

# kaggle 데이터를 활용한 데이터 분석

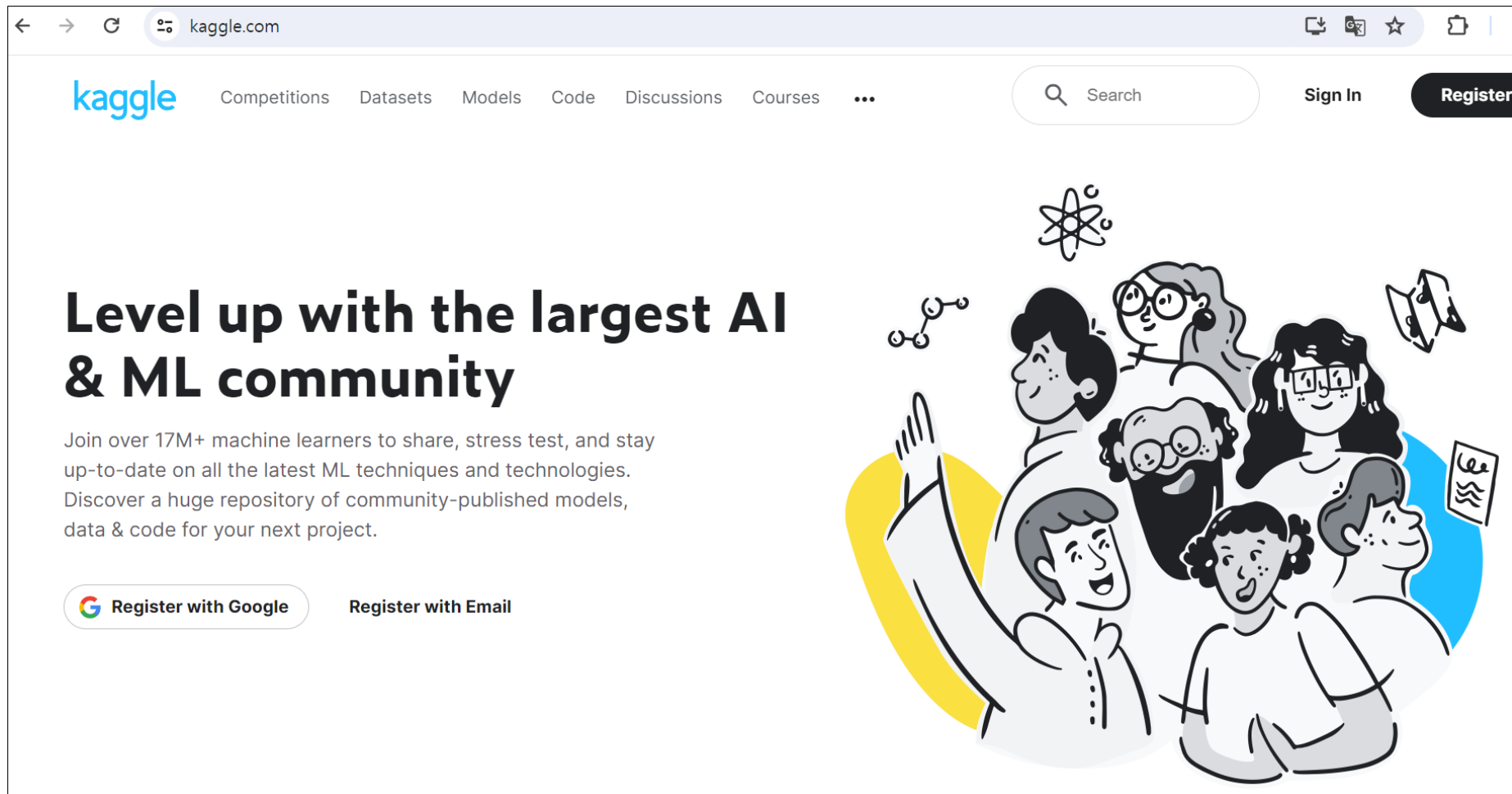


A top-down view of a person's hand typing on a silver laptop keyboard. The laptop is on a wooden desk. In the background, there is a white keyboard and a white mouse. A person's wrist with a black watch is visible in the bottom left corner. The word "Kaggle" is overlaid in large white letters on a semi-transparent blue and yellow rectangular background.

# Kaggle

- 데이터 분석 및 머신러닝에 대한 학습 플랫폼이자, 경쟁할 수 있는 플랫폼이다.
- 기업, 기관 또는 특정 사용자가 데이터를 첨부해서 문제를 제출하면 Kaggle 사용자 누구나 문제에 대한 답을 제출할 수 있다.

■ <https://www.kaggle.com/>



## ■ Competitions

- 기업, 기관 혹은 특정 사용자가 낸 문제들을 볼 수 있다.

### Competitions

Grow your data science skills by competing in our exciting competitions. Find help in the [documentation](#) or learn about [Community Competitions](#).

[Host a Competition](#)

🔍 Search competitions

**All Competitions** ☰  
Everything, past & present


**Featured** ☆  
Premier challenges with prizes

**Getting Started** 📖  
Approachable ML fundamentals


**Research** 🔬  
Scientific and scholarly challenges

**Community**  
Created by fellow Kagglers


📖 **Getting Started**  
Competitions with approachable ML fundamentals.



Titanic - Machine Learning from Disaster



Housing Prices Competition for Kaggle...



House Prices - Advanced Regression Techniques

- **Kaggle에서 얻을 수 있는 것**
  - 다양한 데이터 분석 / 예측 경험
  - 다양한 Competition을 통한 Prize (상금, 명예, 실력)
  - 전 세계의 Data Scientist들과의 소통
  - 우수한 데이터 분석 / 예측 과정 (솔루션)

## ■ 캐글 타이타닉 생존자 예측

The screenshot shows the Kaggle search results for the query 'titanic'. The browser address bar shows 'kaggle.com/search?q=titanic'. The search bar contains 'titanic'. Below the search bar, there are filters for Comments (91,092), Notebooks (77,093), Topics (6,375), Datasets (2,452), and Competitions (289). The results are filtered by 'DATE' and 'CREATOR'. The 'DATE' filter shows 'Last 90 days' (3,771 results), 'This week' (211 results), and 'Today' (22 results). The 'CREATOR' filter shows 'You' (0 results) and 'Others' (177,406 results). The results list includes:

- Titanic Tutorial**  
Notebook · 3y ago · by Alexis Cook  
Change it to something more descriptive, like \*\*\*Getting Started with Titanic\*\*\*. !
- Exploring Survival on the Titanic**  
Notebook · 7y ago · by Meg Risdal  
--- title: 'Exploring the Titanic Dataset' author: 'Megan L.'
- Welcome to the Spaceship Titanic!**  
Discussion Topic · 3y ago · by Ryan Holbrook  
Welcome to the Spaceship Titanic!

## ■ Overview

### Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#)

#### Overview

∞ This competition runs indefinitely with a rolling leaderboard. [Learn more.](#)

#### Description



**Ahoy, welcome to Kaggle! You're in the right place.**

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

**If you want to talk with other users about this competition, come join our Discord! We've got channels for**



## ■ Data

- Data에 관한 간략한 설명 및 .csv 형태로 데이터를 제공하고 있다.

[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#)

---

### Dataset Description

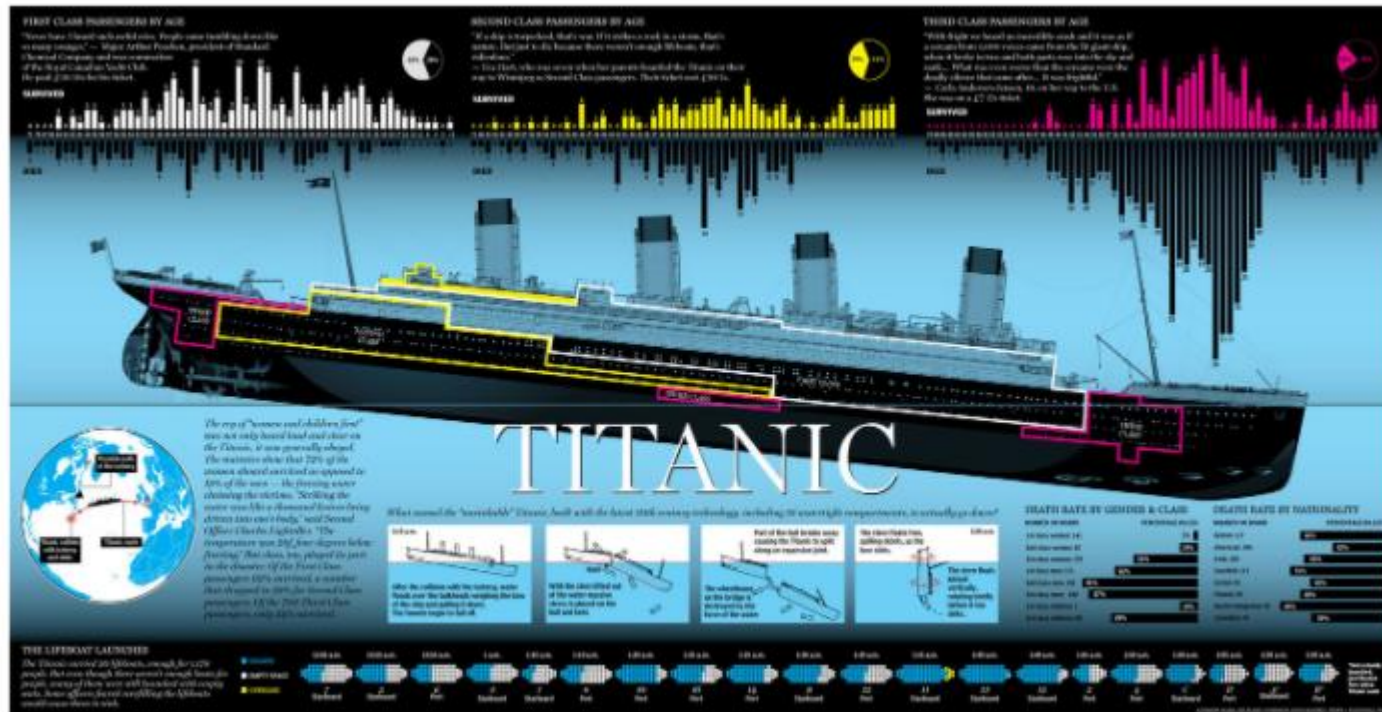
#### Overview

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

# Titanic – 문제 정의하기

- 타이타닉 호에서 탑승했던 사람들의 정보를 바탕으로 생존자를 예측하는 문제
- 타이타닉은 사상 최대 해난사고로써, 1,500여명의 희생자가 생겼다.



# Titanic – 데이터 불러오기

```
import pandas as pd
import numpy as np

train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

```
train.head()
```

	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

## ■ Data Dictionary

### ■ 승객 데이터에서 제공되는 특성

Survived	생존여부	target label 임. 1, 0 으로 표현됨	integer
Pclass	티켓의 클래스	1 = 1st, 2 = 2nd, 3 = 3rd 클래스로 나뉘며 categorical feature	integer
Sex	성별	male, female 로 구분되며 binary	string
Age	나이	continuous	integer
SibSp	함께 탑승한 형제와 배우자의 수	quantitative	integer
Parch	함께 탑승한 부모, 아이의 수	quantitative	integer
Ticket	티켓 번호	alphanum + integer	string
Fare	탑승료	continuous	float
Cabin	객실 번호	alphanum + integer	string
Embarked	탑승 항구	C = Cherbourg, Q = Queenstown, S = Southampton	string

# Titanic - 데이터 분석

```
print('train data shape: ', train.shape)
print('test data shape: ', test.shape)
print('-----[train infomation]-----')
print(train.info())
print('-----[test infomation]-----')
print(test.info())
```

```
train data shape: (891, 12)
test data shape: (418, 11)
-----[train infomation]-----
<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
```

```
-----[test infomation]-----
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId     418 non-null   int64
1   Pclass          418 non-null   int64
2   Name            418 non-null   object
3   Sex             418 non-null   object
4   Age             332 non-null   float64
5   SibSp           418 non-null   int64
6   Parch           418 non-null   int64
7   Ticket          418 non-null   object
8   Fare            417 non-null   float64
9   Cabin           91 non-null    object
10  Embarked        418 non-null   object
dtypes: float64(2), int64(4), object(5)
```

# Titanic – 데이터 분석

## ■ describe()

- 각 feature 가 가진 통계 치들을 반환해 준다.
- PassenserID 숫자와 다른, null data가 존재하는 열(feature)가 있는 것 같다.

```
train.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

# Titanic – 데이터 분석

```
test.describe()
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

## ■ Null data check

```
for col in train.columns:
    msg = 'column: {:>10}#t Percent of NaN value: {:.2f}%'.format(col, 100 * (train[col].isnull().sum() / train[col].shape[0]))
    print(msg)
```

column: PassengerId	Percent of NaN value: 0.00%
column: Survived	Percent of NaN value: 0.00%
column: Pclass	Percent of NaN value: 0.00%
column: Name	Percent of NaN value: 0.00%
column: Sex	Percent of NaN value: 0.00%
column: Age	Percent of NaN value: 19.87%
column: SibSp	Percent of NaN value: 0.00%
column: Parch	Percent of NaN value: 0.00%
column: Ticket	Percent of NaN value: 0.00%
column: Fare	Percent of NaN value: 0.00%
column: Cabin	Percent of NaN value: 77.10%
column: Embarked	Percent of NaN value: 0.22%



## ■ Null data check

```
for col in test.columns:  
    msg = 'column: {:>10} Percent of NaN value: {:.2f}%'.format(col, 100 * (test[col].isnull().sum() / test[col].shape[0]))  
    print(msg)
```

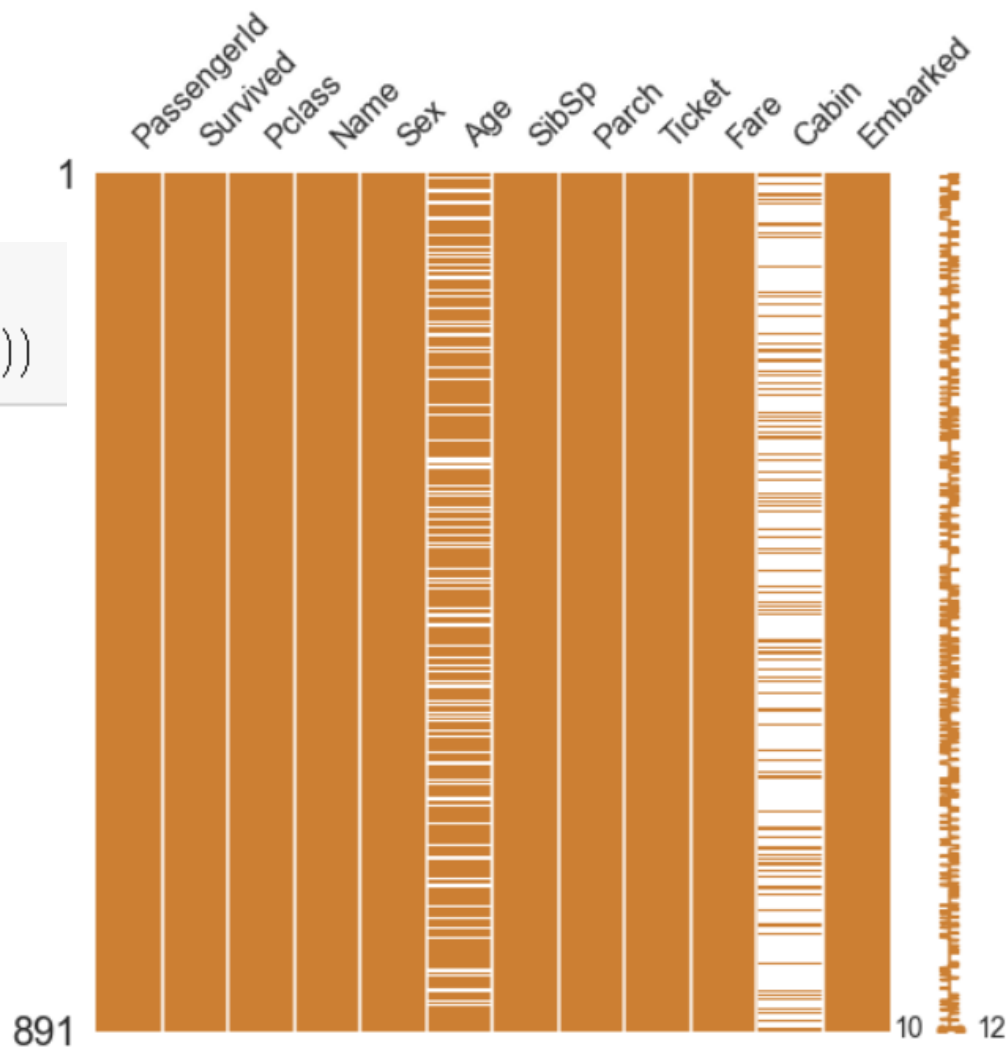
column: PassengerId	Percent of NaN value: 0.00%
column: Pclass	Percent of NaN value: 0.00%
column: Name	Percent of NaN value: 0.00%
column: Sex	Percent of NaN value: 0.00%
column: Age	Percent of NaN value: 20.57%
column: SibSp	Percent of NaN value: 0.00%
column: Parch	Percent of NaN value: 0.00%
column: Ticket	Percent of NaN value: 0.00%
column: Fare	Percent of NaN value: 0.24%
column: Cabin	Percent of NaN value: 78.23%
column: Embarked	Percent of NaN value: 0.00%

- Age[약 20%], Cabin[약 80%], Embarked(Train만 0.22%) null data 존재하는 것을 볼 수 있다.

# Titanic – 데이터 분석

- missingno 라는 라이브러리를 사용하면 null data의 존재를 더 쉽게 볼 수 있다.
- `import missingno as msno`

```
import missingno as msno #결측치에 대해 시각적으로 보고 싶을 때  
msno.matrix(df=train.iloc[:, :], figsize=(8, 8), color=(0.8, 0.5, 0.2))
```



## ■ Target label 확인

- target label 이 어떤 distribution 을 가지고 있는 지 확인해봐야 한다.
- binary classification 문제의 경우에서, 1과 0의 분포가 어떠냐에 따라 모델의 평가 방법이 달라 질 수 있다.

```
#ignore warnings
import warnings
warnings.filterwarnings('ignore')

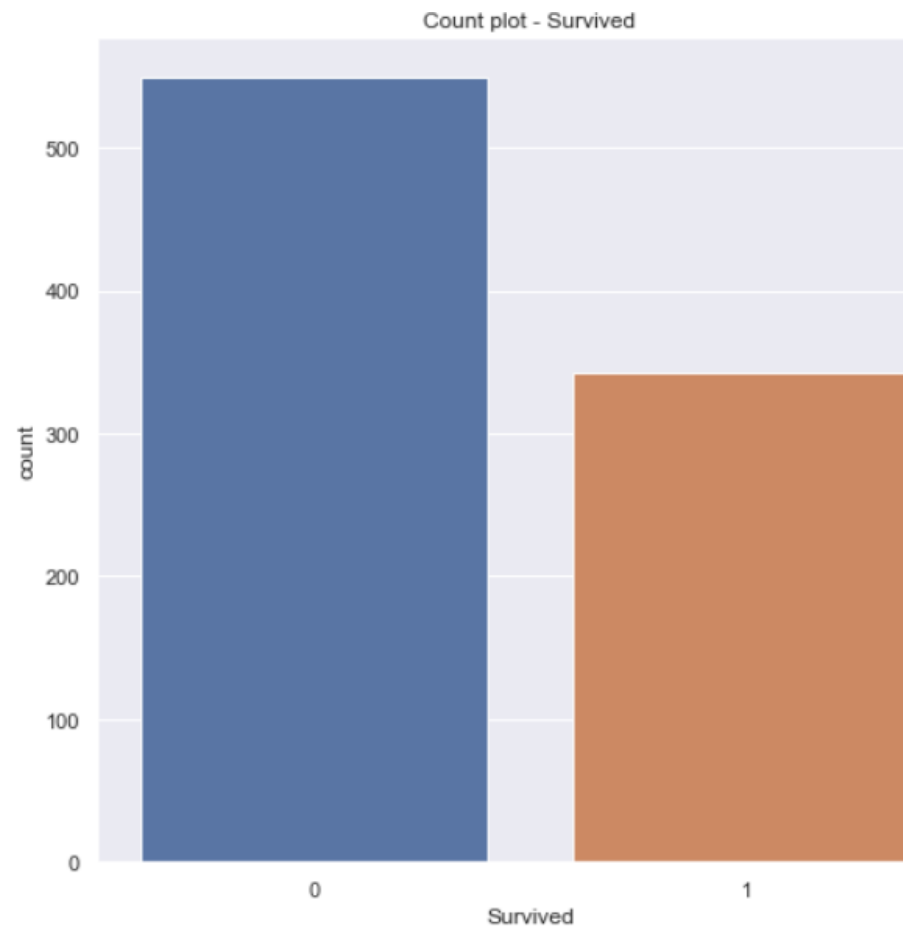
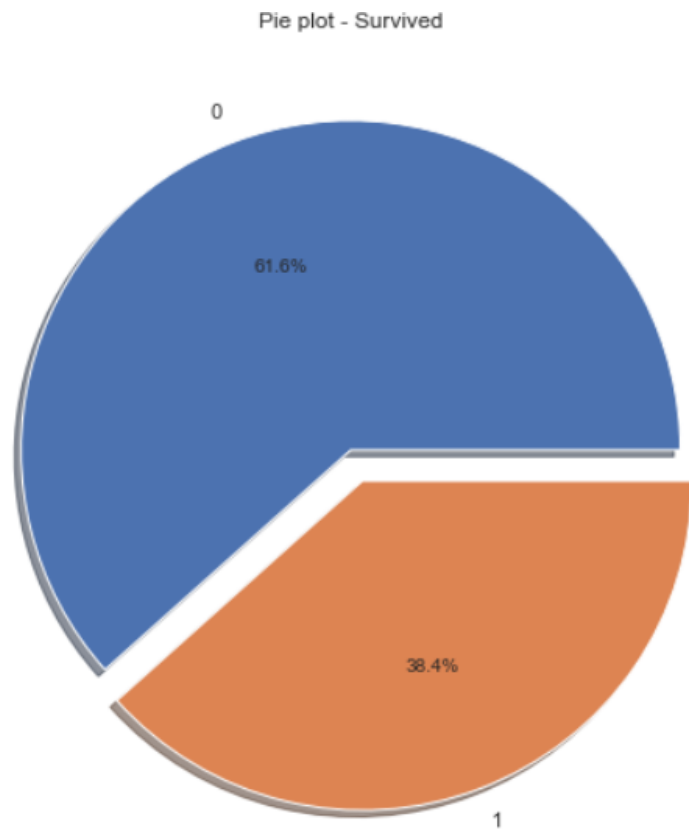
f, ax = plt.subplots(1, 2, figsize=(18, 8))

train['Survived'].value_counts().plot.pie(explode=[0, 0.1], autopct='%1.1f%%', ax=ax[0], shadow=True)
ax[0].set_title('Pie plot - Survived')
ax[0].set_ylabel('')
sns.countplot('Survived', data=train, ax=ax[1])
ax[1].set_title('Count plot - Survived')

plt.show()
```

# Titanic - 데이터 분석

## ■ 38.4% 생존



## ■ Categorical Feature의 분포를 보기 위해서 Pie chart를 만드는 함수 정의

```
import matplotlib.pyplot as plt
```

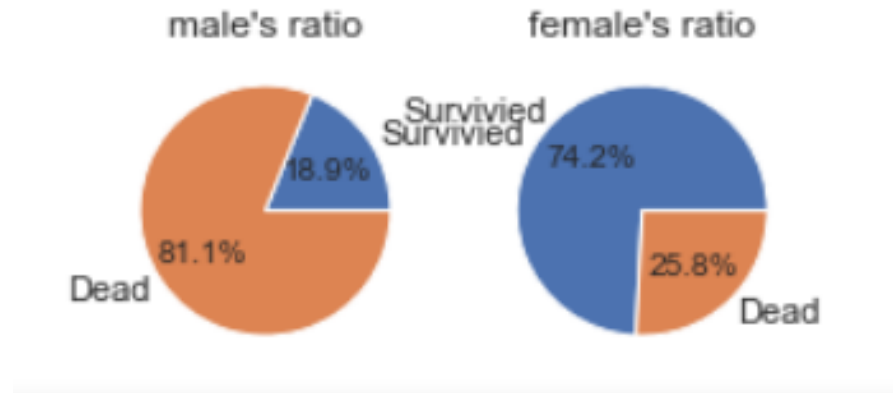
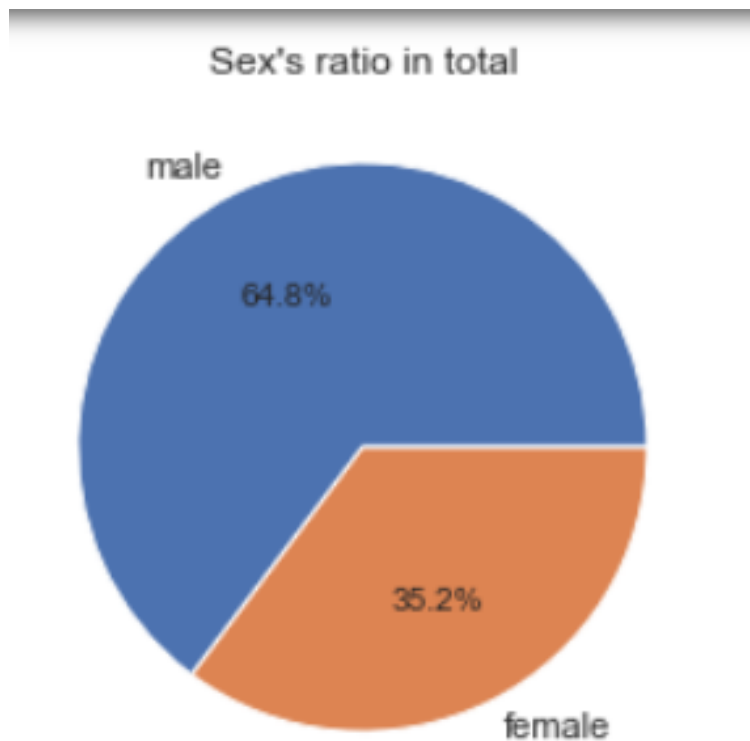
```
def pie_chart(feature):  
    feature_ratio = train[feature].value_counts()  
    feature_size = feature_ratio.size  
    feature_index = feature_ratio.index  
    survived = train[train['Survived'] == 1][feature].value_counts()  
    dead = train[train['Survived'] == 0][feature].value_counts()  
  
    plt.plot(aspect='auto')  
    plt.pie(feature_ratio, labels=feature_index, autopct='%1.1f%%')  
    plt.title(feature + '의 ratio in total')  
    plt.show()  
  
    for i, index in enumerate(feature_index):  
        plt.subplot(1, feature_size + 1, i + 1, aspect='equal')  
        plt.pie([survived[index], dead[index]], labels=['Survived', 'Dead'], autopct='%1.1f%%')  
        plt.title(str(index) + '의 ratio')  
  
    plt.show()
```

## ■ Pie chart for Categorical Feature

```
def pie_chart(feature):  
    feature_ratio = train[feature].value_counts()  
    feature_size = feature_ratio.size  
    feature_index = feature_ratio.index  
    survived = train[train['Survived'] == 1][feature].value_counts()  
    dead = train[train['Survived'] == 0][feature].value_counts()  
  
    plt.plot(aspect='auto')  
    plt.pie(feature_ratio, labels=feature_index, autopct='%1.1f%%')  
    plt.title(feature + '의 ratio in total')  
    plt.show()  
  
    for i, index in enumerate(feature_index):  
        plt.subplot(1, feature_size + 1, i + 1, aspect='equal')  
        plt.pie([survived[index], dead[index]], labels=['Survived', 'Dead'], autopct='%1.1f%%')  
        plt.title(str(index) + '의 ratio')  
  
    plt.show()
```

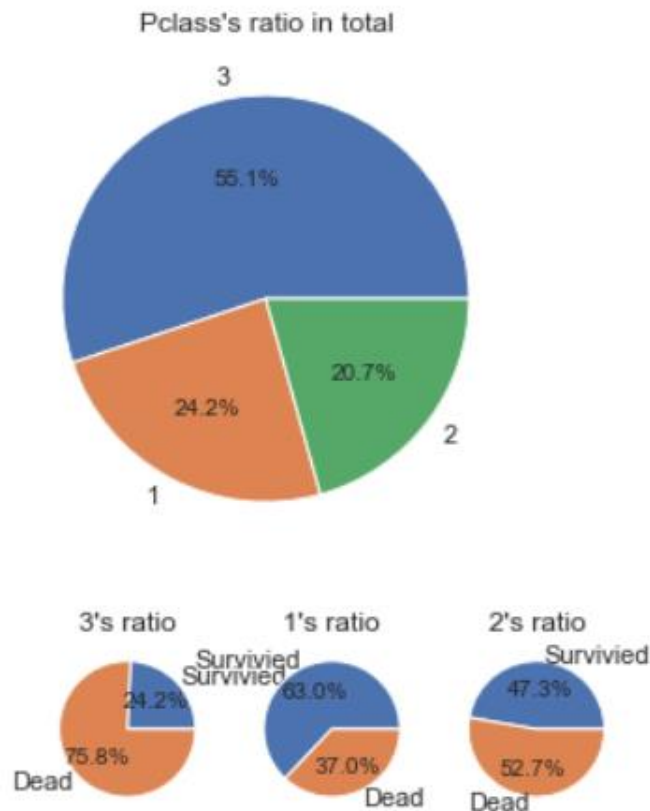
# Titanic – 데이터 분석

- 특성들이 생존에 미치는 영향에 대해서 생각해보자.
- 남성이 여성보다 배에 많이 탔으며, 남성보다 여성의 생존 비율이 높다는 것을 알 수가 있다.



## ■ 사회경제적 지위인 Pclass

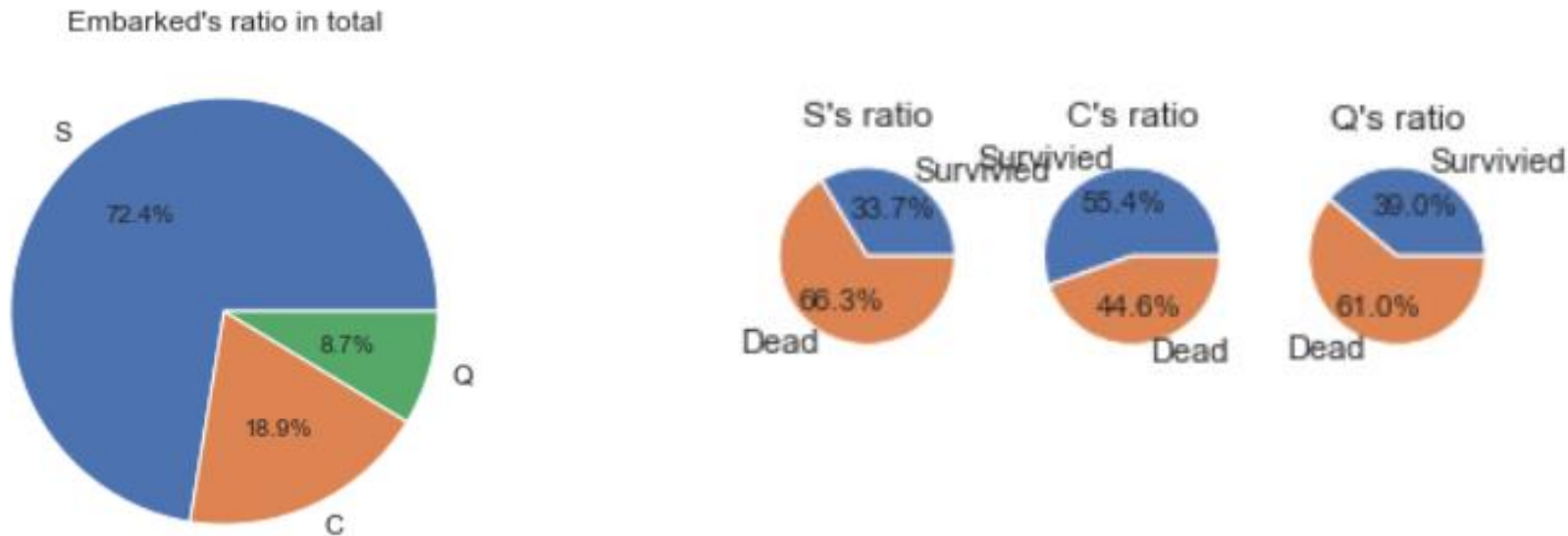
- Pclass가 3인 사람들의 수가 가장 많았으며, Pclass가 높을수록(숫자가 작을수록; 사회경제적 지위가 높을수록) 생존 비율이 높다는 것을 알 수 있다.





# Titanic – 데이터 분석

- 어느 곳에서 배를 탔는지를 나타내는 Embarked
  - Southampton에서 선착한 사람이 가장 많았으며, Cherbourg에서 탄 사람 중에서는 생존한 사람의 비율이 높았고, 나머지 두 선착장에서 탄 사람들은 생존한 사람보다 그렇지 못한 사람이 조금 더 많았다.



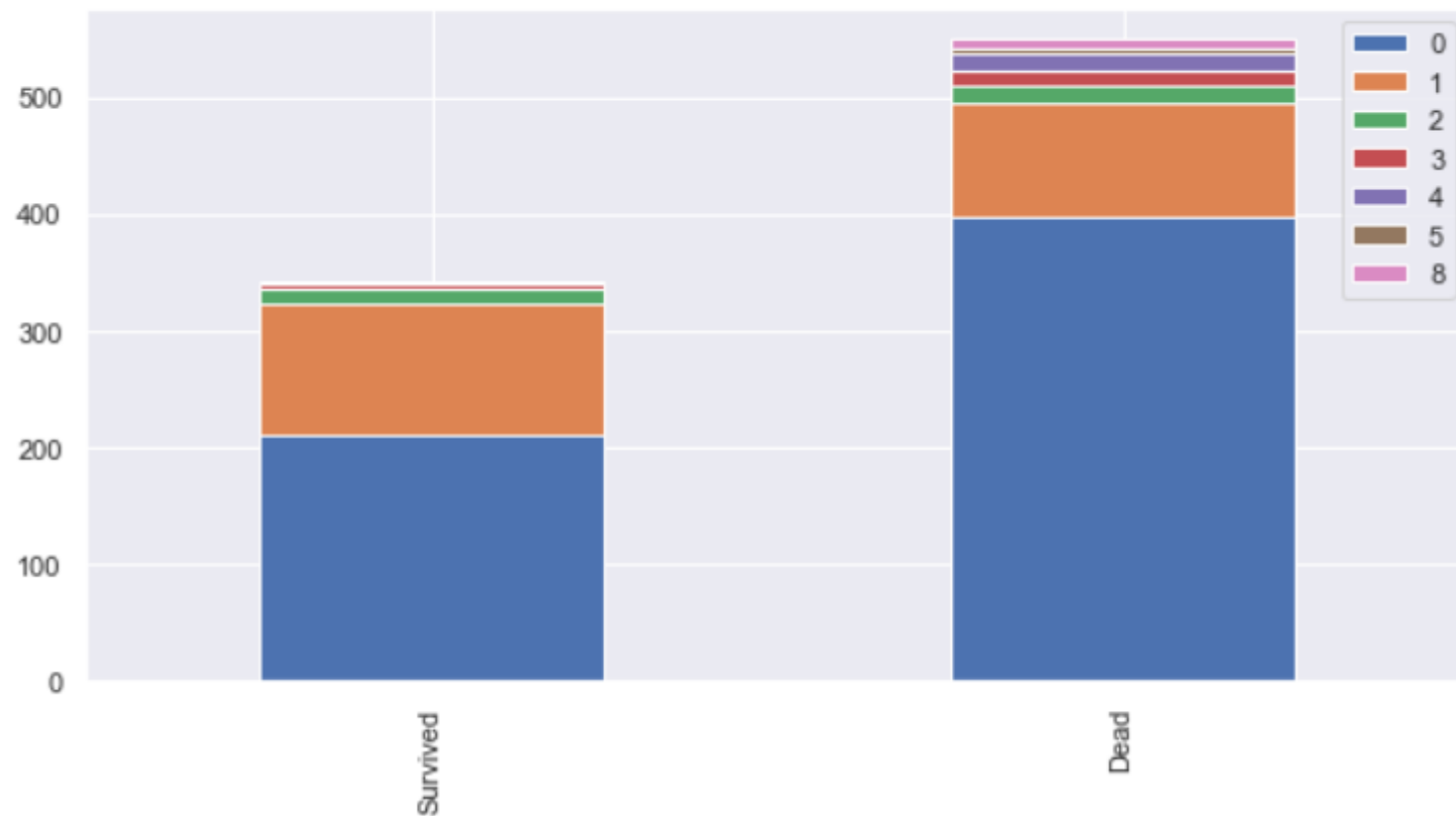
## ■ Bar chart for Categorical feature

- 2명 이상의 형제나 배우자와 함께 배에 탔을 경우 생존한 사람의 비율이 컸다는 것을 볼 수 있고, 그렇지 않을 경우에는 생존한 사람의 비율이 적었다는 것을 볼 수 있다.

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

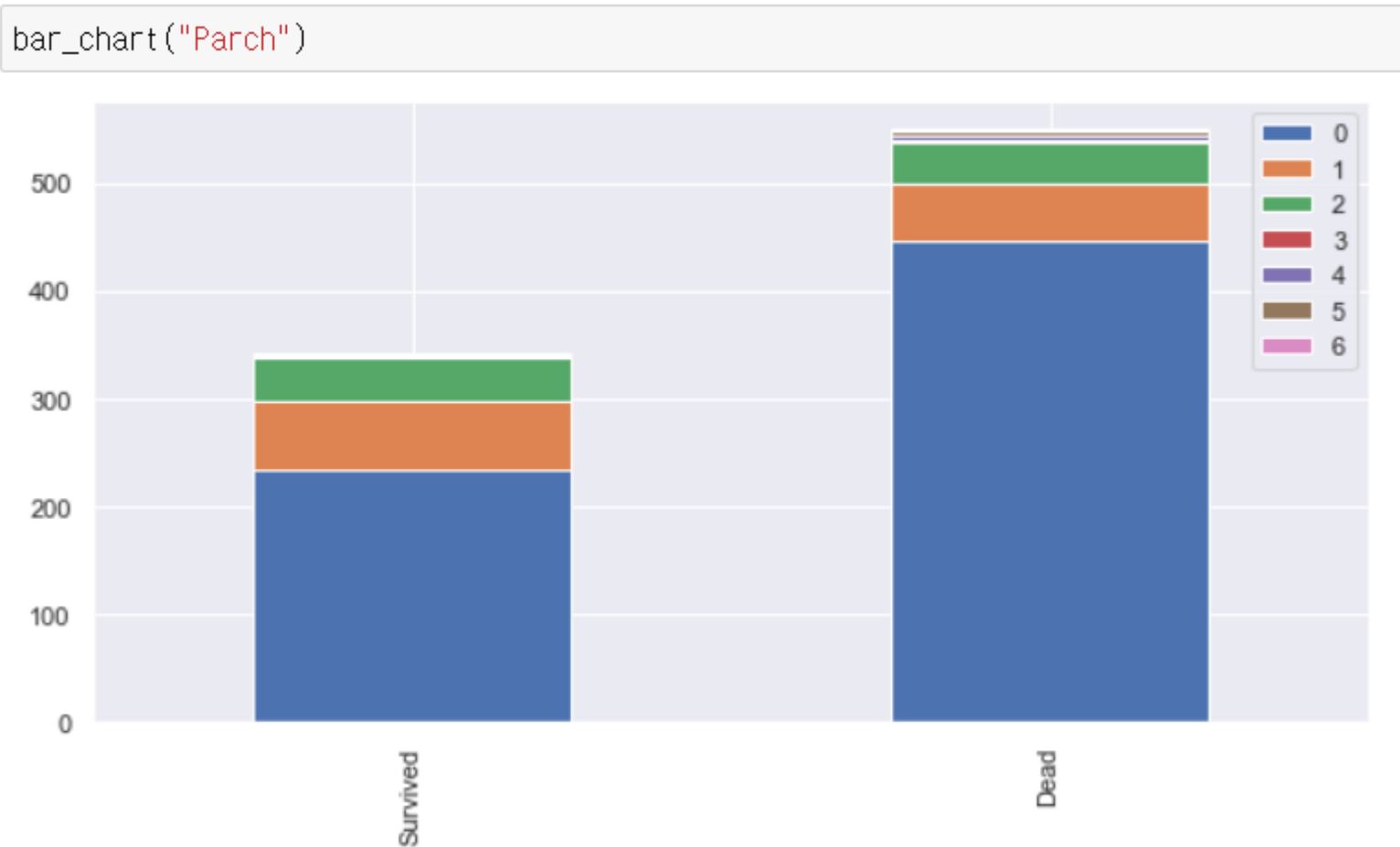
# Titanic – 데이터 분석

```
bar_chart("SibSp")
```



# Titanic – 데이터 분석

- SibSp와 비슷하게 2명 이상의 부모나 자식과 함께 배에 탔을 때는 조금 더 생존했지만, 그렇지 않을 경우에는 생존한 사람의 비율이 적었다.



# Titanic – 데이터 분석

- 성별이 여성일 수록, Pclass가 높을 수록, Cherbourg 선착장에서 배를 탔다면, 형제, 자매, 배우자, 부모, 자녀와 함께 배에 탔다면, 생존 확률이 더 높았다는 것을 볼 수 있다.
- 하지만 하나의 특성과 생존 비율 만을 생각해서 예측하기에는 무리가 있다.
- 예를 들어 높은 금액의 티켓을 산 부유한 사람이 가족들이랑 왔을 경우가 많다고 가정해본다면, 가족들과 함께 왔다고 해서 살 가능성이 높다고 할 수는 없으므로 단일 특성을 가지고 생존 확률을 예측하기 보단 여러 가지 특성을 종합해서 예측을 하는 것이 더 좋을 것이다.

## ■ Pclass

- Pclass 는 ordinal, 서수형 데이터이다. 카테고리이면서, 순서가 있는 데이터 타입이다.
- Pclass 에 따른 생존률의 차이를 살펴보자.
  - 엑셀의 피벗 차트와 유사한 작업을 하게 되는데, pandas dataframe 에서는 groupby 를 사용하면 쉽게 할 수 있다. 또한 pivot 이라는 메소드도 있다.
- 'Pclass', 'Survived' 를 가져온 후, pclass 로 묶는다.
- 각 pclass 마다 0, 1 이 count가 되는데, 이를 평균내면 각 pclass 별 생존률이 나온다.
- count() 를 하면, 각 class 에 몇 명이 있는 지 확인할 수 있으며, sum() 을 하면, 216 명중 생존한(survived=1)사람의 총합을 주게 된다.

# Titanic – 데이터 분석

```
train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=True).count()
```

Survived	
Pclass	
1	216
2	184
3	491

```
train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=True).sum()
```

Survived	
Pclass	
1	136
2	87
3	119

# Titanic – 데이터 분석

- pandas 의 crosstab 을 사용하면 위 과정을 좀 더 수월하게 볼 수 있다.
- grouped 객체에 mean() 을 하게 되면, 각 클래스별 생존률을 얻을 수 있다. class 1 이면 아래와 같다.

$$\frac{80}{(80 + 136)} \approx 0.63$$

```
pd.crosstab(train['Pclass'], train['Survived'], margins=True).style.background_gradient(cmap='summer_r')
```

Survived	0	1	All
Pclass			
1	80	136	216
2	97	87	184
3	372	119	491
All	549	342	891

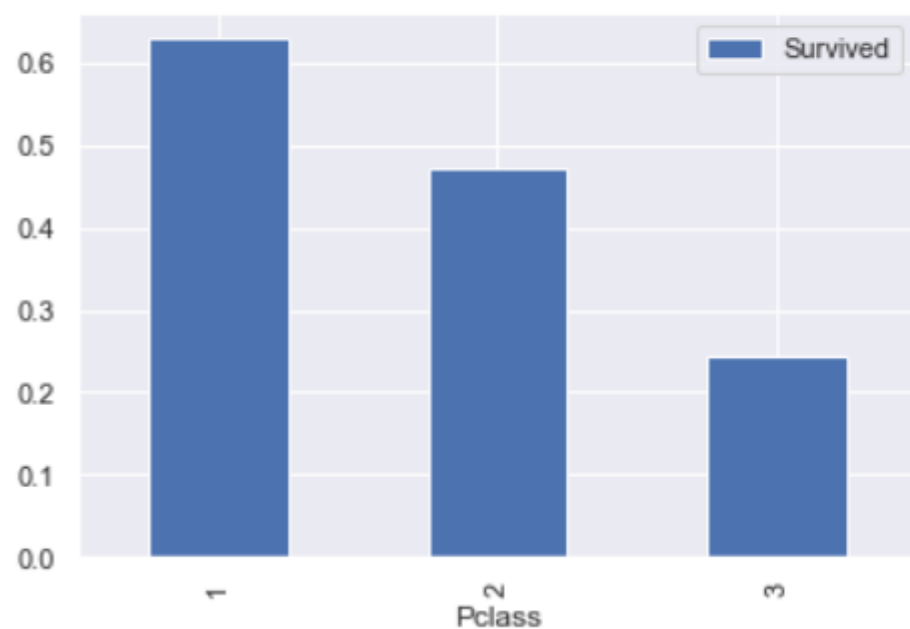


# Titanic - 데이터 분석

- Pclass 가 좋을 수록(1st) 생존률이 높은 것을 확인할 수 있다.

```
train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False).plot.bar()
```

<AxesSubplot: xlabel='Pclass'>



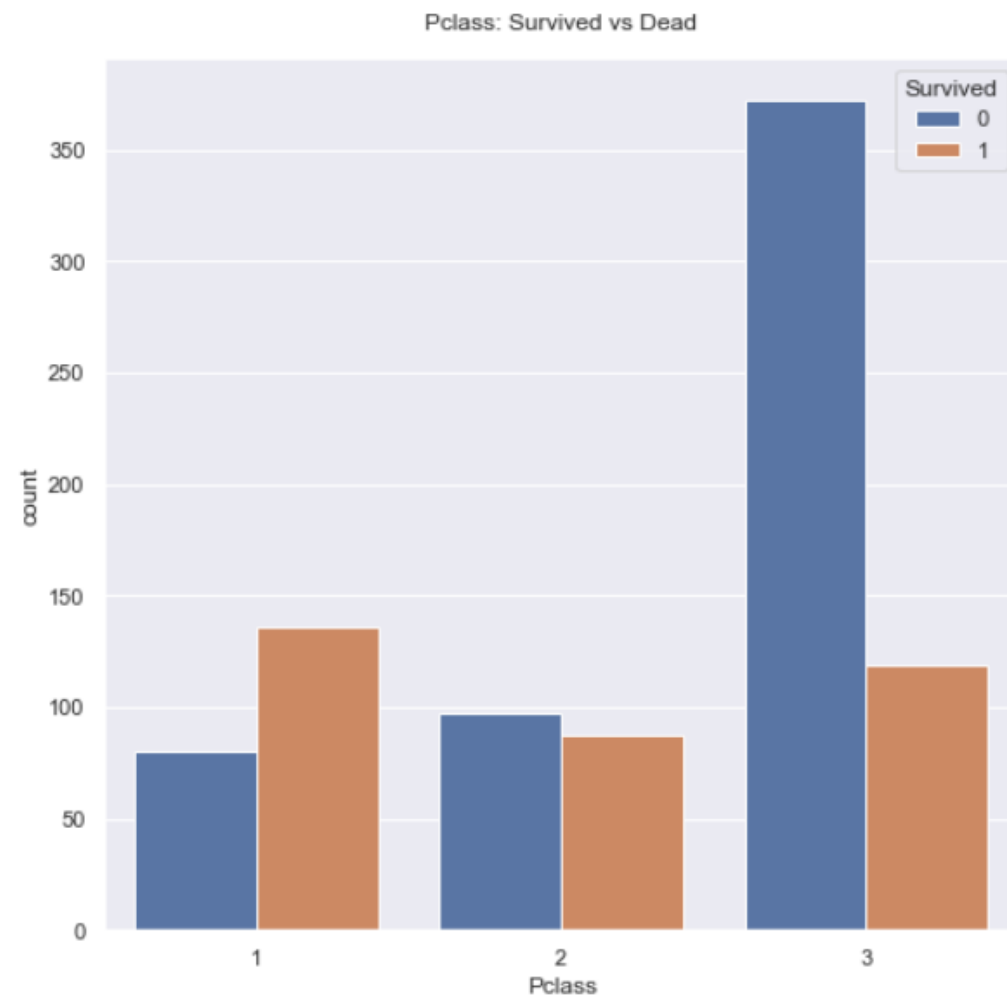
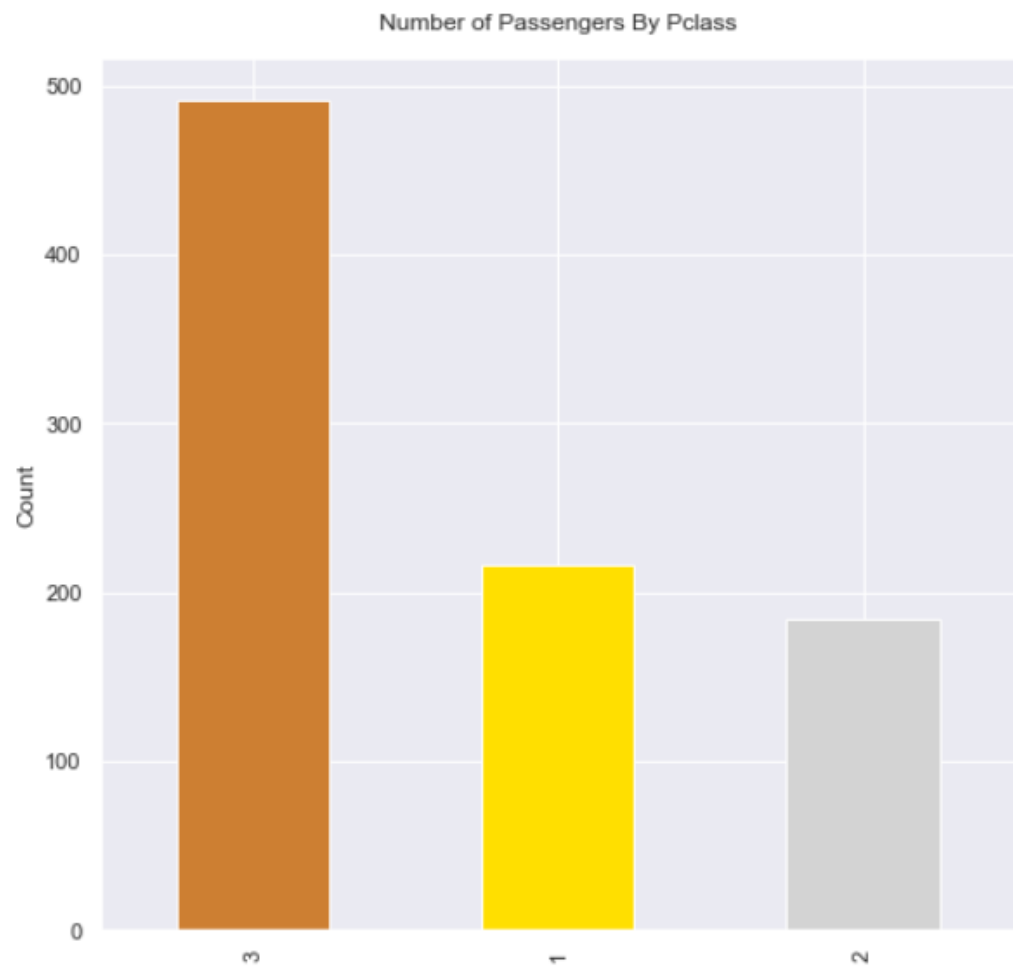
# Titanic – 데이터 분석

- seaborn 의 countplot 을 이용하면, 특정 label 에 따른 개수를 확인해볼 수 있다.

```
y_position = 1.02
f, ax = plt.subplots(1, 2, figsize=(18, 8))
train['Pclass'].value_counts().plot.bar(color=['#CD7F32', '#FFDF00', '#D3D3D3'], ax=ax[0])
ax[0].set_title('Number of Passengers By Pclass', y=y_position)
ax[0].set_ylabel('Count')
sns.countplot('Pclass', hue='Survived', data=train, ax=ax[1])
ax[1].set_title('Pclass: Survived vs Dead', y=y_position)
plt.show()
```

- 클래스가 높을 수록, 생존 확률이 높은걸 확인할 수 있다.
- Pclass 1, 2, 3 순서대로 63%, 48%, 25% 이다.
- 생존에 Pclass 가 큰 영향을 미친다고 생각해볼 수 있으며, 나중에 모델을 세울 때 이 feature 를 사용하는 것이 좋을 것이라 판단할 수 있다.

# Titanic – 데이터 분석



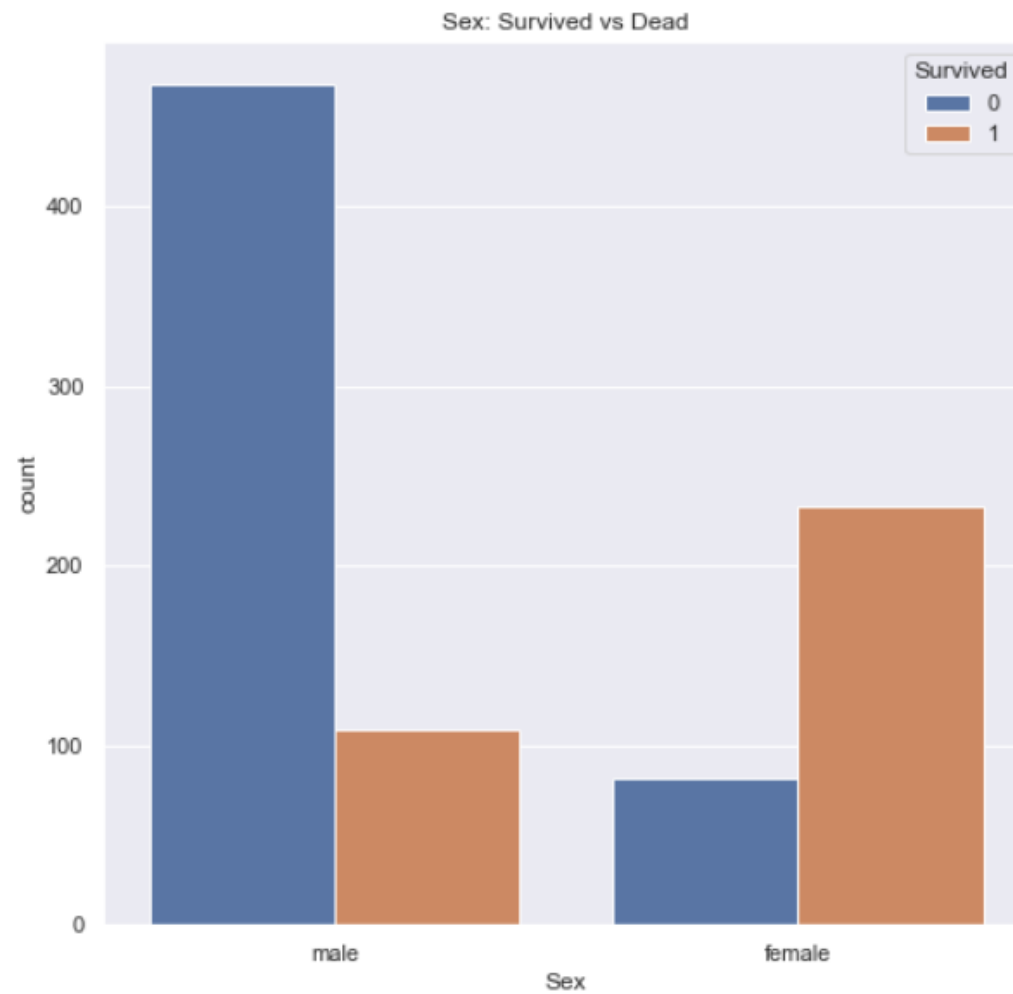
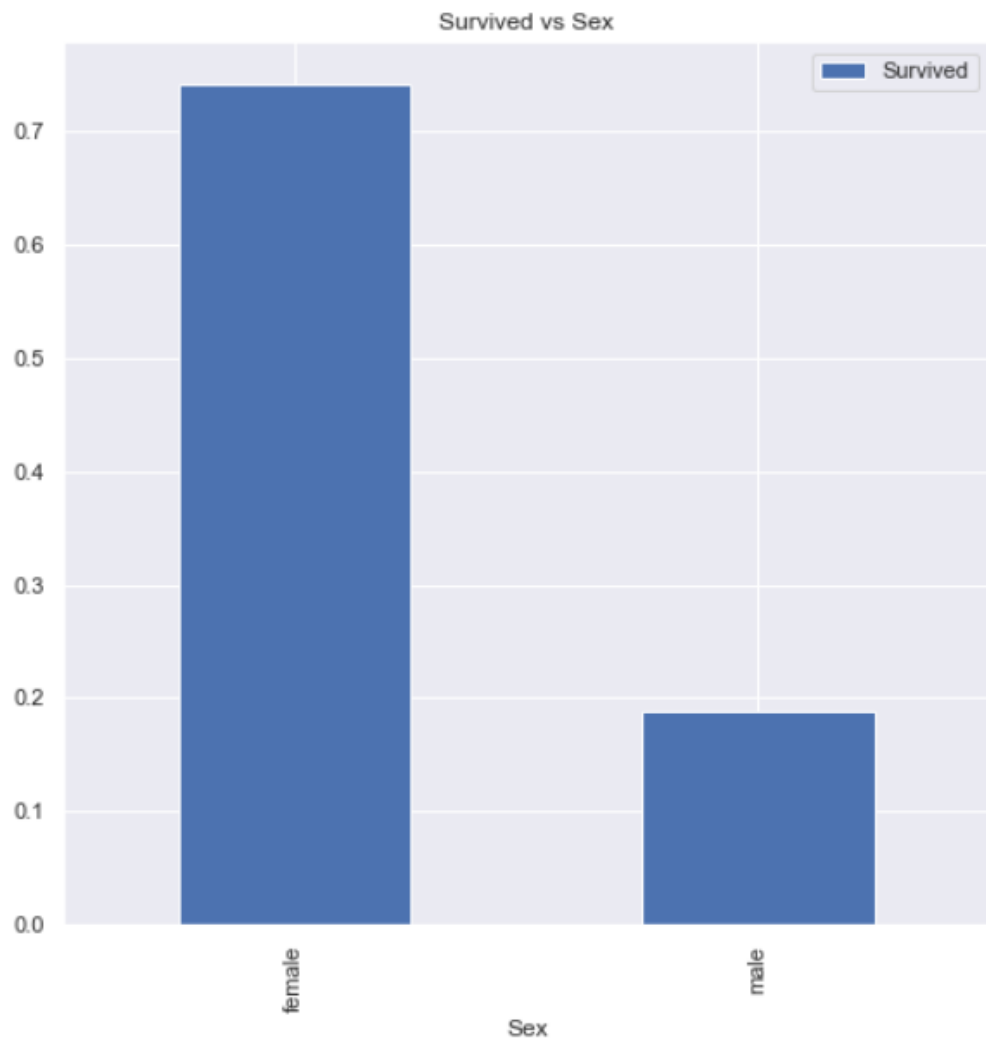
## ■ Sex

- 성별로 생존률이 어떻게 달라지는 지 확인해보자.
- pandas groupby 와 seaborn countplot 을 사용해서 시각화 한다.

```
f, ax = plt.subplots(1, 2, figsize=(18, 8))
train[['Sex', 'Survived']].groupby(['Sex'], as_index=True).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survived vs Sex')
sns.countplot('Sex', hue='Survived', data=train, ax=ax[1])
ax[1].set_title('Sex: Survived vs Dead')
plt.show()
```

# Titanic - 데이터 분석

- 여자가 생존할 확률이 높다.



# Titanic – 데이터 분석

- Pclass 와 마찬가지로, Sex 도 예측 모델에 쓰일 중요한 feature 임을 알 수 있다.

```
train[['Sex', 'Survived']].groupby(['Sex'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

	Sex	Survived
0	female	0.742038
1	male	0.188908

```
pd.crosstab(train['Sex'], train['Survived'], margins=True).style.background_gradient(cmap='summer_r')
```

Survived	0	1	All
Sex			
female	81	233	314
male	468	109	577
All	549	342	891

# Titanic – 데이터 분석

- Both Sex and Pclass

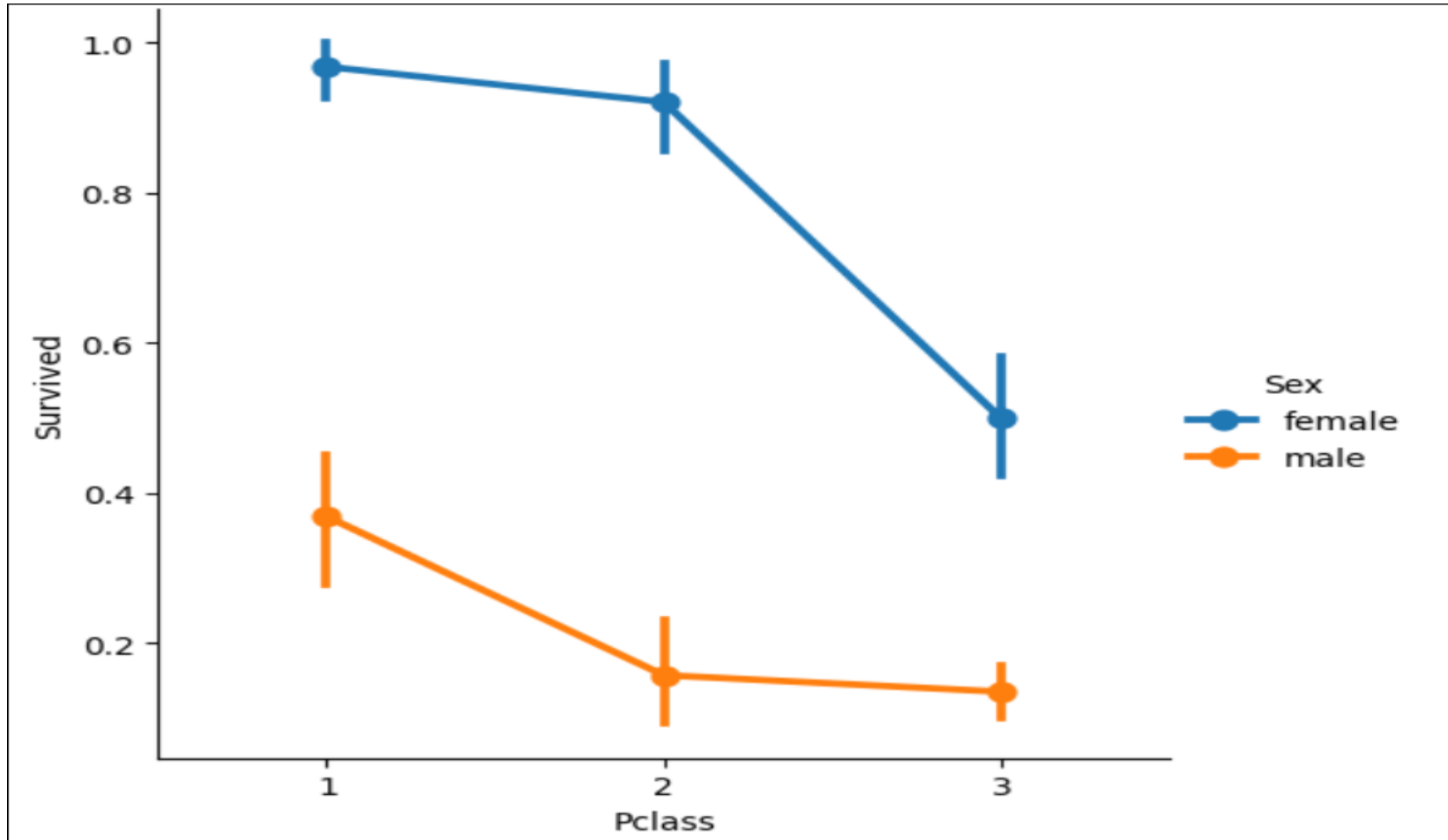
- Sex, Pclass 두가지에 관하여 생존이 어떻게 달라지는 지 확인해 보자.

- seaborn 의 catplot을 이용하면, 손쉽게 3개의 차원으로 이루어진 그래프를 그릴 수 있다.

```
import seaborn as sns
sns.catplot(x='Pclass', y='Survived', hue='Sex', data=t, kind='point')
```

# Titanic – 데이터 분석

- 모든 클래스에서 female 이 살 확률이 male 보다 높은 걸 알 수 있다.
- 남자, 여자 상관없이 클래스가 높을 수록 살 확률 높다.

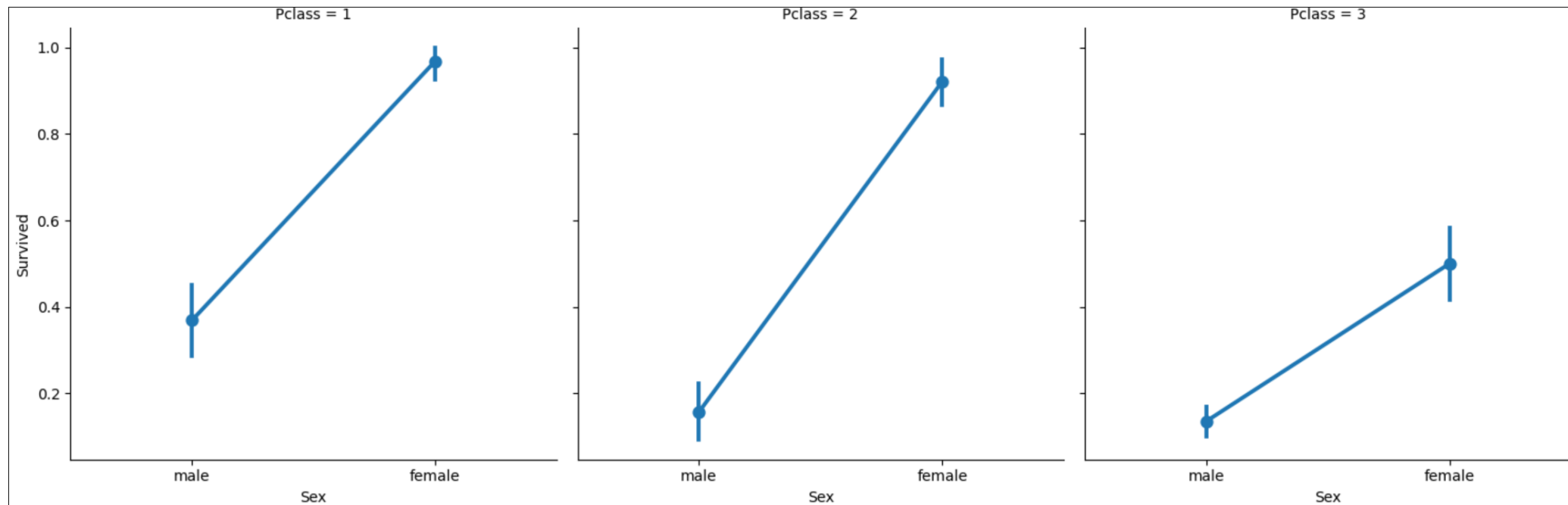




# Titanic – 데이터 분석

- 그래프는 hue 대신 column 으로 하면 다음과 같다.

```
sns.catplot(x='Sex',y='Survived',col='Pclass',data=t,kind='point')
```



## ■ Age feature 를 살펴보자.

```
print('제일 나이 많은 탑승객 : {:.1f} Years'.format(train['Age'].max()))  
print('제일 어린 탑승객 : {:.1f} Years'.format(train['Age'].min()))  
print('탑승객 평균 나이 : {:.1f} Years'.format(train['Age'].mean()))
```

제일 나이 많은 탑승객 : 80.0 Years

제일 어린 탑승객 : 0.4 Years

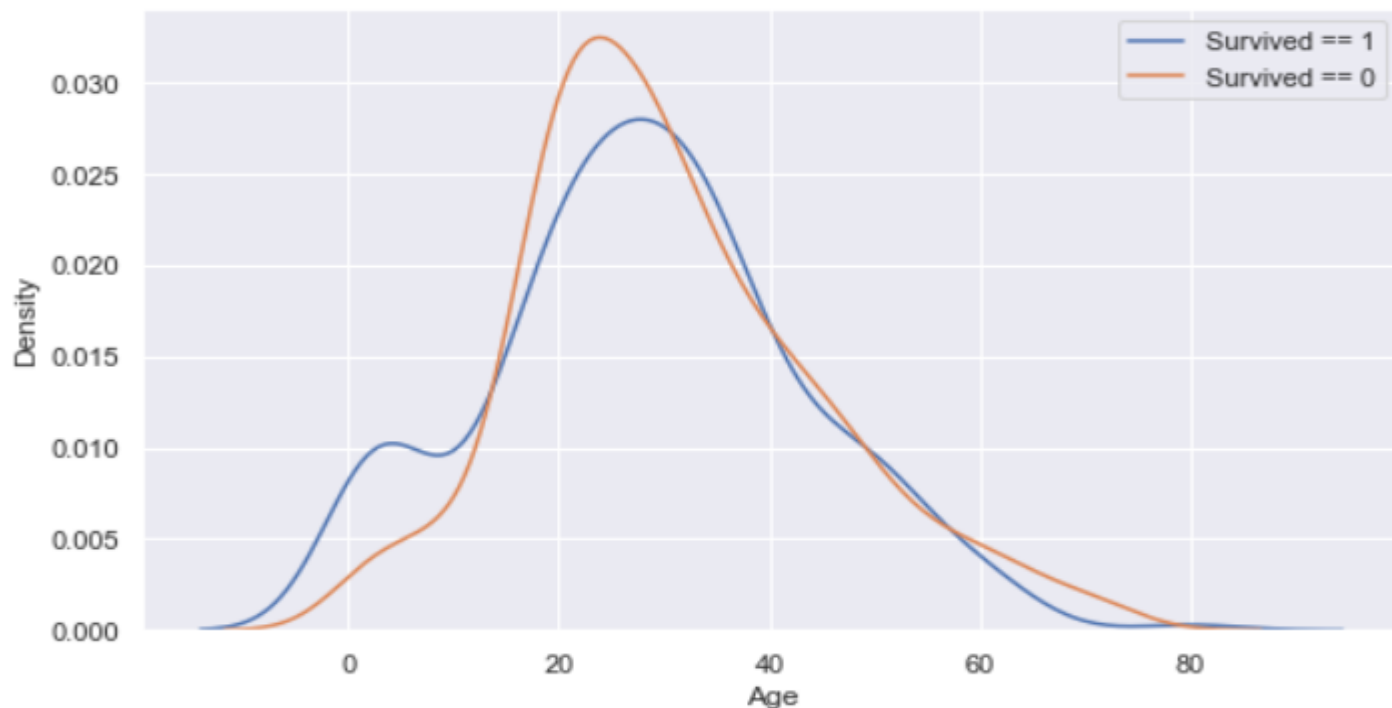
탑승객 평균 나이 : 29.7 Years

# Titanic – 데이터 분석

## ■ 생존에 따른 Age의 histogram

- 생존자 중 나이가 어린 경우가 많음을 볼 수 있다.

```
: fig, ax = plt.subplots(1, 1, figsize=(9, 5))
sns.kdeplot(train[train['Survived'] == 1]['Age'], ax=ax)
sns.kdeplot(train[train['Survived'] == 0]['Age'], ax=ax)
plt.legend(['Survived == 1', 'Survived == 0'])
plt.show()
```



# Titanic – 데이터 분석

- Class 가 높을 수록 나이 많은 사람의 비중이 커진다.

```
# Age distribution within classes
```

```
plt.figure(figsize=(8, 6))
```

```
train['Age'][train['Pclass'] == 1].plot(kind='kde')
```

```
train['Age'][train['Pclass'] == 2].plot(kind='kde')
```

```
train['Age'][train['Pclass'] == 3].plot(kind='kde')
```

```
plt.xlabel('Age')
```

```
plt.title('Age Distribution within classes')
```

```
plt.legend(['1st Class', '2nd Class', '3rd Class'])
```

