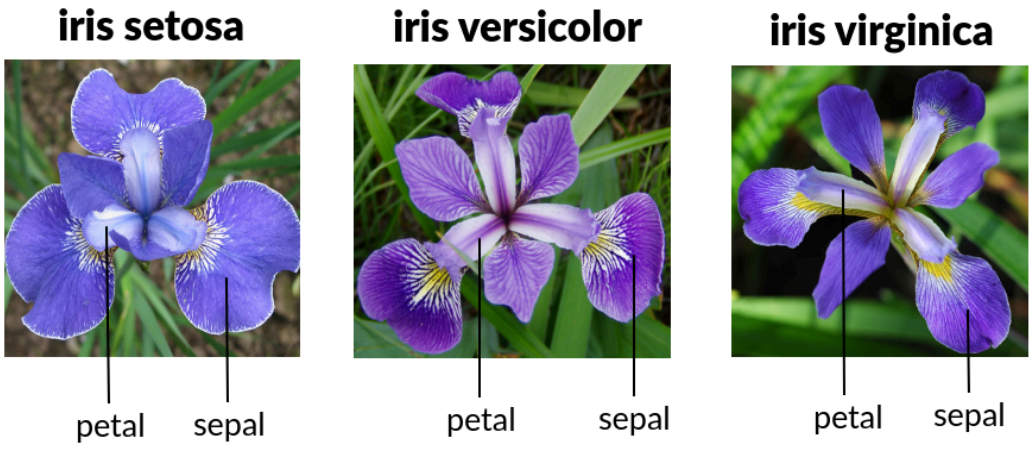
Machine Learning with Python

Life is too short, You need Python

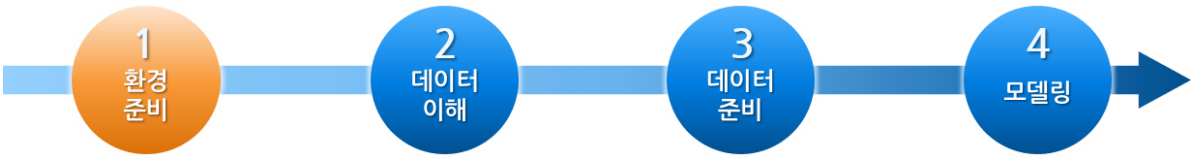
실습 내용

- 머신러닝 모델링을 위한 코딩은 무조건 할 수 있어야 합니다.
- 코딩 내용을 자세히 알지 못해도 무작정 코딩을 진행해봅니다.
- Iris 데이터를 대상으로 모델링해서 붓꽃 품종을 예측해 봅니다.
- DecisionTree 알고리즘을 사용합니다.



1.환경 준비

- 기본 **라이브러리**와 대상 **데이터**를 가져와 이후 과정을 준비합니다.



In [1]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings(action='ignore')
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/Jangrae/csv/master/iris.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 **충분히 이해**할 수 있도록 다양한 **탐색** 과정을 수행합니다.

In [3]:
```
# 상위 몇 개 행 확인
data.head()
```

Out[3]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
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

데이터 정보

- Sepal.Length: 꽃받침의 길이
- Sepal.Width: 꽃받침의 너비
- Petal.Length: 꽃잎의 길이
- Petal.Width: 꽃잎의 너비

In [4]:
```
# 하위 몇 개 행 확인
data.tail()
```

Out[4]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [5]:
```
# 변수 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Sepal.Length  150 non-null    float64
 1   Sepal.Width   150 non-null    float64
 2   Petal.Length  150 non-null    float64
 3   Petal.Width   150 non-null    float64
 4   Species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:
```
# 기술통계 확인
data.describe().T
```

Out[6]:

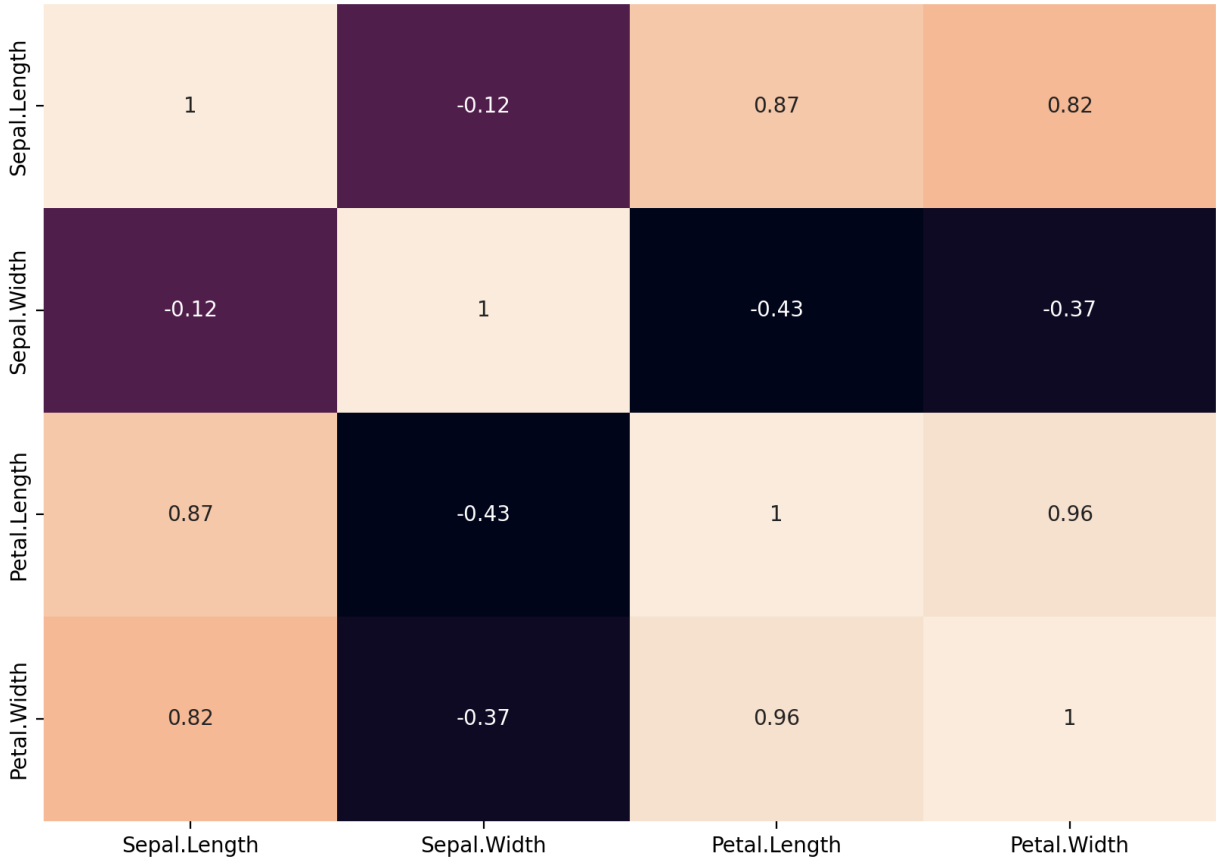
	count	mean	std	min	25%	50%	75%	max
Sepal.Length	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
Sepal.Width	150.0	3.057333	0.435866	2.0	2.8	3.00	3.3	4.4
Petal.Length	150.0	3.758000	1.765298	1.0	1.6	4.35	5.1	6.9
Petal.Width	150.0	1.199333	0.762238	0.1	0.3	1.30	1.8	2.5

In [13]:
```
# 상관관계 확인
data.corr(numeric_only=True)
```

Out[13]:

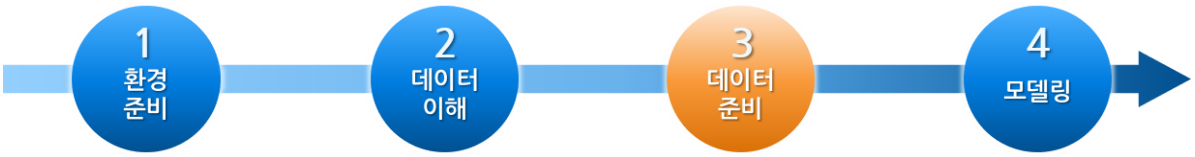
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.000000	-0.117570	0.871754	0.817941
Sepal.Width	-0.117570	1.000000	-0.428440	-0.366126
Petal.Length	0.871754	-0.428440	1.000000	0.962865
Petal.Width	0.817941	-0.366126	0.962865	1.000000

In [20]:
```
# 상관관계 시각화
plt.figure(figsize=(10, 7))
sns.heatmap(data.corr(numeric_only=True), annot=True, cbar=False)
plt.show()
```



3.데이터 준비

- **전처리** 과정을 통해 머신러닝 알고리즘에 사용할 수 있는 형태의 데이터를 준비합니다.



1) x, y 분리

- 우선 target 변수를 명확히 지정합니다.
- target을 제외한 나머지 변수들 데이터는 x로 선언합니다.
- target 변수 데이터는 y로 선언합니다.
- 이 결과로 만들어진 x는 데이터프레임, y는 시리즈가 됩니다.
- 이후 모든 작업은 x, y를 대상으로 진행합니다.

In [21]:
```
# target 확인
target = 'Species'

# 데이터 분리
x = data.drop(target, axis=1)
y = data.loc[:, target]
```

2) 학습용, 평가용 데이터 분리

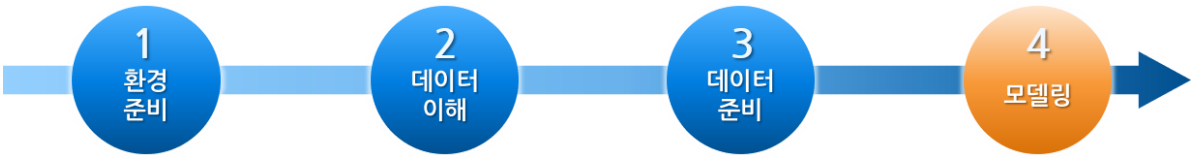
- 학습용, 평가용 데이터를 적절한 비율로 분리합니다.
- 반복 실행 시 동일한 결과를 얻기 위해 random_state 옵션을 지정합니다.

In [29]:
```
# 모듈 불러오기
from sklearn.model_selection import train_test_split

# 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

4.모델링

- 본격적으로 모델을 **선언**하고 **학습**하고 **평가**하는 과정을 진행합니다.
- 우선 **회귀** 문제인지 **분류** 문제인지 명확히 구분합니다.



- 회귀 문제 인가요? 분류 문제인가요?
- 회귀인지 분류인지에 따라 사용할 알고리즘과 평가 방법이 달라집니다.
- 우선 다음 알고리즘과 평가 방법을 사용합니다.
  - 알고리즘: DecisionTreeClassifier
  - 평가방법: accuracy_score

In [30]:
```
# 1단계: 불러오기 # sklearn 은 target(종속변수)은 문자열 허용 feature(독립변수)는 안됨
from sklearn.tree import DecisionTreeClassifier # 의사 결정 나무
from sklearn.metrics import accuracy_score
```

In [31]:
```
# 2단계: 선언하기
model = DecisionTreeClassifier()
```

In [32]:
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[32]:
```
DecisionTreeClassifier()
```

In [33]:
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

In [35]:
```
print(y_test.values[:10])
print(y_pred[:10])
```

```
['setosa' 'versicolor' 'versicolor' 'setosa' 'virginica' 'versicolor'
 'virginica' 'setosa' 'setosa' 'virginica']
['setosa' 'versicolor' 'versicolor' 'setosa' 'virginica' 'versicolor'
 'virginica' 'setosa' 'setosa' 'virginica']
```

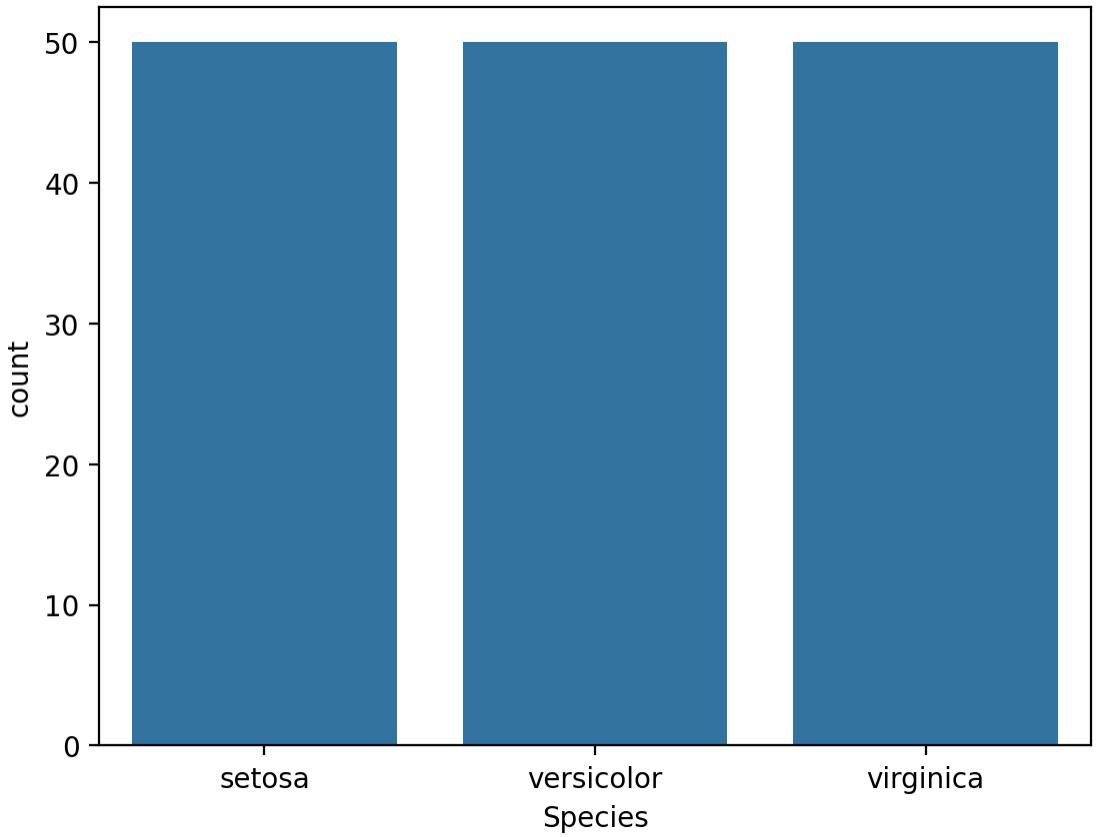
In [38]:
```
# 5단계 평가하기
print('accuracy_score:', accuracy_score(y_test, y_pred)) # 95% 정확도 높네...
```

```
accuracy_score: 0.9555555555555556
```

In [36]:
```
y_train.mode()
```

Out[36]:
```
0    virginica
Name: Species, dtype: object
```

In [39]:
```
sns.countplot(x=data['Species'])
plt.show()
```



In [40]:
```
y_train.value_counts() # 빈도수는 비슷
```

Out[40]:
```
Species
virginica     37
setosa        36
versicolor    32
Name: count, dtype: int64
```

In [ ]: