# Multivariate Data Analysis Assignment #3

## Decision Tree & Neural Network

**산업경영공학부 2020170856 이우진**

**[Q1] 본인이 생각하기에 "예측정확도"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 다운로드 할 수 있는 링크를 반드시 제공하시오.**

Dataset : **Predict Diabetes DataSet**

다운로드 링크 : https://www.kaggle.com/datasets/whenamancodes/predict-diabities

선정 이유 : 해당 데이터셋은 768개의 행과 9개의 변수로 이루어져 있으며, 특정 진단 측정값을 기반으로 당뇨병 여부를 보여주는 데이터이다. 질병 관련 데이터를 분석 및 예측을 하는 이유는 질병의 원인을 파악하고, 그것을 토대로 질병으로부터 미리 예방하는 것에 목적이 있다. 따라서 질병 발생의 중요 요인에 대해 분석을 하는 것이 매우 중요하다.

'Diabetes Dataset'은 혈중 포도당 수치, 혈압, 혈청 인슐린 수치 등 당뇨병에 영향을 미치는 여러 요인들이 독립변수로 존재하고, binary 데이터인 당뇨병 발병 여부가 종속 변수로 존재한다. 이때, 여러 요인들 중 당뇨병에 큰 영향을 끼치는 요인이 무엇인지 분석하고, 결정적으로 당뇨병의 원인을 명확하게 해석할 수 있어야 당뇨병을 예방할 수 있으며, 환자의 치료 계획을 결정하는데 도움을 줄 수 있다. 의사와 같은 의료 분야 종사자들은 환자에게 적절한 치료 옵션을 제공해야 하는데, 발병 원인을 파악해야 환자에게 적합한 진단을 할 수 있기 때문에 해당 데이터셋은 예측 정확도도 중요하지만 예측 결과물에 대한 해석이 매우 중요하다고 생각한다. 예를 들어, 혈중 인슐린 수치가 높아 당뇨병 발병이 된다면, 환자에게 인슐린 수치를 낮추는 치료를 하거나 또는 사전에 예방하는 것이 가능하다. 따라서 본 과제에서는 당뇨병의 발병 원인 및 해당 변수가 얼마나 큰 영향을 끼치는지 분석하여, 궁극적으로 당뇨병의 치료, 예방을 목적으로 하고자 한다.

해당 데이터셋은 1개의 종속 변수와 8개의 설명 변수가 있다.

<종속 변수>

- Outcome : 당뇨병의 발병 여부 (0, 1) / 1 is yes, 0 is No

<설명 변수>

- Pregnancies : 임신 횟수

- Glucose : 혈중 포도당 수치

- BloodPressure : 혈압(mm Hg)

- SkinThickness : 팔 삼두근 뒤쪽의 피하지방 측정값(mm)

- Insulin : 혈청 인슐린(mu U/ml)

- BMI : 체질량지수(체중(kg)/키(m))^2

- DiabetesPedigreeFunction : 당뇨 내력 가중치 값

- Age : 나이

**(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.**

기본적으로 가장 많이 사용되는 비율 중 하나인 60:20:20을 사용하였다. 즉, 460:154:154 개의 데이터로 분할하였다. 그 이유는 다음과 같다.

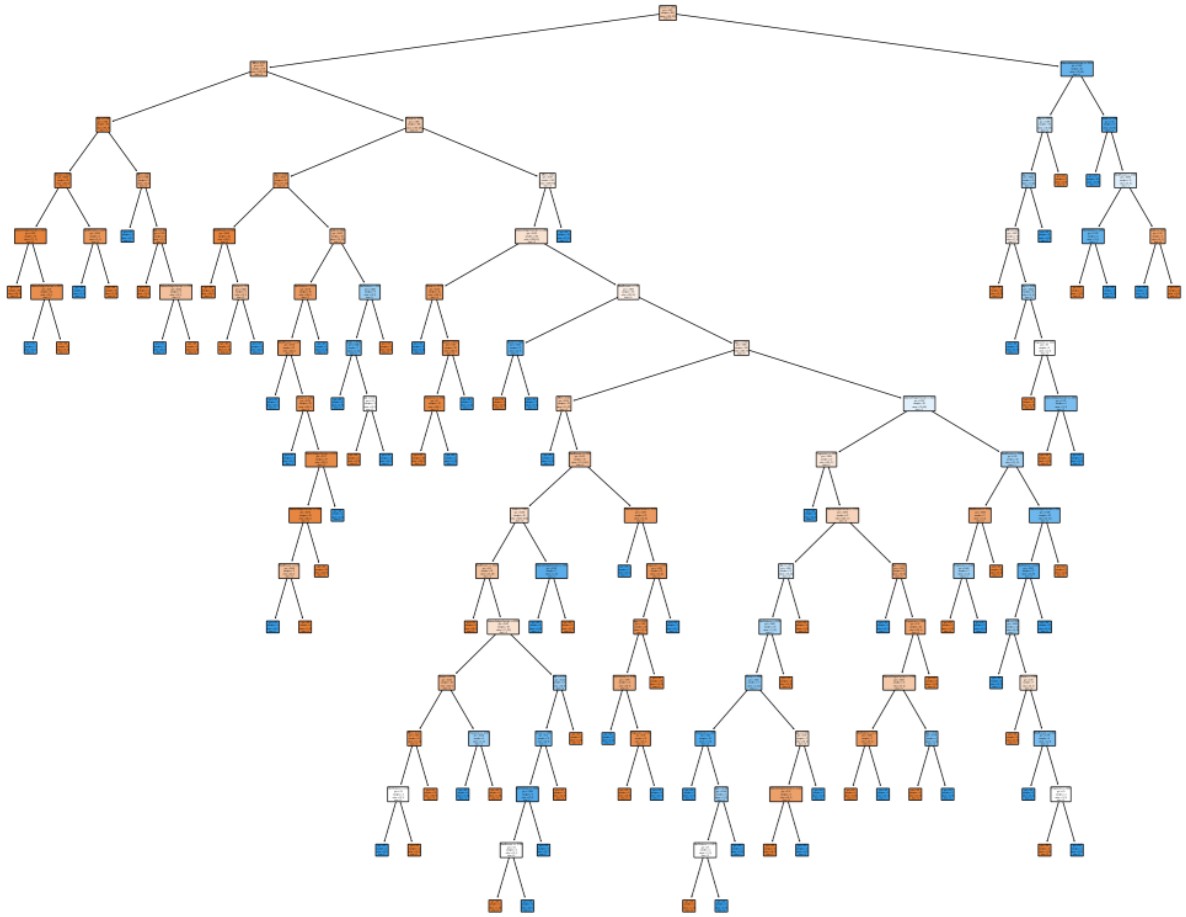
먼저, 모델의 훈련에 사용되는 학습 데이터는 충분한 양이 필요하기 때문에 전체 데이터에서 가장 큰 비율인 60%를 학습에 할당하였다. 만약 더 큰 비율을 학습 데이터셋에 할당한다면, 상대적으로 검증 데이터와 테스트 데이터셋의 크기가 작아지게 되는데, instance의 수가 768개로 많은 편이 아니기 때문에, 검증 데이터와 테스트 데이터셋에 너무 적은 데이터가 할당될 수 있다. 이로 인해 성능 측정이 잘못될 가능성이 있고, 결론적으로 최적의 모델을 찾는데 어려울 수 있다.

따라서 보편적으로 사용되는 또 다른 분배 비율인 80:10:10 대신, 학습에도 많은 데이터를 할당하면서 검증 및 테스트에도 적절한 비율로 데이터를 할당하는 60:20:20을 분배 비율로 선택하였다. 해당 분배 비율을 통해 모델의 훈련, 평가 및 테스트를 적절히 수행할 수 있다고 생각하였다.

**[Q2] (DecisionTree) 아래 두가지의 경우에 대한 테스트 데이터셋에 대한 분류 성능을 평가하고 그 결과를 비교해보시오.**

**2-1. 학습 데이터만을 이용해서 학습한 Full Tree**

DecisionTreeClassifier 모델을 통해 460개의 학습데이터를 학습시켜, Full-Tree를 생성하였고, Full-Tree Plot은 아래와 같다.



다음은 테스트 데이터셋에 대한 Full-Tree의 분류 성능이다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Full-Tree | 0.63636 | 0.56452 | 0.72727 | 0.69481 | 0.68030 | 0.59829 | 0.68182 |

Full-Tree의 성능지표들을 전체적으로 확인해보면, 지표들이 0.56452~0.72727 사이에 분포하는 것을 알 수 있고, 결론적으로 좋지 않은 성능을 보이고 있다고 판단된다. 이는 Full-Tree가 학습 데이터에 과적합되는 경향을 보이기 때문에 나온 결과라고 할 수 있다.

이 중, TNR이 0.72727으로 그나마 준수한 수치를 보이고 있는데, 실제 Negative Class 객체들 중에서 모델에 의해 Negative Class로 정확히 예측된 비율이 약 72.7% 정도 된다는 것을 의미한다.

TNR과 대조적으로 TPR은 0.63636의 수치를 보이는데, 이는 실제 Positive Class 객체들 중에서 모델에 의해 Positive Class로 정확히 예측된 비율이 약 63.6% 정도 된다는 것을 의미한다.

즉, 실제 Positive인 것을 맞추는 비율보다 Negative인 것을 맞추는 비율이 더 높다는 것을 알 수 있다.

Precision을 보았을 때, 양성으로 예측한 것 중 실제 양성인 표본의 비율이 약 56.5%인 것을 알 수 있는데, 이는 거의 50%에 가까운 수치로 다른 지표들에 비해 상당히 낮은 수치를 보여주고 있음을 알 수 있다.

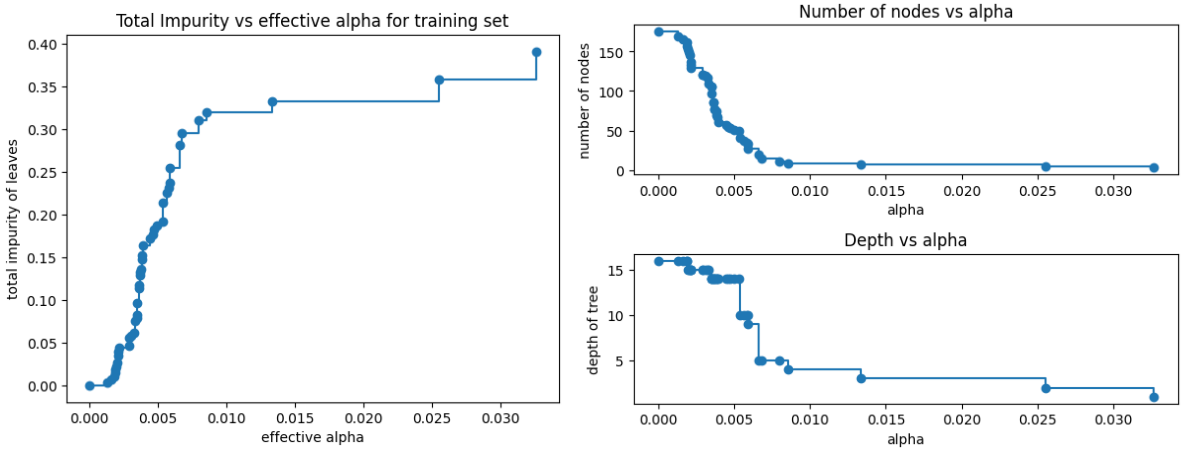
F1-Measure도 0.598로 낮은 수치를 보여주고 있는데 이는 precision과 recall 사이의 balance가 약 59.8%라는 것을 의미한다. 또한, 전체적인 분류 성능을 확인할 수 있는 지표인 AUROC를 보았을 때, 0.682로 0.7보다 낮은 수치를 보여주고, 따라서 해당 분류 모델의 성능이 준수하지 않다고 할 수 있다.

결론적으로 Full-tree는 과적합 되어 있기 때문에, tree plot을 보았을 때도 tree의 복잡도도 높은 것을 알 수 있고, 전체적인 평가지표도 낮은 것을 보아 post-pruning을 진행할 필요가 있다.

### 2-2. 학습 데이터를 사용하여 학습한 후 검증 데이터를 사용하여 Post-pruning을 수행한 Tree

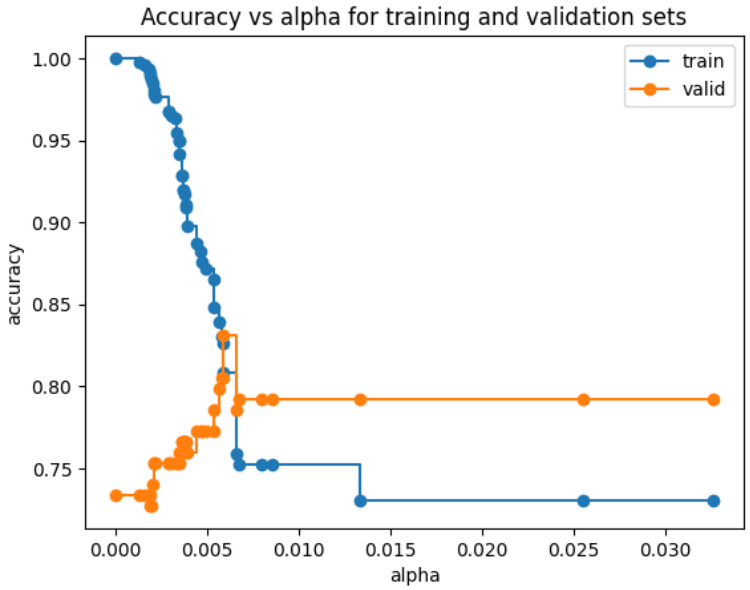
Full tree를 post-pruning할 때, 하이퍼파라미터 알파를 결정해야 한다. 알파 값이 클수록 모델의 복잡도가 감소하고, impurity가 증가하게 된다. 가장 최적화된 알파 값을 찾기 위해, 알파를 적용하여 가지치기를 수행 후 검증 데이터를 사용하여 Accuracy를 평가하고, 그 중 가장 높은 수치를 보이는 알파를 도출하였다.

아래는 알파 값에 따른 Decision tree의 total impurity와 number of nodes, tree depth를 나타낸 그래프이다.

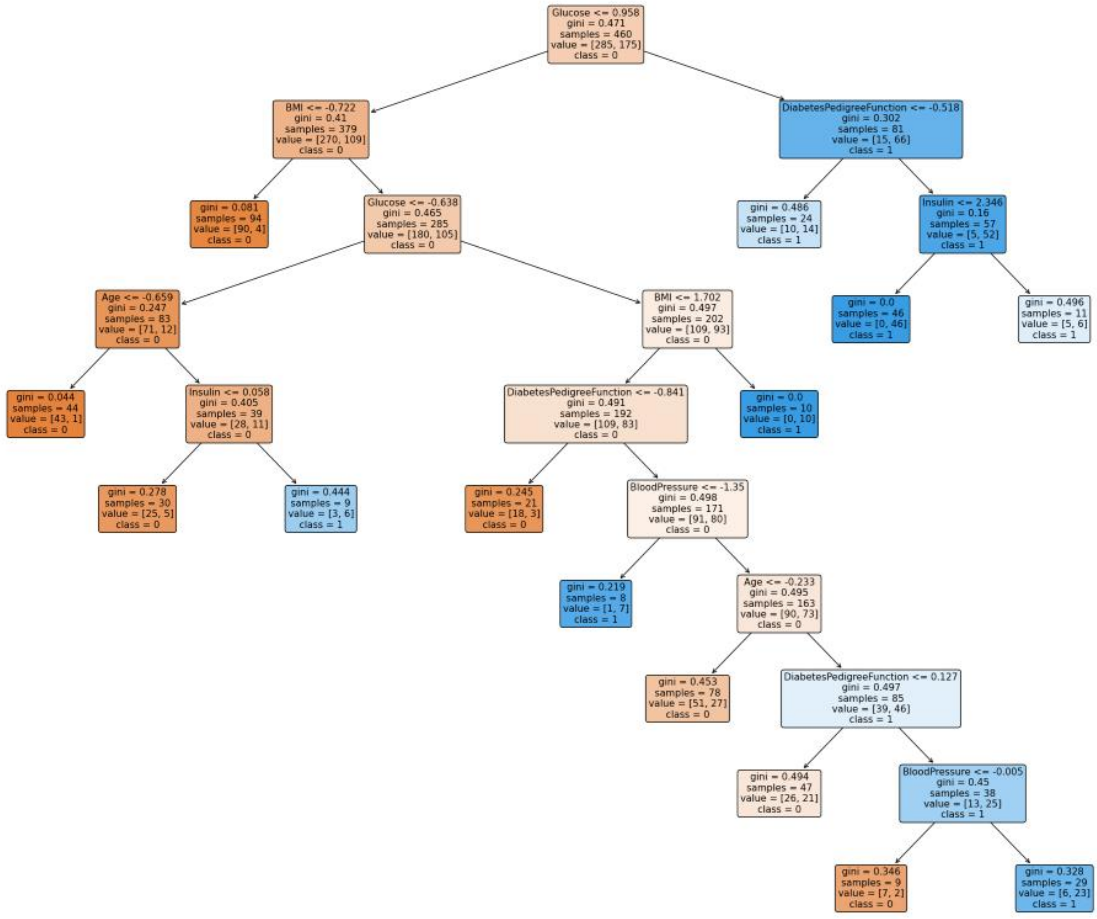



위의 그래프를 보았을 때, 알파 값이 클수록 total impurity값이 증가하고, number of nodes, depth는 감소하는 것을 확인할 수 있다. 결론적으로 tree의 복잡도가 감소한 것을 알 수 있다.

알파 값에 따른 Train/Validation 데이터에 대한 Accuracy를 나타낸 그래프는 아래와 같으며, 검증용 데이터에 대해 가장 높은 수치를 보이는 알파 값은 약 0.081인 것을 알 수 있다. 따라서 validation 데이터셋의 accuracy가 가장 큰 알파 값인 0.081을 이용하여 최적의 모델을 만들었다.




다음은 Post-Pruning을 수행한 Tree의 결과이다.




Post-pruning을 진행한 결과, Full-Tree에 비해 tree의 복잡도가 줄어든 것을 알 수 있다.

다음은 Post-Pruning을 수행한 Tree의 분류 성능이다. 동일하게 테스트 데이터셋으로 평가하였다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Full-Tree | 0.63636 | 0.56452 | 0.72727 | 0.69481 | 0.6803 | 0.59829 | 0.68182 |
| Post-Pruning | 0.56364 | 0.70455 | 0.86869 | 0.75974 | 0.69973 | 0.62627 | 0.71616 |

Post-pruning을 진행한 Tree의 성능을 확인해보면, TPR의 경우 0.56 정도로 낮은 수치를 보이고 있지만, TNR의 경우 0.86 정도로 높은 수치를 보이고 있는 것을 확인할 수 있다.

즉, 실제 Negative Class 객체들 중 모델에 의해 Negative Class로 정확히 예측한 비율은 높지만, 실제 Positive Class 객체들 중 Positive Class로 정확히 예측된 비율은 그보다 30% 낮은 수치를 보인다고 할 수 있다. 이는 Negative인 것을 맞추는 것은 잘하지만, Positive인 것을 맞추는 것은 못한다고 할 수 있고, 이 차이가 크다는 것을 의미한다.

Accuracy를 보면 약 75% 정도로 준수한 성능인 것처럼 보이지만, 전체적인 분류 성능을 평가해줄 수 있는 AUROC는 0.716이고, F1-Measure 또한 0.7보다 낮은 수치를 보인다는 점에서 준수한 성능을 보인다고 하기에 어려워 보인다. 이는 TPR이 다른 지표들에 비해 현저히 낮은 수치를 보이기 때문에 나타난 결과라고 할 수 있다.

Full tree와 Post-pruning을 진행한 결과를 비교하면 다음과 같다.

TPR의 경우, 두 모델 모두 다른 지표에 비해 낮은 수치를 보이는 것을 알 수 있고, 양성 Case를 맞추는 것을 잘 못한다고 할 수 있다.

TPR을 제외한 모든 지표에서, Post-pruning을 진행한 경우가 더 높은 수치를 보이며, 전체적으로 post-pruning을 진행한 Tree가 더 우수하다고 할 수 있다. 이는 post-pruning 결과 불필요한 가지와 노드를 제거하여 일반화 능력이 향상되었기 때문에 나타난 결과라고 할 수 있다.

F1-Measure, AUROC의 경우, Post-pruning을 진행한 모델이 조금 더 높은 수치를 보이지만, 두 모델 모두 높지 않은 수치를 보이고 있음을 알 수 있다. 특히, F1-Measure 수치가 0.6 근처 값인 것을 보아, precision과 recall 사이의 balance가 잘 잡혀 있지 않다고 할 수 있다.

결과적으로 TPR을 제외한 나머지 수치에서 Post-pruning을 거친 Tree가 Full-Tree보다 나은 성능을 보인다고 해석할 수 있지만, positive class를 잘 예측해야 하는 당뇨병 데이터셋이라는 점을 고려했을 때, TPR에서 더 낮은 수치를 보이는 post-pruning이 더 나은 성능을 보인다고 확실하게 결론 내리기 어려워 보인다.

**[Q3](DecisionTree) 학습 데이터와 검증 데이터를 이용하여 Pre-pruning을 수행해보시오. Pre-pruning을 수행하기 위해 사용된 하이퍼 파라미터를 설명하고, 각 하이퍼 파라미터마다 탐색 범위를 어떻게 설정했는지 서술하시오. 검증 데이터에 대한 AUROC를 기준으로 최적의 하이퍼 파라미터 조합을 찾아보시오**

Pre-pruning을 수행하기 위해 사용된 하이퍼 파라미터는 다음과 같다.

1. criterion ["gini", "entropy", "log_loss"]

    이 파라미터는 노드 분할에 사용되는 판단 기준을 결정한다.

    gini의 경우 지니 불순도를 사용하여 분할하고, entropy의 경우 엔트로피(information gain)를 사용하여 분할한다. log_loss는 로그 손실을 사용하여 분할한다.

    impurity가 얼마나 개선되는지 gain의 관점에서 평가하고 분기를 진행하기 때문에, 불순도 계산 함수가 무엇인지에 따라 분기 위치 및 여부가 달라질 수 있다. 따라서 주어진 데이터에 적합한 최적의 분할 기준을 선택하기 위해 위의 후보들을 선정하였다.

2. min_samples_split [2, 5, 10, 25, 50, 100]

    이 파라미터는 노드를 분할하기 위해 필요한 최소 샘플 수를 지정한다.

    해당 값이 작은 경우, 더 복잡한 트리를 생성할 수 있지만, 과적합의 위험이 존재한다.

    반대로 해당 값이 크다면, node가 split되지 않게 해줄 수 있어 과적합을 방지할 수 있다.

    테스트셋의 경우 768개이기 때문에, 여러 경우를 테스트해보기 위해 최소 2부터 최대 100까지 넓은 범위의 값을 선정하였다.

3. max_depth [2, 4, 8, 16]

    이 파라미터는 트리의 최대 깊이를 제한한다.

    더 깊은 트리는 더 complex하게 학습할 수 있지만, 과적합의 위험이 존재한다.

    Full-Tree의 깊이가 16 이기 때문에, 16 이하의 수치들 중 2, 4, 8, 16을 후보로 선정하였다.

4. min_samples_leaf [1, 2, 4, 8, 16]

    이 파라미터는 리프 노드에 필요한 최소 샘플 수를 지정한다.

    해당 값이 작은 경우, 더 복잡한 트리를 생성한 수 있지만, 과적합의 위험이 존재한다.

    반대로 해당 값이 큰 경우 과적합을 방지할 수 있다.

    위 수치는 시행착오법을 통하여 가장 좋은 성능을 보여줄 수 있는 후보들을 선택하였다.

최적의 하이퍼 파라미터 조합을 찾기 위해 validation data를 기준으로 AUROC를 측정하여 가장 높은 수치를 보이는 조합을 선택하였다.

그 결과, 하이퍼 파라미터에 대한 최적의 조합은 다음과 같다.

1. Best criterion: gini

2. Best min_samples_split: 100

3. Best max_depth: 8

4. Best min_samples_leaf: 16

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.7641이다.

여기서, 추가로 동일한 하이퍼파라미터 조합 후보에 대하여 GridsearchCV를 이용하여 학습을 진행해보았다. 즉, 5-fold 교차검증을 통해 AUROC를 계산하고, 가장 높은 수치를 보이는 하이퍼파라미터 조합을 탐색해보았다. 이때, K-fold cross validation은 데이터를 k개로 분할하고, k-1개를 학습용 데이터로, 1개를 검정용 데이터로 사용하여 검증하는 과정이다. 해당 과정을 추가로 수행한 이유는 validation 데이터셋이 154개로 많은 편이 아니기 때문에, 위에서 고른 하이퍼 파라미터 조합은 validation 데이터셋에는 최적화되어 있지만, 다른 데이터셋에 대해서는 최적화된 모델이 아닐 가능성이 있다. 따라서, 더 일반화된 모델을 선택하기 위해 해당 방법으로도 최적의 하이퍼파라미터 조합을 찾아보았고 그 결과는 다음과 같다.

1. Best criterion: entropy

2. Best min_samples_split: 100

3. Best max_depth: 8

4. min_samples_leaf: 8

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.7559이다.

둘의 결과를 보았을 때, 검증 데이터에 대해서 AUROC 값이 큰 차이가 없는 것을 알 수 있다. 또한, 선택한 criterion이 각각 gini, entropy로 다른 것을 알 수 있고, min_samples_leaf 또한 각각 16, 8로 다른 조합을 선택한 것을 알 수 있다. 그 외의 다른 하이퍼파라미터는 동일하다.

검증 데이터셋에서의 AUROC 값이 비슷하기 때문에 2개의 하이퍼파라미터 조합에 대하여 테스트 데이터셋을 사용하여 TPR, TNR, Precision, Accuracy, BCR, F1-Measure, AUROC를 확인하고 가장 최적인 하이퍼파라미터 조합을 선택하고자 한다. 그 결과는 아래와 같다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Pre-Pruning | 0.56364 | 0.73810 | 0.88889 | 0.77273 | 0.70782 | 0.63918 | 0.72626 |
| Pre-Pruning(CV) | 0.76364 | 0.72414 | 0.83838 | 0.81169 | 0.80014 | 0.74337 | 0.80101 |

아래 CV라고 되어 있는 것이 교차검증을 진행한 모델이다.

테스트 데이터셋에 대해 평가를 한 결과, TPR에서 20%의 큰 차이로 교차검증을 진행한 모델이 우수한 성능을 보이는 것을 알 수 있다. 교차검증을 진행한 모델이 Precision, TNR에 대해서는 다소 낮은 수치를 보이지만, Precision은 큰 차이가 없고, TNR도 0.8을 넘어가는 수치를 보이기 때문에 해당 지표에서도 좋은 성능을 보인다고 할 수 있다.

특히, Accuracy, BCR, F1-Meaure, AUROC에서 교차검증을 진행한 모델이 압도적으로 뛰어난 성능을 보이기 때문에, 전체적으로 교차검증을 진행한 모델이 더 뛰어나다고 할 수 있다. 이때, 교차검증을 사용하지 않은 모델은 검증데이터 AUROC는 0.7641이었지만, 테스트데이터 AUROC는 0.7263으로 테스트셋에서의 성능은 더 낮은 것을 확인할 수 있다.

이러한 결과가 나타난 이유는 다음과 같이 설명할 수 있다. 검증 데이터셋의 크기가 154개로 작은 편이기 때문에, 검증 데이터에서의 AUROC 값이 가장 큰 하이퍼파라미터 조합은 특히, 검증용 데이터에 최적화되어 있을 가능성이 높다. 반면, 교차검증은 학습 데이터를 k개로 분할하여 k-1개로 학습하고, 1개로 검증하는 과정을 반복하기 때문에, 더 많은 데이터셋에서 검증이 진행된다. 따라서 데이터가 적을 경우에도 상대적으로 더 잘 작동하고, 더 일반화된 성능을 보이기 때문에, 위와 같은 결과가 나왔다고 해석할 수 있다.

결론적으로 하이퍼파라미터에 대한 최적의 조합은 교차검증 결과로 나온 조합을 사용하고자 하고, 최적의 조합은 다음과 같다.

1. Best criterion: entropy

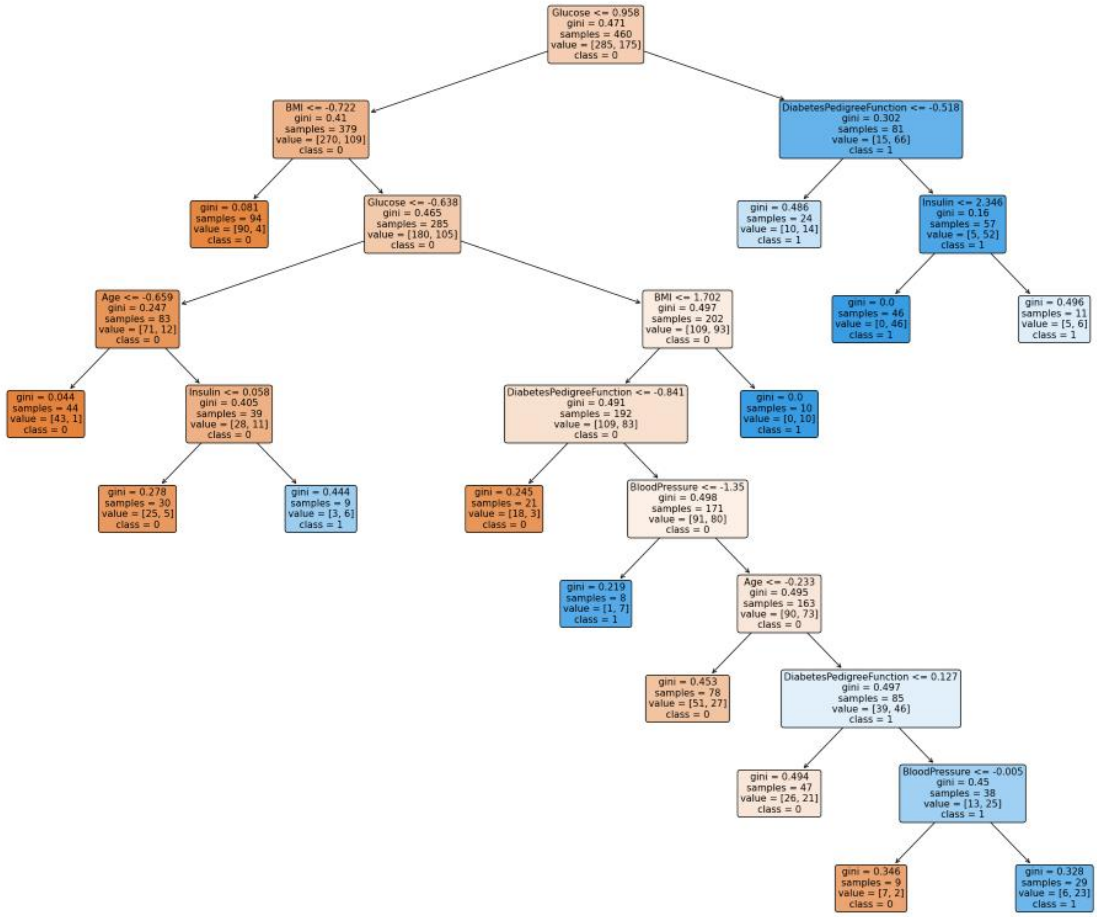
2. Best min_samples_split: 100

3. Best max_depth: 8

4. min_samples_leaf: 8

**[Q4](DecisionTree) [Q2]와 [Q3]에서 생성한 Post-pruning모델과 Pre-pruning모델의 결과물을 각각 Plotting하고 이에 대한 해석을 수행하시오. 각 Pruning방식에 따라 Split에 사용된 변수는 어떤 변화가 있는가?**

먼저 Q2에서 생성한 Post-pruning 모델을 plotting 한 결과는 다음과 같다.

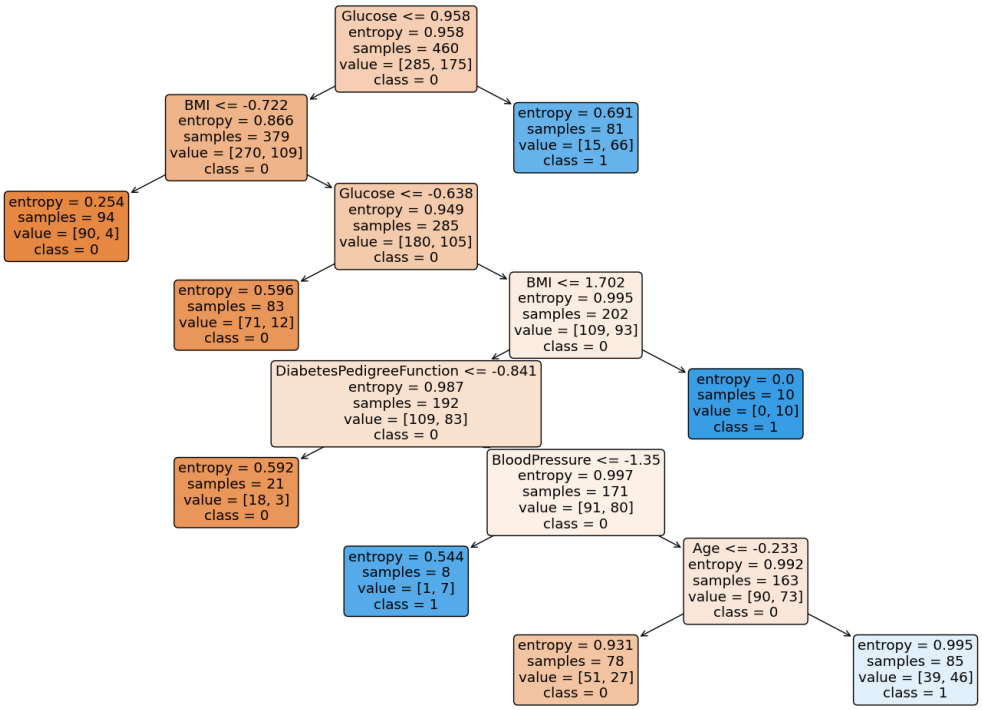



Post-pruning을 진행한 결과, Full-Tree에 비해 tree의 복잡도가 줄어든 것을 알 수 있다.

Tree depth는 9 이고, Leaf node의 수는 총 14개이다. 분기는 총 13번 일어났고, Post-pruning 모델은 Glucose <= 0.958을 split point로 선정하여 분기를 시작한다.

분기에 사용된 변수를 확인해보면, [Glucose, BMI, Age, Insulin, DiabetesPedigreeFunction, BloodPressure]이 있다. 이때, Glucose, BMI, Age, Insulin, BloodPressure 변수는 각각 2회씩 분기에 사용되었고, DiabetesPedigreeFunction은 3회 분기에 사용되었다.

다음으로 Q3에서 생성한 Pre-pruning 모델을 plotting 한 결과는 다음과 같다.




Pre-pruning 모델을 구축한 결과, Tree depth는 5 이고, Leaf node의 수는 총 8개이다. 분기는 총 7번 일어났고, Pre-pruning 모델은 Glucose <= 0.958을 split point로 선정하여 분기를 시작한다.

분기에 사용된 변수를 확인해보면, [Glucose, BMI, Age, DiabetesPedigreeFunction, BloodPressure] 이 있다. 이때, Glucose, BMI 변수는 각각 2회씩 분기에 사용되었다.

두 모델을 비교해 보았을 때, Pre-pruning 모델이 Post-pruning 모델보다 depth가 약 2배 작고, leaf 노드의 수도 약 2배 적은 것을 알 수 있다. 즉, pre-pruning 모델의 복잡도가 더 낮다고 할 수 있다. 또한, 분기에 사용된 변수를 확인해보면 post-pruning과 pre-pruning 모두 공통적으로 [Glucose, BMI, Age, DiabetesPedigreeFunction, BloodPressure] 변수를 사용하는 것을 알 수 있다. 즉, 해당 변수들은 모델의 impurity를 감소시키는 측면에서 중요한 변수라고 할 수 있고, 당뇨병의 발병을 예측하는데 중요하다고 할 수 있다.

반면, 두 모델 모두 사용하지 않는 변수인 [Pregnancies, SkinThickness]는 당뇨병 발병 예측에 중요하지 않은 변수라고 할 수 있다.

각각의 분기에 대해 자세히 해석한 결과는 다음과 같다. pre-pruning 모델과 post-pruning 모델 모두 최상단 노드에서 Glucose <= 0.958을 기준으로 분기를 시작하였다. 또한, Glucose <= 0.958이 True인 경우, 두 모델 모두 BMI <= -0.722를 기준으로 분기를 진행하였다. BMI <= -0.722가 False인 경우에도, 두 모델 모두 Glucose <= -0.638을 기준으로 분기를 진행하였다. 이처럼 pre-pruning 모델과 pre-pruning 모델이 동일한 변수 및 수치를 기준으로 분기를 진행한 부분이 있는

것을 확인하였고, 해당 부분은 당뇨병 예측에 대한 해석에 있어 중요한 변수라는 것을 알 수 있다. 반면, 첫 분기에서 Glucose <= 0.958이 false인 경우, pre-pruning 모델은 분기를 멈추고, 해당 class를 모두 1로 지정하였지만, post-pruning 모델은 DiabetesPedigreeFunction <= -0.518을 기준으로 분기를 계속 진행한 것을 확인할 수 있다. 결론적으로 post-pruning 모델의 복잡도가 더 높은 것을 확인할 수 있다.

**[Q5](DecisionTree) 최적의 결정나무의 Plot을 그리고, 대표적인 세가지 규칙에 대해서 설명해보시오.**

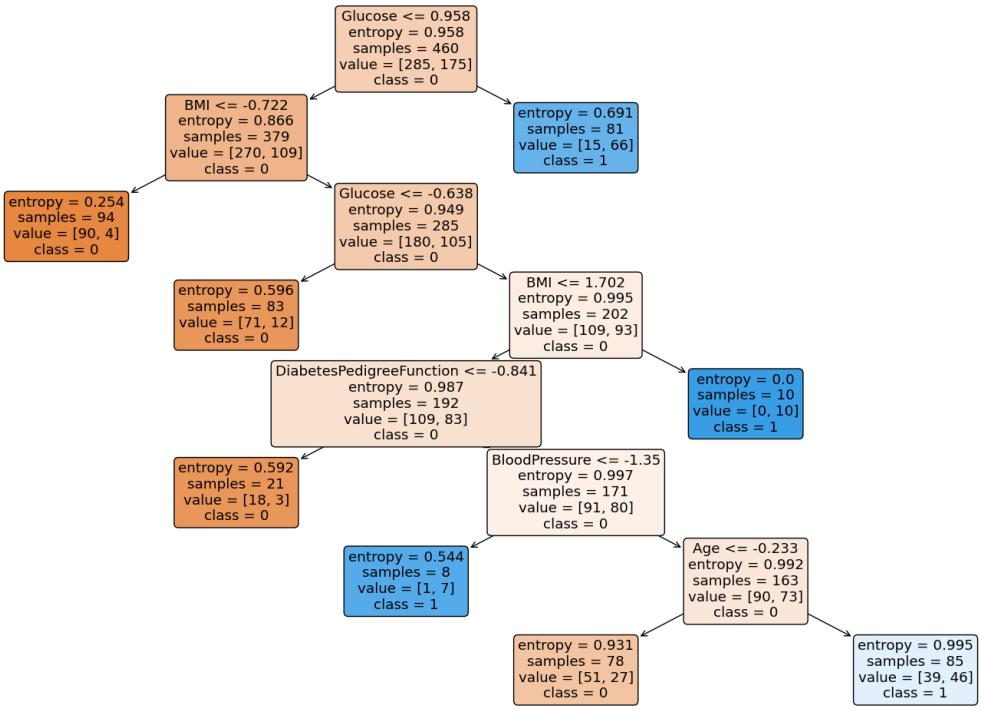
Q2와 Q3의 과정을 통해 생성한 모델의 분류 성능을 다음과 같이 정리하였다. (테스트셋)

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Full-Tree | 0.63636 | 0.56452 | 0.72727 | 0.69481 | 0.68030 | 0.59829 | 0.68182 |
| Post-Pruning | 0.56364 | 0.70455 | 0.86869 | 0.75974 | 0.69973 | 0.62627 | 0.71616 |
| Pre-Pruning | 0.56364 | 0.73810 | 0.88889 | 0.77273 | 0.70782 | 0.63918 | 0.72626 |
| Pre-Pruning(CV) | 0.76364 | 0.72414 | 0.83838 | 0.81169 | 0.80014 | 0.74337 | 0.80101 |

테스트셋에서의 AUROC를 살펴보았을 때, 교차검증을 통한 pre-pruning(CV) 모델이 압도적으로 성능이 좋은 것을 확인할 수 있다. AUROC 뿐 아니라, BCR, F1-Measure, TPR과 같은 지표에서도 pre-pruning(CV) 모델이 다른 모델들에 비해 10~20% 더 높은 수치를 보이는 것을 알 수 있다. Accuracy에서도 유일하게 80%가 넘는 지표를 보여준다. TNR 지표에서는 다른 모델들에 비해 약간 낮은 수치를 보여주지만, 80%가 넘는 수치를 보여준다는 점에서 전체적으로 pre-pruning(CV) 모델이 가장 뛰어난 것을 알 수 있다. 특히, TPR 수치가 다른 모델들에 비해 압도적으로 높은 것을 알 수 있는데, 이는 실제 positive class인 것을 잘 예측한다는 것을 의미한다.

뿐만 아니라, 위에서 각각의 model을 plotting 하였을 때, pre-pruning tree(CV)의 복잡도가 낮은 것을 알 수 있다. 테스트셋에서의 성능도 가장 뛰어나며, 모델의 복잡도도 낮다는 점에서 최적의 결정나무는 교차검증을 진행한 pre-pruning Tree라고 할 수 있다.

pre-pruning(CV) Tree Plot은 다음과 같다.




대표적인 3가지 규칙은 다음과 같다.

(1) Glucose > 0.958 일 경우, class = 1이다.

(2) Glucose <= 0.958 and BMI <= -0.722 일 경우, class = 0이다.

(3) Glucose <= 0.958 and BMI > -0.722 and Glucose <= -0.638 일 경우, class = 0이다.

**[Q6](Neural Network) 동일한 데이터셋에 대하여 Neural Network 학습을 위해 필요한 최소 3가지 이상의 하이퍼파라미터를 선정하고, 각 하이퍼파라미터마다 최소 3개 이상의 후보값(최소 9가지 조합)을 사용하여 grid search를 수행한 뒤, 검증 데이터에 대한 AUROC 기준으로 최적의 하이퍼파라미터 조합을 찾아보시오.**

Neural Network 학습을 수행하기 위해 사용된 하이퍼파라미터는 다음과 같다.

1. hidden_layer_sizes: [(20,), (50,), (100,), (200,)]

    Neural Network에서 은닉층의 크기를 지정하는 하이퍼파라미터로, 이는 모델의 복잡도를 결정하는 중요한 요소 중 하나이다. 이때, 은닉층의 크기가 클수록 과적합의 위험성을 지니기에 적절한 크기를 선정하는 것이 필요하고, 따라서 여러 경우를 테스트하기 위해 20~200까지 다양한 수의 hidden_layer_sizes를 후보로 선정하였다.

2. learning_rate_init: [0.001, 0.01, 0.1]

learning_rate는 학습 속도를 결정하는 하이퍼파라미터로, 가중치 업데이트의 크기를 조절하여 학습 속도 및 수렴 속도를 조정할 수 있다. 이때, 너무 작은 학습률은 수렴 속도를 저하시키고, local minimum에서 나오지 못할 수도 있다. 또한, 너무 큰 학습률은 수렴하지 않고 발산할 가능성이 있으므로 학습률의 후보를 다음과 같이 3가지를 선정하였다.

3. activation: ['logistic', 'tanh', 'relu']

이 파라미터는 은닉층에서 사용할 활성화 함수를 결정한다. 이 때, logistic은 로지스틱 시그모이드 함수, tanh는 하이퍼볼릭 탄젠트 함수, relu는 ReLU(Rectified Linear Unit) 함수이다.

4. max_iter: [100, 200, 300]

이 파라미터는 iteration의 최대값을 결정한다. 이때, 많은 반복 횟수는 더 정확한 학습을 진행시켜 줄 수 있지만, 과적합의 위험이 있기 때문에 위와 같이 100, 200, 300 3가지의 경우를 나누어 살펴보고자 한다.

Grid search를 수행하고, 검증 데이터에 대한 AUROC 값을 기준으로 최적의 하이퍼파라미터 조합을 찾아보았고, 그 결과는 다음과 같다.

1. activation: 'tanh'

2. hidden_layer_sizes: (50,)

3. learning_rate_init: 0.001

4. max_iter: 300

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.8925이다.

여기서, [Q3]와 동일하게 추가로 동일한 하이퍼파라미터 조합 후보에 대하여 GridsearchCV를 이용하여 학습을 진행해보았다. 해당 과정을 추가로 수행한 이유는 validation 데이터셋이 154개로 많은 편이 아니기 때문에, 위에서 고른 하이퍼 파라미터 조합은 validation 데이터셋에는 최적화되어 있지만, 다른 데이터셋에 대해서는 최적화된 모델이 아닐 가능성이 있다. 따라서, 더 일반화된 모델을 선택하기 위해 교차 검증 방법으로도 최적의 하이퍼파라미터 조합을 찾아보았고 그 결과는 다음과 같다.

1. activation: 'relu'

2. hidden_layer_sizes: (20,)

3. learning_rate_init: 0.001

4. max_iter: 200

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.8775이다.

두 모델의 최적의 하이퍼파라미터 조합을 보았을 때, learning_rate_init를 제외한 모든 하이퍼파라미터가 다른 것을 확인하였고, 둘의 AUROC를 비교하였을 때, 교차검증을 진행한 모델이 약간 낮은 수치를 보이지만, 큰 차이가 없는 것을 확인하였다.

따라서, 2개의 하이퍼파라미터 조합에 대하여 테스트 데이터셋을 사용하여 TPR, TNR, Precision, Accuracy, BCR, F1-Measure, AUROC를 확인하고 가장 최적인 하이퍼파라미터 조합을 선택하고자 한다. 그 결과는 아래와 같다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Neural Network | 0.61818 | 0.72340 | 0.86869 | 0.77922 | 0.73281 | 0.66666 | 0.74343 |
| Neural Network(CV) | 0.58182 | 0.69565 | 0.85859 | 0.75974 | 0.70678 | 0.63366 | 0.72020 |

아래 CV라고 적혀있는 것이 교차검증을 진행한 모델이다.

둘을 비교해보았을 때, 교차검증을 진행하지 않은 모델이 교차검증을 진행한 모델보다 모든 평가지표에서 더 높은 성능을 보이는 것을 확인할 수 있다. 결론적으로, 교차검증을 진행하지 않은 모델의 파라미터가 최적의 파라미터라고 할 수 있고, 그 결과는 다음과 같다.

1. activation: 'tanh'

2. hidden_layer_sizes: (50,)

3. learning_rate_init: 0.001

4. max_iter: 300

**[Q7](Decision Tree/Neural Network공통) [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.**

우선, 로지스틱 회귀분석을 실시하여 test 데이터에 대한 분류 성능을 평가하였다. 그 결과는 다음과 같다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Logistic Regression | 0.58182 | 0.74419 | 0.88889 | 0.77922 | 0.71915 | 0.65306 | 0.73535 |

[Q3]에서 선택한 최적의 Pre-pruning Decision Tree 모델과 [Q6]에서 선택한 최적의 Neural Network, 그리고 로지스틱 회귀분석을 사용하여 학습 데이터를 학습하고, 테스트 데이터에 적용한 결과는 다음과 같다.

| Dataset | Model | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|---|
| Diabetes dataset | Logistic Regression | 0.58182 | 0.74419 | 0.88889 | 0.77922 | 0.71915 | 0.65306 | 0.73535 |
| | Decision Tree | 0.76364 | 0.72414 | 0.83838 | 0.81169 | 0.80014 | 0.74337 | 0.80101 |
| | Neural Network | 0.61818 | 0.72340 | 0.86869 | 0.77922 | 0.73281 | 0.66666 | 0.74343 |

이에 대한 결과 해석은 다음과 같다.

전체적으로 Logistic Regression과 Neural Network의 성능 지표가 비슷한 것을 알 수 있다. 공통적으로 TPR, F1-Measure 수치가 0.7을 넘지 못하는 것을 알 수 있고, 특히, TPR은 실제 positive class 객체들 중에서 positive class로 정확히 예측된 비율을 뜻하는데, 이 수치가 0.6에 근접한다는 것은 당뇨병 발병을 제대로 예측하지 못한다는 점에서 모델이 데이터셋을 제대로 분석하지 못했다고 할 수 있다.

반면, Decision Tree의 경우, TPR, Accuracy, BCR, F1-Measure, AUROC에서 가장 높은 수치를 보이는 것을 알 수 있다. 특히, TPR, BCR, F1-Measure, AUROC에 대해 압도적인 성능 차이를 보여주고 있고, 따라서 전체적인 성능이 가장 우수하다고 할 수 있다. 또한, BCR과 F1-Measure를 보았을 때, Decision Tree 모델은 예측 밸런스 측면에서도 가장 뛰어난 성능을 보이는데, 이는 모델이 양성 클래스와 음성 클래스를 밸런스 있게 잘 예측하고 있다는 것을 의미한다.

종합적인 성능에서 Decision Tree가 가장 우수하다고 할 수 있다. 또한, Q1에서 언급하였듯, 해당 데이터셋은 분류 성능도 중요하지만 결과물에 대한 해석이 중요한 데이터 셋인데 Decision Tree는 설명력에 있어서도 강점을 가지는 모델이다. 따라서, 예측 결과 뿐 아니라 해석에서도 뛰어난 성능을 지닌 Decision Tree가 당뇨병 데이터셋에 가장 적합한 모델이라고 할 수 있다.

**[Q8] 이번에는 본인이 생각하기에 "예측정확도"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 가이드 라인과 동일합니다.**

Dataset : **Mushroom Dataset**

다운로드 링크 : https://www.kaggle.com/datasets/prishasawhney/mushroom-dataset

선정 이유 : 해당 데이터셋은 54035개의 행과 9개의 변수로 이루어져 있으며, 버섯의 cap diameter, cap shape, Gill color 등 버섯의 여러 특징들을 토대로 식용 가능 여부를 판별하는 데이터셋이다.

앞의 diabetes 데이터셋의 경우, 데이터 분석 결과를 토대로 병을 치료하거나 예방하는 것이 목적이었기 때문에 데이터 분석 과정에서 유의미한 변수들을 찾아내고, 발병 원인을 찾아내는 것이 중요하였다.

그러나 본 문항에서는 데이터 분석 과정을 통해 버섯의 식용 여부에 어떤 변수가 중요한지 알아내는 것은 크게 중요하지 않다고 생각하였다. 이 데이터셋을 활용한 모델을 현실에 적용한다고 했을 때, 사람들이 궁금해하는 것은 '어떤 요소가 식용 여부에 영향을 미치는지'가 아닌 '식용 여부를 얼마나 정확하게 예측하는지'일 것이다. 식용 여부에 영향을 미치는 요소를 하나하나 안다고 해서 달라지는 것이 없기 때문이다. 따라서, 결과에 대한 해석이 힘들더라도 최대한 버섯의 식용 가능 여부를 정확하게 예측하는 모델을 구축하고자 한다.

해당 데이터셋은 1개의 종속 변수와 8개의 설명 변수가 있다.

<종속 변수>

- class : 버섯의 식용 여부 (0 : edible / 1 : poisonous)

<설명 변수>

- Cap Diameter
- Cap Shape
- Gill Attachment
- Gill Color
- Stem Height
- Stem Width
- Stem Color
- Season

**(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.**

먼저 데이터셋의 instance 수가 54035로 많기 때문에 random sampling을 통해 2000개의 데이터를 추출하였다. target class의 0,1 비율도 거의 비슷한 balance 데이터이고, 학습하고 평가하는데 있어 2000개도 충분하다고 생각하였기 때문에, random sampling을 진행하였다. 2000개의 data를 샘플링 한 후에 value_counts() 함수로 종속 변수의 label을 확인한 결과 1이 1078개, 0이 922개로 balance하게 샘플링 된 것을 확인하고 학습을 진행하였다.

이 데이터셋에서도 기본적으로 많이 사용되는 비율 중 하나인 60:20:20을 사용하였다. 즉, 1200:400:400 개의 데이터로 분할하였다. 그 이유는 다음과 같다.

먼저, 모델의 훈련에 사용되는 학습 데이터는 충분한 양이 필요하기 때문에 전체 데이터에서 가장 큰 비율인 60%를 학습에 할당하였다. 만약 더 큰 비율을 학습 데이터셋에 할당한다면, 상대적으로 검증 데이터와 테스트 데이터셋의 크기가 작아지게 된다. 앞에서 사용했던 diabetes 데이터셋에서 검증 데이터셋의 크기가 크지 않아 검증데이터 AUROC를 기준으로 선택한 모델이 테스트셋에서의 성능은 별로 좋지 않은 경우가 있었다. 즉, 검증 데이터셋의 크기가 충분하지 않다면 성능 측정이 잘못될 가능성이 있고, 결론적으로 최적의 모델을 찾는데 어려울 수 있다.

따라서 보편적으로 사용되는 또 다른 분배 비율인 80:10:10 대신, 학습에도 많은 데이터를 할당하면서 검증 및 테스트에도 적절한 비율로 데이터를 할당하는 60:20:20을 분배 비율로 선택하였고, 해당 분배 비율을 통해 모델의 훈련, 평가 및 테스트를 적절히 수행할 수 있다고 생각하였다.

**[Q9] (Decision Tree/Neural Network공통) [Q8]에서 선택한 데이터셋을 사용하여 [Q3]에서 수행한 최적의 pre-pruning Decision Tree모델 찾기, [Q6]에서 수행한 최적의 Neural Network모델 찾기를 동일하게 수행하시오.**

**9-1. 최적의 pre-pruning Decision Tree 모델 찾기**

[Q3]에서와 같은 방법을 활용하여 Pre-Pruning Decision Tree 모델을 탐색하였다. Pre-pruning을 수행하기 위해 사용된 하이퍼파라미터 후보는 다음과 같다. 세부사항은 [Q3]와 동일하다.

1. criterion ["gini", "entropy", "log_loss"]

2. min_samples_split [2, 5, 10, 25, 50, 100]

3. max_depth [2, 4, 8, 16]

4. min_samples_leaf [1, 2, 4, 8, 16]

최적의 하이퍼 파라미터 조합을 찾기 위해 validation data를 기준으로 AUROC를 측정하여 가장 높은 수치를 보이는 조합을 선택하였다.

그 결과, 하이퍼 파라미터에 대한 최적의 조합은 다음과 같다.

1. criterion: gini

2. min_samples_split: 25

3. max_depth: 16

4. min_samples_leaf: 4

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.76258이다.

여기서, 추가로 동일한 하이퍼파라미터 조합 후보에 대하여 GridsearchCV를 이용하여 학습을 진행해보았다. 즉, 5-fold 교차검증을 통해 AUROC를 계산하고, 가장 높은 수치를 보이는 하이퍼파라미터 조합을 탐색해보았다. 그 결과는 다음과 같다.

1. criterion: gini

2. min_samples_split: 10

3. max_depth: 16

4. min_samples_leaf: 4

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.80474이다.

둘의 결과를 보았을 때, 교차검증을 사용한 모델이 검증 데이터에 대한 AUROC 값이 더 높은 것을 알 수 있다. 2개의 하이퍼파라미터 조합에 대해 테스트 데이터셋을 사용하여 TPR, TNR, Precision, Accuracy, BCR, F1-Measure, AUROC를 확인하고 가장 최적인 하이퍼파라미터 조합을 선택하고자 한다. 그 결과는 아래와 같다

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Pre-Pruning | 0.76577 | 0.83333 | 0.80899 | 0.78500 | 0.78708 | 0.79812 | 0.78738 |
| Pre-Pruning(CV) | 0.83333 | 0.86047 | 0.83146 | 0.83250 | 0.83239 | 0.84668 | 0.83240 |

그 결과, 교차검증을 사용한 pre-pruning 모델이 모든 지표에서 더 뛰어난 성능을 보이는 것을 알 수 있다. 따라서, 최적의 Pre-Pruning Decision Tree 모델로 교차검증을 사용한 pre-pruning 모델을 사용하고자 하고, 이때의 하이퍼파라미터 조합은 다음과 같다.

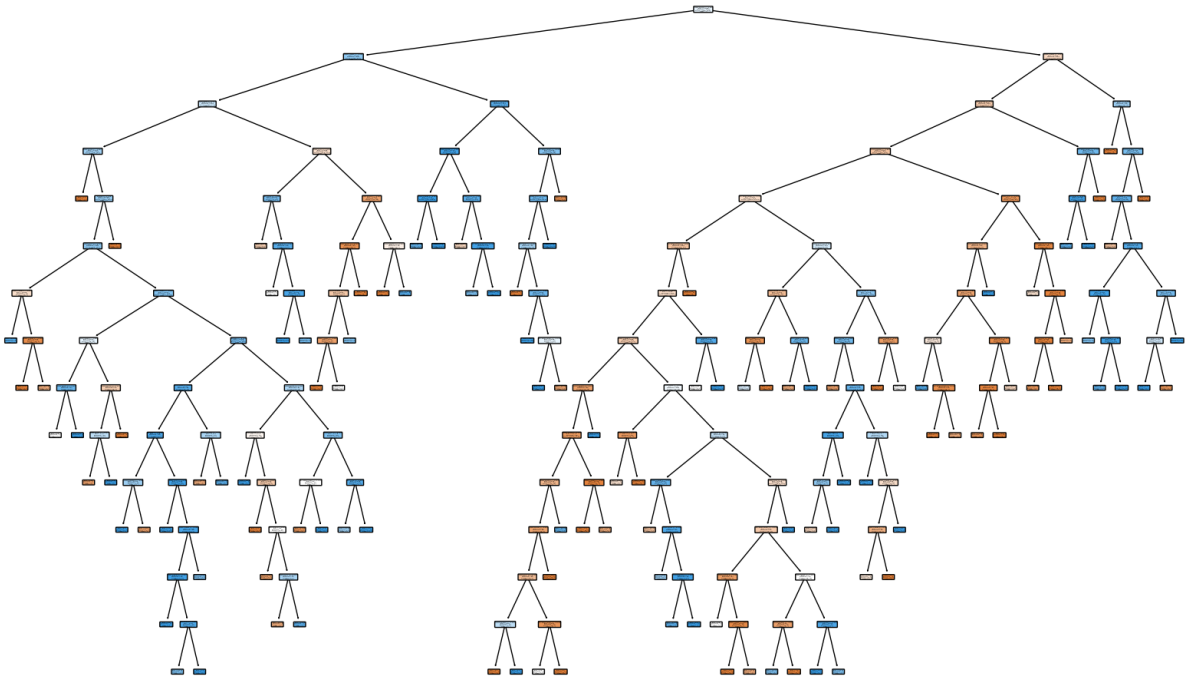
1. criterion: gini

2. min_samples_split: 10

3. max_depth: 16

4. min_samples_leaf: 4

교차검증을 사용한 pre-pruning decision tree를 plotting한 결과는 다음과 같다.



**9-2. 최적의 Neural Network 모델 찾기**

[Q6]에서와 같은 방법을 활용하여 Neural Network 모델을 탐색하였다. Neural Network를 수행하기 위해 사용된 하이퍼파라미터 후보는 다음과 같다. 세부사항은 [Q6]와 동일하다.

1. hidden_layer_sizes: [(20,), (50,), (100,), (200,)]

2. learning_rate_init: [0.001, 0.01, 0.1]

3. activation: ['logistic', 'tanh', 'relu']

4. max_iter: [100, 200, 300]

Grid search를 수행하고, 검증 데이터에 대한 AUROC 값을 기준으로 최적의 하이퍼파라미터 조합을 찾아보았고, 그 결과는 다음과 같다.

1. hidden_layer_sizes: (200,)

2. learning_rate_init': 0.01

3. activation: relu

4. max_iter: 300

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.95143이다.

여기서, [Q3]와 동일하게 추가로 동일한 하이퍼파라미터 조합 후보에 대하여 GridsearchCV를 이용하여 즉, 교차검증을 사용하여 학습을 진행해보았다. 그 결과는 다음과 같다.

1. hidden_layer_sizes: (200,)

2. learning_rate_init: 0.1

3. activation: tanh

4. max_iter: 200

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 AUROC 값은 0.93891이다.

둘의 결과를 보았을 때, 두 모델 모두 검증 데이터 AUROC 수치가 0.9를 넘는 것을 보아 뛰어난 성능을 보이는 것을 알 수 있고, 교차검증을 사용하지 않은 모델이 교차검증을 사용한 모델보다 검증 데이터에 대한 AUROC 값이 더 높은 것을 알 수 있다. 2개의 하이퍼파라미터 조합에 대해 테스트 데이터셋을 사용하여 TPR, TNR, Precision, Accuracy, BCR, F1-Measure, AUROC를 확인하고 가장 최적인 하이퍼파라미터 조합을 선택하고자 한다. 그 결과는 아래와 같다

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Neural Network | 0.83333 | 0.95361 | 0.94944 | 0.88500 | 0.88949 | 0.88942 | 0.89139 |
| Neural Network(CV) | 0.88739 | 0.91628 | 0.89888 | 0.89250 | 0.89312 | 0.9016 | 0.89313 |

전체적으로 보았을 때, 교차 검증을 사용한 모델이 사용하지 않은 모델보다 TPR에서 높은 수치를 보여주지만, Precision, TNR에서는 더 낮은 수치를 보이는 것을 알 수 있다. 전체적인 분류 성능을 확인할 수 있는 지표인 AUROC를 보았을 때, 두 모델의 차이가 거의 없음을 알 수 있다. 전체적으로 두 모델이 모두 뛰어난 성능을 보이고 있지만, 이번 mushroom 데이터셋의 경우, 종속 변수의 값을 보았을 때, 0과 1의 비율이 거의 비슷한 것을 알 수 있다. 따라서, positive class 예측과 negative class 예측을 balance 있게 잘하는 것이 중요하다고 생각하는데 F1-Measure, BCR, Accuracy가 모두 교차 검증을 사용한 모델이 높은 수치를 보이고 있음을 알 수 있고, positive class와 negative class를 더 밸런스 있게 예측한다고 할 수 있다.

결론적으로 두 모델 중, 밸런스 있게 예측을 하는 교차검증 사용 모델을 최적의 Neural Network 모델로 사용하고자 한다. 이때의 하이퍼파라미터 조합은 다음과 같다.

1. hidden_layer_sizes: (200,)

2. learning_rate_init: 0.1

3. activation: tanh

4. max_iter: 200

**[Q10] (Decision Tree/Neural Network공통) [Q8]에서 선택된 최적의 Pre-pruning Decision Tree 모델과 최적의 Neural Network, 그리고 로지스틱 회귀분석을 사용하여 학습데이터를 학습한 뒤, 테스트 데이터에 적용한 결과를 아래의 Confusion Matrix와 같이 작성하고 이에 대한 결과를 해석해보시오. 데이터셋 선정 당시 본인의 예상과 [Q7]의 결과표 및 아래 결과표가 일치하는지 확인해보시오. 일치하지 않는다면 왜 일치하지 않는지 그 이유를 서술해보시오 (일치 여부가 평가점수에 영향을 미치지 않음)**

우선, 로지스틱 회귀분석을 실시하여 test 데이터에 대한 분류 성능을 평가하였다. 그 결과는 다음과 같다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|
| Logistic Regression | 0.64865 | 0.66977 | 0.60112 | 0.62750 | 0.62443 | 0.65904 | 0.62489 |

[Q9]에서 선택한 최적의 Pre-pruning Decision Tree 모델과 최적의 Neural Network, 그리고 로지스틱 회귀분석을 사용하여 학습 데이터를 학습하고, 테스트 데이터에 적용한 결과는 다음과 같다.

| Dataset | Model | TPR | Precision | TNR | Accuracy | BCR | F1-Measure | AUROC |
|---|---|---|---|---|---|---|---|---|
| Mushroom dataset | Logistic Regression | 0.64865 | 0.66977 | 0.60112 | 0.62750 | 0.62443 | 0.65904 | 0.62489 |
| | Decision Tree | 0.83333 | 0.86047 | 0.83146 | 0.83250 | 0.83239 | 0.84668 | 0.83240 |
| | Neural Network | 0.88739 | 0.91628 | 0.89888 | 0.89250 | 0.89312 | 0.9016 | 0.89313 |

이에 대한 결과 해석은 다음과 같다.

Logistic Regression 모델은 세 모델 중, 모든 지표에서 가장 낮은 성능을 보이는 것을 확인할 수 있다. Neural Network는 세 모델 중, 모든 지표에서 가장 높은 성능을 보이는 것을 확인할 수 있고, 결론적으로 Neural Network > Decision Tree > Logistic Regression 순으로 성능이 뛰어난 것을 알 수 있다. 또한, Neural Network와 Decision Tree의 경우, 모든 지표에서 골고루 좋은 성능을 보인다는 점에서 mushroom 데이터의 target 변수를 밸런스 있게 잘 예측한다고 할 수 있다. 이러한 결론이 나오는 이유를 다음과 같이 설명할 수 있다.

Logistic Regression의 경우, input 변수와 target 변수 사이의 선형 관계를 가정하는 선형 모델이기 때문에 데이터가 복잡한 비선형 패턴을 가지고 있다면, 좋은 성능을 내기 어렵다. 즉, 해당 데이터셋은 선형적인 데이터가 아니라고 할 수 있고, 선형적 데이터에서 좋은 성능을 보이는 Logistic Regression은 해당 데이터셋에서 낮은 성능을 보인다고 해석할 수 있다.

반면, Neural Network와 같은 신경망 모델은 여러 층과 많은 뉴런을 통해 데이터의 복잡한 패턴과 상호작용을 포착할 수 있고, 따라서 복잡한 비선형 관계를 잘 학습할 수 있다. 결론적으로 충분한 데이터와 적절한 하이퍼파라미터 튜닝이 이루어진다면 가장 좋은 성능을 보일 가능성이 높은데, 해당 데이터셋에서 모든 평가 지표가 가장 뛰어난 결과를 보여주는 것을 보아, mushroom 데이터셋에 대한 적절한 하이퍼파라미터 튜닝이 이루어졌다고 할 수 있다.

Decision Tree의 경우, 데이터에 대한 분할 규칙을 반복적으로 적용하여 비선형적인 데이터 패턴을 처리할 수 있다. 따라서 선형 결합을 기반으로 하는 logistic regression보다 더 복잡한 패턴을 학습할 수 있고, 결론적으로 mushroom 데이터셋에서 logistic regression보다 더 높은 성능을 보이는 것을 알 수 있다.

다음으로 [Q7]과 [Q10]의 결과를 보고 3가지 모델 (Decision Tree, Logistic Regression, Neural Network)에 대해 종합적인 평가를 하고자 한다. diabetes 데이터셋에 대해서는 Decision Tree가 가장 좋은 성능을 보였고, Neural와 Logistic Regression의 경우 다소 낮은 성능을 보여주었다. 반면, mushroom 데이터셋에 대해서는 Neural Network가 가장 좋은 성능을 보였고, Decision Tree 또한 준수한 성능을 보였다. mushroom 데이터셋에서 logistic Regression은 다른 두 모델보다 현저히 낮은 성능을 보여주었다.

결론적으로 현실 데이터에는 비선형적인 데이터들도 많이 존재하는데, Logistic regression은 비선형적인 데이터에는 낮은 성능을 보이고, 이러한 경우 적합하지 않다고 할 수 있다.

Decision Tree와 Neural Network의 경우 비선형적인 데이터를 학습할 수 있기 때문에, 전체적으로 좋은 성능을 보일 수 있다. Decision tree는 복잡한 데이터에 대해서 Neural Network보다 낮은 성능을 보일 수 있지만, 예측 결과물에 대한 해석이 가능하므로 다소 낮은 성능을 보이더라도 충분히 메리트가 있는 모델이라고 할 수 있다. Neural Network는 다양한 구조와 활성화 함수를 사용하여 복잡한 관계를 모델링 할 수 있지만, diabetes와 같이 데이터의 수가 적은 경우, 다소 낮은 성능을 보일 수 있다. 또한 가장 높은 성능을 보이는 경우에도 모델 해석이 어려울 수 있기 때문에, 모델에 대한 해석이 중요한 경우, decision tree의 성능이 다소 떨어지더라도 decision tree를 사용하는 것이 적절할 수 있다.

마지막으로 데이터셋 선정 시 나의 예상과 Q7의 결과표와 Q10의 결과표가 일치하는지 확인하고자 한다.

먼저, 첫번째 데이터인 diabetes 데이터를 선정할 때, 예측 결과물도 중요하지만 결과 해석이 중요한 데이터를 기준으로 선정하였다. 이때, 해석과 예측에 적합한 모델이 Decision Tree라고 생각하였는데, 그 이유는 Decision Tree가 비선형 관계를 포착할 수 있고, 분류 기준이 명확하여 데이터의 중요한 특징을 시작적으로 파악하기 쉽기 때문이다. 즉, 결과에 대한 해석이 용이하고, 비선형 데이터에 대해 예측도 잘 해줄 것이라 생각하였기에 diabetes 데이터셋에 decision tree 모델이 가장 적합할 것으로 예상하였다. 하지만, 예측에 있어서 가장 성능이 뛰어날 것으로 예상한 모델은 Neural network였다. Neural network가 비선형적이고 복잡한 데이터 패턴을 잘 학습하기 때문에, Neural network가 가장 뛰어난 성능을 보일 것이고, 다음으로 decision tree가 높은 성능을 보일 것으로 예측하였다. 그렇더라도 둘의 성능 차이가 크지 않다면 해석이 용이한 decision tree가 해당 데이터셋에 적합하다고 결론을 내리려 했지만, 전체적인 성능 또한 decision tree가 가장 좋았기 때문에 예측 및 해석에 있어 고민 없이 decision tree가 가장 적합한 모델이라고 결론을 내릴 수 있었다. Neural Network가 예상과 다르게 다소 낮은 성능을 보인 이유는 데이터셋의 크기가 작아서 일 가능성이 있다. 데이터의 수가 많지 않은 편이라 데이터 패턴을 파악하는데 있어 어려움이 있고, 적은 데이터로 인해 과적합 될 가능성이 존재한다. 또한, 하이퍼파라미터 튜닝이 제대로 이루어지지 않았을 가능성도 있다. 더 많은 종류의 하이퍼파라미터를 넓은 범위로 Grid Search한다면 더 좋은 성능을 보여줄 수 있을 것이라고 생각한다.

두번째 데이터인 mushroom 데이터를 선정할 때, 예측 정확도가 예측 결과물에 대한 해석보다 훨씬 중요할 것 같은 데이터를 기준으로 선정하였다. 이때, 위와 동일하게 예측 정확도가 가장 뛰어날 것이라고 생각한 모델은 Neural network였다. 그 이유는 neural network는 hidden layer와 node로 복잡한 내부 구조를 만들 수 있고, 다양한 구조와 활성화 함수를 통해 복잡한 관계도 모델링할 수 있기 때문이다. 다음으로 Decision Tree가 Logistic Regression에 비해 복잡한 패턴을 더

잘 학습한다는 점에서 Neural network 다음으로 Decision Tree가 성능이 뛰어날 것으로 예상하였고, 비선형적 패턴을 잘 잡아내지 못하는 Logistic Regression은 다소 낮은 성능을 보일 것으로 예측하였다. 해당 데이터셋의 경우, 모든 성능 지표에서 Neural Network의 성능이 가장 뛰어나고, Logistic Regression의 성능이 가장 낮은 것을 보아 예상과 정확히 맞아 떨어지는 것을 알 수 있었다. diabetes 데이터셋보다 mushroom 데이터셋의 크기가 더 크다는 점에서 Neural network가 학습하기 더 좋았을 것이라 생각하고, 하이퍼파라미터 조합 또한 적절하게 잘 찾아냈다고 생각한다. 또한, Logistic Regression의 결과가 다른 두 모델에 비해 현저히 낮은 성능을 보여준다는 점에서 mushroom 데이터셋은 독립변수와 종속변수 간의 관계를 선형 함수로 잘 나타낼 수 없는 비선형 데이터셋이라는 해석도 가능하다.