11과 [실습] 숫자 vs 범주

1.환경준비

- 라이브러리 불러오기

In [1]:

```
import pandas as pd
import numpy as np
import random as rd

import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm
```

- 데이터 불러오기 : 다음의 예제 데이터를 사용합니다.

> ① 타이타닉 생존자
> ② 보스톤 시, 타운별 집값
> ③ 아이리스 꽃 분류
> ④ 뉴욕 공기 오염도

In [2]:

```
# 타이타닉 데이터
titanic = pd.read_csv('https://bit.ly/3HaMAtZ')
titanic.head()
```

Out[2]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	AgeGroup	Family	Age_
0	0	3	male	22.0	1	0	7.2500	S	Age21_30	2	0.2
1	1	1	female	38.0	1	0	71.2833	C	Age31_40	2	0.4
2	1	3	female	26.0	0	0	7.9250	S	Age21_30	1	0.3
3	1	1	female	35.0	1	0	53.1000	S	Age31_40	2	0.4
4	0	3	male	35.0	0	0	8.0500	S	Age31_40	1	0.4

In [3]:

```
# 아이리스 꽃 분류 : vesicolor? 1, 0

iris = pd.read_csv('https://raw.githubusercontent.com/DA4BAM/dataset/master/iris.csv')
iris = iris.loc[iris['Species'] != 'setosa']
iris['versicolor'] = np.where(iris.Species == 'versicolor', 1, 0)
iris.drop(['Species','Sepal.Length','Petal.Length'], axis = 1, inplace = True)
iris.head()
```

Out[3]:

	Sepal.Width	Petal.Width	versicolor
50	3.2	1.4	1
51	3.2	1.5	1
52	3.1	1.5	1
53	2.3	1.3	1
54	2.8	1.5	1

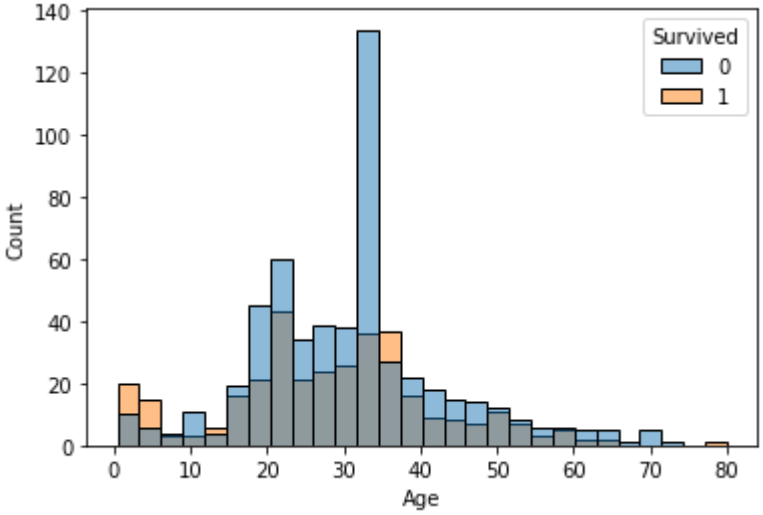
2.시각화

① titanic : Age -- > Survived

- 히스토그램으로 관계를 살펴 봅시다.

In [4]:

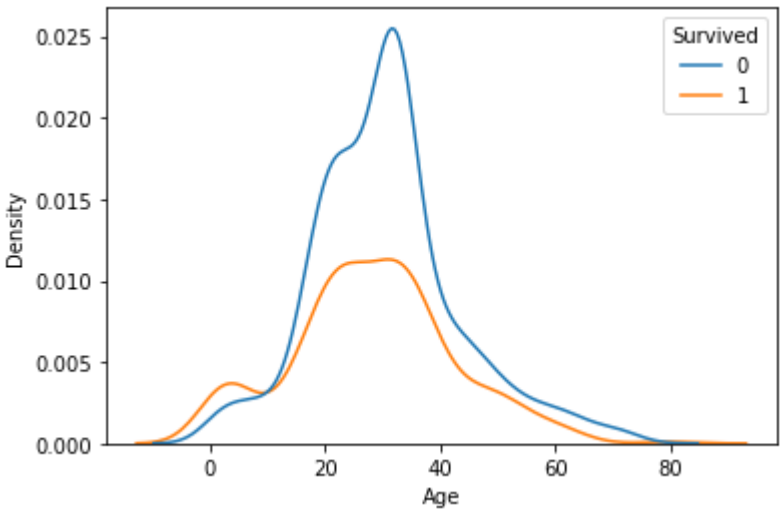
```
sns.histplot(x='Age', data = titanic, hue = 'Survived')
plt.show()
```



- density plot으로 비교해봅시다.
  ① kdeplot(, hue = 'Survived)

In [5]:

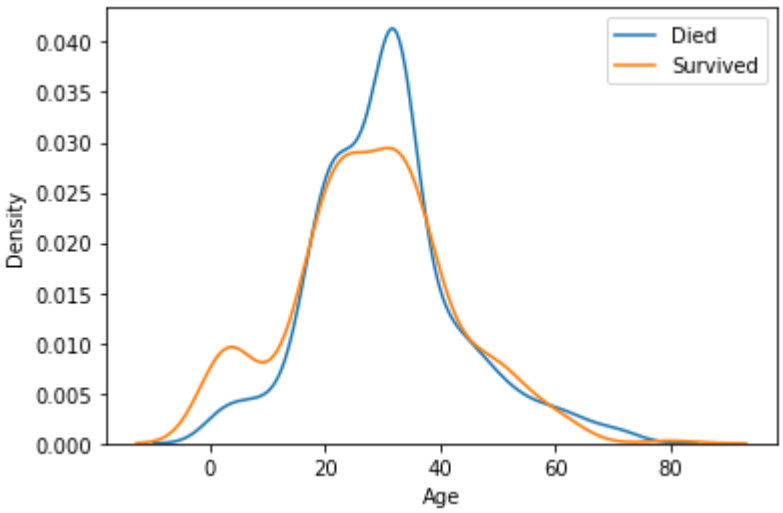
```
sns.kdeplot(x='Age', data = titanic, hue ='Survived')
plt.show()
```



② Survived == 1, kdeplot / Survived == 0, kdeplot 겹쳐서 그리기

In [6]:

```
t0 = titanic.loc[titanic['Survived']==0]
t1 = titanic.loc[titanic['Survived']==1]

sns.kdeplot(x='Age', data = t0, label = 'Died')
sns.kdeplot(x='Age', data = t1, label = 'Survived')

plt.legend()
plt.show()
```



- 차트를 해석해 봅시다.

In [ ]:

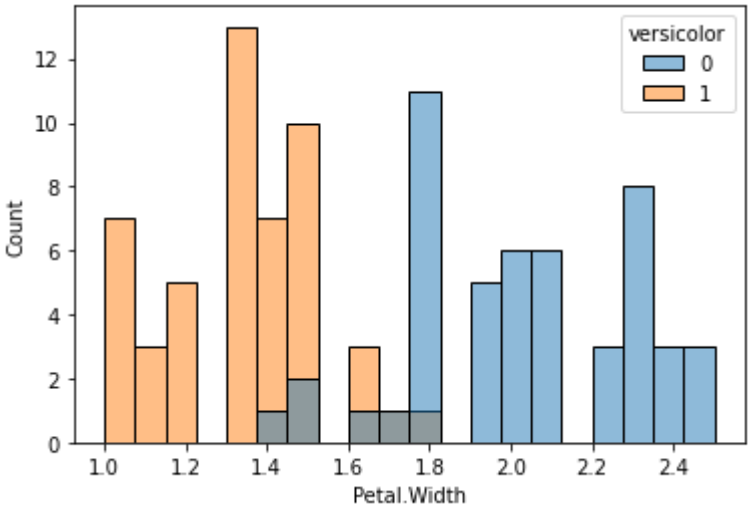
```

```

② iris : Petal.Width -- > versicolor

- 히스토그램으로 관계를 살펴 봅시다.

In [7]:

```
sns.histplot(x='Petal.Width', data = iris, hue = 'versicolor', bins = 20)
plt.show()
```

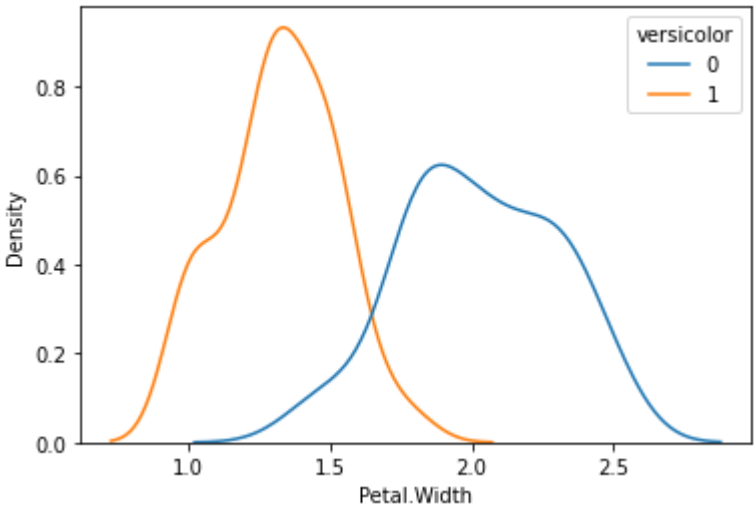


- density plot으로 비교해봅시다.

① kdeplot(, hue =)

In [8]:

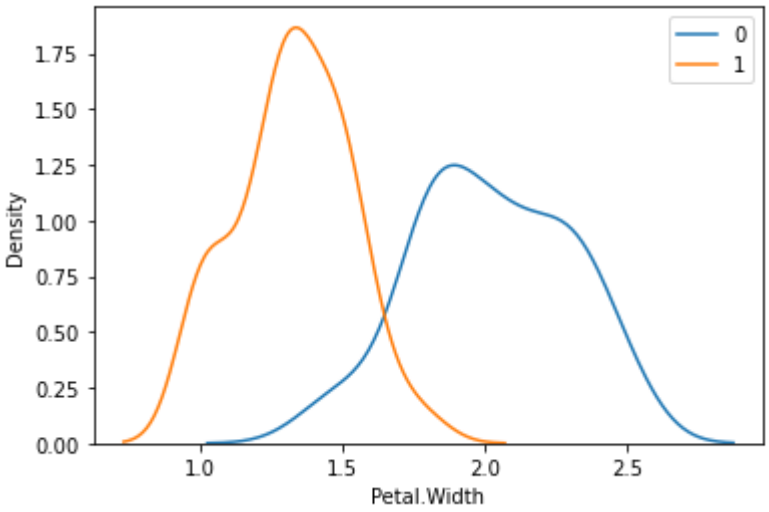
```
# sns.kdeplot(  , hue = )
sns.kdeplot(x='Petal.Width', data = iris, hue = 'versicolor')
plt.show()
```



② kdeplot 겹쳐서 그리기

In [9]:

```
# sns.kdeplot(  )
# sns.kdeplot(  )

v0 = iris.loc[iris['versicolor']==0]
v1 = iris.loc[iris['versicolor']==1]

sns.kdeplot(x='Petal.Width', data = v0, label = '0')
sns.kdeplot(x='Petal.Width', data = v1, label = '1')

plt.legend()
plt.show()
```



- 차트를 해석해 봅시다.

In [ ]:

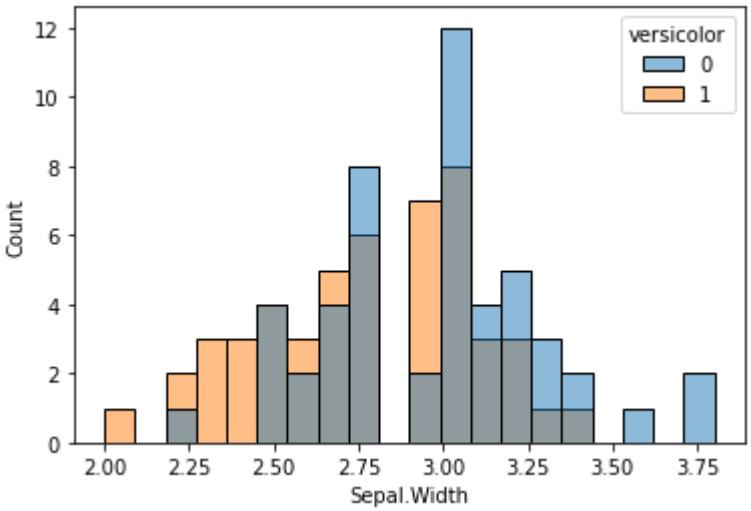
```

```

③ iris : Sepal.Length -- > versicolor

- 히스토그램으로 관계를 살펴 봅시다.

In [10]:

```
1 sns.histplot(x='Sepal.Width', data = iris, hue = 'versicolor', bins = 20)
2 plt.show()
```

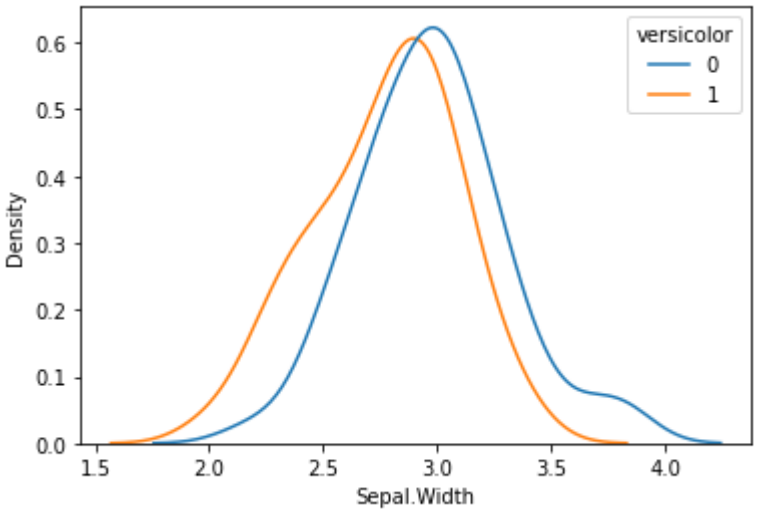


- density plot으로 비교해봅시다.

① kdeplot(, hue =)

In [11]:

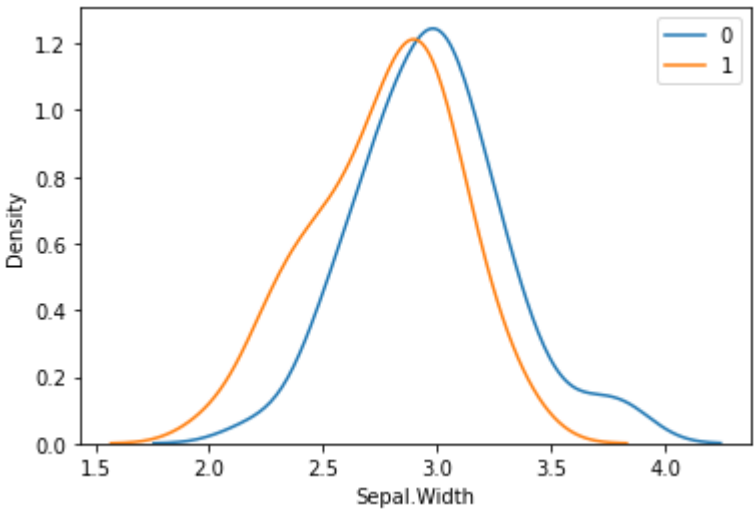
```
# sns.kdeplot(  , hue = )
sns.kdeplot(x='Sepal.Width', data = iris, hue = 'versicolor')
plt.show()
```



② kdeplot 겹쳐서 그리기

In [12]:

```
# sns.kdeplot(  )
# sns.kdeplot(  )

v0 = iris.loc[iris['versicolor']==0]
v1 = iris.loc[iris['versicolor']==1]

sns.kdeplot(x='Sepal.Width', data = v0, label = '0')
sns.kdeplot(x='Sepal.Width', data = v1, label = '1')

plt.legend()
plt.show()
```



- 차트를 해석해 봅시다.

In [ ]:

```
1
```

3.수치화 : 로지스틱 회귀 모델로 부터 p value 구하기

숫자 --> 범주에 대해 딱 맞는 가설검정 도구가 없으므로, 로지스틱 회귀 모델로 부터 p-value를 구해봅시다.

① titanic : Age -- > Survived

- 로지스틱 회귀 모형으로 부터 pvalue를 구해 봅시다.

In [13]:

```
model = sm.Logit(titanic['Survived'], titanic['Age'])
result = model.fit()
print(result.pvalues)
```

```
Optimization terminated successfully.
         Current function value: 0.661967
         Iterations 4
Age    3.932980e-13
dtype: float64
```

- 결과를 해석해 봅시다.

In [ ]:

```
1
```

② iris : Petal.Length -- > versicolor

- 로지스틱 회귀 모형으로 부터 pvalue를 구해 봅시다.

In [14]:

```
model = sm.Logit(iris['versicolor'], iris['Petal.Width'])
result = model.fit()
print(result.pvalues)
```

```
Optimization terminated successfully.
         Current function value: 0.672469
         Iterations 4
Petal.Width    0.044705
dtype: float64
```

- 결과를 해석해 봅시다.

In [ ]:

```
1
```

③ iris : Sepal.Length -- > versicolor

- 로지스틱 회귀 모형으로 부터 pvalue를 구해 봅시다.

In [15]:

```
1 model = sm.Logit(iris['versicolor'], iris['Sepal.Width'])
2 result = model.fit()
3 print(result.pvalues)
```

```
Optimization terminated successfully.
         Current function value: 0.692525
         Iterations 3
Sepal.Width    0.724284
dtype: float64
```

- 결과를 해석해 봅시다.

In [ ]:

```
1
```