


(인공지능과빅데이터) 연습문제_8

1. 머신러닝 데이터셋에서 라벨 불균형(class imbalance)이 발생했을 때, 모델 학습 및 평가에 미치는 영향으로 가장 **거리가 먼** 것은 무엇인가?
 1. 소수 클래스(minority class)의 예측 성능이 저하될 가능성이 높으며, 다수 클래스(majority class) 위주로 학습될 경향이 있다.
 2. 정확도(accuracy)와 같이 다수 클래스의 예측 성능을 위주로 평가하는 지표는 모델의 실제 성능을 과대평가할 수 있다.
 3. 모델이 모든 데이터를 다수 클래스로 예측하더라도 높은 정확도를 보일 수 있어, 모델의 판별 능력을 제대로 평가하기 어렵다.
 4. 과적합(overfitting) 문제가 소수 클래스에서 더욱 두드러질 수 있으며, 일반화 성능을 저해할 수 있다.
 5. 라벨 불균형은 모델 학습 속도를 증가시키고, 학습에 필요한 컴퓨팅 자원 요구량을 크게 늘리는 주요 원인이 된다. 
2. 데이터 라벨 클래스의 불균형을 해소하기 위한 언더샘플링(undersampling) 방법에 대한 설명으로 가장 **부적절한** 것은 무엇가?
 1. 언더샘플링은 다수 클래스(majority class)의 데이터 샘플 수를 줄여 소수 클래스(minority class)의 비율과 균형을 맞추는 기법.
 2. 무작위 언더샘플링(random undersampling)은 다수 클래스의 샘플을 무작위로 제거하여 데이터 불균형을 완화하는 가장 기본적인 방법이다.
 3. 언더샘플링은 다수 클래스의 정보를 손실시킬 수 있으며, 이는 모델 학습에 부정적인 영향을 미칠 수 있다는 단점이 있다.
 4. Tomek Links 제거 방법은 서로 가장 가까운 반대 클래스 쌍을 찾아 다수 클래스 샘플을 제거함으로써 클래스 간의 결정 경계를 명확하게 하는데 도움을 준다.
 5. 언더샘플링은 항상 원본 데이터의 분포를 최대한 보존하면서 다수 클래스의 샘플 수를 줄이는 것을 목표로 한다. 
3. 데이터 라벨 클래스의 불균형을 해소하기 위한 오버샘플링(oversampling) 방법에 대한 설명으로 가장 **적절하지 않은** 것은 무엇인가?
 1. 오버샘플링은 소수 클래스(minority class)의 데이터 샘플 수를 늘려 다수 클래스(majority class)와의 비율을 균형 있게 만드는 기법이다.
 2. 무작위 오버샘플링(random oversampling)은 소수 클래스의 기존 샘플을 단순 복제하여 데이터 수를 늘리는 기본적인 방법이다.
 3. SMOTE(Synthetic Minority Over-sampling Technique)는 소수 클래스의 주변 샘플들을 기반으로 새로운 가상의 소수 클래스 샘플들을


생성하는 방식이다.

4. 오버샘플링은 소수 클래스의 과적합(overfitting) 위험을 증가시킬 수 있으며, 특히 단순 복제 방식은 모델의 일반화 성능을 저해할 수 있다.
5. 오버샘플링은 다수 클래스의 정보를 손실시키지 않으면서 클래스 간의 균형을 맞출 수 있다는 장점이 있다. ☒ 오버샘플링은 다수 클래스의 정보 내용 직접적인 관련이 없다.
4. 데이터 라벨 클래스 불균형을 해소하기 위해 사용되는 SMOTE(Synthetic Minority Over-sampling Technique) 기법에 대한 설명으로 가장 **정확하지 않은** 것은 무엇인가?
 1. SMOTE는 소수 클래스의 각 샘플에 대해 k개의 최근접 이웃(k-Nearest Neighbors)을 찾고, 이웃들과의 특징 공간상에서 선형 보간(linear interpolation)을 통해 새로운 가상 소수 클래스 샘플을 생성.
 2. SMOTE는 단순 복제 방식의 오버샘플링이 갖는 과적합(overfitting) 문제를 완화하고, 소수 클래스의 결정 경계를 더 넓고 일반화된 형태로 학습하는 데 도움을 줄 수 있다.
 3. SMOTE로 생성된 가상 샘플들은 항상 원본 소수 클래스 샘플들과 정확히 동일한 특징 값을 가지므로, 데이터의 다양성을 증가시키는 효과는 미미하다. ☒
 4. SMOTE를 적용할 때, k 값의 선택은 생성되는 가상 샘플들의 특성에 영향을 미치며, 적절한 k 값을 찾는 것이 중요하다.
 5. SMOTE는 수치형(numerical) 특징을 갖는 데이터에 주로 적용 가능하며, 범주형(categorical) 특징을 직접적으로 처리하기 위해서는 별도의 전처리 과정이 필요하다.
5. 파이썬의 사이킷런(scikit-learn) 패키지 외에 불균형 클래스 데이터 처리를 위한 별도의 특화된 패키지 및 그 예시로 가장 **부적절한** 것은 무엇인가?
 1. 사이킷런의 resample 함수는 기본적인 오버샘플링 및 언더샘플링 기능을 제공하지만, 더 복잡한 불균형 데이터 처리 기법은 지원하지 않는다.
 2. imbalanced-learn 패키지는 SMOTE, ADASYN과 같은 다양한 오버샘플링 기법과 Tomek Links, Edited Nearest Neighbors와 같은 언더샘플링 기법을 포함하고 있다.
 3. imblearn이라는 이름으로도 자주 임포트되는 imbalanced-learn 패키지는 파이썬의 머신러닝 생태계에서 불균형 데이터 처리를 위한 사실상의 표준 라이브러리이다.
 4. pandas 패키지는 데이터프레임 형태의 데이터를 효율적으로 다루는 데 유용하지만, 고급 불균형 데이터 처리 기능을 직접적으로 제공하지는 않는다.
 5. numpy 패키지는 배열 기반의 수치 연산을 위한 핵심 라이브러리이며, 불균형 데이터 처리 알고리즘의 기반이며, 고수준의 불균형 해소 기능을 내장하고 있다. ☒
6. 범주형(categorical) 데이터를 머신러닝 모델이 이해할 수 있는 숫자 형식으로 변환하는 One-Hot Encoding과 사이킷런(scikit-learn)을 이용한 One-Hot Encoding 사용 예시로 가장 **부적절한** 것은 무엇인가?

1. One-Hot Encoding은 각 범주형 변수의 고유한 값들을 새로운 이진 (binary) 특성(feature)으로 만들고, 각 데이터 포인트에 대해 해당하는 범주에 1, 그렇지 않은 범주에 0을 할당하는 방식이다.
 2. 사이킷런의 OneHotEncoder 클래스를 사용하면 손쉽게 One-Hot Encoding을 수행할 수 있으며, fit() 메서드로 데이터를 학습하고 transform() 메서드로 실제 변환을 적용한다.
 3. OneHotEncoder를 적용할 때, handle_unknown='ignore' 옵션을 사용하면 학습 데이터에 없던 새로운 범주 값이 등장하더라도 오류를 발생시키지 않고 모든 0으로 인코딩한다.
 4. One-Hot Encoding은 트리 기반 모델(예: 결정 트리, 랜덤 포레스트)과 같이 범주형 데이터를 직접 처리할 수 있는 모델에는 절대로 적용하지 않는다. ☒ 적용하지 않을 수 있지만 절대로 안되는 것은 아니다.
 5. One-Hot Encoding은 범주형 변수의 순서 정보가 중요하지 않거나 명목형(nominal) 변수를 다룰 때 유용하며, 순서형(ordinal) 변수의 경우에는 순서 정보를 보존하는 다른 인코딩 방식이 더 적합할 수 있다.
7. 텐서보드(TensorBoard)에 대한 설명으로 가장 **부적절한** 것을 고르시오?
1. 텐서보드는 텐서플로(TensorFlow) 및 기타 머신러닝 실험 과정을 시각화하고 이해하기 위한 웹 기반의 시각화 도구.
 2. 학습 과정에서의 손실(loss), 정확도(accuracy), 가중치(weights), 활성화 함수(activations) 등의 주요 지표 변화를 실시간으로 모니터링할 수 있도록 지원.
 3. 텐서보드는 스칼라 값, 이미지, 오디오, 텍스트, 그래프 구조, 히스토그램, 임베딩 프로젝터 등 다양한 형태의 데이터를 시각화하는 기능을 제공.
 4. 텐서보드 서버를 실행하고 로그 디렉토리를 지정하면, 웹 브라우저를 통해 시각화 결과를 확인할 수 있다.
 5. 텐서보드는 파이토치(PyTorch)와 같은 다른 딥러닝 프레임워크와는 직접적으로 호환되지 않으며, 텐서플로에서만 사용 가능한 도구이다. ☒
8. 다음 중 텐서보드(TensorBoard), MLflow, Weights & Biases (WandB)와 같은 머신러닝 모델 모니터링 도구들에 대한 설명으로 가장 **부적절한** 것은 무엇인가?
1. 이 도구들은 머신러닝 실험의 진행 상황을 추적하고, 하이퍼파라미터, 메트릭, 모델 아티팩트 등을 기록하여 실험 관리를 용이하게 한다.
 2. 텐서보드는 주로 텐서플로 생태계에서 널리 사용되며, MLflow는 다양한 프레임워크를 지원하는 오픈소스 플랫폼이고, WandB는 클라우드 기반의 상업용 플랫폼이다.
 3. MLflow는 모델의 재현성을 높이기 위해 코드, 환경, 파라미터, 결과를 함께 추적하고 관리하는 데 특화된 기능을 제공.
 4. WandB는 실시간 메트릭 시각화, 하이퍼파라미터 스위핑, 협업 기능 등을 강력하게 지원하여 팀 단위의 실험 관리에 유용.
 5. 이 도구들은 학습된 모델의 성능 비교 및 분석을 돕지만, 배포된 모델의 실시간 성능 모니터링 및 이상 감지 기능은 제공하지 않는다. ☒ 텐서보

드, MLflow, WandB는 주로 모델 학습 과정의 모니터링과 관리에 초점을 맞추고 있지만, MLflow, WandB 등은 일부 기능을 통해 배포된 모델의 기본적인 성능 모니터링을 지원하기도 한다.

9. 파이토치(PyTorch)에서 사용자 정의 데이터셋(Custom Dataset)과 데이터로더(DataLoader)를 구축하는 방법에 대한 설명 중 가장 **부적절한** 것은 무엇가?

1. 커스텀 데이터셋 클래스는 `torch.utils.data.Dataset` 클래스를 상속받아 `__len__` 메서드와 `__getitem__` 메서드를 반드시 구현해야 한다.
2. `__len__` 메서드는 데이터셋의 총 샘플 수를 반환하며, `__getitem__` 메서드는 주어진 인덱스에 해당하는 데이터 샘플(입력과 레이블)을 텐서 형태로 반환한다.
3. `torch.utils.data.DataLoader`는 커스텀 데이터셋 객체를 입력으로 받아 배치(batch) 처리, 데이터 섞기(shuffling), 병렬 로딩(parallel loading) 등의 기능을 제공하여 학습 효율성을 높인다.
4. 데이터로더를 생성할 때 `shuffle=True` 파라미터를 설정하면 매 에폭(epoch)마다 데이터를 무작위로 섞어 모델이 데이터 순서에 과적합되는 것을 방지하는 데 도움이 된다.
5. 커스텀 데이터셋의 `__getitem__` 메서드에서는 이미지 파일 로딩 및 전처리, 텍스트 데이터 토큰화 등의 복잡한 데이터 처리 로직을 직접 구현하는 것은 권장되지 않으며, 데이터 로더의 `collate_fn` 파라미터를 통해 처리하는 것이 일반적이다. 

"__getitem__" 메서드는 주어진 인덱스에 해당하는 데이터 샘플을 반환하는 역할을 수행하지만 여기에 다른 기능을 포함 시킬 수도 있다. "collate_fn"은 여러 개의 샘플을 하나의 배치로 묶는 방식을 사용자 정의할 때 사용되며, 개별 샘플 자체의 로딩 및 기본적인 전처리는 "__getitem__"에서 이루어지는 것이 일반적 이미지나 문자열고 같이 복잡한 경우 "torchvision.transforms"나 "torchtext.transforms"와 같은 별도의 모듈을 활용하여 데이터셋 클래스 내에서 처리

- dataloader에서 `collate_fn`의 사용 예 :
 - 다른 길이 등과 같은 다른 형태의 데이터 샘플을 같은 형태로 맞춰주는 작업을 정의한 메서드를 지정하는 형태임.

```
def my_collate_fn(samples):
    collate_X = []
    collate_y = []
    max_len = max([len(sample['X']) for sample in samples])

    for sample in samples:
        diff = max_len - len(sample['X'])
        if diff > 0:
            zero_pad = torch.zeros(size=(diff,))
            collate_X.append(torch.cat([sample['X'], zero_pad], dim=0))
        else:
            collate_X.append(sample['X'])
    collate_y = [sample['y'] for sample in samples]
```


```

    return {'X': torch.stack(collate_X), 'y': torch.stack(collate_y)}

dataloader_example = torch.utils.data.DataLoader(
    dataset_example,
    batch_size=2,
    collate_fn=my_collate_fn) for d in dataloader_example:
    print(d['X'], d['y'])

```

10. 텐서플로(TensorFlow)에서 사용자 정의 데이터셋을 구성하는 방법에 대한 설명 중 가장 **부적절한** 것은 무엇인가?

1. 텐서플로의 `tf.data.Dataset` API를 사용하여 다양한 소스로부터 데이터를 읽어오고 변환하는 파이프라인을 구축할 수 있다.
2. `tf.data.Dataset.from_tensor_slices()` 메서드를 사용하면 넘파이(Numpy) 배열이나 텐서(Tensor)와 같은 메모리 상의 데이터를 손쉽게 데이터셋 객체로 변환할 수 있다.
3. 텍스트 파일, 이미지 파일, TFRecord 파일 등 다양한 형식의 데이터 소스를 읽어오기 위한 특화된 함수들 (예: `tf.data.TextLineDataset`, `tf.keras.utils.image_dataset_from_directory`, `tf.data.TFRecordDataset`)을 제공한다.
4. 사용자 정의 데이터 처리 및 변환 로직은 `map()` 메서드를 사용하여 데이터셋의 각 요소에 적용할 수 있으며, 이를 통해 데이터 증강(augmentation)이나 전처리(preprocessing) 등을 수행할 수 있다.
5. 텐서플로 데이터셋은 배치 처리(`batch()`), 섞기(`shuffle()`), 반복(`repeat()`) 등의 기능을 내장하고 있으며, 이러한 기능들은 데이터셋 객체 생성 이후에는 변경할 수 없다.  생성된 이후에도 메서드 체이닝(method chaining)을 통해 배치 크기, 섞기 여부, 반복 횟수 등을 유연하게 구성하거나 변경할 수 있다