[파머완] 02 사이킷런으로 시작하는 머신러닝

진규빈 · 약 12시간 전

(1) 사이킷런 소개와 특징

사이킷런(scikit-learn)

- 가장 많이 사용되는 파이썬 머신러닝 라이브러리
- 파이썬 기반의 다른 머신러닝 패키지도 사이킷런 스타일의 API를 지향할 정도로 쉽고 가장 파이썬스러운 API 제공
- 머신러닝을 위한 다양한 알고리즘, 개발을 위한 편리한 프레임워크와 API 제공
- 오랜 기간 실전 환경에서 검증됐으며, 매우 많은 환경에서 사용되는 성숙한 라이브러리
- Anaconda 설치 시 기본적으로 사이킷런까지 설치 완료 ~

(2) 첫 번째 머신러닝 만들어 보기 - 붓꽃 품종 예측하기

> <붓꽃 데이터 세트로 붓꽃의 품종 분류(Classification)하기>
> 붓꽃 데이터 세트는 꽃잎의 길이와 너비, 꽃받침의 길이와 너비 피처(Feature)를 기반으로

지도학습(Supervised Learning)

- 학습을 위한 다양한 피처와 분류 결정값인 레이블(Label) 데이터로 모델을 학습한 뒤, 별도의 테스트 데이터 세트에서 미지의 레이블 예측
- 즉, 명확한 정답이 주어진 데이터를 먼저 학습한 뒤 미지의 정답을 예측하는 방식
- 이때 학습을 위해 주어진 데이터 세트를 **학습 데이터 세트**, 머신러닝 모델의 예측 성능을 평가하기 위해 별도로 주어진 데이터 세트를 **테스트 데이터 세트**로 지칭
- 분류(Classification) & 회귀(Regression) => 대표적인 지도학습 방법이자 주요 두 축

> 새로운 주피터 노트북 생성 후 사이킷런에서 사용할 모듈 임포트

- 사이킷런 패키지 내의 모듈명은 `sklearn` 으로 시작하는 명명규칙이 있음
- `sklearn.datasets` : 사이킷런에서 자체적으로 제공하는 데이터 세트를 생성하는 모듈의 모임
- `sklearn.tree` : 트리 기반 ML 알고리즘을 구현한 클래스의 모임
- `sklearn.model_selection` : 학습 데이터와 검증 데이터, 예측 데이터로 데이터를 분리하거나 최적의 하이퍼 파라미터로 평가하기 위한 다양한 모듈의 모임
- **하이퍼 파라미터**: 머신러닝 알고리즘별로 최적의 학습을 위해 직접 입력하는 파라미터들을 통칭! 이를 통해 머신러닝 알고리즘의 성능 튜닝 가능

> 붓꽃 데이터 세트 생성에는 `load_iris()` 이용
> `load_iris()` 함수로 붓꽃 데이터 세트 로딩 후, 피처들과 데이터 값이 어떻게 구성돼 있는지 확인 위해 DataFrame으로 변환

> **데이터 세트 분리** - 데이터를 학습 데이터와 테스트 데이터로 분리

- 학습용 데이터와 테스트용 데이터는 반드시 분리해야 함 => 학습 데이터로 학습된 모델이 얼마나 뛰어난 성능 가지는지 평가하려면 테스트 데이터 세트가 필요하기 때문
- 사이킷런의 `train_test_split()` API 이용
  - 학습 데이터와 테스트 데이터를 `test_size` 파라미터 입력값의 비율로 쉽게 분할 가능
  - 파라미터1 `iris_data` : 피처 데이터 세트

ex) `test_size=0.2` 로 입력 파라미터 설정 시 전체 데이터 중 테스트 데이터 20%, 학습 데이터 80%로 데이터 분할
- 파라미터4 `random_state` : 호출할 때마다 같은 학습/테스트 용 데이터 세트 생성 위해 주어지는 난수 발생 값

> **모델 학습** - 학습 데이터 기반으로 ML 알고리즘 적용해 모델 학습시킴

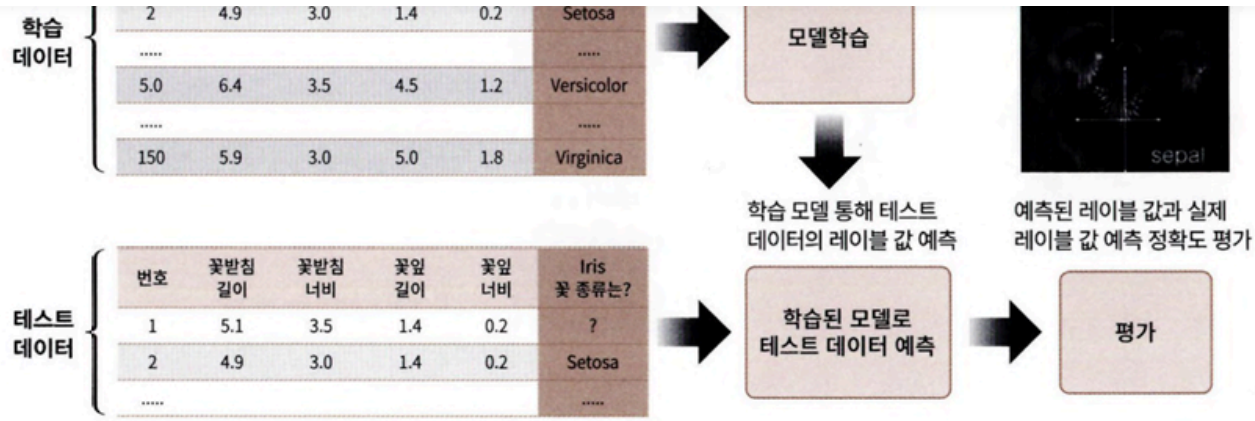
- 사이킷런의 의사 결정 트리 클래스인 `DecisionTreeClassifier` 를 객체로 생성
- 생성된 객체의 `fit()` 메서드에 학습용 피처 데이터 속성과 결정값 데이터 세트를 입력해 호출하면 학습 수행

> **예측 수행** - 학습된 ML 모델 이용해 테스트 데이터의 분류(붓꽃 종류) 예측

- 학습된 객체로 예측 수행 (예측은 반드시 학습 데이터 아닌 다른 데이터 이용, 일반적으로 테스트 데이터 세트 이용)
- `DecisionTreeClassifier` 객체의 `predict()` 메서드에 테스트용 피처 데이터 세트 입력해 호출 => 학습된 모델 기반에서 테스트 데이터 세트에 대한 예측값 반환

> **평가** - 예측된 결과값과 테스트 데이터의 실제 결과값 비교해 ML 모델 성능 평가

- ML 모델의 성능 평가 방법은 여러가지 있으며, 그중 정확도는 예측 결과가 실제 레이블 값과 얼마나 정확하게 맞는지 평가하는 지표
- 사이킷런은 정확도 측정 위해 `accuracy_score()` 함수 제공
- 파라미터1로 실제 레이블 데이터 세트, 파라미터2로 예측 레이블 데이터 세트 입력

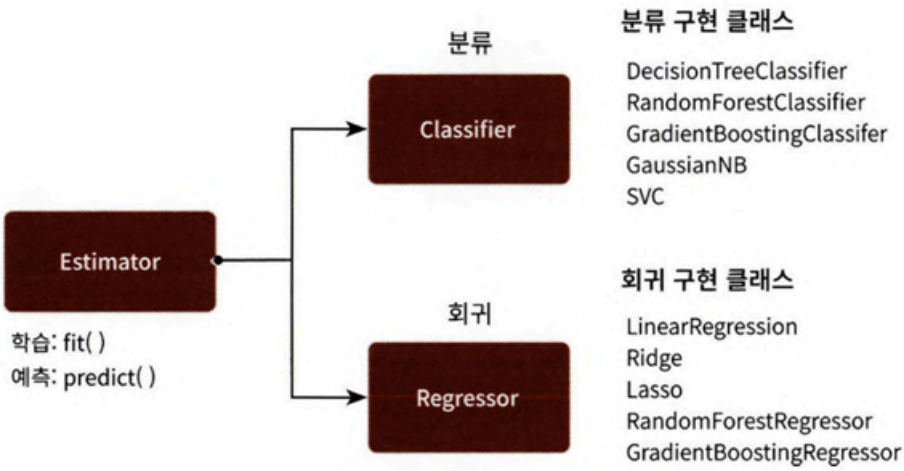


〈붓꽃 데이터 세트 기반의 ML 분류 예측 수행 프로세스〉

(3) 사이킷런의 기반 프레임워크 익히기

Estimator

- 지도학습의 모든 알고리즘을 구현한 클래스를 통칭하는 말
- Classifier(분류 알고리즘 구현한 클래스) + Regressor(회귀 알고리즘 구현한 클래스)
- ML 모델 학습을 위한 `fit()` 과 학습된 모델의 예측을 위한 `predict()` 메서드 내부에서 구현
- `cross_val_score()` 와 같은 evaluation 함수, `GridSearchCV` 와 같은 하이퍼 파라미터 튜닝 지원하는 클래스의 경우 이 Estimator 인자로 받음



- 비지도학습&피처 추출에서 `fit()` => 입력 데이터의 형태에 맞춰 데이터 변환하기 위한 사전 구조 맞추는 작업 (지도학습에서의 학습과 다른 의미)
- 이후 입력 데이터에 대한 실제 작업은 `transform()` 으로 수행
- 사이킷런은 둘을 하나로 결합한 `fit_transform()` 도 함께 제공 => 별도 호출할 필요를 줄여주지만 사용에 약간의 주의 필요

사이킷런의 주요 모듈

분류	모듈명	설명
예제 데이터	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이터 세트
피처 처리	sklearn.preprocessing	데이터 전처리에 필요한 다양한 가공 기능 제공(문자열을 숫자형 코드 값으로 인코딩, 정규화, 스케일링 등)
	sklearn.feature_selection	알고리즘에 큰 영향을 미치는 피처를 우선순위대로 셀렉션 작업을 수행하는 다양한 기능 제공
피처 처리	sklearn.feature_extraction	텍스트 데이터나 이미지 데이터의 벡터화된 피처를 추출하는 데 사용됨. 예를 들어 텍스트 데이터에서 Count Vectorizer나 Tf-Idf Vectorizer 등을 생성하는 기능 제공. 텍스트 데이터의 피처 추출은 sklearn.feature_extraction.text 모듈에, 이미지 데이터의 피처 추출은 sklearn.feature_extraction.image 모듈에 지원 API가 있음.
피처 처리 & 차원 축소	sklearn.decomposition	차원 축소와 관련한 알고리즘을 지원하는 모듈임. PCA, NMF, Truncated SVD 등을 통해 차원 축소 기능을 수행할 수 있음
데이터 분리, 검증 & 파라미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리, 그리드 서치(Grid Search)로 최적 파라미터 추출 등의 API 제공
평가	sklearn.metrics	분류, 회귀, 클러스터링, 페어와이즈(Pairwise)에 대한 다양한 성능 측정 방법 제공 Accuracy, Precision, Recall, ROC-AUC, RMSE 등 제공

ML 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등을 제공
	sklearn.linear_model	주로 선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘을 지원. 또한 SGD(Stochastic Gradient Descent) 관련 알고리즘도 제공
	sklearn.naive_bayes	나이브 베이즈 알고리즘 제공. 가우시안 NB, 다항 분포 NB 등.
	sklearn.neighbors	최근접 이웃 알고리즘 제공. K-NN 등
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사 결정 트리 알고리즘 제공
	sklearn.cluster	비지도 클러스터링 알고리즘 제공 (K-평균, 계층형, DBSCAN 등)
유틸리티	sklearn.pipeline	피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공

내장된 예제 데이터 세트

- 분류나 회귀 연습용 예제 데이터

API 명	설명
datasets.load_boston()	회귀 용도이며, 미국 보스턴의 집 피처들과 가격에 대한 데이터 세트
datasets.load_breast_cancer()	분류 용도이며, 위스콘신 유방암 피처들과 악성/음성 레이블 데이터 세트
datasets.load_diabetes()	회귀 용도이며, 당뇨 데이터 세트
datasets.load_digits()	분류 용도이며, 0에서 9까지 숫자의 이미지 픽셀 데이터 세트
datasets.load_iris()	분류 용도이며, 붓꽃에 대한 피처를 가진 데이터 세트

- fetch 계열 명령은 데이터 크기 커서 패키지에 처음부터 저장되어 있지 않고 인터넷에서 내려받아 홈 디렉터리 아래 scikit_learn_data 서브 디렉터리에 저장 후 불러들이는 데이터 (최초 사용 시 인터넷 연결돼 있어야 함)
  - `fetch_covtype()` : 회귀 분석용 토지 조사 자료
  - `fetch_20newsgroups()` : 뉴스 그룹 텍스트 자료
  - `fetch_olivetti_faces()` : 얼굴 이미지 자료
  - `fetch_lfw_people()` : 얼굴 이미지 자료
  - `fetch_lfw_pairs()` : 얼굴 이미지 자료

- 분류와 클러스터링 위한 표본 데이터 생성기

API 명	설명
datasets.make_classifications( )	분류를 위한 데이터 세트를 만듭니다. 특히 높은 상관도, 불필요한 속성 등의 노이즈 효과를 위한 데이터를 무작위로 생성해 줍니다.
datasets.make_blobs( )	클러스터링을 위한 데이터 세트를 무작위로 생성해 줍니다. 군집 지정 개수에 따라 여러 가지 클러스터링을 위한 데이터 세트를 쉽게 만들어 줍니다.

- 사이킷런에 내장된 분류/회귀 위한 연습용 예제 데이터 세트는 일반적으로 딕셔너리 형태
- 키는 보통 `data` , `target` , `target_names` , `feature_names` , `DESCR` 로 구성
  - `data` : 피처의 데이터 세트 / 넘파이 배열 타입
  - `target` : 분류 시 레이블 값, 회귀 시 숫자 결과값 데이터 세트 / 넘파이 배열 타입
  - `target_names` : 개별 레이블의 이름 / 넘파이 배열 또는 파이썬 리스트 타입
  - `feature_names` : 피처의 이름 / 넘파이 배열 또는 파이썬 리스트 타입
  - `DESCR` : 데이터 세트에 대한 설명과 각 피처의 설명 / 스트링 타입
- 피처의 데이터 값 반환받기 위해서는 내장 데이터 세트 API 호출 후 그 Key 값 지정하면 됨
- 불꽃 데이터 예시 => p.95 ~ 97 & 코드 필사본 참고

(4) Model Selection 모듈 소개

> 사이킷런의 model_selection 모듈
> => 학습 데이터와 테스트 데이터 세트 분리하거나 교차 검증 분할 및 평가, 그리고 Estimator의 하이퍼 파라미터를 튜닝하기 위한 다양한 함수와 클래스 제공

학습/테스트 데이터 세트 분리 - `train_test_split()`

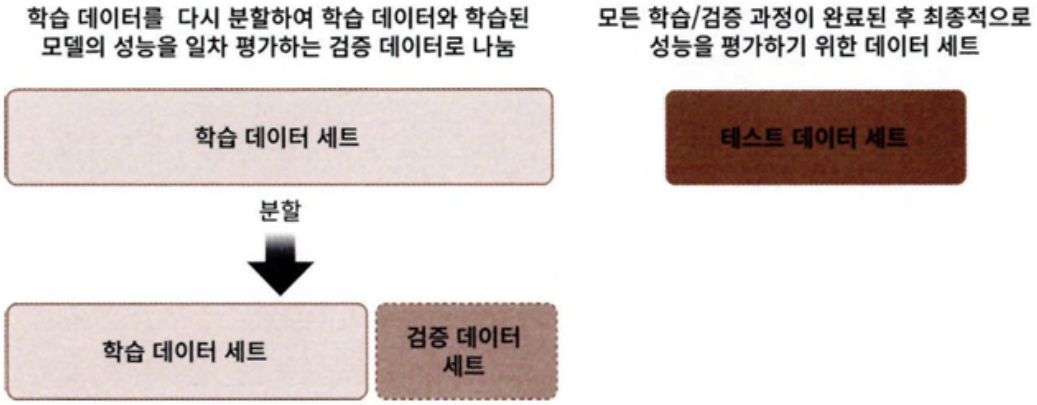
- 테스트 데이터 세트 이용하지 않고 학습 데이터 세트로만 학습 및 예측할 경우, 즉 학습과 예측을 동일한 데이터 세트로 수행할 경우 예측 정확도가 100%이 되는 문제 발생 (이미 학습한 데이터 세트를 기반으로 예측했기 때문)
- 따라서 예측을 수행하는 데이터 세트는 학습을 수행한 학습용 데이터 세트가 아닌 전용의 테스트 데이터 세트여야 함

- `train_test_split()` 의 선택적으로 입력받는 파라미터
  - `test_size` : 전체 데이터에서 테스트 데이터 세트 크기를 얼마로 샘플링할 것인지 결정 / 디폴트 0.25
  - `train_size` : 전체 데이터에서 학습용 데이터 세트 크기를 얼마로 샘플링할 것인지 결정 (잘 사용되지 않음)
  - `shuffle` : 데이터 분리 전에 데이터 미리 섞을지 결정 / 디폴트 True / 데이터 분산시켜 좀 더 효율적인 학습 및 테스트 데이터 세트 만드는 데 사용
  - `random_state` : 호출할 때마다 동일한 학습/테스트용 데이터 세트 생성 위해 주어지는 난수 값
- `train_test_split()` 의 반환값은 튜플 형태 / 순차적으로 학습용 데이터의 피처 데이터 세트, 테스트용 데이터의 피처 데이터 세트, 학습용 데이터의 레이블 데이터 세트, 테스트용 데이터의 레이블 데이터 세트 반환
- 학습을 위한 데이터의 양을 일정 수준 이상으로 보장하는 것도 중요하지만, 학습된 모델에 대해 다양한 데이터를 기반으로 예측 성능을 평가해보는 것도 매우 중요!

> **과적합(Overfitting)**: 모델이 학습 데이터에만 과도하게 최적화되어, 실제 예측을 다른 데이터로 수행할 경우 예측 성능이 과도하게 떨어지는 것

교차 검증

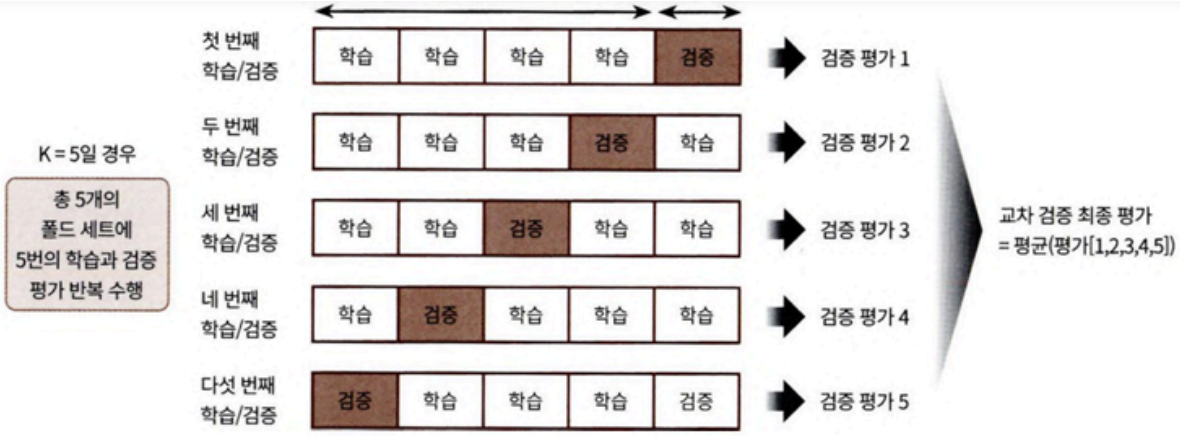
- 고정된 학습 데이터와 테스트 데이터로 평가 하다 보면 테스트 데이터에만 최적의 성능을 발휘할 수 있도록 편향되게 모델을 유도하는 경향 생김
- 해당 데이터에만 과적합되는 학습 모델이 만들어져 다른 테스트용 데이터가 들어올 경우 성능 저하됨
- 이러한 데이터 편중을 막기 위해 별도의 여러 세트로 구성된 학습 데이터 세트와 검증 데이터 세트에서 학습과 평가 수행하는 것이 교차 검증
- 각 세트에서 수행한 평가 결과에 따라 하이퍼 파라미터 튜닝 등의 모델 최적화를 더 손쉽게 할 수 있음
- 대부분 ML 모델의 성능 평가는 교차 검증 기반으로 1차 평가 한 뒤 최종적으로 테스트 데이터 세트에 적용해 평가하는 프로세스
- ML에 사용되는 데이터 세트를 세분화해 학습 / 검증 / 테스트 데이터 세트로 나눌 수 있음



K 폴드 교차 검증

- 먼저 K개의 데이터 폴드 세트를 만들어 K번만큼 각 폴트 세트에 학습과 검증 평가를 반복적으로 수행하는 방법으로, 가장 보편적으로 사용됨

- 5 폴드 교차 검즈이 5개의 폴드된 데이터 세트를 학습과 검증을 위한 데이터 세트로 변경하면서 5번 평가 수행한 뒤, 이 5개의 평가를 평균한 결과 가지고 예측 성능 평가
  - 먼저 데이터 세트를 5등분
  - 첫 번째 반복 => 처음부터 4개 등분을 학습 데이터 세트, 마지막 5번째 등분 하나를 검증 데이터 세트로 설정
  - 학습 데이터 세트에서 학습 수행, 검증 데이터 세트에서 평가 수행
  - 첫 번째 평가 수행하고 나면 두 번째 반복에서 다시 비슷한 학습과 평가 작업 수행
  - 단, 이번에는 학습 데이터와 검증 데이터를 변경 (처음부터 3개 등분까지, 마지막 5번째 등분을 학습으로, 4번째 등분 하나를 검증 데이터 세트로 설정)
  - 이렇게 학습 데이터 세트와 검증 데이터 세트를 점진적으로 변경하며 마지막 5번째까지 학습과 검증 수행
  - 5개의 예측 평가 구했으면 이를 평균해 K 폴드 평가 결과로 반영



- 사이킷런에서는 K 폴드 교차 검증 프로세스를 구현하기 위해 `KFold` 와 `StratifiedKFold` 클래스 제공

- 불꽃 데이터 예시 => p.102 ~ 104 & 코드 필사본 참고

Stratified K 폴드

- 불균형한(imbalanced) 분포도를 가진, 즉 특정 레이블 값이 특이하게 많거나 매우 적어 값의 분포가 한쪽으로 치우치는 레이블(결정 클래스) 데이터 집합을 위한 K 폴드 방식
- K 폴드가 레이블 데이터 집합이 원본 데이터 집합의 레이블 분포를 학습 및 테스트 세트에 제대로 분배하지 못하는 경우, KFold로 분할된 레이블 데이터 시트가 전체 레이블 값의 분포도를 반영하지 못하는 경우의 문제 해결
- 이를 위해 원본 데이터의 레이블 분포를 먼저 고려한 뒤 이 분포와 동일하게 학습과 검증 데이터 세트를 분배
- StratifiedKFold 사용법은 KFold 사용법과 거의 유사
- 하지만 StratifiedKFold는 레이블 데이터 분포도에 따라 학습/검증 데이터를 나누기 때문에 `split()` 메서드에 인자로 피처 데이터 세트뿐만 아니라 레이블 데이터 세트도 반드시 필요 (K 폴드의 경우 레이블 데이터 세트를 `split()` 메서드의 인자로 입력하지 않아도 무방)
- Stratified K 폴드의 경우 원본 데이터의 레이블 분포도 특성을 반영한 학습 및 검증 데이터 세트를 만들 수 있으므로 왜곡된 레이블 데이터 세트에서는 반드시 Stratified K 폴드 이용해 교차 검증 해야 함
- 일반적으로 분류에서의 교차 검증은 K 폴드가 아니라 Stratified K 폴드로 분할돼야 함
- 회귀에서는 Stratified K 폴드 지원 X => 회귀의 결정값은 이산값 형태의 레이블이 아니라 연속된 숫자값이기 때문에 결정값별로 분포 정하는 의미 없기 때문

교차 검증을 보다 간편하게 제공해주는 사이킷런 API

수행하고 예측 성능 반환)을 한꺼번에 수행

```
cross_val_score(estimator, X, y=None, scoring=None, cv=None, n_jobs=1, verbose=0, fit_params=N
```

- 주요 파라미터
  - `estimator` : 사이킷런의 분류 알고리즘 클래스인 Classifier / 회귀 알고리즘 클래스인 Regressor
  - `X` : 피처 데이터 세트
  - `y` : 레이블 데이터 세트
  - `scoring` : 예측 성능 평가 지표
  - `cv` : 교차 검증 폴드 수
- cv로 지정된 횟수만큼 scoring 파라미터로 지정된 성능 지표 측정값을 배열 형태로 반환 => 일반적으로 이를 평균해 평가 수치로 사용
- classifier가 입력되면 Stratified K 폴드 방식으로 레이블값의 분류에 따라 학습/테스트 세트를 분할
- 내부에서 Estimator를 학습(fit), 예측(predict), 평가(evaluation)시켜주므로 간단하게 교차 검증 수행 가능
- 비슷한 API로 `cross_validate()` : 여러 개의 평가 지표 반환 가능, 학습 데이터에 대한 성능 평가 지표와 수행 시간도 같이 제공

GridSearchCV - 교차 검증과 최적 하이퍼 파라미터 튜닝을 한번에

- Classifier나 Regressor와 같은 알고리즘에 사용되는 하이퍼 파라미터를 순차적으로 입력하면서 편리하게 최적의 파라미터를 도출할 수 있는 방안 제공
- for 루프로 모든 파라미터를 번갈아 입력하며 학습시키는 방법을 좀 더 유연하게 API 레벨에서 제공한 것
- 교차 검증 기반으로 하이퍼 파라미터의 최적 값 찾게 해줌, 즉 데이터 세트를 cross-validation 위한 학습/테스트 세트로 자동 분할한 뒤 하이퍼 파라미터 그리드에 기술된 모든 파라미터를 순차 적용해 최적 파라미터 찾을 수 있게 해줌
- 사용자가 튜닝하고자 하는 여러 종류의 하이퍼 파라미터를 다양하게 테스트하면서 최적의 파라미터를 편리하게 찾게 해주지만 동시에 순차적으로 파라미터를 테스트하므로 수행시간이 상대적으로 오래 걸리는 것에 유념
- 주요 파라미터
  - `estimator` : classifier, regressor, pipeline이 사용
  - `param_grid` : key+리스트 값 가지는 딕셔너리가 주어짐 / estimator의 튜닝 위해 파라미터명과 사용될 여러 파라미터값 지정

- `cv` : 교차 검증 위해 분할되는 학습 및 테스트 세트의 개수 지정
- `refit` : 디폴트가 True / True로 생성 시 가장 최적의 하이퍼 파라미터 찾은 뒤 입력된 estimator 객체를 해당 하이퍼 파라미터로 재학습

- 동작 및 성능 예시 => p.113 ~ 115 & 코드 필사본 참고
- 학습 데이터를 `GridSearchCV` 이용해 최적 하이퍼 파라미터 튜닝 수행한 뒤 별도의 테스트 세트에서 평가하는 것이 일반적인 머신러닝 모델 적용 방법

(5) 데이터 전처리

> ML 알고리즘은 데이터에 기반하고 있으므로 어떤 데이터를 입력으로 갖느냐에 따라 결과도 크게 달라질 수 있음 => 데이터 전처리는 ML 알고리즘만큼 중요!

데이터에 대해 미리 처리해야 할 기본 사항

- 결손값(NaN, Null) 값 허용 X => Null 값은 고정된 다른 값으로 변화해야 함
  - 피처 값 중 Null 값이 얼마 되지 않는다면 피처의 평균값 등으로 간단히 대체 가능 but Null 값이 대부분이라면 오히려 해당 피처는 드롭하는 것이 더 좋음
  - 중요도 높은 피처이고 Null을 단순히 피처의 평균값으로 대체할 경우 예측 왜곡이 심할 수 있다면 업무 로직 등을 상세히 검토해 더 정밀한 대체 값을 선정해야 함
- 사이킷런의 ML 알고리즘은 입력값으로 문자열 값 허용 X => 모든 문자열 값은 인코딩돼서 숫자 형으로 변환해야 함
  - 텍스트형 피처는 피처 벡터화 등의 기법으로 벡터화하거나 불필요한 피처라고 판단되면 삭제하는 게 좋음 => 예측에 중요한 요소 될 수 없으며 알고리즘 오히려 복잡하게 만들고 예측 성능 떨어뜨리기 때문

데이터 인코딩

> **레이블 인코딩**
> : 카테고리 피처를 코드형 숫자 값으로 변환하는 방법

- `LabelEncoder` 를 객체로 생성한 후 `fit()` 과 `transform()` 을 호출해 수행
- 문자열 값이 어떤 숫자 값으로 인코딩 됐는지 직관적으로 알 수 없는 경우 `classes_` 속성값으로 확인

- 레이블 인코딩이 일괄적 숫자 값으로 변환 되면서 몇 ML 알고리즘에는 이를 적용할 경우 예측 성능이 떨어지는 경우 발생 => 숫자 값의 경우 크고 작음에 대한 특성 작용하기 때문
- 따라서 레이블 인코딩은 선형 회귀와 같은 ML 알고리즘에는 적용 X

원본 데이터

상품 분류	가격
TV	1,000,000
냉장고	1,500,000
전자레인지	200,000
컴퓨터	800,000
선풍기	100,000
선풍기	100,000
믹서	50,000
믹서	50,000

상품 분류를 레이블 인코딩한 데이터

상품 분류	가격
0	1,000,000
1	1,500,000
4	200,000
5	800,000
3	100,000
3	100,000
2	50,000
2	50,000

> **원-핫 인코딩**
> 레이블 인코딩의 문제점 해결하기 위한 방식으로, 피처 값의 유형에 따라 새로운 피처를 추가해 고유 값에 해당하는 칼럼에만 1 표시하고 나머지 칼럼에는 0 표시하는 방법
> 즉, 행 형태로 되어 있는 피처의 고유값을 열 형태로 차원 변환한 뒤, 고유값에 해당하는 칼럼에만 1표시하고 나머지 칼럼에는 0 표시

- 사이킷런에서 `OneHotEncoder` 클래스로 변환 가능
- 단, 입력값으로 2차원 데이터 필요하고, `OneHotEncoder` 이용해 변환한 값이 희소 행렬(Sparse Matrix) 형태이므로 이를 다시 `toarray()` 메서드 이용해 밀집 행렬(Dense Matrix)로 변환해야 함
- 판다스 `get_dummies()` API 이용 시 문자열 카테고리 값 숫자 형으로 변환할 필요 없이 바로 변환 가능

원본 데이터

상품 분류	가격
TV	1,000,000
냉장고	1,500,000
전자레인지	200,000
컴퓨터	800,000
선풍기	100,000
선풍기	100,000
믹서	50,000
믹서	50,000

숫자로 인코딩

상품 분류	가격
0	1,000,000
1	1,500,000
4	200,000
5	800,000
3	100,000
3	100,000
2	50,000
2	50,000

원-핫 인코딩

TV	냉장고	믹서	선풍기	전자레인지	컴퓨터	가격
1	0	0	0	0	0	1,000,000
0	1	0	0	0	0	1,500,000
0	0	0	0	1	0	200,000
0	0	0	0	0	1	800,000
0	0	0	1	0	0	100,000
0	0	0	1	0	0	100,000
0	0	1	0	0	0	50,000
0	0	1	0	0	0	50,000

> **피처 스케일링(feature scaling)**
> : 서로 다른 변수의 값 변위를 일정한 수준으로 맞추는 작업

<표준화(Standardization)>

- 데이터의 피처 각각이 평균이 0이고 분산이 1인 가우시안 정규 분포 가진 값으로 변환하는 것
- 표준화 통해 변환될 피처 x의 새로운 i번째 데이터를 = 원래 값에서 피처 x의 평균 뺀 값을 피처 x의 표준편차로 나눈 값

$$x_i\_new = \frac{x_i - mean(x)}{stdev(x)}$$

<정규화(Normalization)>

- 일반적으로 서로 다른 피처의 크기를 통일하기 위해 크기를 변환해주는 것, 즉 개별 데이터의 크기를 모두 똑같은 단위로 변경하는 것
- 새로운 데이터 = 원래 값에서 피처 x의 최솟값을 뺀 값을 피처 x의 최댓값과 최솟값의 차이로 나눈 값

$$x_i\_new = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- 사이킷런 전처리에서 제공하는 Normalizer 모듈 => 선형대수에서의 정규화 개념 적용됐으며 개별 벡터의 크기 맞추기 위해 변환하는 것
- 개별 벡터를 모든 피처 벡터의 크기로 나누어 줌
- 새로운 데이터 = 원래 값에서 세 개의 피처의 i번째 피처 값에 해당하는 크기를 합한 값으로 나눈 값

$$x_i\_new = \frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$$

<사이킷런에서 제공하는 대표 피처 스케일링 클래스>

- `StandardScaler`
  - 표준화 쉽게 지원

오늘

- `MinMaxScaler`
  - 데이터값을 0과 1 사이의 범위 값으로 변환 (음수 값 있으면 -1에서 1값)
  - 데이터의 분포가 가우시안 분포 아닐 경우 Min, Max Scale 적용

- 유의점
  - 학습 데이터와 테스트 데이터의 `fit()`, `transform()`, `fit_transform()` 이용해 스케일링 변환 시
  - 가능하다면 전체 데이터의 스케일링 변환을 적용한 뒤 학습과 테스트 데이터로 분리
  - 여의치 않다면 테스트 변환 시에는 `fit()` 이나 `fit_transform()` 적용하지 않고 학습 데이터로 이미 `fit()` 된 Scaler 객체 이용해 `transform()` 으로 변환

진규빈

다음 포스트
[파머완] 03 평가

이전 포스트
[파머완] 01 파이썬 기반의 머신러닝과 생태계 이해

0개의 댓글

댓글을 작성하세요

댓글 작성