4. EDA와 시각화

4.1 막대그래프와 히스토그램

4.1.1 막대그래프

막대그래프는 범주형 데이터를 요약하고 시각적으로 비교하는 데 효과적인 그래프입니다.

📌 bar

> data_cnt = data['target'].value_counts()
> plt.bar(x, height, width=0.8, bottom=None, align='center', data=None)

- x: category와 그에 해당하는 데이터
- height: 높이
- align: x 좌표에 대한 막대 정렬

📍 matplotlib

matplotlib.pyplot 라이브러리는 Python에서 데이터 시각화를 위한 강력하고 널리 사용되는 도구입니다. plt는 matplotlib의 pyplot 모듈을 줄여서 사용하는 약칭으로, 그래프와 플롯을 쉽게 그릴 수 있도록 도와줍니다.

📜 예제

> 범주형 데이터인 와인의 종류별 개수를 확인하는 막대그래프 그리기

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine

# 데이터 불러오기
wine_load = load_wine()
wine = pd.DataFrame(wine_load.data, columns=wine_load.feature_names)
wine.head()
```

Out[ ]:

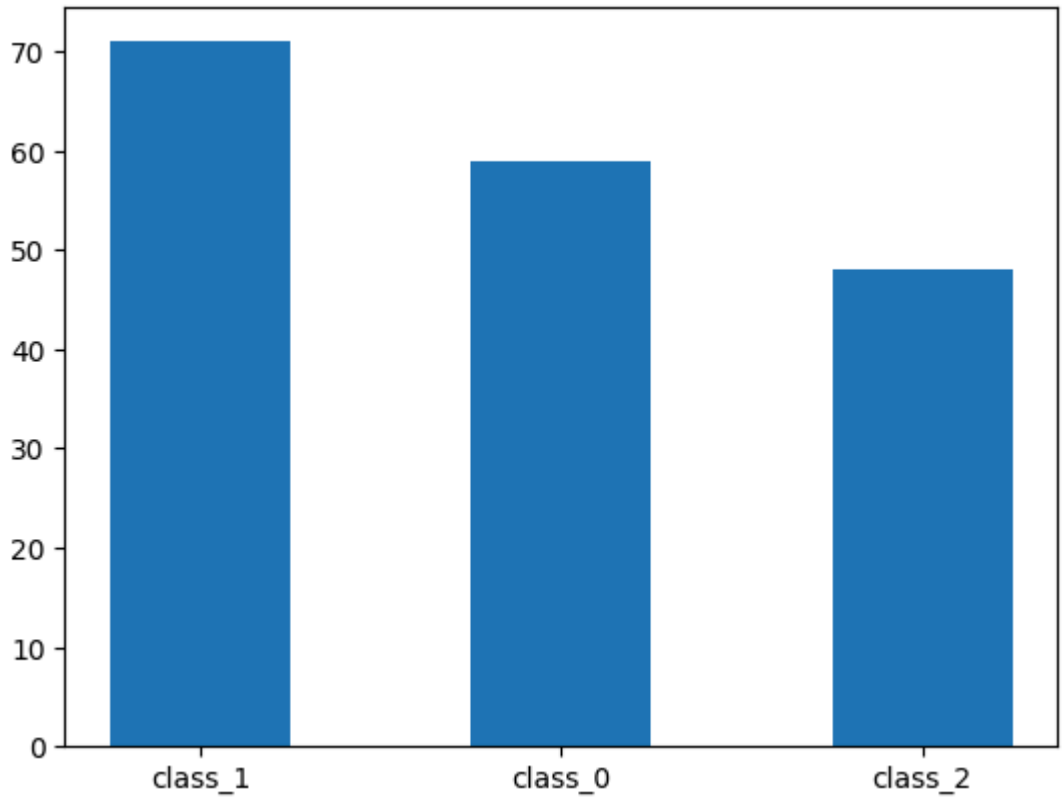
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonfla
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

In [ ]:
```
# 도수 분포표 만들기
wine['Class'] = wine_load.target
wine['Class'] = wine['Class'].map({0:'class_0', 1:'class_1', 2:'class_2'})
wine_type = wine['Class'].value_counts()
wine_type
```

Out[ ]:
```
class_1    71
class_0    59
class_2    48
Name: Class, dtype: int64
```

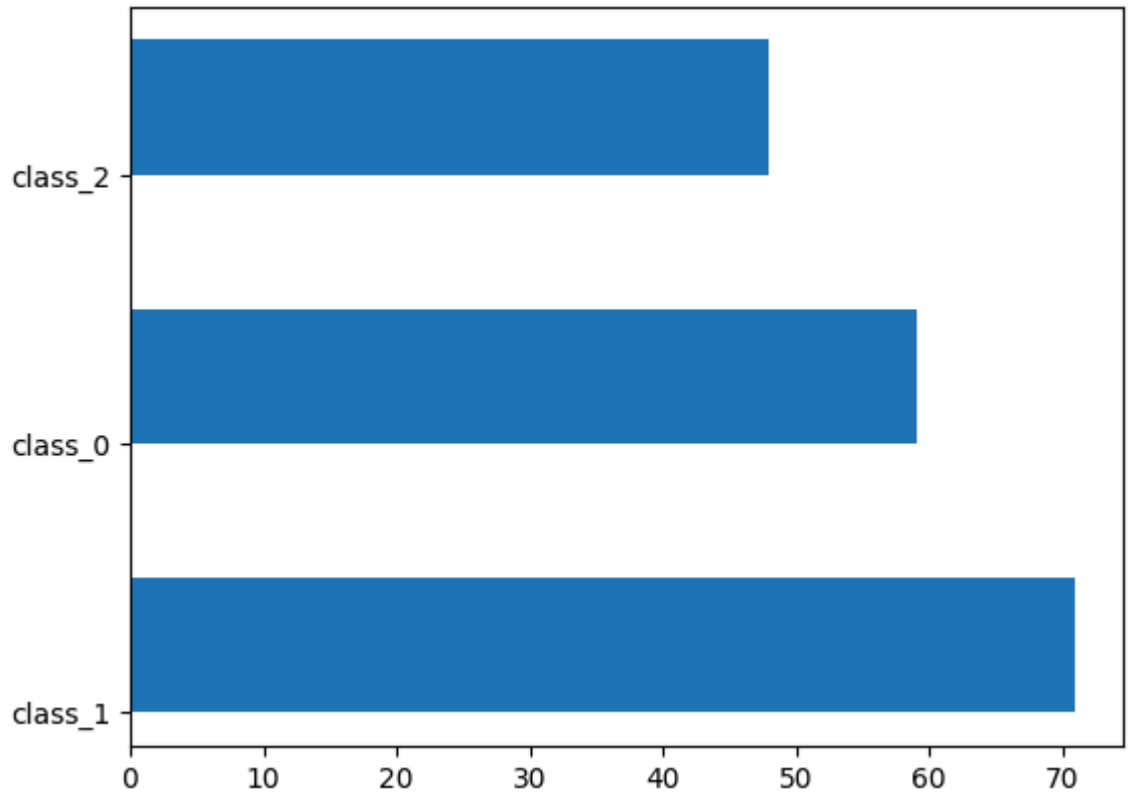
In [ ]:
```
#bar plot 그리기 (정방향)
plt.bar(wine_type.index, wine_type.values, width=0.5, bottom=None, align='ce
```

Out[ ]:
```
<BarContainer object of 3 artists>
```



In [ ]:
```
#bar plot 그리기 (옆방향)
plt.barh(wine_type.index, wine_type.values, height=0.5, left=None, align='ec
```

Out[ ]:
```
<BarContainer object of 3 artists>
```



🔬 **막대그래프 결과해석**

- class_1의 값이 가장 많음
- class_2의 값이 가장 적음
- class_1과 class_2간의 차이가 극단적이지 않음

✏️ **더 알아보기**

- 🤫 : 막대 그래프를 언제 사용하나요?
- 😀 : 각 범주의 값의 개수 차이를 비교하고 개수 차이가 극단적인지를 확인합니다. 주로 분류 문제에서 타겟의 분포가 차이가 많이나는가를 검증하기 위해 사용합니다.

4.1.2 히스토그램

히스토그램은 연속형 자료에 대한 도수분포표를 시각화하여 나타낸 것으로 서로 겹치지 않는 특정 구간에 따른 데이터의 빈도수를 표현합니다. 연속형 데이터이기 때문에 각 구간은 서로 연속되고, 막대는 서로 인접하여있습니다.

📌 **hist**

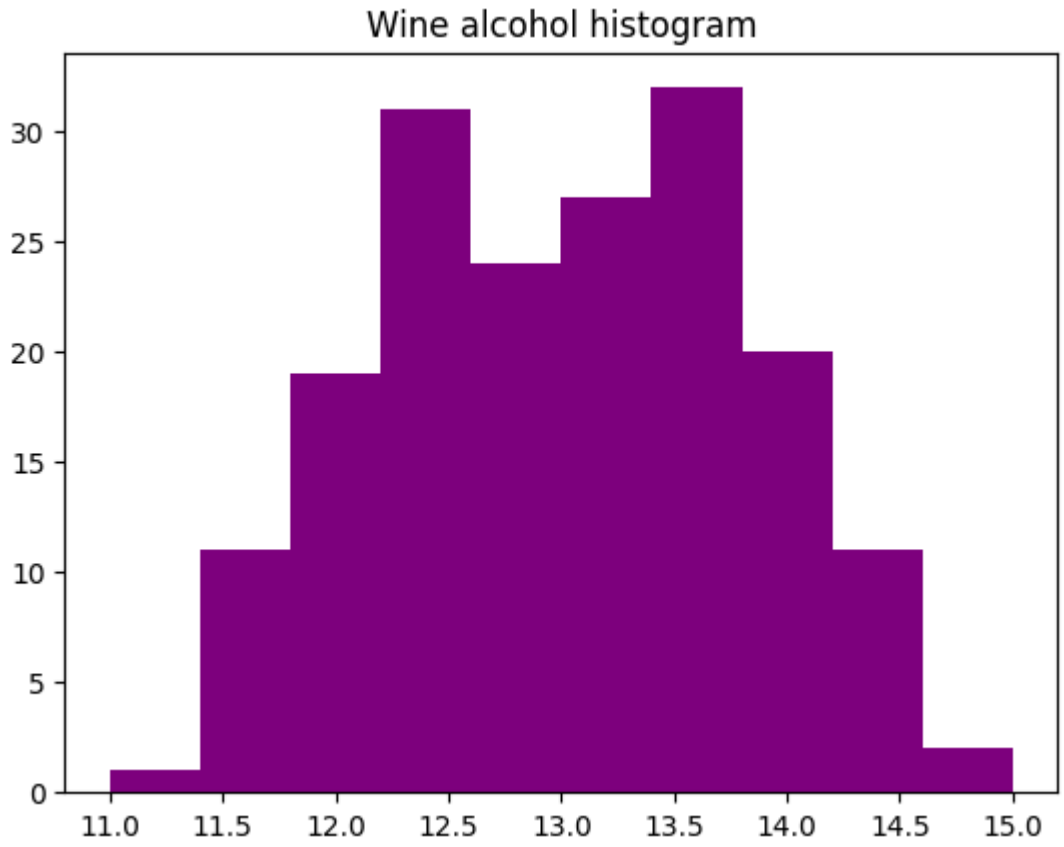
> plt.hist('column_name', bins=None, range=None, data=df)

- bins: 히스토그램의 구간의 개수 정의
- range: bin의 상한값과 하한값 형태로 선언, 예를들어 (x.min(), x.max()) 이렇게 설정 가능
- data: 시각화 대상이되는 데이터프레임

📜 **예제**

> 연속형 데이터인 와인 알콜농도에 대한 히스토그램 그리기

In [ ]:
```
plt.title('Wine alcohol histogram')
plt.hist('alcohol', bins=10,range=(11,15), color='purple', data=wine)
plt.show()
```



🔬 **히스토그램 결과해석**

- 가장 많은 데이터가 차지하는 알콜의 구간은 12~12.5도 사이
- 현재는 값이 특정 구간에 치중되어있음
- 데이터를 더 수집한다면 정규분포 모양을 나타낼 수 있음

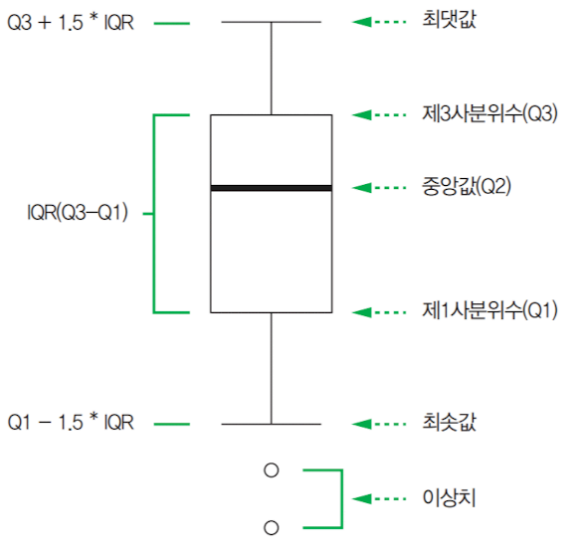
✏️ **더 알아보기**

- 🤫 : 히스토그램을 언제 사용하나요?
- 😀 : 연속형 데이터의 분포를 통해 특정 값에 치우쳐져 있는지, 극단적인 범위의 값(=이상치)이 있는지 확인할 수 있습니다.

4.2 상자 그림

Box Plot은 사분위수를 이용하여 수치형 변수의 값의 분포를 확인하는 그래프입니다. 상자에 수염같은 선이 붙어있다고 하여 상자 수염이라고도 불립니다. 상자의 크기, 중앙값 선의 위치, 수염의 길이를 통해 값으 분포와 대칭 정도, 이상치까지 한 컬럼의 값에 대한 다양한 정보를 한눈에 확인할 수 있습니다.

📍 **상자그래프 해석**



1. 상자

- 데이터 값의 50%가 해당함
- 상자의 아랫면과 윗면은 각 25%(1사분위수), 75%(3사분위수) 위치에 존재하는 값
- 상자 중앙의 두꺼운 선은 중앙값을 의미함
- 중앙값의 위치에 따라 값이 어디에 치우쳐져 있는지 확인할 수 있음

2. IQR

- 중앙 50%데이터가 퍼진 정도
- 3사분위수와 1사분위수의 차이로 구할 수 있음
- 통계학적으로 이상치를 판단하는 수식에 사용됨

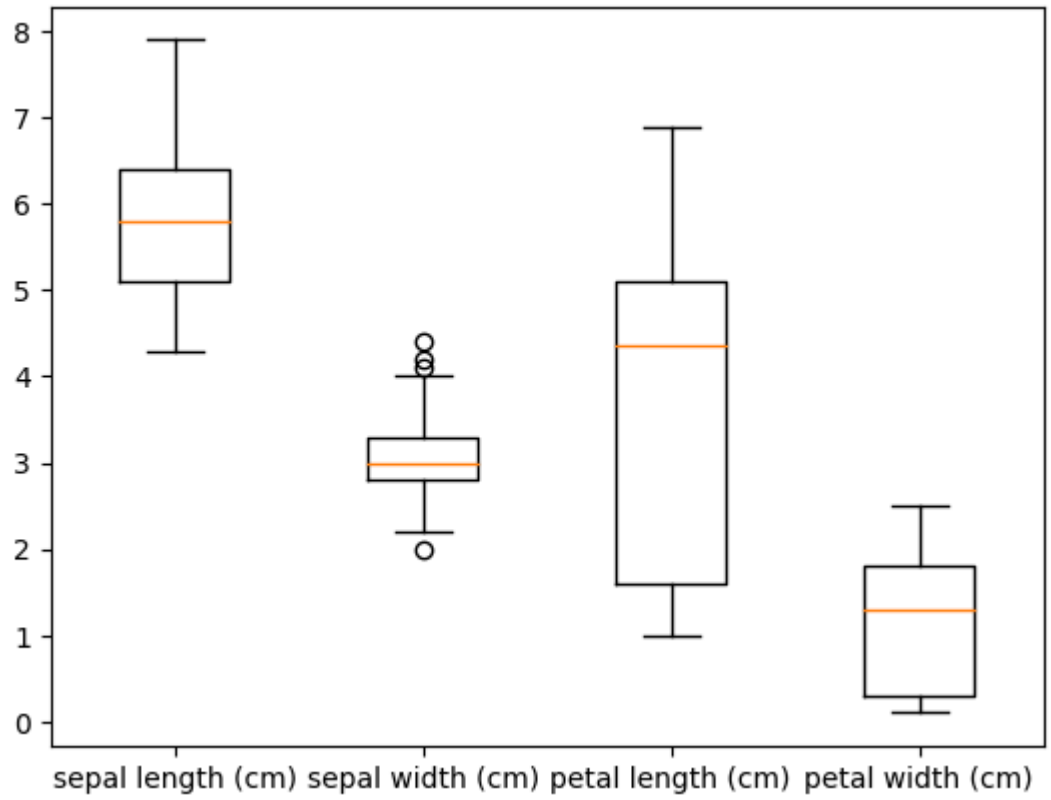
3. 수염

- 1.5 * IQR 범위인 Q1 - 1.5 * IQR 부터 Q3 + 1.5 * IQR 까지의 범위를 수염으로 지정 (1.5가 아닌 다른 값으로도 사용 가능)
- 수염을 벗어나는 값은 이상치로 판단

📜 **예제**

> iris 데이터의 컬럼별 상자 그림을 그려보기

In [ ]:
```
import pandas as pd
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()
iris = pd.DataFrame(iris.data, columns=iris.feature_names)

# boxplot 그리기
plt.boxplot(iris, labels=iris.columns)
plt.show()
```



📜 **예제**

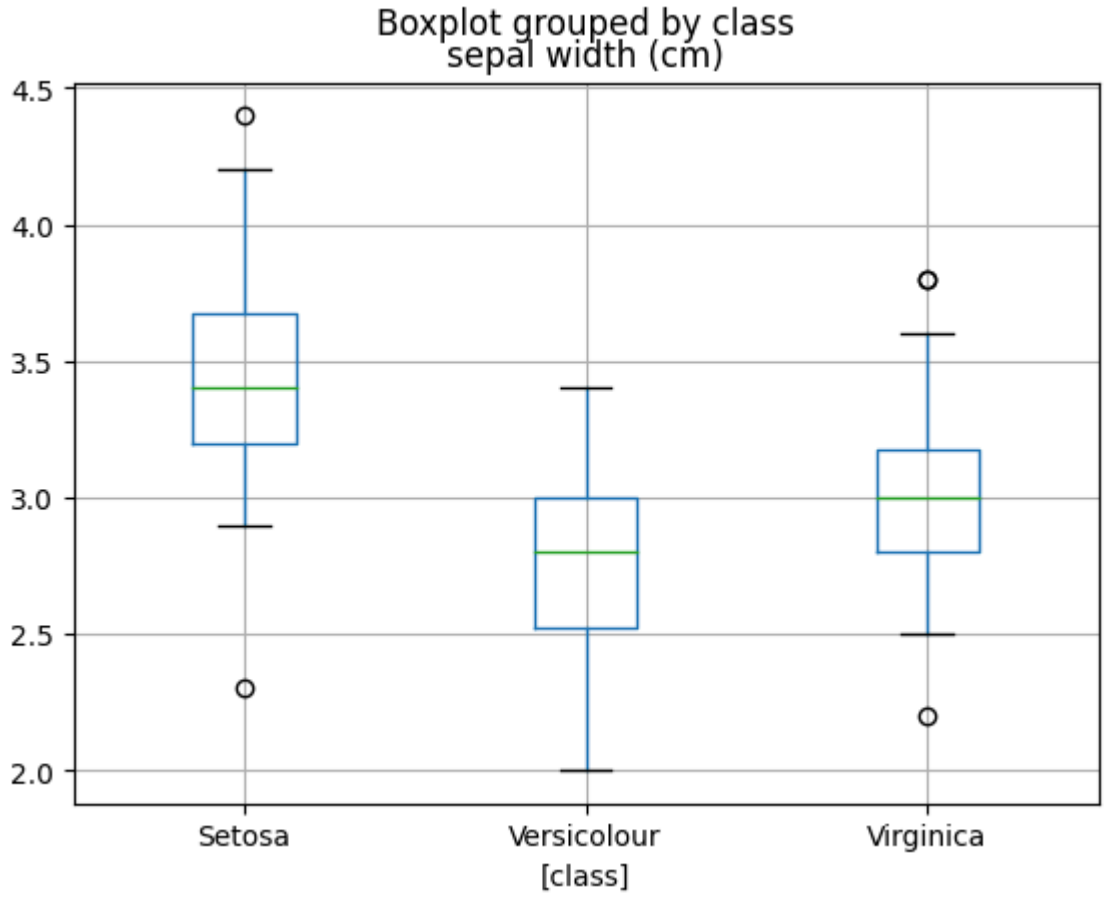
> iris 데이터의 target별 상자 그림 그리기

In [ ]:
```
iris['class'] = load_iris().target
iris['class'] = iris['class'].map({0:'Setosa', 1:'Versicolour', 2:'Virginica
iris.head()
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [ ]:
```
# target별 boxplot 그리기
iris[['sepal width (cm)', 'class']].boxplot(by='class')
plt.show()
```

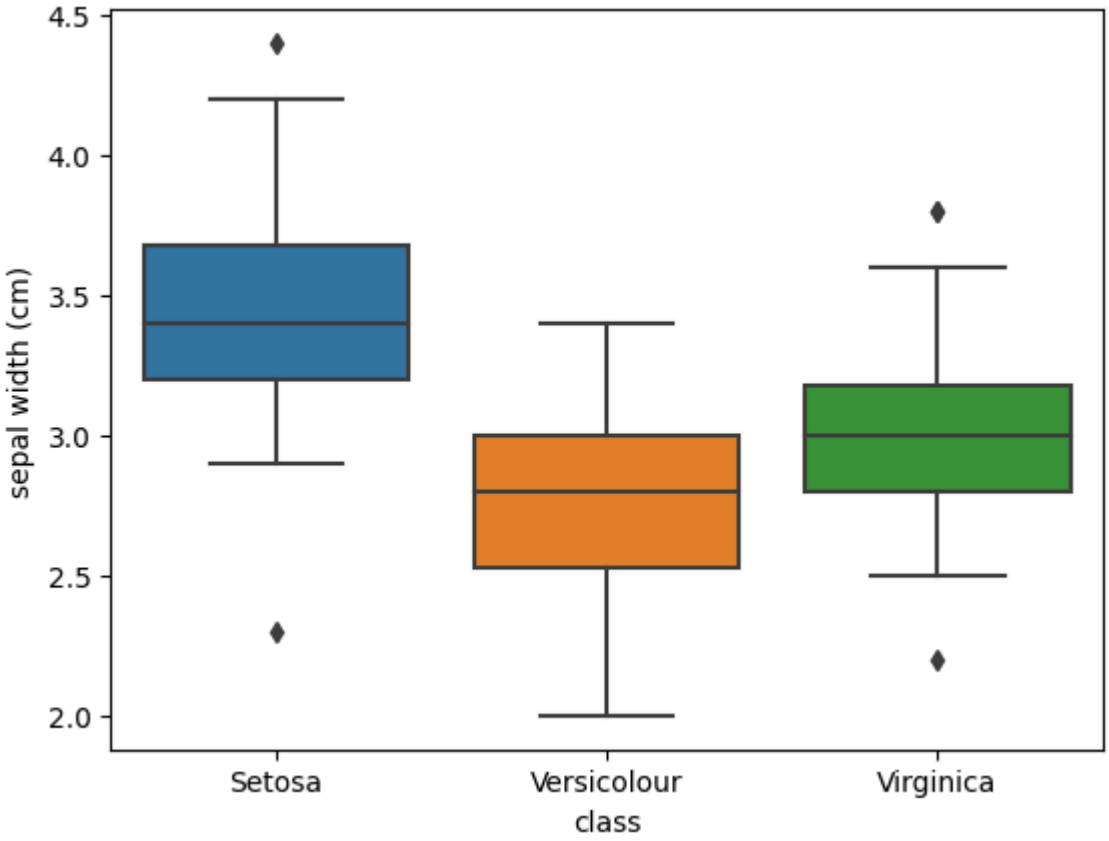


- by에는 category 컬럼이 들어감
- category 컬럼에 따른 다른 컬럼의 값 차이

📍 Seaborn

Seaborn은 Python에서 데이터 시각화를 쉽게 하고, 아름답고 통계적으로 유의미한 그래프를 그릴 수 있도록 도와주는 고수준 시각화 라이브러리입니다. matplotlib를 기반으로 하여 더 간단한 문법과 풍부한 스타일링 옵션을 제공합니다.

In [ ]:
```
import seaborn as sns
sns.boxplot(x="class", y="sepal width (cm)", data=iris)
plt.show()
```



🔥 **꿀팁**

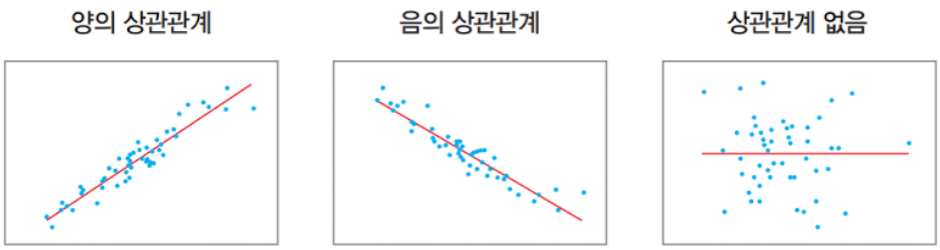
- 여러 컬럼을 확인할 시, 데이터의 스케일이 다르다면 시각화 정보 전달이 잘 안될 수 있음
- 앞서 데이터 요약통계 및 히스토그램을 통해 먼저 범위를 확인하고 수행하여 이상치를 판단

4.3 산점도

산점도는 두 개의 수치형 변수 각각의 분포와 함께 두 변수의 관계를 확인하는 가장 기본적인 그래프입니다. 두 개의 축을 가진 2차원 도표 안에 점들이 흩어져 있는 형태입니다.

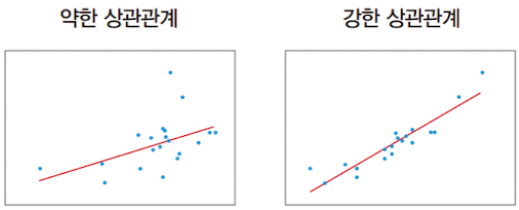
📍 **산점도 해석**

1. 관계의 유형



- 산점도는 두 변수의 관계 유형과 강도를 판단
- 관계의 유형은 점들이 흩어져 있는 모양을 보고 판단

2. 관계의 강도

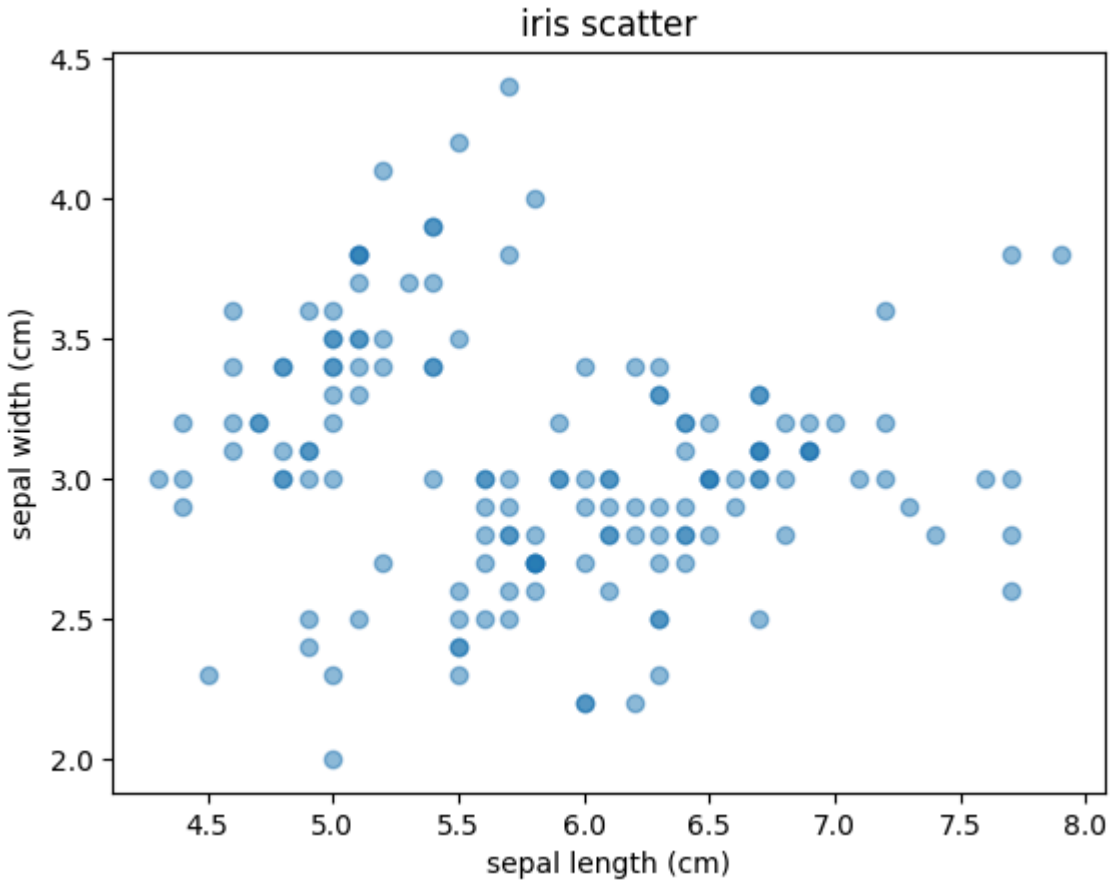


- 데이터가 적합선에 얼마나 가깝게 모여 있는지를 평가하여 두 변수 간 관계의 강도를 추정
- 산점도로는 관계의 해석을 수행하고, 상관관계 분석을 수행하여 강도를 수치화할 수 있음

📜 예제

> iris 데이터를 통해 산점도를 그려보기

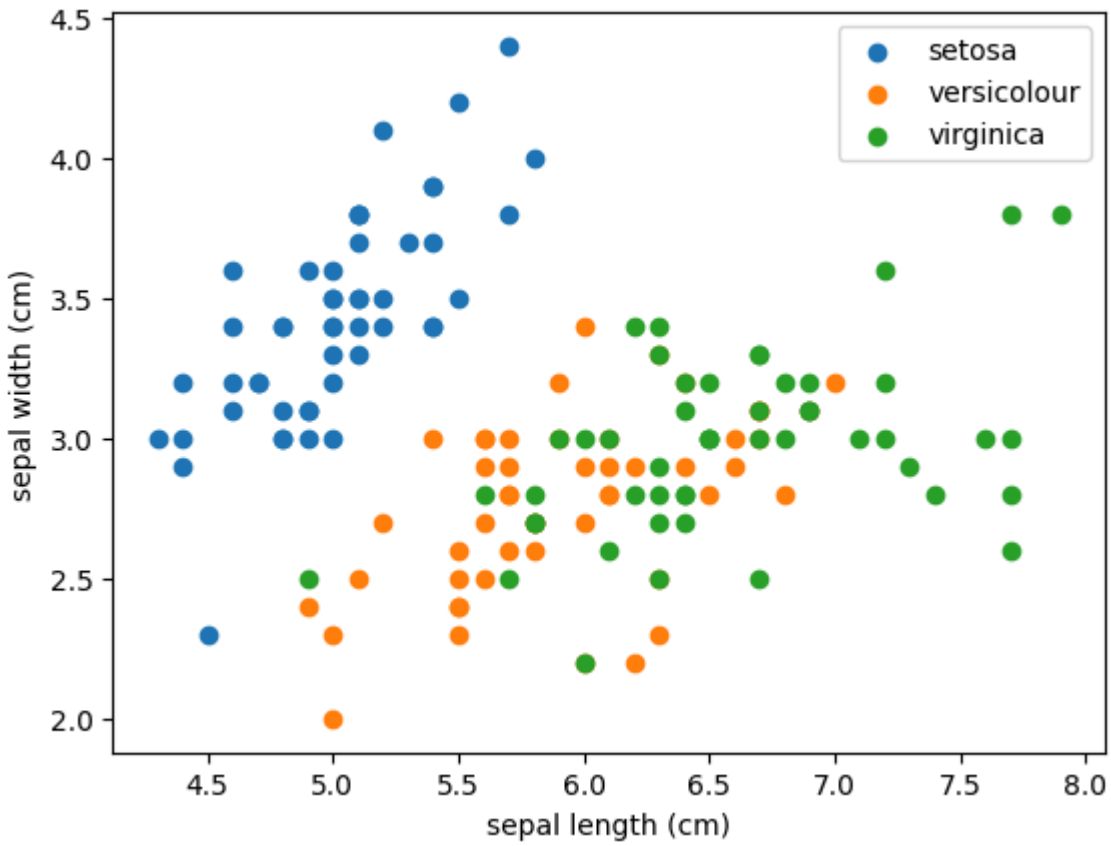
In [ ]:
```
import pandas as pd
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
iris = load_iris()
iris = pd.DataFrame(iris.data, columns=iris.feature_names)
iris["class"] = load_iris().target
iris["class"] = iris["class"].map({0:'Setosa', 1:'Versicolour', 2:'Virginica

# scatter plot 그리기
plt.title('iris scatter')
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.scatter (x = iris['sepal length (cm)'], y = iris['sepal width (cm)'],
alpha = 0.5)
plt.show()
```



📜 예제

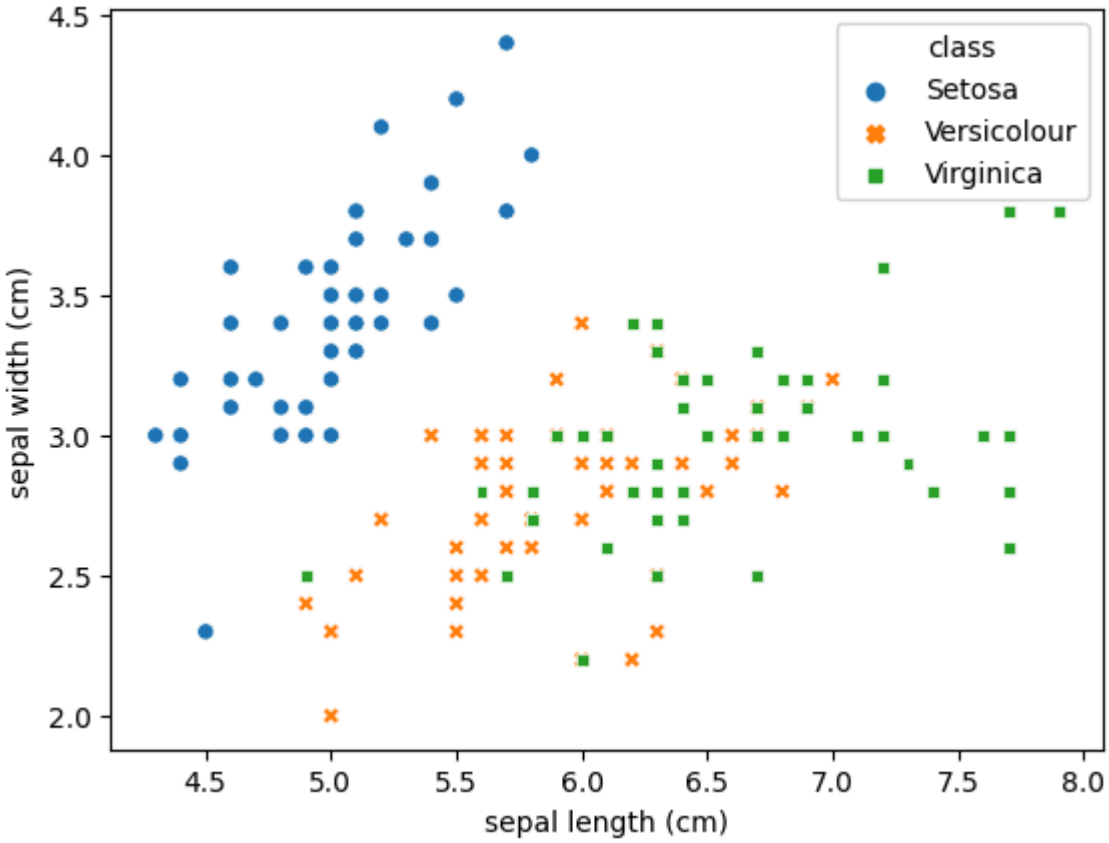
> iris 데이터를 통해 target 별 산점도를 그려보기

In [ ]:
```
import matplotlib.pyplot as plt

# iris 데이터셋을 각 클래스별로 분리
setosa = iris[iris['class'] == 'Setosa']
versicolor = iris[iris['class'] == 'Versicolour']
virginica = iris[iris['class'] == 'Virginica']

# 각 클래스별로 산점도 그리기
plt.scatter(setosa['sepal length (cm)'], setosa['sepal width (cm)'], label='
plt.scatter(versicolor['sepal length (cm)'], versicolor['sepal width (cm)'],
plt.scatter(virginica['sepal length (cm)'], virginica['sepal width (cm)'], l

# 범례 표시
plt.legend()

# 축 라벨 추가
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')

# 그래프 표시
plt.show()
```



📍 seaborn

In [ ]:
```
import seaborn as sns
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', data=iris, hue=
plt.show()
```



4.4 선 그래프

4.4.1 수평선, 수직선 그래프

- 수평선과 수직선은 그래프의 한계점, 평균값 등을 표시하기 위해 사용합니다.

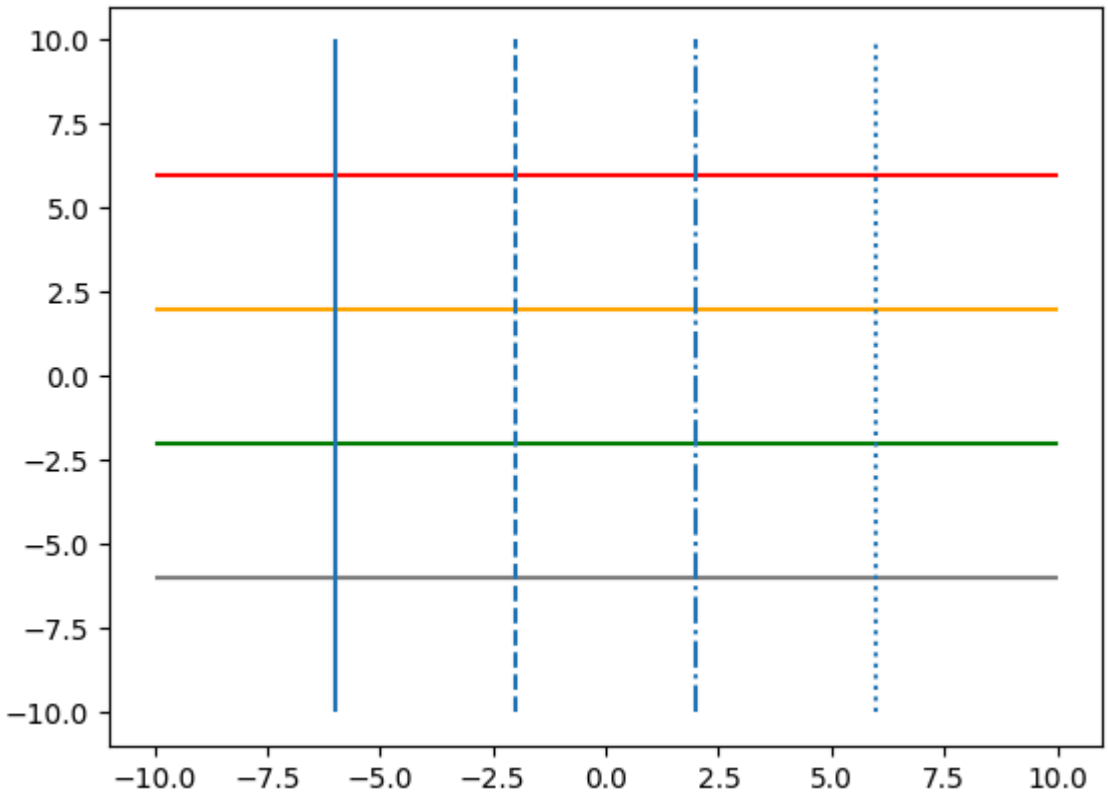
📌 **수평선-hliens**

> plt.hlines(y, xmin, xmax, colors=None, linestyles='solid')

📌 **수직선-vlines**

> plt.hlines(x, ymin, ymax, colors=None, linestyles='solid')

In [ ]:
```
import matplotlib.pyplot as plt
plt.hlines(-6, -10, 10, color='grey')
plt.hlines(-2, -10, 10, color='green')
plt.hlines(2, -10, 10, color='orange')
plt.hlines(6, -10, 10, color='red')
plt.vlines(-6, -10, 10, linestyles='solid')
plt.vlines(-2, -10, 10, linestyles='dashed')
plt.vlines(2, -10, 10, linestyles='dashdot')
plt.vlines(6, -10, 10, linestyles='dotted')
plt.show()
```



4.4.2 꺾은선 그래프(시계열)

- 꺾은선 그래프는 시간의 변화에 따라 값이 지속적으로 변화할 때 유용한 그래프
- 값을 점으로 표기하고 점들을 선으로 이어 나타내며 X축이 시간, Y축이 값을 의미

📜 **예제**

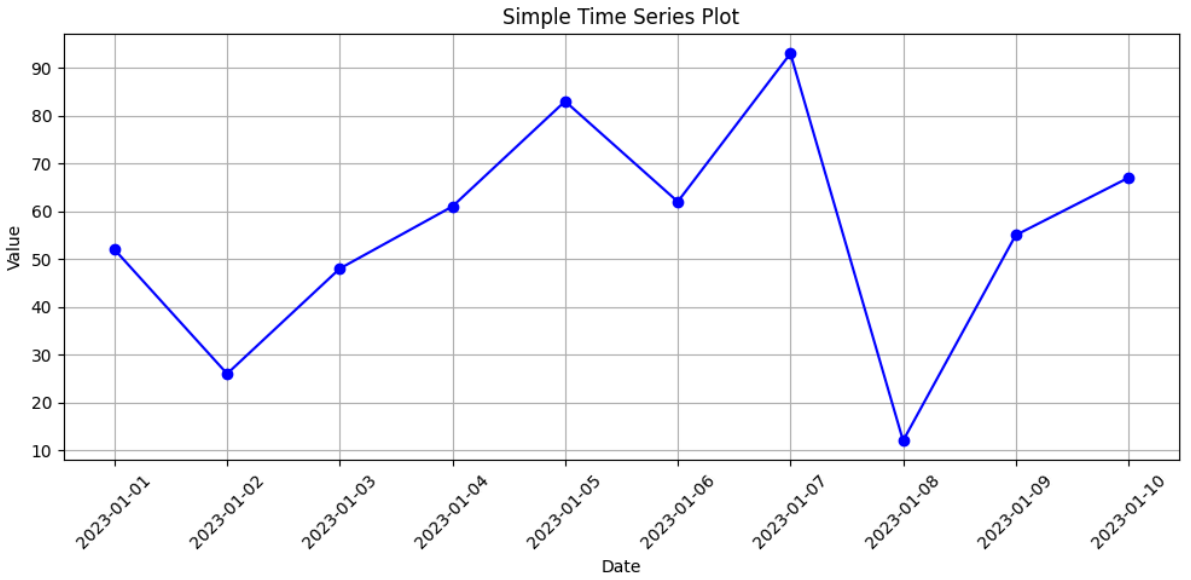
> 시계열 데이터를 생성하여 꺾은선 그래프로 나타내보기

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 데이터프레임 생성
dates = pd.date_range(start='2023-01-01', periods=10, freq='D')
values = np.random.randint(10, 100, size=(10,))
df = pd.DataFrame({'Date': dates, 'Value': values})

# 데이터프레임에서 'Date'를 인덱스로 설정
df.set_index('Date', inplace=True)
df
```

Out[ ]:

Date	Value
2023-01-01	52
2023-01-02	26
2023-01-03	48
2023-01-04	61
2023-01-05	83
2023-01-06	62
2023-01-07	93
2023-01-08	12
2023-01-09	55
2023-01-10	67

In [ ]:
```
# 시계열 그래프 그리기
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Value'], marker='o', linestyle='-', color='b')
plt.title('Simple Time Series Plot')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



📜 예제

> 다양한 범주의 시계열 데이터를 시각화

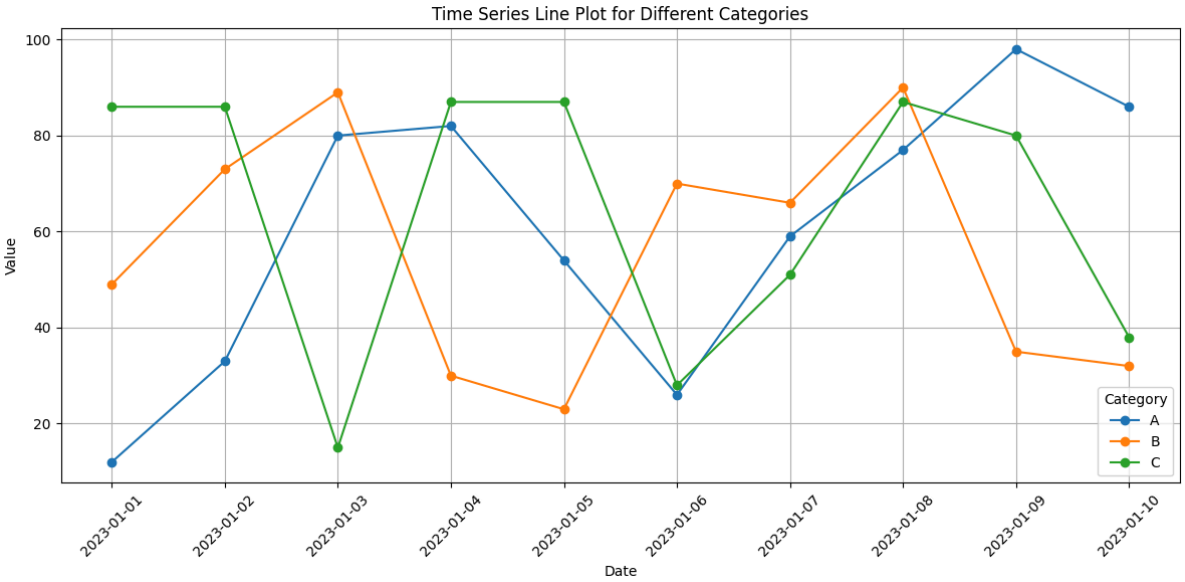
In [ ]:
```
# 날짜 범위 생성
dates = pd.date_range(start='2023-01-01', periods=10, freq='D')

# 카테고리 리스트 생성
categories = ['A', 'B', 'C']
```

```
# 데이터프레임 생성
data = {
    'Date': np.tile(dates, len(categories)),
    'Category': np.repeat(categories, len(dates)),
    'Value': np.random.randint(10, 100, size=len(dates) * len(categories))
}

df = pd.DataFrame(data)
df
```

Out[ ]:

	Date	Category	Value
0	2023-01-01	A	12
1	2023-01-02	A	33
2	2023-01-03	A	80
3	2023-01-04	A	82
4	2023-01-05	A	54
5	2023-01-06	A	26
6	2023-01-07	A	59
7	2023-01-08	A	77
8	2023-01-09	A	98
9	2023-01-10	A	86
10	2023-01-01	B	49
11	2023-01-02	B	73
12	2023-01-03	B	89
13	2023-01-04	B	30
14	2023-01-05	B	23
15	2023-01-06	B	70
16	2023-01-07	B	66
17	2023-01-08	B	90
18	2023-01-09	B	35
19	2023-01-10	B	32
20	2023-01-01	C	86
21	2023-01-02	C	86
22	2023-01-03	C	15
23	2023-01-04	C	87
24	2023-01-05	C	87
25	2023-01-06	C	28
26	2023-01-07	C	51
27	2023-01-08	C	87
28	2023-01-09	C	80
29	2023-01-10	C	38

In [ ]:
```
# 데이터프레임을 날짜와 카테고리별로 그룹화
grouped = df.groupby(['Date', 'Category']).sum().unstack()

# 시계열 꺾은선 그래프 그리기
plt.figure(figsize=(12, 6))

for category in categories:
    plt.plot(grouped.index, grouped['Value', category], marker='o', linestyl

# 그래프 설정
plt.title('Time Series Line Plot for Different Categories')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend(title='Category')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



4.5 상관관계 시각화

상관관계란 2개의 변수 간의 선형관계를 표현하는 통계적 측도입니다. 산점도를 통해 변수 사이의 데이터 분포를 살펴보고 관계의 유형과 강도를 판단해 보았습니다. 이번 상관관계 시각화에서는 이를 수치화하여 데이터 간 관계를 표현합니다.

4.5.1 산점도 행렬

📍 **산점도 행렬 해석 방법**

- 대각선의 히스토그램을 통해 이상치를 확인함
- 종속변수와 설명변수 간의 관계를 파악
- 종속변수가 수치형일 경우 직선 상관관계를 비교
- 종속변수가 범주형일 경우 잘 구분하는 변수를 파악
- 설명변수간의 직선 함수관계를 파악하여 다중공선성 문제를 파악 ☞ 설명변수 끼리의 강한 상관관계가 있는지

📍 **다중공선성**

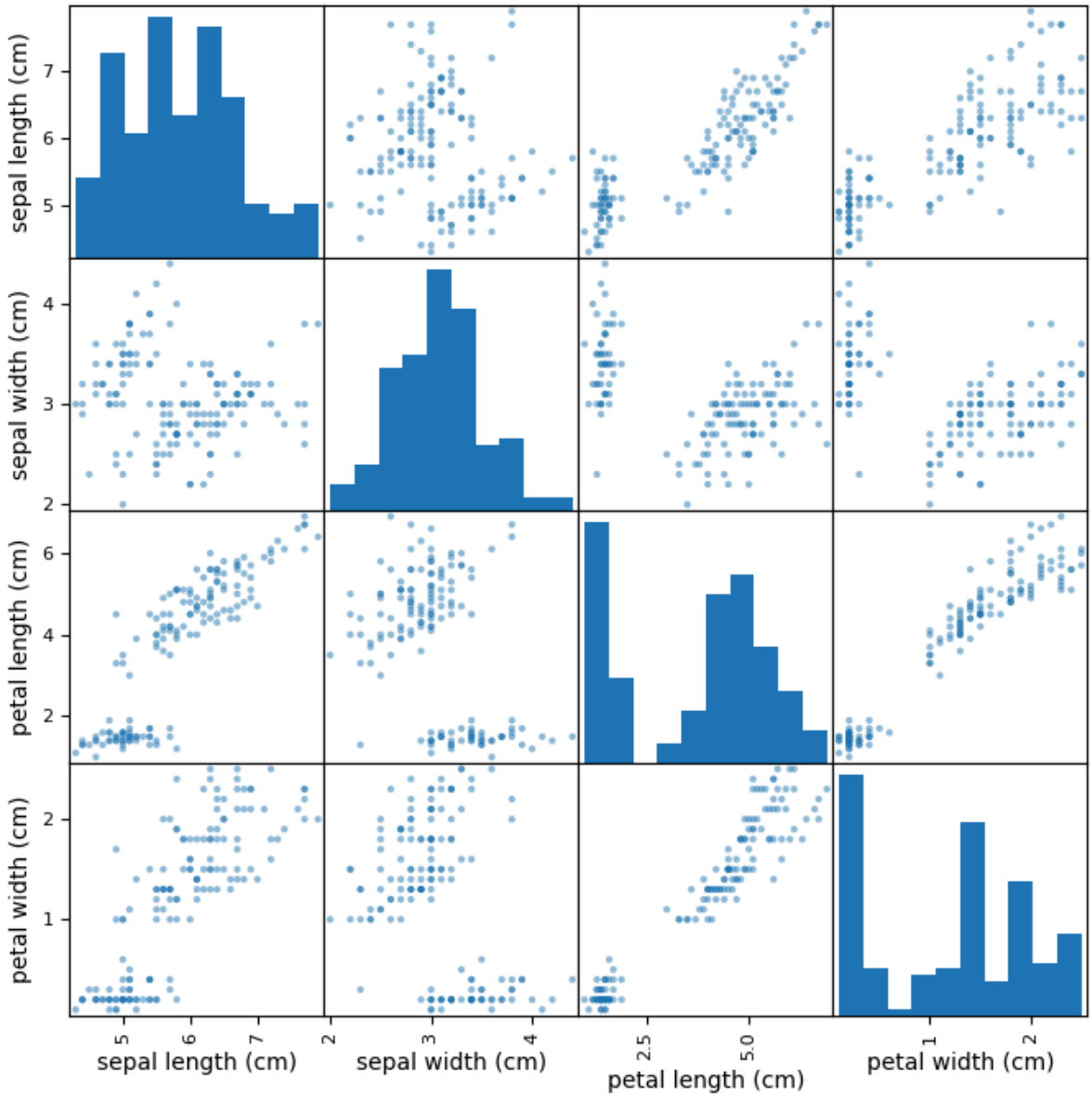
회귀분석에서 설명변수간의 강한 선형관계가 있을 때 발생하는 문제로 설명변수끼리 상호작용하거나 중복적으로 영향을 미치는 상황을 말합니다. 다중공선성 문제가 있을 시 결과해석 및 회귀계수 추정이 불안정해지는 문제를 야기합니다.

📜 예제

> iris 데이터에 대해 산점도 행렬로 상관관계 시각화를 수행해보기

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from sklearn.datasets import load_iris

# 데이터 불러오기 및 데이터프레임 생성
iris = load_iris()
iris = pd.DataFrame(iris.data, columns=iris.feature_names)
iris['Class'] = load_iris().target
iris['Class'] = iris['Class'].map({0:'Setosa', 1:'Versicolour', 2:'Virginica
```

In [ ]:
```
# 산점도 행렬 그리기
scatter_matrix(iris, alpha = 0.5, figsize = (8, 8), diagonal = 'hist')
plt.show()
```



🔬 산점도 행렬 결과해석

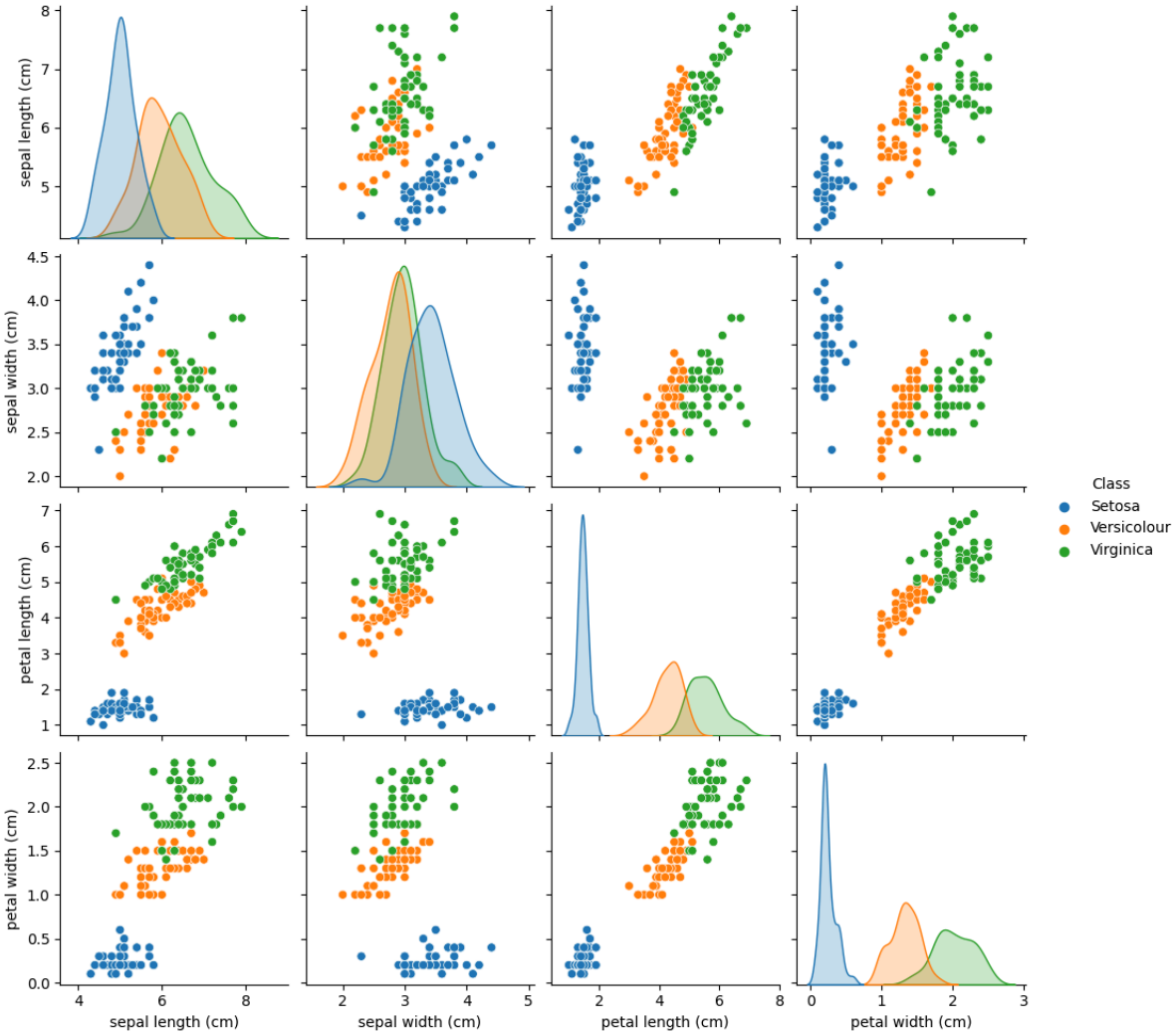
- 설명변수 sepal length와 petal length 간에 양의 상관관계가 있음 ☞ 다중공선성 확인 필요
- 설명변수 petal width와 petal length 간에 양의 상관관계가 있음 ☞ 다중공선성 확인 필요

📜 예제

> iris 데이터의 target별 행렬로 상관관계 시각화를 수행해보기

In [ ]:
```
import seaborn as sns
sns.pairplot(iris, hue = 'Class')
plt.show()
```

```
/opt/homebrew/Caskroom/miniconda/base/envs/ml/lib/python3.9/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



4.5.2 상관계수 행렬 시각화

상관계수 행렬 역시 다수의 변수 간 상관관계를 파악하거나 독립변수 간 다중공선성을 파악하고자 할 때 사용하는 분석 기법입니다. 단, 산점도 행렬보다 수치로 결과를 나타내어 직관적으로 다중공선성 문제를 확인할 수 있습니다.

📍 상관관계 해석하기

- 상관관계는 -1~1 사이의 숫자 값으로 출력됨

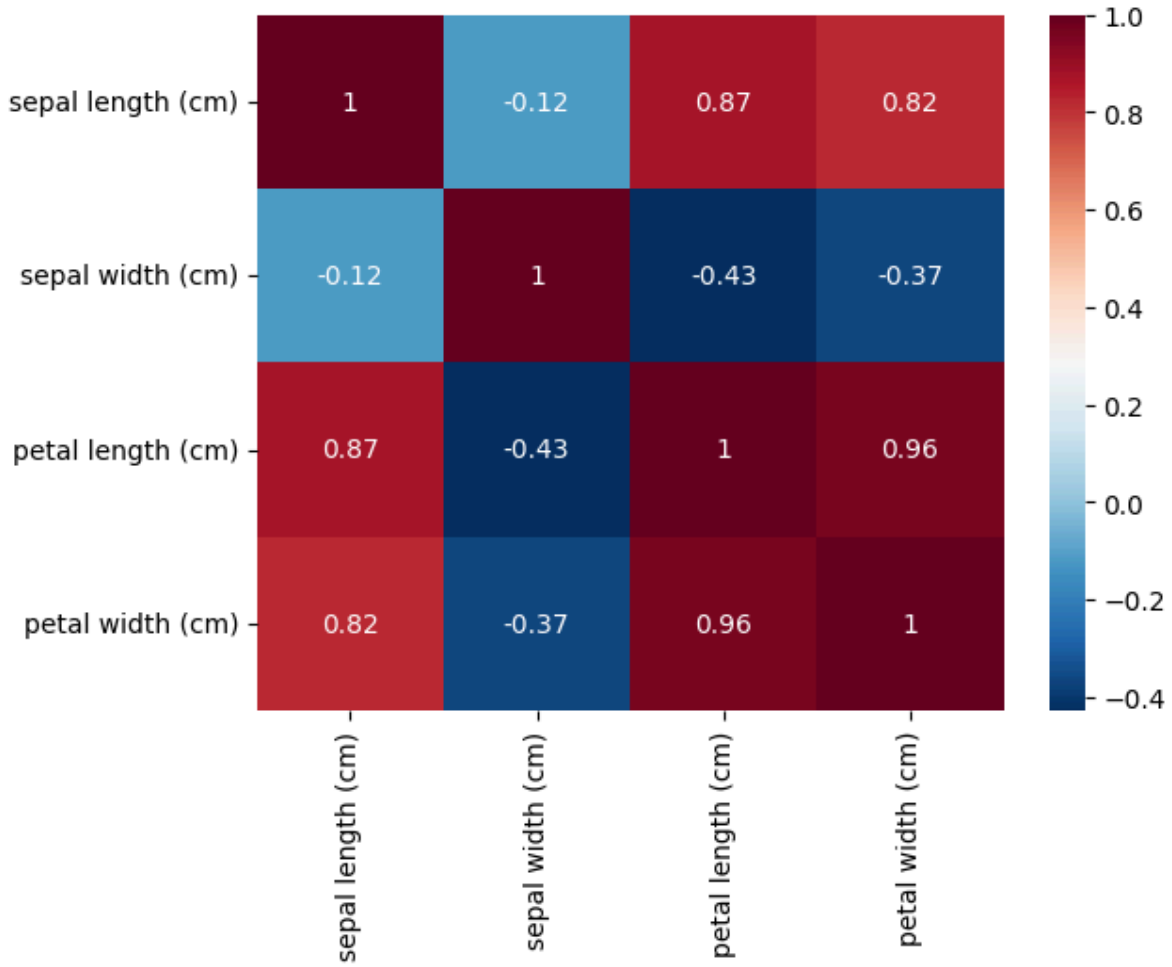
- 아래 상관관계 판단 기준 표에 따라 해석

범위	설명	범위	설명
r <= -0.8	강한 음의 상관관계	0.8 <= r	강한 양의 상관관계
-0.8 < r <= -0.6	음의 상관관계	0.6 <= r < 0.8	양의 상관관계
-0.6 < r <= -0.4	약한 음의 상관관계	0.4 <= r < 0.6	약한 양의 상관관계
-0.4 < r <= 0	거의 상관 없음	0 <= r < 0.4	거의 상관 없음

📜 예제

> iris의 변수간 상관관계 파악을 위해 상관계수 행렬을 시각화해보기

In [ ]:
```
iris_corr = iris.drop(columns='Class').corr(method='pearson')
sns.heatmap (iris_corr, xticklabels = iris_corr.columns,
             yticklabels = iris_corr.columns, cmap = 'RdBu_r', annot = True)
plt.show()
```



🔬 상관계수 행렬 결과해석

- 설명변수 sepal length와 petal length 간에 양의 상관관계가 있음 ☞ 다중공선성 확인 필요
- 설명변수 petal width와 petal length 간에 양의 상관관계가 있음 ☞ 다중공선성 확인 필요