

표 형식 데이터에 대한 딥러닝을
활용한 암 발병 예측



저자1 송민재

저자2 서진욱

저자3 최지광

지도교수 송길태

목 차

1. 서론	1
1.1. 연구 배경	1
1.2. 주요 문제점 분석	2
1.3. 연구 목표	3
1.3.1 표 형식 데이터에 대한 딥러닝 모델 성능 향상	
1.3.2 표 형식 딥러닝에 대한 해석	
1.3.3 암 예측 모델 개발 및 서비스	
2. 연구 배경	4
2.1. SAINT	4
2.1.1. 개요	
2.1.2. 구조	
2.1.3. SAINT 코드	
2.1.4. 학습방법	
2.2. 기술 스택	8
2.2.1 Django	
2.2.2 NGINX	
2.2.3 Unicorn	
2.2.4 Docker	
2.2.5 AWS	
3. 연구 내용	10
3.1. 개발 일정	10
3.1.1. 상세 개발 일정	
3.1.2. 구성원별 역할	
3.2. 딥러닝 모델 개발	11
3.2.1. 모델 비교	
3.2.2. SHAP 모델	
3.2.3. SHAP 코드(SAINT 기반)	

3.3.	딥러닝 모델을 사용한 웹서비스 설계	29
3.3.1.	서비스 구조	
3.3.2.	서비스 설계	
3.4.	웹 서비스	30
3.4.1.	시작페이지	
3.4.2.	사용자 정보 입력 페이지	
3.4.3.	예측 결과 출력 페이지	
3.4.4.	프로젝트 소개 페이지	
4.	연구 결과 분석 및 평가	33
4.1.	SAINT 튜닝	33
4.1.1.	Dropout 적용	
4.1.2.	Learning rate scheduler 추가	
4.1.3.	Batch normalization 적용	
4.1.4.	ReLU -> GELU	
4.1.5.	Epoch 수 조절	
4.2.	연구에 대한 평가	35
5.	결론 및 향후 연구 방향	37
5.1.	결론	
5.2.	향후 연구 방향	
6.	참고 문헌	38

1. 서론

1.1. 연구 배경

국립암센터의 연구 결과에 따르면 10년 후에는 지금보다 암 환자가 약 46% 증가, 암으로 인한 사망은 약 30% 증가할 것으로 예측한다. 세계보건기구(WHO)에서는 의학적인 관점으로 보았을 때 암 발생 인구는 1/3이 예방할 수 있으며 1/3은 조기 진단이 가능하다는 전제하에 완치까지 기대할 수 있고, 나머지 1/3은 적극적으로 치료하여 완화가 가능하다고 보고 있다. 암을 예방하는 것이 가장 중요하지만, 대부분의 암은 명확한 원인을 특정할 수 없어 예방하기 어렵다. 따라서 조기에 발견하는 것이 높은 치료 성적을 얻을 수 있고 좋은 예후를 기대할 수 있다.

암의 치료 후 5년 내 해당 암으로 사망하지 않을 확률을 ‘5년 생존율’이라고 하며 흔히 치료 성공의 척도로 보고 있다. 우리나라에서 가장 흔한 암으로 분류되는 위암의 5년 생존율은 67%인데 조기에 발견된 위암의 경우 적절한 치료가 뒷받침되면 5년 생존율이 90% 이상을 보인다. 또한 국내 4~50대 사망 원인이 1위로 분류되는 간암의 5년 생존율이 1기는 52%, 2기는 36%로 조기에만 발견한다면 비교적 높은 생존율을 보이지만, 3~4기로 진행되면 각 15%, 6%대로 생존율이 급격히 감소한다. 이러한 통계는 암을 조기에 발견하는 것이 환자의 생존율을 많이 증가시킬 수 있음을 명확히 보여준다.

그런데도 많은 사람들이 암 검진을 복잡하고 번거로운 절차로 여겨 정기적인 검진을 소홀히 하는 경향이 있다. 바쁜 일상에서 검진에 대한 필요성을 체감하지 못하거나 비용, 시간 등의 문제로 인해 검진을 미루는 사례가 많다. 이에 따라 조기 진단의 기회를 놓치게 되고, 암이 진행된 후에 발견되어 치료가 어려워지는 경우가 빈번하다. 이러한 문제를 해결하기 위해 최근 의료 기술과 인공지능(AI) 기술이 결합한 암 예측 모델이 주목받고 있다. 인공지능 기술을 활용해 표 형식 데이터에 기반한 암 발병 예측 모델을 개발하면 기존의 복잡한 검사 절차를 단순화하고 더 쉽게 암의 발병 소지를 조기에 예측할 기회를 제공할 것이다. 특히 딥러닝을 이용한 예측 모델은 다양한 신체 요인, 건강 기록, 생활 습관 등을 통합해 더 정확한 예측을 가능하게 하며 이를 통해 암 발병 위험이 높은 사람들을 조기에 선별할 수 있다.

본 연구에서는 암 발병 예측에 딥러닝을 적용하여 암의 조기 발견을 돕고 궁극적으로, 암으로 인한 사망률을 줄이는 것을 목표로 한다. 암 발병과 관련된 다양한 표 형식 데이터를 분석하고 이를 기반으로 예측 모델을 개발함으로써 사람들이 간편하게 검진하고 더 나은

치료 전략을 받아 암 예방 및 치료의 효과를 극대화할 수 있도록 하고자 한다.

1.2. 주요 문제점 분석

딥러닝은 오디오, 이미지, 텍스트와 같은 다양한 형식의 데이터에서 높은 성능을 보여주고 있다. 하지만 현실에서 많이 사용되는 데이터 중 하나인 표 형식 데이터에 대해서는 낮은 성능을 보여주고 있다. 언급되는 문제점은 다음 3가지이다.

1. 표 형식 데이터에 대한 딥러닝 과정에서 과적합(overfitting)이 쉽게 발생한다.

과적합(overfitting)은 학습 과정에서 신경망이 훈련 데이터에 과하게 적응하는 것을 의미한다. 과적합이 발생한 딥러닝 모델은 훈련 데이터에 대해서는 거의 완벽한 성능을 보이지만 실제 데이터에 대해서는 제대로 된 성능이 나오지 않는다. 표 형식 데이터는 이미지나 오디오와 같은 데이터에 비해 단순하고 제한된 형식이기 때문에 과적합이 쉽게 발생할 수 있다.

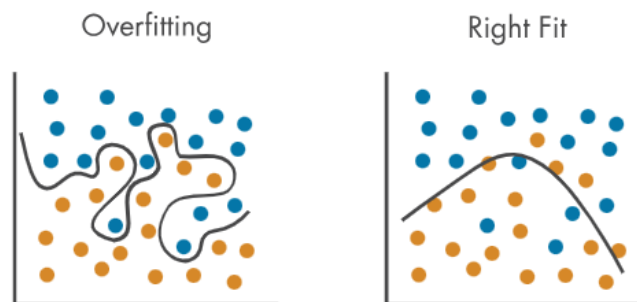


그림1. 과적합

2. 딥러닝은 표 형식 데이터의 특성을 잘 반영하지 못한다.

대부분의 딥러닝 모델은 이미지, 오디오와 같은 복잡한 데이터를 학습하는 데 특화되어 있다. 그렇기 때문에 딥러닝 모델은 표 형식 데이터와 같은 간단한 데이터의 중요 부분과 특성들을 파악하지 못할 것이다.

3. 딥러닝 결과를 설명할 방법이 부족하다.

표 형식 데이터는 주제에 대한 통계 값들과 그에 따른 결과 값들을 담고 있다. 표 형식 데이터에 대한 딥러닝은 이러한 통계 값을 이용해 결과 값을 추론하고 이를 실생활에 적용하는 것을 주로 목표로 한다. 하지만 대부분의 딥러닝 모델은 내부를 알 수 없는 “블랙박스” 형태이기 때문에 왜 이러한 값이 나왔는가 설명할 방법이 부족하다.

1.3. 연구 목표

1.3.1 표 형식 데이터에 대한 딥러닝 모델 성능 향상

표 형식 데이터에 적합한 딥러닝 모델을 선정하고 성능을 향상하는 것이 목표이다. 선정한 딥러닝 모델은 TabNet, MLP, SAINT이며 교수님의 권유로 FT-Transformer, GRANDE를 선택하였다. 추가로 MLP, Random Forest, XGBoost를 결합한 앙상블 모델도 추가하였다. 선정한 모델들에 대해 hyperparameter tuning, optimizer 및 scheduler 수정 등과 같은 방식들을 사용하여 성능을 향상한 후 서로 비교하여 표 형식 데이터에 대한 최적의 딥러닝 모델을 선정할 것이다.

또한 LightGBM, Random Forest와 같은 머신러닝 모델로도 표 형식 데이터에 대한 학습을 진행하여 딥러닝 모델과의 성능 비교도 진행할 것이다.

1.3.2 표 형식 딥러닝에 대한 해석

표 형식 데이터에 대한 딥러닝 결과를 Explainable AI를 통해 해석하는 것이 목표다. Explainable AI, 통칭 XAI는 인공지능 학습의 결과를 사용자에게 설명해 주는 AI이다. 딥러닝은 내부를 알 수 없는 “블랙박스” 형태를 취하고 있기 때문에 사용자는 딥러닝 과정에서 결과 도출 편향 등을 알기 위해서 XAI를 사용한다. 이번 연구에서는 XAI 중 SHAP 모델을 사용하기로 했다. SHAP 모델은 딥러닝에서 학습 결과에 대해 각 feature가 결과 도출 과정에 준 영향을 shap이라는 값으로 표현한다. SHAP 모델을 사용하여 표 형식 데이터에 대한 딥러닝의 결과에 각 feature가 얼마 정도의 영향을 미쳤는지 파악하고 이를 해석할 것이다.

1.3.3 암 예측 모델 개발 및 서비스

암 발병에 관한 표 형식 데이터에 대해 딥러닝과 **Explainable AI (SHAP 모델)**를 적용하여 암 발병 예측 및 예측에 대한 해석을 진행하는 모델 및 서비스를 개발할 것이다. 암 예측 모델 및 서비스의 개요는 아래와 같다.

- 암 발병에 대한 표 형식 데이터를 학습하는 딥러닝 모델 구현
- 사용자에게 각 **feature**에 대한 입력을 받아 암 발병 확률을 출력하는 서비스
- 출력된 암 발병 확률에 대해 **SHAP** 값을 통해 사용자에게 설명하는 서비스
- 서비스 사용 과정에서 사용자에게 편의를 제공하는 부가 서비스

2. 연구 배경

2.1. SAINT

2.1.1. 개요

최근 vision, NLP 분야가 급속도로 성장했지만, **tabular data**는 하지 못했다. **Tabular data**는 **categorical, continuous, ordinal feature**의 집합으로 표현되고 이러한 값은 관계가 있을 수도, 없을 수도 있다. 또한 **tabular data**는 내재적인 위치 정보가 없고 **column**의 순서가 임의적이다. 이러한 특징은 연속되고 위치에 따른 연관성을 가지는 이미지, **text**와 대조된다. 결국 **tabular data**는 다수의 **discrete, continuous** 분포를 가지는 **feature**의 연관성을 위치정보에 의존하지 않고 찾아야 하는데 이러한 **tabular data**를 위해 특화된 모델이 **SAINT(Self-Attention and Intersample Attention Transformer)** 라고 할 수 있다.

SAINT는 **tabular data** 학습의 어려움을 극복하기 위해 여러 방법을 사용한다. **categorical**뿐만 아니라 **continuous data**도 **combined dense vector**로 **projection**하고 **projection**된 **vector**는 토큰으로 취급되어 **transformer encoder**로 들어가서 **self-attention**과 **intersample attention**을 사용한다. **SAINT**에서는 **continuous, categorical** 모두 벡터화해 **transformer**에 통과시키기 때문에 **categorical**과 **continuous feature** 간의 관계성이 버려지지 않는다.

2.1.2. 구조

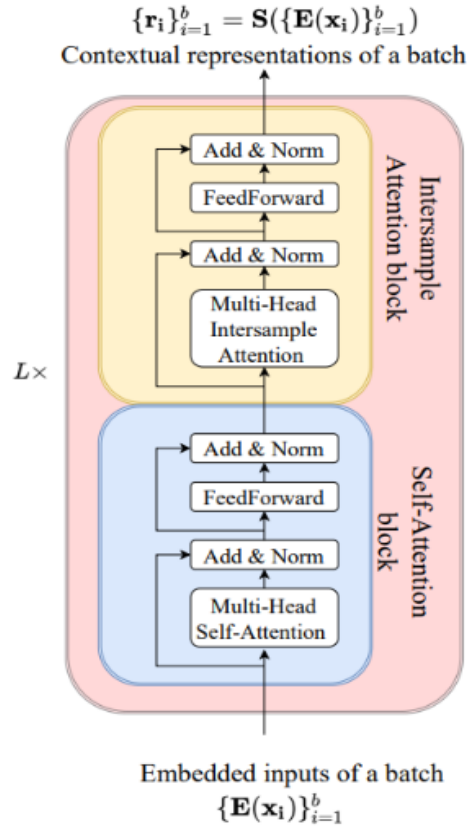


그림2. SAINT 구조

SAINT 모델은 Transformer의 Encoder 부분을 활용하고 Self-Attention 및 Intersample Attention Block이 L 번 반복되는 구조이다. MSA(Multi-Head Self-Attention)와 FF(FeedForward)/GELU, 두 계층으로 구성되며 Add & Norm 계층을 통해 skip connection과 층 정규화가 적용된다. Intersample Attention Block은 Self-Attention Block과 거의 같으나 MSA가 MISA(Multi-Head Intersample Attention)으로 대체된다.

Self-Attention은 한 데이터 내의 features 간 관련성을 파악하는 것이고 Intersample Attention은 입력된 배치 크기 내 데이터 간 연관성을 파악하는 방법이다.

Intersample attention은 row에 대하여 attention을 구하는 것으로 1개 데이터의 feature의 attention을 보는 것이 아닌 batch 단위로 보는 것이다. 만약 하나의 데이터에 noise나 missing data가 존재한다면 batch의 다른 row를 보고 추측할 수 있도록 만들어준다.

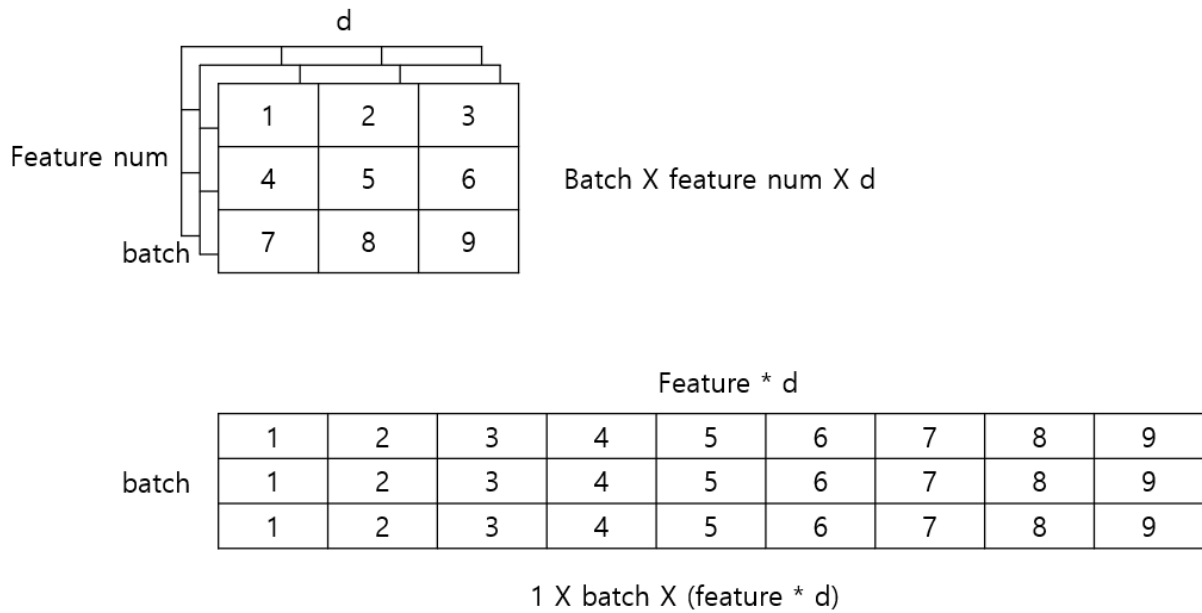


그림 3. self-attention과 intersample attention 비교

위 그림과 같이 self attention은 (feature num X d) 에 작용이 되었다면 intersample attention은 feature와 d를 퍼서 (batch X (feature * d)) 에 작용을 한다.

또한, SAINT 모델의 학습 프로세스로 self-supervised pre-training과 Supervised / Finetuning가 존재한다.

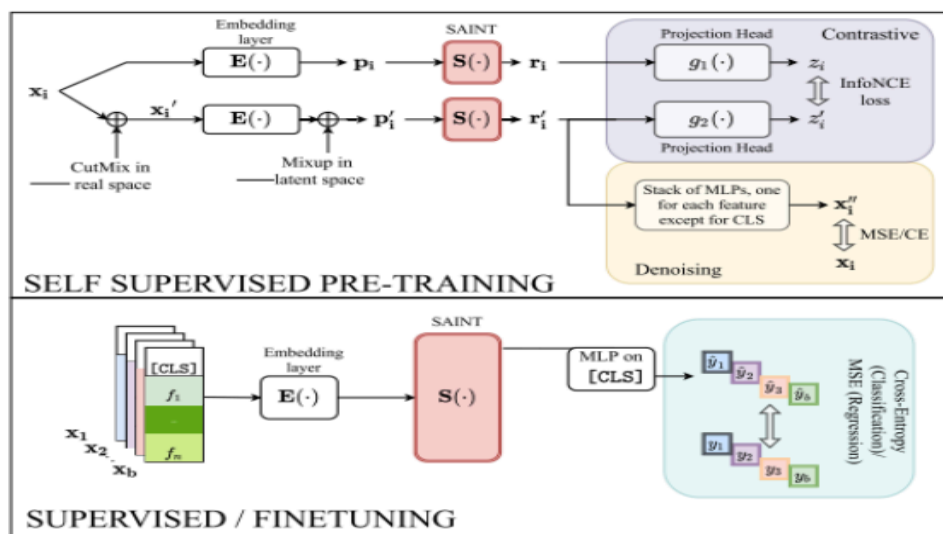


그림 4. self-supervised pre-training과 Supervised / Finetuning

이 2가지 학습 프로세스는 이러하다.

Self Supervised Pre-training

1. Self-Supervised에서 레이블링 없이 데이터를 모델에 입력
2. 데이터 증강을 위해서 Cutmix, Mixup 수행
3. Self-Attention 이후 Intersample Attention이 포함된 SAINT Block을 통해서 임베딩 벡터(Projection Head)에 투영
4. 원천 데이터와 증강 데이터의 벡터값 오차를 Loss로 계산
5. Loss를 줄이기 위해서 증강 및 딥러닝 네트워크의 가중치가 최적화됨

Supervised / Finetuning

1. Self-Supervised에 의해서 최적화된 임베딩 벡터를 SAINT Block에 입력
2. MLP를 통해 CLS 토큰(Cutmix에서 샘플 구분자) 임베딩 값만 받아서 분류 수행

2.1.3 SAINT 코드

```

# SAINT 모델 정의
class SAINT(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout_rate):
        super(SAINT, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.bn1 = nn.BatchNorm1d(hidden_dim)
        self.gelu = nn.GELU()
        self.dropout = nn.Dropout(dropout_rate)
        self.fc2 = nn.Linear(hidden_dim, hidden_dim)
        self.bn2 = nn.BatchNorm1d(hidden_dim)
        self.fc3 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        out = self.fc1(x)
        out = self.bn1(out)
        out = self.gelu(out)
        out = self.dropout(out)
        out = self.fc2(out)
        out = self.bn2(out)
        out = self.gelu(out)
        out = self.dropout(out)
        out = self.fc3(out)
        return out

```

그림 5. SAINT 코드

2.1.4 학습 방법

먼저, categorical data를 LabelEncoder로 변환해 수치형으로 인코딩한 후 독립변수(x)와 종속변수(y)로 분리하고 데이터를 train_test_split을 통해 train과 test 데이터셋으로 분리한다. 그다음 x_train, y_train, x_test, y_test 데이터를 PyTorch의 텐서로 변환해 모델 학습에 사용할 수 있도록 준비한다.

손실함수로는 CrossEntropyLoss를 사용하고 옵티마이저로 Adam을 사용하였으며 추가로 학습률 스케줄러를 설정해 학습 도중 학습률을 감소시켰다.

그다음, 모델을 학습시키고 손실값을 출력해 모델 성능을 확인한다.

2.2. 기술 스택

2.2.1. DJANGO

Django는 파이썬으로 작성된 웹 프레임워크이다. MTV 아키텍처를 따라 웹서비스 개발을

체계적으로 할 수 있다. 프로젝트의 백엔드를 담당하며, 웹 서비스의 데이터 처리 및 암 예측 딥러닝 모델의 API를 제공한다.

2.2.2. NGINX

NGINX는 오픈소스 웹 서버 프로그램으로, HTTP 및 리버스 프록시 서버 구동이 가능하다. Nginx를 사용해 Django와 Gunicorn을 연동해 웹 서비스를 지원한다. 프로젝트에서는 웹 서비스의 성능을 향상시키기 위해 Nginx를 사용해 정적파일을 관리하고, 리버스 프록시를 활용해 웹 서버에서 애플리케이션을 실행한다.

2.2.3. Gunicorn

Gunicorn은 파이썬 웹 애플리케이션을 위한 WSGI HTTP 서버이다. Django와 함께 사용해 애플리케이션을 효율적으로 배포할 수 있다. 프로젝트에서는 Nginx와 연동해 트래픽을 처리한다.

2.2.4. Docker

Docker는 프로그램을 컨테이너로 실행하고 관리하는 오픈소스 프로젝트이다. 프로젝트에서는 개발 및 배포 환경의 일관성을 유지하기 위해 구성 요소들을 Docker 컨테이너로 패키징해 사용했다.

2.2.5. AWS

AWS는 아마존에서 제공하는 클라우드 컴퓨팅 서비스 플랫폼으로 다양한 IT리소스를 사용할 수 있다. 본 프로젝트에서는 웹 애플리케이션을 배포하기 위한 인프라로 AWS를 사용했다. AWS EC2 인스턴스에 NGINX와 Docker를 사용해 Django 애플리케이션을 배포했다.

3. 연구 내용

3.1. 개발 일정

3.1.1. 상세 개발 일정

5월		6월				7월				8월				9월				10월	
3 주	4 주	1 주	2 주	3 주	4 주	1 주	2 주	3 주	4 주	1 주	2 주	3 주	4 주	1 주	2 주	3 주	4 주	1 주	2 주
주제 선정																			
		데이터 전처리, 딥러닝 모델 선택																	
						모델 성능 구현, 중간 보고서 작성													
										최종 딥러닝 모델 결정, XAI 구현									
														암 발병 예측 서비스 개발					
																		서비스 배포, 최종 보고서 작성	

3.1.2. 구성원별 역할

이름	역할
최지광	<ul style="list-style-type: none"> - 모델 구현 및 성능 향상 (MLP, FT-Transformer, GRANDE) - DJANGO를 이용한 암 발병 예측 서비스 구현 - AWS를 통한 웹 서비스 배포

송민재	<ul style="list-style-type: none"> - 모델 구현 및 성능 향상 (lightGBM, TabNet) - DJANGO를 이용한 암 발병 설명 서비스 구현
서진욱	<ul style="list-style-type: none"> - 모델 구현 및 성능 향상 (SAINT, RandomForest) - XAI SHAP 모델 구현
공통	<ul style="list-style-type: none"> - 주제 선정 및 데이터 전처리 - 보고서 작성

3.2. 딥러닝 모델 개발

3.2.1. 모델 비교

1. 랜덤 포레스트 (RandomForest)

랜덤 포레스트는 과적합 방지를 위해 최적의 기준변수를 랜덤하게 선택하는 머신러닝 기법이다. 랜덤으로 일부 **feature**만을 선택해 의사결정나무를 만들고 해당 과정을 반복해 여러 개의 의사결정나무를 형성한다.

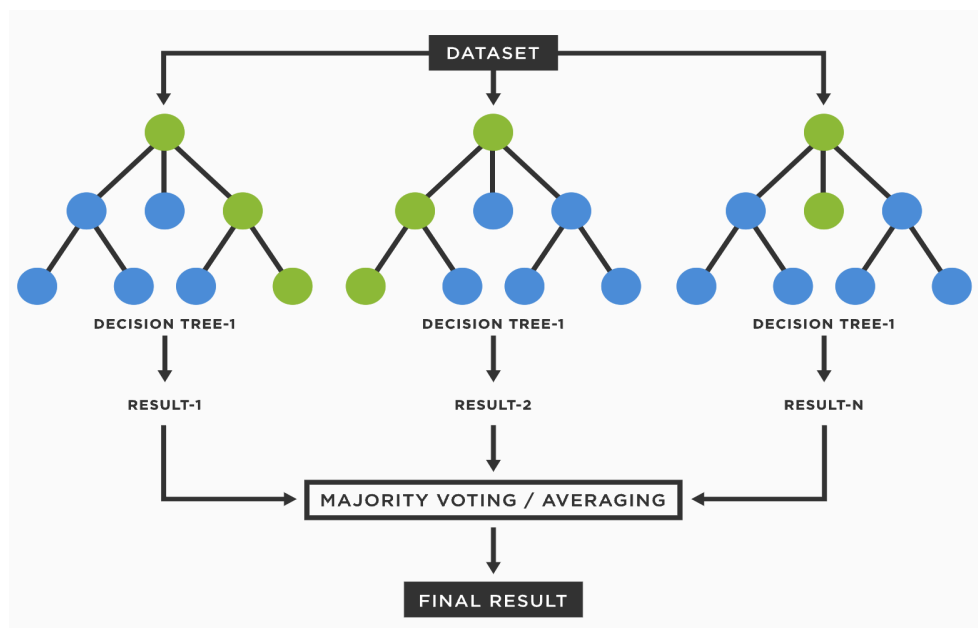


그림6. 랜덤 포레스트 구조

이 여러 개의 의사결정나무에서 나온 예측값을 토대로 가장 많이 나온 값을 최종 예측값으로 선정한다.

랜덤 포레스트는 일반화 및 성능이 우수하고 파라미터 조정이 용이하고 대용량 데이터 처리에 효과적이다. 또한 과적합 문제를 최소화하기 위해 모델의 정확도가 높은 편이다. 하지만 개별 트리 분석이 어렵고 트리 분리가 복잡해지는 경향이 존재하며 훈련 시 메모리 소모가 크다는 단점이 존재한다. 또한 **train data**를 추가해도 모델 성능 개선이 어렵다.

2. SAINT

대부분의 표 형식 데이터에서는 딥러닝보다는 트리 계열의 앙상블 모델이 가장 높은 성능을 보여주었다. 하지만 매우 큰 **dataset**에 대해서는 **SAINT** 딥러닝 모델이 가장 높은 성능을 보이는 경우가 많다. 그렇기에 **SAINT** 모델에 대해서도 성능을 평가하고 다른 모델들과 비교를 해보기로 하였다. 위 2.1. 에서 **SAINT** 모델에 대해서는 다루었으므로 간단하게 설명만 하겠다. **SAINT**는 모든 변수를 **embedding** 공간으로 결합하고 **embedding** 된 값들에 대해 **Self-Attention**과 **Intersample Attention**을 적용한다. 표 형식 데이터의 경우, 범주형, 연속형과 같이 변수 간 속성 및 분포가 달라 이질적인 **embedding** 접근 방식이 필요한데 **SAINT**는 **ReLU(GELU)**를 활용해 1차원인 연속형 변수도 **d**차원으로 **embedding** 하게 된다. 추가로 **self-supervised contrastive pre-training**을 적용하여 **semi-supervised** 문제에 대한 성능을 향상한다. 하지만 코드가 복잡해지고 성능에 큰 변화가 없다고 판단해 추가로 구현하진 않았다.

3. TabNet

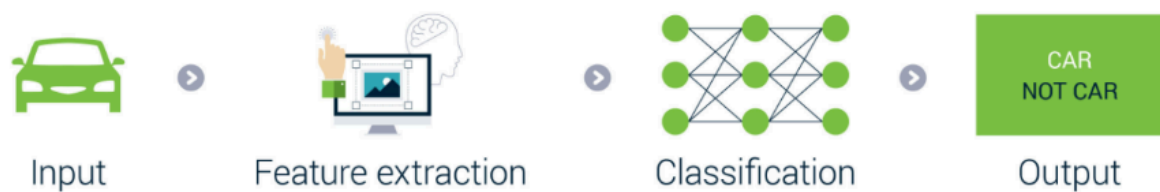
표 형식 데이터에 대한 분석은 대부분 머신러닝 계열의 모델이 높은 성능을 가지고 있다. 이러한 표 형식 데이터에 대해 높은 성능의 딥러닝 모델로 발표된 것이 바로 **Tabnet**이다. **Tabnet**이 표 형식 데이터에 대해 높은 성능을 가지게 해주는 장점은 3가지라고 하였는데 이는 다음과 같다.

1. 전처리 과정을 거치지 않은 날것의 데이터를 입력으로 받을 수 있고, **gradient-based optimization**을 사용하기 때문에 유연한 **end-to-end learning**이

가능하다.

end-to-end learning은 입력 - 출력 과정에서 추가적인 작업 없이 처리가 진행되는 학습을 의미한다. 머신러닝과 같은 일반적인 학습의 경우 입력 전처리, **Hyperparameter** 선택 등을 작업해 줘야 하지만 딥러닝과 같은 **end-to-end learning**은 이러한 과정을 한 번에 처리한다.

Machine Learning



Deep Learning



그림 7. end-to-end learning

Tabnet은 입력 전처리가 필요하지 않고 또한 **gradient**를 사용해 **Loss Function**을 최소화하는 **hyperparameter**를 찾는 **gradient-based optimization**을 사용하기 때문에 **end-to-end learning**이 가능하다.

2. **Sequential attention**을 통해 **feature selection** 단계에서 적절한 **feature**를 선택하여 학습을 진행하기 때문에 효율적인 학습이 가능하다.

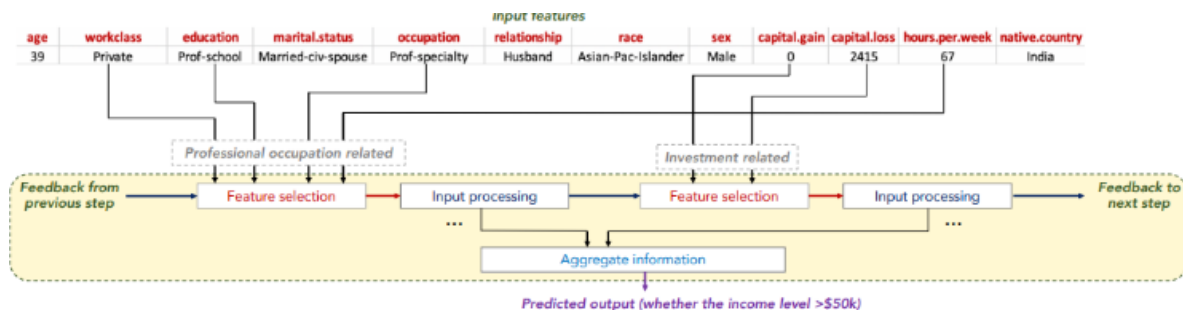


그림 8. sequential attention

위 사진은 **sequential attention**의 예시이다. 학습 주제는 성인 소득 예측으로 소득 \$550k를 기준으로 분류한다. 해당 예시의 학습 과정을 보면 먼저 직업과 같은 **feature**들을 선택하여 처리를 진행하고 이후 투자와 관련된 **feature**들을 선택하여 작업을 진행하고 있다. Tabnet은 이와 같이 다수의 **decision block**에서 특정 **feature**들을 선택하여 진행하기 때문에 효율적인 학습을 진행할 수 있다.

3. Self-supervised learning을 사용하여 좋은 성능을 보여준다.

Self-supervised learning은 unsupervised pre-training과 supervised fine-tuning을 합친 것이다.

unsupervised pre-training은 많은 양의 데이터를 label 없이 사전에 학습하여 이후 목표를 가진 학습에서 시간 축소, lost data 결정 등의 이점을 가진 학습 방식이다.

supervised fine-tuning은 어떤 작업에 특화된 모델을 만들기 위해 기존의 모델을 세세하게 조정하는 과정을 의미한다. 예를 들면 성인 소득 예측 모델에 대해 supervised fine-tuning을 적용하면 성인 소득 예측에 대해 더 집중되고 성능 좋은 모델로 조정할 수 있다는 것이다.

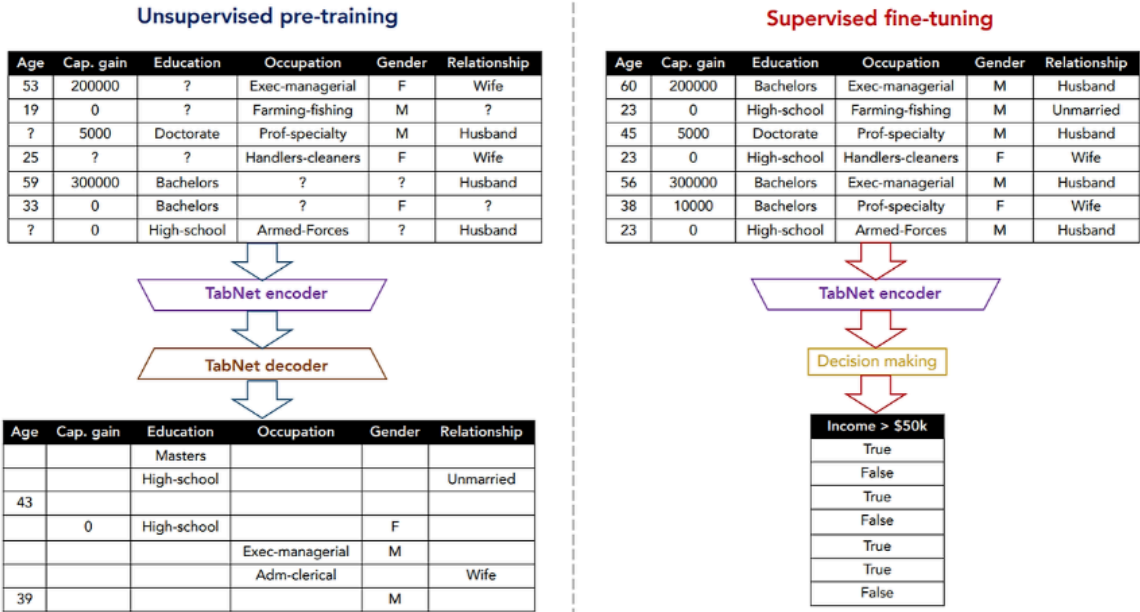
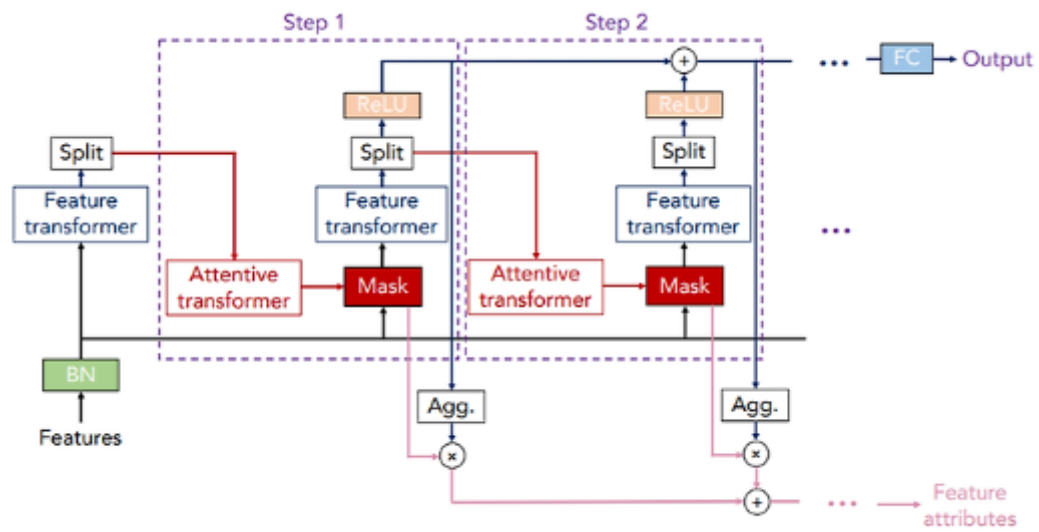


그림9. self-supervised learning

Self-supervised learning을 tabular data에 대해 적용한 것은 tabnet이 처음으로 좋은 성능을 보인다는 것을 확인했다.

tabnet의 핵심요소는 encoder와 decoder로 attentive transformer와 feature transformer로 구성되어 있다. attentive transformer는 decision step에서 각 feature가 결과에 얼마만큼의 영향을 미쳤는지 집계하는 역할을 하며 해당 집계의 결과를 Sparsemax로 일반화한 것을 Mask라 부른다. feature transformer는 주어진 feature를 통해 학습을 진행하는 기능을 가진다.

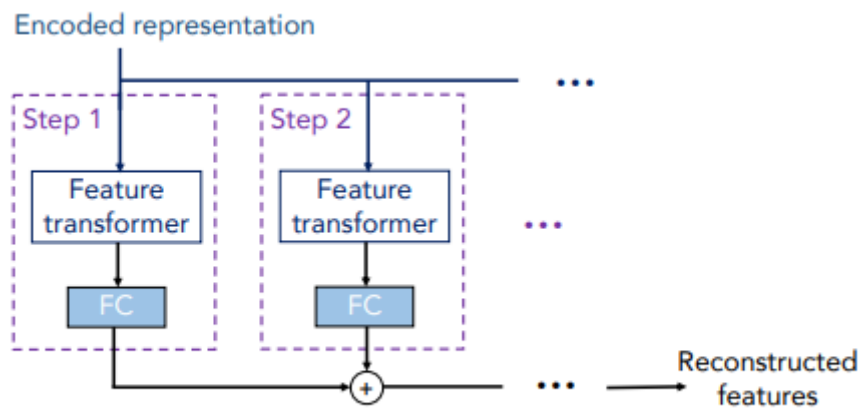
encoder는 다수의 step으로 구성된다. 각 단계는 이전 단계에서 feature transformer로 진행된 학습의 결과를 가져와 attentive transformer를 통해 mask를 생성한다. 각 단계의 mask는 합쳐져 최종 결과에 각 feature가 미친 영향을 의미하는 feature attribute를 생성한다. 생성된 mask는 feature transformer의 입력으로 들어가 데이터를 이용한 학습에 이용된다. feature transformer의 결과는 ReLU를 거쳐 최종결과를 출력할 때 사용, 그리고 다음 step의 입력으로 사용된다.



(a) TabNet encoder architecture

그림 10. TabNet encoder

decoder는 feature transformer를 이용하여 encoding의 결과를 해석할 수 있는 방식으로 decode 한다.



(b) TabNet decoder architecture

그림 11. TabNet decoder

4. lightGBM

LightGBM은 정형 데이터에 대해 좋은 성능을 보이는 머신러닝 모델 중 하나다. 많은 양의 데이터에 대해서도 빠른 속도를 가지며 정확도에 집중한다는 장점을 가지고 있다. 이러한 lightGBM에는 주요한 3가지 특징이 있다.

1. Gradient Boosting Decision Tree (GBDT)를 사용한다.

GBDT는 Decision Tree를 생성하는 과정에서 Gradient Boosting을 사용하는 방식이다. Gradient는 loss function을 최소화하기 위해 gradient의 반대 방향으로 움직이는 방식이고 Boosting은 이전 단계에서 학습이 부족한 부분을 집중적으로 학습하는 방식이다. GBDT는 이 두 가지를 모두 적용하여 Decision tree에서 leaf를 split 하는 과정에서 이 두 가지를 적용한다.

2. Gradient-based One-sided Sampling(GOSS) 알고리즘을 사용한다.

GOSS 알고리즘은 데이터셋의 instance 수를 줄이는 알고리즘이다. 많은 양의 데이터를 학습하는 과정에서 긴 시간이 걸리거나 특정 instance에 대해 학습이 부족한 경우가 생긴다. 이를 해결하기 위한 것이 GOSS 알고리즘이다. GOSS 알고리즘은 gradient가 크면 해당 instance의 변화가 loss function에 큰 변화를 주기 때문에 영향력이 크고 학습이 덜 된 instance라는 가정을 한다. 다음 학습 단계에서 이러한 instance를 포함하면서 instance를 줄여 학습하는 방식을 GOSS 알고리즘이라 한다.

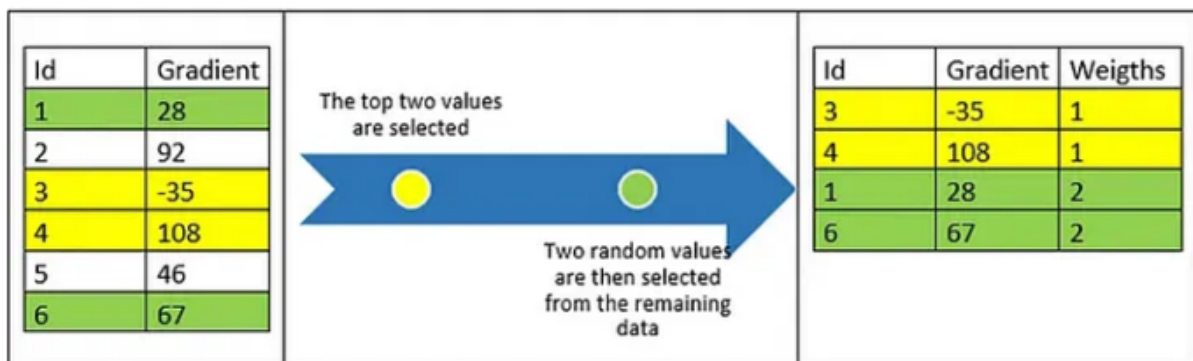


그림 12. GOSS 알고리즘

3. Exclusive Feature Bundling (EFB) 알고리즘을 사용한다.

EFB 알고리즘은 데이터셋에서 Feature의 수를 줄이는 알고리즘이다. Feature의 수가 많은 고차원 데이터에 대해 feature 개수를 줄여 학습을 용이하게 하는 것이 목적이다.

고차원 데이터를 학습할 때 “상호 배반적”인 특성이 많이 보인다. 상호 배반적인 특성은 one-hot encoding 과정에서 둘 이상에서 feature에 대해 동시에 0이 아닌 값을 가지는 경우가 매우 드문 것을 의미한다. 이러한 상호 배반적인 둘 이상의 feature를 특정 방식으로 통합하여 feature 수를 줄이는 방식을 EFB 알고리즘이라고 부른다.

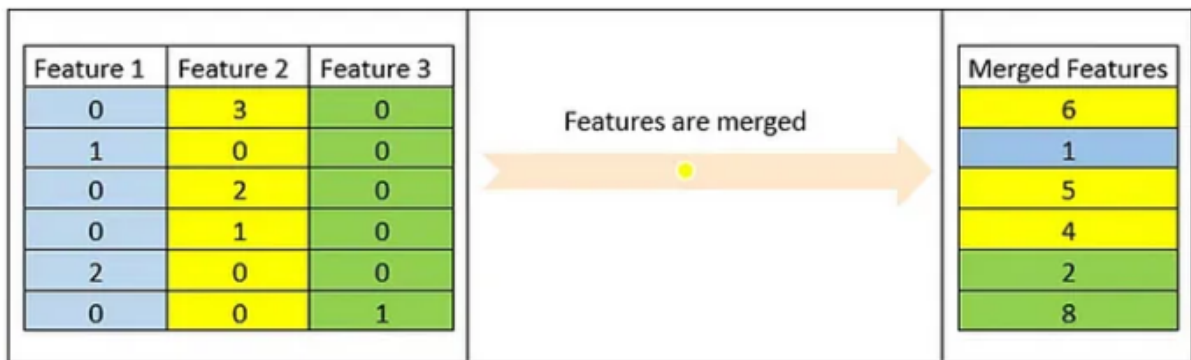


그림 13. EFB 알고리즘

5. MLP

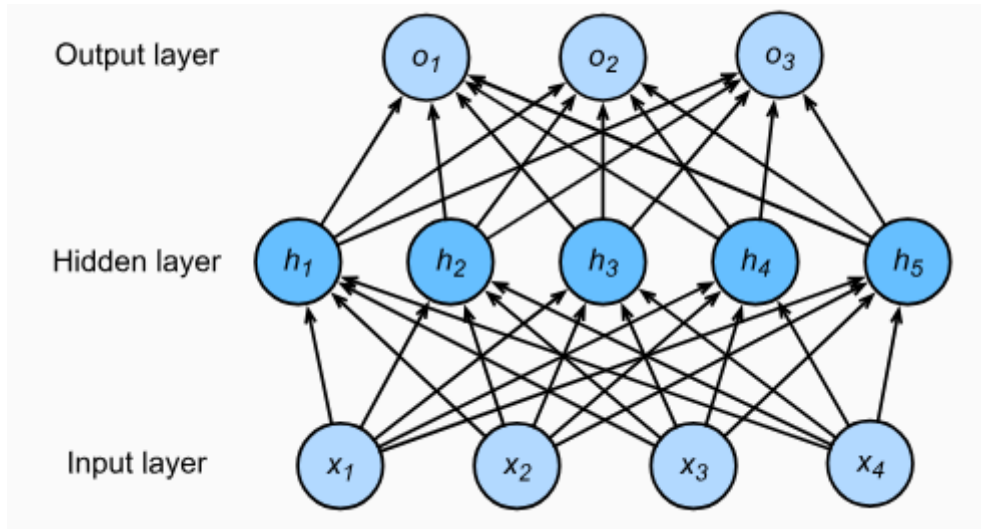


그림 14. MLP 구조

MLP(Multilayer Perceptron)는 ANN의 기본적인 형태 중 하나로, 여러 개의 뉴런으로 구성된 레이어를 사용해 데이터를 처리하고 학습한다. MLP는 입력층, 은닉층, 출력층으로 구성된다. 각 층의 뉴런들은 가중치와 편향을 학습해 예측값을 출력한다. 은닉층은 활성화 함수를 사용해 비선형성을 학습하고, 이를 통해 데이터의 패턴을 모델링 할 수 있다. MLP는 활성화 함수를 사용해 입력과 출력 간의 비선형 관계를 학습할 수 있다. 이에 따라 단순한 선형 모델이 아닌 복잡한 패턴을 학습할 수 있다.

MLP는 수치형 데이터에 적합한 모델이므로, Tabular Data를 사용하기 위해 데이터의 범주형 변수들을 수치형으로 변환하는 인코딩 작업과 정규화, 표준화 같은 전처리 과정을 진행했다.

6. FT-Transformer

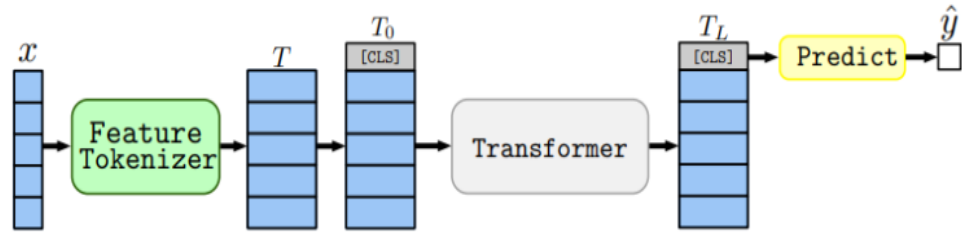


그림15. FT-Transformer 구조

FT-Transformer는 NLP, 이미지 처리에 많이 사용되던 Transformer 모델을 Tabular Data에서 활용할 수 있도록 Feature Tokenizer와 결합한 모델이다. FT-Transformer는 먼저 Feature Tokenizer를 통해 모든 범주형, 수치형 특성을 임베딩 벡터로 변환한다. 변환된 임베딩 벡터는 각 특성의 정보를 유지하면서, Transformer Layer로 전달된다.

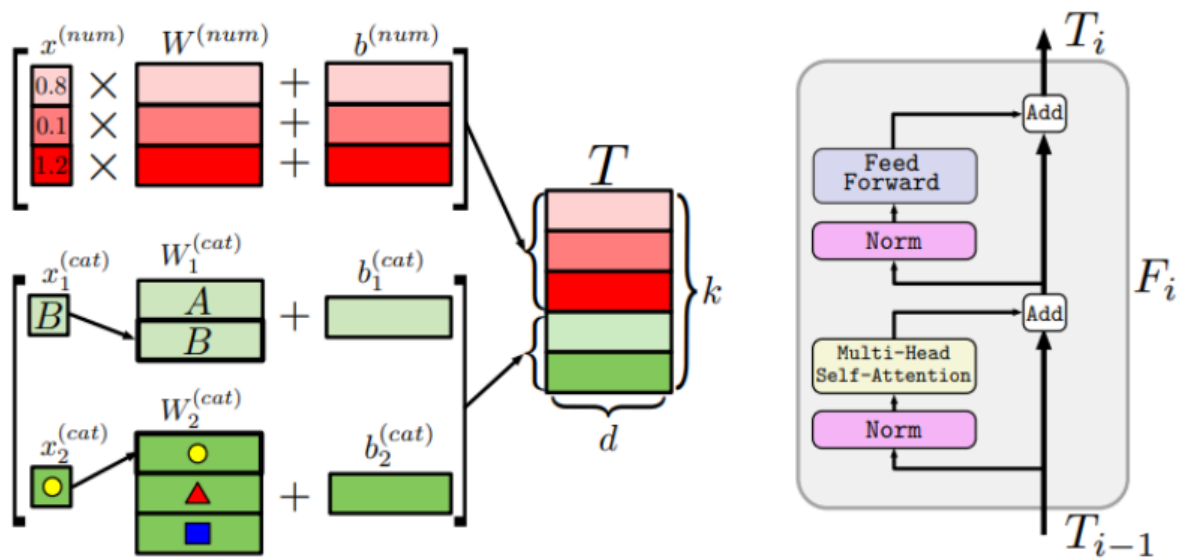


그림16. Feature Tokenizer와 Transformer Layer

임베딩 벡터는 Transformer Layer를 통과하며 학습된다. 각 레이어는 Self-Attention 메커니즘을 사용해 특성 간의 상호작용을 학습하며, 이를 통해 Tabular Data 내의 특성 간 관계를 효과적으로 학습할 수 있다. Transformer 모델에서 사용하는 CLS 토큰을 사용하고 이를 기반으로 예측 결과를 도출한다.

7. GRANDE

GRANDE(Gradient-based Decision Tree Ensembles)는 Tabular Data에서 우수한 성능을 발휘하도록 설계된 결정 트리 앙상블 모델이다. 결정 트리의 Axis-Aligned Splits와 Gradient Descent기반의 최적화를 결합해 End-to-End 학습이 가능하도록 설계되었다.

1. Axis-Aligned Decision Trees

GRANDE는 Axis-Aligned Decision Tree 방식을 채택해 비연속적인 특성을 잘 포착할 수 있다. 이에 따라 기존 딥러닝 모델들과 달리 표형식 데이터에서 비선형적인 규칙을 잘 학습할 수 있다. 특히 Dense Representation을 활용해 효율적으로 학습한다.

2. End-to-End Gradient Descent

GRANDE는 모든 결정트리 앙상블 파라미터를 Gradient Descent를 사용해 최적화한다. 이를 위해 트리 구조의 non-differentiability 문제를 해결하기 위해 Straight-Through 연산자를 활용해 역전파 시에도 Gradient가 안정적이고 효과적으로 계산된다.

3. Instance-Wise Weighting

GRANDE는 각 트리가 모든 샘플에 대해 동일한 예측을 하도록 강제하는 대신, 샘플별로 다른 가중치를 학습하는 방식을 도입했다. 이러한 방식을 통해 Local Interpretability를 높여, 특정 샘플에 대한 예측 이유를 명확히 분석할 수 있다.

8. 성능 평가

	f1 score	recall	AUC
Random Forest	0.9047	0.8995	0.8852

LightGBM	0.8860	0.8834	0.8834
MLP	0.8559	0.8655	0.8693
TabNet	0.9003	0.8976	0.8859
FT-Transformer	0.8862	0.8866	0.8843
GRANDE	0.9115	0.8879	0.9077
SAINT	0.9067	0.9002	0.8899

표1. 모델 성능 평가표

Recall 점수가 가장 높은 SAINT모델을 기반으로 SHAP모델을 구현하려고 한다.

Recall을 성능 지표로 선택한 이유는 암과 같은 중요한 질병의 예측에서 “False Negative(위음성) 의 위험을 최소화하는 것이 중요하기 때문이다. Recall을 높이는 것은 실제 암 환자를 놓치지 않는 것을 의미한다.

3.2.2. SHAP 모델

선형 회귀 또는 의사결정 트리와 같은 기계 학습의 일부 모델은 해석할 수 있다. 예를 들어 의사 결정 트리 분류기를 플로팅하면 특정 예측을 수행하는 방법을 쉽게 이해할 수 있다. 반면 딥러닝 모델은 블랙박스과 같아 모드가 어떻게 예측하는지 쉽게 이해할 수 없다.

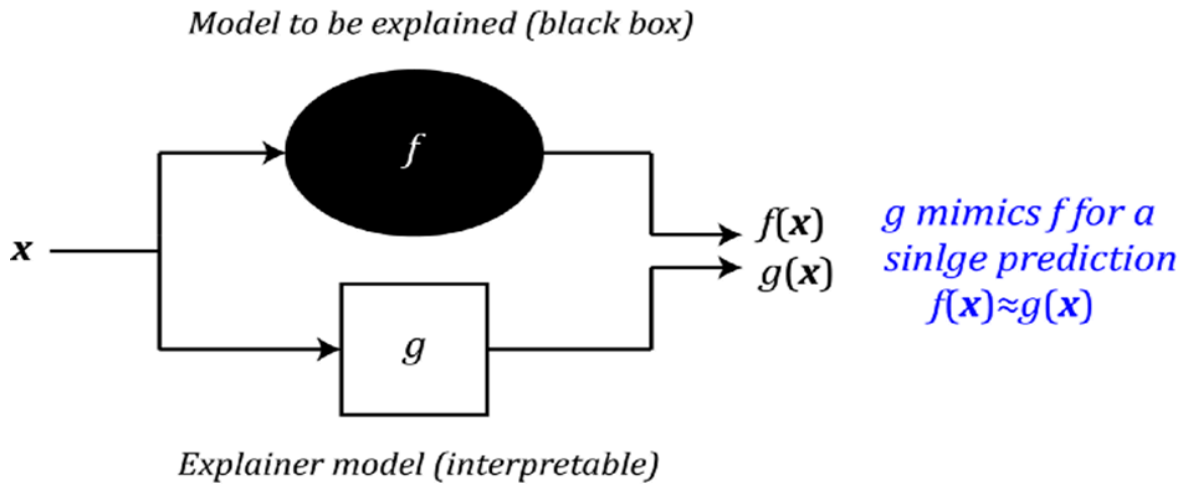


그림 17. 블랙박스과 같은 딥러닝 모델

이러한 문제에 대응하기 위해 최근 SHAP(SHapley Additive exPlanations)이라는 해석 기법이 주목받고 있다. SHAP은 모델이 내부적으로 어떻게 작동하는지 알지 못하더라도 기계 학습 모델을 설명할 수 있으며 게임이론의 개념을 사용해 이를 달성할 수 있다. 이를 이해하려면 Shapley Value에 익숙해져야 한다. Shapley Value는 게임이론을 바탕으로 Game에서 각 Player의 기여하는 정도를 계산하는 방법이다. 하나의 feature에 대한 중요도를 얻기 위해 다양한 feature의 조합을 구성하고 해당 feature의 유무에 따른 평균적인 변화를 통해 얻은 값이고 따라서 Shapley Value는 전체 성과(판단)를 창출하는데 각 feature가 얼마나 공헌했는지 수치로 표현할 수 있다.

SHAP은 Shapley Value의 조건부 기댓값으로 로이드 새플리 (Lloyd Stowell Shapley)가 만든 이론 위에 feature 간 독립성을 근거로 덧셈이 가능하게 활용도를 넓힌 기법이다. Value는 양수도 될 수 있고 음수도 될 수 있으며 이는 feature가 예측 결과를 증가시키는지 감소시키는지 나타낸다.

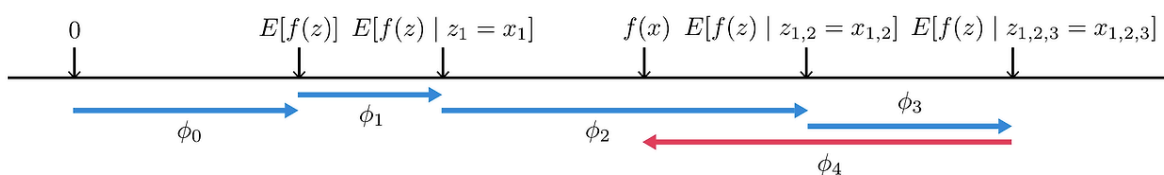


그림18. 예측에 영향을 주는 Shapley Value

오른쪽 화살표 ($\phi_0, 1, 2, 3$) 는 원점으로부터 $f(x)$ 가 높은 예측 결과를 낼 수 있게 도움을 주는 요소이고 왼쪽 화살표 (ϕ_4)는 $f(x)$ 예측에 방해가 되는 요소를 표현한 것이다. 즉, Shapley Value와 피쳐 간 독립성을 핵심 아이디어로 사용한다. 이때, Shapley Value는 전체 성과(판단)를 창출하는데 각 참여자(feature)가 얼마나 공헌했는지 기여도 수치로 표현할 수 있다. 그래서 가장 중요한 feature를 찾고 그 feature가 모델 예측에 미치는 영향을 확인할 수 있다.

SHAP은 딥러닝 모델의 해석 가능성을 크게 향상하는 강력한 도구이다. 이를 통해 모델의 예측 결과를 이해하고, 모델을 개선하거나 신뢰도를 평가하는 데 도움을 줄 수 있다. 또한 SHAP은 어떤 feature나 feature 그룹이 예측 결과에 가장 큰 영향을 미치는지를 알려주기 때문에 이를 통해 더 나은 모델 설계를 할 수도 있다.

3.2.3. SHAP 코드 (SAINT 기반)

SHAP 값을 계산하는 데 사용할 수 있는 다양한 방법은 Python으로 구현되며 Python의 SHAP 라이브러리를 사용해 모델의 SHAP값을 효율적으로 계산할 수 있다. 앞선 모델 비교를 통해 가장 성능이 좋게 나온 딥러닝 모델인 SAINT 모델을 기반으로 SHAP 코드를 구현하였고, SAINT 모델로 예측을 수행한 다음 SHAP값을 사용해 각 feature가 예측 결과에 미치는 영향을 분석하는 데 중점을 두었다.

GradientExplainer를 사용해 딥러닝 모델의 예측에 대한 SHAP 값을 계산하고 특히 PyTorch와 TensorFlow 같은 프레임워크에서 작동하는 모델에 대해 SHAP 값을 계산한다. GradientExplainer는 모델의 출력이 연속적인 값(회귀 문제 등)일 때 적합한 방법이다.

```
# GradientExplainer 사용
explainer = shap.GradientExplainer(model, X_train_tensor)
shap_values_train = explainer.shap_values(X_train_tensor)
shap_values_test = explainer.shap_values(X_test_tensor)
```

그림19. GradientExplainer

SHAP 값은 각 샘플의 각 특성에 대해 계산되므로, 원래는 3차원 배열이다. 이를 `shap_values_train_2d`로 2차원 배열로 변환하여 각 샘플과 feature에 대한 SHAP 값을 쉽게 분석할 수 있게 하였다. 그리고 `shap_values_train_df`는 변환된 SHAP 값을 데이터프레임으로 변환해 각 특성의 SHAP 값을 열로 표현한다. 이를 통해 각 샘플에 대해 feature별 SHAP값을 시각적으로도 확인할 수 있다.

```
shap_values_train_2d = np.squeeze(shap_values_train)
```

그림20. `shap_values_train_2d`

```
shap_values_train_df = pd.DataFrame(shap_values_train_2d, columns=[f'{col}_Shap' for col in columns])
```

그림21. `shap_values_train_df`

SHAP의 중요한 특성 중 하나는 각 샘플의 예측값을 구성하는 feature들의 기여도를 SHAP 값을 통해 합산할 수 있다는 점이다. `train_predictions`는 모델이 예측한 값이고 `ShapValues_Sum`은 해당 샘플에 대해 SHAP 값들의 합이다. 이 SHAP 값의 합에 모델의 기본 예측값(베이스라인)을 더한 것이 실제 예측값과 일치해야 한다. `BaseValue`는 모델이 어떤 특성 정보도 모를 때의 기본 예측값으로 이를 통해 모델이 새로운 데이터를 입력받았을 때, 각 feature가 어떻게 예측값을 바꿨는지 분석할 수 있다.

```
# 모델의 예측값 생성
train_predictions = model(X_train_tensor).detach().numpy()
```

그림22. `train_predictions`

```
# SHAP 값의 합산과 기본값 더하기
data_shap_train['ShapValues_Sum'] = shap_values_train_df.sum(axis=1)
data_shap_train['(ShapValues + BaseValue)'] = data_shap_train['ShapValues_Sum'] + data_shap_train['BaseValue']
```

그림23. `ShapValues_Sum`

```
# 예측값 추가
data_shap_train['Prediction'] = train_predictions

# 예측값의 평균을 기본값으로 설정
base_value = np.mean(train_predictions)
data_shap_train['BaseValue'] = base_value
```

그림24. 예측값 추가 및 **BaseValue** 설정

그 후 특성에 대한 **SHAP** 값의 절대값을 평균 내어, 모델 예측에 있어 각 특성의 기여도를 수치상으로 평가한다. **SHAP** 값이 클수록 해당 특성은 모델의 예측에 더 많은 영향을 미쳤다는 것이다.

`shap.summary_plot`은 **SHAP**에서 제공하는 대표적인 시각화 함수로, 각 특성의 중요도를 직관적으로 보여준다. 코드에서는 막대그래프(**bar**)를 통해 각 **feature**의 평균**SHAP** 값을 시각화하여 어떤 특성이 모델에 가장 큰 영향을 미쳤는지 쉽게 파악하고 비교할 수 있게 하였다. 큰 영향력을 보일수록 암의 발병(**diagnosis**)과 관계성이 크다는 것이다. 즉, 변수의 중요도와 비슷한 개념이다.

```
# bar 그래프로 출력
shap.summary_plot(shap_values_train_2d, X_train, feature_names=columns, plot_type='bar')
```

그림25. **Bar** 그래프 출력 코드

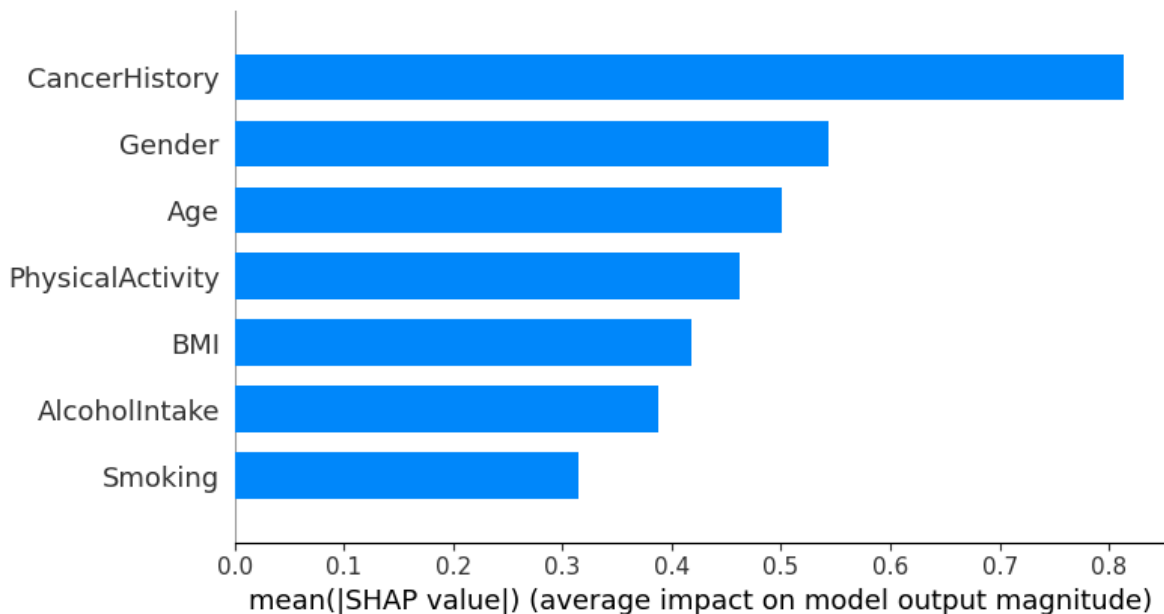


그림 26. Bar 그래프

또한 점 그래프(summary plot)을 사용해 특성에 대해 모든 샘플의 SHAP 값을 시각적으로 표현하여 feature가 모델 예측에 미친 영향을 개별적으로 확인할 수 있게 하였다. x축은 SHAP값으로 양수이면 feature가 예측을 더 높이는 방향으로 기여했다는 것을 뜻한다. 점의 색은 해당 feature의 값 크기로 빨간색은 feature 값이 높을 때 파란색은 feature 값이 낮을 때를 말한다. 따라서 빨간색 점이 양수에 많으면 feature 값이 높을수록 암의 발병에 더 영향을 많이 미친다고 할 수 있다. 예를 들어 AlcoholIntake나 BMI는 클수록 암이 발병될 확률이 높으므로 빨간색 점이 양수부분에 많이 분포되어있다. 해당 변수들에 대한 해석을 하면 아래와 같다.

- CancerHistory : 이전 암 병력이 있으면 암의 발병률이 높은 경향이 있다.
- Gender : 여성(1)일수록, 암의 발병률이 높은 경향이 있다.
- Age : 변수의 값이 높을수록 암의 발병률이 높은 경향이 있다.
- PhysicalActivity : 변수의 값이 낮을수록 암의 발병률이 높은 경향이 있다.
- BMI : 변수의 값이 높을수록, 암의 발병률이 높은 경향이 있다.
- AlcoholIntake : 변수의 값이 높을수록 암의 발병률이 높은 경향이 있다

- Smoking : 흡연자일수록 암의 발병률이 높은 경향이 있다.

```
# 트레이닝 데이터의 SHAP Summary plot
shap.summary_plot(shap_values_train_2d, X_train, feature_names=columns)
```

그림27. 점 그래프 출력 코드

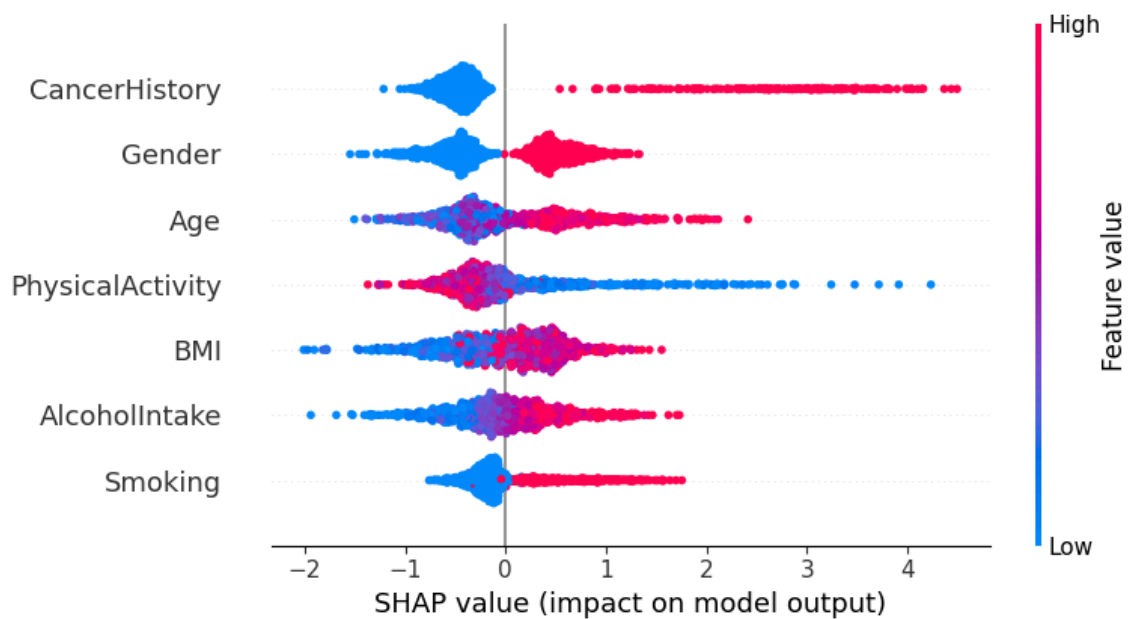


그림28. 점 그래프

또한 confusion matrix로 시각화를 해 예측값과 실제값을 비교해 정확한 비율을 확인할 수 있었다.

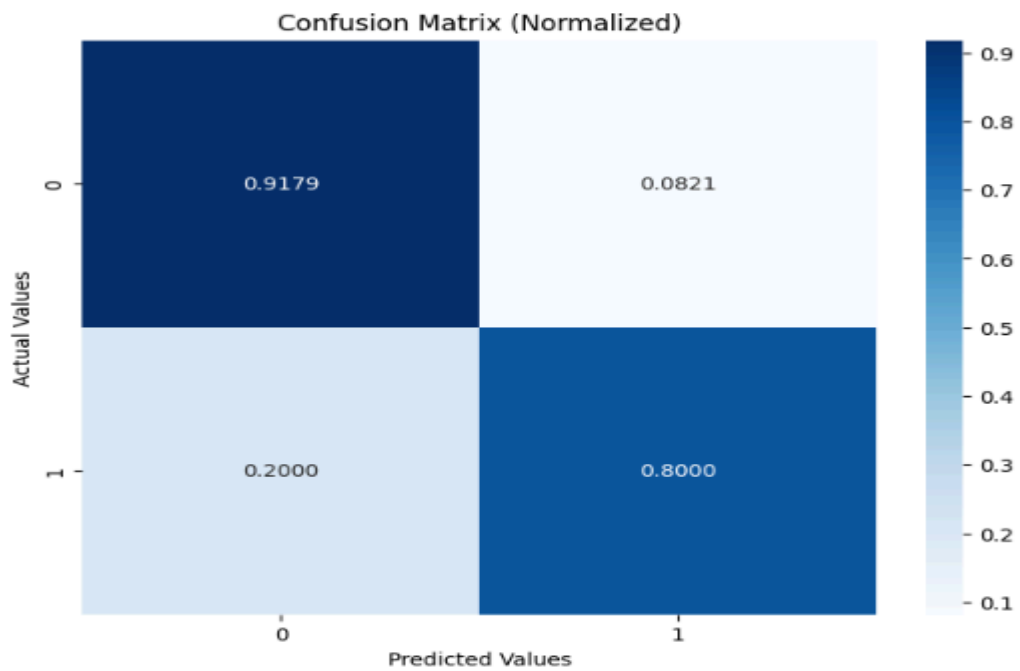


그림 29. Confusion matrix

3.3. 딥러닝 모델을 사용한 웹서비스 설계

3.3.1. 서비스 구조

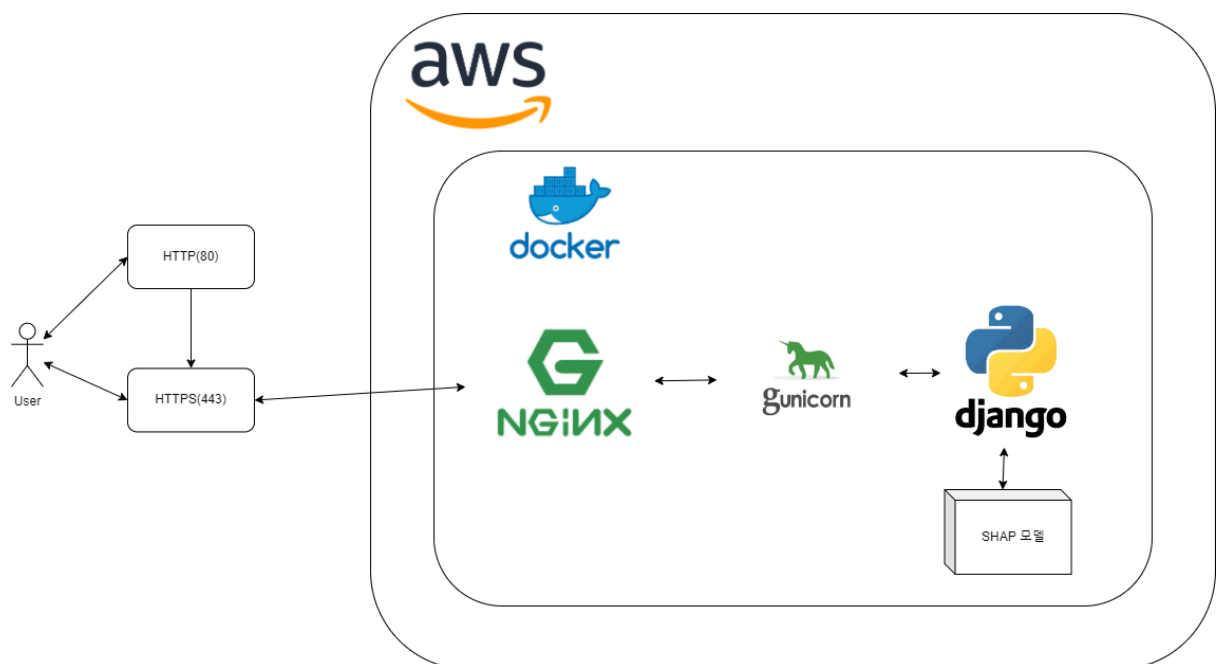


그림 30. 서비스 구조도

사용자는 웹 브라우저를 물음에 대해 입력한다. 해당 입력값은 **AWS** 환경에서 호스팅되고 있는 애플리케이션으로 전달된다. **HTTP**로 들어오는 트래픽은 **HTTPS**로 자동 리다이렉트되며, 이후 **nginx** 웹 서버가 내부의 **Gunicorn** 서버로 트래픽을 전달한다. **Django**는 미리 학습되어 있는 **SAINT** 모델을 바탕으로 사용자의 암 발병 확률을 계산하고, 예측값을 **SHAP**모델이 해석해 각 입력이 출력값에 어떻게 기여했는지 설명한다. 위 예측값과 **SHAP**모델 결과값 2개를 사용자 페이지에 출력한다.

3.3.2 Django Application 개발

어플리케이션 내의 디렉토리를 2개로 나누어 **WSGI**서버 배포를 위한 디렉토리 와 주요 비즈니스 로직이 구현된 디렉토리로 나누었다. **Prediction** 디렉토리에서 미리 학습된 **Saint** 모델 파일을 사용해 사용자 입력에 대한 답변 속도를 향상했다. 사용자 입력값은 1차로 **Saint** 모델을 거쳐 암 발병 예측값을 생성하고, 암 발병 예측값을 **SHAP** 모델의 입력값으로 사용해 해당 입력값에 대한 해석 값과 **Train** 데이터에 대한 해석 값을 비교해 사용자 입력값 중 어떤 부분이 더 영향을 미쳤는지, 덜 영향을 미쳤는지를 확인할 수 있다.

3.4. 웹 서비스

3.4.1 시작페이지

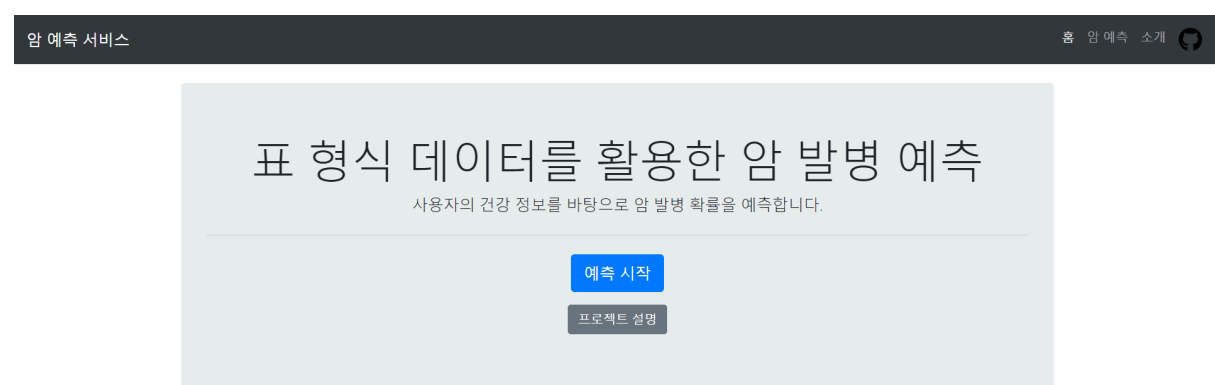
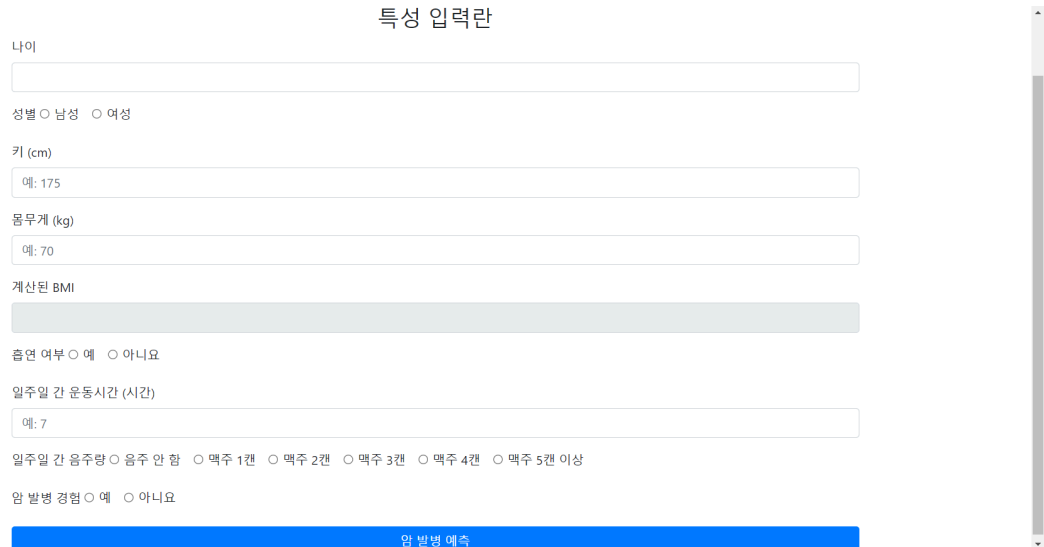


그림31. 시작 페이지

웹 서비스의 메인 페이지이다. 이 페이지에서 사용자 정보입력 페이지와 프로젝트 소개 페이지로 이동할 수 있다.

3.4.2 사용자 정보 입력 페이지



The form is titled '특성 입력란' (Characteristic Input Section). It contains several input fields and radio buttons for user information. The fields are: '나이' (Age) with a text input; '성별' (Gender) with radio buttons for '남성' (Male) and '여성' (Female); '키 (cm)' (Height in cm) with a text input showing '예: 175'; '몸무게 (kg)' (Weight in kg) with a text input showing '예: 70'; '계산된 BMI' (Calculated BMI) with a shaded rectangular area; '흡연 여부' (Smoking status) with radio buttons for '예' (Yes) and '아니요' (No); '일주일 간 운동시간 (시간)' (Weekly exercise time in hours) with a text input showing '예: 7'; and '일주일 간 음주량' (Weekly alcohol consumption) with radio buttons for '음주 안 함' (Don't drink), '맥주 1캔', '맥주 2캔', '맥주 3캔', '맥주 4캔', and '맥주 5캔 이상'. Below these is another radio button for '암 발병 경험' (Cancer experience) with '예' (Yes) and '아니요' (No) options. At the bottom is a prominent blue button labeled '암 발병 예측' (Cancer prediction).

그림32. 사용자 정보 입력 페이지

사용자의 건강 관련 데이터를 입력하는 페이지이다. 각 특성을 입력한 후 '암 발병 예측'버튼을 누르면 예측 결과를 출력하는 페이지로 이동한다.

3.4.3 예측 결과 출력 페이지

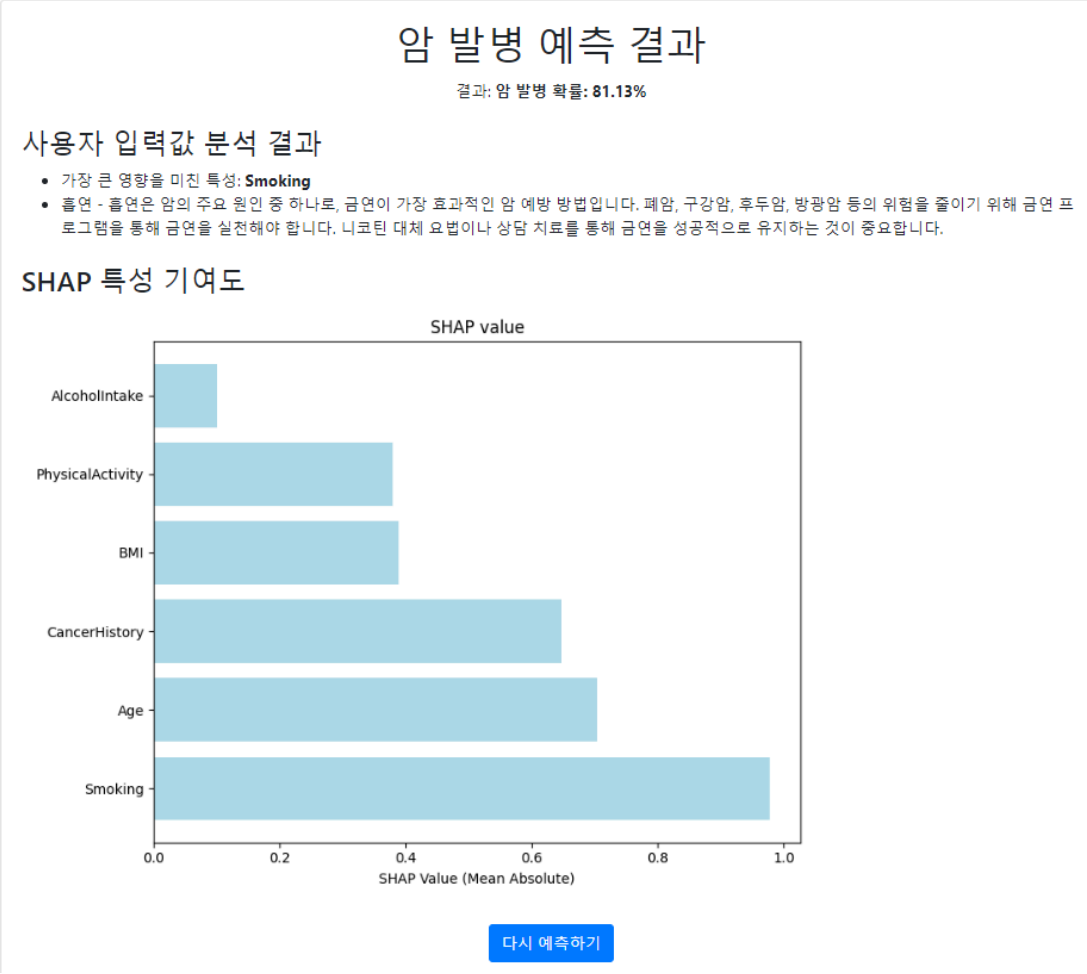


그림 33. 예측 결과 출력 페이지

사용자가 입력한 건강 관련 데이터를 바탕으로 SAINT모델이 예측한 암 발병 확률을 보여준다. 출력으로는 암 발병 확률, SHAP 값을 통해 각 입력 특성이 예측 결과에 미친 영향을 그래프로 출력한다. 그리고 가장 영향을 많이 끼친 특성에 대한 권장 사항을 확인할 수 있다.

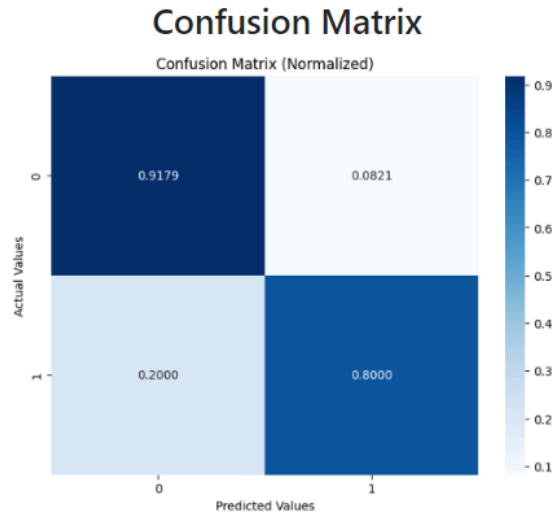
3.4.4. 프로젝트 소개 페이지

프로젝트 설명

본 웹서비스는 표 형식 데이터 딥러닝을 위한 프로젝트로 개발되었습니다.

이 프로젝트는 사용자로부터 다양한 건강 관련 데이터를 입력받아, 암 발생 여부를 예측하는 딥러닝 모델을 통해 결과를 제공합니다. 입력된 데이터는 나이, 성별, 흡연 여부, 운동 시간, 음주량 등 다양한 건강 지표로 구성되어 있습니다.

이 웹사이트는 교육 및 연구 목적으로 사용됩니다.



Confusion Matrix는 SAINT 모델이 예측한 결과의 여러 지표를 나타냅니다. 0과 1의 값은 각각 암 발생 가능성 여부를 나타냅니다.

그림 34. 프로젝트 설명 페이지

웹 서비스의 목적과 SAINT 모델, SHAP에 대한 설명을 제공한다. 훈련된 SAINT 모델의 Confusion Matrix, ROC 그래프와 훈련 데이터를 사용한 SHAP에 대한 그래프와 그에 대한 설명을 출력한다.

4. 연구 결과 분석 및 평가

4.1. SAINT 튜닝

4.1.1. Dropout 적용

Dropout은 과적합(overfitting)을 방지하기 위한 정규화 기법으로 학습 중 특정 뉴런을 무작위로 비활성화하여 모델이 특정 뉴런에 지나치게 의존하지 않도록 한다. 코드에서는 Dropout의 비율을 0.3을 사용했는데 이는 30%의 뉴런을 학습 시 임의로 비활성화한다는 의미이다.

```
self.dropout = nn.Dropout(0.3)
```

그림 35. Dropout 적용

4.1.2. Learning rate scheduler 추가

학습률(Learning Rate)은 모델 학습에서 매우 중요한 하이퍼파라미터로, 너무 크면 최적의 가중치에 도달하지 못하고, 너무 작으면 학습 속도가 느려진다. Learning rate scheduler는 학습이 진행됨에 따라 학습률을 점진적으로 줄여 학습 초반엔 빠르게 수렴하고 후반엔 세밀하게 최적화를 진행할 수 있도록 도와준다. 학습률을 조절하지 않으면 초기에 과도한 학습률로 인해 최적의 가중치를 놓치거나, 너무 작은 학습률로 인해 학습이 지연될 수 있기 때문에 scheduler를 추가해 주었다. 코드에서는 StepLR scheduler를 사용했으며 step_size를 10으로 설정해 10 epochs마다 학습률을 줄이고 gamma는 0.1로 설정해 학습률을 10%로 줄이도록 하였다.

```
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)
```

그림 36. Scheduler 추가

4.1.3. Batch normalization 적용

배치 정규화(Batch normalization)는 각 미니 배치에서 입력의 평균과 분산을 정규화하여 학습을 안정화하고, 학습 속도를 높이는 기술이다. 이는 네트워크가 층을 깊게 쌓아도 각 층의 활성화 함수가 폭발적이거나 소멸하지 않도록 도와준다. 배치 정규화를 적용하면 학습의 불안정성을 줄이고, 더 높은 학습률로 학습할 수 있게 할 수 있고 또한 과적합을 줄이고 모델의 일반화 성능을 향상하기에 적용을 해 최적화를 시키고자 하였다. 코드의 BatchNorm1d 부분은 각 은닉층에 적용된 배치 정규화로 은닉층의 뉴런 출력을 정규화하여 학습을 더 빠르고 안정적으로 만들고자 하였다.

```
self.bn1 = nn.BatchNorm1d(hidden_dim)
```

그림 37. BatchNorm1d (1)

```
self.bn2 = nn.BatchNorm1d(hidden_dim)
```

그림 38. BatchNorm1d (2)

4.1.4. ReLU -> GELU

활성화 함수는 신경망 모델에서 중요한 요소이다. 전통적으로 ReLU(Rectified Linear Unit)가 많이 사용되었으나, GELU(Gaussian Error Linear Unit)는 최근 더 나은 성능을 보이는 것으로 알려져 있다. ReLU는 입력값이 양수일 때는 그대로 통과시키고, 음수일 때는 0으로 만드는 단순한 비선형 함수이지만 GELU는 ReLU보다 부드럽게 작동하며, 신경망이 학습하는 데 있어 더 적합한 성질을 가지고 있다고 볼 수 있다. 특히, GELU는 작은 값도 일부 통과시키는 성질이 있어 더 미세한 학습에 적합하다.

```
self.gelu = nn.GELU()
```

그림 39. GELU 함수

4.1.5. Epoch 수 조절

Epoch는 전체 데이터셋을 몇 번 반복해서 학습할지를 결정하는 하이퍼파라미터이다. 일반적으로 epoch 수가 너무 적으면 모델이 충분히 학습하지 못해 성능이 낮아지고 너무 많으면 과적합이 발생할 수 있다. 그렇기에 적절한 에포크 수를 선택하는 것은 학습의 효율성을 높이고, 적절한 시점에 학습을 중지하여 모델의 일반화 성능을 높이는 데 매우 중요하다.

4.2. 연구에 대한 평가

위와 같이 하이퍼 파라미터를 최적화했을 때 SAINT 성능이 향상되는 것을 볼 수 있었다.

	precision	recall	f1-score	support
Not Cancer	0,8265	0,8710	0,8482	93
Cancer	0,7692	0,7018	0,7339	57
accuracy			0,8067	150
macro avg	0,7979	0,7864	0,7911	150
weighted avg	0,8048	0,8067	0,8048	150

그림36. 최적화 전 SAINT 성능

	precision	recall	f1-score	support
Not Cancer	0,8900	0,9082	0,8990	98
Cancer	0,8200	0,7885	0,8039	52
accuracy			0,8667	150
macro avg	0,8550	0,8483	0,8515	150
weighted avg	0,8657	0,8667	0,8660	150

그림40. 최적화 후 SAINT 성능

SAINT 기반의 SHAP은 SAINT 모델보다 성능 면에서는 다소 저하될 수 있다. 이는 SHAP이 각 feature가 예측에 미치는 영향을 설명하는 데 있어 추가적인 계산 비용이 발생하기 때문이다. 그런데도, SHAP의 강점은 모델의 신뢰성을 높이고 모델 결정에 대한 설명력을 제공하는 것이다. 이에 따라 SAINT 모델의 예측 결과를 더 잘 이해하고 feature importance도 도출할 수 있다.

표 형식 데이터를 다룰 때, 딥러닝 모델은 머신러닝에 비해 성능이 낮게 나오는 경우가 있다. 이는 딥러닝 모델이 주로 이미지나 텍스트와 같은 비정형 데이터를 다루는 데 최적화되어 있기 때문이다. 하지만 본 연구에서는 이를 극복하기 위해 하이퍼 파라미터 최적화에 집중했고 이를 통해 딥러닝 모델의 성능을 개선할 수 있었다.

특히, SAINT 모델을 사용해 성능 향상을 보일 수 있었다. 하이퍼 파라미터 튜닝을 통해 최적화된 SAINT 모델은 표 형식 데이터에서 좋은 예측 결과를 보여주었으며 암 발병 예측에서도 의미 있는 결과를 도출할 수 있었다. 이에 따라 본 연구의 예측 모델은 높은 신뢰도를 바탕으로 암 조기 진단에 중요한 역할을 할 수 있을 것으로 기대된다.

5. 결론 및 향후 연구 방향

5.1 결론

이번 연구에서 Tabnet, MLP, SAINT 등과 같은 딥러닝 모델들을 구현 및 성능 향상을 통해 정형 데이터에 대한 높은 성능의 학습을 진행하였다. 또한 딥러닝 모델과 머신러닝과의 비교를 위해 RandomForest, lightGBM 등과 같은 머신러닝 모델도 간단하게 학습하였다. 딥러닝과 머신러닝 모델들 간의 정형 데이터 학습 결과를 비교하여 딥러닝 모델의 학습 결과가 머신러닝만큼의 성능은 나오지만 크게 뛰어넘는 모델은 찾지 못하였다. 딥러닝 모델 중 제일 좋은 결과를 보인 SAINT 모델을 암 발병 예측 서비스에 사용하기로 결정했다.

추가로 XAI 중 하나인 SHAP 모델을 구현하여 흔히 “블랙박스” 형태를 한 딥러닝 식으로 표현할 수 있다는 것을 확인하고 이를 암 모델의 결과 도출 과정을 표나 그래프로 형발병 예측 서비스에서 결과에 대한 해석을 사용자에게 보여줄 때 사용하기로 했다.

5.2 향후 연구 방향

이번 프로젝트의 향후 연구 방향은 다음 3가지와 같다.

1. 더 풍부한 데이터셋 구하기

이번 프로젝트를 진행하면서 데이터셋에 대한 부족한 점이 몇 가지 있었다. 첫째는 데이터의 양이 부족하다는 것이었다. 기본적으로 의료 데이터는 보안상 구하기가 쉽지 않았다. 그렇기에 프로젝트 과정에서 다소 적은 양의 데이터셋으로 학습을 진행해야 했다. 두 번째는 사용한 데이터의 디테일이 아주 부족했다. 예를 들면 유전 확률에 대한 기준이 정확히 제시되어 있지 않아 학습 과정에서 사용하지 못한 점, 발병한 암의 종류가 제시되어 있지 않아 여성이나 남성이 특히 더 많이 걸리는 암에 대한 정보를 학습 과정에서 사용하지 못한 점 등이 있다.

우리의 목표는 의학 데이터를 정식으로 받을 수 있는 연구 기관 등과 협력하여 더 상세하고 풍부한 데이터를 받아 학습의 질을 향상하고 싶다.

2. 암 발병 확률 예측 서비스의 기능 확장

현재 제공되는 서비스는 입력에 대해 암 발병 확률 예측과 이에 대한 간단한 설명만 제공하고 있다. 우리는 향후 이 서비스에 다음과 같은 추가적인 서비스를 제공하고 싶다.

- 실제 암 관련 기관과 협력하여 암 발병 확률에 대한 자세한 설명 및 대응법을 제시
- 사용자 근처에 암 진단과 관련된 병원이나 단체에 대한 정보 제공
- 그 외 암과 관련된 정보 및 서비스를 제공

3. 정형 데이터에 대한 딥러닝 성능 향상 및 새로운 모델 구현

암 발병 예측은 우리 서비스의 매우 중요한 부분이기 때문에 높은 성능을 보이는 딥러닝 모델을 사용하는 것이 중요하다. 하지만 아직 정형 데이터에 대해 머신러닝에 비해 아주 높은 성능의 딥러닝 모델을 찾지 못했다. 향후 추가적인 공부 및 연구를 통해 정형 데이터에 대해 머신러닝에 비해 매우 높은 성능을 보이는 딥러닝을 구현할 것이다.

6. 참고 문헌

- [1] RABIE EI KHAROUA, (2024.05), Cancer Prediction Dataset,
<https://www.kaggle.com/datasets/rabieelkharoua/cancer-prediction-dataset/data>

[2] Sercan " O. Arık, Tomas Pfister, (2021), TabNet: Attentive Interpretable Tabular Learning,
<https://arxiv.org/pdf/1908.07442>

[3] Guolin Ke, (2017), LightGBM: A Highly Efficient Gradient Boosting Decision Tree,
https://papers.nips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[4] Turkish Technology, (2023.11,28), Light GBM Light and Powerful Gradient Boost Algorithm,
<https://medium.com/@turkishtechnology/light-gbm-light-and-powerful-gradient-boost-algorithm-eaa1e804eca8>

[5] 김해뉴스, (2020.12.08), 암 조기 발견의 중요성

https://www.hgkmc.kr/bbs/board.php?bo_table=notice&wr_id=117&page=3

[6] Gowthami Somepalli 외 4인, (2021), SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training

<https://arxiv.org/pdf/2106.01342>

[7] Urban communicator, (2023.01.13), [ML] SHAP 소개

<https://narrowmoon.tistory.com/9>

[8] yikwon, (2022.06.30) , Random_Forest

https://incodom.kr/Random_Forest