Week2_예습과제_유연

2장 사이킷런으로 시작하는 머신러닝

01 사이킷런 소개와 특징

- 사이킷런: 파이썬 기반의 머신러닝을 위한 가장 쉽고 효율적인 개발 라이브러리 제공
- 사이킷런의 특징
    - 쉽고 가장 파이썬스러운 API 제공
    - 머신러닝을 위한 다양한 알고리즘 & 개발을 위한 편리한 프레임워크와 API 제공
    - 많은 환경에서 사용되며 검증됨
- 아나콘다 설치시, 별도 설치 필요 x

02 첫번째 머신러닝 만들어보기

- 분류: 지도학습
    - label 데이터로 모델을 학습한 뒤, 별도의 테스트 데이터 세트에서 미지의 레이블 예측
    - 명확한 정답이 주어진 데이터를 먼저 학습한 뒤미지의 정답을 예측하는 방식
    - 학습 데이터 세트, 테스트 데이터 세트
- 하이퍼 파라미터: 머신러닝 알고리즘별로 최적의 학습을 위해 직접 입력하는 파라미터 → 머신러닝 알고리즘의 성능을 튜닝
- 데이터 세트로 분류를 예측한 프로세스 정리

1. 데이터 세트 분리: 학습 데이터와 테스트 데이터로 분리
2. 모델 학습: 학습 데이터를 기반으로 ML 알고리즘을 적용해 모델 학습
3. 예측 수행: 학습된 ML 모델을 이용해 테스트 데이터의 분류 예측
4. 평가: 예측된 결괏값과 테스트 데이터의 실제 결괏값을 비교해 ML 모델 성능 평가

03 사이킷선의 기반 프레임 워크 익히기

Estimator 이해 및 fit(), predict() 메서드

- 분류 알고리즘 구현 클래스 Classifier
- 회귀 알고리즘 구현 클래스 Regressor

⇒ Estimator: 지도학습의 모든 알고리즘을 구현한 클래스

  - cross_val_score(), GridSearchCV.fit() 함수 내에서 이 Estimator의 fit()과 predict()을 호출하여 평가하거나 하이퍼 파리미터 튜닝을 수행함
  - fit(), predict()를 내부에서 구현
- 비지도 학습: 차원 축소, 클러스터링, 피처 추출
  - fit(): 입력 데이터의 형태에 맞춰 데이터를 변환하기 위한 사전 구조를 맞추는 작업
  - transform(): 입력 데이터의 차원 변환, 클러스터링, 피처 추출 등의 실제 작업 수행
  - fit_transform() 제공

사이킷런의 주요 모듈

- 예제 데이터
  - sklearn.datasets
- 피처 처리
  - sklearn.preprocessing
  - sklearn.feature_selection
  - sklearn.feature_extraction
- 피처 처리 & 차원 숙소
  - sklearn.decomposition
- 데이터 분리, 검증 & 파라미터 튜닝
  - sklearn.model_selection
- 평가
  - sklearn.metrics
- ML 알고리즘
  - sklearn.ensemble
  - sklearn.linear_model

    - sklearn.naive_bayes
    - sklearn.svm
    - sklearn.tree
    - sklearn.cluster
- 유틸리티
    - sklearn.pipeline

내장된 예제 데이터 세트

- 분류나 회귀 연습용 예제 데이터
    - datasets.load_boston(): 회귀) 보스턴의 집 피처들과 가격에 대한 데이터 세트
    - datasets.load_breast_cancer(): 분류) 유방암 피처들과 악성/음성 레이블 데이터 세트
    - datasets.load_diabetes(): 회귀) 당뇨 데이터 세트
    - datasets.load_digits(): 분류) 이미지 픽셀 데이터 세트
    - datasets.load_iris(): 분류) 붓꽃 데이터 세트
- fetch: 인터넷에서 내려받아 scikit_learn_data 서브 디렉터리에 저장
    - fetch_covtype()
    - fetch_20newsgroups()
    - fetch_olivetti_faces()
    - fetch_lfw_people()
    - fetch_lfw_paris()
    - fetch_rcv1()
    - fetch_mldata()
- 표본 데이터 생성기
    - datasets.make_classifications(): 분류를 위한 데이터 세트를 만듦
    - datasets.make_blobs(): 클러스터링을 위한 데이터 세트를 무작위로 생성
- 내장된 데이터 세트: 딕셔너리 형태
    - data: ndarray

- target: ndarray
- target_name: ndarray or list
- feature_name: ndarray or list
- DESCR: str

04 Model Selection 모듈 소개

교차 검증

- overfitting: 모델이 학습 데이터에 과도하게 최적화되어 다른 데이터로 예측을 수행할 때 성능이 과도하게 떨어지는 것
  - 데이터 편중을 막기 위해, 별도의 여러 세트로 구성된 학습 데이터 세트와 검증 데이터 세트에서 학습과 평가를 수행
  - 데이터 세트를 세분화해서 학습,검증,테스트 데이터 세트로 나눔
- K 폴드 교차 검증
  - k개의 데이터 폴드 세트를 만들어 k번만큼 각 폴드 세트에 학습과 검증 평가를 반복적으로 수행하는 방법
  - k개의 평가를 평균한 결과로 예측 성능 평가
- Stratified K 폴드
  - 불균형한 분포도를 가진 레이블 데이터 집합을 위한 방식
  - K 폴드가 레이블 데이터 집합이 원본 데이터 집합의 레이블 분포를 학습 및 테스트 세트에 제대로 분배하지 못하는 문제 해결
  - 원본 데이터의 레이블 분포를 먼저 고려한 뒤, 이 분포와 동일하게 학습과 검증 데이터 세트를 분배
  - 일반적으로 분류에서 사용
  - 회귀 지원 x → 회귀의 결정값은 이산값 형태의 레이블이 아닌 연속된 숫자값이기 때문
- cross_val_score(): 교차 검증의 과정을 한번에 수행
  - classifier 입력: stratified K 폴드 방식
  - regressor 입력: K 폴드 방식
- cross_validate(): 여러개의 평가 지표 반환

GridSearchCV - 교차 검증과 최적 하이퍼 파라미터 튜닝을 한 번에

- 하이퍼 파라미터: 머신러닝 알고리즘을 구성하는 주요 구성 요소 → 파라미터 값을 조정하여 성능 개선
- Grid: 촘촘하게 파라미터를 입력하면서 테스트
- cross-validation을 위한 학습/테스트 세트로 자동으로 분할
- 그리드에 기술된 모든 파라미터를 순차적으로 적용
- 다양하게 테스트 but 시간이 오래 걸림
- CV k 회, n개 파라미터 조합: k*n 회의 학습/평가

05 데이터 전처리

- 결손값: 허용되지 않으므로 다른 값으로 변환해야함
  - 결손값이 적을 경우: 평균값 등으로 대체
  - 결손값이 많을 경우: drop
  - 결손값이 일정 수준 이상 되는 경우: 정밀하게 대체 값 선정
- 문자열: 인코딩하여 숫자형으로 반환
  - 텍스트형 피처: 피처 벡터화 or drop

데이터 인코딩

- 레이블 인코딩: 카테고리 피처를 코드형 숫자 값으로 변환
  - 가중치를 무시하고 인코딩한다면 성능이 떨어짐
  - 선형 회귀와 같은 ML 알고리즘에 적용 x
  - 트리 계열의 ML 알고리즘은 적용 o
- 원 핫 인코딩: 피처 값의 유형에 따라 새로운 피처를 추가해 고유 값에 해당하는 칼럼에만 1을 표시
  - 입력값으로 2차원 데이터가 필요함
  - OneHotEncoder로 변환한 값이 희소 행렬이므로 toarray() 메서드로 밀집 행렬으로 다시 변환해야함

피처 스케일링과 정규화

- 피처 스케일링: 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업

- 표준화: 데이터의 피처 평균이 0이고 분산이 1인 가우시안 정규 분포를 가진 값으로 변환하는 것
- 정규화: 서로 다른 피처의 크기를 통일하기 위해 크기를 변환 → 개별 데이터의 크기를 모두 똑같은 단위로 변경
- 사이킷런 normalizer: 개별 벡터를 모든 피처 벡터의 크기로 나눠줌

StandardScaler

- 개별 피처를 평균이 0이고 분산이 1인 값으로 대체

MinMaxScaler

- 데이터 값을 0과 1사이의 범위 값으로 변환

학습 데이터와 테스트 데이터의 스케일링 변환 시 유의점

- scaler 객체를 사용할 경우: 테스트 데이터 세트로는 다시 fit()을 수행하지 않고 학습 데이터 세트로 fit()을 수행한 결과를 이용해 transform() 변환을 적용
- 학습 데이터로 fit()이 적용된 스케일링 기준 정보를 그대로 테스트 데이터에 적용
- 전체 데이터의 스케일링 변환을 적용한 뒤, 학습과 테스트로 분리
- 테스트 데이터 변환 시에는 fit() or fit_transform()을 적용하지 않고 학습 데이터로 이미 fit()된 scaler 객체를 이용해 transform()으로 변환

3장 평가

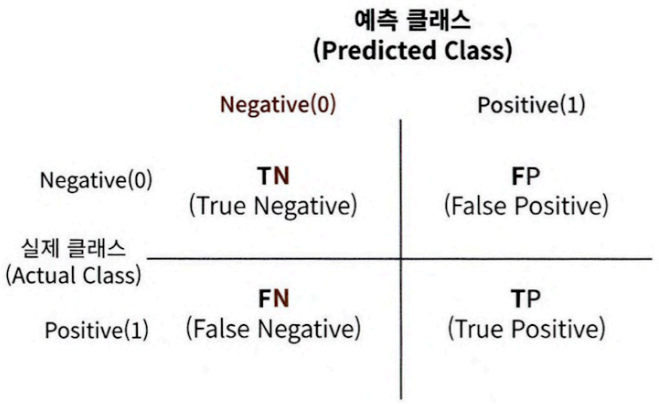
- 머신러닝 프로세스: 데이터 가공/변환, 모델 학습/예측, 평가
- 성능 평가 지표 - 회귀: 실제값과 예측값의 오차 평균값에 기반
- 성능 평가 지표 - 분류
  - 정확도
  - 오차행렬
  - 정밀도
  - 재현율
  - F1 스코어
  - ROC AUC

01 정확도

- 정확도: 모델 예측 성능을 나타내는 평가 지표
  - 이진 분류의 경우: ML 모델의 성능을 왜곡할 수 있음
  - 불균형한 레이블 값 분포에서 적합한 평가 지표가 아님

02 오차 행렬

- 오차행렬: 학습된 분류 모델이 예측을 수행하면서 얼마나 counfused 되는지 보여주는 지표 → 예측 오류가 얼마인지 + 어떤 유형의 오류가 발생하는지
  - TN: 예측값을 negative 값 0으로 예측했고 실제 값 역시 negative 값 0
  - FP: 예측값을 positive 값 1으로 예측했는데 실제 값은 negative 값 0
  - FN: 예측값을 negative 값 0으로 예측했는데 실제 값은 positive 값 1
  - TP: 예측값을 positive 값 1으로 예측했는데 실제 값 역시 positive 값 1



- 정확도: (TN+TP) / (TN+FP+FN+TP)
- 불균형 이진 분류 데이터 세트: positive 데이터 건수가 적으므로, negative로 예측 정확도가 높아지는 경향 발생

03 정밀도와 재현율

- 정밀도 = TP / (FP+TP) : 예측을 positive로 한 대상 중 예측과 실제 값이 positive로 일치한 데이터의 비율
  - FN을 낮추는데 초점
  - 정밀도가 중요한 경우: 실제 negative 데이터를 positive로 잘못 판단하게 되면, 업무상 큰 영향이 발생하는 경우

- 재현율 = TP / (FN+TP) : 실제 값이 positive인 대상 중에 예측과 실제 값이 positive로 일치한 데이터의 비율 ⇒ 민감율 = TPR
    - FP를 낮추는데 초점
    - 재현율이 중요한 경우: 실제 positive 데이터를 negative로 잘못 판단하게 되면, 업무상 큰 영향이 발생하는 경우

정밀도/재현율 트레이드오프

- 정밀도 또는 재현율이 특별히 강조돼야 할 경우: 분류의 결정 임곗값을 조정해 정밀도 또는 재현율의 수치를 높일 수 있음
- Trade - off : 정밀도와 재현율은 상호 보완적인 평가 지표
- 임곗값을 낮추면, 재현율이 올라가고 정밀도가 떨어지는 이유?
    - 분류 결정 임곗값은 positive 예측값을 결정하는 확률의 기준
    - 임곗값을 낮출수록 True 값이 많아짐
    - 재현율 값이 높아짐 - 양성 예측을 많이 하기 때문에, 실제 양성을 음성으로 예측하는 횟수가 줄어듦

정밀도와 재현율의 맹점

- 정밀도가 100% 되는 법: 확실한 기준이 되는 경우만 양성으로 예측하고 나머지는 음성으로 예측
    - 확실한 양성 환자가 1명이라면, TP = 1 , FP = 0 → 1/(1+0)=100%
- 재현율이 100% 되는법: 모든 데이터를 양성으로 예측
    - 실제 양성이 30명이라면, TP=30, FN=0 → 30/(30+0)=100%

04 F1 스코어

- F1 score: 정밀도와 재현율을 결합한 지표

$$F1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 * \frac{precision * recall}{precision + recall}$$

05 ROC 곡선과 AUC

- ROC 곡선: 수신자 판단 곡선
  - FPR이 변할 때 TPR(재현율)이 어떻게 변하는지 나타내는 곡선
  - 민감도(TPR): 실제값 양성이 정확히 예측돼야 하는 수준 - y축
  - 특이성(TNR): 실제값 음성이 정확히 예측돼야 하는 수준
    - TNR=TN/(FP+TN)
    - FPR=FP/(FP+TN) - x축
  - 곡선이 직선에 가까울수록 성능이 떨어지는 것
- AUC: ROC 곡선 밑의 면적
  - 1에 가까울수록 좋은 수치
  - 수치가 커지려면? fpr이 작은 상태에서 얼마나 큰 tpr을 얻을 수 있느냐