

# 다변량분석\_과제#3

산업경영공학부  
2019170802 임청수

## <목차>

1. 데이터 출처와 선정이유
  - 1.1 데이터 출처 & 변수 설명
  - 1.2 데이터 선정 이유 & 분배 비율 근거
  - 1.3 기초 전처리
2. Full Tree와 Post-pruning Tree의 분류 성능 비교
  - 2.1 Full Tree & Post-pruning Tree 학습
  - 2.2 결과 비교 및 해석
3. Pre-Pruning Tree
4. Tree plotting
  - 4.1 두 모델의 Tree plotting 및 해석
5. 최적의 의사결정나무 plotting & rule 설명
  - 5.1 plotting
  - 5.2 세 가지 규칙 설명
6. Neural Network grid search
  - 6.1 Grid search 결과 분석
7. 세 모델의 성능 비교
  - 7.1 세 모델의 5가지 평가지표 결과
  - 7.2 결과 해석
8. 새로운 데이터로 모델링
  - 8.1 데이터 출처와 변수 설명
  - 8.2 선정이유 & 분배 비율 근거
9. Pre-pruning & Neural Network
10. 세 모델 결과 비교
  - 10.1 세 모델의 5가지 평가지표 결과 & 결과해석

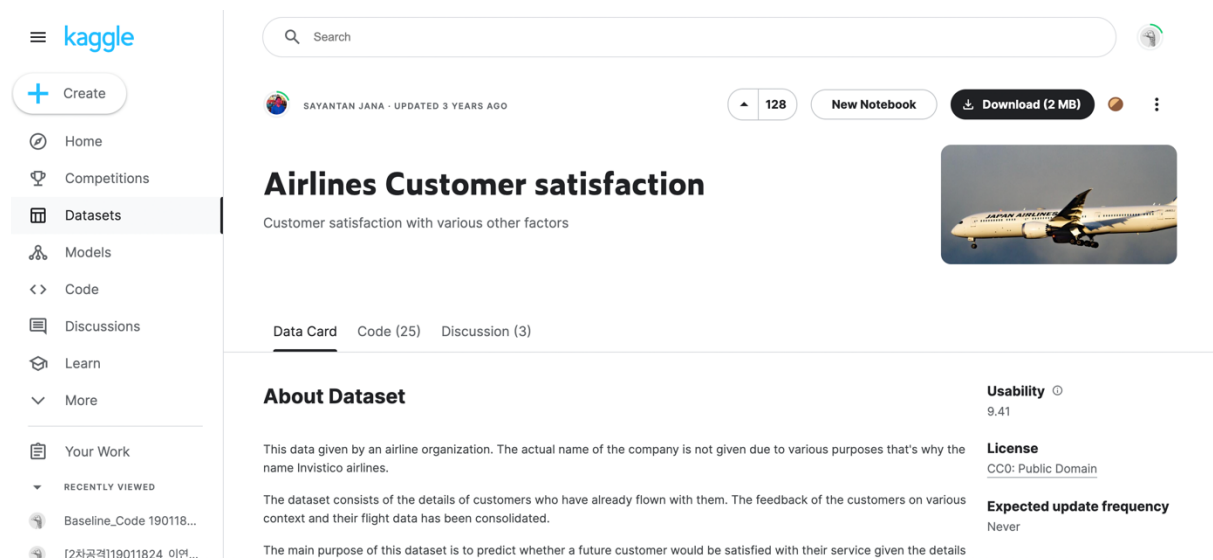
# 1. 데이터 출처와 선정이유

## 1.1 데이터 출처 & 변수 설명

데이터는 kaggle에서 다운받을 수 있다.

다운 받을 수 있는 링크는 아래와 같다.

<https://www.kaggle.com/datasets/sjleshrrac/airlines-customer-satisfaction/code>



The screenshot shows the Kaggle dataset page for 'Airlines Customer satisfaction' by user SAYANTAN JANA. The page includes a sidebar with navigation links (Home, Competitions, Datasets, Models, Code, Discussions, Learn, More, Your Work) and a main content area. The dataset title is 'Airlines Customer satisfaction' with a subtitle 'Customer satisfaction with various other factors'. It features a 'Data Card' tab, a 'Code (25)' tab, and a 'Discussion (3)' tab. The 'About Dataset' section describes the data as being from an airline organization, with details on customer feedback and flight data. The 'Usability' score is 9.41, the 'License' is CC0: Public Domain, and the 'Expected update frequency' is 'Never'. A 'Download (2 MB)' button is also visible.

Index	Column	내용	Example	Category	Datatype
0	Gender	고객 성별	Male/Female	Categorical	Object
1	Customer Type	충성 고객 여부	Loyal/disloyal	Categorical	Object
2	Type of Travel	비행의 목적	Business / personal	Categorical	Object
3	Class	좌석 클래스	Business, eco, other	Categorical	Object
4	Flight Distance	비행거리	50~6951	Numerical	Int64
5	Seat comfort	좌석 편안도	0~5	Categorical	Int64
6	Departure/Arrival time convenient	출발/도착 시간의 만족도	0~5	Categorical	Int64
7	Food and drink	식음료 제공 만족도	0~5	Categorical	Int64
8	Gate location	게이트 위치의 만족도	0~5	Categorical	Int64
9	Inflight wifi service	와이파이 서비스 만족도	0~5	Categorical	Int64
10	Inflight entertainment	오락시설 만족도	0~5	Categorical	Int64
11	Ease of Online booking	온라인 예약 만족도	0~5	Categorical	Int64
12	Leg room service	다리 공간 만족도	0~5	Categorical	Int64
13	Baggage handling	수하물 처리 속도 만족도	0~5	Categorical	Int64
14	Check-in service	체크인 서비스	0~5	Categorical	Int64
15	Cleanliness	청결	0~5	Categorical	Int64
16	Departure Delay in Minutes	출발 지연 시간	0~1592	Numerical	Int64
17	Arrival Delay in Minutes	도착 지연 시간	0~1584	Numerical	Int64
18	satisfaction	만족도(target)	Sati / dissati	Categorical	object
19	Age	나이	7~85	Numerical	Int64
20	On-board service	기내 서비스	0~5	Categorical	Int64
21	Online boarding	온라인 탑승 만족도	0~5	Categorical	Int64
22	Online support	온라인 지원 만족도	0~5	Categorical	Int64

## 1.2 데이터 선정 이유 & 분배 비율 근거

---

### <데이터 선정 이유>

주어진 문제에서 예측 정확도도 중요하지만 “예측 결과물에 대한 해석”이 매우 중요할 것으로 생각되는 분류 문제를 다루고 있는 데이터셋을 선정하는 것을 요구했다. 따라서 우선 분류 문제에 사용할 수 있는 target이 categorical한 데이터셋을 검색했고 분류의 목적을 생각했을 때 왜 이렇게 분류되었는지, 분류 되는 규칙이 무엇인지를 해석할 필요가 있는 데이터셋을 위주로 찾았다.

따라서 선택한 데이터셋은 kaggle의 Airlines Customer Satisfaction 데이터이다.

항공사는 고객에게 좋은 서비스를 제공하고 높은 고객 만족도를 통해 충성 고객을 확보하는 것이 중요하기 때문에 고객이 어떤 서비스에 만족하고 불편함을 느끼는지를 잘 분석해내는 것이 중요하다. 따라서 주기적으로 고객 만족도 데이터를 분석하고 해석하여 서비스를 개선해나가는 과정이 반드시 필요하다.

따라서 얼마나 높은 만족도 정확도를 예측해내는지도 중요하지만 어떤 특성이 직접적으로 만족도를 높이는데 영향을 주는지, 어떤 특성이 불만족에 영향을 주는지를 결과물을 보고 해석해내는 것이 매우 중요한 데이터셋이라고 생각하여 Airlines Customer Satisfaction을 사용하기로 결정했다.

### <데이터 분배 비율 근거>

Seed를 설정한 후 train : valid : test = 60 : 20 : 20으로 분배해주었다. 그 이유는 데이터 양이 많지 않기 때문에 최대한 train data가 많으면 성능 향상에 도움이 되지만 validation data와 test data도 일정 양 이상 확보 되어야 하기 때문이다.

따라서 일정 비율을 유지하는 것으로 결정했고 분배한 비율인 6 : 2 : 2에 대한 근거는 전통적인 머신러닝 방법론에서 주로 train : test : val = 6 : 2 : 2로 나누기 때문에 동일한 비율을 사용하기로 결정했다.

결과적으로 training data = 2033개, validation data = 678개, test data = 678개로 나뉘고 각 데이터를 분리할 때 stratify = airline\_target으로 설정하여 실제 y값의 분포 비율을 유지한 상태로 데이터를 분리할 수 있도록 설정했다.

## 1.3 기초 전처리

---

데이터분석을 진행하기에 앞서 모델링에 적합하도록 전처리를 수행했다.

## 1. Null 값이 포함된 행 제거

우선 전체 데이터 3896개에서 Null값이 포함된 행을 제거하여 12개의 데이터를 제거했다.

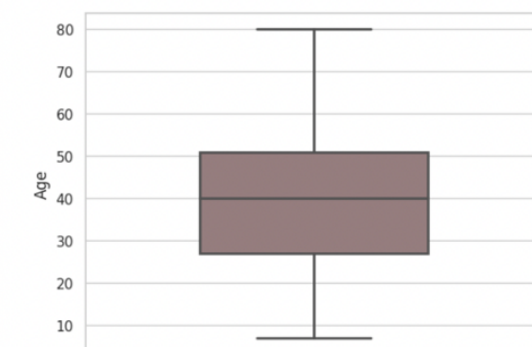
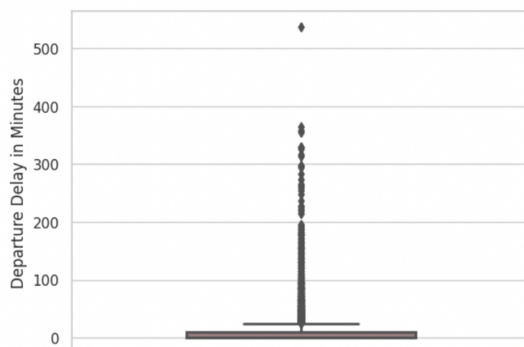
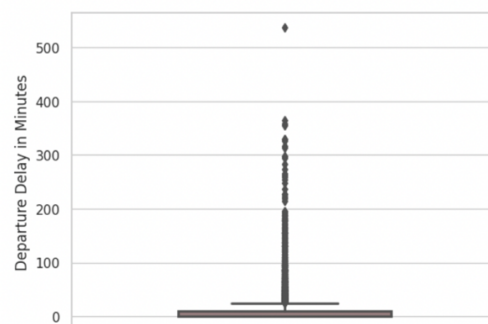
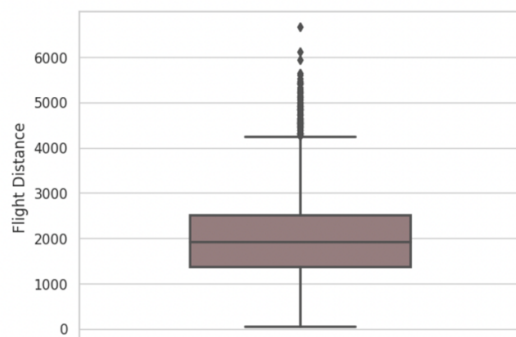
제거한 이유는 Decision Tree와 Logistic Regression, Neural Network 총 3개의 모델에 대해 동일한 데이터셋으로 학습을 하고 성능 비교를 수행하는데 모델마다 null값에 영향을 받는 정도가 다르기 때문이다.

모델마다 전처리를 다르게 하면 안되므로 null값의 영향을 많이 받는 logistic regression을 고려하여 행 자체를 제거했다.

## 2. 이상치 제거

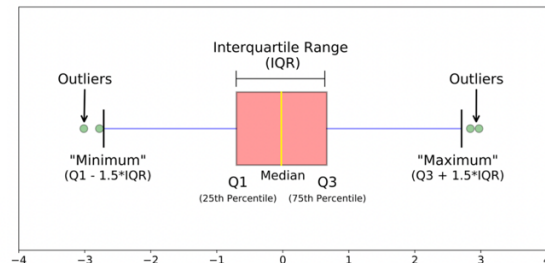
마찬가지로 Decision Tree는 이상치에 영향을 덜 받지만 Logistic Regression은 영향을 많이 받기 때문에 제거했다.

이상치는 연속형 변수들의 box plot을 그렸을 때 boxplot에서 많이 벗어나는 데이터를 식별하여 확인할 수 있었다.



제거한 기준은 box plot에서 Q1(25 percentile)부터 Q3(75 percentile) 사이 간격을 IQR(interquartile range)로 정의한다. 본 과제에서는 아래 식에 해당하는 범위를 정상 데이터로, 해당하지 않는 데이터를 이상치로 정의했다.

$$Q1 - 1.5 * IQR \leq value \leq Q3 + 1.5 * IQR$$



제거한 결과 제거 전 3884개 행에서 제거 후 3389개 행으로 총 495개의 행이 outlier에 해당하여 제거할 수 있었다. 전체 데이터에서 비중을 생각했을 때 과하지 않다고 판단하여 3389개의 데이터로 분석을 진행하기로 결정했다.

### 3. One hot encoding

데이터 중 object 형태로 존재하는 데이터는 Gender, Customer Type, Type of Travel, Class, satisfaction이다. 따라서 one hot encoding을 통해 수치형으로 바꾸고 동시에 여러 개의 항목인 경우는 서로 다른 column으로 구분했다. Drop\_first=True를 적용하여 다중공선성을 방지했고 결과적으로 총 3389개 행과 24개의 열로 구성된 데이터프레임을 데이터셋으로 사용했다.

## 2. Full Tree와 Post-pruning Tree의 분류 성능 비교

### 2.1 Full Tree & Post pruning Tree 학습

Full Tree 모델은 사이킷런의 DecisionTreeClassifier를 사용했고 random\_state = 12345로 설정했다. 학습데이터만 사용해서 학습을 수행했다.

학습 결과 max\_depth는 23, node\_count는 493개임을 확인할 수 있었다.

Post pruning Tree 모델은 training data를 사용하여 학습한 후 ccp\_alpha 값을 조절하면서 validation data를 통해 Post pruning을 시행하여 best validation accuracy를 갖는 모델을 선택했다.

## 2.2 결과 비교 및 분석

다음은 7가지 평가 지표를 활용하여 두 모델의 성능을 비교한 표이다.

	TPR	Precision	TNR	Accuracy	BCR	F1-measure	AUROC
Post-Pruning	0.8798	0.8821	0.8474	0.8657	0.8635	0.8810	0.8637
Full tree	0.6969	0.6611	0.9780	0.9353	0.7699	0.6324	0.8791

성능평가 결과, 지표별로 큰 차이를 보였다. 따라서 지표별로 해석을 수행했다.  
성능 지표 별 해석은 아래와 같다.

용어	산출식	설명	예
Accuracy	$(TP+TN)/(TP+TN+FP+FN)$	탐지율 : 맞게 검출한 비율	실제 양성/정상인지 맞게 예측한 비율
Precision	$TP/(TP+FP)$	정확도 : P로 검출한 것중 실제P의 비율	악성으로 예측한것 중 실제 악성인 샘플의 비율
Recall	$TP/(TP+FN)$	재현율 : 실제 P를 P로 예측한 비율	실제 양성 샘플 중 악성으로 예측한 비율
False Alarm (Fall-out)	$FP/(FP+TN)$	오검출율 : 실제 N을 P로 예측한 비율	실제 정상 샘플을 악성으로 예측한 비율
TPR (True Positive Rate) = Recall	$TP/(TP+FN)$	예측과 실제 모두 P	실제 양성 샘플을 악성으로 예측한 비율
TNR (True Negative Rate)	$TN/(TN+FP)$	예측과 실제 모두 N	실제 정상 샘플을 정상으로 예측한 비율
FPR (False Positive Rate) = False Alarm	$FP/(FP+TN)$	실제 N인데 P로 검출	실제 정상 샘플을 악성으로 예측한 비율
FNR (False Negative Rate)	$FN/(TP+FN)$	실제 P인데 N으로 검출	실제 양성 샘플을 정상으로 예측한 비율

Tpr은 true positive rate = recall을 의미하고  $tp/(tp+fn)$ 으로 계산한다. 즉 예측과 실제 모두 p인 경우를 의미한다. 실제로 양의 값을 가질 때 양으로 예측해낼 확률을 의미하며 post pruning이 더 높은 값을 나타냈다.

Precision은 post pruning이 더 높음에 반해 TNR은 full tree가 더 높은 지표를 보여주었다. 따라서 다른 지표를 통한 추가적인 해석이 필요하다.



Accuracy는 post pruning이 더 높은 값을 보여주었다.

BCR은 balanced classification rate를 의미한다. Post pruning이 더 높은 값을 보여주었다.

F1-measure는 precision과 recall을 통합하여 계산한 평가지표이다. 계산식은 아래와 같다.

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

따라서 precision과 recall의 단점들을 서로 보완해주기 때문에 매우 중요한 지표로 사용할 수 있다.

결과적으로 post-pruning이 더 높은 값을 보여주었다.

Auroc는 full tree가 미세하게 앞섰다.

전반적으로 지표가 많이 엇갈려서 우위를 가리기 어렵다고 판단했다.

### 3. Pre-pruning Tree

#### 3.1 pre-pruning Tree 학습

---

<사용된 하이퍼파라미터>

##### 1. Criterion

Pruning을 판단할 criterion은 총 세 가지에 대하여 탐색 범위를 설정했다. 그 이유는 DecisionTreeClassifier의 criterion은 "gini", "entropy", "log\_loss"만 설정 가능했기 때문이다. Pruning 기준을 어떻게 설정하느냐에 따라 split했을 때의 impurity가 달라지기 때문에 어떤 criterion을 설정하느냐에 따라 pruning의 결과가 유의미하게 달라진다. 특히 entropy는 세 기준 중 가장 넓은 범위를 표현할 수 있기 때문에 더 설명력이 뛰어나다고 볼 수 있다. 따라서 best model은 어떤 criterion을 사용할지가 궁금하여 pruning에 사용했다.

## 2. Min\_split

The minimum number of samples required to split an internal node:

Internal node를 split할 때 요구되는 샘플의 최소 수를 의미한다. 즉 min\_split을 작게 설정하면 split 수가 많아지고 크게 설정하면 설정된 수치 이상으로 sample이 나뉘면 더 이상 split 하지 않는다. 따라서 어떤 값을 설정하느냐에 따라 tree의 depth에 큰 영향을 미친다. Default값은 2이지만 데이터 크기가 2000을 넘기 때문에 좀 더 크게 설정해도 될 것 같다는 생각이 들었다. 따라서 10, 30, 50, 70, 90, 100으로 넉넉하게 설정했고 sample을 잘게 나누지 않았을 때 복잡도가 낮아져 더 높은 성능을 보여주는지 확인하고자 pruning hyperparameter로 사용했다.

## 3. Max\_depth

노드의 깊이가 깊어질수록 복잡도가 증가하기 때문에 최대 depth를 제한하여 pruning을 할 수 있다. 깊이의 범위는 앞서 full tree의 depth가 16이었기 때문에 최소 1에서 최대 16까지 탐색 범위를 설정했다. 총 5가지 경우에 대하여 [1,5,8,11,16]으로 탐색 범위를 지정했다.

Pre-pruning은 학습데이터와 검증 데이터를 이용하여 진행했고 auroc 값을 기준으로 최적의 하이퍼파라미터 조합은 아래와 같다.

하이퍼파라미터	값
Best criterion	Gini
Best min_samples_split	70
Best max_depth	11

Entropy가 아닌 gini가 선정된 것이 의외였고 데이터셋마다 적절한 criterion이 있을 수 있다는 것을 알게 되었다. Best min\_samples\_split은 70으로 너무 잘게 쪼개는 것이 정답이 아니라는 것을 알 수 있었다. Best max\_depth는 11이었고 기존 16에 비해 더 단순해지면서 높은 성능을 얻게 되었음을 알 수 있었다.

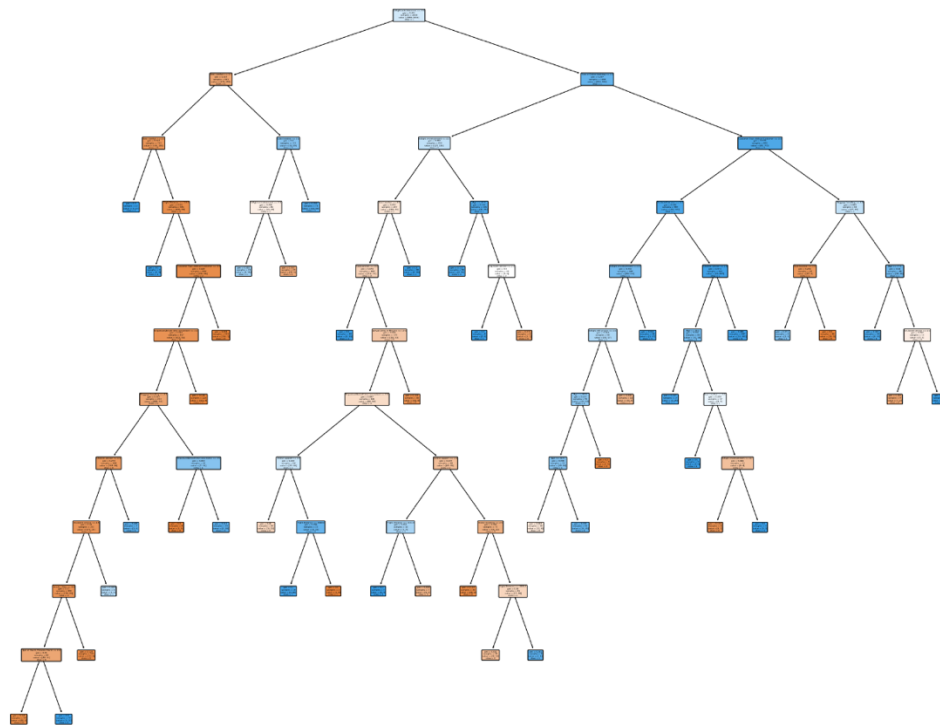
## 4. Tree plotting

## 4.1 두 모델의 Tree plotting

두 트리 모두 층이 깊고 많은 노드를 가지고 있기 때문에 한눈에 파악하기는 어려웠다. 따라서 두 트리의 차이점이 두드러지게 느껴지는 지점을 위주로 나타내었다.

<post pruning tree plotting>

전체 노드의 형태는 아래와 같다.

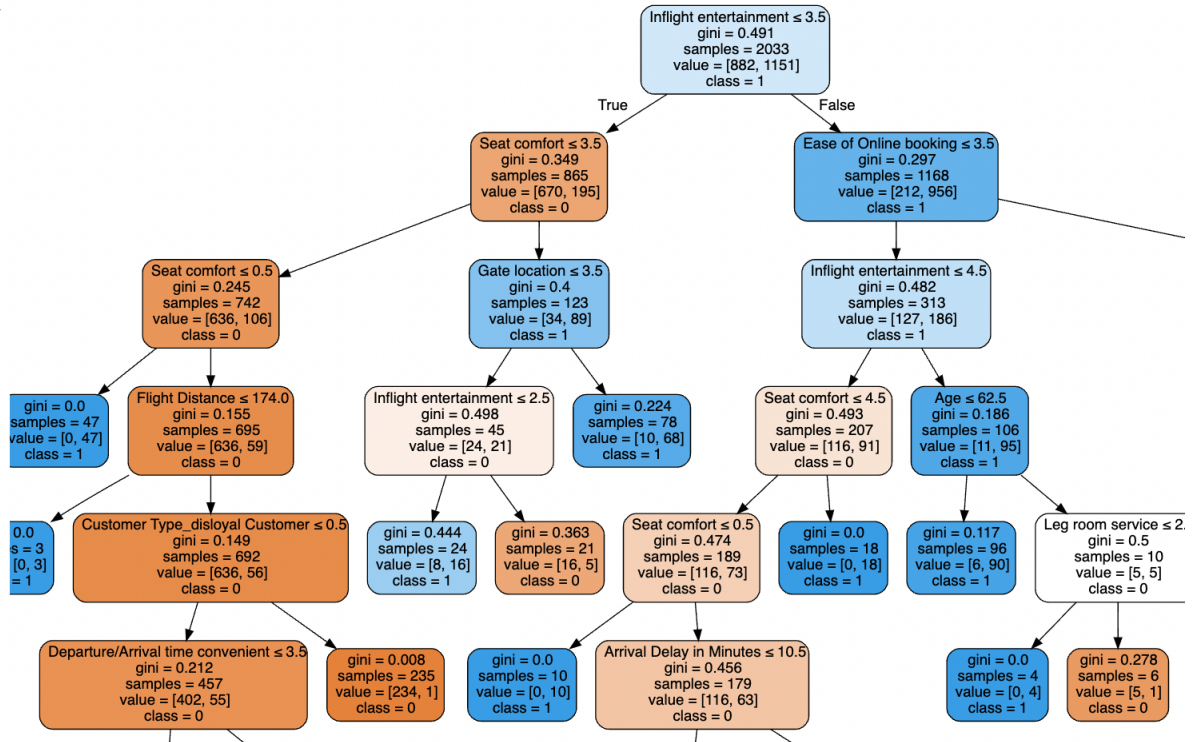


Max\_depth = 11

Leaf node = 46

아래는 전체 중 상위 노드 일부를 캡처한 사진이다.

□



#### <해석>

1. 가장 상위 split node는 inflight entertainment  $\leq 3.5$ 를 사용했다. 그리고 바로 하위 노드로는 seat comfort  $\leq 3.5$ 와 ease of online booking  $\leq 3.5$ 가 사용되었다.

이때 만족도 = 1인 경우는 기내 오락시설에 높은 만족도를 보이고, 온라인 예약 시스템에 높은 만족도 값을 가지면 gini = 0.179, values = [85, 770]로 상당히 높은 비율로 만족한다는 것을 확인할 수 있었다.

그 이유를 생각해볼 때, 만족도는 설문조사를 통해 진행하는데 설문조사를 각 항목마다 따져가면서 체크를 하는 사람이 있는 반면 다 만점 쥘버리는 사람이 있다.

반대로 inflight entertainment에 낮은 점수를 준 사람이라면 문항 자체를 꼼꼼하고 간간하게 보면서 체크를 한다는 것이고 만족도 또한 불만족으로 투표할 가능성이 있다.

생각보다 판단 기준이 명확한 기준이 아니라 조금 애매할 수도 있는 기준이 상위 split node의 위치했다는 점에서 개인적인 예상과 많이 빗나간 것 같다.

2. 상위노드 중 class = 0 즉 불만족 sample이 다수 포함되어 있는 노드의 기준은 seat comfort가 여러 번 등장했고, flight distance가 큰 경우, customer type = disloyal, departure/arrival time convenient 만족도가 낮은 경우가 있었다.

좌석이 불편하면 비행 시간 내내 계속 불편함을 느끼기 때문에 최종적으로 불만족한다고 평가하게 되는 것으로 느껴졌다.

마찬가지로 비행 시간이 길어질수록 불편함이 커지기 때문에 불만족 평가를 하는데 큰 영향을 주는 것 같다.

또한 손님이 disloyal인 경우는 loyal 고객은 만족할 가능성이 높고 검증된 고객이기 때문에 disloyal에서 불만족이 많이 발생한다고 해석할 수 있었다. 마지막으로 출발, 도착 시간에 지연으로 불편함이 있다면 전반적인 비행 전체에 불만족을 느끼는 것을 알 수 있었다.

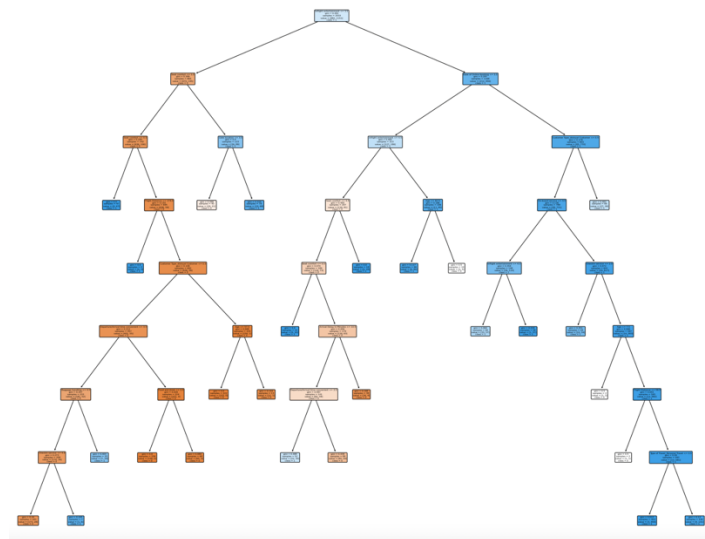
3. 마찬가지로 class = 1에 해당하는 상위 split node의 기준은 ease of online booking 점수가 높을수록, inflight entertainment 점수가 높을수록, age가 62.5세 이하일수록, customer type = loyal일 경우에 해당했다.

온라인 예약 시스템이 편하면 만족할 가능성이 높았다. 그리고 inflight entertainment는 기내에 어떤 재미 요소가 있었는지는 모르겠지만 해당 점수가 높으면 높을수록 전반적인 만족도에 긍정적인 영향을 주는 것 같다.

고령일수록 비행에 많은 불편함을 느낄 것으로 예상된다. 따라서 62.5세를 기준으로 이하면 만족한다고 평가하는 sample이 많아지고 62.5세 이상이면 불만족의 sample이 늘어나도록 분류되었다. Customer type은 불만족 = 0 을 분류할 때와 마찬가지로 loyal일 때는 만족하는 sample이 늘어났고 disloyal일 때는 불만족하는 sample이 증가했다.

<pre pruning tree plotting>

전체 트리의 형태는 아래와 같다.



Depth = 8

Leaf node = 27개이다. Full tree보다 훨씬 모델이 단순해졌음을 알 수 있다.

<해석>

위 post pruning한 모델과 많이 유사한 형태를 보였다. Inflight entertainment가 최 상위 노드에 위

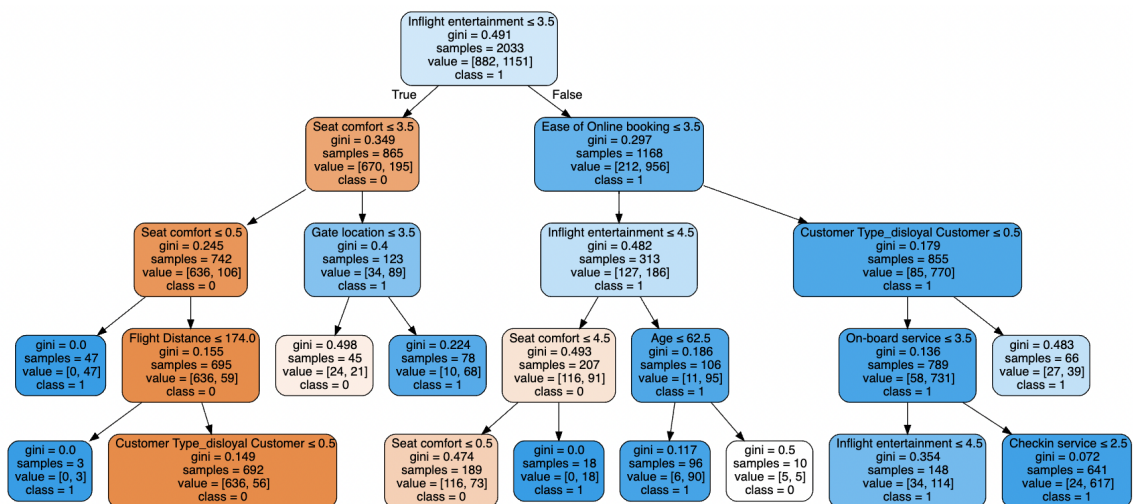
치했고 상위 노드에 여러 번 나타나면서 class = 1을 분류해내는데 중요한 역할을 하는 feature임을 알 수 있었다.

Class = 0 불만족을 분류해내는 feature로는 seat comfort, flight distance, customer type = disloyal 이 마찬가지로 사용되었다.

Class = 1 만족을 분류해내는 feature로는 ease of online booking, age < 62.5, customer type = loyal 등 위 post pruning과 동일한 특성들이 중요하게 사용됨을 알 수 있었다.

새롭게 추가된 feature는 check in service로 점수가 높을수록 고객들이 만족하는 sample이 다수 존재함을 알 수 있었다.

다만 최종적으로 depth는 3만큼, leaf node 수는 19개 감소했기 때문에 상위노드는 비슷하지만 하위노드를 좀 더 세분화하여 rule을 사용한 것이 post pruning tree임을 알 수 있었다.



## 5. 최적의 의사결정나무 plotting & rule 설명

### 5.1 plotting

---

앞서 계산한 full tree, post pruning tree, pre pruning tree의 평가지표는 아래와 같다.

|              | TPR    | Precision | TNR    | Accuracy | BCR    | F1-measure | AUROC  |
|--------------|--------|-----------|--------|----------|--------|------------|--------|
| Post-Pruning | 0.8798 | 0.8821    | 0.8474 | 0.8657   | 0.8635 | 0.8810     | 0.8637 |
| Full tree    | 0.6969 | 0.6611    | 0.9780 | 0.9353   | 0.7699 | 0.6324     | 0.8791 |
| Pre Pruning  | 0.8642 | 0.8826    | 0.8508 | 0.8584   | 0.8575 | 0.8733     | 0.8575 |

지표들이 거의 비슷했지만 전반적으로 높게 나타난 값이 많은 post pruning을 best decision tree로 선정했다.

대표적인 rule 3가지를 설명하고자 한다.

<rule 1>

Inflight entertainment  $\leq 3.5$  & seat comfort  $\leq 3.5$  & gate location  $\geq 3.5$  면 class = 1이다.

Inflight entertainment와 seat comfort가 부정적인 지표를 나타내지만 gate location가 긍정적인 지표값을 나타내어 만족하게 됨을 알 수 있다.

<rule 2>

Inflight entertainment  $> 3.5$  + ease of online booking  $\leq 3.5$  + Inflight entertainment  $\leq 4.5$ 이면 class = 1

따라서 Inflight entertainment과 ease of online booking이 고객이 만족하는데 영향을 많이 줌을 알 수 있다.

Inflight entertainment  $> 3.5$  + ease of inline booking  $\leq 3.5$  + Inflight entertainment  $\geq 4.5$  + leg

room service <= 2.5면 class = 0

따라서 leg room service 즉 다리 공간이 불만에 많은 영향을 끼침을 알 수 있다.

## 6. Neural Network grid search

### 6.1 grid search 결과 분석

---

하이퍼파라미터 후보 및 후보 값 설명

#### 1. Activation function

첫번째 하이퍼파라미터는 활성화함수이다. 활성화함수를 어떻게 설정하느냐에 따라 neural network의 비선형 결합 구조가 달라지게 된다. 특히 layer가 얼마나 복잡하느냐에 따라 활성화함수가 끼치는 영향이 달라진다. 최근에는 relu 함수를 가장 많이 사용하는 것으로 알고 있는데 실제로도 relu함수를 사용했을 때 best activation function이 되는지, 아니면 데이터셋마다 적합한 활성화함수가 있는 것인지 궁금해서 하이퍼파라미터 후보 기준으로 활성화함수를 설정했다.

후보는 익숙한 활성화함수 세 가지를 검증해보고 싶어서 logistic, tanh, relu를 선정했다.

#### 2. Hidden layer

두 번째 하이퍼파라미터는 hidden layer의 개수이다. Neural network의 꽃은 hidden layer이다. Hidden layer를 어떻게 설정하느냐에 따라 모델 명이 달라지고 성능이 개선되어 논문으로 발표된다. 예전에 개인적으로 toy project를 하면서 모델 성능을 위해 무작정 기존 모델에서 layer수를 늘려본 적이 있는데 계속 성능이 떨어졌던 경험이 있다. 따라서 실제로 가장 적절한 hidden layer의 개수를 trial and error로 직접 찾아보고 싶었다. Hidden layer의 개수는 어떻게 설정해야 할지 감이 오지 않아 실습시간에 1~30까지 바꿔 본 것을 유사하게 적용했다. 즉 3부터 30까지 약 3개씩 늘려가면서 총 8개의 case에 대해 실험을 진행했다.

#### 3. Max\_iteration

일반적으로 iteration을 많이 하면 할수록 좋은 성능을 얻게 된다고 알고 있다. 따라서 실제로도 여러 개의 iteration number에 대해 실험하면 가장 많은 iteration을 거쳤을 때 높은 성능을 보이



는지 확인해보고자 했다. Iteration 수는 100 300 500으로 설정했다. 더 크게 설정하는 것은 테스트에 큰 의미가 없을 것이라고 생각하여 유의미하게 적은 횟수인 100을 minimum으로, 500을 maximum으로 설정했다.

#### 4. 최적의 하이퍼 파라미터 조합

위와 같은 하이퍼파라미터에 대해 각각 바꿔가며 grid search를 수행한 결과, 총 72개 모델 중에서 가장 높은 auroc 성능을 보인 모델의 하이퍼파라미터 조합은 아래와 같다.

Best activation = logistic

Best hidden\_layer\_sizes = 12

Best max iter = 300

Auroc = 0.8229

결과적으로 내 예상을 모두 빗나가는 하이퍼파라미터 조합을 확인할 수 있었다. 아마 데이터셋에 종류에 따라 다를 것이고 multi layer perceptron model이기 때문에 deep learning에서 하이퍼파라미터 조건과 다를 수도 있다.

하지만 항상 relu가 좋은 성능을 보이는 것은 아니라는 점, hidden layer 가 높다고 좋은 것은 아니며 trial & error를 통해 적절한 개수를 찾아야 한다는 점, iteration이 많다고 무조건 좋은 성능을 보이는 것은 아니라는 점을 알게 되었다.

이러한 결과가 나온 이유는 데이터셋이 단순하고 분류 문제도 쉬웠기 때문이라고 생각한다. 아마 더 복잡한 문제였다면 layer를 더 많이 사용하면서 복잡한 모델이 만들어지고 iteration도 성능이 수렴하기 위해서 더 많이 필요할 것이며 모델이 복잡하기 때문에 relu 함수를 사용하는 것이 성능 개선에 더 유리하지 않았을까 예상해본다.

## 7. 세 모델의 성능 비교

### 7.1 세 모델의 평가지표 결과 및 해석

---

| Dataset | Model | TPR | TNR | Accuracy | BCR | F1-measure |
|---------|-------|-----|-----|----------|-----|------------|
|---------|-------|-----|-----|----------|-----|------------|

|              |                     |        |        |        |        |        |
|--------------|---------------------|--------|--------|--------|--------|--------|
| Dataset name | Decision Tree       | 0.8642 | 0.8508 | 0.8584 | 0.8575 | 0.8733 |
|              | Logistic Regression | 0.8799 | 0.8474 | 0.8657 | 0.8635 | 0.8810 |
|              | Neural Network      | 0.8433 | 0.7966 | 0.8230 | 0.8196 | 0.8433 |

Logistic Regression이 TNR을 제외하고 나머지 4개 지표에서 모두 가장 좋은 성능을 보여주었다. Neural Network는 3개 모델의 5개 지표에서 모두 가장 낮은 성능을 보여주었다. Decision Tree는 TNR 지표는 세 모델 중 가장 좋은 성능을 보여주었고 다른 4개 지표는 두 번째로 좋은 성능을 보여주었다.

이렇게 결과가 나온 이유를 해석해보자면 Decision Tree는 학습 시 하나의 Feature를 기준으로 이분으로만 분류가 가능하고 이러한 분류를 여러 번 반복하기 때문에 분류 성능을 높이기에는 한계가 있을 것으로 판단했다. 따라서 logistic regression이 더 높은 성능을 보여줄 수 있을 것이다. Neural Network는 이론적으로는 여러 개의 split layer를 비선형적으로 사용하기 때문에 최대 성능의 가능성이 다른 하나의 선형결합만 사용하는 logistic regression이나 차원에 수직으로만 split 선을 사용하는 Decision Tree보다 더 높다고 생각했다.

하지만 결과적으로 가장 낮은 성능을 보인 것은 오히려 문제 주어진 문제가 너무 단순해서 일수도 있고, 하이퍼파라미터가 매우 다양하기 때문에 더 많은 실험으로 하이퍼파라미터를 바꾸면서 최적의 모델을 찾아야 하는데 본 과제에서는 3개의 하이퍼파라미터만 실험했기 때문에 최적 모델을 찾아내지 못했을 수도 있다고 생각했다.

## 8. 새로운 데이터로 모델링

### 8.1 데이터 출처와 변수 설명

---

선정한 데이터는 diabetes prediction dataset이다.

다운받을 수 있는 링크는 아래와 같다.

<https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset>

#### <변수 설명>

Gender – 성별 – female / male

Age – 나이 – 1 ~ 80

Hypertension – 고혈압 여부 – 0 or 1

Heart disease – 심장병 여부 – 0 or 1

Smoking\_history – 흡연경험 – no info / never / other

Bmi – bmi 수치 – 10~95

HbA1c\_level – HbA1c 수치 – 3.5~9

Blood glucose level – 혈당수치 – 80~300

Diabetes -당뇨병 여부 – 0/1

전처리는 one hot encoding, outlier 제거, null 행 제거를 수행했다.

전처리 근거는 1번 문항에서 전처리를 수행한 근거와 동일하다.

## 8.2 선정이유 & 분배 비율 근거

---

주어진 문제에서 예측 정확도가 예측 결과물에 대한 해석보다 훨씬 더 중요할 것이라고 생각되는 분류 문제를 다루는 데이터셋을 선정하라고 요구했다. 따라서 우선 예측 결과물에 대한 해석이 필요없는 데이터셋 탐색했으나 일단 어떤 데이터셋이든 분류를 수행하면 feature를 통해 해석이 가능하기 때문에 해석 자체가 무의미한 데이터셋은 없었다. 따라서 상대적으로 예측 정확도의 중요성이 높은 문제를 다루는 데이터셋을 선정하는 것으로 방향을 설정했다.

따라서 예측을 잘못했을 때 위험성이 큰 데이터를 고려했고 diabetes prediction dataset은 잘못 예측했을 때 문제발생의 리스크가 크기 때문에 예측 정확도를 높이는 것이 중요할 것이라고 생각했다. 따라서 해당 데이터셋을 이번 분석에 사용할 데이터셋으로 결정했다.

#### <분배 비율 근거>

Seed를 설정한 후 train : valid : test = 60 : 20 : 20으로 분배해주었다. 그 이유는 데이터 양이 많지 않기 때문에 최대한 train data가 많으면 성능 향상에 도움이 되지만 validation data와 test data도 일정 양 이상 확보 되어야 하기 때문이다.

따라서 일정 비율을 유지하는 것으로 결정했고 분배한 비율인 6 : 2 : 2에 대한 근거는 전통적인 머신러닝 방법론에서 주로 train : test : val = 6 : 2 : 2로 나누기 때문에 동일한 비율을 사용하기로 결정했다.

## 9. pre-pruning & Neural Network

### 9.1 pre-pruning & NN 결과 비교

---

#### 1. pre pruning decision tree 모델 찾기

<사용된 하이퍼파라미터>

##### 1. Criterion

Pruning을 판단할 criterion은 총 세 가지에 대하여 탐색 범위를 설정했다. 그 이유는 DecisionTreeClassifier의 criterion은 "gini", "entropy", "log\_loss"만 설정 가능했기 때문이다. Pruning 기준을 어떻게 설정하느냐에 따라 split했을 때의 impurity가 달라지기 때문에 어떤 criterion을 설정하느냐에 따라 pruning의 결과가 유의미하게 달라진다. 특히 entropy는 세 기준 중 가장 넓은 범위를 표현할 수 있기 때문에 더 설명력이 뛰어나다고 볼 수 있다. 따라서 best model은 어떤 criterion을 사용할지가 궁금하여 pruning에 사용했다.

##### 2. Min\_split

The minimum number of samples required to split an internal node:

Internal node를 split할 때 요구되는 샘플의 최소 수를 의미한다. 즉 min\_split을 작게 설정하면 split 수가 많아지고 크게 설정하면 설정된 수치 이상으로 sample이 나뉘면 더 이상 split 하지 않는다. 따라서 어떤 값을 설정하느냐에 따라 tree의 depth에 큰 영향을 미친다. Default값은 2이지만 데이터 크기가 2000을 넘기 때문에 좀 더 크게 설정해도 될 것 같다는 생각이 들었다. 따라서 10, 30, 50, 70, 90, 100으로 넉넉하게 설정했고 sample을 잘게 나누지 않았을 때 복잡도가 낮아져 더 높은 성능을 보여주는지 확인하고자 pruning hyperparameter로 사용했다.

##### 3. Max\_depth

노드의 깊이가 깊어질수록 복잡도가 증가하기 때문에 최대 depth를 제한하여 pruning을 할 수

있다. 깊이의 범위는 앞서 full tree의 depth가 23이었기 때문에 최소 1에서 최대 23까지 탐색 범위를 설정했다. 총 5가지 경우에 대하여 [1,5,9,14,19,23]으로 탐색 범위를 지정했다.

Pre-pruning은 학습데이터와 검증 데이터를 이용하여 진행했고 auroc 값을 기준으로 최적의 하이퍼파라미터 조합은 아래와 같다.

| 하이퍼파라미터                | 값    |
|------------------------|------|
| Best criterion         | Gini |
| Best min_samples_split | 100  |
| Best max_depth         | 9    |

Entropy가 아닌 gini가 best criterion으로 선정된 것이 인상깊었다. Min\_samples split또한 잘게 쪼갤수록 더 높은 성능을 나타낸다고 생각했는데 실험했던 6가지 case 중 가장 큰 100이 선정된 것을 통해 만약 더 큰 수치의 min samples split을 설정한다면 더 좋은 성능의 모델을 얻을 수 있을 것이라고 예측할 수 있었다.

마지막으로 max\_depth는 9임을 알 수 있었고 Full tree의 depth가 23이었음을 감안할 때 복잡도 차이가 매우 큼을 알 수 있다. 따라서 위와 같은 pre pruning 과정이 결과적으로 매우 큰 차이를 만들 수 있음을 알 수 있다.

## 2. Best Neural Network 모델 찾기

### 5. Activation function

첫번째 하이퍼파라미터는 활성화함수이다. 활성화함수를 어떻게 설정하느냐에 따라 neural network의 비선형 결합 구조가 달라지게 된다. 특히 layer가 얼마나 복잡하느냐에 따라 활성화함수가 끼치는 영향이 달라진다. 최근에는 relu 함수를 가장 많이 사용하는 것으로 알고 있는데 실제로도 relu함수를 사용했을 때 best activation function이 되는지, 아니면 데이터셋마다 적합한 활성화함수가 있는 것인지 궁금해서 하이퍼파라미터 후보 기준으로 활성화함수를 설정했다.

후보는 익숙한 활성화함수 세 가지를 검증해보고 싶어서 logistic, tanh, relu를 선정했다.

### 6. Hidden layer

두 번째 하이퍼파라미터는 hidden layer의 개수이다. Neural network의. 꽃은 hidden layer이다. Hidden layer를 어떻게 설정하느냐에 따라 모델 명이 달라지고 성능이 개선되어 논문으로 발표된

다. 예전에 개인적으로 toy project를 하면서 모델 성능을 위해 무작정 기존 모델에서 layer수를 늘려본 적이 있는데 계속 성능이 떨어졌던 경험에 있다. 따라서 실제로 가장 적절한 hidden layer의 개수를 trial and error로 직접 찾아보고 싶었다. Hidden layer의 개수는 어떻게 설정해야 할지 감이 오지 않아 실습시간에 1~30까지 바꿔 본 것을 유사하게 적용했다. 즉 3부터 30까지 약 3개씩 늘려가면서 총 8개의 case에 대해 실험을 진행했다.

## 7. Max\_iteration

일반적으로 iteration을 많이 하면 할수록 좋은 성능을 얻게 된다고 알고 있다. 따라서 실제로도 여러 개의 iteration number에 대해 실험하면 가장 많은 iteration을 거쳤을 때 높은 성능을 보이는지 확인해보고자 했다. Iteration 수는 100 300 500으로 설정했다. 더 크게 설정하는 것은 테스트에 큰 의미가 없을 것이라고 생각하여 유의미하게 적은 횟수인 100을 minimum으로, 500을 maximum으로 설정했다.

## 4. 최적의 Neural network의 하이퍼파라미터

Best activation : tanh

Best hidden\_layer\_sizes : 12

Best max\_iter : 500

# 10. 새로운 데이터셋에 대한 세 모델 결과 비교

## 10.1 평가지표 결과 및 해석

---

| Dataset      | Model         | TPR    | TNR    | Accuracy | BCR    | F1-measure |
|--------------|---------------|--------|--------|----------|--------|------------|
| Dataset name | Decision Tree | 0.8642 | 0.7966 | 0.8348   | 0.8297 | 0.8552     |

|  |                     |        |        |        |        |        |
|--|---------------------|--------|--------|--------|--------|--------|
|  | Logistic Regression | 0.8642 | 0.7966 | 0.834  | 0.8297 | 0.8553 |
|  | Neural Network      | 0.4393 | 0.9978 | 0.9610 | 0.6621 | 0.5979 |

성능평가 결과, 지표별로 큰 차이를 보였다. 따라서 지표별로 해석을 수행했다.

Tpr은 true positive rate = recall을 의미하고  $tp/(tp+fn)$ 으로 계산한다. 즉 예측과 실제 모두 p인 경우를 의미한다. 실제로 양의 값을 가질 때 양으로 예측해낼 확률을 의미하며 해당 지표에서는 decision tree와 logistic regression이 동일하게 높은 성능을 보여주었다. 하지만 Neural network는 절반에 가까운 낮은 성능을 보여주면서 양인 경우를 잘 맞추지 못하는 모습을 보여주었다.

Neural network의 성능지표가 특이하여 직접 confusion matrix를 그려본 결과는 아래와 같다.

|      |    |
|------|----|
| 1865 | 3  |
| 74   | 58 |

즉 데이터셋 자체가 positive 값이 많은 상황인데 양을 얼마나 잘 예측해내는지가 성능에 큰 영향을 끼친 것을 알 수 있었다. 그래서 다른 모델에 비해 TPR 차이가 커지게 됨을 확인할 수 있다. 반대로 TNR은 NN가 다른 두 모델보다 훨씬 큰 모습을 알 수 있으며 마찬가지로 실제 negative한 값이 몇개 안되기 때문에 예측 차이가 작더라도 격차가 매우 크게 발생하는 모습을 확인할 수 있다.

다음은 accuracy를 비교했다. Accuracy는 NN가 다른 모델들에 비해 상대적으로 높게 나온 것을 알 수 있다. 하지만 accuracy는 class간의 데이터 비율 차이가 큰 경우 데이터 개수가 많은 class로 labeling 해버리면 오차가 발생할 수 있기 때문에 단순히 accuracy가 높다고 좋은 모델이라고 판단할 수는 없다.

BCR은 balanced classification rate를 의미한다. BCR 값은 Decision Tree와 Logistic regression이 동일하게 0.8297로 NN에 비해 높은 수치를 보여주었다.

F1-measure는 precision과 recall을 통합하여 계산한 평가지표이다. 계산식은 아래와 같다.

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

따라서 precision과 recall의 단점들을 서로 보완해주기 때문에 매우 중요한 지표로 사용할 수 있다.

결과적으로 decision tree와 logistic regression이 각각 0.8552와 0.8553으로 높은 수치를, NN이 0.5979로 낮은 값을 보였다.

전체적으로 고려했을 때 실제 confusion matrix상에서도 decision tree와 logistic regression은 몇 개의 sample만 다르고 거의 유사한 수치값을 보여주었다. 그리고 일부 평가지표는 class 비율이 불균형할 때 편향된 결과를 나타낸다는 점을 고려하면 NN보다는 decision tree와 logistic regression이 더 좋은 성능을 보여준다고 해석했다.