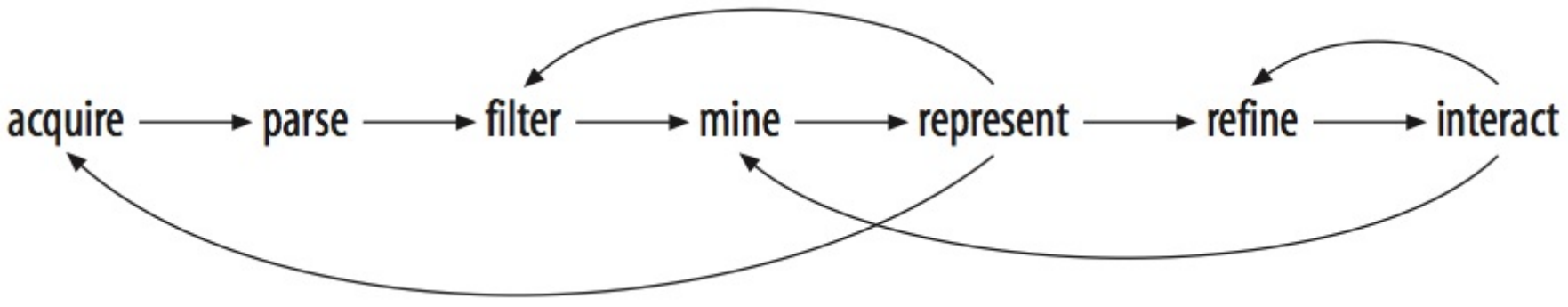
III. 데이터 시각화

벤 플라이의 시각화를 기반으로 시각화 기법 고려

- Visualizing Data의 저자인 벤 플라이어의 정보 시각화 기법.
- 정보 시각화는 데이터 집합의 견지에서 보려고 하지 않으며 질문에서 시작함.
- 왜 데이터가 수집되었나? 어떤 부분이 흥미로웠나? 그리고 어떤 스토리를 말할 수 있는가?
- 데이터를 이해하는 데에있어 가장 중요한 기술들 가운데 하나는 좋은 질문들을 만드는 것.
- 좋은 질문은 데이터에 포함된 흥미로움을 공유하고 다른 이들에게 전달하려고 함. 수학지향적이 아닌 호기심 지향적이다.
- 나만의 질문 조사(助詞) : ~별(그룹), ~에 따른(x축), ~는 어떻게 될까?(y축)
  - A에 따라 어떻게 될까? : x축은 A, y축은 빈도나 백분율
  - A에 따른 B는 어떻게 될까? : x축은 A, y축은 B(B는 산점도와 같은 특정 그래프의 특징이 될 수도, 특정 변수가 될 수 도 있음.
  - A1, A2에 따른 B는 어떻게 될까? : A1, A2 각각의 그래프 그려짐

1. Visulizing Data 절차



- **Acquire :** 데이터를 획득
- **Parse :** 데이터를 구조화 하고 분류
- **Filter :**관심있는 데이터만 추출
- **Mine :** 통계적인 방법 혹은 데이터마이닝 기법을 적용
- **Represent :** 바 그래프, 리스트 혹은 트리 등의 기본적인 시각모델을 선택
- **Refine :** 보다 명확하게, 매력적인 표현으로 개선
- **Interact :**데이터를 변경 혹은 보여지는 특질을 조작하는 방법을 추가

2. 원리

- Each Project Has Unique Requirements : 하나의 시각화는 그 데이터 셋에 표현하는 유일한 특성들만을 표현해야 함. 일반적인 시각화 도구들을 이용해서 표현하는 것은 제대로 된 시각화가 어렵다. 그 데이터만을 위한 고유한 시각화를 만들어 내야함.
- Avoid the All-You-Can-Eat Buffet : 덜 세세한 설명이 더 많은 정보를 전달함. 너무 많은 정보들은 청중을 혼란스럽게 하고, 정작 중요한 것은 전달하지 못함. 가능한 소중한 정보만으로 최소화
- Know Your Audience : 청중이 누구인가? 시각화에 접근하는 이들의 최종목적은 무엇인가? 그들은 무엇을 얻으려고 하는가? 모바일 디바이스와 데스크탑 이용자의 목적은 다르다.

라이브러리

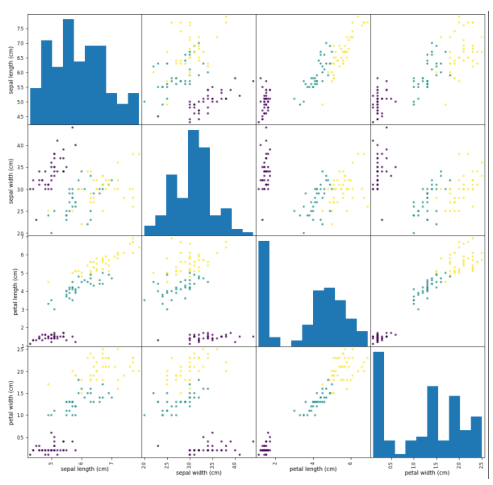
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
```

산점도, 상관 행렬 시각화

- 질문 : 각 특성 수에 따른 산점도는 어떻게 될까?
- 특성의 수가 적을 때 모든 특성을 짝지어 만드는 산점도 행렬(scatter matrix) 사용 권함
- 특성의 선형관계 확인

```
import pandas as pd
import mglearn
iris_dataframe = pd.DataFrame(X_train, columns = iris_dataset.feature_names)

# 데이터프레임을 사용해 y_train에 따라 색으로 구분된 산점도 행렬을 만듦.
pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15),s=60)
```

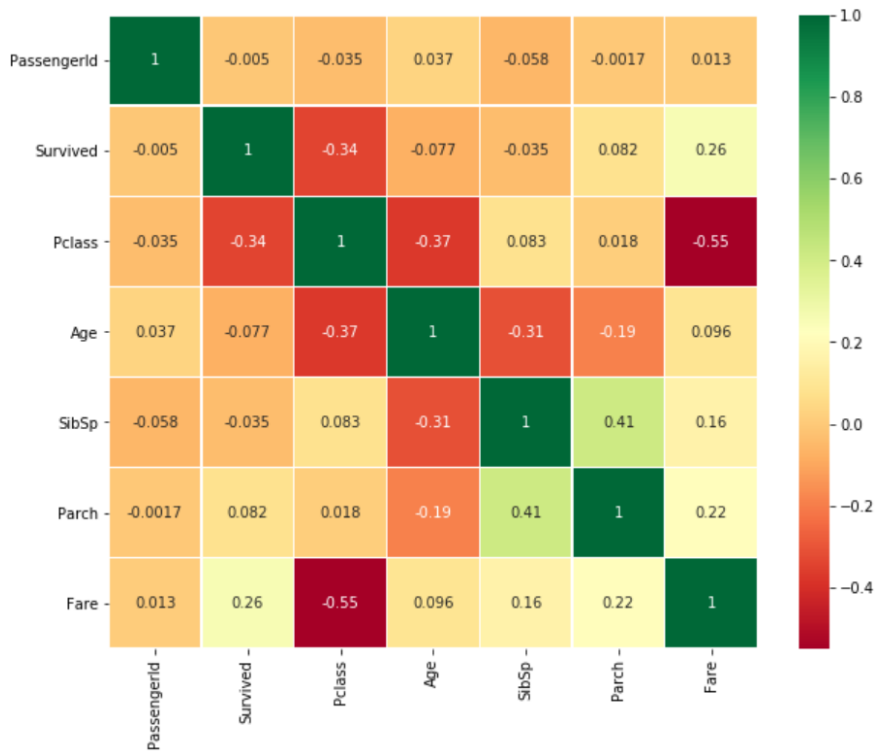


💡 **매개변수**

`iris_dataframe :` 데이터프레임 또는 2차원 배열
`c=` 산점도 위에 뿌려질 벡터
`figsize=`(15, 15) : 그래프 사이즈 조정
`s =` 60 : 점 크기 조정

- 상관행렬 hitmap 표현

```
sns.heatmap(df_train.corr(), annot=True, cmap='RdYlGn', linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(10,8)
```



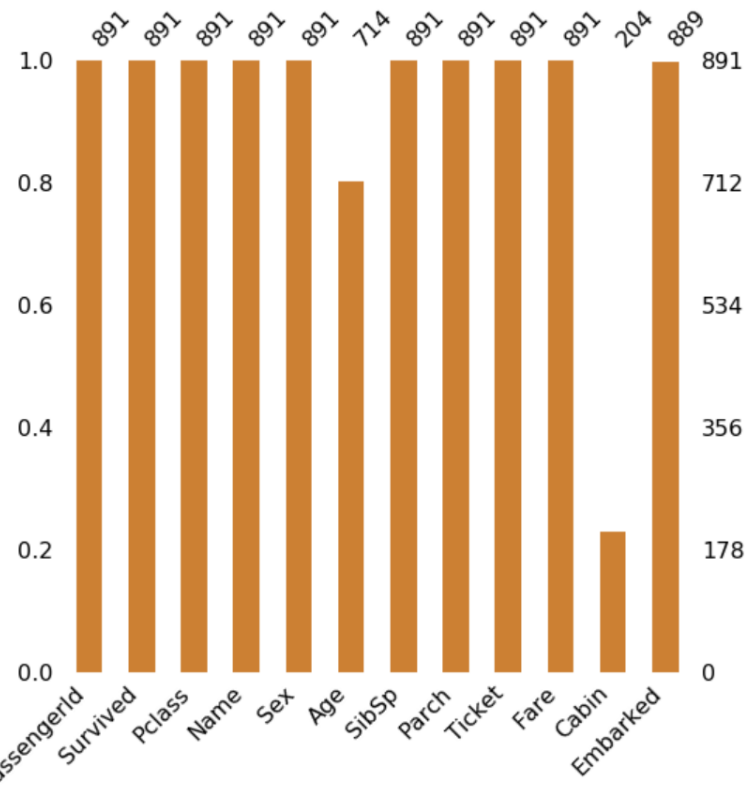
💡 **매개변수**

`df_train.corr() :` DafaFrame.corr() 로 표현
`annot = True` : 각 사각형위 상관계수 표시
`cmap='RdYlGn' :` 상괸계수의 값 (하 중 상)까지 Red, Yellow, Green

변수별 Null 값 시각화

- 질문 : 각 변수별 Null 값의 빈도는 어떻게 될까?
- 변수별 Null 값 파악시 사용
- train과 test 데이터셋 모두 Null을 살펴봐야 함.

```
import missingno as msno
msno.bar(df=df_train.iloc[:,:], figsize=(8,8), color = (0.8,0.5, 0.2))
```



💡 **매개변수**

`figsize=` :그래프 사이즈 조정

`df =` : 데이터셋 명 지정

타겟 레이블 시각화

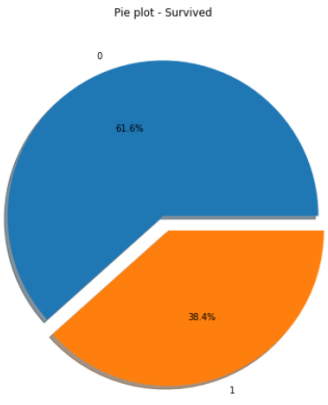
- 질문 : 타깃 변수의 분포는 어떻게 될까? (왼, 오 그래프 모두 해당)
- 타겟 레이블의 비율 확인

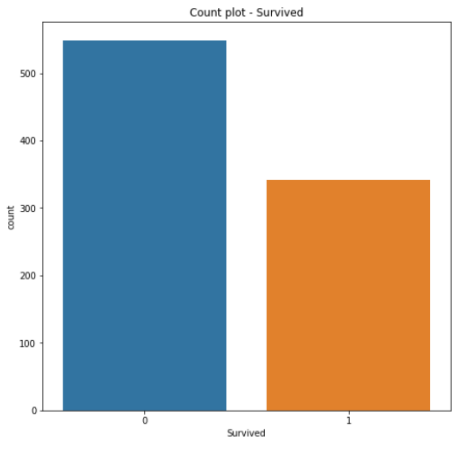
```
import matplotlib.pyplot as plt
import seaborn as sns

f, ax = plt.subplots(1,2, figsize = (18,8))
# 그래프 틀 두개 만들기 f에 틀 , 그려질 그래프를 ax 에 저장

df_train['Survived'].value_counts().plot.pie(explode=[0,0.1], shadow=True, ax=ax[0],
                                             autopct='%1.1f%%')
ax[0].set_title('Pie plot - Survived') # 첫번째 그래프 타이틀
ax[0].set_ylabel('') #첫번쨔 그래프 라벨링

sns.countplot('Survived', data=df_train, ax=ax[1]) # 두번째 그래프 countplot : 막대그래프
ax[1].set_title('Count plot - Survived')
```





💡 **매개변수**

`explode =` : 원조각 분리된 간격 조정

`shadow =` : 그래프 그림자 유/무 조정

`ax = ax[0]` : 위 f, ax에서 그려질 그래프의 위치 지정. ax[0] 은 왼쪽부터 그래프 삽입됨

`autopct=` : 그래프 위에 비율값 생성

타겟 변수(순서형) vs 하나의 변수(순서형) 교차표 시각화

- 질문 : 두 변수의 교차표는 어떻게 될까?
- Survived 0과 1을 Pclass로 묶고 이를 카운트
- 색이 짙은 부분으로 많은 수가 위치하는 곳을 파악할 수 있음.

```
pd.crosstab(df_train['Pclass'], df_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

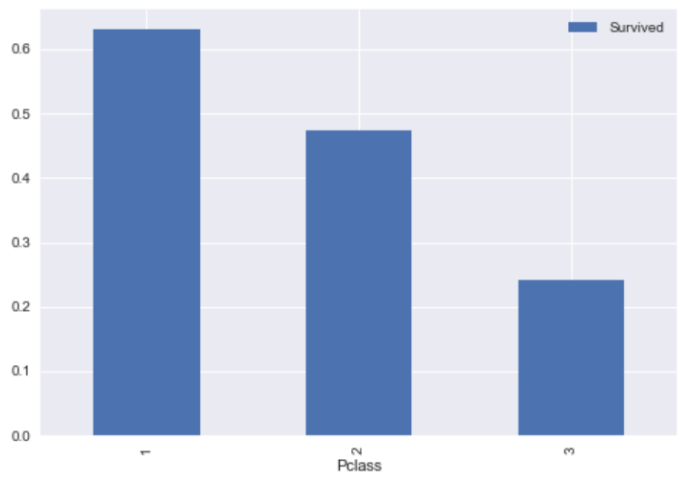
💡 **매개변수**

`margins= True :` 열합, 행합을 나타냄.

`style.background_gradient(cmap='summer_r') :` 색갈을 입힘

타겟 변수(순서형) vs 하나의 변수(순서형) 막대그래프 비교

- 질문 : Pclass에 따른 생존(survived)의 비율은 어떻게 될까?
- 본 예의 타겟 변수는 Survived 이며 비교대상인 변수는 Pclass
- 비교 대상 변수(Pclass)를 사용해 Survived를 그룹화 하여 평균을 낸 값으로 그래프 그림
- 이때 그림은 전체 Pclass의 빈도수를 기준으로 백분율로 표현 됨.
- 이를 통해 Pclass가 높을수록 생존 확률(Survived)이 높은걸 알 수 있음.
  (Pclass는 1,2,3, 순서대로 높다)




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

💡 **매개변수**

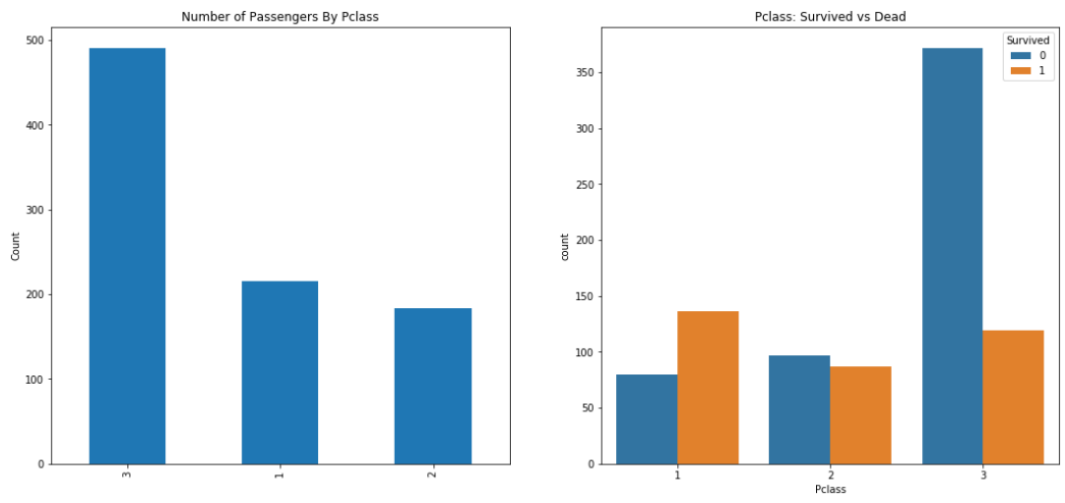
`as_index = True :` 아래 Pclass 이름별로 빈도표 보여줌

`groupby([변수명]) :` 열 역할을 할 변수

특정 변수 단일 빈도표와 타겟별 특정변수 막대그래프 비교

- 질문1) Pclass의 분포는 어떻게 될까 ? - 왼쪽그래프
  질문2) 생존(Survived)별 Pclass의 분포는 어떻게 될까? - 오른쪽 그래프
- 클래스가 높을 수록 생존확률이 높은 걸 확인할 수 있으며 생존에 Pcalss가 큰 영향을 미친다 생각할 수 있음.
- 나중에 모델 세울 때 이 feature을 사용하는 것이 좋을 것이라 판단할 수 있음.

```
f, ax = plt.subplots(1,2,figsize=(18,8))
df_train['Pclass'].value_counts().plot.bar(ax=ax[0])
ax[0].set_title('Number of Passengers By Pclass')
ax[0].set_ylabel('Count')
sns.countplot('Pclass', hue='Survived', data=df_train, ax=ax[1])
ax[1].set_title('Pclass: Survived vs Dead')
```



💡 **매개변수**

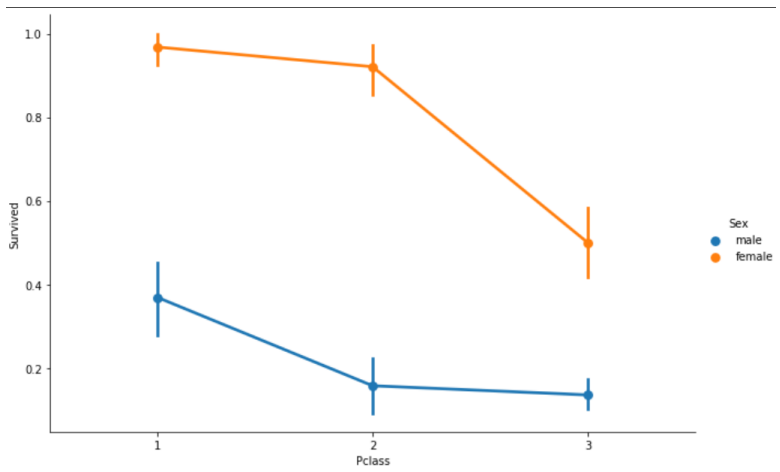
`hue =` : group으로 나눌 변수

타겟 변수와 2개의 명목형 변수 비교 시각화

- 질문 : 성별별로 Pclass에 따른 생존(Survived)는 어떻게 될까?
- Pclass, Survived를 sex 변수에 따라 비교

- 관심있게 볼 변수(타깃변수)는 y 축에 두어 높, 낮음을 시각적으로 표현(만약 없다면 count가 y축)

```
sns.factorplot('Pclass','Survived',hue='Sex',data=df_train, size=6, aspect=1.5)
```

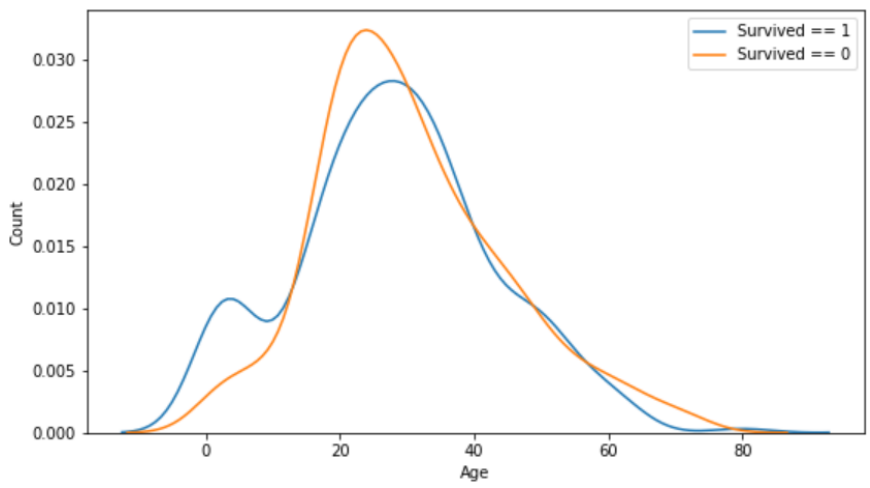


💡 **매개변수**

`hue=` : 그룹별로 볼 변수 지정

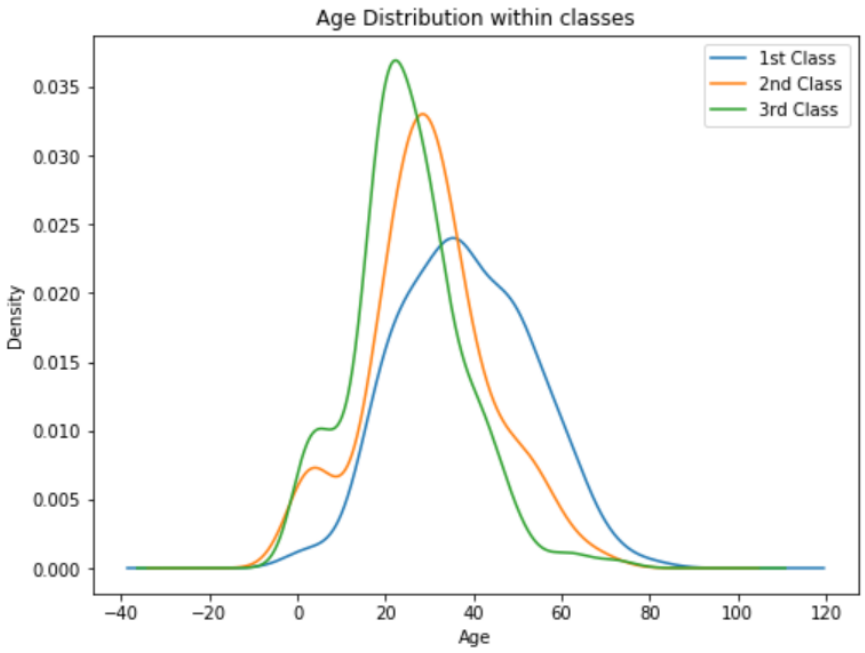
타겟 변수와 1개의 연속형 변수 비교 시각화

- 질문 : 생존별 나이 분포는 어떻게 될까?
- 생존(Survived)에 따른 나이의 히스토그램
- 나이가 어릴수록 생존이 높음을 알 수 있음.




```
fig, ax = plt.subplots(1,1,figsize=(9,5))
# 1,1 면 하나의 도표에 겹쳐 그리겠다. 또한 뒤에 ax=ax로 표현됨.
sns.kdeplot(df_train[df_train['Survived'] == 1]['Age'], ax=ax)
sns.kdeplot(df_train[df_train['Survived'] == 0]['Age'], ax=ax)
ax.set_ylabel('Count')
ax.set_xlabel('Age')
plt.legend(['Survived == 1', 'Survived == 0'])
#범례 표시
```

- 질문 : Pclass별 나이의 분포는 어떻게 될까?
- class가 높을수록 나이 많은 사람의 비중이 커짐.




```
# classes 별 나이 분포
plt.figure(figsize = (8,6))
df_train['Age'][df_train['Pclass']==1].plot(kind='kde')
df_train['Age'][df_train['Pclass']==2].plot(kind='kde')
df_train['Age'][df_train['Pclass']==3].plot(kind='kde')
#Kernal Density Estimate : 커널밀도 추정 , 밀도함수를 붙여 그릴 수 있게 해줌.
plt.xlabel('Age')
plt.title('Age Distribution within classes')
plt.legend(['1st Class', '2nd Class', '3rd Class'])
```

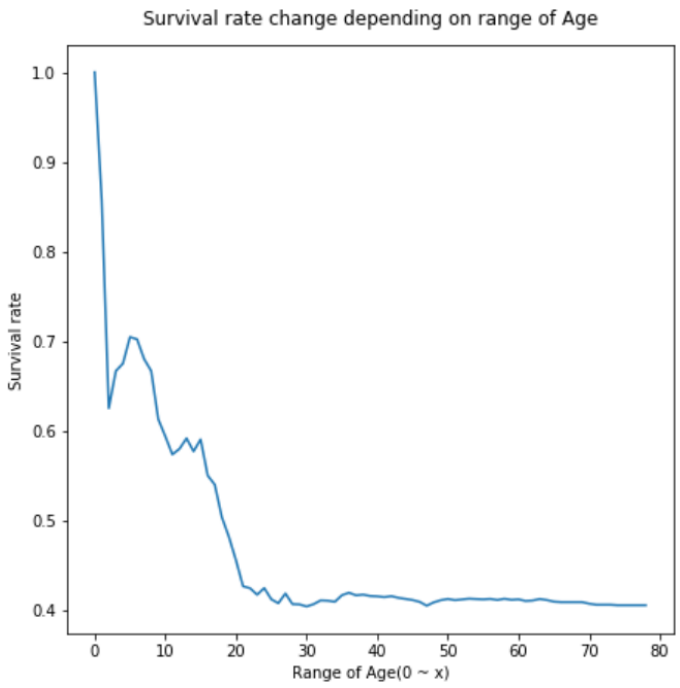
특정 변수(연속형)에 따른 생존(타겟) 그래프

- 질문 : 나이에 따라 생존률이 어떻게 변할까?
- 관심있는 대상(여기서는 타겟)이 y축

1. 1살부터 80살까지 한살씩 늘어날 때마다 해당 나이에 포함되는 인원수를 합함.
2. 해당 인원들 중 생존한 사람들의 수를 합한 값으로 나누어 줌.

- 나이가 어릴수록 생존률이 확실히 높다는 것을 알 수 있음. 따라서 나이가 feature로 쓰일 수 있음을 확인

```
cummulate_survival_ratio=[]
for i in range(1, 80):
    cummulate_survival_ratio.append(df_train[df_train['Age'] < i]['Survived'].\
                                sum()/len(df_train[df_train['Age']<i]['Survived']))
plt.figure(figsize=(7,7))
plt.plot(cummulate_survival_ratio)
plt.title('Survival rate change depending on range of Age', y=1.02)
plt.ylabel('Survival rate')
plt.xlabel('Range of Age(0 ~ x)')
```



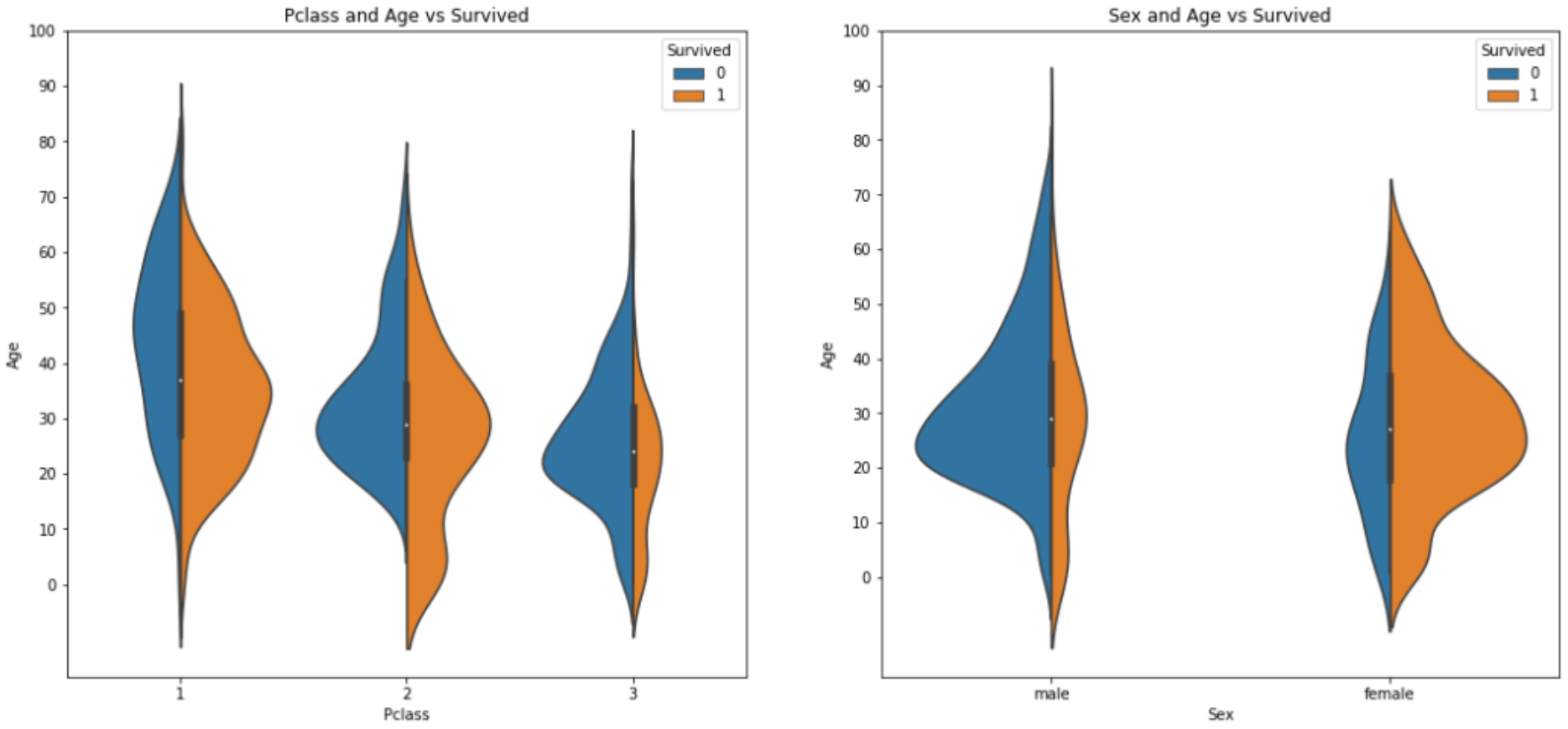
💡 **매개변수**

1개의 타겟(순서형), 2개의 순서형, 1개의 연속형인 그래프

- 질문 : survived ㅂ성별과 Pclass에 따라 나이의 분포는 어떻게 될까?
- x축은 우리가 나눠서 보고 싶어하는 case(Pclass, Sex)를 나타내고, y축은 보고 싶어하는 distribution(age)임.
- 관심있는 대상은 나이이므로 y축.

```
f, ax = plt.subplots(1,2, figsize=(18,8))
sns.violinplot("Pclass","Age", hue="Survived", data=df_train, scale='count', split=True, ax= ax[0])
ax[0].set_title('Pclass and Age vs Survived')
ax[0].set_yticks(range(0,110,10))

sns.violinplot("Sex","Age", hue = "Survived", data =df_train, scale = 'count', split =True, ax=ax[1])
ax[1].set_title('Sex and Age vs Survived')
ax[1].set_yticks(range(0,110,10))
plt.show()
```



💡 **매개변수**

앞의 변수들을 가지고 새로운 변수를 분석할 때

- 글로벌 질문 : 이 새로운 변수는 생존율과 관련이 있을까?
- 1) 전체적으로 빈도 수 확인 [꼬리 질문1 : 항구(Embarked)의 분포는 어떻게 될까?]
- 2) 각 변수 별로 비교
  꼬리질문 2 : 성별별로 항구의 빈도가 어떻게 될까?
  꼬리질문 3 : 생존율별 항구의 빈도가 어떻게 될까?
  꼬리질문 4 : Pclass별 항구의 빈도가 어떻게 될까?
- 앞전에 클래스가 높은 사람들이 생존확률이 높다는 것을 알았다. 이를 꼬리질문 3과 4를 엮어서 보면 C항구의 생존율이 높았던 원천적인 이유는 Pclass의 클래스가 높은 사람들이 많아서 였던 것으로 결론을 낼수 있음. 이와같이 S항구 또한 class가 낮은 사람들이 많아 생존확률이 낮게 나옴.

```
f, ax = plt.subplots(2,2, figsize=(20,15))
```

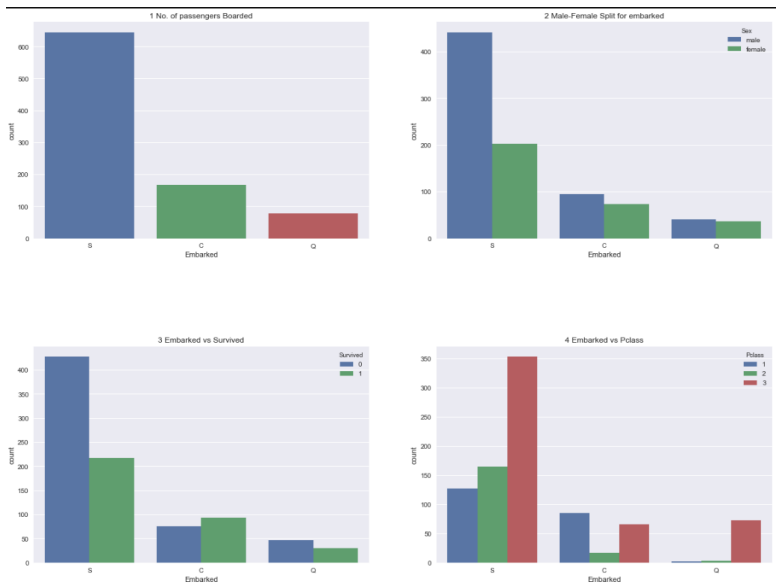
```
sns.countplot('Embarked', data=df_train, ax=ax[0,0])
ax[0,0].set_title('1 No. of passengers Boarded')

sns.countplot('Embarked', hue='Sex', data=df_train, ax=ax[0,1])
ax[0,1].set_title('2 Male-Female Split for embarked')

sns.countplot('Embarked', hue='Survived', data=df_train, ax=ax[1,0])
ax[1,0].set_title('3 Embarked vs Survived')

sns.countplot('Embarked', hue='Pclass', data=df_train, ax=ax[1,1])
ax[1,1].set_title('4 Embarked vs Pclass')

plt.subplots_adjust(wspace=0.2, hspace=0.5)
```

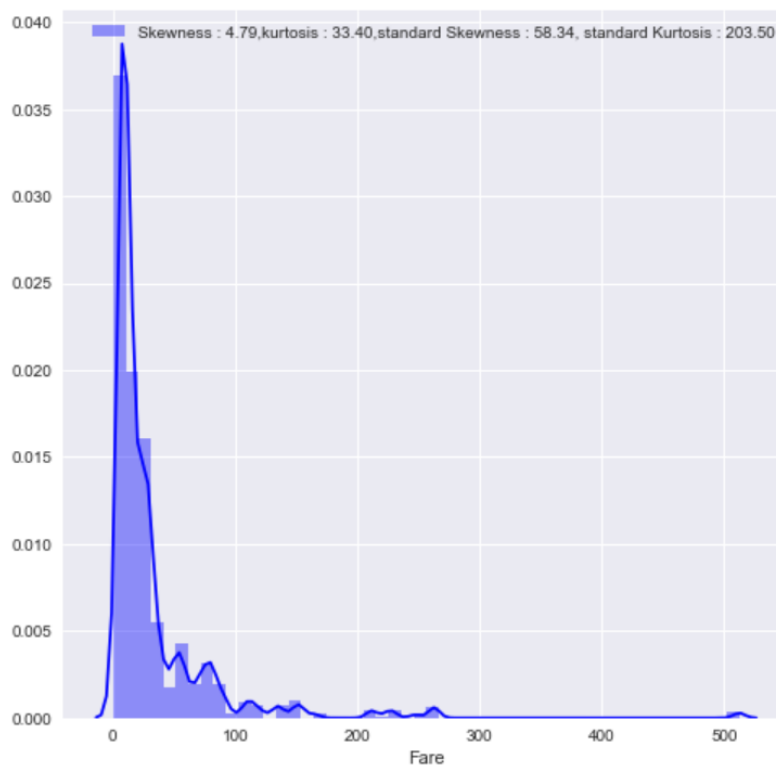


연속형변수의 분포 파악(왜도, 첨도, 표준화 왜도, 표준화 첨도)

- 질문 : 연속형 변수의 정규성은 어떻게 될까?
- 본 예에서는 한쪽으로 치우쳐 있기 때문에 스케일링 후 로그로 변수변환하여 사용

```
standard_skew = df_train['Fare'].skew()/np.sqrt(6/len(df_train['Fare']))
standard_kurt = df_train['Fare'].kurt()/np.sqrt(24/len(df_train['Fare']))

fig, ax = plt.subplots(1,1, figsize=(8,8))
g= sns.distplot(df_train['Fare'], color='b', label='Skewness : {:.2f},\
kurtosis : {:.2f},standard Skewness : {:.2f}, standard Kurtosis : {:.2f}'.\
                format(df_train['Fare'].skew(),df_train['Fare'].kurt()\
                       ,standard_skew, standard_kurt ),ax=ax)
g=g.legend(loc='best')
```



💡 **매개변수**

`standard_skew, standard_kurt` : 왜도, 첨도 표준화 변수

분석기법

- 특징1
- 특징2

코드

💡 **매개변수**

분석기법

- 특징1
- 특징2

코드

💡 **매개변수**