2021 2학기

인공지능응용프로그래밍
텐서플로기반딥러닝프로그래밍

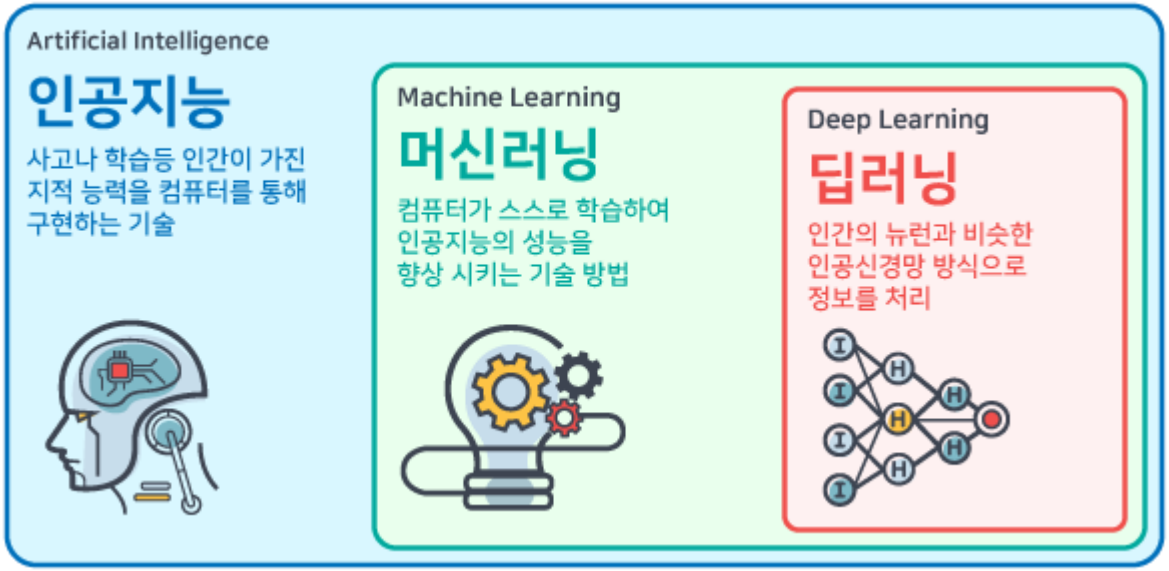
컴퓨터정보공학과
강 환수 교수

머신러닝 개요와 구현

컴퓨터정보공학과
강 환수 교수

머신러닝

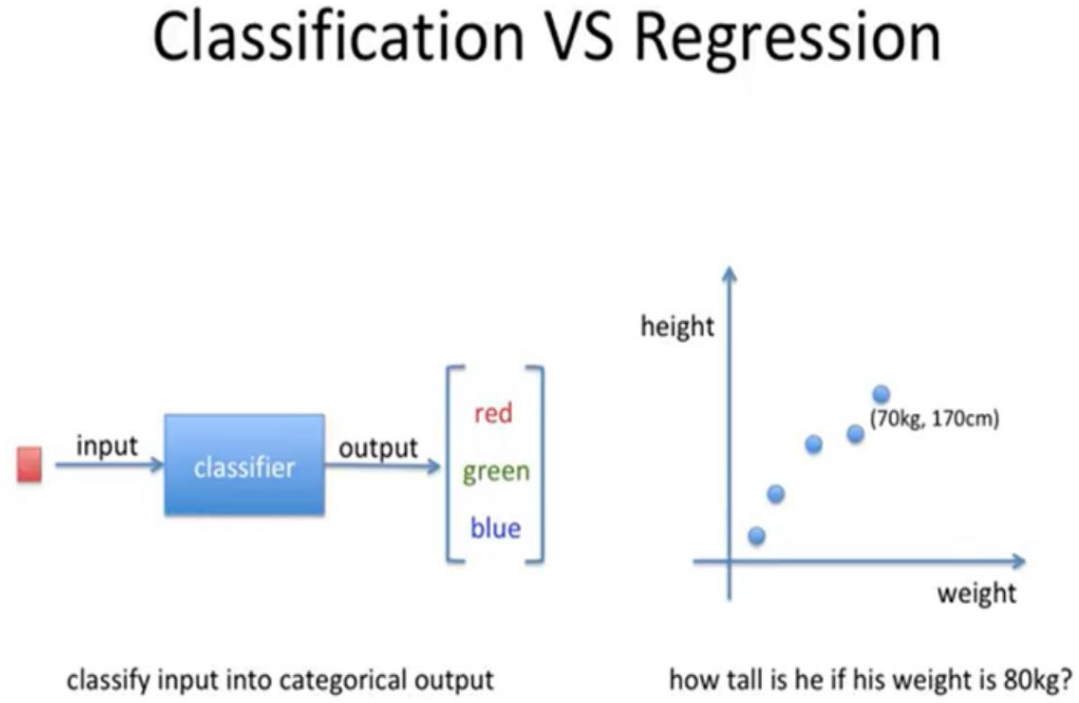
- **회귀와 분류**
- **회귀**
  - 선형 회귀
- **분류**
  - KNN
  - SVM



회귀와 분류 (regression and classification)

회귀(regresson)와 분류(classification)

- **회귀 모델**
  - 연속적인 값을 예측
    - **캘리포니아의 주택 가격이 얼마인가요?**
    - **사용자가 이 광고를 클릭할 확률이 얼마인가요?**
- **분류 모델**
  - 불연속적인 값을 예측
    - **주어진 이메일 메시지가 스팸인가요, 스팸이 아닌가요?**
    - **이 이미지가 강아지, 고양이 또는 햄스터의 이미지인가요?**

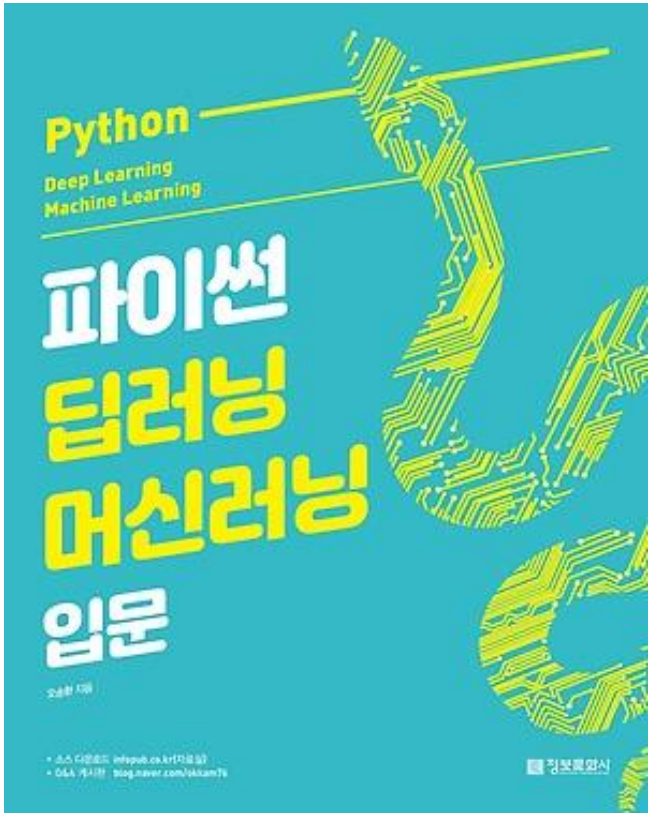


회귀의 어원

- **회귀 분석(regression analysis)**
  - 관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한 뒤 적합도를 측정해 내는 분석 방법
  - 회귀분석은 시간에 따라 변화하는 데이터나 어떤 영향, 가설적 실험, 인과 관계의 모델링 등의 통계적 예측에 이용
- **회귀(영어: regress 리그레스[*])의 원래 의미**
  - 옛날 상태로 돌아가는 것을 의미
  - 영국의 유전학자 프랜시스 골턴은 "평균으로의 회귀(regression to the mean)"
    - **부모의 키와 아이들의 키 사이의 연관 관계를 연구하면서 부모와 자녀의 키 사이에는 선형적인 관계가 있고 키가 커지거나 작아지는 것보다는 전체 키 평균으로 돌아가려는 경향이 있다는 가설을 세웠으며 이를 분석하는 방법을 "회귀분석"이라고 함**
    - **이러한 경험적 연구 이후, 칼 피어슨은 아버지와 아들의 키를 조사한 결과를 바탕으로 함수 관계를 도출하여 회귀분석 이론을 수학적으로 정립**

참고도서: 파이썬 머신러닝 딥러닝 입문

- http://www.infopub.co.kr/index.asp
  - 자료실
    - 749

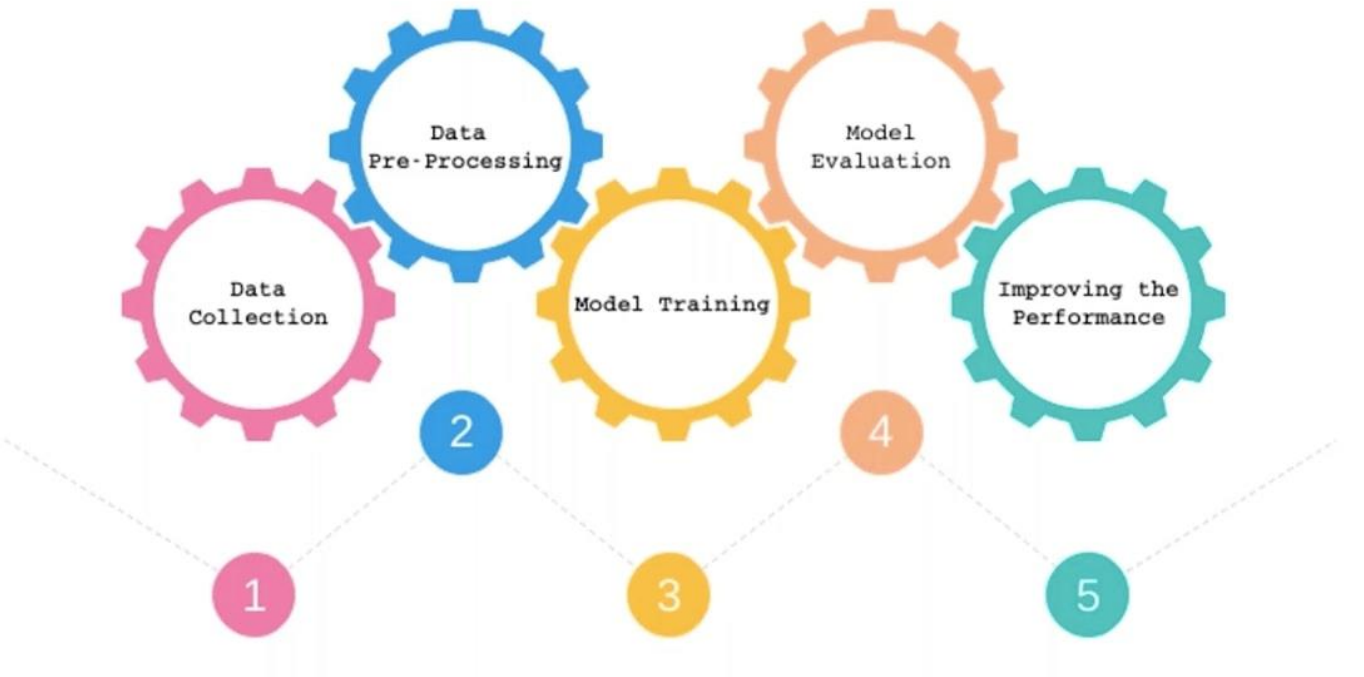


머신러닝 절차와 구현

머신 러닝 절차

General Machine learning algorithm flow

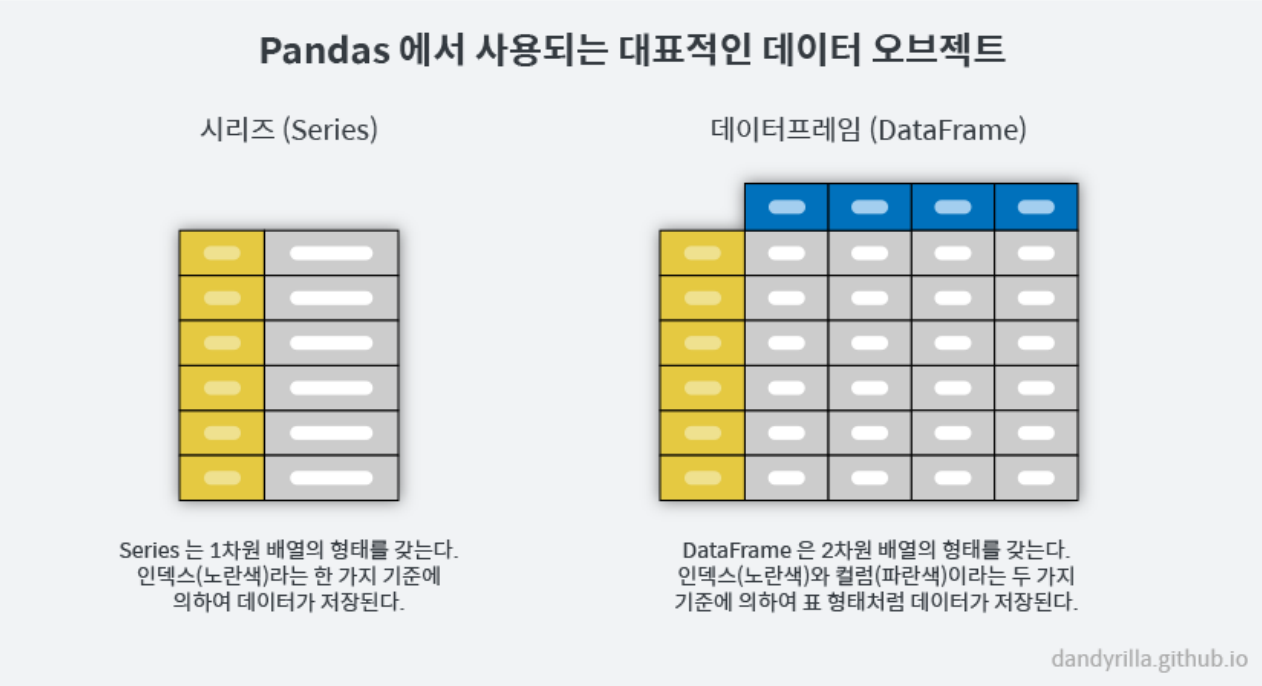
altud



Pandas 개요

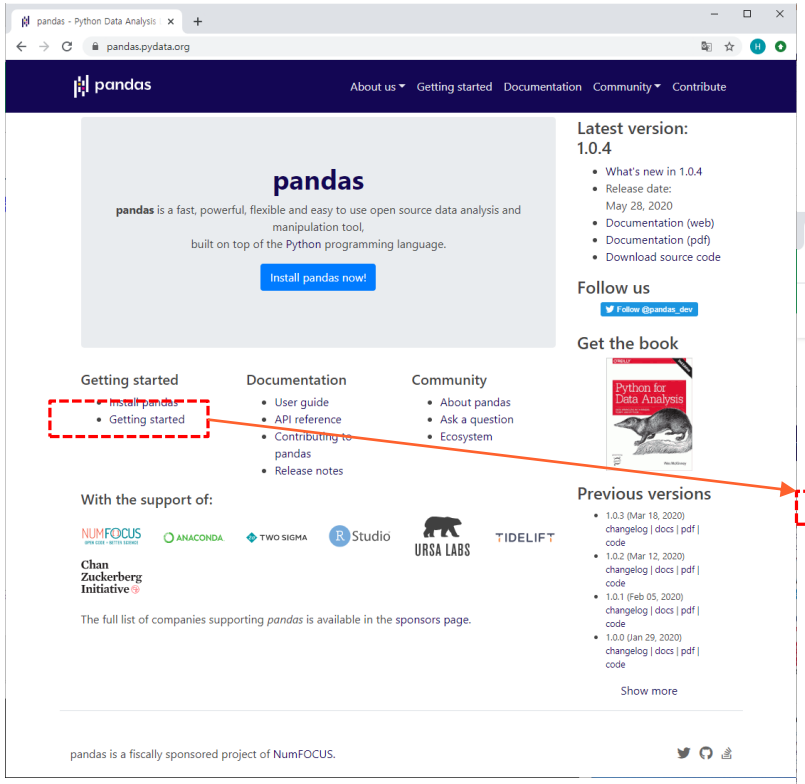
판다스

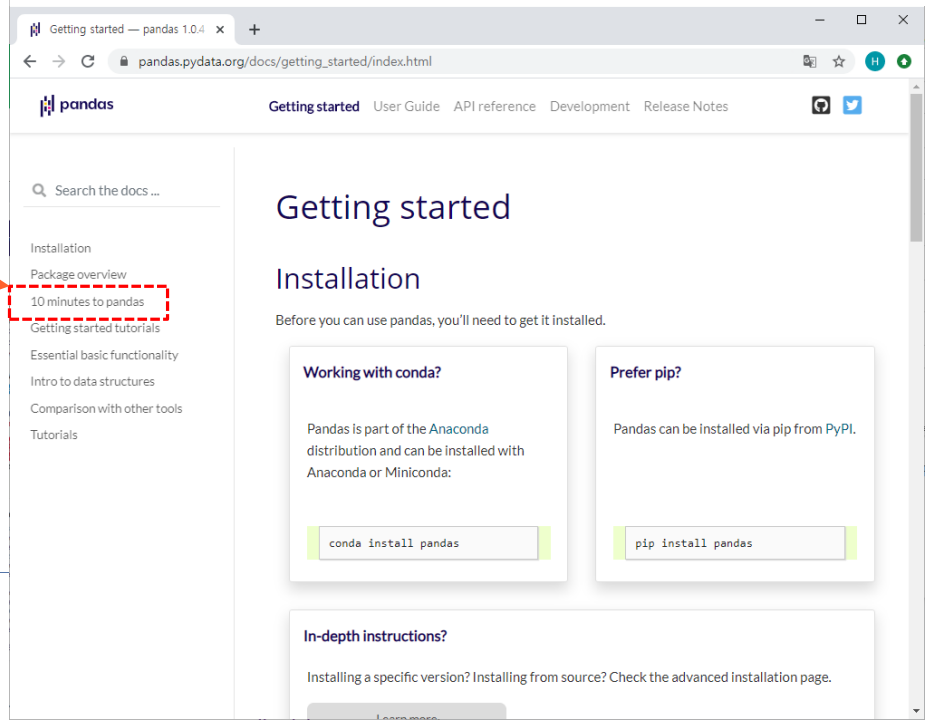
- **개요**
  - 표 형식의 데이터나 다양한 형태의 테이블를 처리하기 위한 라이브러리
    - **2010년부터, 약 800여명의 기여자가 활동**
  - 주 자료 구조
    - **시리지와 데이터프레임**



판다스 홈페이지

- pandas.pydata.org





DataFrame 개요

- **pd.DataFrame**
  - R의 dataframe 데이터 타입을 참고하여 만든 것이 바로 pandas DataFrame
  - 테이블 형태의 자료
    - **행과 열을 인덱스(index)와 칼럼(columns)으로 구분**

	A	B	C	D	E	F	G	H	I
1	Order Date	OrderID	Salesperson	UK Units	UK Order Amt	USA Units	USA Order Amt	Total Units	Total Order Amt
2	1/01/2011	10392	Fuller			13	1440	13	1440
3	2/01/2011	10397	Gloucester	17	716.72			17	716.72
4	2/01/2011	10771	Bromley	18	344			18	344
5	3/01/2011	10393	Finchley			16	2556.95	16	2556.95
6	3/01/2011	10394	Finchley			10	442	10	442
7	3/01/2011	10395	Gillingham	9	2122.92			9	2122.92
8	6/01/2011	10396	Finchley			7	1903.8	7	1903.8
9	8/01/2011	10399	Callahan			17	1765.6	17	1765.6
10	8/01/2011	10404	Fuller			7	1591.25	7	1591.25
11	9/01/2011	10398	Fuller			11	2505.6	11	2505.6
12	9/01/2011	10403	Coghill	18	855.01			18	855.01
13	10/01/2011	10401	Finchley			7	3868.6	7	3868.6

DataFrame 이해

- 여러 시리즈의 모임
- 인덱스와 칼럼

Series + Series = DataFrame

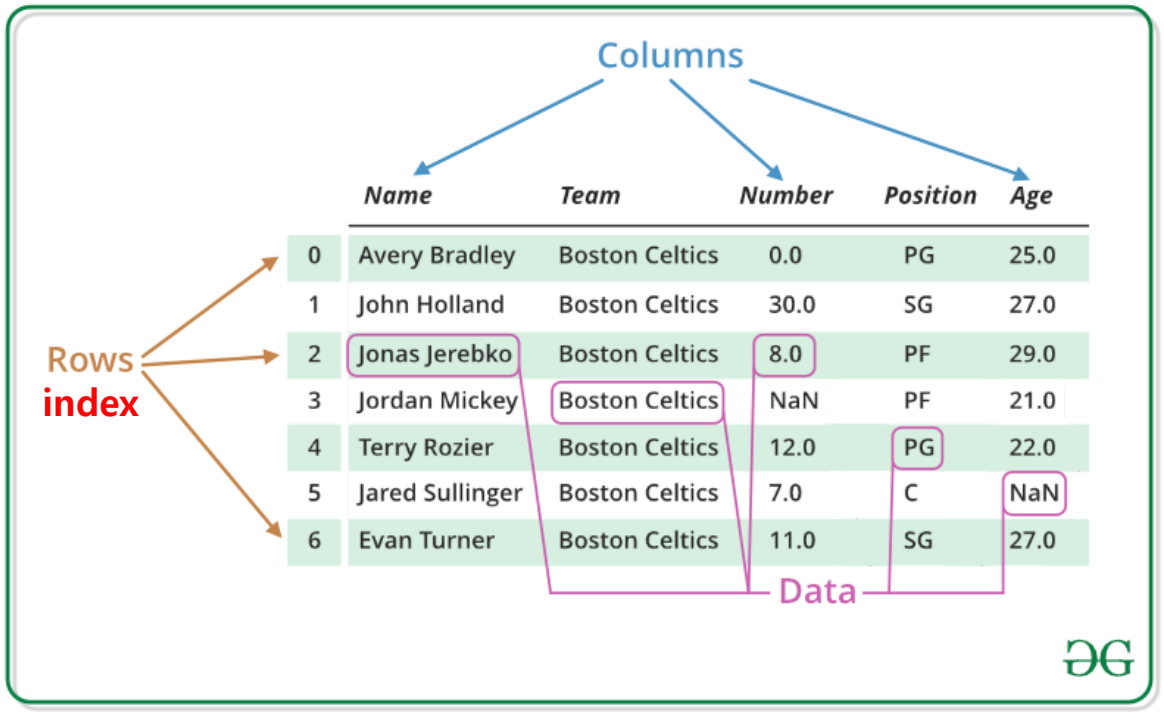
	apples
0	3
1	2
2	0
3	1

+

	oranges
0	0
1	3
2	7
3	2

=

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

판다스 실습 파일

- 21-2-pandas_dataframe.ipynb

선형 회귀 문제

- **일차 함수관계 식 찾기**
  - 선형회귀

x 변수, y 변수 데이터 만들기 (리스트 객체)

```
[1]  x = [-3,  31,  -11,  4,  0,  22, -2, -5, -25, -14]
     y = [ -2,   32,   -10,   5,  1,   23,  -1,  -4, -24,  -13]
     print(x)
     print(y)
```

```
[-3, 31, -11, 4, 0, 22, -2, -5, -25, -14]
[-2, 32, -10, 5, 1, 23, -1, -4, -24, -13]
```

$$y = ax + b$$

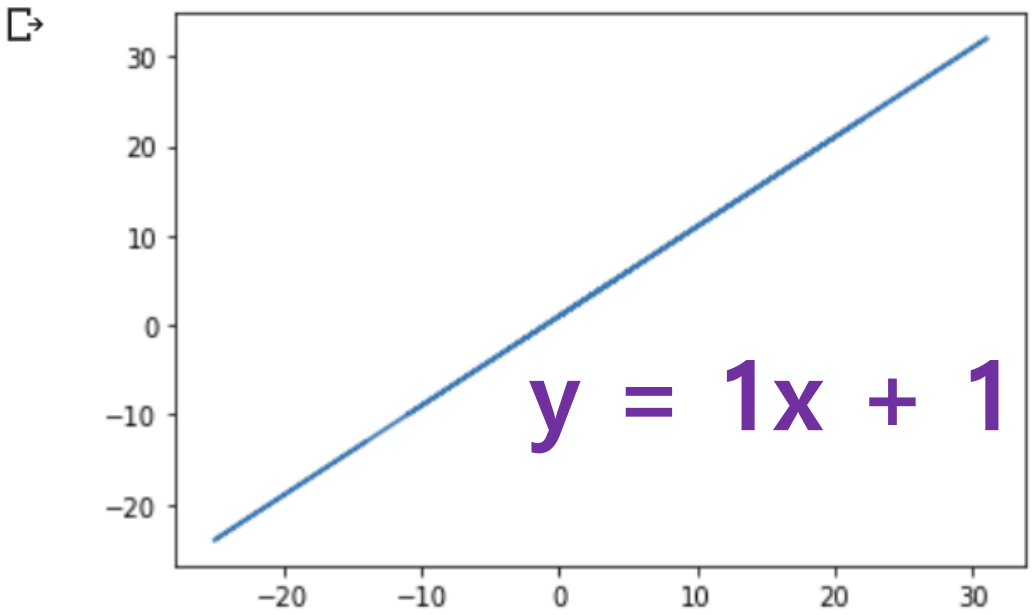
a: 기울기, 가중치(weights)
b: 절편, 편향(bias)

선형 회귀 구현

- **일차 함수관계 식 찾기**
  - 21-3-linear-regression.ipynb

그래프 그리기 (matplotlib)

```
import matplotlib.pyplot as plt
plt.plot(x, y)
plt.show()
```



x 변수, y 변수 데이터 만들기 (리스트 객체)

```
[1]  x = [-3,  31,  -11,  4,  0,  22, -2, -5, -25, -14]
     y = [ -2,   32,   -10,   5,  1,   23,  -1,  -4, -24,  -13]
     print(x)
     print(y)
```

```
[-3, 31, -11, 4, 0, 22, -2, -5, -25, -14]
[-2, 32, -10, 5, 1, 23, -1, -4, -24, -13]
```

선형회귀 모델 구현

$$y = ax + b$$

a: 기울기, 가중치(weights)
b: 절편, 편향(bias)

```
[14] from sklearn.linear_model import LinearRegression
     lr = LinearRegression()
     lr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[15] lr.coef_, lr.intercept_
```

```
(array([[1.]]), array([1.]))
```

$$y = 1x + 1$$

```
[16] print ("기울기 : ", lr.coef_[0][0])
     print ("y절편 : ", lr.intercept_[0])
```

```
기울기 :  0.9999999999999999
y절편 :  0.9999999999999999
```

선형회귀 모델 예측

- **메소드 reshape(1, 1)**
  - 2차원 1행 1열로 모양 변환
- **메소드 reshape(-1, 1)**
  - 2차원 (알아서)행 1열로 모양 변환
- **예측 메소드**
  - lr.predict(2차원_행렬_문제)
  - 정답
    - **2차원_행렬_정답**

```
[17] import numpy as np
     X_new = np.array(11).reshape(1, 1)
     X_new

     array([[11]])

[18] lr.predict(X_new)

     array([[12.]])

[19] X_test = np.arange(11, 16, 1).reshape(-1, 1)
     X_test

     array([[11],
            [12],
            [13],
            [14],
            [15]])

     y_pred = lr.predict(X_test)
     y_pred

     array([[12.],
            [13.],
            [14.],
            [15.],
            [16.]])
```

분류 규현
붓꽃 iris 예측

KNN
SVM

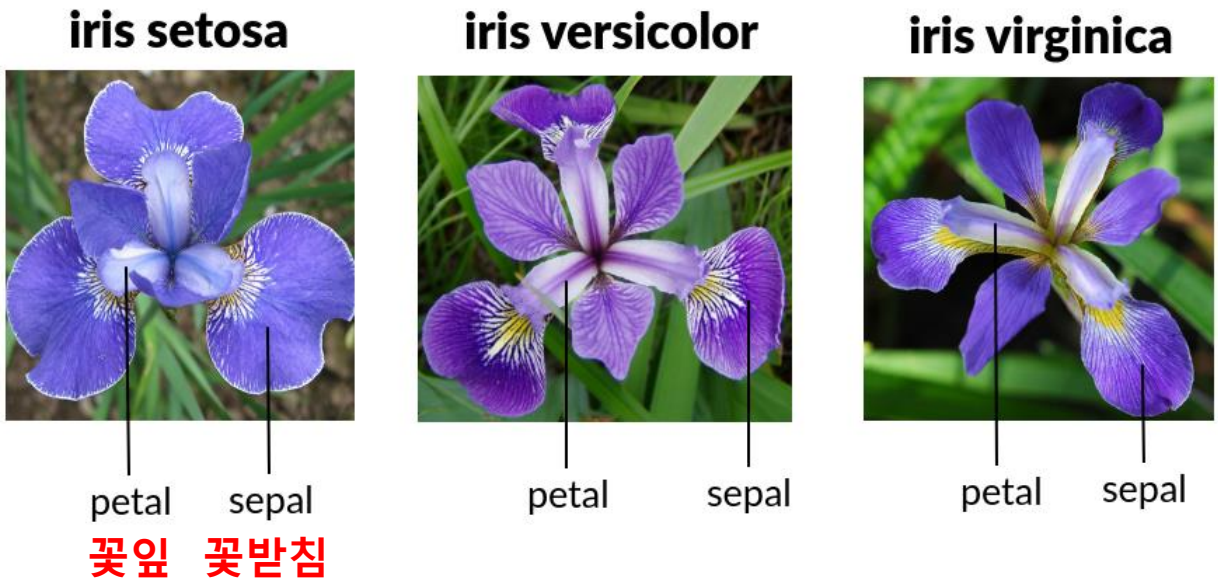
붓꽃 분류 개요

- **붓꽃의 품종 판별**
  - 분류(classification)
    - **# of classes = 3**
      - **Setosa, versicolor, virginica**

그림 3-2 iris의 세 가지 품종(왼쪽부터 Setosa, Versicolor, Virginica)

레이블과 특징 수

- **꽃잎과 꽃받침의 너비와 길이**
  - 4
- **레이블**
  - 3개의 붓꽃 중 하나



datasets.load_iris()

- **자료형 sklearn.utils.Bunch**
  - 파이썬의 사전과 유사

```
# skleran 데이터셋에서 iris 데이터셋 로딩
from sklearn import datasets
iris = datasets.load_iris()

iris
```

```
{'DESCR': '.. _iris_dataset:₩n₩nIris plants dataset₩n------
 'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
```

```
[3] type(iris)
```

```
sklearn.utils.Bunch
```

```
[ ] # iris 데이터셋은 딕셔너리 형태이므로, key 값을 확인
    iris.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

```
[ ] iris['filename']
```

```
'/usr/local/lib/python3.7/dist-packages/sklearn/datasets/data/iris.csv'
```

키 DESCR

- iris['DESCR']

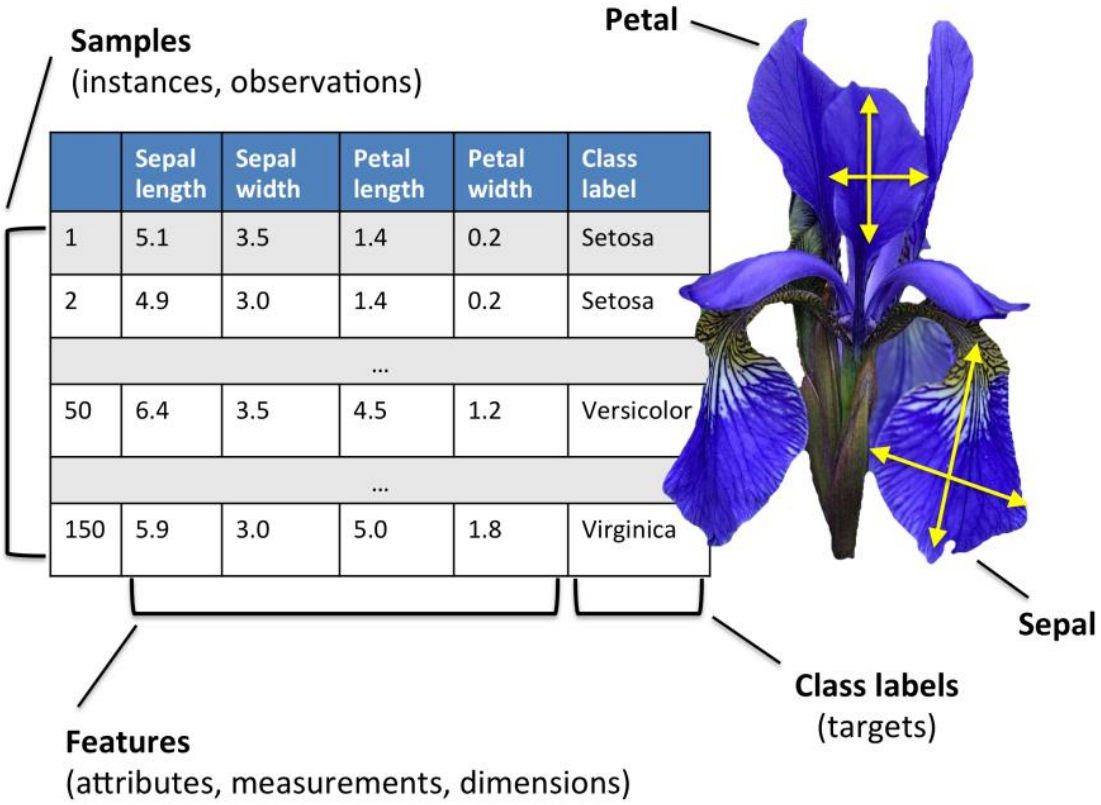
```
Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
...
```

150개의 샘플

네 개의 특징(feature)

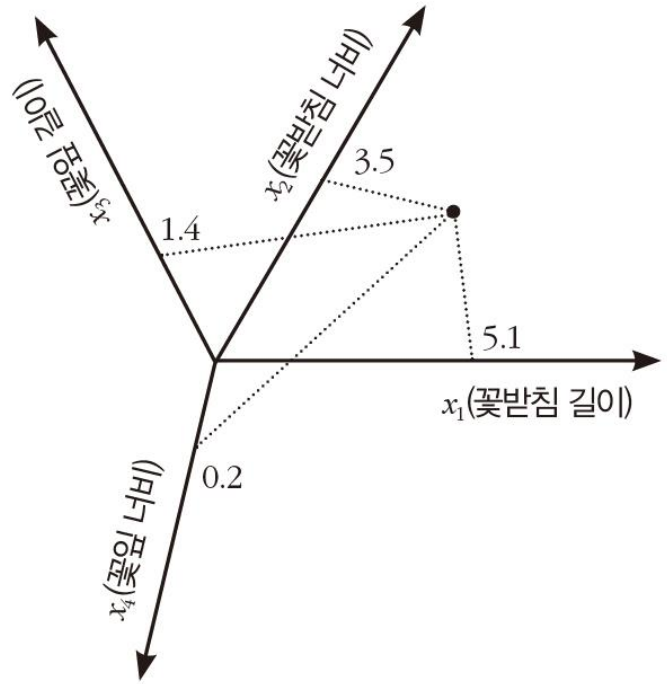
세 개의 부류

붓꽃 분류 문제

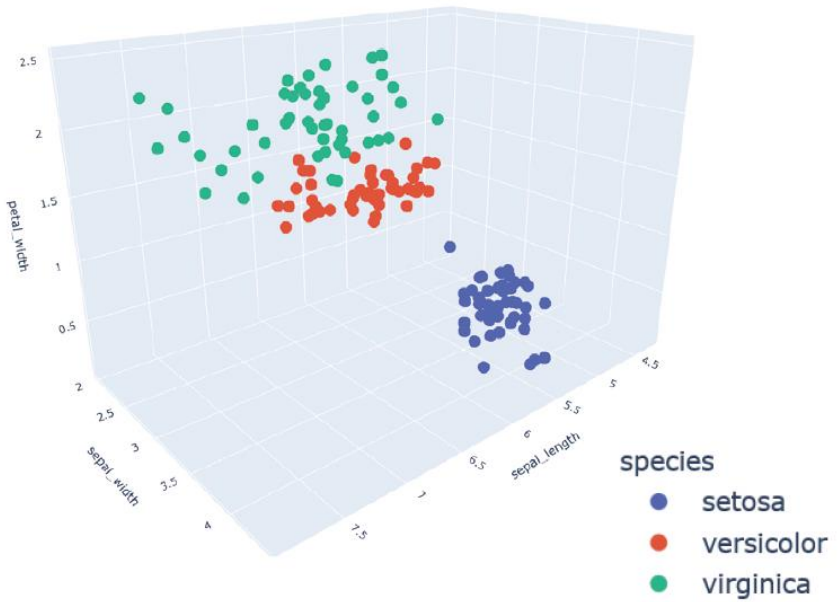


	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

3차원 공간에서의 분류



(a) 4차원 특징 공간(가상의 그림)



(b) 꽃잎 길이 축을 제외한 3차원 특징 공간

그림 3-5 iris 데이터를 특징 공간에 그리기

KNN 알고리즘

- **K 최근접 이웃(KNN) 알고리즘**
  - 근접 이웃 알고리즘(K-Nearest Neighbors)
    - **정답이 있는 지도 학습에 활용, 가장 간단한 알고리즘**
    - **회귀와 분류에 모두 사용**
  - 새로운 입력 자료에 대해 주위에 가장 많은 유형으로 분류
- **알고리즘 이해**
  - 기존 훈련 데이터에서 클래스 A에 속하는 것은 붉은 별표, 클래스 B는 녹색 삼각형
  - 만일 중앙 부근에 위치한 새로운 데이터 ?는 별표와 삼각형 중 무엇으로 분류해야 할까?

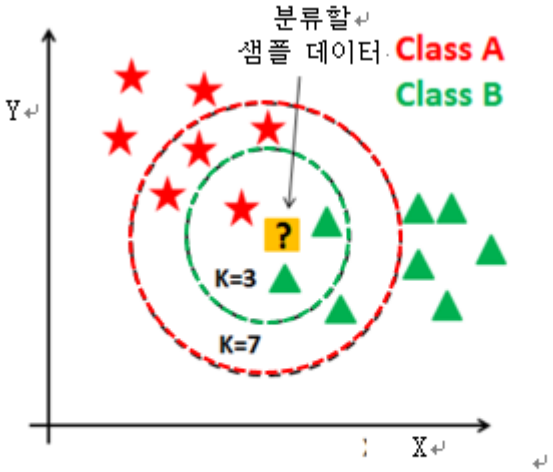


그림. K 최근접 이웃 알고리즘의 예측

IRIS 구현

- **소스**
  - 21-4-iris-classification.ipynb

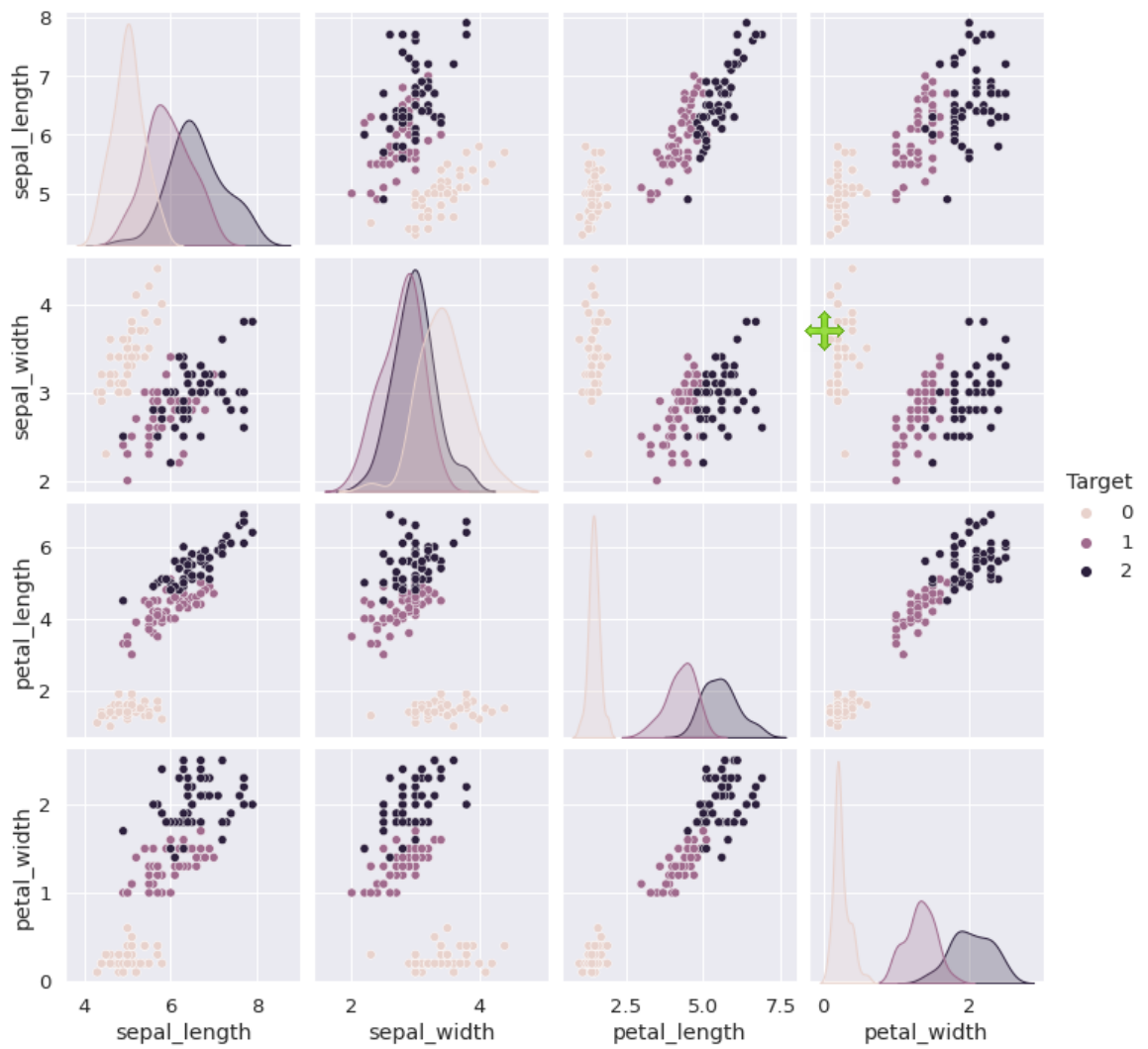
- **EDA(Exploratory Data Analysis): 탐색적 데이터 분석**
  - 자기주도학습

  - 존 튜키라는 미국의 저명한 통계학자가 창안한 자료 분석 방법론
    - **주어진 자료만 가지고도 충분한 정보를 찾을 수 있도록 여러가지 탐색적 자료 분석 방법을 개발**
    - **박스플롯 다양한 시각화 도구로 자료에 대한 충분한 이해를 한 후에 모형 적합 등의 좀 더 정교한 모형을 개발**

Seaborn KDE(Kernel Density Estimator)

- 시각화도구
  - Seaborn
  - 커널 밀도 추정 (KDE) 플롯
    - 데이터의 분포를 서로 비교해 확인

```
# 두 변수 간의 관계
sns.pairplot(df, hue = 'Target', diag_kind = 'kde')
plt.show()
```



자기주도학습

- **자료**
  - [02주 자기주도학습] 구글 Colab 개요와 활용.pptx
  - [03주 자기주도학습] 구글 Colab 고급.pptx

학습 데이터와 테스트 데이터 분리

```
from sklearn.model_selection import train_test_split

X_data = df.loc[:, 'sepal_length':'petal_width']
y_data = df.loc[:, 'Target']

X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    random_state=20)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

df

	sepal_length	sepal_width	petal_length	petal_width	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2

X_train

	sepal_length	sepal_width	petal_length	petal_width
95	5.7	3.0	4.2	1.2
88	5.6	3.0	4.1	1.3
0	5.1	3.5	1.4	0.2
46	5.1	3.8	1.6	0.2
11	4.8	3.4	1.6	0.2

Knn 구현

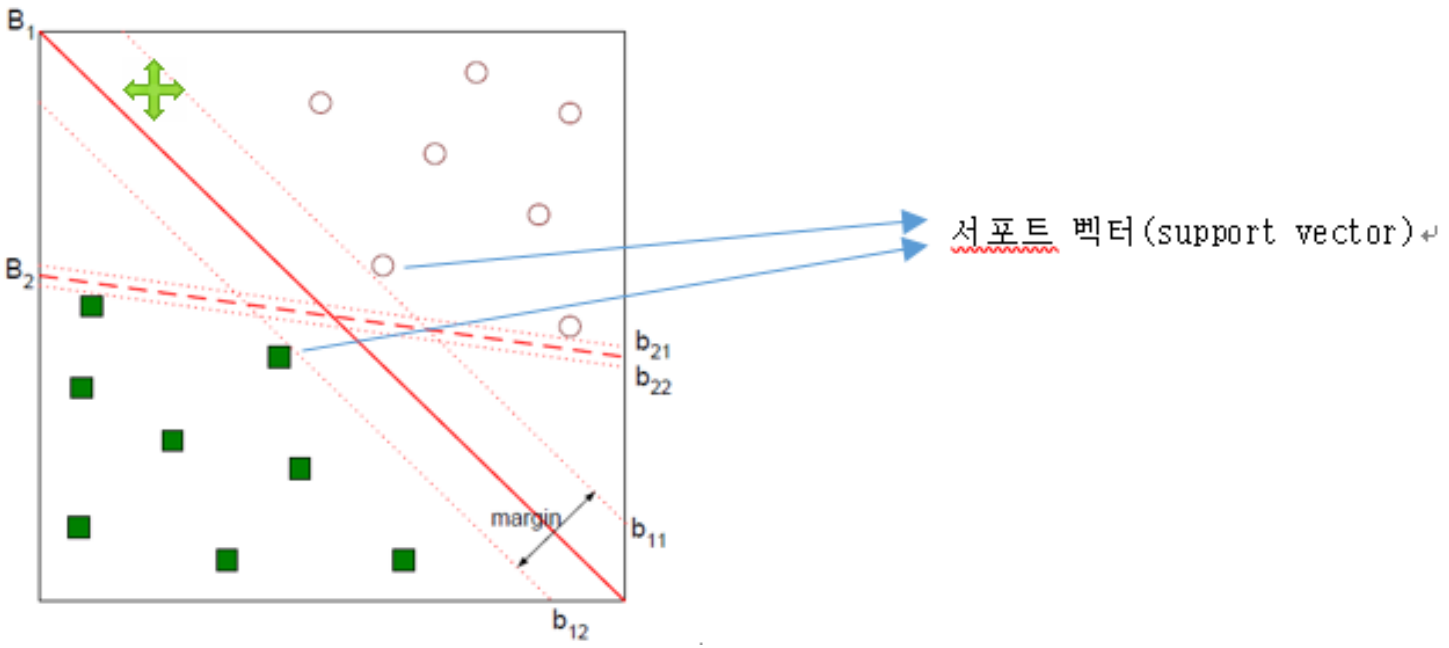
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# 모델 학습
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)

# 예측
y_knn_pred = knn.predict(X_test)
print("예측값: ", y_knn_pred[:10])
print("정답 : ", y_test[:10].values)

# 성능 평가
knn_acc = accuracy_score(y_test, y_knn_pred)
print("Accuracy: %.4f" % knn_acc)
```

KNN을 사용한 붓꽃 분류 전 소스

```
# KNN을 적용한 붓꽃 분류
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import datasets

# 1. 데이터 준비와 전처리
iris = datasets.load_iris()

# 데이터프레임 생성과 열 지정
df = pd.DataFrame(iris['data'], columns=iris['feature_names'])
df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
df['Target'] = iris['target']

# 중복 데이터 제거
df = df.drop_duplicates()

# 2. 학습 데이터, 테스트 데이터 준비
X_data = df.loc[:, 'sepal_length':'petal_width']
y_data = df.loc[:, 'Target']
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.2,
                                                    shuffle=True, random_state=20)

# 3. 모델 학습
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)

# 4. 예측
y_knn_pred = knn.predict(X_test)
print("예측값: ", y_knn_pred[:10])
print("정답 : ", y_test[:10].values)

# 5. 성능 평가
knn_acc = accuracy_score(y_test, y_knn_pred)
print("Accuracy: %.4f" % knn_acc)
```

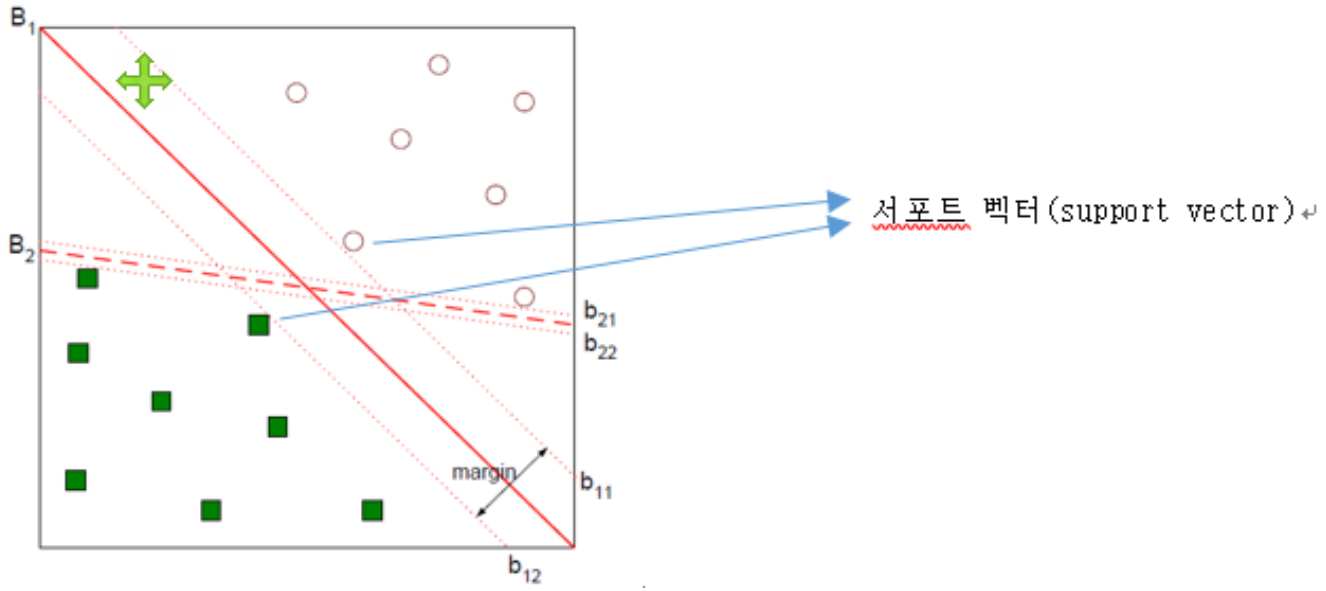
SVM

- **SVM: Support Vector Machine**
  - 두 분류 사이의 거리인 마진(margin)을 최대화하는 분류 기준 경계인 결정 경계(decision boundary)를 찾는 모델
    - **이진 분류에 주로 사용**
    - **딥러닝과 함께 인식률을 매우 좋아 최근까지도 가장 많이 사용되는 알고리즘**



SVM 알고리즘 이해

- **직관적 이해**
  - 직선 B1은 점선 B2보다 마진이 큰 결정 경계가 되어 녹색 네모와 빈 원을 구분
  - 분류되지 않은 새로운 점이 나타나면
    - **경계의 어느 쪽에 속하는지 확인해서 분류 과제를 수행**
- **서포트 벡터(support vectors)**
  - 결정 경계에 가장 가까운 각 클래스의 점들
  - 서포트 벡터(support vectors)를 사용해서 결정 경계(Decision Boundary)를 정의
  - 분류되지 않은 점을 해당 결정 경계와 비교해서 분류



SVM을 사용한 붓꽃 분류 전 소스

```
# SVM을 적용한 붓꽃 분류
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import datasets

# 1. 데이터 준비와 전처리
iris = datasets.load_iris()

# 데이터프레임 생성과 열 지정
df = pd.DataFrame(iris['data'], columns=iris['feature_names'])
df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
df['Target'] = iris['target']

# 중복 데이터 제거
df = df.drop_duplicates()

# 2. 학습 데이터, 테스트 데이터 준비
X_data = df.loc[:, 'sepal_length':'petal_width']
y_data = df.loc[:, 'Target']

X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,
                    test_size=0.2, shuffle=True, random_state=20)

# 모델 학습
svc = SVC(kernel='rbf')
svc.fit(X_train, y_train)

# 4. 예측
y_svc_pred = svc.predict(X_test)
print("예측값: ", y_svc_pred[:10])
print("정답 : ", y_test[:10].values)

# 5. 성능 평가
svc_acc = accuracy_score(y_test, y_svc_pred)
print("Accuracy: %.4f" % svc_acc)
```