Lab : DNN

Prof. Chia-Yu Lin

Environment Setup

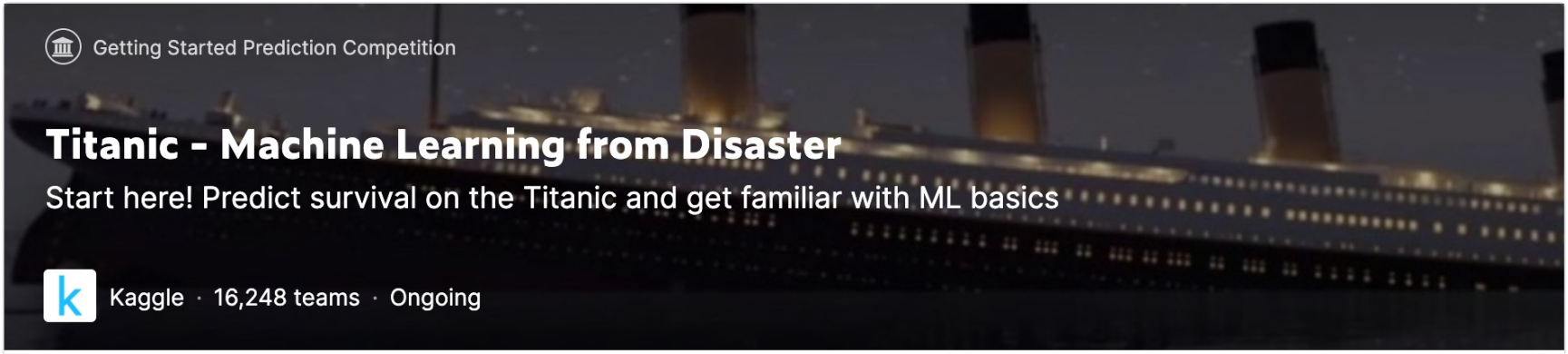
- Open Jupyter Notebook

- Enter the working directory where you store the files
- Open DNN.ipynb

Titanic Survival Prediction

Data Source

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



Install Library

- Seaborn
  - A library for making Matplotlib's charts more beautiful, just importing can make the charts more beautiful, and you can specify several additional styles
- Install seaborn
  - pip install seaborn

Import Library

```
#Import data preprocessing modules
import numpy as np
import numpy.random as random
import scipy as sp
from pandas import Series, DataFrame
import pandas as pd

# Visualization modules
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
%matplotlib inline

# Machine learning module
import sklearn
```

Data Preprocessing

- What is the first step when you receive the data?
  - Observe Data
  - Data Visualization
  - Data Cleaning
  - Fill missing value
  - Mapping the text to 0,1,2....

Dataset Description

- 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)

Observe Data

- Reading the file
- Print the first 50 rows

```
1 #Read the file
2 dataset = pd.read_csv('input/titanic.csv')
3 dataset.head(50) #Show the first 50 rows
```

	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

Observe Data

- Observe data shape

```
1 #Observe the data shape
2 dataset.shape
```

```
(891, 12)
```

- Observe data information
- 12 columns

```
1 #Observe the data information
2 dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

- There are three types of columns
  - float64 (there are 2)
  - int64 (there are 5)
  - object (there are 5)

Data Visualization

- Set seaborn as the default drawing library.

```
1  #Set seaborn as the default drawing library
2  sns.set()
```

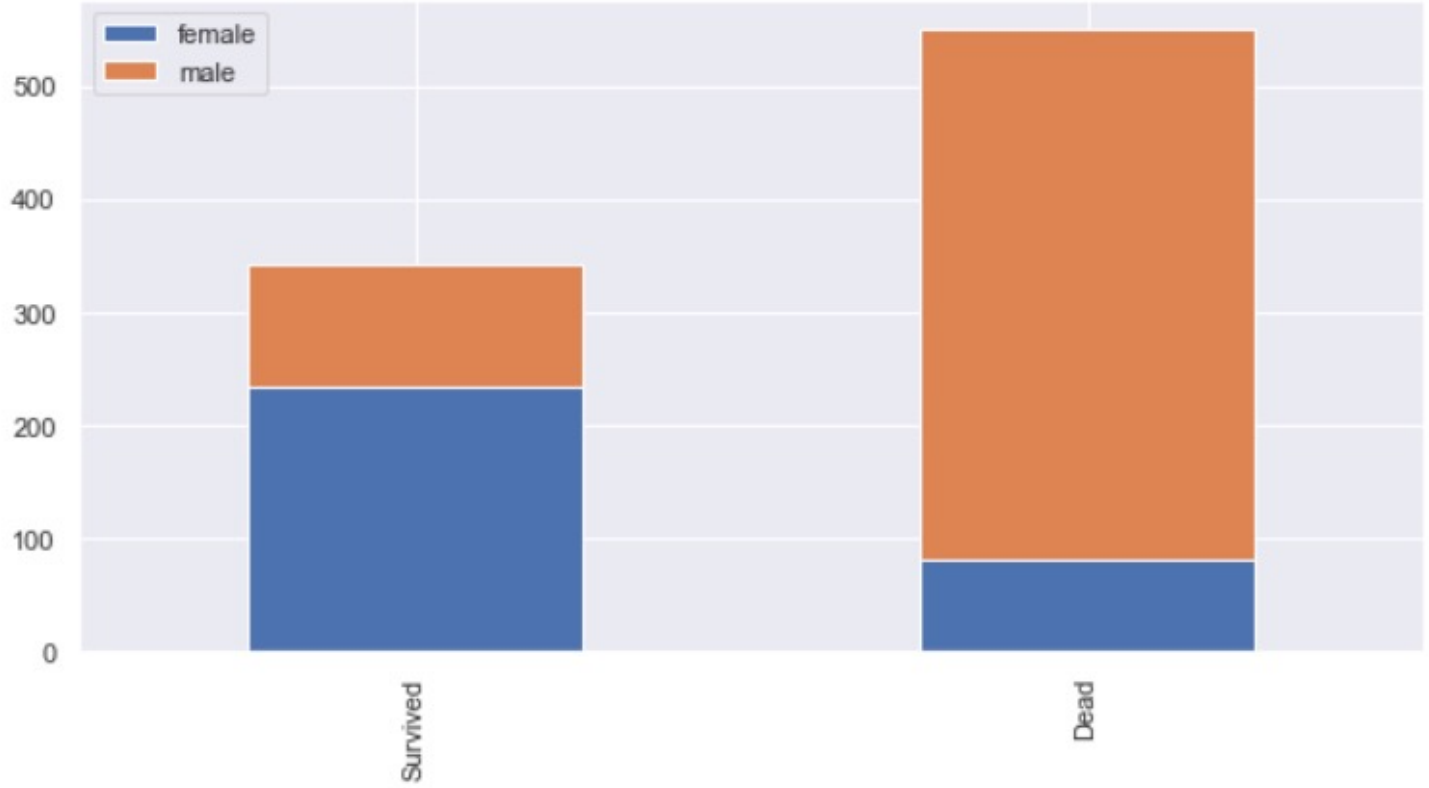
Set Drawing Function

- Since we want to predict the survival rate, we take "survived, dead" as the X-axis.
- Plot the relationship of other features relative to "survived, dead."

```
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))
```

Bar Chat of Survival/Dead Men and Women

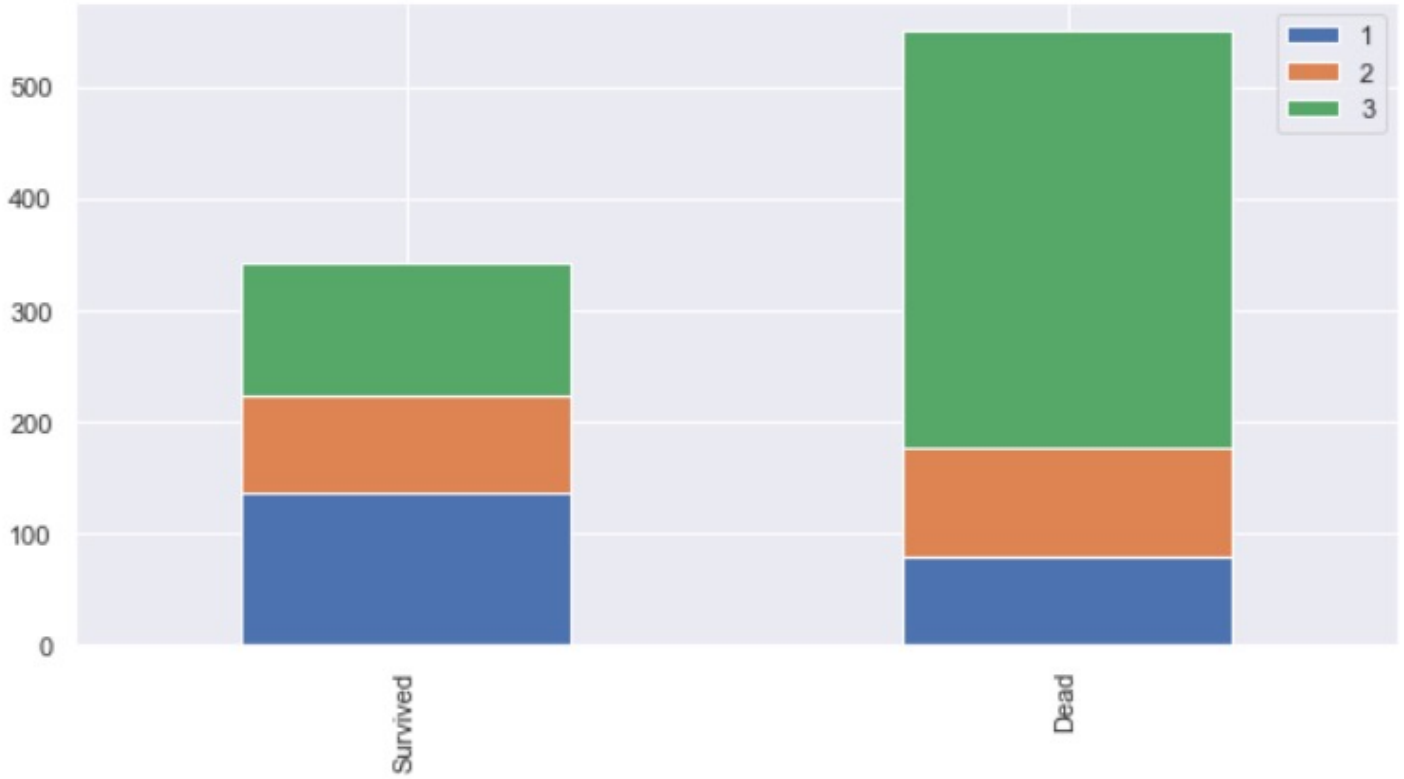
```
1 #Bar Chat of Survival/Dead Men and Women
2 bar_chart('Sex')
```



The Chart confirms Women more likely survivied than Men

Bar Chat of the Cabin of the Survival/Dead Person

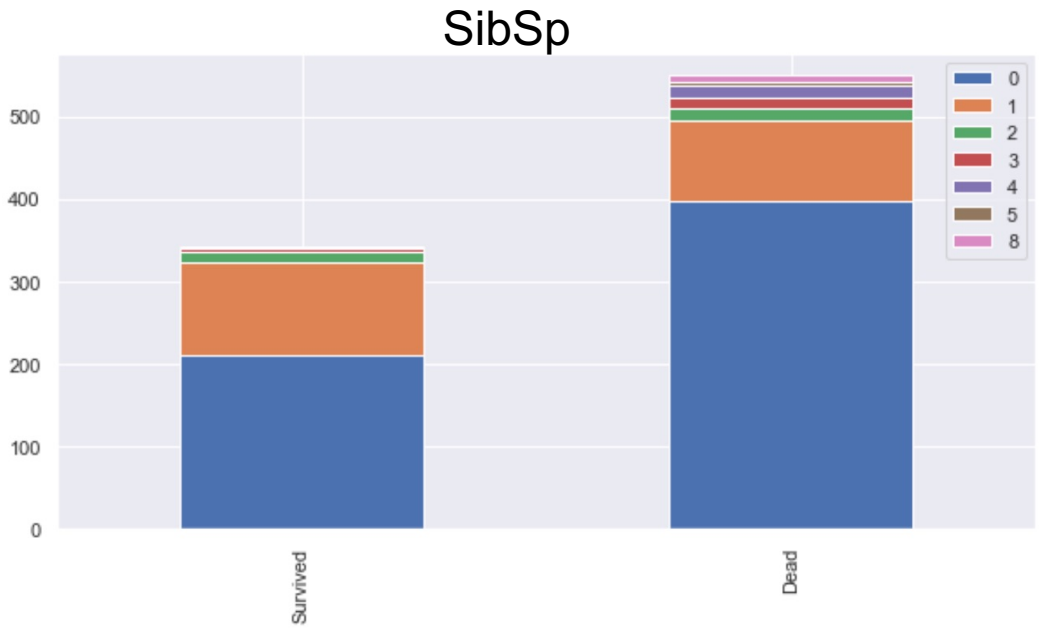
```
1 #Bar Chat of the Cabin of the Survival/Dead Person
2 bar_chart('Pclass')
```



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

bar_chart()

- Input any feature in this function.
- You can observe the features you want to observe.
- You can input any feature you want.



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

Fill missing value

- There are NaN in the dataframe

	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

How many NaNs are in the dataframe (1/3)

- How to find the number of NaNs?
- Remember how we looked at how many "?" when predicting the price of a car

Calculate the number of “?” in the columns

```
dataset.isin(['?']).sum()
```

Calculate the number of “NaN” in the columns

```
dataset.isin(['NaN']).sum()
```

Yes or No??

How many NaNs are in the dataframe (2/3)

- How many NaN in the dataframe?

```
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
```

- Why??
- Because NaN is a special character, it cannot be judged by a string

How many NaNs are in the dataframe (3/3)

- How many NaN in the dataframe?

```
1 #Calculate the number of "NaN" in the column
2 dataset.isnull().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
```

Fill missing value

- How to fill missing value?
- Observe each column to find the column we can choose as reference.

	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

Observe the Name column

- Observe the Name column

```
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
```

- The name implies many additional information Mr., Mrs., Dr, Miss….
- This can be used as a category reference

Define a new column

- Pick out words that match the title of "A-Za-z+."
  - ex. Mr. Mrs.

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

- Regular expression usage
- https://reurl.cc/qeZQE
- https://reurl.cc/Neb8n

Count the Number of People by Each Title

```
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

Think: Mapping Function

- Define a mapping function according to these three categories
- Uniformly numbered with 0, 1, 2, 3

?:fill in by yourself

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

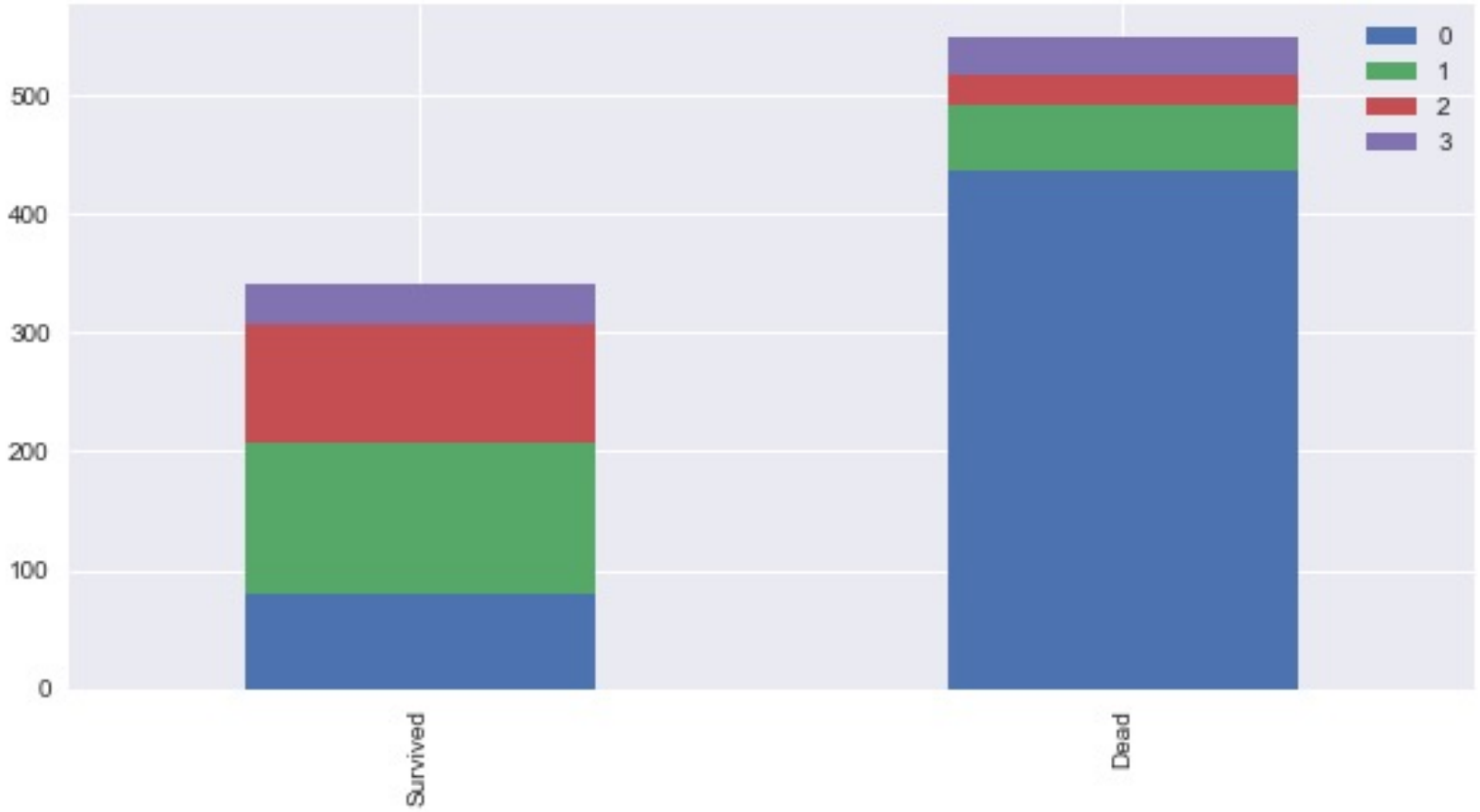
```
dataset.head()
```

	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

Check if the result is the same as here

Show the Distribution of Title/Survival and Dead

- How to write? ?



Data Cleaning (1/2)

- Remove the useless column.
- The title of Name has been converted to a new class.
- Name has nothing to do with survival/death.
- We donot need to keep the “Name”, we can delete the Name column.
- How to write?

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

Data Cleaning (2/2)

- After deleting the Name column

```
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

- The value of Sex column is “text” and we need to deal with this column.

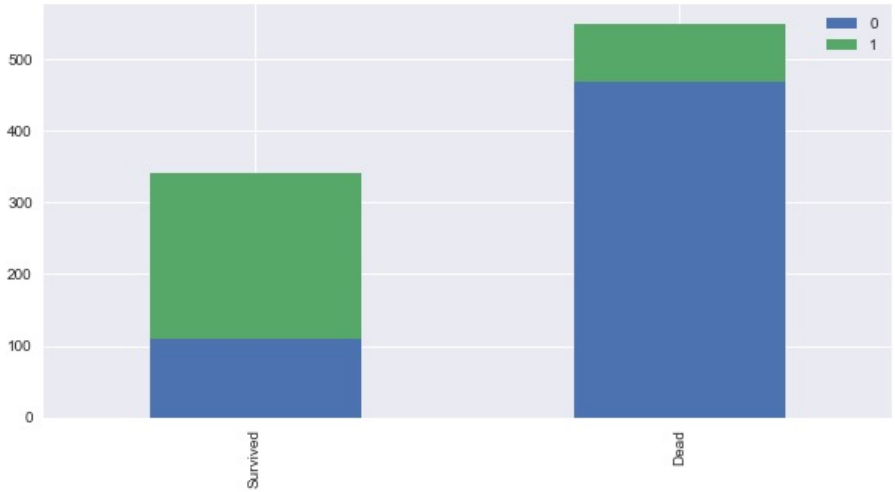
Think: The Effect of Gender on Survival/Death

- What are the class name in the "Sex" column?
  - Men
  - Women
- 0 for men and 1 for women

?:fill in by yourself

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

bar_chart('Sex')
```



Current 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

Dataset column

- 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

Fill Missing Value：Age Column

- How to fill ???
- Fill in the age of mean ?
- Fill in the age of the previous row ?
- Age may be related to Mr. , Mrs. , Miss
- Use Title as the refer of grouping to fill in the median age of the group.

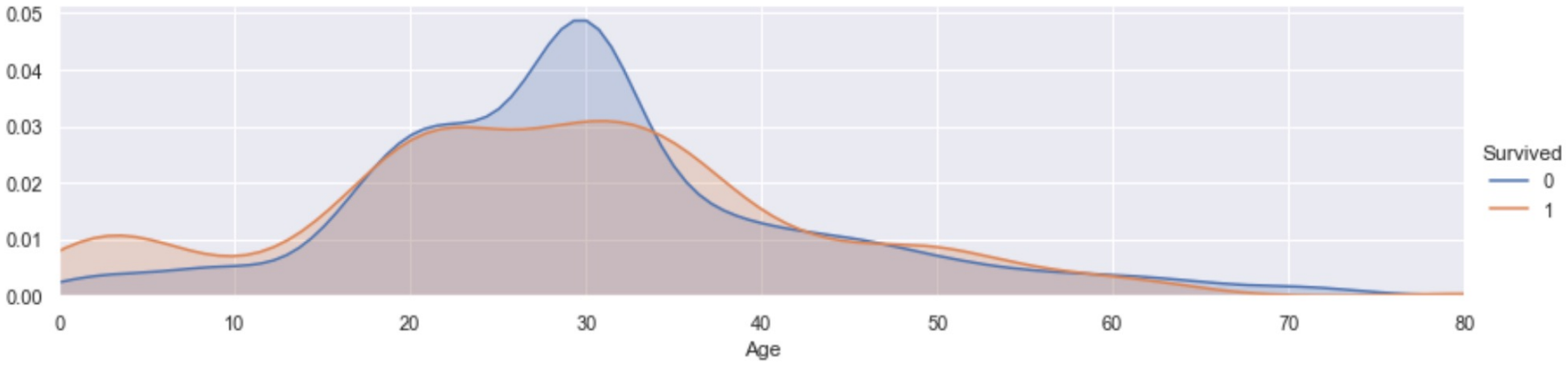
```
# fill missing age with median age for each title (Mr, Mrs, Miss, Others)
dataset["Age"].fillna(dataset.groupby("Title")["Age"].transform("median"), inplace=True)
dataset["Age"]
```

```
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
```

Distribution of Age/Survival and Death

```
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()
```



If the following warnings appear, just ignore them without affecting subsequent experiments

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

Mapping Function According to the Range of Age

- The range of age is too large.
- There are too many categories.
- We define the range of age and map to 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
```

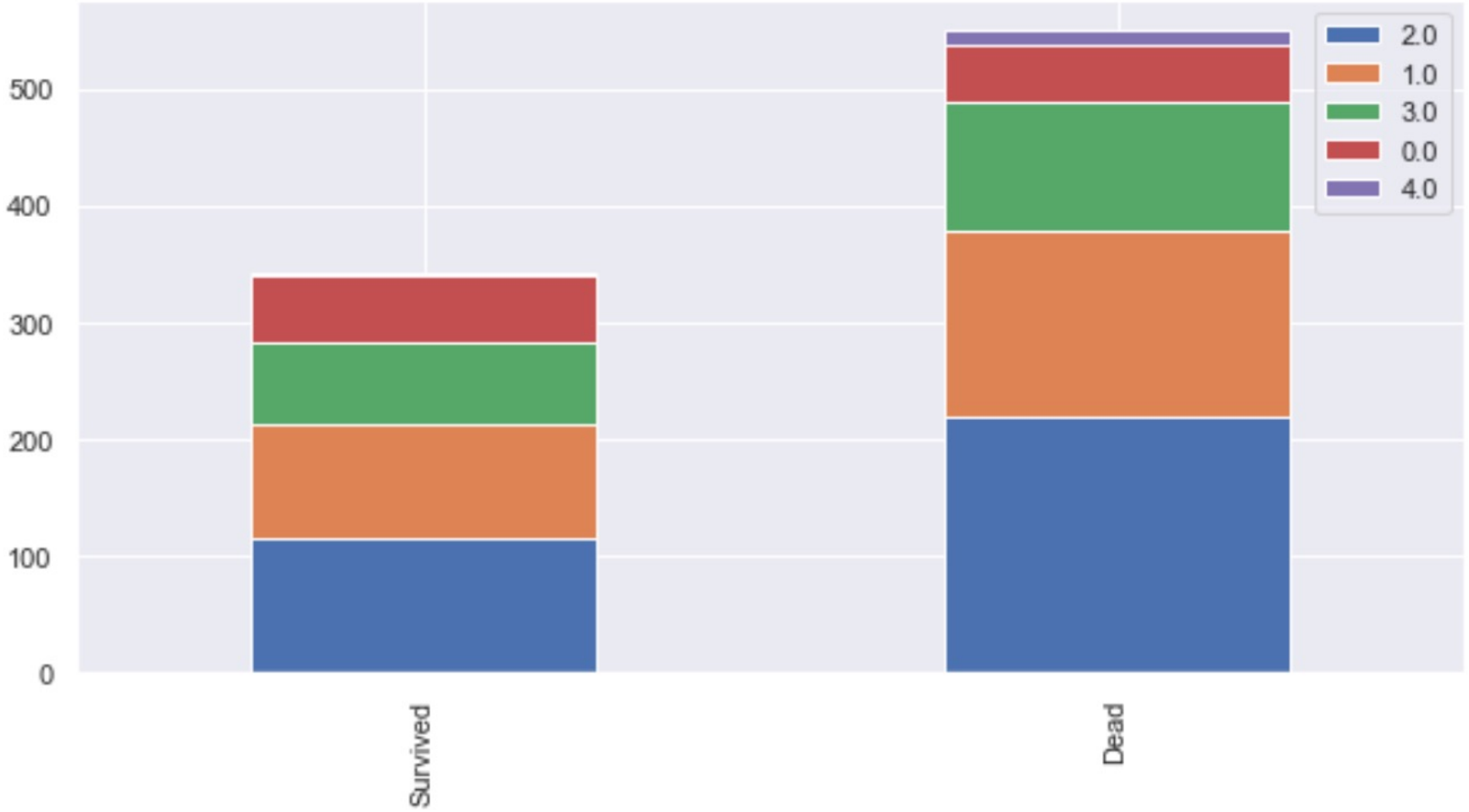
Current DataFrame

```
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

Think: Bar Chart of Age

- How to do ？ ？

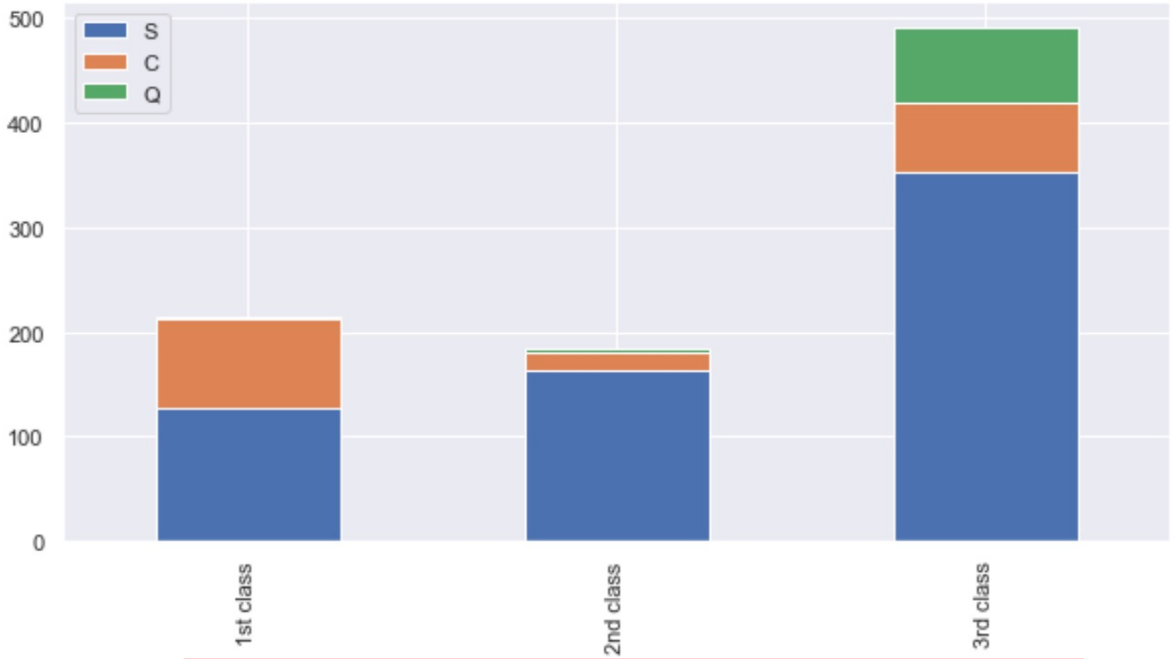


Dataset column

- 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

Analyze the Type of Boarding Tickets at Each Boarding Location

```
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.

Fill Missing Value for Embarked Location

- How to fill ?
- Because there are more than 50% embark “s” of each type of boarding ticket .
- So we can use “s” to fill the embarked location.

```
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

Think: Mapping Function for the Embarked Location

- S is set to 0, C is set to 1, Q is set to 2

?:fill in by yourself

```
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

Dataset column

- 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

Think：Fill in Missing Value for Fares (1/2)

- There are no missing values for the fare of the training data.
- However, the fare of the test data may be missing.
- So we still need to design the filling method in the training data.

- You can refer to the method to fill in the missing value for the age.
- How to do? ?
- What is the “fare” related to?

Think：Fill in Missing Value for Fares (2/2)

?:fill in by yourself

```
# 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

Distribution of Fare/Survival and Death

```
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()
```

Mapping Function According to the Range of Fare

- Fare range is too large
- So many categories
- Use the fare interval to map to 0,1,2,3

?:fill in by yourself

```
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

Dataset column

- 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

Observe the Cabin Number

```
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
```

Observe the cabin number

- Take out the first letter

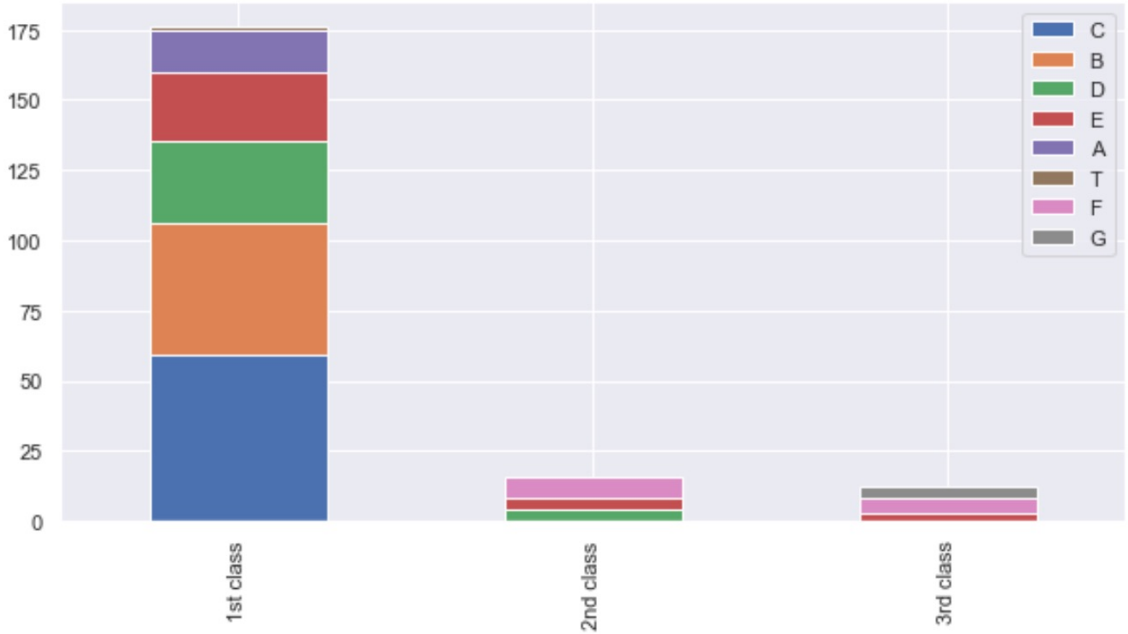
```
dataset['Cabin'] = dataset['Cabin'].str[:1]
dataset['Cabin']
```

```
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
```

Analyze the Number of People in Various Ticket Types and Cabin Types

- The cabin number is related to the ticket type

```
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))
```



Fill in the Missing Value on the Type of Cabin

- How to fill ?
- The type of cabin is related to the type of ticket.
- Fill missing value first? Mapping first?
- Use the ticket type to group, and then take the cabin median.
- So we first do 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)
```

Think: Fill in the Missing Value on the Type of Cabin

- Use the ticket type to group, and then take the cabin median

?:fill in by yourself

```
# 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

The rest has not been dealt with here

Dataset column

- 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

Observe Sibsp and Parch

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

- These two columns are related to family, can they be combined into one column?

Why?

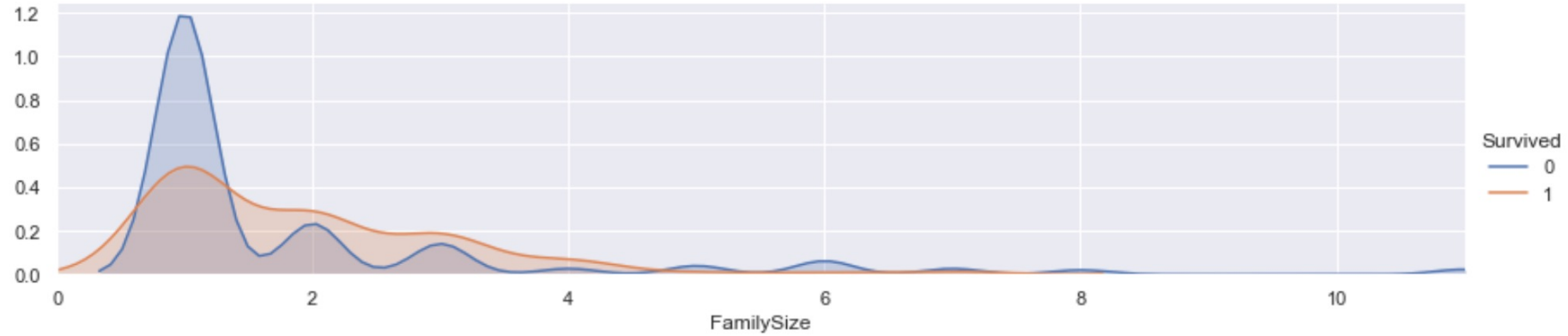
```
dataset["FamilySize"] = dataset["SibSp"] + dataset["Parch"] + 1
```

Add yourself !!

Distribution of Family Population/Survival and Death

```
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)
```



The Mapping Function of Family Population


```
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

Dataset column

- 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

Data Cleaning

- What other columns are not related to survival and death? unnecessary ?
  - 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

Data Cleaning

- Delete Sibsp, Parch, Ticket

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

Set Target Variables and Explanatory Variables

- Target variables: survived
- The rest variables are explanatory variables.
- Store the columns separately.

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

dataset_data.shape, dataset_target.shape
```

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

There should be 1, but survived has only one column,
So it becomes a Series (one-dimensional array)
To convert to DataFrame, add another "[ ]"

```
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))
```

Current DataFrame

- All explanatory variables.

```
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

Data type each column

- Check the types of every column in the training data.
- They should be numeric type.

```
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
```

Start building the model

- DNN

Introduction of Keras

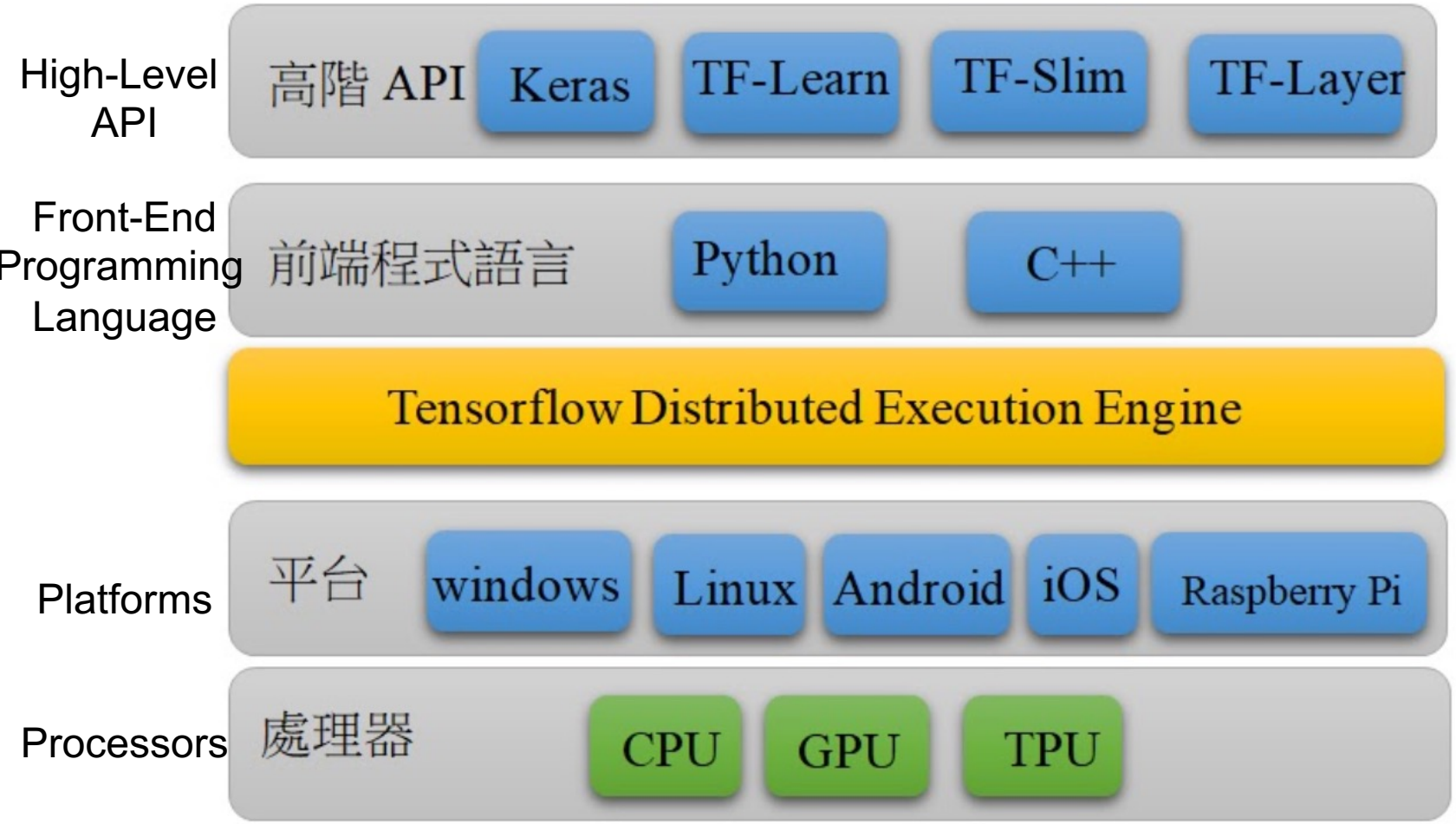
- Keras is a deep learning library for python.
  - Keras provides a lot of commonly used deep learning neural network components, including convolutional layers, recursive layers …etc.
  - Concise and readable.
  - Ex:

```
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 is a machine learning framework
  - If the user has a lot of data, a useful model can be quickly trained through TensorFlow (for example: using a neural network for text recognition)
- TensorFlow is an open-source software library (started in 2015) for machine learning for various perceptual and language understanding tasks.
- Tensorflow is currently used by 50 teams to research and produce many of Google's commercial products, such as speech recognition, Gmail, Google Photos, and Search, many of which used its predecessor DistBelief (started in 2011, Google's first software).
- A generation of in-house deep learning frameworks that can help Google build large neural networks using its own data centers).

Keras and Tensorflow



Install Keras

- pip install keras
- Keras will need tensorflow, so you need to install tensorflow
- pip install tensorflow

Import Keras/Data Preprocessing


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

Think：Create Model Function

```
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
```

?:fill in by yourself

Hint: The number of columns for each data?

Hint: Binary Classification

units: specify the number of neurons

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

Create a Drawing Function

```
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()
```

Changes each round

Feature Normalization

- Normalized numbers. are between 0–1

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

Think：Train Model

- model.fit: Train model

?:fill in by yourself

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

- model.evaluate: Evaluating the model.
- The model generates output based on the input and calculate the metrics function specified in “model.compile.”
- Based on “y_true and y_pred”, derive the loss and standard precision.

```
#Evaluate training result
score = model.evaluate(x=scaledFeatures, y=dataset_target)
print ('\nTrain Loss:', score[0])
print ('\nTrain Acc:', score[1])
```

Hint: What is the input feature? What is the input target?

Training Process

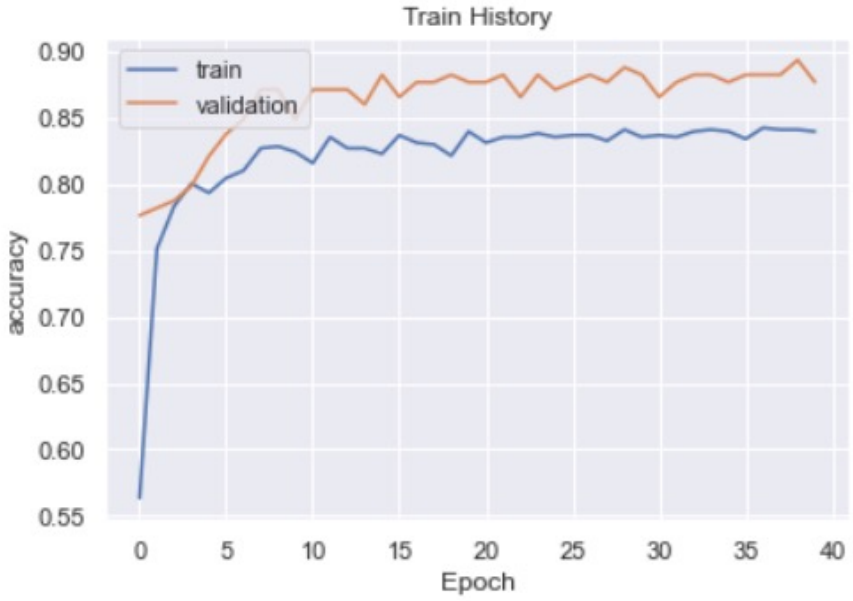
```
Epoch 1/40
24/24 [==============================] - 1s 11ms/step - loss: 0.6873 - acc: 0.4625 - val_loss: 0.6523 - val_acc: 0.77
65
Epoch 2/40
24/24 [==============================] - 0s 4ms/step - loss: 0.6351 - acc: 0.7517 - val_loss: 0.5806 - val_acc: 0.782
1
Epoch 3/40
24/24 [==============================] - 0s 3ms/step - loss: 0.5708 - acc: 0.7708 - val_loss: 0.4795 - val_acc: 0.787
7
Epoch 4/40
24/24 [==============================] - 0s 4ms/step - loss: 0.4746 - acc: 0.8035 - val_loss: 0.4177 - val_acc: 0.798
9
Epoch 5/40
24/24 [==============================] - 0s 4ms/step - loss: 0.4577 - acc: 0.7847 - val_loss: 0.3925 - val_acc: 0.821
2
Epoch 6/40
24/24 [==============================] - 0s 3ms/step - loss: 0.4535 - acc: 0.7926 - val_loss: 0.3810 - val_acc: 0.838
0
                                    ⋮
Epoch 37/40
24/24 [==============================] - 0s 3ms/step - loss: 0.3964 - acc: 0.8401 - val_loss: 0.3325 - val_acc: 0.882
7
Epoch 38/40
24/24 [==============================] - 0s 3ms/step - loss: 0.3940 - acc: 0.8357 - val_loss: 0.3295 - val_acc: 0.882
7
Epoch 39/40
24/24 [==============================] - 0s 4ms/step - loss: 0.4159 - acc: 0.8336 - val_loss: 0.3319 - val_acc: 0.893
9
Epoch 40/40
24/24 [==============================] - 0s 3ms/step - loss: 0.3454 - acc: 0.8596 - val_loss: 0.3394 - val_acc: 0.877
1
28/28 [==============================] - 0s 1ms/step - loss: 0.3802 - acc: 0.8496
```

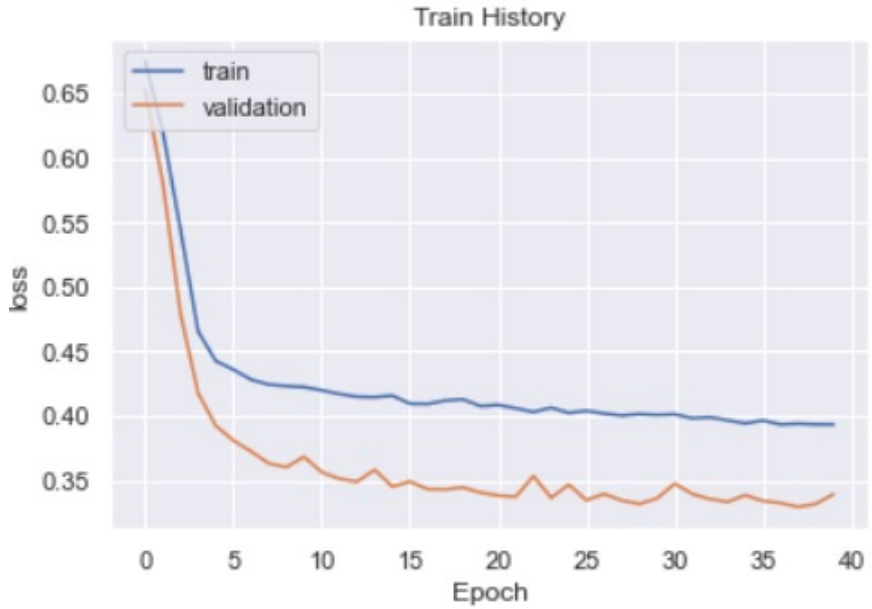
```
Train Loss: 0.3801676034927368

Train Acc: 0.8496071696281433
```

Show Training Process

```
show_train_history(train_history,'acc','val_acc','accuracy')
show_train_history(train_history,'loss','val_loss','loss')
```





Note: Because the initial parameters of the model are randomly generated, the loss and accuracy of the training process won't be exactly same with the slide, but it will be a similar curve.

Use the model to predict

- Predict whether Jack and Rose will survive?

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

Testing Data Preprocessing

- We converted each column of the training data.
- Therefore, before the testing data input the model, it is also necessary to convert columns like the training data.

```
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
```

Use the model to predict

- Use “model.predict" to predict survival probability
- Then judge whether to survive or not according to the probability.

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

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

Model Content (1/2)

- The model is the best combination of w and b.

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

Model Content (2/2)

A dimension has 40 weights (because of 40 neurons)

```
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 dimension

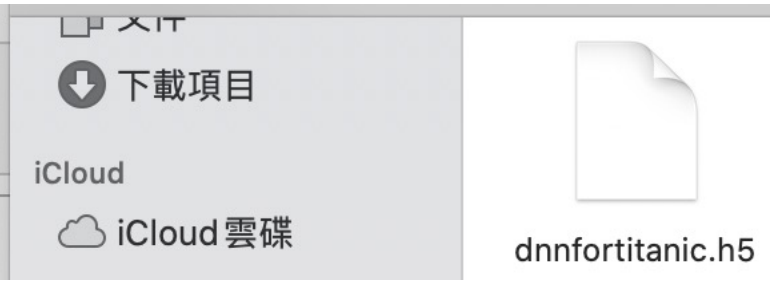
```
biases= [ 0.07150973 -0.02876623  0.00162051  0.01062156 -0.05102419  0.04291807
  0.07323542 -0.00013486 -0.02148618  0.05605646  0.00280015 -0.02650807
  0.0241123   0.05560525 -0.00270608  0.05310432  0.01623039  0.05259741
  0.03690061  0.01868107  0.03088869  0.05800183 -0.02548092 -0.01056132
 -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 neurons，40 bias

Save Model

- Save model in HDF5 format
  - **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')
```



- The next time when you want to use the model, you can load_model.

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

need import

Do it yourself：predict Kaggle's Tesing Data

1. model.predict predict survival probability, then judge whether or not to survive according to the probability

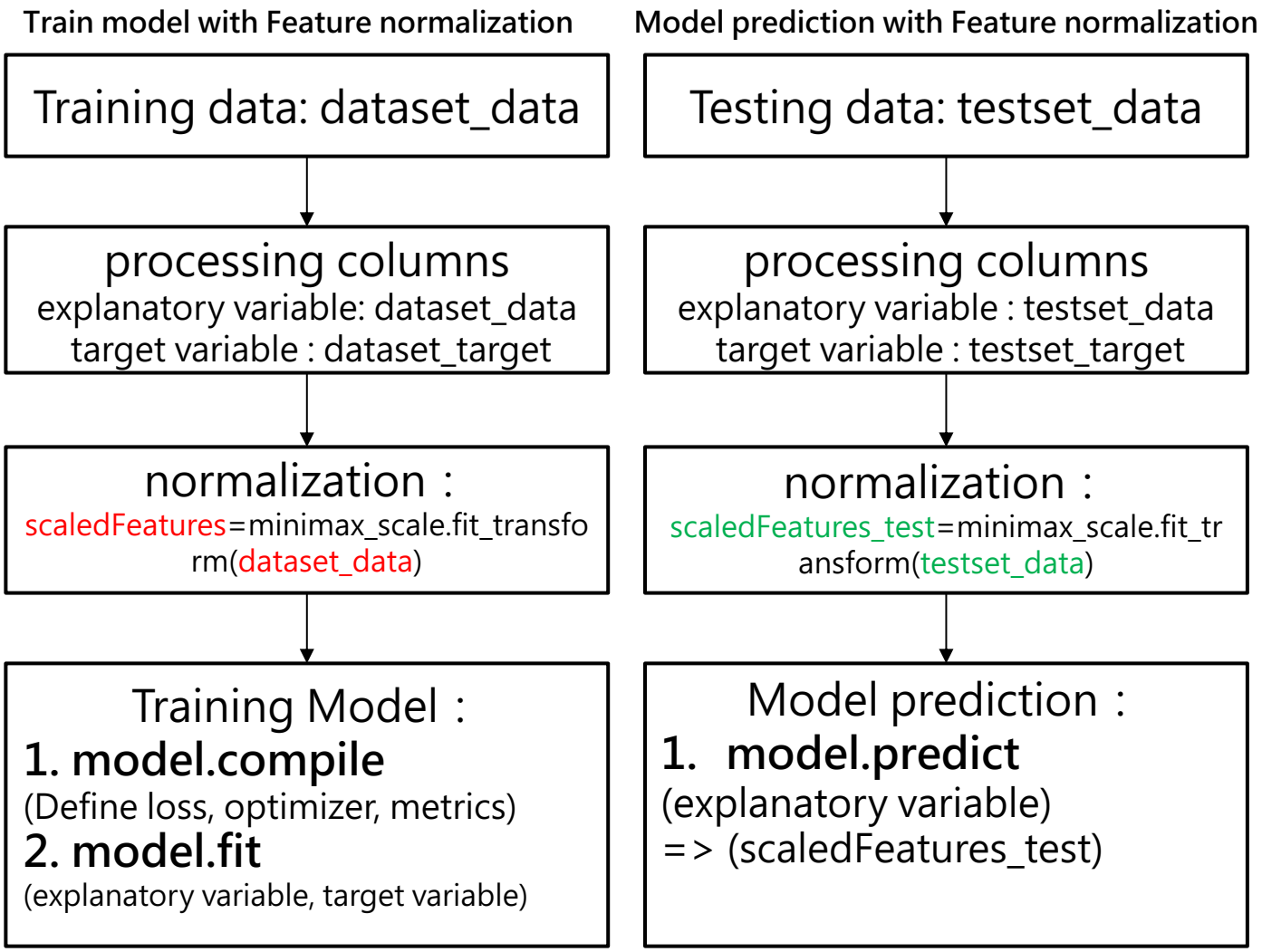
2. Because the prediction result is two-dimensional, it can only be one-dimensional to be stored in csv, so do flatten

```
probability=probability.flatten()
```

3. write submission.csv

```
testdata_write = pd.read_csv('input/test.csv')
submission = pd.DataFrame({
                            "PassengerId": testdata_write['PassengerId'],
                            "Survived": probability
                        })
submission.to_csv('submission.csv', index=False)
```

Summary


Train model with Feature normalization
Training data: dataset_data
processing columns
explanatory variable: dataset_data
target variable : dataset_target
normalization：
scaledFeatures=minimax_scale.fit_transform(dataset_data)
Training Model：
1. model.compile
(Define loss, optimizer, metrics)
2. model.fit
(explanatory variable, target variable)
Model prediction with Feature normalization
Testing data: testset_data
processing columns
explanatory variable : testset_data
target variable : testset_target
normalization：
scaledFeatures_test=minimax_scale.fit_transform(testset_data)
Model prediction：
1. model.predict
(explanatory variable)
=> (scaledFeatures_test)