군집(Clustering)

안 화 수

사이킷런(Scikit-Learn)

- 사이킷런

- 사이킷런 모듈 설치

1. python 에 설치

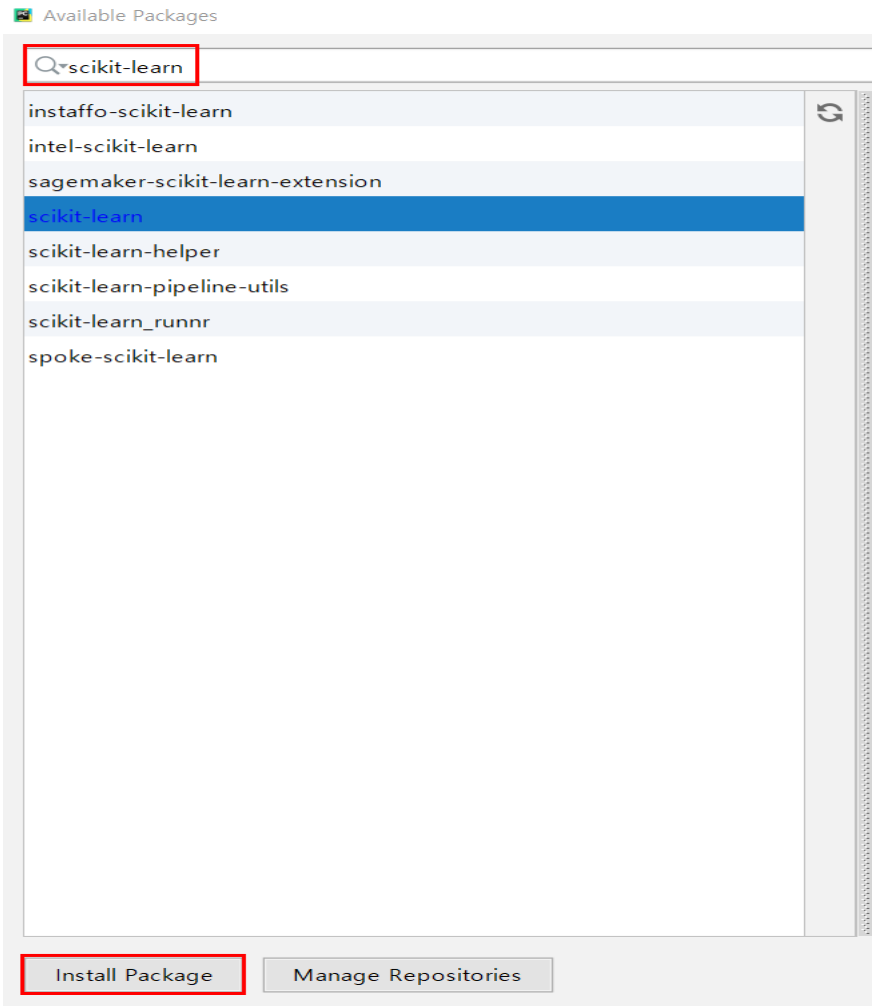
```
c:₩> pip install scikit-learn
```

2. anaconda 에 설치

```
c:₩> conda install scikit-learn
```

사이킷런(Scikit-Learn)

- PyCharm에 scikit-learn 설치



Scikit-Learn DataSet

사이킷런에 내장된 예제 데이터

사이킷런에는 별도의 외부 웹사이트에서 데이터 세트를 다운로드 받을 필요 없이 예제로 활용할 수 있는 간단하면서도 좋은 데이터 세트가 내장되어 있다. 이 데이터는 datasets 모듈에 있는 여러 API를 호출해 만들 수 있다.

<분류나 회귀 연습용 예제 데이터>

API명	설명
datasets.load_digits()	분류용, 0에서 9까지 숫자 이미지 픽셀 데이터셋
datasets.load_iris()	분류용, 붓꽃에 대한 피처를 가진 데이터셋
datasets.load_breast_cancer()	분류용, 위스콘신 유방암 피처들과 악성/음성 레이블
datasets.load_boston()	회귀용, 미국 보스톤 집 피처들과 가격 데이터셋
datasets.load_diabetes()	회귀용, 당뇨 데이터셋

군집

- **군집(clustering)**

  군집은 데이터를 비슷한 것끼리 그룹으로 묶어주는 알고리즘이다.

< 군집 알고리즘 >

Types	Tasks	Algorithms
비지도 학습 (Unsupervised Learning)	군집(Clustering)	**K-Means Clustering**
		DBSCAN Clustering
		Hierarchical Clustering (계층형 군집)

군집

군집(clustering)

군집분석은 데이터셋의 관측값이 가지고 있는 여러 속성을 분석하여 서로 비슷한 특징을 갖는 관측값끼리 같은 클러스터(집단)으로 묶는 알고리즘이다.
다른 클러스터 간에는 서로 완전하게 구분되는 특징을 갖기 때문에 어느 클러스터에도 속하지 못하는 관측값이 존재할 수 있다.
이런 특성을 이용하여 특이 데이터(이상값, 중복값 등)를 찾는데 활용할 수 있다.

한편 군집분석은 비지도학습 유형이다.
관측값을 몇 개의 집단으로 나눈다는 점에서 분류 알고리즘과 비슷하지만, 군집분석은 정답이 없는 상태에서 데이터 자체의 유사성만을 기준으로 판단하는 점에서 정답을 알고 있는 상태에서 학습 과정을 거치는 분류 알고리즘과 차이가 있다.

군집 알고리즘은 신용카드 부정 사용 탐지, 구매 패턴 분석 등 소비자 행동특성을 그룹화 하는데 사용된다. 어떤 소비자와 유사한 특성을 갖는 집단을 구분하게 되면, 같은 집단 내의 다른 소비자를 통해 새로운 소비자의 구매 패턴이나 행동 등을 예측하는데 활용할 수 있다.
군집에는 여러 가지 알고리즘이 있지만, 여기서는 k-means알고리즘과 DBSCAN 알고리즘에 대해서 알아보자.

iris DataSet

- **피셔의 붓꽃**

  iris 데이터셋은 이 데이터셋 값을 공개해 통계학적 분석을 실시한 로널드 피셔(Ronald Fisher, 1890 ~ 1962) 이름에서 따온 것이다.

  이것은 영국 식물학자 에드가 앤더슨(Edgar Anderson, 1897 ~ 1969)이 세 종류의 붓꽃의 유전적 분화를 밝힐 목적으로 여러 개체의 크기를 측정한 것을 정비한 데이터 셋이다.

- **iris 데이터셋이란 ?**

  iris 는 붓꽃 관련 데이터셋으로 피셔의 붓꽃 이라고도 한다.

- **데이터 내용**

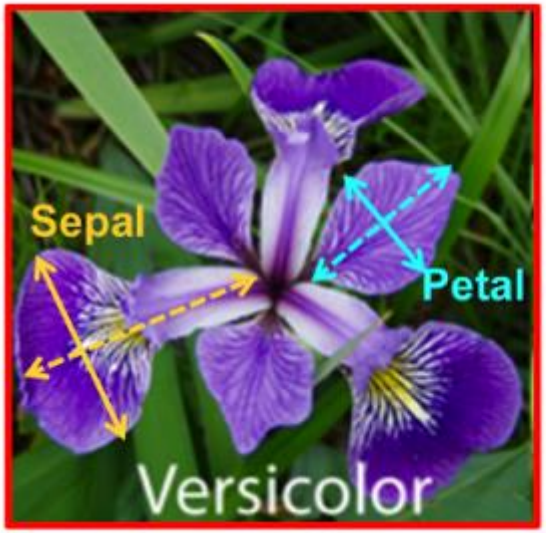
  붓꽃 3종류에서 각각 50개씩의 측정 데이터

- **측정 데이터 내역**

  꽃받침의 길이, 꽃받침의 폭, 꽃잎의 길이, 꽃잎의 폭

iris DataSet

- **iris DataSet**



iris DataSet

- **scikit-learn 의 iris DataSet**

  scikit-learn 에는 iris 데이터셋이 내장되어 있으며, iris 데이터를 읽을 수 있도록 API로 sklearn.datasets.load_iris() 를 준비해 두었다.

```
from sklearn import datasets
iris = datasets.load_iris()
```

- **target_names : 붓꽃의 품종이 등록되어 있음**

```
print( iris['target_names'] )
# ['setosa' 'versicolor' 'virginica']
```

- **feature_names : 데이터 속성의 이름이 등록되어 있음**

```
print( iris['feature_names'] )
# ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
# [ '꽃받침의 길이', '꽃받침의 폭', '꽃잎의 길이', '꽃잎의 폭' ]
```

iris DataSet

- **scikit-learn 의 iris DataSet**

iris_datasets.py (1 / 2)

```
from sklearn import datasets

# iris 데이터 로드
iris = datasets.load_iris()

# 1. data : 붓꽃의 측정값
data = iris['data']
# print(data)

# 2. DESCR
# 피셔의 붓꽃데이터 설명 출력
print(iris['DESCR'])      # iris Data Set Characteristics
# - class:
#                 - Iris-Setosa
#                 - Iris-Versicolour
#                 - Iris-Virginica
```

iris DataSet

- **scikit-learn 의 iris DataSet**

iris_datasets.py (2 / 2)

```
# 3. target : 붓꽃의 품종이 ID 번호로 등록되어 있음
print(iris['target'])

# 4. target_names : 붓꽃의 품종이 등록되어 있음
print(iris['target_names'])
# ['setosa' 'versicolor' 'virginica']

# 5. feature_names
print(iris['feature_names'])
# ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
# [ '꽃받침의 길이', '꽃받침의 폭', '꽃잎의 길이', '꽃잎의 폭' ]
```

K-Means Clustring

- **K-Means Clustring**

K-Means 알고리즘은 데이터 간의 유사성을 측정하는 기준으로 각 클러스터의 중심까지의 거리를 이용한다. 벡터 공간에 위치한 어떤 데이터에 대하여 k개의 클러스터가 주어졌을 때 클러스터의 중심까지 거리가 가장 가까운 클러스터로 해당 데이터를 할당한다. 다른 클러스터 간에는 서로 완전하게 구분하기 위하여 일정한 거리 이상 떨어져야 한다.

한편 몇 개의 클러스터로 데이터를 구분할 것인지를 결정하는 k값에 따라 모델의 성능이 달라진다. 일반적으로 k값이 클수록 모델의 정확도는 개선되지만, k값이 너무 커지면 선택지가 너무 많아지므로 분석의 효과가 사라진다.

K-Means Clustring

K-Means Clustring

K-Means Clustring는 1950년에 한 논문에서 처음 발표한 방법이다.
오래된 방법이지만, 계산이 간단하고 직관적으로 알기 쉽기 때문에 지금도 많이 사용하는 알고리즘이다.

K-Means Clustring의 장점

- 일반적인 군집화에서 가장 많이 활용하는 알고리즘이다.
- 알고리즘이 쉽고 간결하다.

K-Means Clustring의 단점

- 거리 기반 알고리즘으로 속성의 개수가 매우 많을 경우 군집화 정확도가 떨어진다.
- 반복을 수행하는데 , 반복 횟수가 많을 경우 수행 시간이 매우 느려진다.
- 몇 개의 군집(cluster)을 선택해야 할 지 가이드하기가 어렵다.

K-Means Clustring

- **K-Means Clustring 절차**

K-Means는 군집화(Clustring)에서 가장 일반적으로 사용되는 알고리즘이다.
K-Means는 군집 중심점(centroid)이라는 특정한 임의의 지점을 선택해 해당 중심에 가장 가까운 포인트들을 선택하는 군집화 기법이다.

군집 중심점은 선택된 포인트의 평균 지점으로 이동하고 이동된 중심점에서 다시 가까운 포인트를 선택, 다시 중심점을 평균 지점으로 이동하는 프로세스를 반복적으로 수행한다.
모든 데이터 포인트에서 더 이상 중심점의 이동이 없을 경우에 반복을 멈추고 해당 중심점에 속하는 데이터 포인트들을 군집화하는 기법이다.

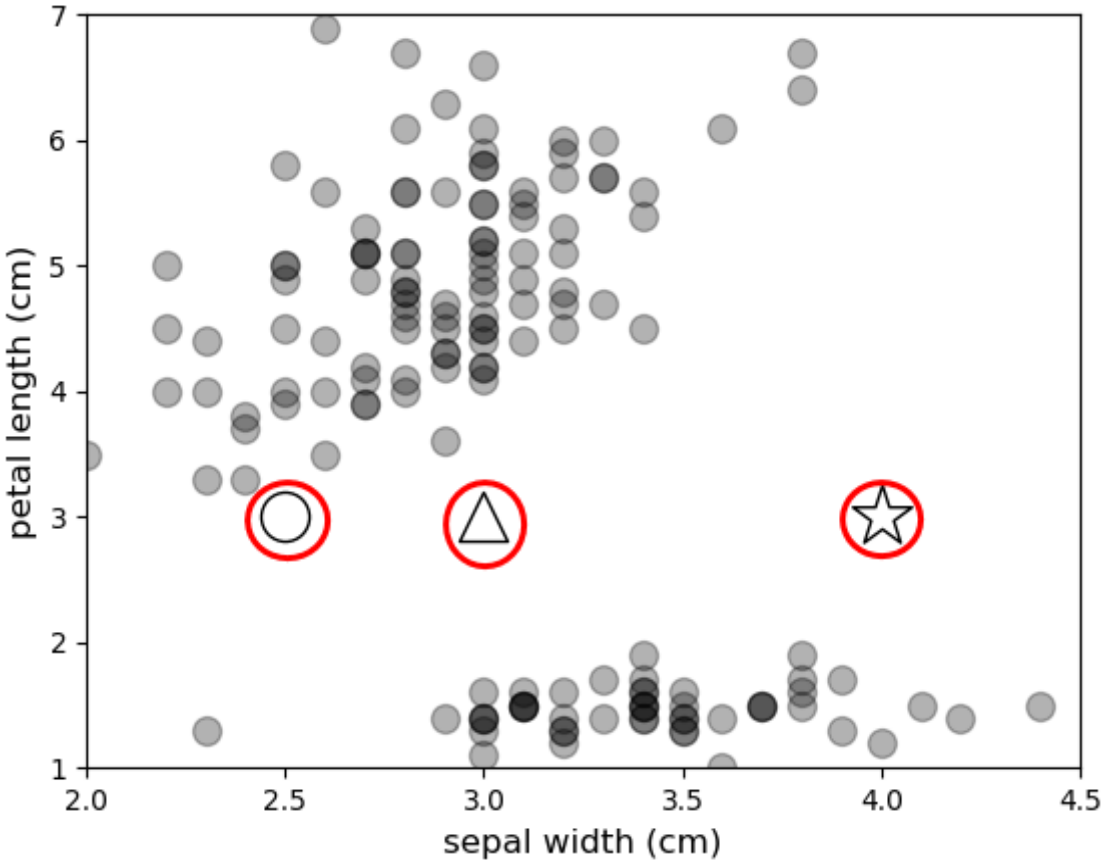
K-Means Clustring

K-Means Clustring 절차

1. 먼저 군집화의 기준이 되는 중심을 구성하려는 군집화 개수만큼 임의의 위치에 가져다 놓는다. 전체 데이터를 3개로 군집화하려면 3개의 중심을 임의의 위치에 가져다 놓는다.
2. 각 데이터는 가장 가까운 곳에 위치한 중심점에 소속된다.
3. 이렇게 소속이 결정되면 군집 중심점을 소속된 데이터의 평균 중심으로 이동한다.
4. 중심점이 이동했기 때문에 각 데이터는 기존에 속한 중심점보다 더 가까운 중심점이 있다면 해당 중심점으로 다시 소속을 변경한다.
5. 다시 중심을 소속된 데이터의 평균 중심으로 이동한다.
6. 중심점을 이동했는데 데이터의 중심점 소속 변경이 없으면 군집화를 종료한다. 그렇지 않으면 다시 4번 과정을 거쳐서 소속을 변경하고 이 과정을 반복한다.

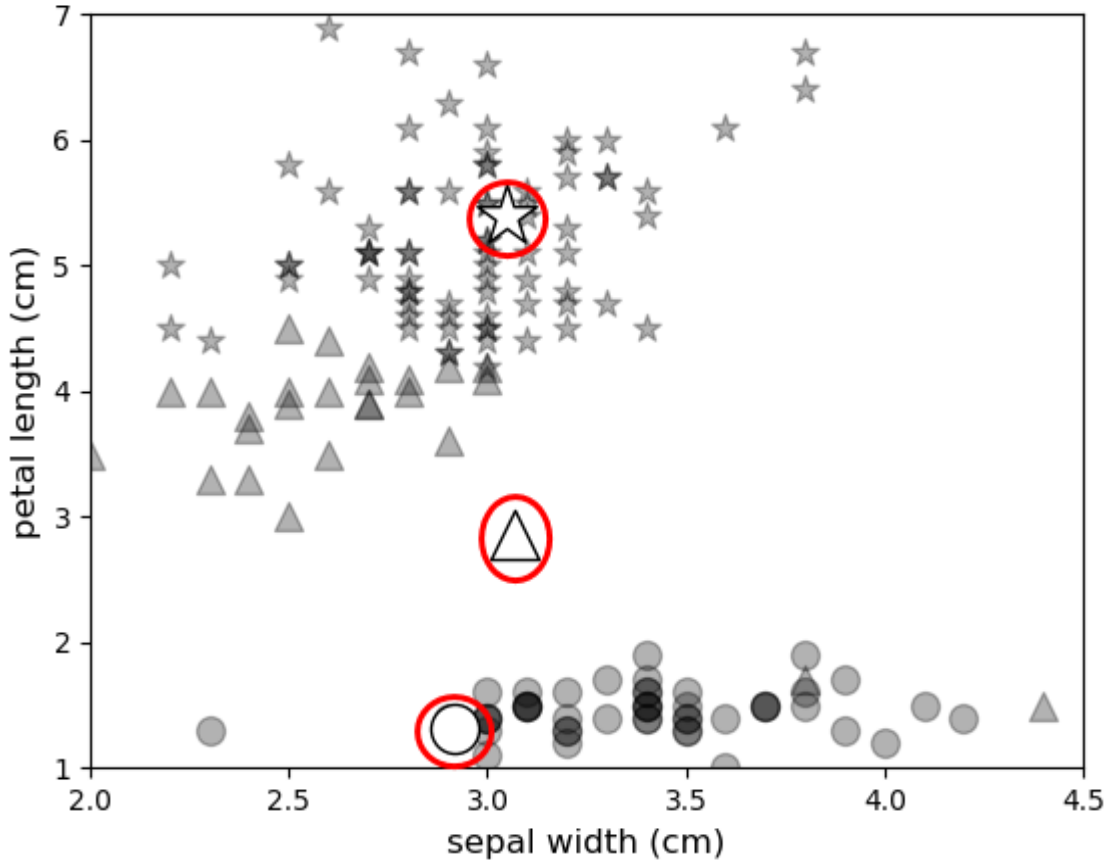
K-Means Clustring

- K-Means Clustring 절차 (1 / 3)



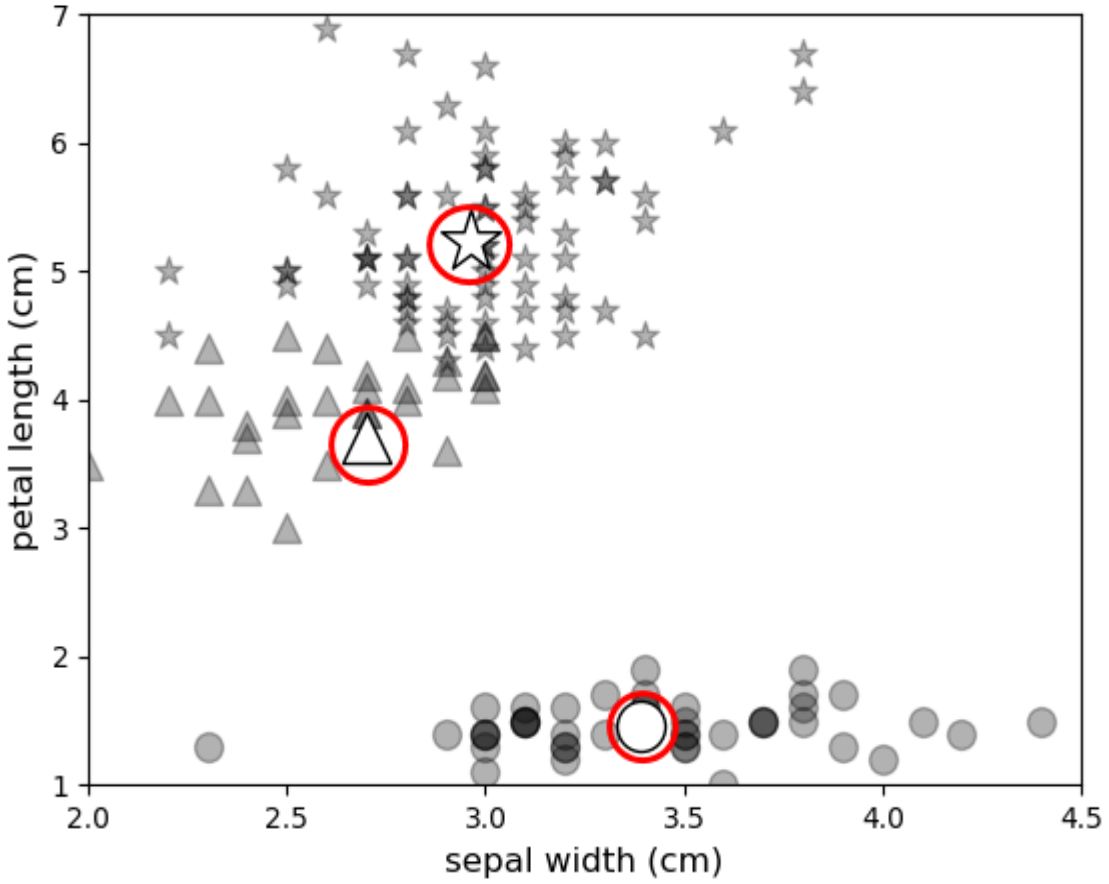
K-Means Clustring

- K-Means Clustring 절차 (2 / 3)



K-Means Clustring

- K-Means Clustring 절차 (3 / 3)



K-Means Clustring

- K-Means Clustring 예제

k_means.py (1 / 4)

```
from __future__ import unicode_literals
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster
from sklearn import datasets

# iris 데이터를 로드
iris = datasets.load_iris()
data = iris["data"]

# 초기 중심점을 정의 : 3개의 중심점 정의
init_centers=np.array([
        [4,2.5,3,0],
        [5,3   ,3,1],
        [6,4   ,3,2]])
```

K-Means Clustring

- K-Means Clustring 예제

k_means.py (2 / 4)

```
# 데이터 정의와 값 꺼내기
x_index = 1
y_index = 2

data_x=data[:,x_index]
data_y=data[:,y_index]

# 그래프의 스케일과 라벨 정의
x_max = 4.5
x_min = 2
y_max = 7
y_min = 1
x_label = iris["feature_names"][x_index]
y_label = iris["feature_names"][y_index]
```

K-Means Clustring

- K-Means Clustring 예제

k_means.py (3 / 4)

```
def show_result(cluster_centers,labels):
    # cluster 0과 중심점을 그리기
    plt.scatter(data_x[labels==0], data_y[labels==0],c='black' ,alpha=0.3,s=100,
                marker="o",label="cluster 0")
    plt.scatter(cluster_centers[0][x_index], cluster_centers[0][y_index],facecolors='white',
                edgecolors='black', s=300, marker="o")

    # cluster  1 과 중심점을 그리기
    plt.scatter(data_x[labels==1], data_y[labels==1],c='black' ,alpha=0.3,s=100,
                marker="^",label="cluster 1")
    plt.scatter(cluster_centers[1][x_index], cluster_centers[1][y_index],facecolors='white', edgecolors='black',
                s=300, marker="^")

    # cluster 와 중심점을 그리기
    plt.scatter(data_x[labels==2], data_y[labels==2],c='black' ,alpha=0.3,s=100, marker="*",label="cluster 2")
    plt.scatter(cluster_centers[2][x_index], cluster_centers[2][y_index],facecolors='white', edgecolors='black',
                s=500, marker="*")
```

K-Means Clustring

- K-Means Clustring 예제

k_means.py (4 / 4)

```
# def show_result(cluster_centers,labels):
# 그래프의 스케일과 축 라벨을 설정 : 함수안에서 출력
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.xlabel(x_label,fontsize='large')
    plt.ylabel(y_label,fontsize='large')
    plt.show()


# 초기 상태를 표시
labels=np.zeros(len(data),dtype=np.int)
show_result(init_centers,labels)

for i in range(5):
        model = cluster.KMeans(n_clusters=3,max_iter=1,init=init_centers).fit(data)
        labels = model.labels_
        init_centers=model.cluster_centers_
        show_result(init_centers,labels)
```

K-Means Clustring

- K-Means 알고리즘

Step1. 데이터 준비

Step2. 데이터 탐색

Step3. 데이터 전처리

Step4. K-Means 군집 모델 학습 및 시각화

K-Means Clustring

Step1. 데이터 준비

UCI(University of California, Irvine) 머신러닝 저장소에서 제공하는 도매업 고객 (wholesale customers) 데이터셋을 사용한다.
도매업 고객 데이터셋은 각 고객의 연간 구매금액을 상품 카테고리별로 구분하여 정리한 데이터이다. 모두 8개의 열에 440개의 관측값이 행으로 되어 있다.
첫 2개 열은 상품 구매금액이 아니라 고객의 일반 정보를 담고 있다.
(Channel 열은 호텔/레스토랑 또는 소매점 등 판매채널 값, Region 열은 고객 소재지)

kmeans_clustering.py (1)

```
#기본 라이브러리 불러오기
import pandas as pd
import matplotlib.pyplot as plt

# UCI 저장소에서 도매업 고객(wholesale customers) 데이터셋 가져오기
uci_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/₩
00292/Wholesale%20customers%20data.csv'
df = pd.read_csv(uci_path, header=0)
print(df)                          # [440 rows x 8 columns]
```

K-Means Clustring

- Step2. 데이터 탐색

kmeans_clustering.py (2)

```
# 데이터 살펴보기
print(df.head())
print('₩n')

# 데이터 자료형 확인
print(df.info())
print('₩n')

# 데이터 통계 요약정보 확인
print(df.describe())
print('₩n')
```

K-Means Clustring

- Step2. 데이터 탐색 : 결과

```
# 데이터 살펴보기
print(df.head())
# Channel 열은 호텔/레스토랑 또는 소매점 등 판매채널 값
# Region 열은 고객 소재지
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

K-Means Clustring

- Step3. 데이터 전처리

kmeans_clustering.py (3)

```
# 데이터 분석에 사용할 속성(열, 변수)을 선택
# k-means는 비지도 학습 모델이기 때문에 예측(종속)변수를 지정할 필요가 없고
# 모두 설명(독립)변수만 사용한다.
x = df.iloc[:, :]                    # 데이터프레임의 모든 데이터 : 8개의 설명변수
print(x[:5])                         # 처음부터 5개의 데이터 구해서 출력
print('₩n')

# 설명 변수 데이터를 정규화
# 학습 데이터를 정규화를 하면 서로 다른 변수 사이에 존재할 수 있는 데이터 값의
# 상대적 크기 차이에서 발생하는 오류를 제거할 수 있다.
from sklearn import preprocessing
x = preprocessing.StandardScaler().fit(x).transform(x)
print(x[:5])
```

K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화

kmeans_clustering.py (4)

```
# sklearn 라이브러리에서 cluster 군집 모델 가져오기
from sklearn import cluster

# k-means 모델 객체 생성
# k-means 모델은 8개의 속성(변수)을 이용하여 각 관측값을 5개의 클러스터로 구분
# 클러스터의 갯수를 5개로 설정 : n_clusters=5
kmeans = cluster.KMeans(n_clusters=5)

# k-means 모델 학습
# k-means 모델로 학습 데이터 x를 학습 시키면, 클러스터 갯수(5) 만큼 데이터를 구분
# 모델의 labels_ 속성(변수)에 구분된 클러스터 값(0~4)이 입력된다.
kmeans.fit(x)
```

K-Means Clustring

❖ Step4. K-Means 군집 모델 학습 및 시각화

kmeans_clustering.py (5)

```
# 예측 (군집) 결과를 출력할 열(속성)의 값 구하기
# 변수 labels_ 에 저장된 값을 출력해보면, 0~4 범위의 5개 클러스터 값이 출력됨
# 각 데이터가 어떤 클러스터에 할당 되었는지를 확인 할 수 있다.
# (매번 실행 할때 마다 예측값의 결과가 달라진다.)
cluster_label = kmeans.labels_
print(cluster_label)                    # 0~4 범위의 5개의 클러스터
print('₩n')

# 예측(군집) 결과를 저장할 열(Cluster)을 데이터프레임에 추가
df['Cluster'] = cluster_label
print(df.head())                        # 9개의 열 출력 : Cluster 열 추가됨
```

K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화 : 결과

```
# 예측 (군집) 결과를 출력할 열(속성)의 값 구하기
# 변수 labels_ 에 저장된 값을 출력해보면, 0~4 범위의 5개 클러스터 값이 출력됨
# 각 데이터가 어떤 클러스터에 할당 되었는지를 확인 할 수 있다.
# (매번 실행 할때 마다 예측값의 결과가 달라진다.)
cluster_label = kmeans.labels_
print(cluster_label)                    # 0~4 범위의 5개의 클러스터
```

```
[2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 2 1 2 1 2 1 1 4 2 2 1 1 2 1 1 1 1 1 1 2 1
 2 2 1 1 1 2 2 2 2 2 4 2 2 1 1 2 2 1 1 4 2 1 1 2 4 2 2 1 4 1 2 1 1 1 1 1 2
 2 1 1 2 1 1 1 2 2 1 2 4 4 1 1 1 1 1 4 1 2 1 2 1 1 1 2 2 2 1 1 1 2 2 2 2 1
 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
 1 1 1 1 1 1 1 2 2 1 2 2 2 1 1 2 2 2 2 1 1 1 2 2 1 2 1 2 1 1 1 1 1 4 1 3 1
 1 1 1 2 2 1 1 1 2 1 1 0 2 0 0 2 2 0 0 0 2 0 0 0 2 0 4 0 0 2 0 2 0 2 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 4 0 0 0 0 0 0 0
 0 0 0 0 0 2 0 2 0 2 0 0 0 0 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 0 2
 0 2 2 0 2 2 2 2 2 2 2 0 0 2 0 0 2 0 0 2 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 2 0
 4 0 2 0 0 0 0 2 2 1 2 1 1 2 2 1 2 1 2 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1
 1 1 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
 2 2 1 1 1 1 1 1 2 2 1 2 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1]
```

K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화 : 결과

```
# 예측(군집) 결과를 저장할 열(Cluster)을 데이터프레임에 추가
df['Cluster'] = cluster_label
print(df.head())                    # 9개의 열 출력 : Cluster 열 추가됨
```

	Channel	Region	Fresh	Milk	...	Frozen	Detergents_Paper	Delicassen	Cluster
0	2	3	12669	9656	...	214	2674	1338	2
1	2	3	7057	9810	...	1762	3293	1776	2
2	2	3	6353	8808	...	2405	3516	7844	2
3	1	3	13265	1196	...	6404	507	1788	1
4	2	3	22615	5410	...	3915	1777	5185	2

K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화

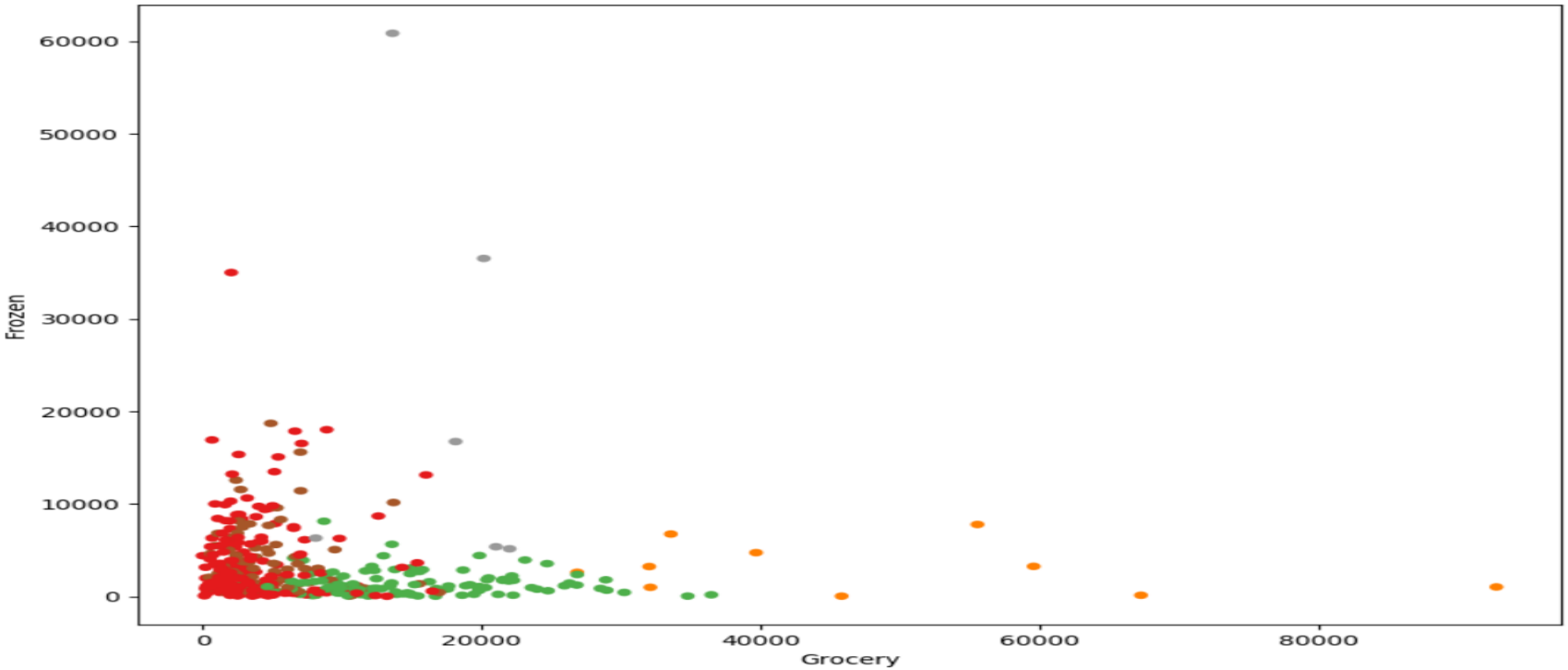
kmeans_clustering.py (6)

```
# 그래프로 시각화 - 클러스터 값 : 0 ~ 4 모두 출력
# 8개의 변수를 하나의 그래프로 표현할 수 없기 때문에 2개의 변수를 선택하여
# 관측값의 분포를 그려보자.
# 모델의 예측값은 매번 실행할 때마다 달라지므로, 그래프의 형태도 달라진다.
# 산점도 : x='Grocery', y='Frozen'                    식료품점 - 냉동식품
# 산점도 : x='Milk', y='Delicassen'                   우유 - 조제식품점
df.plot(kind='scatter', x='Grocery', y='Frozen', c='Cluster', cmap='Set1',
        colorbar=False, figsize=(10, 10))          # colorbar 미적용
df.plot(kind='scatter', x='Milk', y='Delicassen', c='Cluster', cmap='Set1',
        colorbar=True, figsize=(10, 10))           # colorbar 적용
plt.show()
plt.close()
```

K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화 : 결과

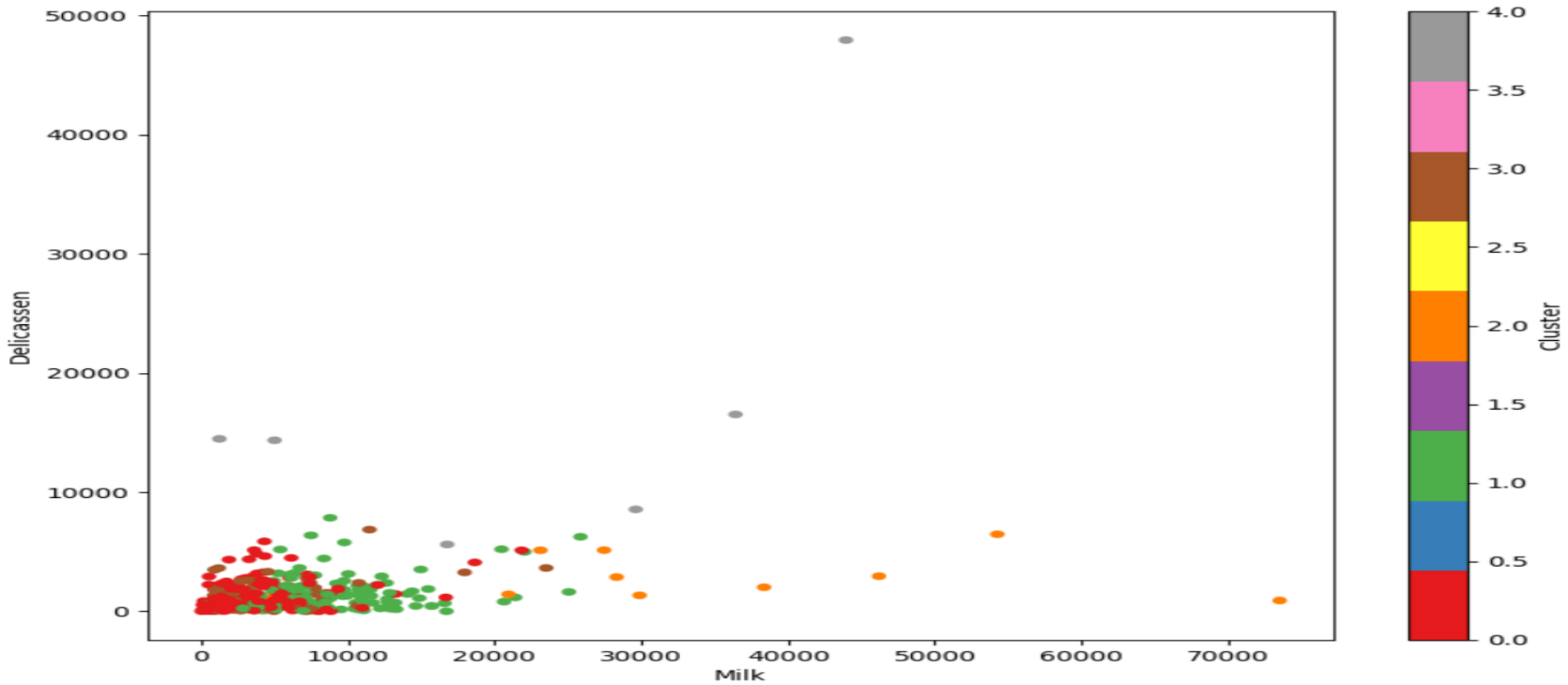
```
# 그래프로 시각화 - 클러스터 값 : 0 ~ 4 모두 출력
# 산점도 : x='Grocery', y='Frozen'                    식료품점 - 냉동식품
df.plot(kind='scatter', x='Grocery', y='Frozen', c='Cluster', cmap='Set1',
        colorbar=False, figsize=(10, 10))          # colorbar 미적용
```



K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화 : 결과

```
# 그래프로 시각화 - 클러스터 값 : 0 ~ 4 모두 출력
# 산점도 : x='Milk', y='Delicassen'                    우유 - 조제식품점
df.plot(kind='scatter', x='Milk', y='Delicassen', c='Cluster', cmap='Set1',
        colorbar=True, figsize=(10, 10))          # colorbar 적용
```



K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화

kmeans_clustering.py (7)

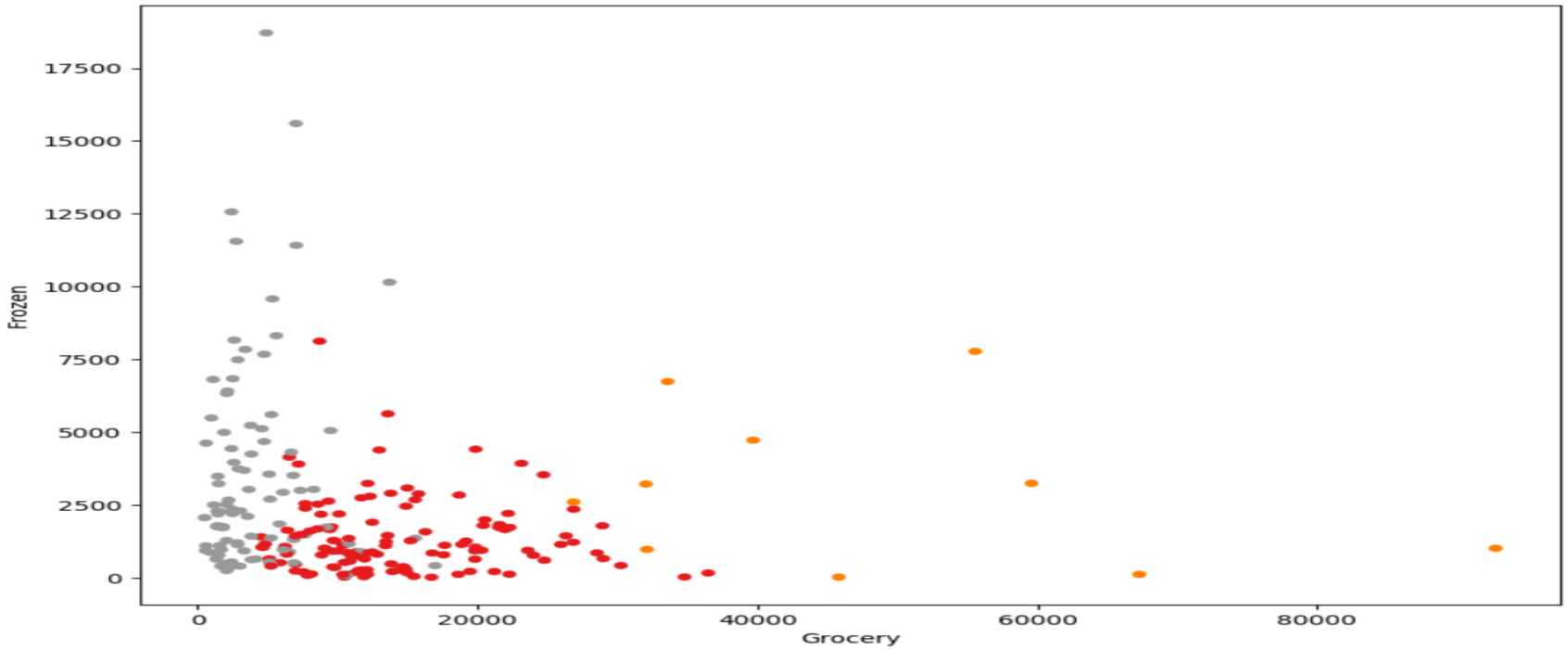
```
# 그래프로 시각화 - 클러스터 값 : 1, 2, 3 확대해서 자세하게 출력
# 다른 값들에 비해 지나치게 큰 값으로 구성된 클러스터(0, 4)를 제외
# 데이터들이 몰려 있는 구간을 확대해서 자세하게 분석
# 클러스터 값이 1, 2, 3에 속하는 데이터만 변수 ndf에 저장함
mask = (df['Cluster'] == 0) | (df['Cluster'] == 4)
ndf = df[~mask]

# 클러스터 값이 1, 2, 3에 속하는 데이터만을 이용해서 분포를 확인
# 산점도 : x='Grocery', y='Frozen'                식료품점 - 냉동식품
# 산점도 : x='Milk', y='Delicassen'               우유 - 조제식품점
ndf.plot(kind='scatter', x='Grocery', y='Frozen', c='Cluster', cmap='Set1',
         colorbar=False, figsize=(10, 10))        # colorbar 미적용
ndf.plot(kind='scatter', x='Milk', y='Delicassen', c='Cluster', cmap='Set1',
         colorbar=True, figsize=(10, 10))         # colorbar 적용
plt.show()
plt.close()
```

K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화 : 결과

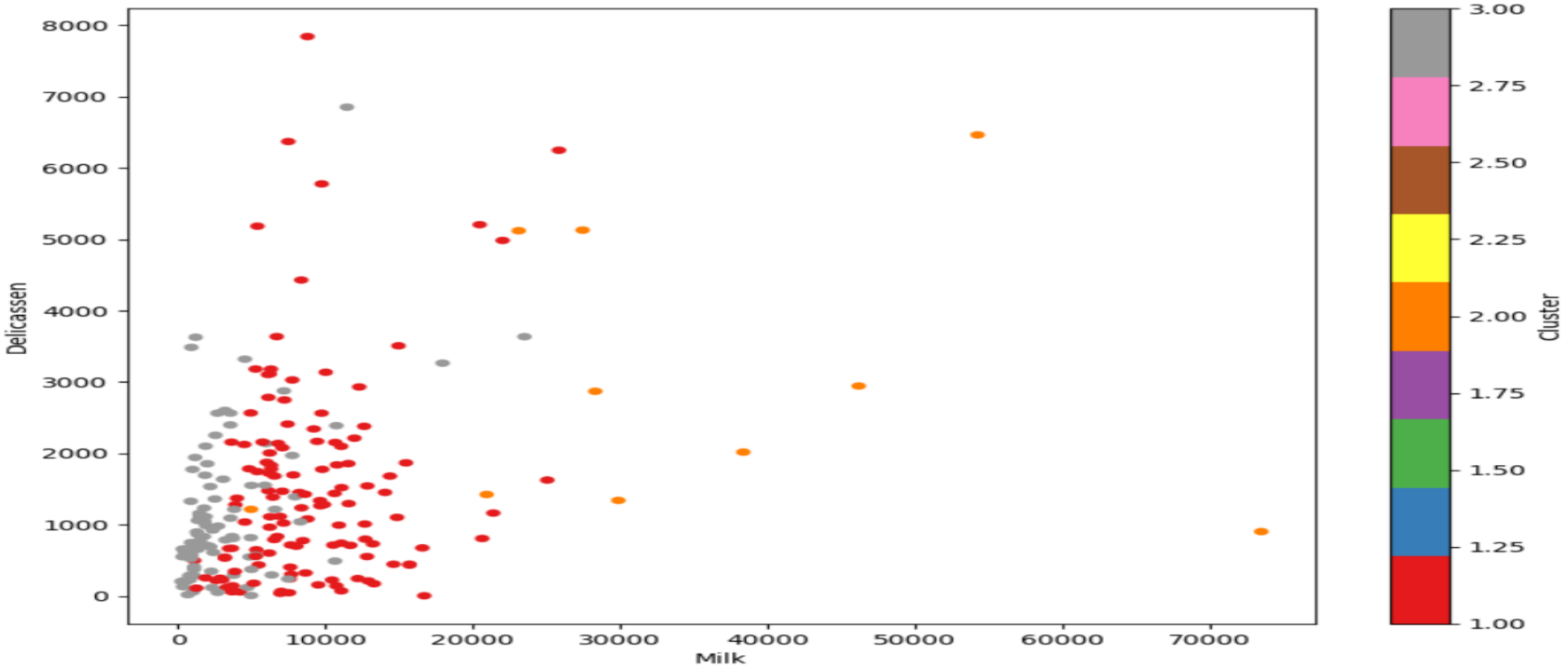
```
# 그래프로 시각화 - 클러스터 값 : 1, 2, 3 확대해서 자세하게 출력
# 산점도 : x='Grocery', y='Frozen'                    식료품점 - 냉동식품
ndf.plot(kind='scatter', x='Grocery', y='Frozen', c='Cluster', cmap='Set1',
         colorbar=False, figsize=(10, 10))          # colorbar 미적용
```



K-Means Clustring

- Step4. K-Means 군집 모델 학습 및 시각화 : 결과

```
# 그래프로 시각화 - 클러스터 값 : 1, 2, 3 확대해서 자세하게 출력
# 산점도 : x='Milk', y='Delicassen'                    우유 - 조제식품점
ndf.plot(kind='scatter', x='Milk', y='Delicassen', c='Cluster', cmap='Set1',
         colorbar=True, figsize=(10, 10))          # colorbar 적용
```



DBSCAN Clustring

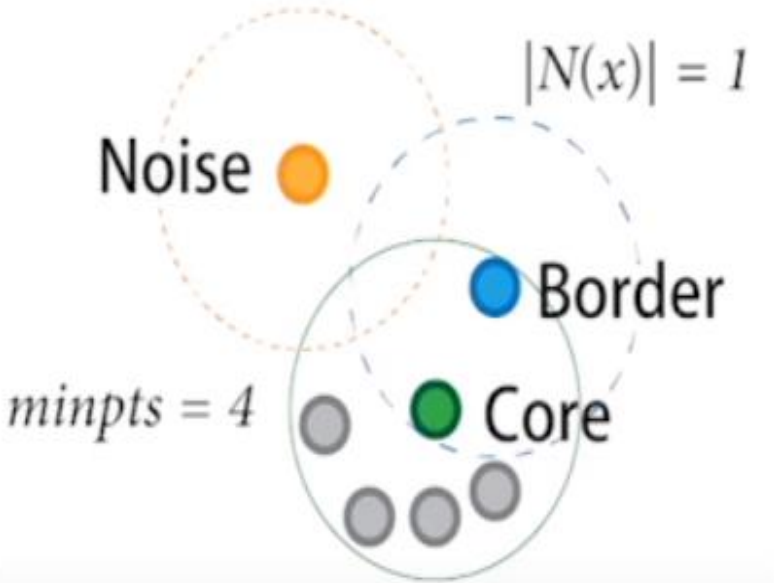
- **DBSCAN Clustring (밀도 기반 클러스터링) 알고리즘**

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)은 데이터가 위치하고 있는 공간 밀집도를 기준으로 클러스터를 구분한다.
자기를 중심으로 반지름 R(epsilon)의 공간에 최소 M개의 포인트가 존재하는 점을 코어 포인트(core point)라고 부른다.
코어 포인트는 아니지만 반지름 R안에 다른 코어 포인트가 있을 경우에 경계 포인트(border point)라고 부른다.
코어 포인트도 아니고 경계 포인트에도 속하지 않는 점을 noise(또는 outlier)라고 부른다.
하나의 클러스터는 반지름이 R안에 서로 위치하는 모든 코어 포인트를 포함하는 방식으로 구성된다. 당연히 각 코어 포인트 주위에 있는 경계 포인트를 포함한다. 서로 밀접한 데이터끼리 하나의 클러스터를 구성하게 되고 어느 클러스터에도 속하지 않는 점들은 noise로 남게 된다.

DBSCAN Clustring

- **DBSCAN Clustring (밀도 기반 클러스터링) 알고리즘**

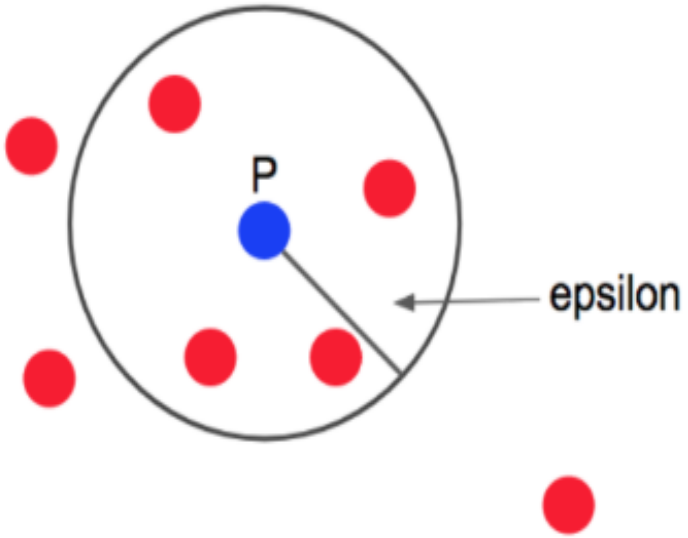
- DBSCAN 알고리즘은 밀도가 높은 부분으로 클러스터링 하는 방식이다.
- 반경과 점의 수로 군집을 만든다.
- 반경 epsilon과 최소 점의 수인 minpts(4) 를 정한다.



DBSCAN Clustring

❖ DBSCAN Clustring (밀도 기반 클러스터링) 알고리즘

- 하나의 점에서 반경 epsilon안에 존재하는 점의 수를 센다. 이때 반경 안에 속한 점이 minpts로 정한 수(minpts=4) 이상이면 해당 점은 core point 라고 부른다.
- 현재 점 p에서 4개 이상의 점이 반경 안에 있기 때문에 p는 core point 가 된다.



DBSCAN Clustring

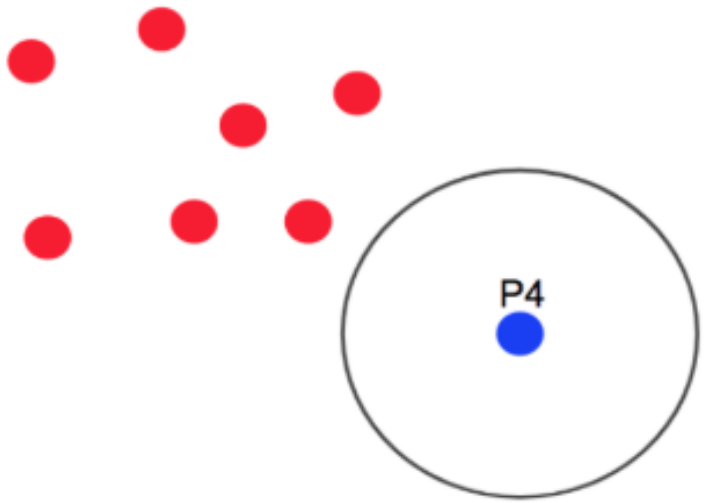
- **DBSCAN Clustring (밀도 기반 클러스터링) 알고리즘**

- Core point에 속한 점들이 4개 미만의 점이 속하게 되면, 해당 점은 border point라고 부른다.
- P2는 반경 epsilon 안에 3개의 점만 존재하므로 minpts=4 미만이기 때문에 border point 가 된다.

DBSCAN Clustring

❖ DBSCAN Clustring (밀도 기반 클러스터링) 알고리즘

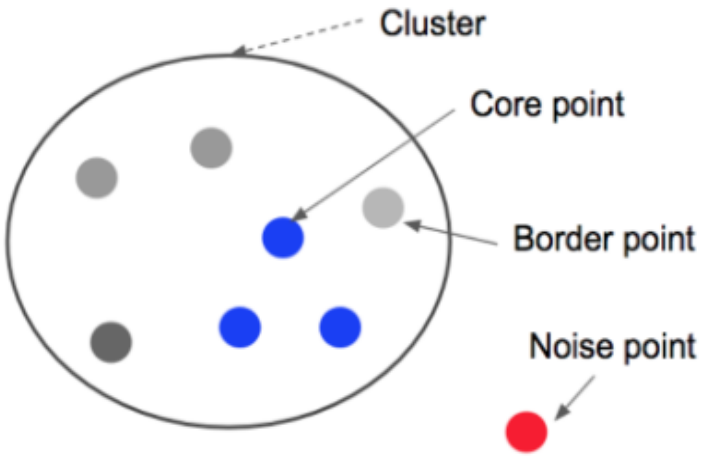
- 하나의 점에서 반경 epsilon을 확인 했을때 어느 군집에도 속하지 않는 점들을 outlier(이상치)라 하고, 각 점들을 noise point 라고 한다.
- P4는 반경 안에 속하는 점이 아무것도 없으므로 noise point 가 된다.
- DBSCAN 알고리즘은 이와 같이 군집에 포함되지 않는 outlier 검출에 효율적이다.



DBSCAN Clustring

❖ DBSCAN Clustring (밀도 기반 클러스터링) 알고리즘

➢ 초반에 지정한 epsilon 반경 안에 minpts 이상의 점으로 구성된다면, 해당 점을 중심으로 군집이 형성되고, core point로 지정한다. core point가 서로 다른 core point군집의 일부가 되면 서로 연결되어 하나의 군집이 형성된다.
이때 군집에는 속해 있지만 core point가 아닌 점들을 border point라고 하며, 아무곳에도 속하지 않는 점은 noise point가 된다.



DBSCAN Clustring

DBSCAN 알고리즘의 장점

- 클러스터의 수를 미리 정하지 않아도 된다.
- 다양한 모양의 크기의 클러스터를 얻는 것이 가능하다.
- 모양이 기하학적인 분포라도, 밀도 여부에 따라 군집도를 찾을 수 있다.
- Outlier 검출을 통해 필요하지 않는 noise 데이터를 검출하는 것이 가능하다.

DBSCAN 알고리즘의 단점

- 반경(epsilon)으로 설정한 값에 민감하게 작용한다.
  DBSCAN 알고리즘을 사용하려면 적절한 epsilon 값을 설정하는 것이 중요하다.

DBSCAN Clustring

- DBSCAN 알고리즘

Step1. 데이터 준비

Step2. 데이터 탐색 및 중학교 위치를 지도 파일로 저장

Step3. 데이터 전처리

Step4. DBSCAN 군집 모델 학습 및 시각화

DBSCAN Clustring

- Step1. 데이터 준비

학교알리미 공개용 데이터 중에서 서울시 중학교 졸업생의 진로현황 데이터셋을 사용하여 고등학교 진학률이 비슷한 중학교끼리 군집(cluster)을 만들어 보자

dbscan_clustering.py (1)

```
# 기본 라이브러리 불러오기
import pandas as pd
import folium

# 학교알리미 공개용 데이터 중에서 서울시 중학교 졸업생의 진로현황 데이터셋
file_path = '2016_middle_shcool_graduates_report.xlsx'
df = pd.read_excel(file_path, header=0)                    # [415 rows x 21 columns]

# IPython Console 디스플레이 옵션 설정하기
pd.set_option('display.width', None)                       # 출력화면의 너비
pd.set_option('display.max_rows', 100)                     # 출력할 행의 개수 한도
pd.set_option('display.max_columns', 30)                   # 출력할 열의 개수 한도
pd.set_option('display.max_colwidth', 20)                  # 출력할 열의 너비
pd.set_option('display.unicode.east_asian_width', True)    # 유니코드 사용 너비 조정

#  데이터프레임의 열 이름 출력
print(df.columns.values)
```

DBSCAN Clustring

- Step2. 데이터 탐색 및 중학교 위치를 지도 파일로 저장

dbscan_clustering.py (2)

```
# 데이터 살펴보기
print(df.head())
print('₩n')

# 데이터 자료형 확인
print(df.info())
print('₩n')

# 데이터 통계 요약정보 확인
print(df.describe())
print('₩n')
```

DBSCAN Clustring

- Step2. 데이터 탐색 및 중학교 위치를 지도 파일로 저장

dbscan_clustering.py (3)

```
# 지도에 위치 표시
mschool_map = folium.Map(location=[37.55,126.98], tiles='Stamen Terrain',
                                zoom_start=12)

# 중학교 위치정보를 CircleMarker로 표시
for name, lat, lng in zip(df.학교명, df.위도, df.경도):
    folium.CircleMarker([lat, lng],
                        radius=5,              # 원의 반지름
                        color='brown',         # 원의 둘레 색상
                        fill=True,
                        fill_color='coral',    # 원을 채우는 색
                        fill_opacity=0.7,      # 투명도
                        popup=name             # 팝업 기능(원형 마커를 클릭하면 학교명이 팝업으로 출력)
    ).add_to(mschool_map)

# 지도를 html 파일로 저장하기
mschool_map.save('seoul_mschool_location.html')
```

DBSCAN Clustring

- Step2. 데이터 탐색 및 중학교 위치를 지도 파일로 저장 : 결과

```
# 데이터 살펴보기
print(df.head())
```

```
   Unnamed: 0    지역                          학교명  코드  유형  ...  기타진학  취업    미상
0           0   성북구  서울대학교사범대학부설중학교.....     3  국립  ...     0.004     0  0.000
1           1   종로구  서울대학교사범대학부설여자중학교...     3  국립  ...     0.031     0  0.000
2           2   강남구              개원중학교     3  공립  ...     0.009     0  0.003
3           3   강남구              개포중학교     3  공립  ...     0.019     0  0.000
4           4   서초구              경원중학교     3  공립  ...     0.010     0  0.000

[5 rows x 21 columns]
```

DBSCAN Clustring

❖ Step2. 데이터 탐색 및 중학교 위치를 지도 파일로 저장 : 결과

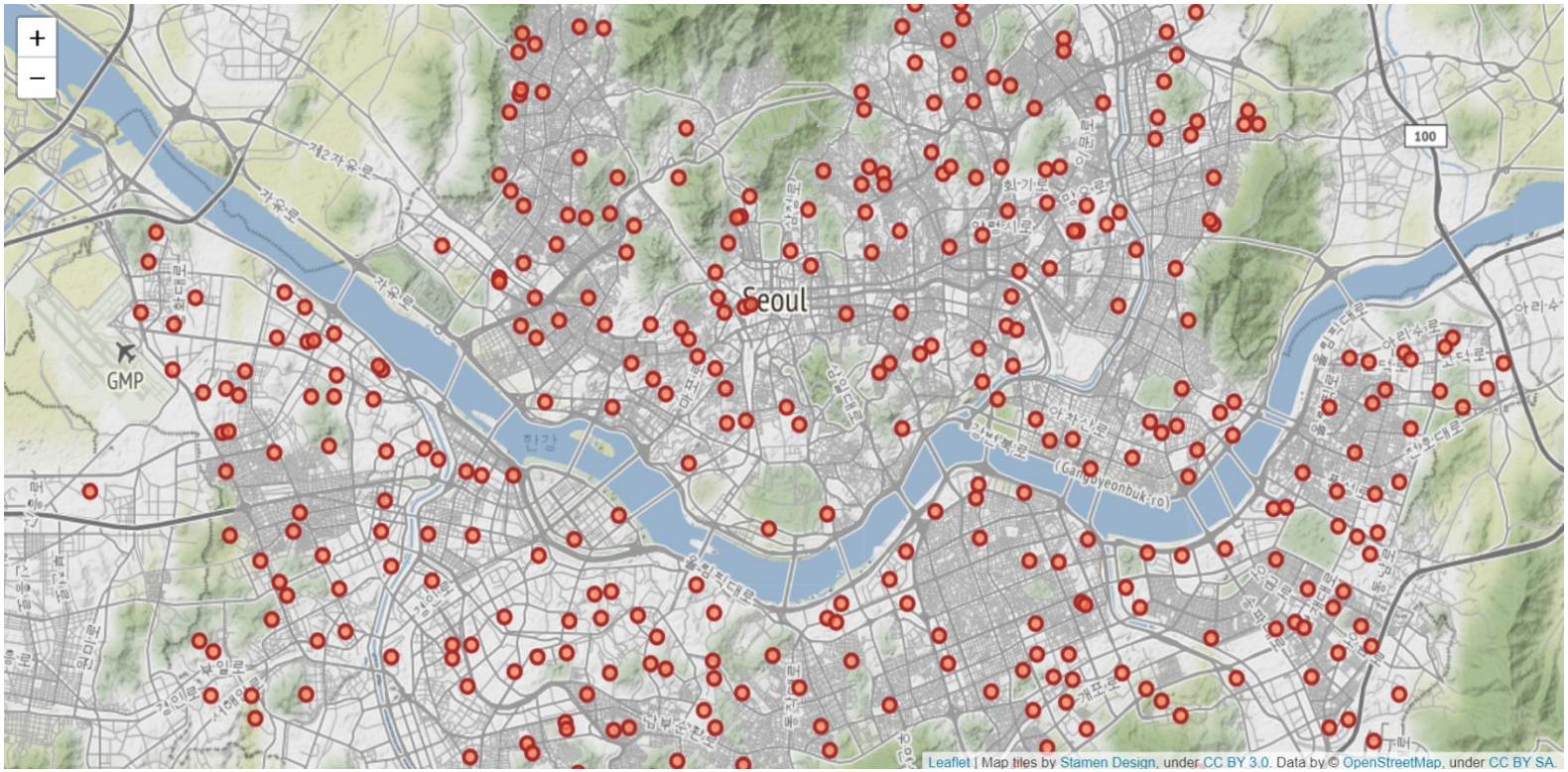
```
# 데이터 자료형 확인
print(df.info())
```

```
RangeIndex: 415 entries, 0 to 414
Data columns (total 21 columns):
Unnamed: 0    415 non-null int64
지역            415 non-null object
학교명           415 non-null object
코드            415 non-null int64
유형            415 non-null object
주야            415 non-null object
남학생수          415 non-null int64
여학생수          415 non-null int64
일반고           415 non-null float64
특성화고          415 non-null float64
과학고           415 non-null float64
외고_국제고        415 non-null float64
예고_체고         415 non-null float64
마이스터고         415 non-null float64
자사고           415 non-null float64
자공고           415 non-null float64
기타진학          415 non-null float64
취업            415 non-null int64
미상            415 non-null float64
위도            415 non-null float64
경도            415 non-null float64
dtypes: float64(12), int64(5), object(4)
```

DBSCAN Clustring

- Step2. 데이터 탐색 및 중학교 위치를 지도 파일로 저장 : 결과

서울 중학교 위치 파일 seoul_mschool_location.html 실행



DBSCAN Clustring

- Step3. 데이터 전처리

dbscan_clustering.py (4)

```
# 지역, 코드, 유형, 주야 열을 원핫인코딩 처리
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()                    # label encoder 생성

# 모델이 인식할 수 없는 문자형 데이터를 원핫인코딩으로 처리하여 더미 변수에 저장
onehot_location = label_encoder.fit_transform(df['지역'])       # 지역구 이름
onehot_code = label_encoder.fit_transform(df['코드'])           # 3, 5, 9
onehot_type = label_encoder.fit_transform(df['유형'])           # 국립, 공립, 사립
onehot_day = label_encoder.fit_transform(df['주야'])            # 주간, 야간

# 원핫인코딩된 결과를 새로운 열(변수)에 할당
df['location'] = onehot_location                                # 지역
df['code'] = onehot_code                                        # 코드
df['type'] = onehot_type                                        # 유형
df['day'] = onehot_day                                          # 주야

print(df.head())
```

DBSCAN Clustring

❖ Step3. 데이터 전처리 : 결과

dbscan_clustering.py (4)

```
# 지역, 코드, 유형, 주야 열을 원핫인코딩 처리 결과 확인
print(df.head())
```

	Unnamed: 0	지역	학교명	코드	유형	...	경도	location	code	type	day
0	0	성북구	서울대학교사범대학부설중학교.....	3	국립	...	127.038909	16	0	1	0
1	1	종로구	서울대학교사범대학부설여자중학교...	3	국립	...	127.003857	22	0	1	0
2	2	강남구	개원중학교	3	공립	...	127.071744	0	0	0	0
3	3	강남구	개포중학교	3	공립	...	127.062201	0	0	0	0
4	4	서초구	경원중학교	3	공립	...	127.008900	14	0	0	0

[5 rows x 25 columns]

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1

dbscan_clustering.py (5)

```
# sklearn 라이브러리에서 cluster 군집 모델 가져오기
from sklearn import cluster

# 분석1. 과학고, 외고국제고, 자사고 진학률로 군집
# 분석에 사용할 속성을 선택 (과학고, 외고국제고, 자사고 진학률)
print('분석1. 과학고, 외고국제고, 자사고 진학률로 군집')
columns_list = [10, 11, 14]                                    # 각 컬럼의 인덱스 번호
x = df.iloc[:, columns_list]
print(x[:5])
print('₩n')

# 설명 변수 데이터를 정규화
x = preprocessing.StandardScaler().fit(x).transform(x)

# DBSCAN 모델 객체 생성
# 밀도 계산의 기준이 되는 반지름 R(eps=0.2)과 최소 포인트 개수 M(min_samples=5) 설정
dbm = cluster.DBSCAN(eps=0.2, min_samples=5)

# DBSCAN 모델 학습
dbm.fit(x)
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1

dbscan_clustering.py (6)

```
# 예측 (군집) 결과를 출력할 열(속성)의 값 구하기
# 모델의 labels_ 속성으로 확인하면 5개의 클러스터 값 ( -1, 0, 1, 2, 3 ) 으로 나타남
cluster_label = dbm.labels_
print(cluster_label)                                    # -1, 0, 1, 2, 3
print('₩n')

# 예측(군집) 결과를 저장할 열(Cluster)을 데이터프레임에 추가
df['Cluster'] = cluster_label                           # Cluster 열 추가됨
print(df.head())
print('₩n')

# 클러스터 값으로 그룹화하고, 그룹별로 내용 출력 (첫 5행만 출력)
grouped_cols = [1, 2, 4] + columns_list         # 1:지역명, 2:학교명, 4:유형
grouped = df.groupby('Cluster')
for key, group in grouped:
    print('* key :', key)                               # 클러스터  값: -1, 0, 1, 2, 3
    print('* number :', len(group))                     # 각 클러스터 속한 학교수
    print(group.iloc[:, grouped_cols].head())     # 5개의 데이터 출력
    print('₩n')
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1

dbscan_clustering.py (7)

```
# 그래프로 표현 - 시각화 ( 각 Cluster 번호별로 보여줄 색깔 정의 )
colors = {-1:'gray', 0:'coral', 1:'blue', 2:'green', 3:'red', 4:'purple',
          5:'orange', 6:'brown', 7:'brick', 8:'yellow', 9:'magenta', 10:'cyan'}

cluster_map = folium.Map(location=[37.55,126.98], tiles='Stamen Terrain',
                         zoom_start=12)

for name, lat, lng, clus in zip(df.학교명, df.위도, df.경도, df.Cluster):
    folium.CircleMarker([lat, lng],
                        radius=5,                    # 원의 반지름
                        color=colors[clus],          # 원의 둘레 색상
                        fill=True,
                        fill_color=colors[clus],     # 원을 채우는 색
                        fill_opacity=0.7,            # 투명도
                        popup=name
    ).add_to(cluster_map)

# 지도를 html 파일로 저장하기
cluster_map.save('seoul_mschool_cluster.html')
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

```
print('분석1. 과학고, 외고국제고, 자사고 진학률로 군집')
columns_list = [10, 11, 14]          # 각 컬럼의 인덱스 번호
x = df.iloc[:, columns_list]
print(x[:5])
```

```
분석1. 과학고, 외고국제고, 자사고 진학률로 군집
   과학고  외고_국제고  자사고
0  0.018      0.007  0.227
1  0.000      0.035  0.043
2  0.009      0.012  0.090
3  0.013      0.013  0.065
4  0.007      0.010  0.282
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

```
# 예측(군집) 결과를 저장할 열(Cluster)을 데이터프레임에 추가
df['Cluster'] = cluster_label                    # Cluster 열 추가됨
print(df.head())
```

```
   Unnamed: 0   지역                학교명  코드  유형  ...  location  code  type  day  Cluster
0           0  성북구  서울대학교사범대학부설중학교.....     3  국립  ...        16     0     1    0       -1
1           1  종로구  서울대학교사범대학부설여자중학교...     3  국립  ...        22     0     1    0       -1
2           2  강남구          개원중학교     3  공립  ...         0     0     0    0       -1
3           3  강남구          개포중학교     3  공립  ...         0     0     0    0       -1
4           4  서초구          경원중학교     3  공립  ...        14     0     0    0       -1

[5 rows x 26 columns]
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

```
# 클러스터 값으로 그룹화하고, 그룹별로 내용 출력 (첫 5행만 출력)
grouped_cols = [1, 2, 4] + columns_list          # 1:지역명, 2:학교명, 4:유형
grouped = df.groupby('Cluster')
for key, group in grouped:
    print('* key :', key)                        # 클러스터  값: -1, 0, 1, 2, 3
    print('* number :', len(group))              # 각 클러스터 속한 학교수
    print(group.iloc[:, grouped_cols].head())    # 5개의 데이터 출력
```

클러스터 0 : 외고_국제고와 자사고 합격률은 높지만 과학고 합격자가 없다.
클러스터 1 : 자사고 합격자만 존재하는 그룹
클러스터 2 : 자사고 합격률이 매우 높으면서 과학고와 외고_국제고 합격자도 일부 존재한다.
클러스터 3 : 과학고 합격자 없이 외고_국제고와 자사고 합격자를 배출한 점은 클러스터 0과 비슷하지만, 외고_국제고 합격률이 클러스터 0에 비해 현저하게 낮다.

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

클러스터 0 : 외고_국제고와 자사고 합격률은 높지만 과학고 합격자가 없다.

* key : 0
* number : 102

	지역	학교명	유형	과학고	외고_국제고	자사고
13	서초구	동덕여자중학교	사립	0.0	0.022	0.038
22	강남구	수서중학교	공립	0.0	0.019	0.044
28	서초구	언남중학교	공립	0.0	0.015	0.050
34	강남구	은성중학교	사립	0.0	0.016	0.065
43	송파구	거원중학교	공립	0.0	0.021	0.054

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

클러스터 1 : 자사고 합격자만 존재하는 그룹

* key : 1
* number : 45

	지역	학교명	유형	과학고	외고_국제고	자사고
46	강동구	동신중학교	사립	0.0	0.0	0.044
103	양천구	신원중학교	공립	0.0	0.0	0.006
118	구로구	개봉중학교	공립	0.0	0.0	0.012
126	영등포구	대림중학교	공립	0.0	0.0	0.050
175	중랑구	혜원여자중학교	사립	0.0	0.0	0.004

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

클러스터 2 : 자사고 합격률이 매우 높으면서 과학고와 외고_국제고 합격자도 일부 존재한다.

* key : 2
* number : 8

	지역	학교명	유형	과학고	외고_국제고	자사고
20	서초구	서초중학교	공립	0.003	0.013	0.085
79	강동구	한영중학교	사립	0.004	0.011	0.077
122	구로구	구일중학교	공립	0.004	0.012	0.079
188	동작구	대방중학교	공립	0.003	0.015	0.076
214	도봉구	도봉중학교	공립	0.004	0.011	0.072

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

클러스터 3 : 과학고 합격자 없이 외고_국제고와 자사고 합격자를 배출한 점은 클러스터 0과 비슷하지만, 외고_국제고 합격률이 클러스터 0에 비해 현저하게 낮다.

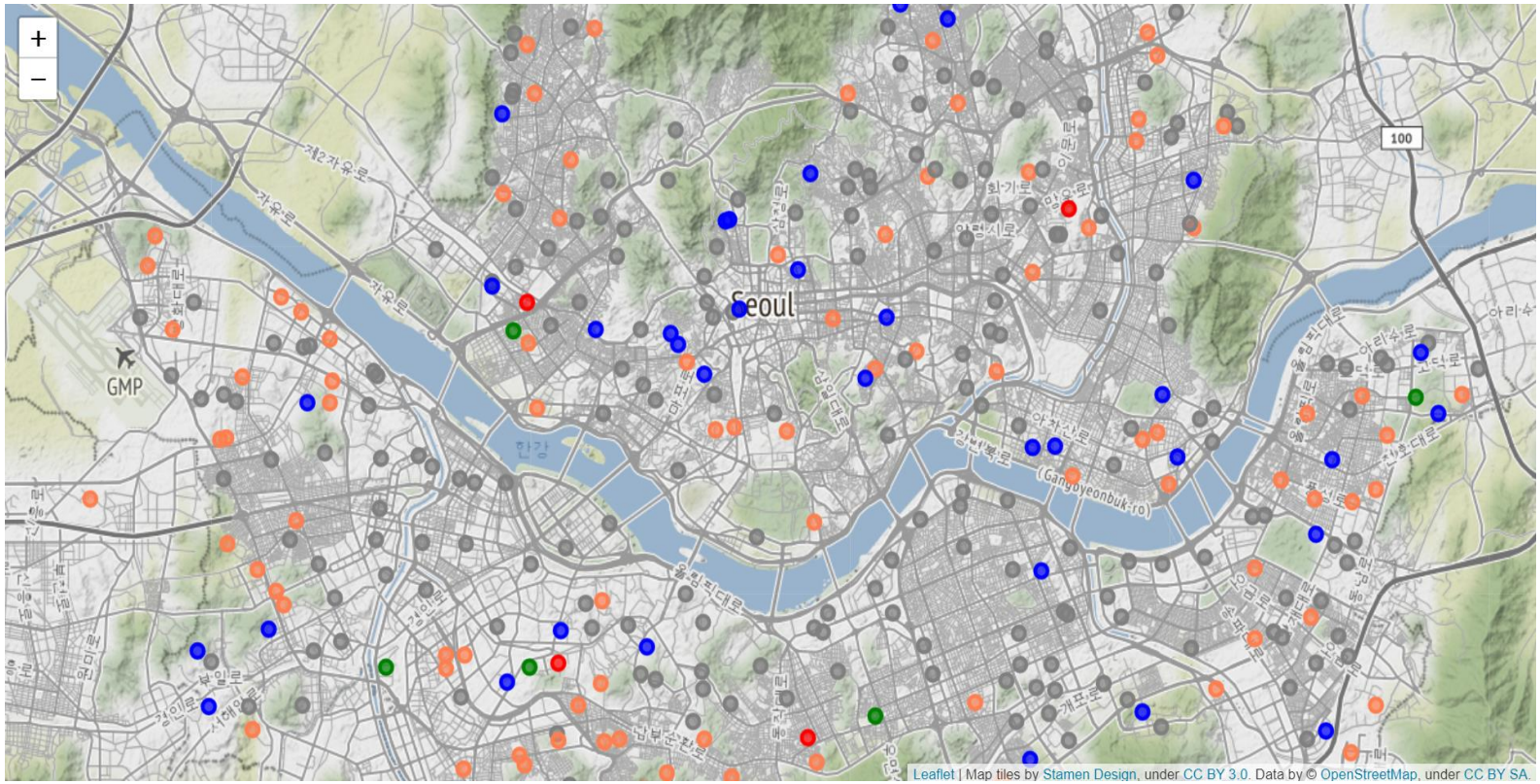
* key : 3
* number : 5

	지역	학교명	유형	과학고	외고_국제고	자사고
35	서초구	이수중학교	공립	0.0	0.004	0.100
177	동대문구	휘경중학교	공립	0.0	0.004	0.094
191	동작구	문창중학교	공립	0.0	0.004	0.084
259	마포구	성사중학교	공립	0.0	0.004	0.078
305	강북구	강북중학교	공립	0.0	0.004	0.088

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석1 결과

군집화된 데이터를 지도 시각화한 파일 seoul_mschool_cluster.html 실행



DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석2

dbscan_clustering.py (8)

```
# 분석2. 과학고, 외고_국제고, 자사고 진학률, 유형(국립,공립,사립)으로 군집
# X2 데이터셋에 대하여 위의 과정을 반복(과학고, 외고_국제고, 자사고 진학률, 유형)
print('분석2. 과학고, 외고_국제고, 자사고 진학률, 유형(국립,공립,사립)으로 군집')
columns_list2 = [10, 11, 14, 23]
x2 = df.iloc[:, columns_list2]
print(x2[:5])
print('₩n')

# 설명 변수 데이터를 정규화
x2 = preprocessing.StandardScaler().fit(x2).transform(x2)

# DBSCAN 모델 객체 생성
# 밀도 계산의 기준이 되는 반지름 R(eps=0.2)과 최소 포인트 개수 M(min_samples=5) 설정
dbm2 = cluster.DBSCAN(eps=0.2, min_samples=5)

# DBSCAN 모델 학습
dbm2.fit(x2)
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석2

dbscan_clustering.py (9)

```
# 예측(군집) 결과를 저장할 열(Cluster2)을 데이터프레임에 추가
df['Cluster2'] = dbm2.labels_                          # Cluster2 열 추가됨

# 클러스터 값으로 그룹화하고, 그룹별로 내용 출력 (첫 5행만 출력)
grouped2_cols = [1, 2, 4] + columns_list2          # 1:지역명, 2:학교명, 4:유형
grouped2 = df.groupby('Cluster2')
for key, group in grouped2:
    print('* key :', key)                           # 클러스터  값: -1, 0 ~ 10
    print('* number :', len(group))                # 각 클러스터 속한 학교수
    print(group.iloc[:, grouped2_cols].head())     # 5개의 데이터 출력
    print('₩n')
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석2

dbscan_clustering.py (10)

```
cluster2_map = folium.Map(location=[37.55,126.98], tiles='Stamen Terrain',
                                    zoom_start=12)

for name, lat, lng, clus in zip(df.학교명, df.위도, df.경도, df.Cluster2):
    folium.CircleMarker([lat, lng],
                        radius=5,                    # 원의 반지름
                        color=colors[clus],          # 원의 둘레 색상
                        fill=True,
                        fill_color=colors[clus],     # 원을 채우는 색
                        fill_opacity=0.7,            # 투명도
                        popup=name
    ).add_to(cluster2_map)

# 지도를 html 파일로 저장하기
cluster2_map.save('seoul_mschool_cluster2.html')
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석2 결과

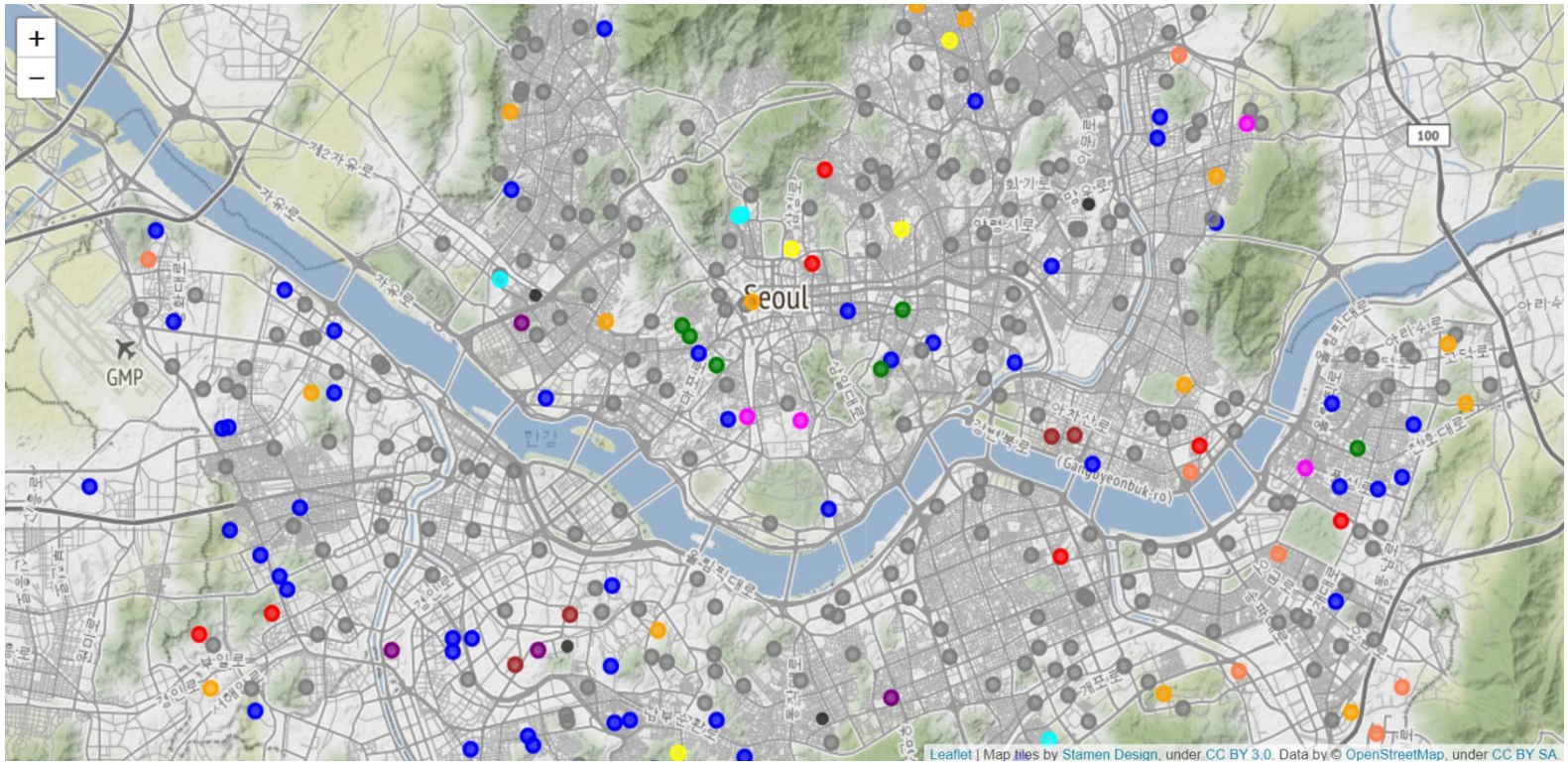
```
print('분석2. 과학고, 외고_국제고, 자사고 진학률, 유형(국립,공립,사립)으로 군집')
columns_list2 = [10, 11, 14, 23]
x2 = df.iloc[:, columns_list2]
print(x2[:5])
```

```
분석2. 과학고, 외고_국제고, 자사고 진학률, 유형(국립,공립,사립)으로 군집
   과학고  외고_국제고  자사고  type
0  0.018      0.007  0.227     1
1  0.000      0.035  0.043     1
2  0.009      0.012  0.090     0
3  0.013      0.013  0.065     0
4  0.007      0.010  0.282     0
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석2 결과

군집화된 데이터를 지도 시각화한 파일 seoul_mschool_cluster2.html 실행



DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석3

dbscan_clustering.py (11)

```
# 분석3. 과학고, 외고_국제고 군집
# x3 데이터셋에 대하여 위의 과정을 반복(과학고, 외고_국제고)
print('분석3. 과학고, 외고_국제고 군집')
columns_list3 = [10, 11]
x3 = df.iloc[:, columns_list3]
print(x3[:5])
print('₩n')

# 설명 변수 데이터를 정규화
x3 = preprocessing.StandardScaler().fit(x3).transform(x3)

# DBSCAN 모델 객체 생성
# 밀도 계산의 기준이 되는 반지름 R(eps=0.2)과 최소 포인트 개수 M(min_samples=5) 설정
dbm3 = cluster.DBSCAN(eps=0.2, min_samples=5)

# DBSCAN 모델 학습
dbm3.fit(x3)
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석2

dbscan_clustering.py (12)

```
# 예측(군집) 결과를 저장할 열(Cluster3)을 데이터프레임에 추가
df['Cluster3'] = dbm3.labels_                          # Cluster3 열 추가됨

# 클러스터 값으로 그룹화하고, 그룹별로 내용 출력 (첫 5행만 출력)
grouped3_cols = [1, 2, 4] + columns_list3              # 1:지역명, 2:학교명, 4:유형
grouped3 = df.groupby('Cluster3')
for key, group in grouped3:
    print('* key :', key)                              # 클러스터  값: -1, 0 ~ 6
    print('* number :', len(group))                    # 각 클러스터 속한 학교수
    print(group.iloc[:, grouped3_cols].head())         # 5개의 데이터 출력
    print('₩n')
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석3

dbscan_clustering.py (13)

```
cluster3_map = folium.Map(location=[37.55,126.98], tiles='Stamen Terrain',
                                    zoom_start=12)

for name, lat, lng, clus in zip(df.학교명, df.위도, df.경도, df.Cluster3):
    folium.CircleMarker([lat, lng],
                        radius=5,                    # 원의 반지름
                        color=colors[clus],          # 원의 둘레 색상
                        fill=True,
                        fill_color=colors[clus],     # 원을 채우는 색
                        fill_opacity=0.7,            # 투명도
                        popup=name
    ).add_to(cluster3_map)

# 지도를 html 파일로 저장하기
cluster3_map.save('seoul_mschool_cluster3.html')
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석3 결과

```
print('분석3. 과학고, 외고_국제고 군집')
columns_list3 = [10, 11]
x3 = df.iloc[:, columns_list3]
print(x3[:5])
```

```
분석3. 과학고, 외고_국제고 군집
   과학고  외고_국제고
0   0.018       0.007
1   0.000       0.035
2   0.009       0.012
3   0.013       0.013
4   0.007       0.010
```

DBSCAN Clustring

- Step4. DBSCAN 군집 모델 학습 및 시각화 : 분석3 결과

군집화된 데이터를 지도 시각화한 파일 seoul_mschool_cluster3.html 실행