

가상데이터 셋 생성과 기계학습 알고리즘을 이용한 분류 비교 분석

Analysis of Machine Learning algorithms on Generated Datasets

유호영¹ · 이가람²

You, Ho Yeong · Lee, Ga Ram

Abstract: We have applied supervised and unsupervised learning, widely used in machine learning, to classify and cluster on a virtual dataset. We utilized various virtual data generation functions provided by the Scikit-learn package. Supervised learning is broadly divided into regression and classification. The representative supervised learning algorithms that are publicly available include Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Decision Trees. Algorithms used for unsupervised learning include methods such as K-means clustering, PCA and t-SNE. Before applying these models, we preprocessed (standardized and normalized) the data created with the virtual data generation function. Afterward, to discover the optimal hyperparameters of the model, we extracted the hyperparameters through the grid search feature. Through this, we were able to obtain an enhanced performance machine learning model.

요약: 기계학습에서 많이 쓰이는 지도 학습과 비지도 학습을 활용하여 분류 및 클러스터링을 가상 데이터 셋에 적용해 보았다. 특히 Scikit-learn 패키지에서 제공하는 다양한 가상 데이터 생성 함수를 이용했다. 지도 학습에는 크게 회귀와 분류로 나누어 진다. 공개되어 있는 대표적인 지도 학습 알고리즘은 로지스틱 회귀, Support Vector Machine(SVM), K-최근접 이웃(KNN), 결정 트리 등이다. 비지도 학습을 하기 위해 사용된 알고리즘은 K-means 클러스터링, PCA, t-SNE 등의 방법이다. 이러한 모델들을 적용하기 이전에 가상 데이터 생성 함수로 만든 자료들을 전처리(표준화, 정규화) 하는 과정을 거쳤다. 이후, 모델의 최적 하이퍼 파라미터를 알아내기 위해 그리드 서치 기능을 통하여 하이퍼 파라미터를 추출하였다. 이를 통해 최적화된 기계학습 모델을 얻을 수 있었다.

Github link: <https://github.com/leeolivine/PBML-assignment/>

¹ 에너지자원공학과 4학년

² 지구자원시스템공학과 3학년

1. 서 론

기계학습은 최근 떠오르는 인공지능의 중요한 분야 중 하나로, 컴퓨터가 데이터를 통해 스스로 학습하고, 예측하는 능력을 개발하는 과정을 말한다. 학습된 지식을 사용하여 사회의 다양한 문제와 많은 어려운 과제들을 해결할 수 있다. 하지만 기계학습을 통한 유의미한 결과를 얻으려면 적절한 데이터 셋을 사용하는 것이 필수적이다. 우리가 만드는 모델의 성능은 주로 학습 데이터에 영향을 많이 받는다. 모든 경우의 수를 고려할 수 있도록 가능한 모든 수를 반영하는 일반적인 데이터 셋을 잘 활용하는 것이 정확한 예측을 할 수 있도록 해준다. 반대로, 특정 상황에 맞는 예측을 위해서는 특정 자료와 파라미터를 잘 선정하는 것이 중요할 수도 있다. 이렇게, 더 많은 상황과 변수를 제어할 수 있도록 가상 데이터를 직접 생성해서 사용할 수 있다. 이를 통해 특정한 문제 상황, 변수들이 결과에 미치는 영향을 조사하는 데 유용하게 활용될 수 있다. 또한, 실제 빅데이터를 취득하는 과정은 비용과 시간이 많이 듈다. 가상 데이터를 생성해서 이를 절약할 수 있는 장점도 있다. 따라서 본 연구에서는 Scikit-learn 패키지를 활용한 가상 데이터 생성과, 자료 전처리, 모델 선정과 학습 그리고 검증 단계까지 진행해 보려 한다.

2. 연구 방법

2.1 Scikit-learn

Scikit-learn은 기계학습에 사용되는 강력하고 유연한 파이썬 라이브러리이다. Scikit-learn은 지도학습, 비지도학습, 모델 선택, 검증, 데이터 전처리 등의 기능을 제공하고 있으며 무료로 사용이 가능하다. 또한 내장 데이터셋과 가상 데이터셋 생성 기능을 활용할 수 있어 기계학습에 대해 연습하기 좋은 라이브러리이다.

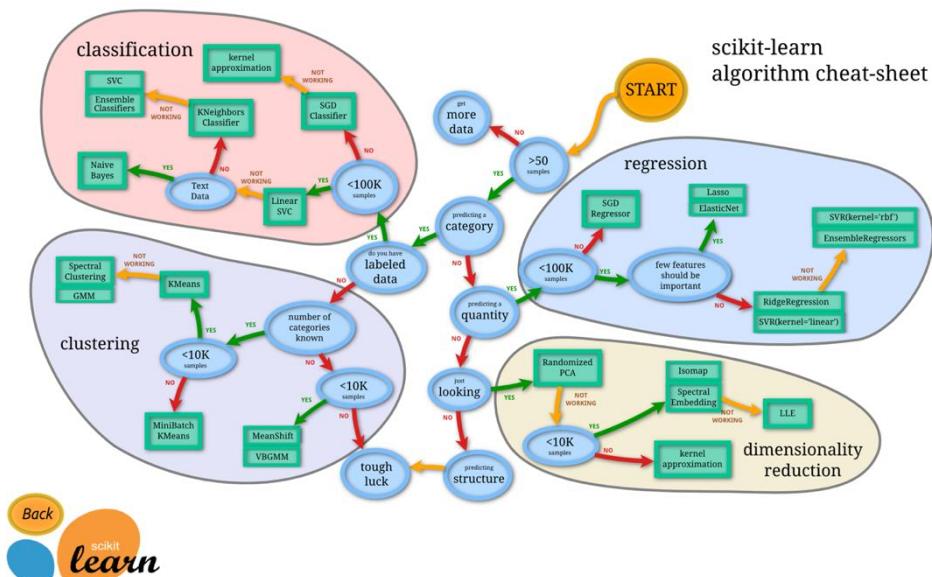


Fig 1. Scikit-learn algorithm cheat-sheet

2.1.1 Supervised Learning

지도학습은 주어진 입력 데이터와 이에 해당하는 레이블(Label)로부터 학습하는 방식이다. 이 방식을 통해 학습된 모델은 새로운 입력 데이터에 대해 예측을 수행하게 된다. 지도 학습에는 주로 두 가지 종류의 방법이 있다: 회귀(Regression)와 분류(Classification). 회귀는 연속적인 값 예측에 사용되고, 분류는 이산적인 클래스 레이블 예측에 사용된다. 알고리즘으로는 선형 및 로지스틱 회귀, Support Vector Machine(SVM), K-최근접 이웃(KNN), 결정 트리 등이 있다.

선형 회귀는 가장 간단하고 보편적인 회귀용 선형 알고리즘이며, 예측 변수와 목표 변수 사이의 관계가 선형 관계임을 가정한다. 선형 회귀는 주어진 데이터셋을 가장 잘 설명하는 선을 찾는 과정이다. 로지스틱 회귀는 분류 문제에 주로 사용되는 선형 모델이다. 로지스틱 회귀는 선형 회귀와 비슷하게, 데이터의 특징을 바탕으로 특정 결과를 예측하려고 한다. 하지만 로지스틱 회귀의 값은 선형 회귀와는 다르게, 로지스틱 함수를 통과시켜서 출력이 0과 1 사이의 값이 되게끔 하는 것이 특징이다. SVM은 분류나 회귀, 이상치 탐지 등에 사용할 수 있는 기계학습 모델이다. SVM은 데이터를 고차원 공간에 매핑하고, 이 공간에서 초평면(hyperplane)을 찾아 두 클래스를 분리하는 원리를 사용한다. KNN 알고리즘은 분류 알고리즘 중 하나이다. 새로운 데이터 포인트를 분류할 때, KNN은 기준의 데이터 포인트 중에서 가장 ‘가까운’ K개의 이웃을 찾아, 그들의 레이블을 기반으로 새로운 데이터 포인트의 레이블을 결정한다. 여기서 ‘가까운’의 정의는 일반적으로 유clidean 혹은 마할라노비스 거리를 사용하며, K는 사용자가 정의하는 변수이다. 결정 트리는 속성(Feature)을 바탕으로 계층적인 분기를 통해 값을 예측하는 모델이다. 분류와 회귀 둘 다에 사용할 수 있다. 각 분기점(node)에서는 데이터의 한 속성(Feature)이 테스트되며, 이 테스트의 결과에 따라 다음 분기점으로 이동한다. 이 과정은 트리의 리프 노드(leaf node)에 도달할 때까지 반복되며, 리프 노드에는 예측된 클래스 레이블(분류의 경우) 또는 값(회귀의 경우)이 포함되어 있다.

2.1.2 Unsupervised Learning

비지도학습은 입력 데이터에 대한 명시적인 레이블 없이도 학습을 수행하는 기계학습의 한 방법이다. 비지도 학습의 목적은 데이터의 숨겨진 패턴, 구조, 상관 관계 등을 발견하는 것이며, 이는 데이터 셋의 분포를 이해하거나 데이터를 새로운 방식으로 표현하는 데 유용하다. 비지도 학습은 주로 클러스터링, 차원 축소, 연관 규칙 학습 등의 작업에 사용된다. 대표적인 알고리즘으로는 K-means 클러스터링, 주성분 분석(Principal Component Analysis, PCA), t-SNE(t-Distributed Stochastic Neighbor Embedding) 등이 있다.

K-means 클러스터링은 주어진 데이터를 K개의 클러스터로 분할하는 알고리즈다. 알고리즘은 먼저 데이터 공간에서 임의의 K개의 중심을 선택하고, 각 데이터 포인트를 가장 가까운 중심에 할당하여 클러스터를 형성한다. 이후, 각 클러스터의 중심을 새로 계산한다. 이 과정을 중심이 변하지 않을 때까지

반복한다. PCA는 데이터의 차원을 줄이는 동시에 데이터의 분산을 최대한 유지하는 선형 변환 기법이다. 이는 데이터의 고차원 속성들이 가지는 상관관계를 이용하여 분산이 큰 주성분을 추출하고, 이들 주성분을 새로운 기저로 사용하여 데이터를 표현한다. t-SNE는 고차원 데이터를 시각화하기 위해 주로 사용되는 비선형 차원 축소 방법이다. t-SNE는 고차원 공간의 데이터 포인트 간 유사도를 확률적으로 정의하고, 이와 비슷한 확률 분포를 낮은 차원에서 재 생성한다. 이 방법은 특히 데이터의 군집 구조를 보존하는 데 효과적이다.

각 알고리즘은 데이터의 특성에 따라 다르게 작동하며, 올바른 알고리즘을 사용하기 위해서는 각 알고리즘의 가정과 제약 조건을 잘 이해해야 한다. 예를 들어, K-means 클러스터링은 클러스터가 대칭일 때 가장 잘 작동하는 경향이 있다. 이런 가정에 데이터가 맞지 않으면 K-평균은 잘못된 클러스터를 형성할 수 있다. PCA는 데이터가 선형적인 구조를 가지고 있을 때 가장 잘 작동한다. 만약 데이터가 비선형 구조를 가지고 있다면, PCA는 정보 손실이 발생할 수도 있다. t-SNE는 데이터의 구조를 보존하는데 강점을 가지고 있지만, 군집 간의 거리를 보존하는 데는 약점이 있다. 즉, t-SNE는 군집 자체는 잘 형성하지만 이 군집들이 어떻게 공간에 배치되는지는 신뢰하기 어려울 수 있다. 따라서, 알고리즘을 선택할 때는 항상 해당 알고리즘의 장단점을 이해하고, 주어진 데이터와 문제에 어떻게 적용되는지 고려해야 한다. 또한, 최종 결과를 해석할 때는 이러한 특성을 염두에 두는 것이 중요하다.

2.1.3 Model Selection and Validation

주어진 데이터와 문제를 고려해 가장 적합한 모델을 선택하는 것은 매우 중요하다. 모델의 장단점, 결과물에 영향을 미칠 파라미터들을 잘 확인한 후 이를 결정해야 한다. 이후, 선택한 모델이 잘 작동하는지 평가하기 위해 검증 과정을 거친다. Train 데이터는 학습에 사용되고, test 데이터를 통해 모델의 성능을 평가할 수 있다. 검증하는 대표적인 방법은 다음과 같다: 교차 검증(Cross-validation), Grid search, 모델 평가 지표(Model Evaluation Metrics).

교차 검증은 데이터를 여러 세트로 나누고, 각 세트를 돌아가며 모델 성능을 평가하는 방법이다. 한번 나눈 것 보다는 여러 세트의 평균적인 성능을 확인하는 것이 더 정확성이 높기 때문에 이러한 방법을 사용한다. 가장 널리 사용되는 교차 검증 방법 중 하나는 K-fold cross-validation이며, 이는 데이터를 K개로 나눈다. K-1개를 Train 데이터로 그리고 나머지는 test 데이터로 사용하여 K번 반복하여 성능을 평가한다. Grid Search는 모델의 하이퍼 파라미터를 튜닝하는 방법 중 하나로, 가능한 모든 하이퍼 파라미터 조합을 시도하여 최적의 조합을 찾는 과정을 말한다. 이 과정은 교차 검증과 함께 수행되어 각 하이퍼 파라미터 조합의 성능을 평가한다. 모델 평가 지표에는 다양한 방법들이 존재한다. 분류 작업의 경우 정확도, 정밀도, 재현율, F1 점수, ROC-AUC 커브 등이 사용된다. 회귀 작업의 경우 평균 제곱 오차(Mean Squared Error, MSE), 평균 절대 오차(Mean Absolute Error, MAE), R^2 점수 등이 사용된다.

2.1.4 Data Preprocessing

데이터 전처리 과정은 기계학습 수행하기 이전에 수행되는 과정으로 결과물에 큰 영향을 끼칠 수 있으므로 매우 중요하다. 처리 작업으로는 주로 데이터 클리닝, 스케일링, 속성 추출 등의 과정이 포함되어 있다. 데이터 클리닝은 일반적으로 누락된 값이나 이상치를 처리하는 것을 말한다. 누락된 값은 특정 값들로 채워질 수 있거나(Padding), 해당 데이터를 아예 제거(Remove)하여 사용할 수도 있다. 이상치는 모델의 성능을 저하시키므로, 이를 식별하고 처리해야 한다. 또한, 결과물들은 속성들의 스케일에 따라 다르게 나올 수 있으므로 동일 스케일로 변환하는 스케일링 과정은 필요하다. 스케일링에는 주로 정규화(Normalization)와 표준화(Standardization) 두 가지 방법이 있다. 정규화는 속성의 범위를 [0, 1]로 조정하고, 표준화는 피처의 분포를 평균이 0이고 표준편차가 1인 정규 분포로 만드는 과정을 말한다. 마지막으로, 원본 데이터에서 특정 속성값들만 추출하는 것이 유용할 수도 있다. 예를 들어, 날짜와 시간에서 요일이나 특정 날씨의 정보들만 따로 추출하여 사용하는 것이 유리한 상황이 있을 수 있다. 속성 추출은 가장 결과에 도움이 될 만한 속성들을 선택하고 덜 중요한 속성은 제거하여 모델의 복잡성을 줄이고 과적합을 방지하는 데 도움이 될 수 있다.

2.2 Scikit-learn Datasets

2.2.1 *sklearn.datasets*

sklearn.datasets 모듈은 Scikit-learn 라이브러리에 포함된 모듈로, 기계학습 및 데이터 분석을 위한 다양한 데이터셋을 제공한다. 이 모듈의 주요 목적은 기계학습 알고리즘을 쉽게 실험하고 테스트할 수 있도록 사용자들에게 편리한 방법을 제공하는 것이어서 학습하기 매우 좋은 도구이다. 잘 알려진 대표적인 데이터셋은 다음과 같다: Iris Data(붓꽃 데이터), Diabetes(당뇨병 데이터), Digits(손 글씨 데이터), 가상 데이터셋 등.

특히 가상 데이터셋 생성의 경우 사용자들의 개인적인 용도의 실험을 위한 특정 유형의 데이터를 생성할 수 있게 한다. 예를 들어, ‘make_classification’, ‘make_regression’, ‘make_blobs’ 등의 함수는 사용자가 지정한 특성에 따라 분류, 회귀, 클러스터링용 데이터를 생성할 수 있다. 각 데이터셋은 일반적으로 데이터, 타깃, DESCRIPTOR 등의 정보를 포함한다.

2.2.2 *make_blobs* 함수

sklearn.datasets 모듈의 *make_blobs* 함수는 기계학습 알고리즘을 테스트하거나 시각화하기 위한 간단한 다변량 데이터를 생성하는 데 사용되는 도구이다. 샘플과 클러스터 레이블을 포함하는 튜플을 반환한다. 이는 분류 또는 클러스터링 알고리즘을 테스트하는 데 사용될 수 있다. 그렇기에 특히 클러스터링 알고리즘이나 분류 알고리즘을 시험하거나 시각화하는데 유용하게 사용될 수 있다. 사용자가 지정한 수의 클러스터를 가진 데이터 포인트를 생성하며, 각 클러스터는 정규 분포를 따른다.

참고로, ‘*make_blobs*’는 완전히 무작위로 데이터를 생성하기 때문에 실제 기계학습 문제를 완전히 반

영하지는 못한다. 그러나 알고리즘을 이해하고 테스트하는 데는 매우 유용한 도구이다.

다음은 `make_blobs` 함수의 주요 매개변수이다. 대표적으로 생성할 샘플 개수를 정하는 ‘n_samples’, 속성 변수들의 개수를 나타내는 ‘n_features’, 클러스터의 수를 나타내는 ‘centers’, 표준 편차 조정 변수인 ‘cluster_std’ 등이 있다.

종류	함수	설명
Parameters:	n_samples (int or array-like)	정수일 시, 생성할 총 데이터 포인트의 수. 배열형태일 시, 시퀀스의 각 요소는 클러스터 당 샘플 수를 나타냄. (Default = 100)
	n_features (int)	각 데이터 포인트의 변수(features)의 수. (Default = 2). 2차원 공간에서 데이터를 시각화 하는데 유용함.
	centers	생성할 클러스터의 수 또는 중심 위치. 이 매개변수를 통해 클러스터의 위치와 수를 제어할 수 있음. (Default = None)
	cluster_std	클러스터의 표준편차. (Default = 1.0)
	center_box	클러스터 중심을 무작위로 선택할 때 사용되는 바운딩 박스. (Default = (-10.0, 10.0))
	shuffle	샘플들을 섞음. (Default = True)
	random_state	데이터 생성에 사용되는 난수 시드. 이 값을 지정하면 동일한 결과를 생성하는 데 사용할 수 있음. (Default = None)
	return_centers	참으면, 클러스터의 중심을 반환함. (Default = False)

Table 1. Parameters of ‘make_blobs’ function

3. 연구 과정

3.1 Generating Datasets

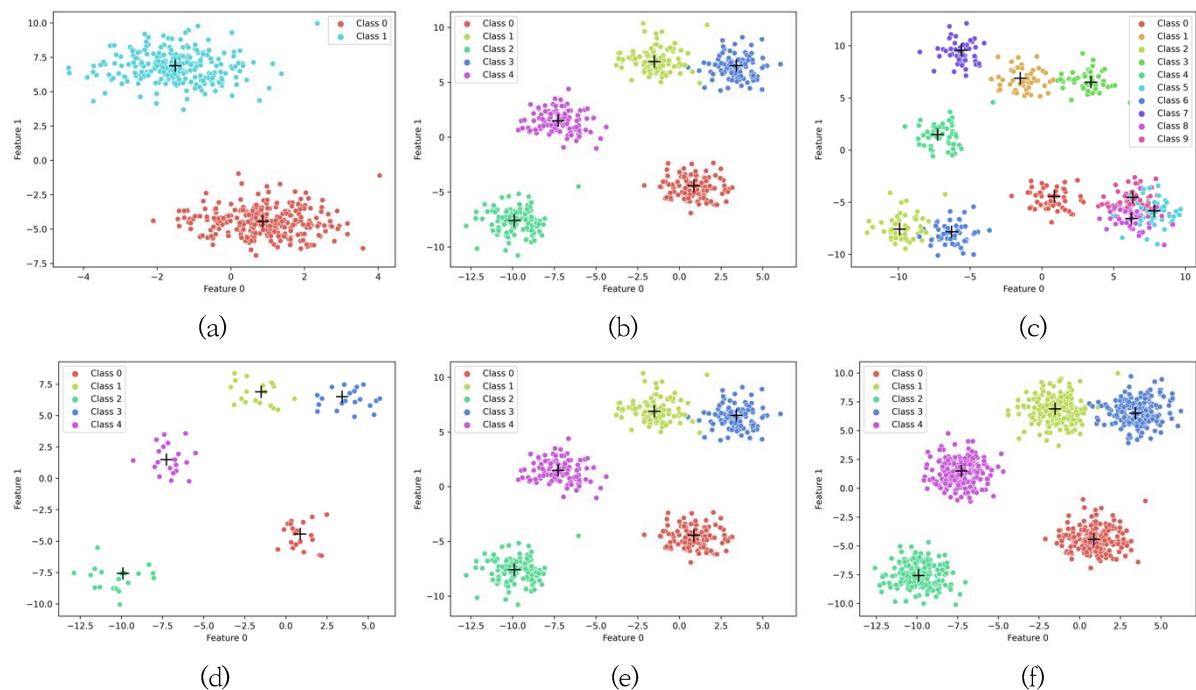
본 연구에서는 4개의 알고리즘을 테스트 및 시각화하기 위해 sklearn.datasets 모듈의 ‘make_blobs’을 통해 총 9가지의 데이터 세트를 생성하였다. 이때 조정한 파라미터는 ‘n_samples’, ‘centers’, ‘cluster_std’의 세 가지 값을 조정해 주었다. 나머지 파라미터는 분류 작업 시 해당 파라미터들이 큰 영향을 미치지 않을 것으로 가정 했기에 조정을 거치지 않았고, 세 가지 파라미터는 분류 작업에 있어 가장 영향을 끼칠 것이라고 생각했기에 조정해 주었다.

‘n_samples’의 경우 생성된 데이터 포인트의 총 개수를 의미하기 때문에 값을 조정해 데이터의 개수에 따라 알고리즘의 성능을 분석하고, 알고리즘에 있어 적절한 데이터 개수를 찾을 수 있다. 본 연구에서는 ‘n_samples’의 값을 100개, 500개, 1000개로 조정해 주었으며 여러 번의 시도 끝에 원래 함수의 Default 값인 100개보다 더 많은 500개로 설정해줘야 다른 파라미터들의 결과가 잘 보인다는 것을 파악한 후 Default 값을 500개로 설정해주었다.

‘centers’는 생성된 클러스터의 개수를 의미하며, 이 값을 조정해 클러스터의 개수에 따라 알고리즘의 성능을 분석할 수 있다. 본 연구에서는 ‘centers’의 값을 2, 5, 10으로 조정해 주었으며 Default 값은 원래의 값과 다른 5로 설정하였는데, 이진으로 분류가 쉬운 2의 값이 아닌, 분류가 어려우면서도 다른 파라미터의 특성을 해치지 않는 값인 5로 설정해주었다.

‘cluster_std’는 각 클러스터의 표준 편차를 나타낸다. 이 값이 클 경우에는 클러스터 내의 데이터 포인트가 중심으로부터 퍼져 있어 클러스터들이 겹치게 되면서 분류 작업이 어려워질 수 있다. 그렇기에 데이터가 겹치는 상황 속에서도 알고리즘이 분류 작업을 잘 수행하는지 확인할 수 있다. 본 연구에서는 1, 2, 3의 값으로 조정해주었으며 Default 값은 원래 함수와 동일하게 1로 설정하였다.

3개의 파라미터를 조정할 때, 조정해주는 한 개의 파라미터만 변화하고 나머지 2개의 파라미터는 본 연구에서 설정한 Default 값으로 고정해 주었다. 그 이유는 본 연구에서는 각 파라미터의 특성이 알고리즘 성능에 어떤 영향을 주는지 확인하고 어떤 데이터 셋을 활용할 때 가장 적절한 알고리즘인지 확인하고자 하는 것이 목적이기 때문이다. 다음은 데이터 셋을 시각화한 그림이다.



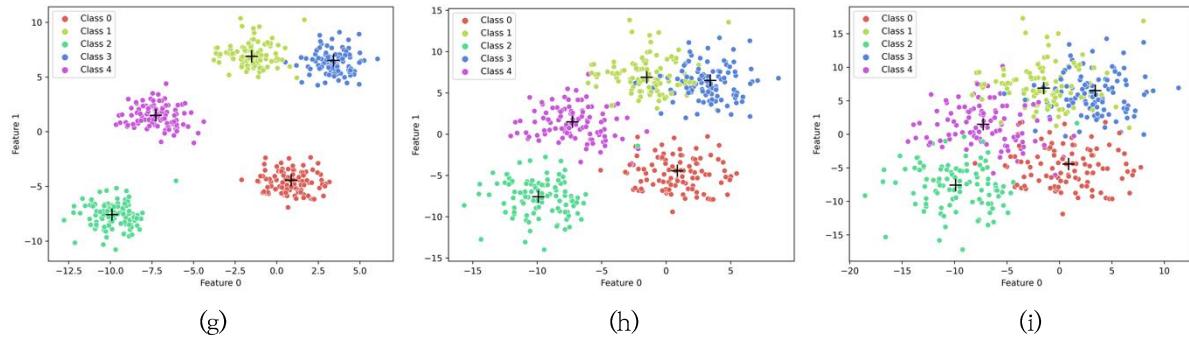


Fig 2. Datasets (a) centers=2, (b) centers=5, (c) centers=10, (d) n_samples=100, (e) n_samples =500, (f) n_samples =1000, (g) cluster_std=1, (h) cluster_std =2, (i) cluster_std =3

3.2 Supervised Learning Model

본 연구에서 지도 학습을 위해 사용한 알고리즘은 로지스틱 회귀와, Support Vector Machine (SVM) 방법이다. 생성한 데이터 셋을 훈련 데이터와 검증 데이터로 나누어 검증 데이터는 레이블에 할당한다. 이때, 레이블은 입력 데이터와 이에 상응하는 출력을 의미한다. 즉, 지도 학습 모델은 주어진 입력에 대한 올바른 출력을 이미 알고 있으며, 이를 바탕으로 학습한다. 지도 학습 모델의 경우 주어진 학습 데이터와 레이블을 지나치게 의존하여 과적합이 될 위험이 있다. 이러한 상황을 예방하기 위해 교차 검증, 정규화와 같은 처리 및 검증 단계를 거치게 된다. 본 연구에서는 훈련 데이터와 시험(검증) 데이터의 비율을 8:2의 비율로 나누었다.

3.2.1 Logistic Regression

로지스틱 회귀는 분류 문제를 해결하는 데 사용되는 통계적인 기계학습 알고리즘 중 하나이다. 이 기법은 주로 이진 분류 문제에 사용된다. 즉, 결과 변수가 두 가지 다른 범주 중 하나에 속하도록 예측하는 것이 목표이다. 로지스틱 회귀는 선형 회귀와 비슷한 방식으로 작동하지만, 결과를 이진 결과로 변환하는 데 로지스틱 함수를 사용한다는 점에서 차이가 있다. 지도 학습의 처리 과정은 다음과 같다.



Fig 3. Flow chart for Supervised Learning

Algorithm 1: Logistic Regression

Input: Datasets, Model Parameters

Output: Accuracy Score, Confusion Matrix, Classification result

1. Generate blobs of data (X, y , centers)

-
2. Split Data ($X, y \rightarrow (X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}})$)
 3. Data Preprocessing \rightarrow Normalization or Standardization
 4. Hyperparameter (Dict) for Logistic Regression
 5. KFold for Cross validation
 6. Grid Search \leftarrow sklearn GridSearchCV
 7. Classifier \leftarrow Best estimator and parameter
 8. Classifier.fit(Training Data)
 9. Classification report, accuracy score, and confusion matrix
 10. Validation using y_{predict}
-

Table 2. Pseudo code for Logistic Regression

3.2.2 Support Vector Machine

SVM은 패턴 인식, 자료 분석을 위한 지도 학습 모델로, 주로 분류와 회귀 분석을 위해 사용된다. SVM 데이터를 높은 차원 공간으로 매팅하고, 이 공간에서 초평면을 찾아내어 원래의 차원에서 복원한다. 이 기법은 분류, 회귀 뿐만 아니라 이상치 탐지 등 다양한 문제에도 적용될 수 있으며, 그 성능은 커널 선택과 하이퍼 파라미터 설정에 크게 영향을 받는다. SVM의 한 가지 단점은 학습 데이터가 많아 질수록 계산 복잡도가 높아져 처리 시간이 오래 걸린다는 점이다.

Algorithm 2: Support Vector Machine

- Input:** Datasets, Model Parameters
- Output:** Accuracy Score, Confusion Matrix, Classification result
1. Generate blobs of data ($X, y, \text{centers}$)
 2. Split Data ($X, y \rightarrow (X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}})$)
 3. Data Preprocessing \rightarrow Normalization or Standardization
 4. Hyperparameter (Dict) for Support Vector Machine
 5. KFold for Cross validation
 6. Grid Search \leftarrow sklearn GridSearchCV
 7. Classifier \leftarrow Best estimator and parameter
 8. Classifier.fit(Training Data)
 9. Classification report, accuracy score, and confusion matrix
 10. Validation using y_{predict}
-

Table 3. Pseudo code for Support Vector Machine

3.3 Unsupervised Learning algorithm

비지도 학습 알고리즘으로는 PCA와 t-SNE를 선택하였다. 두 알고리즘 모두 차원 축소 기법으로 데이터의 차원 축소와 시각화에 사용되는 효과적인 도구이다.

PCA와 t-SNE를 특히 선정한 이유는 PCA와 t-SNE는 모두 데이터의 구조와 패턴을 시각화 하여 구조적 이해에 도움을 주고, 데이터 포인트들의 유사성을 시각적으로 파악할 수 있으며, 클러스터 간의 분리도 확인할 수 있기 때문이다.

여기서 PCA는 데이터의 분산에 민감하여 ‘StandardScale’ 함수를 통해 정규화 해주었다. t-SNE는 scaling에 민감하지 않지만 PCA와 비교를 해보기 위해 똑같이 ‘StandardScale’ 함수를 이용해주었다.

시각화 된 자료로만 검증 시 주관적인 판단이 들어갈 것이라 생각해 추가적으로 두 차원 축소 알고리즘을 적용한 후 K-means로 군집 분석 후 실루엣 검증을 해주었다. K-means를 사용한 이유는 차원 축소 알고리즘의 경우 군집화가 목적이 아니므로 시각화 된 자료로만 분류의 정도를 확인할 수 있기에, 군집 분석 후 실루엣 검증을 해준다면 어느정도 정량적인 지표를 구할 수 있기 때문이다. 그렇게 되면 두 알고리즘의 비교 분석이 가능해진다.

또한, PCA, t-SNE, K-means 비지도학습 모두 하이퍼 파라미터 조정에 큰 영향을 받지 않아 random_state만 100으로 설정해주고 축소하는 차원을 2라고 지정만 해주었다. 데이터를 나누지 않았는데, 데이터의 전체 구조를 고려하기에는 데이터를 일부만 사용하면 전체 구조를 반영하지 못할 수 있기 때문이다. 데이터를 나누면 각 부분에서 얻은 차원 축소 결과를 비교하거나 시각화하기가 어려워질 수 있어 데이터 전체를 이용했다.

3.3.1 Principal Component Analysis

PCA는 데이터의 주성분을 찾아내어 데이터를 새로운 저차원의 공간으로 투영하는 기법이다. 데이터를 가장 잘 설명하는 주요한 변수들, 즉 주성분들을 추출하여 데이터 차원을 축소한다. 주성분은 데이터의 분산을 최대화하는 방향으로 정의되며, 이를 통해 데이터를 가장 잘 구분하는 특성들을 확인할 수 있다. PCA는 데이터의 분포를 가장 잘 표현할 수 있는 방향을 찾아내므로, 클러스터 데이터의 분류와 시각화에 유용하다.

Algorithm 3: Principal Component Analysis

Input: Datasets

Output: Silhouette Score, Classification result

1. Generate blobs of data ($X, y, centers$)
 2. Data Preprocessing → Standardization
 3. PCA.fit(X)
 4. K-means.fit(X_{pca})
-

5. Classification result visualization

6. print Silhouette score

Table 4. Pseudo code for Principal Component Analysis

3.3.2 t-SNE

t-SNE는 고차원 데이터를 저차원의 공간으로 매핑하여 데이터 간의 유사성을 보존하는 방식으로 시각화 한다. 특히, t-SNE는 데이터 포인트 간의 국부적인 구조를 보존하는데 초점을 둔다. t-SNE는 각 데이터 포인트를 저차원 공간에서 확률 분포로 표현하고, 고차원 공간에서의 이웃관계와 저차원 공간에서의 이웃관계 사이의 유사성을 최대한 보존하는 매핑을 수행한다. 이를 통해 데이터의 군집 구조를 시각적으로 파악할 수 있다.

Algorithm 4: t-SNE

Input: Datasets

Output: Silhouette Score, Classification result

1. Generate blobs of data (X, y, centers)
 2. Data Preprocessing → Standardization
 3. t-SNE.fit(X)
 4. K-means.fit(X_tsne)
 5. Classification result visualization
 6. print Silhouette score
-

Table 5. Pseudo code for t-SNE

4. 결론

기계학습에서 지도 학습 모델의 성능을 측정하는 방법 중 하나로 Confusion Matrix를 사용하였다. 이 행렬은 True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FN), 총 4개의 요소로 구성되어 있다. TP는 모델이 Positive를 잘 예측하여, 실제 값과 예측 값이 모두 Positive인 경우를 말한다. TN의 경우 Negative를 잘 예측하여, 실제 값과 예측 값 모두가 Negative인 경우를 말한다. 반면에, 실제 값은 Negative인데, 모델이 Positive라고 예측한 경우 이를 FP 혹은 Type I 에러라고 한다. 마찬가지로, 실제 값은 Positive인데 모델이 Negative라고 예측한 경우 이를 Type II 에러라고 말한다. Confusion matrix를 통해 분류 모델의 성능을 정확도, 정밀도, 재현율, F1 스코어 등 여러 가지 지표를 통해 평가할 수 있다. 결과는 ‘classification_report’ 함수를 활용하여 정밀도(Precision), 재현율(Recall), F1 Score 그리고 Confusion Matrix를 통해 나타냈다. 또한 실제 값과 예측 값을 동시에 시각화 하여 시각적 비교도 수행해 보았다. Table 6부터 Table 23까지는 ‘cluster_std’, ‘n_samples’,

'centers' 값을 각각 1,2,3 / 100,500,1000 / 2,5,10 으로 수행했을 때의 점수들을 보여준다.

비지도학습은 모델 성능을 검증하는 방법으로 시각화와 실루엣 검증을 활용하였다. 지도 학습과 동일하게 Confusion Matrix를 사용하지 않은 이유는 비지도 학습이 입력데이터의 구조나 패턴을 파악하는 것으로 목표로 하기 때문에 정답 레이블이 없기 때문이다. 그렇기에 정량적인 검증을 하는 것 자체가 모 험적인데, 따라서 본 연구에서는 군집화 결과를 시각화하여 직관적이며 주관적 평가를 주로 한다. 주관적인 평가와 더불어 실루엣 검증을 진행하였는데, 실루엣 검증이란 데이터 포인트가 같은 군집 내의 다른 데이터 포인트와 비교하여 군집 내의 응집력을 측정하고 다른 군집의 데이터 포인트와 비교하여 군집 간의 분리도 측정한다. 실루엣 계수는 응집력과 분리도를 종합적으로 고려하여 개별 데이터 포인트의 실루엣 값을 평균한 값으로 -1부터 1까지의 범위를 가지며 1에 가까울수록 군집화가 잘 되었음을 의미한다. 실루엣 계수는 원래 -1에서 1 사이의 값을 가지나 scikit-learn 모듈에서는 0에서 1의 값을 가지며, 1에 가까울수록 응집력이 높고 분리도가 높은 좋은 클러스터링 결과를 의미한다. 대체로 0.5보다 크면 클러스터링이 적절하다고 평가한다.

4.1 Logistic Regression Results

해당 변수들이 커질수록 분산, 샘플 수, 클러스터의 수도 커지므로, 로지스틱 회귀 모델을 사용했을 때의 정확도(Accuracy Score) 역시 조금씩 작아지는 것을 확인할 수 있었다. 또한 Confusion matrix를 통해 각 파라미터 값이 커질수록 대각 성분의 불균형을 통해 특정 클러스터에 대해서는 정확한 분류를 수행하지 못하다는 결론을 도출할 수 있었다. 현재는 하나의 파라미터를 다룰 때는 다른 파라미터들은 Default 값으로 설정했지만, 기타 변수들 역시 동시에 조정한다면 정확도 점수가 더 낮아질 수 있음을 확인했다. 지도 학습 모델들의 하이퍼 파라미터 튜닝에 대한 최적의 파라미터들은 각 Table의 'Best_param'을 통해 확인할 수 있다. 전체적으로, 다른 파라미터들보다 분산(std) 값이 클수록 분류에 대한 어려움이 많이 발생한다는 것을 알 수 있었다. 그 다음으로는 클러스터의 수가 성능에 영향을 끼친다는 것을 확인할 수 있었다. 샘플의 수는 큰 영향을 끼치지 않고 오히려 샘플이 1000개 였을 때 성능이 향상된 값을 볼 수 있었다.

4.1.1 Parameter: 'cluster_std'

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	23
1	0.96	0.96	0.96	23
2	1.00	1.00	1.00	17
3	0.95	0.95	0.95	20
4	1.00	1.00	1.00	17
Accuracy	0.98			100

Best_param	{'C': 0.1, 'penalty': 'l2'}
------------	-----------------------------

Table 6. Classification report for Logistic Regression, 'cluster_std':1

Index \ Score	Precision	Recall	F1 Score	Support
0	0.96	1.00	0.98	23
1	0.91	0.87	0.89	23
2	0.94	1.00	0.97	17
3	0.86	0.90	0.88	20
4	1.00	0.88	0.94	17
Accuracy	0.93			100
Best_param	{'C': 0.1, 'penalty': 'l2'}			

Table 7. Classification report for Logistic Regression, 'cluster_std':2

Index \ Score	Precision	Recall	F1 Score	Support
0	0.91	0.87	0.89	23
1	0.61	0.74	0.67	23
2	0.85	1.00	0.92	17
3	0.75	0.60	0.67	20
4	0.86	0.71	0.77	17
Accuracy	0.78			100
Best_param	{'C': 10, 'penalty': 'l2'}			

Table 8. Classification report for Logistic Regression, 'cluster_std':3

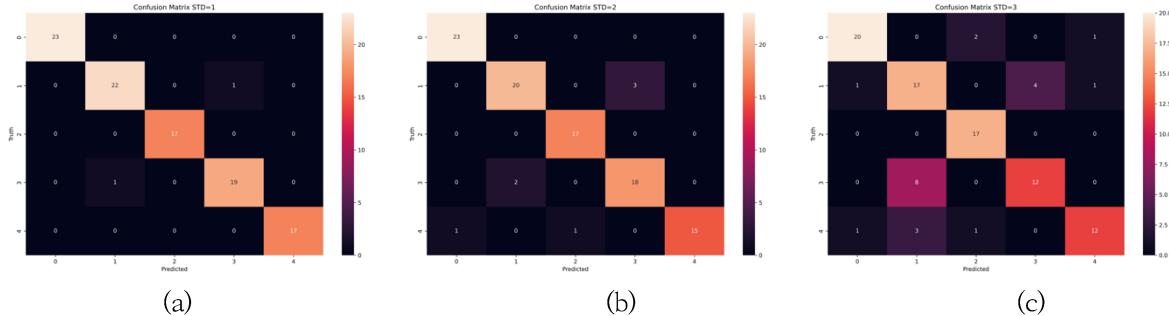


Fig 4. Confusion Matrix for Logistic Regression, 'cluster_std'

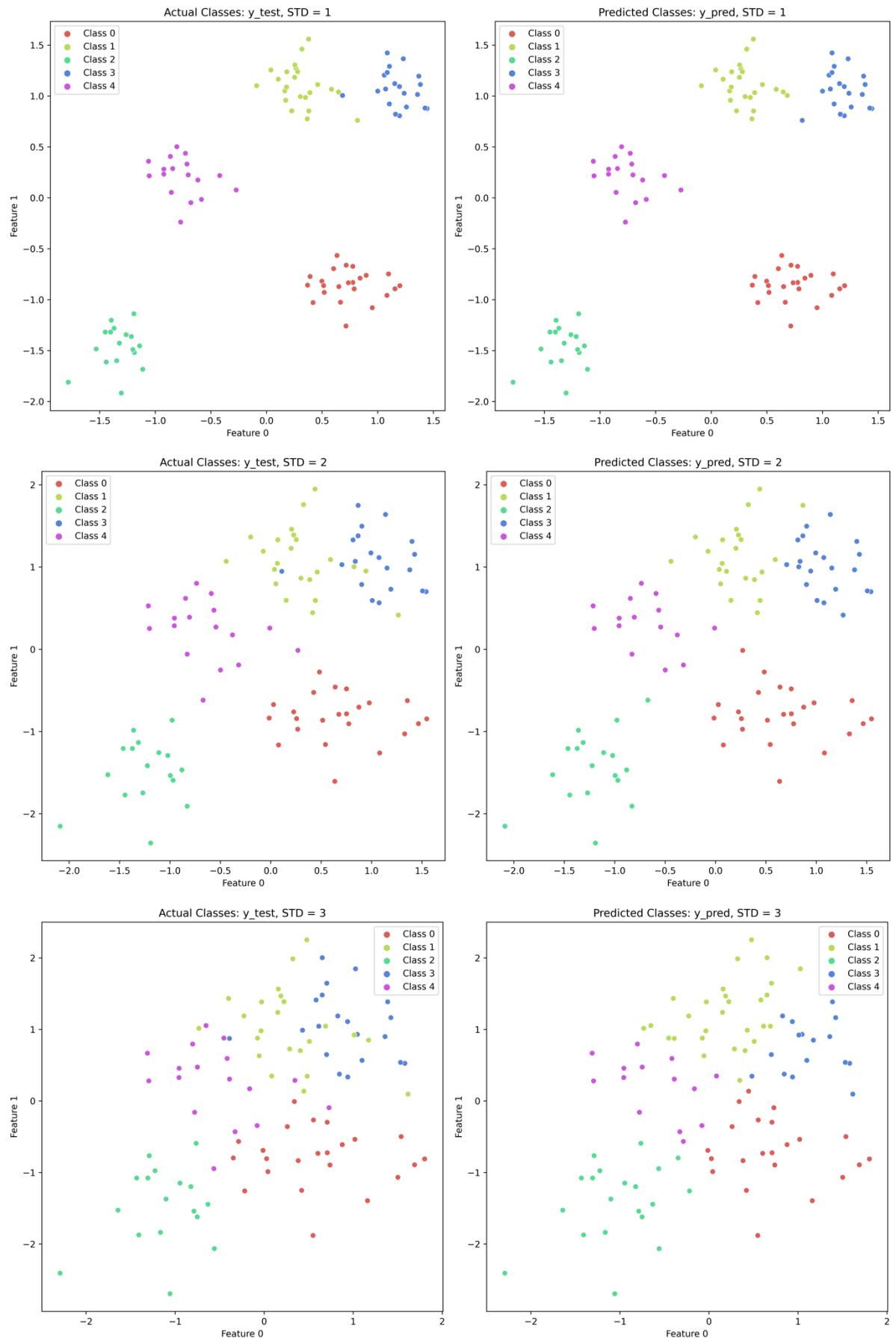


Fig 5. Classification result Logistic Regression, ‘cluster_std’: 1,2,3

4.1.2 Parameter: 'n_samples'

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	04
1	1.00	1.00	1.00	03
2	1.00	1.00	1.00	02
3	1.00	1.00	1.00	05
4	1.00	1.00	1.00	06
Accuracy		1.00		20
Best_param		{'C': 1, 'penalty': 'l2'}		

Table 9. Classification report for Logistic Regression, 'n_samples':100

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	23
1	0.96	0.96	0.96	23
2	1.00	1.00	1.00	17
3	0.95	0.95	0.95	20
4	1.00	1.00	1.00	17
Accuracy		0.98		100
Best_param		{'C': 0.1, 'penalty': 'l2'}		

Table 10. Classification report for Logistic Regression, 'n_samples':500

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	41
1	0.97	0.94	0.96	36
2	1.00	1.00	1.00	35
3	0.95	0.98	0.96	42
4	1.00	1.00	1.00	46
Accuracy		0.985		200
Best_param		{'C': 1000, 'penalty': 'l2'}		

Table 11. Classification report for Logistic Regression, 'n_samples':1000

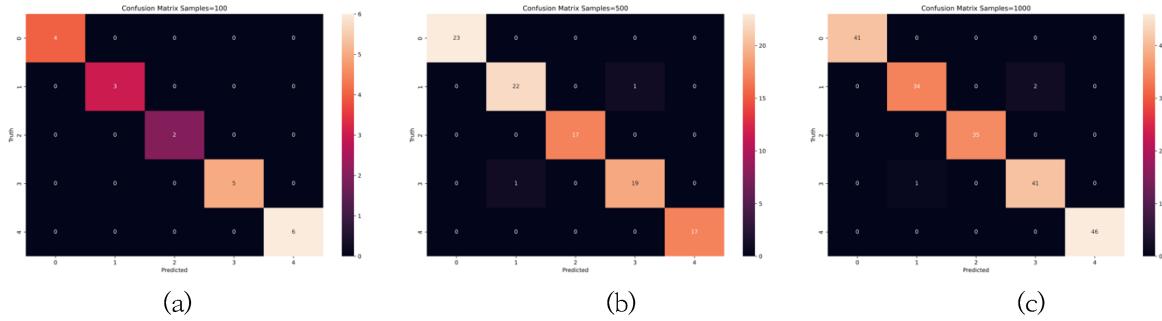
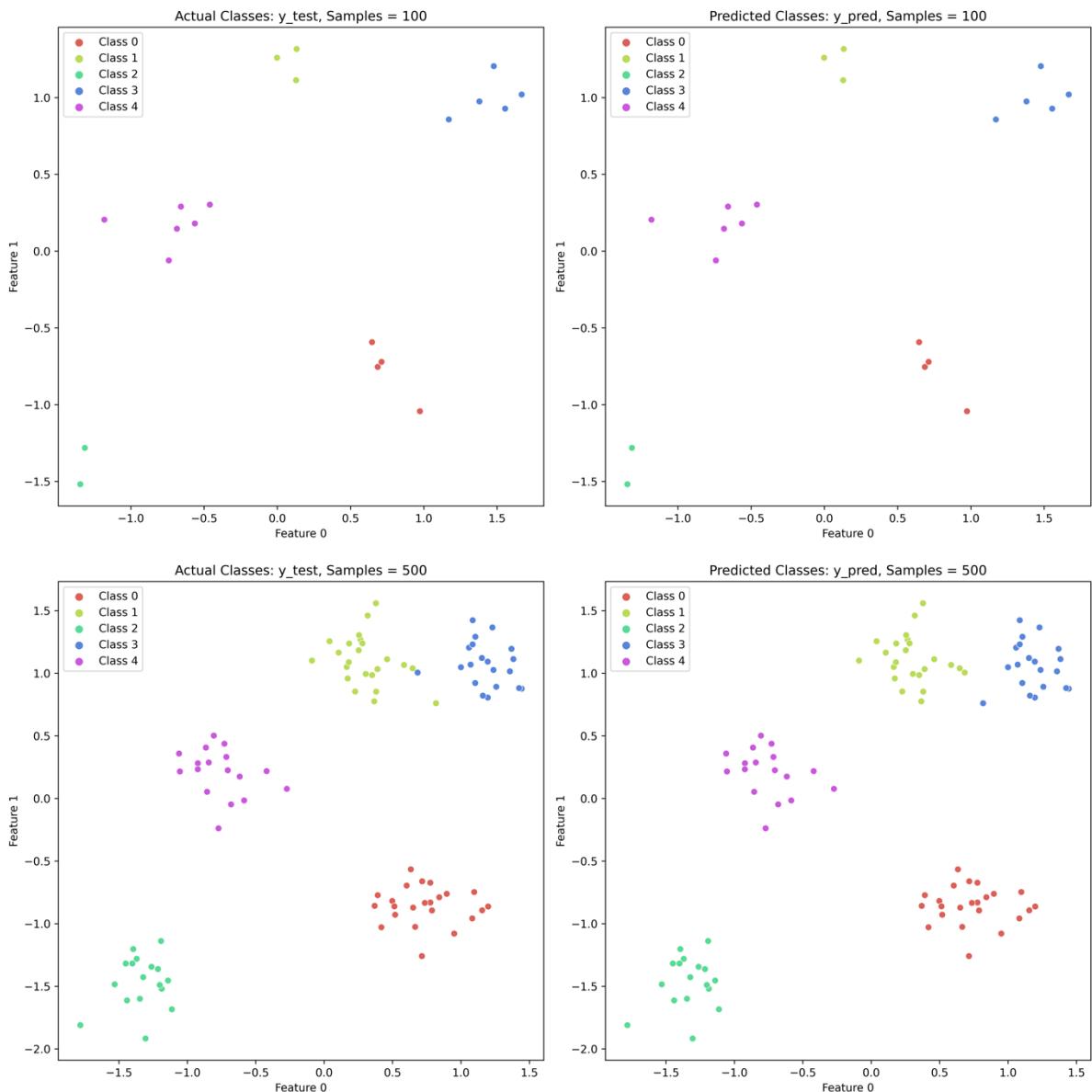


Fig 6. Confusion Matrix for Logistic Regression, 'n_samples'



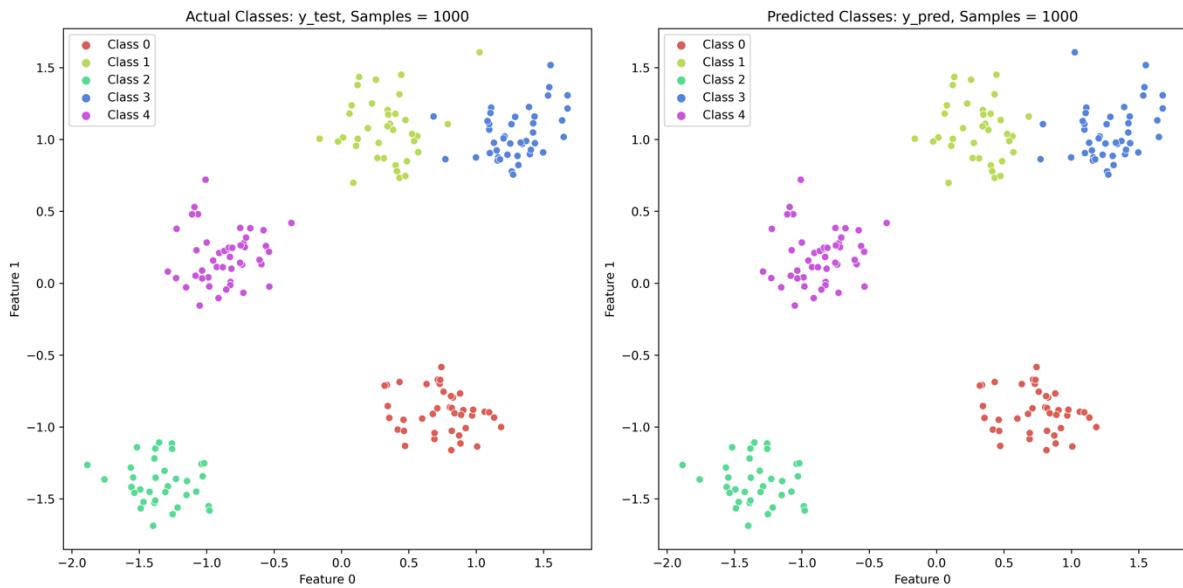


Fig 7. Classification result Logistic Regression, ‘n_samples’: 100,500,1000

4.1.3 Parameter: ‘centers’

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	49
1	1.00	1.00	1.00	51
Accuracy	1.00			100
Best_param	{'C': 0.01, 'penalty': 'l2'}			

Table 12. Classification report for Logistic Regression, ‘centers’:2

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	23
1	0.96	0.96	0.96	23
2	1.00	1.00	1.00	17
3	0.95	0.95	0.95	20
4	1.00	1.00	1.00	17
Accuracy	0.98			100
Best_param	{'C': 0.1, 'penalty': 'l2'}			

Table 13. Classification report for Logistic Regression, ‘centers’:5

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	11

1	1.00	1.00	1.00	06
2	1.00	1.00	1.00	06
3	1.00	1.00	1.00	12
4	1.00	1.00	1.00	17
5	0.30	0.38	0.33	08
6	1.00	1.00	1.00	12
7	1.00	1.00	1.00	09
8	0.71	1.00	0.83	05
9	0.70	0.50	0.58	14
Accuracy	0.88			100
Best_param	{'C': 100, 'penalty': 'l2'}			

Table 14. Classification report for Logistic Regression, ‘centers’:10

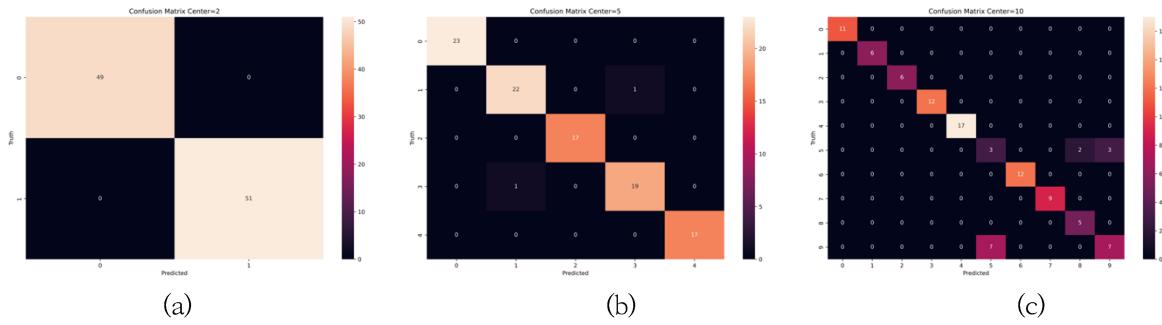
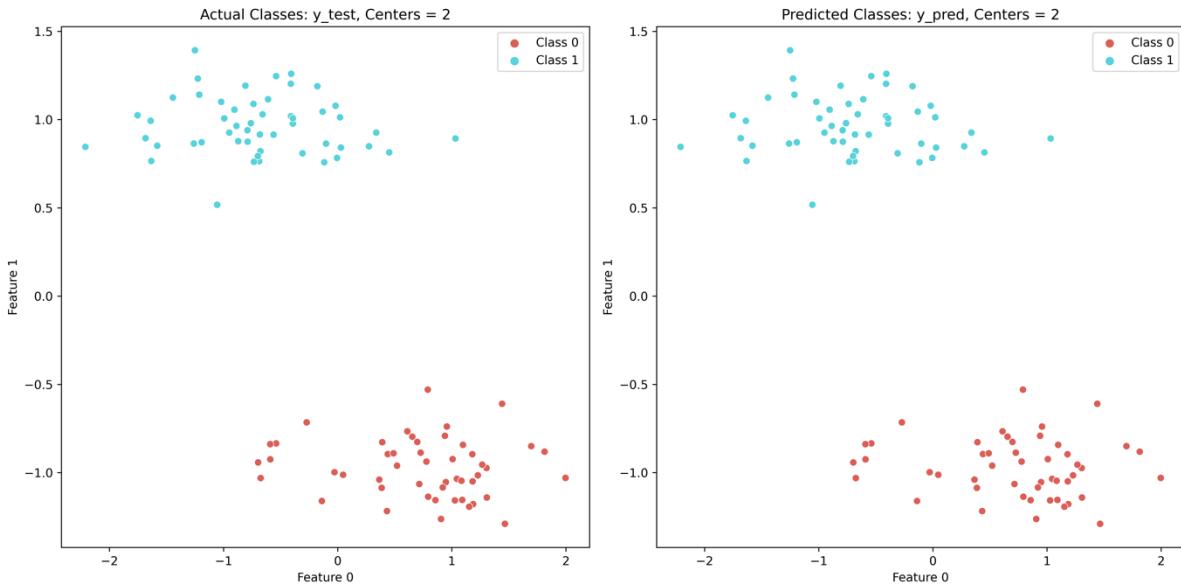


Fig 8. Confusion Matrix for Logistic Regression, ‘centers’



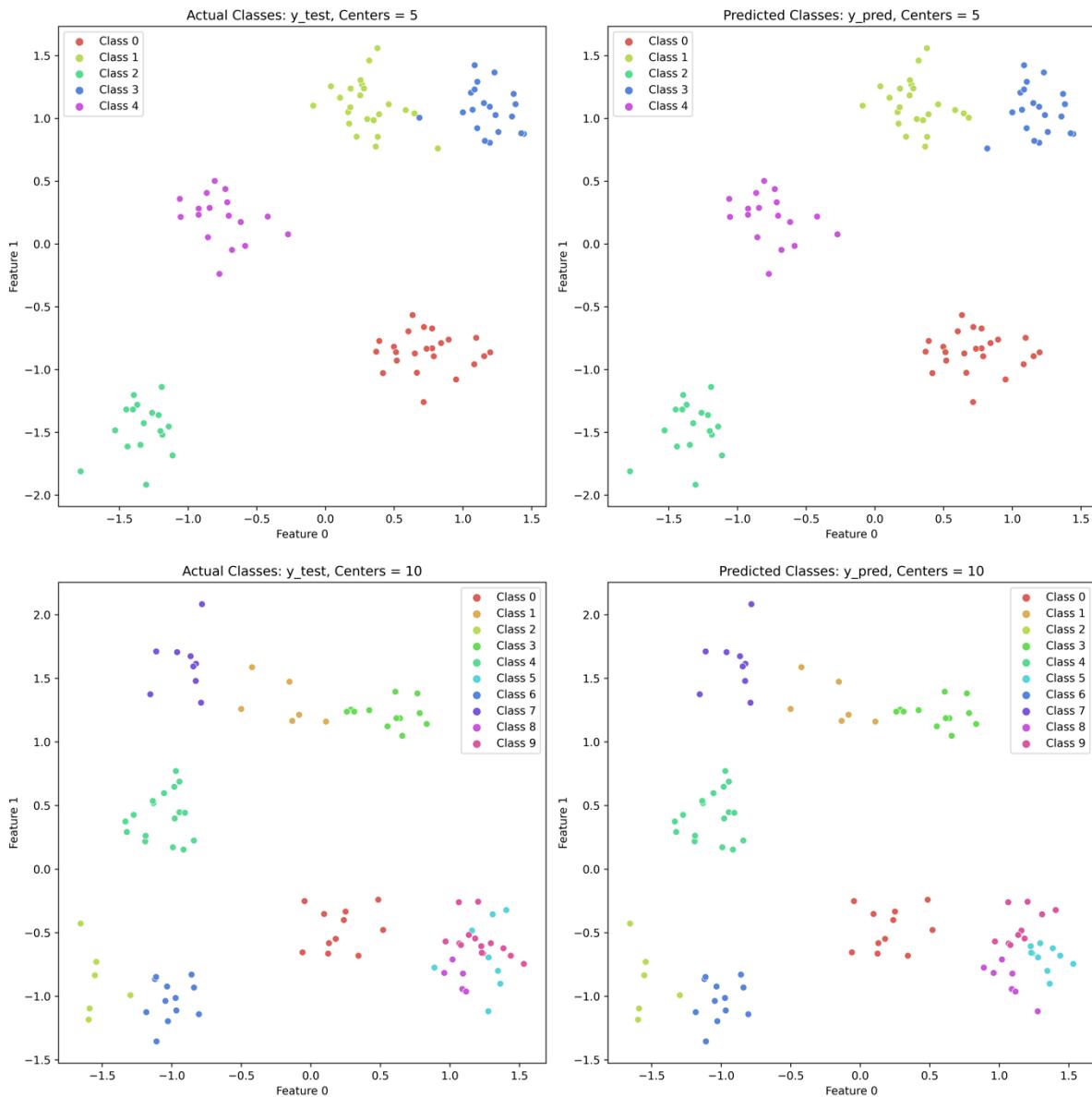


Fig 9. Classification result Logistic Regression, ‘centers’: 2,5,10

4.2 SVM Results

SVM을 활용했을 때 로지스틱 회귀보다 ‘cluster_std’ 3값에 대한 분류 성능이 조금 더 좋아짐을 볼 수 있었다. 하지만, ‘centers’ 변수와 ‘n_samples’ 변수에 대해서는 로지스틱 회귀 모델 성능이 SVM보다 뛰어났다. 하이퍼 파라미터 튜닝 이전보다 이후로 적용했을 때 향상된 정확도 점수를 얻을 수 있었다. 모델마다 하이퍼 파라미터들이 다르고, 영향을 끼치는 요인들이 달라서 이를 잘 확인하는 동시에 max_iteration까지 고려하게 되면 최적의 파라미터 값을 얻을 수 있다. 아래의 표들은 SVM을 활용했을 때의 정확도, 정밀도, 재현율, F1 Score, Confusion matrix에 대한 결과 값이다. 또한, 실제 값과 예측 값을 시각화 하여 구별하기 쉽게 해놓았다.

4.2.1. Parameter: ‘cluster_std’

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	23
1	0.96	0.96	0.96	23
2	1.00	1.00	1.00	17
3	0.95	0.95	0.95	20
4	1.00	1.00	1.00	17
Accuracy	0.98			100
Best_param	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}			

Table 15. Classification report for SVM, ‘cluster_std’:1

Index \ Score	Precision	Recall	F1 Score	Support
0	0.96	1.00	0.98	23
1	0.83	0.87	0.85	23
2	1.00	1.00	1.00	17
3	0.84	0.80	0.82	20
4	1.00	0.94	0.97	17
Accuracy	0.92			100
Best_param	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}			

Table 16. Classification report for SVM, ‘cluster_std’:2

Index \ Score	Precision	Recall	F1 Score	Support
0	0.92	0.96	0.94	23
1	0.64	0.78	0.71	23
2	0.94	1.00	0.97	17
3	0.76	0.65	0.70	20
4	0.92	0.71	0.80	17
Accuracy	0.82			100
Best_param	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}			

Table 17. Classification report for SVM, ‘cluster_std’:3

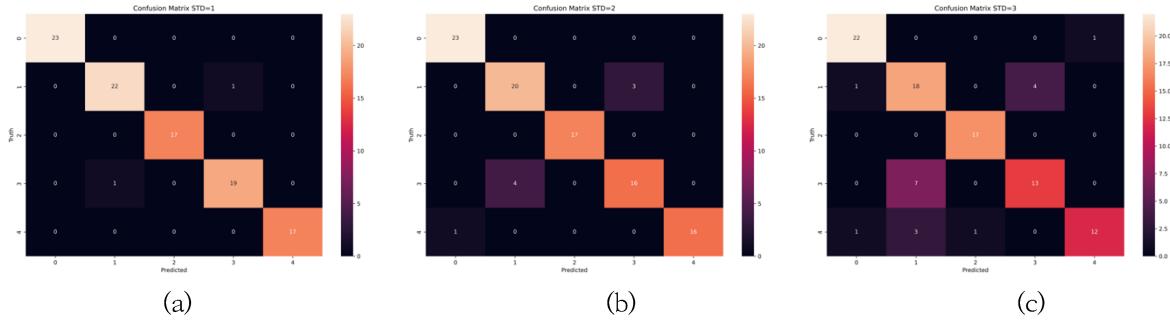
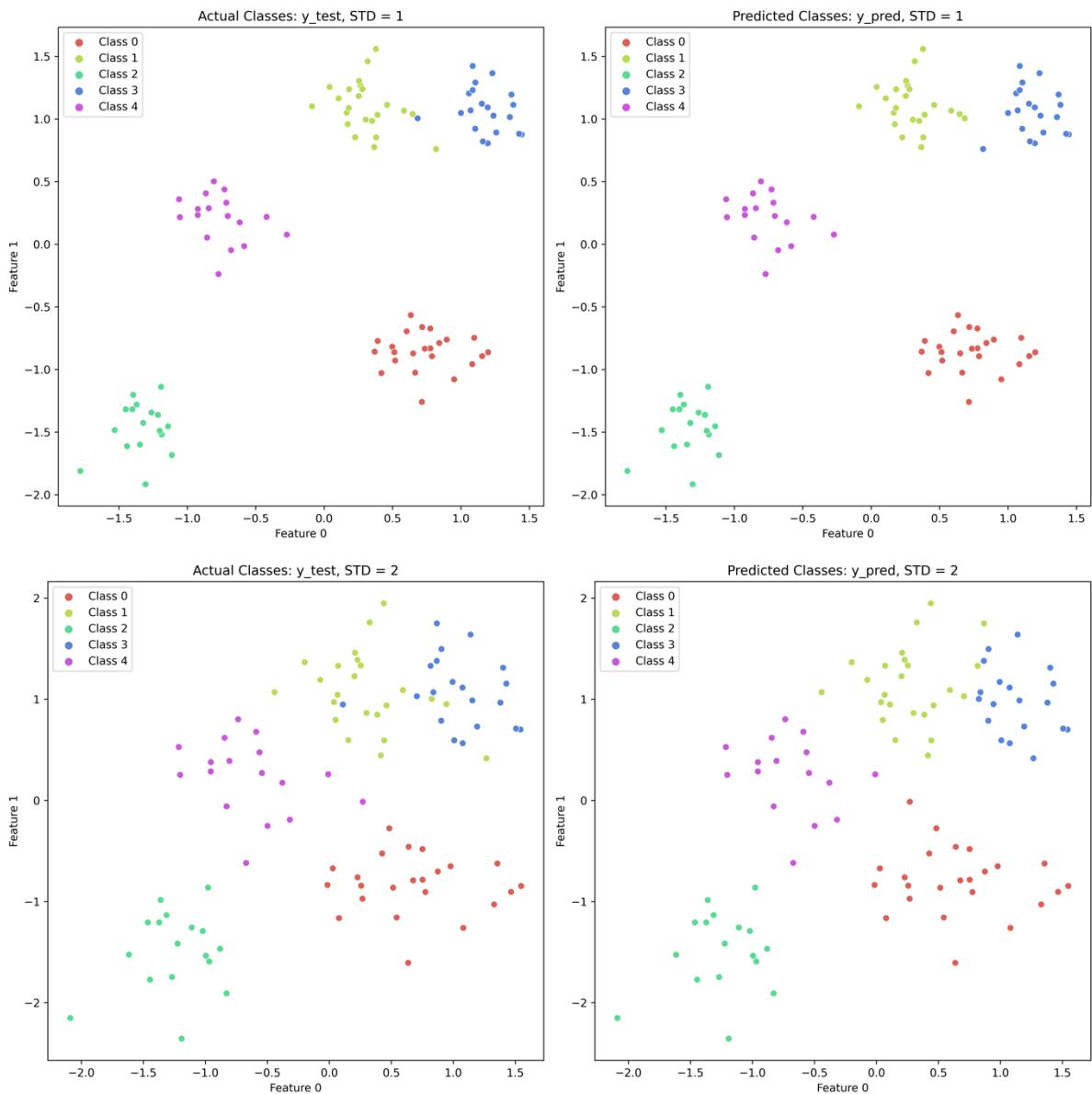


Fig 10. Confusion Matrix for Logistic Regression, 'cluster_std'



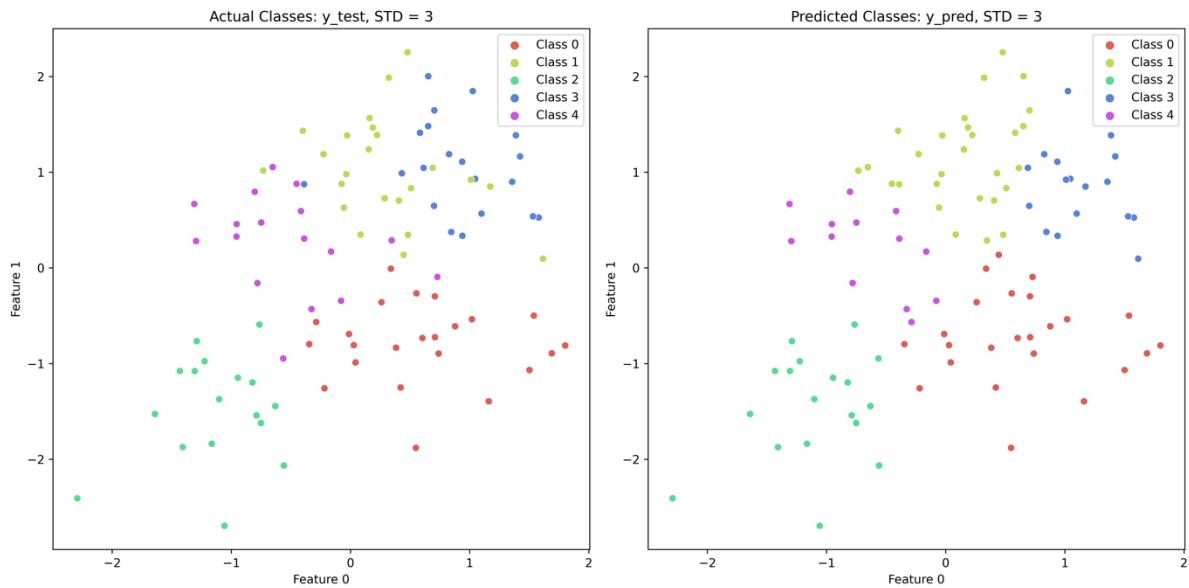


Fig 11. Classification result Logistic Regression, ‘cluster_std’: 1,2,3

4.2.2. Parameter: ‘n_samples’

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	04
1	1.00	1.00	1.00	03
2	1.00	1.00	1.00	02
3	1.00	1.00	1.00	05
4	1.00	1.00	1.00	06
Accuracy	1.00			20
Best_param	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}			

Table 18. Classification report for SVM, ‘n_samples’:100

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	23
1	0.96	0.96	0.96	23
2	1.00	1.00	1.00	17
3	0.95	0.95	0.95	20
4	1.00	1.00	1.00	17
Accuracy	0.98			100
Best_param	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}			

Table 19. Classification report for SVM, ‘n_samples’:500

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	41
1	1.00	0.94	0.97	36
2	1.00	1.00	1.00	35
3	0.95	1.00	0.98	42
4	1.00	1.00	1.00	46
Accuracy	0.82			200
Best_param	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}			

Table 20. Classification report for SVM, 'n_samples':1000

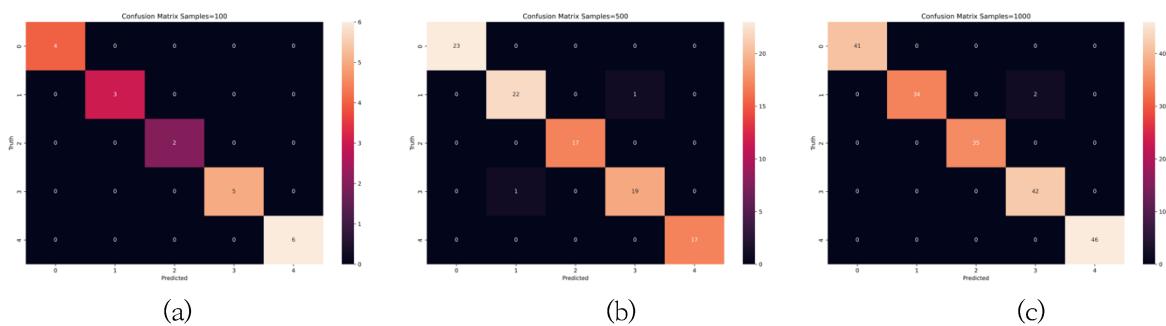
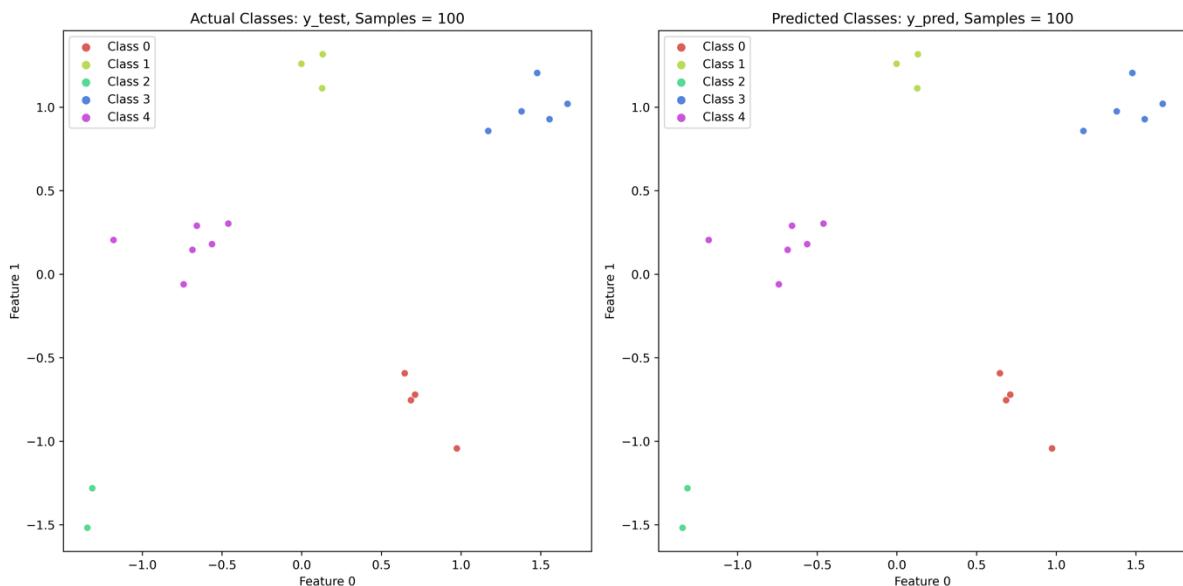


Fig 12. Confusion Matrix for Logistic Regression, 'n_samples'



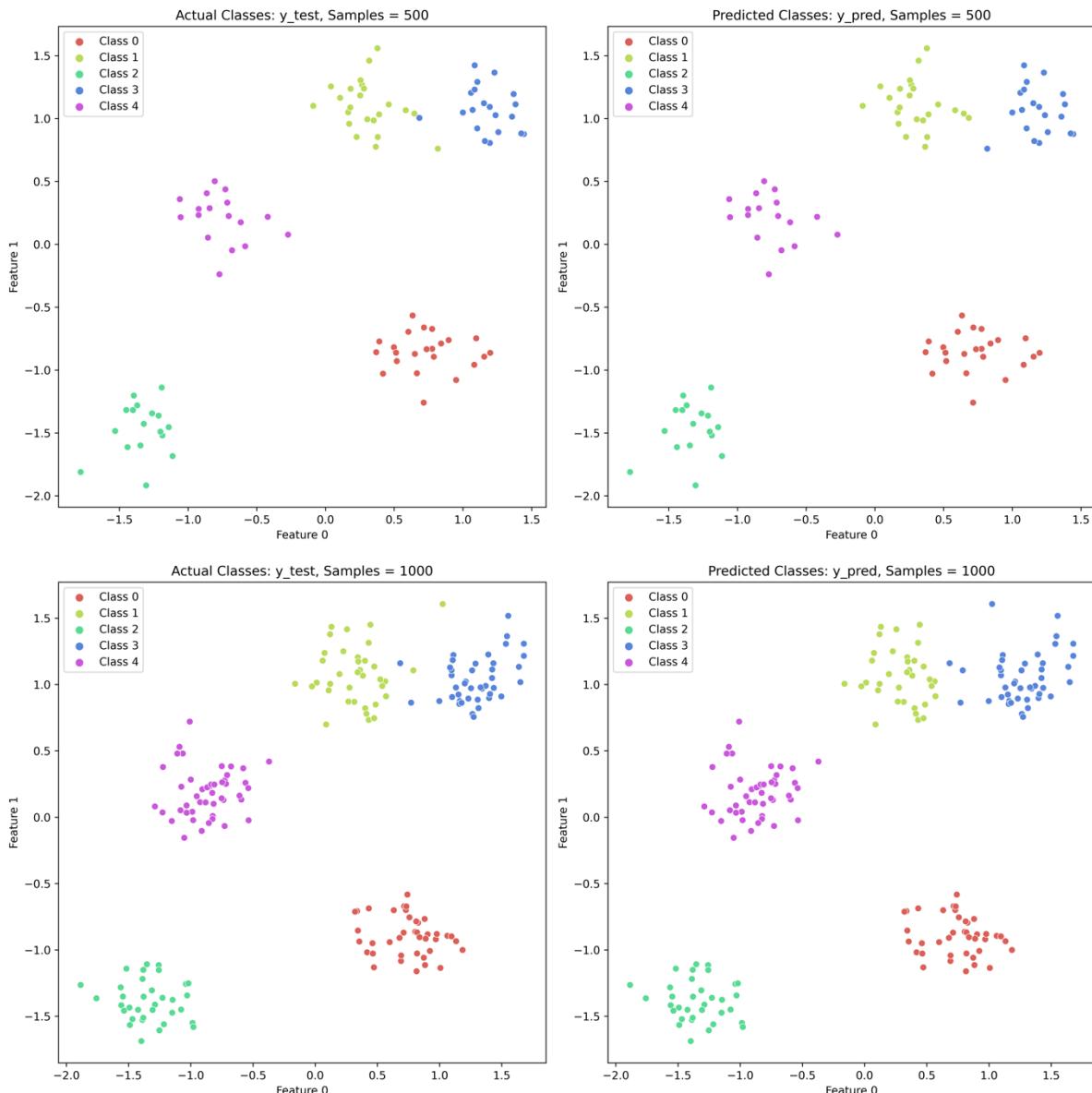


Fig 13. Classification result Logistic Regression, ‘n_samples’: 100,500,1000

4.2.3. Parameter: ‘centers’

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	49
1	1.00	1.00	1.00	51
Accuracy	1.00			100
Best_param	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}			

Table 21. Classification report for SVM, ‘centers’:2

Index \ Score	Precision	Recall	F1 Score	Support
---------------	-----------	--------	----------	---------

0	1.00	1.00	1.00	23
1	0.96	0.96	0.96	23
2	1.00	1.00	1.00	17
3	0.95	0.95	0.95	20
4	1.00	1.00	1.00	17
Accuracy	0.98			100
Best_param	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}			

Table 22. Classification report for SVM, ‘centers’:5

Index \ Score	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	11
1	1.00	0.83	0.91	06
2	1.00	1.00	1.00	06
3	1.00	1.00	1.00	12
4	1.00	1.00	1.00	17
5	0.22	0.25	0.24	08
6	1.00	1.00	1.00	12
7	0.90	1.00	0.95	09
8	0.62	1.00	0.77	05
9	0.70	0.50	0.58	14
Accuracy	0.86			100
Best_param	{'C': 100, 'gamma': 1, 'kernel': 'rbf'}			

Table 23. Classification report for SVM, ‘centers’:10

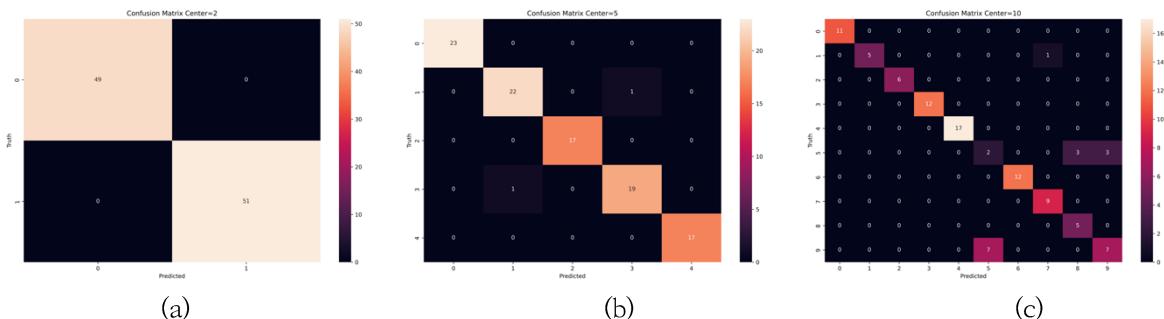


Fig 14. Confusion Matrix for Logistic Regression, ‘centers’

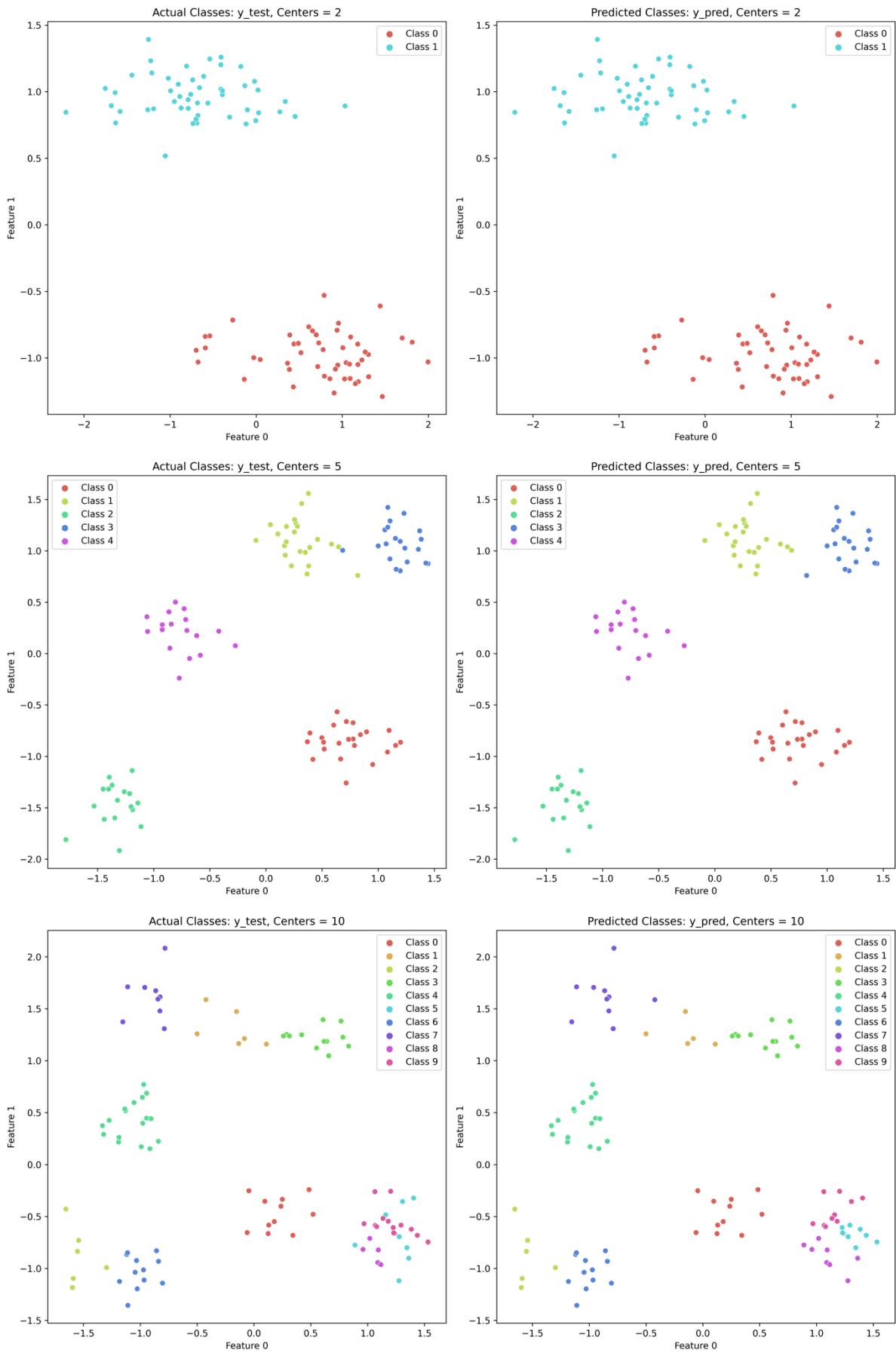


Fig 15. Classification result Logistic Regression, ‘centers’: 2,5,10

4.3 PCA Results

4.3.1 Parameter: ‘cluster_std’

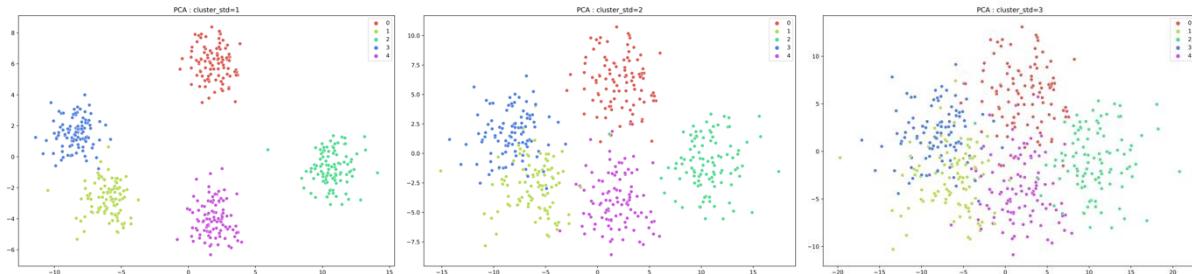


Fig 16. Classification result PCA, ‘cluster_std’: 1,2,3

첫번째로 cluster_std를 1, 2, 3으로 주었을 때의 변화를 살펴보려고 한다. 값이 커질수록 중심값으로부터 데이터가 퍼져 생성되기 때문에 클러스터끼리 섞여 분류 작업이 잘 이루어지지 않았음을 시각화 자료를 통해 볼 수 있다. 1일 땐, 클러스터끼리 거의 안 섞인 상태로 잘 분류되어 있는 듯 보이지만 점점 구분이 힘들 정도로 섞였음을 도출할 수 있었다. 객관적인 판단을 위해 실루엣 계수를 도출하였다. 다음이 도출된 실루엣 검증 값이다.

cluster_std	1	2	3
silhouette score	0.73	0.51	0.38

Table 24. ‘cluster_std’ dataset: silhouette score

시각화 된 자료와 실루엣 계수가 거의 일치한다는 것을 알 수 있다. cluster_std 값이 1에 가까울수록 분류가 잘되었고, 커질수록 분류가 제대로 되지 못해 실루엣 계수 값이 낮아짐을 확인할 수 있었다. 여기서 cluster_std가 2일 때까지만 해도 0.5 이상으로 적절함을 알 수 있었는데 그 이상으로 높아질수록 PCA의 성능이 떨어진다는 결론을 도출할 수 있었다.

4.3.2 Parameter: ‘n_samples’

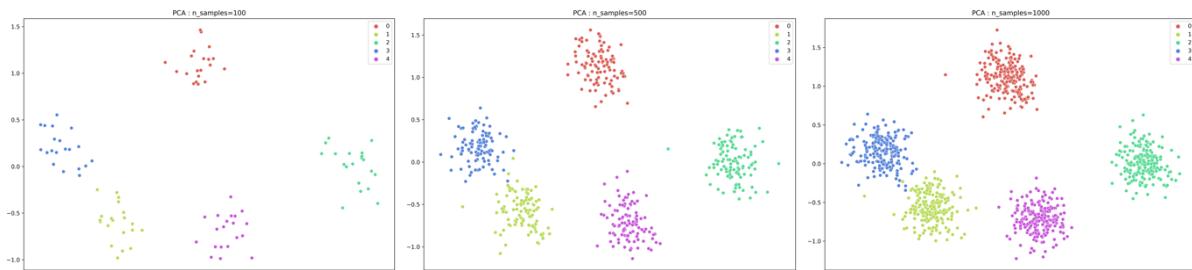


Fig 17. Classification result PCA, ‘n_samples’: 100,500,1000

두번째로 n_samples를 100, 500, 1000으로 주었을 때의 변화를 살펴보려고 한다. 값이 커지든 작아지든 분류가 대체적으로 잘 되었음을 시각화 자료를 통해 볼 수 있다. 연구 시작 전에는 n_samples에 따라 결과값에 큰 영향을 끼칠 것이라고 가정했던 것과는 다른 결과가 나왔음을 알 수 있었다. 객관적인 판단을 위해 실루엣 계수를 도출하였다. 다음이 도출된 실루엣 검증 값이다.

n_samples	100	500	1000
silhouette score	0.74	0.73	0.74

Table 25. ‘n_samples’ dataset: silhouette score

시각화 된 자료와 실루엣 계수가 거의 일치한다는 것을 알 수 있다. 모든 데이터 셋이 대체적으로 분류가 잘 되어 보였는데 실루엣 계수 또한 대체적으로 모두 잘 분류가 되었음을 확인할 수 있었다. 여기서 n_samples가 100일 때는 sample 수가 적어 응집도는 떨어지지만 분리도가 높아 0.74로, 1000일 때는 분리도는 낮지만 sample 수가 많다보니 응집도가 높아 0.74로 나왔다. 500개일 때는 둘의 중간이므로 0.01정도로 성능이 떨어졌지만 유의미한 차이는 아니라고 판단되어 n_samples는 PCA의 성능에 큰 영향을 미치지 않는다는 결론을 도출할 수 있었다.

4.3.3 Parameter: ‘centers’

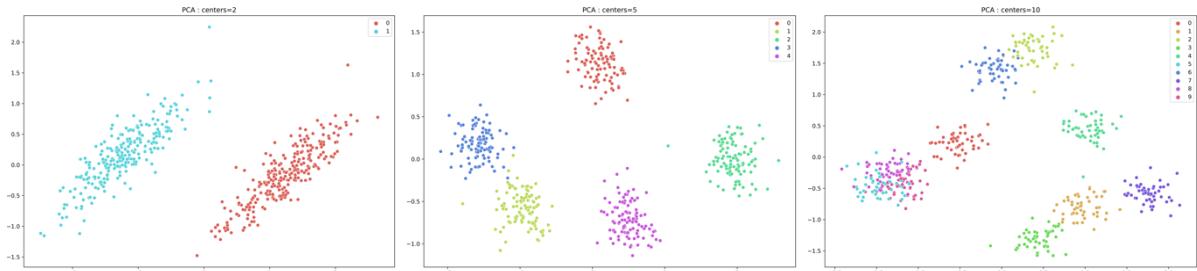


Fig 18. Classification result PCA, ‘centers’: 2,5,10

세번째로 centers를 2, 5, 10으로 주었을 때의 변화를 살펴보려고 한다. 값이 커질수록 분류가 잘 안 되었음을 시각화 자료를 통해 볼 수 있다. center가 2인 데이터 셋은 이진 분류이기 때문에 명확하게 구분되고 있고, 5인 데이터셋은 잘 분류된 듯 하지만 클러스터끼리 거리가 가까운 듯 보이고, 10인 데이터 셋은 클러스터가 섞이기까지 하면서 분류가 잘 수행되지 않았음을 알 수 있었다. 객관적인 판단을 위해 실루엣 계수를 도출하였다. 다음이 도출된 실루엣 검증 값이다.

centers	2	5	10
silhouette score	0.69	0.73	0.49

Table 26. ‘centers’ dataset: silhouette score

시각화 된 자료와 실루엣 계수가 center의 값이 2일때 일치하지 않는다는 것 알 수 있다. 값이 5, 10일 때는 시각화 자료와 실루엣 계수가 일치해 보이지만 2일 때는 클러스터의 응집도가 낮은 양옆으로 퍼져 있는 모양이라 실루엣 계수와 일치하지 않은 듯 보인다고 분석하였다. 이는 실루엣 계수의 단점이 차원축소 알고리즘을 사용할 시 시각화 된 자료로 분류에 대한 판단을 하는 가장 큰 이유라고 할 수 있다. 그럼에도 불구하고 0.5보다 큰 값인 0.69이 나와 클러스터링이 적절했음을 확인할 수 있다. 결론적으로 시각화 된 자료를 기준으로 판단했을 때 center의 수가 적어야 분류 작업이 가장 잘 수행되어 PCA의 성능이 향상될 수 있다는 결론을 도출할 수 있었다.

최종적으로 PCA 알고리즘을 사용할 때에는 데이터의 분산도가 낮고 center의 개수가 적은 데이터에 있어서 분류를 효과적으로 해내지만 종합적으로 약 0.73의 실루엣 계수를 나타내는 것으로 보아 분류에 있어 최적의 알고리즘이 아니라는 결론을 냈다.

4.4 t-SNE Results

4.4.1 Parameter: ‘cluster_std’

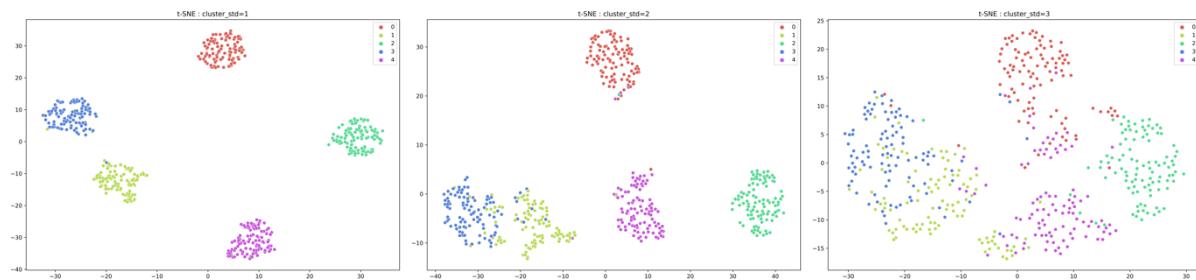


Fig 19. Classification result t-SNE, ‘cluster_std’: 1,2,3

첫번째로 cluster_std를 1, 2, 3으로 주었을 때의 변화를 살펴보려고 한다. PCA와 마찬가지로 값이 커질수록 분류 작업이 잘 이루어지지 않았음을 시각화 자료를 통해 볼 수 있었다. 1일 땐, 소수의 몇몇 데이터가 다른 클러스터에 잘못 위치해 있어 분류가 덜 되어있음에도 응집도와 다른 클러스터끼리 분리가 명확하게 되어 잘 분류되어 있는 듯 보이지만 점점 std 값이 커질수록 구분이 힘들 정도로 섞였음을 도출할 수 있었다. 객관적인 판단을 위해 실루엣 계수를 도출하였다. 다음이 도출된 실루엣 겹증 값이다.

cluster_std	1	2	3
silhouette score	0.82	0.69	0.49

Table 27. ‘cluster_std’ dataset: silhouette score

시각화 된 자료와 실루엣 계수가 거의 일치한다는 것을 확인할 수 있다. cluster_std 값이 1인 경우에 약간의 오분류로 0.82라는 1에 조금 못 미치는 값을 도출할 수 있었고, 2일 때는 그래도 0.5보다 큰 0.69라는 값이 나와 클러스터링이 적절했음을 알 수 있었다. 반면에 값이 3인 경우에는 0.5 이하인

0.49로 약간의 차이로 클러스터링이 적절하지 않았다고 판단하였다. 결론적으로 cluster_std가 3 이상으로 높아질수록 t-SNE의 성능이 떨어진다는 결론을 도출할 수 있었다.

4.4.2 Parameter: ‘n_samples’

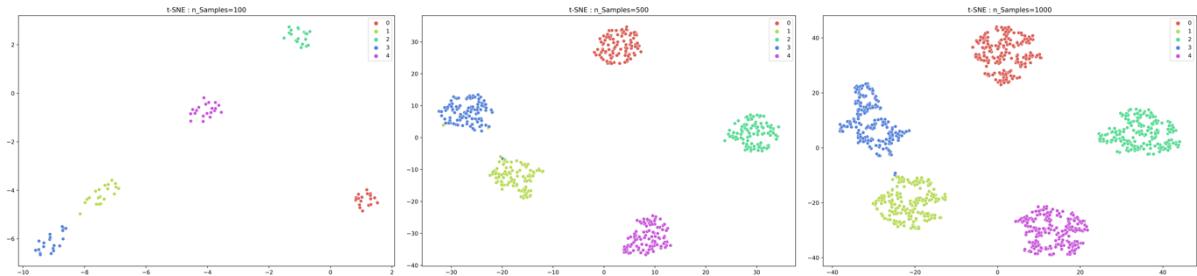


Fig 20. Classification result t-SNE, ‘n_samples’: 100,500,1000

두번째로 n_samples를 100, 500, 1000으로 주었을 때의 변화를 살펴보려고 한다. PCA와 마찬가지로 값이 커지든 작아지든 분류가 대체적으로 잘 되었음을 시각화 자료를 통해 볼 수 있다. 객관적인 판단을 위해 실루엣 계수를 도출하였다. 다음이 도출된 실루엣 검증 값이다.

n_samples	100	500	1000
silhouette score	0.84	0.82	0.74

Table 28. ‘n_samples’ dataset: silhouette score

시각화 된 자료와 실루엣 계수가 거의 일치한다는 것을 알 수 있다. 시각화 된 데이터를 분석하는데 있어 놓친 디테일을 확인할 수 있었는데, sample의 개수가 커질수록 약간씩 섞이는 데이터가 존재하고 응집도가 떨어진다는 것을 추가적으로 분석할 수 있었다. 그렇지만, 모든 실루엣 계수가 0.5 이상으로 적절한 클러스터링이었음을 확인할 수 있었고, sample의 개수가 적을수록 t-SNE의 성능이 향상될 수 있다는 결론을 도출할 수 있었다.

4.3.3 Parameter: ‘centers’

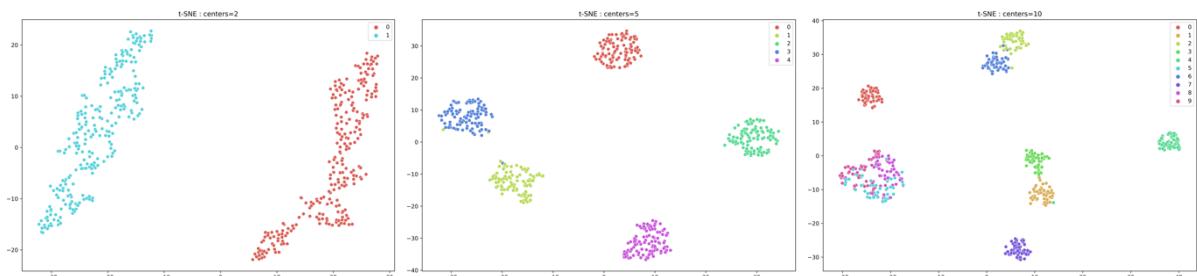


Fig 21. Classification result t-SNE, ‘centers’: 2,5,10

세번째로 centers를 2, 5, 10으로 주었을 때의 변화를 살펴보려고 한다. PCA와 마찬가지로 값이 커질수록 분류가 잘 안 되었음을 시각화 자료를 통해 볼 수 있다. 객관적인 판단을 위해 실루엣 계수를 도출하였다. 다음이 도출된 실루엣 점증 값이다.

centers	2	5	10
silhouette score	0.66	0.82	0.66

Table 29. ‘centers’ dataset: silhouette score

이번에도 시각화 된 자료와 실루엣 계수가 center의 값이 2일때 일치하지 않는다는 것 알 수 있다. 여전히 center 값이 2일 때 클러스터의 응집도가 낮은 양옆으로 펴져 있는 모양이라 실루엣 계수와 일치하지 않은 듯 보인다고 분석하였다. 전체적으로 봤을 때는 모두 0.5보다 큰 값인 0.66이 나와 클러스터링이 적절했음을 확인할 수 있다. 하지만 시각화 된 자료로 판단했을 때 center의 값이 2인 데이터가 가장 분류가 잘 되었다고 판단이 되므로 t-SNE의 성능이 향상될 수 있다는 결론을 도출할 수 있었다.

최종적으로 t-SNE 알고리즘을 사용할 때에는 데이터의 분산도가 낮고 center의 개수가 적으며 sample이 적은 데이터에 있어서 분류를 효과적으로 해냄을 알 수 있었고, 종합적으로 std의 값이 3일 때만 실루엣 계수가 0.5 이하이고 나머지 값이 높은 것으로 보아 대체적으로 분류에 있어 적절한 알고리즘이라는 결론을 냈다.

4.5 비교 분석

4.5.1 지도학습 모델 비교

Logistic Regression과 SVM 비교

Algorithm	Logistic Regression	SVM
Score(STD-3)	0.78	0.82
Score(Samples-1000)	0.985	0.82
Score(Centers-10)	0.88	0.86

Table 30. LR vs SVM : ‘cluster_std’-3, ‘n_samples’-1000, ‘centers’-10

각 파라미터들 중 최대값들의 정확도 점수를 Table 30을 통해 확인할 수 있다. 분산이 커질수록 서로 겹쳐지는 샘플들에 의해 구별하기가 어려워지는데, SVM이 로지스틱 회귀보다 조금 더 나은 결과를 보였다. 반면에 샘플 및 클러스터 수의 증가에 의한 분류 성능은 로지스틱 회귀가 SVM보다 더 훌륭한 결과를 보여주었다. 따라서 문제 상황에 맞는 적절한 모델을 사용하는 것이 바람직하다는 결론을 얻었다. 이와 더불어 더 많은 파라미터들의 상호작용 속에서 더 많은 경우의수가 나올 것으로 예상한다.

`make_blobs`를 통한 가상 데이터셋 생성을 통해 미리 원하는 파라미터들과 모델에 대입해 본 뒤에 작업을 수행한다면 향상된 결과를 얻을 수 있을 것이다.

4.5.2 비지도학습 모델 비교

PCA와 t-SNE 비교

Algorithm	PCA	t-SNE
silhouette score	0.73	0.82

Table 31. PCA vs t-SNE : Default dataset

모든 값이 Default일 때 PCA의 실루엣 계수는 0.73, t-SNE의 실루엣 계수는 0.82로 실루엣 계수만으로 판단했을 때 t-SNE의 성능이 더 좋음을 알 수 있다. 그 뿐만이 아니라 시각화 된 자료로 비교했을 때도 t-SNE의 클러스터 응집도가 높고 다른 클러스터와의 분리 정도도 높아 눈으로 봤을 때도 PCA 보다 분류가 잘 되었다고 판단하였다.

cluster_std	1	2	3
PCA silhouette score	0.73	0.51	0.38
t-SNE silhouette score	0.82	0.69	0.49

Table 32. PCA vs t-SNE : ‘cluster_std’ data set

추가적으로 실루엣 계수와 시각화 자료가 일치하면서 값의 변화가 크게 차이 난 std 변화 데이터 셋만 두고 봤을 때도 약 0.1이라는 큰 차이가 존재하며 시각화 된 자료로 봤을 때도 t-SNE가 좀 더 클러스터링 되었음을 확인할 수 있었다. 결론적으로 t-SNE가 PCA에 비해 `make_blobs` 가상 데이터 셋 분류에 보다 더 적합하다고 도출하였다.

또한, `make_blobs`의 경우 레이블(정답 데이터)이 존재하지 않는 비지도학습보다 지도학습에 대해 성능이 더 높게 나오는 것을 확인할 수 있다. 비지도학습에서의 성능을 향상시키기 위한 고민이 향후 연구에서 필요할 것 같다.

5. 향후 연구

Scikit-learn 패키지의 함수를 활용하여 가상 데이터셋을 생성해 보았다. 향후 연구에서는 실제 데이터에서의 적용을 해보고자 한다. 특히 최근 위성 영상에 대한 무료 접근이 가능해져, 방대한 양의 위성 빅데이터가 취득이 가능해짐에 따라 이를 활용한 기계학습을 수행해보고자 한다. 본 연구는, 실제 데이터를 바로 학습을 시키기 이전에 적절한 파라미터 값을 찾기 위한 과정을 잘 보여준다. 하지만 모든 변수들을 다루어 본 것이 아니기 때문에 ‘n_feature’과 같이 속성 값이 늘어났을 때 모델의 어떠한 영

향을 끼치는지에 대한 작업을 하려 한다. 현재 속성 값이 2일 때의 시각화는 가능했지만, 만약 이 값이 3 이상으로 갈수록 시각화를 어떠한 방법으로 할지에 대한 고민도 필요할 것 같다. 이와 더불어, 비지도 학습이 지도 학습에 비해 성능이 낮게 나온 이유에 대한 탐구와 이를 보완하기 위한 방법들에 대해 추가적으로 연구 해보고 싶다.

6. 첨언

본 연구는 세종대학교 에너지자원공학과 유효영과 지구자원시스템공학과 이가람에 의해 공동 진행되었다.

7. 참고 문헌

- [1] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1), 3-24.
- [2] Celebi, M. E., & Aydin, K. (Eds.). (2016). *Unsupervised learning algorithms* (Vol. 9, p. 103). Cham: Springer.
- [3] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html (Scikit-Learn)
- [4] Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255-260.
- [5] Noble, W. S. (2006). What is a support vector machine?. *Nature biotechnology*, 24(12), 1565-1567.
- [6] Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4), 433-459.