Machine Learning Assignment 2

20186889 권용한

1. Experimental setup

   윈도우 11, VSCode 파이썬 3.10.10에서 실행하였습니다.

2. 설계 및 소스코드 분석

   1) KMeans

```
pca = PCA(n_components= 100)
test_PCA = pca.fit_transform(X_)
test_PCA = pd.DataFrame(data = test_PCA)

X_test_tSNE = tSNE.fit_transform(test_PCA.loc[:,:])
X_test_tSNE = pd.DataFrame(data=X_test_tSNE)
testDF = pd.DataFrame(data=X_test_tSNE.loc[:,:], index=test_PCA.index)
testDF = pd.concat((testDF,y_), axis=1, join="inner")
testDF.columns = ["x-axis", "y-axis", "Label"]

############ This part should include K-means algorithm ################

kmeans=KMeans(n_clusters=10,random_state=0,n_init="auto").fit(X_test_tSNE)
labels=kmeans.labels_
########################################################################

testDF["Label"] = labels
sns.lmplot(x="x-axis", y="y-axis", hue="Label", data=testDF, fit_reg=False,
height=8)
plt.title("Clustering Result")
plt.grid()

metrics.adjusted_rand_score(labels,y_)
```

   기본적인 소스코드는 해당 소스코드에서 input을 조정하거나 pca의 parameter를 조정하는 방식으로 작성하였습니다.

   Pca를 적용하지 않은 경우엔 kmeans에 X_test_tSNE이 아닌, X_를 input으로 사용하였으며, pca를 적용한 경우엔 pca의 parameter n_components를 100, 50, 10으로 변경하여 진행하였습니다.

   이후 배정된 label을 실제 label과 비교해 정확도를 측정하였습니다.

2) DBSCAN

```
pca = PCA(n_components= 100)
test_PCA = pca.fit_transform(X_)
test_PCA = pd.DataFrame(data = test_PCA)

X_test_tSNE = tSNE.fit_transform(test_PCA.loc[:,:])
X_test_tSNE = pd.DataFrame(data=X_test_tSNE)
testDF = pd.DataFrame(data=X_test_tSNE.loc[:,:], index=test_PCA.index)
testDF = pd.concat((testDF,y_), axis=1, join="inner")
testDF.columns = ["x-axis", "y-axis", "Label"]

################## This part should include K-means algorithm ##################

dbscan=DBSCAN(eps=2,min_samples=6).fit(X_test_tSNE)
labels=dbscan.labels_
################################################################################

testDF["Label"] = labels
sns.lmplot(x="x-axis", y="y-axis", hue="Label", data=testDF, fit_reg=False,
height=8)
plt.title("Clustering Result")
plt.grid()

metrics.adjusted_rand_score(labels,y_)
```

DBSCAN의 경우에도, KMeans와 동일한 방식으로 코드를 작성하였습니다.

Pca를 적용하지 않은 경우엔 X_를 input data로 사용하였으며, pca를 적용한 경우엔 각각 100, 50, 10으로 parameter를 조정하여 진행했습니다.

마찬가지로, 배정된 label을 실제 label과 비교해 정확도를 출력하였습니다.

DBSCAN의 parameter인 eps와 min_samples의 경우엔, 각각 2과 6를 선택하였습니다.

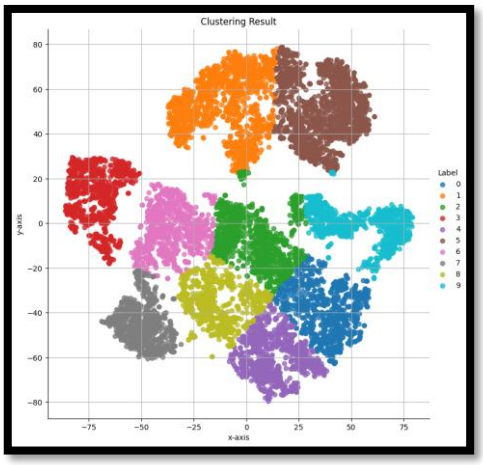
이는 가능한 label이 10개에 가깝게 나올 수 있는 수치를 수차례의 실행을 통해 찾아낸 결과값입니다.

3. Result

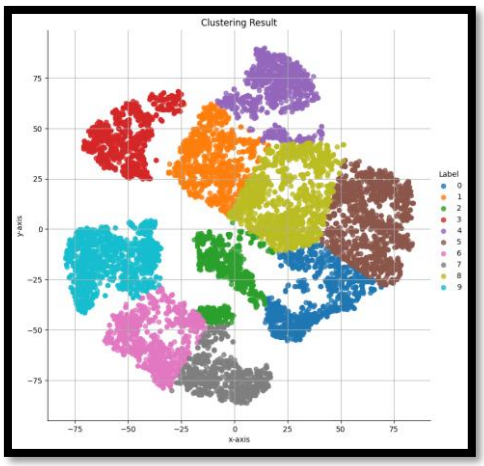
Table for KMeans

dimensions	784	100	50	10
ARI	0.4608202842559773	0.40056670421173496	0.4034428974673212	0.42136209155864074

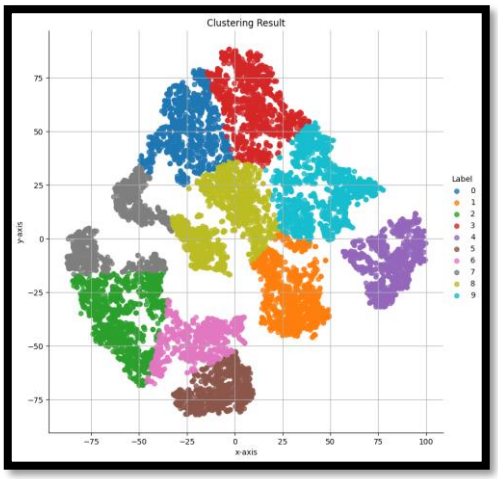
KMeans without PCA



KMeans PCA 100



KMeans PCA 50



KMeans PCA 10

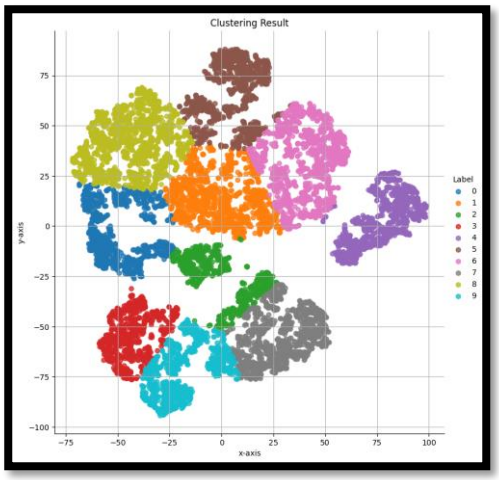
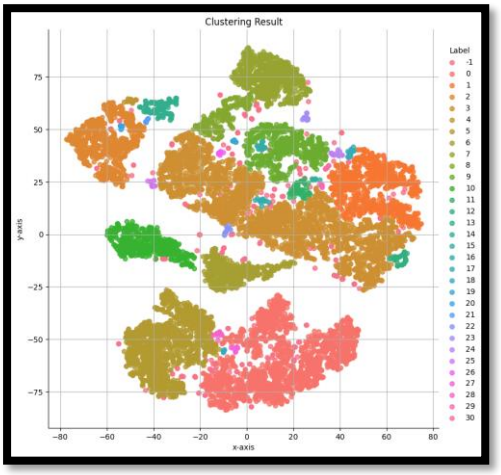


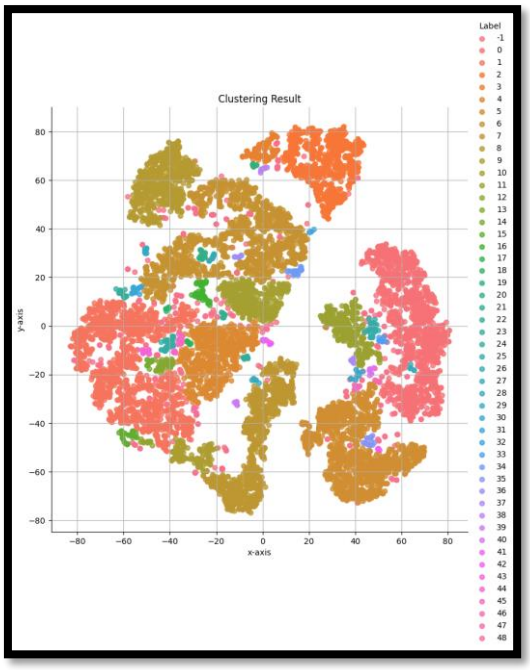
Table for DBSCAN

dimensions	784	100	50	10
ARI	0.34835031666065536	0.4655138233687382	0.30067866282432276	0.35539525894625545

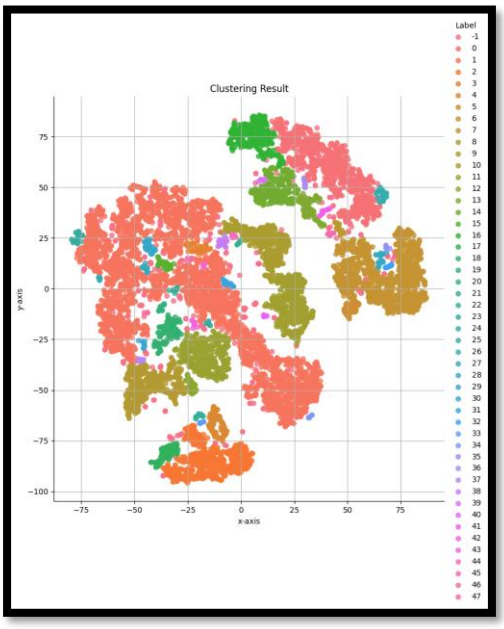
DBSCAN without PCA



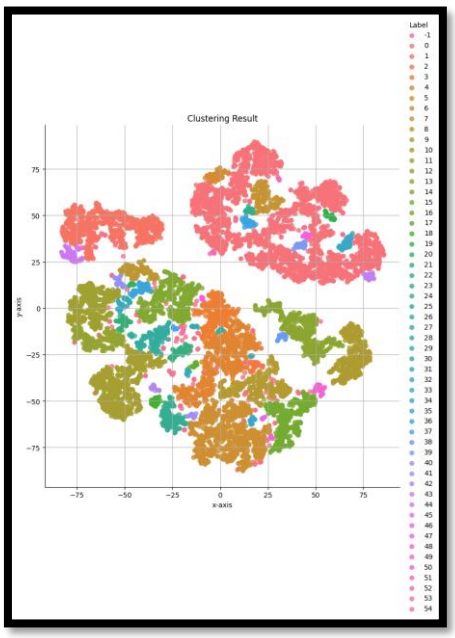
DBSCAN PCA 100



DBSCAN PCA 50



DBSCAN PAC 10



KMeans를 쓴 경우 대략 0.4 정도의 값이 나오면서 약 0.3의 결과를 보여주는 DBSCAN을 사용한 경우보다 미세하게 더 우수한 성능을 보여주는 것을 확인할 수 있었습니다.

PCA를 사용하거나 n_components 값을 조정하더라도 일정한 방향성을 보이지 않고, 대체로 유사한 수치를 유지하는 것을 확인 가능했으며, 실행 시간의 측면에서 PCA를 통해 feature의 수를 줄인 경우에 결과가 더 빨리 도출되는 것을 확인할 수 있었습니다.

이를 통해 PCA를 사용하면 연산시간과 메모리를 효과적으로 사용하면서도 기존 데이터와 유사한 결과를 얻을 수 있음을 확인하였습니다.

KMeans와 DBSCAN의 결과 사진을 비교해보면, KMeans의 경우, 기존의 label 개수와 같은 수로 분리된 반면, DBSCAN의 경우 label이 수십개로 된 모습을 확인할 수 있는데, 이러한 측면에서는 k의 값을 안다면 KMeans를 사용하는 것이 더 훌륭한 결과값을 보여주는 것을 확인할 수 있었다고 생각됩니다.

다만 DBSCAN을 사용할 경우 not convex한 경우에서도 잘 clustering하는 것을 확인할 수 있었습니다. 반면 KMeans의 경우, fitting 결과가 convex하며, 때문에 경계값에 있는 값들의 경우 혹은 분포가 다른 분포에 속하더라도 거리가 더 가깝다는 이유로 다른 label로 판단되는 것을 확인할 수 있었습니다.

Eps와 min_samples를 조정한다면, DBSCAN에서도 결과가 다르게 나오는 것을 볼 수 있으며, 제가 eps와 min_samples를 결정한 방법은 간단하게 PCA를 적용하지 않은 모델을 for문을 통해 여러 번 생성해보고, 그 중 가장 우수한 ARI를 보여주는 경우의 eps와 min_samples를 찾았습니다. 이를 통해 획득한 수치로 모델을 만들고 확인해 본 결과, outlier나 주된 분포에서 조금 떨어진 부분을 제외하고 큰 분포 위주로 확인한다면 기존의 y label과 유사한 수로 구분된 것을 확인할 수 있었습니다.

위의 결과를 종합해보면, 현재 데이터에서 KMeans는 K를 정확히 알고 있어 훌륭한 결과를 얻었으나, eps와 min_samples를 잘 조정해서 DBSCAN의 분류 결과 label의 수가 K와 같거나 유사하게 도출할 수 있다면, DBSCAN 또한 훌륭한 결과를 보여주는 것을 확인할 수 있으며, 데이터에 따라서는 더 훌륭한 결과를 보여줄 수 있을 것이라 생각합니다.