2024 - 1

다변량분석

Decision Tree & Neural Network

담당 교수: 강필성

강의명: 다변량분석

제출자: 정동은

전공: 경영학과

학번: 2020120120

제출 날짜: 2024-05-28

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

1.1. 데이터 셋

이름: Cardiovascular Disease dataset

사이트: Kaggle

링크:https://www.kaggle.com/datasets/sulianova/cardiovascular-disease dataset?datasetId=107706&searchQuery=eda

1.2. 선정 이유

선정한 데이터셋은 심혈관 질환 여부와, 심혈관 질환과 관련성이 있을 수 있는 정보들을 모아놓은 데이터셋이다. 질환을 치료할 때는 어떤 요소가 원인이 되어 질환을 야기시켰는지 중요하지 않을 수 있다. 가장 시급한 문제는 질환을 치료하는 것이기 때문이다.

그러나 해당 질환을 지녔는지 판단할 때는 원인 요소를 파악하는 것이 매우 중요하다. 질환을 지녔음에도 없다고 판단하거나, 질환을 지니지 않았음에도 있다고 판단하면 한 사람의 인생이 달라질 수 있기에 최대한 구체적으로 원인 요소를 파악하여 종합적으로 판단하는 것이 중요하기 때문이다. 이러한 이유로 어떤 요소가 원인이 되어 심혈관 질환을 유발하였는지에 관한 '예측 결과물에 대한 해석'이, 선정한 데이터셋에서는 중요할 것이라고 판단하였다.

나아가 해당 데이터셋을 더욱 엄밀히 분석할 수 있다면 심혈관 질환 여부 판단을 넘어 심혈관 질환 예방으로까지(dsecriptive→predictive) 나아갈 수 있는, 인류사에 있어 중요한 과업이기에 데이터셋을 선정하게되었다.

1.3.데이터 구성

데이터의 변수는 독립변수 11개와 종속변수 1개로 구성되어있다. 독립변수는 연속형 변수 5개, 범주형 변수 6개로 구성되어있다. 각 독립변수의 설명은 다음과 같다.

번호	변수명	속성	설명
1	age	연속형	환자의 나이
2	height	연속형	환자의 신장
3	weight	연속형	환자의 몸무게
4	gender	범주형	환자의 성별
5	ap_hi	연속형	수축기 혈압을 의미
6	ap_lo	연속형	확장기 혈압을 의미
7	cholesterol	범주형	콜레스테롤 수치를 의미
8	gluc	범주형	포도당을 의미
9	smoke	범주형	흡연 여부를 의미
10	alco	범주형	음주 여부를 의미
11	active	범주형	신체 활동여부를 의미

종속형 변수는 다음과 같다.

cardio: 심혈관 질환 여부를 의미하는 이진(0, 1) 범주형 변수이다.

본 데이터셋의 독립변수는 데이터 획득 방법에 따라 다음과 같은 추가정보를 제공한다.

종 류	설 명	독립변수
Objective	factual information	age, height, weight, gender
Examination	results of medical examination	ap_hi, ap_lo, cholesterol, gluc
Subjective	information given by the patient	smoke, alco, active

(가이드라인) 해당 데이터셋에 대해서 학습:검증:테스트 용도로 적절히 분배하시오(예: 60:20:20). 본인이 분배한 비율에 대해서 간략히 근거를 설명하시오. 분류 성능을 평가/비교할 때는 TPR, TNR, Precision, Accuracy, BCR, F1-Measure, AUROC를 복합적으로 고려하여 서술하시오.

1.4. 학습:검증:테스트 분배

분배는 test, validation set이 original 데이터셋의 변동성을 최대한 잘 대변할 수 있도록 분배하여야한다. 데이터 셋의 크기가 작을수록, test, validation set의 비율을 늘린다는 속설이 있으나 데이터 셋의 크기가 작다는 기준이 명확하지 않고, 근거를 바탕으로 엄밀한 분석을 진행하고 싶어 다음 문헌을 참고하여 데이터를 분배하였다.

참고문헌: https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/)

필자의 개인적인 생각으로 데이터셋의 크기가 작거나, 클래스 값이 불균형하다면 분산이 증가하기에 original 데이터의 변동성을 잘 대변하지 못할 확률이 높다고 생각한다. 하지만 선정한 데이터셋은 크기가 70,000이고 종속변수의 값 0,1의 비율도 50:50정도로 매우 비슷하기에 분배 방법에 관계없이 큰 차이가 없을 것으로 예상하였다. 그럼에도 아래의 세 가지 시나리오를 바탕으로 어떤 시나리오가 Baseline (original)과 validation set에서 가장 유사한 타겟 변수의 비율을 지녔는지 실험해보았다. 실험결과는 난수(seed)별 영향을 최소화하고자 서로다른 난수를 바탕으로 30번의 반복실험을 진행한 후, 평균 값을 계산하여 결과를 도출하였다.

	타겟 변수 1의 비율 (%)
Baseline (분배 전)	50.030%
80/10/10	48.361%
70/15/15	48.361%
60/20/20	48.361%

실험 결과 세 가지 시나리오 모두 validation set에서 동일한 타겟 변수의 비율을 보였다. 이는 앞서 예상한 바와같이 데이터의 크기가 크고, 기존 타겟 변수 1의 비율이 50.030%로, 타겟 변수 1이 다른 타겟 변수 0과 거의 동일한 비율로 존재하기에 나온 결과라고 해석하였다. 그럼에도 세 가지 시나리오 중 하나를 선택하여야하기에, 60/20/20 비율을 선택하였다. 그 이유는 다음과 같다. 검증 데이터를 바탕으로 K-fold 검증을 수행할 때 K-1 부분은 train으로, 1 부분은 test로 나누어 진행한다. 이는 검증 데이터를 또 K 분할한다는 뜻이니, 검증 데이터 속에서 분할된 1 부분은 상

대적으로 적은 데이터만을 지니고 있음을 의미한다. 따라서 과적합 방지 및 검증 데이터의 적절한 확보를 위해 60/20/20의 비율을 선택하였다.

참고로 stratify 값을 설정하지 않았을때의 80/10/10에서 타겟 변수 0의 비율은 48.310%이고, stratify 값을 설정하였을 때의 80/10/10에서 타겟 변수 1의 비율은 48.361%이다. 따라서 stratify 값을 True로 설정하여 타겟변수를 최대한 Baseline과 유사하게 분배되도록 하였다.

Q2.

(Decision Tree) 아래 세 가지의 경우에 대한 테스트 데이터셋에 대한 분류 성능을 평가하고 그 결과를 비교해보시오.

2.1. 학습 데이터만을 이용해서 학습한 Full Tree

Full Tree 모델의 Training/Test 데이터에 대한 평가 결과는 다음과 같다.

	Training Data	Test Data
Accuracy	0.980	0.642
BCR	0.980	0.641
AUROC	0.980	0.642
F1-Score	0.980	0.642
Precision	0.996	0.637
Recall	0.965	0.638
TPR	0.965	0.642
TNR	0.996	0.643

Full Tree는 불순도가 0이 될때까지 가지를 뻗어나가기에 Training Data의 모든 평가 지표의 성능이 0.965 이상의 결과가 나왔다. Full Tree는 max depth 등의 하이퍼파라미터에 대한 아무런 제약 없이, 오직 불순도 최소화만을 기반으로 전개된다. 따라서 Training data의 Noisse까지 학습하여 과적합될 확률이 증가한다. 위의 실험 결과에서도 볼 수 있듯이 Test data의 평가지표들은

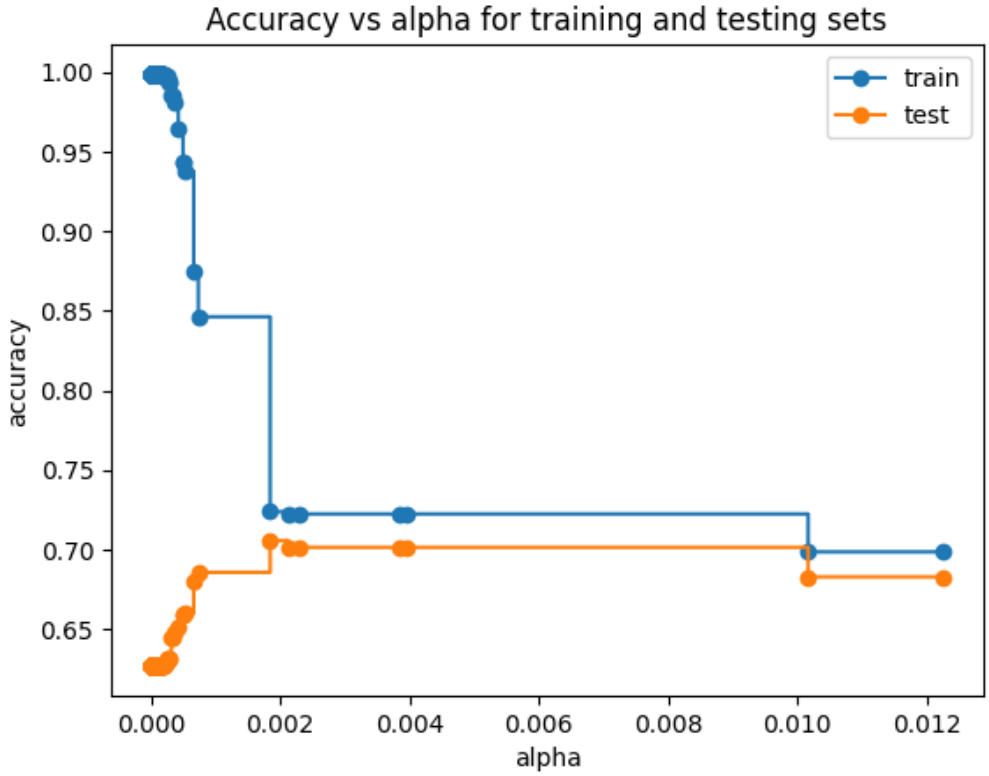
Training data와 달리 0.637 ~ 0.642 사이에 위치해있고, Training data의 평가지표들과는 차이가 많이 난다. 이를 통해 Full Tree가 과적합되어있음을 확인할 수 있다.

2.2. Post-pruning을 수행한 Tree

Post-pruning에서 complexity 개선을 위해 적절한 alpha값 설정을 먼저 진행하였다. 70,000개 데이터를 모두 사용하여 코드를 실행해보았는데, 2시간이 넘도록 결과물이 출력되지 않았다. 빠른 실행을 위해 1/20로 축소된 sample인 3,500개 데이터만을 바탕으로 적절한 alpha값을 먼저 찾았고, 해당 alpha를 바탕으로 모델을 학습시켰다.

alpha가 크다는 의미는, ipmurity의 개선이 커야만 Tree를 split한다는 의미이다. Tree의 split이 어려워지면 분기된 node가 줄어들기에 # of nodes는 줄어들고, 이와 마찬가지로 Tree의 depth도 작아질 것이다.

FullTree는 train data에 최적화된 형태이기에 accuracy가 1에 육박한다. 하지만 FullTree는 noise도 함께 학습하였기에, 앞선 결과에서 확인하였듯이 test data의 관점에서는 overffitting 되어있는 형태이다. 따라서 그래프 및 아래 표에서 확인할 수 있듯이 test data에서의 accuracy는 좋지 않다. alpha를 키울수록 split되는 노드가 줄어들고 overfitting이 점점 줄어드는 것을, 아래의 그림에서 train과 test의 격차가 줄어드는 것을 통해 확인할 수 있다. 최종적으로 train과 test의 성능이 가장 유사하면서, 좋은 성능을 발휘하는 alpha는 0.093으로 확인되었다.



alpha는 0.093으로 Post-pruning을 수행한 tree의 성능은 다음과 같다.

	Test Data
Accuracy	0.715
BCR	0.715
AUROC	0.715
F1-Score	0.738
Precision	0.683
Recall	0.803
TPR	0.803
TNR	0.628

Q3.

(Decision Tree) 학습 데이터와 검증 데이터를 이용하여 Pre-pruning을 수행해보시오. Pre-pruning을 수행하기 위해 사용된 하이퍼파라미터를 설명하고, 각 하이퍼파라미터마다 탐색 범위를 어떻게 설정했는지 서술하시오. 검증 데이터에 대한 AUROC를 기준으로 최적의 하이퍼파라미터 조합을 찾아보시오.

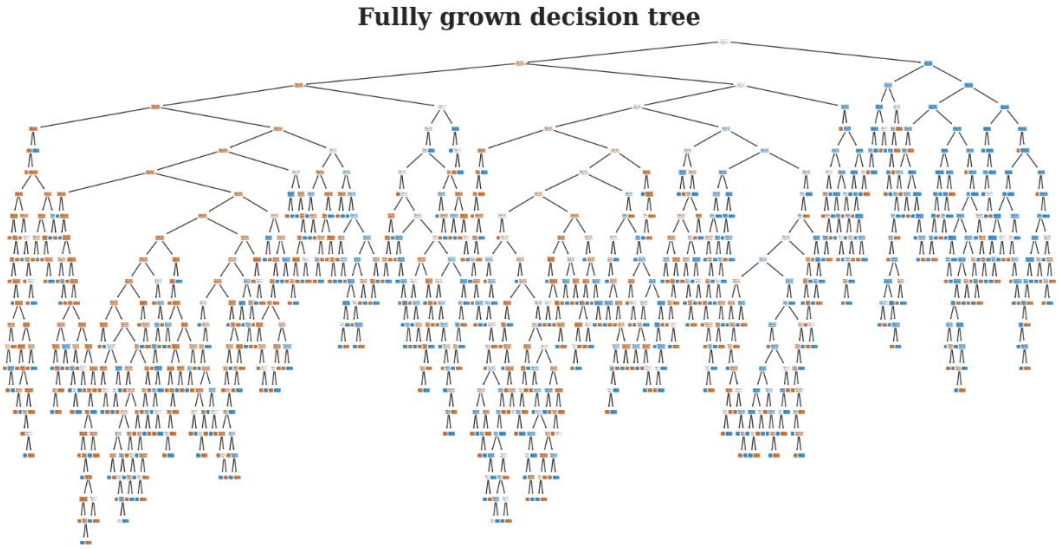
3.1. 하이퍼파라미터 설정

Pre-pruning에서 설정 가능한 하이퍼파라미터는 다음과 같다.

하이퍼파라미터	설 명
max_depth	Decision Tree의 최대 깊이를 제한
min_samples_split	노드를 분할하는데 필요한 최소 샘플 개수
min_samples_leaf	Leaf 노드에 존재하는 최소 샘플 개수
max_leaf_nodes	최대 Leaf 노드 개수

하이퍼파라미터 값을 어떻게 조정할지는 데이터의 성격, 형태에 따라 매우 다르다. 따라서 먼저 데이터를 제대로 이해하기위해 Full Tree의 전개 방식을 그림을 통해 살펴보았다.

그림을 그릴때 기존의 70,000개 데이터를 모두 사용하니, 1시간이 넘도록 결과물이 출력되지 않았다. 따라서 3,500개 데이터만을 바탕으로 그림을 그렸고, 이는 기존 데이터 대비 1/20 로 축소된 sample로 결과물을 출력하였음을 의미한다. Full Tree의 전개 방식을 표현한 결과물은 다음과 같다.



Full Tree의 전개 방식을 살펴본 이유는 Pre-pruning 하이퍼파라미터 설정을 위해서이다. Tree의 크기와 데이터의 성격에 따라 Pre-pruning에 적용할 하이퍼파라미터는 천차만별로 값이 달라지기 때문이다. 3,500개의 데이터로 출력한 4가지 하이퍼파라미터에관한 통계값은 다음과 같다.

파라미터	최소값	최대값	평균
Depth	4	23	13.28
# of samples in Leaf node	1	59	3.83
# of Leaf node	549		
# of samples for split	2	2100	45.38

위의 결과값은 70,000개의 데이터중 1/20로 축소된 3,500개 데이터만을 사용하여 학습한 결과이다. 따라서 위 값을 보정해줘야한다. 보정은 특정한 알고리즘이 아니라, 필자의 개인적인 논리로 진행하였다.

Depth는 데이터의 개수에 강건하다고 생각한다. 최대값은 작은 leaf node들 때문에 변동성이 클 수 있겠지만, 큰 줄기까지의 평균 depth는 일정한 규칙하에서 전개되었다고 생각하기 때문이다. 즉 극단값을 제외한 depth의 평균은 일정 경향을 띈다고 생각하였다. 또한 depth는 max값이 궁금한 것이고, 3,500개 데이터에서는 max depth=23임을 확인하였다. max depth에는 sample을 1개 근방으로 지닌 leaf node도 포함되어있다. max depth를 설정하는 이유는 sample을 1개 근방으로 지닌 과적합되어있는 leaf node를 제거하기위해 설정하는 것이기에, 70,000개의 데이터에서 최대 25 ~ 30정도의 보정된 max depth를 설정해도 줄기의 흐름을 놓치는 경우가 없을 것이라고 판단하였다.

# of samples in Leaf node는 최대 59개까지 지니고 있었고, 이를 70,000개의 데이터로 환산하면 20 x 59 = 1,180이다. 마찬가지 방법으로 평균도 환산하면 평균 134개의 sample을 지니고 있다. 하지만 하이퍼파라미터에서 설정하고자하는 값은 최소 몇 개 지니고 있어야 하는지에 관한 것이므로, 35 ~ 134사이의 값들을 하이퍼파라미터로 설정하였다.

# of leaf node는 549개였고, 70,000개의 데이터일 경우에는 이보다 훨씬 많아질 것이다. 그러나 그중 대부분은 sample 수가 1개 근방으로 overfitting을 유발하는 leaf node일 것이므로 고려하지 않아도 된다고 판단하였다. 즉 70,000개의 데이터를 사용할 경우에, leaf node는 엄청 많아질 것으로 예상하지만 그중 대부분은 overfitting을 유발하는 leaf node일 것이라고 추측하였기에, 70,000개의 데이터에서도 실질적으로 의미있는 leaf node는 549개와 큰 차이가 없을 것이라는 의미이다. 따라서 하이퍼파라미터로 설정한 최대 leaf node는 549로 설정하였다.

# of samples for split은 최소값이 2였고, 이를 70,000개의 데이터로 환산하면 2 x 20 = 40이다. 하지만 40보다는 커야 하이퍼파라미터를 설정하여 overfitting을 방지하는 효과를 볼 수 있기에 40보다 크게 설정한다.

70,000개의 데이터로 보정된 하이퍼파라미터별 최대(최소) 가능값은 다음과 같다.

하이퍼파라미터	대소비교	최대(최소) 가능값
max_depth	Full Tree > Pre-pruning	30(최대)
min_samples_split	Full Tree < Pre-pruning	40(최소)
min_samples_leaf	Full Tree < Pre-pruning	35(최소)
max_leaf_nodes	Full Tree > Pre-pruning	549(최대)

위의 대소비교 범위내에서 Pre-pruning에 사용할 하이퍼파라미터 값의 범위를 설정하였다. 또한 불순도 계산방법에 따라 차이가 존재할 수 있기에, 각각 다른 계산방법을 사용하는 criterion 하이퍼파라미터도 추가하였다. 최종적으로 pre-pruning에 사용한 하이퍼파라미터 5가지이고, 설정값은 다음과 같다.

하이퍼파라미터	설정값
criterion	gini, entropy, log_loss
max_depth	10, 20, 30, None

min_samples_split	40, 60, 100, None
min_samples_leaf	35, 40, 50, None
max_leaf_nodes	450, 500, 549

grid search(cv=5)를 통해 하이퍼파라미터 설정값의 모든 조합을 검증 데이터의 AUROC를 기준으로 비교해보았고 결과는 다음 장에서 언급한다.

3.2. 하이퍼파라미터 조합 성능 비교

Grid search를 통해 총 576가지 조합을 확인하였고, 조합의 검증 데이터에 대한 AUROC 통계값은 다음과 같다.

종류	AUROC 값
Maximum AUROC	0.794
Minimum AUROC	0.785
Average AUROC	0.788

AUROC의 Maximum, Minimum, Average AUROC 값에 큰 차이가 없다. 변동성이 적은 이유는 많은 양의 데이터로 학습을 해서, Rule이 강건하게 적용된 것이 효과적이었다고 생각한다. 추가로 criterion 하이퍼파라미터를 제외한 4개의 하이퍼파라미터로 확인해보았는데, 이때는 Average AUROC가 0.778로 criterion을 사용하였을 때보다 Average AUROC가 0.1 줄어들었다. 신뢰구간을 확인해보지않아 확신할 수 없지만, 어떤 하이퍼파라미터를 사용하느냐에 따라 성능 차이가 발생한다고 예상할 수 있다.

576가지 조합중 AUROC 측면에서 최적의 하이퍼 파라미터 조합은 다음과 같다.

criterion: log_loss, max_depth: 10, max_leaf_nodes: 450,

min_samples_leaf: 50, min_samples_split: 100

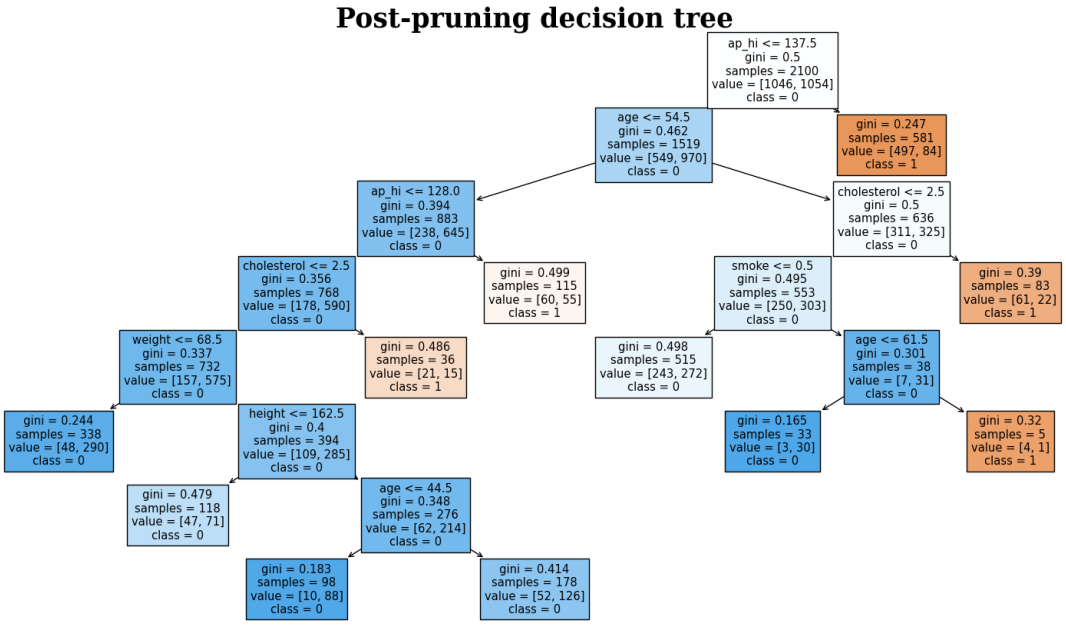
Q2에서 살펴본 Full Tree의 AUROC는 0.714이었고, 이는 Test data 기준이었다. 최적의 하이퍼파라미터를 바탕으로 전개한 Pre-pruning은 Validation data에서의 AUROC만을 확인하였다. Q2와 마찬가지 조건하에서 성능을 비교하기위해, 최적의 하이퍼파라미터 조합을 Test data로 AUROC를 측정해보았고, Test data의 AUROC는 0.730이었다. 즉 Pre-pruning은 Full Tree보다 Test data를 기준으로 더 좋은 성능을 발휘한다. 이는 overfitting 문제를 해결, 즉 noise의 학습을 최소화하였기에 Training data와 다른 noise를 지닌 Test data에서 좋은 성능을 발휘한 것으로 생각한다.

Q4.

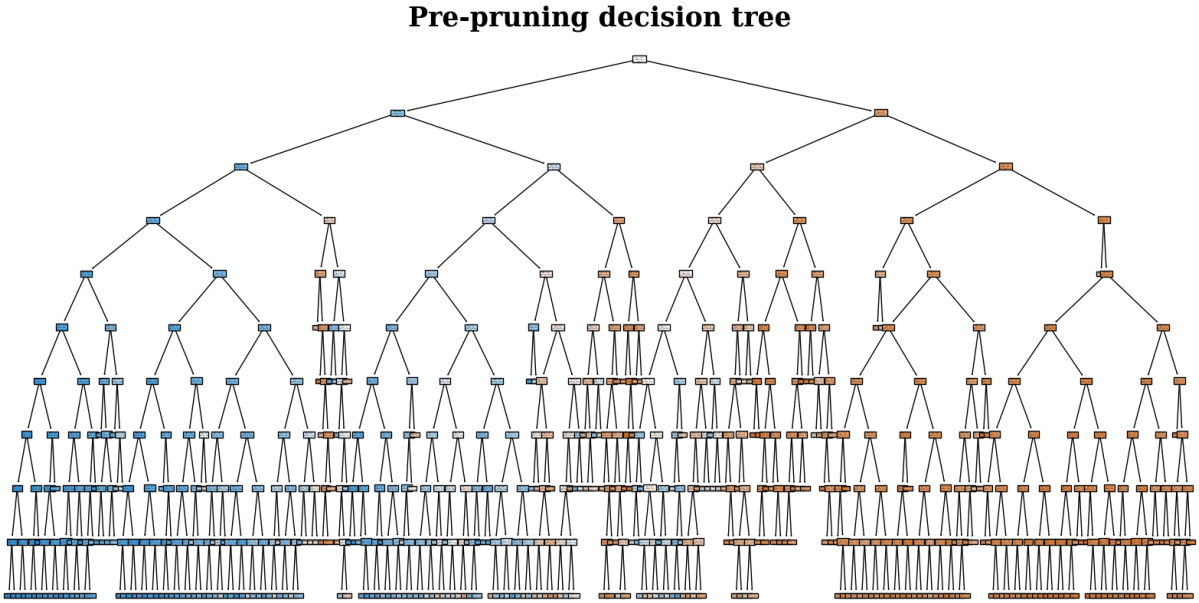
(Decision Tree) [Q2]와 [Q3]에서 생성한 Post-pruning 모델과 Pre-pruning 모델의 결과물을 각각 Plotting하고 이에 대한 해석을 수행하시오. 각 Pruning 방식에 따라 Split에 사용된 변수는 어떤 변화가 있는가?

4.1. 결과물 Plotting

Post-pruning의 결과물은 다음과 같다.



최적의 하이퍼파라미터 조합을 바탕으로 전개한 Pre-pruning 결과물은 다음과 같다.



4.2. 결과물 해석

언뜻 보기에도 Post-pruning과 Pre-pruning의 결과물은 확연히 달라보인다. 그렇다고 Pre-pruning에서 가지치기를 하지 않은 것은 절대 아니다. 하이퍼파라미터 범위 설정을 위해 3,500개의 데이터만 사용한 FullTree에서의 max depth가 23인데 반해, 70,000개의 데이터를 모두 활용한 Pre-pruning에서의 max depth는 10인 것을 통해, 가지치기가 수행되었음 확인할 수 있다.

그럼 왜 두 Tree는 확연히 다른 전개방식을 보일까? 그 이유는 다음과 같다고 생각한다. Post-pruning에서는 연산을 통해 최적의 alpha를 찾은 후 학습을 진행한데 반해, Pre-pruning에서는 필자의 논리를 바탕으로 하이퍼파라미터 범위를 설정한 후, 제한된 설정값 중에서 grid search를 통해 선택한 것이다. 즉 Post-pruning과 Pre-pruning에서의 하이퍼파라미터 설정 방법을 다르게 채택하였고, Post-pruning은 전역 탐색을 진행한데 반해, Pre-pruning은 그렇지 못하였기에 다른 결과가 나왔다고 생각한다.

Post-pruning에서 사용된 변수는 ap_hi, age, cholesterol, weight, height, smoke로 총 11개의 독립변수 중 6개가 사용되었다. 처음 분기는 ap_hi로 진행되었고, 가장 많은 sample을 지닌 leaf node는 ap_hi만으로 판단되었다. 따라서 ap_hi만으로 가장 많은 sample을 구분할 수 있다고 말할 수 있다. age는 연속변수이기에 구분할 수 있는 부분이 많아서인지, 2번 이상의 분기 기준으로 사용되었다는 특징을 보인다.

Pre-pruning은 그림을 통해 세부적인 내용조차 식별할 수 없을 정도로 넓고 깊게 분기되었다. 확연히 차이나는 너비와 깊이를 통해 분기 기준의 수가 많음을 확인할 수 있다. 분기 기준이 많아진다면 분기에 사용된 변수 수가 증가하였을 것이라고 유추할 수 있지만, 소수의 변수가 중복되어 사용되었을 수 있기에 단정지을 수는 없다.

정량적 지표로 분석을 진행하고자 feature_importances_ 내장함수를 통해 각 변수의 중요도를 확인해보았다. 참고로 feature_importances_는 다음 기준을 바탕으로 값을 계산한다.

1. 분기 기준에서 변수를 사용한 횟수: 변수가 Decision Tree에서 분기 기준으로 사용된 횟수

2. 변수의 중요도: 변수의 사용 횟수를 해당 변수가 사용된 총 횟수로 나눈 비율로 계산

즉, 특정 변수가 더 많이 사용되면서 모델의 예측에 더 중요한 역할을 하는 경우 해당 변수의 중요도가 높게 산출된다. 반대로, 특정 변수가 거의 사용되지 않거나 예측에 중요한 역할을 하지 않는 경우 해당 변수의 중요도는 낮게 산출된다.

Post-pruning, Pre-pruning에서의 각 독립변수의 중요도는 다음과 같다.

변수명	Post-pruning에서의 중요도	Pre-pruning에서의 중요도
age	0.146	0.126
gender	0	0.003
height	0.016	0.013
weight	0.022	0.028
ap_hi	0.728	0.727
ap_lo	0	0.012
cholesterol	0.069	0.074
gluc	0	0.007
smoke	0.019	0.003
alco	0	0.00
active	0	0.008

* 0은 중요도가 0을 의미, 0.00은 0은 아니지만 소수점 둘째자리까지의 반올림하면 0이 나오는 값을 의미

Post-pruning에서는 총 6개의 독립변수가 사용되었고, Pre-pruning에서는 총 11개의 독립변수가 모두 사용되었다. 각 변수의 중요도는 비슷한 추세를 보인다. ap-Hi의 중요도는 0.728, 0.727로 두 모델 모두에서 중요한 변수임을 확인할 수 있다. 다른 변수에서도 비슷한 양상을 보인다. 그럼에

도 Pre-pruning에서 더 많은 변수가 사용된 이유는, 사용자가 임의로 하이퍼파라미터 범위를 설정하였기에, Tree의 과적합을 Post-pruning보다는 방지하지못해 Tree가 넓고 깊게 전개된 이유에서라고 생각한다. 예를 들어 깊이 2에서 stop해도 괜찮은 Rule일 수 있음에도 깊이 5까지 전개된 Rule들이 Pre-pruning에서 상대적으로 많았을 것이라는 의미이다.

4.3. Pre-pruning & Post-pruning 결과 비교

평가지표별 성능 차이가 궁금하여, 성능도 확인해 보았다. Pre-pruning과 Post-pruning의 성능은 다음과 같다.

	Pre-pruning	Post-pruning
Accuracy	**0.714**	0.710
BCR	**0.714**	0.710
AUROC	**0.714**	0.710
F1-Score	0.704	**0.738**
Precision	**0.728**	0.673
Recall	0.682	**0.817**
TPR	0.682	**0.817**
TNR	**0.746**	0.603

Q5.

(Decision Tree) 최적의 결정나무의 Plot을 그리고, 대표적인 세 가지 규칙에 대해서 설명해보시오.

최적의 Decision Tree는 사람마다 판단하는 기준이 다를 것이다. 그러나 필자는 Post-pruning에서 얻은 Decision Tree가 확인한 Decision Tree 중에서 가장 최적이라고 판단한다. 그 이유는 성능 평가지표를 통해 설명한다. (성능을 비교하기전에 Pre-pruning에서 얻은 성능이 Grid Search를 통해 확인한 Pre-pruning 조합 중 가장 최적이라고 가정한다. 가정을 하는 이유는 Validation set에서의 AUROC만으로 해당 Pre-pruning 하이퍼파라미터를 선정하였기에, AUROC 이외의 기준으로는 어떤 조합이 최적인지 비교하지 못하였기 때문이다)

	Pre-pruning	Post-pruning
Accuracy	**0.714**	0.710
BCR	**0.714**	0.710
AUROC	**0.714**	0.710
F1-Score	0.704	**0.738**
Precision	**0.728**	0.673
Recall	0.682	**0.817**
TPR	0.682	**0.817**
TNR	**0.746**	0.603

8개의 평가지표중 상대적으로 Pre-pruning에서 5개, Post-pruning에서 3개의 평가지표에서 좋은 성능을보였다. 개수로만보면 Pre-pruning이 더 좋은 성능을 발휘하였다고 생각할 수 있지만, 어떤 평가지표에서 각 모델이 좋은 성능을 발휘하였는지도 함께 고려하여야한다. 여기서 잠깐, 데이터셋 선정 기준을 다시 살펴보자.

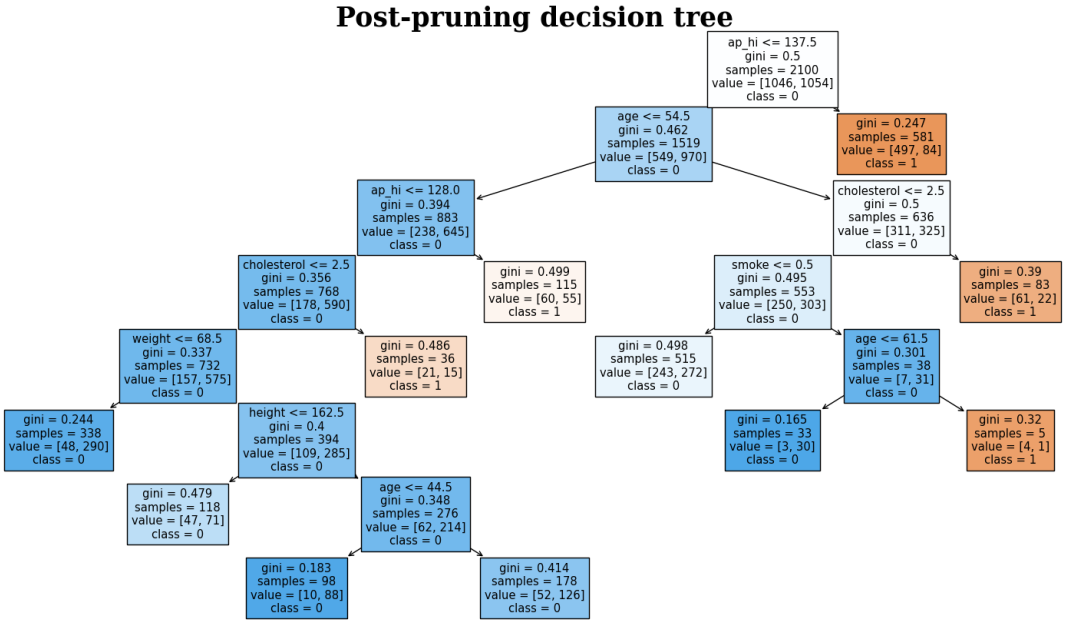
"예측 정확도"도 중요하지만 "예측 결과물에 대한 해석"이 매우 중요할 것으로 생각되는 분류 문제를 다루고 있는 데이터셋을 1개 선정

그럼 위의 주제에 부합하는 평가지표는 무엇일까? 개인적으로는 F1-Score, AUROC라고 생각한다.

F1-Score는 Precision과 Recall의 조화평균, 즉 양성으로 예측한 데이터 중 실제 양성과, 실제 양성인 데이터중 양성으로 정확하게 예측한 데이터의 적절한 조화평균을 의미한다. F1-Score가 중요하다고 판단한 이유는 아무리 Accuracy가 높아도 매우 작은 위음성이 엄청난 타격을 주는 상황이라면, Accuracy보다는 위음성을 최대한 줄이는 것이 더 중요할 수도 있기 때문이다. 선정한 데이터는 환자의 심혈관 질환 유무를 판단하는 데이터이고, 잘못된 진단은 한 사람의 운명을 바꿀 수도 있기에 위음성을 최대한 줄이는 것이 더 중요하다고 판단하였다. F1-Score는 Post-pruning이 0.738로, Pre-pruining보다 0.034 더 높은 수치로, 좋은 성능을 발휘하였다.

마찬가지 이유로 **AUROC**도 중요하다고 생각하는데 이는 Pre-pruning에서 0.714로 더 좋은 성능을 발휘하였다. 그러나 Post-pruning보다 0.004더 높은 수치로, F1-Score에서의 차이(0.034)보다 확연히 작다. 따라서 해당 데이터에서는 Post-pruning을 최적의 Decision Tree라고 판단하였다.

Post-pruning Decsion Tree의 결과물은 다음과 같다.



대표적인 규칙은 Leaf node 중 가장 많은 samples을 포괄할 수 있는 상위 3개의 Leaf node의 규칙으로 선정하였다. 이유는 가장 많은 samples을 포괄할 수 있다는 의미는 noise를 학습한 규칙보다는 범용적인 규칙에 가깝다고 판단할 수 있기 때문이다.

규칙 1: ap_hi >137.5 이면 class = 1로 예측

ap_hi는 Systolic blood pressure을 의미하고 이는 수축기 혈압을 의미한다. 수축기 혈압이 137.5면

고혈압에 속하고, 고혈압일수록 혈관이 막히기 쉽다는 걸 의미하니 심혈관 질환을 지니고 있다고(class=1) 판단하는 규칙이다. 이는 우리의 경험적으로도 이해가능한 부분이다.

규칙 2: ap_hi <=137.5 → age > 54.5 → cholesterol <= 2.5 → smoke <= 0.5 이면 class = 0으로 예측

규칙 1과 다르게 수축기 혈압은 낮으면서, 54.5세 초과이고, 콜레스테롤 수치는 2.5이하이고 담배는 피지 않으면 심혈관 질환을 지니고 있지 않다고(class=0) 판단하는 규칙이다. 수축기 혈압이 낮고, 콜레스테롤 수치가 낮고, 담배를 피지 않음은 심혈관 질환 확률을 낮춘다고 상식선에서 납득가능한 부분이다. 다만 54.5세 초과인 부분은 의아하여, 54.5세 이하인 규칙들을 살펴보니 압도적으로 class=0으로 나아가는 규칙들이 많았다. 즉, 54.5세 초과라 심혈관 질환을 지니고 있지 않다고 판단하였다는 것이 아니라, 54.5세 이하라도 충분히 심혈관 질환을 지니고 있지 않다고 판단하는 규칙들이 존재한다는 것이다. 다만 규칙 2에서는 수축기 혈압이 낮으며, 콜레스테롤 수치가 낮고, 담배를 피지 않는다는 요인이 곁들여지면서 심혈관 질환을 지니고 있지 않는다는 규칙이 생성된 것이다.

규칙 3: ap_hi <=137.5 → age <= 54.5 → ap_hi <= 128.0 → cholesterol <= 2.5 → weight <= 68.5 이면 class = 0으로 예측

규칙 2에서 의아하였던 부분은 규칙 3이라는 하나의 사례를 통해 해결된다. 규칙 3은 수축기 혈압이 137.5보다 낮고, 54.5세 이하이면서, 그 중 수축기 혈압이 또 128보다 낮으면서 콜레스테롤 수치는 2.5 이하이면서 몸무게도 68.5kg 이하이면 심혈관 질환을 지니고 있지 않다고(class=0) 판단하는 규칙이다. 규칙 3의 모든 요인들은 심혈관 질환을 낮춘다고 상식선에서 공감되는 부분들이다. 수축기 혈압이 낮으면 고혈압 확률이 낮고, 젊을수록 건강할 확률이 높고, 콜레스테롤 수치가 낮을수록 혈관이 건강하고, 또한 몸무게가 적을수록 혈관이 막힐 확률이 적기에 규칙 3은 쉽게 납득가능한 부분이다.

Q6.

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

6.1. NeuraNetwork에서의 하이퍼파라미터

NeuralNetwork에서 조절가능한 하이퍼파라미터는 다음과 같다.

하이퍼파라미터	설 명
Learning rate	수렴 속도와 학습 과정의 안정성 조절
Activation Functions	비선형성 도입을 위한 근사 함수
# of Hidden layers	네트워크의 복잡한 표현을 가능하게 해줌
# of Hidden nodes	복잡한 패턴을 학습
Optimizer	가중치 업데이트 방식을 결정
Batch Size	업데이트에 필요한 샘플 수 결정

위의 하이퍼파라미터 모두 Grid Search를 시도하기에는 조합이 너무 많다. 하이퍼파라미터 선별이 필요하고, 다음 기준을 바탕으로 선별을 진행하였다. **1.** 모델을 실행시키기 전에는 결과를 예측하기 어렵고, **2.** 하이퍼파라미터 조합별 변화가 클 것으로 예상되는 하이퍼파라미터만 선택하여 진행하였다. Optimizer는 각 수렴방법에 따라 데이터별로 다른 효과를 낼 수 있지만, 일반적으로 Adam이 가장 좋은 성능을 발휘한다고 널리 알려져있기에 Adam을 선택, Batch Size는 클수록 효과가 좋다고 알려져있다. 따라서 실험을 진행하지 않고 어떤 설정값이 좋은 성능을 발휘할 수 있는지 예측 가능한 Optimzer, Batch Size는 제외하고, 남은 4개의 하이퍼파라미터에대해 실험을 진행하였다.

최종적으로 변화시킬 하이퍼파라미터는 Learning rate, Activation Functions, # of Hidden layers, # of Hidden nodes이다. **# of Hidden nodes**는 NN에서의 line의 개수를 의미하고 이는 데이터별로 차이가 크기에 K-fold를 통해서 hidden nodes의 수에 따른 성능을 먼저 확인해보았다. Grid Search와 함께 탐색도 가능하지만, hidden nodes의 범위가 늘어날수록 Grid Serarch의 조합은 기하급수적으로 늘어나고, 이는 학습 시간의 엄청난 증가를 의미하기에 hidden nodes만으로 먼저 성능을 확인해보았다. K fold=5, max iteration = 300으로 설정하고 실험을 진행하였다.

실험은 다음과 같이 이루어졌다. # of hidden nodes는 5개부터 100개까지 5단위로 증가시키며 총 20개를 확인해보았다. AUROC 값을 바탕으로 hidden nodes가 몇 개일때 AUROC 값이 높을지 아래의 결과를 통해 확인해보았다. 즉, 'hidden nodes가 몇 개일때 좋은 성능을 발휘할까'를 바탕으로 Grid search에서 설정할 hidden node 값의 범위를 설정한다는 의미이다. AUROC 기준 상위 10개의 hidden nodes 수와 AUROC 값은 다음과 같다.

# of hidden nodes	AUROC
65	0.788
50	0.786
85	0.786
30	0.785
95	0.785
5	0.784
25	0.784
55	0.784
35	0.784
40	0.784

순위에따라 AUROC 값의 차이가 크지 않다. 또한 hidden nodes의 범위가 특정 부분에 집중되어 있는 것이 아니라, 폭넓게 분포되어있다. 예를 들어 65(1위) ~ 85(5위)의 범위만 보아도 hidden nodes 수가 20이나 차이가 난다는 의미이다. 즉 hidden nodes 수에 따라 성능 차이가 별로 없었기에 Grid search에서 hidden node는 65 ~ 85사이의 값들을 10간격으로 설정하였다.

다른 하이퍼파라미터에 대한 설명은 다음과 같다.

Learning rate가 크면 속도는 빠르지만 수렴하지 못할 가능성이, 작으면 너무 많은 학습 시간이 걸린다는 단점이 있다. 일반적으로 0.1 이하의 학습률을 사용한다고 수업시간에 배웠고, 따라서 0.01, 0.05, 0.1의 세 가지값을 가능한 학습률로 설정하였다.

Activation Functions은 Relu가 가장 일반적이라고 알려져있는데, 이는 layer가 많은 딥러닝 기준이다. Relu는 기울기 소실 문제를 해결하기위해 나왔고, 기울기 소실은 딥러닝에서 많은 layer를 만들면서 생긴 문제였는데, 본 과제에서는 Hidden layer를 3개 이하로 적용할 것이므로 Relu를 꼭 사용할 필요는 없다. 따라서 Sigmoid와 Relu를 모두 고려하였고, Tanh는 중앙값이 0이라 Sigmoid에서 발생하는 편향 이동이 발생하지 않는다. 또한 sigmoid보다 학습 효율성이 뛰어나다고 알려져있어 추가하였다.

# of Hidden layers는 다음과 같이 설정하였다. 딥러닝에서는 hidden layer가 매우 많지만 딥러닝 이전의 NeuralNet, 즉 프로젝트에서 사용하는 NN에서의 hidden layer는 많아야 3개이다. 따라서 1, 2, 3의 Hidden layer를 설정값으로 사용한다.

최종적으로 선정한 하이퍼파라미터 설정값은 다음과 같다.

하이퍼파라미터	설정값
Learning rate	0.01, 0.05, 0.1
Activation Functions	Sigmoid, Relu, Tanh
# of Hidden layers	1, 2, 3
# of Hidden nodes	65, 70, 75, 80, 85

6.2. 하이퍼파라미터 조합 비교

Grid search를 통해 총 108가지의 조합을 확인하였고, 조합의 검증 데이터에 대한 AUROC는 다음과 같다.

종류	AUROC 값
Maximum AUROC	0.758
Minimum AUROC	0.717
Average AUROC	0.743

최적의 하이퍼 파라미터 조합은 다음과 같다.

Learning Rate: 0.05, Activation Function: relu, Hidden Layers: 1, Hidden Nodes: 75

Validation data 기준으로 가장 높은 AUROC는 0.758이었다. 다른 모델에서의 성능은 Test data에서의 성능을 기준으로 작성하였으니, 동일한 기준에서의 성능을 확인해보기위해 Test data로도 실험해보았다. Test data 기준 AUROC는 0.766으로 Validation data보다좋은 성능을 보여주었고, Pre-pruning Decision Tree의 test data에서의 성능 0.714보다도 좋은 성능을 보여주었다. 참고로 Neural Net의 다른 평가 지표 성능은 다음과 같다. * Test data 기준

	Neural Net
Accuracy	0.709
BCR	0.70
AUROC	0.766

F1-Score	0.742
Precision	0.667
Recall	0.835
TPR	0.835
TNR	0.582

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 Matrix와 같이 작성하고 이에 대한 결과를 해석해보시오.

7.1. 모델 성능 비교

Logistic Regression, Decision Tree(pre-pruning), Neural Network의 Test data 성능은 다음과 같다.

Dataset	Model	TPR	TNR	Accuracy	BCR	F1-Score
Dataset Name	**Logistic Regression**	0.792	0.628	0.710	0.710	0.733
	Decision Tree	0.682	**0.746**	**0.714**	**0.714**	0.704
	Neural Network	**0.835**	0.582	0.709	0.708	**0.742**

TPR은 모델에서 양성으로 올바르게 식별한 실제 양성 사례의 비율을 의미한다. TPR이 높을수록 양성을 올바르게 양성이라고 식별함을 의미하는데, Neural network에서 가장 높고, Decision Tree에서 가장 낮다. 이는 Neural network가 다른 모델에 비해 양성을 양성이라고 올바르게 식별함을 의미한다. 즉 Neural network에서 심혈관 질환 환자들을 가장 올바르게 심혈관 질환 환자라고 판단하였다.

TNR은 모델에서 음성으로 올바르게 식별한 실제 음성 사례의 비율을 의미한다. TNR이 높을수록 음성을 올바르게 음성이라고 식별함을 의미하는데, Decision Tree에서 가장 높고, Neural Network에서 가장 낮다. 이는 Decision Tree가 다른 모델에 비해 음성을 음성이라고 올바르게 식별함을 의미한다. 즉 Decision Tree에서 심혈관 질환 환자가 아닌 사람들을 가장 올바르게 심혈관 질환 환자가 아니라고 판단하였다. 또한 여러 평가지표 중 TNR에 주목해야할 이유가 있는데, 성능 평가지표간 차이가 0.742 ~ 0.582로 가장 크다는 것이다. 즉 사용자의 기준에 따라 차이가 존재하겠지만, 특정 평가지표에서 다른 평가지표에 비해 큰 차이로 성능이 떨어지는 Neural Network를 신뢰하며 사용할 수 있을지 의문이 든다. F1-Score에서 가장 좋은 성능을 발휘한 Neural Network가 TNR에서는 유독 약한 성능을 보이기에, 좋은 모델이라고 말하기는 어렵다는 생각이 든다.

Accuracy는 True 양성과 True 음성을 모두 고려하여 계산한다. Decision Tree에서 가장 높고, Neural Network에서 가장 낮다. 이는 Decision Tree가 다른 모델에 비해 전반적으로 True 양성과 True 음성, 즉 심혈관 질환 환자는 심혈관 질환 환자로, 심혈관 질환 환자가 아닌 사람들은 심혈관 질환 환자가 아니라고 가장 잘 판단하였음을 의미한다. 그러나 세 모델 간 성능에 있어 차이가 거의 없다.

BCR은 TPR 및 TNR의 평균을 계산하여 모델 성능의 전반적인 척도를 제공한다. Decision Tree에서 가장 높고, Neural Network에서 가장 낮다. 이는 Decision Tree가 다른 모델에 비해 양성과 음성을 모두 올바르게 분류하는 데 있어서 종합적인 성능이 가장 우수함을 의미한다.

F1-Score는 정밀도와 재현율을 단일 메트릭으로 결합하여 전반적으로 계산한다. 이는 클래스 간 불균형이 있을 때 유용하게 사용한다고 수업시간에 배웠지만, 불균형 문제가 없어도 사용 가능하다. 참고로 데이터는 50:50으로 불균형 문제는 없다. ogistic Regression에서 가장 좋은 성능을 발휘하였다.

종합: 5개의 평가 지표중에 Decision Tree가 3개의 평가지표에서 가장 좋은 성능을, Neural Network가 2개의 평가지표에서 가장 좋은 성능을 보였다. 수치만 놓고 보면 Decision Tree를 가장 좋은 모델이라고 판단할 수 있다고 생각할 수 있겠지만, 이는 올바르지 않은 방법이다. 이유로는 두 가지고 있다.

1. **데이터셋에서 가장 중점적으로 보아야하는 지표가 무엇인지 고려해야한다.**
2. **Neural Network 의 하이퍼파라미터 조정으로 결과가 바뀔 수 있다.**

데이터셋에서 가장 중점적으로 보야아하는 지표는 데이터의 특성에 따라 다르다. 본 데이터에서는 F1-Score와 AUROC가 가장 중요하다고 판단된다. 이유는 예측 정확도도 중요하지만, 질병 여부를 잘못 예측하면 한 사람의 인생이 바뀌기에, 잘못 예측하였는지도 매우 중요하다. F1-Score에서는 Neural Network가 0.742로 더 좋은 성능을 발휘하였고, AUROC또한 Neural Network가 0.766으로 더 좋은 성능을 발휘하였다. (참고로 Logistic Regression에서의 AUROC는 0.710이다) 하지만 또 이 값만 놓고, 더 좋다고 판단하기 애매한게 Neural Network가 TNR에서 0.582라는 상대적으로 큰 차이로 가장 좋지 못한 성능을 보였기 때문이다. 따라서 사용자의 목적에 따라 각기 다른 모델을 사용해야한다고 말하고 싶다. 평균적으로 좋은 결과를 보이고 싶다면 Neural Network를, 위험 부담에 대한 cost가 크다면 Logistic Regression 혹은 Decision Tree를 추천하고 싶다. 하지만 만약 컴퓨팅 파워가 충분하다면 epoch 및 batch를 키우고 hidden layer를 적절히 조합하면 Neural Network가 결과론적으로는 더 좋은 성능을 보이지 않을까 조심스럽게 생각 및 추천해본다.

Q8.

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

8.1. 데이터 셋

이름: Breast Cancer Wisconsin (Diagnostic)

사이트: Kaggle

링크: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

8.2. 선정 이유

선정한 데이터는 유방암 양성여부를 종속변수로, 세포의 디지털화된 이미지에서 추출한 특성(반지름 평균, 면적 평균, 질감 평균 등)을 독립변수로 구성되어있다. 나아가 질감에 관한 평균, 표준편차 등 여러 통계값들을 독립변수로 사용되기에, 해당 속성들만으로는 예측 결과물에 대한 해석이 힘들수 있겠다고 판단하였다. 예를 들어 질감의 평균은 질병이라고 예측하는 경향이있고, 질감의 표준편차는 질병이 아니라고 예측하는 경향이있다면 예측 결과물에 대한 해석이 힘들기 때문이다. 따라서 데이터셋의 예측 정확도를 높이는 데 초점을 두는 것이 선정한 데이터셋에서는 최선의 방

책이라고 생각하여 선정하게되었다. 더불어 해당 데이터는 예측 정확도가 95% 이상을 보일 정도로 높은 데이터셋으로, 목적에 맞는 성능또한 실제로 발휘한다.

8.3. 데이터 크기

of rows: 569, # of columns → [569 x 32]

(가이드라인) 해당 데이터셋에 대해서 학습:검증:테스트 용도로 적절히 분배하시오(예: 60:20:20). 본인이 분배한 비율에 대해서 간략히 근거를 설명하시오. 분류 성능을 평가/비교할 때는 TPR, TNR, Precision, Accuracy, BCR, F1-Measure, AUROC를 복합적으로 고려하여 서술하시오.

8.4. 학습:검증:테스트 분배

Q1에서 답변한 분배 기준과 동일한 방법으로 분배를 진행하였다. 즉 아래의 세 가지 시나리오중 어떤 시나리오가 Baseline과 validation set에서 가장 유사한 타겟 변수의 비율을 지녔는지 실험해 보았다. 실험결과는 난수(seed)별 영향을 최소화하고자 서로다른 난수를 바탕으로 30번의 반복실험을 진행한 뒤, 평균 값을 산출하였다.

	타겟 변수 1의 비율 (%)
Baseline (분배 전)	62.742%
80/10/10	61.053%
70/15/15	60.275%
60/20/20	60.760%

Q1에서는 종속 변수의 비율이 거의 동일하였지만(50.03:49.97), 이번 데이터셋에서는 타겟 변수의 비율이 62.742%로 Q1보다는 차이가 존재하였다. 이러한 이유로 변동성이 생겨 이번에는 시나리오별로 다른 결과가 도출되었다. Baseline과 가장 유사한 시나리오, 즉 validation set에서의 타겟 변수 비율이 가장 비슷한 시나리오는 80/10/10이기에 80/10/10을 분배 기준으로 선택하였다.

Q9.

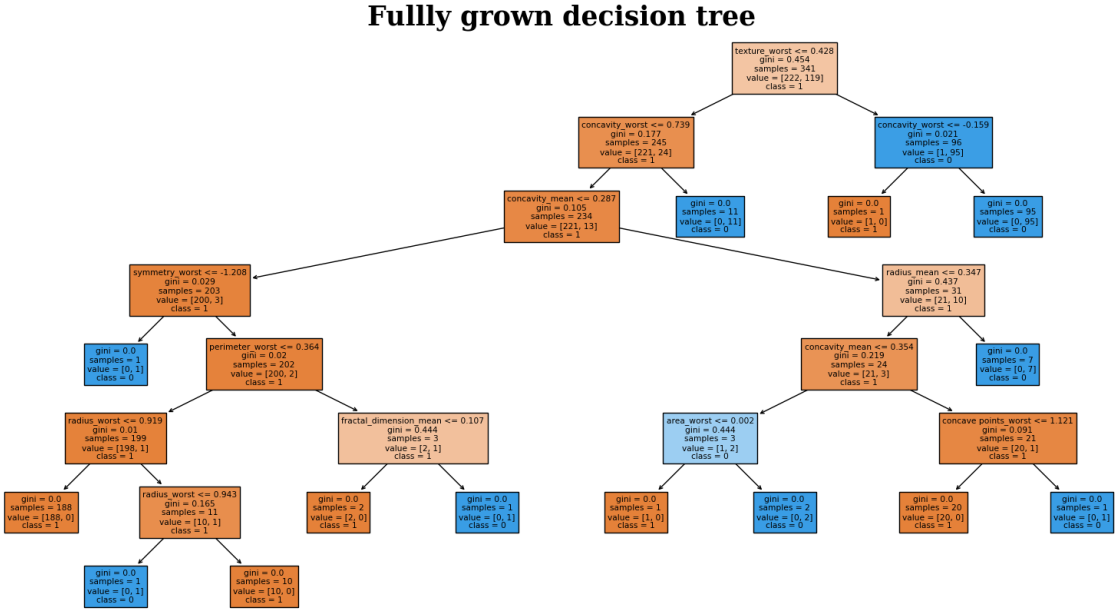
(Decision Tree/Neural Network 공통) [Q8]에서 선택한 데이터셋을 사용하여 [Q3]에서 수행한 최적의 pre-pruning Decision Tree 모델 찾기, [Q6]에서 수행한 최적의 Neural Network 모델 찾기를 동일하게 수행하시오.

9.1. (Decision Tree) Pre-pruning을 위한 하이퍼파라미터 설정

Pre-pruning에서 설정 가능한 하이퍼파라미터는 다음과 같다.

하이퍼파라미터	설 명
max_depth	Decision Tree의 최대 깊이를 제한
min_samples_split	노드를 분할하는데 필요한 최소 샘플 개수
min_samples_leaf	Leaf 노드에 존재하는 최소 샘플 개수
max_leaf_nodes	최대 Leaf 노드 개수

하이퍼파라미터 값을 어떻게 조정할지는 데이터의 성격, 형태에 따라 매우 다르다. 따라서 먼저 데이터를 이해하고, Full Tree의 전개 방식을 그림을 통해 살펴보았다.



Full Tree의 depth는 7, Leaf node의 최소 샘플 개수는 1, Leaf node의 수는 9개였다. 또한 노드를 분할하는데는 필요한 가장 작은 sample 수는 3개였다. Pre-pruning 하이퍼파라미터 설정을 위해서 먼저, 기존의 Full Tree와 비교를 진행하였다. 비교 결과는 다음과 같다.

하이퍼파라미터	대소비교	최대(최소) 가능값
max_depth	Full Tree > Pre-pruning	6(최대)
min_samples_split	Full Tree < Pre-pruning	2(최소)
min_samples_leaf	Full Tree < Pre-pruning	4(최소)
max_leaf_nodes	Full Tree > Pre-pruning	8(최대)

위의 대소비교에 맞게 설정한 Pre-pruning 하이퍼파라미터는 다음과같다.

하이퍼파라미터	설정값
max_depth	3, 5, 6, None
min_samples_split	3, 5, 10, None
min_samples_leaf	10, 15, 20, None
max_leaf_nodes	5, 6, 7, None

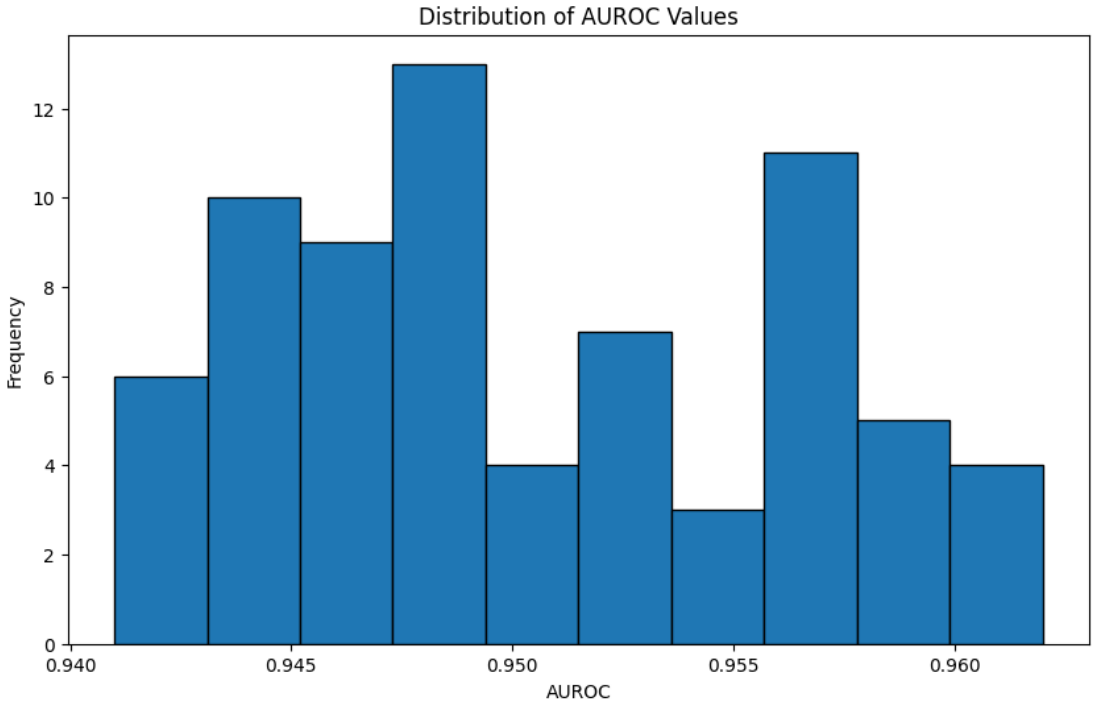
grid search(cv=5)를 통해 하이퍼파라미터 설정값의 모든 조합을 검증 데이터의 AUROC를 기준으로 비교해보았다.

9.2. (Decision Tree) Pre-pruning을 위한 하이퍼파라미터 조합 성능 비교

Grid search를 통해 총 256가지의 조합을 확인하였고, 조합의 검증 데이터에 대한 AUROC는 다음과 같다.

종류	AUROC 값
Maximum AUROC	0.962
Minimum AUROC	0.941
Average AUROC	0.950

256가지 조합의 검증 데이터 AUROC 분포는 아래 그림과 같다.



최적의 하이퍼 파라미터 조합은 다음과 같다.

max_depth: 3, max_leaf_nodes: 7, min_samples_leaf: 20, min_samples_split: 5

Full Tree의 AUROC는 0.937이었고, 이는 Test data 기준이었다. 최적의 하이퍼 파라미터 조합을 Test data로 AUROC를 측정해보면 0.909이다. Validation data만 보았을 때는 pre-pruning 이후에 성능이 개선되었다고 착각할 수 있지만, 동일한 test data로 확인해보면 성능은 저하되었음을 확인 가능하다.

9.3. (Neural Network) 하이퍼파라미터 설정

Neural Network에서 조절가능한 하이퍼파라미터는 다음과 같다.

하이퍼파라미터	설 명
Learning rate	수렴 속도와 학습 과정의 안정성 제어
Activation Functions	비선형성 도입을 위한 근사 함수
# of Hidden layers	네트워크의 복잡한 표현을 가능하게 해줌
# of Hidden nodes	복잡한 패턴을 학습
Optimizer	가중치 업데이트 방식을 결정

Batch Size	업데이트에 필요한 샘플 수 결정

위의 하이퍼파라미터를 모두 Grid Search를 시도해보기는 많으니, 변화가 클 것으로 예상되는 하이퍼파라미터만 선택한다. 이를 위해 먼저, 고정시킬 하이퍼파라미터부터 선택한다. 고정시킬 하이퍼파라미터는 사용하는 데이터의 특성에 상관없이 성능을 발휘할 수 있는 하이퍼파라미터들로 선택하였다. 기준 및 실험은 Q6에서와 동일한 방법으로 진행하였다.

# of hidden nodes는 5개부터 100개까지 5단위로 증가시키며 총 20개를 확인해보았다. AUROC 값을 바탕으로 hidden nodes가 몇 개 정도일 때 AUROC 값이 높을지 아래의 결과를 통해 추측할 수 있다. 즉, 실험한 '# of hidden nodes가 몇 개일때 좋은 성능을 발휘할까'를 바탕으로 grid search에서 설정할 hidden node 값의 범위를 좁힌다는 의미이다.

AUROC 기준 상위 10개의 hidden nodes와 AUROC 값은 다음과 같다.

# of hidden nodes	AUROC
60	0.991
70	0.990
55	0.988
95	0.987
40	0.987
85	0.986
65	0.987
30	0.985
45	0.985
80	0.984

AUROC 값의 차이가 크지도 않고, hidden nodes의 범위가 특정 부분에 집중되어있는 것이 아니라, 폭넓게 분포되어있다. 예를 들어 60(1위) ~ 30(8위)의 범위만해도 hidden nodes 수가 30의차이가 난다는 의미이고, 1위와 10위의 AUROC 값의 차이는 0.007의 정도로 매우 크다고 말할 수 없는 정도이다. 따라서 hidden node는 50 ~ 80사이의 값들을 10간격으로 설정하였다.

Learning rate가 크면 속도는 빠르지만 수렴하지 못할 가능성이, 작으면 너무 많은 학습 시간이 걸린다는 단점이 있다. 일반적으로 0.1 이하의 학습률을 사용한다고 수업시간에 배웠고, 따라서 0.01, 0.05, 0.1의 세 가지값을 가능한 학습률로 설정하였다.

Activation Functions은 Relu가 가장 일반적이라고 알려져있는데, 이는 layer가 많은 딥러닝 기준

이다. Relu는 기울기 소실 문제를 해결하기위해 나왔고, 기울기 소실은 딥러닝에서 많은 layer를 만들면서 생긴 문제였는데, 본 과제에서는 Hidden layer를 3개 이하로 적용할 것이므로 Relu를 꼭 사용할 필요는 없다. 따라서 Sigmoid와 Relu를 모두 고려하였고, Tanh는 중앙값이 0이라 Sigmoid에서 발생하는 편향 이동이 발생하지 않는다. 또한 sigmoid보다 학습 효율성이 뛰어나다고 알려져있어 추가하였다.

# of Hidden layers는 다음과 같이 설정하였다. 딥러닝에서는 hidden layer가 매우 많지만 딥러닝 이전의 NeuralNet, 즉 프로젝트에서 사용하는 NN에서의 hidden layer는 많아야 3개이다. 따라서 1, 2, 3의 Hidden layer를 설정값으로 사용한다.

최종적으로 선정한 하이퍼파라미터 설정값은 다음과 같다.

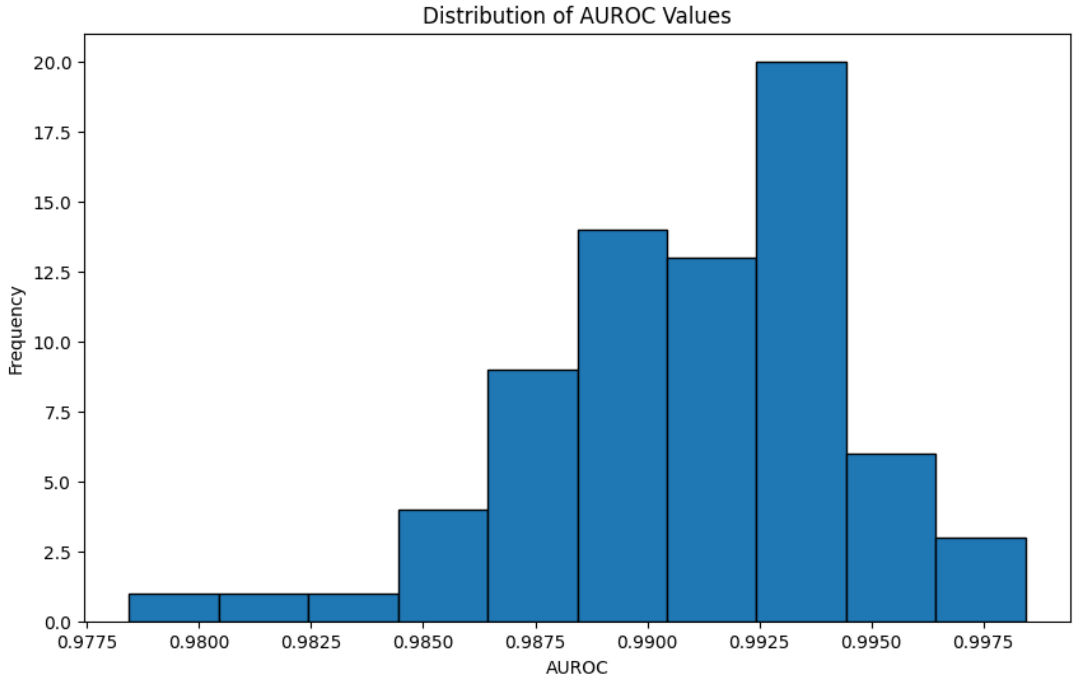
하이퍼파라미터	설정값
Learning rate	0.01, 0.05, 0.1
Activation Functions	Sigmoid, Relu, Tanh
# of Hidden layers	1, 2, 3
# of Hidden nodes	20, 25, 35, 40

9.4. (Neural Network) 하이퍼파라미터 조합 성능 비교

Grid search를 통해 총 108가지의 조합을 확인하였고, 조합의 검증 데이터에 대한 AUROC는 다음과 같다.

종류	AUROC 값
Maximum AUROC	0.998
Minimum AUROC	0.978
Average AUROC	0.991

108가지 조합의 검증 데이터 AUROC 분포는 아래 그림과 같다.



최적의 하이퍼 파라미터 조합은 다음과 같다.

Learning Rate: 0.05, Activation Function: relu, Hidden Layers: 3, Hidden Nodes: 40

Validation data 기준으로 가장 높은 AUROC는 0.998이었다. 이 값은 다른 모델에 비해 월등히 높은 성능을 보일 확률이 존재함을 의미하였고, 정확히 판단하고자 동일한 기준인 Test data로도 실험해보았다. Test data 기준 AUROC는 0.981로 Validation data보다는 다소 성능이 떨어졌지만, Decision Tree보다는 좋은 성능을 보여주었다. Neural net의 다른 평가 지표의 성능은 다음과 같다.

* Test data 기준

	Neural Net
Accuracy	0.965
BCR	0.958
AUROC	0.981
F1-Score	0.952
Precision	0.976
Recall	0.930
TPR	0.930
TNR	0.986

결과에서 흥미로웠던 점은 Neural Net의 성능이 Decision Tree보다 우수했다는 점이다. 일반적으로 Neural Net의 성능이 더 좋고, 데이터가 많을수록 그런 경향을 띈다고 알려져있다. 그렇지만 Data의 수가 569개로 워낙 적었고 8/1/1로 데이터를 분할하여 더욱더 적은 validation, test data만을 지니고 있었기에, 명확한 Rule이 없는 Neural Net의 성능이 Rule을 만들어 놓은 decision tree보다는 다소 떨어지지 않을까 예상하였기 때문이다.

Q10.

(Decision Tree/Neural Network 공통) [Q8]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 Matrix와 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(일치 여부가 평가 점수에 영향을 미치지 않음).

10.1. 모델 성능 비교

Logistic Regression, Decision Tree(pre-pruning), Neural Network의 성능은 다음과 같다.

Dataset	Model	TPR	TNR	Accuracy	BCR	F1-Score
Dataset Name	**Logistic Regression**	**0.977**	0.972	**0.974**	**0.974**	**0.966**
	Decision Tree	0.860	0.958	0.921	0.909	0.892
	Neural Network	0.930	**0.986**	0.965	0.958	0.952

TPR은 모델에서 양성으로 올바르게 식별한 실제 양성 사례의 비율을 의미한다. TPR이 높을수록 양성을 올바르게 양성이라고 식별함을 의미하는데, Logistic Regression에서 가장 높고, Decision Tree에서 가장 낮다. 이는 Logstic Regression이 다른 모델에 비해 양성을 양성이라고 올바르게 식별함을 의미한다. 즉 Logistic Regression에서 유방암 환자들을 가장 올바르게 유방암 환자라고 판단하였다.

TNR은 모델에서 음성으로 올바르게 식별한 실제 음성 사례의 비율을 의미한다. TNR이 높을수록 음성을 올바르게 음성이라고 식별함을 의미하는데, Neural Network에서 가장 높고, Decision Tree에서 가장 낮다. 이는 Neural Network가 다른 모델에 비해 음성을 음성이라고 올바르게 식별함을 의미한다. 즉 Neural Network에서 유방암 환자가 아닌 사람들을 가장 올바르게 유방암 환자가 아니라고 판단하였다.

Accuracy는 True 양성과 True 음성을 모두 고려하여 계산한다. Logistic Regression에서 가장 높고, Decision Tree에서 가장 낮다. 이는 Logstic Regression이 다른 모델에 비해 전반적으로 True 양성과 True 음성, 즉 유방암 환자는 유방암 환자로, 유방암 환자가 아닌 사람들은 유방암 환자가 아니라고 가장 잘 판단하였음을 의미한다.

BCR은 TPR 및 TNR의 평균을 계산하여 모델 성능의 전반적인 척도를 제공한다. Logistic Regression에서 가장 높고, Decision Tree에서 가장 낮다. 이는 Logistic Regression이 다른 모델에 비해 양성과 음성을 모두 올바르게 분류하는 데 있어서 종합적인 성능이 가장 우수함을 의미한다.

F1-Score는 정밀도와 재현율을 단일 메트릭으로 결합하여 전반적으로 계산한다. 이는 클래스 간 불균형이 있을 때 유용하게 사용한다. 하지만 사용한 데이터는 클래스 불균형 문제는 없다. 이 또한 Logistic Regression에서 가장 좋은 성능을 발휘하였다.

5개의 평가 지표중에 Logistic Regression이 4개에서 가장 좋은 성능을, Neural Network가 1개에서 가장 좋은 성능을 보였다. 수치만 놓고 보면 Logistic Regression을 사용함이 가장 적절해 보이겠지만, 이는 올바르지 않은 방법이다. 이유로는 두 가지고 있다.

1. **데이터셋에서 가장 중점적으로 보아야하는 지표가 무엇인지 고려해야한다.**
2. **Neural Network 의 하이퍼파라미터 조정으로 결과가 바뀔 수 있다.**

데이터셋에서 가장 중점적으로 보야아하는 지표는 데이터의 특성에 따라 다르다. 본 데이터에서는 **Accuracy**가 가장 중요하다고 판단된다. 데이터셋을 선정할 때 주어진 조건이 예측 정확도가 훨씬 중요한 데이터였기 때문이다. Accuracy에서 가장 좋은 성능을 발휘한 모델은 Logistic Regression이고 가장 나쁜 성능을 발휘한 모델은 Decision Tree였다. 만약 Accuracy에서 Logistic Regression이 좋은 성능을 발휘하지 못하였다면 다른 결론을 도출할 수 있겠지만, 가장 중요한 평

가지표인 Accuracy와 더불어 5개중 4개의 평가지표에서 가장 좋은 성능을 발휘하였기에 Logistic Regresssion을 세 모델 중에서 추천한다. 다만 Q7에서도 언급하였듯이, 만약 컴퓨팅 파워가 충분하다면 epoch 및 batch를 키우고 hidden layer를 적절히 조합하여 Neural Network가 더 좋은 성능을 발휘할 여지가있는지 검증해본 뒤 사용하기를 추천한다.

10.3. Q7 데이터셋 선정 당시 예상

데이터셋을 선정할 당시에 예상한 성능과 Q10의 결과표를 바탕으로 확인한 실제 성능은 다음과 같다.

예상: Neural Network > Logistic Regression > Pre-pruning Decision Tree

Q7 결과표: Neural Network > Logistic Regression > Pre-pruing Decision Tree

* 앞서 언급하였다시피 Q7의 결과는 목적에따라 모델 순서가 바뀔 수 있음)

Pre-pruning Decision Tree: Decision Tree는 데이터에서 복잡한 상호 작용 및 비선형 관계를 포착하는 것으로 알려져있다. 해당 모델을 가장 좋지 못한 성능을 발휘할 것이라고 예상한 이유는, 모델이 좋지 않아서가 아니다. 다만, 다른 모델에 비해서 특출난 효과가 없을 것이라고 판단하였기 때문이다.

심혈관 질환 데이터는 몸무게, 수축기 혈압, 나이 등 규칙을 만들 때 상식선에서 납득가능한 속성들로 이루어져있다. 그럼에도 Decision Tree가 상대적으로 좋지 못한 성능을 발휘할 것이라고 예상하였던 이유는 두 가지이다. **1**. 다른 모델에서 확실하게 예상가는 장점이 있다. 이는 추후 다른 모델에서 설명한다. **2.** 특정 속성이 규칙을 따르지 않는 경우가 존재한다. 예를들어 현대인들의 패스트푸드 섭취 등 나빠진 식습관으로 인해, 나이에 관계없이 심혈관 질환을 겪는 환자가 증가하고 있고, 고혈압 등은 가족력으로 생기는 경우도 있기 때문이다. 물론 그렇지 못한 경우도 많지만 Decision Tree는 Rule based이기에 최대한 정확하고 일반적인 Rule 생성이 중요하기에, 해당 데이터는 일반적인 Rule 생성은 위와 같은 이유로 어려움을 겪을 것이라고 판단하였다.

Neural Net: 신경망은 데이터에서 복잡한 패턴과 표현을 학습할 수 있는 모델로, 복잡한 관계를 포착하는데 탁월하며 고차원 데이터를 효과적으로 처리할 수 있다.

Neural Net에서 가장 좋은 성능을 발휘할 것이라고 판단한 이유는 데이터의 수에 방점을 두었다. 딥러닝을 수행하기 위해서는 10,000개 이상의 데이터가 필요하다는 속설이 있는데, 선정한 데이터는 무려 70,000개의 데이터를 지니고 있다. Neural net은 복잡한 패턴을 잘 학습하고, 데이터가 많아질수록 정확해지는 경향이 존재한다. 또한 Neural net은 Decision Tree에서 문제로 삼았던 '특정 속성을 따르지 않는'을 상대적으로 잘 학습한다. Decision Tree와 다르게 rule이 없기에 classify 한 이유를 알 수는 없지만 그러한 경우를 잘 학습하기에 가장 좋은 성능을 발휘할 것이라고 판단하였다.

Logistic Regression: 로지스틱 회귀는 입력 기능과 이진 대상 변수 간의 관계를 모델링하는 선형 분류 알고리즘이다. 대상이 특정 클래스에 속할 확률을 추정하기 위해 로지스틱 함수를 사용하고, 데이터에 선형 관계가 있는 경우에 특히 적합하다.

심혈관 질환 데이터는 선형관계가 존재하는 부분이 있다고 판단하였다. 몸무게가 높을수록 혈관이 좁아져 심혈관 질환 확률이 증가하고, 콜레스테롤 수치가 높을수록 심혈관 질환 확률이 증가하는 등 선형관계가 존재할것이라고 가정할 수 있는 속성들이 대부분이기 때문이다. 다만 Decision Tree에서 언급한 반례는 선형성을 방해하는 케이스이긴하지만, 그러한 반례가 중복되어 Rule을 생성하는 Decision Tree와 다르게 Logistic Regression에서는 반례가 발휘할 수 있는 힘이 평균적인 가중치로 계산되기에 줄어든다고 판단하였다.

결론적으로 말하자면 세 가지 모델을 바탕으로 예상하였던 모델별 성능 순위와 실제 Q7결과표는 일치하였다. 다만 언급하였다시피 분석 목적이 평균적인 성능인지, 오작동시 비용 최소화인지 등에 따라 모델의 성능 순위는 바뀔 수 있음을 다시 한번 언급한다.

10.2. Q10 데이터셋 선정 당시 예상

데이터셋을 선정할 당시에 예상한 성능과 Q10의 결과표를 바탕으로 확인한 실제 성능은 다음과 같다.

예상: Logistic Regression > Pre-pruning Decision Tree > Neural Net

Q10 결과표: Logistic Regression > Neural Net > Pre-pruning Decision Tree

Pre-pruning Decision Tree: Decision Tree는 데이터에서 복잡한 상호 작용 및 비선형 관계를 포착하는 것으로 알려져있다. 그러나 적절한 가지치기가 없으면 Decision Tree는 훈련 데이터에 과적합되는 경향이 있어 보이지 않는 예에 대한 일반화가 제대로 이루어지지 않는다. 따라서 Pre-pruning을 진행하여 최대한 과적합을 방지하려고 하는데, 필자는 Pre-

pruning이 잘 이루어지지 않을수도 있겠다고 생각하였다.

그 이유는 유방암 환자 데이터의 경우 종양의 지름이나 넓이 등 한 가지 기준으로 확실히 종양이라고 판단하기 애매한 속성들이 존재하였고, 나아가 해당 속성들의 평균, 표준편차 등 통계값을 바탕으로 새로운 속성을 생성하였기 때문이다. 즉 분류기준이 애매한 속성들과, 그 속성들에 종속적인 여러 속성들이 결합되어있는 데이터이기에 Pre-pruning이 잘이루어지지 않을수도 있겠다고 생각하였다. Logistic Regression보다는 좋은 성능을 발휘하지 못할 수도 있겠다고 생각하였다.

Neural Net: 신경망은 데이터에서 복잡한 패턴과 표현을 학습할 수 있는 모델로, 복잡한 관계를 포착하는데 탁월하며 고차원 데이터를 효과적으로 처리할 수 있다.

하지만 유방암 환자 데이터의 경우 569개의 행밖에 존재하지 않아 좋은 성능을 발휘하지 못할 것이라고 생각하였다. 왜냐하면 Neural Net은 Decision Tree, Logistic Regression과 비교하여 명확한 분류 기준이 없기 때문이다. 즉 Decision Tree의 경우 Rule을 생성하여 왜 이러한 기준으로 분류하였는지 확인가능하지만, Neural Net은 인간의 뇌를 모방하였기에 정확한 작동원리를 알지 못한다.

Logistic Regression: 로지스틱 회귀는 입력 기능과 이진 대상 변수 간의 관계를 모델링하는 선형 분류 알고리즘이다. 대상이 특정 클래스에 속할 확률을 추정하기 위해 로지스틱 함수를 사용하고, 데이터에 선형 관계가 있는 경우에 특히 적합하다.

유방암 환자 데이터의 경우 선형 관계가 있다고 생각해볼 수 있다. 종양을 떠올려보자. 원래 우리 몸에 존재하지 않던 덩어리가 생겨, 점점 커지기 시작하면 질병이 점점 악화됨을 의미한다. 즉, 종양의 넓이, 지름과 유방암 여부 사이에는 선형관계가 있다는 의미이다. 그리고 해당 데이터셋은 종양의 넓이, 지름 등의 특성과 통계값을 속성으로 사용하였기에 선형 관계가 있다고 생각할 수 있다.

결론적으로 말하자면 세 가지 모델 중 예상과 Q10의 결과와 일치하였던 결과는 가장 좋은 성능을 발휘할 것이라고 판단하였던 Logistic Regression이다. Pre-pruning Decsion Tree와 Neural Net은 Q10 결과표를 기준으로 Neural Net이 모든 성능 평가지표에서 Pre-pruining Decision Tree를 dominant하기에 Neural Net > Pre-pruning Decsion Tree이고, 필자의 예상과 다른 결과를 도출하였다.

10.4. (추가) Q7과 Q10 결과표 비교

Q7에서 사용한 데이터와 Q10에서 사용한 데이터는 목적이 달랐다. Q7의 데이터는 '예측 결과물에 대한 해석'에 방점을 두었고, Q10의 데이터는 '예측 정확도'에 방점을 두었다. 각기 다른 목적으로 데이터분석이 수행되었기에 우선적으로 확인하였던 성능 평가지표도 각기 달랐다. Q7은 F1-Score와 AUROC를 중점적으로 확인하였고, Q10은 Accuracy를 중점적으로 확인하였다. 각기 다른 성능 평가지표를 바탕으로 내린 결론은 다음과 같았다.

Q7에서는 Neural Network 전체적으로 좋은 성능을 보였지만, TNR에서 0.582라는 상대적으로 큰 차이로 가장 좋지 못한 성능을 보였다. 에 따라서 사용자의 목적에 따라 각기 다른 모델을 사용해야한다고 결론을 도출하였다. 평균적으로 좋은 결과를 보이고 싶다면 Neural Network를, 위험부담에 대한 cost가 크다면 Logistic Regression 혹은 Decision Tree를 추천하였다. 하지만 상황에 상관없이 한 가지 모델을 필수적으로 선정해야 한다면, Neural Network를 추천하고 싶다. 이유는 평균적으로 좋은 결과를 보이는 동시에, epoch 및 batch 조정을 통해 더 좋은 성능을 보일 수 있는 여지가 있는 모델이기 때문이다. Logistic Regression은 선형성을 가정하기에, 개선의 여지가 거의 없고 Decision Tree 또한 Pruning 하이퍼파라미터 조정의 여지밖에 없기 때문이다.

Q10에서는 Logistic Regression을 추천하였다. Q7에서 가장 중요한 성능 평가지표인 Accuracy와 더불어 5개중 4개의 평가지표에서 가장 좋은 성능을 발휘하였기에 Logistic Regresssion을 세 모델 중에서 추천하였다.

각기 다른 데이터와 목적을 바탕으로 수행되었기에 특정 모델이 항상 좋다고 말할 수 없다. 또한 특정 상황에서 특정 모델을 사용하여야한다고도 말할 수 없다. 그러나 프로젝트를 수행한 두 데이터에 관해서는, 세 가지 모델 중 특정 모델을 조심스럽게 추천해볼 수 있는 프로젝트 결과였다고 생각된다.