Data Analysis

1. 先做EDA了解資料大致在做什麼

In [1]:
```
import seaborn as sns
sns.set_style("darkgrid")
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
```

HW requirement 1

1.將資料讀取進來(可用pandas套件)

In [2]:
```
battles = pd.read_csv('battles.csv')
character_death = pd.read_csv('character-deaths.csv')
```

In [3]:
```
battles.shape
```

Out[3]:
```
(38, 25)
```

In [4]:
```
battles.head()
```

Out[4]:

	name	year	battle_number	attacker_king	defender_king	attacker_1	attacker_2	attacker_3	attac
0	Battle of the Golden Tooth	298	1	Joffrey/Tommen Baratheon	Robb Stark	Lannister	NaN	NaN	
1	Battle at the Mummer's Ford	298	2	Joffrey/Tommen Baratheon	Robb Stark	Lannister	NaN	NaN	
2	Battle of Riverrun	298	3	Joffrey/Tommen Baratheon	Robb Stark	Lannister	NaN	NaN	
3	Battle of the Green Fork	298	4	Robb Stark	Joffrey/Tommen Baratheon	Stark	NaN	NaN	
4	Battle of the Whispering Wood	298	5	Robb Stark	Joffrey/Tommen Baratheon	Stark	Tully	NaN	

5 rows × 25 columns

In [5]:
```
character_death.shape
```

Out[5]:
```
(917, 13)
```

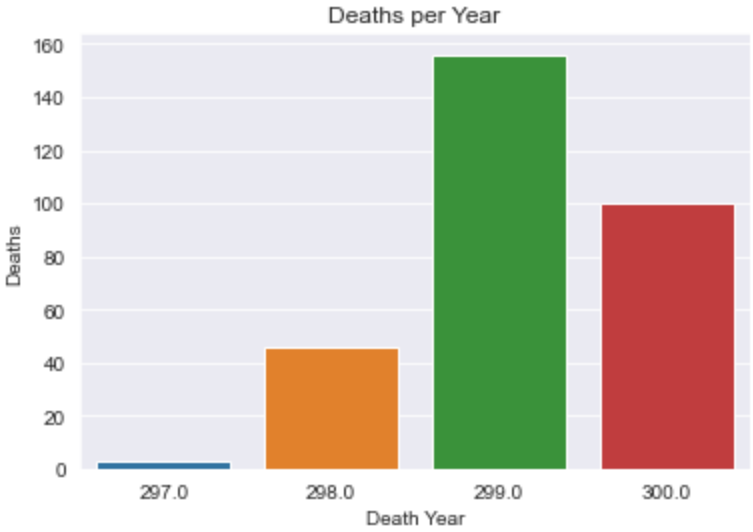
In [6]:
```
character_death.head()
```

Out[6]:

	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

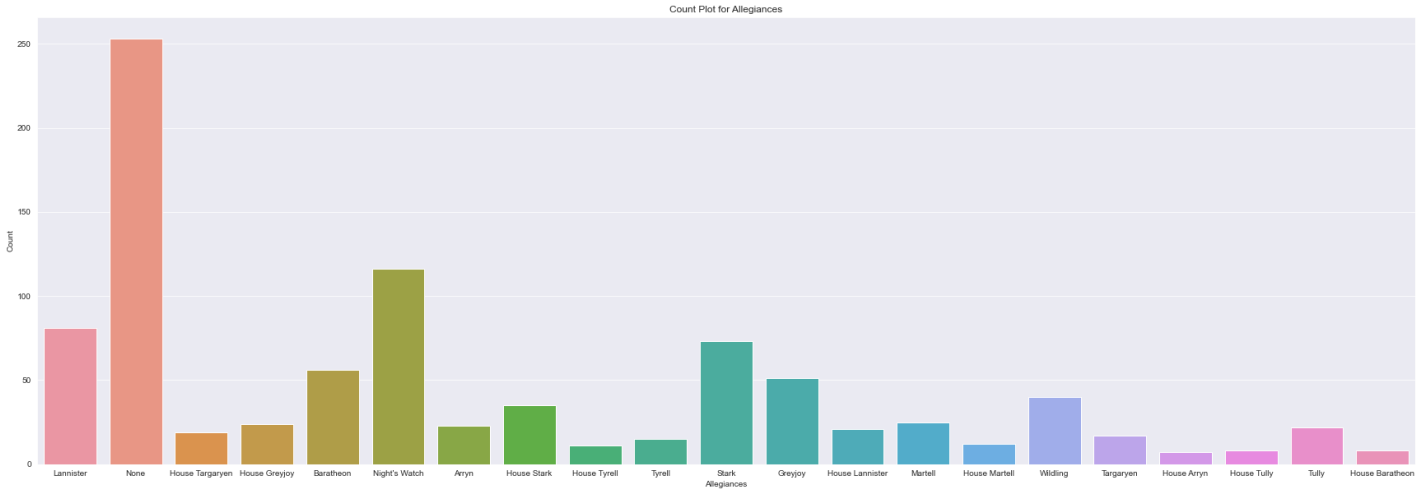
In [7]:

```
sns.countplot(x=character_death["Death Year"])
plt.title("Deaths per Year")
plt.xlabel("Death Year")
plt.ylabel("Deaths")
plt.show()
```



In [8]:

```
plt.rcParams["figure.figsize"] = (30, 10)
sns.countplot(x=character_death["Allegiances"])
plt.title("Count Plot for Allegiances")
plt.ylabel("Count")
plt.show()
```



Data Preprocessing

HW requirement 2

2-1. 空值以0替代

In [9]:
```
character_death.isna().sum()
```

Out[9]:
```
Name                    0
Allegiances             0
Death Year            612
Book of Death         610
Death Chapter         618
Book Intro Chapter     12
Gender                  0
Nobility                0
GoT                     0
CoK                     0
SoS                     0
FfC                     0
DwD                     0
dtype: int64
```

In [10]:
```
df = character_death.fillna(0)
df.isna().sum()
```

Out[10]:
```
Name                  0
Allegiances           0
Death Year            0
Book of Death         0
Death Chapter         0
Book Intro Chapter    0
Gender                0
Nobility              0
GoT                   0
CoK                   0
SoS                   0
FfC                   0
DwD                   0
dtype: int64
```

2-2. Death Year , Book of Death , Death Chapter三者取一個，將有數值的轉成1

In [11]:
```
#只留下Book of Death

cols = ['Name','Death Year','Death Chapter']
```

```
character_death = character_death.drop(cols,axis = 1)
character_death.head()
```

Out[11]:

	Allegiances	Book of Death	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
0	Lannister	NaN	56.0	1	1	1	1	1	1	0
1	None	3.0	49.0	1	1	0	0	1	0	0
2	House Targaryen	NaN	5.0	1	1	0	0	0	0	1
3	House Greyjoy	5.0	20.0	1	1	0	0	0	0	1
4	Lannister	NaN	NaN	1	1	0	0	1	0	0

In [12]:

```
#Book of Death column的數值轉換 (number ->1 / NaN ->0)
#使用numpy.where (condition[, x, y])
BD = np.where(character_death['Book of Death'].isnull(),0,1)
character_death['Book of Death'] = BD
character_death['Book Intro Chapter'] = character_death['Book Intro Chapter'].fillna(0)

character_death['Book of Death']
```

Out[12]:

```
0       0
1       1
2       0
3       1
4       0
       ..
912     0
913     1
914     1
915     1
916     1
Name: Book of Death, Length: 917, dtype: int64
```

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)

In [13]:

```
#get_dummies 是利用pandas實現one hot encode的方式
character_death = pd.get_dummies(character_death,columns = ['Allegiances'])
```

2-4亂數拆成訓練集(75%)與測試集(25%)

-> why亂數？防止兩者資料分布差異太大，容易overfitting

In [14]:

```
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
```

In [15]:

```
#target
dy = character_death['Book of Death']
#target "Book of Death"不能出現在訓練集裡面，需拿掉
character_death = character_death.drop('Book of Death',axis = 1)
dx = character_death #data without ground truth
train_x,test_x,train_y,test_y = train_test_split(dx, dy, random_state=100, train_size=0.75
```

Model Building

1. 使用scikit-learn的DecisionTreeClassifier進行預測(可以先試著將網頁範例(iris)跑出來在使用這次作業的資料集)

4)做出Confusion Matrix，並計算Precision, Recall, Accuracy (提示: 可使用sklearn.metrics)

- Recall(召回率) = TP/(TP+FN)
- Precision(準確率) = TP/(TP+FP)
- F1-score = 2 *Precision* Recall / (Precision + Recall)

In [21]:

```
model = tree.DecisionTreeClassifier(criterion = 'entropy', max_depth=10, min_samples_split
model.fit(train_x,train_y)
pred_y = model.predict(test_x)
#Calculating Accuracy
acc = model.score(test_x,test_y)
#confusion matri
confusion_matrix(test_y,pred_y)
print(classification_report(test_y, pred_y))
```

```
              precision    recall  f1-score   support

           0       0.71      0.75      0.73       140
           1       0.58      0.53      0.55        90

    accuracy                           0.67       230
   macro avg       0.65      0.64      0.64       230
weighted avg       0.66      0.67      0.66       230
```

In [39]:

```
#seq = list(range(len(character_death.index) + 1))
#output = pd.DataFrame()
#output['Character'] = seq
#output['Death'] = pred_y
#pred_y
#['Survived'] = pred_y
#submit.to_csv('submit.csv', index = False)
#output
#character_death
```

Out[39]:

	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD	Allegiances_Arryn	Allegiances_Baratheon	...	A
0	56.0	1	1	1	1	1	1	0	0	0	...	
1	49.0	1	1	0	0	1	0	0	0	0	...	
2	5.0	1	1	0	0	0	0	1	0	0	...	
3	20.0	1	1	0	0	0	0	1	0	0	...	
4	0.0	1	1	0	0	1	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
912	21.0	1	0	0	0	1	0	0	0	0	...	
913	47.0	1	0	0	0	0	0	1	0	0	...	
914	25.0	1	1	0	0	0	0	1	0	0	...	
915	73.0	1	0	0	0	1	0	0	0	0	...	
916	29.0	1	1	0	0	0	1	0	0	0	...	

917 rows × 29 columns

1. 產出決策樹的圖

In [17]:
```
pip install graphviz
```

```
Requirement already satisfied: graphviz in /Users/USER/opt/anaconda3/envs/tf/lib/python3.
7/site-packages (0.20.1)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```
import graphviz
dot_data = tree.export_graphviz(model, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("HW1_310706043_肇綺筠")
graph.view()
```

Out[20]:
```
'HW1_310706043_肇綺筠.pdf'
```

Hyperparameters Tuning

In [23]:
```
dt = tree.DecisionTreeClassifier(random_state=42)
from sklearn.model_selection import GridSearchCV

#parameters for Decision tree
params = {
    'max_depth': [2, 3, 5, 10, 20],
    'min_samples_leaf': [5, 10, 20, 50, 100],
    'criterion': ["gini", "entropy"]
}
```

In [25]:
```
grid_search = GridSearchCV(estimator=dt,
                           param_grid=params,
                           cv=4, n_jobs=-1, verbose=1, scoring = "accuracy")

grid_search.fit(train_x, train_y)
```

Out[25]:
```
Fitting 4 folds for each of 50 candidates, totalling 200 fits
GridSearchCV(cv=4, estimator=DecisionTreeClassifier(random_state=42), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100]},
             scoring='accuracy', verbose=1)
```

In [29]:
```
dt_best = grid_search.best_estimator_
```

In [30]:
```
print(classification_report(test_y, dt_best.predict(test_x)))
```

```
              precision    recall  f1-score   support

           0       0.70      0.74      0.72       140
           1       0.56      0.51      0.53        90

    accuracy                           0.65       230
   macro avg       0.63      0.63      0.63       230
weighted avg       0.65      0.65      0.65       230
```