다변량데이터분석

과제 3

2021150456 이예지

목차

[다변량데이터분석][과제 3]

[Q1] 본인이 생각하기에 "예측정확도"도 중요하지만 "예측 결과물에 대한 해석"이 매우 중요할 것으로 생각되는 분류 문제를 다루고 있는 데이터셋을 1 개 선정하고 선정 이유를 설명하시오. 데이터셋 탐색은 아래에서 제시된 Data Repository 를 포함하여 여러 Repository 를 검색해서 결정하시오. 보고서에는 데이터를 다운로드할 수 있는 링크를 반드시 제공하시오.

선정한 데이터셋은 "Fetal Health Classification"으로, 아이와 산모의 사망을 막기 위해 태아의 건강을 분류하는 데이터셋이다.

보건, 의료 계열의 데이터는 예측정확도도 중요하지만, 생명과 관련이 있어 예측 결과물에 대한 해석이 매우 중요하다. 위 데이터셋 역시 정확한 예측도 중요하지만 산모가 아이를 건강히 출산할 수 있도록 돕기 위해서는 어떠한 이유로 태아의 건강 상태가 판별되었는지 알고, 건강이 좋지 않다고 판별된 경우 결과 해석을 통해 적절한 조치를 취하는 것이 중요하므로 분석 데이터셋으로 선정하였다.

링크: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification

주요 변수들은 다음과 같다.

[표 1] Fetal Health Dataset 변수 설명

변수명	설명
baseline value	기본 태아 심박수
accelerations	가속 횟수/sec
fetal_movement	태아 움직임 횟수/sec
uterine_contractions	자궁 수축 횟수/sec
light_decelerations	경미한 감속 횟수/sec
severe_decelerations	심한 감속 횟수/sec
prolongued_decelerations	지속적 감속 횟수/sec
abnormal_short_term_variability	비정상 단기 변동성 비율
mean_value_of_short_term_variability	단기 변동성의 평균값
percentage_of_time_with_abnormal_long_term_variability	비정상 장기 변동성 비율
mean_value_of_long_term_variability	장기 변동성의 평균값
histogram_width	모든 관측치로 만든 히스토그램의 폭
histogram_min	히스토그램 최솟값
histogram_max	히스토그램 최댓값
histogram_number_of_peaks	검사 히스토그램의 피크 수
histogram_number_of_zeroes	검사 히스토그램의 0 수
histogram_mode	히스토그램 최빈값
histogram_mean	히스토그램 평균값
histogram_median	히스토그램 중위값
histogram_variance	히스토그램 분산
histogram_tendency	히스토그램 추세
fetal_health(Target Variable)	태아 건강 상태: 1 - 정상, 2 – 의심, 3 - 병리

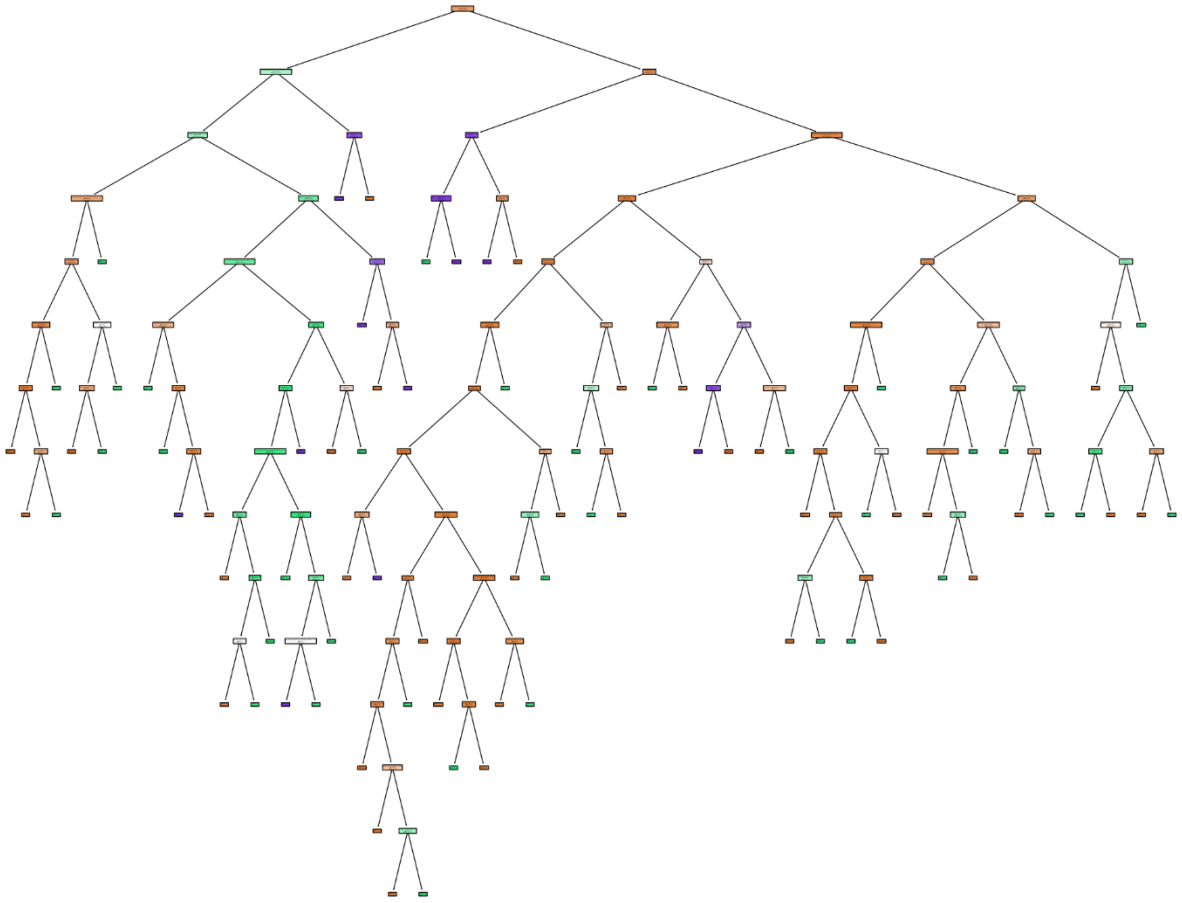
총 21 개의 설명변수와 1 개의 종속변수로 이루어져 있으며, 전체 데이터 개수는 2126 개이다. 종속변수 내 범주 비율은 1, 2, 3 이 각각 약 78%, 14%, 8% 정도로 불균형한 분포를 보이고 있음을 알 수 있다.

학습:검증:테스트 데이터셋의 비율은 검증 및 테스트셋의 데이터를 충분히 확보할 수 있는 범위 내에서 6:2:2 부터 8:1:1 까지로 분배할 수 있다. 현재 종속변수의 범주가 불균형하게 분포되어 있어, 8:1:1 로 분배할 경우 fetal_health=3 인 데이터가 검증 및 테스트 데이터셋에는 각각 약 17 개 정도만이 속하게 된다. 이는 모델의 성능을 평가하기에 충분치 않은 개수이므로, 학습:검증:테스트 데이터를 6:2:2 로 분배하여 가능한 한 fetal_health=3 인 데이터가 검증 및 테스트 데이터셋에 속할 수 있도록 하고자 한다. 학습:검증:테스트 데이터셋으로 분리할 때 종속변수 내 범주 비율은 동일하게 유지될 수 있도록 설정해주었다.

[Q2] (Decision Tree) 아래 두 가지의 경우에 대한 테스트 데이터셋에 대한 분류 성능을 평가하고 그 결과를 비교해보시오.

1) 학습 데이터만을 이용해서 학습한 Full Tree

1275 개의 학습 데이터를 이용하여 학습한 Full Tree 는 다음과 같은 구조를 가지고 있다.



[그림 1] Train Dataset 을 통해 학습된 Full Tree

테스트 데이터셋에 대한 Full Tree 의 성능을 평가한 결과는 다음과 같다.

[표 2] Full Tree 의 성능 평가 결과[1]

Model	TPR[2]	Precision[3]	TNR[4]	Accuracy[5]	BCR[6]	F1-Measure[7]	AUROC[8]
Full Tree	**0.824956**	**0.852537**	**0.858939**	**0.896714**	**0.817384**	**0.837565**	**0.863508**

모두 80% 이상의 성능을 보여주고 있어, 수치로만 보면 모델이 준수한 성능을 보여주고 있다고 할 수 있다. 그러나, Accuracy 와 BCR 이 약 8%의 성능 차이를 보여주고 있으며, 종속변수 내에 imbalance 가 존재한다는 사실을 고려하면, 각각의 성능지표에 대해 조금 더 면밀한 해석이 필요할 것으로 예상된다.

성능 지표 계산에 쓰인 Confusion Matrix 는 다음과 같다.

[표 3] Full Tree 의 Confusion Matrix

Confusion Matrix		Predicted		
		1(정상)	2(의심)	3(병리)
Actual	1(정상)	312	19	1
	2(의심)	19	40	0
	3(병리)	3	2	30

위 Confusion Matrixm 를 참고하여 성능 지표를 해석해보면 아래와 같이 해석할 수 있다.

- 실제 범주는 2 일 때, 1 로 잘못 예측된 경우가 거의 30%의 비율을 차지해, TPR 의 값이 Accuracy 보다 작아진 것을 알 수 있다.
- 마찬가지로, 예측한 범주가 2 일 때, 실제 범주는 1 혹은 3 인 경우가 1/3 을 차지하여 Precision 의 값이 Accuracy 보다 작은 것을 알 수 있다.
- 그러나, 실제 범주는 3 일 때 잘못 예측된 경우보다, 예측 범주가 3 일 때 실제와는 다른 경우가 더 적어 Precision 이 TPR 보다 큰 값을 가지게 된 것을 알 수 있다. 실제 범주가 1 일 때 잘못 예측한 경우보다, 예측 범주가 1 일 때 실제와 다른 경우가 조금 더 많았지만 정상 범주는 절대적인 숫자가 많아 비율 상으로는 거의 차이나지 않는다. 따라서 TPR 과 Precision 이 정상 범주보다 수가 적은 병리 범주에 크게 영향을 받기 때문에 차이가 생긴 것으로 해석할 수 있다.

---

[1] 본 데이터셋은 3 개의 Class 를 가진 Multi-Class Classification 데이터셋이므로, 하나의 Class 를 Positive Class 로, 나머지 Class 들을 Negative Class 로 설정하여 성능 지표들을 계산하였다.

[2] 실제 Positive Class 객체들 중에서 모델에 의해 Positive Class 로 정확히 예측된 비율이다.

[3] 모델이 Positive Class 로 예측한 객체들 중에서 실제 Positive Class 인 비율이다.

[4] 실제 Negative Class 객체들 중에서 모델에 의해 Negative Class 로 정확히 예측된 비율이다.

[5] 전체 객체들 중 범주와 무관하게 정확히 예측된 비율이다.

[6] 균형정확도로, 각 범주에 대한 정확도를 따로 계산한 후 기하평균을 취한 지표이다.

[7] 재현율과 정밀도의 조화 평균이다. 본 데이터셋은 3 개의 범주를 분류하는 경우이므로, 각 범주에 대한 재현율과 정밀도를 통해 각 범주에 대한 F1-Measure 를 구하고, 평균을 취한 Macro F1 을 사용하였다.

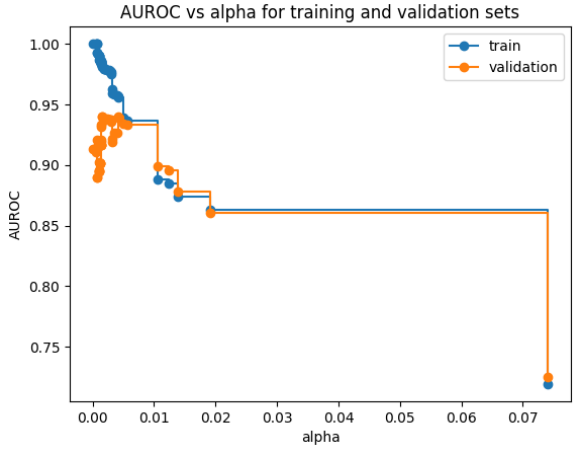
[8] ROC curve 아래의 면적으로, 1 에 가까운 값을 가질수록 성능이 좋다고 이야기할 수 있다.

- 두 범주의 데이터수를 분모로 가지다보니, 데이터수가 적은 범주의 영향을 비교적 덜 받아 TNR 이 TPR 보다는 높은 값을 가지게 된 것으로 보인다.
- BCR 과 F1-Measure 는 데이터수가 적은 범주에 대한 예측을 틀릴 경우, 데이터수가 많은 범주에 대한 예측을 틀릴 때보다 더 패널티를 받게 된다. 이는 분모가 작기 때문에 일어난다. 따라서 imbalance 가 존재하는 데이터의 경우, Accuracy 와 BCR/F1-Measure 값 사이에 차이가 생기게 된다. 현재 사용된 데이터가 imbalance 가 존재하는 데이터이므로, Accuracy 보다 BCR/F1-Measure 의 값이 낮은 것을 확인할 수 있다.
- AUROC 는 약 0.86 의 값을 가져, 모델이 높은 성능을 가지고 있음을 확인할 수 있다. 그러나 [그림 1]을 보면 Full Tree 의 depth 는 14 로, 한눈에 보기에도 과적합된 상태인 것을 파악할 수 있다. 따라서 Pruning 과정을 통해 과적합을 방지해야 한다.

2) 학습 데이터를 사용하여 학습한 후 검증 데이터를 사용하여 Post-pruning 을 수행한 Tree

Post-pruning 을 수행하기 위해서는 검증데이터에 대한 오분류율과 구조의 복잡도를 결합할 때 사용하는 가중치인 α를 설정해야 한다. 이때, α값이 높으면 구조는 단순해지나 전체 불순도가 높아지는 경향이 있어 적절한 α값을 선정할 필요가 있다.

cost_complexity_pruning_path 함수를 통해 α값과 불순도를 계산하며 Post-pruning 을 진행하였고, 계산된 α값에 따라 훈련, 검증 데이터셋에 대한 AUROC 를 비교한 그래프는 아래와 같다.[9]

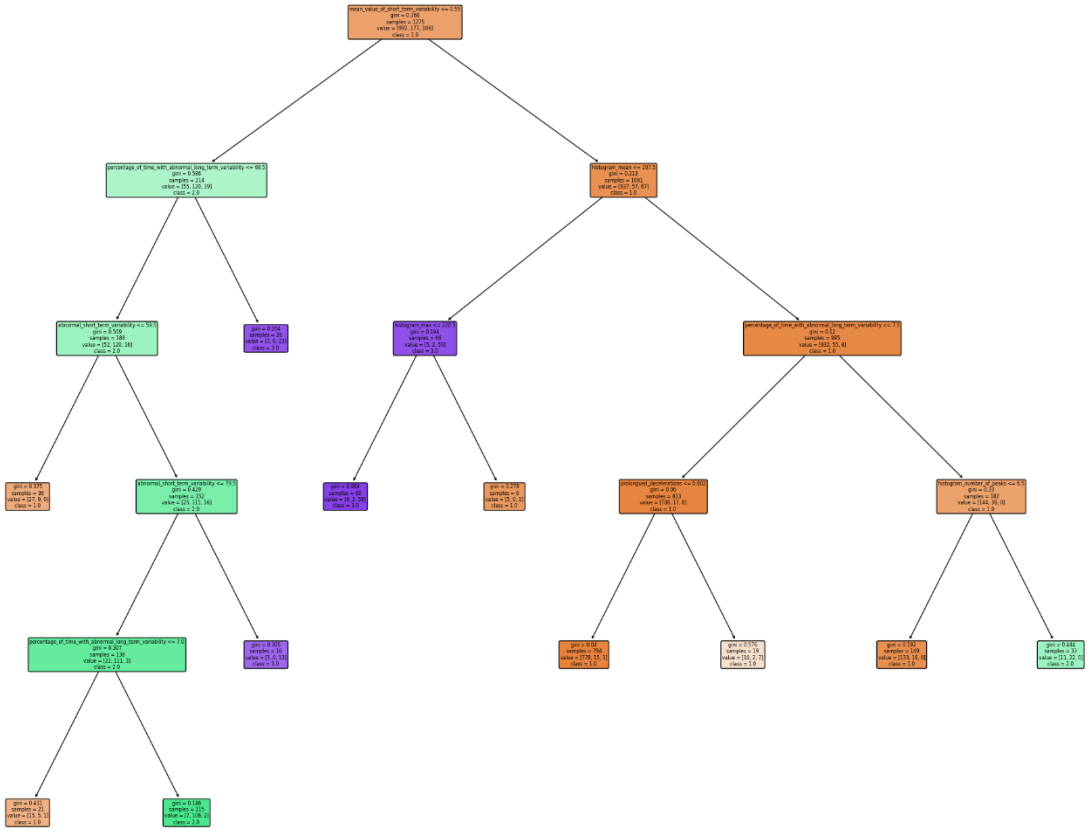


[그림 2] α값에 따른 훈련데이터와 검증데이터의 AUROC

α값이 작을 때 높은 정확도를 보이고 있음을 알 수 있다. 최종적으로 선택된 α는 0.004 정도의 값을 가지고 있다.

Post-pruning 을 거친 Tree 의 구조는 다음과 같다.

[9] 이후 문제들에서 AUROC 를 평가 지표로 사용하므로 α값을 결정할 때에도 AUROC 를 사용하였다.



[그림 3] Train Dataset 을 통해 학습된 Post-pruning Tree

depth 는 5 로, Full Tree 에 비해 복잡도가 낮아진 것을 알 수 있다.

테스트 데이터셋에 대한 Post-pruning Tree 의 성능을 평가한 결과는 다음과 같다.

[표 4] Post-pruning Tree 의 성능 평가 결과[10]

Model	TPR	Precision	TNR	Accuracy	BCR	F1-Measure	AUROC
Post-pruning	0.805653	0.898962	0.860602	0.913146	0.794381	0.846398	0.917399

Post-pruning Tree 역시 준수한 성능을 보여주고 있음을 확인할 수 있다. 그러나 데이터셋에 imbalance 가 존재하므로 Confusion Matrix 를 참고할 필요가 있다.

성능 지표 계산에 쓰인 Confusion Matrix 는 다음과 같다.

[표 5] Post-pruning Tree 의 Confusion Matrix

Confusion Matrix		Predicted		
		1(정상)	2(의심)	3(병리)
Actual	1(정상)	323	8	1
	2(의심)	21	38	0

[10] [표 2]와 마찬가지로 하나의 Class 를 Positive Class 로, 나머지 Class 들을 Negative Class 로 설정하여 성능 지표들을 계산하였다.

	3(병리)	6	1	28

- 실제 범주는 2 일 때, 1 로 잘못 예측된 경우가 약 36%의 비율을 차지해, TPR 의 값이 Accuracy 보다 작아진 것을 알 수 있다.
- 마찬가지로, 예측한 범주가 2 일 때, 실제 범주는 1 혹은 3 인 경우가 약 19%를 차지하여 Precision 의 값이 Accuracy 보다 작은 것을 알 수 있다.
- 실제 범주는 3 일 때 잘못 예측된 경우보다, 예측 범주가 3 일 때 실제와는 다른 경우가 더 적어 Precision 이 TPR 보다 큰 값을 가지게 된 것을 알 수 있다.
- 두 범주의 데이터수를 분모로 가지다보니, 데이터수가 적은 범주의 영향을 비교적 덜 받아 TNR 이 TPR 보다는 높은 값을 가지게 된 것으로 보인다.
- 현재 사용된 데이터가 imbalance 가 존재하는 데이터이므로, Accuracy 보다 BCR/F1-Measure 의 값이 낮은 것을 확인할 수 있다.
- AUROC 는 약 0.92 의 값을 가져, 모델이 높은 성능을 가지고 있음을 확인할 수 있다.

[표 6] Full Tree & Post-pruning Tree 의 성능 평가 결과

Model	TPR	Precision	TNR	Accuracy	BCR	F1-Measure	AUROC
Full Tree	0.824956	0.852537	0.858939	0.896714	0.817384	0.837565	0.863508
Post-pruning	0.805653	0.898962	0.860602	0.913146	0.794381	0.846398	0.917399

Full Tree 와 Post-pruning Tree 의 성능을 비교해보면 아래와 같다.

[표 3]과 [표 5]를 통해 Post-pruning Tree 가 정상과 병리를 조금 더 잘 맞추지만, 의심은 조금 더 많이 틀리는 것을 확인할 수 있었다. 따라서 지표에 따라 Full Tree 가 더 우수하다고 볼 수도, Post-pruning Tree 가 더 우수하다고 볼 수도 있다. 다만, 데이터셋에 imbalance 가 존재하고 보다 중요한 범주가 존재하므로 Accuracy 나 BCR 보다 F1-Measure 를 중요 지표로 판단할 것이다.
F1-Measure 와 AUROC 를 보면, Post-pruning Tree 가 Full Tree 보다 조금 더 높은 것을 확인할 수 있다. 이는 Pruning 과정을 통해 Full Tree 의 과적합 문제가 해소되어 모델의 일반화 능력이 높아졌기 때문으로 보인다.
뿐만 아니라, 두 모델의 성능이 큰 차이를 보이지 않기 때문에, 모델 복잡도 측면에서도 depth 가 5 인 Post-pruning Tree 가 depth 가 14 인 Full Tree 보다 선호된다.

[Q3] (Decision Tree) 학습 데이터와 검증 데이터를 이용하여 Pre-pruning 을 수행해보시오. Pre-pruning 을 수행하기 위해 사용된 하이퍼파라미터를 설명하고, 각 하이퍼파라미터마다 탐색 범위를 어떻게 설정했는지 서술하시오. 검증 데이터에 대한 AUROC 를 기준으로 최적의 하이퍼파라미터 조합을 찾아보시오.

Pre-pruning 을 수행하기 위해 사용한 하이퍼파라미터는 다음과 같다.

[표 7] Pre-pruning 을 수행하기 위해 사용된 하이퍼파라미터 종류

종류	범위	설명

criterion	**["gini", "entropy", "log_loss"]**	**분기 시 불순도를 측정하는 함수. "gini"는 Gini impurity 를, "log_loss"와 "entropy"는 Information gain 을 나타낸다.**
max_depth	**[3, 5, 7, 10, None]**	**Tree 의 최대 깊이. None 으로 설정할 경우 모든 leaf node 가 pure 하거나 모든 leaf node 가 min_samples_split sample 보다 적을 때까지 팽창한다.**
min_samples_leaf	**[5, 10, 20, 30, 50]**	**leaf node 에 요구되는 최소 샘플 수.**
min_samples_split	**[10, 30, 50, 70, 100]**	**내부 노드를 분할하기 위해 필요한 최소 샘플 수.**

각 하이퍼파라미터의 범위 선정 이유는 다음과 같다.

1) criterion

   DecisionTreeClassifier 에서 사용 가능한 모든 criterion 을 후보로 사용하였다.

2) max_depth

   앞서 보았듯이 Full Tree 의 depth 가 14 이고, Post-pruning Tree 의 depth 는 5 이다. 따라서 Full Tree 보다는 depth 가 작아야 하고, Post-pruning Tree 보다는 깊은 범위까지 탐색해볼 필요가 있다고 생각했다. max_depth 를 설정하지 않았을 때 최적의 모델이 만들어질 가능성 역시 존재하기 때문에 범위에 'None'도 추가하였다.

3) min_samples_leaf

   min_samples_leaf 가 너무 작으면 모델이 과적합될 가능성이 높다. 현재 데이터 개수는 1700 개[11]이므로, 위와 같이 범위를 설정하는 것이 적당하다고 판단했다.

4) min_samples_split

   min_samples_leaf 에 맞추어 min_samples_leaf 값의 2 배 이상이 되도록 설정하였다. 추가적으로 min_samples_split 가 min_samples_leaf 에 영향을 미치지 못하는 경우도 고려했다.

선정된 최적의 하이퍼파라미터 조합은 다음과 같다.

[표 8] 최적의 하이퍼파라미터 조합

종류	최적 값
criterion	**entropy**
max_depth	**7**
min_samples_leaf	**20**
min_samples_split	**10**

위 조합의 AUROC 값은 약 0.95 로, 매우 높은 것을 알 수 있다. min_samples_split 의 값이 min_samples_leaf 보다 작은 것을 통해 위 조합에서는 min_samples_split 의 값이 min_samples_leaf 에 영향을 미치지 못하는 것을 알 수 있다.

Pre-pruning Tree 의 Confusion Matrix 와 성능 평가 결과는 아래와 같다.

[11] GridSearchCV 함수는 실행 시 k-fold cross validation 을 진행하므로, 앞서 나눠둔 train dataset 과 validation dataset 을 병합하였다. 본 과정에서는 5-fold cross validation 으로 진행하였다.

[표 9] Pre-pruning Tree 의 Confusion Matrix

Confusion Matrix		Predicted		
		1(정상)	2(의심)	3(병리)
Actual	1(정상)	321	8	3
	2(의심)	18	40	1
	3(병리)	7	4	24

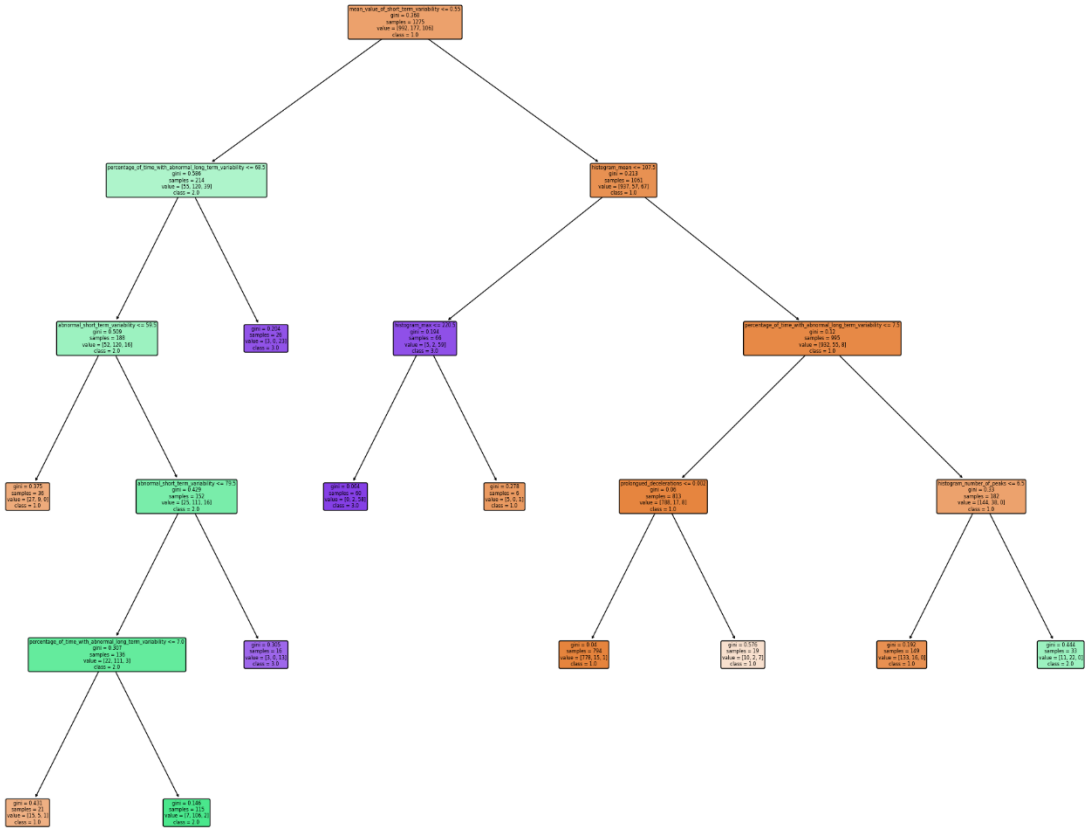
[표 10] Full Tree & Post-pruning Tree & Pre-pruning Tree 의 성능 평가 결과

Model	TPR	Precision	TNR	Accuracy	BCR	F1-Measure	AUROC
Full Tree	0.824956	0.852537	0.858939	0.896714	0.817384	0.837565	0.863508
Post-pruning	0.805653	0.898962	0.860602	0.913146	0.794381	0.846398	0.917399
Pre-pruning	0.776849	0.851373	0.848060	0.903756	0.766019	0.809843	0.950073

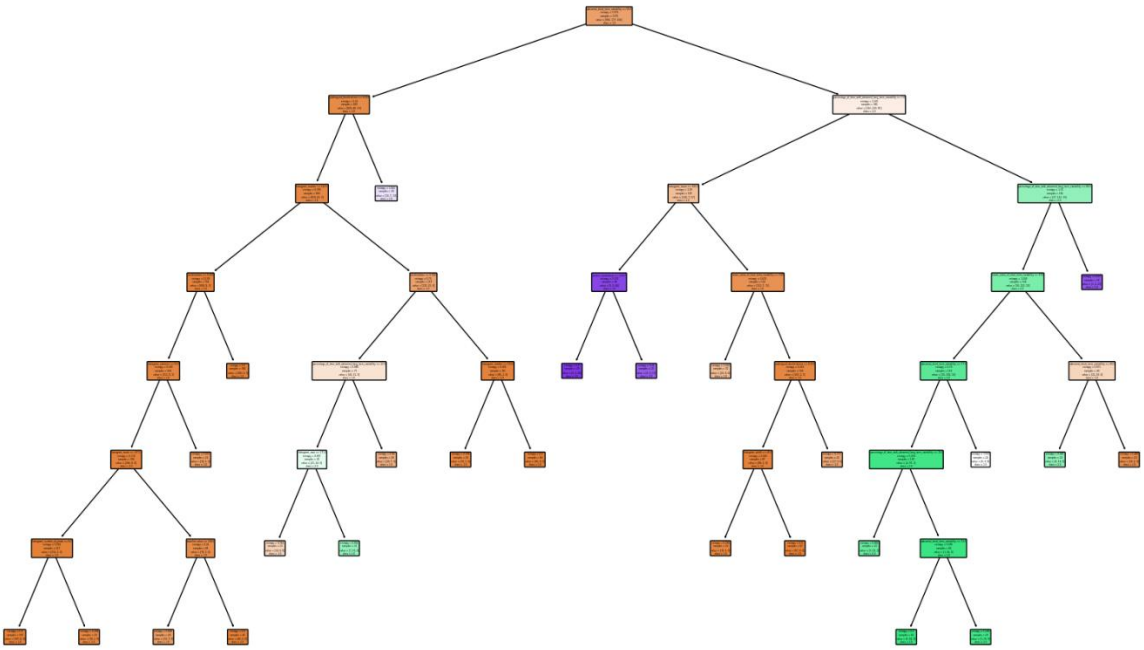
[표 3], [표 5], [표 9]를 통해 Pre-pruning Tree 가 Full Tree 와 Post-pruning Tree 보다 범주 3 을 잘 맞추지 못해 TPR 이 낮은 것을 확인할 수 있다. BCR 과 F1-Measure 역시 가장 낮은 것을 확인할 수 있다. 다만, AUROC 가 Full Tree 에 비해 약 9% 정도 더 높은 것이 눈에 띈다. 다른 모델에 비해 비교적 Pre-pruning Tree 가 cut-off 에 영향을 덜 받는다고 할 수 있다.

[Q4] (Decision Tree) [Q2]와 [Q3]에서 생성한 Post-pruning 모델과 Pre-pruning 모델의 결과물을 각각 Plotting 하고 이에 대한 해석을 수행하시오. 각 Pruning 방식에 따라 Split 에 사용된 변수는 어떤 변화가 있는가?

[다변량데이터분석][과제 3]



[그림 4] Train Dataset 을 통해 학습된 Post-pruning Tree



[그림 5] Train Dataset 을 통해 학습된 Pre-pruning Tree

먼저, 각 모델의 depth 와 leaf node 수는 다음과 같다.

[표 11] Full Tree & Post-pruning Tree & Pre-pruning Tree 의 depth 와 leaf node 수

Model	depth	number of leaf nodes
Full Tree	**14**	**80**
Post-pruning	**5**	**11**
Pre-pruning	**7**	**25**

Full Tree 에 비해 pruning 을 수행한 다른 모델들의 depth 가 더 작고, leaf node 의 수 역시 줄어들어 일반화 성능이 더 높을 것이라 기대할 수 있다. 추가로, 두 모델 다 Full Tree 에 비해 모델의 해석이 용이해진 것을 알 수 있다. Pre-pruning Tree 에 비해 Post-pruning Tree 의 복잡도가 더 낮은 것 역시 알 수 있다.

Post-pruning 과정과 Pre-pruning 과정에서 split 에 사용된 변수와 그 횟수는 다음과 같다.

[표 12] Pruning 과정에서 사용된 변수와 횟수

변수명	Post-pruning	Pre-pruning
baseline value	0	1
accelerations	0	2
fetal_movement	0	0
uterine_contractions	0	1
light_decelerations	0	0
severe_decelerations	0	0
prolongued_decelerations	1	2
abnormal_short_term_variability	2	4

mean_value_of_short_term_variability	1	2
percentage_of_time_with_abnormal_long_term_variability	3	4
mean_value_of_long_term_variability	0	0
histogram_width	0	1
histogram_min	0	0
histogram_max	1	1
histogram_number_of_peaks	1	1
histogram_number_of_zeroes	0	0
histogram_mode	0	0
histogram_mean	1	2
histogram_median	0	2
histogram_variance	0	1
histogram_tendency	0	0

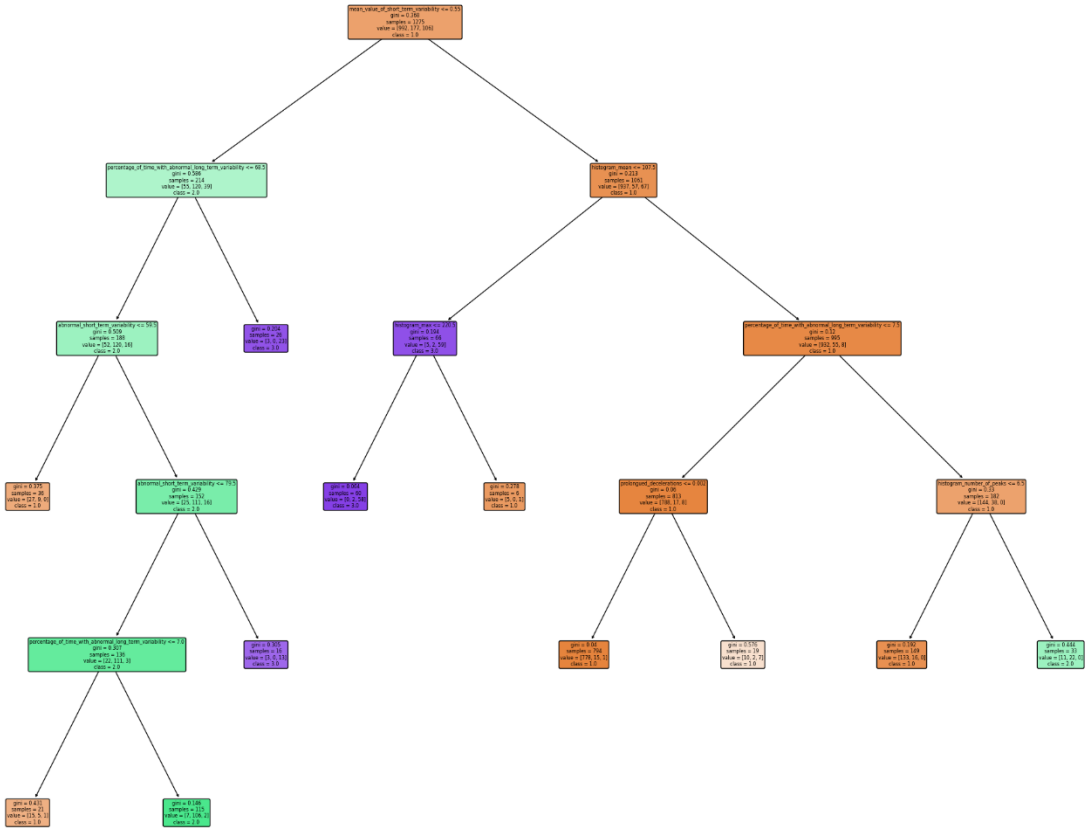
- Post-pruning 에서는 총 7 개의 변수가, Pre-pruning 에서는 총 13 개의 변수가 사용되었다. Post-pruning 에서 두번 이상 사용되었던 'abnormal_short_term_variability'와 'percentage_of_time_with_abnormal_long_term_variability'는 Pre-pruning 에서도 4 번씩 사용되어 종속변수의 범주를 예측하는데 중요한 역할을 하는 변수임을 알 수 있다.
- Post-pruning 에서는 'mean_value_of_short_term_variability'가, Pre-pruning 에서는 'abnormal_short_term_variability'가 가장 먼저 split 에 사용되어 중요한 변수임을 알 수 있다.
- Post-pruning 과 Pre-pruning 모두 두번째 분기에서 'percentage_of_time_with_abnormal_long_term_variability'가 기준 변수로 사용되나, Post-pruning 에서는 'percentage_of_time_with_abnormal_long_term_variability'가 68.5 이상인 경우, Pre-pruning 에서는 7.0 이하인 경우를 기준으로 split 된다. Pruning 방식에 따라 같은 변수를 기준으로 사용하더라도 splitting point 가 달라지는 것을 확인할 수 있다.
- Post pruning 은 'percentage_of_time_with_abnormal_long_term_variability' 이후 'abnormal_short_term_variability'를 기준 변수로 선택하고, Pre-pruning 은 'abnormal_short_term_variability'을 'percentage_of_time_with_abnormal_long_term_variability' 이전의 기준 변수로 선택한다. 따라서 Pruning 방식에 따라 선택되는 변수 순서에도 차이가 있음을 알 수 있다.

[Q5] (Decision Tree) 최적의 결정나무의 Plot 을 그리고, 대표적인 세 가지 규칙에 대해서 설명해보시오.

[다변량데이터분석][과제 3]

먼저 Full Tree 는 과적합된 Tree 이므로 후보에서 제외하고, 앞서 만든 Post-pruning Tree 와 Pre-pruning Tree 만을 비교할 것이다.[12]

[표 10]을 보면, Pre-pruning Tree 가 AUROC 는 높으나, 나머지 지표들이 Post-pruning Tree 에 비해 낮은 것을 알 수 있다. 그러나, 각 지표들의 차이가 유의미하게 크지 않아 단순 지표 비교로 최적의 결정나무를 정하는 것은 어려워 보인다. 따라서 [표 11]과 [표 12]를 참고할 필요가 있다. 성능이 비슷한 경우, 조금 더 단순한 모델을 선택하는 것이 비용과 해석 면에서 용이하다. 따라서 더 적은 변수를 사용하고, 더 단순한 구조를 가진 Post-pruning Tree 를 최적의 Tree 로 선정하는 것이 옳아 보인다.



[그림 6] Train Dataset 을 통해 학습된 Post-pruning Tree

위 Tree 의 대표적인 세 가지 규칙은 다음과 같다.

1) 'mean_value_of_short_term_variability' ≥ 0.55 & 'histogram_mean' ≥ 107.5 & 'percentage_of_time_with_abnormal_long_term_variability' ≤ 7.5 & 'prolongued_decelerations' ≤ 0.002 이면 class = 1 이다.

[12] 최적의 하이퍼파라미터를 선정하는 과정을 통해 만들어진 Tree 이기 때문에 추가적으로 Tree 를 생성하는 것은 의미가 없다고 판단했다.

2) 'mean_value_of_short_term_variability' ≥ 0.55 & 'histogram_mean' ≤ 107.5 & 'histogram_max' ≤ 220.5 이면 class = 3 이다.
3) 'mean_value_of_short_term_variability' ≤ 0.55 & 'percentage_of_time_with_abnormal_long_term_variability' ≤ 68.5 & 'abnormal_short_term_variability' ≥ 59.5 & 'abnormal_short_term_variability' ≤ 79.5 & 'percentage_of_time_with_abnormal_long_term_variability' ≥ 7.0 이면 class = 2 이다.

[Q6] (Neural Network) 동일한 데이터셋에 대하여 Neural Network 학습을 위해 필요한 최소 3 가지 이상의 하이퍼파라미터를 선정하고, 각 하이퍼파라미터마다 최소 3 개 이상의 후보 값(최소 9 가지 조합)을 사용하여 grid search 를 수행한 뒤, 검증데이터에 대한 AUROC 기준으로 최적의 하이퍼파라미터 조합을 찾아보시오.

최적의 하이퍼파라미터 조합을 찾기 위해 사용한 하이퍼파라미터의 종류와 범위는 다음과 같다.[13]

[표 13] 사용된 하이퍼파라미터 종류와 범위

종류	범위	설명
hidden_layer_sizes	[(5,), (10,), (15,), (20,), (5,5), (10,10), (15,15), (20,20)]	hidden layer 내에 있는 hidden neuron 의 개수. i 번째 요소는 i 번째 hidden layer 를 나타낸다.
activation	['identity', 'logistic', 'tanh', 'relu']	hidden layer 에 적용되는 activation function.
learning_rate_init	[0.001, 0.01, 0.1]	초기 학습률. 가중치 업데이트 시 step-size 를 제어한다.
max_iter	[100, 300, 500]	반복의 최대 횟수. slover 가 수렴할 때까지 혹은 max_iter 만큼 반복하며, 확률적 slover 의 경우, max_iter 는 반복 횟수가 아니라 epoch 수를 결정함.

각 하이퍼파라미터의 범위 선정 이유는 다음과 같다.

1) hidden_layer_sizes
   입력변수의 차원과 동일하거나 더 낮게 설정함으로써 과적합을 방지하고자 하였다. 일반적으로 hidden layer 는 3 개 이상 사용하여도 성능이 높아지지 않으며, 각 hidden layer 의 node 수는 동일하게 설정해주므로 범위를 위와 같이 설정하였다.
2) activation
   MLPClassifier 에서 사용 가능한 activation function 전부를 후보로 선정하였다.
3) learning_rate_init
   learning_rate_init 이 너무 작으면 학습이 너무 느려 수렴까지 오래 걸린다는 단점이 존재하고, learning_rate_init 이 너무 크면 gradient 가 발산할 수 있다는 단점이 존재한다. 따라서 일반적으로 많이 사용하는 0.001 부터 조금 더 큰 값까지 후보로 선정하였다.

[13] grid search 를 수행하기 전, 먼저 StandardScaler()를 통해 입력변수의 정규화를 진행하였다.

4) max_iter

learning_rate_init = 0.1 일 때 gradient 가 수렴할 것이라고 가정하면, max_iter = 100 정도로 설정해도 충분히 수렴할 수 있을 것이라고 판단하여 최솟값을 100 으로 설정하고, 더 높은 값을 추가해주었다.

선정된 최적의 하이퍼파라미터 조합은 다음과 같다.

[표 14] 최적의 하이퍼파라미터 조합

종류	최적 값
hidden_layer_sizes	(20, 20)
activation	relu
learning_rate_init	0.01
max_iter	300

위 조합의 AUROC 값은 약 0.97 로 매우 높은 성능을 보이고 있다.

[Q7] (Decision Tree/Neural Network 공통) [Q3]에서 선택한 최적의 Pre-pruning Decision Tree 모델과 [Q6]에서 선택한 최적의 Neural Network, 그리고 로지스틱 회귀분석을 사용하여 학습 데이터를 학습한 뒤, 테스트 데이터에 적용한 결과를 아래의 Confusion Matrix 와 같이 작성하고 이에 대한 결과를 해석해 보시오.

Neural Network 와 Logistic Regression 의 Confusion Matrix 와 성능 평가 결과는 아래와 같다.

[표 15] Logistic Regression 의 Confusion Matrix

Confusion Matrix		Predicted		
		1(정상)	2(의심)	3(병리)
Actual	1(정상)	318	14	0
	2(의심)	23	33	3
	3(병리)	3	7	25

[표 16] Neural Network 의 Confusion Matrix

Confusion Matrix		Predicted		
		1(정상)	2(의심)	3(병리)
Actual	1(정상)	324	8	0
	2(의심)	15	42	2
	3(병리)	1	6	28

[표 17] Logistic Regression & Decision Tree & Neural Network 성능 평가 결과

Dataset	Model	TPR	TNR	Accuracy	BCR	F1-Measure
Fetal_health	Logistic Regression	0.743813	0.816441	0.882629	0.726007	0.772850
	Decision Tree	0.776849	0.848060	0.903756	0.766019	0.809843
	Neural Network	0.829256	0.879957	0.924883	0.822176	0.852086

- [표 9], [표 15], [표 16]을 참조하였을 때, Neural Network 는 세 범주 모두에서 정답을 가장 많이 맞힌 것을 알 수 있다. 이는 모든 지표에서 Neural Network 가 가장 뛰어난 것과 일맥상통한다.
- 전체적으로 2 번 범주의 데이터를 잘 예측하지 못하는 경향을 보인다.
- 세 모델 모두 Accuracy 에 비해 TPR 과 TNR 이 낮은 것을 확인할 수 있다. 이는 2, 3 번 범주의 데이터수가 작아, 해당 범주의 데이터를 잘못 예측할 경우 정답 비율이 확연히 낮아지기 때문이다.
- 세 모델 모두 Accuracy 보다 BCR, F1-Measure 의 값이 작다. 이는 데이터셋의 imbalance 로 인한 것으로 볼 수 있다.
- 데이터셋에 imbalance 가 존재하고, 보다 중요한 범주(2, 3 번 범주)가 존재하므로 Accuracy, BCR 보다는 F1-Measure 를 주요 평가 지표로 활용하는 것이 좋다. 따라서 가장 높은 F1-Measure 값을 가진 Neural Network 의 성능이 가장 뛰어나다고 할 수 있다.
- 본 데이터셋이 선형으로 데이터들을 구분하기에는 어려움이 있는 조금 복잡한 형태의 데이터셋일 것으로 예상된다. 따라서 Logistic Regression 이나 Decision Tree 는 분류 경계면을 정하는데 제약이 있어, 아무런 제약이 없는 Neural Network 가 가장 좋은 성능을 보였을 것이라 추측된다.
- 성능 지표만으로 판단하였을 때에는 Neural Network 가 가장 뛰어난 성능을 보였으나, 본 데이터셋이 "예측 결과물에 대한 해석"이 중요한 데이터셋이므로 성능 차이가 크지 않고, 해석이 보다 용이한 Decision Tree 를 최종 모델로 선정하는 게 더 좋을 수 있다.

[Q8] 이번에는 본인이 생각하기에 "예측 정확도"가 "예측 결과물에 대한 해석"보다 훨씬 더 중요할 것으로 생각되는 분류 문제를 다루고 있는 데이터셋을 1 개 선정하고 선정 이유를 설명하시오. 이 외 가이드라인은 [Q1]의 가이드라인과 동일합니다.

선정한 데이터셋은 "Apple Quality"로, 사과의 특징을 통해 품질을 판단하는 데이터셋이다. 사과를 재배하는 사람에게는 어떠한 근거로 해당 사과의 품질이 판단되었는지도 중요한 요소이지만, 사과를 판매하는 소매업자 등은 사과의 품질 판단 이유보다는 정확한 품질 예측이 중요하다. 따라서 위 데이터셋은 "예측 결과물에 대한 해석"도 중요하지만, "예측 정확도"가 더욱 중요할 것이라 생각했다.

링크: https://www.kaggle.com/datasets/nelgiriyewithana/apple-quality

주요 변수들은 다음과 같다.

[표 18] Fetal Health Dataset 변수 설명

변수명	설명
A_id	각 과일의 고유 식별자
Size	과일의 크기
Weight	과일의 무게

Sweetness	과일의 단맛 정도
Crunchiness	과일의 아삭함을 나타내는 질감
Juiciness	과일의 즙이 많은 정도
Ripeness	과일의 성숙 단계
Acidity	과일의 산도 수준
Quality(Target Variable)	과일의 전체적인 품질

총 8 개의 설명변수와 1 개의 종속변수로 이루어져 있으며, 전체 데이터 개수는 4001 개이다.[14] 종속변수 내 범주 비율은 good:bad 가 5:5 정도로 균등한 분포를 보이고 있다.

학습:검증:테스트 데이터셋의 비율은 검증 및 테스트셋의 데이터를 충분히 확보할 수 있는 범위 내에서 6:2:2 부터 8:1:1 까지로 분배할 수 있다. 현재 종속변수의 범주가 균등하게 분포되어 있으며, 높은 예측 정확도를 갖추는 것이 목적이므로 가능한 한 많은 데이터를 학습에 사용하고자 한다. 따라서 데이터셋의 비율은 8:1:1 로 분배할 것이다.

[Q9] (Decision Tree/Neural Network 공통) [Q8]에서 선택한 데이터셋을 사용하여 [Q3]에서 수행한 최적의 pre-pruning Decision Tree 모델 찾기, [Q6]에서 수행한 최적의 Neural Network 모델 찾기를 동일하게 수행하시오.

1) 최적의 Pre-pruning Decision Tree 모델 찾기

Decision Tree 는 전처리 과정이 필요하지 않으나, Neural Network 를 고려하여 전처리를 진행하였다. 먼저, 결측치가 존재하는 1 개의 관측치를 제거하고, 학습에 필요하지 않은 "A_id" 변수를 삭제하였다. 추가로, StandardScaler 를 통해 입력변수의 정규화를 진행해주었다.

[표 19] Pre-pruning 을 수행하기 위해 사용된 하이퍼파라미터 종류

종류	범위	설명
criterion	**["gini", "entropy", "log_loss"]**	**분기 시 불순도를 측정하는 함수. "gini"는 Gini impurity 를, "log_loss"와 "entropy"는 Information gain 을 나타낸다.**
max_depth	**[7, 10, 15, None]**	**Tree 의 최대 깊이. None 으로 설정할 경우 모든 leaf node 가 pure 하거나 모든 leaf node 가 min_samples_split sample 보다 적을 때까지 팽창한다.**
min_samples_leaf	**[10, 20, 30, 40, 50]**	**leaf node 에 요구되는 최소 샘플 수.**
min_samples_split	**[10, 30, 50, 70, 100]**	**내부 노드를 분할하기 위해 필요한 최소 샘플 수.**

각 하이퍼파라미터의 범위 선정 이유는 다음과 같다.

1) criterion

   DecisionTreeClassifier 에서 사용 가능한 모든 criterion 을 후보로 사용하였다.

2) max_depth

   Full Tree 와 Post-pruning Tree 를 학습시킨 결과, Full Tree 의 depth 는 21, Post-pruning Tree 의 depth 는 18 이었다. 그러나 Post-pruning Tree 의 leaf node 는 289 개로, 아직

[14] 결측치가 존재하는 1 개의 sample 을 제외한 숫자이다.

많은 leaf node 를 가진 것을 확인할 수 있었다. 따라서 일반화 성능을 더 높이고자 최대 depth 가 두 Tree 보다는 작도록 설정해주었다.

3) min_samples_leaf

min_samples_leaf 가 너무 작으면 모델이 과적합될 가능성이 높다. 현재 데이터 개수는 3600 개[15]이고, Full Tree 의 leaf node 가 383 개이므로 min_samples_leaf 는 10 이상의 값을 가져야 한다고 판단했다.

4) min_samples_split

min_samples_leaf 에 맞추어 min_samples_leaf 값의 2 배 이상이 될 수 있도록 하였다. 추가적으로 min_samples_split 가 min_samples_leaf 에 영향을 미치지 못하는 경우도 고려했다.

선정된 최적의 하이퍼파라미터 조합은 다음과 같다.

[표 20] 최적의 하이퍼파라미터 조합

종류	최적 값
criterion	**entropy**
max_depth	**10**
min_samples_leaf	**10**
min_samples_split	**30**

위 조합의 AUROC 값은 약 0.76 으로, 꽤 높은 성능을 보여주고 있다. depth = 10, leaf node 의 수 = 101 로, Full Tree 와 Post-pruning Tree 에 비해 depth 와 leaf node 의 수 모두 많이 줄어든 것을 확인할 수 있다.

2) 최적의 Neural Network 모델 찾기

최적의 하이퍼파라미터 조합을 찾기 위해 사용한 하이퍼파라미터의 종류와 범위는 다음과 같다.

[표 21] 사용된 하이퍼파라미터 종류와 범위

종류	범위	설명
hidden_layer_sizes	**[(3,), (5,), (7,), (3,3), (5,5), (7,7)]**	**hidden layer 내에 있는 hidden neuron 의 개수. i 번째 요소는 i 번째 hidden layer 를 나타낸다.**
activation	**['identity', 'logistic', 'tanh', 'relu']**	**hidden layer 에 적용되는 activation function.**
learning_rate_init	**[0.001, 0.01, 0.1]**	**초기 학습률. 가중치 업데이트 시 step-size 를 제어한다.**
max_iter	**[100, 300, 500]**	**반복의 최대 횟수. slover 가 수렴할 때까지 혹은 max_iter 만큼 반복하며, 확률적 slover 의 경우, max_iter 는 반복 횟수가 아니라 epoch 수를 결정함.**

각 하이퍼파라미터의 범위 선정 이유는 앞서 보았던 [Q6]와 동일하다.

선정된 최적의 하이퍼파라미터 조합은 다음과 같다.

[15] GridSearchCV 함수는 실행 시 k-fold cross validation 을 진행하므로, 앞서 나눠둔 train dataset 과 validation dataset 을 병합하였다. 본 과정에서는 5-fold cross validation 으로 진행하였다.

[표 22] 최적의 하이퍼파라미터 조합

종류	최적 값
hidden_layer_sizes	(7, 7)
activation	tanh
learning_rate_init	0.01
max_iter	300

위 조합의 AUROC 값은 약 0.88 로, 높은 성능을 보이고 있다.

[Q10] (Decision Tree/Neural Network 공통) [Q9]에서 선택된 최적의 Pre-pruning Decision Tree 모델과 최적의 Neural Network, 그리고 로지스틱 회귀분석을 사용하여 학습 데이터를 학습한 뒤, 테스트 데이터에 적용한 결과를 아래의 Confusion Matrix 와 같이 작성하고 이에 대한 결과를 해석해 보시오. 데이터셋 선정 당시 본인의 예상과 [Q7]의 결과표 및 아래 결과표가 일치하는지 확인해보시오. 일치하지 않는다면 왜 일치하지 않는지 그 이유를 서술해보시오.

각 모델의 Confusion Matrix 와 성능 평가 결과는 다음과 같다.

[표 23] Logistic Regression 의 Confusion Matrix

Confusion Matrix		Predicted	
		0 (good)	1 (bad)
Actual	0 (good)	157	44
	1 (bad)	61	138

[표 24] Decision Tree 의 Confusion Matrix

Confusion Matrix		Predicted	
		0 (good)	1 (bad)
Actual	0 (good)	155	46
	1 (bad)	51	148

[표 25] Neural Network 의 Confusion Matrix

Confusion Matrix		Predicted	
		0 (good)	1 (bad)
Actual	0 (good)	179	22
	1 (bad)	26	173

[표 26] Logistic Regression & Decision Tree & Neural Network 성능 평가 결과

Dataset	Model	TPR	TNR	Accuracy	BCR	F1-Measure
Fetal_health	Logistic Regression	0.693467	0.781095	0.7375	0.735978	0.724409
	Decision Tree	0.743719	0.771144	0.757500	0.757307	0.753181
	Neural Network	0.869347	0.890547	0.880000	0.879883	0.878173

- 학습에 사용된 입력변수가 7 개로 적은 편이라, [Q7]의 모델들보다 높은 성능을 보여줄 것이라 생각했다. 그러나 Logistic Regression 과 Decision Tree 는 [Q7]보다 더 낮은 성능을 보여주었다. 이는 "Apple Quality" 데이터셋의 관측치들이 "Fetal Health"

데이터셋보다 특정 범주들로 분리되기 어렵기 때문으로 보인다. 또, Full Tree 를 생각해보면, "Fetal Health" 데이터셋을 학습한 Full Tree 의 depth 와 leaf node 수가 "Apple Quality" 데이터셋을 학습한 Full Tree 의 depth 와 leaf node 수보다 작았다. Pruning 을 수행해도 마찬가지다. 이는 "Apple Quality" 데이터셋의 입력 변수들은 모두 범주를 예측하는 데 있어 중요한 변수이나, "Fetal Health" 데이터셋의 입력 변수들 중에는 범주를 예측하는 데 있어 중요하지 않은 변수가 여럿 포함되어 있어 결국 사용되는 입력 변수 수가 비슷해지는 것도 하나의 원인이 될 수 있을 것 같다.

- Confusion Matrix 를 확인해보아도, 한 모델 내에서 각 범주에 대한 예측을 틀린 비율이 비슷한 것으로 보아, 분류 경계면 근방에서 두 범주의 관측치들이 서로 뒤섞여 있었을 것으로 추정된다.
- Neural Network 는 Accuracy 외에는 전부 더 높은 성능을 보여주었다. [Q7]의 데이터셋은 불균형해서 Accuracy 보다는 F1-Measure 로, [Q10]은 균등하게 분포되어 있어 Accuracy 로 성능을 판단한다. 그러나, 서로 다른 두 지표를 1 대 1 로 비교할 수는 없으므로 두 모델이 비슷한 성능을 지닌 것으로 판단할 것이다. 이 결과는 "예측 정확도"가 중요한 "Apple Quality" 데이터셋이 "Fetal Health" 데이터셋에 비해 높은 예측 정확도를 보여줄 것이라고 기대했던 나의 예상과는 조금 다른 결과이다. 생각해보면, "예측 정확도"는 데이터셋에, "예측 결과물에 대한 해석"은 모델에 달린 것이므로 이러한 결과가 나올 수 있다.
- Neural Network 는 비선형으로 분류 경계면을 설정할 수 있으므로, 분류 경계면에 제약이 있는 Logistic Regression, Decision Tree 보다 높은 정확도를 보여줄 것이라 생각했다. Logistic Regression, Decision Tree 보다 Neural Network 가 12~15% 정도 높은 Accuracy[16]를 보여주고 있어, 예상과 결과가 일치한다. 또한, "Apple Quality" 데이터셋이 선형보다는 비선형으로 분류 경계면을 설정하는 것이 유용하다는 것을 알 수 있다.
- [Q7]보다 적은 입력변수를 사용하였기 때문에 Decision Tree 의 복잡도가 더 낮을 것으로 예상하였다. 그러나 예상과는 달리, Pruning 과정을 거쳐도 매우 복잡한 Tree 가 형성되어, 해석이 용이하다는 장점이 사라진 것처럼 보인다.

[16] 본 데이터셋은 균등하게 분포된 데이터셋이므로 Accuracy 를 주요 지표로 사용한다.