
 3   Nobility            917 non-null    int64
 4   GoT                 917 non-null    int64
 5   CoK                 917 non-null    int64
 6   SoS                 917 non-null    int64
 7   FfC                 917 non-null    int64
 8   DwD                 917 non-null    int64
 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
dtypes: float64(1), int64(8), uint8(21)
memory usage: 83.4 KB
```

2-4 亂數拆成訓練集 (75%) 與測試集 (25%) 套用 sklearn 的 train_test_split，random_state 任意取 (這裡取 6)

[10]:
```
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=6)
```

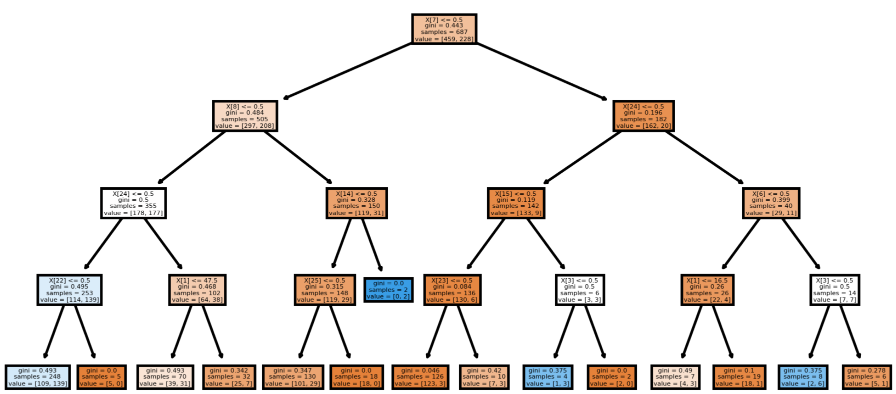
1.0.1 3) 使用 scikit-learn 的 DecisionTreeClassifier 進行預測

最大深度設定到 4 層，避免過深作圖時不方便觀測

[11]:
```
from sklearn.tree import DecisionTreeClassifier
# from sklearn.model_selection import cross_val_score
from sklearn import tree
clf = DecisionTreeClassifier(max_depth=4,random_state=0)
clf = clf.fit(X_train, y_train)
clf
```

[11]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=4, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

[12]:
```
prediction = clf.predict(X_test)
import matplotlib.pyplot as plt
from matplotlib.pyplot import savefig
fig = plt.figure(figsize=(6, 3), dpi=350)
tp =  tree.plot_tree(clf,  filled=True)
```



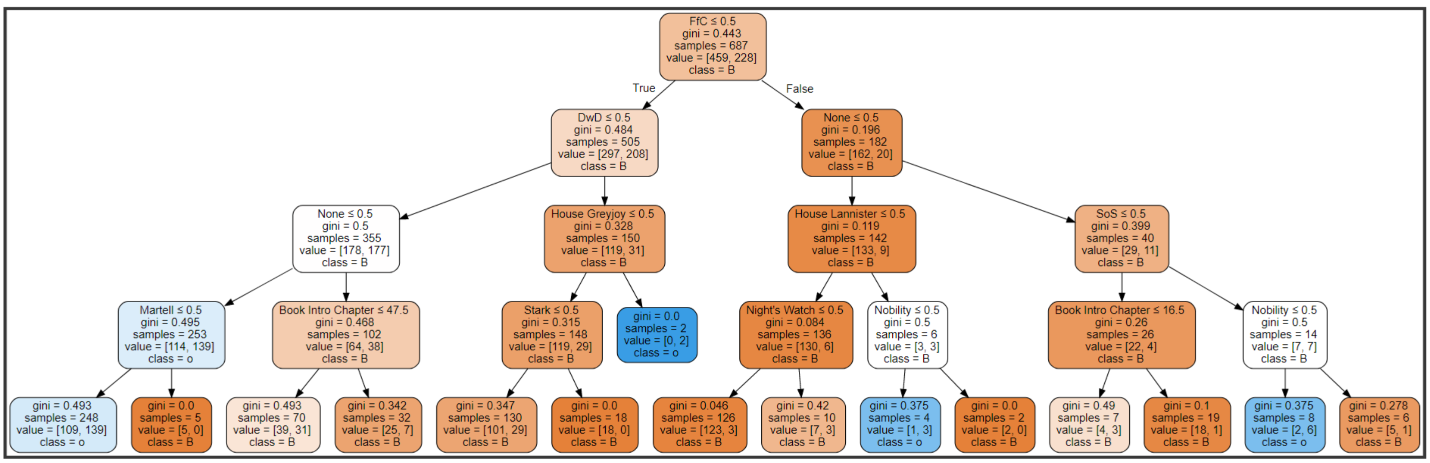
由於上述的 **decision tree** 是針對 **index** 來顯示，不便閱讀　故使用其他方式來將 feature name 放進圖表裡面，能更方便觀察如何做決策的
其中裡面會計算 gini 值、到子葉的樣本數以及分類數量結果

此處因位在本機端電腦無法順利執行程式，改由使用 **Google** 的 **Colab** 執行，將結果貼至下方表格　分類分別為 Book to Death ，也就是死亡，以及 0：代表存活

[13]:
```
# import graphviz
# dot_data = tree.export_graphviz(clf, out_file=None,
#                      feature_names=list(X_train.columns.values),
#                      class_names=list(y_train.name),
#                      filled = True,
#                      rounded = True,
#                      special_characters = True)

# graph = graphviz.Source(dot_data)
# graph.render('mygraph',view = True)
```

[14]:
```
# graph = graphviz.Source(dot_data)
# graph
```



2 Training value

觀察在這個深度下的樹訓練資料精確率以及相關的其他數值呈現

[15]:
```
from sklearn.metrics import confusion_matrix
print('Confusion Marix : ')
print(confusion_matrix(clf.predict(X_train), y_train))
tn, fp, fn, tp = confusion_matrix(clf.predict(X_train), y_train).ravel()
accuray = (tp+tn)/(tp+tn+fp+fn)
precision = tp/(tp+fp)
recall = tp/(tp+fn)
sensitivity = tp/(tp+fn) # TPR
specficity = tn/(tn+fp) # FPR
f1_score = 1/(1/sensitivity+1/specficity)
print(f'''Accuary     = {accuray},
Precision   = {precision},
Recall      = {recall},
Sensitivity = {sensitivity},
Specficity  = {specficity},
F1_Score    = {f1_score}.''')
```

```
Confusion Marix :
[[347  78]
 [112 150]]
Accuary     = 0.7234352256186317,
Precision   = 0.6578947368421053,
Recall      = 0.5725190839694656,
Sensitivity = 0.5725190839694656,
Specficity  = 0.8164705882352942,
F1_Score    = 0.3365359747581855.
```

[16]:
```
# 在位來可能會使用到，先練習如何使用
from sklearn.metrics import log_loss
log_loss(prediction, y_test)
```

[16]: 10.662102211917416

3 Testing Value

觀測在前面訓練測資完成後的 Decision Tree，面對未知的問題是否能有效的分類
一樣是數值結果呈現

[17]:
```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(prediction, y_test).ravel()
print('Confusion Marix : ')
print(confusion_matrix(prediction, y_test))
accuray = (tp+tn)/(tp+tn+fp+fn)
precision = tp/(tp+fp)
recall = tp/(tp+fn)
sensitivity = tp/(tp+fn) # TPR
specficity = tn/(tn+fp) # FPR
f1_score = 1/(1/sensitivity+1/specficity)

print(f'''Accuary     = {accuray},
Precision   = {precision},
Recall      = {recall},
Sensitivity = {sensitivity},
Specficity  = {specficity},
F1_Score    = {f1_score}.''')
```

```
Confusion Marix :
[[118  38]
 [ 33  41]]
Accuary     = 0.691304347826087,
Precision   = 0.5189873417721519,
Recall      = 0.5540540540540541,
Sensitivity = 0.5540540540540541,
Specficity  = 0.7564102564102564,
F1_Score    = 0.3198043363299842.
```

3.0.1 對 Testing set 利用熱擴散圖畫出混淆矩陣數量分布

上方橫列代表的是 Ground Truth，側邊縱列代表的是 Prediction
0：表示存活，1：表示死亡
(最原始圖效果不佳，加上對應的表格數量值，顏色軸，說明所佔的數量分別為何)

[18]:
```
import seaborn as sns
# confusion_matrix(prediction, y_test),
# 上橫 : GT , 側縱 : Pred
print('上方橫列代表的是 Ground Truth，側邊縱列代表的是 Prediction.')
print('0 : 表示存活， 1 : 表示死亡')
fig =sns.heatmap(confusion_matrix(prediction, y_test),
                 cmap = 'Spectral', # Spectral, YlOrRd
                 annot = True,
                 annot_kws={"size":30},
                 fmt = 'd',
                 linewidths = 1.5,
                 linecolor = 'Black')
```

```
上方橫列代表的是 Ground Truth，側邊縱列代表的是 Prediction.
0 : 表示存活， 1 : 表示死亡
```