

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
➤Intro	•학습열기	<ul style="list-style-type: none">인간의 뇌를 흉내 낸 인공지능, 심장을 진단하다 <p>여러분은 혹시 병원에서 인공지능이 여러분의 건강 상태를 진단해주는 장면을 상상해보신 적 있으신가요? 요즘 일부 병원에서는 실제로 AI 모델이 심장병, 당뇨병, 암과 같은 질환의 가능성을 예측하는 데 활용되고 있습니다. 이 모델의 핵심에는 바로 사람의 뇌를 본떠 만든 인공 신경망(Artificial Neural Network) 기술이 있습니다. 특히 여러 층을 가진 DNN(Deep Neural Network)은 복잡한 건강 정보를 분석하고 환자의 상태를 예측하는 데 뛰어난 성능을 발휘합니다. 예를 들어, 환자의 나이, 혈압, 콜레스테롤 수치 같은 수십 개의 특성을 바탕으로 "이 사람이 심장병일 확률이 얼마나 될까?"를 계산하죠. 우리는 지금부터 PyTorch를 이용해 이런 DNN 모델을 직접 구현하고, 학습시키고, 평가해보는 여정을 시작합니다. 단순한 이론이 아닌 실제 건강 예측 모델을 만들어본다고 생각하면, 훨씬 더 의미 있고 흥미롭지 않으신가요?</p>			① 본 학습 내용으로 들어가기 전, 학습 주제의 흥미를 이끌 만한 도입부의 내용이 있다면 제시해주세요.
	•학습목표				② ex. 관련 뉴스기사, 실생활과 관련된 이야기 등
	➤ 학습하기				③ 저작권 침해가 되지 않도록 내용을 구성해 주세요.
	1. DNN 구현				④ 출처가 있을 경우 반드시 작성해 주세요.
	2. DNN 학습				
3. DNN 평가					
➤ 적용하기					용어설명
➤ Outro					
•문제풀기					
내레이션					
3					

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	5	화면설명
➤Intro •학습열기 •학습목표 ➤학습하기 1. DNN 구현 2. DNN 학습 3. DNN 평가 ➤적용하기 ➤Outro •문제풀기	<div>◆ 학습목표</div> <div>1. PyTorch에서 심층 신경망을 구현할 수 있다.</div> <div>2. 손실 함수와 옵티마이저를 적용하여 모델을 학습할 수 있다.</div> <div>3. 모델의 성능을 검증하고 평가 지표를 적용할 수 있다.</div> <div>◆ 학습내용</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div>					① 학습내용과 학습목표는 강의계획서와 일치해야 하며, 필요시 강의계획서를 수정할 수 있습니다.
						② 학습목표
						③ 학습내용
<div>용어설명</div>						
내레이션	4					

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div><div>간지</div><div>DNN 구현</div></div>					
내레이션						5

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 심장별 판별 DNN 구현 절차</div> <div># 라이브러리 가져 오기</div> <div># 1. 지난 수업에서 만든 심장병 데이터셋 파일 불러오기</div> <div>원-핫 인코딩, 표준 정규화 전처리, 23개 속성, target이 결과 값</div> <div>heart_disease_zscore.csv</div> <div># 2. 특성과 타겟 분리</div> <div># 3. 훈련/테스트 데이터 분할</div> <div># 4. Dataset 클래스 정의</div> <div># 5. 모델 정의</div> <div># 6. 손실함수와 옵티마이저</div> <div># 7. 학습 루프</div> <div># 8. 평가</div> <div># 9. 결과 출력</div>			
		용어설명		

내
레
이
션

6

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div> <div>➤Intro</div> <div> <div>•학습열기</div> <div>•학습목표</div> </div> <div>➤학습하기</div> <div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> </div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div> </div>	<div> <div> <div>• 지난 수업에서 만든 심장병 데이터셋 파일 불러오기</div> <div>파일을 드래그 드롭으로 colab에 업로드</div> <div>다음으로 전처리한 파일 사용: heart_disease_zscore.csv</div> <div>원-핫 인코딩</div> <div>명목형 데이터(nominal data)는 이름만 있고 순서나 크기 개념이 없는 범주형 데이터</div> <div>표준 정규화</div> <div>연속형 수 값 데이터</div> <div> <div># 라이브러리 가져 오기</div> <div>import pandas as pd</div> <div>import numpy as np</div> <div>import torch</div> <div>import torch.nn as nn</div> <div>import torch.optim as optim</div> </div> <div> <div># 1. 지난 수업에서 만든 심장병 데이터셋 파일 불러오기</div> <div>df = pd.read_csv("heart_disease_zscore.csv")</div> </div> </div> </div>				
					용어설명
내레이션					7

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 특성과 타겟 분리, 훈련 · 테스트 데이터 분할</div> <div>‘target’ 필드 제외한 내용을 X, target은 y로 분리, 모두 float32 형태로 변환</div> <div>• 8:2 비율으로 train/test 데이터 분할</div> <div>함수 train_test_split()의 패러미터 stratify=y</div> <div>훈련/테스트 데이터셋에서 타겟 클래스(0/1)의 비율을 원본과 동일하게 유지하도록 보장</div> <div>→ 불균형 데이터에서도 신뢰도 높은 평가 가능</div> <div># 2. 특성과 타겟 분리</div> <div>X = df.drop(columns=["target"]).astype(np.float32)</div> <div>y = df["target"].astype(np.float32)</div> <div># 3. 훈련/테스트 분할</div> <div>from sklearn.model_selection import train_test_split</div> <div>X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=0.2, random_state=42, stratify=y)</div>			
				용어설명
내레이션	8			

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div> <div>• 8:2 비율으로 train/test 데이터 분할</div> <div>test_size=0.2: 전체 데이터의 20%를 테스트 세트로 분할</div> <div>stratify=y: 타겟 비율을 유지하며 분할 (클래스 불균형 방지)</div> </div> <div> <p>BEFORE: 원본 데이터</p> <p>X (Features)</p> <p>y (Labels)</p> <p>AFTER: 분할된 데이터</p> <p>Training Set (80%)</p> <p>X_train</p> <p>y_train</p> <p>Test Set (20%)</p> <p>X_test</p> <p>y_test</p> <p>강환수, 직접 그림</p> </div> <div> <div>50</div> <div>전체 샘플 수</div> <div>40</div> <div>훈련 세트 샘플</div> <div>10</div> <div>테스트 세트 샘플</div> <div>50.0%</div> <div>클래스 0 비율</div> <div>50.0%</div> <div>클래스 1 비율</div> </div>			용어설명

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
과정명	➤Intro	<ul style="list-style-type: none">Dataset 클래스 정의 <p>PyTorch는 데이터를 순차적으로 다루는 데 특화된 Dataset과 DataLoader 시스템을 사용</p> <p>HeartDataset: Dataset을 상속해 사용자 정의 데이터셋 클래스를 생성함</p> <p>Pandas 기반의 데이터프레임을 PyTorch 학습에 적합한 Tensor 형태로 변환</p> <p>__init__: 입력 데이터와 타겟을 float32 텐서로 변환, y는 2D로 reshape</p> <p>__getitem__: 주어진 인덱스의 (X, y) 한 쌍을 반환 → DataLoader가 내부에서 사용</p> <p># 4. Dataset 클래스 정의</p> <pre>from torch.utils.data import Dataset, DataLoader</pre> <pre>class HeartDataset(Dataset): def __init__(self, X, y): self.X = torch.tensor(X, dtype=torch.float32) self.y = torch.tensor(y, dtype=torch.float32).view(-1, 1) def __len__(self): return len(self.y) def __getitem__(self, idx): return self.X[idx], self.y[idx]</pre>			
	•학습열기				
	•학습목표				
➤학습하기	1.DNN 구현				
	2.DNN 학습				
	3.DNN 평가				
➤적용하기					용어설명
➤Outro					
•문제풀기					
내레이션	10				

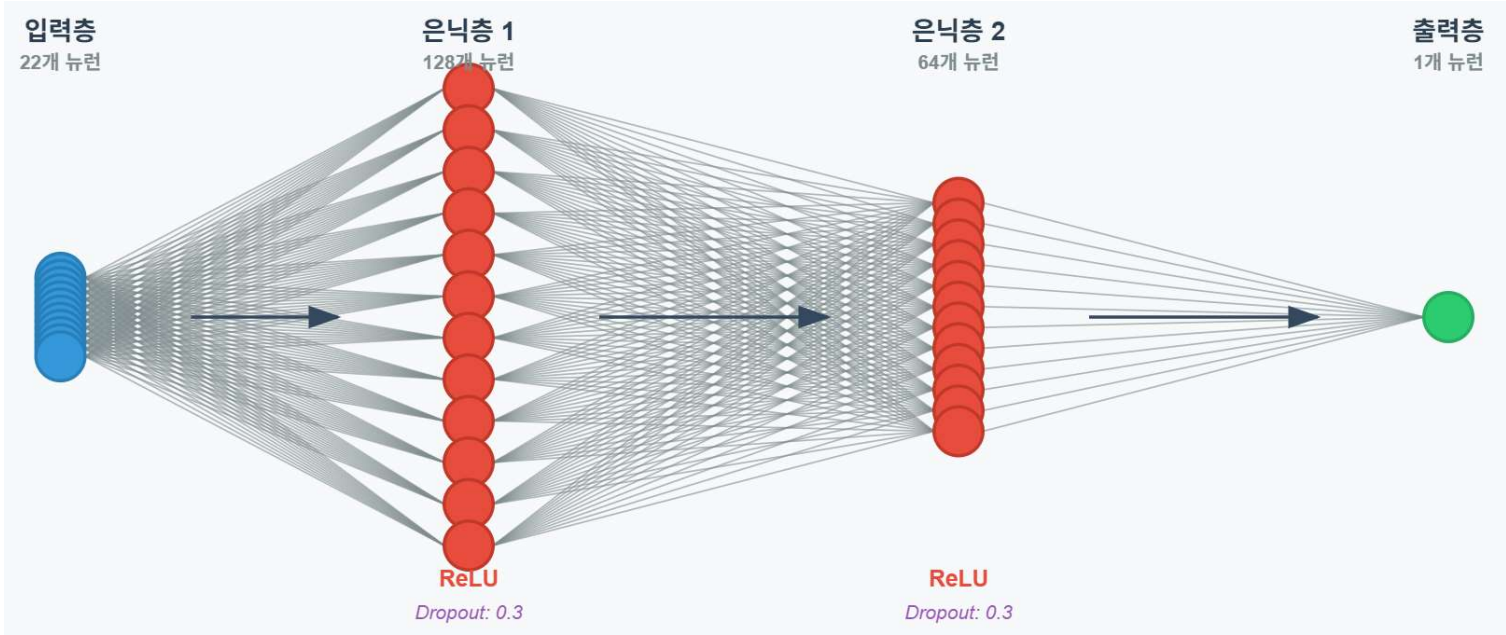
과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
과정명	➤Intro	• Dataset 클래스로 학습과 테스트 데이터 객체 생성 train_dataset, test_dataset: 학습/테스트용 Dataset 객체 생성 train_loader, test_loader: 배치 단위로 데이터를 공급하는 반복 가능한 DataLoader 객체 batch_size=32: 32개씩 묶어서 모델에 전달 shuffle=True: epoch마다 순서를 랜덤하게 섞음 → 일반화 성능에 도움 shuffle=False: 테스트는 재현성을 위해 순서를 섞지 않음 # 4. Dataset 클래스 정의 from torch.utils.data import Dataset, DataLoader class HeartDataset(Dataset): def __init__(self, X, y): self.X = torch.tensor(X, dtype=torch.float32) self.y = torch.tensor(y, dtype=torch.float32).view(-1, 1) ... train_dataset = HeartDataset(X_train, y_train) test_dataset = HeartDataset(X_test, y_test) train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True) test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)			
	•학습열기				
•학습목표					
➤학습하기					
1.DNN 구현					
2.DNN 학습					
3.DNN 평가					
➤적용하기					
➤Outro					
•문제풀기					

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
➤Intro •학습열기 •학습목표 ➤학습하기 1.DNN 구현 2.DNN 학습 3.DNN 평가 ➤적용하기 ➤Outro •문제풀기	• DNN 모델 정의 입력 층(22개 특성), 은닉 층 2개(128, 64), 출력 층(1) # 5. 모델 정의 class DNN(nn.Module): def __init__(self, input_dim): super(DNN, self).__init__() self.net = nn.Sequential(nn.Linear(input_dim, 128), nn.ReLU(), nn.Dropout(0.3), nn.Linear(128, 64), nn.ReLU(), nn.Dropout(0.3), nn.Linear(64, 1) # Sigmoid 제거) def forward(self, x): return self.net(x) model = DNN(input_dim=X.shape[1])				
					용어설명

내레이션

12

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
과정명	➤Intro	• 층 사이에 다음을 Dropout: 과적합(overfitting) 방지 학습 시 일부 뉴런을 랜덤하게 비활성화하는 방법 특정 뉴런에 과도하게 의존하는 현상을 방지 테스트 시에는 모든 뉴런을 사용함 ReLU (Rectified Linear Unit): 비선형성 부여, 역전파 가능 음수 입력을 0으로, 양수는 그대로 출력하는 비선형 함수 비선형성을 부여해 신경망이 복잡한 함수도 학습 가능하도록 함 계산도 빠르고 기울기 손실(gradient vanishing) 문제가 적어 자주 사용됨			
	•학습열기				
	•학습목표				
➤학습하기	1. DNN 구현				
	2. DNN 학습				
	3. DNN 평가				
➤적용하기					용어설명
➤Outro					
•문제풀기					
내레이션	13				

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• DNN 모델 정의</div> <div>입력 층(22개 특성), 은닉층 2개(128, 64), 출력 층(1)</div> <div></div>			
	<div>강환수, 직접 그림</div>			
				<div>용어설명</div>

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div><div>간지</div><div>DNN 학습</div></div>					
내레이션						15

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명			
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 손실함수와 옵티마이저</div> <div>nn.BCEWithLogitsLoss(): 이진 분류를 위한 가장 안전하고 권장되는 손실 함수</div> <div>이진 분류(binary classification) 문제에 사용되는 손실 함수</div> <div>Sigmoid 연산과 Binary Cross Entropy를 한 번에 계산함</div> <div>마지막 출력층에 Sigmoid를 따로 쓰지 않아도 됨</div> <div>optim.SGD()</div> <div>확률적 경사 하강법(전통적인 옵티마이저)</div> <div>lr=0.01: 초기 학습률 설정. 너무 크면 발산, 너무 작으면 학습 속도 저하</div> <div>optim.lr_scheduler.CosineAnnealingLR()</div> <div>학습률을 코사인 곡선을 따라 점점 줄여줌</div> <div>T_max=10이면, 10 에폭(epoch) 동안 학습률이 최솟값에 도달</div> <div># 6. 손실함수와 옵티마이저</div> <div>crit_{erion} = nn.BCEWithLogitsLoss()</div> <div>optim_{izer} = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)</div> <div>sched_{uler} = optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=10)</div>						
							용어설명

내레이션

16

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 학습</div> <div>학습을 50 epoch 동안 반복</div> <div>매 epoch마다 모델을 학습 모드로 전환</div> <div>미니배치 단위로 forward, loss 계산, backward, optimizer step 수행</div> <div>scheduler.step()으로 학습률 조정</div> <div>10 에폭마다 평균 손실값을 출력하여 과적합 여부나 학습 상태를 점검</div> <div><div># 7. 학습 루프</div><div>epochs = 50</div><div>Epoch 10 Loss: 0.3402</div><div>for epoch in range(epochs):</div><div>Epoch 20 Loss: 0.2827</div><div>model.train()</div><div>Epoch 30 Loss: 0.3018</div><div>running_loss = 0</div><div>Epoch 40 Loss: 0.2551</div><div>for X_batch, y_batch in train_loader:</div><div>Epoch 50 Loss: 0.2449</div><div>optimizer.zero_grad()</div><div>outputs = model(X_batch)</div><div>loss = criterion(outputs, y_batch)</div><div>loss.backward()</div><div>optimizer.step()</div><div>running_loss += loss.item()</div><div> </div><div>scheduler.step()</div><div>avg_loss = running_loss / len(train_loader)</div><div>if (epoch + 1) % 10 == 0:</div><div>print(f"Epoch {epoch+1:2d} Loss: {avg_loss:.4f}")</div></div>				

용어설명

내레이션

17

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	5	화면설명	
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div>		<div><div>간지</div><div>DNN 평가</div></div>					
						용어설명	
		<div>➤적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>					
내레이션							18

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
과정명	➤Intro	•학습열기 model.eval() 평가 모드로 전환 Dropout 등이 평가 시 동작에 맞게 작동하도록 함 (예: Dropout 비활성화) y_pred, y_true: 예측값과 실제 정답 레이블을 저장할 리스트 초기화 with torch.no_grad(): 그라디언트 계산 비활성화, 메모리 사용 줄이고 연산 속도 향상 (평가 시 필수) # 8. 평가 model.eval() y_pred = [] y_true = [] with torch.no_grad(): for X_batch, y_batch in test_loader: outputs = model(X_batch) probs = torch.sigmoid(outputs) preds = (probs > 0.5).float() y_pred.extend(preds.cpu().numpy()) y_true.extend(y_batch.cpu().numpy())			
	•학습목표				
	➤ 학습하기				
	1. DNN 구현				
	2. DNN 학습				
	3. DNN 평가				
	➤적용하기				용어설명
	➤Outro				
	•문제풀기				
내레이션					

19

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div>	<div>• 평가 루프 내부</div> <div>테스트 데이터를 모델에 넣어 예측값(logits)을 얻음</div> <div>출력값에 sigmoid 적용해 확률로 변환</div> <div>0.5 초과면 클래스 1, 아니면 0으로 이진 분류 결과 결정</div> <div>예측 결과를 리스트에 추가 (CPU로 이동해 numpy 배열로 변환)</div> <div>실제 정답 레이블도 리스트에 추가</div> <div># 8. 평가</div> <div>model.eval()</div> <div>y_pred = []</div> <div>y_true = []</div> <div>with torch.no_grad():</div> <div>for X_batch, y_batch in test_loader:</div> <div>outputs = model(X_batch)</div> <div>probs = torch.sigmoid(outputs)</div> <div>preds = (probs > 0.5).float()</div> <div>y_pred.extend(preds.cpu().numpy())</div> <div>y_true.extend(y_batch.cpu().numpy())</div>				용어설명
		내레이션	20		

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
➤Intro •학습열기 •학습목표 ➤학습하기 1.DNN 구현 2.DNN 학습 3.DNN 평가 ➤적용하기 ➤Outro •문제풀기	• 결과 출력 Accuracy (정확도): 전체 예측 중 정답 비율 Precision (정밀도): 모델이 1이라고 예측한 것들 중 실제 1인 비율 Recall (재현율): 실제 1인 것들 중 모델이 1이라고 예측한 비율 F1 Score: Precision과 Recall의 조화 평균 (둘 사이 균형 평가) # 9. 결과 출력 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score print("=== Evaluation Metrics ===") print(f"Accuracy : {accuracy_score(y_true, y_pred):.4f}") print(f"Precision: {precision_score(y_true, y_pred, zero_division=0):.4f}") print(f"Recall : {recall_score(y_true, y_pred, zero_division=0):.4f}") print(f"F1 Score : {f1_score(y_true, y_pred, zero_division=0):.4f}") === Evaluation Metrics === Accuracy : 0.8852 Precision: 0.8182 Recall : 0.9643 F1 Score : 0.8852			
				용어설명

내레이션

21

2

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명																			
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div>	<div>• 혼돈 행렬(confusion matrix)</div> <div>분류 모델의 예측 결과를 요약해 주는 2차원 표</div> <div>주로 이진 분류(binary classification) 문제에서 사용</div> <div>실제 값과 예측 값이 일치하는지 아닌지를 구체적으로 표시</div> <div>• TP, TN, FP, FN 용어 이해(True, False, Positive, Negative)</div> <div>앞의 True/False: 예측이 맞음/틀림</div> <div>뒤의 Pos/Neg: 예측을 양성(1)/음성(0)</div> <table><thead><tr><th>이름</th><th>실제</th><th>예측</th><th>해석</th></tr></thead><tbody><tr><td>True Positive</td><td>병 있음 (1)</td><td>병 있음 (1)</td><td>✓ 잘 맞췄음</td></tr><tr><td>False Positive</td><td>정상 (0)</td><td>병 있음 (1)</td><td>✗ 정상이 병으로 나옴</td></tr><tr><td>True Negative</td><td>정상 (0)</td><td>정상 (0)</td><td>✓ 잘 맞췄음</td></tr><tr><td>False Negative</td><td>병 있음 (1)</td><td>정상 (0)</td><td>✗ 병인데 놓침 (진짜 문제)</td></tr></tbody></table>				이름	실제	예측	해석	True Positive	병 있음 (1)	병 있음 (1)	✓ 잘 맞췄음	False Positive	정상 (0)	병 있음 (1)	✗ 정상이 병으로 나옴	True Negative	정상 (0)	정상 (0)	✓ 잘 맞췄음	False Negative	병 있음 (1)	정상 (0)	✗ 병인데 놓침 (진짜 문제)
	이름	실제	예측	해석																				
True Positive	병 있음 (1)	병 있음 (1)	✓ 잘 맞췄음																					
False Positive	정상 (0)	병 있음 (1)	✗ 정상이 병으로 나옴																					
True Negative	정상 (0)	정상 (0)	✓ 잘 맞췄음																					
False Negative	병 있음 (1)	정상 (0)	✗ 병인데 놓침 (진짜 문제)																					
<div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>용어설명</div>																							

내레이션을

22

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명										
과정명	<div>➤Intro<ul style="list-style-type: none">•학습열기•학습목표</div> <div>➤ 학습하기<ul style="list-style-type: none">1. DNN 구현2. DNN 학습3. DNN 평가</div> <div>➤ 적용하기</div> <div>➤ Outro<ul style="list-style-type: none">•문제풀기</div>			<div><ul style="list-style-type: none">• 혼돈 행렬의 이해<ul style="list-style-type: none">단순한 평가 표가 아니라 모델이 어떤 실수를 하고 있는지 파악하게 해주는 해부도• 병원에서 의사가 환자를 검사한다고 가정<ul style="list-style-type: none">TP: 환자로 맞게 예측(환자가 아프고, 의사도 아프다고 함) → 잘했어!TN: 정상으로 맞게 예측(환자가 건강하고, 의사도 건강하다고 함) → 잘했어!FP: 환자로 잘못 예측(환자는 건강한데, 의사가 병이라고 함) → 과잉진단 (검사비 낭비)FN: 정상으로 잘못 예측(환자는 아픈데, 의사가 괜찮다고 함) → 최악(목숨이 위험해질 수 있음)</div> <div><div>가로는 예측 값</div><table><tr><td></td><td>Negative (0)</td><td>Positive (1)</td></tr><tr><td>Negative (0)</td><td>TN = 27</td><td>FP = 6</td></tr><tr><td>Positive (1)</td><td>FN = 1</td><td>TP = 27</td></tr></table><div>세로는 실제 값 0: 정상인 1: 심장병(환자)</div></div>		Negative (0)	Positive (1)	Negative (0)	TN = 27	FP = 6	Positive (1)	FN = 1	TP = 27	용어설명
						Negative (0)	Positive (1)							
Negative (0)	TN = 27	FP = 6												
Positive (1)	FN = 1	TP = 27												
내레이션	23													

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 혼돈 행렬과 평가 지표</div> <div>Accuracy (정확도): 전체 예측 중 정답 비율</div> <div>계산식: $(TP + TN) / (TP + TN + FP + FN)$</div> <div>Precision (정밀도): 모델이 1(환자)이라고 예측한 것들 중 실제 1인 비율</div> <div>계산식: $TP / (TP + FP)$</div> <div>FP가 많으면 precision은 낮아짐 (거짓 양성 조심)</div> <div>Recall (재현율): 실제 1(환자)인 것들 중 모델이 1이라고 예측한 비율</div> <div>계산식: $TP / (TP + FN)$</div> <div>FN이 많으면 recall은 낮아짐 (거짓 음성 조심)</div> <div>심장병(또는 암환자)처럼 중병 판별에서는 recall이 매우 중요</div> <div>F1 Score: Precision과 Recall의 조화 평균 (둘 사이 균형 평가)</div> <div>계산식: $2 * (Precision * Recall) / (Precision + Recall)$</div>			<div>Confusion Matrix + Metrics</div> <div>Negative (0)Positive (1)</div> <div>True Label</div> <div>Negative (0)</div> <div>Positive (1)</div> <div>Predicted Label</div> <div>25</div> <div>20</div> <div>15</div> <div>10</div> <div>5</div> <div>Accuracy = $(TP+TN)/(TP+TN+FP+FN) = (27+28)/(27+28+5+1) = 0.9016$</div> <div>Precision = $TP/(TP+FP) = 27/(27+5) = 0.8438$</div> <div>Recall = $TP/(TP+FN) = 27/(27+1) = 0.9643$</div> <div>F1 Score = $2*P*R/(P+R) = 2*0.8438*0.9643/(0.8438+0.9643) = 0.9000$</div>

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명															
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div>	<div>• 혼돈 행렬과 평가 지표</div> <table><tr><th>지표</th><th>무엇을 측정하나?</th><th>중요할 때 예시</th></tr><tr><td>Accuracy(정확도)</td><td>전체 중 맞춘 비율</td><td>일반적인 분류 상황</td></tr><tr><td>Precision(정밀도)</td><td>양성(환자) 예측의 정확도</td><td>거짓 양성을 피하고 싶을 때</td></tr><tr><td>Recall(재현율)</td><td>양성(환자)을 놓치지 않는 능력</td><td>놓치면 위험한 경우</td></tr><tr><td>F1 Score(조화평균)</td><td>정밀도/재현율 균형 종합값</td><td>두 지표가 모두 중요할 때</td></tr></table>				지표	무엇을 측정하나?	중요할 때 예시	Accuracy(정확도)	전체 중 맞춘 비율	일반적인 분류 상황	Precision(정밀도)	양성(환자) 예측의 정확도	거짓 양성을 피하고 싶을 때	Recall(재현율)	양성(환자)을 놓치지 않는 능력	놓치면 위험한 경우	F1 Score(조화평균)	정밀도/재현율 균형 종합값	두 지표가 모두 중요할 때	
					지표	무엇을 측정하나?	중요할 때 예시													
					Accuracy(정확도)	전체 중 맞춘 비율	일반적인 분류 상황													
					Precision(정밀도)	양성(환자) 예측의 정확도	거짓 양성을 피하고 싶을 때													
					Recall(재현율)	양성(환자)을 놓치지 않는 능력	놓치면 위험한 경우													
F1 Score(조화평균)	정밀도/재현율 균형 종합값	두 지표가 모두 중요할 때																		
				용어설명																

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명									
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 구현</div> <div>2.DNN 학습</div> <div>3.DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 심장별 판별 DNN 결과와 혼돈 행렬</div> <div>=== Evaluation Metrics ===</div> <div>Accuracy : 0.8852</div> <div>Precision: 0.8182</div> <div>Recall : 0.9643</div> <div>F1 Score : 0.8852</div> <div>현재 값: TP = 27, TN = 27, FP = 6, FN = 1</div> <div>혼돈행렬</div> <table><tr><td></td><td>예측 Negative (0)</td><td>예측 Positive (1)</td></tr><tr><td>실제 Negative (0)</td><td>TN 27</td><td>FP 6</td></tr><tr><td>실제 Positive (1)</td><td>FN 1</td><td>TP 27</td></tr></table> <div>강환수, 직접 그림</div>				예측 Negative (0)	예측 Positive (1)	실제 Negative (0)	TN 27	FP 6	실제 Positive (1)	FN 1	TP 27	<div>🎯 Accuracy (정확도)</div> <div>(TP + TN) / (TP + TN + FP + FN)</div> <div>88.5%</div> <div>전체 예측 중 정답인 비율</div> <div>🔍 Precision (정밀도)</div> <div>TP / (TP + FP)</div> <div>81.8%</div> <div>Positive로 예측한 것 중 실제 Positive인 비율 (False Positive 주의)</div> <div>📊 Recall (재현율)</div> <div>TP / (TP + FN)</div> <div>96.4%</div> <div>실제 Positive 중 올바르게 예측한 비율 (False Negative 주의)</div> <div>🏆 F1 Score</div> <div>2 × (Precision × Recall) / (Precision + Recall)</div> <div>88.5%</div> <div>Precision과 Recall의 조화평균 (균형적 평가)</div>
		예측 Negative (0)	예측 Positive (1)										
실제 Negative (0)	TN 27	FP 6											
실제 Positive (1)	FN 1	TP 27											
용어설명													

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div>	<div>• 혼돈 행렬 그림과 평가 지표 결과를 함께 그림</div> <div><div>Confusion Matrix + Metrics</div><div><div>Negative (0)</div><div>Positive (1)</div></div><div><div>True Label</div><div>Negative (0)</div><div>Positive (1)</div><div>Predicted Label</div></div><div><div>TN = 28</div><div>FP = 5</div><div>FN = 1</div><div>TP = 27</div></div><div><div>25</div><div>20</div><div>15</div><div>10</div><div>5</div></div></div> <div><div>Accuracy = (TP+TN)/(TP+TN+FP+FN) = (27+28)/(27+28+5+1) = 0.9016</div><div>Precision = TP/(TP+FP) = 27/(27+5) = 0.8438</div><div>Recall = TP/(TP+FN) = 27/(27+1) = 0.9643</div><div>F1 Score = 2*P*R/(P+R) = 2*0.8438*0.9643/(0.8438+0.9643) = 0.9000</div></div> <div>강환수, 직접 그림</div>				
내레이션					용어설명

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	5	화면설명		
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 구현</div> <div>2. DNN 학습</div> <div>3. DNN 평가</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>		<div>• "심장병 진단과 같은 이진 분류 문제에서 DNN 모델을 적용할 경우, 모델의 성능을 평가하기 위한 지표로 Accuracy, Precision, Recall, F1 Score 중 어떤 것을 가장 중요하게 고려해야 하는지 설명하고, 그 이유를 실제 의료적 상황과 연계하여 서술해보세요.</div> <div>• 답안사례</div> <div>의료 진단에서 가장 중요한 평가 지표는 Recall(재현율)이라고 할 수 있습니다. 이는 실제로 심장병 환자인 사람들 중에서 모델이 얼마나 잘 찾아냈는지를 나타내며, 환자를 놓치지 않는 것이 중요한 의료 상황에서는 필수적인 지표입니다. 예를 들어, Recall이 낮다는 것은 심장병 환자를 '정상'으로 잘못 판단할 수 있는 가능성이 높다는 뜻이며, 이는 생명과 직결된 중대한 문제입니다.</div>			<div>① 학습 내용과 관련하여 실제 적용력을 높일 수 있는 문제, 혹은 주제를 작성해 주세요.</div> <div>② ex. 사례 제시 후 전문가 의견, 실습과제, 응용 예시 시뮬레이션 등</div> <div>③ 저작권 침해가 되지 않도록 내용을 구성해 주세요.</div> <div>④ 출처가 있을 경우 반드시 작성해 주세요.</div>		
					용어설명		

내레이션

28