

다변량데이터분석

Decision Tree

Neural Network

과제 3

산업경영공학부

2019170815

신건우

# Table of contents

[Q1] Dataset Selection and Reason for Selection

[Q2] Comparison of Full Decision Tree (DT) and Post-pruning DT

[Q3] Pre-pruning DT Hyperparameter Search

[Q4] Comparison of Post-pruning and Pre-pruning DT Models

[Q5] Selection of Optimal DT and Analysis of Decision Rules

[Q6] Neural Network (NN) Hyperparameter Search

[Q7] Comparison of Logistic Regression, DT, and NN

[Q8] Dataset Selection and Reason for Selection

[Q9] Pre-pruning DT and NN Hyperparameter Search

[Q10] Comparison of Logistic Regression, DT, and NN

[Appendix]

1. churn for bank customer data feature visualization

2. plot of trees

3. vehicle claim fraud data feature visualization

[Q1] 본인이 생각하기에 “예측정확도”도 중요하지만 “예측결과물에 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 link : <https://www.kaggle.com/adammaus/predicting-churn-for-bank-customers>

## 선정 이유

해당 데이터는 은행 고객의 이탈과 관련된 데이터이다. 고객의 이탈 여부에 대해 단순히 예측하는 것 보다는 어떤 고객들이 이탈을 하는지 파악하는 것도 서비스 개선에 있어 중요하다. 어떤 요소를 지닌 고객이 이탈율이 높은지 파악한다면, 그를 바탕으로 마케팅 전략을 수립하거나 금융 상품을 개발 할 수 있을 것이다. 요약하자면 다음과 같다.

### 1. 고객 유지 관리:

신용카드 고객의 이탈을 예측하는 것은 금융기관에게 매우 중요한 문제다. 고객이 이탈하면 수익이 감소하고, 새로운 고객을 유치하는 비용이 증가하기 때문이다.

### 2. 해석의 중요성:

단순히 고객이 이탈할지 여부를 예측하는 것만으로는 충분하지 않다. 고객이 왜 이탈하는지, 어떤 요인들이 이탈에 영향을 미치는지를 이해하는 것이 중요하다. 이를 통해 금융기관은 이탈을 방지하기 위한 적절한 대응책을 마련할 수 있다.

### 3. 비즈니스 인사이트 제공:

예측 결과를 해석함으로써 고객 행동, 제품 및 서비스의 약점, 마케팅 전략의 효과 등을 분석할 수 있다. 이는 고객 관계 관리(CRM) 및 제품 개선에 큰 도움이 된다.

## 데이터셋 설명

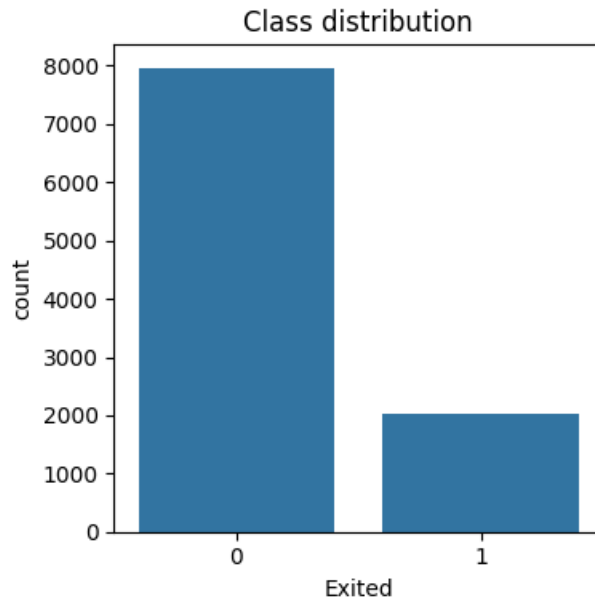
원본 데이터에 존재하는 'RowNumber', 'CustomerId', 'Surname' column은 예측에 유의미하지 않은 feature로 판단하여 제거했다. 각 feature에 대한 간략한 설명과 feature 별 개선에 활용할 수 있는 요소는 다음과 같다.

- CreditScore (신용 점수): 고객의 신용 점수는 금융 기관이 고객의 신용 위험을 평가하는 데 중요한 요소다. 예측 결과가 신용 점수와 어떻게 연관되어 있는지 이해하는 것은 고객의 신뢰도와 관련된 중요한 정보를 제공한다.
- Geography (지리): 고객의 위치 정보는 지역별로 다른 고객 행동 패턴을 이해하는 데 도움이 된다. 예측 결과가 특정 지역에서 높은 이탈률을 보인다면, 해당 지역의 경제적 또는 사회적 요인들을 고려해야 한다.
- Gender (성별): 성별에 따른 이탈률 차이는 마케팅 전략과 고객 관리 방안을 달리 설정하는 데 중요하다. 예를 들어, 여성 고객이 남성 고객보다 높은 이탈률을 보이는 경우, 여성 고객을 대상으로 한 맞춤형 서비스를 제공할 수 있다.
- Age (나이): 나이에 따른 이탈 패턴을 이해하는 것은 매우 중요하다. 예를 들어, 특정 연령대의 고객이 높은 이탈률을 보이면, 해당 연령대를 대상으로 한 특별한 유지 전략이 필요하다.
- Tenure (관계 유지 기간): 고객이 은행과의 관계를 유지한 기간은 이탈 가능성을 예측하는 데 중요한 지표다. 장기 고객이 이탈하는 경우, 이는 은행의 충성도 프로그램이나 서비스에 문제가 있을 수 있음을 시사한다.
- Balance (잔액): 고객의 계좌 잔액은 이탈 여부와 밀접한 관련이 있을 수 있다. 높은 잔액을 가진 고객이 이탈하는 경우, 이는 큰 손실로 이어질 수 있으며, 이러한 고객을 유지하는 방안을 마련해야 한다.
- NumOfProducts (상품 수): 고객이 은행에서 이용하는 금융 상품의 수는 이탈 여부와 중요한 관계가 있다. 많

은 상품을 이용하는 고객이 이탈할 경우, 이는 은행의 서비스 품질에 문제가 있을 수 있다.

- HasCrCard (신용카드 보유 여부): 신용카드 보유 여부는 고객의 은행과의 관계를 나타낸다. 신용카드를 보유한 고객이 이탈할 경우, 신용카드와 관련된 서비스나 혜택에 문제가 있을 수 있다.
- IsActiveMember (활성 회원 여부): 활성 회원 여부는 고객의 은행 서비스 사용 빈도를 나타낸다. 비활성 회원의 이탈률이 높다면, 이를 활성화시키기 위한 전략이 필요하다.
- EstimatedSalary (예상 연봉): 고객의 예상 연봉은 이탈 가능성에 영향을 줄 수 있다. 고소득 고객의 이탈을 방지하기 위해 맞춤형 금융 상품이나 서비스를 제공할 필요가 있다.
- Exited: 고객이 은행을 이탈했는지 여부. 이탈 여부는 고객이 은행과의 관계를 종료했는지를 나타내며, 1은 이탈함을, 0은 이탈하지 않음을 의미한다.

추가 전처리 필요 여부를 파악하기 위해 데이터에 존재하는 각 feature에 대한 시각화를 진행했다. 결과는 appendix에 첨부했다. 시각화 결과 명목형 변수에 대해 one-hot-encoding이 필요하다고 판단했고, 이를 진행해주었다.<sup>1</sup> 그 결과로 데이터셋에는 11개의 input feature가 존재한다. 또한 class imbalance가 존재함을 확인할 수 있다.



(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.

### Split 비율 선정 이유

Split 비율 선정에 정해진 정답은 없다. 현재 프로젝트에서는 크게 모델의 목적, 데이터 셋의 크기와 모델의 복잡도를 고려하여 data split 비율을 선정했다.

기본적으로 가장 많이 이용되는 분배 비율 중 하나인 6:2:2을 선택했다.

먼저 기본적으로 학습에 가장 많은 데이터를 사용해야 하기에 검증 및 테스트 데이터에 비해 학습 데이터에 더 많은 데이터를 할당하였다.

<sup>1</sup> one-hot-encoding을 하지 않으면 모델 학습이 이루어지지 않는 문제가 발생

해당 데이터셋은 객체의 수가 10000개로 중간 규모의 데이터셋이다. 보편적으로 사용되는 다른 분배 비율인 8:1:1 또한 고려할 수 있으나, 테스트 데이터셋이 더 많은 6:2:2을 사용해 모델의 일반화 성능에 더 집중하고자 했다. 또한, 구성하고자 하는 모델의 복잡도에 10000개의 60%를 사용해도 충분하다고 생각했다.

따라서 학습에 가장 많은 데이터를 활용하면서, 검증 및 테스트에 동등한 비율이 할당되고, 일반화 성능에 더 주목하는 6:2:2을 분배 비율로 선택했다.

사용한 지표들에 대한 간략한 설명은 다음과 같다.

- **TPR (True Positive Rate):** 실제 양성 중에서 모델이 양성으로 올바르게 예측한 비율.
- **TNR (True Negative Rate):** 실제 음성 중에서 모델이 음성으로 올바르게 예측한 비율.
- **Precision:** 모델이 양성으로 예측한 사례 중 실제 양성인 비율.
- **Accuracy:** 전체 예측 중에서 올바르게 예측한 비율.
- **BCR (Balanced Classification Rate):** TPR과 TNR의 평균.
- **F1-Measure:** Precision과 Recall의 조화 평균.
- **AUROC (Area Under the Receiver Operating Characteristic Curve):** threshold 값에 무관하게 모델의 분류 성능을 종합적으로 평가하는 지표로, 1에 가까울수록 성능이 좋음을 의미.

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

1) 학습 데이터만을 이용해서 학습한 Full Tree

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

현재 데이터 분석의 목적은 양성 클래스(이탈)를 정확히 분류하는 것에 있다. 따라서 recall(TPR) 지표가 보다 중요하다고 볼 수 있다. 현재 class imbalance가 존재하므로 두 가지 방식으로 model 학습을 진행했다.

## 1. 기본 진행

단순하게 학습을 진행했다.

## 2. 양성 클래스 가중치 부여

Class imbalance 데이터셋에서는 모델이 다수 클래스에 편향되어 소수 클래스에 대한 예측 성능이 떨어질 가능성이 높다. 이를 고려하여 모델이 학습할 때 소수 클래스의 오류에 더 큰 패널티를 부여하여 소수 클래스에 대한 예측 성능을 향상시킬 수 있다.

Sklearn의 `compute_class_weight` 함수를 활용하여, 클래스에 대한 가중치를 계산했다. `Class_weight` 매개변수를 `balanced`로 설정하여 각 클래스의 가중치를 데이터셋에서의 클래스 빈도에 반비례하도록 설정했다.

Full tree, post pruning tree에 대해 1번과 2번을 진행한 결과는 다음과 같다.

| Model(Tree)                          | TPR           | Precision     | TNR           | Accuracy | BCR           | F1            | AUROC         |
|--------------------------------------|---------------|---------------|---------------|----------|---------------|---------------|---------------|
| Full-Tree                            | 0.5209        | 0.4589        | 0.8431        | 0.7775   | 0.6820        | 0.4879        | 0.6820        |
| Post-pruning Tree                    | 0.4693        | <b>0.7235</b> | <b>0.9542</b> | 0.8555   | 0.7117        | <b>0.5693</b> | 0.8350        |
| Full-Tree with Class Weights         | 0.4693        | 0.4961        | 0.8782        | 0.7950   | 0.6738        | 0.4823        | 0.6738        |
| Post-pruning Tree with Class Weights | <b>0.7715</b> | 0.4392        | 0.7483        | 0.7530   | <b>0.7599</b> | 0.5597        | <b>0.8394</b> |

Post-pruning을 수행한 모델이 Full Tree보다 전반적으로 더 나은 성능을 보인다.

특히, Post-pruning Tree with Class Weights는 긍정 클래스 예측에서 매우 높은 성능을 보이며, AUROC도 가장 높아 모델 성능이 매우 우수함을 알 수 있다.



Figure 1 full tree

추가적으로, 적합한 full tree는 시각화하기 어려울 정도의 복잡도를 가지고 있다. 모델의 설명력이 훼손되는 것과 더불어 과적합의 위험 또한 높으므로, 적절한 가지치기가 필요해 보인다.

[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.

고려한 하이퍼파라미터와 범위, 선정 이유는 다음과 같다.

1. **criterion:**

- **설명:** 노드를 분할하는 데 사용되는 기준. 'gini'는 Gini 불순도를, 'entropy'는 정보 이득을 사용합니다.
- **탐색 범위:** ['gini', 'entropy']
- **설정 이유:** 두 가지 모두 많이 사용되는 기준이므로, 어느 것이 더 나은지 비교하기 위해 포함했다.

2. **min\_samples\_split:**

- **설명:** 노드를 분할하기 위한 최소 샘플 수. 이 값이 클수록 트리가 덜 복잡해진다.
- **탐색 범위:** [2, 5, 10]
- **설정 이유:** 기본값인 2 외에도 일반적으로 사용되는 몇 가지 값을 포함하여 트리 복잡도를 조절한다.

3. **max\_depth:**

- **설명:** 트리의 최대 깊이. 이 값이 클수록 트리가 복잡해지고 과적합될 수 있다.
- **탐색 범위:** [5, 10, 15, None]

- **설정 이유:** 트리의 깊이를 제한함으로써 과적합을 방지하고자 여러 깊이를 탐색한다. 'None'은 제한 없이 최대 깊이로 성장하는 경우다.

#### 4. min\_impurity\_decrease:

- **설명:** 분할이 이루어지기 위한 최소 불순도 감소. 이 값이 클수록 덜 중요한 분할이 제거된다.
- **탐색 범위:** [0.0, 0.01, 0.1]
- **설정 이유:** 트리의 성장을 제어하고 중요하지 않은 분할을 제거하기 위한 값들을 포함한다.

#### 5. max\_leaf\_nodes:

- **설명:** 리프 노드의 최대 수. 이 값이 작을수록 트리가 간단해진다.
- **탐색 범위:** [10, 20, 30, None]
- **설정 이유:** 트리의 복잡도를 제어하기 위해 다양한 값을 탐색한다. 'None'은 리프 노드 수에 제한이 없음을 의미한다.

#### 6. min\_samples\_leaf:

- **설명:** 리프 노드가 되기 위한 최소 샘플 수. 이 값이 클수록 리프 노드가 더 커지고 트리가 덜 복잡해진다.
- **탐색 범위:** [1, 2, 5]
- **설정 이유:** 노드에 최소 샘플 수를 지정하여 트리의 복잡도를 조절하고 일반화 성능을 높인다.

총 864개의 하이퍼파라미터 조합에서 grid search로 나타난 최적의 하이퍼파라미터 조합은 다음과 같다.

- **criterion:** 'entropy'
- **min\_samples\_split:** 2
- **max\_depth:** 10.0
- **min\_impurity\_decrease:** 0.0
- **max\_leaf\_nodes:** 20.0
- **min\_samples\_leaf:** 1

[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?

가지치기를 진행한 두 모델에 대해서는 [Q3]에서 좋은 성능을 보인 class weight를 사용한 두 모델을 이용해 답변하겠다.

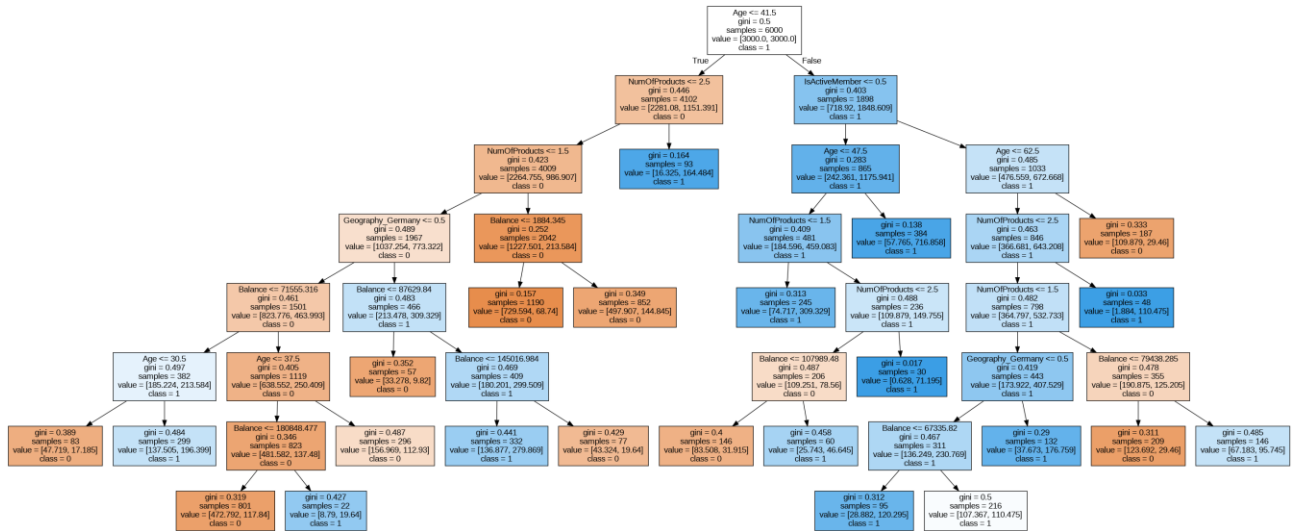


Figure 2 post-pruning tree

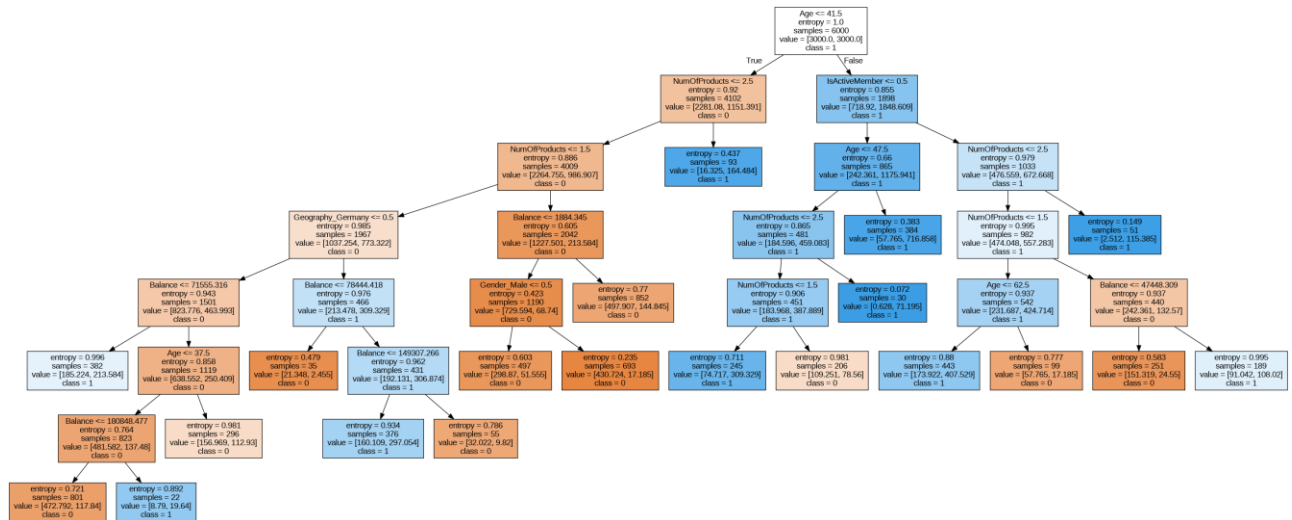


Figure 3 pre-pruning tree

각 tree plot을 확대하여 appendix에 첨부했다.

먼저 Full tree(no class weight)는 'CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Geography\_Germany', 'Geography\_Spain', 'Gender\_Male' 의 11가지 변수 모두를 분기에 사용했으며, 853개의 leaf node를 가진다.



Full tree에서 각 변수가 split에 사용된 횟수와 features\_importances\_로 출력한 특성 중요도는 다음과 같다.

| Variable Name     | Usage Count | Feature Importance |
|-------------------|-------------|--------------------|
| Age               | 116         | 0.2311             |
| NumOfProducts     | 11          | 0.1150             |
| Geography_Germany | 9           | 0.0247             |
| Balance           | 148         | 0.1641             |
| EstimatedSalary   | 185         | 0.1429             |
| Tenure            | 109         | 0.0729             |
| Geography_Spain   | 23          | 0.0166             |
| CreditScore       | 179         | 0.1349             |
| Gender_Male       | 33          | 0.0243             |
| IsActiveMember    | 16          | 0.0568             |
| HasCrCard         | 23          | 0.0168             |

### Leaf node의 수

post pruning tree의 leaf node의 수는 27개, pre pruning tree의 leaf node의 수는 20개로, full tree에 비해 매우 복잡도가 감소하였다. 따라서 full tree에 비해 설명력이 증가(복잡도가 감소)했다고 이야기할 수 있다.

tree depth에 있어서는 두 방식에 따른 큰 차이가 보이지 않는다.

### 분기에 사용된 변수

Post pruning tree는 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember', 'Geography\_Germany'의 5개 변수를 split에 사용했다.

| Variable Name     | Usage Count | Feature Importance |
|-------------------|-------------|--------------------|
| Age               | 5           | 0.4572             |
| NumOfProducts     | 6           | 0.3150             |
| Balance           | 8           | 0.1207             |
| IsActiveMember    | 1           | 0.0643             |
| Geography_Germany | 2           | 0.0428             |
| Tenure            | -           | 0.0000             |
| HasCrCard         | -           | 0.0000             |
| Geography_Spain   | -           | 0.0000             |
| Gender_Male       | -           | 0.0000             |
| EstimatedSalary   | -           | 0.0000             |
| CreditScore       | -           | 0.0000             |

Pre pruning tree는 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember', 'Geography\_Germany', 'Gender\_Male'의 6개 변수를 split에 사용했다.

| Variable Name     | Usage Count | Feature Importance |
|-------------------|-------------|--------------------|
| Age               | 4           | 0.4242             |
| NumOfProducts     | 6           | 0.3516             |
| IsActiveMember    | 1           | 0.0723             |
| Geography_Germany | 1           | 0.0311             |
| Balance           | 6           | 0.1093             |
| Gender_Male       | 1           | 0.0115             |
| Tenure            | -           | 0.0000             |
| HasCrCard         | -           | 0.0000             |
| Geography_Spain   | -           | 0.0000             |
| EstimatedSalary   | -           | 0.0000             |
| CreditScore       | -           | 0.0000             |

두 가지치기 방식 모두에서 'Age'가 분류에 있어서 가장 중요하게 사용된 변수로 드러났다.

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

다음은 test 데이터에 대한 각 모델 별 분류 성능 지표다. 해당 지표와 모델의 복잡도를 바탕으로 최적 결정나무모델을 선정하고자 한다.

| Model(Tree)                          | TPR    | Precision | TNR    | Accuracy | BCR    | F1     | AUROC  |
|--------------------------------------|--------|-----------|--------|----------|--------|--------|--------|
| Full-Tree                            | 0.5209 | 0.4589    | 0.8431 | 0.7775   | 0.6820 | 0.4879 | 0.6820 |
| Post-pruning Tree                    | 0.4693 | 0.7235    | 0.9542 | 0.8555   | 0.7117 | 0.5693 | 0.8350 |
| Pre-pruning Tree                     | 0.7887 | 0.4303    | 0.7332 | 0.7445   | 0.761  | 0.5568 | 0.8315 |
| Full-Tree with Class Weights         | 0.4693 | 0.4961    | 0.8782 | 0.7950   | 0.6738 | 0.4823 | 0.6738 |
| Post-pruning Tree with Class Weights | 0.7715 | 0.4392    | 0.7483 | 0.7530   | 0.7599 | 0.5597 | 0.8394 |
| Pre-pruning Tree with Class Weights  | 0.8059 | 0.4321    | 0.7294 | 0.7450   | 0.7677 | 0.5626 | 0.8419 |

대체로 full-tree를 제외하고 class weight를 부여한 모델이 그렇지 않은 모델보다 전반적인 성능(양성 클래스 분류 성능)이 좋다. 더 자세하게는, pre-pruning을 수행한 모델들의 성능이 대체로 좋다. 복잡도(설명력)의 관점에서 post pruning tree와 pre pruning tree의 복잡도 차이가 크지 않다. 따라서 지표와 모델 복잡도를 바탕으로 Pre pruning Tree with class weights를 최적 결정나무로 선정했다.

## 최적 결정나무 plot

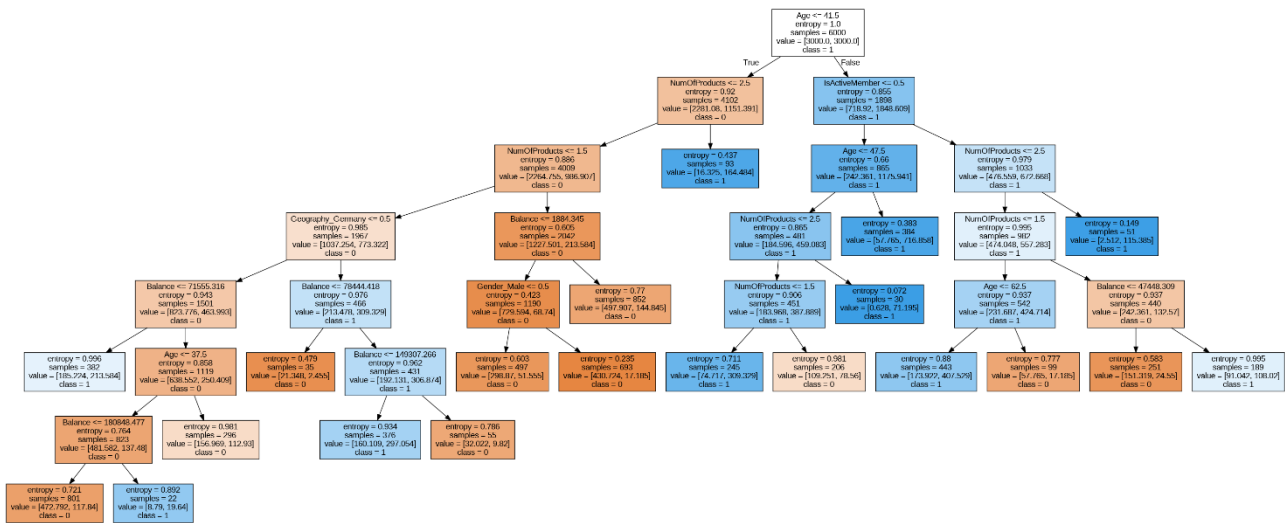


Figure 3과 동일하다.

## 대표 규칙 3가지

재귀를 통한 Tree 순회를 이용하여 해당 DT모델이 train 데이터를 통해 학습한 분류 규칙을 일련의 문장으로 만들 수 있다.

```
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts > 1.50 AND Balance > 1884.34 => Class: 0 (Samples: 852)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts > 1.50 AND Geography_Germany <= 0.50 AND Balance > 71555.32 AND Age <= 37.50 AND Balance <= 180848.48 => Class: 0 (Samples: 801)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts > 1.50 AND Balance <= 1884.34 AND Gender_Male > 0.50 => Class: 0 (Samples: 693)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts > 1.50 AND Balance <= 1884.34 AND Gender_Male <= 0.50 => Class: 0 (Samples: 497)
Rule: Age > 41.50 AND IsActiveMember > 0.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Age <= 62.50 => Class: 1 (Samples: 443)
Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND Age > 47.50 => Class: 1 (Samples: 394)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Geography_Germany <= 0.50 AND Balance <= 71555.32 => Class: 1 (Samples: 382)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Geography_Germany <= 0.50 AND Balance <= 78444.42 AND Balance > 149307.27 => Class: 1 (Samples: 376)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Geography_Germany <= 0.50 AND Balance > 71555.32 AND Age > 37.50 => Class: 0 (Samples: 296)
Rule: Age > 41.50 AND IsActiveMember > 0.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Balance <= 47448.31 => Class: 0 (Samples: 251)
Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND Age <= 47.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 => Class: 1 (Samples: 245)
Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND Age <= 47.50 AND NumOfProducts <= 2.50 AND NumOfProducts > 1.50 => Class: 0 (Samples: 206)
Rule: Age > 41.50 AND IsActiveMember > 0.50 AND NumOfProducts <= 2.50 AND NumOfProducts > 1.50 AND Balance > 47448.31 => Class: 1 (Samples: 189)
Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Age > 62.50 => Class: 0 (Samples: 99)
Rule: Age <= 41.50 AND NumOfProducts > 2.50 => Class: 1 (Samples: 93)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND Geography_Germany <= 1.50 AND Balance > 78444.42 AND Balance > 149307.27 => Class: 0 (Samples: 55)
Rule: Age > 41.50 AND IsActiveMember > 0.50 AND NumOfProducts > 2.50 => Class: 1 (Samples: 51)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Geography_Germany > 0.50 AND Balance <= 78444.42 => Class: 0 (Samples: 35)
Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND Age <= 47.50 AND NumOfProducts > 2.50 => Class: 1 (Samples: 30)
Rule: Age <= 41.50 AND NumOfProducts <= 2.50 AND NumOfProducts <= 1.50 AND Geography_Germany <= 0.50 AND Balance > 71555.32 AND Age <= 37.50 AND Balance > 180848.48 => Class: 1 (Samples: 22)
```

Figure 4 decision rule of pre pruning DT

중요 규칙 3가지로 양성 클래스로 분류한 규칙이면서 불순도 지표가 낮은 규칙 3가지를 골랐다.

**Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND Age <= 47.50 AND NumOfProducts > 2.50 => Class: 1 (Samples: 30, Impurity: 0.07)**

나이가 41.5세 초과이며, 활성 회원 여부가 0.5 이하(즉, 활성 회원이 아님), 나이가 47.5세 이하이고, 보유한 상품의 수가 2.5개 초과인 경우.

**인사이트 :** 활성 회원이 아님에도 불구하고 여러 상품을 보유한 이유를 탐구해 볼 필요가 있다. 이들 고객이 특정 상품에 대한 만족도가 높거나, 특정 서비스에 대한 필요를 느끼고 있을 가능성이 있다.

Rule: Age > 41.50 AND IsActiveMember > 0.50 AND NumOfProducts > 2.50 => Class: 1 (Samples: 51, Impurity: 0.15)

나이가 41.5세 초과이며, 활성 회원 여부가 0.5 초과(즉, 활성 회원임), 보유한 상품의 수가 2.5개 초과인 경우.

**인사이트** : 이미 회사의 서비스에 만족하고 있으며, 다양한 상품을 사용하고 있습니다. 이들의 충성도를 유지하고 더 높은 만족도를 제공하는 것이 중요하다.

Rule: Age > 41.50 AND IsActiveMember <= 0.50 AND Age > 47.50 => Class: 1 (Samples: 384, Impurity: 0.38)

나이가 41.5세 초과이며, 활성 회원 여부가 0.5 이하(즉, 활성 회원이 아님), 나이가 47.5세 초과인 경우.

**인사이트** : 상대적으로 나이가 많고, 활성 회원이 아니기 때문에 회사의 서비스에 대한 인식이 낮거나 필요성을 느끼지 못할 수 있다. 이들의 필요와 관심사를 이해하고, 이를 반영한 서비스 제공이 필요하다.

해당 중요 규칙의 원인을 심층, 정성적으로 분석한다면 서비스 개선에 있어서 더 좋은 인사이트를 얻을 수 있을 것으로 생각된다.

[Q6] (Neural Network) 동일한 데이터셋에 대하여 Neural Network 학습을 위해 필요한 최소 3가지 이상의 하이퍼파라미터를 선정하고, 각 하이퍼파라미터마다 최소 3개 이상의 후보 값(최소 27가지 조합)을 사용하여 grid search를 수행한 뒤, 검증데이터에 대한 AUROC 기준으로 하이퍼파라미터 조합을 찾아 보시오.

DT와 다르게, NN에서는 Gradient descent 알고리즘으로 파라미터를 업데이트하는 과정에서 input data의 값의 크기가 영향을 준다.

따라서 각 feature 별로 존재하는 단위에 의한 효과를 정규화 과정을 통해 제거해 줌으로써 단위에 의해 왜곡되지 않은 각 feature의 패턴을 학습하게 했다. 정규화에는 이상치에 보다 robust한 standard scaler를 사용했다.

## 모델링 세부 사항

sklearn의 MLPClassifier보다 모델링과 학습과정의 커스터마이징이 용이한 pytorch를 사용하여 NN을 모델링했다.

## 학습 과정 구현 세부 사항

학습을 진행하기 이전에 실험 재현성을 어느정도 보장하기 위해 시드 고정을 진행했다. 하지만 torch 자체의 non deterministic한 알고리즘에 완벽한 재현이 보장되지 않는다는 문제점이 관찰되었다.

Google Colab 환경에서 Gpu(T4)를 사용하여 모델 학습을 진행했다. 또한, Torch의 dataloader에서 batch\_size 옵션을 256로 사용하여 모델 가중치 업데이트에 Mini-Batch Gradient Descent 방식을 사용했다. 해당 방식의 이점은 다음과 같다.

### 1. 메모리 효율성

- 한 번에 전체 데이터셋을 메모리에 올리지 않고 작은 배치 단위로 처리하기 때문에 메모리 사용량을 줄일 수 있다.

## 2. 학습 속도 향상:

- 각 배치에 대해 병렬 처리가 가능해져서 학습 속도를 높일 수 있다. 또한, 미니 배치 단위로 가중치를 업데이트하기 때문에 전체 데이터셋을 사용할 때보다 더 빠르게 수렴할 수 있다.


## 3. 일반화 성능 향상:


- 모든 배치에 대해 가중치를 업데이트하기 때문에 모델이 더 다양한 데이터 패턴을 학습하게 되어 과적합(overfitting)을 방지하는 데 도움이 된다.


## 4. 노이즈 완화:


- 미니 배치를 사용하면 경사 하강법의 업데이트가 각 배치 샘플에 의해 영향을 받으므로, 완전한 데이터셋을 사용하는 것보다 약간의 노이즈가 추가된다. 이는 모델이 더 나은 일반화 성능을 갖도록 하는 효과가 있다.

손실 함수는 이진 분류 문제에 주로 사용하는 BCEWithLogitsLoss를 사용했다. 해당 손실 함수는 NN의 출력을 input으로 받아 내부적으로 확률을 계산하여 손실 함수를 계산한다. 추가적으로 과적합을 방지하기 위해 validation 과정에서 early stopping을 적용했다. 3번의 epoch 동안 validation data에 대해 BCEWithLogitsLoss가 개선되지 않으면, 학습을 멈추고 모델을 반환하도록 구현했다.

```
Early stopping triggered
Training with params: {'hidden_size': 16, 'layer_num': 3, 'activation_func': 'Tanh'}
35%  35/100 [02:11<06:22, 5.89s/it]

Early stopping triggered
Training with params: {'hidden_size': 16, 'layer_num': 3, 'activation_func': 'Sigmoid'}
41%  41/100 [02:54<06:43, 6.85s/it]

Early stopping triggered
Training with params: {'hidden_size': 16, 'layer_num': 5, 'activation_func': 'ReLU'}
16%  16/100 [00:39<04:16, 3.05s/it]

Early stopping triggered
Training with params: {'hidden_size': 16, 'layer_num': 5, 'activation_func': 'Tanh'}
10%  10/100 [00:20<03:23, 2.26s/it]


Early stopping triggered
Training with params: {'hidden_size': 16, 'layer_num': 5, 'activation_func': 'Sigmoid'}
16%  16/100 [00:38<04:11, 2.99s/it]
```

Figure 5 early stop in training

## 하이퍼파라미터 선정

고려한 하이퍼파라미터는 다음과 같다. 기본적인 NN 구조에서 모델 성능에 큰 영향을 주는 것으로 판단되는 하이퍼파라미터를 선정했다.

- **Activation function** : ELU, ReLU, Tanh, Sigmoid

활성화 함수는 input의 선형 결합을 비선형 변환하여 모델의 복잡도를 증가시킴으로써 보다 복잡한 패턴을 인식하고, 학습할 수 있도록 만든다.

각 활성화 함수는 개별적인 특징을 가지고 있으며, 이를 고려하여 3개의 활성화 함수를 선택했다.

- ELU

- ◆ 비선형 함수로, 양수 입력에 대해서는 입력 값을 그대로 출력하고, 음수 입력에 대해서는 지수 함수를 적용하여 출력한다.
- ◆ 음수 입력에 대해 부드러운 포화를 제공하여, 네트워크가 음수 입력에 대해서도 효과적으로 학습할 수 있게 한다.
- ◆ ReLU의 '죽은 뉴런' 문제를 완화할 수 있으며, 학습 초기에 빠른 수렴을 도울 수 있다.
- ◆ 계산 비용이 ReLU보다 약간 높을 수 있다.

- ReLU

- ◆ 비선형 함수로, 음수를 0으로 변환하고 양수는 그대로 출력한다.
- ◆ 계산이 간단하여 학습이 빠르고, 대규모 심층 신경망에서 효율적이다.
- ◆ gradient vanishing 문제를 어느 정도 해결해준다.
- ◆ 하지만, 음수 입력에 대해 기울기가 0이 되어 '죽은 뉴런' 문제를 발생시킬 수 있다.

- Tanh

- ◆ 출력 범위가 -1과 1 사이로, 데이터가 중심(0)에 가까워져 학습이 안정되고 효율적일 수 있다.
- ◆ 비선형 함수이며, Sigmoid 함수보다는 gradient vanishing 문제에 덜 민감하다.
- ◆ 그러나 여전히 깊은 네트워크에서는 gradient vanishing 문제가 발생할 수 있다.
- ◆

- Sigmoid

- ◆ 출력 범위가 0과 1 사이로, 특히 이진 분류 문제에서 출력층의 활성화 함수로 적합하다.
- ◆ 비선형 함수이며, 모든 입력에 대해 출력값이 항상 양수다.
- ◆ gradient vanishing 문제가 발생하기 쉬워 깊은 네트워크에서는 학습이 어려울 수 있다.
- ◆ 또한, 출력값이 항상 양수이기 때문에 출력의 평균이 0이 아니어서 가중치 업데이트가 비효율적일 수 있다.

- **Number of hidden nodes** : 16, 32, 64, 128

은닉층의 노드 수가 많을수록 모델의 표현력이 증가하고, 연산 복잡도가 증가하며, 과적합 가능성이 높아질 수 있다.

- **Number of hidden layers** : 1, 3, 5 (num\_layers : 3,5,7)

은닉층 수가 많을수록, 모델이 더 복잡한 패턴을 학습할 수 있고, 학습 시간이 길어질 수 있으며, 과적합과 기울기 소실/폭발 문제가 일어날 가능성이 높아진다.

언급하지 않은 기타 하이퍼파라미터는 다음과 같이 일반적인 수치로 설정했다.

- **Learning rate** : 0.005
- **Optimizer** : Adam
- **Epoch** : 100

총 27가지 조합에 대해 grid search를 진행한 결과 auroc 기준으로 선택된 하이퍼파라미터는 다음과 같다.

|                     |                    |
|---------------------|--------------------|
| Hidden node size    | 16                 |
| Layer num           | 3 (hidden layer 1) |
| Activation function | Tanh               |
| AUROC               | 0.8750             |

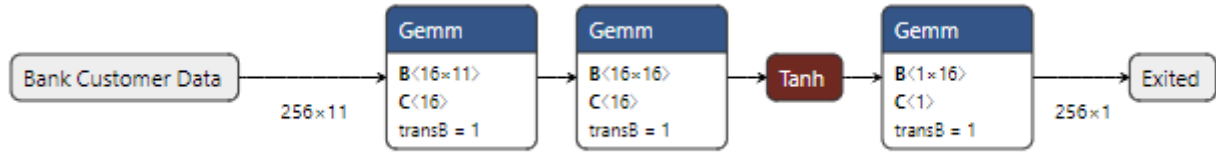


Figure 6 best NN model structure

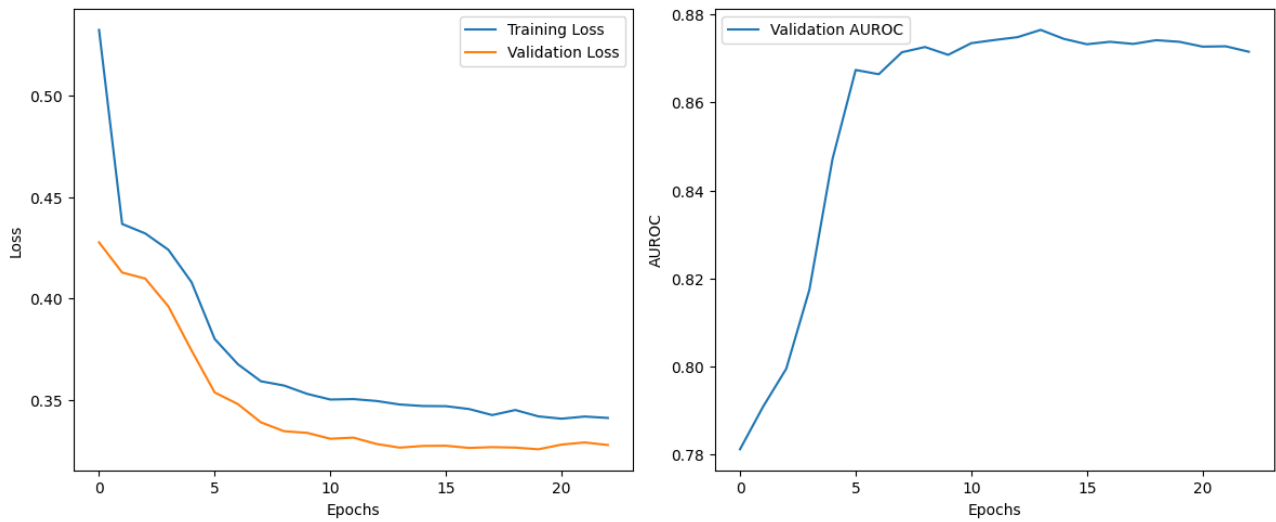


Figure 7 best NN model learning graph

학습과정을 시각화하여 확인한 결과, 학습 곡선이 대체로 완만하며, early stop이 적절히 적용되어 과적합이 의심되는 패턴이 등장하지 않았다.

추가적으로 DT모델에서 진행한 것처럼 class weight을 통해 positive class에 가중치를 두어 학습하도록 한 뒤에 성능을 비교했다.

|                     |                    |
|---------------------|--------------------|
| Hidden node size    | 128                |
| Layer num           | 3 (hidden layer 1) |
| Activation function | Sigmoid            |
| AUROC               | 0.8787             |

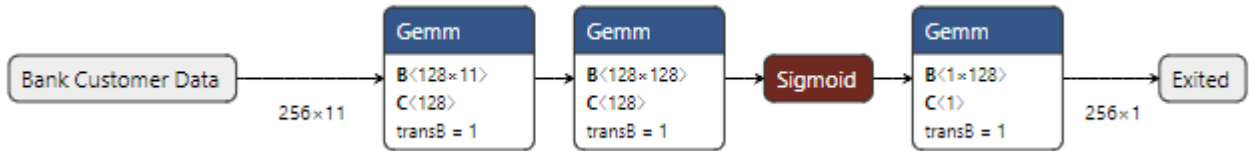


Figure 8 best NN model with class weight structure

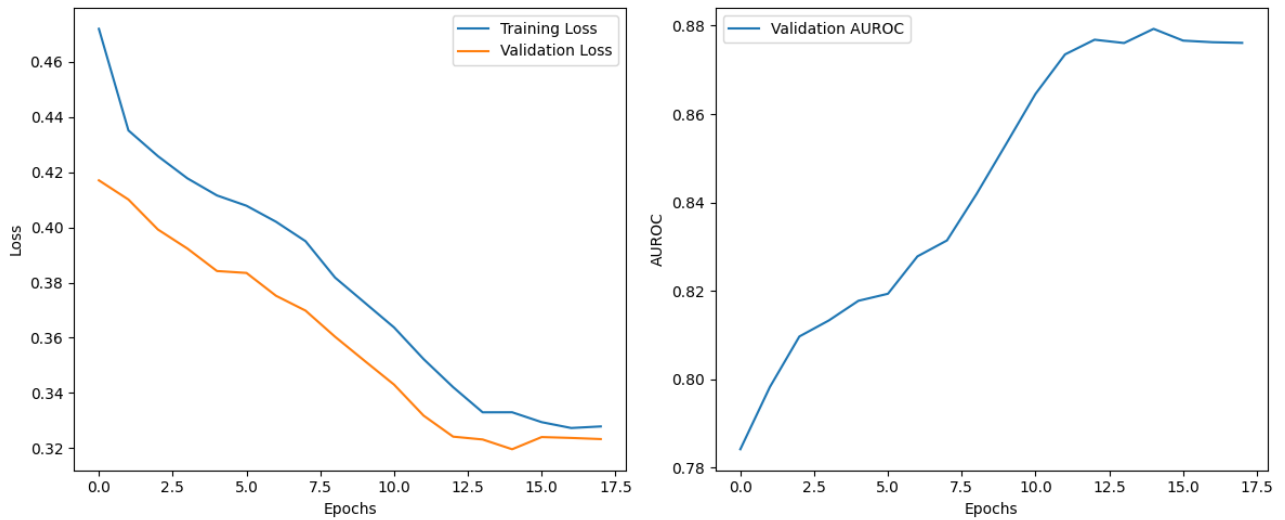


Figure 9 best NN model with class weight learning curve

Test data에 대한 각 결과는 다음과 같다.

| Model                | TPR           | Precision | TNR    | Accuracy | BCR    | F1     | AUROC         |
|----------------------|---------------|-----------|--------|----------|--------|--------|---------------|
| NN                   | 0.4644        | 0.7354    | 0.9573 | 0.8570   | 0.7108 | 0.5693 | 0.8578        |
| NN with class weight | <b>0.5061</b> | 0.7203    | 0.9498 | 0.8595   | 0.7280 | 0.5945 | <b>0.8667</b> |

recall(TPR)을 통해 판단했을 때, 양성 클래스 분류에서 보다 성능이 좋은 NN with class weight 모델을 사용하기로 했다. 또한 각 모델에 대한 분류 지표를 계산할 때, 최종 값을 sigmoid에 통과시켜 확률을 얻고 threshold를 기본인 0.5로 사용해 이진분류를 진행했으므로, threshold에 독립적인 지표인 AUROC의 관점에서 NN with class weight 모델이 더 낫다고 할 수 있다. 이는 class weight 도입 취지에 부합하는 결과이기도 하다.



[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 해석해 보시오.

로지스틱 회귀 모형 학습에는 NN 모델에 사용한 것과 같은 수렴 속도 향상을 위해 standard scaler로 정규화된 데이터를 사용했다. class\_weight는 balanced로 설정했다.

| Dataset                  | Model                      | TPR           | TNR           | Accuracy | BCR           | F1-Measure    |
|--------------------------|----------------------------|---------------|---------------|----------|---------------|---------------|
| Churn for bank customers | Logistic Regression        | 0.6978        | 0.6999        | 0.6995   | 0.6989        | 0.4859        |
|                          | Decision Tree(pre-pruning) | <b>0.8059</b> | 0.7294        | 0.7450   | <b>0.7677</b> | 0.5626        |
|                          | Neural Network             | 0.5061        | <b>0.9498</b> | 0.8595   | 0.7280        | <b>0.5945</b> |

Logistic Regression 모델은 TPR과 TNR이 비슷하며, 전반적으로 균형 잡힌 성능을 보여주고 있다.

Pre-pruning을 적용한 DT 모델은 Logistic Regression 모델 보다 높은 TPR과 TNR을 보여준다. 이는 특히 양성 클래스를 잘 예측하는 능력이 좋음을 의미한다. 또한, F1 점수 역시 더 높아, 양성 클래스에 대한 모델의 TPR과 TNR이 모두 향상되었음을 나타낸다.

NN 모델은 매우 높은 TNR을 보여주며, 이는 음성 클래스를 매우 잘 예측한다는 것을 의미한다. 반면에 TPR은 상대적으로 낮아 양성 클래스를 놓치는 경우가 많음을 나타낸다. 그러나 전체적인 정확도는 가장 높으며, F1 점수도 가장 높아, 음성 클래스에 대한 예측이 강점인 것으로 보인다.

[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.

Dataset link : <https://www.kaggle.com/datasets/shivamb/vehicle-claim-fraud-detection>

## 데이터셋 설명

해당 데이터 셋은 차량보험 사기 탐지 데이터다. 차량보험 사기는 사고 후 재산상 손해나 상해를 수반하는 허위·과장 청구를 공모하는 것이다. 사기범이 고의로 사고를 ‘알선’하는 단계적 사고, 사고 현장에 있지도 않은 사람이 중상을 입었다고 주장하는 팬텀 탑승자 이용, 손해가 심할 정도로 과장된 허위 대인 청구 등이 대표적이다. 이 데이터 셋은 정책 세부 정보와 함께 차량 데이터 세트 - 속성, 모델, 사고 세부 정보 - 정책 유형, 수명 - 등을 포함한다.

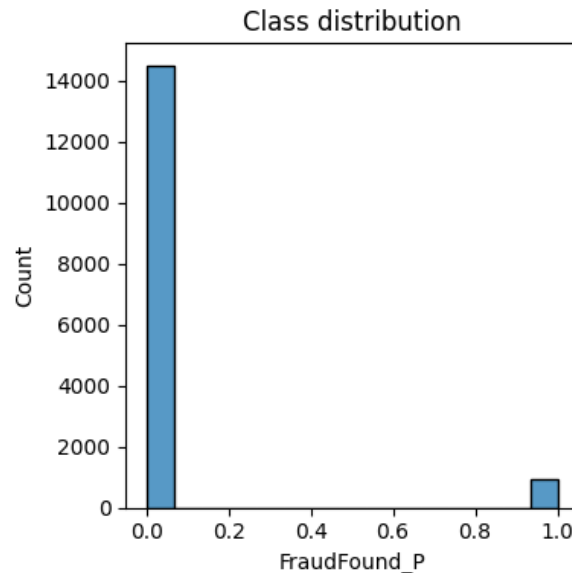
목적은 보험 사기 여부를 탐지(분류)하는 것이다.

각 feature에 대한 설명은 다음과 같다.

- **Month**: 사고 발생 월.
- **WeekOfMonth**: 사고 발생 주의 순서 (해당 월의 몇 번째 주).
- **DayOfWeek**: 사고 발생 요일.
- **Make**: 차량 제조사.
- **AccidentArea**: 사고 발생 지역 (Urban 또는 Rural).
- **DayOfWeekClaimed**: 클레임 제출 요일.
- **MonthClaimed**: 클레임 제출 월.
- **WeekOfMonthClaimed**: 클레임 제출 주의 순서.
- **Sex**: 성별.
- **MaritalStatus**: 결혼 상태.
- **Age**: 운전자의 나이.
- **Fault**: 사고 책임 여부 (Policy Holder 또는 Third Party).
- **PolicyType**: 보험 정책 유형.
- **VehicleCategory**: 차량 종류.
- **VehiclePrice**: 차량 가격 범위.
- **FraudFound\_P**: 사기 여부 (0: 정상, 1: 사기).
- **PolicyNumber**: 보험 정책 번호.
- **RepNumber**: 담당자 번호.
- **Deductible**: 공제액.
- **DriverRating**: 운전자 평가 점수.
- **Days\_Policy\_Accident**: 보험 가입 후 사고 발생일까지의 일수.
- **Days\_Policy\_Claim**: 보험 가입 후 클레임 제출일까지의 일수.
- **PastNumberOfClaims**: 과거 클레임 횟수.
- **AgeOfVehicle**: 차량 연식.
- **AgeOfPolicyHolder**: 보험 가입자의 나이.
- **PoliceReportFiled**: 경찰 보고서 제출 여부.

- **WitnessPresent**: 목격자 존재 여부.
- **AgentType**: 대리인 유형 (Internal 또는 External).
- **NumberOfSuppliments**: 보충 서류의 수.
- **AddressChange\_Claim**: 클레임 제출 시점으로부터 최근 주소 변경 시점 (범주형).
- **NumberOfCars**: 소유 차량 대수.
- **Year**: 사고 발생 연도.
- **BasePolicy**: 기본 보험 유형.

명목형 변수들에 대해 one-hot-encoding을 수행하여 총 124개의 input feature가 만들어졌다. 각 feature에 대한 시각화는 appendix에 첨부했다. 이전 데이터와 마찬가지로 class imbalance가 두드러지게 나타난다. Class imbalance 문제는 학습 데이터셋 구성과 손실 함수에서 고려해줄 것이다.



**Figure 10 Class distribution**

## 선정 이유

보험과 관련된 경제 이론으로 역선택(adverse selection) 현상이 있다. 역선택은 보험 시장에서 발생하는 정보 비대칭 문제 중 하나로, 주로 보험 가입자와 보험 회사 간의 정보 불균형으로 인해 발생한다. 이 문제는 보험 회사가 보험 가입자의 위험 수준을 정확히 평가하지 못하는 상황에서 발생하며, 주로 다음과 같은 방식으로 나타난다

### 1. 정보 비대칭:

- 보험 사기는 보험 가입자가 자신의 위험 수준이나 사고 발생 여부에 대해 보험사보다 더 많은 정보를 가지고 있는 상황에서 발생한다.

## 2. 보험 사기의 발생:

- 일부 보험 가입자는 고의적으로 사고를 일으키거나 사고를 과장하여 보험금을 청구한다.
- 이러한 사기 행위는 보험사의 손실을 증가시키고, 전체 보험 청구 비용을 늘린다.

## 3. 보험료 인상:

- 보험사는 사기 행위로 인한 손실을 메우기 위해 보험료를 인상할 수밖에 없다. 이는 전체 보험 시장의 평균 보험료를 상승시키는 결과를 낳는다.

## 4. 저위험 가입자의 이탈:

- 보험료가 인상되면, 실제로 사고 발생 가능성이 낮은 저위험 가입자들은 보험료를 지불하는 것이 비합리적이라고 판단하고 보험을 해지하거나 가입하지 않게 된다.
- 저위험 가입자의 이탈로 인해 고위험 가입자의 비율이 상대적으로 증가한다.

## 5. 악순환:

- 저위험 가입자가 이탈함에 따라, 보험사의 손실 위험은 더욱 커지고, 이는 다시 보험료 인상으로 이어진다.
- 이러한 악순환은 보험사의 재정 건전성을 위협하고, 전체 보험 시장의 안정성을 해칠 수 있다.

역선택 문제를 방지하는 방법 중의 하나로 선별(screening)이 있다. 선별이란, 정보를 갖지 못한 측에서 불충분하지만 주어진 자료를 이용하여 상대방의 특성을 파악하려고 하는 것을 의미한다. 선별 과정에서 고객 정보, 보험 사기 데이터를 활용할 수 있을 것이다. 특히 실시간 탐지가 중요한데, 사후 처리는 비용 효율적인 측면에서 비효율적이기 때문이다. 즉, 보험 사기 탐지에는 예측 결과에 대한 사후적인 해석보다, 사기 여부를 실시간으로 정확히 탐지해내야 적절한 조치를 취할 수 있고, 이를 통해 역선택 문제를 일정 부분 해결할 수 있다. 따라서 해당 데이터셋이 주제에 적합하다고 생각하여 선정하게 되었다.

[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.

### Pre-pruning DT model

이전 문제와 동일한 하이퍼파라미터에 대해 grid search를 수행하여 최적의 pre-pruning tree를 찾고, test 데이터에 대한 성능을 확인했다. 이전에 학습 과정에서 class weight를 적용해준 것이 성능이 좋았으므로, 마찬가지로 class weight를 적용하여 진행했다

- criterion: 'gini'
- min\_samples\_split: 2
- max\_depth: 15.0
- min\_impurity\_decrease: 0.0

- max\_leaf\_nodes: 30.0
- min\_samples\_leaf: 5

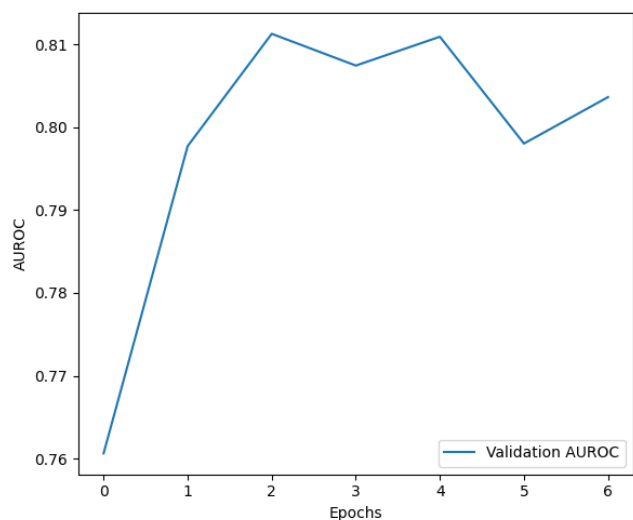
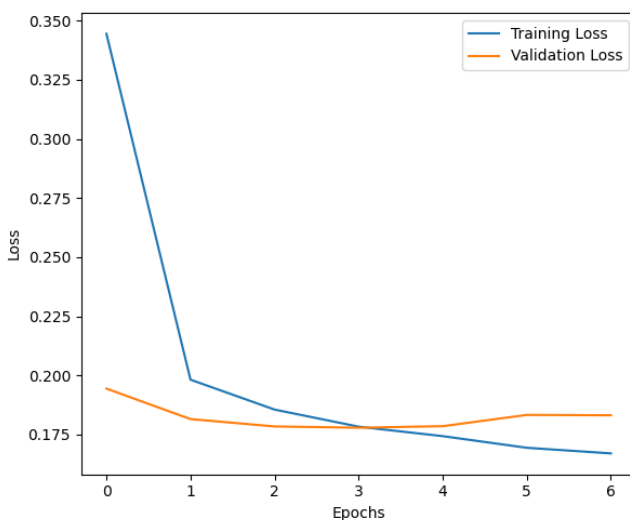
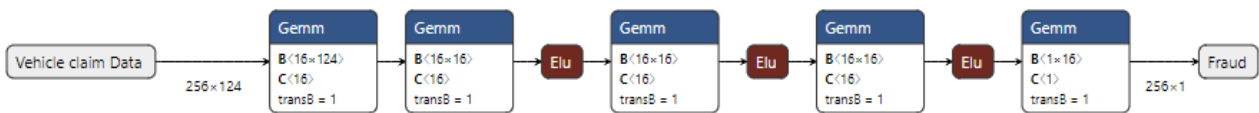
| Model                              | TPR    | Precision | TNR    | Accuracy | BCR    | F1     | AUROC  |
|------------------------------------|--------|-----------|--------|----------|--------|--------|--------|
| Pre-pruning tree with Class Weight | 0.3261 | 0.3871    | 0.9672 | 0.9290   | 0.6467 | 0.3540 | 0.8584 |

## NN model

확인한 하이퍼파라미터 조합은 이전 문제와 같은 48가지로 동일하게 grid search를 수행하고, validation 데이터에 대한 auroc로 최적 하이퍼파라미터를 선정했다.

위에서 언급했듯 보험 사기 데이터 또한 class imbalance가 존재한다. 이전 문제에서 수행한 것과 마찬가지로 class weight를 계산하여 양성 클래스 분류에 대해 가중치를 두어 학습하도록 했다. 하지만 테스트 데이터에 대한 결과를 확인한 결과 양성 클래스에 대해 거의 학습하지 못했다고 판단했다.

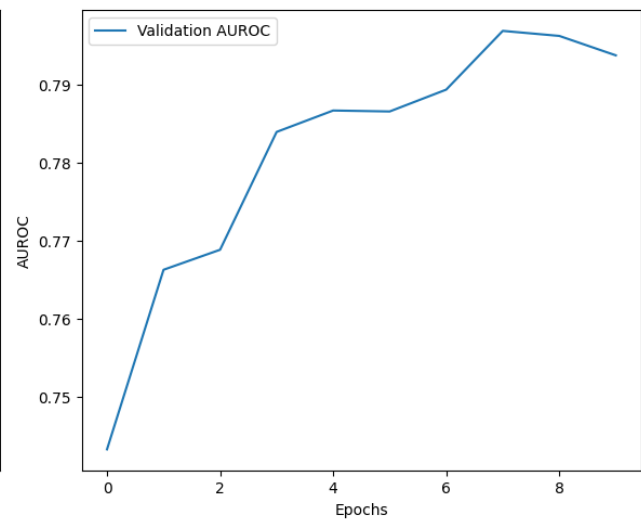
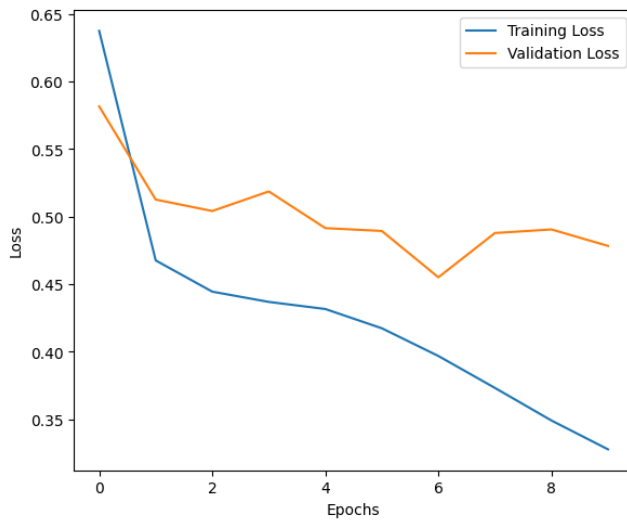
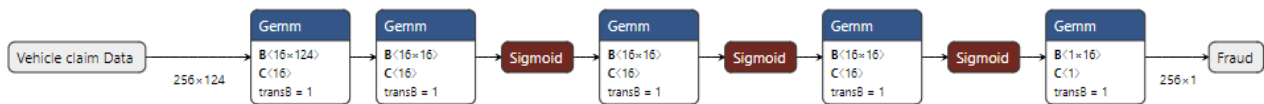
|                     |                    |
|---------------------|--------------------|
| Hidden node size    | 16                 |
| Layer num           | 5 (hidden layer 3) |
| Activation function | ELU                |
| AUROC               | 0.7944             |



따라서 대안으로 oversampling 기법 중 하나인 SMOTE(Synthetic Minority Over-sampling Technique)을 활용해보기로 했다. SMOTE은 불균형한 클래스 분포를 가진 데이터셋에서 소수 클래스의 샘플을 합성하여 증가시키는 방법이다.

이 기법은 소수 클래스의 샘플을 임의로 선택하고, 가장 가까운 이웃 중 하나를 선택하여 이 두 샘플 사이에 새로운 샘플을 생성함으로써 작동한다. 이렇게 함으로써, 모델이 소수 클래스를 더 잘 학습할 수 있도록 돕는다.

|                     |                    |
|---------------------|--------------------|
| Hidden node size    | 16                 |
| Layer num           | 5 (hidden layer 3) |
| Activation function | Sigmoid            |
| AUROC               | 0.7819             |



두 모델의 test 데이터 셋에 대한 분류 성능 지표는 다음과 같다.<sup>2</sup>

| Model                | TPR           | Precision | TNR    | Accuracy | BCR           | F1            | AUROC  |
|----------------------|---------------|-----------|--------|----------|---------------|---------------|--------|
| NN with class weight | 0.0489        | 0.3600    | 0.9945 | 0.9381   | 0.5217        | 0.0861        | 0.7944 |
| NN with SMOTE        | <b>0.6196</b> | 0.1518    | 0.7803 | 0.7708   | <b>0.7000</b> | <b>0.2439</b> | 0.7819 |

SMOTE를 사용한 경우 눈에 띄게 recall(TPR)이 높아진 것을 확인할 수 있다. BCR, F1을 통해 알 수 있듯, class balance를 고려한 정확도가 높고, 데이터 활용의 목적이 사기 탐지에 있으므로 recall이 높은 SMOTE를 적용한 NN 모델을 사용하겠다.

<sup>2</sup> DT with SMOTE 또한 시도했으나 class weight 방법론보다 낮은 성능을 보였다.

[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 (일치여부가 평가 점수에 영향을 미치지 않음).

| Dataset             | Model               | TPR           | TNR           | Accuracy | BCR           | F1-Measure    |
|---------------------|---------------------|---------------|---------------|----------|---------------|---------------|
| Vehicle claim fraud | Logistic Regression | <b>0.8043</b> | 0.6590        | 0.6676   | <b>0.7317</b> | 0.2241        |
|                     | Decision Tree       | 0.3261        | <b>0.9672</b> | 0.9290   | 0.6467        | <b>0.3540</b> |
|                     | Neural Network      | 0.6196        | 0.7803        | 0.7708   | 0.7000        | 0.2439        |

Logistic Regression 모델은 상대적으로 높은 TPR을 보이며, 이는 모델이 실제 긍정 클래스를 잘 예측한다는 것을 의미한다. 그러나 TNR이 상대적으로 낮아, 부정 클래스를 예측하는 데에는 다소 어려움을 겪고 있는 것으로 보인다. BCR이 0.7317인 점에서 클래스 불균형에 대해 어느 정도 균형 잡힌 성능을 보여주고 있다. 하지만 F1-Measure가 낮은 것은 Precision이 낮음을 의미한다.

Decision Tree 모델은 매우 높은 TNR을 보이며, 부정 클래스를 잘 예측하는 데 뛰어난 성능을 보인다. 그러나 TPR이 매우 낮아, 긍정 클래스를 잘 예측하지 못하는 것으로 보인다. BCR이 0.6467인 점에서 클래스 불균형에 대해 덜 균형 잡힌 성능을 보여준다. F1-Measure가 Logistic Regression보다 조금 더 높지만 여전히 낮은 편이다.

Neural Network 모델은 TPR과 TNR 모두 중간 정도의 성능을 보이며, 두 클래스 모두를 비교적 균형 있게 예측하고 있다. BCR이 0.7000인 점에서 클래스 불균형에 대해 균형 잡힌 성능을 보여준다. 그러나 F1-Measure는 여전히 낮은 편이다.

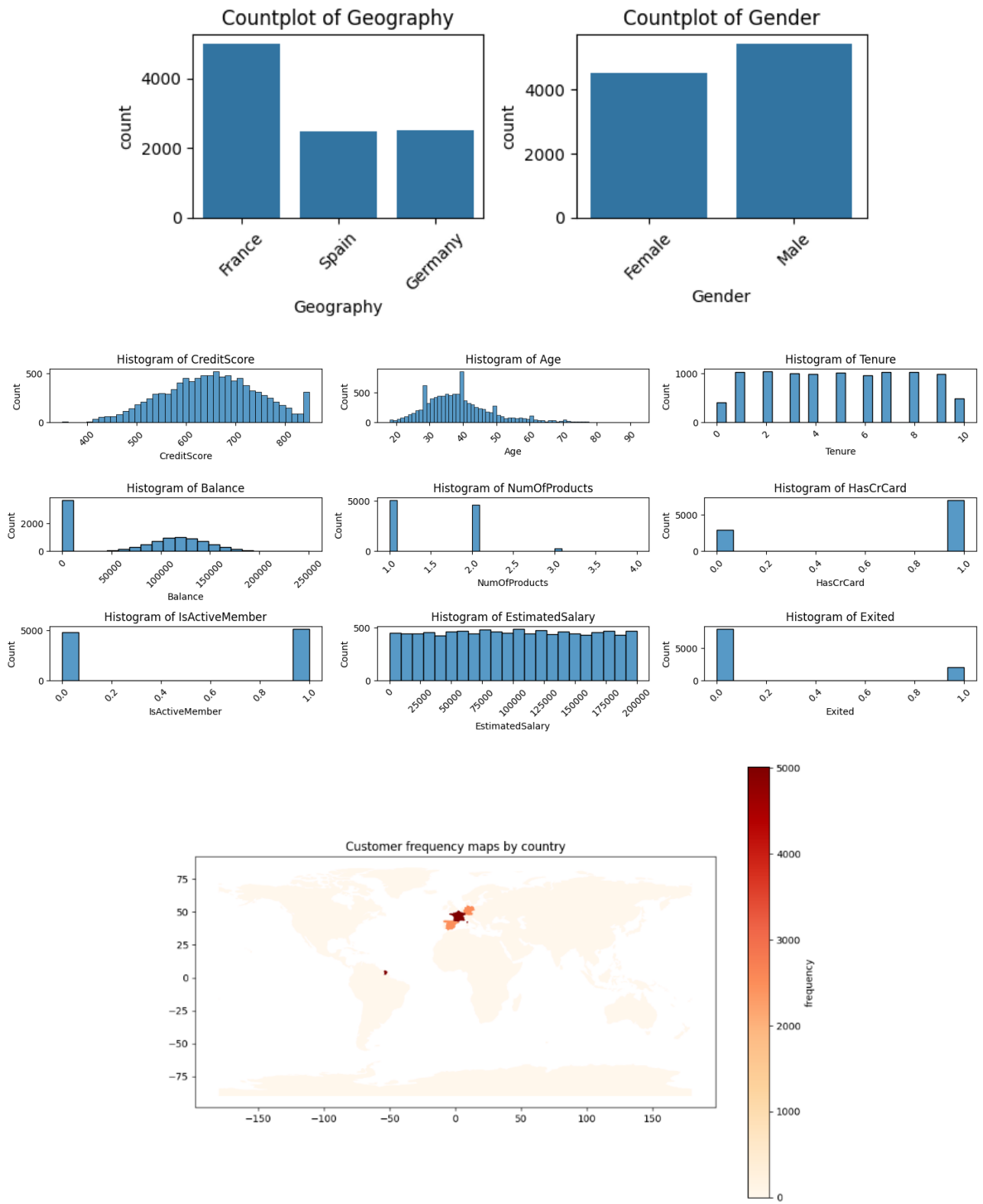
만약에 모델을 사용해야한다면, Logistic Regression을 사용할 것 같다. 간단한 모델이나, TPR 분류 성능이 좋다는 점에서 사기 탐지에 있어 조기 경보의 역할을 담당할 수 있을 것으로 보인다. 만약 오분류에 따른 비용이 크다면, 보다 균형 잡힌 NN 모델을 사용할 것이다.

모델의 종류에 따른 성능 우위가 [Q7]에서의 결과와 일치하지 않을 것이라고 생각했는데, 그 이유는 머신 러닝의 'No Free Lunch Theorem'에 근거한다.

어떤 알고리즘도 모든 문제에서 최적의 성능을 보장하지 않는다. 각 알고리즘은 특정 유형의 데이터나 문제에 대해 더 잘 동작할 수 있으며, 데이터의 특성에 따라 성능이 달라질 수 있다. 따라서, 특정 데이터셋에 대해 최적의 성능을 보이는 모델이 다른 데이터셋에서도 동일한 성능을 보장하지 않는다.

Appendix

1. churn for bank customer data feature visualization





## 2. plot of trees

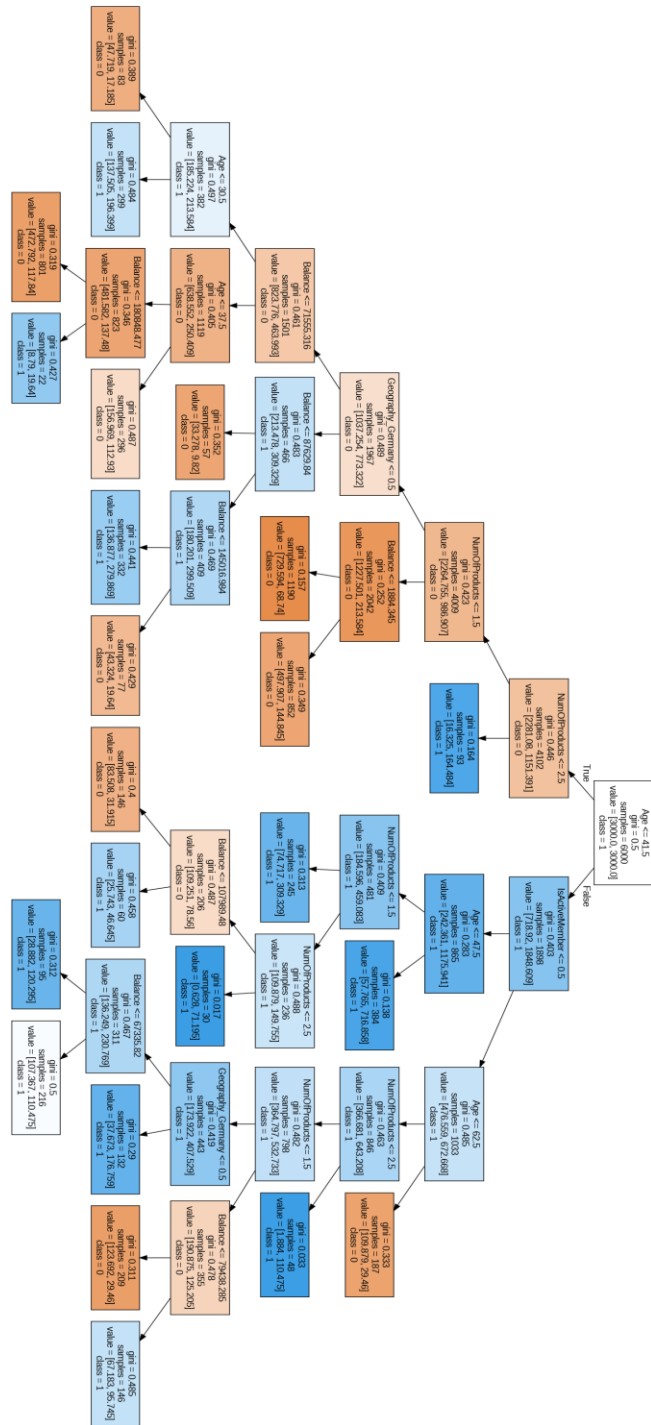


Figure 11 post pruning tree



### 3. vehicle claim fraud data feature visualization

