DNN Lab

Prof. Chia-Yu Lin
Yuan Ze University
2021 Spring

Environment Setup

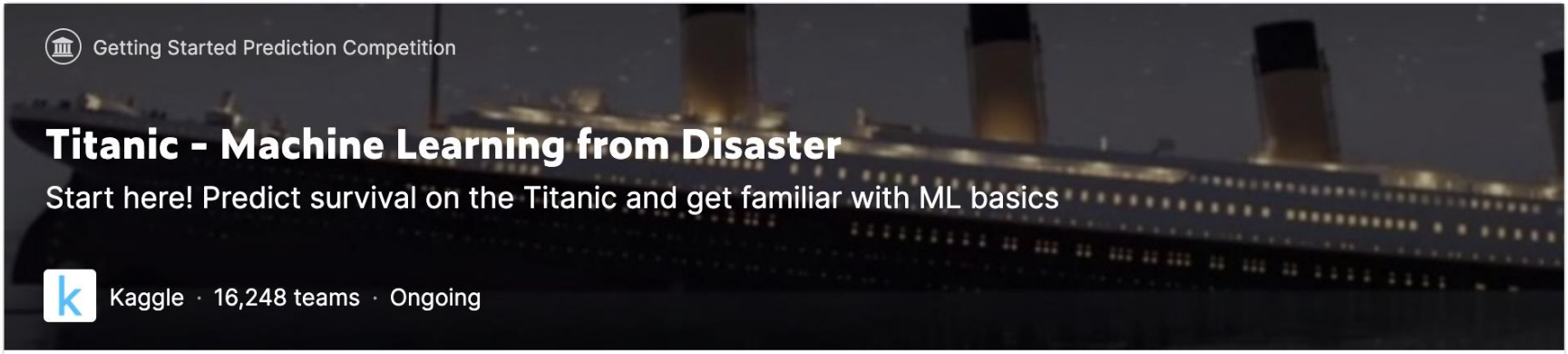
- 開啟 Jupyter Notebook

- 進入您存放檔案(資料集、code)的工作目錄
- 開啟DNN.ipynb檔進入Python編譯環境


鐵達尼號生存預測

資料來源

- Kaggle
- https://www.kaggle.com/c/titanic/notebooks



安裝需要的library

- Seaborn
  - 讓Matplotlib的圖表更美觀的函式庫，只需匯入便能讓圖表變得更美觀，還能指定數個增加的樣式
- 安裝seaborn
  - pip install seaborn

引入資料處理需要的library

```
#先導入資料處理會用到的模組
import numpy as np
import numpy.random as random
import scipy as sp
from pandas import Series, DataFrame
import pandas as pd

# 可視化模組
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
%matplotlib inline

# 機器學習模組
import sklearn
```

資料前處理

- 收到資料第一步要做什麼？
- 了解資料
- 可視化資料
- 資料清洗
- 填補空缺值
- 把文字Mapping成0,1,2….

資料集說明

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

觀察資料

- 讀檔
- 秀出前50筆資料

```
dataset = pd.read_csv('input/titanic.csv')
dataset.head(50) #秀出前50筆資料
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S

觀察資料

- 觀察資料幾列幾行

```
#觀察資料幾列幾行
dataset.shape
```

```
(891, 12)
```

- 觀察整個資料集的資訊
- 12個欄位
- 欄位的type總共三種
  - float64(有2個)
  - int64(有5個)
  - object(有5個)

```
#觀察整個資料集的資訊
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId     891 non-null int64
Survived        891 non-null int64
Pclass          891 non-null int64
Name            891 non-null object
Sex             891 non-null object
Age             714 non-null float64
SibSp           891 non-null int64
Parch           891 non-null int64
Ticket          891 non-null object
Fare            891 non-null float64
Cabin           204 non-null object
Embarked        889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

可視化資料

- 設定seaborn為預設繪圖library

```
#設定seaborn為預設繪圖library
sns.set()
```

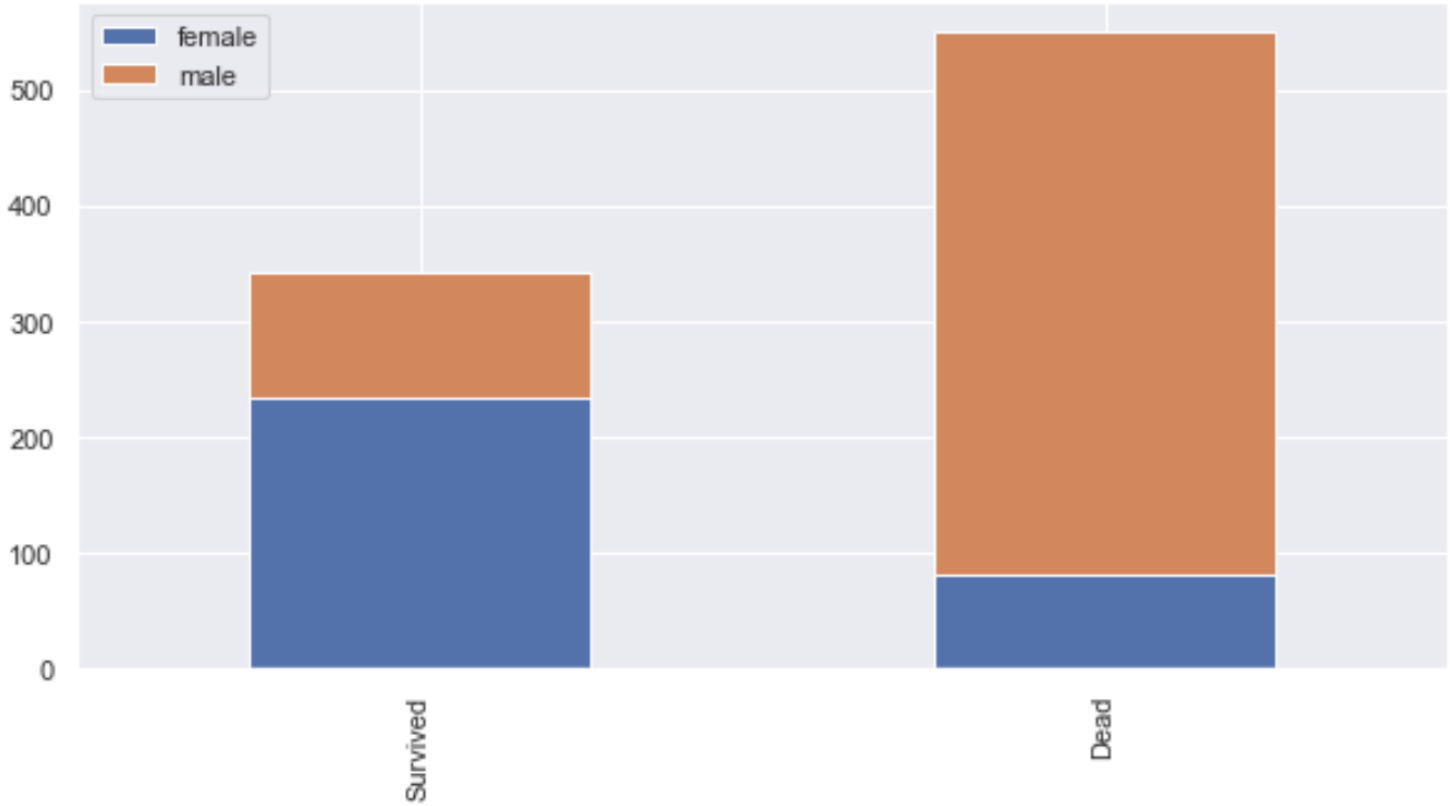
設定繪圖的函式

- 因為要預測生還率
- 所以以「survived、dead」為X軸
- 繪製其他feature相對於「survived、dead」的關係

```
def bar_chart(feature):
    survived = dataset[dataset['Survived']==1][feature].value_counts()
    dead = dataset[dataset['Survived']==0][feature].value_counts()
    df = pd.DataFrame([survived,dead])
    df.index = ['Survived','Dead']
    df.plot(kind='bar',stacked=True, figsize=(10,5))
```

繪製生還/死亡的男女長條圖

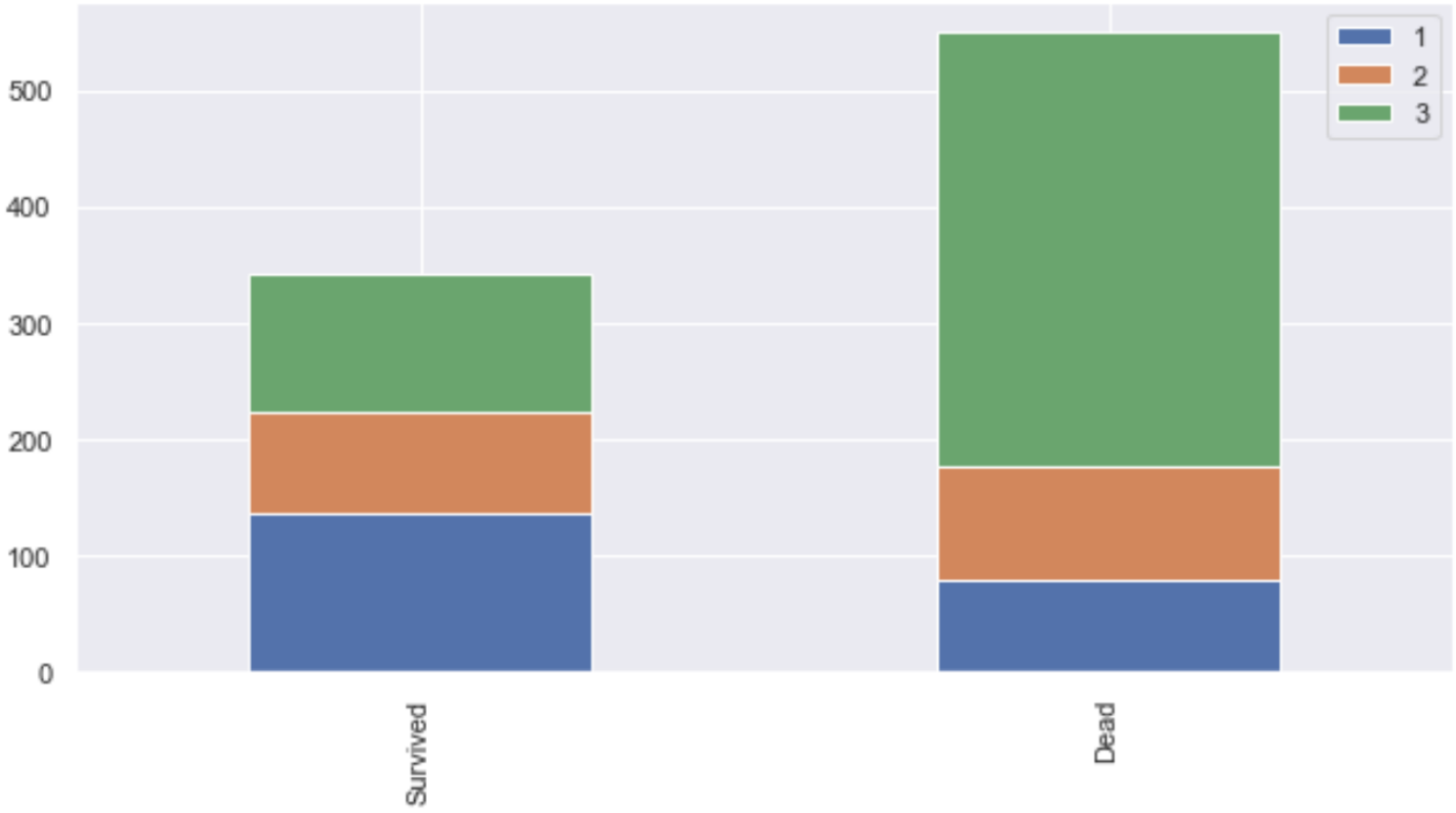
```
#分別秀出生還/死亡的男女長條圖
bar_chart('Sex')
```



The Chart confirms Women more likely survivied than Men

繪製生還/死亡的艙等長條圖

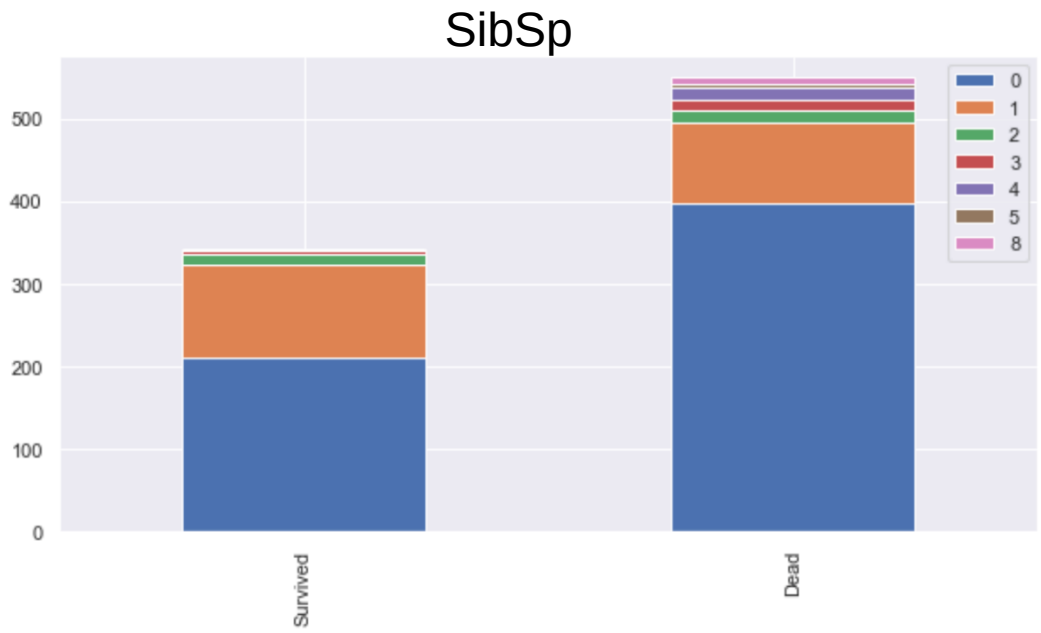
```
#分別秀出生還/死亡的艙等長條圖
bar_chart('Pclass')
```



The Chart confirms 1st class more likely survivied than other classes The Chart confirms 3rd class more likely dead than other classes

bar_chart()函式

- 在這個function裡輸入任何feature
- 就可以觀察你想看的feature
- 同學可以自行輸入



The Chart confirms a person aboarded with more than 2 siblings or spouse more likely survived The Chart confirms a person aboarded without siblings or spouse more likely dead

資料補值

- 資料中有NaN

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S

觀察資料中有幾個NaN (1/3)

- 如何做？
- 昨天在預測汽車價錢時有觀察「？」有幾個

```
# 計算各個行(欄位)裡有多少個"？"
dataset.isin(['?']).sum()
```

```
# 計算各個行(欄位)裡有多少個"NaN"
dataset.isin(['NaN']).sum()
```

Yes or No??

觀察資料中有幾個NaN (2/3)

- 觀察資料中有幾個NaN

```
# 計算各個行(欄位)裡有多少個"NaN"
dataset.isin(['NaN']).sum()
```

```
PassengerId     0
Survived        0
Pclass          0
Name            0
Sex             0
Age             0
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin           0
Embarked        0
dtype: int64
```

- 為什麼？？
- 因為NaN是特殊字元，所以不能用字串判斷

觀察資料中有幾個NaN (3/3)

- 觀察資料中有幾個NaN

```
# 計算各個行(欄位)裡有多少個"NaN"
dataset.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

資料補值

- 如何補值？
- 可以參考哪個欄位
- 觀察各個欄位

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S

觀察Name欄位

- 觀察Name欄位

```
dataset['Name']
```

```
0                                  Braund, Mr. Owen Harris
1        Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                  Heikkinen, Miss. Laina
3            Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                                  Allen, Mr. William Henry
                                ...
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

- 名字當中隱含許多額外的資料Mr.、Mrs. 、Dr 、Miss....
- 可以作為之後補值的類別參考

定義一個新類別

- 將符合「A-Za-z+.」的稱謂的詞挑出
  - ex. Mr. Mrs.

```
dataset['Title'] = dataset['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
#https://reurl.cc/qeZQE
#https://reurl.cc/Neb8n
```

- 正則表達式用法
- https://reurl.cc/qeZQE
- https://reurl.cc/Neb8n

統計各稱謂的人數

```
dataset['Title'].value_counts()
```

```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Major         2
Col           2
Mlle          2
Ms            1
Sir           1
Jonkheer      1
Don           1
Lady          1
Mme           1
Countess      1
Capt          1
Name: Title, dtype: int64
```

0

1

2

3

想看看:Mapping Function

- 依照這三大類去定義一個mapping function
- 統一以0、1、2、3去做編號

?:自行填入的地方

```
title_mapping = {"Mr": 0, ?: ?, ?: ?, ....}
dataset['Title'] = dataset['Title'].map(title_mapping)
```

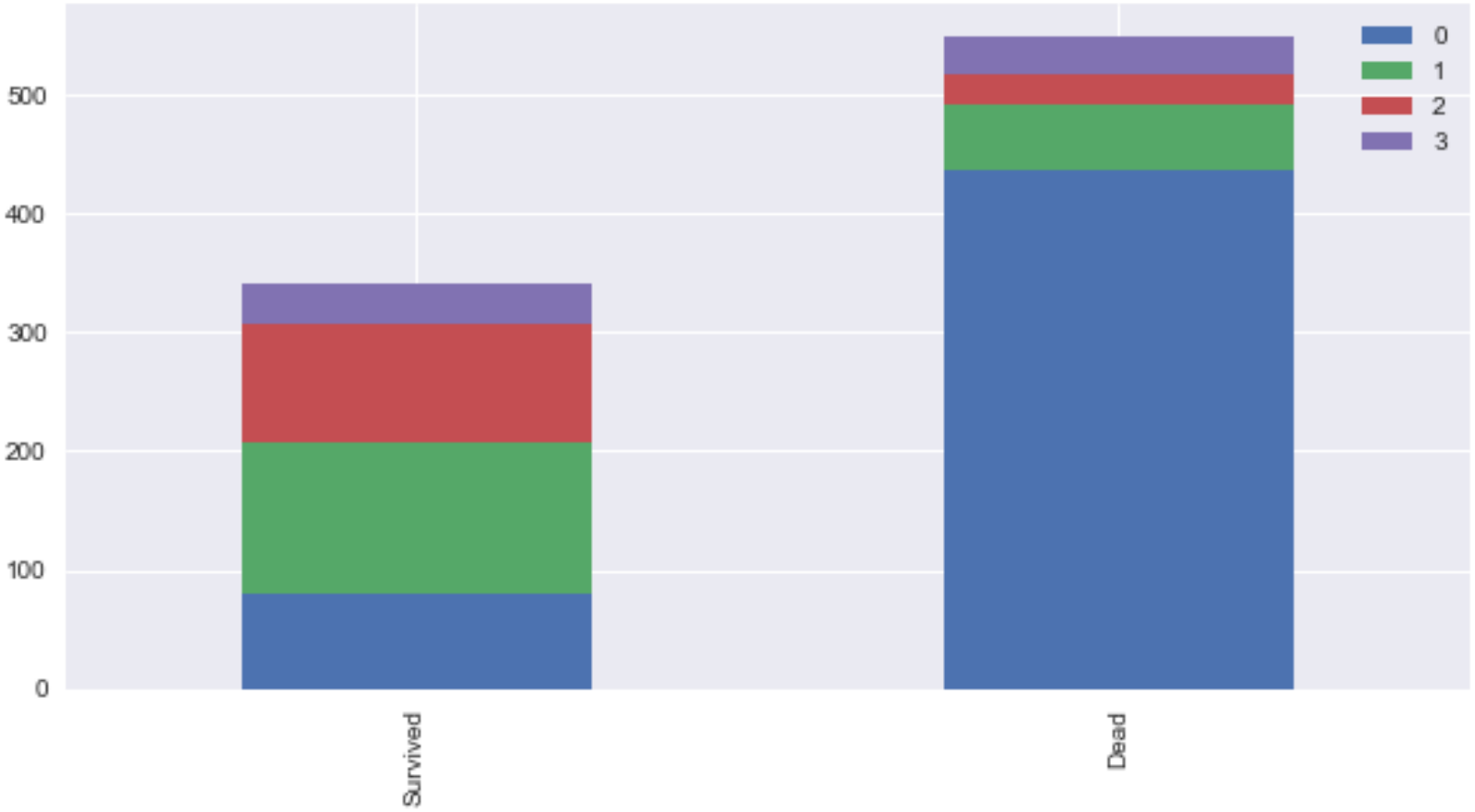
```
dataset.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0

檢查結果有沒有和這裡一樣

秀出Mapping後的結果

- 如何寫？？



資料清洗 (1/2)

- 刪除不需要的欄位
- Name的稱謂已經轉換成一個新的class了
- Name與生存/死亡無關
- 不需要保留，可刪除Name欄位
- 如何寫？

```
# delete unnecessary feature from dataset
??.??('?', axis=?, inplace=True)
```

資料清洗 (2/2)

- 刪除Name欄位之後

```
dataset.head()
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C	2
2	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	1	female	35.0	1	0	113803	53.1000	C123	S	2
4	5	0	3	male	35.0	0	0	373450	8.0500	NaN	S	0

- Sex是文字，需要進行處理

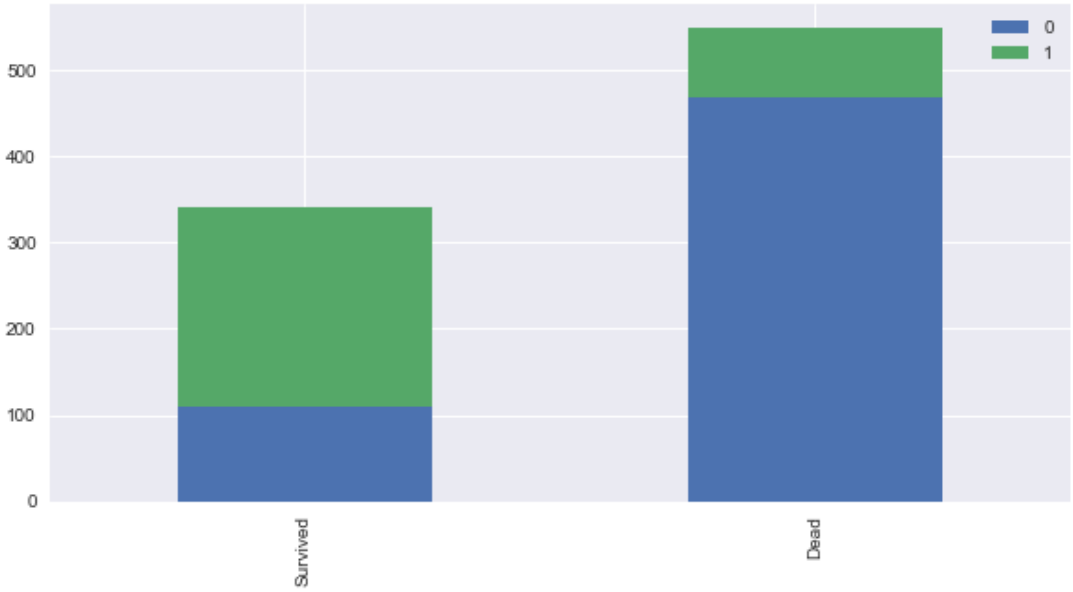
想看看：性別對於生還死亡的影響

- 去Sex欄位裡面找有什麼類別名稱
- 男性訂為0 女性訂為1

?:自行填入的地方

```
sex_mapping = {?: ?, ?: ?}
dataset['Sex'] = dataset['Sex'].map(sex_mapping)

bar_chart('Sex')
```



目前的DataFrame

```
dataset.head(100)
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	C85	C	2
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	1	1	35.0	1	0	113803	53.1000	C123	S	2
4	5	0	3	0	35.0	0	0	373450	8.0500	NaN	S	0
...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	0	3	0	NaN	0	0	374910	8.0500	NaN	S	0
96	97	0	1	0	71.0	0	0	PC 17754	34.6542	A5	C	0
97	98	1	1	0	23.0	0	1	PC 17759	63.3583	D10 D12	C	0
98	99	1	2	1	34.0	0	1	231919	23.0000	NaN	S	2
99	100	0	2	0	34.0	1	0	244367	26.0000	NaN	S	0

100 rows × 12 columns

資料集欄位

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Title

填補空缺值：年齡欄位

- 如何填補？？
- 補平均值？
- 補前一列的年齡？
- 年齡或許與Mr.、Mrs.、Miss 有關
- 利用Title變成一組，去填補年齡的中位數

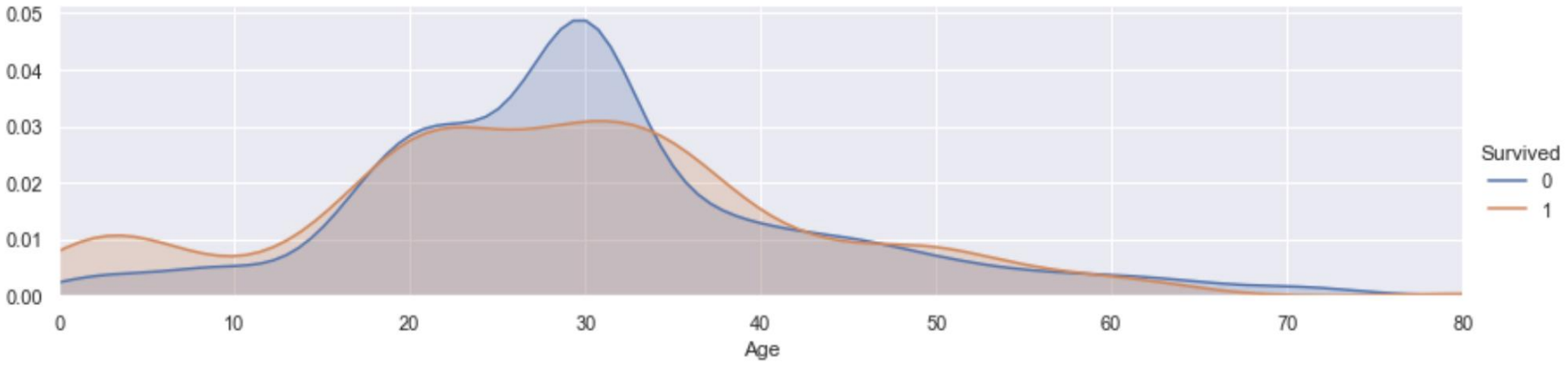
```
# fill missing age with median age for each title (Mr, Mrs, Miss, Others)
dataset["Age"].fillna(dataset.groupby("Title")["Age"].transform("median"), inplace=True)
dataset["Age"]
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    21.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

年齡/生還死亡分布圖

```
facet = sns.FacetGrid(dataset, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, dataset['Age'].max()))
facet.add_legend()

plt.show()
```



如果出現下列warning直接忽略即可，不影響後續實驗

FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result. return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

依年齡區間做mapping function

- 年齡範圍太大
- 這樣類別太多
- 使用年齡區間去mapping成0,1,2,3,4

```
dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0,
dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 26), 'Age'] = 1,
dataset.loc[(dataset['Age'] > 26) & (dataset['Age'] <= 36), 'Age'] = 2,
dataset.loc[(dataset['Age'] > 36) & (dataset['Age'] <= 62), 'Age'] = 3,
dataset.loc[ dataset['Age'] > 62, 'Age'] = 4
```

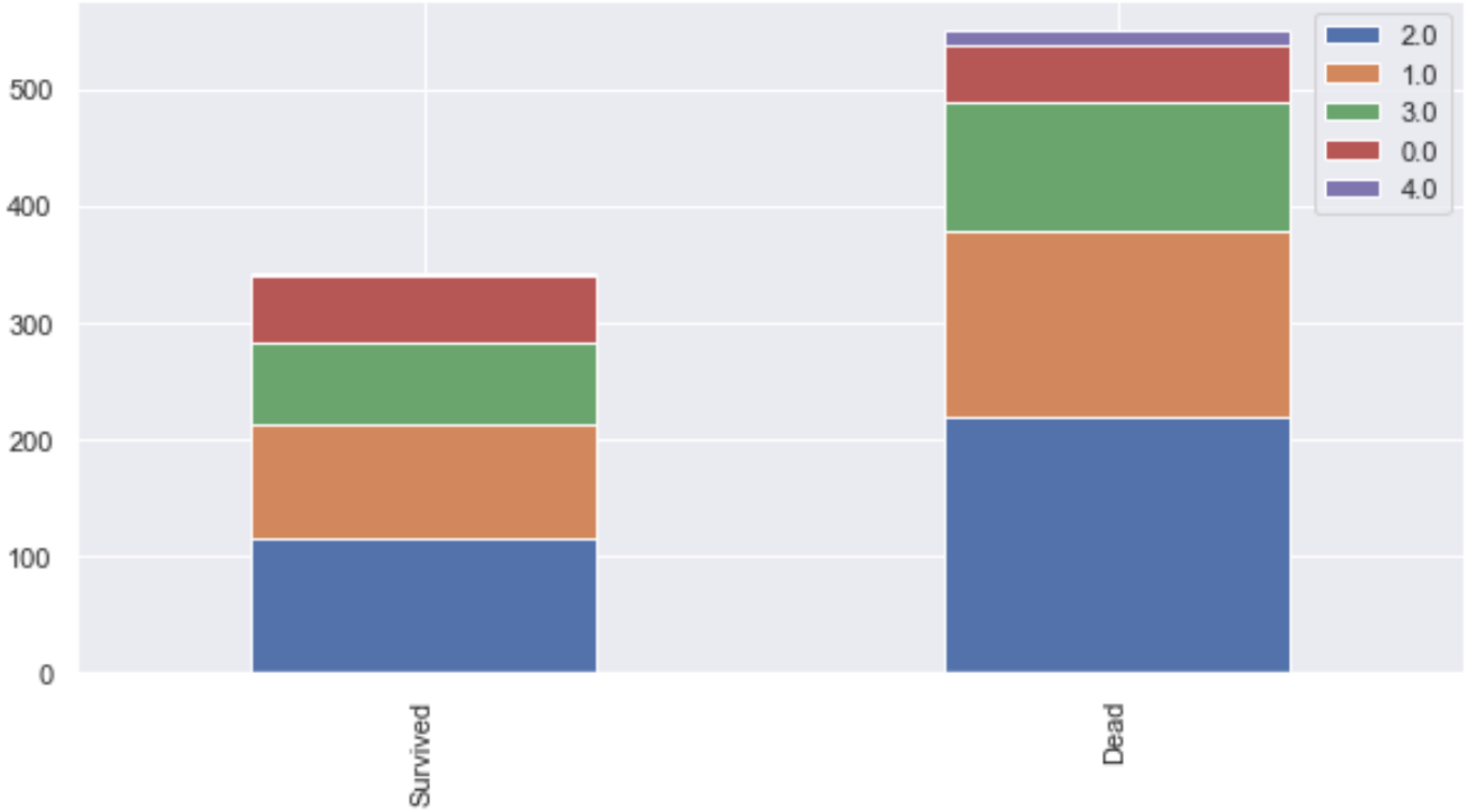
目前的DataFrame

```
dataset.head()
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	C	2
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	S	2
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	S	0

想看看：年齡的長條圖

- 如何做？？

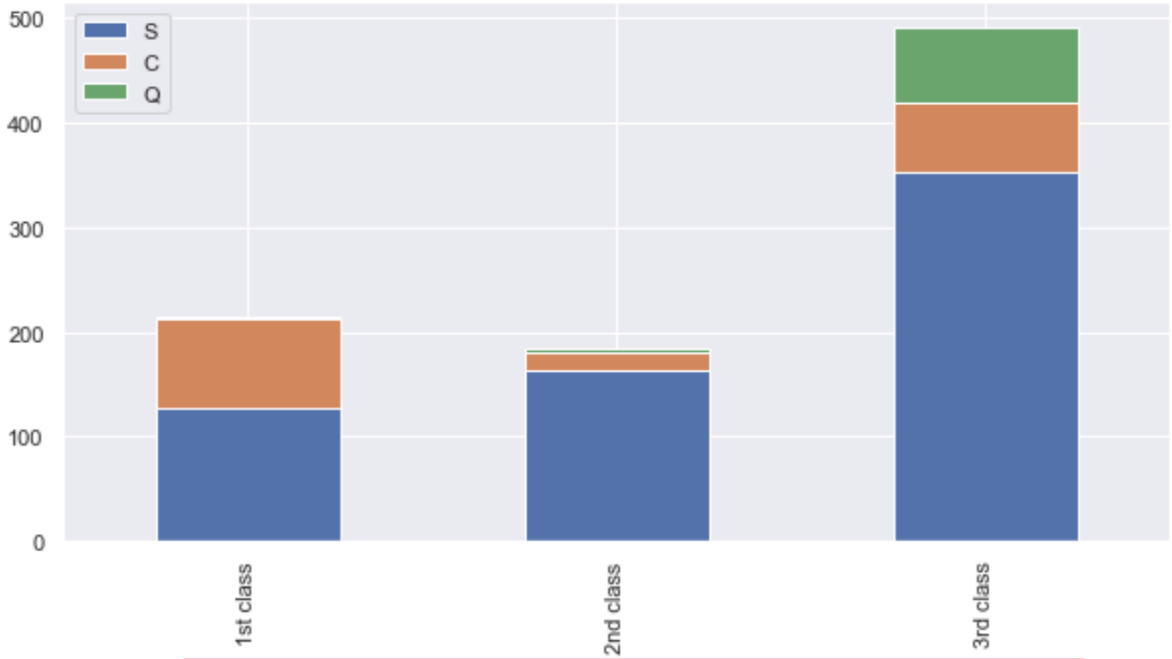


資料集欄位

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Title

分析各票種登船地點人數

```
Pclass1 = dataset[dataset['Pclass']==1]['Embarked'].value_counts()
Pclass2 = dataset[dataset['Pclass']==2]['Embarked'].value_counts()
Pclass3 = dataset[dataset['Pclass']==3]['Embarked'].value_counts()
df = pd.DataFrame([Pclass1, Pclass2, Pclass3])
df.index = ['1st class','2nd class', '3rd class']
df.plot(kind='bar',stacked=True, figsize=(10,5))
```



more than 50% of 1st class are from S embark.
more than 50% of 2nd class are from S embark.
more than 50% of 3rd class are from S embark.

補登船地點缺失資料

- 如何補值？
- 因為每個艙等都超過50%是從“s”登船的
- 所以登船地點可以補上“s”

```
dataset['Embarked'] = dataset['Embarked'].fillna('S')
dataset.head(100)
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	C	2
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	S	2
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	S	0
...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	0	3	0	2.0	0	0	374910	8.0500	NaN	S	0
96	97	0	1	0	4.0	0	0	PC 17754	34.6542	A5	C	0
97	98	1	1	0	1.0	0	1	PC 17759	63.3583	D10 D12	C	0
98	99	1	2	1	2.0	0	1	231919	23.0000	NaN	S	2
99	100	0	2	0	2.0	1	0	244367	26.0000	NaN	S	0

100 rows × 12 columns

想看看：登船地點的Mapping Function

- S訂為0 、C訂為1、Q訂為2

?:自行填入的地方

```
embarked_mapping = {?: ?, ?: ?, ?: ?}
dataset['Embarked'] = dataset['Embarked'].map(embarked_mapping)
dataset.head(100)
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	0	0
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	1	2
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	0	1
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	0	2
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	0	3	0	2.0	0	0	374910	8.0500	NaN	0	0
96	97	0	1	0	4.0	0	0	PC 17754	34.6542	A5	1	0
97	98	1	1	0	1.0	0	1	PC 17759	63.3583	D10 D12	1	0
98	99	1	2	1	2.0	0	1	231919	23.0000	NaN	0	2
99	100	0	2	0	2.0	1	0	244367	26.0000	NaN	0	0

100 rows × 12 columns

資料集欄位

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Title

想看看：填補票價缺失的資料 (1/2)

- 訓練資料的票價沒有缺失值
- 但是測試資料的票價可能有缺失
- 所以可以先在訓練資料選擇填補方式

- 可參考填補年齡的方式
- 如何做？？
- 票價與什麼有關？

想看看：填補票價缺失的資料 (2/2)

?:自行填入的地方

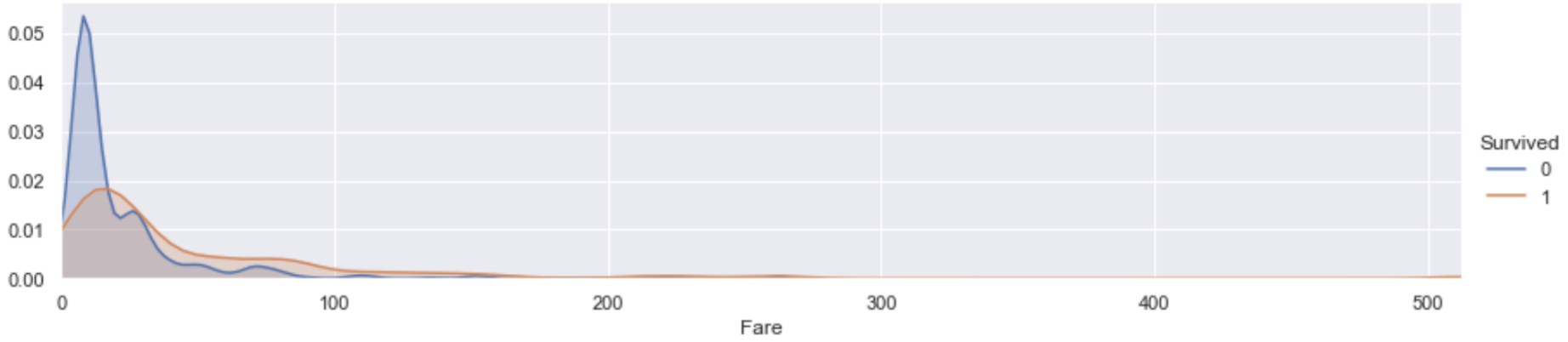
```
# fill missing Fare with median fare for each Pclass
dataset[?].fillna(dataset.groupby(?)[?].transform(?), inplace=True)
dataset.head(50)
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	0	0
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	1	2
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	0	1
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	0	2
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	0	0
5	6	0	3	0	2.0	0	0	330877	8.4583	NaN	2	0
6	7	0	1	0	3.0	0	0	17463	51.8625	E46	0	0
7	8	0	3	0	0.0	3	1	349909	21.0750	NaN	0	3
8	9	1	3	1	2.0	0	2	347742	11.1333	NaN	0	2
9	10	1	2	1	0.0	1	0	237736	30.0708	NaN	1	2
10	11	1	3	1	0.0	1	1	PP 9549	16.7000	G6	0	1
11	12	1	1	1	3.0	0	0	113783	26.5500	C103	0	1
12	13	0	3	0	1.0	0	0	A/5. 2151	8.0500	NaN	0	0
13	14	0	3	0	3.0	1	5	347082	31.2750	NaN	0	0
14	15	0	3	1	0.0	0	0	350406	7.8542	NaN	0	1
15	16	1	2	1	3.0	0	0	248706	16.0000	NaN	0	2

票價/生還死亡分布圖

```
facet = sns.FacetGrid(dataset, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Fare',shade= True)
facet.set(xlim=(0, dataset['Fare'].max()))
facet.add_legend()

plt.show()
```



依票價區間做mapping function

- 票價範圍太大
- 這樣類別太多
- 使用票價區間去mapping成0,1,2,3

?:自行填入的地方

```
dataset.loc[ dataset['Fare'] <= 17, 'Fare'] = ?,
dataset.loc[(dataset['Fare'] > 17) & (dataset['Fare'] <= 30), 'Fare'] = ?,
dataset.loc[(dataset['Fare'] > 30) & (dataset['Fare'] <= 100), 'Fare'] = ?,
dataset.loc[ dataset['Fare'] > 100, 'Fare'] = ?

dataset.head()
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	1.0	1	0	A/5 21171	0.0	NaN	0	0
1	2	1	1	1	3.0	1	0	PC 17599	2.0	C85	1	2
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	0.0	NaN	0	1
3	4	1	1	1	2.0	1	0	113803	2.0	C123	0	2
4	5	0	3	0	2.0	0	0	373450	0.0	NaN	0	0

還是文字且有NaN

資料集欄位

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Title

觀察船艙編號資料

```
dataset['Cabin'].value_counts()
```

```
G6                 4
B96 B98            4
C23 C25 C27        4
C22 C26            3
F2                 3
                  ..
D47                1
C7                 1
B4                 1
C106               1
A31                1
Name: Cabin, Length: 147, dtype: int64
```

觀察船艙編號資料

- 取出第一個字母

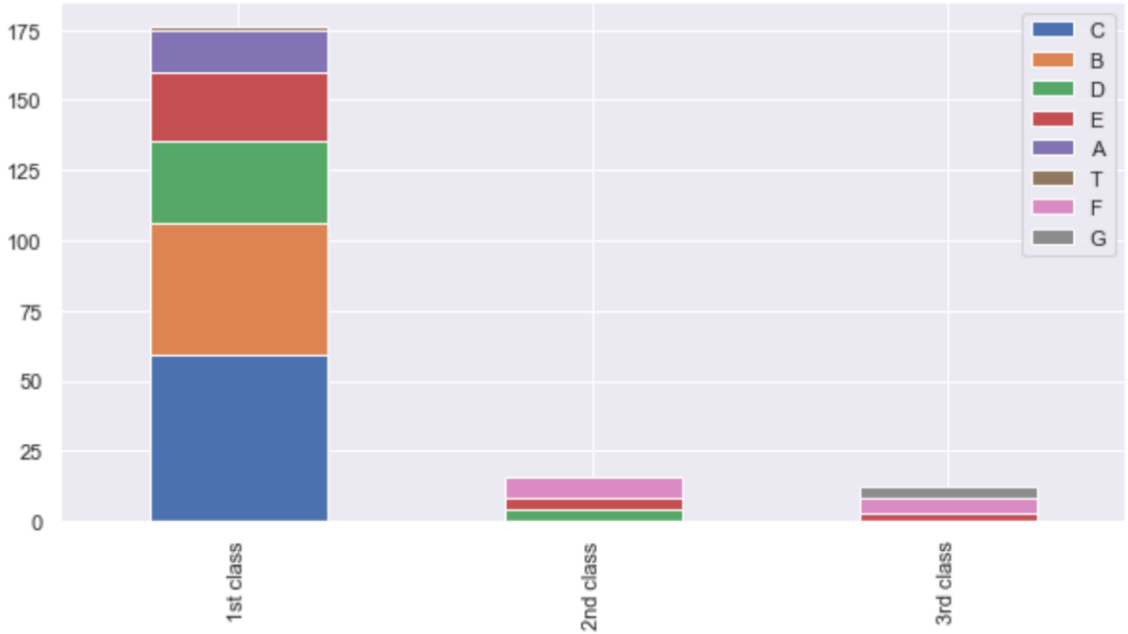
```
dataset['Cabin'] = dataset['Cabin'].str[:1]
dataset['Cabin']
```

```
0      NaN
1        C
2      NaN
3        C
4      NaN
      ...
886    NaN
887      B
888    NaN
889      C
890    NaN
Name: Cabin, Length: 891, dtype: object
```

分析各票種船艙種類人數

- 船艙編號與票種有關
- 分析船艙編號種類與票種之人數

```
Pclass1 = dataset[dataset['Pclass']==1]['Cabin'].value_counts()
Pclass2 = dataset[dataset['Pclass']==2]['Cabin'].value_counts()
Pclass3 = dataset[dataset['Pclass']==3]['Cabin'].value_counts()
df = pd.DataFrame([Pclass1, Pclass2, Pclass3])
df.index = ['1st class','2nd class', '3rd class']
df.plot(kind='bar',stacked=True, figsize=(10,5))
```



填補船艙種類缺失資料

- 如何補值？
- 船艙種類與票種有關
- 先補值？ 先mapping?
- 使用票種分群，再取cabin中位數補值
- 所以先mapping

```
cabin_mapping = {"A": 0, "B": 0.4, "C": 0.8, "D": 1.2, "E": 1.6, "F": 2, "G": 2.4, "T": 2.8}
dataset['Cabin'] = dataset['Cabin'].map(cabin_mapping)
```

想看看：填補船艙種類缺失資料

- 使用票種分群，再取cabin中位數補值

?:自行填入的地方

```
# fill missing Fare with median fare for each Pclass
dataset[?].fillna(dataset.groupby(?)[?].transform(?), inplace=True)
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	0	1.0	1	0	A/5 21171	0.0	2.0	0	0
1	2	1	1	1	3.0	1	0	PC 17599	2.0	0.8	1	2
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	0.0	2.0	0	1
3	4	1	1	1	2.0	1	0	113803	2.0	0.8	0	2
4	5	0	3	0	2.0	0	0	373450	0.0	2.0	0	0

剩下這邊還沒處理

資料集欄位

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Title

觀察Sibsp與Parch

- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic

- 這兩個欄位與家族有關，是否可以合併成一個欄位？

Why?

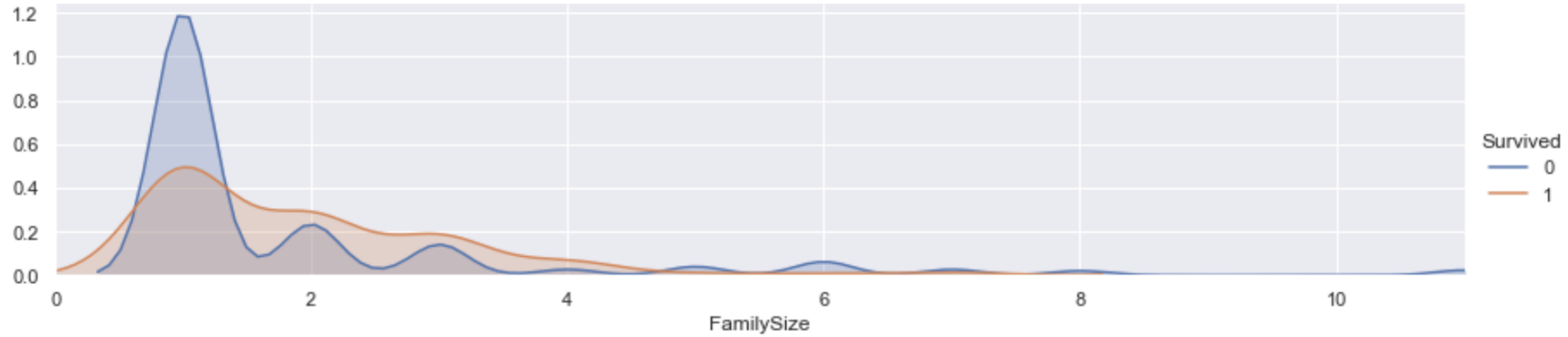
```
dataset["FamilySize"] = dataset["SibSp"] + dataset["Parch"] + 1
```

要加上自己本人！！

家族人口/生還死亡分布圖

```
facet = sns.FacetGrid(dataset, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'FamilySize',shade= True)
facet.set(xlim=(0, dataset['FamilySize'].max()))
facet.add_legend()
plt.xlim(0)
```

```
(0.0, 11.0)
```



家族人口的Mapping Function

```
family_mapping = {1: 0, 2: 0.4, 3: 0.8, 4: 1.2, 5: 1.6, 6: 2, 7: 2.4, 8: 2.8, 9: 3.2, 10: 3.6, 11: 4}
dataset['FamilySize'] = dataset['FamilySize'].map(family_mapping)
```

```
dataset.head()
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	FamilySize
0	1	0	3	0	1.0	1	0	A/5 21171	0.0	2.0	0	0	0.4
1	2	1	1	1	3.0	1	0	PC 17599	2.0	0.8	1	2	0.4
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	0.0	2.0	0	1	0.0
3	4	1	1	1	2.0	1	0	113803	2.0	0.8	0	2	0.4
4	5	0	3	0	2.0	0	0	373450	0.0	2.0	0	0	0.0

資料集欄位

- Survival: 0 = No, 1 = Yes
- Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: male,female
- Age: Age in years
- Sibsp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Title
- FamilySize

資料清洗

- 還有哪些欄位與生存死亡無關？不需要？
  - Survival: 0 = No, 1 = Yes
  - Pclass: Ticket class(1 = 1st, 2 = 2nd, 3 = 3rd)
  - Sex: male,female
  - Age: Age in years
  - Sibsp: # of siblings / spouses aboard the Titanic
  - Parch: # of parents / children aboard the Titanic
  - Ticket: Ticket number
  - Fare: Passenger fare
  - Cabin: Cabin number
  - Embarked: Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
  - Title
  - FamilySize

資料清洗

- 刪除Sibsp, Parch, Ticket

```
features_drop = ['Ticket', 'SibSp', 'Parch','PassengerId']
dataset = dataset.drop(features_drop, axis=1)
```

設定預測目標變數與解釋變數

- 目標變數:survived
- 其餘為解釋變數
- 將欄位分開儲存

```
dataset_data = dataset.drop('Survived', axis=1)
dataset_target = dataset['Survived']

dataset_data.shape, dataset_target.shape
```

```
((891, 8), (891,))
```

應該要有1，但是survived只有一個欄位，
所以變成Series(一維陣列)
若要轉成DataFrame則再加入一個”[]”

```
dataset_data = dataset.drop('Survived', axis=1)
dataset_target = dataset['Survived']



#survived為series，加入中括號轉乘dataframe
dataset_target2 = dataset[['Survived']]

dataset_data.shape, dataset_target.shape,dataset_target2.shape
```

```
((891, 8), (891,), (891, 1))
```

目前的DataFrame

- 所有的解釋變數

```
dataset_data.head()
```

	Pclass	Sex	Age	Fare	Cabin	Embarked	Title	FamilySize
0	3	0	1.0	0.0	2.0	0	0	0.4
1	1	1	3.0	2.0	0.8	1	2	0.4
2	3	1	1.0	0.0	2.0	0	1	0.0
3	1	1	2.0	2.0	0.8	0	2	0.4
4	3	0	2.0	0.0	2.0	0	0	0.0

欄位的資料型態

- 確認訓練資料的型態均為數值型別

```
dataset_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Pclass      891 non-null    int64  
 1   Sex         891 non-null    int64  
 2   Age         891 non-null    float64
 3   Fare        891 non-null    float64
 4   Cabin       891 non-null    float64
 5   Embarked    891 non-null    int64  
 6   Title       891 non-null    int64  
 7   FamilySize  891 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 55.8 KB
```

開始建置模型

- DNN

Keras簡介

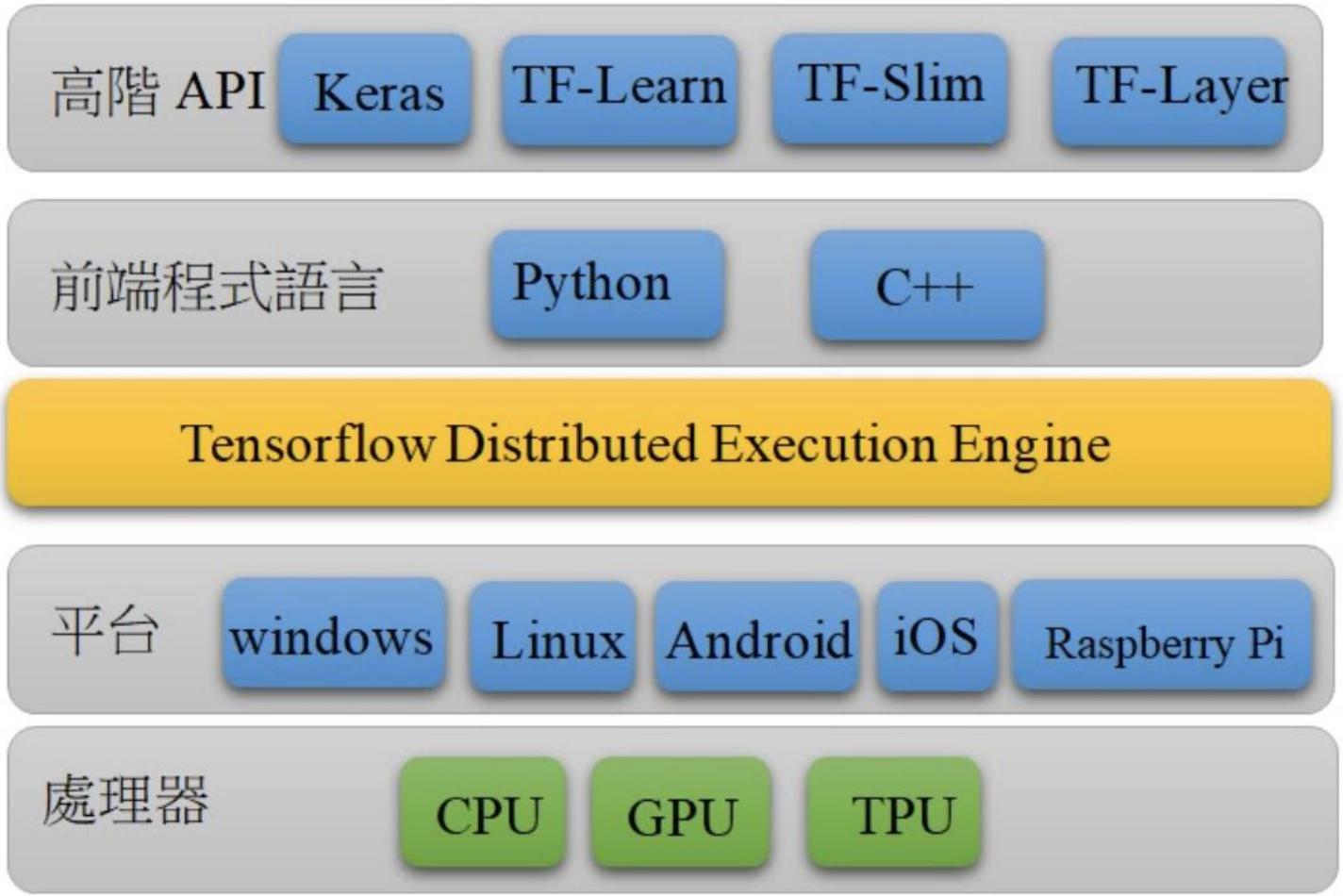
- python的深度學習函式庫
  - 提供非常多常用的深度學習的類神經網路元件
  - 包括卷積層、遞歸層
  - 簡潔、可讀性高
  - Ex:

```
model = Sequential()
model.add(Convolution2D())
model.add(Activation())
model.add(Convolution2D())
model.add(Activation())
model.add(Flatten())
model.add(Dense())
model.add(Activation())
model.add(Dense())
model.add(Activation())
model.compile()
model.fit(X, Y)
```

Tensorflow

- TensorFlow是一個機器學習框架
  - 如果使用者有非常多的資料就可以透過TensorFlow快速的訓練一個有用的模式(例如:利用類神經網路做文字的辨識)。
- TensorFlow是一個開源軟體庫(2015年開始使用)，用於各種感知和語言理解任務的機器學習。
- 目前被50個團隊用於研究和生產許多Google商業產品，如語音辨識、Gmail、Google 相簿和搜尋，其中許多產品曾使用過其前代軟體DistBelief (2011年開始使用，它是Google推出的第一代內部深度學習框架，能夠幫助Google利用自家的數據中心構建大型的神經網絡)。

Keras與Tensorflow



安裝Keras

- pip install keras
- Kearas會需要tensorflow，所以需要安裝tensorflow
- pip install tensorflow

載入Keras 套件/數據預處理

```
from keras.models import Sequential
from keras.layers.core import Dense,Activation
from keras.optimizers import Adam
from sklearn import preprocessing
```

想看看：建立模型函式

```
def build_model():
        #建立模型
        model = Sequential()
        #將模型疊起
        model.add(Dense(input_dim=?,units=40))
        model.add(Activation('relu'))
        model.add(Dense(units=100))
        model.add(Activation('relu'))
        model.add(Dense(units=10))
        model.add(Activation('relu'))
        model.add(Dense(units=?))
        model.add(Activation('sigmoid'))
        model.summary()
        return model
```

?:自行填入的地方

Hint:每筆資料的欄位數?

Hint:二元分類

units: 指定神經元的數量

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 40)                360
_________________________________________________________________
activation (Activation)      (None, 40)                0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               4100
_________________________________________________________________
activation_1 (Activation)    (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
```

建置繪圖函式

```
def show_train_history(train_history,train,validation,label):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train History')
    plt.ylabel(label)
    plt.xlabel('Epoch')
    plt.legend(['train','validation'],loc='upper left')
    plt.show()
```

不同回合的變化

Feature標準化

- 標準化之後的數字。都介於0–1之間

```
#feature標準化
minmax_scale = preprocessing.MinMaxScaler(feature_range=(0, 1))
scaledFeatures=minmax_scale.fit_transform(dataset_data)
```

想看看：訓練模型

?:自行填入的地方

```
model = build_model()
#開始訓練模型
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=['acc'])
train_history = model.fit(?,?,validation_split=0.2,batch_size=30,epochs=20)
```

Hint: 輸入的feature是什麼? 輸入的目標是什麼?

訓練過程

```
Epoch 1/20
24/24 [==============================] - 2s 46ms/step - loss: 0.6673 - acc: 0.6492 - val_loss: 0.6062 - val_acc: 0.80
45
Epoch 2/20
24/24 [==============================] - 0s 3ms/step - loss: 0.5984 - acc: 0.7717 - val_loss: 0.5247 - val_acc: 0.815
6
Epoch 3/20
24/24 [==============================] - 0s 3ms/step - loss: 0.5249 - acc: 0.7913 - val_loss: 0.4416 - val_acc: 0.815
6
Epoch 4/20
24/24 [==============================] - 0s 3ms/step - loss: 0.4416 - acc: 0.8180 - val_loss: 0.3976 - val_acc: 0.849
2
Epoch 5/20
24/24 [==============================] - 0s 3ms/step - loss: 0.4253 - acc: 0.8156 - val_loss: 0.3828 - val_acc: 0.843
6
```

⋮

```
Epoch 16/20
24/24 [==============================] - 0s 3ms/step - loss: 0.3985 - acc: 0.8325 - val_loss: 0.3582 - val_acc: 0.871
5
Epoch 17/20
24/24 [==============================] - 0s 3ms/step - loss: 0.4213 - acc: 0.8278 - val_loss: 0.3537 - val_acc: 0.865
9
Epoch 18/20
24/24 [==============================] - 0s 3ms/step - loss: 0.4103 - acc: 0.8143 - val_loss: 0.3511 - val_acc: 0.865
9
Epoch 19/20
24/24 [==============================] - 0s 3ms/step - loss: 0.4404 - acc: 0.8116 - val_loss: 0.3525 - val_acc: 0.877
1
Epoch 20/20
24/24 [==============================] - 0s 3ms/step - loss: 0.4304 - acc: 0.8232 - val_loss: 0.3555 - val_acc: 0.865
9
28/28 [==============================] - 0s 882us/step - loss: 1.1947 - acc: 0.6801

Train Acc: 0.680134654045105
```

顯示訓練過程

```
#顯示訓練結果
show_train_history(train_history,'acc','val_acc','accuracy')
show_train_history(train_history,'loss','val_loss','loss')
```

備註: 因為模型初始參數是隨機生成，訓練過程的loss與accuracy，
理論上不會跟講義一模一樣，但會是類似的曲線

使用模型進行預測

- 預測Jack和Rose是否生存？

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	3	Jack, Mr.	male	23	0	0	A/5 21171	5	F87	S
2	1	Rose, Miss.	female	20	0	1	PC 17599	100	C85	S

測試資料前處理

- 前面我們將訓練資料的各個欄位做了轉換
- 所以測試資料丟入模型之前，也需要和訓練資料一樣，做欄位的轉換

```
dataset_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Pclass      891 non-null    int64  
 1   Sex         891 non-null    int64  
 2   Age         891 non-null    float64
 3   Fare        891 non-null    float64
 4   Cabin       891 non-null    float64
 5   Embarked    891 non-null    int64  
 6   Title       891 non-null    int64  
 7   FamilySize  891 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 55.8 KB
```

使用模型預測

- 直接預測出機率
- 再從機率判斷是否生還

```
probability = model.predict(testdata)
probability
```

```
array([[0.00393182],
       [0.7465304 ]], dtype=float32)
```

評估模型預測效果

- 給予test的正確答案

```
testdata_target=[0,1]
testdata_target=DataFrame (testdata_target,columns=['Survived'])
testdata,testdata_target
```

```
(    Pclass  Sex  Age  Fare  Cabin  Embarked  Title  FamilySize
 0       3    0  1.0   0.0    2.0         0      0         0.0
 1       1    1  1.0   2.0    0.8         0      1         0.4,
    Survived
 0         0
 1         1)
```

- 使用模型評估預估的準確度

```
score = model.evaluate(x=testdata, y=testdata_target)
print ('\nTest Acc:', score[1])
```

```
1/1 [==========] - 0s 47ms/step - loss: 0.1481 - acc: 1.0000

Test Acc: 1.0
```

模型的內容 (1/2)

- 模型就是最佳的w,b的組合

```
#get weights                                          get第一層的w,b
W, b = model.layers[0].get_weights()
print("weights = {}, \n\n biases= {}".format(W, b))
```

模型的內容 (2/2)

一個dimension
有40個weights
(因為40個神經元)

```
weights = [[-0.3367335   0.3044695   0.4089243   0.0733672   0.32975808  0.25855696
  -0.2602524  -0.02493989 -0.16008916 -0.15439107  0.32407358 -0.12091316
   0.23078144 -0.04634802  0.32769966 -0.01082976 -0.18585522  0.01145001
   0.08934474  0.26414979  0.26524583  0.05979327 -0.16150229  0.28055876
   0.2356595   0.081496    0.11983839 -0.08324693 -0.2989114   0.19593443
  -0.3379509  -0.34954256  0.24753243 -0.12448855 -0.29598483  0.32633173
  -0.21903029 -0.6423083   0.3132761   0.15683785]
 [ 0.24290541 -0.08765236 -0.2252202  -0.19573006  0.18568063  0.18174979
   0.02070362 -0.12933183 -0.0523845   0.02577734  0.09819992  0.20554914
  -0.05855617 -0.38649634  0.23978125  0.2655271   0.2877766   0.2494148
   0.3679629  -0.2524917  -0.17983025 -0.24823222  0.13873811 -0.16411321
   0.15423532  0.29401064 -0.11695129 -0.22810999  0.26889366  0.05692591
  -0.22621153 -0.30027694  0.20515452  0.38684857  0.02287883  0.02057466
  -0.27339295  0.20191331  0.16011843 -0.2932741 ]
 [ 0.04291374  0.2831202   0.3073469   0.32786733 -0.30926442 -0.37983805
  -0.24020426  0.03493049 -0.11916129 -0.11684003 -0.25786567 -0.08324005
  -0.1893343  -0.25938305 -0.14270025  0.21357551  0.04674786 -0.2047807
  -0.05158462 -0.0845309   0.23675735 -0.13837901  0.21520457  0.13058992
   0.285413   -0.09508758  0.43250683  0.15740386  0.32179776 -0.37911007
   0.15066333  0.34372663 -0.2637193  -0.04262189 -0.10576533  0.12387013
   0.29142663  0.11003023  0.10823315 -0.34683308]
 [ 0.4372344  -0.15596156 -0.06294589  0.24978037  0.14708363 -0.39674902
   0.31105673  0.13530548  0.46444255  0.39709967 -0.04913068 -0.26187438
  -0.5998156   0.16542107  0.174921    0.03693036  0.02343213  0.31883574
   0.35300088  0.35061085 -0.3334052  -0.00439236  0.38311052 -0.08826411
   0.40019906 -0.03352882 -0.18386348 -0.2599297   0.41032526 -0.06067289
```

8個dimension

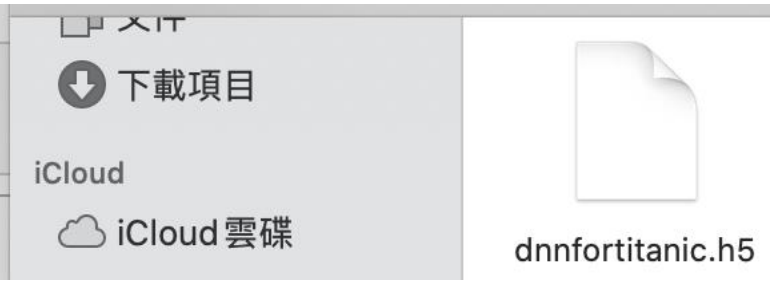
```
biases= [ 0.07150973 -0.02876623  0.00162051  0.01062156 -0.05102419  0.04291807
  0.07323542 -0.00013486 -0.02148618  0.05605646  0.00280015 -0.02650807
  0.0241123   0.05560525 -0.00270608  0.05310432  0.01623039  0.05259741
  0.03690061  0.01868107  0.03088869  0.05800183 -0.02548092 -0.01056132
 -0.07064764  0.01645724  0.00870797 -0.0130707   0.08753705  0.00760644
  0.03149424  0.06765343  0.03085114  0.0133551   0.          0.02302689
 -0.00028909  0.07383139 -0.03774234  0.        ]
```

40個神經元，所以有40個bias

儲存模型

- 以HDF5格式儲存模型
  - **HDF5** lets you store huge amounts of numerical data, and easily manipulate that data from NumPy.

```
9]: #存檔模型
    model.save('dnnfortitanic.h5')
```



- 下次要使用模型時，再load_model

```
#下次要使用模型時
from keras.models import load_model
model=load_model('dnnfortitanic.h5')
```

要先import