4th Week: Clustering

In [ ]:

```
# importing tools
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
```

1. Clustering: K-means

In [ ]:

```
# make-blobs in sklearn: Generate isotropic Gaussian blobs for clustering
# n_samples (default: 100), n_features (default: 2)
X, y = make_blobs(n_samples=120, random_state=20200920, centers=3)
print(X.shape, y.shape, type(X))
print(y[:10])
print(X[:10])
rgb = np.array(['r','g','b','y'])
plt.scatter(X[:,0],X[:,1], color=rgb[y])
plt.show()
```

In [ ]:

```
kmeans = KMeans(n_clusters = 3).fit(X)   # creating a clustering model and fitting
print(kmeans.cluster_centers_)
print(kmeans.labels_)
print(kmeans.predict([[4,-2], [0,6], [-9., 0]]))
print(kmeans.inertia_)                   # Sum of squared distances of samples to their closest
 cluster center.

wrong = [(p, e) for (p, e) in zip(kmeans.labels_, y) if p != e]
print(wrong)
```

In [ ]:

```
# Choosing the optimal number of clusters: elbow method
cost =[]
for i in range(1, 11):
    KM = KMeans(n_clusters = i, max_iter = 500)
    KM.fit(X)
    cost.append(KM.inertia_)

# plot the cost against K values
plt.plot(range(1, 11), cost, color ='g', linewidth ='3')
plt.xlabel("Value of K")
plt.ylabel("Sqaured Error (Cost)")
plt.show() # clear the plot
```

In [ ]:

```
# bmi data
bmi = pd.read_csv('data/bmi.csv')
print(bmi.shape)
bmi.head()
```

In [ ]:

```
set(bmi['label'])
```

In [ ]:

```
X_bmi = bmi[['height', 'weight']]
y_bmi = bmi['label']

bmi_KM = KMeans(n_clusters = 3).fit(X_bmi)
print(bmi_KM.cluster_centers_)
labels = list(bmi_KM.labels_)
print(labels.count(0), labels.count(1), labels.count(2))
print(list(y_bmi).count('fat'), list(y_bmi).count('normal'), list(y_bmi).count('thin'))
```

In [ ]:

```
rgb1 = np.array(['r','g','b'])
Xbmi = np.array(X_bmi)
ybmi_idx, ybmi_value = pd.factorize(y_bmi)
print(Xbmi.shape, ybmi_idx.shape, ybmi_value.shape)
print(Xbmi[:10])
print(ybmi_idx[:20])
print(list(y_bmi[:20]), end = ' ')
plt.figure(figsize=(8,6))
plt.scatter(Xbmi[:,0],Xbmi[:,1], color=rgb[ybmi_idx])
plt.show()
```

1. Clustering: DBSCAN

In [ ]:

```
from sklearn.datasets import make_moons
import mglearn                                  # install mglearn

Xmoon, ymoon = make_moons(n_samples = 200, noise = 0.05, random_state = 0)
print(Xmoon.shape, ymoon.shape, Xmoon[:3], ymoon[:5])

y_pred = KMeans(n_clusters = 3).fit_predict(Xmoon)

mglearn.discrete_scatter(Xmoon[:,0], Xmoon[:,1], y_pred)
plt.legend(['cluster 0','cluster 1','cluster2'], loc='best')
plt.xlabel('feature 0')
plt.ylabel('feature 1')
plt.show()
```

In [ ]:

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN()
dbscan.eps = 0.2
clusters = dbscan.fit_predict(Xmoon)

mglearn.discrete_scatter(Xmoon[:,0], Xmoon[:,1], clusters)
plt.legend(['cluster 0','cluster 1','cluster2'], loc='best')
plt.xlabel('feature 0')
plt.ylabel('feature 1')
plt.show()
```

In [ ]:

```

```