Unsupervised Learning

학습목표

- Unsupervised Learning 알고리즘에 대해 설명할 수 있다.
- Unsupervised Learning 알고리즘을 코드로 구현 할 수 있다.

학습내용

- Unsupervised Learning의 개념
- Unsupervised Learning 알고리즘
- Unsupervised Learning 의 평가 방법

Contents

01 비지도 학습

02 Clustering

03 k-Means Clustering

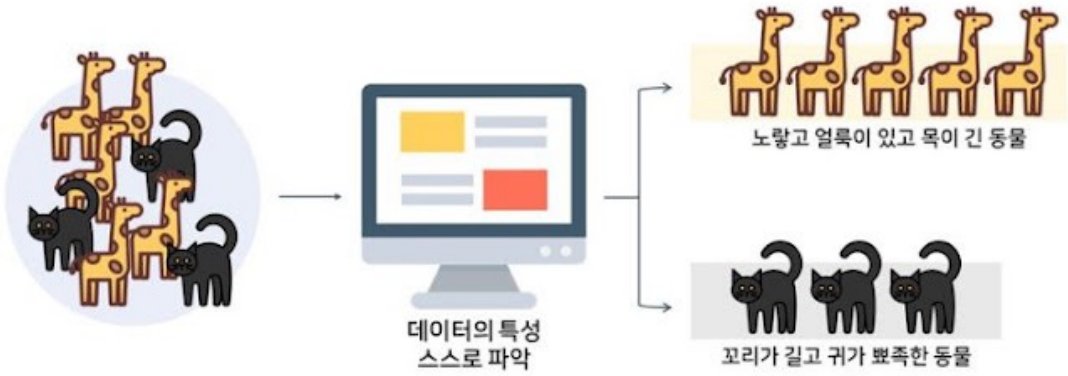
04 Hierarchical Clustering

05 비지도 학습의 평가

01 비지도 학습

비지도 학습

- 비지도 학습(자율학습, Unsupervised Learning)
  - 기계학습의 일종으로, 데이터가 어떻게 구성되었는지 알아내는 문제의 범주
  - 이 방법은 지도학습(Supervised Learning) 혹은 강화학습(Reinforcement Learning)과는 달리 입력값에 대한 목표치(정답)가 주어지지 않음
- 비지도 학습은 통계의 밀도추정(Density Estimation)과 깊은 연관이 있음, 이러한 비지도 학습은 데이터의 주요 특징을 요약하고 설명할 수 있음
- 비지도 학습의 예 : 클러스터링(Clustering),
  독립 성분 분석(Independent Component Analysis)



02 Clustering

Clustering

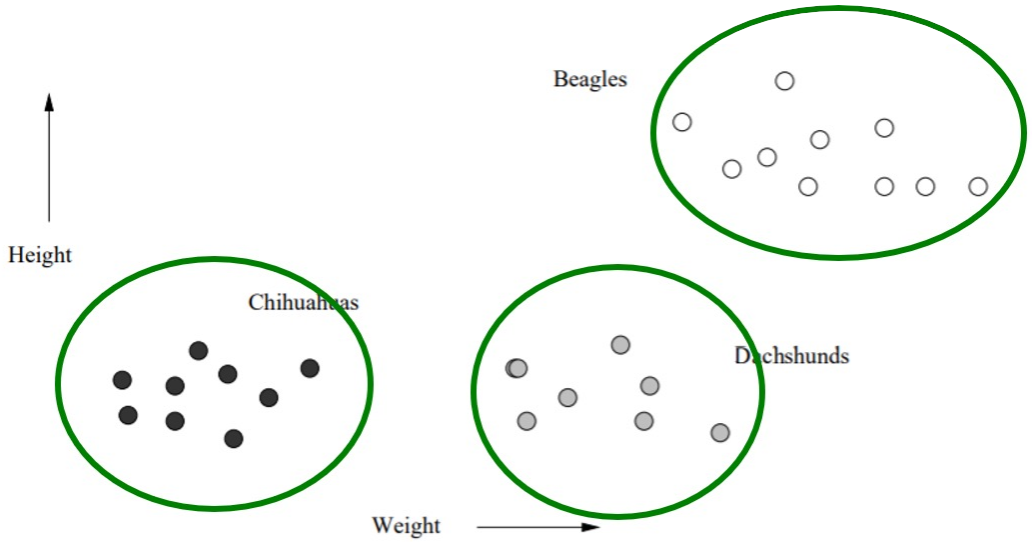


Figure 7.1: Heights and weights of dogs taken from three varieties

출처: https://www.secmem.org/blog/2019/05/17/clustering/

- 클러스터링에 사용되는 알고리즘은 크게 계층적 군집(Hierarchical clustering)과 Point assignment clustering 두 가지 방법으로 나눌 수 있다.

- 계층적 군집 방법은 다시 큰 하나의 클러스터로부터 시작해서 모든 클러스터가 정확히 하나의 원소를 가질 때까지 계속 쪼개는 divisive(top-down)한 방법과, 각각의 점을 원소로 가지는 클러스터들로부터 전체를 포함하는 클러스터 하나를 만들 때까지 반복적으로 두 개의 "가까운" 클러스터를 합치면서 진행하는 Agglomerative(bottom-up)한 방법으로 나누어진다.

03 k-Means Clustering

K-Means Clustering

- k-means 클러스터링은 대표적인 클러스터링 알고리즘 중 하나로
  각 클러스터에 할당된 데이터 포인트의 평균 좌표를 이용해 중심점을 반복적으로
  업데이트 하며 클러스터를 형성하는 알고리즘

- k-Means 클러스터링 진행 단계

  Step 1: 각 데이터 포인트에서 가장 가까운 중심점을 찾아 그 중심점에 해당되는
  클러스터 할당

  Step 2: 할당된 클러스터를 기반으로 새로운 중심점 계산,
  이 때 중심점은 클러스터 내부 점들의 좌표의 산술 평균

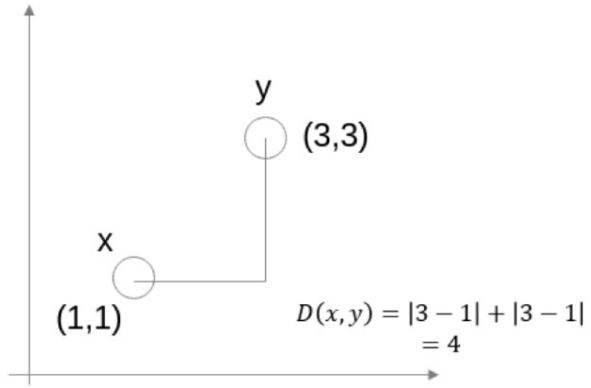
  Step 3: 각 클러스터의 할당이 바뀌지 않을 때까지 반복

K-Means Clustering

- 점과 점 사이의 거리를 어떻게 측정할 수 있을까? (k-means 클러스터링은 거리 기반 알고리즘)

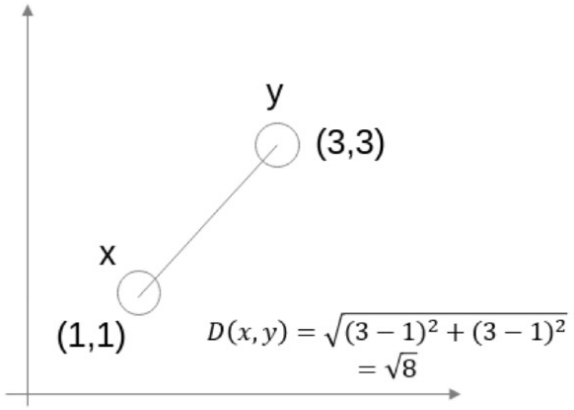
1. Manhattan Distance – 각 축에 대해 수직으로만 이동하는 방식

$$D(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$



2. Euclidean Distance – 점과 점사이의 가장 짧은 거리를 계산하는 거리 측정방식

$$D(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

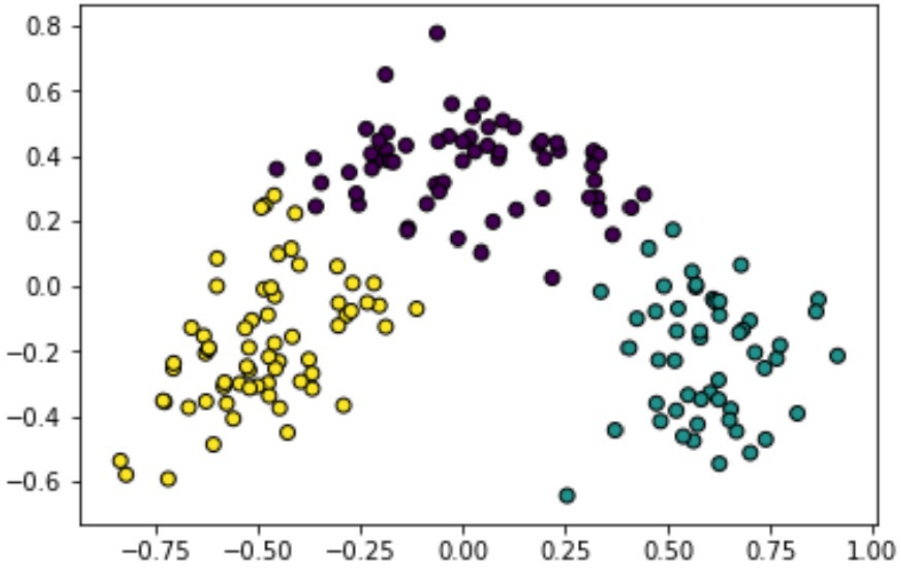


K-Means Clustering


```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)

kmeans.fit(data)

cluster = kmeans.predict(data)
```



04 Hierarchical Clustering

Hierarchical Clustering

- Hierarchical 클러스터는 거리(Distance) 또는 유사도(Similarity)를 기반으로 클러스터를 형성하는 알고리즘
- Hierarchical 클러스터 진행 단계

  Step 1: 각 데이터 포인트를 클러스터로 할당 (n개의 클러스터)

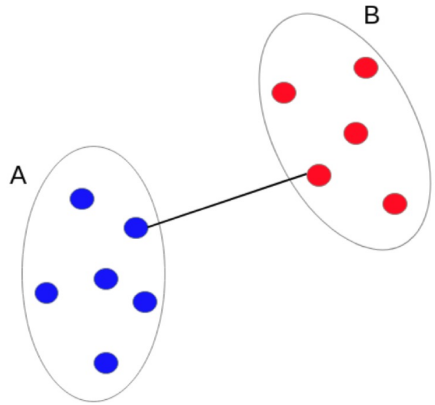
  Step 2: 가까운 클러스터끼리 병합

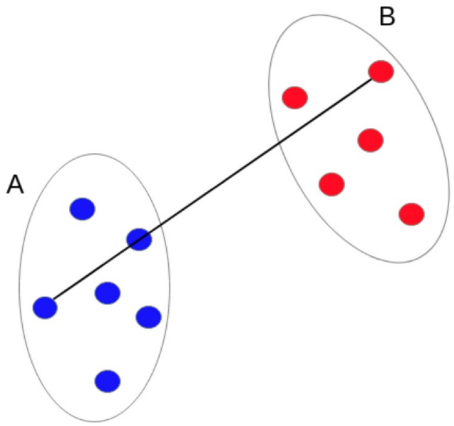
  Step 3: 1개의 클러스터가 될 때까지 반복

Hierarchical Clustering

- 어떻게 가장 가까운 클러스터를 찾을 수 있을까?

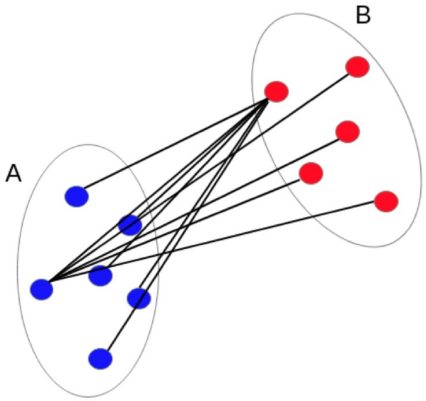
1. Single Linkage – 두 클러스터 내의 가장 가까운 점 사이의 거리





2. Complete Linkage – 두 클러스터 내의 가장 먼 점 사이의 거리

3. Average Linkage – 두 클러스터 내의 모든 점 사이의 평균 거리

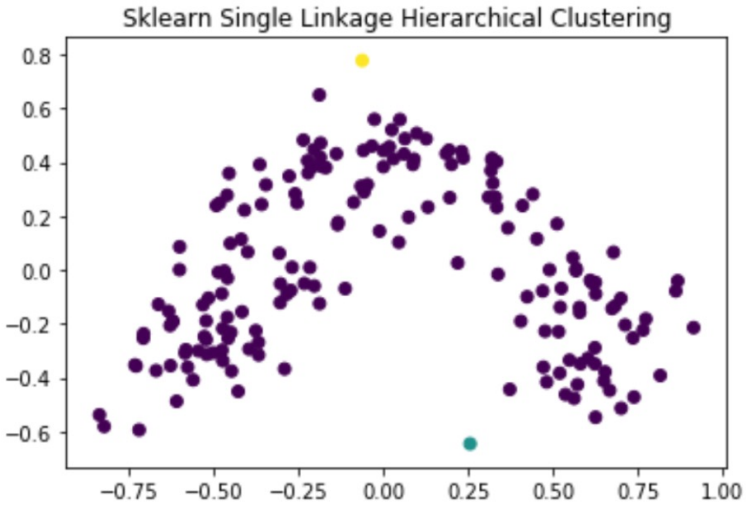


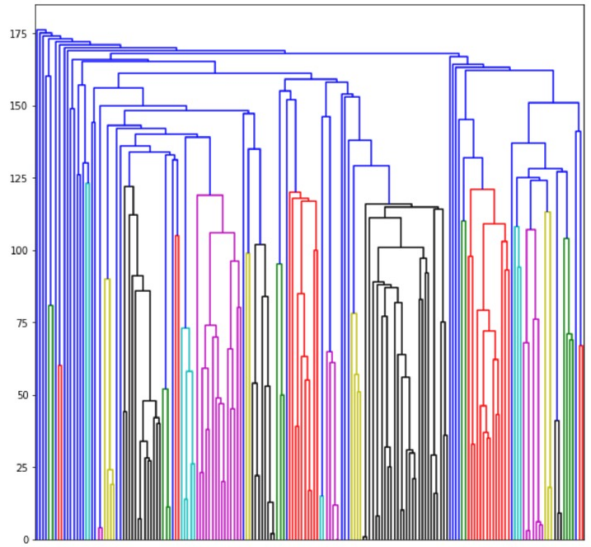
Hierarchical Clustering(Bottom-up, Single Linkage)


```
from sklearn.cluster import AgglomerativeClustering
single_clustering = AgglomerativeClustering(linkage='single', n_clusters=3)
```

```
single_clustering.fit(data)
```

```
single_cluster = single_clustering.labels_
```



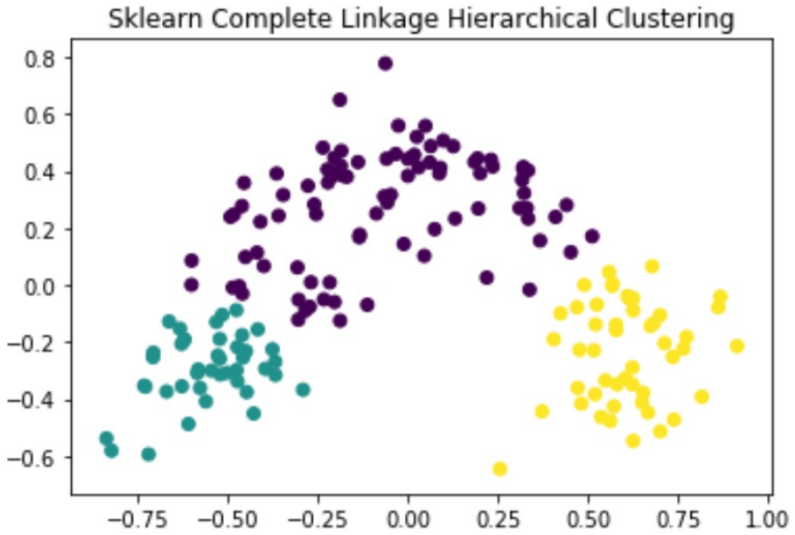


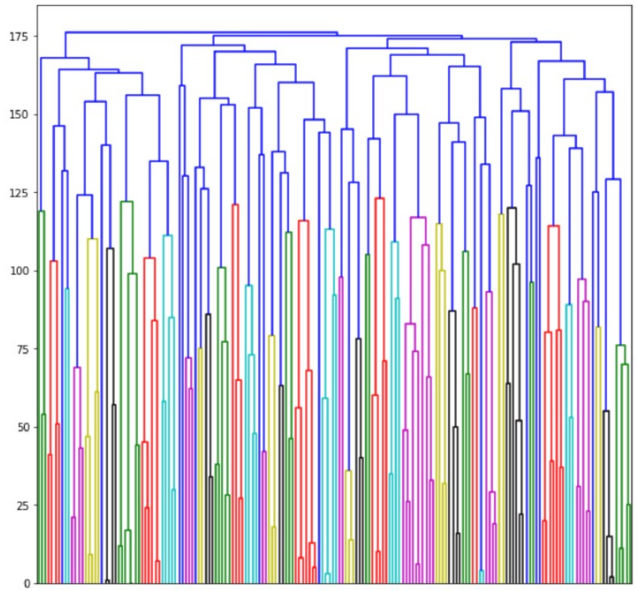
Hierarchical Clustering(Bottom-up, Complete Linkage)


```
complete_clustering = AgglomerativeClustering(linkage='complete', n_clusters=3)
```

```
single_clustering.fit(data)
```

```
single_cluster = single_clustering.labels_
```



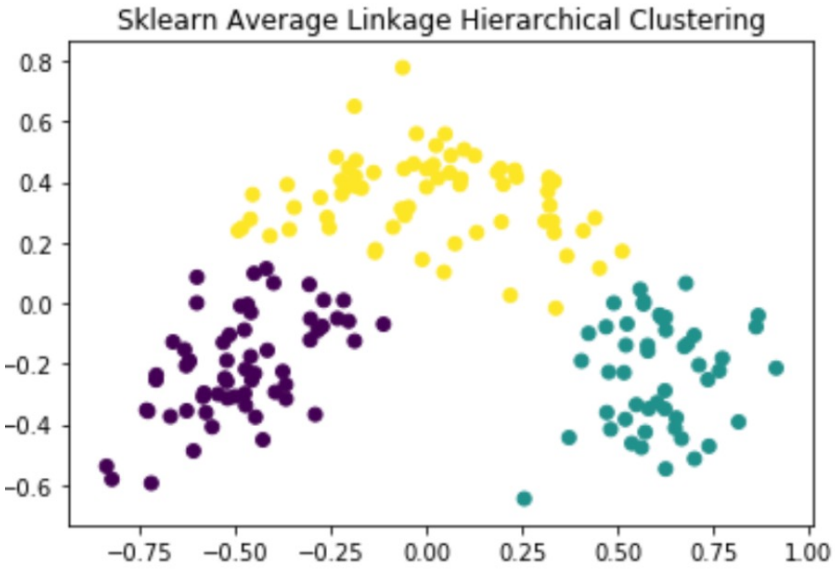


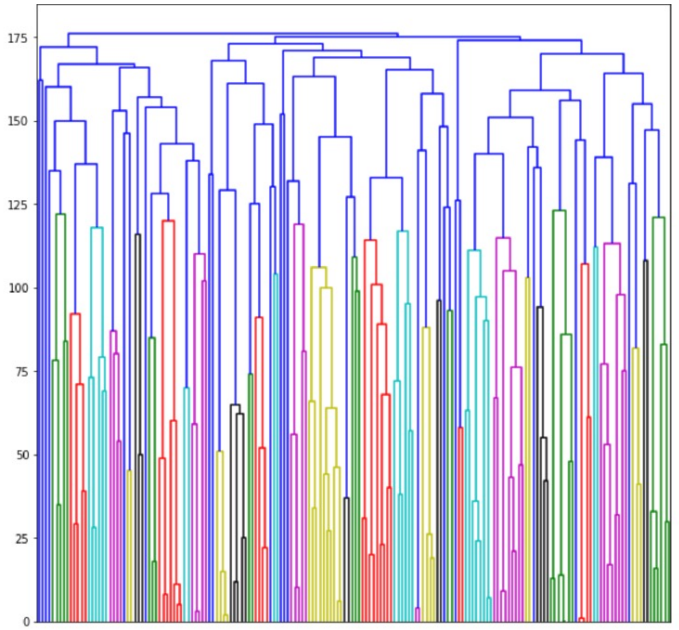
Hierarchical Clustering(Bottom-up, Average Linkage)


```
average_clustering = AgglomerativeClustering(linkage='average', n_clusters=3)

single_clustering.fit(data)

single_cluster = single_clustering.labels_
```





05 비지도 학습의 평가

Silhouette

- 실루엣 값은 클러스터 안의 데이터들이 다른 클러스터와 비교해서 얼마나 비슷한가를 나타냄
- 같은 클러스터 내의 점들 간 거리는 가깝고(cohesion)
  서로 다른 클러스터 간의 거리는 멀수록(separation) 높은 값
- 실루엣 값이 1에 근접한다는 것은 같은 클러스터 내의 평균 거리가 다른 클러스터와의
  평균거리보다 가깝다는 것을 의미

$$S_i = \frac{(b_i - a_i)}{max(a_i, b_i)}$$

a_i : 같은 클러스터 내의 모든 점들 간 평균 거리

b_i : $\bar{d} = (i, c)$의 최솟값

$\bar{d} = (i, c)$: 다른 클러스터 c와 i번째데이터와의 평균 거리

Silhouette


```python
from sklearn.metrics import silhouette_score

best_n = 1
best_score = -1

for n_cluster in range(2, 11):
    kmeans = KMeans(n_clusters=n_cluster)
    kmeans.fit(data)
    cluster = kmeans.predict(data)
    score = silhouette_score(data, cluster)

    print('클러스터의 수 : {}, 실루엣 점수 : {:.2f}'.format(n_cluster, score))
    if score > best_score :
        best_n = n_cluster
        best_score = score

print('가장 높은 실루엣 점수를 가진 클러스터 수 : {}, 실루엣 점수 : {:.2f}'.format(best_n, best_score))
```

```
클러스터의 수 : 2, 실루엣 점수 : 0.49
클러스터의 수 : 3, 실루엣 점수 : 0.57
클러스터의 수 : 4, 실루엣 점수 : 0.49
클러스터의 수 : 5, 실루엣 점수 : 0.42
클러스터의 수 : 6, 실루엣 점수 : 0.42
클러스터의 수 : 7, 실루엣 점수 : 0.40
클러스터의 수 : 8, 실루엣 점수 : 0.40
클러스터의 수 : 9, 실루엣 점수 : 0.39
클러스터의 수 : 10, 실루엣 점수 : 0.39
가장 높은 실루엣 점수를 가진 클러스터 수 : 3, 실루엣 점수 : 0.57
```

Silhouette


```python
from sklearn.metrics import silhouette_score

best_n = 1
best_score = -1

for n_cluster in range(2, 11):
    average_clustering = AgglomerativeClustering(n_clusters=n_cluster, linkage='average')
    average_clustering.fit(data)
    cluster = average_clustering.labels_
    score = silhouette_score(data, cluster)

    print('클러스터의 수 : {}, 실루엣 점수 : {:.2f}'.format(n_cluster, score))
    if score > best_score :
        best_n = n_cluster
        best_score = score

print('가장 높은 실루엣 점수를 가진 클러스터 수 : {}, 실루엣 점수 : {:.2f}'.format(best_n, best_score))
```

```
클러스터의 수 : 2, 실루엣 점수 : 0.49
클러스터의 수 : 3, 실루엣 점수 : 0.56
클러스터의 수 : 4, 실루엣 점수 : 0.48
클러스터의 수 : 5, 실루엣 점수 : 0.42
클러스터의 수 : 6, 실루엣 점수 : 0.37
클러스터의 수 : 7, 실루엣 점수 : 0.34
클러스터의 수 : 8, 실루엣 점수 : 0.34
클러스터의 수 : 9, 실루엣 점수 : 0.37
클러스터의 수 : 10, 실루엣 점수 : 0.33
가장 높은 실루엣 점수를 가진 클러스터 수 : 3, 실루엣 점수 : 0.56
```

붓꽃 예제 실습

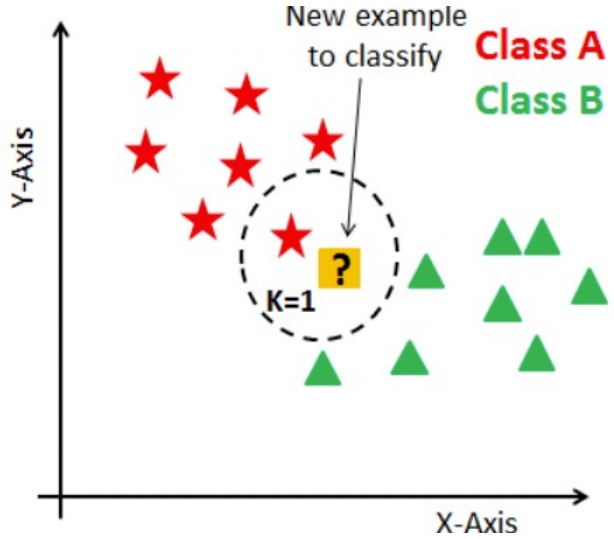
- 주제 : 붓꽃 품종 예측 머신러닝 모델 생성
- 상황
  - 한 아마추어 식물학자가 들에서 발견한 붓꽃의 품종을 알고 싶음.
  - 이 식물학자는 붓꽃의 꽃잎 petal과 꽃받침 sepal의 혹과 길이를 센티미터 단위로 측정함
  - 전문 식물학자가 setosa, versicolor, virginica종으로 분류한 붓꽃 측정 데이터 보유
  - 이 측정값을 이용해서 앞에서 채집한 붓꽃의 품종을 구분하고자 함
- 알고리즘 : 지도학습 – 분류(Classification)
- 클래스(class) : 출력될 수 있는 값(붓꽃의 종류-세개)
- 레이블(label) : 특정 데이터 포인트에 대한 출력(품종)

출처: https://tensorflow.blog/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D/1-7-%EC%B2%AB-%EB%B2%88%EC%A7%B8-%EC%95%A0%ED%94%8C%EB%A6%AC%EC%BC%80%EC%9D%B4%EC%85%98-%EB%B6%93%EA%BD%83%EC%9D%98-%ED%92%88%EC%A2%85-%EB%B6%84%EB%A5%98/

머신러닝 모델 : k-최근접 이웃 알고리즘

- scikit-learn은 다양한 분류 알고리즘을 제공하며,
  본 실습에서는 k-최근접 이웃k-Nearest Neighbors, k-NN 분류기를 이용

- k-최근접 이웃k-Nearest Neighbors, k-NN 분류기
  - 단순히 훈련 데이터를 저장하여 구성
  - 새로운 데이터 포인트에 대한 예측이 필요하면 알고리즘은 새 데이터 포인트에서 가장 가까운 훈련 데이터 포인트를 찾는 방식
  - 이후 찾은 훈련 데이터의 레이블을 새 데이터 포인트의 레이블로 지정



출처: https://bioinformaticsandme.tistory.com/179

붓꽃 데이터의 예측(Predict)과 모델 평가

- 붓꽃 품종 예측
  - 꽃받침 길이: 5cm, 폭: 2.9cm, 꽃잎의 길이: 1cm, 폭: 0.2cm

```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))

prediction = knn.predict(X_new)
print("예측: {}".format(prediction))
print("예측한 타깃의 이름: {}".format( iris_dataset['target_names'][prediction]))
```

- 모델 평가

```
y_pred = knn.predict(X_test) print("테스트 세트에 대한 예측값:\n {}".format(y_pred))

print("테스트 세트의 정확도: {:.2f}".format(np.mean(y_pred == y_test)))
```

Summary

Unsupervised Learning

핵심정리_Unsupervised Learning

1. 비지도 학습은 Label이 없는 상태에서 존재하는 데이터의 패턴을 유추하는 방식으로 만약 Label 있다면 Classification 계열을 사용하는 것이 나음

2. Single Linkage, Complete Linkage, Average Linkage 등의 구별은 클러스터 사이의 거리를 구하는 방식에 따라서 나누어 짐

3. 비지도 학습을 평가하는 방식은 실루엣을 구하는 방식으로 실루엣 값은 클러스터 안의 데이터들이 다른 클러스터와 비교해서 얼마나 비슷한가를 나타냄

감사합니다.