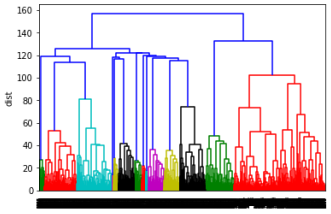
고객의 그룹화

계층형 클러스터링

계층형 클러스터링은 거리가 가까운 데이터부터 순서대로 병합해 그룹을 만듭니다. 그 결과는 덴드로그램(Dendrogram)으로 표시합니다.

```
# Linkage, Dendrogram의 로드
from scipy.cluster.hierarchy import linkage, dendrogram
# Matplotlib의 로드
import matplotlib.pyplot as plt
%matplotlib inline

# 계층적 클러스터링의 실행
hcls = linkage(bank_df_sc, metric='euclidean', method='ward')
dendrogram(hcls)

# 덴드로그램의 작성
plt.ylabel('dist')
plt.show()
```



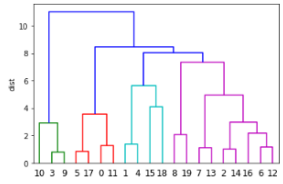
고객간 거리는 유클리드법 즉, 벡터간 거리 계산 방식을 사용하여 산출하고, 고객간 병합은 ward 법을 사용합니다.
그리고 고객별로 그룹을 확인합니다.

```
# fcluster의 로드
from scipy.cluster.hierarchy import fcluster

# 고객별로 클러스터ID를 부여
cst_group = fcluster(hcls, 100, criterion='distance')

# 고객별 클러스터 ID를 확인
print(cst_group)
```

```
[14 14 14 ... 12 12 12]
```

그런데 고객수가 너무 많아 감이 잘 안오죠? 고객수를 **20**개로 제한하여 덴드로그램을 다시 그려보세요.

```
hcls0to19 = linkage(bank_df_sc[:20], metric='euclidean', method='ward')
dendrogram(hcls0to19)
plt.ylabel('dist')
plt.show()
```




```
cst_group0to19 = fcluster(hcls0to19, 10, criterion='distance')
print(cst_group0to19)
```

```
[2 2 2 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2]
```

이제 좀 쉽게 파악이 되나요?

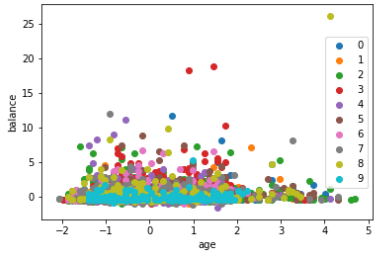
비계층형 클러스터링

비계층형 클러스터링 중 유명한 **k** 평균법**(k-Means** 법**)**을 사용하여 그룹화해보겠습니다. **k** 평균법에서는 전체 데이터를 성질이 비슷한 **k**개의 덩어리로 분할해 그룹을 만듭니다.

```
# KMeans의 로드
from sklearn.cluster import KMeans

# 고객을 10개그룹으로 분할
kcls = KMeans(n_clusters=10)
cst_group = kcls.fit_predict(bank_df_sc)

# 고객별 클러스터ID를 확인
print(cst_group)
```

```
[4 4 4 ... 3 3 3]
```

```
# 고객의 클러스터 ID마다 색을 부여하여 산포도를 작성
for i in range(10):
    tmp = bank_df_sc[cst_group == i]
    plt.scatter(tmp['age'], tmp['balance'], label=i)

plt.legend()
plt.xlabel('age')
plt.ylabel('balance')
plt.show()
```



```
# 데이터 세트에 클러스터 ID를 추가
bank_df_sc['group'] = cst_group

bank_df_sc.head()
```

	age	default	balance	housing	loan	day	duration	campaign
0	1.647908	-0.137148	0.250618	0.876711	-0.444540	-1.275382	-0.006613	-0.574193

```
# 그룹별 데이터 건수를 확인
print(bank_df_sc['group'].value_counts())
```

```
4    1238
7    1096
2    1081
5     916
6     752
3     644
1     464
8     429
9     189
0     124
Name: group, dtype: int64
```

```
# 클러스터0의 통계량을 확인
bank_df_sc[bank_df_sc['group']==0].describe()
```

	age	default	balance	housing	loan	day	duration	campaign	pdays
count	124.000000	1.240000e+02	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000
mean	0.678904	-1.371485e-01	0.344301	-0.750174	-0.205552	0.164835	0.283332	-0.387322	0.518089
std	1.615764	2.786817e-17	1.549163	0.800241	0.769093	0.991199	1.501713	0.271858	1.325311
min	-1.964393	-1.371485e-01	-0.448827	-1.140627	-0.444540	-1.758049	-0.960790	-0.574193	-0.418664
25%	-0.633545	-1.371485e-01	-0.385590	-1.140627	-0.444540	-0.430714	-0.495815	-0.574193	-0.418664
50%	0.222000	-1.371485e-01	-0.096167	-1.140627	-0.444540	0.051953	-0.185521	-0.574193	-0.418664
75%	2.028150	-1.371485e-01	0.412016	-1.140627	-0.444540	0.926788	0.408976	-0.238367	1.409163
max	4.024421	-1.371485e-01	11.750828	0.876711	2.249514	1.741289	11.260878	0.433285	7.706127

8 rows × 32 columns

주성분 분석(PCA, Principle Component Analysis)

PCA에서는 데이터의 산란(분산) 정도에 주목해 분산이 최대가 되는 방향으로 새로운 축(제 1 주성분)을 설정합니다. 그리고 분산이 두번째로 커지는 방향으로 제1주성분에 직교하는 제2주성분을 설정합니다. 이렇게 계속하여 주성분을 설정하여 누적기여율이 70%~80%에 도달할 때까지 주성분을 채워나갑니다. 기여율은 각 주성분이 가지고 있는 정보가 데이터에 의해 어느정도 영향을 미치는지 나타내는 지수입니다.

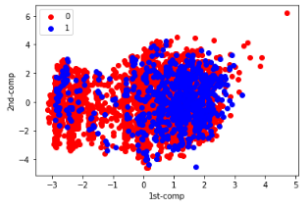
```
# PCA의 로드
from sklearn.decomposition import PCA

# 주성분을 누적기여율 80% 까지 추출
pca = PCA(0.80)
bank_df_pca = pca.fit_transform(bank_df_sc)

# 주성분을 확인
print(pca.n_components_)

# 차원압축한 데이터 세트의 사이즈 확인
print(bank_df_pca.shape)
```

```
18
(6933, 18)
```

```
# 파일에서 목적변수만 리드
y = pd.read_csv('bank-prep.csv', sep=',')['y']

# 데이터 세트에 y를 추가
bank_df_pca = pd.DataFrame(bank_df_pca)
bank_df_pca['y'] = y

# 데이터 세트의 사이즈를 확인
print(bank_df_pca.shape)
```

```
(6933, 19)
```

```
import matplotlib.pyplot as plt
%matplotlib inline

# y=0의 제1주성분과 제2주성분을 작성
bank_df_pca_0 = bank_df_pca[bank_df_pca['y'] == 0]
bank_df_pca_0 = bank_df_pca_0.drop('y', axis=1)
plt.scatter(bank_df_pca_0[0], bank_df_pca_0[1], c='red', label=0)

# y=1의 제1주성분과 제2성분을 작성
bank_df_pca_1 = bank_df_pca[bank_df_pca['y'] == 1]
bank_df_pca_1 = bank_df_pca_1.drop('y', axis=1)
plt.scatter(bank_df_pca_1[0], bank_df_pca_1[1], c='blue', label=1)

plt.legend()
plt.xlabel('1st-comp')
plt.ylabel('2nd-comp')
plt.show()
```



이미지 데이터의 전처리

```
from google.colab import files
img0 = files.upload()
```

```
파일 선택 swiss-army-ant.jpg
• swiss-army-ant.jpg(image/jpeg) - 16013 bytes, last modified: 2018. 12. 31. - 100% done
Saving swiss-army-ant.jpg to swiss-army-ant.jpg
```

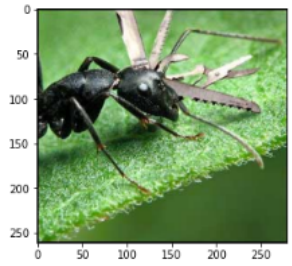
roadbook/242 에서 bees.zip과 ants_2.zip을 각각 다운받습니다. 압축을 해제하세요. ants_2 폴더에서 'swiss-army-ant.jpg'을 업로드합니다.

```
import cv2
img = cv2.imread('swiss-army-ant.jpg')
img.shape
```

```
(261, 280, 3)
```

위 그림처럼 이미지를 읽습니다.

```
import matplotlib.pyplot as plt
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```



위 그림처럼 이미지를 출력해 봅니다. imread에서는 BGR로 읽습니다. cvtColor는 BGR로 읽은 순서를 RGB로 변경해 줍니다.

```
print(img)
```

```
[[[ 60   4  33]
  [ 28   0  10]
  [ 12   7   9]
  ...
  [  4   2   0]
  [ 13   0   2]
  [ 17   0   4]]
```

이미지를 인쇄해 보세요. 수치를 통해서도 대략적인 모양을 확인해볼 수 있습니다. 배열 전체의 크기와 첫번째 행의 크기, 첫번째 행과 첫번째 열의 크기를 확인해 봅니다.

```
print(len(img))
print(len(img[0]))
print(len(img[0][0]))
```

```
261
280
3
```

다만, 이 상태로는 좀 어림짐작 형태일뿐이므로 보다 확실하게 판다스로 표시해보겠습니다.

```
import pandas as pd
b, g, r = cv2.split(img)
b_df = pd.DataFrame(b)
print(b_df.shape)
b_df.head()
```

(261, 280)

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
0	60	28	12	0	0	0	1	0	0	0	...	2	0	0	6	0	0	0	4	13	17
1	28	16	21	0	0	0	0	0	0	0	...	1	0	0	0	0	2	12	15	19	4
2	18	24	92	81	83	69	86	89	89	85	...	116	113	119	115	107	120	125	127	99	11
3	0	0	82	77	74	60	75	78	72	71	...	109	108	107	97	96	110	116	130	92	0
4	0	0	84	75	66	55	67	69	70	68	...	96	94	93	87	90	102	109	135	105	0

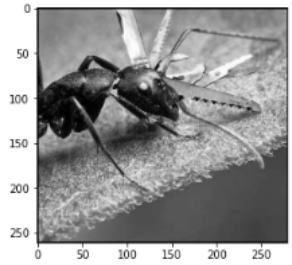
5 rows × 280 columns

이미지를 b, g, r로 분할하고 판다스 자료로 만든 후 인쇄한 모습입니다. 이미지의 픽셀 형태를 한 눈에 파악할 수 있습니다.

다음의 코드는 bgr 이미지를 그레이스케일로 변환하여 표시해 줍니다. 그레이스케일은 RGB 이미지와는 달리 1차원의 픽셀값만을 가지고 있고, 밝기를 나타냅니다. cmap으로 그래프의 색을 'gray'로 지정합니다.

```
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print(gray_img.shape)
plt.imshow(gray_img, cmap='gray')
plt.show()
```

(261, 280)



판다스 데이터프레임으로도 확인할 수 있습니다.

```
gr_df = pd.DataFrame(gray_img)
print(gr_df.shape)
gr_df.head()
```

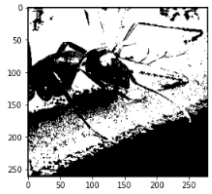
(261, 280)

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
0	19	6	8	3	8	6	9	3	5	5	...	1	1	2	13	4	4	2	2	2	3
1	6	4	30	14	21	11	12	6	12	12	...	10	9	6	12	9	18	24	22	17	0
2	14	33	127	131	138	122	130	130	130	129	...	146	145	153	150	144	155	157	154	117	27
3	3	16	132	143	145	129	135	135	129	129	...	152	152	153	145	144	158	159	167	122	6
4	6	22	139	146	144	131	133	131	134	134	...	144	144	143	138	141	153	155	176	139	14

5 rows × 280 columns

그레이스케일보다 정보를 더 줄여서 특정량을 더 부각시킨 것이 이진화 이미지(binarization image)입니다. 이진화 이미지는 픽셀값이 경계값보다 크면 255(흰색)을 주고, 경계값보다 작으면 0(검은색)을 넣어서 흑백이미지로 만듭니다.

```
ret, bin_img = cv2.threshold(gray_img, 128, 255, cv2.THRESH_BINARY)
plt.imshow(bin_img, cmap='gray')
plt.show()
```



이미지값을 출력해보면 0과 255로 구성된 것을 알 수 있습니다.

```
print(bin_img)
```

```
[[  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ... 255   0   0]
 ...
 [  0   0   0 ...   0   0   0]
 [  0   0 255 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
```

데이터프레임으로 다시 한 번 출력해보겠습니다.

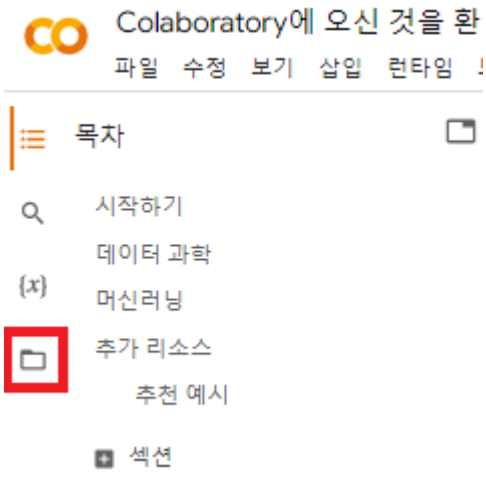
```
bin_df = pd.DataFrame(bin_img)
print(bin_df.shape)
bin_df.head()
```

(261, 280)

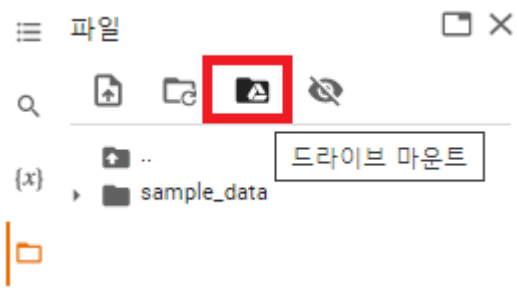
	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	255	255	0	255	255	255	255	...	255	255	255	255	255	255	255	255	0	0
3	0	0	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	0	0
4	0	0	255	255	255	255	255	255	255	255	...	255	255	255	255	255	255	255	255	255	0

5 rows × 280 columns

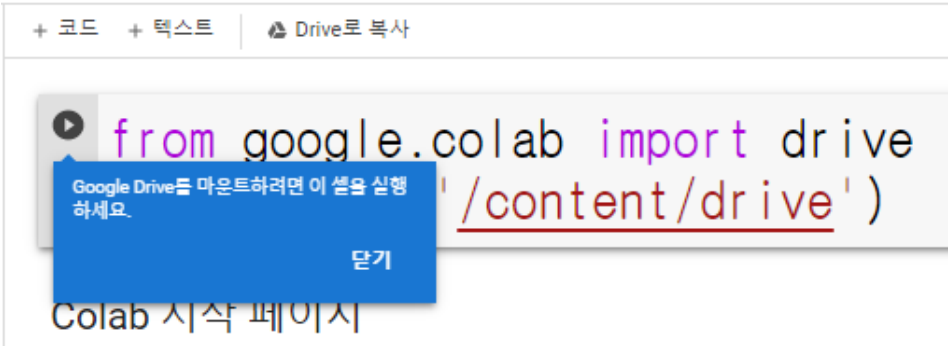
이제부터 개미와 벌의 이미지를 읽어서 개미인지 벌인지를 구분하려고 합니다. 코랩에 두개의 폴더가 필요한데, 코랩 자체적으로는 작업이 되지 않으므로 구글드라이브를 통해 연동되도록 만들겠습니다. 현재 작업하고 있는 브라우저는 그대로 두고, 브라우저 탭을 열거나 크롬 새창을 열어 구글 코랩에 다시 들어갑니다.



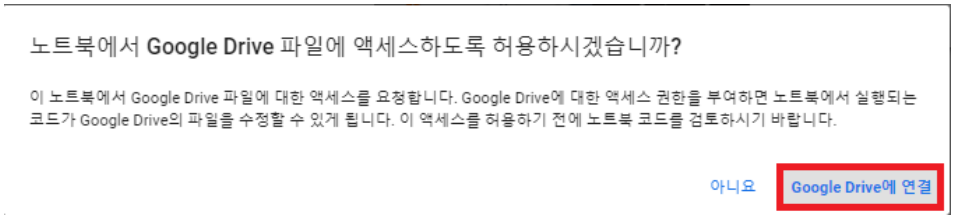
위 그림처럼 폴더 이미지를 클릭하세요.



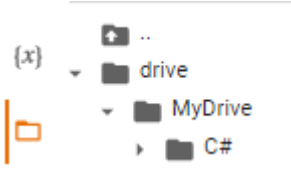
그러면 보이는 아이콘 중 드라이브 마운트를 클릭하세요.



위 셀이 나타나면 셀을 실행합니다. 그러면 다음 창이 뜹니다.



'Google Drive에 연결'을 선택하면 구글 계정 선택 팝업창이 뜹니다. 자기 계정을 선택하고 허용을 눌러줍니다. 그러면 원래 있는 sample_data 폴더



위에 drive 폴더가 생깁니다. drive 폴더를 눌러주면 MyDrive라는 구글 드라이브가 나타나며 하위 폴더에 본인의 구글 드라이브 폴더가 표시됩니다. MyDrive 폴더 밑에 data 폴더를 생성하고, data 폴더 밑에 ants와 bees 폴더를 각각 만들어줍니다. data 폴더 만드는 일은 구글 드라이브에서 해야 하겠죠? https://roadbook.co.kr/242에서 bees.zip과 ants_2.zip을 각각 다운받습니다. 압축 해제 후 bees와 ants_2 폴더에서 각각 10장 정도의 이미지를 구글드라이브의 해당 폴더에 복사합니다. ants_2에는 'swiss-army-ant.jpg'를 포함하도록 하세요. 모든 이미지를 복사하면 좋겠지만 실행에 시간이 걸릴 수 있으므로 대상 파일을 좀 줄이는 것이 더 나을 것 같습니다.