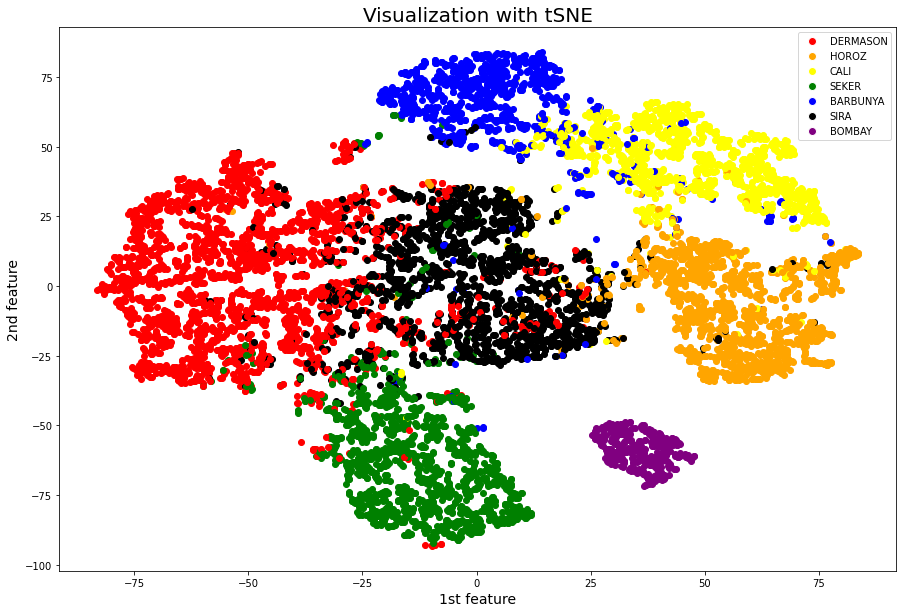
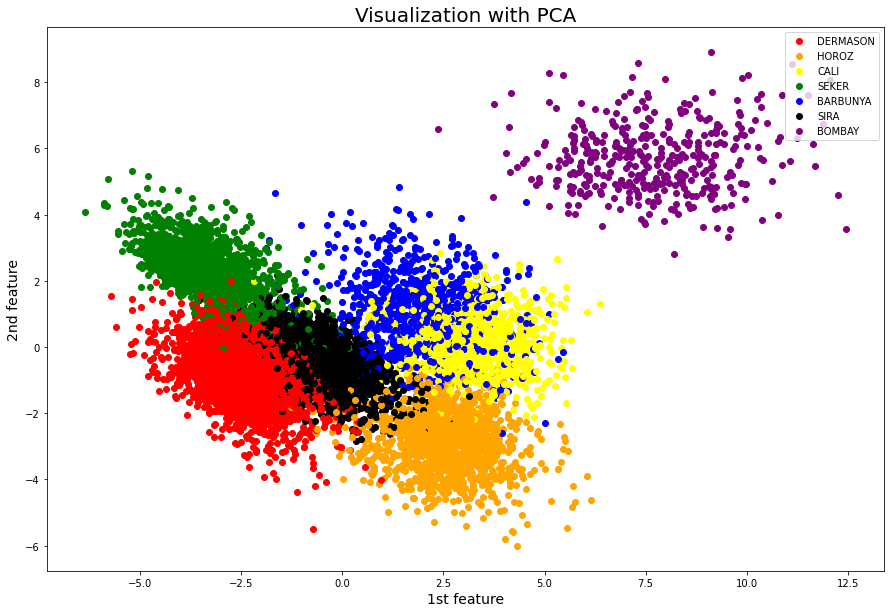
**Assignment 4 : Data Visualization**

**2016314364 박수헌**

UCI Machine Learning Repository에서 Dry Bean Dataset을 선택했다. 총 데이터 instance의 개수는 13611이며, 각 instance 마다 attribute의 개수는 17개이다. 그 중 마지막 한 개는 class로 어떤 종류의 dry bean인지에 대한 정보를 갖고 있다. Class로는 Seker, Barbunya, Bombay, Cali, Dermosan, Horoz, Sira가 있다. 따라서 이는 16개의 정수, 실수의 attribute 값들을 가지는 classification 문제인 것을 알 수 있었다. attribute들로는 area, perimeter, major/minor axis length, aspect ratio 등등이 있었다.

우선 dataset의 파일이 xlsx 파일이었으므로, 이를 csv 파일로 변환했다. 그 후, 이 파일을 읽어 들여, 각 줄에 대해 마지막 attribute는 label이므로, y라는 list에 넣어 저장했고, 나머지는 x라는 list에 float 형식으로 넣어 저장했다.

그 후, train\_test\_split을 이용해 전체데이터셋의 25퍼센트를 test set으로 분리했다. Test set과 train set의 attribute 값들 모두 standard scaler를 이용해 scaling해주었다. 그래프로 쉽게 보이게 하기 위해, PCA와 tSNE의 n\_components들을 모두 2로 설정했다. 그에 따른 결과는 아래와 같았다. x축과 y축은 각각 2차원으로 축소된 attribute들 두 개를 나타낸다. 그에 따른 label값은 점들의 색깔로 나타냈다. 점들의 범례는 차트의 우상단에 색깔별로 표기되어 있다.



우선, 각 결과의 x축과 y축의 범위를 보면, PCA를 활용했을 때는 폭이 20을 넘지 않았다. 하지만 tSNE를 사용했을 때에는 폭이 x, y축 모두 150이 넘는 수치를 보였다. 그럼에도 불구하고 PCA를 활용한 그래프에서 점들이 더 잘 밀집해 있는 모습을 볼 수 있었다. 또한, 겹치는 영역이 존재하긴 하지만 tSNE에 비교해보았을 때는 비교적 확실한 경계선을 가지고 있다. 즉, 이 데이터셋의 compression 방법으로는 PCA가 더 알맞다는 것을 알아냈다.

tSNE 방법은 가까운 데이터들은 가깝게, 멀리 있는 데이터들은 더 멀어지게 차원을 축소한다. 따라서, 다른 데이터들에 비해 확연하게 멀리 떨어져 있던 BOMBAY를 보면 그 차이를 알 수 있다. PCA는 전체적으로 꽤나 잘 분리되어 보이지만, BOMBAY class를 더 잘 구별해낸 것은 tSNE라고 볼 수 있다. 원래 멀리 있던 데이터를 확실히 구별해내는 데에는 tSNE가 더 나은 성능을 보인다는 것을 알아냈다.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

file = open('Dry\_Bean\_Dataset.csv')

x = list()

y = list()

first = file.readline()

while True:

  data = file.readline()

  if data == '' :

    break

  data\_split = data.split(',')

  y.append(data\_split[-1][1:-2])

  data\_split.pop()

  temp = list()

  for i in data\_split:

    temp.append(i[1:-1])

x.append(temp)

x\_train,x\_test, y\_train,y\_test = train\_test\_split(x,y,stratify=y, random\_state = 1016)

scaler = StandardScaler()

scaler.fit(x\_train)

x\_train\_sc = scaler.transform(x\_train)

x\_test\_sc = scaler.transform(x\_test)

pca = PCA(n\_components=2)

pca.fit(x\_train\_sc)

x\_train\_pca = pca.transform(x\_train\_sc)

x\_test\_pca = pca.transform(x\_test\_sc)

print("Original : {}".format(str(x\_train\_sc.shape)))

print("Reduced : {}".format(str(x\_train\_pca.shape)))

color\_label = y\_train[:]

for i in range(len(color\_label)):

  if color\_label[i] == 'DERMASON':

    color\_label[i] = 'red'

  elif color\_label[i] == 'HOROZ':

    color\_label[i] = 'orange'

  elif color\_label[i] == 'CALI':

    color\_label[i] = 'yellow'

  elif color\_label[i] == 'SEKER':

    color\_label[i] = 'green'

  elif color\_label[i] == 'BARBUNYA':

    color\_label[i] = 'blue'

  elif color\_label[i] == 'SIRA':

    color\_label[i] = 'black'

  elif color\_label[i] == 'BOMBAY':

    color\_label[i] = 'purple'

from matplotlib.lines import Line2D

import matplotlib.pyplot as plt

%matplotlib inline

plt.figure(figsize=(15, 10))

for i in range(len(x\_train\_pca)):

  plt.scatter(x\_train\_pca[i][0],

            x\_train\_pca[i][1],

            c = color\_label[i])

legend\_elements = [Line2D([0], [0], marker='o', color = 'w', label = 'DERMASON', markerfacecolor='red', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'HOROZ', markerfacecolor = 'orange', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'CALI', markerfacecolor = 'yellow', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'SEKER', markerfacecolor = 'green', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'BARBUNYA', markerfacecolor = 'blue', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'SIRA', markerfacecolor = 'black', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'BOMBAY', markerfacecolor = 'purple', markersize=8),]

plt.legend(handles = legend\_elements, loc='upper right')

plt.title('Visualization with PCA', fontsize=20)

plt.xlabel('1st feature', fontsize=14)

plt.ylabel('2nd feature', fontsize=14)

plt.show()

from sklearn.manifold import TSNE

tsne = TSNE(random\_state = 1016)

x\_train\_tsne = tsne.fit\_transform(x\_train\_sc)

print("Original : {}".format(str(x\_train\_sc.shape)))

print("Reduced : {}".format(str(x\_train\_tsne.shape)))

from matplotlib.lines import Line2D

import matplotlib.pyplot as plt

%matplotlib inline

plt.figure(figsize=(15, 10))

for i in range(len(x\_train\_tsne)):

  plt.scatter(x\_train\_tsne[i][0],

            x\_train\_tsne[i][1],

            c = color\_label[i])

legend\_elements = [Line2D([0], [0], marker='o', color = 'w', label = 'DERMASON', markerfacecolor='red', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'HOROZ', markerfacecolor = 'orange', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'CALI', markerfacecolor = 'yellow', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'SEKER', markerfacecolor = 'green', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'BARBUNYA', markerfacecolor = 'blue', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'SIRA', markerfacecolor = 'black', markersize=8),

                   Line2D([0], [0], marker='o', color= 'w', label = 'BOMBAY', markerfacecolor = 'purple', markersize=8),]

plt.legend(handles = legend\_elements, loc='upper right')

plt.title('Visualization with tSNE', fontsize=20)

plt.xlabel('1st feature', fontsize=14)

plt.ylabel('2nd feature', fontsize=14)

plt.show()