sampling, hyper-parameter tuning

# 주차	13

over-sampling and under-sampling

주로 분류 문제에서 불균형한 데이터셋을 다루기 위해 활용한다. (e.g. 이진 분류 문제에서 클래스 0의 샘플이 90%, 클래스 1의 샘플이 10%인 경우)

over-sampling

minority class 샘플 수를 늘려서 데이터 불균형을 해소한다.

1. **random over-sampling**

   minority class의 데이터를 단순히 복제하는 방법: 쉬운 방법이지만, 중복된 데이터가 많아져 모델이 overfitting될 가능성이 있다.

2. **SMOTE synthetic minority over-sampling technique**

   minority class의 데이터를 기반으로 새로운 가상 데이터를 생성하는 방법: 중복 문제를 완화하면서도 데이터 수를 늘릴 수 있다.

under-sampling

majority class 샘플 수를 줄여서 데이터 불균형을 해소한다.

1. **random under-sampling**

   majority class의 데이터를 무작위로 선택하여 삭제하는 방법: 간단하지만, 중요한 정보를 잃어버릴 가능성이 있다.

2. **NearMiss**

   majority class의 데이터 중에서 minority class와 가까운(유사한) 데이터만 선택하는 방법: 데이터의 다양성을 유지하면서도 불균형을 해결할 수 있다.

때로는 두 방법을 함께 사용하는 hybrid 접근법도 사용된다.

evaluation

```python
# sampling 적용하지 않은 경우

y_pred = model.predict(X_test)
y_pred_prob = model.predict_proba(X_test)[:, 1] # 각 클래스에 속할 확률을 계산, 그 중 클래스 1에 속할 확률을 반환

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(conf_matrix)
print(class_report)
print(accuracy)
```

Python

- `predict_proba`는 각 샘플에 대해 클래스 0과 클래스 1에 속할 확률을 반환한다.

    - `[:, 1]`이라고 했으므로, 클래스 1에 속할 확률을 추출한 것
- accuracy: 전체 샘플 중에서 정확하게 예측한 샘플의 비율
- precision: 1이라고 예측한 샘플 중에서 진짜 1인 샘플의 비율
- f1 score: accuracy와 precision의 조화평균으로, 두 성능 지표 간의 균형을 나타냄
- ROC AUC: ROC 곡선 아래 영역인 AUC를 측정함, 이는 클래스 1에 대한 예측 확률과 관련됨
- confusion matrix: 실제 클래스와 예측 클래스 간의 관계를 나타내는 행렬로, 다음과 같이 구성됨
    - TP true positive: 실제 클래스 1, 예측 클래스 1
    - FP false positive: 실제 클래스 0, 예측 클래스 1
    - FN false negative: 실제 클래스 1, 예측 클래스 0
    - TN true negative: 실제 클래스 0, 예측 클래스 0
- classification report: 각 클래스에 대한 precision, recall, f1 score을 포함하는 상세한 성능 평가 결과를 출력

hyper-parameter tuning

hyper-parameter은 모델이 학습하는 동안 자동으로 조정되지 않는 parameter로, 이 값을 조정하여 모델의 성능을 극대화할 수 있다.

1. 모델과 hyper-parameter 선택

   예를 들어 Logistic Regression의 경우, `c` 정규화 강도와 같은 hyper-parameter가 있다.

2. hyper-parameter 값의 범위 설정

   각 hyper-parameter에 대해 시도해볼 값의 범위를 설정한다. 예를 들어 `c` 정규화 강도의 경우, `0.001, 0.01, 0.1, 1, 10, 100`과 같은 값으로 설정할 수 있다.

3. tuning method 선택

   일반적으로 사용하는 방법은 다음과 같다.

   - Grid Search: 설정된 hyper-parameter 값의 모든 조합을 탐색한다.
   - Random Search: 설정된 범위 내에서 무작위로 hyper-parameter 값을 선택하여 탐색한다.
   - Bayesian Optimization: 이전 평가 결과를 바탕으로 다음으로 실행할 hyper-parameter를 선택한다.

4. 교차 검증을 통한 성능 평가

   이 과정에서 최적의 hyper-parameter를 찾는다.

5. 최적의 hyper-parameter 적용