

1. SVM 으로 multi-class 분류 문제를 해결합니다. "from the scratch" 방식의 coding으로 구현해 주시기 바랍니다. (데이터는 따로 정해져 있지 않으며 각자 적절히 선정하여 적용합니다.)

데이터는 붓꽃 데이터를 기반으로 하였다. 길이 등의 특징을 가지며 이 특징들을 구분하는 작업을 진행하고 결과를 비교해보았다. Train & Test set 을 나누어 예측을 진행하였다.

[데이터 및 학습 파라미터 설정]

- 정확한 구분을 위해 활성화함수 커널을 linear, sigmoid, RBF 로 지정하여 서로 비교하였다.
- epoch = 20

[결과]

1) Linear Kernel (20 회)

[cross valid score & accuracy & std]

> score = [1 0.83 1 1 1 0.83 1 1 1 1 0.83 1 1 1 1 0.8 1 1 0.8 0.8]

> accuracy = 0.97368

> std = 0.085

> predict (test set)

[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1 0]

> confusion Matrix

$$M_{confusion} = \begin{bmatrix} 13 & 0 & 0 \\ 0 & 15 & 1 \\ 0 & 0 & 9 \end{bmatrix}$$

[Summary]

	Precision	Recall	F1-score	Support
Setosa	1.00	1.00	1.00	13
Versicolor	1.00	0.94	0.97	16
Virginica	0.90	1.00	0.95	9
Accuracy	-	-	0.97	38
Macro Average	0.97	0.98	0.97	38
Weighted Average	0.98	0.97	0.97	38

2) Sigmoid Kernel (20 회)

[cross valid score & accuracy & std]

> score = [0.6667 1. 1. 1. 1. 0.8333 1. 1. 1. 0.6667 0.8333 1]

> accuracy = 0.89474

> std = 0.1337

> predict (test set)

[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1 0]

> confusion Matrix

$$M_{confusion} = \begin{matrix} & \begin{matrix} 13 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 13 & 3 \\ 0 & 1 & 8 \end{matrix} & \end{matrix}$$

[Summary]

	Precision	Recall	F1-score	Support
Setosa	1.00	1.00	1.00	13
Versicolor	0.93	0.81	0.87	16
Virginica	0.73	0.89	0.80	9
Accuracy	-	-	0.89	38
Macro Average	0.89	0.90	0.89	38
Weighted Average	0.91	0.89	0.90	38

3) RBF Kernel (20 회)

[cross valid score & accuracy & std]

> score = [1 0.8333 1 1 1 0.8333 1 1 1 1 0.8333 1 1 1 1 1 0.8 1 0.8 1]

> accuracy = 0.97368

> std = 0.0784

> predict (test set)

[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1 0]

> confusion Matrix

$$M_{confusion} = \begin{matrix} & \begin{matrix} 13 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 15 & 1 \\ 0 & 0 & 9 \end{matrix} & \end{matrix}$$

[Summary]

	Precision	Recall	F1-score	Support
Setosa	1.00	1.00	1.00	13
Versicolor	1.00	0.94	0.97	16
Virginica	0.90	1.00	0.95	9
Accuracy			0.97	38
Macro Average	0.97	0.98	0.97	38
Weighted Average	0.98	0.97	0.97	38

- 정확도는 RBF, Linear Kernel이 비슷했으며, 이들은 Sigmoid Kernel 보다 좋은 성능을 보였다.
- 세 가지 Kernel 모두 붓꽃의 데이터는 동일하게 예측하였다.
- 세 가지 학습 속도는 차이가 거의 없기 때문에 RBF, Linear Kernel을 사용하는 것이 적절하다고 볼 수 있다.

2. (Ensemble 방식) Random Forest 를 구현합니다. 데이터는 수업시간에 예제로 활용한 "배드민턴" 경기의 진행여부를 포함한 변수 데이터 테이블을 활용합니다. 이 때, 적당한 분량을 테스트 데이터로 분리하여, 하나의 Decision Tree 방식을 구현한 결과와 비교 분석하여 결과를 정리합니다.

[데이터 및 학습 파라미터 설정]

데이터는 날씨, 바람, 온도, 습도 총 네 가지 특성을 사용하여 배드민턴의 활동 여부(1,0)을 결정하였다. 그 수준은 0~1 혹은 0~2로 설정하였으며, 이를 단순 결정트리 방식과 Random Forest 방식을 통해 그 특성의 중요성을 표기하고 비교해볼 수 있었다.

먼저 기본 이론 수식에 의해 나온 각 특성의 테스트 성능은 아래와 같다.

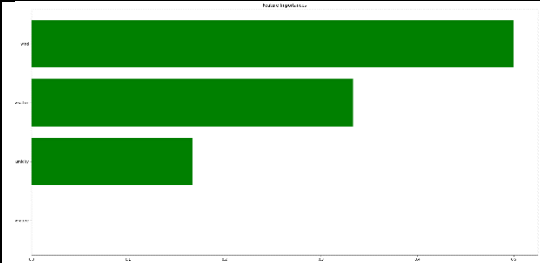
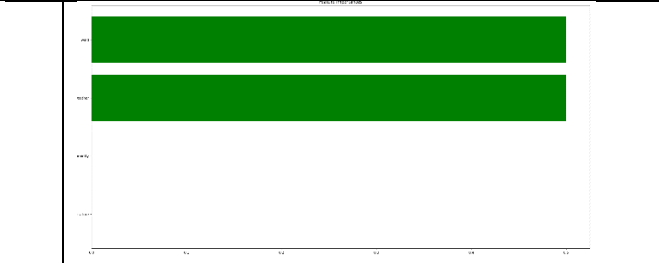
테스트 특성	성능
Weather	0.5
Wind	0.606875
Temperature	0.688725
Humidity	0.938725

기존 이론대로라면 배드민턴 활동 여부에 영향을 크게 주는 것은 날씨와 바람이며, 온도와 습도는 비교적 적은 영향(Importance)을 주는 것을 확인할 수 있다.

[결과]

결정나무와 Random Forest로 그 중요성을 검증한 결과표는 아래와 같다.

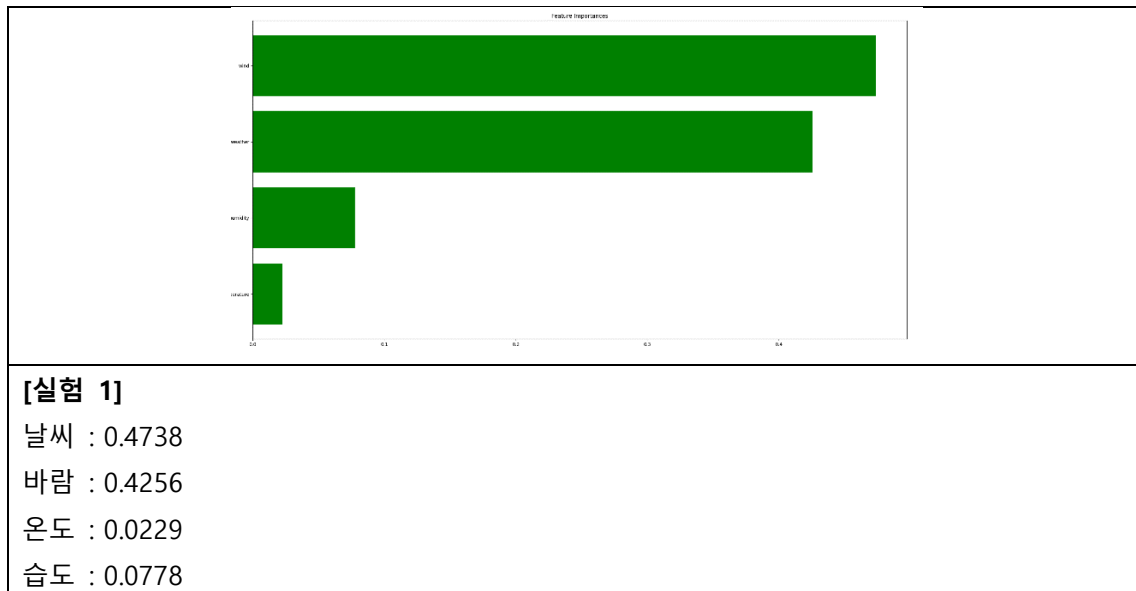
> 결정나무 그래프

 <p>A horizontal bar chart showing feature importance for a decision tree. The x-axis ranges from 0.0 to 0.5. The y-axis lists features: Weather (importance ~0.5), Wind (importance ~0.333), and Humidity (importance ~0.167). Temperature has an importance of 0.</p>	 <p>A horizontal bar chart showing feature importance for a random forest model. The x-axis ranges from 0.0 to 0.5. The y-axis lists features: Weather (importance ~0.5), Wind (importance ~0.5), and Humidity (importance ~0.938725). Temperature has an importance of 0.</p>
<p>[실험 1] 날씨 : 0.5 바람 : 0.333 온도 : 0 습도 : 0.167</p>	<p>[실험 2] 날씨 : 0.5 바람 : 0.5 온도 : 0 습도 : 0</p>

결정나무 그래프의 경우 매 학습마다 두 가지 형태의 특성으로 구분하였다.

Temperature를 활동 여부에 영향을 조금이나마 주는 것(좌측)과 전혀 영향이 없는 것으로 간주하는 것(우측)으로 구분하는 것이었다. 이는 이론값과는 약간 다르게 온도는 전혀 고려하지 않는 것으로 볼 수 있다.

> Random Foreset



Random Forest의 경우 이론값에 유사한 것을 확인할 수 있었다. 영향력이 적은 특성이라도 그 비중을 충분히 고려하여 모델을 생성하였음을 알 수 있었다.

두 결과에 따라 Random Forest의 모델이 결정트리 모델보다는 비교적 정확하고 합리적이라고 판단할 수 있다.

3. (Ensemble 방식) SVM과 ANN을 활용하여 Adaboost 를 구현 합니다. (데이터는 따로 정해져 있지 않으며 각자 적절히 선정하여 적용합니다.)

데이터셋으로는 유방암 데이터를 활용하였다. 이는 30가지 특성을 가지고 있으며, 이 중에서 가장 영향력 있는 특성을 고려할 수 있는 Adaboost 방식을 통해 모델학습을 진행한다. 이 데이터는 569개이며, 이미지 사진에서 보이는 종양 의심 부분이 얼마나 크지와 그 외의 경계선 선명도 등을 특징으로 한다. 이 때 크기는 그 지름뿐만 아니라 면적의 크기, 편차 등을 활용한다.

[데이터 및 학습 파라미터 설정]

- 데이터셋은 Train과 Test 셋을 랜덤하게 나눈다.
- 특성별 데이터의 편차가 심하므로 StandardScaler()를 통해 전처리를 먼저 진행한다.
- Adaboost의 경우 학습 인자로서 평가모델(예측기)의 개수(estimators)를 1개만 돌릴 경우 부정확할 것을 고려하여 1, 10, 20, 40 개로 설정하여 개수에 따라 결과가 다른지를 분석한다.
- 학습한 모델을 기반으로 예측을 진행 후 target 값과 비교한다. 이 때, 예측이 틀린 인자들의 개수를 측정하여 각 평가모델 개수에 따라 적중률을 확인한다.

1) SVM 방식

```

-----
| Ada boost SVM Result |
-----
<score>
  1 : 0.9391304347826086
 10 : 0.9391304347826086
 20 : 0.9304347826086957
 40 : 0.8956521739130434
<<<Prediction>>>
target = [1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0
1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0
1 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
  1 : [1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 0 0]
 10 : [1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1
1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
 20 : [1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1
1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
 40 : [1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1
1 1 0]
<<<Failure Rate in each number of estimator>>>
[Number] [Rate (%)]
  1 : 0.03508771929824561
 10 : 6.140350877192982
 20 : 5.263157894736842
 40 : 32.45614035087719
<<<Accuracy>>>
  1 : 0.9649122807017544
 10 : 0.9385964912280702
 20 : 0.9473684210526315
 40 : 0.6754385964912281
process finished

```

SVM 방식의 경우 평가 모델의 개수가 늘어날수록 score가 낮아지는 양상을 보였다. 또한 test data set을 Adaboost 모델에 모두 넣어 예측을 진행했는데, SVM평가모델의 개수가 늘어날수록 target값과 다르게 예측하는 비율이 늘어났고, 특히 SVM을 40개로 설정했을 때, 그 적중률은 70%도 채 되지 않는 결과를 보였다. 오히려 Overfitting이 된 것이라고 보여진다. 따라서 평가용 모델의 개수도 너무 늘릴 경우 전체 모델이 부정확해진다는 결과를 얻어볼 수 있었다.

2) ANN 방식

```

-----
| Ada boost ANN Result |
-----
<score>
  1 : 0.9379446640316205
 10 : 0.9379446640316205
 20 : 0.9379446640316205
 40 : 0.9292490118577076
<<<Prediction>>>
target = [1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
1 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
  1 : [1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1
1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0
1 1 0]
 10 : [1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
1 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
 20 : [1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
 40 : [1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
1 1 0]
<<<Failure Rate in each number of estimator>>>
[Number] [Rate (%)]
  1 : 0.10526315789473684
 10 : 1.7543859649122806
 20 : 3.508771929824561
 40 : 3.508771929824561
<<<Accuracy>>>
  1 : 0.8947368421052632
 10 : 0.9824561403508771
 20 : 0.9649122807017544
 40 : 0.9649122807017544
process finished

```

ANN 방식의 경우 score는 평가 모델 개수와 상관없이 거의 동일하다. SVM방식과는 다르게 score 자체에는 큰 이변이 없었으나 평가모델이 40개인 경우에는 소폭 하락이 있었다. 또한 score와는 다르게 실제 test data set을 각 Adaboost 모델로 평가한 결과 SVM방식과 마찬가지로 적중률이 하락하는 것을 볼 수 있었다.

[결론]

SVM방식, ANN 방식 모두 평가모델의 개수가 적을수록 그 정확도는 높다고 볼 수 있었다. 또한 평가모델의 개수가 늘어날수록 오히려 Overfitting이 되어 다른 데이터에 대해 예측정확도가 떨어진다는 것을 알 수 있었다.

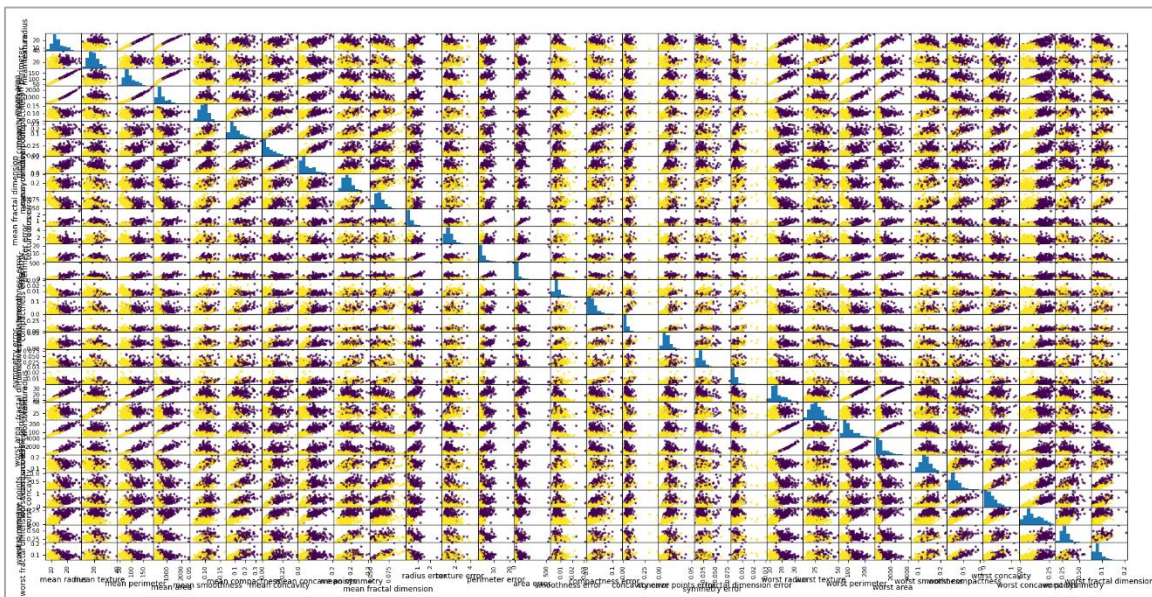
ANN의 가장 좋은 모델에의 오예측 비율 0.10 %보다 SVM에서의 가장 좋은 모델의 오예측 비율이 0.05 %이므로 SVM 방식의 Adaboost가 유방암 데이터에서 더 정확한 모델이라고 할 수 있다. (물론 평가 모델의 개수는 1개일 때이다.)

4. 차원 축소 문제로서, 수업시간에 활용 하지 않은 데이터를 대상으로 분류 문제를 정의하고, PCA와 LDA 방식을 적용하여 분류한 결과를 비교합니다. (예시, 분류하고자하는 데이터와 문제를 정의 -> 데이터를 PCA로 차원 축소 변환 -> 축소된 데이터를 학습하여 분류 테스트 수행 -> 같은 방식으로 LDA로 차원 축소하여 얻은 결과를 비교 정리)

데이터셋으로는 유방암 데이터를 활용하였다. 이는 30가지 특성을 가지고 있으며, 이 중에서 가장 영향력 있는 특성을 고려할 수 있는 Adaboost 방식을 통해 모델학습을 진행한다. 이 데이터는 569개이며, 이미지 사진에서 보이는 종양 의심 부분이 얼마나 크지와 그 외의 경계선 선명도 등을 특징으로 한다. 이 때 크기는 그 지름뿐만 아니라 면적의 크기, 편차 등을 활용한다.

[데이터 및 학습 파라미터 설정]

- 데이터셋은 Train과 Test 셋을 랜덤하게 나눈다.
- 각 방식에 따라 차원축소를 진행하기 전, 특성별 데이터의 편차가 심하므로 StandardScaler()를 통해 전처리를 먼저 진행한다.
- 기존 특성은 총 30개로 30차원이다.
- > 기존 데이터 특성 30개의 전체 분산도



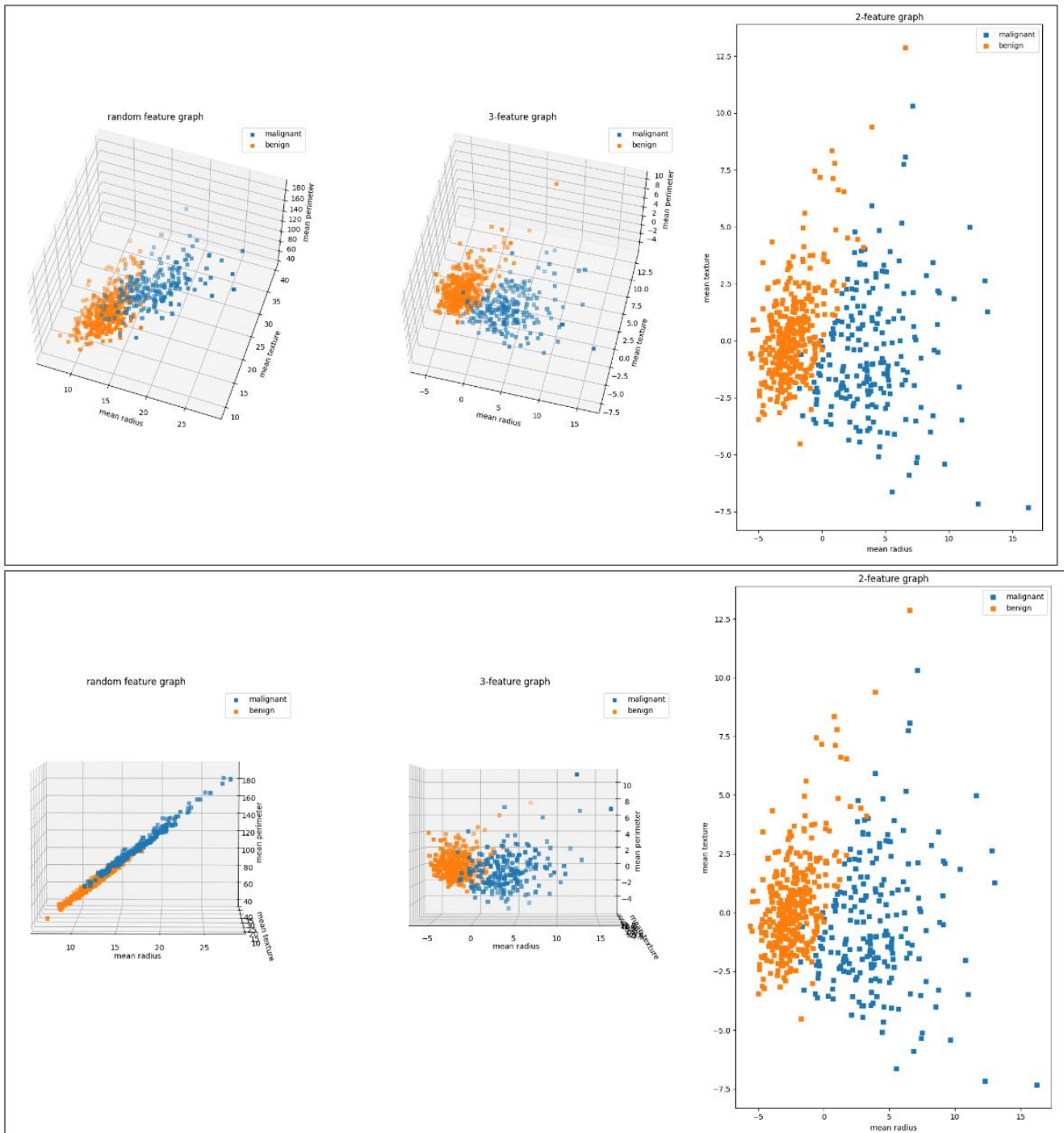
각 특성과 그 외의 특성 중 한 개를 매칭하여 2차원 분산도로 표현한 그래프이다. 노란색(유방암 X), 보라색(유방암 O)의 전체적인 분포를 확인할 수 있었으며, 군집도가 육안으로도 잘 되어있는 특성들이 있는 반면, 그렇지 않고 섞여진 모양을 보이는 특성들도 볼 수 있었다. 이 데이터를 차원축소를 통해 전체적으로 얼마나 잘 예측해내는지 확인해본다.

[결과]

1) PCA

- 특성들의 데이터 분산정도를 먼저 확인한 후 축소되어지는 과정에서 축소 과정에서 원본데이터를 얼마나 잘 표현하는지(데이터의 분산)를 확인한 후 각 score를 확인한다.
- 차원 축소는 차원을 10개, 3개, 2개로 줄여보면서 각 차이를 확인해본다.
- 그래프는 3차원까지만 그려지므로 10개로 차원축소한 경우는 결과값만을 확인한다.

> 차원 축소 후 3차원, 2차원 그래프를 Top View 및 Side View 확인



- 주황색은 유방암이 아니고, 파란색이 유방암인 데이터이다.

3차원 그래프로 확인해본 결과, 특성별로 각자의 성격을 잘 나타내고 있다고 확인해볼 수 있었다. 또한 2차원으로 진행되면서 확실히 각 특성이 자신의 영역을 더 잘 표현하고 있음을 볼 수 있었다.

> 차원 축소 차원별 분산정도(Variance Ratio), 원본데이터의 표현률, score 측정.

<<<Variance Ratio>>>

Dim) 10 : [0.44896035 0.18472104 0.09183385 0.06446333 0.05351866 0.03895187
0.02208771 0.0156405 0.01344822 0.01131915]

3 : [0.44896035 0.18472104 0.09183385]

2 : [0.44896035 0.18472104]

각 차원의 초기 2(3)개의 주성분 요소로 원본데이터의 약 72.55152452459441%를 표현

<<<Accuracy>>>

Dim) 30(pure) : 0.9560627231796305

10 : 0.9648191274646795

3 : 0.9665890389691041

2 : 0.9631268436578171

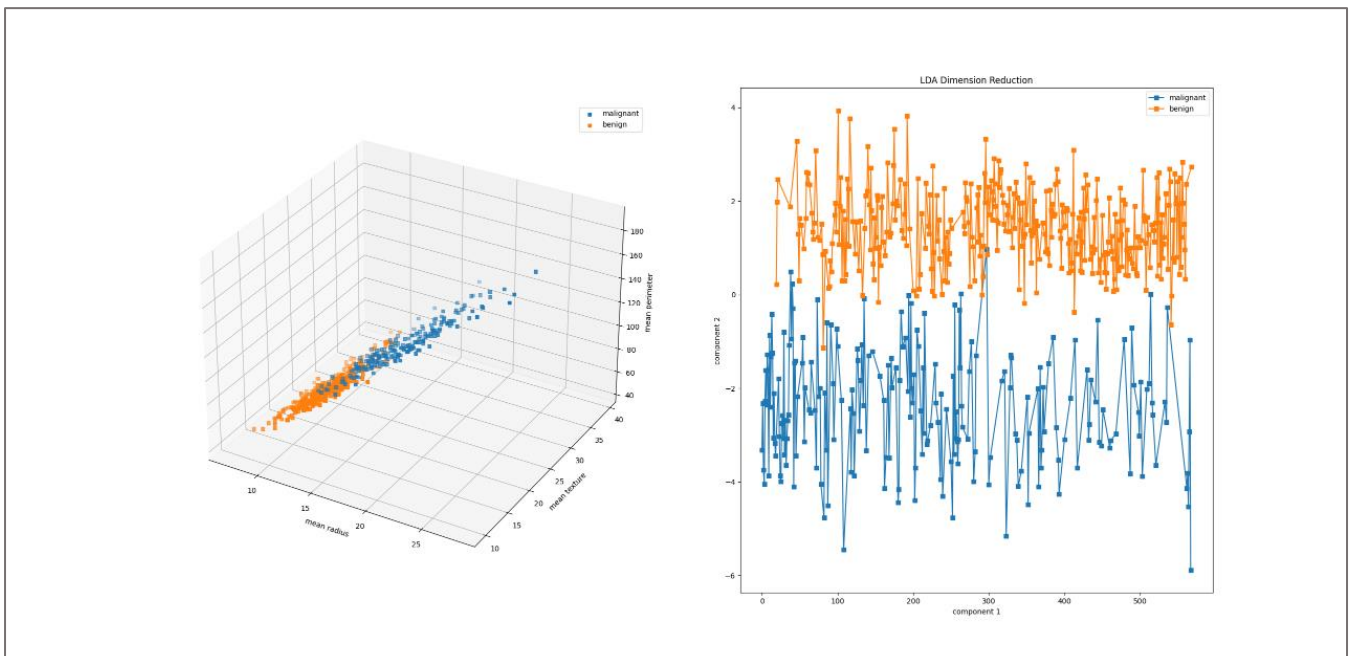
기본 차원은 30임에 대해 10차원 이하로 내려가면서 그 분산의 정도는 유사해졌다. 특히 Variance Ratio의 데이터는 맨 앞부터 가장 영향력 있는 특성들을 그 분산정도의 크기에 따라 내림차순이 되어 있는데, 특히 앞의 3개의 특성이 약 72%의 원본 표현률을 가지고 있다고 볼 수 있다.

정확도(Accuracy)의 경우 차원이 감소하면서 원본 데이터에 비해서 소폭 증가함을 보였다. 가장 높은 정확도는 3차원으로 줄였을 때이며, 특히 2차원으로 축소할 경우 3번째 영향력 있던 특성이 없어지면서 3차원으로 축소할 경우보다 정확도가 감소하였다.

2) LDA

- 특성들의 데이터 분산정도를 먼저 확인한 후 축소되어지는 과정에서 축소 과정에서 원본데이터를 얼마나 잘 표현하는지(데이터의 분산)를 확인한 후 각 score를 확인한다.
- 차원 축소는 차원을 3개, 2개로 줄여보면서 각 차이를 확인해본다.

> 차원 축소 후 3차원, 2차원 그래프확인



우측 사진은 유방암 데이터가 한 차원(y축)에 대해서 거의 모든 데이터를 분류할 수 있을 것처럼 보인다. 즉, 2차원의 데이터를 보면 분리하기에 아주 좋은 데이터를 보이고 있다.

> 차원 축소 차원별 분산정도(Variance Ratio), 원본데이터의 표현률, score 측정

```
<<<Variance Ratio>>>
      2 : [1.]
<<<Accuracy>>>
      Dim) 30(pure) : 0.9578326346840553
      2          : 0.9613724576929048
[0.92982456 0.93859649 0.98245614 0.97368421 0.96460177]
```

그래프에서 본 것처럼 2차원으로 차원축소가 진행됨에 따라서 그 정확도가 증가했음을 볼 수 있다. 하지만 PCA의 3차원 축소(0.96589)에 비해 약간 낮은 정확도를 보인다.

따라서 PCA축소가 유방암 데이터를 분리하기에는 조금 더 적합한 모델이라고 볼 수 있다.

Source Code

1.

```
from sklearn import datasets

from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import pandas as pd

# 1. 데이터 준비
raw_iris = datasets.load_iris()
X = raw_iris.data
y = raw_iris.target
pd.DataFrame(X).head()

# 2. 데이터 전처리
X_tn, X_te, y_tn, y_te = train_test_split(X, y, random_state = 1)
std_scale = StandardScaler()
std_scale.fit(X_tn)
X_tn_std = std_scale.transform(X_tn)
X_te_std = std_scale.transform(X_te)

# 3. 모델 학습
clf_svm_ln = svm.SVC(kernel = 'linear', random_state = 1)
clf_svm_ln.fit(X_tn_std, y_tn)
clf_svm_sg = svm.SVC(kernel = 'sigmoid', random_state = 1)
clf_svm_sg.fit(X_tn_std, y_tn)
clf_svm_rbf = svm.SVC(kernel = 'rbf', random_state = 1)
clf_svm_rbf.fit(X_tn_std, y_tn)

cv_scores = cross_val_score(clf_svm_ln, X_tn_std, y_tn, cv = 20, scoring = 'accuracy')
print(cv_scores)
print(cv_scores.mean())
print(cv_scores.std())
cv_scores = cross_val_score(clf_svm_sg, X_tn_std, y_tn, cv = 20, scoring = 'accuracy')
print(cv_scores)
print(cv_scores.mean())
print(cv_scores.std())
cv_scores = cross_val_score(clf_svm_rbf, X_tn_std, y_tn, cv = 20, scoring = 'accuracy')
print(cv_scores)
print(cv_scores.mean())
print(cv_scores.std())

# 4. 결과 확인
pred_svm_ln = clf_svm_ln.predict(X_te_std)
print(pred_svm_ln)
pred_svm_sg = clf_svm_sg.predict(X_te_std)
print(pred_svm_sg)
```

```

pred_svm_rbf = clf_svm_rbf.predict(X_te_std)
print(pred_svm_rbf)

accuracy_ln = accuracy_score(y_te, pred_svm_ln)
print(accuracy_ln)
accuracy_sg = accuracy_score(y_te, pred_svm_sg)
print(accuracy_sg)
accuracy_rbf = accuracy_score(y_te, pred_svm_rbf)
print(accuracy_rbf)

conf_matrix_ln = confusion_matrix(y_te, pred_svm_ln)
print(conf_matrix_ln)
conf_matrix_sg = confusion_matrix(y_te, pred_svm_sg)
print(conf_matrix_sg)
conf_matrix_rbf = confusion_matrix(y_te, pred_svm_rbf)
print(conf_matrix_rbf)

class_report_ln = classification_report(y_te, pred_svm_ln)
print("---- linear ----")
print(class_report_ln)
class_report_sg = classification_report(y_te, pred_svm_sg)
print("---- sigmoid ----")
print(class_report_sg)
class_report_rbf = classification_report(y_te, pred_svm_rbf)
print("---- rbf ----")
print(class_report_rbf)

```

2.

1) Decision Tree

```

import numpy as np

import matplotlib.pyplot as plt
import pandas as pd
import datetime

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

# 1. 데이터 준비
data = pd.read_csv('../docs/badminton_weather_data.csv')
print(data)
...

   weather  wind  temperature  humidity  target
0         0     1             0         1         0
1         1     1             2         1         0
2         0     0             2         1         1
3         2     0             1         2         1
4         0     0             1         0         1
5         1     0             0         2         0
6         1     0             1         2         0
7         0     1             2         0         0
...

```

2. 데이터 전처리

```
feature = ['weather', 'wind', 'temperature', 'humidity']
data_input = data[feature].to_numpy()
data_target = data[['target']].to_numpy()
input_train, input_test, target_train, target_test = train_test_split(data_input,
data_target, random_state=42)
```

3. 모델 생성 및 학습

```
model_dctree = DecisionTreeClassifier(max_depth=None)
model_dctree.fit(input_train, target_train)
```

```
imp = model_dctree.feature_importances_
a = model_dctree.n_outputs_
b = model_dctree.ccp_alpha
c = model_dctree.class_weight
d = model_dctree.criterion
e = model_dctree.tree_.impurity
f = model_dctree.tree_.value
```

```
print(e)
print(f)
indices = np.argsort(imp)
```

```
print(model_dctree.score(input_train, target_train))
print(model_dctree.score(input_test, target_test))
print("----- importance (impurity) -----")
print(feature)
print(imp)
# plt.figure(figsize=(7,7))
plt.title('Feature Importances')
plt.barh(range(len(indices)), imp[indices], color='g', align='center')
plt.yticks(range(len(indices)), [feature[i] for i in indices])
plt.show()
```

```
print('process finished')
```

2) Random Forest

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

1. 데이터 준비

```
data = pd.read_csv('../docs/badminton_weather_data.csv')
print(data)
'''
```

	weather	wind	temperature	humidity	target
0	0	1	0	1	0
1	1	1	2	1	0

```

2      0      0      2      1      1
3      2      0      1      2      1
4      0      0      1      0      1
5      1      0      0      2      0
6      1      0      1      2      0
7      0      1      2      0      0
...

```

2. 데이터 전처리

```

feature = ['weather', 'wind', 'temperature', 'humidity']
data_input = data[feature].to_numpy()
data_target = data[['target']].to_numpy().ravel()
input_train, input_test, target_train, target_test = train_test_split(data_input,
data_target, random_state=42)

```

3. 모델 생성 및 학습

```

model = RandomForestClassifier(n_estimators=10, max_depth=2, random_state=0)
model.fit(data_input, data_target)

```

4. 결과 분석

```

imp = model.feature_importances_
indices = np.argsort(imp)
print(model.score(input_train, target_train))
print(model.score(input_test, target_test))

```

----- graph -----

```

plt.title('Feature Importances')
plt.barh(range(len(indices)), imp[indices], color='g', align='center')
plt.yticks(range(len(indices)), [feature[i] for i in indices])
plt.show()

```

```

print('process finished')

```

3.

1) SVM

```

import numpy as np
import matplotlib as plt
import pandas as pd

from sklearn.svm import SVC
# from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import plot_tree
from sklearn.model_selection import cross_val_score

cancer_data = load_breast_cancer()
# cancer_data = load_iris()
input_train, input_test, target_train, target_test = train_test_split(cancer_data['data'],
cancer_data['target'], test_size=0.2, random_state=42)

```

```

model_svm = SVC(probability=True, kernel='linear', verbose=1)

model_ABC = AdaBoostClassifier(n_estimators=1, estimator=model_svm, learning_rate=1)
model_ABC_100 = AdaBoostClassifier(n_estimators=10, estimator=model_svm, learning_rate=1)
model_ABC_200 = AdaBoostClassifier(n_estimators=20, estimator=model_svm, learning_rate=1)
model_ABC_400 = AdaBoostClassifier(n_estimators=40, estimator=model_svm, learning_rate=1)

model_ABC.fit(input_train, target_train)
model_ABC_100.fit(input_train, target_train)
model_ABC_200.fit(input_train, target_train)
model_ABC_400.fit(input_train, target_train)

pred_ABC = model_ABC.predict(input_test)
pred_ABC_100 = model_ABC_100.predict(input_test)
pred_ABC_200 = model_ABC_200.predict(input_test)
pred_ABC_400 = model_ABC_400.predict(input_test)

score = cross_val_score(model_ABC, input_test, target_test, cv=5)
print("<score>")
print(score.mean())
# print(score.std())

# imp = model_ABC.feature_importances_
class_ABC = model_ABC.get_params()
print(class_ABC)

# target 과 predict 값을 연산비교하여 일치하지 않는 개수가 얼마나 되는지 확인
fail_pred_ABC = np.logical_xor(target_test, pred_ABC)
fail_pred_ABC_100 = np.logical_xor(target_test, pred_ABC_100)
fail_pred_ABC_200 = np.logical_xor(target_test, pred_ABC_200)
fail_pred_ABC_400 = np.logical_xor(target_test, pred_ABC_400)
# 일치하지 않는 비율 구하기
rate_of_fail_pred_ABC = fail_pred_ABC.sum() / len(fail_pred_ABC)
rate_of_fail_pred_ABC_100 = fail_pred_ABC_100.sum() / len(fail_pred_ABC_100) * 100
rate_of_fail_pred_ABC_200 = fail_pred_ABC_200.sum() / len(fail_pred_ABC_200) * 100
rate_of_fail_pred_ABC_400 = fail_pred_ABC_400.sum() / len(fail_pred_ABC_400) * 100

print("-----")
print(" | Ada boost SVM Result | ")
print("-----")
print("<<<rate of fail matches>>>")
print("target = ", target_test)
print("number of estimator")
print("      1 : ", rate_of_fail_pred_ABC)
print("     10 : ", rate_of_fail_pred_ABC_100)
print("     20 : ", rate_of_fail_pred_ABC_200)
print("     40 : ", rate_of_fail_pred_ABC_400)

print("<<<Accuracy>>>")
print("      1 : ", metrics.accuracy_score(target_test, pred_ABC))

```



```

print("      10 : ", metrics.accuracy_score(target_test, pred_ABC_100))
print("      20 : ", metrics.accuracy_score(target_test, pred_ABC_200))
print("      40 : ", metrics.accuracy_score(target_test, pred_ABC_400))

print('process finished')

```

2) ANN

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.wrappers.scikit_learn import KerasRegressor
from keras.optimizers import SGD
from sklearn.model_selection import cross_val_score
import keras.utils
from sklearn import metrics

cancer_data = load_breast_cancer()
input_train, input_test, target_train, target_test = train_test_split(cancer_data['data'],
cancer_data['target'], test_size=0.2, random_state=42)

model_ann = Sequential()
model_ann.add(Dense(512, input_dim=input_train.shape[1], kernel_initializer='normal',
activation='relu'))
# model_ann.add(Dropout) # 과적합 방지
model_ann.compile(optimizer='adam',
                  # loss='sparse_categorical_crossentropy',
                  loss='mean_squared_error',
                  metrics=['accuracy'])
model_ann.fit(input_train, target_train, epochs=10, verbose=2)

model_ABC = AdaBoostClassifier(n_estimators=1, learning_rate=1)
model_ABC_100 = AdaBoostClassifier(n_estimators=10, learning_rate=1)
model_ABC_200 = AdaBoostClassifier(n_estimators=20, learning_rate=1)
model_ABC_400 = AdaBoostClassifier(n_estimators=40, learning_rate=1)

model_ABC.fit(input_train, target_train)
model_ABC_100.fit(input_train, target_train)
model_ABC_200.fit(input_train, target_train)
model_ABC_400.fit(input_train, target_train)

pred_ABC = model_ABC.predict(input_test)
pred_ABC_100 = model_ABC_100.predict(input_test)
pred_ABC_200 = model_ABC_200.predict(input_test)

```

```

pred_ABC_400 = model_ABC_400.predict(input_test)

score = cross_val_score(model_ABC, input_test, target_test, cv=5)
print("<score>")
print(score.mean())
# print(score.std())

class_ABC = model_ABC.get_params()
print(class_ABC)

# target 과 predict 값을 연산비교하여 일치하지 않는 개수가 얼마나 되는지 확인
fail_pred_ABC      = np.logical_xor(target_test, pred_ABC      )
fail_pred_ABC_100  = np.logical_xor(target_test, pred_ABC_100)
fail_pred_ABC_200  = np.logical_xor(target_test, pred_ABC_200)
fail_pred_ABC_400  = np.logical_xor(target_test, pred_ABC_400)
# 일치하지 않는 비율 구하기
rate_of_fail_pred_ABC = fail_pred_ABC.sum() / len(fail_pred_ABC)
rate_of_fail_pred_ABC_100 = fail_pred_ABC_100.sum() / len(fail_pred_ABC_100) * 100
rate_of_fail_pred_ABC_200 = fail_pred_ABC_200.sum() / len(fail_pred_ABC_200) * 100
rate_of_fail_pred_ABC_400 = fail_pred_ABC_400.sum() / len(fail_pred_ABC_400) * 100

print("-----")
print(" | Ada boost ANN Result | ")
print("-----")
print("<<<rate of fail matches>>>")
print("target = ", target_test)
print("number of estimator")
print("      1 : ", rate_of_fail_pred_ABC)
print("     10 : ", rate_of_fail_pred_ABC_100)
print("     20 : ", rate_of_fail_pred_ABC_200)
print("     40 : ", rate_of_fail_pred_ABC_400)

print("<<<Accuracy>>>")
print("      1 : ", metrics.accuracy_score(target_test, pred_ABC))
print("     10 : ", metrics.accuracy_score(target_test, pred_ABC_100))
print("     20 : ", metrics.accuracy_score(target_test, pred_ABC_200))
print("     40 : ", metrics.accuracy_score(target_test, pred_ABC_400))

print('process finished')

```

4.

1) PCA

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier

```

```

from pandas.plotting import scatter_matrix
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.wrappers.scikit_learn import KerasRegressor
from keras.optimizers import SGD
from sklearn.model_selection import cross_val_score
import keras.utils
from sklearn import metrics

G_SHOW_VARIANCE = 0

# 1. 데이터 준
cancer_data = load_breast_cancer()
df_cancer_data = pd.DataFrame(data=cancer_data.data, columns=cancer_data.feature_names)
df_cancer_data['target'] = cancer_data.target

print(cancer_data.feature_names)
print(df_cancer_data.head())
# 기본 데이터 분산정도 확인
if G_SHOW_VARIANCE:
    plt.figure("Variance of cancer data (all)")
    plt.title("Variance of cancer data (all)")
    scatter_matrix(df_cancer_data, c=cancer_data['target'], marker='o', s=10, alpha=.8)
    plt.show()

# 2. 데이터 전처리
scaler = StandardScaler()
cancer_data_std = scaler.fit_transform(df_cancer_data)

# 일부 특성의 기본 데이터셋 분산도
markers = ["*", "s"]
fig = plt.figure('3-feature variances', figsize=(20,20))
ax = fig.add_subplot(131, projection='3d')
for i, markers in enumerate(markers):
    f1 = df_cancer_data[df_cancer_data["target"]==i]['mean radius']
    f2 = df_cancer_data[df_cancer_data["target"]==i]['mean texture']
    f3 = df_cancer_data[df_cancer_data["target"]==i]['mean perimeter']
    ax.scatter(f1,f2,f3, marker='s', label=cancer_data.target_names[i])
ax.set_xlabel('mean radius')
ax.set_ylabel('mean texture')
ax.set_zlabel('mean perimeter')
plt.title("random feature graph")
plt.legend()

# PCA
# 10th dimension
pca10 = PCA(n_components=10)
pca10.fit(cancer_data_std)
df_pca10 = pca10.transform(cancer_data_std)
df_pca10 = pd.DataFrame(data=df_pca10)
df_pca10['target'] = cancer_data['target']

```

```

# 3rd dimension
pca3 = PCA(n_components=3)
pca3.fit(cancer_data_std)
df_pca3 = pca3.transform(cancer_data_std)
df_pca3 = pd.DataFrame(data=df_pca3)
df_pca3['target'] = cancer_data['target']
# 차원 축소 데이터 그래프
markers = ["*", "s"]
ax = fig.add_subplot(132, projection='3d')
for i, markers in enumerate(markers):
    f1 = df_pca3[df_pca3["target"]==i][0]
    f2 = df_pca3[df_pca3["target"]==i][1]
    f3 = df_pca3[df_pca3["target"]==i][2] # scaling
    ax.scatter(f1, f2, f3, marker='s', label=cancer_data.target_names[i])
ax.set_xlabel('mean radius')
ax.set_ylabel('mean texture')
ax.set_zlabel('mean perimeter')
plt.title("3-feature graph")
plt.legend()

```

```

# 2nd dimension
pca2 = PCA(n_components=2)
pca2.fit(cancer_data_std)
df_pca2 = pca2.transform(cancer_data_std)
df_pca2 = pd.DataFrame(data=df_pca2)
df_pca2['target'] = cancer_data['target']
# 차원 축소 데이터 그래프
markers = ["*", "s"]
ax = fig.add_subplot(133)
for i, markers in enumerate(markers):
    f1 = df_pca2[df_pca2["target"]==i][0]
    f2 = df_pca2[df_pca2["target"]==i][1]
    ax.scatter(f1, f2, marker='s', label=cancer_data.target_names[i])
ax.set_xlabel('mean radius')
ax.set_ylabel('mean texture')
plt.title("2-feature graph")
plt.legend()

```

```

plt.show()

```

```

var_ratio_pca10 = pca10.explained_variance_ratio_
var_ratio_pca3 = pca3.explained_variance_ratio_
var_ratio_pca2 = pca2.explained_variance_ratio_
print("<<<Variance Ratio>>>")
print("    Dim) 10 : ", var_ratio_pca10)
print("        3 : ", var_ratio_pca3)
print("        2 : ", var_ratio_pca2)
print("각 차원의 초기 2(3)개의 주성분 요소로 원본데이터의 약 {}%를  
표현" .format(np.sum(var_ratio_pca10[0:3])*100))

```

```

# Random Forest 분류기로 비교
rf = RandomForestClassifier()

```

```

scores_pure = cross_val_score(rf, cancer_data['data'], cancer_data['target'],
scoring='accuracy', cv=5)
scores_pca10 = cross_val_score(rf, df_pca10.iloc[:, :-1], cancer_data['target'],
scoring='accuracy', cv=5)
scores_pca3 = cross_val_score(rf, df_pca3.iloc[:, :-1], cancer_data['target'],
scoring='accuracy', cv=5)
scores_pca2 = cross_val_score(rf, df_pca2.iloc[:, :-1], cancer_data['target'],
scoring='accuracy', cv=5)
print("<<<Accuracy>>>")
print("    Dim) 30(pure) : ", np.mean(scores_pure))
print("    10      : ", np.mean(scores_pca10))
print("    3       : ", np.mean(scores_pca3))
print("    2       : ", np.mean(scores_pca2))

```

2) LDA

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_breast_cancer
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from pandas.plotting import scatter_matrix
from sklearn.model_selection import cross_val_score

G_SHOW_VARIANCE = 0

# 1. 데이터 준비
cancer_data = load_breast_cancer()
df_cancer_data = pd.DataFrame(data=cancer_data.data, columns=cancer_data.feature_names)
df_cancer_data['target'] = cancer_data.target

print(cancer_data.feature_names)
print(len(cancer_data.feature_names))
print(df_cancer_data.head())
# 기본 데이터 분산정도 확인
if G_SHOW_VARIANCE:
    plt.figure("Variance of cancer data (all)")
    plt.title("Variance of cancer data (all)")
    scatter_matrix(df_cancer_data, c=cancer_data['target'], marker='o', s=10, alpha=.8)
    plt.show()

# 2. 데이터 전처리
scaler = StandardScaler()
cancer_data_std = scaler.fit_transform(cancer_data.data)

# 일부 특성의 기본 데이터셋 분산도
markers = ["*", "s"]
fig = plt.figure('3-feature variances', figsize=(20,20))
ax = fig.add_subplot(121, projection='3d')
for i, markers in enumerate(markers):

```

```

    f1 = df_cancer_data[df_cancer_data["target"]==i]['mean radius']
    f2 = df_cancer_data[df_cancer_data["target"]==i]['mean texture']
    f3 = df_cancer_data[df_cancer_data["target"]==i]['mean perimeter']
    ax.scatter(f1,f2,f3, marker='s', label=cancer_data.target_names[i])
ax.set_xlabel('mean radius')
ax.set_ylabel('mean texture')
ax.set_zlabel('mean perimeter')
plt.legend()

# LDA
lda2 = LinearDiscriminantAnalysis(n_components=1)
lda2.fit(cancer_data_std, cancer_data['target'])
df_lda2 = lda2.transform(cancer_data_std)
df_lda2 = pd.DataFrame(data=df_lda2)
df_lda2['target'] = cancer_data['target']

# 차원 축소 데이터 그래프
markers = ["s","s"]
ax = fig.add_subplot(122)
for i,markers in enumerate(markers):
    f1 = df_lda2[df_lda2["target"]==i][0]
    ax.plot(f1, marker='s', label=cancer_data.target_names[i])
ax.set_xlabel('component 1')
ax.set_ylabel('component 2')
plt.title("LDA Dimension Reduction")
plt.legend()

plt.show()
var_ratio_lda2 = lda2.explained_variance_ratio_
print("<<<Variance Ratio>>>")
print("          2 : ", var_ratio_lda2)

# Random Forest 분류기로 비교
rf = RandomForestClassifier()
scores_pure = cross_val_score(rf, cancer_data['data'], cancer_data['target'],
scoring='accuracy', cv=5)
scores_lda2 = cross_val_score(rf, df_lda2.iloc[:, :-1], cancer_data['target'],
scoring='accuracy', cv=5)
print("<<<Accuracy>>>")
print("      Dim) 30(pure) : ", np.mean(scores_pure))
print("          2          : ", np.mean(scores_lda2))

print(scores_pure)

```