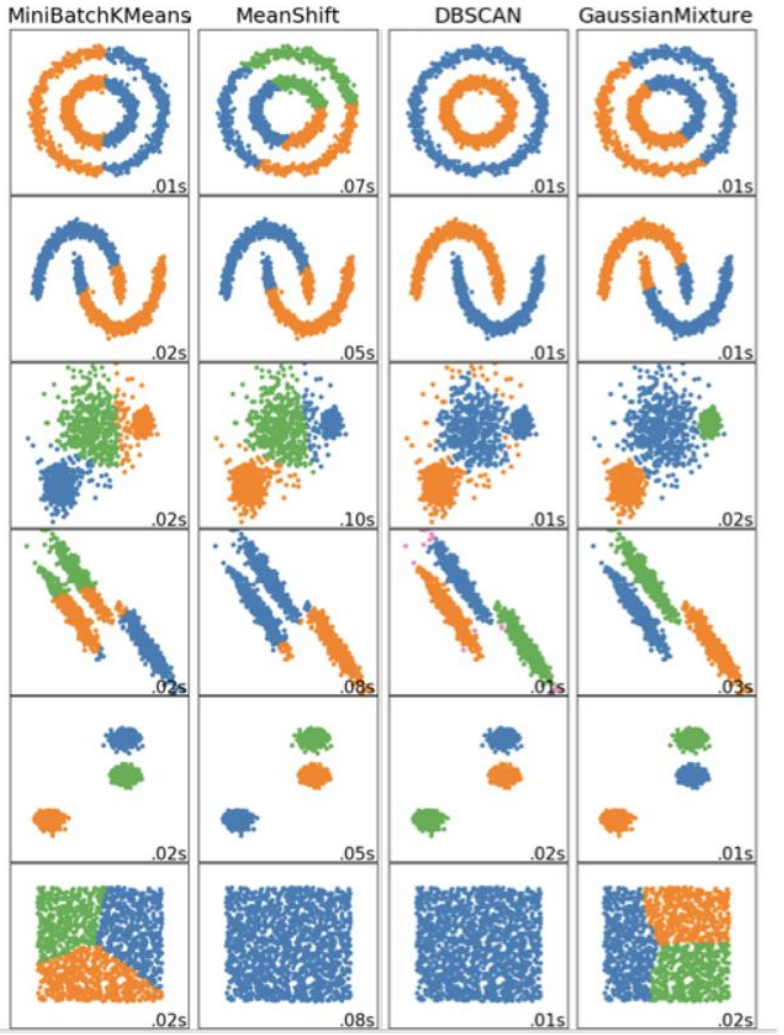
Clustering

Clustering?

- 비지도학습의 대표적인 기술로 레이블이 지정 되어있지 않은 데이터를 그룹핑하는 분석 알고리즘
- 데이터들의 특성을 고려해 데이터 집단(클러스터)을 정의하고 데이터 집단의 대표할 수 있는 중심점을 찾는 것으로 데이터 마이닝의 한 방법이다.
- 추천 엔진 : 사용자 경험을 개인화하기 위해 비슷한 제품 묶어주기
- 검색 엔진: 관련 주제나 검색 결과 묶어주기
- 시장 세분화(segmentation): 지역, 인구 통계, 행동에 따라 비슷한 고객들 묶어주기

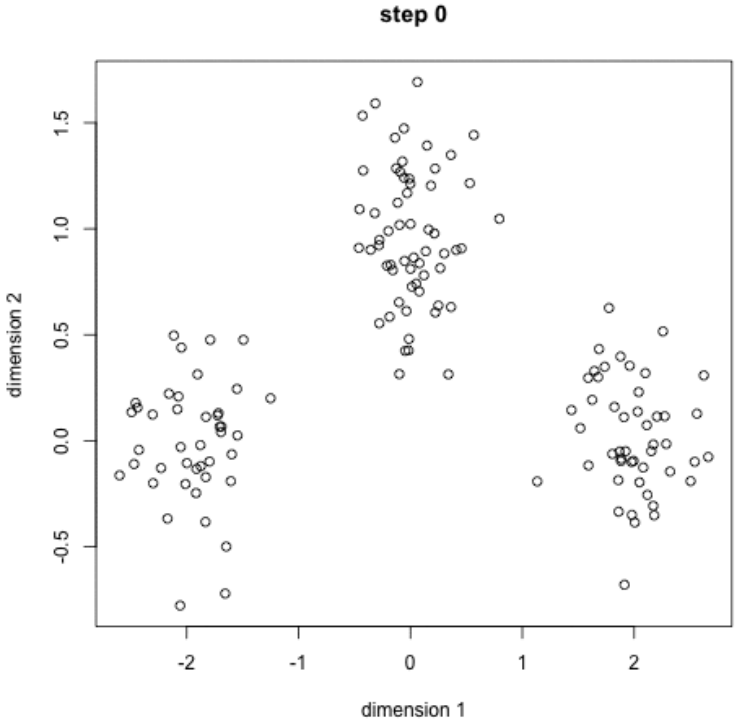
clustering algorithm 종류

- K-Means Clustering
- Mean-Shift Clustering
- DBSCAN
- GMM



K-Means Clustering

- “K“는 데이터 세트에서 찾고자 하는 군집 개수
- “Means“는 각 데이터로부터 그 데이터가 속한 클러스터의 중심까지의 평균 거리를 의미
- (이 값을 최소화하는 게 알고리즘의 목표가 된다.)
- 군집 중심점(centroid)라는 특정한 임의의 포인트들을 배치하고 각 데이터들을 가장 가까운 중심점으로 할당함.
- 군집으로 지정된 데이터들을 기반으로 다시 중심점을 평균 지점으로 이동하는 프로세스를 반복적으로 수행



K-Means Clustering 장단점

- 장점 :
- 이해하기 쉽고 구현하기 쉽기 때문에 가장 잘 알려진 클러스터링 알고리즘임
- 대용량 데이터에도 활용이 가능
- 계산량은 O(n)으로 아주 빠름.

- 단점:
- 클래스/그룹 수(K)를 내가 정해야 한다.
- 반복을 수행하는데 반복횟수가 많을 경우 매우 느려짐
- 거리기반 알고리즘으로 속성의 개수가 매우 많을수록 군집화 정확도가 떨어짐(PCA 차원감소 적용)
- 처음에 point를 랜덤하게 고르기 때문에 돌릴 때마다 다른 결과가 나온다.

K-Means++

알고리즘 :

- 데이터 포인트 중에서 무작위로 균일하게 하나의 중심을 선택
- 나머지 데이터 포인트들에 대해 그 첫번째 중심점까지의 거리를 계산함.
- 두번째 중심점은 각 점들로부터 거리비례 확률에 따라 선택한다. 즉, 이미 지정된 중심점으로부터 최대한 먼 곳에 배치된 데이터포인트를 그 다음 중심점으로 지정한다는 뜻임.
- 중심점이 k개가 될 때까지 반복함.
- 이제 초기 중심이 선택되었으므로 표준 k- means 클러스터링을 사용하여 진행
- => k- means 의 최종 오류에서 상당한 개선
- => 알고리즘의 초기 선택에는 추가 시간이 걸리지만 k- means 부분 자체는 이 시드 후에 매우 빠르게 수렴하므로 알고리즘은 실제로 계산 시간을 줄임

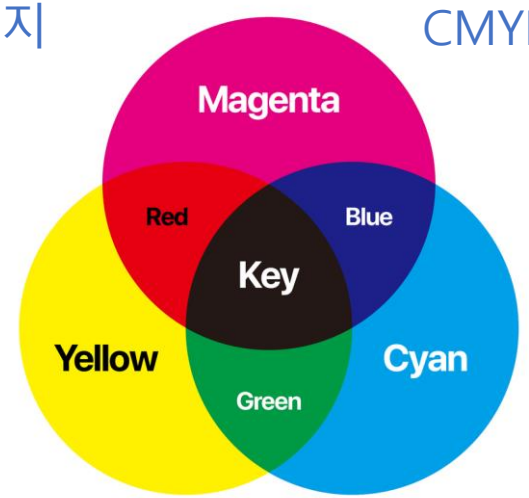
Color Detection

```
# cluster the pixel intensities
k = 5
clt = KMeans(init='k-means++', n_clusters = k)
clt.fit(image)
```

default

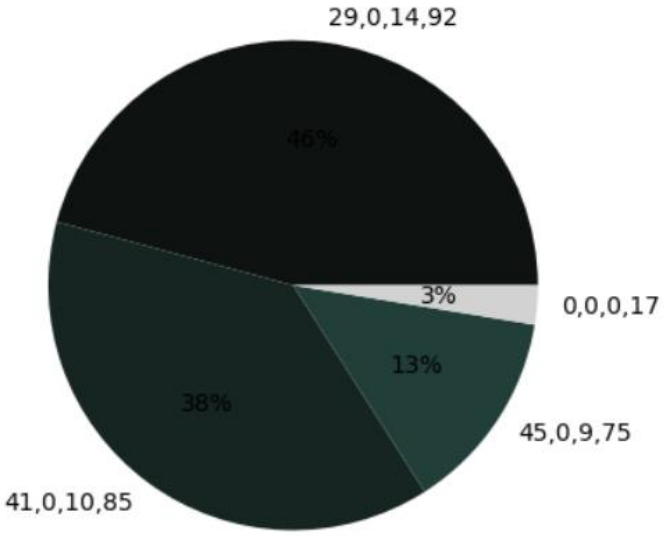
결과 이미지

CMYK



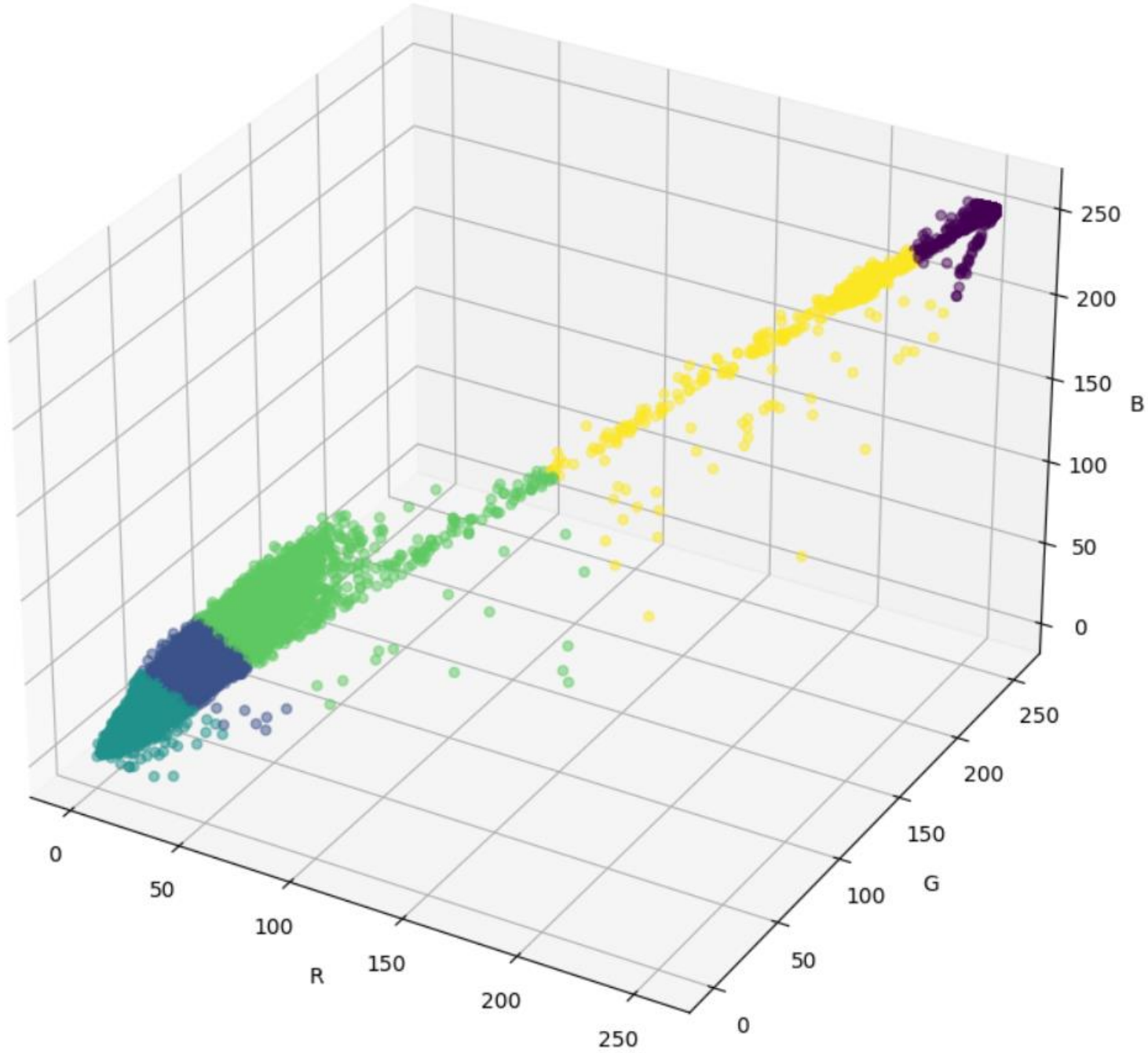
4 Colors Detection

CMYK



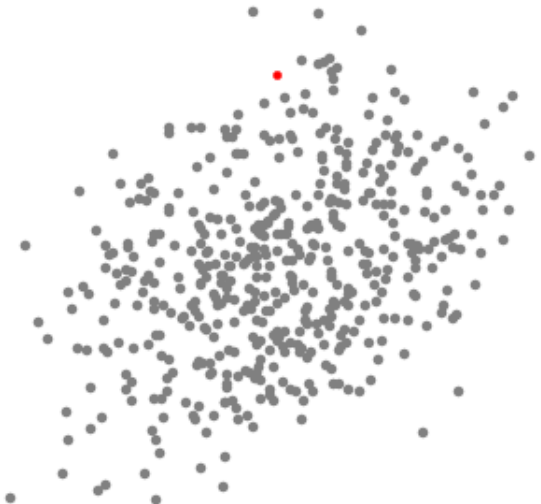

```
background : 31%
4 color detection except for background color /
Main Color : CMYK(29,0,14,92)
Sub Color : CMYK(41,0,10,85) CMYK(45,0,9,75)
```

셔츠에서 차지하는 비율이 10% 미만은 제외시킴


0
50
100
150
200
250
R
0
50
100
150
200
250
G
0
50
100
150
200
250
B

Mean-Shift Clustering
(평균점 이동 클러스터링)

- K-means와 유사하지만 거리 중심이 아니라 데이터가 모여 있는 밀도가 가장 높은 쪽으로 군집 중심점을 이동하면서 군집화를 수행함.
- 정형 데이터 세트보다 이미지나 영상 데이터에서 특정 개체를 구분하거나 움직임을 추적하는데 뛰어난 역할을 수행하는 알고리즘임.
- Sliding window의 size (kernel = 반지름(r))를 지정해줌.
- Cluster의 centroid가 되는 data point를 랜덤하게 설정.
- 정해진 크기의 window 안에서 data의 분포가 높은 방향으로 cluster의 centroid가 이동함. (가장 밀도가 높은 곳을 찾기 위해 KDE함수를 사용함)
- Sliding window가 중첩된 경우, data point가 많은 window를 유지.
- Cluster의 이동/변화가 멈출 때까지 해당 작업 반복함.
- 여러 개의 sliding window 안에 모든 point들이 포함될 때까지 작업이 반복되는 것임.

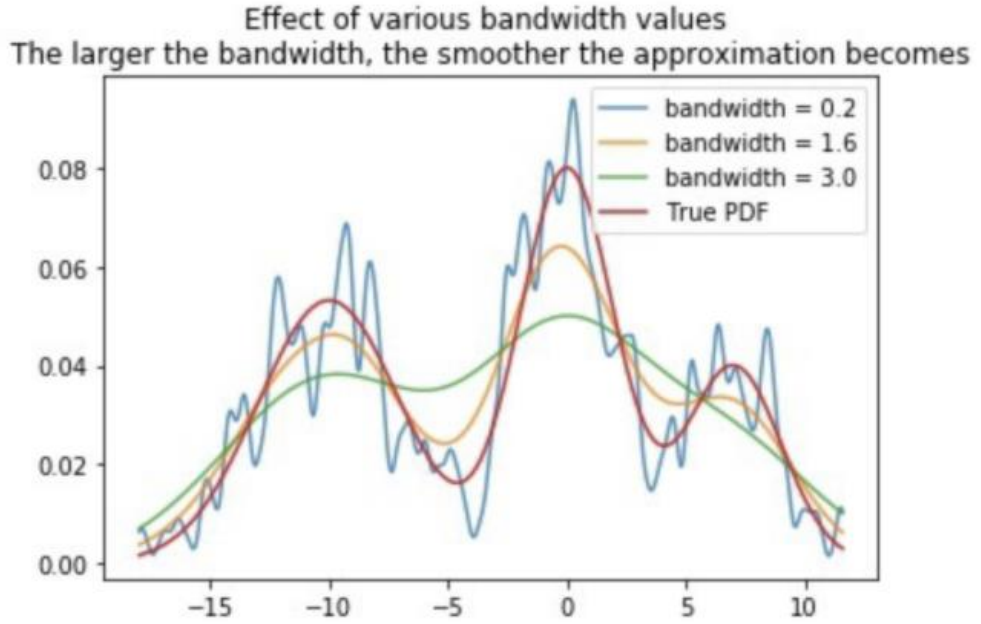


KDE

- 개별 데이터 포인트들에 커널함수를 적용한 후 반환값들을 모두 합한 뒤에 개별 데이터 포인트들의 개수로 나누어서 확률 밀도 함수를 추정하는 방식임.
- 가장 대표적인 커널함수로는 Gaussian 분포 함수(정규분포 함수)가 사용됨.

$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

KDE의 수식



Mean-Shift Clustering 장단점

- 장점 :
- K-Means와 달리 군집 수를 정할 필요가 없음.

- 단점 :
- kernel = 반지름 r 사이즈를 선택해야 한다.
- 알고리즘 수행시간이 오래 걸림
- Bandwidth의 크기에 따른 군집화 영향도가 매우 큼.
- 특이한 형태를 지니는 data를 clustering하기에는 한계가 있음 -> 이를 보완할 수 있는 모델에 DBSCAN 이 있음.

Color Detection

너무 오래걸림->

```
# quantile : 데이터 개수의 일정 비율만큼 샘플링하면서 meanshift하게됨
# 따라서, 데이터 개수가 엄청 많아질시 quantile값이 적으면 시간이 너무 오래걸림
bandwidth = estimate_bandwidth(image, quantile=0.8)
print("최적의 bandwidth 값:", bandwidth)
# cluster the pixel intensities
meanshift = MeanShift(bandwidth=bandwidth)
# clustering 레이블 반환
cluster_labels = meanshift.fit_predict(image)
print('Cluster Label 유형:',np.unique(cluster_labels))
```

결과

```
최적의 bandwidth 값: 397.4652315563376
Cluster Label 유형: [0]
```

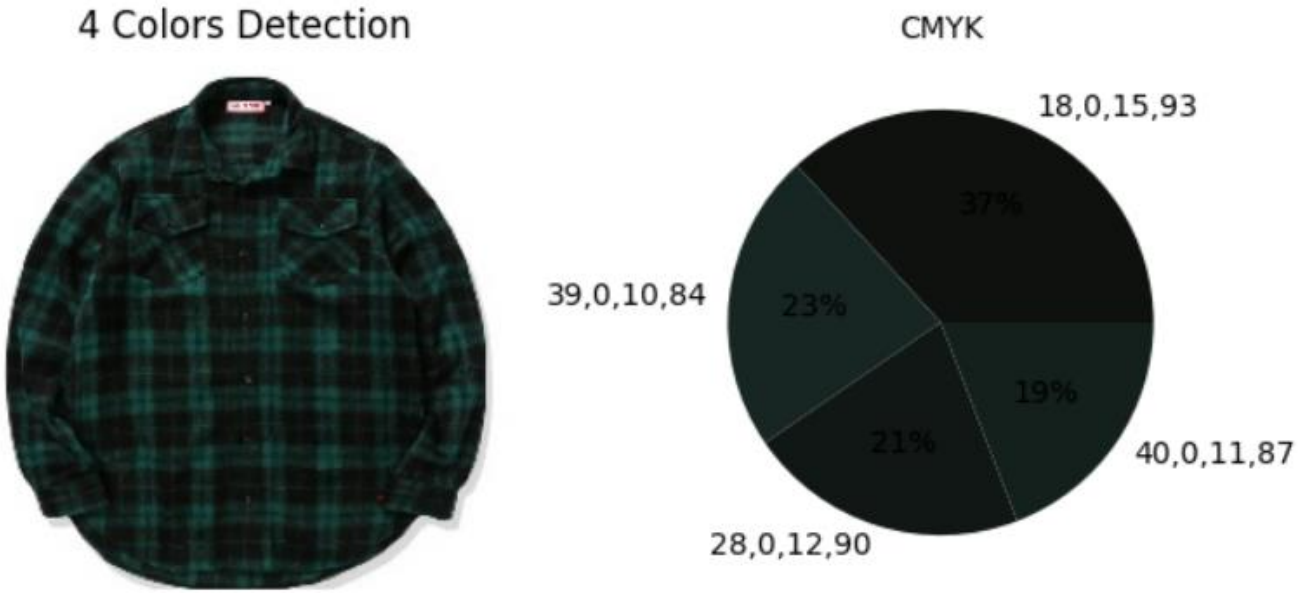
Color Detection

```
# cluster the pixel intensities
meanshift = MeanShift(bandwidth=1.6)
# clustering 레이블 반환
cluster_labels = meanshift.fit_predict(image)
```

결과

Cluster Label 유형 : 0 - 152

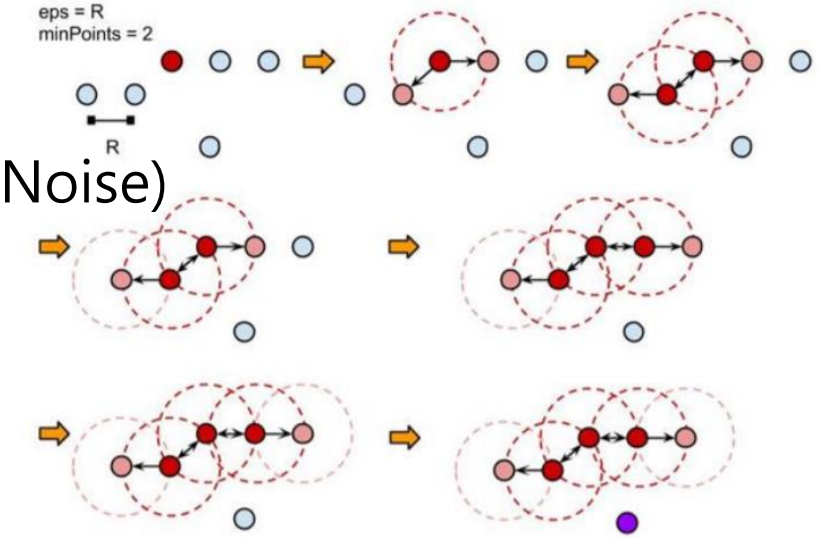
```
background : 31%
4color detection except for background color /
Main Color : CMYK(18,0,15,93)
Sub Color : CMYK(39,0,10,84) CMYK(28,0,12,90) CMYK(40,0,11,87)
```



결과 이미지

DBSCAN

(Density –Based Spatial Clustering of Applications with Noise)



- 밀도 기반 군집화의 대표적인 알고리즘
- 사용자는 eps와 minPoints를 parameter로 지정
- eps를 반지름으로 하는 원 안에 있는 관측치를 찾음
- 찾으면 그 관측치를 군집내 포함시키고, 해당 관측치를 기준으로 다시 관측치를 찾아나섬.
- 이 군집이 최소한의 관측치 개수 minPoints를 넘기면 군집으로 인정이 되는 것이다.
- 그리고 연결되지 않은 관측치들은 이상치로 분류함.

DBSCAN 장단점

- 장점 :
- 가장 큰 장점은 원형이 아닌 데이터들의 군집을 잘 분류해내고, 이상치에 반응을 하지 않는다는 것이다. 그리고 K-means처럼 군집의 수를 설정할 필요가 없음.

- 단점 :
- 알고리즘 수행시간이 오래 걸림
- 하나의 데이터는 한 군집에 속해야 하는데, 이는 시작점에 따라 다른 군집으로 형성될 가능성이 존재함.
- Eps의 크기에 의해 DBSCAN의 성능이 크게 좌우됨. -> Eps와 MinPoints parameter를 잘 설정할 필요가 있음.
- Eps설정은 일반적으로 K-nearest neighbor 그래프의 distances를 그래프로 나타낸 후 거리가 급격하게 증가하는 지점을 eps로 설정하는 방법이 있음.

Color Detection

```
dbscan = DBSCAN(eps=1.5, min_samples=5) ##
dbscan_labels = dbscan.fit_predict(image)
```

결과

```
Estimated number of clusters: 504
Estimated number of noise points: 2026
```

```
background : 31%
4 color detection except for background color /
Main Color : CMYK(33,0,14,89)
Sub Color : CMYK(5,0,15,93) CMYK(34,0,4,88) CMYK(17,0,4,64)
```

결과 이미지

HDBSCAN
(Hierarchical DBSCAN)

- DBSCAN의 upgrade version임.
- DBSCAN과 계층적(Hierarchical) 군집분석 개념을 통합한 기법임. -> DBSCAN은 local density에 대한 정보를 반영해줄 수 없고, 또한 데이터들의 계층적 구조를 반영한 clustering이 불가능하다. 이를 개선한 알고리즘이 HDBSCAN임.

- ~~dataset의 값들을 density/sparsity 의미를 포함한 값으로 변환함~~
- ~~point간의 mutual reachability distance값을 기반으로 MST(minimum spanning tree)를 만듬~~
- ~~connected component들의 클러스터 hierarchy를 구축함~~
- ~~설정된 MinPts값을 기반으로 클러스터 hierarchy를 축약함~~
- ~~축약된 클러스터 hierarchy에서 stable한 클러스터만 선택함~~

HDBSCAN 장단점

- 장점 :
- DBSCAN의 eps를 설정할 필요가 없어짐 -> DBSCAN보다 하이퍼 파라미터(eps, min_samples)에 덜 민감함.

- 단점 :
- 항상 DBSCAN보다 좋은 결과를 보장하는 것은 아님.

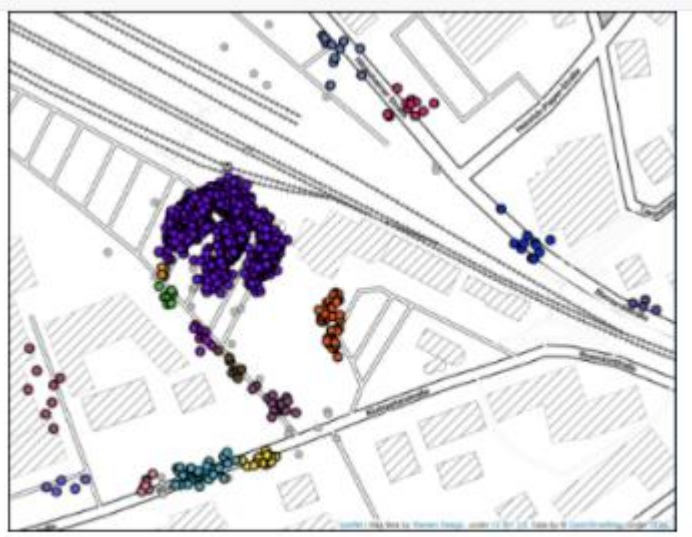
DBSCAN 결과 역 주변 밀집된 인구 클러스터를 잘 나타낸다.

HDBSCAN 결과 너무 세분화된 인구 클러스터가 도출되었다.

HDBSCAN & DBSCAN

- 주어진 cluster_selection_epsilon 거리를 활용하여 DBSCAN 수행 후 , 나머지 데이터에 대해서 HDBSCAN을 수행함.
- 이렇게 하면, 원하는 만큼의 밀집도를 가지는 데이터는 하나의 클러스터에 묶이는 것을 보장할 수 있음.

*cluster_selection_epsilon : DBSCAN 알고리즘을 적용할 최대 거리 명시



cluster_selection_epsilon 파라미터를 사용한 HDBSCAN 결과

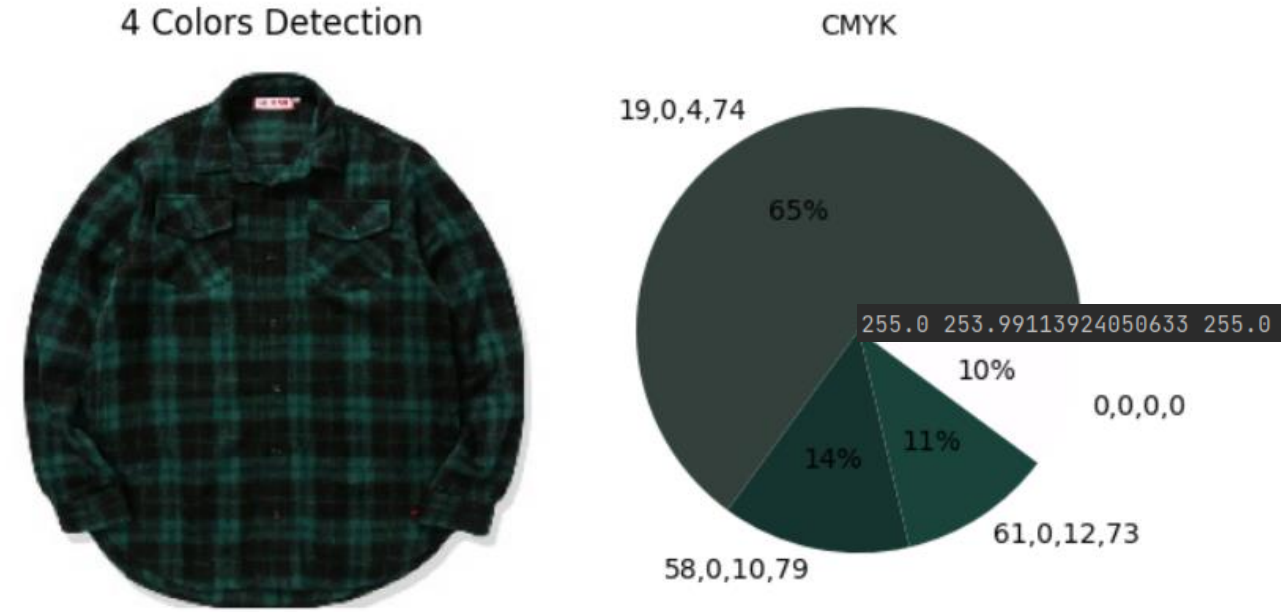
Color Detection

```
# hdbscan
epsilon = 0.5
clusterer = hdbscan.HDBSCAN(min_cluster_size=60, min_samples=5, cluster_selection_epsilon=epsilon)
dbscan_labels = clusterer.fit_predict(image)
```

결과

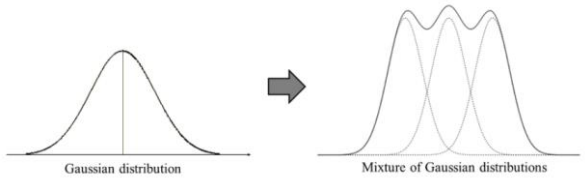
```
Estimated number of clusters: 253
Estimated number of noise points: 5059
```

```
background : 25%
4 color detection except for background color /
Main Color : CMYK(19,0,4,74)
Sub Color : CMYK(58,0,10,79) CMYK(61,0,12,73) CMYK(0,0,0,0)
```



결과 이미지

GMM(Gaussian Mixture Model)



- 전체 데이터 세트는 서로 다른 정규분포를 가진 여러가지 확률 분포 곡선으로 구성되어 있다고 가정하며, 정규분포에 기반에 군집화를 수행하는 것이 GMM 군집화 방식
- 일정한 데이터 세트가 있으면 이를 구성하는 여러개의 정규 분포 곡선 추출 뒤, 개별 데이터가 이 중 어떤 정규분포에 속하는지 결정
- 이러한 방식을 모수추정이라고 하며 "개별 정규 분포의 평균과 분산", "각 데이터가 어떤 정규 분포에 해당되는지의 확률"을 구하기 위해서 추정함.
- GMM은 모수추정을 하기 위해 EM(Expectation and Maximization)방법을 적용하고 있음.

EM(Expectation - Maximization)

- E-step / M-step 2단계로 나뉘어 지며, 2단계의 Step을 반복하면서 최적의 파라미터 값을 찾아감
- E-step : 주어진 임의의 파라미터 초기값에서 likelihood 값을 계산함
- M step : E-step에서 계산된 likelihood를 최대화하는 새로운 값을 얻음

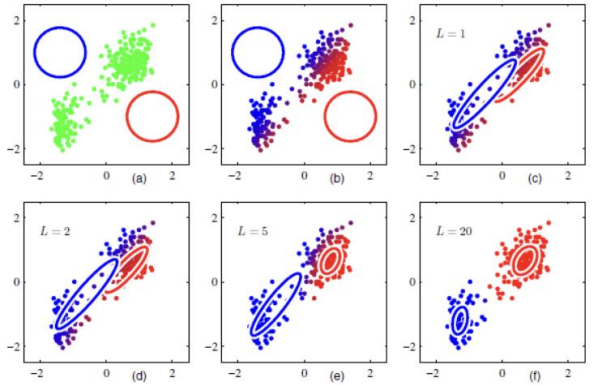
가능도 = L(확률분포D | 관측값X)

*likelihood(가능도) : 어떤 값이 관측되었을 때, 이것이 어떤 확률 분포에서 왔을지에 대한 확률

GMM을 이용한 clustering

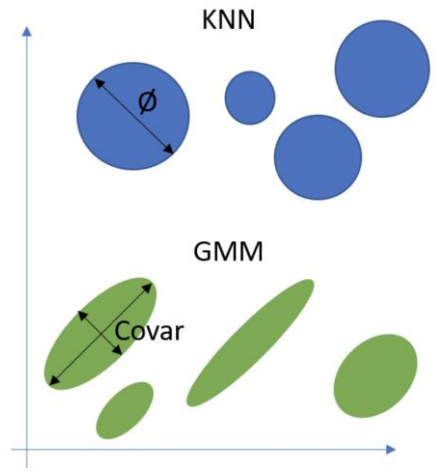
Expectation-Maximization (EM) Clustering using Gaussian Mixture Models (GMM)

- 앞선 EM알고리즘에서 구했던 각각의 가우시안 분포의 평균값이 cluster center가 되게 하면 clustering을 할 수 있음.
- 그림 K=2일때, EM알고리즘 적용을 나타낸다. 여기서 동그라미 두개는 두 가우시안의 단위표준편차 경로이고,
- 초록색 점들이 빨강,파랑으로 점점 칠해지는것은 확률 값을 기준으로 칠한 것이다.
- b)는 Estep을 나타내고 (확률(책임값) 계산)
- c)는 Mstep을 나타낸다. (평균이 옮겨감)
- 이를 20사이클 반복한 것이 f)이다. 여기서 알고리즘이 거의 수렴한 것을 살펴볼 수 있다.



GMM 장단점

- 장점 :
- K-means clusterin의 중요한 특징으로 hard clustering method(각 포인트를 하나의 클러스터에만 연결함)이 있음. 이 접근법은 데이터포인트가 특정 클러스터와 얼마나 관련이 있는지 알려주는 불확실성 혹은 확률이 없다는 것이 한계임-> 이런 한계를 해결하기 위해 GMM을 사용해볼 수 있음.
- 주로 타원형으로 길게 늘어진 데이터 분포에 적용이 용이하다.



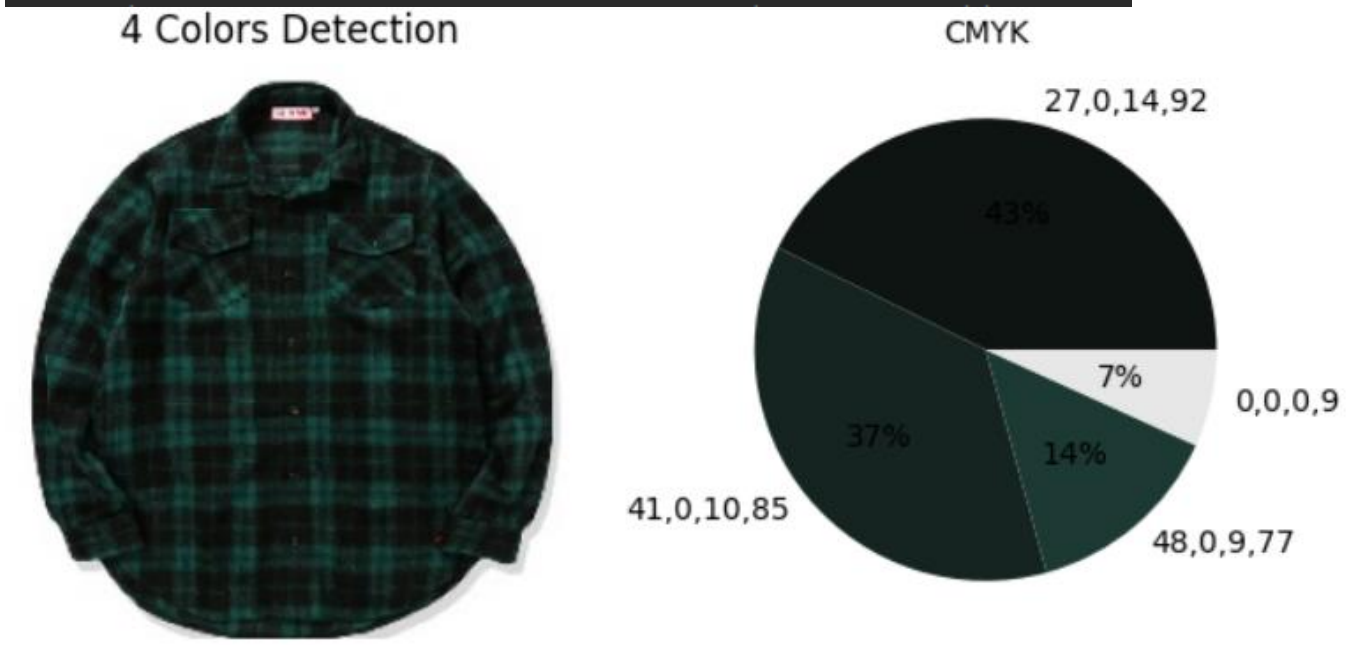
- 단점:
- 임의의 "K"값을 설정해야함.
- 계산이 복잡하고 느림.

Color Detection

```
# cluster the pixel intensities
k = 5
# n_components로 미리 군집 개수 설정
gmm = GaussianMixture(n_components=k, random_state=42)
gmm_labels = gmm.fit_predict(image)
```

결과

```
background : 28%
4 color detection except for background color /
Main Color : CMYK(27,0,14,92)
Sub Color : CMYK(41,0,10,85) CMYK(48,0,9,77)
```



결과 이미지

Run time

- 각각 모델이 clustering하는데 걸린 시간
- 실행시마다 약간 다르게 나옴 그래도 아래 순서대로임
- K-means < GMM << DBSCAN ≒ HDBSCAN&DBSCAN <<<<<< Mean-Shift

모델명	K-means	Mean-Shift	DBSCAN	HDBCAN& DBSCAN	GMM
걸린 시간(s)	0.33	76.17	4.28	4.53	0.44

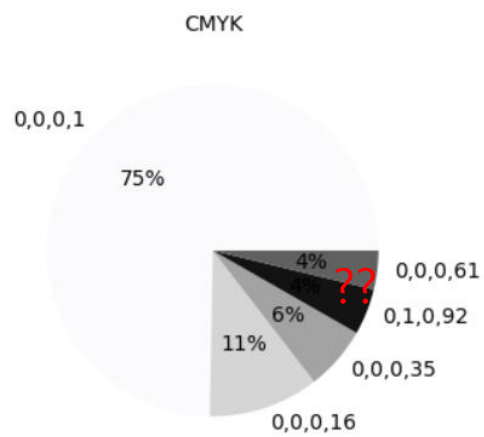
제 노트북 파이참으로 한번 측정해본 결과들입니다.

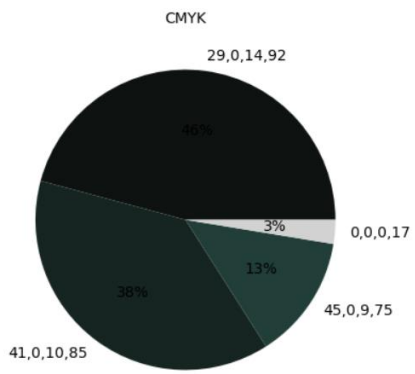
Color Detection with k-means

5가지 클래스의 패턴 이미지

- K = 5

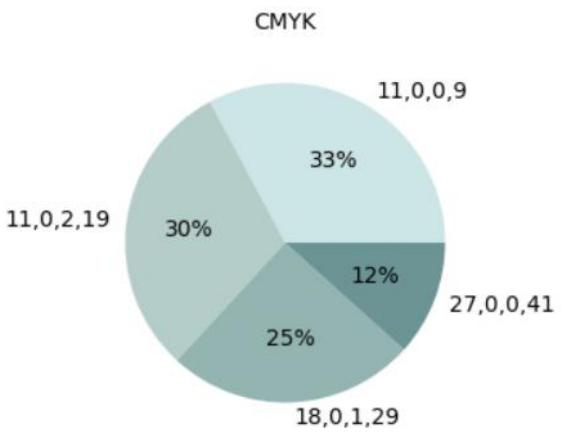
dot





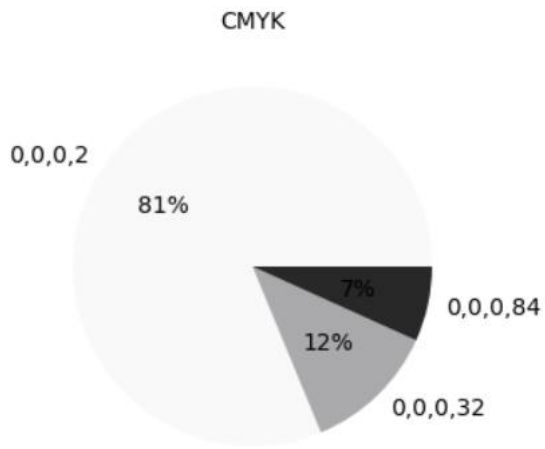
check

floral



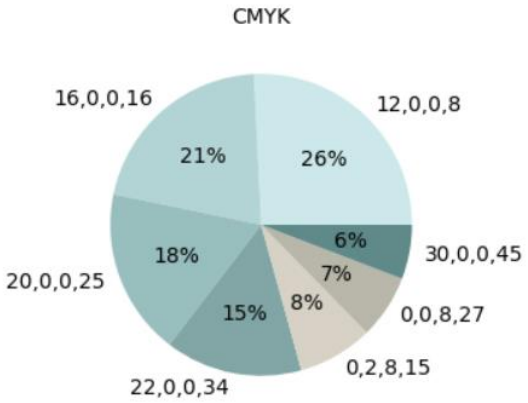
K = 3


3 Colors Detection


CMYK
0,0,0,2
81%
7%
0,0,0,84
12%
0,0,0,32

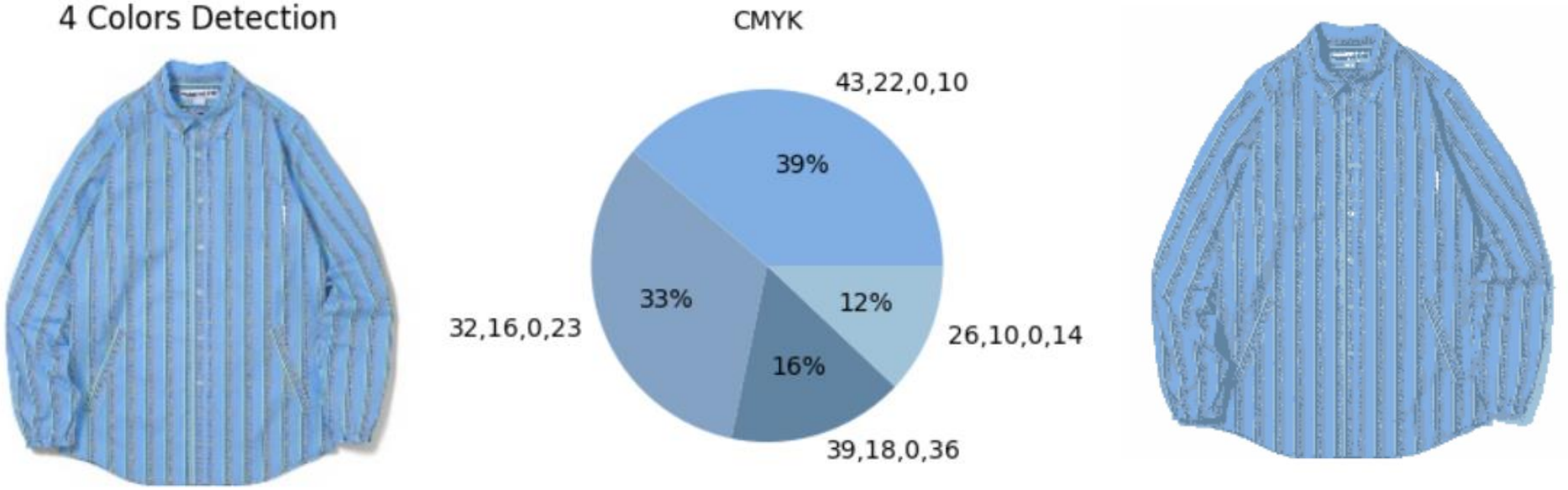
K = 8


7 Colors Detection

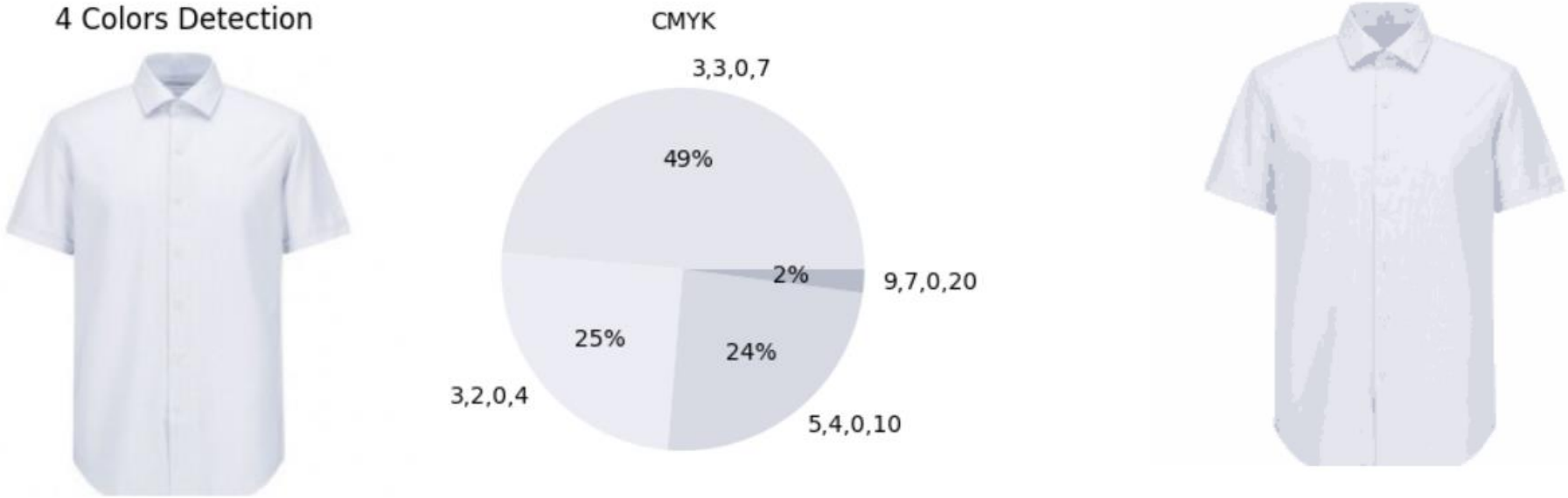

CMYK
16,0,0,16
21%
12,0,0,8
26%
6%
30,0,0,45
18%
20,0,0,25
7%
8%
0,0,8,27
15%
0,2,8,15
22,0,0,34

- K = 5

stripe



solid



Result & Opinion

- 각 모델 별로 장단점이 있고, 같은 데이터셋도 사용하는 모델에 따라 clustering 결과가 다양하게 나타나므로 data의 형태 / 분석 방향에 알맞은 모델을 기반으로 clustering 하는 것이 중요합니다.

- 군집화 모델을 결정해서 알맞은 파라메타 값을 찾는 연구가 더 필요할 것 같습니다. ~~그리고 이미지의 클래스별로 그 연구가 필요할 듯 싶습니다.~~

참고

- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster
- https://brunch.co.kr/@mnc/10
- https://techblog-history-younghunjo1.tistory.com/108
- https://heave.tistory.com/60
- https://hdbscan.readthedocs.io/en/latest/parameter_selection.html
- 등등,,,