Homework #2

2017012251 윤영훈

선행과정

K-means, DBSCAN 알고리즘을 구현하기 전 동일한 환경을 구축하기 위해 양쪽 코드 모두 필수 library import 및 Data set 구축

In [1]:
```
%matplotlib inline
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

◀ pandas, matplotlib import

In [3]:
```
df = pd.read_csv('cluster2.csv')
print("Dimensions of the data = {}".format(df.shape))
```

```
Dimensions of the data = (1300, 2)
```

◀ data set read

In [4]:
```
df[:5]
```

Out [4]:

	X	Y
0	1.070487	1.328147
1	1.072777	1.191249
2	0.328029	1.261713
3	0.600926	1.254465
4	0.759281	1.284541

In [5]:
```
X = df.values
X[:5]
```

◀ data set values를 array X에 저장

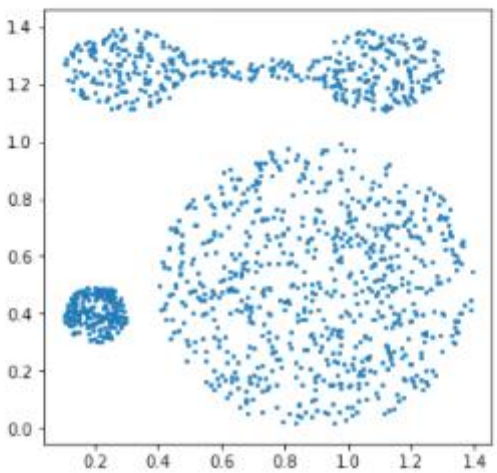
Out [5]:
```
array([[1.07048688, 1.3281469 ],
       [1.07277723, 1.19124898],
       [0.3280287 , 1.26171275],
       [0.60092578, 1.2544653 ],
       [0.75928098, 1.28454059]])
```

In [6]:
```
plt.figure(figsize=(5, 5))

plt.scatter(X[:, 0], X[:, 1], s=4)

plt.show()
```

◀ plot 차트를 통해 data set의 분포 확인



K-means

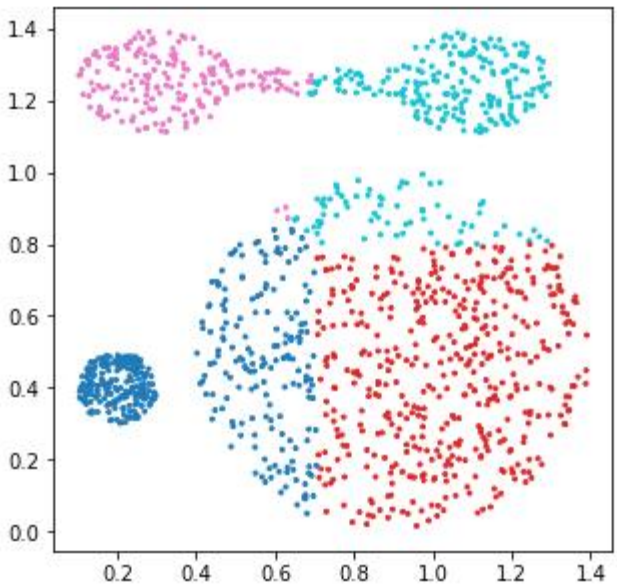
K-Means Clustering

In [8]:
```
from sklearn.cluster import KMeans
k_means = KMeans(n_clusters=4, random_state=0)
```

In [9]:
```
y_pred = k_means.fit_predict(X)
print(y_pred[:10])
```

```
[3 3 2 2 3 2 2 3 2 3]
```

In [10]:
```
plt.figure(figsize=(5, 5))
plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=4, cmap="tab10")
plt.show()
```



◀ scikit-learn에서 기본적으로 구현된 KMeans 알고리즘을 사용.
이때 k = n_clusters = 4 로 assign

◀ 선행과정에서 선언한 data array X를 대입하여 clustering result assign

◀ plot 차트를 통해 data set 분포 확인 및 clustering result 확인.
k = 4에 따라 4개의 cluster로 구분됨.

- **K-means clustering 특징**
  - cluster의 수 (k) 를 직접 설정해줘야 함
  - K개의 cluster의 중심 좌표를 고려하여 모든 data point를 가까운 거리의 cluster로 assign
  - outliers에 민감, 하나의 outlier도 전체 평균치에 유의미한 영향을 끼침
  - 중심 좌표와의 거리에 따라 cluster가 assign되므로 위의 plot graph에서 우측 하단 가장 큰 원이 하나의 cluster로 인식되지 못하고 분리되는 것과 같은 결과가 나올 수 있음

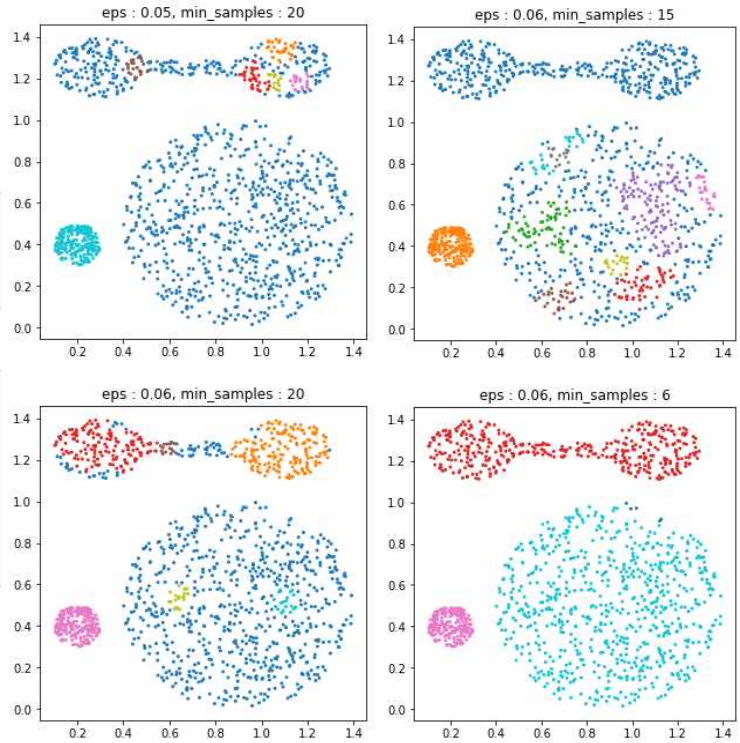
DBSCAN

DBSCAN

In [7]:
```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.05, min_samples=20)
y_pred = dbscan.fit_predict(X)
print(y_pred[:10])
```

```
[ 0  4 -1 -1 -1 -1 -1 -1 -1 -1]
```

In [8]:
```
for i, (eps, min_samples) in enumerate(
    ((0.05, 20), (0.06, 20), (0.06, 15), (0.06, 6))):
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    y_pred = dbscan.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1], cmap="tab10", s=4, c=y_pred)
    plt.title("eps : {}, min_samples : {}".format(eps, min_samples))
plt.show()
```



core point / border point 여부를 검증하는
(eps, min_samples) 값을 알맞게 설정하여
(0.05, 20) / (0.06, 20) / (0.06, 15) / (0.06, 6)
총 4가지의 case 비교

- **DBSCAN clustering 특징**
  - 임의의 점 p를 선정하고 해당 점 p로부터 일정 거리 (eps) 이내에 일정 개수 (min_samples)의 reachable point가 있다면 해당 p를 core point로 선정, 이후 주변의 reachable point에게도 동일한 알고리즘 적용하여 core point 여부 판명
  - 임의의 점 주변에 min_samples개의 reachable point가 존재하지 않는다면 해당 점은 border point로 선정, 더 이상 주변 reachable point 에게 확장되지 않고 clustering을 마침
  - outlier는 border point이고 다른 cluster로 분리되므로 k-means clustering과 달리 다른 cluster의 평균값에 영향을 미치지 않음
  - 단, (eps, min_samples)의 값을 목적에 알맞게 설정하지 않는다면 아래와 같은 문제가 발생할 수 있음