머신러닝 실습 프로젝트 #3

딥러닝 기본 구조 실습 및 전통 ML(XGBoost) 성능 비교

1. 실습 개요  
    이 실습은 PyTorch를 이용해 딥러닝 모델의 기본 구조를 직접 구현하고,  
   전통적인 머신러닝 모델(XGBoost)과의 성능을 비교하는 프로젝트이다. 핵심 목적은 다음과 같다.   
   **(1) DNN 모델의 구조와 학습 과정을 이해하고 직접 구현해본다.  
   (2) 하이퍼파라미터 튜닝을 통해 모델 성능을 개선하는 과정을 실험한다.**

**(3) DNN과 전통 ML 모델의 성능 차이를 비교하고, 그 원인을 분석한다.**

1. 데이터 출처

데이터는 Kaggle의 Santander Customer Transaction Prediction 대회에서 제공된 것임. (https://www.kaggle.com/competitions/santander-customer-transaction-prediction)

총 200,000개의 샘플로 구성되어 있으며, 각 샘플은 고객의 익명화된 수치형 변수 200개(var\_0 ~ var\_199)를 포함한다.

타깃 레이블 target은 0 또는 1의 값을 가지며, 1은 해당 고객이 특정 거래를 수행했다는 의미이다. 데이터 불균형을 없애고, 두 클래스의 비율을 1:1로 맞춘 데이터를 올려두었으니 다운받아서 이용한다.

* 파일명: train\_small.tsv, valid\_small.tsv

1. 실습 목표

 - PyTorch를 사용하여 DNN 모델을 처음부터 구현하고 학습해본다.

 - 중간 레이어 크기, dropout 확률, learning rate 등 하이퍼파라미터를 조정해 성능 향상을 시도한다.

 - 동일한 데이터에 대해 XGBoost를 적용하고 DNN과 성능을 비교한다.

 - 두 모델 간 성능 차이가 발생하는 이유를 데이터 특성과 모델 구조 측면에서 추론한다.

1. 실습 과제

과제 1. DNN 기본 코드 실행

* + - 제공된 PyTorch 코드 실행. 성능 확인
    - 아래 두 가지를 코드 실행 후 확인:
      1. **X\_train\_tensor.shape, X\_val\_tensor.shape 출력 결과 확인**
      2. **unsqueeze(1)이 필요한 이유 추측**

과제 2. DNN 성능 향상 실험

아래 항목 중 2가지 이상을 선택해 실험하고, 각 실험별 성능(Accuracy, AUC)을 비교:

* + - 중간 레이어 크기 변경 (예: 128 → 64 또는 256)
    - Dropout 확률 조정 (예: 0.3 → 0.1 또는 0.5)
    - 학습률 변경 (예: 0.001 → 0.01 또는 0.0005)
    - 에폭 수 변경 (예: 10 → 20)
    - 배치 크기 변경 (예: 64 → 32 또는 128)

과제 3. XGBoost 모델 학습 및 비교

* + - 동일한 학습/검증 데이터를 사용해 XGBoostClassifier 학습
    - 평가 지표는 Accuracy, ROC AUC
    - DNN과의 성능을 비교하고, 결과 차이를 분석
    - XGBoost가 더 좋은 경우 이유를 추측

1. 제출 안내  
   다음 항목을 제출합니다.  
   (1) 실습 코드 파일 (.py 또는 .ipynb 형식)  
   (2) 간단한 결과 보고서 (.hwp, .pdf, .docx 중 택 1)  
   보고서에는 다음 내용을 포함해야 합니다.

과제1. 기본 코드 실행 결과. Shape 확인 결과. Unsqueeze필요한 이유 서술

과제2. 모델 변경시도하면서 얻게 되는 성능

과제3. XGBoost와의 성능비교

정리. ML 이 DNN보다 좋다면 그 이유를 추측하여 기술.