

```
[3] #Trong phân tích dữ liệu thường hay sử dụng các thư viện này để làm
import pandas as pd
import numpy as np
import seaborn as sb
```

```
[4] #Thu thập dữ liệu gốc, chuyển về dưới dạng CSV, đưa lên google driver để chạy
df = pd.read_csv("/content/drive/MyDrive/Data Analys/titanic.csv")
```

```
#Đọc file vừa Load lưu trên df (Data Frame)
#Thông tin các khách hàng trong vụ chìm tàu Titanic huyền thoại, Survived (Được cứu -1/ không được cứu - 0)
#Trường survived là trường kết quả (Quyết định)
#các trường khác là các thuộc tính ảnh hưởng tới việc được cứu/ không được cứu này
#Đó là PClass - Phân tầng hạng vé thượng lưu - 1/trung lưu-2/hạ lưu-3. ta thường biết đội thượng lưu sẽ ở vị trí dễ cứu nhất
# Và hạ lưu sẽ ở dưới các khoang dưới của tàu, cho nên chết nhiều nhất lớp này
# sex - Giới tính, sibsp - số người thân (Vợ chồng) trên tàu, Parch - số người thân (Con cái. bố mẹ)
#Ticket - Số vé, Fare - Giá vé hành khách, Cabin - Mã số ca bin, Embarked - Cổng vào
df
```

	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

```
[6] df.head(10)
```

	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

✓
1
giây

```
#Xóa một số cột không được sử dụng trong phân lớp
#Phân tích dữ liệu: Ta thấy rằng Tên hành khách (Name), Mã số vé (Ticket), Mã số cabin (Cabin) là không ảnh hưởng tới việc được cứu hay
#Không được cứu. Cho nên ta sẽ làm sạch dữ liệu bằng cách xóa các cột dưới đây khỏi Data Frame:
#name, ticket, cabin
cols_to_drop = ['Name', 'Ticket', 'Cabin']
df = df.drop(cols_to_drop,axis=1)
```

✓
0
giây

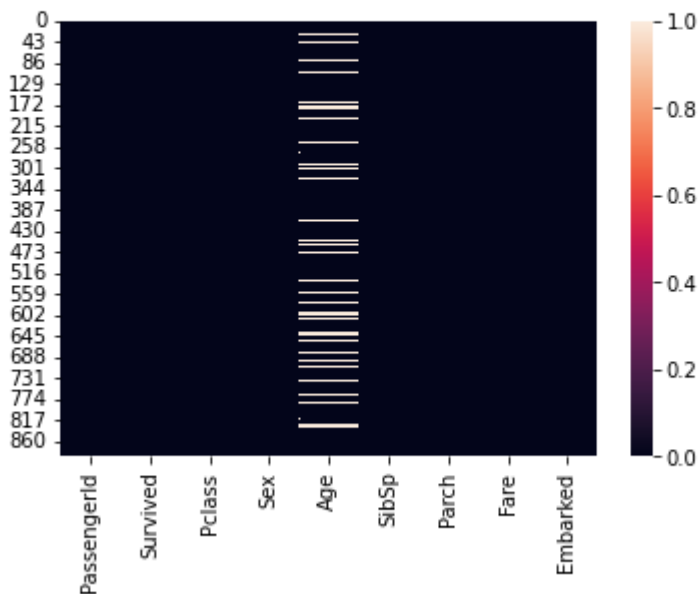
[8] #Dữ liệu sau khi xóa
df

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S
...
886	887	0	2	male	27.0	0	0	13.0000	S
887	888	1	1	female	19.0	0	0	30.0000	S
...

✓
1
giây

```
#Chuẩn hóa dữ liệu
#Kiểm tra có ô nào có giá trị NULL - Not Value không?
sb.heatmap(df.isnull())
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fb69217c1d0>



✓
0 giây [10] #Ví dụ hàng 43
df.iloc[42]

```
PassengerId      43
Survived          0
Pclass           3
Sex              male
Age              NaN
SibSp            0
Parch            0
Fare            7.8958
Embarked         C
Name: 42, dtype: object
```

✓
0 giây [11] #Ta thấy cột Age là NaN - Rỗng
df['Age'].iloc[42]

nan

✓
0 giây [12] #Thay thế các giá trị rỗng bằng các giá trị thêm vào, nội suy, tức là suy diễn với các giá trị Age xung quanh,
Ở đây sử dụng hàm nội suy interpolate()
df['Age'] = df['Age'].interpolate()

✓
0 giây [13] #Giá trị nội suy
df['Age'].iloc[42]

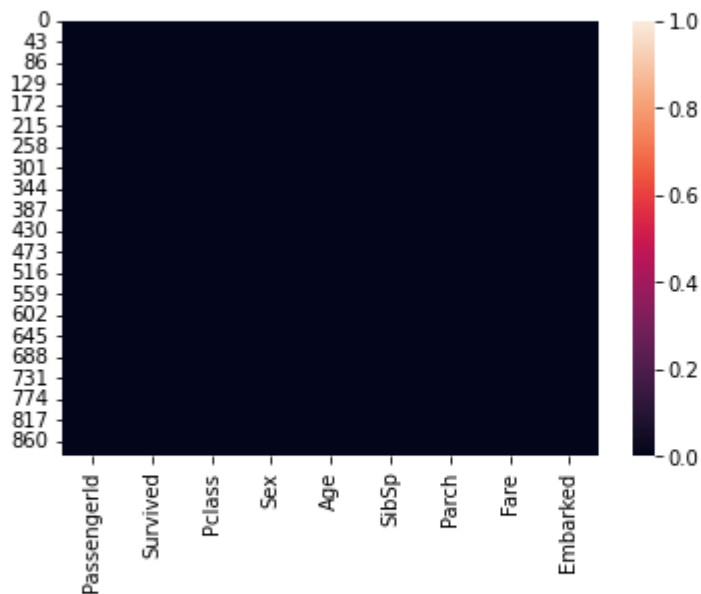
15.0



#Kiểm tra dữ liệu lần nữa. không thấy còn NaN
sb.heatmap(df.isnull())



<matplotlib.axes._subplots.AxesSubplot at 0x7f1f06f10190>



[14] #Xóa hàng - rows với các giá trị rỗng
df = df.dropna()

✓ [15] df.head()

0 giây

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S



#Đọc thông tin toàn bộ bảng dữ liệu, ta sẽ thấy kiểu dữ liệu của từng trường
Nếu trường kiểu số: int, float, double,... thì tính toán được
#Còn các trường kiểu string, object thì cần chuyển về kiểu số, ví dụ: Sex, Embarked
df.info()



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PassengerId  889 non-null   int64
1   Survived     889 non-null   int64
2   Pclass       889 non-null   int64
3   Sex          889 non-null   object
4   Age          889 non-null   float64
5   SibSp        889 non-null   int64
6   Parch        889 non-null   int64
7   Fare         889 non-null   float64
8   Embarked     889 non-null   object
dtypes: float64(2), int64(5), object(2)
memory usage: 69.5+ KB
```

✓
0
giây



df

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S
...
886	887	0	2	male	27.0	0	0	13.0000	S
887	888	1	1	female	19.0	0	0	30.0000	S
888	889	0	3	female	22.5	1	2	23.4500	S
889	890	1	1	male	26.0	0	0	30.0000	C
890	891	0	3	male	32.0	0	0	7.7500	Q

889 rows x 9 columns

✓
1
giây

```
[18] #Cột sex đang ở dạng Object (male/female) => cần chuyển về 0/1 mới phân hoạch được  
#Tương tự Embarked  
#Để thực hiện được việc đó ta tạo 1 cột giả với cột cần chuyển đổi, sau đó đưa vào bảng  
EmbarkedColumnDummy = pd.get_dummies(df['Embarked'])  
SexColumnDummy = pd.get_dummies(df['Sex'])
```

✓
0
giây


```
[19] #Bây giờ ta đã chuyển đổi sex - thành dạng 0/1, Embarked - đưa vào thành 001, 010, 101,... theo 3 cửa - C, Q, S  
df = pd.concat((df,EmbarkedColumnDummy,SexColumnDummy),axis=1)
```

✓
0
giây



#Kết quả sau khi phân tích dữ liệu, Ta cần xóa đi 2 cột Sex và Embarked cũ đi
df


	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	C	Q	S	female	male
0	1	0	3	male	22.0	1	0	7.2500	S	0	0	1	0	1
1	2	1	1	female	38.0	1	0	71.2833	C	1	0	0	1	0
2	3	1	3	female	26.0	0	0	7.9250	S	0	0	1	1	0
3	4	1	1	female	35.0	1	0	53.1000	S	0	0	1	1	0
4	5	0	3	male	35.0	0	0	8.0500	S	0	0	1	0	1
...
886	887	0	2	male	27.0	0	0	13.0000	S	0	0	1	0	1
887	888	1	1	female	19.0	0	0	30.0000	S	0	0	1	1	0
888	889	0	3	female	22.5	1	2	23.4500	S	0	0	1	1	0

✓ 0 giây  #Xóa các cột Embarked và Sex cũ dư thừa đi
df = df.drop(['Sex', 'Embarked'], axis=1)

✓ 0 giây [22] df

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	Q	S	female	male
0	1	0	3	22.0	1	0	7.2500	0	0	1	0	1
1	2	1	1	38.0	1	0	71.2833	1	0	0	1	0
2	3	1	3	26.0	0	0	7.9250	0	0	1	1	0
3	4	1	1	35.0	1	0	53.1000	0	0	1	1	0
4	5	0	3	35.0	0	0	8.0500	0	0	1	0	1
...
886	887	0	2	27.0	0	0	13.0000	0	0	1	0	1
887	888	1	1	19.0	0	0	30.0000	0	0	1	1	0
888	889	0	3	22.5	1	2	23.4500	0	0	1	1	0
889	890	1	1	26.0	0	0	30.0000	1	0	0	0	1
890	891	0	3	32.0	0	0	7.7500	0	1	0	0	1

889 rows x 12 columns

✓  #Chuẩn hóa (standard) dữ liệu trường tuổi và trường Fare theo kiểu min-max-scale,
#giá trị mới = (giá trị cũ - min)/(max - min) của cột đó
#Nếu chúng ta không chuẩn hóa dữ liệu thì chúng ta sẽ thấy 1 điều rằng cột nào có giá trị quá lớn so với các cột khác
#Trong bảng, thì nó sẽ chỉ phớt lờ hết toàn bộ kết quả của bảng đó. Như thế thì không đúng.
#Ví dụ: tính chỉ số con người có chiều cao 1.7, cân nặng 64kg, như thế chiều cao, cân nặng ở 2 thang đo khác nhau => không cùng hệ quy chiếu
#Cho nên chúng ta cần chuyển về cùng 1 hệ quy chiếu lúc đó mới so sánh được
#Age, Fare chuyển về cùng kiểu [0-1]
Age = df['Age']

[24] #Lọc cột tuổi ra
Age

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
...
886   27.0
887   19.0
888   22.5
889   26.0
890   32.0
```

Name: Age, Length: 889, dtype: float64

✓
1
giây

```
[30] #Sử dụng thư viện sklearn để chuyển hệ tham chiếu cho trường Age về từ 0-1
      from sklearn import preprocessing
      Age = Age.reshape(-1, 1) #returns a numpy array
      min_max_scaler = preprocessing.MinMaxScaler()
      Age_scaled = min_max_scaler.fit_transform(Age)
```

✓
0
giây

```
[31] #Tuổi sau khi scaled
      Age_scaled
```

```
[0.28373963],
[0.22090978],
[0.48479517],
[0.25860769],
[0.32772053],
[0.39683338],
[0.32143755],
[0.24604172],
[0.19577783],
[0.37170143],
[0.4282483 ],
[0.2083438 ],
[0.52249309],
[0.47851219],
[0.43453129],
[0.34656949],
[0.19577783],
[0.04498618],
[0.92460417],
[0.10781603]
```

✓ 0 giây hoàn thành lúc

✓
0
giây

```
[32] #Gán giá trị Age mới vào bảng
      df['Age'] = Age_scaled
```

✓
0
giây

```
[33] df
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	Q	S	female	male
0	1	0	3	0.271174	1	0	7.2500	0	0	1	0	1
1	2	1	1	0.472229	1	0	71.2833	1	0	0	1	0
2	3	1	3	0.321438	0	0	7.9250	0	0	1	1	0
3	4	1	1	0.434531	1	0	53.1000	0	0	1	1	0
4	5	0	3	0.434531	0	0	8.0500	0	0	1	0	1
...
886	887	0	2	0.334004	0	0	13.0000	0	0	1	0	1
887	888	1	1	0.233476	0	0	30.0000	0	0	1	1	0
888	889	0	3	0.277457	1	2	23.4500	0	0	1	1	0
889	890	1	1	0.321438	0	0	30.0000	1	0	0	0	1
890	891	0	3	0.396833	0	0	7.7500	0	1	0	0	1

✓
0 giây

```
[35] #Tương tự fare
Fare = df['Fare']
Fare = Fare.values.reshape(-1, 1) #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
Fare_scaled = min_max_scaler.fit_transform(Fare)
df['Fare'] = Fare_scaled
```

✓
0 giây

```
[36] df.head()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	Q	S	female	male
0	1	0	3	0.271174	1	0	0.014151	0	0	1	0	1
1	2	1	1	0.472229	1	0	0.139136	1	0	0	1	0
2	3	1	3	0.321438	0	0	0.015469	0	0	1	1	0
3	4	1	1	0.434531	1	0	0.103644	0	0	1	1	0
4	5	0	3	0.434531	0	0	0.015713	0	0	1	0	1

✓
0 giây

```
[37] #Phân dữ liệu vào trục X và trục Y, mảng
#X là bộ dữ liệu (các cột thuộc tính huấn luyện) - các thuộc tính gây ra ảnh hưởng tới survived - Tập vào
#Y là kết quả phân hoạch - Được cứu/ không được cứu (Survived) - Tập đích
# Cho nên đầu tiên chúng ta chuyển dữ liệu vào X
X = df.values
#Chuyển dữ liệu đích vào Y
Y = df['Survived'].values
```

✓
0 giây

```
[38] X
```

```
array([[ 1.,  0.,  3., ...,  1.,  0.,  1.],
       [ 2.,  1.,  1., ...,  0.,  1.,  0.],
       [ 3.,  1.,  3., ...,  1.,  1.,  0.],
       ...,
       [889.,  0.,  3., ...,  1.,  1.,  0.],
       [890.,  1.,  1., ...,  0.,  0.,  1.],
       [891.,  0.,  3., ...,  0.,  0.,  1.]])
```

✓
0 giây

```
[39] Y
```

```
array([0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
```



```
✓ [40] #Xóa cột Survived từ X (Tại vì X không có cột Survived)
0 giây
# 0 - PassengerID, 1-Survived, ....
X = np.delete(X, 1, axis =1)
```

```
✓ [41] X
0 giây
array([[1.00000000e+00, 3.00000000e+00, 2.71173662e-01, ...,
        1.00000000e+00, 0.00000000e+00, 1.00000000e+00],
       [2.00000000e+00, 1.00000000e+00, 4.72229203e-01, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [3.00000000e+00, 3.00000000e+00, 3.21437547e-01, ...,
        1.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       ...,
       [8.89000000e+02, 3.00000000e+00, 2.77456647e-01, ...,
        1.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [8.90000000e+02, 1.00000000e+00, 3.21437547e-01, ...,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00],
       [8.91000000e+02, 3.00000000e+00, 3.96833375e-01, ...,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

```
✓ [42] #Phân tách bộ dữ liệu
0 giây
#Phân dữ liệu thành 70% train, 30% test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,random_state=0)
```

```
✓ [43] #1. Xây dựng phân lớp cây quyết định (decision tree classifier)
0 giây
from sklearn import tree
dt_clf = tree.DecisionTreeClassifier(max_depth=5) #Xây dựng mô hình
dt_clf.fit(x_train,y_train) #Huấn luyện
dt_clf.score(x_test,y_test) #Đưa ra dự đoán

y_pred = dt_clf.predict(x_test)
#Độ chính xác của mô hình dự đoán
dt_clf.score(x_test,y_test)
```

0.7827715355805244

```
✓ [44] #dự đoán
0 giây
y_pred
array([1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1])
```

✓
0 giây
[45] y_pred = dt_clf.predict(x_test)
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)

```
array([[136, 21],  
       [ 37, 73]])
```

✓
0 giây
[47] #Như vậy với bộ dữ liệu kiểm tra được đặt trong 1 file CSV khác có tên titanic_test.csv
#PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked
892 ??? 2 Tùng Dương male 26 1 0 113812 53.1000 C123 S
Hỏi dự đoán Survived???
Các bước làm tương tự
df1 = pd.read_csv("/content/drive/MyDrive/Data Analys/titanic_test.csv")

✓
0 giây
[48] df1

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	?	1	S?n la	male	29	1	1	112053	30.00	B42	S
1	893	?	2	Mai Hoa	female	27	1	2	W./C. 6607	33.00	B12	S
2	894	?	2	Bac Son	male	26	0	0	111369	7.00	C148	C
3	895	?	3	Nam dong	female	16	1	1	370376	7.75	C18	Q
4	896	?	2	Tung duong	male	26	1	0	113812	53.00	C123	S

✓
0 giây
[49] #Xóa một số cột không được sử dụng trong phân lớp
#name, ticket, cabin
cols_to_drop = ['Name','Ticket','Cabin']
df1 = df1.drop(cols_to_drop,axis=1)

✓
0 giây
[50] #Cột sex đang ở dạng Object (male/female) => cần chuyển về 0/1 mới phân hoạch được
#Tương tự Embarked
#Để thực hiện được việc đó ta tạo 1 cột giả với cột cần chuyển đổi, sau đó đưa vào bảng
EmbarkedColumnDummy = pd.get_dummies(df1['Embarked'])
SexColumnDummy = pd.get_dummies(df1['Sex'])

✓
0 giây
[51] df1 = pd.concat((df1,EmbarkedColumnDummy,SexColumnDummy),axis=1)

✓
0 giây
[52] df1

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	C	Q	S	female	male
0	892	?	1	male	29	1	1	30.00	S	0	0	1	0	1
1	893	?	2	female	27	1	2	33.00	S	0	0	1	1	0
2	894	?	2	male	26	0	0	7.00	C	1	0	0	0	1
3	895	?	3	female	16	1	1	7.75	Q	0	1	0	1	0
4	896	?	2	male	26	1	0	53.00	S	0	0	1	0	1

✓
0
giây

```
[53] #Xóa các cột Embarked và Sex cũ dư thừa đi  
df1 = df1.drop(['Sex', 'Embarked'], axis=1)
```

✓
0
giây



```
#Chuẩn hóa Age  
Age = df1['Age']  
Age = Age.values.reshape(-1, 1) #returns a numpy array  
min_max_scaler = preprocessing.MinMaxScaler()  
Age_scaled = min_max_scaler.fit_transform(Age)  
df1['Age'] = Age_scaled
```

✓
0
giây

```
[56] #Tương tự fare  
Fare = df1['Fare']  
Fare = Fare.values.reshape(-1, 1) #returns a numpy array  
min_max_scaler = preprocessing.MinMaxScaler()  
Fare_scaled = min_max_scaler.fit_transform(Fare)  
df1['Fare'] = Fare_scaled
```

```
[ ] #Test cho từng bộ, ví dụ ID 892
```

```
x892 = df1[0:1].values
```

```
[ ] #Test cho từng bộ, ví dụ ID 892
```

```
x892 = df1[0:1].values
```

```
[ ] #Test cho từng bộ, ví dụ ID 892
```

```
x893 = df1[2:3].values
```

```
[ ] x892
```

```
array([[892, '?', 1, 1.0, 1, 1, 0.5, 0, 0, 1, 0, 1]], dtype=object)
```

```
[ ] #Xóa cột Survived từ X (Tại vì X không có cột Survived)
```

```
# 0 - PassengerID, 1-Survived, ....
```

```
x892 = np.delete(x892, 1, axis =1)
```

```
[ ] x892
```

```
array([[892, 1, 1.0, 1, 1, 0.5, 0, 0, 1, 0, 1]], dtype=object)
```

```
[ ] y892_pred = dt_clf.predict(x892)
```

```
[ ] #Dự đoán  
y892_pred
```

```
array([1])
```

```
[ ] #Độ chính xác  
dt_clf.score(x892,y892_pred)
```


```
1.0
```

```
[ ] #Phân lớp với Random forest - Tương tự cây quyết định  
#Mô hình học có giám sát  
from sklearn import ensemble  
rf_clf = ensemble.RandomForestClassifier(n_estimators=100)  
rf_clf.fit(x_train,y_train)  
rf_clf.score(x_test,y_test)
```

```
0.7940074906367042
```

```
[ ] y_pred = rf_clf.predict(x_test)  
#Độ chính xác của mô hình dự đoán  
rf_clf.score(x_test,y_test)
```


```
0.7940074906367042
```

 y_pred

```
array([1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1])
```

```
[ ] #Naive Bayes Classifier
    from sklearn.naive_bayes import GaussianNB
    nb_clf = GaussianNB()
    nb_clf.fit(x_train,y_train)
    y_pred = nb_clf.predict(x_test)
    #Độ chính xác của mô hình dự đoán
    nb_clf.score(x_test,y_test)
```

0.7602996254681648

 y_pred

```
array([1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1])
```



```
#K-Nearest Neighbor Classifier
from sklearn.neighbors import KNeighborsClassifier
knn_clf = KNeighborsClassifier(n_neighbors=3)
knn_clf.fit(x_train,y_train)
y_pred = knn_clf.predict(x_test)
#Độ chính xác của mô hình dự đoán
knn_clf.score(x_test,y_test)
```

```
0.5430711610486891
```

```
[ ] y_pred
```

```
array([1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 1])
```

```
[ ] #Phân lớp hồi quy tuyến tính
from sklearn.linear_model import LogisticRegression
lr_clf = LogisticRegression()
lr_clf.fit(x_train,y_train)
y_pred = lr_clf.predict(x_test)
#Độ chính xác của mô hình dự đoán
lr_clf.score(x_test,y_test)
```

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
0.7528089887640449



```
#Phân lớp SVM
from sklearn.svm import SVC
sv_clf = SVC(probability=True, kernel='linear')
sv_clf.fit(x_train, y_train)
y_pred = sv_clf.predict(x_test)
#Độ chính xác của mô hình dự đoán
sv_clf.score(x_test, y_test)
```

0.7715355805243446

[]